E-Motion
A Library for Motor Control

Matias Guijarro, Cyril Guilloud, Manuel Perez - BCU - ESRF

The European Synchrotron

FORK ME ON GITHUB
http://github.com/esrf-emotion/emotion

E-Motion is a generic python library for motors control. It provides an easy and rapid way to interface motor controllers, offering a common API for standard features.

Project goals

We have a large and constantly increasing number of motor controllers to deal with. This creates many maintenance and evolution problems:

-Integration of new controllers.
-Maintenance and migration from old platforms.
-Implementation of new features like trajectories or continuous scans.
-Experiments require more and more specific features from motor controllers.

Our objectives :

-To propose a library for easy implementation of new controllers sharing common functionalities.
-To ease the usage of motors in control systems by providing a common and simple API.
-To take benefits from standard and powerful features.

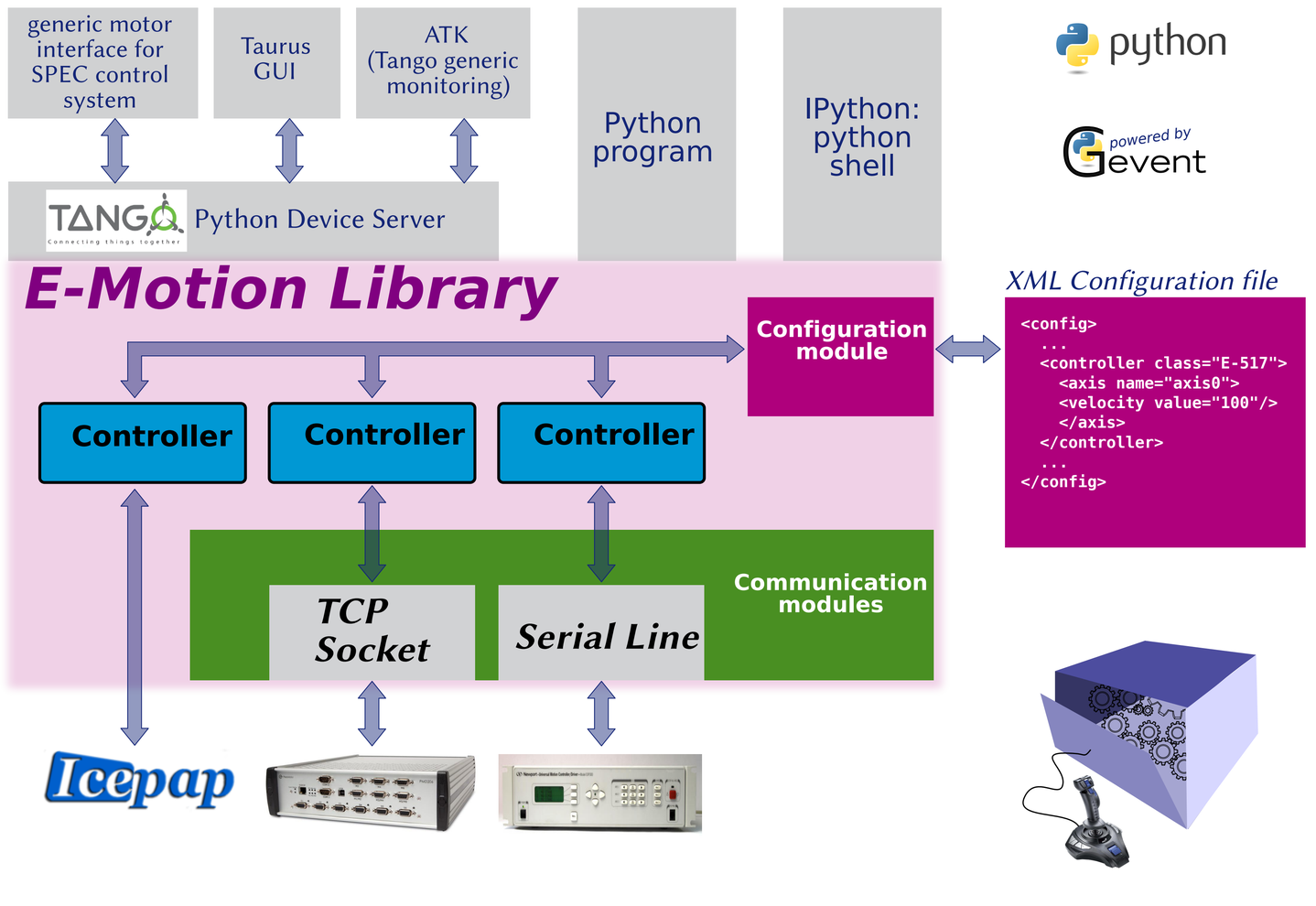
Integration of a new hardware controller

To integrate a new device, a new controller has to be written. This task is made easy as the creator has only to code few entry points: that is to say to write about 10 standard methods like :

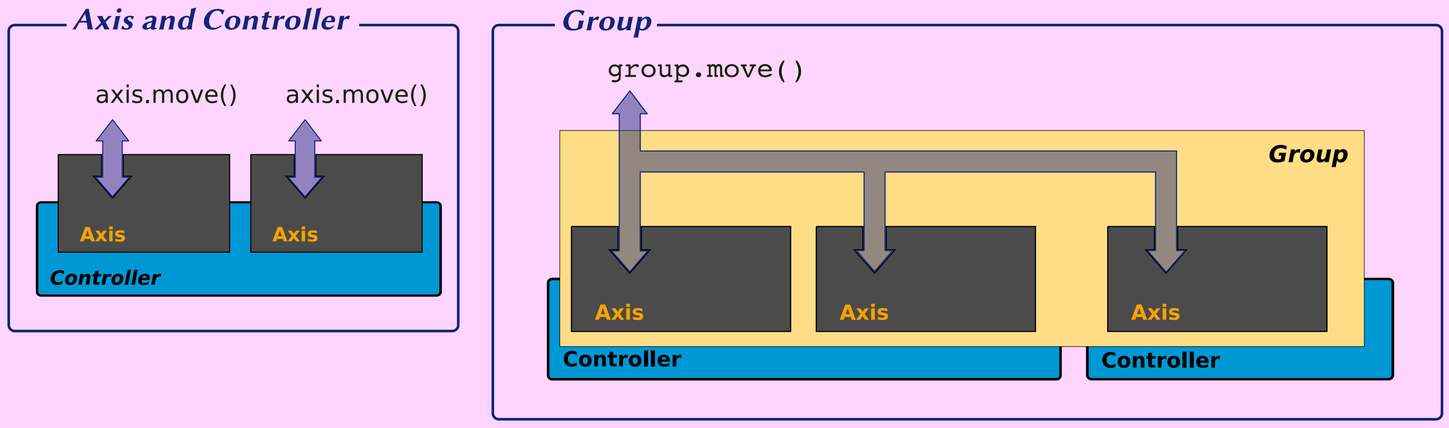
```
initialize_axis()   start_one()   start_all()
set_velocity()      state()
[...]
```

We also provide some modules to simplify the communications via standard protocols (serial line; socket; gpib)

Architecture

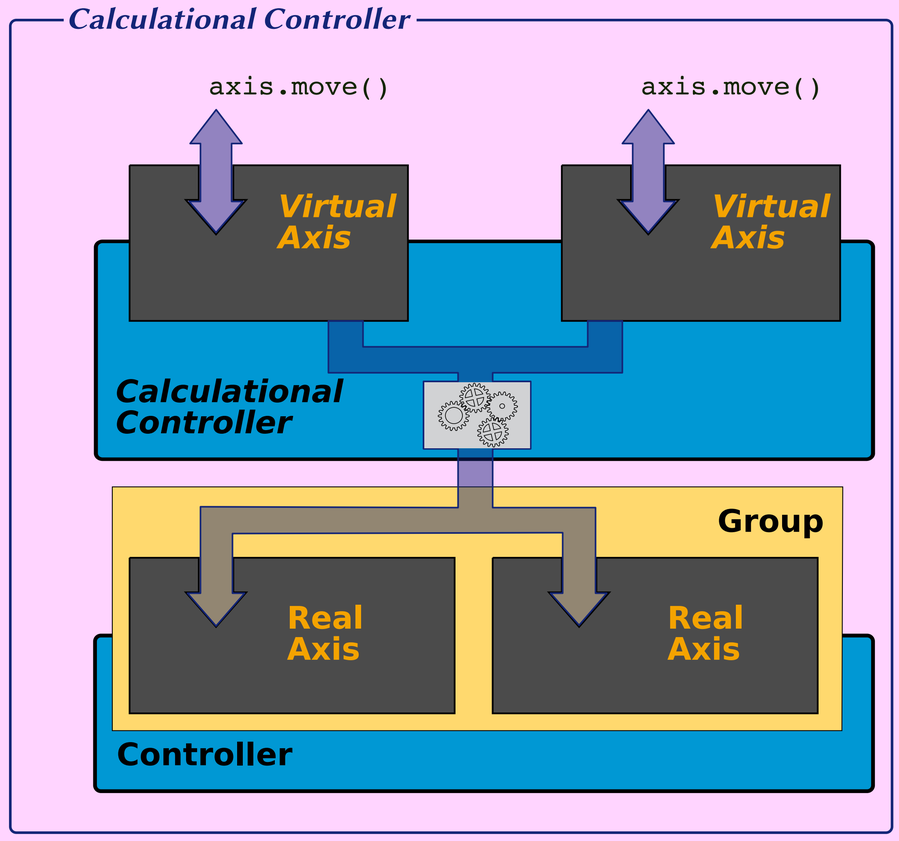


E-Motion Objects



E-Motion mechanisms are based on **Controllers**, **Axis** and **Group** objects:

- A **Controller** exports an **Axis** object for each single motor.
- A **Group** allows to manage a set of motors simultaneously.
- A **Calculational Controller** exports virtual axes and allow to drive slave real axes according to a user-defined mathematical relation (ex: tripod or spectrometer linked to beam energy).



E-Motion API

E-Motion API tried to be very simple in order to be easily used in control systems. Examples of API functions :

```
start_one()      acceleration()    steps_per_unit()
position()       home_search()     limits()
state()          is_moving()       measured_position()
velocity()
```

Usage example

```
import emotion
emotion.load_config(xml_config)

my_axis = emotion.get_axis("axis0")

print my_axis.position()
my_axis.move(42)
```

Implemented controllers

Hardware controllers

PI E-517; PI E-753; IcePap; PMD206; NewFocus 8753; NanoMotion FlexDC

Calculational Controllers

Slits; Spectrometer; Qsys Tetrapod; Tripod

Deployment

E-Motion prototype is already in use on ESRF beamlines :

- It drives the sample stage on ID16a nano-imaging beamline (nm repeatability motion achieved with a piezo hexapod)
- It is used for a spectrometer prototype on ID26 beamline
- It drives all stepper motors and piezo on ID30 crystallography beamline

Perspectives

-To increase the number of supported controllers
-Interface controllers provided with Windows DLLs
-Advanced functionalities : Trajectories...

Gevent

E-Motion heavily relies on **gevent**[1] ; gevent is a coroutine-based Python networking library that uses **greenlet**[2] to provide a high-level synchronous API on top of the **libev**[3] event loop.
The benefits of gevent for E-Motion are:

- **no OS-level threading**; only cooperative, **lightweight execution units** based on greenlet
- simplified coding of motion loops, thanks to **cooperative asynchronous calls** written like synchronous ones (avoid callbacks spaghetti)
- **futures, timeouts**: features from gevent API
- **fast event loop** based on libev

E-Motion can integrate seamlessly **in any host application running the gevent loop**.

[1]: http://www.gevent.org
[2]: http://greenlet.readthedocs.org
[3]: http://software.schmorp.de/pkg/libev.html

Acknowledgments

Thanks to all the Beamline Control Unit of ESRF for their advices and suggestions.

Made with INKSCAPE

GitHub
http://github.com/esrf-emotion/emotion