CSMA 2022
15ème Colloque National en Calcul des Structures
16-20 Mai 2022, Presqu'île de Giens (Var)

CRAPPY: Command and Real-time Acquisition in Parallelized PYthon

J.-F. Witz[1], A. Weisrock[1], V. Couty[1], F. Lesaffre[1]

[1] *LaMCube, Lille*

Résumé — CRAPPY is an acronym and stands for Command and Real-time Acquisition in Parallelized PYthon.

Crappy is developed at the LaMCube, a mechanical research laboratory based in Lille, France to provide a powerful and easy-to-use framework for materials testing.

In order to understand the mechanical behaviour of materials, we tend to perform tests with more and more sensors and actuators from various suppliers. There's thus an increasing need to drive these devices in a synchronized way while managing the high complexity of the setups.

As we are one step ahead of industrials, the commercially available testing solutions may also not be well-suited to our objectives. Custom software solutions thus need to be developed in order to further improve our tests.

These are the original purposes of Crappy : providing a framework for controlling tests and driving hardware in a synchronized and supplier-independent software environment.

To this end, choices were made that are now keys to our framework :

— open-source : It is important for us that everyone can use our work, and bring its own code to the world.

— modular : We provide a software basis that can easily be extended to drive new hardware and perform custom operation on data.

— simple : Python has been chosen for its high level. We are not developers, and neither are our users, so we cannot afford to use a low-level programming language. We work with typical loop time of more than 1 millisecond (10ms most of the time), and Python is enough for that. It is also pretty easy to add a small piece of C/C++ to the Python code if a speedup is needed.

— performance : A great deal of work is made to ensure the performance of the framework. Most tests require a good repeatability and stability, and may become hazardous in case of non-handled issue. parallelization : The key to a good test is the synchronisation between the different sensors. This is why we chose to massively parallelize our framework, ensuring every device can run simultaneously in a same time basis. This is also one of the major difficulties we have to deal with in Python.