Acta Polytechnica CTU Proceedings **18**:15–19, 2018
doi:10.14311/APP.2018.18.0015

MULTI-CHANNEL CONTROL SYSTEM FOR IN-SITU LABORATORY LOADING DEVICES

Václav Rada[a, b, *], Tomáš Fíla[a, b], Petr Zlámal[a, b], Daniel Kytýř[a], Petr Koudelka[b]

[a] *Czech Academy of Sciences, Institute of Theoretical and Applied Mechanics, Prosecká 76, 190 00 Prague 9, Czech Republic*

[b] *Czech Technical University in Prague, Faculty of Transportation Sciences, Konviktská 20, 110 00 Prague 1, Czech Republic*

[*] corresponding author: rada@itam.cas.cz

Abstract. In recent years, open-source applications have replaced proprietary software in many fields. Especially open-source software tools based on Linux operating system have wide range of utilization. In terms of CNC solutions, an open-source system LinuxCNC can be used. However, the LinuxCNC control software and the graphical user interface (GUI) could be developed only on top of Hardware Abstraction Layer. Nevertheless, the LinuxCNC community provided Python Interface, which allows for controlling CNC machine using Python programming language, therefore whole control software can be developed in Python. The paper focuses on a development of a multi-process control software mainly for in-house developed loading devices operated at our institute. The software tool is based on the LinuxCNC Python Interface and Qt framework, which gives the software an ability to be modular and effectively adapted for various devices.

Keywords: CNC, controlling, LinuxCNC, Python Interface, Python, parallel programming, Qt, PyQt, Qwt, PythonQwt, LabJack.

1. Introduction

Computer numerical control (CNC) is used for automation of a broad range of machines e. g. milling machines, lathes, 3D printers etc. CNC allows for very effective and precise manufacturing of parts with complex shape and has many other advantageous applications in various industrial fields [1]. There are many CNC software solutions, which vary in price, performance or closed-source commercial (Siemens, FANUC, LabVIEW) and open-source solutions (LinuxCNC, Arduino - primarily for hobby operation). Our research group tends to use open-source solutions, because our needs are different from conventional industrial CNC applications [2]. Therefore, some properties of the software has to be operationally modified according to our requirements. For this purpose, closed-source commercial CNC software is not suitable for use with our devices. In our applications, several types of actuators are used including stepper, servo-motors and linear voice coil actuator. Purpose of the designs can be divided into three groups:

- mechanical loading machines (e. g. in-situ loading devices for X-ray computed tomography)
- positioning machines (e. g. optics and sample positioning)
- sample preparation devices (e. g. automatic grinders)

Each machine type is equipped with common parts such as actuator, encoder, limit switches and with application specific equipment such as load cell, thermometer etc. The software package introduced in this paper allows for controlling of both the common and application specific equipment and user the interface can be tailored for the specific application owing to LinuxCNC [3] Python Interface [4].

2. Approach

In Python, parallel programming [5] is a complex task, particularly due to Python global interpreter lock (GIL). GIL prevents multiple threads to access interpreter internals at the same time by serializing the requests. Python threads share all types of variables natively which makes them easy to use, but these threads cannot effectively bring any performance increase by utilizing multiple CPU cores. In order to force Python interpreter to perform multiple tasks in parallel and utilize multiple CPU cores, separated Python processes must be used, which allows for executing tasks of each process simultaneously. To demonstrate Python multithreading and multiprocessing performance, a computational test of prime factorization was performed on 10 million integers. The testing scripts [6] was run on a 4-core, 8-thread (Intel Core i7-4790K @4.4 GHz) machine. The computation time results are shown in figure 1. The results confirm that Python multiprocessing threads do not bring any performance increase. In fact, multiprocessing threads decrease the script performance substantially due to overhead caused by thread switches made by

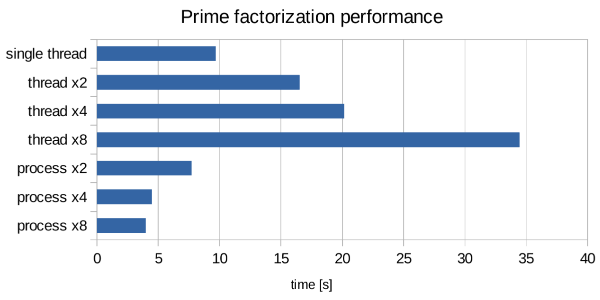


FIGURE 1. Comparison of multithreading and multiprocessing performance

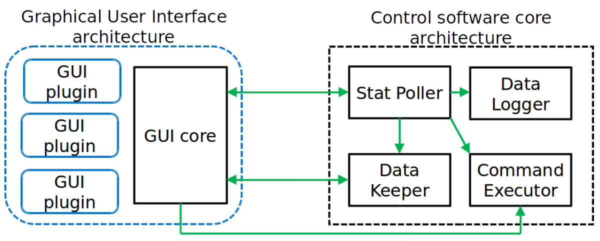


FIGURE 2. Inter-process communication scheme

Python interpreter. On the other hand, multiprocessing processes do increase the performance because of parallelism.

However, sharing variables between Python processes is in most cases very ineffective. Mostly, it is more effective to implement some kind of inter-process communication. Python supports multiprocessing library [7] which provides communication between processes using pipes or queues. Python multiprocessing pipes are the simplest approach of multiprocessing communication in Python. The pipes connect only two processes. One end of the pipe is used to send data by one process, the other end is used to receive the data by the other process. In order to provide connection between more than two processes, multiprocessing queues can be used, but multiprocessing pipes give more performance due to their simplicity.

The control software consists of a multi-process core and graphical user interface (GUI) process connected by multiprocessing pipes. The core includes four processes, stat poller, data logger, data keeper and command executor. Its interconnection is shown in figure 2.

The crucial process in the control software core is the stat poller. It is based on `linuxcnc.stat()` of LinuxCNC Python Interface and extended by methods providing communication with sensors, output log file management etc. Stat poller periodically updates machine status variables calling `poll()` method and obtains values from sensors. Obtained data is sent to data logger, data keeper and to GUI. Data logger receives data from stat poller and saves it periodically to output file. Data keeper keeps received data from stat poller in an array and sends the data to GUI on request to show the data in a graph. The control software core contains also command executor process which is based on `linuxcnc.command()` of LinuxCNC Python Interface. It provides execution of dynamically generated Python string commands received from GUI or stat poller using exec statement.

3. User Interface

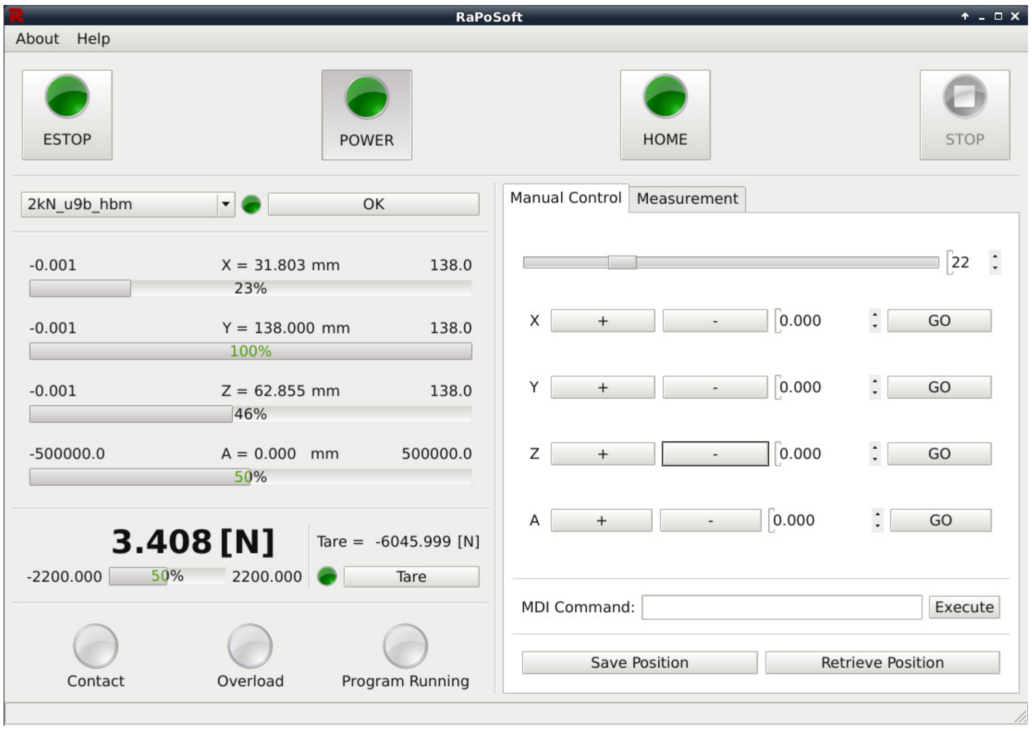
The main objective of the GUI is to not affect the control software core performance, so the user interface runs in a process separate from the control software core and is connected with it using multiprocessing pipes. All the pipes leading from control software core are connected to GUI core, which handles communication between control software core and GUI process itself.

The user interface is aimed to be modular and straightforwardly adaptable for various experimental devices. Each experimental device uses its specific LinuxCNC initializing file, which contains data needed for control software startup. To take into consideration differences of particular devices, the initialization files were extended by specific entries. These entries define active axes of a device, axes equipped with a sensor (measurement axes), axes units etc. On control software startup, initializing file is loaded and user interface is dynamically generated considering specific entries in initializing file and individual experimental device needs. The user interface of the control software is developed using Qt framework in cooperation with Python binding PyQt [8]. The user interface is designed as a set of plugins (QWidgets), which contributes to straightforward adaptation to new experimental devices and helps to keep the user interface source code well structured.

Currently, various GUI plugins were created to cover specific needs of laboratory devices. Emergency stop button, power button etc. are elementary plugins and are common to all devices. Other plugins allow for controlling by executing manual data input (MDI) commands based on G-code (RS-274), manual control of positioning axes of the device or sensors management.

For proper sensors functionality initializing files of all supported sensors were created. On control software startup, these initializing files are loaded into software internal database. The files consist of numerical values important for appropriate operation of the sensor specified by calibration protocol or user requirement (measurement range, safe overload, amplification, sensitivity).

After the system startup user specifies appropriate sensors which are mounted in particular experimental device using plugin for sensors choice. After the sensors are chosen, GUI sends a command to control software core to start obtaining data from sensors periodically (e. g. force, temperature etc.). The control software core sends these data to GUI and the data is shown in e. g. load cell plugin. All the data is periodically sent to data logger and saved to output



2kN_u9b_hbm

-0.001 X = 31.803 mm 138.0
23%
-0.001 Y = 138.000 mm 138.0
100%
-0.001 Z = 62.855 mm 138.0
46%
-500000.0 A = 0.000 mm 500000.0
50%

3.408 [N]
Tare = -6045.999 [N]
-2200.000 50% 2200.000

22
X 0.000
Y 0.000
Z 0.000
A 0.000
MDI Command:

Figure 3. Overall view of the control software user interface

file. Thus a plugin managing data logging was created. It enables the user to export output file either to a static config-given folder or using a save dialog interface. The plugin also enables creating new empty output file leaving the old data in temporary folder to prevent accidental data loss.

The control software purpose is to reliably perform mechanical experiments. There was a special plugin created for controlling the displacement-driven and force-driven experiments.

The control software provides a plugin allowing for plotting the data during experiment in a live graph. The live graph window is based on Qwt (Qt Widgets for Technical Applications) library written in C++ and Python binding for this library named PythonQwt . The Qwt library was chosen for its performance to achieve higher live graph FPS. The plugin enables to specify the amount of samples to plot. The refresh rate of the live graph decreases with increasing amount of samples. The optimal refresh rate is calculated automatically based on the number of samples using linear approximation.

Overall view of the control software user interface is shown in figure 3.

4. Applications

The modular design of the control software gives it an ability to control various laboratory devices separatedly or combine them in laboratory setups and control multiple devices as a single complex closed-loop controlled system [9]. Moreover it is possible to integrate external procedures (e.g. auto-calibration [10]). Functionality of the system is hereinafter demonstrated on the setup consisting of uni-axial loading device in arrangement for optical strain measurement where the camera is placed on triaxial table. The experimental procedure is in detail described in [11].

This is achieved by creating LinuxCNC custom initializing file for each device and extending it with `[RAPO]` section. The section name comes from the control software name RaPoSoft (Rada Positioning Software). The section contains specific additional parameters. For instance, the additional parameters of a triaxial motorized table are as follows:

```
[RAPO]
AXES\_ACTIVE␣=␣X␣Y␣Z
AXES\_UNITS␣=␣mm␣mm␣mm
AXES\_MEASUREMENT␣=
LJ\_AINS\_ACTIVE␣=
LJ\_AINS\_GAIN␣=
LJ\_AINS\_INVERT␣=
```

The motorized table is equipped with 3 active axes (X, Y, Z), all of the axes units are millimeters and it has no measurement (loading) axis, i. e. all of the axes are used only for positioning. Devices with measurement (loading) axes included mostly use various sensors (typically load cells). Parameters needed for sensors operation come with `"LJ"` prefix, which refers to the LabJack measurement cards (T7-PRO, LabJack Corporation, USA) , used at our institute for establishing connection with the sensors using `labjack.ljm` library for Python.

Usage of these parameters is demonstrated on uni-axial loading device.

```
[RAPO]
AXES\_ACTIVE␣=␣X
AXES\_UNITS␣=␣um
AXES\_MEASUREMENT␣=␣X
LJ\_AINS\_ACTIVE␣=␣0
LJ\_AINS\_GAIN␣=␣51
LJ\_AINS\_INVERT␣=␣1
```

The uniaxial loading device is equipped with a single active axis, the units are micrometers and the active axis is also measurement (loading) axis. The loading axis is equipped with a sensor (load cell) connected to LabJack analog input no. 0. The LabJack analog input signal can be amplified using LJTick-InAmp amplifier and the amplification coefficient of the analog input is given by the gain parameter. The analog input signal can be inverted to set the appropriate load cell polarity while performing tension and compression tests.

The motorized table and the uniaxial loading device can be used together as well. The parameters in initializing file look as follows:

```
[RAPO]
AXES\_ACTIVE␣=␣X␣Y␣Z␣A
AXES\_UNITS␣=␣mm␣mm␣mm␣um
AXES\_MEASUREMENT␣=␣A
LJ\_AINS\_ACTIVE␣=␣0
LJ\_AINS\_GAIN␣=␣51
LJ\_AINS\_INVERT␣=␣1
```

The laboratory setup consists of 4 active axes in total, three of them are positioning axes of the table with millimeter units, one of them is measurement (loading) axis with micrometer units. Parameters for the LabJack analog inputs configuration are the same as for the single uniaxial loading device usage.

The control software allows for controlling devices equipped with multiple sensors. This can be demonstrated on initializing file of a loading device developed at our institute used for four-point bending tests. The initializing file looks as follows:

```
[RAPO]
AXES\_ACTIVE␣=␣X␣Y
AXES\_UNITS␣=␣um␣um
AXES\_MEASUREMENT␣=␣X␣Y
LJ\_AINS\_ACTIVE␣=␣0␣1
LJ\_AINS\_GAIN␣=␣51␣51
LJ\_AINS\_INVERT␣=␣1␣1
```

The device consists of 2 active axes, both of them are loading (measurement) axes with micrometer units. The device is equipped with 2 sensors (load cells) connected to LabJack analog inputs no. 0 and no. 1. Both of the analog inputs are amplified using coefficient 51 and their signal is inverted.

5. Conclusions

The presented control software allows for controlling laboratory devices developed at our institute, which are used for mechanical testing in different loading scenarios including time-dependent processes under controlled ambient conditions. It allows to control laboratory devices actuated by stepper, servo-motors, linear voice coil actuator etc. It provides support for various sensors operating on analog signal proportional to the applied excitation voltage (mV/V) e.g. load-cells or thermometers. Moreover it is possible to integrate external closed-loop driven procedures. The modular design enables very effective adaptability and extendability which gives the control software capability to be used in a long-term way.

Acknowledgements

The research has been supported by the European Regional Development Fund in frame of the project Kompetenzzentrum MechanoBiologie (ATCZ133) in the Interreg V-A Austria - Czech Republic programme and by Operational Programme Research, Development and Education in project Engineering applications of microworld physics (CZ.02.1.01/0.0/0.0/16_019/0000766).

References

[1] P. Smid. *CNC Programming Handbook, Third Edition.* Industrial Press, Inc., 2007.

[2] S. Anderberg, T. Beno, L. Pejryd. Energy and cost efficiency in CNC machining from a process planning perspective. In S. G. (ed.), *Sustainable Manufacturing*, pp. 393–398. Springer, 2012.

[3] The LinuxCNC Team. *LinuxCNC Getting Started Guide.* Samurai Media Ltd., 2016.

[4] LinuxCNC Python Interface Documantation. `http://linuxcnc.org/docs/2.6/html/common/python-interface.html`. Accessed: 2018-06-15.

[5] G. Zaccone. *Python Parallel Programming Cookbook.* Packt Publishing Ltd., 2015.

[6] E. Bendersky. Python - parallelizing cpu-bound tasks with multiprocessing. `https://eli.thegreenplace.net/2012/01/16/python-parallelizing-cpu-bound-tasks-with-multiprocessing`. Accessed: 2018-07-03.

[7] B. Jones, D. Beazley. *Python Cookbook - Recipes for Mastering Python 3.* O'Reilly Media, 2013.

[8] M. Summerfield. *Rapid GUI Programming with Python and Qt: The Definitive Guide to PyQt Programming.* Prentice Hall Press, 1st edn., 2007.

[9] D. Kytyr, P. Zlamal, P. Koudelka, et al. Deformation analysis of gellan-gum based bone scaffold using on-the-fly tomography. *Materials and Design* **134**:400–417, 2017. doi:10.1016/j.matdes.2017.08.036.

[10] M. Vopalensky, D. Vavrik, I. Kumpova. Optimization of the X-ray tube voltage with respect to the dynamical resolution in radiography and tomography. *NDTnet Journal* **2018-02**, 2018.

[11] T. Doktor, I. Kumpova, M. Sniechowski, et al. Influence of printing and loading direction on mechanical response in 3d printed models of human trabecular bone. *Acta Polytechnica CTU Proceedings 18(10):24–27* 2018. doi:10.14311/APP.2018.18.0024.