GUI-based beamline control system at P02.1

M. Domaracky, J. Bednarcik, J. Michalikova, K. Glazyrin, H.P. Liermann

P.J. Šafárik University in Košice, Faculty of Science, Park Angelinum 9, 04145 Košice
1 Deutsches Elektronen-Synchrotron (DESY), FS-PE group, Notkestr. 85, 22607 Hamburg, Germany

The main goal of this work was to create a modular, high performance, robust, generic user interface for control devices that are used on the beamline during experiments and provide users with GUI (graphical user interface) based tools to create macros and scans, which are easy to create but robust enough to satisfy their needs.

Figure 1 shows beamline main GUI application we developed for P02.1 beamline. This interface sends commands to TANGO servers via network interface. Each piece of hardware has its own controller which can be operated by TANGO servers. As a result of this, user can move motors and open shutters as easy as one click on device icon. No need to write sequence of commands or know how to program at all. Python was used as main programming language. Each component (widget) used in the application is created in the most general way using the QT designer [1]. Source code is available for others in GIT repository [2]. Each beamline device is polled every 0.5 second for its state. This can overload main CPU, if many devices are polled simultaneously. To overcome this, each device controller knows if it is visible or not. If device controller is hidden then its polling stops. This approach reduces CPU load significantly. To reduce CPU load even more, every widget runs in its own thread. From a program point of view, the beamline control consists of various general widgets. It is based on a generic beamline widget which serves as a template. For example, every motor has its own motor control widget. Such widgets can be stacked in other widgets and create stage controller. This approach can be applied to any other device on the beamline. To create

Figure 1: P02.1 beamline main GUI provides user with visualization of the beamline status. Each component reflects its current state and position in a beam. User can simply click on a component and change its state, which is immediately reflected on a real device. Some components have dedicated controller tab. User can stop all movements with one button. Some components can be controlled only in the expert mode, which is password protected. User is properly notified when some device error occurs and application is trying to reconnect in the background.
user defined macros and scans we developed a program named REVOLVER. The REVOLVER was created to allow user efficiently automate spatially and/or time resolved macros and scans. Every macro consists of defined steps which are executed one after another (see Fig.2). User have options to create robust macros but as easy as possible. To create one macro step, user need to define detector, motor and optional macro settings (see Fig.3). Motor position can be defined as a discrete value or as an interval. User can define three types of macros: (i) **simple motor macro**, which moves motor to defined positions and takes XRD pattern, (ii) **looping motor macro**, which moves two motors in a loop to record XRD patterns, (iii) **time macro**, which periodically takes series of XRD patterns. Furthermore there are two types of scans with the 2D detector: (a) **basic scan**, which moves motors to defined positions and shows average pixel intensity within defined regions of 2D XRD pattern, (b) **stroboscope**, which lets user to move one defined motor and continuously shows average count from user defined regions.

The third application we developed is the temperature control application. Main purpose of this application is to collect data in-situ while temperature is rising or sinking. Sample environments that can currently be controlled through such a macros are a HOT-AIR-Blower or a Cryostreamer. To ensure that the setpoint temperature has been reached, an algorithm monitors the moving average value calculated from several past values stored in a FIFO (First-In-First-Out) buffer. Reaching desired temperature can be divided into two steps: ramping and stabilizing phase. Each step has its own time limit for reaching the setpoint. Additionally, user can define a temperature threshold, which is used in algorithm to decide whether temperature is stabilized or not.

**Acknowledgement** M.D. acknowledges the financial support from University Science Park TECHNICOM for Innovation Applications Supported by Knowledge Technology ITMS: 26220220182, supported by the Research & Development Operational Programme funded by the ERDF. Parts of this research were carried out at the light source PETRA III at DESY, a member of the Helmholtz Association (HGF).

**References**

[2] https://github.com/DESY-Petra-III/P02