ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEDPR03

SYNOPTIC GUIS IN NSRC SOLARIS FOR BEAMLINES AND ACCELERATORS VISUALIZATION AND CONTROL

M.K. Fałowski†, T.R. Noga, N. Olszowska, M. Zajac
National Synchrotron Radiation Centre, SOLARIS, Krakow, Poland

Abstract

High demand from scientists and operators to create new, clear and intuitive SCADA graphical interfaces for new beamlines and replace or supplement existing beamlines' and accelerators' graphical user interfaces is a challenging task. This is not only time consuming but very often requirements from users vary, change quickly and even sometimes they are mutually exclusive. To meet this challenge and provide clear, scalable and ergonomic graphical user interfaces, SOLARIS chose 'Taurus' and 'svgsynoptic2' to create synoptic applications which allow to visualize and control beamlines and accelerators with ease. In addition, it was decided to use identical scheme of visualization and control for synoptic applications on all beamlines, so scientists can get used to it, even if they carry out research on different beamlines. This paper presents the overall architecture and functionality of the applications.

INTRODUCTION

One of the most important roles of the control systems is to show the actual state of controlled process. Visualisation of the state of the process is crucial for operators to properly control it and react in any case of event or emergency occurrence during the process. This task is achieved with SCADA systems. One of the elements of SCADA are synoptic views, which are graphical representation of the process. They should provide as much information about control system as possible but also preserve readability, be reliable and intuitive.

In NSRC SOLARIS there are two accelerators, linear and synchrotron, two fully operational beamlines, one during commissioning and next four under construction. Each system contains hundreds of devices and signals. For now, the main task for SOLARIS is to deliver stable beam for scientists with interruption and breaks as short as possible. To achieve this goal it is critical to monitor accelerators and beamlines and in case of any accidents fast recognition and repair of the fault.

PREVIOUS SOLUTIONS

Since the start of operation, applications to represent beamlines and accelerators in SOLARIS were prepared with JDraw synoptic and Qt GUIs developed with Taurus. They had a lot of limitations and were not very intuitive. Moreover, they could not represent entire subsystems in one view and had to be divided into tabs or segments.

Example of an application is showed in Fig. 1. It is used to visualize segments of the beamline or the storage ring. This application parses configuration file and generates Taurus widgets. It has many disadvantages, such as very complicated and hard to modify code (e.g. adding new type of device or adapting to newer version of dependency libraries was very time consuming), switching between segments was taking a lot of time, resulting in not showing any panel for short period, and configuration files had custom and hard to read format. In addition, non-graphical representation required operators to learn positions of the elements to quickly recognise state of the subsystems. Last but not least, application used custom and unintuitive colours for defining states (e.g. orange colour represented closed state, where in Tango it represents alarm state). Next example is the application responsible for displaying interlocks shown in Fig. 2. It had many tabs so searching for an interlock was difficult and sometimes caused oversight of it by operators. Last example is JDraw application of LINAC showed in Fig. 3. Due to static view, LINAC segments had to be separated into many tabs.

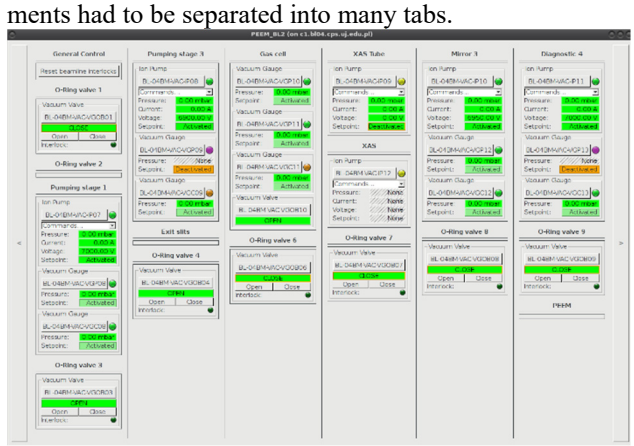

Figure 1: Application representing one of the segments of the beamline. It contains states of the devices, such as valves and thermocouples, and allows to control them.

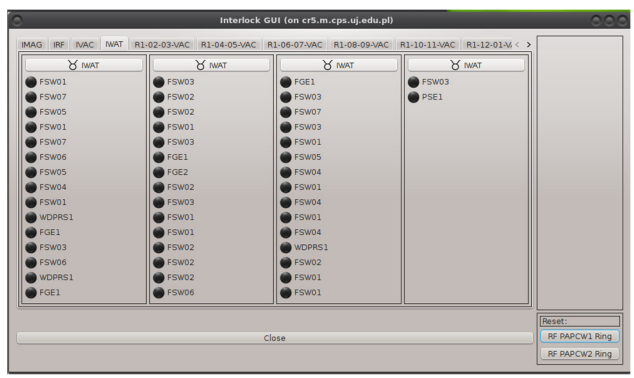

Figure 2: Application responsible for displaying interlocks in the accelerators.

† michal.falowski@uj.edu.pl

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEDPR03

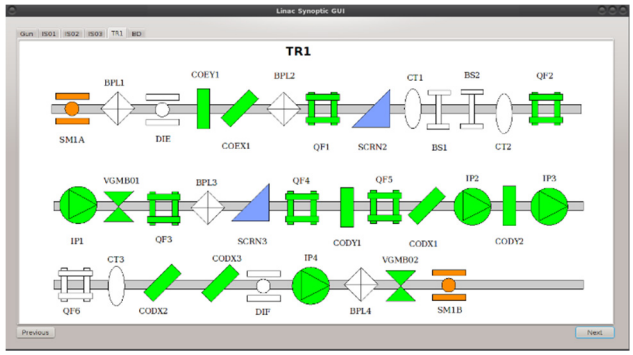

Figure 3: Synoptic representing one of the sections of the LINAC.

Poor system state representation, long-time development of the new subsystems and features, and growing requirements from users and new beamlines being constructed was the motivation to change used tools and search for new allowing to prepare intuitive, clear and with high functionality synoptic views of the SOLARIS systems.

CHOICE OF TECHNOLOGY

Requests for quick to develop, readable and scalable synoptics were the most important factors affecting the software selection. Moreover, software choice was also affected by TANGO controls which is main framework used for development of the control system in SOLARIS. Selected tools greatly improved quality and time of delivery of the new applications.

Taurus

Taurus is a Python-based SCADA framework allowing rapid creation of user interface applications for scientific and industrial control systems [1]. It is a commonly used tool in SOLARIS and has good integration with TANGO so it was a natural candidate for creating synoptic applications. It uses PyQt for the GUI and was mostly used to create control widgets in synoptics. It is also being used by the svgsynoptic library.

SVG Synoptic

JDraw is a specialised drawing tool for creating synoptics provided by TANGO [2]. Unfortunately, its usage is very limited, it provides only simple accessories and its viewers provide only static view. For this reason MAX IV created a library called svgsynoptic, which provides interactive visualisation of control system based on SVG files and references to control system entities inside SVG objects [3].

Implementation of the library is based on Taurus, PyQt binding to JavaScript and QtWebKit. It has features like panning, zooming and zoom levels, displaying/hiding layers or invoking commands, like displaying device widget based on TaurusDevicePanel.

This library, which now has second version – svgsynoptic2 [4] – is the main core of the SOLARIS' synoptics as it allows for fast development and design, and fine control of them.

Inkscape

Inkscape is an advanced, free and open-source vector graphics editor with many tools and accessories [5].

It is used to create SVG files for the synoptic views and thanks to its features, like cloning, advanced alignment, grouping or keyboard shortcuts, it is possible to create very complicated synoptic views of the accelerators and beamlines in a short time. Furthermore, it can also be used by users to create or modify their own shapes, symbols and views.

Inkscape is a great duet with svgsynoptic and both had large impact on design of applications in SOLARIS.

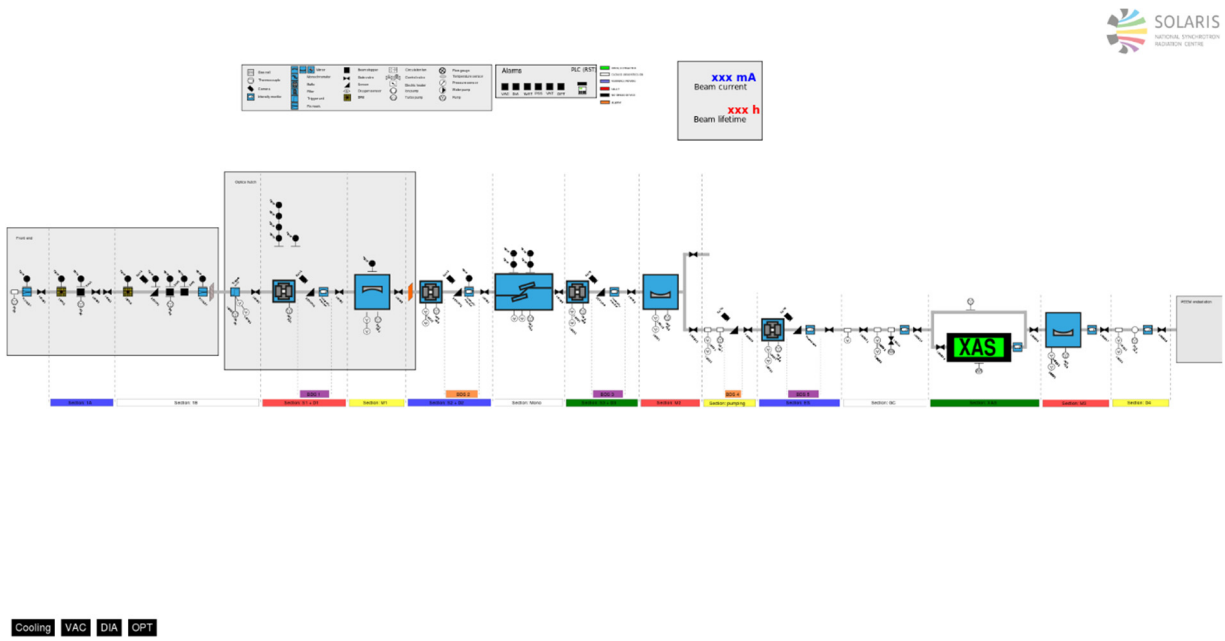

Figure 4: First synoptic view in SOLARIS based on SVG, representing one of the beamlines.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEDPR03

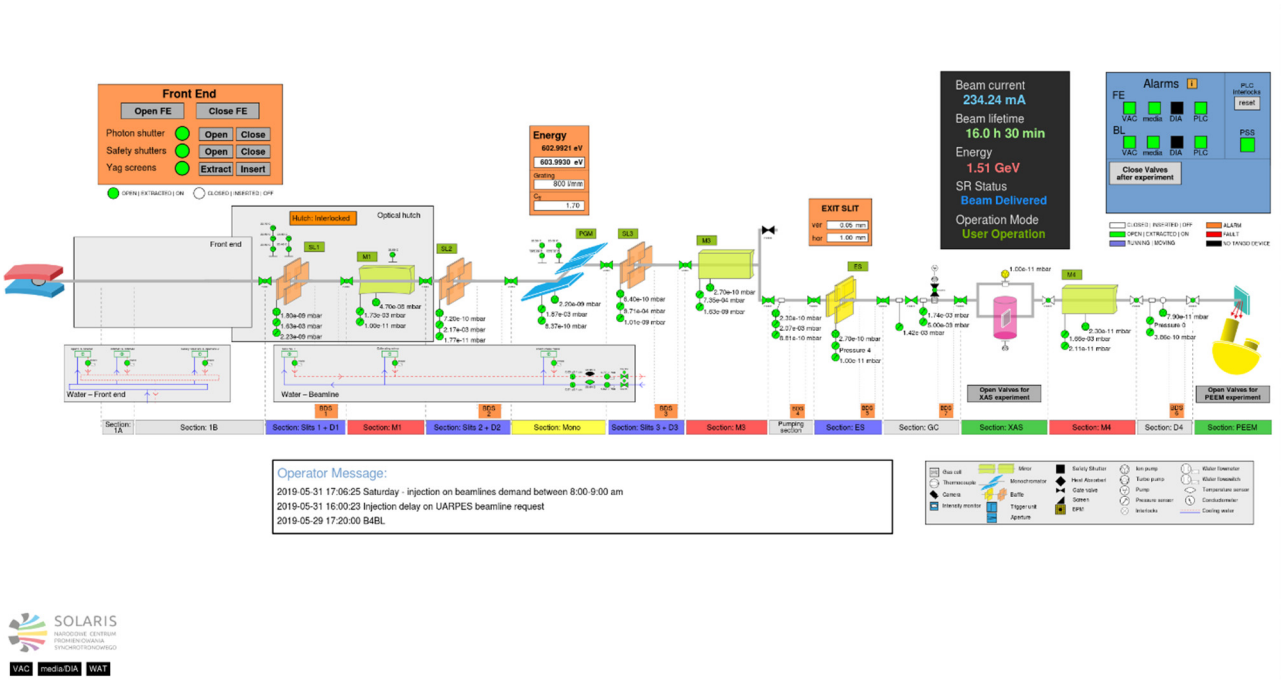

Figure 5: Improved synoptic application of the same beamline showed in Fig. 4.

Facadedevice

Facadevice is a python library extending high level server API of the PyTango library. It is based on TANGO events and uses reactive programming paradigm [6].

It was used to create high level TANGO devices providing more advanced commands like opening valves in sequences, and aggregation of the states of the devices, reducing amount of the control code in the synoptic GUIs.

FIRST SYNOPTICS

First synoptic created with svgsynoptic in SOLARIS was prepared for one of the beamlines (Fig. 4). It contained many simple and "flat" symbols, division into layers (like vacuum, water, diagnostic subsystems) and sections, and allowed to run simple commands on the devices such as opening and closing valves. It was gladly accepted but there were new request from users to create it more clear and functional.

Next iteration of the synoptic view for the beamline has improved and more isometric view (Fig. 5). It gained buttons for opening and closing frontend and valves on the entire beamline, and resetting interlocks (responsible for closing and opening valves or resetting interlocks are high level Tango devices created with facadedevice, synoptic only runs command provided by them), two zoom levels (frontend and some diagnostic elements are shown only when zoomed). There is also possibility to run external application (for controlling optics, displaying cameras stream and others) or modify some device parameters (using popup widgets). The same type of view was made for second operational beamline.

For accelerators three synoptic views were made, one for LINAC, one for accelerators' interlocks (fig .6) and one for beamline frontend being commissioned. The interlocks synoptic contains over one hundred of signals and is divided into two zoom levels (Fig. 6 and Fig. 7). On the first zoom level interlocks are grouped into sections, so one indicator shows whether there is any interlock in one of the sections of the accelerators (it is achieved thanks to facadedevice by aggregating states of each interlock in one Tango device). On the second zoom level each interlock signal is represented by one indicator.

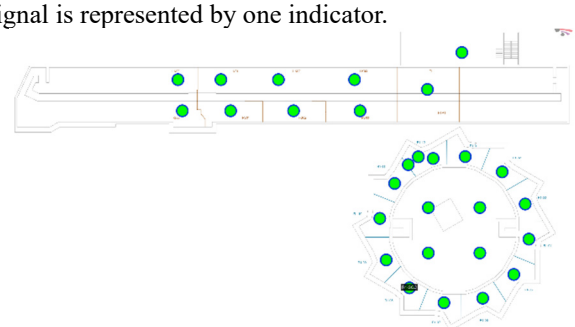

Figure 6: Accelerators' interlocks synoptic application zoom level 0.

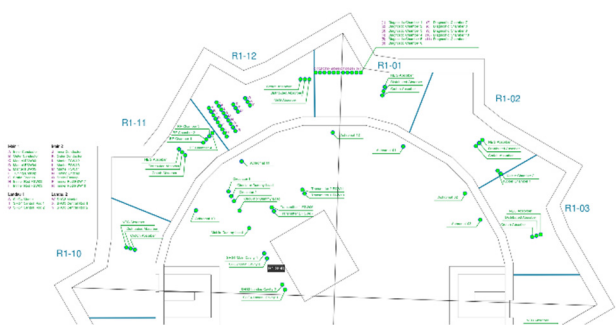

Figure 7: Accelerators' interlocks synoptic application zoom level 1.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEDPR03

IMPORVEMENTS AND STANDARIZATION

Despite the great utility and improvements, prepared synoptics have some disadvantages. Control logic of each synoptic was defined in every project separately and it was just being copied between them. The result was that any simple change in one of the applications required modifying every project and in the end the control logic began to differ. Besides, users sometimes have mutually exclusive requirements so the maintenance of each application gets harder. Moreover, beamline synoptics have bright colours of some elements and panels which are too similar to colours of states defined by TANGO like “alarm” or “moving”. To prevent these situations three steps were taken.

First of all, it was necessary to reach compromise with users about general look, functionality and actions (like left and right mouse button clicks). Together with operators and beamline managers it was decided to use the same colour scheme for all beamlines and accelerators, similar arrangement and configuration of information and command panels, and to have the same defined actions. Furthermore, symbols of elements such as water flowmeters or temperature sensors were unified in cooperation with Technical Department, so they are now the same or similar to symbols from technical drawings. This helped to keep simplicity of the view, homogeneity with the documentation and synoptics can also serve engineers from Technical Department.

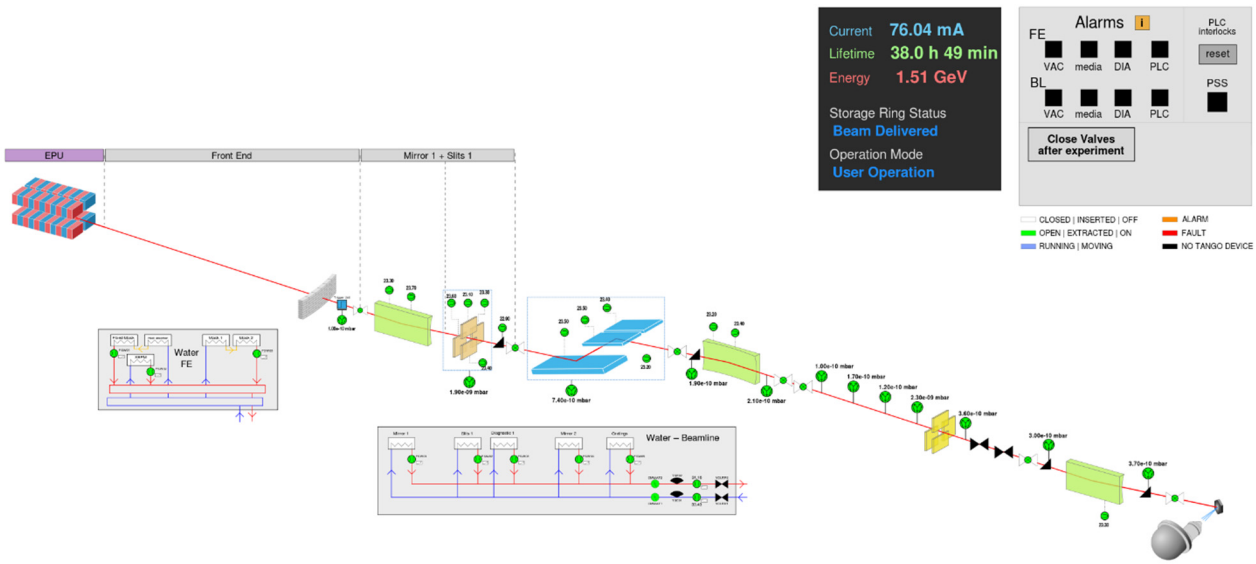

Figure 8: Synoptic application of the beamline being commissioned. It is based on a new view pattern for beamlines.

Secondly, view pattern for beamlines and frontends has been created. It has isometric perspective, colours of elements and panels became more toned and their palette was limited, and some models have been updated (example is shown on Fig. 8). For now, it is the standard for all new beamlines and there are planned future works to update existing synoptics of two operational beamlines.

Thirdly, a Python package was prepared as an extension of svgsynoptic. It defines actions, commands, controls and structure of additional panels and widgets in SOLARIS’ synoptics. Because all synoptic applications import this package and use its classes and methods, it is easy to maintain and develop new features for synoptics without necessity of editing every project.

MAIN FEATURES OF SYNOPTICS IN SOLARIS

Main features of synoptics in SOLARIS are presented in this section.

Executing Tango commands

Executing Tango commands is realised by adding command name and parenthesis at the end of the Tango device name set as a model inside the SVG file. An example of defined command is “model=vac/valve/Open()”. It can also have some parameters, but for now they are static and user cannot insert any other. Execution of command is invoked with the left mouse button and confirmation popup is showed. Example of command confirmation popup is shown in Fig. 9.

Figure 9: Confirmation popup for command execution.

ISBN: 978-3-95450-209-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2019-WEDPR03

Device Class Commands

There can be available commands defined available for specific Tango device class. When appropriate device is clicked with the left mouse button, popup (Fig. 10) shows for some period of time allowing to run specific commands. Shown commands depends on the state of the device. Default device classes with specific commands are valves and ion pumps.

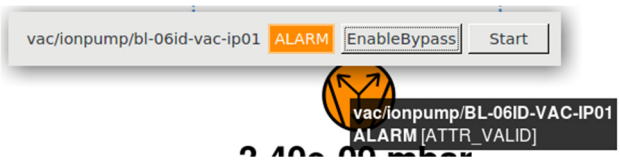

Figure 10: Example of device class commands.

Attributes Widget

In synoptics it is possible to define popups where user can set values of attributes of the devices. Attributes are grouped so clicking on any of the attributes from the group shows widget with all attributes from that group. As an edit widget TaurusWheelEdit class is used, which has two edit modes. Example of this popup is shown in Fig. 11.

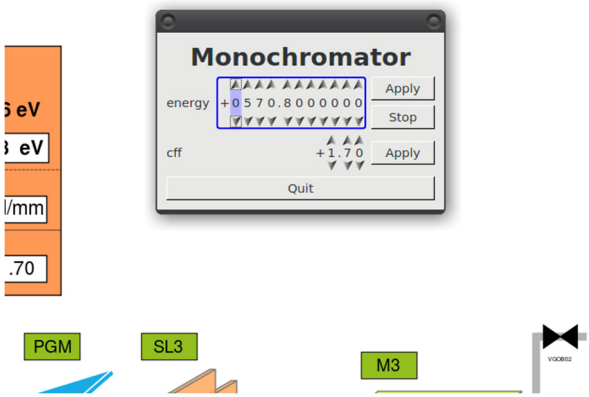

Figure 11: Example of attributes widget popup.

External Applications

It is also possible to open external applications. Each application is connected to a particular device and clicking on it runs application as a separate process.

Additional Elements

In synoptics there can be found buttons for resetting interlocks, opening and closing valves or frontends, setting position of mirrors etc. Mostly, their usage is based on Tango commands provided by facadedevices aggregating other Tango devices. There are also some symbols displaying states which are not the representation of any particular hardware but rather subsystems. They are also based on facadedevices.

CONCLUSION AND FUTURE PLANS

Selection of tools was crucial for developing readable, easy to configure and functional synoptics. This paper presents synoptic GUIs in NSRC SOLARIS based on svgsynoptic library. This library allows for fast creation of clear and intuitive synoptics. Moreover, it allows to control system through the synoptics, which brings great functionality and facilitation for beamline scientists and operators of accelerators.

Graphical representation of the systems in SOLARIS with SVG synoptics was gladly received. On the one hand, they helped reduce time needed to create such representation. On the other hand, users' expectations and demand for new functionalities and synoptics have increased greatly.

Next steps are to replace synoptics of the operational beamlines to the newest standard, improve and add new tools to control devices, and visualise the entire storage ring system with svgsynoptic.

Further plans for synoptics are introduction of possibility to modify values of attributes directly on the synoptic views, and animations improving readability of the actual state of the systems.

ACKNOWLEDGEMENTS

The author would like to express special thanks of gratitude to the current and the former team of NSRC SOLARIS and MAX IV Laboratory for help and suggestions.

REFERENCES

[1] Taurus Scada, https://taurus-scada.org/

[2] F. Poncet, J.L. Pons, 10th ICALEPCS Int. Conf. on Accelerator, & Large Expt. Physics Control Systems. Geneva, 10 - 14 Oct 2005, FR2.1-6O (2005).

[3] J. Forsberg, V. H. Hardion, and D. P. Spruce, "A Graphical Tool for Viewing and Interacting with a Control System", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, pp. 681-684. doi:10.18429/JACoW-ICALEPCS2015-WEM309

[4] svgsynoptic2 repository, https://github.com/MaxIV-KitsControls/lib-maxiv-svgsynoptic

[5] Inkscape website, https://inkscape.org/en

[6] facadedevice documentation, https://tango-facadedevice.readthedocs.io/en/latest/