ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV009

WEB GUI DEVELOPMENT AND INTEGRATION IN LIBERA INSTRUMENTATION

D. Bisiach†, M. Cargnelutti, P. Leban, P. Paglovec, L. Rahne, M. Škabar, A.Vigali
Instrumentation Technologies doo, Solkan, Slovenia

Abstract

During the past 5 years, Instrumentation Technologies expanded and added to the embedded OS running on Libera instruments (beam position instrumentation, LLRF) a lot of data access interfaces to allow faster access to the signals retrieved by the instrument. Some of the access interfaces are strictly related to the user environment Machine control system (EPICS/TANGO), and others are related to the user software preferences (Matlab/Python). In the last years, the requirement for easier data streaming was raised to allow easier data access using PC and mobile phones through a Web browser. This paper aims to present the development of the web backend server and the realization of a web frontend capable to process the data retrieved by the instrument. A use-case will be presented, the realization of the Libera Current Meter Web GUI as a first development example of a Web GUI interface for a Libera instrument and the starting point for the Web GUI pipeline integration on other instruments. The HTTP access interface will become in the next years a standard in data access for Libera instrumentation for quick testing/diagnostics and will allow the final user to customize it autonomously.

INTRODUCTION

In the accelerator environment, the data access is usually performed with well-established and standard access using EPICS, TANGO, Labview, and Matlab interfaces. This software does not allow only access to data but also integrates the control system and safety interlocks to prevent damage to the accelerator block unit.

The drawback of this reliable and safe type of implementation is that any new instrument that is placed in the accelerator environment needs to be configured on the server-side and usually this procedure requires time and effort since the connection is not immediate.

The need for a faster way for evaluation and testing of new devices triggered the attention to the Web interfaces that were already developed in-house by the Red Pitaya project [1] that integrates a fast and easy to access interface that can be used for quick data acquisition.

INTERFACE BETWEEN INSTRUMENT AND APPLICATION SOFTWARE

The access to the setting and the data provided by the instrument is allowed by the software structure reported in Fig.1. The lower layer is tightly bonded to the hardware interface and is responsible to communicate at a lower level with the FPGA and the CPU processes. The second layer called Machine Control Interface (MCI) connects all the user interfaces by providing APIs that enable the servers to access the configuration parameters, the status information, and the data acquired by the instrument. The server-side applications expose through the network to the client-side different application protocols: libera-ioc (EPICS), libera-ds (TANGO), libera-telnet (Labiew/Matlab), libera-cli (user access with the bash) and, as a new feature described in this paper, the libera-http interface that enables the Web access.

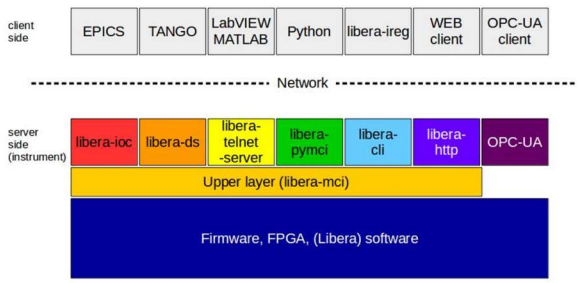

Figure 1: Application stack layer.

Some of these protocols can run in parallel (e.g. EPICS, bash, and HTTP) which makes the troubleshooting of the instrument much easier and more efficient.

HTTP APPLICATION ARCHITECTURE BASED ON REST API

As mentioned in the previous paragraph, access to the instrument can be performed by any device that can access the same network using the HTTP protocol. A typical use case is reported in Fig. 2 where the instrument is accessible using the wired and wireless network:

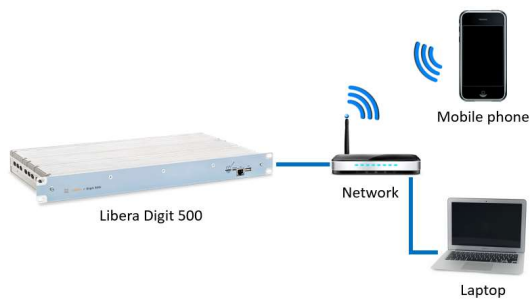

Figure 2: Access to the instrument in a local network.

The system architecture is based on the REpresentational State Transfer (REST) software architectural style and provides the services through Application Programming Interfaces (API) that allow the programmer to easily implement access to the instrument internal interfaces.

The server interface starts during the boot of the instrument and opens a port that is accessible to the other devices

† danilo.bisiach@i-tech.si

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV009

connected to the same network environment. The Libera instrument acts as a server, and any local device acts as a client. The communication API uses the JavaScript Object Notation (JSON) to exchange the data between the server and the client is reported in Fig. 3 and Fig. 4:

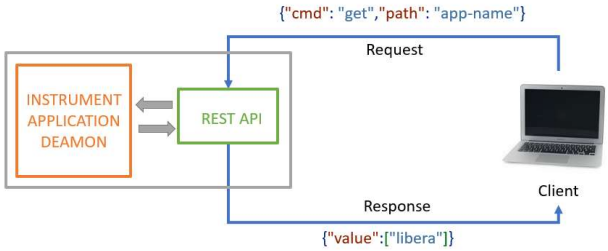

Figure 3: HTTP REST API application architecture.

The requests are performed on the client-side by sending desired parameters in JSON format to the REST API. The server will then return the requested node in JSON format that will be processed by the client and presented in a human-readable format on the WebGUI.

Figure 4: HTTP REST API application architecture node detail.

The integration of the HTTP server on different instruments is almost immediate since the building of the application is fully automated by the existing revisioning and building system. Different instruments, so different software applications, can share the same code for the server application and this feature allows high scalability of the server application.

The only part that needs to be customized development for every new application is the WebGUI code which includes the path to the application nodes. All the settable nodes on the instrument are specified by the node-set labels where the path of a specified node can be accessed as reported in Fig. 5:

The node-set tag is composed by the name we want to give to the variable in the WebGUI, by the path label that is representing the target node in MCI application and allows access to a specific settable variable or a data stream on the instrument, and by the polling-period which specified the refresh time in the WebGUI application.

```
<div class="ireg-node-container">
  <div class="ireg-node-set-name">
ATTENUATION
    </div>
  <tablev>
    <node-set name="Channel 1" path="application.hk.attenuation.Ch1" :pooling-period=1000></node-set>
    <node-set name="Channel 2" path="application.hk.attenuation.Ch2" :pooling-period=1000></node-set>
    <node-set name="Channel 3" path="application.hk.attenuation.Ch3" :pooling-period=1000></node-set>
    <node-set name="Channel 4" path="application.hk.attenuation.Ch4" :pooling-period=1000></node-set>
  </tablev>
</div>
```

Figure 5: Code snippet with the implementation of a node readout from the system.

HTTP APPLICATION ARCHITECTURE BASED ON WEBSOCKET

An additional high-performance interface based on WebSocket technology was integrated into the Libera software.

The interface is compliant with the RFC 6455 and compatible with the HTTP protocol by allowing a full-duplex communication between the client and the server running without the need to re-establish the communication at every request but by keeping it open to facilitate the real-time data transfer and data streaming over TCP. Figure 6 reports the architecture implemented with the WebSocket architecture:

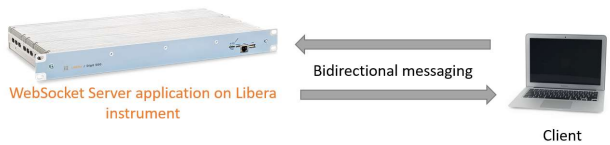

Figure 6: WebSocket application architecture.

The main benefit of this implementation is to allow a maximum performance on data exchange with minimum overhead, and this can make the difference with the REST/API.

It was chosen for the implementation of the 12 Channel Libera Current Meter WebGUI interface which can handle and plot the data acquired from the instrument with minimum latency and high robustness.

EXTENSION AND CUSTOMISATION OF THE HTTP USER INTERFACE: DEVELOPMENT AND INTEGRATION OF A 12 CHANNEL CURRENT METER

One of the main benefits of the HTTP implementation is to allow the final user to customize and integrate the application.

The extensions of the WebGUI can consist of the simple addition of buttons/graphical effects or also of some more consistent improvements. The realization of the 12 Channel Current Meter GUI interface was a very extensive customization of the user software that consisted in the implementation of really demanding features listed below:

1. Capability to perform calculations on the data acquired such as mean values and standard deviation for the acquired signals.
2. Continuous backup of the acquired data in .csv file format.

ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV009

3. Integrate the data acquisition from 3 independent 4-channel acquisition instruments as a 12 channel acquisition instrument.
4. The ability of the WebGUI to run smoothly for 1-month operation, without any loss of data and by keeping the measurement going during the operation.

These features required a high number of additions, testing, and revision of the actual HTTP server implementation. Due to these features, the decision was to implement all the communication based on WebSockets.

All the 3 integrated instruments were acting as an independent device with 12 channels by maintaining a constant synchronization between the 12 channels. The implemented architecture is reported in Fig. 7 where one 4-channel Libera Current meter unit is acting as the master unit which includes the HTTP server on which the client will access the acquired data. The two other 4-channel Libera Current meter are synchronized with the master unit and continuously provide the acquired data stream:

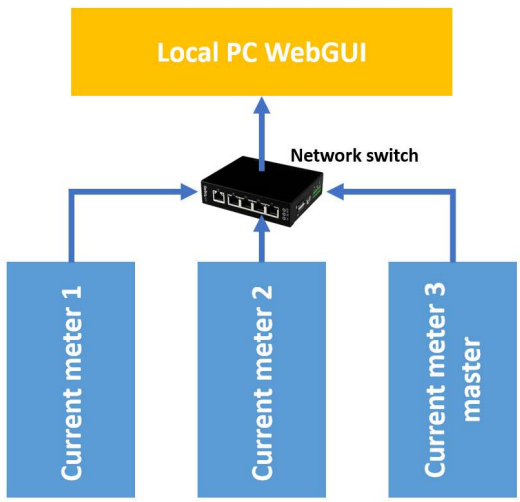

Figure 7: 12 Channel Libera Current meter integration. 3 independent instruments are acting as one on the WebGUI interface side.

The WebGUI screenshot of Fig. 8 reports the final result of the implementation.

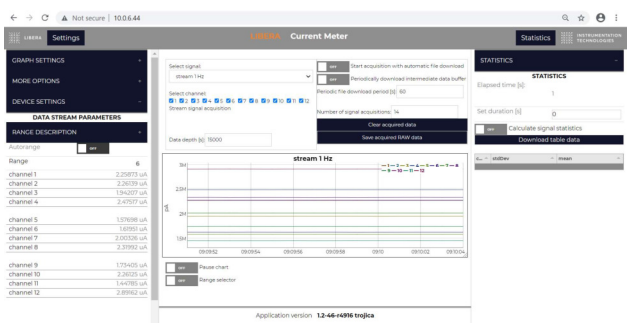

Figure 8: 12-Channel Libera Current meter integration WebGUI.

The most critical point was the testing of the user interface by checking the ability to run smoothly for one month time. These tests were performed by adding to the application software many additional data exchange requests and to stress the memory consumption of the Web browser. The interface was demonstrated to work smoothly for 30 consecutive days with this additional workload on Microsoft Edge (based on the Chromium engine). The main parameters were constantly monitored using the Web browser diagnostics features such as the task manager and the performance options in the Web browser developer mode. The use of the WebSocket architecture allowed the interface to run smoothly even if a lot of data were continuously accumulated and plotted.

As a first experience and proof of concept about the use of WebGUI interfaces based on Javascript for instrument data acquisition, the decision is to integrate the Selenium plugin for Python into the future projects to automate the GUI regression tests and scalability during the QC operations.

RESULTS AND CONCLUSIONS

The development of an HTTP REST API and WebSocket interfaces required a lot of effort in terms of system implementation, integration, testing, and debugging. One of the main benefits is that such a system can be easily accessed by any final user without the need for any control system software (EPICS, TANGO) or proprietary software (Matlab, Labview) already running on the data acquisition system. These features made it very reliable and the first choice of use during system troubleshooting and installation.

The other benefit was to introduce high scalability of the interface: many instruments can benefit from this implementation since the HTTP server is available as a quick add-on for all the Libera instruments with minimal effort dedicated to porting. The performance of the WebSockets can make a difference when a large amount of data needs to be retrieved from the instrument.

Another future implementation within the HTTP or WebSocket can be the GRPC [2] Node-based implementation, which will unify high performance in an easy-to-use platform.

As an ending note, we can confirm that the developed HTTP REST API and WebSocket interface has now became one of the first choice data access interfaces for the testing of the instruments.

ONLINE SOURCE

[1] RedPitaya website, https://redpitaya.com/
[2] GRPC framework, https://grpc.io/