PAUL SCHERRER INSTITUT

PSI

Simon Rees :: Software Engineer :: GFA Controls Section :: Paul Scherrer Institute

A quick look at WICA: an *extra* tool in our Control System Toolkit

SLS HLA Workshop: Jan 19, 2024

PAUL SCHERRER INSTITUT

Introduction

At PSI we are mostly using traditional, non-web, technologies

A few of us both here at PSI, and in other science labs, have been working on projects using web technology. The use of any new technology is always a trade-off. There are rarely any magic bullets.

A few years ago, I was asked to work on a project to make the **facility status displays** available **externally on the web**, so I got a chance to look at some of those trade-offs.

The result of that work was a software product called **WICA** which enables us to bring live EPICS control system data from any of our facilities to a user's web page, on a variety of mobile devices, either at PSI, or off site.

My goal for this workshop is **not** to spend time debating the merits or otherwise of the web but to get you to try out **WICA** for yourselves so you can decide whether there are some situations where you might find it useful.

What is WICA ?

WICA stands for *Web Interface for Controls Applications*

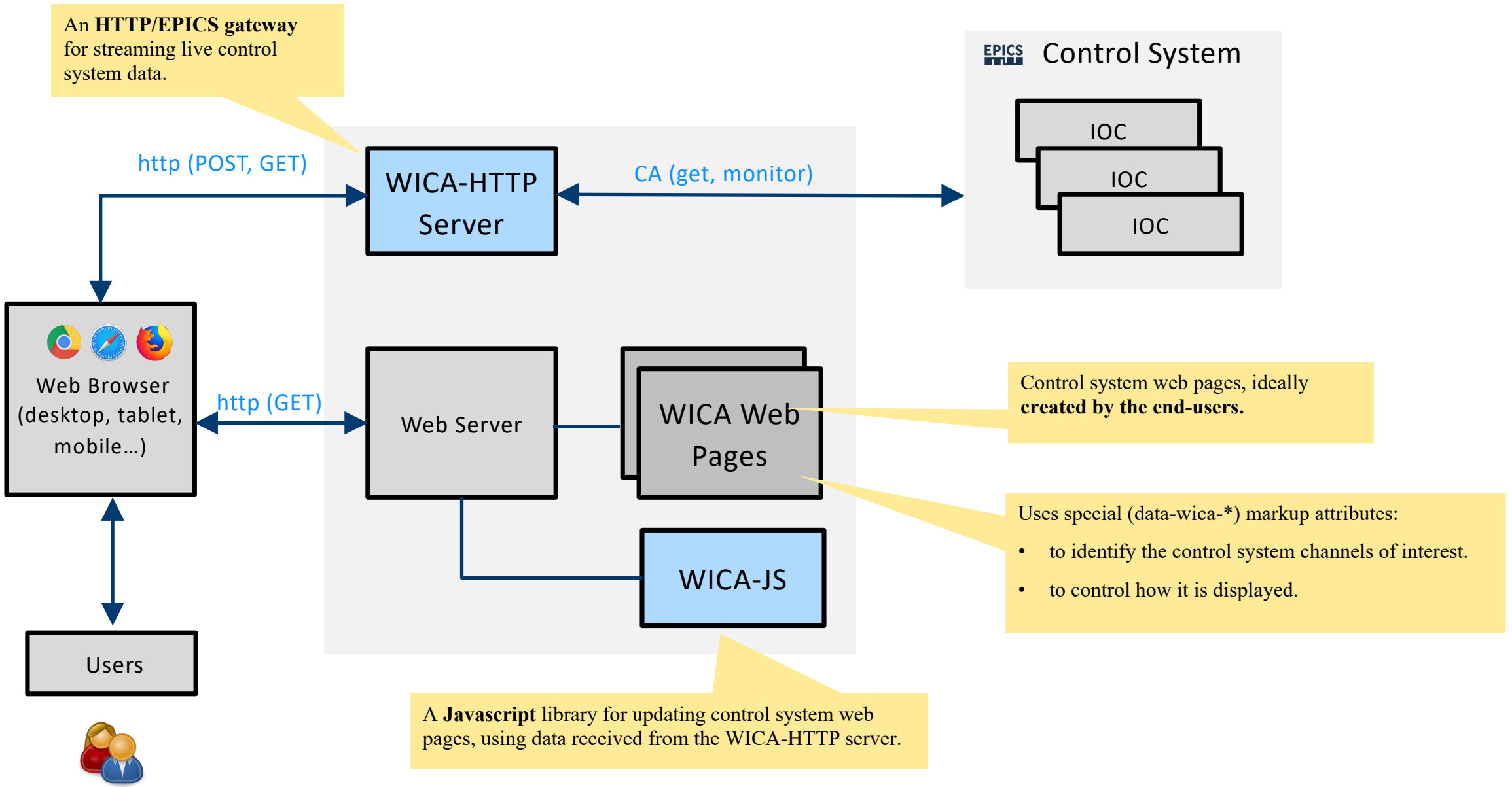
The basic idea is to support the streaming of live EPICS control system data to update a user's web pages in real-time.

There are two main components:

- **Wica-HTTP** - this is a gateway which can stream EPICS channel-access data over the web.

- **Wica-JS** - this is a Javascript library which you include on your web page to get hold of the data and to render it.

WICA Overview Picture

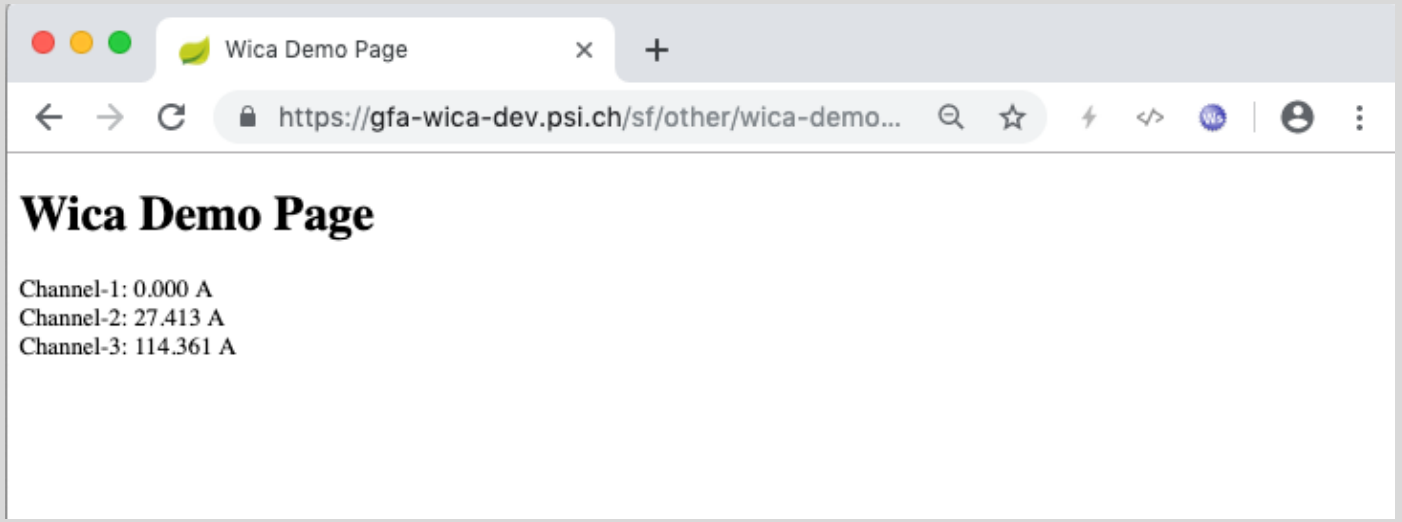


The simplest Wica Web Page

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8"/>
    <title>Wica Demo Page</title>
    <script src="/wica/wica.js" type="module"></script>
</head>
<body>
    <h1>Wica Demo Page</h1>
    <label>Channel-1:</label> <span data-wica-channel-name="SINEG01-MBND300:I-READ"></span> <br>
    <label>Channel-2:</label> <span data-wica-channel-name="SINLH02-MBND100:I-READ"></span> <br>
    <label>Channel-3:</label> <span data-wica-channel-name="SINBC02-MBND100:I-READ"></span> <br>
</body>
</html>
```

Loads the Wica-JS library

Identify the EPICS channels



WICA-Http and Wica-JS are on GitHub

https://github.com/paulscherrerinstitute/wica-http

README GPL-3.0 license

Overview

Wica-HTTP CI passing | pages-build-deployment passing | codecov 71% | release v1.14.0 | dockerhub v1.14.0

This is the **Wica-HTTP** git repository, one component of PSI's WICA software suite.

WICA stands for *Web Interface for Controls Applications*. The basic idea is to support the streaming of live data from a distributed control system to update a user's web pages in real-time.

Wica comprises two main components:

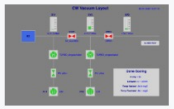
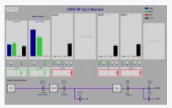
- Wica-HTTP - this is a backend HTTP server which receives incoming requests from the web and which generates live data streams containing information for the control system points of interest.
- Wica-JS - this is a frontend Javascript library which scans a user's web page for HTML5 tags defining points of interest in the control system. The library then generates requests to the backend server to obtain the necessary data and to update the user's web pages in real-time.

Further details about how these components interoperate is provided in the how it works documentation.

Currently WICA interoperates with the EPICS Control Systems using its well established Channel Access (CA) protocol and by means of the Java Channel Access Client Library.

Examples

The screenshots below show examples of the types of panels that can be created using WICA. The examples webpages were created using a mixture of standard HTML, SVG, and in some cases an open-source plotting library (currently Plotly).

Medical Cyclotron	Vacuum Layout	RF Status Page	Camera Real Time Image
			

https://github.com/paulscherrerinstitute/wica-js

README GPL-3.0 license

Overview

release v1.5.4 | dockerhub v1.5.4

This is the **Wica-JS** git repository, one component of PSI's WICA software suite.

WICA stands for *Web Interface for Controls Applications*. The basic idea is to support the streaming of live data from a distributed control system to update a user's web pages in real-time.

Wica comprises two main components:

- Wica-HTTP - this is a backend HTTP server which receives incoming requests from the web and which generates live data streams containing information for the control system points of interest.
- Wica-JS - this is a frontend Javascript library which scans a user's web page for HTML5 tags defining points of interest in the control system. The library then generates requests to the backend server to obtain the necessary data and to update the user's web pages in real-time.

Further details about how these components interoperate is provided in the how it works documentation.

Main Features

- Collaborates with Wica-HTTP backend server to stream control system data to update user web pages.
- Supports configuration of streaming options including backend data acquisition modes (poll or monitor) and channel filtering (eg noise or rate limiting).
- Supports all modern web browsers including Chrome, Firefox, Safari and Edge.
- Supports all current web platforms including desktop, tablet and mobile.
- Supports HTML element text rendering (including visualisation of alarm and connection status).
- Supports JS event generation to enable custom calculations or rendering.
- Implemented as Javascript ES6-module with few external dependencies.
- Tooling and dependency management handled by Node.Js/NPM.

Workshop Exercises

Let’s make a start…

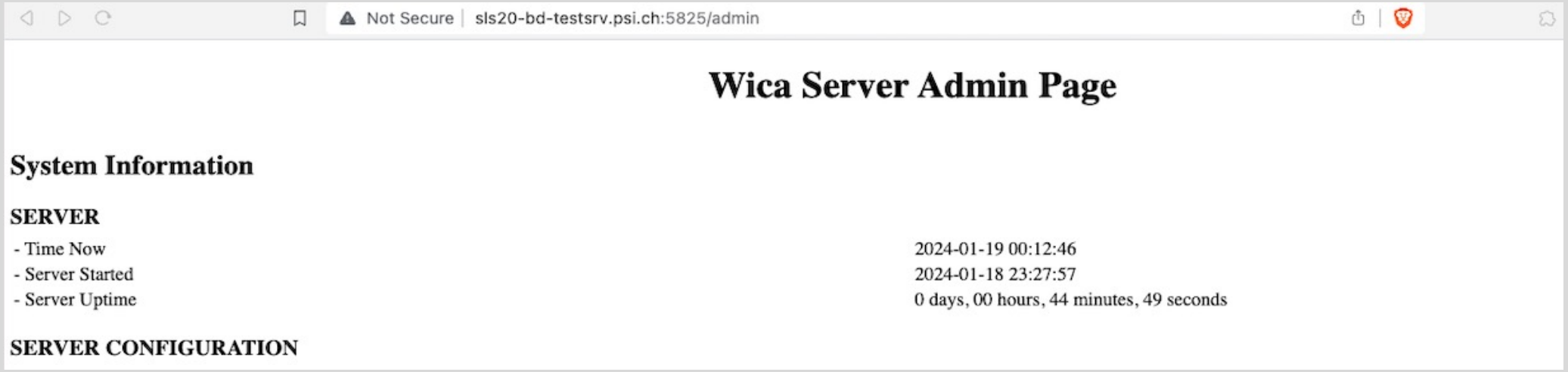
Exercise 1: Download, and run the Wica-HTTP server locally

⏰ 5 mins

Follow the instructions at the following URL to download and run the Wica-Http server in your own user account and unique port on the HLA test machine

- `https://github.com/paulscherrerinstitute/wica-http/blob/master/docs/hla/ex1.md`

The server admin page should look like this:

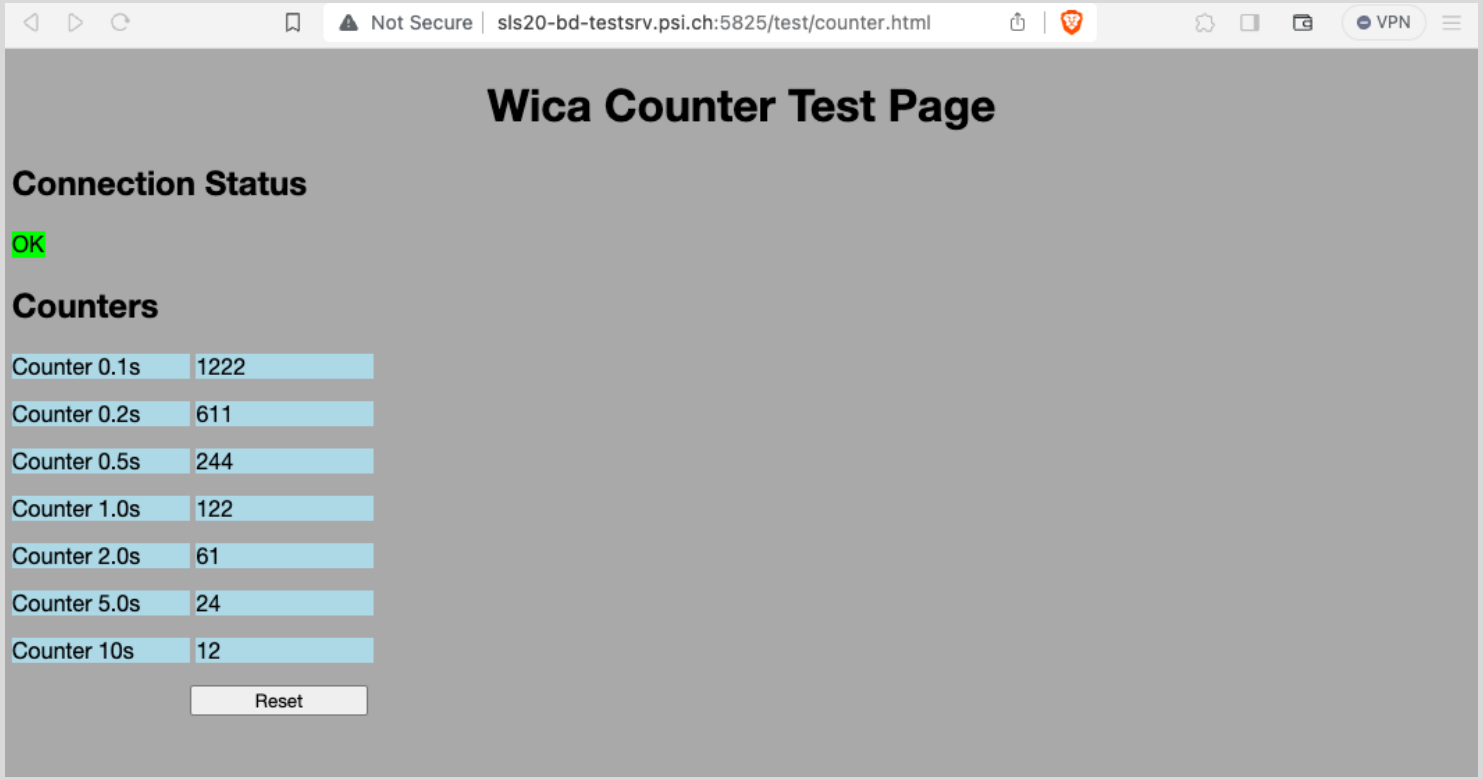


Exercise 2: I will run the EPICS ‘counter.db’ test database on the local machine. See if your local server instance can connect to it.

⏰ 5 mins

Navigate to the following URL, but swap the port number of your local server for the one below

- **`http://sls20-bd-testsrv.psi.ch:5825/test/counter.html`**



The ‘counter.db’ database looks like this

```
record( calc, "wica:test:counter01" ) {
  field( SCAN,".1 second")
  field( CALC,"VAL+1")
}
record( calc, "wica:test:counter02" ) {
  field( SCAN,".2 second")
  field( CALC,"VAL+1")
}
record( calc, "wica:test:counter03" ) {
  field( SCAN,".5 second")
  field( CALC,"VAL+1")
}
record( calc, "wica:test:counter04" ) {
  field( SCAN,"1 second")
  field( CALC,"VAL+1")
}
```

```
record( calc, "wica:test:counter05" ) {
  field( SCAN,"2 second")
  field( CALC,"VAL+1")
}
record( calc, "wica:test:counter06" ) {
  field( SCAN,"5 second")
  field( CALC,"VAL+1")
}

record( calc, "wica:test:counter07" ) {
  field( SCAN,"10 second")
  field( CALC,"VAL+1")
}
```

Provides a series of counter channels running at different rates…

The ‘counter.html’ web page looks like this

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8"/>
    <meta name="viewport" content="width=device-width, initial-scale=1"/>
    <title>Wica Test Counter</title>
    <!-- Load the Wica library from the local machine -->
    <script src="../wica/wica.js" type="module"></script>

<style>
<!-- add CSS styling to web page here -->
</style>
</head>

<body>
<div>
    <h2>Counters</h2>
    <label>Counter 0.1s </label> <label data-wica-channel-name="wica:test:counter01">NC</label><br>
    <label>Counter 0.2s </label> <label data-wica-channel-name="wica:test:counter02">NC</label><br>
    <label>Counter 0.5s </label> <label data-wica-channel-name="wica:test:counter03">NC</label><br>
    <label>Counter 1.0s </label> <label data-wica-channel-name="wica:test:counter04">NC</label><br>
    <label>Counter 2.0s </label> <label data-wica-channel-name="wica:test:counter05">NC</label><br>
    <label>Counter 5.0s </label> <label data-wica-channel-name="wica:test:counter06">NC</label><br>
    <label>Counter 10s  </label> <label data-wica-channel-name="wica:test:counter07">NC</label><br>
    <label></label><button onclick="resetCounters()">Reset</button><br>
</div>

<script>
<!-- add JS “intelligence” to your web page here -->
</script>
</body>
</html>
```

Loads the Wica-JS library

data-wica-channel-name
Defines the EPICS channels you want to display

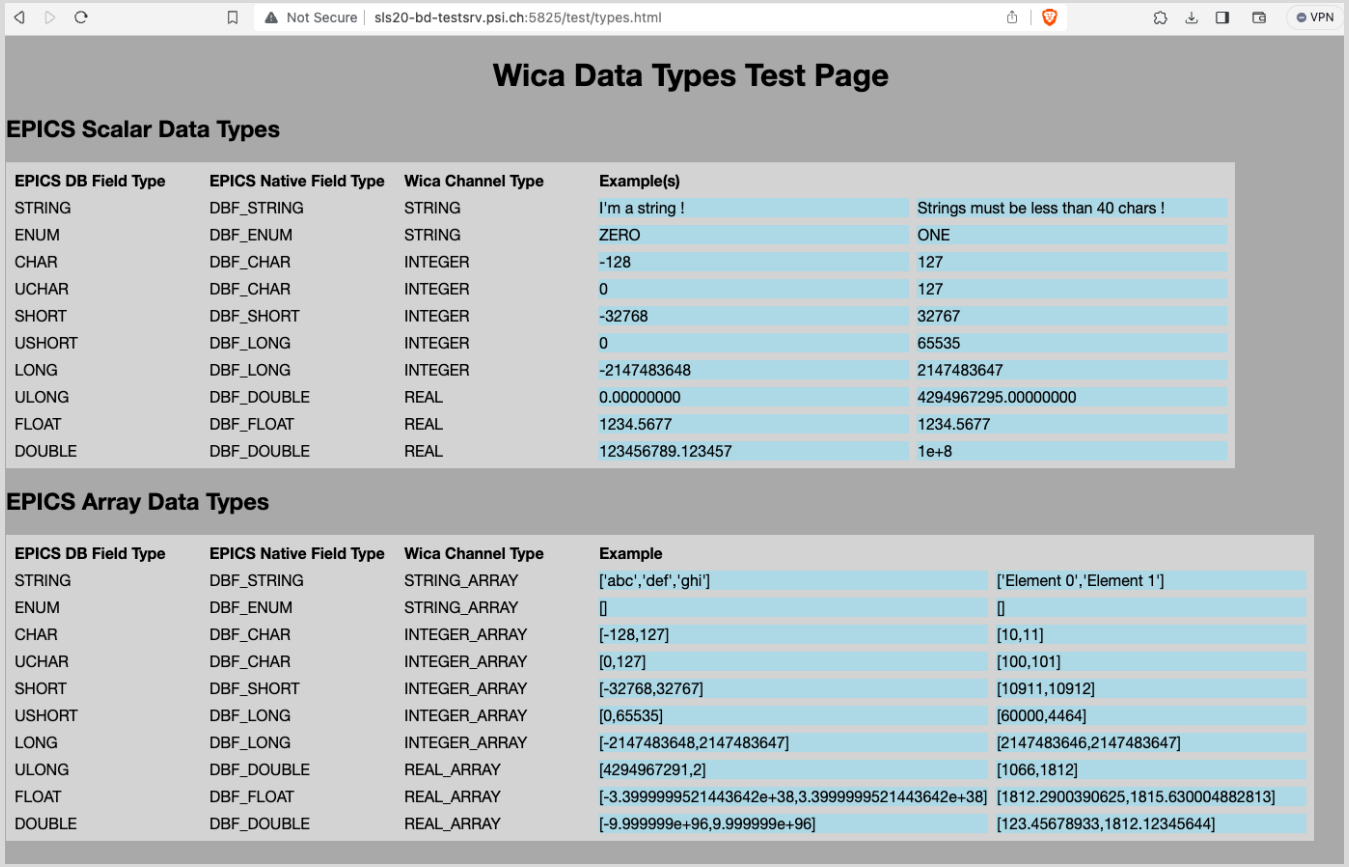
Content gets rendered ‘automagically’ in real-time

Exercise 3: I will run the EPICS ‘types.db’ test database on the local machine. See if your local server instance can connect to it.

⏰ 5 mins

Navigate to the following URL, but swap the port number of your local server for the one below

- **`http://sls20-bd-testsrv.psi.ch:5825/test/types.html`**



The ‘types.db’ database looks like this

```
record( stringout, wica:test:scalar:string1 ) {
  field( VAL, "I'm a string !" )
}
record( stringout, wica:test:scalar:string2 ) {
  field( VAL, "Strings must be less than 40 chars !" )
}
record( bo, wica:test:scalar:enum1 ) {
  field( VAL, "0" )
  field( ZNAM, "ZERO" )
  field( ONAM, "ONE" )
}
record( bo, wica:test:scalar:enum2 ) {
  field( VAL, "1" )
  field( ZNAM, "ZERO" )
  field( ONAM, "ONE" )
}
record( aSub, "wica:test:scalar:integers" ) {
  field( FTA, "CHAR" )
  field( INPA, "-128" )
  field( FTB, "CHAR" )
  field( INPB, "127" )
  field( FTC, "UCHAR" )
  field( INPC, "0" )
  field( FTD, "UCHAR" )
  field( INPD, "127" )
  field( FTE, "SHORT" )
  field( INPE, "-32768" )
  field( FTF, "SHORT" )
  field( INPF, "32767" )
  field( FTG, "USHORT" )
  field( INPG, "0" )
  field( FTH, "USHORT" )
  field( INPH, "65535" )
  field( FTI, "LONG" )
  field( INPI,  "-2147483648" )
  field( FTJ, "LONG" )
  field( INPJ,  "2147483647" )
}
record( aSub, "wica:test:scalar:reals" ) {
  field( FTA, "ULONG" )
  field( INPA,  "0" )
  etc
…
```

Provides EPICS channels of all the standard types

The ‘types.html’ web page looks like this

```
<!doctype html>
<html lang="en">
<head>
<script src="../wica/wica.js" type="module"></script>
</head>

<body>
<table>
 …
<tr><td>FLOAT</td><td>DBF_FLOAT</td><td>REAL</td>
<td data-wica-channel-name="wica:test:scalar:reals.C" data-wica-channel-props='{ "prec": 4}' data-wica-rendering-props ='{ "prec": 4}'>NC</td>
<td data-wica-channel-name="wica:test:scalar:reals.D" data-wica-channel-props='{ "prec": 4}' data-wica-rendering-props ='{ "prec": 4}'>NC</td>
</tr>
<tr><td>DOUBLE</td><td>DBF_DOUBLE</td><td>REAL</td>
<td data-wica-channel-name="wica:test:scalar:reals.E" data-wica-channel-props='{ "prec": 12}' data-wica-rendering-props='{ "prec": 6}'>NC</td>
<td data-wica-channel-name="wica:test:scalar:reals.F" data-wica-channel-props='{ "prec": 12}' data-wica-rendering-props='{ "exp": true }'>NC</td>
</tr>
</table>

<script>
<!-- add JS "intelligen   to your web page here -->
</script>
</body>
</html>
```

data-wica-channel-props
Define the properties used for the data transfer from the server

data-wica-rendering-props
Define the properties used for displaying the data on the web page

data-wica-channel-name
Define the EPICS channels you want to display

Controlling data acquisition: ‘data-wica-channel-props’

WicaChannelProperties

Provides a type definition for a JS Object that specifies the properties of a Wica Channel. When not specified the property values will be taken from the WicaChannelPropertyDefaults.

Properties:

Name	Type	Attributes	Description
daqmode	string | null	<optional>	The data acquisition mode.
pollint	number | null	<optional>	The interval between successive polls of the channel value.
fields	string | null	<optional>	A semicolon delimited list defining the data fields that should be included when sending value information for this channel.
prec	number | null	<optional>	The precision (= number of digits after the decimal point) to be used when sending numeric information.
filter	module:shared-definitions.WicaChannelFilterType	<optional>	The type of filtering to be used on the channel. See WicaChannelFilterType.
n	number	<optional>	The filter number of samples. See WicaChannelFilterTypeLatestValueSampler.
m	number	<optional>	The filter cycle length. See WicaChannelFilterTypeFixedCycleSampler.
interval	number	<optional>	The filter sampling interval. See WicaChannelFilterTypeRateLimitingSampler.
deadband	number	<optional>	The filter deadband. See WicaChannelFilterTypeChangeFilteringSampler.

Supports configuration of data transfer options

- Poll or monitor channels
- Data transfer precision
- channel filtering (eg noise or rate limiting)

Controlling text rendering: ‘data-wica-rendering-props’

WicaTextRenderingProperties

Provides a type definition for a JS Object that specifies the properties to be used when rendering a Wica element's textual content. When not specified the property values will be taken from the WicaTextRenderingPropertyDefaults.

Properties:

Name	Type	Attributes	Description
disable	boolean	<optional>	Disables rendering for this channel.
units	string	<optional>	The units to be displayed when rendering numeric information. When this property is specified it will be used. When not specified an attempt will be made to obtain the units from the metadata.
exp	boolean	<optional>	Sets the rendering format for channels which return numeric data. Possible values: [true: (use exponential format, eg 1.27E-1), false: (use fixed decimal point format, eg 0.127)].
prec	number	<optional>	The precision (= number of digits after the decimal point) to be used for channels which return numeric data.

Supports configuration of how the text is displayed

- precision
- display engineering units
- exponential format

Exercise 4: Explore the operations available on the Wica-Http Server

⏰ 5 mins

Follow the instructions at the following URL:

- `https://github.com/paulscherrerinstitute/wica-http/blob/master/docs/hla/ex4.md`

Use curl to **get** or **set** some EPICS channel values

Use curl to **create** and **subscribe** to a Wica stream

Exercise 5: Read channel values from PSI science facilities

⏰ 5 mins

Navigate a web browser to some of the URL's below:

- PROSCAN - https://gfa-wica.psi.ch/ca/channel/XPROSCAN:TIME:2
- HIPA - https://gfa-wica.psi.ch/ca/channel/XHIPA:TIME
- SwissFEL - https://gfa-wica.psi.ch/ca/channel/SARUN:FELPHOTENE
- Try your own channels…

Exercise 6: A quick look at PSI's WICA main website

⏰ 5 mins

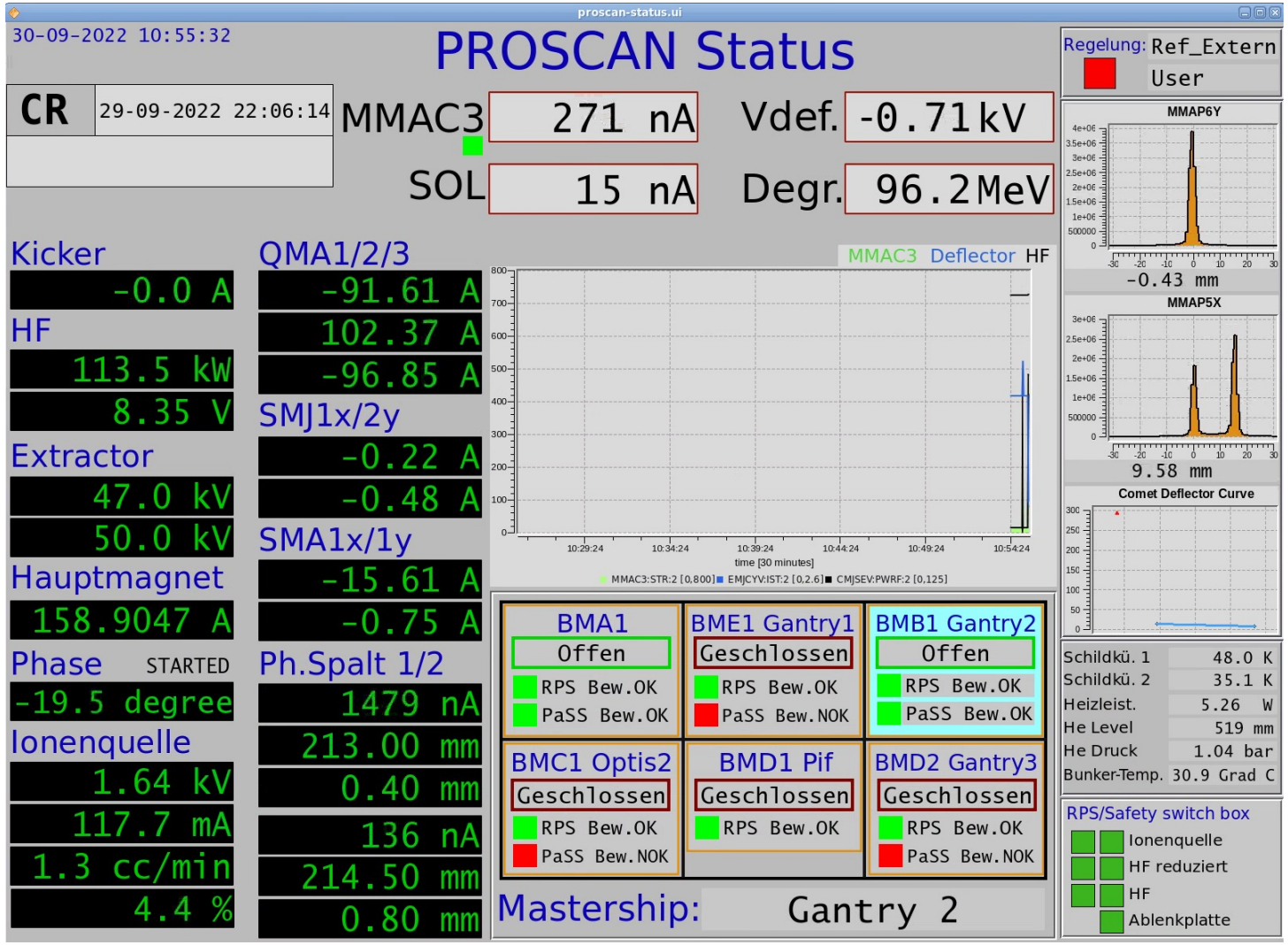
Navigate a web browser here:

- Internal - https://gfa-wica.psi.ch
- External - https://wica.psi.ch

And look around at some of the pages.

Some example pages…

PROSCAN caQtdm Status Page – the gold standard to try to copy

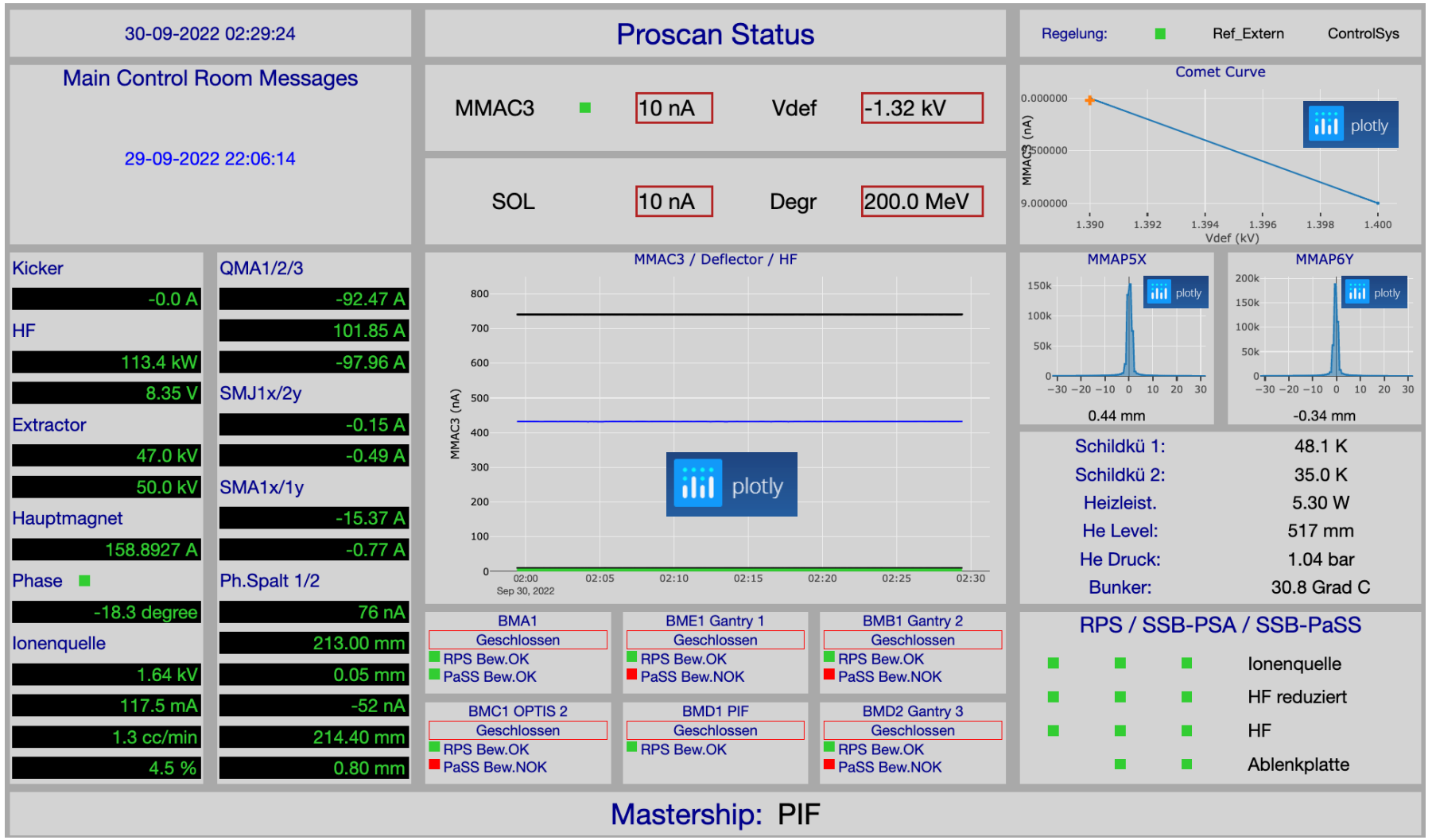


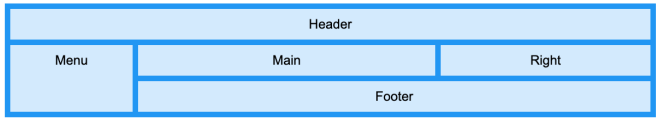
The original display page made with caQtDm

Ugly, but used effectively by many people !

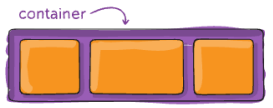
Goal: make something just as ugly, but available on the web

PROSCAN Status Page – Mixed Text & Plot Example



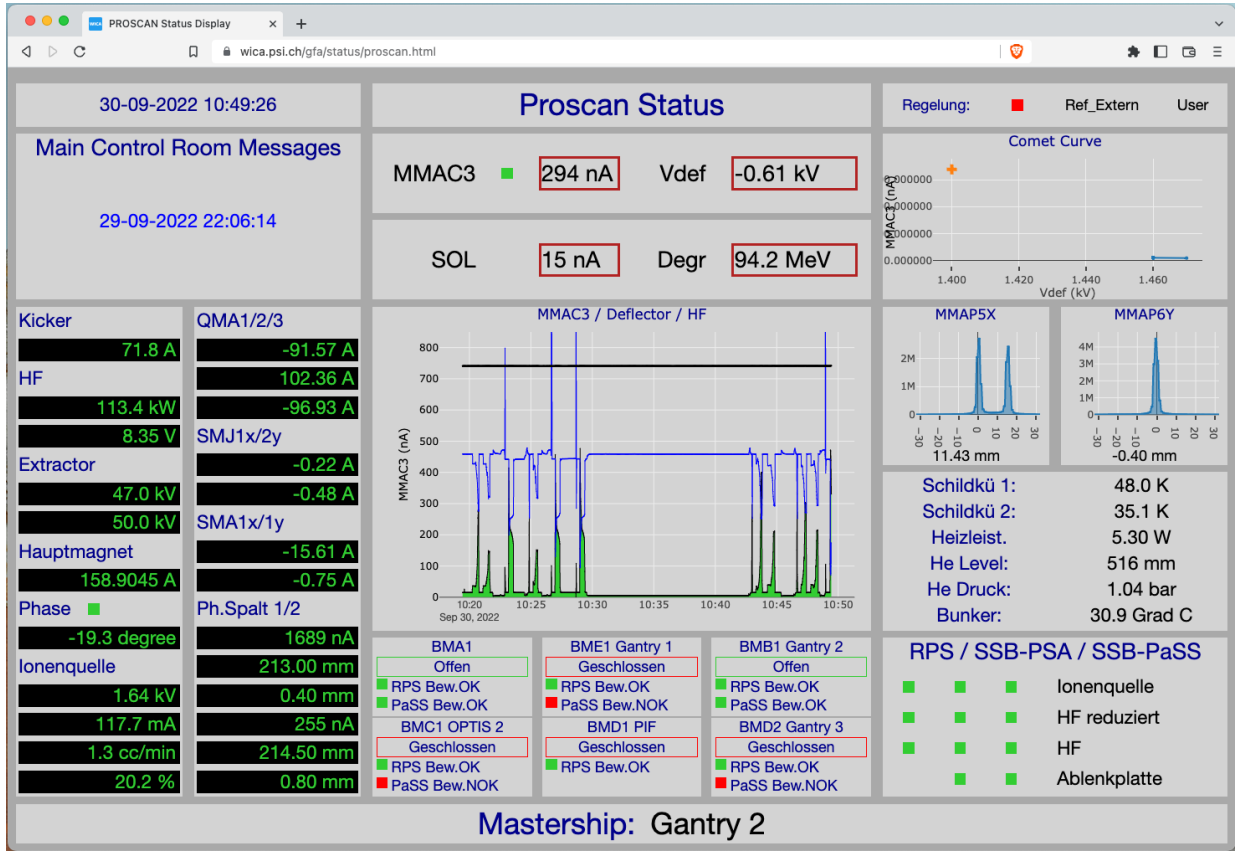


CSS3 Grid

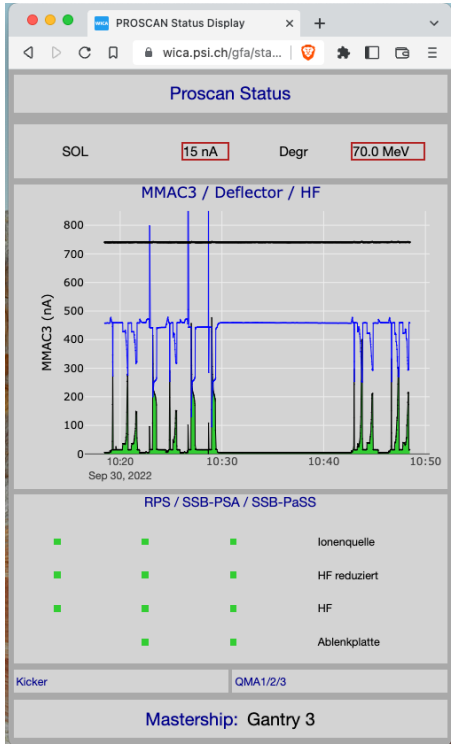


CSS3 Flexbox

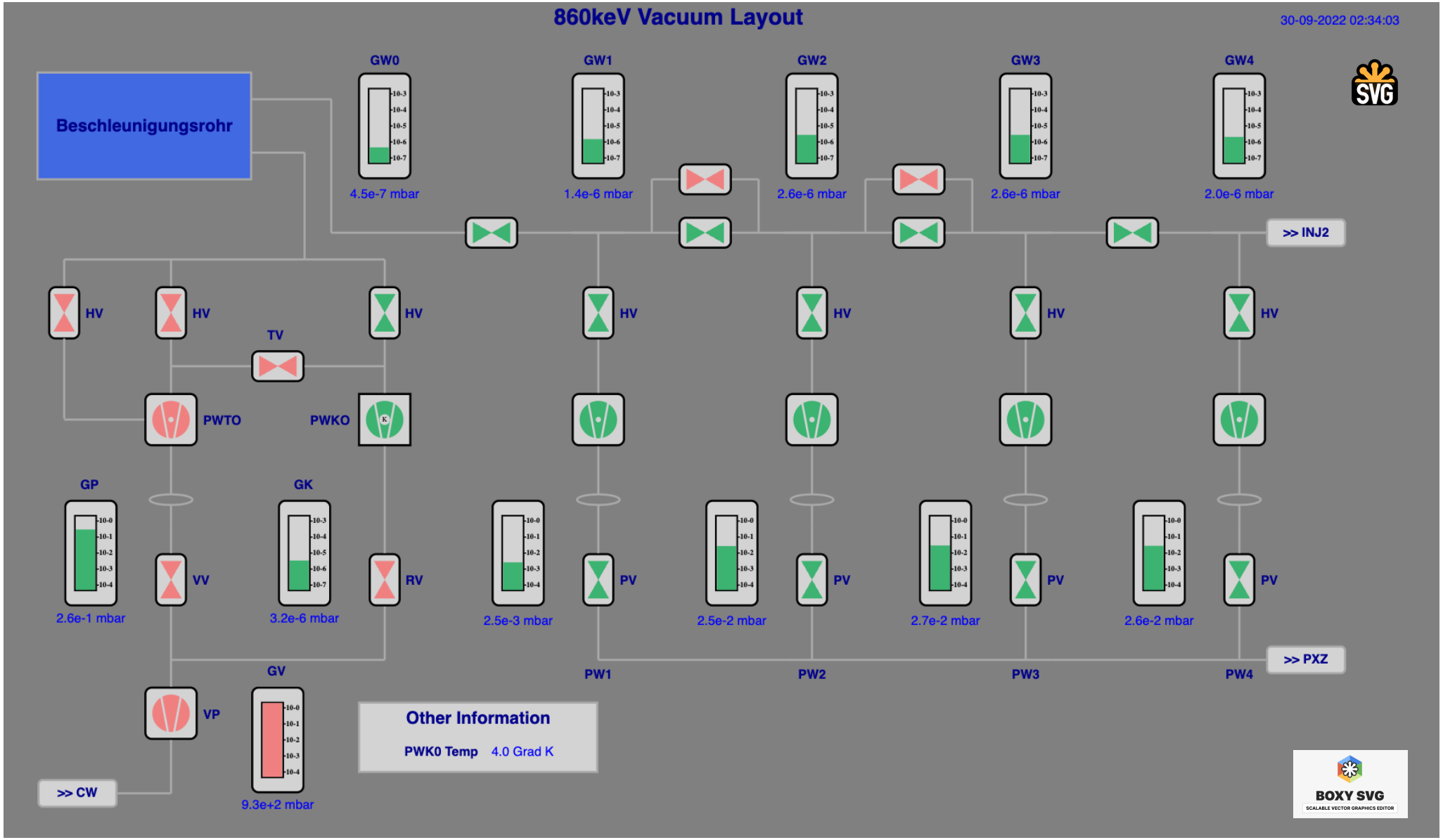
PROSCAN Status Page – responsive design



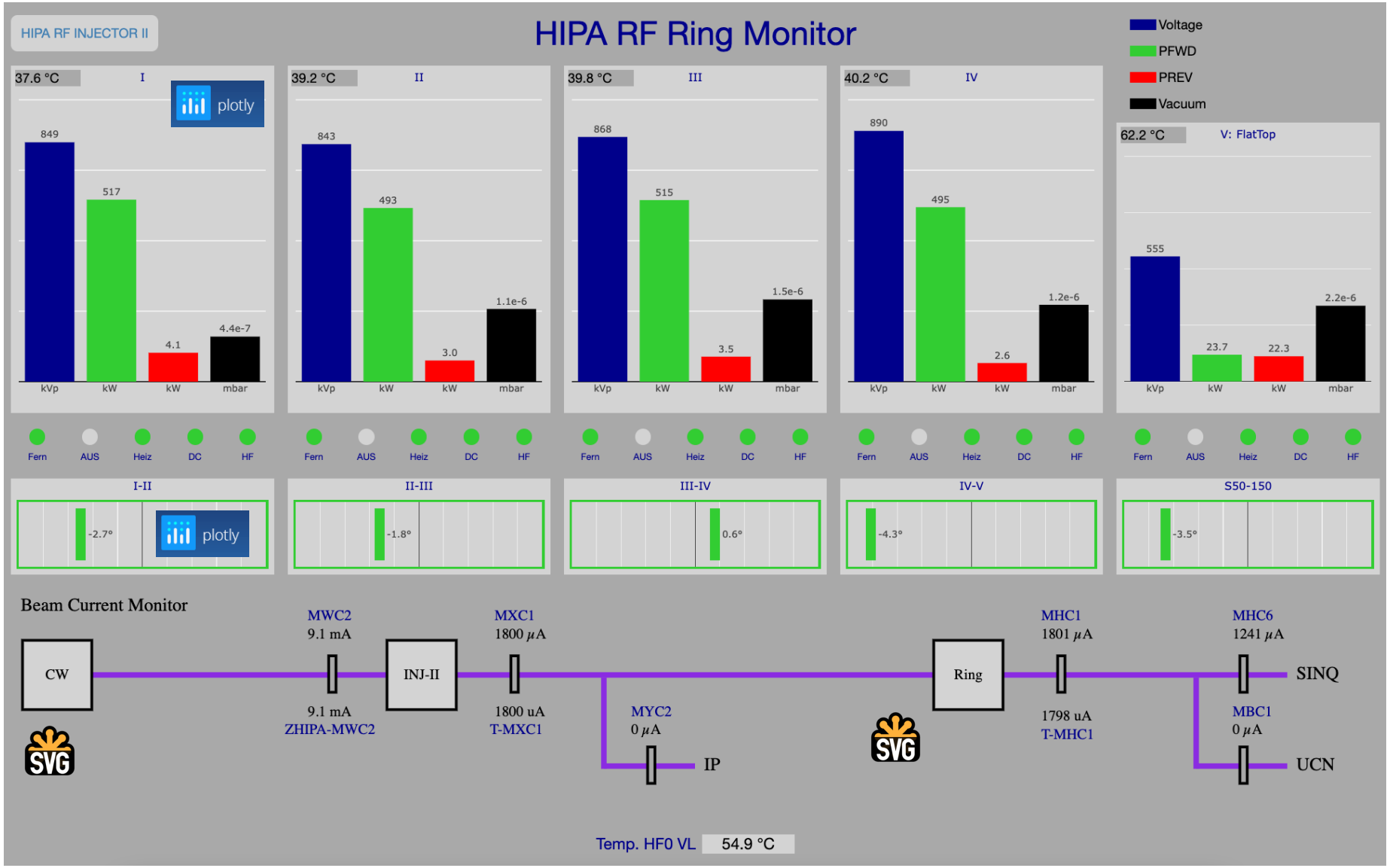
Because the pages just use normal web technology, we can use CSS media queries to change the formats according to the features of the viewing device…



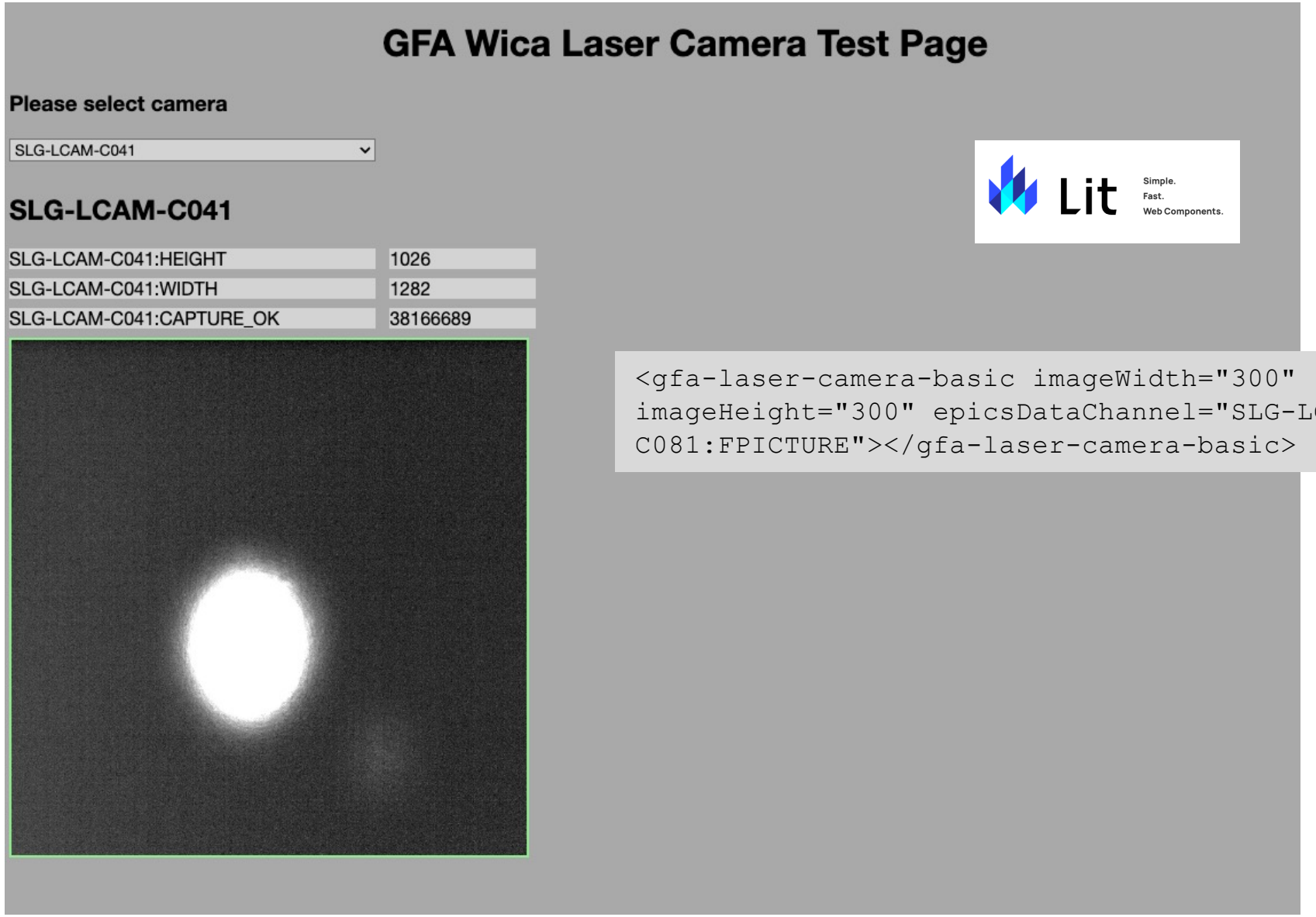
HIPA Vacuum System Page – SVG Example



HIPA RF System Page – SVG + Plot Example



SwissFEL Camera Page – Web Component Example




```
<gfa-laser-camera-basic imageWidth="300"
imageHeight="300" epicsDataChannel="SLG-LCAM-
C081:FPICTURE"></gfa-laser-camera-basic>
```

Using Web Components enable us to create very clean html pages

But creating a web component control system library would require significant effort

Leveraging WICA – possible workflows

Text-only Display Page

Write some html to define your control points

WICA will render the text content of your elements directly - no JS required.

Display Page with Text and Plots

Write some html to define your control points

Choose a JS plot library eg *Plotly* or *Highcharts*

Write some JS to update the plot using the received data

Display Page with Text and SVG

Write some html to define your control points

Use an SVG Editor eg *Boxy-SVG* to create your graphic. Or create graphic with text editor

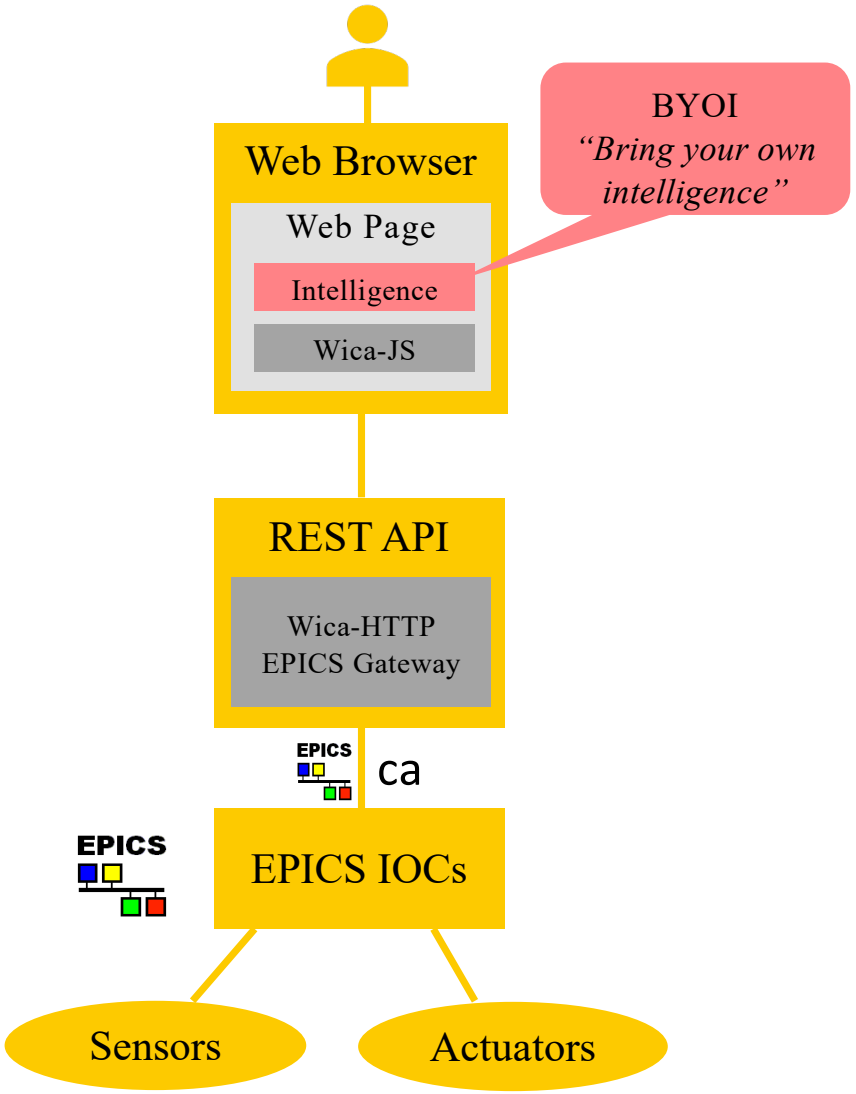
Write some JS to update the graphic using the received data

Display Page with eg Camera Images

Obtain a Web Component that supports rendering camera image data. At PSI we use *Lit*

Write some html to define the control points associated with each web component

WICA Learning Experiences so far – a partial success ?



- WICA is the only display tool that works off-site ☺
- Getting the data to the user's webpage is easy ☺
- It's also easy to make a webpage with text channels that you can look at offsite. ☺
- Currently to make sophisticated webpages with plots, fancy graphics or camera images takes some programming effort. ☹
- Providing a tool or workflow for the users (who are not programmers) has been difficult ☹
- caqtdm is not under threat ! ☺

Thank you for your attention :-)

If you have questions, or would like to find out more, please get in touch.