REACT AUTOMATION STUDIO: A NEW FACE TO CONTROL LARGE SCIENTIFIC EQUIPMENT

William Duckitt

Justin Abraham

science & innovation
Department:
Science and Innovation
REPUBLIC OF SOUTH AFRICA

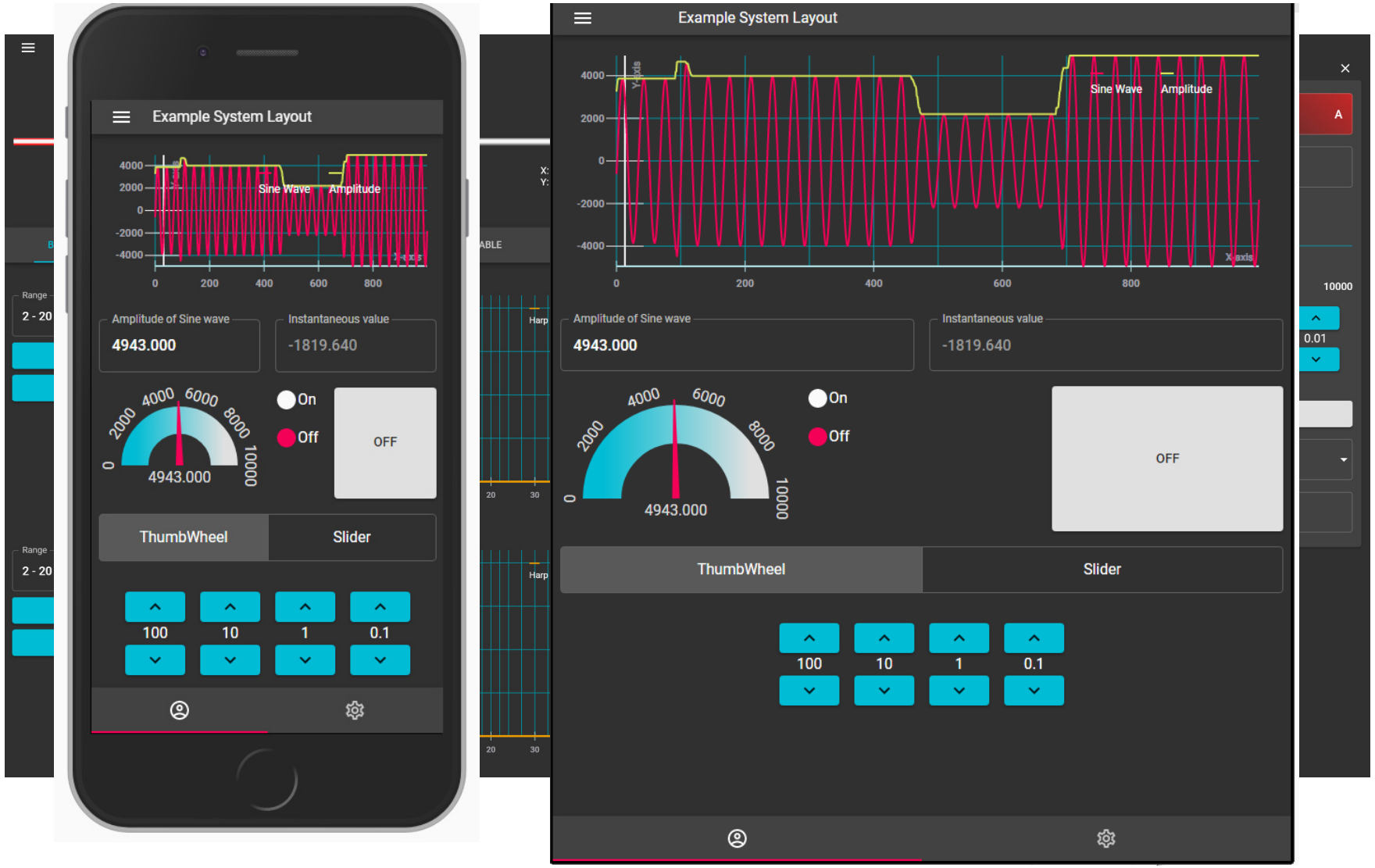
CYC 2019 22-27 SEPTEMBER CAPE TOWN

NRF National Research Foundation 20 1999-2019

Outline

- Introduction to React Automation Studio
- What, why and how we did it
- Examples of web, desktop and mobile UI’s
- Future work and community involvement

React Automation Studio



Progressive web application (PWA) framework to create cross platform and cross device UIs for EPICS

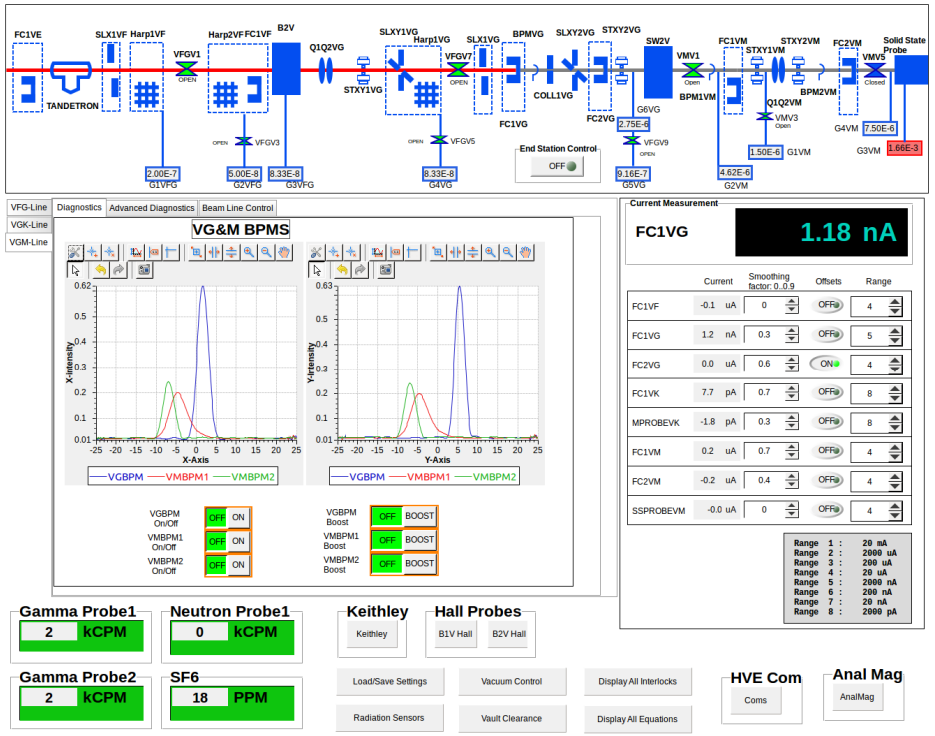
Runs on your desktop

On your mobile

And in the web browser

Evolutionary leap in control system UI design

CS-Studio



React Automation Studio

We used all our experience and knowledge in creating unified EPICS-EtherCAT control systems with centralized and hierarchically structured CS-Studio UI's

Built a system that is leveraged on top the R&D of the worlds biggest corporations and their open-sourced software frameworks

Cross-platform and cross-device

Themeable

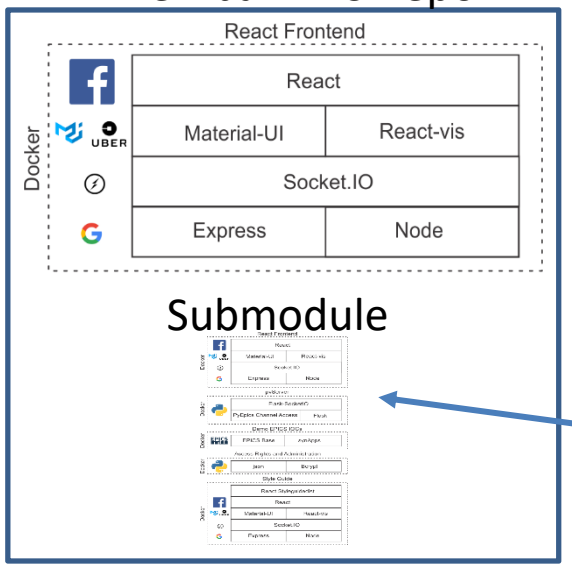
Implements the best features of CS-studio, without its limitations.

Software Stack

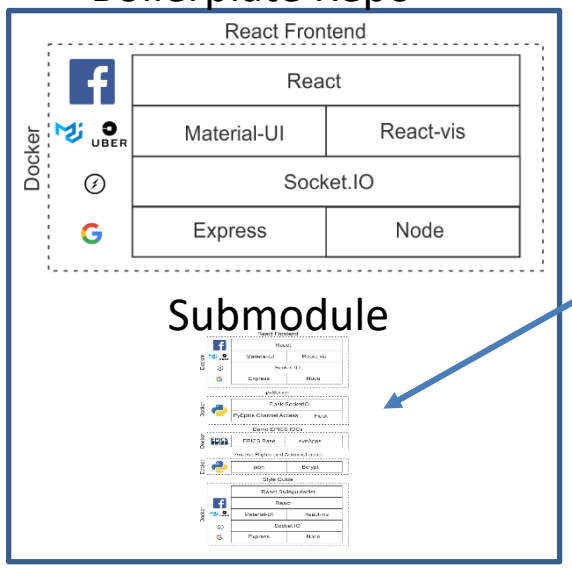
Non-Public

Public Customizable

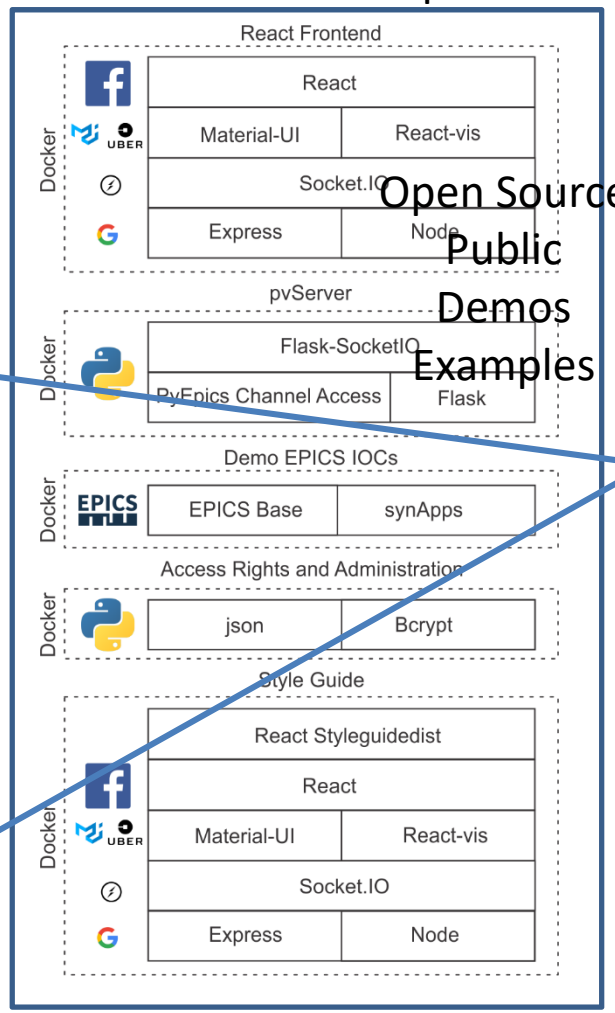
iThemba LABS Repo



Boilerplate Repo



Git Mono Repo

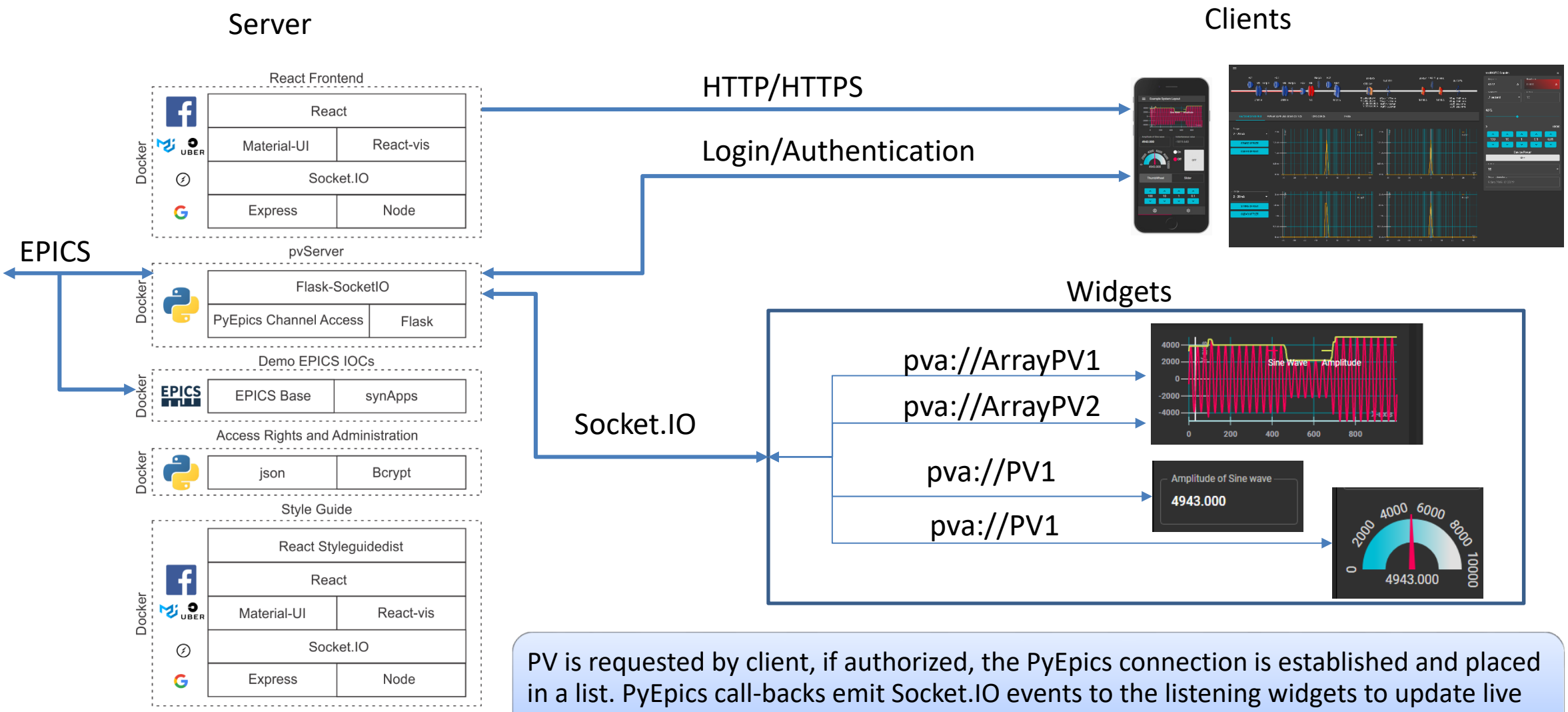


Open Source

Public Demos Examples

- Development and production versions have been containerized with Docker, deployed with one command as micro-services
- Code entry is in React JavaScript, don't need knowledge of HTML or CSS
- Wrappers on Material-UI components and React-vis graphs
- PyEpics based backend
- Socket-IO communication between components and pvServer
- Containerized Demo IOC
- Utilities for user management and access rights administration
- Fully interactive help and style guide

Software Protocol

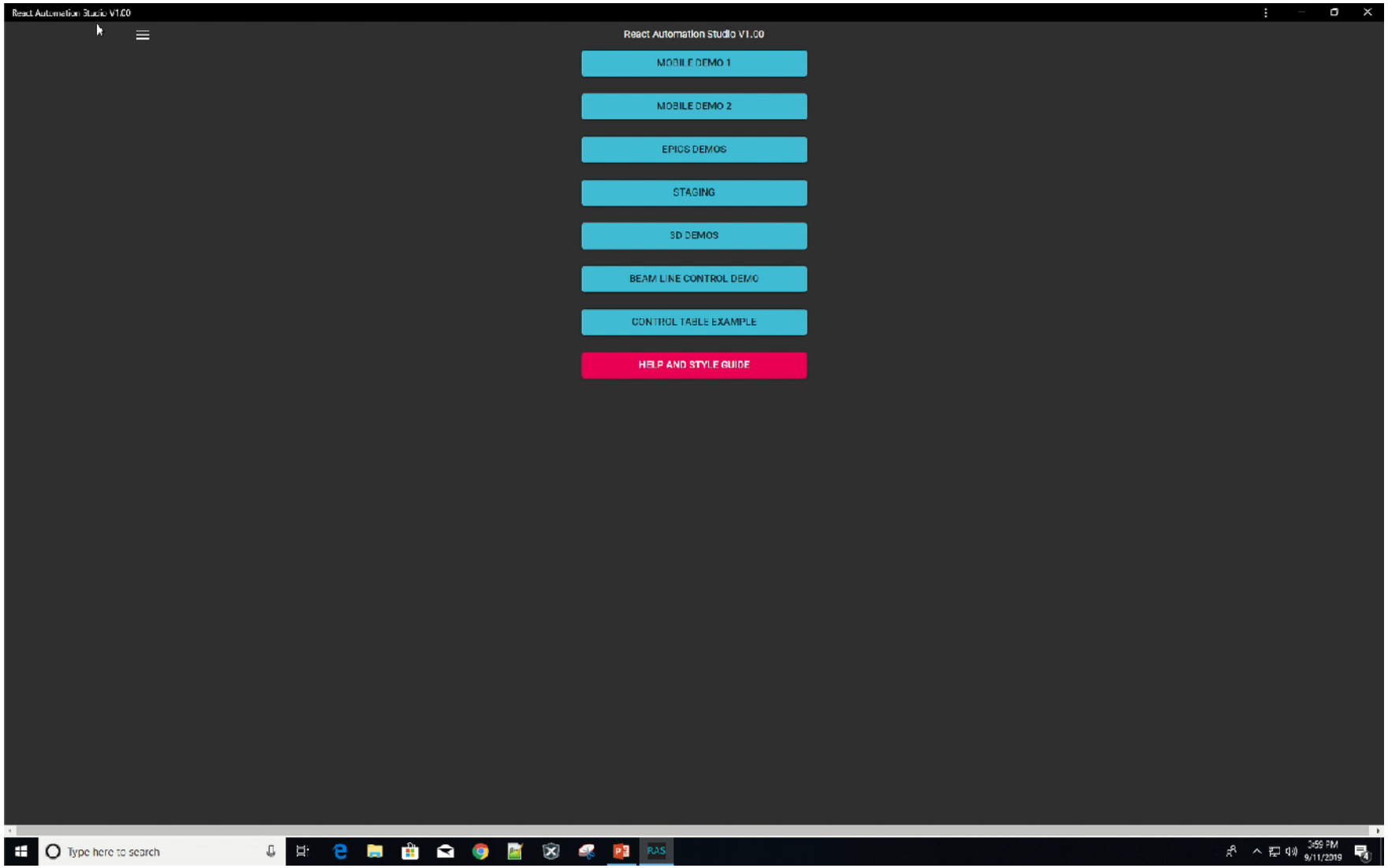


PV is requested by client, if authorized, the PyEpics connection is established and placed in a list. PyEpics call-backs emit Socket.IO events to the listening widgets to update live data, connection status and the meta data.

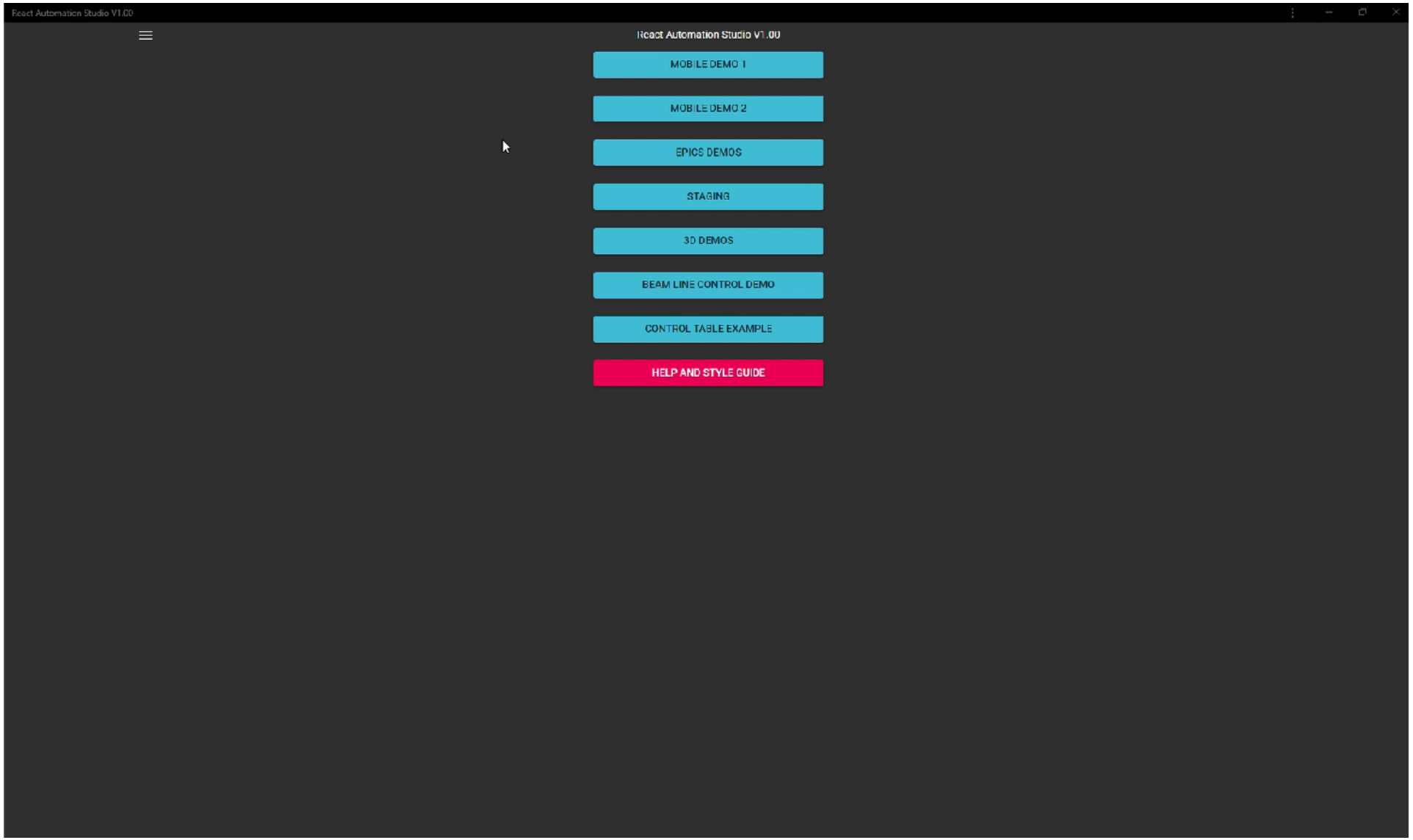
Responsive PWA Demo

- Server running on Linux machine
- Clients on Windows
- Demonstrate PWA installation
- Demonstrate real-time responsiveness

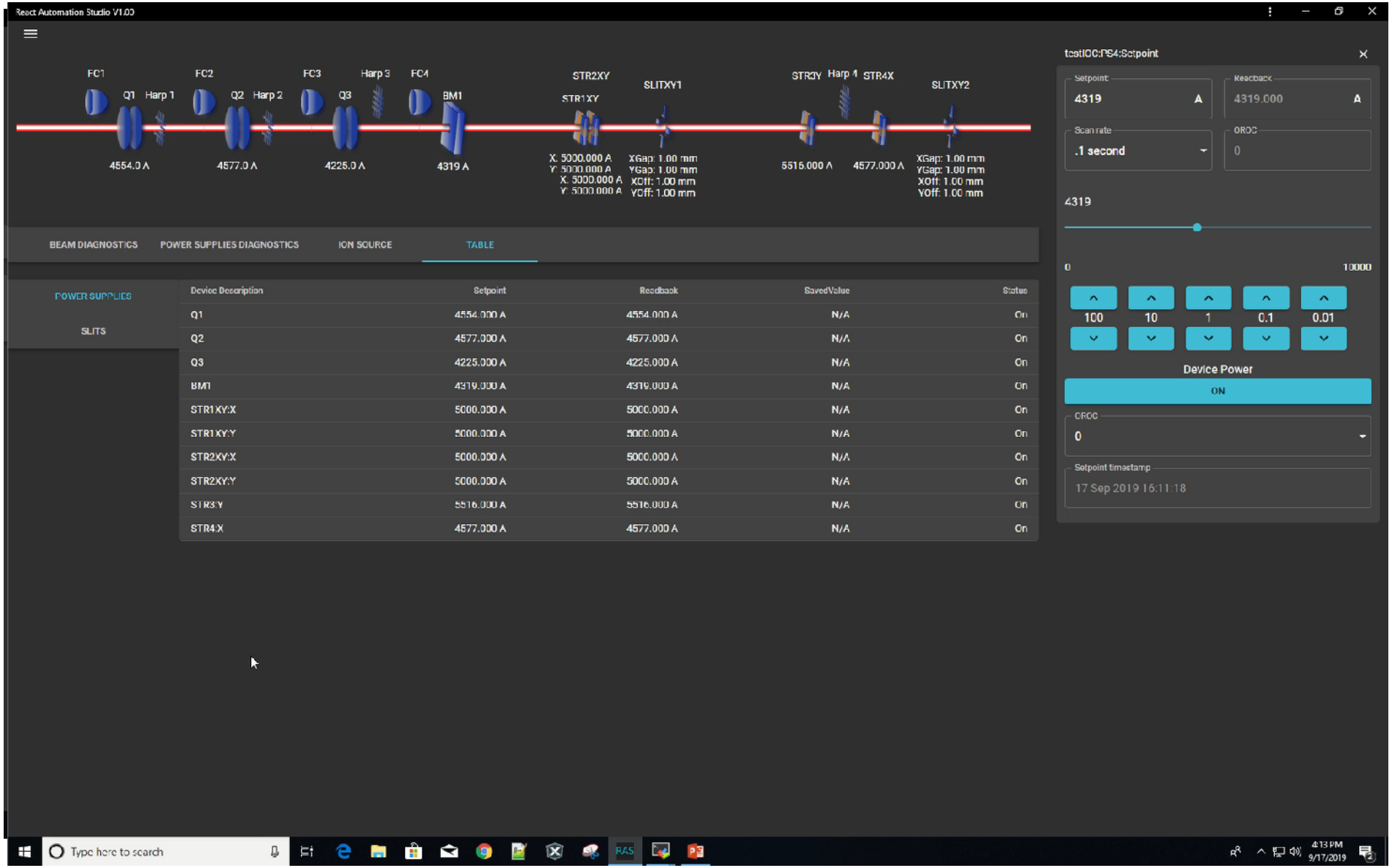
EPICS Demos


React Automation Studio V1.00
React Automation Studio V1.00
MOBILE DEMO 1
MOBILE DEMO 2
EPICS DEMOS
STAGING
3D DEMOS
BEAM LINE CONTROL DEMO
CONTROL TABLE EXAMPLE
HELP AND STYLE GUIDE
Type here to search
RAS

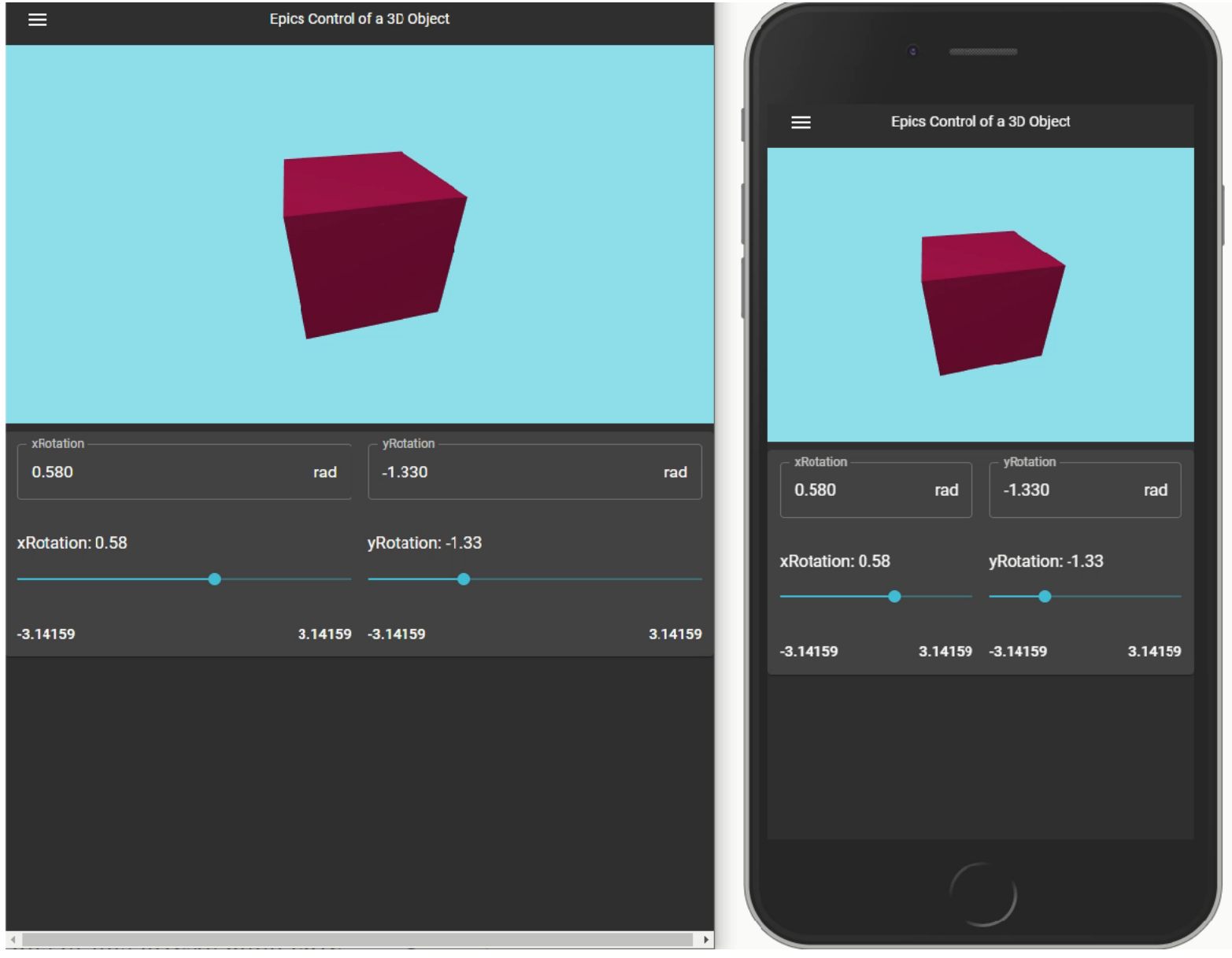
Desktop Beam Line Control Demo



Diagnostic features



3D Demo


Epics Control of a 3D Object
xRotation
0.580
rad
yRotation
-1.330
rad
xRotation: 0.58
yRotation: -1.33
-3.14159
3.14159
-3.14159
3.14159
Epics Control of a 3D Object
xRotation
0.580
rad
yRotation
-1.330
rad
xRotation: 0.58
yRotation: -1.33
-3.14159
3.14159
-3.14159
3.14159

Development Mode

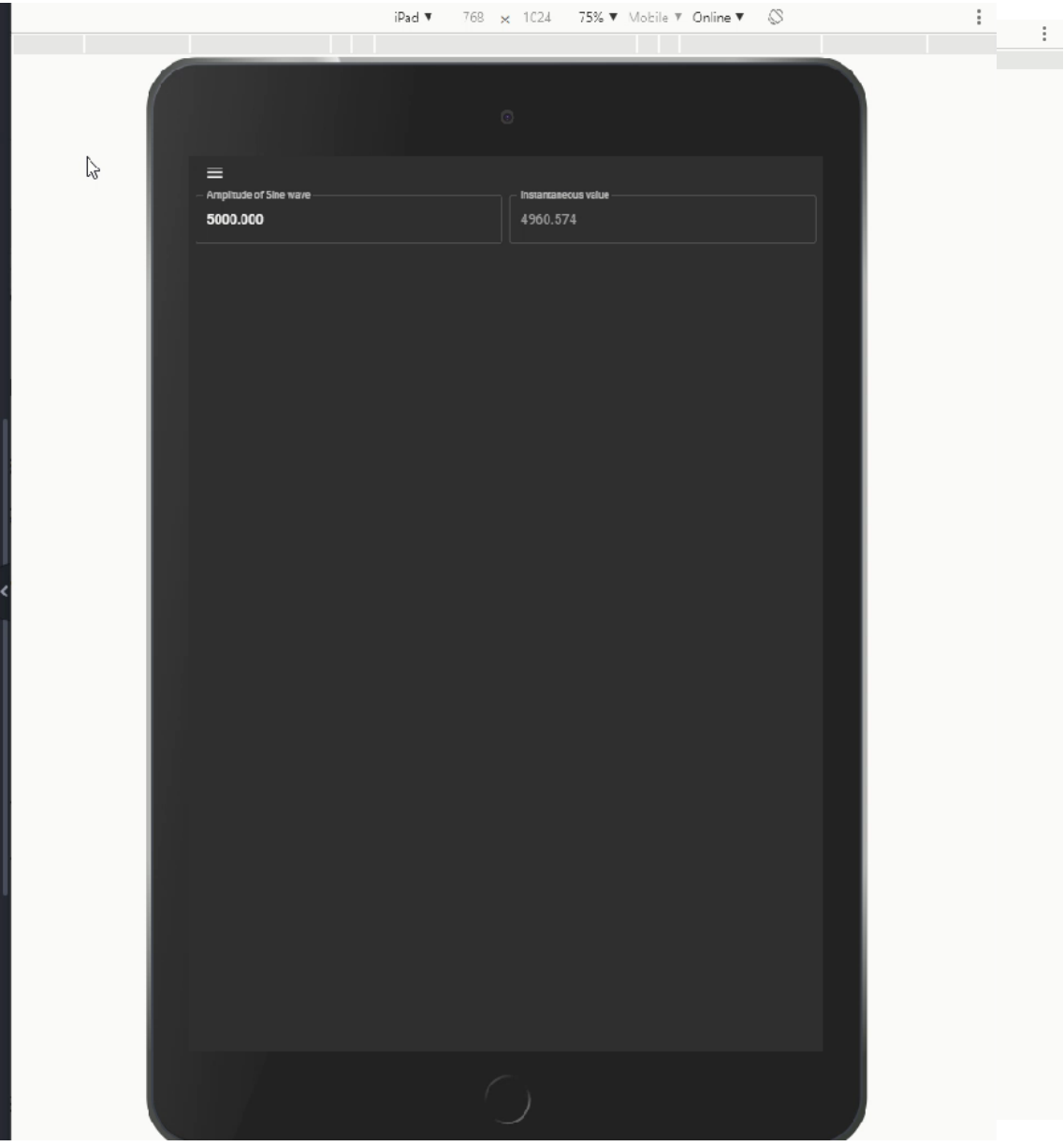
```
]);
class Example2 extends React.Component {


  render() {
    //  console.log("state: ",this.state);

    return (
      <React.Fragment>
        <SideBar/>
        <div >
        <Grid
           style={{padding:8}}
           container item
           direction="row"
           justify="center"
           spacing={1}
           alignItems="stretch"
        >
          <Grid item xs={6}  >
            <TextInput
                pv='pva://$(device):amplitude'
                macros={{'$(device)':'testIOC'}}
                usePvLabel={true}
                usePrecision={true}
                prec={3}
                alarmSensitive={true}
            />
          </Grid>
          <Grid item  xs={6}>
            <TextOutput
              pv='pva://$(device):test3'
              macros={{'$(device)':'testIOC'}}
              usePvLabel={true} usePrecision={true}
              prec={3} alarmSensitive={true}
            />
          </Grid>



          {/* <Grid item xs={12} >
            <div style={{ height: '59vh', width: '96vw',}}>
              <GraphMultiplePVs  pvs={['pva://testIOC:test4','pva://testIOC:test5'] } legend={['Sine Wave','Am
            </div>
          </Grid> */}
          {/* <Grid item xs={12}  >
```



Introduction

React Automation Studio is a new software platform to enable the control of large scientific equipment through EPICS.

The system has been containerised with Docker and version controlled as a mono-repository using Git.

Each of the Docker containers are deployed as micro services and environment variables can be configured to deploy the system on different ports, or to enable user authentication and authorisation or to serve the application on a unique URL or on the localhost. Separate Docker commands exist to load the development and production version. These containerised environments allows for precise versioning of packages used and prevents deployment dependency issues.

The software stack for React Automation Studio is shown in Fig. 1 and an overview of the system components are give below:

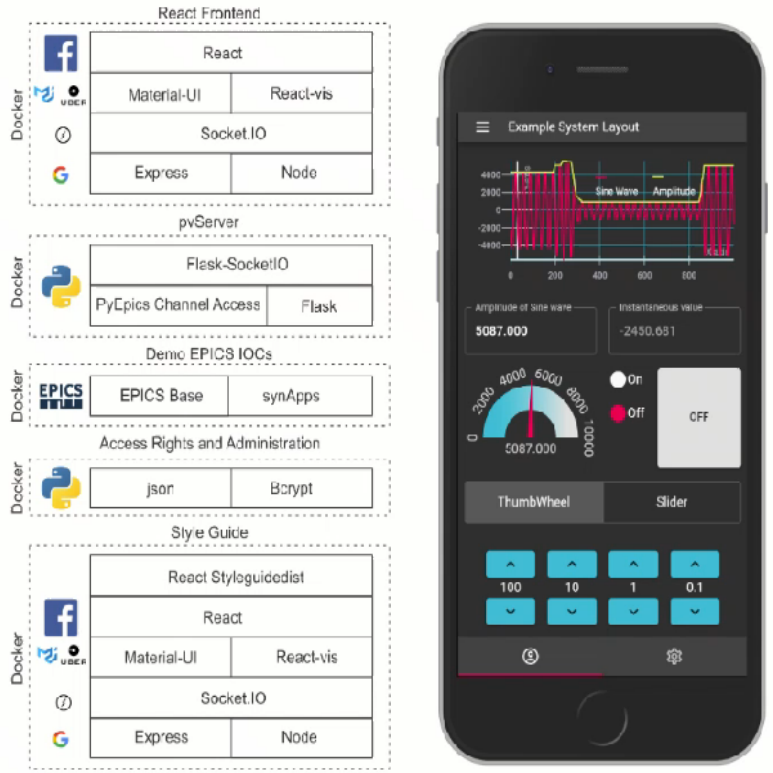
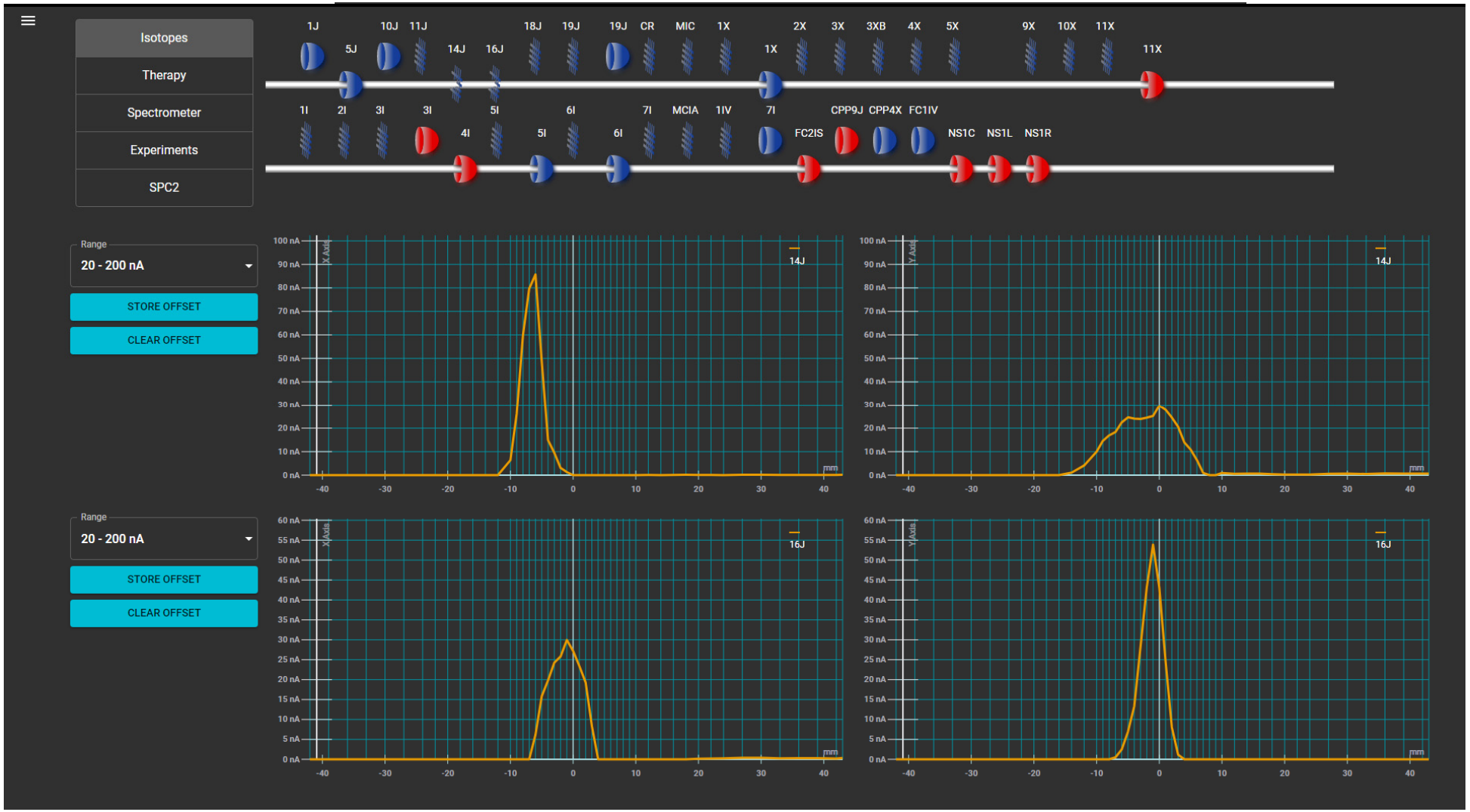


Fig 1. The current software stack and an example mobile layout.

An overview of the system components are give below:

Current Implementation at iThemba LABS



Currently we are using React Automation Studio for all the beam diagnostics which control the harps and Faraday cups - 1683 PVs

Also used as the control system frontend Low Energy Radioactive Ion Beam demonstrator (LERIB) – 648 PVs

We will shortly role out the table based frontend to control the cyclotrons and beamlines per area

Future Work

We will be upgrading all our current user interfaces to React Automation Studio

We also plan to add in some new features such as an archived data viewer, an area based alarm handler plus some new widgets

We will also be releasing the software to the community shortly

Conclusion

Successfully designed new PWA software platform to enable the control of large scientific equipment through EPICS

The system is cross device and cross platform compatible

Production ready

We are in the process of open sourcing the software

We encourage the community to test, evaluate and contribute to React Automation Studio

Thank you