MODULAR SOFTWARE ARCHITECTURE AND SYSTEM TESTING IN THE PROCESS ENGINEERING SECTORS FOR PRODUCING GREEN HYDROGEN
Founded 1945

Solar energy
First solar inverter 1995

Perfect welding
Digitalization 1998

Welding technologies
Welding Robots 2015

Solar energy
New inverter technologies 2001

Hydrogen Technology
First Solhub 2018

Welding technologies

Charging
Three phase charger 2013

Perfect welding
CMT technology 2005

Hydrogen Technology
Start Hydrogen research 2002

First
Solhub
2018

Solar energy

Hydrogen Technology
Start Hydrogen research 2002

Welding technologies

Charging
Three phase charger 2013

Perfect welding
CMT technology 2005

Welding technologies
Start with hydrogen technology research 2002

Hydrogen intralogistics at BMW 2017

Wind to hydrogen 2015

Solhub prototype 2018

Energy self-sufficient house 2012

Hydrogen in intralogistics 2007

Founded new Team for hydrogen research and development 2019

HySnow research project 2020
Joachim Danmayr

University of Hagen
Computer Science
2013 -

JKU Linz
Technical physics
2009 - 2010

HTL Steyr
Electronics
2002 - 2007

Fronius
technical lead
software development
hydrogen systems
2019 -

Fronius
software architect
embedded systems
solar inverter
2015 - 2019

Fronius
hard- and software developer
power electronics control
solar inverter
2008 - 2015
Fabian Neubacher

University of Applied Sciences
Upper Austria
Human-centered Computing
(MSc)
2017 - 2019

University of Applied Sciences
Upper Austria
Medical Engineering (BSc)
2011 - 2014

HTL Vöcklabruck
Machine and Plant
Engineering
2005 - 2010

Fronius
Technical lead
testsystem development
hydrogen systems
2019 -

Novartis
Project Manager Engineering
Qualification Engineer
NTO - Aseptics
2016 - 2019

Austrian Red Cross
Emergency Paramedic
Vöcklabruck
2015 - 2016
1. Photovoltaics
2. Inverter
3. Electrolyzer
4. Fuel Cell
5. Hydrogen storage
6. Dispenser
7. Grid
- Modular system design with centralized IPC and cloud connection.
- Goal to bring the complexity to the IPC
• Classically the main functions are implemented in PLCs.

• We are focused on outsourcing most of the functionality to the IPC.

• The IPC has a microservice based architecture design which allows us to develop evolutionary.
What is a microservice and why we need it?

**microservices**| also known as the **microservice** architecture - is an architectural style that structures an application as a collection of services that are highly maintainable and testable, loosely coupled and independently deployable. [1]

What is a microservice and why we need it?

advantages| complex systems can be divided into little, less complex components. These components can be developed independently from each other and put together later on.
How to get such independent loosely coupled software components?

**docker** software programs depends on other libraries and the operating system the program is executed on. To resolve these dependencies the program with all its dependencies and parts of the operating system are packed together. This is done with so called docker images.

```
>> docker build
```
How to execute and manage these docker images?

Kubernetes now the microservices are packed into images. For deployment, executing and monitoring Kubernetes is used. With Kubernetes docker images can be started and stopped as well as updated. Kubernetes allows to create clusters which manage load balancing.

>>> kubectl apply -f myDeployment.yaml
Standalone hydrogen system...
**Agend system**... 

**agend** | agend microservices are used for representing an embedded system as microservice in the high level system.
Connection to cloud...
Reusability of services...

**Reusability** the microservices can be used for different projects and application. If new functions are needed they can just be implemented and added to the pool of functions.
How does this work?

registry | There is a centralized Fronius docker registry in the cloud. This registry can be accessed from all Fronius devices. The registry contains all available microservices. Kubernetes downloads the needed images from this registry and starts the contains within its cluster.

>> docker pull
SYSTEM TESTING
• Modular system design with centralized IPC and cloud connection.
• Goal to bring the complexity to the IPC
BASICS: HIL-SYSTEM

- HIL = Hardware in the loop
- QA for software controlled components according to 61508
- Real controller / simulated system

Diagram:
- Reference
- Measured error
- Controller
- System input
- System
- Measured output
- Sensor
- System output
TEST-SYSTEM

IPC = Industrial Personal Computer
PLC = Programmable Logic Controller
HIL = Hardware in the loop

Fronius International GmbH / Fabian Neubacher / 30.11.2020
OPAL-RT SYSTEM

OFFLINE
All simulated

RCP
Simulated controller, real plant

HYBRID

HIL
Real controller, simulated plant

PHIL
Real plant, simulated plant
ACTUAL STRUCTURE

Plant Production Model

Fake PLC controller under test

OP4200

Controlling the two simulations from a Python script

CAN
ACTUAL STRUCTURE

Simulation environment:
Matlab/Simulink, Simscape
FROM MODEL TO REAL TIME

1. **Edit**
   Open your Simulink™ model directly via RT-LAB.

2. **Compile**
   Transform your Simulink™ model into a real-time application.

3. **Execute**
   Run the simulation on your real-time target using multiple cores.

4. **Interact**
   Use the graphical interface to change controls and acquire data.
## CE-COMPLIANT TEST ARCHITECTURE

<table>
<thead>
<tr>
<th>AREA</th>
<th>DESIGN</th>
<th>VERIFICATION</th>
<th>TEST</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIL (OPAL-RT Rack)</td>
<td>I/O Table including sampling rate</td>
<td>Factory calibration</td>
<td>Qualification Recalibration</td>
</tr>
<tr>
<td></td>
<td>Design verification</td>
<td>Design verification</td>
<td></td>
</tr>
<tr>
<td>Signal Conditioning</td>
<td>Requirements</td>
<td>Initial calibration</td>
<td>Recalibration</td>
</tr>
<tr>
<td>OPAL-RT Lab (Software)</td>
<td>Standards</td>
<td>NA</td>
<td>Newest release (maintenance agreement)</td>
</tr>
<tr>
<td></td>
<td>Proven on the market</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mathworks -&gt; C</td>
<td>Proven on the market</td>
<td>Code review (sample)</td>
<td>Newest release (maintenance agreement)</td>
</tr>
<tr>
<td></td>
<td>State of the art</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Models</td>
<td>Tracing back to physical and mathematical</td>
<td>Review</td>
<td>Operation within defined limits</td>
</tr>
<tr>
<td></td>
<td>principles</td>
<td>(comparison with measured data)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Derivation from measured data</td>
<td>Review</td>
<td>Operation within defined limits</td>
</tr>
<tr>
<td>Test - Programs</td>
<td>Styleguides</td>
<td>Review</td>
<td>Implicit in test execution</td>
</tr>
</tbody>
</table>

/ Perfect Welding / Solar Energy / Perfect Charging