4diac as base technology

Gerhard Ebenhofer

6th September 2016, Berlin

LEADING
INNOVATIONS

WWW.PROFACTOR.AT

Overview

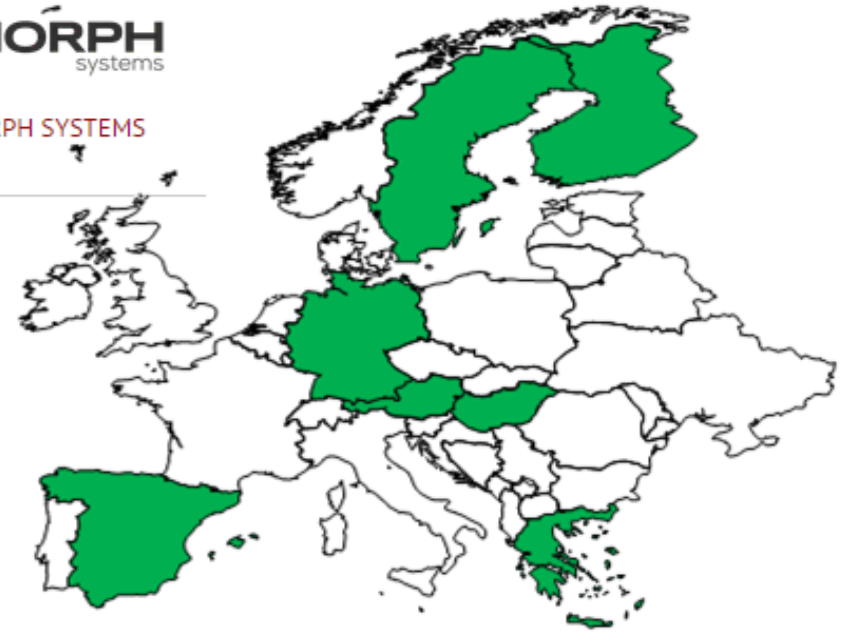
- SYMBIO-TIC Project
  - Objectives
  - Consortium
  - Project Overview
  - Main Topics
- RESTful API
- Example/Demo

SYMBIO-TIC – Objectives: Human-Robot collaboration

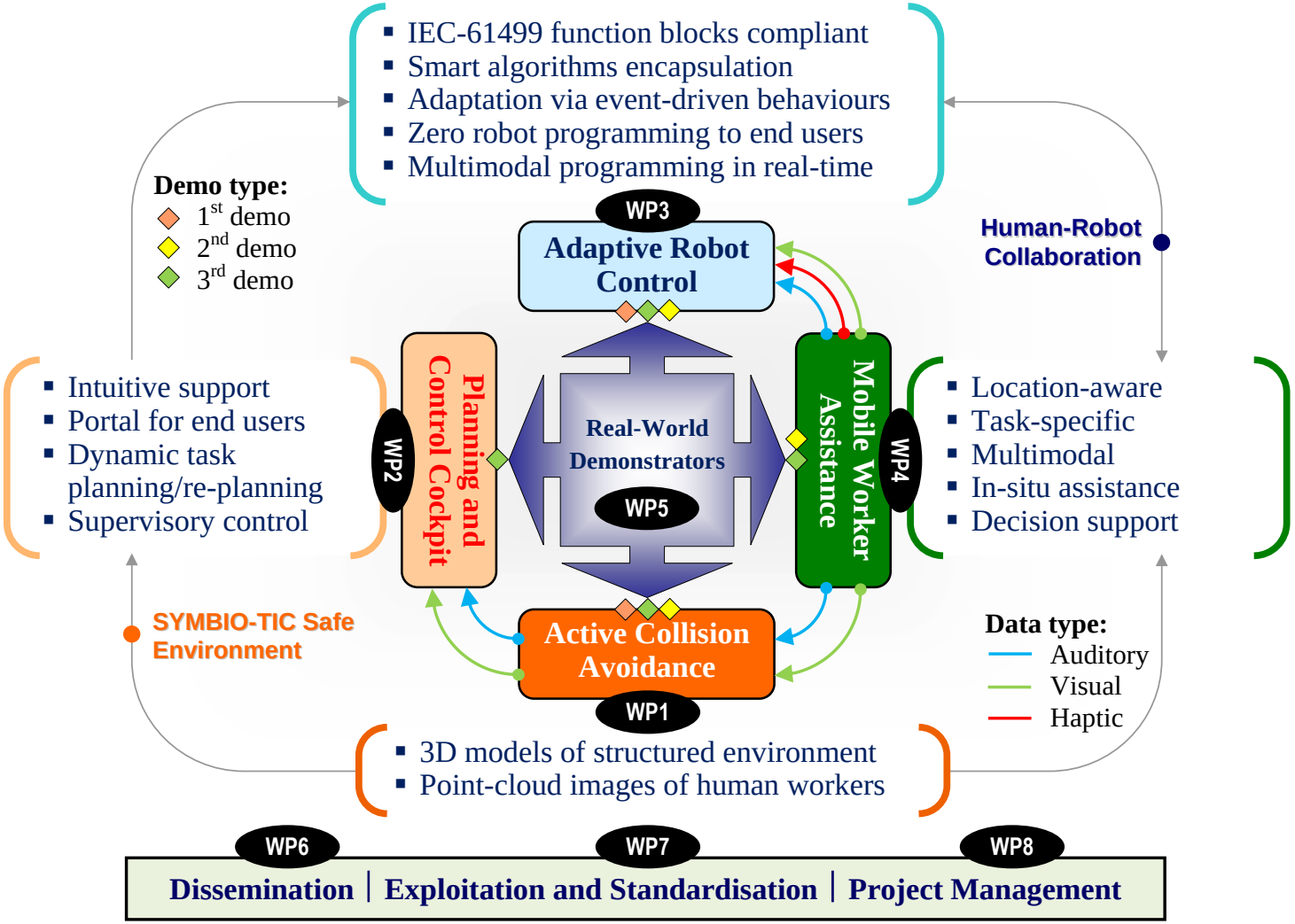
- This SYMBIO-TIC project is addressing the topic H2020-FoF-6-2014, aiming at a novel hybrid assembly/packaging ecosystem in dynamic factory environment based on human robot collaboration.
- **Objective 1**: To develop an active collision avoidance subsystem to safeguard human workers.
- **Objective 2**: To generate adaptive task plans appropriate to both robots and human workers.
- **Objective 3**: To adapt to dynamic changes with intuitive and multimodal programming.
- **Objective 4**: To provide human workers with in-situ assistance on what-to-do and how-to-do.
- **Objective 5**: To demonstrate and validate the project concept and solutions.

SYMBIO-TIC - Consortium

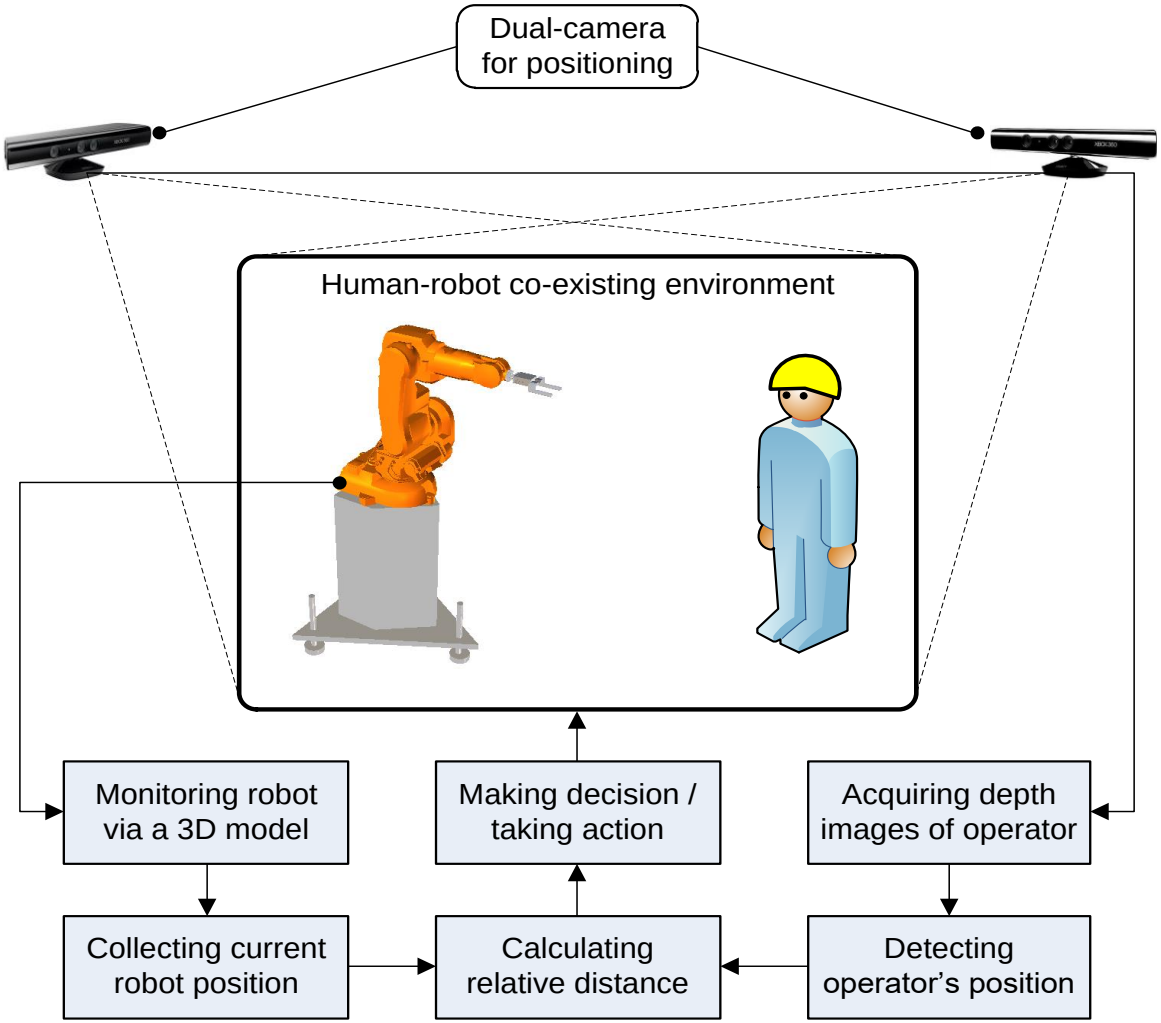
KTH	COMPUTER AND AUTOMATION RESEARCH INSTITUTE	LMS	PROFACTOR
HOGSKOLAN I SKOVDE	IDEKO	FRAUNHOFER	VOLVO CARS
FUNDACION PRODINTEC	SANXO SYSTEMS KFT.	ABB AB	AMORPH SYSTEMS
ROBOMOTION GMBH	ACITURRI ENGINEERING SL	VTT	



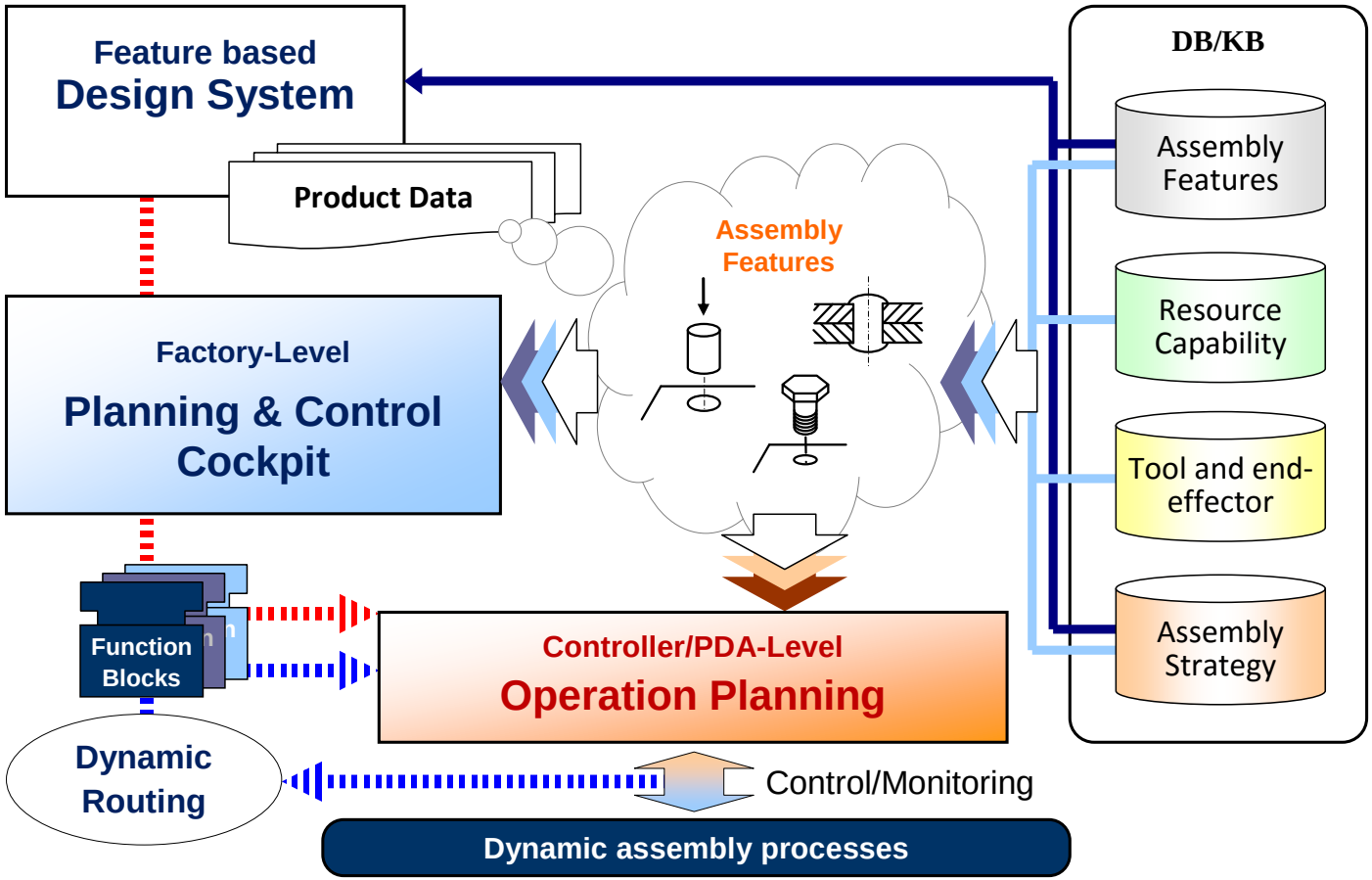
SYMBIO-TIC - Overview



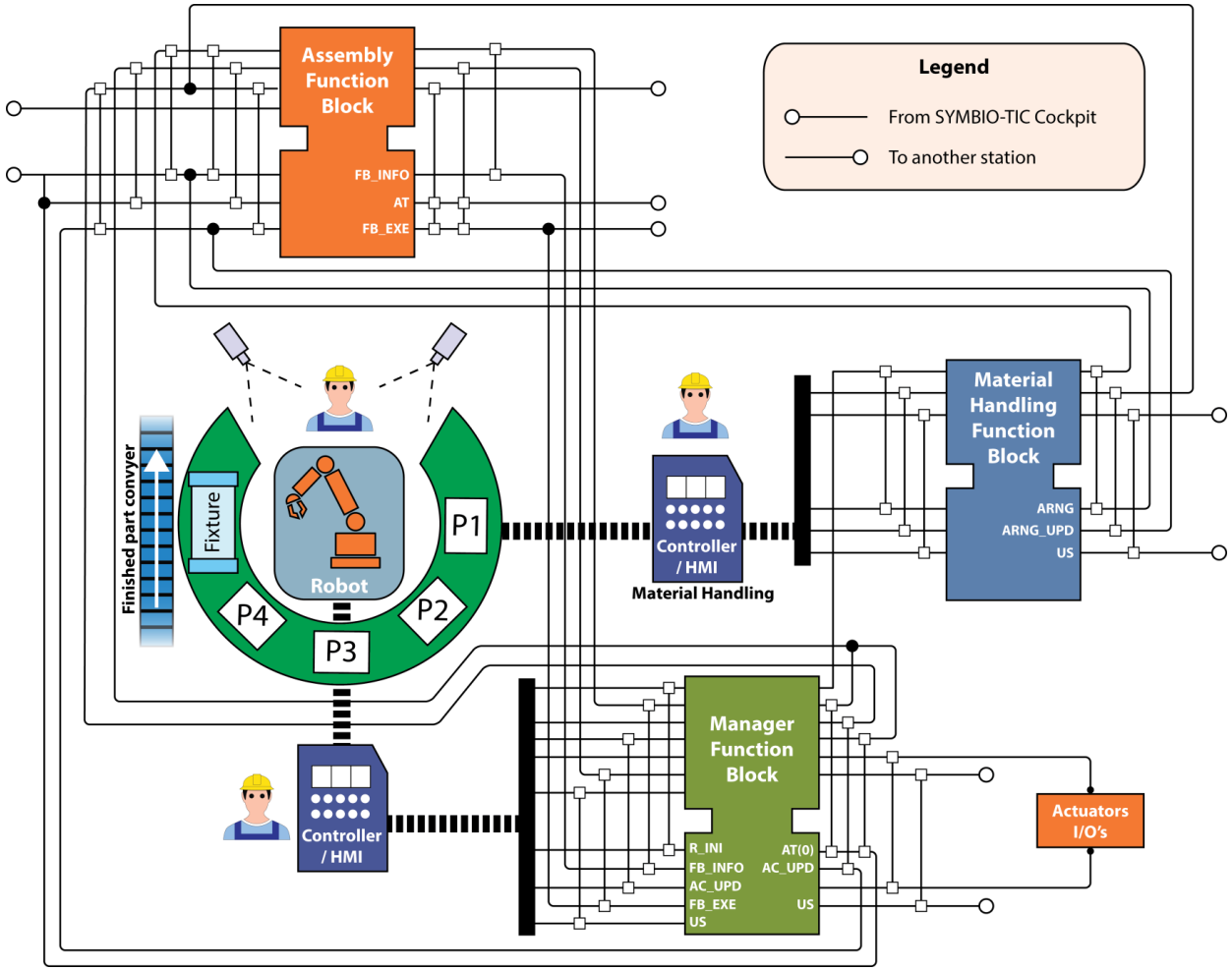
Active Collision Avoidance



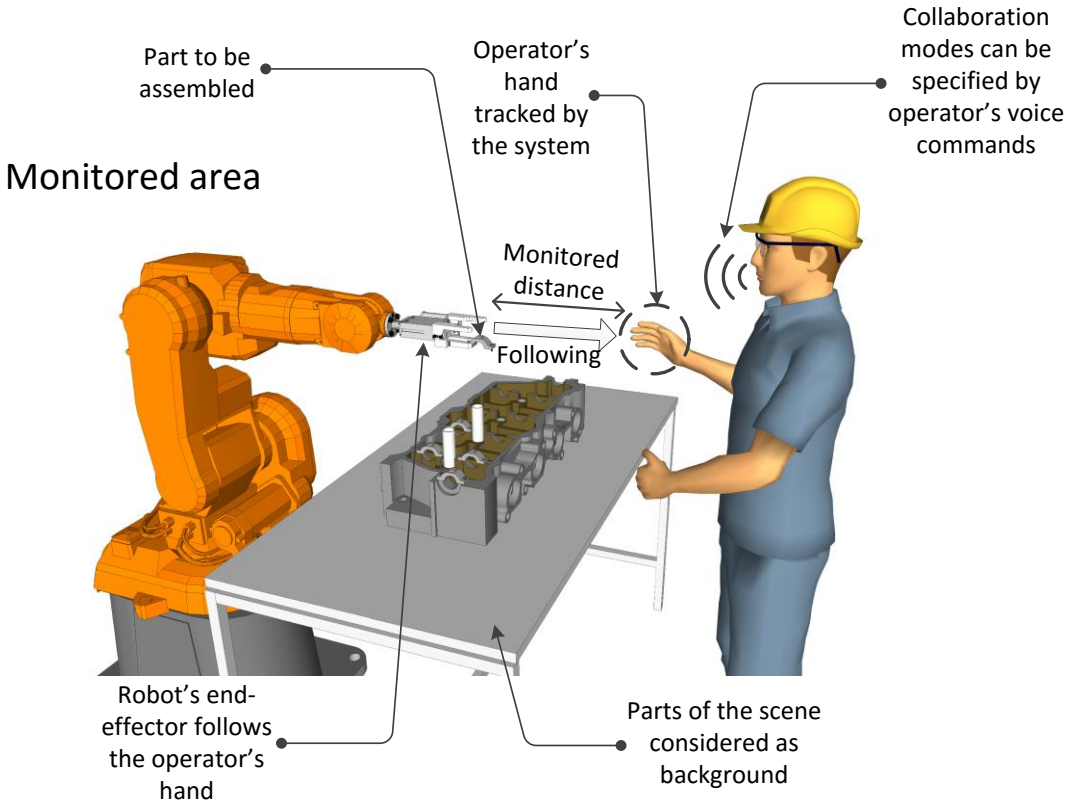
Dynamic Assembly Planning



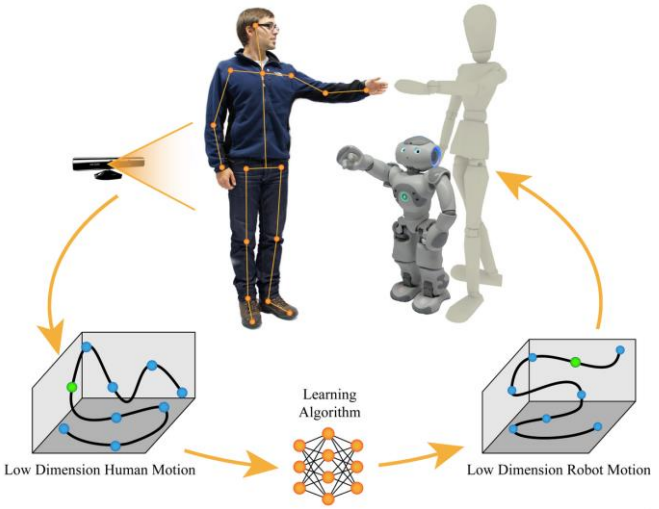
Adaptive Robot Control



Human-Robot Collaboration



- **The capability to track an operator's position allows controlling a robot to follow the operator**



RESTful API

- REST - Representational State Transfer
  - **Architectural constraints**
    - Stateless Client-Server Architecture
    - Cacheable (responses must be marked as cacheable or not)
    - Uniform Interface
      - Identification of resources (e.g. using URIS to identify individual resources, e.g. server sends data as HTML, XML or JSON)
      - Manipulation of resources through these representations
      - Self-descriptive messages (how to process the message – e.g. which parser to use)
      - Hypermedia as the engine of application state (clients make state transitions through actions – e.g. by hyperlinks)
  - Applied to web services that adhere the above constraints
    - Called RESTful API

Why RESTful API?

- Separation of business logic and user interface
- Using of HTTP supports a lot of device
- Updates can be on the server side
- Widely used in Web services (e.g. Twitter, Facebook, Google)
- Pay-per-use, protection of Intellectual properties
- User interfaces can be developed by web designer

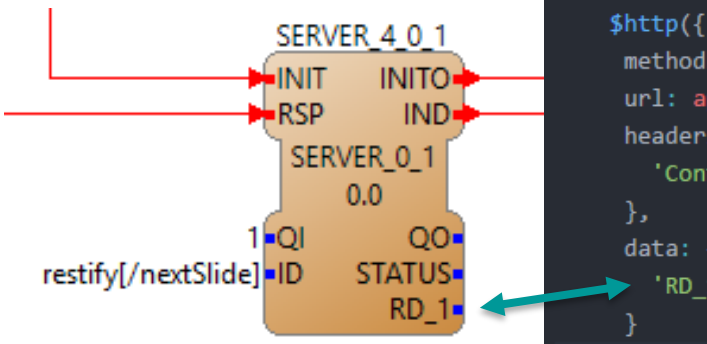
Integration RESTful API in 4DIAC

- Implemented as NetworkLayer
- Using cpp-restify (https://github.com/cheind/cpp-restify)
- Server FB
  - Offer Service
- Client FB
  - Call Service

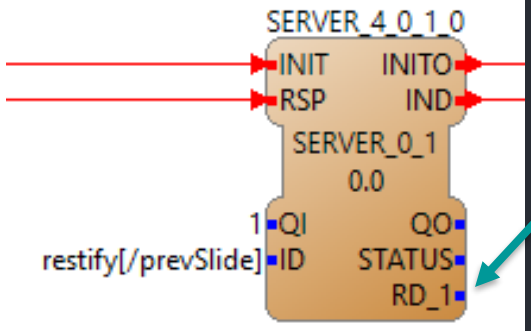
Example RESTful API - offering Service

```
<p><a class="btn btn-primary pull-right" role="button" ng-click="nextSlide('Next')"><span class="glyphicon glyphicon-forward"></s
```

```
<p><a class="btn btn-primary pull-left" role="button" ng-click="prevSlide('Previous')"><span class="glyphicon glyphicon-backward"
```

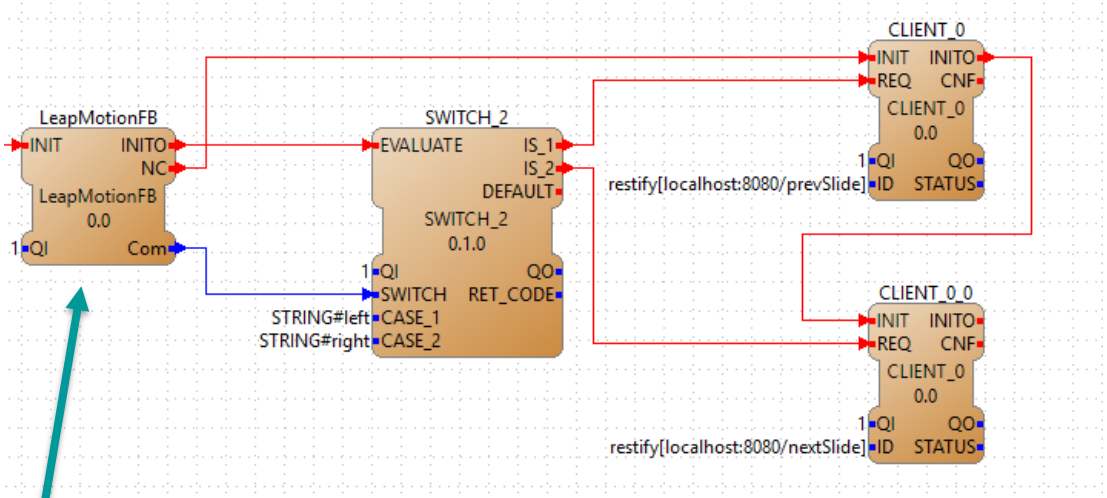



```
$scope.nextSlide = function(direction) {
  $http({
   method: 'POST',
   url: apicfg.url + 'nextSlide',
   headers: {
     'Content-Type': 'application/json'
   },
   data: {
     'RD_1': direction
   }
   }).then((response) => {
     refreshBackendStatus();
   })
 };
```




```
$scope.prevSlide = function(direction) {
  $http({
   method: 'POST',
   url: apicfg.url + 'prevSlide',
   headers: {
     'Content-Type': 'application/json'
   },
   data: {
     'RD_1': direction
   }
   }).then((response) => {
     refreshBackendStatus();
   })
 };
```

Example RESTful API – using Service



Thanks for our attention

- Questions?