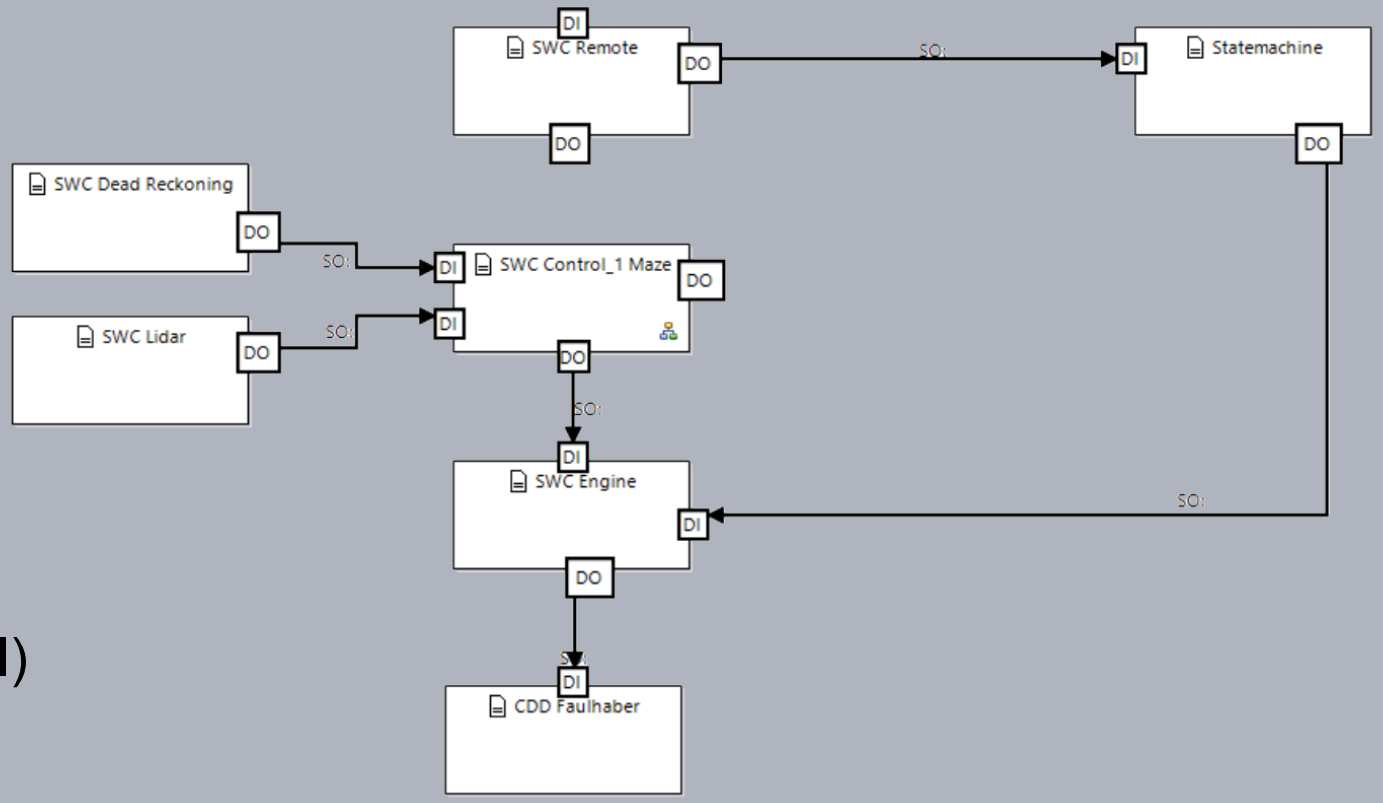
embeddedworld
Exhibition&Conference
... it´s a smarter world

FZI

HIGHTEC

Modeling and Assessment of Safety Critical Systems

Thomas Barth (h_da)
Victor Pazmino Betancourt (FZI)
Bo Liu (FZI)
Prof. Dr.-Ing. Peter Fromm (h_da)
Prof. Dr.-Ing. Dr. h. c. Jürgen Becker (FZI)



The project


Complex applications (e.g. ADAS)
Functional Safety
Multicore
Effort

The project

Application complexity

Multicore complexity

Time to market
Costs

But...
Is it safe?

Content

- Tool concept
- EMF
  - Meta-Modeling
  - Editors
  - Code generation
  - Assessment
- Status and demonstrator
- Conclusion and outlook

Project: Future Technology Multicore

Supported by:

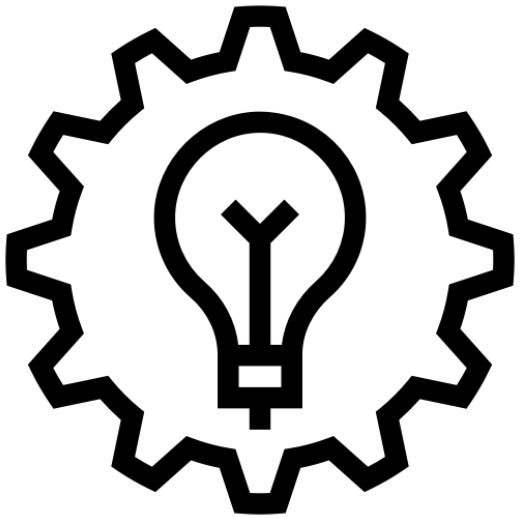
Federal Ministry
for Economic Affairs
and Energy

on the basis of a decision
by the German Bundestag

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbeit
FACHBEREICH ELEKTROTECHNIK
UND INFORMATIONSTECHNIK

Tool concept

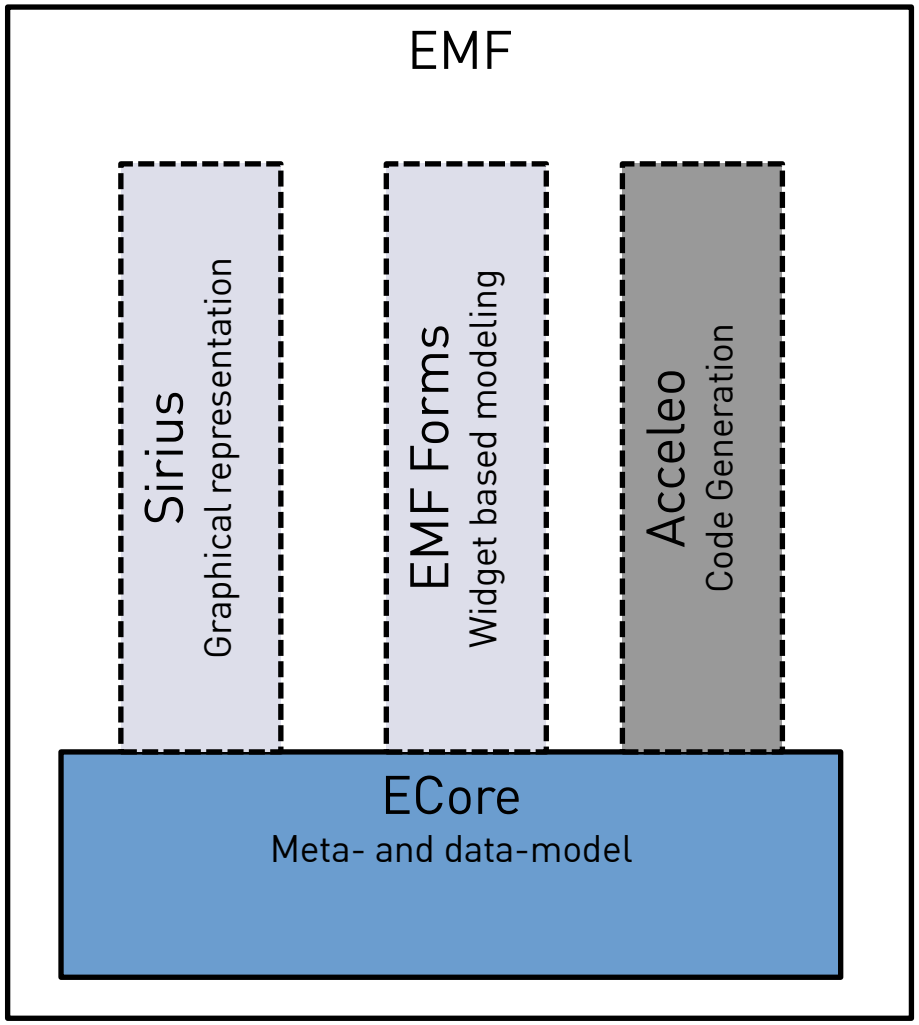
- The tool has to work in a “what you see is what you get” fashion.
- All the data entered should be stored in a singleton data-model with clear dependencies, avoiding detached data.
- Graphical and tabular editors to model signal- and control-flows.
- The tool has to be capable to perform assessment of the entered data.
  - Aspects like Freedom from Interference shall be possible to check.
- Code generation has to be supported
  - A runtime environment (which is currently under development) shall be generated by the tool.



EMF

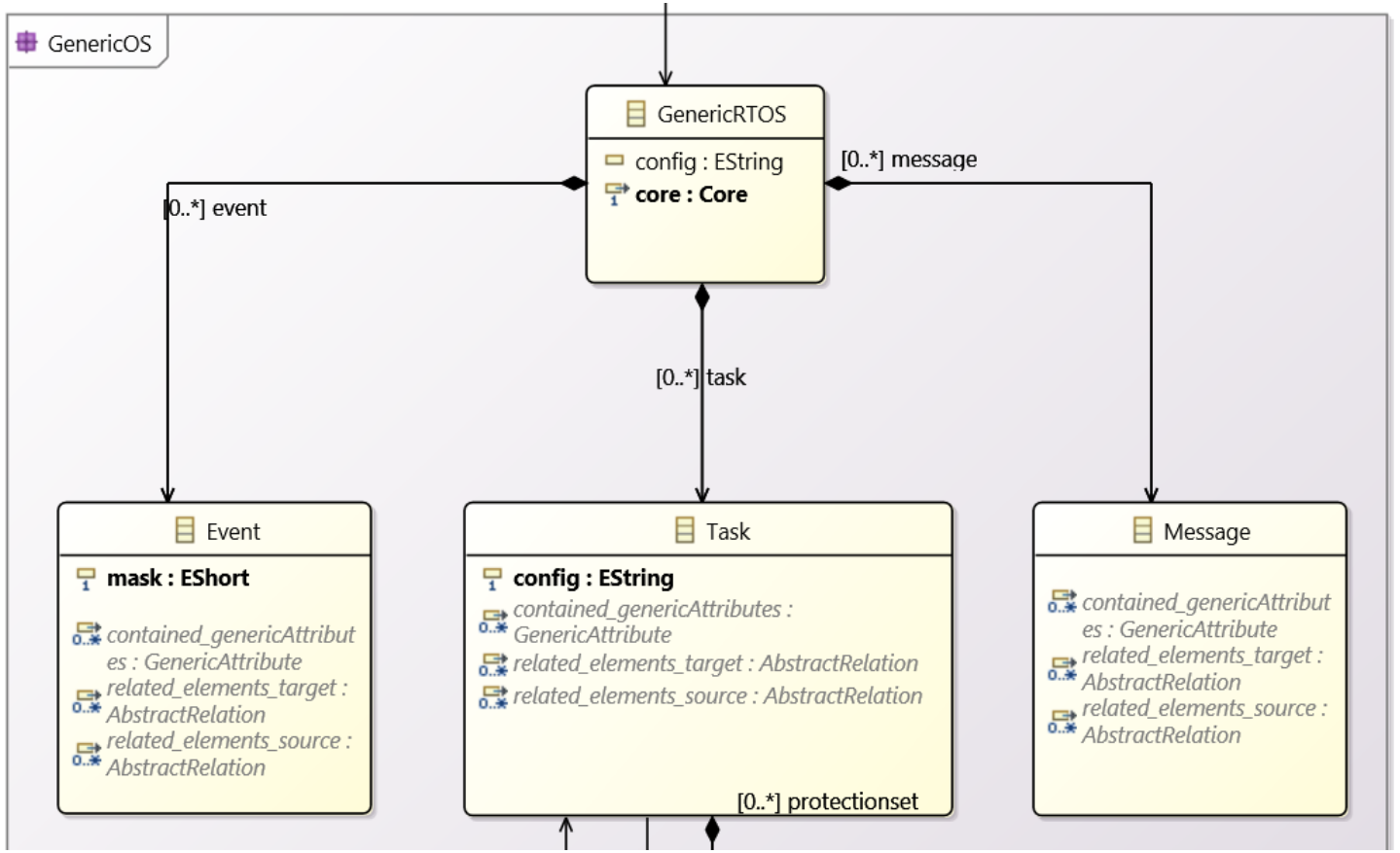
- Eclipse Modeling Framework (EMF)
- Open source Java Modeling framework based on Eclipse

- Extensions for editors, also graphical, available
- Extension for code generation available
- Ready to use solution “Obeo Designer”
- Extensions can be installed individually in Eclipse



EMF Meta-Modeling

The Meta-Model is the base of the complete tool, and defines the structure of data-models. It is comparable to Microsoft ViewModels.

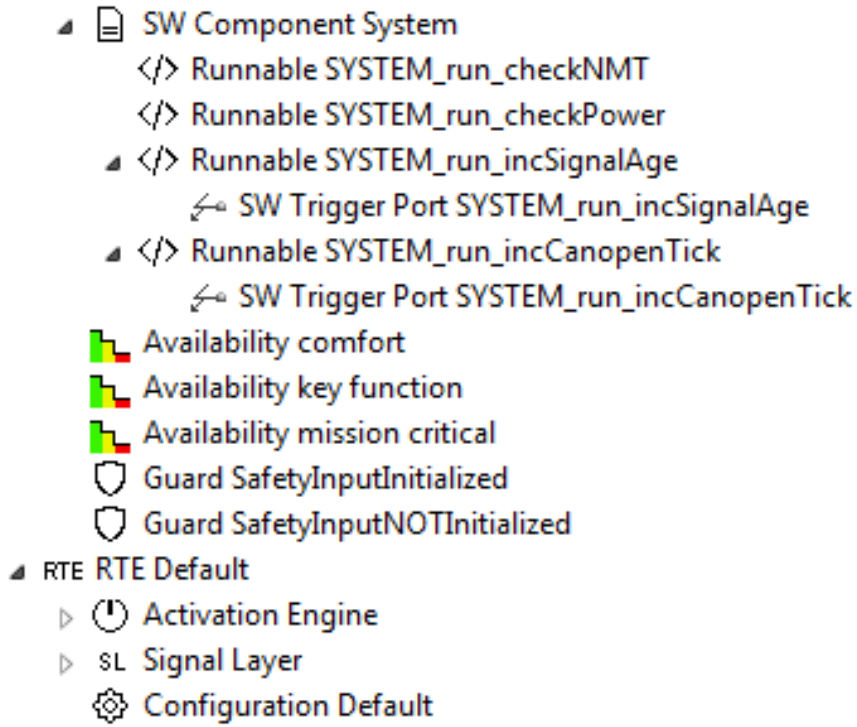


EMF Editors

Tree Editor

- Automatically generated
- Allows data manipulation in the data-model
- Raw view on the data-model
- Mainly intended for debugging

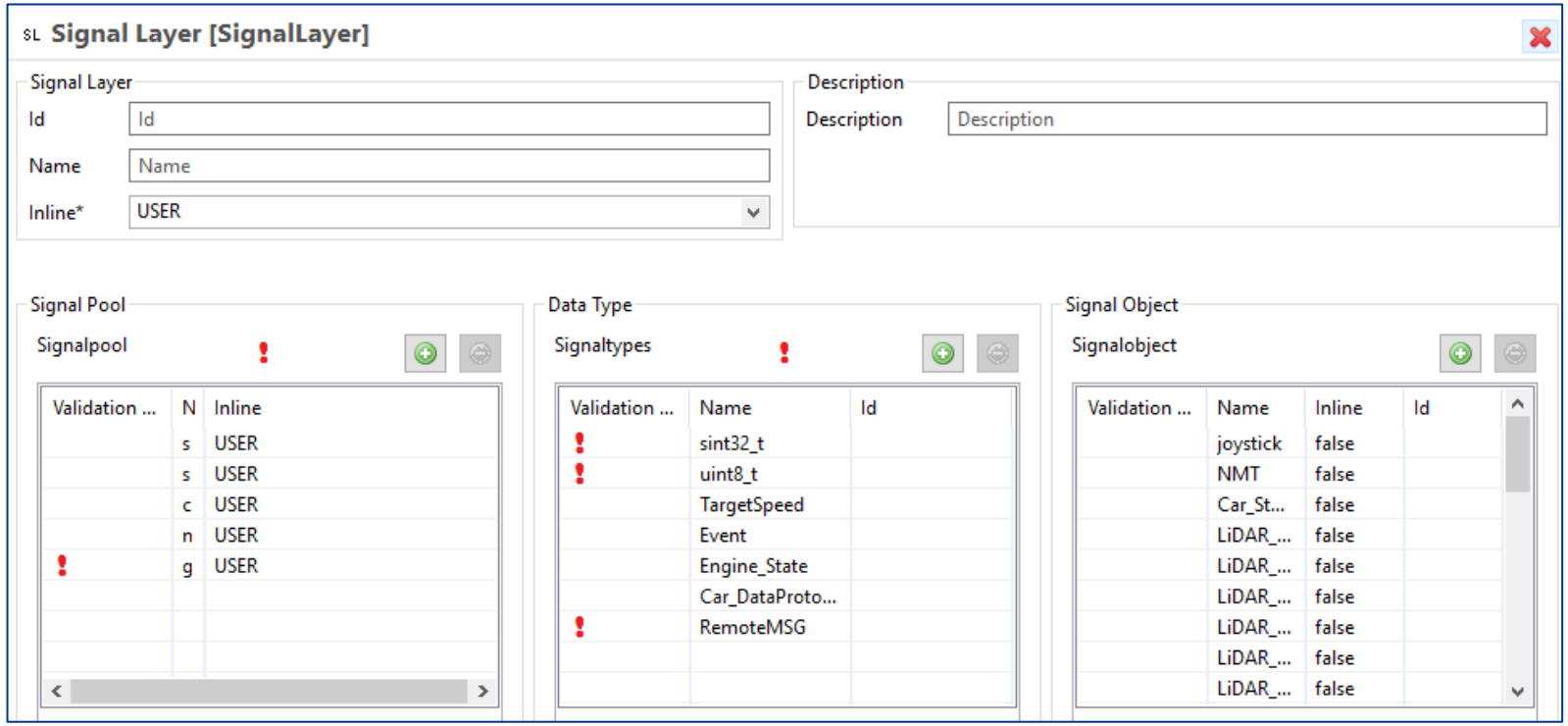
Based on the meta-model, we get a first editor out of the box. Great for verification



EMF Editors

EMF Forms

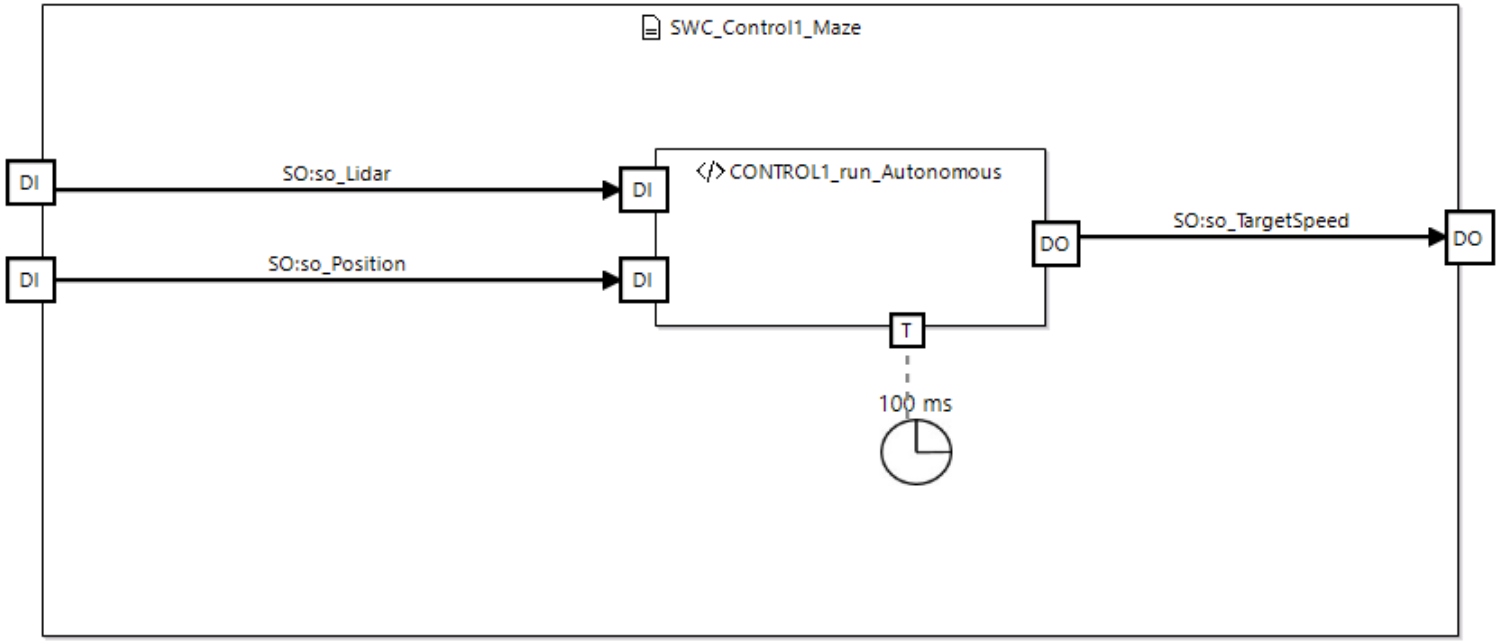
- Data manipulation with well-known widgets (textbox, dropdown)
- Mainly intended to edit the attributes of a single object



EMF Editors

Sirius

- Graphical editor for application specific views
- Supports the visualization of dependencies between objects
- Appearance customizable
- Can iterate trough the data-model and collect data
- Here: Focus on signal-flow

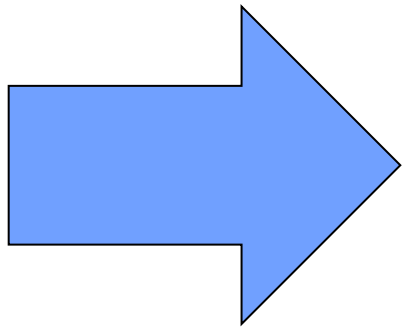


EMF Code generation

Acceleo

- Text (code/documentation) generator introducing Acceleo Query Language (AQL)
  - AQL is a superset of the Object Constraint Language (OCL)
    - Which is part of the UML
- Allows invocation of Java code
- AQL supports iteration trough the model

```
[for (pool : SignalPool | SignalLayer.signalpool)]
[h1(pool.name)/]
/*  [pool.description/]
 *
 * RAM section: [pool.RAMSection.name/] ([pool.RAMSection.description/])
 * ROM section: [pool.ROMSection.name/] ([pool.ROMSection.description/])
 */
.SL.[pool.name/] : {
              . = ALIGN(8);
              PROVIDE([pool.pool_LinkerSym_BEG()/]= .);

              *(.[pool.name/])

              . = ALIGN(8);
              PROVIDE([pool.pool_LinkerSym_END()/] = .);

} > <memory>


[/for]
```



```
/*=========================================[ control  ]======*/
/*  Contains all signals required for controlling the car
 *
 * RAM section: .sl_control (control signals)
 * ROM section: .rodata (Read Only data)
 */
.SL.control : {
              . = ALIGN(8);
              PROVIDE(ADRL_SL_CONTROL_BEGIN= .);

              *(.control)

              . = ALIGN(8);
              PROVIDE(ADRL_SL_CONTROL_END = .);

} > <memory>
```

EMF Assessment

Since the meta-model generated Java classes, Java can be used to perform even complex assessments. Simple assessments might also be performed in Acceleo.

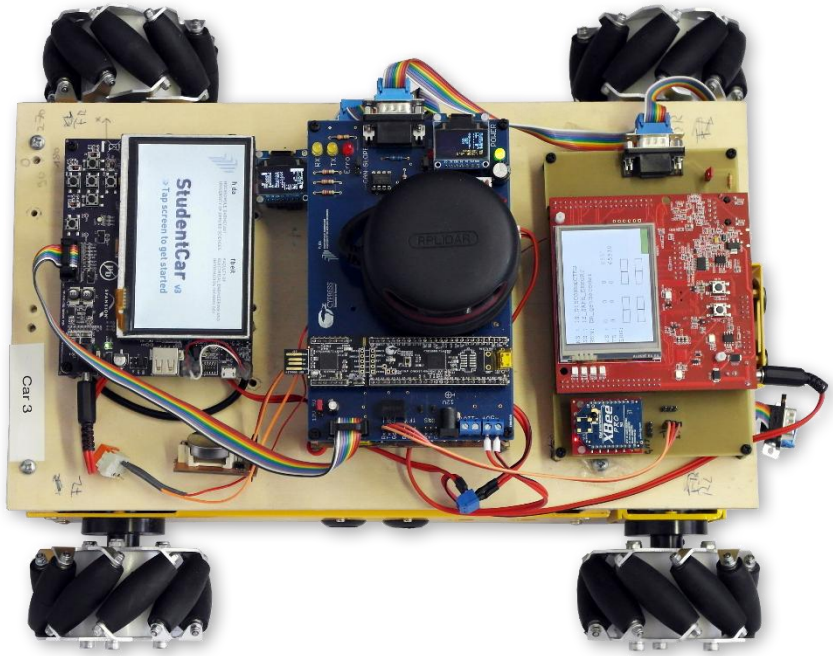
```
public boolean isSignalNameUnique(SignalObject signal){

    //get containing SignalLayer
    EObject current_SL=signal.eContainer();

    //get signal Name
    String signal_name= signal.getName();

    //get all signals in the SignalLayer
    EList<EObject> SL_childs = current_SL.eContents();

    //iterate trough all children
    for (int j=0; j<SL_childs.size(); j++) {

        //only check SignalObjects
        if(SL_childs.get(j) instanceof SignalObject) {

            //check if we have duplicates
            if(((SignalObject)SL_childs.get(j)).getName()==signal_name)
                return false;
        }
    }

    //no duplicates found
    return true;
}
```

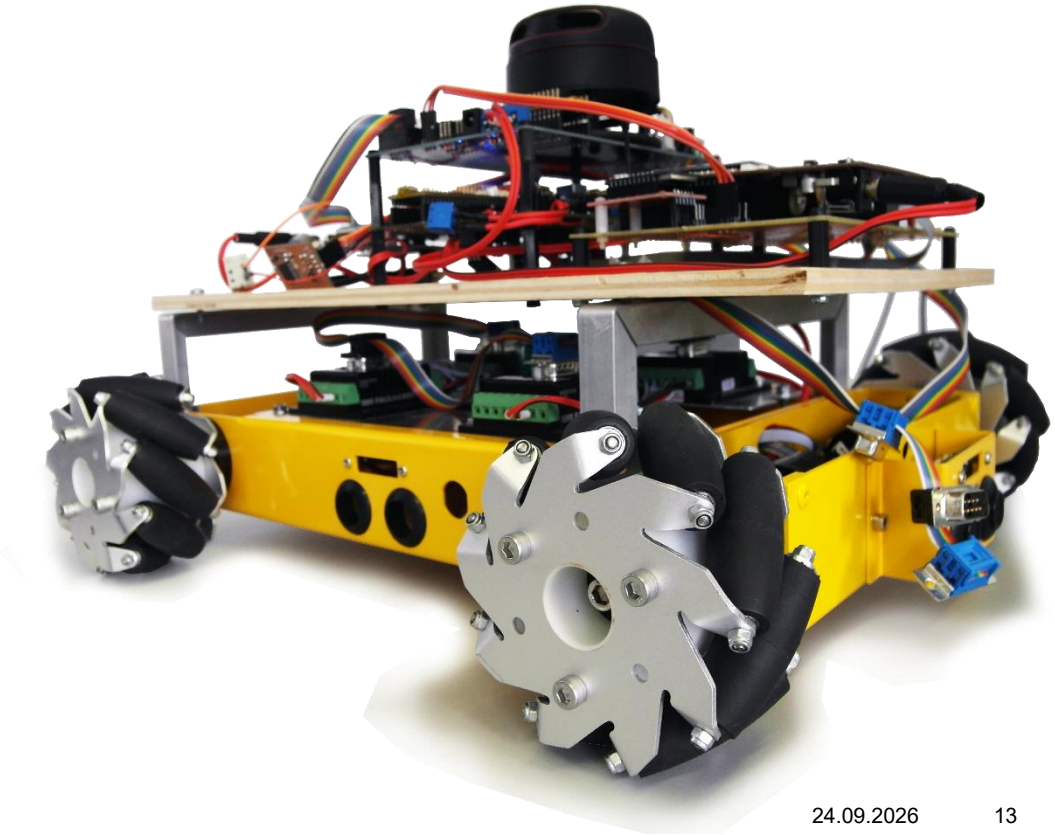
```
[query public isSignalNameUnique(signal : SignalObject) : Boolean =
    signal.eContainer(SignalLayer)
    .eContents(SignalObject)
    ->select(name=signal.name)
    ->size()=1
 /]
```

Status and demonstrator

As part of our research, an omnidirectional robot is built, which shall act as an example.

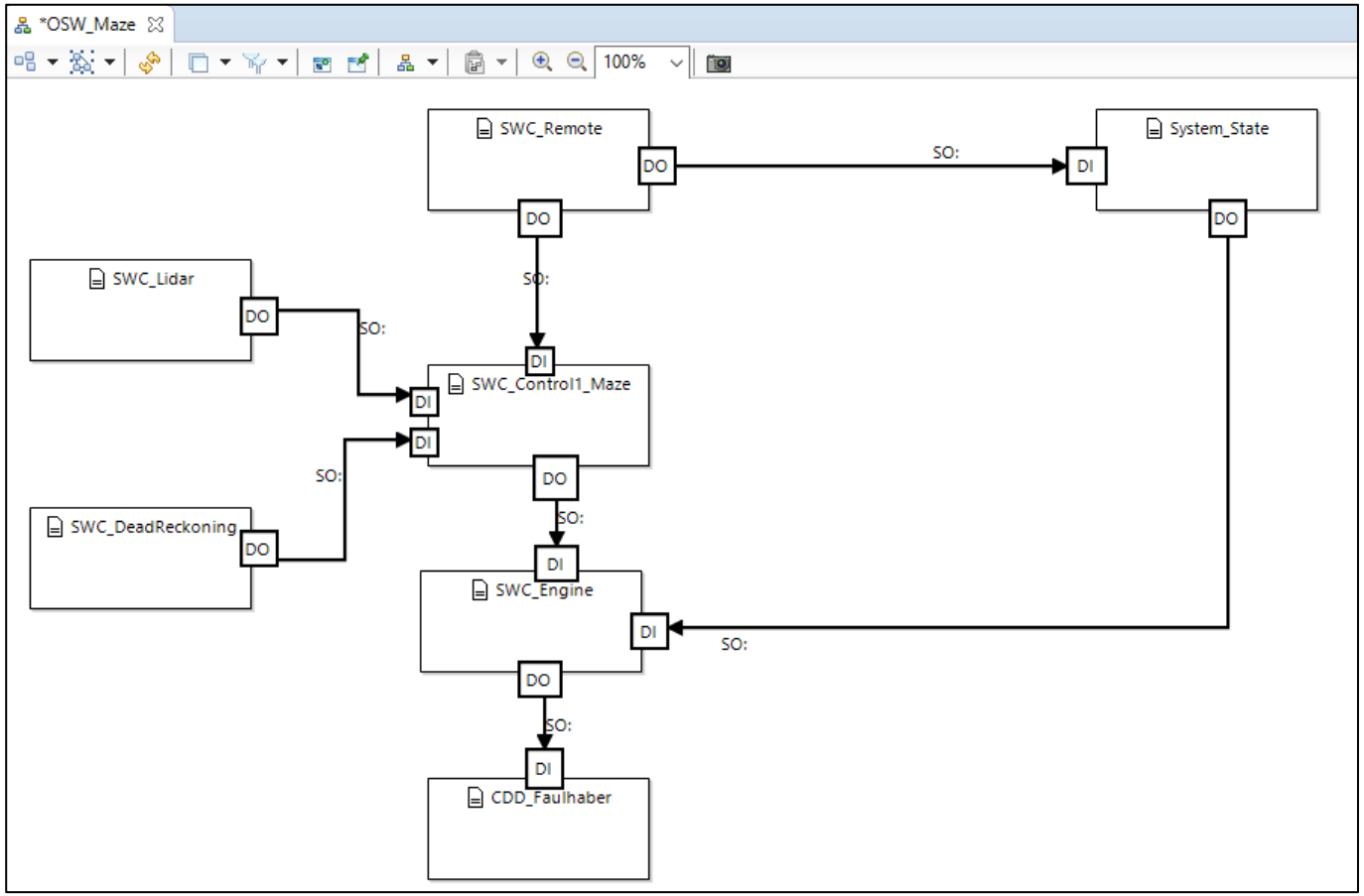
One operational mode of the robot is the autonomous navigation through a maze. As the driving function is considered safety critical, measures for functional safety need to be applied.





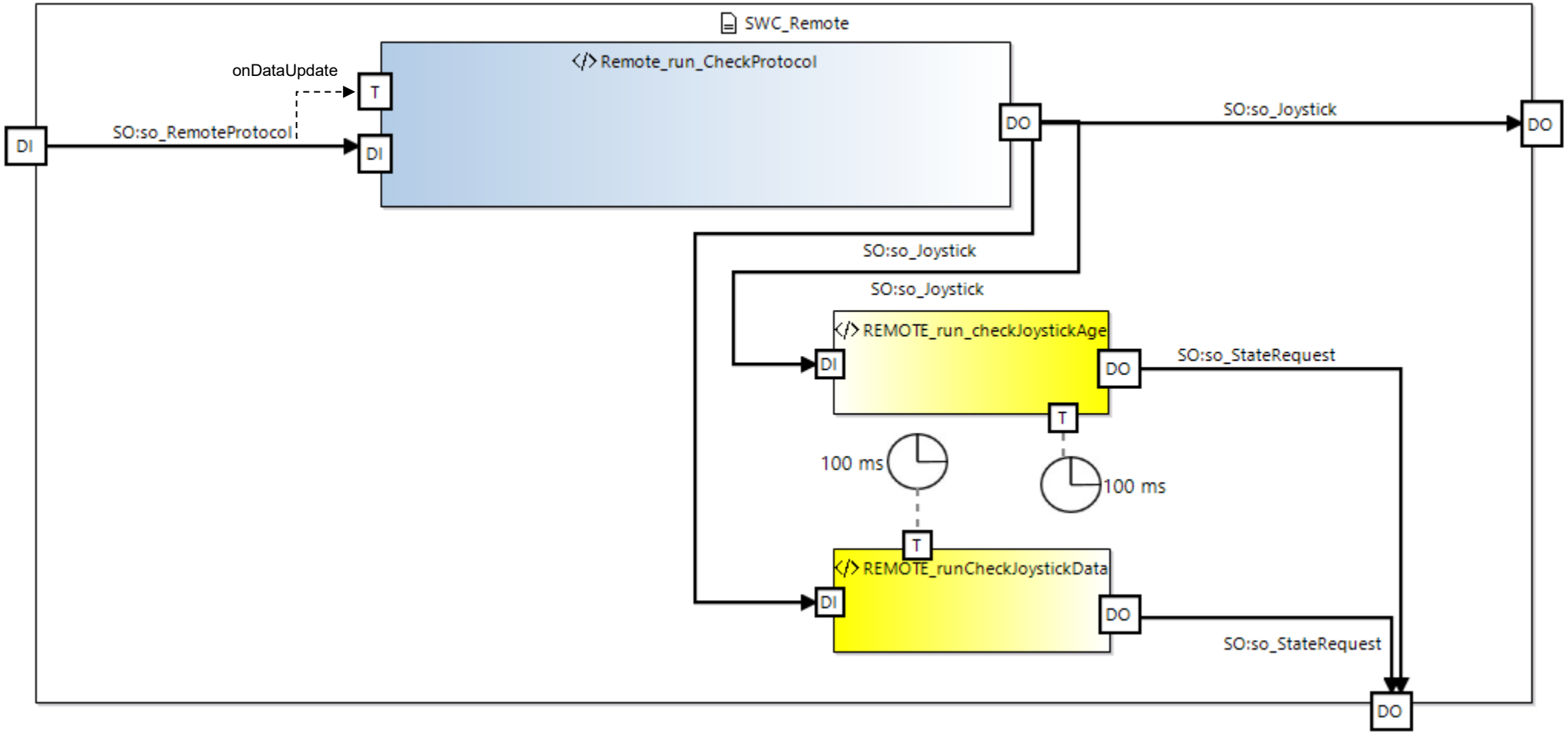
Status and demonstrator

This diagram, which is generated using the Sirius extension, shows the high-level signal flow of the intended functionality. It is possible to create individual diagrams showing only a selected subset of the complete model.



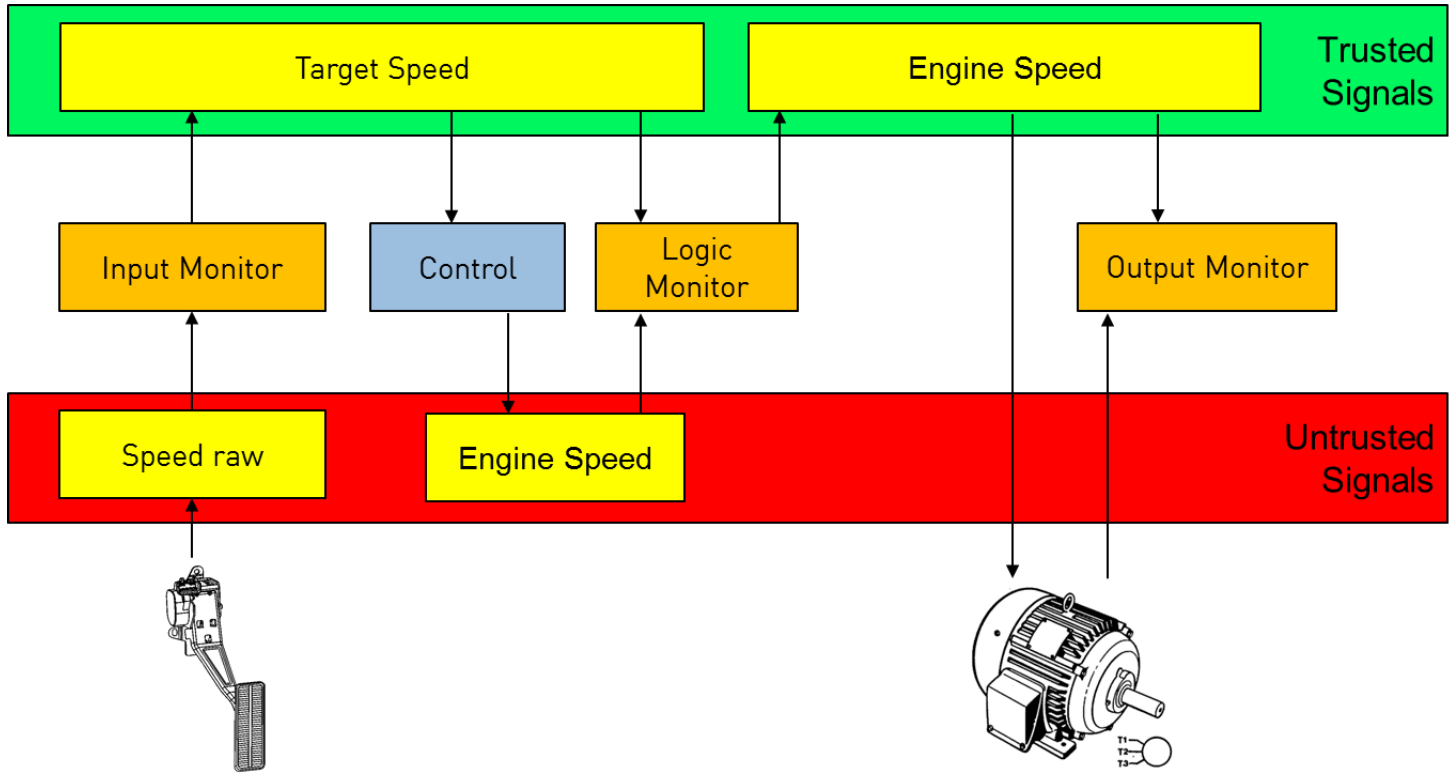
Status and demonstrator

Detailed view of a Software Component, showing triggers and internal dataflow



Status and demonstrator

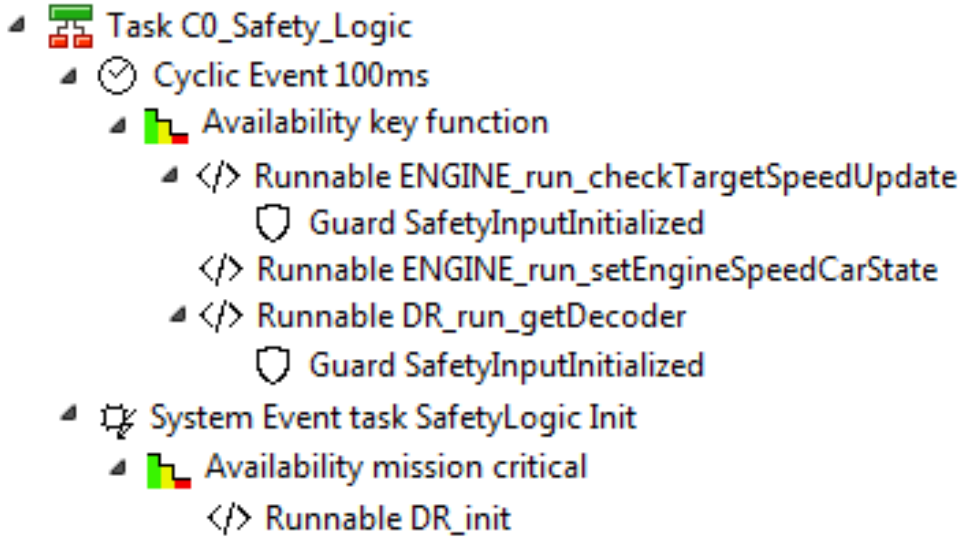
For each signal defined in the model a C++ class will be generated, where the API is tailored to the signal configuration. The signal layer supports organization of signals into pools which are protected by the MPU.



Status and demonstrator

The activation of runnables is centrally handled by the so called activation engine. Each task has a unique and custom activation object, which invokes runnables, establishes communication with other tasks and is controlling the task.

The activation engine supports graceful degradation as well as guard function for each runnable.



Status and demonstrator

Beside code files, the generator also generates documentation in the markdown format.

Activation for task "C2_TFT"

Handles all actions related to the TFT display

System:

Init event "Task TFT Init":

Runnable Name	Guards	SIL Level	Availability	Order
GFXDIAGNOSTIC_display_init	none	QM	comfort	0
GFXDIAGNOSTIC_touch_init	none	QM	comfort	0

Cyclic:

100ms

Runnable Name	Guards	SIL Level	Availability	Order	has Return
GFXDIAGNOSTIC_display_run	none	QM	comfort	0	false
GFXDIAGNOSTIC_touch_run	none	QM	comfort	0	false

Signal Pools Overview

Pool Name	Description	Signals
control	Contains all signals required for controlling the car	Car_Position; Car_TargetSpeed; DiagnosticMapTransfer; DiagnosticScanTransfer; DiagnosticTransferIn; DiagnosticVectorTransfer; LidarMap; LidarVector; MAZE_01; Remote_MsgFromControl;
global	globally accessible signals	LidarTriggerPDO; TEST; ev_emergencyStop; ev_setLidarTargetspeed; ev_setSpeed; ev_stopCar;
misc	Contains all miscellaneous signals	Touch_event; touch;

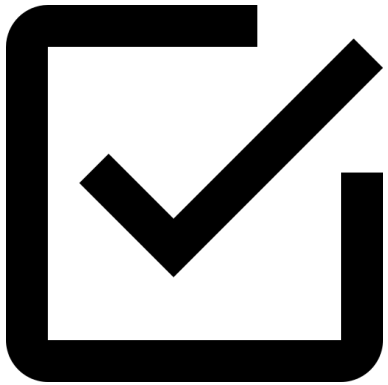
Conclusion and outlook

With the current pre-release, it is possible to model and generate the code framework for a complex multicore safety application.

The model development for the demonstrator showed, that the graphical representation realized in Sirius for functional, software and hardware level is extremely helpful to understand the system behavior. The EMF Forms solution for entering detailed data is fast and intuitive.

The generated code using the Acceleo extension is readable and performant.

A first stand-alone package has been generated and is currently in the stabilization phase



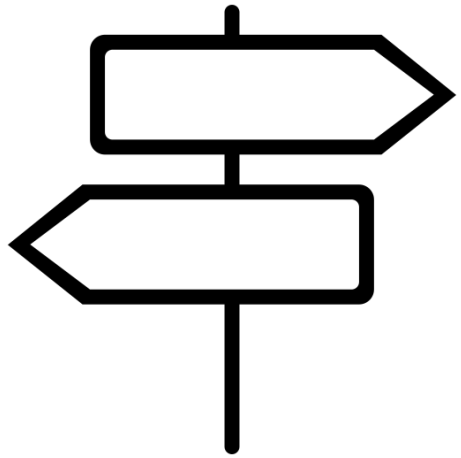
Conclusion and outlook

Future work will include:

Requirements modeling
A meta-model for safety requirements will be defined which allows the definition of safety-goals as well as mapping to the actual implementation.

Advances safety assessment
Right now only fundamental assessments are possible.

PxROS* configuration
The model contains enough information to configure PxROS* tasks.
In future implementations, the tasks will be generated by the tool.



*PxROS is a RTOS by HighTec

Contact

M.Sc. Thomas Barth
Prof. Dr.-Ing. Peter Fromm

Hochschule Darmstadt - University of Applied Sciences
FB Elektrotechnik und Informationstechnik (EIT)
Birkenweg 8
64295 Darmstadt

Email: thomas.barth@h-da.de; peter.fromm@h-da.de
Web: www.eit.h-da.de