Project Number: 257909

SPRINT

Software Platform for Integration of Engineering and Things

Collaborative project
Information and Communication Technologies

D4.14 Prototype of physical devices integration

Start date of project: October, 1st 2010
Duration: 42 months

Deliverable	Contract and Sensor and Actuator integration services definition		
Confidentiality	PU	**Deliverable type**	P
Project	SPRINT	**Date**	2014-02-7
Status	FINAL	**Version**	1.0
Contact Person	Michael Wagner	**Organisation**	Fraunhofer FOKUS
Phone	+49 30 3463 7391	**E-Mail**	Michael.Wagner @fokus.fraunhofer.de

THE RESEARCH LEADING TO THESE RESULTS HAS RECEIVED FUNDING FROM THE SVENTH FRAMEWORK PROGRAMME UNDER GRANT AGREEMENT N° 257909

AUTHORS TABLE

Name	Company	E-Mail
Michael Wagner	Fraunhofer FOKUS	Michael.Wagner@fokus.fraunhofer.de
Björn Riemer	Fraunhofer FOKUS	Bjoern.Riemer@fokus.fraunhofer.de
Massimiliano Dangelo	ALES s.r.l.	massimiliano.dangelo@ales.eu.com

CHANGE HISTORY

Version	Date	Reason for Change	Pages Affected
1.0	2014-02-07	Final Version	all

CONTENT

Content of Figures

Content of Tables

Content of Appendix

None.

1 Introduction

This document is the final public version of the restricted "D4.13 Prototype of physical devices integration" to be made public at the end of the project.

1.1 Overview and Purpose of the Document

The purpose of this document is to describe the prototype of physical devices integration. Here the focus of the document is to give an overview and a guide to the prototype.

This document is based on the input from:

- D.2.1 SPRINT Requirements
- D.4.9 Contract and Sensor and Actuator integration services definition

It also conforms to the input from:

- D.5.4 Architectural principles for Internet of System Design and the IoT

1.2 Scope of the Document

The scope of the document is on the description of the prototype of physical devices integration.

The scope of this document contains an:

- Overview on the devices of the prototype
- Overview on implementation details

2 SPRINT Requirements

This section contains all relevant requirement input for this document. This includes high level requirements, use cases and technical requirements.

2.1 High Level Requirements

This section lists all in WP2 identified requirements relevant for this document.

ID	Description	
6.1.8	The SPRINT platform shall adopt standard or de-facto standard technologies wherever available.	
6.1.9	The communication among the components of the SPRINT platform shall be based on RESTful web services.	
6.2.2	Meta-models description shall support standards technologies such as EMF (Eclipse Modelling Framework), Ecore, Resource Description Framework (RDF) and (Web Ontology Language) OWL.	
6.2.5	All resources shall be accessible via a URI and have a single owner.	
6.4.6	The integration of physical devices and System designs shall be done using semantic mediation.	
6.5.3	The SPRINT platform shall allow for the verification of contract satisfaction by using contract monitoring of deployed physical devices.	
7.1.3	SPRINT Engineering Environments shall allow for co-simulation of models with HiL (Physical Devices)	
7.1.4	The communication among the components of the SPRINT platform shall be firewall friendly.	
8.1.1	SPRINT Engineering Environment shall allow for contract monitoring\analysis of Physical Devices.	
8.1.2	SPRINT Engineering Environment shall allow for remote monitoring of Physical Devices.	
8.1.3	Physical SPRINT devices shall be REST compliant.	
8.1.4	SPRINT physical devices shall communicate via the internet to SPRINT tools.	

Table 2-1: SPRINT requirements considered in this deliverable

2.2 Goal

The goal is that physical devices like sensors and actuators can be as easily integrated into all the phases of the developments as normal design artifacts (e.g. models).

2.3 Technological Requirements

The main technological requirements are already given through the requirements of D2.1:

- Rest full and HLA communication
- Fire wall friendly communication over the internet
- Support for integration via semantic mediation
- URI for accessing elements
- Allow for querying and browsing (basic internet capabilities)

3 Overview on Devices

This section gives an overview on sensors, actuators and contracts in order to better understand the challenges and innovation in the integration.

3.1 Tinkerforge

The heterogeneity of physical devices may lead to ad-hoc integration so that an abstraction for commonalities is needed. This includes the identification of classes of physical device that can be treated in a similar way.

Tinkerforge is a platform of stackable microcontroller building blocks (Bricks) that can control different modules (Bricklets).

Bricks can evaluate measurements, control motors and communicate with other building blocks. Each Brick has a 32-Bit ARM microcontroller, a USB connector and connectors for more Bricks and Bricklets.

Bricklets extend the features of Bricks. They provide means for in- and output of data.

3.1.1 Blocks

The heterogeneity of physical devices may lead to ad-hoc integration so that an abstraction for commonalities is needed. This includes the identification of classes of physical device that can be treated in a similar way.

3.1.1.1 Servo Motor Sensor

The Servo Motor Sensor is implemented by the Tinkferfoge Servo Brick. This Brick is used to control up to seven RC servos.

Figure 1 The Tinkerforge Servo Brick

Name	Type	Direction	Description
Servo	Float	Input / Output	The current velocity of the servo motor. Values are in range from -9000 and 9000

3.1.1.2 IR Temperature Sensor

The IR Temperature Sensor is implemented by the Tinkerfoge Bricklet. This Bricklet is equipped with an infrared thermometer and is used to measure the ambient temperature.

Figure 2 The Tinkerforge IR Temperature Bricklet

Name	Type	Direction	Description
IR Temperature	Float	Output	The current ambient temperature in a range from -40°C to 85°C

3.1.1.3 Humidity Sensor

The Humidity Sensor is implemented by the Tinkerforge Humidity Bricklet. This Bricklet is used to measure the relative humidity.

Figure 3 The Tinkerforge Humidity Bricklet

Name	Type	Direction	Description
Humidity	Float	Output	The current relative humidity in the range of 0% to 100%

3.1.1.4 Accelerometer Sensor

The Acceleromter Sensor is implemented by the IMU Brick. This Brick is equipped with an internal measurement unit. It has nine degrees of freedom and consists of a 3-axis accelerometer, magnetometer and gyroscope. The IMU Brick is used to measure the acceleration.

Figure 4 The Tinkerforge IMU Brick

Name	Type	Direction	Description
Accelerometer	Float	Output	The current acceleration in X- /Y- /Z-Direction. Values in the range from -400°/s to 400°/s

3.1.1.5 LED Sensor

The LED Sensor is implemented by the Tinkerfoge IO-16 Brickklet. This Bricklet features 16 I/O pins where each pin can independently be configured.

Figure 5 The Tinkerfoger IO-16 Bricklet

Name	Type	Direction	Description
Pin	Float	Input / Output	The value is 1 if the pin is turned on. The value is 0 if the pin is turned off.

3.1.1.6 WIFI Extension

The WIFI Extension is used to control Bricks and Bricklets wirelessly.

Figure 6 The Tinkerforge WIFI Extension

3.1.2 Devices

The heterogeneity of physical devices may lead to ad-hoc integration so that an abstraction for commonalities is needed. This includes the identification of classes of physical device that can be treated in a similar way.

3.1.2.1 TinkerCar

The TinkerCar is a toy car extended with several Tinkerforge sensors. It makes use of the Tinkerforge Servo Motor Bricklet, the IR Temperature Bricklet, the Humidity Bricklet, the IMU Brick and the WIFI Extension.

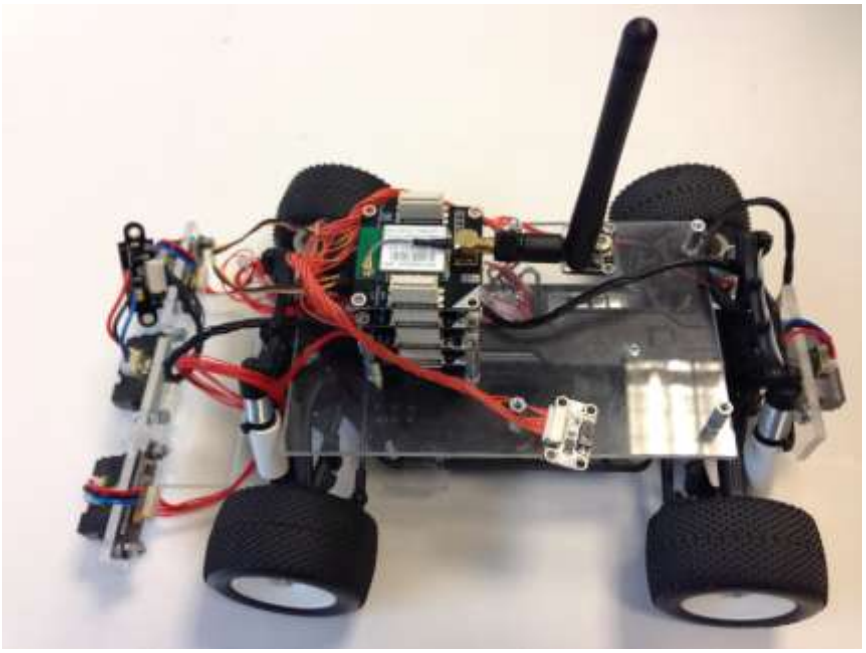

Figure 7 The TinkerCar build with the Tinkerforge sensors

Name	Type	Direction	Description
Velocity	Float	Input / Output	The current velocity of the TinkerCar. Values are in range from -6000 (full speed backwards) and 6000 (full speed forwards)
Steering	Float	Input / Output	The current Steering of the TinkerCar. Values are in range from -6000 (full left side steering) and 6000 (full right side steering)

IR Temperature	Float	Output	The current ambient temperature in a range from -40°C to 85°C
Humidity	Float	Output	The current relative humidity in the range of 0% to 100%
Accelerometer	Float	Output	The current acceleration of the TinkerCar in X- /Y- /Z-Direction. Values in the range from -400°/s to 400°/s

3.1.2.2 TinkerGate

The TinkerGate makes use of the Tinkerforge Servo Bricklet in order to open and close the gate and it makes use of the Tinkerforge IO-16 Bricklet in order to control the traffic lights

Description:

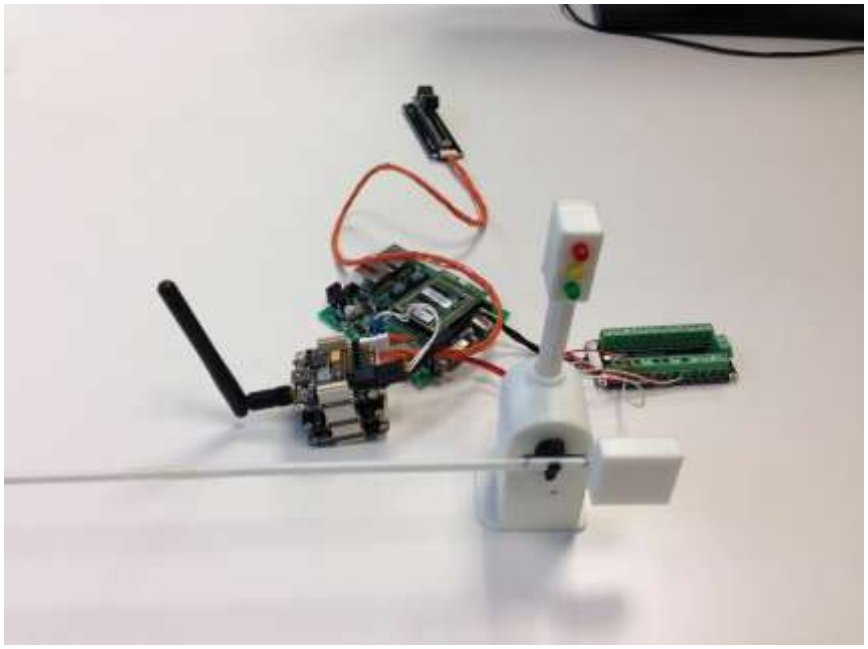

Figure 8 The TinkerGate

Name	Type	Direction	Description
Position	Float	Input / Output	The relative position of the gate. Values are in range from 0 (gate is closed) to 100 (gate is completely open).
GateIsOpen	Boolean	Output	Indicates whether the gate is open (true) or not (false). The value equals 1 while position is greater than 0.
OpenCmd	Boolean	Output	Value is false if the gate is closing and true if the gate is opening.
Light	Float	Input / Output	Changes the current light to red (value equals 0), yellow (value equals 1) or green (value equals 2).

3.2 Android

The heterogeneity of physical devices may lead to ad-hoc integration so that an abstraction for commonalities is needed. This includes the identification of classes of physical device that can be treated in a similar way.

Android is an operating system based on the Linux kernel and is designed primarily for mobile devices like smartphones and tablet computers. Android offers a consistent interface to interact with sensors in all devices, which are often produced by different manufactures.

Figure 9 The Morola Milestone 2 Android device

Name	Type	Direction	Description
Acceleration	Float	Output	The acceleration of the device in X- /Y- /Z-Direction. Range of values depends on devices.
Magnetic Field Sensor	Float	Output	The magnetic field sensor is used to capture the device's orientation.

.

4 Overview on implementation details

This section gives an overview to the implementation details of the physical device integration not mentioned earlier in other deliverables (e.g. D4.9 Contract/Sensor/Actuator integration services definition). This section is split up in two sub sections concerning design and runtime implementation details.

4.1 Implementation of Integration for Design Time

This subsection describes the design time implementation details, this is how the physical devices register themselves in the sprint engineering environment.

4.1.1 Tinkerforge

Registration of Tinkerforge physical devices on the sprint engineering environment is implemented via the proxy. The proxy uses the Tinkerforge API which is communicating with the Tinkerforge master via TCP in order to collect all meta-information on the connected bricklets and extensions. The retrieved information is converted by the Registrator in RDF and BSO representation and then posted via HTTP to the sprint engineering environment.

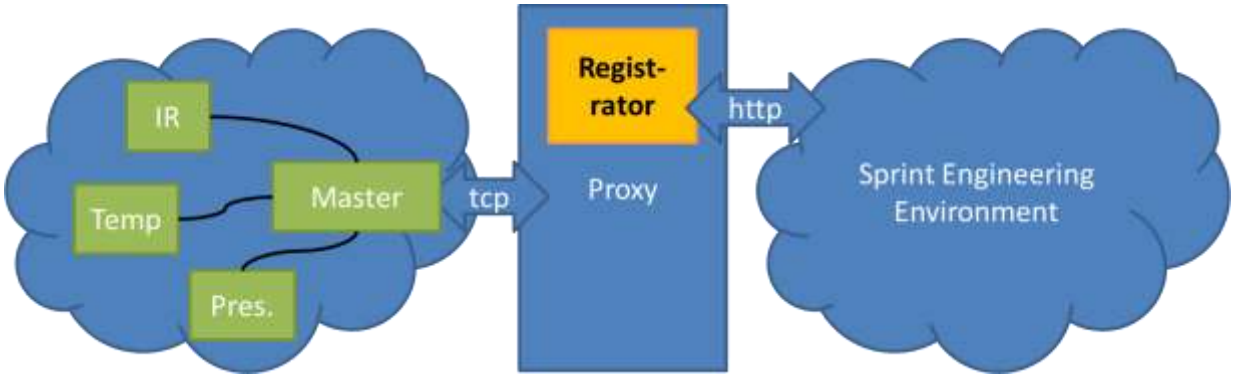

Figure 10 Integration at Design Time Tinkerforge

4.1.2 Android

Registration of the android device and its sensors on the sprint engineering environment is implemented via a Registrator running on the android device itself. The proxy uses the android API in order to collect all meta-information on the available sensors. The retrieved information is converted by the Registrator in RDF and BSO representation and then posted via HTTP to the sprint engineering environment.

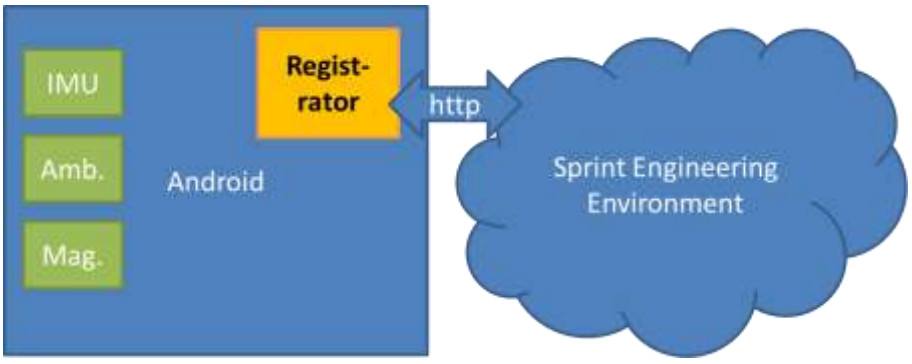

Figure 11 Integration at Design Time Android

4.2 Implementation of Integration during Simulation Time

This subsection describes the run time implementation details, meaning how the physical devices make their live data available (e.g. for simulation).

4.2.1 Tinkerforge

Delivering live information to and from the Tinkerforge physical devices to the sprint engineering environment is implemented through a the proxy due to resource constraints. The proxy uses the Tinkerforge API which is communicating with the Tinkerforge master via TCP in order to collect and post all live data on the connected bricklets and extensions. The information is then posted via TCP to the HLA server using the jcerti API.

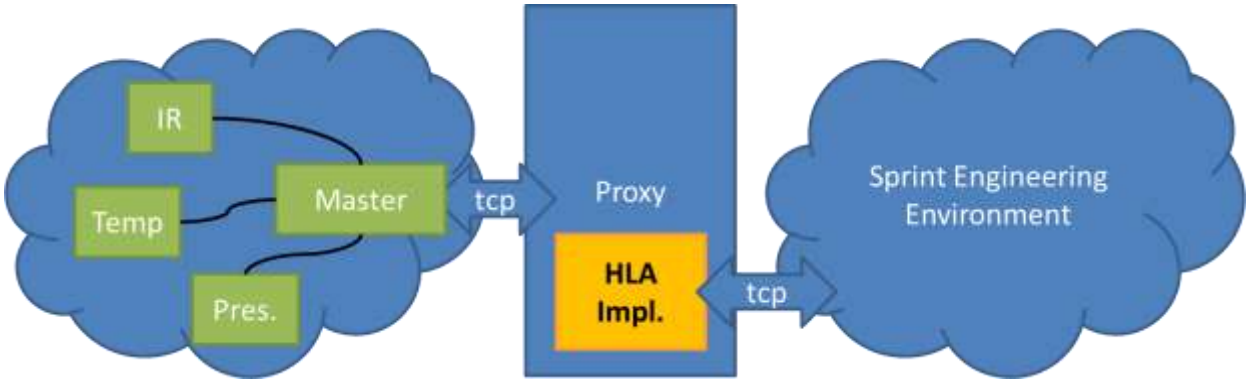

Figure 12 Integration during Simulation Time Tinkerforge

4.2.2 Android

Delivering live information to and from the Android physical device to the sprint engineering environment is implemented via a proxy due to resource constraints. The proxy uses the Android API in order to collect and post all live data available on the device to the proxy via http for forwarding. The information is then posted via TCP to the HLA server using the jcerti API.

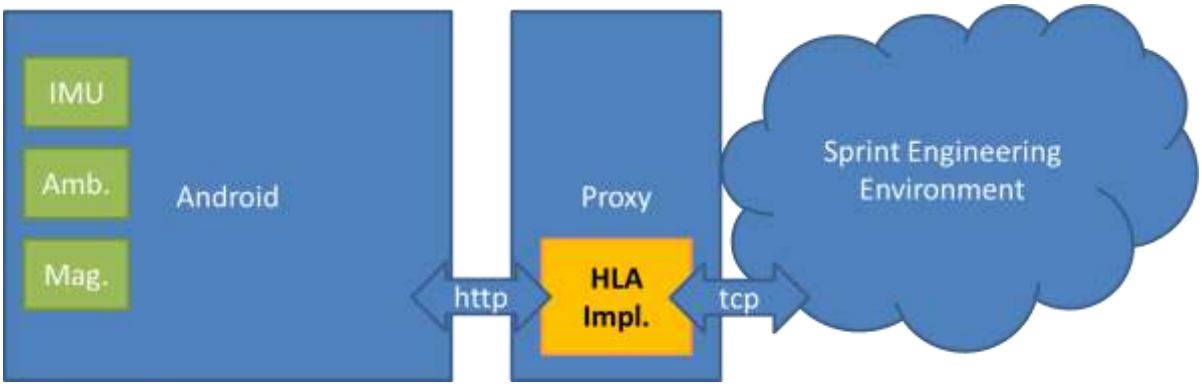

Figure 13 Integration during Simulation Time Android

5 Abbreviations and Definitions

Architecture	The fundamental organization of a system (maybe on different abstraction levels) embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.
Artefact	A physical piece of information that is used or produced by a software development process. Examples of Artefacts include models, source files, scripts, and binary executable files.
Attribute	A piece of information associated with an entity. Used to describe a characteristic of an entity. In the meta-model, an attribute is a placeholder for the actual information that is defined in the models.
BSO	Basic Structural Ontology, Ontology introduced by the Sprint project and described in [D3.2]
Entity	An object with an identity that is distinguishable from other entities.
Error	Discrepancy between a computed, observed or measured value or condition and the true, specified or theoretically correct value or condition.
Extensible Markup Language	Is a set of rules for encoding documents in machine-readable form defined by W3C. (See: http://www.w3.org/TR/2008/REC-xml-20081126/)
Instance	An entity that is obtained from another entity by the process of instantiation.
Interface	Abstraction of a service that only defines the operations supported by that service (publicly accessible variables, procedures, or methods), but not their implementation.
Link	Representation of a relation between two objects.
Metamodel	A model for describing (structure of) other models on a meta level.
Model	A semantically closed abstraction of a software or hardware system, i.e. a simplification of reality that gives a complete description a system from a particular perspective.
Open Services for Lifecycle Collaboration	Open Services for Lifecycle Collaboration (also known as OSLC or Open Services) is a community and set of specifications for Linked Lifecycle Data. The community's goal is to help product and software delivery teams by making it easier to use lifecycle tools in combination. (See: http://open-services.net/html/Home.html)
OSLC	See: Open Services for Lifecycle Collaboration
RDF	See: Resource Description Framework
Resource Description Framework	It's a family of W3C specifications for conceptual description or modelling of information that is implemented in web resources. (See: http://www.w3.org/TR/rdf-primer/)
System	A collection of components organized to accomplish a specific function or set of functions.
Traceability	Traceability is a technique used to provide relationships between objects (e.g. in requirements, design and implementation of a system in order to manage the effect of change).

View	A representation of a whole system from the perspective of a related set of concerns.
W3C	See: World Wide Web Consortium
World Wide Web Consortium	Is the main international standardization organization for the World Wide Web. (See: http://www.w3.org)
XML	See: Extensible Markup Language

6 References

[D3.2] D3.2 Definition of the Semantic Services Integration Layer

[D4.9] D4.9 Contract/Sensor/Actuator integration services definition

[D5.4] D5.4 Architectural principles for Internet of System Design and the IoT, Sprint Deliverable