Project Deliverable

Project Number:	Project Acronym:	Project Title:
287901	BUTLER	uBiquitous, secUre inTernet-of-things with Location and contEx-awaReness

Instrument:	Thematic Priority
Integrated Project	Internet of things

D2.4 - Selected technologies for the BUTLER platform

Contractual Delivery Date:	Actual Delivery Date:
March 2013	April 2013

Start date of project:	Duration:
October, 1st 2011	36 months

Organization name of lead contractor for this deliverable:	Document version:
CEA	V 1.0

Dissemination level (Project co-funded by the European Commission within the Seventh Framework Programme)

PU	Public	**X**
PP	Restricted to other programme participants (including the Commission	
RE	Restricted to a group defined by the consortium (including the Commission)	
CO	Confidential, only for members of the consortium (including the Commission)	

Authors (organizations) :

Christine Hennebert, Benoit Denis (CEA)

Franck Le Gall, Bertrand Copigneaux, Fabrice Clari (Inno)

Francesco Sottile, Francesco Mauro (ISMB)

Philippe Smadja (Gemalto)

Stefano Pascali (ST-I)

Davy Preuveneers, Arun Ramakrishnan (KUL)

Juan Sancho (TST)

Anup Shrestha (ZigPos)

Massimo Valla (TIL)

Maria Fernandez Salazar, Miguel-Angel Monjas (Ericsson)

Davide Macagnano (UOulu)

Jani Korhonen (Cascard)

Reviewers (organizations) :

Foued Melakessou, Thibault Cholez (UL)

Abstract :

This document aims to present the candidate technologies able to achieve the BUTLER platform. The selected ones must cope with the BUTLER requirements, and communicate together to realize an integrated platform supporting the proof-of-concepts. In the first chapter, the BUTLER components are defined to avoid any ambiguity and the BUTLER architecture is overviewed. The chapter 2 is dedicated to the new technological bricks developed and analysed for the BUTLER project taking into account the context-aware constraints. Security, geo-localization and human behaviour schemes are studied and innovative methods and algorithms are presented. The chapter 3 details Smart Object technologies that are used in the network. Some of these objects may integrate technological bricks. The chapter 4 is dedicated to the Smart Mobile. The HTML5 technology appears as the most relevant one and the motivations to make this choice are discussed. The Smart Server is presented in chapter 5. In chapter 6, we show how all these technologies communicate together in order to realize a unified BUTLER platform that will be the support of the horizontal scenario. An example of such a scenario is briefly explained at the end of the document.

Keywords :

Technology, BUTLER devices, Smart Object, Smart Mobile, Smart Server, Integrated platform, BUTLER field trial.

Disclaimer

Revision History

The following table describes the main changes done in the document since it was created.

Revision	Date	Description	Author (Organisation)
0.1	13-01-28	Initial TOC	Ch. Hennebert – CEA
0.2	13-02-14	Rev after 1st conf call,	Ch. Hennebert – CEA
0.3	13-02-14	Rev of section 2 after WP2 conf call	Fr. Sottile – ISMB
0.4	13-03-12	Rev after plenary meeting at Madrid	Ch. Hennebert – CEA
0.5	13-04-02	Rev after gotomeeting	Ch. Hennebert – CEA
0.6	13-04-09	Rev after gotomeeting	Ch. Hennebert – CEA
0.7	13-04-16	Rev after gotomeeting	Ch. Hennebert – CEA
0.9	13-04-23	Ready for Review	Ch. Hennebert – CEA
1.0	13-04-25	Review (UL)	F. Melakessou T.Cholez – UL

Acronyms & Abbreviations

3DES	Triple Data Encryption Standard
3GPP	The Third Generation Partnership Project
A3	Algorithm 3, authentication algorithm; used for authenticating the subscriber
A5	Algorithm 5, cipher algorithm; used for enciphering/deciphering data
A8	Algorithm 8, cipher key generator; used to generate Kc
AAA Server	Authentication, Authorization, and Accounting. An entity that provides an authentication service to an authenticator. RADIUS is an AAA server
ACL	Access Control List
ADC	Analog to Digital Converter
AES	Advanced Encryption Standard
AKA	Authentication and Key Agreement
AMF	Authentication and Key Management Field to allow handling of multiple authentication algorithms and keys (AKA). This field allows operators to personalize the results of the AKA f1 function. The value of this property should be between 0 and 65535
ANSI	American National Standards Institute
ANT	ANT Network Protocol
API	Application Programming Interface
APNS	Apple Push Notification service
APS	Application Support
AS	Authorization Server
ASL	Apache Software License
AuC	Authentication Centre. It is the GSM network element that provides the authentication triplets for authentication the subscriber
Authenticator	The component that initiates the EAP authentication
AUTN	AKA parameter. AUTN is the network authentication value generated by the AuC which together with the RAND authenticates the server to the peer, 128 bits.
BBC	British Broadcasting Channel

BRMS	Business Roule Management System
BSD	Berkeley Software Distribution
BSF	Bootstrapping Server Function
CA	Certification Authority
CAN	controller area network
CAP	Compact Application Protocol
CEP	Complex Event Processing
CIS	Circular Interval Scaling
CLS	Cooperative Links Selection
COAP	Constrained Application Protocol
CPU	Central Processing Unit
CRC	Cyclic redundancy check
CRLB	Cramér–Rao lower bound
CRUD	Create Read Update Delete
CSS	Cascading Style Sheets
DC	Distance Contraction
DC	Direct Current
DNS	Domain Name System
DOM	Document Object Model
DPWS	Device Profile for Web Services
DTLS	Datagram Transport Layer Security
EAP	Encapsulated Authentication Protocol
EEPROM	Electrically Erasable Programmable Read-Only Memory
EET	Energy Efficient Tracking
EJB	Enterprise Java Bean
EKF	Extended kalman Filter
ETM	Embedded Trace Macrocell
EU	European Union
F-FSK	Binary Frequency Shift Keying
FFT	Fast Fourier Transform
FG	Functional Group in the Architecture
FQDN	fully qualified domain name
FRI	Finite Rate of Innovation
GBA	Generic Bootstrapping Architecture
GCC	Gnu C Compiler
GCLABN	Globally Connected Locally Autonomic Bayesian Networks
GCM	Google Cloud Messaging
GIS	Geographic Information System
GNU	GNU's Not Unix
GPIO	General Purpose Input Output
GPRS	General Packet Radio Service
GPS	Global Positioning System
GUI	Graphical User Interface
GW	GateWay
HLR	Home Location Register
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secured

HVAC	heating, ventilation, and air conditioning
IBM	International Business Machines Corporation
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IMM	Interacting Multiple Model
IMS	Infrastructure Management Service
IMSI	Intenational Mobile Subsciber Identity
IoT	Internet of Things
IoT-A ARM	Architectural Reference Model from FP7 project IoT-A
IP	Internet Protocole
IPSec	Internet Protocole Security
IR-UWB	Impulse-Radio Ultra-Wideband
I-SCAL	Interval Scaling
ISIM	IP Multimedia Service Identity Module
IT	Information Technologies
JS	JavaScript
JSON	JavaScript Object Notation
JSP	Java Server Pages
JTAG	Joint Test Action Group
LAN	Local Area Network
LED	light-emitting diode
LoWPANs	Low Power Wireless Area Networks
LQI	Link Quality Indicator
LS	Least Squares
LSEND	Lightweight Secure Neighbour Discovery
LSP	Linux Support Package
MAC	Media Access Control (address) Alternatively: Message Authentication Code
MCU	Micro Controller Unit
MDS	Multidimensional Scaling
MEMS	Microelectromechanical systems
MII	Media Independent Interface
MIM	M2M Identification Module
MIME	Multipurpose Internet Mail Extensions
MIT	Massachusest Institute of Technology
MMU	Memory Management Unit
MQTT-S	Message Queue Telemetry Transport - Sensor
MVC	Model View Controller (design pattern)
NAF	Network Application Functions
NDRNG	Non Deterministic Random Number Generators
NFC	Near Field Communication
NIST	National Institute of Standards and Technology (US)
NLoS	Non Line of Sight
NWK	Network
OASIS	Organization for the Advancement of Structured Information Standards
OAUTH 2.0	open standard for authorization 2.0
OGC	Open Geospacial Consortium

OMA NGSI	Open Mobile Alliance - Next Generation Service Interface
ORM	Object Relational Mapping
OS	Operating System
OSI	The Open Systems Interconnection model is a product of the Open Systems Interconnection effort at the International Organization for Standardization
OTA	Over the air programming
OWL	Web Ontology Language
PAN	Personal Aera Network
PCSC	Personal Computer/Smart Card
PF	Particle Filter
PHY	Physical Layer of the OSI model
PKI	Public Key Infrastructure
PLC	Programmable Logic Controller
PLR	Packet Loss Rate
PoC	Proof of Concept
PRNG	Pseudo Random Number Generators
RAM	Random Access Memory
RDBMS	Relationship Data Base Management System
REST	Representational State Transfer
RF	Radio Frequency
RFID	Radio Frequency Identification
RH	Relative Humidity
ROM	Read Only Memory
RSA	Ron Rivest, Adi Shamir and Leonard Adleman, Security Algorithm
RSSI	Received Signal Strength Indication
RT-TOF	Round Trip - Time of Flight
SAML	Security Assertion Markup Language
SD	Secure Digital
SDK	Software Development Kit
SEP	Smart Energy Profile
SIM	Subscriber Identity Module
SLP	Service Location Protocols
SMDS	Super Multidimensional Scaling
SNR	Signal to Noise Ratio
SOAP	Simple Object Access Protocol
SPA	Single Page Application
SPI	Serial Peripheral Interface
SQL	Structured Query Language
SSDP	Simple Service Discovery Protocol
SSL	Secure Sockets Layer
TLS	Transport Layer Security
TOA	Time of Arrival
TV	TeleVision
UART	Universal Asynchronous Receiver/Transmitter
UDP	User Datagram Protocol
UE	User Equipment
UI	User Inteface

UPnP	Universal Plug and Play
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USB	Universal Serial Bus
USIM	Universal Subscriber Identity Module
UWB	Ultra Wideband
VDC	Volts of Direct Current
WADL	Web Application Description Language
WEKA	Waikato Environment for Knowledge Analysis
WLS	Weighted Least Square
WSN	Wireless Sensor Network
XML	eXtensible Markup Language
ZC	Zigbee Coordinator
ZDO	ZigBee Device Object
ZED	ZigBee End Device
ZR	ZigBee Router

Executive Summary

The BUTLER project aims at designing a wide and heterogeneous context-aware Internet of Things network whose goal is to offer new services and applications to the citizens via their Smart Phone (called Smart Mobile) according to their actual environment, hobby or behaviour.

The understanding of what the context is and how it can be used allows designing new and innovative technological bricks for the future objects of the Internet of Things (IoT). A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user’s task. The choice of a relevant context has led to differently handle the security and the privacy topics, to tackle the question of the geo-temporal localization or to highlight the need of modelling the user behaviour. A context-aware architecture has been proposed to integrate the features for new context-aware applications and services. This architecture is based on three main components: the smart objects providing sensing information about their environment or able to perform a simple action, the smart mobiles that present synthetic and relevant information to the users, and the smart servers handling wide computational and memory resources which orchestrate the whole system.

Many technological bricks based on the context of a given object have been proposed. In this document, we should evaluate how they fit with the BUTLER objectives and achieve interoperability and how they can be integrated into various heterogeneous smart objects to provide new and secure services for the final user.

The Smart Objects may sense physical data of the environment. They can be actuators. They wirelessly communicate in a secure way via a low power standard. They may handle a light operating system. They are very heterogeneous from one device to another and can operate very differently. All their particularities must be hidden to the other devices. They must handle the features to allow a homogeneous way to communicate with the others entities of the system whatever the data they sense or how they act.

The Smart Mobile devices are owned by a final user. It must provide applications able to use the context they require without having to worry about how the context has been sensed. It can interact with the Smart Server including “the cloud”, or directly with some Smart Objects. It presents consistent and relevant information to the user.

The Smart Server handles high computing capabilities and many memory resources. It can communicate with any standard. We can consider its resources and capabilities as unlimited. It handles also the cloud where the contextual information sensed by the Smart Object is selected to provide to the user relevant context-aware information. Its role is to collect, aggregate and/or interpret information coming from the Smart objects to make it efficient and relevant for the Smart Mobile users.

The technologies are selected according to precise criteria answering to the BUTLER project objectives and the need of interoperability between the various entities to achieve a context-aware end-to-end system. They are then used in a realistic scenario in order to illustrate a day of the future life.

A final BUTLER platform that integrates all the selected technologies is proposed and detailed. Its goal is to enable communications between the BUTLER devices and remote servers in order to provide available new context-aware technological bricks for the developers of applications. The final BUTLER platform will support the proof-of-concept proposed for BUTLER into a unified network.

Table of Contents

Table of figures

Introduction

In the BUTLER project, taking into account the user's context to offer new context-aware services and application is the central objective. The notion of context can be very various and is understood as the user's immediate environment, the user's remote private environment. It can also be a public environment that interacts to provide new services to the citizens. The new, innovative and context-aware services and applications developed for the BUTLER project should be relevant for the final user, adapted to its needs and intuitive to access and to use. Providing these new perspectives to the citizens needs a complete Internet of Things (IoT) network infrastructure composed of different BUTLER devices that interact together thanks to various protocols and communication standards.

In this document, we take an inventory of the various and numerous technologies that allow BUTLER reaching this challenging objective. The proposed technologies are chosen with the focus to be integrated or to take part of the final context-aware BUTLER platform that validates the proof of concept imagined for different scenario of a day of life.

In the first chapter, the notions of Smart Object, Smart Gateway, Smart Mobile and Smart Server are defined. Their respective roles in the network are explained thanks to an example of instantiation of the BUTLER devices to build a possible system. Then, the complete architecture of the system is detailed, implementing these different BUTLER devices and their interactions.

The second chapter presents new technological context-aware bricks studied for the BUTLER project. The notion of the data security, the authentication of the users and the devices, and the privacy are tackled. The security is a crucial element to take into account at the beginning of the project to ensure the protection of people and property and to make adopted the new technologies, services and applications by the citizens. Many contextual applications rely on localization information. In fact, the services and applications proposed to a given user will not be the same depending on whether the user stands in an airport or on a beach. In this document, numerous localization techniques are detailed. They rely on physical measures providing by the Smart Objects. A summary table allows the reader to understand the differences between each technology and to choose what method is the most suitable to a given application type. The notion of single user and group behaviour is also studied in this part.

The three next chapters are dedicated to the technologies selected for the BUTLER devices and to the interactions and communications between them. First of all, in the third chapter, the retained technologies deployed for the Smart Objects are described. The reasons of each choice are discussed. Finally, the selected Smart Objects are heterogeneous in term of size, resources, and capacities to support sensors and/or actuators, communication standards or operating systems. This heterogeneity appears as an asset that allows handling various services or applications to answer to the multiple needs of the final users.

In the fourth chapter, the technologies selected to equip the Smart Mobiles are described. The architecture and the needs linked to the introduction of new context-aware services are argued as the Smart Mobile includes the interface between the network and the final user. The HTML5 technology is highlighted as it appears as the best candidate.

Then, the fifth chapter presents the Smart server technologies and how the Smart Server interacts with the others devices and with the BUTLER infrastructure.

An attempt to integrate all these elements to build a unique and complete BUTLER platform composed of interacting BUTLER devicesing, is discussed in the chapter 6. A whole security framework is detailed to ensure the end-to-end security on the network and to guarantee the authentication, the security and the privacy of the people and the legitimacy of the devices that compose the system. A concrete and pragmatic scenario illustrates the end of the document. It implements many selected components to show how they can be used and interacted in the complete system, and what the BUTLER platform could be. The various technologies selected for the BUTLER project and presented in this document should be a part of a whole network involving BUTLER devices handling innovative technologies and technological bricks, whose goal is to offer to the citizen relevant context-aware services to improve their life.

1. Definitions

In this section, the three BUTLER Smart platforms: Smart Object/Gateway, Smart Mobile, and Smart Server are defined in order to give a common basis and understanding for the following of the project.

It's important to underline that the terms given here are a way to define *roles* in the BUTLER horizontal platform, not physical implementations.

So for example a Smartphone device used in BUTLER, on which the user runs some BUTLER applications built on top of the Smart Mobile framework, plays the role of BUTLER Smart Mobile; in addition, since the same Smartphone is also able to collect context data, for example location or battery level, then the same device also plays the role of Smart Object. As another example, a home Personal Computer installed in a household, could run some Smart Server component; the same Personal Computer could also provide Smart Object Gateway functionalities, thanks to additional installed software components, to discover and interconnect Smart Objects around the house: in this case the Personal Computer plays both the BUTLER roles of Smart Object Gateway and Smart Server.

The interpretation of the following definitions as of *roles* instead of physical components is intended to guarantee the flexibility to deploy functionalities in different targets scenarios, where different physical devices can be used to implement the various components and roles of the BUTLER horizontal architecture.

1.1. Definition of Smart Object and Smart Object Gateway

The aim of this section is to formulate a formal definition for the Smart Object in the context of the BUTLER project. The *main role of a Smart Object is to interface the BUTLER system with the physical world*, by providing sensing and actuating capabilities.
The simplest instance of a Smart Object is made up of the following components:

- Sensing unit: Measures properties of a thing[1]/group of things of the real world and converts the analogical signals produced by the sensor to digital signals, feeding them into the processing unit;
- Actuator unit: Influence the properties of a thing. Its unit is composed of a system able to convert digital information, coming from the processing unit, to some actions;
- Power Unit. Supplies power by small size batteries which makes the energy a scarce resources;
- Processing Unit: Associates with small storage unit (tens of kilo bytes order). Manages the procedures to collaborate with other nodes to carry out the assigned sensing task;
- Transceiver Unit: Connects the node to the network via various possible transmission media such as infra-light, radio and so on.

[1] One instance of a physical object, living organism, person or concept interesting to a person

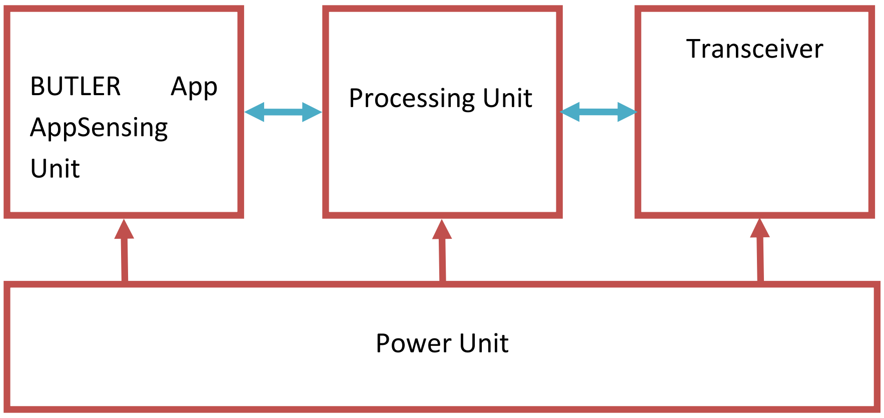


Figure 1: Wireless sensor/actuator scheme

We can image Smart Objects as headless small computers with sensors or actuators (as thermometers, car engines, light switches or industry machinery) and a communication device. Until recently, Smart Objects were realized with limited communication capabilities, such as RFID tags, but the new generation of devices has bidirectional wireless communication and sensors that provide real-time data such as temperature, pressure, vibrations, and energy measurement. Smart Objects can be battery-operated or supplied thanks to solar panels, and typically have a 8-bit or 16-bit micro-controller (sometime even 32-bit MCU), a few tens of kilobytes of memory and a low-power wireless communication device (from a few kilobits/s to a few hundreds of kilobits/s).

Smart Objects gather information in order to provide context-aware services: they will sense data with respect to the physical environment (e.g. accelerometer, lighting, temperature…), the status of smart appliances (e.g. status of a washing machine), the user behavior (e.g. the user motion), etc...

Smart Objects will be implemented on a wide range of physical devices (also called “Things”), from appliances to sensors and actuators. Most of the Smart Objects are characterized by low computational capacities and few available memories, while wireless objects will also have tight power consumption requirements. On such resource constrained object platforms, particular emphasis is put on the efficient implementation of the networking protocol stack.

Smart Objects can handle an operating system and they must be compliant with the networking protocol chosen to communicate with the other Smart Objects and, through Smart Object Gateways, to the other platform components.

Smart Objects communicate using an existing standard which can optionally be enhanced to answer to a dedicated need of the BUTLER project. The most widely adopted standards are IEEE 802.15.4 for PHY/MAC layer, 6loWPan as an adaptation between the IP network layer and the IEEE 802.15.4 MAC layer, and on-going IETF’s COAP, ISA100.11a or Zigbee application profiles for the application layer.

The Smart Objects compliant with IPv6 6LoWPAN technology and also alternative Wireless Sensors Network (WSN) can be considered as long as they embed an IEEE 802.15.4 communication interface.

No user applications or services are run on Smart Objects: they simply provide functionalities related to the physical world without any pre-defined association to the user application requiring the functionality.

To guarantee that all Smart Objects, even those using communication protocols that are incompatible with actual internet standards, can communicate with Smart Mobile and Smart Server platform components, a Smart Object Gateway (or simply Smart Gateway) may be needed.

The *main role of a Smart Object Gateway is to ensure interoperability between a cluster of Smart Objects and other platform components*: Smart Mobile / Smart Server.

The Smart Object Gateway must include a 6LoWPAN “Edge Router” functionality allowing the implementation of a multi-core multi-purpose embedded system. The gateway must also allow the management of the whole WSNs, whatever their technology, for node configuration and reprogramming, node monitoring or real-time evaluation.

The gateway handles several wired and wireless communication standards to interconnect various devices. It achieves networking tasks as: discovery, authorization, security or monitoring. In the OSI model, the gateway assures both physical access, logic access and data access. It could realize protocol adaptation or data translation to guarantee the communication between two sub-networks. It does not handle computational resources or process application data, but it can be used to cache or to aggregate data. The Smart Object Gateway is managed by a local gateway operator.

1.2. Definition of a Smart Mobile

Context-aware applications consider inbuilt requirements for user physical mobility. So, user interfaces should be developed for the mobile user to offer an access to the services provided by the cloud.

The *main role of the BUTLER Smart Mobile is to provide an interface to the user*: in fact through the Smart Mobile platform:

- mobile side applications are executed,
- the user is identified and authenticated,
- information about the user is collected,
- application input from the user is collected,
- relevant data is shown to the user,
- the user is notified about new data important to his situation.

The interaction with other surrounding Smart Objects can also be mediated by the Smart Mobile.

Thanks to the Smart Mobile platform, the integration and the customization of a given BUTLER application should be as little as possible. In short, the *Smart Mobile platform is the basic software* *framework* to provide BUTLER application developers all those components that are common to all BUTLER applications.

Since application developers will need to concentrate on the business logic and behavior of their applications, the Smart Mobile platform will take care of features common to all applications such as: provide common User Interface components and look-and-feel, user registration and authentication, protected access to BUTLER server platform APIs and the notification system.

The Smart Mobile framework will also enable the integration of components such as navigation technologies and data-mining algorithms.

The middleware developed for the Smart Mobile platform must also be designed to ensure the compatibility with the Smart Objects and with the Smart Object Gateways and to interact with the Smart Server platforms APIs to perform application specific functionalities.

Given the fragmented nature of mobile devices, the Smart Mobile platform will also provide the required abstraction form the specific mobile operating systems, thanks to standard technologies such as HTML5. In any case the BUTLER Smart Mobile platform prototype will be developed addressing a specific mobile operating system, but maintaining a design that will allow faster portability to other future operating environments.

Every device running BUTLER applications which are based on this common software framework plays the role of a Smart Mobile.

The Smart Mobile framework will consist of several components as illustrated in the following figure.

- Abstraction layer from the underlying mobile Operating System
- Common functionalities such as:
  - Graphical User Interface common components
  - User Authorization and Authentication
  - Notification system

BUTLER applications will run on top of the Smart Mobile framework and re-use components to simplify and uniform functionalities that are common to all of them.

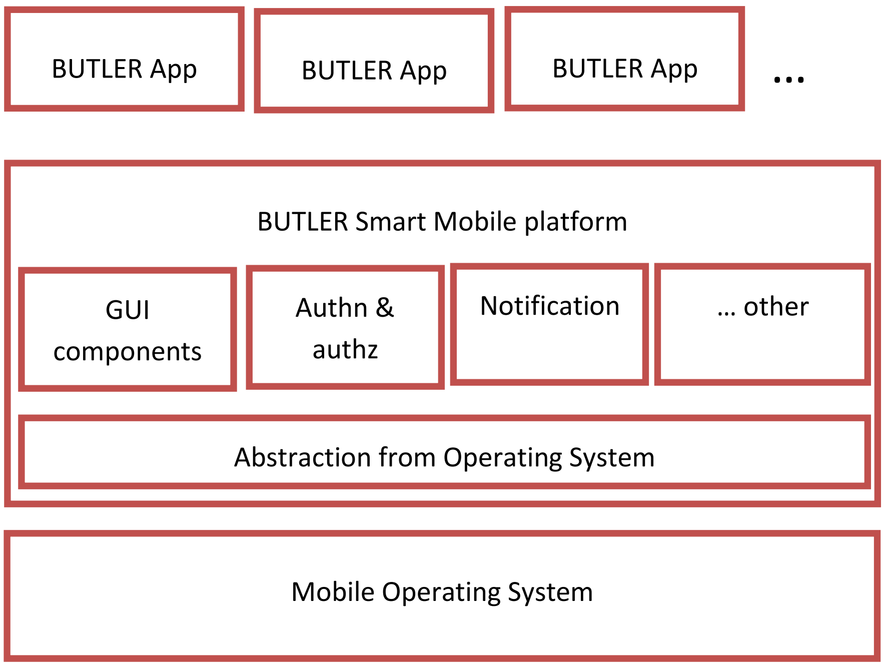


Figure 2: Smart Mobile platform general scheme

The BUTLER Smart Mobile framework will not offer APIs to other entities: the only entry point will be constituted by the notification system that other platform entities could use to alert or wake up applications on the mobile.

1.3. Definition of a Smart Server

In BUTLER a Smart Server is a set of software components that provide functionalities through a set of APIs to applications, SmartObjects/Gateways and SmartMobile.

In short *the role of Smart Server is to provide an interface to the world of digital information*, processing and mediating information that is relevant to the user and the applications he is using.

In the BUTLER horizontal architecture, there will be several Smart Servers that provide different functionalities and APIs: the BUTLER application developer will be able to select the required functionalities and invoke the corresponding APIs.

Smart Servers can provide both specific application functionalities and common functionalities that could be used by several applications. The Smart Server makes available server-side processing for mobile applications, transparent access to the context data collected by Smart Objects (sensors and actuators) and aggregation of information provided by various sources. The Smart Server offers also the possibility to realize data processing and integration in the cloud: this includes a novel combined local and global localization engine, a context management framework, a behavior model and prediction system, market-place/advertising services and security strategies.

It's important to underline that Smart Servers can be deployed in different ways.

Remote or Cloud Smart Servers

Smart Servers components, providing general functionalities and access to information available from different Internet sources and offering a horizontal set of APIs will be deployed in what could be defined as “the BUTLER Cloud”.

The *main role of Remote Smart Servers is therefore to connect Smart Objects/Gateways and Smart Mobile with 3rd party applications or web data sources and to offer general centralized functionalities*.

Local Smart Server

A Smart Server component can also be deployed locally in an environment such as a household, an office building or a train. In this case server side functionalities will be offered again thought a set of APIs that are directly accessible within the local network. Service discovery and security could be offered through the Local Smart Server.

The Local Smart Server could also represent an application level proxy for functionalities available remotely on the cloud, to introduce security, authentication, optimized data transmission and overcome intermittent connectivity with the remote Smart Server.

A uniform programming interface (API) model to develop Smart Server functionalities will be proposed in BUTLER, so that an application should be able, using the same integration technologies, to invoke functionalities and APIs offered by different Smart Server components, and perform the invocation on a Local or Remote Smart Server without any difference.

1.4. Smart Platform roles and physical implementations

As anticipated, the three BUTLER Smart platforms do not consist in a physical implementation of any BUTLER devices. Instead, the definitions given here determine the ***roles*** of the heterogeneous Smart elements in order to build a consistent system achieving end-to-end communications.

In fact, a given electronic device could run any combination of Smart Objects/Gateway, Smart Mobile and Smart Server roles so that the final system offers a consistent context-aware system.

An overview of the different roles and their typical implementation on physical devices along with their connectivity realizing a BUTLER Smart Platform is showed in Figure 3.

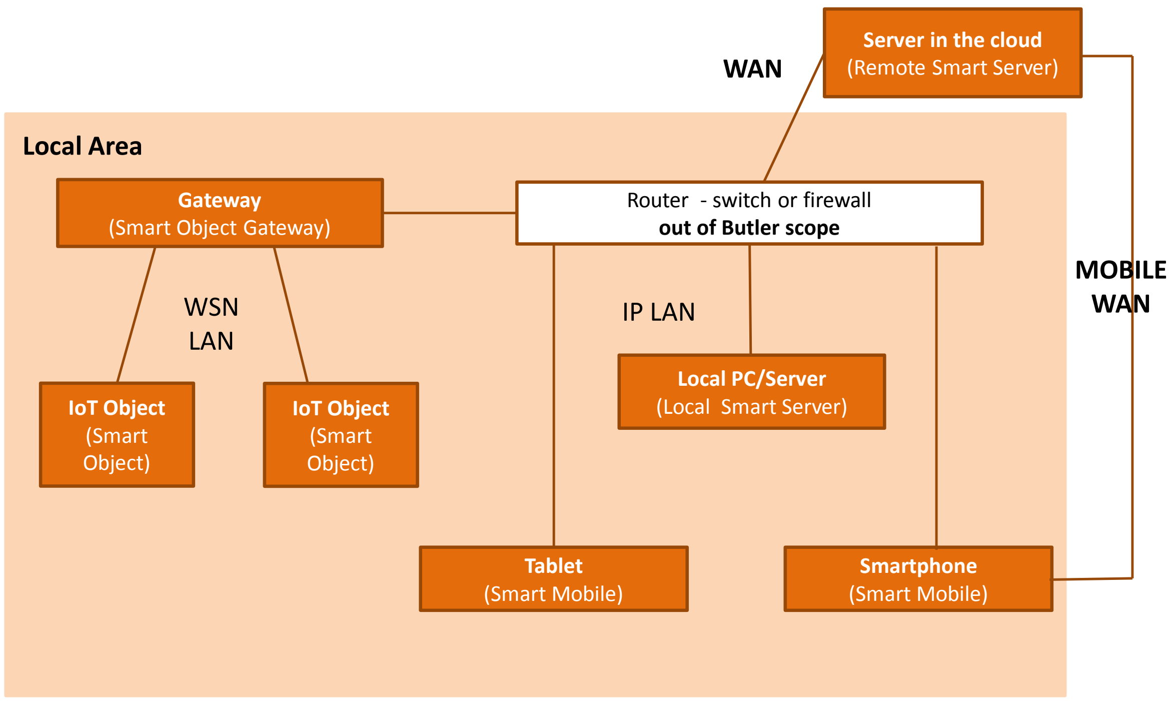


Figure 3: Example of Typical roles (in parenthesis) of the BUTLER devices into a Smart Platform

1.5. Overview of the architecture and interaction

The BUTLER Architecture is composed by a collection of different software functions. They are created to be generic and follow a number of principles such as separation of concerns, loose coupling and composition. The software functions are used to implement composite domain-specific applications for different areas and comprise a number of functional groups described below. Each of them provides different functions and features and can be deployed in any of the different BUTLER entities (Smart Object/Gateway, Smart Server and Smart Mobile).

A number of Functional Groups (FG) are defined and can be seen in the figure below:

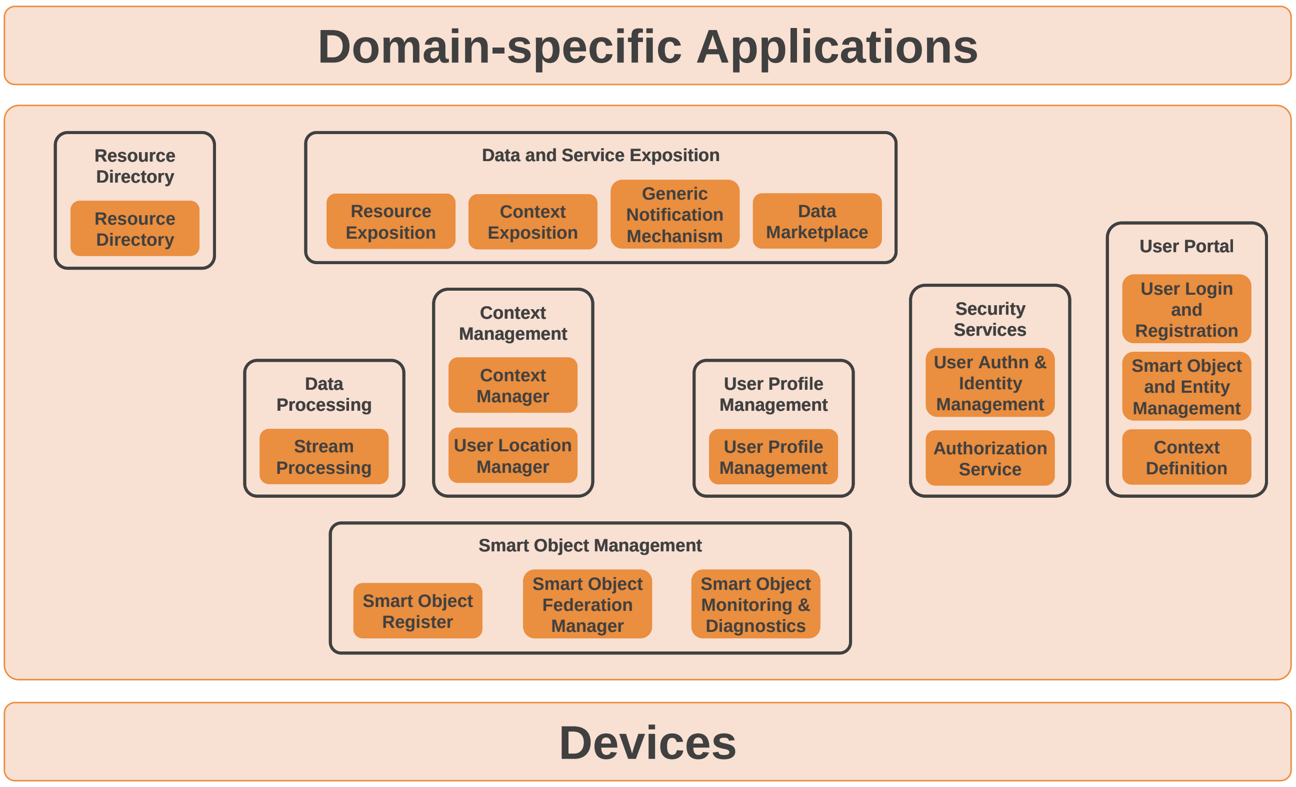


Figure 4: Abstract BUTLER Architecture

Domain-specific Applications (custom business logic for the implementation of specific BUTLER scenarios) and physical devices are not integrated into the BUTLER architecture. Domain-specific Applications are usually deployed in Smart Mobiles, but also can be deployed in the Smart Server. The horizontality mission of BUTLER requires from the domain-specific applications to use the horizontal services provided by BUTLER.

The following Functional Groups (FG) are defined in the BUTLER Architecture:

- **Resource Directory FG**: Takes care of the management (registration, update, deletion) of the resources provided by Smart Objects. On the other hand, it also provides the means for the discovery of said resources, usually through a map-based interface.
- **Data Processing FG:** Provides the means for the handling of large amounts of events provided by Smart Objects and other data sources. On one hand, it supports the dynamic management of context information, so that when the context changes, notifications are delivered to the entities subscribed to those changes. On the other hand, it dynamically distributes incoming data to the domain-specific applications that subscribed to them. It also supports decision-making on said data.
- **Context Management FG**: Provides the means for accessing context and location information related to the entities managed by BUTLER. Access to the context is static (query-response based on actual context value at a given time) or dynamic (subscriptions to notification updates of contextual data or events must be also provided).
- **Data and Service Exposition FG:** Deals with the exposition of the context and resources to domain-specific applications. Other supporting functionalities that can be consumed by the said applications or

by developers in charge of creating BUTLER services are also provided (a marketplace for the discovery and purchase of data sources, a generic notification mechanism and so on).

- **User Profile Management FG**: Provides the means to manage a unified BUTLER user profile across different domain-specific applications.
- **Security Services FG**: Provides the means for authenticated and authorized access to Smart Objects and to the horizontal services provided by BUTLER.
- **User Portal FG**: Provides the means for the different actors in the BUTLER system to perform user registration and login (end-users), management of Smart Objects (smart object administrators) and definition of context (developers).
- **Smart Object Management FG:** Is a variety of tools to manage the Smart Object life-cycle, the association of objects to users and the monitoring and diagnostics of the Smart Objects.

2. Selected enabling technologies

Although not explicitly shown in the architecture, BUTLER requires data storage capabilities for both the actual storage and for the caching of information. Besides information that is managed or generated by the Smart Objects (cache), external data sources must be also considered, especially when they can help to build context information: weather conditions, city-wide energy (gas, water) consumption and forecast, energy bill… Storage is also needed for handling the meta-information (including identifiers) describing Objects and other entities handled by the system.

2.1.1. Architecture

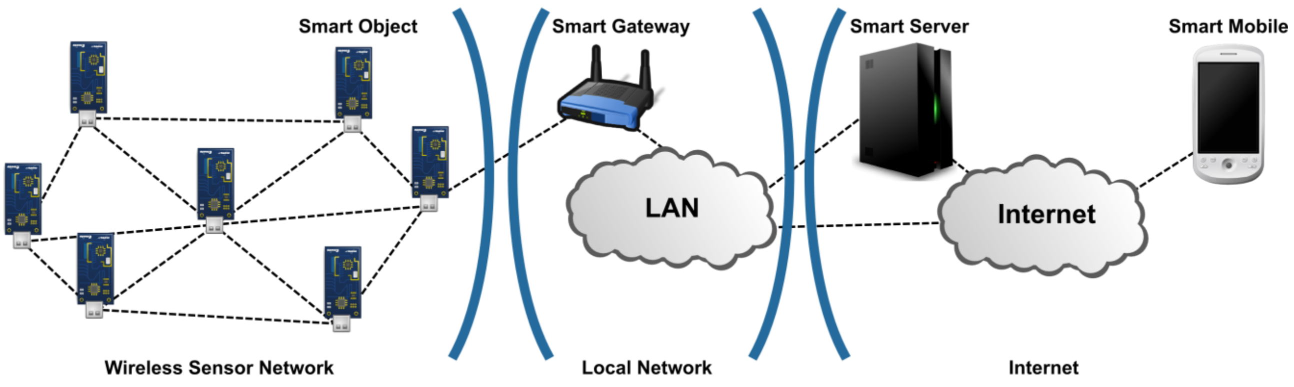


Figure 5 Physical Architecture

We can see three distinct domains where security needs to be handled:

- Wireless sensor network interconnecting smart objects
- Local network
- Internet interconnecting smart mobiles and smart servers

2.1.2. Selected mechanisms for secure communications

ZigBee 2007/SEP 2.0

The ZigBee security architecture supports confidentiality, integrity, authenticity, authority, availability and non-repudiation. Same level of security is supported at the different layers of the reference protocol stack. 128-bits AES based mechanisms are adopted. Entity authentication, access control, key management and establishment functionalities are supported within a ZigBee Network. To this aim, security features have been introduced within the nodes and coordination capabilities have been defined within the ZigBee Coordinator and the Trust Center.

Commercial solutions implementing the full ZigBee 2007 standard (including the security features) are already available in the market and use resource constrained devices. The Trust Center needs to store some key material for all the nodes of the networks and manage some security related operations. This would introduce more strict requirements in terms of storage and processing power. This issue can be addressed by using a more powerful node for the Trust Center or by using a ZigBee Gateway and the provided API to delegate some of the workload related to Trust Center operations to an external more powerful device.

SEP 2.0 mainly focuses on application related aspects. For lower layers, the standard leverages on other existing solutions. The SEP 2.0 architecture is mainly based on public cryptography and adopts IETF security technologies.

Commercial implementation are available in preliminary releases that confirm the suitability of SEP 2.0 for IoT device. More resources are required for SEP 2.0 nodes with respect to ZigBee 2007 nodes. However, the design of the overall solution is being performed in compliance with reference IoT constraints.

The integration of SEP2.0 into the BUTLER architecture at the smart object level will depend on the communication protocol selected for the Wireless Sensor Network. The other main candidate is 6loWPan.

6LoWPAN

Security can be handled by TLS (Transport Layer Security) or DTLS (Datagram TLS). IPSec (IPv6) is possible at the network layer but it consumes a lot of resources. Limitations in 6LoWPANs prevent the use of the full IPSec suite. The management of cryptographic keys using the minimum payload, and the limitation of exchanged messages between nodes are required. LSEND (Lightweight Secure Neighbour Discovery) is an extension of the protocol SEND that permits to secure the Neighbour Discovery mechanisms in 6LoWPAN networks (6LoWPAN layer). AES enables to secure the link layer. Link-layer security inside 6LoWPANs, based on the IEEE 802.15.4 128-bit AES encryption, provides some protection. Authentication can be done in respect with the resurrecting duckling model.

Heavy Security protocols needs to be avoided as much as possible due to the intrinsic low power and computational resources of 6LoWPAN smart objects. Symmetric encryption is privileged. Node and network limitations in 6LowPAN prevent the use of the full IPSec suite, transport layer (“socket“) security or the use of sophisticated firewalls on each node. However DTLS can be used on low power nodes, and it provides a suitable level of end to end security between web clients and embedded web servers.

If 6LoWPAN is selected as the communication protocol for the wireless sensor network, DTLS, LSEND and AES encryption are the related security components that the BUTLER architecture must use.

IPSec

IPSec (Internet Protocol Security) is a standard defined by IETF to assure secure private communications on IP networks. IPSEC is mandatory for IPv6 and optional for IPv4. It provides authentication, confidentiality and integrity at the Network layer. IPSec implementation is CPU and memory consuming. Nevertheless an IPSec stack has been implemented over Contiki OS. While IPSec is not well adapted for Smart Object, communications between Smart Gateway, Smart Server and Smart Mobile can use IPSec. The integration of IPSec in BUTLER will depend if all security needs cannot be fulfilled by other protocols like TLS.

EAP

EAP (Encapsulated Authentication Protocol) runs over Data Link layer. It is designed for use in network access authentication. In the nominal use case, the Peer Identity is sent in clear text, therefore it is subject to privacy issue (an attacker can retrieve the identity) or security (the attacker modifies the identity).

EAP-SIM is an authentication method that uses the authentication engine of a SIM card (A3/A8 algorithm, IMSI/Ki) and the authentication infrastructure of a telecom operator (HLR) to prove user's identity. It provides mutual authentication, confidentiality (Link Layer can use EAP-SIM generated encryption key) and integrity (Link Layer can use EAP-SIM generated MAC key). During the first authentication, the identity of the card is revealed to the network. Next authentication may use pseudonym technique – pseudonym is renewed on each successful authentication. The newly generated pseudonym is located either on the SIM card and the server. As the session keys are generated by the telecom operator, all data (including context information) can be revealed at telecom operator server side. EAP-SIM normally runs in SIM card and relies on symmetric cryptography technique therefore it does not require huge memory. EAP-AKA provides a similar method.

Within EAP-TLS (Transport Layer Security), the security of the TLS protocol is strong, as long as the user understands potential warnings about false credentials. EAP-TLS uses PKI to secure communication to the RADIUS authentication server or another type of authentication server. Mutual authentication can be provided, like also Confidentiality and Integrity. Client authentication is generally not performed because it implies deployment of client side certificate which is a huge and expensive process. The technology requires complex and huge deployment process of client certificates and therefore it is not well suited for M2M. More, the required bandwidth is larger than other technology. Generally, the client authentication is based on RSA technology which is time and CPU consuming. Anyway, message encryption and integrity is performed using symmetric cryptography (3DES, AES) which is well suited for smart objects.

The EAP is therefore best suited to perform the authentication tasks at the Smart Server-Smart Mobile level or at the Smart Gateway-Smart Server level.

TLS

TLS (Transport Layer Security) is a major protocol securing communications on the Internet. The earlier version of the protocol was called Secure Sockets Layer (SSL). Basically, TLS builds a secure tunnel through the network between two hosts. Data packets are encrypted so that only the specified receiver can access the content to prevent eavesdropping, tampering, or message forgery. TLS secures the link from eavesdroppers but if the host is directly compromised, TLS cannot secure the communication. If the client needs to be authenticated to the server, the technology requires complex and huge deployment process of client certificates and therefore it is not well suited for M2M. The CA must be trustworthy and is a single point of failure in the system. It is not suited to secure pervasive and peer-to-peer M2M communications. Moreover, the required bandwidth and resource consumption have not been optimized for M2M communications with strict constraints. However, message encryption and integrity is performed using symmetric cryptography (AES) which is well suited for smart objects. TLS will be integrated in the BUTLER architecture on Smart Gateway, Smart Server and Smart Mobile, securing the communications between these components.

DTLS

DTLS (Datagram Transport Layer Security) provides communications privacy for datagram based networks.It includes a cookie exchange designed to protect against denial of service. Reliable transmissions are achieved by implementing suitable packet loss handling mechanisms and re-ordering techniques. Message encryption and integrity is performed using symmetric cryptography (AES) which is well suited for smart objects. Therefore, DTLS can be integrated to secure SmartObjects communications, especially if the selected routing protocol is 6lowpan.

SAML

SAML (Security Assertion Markup Language) is issued by the OASIS standard in 2005. This is a XML-based framework for federated identity management. Based on standard security schemes, SOAP message security model, XML Encryption and XML Signature. “XML Encryption” is not very often used during the exchanges of information. Its implementation relies on Secure Transport like SSL/TLS. Nevertheless, if XML Encryption is not used, system shall take care about the security at the user side - the SAML messages will be clear at the client side, so it could be retrieved by attackers. If Encryption is not used, it is recommended to support at least Signature of the messages. The messages are full-text and do not require too much processing for parsing. The signature algorithm is a symmetric one and easy to compute by small embedded systems. SAML could be an alternative to EAP to perform the authentication tasks at the Smart Object level.

OpenAuth

OAuth is an open protocol that allows an application to access protected resources from a Resource Provider. It is not really an identity management protocol but used as if by some systems (Google, FaceBook). In this case, the Resource Provider plays the role of Identity Provider. Used as an identification protocol, the security of this protocol is very weak because one can reuse an access token obtained for a given application to log into another one. Security relies completely on the HTTPS protocol. OAuth 2.0 requires some TLS features which may require a cryptographic coprocessor on small embedded systems. This can be implemented with Security Element as a Smart Card. SAML could be another alternative to EAP to perform the authentication tasks at the SmartObject level.

GBA

GBA (Generic Bootstrapping Architecture) provides an application free mechanism able to build a shared secret between a user agent (generally running on a mobile phone) and a server. This shared secret enables client-server authentication. GBA relies on 3GPP AKA (Authentication and Key Agreement) used in 3G networks (USIM application), and also used in IMS (Infrastructure Management Services) infrastructure. GBA may also rely on 2G network infrastructure via ISIM application. GBA provides mutual Authentication of Card and BSF, mutual Authentication of BSF and NAF, NAF-UE Confidentiality (Link Layer can use GBA generated encryption key) and NAF-UE Integrity (Link Layer can use GBA generated MAC key). The Card does not authenticate the NAF, but the NAF is able to check if the UE is a valid one. GBA is a low consuming protocol that runs at the SIM card. Sensor embedding MIM card can easily rely on SIM based security. GBA could replace EAP to perform the authentication tasks at the Smart Server-Smart Mobile level and even at the Smart Object level if the sensors are compatible.

2.1.3. Enhancement of the Security Protocols at the Smart Object level

The security protocols cited above should assure end-to-end security in the network. If they offer a robust and mature security scheme in the upper layers (IPSec or EAP) where the resources are sufficient, the security remains weak in the lower layers, and notably at the Smart Objects level. The Smart Objects generally are headless, offer scarce resources and communicate in the Wireless Sensor Network thanks to low power standards (802.15.4, ZigBee, 6lowPan).

The security protocols used to secure the communication with the Smart Objects widely imply random numbers. A typical example of such a secure protocol using symmetric cryptography between a Smart Object and a Smart Gateway/Server is presented below. Other protocols are detailed in [1].

Authentication of a new Smart Object in the network

1. $\mathbf{U}\rightarrow\mathbf{S}$: "get_nonce", ID_u
2. $\mathbf{S}\rightarrow\mathbf{U}$: "send_nonce", ID_u, r_A
3. $\mathbf{U}\rightarrow\mathbf{S}$: "get_key", ID_u, M_1 = E(m_1=("get_key", ID_u, r_A), $\mathbf{K_u(0)}$), Mac_1 = MAC(m_1, $\mathbf{K_u(0)}$), r_B
4. $\mathbf{S}\rightarrow\mathbf{U}$: "send_key", ID_u, M_2 = E(m_2 =("send_key", ID_u, $K_u(1)$, r_B), $\mathbf{K_u(0)}$), Mac_2 = MAC(m_2, $\mathbf{K_u(0)}$), r_C
5. $\mathbf{U}\rightarrow\mathbf{S}$: "ack_key", ID_u, M_3 = E(m_3=("ack_key", ID_u, r_C), $\mathbf{K_u(1)}$), Mac_3 = MAC(m_3, $\mathbf{K_u(1)}$)

r_A, r_B, r_C stand for "nonce", ID_u is the identity of the Smart Object u, $K_u(0)$ is the initial key of the Smart Object u and $K_u(1)$ is the new session key affected to the Smart Object u to communicate. E stands for "Encryption" and MAC for "Message Authentication Code".

In this basic protocol, random numbers are used to create "nonce" in order to avoid replay attacks, to generate the session key, and as AES algorithm is often used to achieve symmetric encryption, random numbers are used to complete the last encryption block to the required size.

As the Smart Objects are headless, the random numbers are actually generated with pseudo-random generators using a seed. Pseudo-Random Number Generators (PRNG) present two main drawbacks: first, they generate a sequence of random numbers that is cyclic, and second, they always provide the same random number sequence when they are reset. To guarantee a good security of the system, it is better to use Non-Deterministic Random Number Generators (NDRNG) based on entropy sources. But, on headless devices, entropy sources are scarce.

Selected entropy sources:

Two types of embedded entropy sources have been studying:

- The physical sensors,
- The wireless channel statistics.

A lot of Smart Objects are very small and some might even not include sensors. This is the case of the wireless router for example. For these devices, the challenge will be to use the wireless statistics as entropy sources. Three wireless statistics have been studied: Link Quality Indicator (LQI), Received Signal Strength Indication (RSSI) and packet error (erroneous CRC). Data collected using the low-cost headless devices eZ430-RF2500 sensor nodes from different setups are analyzed to characterize the relevance of these sources to generate entropy.

Selected methodology:

To compare entropy sources, we need to have some metrics for security and efficiency. The Shannon entropy and the min-entropy are the main metrics for security. Latency and energy consumption are the performance metrics.

Definition of the Entropy:

Let S be a source of entropy which produces values over X. We associate to these outputs a random variable X and x denotes a value sampled from X. The Shannon entropy and min-entropy for the source are given by:

$$H(X) = -\sum_x \mathbf{Pr}[X = x] \log_2 \mathbf{Pr}[X = x], \qquad (1)$$

$$H_\infty(X) = -\log_2(\max_x \mathbf{Pr}[X = x]). \qquad (2)$$

The Shannon entropy (Equation 1) was the first obvious choice to consider when studying a source. However, the Shannon entropy fails to measure precisely the capability of an adversary. In [2], the authors show that min-entropy (Equation 2) is a better metric. It is also the metric recommended by the NIST [3]**.**

Remark: Determining the min-entropy of a source requires the prior knowledge of the corresponding probability distribution. The term min-entropy of a source S is used in an ambiguous way in many papers. Let us define the random variable Y associated to an observation of S. This observation of S is defined as a set of n values sampled from S and belonging to the set Y $\subseteq$ X. At a first sight, computing H_∞(Y) cannot be considered as a good estimator of H_∞(X). There are two solutions to solve this deadlock. First, we are able to make a complete or unbiased observation. It means that Y = X and n is large enough such that the probability distribution of X and Y are much closed. Both conditions seem hard to achieve. Second, we use an estimator of H_∞(X) based on the observation Y. We denote this value $Est_{H\infty}$ (Y) or Est(Y) for short. We have chosen the second solution. The NIST has proposed several tools to evaluate Est(Y).

The NIST has recently released a first draft of recommendations in order to evaluate both the entropy source model and the meaning of entropy [3]. In [3], the NIST gives several tests applied to data in order to evaluate the amount of entropy provided by the sources. These tests are not to be mistaken with the NIST statistical test suite [4] which is made to test if a bit-string is conformed or not to some characteristics of an independent and identically distributed (i.i.d.) sequence. Applying the NIST statistical tests suite to the data produced by our sensors is not particularly significant because it is quite obvious that obtained sequences are biased.

The methodology [3] described to estimate min-entropy is different if the probability distribution of S is known or not. If the source S is i.i.d., some specific tests are applied. Otherwise, five tests are used to compute different statistic on the digitized samples and to provide information about the structure of the data. Their application to non-i.i.d. data will produce an underestimation of the contained min-entropy. No information is assumed on the probability distribution associated to the output of S. Each test reveals information about the unknown distribution given a statistical measurement. The entropy is estimated by minimizing over this set of distribution. For the following tests, a confidence level of 95% is considered.

The tests are based on the property that the expectation of a random value in the probability domain is analogous to the mean of a statistical set in the statistical domain. So, in order to find the probability distribution that characterizes the data structure it is assumed that the probability is equal to the most pessimistic estimate of the mean of the observed series that is the low-bound of the confidence interval. There are five tests: the frequency test (Est1(Y)), the collision test (Est2(Y)), the partial collection (Est3(Y)), the compression test (Est4(Y)) and the Markov test.

The five previous tests operate on relatively small values. This limitation is clearly an issue for the analyst who needs to study sequences of large values. Due to this problem, we were unsuccessful with the Markov test because our data were too large to have a reasonable computation time. The results provided by each of these tools represent a lower bound of the entropy that is present in a data source. We will consider the min-entropy as the minimum of values computed by all entropy estimators:

$$Est(Y) = \min(H(Y) ; H1(Y) ; Est1(Y) ; Est2(Y) ; Est3(Y) ; Est4(Y))$$

Note that for all our computations, the lowest entropy value was returned by the partial collection test, and the results provided in this section are given by this test.

Selected designs:

The embedded design will be based on the NIST recommendations detailed in [5] and [6]. Several cryptographic designs should be proposed according to the entropy sources available and to the device resources. The performance of some lightweight block ciphers, as thus proposed by [7], will be embedded and evaluated.

2.2. Joint Security/Privacy and Contextual Localization

Physical layer characteristics such as wireless channel profiles, physical locations or relative distances between objects have enabled the emergence of the new paradigm of Physical Layer Security, which deals mainly with two aspects. The first one concerns the introduction of the notion of information-theoretic security or keyless security which consists in achieving secure communication by using channel coding techniques that exploit the randomness of the wireless channel. Secret sharing using a common source of randomness is the second main facet of Physical Layer Security and the basis of generating secret keys from location-dependent radio signatures, which is one of the objectives of the proposed technology. One possibility is also to exploit in-depth the intrinsic relationship between such location-dependent physical signatures and the relative or absolute location information that can be made locally available in Smart Objects e.g., through cooperative algorithms.

The generation of secret keys, as described above, is particularly interesting for WSN islands formed by Smart Objects because it avoids the infrastructural complexity of public or private key distribution systems by replacing it with a decentralized key extraction algorithm and a public discussion protocol for key agreement. From the application point of view, the transmission of sensitive and/or personal information between Smart Objects or between Smart Objects and Gateways requires such secure communication links. Those links can therefore be protected using pair-wise private keys generated from physical layer metrics which should be sufficiently symmetrical in order to limit the public exchanges and also spatially uncorrelated so as to guarantee the privacy of the key between the two legitimate parties. Thus this enabling technology could concern primarily the SmartCity POC, e.g. to secure remote payment transactions and personal authentication depending on the occupied slot in underground car parking areas. But it could also concern any indoor service requiring payment, authentication or personal data exchanges, most probably in SmartHome/Office (e.g. Localization of authorized persons) or SmartShopping domains.

The intended complete secret key generation routine can be modelled with the following main blocks: metrics estimation over single links at the physical layer level (i.e. signal sampling, quantization and processing), optional information exchanges for the support of jointly cooperative key generation and ranging/localization at upper layers (i.e. MAC, Network and so up to the localization application) and finally, a local bit extraction algorithm followed by key agreement discussions. Our current work focuses on the first and the third blocks with the aim of proposing and evaluating different single link key generation schemes using Ultra Wideband (UWB) channel profiles enriched e.g. by the Round Trip – Time Of Flight (RT-TOF) or possible positional information. The validation tests concern mainly the achievable key lengths, the randomness properties of the keys and the bit agreement ratio between legitimate parties or between legitimate parties and a passive attacker generating its own key after signal acquisition and eavesdropping on public exchanges for key agreement. The performances of the proposed schemes will be tested in both perfect and degraded conditions of SNR and synchronization.

Regarding more specifically the early channel response estimation phase, we aim at comparing the direct sampling method using a wide range of sampling rates with reconstructing techniques for sparse signals such as Finite Rate of Innovation (FRI). This signal processing method will be used to estimate the amplitudes and times of arrival of multipath components from fewer samples, leveraging the prohibitive large sampling rate needed conventionally in UWB systems. It is also expected to improve the bit agreement ratio between legitimate nodes when they are not perfectly synchronized, which is usually the case, adding robustness and limiting the public exchanges. Nonetheless, the key lengths might be lower and a certain level of complexity will be added to the digital processing block. Then, the bit extraction or bit quantization mechanism will be implemented using two alternative algorithms: sample-based bit quantization already proposed in the recent State-of-the-Art as well as a time-based bit quantization method more appropriate for FRI channel estimation. Both of them could be enforced by the use of the RT-ToF information (in samples or continuous time), known only by the two legitimate nodes, in order to add further randomness to the key itself using for instance circular permutations or to protect the public exchanged information.

Parallel studies inspired by information theory for measuring the mutual information available between the sources of common randomness (the channel impulse response in UWB) have also been initiated. The main interest of this approach is to quantify upper bounds of the maximum key generation rate which is in fact the mutual information between the sources and to eventually use it as a metric for performance evaluation and benchmark of our different key generation schemes.

The concept of using inherent randomness of location-dependent physical metrics for key generation is an alternative to cumbersome key distribution solutions. Moreover, we can somehow see this type of scheme as a low-level context-aware mechanism because it uses the actual environment information to create the secret keys which, in time, will change adaptively with the environment.

2.3. Contextual geo-localization

Contextual geo-localization is an important technological brick of the BUTLER system. In fact, localization is mainly employed to enable context-aware applications. From the deliverable D1.1 [8], the mainly required features are accuracy, real-time operation, low-power consumption, reliability and scalability over heterogeneous networks. Moreover, different types of localization services are seamlessly required, from a combined geo and temporal information (smart transport) to accurate indoor positioning (smart shopping) and even to binary classification (outside or inside the smart home, inside a specific room, etc.). In order to accomplish such different, simultaneous and ubiquitous tasks, and at the same time to ensure the above mentioned features, the localization research effort was mainly focused into three directions:

- Measurement techniques
- Localization and tracking algorithms
- Hardware and devices

In particular, this section first presents a set of localization algorithms that were designed along the first year of the BUTLER project [9] and which are here further analysed, and finally proposed as the candidate ones to be implemented in the BUTLER system. These algorithms are then compared in section 2.3.1.7 in terms of types of input measurements, complexity and addressed scenarios. Finally, alocalization architecture is proposed in section 2.3.1.6 in order to meet the above mentioned horizontal requirements.

2.3.1. Proposed Algorithms

2.3.1.1. Super Multidimensional Scaling (SMDS)

For the simultaneous localization of a large number of targets in a Line-of-Sight (LoS) and Non-Line-of-Sight (NLoS) with no a priori measurement models, a fully algebraic solution is proposed whose computational complexity is almost represented by a set of matrix multiplication. A new formulation of the multidimensional scaling (MDS) technique referred to as *Super* MDS allows distance and angle information to be processed algebraically (without iteration).

The SMDS is a deployable localization method which relies on ranging and angle measurements, acquired by the system itself after deployment. It maps ranging and angle information onto the estimated coordinates of the sources to be localized. The angle information can be obtained by estimating the angle of arrival (AoA) signals at different receivers. Also, the ranging can be performed from time (difference) of arrival (ToA and TDoA) or received signal strength estimates.

This is a MDS-based localization problem in which the network is modelled as a complete oriented graph. The entities are the vectors pointing from a source to another, known as edges of the graph, and the measure of dissimilarity is their inner product.

The new algorithm, referred to as *Super* MD,S is shown to out-perform the classic MDS algorithm in the presence of realistic ranging estimation errors. It offers the following advantages:

- It permits a unified processing of angle and ranging information.
- It leads to a simplified structure of the positive semi-definite Gram Kernel Matrix at the heart of the MDS algorithm called Edge Gram Kernel Matrix.
- The localization under an absolute coordinate system is achievable with the knowledge of the coordinates of a single node.

All this is achieved at a lower complexity than the classic MDS, where the *Super* MDS eliminates the need for computing a "double-centred" kernel matrix and thereby implementing a reduced Gram kernel matrix. As a result, error propagation is largely avoided, since the Edge Gram Kernel is computed (element-by-element) from distance and angle estimates.

The *Super* MDS algorithm is suitable for static positioning and tracking as well by combining it with filters such as Particle Filter (PF), Interacting Multiple Model (IMM), Extended Kalman Filter (EKF), etc. As a result of its static positioning and tracking capability, the *Super* MDS can be used in all scenarios which require the localization of static or moving nodes such as Smart home/office, city/transport, health and shopping.

2.3.1.2. Interval Scaling (I-SCAL)

The Interval-Scaling (I-SCAL) framework addresses the positioning problem by minimizing a proposed weighted least-square (WLS) cost function. Specifically, a lower and upper bound of range measurements between nodes in the network, e.g. sets of two distances per link computed from received signal strength indicator (RSSI) or time-of-arrival (TOA) values, are used as input to the algorithm.

The I-SCAL algorithm presented in [9] is designed to solve static positioning problem in a centralized manner. However, differently from standard WLS solutions to the localization problem, which only provide the estimated targets' locations, the Circular Interval-Scaling (CIS) proposed solution [10] [11] is an area, namely a circle, centred on the estimated target location, and with area proportional to the range errors. This was shown in [10], where the output of the CIS algorithm was compared against the CRLB error ellipse [11], which is known to capture both measurement and geometrical error components. This capability of the CIS algorithm to capture the uncertainties on the targets' locations is illustrated in Figure 6 for a four target scenario, in which the corresponding CRLB error ellipses from the CIS algorithm are compared versus the theoretical ones.

In [12] it was shown that optimization of the CIS cost function can be computed through its upper bound and that, this can be efficiently solved using vector extrapolation technique [13], resulting in a low computational complexity solution to the problem.

In light of the above, the CIS algorithm represents an efficient algorithm to solve the positioning problem in scenarios in which both the position estimate and the corresponding confidence are required.

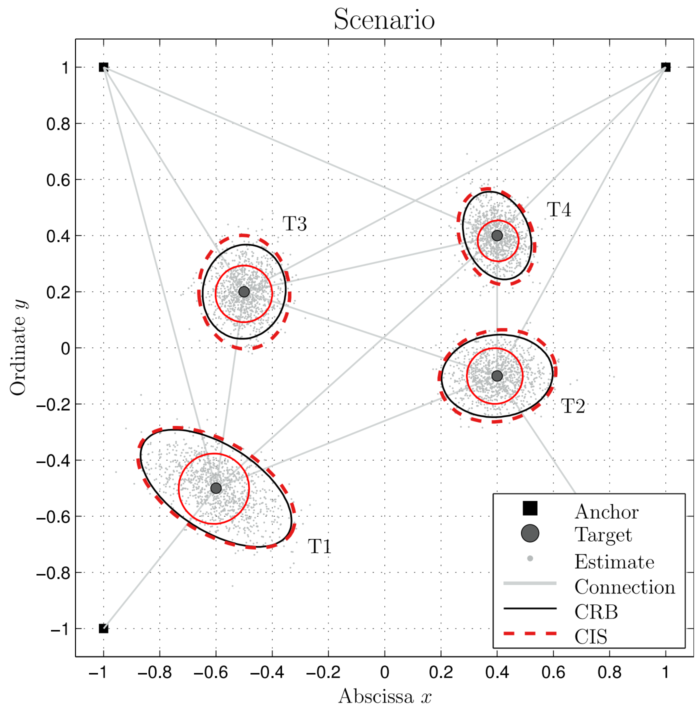


Figure 6: Graphical performance of the Interval Scaling Algorithm.

2.3.1.3. Energy Efficient Tracking (EET)

The 'Energy Efficient Tracking' (EET) algorithm [12], proposed and presented in detail in [9], is based on the Extended Kalman Filter (EKF) approach. The main feature of this algorithm is to localize mobile nodes with location accuracy that meets the requirement, P_a, imposed by a generic location-based application, while the energy consumption for ranging and communication is minimized. In particular, given as input the set of range measurements performed by an unknown mobile node, with respect to its neighbouring anchors, the proposed EET algorithm selects, on the basis of the Cramér-Rao lower bound (CRLB) [11], the closest set of anchors such that the resulting positioning error is lower than the requirement P_a. Thus, given the selected anchors, the EET algorithm performs the estimation of the mobile node position and then, in order to minimize the energy, it dynamically adapts the transmitted power, P_t, to be used in the next positioning estimation step. The less connectivity radius is, the less transmitted power is.

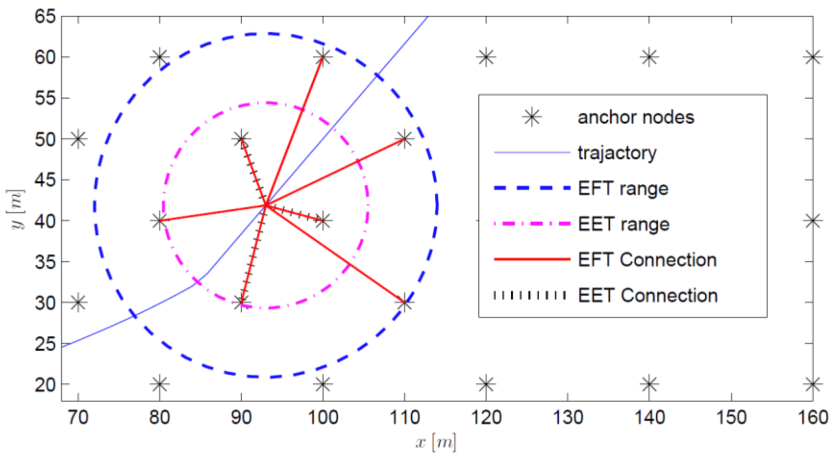


Figure 7: Example of the connectivity radius adaptation of the EET algorithm.

Figure 7 [12] shows in a certain point of the trajectory the connectivity information of the EET algorithm and the one of a classical tracking algorithm named EFT that uses a constant transmission power. As it can be observed, the EET is connected with only 3 anchors while the EFT with 7, in fact, the EET periodically adapts its transmission power with the aim to meet the positioning requirement and to optimize the energy consumption.

The EET approach presented in [9] uses range measurements based on time-of-arrival (ToA). However, the EET can be extended to work also with RSSI measurements performed by constrained smart objects. The EET algorithm has been selected mainly for the localization of smart objects in indoor mobility scenarios, thus it would be suitable for smart home/office related applications. In addition, since EET is based on EKF, it is possible to use also RSSI measurements performed by a Wi-Fi interface of the Smart Mobile platform.

2.3.1.4. Cooperative Links Selection and Optimized Medium Access Mechanisms for Enhanced Tracking (CLS)

Mobile tracking solutions, e.g. based on decentralized EKF filtering, are coupled with context-aware cooperative links selection mechanisms (e.g. based on the analysis of theoretical location quality indicators) and deliberate sensor-level mechanisms (i.e. either in Tx/Rx modes with respect to neighbouring mobiles). The idea is to experiment optimal localization/tracking precision, given connectivity conditions with respect to both available mobiles and fixed elements of infrastructure, with the minimum system cost in terms of energy consumption and computational complexity at mobiles, location-specific traffic and global network load.

The performance indicators used for links selections, which are based on beacon information broadcasted by 1-hop cooperative neighbouring mobiles, can be adapted to cope with homogeneous or heterogeneous wireless contexts. They can also take into account and compensate for the mobility and natural asynchronism of mobile neighbours. This selection step is used not only to identify and predict in a reasonably near future the minimum set of neighbours and measurements to be involved in the localization/tracking process (i.e. while limiting as much as possible the neighbourhood refreshment rate), but also to tailor in a decentralized way (i.e. at each mobile) the length and periodicity of Rx observation windows, the duration of active and sleep modes, or the data content and the periodicity of the beacons broadcasted to neighbours. These mechanisms can be applied indifferently to static and dynamic contexts, but they make sense mostly under mobility, when connectivity conditions, links quality and neighbourhood density are subject to fast changes. Different radio access technologies can be concerned, including short-range WSN low data rate technologies for peer-to-peer cooperative transactions (e.g. Impulse Radio - Ultra Wideband or Zigbee) and longer-range radio technologies (e.g. WiFi) with respect to fixed Access Points. Depending on these available radio technologies, RT-ToA and RSSI can be considered as single-link metrics in most cases.

The links selection and tracking blocks are implemented at the application level in Smart Objects or Smart Mobiles. They are fed by basic peer-to-peer ranging information relying on lower layers (i.e. physical and medium access), but then they provide feedback to those lower layers for optimized communications and energy saving, as described before.

From a mobile end-user perspective, this enabling technology can be applied for the tracking of SmartMobiles in the SmartHome/Office POC (indifferently for the localization of authorized persons, In-office wireless media access or Cocoon use cases), where mobile users should be tracked seamlessly and precisely from rooms to rooms in typical indoor environments. It shall also be useful to the SmartShopping POC for enhanced indoor navigation purposes (distance and path to the store location). But more generally, these links selection and energy/traffic efficient mechanisms can also concern any wireless network of mobile SmartObjects that would require precise geo-referencing of their sensing data with the minimum energy cost.

2.3.1.5. Distance Contraction (DC)

The distance contraction (DC) algorithm copes with distance-based positioning scenarios of a single target under non-line-of-sight (NLOS) channel conditions. The proposed approach is based on a non-Bayesian framework, namely a least-squares (LS) optimization, and offers a robust solution to circumvent the well-known non-convexity problem of the LS objective function and mitigate errors due NLOS channel conditions. The DC principle, which is derived analytically in [14], consists of correcting the distance measurements with “contraction terms” that yield negative (contracted) ranging values; thus transforming the non-convex LS function into a convex one; and shifting the global minimum towards the true target location. The proposed framework was applied to two classic algorithms, namely the Weighted Centroid (WC) and the Non-Linear-Least Squares (NLS), leading to polynomial-order complexity and accurate solutions. The performance of these algorithms is compared to state-of-the-art techniques both via simulations and experiments. The latter shows that DC-based positioning algorithms can provide an accuracy of a few tens of centimeters in a typical office/laboratory environment characterized by mixed LOS and NLOS conditions. The experimental results are obtained with commercial impulse-radio ultra wideband (IR-UWB) devices.

2.3.1.6. Semantic Localization

Humans do not necessarily rely on quantitative information or mathematical models to perceive space to travel from one location to another. They recognize relevant objects that serve as points-of-interest or landmarks that define the location or space, and remember that spatial knowledge for later use.

Positioning systems implement precise 2D or 3D relative or absolute localization techniques relying on accurate metric data, but this information is usually meaningless to humans. To bridge this gap, we need to map geo-localization schemes to semantically richer descriptions that make sense to human beings. We implement this approach in 4 steps:

- Definition of metrics for a predefined coordinate reference system (a.o. to define distance)
- Definition of reference points of interest or landmarks as nodes in a metric map
- Definition of semantic annotations to landmarks (e.g. door, rooms, floors, buildings)
- Definition of semantic spatial qualitative relationships between landmarks

From a semantic point of view, we can define spatial relationships with hierarchies and subsumption relationships by redefining subclasses as subregions. The latter results in a semantic map based on a graph representation of objects. For example, if the system knows a person of interest is in his office, it can also infer on which particular floor and in which particular building he is. Basically, the office defines a region which is part of other regions. As such, containment of regions or other spatial features can be implemented in ontologies (e.g. using the Web Ontology Language (OWL)) using single inheritance hierarchies). Unfortunately, the expressive power, and hence the spatial reasoning capabilities, are fairly limited with such an approach. Allen's interval algebra (1D) and the Region Connected Calculus 8 (2D) are more powerful qualitative spatial relationship models, the latter defining 8 spatial relationships:

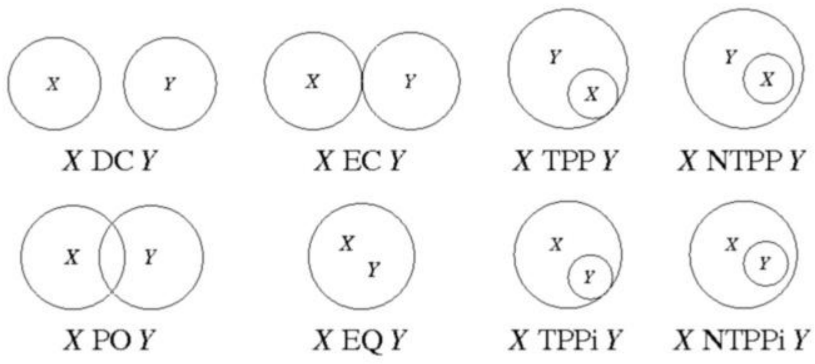


Figure 8: Region Connected Calculus 8 [Source: http://en.wikipedia.org/wiki/File:RCC8.jpg]

- **DC:** disconnected
- **EC:** externally connected
- **EQ:** equal
- **PO:** partially overlapping
- **TPP:** tangential proper part
- **TPPi:** tangential proper part inverse
- **NTPP:** non-tangential proper part
- **NTPPi:** non-tangential proper part inverse

Unfortunately, previous works [13], [14], [15] have shown there is no straightforward way to map these qualitative spatial relationships onto OWL-DL, nor in version 2.0 of OWL due to the lack of expressive power and tractability issues.

The practical mapping we propose tries to find a balance between the tractable semantic relationships we wish to support (being DC, EC, PO, EQ and PP, ignoring the tangential or non-tangential aspects), and implement this mapping on top of state-of-the-art ontology reasoners (such as OWL Pellet or HermiT).

Ontology reasoners are known to be resource demanding, making them unfit for deployment on sensors. Deployment on tablets or mobile devices is feasible but rather impractical due to the possibly large knowledge base the mobile has to manage. Due to the fairly high resource constraints of these enabling technologies, we will integrate them as a component in the Smart Server platform. An additional bonus is that it offers better scalability through caching and reuse of inferred spatial relationships for other clients or consumers. Such an approach is infeasible if every Smart Mobiles are supposed to infer these relationships individually.

This semantic localization scheme does not aim to position or track entities on its own, but builds upon other localization schemes, and aims to extend it with semantic topological and directional information. As such, Smart Objects and Smart Mobiles play a role in providing the lower level location information upon which the semantic localization is built. Furthermore, semantic localization would offer qualitative reasoning capabilities which the aforementioned algorithms do not focus on.

From a conceptual point of view there is no limitation regarding to which vertical domain it would be most suited for. However, it does assume prior knowledge (i.e. the existence of a semantic map and the boundaries of regions of interest). This kind of knowledge is less likely to exist for outdoors environments, but could be gradually built through crowdsourcing techniques.

2.3.1.7. Algorithms comparison

This section compares the above presented selected localization algorithms in terms of type of measurements, dynamicity of the unknown nodes, complexity and addressed scenarios. Results are listed in the table below.

Algorithm	Type of measurement	Dynamicity of unknown nodes	Complexity	Scenarios Addressed
SMDS	Range (e.g. ToA/RSSI) and angle (e.g. ToA and AoA)	Localization algorithm	Low computational complexity (O(N^3))	Smart Home/Office,
I-SCAL	Range (e.g. ToA/RSSI)	Localization algorithm	Low computational complexity (if implemented with vector extrapolation block)	Smart Home/Office
EET	ToA/RSSI	Tracking algorithm (pedestrian speed)	Comparable to the complexity of EKF	Smart Home/Office/Shopping (indoor) EET can be extended to include also GPS based measurements in order to be applied to Smart City scenarios.
CLS	ToA/RSSI	Tracking algorithm	Comparable to the complexity of	Smart Home/Office/Shopping

		(pedestrian speed)	EKF	
DC	ToA/RSSI	Source localization algorithm	Low computational complexity, O(N^3) with N as the number of anchor nodes	Smart Home/Office

2.3.2. Localization Architecture

This section proposes a preliminary localization architecture that has been designed in order to guarantee a localization service that seamlessly operates across different vertical scenarios (smart home/office, smart city, smart shopping, etc.). Since smart objects are very constrained devices, it would be better to execute localization algorithms in a module with higher computation capacity. Also the GW node may not be suitable to this purpose, thus the final choice is to implement and execute the localization algorithms in a centralized manner in the smart server platform connected with the smart GW through a LAN as showed in Figure 9. As it can be observed, the smart mobile is also taken into account. When the mobile module is indoor (no GPS coverage), it can provide to the location engine ranging measurements based on Wi-Fi. When the mobile node is outside, it can rely on its GPS module and send to the smart server the GPS-based estimated position. It can be supposed that the mobile node has also an IEEE 802.15.4 interface, thus in this case the mobile platform can behave like a smart object in which it sends the range measurements to the smart server where the localization process is executed.

Alternatively, if the computational capacity offered by the smart objects is enough, the localization algorithms can be executed directly on the smart objects and the final estimated positions can be sent periodically to the smart server platform. This type of distributed implementation improves the energy efficiency of the EET and E&TET algorithms as well as the cooperation among mobile nodes.

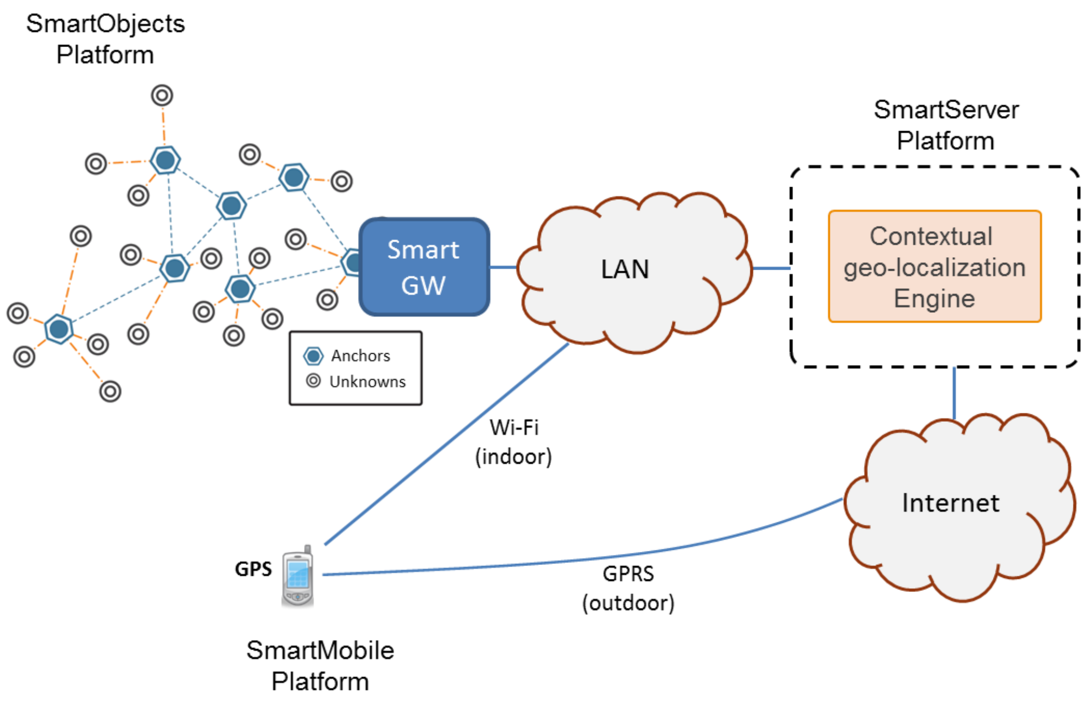


Figure 9. Contextual geo-localization physical architecture.

Figure 10 shows both the localization data flow and how the different localization components are mapped into the three platforms: smart objects, smart mobile and smart server. In particular, smart objects perform range measurements and send them to the geo-location engine through the smart GW. The smart mobile platform sends to the smart server RSSI measurements when it is indoor or GPS-based position estimates when it is outdoor. Finally, nodes' positions are estimated in a centralized manner by the geo-localization engine that runs on the smart server. There is not a specific algorithm that runs on the location engine but there will be a mechanism that according to the current scenario (indoor, outdoor, static node, degree of mobility, etc.) selects the most suitable localization algorithm to be executed among the ones that have been presented in section 2.3.1. Finally, as it can be observed, the location engine exposes several APIs for the other BUTLER components.

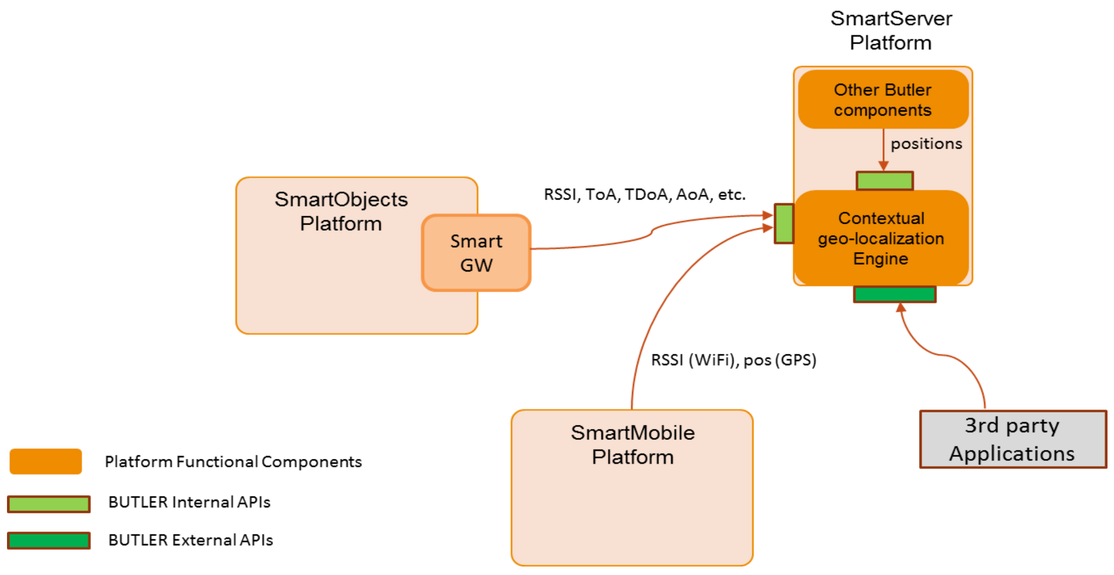


Figure 10: Contextual geo-localization data architecture

2.4. Geospatial databases

2.4.1. Definition and requirements

A spatial database is designed to optimize storage and query processes involving geographic information (points, lines, polygons, etc.)[2] by adding new geographical data types and spatial indexes. A spatial database, also known in this section as **geodetic database**, can be an extension of an already existing database or a dedicated one. There are spatial plug-ins for almost every popular RDBMS and there are also, as said before, some examples of dedicated spatial databases (for more information about related products please check the examples provided at the end of this section or check for more complete list and description of OGC compliant[3] products and the master database[4])

As said, the main standardization committee for the standardization that geo-enables the web is the Open Geospatial Consortium (OGC)[5]. Following the description provided by OGC: *"The Open Geospatial Consortium (OGC) is an international industry consortium of 481 companies, government agencies and universities participating in a consensus process to develop publicly available interface standards. OGC® Standards support interoperable solutions that "geo-enable" the Web, wireless and location-based services and mainstream IT. The standards empower technology developers to make complex spatial information and services accessible and useful with all kinds of applications."* [6]

Standards relevant for our project published by this organization are:

- OpenGIS Implementation Specification for Geographic information - Simple feature access - Part 1: Common architecture. [7]
- OpenGIS Implementation Specification for Geographic information - Simple feature access - Part 2: SQL option[8]

This section provides an overall view about why this type of databases is needed in the context of BUTLER. It also provides some basic concepts of the spatial databases and finally gives a short list of products that already are available in the market.

Geodetic databases are a key element in some of the functional components identified in the BUTLER architecture (especially in the Resource Directory when physical users are involved). The required geographical capabilities that have been identified as basic requirements for the horizontal platform are the following ones:

- Verify the inclusion of a point in a polygon.
- Verify the overlapping of polygonal areas.

[2] http://en.wikipedia.org/wiki/Spatial_database

[3] http://www.opengeospatial.org/compliance

[4] http://www.opengeospatial.org/resource/products/compliant

[5] http://www.opengeospatial.org/

[6] http://www.opengeospatial.org/ogc

[7] http://portal.opengeospatial.org/files/?artifact_id=25355

[8] http://portal.opengeospatial.org/files/?artifact_id=25354

2.4.2. Advantages of a spatial database

The use of a spatial database is a need for the BUTLER horizontal platform because of the requirements related to the location of devices and resources and its inclusion within virtual entities. There is also a basic requirement related to the need of querying the system to identify overlapping among different surfaces (entities) and inclusions of points (devices, resources) within them. The device exposition makes use of different spatial requirements and therefore the inclusion into the architecture of a spatial database is something straightforward.

Optimizing queries and insertions of geographic objects in a database is a mandatory requirement when using systems that have in their foundations high availability and quick response times for large set of geographical information data.

In addition to the most used data types defined in general-purpose relational databases (String, Numbers and dates); spatial databases have special data types that allow the representation of geographic features. The main issue tackled by spatial databases is to answer the question of which objects are within a particular bounding box[9]. Within relational databases, it is very common to use B-Trees[10] to sort and index the information, making them optimized to searches, sequential access, insertions, and deletions in logarithmic time. When the information is spatial, what the user wants to know in an efficient way is the pertinence of a geographical object to another one. This is performed by using spatial indexes such as R-Trees, Quadtrees, etc. In the case of PostGis, the solution chosen by this project together with Postgress, is the R-Tree spatial index. The figure below shows an example of an R-Tree.

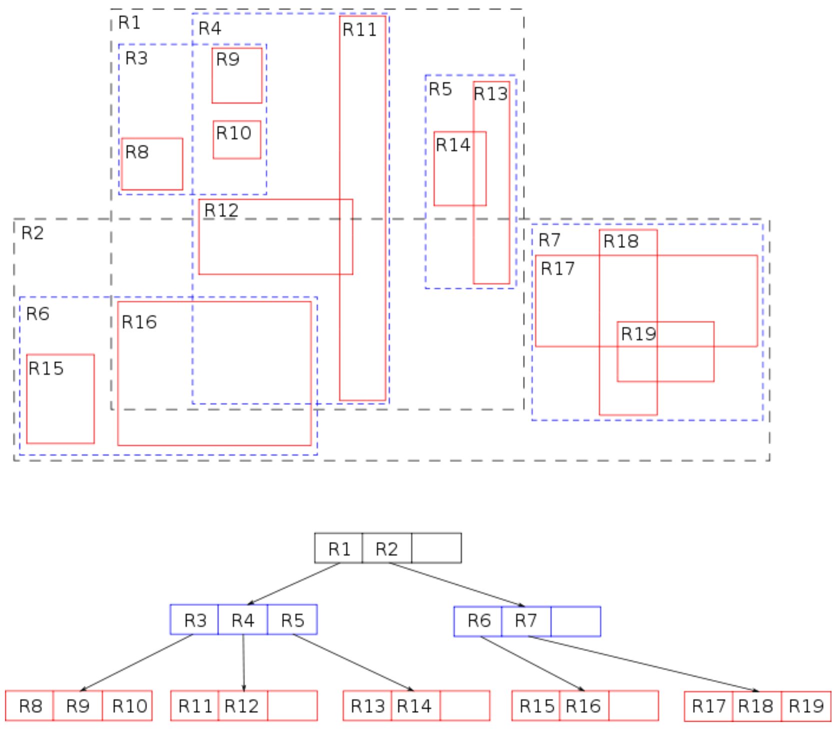


Figure 11: Example of R-Tree (used in PostGis)[11]

[9] http://workshops.opengeo.org/postgis-intro/introduction.html

[10] http://en.wikipedia.org/wiki/B-tree

[11] Source: Wikipedia. http://en.wikipedia.org/wiki/R-tree

Spatial Database integrated in the horizontal platform

As mentioned above, many GIS products rely on a (possibly 3PP) standard RDBMS to store geographical data. Oracle, SQL Server and Postgres+PostGIS all support geographical data out of the box. Other systems often build their products as plugins of these databases, or applications on top of them. A few exceptions, like Teradata and Altibase, provide their own Spatial DB products with a special focus (e.g. Altibase provides both in-memory and disk based data handling).

Most of these DB products, including mainly both Oracle and SQL Server, use a commercial license, whereas PostgreSQL uses a license[12] similar to the MIT and BSD licenses. Commercial support is listed in the PostgreSQL web page[13]. The product of choice for the horizontal platform has been PostgresSQL+PostGIS, open source products that have available commercial support.

"PostGIS follows the Open Geospatial Consortium's "Simple Features for SQL Specification" and has been certified as compliant with the "Types and Functions" profile. PostGIS is open source software, released under the GNU General Public License."[14]

Product Listing

Below are provided some examples of the products that are listed in the OGC database[15]:

- Altibase provides a database with geo-spatial support (http://www.altibase.com/)
- ESRI provides GIS software that can use relational databases as backends to store geometry data (http://www.esri.com/software/arcgis/geodatabase/storage-in-an-rdbms.html).
- Intergraph (http://www.intergraph.com/sgi/products/default.aspx) offers geospatial management products.
- Oracle (http://www.oracle.com/technetwork/database-options/spatialandgraph/overview/index.html?origref=http://www.oracle.com/technetwork/database/options/spatial/index.html) includes a spatial option for Oracle Database 11g Enterprise Edition. It seems to fully support SQL/MM as defined by the OGC.
- PostGIS (http://postgis.refractions.net/) is an add-on product that adds geographical constructs (types and functions) on top of the PostgreSQL relational database. PostGIS supports SQL MM extensions, like circular segments, but does not support some functions like the ST_Overlap function defined in OGC.
- SQL Server (http://msdn.microsoft.com/en-us/library/bb933876.aspx) supports OGC geometry types and functions.

MySQL (http://dev.mysql.com/doc/refman/5.6/en/spatial-extensions.html) supports basic geometry types, including Points and Polygons, but not CircularStrings.

[12] http://www.postgresql.org/about/licence/

[13] http://www.postgresql.org/support/professional_support/

[14] http://postgis.net/

[15] http://www.opengeospatial.org/resource/products/compliant

2.5. Context management based on complex event processing

Context-awareness is one of the distinctive features of BUTLER. However, precise definition of context-awareness is difficult to obtain, although there is a consensus on it being a complement of location-awareness. Context-awareness originated as a term from ubiquitous computing (or as so-called pervasive computing) which sought to deal with linking changes in the environment with computer systems, which are otherwise static.

On the other hand, BUTLER focuses on IoT context-awareness and therefore, context will be defined in BUTLER as associated to the entities handled within the BUTLER platforms. Therefore, so-called context entities (as defined in OMA NGSI[16]) will be the entities considered by BUTLER. That is, Smart Objects, virtual entities, Smart Mobiles and users. Taking **OMA NGSI Context Management** as conceptual framework (and specifically the **NGSI-10: Context Information Interface**), two dimensions of the context consumption can be considered (it is important to note that such a specification only considers the interfaces provided by the so-called Context Management component, but not the way the context is managed by the said component in order to implement the interfaces):

- A **synchronous** dimension where the clients query the Context Management component asking for context information. This synchronous dimension involves the computation and retrieval of the current value of the context the client is interested in. That is, when the client asks for the context associated to an entity, such context is extracted (or generated if more complex iterations are required to create a context attribute) and given back to the client.
- And an **asynchronous** dimension where clients subscribe to changes in the context they are affected by. As the context associated to a context entity may change as time passes by, clients will receive notifications when the change surpasses a usually client-defined threshold.

Such dimensions can be respectively mapped to the **Query** and **Subscription and Notification** operations in the NGSI-10 interface mentioned above.

Once the context attributes associated to a context entity have been defined, each dimension will be handled in different way. As said, we will follow a model in which the context attributes associated to a context entity are defined beforehand. In the BUTLER context, and following the IoT-A ARM approach, virtual entities are abstractions representing physical entities in the real world. However, users, devices (Smart Objects in the BUTLER parlance) and user terminals (Smart Mobiles) can be also modelled as entities and therefore become context entities and have context attributes associated to them. According to the broad definition mentioned above (context-awareness would be related to “changes in the environment”) any element of the environment of such an entity could be deemed as a context attribute (following again the definitions of OMA NGSI Context Management) but such an unbounded approach would prove useless for real life services. Instead, the following iteration will be suggested:

[16] Context Entity: “Any Principal and object, which has a state. This state can be described using Context Information”

- For any given domain-specific application, the (relevant) context of the relevant entities will be defined at the service creation time (the BUTLER platform will provide an interface for the discovery of existing virtual entities and data sources[17]). Service developers will access the BUTLER platform and will have the means to build the context of an entity from the existing data sources (that could be explicitly associated to the entity or not). For instance, some context attributes associated to a given device (a Smart Object in BUTLER parlance) could be related to values sensed by nearby devices[18]. However, on a higher level, the context associated to a virtual entity could include a composition of values sensed by devices logically related to the entity[19]. The BUTLER platform will allow developers to use simple rules to compose the desired elements of the context associated to an entity. This step defines the context attributes associated to a context entity. Note that this approach supersedes the NGSI-9: Context Entity Discovery Interface as the focus here is not a generic framework for the management of context but how to enable synchronous and asynchronous retrieval of context information (see below the interface for the discovery of entities and data sources).

[17] It is assumed that data sources in the context of BUTLER will be mainly associated to devices.

[18] The context associated to a temperature sensor could include the presence of persons in the surrounding area. Such information is provided by a presence sensor.

[19] Let's consider a Smart Home scenario. The context associated to an entity belonging to the "house" type could include the following attributes:

- The energy consumption, as sensed by a smart meter.
- The energy production, adding all the instant values of generation senses by sensors in the microgeneration grid.
- Temperature, an average of the temperature measured by all the indoor temperature sensors.
- And the number of occupied rooms, by taking all the presence sensors detecting a person in the room they monitor.

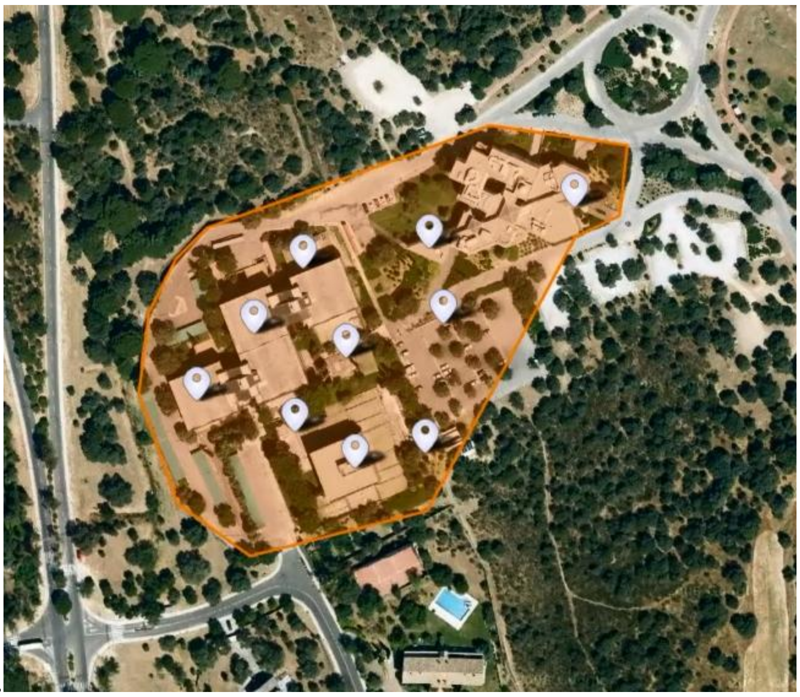


Figure 12: Example of the BUTLER interface for the discovery of entities and data sources based on a map tool (the entity is here depicted as a polygonal shape; data sources as map pins)

BUTLER will also provide a rule-based toolbox for Service developers to help them to build each context attribute definition from the available data sources (combination of sources, carry out simple math operations such as average, addition and so on). The BUTLER platform will store such a context definition to subsequently handle the synchronous and asynchronous dimensions of context management.

- When a client synchronously asks for the aforementioned context information, the definition of the context will be used to generate the current value of said context and return it to the client. Appropriate transformation of the context definition, expressed in terms of data sources and rules, will be carried out for subsequent processing of context queries. For instance, if the context attributes associated to an entity involves three different data sources, said data sources are queried to retrieve its values and the context is returned to the client with the appropriate transformations[20].
- The asynchronous management of the context will follow a different process:
    - First of all, the context definition is also decomposed in order to determine which data sources are involved.[21] Optionally, the BUTLER platform could send a notification to the involved sensors for them to know they must send periodic *heart-beats* with the data being sensed (if no client is interested in their data it would be useless to send such information).

[20] Data sources, even if associated to a device, can be handled in many ways:Devices supporting polling can be polled.

Devices sending only notification could require cache storage in the server. Therefore, said storage is queried to get appropriate values.

[21] In the example provided above, the smart meter, the microgeneration meter, the indoor temperature sensors and the indoor presence sensors.

- Next, the context definition will be transformed into a language or executable code (rules) that could be processed by a Complex Event Processing (CEP) engine. Such an engine will be configured to receive the information being sensed by devices and whenever a change in one of the sources of a given context is detected, a separate process will be triggered to compute the current value of the context and send it back to clients. The applications fed by the CEP engine will use time windows in order to determine when the status of a sensor, according to the value being sensed, changes.
- The developer should also set the thresholds above which a notification must be issued. Such thresholds will be also added to the CEP applications so that notifications are triggered not when a value changes but when it does it beyond a given threshold.

Appropriate procedures should be used in order to a) guarantee that events coming from sensors are adapted to the event formats handled by the CEP engine; and b) handle data sources that cannot send notifications but only support polling (for instance, by developing a polling system that reads data from sensors, or other data sources, and generates events to be sent to the CEP engine).

2.6. Behavior awareness of single and multiple users

Activity recognition is a key enabler in smart environments. In BUTLER, we broadly divide user activities into two categories – (1) tasks that are pre-defined and planned in space and time, and (2) actions that are spontaneous and atomic, and that can be sensed directly or indirectly.

2.6.1. Context-based Optimized Communications to Gateways in Wireless Networks of Smart Objects

The core concept within this proposed technology is to define priority metrics and mechanisms to establish adaptive and optimized wireless communications between mobile sensing nodes or end-devices (i.e. Smart Objects) and Gateways or WSN sinks, depending on the local and temporal context (i.e. in terms of nodes density, activity, mobility, energy autonomy, etc.).

Accordingly, the investigated metrics can include and combine different contextual parameters such as:

- The data traffic model and a priori priority (e.g. periodic monitoring or alarm);
- The local data buffer status;
- The local battery status;
- The number of hops to reach the sink (rank/degree, set as a route cost);
- Possible latency constraints from the application;
- Detected individual mobility profiles, according to an a priori and/or self-learnt classification;
- Possibly, detected groups with collective mobility profiles, and/or with common information data to broadcast and/or with similar signalling/control information to receive from the coordinator.

Regarding the two last points and more specifically the mobility estimator, the idea is to recognize particular patterns or behaviours (with specific impacts in terms of medium access or networking) among a finite set or family of known patterns, for instance based on coarse geometric criteria in first approximation. One can also assume gradually refined absolute/relative location information, or even simply 2D speed information, from radiolocation blocks.

The proposed blocks are rather generic but concern mainly the medium access and networking layers of the protocol stack implemented in turn at Smart Objects, somehow regardless of the physical radio layer (which can be anyway properly abstracted whenever it has to be integrated in the priority metrics, e.g. through over-the-air PER modelling), and at first sight in homogeneous networks. However, they may also be compatible with BUTLER interoperability requirements in heterogeneous wireless contexts, as long as a low-level compatibility is ensured between different low data rate radio access technologies through adequate adaptation sub-layer(s).

The performance of the proposed technology will be assessed through cross-layer simulations based on dedicated event-driven packet-oriented tools.

2.6.2. Pre-defined activities planned in space and time

Typical examples of category 1 activities are the work-day routine of a user, such as having breakfast and commuting to office. These activities are characterized by a high level intuitive notion of atomic actions, but over time difficult to infer directly from low-level sensors. Each of these activities either has a set of measurable goals such as 'reaching office' or a set of associated temporal events such as 'using stove' or 'using coffee machine'.

'Human input', such as list of associated objects or places and their relation to the activity under consideration, is the most important part of the knowledge for predicting them. Hence semantic abstractions are provided for representing any associated event (using an object, being in a location, etc.,) and their relationships with the tasks. A task itself is characterized by a set of attributes such as, pre-conditions, situations, events, transitions, and rule sets. We model these task related attributes using our Situation Studio tool.

Situation: The activity or task performed by the user that has to be inferred, e.g. ‘preparing breakfast’, ‘commuting to office’, etc.

Pre-condition: These are a set of constraints that have to be satisfied for the situation to be possibly true. It can relate to another situation or any other relevant context information. For instance the situation ‘preparing breakfast’ can have preconditions such as ‘the user is out of bed’ and ‘the user is at home’ (not in a hotel/guestroom where the breakfast will be provided).

Events: This typically refers to sensor outputs that are observable during a situation. Order of events can somehow influence or distinguish situations. It is not modelled yet in the system. Similarly, the time of occurrence will be considered. Typical examples are, ‘coffee machine is on’, ‘opening the front door’ etc. Please note that there is no conjunctive relation between these events and each of these events will have a prior probability of observation that can be either automatically learnt or user-specified.

Transitions: Transitions represent a finite state machine model of the possible transitions of situations that are possible from the current situation. For example, ‘preparing breakfast’ can be succeeded by the situations ‘commuting to office’, ‘work from home’, etc.

Also, each task is characterized by properties such as the current status (started, finished, interrupted, resumed, restarted) and whether it is optional or mandatory, interruptible, stand-alone or a routine task.

Once the common situations of a user along with their attribute details are collected, a probabilistic model is built to infer the situations when pre-defined events are observed.

Modeling step: $p(E_1,E_2,\ldots..E_n|S) \alpha\, p(S)\, p(E_1|S)p(E_2|S) \ldots..p(E_n|S)$

Inference step: $p(S|E_1,E_2,\ldots..E_n) \alpha\, p(E_1)\, p(E_2)\, p(E_n)\, p(S|E_1)\, p(S|E_2)\, p(S|E_n)$

Strong conditional independence assumption is incorporated in the Naïve implementation of the tool. A Bayesian implementation with more relaxed independence will be considered if any of the BUTLER scenarios need it.

Once the common routines of a user are modelled, case based reasoning is applied to predict the possible activity of a user at any given time or place (without any real-time sensing) using appropriate similarity measures. For instance, in the Situation Studio it is possible to represent time of the day and location of the user as events and associate it to the activity (or task) being modelled. As explained above, events such as object usage are used to determine the current activity (or task) of the user while corresponding time of the day and location are learned. Later, when a time of the day and a location are presented, the system lists (and ranks) possible activities.

An example of how to model a situation along with its associated attributes is shown below,

```
<?xml version=”1.0” encoding=”UTF-8” ?>
<behaviour-task-model>
  <situation id = “PreparingBreakfast”>
            <pre-conditions>
                      <pre-condition name = “UserOutofBed” />
                              <pre-condition name = “UseratHome” />
            </pre-conditions>
            <events>
                      <event name = “CoffeeMachineON” prior = ”0.9” />
                      <event name = “KettleON” prior = ”0.25” />
                      <event name = “OpeningRefrigerator” prior = ”0.9” />
            </events>
            <conditional-probabilities>
                      <conditional-probabilityvalue = ”0.8”>
                                <event name = “CoffeeMachineON” />
                              </conditional-probability>
                      <conditional-probabilityvalue = ”0.75”>
                                <event name = “KettleON” />
                              </conditional-probability>
                      <conditional-probabilityvalue =”0.4”>
                                <event name = “OpeningRefrigerator” />
                              </conditional-probability>
                      <conditional-probabilityvalue = ”0.9”>
                                <event name = “CoffeeMachineON” />
                                <event name = “KettleON” />
                              </conditional-probability>
             </conditional-probabilities>
             <transitions>
                       <transition name = “CommuntingtoWork” probability = “0.7” />
                       <transition name = “WorkfromHome” probability = “0.3” />
             </transitions>
  </situation>
</behaviour-task-model>
```

As you can see in the example above, we do not specify any order of events for CoffeeMachineON and KettleON to define the conditional probability of the situation given the occurrence of these two events.

2.6.3. Spontaneous activities

Category 2 consists of elementary activities that are observable (with low-level sensors) and automatically learned with machine learning techniques such as typing on a keyboard, entering a room, walking, falling down, etc. They are spontaneous and momentary compared to category 1 activities. In BUTLER, we infer these activities either directly using appropriate sensors or indirectly using other related context information. For example, physical activities can be measured using an accelerometer attached to the body of the user. Nevertheless, if a user is located to be at his desk then we can be sure that he is only sitting/standing.

In BUTLER, typical activities related information that interests us are – current activity of the user, physical activity of the user over certain time period (say yesterday or till this evening). All these information accessible for any activity aware application through appropriate API explained at the end of this section. The below pseudo code explains typical physical activity recognition to recognize activities such as standing, walking and running with accelerometer data integrated into BUTLER platforms.

<u>LEARNING PHASE</u>

Data Collection

1. Accelerometer data corresponding to 3-axes are recorded at a pre-determined refresh rate

Pre-processing

2. Remove high-frequency noises using a low-pass filter

Learning

3. Compute the amplitude of the signal

Peak-detection Algorithm

4. Learn the sensitivity parameters

Decision tree

5. Compute FFT coefficients
6. Learn the decision tree using FFT coefficients and amplitude

CLASSIFICATION PHASE

Data Collection

1. Accelerometer data corresponding to 3-axes are recorded at a pre-determined refresh rate

Pre-processing

2. Remove high-frequency noises using a low-pass filter

Inference

3. Compute the amplitude of the signal

Peak-detection Algorithm

4. Detect the peaks
5. Count the steps based on the rate of peaks detected

Decision-tree

6. Compute FFT coefficients
7. Classify with the learned decision tree

The major concern in activity recognition in IoT environments is the growing number of device with varying capabilities. In BUTLER, we tackle this with an introspection API which is used to discover a set of currently available sensors and their corresponding activity recognition capabilities. For example, when run in Android phone, it can discover accelerometer and indicate it is capable of direct sensing of physical activities such as standing/sitting, walking and running. Also, it can list other sensors such as GPS, Bluetooth connectivity, proximity sensors for indirect sensing of user activities.

Multi-context fusion is another major concern in IoT where the possibility of acquiring multiple contexts is substantial but the availability of all these contexts (at all time) is unforeseeable. Hence, a hybrid Bayesian fusion algorithm - Globally Connected Locally Autonomic Bayesian Networks (GCLABN) is proposed and implemented with the WEKA library. GCLABN models different high level contexts and their observable nodes as separate centred-Bayesian networks and learn the relation between these centre nodes (contexts) over time. It is modular, highly distributed and linearly scalable making it especially suitable for multi-platform horizontal architecture of BUTLER.

2.6.4. Group behavior

Contextual networking in BUTLER refers to the ability of the system to model the preferences, intentions and activities of a group on the whole as a single entity.

Prototyping crowd behavior with few dominant individuals:

One of the key challenges in modeling group behavior is to identify few dominant individuals who can represent the group and/or drive its desired/wanted patterns. Identification and analysis of these individuals is necessary to learn and predict the next step of the group.

First, a knowledge base of the user with profile data from each user with typical profile details such as gender, age, marital status, locality, occupation, etc., and preference/routine tasks of the users are captured in the system. Then, the similarity between the individuals in the group and the group preference is measured using a suitable similarity metric such as Euclidean, Manhattan or Pearson distance. Top ranked individuals whose preferences match with that of the group are selected and their weighted task models/work flows are used to predict the behavior of the group. As the composition of the group can vary dynamically, the ‘top ranked’ representatives of the groups are computed periodically.

2.6.5. Lack of activity

As much as we are interested in the mobility and current activity of the user, we are also interested in detecting 'lack of activity' of a user in BUTLER. Unlike activity recognition, 'lack of activity' recognition is difficult to define and be quantized as unable to be registered by a set of sensors does not necessarily mean being inactive. Hence, we gather all possible contexts available in the system and only if any of the typical tasks (or activities) done in that particular place or time is not recognized, we conclude that the user is being inactive.

2.6.6. Selected technologies and APIs

In order to design and implement the above algorithms and tools, we use the following enabling technologies.

Software technologies:

a) Esper, a Java based component for complex event processing to identify and match complex patterns in input data stream and to recognize them as events which in turn can be used as inputs by the situation studio tool.

   Esper is an open-source component for complex event processing (CEP) developed by EsperTech. Complex Event Processing is a technology that offers real-time management of big amounts of messages avoiding the need for storing all of them for their later analysis.

   Its main feature is to process events streams as input. That processing can consist in filtering those events looking for predefined patterns or analysing them by extracting inner data.

   Since it is written in Java and does not provide any kind of GUI but an API for Java, it is clearly oriented for developers. Events are represented by Java objects.

   Esper engine can be programmed using patterns or statements. In case of using patterns, when an event matches the pattern conditions the engine triggers an action. Statements provide different possibilities than patterns because they allow extracting data out of the events and combining them with data from other events streams. Statement programming is similar to querying a database, but with a real-time and continuous performance and response.

b) WEKA, a Java library for machine learning algorithms implementing various pre-processing, classification, clustering and visualization techniques. For instance, we have implemented physical activity recognition algorithm using decision tree as the resulting model is much more user understandable (compared to techniques such as Artificial Neural Networks and Support Vector Machines). Also, other techniques are implemented on case by case basis when they offer additional advantages. For example, Naive Bayes is used when continuous learning is required by the application (decision tree does not support incremental learning in WEKA).

c) Rapid miner, an open-source environment built as a wrapper around WEKA is used for offline analysis of data as it has powerful graphical user interface to simplify work-flow creations.

d) HermiT, a semantic reasoner for ontology languages such as OWL2. OWL2 ontology provides classes, properties, individuals and data values that can be stored as semantic web documents.

Hardware technologies:

a) Accelerometers

   To generate time series data corresponding to the motion of the user (or attached body parts). Difference in waveform or magnitude of acceleration and other features in time/frequency domain are good enough to distinguish physical activities such as standing/sitting, walking, running and fall detection.

b) Microphones

   To fine tune the predictions with sensed noise levels. If the user is indoor and the noise level is high, then there is a party (assuming the TV is switched off)

c) Ambient light sensor

   To detect the context of the use of smart phones – whether they are inside or outside (of a pocket or bag)

d) Fitbit

   A third party device with in-built accelerometer to count the number of steps and track it through a web-based application. As it uses proprietary algorithms and protocol, only the end results are accessible by us.

Core APIs:

- Introspection API
  - Methods

    [Sensor] getAvailableSensors()
    [Activity] getDetectableActivities()
    [Activity] getDetectableActivities([Sensor s])
    [Activity] getActivities(SemanticLocation l)

- Get current activity
  - Methods

    [Activity] getCurrentActivity()

    [Activity] getNextActivity()

    Float getActivityProbability(Activity a)

Application specific APIs:

- Get physical activity of the user
  - Methods

    Int getNumberofSteps()

    Int getCalorieExpenditure()

    Int getNumberofSteps(Day d)

    Int getNumberofSteps(Week w)

    Int getNumberofSteps(Month m)

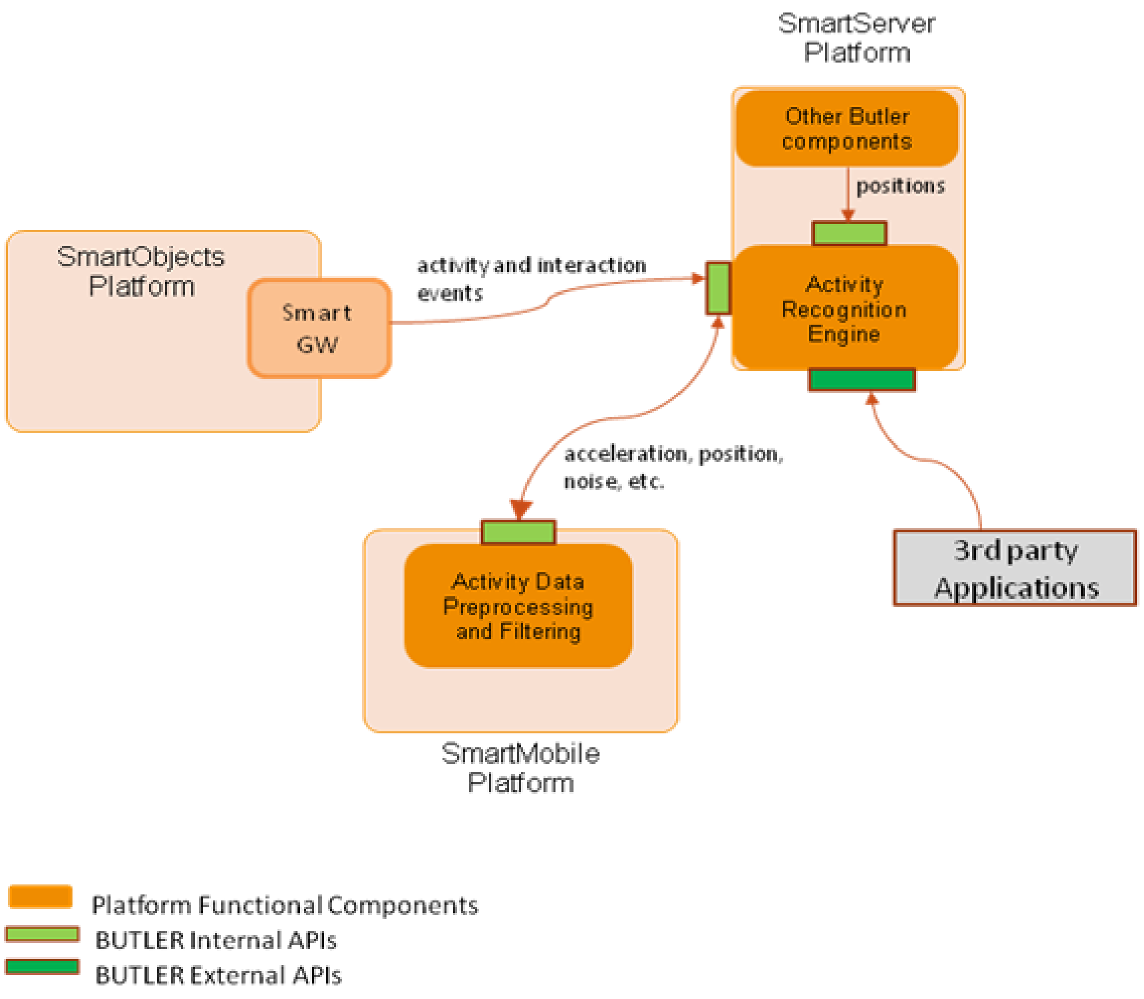

SmartServer Platform
Other Butler components
positions
Activity Recognition Engine
SmartObjects Platform
Smart GW
activity and interaction events
acceleration, position, noise, etc.
Activity Data Preprocessing and Filtering
SmartMobile Platform
3rd party Applications
Platform Functional Components
BUTLER Internal APIs
BUTLER External APIs

Figure 13: Behaviour awareness architecture

3. Smart Object Technology

3.1. Goals and needs

By the definition given in the first section, we already know that a Smart Object has two main behavioural properties: the interaction with the physical world and communication.

Smart objects interact with the physical world capturing information from it with their sensors and by affecting the physical world with their actuators.

Smart objects communicate. Communication is essential to the behaviour of smart objects. The real power of smart objects comes from their ability to communicate.

The technical challenges for smart objects include the node-level internals of each smart object, such as power consumption and physical size, as well as the network-level mechanisms and structures formed by the smart objects.

The node-level challenges of smart objects primarily have to do with power consumption, physical size, and cost. The severe power consumption constraints have design implications for the hardware, software, the network protocols, and even the network architecture.

The node-level challenges of smart objects deal with the small scale of available resources, whereas the network-level challenges deal with the large scale of the smart object networks. The large scale of network impacts on routing algorithms, on the huge amount of generated data and on network management.

Interoperability is another fundamental need for smart object. Interoperability is essential between smart object devices and the surrounding technological ecosystem.

Due to the smart object requirements, we believe that the IP technology is the best choice for smart objects, since it guarantees high level of interoperability, scalability, evolution, configuration, management and support to application diversity.

However, even if IP technology allows smart objects to be directly connected to the Internet, their restricted resource obliges to interconnect them by the mean of a gateway. Indeed, IETF protocols use too much energy, spectrum, and memory to be applicable to smart objects. Energy-constrained nodes have low duty cycle, little ROM space for code and little RAM space for data. A smart object gateway makes up all the necessary protocols adaptation/compression and mapping to overcome the smart objects limitation.

Moreover, the BUTLER SmartObject Gateway, provides the following additional functionalities:

- **Resources Management**: It handles information of connected devices and the resources they host. It is able, by doing lookup searches, to create/delete/update and publish on the resource directory (a gateway functional component) a resource descriptor related to a given Resource whenever it is available, no more available, changed or ready to be externally published.
- **SmartObject cache** (Discontinuous connectivity (sleeping devices) management): This component acts as a cache for gateway incoming resource requests whenever the wanted resource is in its sleeping period. The cache is also implemented in the opposite direction whenever the device wants to notify information about its resource to a sleeping consumer.

- **Data I/O** (Resources Accessing): It enables basic communication with resources served by the gateway (hosted inside or outside the gateway). Mainly it enables CRUD (Create, Read, Update, Delete) operation on resources.
- **SmartObject Security** (Authentication, Authorization, Data Integrity and Confidentiality): It controls that services and data provided by BUTLER are accessed/manipulated only by authorized users. Moreover this component enforces data integrity and confidentiality whenever the context application requires.
- **Devices subscription and notification**: This functional component allows a generic subscriber to manage subscription to a gateway served resource in order to be notified whenever some resource condition happen. Subscription by itself will be seen as resource.
- **Data Processing**: Smart Objects can, optionally, include Context Management functionalities to generate the context associated to them.
- **Business Logic**: The Smart Object business logic handling the native functionality of the Smart Object.

Due to the usage of a Smart Object Gateway the support of “legacy device”, meaning devices communicating using non-IP based protocols, can be feasible, whenever is needed.

In order to reach this functionality the Smart Object should provide (by the mean of Adaptors components) a generic device access API exposing a high-level, protocol agnostic API towards consumer gateway internal functional component. That internal API should provide the following basic functionalities:

- Network and service discovery
- Network start-up and joining operation
- Device parameter update (read and write attributes, configure and report events)

The figure below graphically illustrates the main functional components that are foreseen by the architectural BUTLER Working Group for Smart Object platform.

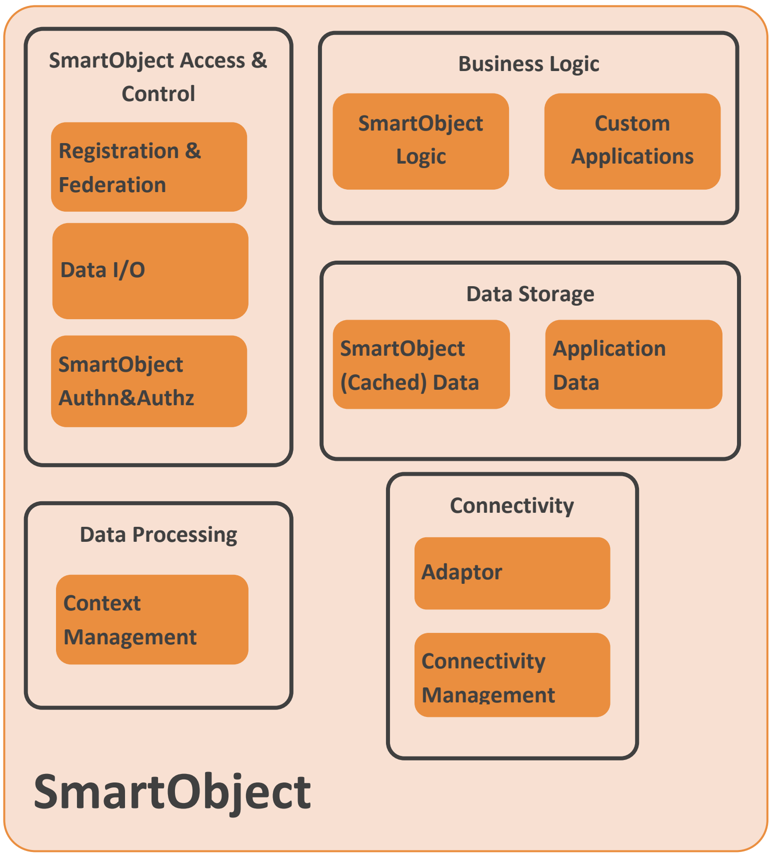


Figure 14 : Smart Object Architecture Overview

3.2. Selection process

The following section covers the Smart Object assets, available from BUTLER partners. From the hardware point of view, these Smart Objects are based, in most of the cases, on commercial evaluation boards, developed by System-on-Chip manufacturers, for demonstrating and developing purposes. Ad-hoc solutions developed by partners are also available. Referring to protocols stack and to communication technologies, the smart object can be divided into two main categories: the IP and the non-IP smart objects. Before starting this Smart Objects review, a set of relevant features used for selection processes is illustrated.

3.2.1. Criteria to classify the selected technologies

In this paragraph the most important Smart Object features are illustrated, from the hardware design, operating system, the physical communication standard, protocols standard, services discovery, to Web interface.

3.2.1.1. Hardware Design

The aim of Smart Object hardware design is to put together transceiver, microcontroller, protocol stack and application into few hardware components. Resulting Smart Objects are expected to be low cost, low power and compact devices.

3.2.1.1.1. System on Chip solution

Single Chip radio technology is used where the radio front-end, transceiver and microcontroller are integrated together with flash, memory and other peripherals. A very small bill-of-materials is usually required in addition to the SoC in order to make an independent wireless node, usually including RF matching components, the antenna, crystals and power supply, along with possible sensors and actuators.

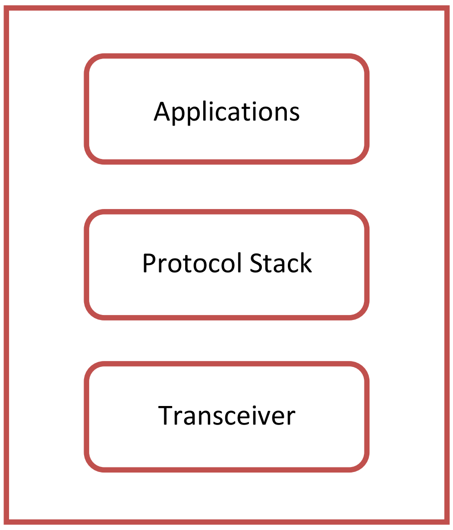


Figure 15: SoC architecture

All the software for the device runs on the SoC microcontroller and is stored in its flash. This includes hardware drivers, a possible embedded operating system, the protocol stack and the smart object application.

System on chip solution is attractive because it minimizes cost and size, and takes full advantage of the system integration. The drawback of this approach is that the integration of all software on the same microcontroller increases complexity and development time.

3.2.1.1.2. Two chips solution

In a two-chip solution, the application processor and transceiver are separate. A variation of this approach, where the transceiver also includes the protocol stack, is called a network processor and is discussed in the next section. An application processor typically communicates with the transceiver over a universal asynchronous receiver/transmitter (UART) or serial peripheral interface (SPI). The protocol stack is integrated with the embedded application, drivers and OS on the application microcontroller. This leaves the developer almost complete freedom in the choice of application microcontroller, which may have special embedded control, signal processing or performance requirements.

The downside of this approach is that the protocol stack and the embedded application need to be integrated in the same microcontroller. As with the single-chip solution, this integration may require extensive engineering and testing. In addition, many protocol stacks are tied to a particular microcontroller or radio transceiver, making it difficult to move between microcontrollers.

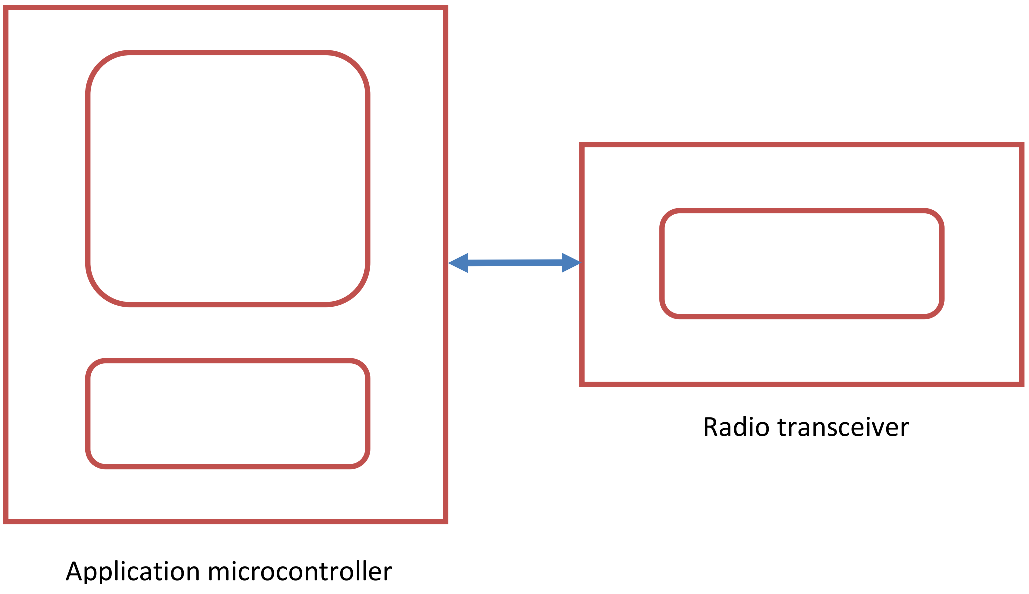


Figure 16: Two-chip architecture

3.2.1.1.3. Network processor

This configuration is also based on the use of two chips, even if the code partitioning differs from the previous one. In this case, the application microcontroller hosts exclusively the application; protocol stack and transceiver are put together into a network processor.

A network processor is often realized on a SoC radio chip, with network processor firmware. Typically a much smaller SoC (less flash and RAM) is required than for a single-chip solution as there are no applications running on the chip.

Communication with a network processor is typically over a UART or SPI interface. This is often realized as an extended socket-like protocol. Thus, with this model, the use of a 6LoWPAN network requires no integration with the application microcontroller other than the use of a protocol over the local interface. The network processor model is often used also with edge routers, and is easy to integrate with operating systems such as Linux.

The downside of this approach is that two chips are required, which may not be possible for devices with extreme cost limitations. Furthermore, network processors cost slightly more than transceivers as they include a microcontroller, flash and memory.

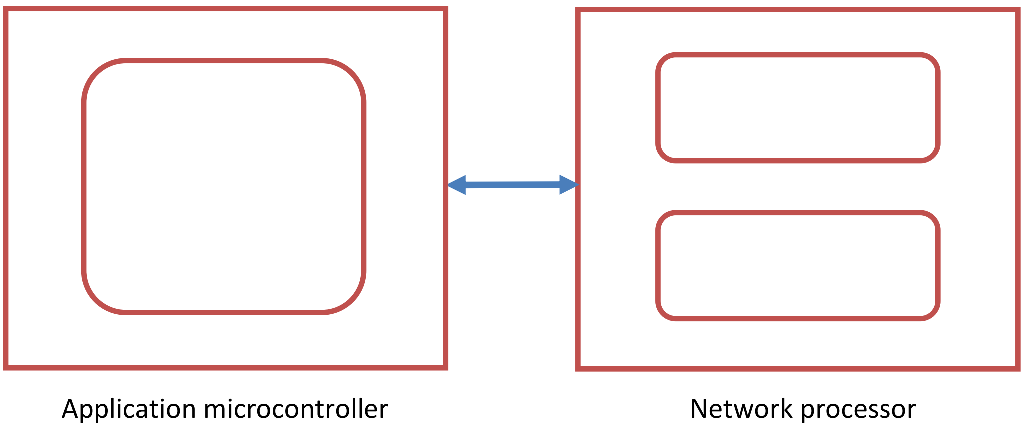


Figure 17: Network processor architecture

3.2.1.2. Operating Systems

The behavior of a smart object is defined by the software running on its microcontroller. It resides into the microcontroller's ROM. When the smart object is switched on, the microcontroller runs the software. Due to the power and cost constrains, smart objects have few amount of RAM (in the order of tens kilobytes) and few amount of ROM (in the order of hundreds of kilobytes). For this reason, the memory footprint of the software must be small enough to run within the given limitations and must not use too much dynamic memory.

Usually, smart objects use an operating system, which is much smaller and less resource-consuming than general purpose operating system. Nowadays, both event-driven (Contiki, TinyOS) and multi-threaded (FreeRTOS) operating systems are available for smart objects.

Event-driven programming requires less memory than multi-threaded programming because there are no threads that require stack memory. The entire system can run as a single thread, which requires only one single stack. The drawback is that only one program can run at the same time.

Moreover, the smart object software must implement the communication protocols used by the smart objects. Typically the protocols stack is provided by the operating system.

3.2.1.3. Physical Communication Standards

Smart objects communicate with each other, but the choice of communication technologies varies between different applications and different environments. Next, we discuss three different physical communication mechanisms for smart objects.

IEEE 802.15.4 is a low-power radio standard designed for low-data-rate applications such as smart objects. It has a maximum data rate of 250,000 bits/s and operates over a set of license-free radio bands in the 868 MHz, 918 MHz, and 2.4 GHz ranges. Packets have a maximum size of 127 bytes. Many emerging standards and specifications are built on top of 802.15.4 including WirelessHART, ISA100a, 6LoWPAN, and ZigBee. There are several implementations of 802.15.4 available, both in hardware and as a combination of hardware and software.

Radio Frequency Identification (RFID) is the wireless non-contact use of radio-frequency electromagnetic fields to transfer data, for the purposes of automatically identifying and tracking tags attached to objects. Some tags require no battery and are powered and read at short ranges via magnetic fields (electromagnetic induction). Others use a local power source and emit radio waves (electromagnetic radiation at radio frequencies). The tag contains electronically stored information which may be read from up to several meters away. Unlike a bar code, the tag does not need to be within line of sight of the reader and may be embedded in the tracked object.

RFID tags are used in many industries. An RFID tag attached to an automobile during production can be used to track its progress through the assembly line. Pharmaceuticals can be tracked through warehouses. Livestock and pets may have tags injected, allowing positive identification of the animal.

Since RFID tags can be attached to clothing, possessions, or even implanted within people, the possibility of reading personally-linked information without consent has raised privacy concerns which will be tackled in another specific deliverable.

Near Field Communication (NFC) is a set of standards for smartphones and similar devices to establish radio communication with each other by touching them together or bringing them into close proximity, usually, no more than a few centimeters. Communication is also possible between an NFC device and an unpowered NFC chip, called a "tag". NFC is a set of short-range wireless technologies, typically requiring a distance of 10 cm or less. NFC operates at 13.56 MHz on ISO/IEC 18000-3 air interface and at rates ranging from 106 kbit/s to 424 kbit/s. NFC always involves an initiator and a target; the initiator actively generates an RF field that can power a passive target.

3.2.1.4. Protocols Stacks

Before the consensus to adopt IP for smart objects became a reality, several non-IP solutions were developed and are still being deployed. Until recently, the IP architecture was often considered too heavyweight to use for low-power short-range networks. Therefore, a number of custom protocol stacks and architectures were developed. In this paragraph, we provide a high-level overview of two non-IP protocol specifications: ZigBee and Z-Wave . Both have been developed for specific low-power, short-range, and low-bit-rate radios, and smart objects are their main application areas. Generally, custom protocol stacks are incompatible with the IP architecture. At the end of the paragraph the 6loWPAN protocol stack is briefly illustrated.

ZigBee is a proprietary specification for wireless communication between smart objects based on the specific IEEE 802.15.4 radio link layer. The ZigBee specification is owned by the ZigBee Alliance.

ZigBee is based on the IEEE 802.15.4 standard and does not provide any alternatives as underlying radios. The ZigBee protocols are defined around the concepts and addressing modes provided by the underlying IEEE 802.15.4 radio, making it difficult to adapt the ZigBee protocols to other radios.

ZigBee specifi es three different device types: the ZigBee Coordinator (ZC), the ZigBee Router (ZR), and the ZigBee End Device (ZED). These three devices play different roles in a ZigBee network.

Each of the ZigBee device types has been designed for a specific deployment. ZCs and ZRs have a higher power requirement than ZEDs and cannot be battery-powered. The ZED has a lower power requirement and achieves a long lifetime on batteries. Regarding IEEE 802.15.4, ZC and ZR are fully functional devices (FFDs), whereas the ZEDs are reduced function devices (RFDs).

The ZC is responsible for bootstrapping the network. During the bootstrapping process, the ZC chooses the Personal Area Network (PAN) identifier that will be used by the network, as well as the physical radio channel on which the network will operate. After bootstrapping, the ZC acts as a normal ZR device. ZEDs are off most of the time, thus they are not able to receive any traffic sent to them. Instead, they periodically wake up and check for messages at the ZR with which they are associated. The ZR buffers data sent to their ZED nodes and sends these data whenever they get a poll request from a ZED. The ZED transmits data to the ZR at any time, since the ZR is always awake. The wake-up schedule for ZED is defined by the application developer, not by the ZigBee specification. The number of ZEDs associated with a ZR is limited. In the ZigBee 2007 specification, a ZR can handle a maximum of 14 ZEDs.

The ZigBee specification is divided into three layers: the network (NWK) layer, the application support (APS) layer and the application framework (AF) layer. In addition to the three layers, a cross-layer entity called the ZigBee Device Object (ZDO) is also present in the architecture.

A detailed description of the protocol stack is out of the scope of the present document.

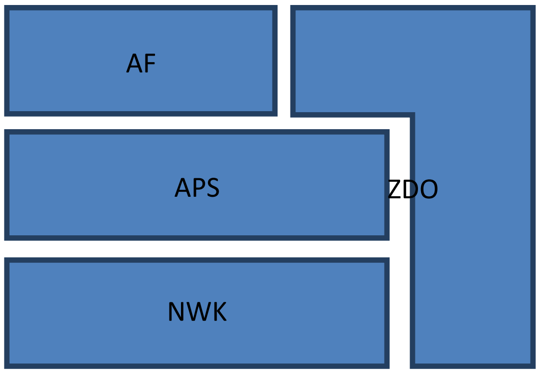


Figure 18: ZigBee Protocol Stack

Z-Wave is an alliance that developed its own patented low-power RF technology for home automation and small residential environment. The Z-wave technology is not IP-based and has its own physical, MAC, networking, transport, and application layers. The application layer makes use of command classes that describe devices and the language used to communicate with these devices.

The main application of Z-Wave products is home automation such as garage doors, alarm systems, door locks, sensors for HVAC and energy management, lighting and windows, home healthcare, sprinklers, and other home applications. Z-Wave provides a developer kit that allows developers and Original Equipment Manufacturers (OEMs) to develop products using the Z-Wave technology due to the use of an API and handles its own certification program.

Z-Wave technology has been designed to be plug-and-play, requiring minimal manual intervention to connect new devices that expand the meshed network. The Z-Wave technology has been designed to be low power so it is used on both main-powered and battery-operated nodes such as smoke detectors or other types of sensors.

The RF Z-Wave technology uses Binary Frequency Shift Keying (B-FSK) modulation and operates in the sub-1 GHz band. Since 2008 it also supports the 2.5 GHz band for a throughput between 9.6 and 40 Kilobits/s (performances indicated by Zensys). The MAC layer uses link layer acknowledgment and retransmission with collision avoidance and checksum for error detection.

The ZM3102 Z-Wave ® Module is an integrated RF communication module using the unlicensed short-range frequency band 902 — 928 MHz in the United States and 868.0 – 868.6 MHz in Europe.

The technology is supported and promulgated by the Z-Wave Alliance (www.z-wave.com).

IPv6 over Low power Wireless Personal Area Network (6LoWPAN) standards enable the efficient use of IPv6 over low-power, low-rate wireless networks, such as IEEE 802.15.4, on simple embedded devices through an adaptation layer and the optimization of related protocols.

Despite the initial scepticism of many researchers about the suitability of the Internet architecture for sensor networks, today the general trend is to move away from proprietary or closed standards solutions to embrace IP. In fact, the performance advances of recent 32-bit microcontrollers and the availability of highly optimized protocol stack implementations, makes it feasible to add IP connectivity to smart objects. This trend is also confirmed by the ZigBee Alliance and its choice of IP for the Smart Energy 2.0 Profile. Thanks to IPv6 huge address space, every smart object can be connected readily to other IP-based networks, without the need for intermediate entities like translation gateways or proxies.

Given the limited packet size and other constraints of Low-Power Wireless Personal Area Networks, an adaptation layer to perform header compression, fragmentation and address auto-configuration is needed to use IPv6. The 6LoWPAN IETF Working Group has already defined the format for adaptation between IPv6 and IEEE 802.15.4. The ideal use of 6LoWPAN is with applications where embedded devices need to communicate with Internet-based services using open standards able to scale across large network infrastructures. In Figure 19, the 6LoWPAN protocol stack is shown.

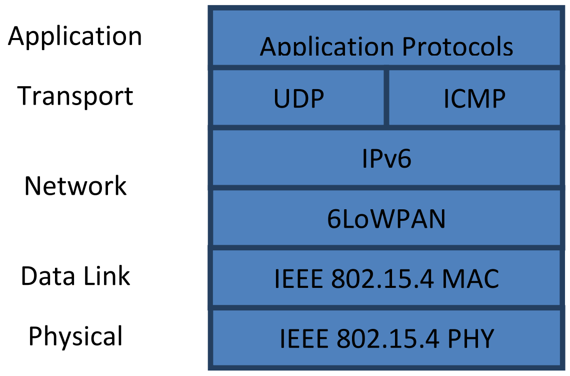


Figure 19: 6LoWPAN Protocol Stack

The 6LoWPAN architecture is made up of low-power wireless area networks (LoWPANs), which are connected to other IP networks through edge routers. The edge router plays an important role as it routes traffic in and out of the LoWPAN, while handling 6LoWPAN compression and Neighbour Discovery for the LoWPAN. If the LoWPAN needs to be connected to an IPv4 network, the edge router will also handle IPv4 interconnectivity. Each LoWPAN node is identified by a unique IPv6 address, and is capable of sending and receiving IPv6 packets. Typically LoWPAN nodes support ICMPv6 traffic such as “ping”, and use the User Datagram Protocol (UDP) as a transport protocol. Adaptation between full IPv6 and the LoWPAN format is performed by routers at the edge of 6LoWPAN islands, referred to as edge routers. This transformation is transparent, efficient and stateless in both directions. Furthermore, 6LoWPAN does not require an infrastructure to operate, but may also operate as an ad hoc LoWPAN.

3.2.1.5. Service and Resource Discovery

Service and resource discovery is the process by which devices on a network learn what services/resources are available. Without a mechanism for service/resource discovery, new devices do not function properly as they have no way to discover services/resources they may need, and no way to announce that they have services/resources available.

Service/Resource discovery is important both for bootstrapping a network and for performing periodic discovery of a network in steady state. In steady state, new devices enter and offer new services/resources to the network. Also, the network may provide a new service/resource that the devices can use.

Typical protocols used for service discovery on embedded devices include the service location protocol (SLP), universal plug-n-play (UPnP) and devices profile for web services (DPWS). Some application protocols such as ZigBee CAP or MQTT-S have their own built-in discovery features. Frameworks such as OGC or CoRE also have built-in service discovery and description mechanisms.

3.2.1.6. Application Protocols

This paragraph covers technologies related to smart objects accessibility and interoperability across Internet. Even if these technologies are strictly related to the IP world, they are also used for non-IP smart objects, assuming that they are interconnected to Internet by the mean of an IP application gateway.

Application protocols can be defined as all the messages and methods having to do with inter-process communication via the Internet Protocol. IP-based smart objects make use of most of the same application protocols used in the Internet for communication between machines and services. Application protocols are equally important for 6LoWPAN, and 6LoWPAN is challenging in this respect.

The limitations of 6LoWPAN such as small frame sizes, limited data rates, limited memory, sleeping node cycles, along with the mobility of devices make the design of new application protocols and the adaptation of existing ones difficult. Furthermore, the autonomous nature of simple embedded devices makes autoconfiguration, security and manageability all the more important.

One of the mostly used paradigms of Internet application protocols is the web services paradigm. Web services are defined by the W3C as a software system designed to support interoperable machine-to-machine communications over a network. Web services enable the communication between processes using well-defined message sequences with the Simple Object Access Protocol (SOAP), or stateless resources with REpresentational State Transfer (REST) style design.

There are several ways to implement the web service concept. Some implementation are built on mechanisms (service-oriented) that require significant processing power and communication bandwidth, whereas others (resource-oriented) are more lightweight.

The resource constraints inherent in smart objects regarding processing power, energy, and communication bandwidth necessitate the use of lightweight mechanisms.

The REST paradigm models objects as web resources, each with a URL accessible using standard HTTP or CoAP methods (GET, POST, PUT and DELETE). These interfaces can be described using the Web Application Description Language (WADL). This REST paradigm is widely used on the Internet between web sites. The content of REST HTTP/CoAP messages can be of any MIME content, although XML and JSON are common in machine-to-machine applications.

3.2.2. Chosen smart object technologies

This paragraph briefly describes the Smart Object assets available from BUTLER partners. The following table summarizes them. All the Smart Objects described in this paragraph fulfill the BUTLER requirements.

Hardware	Operating System	Network Stack	Why this smart object has been chosen?
STM32W eval-boards	Contiki-2.6	6LoWPAN/CoAP	Low Cost, several protocol stacks availability, sensors & actuators can be easily added
ST SPEAr 320 Eval board	Linux/ JVM/OSGi	IP, 6LoWPAN, ZigBee PRO	Low cost and power, connectivity and complex system support
Z1 platform	Contiki-2.6 TinyOS	802.15.4 (ZigBee), 6LowPAN	Low cost and power – MAC layer open – Very large variety of analog sensors
TsmoTe	FreeRTOS	IP, ZigBee PRO, NFC, 6LowPAN, CoAP	Low-cost, low-power platform for smart scenarios, enabling fast development of M2M applications with a wide range of communication interfaces and sensors/actuators
Waspmote	No OS	802.15.4, ZigBee PRO, 868kHz	Mother board dedicated to Smart domains

3.2.2.1. STM32W System on Chip

The STM32W microprocessor is a fully integrated System-on-Chip that integrates a 2.4 GHz, IEEE 802.15.4-compliant transceiver, 32-bit ARM® Cortex™-M3 microprocessor, Flash and RAM memory, and peripherals of use to designers of WSN systems.

Figure 20 provides a high level block diagram of the STM32W System-on-Chip (SoC).

STM32W is supported by the Contiki Operating System.

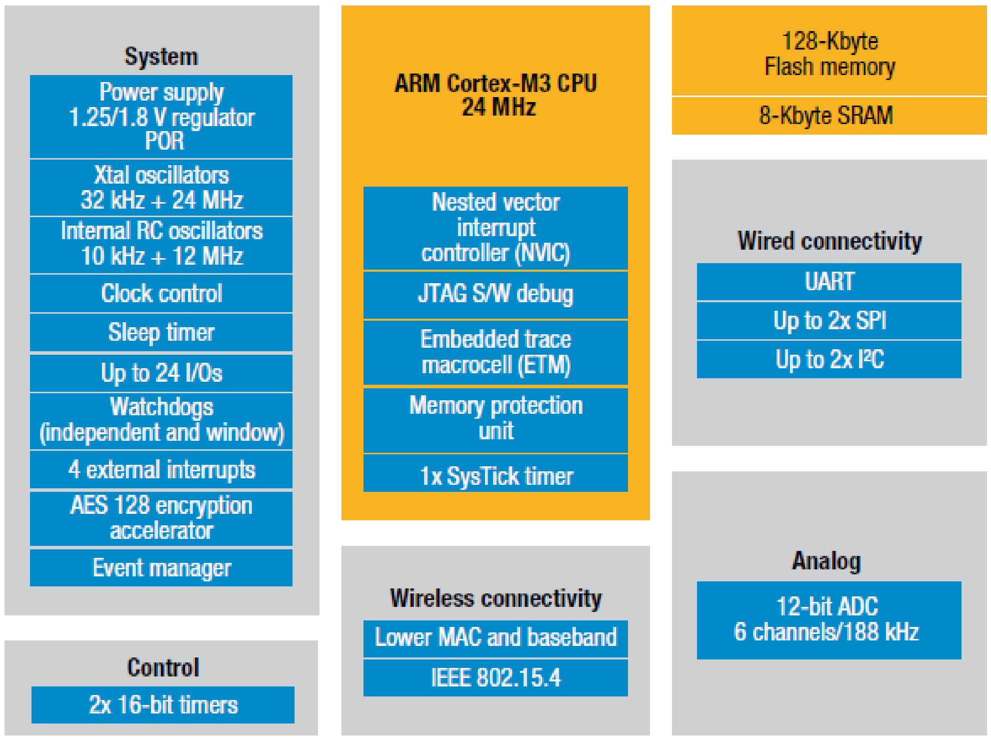


Figure 20: STM32W block diagram

3.2.2.1.1. Evaluation Boards

There are two evaluation boards for STM32W chipset: MB851 and MB954 (shown respectively in Figure 21 and Figure 22) based on STM32W108CB/ STM32W108CB microcontroller. Both boards are equipped with the following:

- Precision analog temperature sensor (STLM20W87F)
- Digital MEMS 3-axis accelerometer (LIS302DL).
- Button sensor to catch external events
- LED to simulate on/off actuators
- UART1 for serial line I/O (input events and debugging)
- Antenna

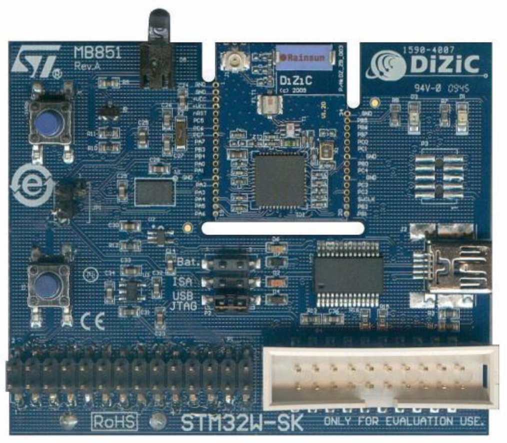


Figure 21: MB851 application board

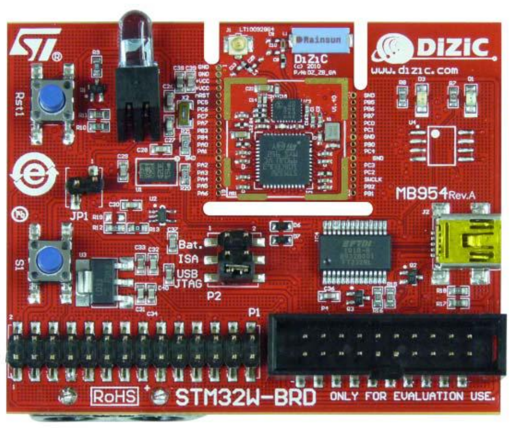


Figure 22: application board with Power Amplifier

3.2.2.1.2. Additional Sensors and Actuators

Both MB851 and MB954 feature an extensive set of GPIO pins that allows the board to connect external peripherals such as additional sensor and/or actuator hardware. In particular, the related STM32W GPIO pins may be individually configured as:

- General purpose output or alternate output controlled by peripheral. (Both in push-pull and open-drain mode),
- Digital input (Push-pull, open-drain, and floating mode),
- Analog input22.

These features offer a high degree of flexibility that makes it easy to meet demanding system requirements For example, a power switch able to turn on/off appliances or perform dimming functionalities could be added to the node, connecting it, by the means of dedicated power driver, to a GPIO pin configured in the output mode.

3.2.2.1.3. Software

The software, as illustrated in the following figure, will be made of:

- application implemented on top of the Contiki Operating System,
- STM32W peripheral drivers,
- Hardware Abstraction Layer for ST implementation of the IEEE 802.15.4 standard (802.15.4 library).

The firmware image is composed of the OS, the drivers and the application, all linked together.

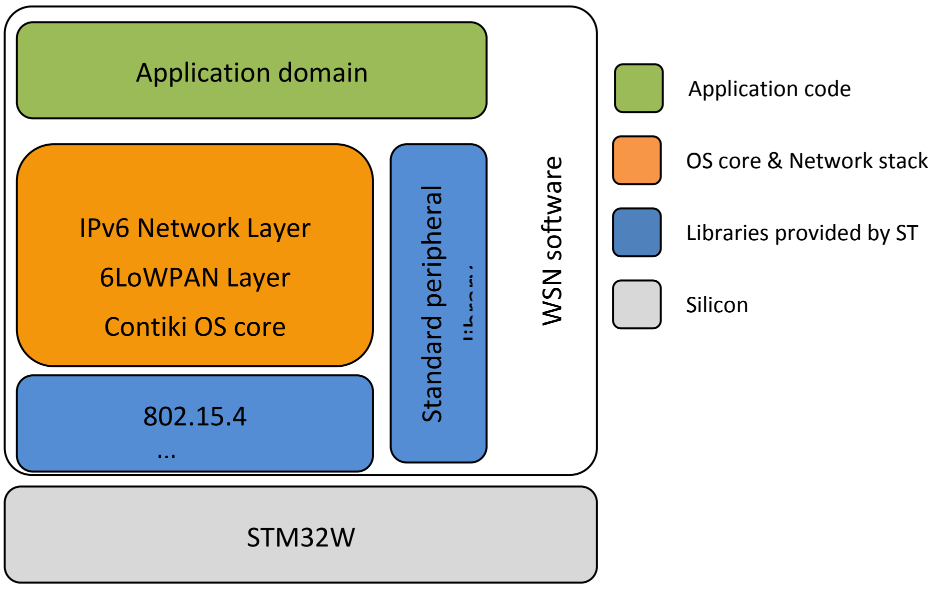


Figure 23: Smart Object software architecture

[22] Actually, only six GPIO pins can be configured as analog inputs for the internal Analog/Digital Converter (ADC).

Additionally, another line can be used to input external analog voltage reference to the ADC, or it can output the internal analog voltage reference from the ADC.

3.2.2.1.4. Motivations

The STM32W bring outstanding radio and low-power microcontroller performances in a single system-on-chip (SoC). With a configurable total link budget of up to 109 dB and the efficiency of the ARM Cortex-M3 core, the STM32W is the perfect fit for the wireless sensor network market. Compliant with the IEEE 802.15.4 radio standard, this open and flexible platform supports the most popular protocol stacks such as RF4CE, ZigBee IP and Smart Energy Profile 2.0 and more.

Another key feature of STM32W SoC is the huge availability of peripherals as configurable Serial Controller (I2C, SPI, UART), configurable I/Os, ADC and Timers. This allows application integration with external extra resources.

The chosen STM32W evaluation boards (MBXXX) well fit the BUTLER smart home/office scenarios, since they already come with an analog temperature sensor, 3-axis digital accelerometer and on/off switch actuators. Moreover, thanks to GPIO and serial controller availability, they can be easily integrated with others sensing or acting devices.

3.2.2.2. SPEAr320 Embedded MPU

The SPEAr320 is a member of the SPEAr family of embedded MPUs, optimized for industrial automation and consumer applications. It is based on the powerful ARM926EJ-S processor (up to 333 MHz), widely used in applications where high computation performance is required. It is an excellent candidate for running Machine-to-Machine (M2M) gateway functionalities for example.

In addition, SPEAr320 has a MMU for virtual memory management - making the system compliant with Linux operating system. It also offers 16 KB of data cache, 16 KB of instruction cache, JTAG and ETM (Embedded Trace Macrocell) for debug operations.

A full set of peripherals allows the system to be used in many applications, some typical applications being factory automation, printer and consumer applications. Figure 24 shows a top level block diagram of the processor.

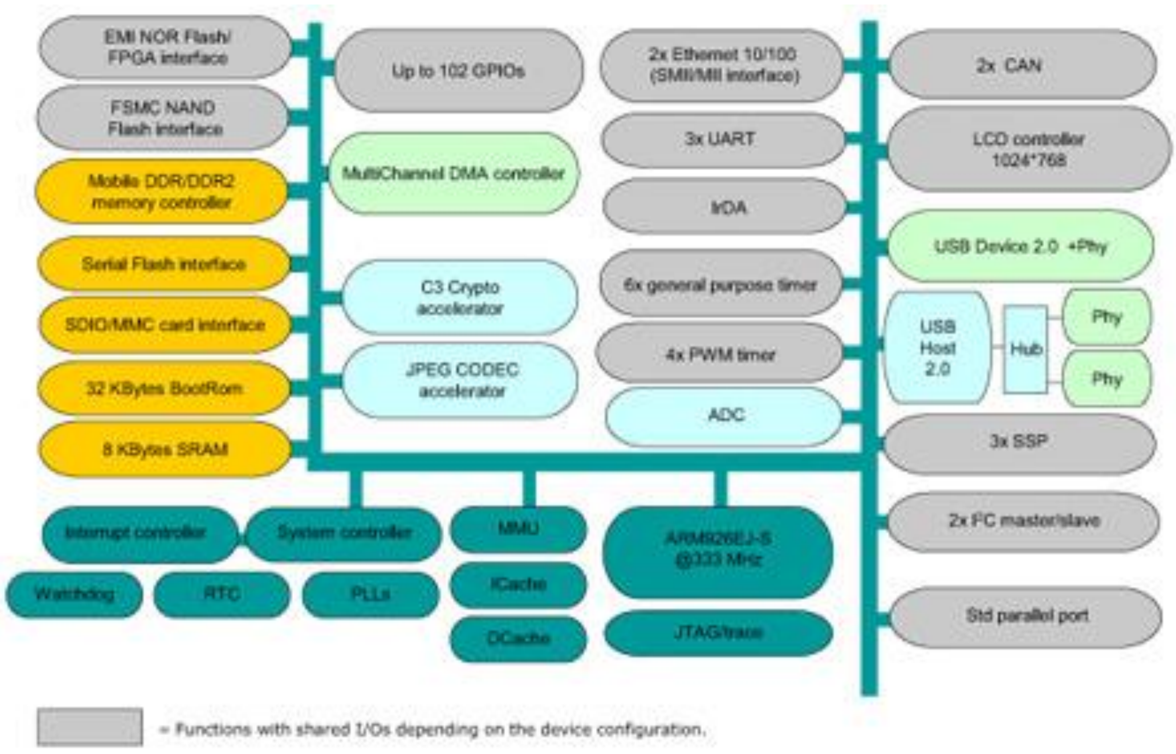


Figure 24: SPEAr320 eMPU Block Diagram.

3.2.2.2.1. Evaluation Boards

The SPEAr320 PLC Evaluation Board kit consists of a CPU board and an application board.

The CPU (SPEAr core) is mounted on the CPU board that, in turn, can be plugged on the application board in order to evaluate the MII (Media Independent Interface) automation mode.

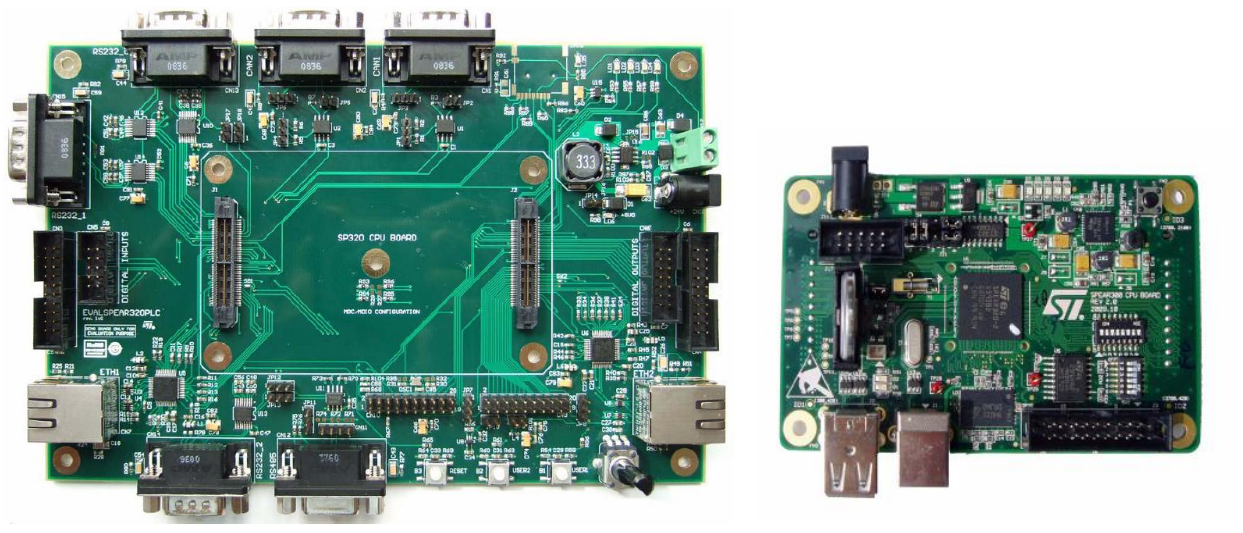


Figure 25: SPEAr320 MII Mode Application Board and SPEAr320 CPU evaluation board.

CPU board features:

- SPEAr320 embedded MPU (333 MHz)
- Up to 2 Gbit DDR2 333 MHz (standard 128 Mbytes)
- Up to 16 Mbyte Serial Flash memory (standard 8 Mbytes)
- Two USB 2.0 full host port channels
- One USB 2.0 host device port
- One serial port (up to 115 baud)
- JTAG Debug ports

Application Board features:

- 2 x Ethernet RJ-45 connectors (ST802RT1A)
- 2 x CAN DB9 plug connectors
- 3 x RS-232 DB9 plug connectors (ST3232EBTR)
- 1 x RS-485 DB9 socket connector (ST3485EBDR)
- Digital input connectors (parallel and serial) compatible with STEVAL-IFP007V1, STEVAL-IFP008V1 and STEVAL-IFP004V1 evaluation boards
- Digital output connectors (parallel and serial) compatible with STEVAL-IFP009V1, STEVAL-IFP001V1, STEVAL-IFP002V1 and STEVAL-IFP006V1 evaluation boards
- On-board temperature sensor (STLM20W87F) and potentiometer (analog input for ADC)
- Analog extension connector featuring 8 ADC lines
- General-purpose extension connector with GPIOs and I2C functionality
- DC/DC converter L7986A (+24 V / +5 V)
- MicroSD card socket
- 4 LEDs, 2 general-purpose buttons and system reset button

3.2.2.2.2. Software

The SPEAr320 CPU board runs a customized Linux distribution, called the SPEAr Linux Support Package (LSP). SPEAr Linux Support Package consists of a collection of all the Linux drivers that control the specific hardware controllers embedded in SPEAr as well as the set of bootloaders to perform the low level hardware configuration and the loading of the Linux OS.

The user manual of the LSP is available for public download from the following link: http://www.st.com/internet/mcu/product/247244.jsp.

3.2.2.2.3. Motivations

Embedded applications today demand increasingly higher levels of performance and power efficiency for computing, communication, control, security and multimedia.

ST's SPEAr® family of embedded MPUs meet these challenges head-on with state-of-the-art architecture, silicon technology and intellectual property, targeting networked devices used for communication, display and control.

The new SPEAr300 delivers robust processing with a 333 MHz ARM926EJ-S core that supports complex operating systems like Linux. The CPU also offers 16 Kbytes of data cache, 16 Kbytes of instruction cache, JTAG and ETM (embedded trace macrocell) for debug operations.

A set of tailored peripheral interfaces, hardware accelerators and controllers make the SPEAr300 a smart choice for BUTLER Smart Object Gateway platform implementation.

3.2.2.3. Zolertia Z1 platform

The Z1 module from Zolertia is a general purpose platform for Wireless Sensor Network (WSN) compatible with the Tmote™ family motes with several enhancements. It is designed to facilitate the enable object and environment to interact in seamless way to monitor and control environment. Equipped with two on-board digital sensors (accelerometer and temperature), it makes possible to build smart networks from scratch. The application can be directly connected to the Internet of Things (IoT) via IPv6. The Z1 module holds a MSP430 microcontroller and a CC2400 TI antenna to communicate with the IEEE 802.15.4 standard. It offers a maximum flexibility and expandability while improving low power consumption with low power sensors.

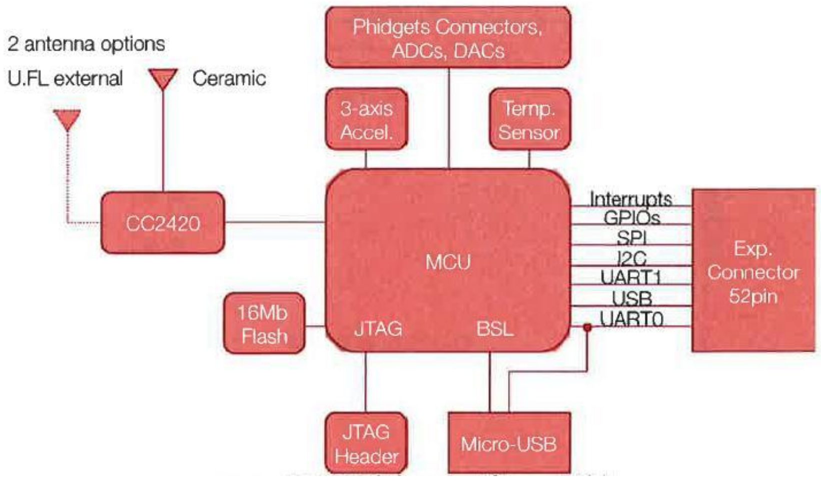


Figure 26: Functional block diagram of the Z1 module

3.2.2.3.1. Evaluation board

The Z1 platform is an evaluation board holding a MSP430 microcontroller that is a 16-bits ultra-low power MCU. The embedded antenna emits at 2.4GHz and is compliant with the IEEE 802.15.4 standard and with the 6lowPAN adaptation layer. Two on-board digital sensors are present: a 3 axes accelerometer and a low power digital temperature sensor with ±0.5°C accuracy. The memory is composed of 92kbytes of flash and 8kbytes of RAM. A serial flash of 16Mbit is used for extra storage. The board is powered by two AAA batteries, by coin cell or by USB cable. 52-pins expansion connector allows adding external ADC, DAC, UART, USB, SPI or interrupts. Four analog Phidgets sensors can be simultaneously connected to the board. TinyOS or Contiki may be used as ultra-light embedded operating systems.

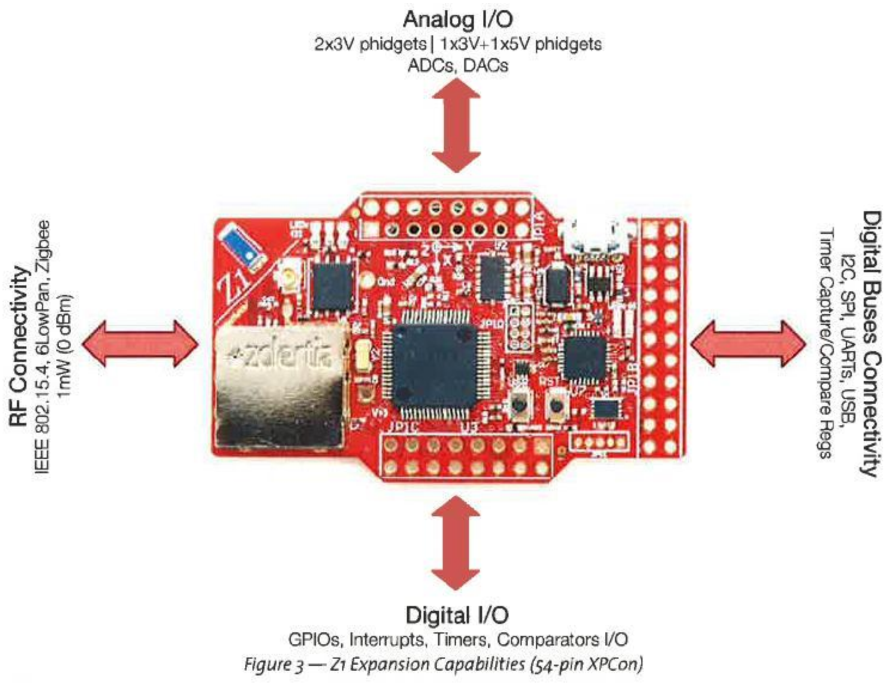

Analog I/O
2x3V phidgets | 1x3V+1x5V phidgets
ADCs, DACs
RF Connectivity
IEEE 802.15.4, 6LowPan, Zigbee
1mW (0 dBm)
Digital Buses Connectivity
I2C, SPI, UARTs, USB,
Timer Capture/Compare Regs
Digital I/O
GPIOs, Interrupts, Timers, Comparators I/O
Figure 3 — Z1 Expansion Capabilities (54-pin XPCon)

Figure 27: Z1 evaluation board

3.2.2.3.2. Additional sensors and actuators

We have used many Phidgets sensors in the context of the Bulter projects:

- Vibration Sensor: this sensor measures vibrations on a surface. It embeds a piezoelectric transducer. As the transducer shifts from the mechanical neutral axis, bending creates strain within the piezoelectric element and generates voltage.
- Magnetic sensor: this is a ratio-metric Hall-effect sensor which provides a voltage output that is proportional to the applied magnetic field.
- Motion Sensor: it detects changes in infrared radiation that occurs when there is an object/person's movement with a different temperature from the surroundings. This sensor is also characterized by a narrow sensing area.
- Gas pressure sensor: it measures absolute gas pressure from 20 to 250 kPa (2.9 to 36.3 psi) with a maximum error of ±1.5%. It is suitable for measuring vacuum, or atmospheric pressure; it can also be used as a crude barometer.
- Temperature & humidity sensor: it measures the relative humidity (RH) from 10% to 95% with a typical error of 2% RH at 55% RH. It also measures ambient temperature in the range of -30°C to +80°C with a typical error of ±0.75°C in the range 0°C to 80°C.

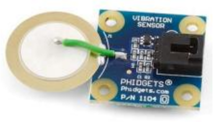
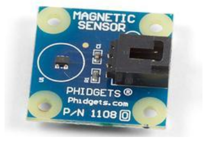
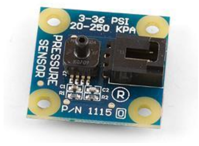
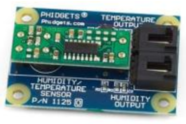
Phidget sensors	vibration	magnetic	motion	pressure	temperature & humidity
Photo					
Accuracy		± 0.5%		±1.5%	±2°C - ±5%
Sensibility		[0,1000] Gauss	5 meters	[20,250] kPa	[-40,100]°C [10,90]% RH
Maximum conso	400µA	2mA	15µA	5mA	3.9mA 3.9mA
Operating temp. interval	[-20,70]°C	[-20,85]°C	[-20,85]°C	[0,85]°C	[-30,80]°C [-30,80]°C

Figure 28: Z1 Plug-in sensors characteristics

3.2.2.3.3. Software

The Z1 platform supports two operating systems: TinyOS and Contiki. It seems to be easier to use the Phidget sensors and to develop applications with Contiki. As Contiki is an open source operating system for the Internet of Things, it allows tiny, battery-operated low-power systems to communicate with the Internet. It is well suited to our problematic.

3.2.2.3.1. Motivations

The Z1 platform is a low cost, low power platform. It handles a wide variety of Phidget analog sensors. It is easy to use and to program and is compatible with the 6LowPAN adaptation layer. The data sensed by the Phidget are easy to collect thanks to Contiki operating system. This platform is very open and enables various experiments at the physical and MAC layers. We use it to collect large data set from various physical sensors and from the radio indicators in order to harvest entropy to forge true random number in a very scarce device with the final goal to enhance the security.

3.2.2.4. The TSmarT Platform

The TSmarT platform for wireless monitoring, remote control and M2M applications consists of the programmable devices TSmoTe and TSgaTe, and a full set of expansion modules supporting the most common wireless technologies (IEEE 802.15.4, ZigBee, Wi-Fi, GPRS/3G, GPS, RFID/NFC, ...). The expansion modules may be plugged into the TSmoTe or TSgaTe when needed by each use case scenario. This way, the TSmarT platform can be used in heterogeneous scenarios keeping the cost of a large scale WSN deployment reduced to the minimum. Furthermore, the TSmarT platform completes with a full documented public API and software libraries to facilitate program developers in the process of writing C applications for the TSmoTe and TSgaTe boards. In addition, from TST website can be downloaded the

TSmarT SDK to setup the development environment, which is based on a pure Open Source toolchain (Eclipse, CodeSourcery and Yagarto).

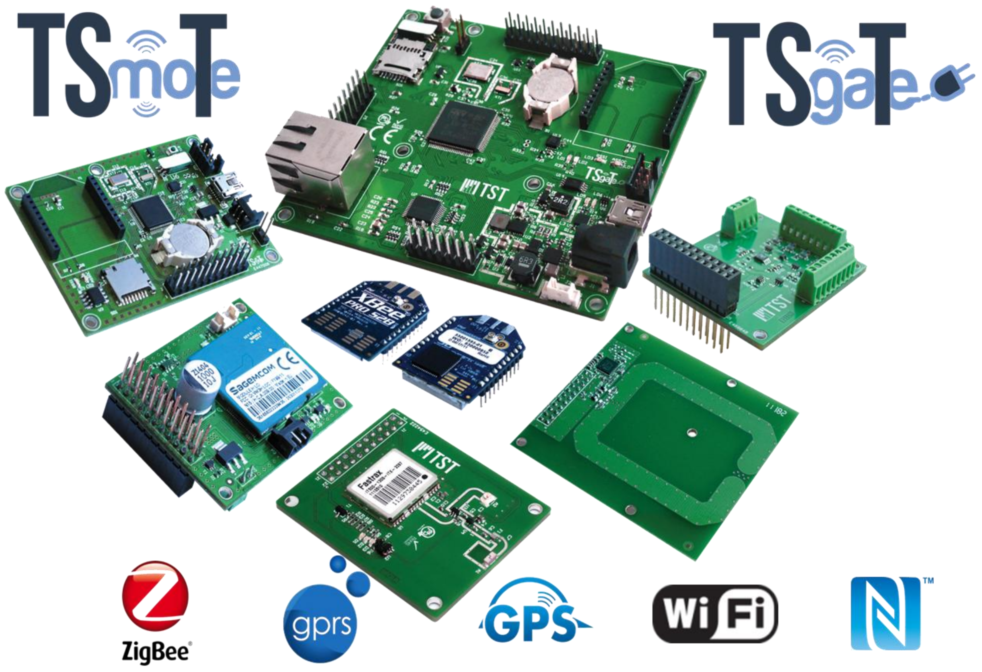


Figure 29: The TSmarT hardware platform

3.2.2.4.1. The TSmoTe as an IoT SmartObject

The TSmoTe is an embedded system that enables a fast and simple development of wireless applications and reduces the time-to-market. It consists on a low-power 32 bit microcontroller with an ARM Cortex-M3 core at 72 MHz with 96 KB RAM and 1 MB Flash memory. Sensors, actuators and other devices can be connected to the TSmoTe through I/Os and serial ports. There are multiple expansion modules available for the TSmoTe: ZigBee, Wi-Fi, GPRS, RFID/NFC, GPS, RS485. At software level, the TSmoTe includes TCP/IP, HTTP and Modbus stacks, as well as the drivers for the expansion modules. Everything runs on top of a real time operating system. Due to its modular design, it is possible to combine the communication technologies needed for a certain application. Thanks to the software libraries and API provided by TST, programming the user application is extremely simple.

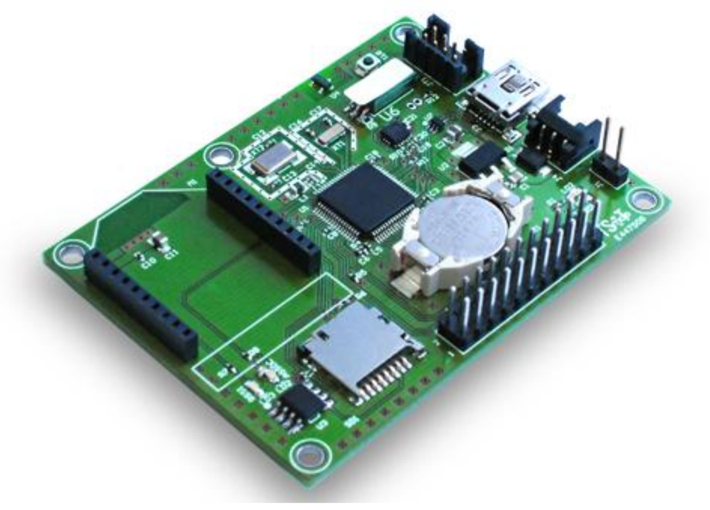


Figure 30: The TSmoTe IoT SmartObject

ELECTRICAL	
Input voltage	4.5 - 12 VDC
Internal voltage	3.3 VDC
Current MCU On	40 mA
Current MCU stand-by	23 uA
Coin cell	CR1225

MECHANICAL	
Dimensions	70 x 52 mm
Connectors	22 pins female slot for expansion modules
	Double row female slot for expansion modules
	8 pin JTAG connector
	Mini USB

MCU	
Microcontroller	32 bits STM with ARM Cortex-M3 core
Clock	72 MHz
Flash	1 MB
RAM	96 KB
SD card	Slot for microSD cards up to 2 TB
Serial interfaces	3 UART, 2 I2C, 1 SPI
I/Os	Up to 6 analog, up to 20 digital

EXPANSION MODULE
ZigBee, Wi-Fi, IEEE802.15.4, GPRS, NFC/RFID, GPS, RS485, Industrial sensor adapter

EMBEDDED SOFTWARE
TCP/IP, HTTP, Modbus stacks

FreeRTOS operating system
TST software libraries

ENVIRONMENTAL	
Operation temperature	-20ºC / +70ºC
Storage temperature	-40ºC / +85ºC
Certifications	CE, RoHS

3.2.2.4.2. Software

The API provided by TST offers a hardware abstraction layer for the TSmarT platform, enabling a high-level programming of the expansion modules with no need to learn their native commands and programming languages. The documentation of TST's software libraries is available for software developers at api.tst-sistemas.es. TST's complete documentation includes manuals for the TSmoTe, TSgaTe and expansion modules, an online API description, sample code and video-tutorials. The documentation tries to facilitate programmers the development of their applications.

The TSmarT family allows advanced debugging options of the source code, breaking points insertions to stop the microcontroller, anytime access to the registers to read the values of the variables and reducing the validation time of the applications. The programming language used to develop software for the TSmarT family is standard ANSI C, well-known by every programmer.

FreeRTOS, the embedded real-time operating system for the TSmarT family, supports multitasking. A complex software application can be divided into several tasks, which are simpler and with less risk to introduce software bugs. The GCC compiler, the development environment Eclipse and the other tools needed to program the TSmarT platform are free and open source, so there's no need to spend any additional amount of money in software licenses to work with these products.

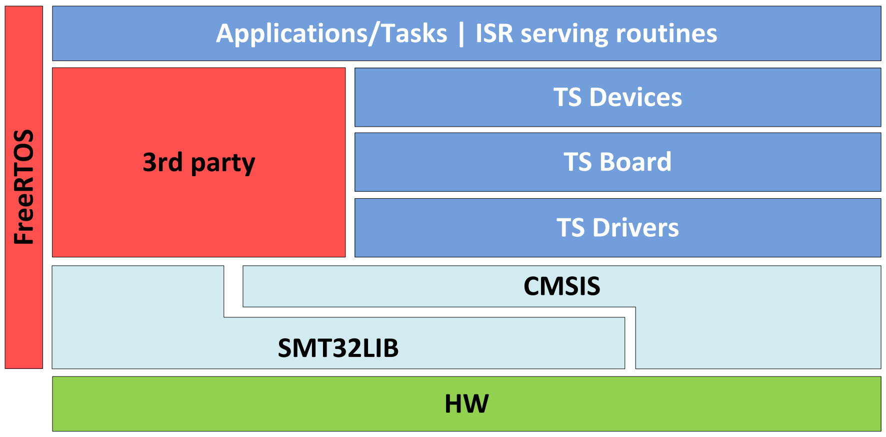


Figure 31: The TSmarT Software Architecture

3.2.2.4.3. Additional Sensors and Actuators

TSmarT allows connecting any type of sensors or data acquisition devices. As an example, here is a list of sensors that have been already integrated into various projects within the TSmarT platform.

Sensor Type	Manufacturer	Data output	Application
Ferromagnetic Parking	Honeywell	Digital I2C	Smart Cities
Container level	Maxbotics	RS-232	Smart Cities
Tank level	Global Water	4-20 mA	Smart Cities
Noise level	JLI Electronics	Analog	Smart Cities
Door/window opening	Medel	Potential free contact	Building Automation
Temperature/Humidity probe	Regin	0-10 V	Building Automation
Soil moisture	Decagon	Analog voltage	Agriculture
Leaf moisture	Davis	Analog voltage	Agriculture
Water pH	Global Water	4-20 mA	Agriculture
Solar radiation	Weatherlink	UART	Agriculture
Anemometer	Davis	Pulses	Agriculture
Rain gauge	Davis	Analog voltage	Agriculture
CO2	E+E	Analog voltage	Environmental
CO	HK Instruments	4-20 mA	Environmental
Power meter	Schneider Electric	Modbus RTU	Energy
Water meter	Itron/Actaris	Pulses	Energy
Current Flow	LEM	4-20 mA	Energy
O2 saturation	Corscience	UART	Healthcare
ECG	Corscience	UART	Healthcare
Temperature	Microchip	Analog voltage	Multiple
Humidity	Honeywell	Analog voltage	Multiple
3-axis accelerometer	Honeywell	Digital I2C	Multiple

3.2.2.4.4. Motivations

The TSmarT platform plays a perfect role as a SmartObject. As a commercial-ready product, TST has developed specific solutions for lighting control, proximity marketing applications and building automation solutions based on ambient intelligence, to name only a few ones.

The TSmarT platform fulfills the requirements imposed by WP1&WP3, what has been proven during the development of the SmartParking solution in the SmartCity scenario. As a live platform, TST engineers keep developing new functionalities to move forward beyond SotA embedded solutions. In new iterations of this SmartParking solution the TSmarT platform will be greatly enriched thanks to the feedback reported during the demonstrator setup.

With its 72 Mhz clock and 32 bits uC the TSmoTe device is capable of running any algorithm proposed in WP2. But not only that, in case of tight limitations in energy efficiency, TSmoTe's clock speed can be tuned so to decrease battery consumption to the minimum. In addition, the different expansion modules allow the TSmoTe device to act both as a SmartObject and a SmartObject Gateway, thus being perfect for the five BUTLER scenarios: SmartCity, SmartHome, SmartShopping, SmartTransport and SmartHealth.

3.2.2.5. The Waspmote Platform

For BUTLER, horizontality means the ability to build applications on top of heterogeneous devices using different communication technologies. We have integrated waspmote nodes into our test platforms in order to include Zigbee integration to the ”BUTLER supported technologies” portfolio. Waspmote is an open source wireless sensor platform specially focused on the implementation of low consumption modes to allow the sensor nodes ("motes") to be completely autonomous and battery powered, offering a variable lifetime between 1 and 5 years depending on the duty cycle and the radio used. The processor consumption is 9 mA and 62 µA (sleep mode) and sensor board consumes maximum of 200mA and maximum peak of 400mA. It uses an AtMega1281L processor of 8 MHz and has a memory capacity of 8K for SRAM and 4 K for EEPROM, with 256K of flash memory and 2GB of SD card capacity.

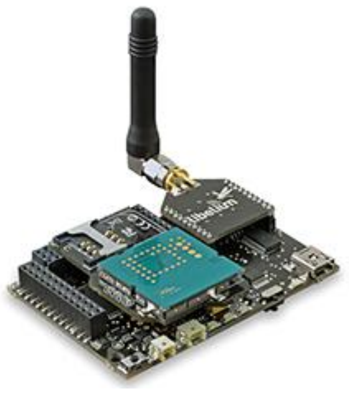


Figure 32: Waspmote communication module

There are several interesting features of the waspmotes, for instance its modular philosophy. By using the same communication nodes, we can connect different sensor modules such as temperature sensors, humidity sensors, force sensors, vibration sensors, GPS modules, GPRS modules, etc.

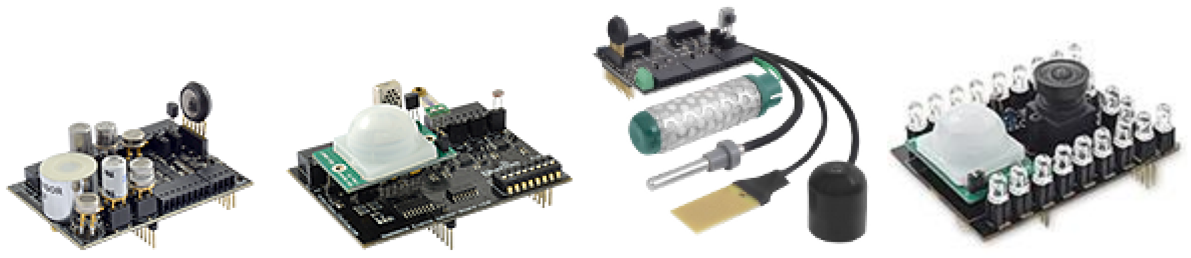


Figure 33: Different types of sensors can be plugged into the same communicating node

Another interesting feature of the Waspmote is to be able to perform over the air programming. This allows deploying test applications without the need of physically connecting to the devices, which is a difficult task once the sensors are deployed in the environment.

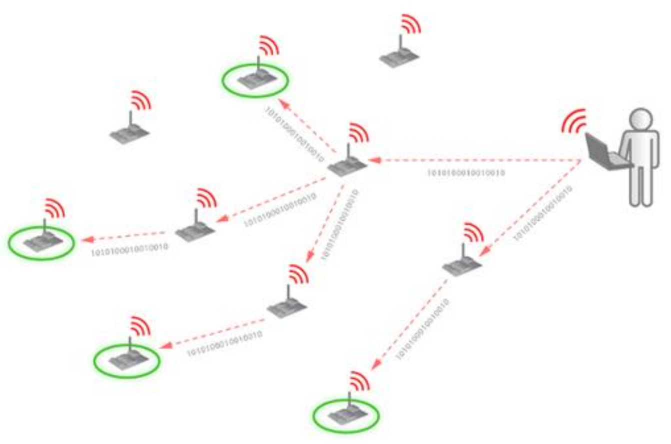


Figure 34: OTA programming of Waspmotes

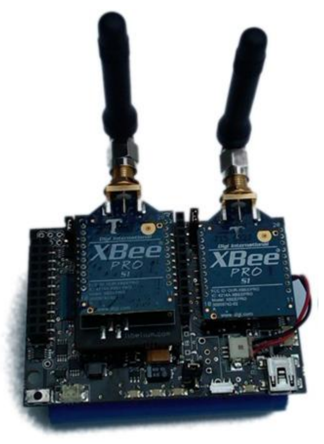


Figure 35: Waspmote node with 2 transceivers

This also implies the ability to support different applications, services or entities running concurrently on the experimentation platform. In order to cope with such aspects, we agreed that having two transceivers in the infrastructure nodes could become a significant advantage as it reduces contention to the communication channel by concurrent applications. One of the transceiver would basically provide support to service provision (and additionally to network control and management) while the other would provide support to the experimentation. The Waspmote platform depicted in Figure 34 is a clear example of such solution. It provides concurrent support to both experimentation (relying on a native IEEE 802.15.4) and service provision (relying on an IEEE 802.15.4 with Digimesh protocol on top). Although both cards are under the controlled of a single microprocessor, interruptions handling allows Waspmote to support both profiles simultaneously.

3.2.2.5.1. Fitbit device: a Smart Object platform

Fitbit develops and markets a family of wearable devices that can help to monitor the physical activities of a user such as walking or being still and reports it in terms of the number of steps taken over certain time period (say today). Unlike other specialized platforms, these devices are designed for common people who engage in common activities such as walking and climbing stairs. The data (number of steps) can also be uploaded to a remote service which in turn generates more detailed information such as calorie consumption, distance travelled or the quality of sleep graphs by integrating other profile data.

We have used the fitbit-one device in the SmartHealth Proof-of-Concept of BUTLER and demonstrated how to measure the physical activity of the user and integrate it with the Diabetes application. As Fitbit is a commercial company, all their products (both hardware and software) are proprietary and detailed technical specification are not publicly available.

The device has a MEMS 3-axis accelerometer to monitor the motion behaviour and an altimeter to measure the number of stairs climbed[23]. The device can be connected to a base station (a wireless sync dongle) which in turn communicates to the computers (only Windows and Mac) wirelessly using the proprietary ANT protocol[24]. Once connected to the computers, the devices can sync their latest data to the remote web service. The wireless communication protocol stack in the ANT protocol enables a wide variety of BUTLER

[23] http://www.fitbit.com/one/specs

[24] http://en.wikipedia.org/wiki/ANT_(network)

devices to communicate in the IMF band with high efficiency and low computational overhead helping the devices to last longer. Also, the latest version of fitbit-one supports Bluetooth 4.0 enabling seamless sync with a selected range of smart phones (iPhone 4S, iPhone 5, iPad-3rd generation, iPod touch-5th generation, Samsung Galaxy S III and Samsung Galaxy Note II). Also, the Lithium-ion polymer type battery is designed to last for 5-7 days under normal operating conditions.
Despite its high performance and efficiency, the fitbit devices have some limitations that do not favor them from being integrated in the BUTLER platforms.

1 The primary concern is the inability to access the data directly from the device. Although there are certain developers APIs available to access the daily summaries, the need to obtain a registration key, lack of instantaneous update interfaces or access to raw accelerometer data logs (or the derived features from that data) is a major concern for seamless integration of these devices to BUTLER horizontal platform.
2 Also, the devices cannot offload the data to web-servers via Linux or Android devices (except the above mentioned Galaxy devices). This constraint hinders the vision of developing a horizontal platform across domains.

Hence, we have decided to use the fitbit devices only for the purpose of benchmark studies where the performance (prediction accuracy) of these devices will be compared with the versions implemented in Android devices (deploying in-home developed sensing and processing components).

3.3. Selected technologies

3.3.1. Advantages of 6LoWPAN

There are a huge range of applications which could benefit from a Wireless Embedded Internet approach (the use of IP technologies for wireless sensor network). Today these applications are implemented using a wide range of proprietary technologies which are difficult to integrate into larger networks and with Internet-based services. The benefits of using Internet protocols in these applications, and thus integrating them with the Internet of Things include:

- IP-based devices can be connected easily to other IP networks without the need for translation gateways or proxies.
- IP networks allow the use of existing network infrastructure.
- IP-based technologies have existed for decades, are very well known, and have been proven to work and scale. The socket API (Application Programming Interface) is one of the most well-known and widely used APIs in the world.
- IP technology is specified in an open and free way, with standards processes and documents available to anyone. The result is that IP technology encourages innovation and is better understood by a wider audience.
- Tools for managing, commissioning and diagnosing IP-based networks already exist (although many management protocols need optimization for direct use with 6LoWPAN)

3.3.2. Technology integrated in the BUTLER platform

Many technological bricks studied for the BUTLER project and relevant in a context-aware system have been detailed in chapter 2. Some of these algorithms are being studied and are simulated. Others are more mature and can be integrated for a first deployment in the future BUTLER platform. In this paragraph, we analyse the candidate technological bricks that may be integrated in the Smart Object of the final platform and may participate to the proof of concept.

3.3.2.1. Security and privacy

The Smart Object handles an identity number that is registered by the authorization server. Thanks to an authentication protocol, the server generates a token. The Smart Object verifies the token validity that allows actions as data processing or resource identification. The whole Smart Object authentication process is detailed in 6.2.

3.3.2.2. Contextual geo-localization

According to the designed localization architecture reported in section 2.3.2, the localization engine is implemented in a centralized manner in the private smart server. This choice has been driven by the fact that smart objects, which are battery powered devices, might not have enough computation capacity to execute the algorithms. Thus, according to this approach, every unknown node just needs to be able to perform range measurements (e.g. RSSI, ToA, TDoA, etc.) with respect to its 1-hop neighbors and then send these collected ranging data to the local smart server where a suitable localization algorithm estimates the position of unknown smart objects. Ranging methods such as RSSI, ToA and TDoA were described in detail in deliverable D2.2 [9] along with their advantages and limitations. To sum up, RSSI is very easy to be implemented in constrained devices but it suffers from large variance and the related range error increases with the distance. Thus, RSSI would be suitable to be applied to room based localization where distances are short. Alternatively, ToA and TDoA are more accurate than RSSI but they require a particular structure of the radio-frequency signal. For instance, IR-UWB systems use impulses of very short duration, in the order of sub-nanoseconds, that allow centimeter level range accuracy. However, UWB devices need larger processing capacity. The current cost is high.

Note that an unknown node can perform range measurements with respect not only to its neighboring anchors (whose coordinates are a priori known) but also with respect to its neighboring unknown nodes. In this last case, it is possible to implement a cooperative localization approach that exploits also the localization data from unknown neighbors (e.g. ranging and estimated position). On the contrary, in a non-cooperative approach, unknown nodes perform range measurements only from the visible anchors. Since a cooperative approach uses more information than a non-cooperative one, it provides improved positioning availability and accuracy. Moreover, thanks to the cooperation, it is possible to reduce energy consumption on the unknown node by simply decreasing the transmitted power such that the node performs range measurements with a closest set of neighbors that allow meeting the required localization accuracy as described in section 2.3.1.3 and D2.2 [9].

According to the chosen smart object platforms reported in section 3.2, the available types of range measurement methods provided by platforms are listed in the following table.

Smart Object Platform	Type of range methods
STM32W eval-boards	RSSI
ZigPos	Phase measurement
TST	RSSI
TI CC2530	RSSI

As it can be observed, all smart object platforms except ZigPos are able to provide RSSI measurements. Thus, those platforms can be used for short range scenarios where the required localization accuracy is more relaxed, for instance, in the follow me application. The ZigPos platform is based on ATMEL's new phase measurement ranging technology and it is suitable for highly accurate real time radio location tracking system. For instance, it can be used in smart shopping and smart transport scenarios.

The ZigPos platform aims to measure the phase. The TOA measurement is difficult in low cost microcontroller applications for the smart objects as it requires high accurate timers. The Atmel phase measurement ranging technology (Atmel Corporation, U.S. Patent 8,405,543 B2) is used instead for the positioning application. In a reflection free scenario, the algorithm can provide positioning accuracy in the range of few centimetres.

The distance measurement based on the phase measurement uses two unmodulated carrier signals with a small frequency difference Δf. These signals are transmitted one after another and the receiver evaluates the phase of the received wave $\emptyset_1$and $\emptyset_2$. The distance d between the stations can then be calculated as:

$$d = \frac{(\emptyset_1 - \emptyset_2)c}{2\,\pi\Delta f}$$

Here, c is the speed of the light.

Each anchor (4-5) is in standard positions which compute this distance measurement. The distance data are communicated to the coordinator that estimates the position from this data. The coordinator is connected to a smart- gateway that provides the position data to the external network.

3.3.2.3. Behavior awareness

Behaviour awareness is a complex module comprising multiple sensing, pre-processing and classification components. As shown in Figure 13, the behaviour awareness architecture can be either centralized or distributed (with varying degrees of distribution) depending on the devices being used for a particular use case. Hence, the framework is being developed in a modular way to optimize sensing and further processing to respect the dynamic resource constraints of the platforms.

Indisputably, the sensing components are deployed on the smart object platforms (including smart phones) but the deployment of the rest of the behaviour awareness components is dynamically decided. Common factors that influence the development, deployment decisions and optimization criteria are:

1) The Quality-of-Context (QoC) requirements of the applications,

2) The resource availability of the devices available,

3) The developmental complexity and manageability (of the code).

As the design and development of context-aware applications of BUTLER take into account the Quality-of-Context (QoC), the primary criteria for selecting a smart object platform over others or preferring a particular deployment configuration is its ability to match the QoC requirements of the application for the use case under study. For example, the platforms with integrated sensors such as tri-axial accelerometers and localization sensors are preferred as the detection of spontaneous activities heavily rely on these sensor inputs. Similarly, certain use cases such as fall detection require a high sampling rate (say 100 Hz) which in turn affects the decision to select the desired smart object platform that supports high sampling rate. Another major criteria in selecting a platform is the resource availability of the devices being used. Despite the fact that most of the smart object platforms described are battery powered and designed for sensing tasks, some pre-processing and inference components can be deployed on smart object platforms on ad hoc basis. For example, the step counting algorithm along with its pre-processing components (such as normalizer, low-pass filter) was deployed completely on SunSPOT sensors [19]. Whereas more complex components such as FFT analysers or learning algorithms for machine learning techniques, require much more resources (memory and processing power) and are always deployed in the SmartServers. Nevertheless, the deployment strategies are not clear-cut as our research shows that many performance and resource trade-offs exist with respect to the QoC requirements of the applications [20]. Also, the paper has demonstrated the benefits of a modular framework in achieving the desired Quality of Service (QoS) requirements without violating the dynamic resource constraints of the devices

Apart from application requirements and device availability, another influential decision factor is the manageability of the code. Most of the components developed as part of the behaviour awareness module (for example in SmartHealth Proof-of-Concept) are in Java. Hence platforms that support Java (or in which the already developed code can be easily ported) are preferred than others in order to mitigate the developmental efforts and to increase the manageability of the code. Nevertheless, platforms supporting C/C++ languages can be used as well when required.

4. Smart Mobile Technology

4.1. Goals and needs

This section aims at defining what will be achieved from a technical point of view concerning the Smart Mobile platform and at listing selected technologies, and why those technologies have been chosen.

The selected technologies must fit with the BUTLER requirements for the Smart Mobile devices.

4.1.1. Definitions

The SmartMobile platform is basically a set of applications that provides users with access to BUTLER services. Those applications run on any device targeted by the BUTLER project, such as mobile phones, tablets and TV sets. Furthermore, the SmartMobile platform will provide a framework for building BUTLER application, through a set of user interface elements, API and shared features.

On an architecture perspective, the SmartMobile platform represents a BUTLER user, who could be either authenticated or anonymous.

4.1.2. Architecture

The core of SmartMobile platform is a HTML5 web application that can be used in various environments according to the context and the use case:

- Embedded within a native application: while, as it will be described later, the core of the application is HTML5-based, and the SmartMobile platform will provide a native application. Indeed, such applications provided a lot of added value, both for developers (from BUTLER and/or from third-parties providing BUTLER services) and for end-users.
- Used as a regular web application through a web browser compliant with HTML5 and its related technologies.

Providing two different “versions” of the same platform makes the BUTLER services available on a wide variety of devices. Indeed, the hybrid layer will not be available on all devices, whereas HTML5 browser exists on all operating systems.

4.1.2.1. Mobile architecture, hybrid

The hybrid version of the SmartMobile platform aims at taking advantages of both worlds: webapps and native apps. By being a native app, it gives developers the possibility to access low level devices features, such as contacts, NFC readers, etc. It gives end users a good user-experience. Integrating a webapp in it gives developers the possibility to use advanced HTML5 features and overall gives the opportunity to develop a common UI to be used in all applications (being hybrid or full web).

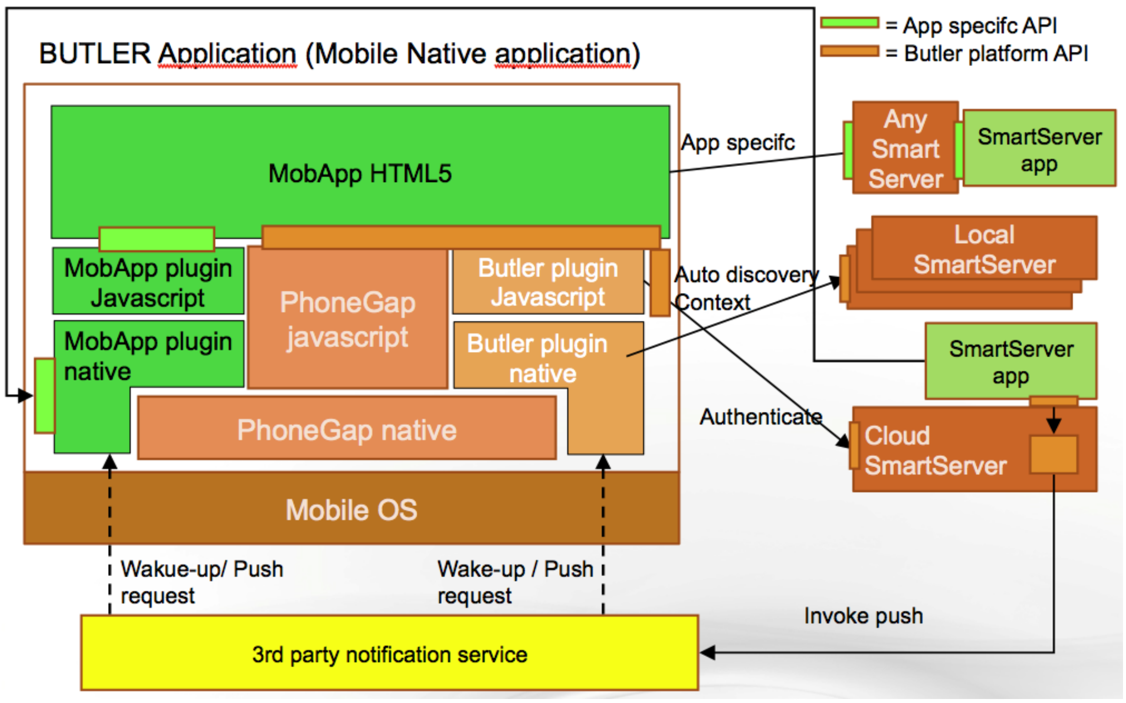


Figure 36 - Mobile native application

As shown on the Figure 36, the BUTLER SmartMobile application, in its hybrid version (it is called "hybrid" because of the usage of both technologies: web and native), is composed of various layers. The first of them is the mobile OS layer. **While the main targeted platform is Android, such architecture will allow BUTLER to deploy the Smart Mobile application on other operating systems**. On top of the operating system is placed the PhoneGap layer. PhoneGap is a framework that allows developers to build HTML5 native applications. Basically, it embeds a web browser within a native application. HTML5 document and related resources (such as images, CSS and JavaScript files and libraries) are then executed within this browser. Moreover, frameworks such as PhoneGap (see paragraph 4.2.4 for a list of similar products) also give access to low-level device features through JavaScript libraries.

So, on top of the PhoneGap layer is placed a set of components:

- The PhoneGap JavaScript library, giving access to mobile phone features;
- The BUTLER components:
  - BUTLER plugin JavaScript: this plugin will provide a set of common features to developers such as authentication, service discovery, etc. And any common service that will have to be shared by all applications. Moreover, it will act as a proxy to the BUTLER plugin native
  - BUTLER plugin native: in some cases, developers will have to write low-level components. For instance, in order to access an NFC reader on a mobile phone, a specific piece of native code will have to be developed. This component will be considered as a BUTLER native plugin and will provide access to the reader. In such a context, BUTLER services developers will access the reader through the BUTLER JavaScript plugin, which will in turn access the native component.

- The MobApp components: those components will act as the two BUTLER JavaScript components described above but will be dedicated to specific use cases, they will not be shared by all applications.
- Finally, on top of those layers will be placed the MobApp HTML5 application that will hold the application logic and the user interface.

It is worth noting that various layers, according to use cases, will be able to access remote servers (SmartServer or other remote services).

4.1.2.2. Mobile architecture, HTML5 only

The Figure 37 shows the BUTLER SmartMobile application in its HTML5-only version.

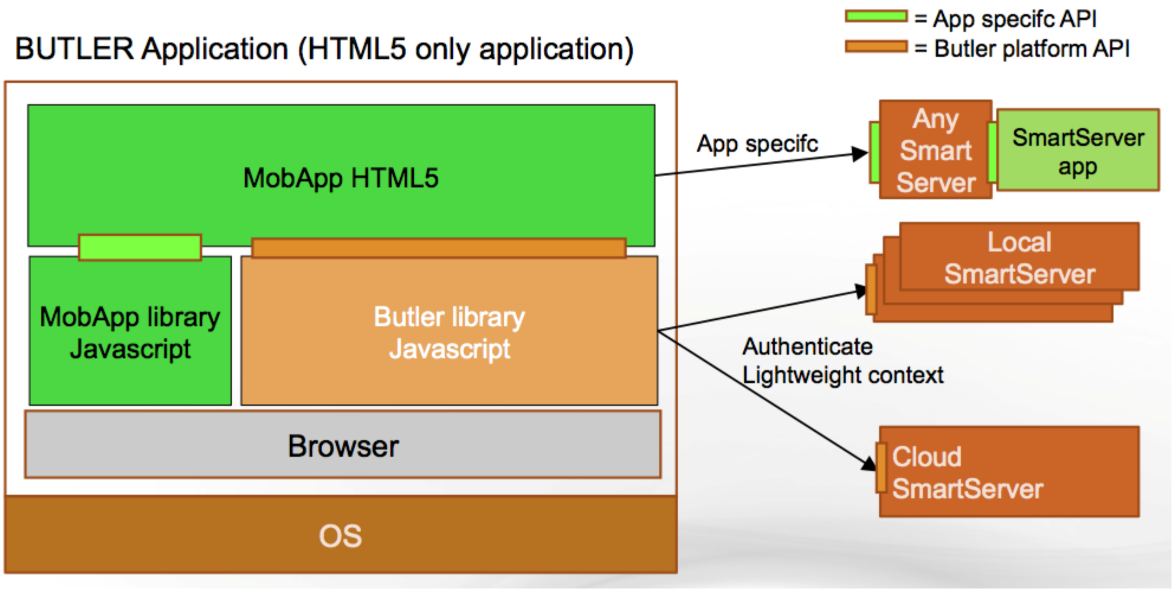


Figure 37 - Mobile application, HTML5 only

The main differences between both applications reside in the fact that the HTML5-only one won't be able to access low-level features (but those provided by HTML5) and as a consequence would provide a subset of features as compared to the native version.

4.1.2.3. Pros and cons of the hybrid platform

The following table shows pros and cons of the hybrid platform.

Pros	Cons
Platform independent: single codebase, which means that the BUTLER application will be available even on platform that are not directly targeted. However, a word of caution needs to be added: native component (low-level) will be platform dependent and even if PhoneGap is cross-platform, some differences exist.	Performance issues (initially, however performance, critical components can be implement with native language) and various solutions exist to overcome performance issues, such as single page application to get a fluid workflow between pages.

Web technologies: easier to use and learn than native languages. As previously described, the BUTLER application is an application framework that will be made available to other developers. Having a framework based on HTML5 will make it easier to use.	Interpreted code as opposed to compiled code. Interpreted code is always slower that compiled code.
Capable to function just as native application, which gives the end-user a good feeling.	Each platform has its own implementation: behaviour differences
Reduce development time & costs due to the fact that development will be done in HTML5 and related technologies.	
Open source (PhoneGap/Cordova)	
Core web app can be easily exported to be used as a standalone HTML5 web application	

While this table highlights various cons for the hybrid solution, their impact will be limited on the Smart Mobile application. For instance, in order to avoid performance issues, a technical approach called Single Page Application (SPA) will be used (see below).

4.2. Selection process

4.2.1. Architectures requirements and needs

The BUTLER Smart Mobile platform should cover the following needs:

- **Application framework**: as previously described, the Smart Mobile platform will be an application framework provided to developers so that they can build services on top of it.
- **UI consistency**: a common user interface will be provided for all services developed with the platform. Therefore, it will embed a set of well pre-defined user interface elements along with a common look and feel.
- **Responsive design**: user interface must be developed in order to feat various screen sizes, e.g. mobile phones, tablets, laptops, TV set…To achieve such need, the framework will provide developpers with a responsive design, which will adjust itself to the display.
- **Secure framework** (data privacy): data managed by BUTLER Smart Mobile application will have to be kept secured, either locally and during exchanges with servers.
- **Push**: the platform will provide a push mechanism so notifications can be triggered upon external events.
- **Service discovery**: a service discovery component will be required in order to determine which services are available at a specific location, within a specific context.

- **Context processing**: this component is required to process context-specific information and trigger related services and/or behaviours.
- **Settings**: this component manages local settings, mainly used by the end-users. It will contain services-specific data along with user management data (password, etc..
- **Authentication**: this common component handles authentication against BUTLER servers.
- **Access to low level features**: it is the capacity for any application to access low-level device features and components, such as an accelerometer or a NFC reader for instance.
- **Communication stack**: it represents the stack of protocols to access remote services.

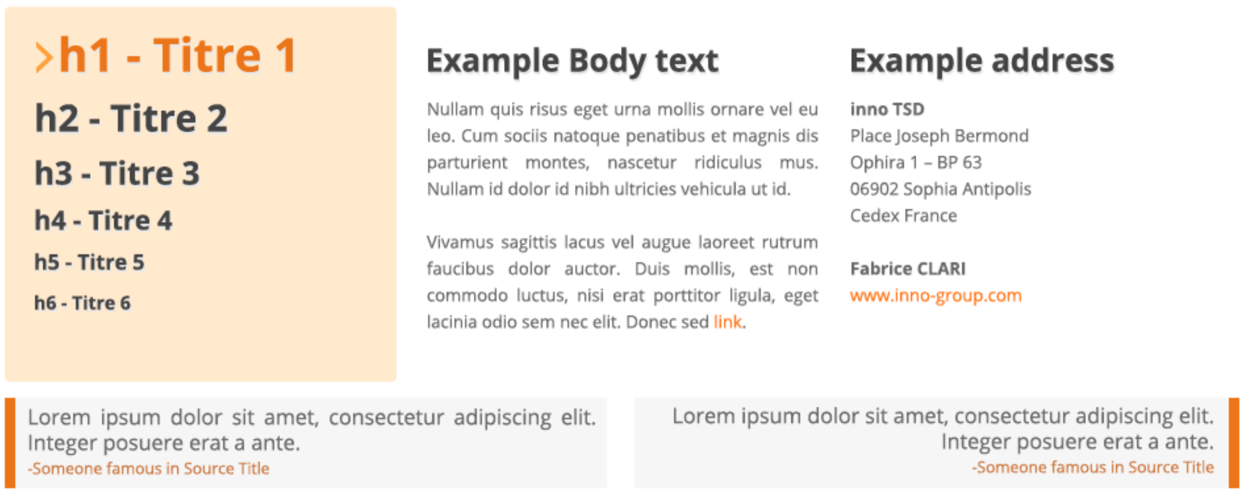
4.2.2. Mobile software web-apps requirements and needs

The mobile web application is meant to be used where the hybrid version is not available. Therefore, low-level components and services that are accessed by the hybrid version and which are not made available to the browser through HTML5 won't be used in the mobile version. The table below shows where features will be available.

Requirements/Needs	**Webapp, hybrid app or both?**
Application framework	Both the hybrid app and the web app will share the same HTML5 core application. So the HTML5 BUTLER framework will be available for both the web app and hybrid app.
UI consistency	Both types of applications will share the same user interface.
Secure framework	…
Push	Will be available in both scenarios, but concerning the web app, notifications will be possible only when the web app will be running as there is no OS specific component to wake up a web app.
Service discovery	Hybrid app only as it will rely on low-level protocols and components
Context processing	…
Settings	Both but the web app will provide access only to settings of services available.
Authentication	Both.
Access to low level features	Hybrid.
Communication stack	Both, but with limitation on the web app.

4.2.3. User interface

The SmartMobile platform will provide a common set of UI elements. Those elements have been designed based on the BUTLER look'n feel (the one used for the project's website and for documents). The following figure shows various UI elements already designed.



>h1 - Titre 1

h2 - Titre 2

h3 - Titre 3

h4 - Titre 4

h5 - Titre 5

h6 - Titre 6

Example Body text

Nullam quis risus eget urna mollis ornare vel eu leo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nullam id dolor id nibh ultricies vehicula ut id.

Vivamus sagittis lacus vel augue laoreet rutrum faucibus dolor auctor. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit. Donec sed link.

Example address

inno TSD
Place Joseph Bermond
Ophira 1 – BP 63
06902 Sophia Antipolis
Cedex France

Fabrice CLARI
www.inno-group.com

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere erat a ante.
-Someone famous in Source Title

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere erat a ante.
-Someone famous in Source Title

Figure 38 – UI: typography

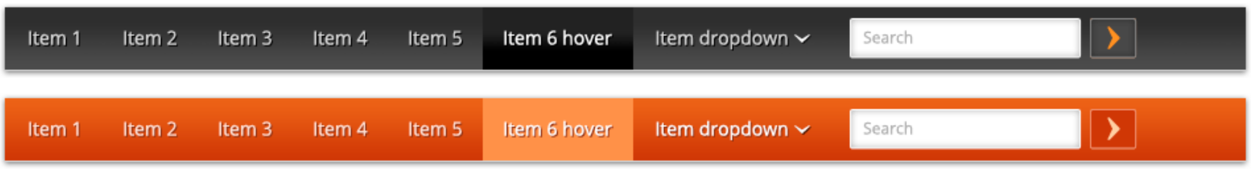
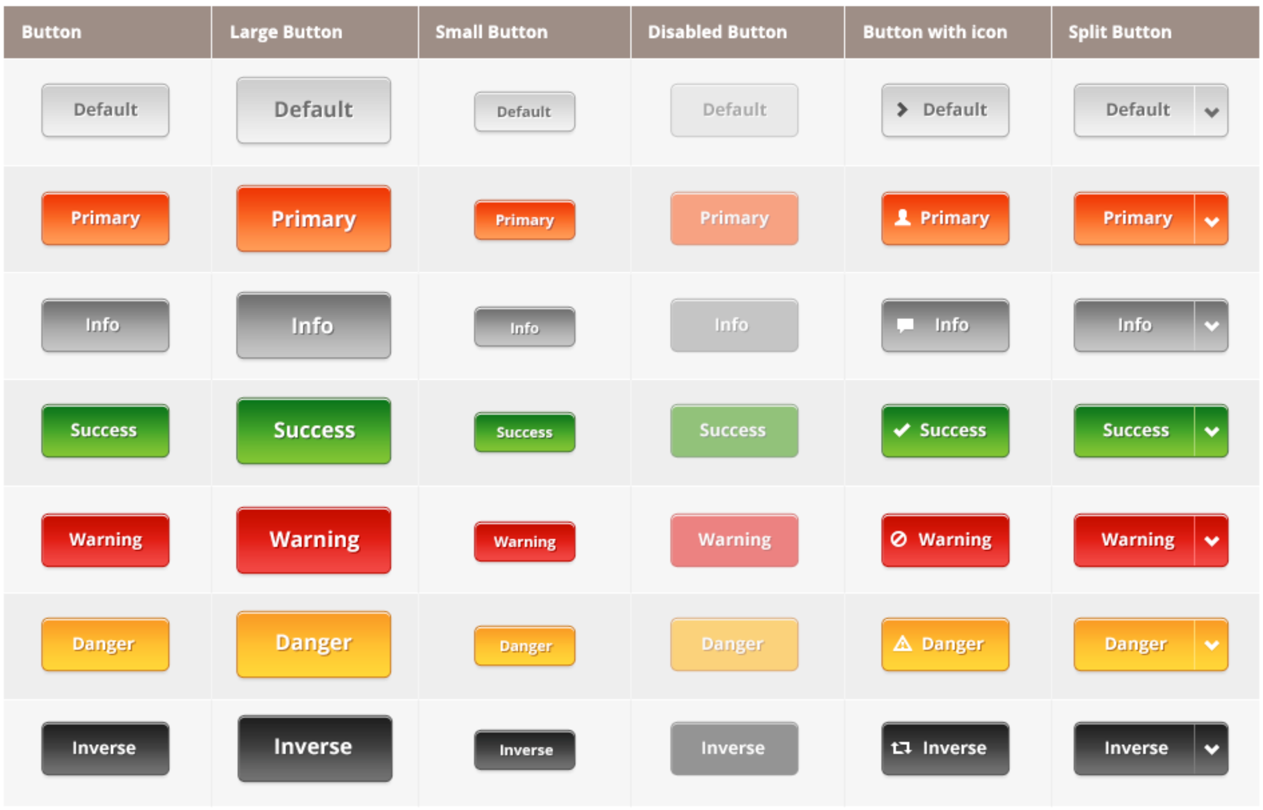


Figure 39 - UI: navigation bars



Button	Large Button	Small Button	Disabled Button	Button with icon	Split Button
Default	Default	Default	Default	Default	Default
Primary	Primary	Primary	Primary	Primary	Primary
Info	Info	Info	Info	Info	Info
Success	Success	Success	Success	Success	Success
Warning	Warning	Warning	Warning	Warning	Warning
Danger	Danger	Danger	Danger	Danger	Danger
Inverse	Inverse	Inverse	Inverse	Inverse	Inverse

Figure 40 - UI: buttons

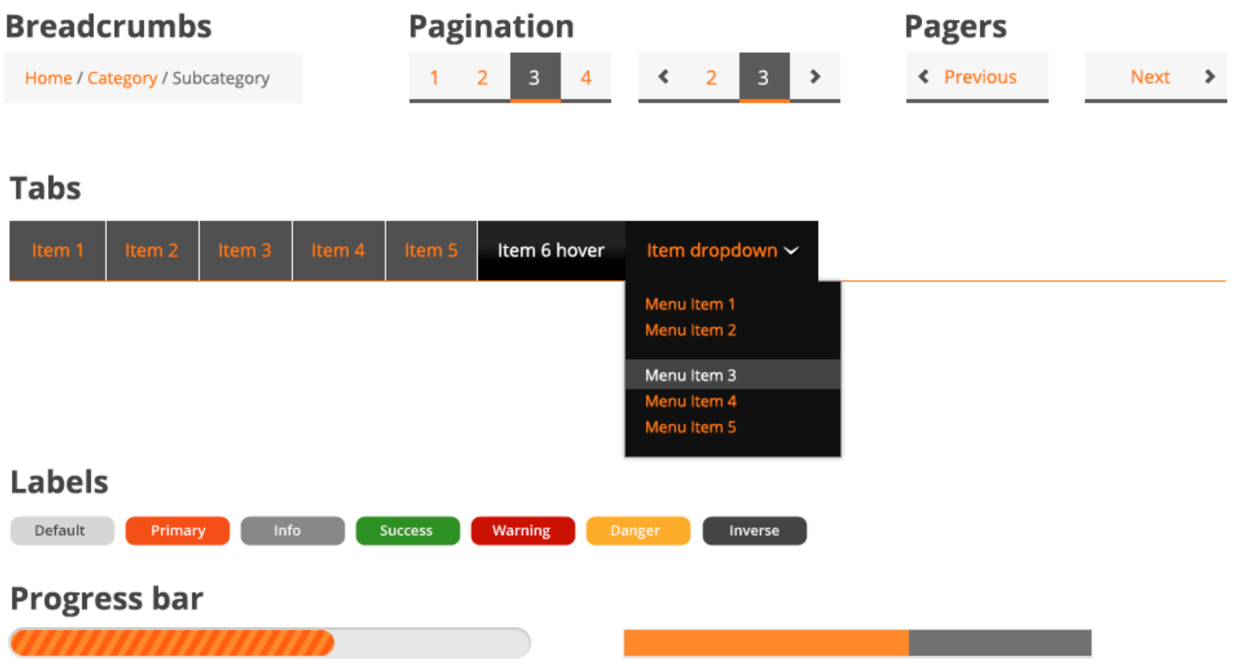


Figure 41 - UI: miscellaneous

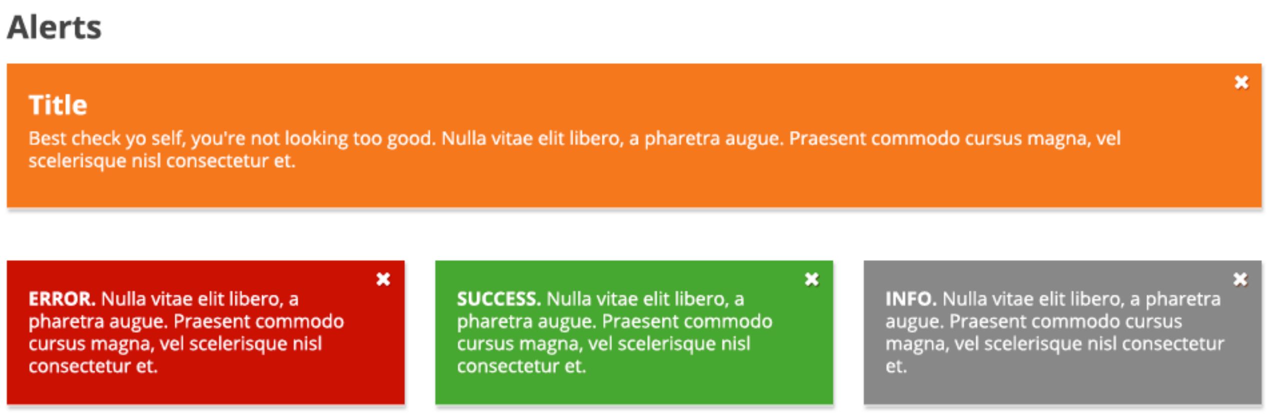

Alerts
Title
Best check yo self, you're not looking too good. Nulla vitae elit libero, a pharetra augue. Praesent commodo cursus magna, vel scelerisque nisl consectetur et.
ERROR. Nulla vitae elit libero, a pharetra augue. Praesent commodo cursus magna, vel scelerisque nisl consectetur et.
SUCCESS. Nulla vitae elit libero, a pharetra augue. Praesent commodo cursus magna, vel scelerisque nisl consectetur et.
INFO. Nulla vitae elit libero, a pharetra augue. Praesent commodo cursus magna, vel scelerisque nisl consectetur et.

Figure 42 - UI: alert boxes

4.2.4. Candidates (Hybrid App Frameworks)

Two competitors, PhoneGap and Appcelerator, mainly dominate the hybrid framework market:

- **PhoneGap** was initially developed by Nitobi and then acquired by Adobe. Now open source under Apache Software Foundation (Apache Cordova). Users: Adobe, Wikipedia, Facebook (previously, with forked Cordova), Microsoft, Zynga, BBC, LinkedIn
- **Appcelerator Titanium**, with a commercial licence. Customers: eBay, PayPal, NBCUniversal

Other solutions exist, for instance Red Foundry, RhoMobile and Clouch.

4.2.5. Chosen smart mobile technologies

4.2.5.1. Hybrid framework: PhoneGap

PhoneGap has been selected as the hybrid framework for the SmartMobile application.

Built on top of Apache Cordova, it allows the development of native mobile applications with HTML5, CSS4 and JavaScript, and offers a set of APIs to access native device functions (camera, sensors, compass, contacts, media, storage, geolocation).

Moreover, through a plug-in architecture, the framework can be easily extended with native code (for instance in order to access an NFC reader).

The Figure 43 shows a layered view of an hybrid architecture that highlights how PhoneGap works: the hybrid app is built with a set of resources (HTML, CSS, JS files) that are interpreted by a rendering engine that is run inside a native application.

PhoneGap supports 8 different mobile platforms (iOS, Android, Windows Phone, Blackberry etc.) and applications developed can be distributed on main app stores (app Store, Google Play…).

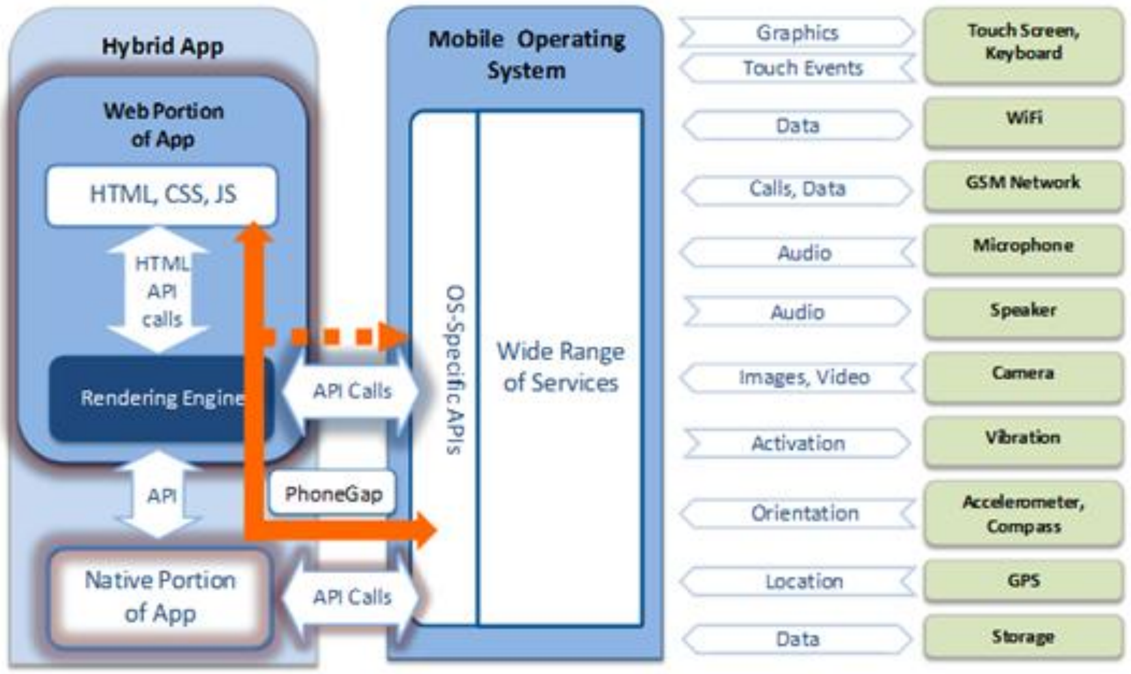


Figure 43: Hybrid architecture (source: IBM Worklight)

4.2.5.2. Web app framework: HTML5 and related technologies

HTML5 app will propose limited features (subset, no access to features based on native APIs).

HTML5 is the latest version of the HTML standard and provides a full stack of technologies:

- Markup language (plus CSS3 and JavaScript) to present content for the browser
- HTML5 API (in general, implementation specific differences exist)
  - Geo-location
  - Offline Web Applications
  - Web Storage & client side SQL
  - File API

  - New media APIs
  - Canvas support
  - Dynamic scriptable rendering
  - Web sockets
  - And more

CSS, in its third version, has been improved:

- Accessibility and flexibility
- Hardware acceleration
- Animations / effects
- Native look and feel

A CSS framework will be used in order to layout the UI. The selected CSS framework is Twitter Bootstrap.

In SmartMobile web applications, HTML5 will be extensively used. Also, a set of additional JavaScript libraries will be involved, such as:

- Backbone.js (or AngularJS): one of the drawbacks of the HTML5 webapp is the performance, especially when embedded in the native app (due to the fact that the hardware acceleration is not enabled in PhoneGap). To overcome performances issues, it has been decided to build applications as Simple Page Application. In such context, most of the pages (e.g. screens) are embedded in a single HTML5 page and displayed through a manipulation of the DOM tree. At the time of writing, Backbone.js and AngularJS have been shortlisted.
- DOM-manipulation: jQuery or Zepto.js

4.2.5.3. Communication with Smart Server

Communications with Smart Servers will be done through HTTP (or HTTPS for secured communications) and data will be exchanged by using the JSON format that facilitates data exchange in JavaScript.

4.2.5.4. Application structure

As previously described, the core of the application will be HTML5 based. Even if HTML is an easy to learn and use language, using a huge application on the client side can be tricky. For years, various solutions have been developed to improve applications' structure. Various design patterns have been developed and used in software development, the most famous framework being MVC, which stands Model View Controller.

In a nutshell, the MVC pattern allows developers to structure their application by clearly separating codes according to their responsibility. As its acronym says, Model View Controller, this pattern forces the separation of the view (the code being in charge of rendering data to the user), the model (the code holding the business logic) and the controller (the code acting as the glue).

4.2.5.5. Single Page Application paradigm

As previously explained, the main drawback of a hybrid application is the performance. It can produce native applications that do not look native. To overcome that potential issue, the BUTLER hybrid app will be built upon the Single Page Application (SPA) paradigm. Basically, this architecture allows developers to build web applications that offer a very good user-experience in terms of page loading and interactions between the end-users and the app.

Regular web applications are composed with various resources (such as HTML documents, images, Javascript and CSS files). When a page is loaded, the web browser loads all associated resources. Then, when the user wants to navigate from one page to another, the web browser sends a HTTP request to the server and downloads new resources (in case of a local app, it loads files from the local storage). That means that multiple server roundtrips are required. In the context of a hybrid application, this can slow drastically the user interface. To overcome such problems, the Single Page Application paradigm allows the development of web apps that embeds all HTML code in a single page and which dynamically loads other resources to the page as necessary, usually in response to user action.

Such architectures mean that the server is moved from the server to the client. Obviously, client/server communications still exist but are dedicated to data (content). Workflow is treated locally and pages are updated through JavaScript. The following picture gives an overview of the architecture.

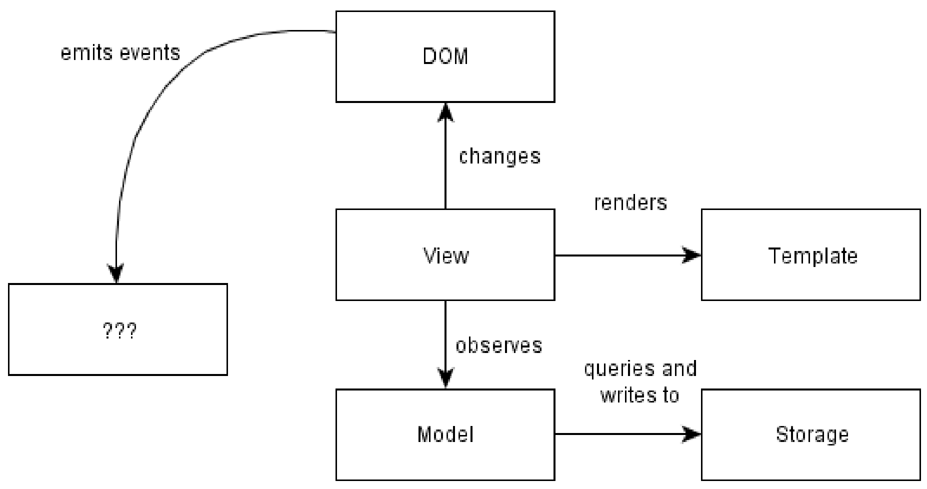


Figure 44 - SPA architecture (source: http://singlepageappbook.com/goal.html)

This pattern is also suitable for applications that reside only on the client-side and various libraries providing such behaviour exist, such as for example AngularJs and Backbone.js.

4.2.5.6. Testing and debugging

The testing stack will be composed of the following libraries:

- Behaviour Driven Development: Jasmine.js or Sinon.js
- Automation: PhantomJS

4.2.5.7. Notification framework (e.g. push)

In various use cases, the end-user will have to be notified by an external event, e.g. data will need to be pushed from a server to the device. Various solutions exist to achieve this notification goal. Worth noting that push will have to work in two scenarios: when a BUTLER app is running, and when no BUTLER app is running (wake up).

For the wake up scenario, most of the solutions are platform dependant (GCM on Android and APNS on iOS).

For in-app notifications, HTML5 server-sent event is a candidate. Solutions such as a periodic polling will be avoided. The HTML5 notification has a drawback: it can be used only when the web browser is running.

4.2.5.8. Service discovery

Service discovery protocols make possible the detection of services and devices that are available on a specific network. In the concept of BUTLER, such protocols are required in order to develop context-aware services.

Whereas a large number of protocols exist, such as Multicast DNS (Apple), SSDP (UPnP, Microsoft), SLP (IETF), a simple solution will be setup for the Smart Mobile. Indeed, the mobile platform needs to communicate with Smart Servers. Two kind of Smart Servers exist: local (e.g. located in a specific place, at home for instance) or remote (e.g. in the cloud, on the Internet).

The idea is to have a common FQDN for both servers. For instance smartserver.iot-BUTLER.eu. When connected from home, one will launch the SmartMobile app, which will in turn try to access a server. It will therefore send a DNS request to the local server that will return the IP address of the SmartServer located within the house. When outside, the device will use DNS located on the Internet that will only be aware of SmartServer that are located on the Internet.

4.2.5.9. Security layer

The BUTLER SmartMobile framework will adhere to the security principle as defined in the section 6.2. Moreover, HTTPS/SSL protocols will be used to encrypt communications between devices and SmartServers.

5. Smart Server Technology and BUTLER Infrastructure

5.1. Goals and needs

As defined earlier, in BUTLER a Smart Server is a set of software components that provide to BUTLER applications, Smart Objects/Gateways and Smart Mobile a set of functionalities through defined APIs.

In this section we would like to clarify what we want to achieve from a technical point of view at the Smart Server level in the BUTLER project and which technologies can be selected to be used and deployed on the BUTLER Smart Servers.

Selected technologies must fit with the requirements and the context-aware architecture designed for a heterogeneous network where different objects, devices and application components will be integrated to enable the BUTLER horizontal service experience.

Smart Servers will communicate both with Smart Objects/Gateways and Smart Mobiles, and will need to provide both general common functionalities and specific functionalities related to the specific services to be offered.

In general, the Smart Server platform defines a set of “Open Application Programming Interfaces” which application developers can use and “mesh-up” to integrate functionalities in their applications, similarly to what web developers can do with the web APIs offered by web companies like: Google, Yahoo!, Microsoft, Facebook etc.. These web APIs offer very different functionalities to application developers, ranging from Maps and positioning, to information Search, social data navigation, image storage and retrieval, personal data and many others. Web developers have at their disposal a rich set of functionalities offered using the same integration technologies: the classical web protocols, like HTTP/s and few others.

The aim for the BUTLER Smart Server platform task is to provide a “***component and API catalogue”*** and relative implementations to offer to BUTLER application developers the same “mesh-up” approach they are used to when programming on the Web.

Functionalities will be offered by several Smart Servers, each one offered through specific APIs endpoints that applications will be able to discover and use.

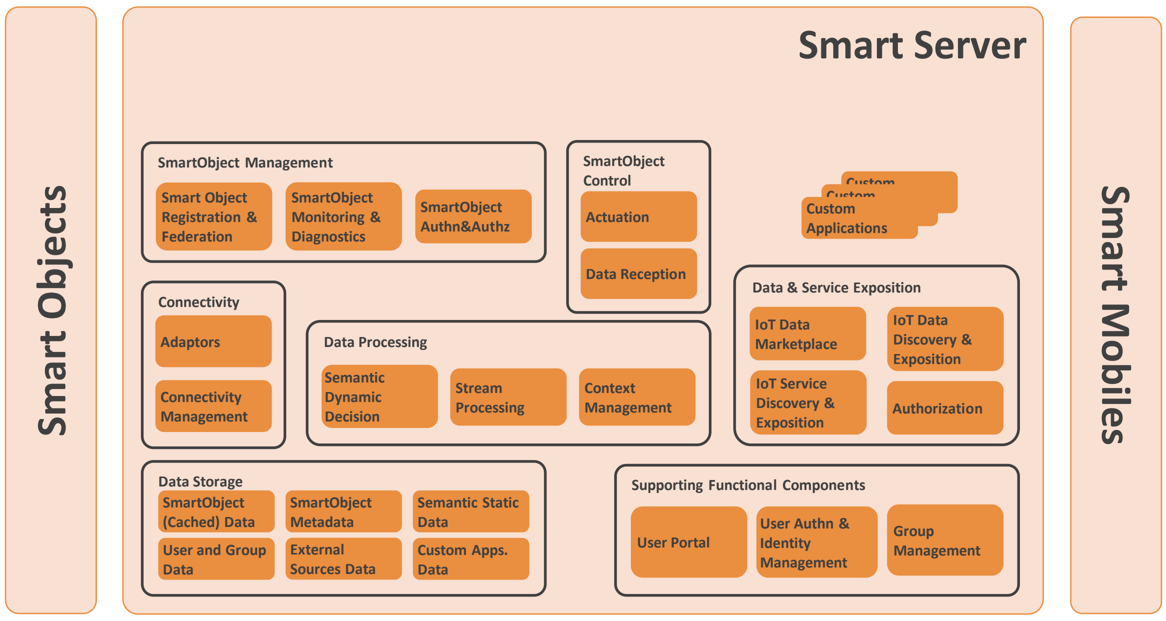


Figure 45 – Smart Server platform high level architecture

5.2. Selection process

In server development, several technologies can be used to offer functionalities to clients, so when looking at the BUTLER Smart Server platform, a collection of state-of-the-art server technologies have been analysed and listed from the project partners to highlight their expertise, best practices, usage in already existing software assets or past projects, and opportunities to re-use the same approach in BUTLER.

The selection process went through the following steps:

1. Selection of server technologies used/well known by partners
2. Selection of technologies already used by partners in their existing software assets
3. Classification of server technologies in different groups
4. Analysis of platform requirements in BUTLER based on analysis of functional and non functional requirements and analysis of potential enabling technologies.
5. Analysis of general platform requirements from the development viewpoint
6. Match of proposed/used technologies with requirements
7. Choice of a selected technology for BUTLER Smart Server satisfying the requirements

The general outcome of these steps is described in the following paragraphs.

5.2.1. Selection of server technologies

The following table summarizes the server technologies that project partners have listed as being well known, favorite or currently used in their existing software assets.

Partner	Software Asset or Technology expertise	Programming Language	API technology	Middleware / Operating System
TIL	Context Management Framework	Java	XML and JSON over HTTP, RESTfull	J2EE / JBoss Application Server
TIL	Social Advanced User Profile	Java	XML and JSON over HTTP, RESTfull	J2EE / JBoss Application Server
IHL	Expertise on Server-side integration	Java, C, C++, C#	XML over HTTP, web APIs	OSGi, Tomcat
ERC	IoT Processing Core	Java	XML over HTTP, Java API	Esper, S4
ERC	Energy Management functionality	Java	Java APIs	Esper, S4
ERC	Smart Shopping functionality	Java	Java APIs	Esper, S4
ERC	IoT Data Brokering Core for device data exposition	Java	XML over HTTP, Java API	Esper, S4
ERC	IoT Data Navigator	Java, HTML5	HTML front-end	
ISMB	Position Brokering Server	Java & Matlab	JSON over HTTP	OSGi, Tomcat
MAYA	Multi-room M2M framework	Java	XML over HTTP, Java API	Java
GTO	eGo server / service	C	PCSC	Windows, Linux

As it can be seen from the table, most partners have knowledge or are already using the Java programming language and, in terms of API exposure, choose to expose server functionalities by offering XML, JSON or Java APIs that are usually offered through the HTTP protocol.

In terms of middleware or execution platforms, several technologies are used, the most recurrent being J2EE/JBoss, Tomcat, OSGi and the Esper/S4 framework.

5.2.2. Criteria for selected technologies

The following criteria have been taken into consideration when choosing and selecting the technologies to be used for the BUTLER Smart Server.

Requirement/Need	Description
Integration of enabling technologies	Each Smart Server should be able to offer the underlying enabling technology through a uniform API.
System interoperability	Each Smart Server should be able to communicate with other platform components: other Smart Server, Smart Object/Gateway and Smart Mobile independently from the specific operating system, programming language or middleware they are using. In particular Smart Mobiles may be deployed on different devices and mobile operating systems like Android or iOS. Smart Server must guarantee interoperability with all the systems to be integrated in BUTLER.
Easy integration	Each API offered by a Smart Server should be easy to be integrated by application developers.
Common API approach	Each Smart Server should follow a common API approach and offer to developer endpoints that are consistent with other offered on different servers.
Data interoperability	Data that is obtained or sent to Smart Server by client should be easy to be interpreted and processed in an efficient way at the client-side. In particular data must be easily handled by Smart Mobile clients and the programming technologies used to develop applications.
Compatibility with Internet and Web infrastructure	Communication between Smart Server and other BUTLER platform components should be possible through the Internet, and in general, through web-enabled communication infrastructure components such as home or company proxies and firewalls. Usually, the protocol that is commonly available and widely supported is HTTP/S.
Performance and Stability	Smart Server should support several users and their devices, and integration with potentially millions of Smart Objects must be taken into account. Also, to be able to conduct field trials, the Smart Server must present an excellent grade of stability to provide continuous operation and service availability.

5.2.3. Chosen smart server technologies

Based on the selection of server technologies from partners, the general criteria for server integration listed above, and the requirements that were selected from the analysis of functional and non functional requirements and analysis of potential enabling technologies, the following main server technologies have been chosen for the first implementation of the BUTLER Smart Servers. The list will be extended as development of the Smart Server should require additional technologies to be included for specific application support, but the technologies listed here should guarantee maximum integration and compatibility between Smart Server and other platforms.

Smart Server Technology	Technology group	Available from partners	Why has been chosen?
Java language	Programming Language	All	System interoperability, well known language, richness of APIs and libraries
JSON	Data representation language	All, adaptation needed for some interfaces	System/data interoperability, easy integration (easy to be processed on client side)
Java application server technologies (J2EE, Tomcat, etc.)	Application Server middleware	TIL, ISMB, ERC, IHL	Stability, performance, interoperability
OSGi	Component middleware	ISMB, CEA, IHL	Easy integration, stability
MySQL	DBMS	All	Open source, stability, availability
HTTP	Application protocol	All	Interoperability, compatibility with existing infrastructures
HTML-5	Markup language for structuring and presenting content	All	Compatibility, powerful features, standard support
Esper/S4	Java complex event processing engine	ERC	Performance, stability
Linux	Server Operating System (different distributions)	All	Stability, open source
RESTfull API	API design	TIL, ISMB, ERC, …	Good common API design approach, easy and simple integration from developers

The specific version of the selected technologies will be verified and detailed as the implementation of the Smart Servers will proceed, maximizing the alignment between the different implementations from partners.

Some of the selected technologies are well known in the literature (e.g. Linux, PostgreSQL, etc.) and a description for them in this document is out of scope.

For other technologies a description has already been provided in previous sections of this document: for some of the other technologies a summary description is given in the following paragraphs.

5.3. Selected technologies in the BUTLER platform

In the following section, a brief description of some of the selected server-side technologies selected for BUTLER is given, along with the motivations why it’s suitable for the BUTLER project.

5.3.1. RESTful Interface design

REST (REpresentational State Transfer) [22] is an architectural style defined in 2000 by Dr Roy Fielding. [19] [20] [21]

The basic principles behind this architectural style are that the system must follow the client-server paradigm and that the architectural components interact via requests and responses and must be accessible through uniform interfaces. Another key notion is the resource concept: it is everything which is accessible, its state can be transferred, and can be univocally identified and addressed by a Uniform Resource Identifier (URI).

For example an object resource can be identified by the following URI:

http://example.com/objects/1234

where the path part ‘1234’ is the identifier of the object.

A REST system performs its functions trough the CRUD (Create, Read, Update, Delete) operations on resources, using some of the HTTP methods. In particular CRUD operations act on a resource in the following way:

- Create: it is mapped into the HTTP POST in order to create (or add) a new resource
- Read: it is mapped into the HTTP GET in order to access the resource
- Update: it is mapped into the HTTP PUT in order to modify the resource
- Delete: it is mapped into the HTTP DELETE in order to destroy the resource

These operations act on the same way on a single resource (e.g. http://example.com/objects/1234) as well as in a collection of resources (e.g. http://example.com/objects).

By reusing these verbs, as well as HTTP principles of authentication, caching and content negotiation, it is possible to build relatively simple APIs.

REST, as said, is a design style and not a strict standard and is therefore very flexible. First of all, the data format support: REST permits many different data formats (e.g. XML, JSON, et.) whereas SOAP only permits XML. This relates with the selection process for the Smart Mobile platform, where the hybrid Phonegap framework and JSON format have been selected. They are compatible with the REST architecture style for the whole system.

Moreover, REST system can be implemented using a layered structure and “code-on-demand” (extending functionality at runtime through applets or scripts) making it easy to add system functionalities without impacting the interfaces.

5.3.2. J2EE and Java-related application servers technologies

Some of the frameworks for Java application side development and deployment have been chosen at a general and system-specific level.

J2EE (Java 2 Platform, Enterprise Edition) is a platform designed for the server-side computing typical of large enterprises. J2EE simplifies application development and decreases the need for programming, and programmer training, by creating standardized, reusable modular components and by enabling the tier to handle many aspects of programming automatically.

J2EE includes a number of components added to the J2SE (Java Standard Edition) model, such as the following:

- Full support is included for Enterprise JavaBeans (EJB). EJB is a server-based technology for the delivery of program components in an enterprise environment.
- The Java servlet API (application programming interface)
- Java Server Pages (JSP) for dynamic Web-enabled data access and manipulation.

JAX-RS (or JSR-311), is a new JCP specification that provides a Java API for RESTful Web Services over the HTTP protocol.

JBoss [23] represents a widely adopted and deployed application server framework that implements the latest J2EE specifications. It has been proved to support high loads in production environments.

In addition to the generic application server environment provided by JBoss, the use of additional application frameworks has been appointed as a best practice for J2EE application development [23], since these provide a standardized and well-proven set of models and tools often supported by a community-driven development effort. From this approach, the following development frameworks have been adopted as part of the BUTLER Smart Server implementations:

Tomcat is a light weight yet powerful Java application server, which does not implement the full J2EE specifications but provides an efficient and powerful framework where Java web applications using technologies such as Java servlet can be deployed.

Hibernate is the reference Object-Relational Mapping (ORM) framework for the Java platform. It provides a uniform translation between the relational model and the object-oriented model, avoiding through that some potential issues related to relational databases, by providing abstract handlers to manage the database operations.

RESTEasy provides, within the JBoss server environment, the reference implementation for the Java API for RESTful web services, known as JAX-RS. RESTEasy is a portable implementation of this specification which can run in any Servlet container. Tighter integration with JBoss Application Server is also available to make the user experience nicer in that environment.

RESTEasy provides an annotation-based mechanism that simplifies the development of RESTful web services on Java applications.

Some of the advantages of the RESTEasy framework are:

- Portable to any app-server/Tomcat that runs on JDK 5 or higher
- Embeddable server implementation for junit testing
- Client framework to make writing HTTP clients easy (JAX-RS only define server bindings)

RESTEasy is distributed under the ASL 2.0 license.

5.3.3.JSON JavaScript Object Notation

JSON (JavaScript Object Notation) is an open standard designed for transmitting structured data over a network connection while remaining human-readable. This representation format is in fact text-based and is derived from the JavaScript scripting language for representing simple data structures and associative arrays, called objects.

JSON is not strictly tighten to JavaScript but instead is a language-independent format and parsers to read and write JSON data are available for many languages. JSON was originally specified by Douglas Crockford, and is now standardized by IETF in RFC 4627.

The JSON format is notably used by APIs all over the web and is a fast alternative to XML in Ajax requests.

Compared to the XML syntax, the JSON syntax is even more simplified. JSON syntax is built around key-values pairs which correspond to objects/structs in common programming languages. The language defines the following data types:

- *String* (Unicode string with backslash escaping): `"sample string"`
- *Number* (similar to C syntax except octal and hex formats): -47, -10.34, 2e+-7 , ...
- *Boolean* : true or false
- *Array* (ordered values separated by commas enclosed in square brackets):
  - `["name":"John", "name":"Andy", "name":"Billy"], ...`
- *Object* (comma separated values enclosed in curly brackets):
  - `{}` (an empy object)
  - `{"name":"John", "age":21}, ...`
- *null* : an empty/missing value
- *undefined*

The JSON language is a good choice for the serialization data language in BUTLER. Especially, it simplifies data processing on the client side where technologies such as JavaScript are being used to develop the client logic of BUTLER applications.

5.3.4. Description of Smart Server role

As illustrated in the next diagram, the Smart Server platform has been decomposed in several Functional Groups and “component roles”. Each component role can be implemented by a specific instance of the BUTLER Smart Server to make its functionality available horizontally to applications and other platforms entities.

Each Smart Server implementing a specific component role will offer its functionalities according to a uniform API design based on common technologies: RESTfull design, HTTP transport and JSON data serialization. By offering functionalities following a common design, the integration of applications is highly simplified and the horizontal approach for the BUTLER platform is enforced.

In the rest of this section a brief description of each component role is given; the specific API details for each component, the documentation on how to use the API and useful examples for developers will be detailed in future deliverables, namely the BUTLER deliverable D4.1 “BUTLER Smart Server Platform and Enabling Technologies”.

- **Resource Directory FG**:

The Resource directory is a Smart Server service that implements the listing and discovery of the resources in the BUTLER horizontal platform, for example API endpoints.

- **Data and Service Exposition FG:**

The Resource Exposition and **Context Exposition** will be implemented as Smart Server APIs to offer to applications and 3rd party systems an access to resources and context information that is available in the BUTLER platforms.

The Generic Notification Mechanism offers APIs to send notification messages to the Smart Mobile devices, leveraging on specific mobile operating system notification services.

The Data Marketplace offers functionalities to provide BUTLER data sources as paid information services.

- **Data Processing FG:**

The Stream Processing will be implemented as Smart Server entities that apply stream event processing software engines like Esper/S4 to handle large amounts of data and events coming from objects and context data.

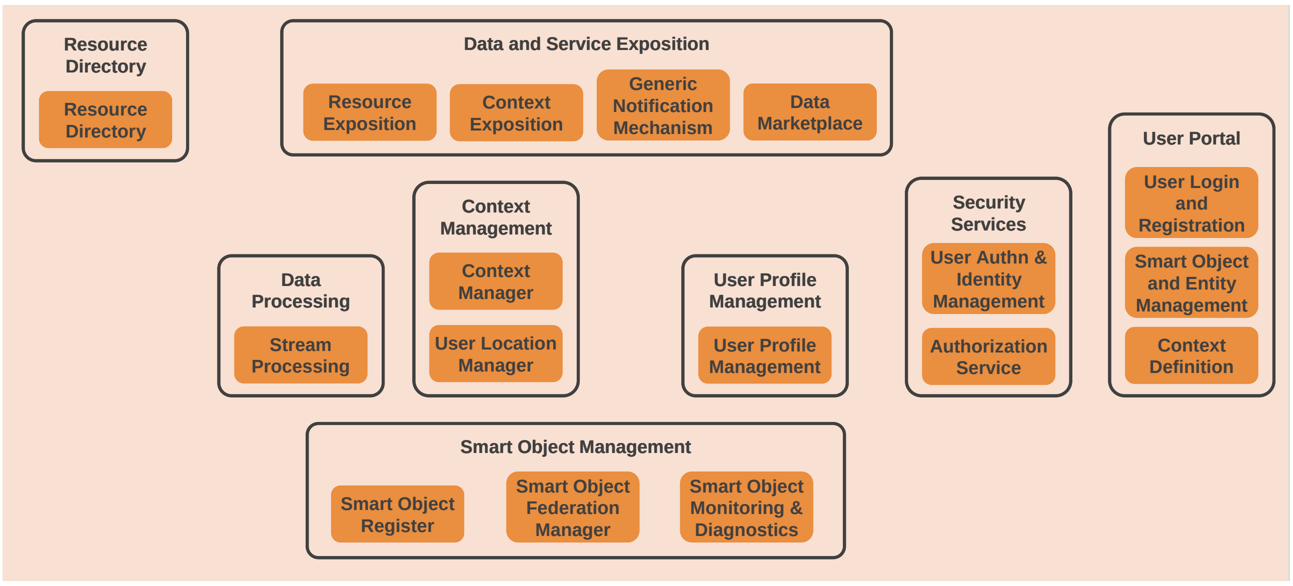


Figure 46 : Smart Server Components

- **Context Management FG:**

The Context Manager and the **User Location Manager** will be implemented as Smart Server instances and offer APIs to update, request, subscribe and get notified about the context and manage the location information of users and devices.

- **User Profile Management FG**:

The User Profile Management is a Smart Server component to store user profile and other user-related static data. The User registration APIs and user profile update APIs will be implemented.

- **Security Services FG**:

The User Authentication & Identity Management offer mechanisms to enforce the entity identification and enable access to other APIs and Smart Objects.

The **Authorization Service** will offer API endpoints related to the token-based authorization.

- **User Portal FG:**

These Smart Servers will provide final users web interfaces to perform operations related to their account (new user registration and login) and the association of other entities (objects) to their account and the definition of the context information. Three Smart Server instances are identified: **User Login and Registration**, **Smart Object and Entity Management** and **Context Definition.**

- **Smart Object Management FG:**

Finally the Smart Server in this functional group is implemented to guarantee the Smart Object lifecycle, in particular APIs will be offered to register and discover new objects (**Smart Object Register**), to associate objects to users (**Smart Object Federation Manager**) and to perform monitoring activities on objects (**Smart Object Monitoring & Diagnostics**).

5.4. Link between SmartServer and SmartObject

By using in BUTLER standard Web integration technologies like RESTfull, HTTP(s) and JSON, integration is highly simplified between Smart Server and Smart Object/Gateway. In particular interaction between Smart Object and Smart Server is bi-directional and both platforms will be able to invoke APIs through defined endpoints. For example a Smart Server component where a server side application may be run, could contact using a RESTfull call a Smart Object Gateway HTTP endpoint to request available objects, and then, request data from them. Data returned by the Smart Object will be represented preferably using the JSON format.

Similarly, a Smart Object Gateway can contact a Smart Server API endpoint to perform some common functionality, like verifying the access credentials to object data in relation to a specific user.

The use of the HTTP protocol to perform API invocation and return related data is compatible with currently deployed network infrastructures, and also brings the advantages of HTTP-related encryption, or HTTPS, which is already available in all web and application servers and well supported by intermediate network nodes and clients.

Depending on the current deployment solution, the communication between a Smart Server and Smart Object/Gateway could be:

- if the Smart Server is a Local Smart Server: through a local network (for example WiFi or local LAN),
- if the Smart Server is deployed in the BUTLER Cloud: through the public Internet.

In the latter case, authentication and encryption must be applied to ensure that the communication between Smart Server and Smart Objects is protected, so specific functionalities both on Smart Object Gateway and Smart Server will be activated.

5.5. Link between SmartServer and SmartMobile

Smart Server and Smart Mobile will be integrated together using standard Web integration technologies: RESTfull, HTTP(s) and JSON. Applications on the Smart Mobile will be able to invoke APIs endpoints made available by a Local or Remote (Cloud) Smart Server. Connection to a local Smart Server could be delivered through WiFi, while a connection to a Remote (Cloud) Smart Server will be made using the public internet, thanks to the mobile device data connection (WiFi or 3/4G).

In cases where services will require higher data delivery performances, protocols other than HTTP, for example streaming protocols, could be used to exchange data between a Smart Mobile and a local or remote Smart Server, for example to play multimedia content from a Smart Server to a Smart Mobile.

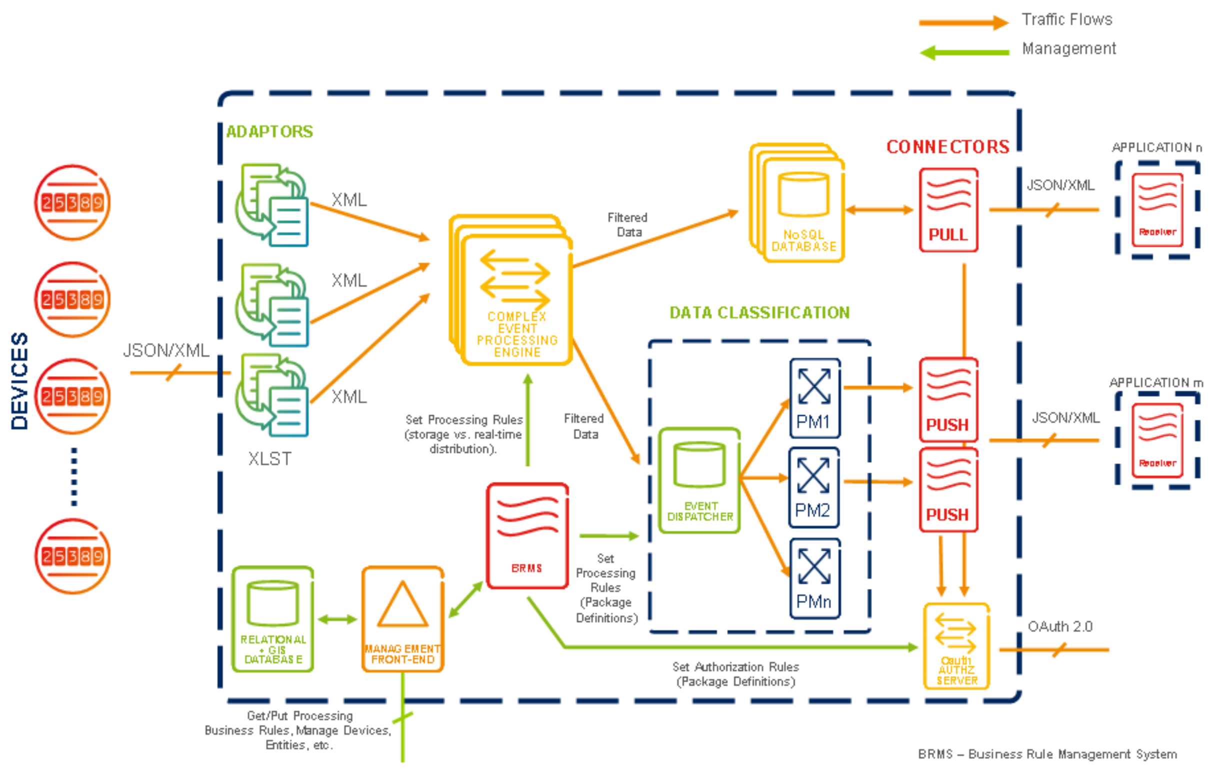
Differently from the interaction with Smart Objects, the Smart Mobile will not directly offer an APIs endpoint toward the Smart Server, but instead a Notification mechanism is used for the Smart Server to trigger and request activity on the Smart Mobile. For example, a server-side component that requires awaking applications or receiving data from a Smart Mobile, will send a notification using the BUTLER Notification API endpoint. The BUTLER Notification component will deliver this notification message to the mobile device properly, using a 3rd party notification service that is specific to the mobile operating system (Google Cloud Messaging on Android or Apple Push Notification Service on iOS, for instance). When receiving a notification, the BUTLER Smart Mobile will then awake the interested BUTLER applications that could then contact the Smart Server to deliver the requested data.

6. Integration of the BUTLER smart platforms in the network

In this section, the Smart Server platform is detailed as it is a central component to achieve the unified BUTLER platform. It provides resources, functionalities and a dedicated architecture to realize end-to-end communication between several BUTLER devices. It contains also several new technological bricks available to build new services or applications for the final user using environmental measurements. An end-to-end security framework is deeply detailed as it is an essential feature to secure the data and the citizen and to make the acceptance of the technology proposed by such a system. Finally, a horizontal example using the whole platform is described to illustrate how the BUTLER platform can be used in a pragmatic way.

6.1. Smart Server Platform

The Smart Server platform comprises a set of components and interfaces providing services that are offered to the BUTLER community through the remote components. The Smart Server Platform provides the required functionality to be used by other BUTLER components and entities and third parties. It provides functionalities such as real time classification of data flows generated by IoT devices, storage of relevant data defined by rules, geodetic database to improve the management and discovery of device locations, data push and pull interfaces towards domain-specific applications developed by third parties, and access control based on OAuth 2.0. Below, a high level architecture of the platform solution can be seen.



Traffic Flows
Management
ADAPTORS
XML
XML
XML
XLST
DEVICES
JSON/XML
COMPLEX EVENT PROCESSING ENGINE
Filtered Data
NoSQL DATABASE
CONNECTORS
PULL
APPLICATION n
JSON/XML
Receiver
DATA CLASSIFICATION
Set Processing Rules (storage vs. real-time distribution).
Filtered Data
EVENT DISPATCHER
PM1
PM2
PMn
PUSH
PUSH
APPLICATION m
JSON/XML
Receiver
BRMS
Set Processing Rules (Package Definitions)
RELATIONAL + GIS DATABASE
MANAGEMENT FRONT-END
Set Authorization Rules (Package Definitions)
OAuth AUTHZ SERVER
OAuth 2.0
Get/Put Processing Business Rules, Manage Devices, Entities, etc.
BRMS – Business Rule Management System

Figure 47: Smart Server high-level description

The set of horizontal back-end functional components includes:

- **Geodetic database** is a relational database with GIS extensions for the storage of the information describing the location of the sensors and the different virtual entities that contains said sensors;
- **Complex Event Processing** (CEP) engine and the associated event processing functionality define how the CEP engine processes the real-time information, how incoming data is either distributed or stored, and how it is organized into packages ready to be offered in the market place and subsequently accessed by third-parties;
- **Adaptors** cope with data coming from sensors. Their purpose is to carry out protocol adaptation and data model translation into the internal event data model used by the CEP engine;
- **Business Rule Management System** (BRMS) is integrated with the CEP engine and connected to the management interface to enable users to organize real-time data flows into packages. It is also used to define the business rules that will be used to define the real time data flows that will be stored in the noSQL database.
- **No-SQL Database** caches relevant information from devices to enable subsequent delivery to customers;
- Web Services-based connectors delivers information generated by the platform to external applications (either in push or pull mode). Said interfaces will be integrated with OAuth 2.0 for enabling access control;
- An OAuth 2.0 Authorization Server.

Related to the integration interfaces:

- A **management front-end** enables interaction with users so that they can set the proper rules that define the way data flows are handled. It connects with the BRMS (Drools), providing access to the geodetical database to manage virtual entities and devices;

6.2. Security framework

The Privacy & security functional specifications shall apply for the Horizontal Scenario. As stated in "**D2.1 Requirement, Specification and Security Technologies for IoI Context-Aware networks**", the application layer security is particularly adapted "*when application provider cannot rely on single network but on heterogeneous environment either at network level or application level (D2.1 / Application Layer)".*

Each application layers security technologies shall be studied in the context of the horizontal scenario and take into account the "Privacy Principles" stated in D2.1 / Privacy Principles.

Privacy – Principles:

The BUTLER Privacy Principles are summarized as follow:

Transparency of usage of the data.

User – *data subject* in the European Union (EU) parlance - shall give explicit consent of usage of data.

Collected Data shall be adequate, relevant and not excessive

The data shall be collected on “need to know” principle. This principle is also known as “Data Minimization”. The principle also helps to setup the user contract, to fulfill the data storage regulation and enhance the “Trust” paradigm.

Collector shall use data for explicit purpose.

Data shall be collected for legitimate reasons and shall be deleted (or anonymized) as soon as data is no longer relevant.

Collector shall protect data at communication level.

The Integrity of the information is important because modification of received information could have serious consequences for the overall system availability. User has accepted to disclose information to a specific system, not all the systems. The required level of protection depends on the data to be protected according to the cost of the protection and the consequence of data disclosure to unauthorized systems.

Collector shall protect collected data at data storage

User has accepted to disclose information to a specific system, not all the systems. It also could be mandatory to get infrastructure certification. The required level of protection depends on the data to be protected according to the cost of the protection and the consequence of data disclosure to unauthorized systems. For example, user financial information can be used to perform automatic malicious billing. Such data shall be carefully protected. Security keys at device side and server side are very exposed and shall be properly protected against hardware attacks.

Collector shall allow user to access / remove Personal Data.

Personal Data may be considered as a property of the user. User shall be able to verify correctness of the data and ask – if necessary – corrections. Dynamic Personal Data – for instance home electricity consumption – shall also be available to the user for consultation. For static user identity, this principle is simply the application of current European regulations according access to user profile.

Regulation

For Dynamic Personal data, for now, no regulation applies. Anyway, BUTLER shall anticipate that the incoming regulations will reinforce the control of the user with respects to personal data. This directive shall be endorsed by EU countries. EU is working on Data Protection Framework regulation based on the Data Protection Directive 95/46/EC. At EU level, regulation is stricter than a directive (except for aspect related to police or criminal investigation), this means that EU countries will have obligation to follow the new incoming regulation.

Applying the Privacy Principles in the context of horizontal scenario:

Horizontal Scenarios involve many stakeholders with different business models. For instance, to authenticate the user, Service Provider may rely on Mobile SIM card – this implies such Service Provider having technical and commercial contract with Mobile Operator, other Service Provider does not require user authentication but only user anonymous context – for instance, the “user is walking in street near a bookshop and he/she is known as book addict". In this example, a Service Provider (for instance a bookshop) does not care about user identity but only on user specific context. This example shows also that the user personal data – here its location and the fact he/she is book addicted – is disclosed to the bookshop. This disclosure shall fulfill the principle "**Transparency of usage of the data**".

In horizontal scenario, there are many entities that provide data – for instance the location of the user in the above example – and entities that consume data – for instance the bookshop that is interested by a user specific context even if this context is anonymous.

6.2.1. Application Level Security roles

The abstraction of the stakeholders highlights the concepts of **user, resource provider** and **resource consumer**. Users shall allow resource consumer accessing data from data provider**.** User shall be able to control the usage of its personal data which is also referred as protected resource. In horizontal scenario involving different stakeholders, user shall have a single point to manage the authorization to access its personal data which are distributed among different Resource Providers; this single point is called the **authorization server.**

Security Roles - definition

Role	**Definition**
User	User entity granting access to a resource. Generally, the user refers to a person, but can also refer to an application. The user shall be authorized to access the resource by the owner of the resource.
Resource Provider	Entity providing (and optionally updating) a resource. The Resource Provider shall check an *access-token* to provide/update the resource. Resource Metadata shall be registered in the Authorization Server (AS)
Resource Consumer	Client application getting and consuming resource on behalf of a user. Such user must be authorized to access the resource.
Authorization Server	The Authorization Server plays the role of Resource Directory. It implements access controls management. The Authorization Server authenticates the user and authorizes resource-consumer getting resource by issuing a resource related *access-token.* Optionally, it may delegate the user authentication to an External Authentication Server
(optional) ***Authentication Server***	*This optional role can be used by Authorization Server to rely on authentication protocol not natively implemented in the Authorization Server. It means that the Authorization Server and Authentication Server shall federate some user identities.*

Security Roles - High Level Interactions:

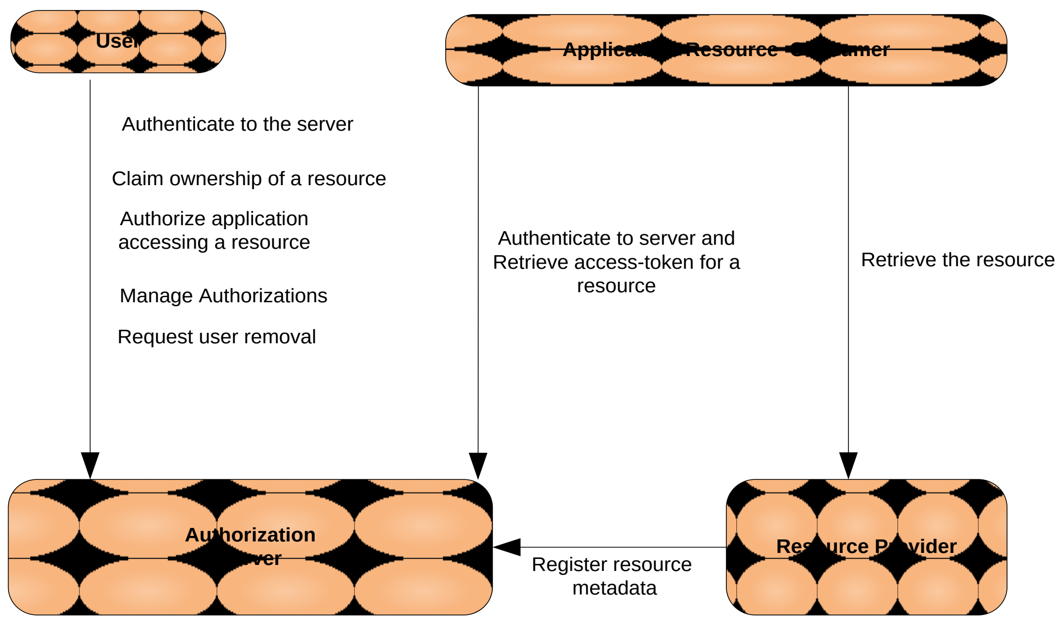

User
Application – Resource Consumer
Authenticate to the server
Claim ownership of a resource
Authorize application accessing a resource
Manage Authorizations
Request user removal
Authenticate to server and Retrieve access-token for a resource
Retrieve the resource
Authorization Server
Register resource metadata
Resource Provider

Figure 48 Security Roles High Level Interactions.

The above figure presents the interactions between Security Roles.

A typical use case can be represented as follow:

1. The Resource Provider registers a resource metadata to the Authorization Server
2. The User requires a service to an application – Resource Consumer.
3. The Application connects to the Authorization Server and retrieves a resource access-token on behalf of an authenticated user.
4. The application retrieves the resource and provides the related service.

6.2.2. User Interaction - Generic Access to Resource

The following figure presents the overall message flow to access a protected resource. The access shall be authorized by the user.

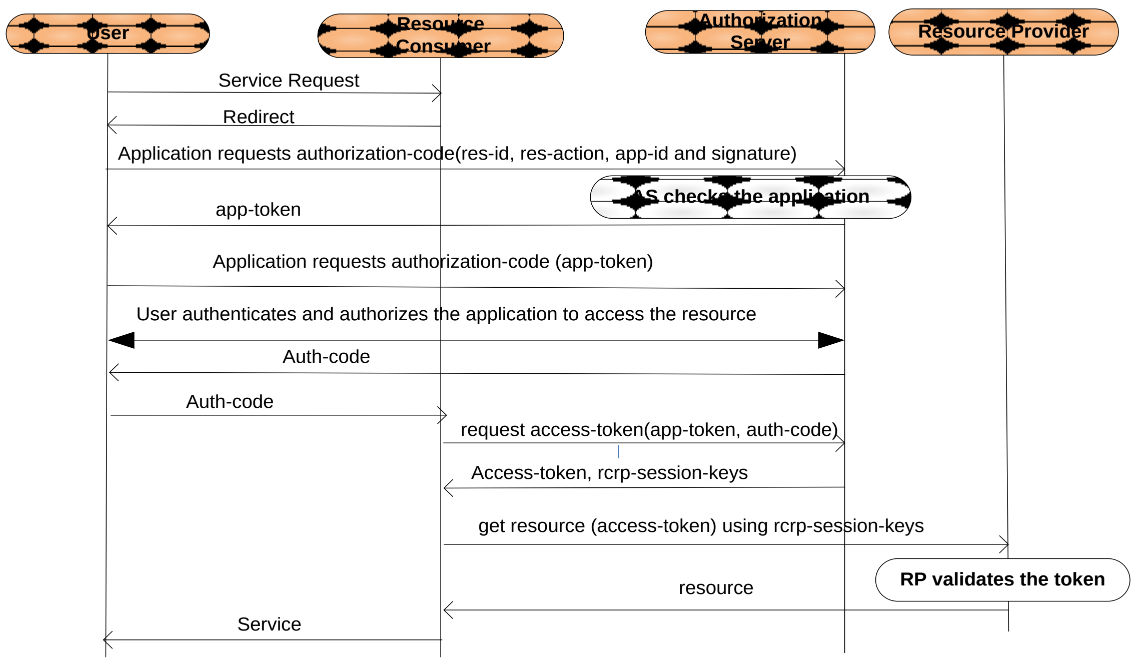

User
Resource Consumer
Authorization Server
Resource Provider
Service Request
Redirect
Application requests authorization-code(res-id, res-action, app-id and signature)
AS checks the application
app-token
Application requests authorization-code (app-token)
User authenticates and authorizes the application to access the resource
Auth-code
Auth-code
request access-token(app-token, auth-code)
Access-token, rcrp-session-keys
get resource (access-token) using rcrp-session-keys
RP validates the token
resource
Service

Figure 49: Generic Access to Resource

The generic message flow is based on protocol OAuth 2.0.

The message flow is the following:

1. Using his user-agent, the user requests a service to a Service Provider. Here the Service Provider plays the role of Resource Consumer.
2. Through the user-agent, the application requests an *authorization-code* to access the resource. The Authorization Server checks the application and returns an *application-token*.
3. With the *application-token,* the application requests again an *authorization-code.*
4. The user authenticates to the Authorization Server and authorizes the application to access the required resource.
5. On behalf of the user – through the *authorization code* - the application requests the *access-token.*
6. The authorization server generates the *access-token* and randomly generated rcrp-session-keys.
7. Using the *access-token* and the *rcrp-session-keys,* the application requests the resource to the Resource Provider.
8. The Resource Provider checks the *access-token* and provides the resource.
9. The application consumes the resource and provides the service.

NOTE: all the communications between the Resource Consumer and the Authorization Server, and also the communications between the User Agent and the Authorization Server rely on SSL protocol with Server Authentication. The Client Authentication is already implemented by the protocol OAUTH 2.0; therefore the communication between the Resource Consumer and the Authorization Server does not require SSL Client Authentication.

6.2.3. Authorization Server

The Authorization Server generates, for a Resource Consumer, an access-token to a resource on behalf of an authenticated user. The access-token encompasses some attributes that shall be checked by the Resource Consumer.

The Authorization Server implements the role of Resource Directory. Each Resource Provider shall expose one (or more) resources to the external world. A resource is described by a resource metadata. The resource metadata consists of:

Resource Metadata:

Semantic	Semantic description of the resource. For instance: localization, temperature…
Resource Identifier	The URL of the resource exposed by the Resource Provider.
Allowed Actions	The action a resource "consumer" can perform. The Resource Provider shall list here the actions that can be performed on the resource. For instance, if RP is only a sensor, the allowed action is "GET"; if it is an actuator, the allowed action is "SET". More sophisticated Resource Provider may expose other actions on resource.
Key Material	The key material required to build the access-token. The KM defines the security algorithms and the data needed to build the security keys.

Authorization Server – User:

The authorization server does not register any user profile.

A user is represented only as follow.

User Identifier	The unique user identifier.
User Credential	The credential to be used for user authentication purpose. It is generally a *password.*

Authorization Server – Access Control List:

The Authorization Server implements Access Control List (ACL) for each resource. The resource owner is allowed to manage the ACL. The Authorization Server encompasses the following table.

Resource Identifier	The identifier of the resource metadata
User Identifier	The unique user identifier
Access Rights	Action1, authorized/forbidden
	…
	Action-N, authorized/forbidden
Is Resource Owner?	If true, the user is the owner of the Resource. At first registration of a new resource metadata, the resource metadata is not associated to any user. If the owner is not set, any user can claim ownership of the resource. The resource owner can perform any actions to the resource and is allowed to manage the related Access Control List.

6.2.4. Resource Provider

A Resource Provider is an entity which exposes Resource to the external world. A resource is described by resource metadata. The resource metadata shall be registered in Authorization Server that plays the role of Resource Directory. Smart Objects, Smart Object Gateways, Smart Servers can implement the role of Resource Provider.

A Resource Provider shall check the *access-token* before providing the resource to the Resource Consumer. A Resource Provider is addressable by the Resource Consumer.

Smart Object as Resource Provider:

A Smart Object shall expose one (or more) Resource to the Authorization Server. A resource is represented by Resource Metadata. The provisioning of the metadata(s) must be available at object manufacturing and personalization time.

Smart Object Resource Metadata:

At manufacturing time, the object is provided with a list of resources.

Semantic	Semantic description of the resource. For instance: localization, temperature…
Resource Identifier	The identifier of the exposed resource. The Resource Identifier encompasses the object identifier. Each resource of object shall have the same object identifier.
Allowed actions:	The action a resource "consumer" can perform.
Object Key Material	The Key Material information is used by the Authorization Server to retrieve the Object Security keys required to build the *access-token* and by the Personalization Center to retrieve the *key-encryption-key* required to encrypt the *access-token-signature-key* and the *access-token-encryption-key.*

Object Security Keys:

For the Smart Object, the default Security Key mechanism is based on symmetric cryptography. This choice allows simple and low cost security implementation avoiding PKI certificate management.

A particular object implementation may rely on other mechanism. The mechanism shall be supported by the Authorization Server and declared though the Key-Material.

At manufacturing time, the object is initialized with

Key-encryption-key	Factory-key-encryption-key = function3(key-material, FACTORY-KEY-ENC-MASTER-KEY)

At personalization time, using the Key-encryption-key, the object is personalized with.

Access-token-signature-key	Access-token-signature-key = function1(key-material, AUTHORIZATION-SERVER/PERSO-CENTER-ACCESS-TOKEN-SIGN-MASTER-KEY)
Access-token-encryption-key	Access-token-encryption-key= function2(key-material, AUTHORIZATION-SERVER/PERSO-CENTER-ACCESS-TOKEN-ENC-MASTER-KEY)

Smart Server as Resource Provider:

Any Smart Server may expose resource to the Authorization Server. For instance, the Context Management shall expose some Context Resources which are available to applications – Resource Consumers - as other resources of the system. As other resources, using the Authorization Server, the user shall allow applications to access the Smart Server resources.

Smart Server Resource Metadata:

At configuration of a Smart Server, the Server implements some resources that shall be registered at the Authorization Server.

Semantic	Semantic description of the resource. For instance: context, user profile…
Resource Identifier	The identifier of the exposed resource
Allowed actions:	The action a resource "consumer" can perform.
Server Key Material	The Key Material information is used by the Authorization Server to retrieve the Server Security keys required to build the *access-token.*

Smart Server Security Keys:

By default, the Key Material is the Smart Server encryption certificate. The Authorization Server will encrypt the *access-token* with the Smart *Server (the Resource Provider) public key* and sign the *access-token* with its own private key. To avoid passing the Authorization Server signature certificate in the *access-token,* the Smart Server shall be provided with such certificate out-of-band.

Smart Mobile as Resource Provider:

The Smart Mobile does not provide any API to external world; therefore it does not provide any resource to external world. Anyway, a mobile location can be sent to a Localization Smart Server which exposes a Location Resource.

On user consent, a mobile application can populate a Localization Smart Server for further purposes.

6.2.5.Resource Consumer

A resource consumer is a client application getting and consuming resource on behalf of a user.

A Resource Consumer requests the Authorization Server for *access-token* related to the required resource. On completion, it connects the Resource Provider with the *access-token*.

The Resource Consumer has the responsibility to take into account the "Collector …" Privacy Principles. See Privacy Principles / Implementation matrix.

6.2.6. Claiming Ownership of an object

Generally, after the personalization process, an object is not owned by a user. In case, it is done, this use case is not applicable and the claiming of the ownership is supported at registration of the resource to the Authorization Server.

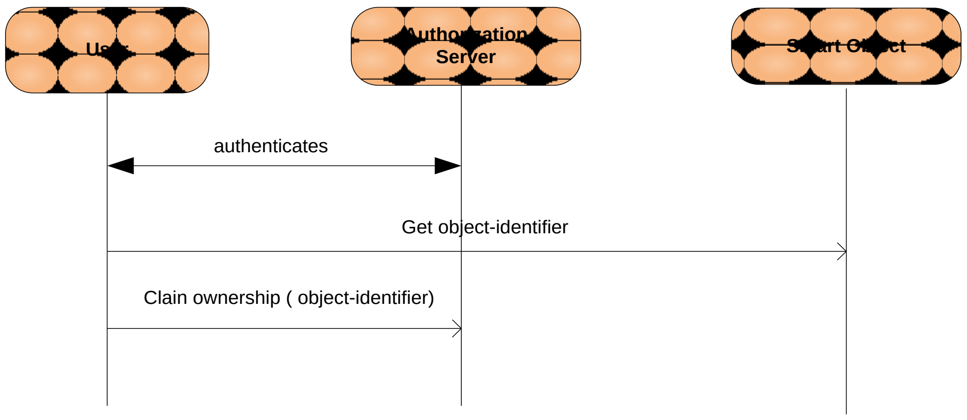


Figure 50: Claiming Ownership of an object

Each object shall provide a method for a user to retrieve its identifier. Once the resources of the object are registered in the Authorization Server, the user shall claim ownership of the object.

6.2.7. Building Access Token.

Access-token shall be provided to a Resource Provider to retrieve a specific resource. The resource is identified by a resource-identifier.

The OAUTH 2.0 protocol does not specify how to build and checks the access token.

For BUTLER, the Access Token shall implement the following requirements.

1. The *access-token* can be used several times until an expiry-date.
2. The *access-token* refers to a specific resource.
3. Receiving the *access-token,* the Resource Provider shall be able to check that the *access-token* comes from the authorized application See End-to-end Security.

NOTE: Because the Resource Provider does not encompass any user information, the *access-token* does not refer to user related information.

The Authorization Server computes the access-token-data as follow:

Access-token-data	Access-token-data := <hash(res-id|res-action)> | <hash(application-id)> | <rcrp-session-keys> | <expiry-date> | <to be completed>

Resource Consumer Resource Provider Session keys:

The Authorization Server randomly generates a *session-encryption-key* and *random-signature-key.*

RCRP-session-keys.	Rcrp-session-keys := <random-encryption-key> | "-" | <random-signature-key>

These keys are used to securely transport data from Resource Consumer and Resource Provider. See End-to-End Security.

Building Access Token with symmetric cryptography: The mechanism is the default mechanism for Smart Object Resource Provider.

The following keys are recomputed by the Authorization Server.

Access-token-signature-key	Access-token-signature-key = function1(key-material, AUTHORIZATION-SERVER/PERSO-CENTER-ACCESS-TOKEN-SIGN-MASTER-KEY)
Access-token-encryption-key	Access-token-encryption-key= function2(key-material, AUTHORIZATION-SERVER/PERSO-CENTER-ACCESS-TOKEN-ENC-MASTER-KEY)

Finally:

Access-token	<sym-encrypt(access-token-data, access-token-encryption-key)> | <sym-sign(access-token-data, access-token-signature-key)>

Building Access Token with asymmetric cryptography:

The mechanism is the default mechanism for Smart Server Resource Provider.

The key-material of the resource is the Smart Server encryption certificate.

The Authorization Server owns a signature certificate and the associated private-key.

Access-token	<say-encrypt (access-token-data, resource-provider-public-key)> | <say-signature(access-token-data, authorization-server-private-key>

NOTE: the Authorization Server certificate shall be provided out-of-band to the Resource Provider.

6.2.8. End-to-End Security

The resource shall be retrieved in a secure way. The protocol implements the following requirements:

- Application data shall be transported securely.
- The Resource Provider shall verify that the *access-token* is a valid one.
- In case the *access-token* has been retrieved by a fraudulent application, the fraudulent application shall NOT be able to use it.

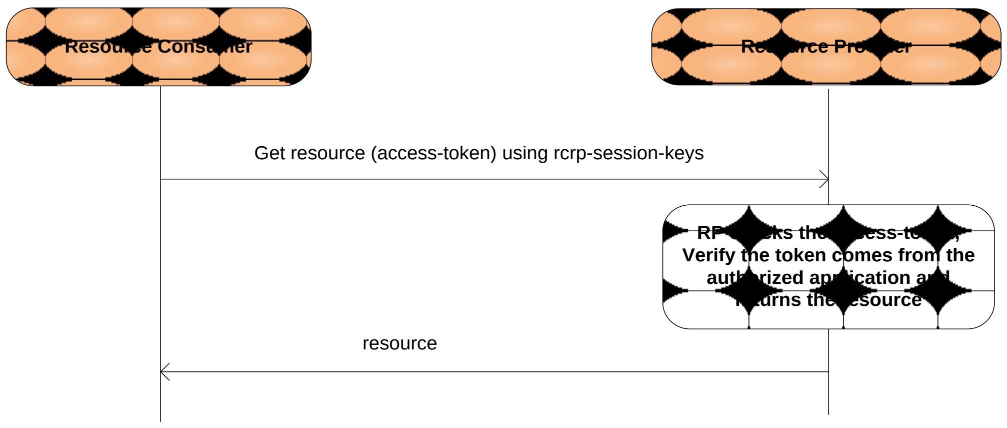

Resource Consumer
Resource Provider
Get resource (access-token) using rcrp-session-keys
RP checks the access-token, Verify the token comes from the authorized application and returns the resource
resource

Figure 51: End to end security

Implementing End-to-End security requirements:

Requirements	Implementation
Requirement 1 Application data shall be transported securely	**RC to RP request.** *<Access-token>* | <sym-encrypt(<application-data>, rcrp-session-keys.encryption-key)> <sym-sign (access-token | application-data, rcrp-session-keys.signature-key)> With Application-data := <hash(application-id)> | <res-action> | <request-payload> **RP processing** RP decrypts the *access-token* to have the *access-token-data.* RP retrieves the *rcrp-session-keys* from the *access-token-data*. RP decrypts the request to have the *application-data.*

	RP checks that *application-data.hash<application-identifier>* == *access-token-data.hash(application-id)* RP checks that *application-data.res-action* is a valid action. **RP to RC response** <sym-encrypt(response-data, rcrp-session-keys.encryption-key) | <sym-sign (access-token | response-data, rcrp-session-keys.signature-key)
Requirement 2. In case the *access-token* has been retrieved by a fraudulent application, the fraudulent application shall NOT be able to use it.	The Resource Provider retrieves the rcrp-session-keys from the *access-token*. The delivery of the rcrp-session-keys to the Resource Consumer is protected by SSL protocol – see Generic Access to Resource. The application-data shall be encrypted and signed by the application, so the *access-token* can be sent in clear in the RC-to-RP request. Optionally, we can reinforce the security by transporting the RC-to-RP request and RP-to-RC response over SSL, but it could be huge and generally require asymmetric cryptography and certificate management in the RP. The risk of *access-token* disclosure is limited to security of the RC site. This risk does not require superseding the security of the RC-RP use case.
Requirement 3. The Resource Provider shall verify that the *access-token* is a valid one.	The RP decrypts the *access-token.* It verifies by: - Checking access-token-data.expiry-date - Checking application - data - see above implementation of **Requirement 1.** - Checking hash(<resource-ident> | application-data.res-action) == access-token-data.hash(res-id|res-action)

6.3. Privacy Principles / Implementation matrix

In this chapter, we recall, the results of the D2.1 deliverables such as Privacy Principles and which principles are taken into account in this high level specification.

Transparency of usage of the data:

How it is implemented: *The Authorization Server registers all user related resource metadata. The user controls the authorization to access the resource by any applications and users. Anyway, the Authorization Server does not store any resource related data.*

Collected Data shall be adequate, relevant and not excessive:

How it is implemented:

The current security specification does not implement this requirement. It is up to Resource Consumers and Resource Providers – which collect data - to fulfill it.

Collector shall use data for explicit purpose:

Data shall be collected for legitimate reasons and shall be deleted (or anonymize) as soon as data is no longer relevant.

How it is implemented:

The current security specification does not implement this requirement. It is up to Resource Consumers to fulfill it.

Collector shall protect data at the communication level:

How it is implemented:

The specification defines the end-to-end security, therefore it fulfills the Principle.

NOTE: at this level of specification, the communication network is seen unsecure. The security can be enforced by applying security at lower level in the ISO communication stack.

Collector shall protect collected data at the data storage

How it is implemented:

The current security specification does not implement this requirement. It is up to Resource Consumers and Resource Providers to fulfill it.

Collector shall allow user to access / remove Personal Data:

How it is implemented:

The Authorization Server registers the user credentials for authentication and authorization purpose. The User is able to manage authorization for accessing any own resources available in the horizontal system. He/she is able to add/remove authorizations.

Except for the user credentials, the Authorization Server does not register any other Personal Data. The User can request to be removed from the system.

Regulation:

How it is implemented:

The architecture enables the Data Protection Framework by separating the Authorization Process to access a resource and the resource related data which are not stored in the server.

6.4. Communication between BUTLER devices

This section presents a draft version of the BUTLER communication architecture that allows the communication between BUTLER devices. A high-level description of the relevant components is also included. We remark that the design of this layer is still in progress in the project and the final version of the architecture will be reported in the deliverable D3.2 due at M24 (September 2013).

The communication layer is responsible for the end-to-end holistic communication, enabling the connection and the interoperability of the smart objects, smart mobiles and smart servers. In addition, it takes care of the devices, users and servers authentication issues.

The current version of the architecture is shown in Figure 52. The communication is based on the IP network. However, as it can be observed from the architecture, smart objects are characterized by different heterogeneous standards and protocols (such as ZigBee, KNX, NFC, CoAP, etc.) that need to be adapted in order to be able to communicate with the BUTLER system and between each other. To this aim, an *IoT Protocol Adapter* module is proposed that ensures the interoperability across different communication technologies. In particular, this module hides the underlying heterogeneous communication technologies by adapting the specific legacy technology to a uniform BUTLER communication interface.

In addition, the communication layer provides also some capabilities regarding the discovery of BUTLER devices. In fact, a new device joining the BUTLER network needs to be discovered, enabling its remote identification. A repository for BUTLER devices should be also maintained. In fact, a *Device Directory* module is responsible for storing data and information about the devices.

Since smart objects are low power devices, these are subject to disconnections caused, for instance, by temporary high interferences. Thus, it would be useful to monitor the connectivity status of every smart object by using a *Network Monitoring* module that measures quantifiable network parameters, such as PLR (Packet Loss Rate) and latency.

The communication layer provides also a module to manage the connectivity of the users' devices (which can use different technologies such as 3GPP, Wi-Fi, Ethernet) to the remote or local BUTLER server. Similarly, another module is responsible for managing the connectivity of the servers in the BUTLER network. As for the BUTLER devices, a functionality is provided for maintaining directories of user devices and servers.

For security purposes, the BUTLER communication layer provides services for authentication. In particular, the authentication functionalities ensure that only identified and authorized devices (and servers) can join the BUTLER network in order to make secure the access control.

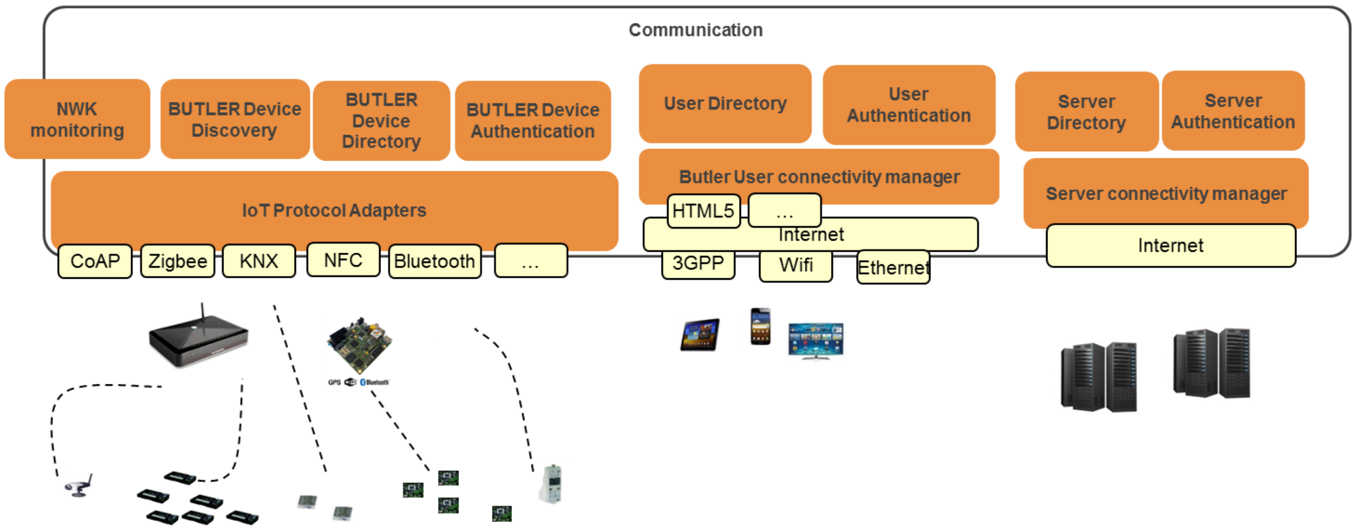


Figure 52: Architecture of the BUTLER communication layer.

Going into more detail of the *Protocol Adapter*, this component ensures interoperability among the different legacy technologies and protocols used by physical devices. A more detailed architecture of this component is shown in Figure 53.

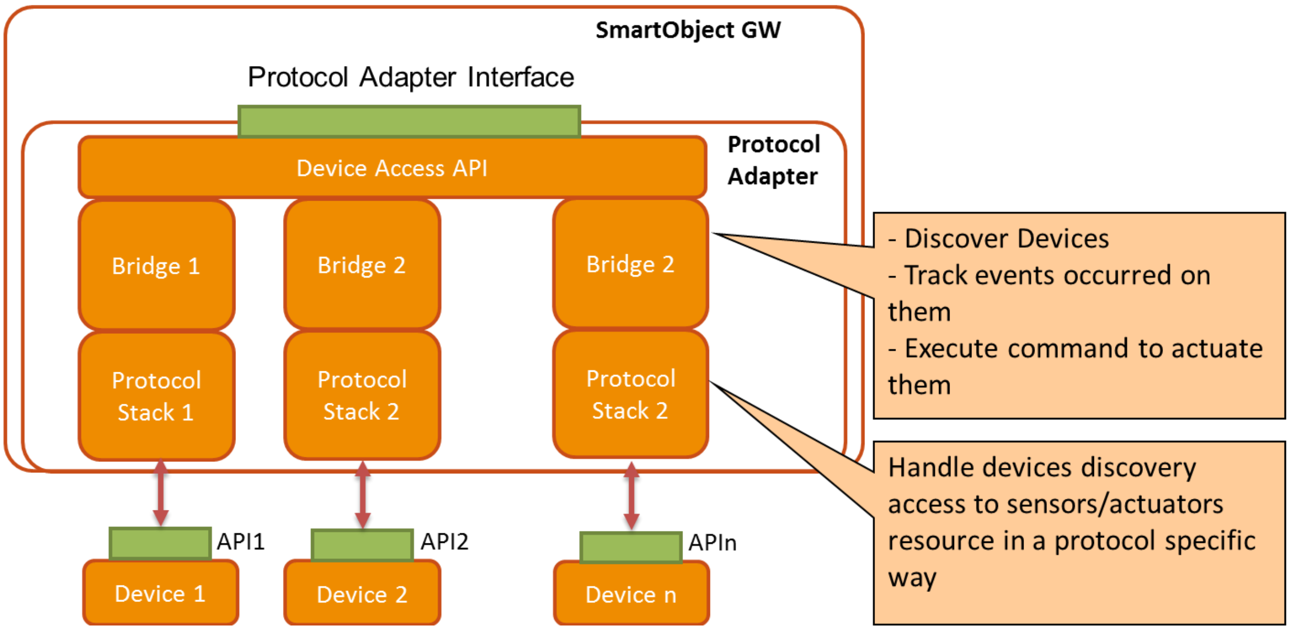

SmartObject GW
Protocol Adapter Interface
Protocol Adapter
Device Access API
Bridge 1
Bridge 2
Bridge 2
Protocol Stack 1
Protocol Stack 2
Protocol Stack 2
- Discover Devices
- Track events occurred on them
- Execute command to actuate them
Handle devices discovery access to sensors/actuators resource in a protocol specific way
API1
API2
APIn
Device 1
Device 2
Device n

Figure 53: Block scheme of the protocol adapter module.

As it can be observed, the *Protocol Adapter* module exposes some APIs, which are protocol agnostic, towards the higher levels of the Gateway. In particular, this interface, implemented in the *Device Access* component, provides "create", "read", "update" and "delete" operations. Moreover, it allows subscription/notification messages to be sent and it is able to publish resource capabilities in the Resource Directory.

The Bridge components adapt the underlying protocols to the BUTLER system. For instance, the device discovery functionalities related to legacy systems are mapped according to the ones defined by the BUTLER layer. In addition, the bridge components manage the IP device's accessibility. In fact, when a device is discovered in a legacy technology, a notification is received by the relative bridge and an IP address is assigned to the device. We remark that the smart object GW architecture will be described in more details in the deliverable D4.3 (BUTLER SmartObject Platform and Enabling Technologies) due at M24.

6.5. Example of horizontal scenario using BUTLER architecture

6.5.1. "Horizontal" Scenario

Scenario of a remote user that consults the weather forecast application (in the cloud), the temperature inside its chalet and starts the heating system at a specific day. User wants to be notified if the temperature of the chalet is out of a specific range. User may also allow a friend to perform the same scenario

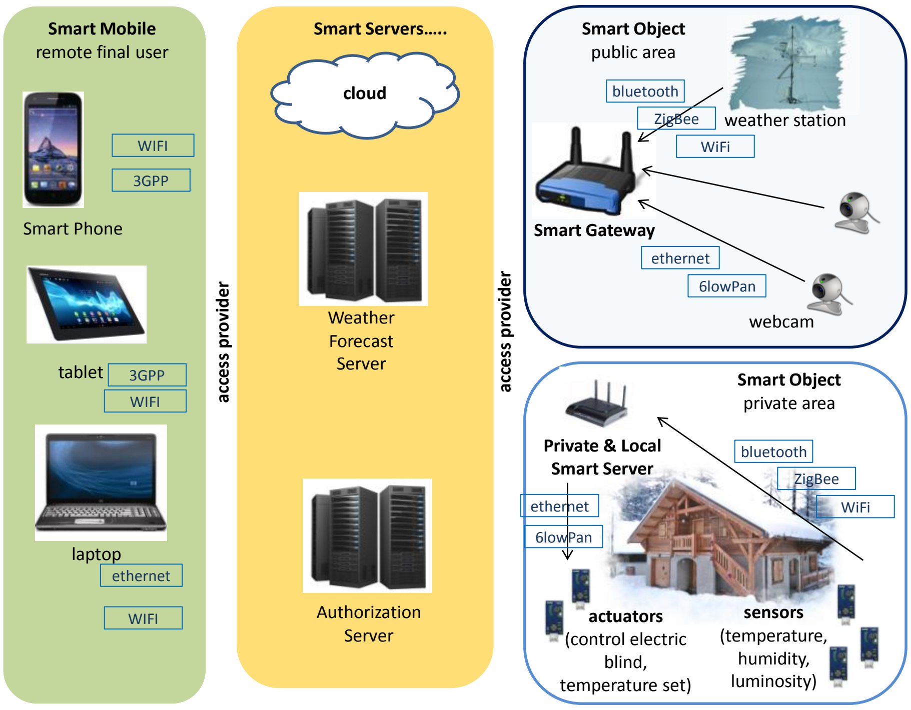


Figure 54: Example of BUTLER horizontal scenario.

"Horizontal Scenario" Analysis.

This scenario high-lights 2 different domain of work:

1. **Public Domain** for weather forecast application.
2. **Private Domain** to handle the private space. Private domain means a domain where the access is fully controlled by the owner of the domain.

Public Domain - privacy and security requirements.

In this scenario, the user gets weather forecast data from cloud application. This weather forecast information is one way. It can be delivered to the final user for free or user shall pay (on demand, subscription …).

To summarize – *for public domain* - we shall manage the following data.

Resource Consumer	Resource Provider	Data	Access Rule	Security Requirements
Any user	Forecast Station	Weather forecast	Read	Integrity

Private Domain - privacy and security requirements.

In this domain, the user reads private data from its chalet and control actuators according weather forecast information from public domain.

To summarize – *for private domain* - we shall manage the following data.

Resource Consumer	Resource Provider	Data	Access Rule	Security requirements
Owner (as any user)	Forecast Station	Weather forecast	Read	Integrity
Authorized users (owner…)	Chalet Local Server	Inside temperature	Read	Integrity, Confidentiality
Authorized users (owner…)	Chalet Local Server	Control Heating system	Write	Integrity, Confidentiality, Authenticity
Authorized users (owner…)	Chalet Local Server	Register Notification URL	Write	Integrity, Confidentiality, Authenticity

6.5.2.Implementation Specification.

The implementation of the scenario instantiates the BUTLER Security Frame defined in Deliverable D2.4 / Security Framework. In this example, the Local System relies on BUTLER authorization system instead of implementing authorization by itself.

Publishing resources to Authorization Server.

Resource Provider	Resource URL	Access Rule
Forecast Station	http://weather-forecast	Read – all users authorized
Chalet Local Server	http://john-smith/gettemperature	Read for authorized users
Chalet Local Server	http://john-smith/controlheating	Write for authorized users

1. *"Weather Forecast" - High Level Message Flow.*

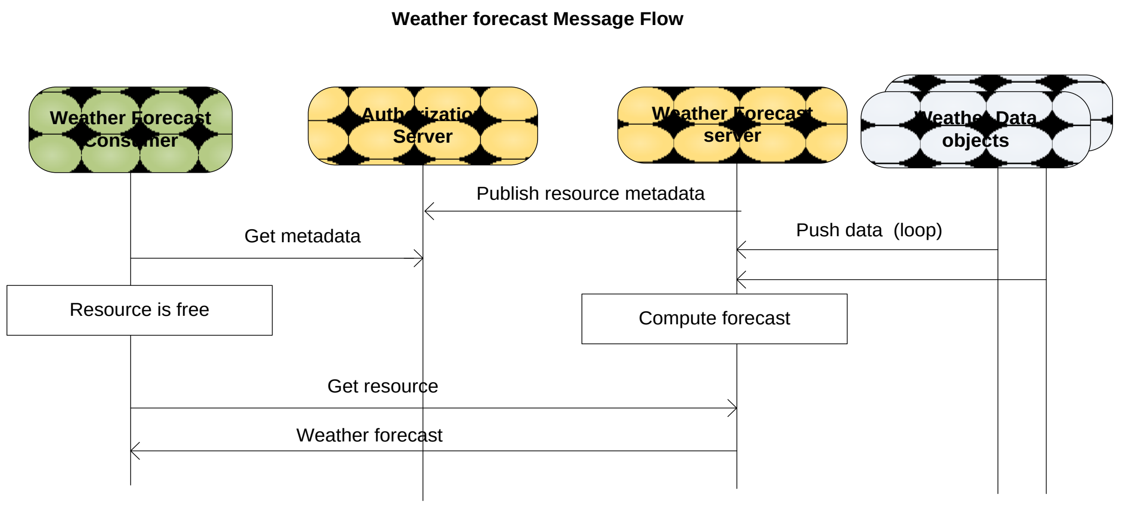


Figure 55 Weather forecast message flow

In this scenario, the resource is free for reading – the Weather Forecast consumer application gets weather forecast metadata from the Authorization Server. The metadata includes the resource URL and access right. The metadata shows that the corresponding resource is free for reading. The Smart Mobile calls the resource URL to read the weather forecast.

2. *Getting Chalet Temperature - High level Message Flow*

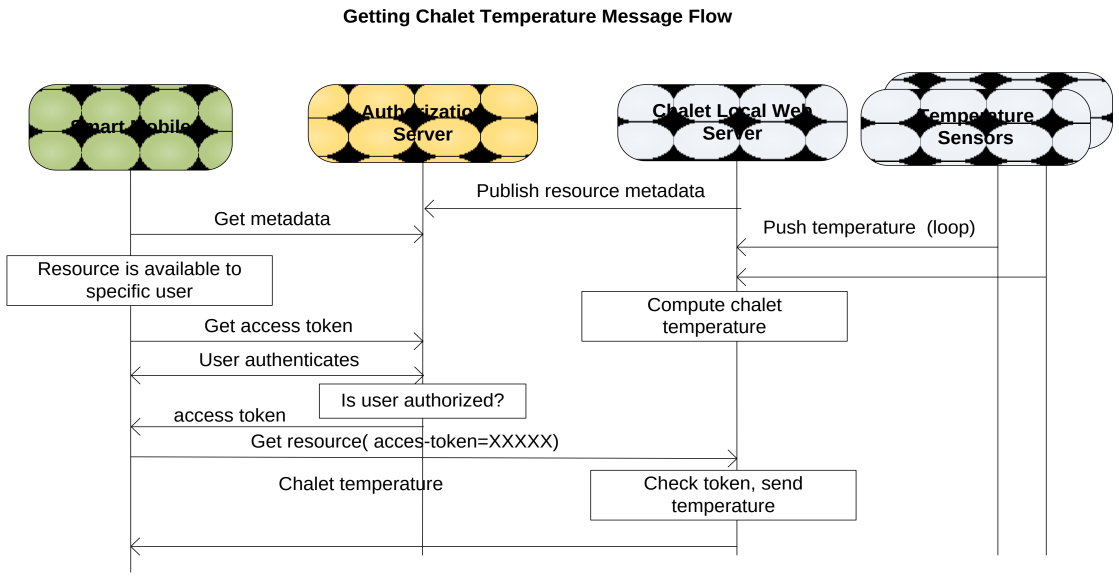


Figure 56 Get Chalet Temperature

In this scenario, the temperature of the chalet is available for read to specific users. The Resource Consumer application (here the Smart Mobile application) retrieves an access-token on behalf of the user. The Chalet Web Server checks the access token and returns the temperature of the chalet.

3. *"Controlling Heating System" - High level Message Flow*

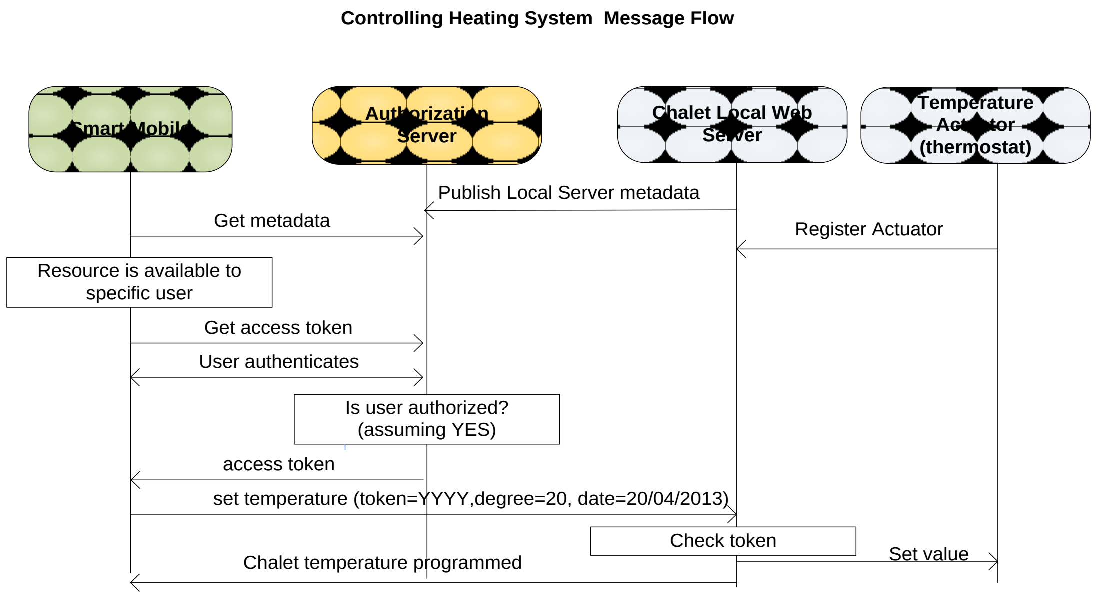

Controlling Heating System Message Flow
Smart Mobile
Authorization Server
Chalet Local Web Server
Temperature Actuator (thermostat)
Publish Local Server metadata
Get metadata
Register Actuator
Resource is available to specific user
Get access token
User authenticates
Is user authorized? (assuming YES)
access token
set temperature (token=YYYY,degree=20, date=20/04/2013)
Check token
Set value
Chalet temperature programmed

Figure 57 Controlling Heating System

According the “Smart Mobile” application, the “Controlling Heating System” use case is equivalent to the “Getting Chalet Temperature”. The difference is for the local part of the scenario. The actuator shall be registered at Local Server side – this allows the Local Server sending value to thermostat.

4. *“Temperature out of range notification” - High Level Message Flows.*

Programming the Notification URL.

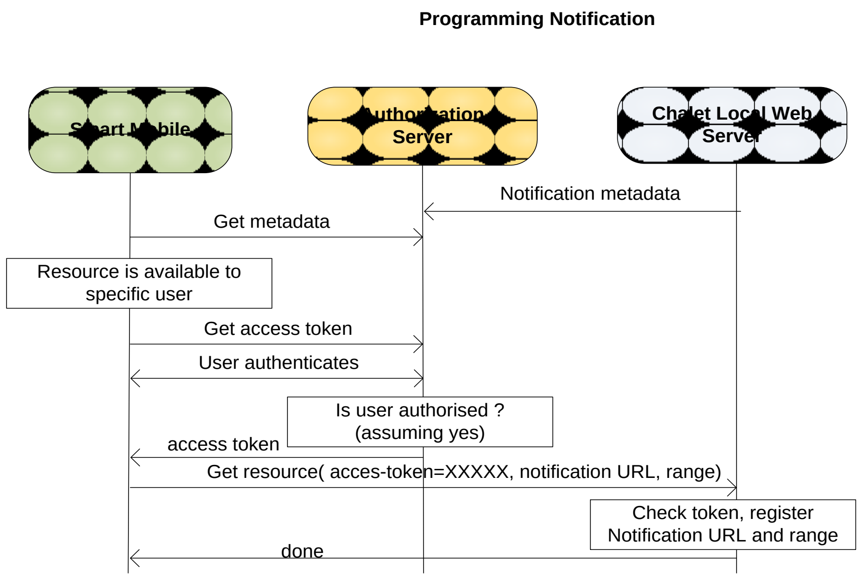

Programming Notification
Smart Mobile
Authorisation Server
Chalet Local Web Server
Notification metadata
Get metadata
Resource is available to specific user
Get access token
User authenticates
Is user authorised ? (assuming yes)
access token
Get resource(acces-token=XXXXX, notification URL, range)
Check token, register Notification URL and range
done

Figure 58 - Programming Notification URL

This scenario is similar to the “Controlling Heating System” except that the Local Server, instead of programming the thermostat, registers the notification URL (for instance a mail address) and the normal range of the temperature.

Notifying the User when the temperature of the chalet is out-of-range

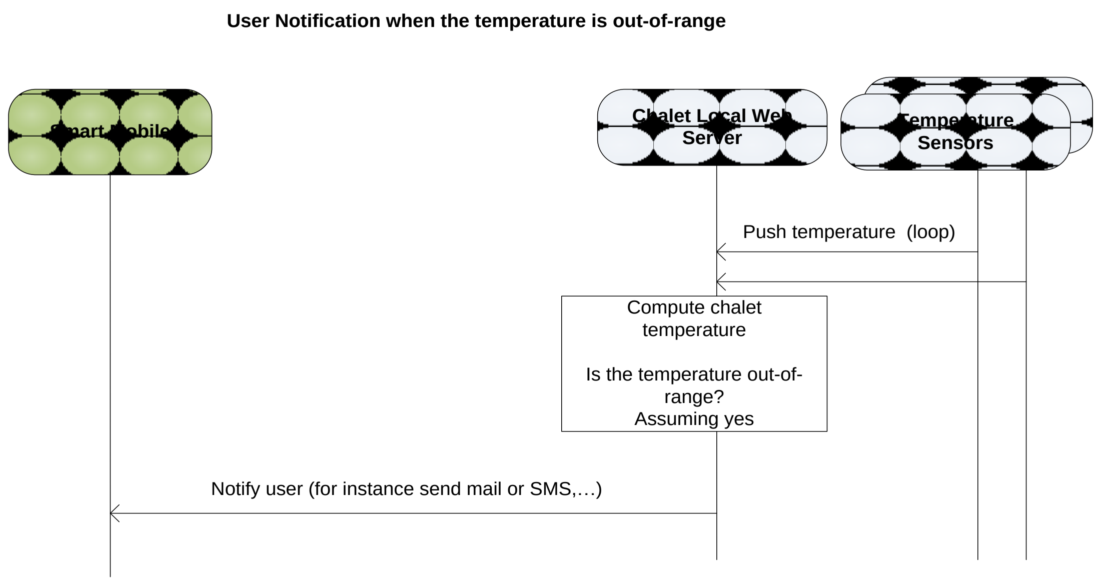

User Notification when the temperature is out-of-range
Smart Mobile
Chalet Local Web Server
Temperature Sensors
Push temperature (loop)
Compute chalet temperature
Is the temperature out-of-range?
Assuming yes
Notify user (for instance send mail or SMS,…)

Figure 59 - Notifying the User

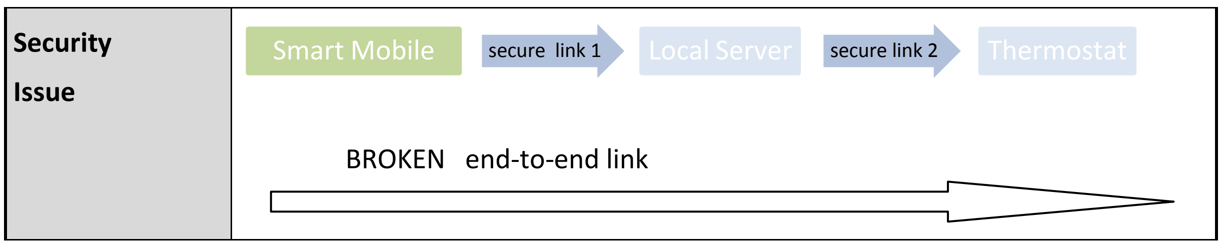
This scenario does not involve any Smart Server. Mailing systems or SMS provider is out of BULTER scope.

6.5.3. "Implementing Security Requirements"

The BUTLER Security Framework defines the security protocol to be implemented to exchange *access-token* and *data* between Resource Consumers and Resource Providers. The above scenarios do not highlight specific issues except the "Controlling Heating System" use case.

Controlling Heating System - Security Consideration.

For the security perspective, we may have a security threat. Actually, there is a protocol translation in the Local Smart Server where the data is in (temporary) is clear. We are in the following situation.



A countermeasure of this threat may consist of implementing the protocol translation in a Secure Element of the Local Server. The data will (temporary) in clear in the Secure Element, but cannot be disclosed outside the secure element.

Another solution could consist of exposing Local Server resources evolving Sensors/Actuator cryptographic key materials instead of Local Server cryptographic key materials. In this schema, the Local Server serves as Smart Gateway for connectivity and will not be able to process any data. It will be a pass-though gateway between application and sensors/actuator.

7. Conclusion

The BUTLER project aims at realizing a technical platform for secure and context-aware IoT that illustrates many scenario of a day of life. This platform is based on several heterogeneous BUTLER devices and remote server that communicate together to exchange data between many application domains. The field trials, presented in the document D1.2 [24] of BUTLER, aims to describe a real-life system involving final users. It gathers the data exchanged in the network and evaluates their relevance and the associated functionalities of the BUTLER architecture. The field trial is based on three components: the fixed infrastructure installed at partner sites, the real mobile devices owned by end users and the real objects including identification tags.

In this document, our goal was to discern and select the relevant technologies able to face BUTLER's challenges and to build the BUTLER system. The BUTLER system is a wide network of heterogeneous communicating objects of any size, technologies, and communication standards. Like for the field trial purpose, the components of the BUTLER network are divided into three categories:

- The Smart Objects are headless devices with scarce resources embedded sensors or actuators. They communicate thanks to low power communication standards as RFID, NFC, 802.15.4 or ZigBee. They are battery operated or equipped with a small solar panel.
- The Smart Mobile provides an interface to the user (regardless of the device used) to offer new context-aware applications and services.
- The Smart Server is a set of components with many computational and storage resources. It may include the "cloud" which allows users to access to services and applications thanks to the Internet. It provides different functionalities and API for BUTLER application developers.

The candidate technologies for each of these components were analysed and selected in this document. The choice of each technology is motivated according to the BUTLER requirements, goals and needs in order to build a wide heterogeneous network. These components integrate new technological bricks developed for the BUTLER project to bring new context-aware features. For example, the Smart Objects handle an enhanced security scheme able to reinforce the protocol security at low layer. They are able to authenticate thank to token as it is identified by a unique identifier. They can integrate actuators to realize a dedicated action. They can also sense physical measurement and send them to the Smart Mobile or to the Smart Server platforms as real and physical inputs to context-aware functionalities. Many geo-localization schemes are proposed to cope with the localization needs. They use different measurement as input and offer to the developer a wide variety of algorithms characterized according to pragmatic criteria. Others functionalities are based on single user and group behaviour analysis and deal with the preference of a person and a group of people according to the human activity. All the selected technologies are finally integrated to build a unique and wide system where the components interact in order to provide new context-aware services and applications to the final user thanks to measurements in a given environment. The final BUTLER platform achieves an end-to-end security: each component of the system is authenticated by an authorization server. So, the data exchanged, the hardware components and the user are secured.

A horizontal scenario was finally briefly discussed to illustrate how the different BUTLER technologies will interact to form a wide network available to the user. The next step will be to map several horizontal scenario on the field trials of the BUTLER platform which will be implemented with these selected technologies.

8. Bibliography

1. **C. Hennebert, V. Berg.** "A framework of Deployment Strategy for Hierarchical WSN Security Management". *4th SETOP International Workshop on Autonomous and Spontaneous Security.* September 2011.

2. **B.Barak, S. Halevi.** "A model and arcitecture for pseudo-random generation with applications to /dev/radom". *ACM Conference on Computer and Communications Security, CCS, Alexandria, USA.* November 2005, pp. 203-212.

3. **E. Barker, A. Roginsky.** "Recommandation for the Entropy Sources Used for Random Bit Generation". *Draft NIST Spcial Publication 800-90B.* August 2012.

4. **A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, S. Vo.** "A statistical test suite for random and pseudo-random number generators for cryptographic applications". 2001.

5. **E. Barker, A. Roginsky.** "Recommandation for Random Bit Generator Construction". *Draft NIST Special Publication 800-90C.* August 2012.

6. —. "Recommendation for Random Number Generation Using Deterministic Random Bit Generators". *NIST Special Publication 800-90A.* January 2012.

7. http://www.ecrypt.eu.org/lightweight/index.php/Block_ciphers.

8. **Delivrable, BUTLER Project.** *"D1.1 Requirements and Exploitation Strategy".* January 2012.

9. —. *"D2.2 Requirements, Specifications, Localization and Context-acquisition for IoT Context-aware Networks.* October 2012.

10. **J. Saloranta, S. Severi, D. Macagnano, G. Abreu.** "Sensor Localization with Algebraic Confidence". *46th Annual Conference on Signal, Systems and Computers.* November 2012.

11. **Dai MingBo, F. Sottile, M.A. Spirito, R. Garello.** "An Energy Efficient Tracking Algorithm in UWB-based Sensor Networks". *8th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob).* October 2012, pp. 173-178.

12. **Jia Tao, R.M. Buehrer.** "A new Cramer-Rao lower bound for ToA-based localization". *Military Communications IEEE Conference, MILCOM.* November 2008, pp. 16-19.

13. **R. Grutter, B. Bauer-Messmer.** "Towards Spatial Reasoning in Semantic Web: A Hybrid Knowledge Representation System Architecture". *10th AGILE International Conference on Geographical Information Science, AGILE.* Springer 2007, pp. 349-364.

14. **G. Destino, G. Abreu.** "Mobile Positioning with Distance Contraction". *IEEE Transaction on Signal Processing (conditionnaly accepted).* 2013.

15. **M. Stocker, E. Sirin.** "PelletSpatial: A Hybrid RCC-8 and RDF/OWLReasoning and Query Engine". *CEUR Workshop Proceedings, OWLED.* 2009, Vol. 259, pp. 2-31.

16. **S. Batsakis, E. Petrakis.** "SOWL: A Framework for Handling Spatio-Temporal Information in OWL 2.0". *5th international Symposium on Rules, Barcelona, Spain.* July 2011, pp. 19-21.

17. **Arun Ramakrishnan, Zubair Wahood Bhatti, Davy Preuveneers, Yolande Berbers, Aliaksei Andrushevich, Rolf Kistler, Alexander Klapproth.** Behavior modeling and recognition methods to facilitate transitions between application-specific personalized assistance systems. *International Joint conference on Ambient Intelligence, Pisa, Italy.* November 2012.

18. **Nayyab Zia Naqvi, Arun Ramakrishnan, Davy Preuveneers, Yolande Berbers.** Walking in the clouds: Deployment and Performance trade-offs of Smart Mobile Applications for Intelligent Environments. *9th International Conference on Intelligent Environments, Athen, Greece.* 2013.

19. **J. Cano, N. Madrid, R. Seepold.** OSGI services design process using model driven architecture Computer Systems and Applications. *IEEE ACS International Conference.* May 2009, pp. 791-794.

20. **M. Lanthaler, C. Gutl.** Towards a RESTful service ecosystem. *4th IEEE International Conference on Digital Ecosystems and Technologies (DEST).* April 2010, pp. 209-214.

21. **T.Erl.** *Service-Oriented Architecture: Concepts, Technology and Design.* New Jersey, USA : Prentice-Hall PTR, Upper Saddle River, 2005.

22. **Johnson, R.** J2EE development frameworks. *Computer.* January 2005, Vol. 38, 1, pp. 107-110.

23. **Various authors.** The JBoss Application Server. *http://www.jboss.org/jbossas.* [En ligne]

24. **Consortium, BUTLER.** *D1.2 Redifined Proof of Concept andField Trial Specification.* 2013.

25. **Zach Shelby, Carsten Bormann.** *6LoWPAN: The Wireless Embedded Internet.* s.l. : Wiley.

26. **Jean-Philippe Vasseur, Adam Dunkels.** *Interconnecting Smart Objects with IP.* s.l. : Morgan Kaufmann Publishers.