ROBOTS IN ASSISTED
LIVING ENVIRONMENTS

UNOBTRUSIVE, EFFICIENT, RELIABLE AND MODULAR
SOLUTIONS FOR INDEPENDENT AGEING

Research Innovation Action

Project Number: 643892 Start Date of Project: 01/04/2015 Duration: 36 months

DELIVERABLE 4.1

Architecture for extending smart homes with robotic platform

Dissemination Level	**Public**
Due Date of Deliverable	Project Month 6, September 2016
Actual Submission Date	30 September 2015
Work Package	WP4, *Physical home architecture development and integration of cost-effective, reliable and power-efficient RADIO components for elder monitoring and caring*
Task	T4.1, *Designing device interconnection and interfacing*
Lead Beneficiary	RUB
Contributing beneficiaries	TWG, S&C, AVN
Type	Report
Status	Submitted
Version	Final

Project funded by the European Union’s Horizon 2020 Research and Innovation Actions

Abstract

Architecture document, pertaining to RADIO device interconnection and interfacing; specifications on interfacing the different domains; and on fast and energy efficient data processing in the distributed RADIO environment.

History and Contributors

Ver	Date	Description	Contributors
01	01 Sep 2015	First draft, establishing document structure and work allocation. Writing Introduction and outlining the content of the first chapter.	RUB
02	09 Sep 2015	Writing Z-Wave device interconnection and interface. Writing outline for cross domain specifications	RUB, S&C
03	22 Sep 2015	Writing the first version of Data processing section.	AVN
04	24 Sep 2015	First complete version of data processing section.	TWG, RUB
05	25 Sep 2015	New version based on comments from NCSR-D	TWG, RUB
06	28 Sep 2015	Pre-final document preparation and internal review submission	RUB
07	30 Sep 2015	Internal review comments and corrections	NCSR-D
Fin	30 Sep 2015	Final document preparation and submission	NCSR-D

Abbreviations and Acronyms

BLE	Bluetooth Low Energy
ID	Identification
IoT	Internet of Things
WIFI	Wireless Fidelity
LAN	Local Area Network
API	Application Programming Interface
HCI	Host Controller Interface
LLC	Logical Link Control
ISM	Industrial Scientific Medical
SoC	System on Chip
FPGA	Field Programmable Gate Array

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 Purpose and Scope

This deliverable should be seen as an architecture document, pertaining to RADIO device interconnection and interfacing, specifications on interfacing the different domains, and on fast and energy efficient data processing in the distributed RADIO environment.

Within the scope of this document is:

- To design the physical architecture of the RADIO Home, and especially the wireless communications architecture between the mobile robot platform and the Smart Home devices that make up each RADIO Home.
- To design the architecture of the heterogeneous computing elements of the RADIO Home, including the central server, FPGAs, and the on-board robot controller.

Outside the scope of this document is the architecture (either conceptual or physical) of the communication between the RADIO Home and other nodes of the RADIO ecosystem, such as cloud storage components and components meant to be used by hospital personnel or informal care-givers. This will be dealt in Task 5.1.

1.2 Approach

This deliverable documents work in Task 4.1, which specifies and designs the interconnection structure and interfaces to exchange data between the home automation infrastructure and the robotic platform. This task also specifies the sensors and the processing units such as FPGAs, the robots on-board computer, or other computers on the premises which comprise the *RADIO Home*, the part of the overall RADIO system that is deployed within a single home. Here, Task 4.1 also tackles the following assignments:

- Investigating the most efficient way, in terms of power and delay overhead, to process different kinds of sensor data in the distributed RADIO environment.
- Observing privacy for the user.
- Observing technical limitations such as bandwidth and processing power.
- Striving for robustness through device redundancy.

Alternative hardware configurations are also investigated as part of this task with the focus on power and performance trade-offs between fixed function accelerators and more programmable solutions such as reconfigurable co-processors with adaptive data paths (e.g. vector or multiscalar processors, ALU adaptation). These Programmable solutions offer more flexibility to provide several dedicated services to the end-users through software updates or extensions.

We have approached work in this task as follows:

- We designed the physical communication architecture for each of the communication groups abstractly specified in D3.1, Section 4. This results in specifying three communication domains: ZWave, Bluetooth, and WiFi (Section 2).
- We then designed the interconnection between these communication domains (Section 3).
- We finally specified the computational nodes (FPGA and general-purpose computer) needed in order to execute the ADL and mood recognition specified in D3.1, Sections 2 and 3. We also specified the interfaces with their respective infrastructural system components, such as nodes or/and sensors.

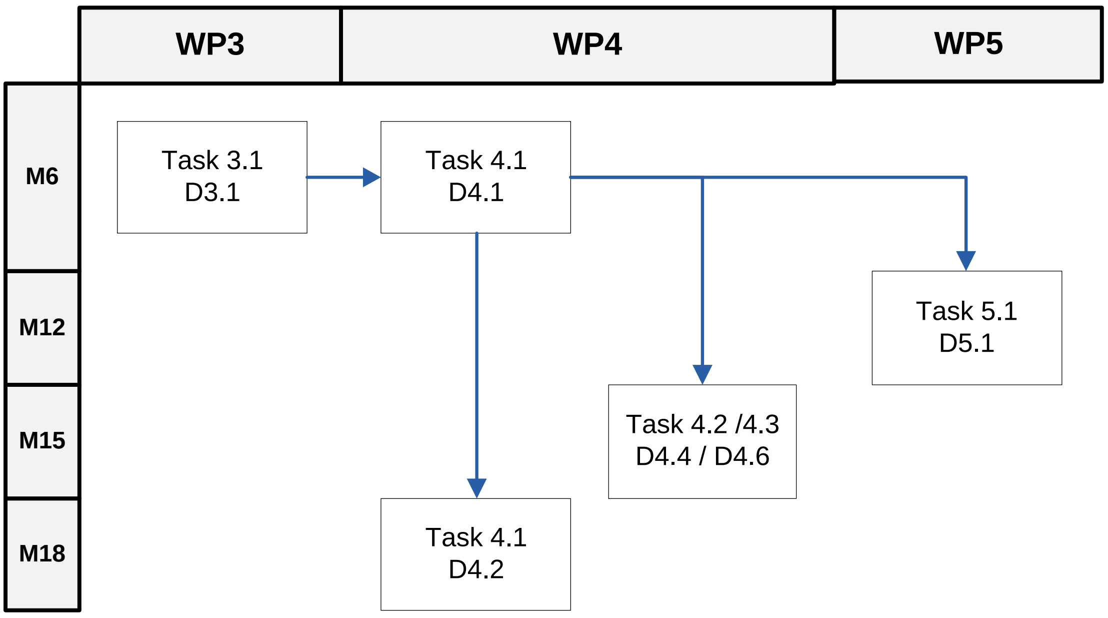

Figure 1: Relation to other Work Packages and Deliverables

The definition of the conceptual architecture in Task 4.1 has also allowed the consortium to refine the approach to WP4 as a whole and to establish the following work plan:

- **TWG** and **AVN** will design the interface for data transfer and communication among network nodes, and design hardware modules‘ interfaces with their respective infrastructural system components such as nodes or/and sensors.
- **TWG** and **S&C** will ensure compatibility of the RADIO-prototyped components with the rest of the system through emulation or detailed analysis.
- **RUB** and **TWG** will explore various hardware configurations (fixed-function vs. programmable accelerators) and investigate in adaptive processors e.g. with a reconfigurable data path.

1.3 Relation to other Work Packages and Deliverables

This document is the first in a series of closely related deliverables. The initial and intermediate versions due in month M6 (September 2015) and M18 (September 2016) are used to drive development in Tasks 4.2 and 4.3. From M19 until M30, this is a living document updated to record adjustments necessitated as development in Tasks 4.2 and 4.3 progresses. The final version (M30) documents the architecture and interfacing of the final hardware components and robotic platform.

This deliverable is prepared based on the conceptual RADIO Home architecture (D3.1) as well as medical and ethical requirements set in D2.2 *Early detection methods and relevant system requirements* and D2.4 *Actual and perceived privacy considerations and ethical requirements*. The physical architecture developed in this deliverable is used by Task 5.1 in order to prepare the first version of the architecture of the overall RADIO system, D5.1 *Architecture of the RADIO ecosystem*.

2 DEVICE INTERCONNECTION AND INTERFACING

This chapter specifies the interconnection between the different devices within the smart home environment and defines how the respective devices are interfaced with the smart home infrastructure. This information is required for the further development in Tasks 4.2 and 4.3.

2.1 Interconnections in the Smart Home

The smart home is comprised of several devices with different communication protocols. In order to ensure fault-free data exchange between all devices, the required interconnections have to be identified. Figure 2 shows all available devices within the smart home environment and their respective interconnections.

As can be seen, there are three different communication protocols in the smart home e.g. WIFI/LAN, Bluetooth, and Z-Wave. WIFI/LAN is used for coarse grained communication between the gateways and the router which provide access to the internet. Additionally, the robot is connected via WIFI to the router and therefore is able to access the gateways which serve as data hub for the Bluetooth and the Z-Wave network. Note, that the robot has a Bluetooth connection to the BLE devices. However, the robot only accesses specific Bluetooth devices such as beacons for localization and context information. The main access point for all smart home devices are the gateways which are connected to the smart home devices via Z-Wave and to the BLE devices via Bluetooth. It is therefore very important to define for the three networks within the smart home, the respective interfaces on the communication participants i.e. the robot platform, the router, the gateways. The interface of the robot will be explained in the following subsections.

2.2 Robot Interface to Smart Home

As depicted in Figure 2 the robot is expected to support two wireless communication interfaces. On one hand it is required to support a BLE communication interface so as to communication with BLE beacons indication localization information which is critical with respect to the efficient navigation of the robot itself. BLE beacons only transmit their unique ID which can then be used to access position sensitive information. Therefore, they are generally placed at fixed positions in order to guarantee accurate information when receiving the ID. Additional meta-data for localization can be extracted from the BLE beacons when considering the *received signal strength (RSS)*. A typical off-the-shelf BLE-USB dongle that can be utilized for such purposes is depicted in Figure 6.

On the other hand the robot must be able to interact with the RADIO Home Controller in order to access services external to the RADIO Home (Section 4.4). Physical connection between the mobile robot and the RADIO Home Controller will be achieved through WiFi connection to the ADSL router that also provides internet connectivity. An off-the-shelf WiFi sub dongle will suffice for such connectivity requirement.

2.3 Bluetooth Devices Interface

Bluetooth Low Energy (BLE), a distinctive feature of the Bluetooth 4.0 specification[1], constitutes an emerging wireless technology for short range communication which is expected to be increasingly employed in numerous control and monitoring applications that require low power operation. Below, the main characteristics and mechanisms of the Bluetooth protocol stack are described, as well as the referenced topology and network roles which refer to the nodes in a Bluetooth network.

[1] http://www.sigmadesigns.com/ Bluetooth Special Interest Group, “Bluetooth Core Specification v4.0”, 2010, Available at: https://www.bluetooth.org/en-us/specification/adopted-specifications

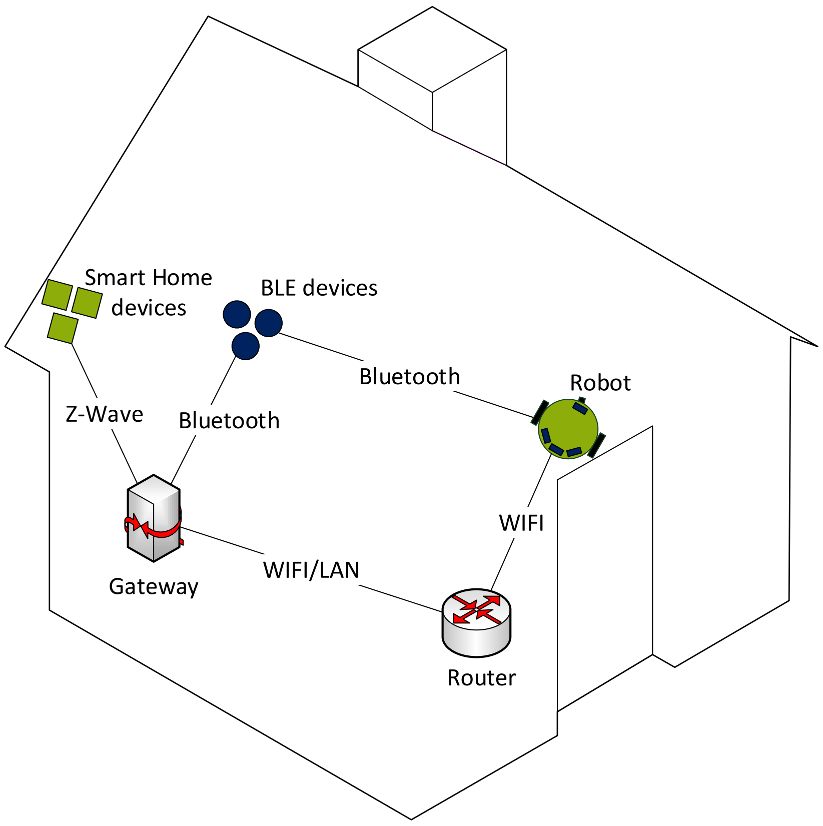

Figure 2: Device interconnection within the smart home environment

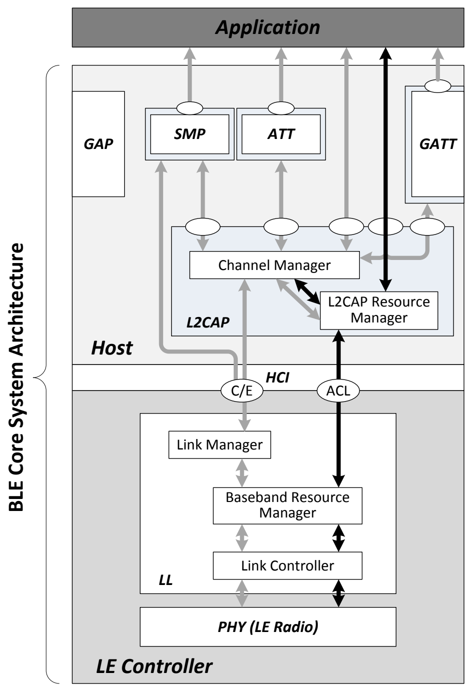

Figure 3 Bluetooth Low Energy (BLE) Core System Architecture (adapt. [1])

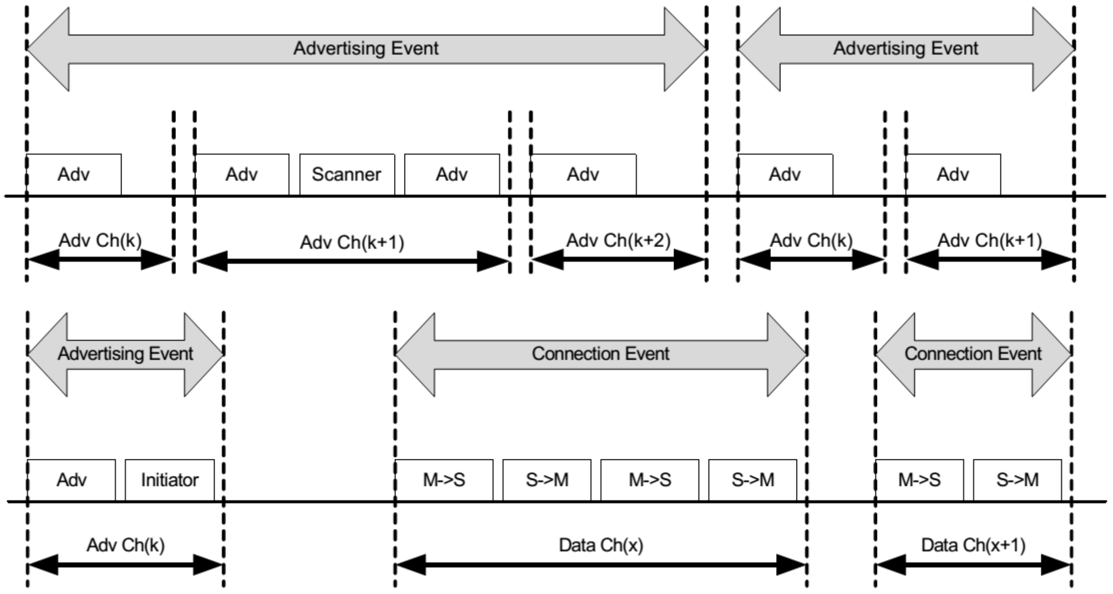

Figure 4 Advertising events and Connection events

Based on the structure of the protocol stack used in previous versions, the BLE protocol stack consists of a Host and a Controller, which are separated by a standardized interface, the *Host Controller Interface (HCI)*. As shown in Figure 3, the Host is a logical entity defined as a set of layers below the application layer (or non-core profiles) and above HCI, typically running on an application processor and including the Logical Link *Control and Adaptation Protocol (L2CAP)*, the *Attribute Protocol (ATT)*, the *Generic Attribute Profile (GATT)*, the *Security Manager Protocol (SMP)* and the *Generic Access Profile (GAP)*. On the other hand, the Controller is defined as a set of layers below HCI, typically being implemented on a small *System-on-Chip (SoC)* that provides wireless communication via a radio module and including the Physical Layer and the Link Layer. An implementation of the Host and the Controller may contain the respective parts of the HCI, the upper and the lower ones, respectively. In the rest of this section, each of those layers and components comprising the Host and the Controller parts of the BLE core system architecture are shortly described.

2.3.1 Physical Layer

BLE defines 4.0 radio frequency channels in the Industrial Scientific Medical (ISM) band of 2.4 GHz with 2 MHz channel spacing. However, those PHY channels are used for different operations, namely advertising and data transfer, thus resulting in 3 advertising channels which are used for broadcast transmission, device discovery and connection establishment, and 37 data channels (the rest of the band) which are used for bidirectional communication between connected devices. In order to deal with interference imposed by other wireless technologies using the 2.4 GHz band (e.g. Wifi), as well as wireless propagation issues in general (i.e. multipath, fading), an adaptive frequency hopping mechanism is used, selecting one of the 37 available data channels for communication between connected devices during each particular time interval (the sequence of which is determined at the beginning of the connection), as shown in Figure 4. The reached data rate is about 1 Mb/s.

Considering those two types of PHY channels in terms of usage, the communication over the advertising or the data channels forming a physical channel is subdivided into time units known as events, in which data packets are transmitted between the devices and depending on the operation they are called advertising or connection events. Based on these, devices that transmit advertising packets over the advertising PHY channels are referred to as *advertisers* and devices that just receive advertising packets without making any connection are referred to as *scanners*, while devices which need to form a connection with devices transmitting (connectable) advertising packets are referred to as *initiators*. When a connection is established between an advertiser and an initiator, they become the *slave* and the *master* devices, respectively, of a *piconet* (see BLE topology). The beginning of each advertising event is defined by the advertiser, while the beginning of each connection event is defined by the master device in a piconet.

2.3.2 Link Layer

In essence, the Link Layer provides the basic acknowledgement/repeat request (ARQ) protocol in BLE. When a device only needs to broadcast data, it forms advertising packets and transmits them over the advertising PHY channels in the time intervals defined by the advertising events, therefore it needs to become an advertiser. On the other hand, when a bidirectional data communication between two devices is required, they need to be connected to each other, thus forming a piconet, where the advertiser device becomes a slave device and the initiator device becomes the master device. The procedure being followed for the creation of a connection between two devices includes an advertiser transmitting over the advertising channels connectable advertising packets, that is, advertising packets indicating that the particular device accepts connections. An initiator, upon listening the advertisements, transmits a connection request message to the advertiser (if interested), thus creating a point-to-point connection between the two devices. A 32-bit access address is assigned to the link in order to identify communication over the data PHY channel sequence indicated by the initiator (piconet master). Of course, the master may initiate connections with other advertisers (piconet slaves), too. Therefore it forms a star topology, while it coordinates the medium access (TDMA scheme) by determining the time and the PHY channel the slaves are required to listen for packets being sent by the master and transmit packets as response.

More specifically, focusing on the comprising components, the Link Manager uses a Link Layer Protocol (LL) in order to create, modify and release logical links, as well as update parameters related to physical links between devices. The Baseband Resource Manager is responsible for all access to the radio medium, negotiating and scheduling the access over the physical channels of the various entities that use the BLE Controller. Finally, the Link Controller is responsible for the encoding and decoding of Bluetooth packets from the data payload of the transmitted packets, as well as for the communication of flow control, acknowledgement and retransmission request signals based on the employed Link Layer Protocol.

2.3.3 L2CAP

The main responsibility of L2CAP is to multiplex the data of the higher layers (i.e. ATT, SMP. GATT) over the connection being created by the Link Layer, by assigning different channels to each of the corresponding services. Therefore, the Channel Manager creates, manages and closes L2CAP channels for the transport of data related to service protocols and applications (i.e. establishes and handles connection endpoints/channels for entities in different BLE devices). Furthermore the L2CAP Resource Manager manages the (order of) packet submission to the controller for transmission over a physical channel dealing with resource exhaustion issues related to the Controller functionality.

2.3.4 SMP

The SMP is the peer-to-peer protocol that is used to generate and manage the encryption and identity keys in the case of encryption and pairing procedures. Its operation is assigned to a fixed L2CAP channel. SMP effectively offers a series of security algorithms offering the three main security services required, i.e. privacy, authentication and authorization. In essence through defined procedures and adopted security algorithms communication peers exchange data packet securely over encrypted link and are able to trust the identity of the remote device. All such services occur between two entities i.e. the initiator (always corresponding to the Link Layer master) and the responder (always corresponding to the Link Layer slave). Specifically SMP defines three procedures as follows:

- *Pairing*: Generation procedure of a temporary common security encryption key required so as to switch to a secure, encrypted link. The lifetime of the particular key is only the during the current connections
- *Bonding*: Following a successful sequence of pairing this procedure defines the generation and exchange of permanent security keys. Stored in memory these keys allow the peers to quickly set up a secure link in subsequent connections without having to perform a bonding procedure again.
- *Encryption Re-establishment*: Third sequence after bonding defining the way permanently stored key can be used to re-establish a secure, encrypted connection without having to go through the pairing (or bonding) procedure again.

Finally in order to have a good understanding of SMP offers the required services one must note the three security mechanisms which can be used either independently from each other or concurrently.

- *Encryption*: Encryption of all packets transmitted over an established connection.
- *Privacy*: An advertiser is allowed to hide its public Bluetooth address by using temporary, randomly generated addresses that can be recognized by a scanner that is bonded with the advertising device.
- *Signing*: With this service the source of a packet can be verified and packet can be sent unencrypted.

2.3.5 ATT

The ATT defines the peer-to-peer communication of attribute data between two devices over a fixed L2CAP channel, having the roles of a client and a server. The ATT server maintains a set of attributes which are data structures including information managed by the GATT. The ATT client can access the server’s attributes by sending commands, requests and confirmations to the ATT server, and the latter sends responses, notifications (not requiring client confirmation) and indications (requiring client confirmation) to the ATT client.

2.3.6 GATT

The GATT represents the functionality of ATT server and (optionally) the ATT client. It provides a framework that uses the ATT protocol to discover, read, write and indicate attributes and characteristics related to services (profiles) provided by a device, organized in specific hierarchies. Such data structures include a value and some properties describing the data (e.g. format) and defining the access permissions.

2.3.7 GAP

The GAP implements the basic functionality which is common to all BLE devices, including services such as device and service discovery, connection mode selection, connection establishment, and connection security, authentication and association models. Each of those services are described in terms of a profile that can be used and extended by the applications (application profile). The GAP also specifies four device roles which correspond to specific requirements to be provided by the LE controller, namely Broadcaster, Observer, Peripheral and Central. A Broadcaster device just broadcasts data using the advertising channels and does not establish connections with other devices, while an Observer device receives the data transmitted by a Broadcaster device and send responds (scan requests). On the other hand, a Peripheral device is connected and handled by a Central device over an established connection supporting the slave and master role, respectively, in a formed piconet.

Figure 5 and Figure 6 show indicative examples of commercial hardware devices that will be used to support BLE connectivity in the context of RADIO. Specifically, Figure 5 is effectively a very small sensor tag from Texas Instruments offering full support for the Bluetooth Core 4.1 specifications.

Figure 5 TI BLE Sensor Tag

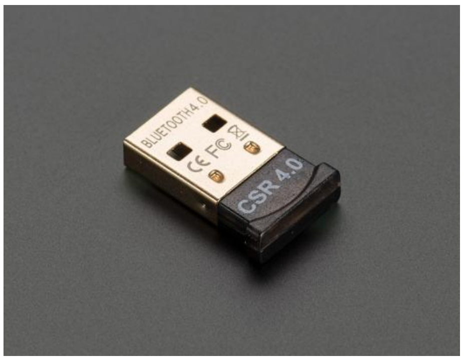

Figure 6 BLE USB module

It is over the air upgradable, offering a Bluetooth Smart network processor solution with simple API while enabling advanced topology control concurrent master/slave operation and support for up to 8 connections. On the other hand, Figure 6 depicts a BLE USB module which basically adds BLE capabilities to any computer with a USB. The main objective of this module is to offer BLE standard based interface to the respective *Gateway (GW)* implemented in a Linux Raspberry Pi. Consequently attention has been paid on report indicating proven cooperation with Linux running raspberries. Additionally, it hosts the open source Bluez stack offering additional advantages.

Furthermore, the experimental grade development platform will also be considered offering opportunities for enhancements and extensions if needed. Specific examples that will be considered in RADIO are presented in Figure 7 and Figure 8. The former one basically is a TI BLE development platform providing a complete hardware performance test platform and generic software development environment for single-mode Bluetooth® low energy (BLE) applications. The evaluation modules can be used as reference modules for prototyping and for verifying the performance of the CC2540 RF IC. SmartRF Studio and SmartRF Packet Sniffer can be used with the hardware in the kit for testing and debugging purposes. The evaluation boards has an in-circuit debug probe, so no additional hardware tools are required for debugging of software running on the CC2540. The former case is a BLE development platform from CSR being another key player in this area which offers mesh connectivity capabilities. CSRmesh™ is a protocol that was built with the aim to run on Bluetooth 4.0 and later. Bluetooth Smart is the only prerequisite necessary for CSRmesh to work. Such protocols can provide valuable information on aspects enabling to extend coverage area of a network while preserving minimum power consumption.

The gateway will host the RADIO Home Controller which manages all connections with BLE-based devices, aggregates acquired data, and issues required commands. The RADIO Home Controller also provides access to services external to the RADIO Home (Section 4.4).

Concluding, this short introduction clearly depicts all critical characteristics of Bluetooth Low Energy technology highlighting it as a prominent candidate for short range wireless communications in RADIO project able to meet all requirements as identified and reported in D3.1. It is noted that adequate enhancements, extensions and additions on WSN communication technologies may be adopted in the next phases of the RADIO prototype development as needed so as to meet the overall project objectives.

Figure 7 TI's BLE Development Kit

Figure 8 CSR's BLE Development Board

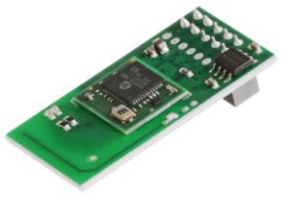
Figure 9 zWave module (called RaZberry) for interfacing the gateways to the Z-Wave network

2.4 Z-Wave Devices Interface

Z-Wave devices use a proprietary protocol from Sigma Designs, http://www.sigmadesigns.com. Z-Wave devices can communicate directly, but in the most common case, they form a wireless network being managed by a node called controller. The controller is responsible to manage the Z-Wave network and Z-Wave devices, and provides APIs to the controller application in order to create simple or complex systems through the combination of the different device functionalities that forms the network. In the case of Sensing & Control's smart home system, the system provides functions for security, comfort, home automation and energy management.

Figure 9 shows the chosen hardware device that brings Z-Wave connectivity in order to communicate with Z-Wave products. Those devices are used by controllers, in the case of Sensing & Control smart home. The controller and smart home application use the Z-Wave module called RaZberry for the Raspberry Pi as seen in Figure 9.

In the current version of the smart home, the main controller runs in the Linux-based RADIO Home Controller, the gateway that is the only point for communication between Z-Wave devices and services external to the RADIO Home (Section 4.4).

For the first integrated prototype, any clients of the Z-Wave Smart Home devices need to access them through the RESTful API provided by the cloud-based IoT Platform. For subsequent prototypes, the RADIO Home Controller GW will provide local RESTful API for direct communication with smart home devices and functions without accessing the cloud-based IoT Platform. The latter will then be restricted to receiving abstract high-level events (Section 4.4).

Client functions may be integrated within the smart home application running in the Z-Wave gateway if needed. This possibility will depend on the amount of computational resources these potential functions need, it should be carefully evaluated in order not to compromise the performance of smart home functions already implemented at the gateway due to its limited hardware resources (CPU power and amount of RAM memory need).

3 Specifications on Interfacing the different Domains

This chapter deals with the challenge of transferring data through several different protocol domains. In order for this to be consistent, it must be specified which entity in the smart home actually requires this cross-domain communication and what kind of data has to be transferred when crossing from one domain into the other. In the smart home environment, there are three protocol domains:

- Bluetooth domain
- Z-Wave domain
- WIFI/LAN domain

In each domain, different entities are present such as the robot platform, the Bluetooth devices, the Z-Wave devices, the gateways and the router. Some of these entities are able to interface with several protocol domains. Each domain must have a specification on how it is able to interface with another domain and what kind of data will be made available to the other respective domain.

The RADIO Home Controller's main function is to connect to the IoT Platform (running in the cloud) in order to store high-level events relevant to interRAI assessment items, as well as emergency notifications. The IoT Platform is used as the central repository of abstract, high-level information and it provides a RESTful API to exploit high-level events relevant to interRAI assessment items and other smart home historical data and to interact with the devices at home as it is for Z-wave based devices as well.

3.1 Bluetooth domain

The Bluetooth domain interfaces with two different entities, the smart home gateways and the robot platform. The robot platform has a direct connection to the Bluetooth beacons and to the LAN of the smart home. Therefore, the robot can potentially function as cross-domain interface for Bluetooth-to-LAN. Since the robot functions as an information sink when regarding the Bluetooth beacons, no domain crossing will be performed between the Bluetooth protocol and the LAN protocol when the robot is concerned.

The smart home gateways are also connected to the Bluetooth domain. In the Bluetooth context, the smart home gateways have two functions. The first function is to enable data exchange between different Bluetooth devices. The second function of the gateways is to collect the data from Bluetooth devices and send it to the central RADIO Home controller for home-level analysis and aggregation before storing to the IoT Platform (Section 4.4).

Therefore, the gateways need to extract the payload of each Bluetooth message and send it via LAN to the router and then to the IoT platform. Additionally, provision should be offered regarding communication in the reverse direction, i.e. from the IoT to the gateway (using LAN/WiFi communication) and from the gateway (using Bluetooth communication). In this way, cases of issuing commands, reconfiguration options and even reprogramming could be supported. This method also enables the communication of Bluetooth devices which are not connected via the same gateway, since all gateways are connected via LAN. A summary of the Bluetooth cross domain specifications is given in Table 1. Consequently, in the gateway a software infrastructure will be developed which will interact with the Bluetooth devices through serial port data transfer, on one hand, and support the IoT platform web-service communication protocol on the other hand.

Furthermore, an on-going critical aspect concerns the management of information from the moment useful information is extracted from Bluetooth packets until the moment of the web-service invocation. In that respect different approaches will be evaluated ranging from straightforward parsers to sophisticated data management infrastructures. The former approach pertains to low complexity developments which focus on the immediate payload extraction from one side and immediate passing to the other without any data handling without any actual interaction with the useful information. Therefore respective solutions can be executed on platforms of scarce resources but they are usually deprived from features like data buffering capabilities, application of specific traffic balancing policies, traffic bursts optimal handling, security policy application as well as mechanism to cope with temporal connectivity loss in either side. Regarding the latter aspects solutions will be evaluated based upon messaging passing protocols widely used in various domains nowadays. In such context an entity usually called "broker" received data being "published" by the e.g. sensors and handles them in the form of queues. Then the broker effectively sends data to "subscribers" which in RADIO case is the IoT platform or platforms. Integrating "broker" functionalities in the Bluetooth gateway could enable the application, modification, extension or even from scratch development of adequate mechanisms handling data residing in queues so as to better meet end to end communication objectives.

Table 1: Specification for the cross domain interfaces for the Bluetooth domain

Bluetooth Domain Specification		
Cross domain interfaces	**Bluetooth – Z-Wave**	**Bluetooth – WIFI/LAN**
Necessity	not required	required
Participating entities	Smart Home gateways	Robot platform Smart Home gateways
Information exchange	commands for device manipulation	None position, type of device, functionality, payload

Table 2: Specification for the cross domain interfaces for the Z-Wave domain

Z-Wave Domain Specification		
Cross domain interfaces	**Z-Wave – Bluetooth**	**Z-Wave – WIFI/LAN**
Necessity	not required	required
Participating entities	Smart Home gateways	Smart Home gateways
Information exchange	commands for device manipulation	position, type of device, functionality, payload

3.2 Z-Wave domain

As seen in Figure 2, the Z-Wave domain only interfaces with the smart home gateways. The gateways however are connected to the Bluetooth and to the WIFI/LAN domain. In general, this makes interfacing to all other domains possible. Currently, a Z-Wave-to-Bluetooth domain interface for cross domain device manipulation is not planned in the context of RADIO. Therefore, no specification for this type of interface is required.

In order to analyse data from Z-Wave devices, an interface between the Z-Wave domain and the WIFI/LAN domain is required. For this interface, it is important to not only be able to control the type of data that is transferred outside of the RADIO Home, but also to enrich raw data with metadata needed for analysis, such as metadata regarding the functionality and position of the collecting device. A summary of the Z-Wave cross domain specifications is given in Table 2.

3.3 WIFI/LAN domain

The WIFI/LAN domain is only present between the gateways, the router and the robot platform, see Figure 2. As has already been described in the preceding sections, a cross domain interface between WIFI/LAN and Bluetooth as well as WIFI/LAN and Z-Wave is required for the functionality of the smart home. Since the WIFI/LAN domain is directly connected to the router, no further domain crossings have to be considered when WIFI/LAN domain devices send data to the IoT platform. However, the robot platform which is connected via WIFI to the gateway network is able to access all data from each gateway in the smart home. This enables the robot platform to access the data from every Bluetooth and Z-Wave device in the smart home, thus extending the potential data processing capabilities of the robot platform. Currently, the robot will only access the data which is relevant for the robot to complete its defined tasks. This setup will enable simple functionality extensions within the smart home should they become necessary. A summary of the WIFI/LAN cross domain specifications is given in Table 3. As far as WiFI/LAN - Bluetooth interaction respective information is as described in section 3.1.

Table 3: Specification for the cross domain interfaces for the WIFI/LAN domain

WIFI/LAN Domain Specification		
Cross domain interfaces	**WIFI/LAN – Z-Wave**	**WIFI/LAN – Bluetooth**
Necessity	required	required
Participating entities	Smart Home gateways	Robot platform Smart Home gateways
Information exchange	position, type of device, functionality, payload	None position, type of device, functionality, payload

4 Fast and Energy Efficient Data Processing

There are two types of data which are going to be processed in the RADIO system:

- Streaming data: This is high throughput data which comes from continually receiving the output of a microphone (audio stream) or a camera (video stream).
- Event/measurement data: This is event or control-like data with relatively small size, collected by sensors or the detection mechanisms. Event/measurement data can also be the outcome of an algorithm which analyses streaming data, e.g. processing of video can lead to the generation of an "exit" event if the camera looks towards the door.

The various interfaces described in previous sections are using networking stacks targeting to transfer events, measurements or commands with emphasis on low power wireless connectivity and minimal usage of the RF spectrum. This makes them not suitable for conveying real time streaming data like audio or video. Our view in RADIO is to perform the processing of the multimedia workloads locally in the robot (either in the main robot processing engine or in the FPGA platform).

In general, the RADIO approach heavily relies on the collection and processing of audio and video streams for analysing and recognising activities of daily life (ADL). Recognizing the emotional status of patients and the identification of emergency situations, such as the detection of falls, are also of high importance. As a result, the main processing engine(s) of robot must be designed as a power-efficient architecture for streaming data processing. In brief, the goals of the ROBOT processing engine(s) are to:

- use and interact with the –already described- wireless interconnection infrastructure,
- be capable of performing the recognition, navigation and reporting tasks in real time, and
- be energy efficient both in terms of energy consumed to transfer the data and in terms of energy consumed during data processing and analysis.

The following diagram (Figure 10) illustrates the on robot processing elements and their interfaces. As the figure indicates, the bulk of processing is performed on the FPGA platform as this is the point where sources of audio (microphone) and visual (camera) streams are located. These two sources of high-content streaming data are directly connected via wired links to the on-board memory of the robot's FPGA (in this way the memory transfers are minimized).

The FPGA platform of the robot will be a Xilinx PicoZed with a Zynq-7000 all programmable System on Chip (APSoC[2]). It includes an ARM Cortex A9 dual core processor (equipped with a Neon co-processor) interfaced with programmable logic allowing high flexibility and performance. In other words and as it will be further analyzed below, the FPGA platform combines a processing element that executes algorithms at a software environment (the ARM processor) and a processing element that can efficiently accelerate demanding tasks in reconfigurable logic (the FPGA itself).

[2] http://www.robotnik.eu/, date of access: August 2015. 19. Xilinx Inc. „Zynq-7000 All Programmable SoC Overview", DS190 (v1.8), 2015.

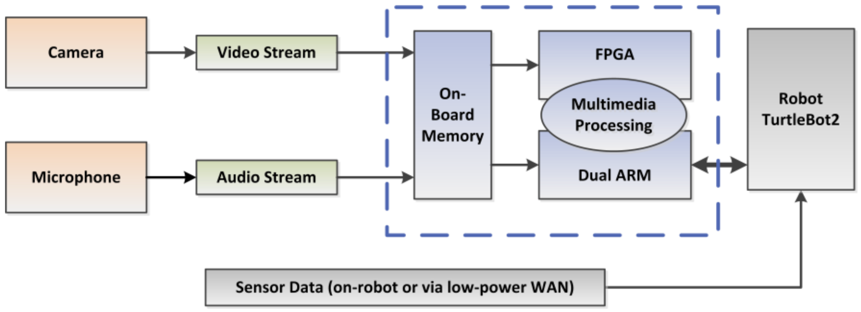

Figure 10 A high level view of the on robot processing nodes

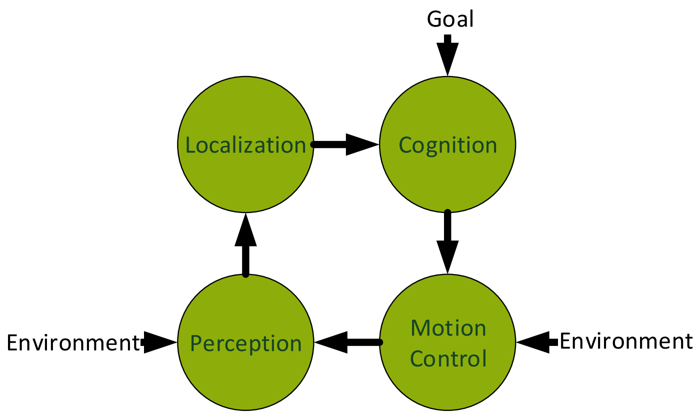

Figure 11 The four task phases of an autonomous robot

The camera and audio data streams will be continuously monitored by the processing elements of the FPGA platform and when activity is detected the corresponding algorithms (which can analyse and recognise the activity) will be triggered. Depending on the specific combination of algorithms that get triggered, some or all computational tasks may be executed in the processor (ARM cores) or accelerated with fixed logic or reconfigurable hardware components inserted in the FPGA reprogrammable logic.

4.1 Robot Platform

A TurtleBot2 provided by ROBOTNIK is used to control the robot. Out of the box, the TurtleBot2 features odometric sensors for accurate odometric measurements as well as bump, cliff, and wheel drop sensors. One advantage of the TurtleBot2 is the open source philosophy of the underlying firmware allowing an easy integration into different frameworks such as the RADIO environment. Since the robot should act autonomously and not be remotely controlled, the robot has to perform several compute intensive tasks perpetually. This continual loop is shown in Figure 11. The four main task areas, need to be handled by the robot, are cognition, motion control, perception, and localization. The cognition phase is responsible of planning a path or a sequence of movements towards the robots current goals. The motion control task is responsible for the physical interaction with the environment. The robots perception task is required in order to process all available sensor information and let the robot gain a sense of its environment. Lastly, the localization task tries to determine the robots location inside the current environment and its current internal state in order for the following task to adapt to the current situation the robot is in.

4.2 Managing the RADIO Computation Platform using Software Analysis Tools

As noted, a PicoZed FPGA platform of Xilinx Inc. will act as the computational heart of RADIO. This FPGA will be responsible to manage the multimedia workloads (image, camera, sound, and speech workloads) targeting to extract specific characteristics from the observed elderly people. In addition, information captured by the various communication nodes of AAL environment will be also conveyed to FPGA for further processing and constellation. The various multimedia and communication workloads must be executed simultaneously in order to shape a global and centralized view of all the components of AAL. Most importantly, the mentioned workloads must be processed in real-time further increasing the computational burden posed to FPGA platform.

To be able to deal with the above requirements, we must take advantage of all processing capabilities of the PicoZed platform. In particular, our view is not to “see” the PicoZed as a typical FPGA acceleration mean, but as an effective heterogeneous multicore platform. Under this direction, three different types of processing nodes are co-allocated in a PicoZed platform. Those are, the dual core ARM processors, the Neon co-processor, and the FPGA programmable logic itself. Operating this apparent heterogeneous system in tandem formulates, inter alia, a mapping-allocation problem. As a first step, we must extract specific and dedicated characteristics of each workload and second to dispatch each workload (or part of a workload) to the appropriate computational node for processing.

Of course, the problem is more complex since the programmable FPGA logic is a “morphable” computation resource without predefined computational capabilities. In any case, the first step in managing efficiently the underlying heterogeneous system is to formulate a toolchain of profiling, visualization and software analysis tools. The target will be to extract specific characteristics from the executing applications as a whole (even from the kernels and/or program phases of the these applications) that will allow us to i) make safe mapping-resource allocation decisions and ii) guide to the extent possible the upcoming FPGA implementation phase. Interestingly, there are a plethora of available profiling tools both open-source and proprietary. After carefully reviewing the characteristics of the majority of those tools, fortunately, we end up to a set of open-source tools that exhibit all the required aspects. Of course, there are intrinsic drawbacks and limitations in each tool, e.g., if a tool is based on instrumentation or sampling.

As a result, a group of profiling tools is needed and the process of selecting what tool must be used to extract a particular application characteristic (e.g., computational vs. memory bound phases of an application) is challenging. Based on our analysis, the combination of valgrind[3], oprofile[4], and vampir[5] offer all the necessary characteristics to assist us through the envisioned direction (the latter tool will be used for visualization purposes). The combination of those tools allows to effectively extracting the ILP (instruction-level-parallelism) properties, the instruction dependencies, cache and memory requirements both at function and instruction level of target applications and most importantly in a nice graphical representation. Exercising and operating those tools in an integrated manner is one of our current activities.

4.3 Identification of High Priority Data Processing Tasks and Hardware Task Scheduling

The goal of the software profiling tools mentioned in the previous section is to guide the development and allocation phases during the implementation of a specific AAL related algorithm. Apart from this, we have to take into consideration that a plethora of AAL-related algorithms will be run

[3] Valgrind Developers. http://valgrind.org, date of access: August 2015.

[4] OpenSource project. http://oprofile.sourceforge.net, date of access: August 2015.

[5] GWT-TUD GmbH. https://www.vampir.eu, date of access: August 2015.

simultaneously in the RADIO platform. As a result, an identification and categorization of the executing algorithms must be performed. High priority algorithms must be immediately executed or not be pre-empted by other tasks. For example, if an elderly has fallen, this event must be recognized in real-time. On the other hand, there is a set of algorithms that do not require high response time. For example, the outcome of bathing or closing related activities may be known after a relatively large delay e.g., after 5 or 10 seconds.

Moreover, the identification of high priority analyses algorithms will also define the allocation of the processing tasks between the ARM processor and the FPGA in order to end up with an optimal balance between the computation speed and energy efficiency. Another open issue is related to the hardware task scheduling in the FPGA re-programmable logic which is explained below.

Since the main processing unit of the robot platform is a PicoZed with a Zynq-7000 APSoC, two different types of scheduling have to be considered. The first handles the traditional scheduling of processes on the available hardware resources. The second deals with scheduling hardware tasks on the programmable logic. These hardware tasks can interpreted as dedicated hardware resources which are specialized in executing one specific task very efficient and fast. However, the services these specialized tasks provide are not required continuously during runtime of the system. This results in very inefficient area usage of the FPGA when the whole hardware configuration is considered to be static. The decision which task should be executed on which processing platform needs to be determined by a scheduler. However, this method requires information about all available tasks and their designated priority. Therefore, dynamic hardware task scheduling has to be considered when designing the functionality of the robot. The robot should only provide the necessary acceleration hardware for the currently executed tasks. If another task needs to be performed, the processing platform of the robot should then automatically reconfigure its hardware to suit its current needs. This behavior needs to be regulated through hardware task scheduling and it is also associated to the assigned priority of the new task.

The scheduling method has to handle several conflicting design goals such as real time constraints, resource efficiency, and energy efficiency. As already mentioned, an autonomous robot needs to continually perform the tasks cognition, motion control, perception and localization. All other tasks have to be either computed during one of these task phases or in parallel to these phases. Therefore, the first step would be to analyze the maximum number of parallel threads the processing unit has to maintain for acceptable responsiveness. Then, some tasks require direct reaction while others do not have such a high priority. These different attributes of tasks need to be considered by the hardware task schedulers while maintaining a certain degree of fairness. The next step is to create a sequence with all relevant tasks with their dependencies.

This task sequence has to be analyzed and possible parallelization possibilities need to be identified taking into account the reconfiguration overhead as well. Based on the possibility of unrolling several iterations of the continual task sequence several different task graphs can be determined. Based on the defined application constraints, such as real time constraints, maximum number of parallelism, resource usage, the best task graph and the resulting sequence have to be determined by the task scheduler.

4.4 Concept within the Distributed RADIO Environment

As shown in Figure 12, apart from the FPGA and the robot platform, the RADIO Home environment contains also the gateway that connects the RADIO Home with remote elements of the RADIO ecosystem, such as the IoT Platform. The main goal of the RADIO Home Controller gateway is to act as the single point of communication between all the different network technologies that coexist in the RADIO Home environment and the remote elements of the ecosystem.

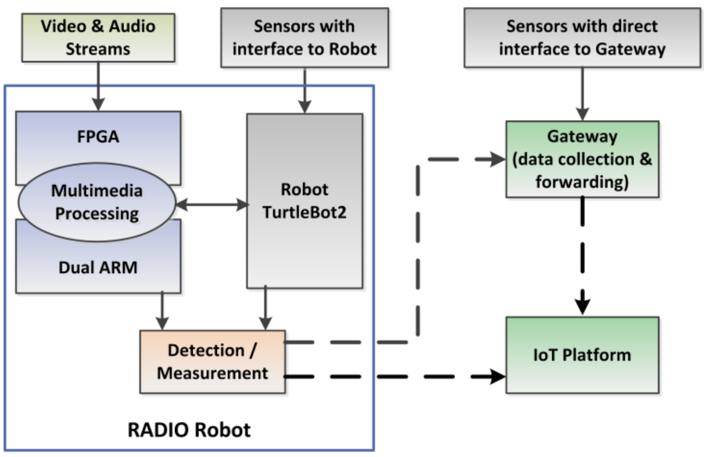

Figure 12 Diagram illustrating the data flow between the ROBOT, gateway and IoT platform

The overall RADIO ecosystem will be specified and developed in the context of WP5. At this point, it suffices to say that the ecosystem must role of the IoT Platform is to provide to other nodes of the RADIO Ecosystem the relevant clinical data.

On the contrary, the processing of the computation intensive video and audio tasks and the extraction of the ADL activities will be performed solely in the FPGA platform (either in the ARM processor or in the re-programmable logic). The RADIO Home Controller gateway and the IoT Platform will be responsible to manage event-like or control-like data i.e., to convey communication messages between the various nodes or processing engines within the RADIO environment. The reason of this decision is that the FPGA is the point where the sources of audio (microphone) and visual streams (camera) are located (the transmission of high-content streaming data is eliminated).

As it is also illustrated in Figure 12, distributed sensors are supposed to be directly interfaced to the gateway. However it is possible that there will also be sensors connected to the robot as well. Either because of the communication technology they employ or because their input is needed by the streaming data analysis algorithms. The following table relates the various data processing and transfer interfaces to the available domains.

Table 4 Data Transfer and Process in relation to interfacing domains

Data	WIFI/LAN	Z-Wave	Bluetooth
Sensor data needed for analysis of audio and/or video streams		(optional)	X
ADL and mood recognition event log generated by the analysis of streams	X		
Sensor data that can be directly forwarded for remote processing		X	Optional
Robot location and status data	X		

5 Conclusions

This deliverable aims to describe the architecture for extending the smart with the robotic platform and defines the initial interconnection and interface specifications. It specifies which smart home components require which network protocol interfaces and what kind of data will be exchanged between the respective components. The Z-Wave components are only connected to the respective RADIO Home Controller gateways which in turn serve as a central data hub for all protocols. The BLE devices have the possibility to connect to the robot platform and the RADIO Home Controller gateways. This sort of architecture could potentially allow manipulating devices across protocols e.g., a BLE device controlling a Z-Wave device.

Another aspect that is discussed in this deliverable is the connection of two masters to the same BLE device, since both the robot platform and the RADIO Home Controller gateways have the possibility to be connected to the BLE devices. In the current architecture, the robot and the RADIO Home Controller gateway do not share their respective devices. This is due to the fact that the BLE-Beacons only serve as localization information source for the robot. This information is not needed by the smart home gateways since their positions are fixed within the smart home. Additionally the BLE-Beacons data do not have to be sent to the IoT Platform via the robot since the robot processes and provides already analysed data, like current position and task, to the IoT Platform for further analysis.

The backbone of the smart home architecture is the WIFI/LAN interconnection between the router, robot, and RADIO Home Controller gateways. The router enables the communication with the IoT Platform and the WIFI/LAN infrastructure enables the information exchange between each component. In the first step of RADIO, no raw data from any device are sent to the IoT Platform, but only pre-processed data. This reduces the communication overhead and provides more flexibility when expanding the basic functionality of the RADIO architecture. These specifications are needed in order to ensure energy efficient and fast data processing within the smart home environment without sacrificing functionality.

There are two potential processing platforms in RADIO’s smart home architecture. The first is the RADIO Home Controller gateway and the second is the robot platform. The RADIO Home Controller gateway has a general purpose processor as processing unit which executes C/C++ software. The processing unit of the robot platform is an APSoC with a processing system and a FPGA programmable logic. This enables the robot to perform computation intensive tasks very efficiently and very fast. However, in order to extract the most performance out of this processing unit, all tasks that need to be executed by the robot platform have to be analysed and scheduled accordingly on the processing system or on the programmable logic. Furthermore, all hardware tasks cannot be available all the time on the programmable logic. This challenge must be solved with a hardware task-scheduling scheme. These topics have been discussed and presented in this deliverable. While still subject to change, the following Tasks 4.2 and 4.3 can reference this deliverable for design parameters and general communication architecture.