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 4.2

Architecture for extending smart homes with robotic platform II

Dissemination Level	**Public**
Due Date of Deliverable	Project Month 18, September 2016
Actual Submission Date	6 April 2017
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	NCSR-D, 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	07 Sep 2016	First draft, establishing document structure and work allocation. Writing Introduction and outlining the content of the first chapter.	RUB
02	21 Sep 2016	Contribution of S&C to section 2.3 "Z-Wave Devices Interface"	S&C
03	29 Sep 2016	Contribution of AVN and TWG to chapter 4 "Fast and Energy Efficient Data Processing: Progress"	TWG, AVN
04	30 Sep 2016	Merging of all contributions and writing conclusion	RUB
05	16 Oct 2016	Internal peer review	NCSR-D
06	3 Feb 2017	Addresses peer review comments	AVN, TWG, RUB
07	5 Apr 2017	Added Section 7 about Main Controller and components relevant to consuming the events recognized by WP3 methods.	NCSR-D
08	6 Apr 2017	Internal peer review of Section 7	TWG
Fin	6 Apr 2017	Addressing review comments, final preparations, 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 is the physical architecture of the RADIO Home, covering 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 RADIO Robot platform, the Smart Home devices, and the Main Controller 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 with 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 Robot on-board computer, or other computers on the premises which comprise the *RADIO Home*, i.e., the part of the overall RADIO system that is deployed within a single home. In addition, Task 4.1 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 and sensor positioning 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 (or even pure software) solutions. The programmable solutions offer more flexibility to provide several dedicated services to the end-users through software updates or extensions.

We have extended the work done in D4.1 for this task as follows:

- We analysed the physical communication architecture for each of the communication groups and identified problems with BLE and WiFi communication interfaces on the Robot platform (Section 2).
- No changes were conducted in the interconnection domains (Section 3).
- We started initial measurements for the specified computational nodes (FPGA and general-purpose computer) in order to determine which task needs to be mapped onto which platform. We also specified a behavior concept for the Robot in order to enable an ultra-low power mode.
- We set forward two scenarios in which the input camera is either located physically at NUC or at the FPGA. This will enable us to investigate different power savings modes based on user behaviour.

- We presented an ADL based scenario in which the autonomy of the Robot can be significantly increased when the camera is attached directly to the FPGA board.

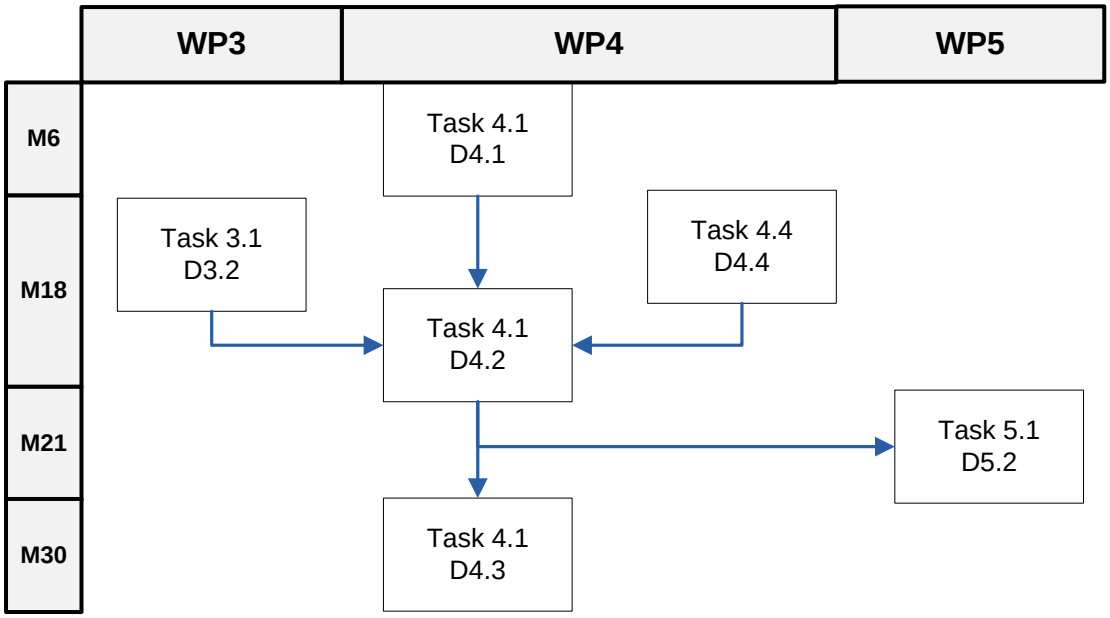


Figure 1: Relation to other Work Packages and Deliverables

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 and task mapping policies among all the processing units of the RADIO ecosystem in order to extract the best solution in terms of latency, power consumption, and area.
- **AVN** and **TWG** will investigate system-level power savings modes taking as input the user behavior targeting to increase the autonomy of the Robot in terms of battery charges.

1.3 Relation to other Work Packages and Deliverables

This document is the second in a series of closely related deliverables. The intermediate version due in M18 (September 2016) is used to drive the development in Task 5.1. From M19 until M30, this is a living document updated to record adjustments necessitated as developments 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 updated and extended in deliverable D4.1. 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: PROGRESS

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.

2.1 Interconnections in the Smart Home

As introduced in deliverable D4.1, the smart home is comprised of several devices with different communication protocols. Figure 2 shows all available devices within the smart home environment and their respective interconnections.

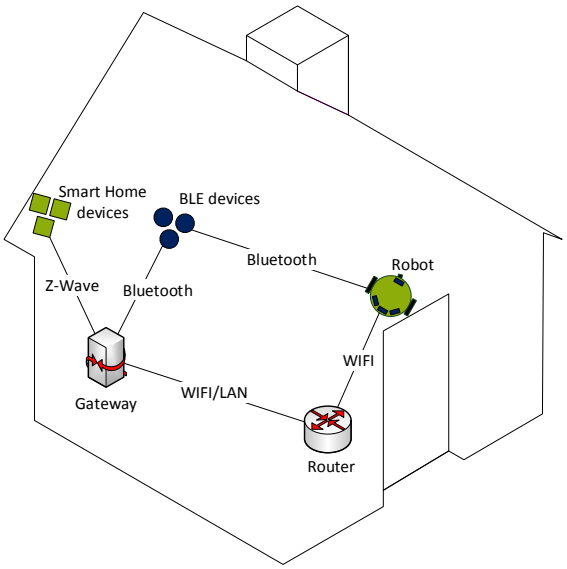


Figure 2: Device interconnection within the smart home environment

2.2 Robot Interface Design

As already mentioned in D3.1, a TurtleBot2 provided by ROBOTNIK[1] is used as Robot platform. Since the Robot should act autonomously and not be controlled remotely, it needs to be interconnected to the smart home environment. The Robot is outfitted with two processing platforms: an Intel NUC and an Avnet PicoZed. The NUC is responsible for controlling the base platforms sensors and actuators. Therefore, it is directly connected to the TurtleBot2 base platform via USB. The Avnet PicoZed serves as accelerator platform to reduce the computation load of the Intel NUC. Additional devices that are placed on the robot are an Asus Xtion Pro camera and a Hokuyo laser scanner. These two devices can either be connected to the Intel NUC or to the Avnet PicoZed.

As depicted in Figure 2, the Robot is expected to support two wireless communication interfaces. BLE is required to access and locate BLE devices. In deliverable D4.1, BLE devices were described as being placed at fixed positions. This concept was extended by also allowing mobile BLE devices. These BLE devices, such as a remote control for the TV equipped with BLE technology, can be located by the Robot platform. For localization, the *received signal strength indication (RSSI)* value of the desired BLE device is used. The Intel NUC supports two wireless communication interfaces, the Bluetooth Low energy (BLE) and the WiFi interface. BLE connectivity is used for localization tasks performed by the Robot, while the WiFi interface is required to connect the smart home environment to the IoT platform. The setup of the Robot platform is also depicted in Figure 3.

[1] ROBOTNIK Automation S.L.L. http://www.robotnik.eu/, date of access: August 2015

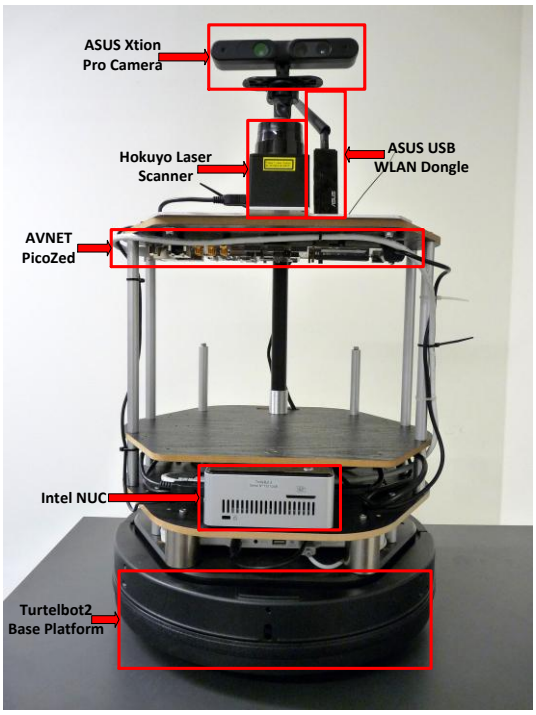


Figure 3 RADIOs Robot platform outfitted with camera, laser scanner, the two processing platform and the USB WLAN dongle.

Initial measurements have shown that the usage the BLE and WIFI transceiver simultaenously results in degraded RSSI performance of the BLE transceiver. This is because the localization task requires accurate readings of the received signal strength indication values of the respective BLE devices. Because both interfaces are placed on the same network chip, the Wifi interface interferes with the BLE interface and vice versa. Additionally, both signals use the ISM band around 2.4GHz so that spectral overlapping of the signals can occur. This can results in strong signal interference. Exemplary measurements are shown in Figures Figure 4, Figure 5, and Figure 6.

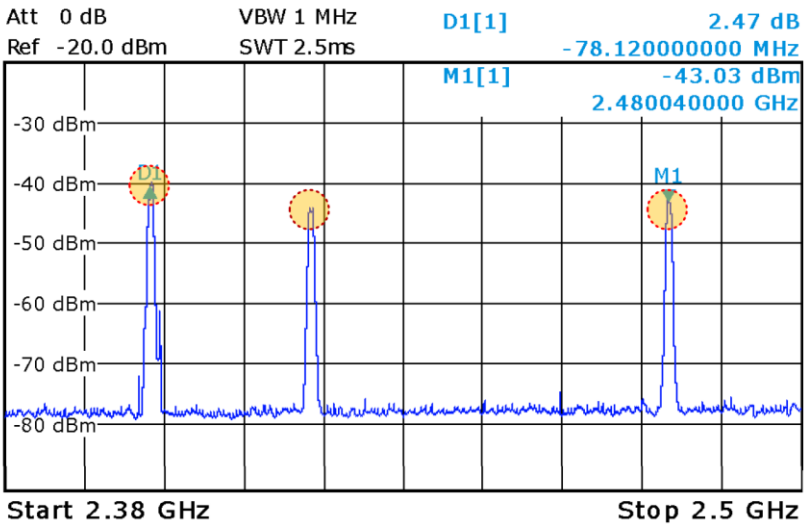


Figure 4 Spectrum BLE

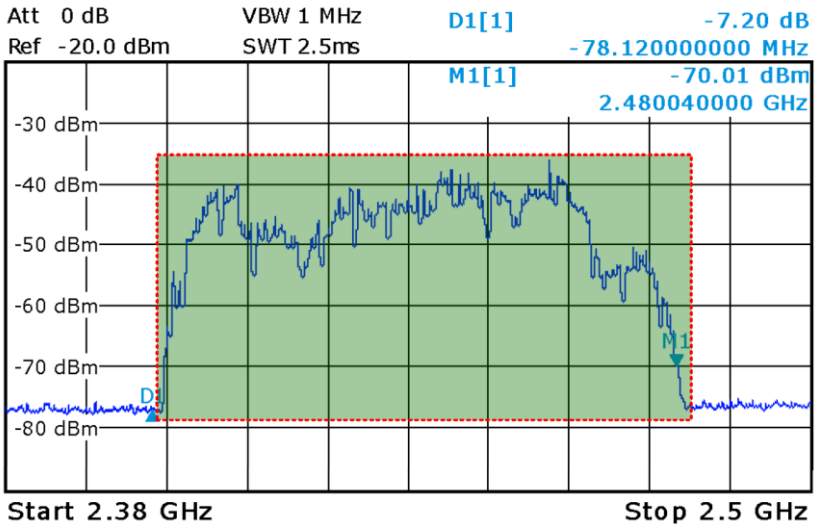


Figure 5 Spectrum 2.4 GHz WLAN

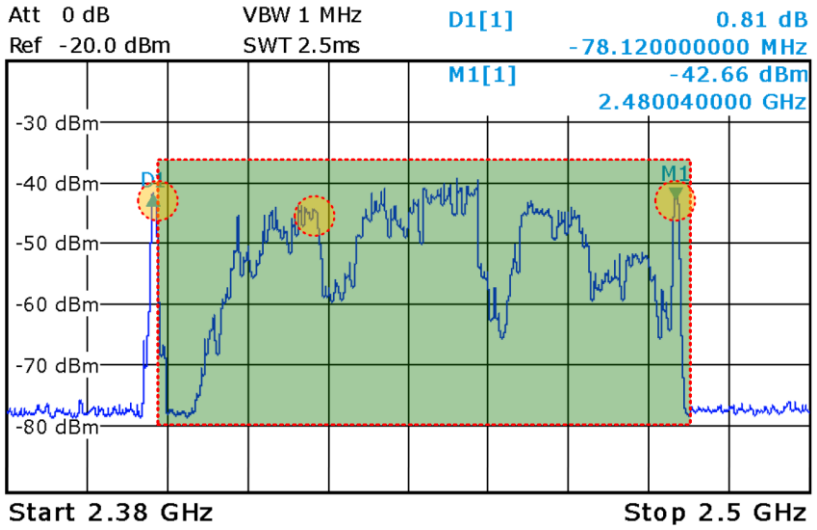


Figure 6 Spectrum BLE and WLAN

The figures show the signal strength in dBm over the frequency in GHz. Figure 4 depicts the spectrum of the BLE broadcaster. The three peaks show the frequency band of an advertising channel of the spectrum. They are highlighted in yellow. The mean value of the singal strength equals to -41.3 dBm. In contrast, Figure 5 shows the spectrum over 2.4 GHz 802.11 b/g/n wireless LAN transceiver. Its region of influence starts at 2.395 GHz and ends at 2.484 GHz and is hihglighted in green. The mean signal strength equals to approximately -44 dBm. The combination of both signals is shown in Figure 6. Based on the measurements shown in Figure 4 and Figure 5, an overlapping of the signals can be seen in the frequency band. Especially the advertising channel 38 is located in the spectrum of WLAN. In order to reduce this interference, an external Wifi USB interface is used and connected to the Intel NUC, see ASUS USB WLAN Dongle in Figure 3.

2.3 Z-Wave Devices Interface

As explained in the deliverable D4.1, the access to Z-Wave devices is done through a wireless network managed by the controller. The RADIO Home Controller was the only point for the communication between the Z-Wave devices and external services through the RESTful API provided by the cloud-based IoT platform.

In this updated integrated prototype, the RADIO Home Controller is capable of communicating smart home devices with external services without using the cloud-based IoT platform. We have implemented a fully local RESTful API that allows that client functions can access data structures in Z-Wave and execute per devices and handling network management functions. This allows that client functions can be integrated within the smart home application by running in the own controller.

The local RESTful API implements the whole control logic of the Z-Wave network. The two main functions are:

- Management of the network. This includes adding and removing devices and managing the routing of the network. In the Z-Wave terminology all these functions are called 'function classes'. They are functions offered by the controller itself.
- Execution of commands through 'command classes' that allows the control of Z-Wave device functions. Command classes offer the variables and the commands according to the abilities of the respective devices. Command classes consist of two types of commands: commands for users (most of them are "GET" and "SET") and commands for configuration.

Most of the Z-Wave commands can be controlled by the local RESTful API but only a small subset is accessible for external applications for security reasons.

The basic Z-Wave commands accessible for external applications are:

Table 1: Z-Wave commands

Z-Wave command	**Type of Class**
sendData	FUNCTION_CLASS
AddNodetoNetwork	FUNCTION_CLASS
RemoveNodeFromNetwork	FUNCTION_CLASS
setValue	FUNCTION_CLASS
SetNodeLocation	FUNCTION_CLASS
SetNodeName	FUNCTION_CLASS
toggleActuatorSensor	COMMAND_CLASS
toggleDimmableSensor	COMMAND_CLASS
setThermostatSetPoint	COMMAND_CLASS
setThermostatMode	COMMAND_CLASS
setThermostatFanMode	COMMAND_CLASS

3 SPECIFICATIONS ON INTERFACING THE DIFFERENT DOMAINS

This chapter deals with the challenge of transferring data through several different protocol domains. The main points addressed in this chapter have been described in deliverable D4.1. No changes have occurred in this topic.

4 FAST AND ENERGY EFFICIENT DATA PROCESSING: PROGRESS

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 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).

Error! Reference source not found. illustrates two different scenarios of positioning the Robot processing elements and their interfaces. In both cases, the bulk of processing is indented to be performed on the FPGA platform. However, the two scenarios differ in the points where the sources of audio (microphone) and visual (camera) streams are located. The advantages and disadvantages of each scenario are analysed in the rest of this section. In any case, the camera and audio data streams will be continuously monitored 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 NUC, in the Zynq ARM processor, or accelerated with fixed logic or reconfigurable hardware components inserted in the FPGA reprogrammable logic.

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

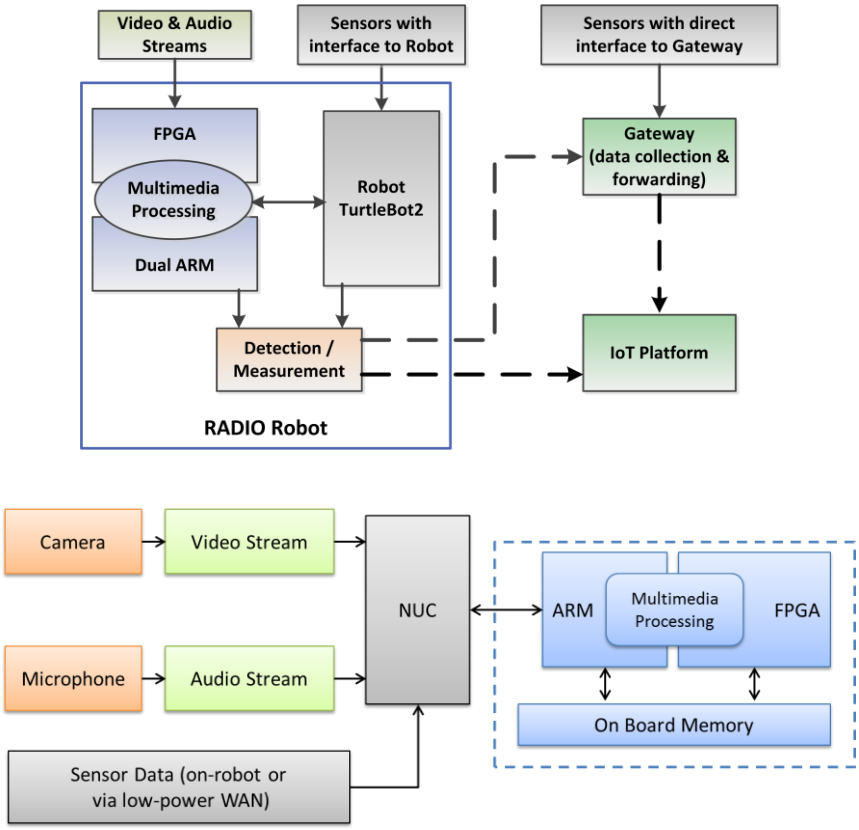


Figure 7 Diagram illustrating two different scenarios of the data flow between the ROBOT, gateway and IoT platform

In the second scenario depicted in Figure 7, the camera and audio data streams, as provided through the NUC, will be continuously monitored by the processing elements of the FPGA platform. Therefore, a pipe that brings the streams down on the FPGA platform is implemented on the NUC using ROS channels. To avoid overloading the interface with unnecessary memory transfers, the ROS channels are constructed so that image and audio data is provided to the FPGA platform on a need-to-know basis. When the captured image data are transferred to the FPGA device (through ROS messages), the RAM which resides on board will be directly accessible from both the dual ARM core and the FPGA fabric (to minimize on-board memory transfers).

To provide a specific example of this concept, we describe here the implementation for the algorithm used in ADL "Measure time to get out of bed". Data processing with this algorithm consists of following (simplified) steps:

- For each Frame
  - Store frame in RAM to be used as "previous" in next iteration
  - Split Frame in blocks of 9x9 pixels
  - Compare blocks with "previous frame" blocks to detect difference
  - Get bounding rectangle of all different blocks
  - Get centre of all different blocks
  - Check bounding rectangle and centre against the set limits
  - Report any events / findings

To optimise these steps, we modify the execution order and assign to various blocks on the above architecture as follows:

- For each frame (row 1 to row 469)
  - Get next 9 rows [Camera -> Video Stream -> NUC -> ARM]
  - For each row:
    - Split in 9x9 blocks [ARM->OnBoardRAM]
    - For each block:

```
        • Read 9x9 [OnBoardRAM->FPGA]
        • Compare and store "previous" [FPGA]
        • Report per-block result [FPGA->ARM]
    o Calculate Bounding and Centre [ARM]
    o Report to NUC [ARM->NUC]
```

This simplified example illustrates how a method can be adapted to fit the specific requirements of the architecture, achieving optimized memory transfer, speed, and power consumption.

The potential benefits inherently offered by the first scenario (illustrated in Figure 7) is further analysed below.

4.1 Concept within the Distributed RADIO Environment

The need for fast and efficient processing of the data streams in the distributed RADIO environment comes from the need to be:

- Responsive: in some cases the data collected on the Robot and the measured / detected quantities and events are correlated centrally with data from the smart home sensors or other sources. E.g. detection of existing the room can be correlated with a motion sensor mounted on the room walls.
- Efficient, in terms of energy consumption: as the Robot relies on battery, the processing of data should be limited to what is necessary and executed on this node which provides the lowest overall power consumption.

To put this into perspective, an example state-diagram of the Robot is presented in Figure 8.

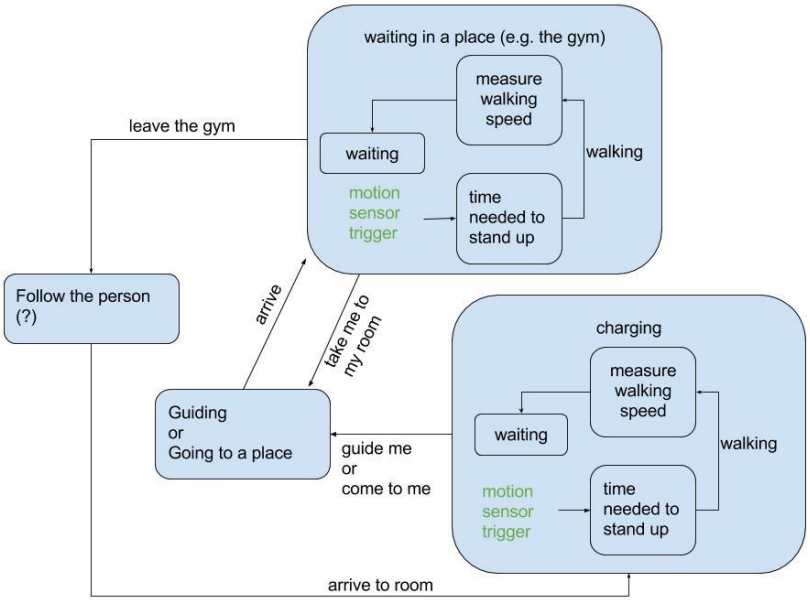


Figure 8 State diagram highlighting different (power) modes

In this diagram, we have two points (in green fonts) where entering is triggered by the smart-home infrastructure through getting the event from a motion sensor. There are also a number of states that involve heavy data processing and we have to ensure that this is either:

- very low power
    - time needed to stand up (gym)
    - measure walking speed (gym)
    - follow the person
    - guiding or going to a place

or

- executed when the Robot is stationed on its charging station
    - time needed to stand up (in the room)
    - measure walking speed (in the room)

5 THE NEED FOR ENERGY EFFICIENT EXECUTION

5.1 RADIO Robot Energy Usage Profile

The RADIO project depends on methods for detecting ADL, and wireless transfer of the detected information to other stations in a RADIO house. This requires many subsystems to be active even during long inactive periods of the persons, leading to unnecessary energy consumption for processing and transferring data. Due to the limited power of any battery –besides energy consumption – this also poses the risk of the Robot being out of energy when there is an opportunity to recognize interesting events. To minimize this risk, dedicated processing elements for ADL detection are designed. These elements use only the minimum energy required to preprocess sensor data, keeping all other subsystems in low-power mode and waking up the Robot only when preprocessing shows interesting activity.

Many components of the RADIO system are supplied power by being connected to main supplies. However, there are two types of components which have to reside on battery power:

- The wireless BLE beacon devices
- The Robot itself

The beacon devices employ Bluetooth 4.0 Low Energy techniques, achieving a very long battery life. They are commercially available components so there is little room (and little need, anyway) for further improving their energy efficiency.

The RADIO Robot on the other hand is a unit which has many energy-hungry subsystems:

- Main processor to control Robot movement (CPU)
- FPGA to accelerate ADL recognition methods
- Sensors, especially the image sensor (camera)
- Mechanical subsystem (motors)
- Wireless subsystem (network)

The RADIO robotic system is moving and staying in locations where access to charging cannot be guaranteed, especially in the case of home care installations. To achieve its goals, the Robot must be able to operate at minimum power for preserving battery charge effectively extending its autonomy.

If all subsystems are always active, the radio Robot will need to recharge itself every few hours. This is feasible in a controller environment but results in long period of Robot non-availability. So we had to understand how each subsystem is used and if it indeed needs to be active at each use case. The following table provides, an overview, assuming that Robot activity can be classified in the following states:

- Waiting: at this state, the Robot is not moving, neither is it processing sensor data. It will be triggered by some external event.
- Moving: when leading the way or following a person.
- Monitoring: at this state, the Robot is not moving but it is processing sensor input in order to detect some ADL or understand user's mood.

For some of these states, there is a difference on whether the Robot is on its charging station or away in another room (Table 2).

Energy consumed at each state by each subsystem is not the same. For example, the CPU while waiting can be clocked at lower frequency, drastically reducing the required power. Also, sensors and FPGA can perform only basic data capture and processing when monitoring away from the charging dock, and revert to full-power processing when this power is available. Although a number of such techniques are

used, their impact on power consumption is not drastic. To cope with this problem, our view is to develop dedicated hardware components that allow the Robot to turn-off complete subsystems in some cases; turning them on only on demand and just for the short period when they are needed. The goal is to have an improved energy profile (Table 3).

Table 1: Robot Subsystems Energy Usage

State	CPU	FPGA	Sensors	Motors	Network
Waiting	Used	Not used	Not used	Not used	Used
Moving	Used	Used	Used	Used	Used
Monitoring/Away	Used	Used	Used	Not used	Used
Monitoring/Charging	Used	Used	Used	Not used	Used

Table 2: Improved Energy Profile by using dedicated HW

State	CPU	FPGA	Sensors	Motors	Network
Waiting	On demand	Used	Not used	Not used	On demand
Moving	Used	Used	Used	Used	Used
Monitoring/Away	On demand	Used	Used	Not used	On demand
Monitoring/Charging	Used	Used	Used	Not used	Used

More details for each case are presented below:

Waiting State: The FPGA can connect only to a BLE scanning device. When the user or any other RADIO system wants to instruct the robot, it should first connect to this device, and send a handshake command. This command is interpreted by the FPGA. E.g. it can be used to turn-on the CPU and perform a simple action. If more complex control is needed, e.g. a user request via the tablet GUI, the CPU will turn on the Network subsystem.

Monitoring/Away State: At this state the FPGA gets triggered by external events or continuously monitors live sensor signals. Only when some (external or sensor) activity occurs, the HW component in the FPGA will pre-process it and decide whether the CPU or/and the Network subsystem has to be turned on.

Monitoring/Charging and **Moving States:** At these states we may not need to employ any on-demand approach for the CPU and/or the Network. However, having the dedicated hardware components in the FPGA will allow some of the processing to be offloaded there, which also yields modest energy benefits.

5.2 A typical dedicated HW Component

The key concept of a dedicated HW component is a specially designed circuit:

- It is implemented in FPGA for configurability and future upgradability
- It should be connected directly to the other subsystems
    - In the prototype phase of the RADIO project, these direct connections will be emulated by interface functions between the FPGA and the CPU, however to achieve all benefits in a widely deployed system these will have to be replaced with actual hardware links

- The component is processing signals from sensors, so that simple decisions on whether other subsystems have to be employed or not can be made

A typical such component has the following parts:

- Triggering mechanism, which initiates sensor data capture and processing
- Local Memory, which hold processed sensor data so that the main system RAM does not have to be used
- Signal processing acceleration functions in FPGA
- Control interfaces to turn-on and notify (or get notified by) other subsystems

5.3 Example ADL Use Case

To prototype and experiment with the alternative approach discussed in this report, we selected a small number of ADLs as target use cases for the monitoring state of the robot. The selected ADL recognition methods must:

- Be able to be detected when the Robot is not moving
- Have a significant part of pre-processing, which can be done on the FPGA
- Get triggered by some external signal or some very simple HW-only method
- Not rely on information exchanged over the WiFi or Smart-home Network

The selected methods will be the ones which detect:

- The time that is needed by the patient to get out of bed. This ADL is based on image processing algorithms that observe the patient while getting out of bed. The image processing algorithms can be parallelized availing themselves for the acceleration within the FPGA hardware. This algorithm divides the image into different regions. If the center of mass of moving pixels over succeeding images lies in one of these defined regions, an event is triggered. Thus, this algorithm is able to detect if a person is sleeping, awake but not going out of bed and awake and standing up.
- Picking up medication cups. The image processing methods used to detect this ADL benefit from the acceleration through the FPGA hardware as they rely on a complex algorithm.

The acceleration does not involve the complete method; but rather focuses on early detection of a high-possibility for an event so that SW-based processing can be invoked. Specifically, for the above mentioned ADLs:

- For the time-to-stand-up ADL, the hardware component will collects and calculates data from all regions, providing a trigger to software when a given activity threshold is crossed.
- For the cup-detection ADL, since this is manually triggered by the operator, hardware accelerations is not related to the recognition but to the stabilization and centering of the image. It has been observed through field trials that the Robot can slightly move while waiting; a movement that might create false positives. An always running HW component will be monitoring such small movements and constantly re-center the view.

5.3.1 Optimizing for Power

In order to determine the SW-HW co-design of the FPGA-ARM system, extensive profiling of the image processing algorithms is needed (see Section 6). Generally, FPGAs provide high-performance when manipulating the images pixel-wise or in small blocks. This allows several hardware implementations with different degrees of parallelism. Each hardware design then must be evaluated within the overall RADIO framework in order to determine the best implementation.

During the 1st period, work was allocated to have a working HW acceleration framework, and not to the specific optimization of each HW component. In this version of the report, we profiled three options,

so that the expected benefits of various optimization approaches can be quantified, allowing to focus on these solutions that will yield the most benefits.

The three analyzed options – as described earlier– are:

- No offloading, all processing is performed on the robot's main processing unit (NUC)
- Offload on embedded ARM core of the FPGA (no HW acceleration)
- Offload on dedicated low-level hardware blocks in the FPGA (ARM core can be power down)

5.3.2 Profiling on daily activity use cases

To make a realistic profiling, we identified typical activity use cases with the help of non-technical partners. Each typical activity is depicted as a combination of five states for the Robot subsystem:

- Moving, where the Robot is actually moving and uses its motor, sensors and camera
- Looking, where the Robot is waiting for an event to be triggered by what it can see
- Sensing, where the Robot is using its onboard sensors or communicates with smart home
- Processing, where heavy processing to analyze sensor and camera input is required
- Idle, where the Robot is on but is doing nothing of the above

It is important to understand that a specific human activity (e.g., having lunch) will combine more than one of the above states (e.g., looking, sensing, and processing).

By accumulating the energy needs at each activity, we take a daily activity profile in terms of energy consumption:

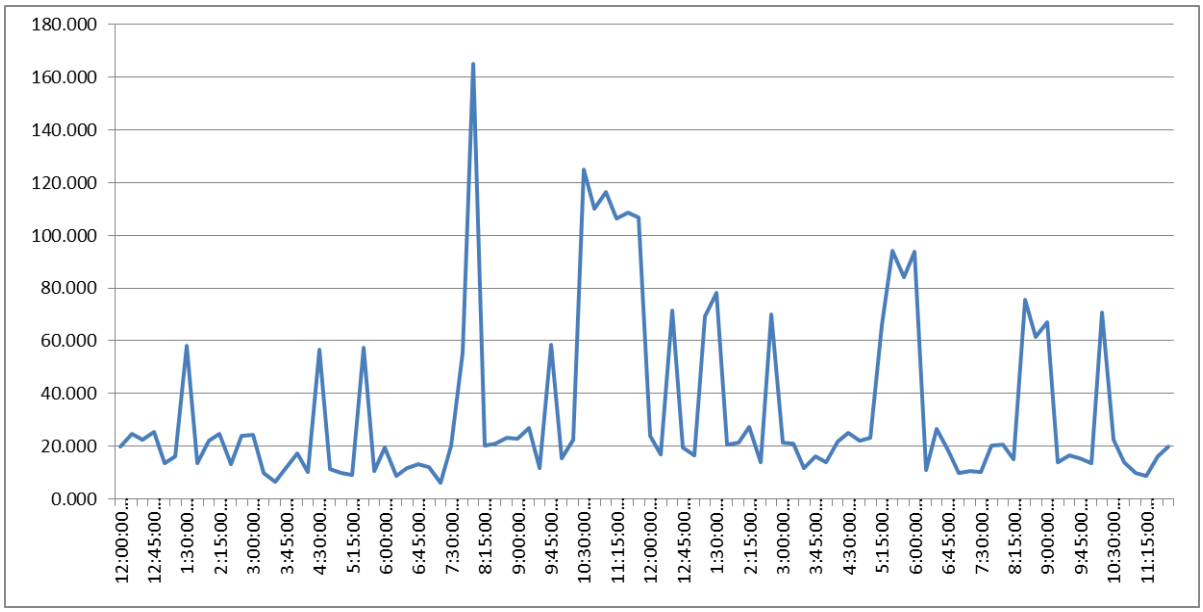


Figure 9 Dialy activity profile

The three options listed in section 5.3.1 were then tested on these profiles. The main reason for doing this activity was to identify if there is potential for maximizing battery life, i.e., reducing the number of needed re-charges during one day – thus we added a battery load calculation in our results:

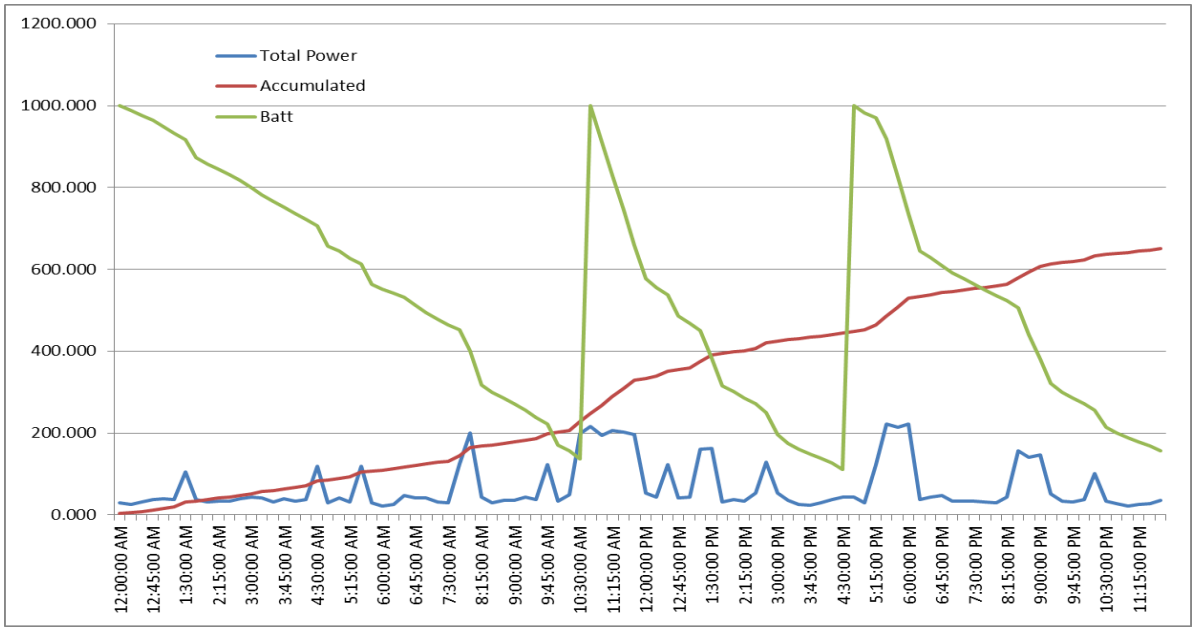


Figure 10 No offloading, all processing is performed on the robot's main processing unit (NUC)

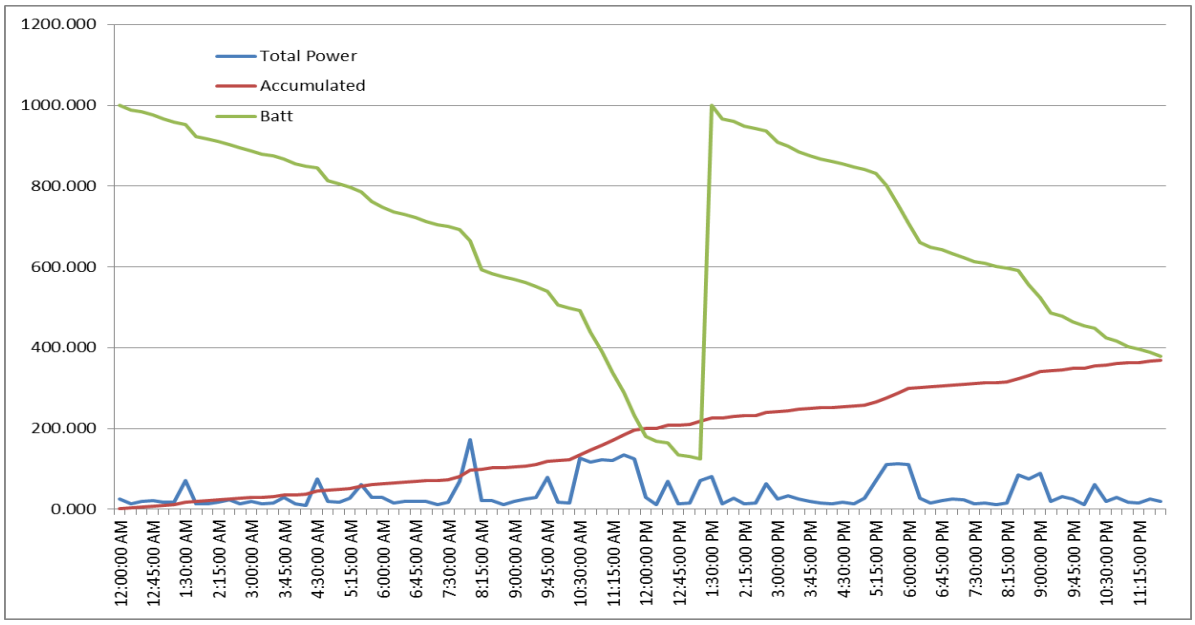


Figure 11 Offload on embedded ARM core of the FPGA (no HW acceleration)

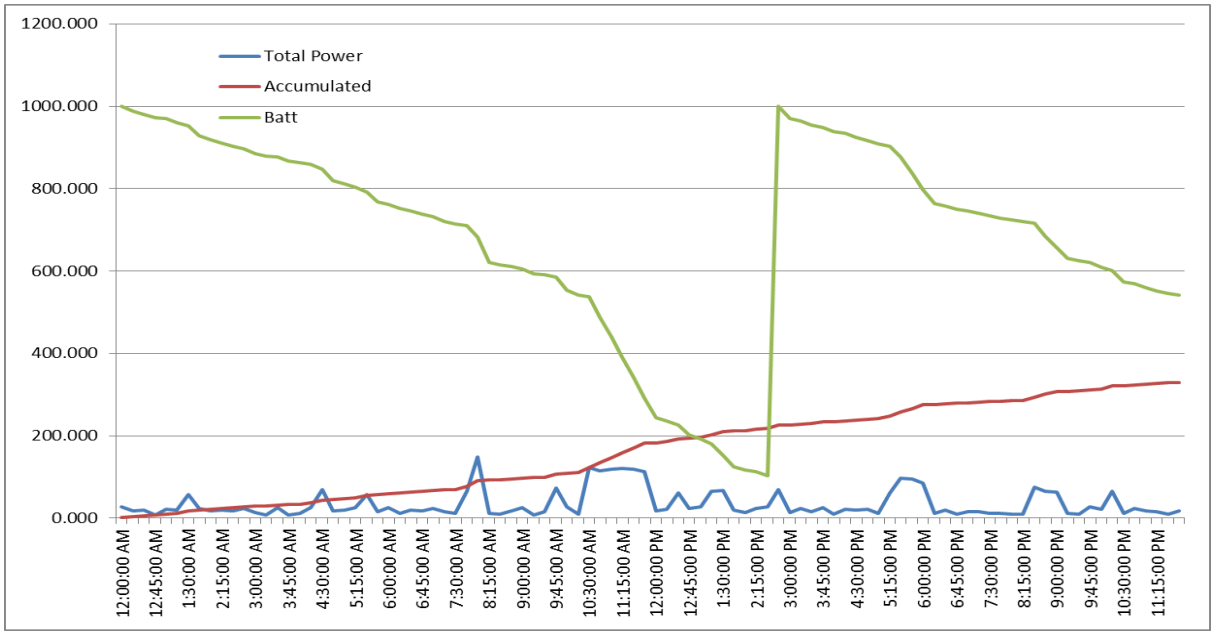


Figure 12 Offload on dedicated low-level hardware blocks in the FPGA (ARM core can be off)

The results show a very promising opportunity. Next steps therefore are to use the created hardware acceleration framework and finalize the acceleration HW blocks. For each of the ADLs selected in 5.1 we break down the steps of the algorithm, their SW-only implementation, their SW-HW implementation and a HW-only preprocessing part that will be used to trigger SW processing only when needed (further details are provided in Section 6).

The final version of this report (due M30) also will report on whether the choices we have made are indeed provided this benefit.

5.4 Long-term Integration

The RADIO project is a research oriented work, aiming not to a final product but to the exploration and analysis of all aspects that come with the project's proposal. When this comes to deployment, further work will be needed to effectively integrate the developed dedicated HW components in the robot. Towards this direction, perhaps the most useful directions would be:

(a) to develop a direct connection between the camera and the FPGA so that images do not have to be copied by the CPU. This will allow much more extensive pre-processing in the FPGA while the CPU is not consuming any energy at all.

(b) to design a special smart-home node device which would bring notifications from smart-home sensors directly on the FPGA. This would allow all other subsystems to be at sleep mode, until some external event triggers activity.

6 MANAGING THE RADIO COMPUTATION PLATFORM USING SOFTWARE ANALYSIS TOOLS

To deal with the above computational requirements, we must take advantage of all processing capabilities of the PicoZed platform. 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 apparently heterogeneous system in tandem formulates, inter alia, a mapping-allocation problem.

As a proof of concept scenario we showcase this CPU-FPGA interplay using the well-known DCT (discrete cosine transform) kernel. DCT is a widely used kernel in many signal processing applications e.g., jpeg compression algorithm. The kernel mainly consists of two parts: the 2D transformation (Task A) and the transpose (Task B) part. The two parts work in a pipeline fashion. As a first step, we gather the latency of both tasks on the ARM CPU and the FPGA. In the first case, the highest available CPU frequency is assumed. For the second case, the Vivado HLS tool with the default optimization parameters is used. Our tentative results are shown in Table 4.

	CPU	**FPGA**
Task A	1,93ms	0,45
Task B	0,09ms	0,86

Table 4: Latency and power profiling of DCT's tasks.

As shown, the two tasks exhibit different performance characteristics. Task A gets the full advantage of the parallel ALUs (FPGA DSPs), while Task B prefers CPU due to smarter cache systems (spatial prefetching + normal / sequential prefetchers). However, these trade-offs might change if we include more parameters in our analysis. For example, if we utilize the NEON coprocessor on ARM or the two ARM cores are used in parallel. On the other side, if smarter FPGA address generators are designed the situation for Task B may change as well. Furthermore, if we extract the power figures (in the ARM CPU and in the FPGA) for the two tasks, a different trade-off will be formulated.

In general, what really complicates the problem is that the FPGA logic is a "morphable" computation resource without predefined capabilities. Modeling this "morphable" computation resource is proven to be quite challenging due to the large design space. Fortunately, Xilinx has released a framework that is suitable for our purposes. The framework is called SDSoC and currently we are in the phase of exploiting its capabilities in taking the full capacity of the Zynq based heterogeneous multicore platforms.

7 THE RADIO MAIN CONTROLLER

7.1 Architecture

The RADIO Main Controller is the main orchestrator of the behaviours of the RADIO Home and the main keeper of the information collected and analysed by the various RADIO Home systems. Its functionalities include:

- System orchestration
- Bridging between the different sub-systems
- Storing and serving ADL recognition results

The Main Controller is (physically) partially distributed between the home computer and the robot computer, via a dual-ROS core architecture (Figure 7). This adds integration complexity compared to the earlier single-ROS core RADIO architecture, but address the problem that:

- The Main Controller would be unable to operate with the robot turned off or having run out of battery, if the robot's computer executed the only ROS core process in the system.
- The bandwidth-hungry communication channels between the sensors and the perception modules would have to use the wifi, if the home computer executed the only ROS core process in the system.

7.2 Orchestration

The action and node manager orchestrates the overall system, including reacting to user initiatives through the user device and initiating automated actions, except for home automation directly handled by the S&C suite.[3] Orchestration is implemented by sending control messages and by starting and stopping ROS nodes. The node manager also monitors ROS node execution to re-start nodes that have crashed.

Action and node management is distributed between two nodes:

- The main node that executes at the home computer:
  https://github.com/RADIO-PROJECT-EU/radio_node_manager_main_controller
- The robot-side node that executes at the robot's on-board computer:
  https://github.com/RADIO-PROJECT-EU/radio_node_manager

The main node relegates to the robot the distribution of control messages for the ROS nodes executing on the robot. Only the main node is required for the operation of the overall RADIO Home, so that functionalities not related to the robot remain active even if the robot is off-line or turned off.

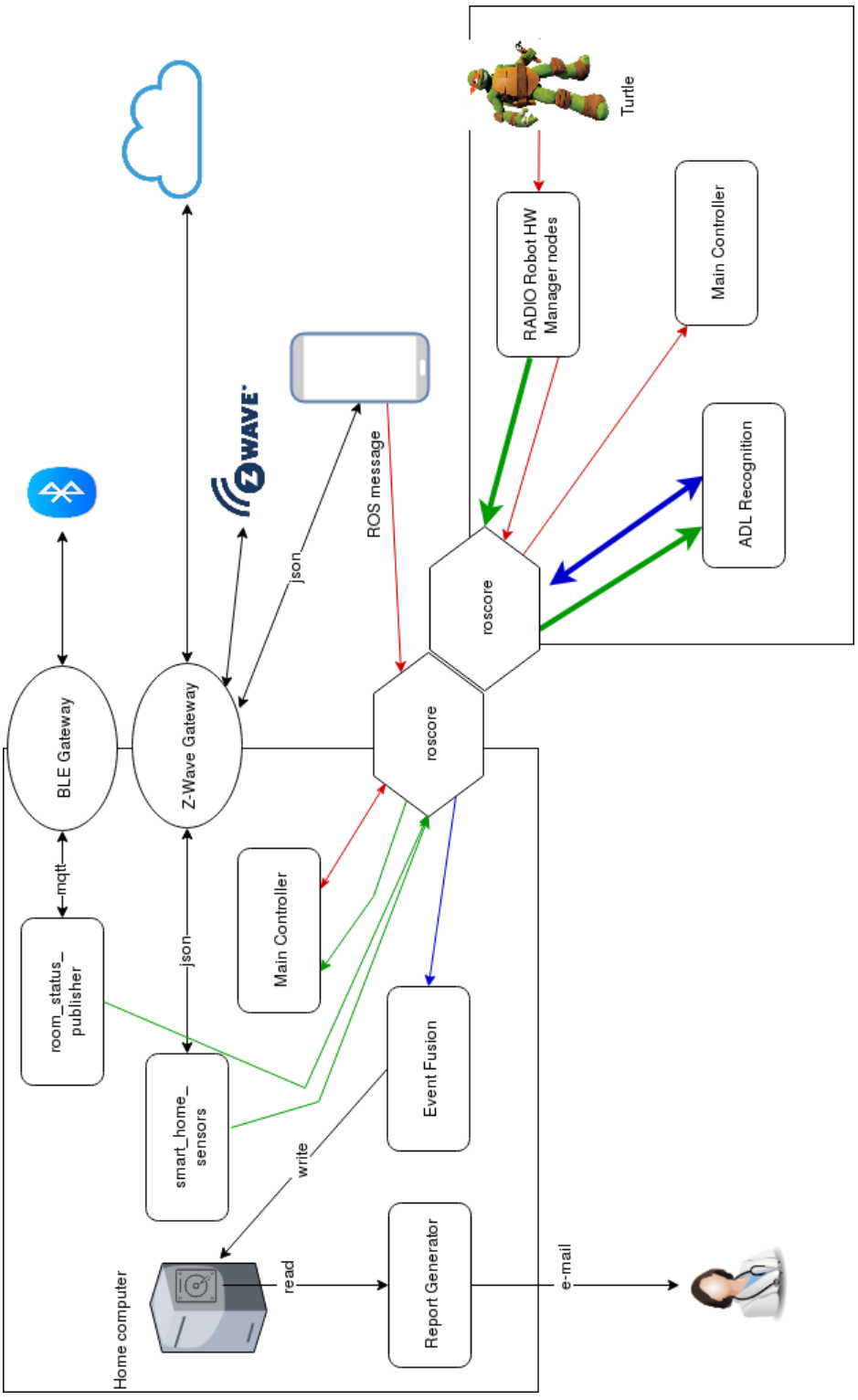
7.3 ZWave and MQTT Network Bridges

The Main Controller bridges between the ROS middleware/wifi network and two other communication infrastructures present in the RADIO Home:

- The REST API to the ZWave network of home automation sensors and actuators, via the S&C Gateway:
  https://github.com/RADIO-PROJECT-EU/snc_sensors_publisher
- The MQTT middleware used by the BLE network:
  https://github.com/RADIO-PROJECT-EU/room_status_publisher

These components bridge between networks by simultaneously being REST/ROS client and MQTT/ROS client, respectively.

[3] The cloud-based S&C rule engine that implements pre-configured automations and the EnControl GUI for monitoring sensors, see also D5.5 User Interfaces.



The colour of the arrows indicates the type of data: red is used for control signals, either user or system initiated; green is used for raw data and primary perception results; blue is used for secondary/high level ADL recognition results; black is used for communication that this relevant to the Main Controller, but not directly accessed by the Main Controller. The thick arrows imply more voluminous data than the thinner ones.

Figure 7. Interconnections between the Main Controller, Turtlebot, and the home automation components.

7.4 ADL Recognition Wrappers and Report Generator

The ADL Wrappers are a collection of ROS nodes that are aware of the RADIO Home database schema and of the semantics of the ROS messages published by the ADL recognition methods. These wrappers listen to the ADL recognition methods and make, where necessary, calculations such as extracting a duration from an event marking the start of an ADL and the matching event marking the completion of the ADL. These wrappers output to a temporary, short-term database. This database is used by the Report Generator to compute the daily or other aggregations that need to be reported and stored in the long-term database.

- Wrapper for walking pattern recognition in rage data (D3.4, Section 2): https://github.com/RADIO-PROJECT-EU/hpr_wrapper
- Wrapper for visual recognition of motion events (D3.4, Section 4): https://github.com/RADIO-PROJECT-EU/motion_analysis_wrapper
- Wrapper for moving object tracking (D3.4, Section 3) and classification (D3.5, Section 2): https://github.com/RADIO-PROJECT-EU/ros_visual_wrapper
- Report Generator: https://github.com/RADIO-PROJECT-EU/radio_report_generator

Similar wrappers will also be developed for the acoustic event recognition method (D3.5, Section 3) and for the rules that extract events from the home automation sensors (D3.5, Section 4).

Table 5: Access levels and authentication for the RADIO Home database

Component	Access Level	Authentication	Explanation
RADIO Home components	Write access	Only accessible from the internal RADIO Home network	The RADIO Home components that recognize events update the event log.
Report Generator	Read access	SSL-based authentication.	Read access for the formal caregiver of this specific home, using conventional authentication and access control mechanisms.
Notification Generator	Read access	Only accessible from the internal RADIO Home network	Filters the data for events that trigger notifications.
RASSP	Read access	Only accessible from the internal RADIO Home network	The RADIO privacy-preserving data mining component accesses all data to respond to queries that observe the RASSP Protocol (cf. D5.6). Access through RASSP guarantees that these responses allow statistical aggregates to be computed over many RADIO Homes without revealing the values of any one of these Homes.

7.5 Data Services

The main requirements that must be satisfied by the technologies used originate from the nature of the stored data and the nature of the consumers of those data. The output of the analysis algorithms (cf. Section 5.2, D3.3 *Conceptual Architecture*) is the log of the recognized events annotated with the type of the event, the actual time and date that the event occurred, and the duration or other measurement associated with the event, if any.

Since the recognized events are recurrent this log forms essentially a set of time-series for each event type. A time-series database is a database that is optimized for handling time series data, promoting time as a first-class citizen and implement time-based operations in a more efficient way.

This database needs to provide access as foreseen in Table 5.

We used the InfluxDB database management system,[4] an open source scalable time-series database that targets use cases that heavily use time-based metrics and sensor data in the IoT context. The current RADIO data schema contains a *measurement* (i.e., a database table) that includes all the higher-level events produced by the RADIO Home analysis algorithms. This measurement has the following fields:

- **event_type:** the type of the event recognized, as tagged by the recognition algorithms, such as “4m-walking”, “Sitting-to-Standing”
- **time:** the timestamp at which the event was recorded
- **duration:** the duration of the event, if applicable

[4] Cf. https://www.influxdata.com

8 CONCLUSIONS

This deliverable expands the architecture and the interconnection and interface specifications of deliverable D4.1. The Robot interface design has been updated based on additionally information regarding the Robot platform and initial tests with WiFi and BLE connectivity. The Z-Wave devices are now able to be accessed through the RESTful API in the Home Controller gateway.

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. The interfacing between the different communication protocols has been defined in D4.1. No changes have been made to these definitions and therefore the information provided in D4.1 is up-to-dated.

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 Robot has two processing units. The first processing unit is an Intel NUC and the second processing unit is an APSoC with a processing system (ARM-based) and a FPGA programmable logic. In this deliverable, further details have been provided regarding the distribution of processing tasks between both processing units. Additionally, this deliverable provides insights on how to decide which tasks benefit from an implementation on programmable hardware. Two examples with measurements are given showing that not every algorithm benefits from a hardware implementation. In order to reduce this large design space, high-level synthesis tools such as Vivado HLS or SDSoC are used. Finally, a general behaviour concept is presented in this deliverable, which allows the Robot to switch between extremely low power states and active state when necessary based on user behaviour. The goal in this case is to increase to the extent possible the autonomy of the Robot measured in terms of battery charges.