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 4.9

Integrated smart home with robotic platform extensions II

Dissemination Level	**Public**
Due Date of Deliverable	Project Month 30, 30 September 2017
Actual Submission Date	9 May 2018
Work Package	WP4: *Physical home architecture development and integration of cost-effective, reliable and power-efficient RADIO components for elder monitoring and caring*
Task	T4.4: *Smart home design and integration*
Lead Beneficiary	S&C
Contributing beneficiaries	NCSR-D, TWG, ROBOTNIK
Type	DEM
Status	Submitted
Version	Final

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

Abstract

This deliverable is the RADIO Home prototype, a set of hardware devices and the corresponding software. The prototype demonstrates a complete RADIO Home, but without its interconnections with other RADIO Homes or other elements of the RADIO Ecosystem.

History and Contributors

Ver	Date	Description	Contributors
00	27 Oct 2017	Document structure, assuming D4.8 as a starting point.	NCSR-D and S&C
01	11 Jan 2018	New software components and updated versions of software components already in D4.8.	NCSR-D
02	2 May 2018	Demonstration video produced and published.	NCSR-D and TWG
03	5 May 2018	Detailed demonstration descriptions.	TWG
Fin	9 May 2018	Review, final preparations and submission.	NCSR-D and S&C

Abbreviations and Acronyms

NCSR-D	National Centre for Scientific Research “Demokritos”
TWG	Technical Educational Institute of Western Greece
RUB	Ruhr Universitaet Bochum
ROBOTNIK	Robotnik Automation SLL
S&C	Sensing & Control Systems S.L.
AVN	AVN Innovative Technology Solutions Ltd.
FSL	Fondazione Santa Lucia
FHAG	Fundació Hospital Asil de Granollers
FZ	Frontida Zois
ADL	Activities of Daily Life
BLE	Bluetooth Low Energy
CPS	Cyber-Physical Systems
ICT	Information and Communications Technology
ROS	Robot Operating System

CONTENTS

LIST OF FIGURES

1 INTRODUCTION

1.1 Purpose and Scope

This deliverable demonstrates the first RADIO Home prototype, integrating smart home sensing and automation with the RADIO Robot. Within the scope of this deliverable is to demonstrate the hardware prototypes and to publish the source code of the software developed for the Main Controller, the network gateways, and other software needed for integration purposes.

1.2 Approach

This deliverable is prepared within *Task 4.4: Smart home design and integration*. This task uses the architecture and components developed within the previous tasks in WP4 and complements them with commercially available smart home and home automation devices in order to design and integrate the specific smart home environment that will be deployed for the RADIO pilots. This task also implements the Main Controller that orchestrates the overall RADIO Home system and the gateways needed to make existing solutions interoperable. Work in Task 4.4 is in tandem with work in Task *4.1 Designing device interconnection and interfacing*, to update the physical architecture, a living document that documents the integrated system as it evolves (D4.3).

1.3 Relation to other Work Packages and Deliverables

This deliverable is prepared following the *Architecture for extending smart homes with robotic platforms II* (D4.2) and updates its predecessor (D4.8) with components developed within WP4 (D4.5 and D4.7). This integrated prototype is documented by the final version of the architecture, D4.3.

This deliverable will be integrated into the final WP5 prototype, D5.9.

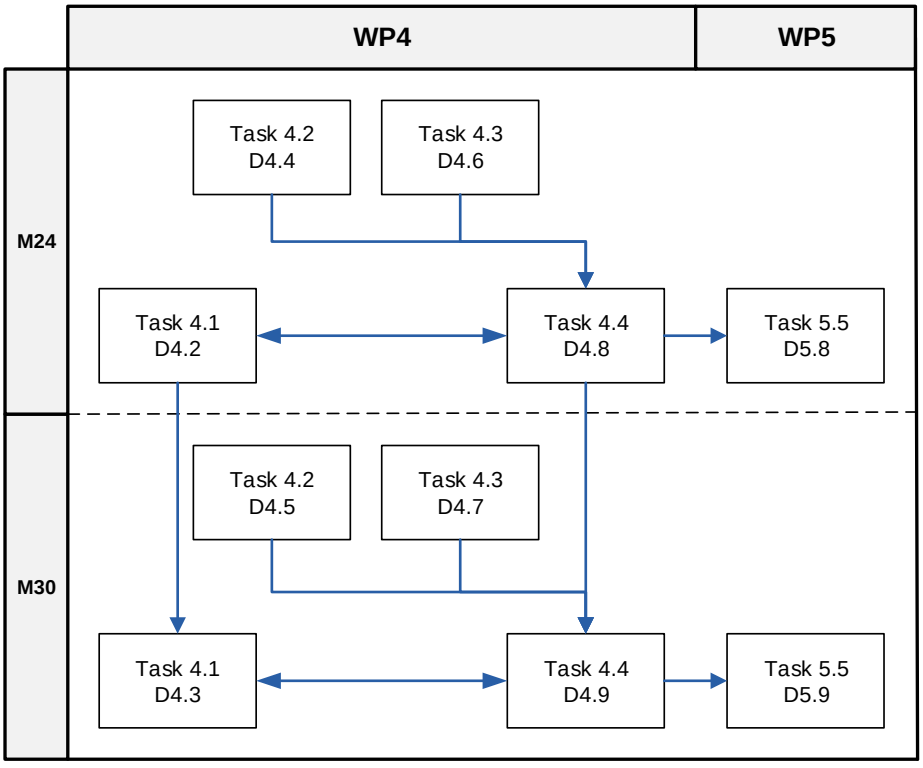

Figure 1: Relation to other Work Packages and Deliverables

2 PROTOTYPE HARDWARE

The final RADIO Home prototype integrates the hardware components listed below.

Integration base	Component
ROS/Wifi	Final RADIO Robot prototype [D4.7].
BLE	BLE Beacons and tags for localization.
ZWave	Multisensor 6 for general comfort sensors (e.g., temperature).
ZWave	Presence sensor and pressure sensor for detecting presence and using furniture (chairs, bed).
ZWave	Magnetic sensors for detecting door opening.
ZWave	Smart switch for controlling lights
ZWave	Smart motors for controlling blinds.
ZWave	Smart plugs for controlling and detecting usage of electrical appliances (kettle, microwave, TV, etc.)
ZWave	Cooktop power consumption sensor for detecting usage of cooker
ROS/Wifi/BLE/ZWave	The RADIO Home Cluster, a stack of three RaspberryPi devices, executing the Main Controller and the gateways to the BLE and ZWave networks.

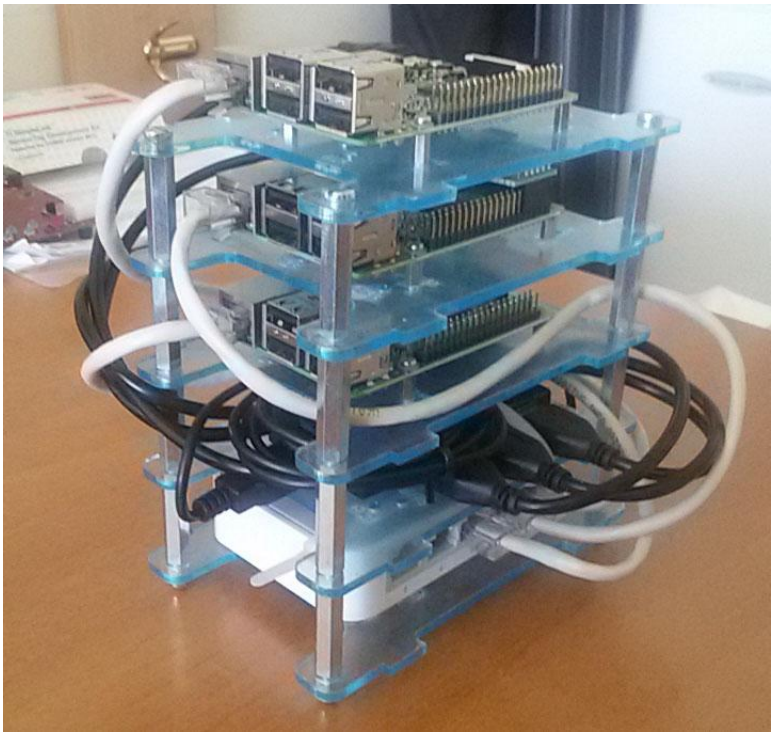

Figure 2: The RADIO Home Cluster

Figure 2 depicts the integrated Home RADIO Gateway, comprised of the following components:

- Main Controller: Raspberry pi 3 taking advantage of the on board WiFi wireless interface
- ZWave Gateway: Raspberry pi 2 which used the onboard WiFi wireless interface in conjunction with an integrated Z-Wave wireless interface
- BLE Gateway: Raspberry pi 3 using the onboard WiFi and BLE wireless interfaces.
- Ethernet switch

Effectively the this is a self-contained Raspberry cluster, requiring only one Ethernet connection to connect all components to the internet. The cluster is very efficient, with all components powered through an active 4-port USB3 hub, with a maximum power consumption of 18W.

3 PROTOTYPE SOFTWARE

3.1 Robot Packages

The robot software stack, delivered as D4.7 *Integrated robotic platform II*, is repeated here for completeness.

Package name and description	Source code repository and release used for this deliverable	
turtlebot: The basic drivers for running and using a TurtleBot with ROS. Forked from the official Turtlebot repository and modified and adapted for the RADIO Robot, creating a new model of the robot and sensors.	https://github.com/radio-project-eu/turtlebot	v1.1
turtlebot_apps: A group of demos and examples used as templates for TurtleBot/ROS packages. Forked from the official Turtlebot repository and modified and adapted, creating configuration files specific to RADIO localization and navigation.	https://github.com/radio-project-eu/turtlebot_apps	v2.0
kobuki: Software controllers for the Kobuki mobile base. Forked from the official Turtlebot repository and slightly modified for the RADIO Robot.	https://github.com/radio-project-eu/kobuki	v1.0
kobuki_core: Software controllers for the Kobuki mobile base. Forked from the official Turtlebot repository and modified for the auto-docking procedure of the RADIO robot.	https://github.com/radio-project-eu/kobuki_core	v1.0
turtlebot_radio_bringup: Bringup files for the robot. It contains all the configuration and launch files to run all the RADIO Robot components.	https://github.com/radio-project-eu/turtlebot_radio_bringup	v1.2
robotnik_msgs: Definition of messages and services used by the core packages.	https://github.com/radio-project-eu/robotnik_msgs	v1.0
turtlebot_radio_emergency: Node that implements safety stop. It manages the emergency button of the robot and disables any movement.	https://github.com/radio-project-eu/turtlebot_radio_emergency	v1.0
marker_mapping: Package to localize ar_track_alvar markers and use them to initialize the localization module with a known pose every time the robot initializes. To avoid manual initialization, this node localizes visual QR markers onto the map and uses them inversely to calculate the correct pose of the robot.	https://github.com/radio-project-eu/marker_mapping	v1.0
HumanPatterRecognition: Recognizes human walking patterns in laser scans and tracks walking.	https://github.com/radio-project-eu/HumanPatternRecognition	v3.0.0

Table continues on next page

Table continued from previous page

HPR Wrapper: Uses HPR output to recognize and time "walked 4m" events.	https://github.com/radio-project-eu/hpr_wrapper	v2.0
ROSVisual: Tracks moving objects in the RGB/depth modality and classifies motion as bed or chair transfer.	https://github.com/radio-project-eu/ros_visual	v2.0
ROSVisual Wrapper: Uses the output from ROSVisual to time chair and bed transfer events and to recognize and time "walked 4m" events.	https://github.com/radio-project-eu/ros_visual_wrapper	v2.0
Motion Analysis: Recognizes motion and classifies it as "bed transfer" and "pill intake" events.	https://github.com/radio-project-eu/motion_analysis	v2.0
Motion Analysis Wrapper: Uses the output from motion analysis to time the bed transfer event.	https://github.com/radio-project-eu/motion_analysis_wrapper	v2.0
AUROS: Recognizes talking, watching TV, listening to music, and doing housework by acoustic analysis of the audio modality.	https://github.com/radio-project-eu/AUROS	v1.0
Map convergence: Consumes robot pose messages from the localization package and converts them to the corresponding coordinates in the RADIO Home model.	https://github.com/radio-project-eu/map_convergence	v1.0

The robot software stack also comprises the following components that are mostly relevant to integrating the robot into the RADIO Home:

Description	Source code repository and release used for this deliverable	
rostune: Helps ROS developers distribute their nodes in the most effective way. It collects statistics for topics and nodes, such as CPU and network usage.	https://github.com/radio-project-eu/rostune	v1.0.7
connectivity checker: Checks the health of the ROSCore processes foreseen by the architecture of a multi-master ROS system (here, on-board NUC, RasberryPi, and FPGA on-board ARM)	https://github.com/radio-project-eu/radio_ros_connectivity_checker	v1.0

Further to these, the packages below have been developed specifically for the purpose of integrating the RADIO Robot into the RADIO Home.

Package name and description	Source code repository and release used for this deliverable	
Node manager: The robot-side node of the Node Manager [D4.2, Section 7.2].	https://github.com/radio-project-eu/radio_node_manager	v1.0
Anti-theft alarm: Identifies being picked up or moved from the robot's IMU, and sends alarm notifications to the RADIO backend system.	https://github.com/radio-project-eu/anti_theft_alarm	
REST/ROS bridge: Software interface between a REST service and ROS MoveBase, used for interfacing the first GUI prototype [D5.4, Section 2.3.2]. This has been deprecated, as the new GUI prototype directly accesses ROS.	https://github.com/radio-project-eu/radio_actions_manager	v1.0

3.2 Main Controller

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. The packages that implement the functionalities of the main controller are listed below.

Package name and description	Source code repository and release used for this deliverable	
Node manager: The home computer-side node of the Node Manager [D4.2, Section 7.2].	https://github.com/radio-project-eu/radio_node_manager_main_controller	v2.0
Node manager: The robot-side node of the Node Manager.	https://github.com/radio-project-eu/radio_node_manager	v2.0
Influx data service: The database that provides the RADIO Home's data services [D4.2, Section 7.5].	https://github.com/radio-project-eu/maincontroller	
Report generator: The aggregator that goes through the events log in the data service to generate ADL reports [D4.2, Section 7.4].	https://github.com/radio-project-eu/radio_report_generator	v1.0

The Main Controller also bridges between the ROS middleware/wifi network and the two other communication infrastructures present in the RADIO Home. The packages that implement these bridges are listed below.

Package name and description	Source code repository and release used for this deliverable	
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	v1.0
The MQTT middleware used by the BLE network	https://github.com/radio-project-eu/room_status_publisher	

3.3 BLE Localization

Besides the robot-side packages listed above, the following packages have also been developed to support BLE localization and composite services over BLE localization.

Package name and description	Source code repository and release used for this deliverable
Multihop localization: Firmware code for the CSR µEnergy® CSR1010™ Development kit. The nodes are distinguished in fixed infrastructure and mobile nodes. Fixed infrastructure nodes inform the BLE Gateway of the objects that are in their vicinity and the value of the RSSI.	https://github.com/radio-project-eu/relative_multihop_localization
BLE Gateway: interface which connects IP networks with Heterogeneous Sensor Networks. Utilizes the MQTT for the communication between cloud applications, gateway applications, and all these applications to communicate with the WSN. Implementing functionalities, like local storages, routing, local data processing, applications execution.	https://github.com/radio-project-eu/ble_gateway
Indoor relative localization: A BLE Gateway application that communicates with BLE Gateway to estimate relative locations of BLE tagged objects. The application receives from the fixed nodes packets that record the surrounding Radio-enabled mobile BLE nodes along with the RSSI.	https://github.com/radio-project-eu/indoor_relative_localization

4 DEMONSTRATION

The first prototype of the RADIO Home was deployed in March 2017 at FHAG premises, in preparation for the first round of the piloting study.

A video is at https://vimeo.com/267641474 demonstrating the following:

- Hardware-based acceleration [00:00 – 01:36]
- Localization [01:37 – 03:38]

Hardware-based acceleration [00:00 – 01:36]: A demonstration of motion detection using two different scenarios: the processing is performed in FPGA (hardware version) vs in the on-board NUC (software version). Specifically, 00:23 – 00:40 shows the software-based approach where images from a conventional camera are captured and processed on the on-board PC. During 00:41 – 00:53 the installation of the python (FMC) camera on the Picozed FPGA is shown so as to showcase the hardware based solution developed in RADIO. Then, 00:41 – 01:09 shows that the processing is performed on the programmable logic of the FPGA by the implemented fixed-logic, hardware accelerators. The actual captured image **is neither stored nor transmitted** and the hardware FPGA accelerator directly outputs recognition results, printed to the terminal for the demonstration. From 01:10 onwards, the power consumption of the two main processing entities is depicted (NUC vs. hardware accelerators). On the left side, the current drawn by the fixed logic hardware accelerator is measured and on the right side the current drawn by the **NUC (in standby mode)** is measured. At 01:21 an event is detected by the FPGA-based hardware accelerator and the NUC powers up. From that point and till 01:36, the power consumption of the NUC spikes and varies significantly. The difference between the latter current measurements and the ones measured in the previous phase is what RADIO platform conserved regarding aggregate current drawn.

Localization [01:37 – 03:38]: A demonstration of robot localization and navigation, and a demonstration of the ability to localize BLE-tagged objects and guide the user to their location. The demonstration also shows the integration of the RADIO system in the Atlas IoT environment for technically validating the system and collecting logs for problem-shooting. Specifically, up to 02:45 various scenarios are depicted of real time monitoring of the robot's movements through a representation of the house premises. ATLAS has the capability of presenting previous data that are relevant to the movement patterns of the end user or/and malfunctions of the robots and in general for debugging purposes (02:45 – 02:58). Additionally, the relative localization capabilities are demonstrated. ATLAS backend infrastructure interacts with BLE devices and localizes specific objects relative to BLE beacons in known positions (e.g. the kitchen area) or/and relatively to other BLE-tagged objects (e.g. the kitchen table). Up to 03:08 the video shows a browser based graphical time line of localization results. Then (until 03:33) the video demonstrates how the robot uses localization information to physically approach the misplaced object and emits an audio signal to notify the user of its position.