The Framework Programme for Research & Innovation
Innovation actions (IA)

Project Title:

Autonomous self powered miniaturized intelligent sensor for environmental sensing and asset tracking in smart IoT environments

AMANDA

Grant Agreement No: 825464

[H2020-ICT-2018-2020] Autonomous self powered miniaturized intelligent sensor for environmental sensing and asset tracking in smart IoT environments

Deliverable

D1.7 Architecture design of the AMANDA system delivered (for both breadboard and integrated/miniaturized system)

Deliverable No.		**D1.7**	
Workpackage No.	**WP1**	Workpackage Title and task type	**System Specifications, Requirements and Use Cases**
Task No.	**T1.5**	Task Title	**Task 1.5: System specifications, Overall Architecture and Design**
Lead beneficiary		**CERTH**	
Dissemination level		**PU**	
Nature of Deliverable		**R**	
Delivery date		**31 August 2019**	
Status		**Final**	
File Name:		**AMANDA_D1.7_Architecture_design_of_the _AMANDA_system-v1.1**	
Project start date, duration		**02 January 2019, 36 Months**	

This project has received funding from the European Union's Horizon 2020 Research and innovation programme under Grant Agreement n°825464

Authors List

Leading Author (Editor)				
Surname		*Initials*	*Beneficiary Name*	*Contact email*
Kouzinopoulos		CS	CERTH	kouzinopoulos@iti.gr
Co-authors (in alphabetic order)				
#	*Surname*	*Initials*	*Beneficiary Name*	*Contact email*
1	Meli	MM	ZHAW	marcel.meli@zhaw.ch
2	Bruetsch	MB	ZHAW	manuel.bruetsch@zhaw.ch
3	Bruelisauer	CB	ZHAW	cornel.bruelisauer@zhaw.ch
4	Bembnowicz	PB	IMEC	pawel.Bembnowicz@imec.nl
5	Kanlis	AK	CERTH	alexkanlis@iti.gr
6	Sideridis	PS	CERTH	sideridis@iti.gr
7	Tzitzios	IT	CERTH	jtzitzios@iti.gr
8	Kauer	MK	Lightricity	matthias.kauer@lightricity.co.uk
9	Pasero	DP	Ilika	denis.pasero@ilika.com
10	Schellenberg	MS	Microdul	martin.schellenberg@microdul.com
11	Vujičić	OV	PENTA	oskar.vujicic@penta.hr

Reviewers List

List of Reviewers (in alphabetic order)				
#	*Surname*	*Initials*	*Beneficiary Name*	*Contact email*
1	Bellanger	MB	Lightricity	Mathieu.bellanger@light-ricity.co.uk
2	De Vos	JD	E-peas	julien.devos@e-peas.com

Document history			
Version	Date	Status	Modifications made by
V0.1	01/06/2019	ToC	CERTH
V0.3	05/07/2019	Contribution from Partners	CERTH, ZHAW, Microdul, Lightricity
V0.4	09/07/2019	Additional contribution	Penta, Ilika, IMEC
V0.6	20/07/2019	Content changes, layout	CERTH
V0.7	31/07/2019	Final draft, submitted for internal review	CERTH
V0.8	15/08/2019	Comments from Reviewers received	Lightricity, E-peas
V1.0	27/08/2019	Final version. Reviews addressed, ready for submission	CERTH
V1.1	08/11/2019	Updated based on comments from the 1st Review Meeting	CERTH

List of definitions & abbreviations

Abbreviation	Definition
ASSC	Autonomous Smart Sensing Card
BLE	Bluetooth Low Energy
CSP	Chip-Scale-Package
CTAT	Complementary To Absolute Temperature
DC	Direct Current
LPWAN	Low-Power Wide-Area Network
MCU	Micro-Controller Unit
OTP	One-Time Programmable memory
PCB	Printed Circuit Board
PLL	Phase-Locked Loop
PMIC	Power Management Integrated Circuit
PTAT	Proportional To Absolute Temperature
RTC	Real Time Clock
SoA	State-of-the-Art
UC	Use Case
UML	Unified Modeling Language
WPAN	Wireless Personal Area Network

Executive Summary

The present document is a Deliverable of AMANDA project, funded by the European Commission's Directorate-General for Research and Innovation (DG RTD), under its Horizon 2020 Research and Innovation Programme (H2020).
This document is part of **WP1 – System Specifications, Requirements and Use Cases of the AMANDA project**. The aim of WP1 is to provide an overall framework for the project, to ensure a common reference point regarding the system requirements that arise from use analysis and to provide overall consideration and guidelines with respect to the solution introduced. This Deliverable describes the first version of the architecture design of the AMANDA system as part of **Task 1.5: System specifications, Overall architecture and Design** and will be updated on **M18** and **M34**.

Table of Contents

List of Figures

List of Tables

1 Introduction

This Section outlines the purpose, context, scope and project phases covered by this Deliverable as well as the structure of the rest of the document.

1.1 Purpose, context and scope of this Deliverable

The main goal of the AMANDA project is to introduce, design and develop a maintenance-free, miniaturized and easily deployable ASSC for environmental sensing, as well as for asset and people tracking/monitoring in smart living and working environments.

The purpose of this Deliverable is to introduce detailed specifications for the components used for the AMANDA system, focusing on providing a complete overview of the ASSC architecture design. Moreover, a list of recommendations is proposed to administer further directions on the AMANDA consortium's work with regards to **Task 1.5: System specifications, Overall architecture and Design.** Task 1.5 focuses on two different aspects of the system design; the development of a breadboard circuit and the integration of the miniaturized system, which will be the final PCB design implementation.

This Deliverable aims at defining a main architecture framework for the AMANDA platform. In particular:

- The specifications of the building blocks and the individual components of the AMANDA platform will explain the detailed design and implementation during the upcoming technical work packages (notably WP2, WP3 and WP4)
- The specification of the interactions/communications between the building blocks of the architecture will be the drive of the integration of the AMANDA system in WP5
- The architecture presented in this document provides an overview of the AMANDA system, focusing on the way the platform will be integrated and customized in order to fulfil any end-user demands, as identified in Deliverable D1.3 – Voice-of-the Customer completed

This document is structured as follows:

- Section 2 presents the methodology used to develop the architecture of the AMANDA ASSC during Phase One of the project. Moreover, this Section illustrates the requirements and the architectural fundamentals of the platform
- Section 3 discusses the conceptual architecture of the AMANDA system and describes its major building blocks
- Section 4 illustrates the development viewpoint, and details the process that will be followed for the development of the AMANDA ASSC firmware. It defines the Partners' responsibilities per component, presents a set of guidelines for source code handling and configuration management with details for codding convention and version control. Moreover, it describes the sensor/data fusion and low power edge intelligence and energy availability assessment as well as the gradual decision making for the AMANDA project
- Section 5 discusses the security view of the AMANDA ASSC, both in terms of hardware and software levels
- Section 6 summarises the main aspects of the AMANDA architecture specification process and includes some recommendations on future work

1.2 Phases of the AMANDA project

The overall approach and methodology of the AMANDA project consists of five interrelated phases that are presented below:

- **Phase one**: Definition Requirements and System Design, sets the Groundwork of the project and contains all the work plan-required definitions for the AMANDA ASSC development. During this phase, the provided information will cover a set of requirements for all scientific topics of the project
- **Phase two**: Technical Development, this phase focusses on the design and development of the technology blocks for the ASSC. The results of this phase will transform the AMANDA specifications into operational architectural elements. Phase two constitutes the main part of the project
- **Phase three**: System Integration and Prototyping, simultaneously with phase two. It is focussed on the integration of all the available components together. In the future, the work plan of the project will deal with the creation and integration of all the modules into the miniaturized ASSC PCB
- **Phase four**: Iterative Deployment, Demonstration and Validation, for this phase an assessment takes place with regards to the validation and the continuous enhancement of the AMANDA platform. Any required changes are then implemented.
- **Horizontal activities**: Project Management, Dissemination and Exploitation, this phase includes all the horizontal activities of the project and is divided into two groups: overall project management activities and public awareness, dissemination and exploitation activities.

This report is part of the first phase of the project and the objectives to be addressed in this report are to analyse the system architecture based on the important functionalities and interconnections. Figure 1 shows the different phases of the AMANDA development approach.

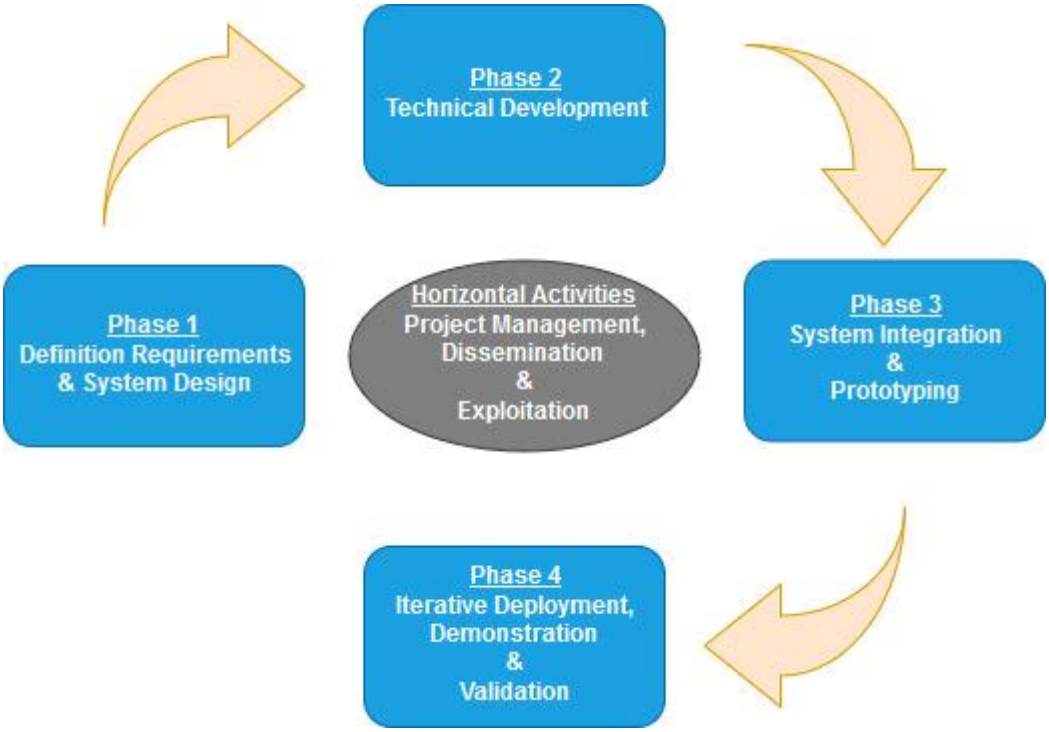

Figure 1 Phases of the AMANDA development approach

The goal of this Deliverable is to provide a high-level overview of the AMANDA ASSC architecture, to summarise the technical and functional design of the system and to provide functionality details of each individual component or module, and the AMANDA platform as a whole. Figure 2 depicts a detailed conceptual view of the Methodology that is followed by the AMANDA project. This Deliverable corresponds to the System Specification step.

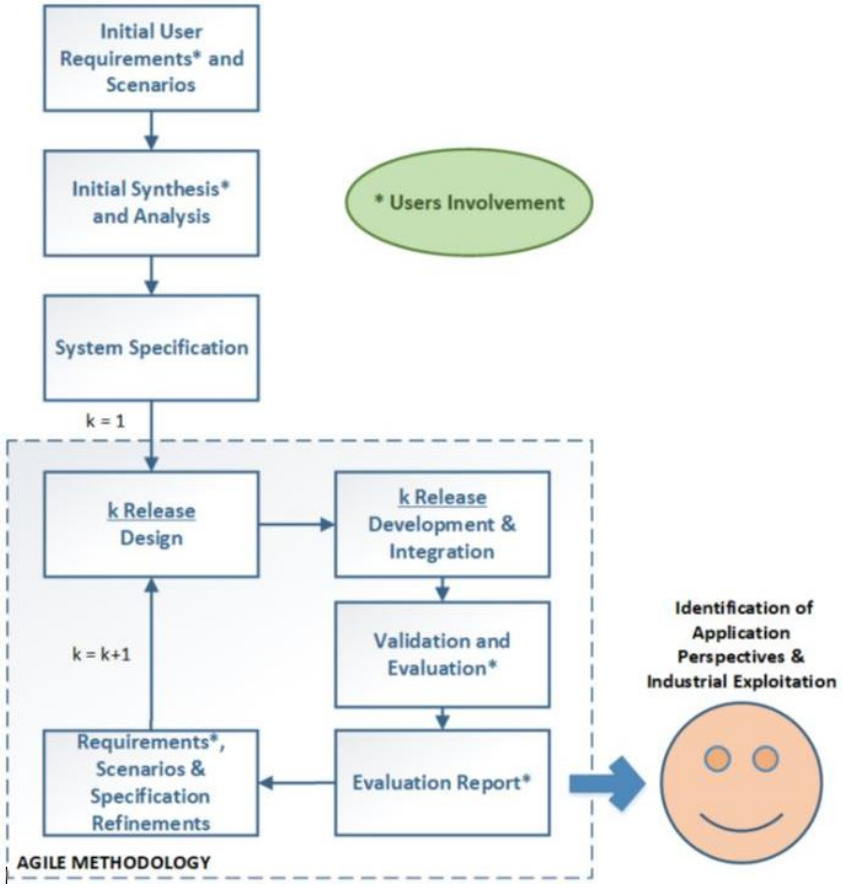

Figure 2 Conceptual implementation methodology

2 Methodology

To develop the architecture of the AMANDA ASSC, the project applied the ISO/IEC/IEEE 42010:2011 standard, titled Systems and software engineering - Architecture description [1]. The standard defines requirements on the system and software architecture in order to assist the understanding of the system's essence and key properties pertaining to its behaviour, composition and evolution. This will in turn affect concerns such as the feasibility, utility and sustainability of the system. It introduces the notion of a system architecture and defines core conceptual foundations of architecture description including viewpoints, views and models, identifies the notion of stakeholders and highlights ways to apply stakeholders' concerns to the architecture. The methodology is applied in phase one of the project – Definition Requirements and System Design. To implement and execute the methodology, the approach introduced in [2], and further described below, was followed.

2.1 Requirements and architecture

Initially, a SoA analysis, a feasibility study and benchmarking of best practices were performed under Task **T1.1 – SoA Analysis, feasibility study & benchmarking of best practices.** The findings of Task T1.1 were presented in Deliverable **D1.1 - SoA and Gap analysis/recommendations on ESS features report.**

To gather user needs and issues, surveys and interviews were conducted, focus groups were organized where stakeholders were gathered and asked to share their needs related to the project and phone interviews were arranged, as described in Deliverable **D1.3 - Voice-of-the Customer completed.** The needs and issues of the stakeholders were translated in a structured way into specific user and technical requirements for each use case.

2.2 Fundamentals

In terms of systems architecture, *stakeholders* can be defined as the individuals or entities that have an interest in a system. The specific requirements, needs and interests of the stakeholders are expressed as *concerns*. Different stakeholders for the AMANDA ASSC as well as their concerns have been identified as part of Task **T1.2 – System Requirements and Needs** and analysed in Deliverable **D1.3 – Voice-of-the Customer completed**. A brief overview of the identified stakeholders and the influence they could potentially have on the architecture of the AMANDA ASSC is provided below and depicted in Figure 3:

- **End users**. They are stakeholders that will use the final AMANDA system covering different identified use cases, for example as an automated room controller (UC2), an air quality station (UC6) or a payment device (UC13)
- **SMEs**. They might use the AMANDA ASSC for monitoring and automation purposes such as asset tracking (UC10), worker comfort level monitoring (UC20) or automotive sensing and cabin interior monitoring (UC25)
- **Research**. Stakeholders from a research discipline will mainly have a concern on the technical aspects of the ASSC and the individual components, as well as its mechanical and electrical properties.

The stakeholders will be mainly concerned with the accuracy, usability and autonomy of the final system and have privacy and security concerns. Feedback from end users may be collected from end users under **T6.3 – Overall Evaluation, Lessons Learnt and Improvements**, during preliminary in-field testing after the lab environment validation of the ASSC.

A major concept of the ISO/IEC/IEEE 42010:2011 standard is the viewpoint. According to [3], the architectural description of a system can be divided into a number of interrelated views. Then, a viewpoint can be characterized as a structured specification that supports the definition of such a view on the system.

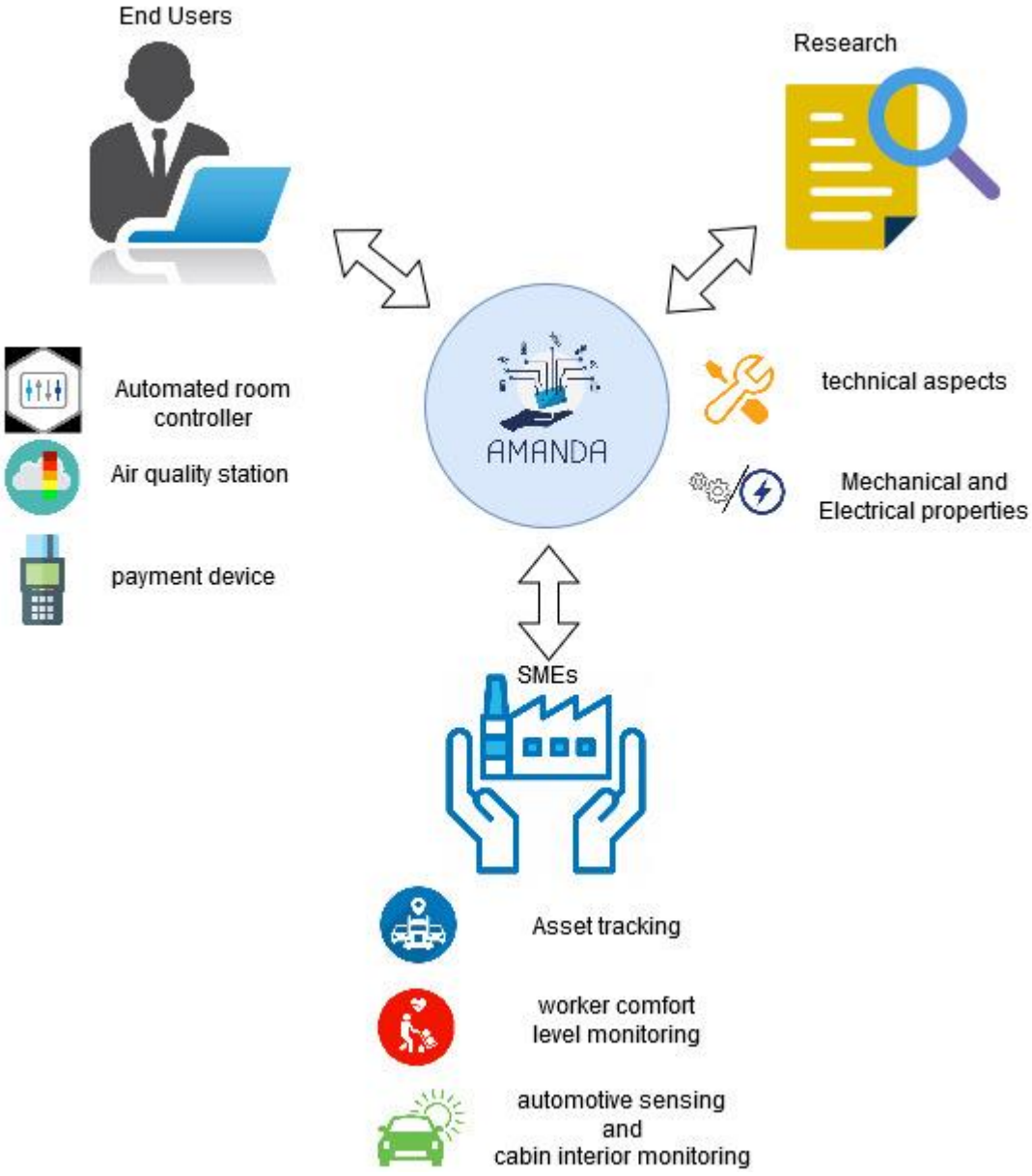

Figure 3 Identified stakeholders

The notions of views and viewpoints are defined in [1] as follows:

- **View**. An expression of structural aspects of the architecture of a system that illustrate the way the architecture addresses specific system concerns, held by one or more of its stakeholders
- **Viewpoint.** An establishment of the conventions for the construction, interpretation and use of architecture views to frame specific system concerns. It defines the stakeholders whose concerns are reflected in the viewpoint and guidelines, principles and template models for the construction of its views [4].

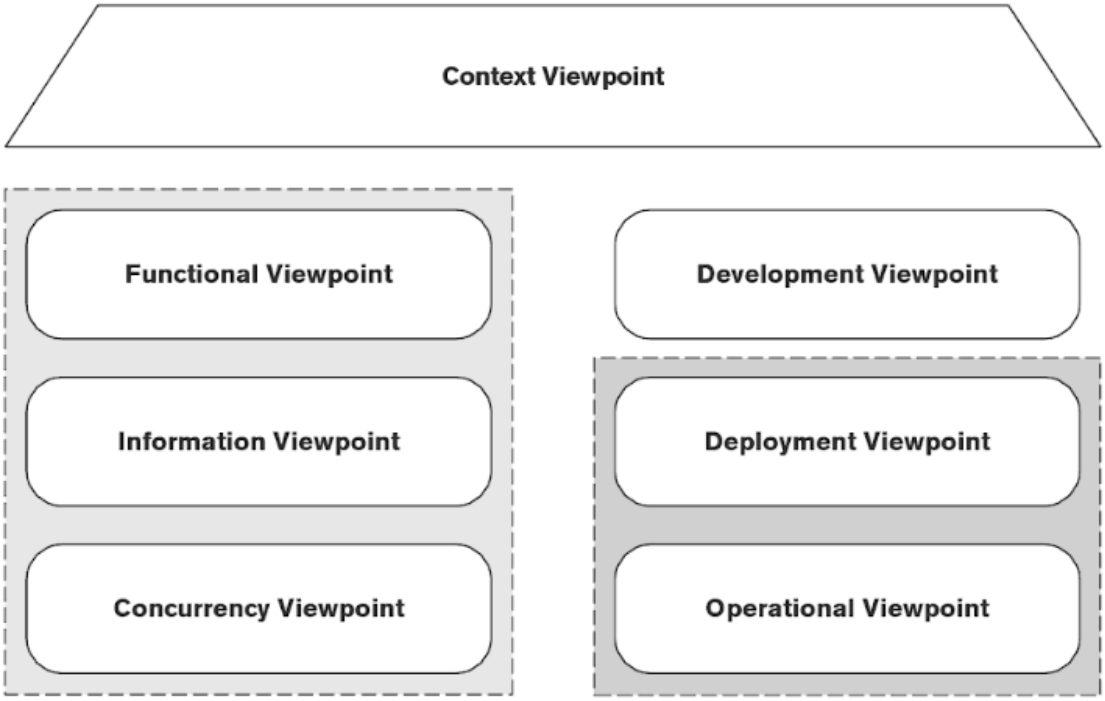

Figure 4 Viewpoint groupings [2]

2.3 Viewpoint catalogue

As proposed in [2], there are seven core viewpoints for the architecture of information systems; context, functional, information, concurrency, development, deployment and operational:

- **Context.** Describes the relationships, dependencies and interactions between the system and its environment
- **Functional.** Characterizes, together with the information and concurrency viewpoints, the fundamental organization of the system. Describes its functional elements, their responsibilities and primary interactions with other elements.
- **Information.** Describes the way information is stored, managed and distributed in the architecture
- **Concurrency**. Describes the concurrency of the system and defines the parts of the system that can run at the same time and the way this is controlled
- **Development.** Defines the system's implementation constraints, as discussed in Section 4
- **Deployment.** Describes the environment where the system will be deployed. Includes the documentation of the hardware requirements for the components and how they map to the runtime environment that will execute them
- **Operational.** Describes the way the system will be operated and supported once it is deployed

Although the different viewpoints are largely independent, they can be loosely grouped together, as shown in Figure 4. Figure 5 depicts the way the different viewpoints are inter-related. Since the AMANDA ASSC is an IoT device, its security, in both a hardware and software level, is of uttermost importance. The security viewpoint is discussed in Section 5. This Deliverable focuses on the development and security viewpoints as applicable to the AMANDA project.

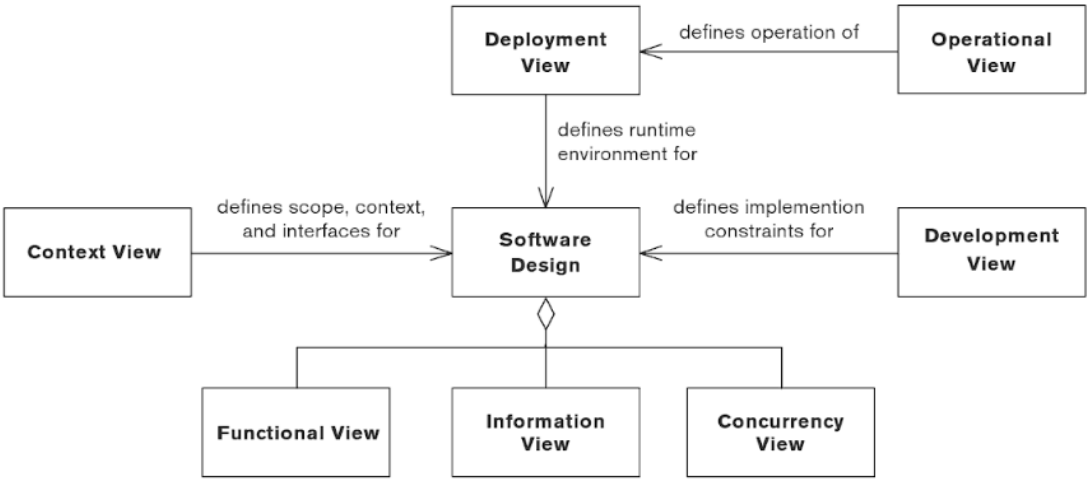

Figure 5 Viewpoint catalogue [2]

3 Conceptual architecture

This Section details the high-level perspective of the AMANDA system architecture. An overview regarding the purpose of each architectural block is provided and discloses details about the major building blocks of the platform.

AMANDA will act as a maintenance-free, miniaturized and easily deployable IoT solution for environmental sensing and asset & people tracking/monitoring in smart living and working environments. In order to ensure the ASSC's viability as an IoT solution, its conceptual architecture consists of 3 major building blocks as shown in Figure 6 and detailed below:

- The **Sensor and Edge Intelligence** block, which incorporates the sensing and processing capabilities of the ASSC
- The **Wireless and Security** block with the aim to setup the appropriate communication tools and infrastructure to match the ASSC's operational needs and intrinsic limitations of each version as detailed in Deliverables D1.2 and D1.3. Moreover, the software and hardware security mechanisms that will eradicate any possible vulnerability of the ASSC are designed and implemented as part of this architectural stage
- The **Energy Management** block, which comprises the Energy Harvesting, Power Management and Rechargeable Storage Modules, providing complete autonomy to that the AMANDA platform

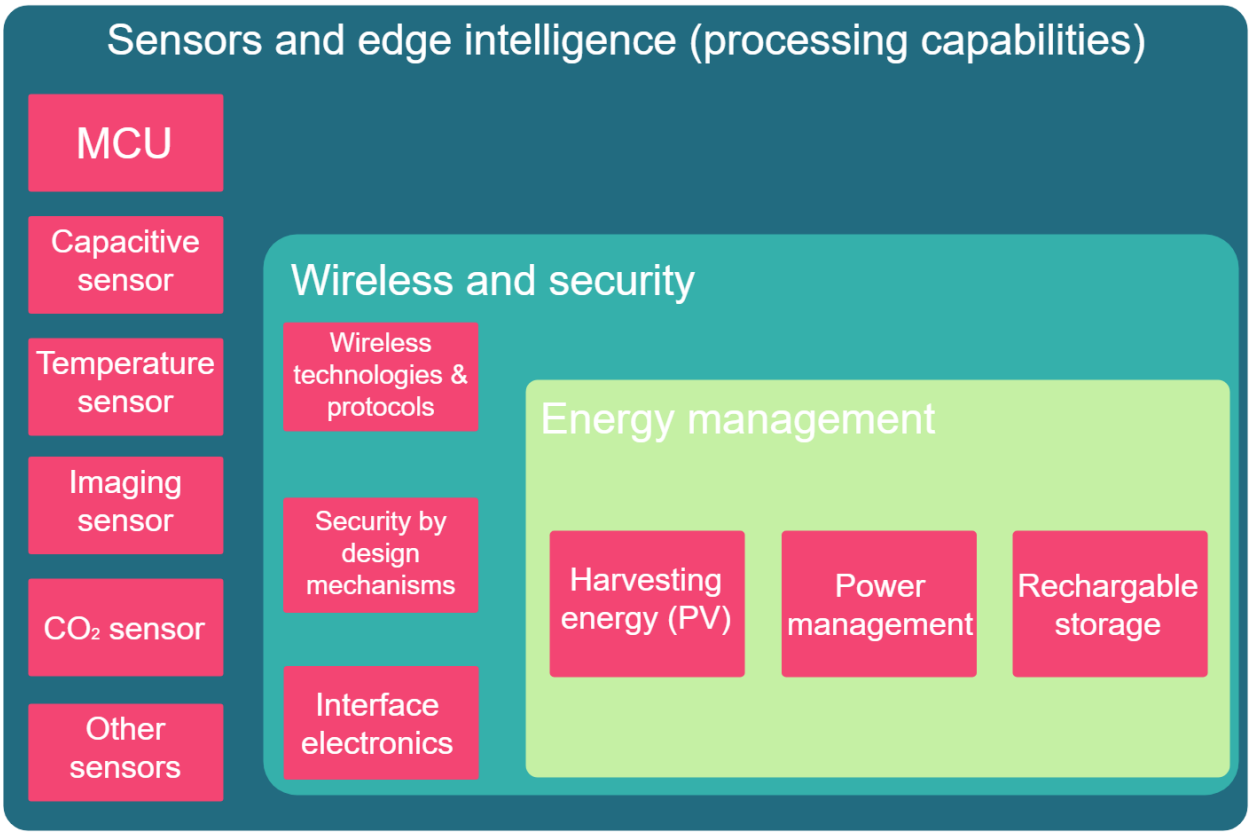

Figure 6 Conceptual architecture of the AMANDA ASSC

Further information regarding the high level components which form the aforementioned design blocks is presented in the following Sections. The methodology of the interconnections and interactions of the system, as they were defined in Deliverables D1.2 and D1.3, is also explained.

3.1 Sensor and edge intelligence block

The Sensor and edge intelligence block refers to the sensors integrated into the AMANDA ASSC. These include all sensors developed by the AMANDA partners and the off-shelf solutions that will be interfaced in the final system. In this block, the sensors are interconnected with the MCU and gather different information depending on the ASSC version (eg. indoor, outdoor or wearable). Then, the data is analysed and aggregated, in order to create the necessary decisions for automations/actions which will eventually achieve the pre-set conditions that are not currently met.

This block comprises the hardware (self-developed and off-the-shelf sensors) implementations that will cover the end-user needs, and the respective low-power algorithms that offer edge intelligence on the MCU. The different protocols that will be used to interconnect the various sensors and are implemented for the ASSC include UART, SPI, I^2C and I^2S.

3.2 Wireless and security block

The next block in the conceptual architecture is the wireless and security block. Within the sensor and edge Intelligence block, the ASSC handles the data fusion based on all the information provided by the individual components, and transmits all relevant information wirelessly. Therefore, there is a need for wireless communication strategies and processes that support mesh networks of AMANDA multiple-sensor cards, in combination with edge computing practices that aim at reducing overall power requirements.

3.3 Energy management block

The final, yet critical for the system, part of the ASSC's conceptual architecture is the energy management block. It has the same components for all the ASSC versions, similar to the wireless and security block. A thorough analysis of the electrical specifications and needs of the individual components is presented in Deliverable D1.2.

4 Development view

This Section provides an overview of the development viewpoint, describing in detail the process that will be followed for the development of the AMANDA ASSC firmware. The Section includes an overview of the Partners responsible for the development of software for each component, discusses the way the development and revision control of source code is handled, details the decision making process of the ASSC in terms of sensor/data fusion and low power edge intelligence and provides discussion on energy availability assessment and gradual decision making.

4.1 Firmware development

Firmware development is part of activities for **WP2 – Sensor development and multi-sensorial optimization**, **WP4 – cyber-secure mesh communication and processing and WP5 – Smart Interconnect PCB Development and System Integration**. Development will focus on the following areas:

- Energy availability assessment
- Sensor driver development for measurement execution
- Data fusion and end result calculations
- Wireless transmission drivers with cyber security mechanisms

The main functionality of the ASSC is to wake up, assess energy availability, execute measurements, calculate results and/or do decisions, transmit results and then go back to sleep.

The main firmware will assess energy availability as a first step. If not enough energy is available the system will abort and go back to sleep without losing valuable energy to execute the rest of the steps. Also measurements are assessed in every step in order to avoid useless transmissions.

The sensor drivers will be developed with a focus on low-power; it will be used to optimize the energy consumption of the overall sensing functionality, allowing for smart duty-cycling and enabling or disabling sensing functionality depending on the target use cases as defined in Deliverable D1.3.

A hardware and firmware implementation of the wireless solutions utilized for the AMANDA ASSC will be performed under WP4, and the results will be presented in Deliverable **D4.1 – Hardware and firmware implementation of the wireless solutions**. Under WP5, a basic firmware will be developed to test the main functionality of the first prototype of the AMANDA ASSC. The firmware will be finalized in subsequent stages of the project by adding more advanced functionality to the device and optimizing power consumption.

An overview of the responsible partners and the software requirements and dependencies for each component's driver for the MCU of the AMANDA ASSC is provided in Table 1 below.

Component	Owner	Programming Language/ Technology
PMIC	E-peas	C/C++
Capacitive sensor	Microdul	C/C++
Temperature sensor	Microdul	C/C++
Imaging sensor	E-peas	C/C++
CO_2 sensor	IMEC	C/C++
Edge intelligence layer	CERTH	C/C++
Security layer	CERTH	C/C++

Table 1 MCU component driver development

4.2 Source code and configuration management

An important part of the software development phase is the configuration management. Software configuration management is defined as the discipline of managing and controlling change in the evolution of software systems [5]. It deals with the management of complex software systems. At its core, configuration management is made up of a version control system. According to [6], software configuration management can be grouped into four functions:

- **Identification of configuration items.** Involves the unique identification of items such as system components and its work products
- **Change control.** Controls changes to the system and releases through developers, management or a control instance to ensure consistency
- **Status accounting**. Records the status of individual components to allow developers to distinguish versions and track issues related to the changes
- **Auditing**. Validation of releases by a quality control team to ensure the completeness, consistency and quality of the end product

As discussed in [7], some of the issues that are traditionally the domain of configuration management systems include:

- Version management. The maintenance of version history for all of the programming assets per component
- Temporary variation. The simultaneous modification of the same programming asset by two separate developers. By using developer-specific branches, developers do not see changes made by other developers until the final integration
- Distributed development. The distribution of configuration management databases over the world

4.2.1 Coding conventions

To optimize the readability of the source code of AMANDA and thus improve its quality and the ability to maintain and improve it, adhering to common coding conventions are mandatory for all developers contributing to the project. For C and C++ code, the Google C++ Style Guide [8] will be used. Some notable examples of the formatting guidelines include:

- Names should be descriptive and follow a camel case convention: *AMANDAProject, ContainerFactory, getLayerParameters*
- Only spaces should be used instead of tabs, to maintain visual coherency of the source code for all developers
- Each indentation level should be 2 spaces long
- Each line should have a maximum width of 120 characters
- Curly braces should be placed on the same line for if statements, for loops and so on and on a different line for functions/methods and structures

To easily maintain the coding conventions, ClangFormat [9] will be utilized, a set of tools built on top of LibFormat. ClangFormat uses a configuration file, .clang-format, which defines formatting rules to be applied to a source tree. The AMANDA configuration for ClangFormat can be found in the Appendix of this Deliverable.

4.2.2 Version control

The software configuration management of AMANDA, for firmware developed by the partners per component of the ASSC and for the final integrated system, is based on the Git distributed version control system [10], to address the issues of configuration management systems listed above. Git is used to track changes of development files and thus assist in the coordination between different developers. As opposed to local or centralized version control systems, developers working with Git fully mirror the repository locally, including its full history. Thus, if

any server dies, and these systems were collaborating via that server, any of the client repositories can be copied back up to the server to restore it. Every clone is really a full backup of all the data [11].

An important feature of Git is the concept of *branches.* Effectively a pointer to a snapshot of changes, branches can be used by developers to branch out of the main software tree and encapsulate changes before their merge back to the original code base.

The Git branching model that is used by the ALICE experiment at CERN to develop the O2 software project [12] will be adapted for AMANDA, as it is used to efficiently manage and track the development of a large multi-partner European project.

For the AMANDA Git branching model, there will be two long-term Git branches:

- The master branch
- The dev branch

The master branch will contain the most stable code. It will always be ready to build and always kept in a releasable state. To keep the repositories of all users synchronized, no history changes will be allowed on master, including *git push –f*. All new software patches will be introduced in master only via the *git merge --ff-only* command.

The dev branch will be the current development branch; it is the place where developers' code is merged into. It will be inherited from the latest master branch and its tip will always be at or ahead the tip of master.

From the developers' perspective, they will not *clone* the upstream AMANDA repository. Instead they will *fork* it and create a local copy. Then, locally, a different branch is created for each different feature or bug the developer is currently working on. Before pushing a local branch to the Git server the following actions must be performed:

- The code should be tested and ensured that it can be built and executed without errors
- The local branch should be rebased against the upstream dev branch and checked that can be applied cleanly
- Multiple commits per feature or bug should be squashed into a single commit

The developer then should initiate a new merge request from the AMANDA Git interface. It is the maintainers' responsibility to merge the branch to the upstream dev using fast forward to combine the commit history. The Git branching model of AMANDA can be seen in Figure 7 below.

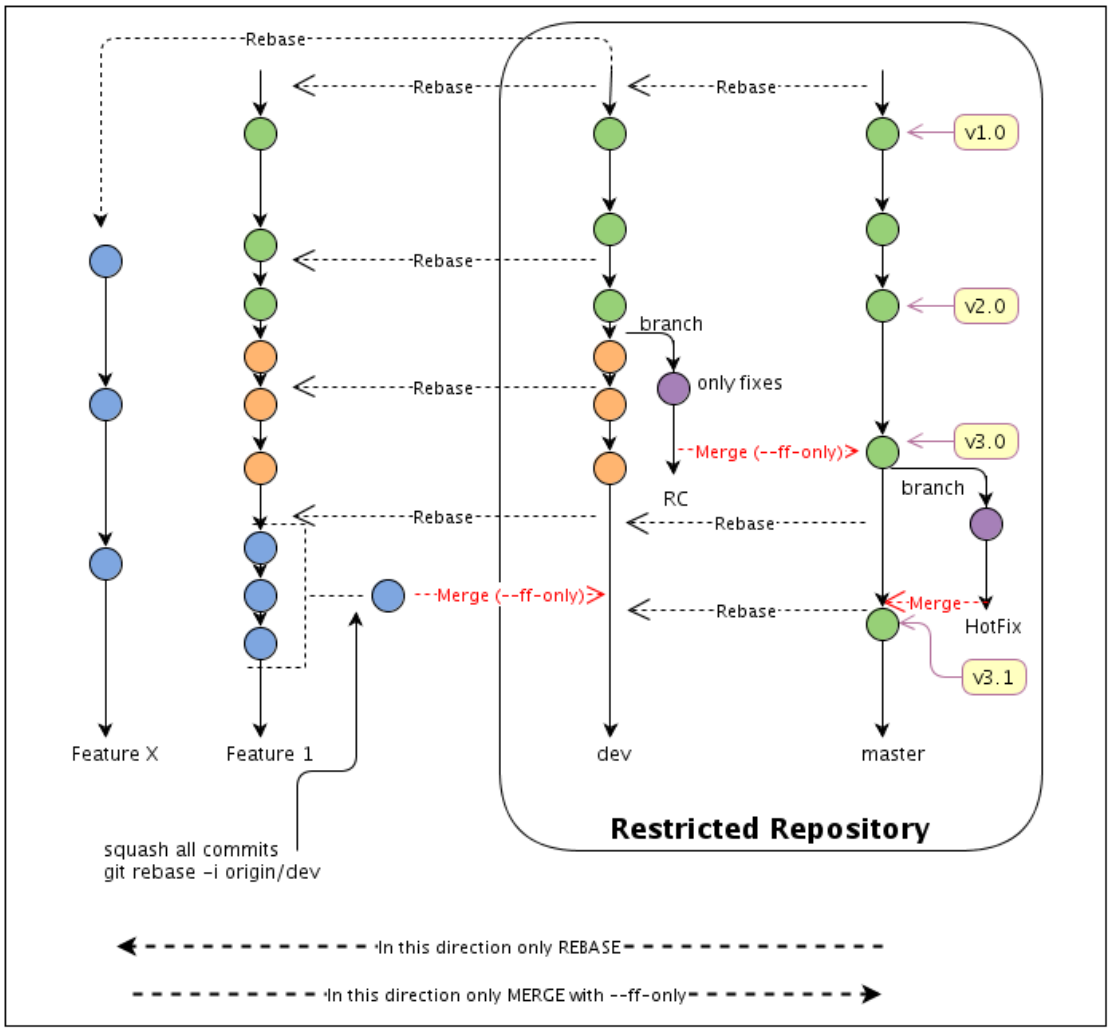

Figure 7 The Git branching model of AMANDA [12]

4.3 Sensor/data fusion and low power edge intelligence

Data fusion is a key enabling technology in which information from a number of sources is integrated to form a unified picture. The idea is that fusion of complementary information available from different sensors will yield more accurate results for information processing problems than treating the information from each sensor separately. In general, all tasks that demand any type of parameter estimation from multiple sources can benefit from the use of data/information fusion methods.

Hall and Llinas [13] have defined sensor data fusion as a method of combining sensor data from multiple sensors to produce more accurate, more complete, and more dependable information that could not be possible to achieve through a single sensor. Nakamura [14] have defined data fusion based on three key operations: complementary, redundant, and cooperative.

- **Complementary**. Assembling bits and pieces of a large picture together. A single sensor cannot say much about the environment as it would be focused on measuring a single factor such as temperature. However, when data exist that have been sensed through a number of different sensors, the environment can be understood in a much better way
- **Redundant.** The same environmental factor is sensed through different sensors. It helps to increase the accuracy of the data. For example, averaging the temperature value sensed by two sensors located in the same physical space would produce more

accurate information compared to a single sensor. It also reduces the resulting data through the combination of two or more sets of data streams together

- **Cooperative** operations combine the sensor data together to produce new knowledge. For example, reading RFID tags recorded in a supermarket can be used to identify such events as shoplifting. Let's consider a scenario where RFID reader in a supermarket shelf detects that an item has been removed from a shelf. The RFID sensor in the counter does not see the object during payments. Later, the RFID sensor in the exit door detects the item that was removed from the shelf earlier. This sequence of actions can be simply inferred as a shoplifting event

Data fusion is a data processing technique that associates, combines, aggregates, and integrates data from different sources. It helps to build knowledge about certain events and environments which is not possible using individual sensors separately. Data fusion also builds a context-awareness model that helps to understand situational context. The sensor data filtering stresses the requirement of filtering data to avoid large volumes of data transmission over the network.

A basic sensor data fusion example that is used widely in smart phones is the digital compass. It uses a combination of 3-axis magnetometer and the 3-axis accelerometer to provide compass functionality.

Data fusion is very important for a system like AMANDA. The ultra-low energy character of the system does not allow massive amount of data to be transmitted. Effort should be made to find a good equilibrium between data processing and assessment that will be executed in the card and data transmitted from it. AMANDA will apply software engineering optimization techniques to develop data fusions strategies and algorithms to minimize the energy requirements for data processing. Similarly, low power algorithms will be developed to offer edge intelligence at ASSC level in support of decisional autonomy under the anticipated environmental monitoring and tracking services. The overall processing module will interface with the power management unit to ensure optimum energy use at ASSC level taking into account the particular requirements of each application scenario.

4.4 Energy availability assessment

Since this system is an energy autonomous system with absolute dependency on energy harvested from the environment, the availability of energy should be assessed both directly and historically. This means that the device should be able to determine its current energy available in the battery in order to decide which measurements can be executed during this specific run cycle but it should also do a historical energy availability assessment in order to evaluate long term availability of energy. This will help to dynamically schedule the rate of future measurements and transmissions. In more simple words, the system should be able to determine beforehand which type of measurements can be executed and what amount of data can be transmitted, and also dynamically decide how often it should wake up for a new cycle.

4.4.1 Direct assessment of energy availability in battery

The energy availability of the battery at any given time can be determined from its State Of Charge (SOC). The SOC refers to how much charge remains in the battery and can be determined by the potential of the battery when not under load. The voltage can then be directly related to the proportion of charge left remaining in the battery through a simple look up table. The look up table and exact SOC voltages for the battery will be determined by Ilika once a prototype of the AMANDA battery is available and disseminated to partners. It should be noted that the potential of the battery may drop significantly when it is under load and therefore this will have to be taken into account during the energy availability assessment. If the potential were to fall below ~2V for a prolonged period then irreversible damage to the battery is likely to occur.

4.4.2 Evaluation of historical data for assessing daily energy availability from the power harvester

Since all the ASSC components depend on the average energy supplied by Lightricity energy harvesting module, it is also critical to assess the real-time and historical energy generated by the harvester. The nearly instantaneous power measurement requires access to two key parameters: the open-circuit voltage and the short-circuit current of EH module. The open-circuit voltage (V_{oc}) is directly measured by the e-peas PMIC every 5 sec when the maximum powerpoint control circuit disconnects the EH source in order to define the optimal level of voltage. The short-circuit current (I_{sc}) can be indirectly measured by the light sensor present on the ASSC, since the I_{sc} is directly proportional to the light intensity (such as lux level).

The V_{oc} and I_{sc} analog information can be sampled by an ADC, fed into the main MCU, stored in FRAM memory and/or sent wirelessly, or even displayed onto the ASSC for the versions with ultra-low power display. The product of V_{oc} and I_{sc} is directly proportional to the amount of power generated by the EH module (corrected by a fill-factor coefficient comprised between 75% and 85% across the relevant illumination levels). Another correction due to PMIC efficiency losses could also be implemented into a look-up table in order to derive the actual power transferred into the Ilika rechargeable battery.

The energy generated over a period of time T can be obtained by integrating the instantaneous power over that period T.

Critical decisions, such as change of operation mode and measurement/transmission rates, can be made by the main MCU using pre-defined algorithms that will take into account the instantaneous power, average energy, accumulated energy inside the battery, rate of discharge (due to load power consumption). These algorithms, described in the following Section, will make the ASSC self-adaptive to its ambient environment, a key and unique feature of the ASSC. As they are expected to be application specific, they will be developed and implemented on a case-by-case study.

4.5 Gradual decision making

The ASSC implements a newly gradual decision-making architecture. A core component of this is a predictive algorithm in the ASSC's firmware. The goal of this algorithm is to optimise the energy usage of the system according to the demands of the use case and if necessary to adapt the tasks to suit the energy needs. The benefits for the ASSC are longer operation times with limited input power available, delivery of data based on depreciation levels allowed by the use case.

4.5.1 General overview of the predictive algorithm

The implemented predictive algorithm is a further development on the energy management system previously used by ZHAW in [13]. A block diagram of this earlier implementation is displayed in Figure 8.

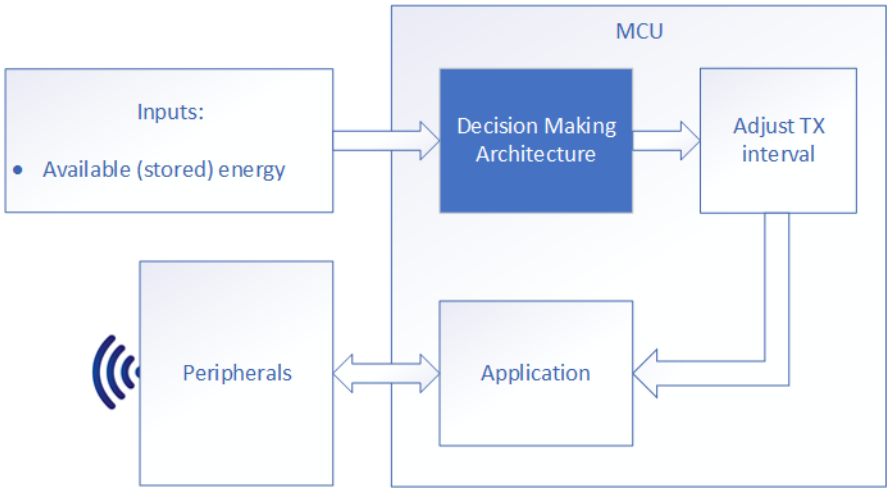

Figure 8 Earlier version of the predictive algorithm used in [13]

The system in [13] uses energy harvesting and sends measurement data via Bluetooth Smart. To ensure, that enough energy is available to execute all tasks, it monitors the charge state of its storage element. If the energy falls below a predefined value, the sending interval is increased by the firmware. When the energy level is high enough again, the interval is shortened again. This implementation is simple in comparison to the intended predictive algorithm used in AMANDA.

The algorithm for AMANDA is normally executed at a fixed interval on the system. The intervals can also be modified on the run, especially based on some input events. The algorithm should be energy efficient in order to reduce its impact on the energy and yield a positive energy balance. This is why the decision algorithm should only use a non-significant amount of energy during operation. Otherwise, it would not agree with the purpose of the gradual decision-making architecture. Figure 9 shows a block diagram of the predictive algorithm and its inputs and outputs.

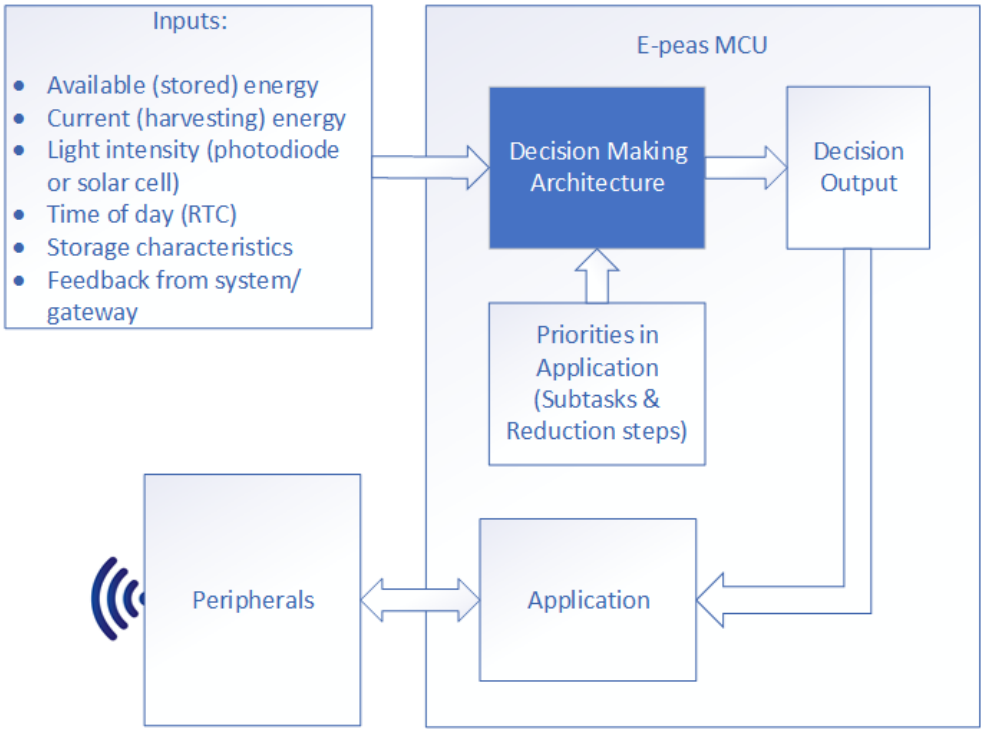

Figure 9 Block diagram of predictive algorithm

The predictive algorithm uses external live inputs, expected and historical data, predefined calculations and application priorities with subtask optimisations for decision making. It will give instructions to the application for execution intervals, subtask execution and subtask parameters (e.g. sensor resolution) according to the decisions made with the available information at execution. The core ASSC platform, described in D1.3, can provide live inputs in every use case. Table 2 shows the core elements and their possible live inputs for the algorithm.

Core Component	Possible Inputs
Energy Storage Units	Currently available energy stored in the component
PV harvester + light sensor	Instantaneous harvested power
PMIC	Available input power, charge ready thresholds (“power good”)
BLE + Zigbee radio	External commands/feedback by the user
LoRa/LoRaWAN radio	External commands/feedback by the user
NFC interface	External commands/feedback by the user
RTC Timer	Time of day, periodic trigger/interrupt

Table 2 Core components of the ASSC and inputs for the decision-making architecture

The inputs provide information about the currently available and harvested energy. They can also give user commands or system feedback. The inputs described in Table 1 contribute to the algorithm in the following way:

- **Energy storage units.** The charge status of the energy storage units tells the algorithm how much energy is currently stored in the system. Tasks can be executed according to their energy needs and the available energy
- **Light sensor and PV harvester.** The light intensity and PV voltage information gives information on how much energy per unit of time is being harvested now. If the ASSC is used in an outdoor environment and input information can be stored (for example in FRAM), the PV harvester also show the system for how much longer this amount of light intensity can be available based on historical data. This feature has to be implemented in combination with the RTC timer
- **PMIC.** The PMIC provides the momentarily harvested power form the PV harvesters. The algorithm can derive the available energy at the next execution cycle
- **BLE + Zigbee radio.** The user or the system operator can give inputs to the ASSC via the BLE + Zigbee wireless link. This is only possible, if the wireless link on the ASSC is configured bi-directionally, e.g. connected mode with BLE
- **LoRa/LoRaWAN radio.** The system operator has the possibility to give inputs and directives to the ASSC via gateway commands
- **NFC interface.** The user can influence the behaviour of the predictive algorithm by transferring information and configurations to the ASSC
- **RTC timer.** The RTC provides the exact date time and trigger to the predictive algorithm. According to this and the information coming from the PV harvesters, the algorithm can predict the energy harvested in the future. This works best in an outdoor application

In addition to the live inputs, the predictive decision-making architecture relies on predefined calculations, application priorities with subtask optimisations.

The values for the predefined calculations originate from the simulation tool, described in Deliverable 1.5. The partners are able to estimate the energy consumption of specific components and tasks with the tool. The parameters obtained by the tool are used by the algorithm to calculate, if the system can sustain a certain subtask or not.
The application priorities define which subtasks are most important to the functionality of the ASSC and are depending on the use case and ASSC version (indoor, outdoor or wearable). These priorities can be set at compilation time and can also be reconfigured through user input via the above-mentioned possibilities. The system will only execute a specific task, if the algorithm predicts that it can be executed to its completion.

4.5.2 Example of the predictive algorithm

An example use case is presented in Figure 10. It analyses the inputs and possible adjustments by the predictive algorithm in this use case.

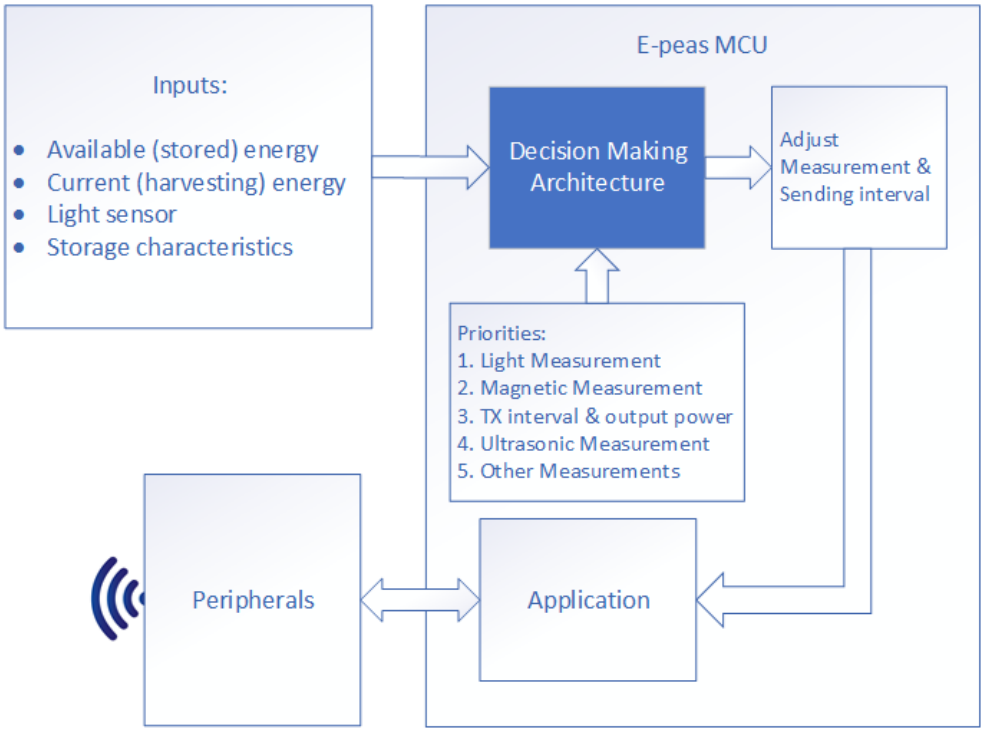

Figure 10 Block diagram of predictive algorithm in UC4

UC4 implements the ASSC for detecting parking slot occupancy. The card should be able to detect the arrival and departure of a car to/from a parking spot. Temperature, humidity, increased concentrations of toxic gases and light intensity are collected, if the ASSC is placed indoors (for example in UC4). The algorithm has therefore the following live inputs:

- Stored energy in Ilika battery
- Currently harvested energy from the PV harvesters (accessed via the PMIC)
- Light sensor
- Storage characteristics from the Ilika battery

The information gained from those inputs can be used by the algorithm to make an energy budget for the current cycle. It knows how much energy is available and can now adjust the tasks executed by the application.
The tasks which need to be completed are (from most to least important):

- **Light measurement.** The light sensor uses the least amount of energy of all used sensors in the system. It can thus be used most of the time.

- **Magnetic measurement.** The magnetic measurement in combination with the light measurement increases the identification of a vehicle and is less energy consuming than the ultrasonic measurement
- **TX interval.** With some sensor data, it is now important, that the sensed information is transferred to the user. Otherwise no detection can be made. Adjusting the TX interval and often also the output power can decrease the energy consumption
- **Ultrasonic measurement.** The ultrasonic measurement is considered supplementary to the other two measurements. In combination with its high energy demand, it is not as important as the previous tasks
- **Other measurements.** The other measurements (temperature, humidity, CO_2) provide additional environmental information (for statistical analysis) that are not critical to the core function of this use case. Thus, they have the lowest priority in this use-case

The algorithm is now calculating, which application sub-tasks can be executed with the available energy according to their importance to the system.

5 Security view

This Section details the security view of the AMANDA platform. First, an overview regarding the overall security view is presented. Then, specifics concerning the individual parts of this view are provided. The biggest IoT vulnerabilities and security issues that exist are divided in two categories:

- **Physical weaknesses**. Exist when a hacker can easily disassemble a device and access its storage. Even having exposed any type of ports can lead to hackers gaining access to the device's storage medium and compromising any data on the device. There are two types of physical vulnerabilities, invasive and non-invasive:
  - **Invasive attacks** need the chip surface to be exposed, meaning that the chip can be physically manipulated
  - **Non-invasive attacks** require the attacker to be close enough to target the chip and sense electrical characteristics, this allows the attackers to change the behaviour of the devices or gather sensitive information
- **Remote vulnerabilities**. They include the following:
  - Connectivity issues or the need for end-users to manually download updates directly, leaving devices open to newly discovered security vulnerabilities
  - Easily decipherable passwords, which might be left in place by vendors and end-users. When left open to remote access, these devices become easy prey for attackers running automated scripts for bulk exploitation.
  - APIs are commonly targeted by a variety of threats, including Man in the Middle (MITM), code injections (e.g., SQLI), and distributed denial of service (DDoS) assaults.

This is why AMANDA provides a secure-by-design IoT solution by safeguarding the interconnections on a hardware/software level as well as the wireless communication protocols used by the ASSC. The security of the ASSC can be clustered into two different levels:

- **Hardware and software level.** The design mechanisms used by the AMANDA consortium to ensure the safety of the ASSC
- **Wireless communication level.** The proper wireless communication protocols that need to be implemented

Further details on how the ASSC will be protected are presented below.

5.1 Hardware and software level

The compact design and high-end technologies used on the hardware level of the ASSC when it comes to the platform's encapsulation, as detailed in Deliverable D1.6 - Full System Specification & BOM, offer the necessary safety measures to protect the card from the major types of hardware attacks like side channels analysis attacks (power, EM, timing), tampering attacks (probing, de-packaging), damage attacks and others.

When it comes to the software aspect of this security level, there are various parameters that are taken into consideration from the multi-layered unified firmware will be used since it will provide a mixed level of threat protection and it will cover the following:

- Authentication
- Establishing principles for Internet of Things security
- Whether the data needs to be trusted
- If the safe and/or timely arrival of data is important
- Whether it is necessary to restrict access to or control of the device
- If it is necessary to update the software on the device
- Whether the ownership of the device needs to be managed or transferred in a secure manner
- Whether the data need to be audited

- The necessary encryption/decryption algorithms for data acquisition and data transmission

5.2 Wireless communication level

The only wireless protocols that offer support mesh networks of multiple ASSCs while offering secure over the air connections with low power consumption and can cover all the needs of the ASSC are the LoRaWAN [14] and BLE [15] protocols.

5.2.1 LoRaWAN protocol

The LoRaWAN protocol stack is shown in Figure 11 below. It consists of an application layer, a MAC layer and a PHY layer. Data from the application layer is mapped into the MAC Payload. The MAC layer constructs the MAC frame using the MAC payload which in return contains a frame header, a frame port, and the frame payload.

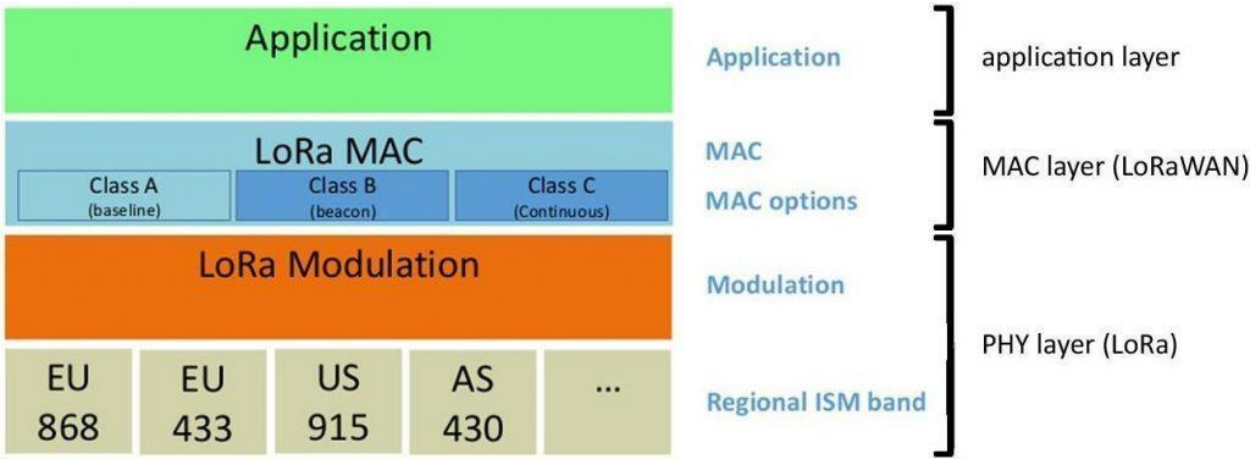

Figure 11 LoRaWAN protocol stack [14]

The RF parameters are encapsulated in the LoRaWAN RF or physical layer parameters. LoRaWAN utilizes two layers of security as shown in Figure 12. One for the network and one for the application layer. The network layer security ensures the authenticity of the device in the network, while the application layer security ensures that no network operator has access to the end user's application data.

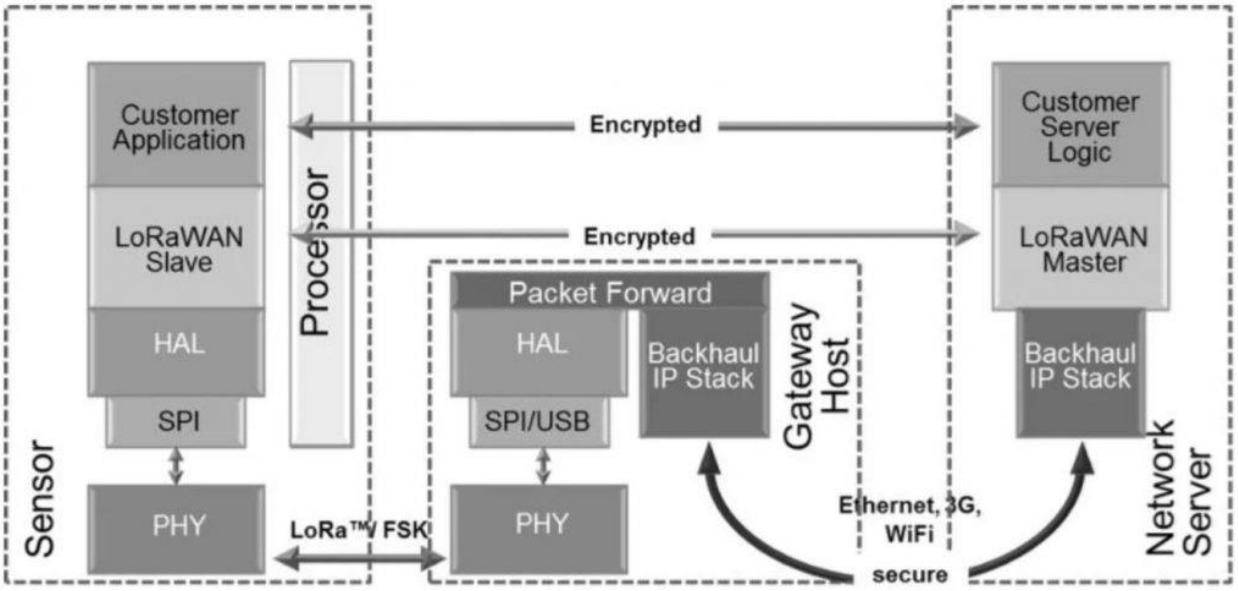

Figure 12 LoRaWAN dual layer security [14]

5.2.2 BLE protocol

The majority of the current low-energy application profiles are based on the Generic Attribute Profile (GATT), a general specification for sending and receiving short pieces of data, known as attributes, over a low energy link. The Bluetooth mesh profile is an exception since it is based on the General Access Profile (GAP). The GAP modes and procedures form the keystone BLE operations:

- Discover and connect with peers
- Broadcast data
- Establish secure connections

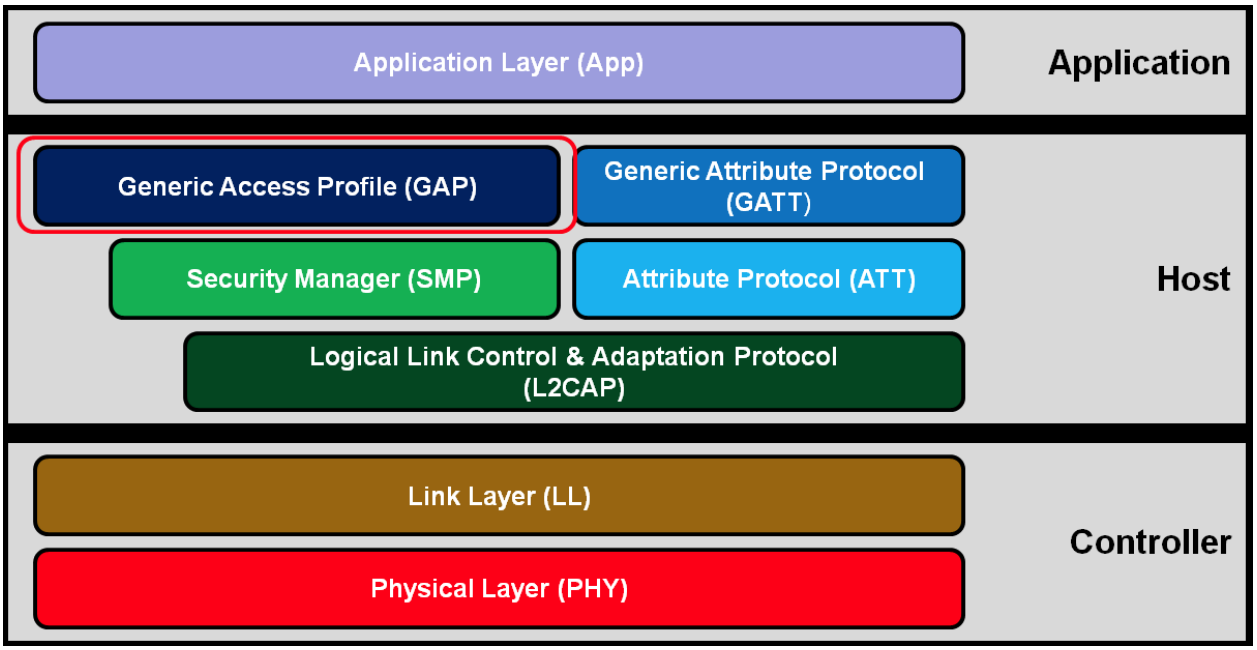

Figure 13 BLE protocol stack [15]

For a BLE connection, GAP defines two security modes along with several security levels per mode, security mode 1 and security mode 2. Each BLE connection starts in security mode 1,

level 1, and can later be switched to any security level by means of an authentication procedure.

Security mode 1 enforces security by means of encryption, and contains four levels:

- **Level 1**. No Security - No authentication and no encryption
- **Level 2**. Unauthenticated pairing with encryption
- **Level 3**. Authenticated pairing with encryption
- **Level 4**. Authenticated LE Secure Connections pairing with encryption.

On the other hand, *Security Mode 2* enforces security by means of data signing, and contains two levels:

- **Level 1**. Unauthenticated pairing with data signing
- **Level 2**. Authenticated pairing with data signing

BLE uses the AES-CCM cipher with 128-bit key length to provide data encryption and integrity over a wireless link and is considered secure until 2030. The key is generated using Diffie-Hellman method with elliptic curve cryptography (ECC). Each device generates the same AES-CCM key using the ECC public key received from the peer and its own secret ECC private key. The Bluetooth sensor and gateway must be paired with one another before they can perform any data transfer. The two main pairing methods in BLE are:

- **Passkey entry pairing.** A 6-digit pairing code is displayed and the user enters it in the gateway or vice versa. Alternatively, if both devices have numeric keypads, the user can enter the same pairing code in them. Each device then verifies the code with the peer. The verification is performed by sending 20 wireless messages by each device. Each message carries cryptographic hash of a data block containing one bit of the pairing code (code is 20-bit binary number).
- **Numeric comparison pairing.** A pairing code is generated by each device using cryptographic techniques and displayed to the user. A user then compares the codes displayed on the two devices and provide yes/no feedback on whether they match.

6 Conclusions and future work

This **Deliverable D1.7 Architecture design of the AMANDA system** completed report is a part of **Task 1.5: System specifications, Overall architecture and Design** and it is the first version delivered by **Milestone MS8**. Two revisions on **M18** and **M34** will be followed up. The implemented methodology applied to **Phase one: Definition Requirements and System Design** provides the project with a well-defined process and structure for describing the AMANDA ASSC architecture design. The IEEE 1471 standard was followed during the designing of the system architecture. This Deliverable introduced the AMANDA ASSC architecture, which specifies the main building blocks of the ASSC system that will grow in this project and provides with guidelines towards their integration.

This report consists of the following Sections. After a comprehensive introduction with details for Phases of the AMANDA project, Section 2 provided the methodology and requirements of the ASSC architecture. Section 3 contains a compendious overview of the conceptual architecture design regarding all the main system buildings blocks such as the sensor and edge Intelligence block, the wireless and security block and the energy management block. Section 4 illustrated the Development view of the AMANDA platform while Section 5 detailed the Security view. The work conducted in this Deliverable will assist in the definition of a complete overview of the AMANDA ASSC architecture. Future work will focus on fine-tuning the architecture design of the AMANDA device and define a high-level overview of the components. Additionally, the project will focus on functionality design and system miniaturization to increase flexibility, allowing for multiple iterations during the integration process.

7 Bibliography

[1] ISO, "IEC/IEEE Systems and Software Engineering: Architecture description," *ISO/IEC/IEEE 42010: 2011,* 2011.

[2] N. Rozanski and E. Woods, Software systems architecture second edition: working with stakeholders using viewpoints and perspectives, Addison-Wesley, 2011.

[3] A. Heuer, T. Kaufmann and T. Weyer, "Extending an IEEE 42010-compliant viewpoint-based engineering-framework for embedded systems to support variant management," *International Embedded Systems Symposium,* pp. 283-292, 2013.

[4] Adapt4EE, "Deliverable D2.1 - Conceptual Architecture," [Online]. Available: http://adapt4ee.eu/adapt4ee/files/document/deliverables/Adapt4EE-Deliverable-D2.1.pdf.

[5] IEEE, "1042-1987 - IEEE Guide to Software Configuration Management," [Online]. Available: https://standards.ieee.org/standard/1042-1987.html.

[6] B. Bernd and A. H. Dutoit, "Object-Oriented Software Engineering Pearson New International Edition. Using UML, Patterns, and Java," 2014.

[7] R. V. Ommering, "Configuration management in component based product populations," *Software Configuration Management,* 2001.

[8] Google, "Google C++ Style Guide," [Online]. Available: https://google.github.io/styleguide/cppguide.html.

[9] "ClangFormat," [Online]. Available: https://clang.llvm.org/docs/ClangFormat.html.

[10] "Git distributed version control system," [Online]. Available: https://git-scm.com/.

[11] S. Chacon and B. Straub, Pro Git, Apress, 2014.

[12] C. Kouzinopoulos, "Development of the O2 prototype," [Online]. Available: https://indico.cern.ch/event/351206/contributions/824366/attachments/694238/953275/Presentation6.pdf.

[13] M. Meli and L. Hegetschweiler, "Affordable Energy Autonomous Wireless Sensor for Day and Night," February 2016. [Online]. [Accessed 29 July 2019].

[14] L. Alliance, "LoRa Alliance," [Online]. Available: https://lora-alliance.org/.

[15] B. SIG, "Bluetooth," [Online]. Available: https://www.bluetooth.com/.

[16] AMANDA consortium, "D1.3 Voice-of-the Customer completed," 2019. [Online]. Available: https://www.amanda-project.eu/documents/public-deliverables.

[17] Analoge Devices, "Precision Analog Microcontroller with Chemical Sensor Interface - UG-1262: ADuCM355 Reference Manual (Rev. 0)," [Online]. Available: https://www.analog.com/en/products/aducm355.html#product-overview. [Accessed 24 07 2019].

[18] O. Semiconductor, "ON Semiconductor," 05 2018. [Online]. Available: https://www.onsemi.com/pub/Collateral/RSL10-D.PDF. [Accessed 07 15 2019].

[19] Semtech, "Semtech," 12 2017. [Online]. Available: https://www.semtech.com/uploads/documents/DS_SX1261-2_V1.1.pdf. [Accessed 07 15 2019].

[20] R. W. World, "RF Wireless World," [Online]. Available: https://www.rfwireless-world.com/Terminology/BLE-Protocol-Stack-Architecture.html.

[21] M. Azure, "Virtual machine extensions and features for Windows," [Online]. Available: https://docs.microsoft.com/en-us/azure/virtual-machines/extensions/features-windows.

[22] Wikipedia, “Blue Pill Software Wikipage,” [Online]. Available: https://en.wikipedia.org/wiki/Blue_Pill_(software).

[23] C. Kallenberg, X. Kovah, J. Butterworth and S. Cornwell, “Extreme Privilege Escalation On Windows 8/UEFI Systems”.

[24] A. Cui, M. Costello , J. Kataria and S. Stolfo, Stepping P3wns, Black Hat USA, 2013.

Appendix I: Configuration of the Clang Format tool

BasedOnStyle: Google
AccessModifierOffset: -1
AlignEscapedNewlinesLeft: true
AlignTrailingComments: true
AllowAllParametersOfDeclarationOnNextLine: false
AllowShortFunctionsOnASingleLine: true
AllowShortIfStatementsOnASingleLine: false
AllowShortLoopsOnASingleLine: false
#AlwaysBreakBeforeMultilineStrings: true
AlwaysBreakTemplateDeclarations: true
BinPackParameters: true
BreakBeforeBinaryOperators: false
BreakBeforeBraces: Linux
BreakBeforeTernaryOperators: true
BreakConstructorInitializersBeforeComma: false
ColumnLimit: 0
CommentPragmas: '^ IWYU pragma:'
ConstructorInitializerAllOnOneLineOrOnePerLine: true
ConstructorInitializerIndentWidth: 2
ContinuationIndentWidth: 2
Cpp11BracedListStyle: false
DerivePointerBinding: false
ExperimentalAutoDetectBinPacking: false
IndentCaseLabels: true
IndentFunctionDeclarationAfterType: true
IndentWidth: 2
Language: Cpp
MaxEmptyLinesToKeep: 1
KeepEmptyLinesAtTheStartOfBlocks: true
NamespaceIndentation: None
ObjCSpaceAfterProperty: false
ObjCSpaceBeforeProtocolList: false
PenaltyBreakBeforeFirstCallParameter: 1
PenaltyBreakComment: 300
PenaltyBreakFirstLessLess: 120
PenaltyBreakString: 1000
PenaltyExcessCharacter: 1000000
PenaltyReturnTypeOnItsOwnLine: 200
SortIncludes: false
SpaceBeforeAssignmentOperators: true
SpaceBeforeParens: ControlStatements
SpaceInEmptyParentheses: false
SpacesBeforeTrailingComments: 1
SpacesInAngles: false
SpacesInContainerLiterals: true
SpacesInCStyleCastParentheses: false
SpacesInParentheses: false
Standard: Cpp11
TabWidth: 2
UseTab: Never