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

D4.3 AMANDA Edge Intelligence and User Interfacing

Deliverable No.		**D4.3**	
Work package No.	**WP4**	Work package Title and task type	**Cyber-secure mesh communication and processing**
Task No.	**T4.3**	Task Title	**T4.3 Edge Intelligence and User Interfacing**
Lead beneficiary		**CERTH**	
Dissemination level		**PU**	
Nature of Deliverable		**R**	
Delivery date		**M30**	
Status		**Final**	
File Name:		**AMANDA_D4.3_AMANDA_Edge_Intelligence_and_User_Interfacing-v1.0**	
Project start date, duration		**02 January 2019, 42 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			
Surname	**Initials**	**Beneficiary Name**	**Contact email**
Papaioannou	AP	CERTH	alexiopa@iti.gr

Co-authors				
#	**Surname**	**Initials**	**Beneficiary Name**	**Contact email**
1	Kanlis	AK	CERTH	alexkanlis@iti.gr
2	Karanassos	DK	CERTH	dkaranassos@iti.gr
3	Kouzinopoulos	CS	CERTH	kouzinopoulos@iti.gr
4	Sideridis	PS	CERTH	sideridis@iti.gr
5	Meli	MM	ZHAW	mema@zhaw.ch

Reviewers List

List of Reviewers				
#	**Surname**	**Initials**	**Beneficiary Name**	**Contact email**
1	Bellanger	MB	Lightricity	mathieu.bellanger@lightricity.co.uk
2	Stajic	SS	ZHAW	stji@zhaw.ch

Document history

Version	Date	Status	Modifications made by
V0.1	1/09/2020	Initial draft	CERTH
V0.2	15/10/2020	SoA & Edge Intelligence module architecture	CERTH
V0.3	20/11/2020	First draft of User interfacing	PENTA
V0.4	15/12/2020	First draft for embedded storage & processing optimization	ZHAW
V0.5	20/02/2021	Edge Intelligence module for all scenarios	CERTH
V0.6	17/04/2021	Update on Edge Intelligence module architecture	CERTH
V0.7	1/05/2021	Updated on embedded storage & processing optimization	ZHAW
V0.8	12/05/2021	Updated on User interfacing	PENTA
V0.9	15/06/2021	Final draft, submitted for internal review	CERTH
V1.0	30/06/2021	Final version, ready for submission	CERTH, ZHAW, Lightricity

List of definitions & abbreviations

Abbreviation	Definition
AI	Artificial Intelligence
ASHRAE	American Society of Heating, Refrigerating and Air-Conditioning Engineers
ASSC	Autonomous Smart Sensing Card
AMR	Anisotropic Magneto-Resistive
ANN	Artificial Neural Networks
AUC	Area Under the Curve
BLE	Bluetooth Low Energy (also called Bluetooth Smart)
CNN	Convolution Neural Network
DNN	Deep Neural Network
DS	Dempster-Shafer
DT	Decision Tree
FZ	Fuzzy Logic
GAN	Generative Adversarial Network
GMR	Giant Magnetoresistance
HDF5	Hierarchical Data Format version 5
IoT	Internet of Things
ITS	Intelligent Transportation System
KFD	Kernel Fisher Discriminant
KNN	K-Nearest Neighbours
ML	Machine Learning
MAE	Mean Absolute Error
MCU	Micro Controller Unit

MLP	Multilayer Perceptron
NB	Naive Bayes
NFC	Near Field Communication
RBF	Radial Basis Function
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristics
SoA	State-of-the-Art
SPI	Serial Peripheral Interface
SRAM	Static Random-Access Memory
SVM	Support Vector Machine
UI	User Interface
VOC	Volatile organic compounds
WSN	Wireless Sensor Network

Executive Summary

This Deliverable is part of **Task T4.3 - Edge Intelligence and User Interfacing.** It provides a thorough State-of-the-Art analysis of different edge intelligence architectures and techniques and discusses their applicability for low-power embedded systems. Additionally, an extensive analysis is given of the AMANDA's edge intelligence core and information is provided on the implementations on different use cases as described in **Deliverable D1.10 - AMANDA Operational Scenarios Definition v2.** For the implementations, experimental results are detailed on the total power consumption, including the power consumption of the existing sensors, as well as the accuracy of the predictions. A presentation of the configurable user interface that can be utilised to setup and develop upon the AMANDA's edge intelligence core is also given. Lastly, the result of the evaluation of another SoA architecture suitable for low-power edge processing is presented.

Task T4.3 - Edge Intelligence and User Interfacing is part of **WP4 - Cyber-secure mesh communication and processing** of the AMANDA project. The aim of WP4 is to implement the appropriate wireless technologies for the card, as well as the appropriate cyber security, cloud platform connection and user interfacing.

The research results of this document will provide further directions to optimize the edge intelligence capabilities of the AMANDA platform. Since the individual, partner-developed components of the system are not in their final form and the miniaturized PCB prototype is still under design and development, additional evaluation and optimization of the data fusion component will be performed as part of **Task T5.3 - Software optimization for enhanced functionality and autonomy boost.** The results of this process will be reported in **Deliverable D5.3 - Report on Software optimization**.

Table of Contents

List of Figures

List of Tables

1 Introduction

The main purpose of **Task T4.3 - Edge Intelligence and User Interfacing** is to design and develop a multi-functional and lightweight edge intelligence engine to support the AMANDA card's low-power and intelligent capabilities, to provide a usable and configurable user interface to developers for enabling the setting up and additional development of the core engine and to investigate the embedded storage and processing optimisation of the AMANDA ASSC.

This Deliverable details the implementation of different intelligence models that utilise AI techniques to provide predictive and reactive intelligence to the AMANDA ASSC. Supervised learning was used together with classification and regression ML methods to identify patterns, extract and select features, perform predictions and make decisions with minimal human involvement. Furthermore, different optimisation techniques have been applied during the development of edge intelligence strategies and algorithms to further reduce their energy requirements. During the design and evaluation phases, key performance metrics were taken into consideration related to power consumption, memory and computing power requirements. This Deliverable includes detailed evaluation of the performance of the ML models in terms of power consumption, memory requirements as well the accuracy of each model for the different Scenarios of the project.

ML is a subset of AI and computer science, based on the idea that a system can be trained without being explicitly programmed by using historical data and algorithms, in order to mimic the way that humans learn, gradually improving its accuracy. Recommendation engines are a common use case for ML applications. Other popular uses include image and speech recognition, traffic prediction, fraud detection, spam filtering, malware threat detection as well predictive maintenance.

Deep learning is a subset of ML that consists of two basic elements: training and inference, both of which can be performed on different processing platforms. Figure 1 illustrates the two main pieces and the processing platforms which are needed to execute each phase of an ML model. The training phase typically occurs offline, on high performance computer nodes or in the cloud and involves feeding massive labelled data sets into a DNN. The result of the training phase is a trained neural network that, when deployed, can perform a specific task such as counting and tracking people within a working place, monitoring transportation conditions of vaccines, or determining the authenticity of a bill. Inference is the process of deploying a trained neural network on an embedded system that executes the algorithm. Due to the embedded system's constraints, the neural network is often trained on a different processing platform than the one that performs the inference.

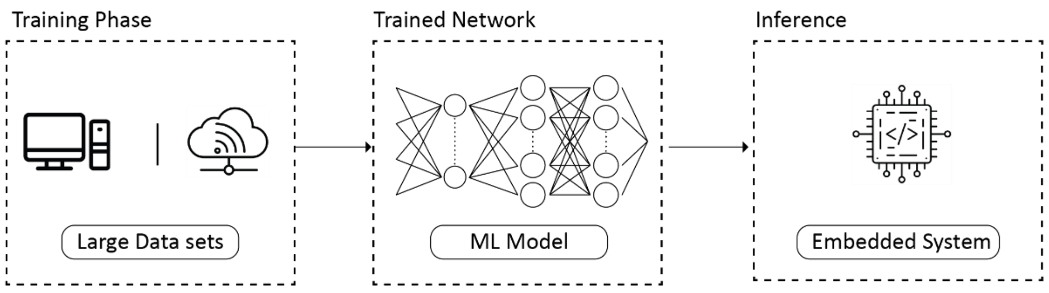


Figure 1 Deep learning development workflow

UI design is the process that designers use to build interfaces in software applications, focusing on looks or style and ensuring that the interface contains elements that are simple to access, understand and use in order to facilitate those actions. Also, it is responsible for interactions between the user and the application and it is one of the most important part of all applications. A successful user interface should be simple, effective, easy to use and should support the effective functionality of the application. Interface elements commonly include:

- Input controls: a component that allows users to enter data, e.g. buttons, text fields, checkboxes, radio buttons, dropdown lists, list boxes, toggles, date field
- Navigational components: a component that helps users move around an application or website, e.g. breadcrumb, slider, search field, pagination, slider, tags, icons
- Informational components: a component that shares information with users, including tooltips, icons, progress bar, notifications, message boxes and modal windows
- Containers: a component hold related content together, e.g. accordion

A usable and configurable interface has been designed as part of Task T4.3, which will be provided for developers to setup and develop upon the core edge intelligence engine of the AMANDA card. The available configurations for each Scenario are presented in Section 3. An extended report about the efficiency of the edge intelligence module including measurements on different conditions for each scenario, will be provided in **Deliverable D5.3 - Report on software optimization.**

This document is structured as follows:

- Section 1 includes the introduction and context of the document
- Section 2 presents the AMANDA edge intelligence component for each scenario as well as its evaluation
- Section 3 gives details on the user interface, including mock-ups for each scenario
- Section 4 illustrates an evaluation of a specialized low-power edge processor core
- Section 5 provides a summary and the conclusions of this report

2 Edge intelligence

Edge computing is an emerging paradigm that pushes computing tasks and services from the network core to the network edge [1]. With the push from cloud services and pull from IoT, data is increasingly produced at the edge of the network, therefore, it can be more efficient to also process the data at the edge of the network [2]. Edge intelligence refers to a collection of connected systems, sensors and devices in which data is collected, processed and analysed close to where it is captured in a network. It generally consists of 4 main methods, as illustrated in Figure 2 [3]:

- **Edge caching.** It is responsible for processing and storing incoming data captured by the edge devices via a number of sensors as well as to support intelligent applications for users at the edge
- **Edge training.** A method of discovering hidden patterns in training data collected at the edge or learning optimum weight and biases for models implemented on collected data
- **Edge inference.** Evaluates the performance of a trained model or algorithm on the collected data that is for testing, by computing the outputs on the edge device
- **Edge offloading.** Allows an edge device to delegate any of its tasks, such as edge training, edge caching or edge inference to other edge devices in the network. It's similar to the distributed computing model, in which edge devices form a smart ecosystem

Compared to traditional cloud-based intelligence, which includes devices that collect and upload data to a remote cloud, edge intelligence processes and analyses input data locally [3], [4]. As a result, this has some benefits over the traditional approach:

- **Low latency**. A fast response is required for many applications as may be unable to tolerate the time delay of transmitting data to be processed elsewhere. Edge intelligence is much more efficient in cases where execution of low-latency operations close to the field is required. This is due to the fact that large amounts of data do not need to be transferred to the cloud, which can cause significant network latency
- **Increased privacy.** The use of edge intelligence eliminates the need for data to be transferred and stored on cloud servers. This reduces the risk of data breaches and privacy leaks, which is especially important for applications that handle sensitive data such as citizens' personal information, intellectual property or business secrets
- **Reliability.** It is not always possible to rely on an internet connection, thus edge intelligence offers an alternative to data transmission over long distances between connected devices and remote cloud servers. This is very important for the AMANDA Card that is limited in its long-range communication resources
- **Reduced power consumption.** Power is always a priority for embedded systems, as in several cases they should operate for a long time without being charged. Moving data consumes power, thus a reduction or elimination of data transfer offered by edge intelligence can reduce the overall power consumption. This is highly significant in the case of the AMANDA card

Edge intelligence is a combination of advanced computing and AI, located near devices that generate, acquire and use data. It represents the implementation and integration of developments in industrial monitoring, digital processing, utility administration and telecommunications, as well as cloud computing, data analytics and AI. Edge intelligence moves these capabilities near data that requires rapid review and response, allowing them to be acted on directly or to be filtered in order to transmit only the most relevant bits to the centre [5].

Edge intelligence is a core component of the AMANDA ASSC. The ultra-low power character of the system does not allow for massive amount of data to be transmitted or received. For this reason, low-power algorithms have been designed and implemented to offer intelligent processing at the ASSC level, in support of decisional autonomy under the anticipated environmental monitoring and tracking services.

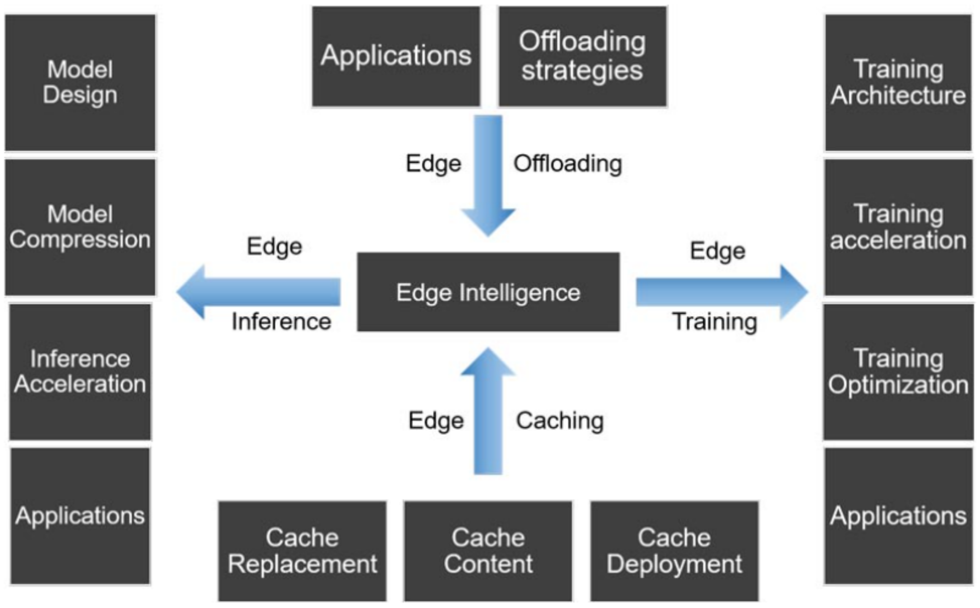


Figure 2 Components of edge intelligence [3]

2.1 State-of-the-Art analysis

The use of AI in edge computing Scenarios is referred to as edge intelligence. Edge intelligence is based on moving AI support away from the cloud and on imparting intelligence at the network edge. Also, it tries to transfer as many learning and inference calculations as possible to the edge, reducing cloud resource needs. AMANDA takes advantage of edge intelligence, incorporating knowledge and effective techniques from a variety of fields, including but not limited to AI, computer architecture and embedded systems, as well as compressive sensing, to optimize the specific implementations for each Scenario of the project and to reduce the power consumption of the ASSC.

AI and ML enhance the ability to extract information from massive amounts of data in order to develop intelligent solutions. An AI/ML-enabled edge cloud management platform can make data-driven inferences, predictions and decisions based on the knowledge acquired from previous data, with minimal human interaction. ML methods can be classified mainly into supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning:

- **Supervised learning.** Supervised learning is a method of training a computer system using input data that has been labelled for a certain output [6]. The model is trained until it can detect the underlying patterns and relationships between the input data and the output labels, enabling it to yield accurate labelling results when presented with never-before-seen data. In supervised learning, the aim is to make sense of data within the context of a specific question. The following are some of the most frequent algorithms used in supervised learning on embedded systems:
    - **Naive bayes (NB).** A simple and effective classification model that is based on Bayes' theorem [7] [8]. It is a group of simple probabilistic classifier algorithms that follow the same principle
    - **Decision Tree (DT)**. A straightforward model that has the benefit of providing criteria for making judgments. It is able to deal with linearly inseparable data and can handle redundancy, missing values, and numerical and categorical types of data. Also, it is negatively affected by high dimensionality and high numbers of classes, due to error propagation [9] [10]
    - **K-Nearest Neighbours (KNN).** A feature similarity-based technique that assigns the class of the nearest set of previously labelled points to a sample

point. The number of neighbours K, the voting threshold (for K > 1), and the training data size all influence the efficiency and performance of KNN. Noise and irrelevant features might reduce its performance [9] [11]
    - **Support Vector Machine (SVM).** A linear classifier that calculates the hyper-plane in an N-dimensional space that separates data points into distinct classes. It is a memory-efficient inference technique that can capture complicated data-point interactions [8] [12]. Also, the algorithm deals well with non-linear problems using kernels that map the original data in a higher dimension
    - **Artificial Neural Networks (ANN).** Mimic the network of neurons that are contained in the human brain. They can learn things with the training phase and then make a decision like humans. Also, ANN can model complex linear and nonlinear problems [13]. Some of the most commonly used ANN on embedded systems include CNN, RNN, GAN as well MLP
- **Unsupervised learning**. Unsupervised Learning is an ML methodology in which the model does not need the users' supervision. Instead, it allows the model to work independently to uncover previously undetected trends and knowledge. It is comparable to the learning that occurs in the human brain through learning new information. Also, it mainly deals with the unlabelled data. The research about unsupervised learning on embedded systems are limited and are restricted to anomaly detection with fraud detection as well automated problem identification [14] [15] [16]. Some of the most used unsupervised learning algorithms are K-mean [17], hierarchical clustering [18], as well density-based algorithm [19]
- **Reinforcement learning**. Reinforcement learning is a technique for an application to learn by experimenting and receiving positive reinforcement (reward) for good responses to events. A reinforcement learning device that detects abnormal sensor readings from a group of machines, for example, might attempt to restore normal sensor readings by increasing coolant flow, lowering room temperature, lowering machine load, and so on. Since learning the action resulted in success, the machine would be able to perform that action more effectively the next time it encounters the same anomalous readings. Some of the most commonly used of Reinforcement learning on embedded systems is for dynamic power management [20] as well for autonomous embedded systems [21] [22]

The main categories of AI, as well as the most common applications on embedded systems, are depicted in Figure 3 below.

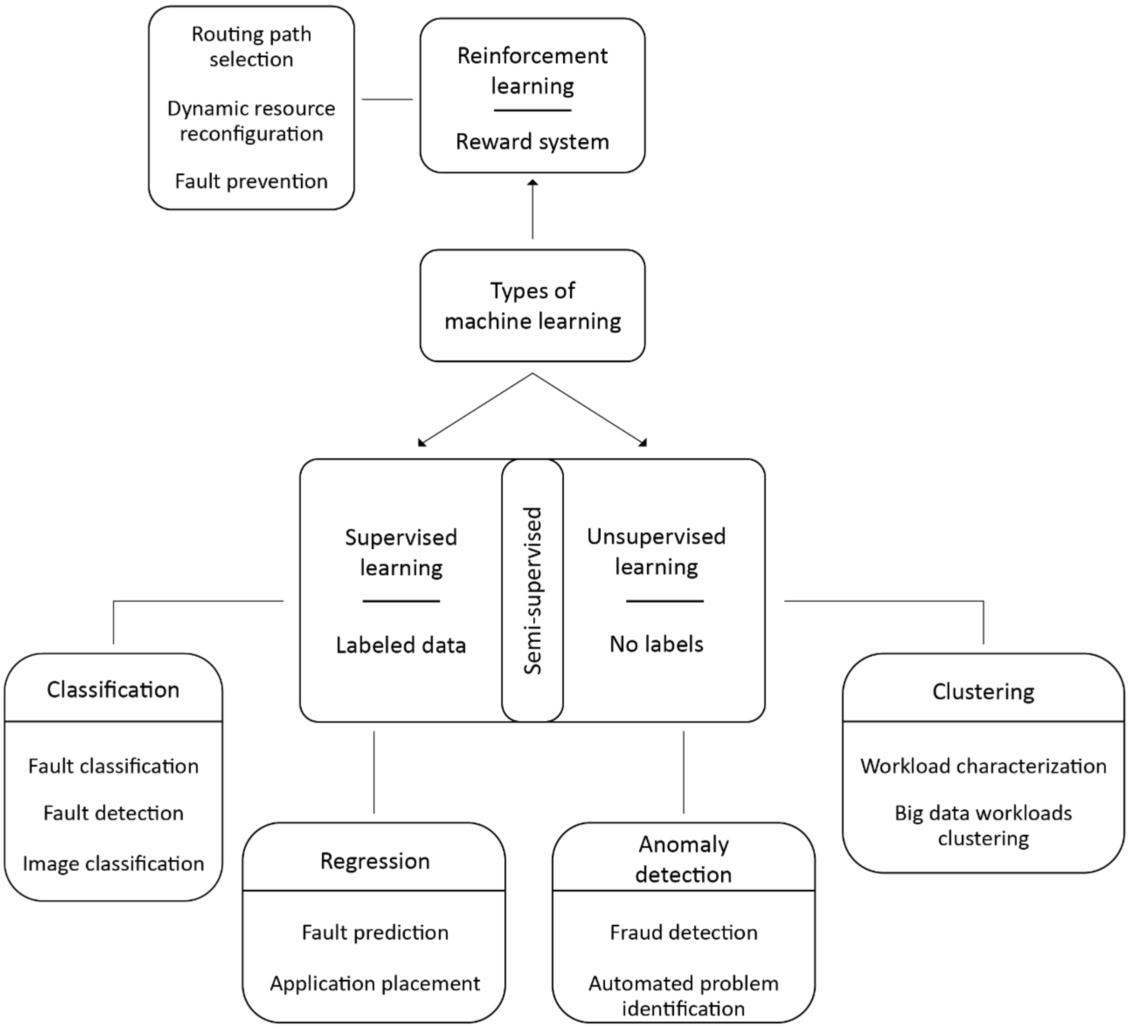


Figure 3 Classification of ML algorithms

Several implementations have been made using supervised learning and are presented in the following Sections. On the other hand, implementations on unsupervised and reinforcement learning are limited. The different Scenarios of the AMANDA project are mainly related to supervised learning techniques using classification methods (attempt to classify the category of a new observation within a set of categories) and regression methods (attempt to find the correlation between variables and to forecast the continuous output variable dependent on one or more predictor variables). The most commonly used algorithms include the following: NB, FZ, DT with Random Forests, KNN, SVM and ANN with MLP, CNN, RNN and GAN.
Table 1 summarizes the advantages and disadvantages of the above popular ML methods and their use cases. More details about the State-of-the-Art for each Scenario can be found in Section 2.3 below.

Method	**Advantages**	**Disadvantages**	**Use cases**
NB	• Does not require as much training data • Is capable of dealing with both continuous and discrete data • Is fast and can be used to make predictions in real-time • Is not affected by irrelevant features	• The assumption of independent predictors • The zero frequency • Is a classification algorithm that cannot be used to predict a continuous numerical value	• Personal thermal comfort [25], [27]
FZ	• Simple and justifiable structure • Deal with inaccurate and uncertain data • Easily to modify it to improve system performance	• Dependent on human knowledge and expertise • Is not always accurate • Validation and verification need extensive testing	• Personal thermal comfort [25] • Fire monitoring [29], [30], [31], [33], [34]
DT	• Less effort is needed for data preparation during pre-processing • Does not require normalization of data • Does not affect by missing values	• Higher time to train the model • Predictions are relatively expensive in terms of complexity and execution time • High memory is needed as all of the training data must be stored	• Personal thermal comfort [25],[27],[28] • Parking spot detection [46], [56] • Transportation conditions [67], [68]
Random Forests	• Reduces overfitting in decision trees • Can handle both categorical and continuous data	• Requires a lot of computing capacity and energy as it creates multiple trees • Requires much time for training	• Personal thermal comfort [25],[27],[28]

	• Does not require normalization of data		
KNN	• Fast model building • Simple implementation • High accuracy • Useful for regression and classification	• Slow prediction, especially with large datasets with a lot of features • Curse of dimensionality • Is sensitive to outliers • High memory is needed as all of the training data must be stored	• Personal thermal comfort [24],[25] • Fall detection [61]
SVM	• Can handle high dimensional data • Is memory efficient • The risk of over-fitting is less • Both unstructured and semi-structured data, such as text, images, and trees, works well	• Low processing capacity with large data sets • Can't easily deal with incomplete and noise data • Final model is difficult to understand and interpret	• Personal thermal comfort [24], [25], [26] • Parking spot detection [44], [46], [56] • Fall detection [62] • Transportation conditions [67], [68], [69]
ANNs	• High accuracy and precise prediction • Can handle incomplete dataset • Suitable for linear and nonlinear data • Can reduce the computational requirements, power consumption and the memory footprint	• High number of features required • Are capable of working with numerical data • Training is expensive due to complex data models	MLP • Personal thermal comfort [25] • Parking spot detection [44], [45], [46], [56] • Vehicle detection [48] • Fall detection [63], [64] • Transportation conditions [66], [67], [68] CNN

			• Fire monitoring [29] • Fall detection [57], [59] • Crowd counting [71], [72] RNN-LSTM • Parking spot detection [43], [45]

Table 1 Summary of popular ML methods

2.2 Edge intelligence architecture

Different techniques, implementation methods and architectures exist for each AI research topic, as detailed in Section 2.1. There are three main approaches for edge intelligence computations, as depicted in Figure 4 [23].

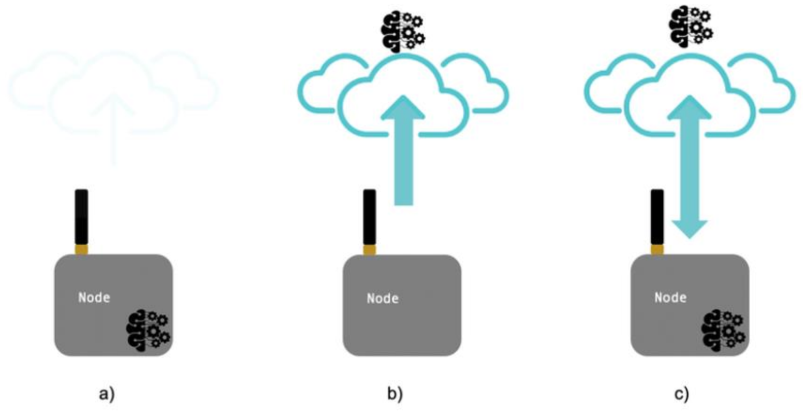


Figure 4 Different approaches of edge intelligence [21]

- **On-device computation.** On-device methods have primarily been proposed in the industrial sector to improve the provisioning of services and applications for mobile devices. AI models are trained offline in a local system or cloud and the export file with the weights of the models are imported to the edge device [24]. Some of the advantages as mentioned above are:
    - Lower latency with faster predictions, as data is held on the IoT network's edges and can be operated on immediately
    - Increased security and privacy as there is no need for data transfer on cloud servers
    - Increased reliability, as there is no dependence on internet connection
    - Reduced operating costs, since the process has a single destination rather than circling from the centre to local drives
- **Edge server-based architecture (Fog computing).** A form of decentralized computing in which data, computational resources, storage, and applications are distributed between the data source (node) and the cloud [23]. The main difference between edge and fog computing is the place where the intelligence and compute power are located. In fog computing, intelligence is at the local area network and data is sent from endpoints to a fog gateway, where it is processed before being sent again. Some of the advantages of this approach include:
    - Bandwidth conservation, as it reduces the amount of data transmitted to the cloud

  - Latency is minimized and overall responsiveness is increased as the original data processing happens near the data
  - Flexibility of storage, as data can be stored locally or retrieved from local drives
- **Distributed computing.** Computing, storage, and networking resources are shared by several systems and operate as a single machine, with the purpose of achieving a single goal. Computers, servers, and workstations may be located in the same data centre or building and linked through a local network, or they may be located in various locations, some very different geographic locations, and connected through a wide area network or the cloud. Compared to the other two architectures the distributed computing may occur overhead. This happens as all of the workstations attempt to send at the same time. Even though this essentially brings desired results, eventually there will be an increase in computing time as well on the system's response time. Some of the advantages include:
  - Scalability
  - Reliability in terms of errors. Distributed networks can be more reliable than single systems
  - Efficiency in terms of energy consumption, load balancing or overhead

The on-device computing methodology was chosen to design the edge intelligence component of the AMANDA card. The main reasons behind the use of this methodology was to cover the low-power requirements of the ASSC with the best possible precision of the AI algorithms, the low latency as well as fast predictions. Also, the on-device computing architecture contributes to the cyber security and privacy of the ASSC, with more information included in **Deliverable D4.2 - AMANDA Cyber Security Mechanisms**. The AMANDA edge intelligence component, as presented in Figure 5 below, includes two phases of execution:

- Training phase
- Prediction phase

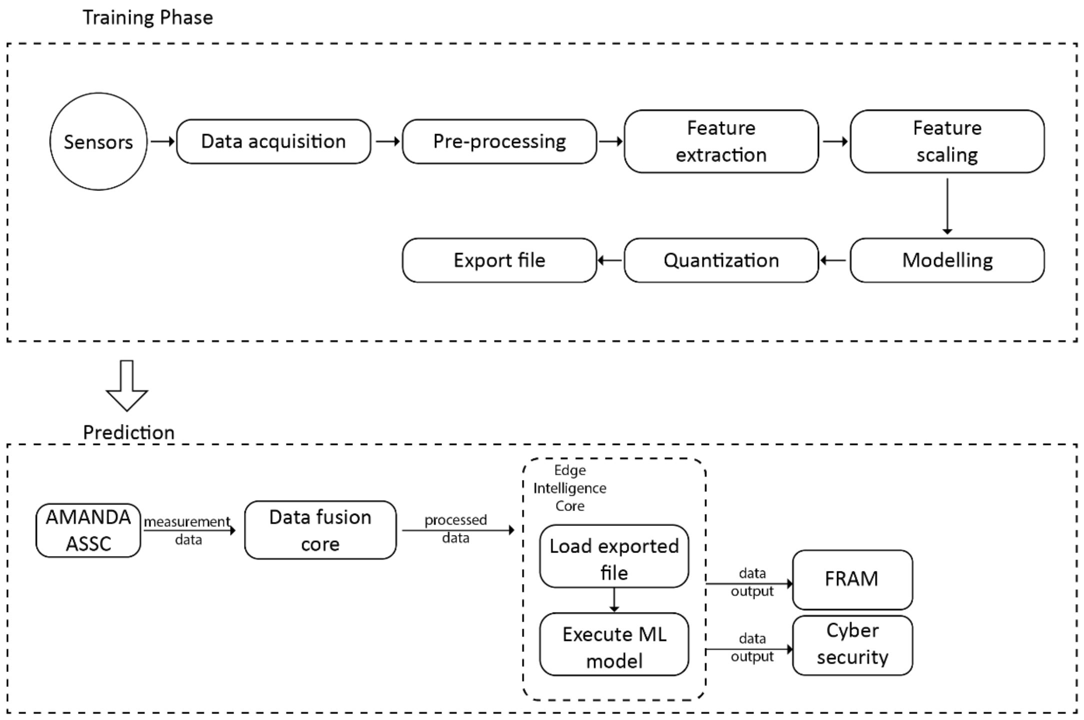


Figure 5 AMANDA edge intelligence architecture

As the training phase is a demanding process, systems with ample resources in terms of memory requirements and computational complexity are needed. The AMANDA ASSC, as described in **Deliverable D1.9 - Architecture design of the AMANDA system delivered (for both breadboard and integrated/miniaturised system) v2,** is a system with a restrained MCU that has limited capabilities in terms of performing complex computations, thus a high-end computer node was used for the training phase. The training process is described by the following procedures:

- **Data acquisition** is the method of measuring signals that quantify real-world physical conditions and converting the corresponding observations into digital numeric values that a device can manipulate. For the AMANDA project, the data acquisition process includes raw data captured from sensors integrated into the ASSC
- **Data pre-processing** is the process of raw data preparation to be used in an ML model. It is the first and most important step in building an ML model and includes the following steps for feature extraction and feature scaling. The input raw data acquired by the sensors can contain missing values or outliers which should be subsequently removed in order to not affect the accuracy of the predictions. Also, some features may not provide useful information to the model and therefore should be omitted. This is achieved through the process of feature extraction
- **Feature scaling** is part of data pre-processing and is a technique to normalize or standardize a collection of independent variables or features of data. There are several feature scaling techniques, but three methods are used for the AMANDA project, standardization and mean normalization, as they need less execution time and they have the best overall accuracy
    - Standardization is a method that can reduces the numeric ranges between the different data attributes while it helps to avoid numerical difficulties during calculations. The process is described by the following equation:

$$X_{scaled} = \frac{X - mean}{Standard\ deviation}$$

    - Mean normalization is a method that changes the values of numeric columns in a dataset to a standard scale while preserving variations in value ranges. The equation for mean normalization is given as:

$$X_{normalized} = \frac{X - average(X)}{\max(X) - \min(X)}$$

    - Min-max normalization is a method that transforms the value of each numeric feature into a decimal between 0 and 1. The minimum value of the features gets transformed into 0 while the maximum value gets transformed into 1. The equation for min-max normalization is given as:

$$X_{normalized} = \frac{X - min(X)}{\max(X) - \min(X)}$$

- **Modelling** is the phase where the selection of the predictive model has to be performed according to the objective that the AI model aims to achieve. There are a variety of models to choose from, including supervised methods with regression and classification models as well as unsupervised methods with clustering models, as mentioned above. The choice of models largely depends on the type of data that is being used for training and prediction. This phase also includes the training procedure of the AI algorithm by feeding the input data captured by the sensors with the parameter tuning as well as its evaluation using an evaluation dataset. The following Sections provide more information about the ML models that are used for each individual AMANDA use case Scenario

- **Quantization** is a technique that reduces the number of bits that are used to store the values of weight and activation functions of ANN, converting the floating-point values to fixed point integers. A quantized model requires less storage space compared to the initial model and the operations between weights are faster as the number of bits is smaller but at the expense of the model's performance. Thus, there is a trade-off between the number of bits that represent weights and the accuracy of the model and some information may be lost [25]. Typically, the values of weights of an ANN are stored as 32-bit floating-points while a quantized model is represented with a small number of bits, such as 16 or 8 bits or even a single bit. The mapping of float values to integers is defined as:

$$q_{uint8} = \frac{r_{float32}}{s_{float32}} + z_{uint}$$

  Where *q* is the quantized representation, *r* the real value, *s* the scale of initial data and *z* the offset
- The output of the training phase is a model file that contains multidimensional arrays with the weights of the trained model and it will be used to make real-time predictions on the new data inputs from the sensors. Typically, the model files are stored in the HDF5 format. HDF5 is an open-source file format that supports large, complex and heterogeneous data. The file can contain two different type of objects:
  - Datasets, which contain array-like collections of raw values as well metadata that describes the data
  - Groups, which organize data objects and includes a root group that can contains other groups or be linked to objects in other files

  The HDF5 format is self-describing and thus each file, group or dataset can have associated metadata that describe in precise the type of the data. Also, HDF5 allows compression filters to be applied to a dataset to minimize the amount of space the dataset consumes

The prediction phase is executed by the MCU of the AMANDA ASSC. It is performed by the edge intelligence component and interacts with two software components of the card, the data fusion and the cyber security component.

- **Data fusion component.** Details on the component have been provided in **Deliverable D2.4 - AMANDA Data fusion optimization engine.** In summary, data fusion is the process of integrating multiple data sources to produce more consistent, accurate, and useful information compared to the information provided by any individual data source. Data fusion is a core component of the AMANDA system. The ultra-low energy character of the system does not allow for massive amount of data to be transmitted. Effort should be made to achieve a good equilibrium between data processing and assessment that will be executed in the card and data transmitted from it. The focus of the AMANDA project is sensor fusion which relates to combining sensory data or data derived from disparate sources such that the resulting information has less uncertainty than would be possible when these sources are used individually. 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. Thus, multi-purpose and lightweight data fusion

& optimization engine was designed and developed to support the low-power and intelligent capabilities of the AMANDA ASSC

- **Cyber security component.** Oversees the authentication mechanisms of the ASSC and is detailed in **Deliverable D4.2 - AMANDA Cyber Security Mechanisms**. The primary goal was to build a complete security module containing different security techniques, in order to protect the AMANDA ASSC from malicious users or attacks. A lot of effort has been put to make these mechanisms compliant with the constrained capabilities of the AMANDA ASSC, in terms of processing capabilities and memory capacity. Several authentication mechanisms have been deployed to verify if user access is authorized. The authentication process consists of four different stages: voice analysis, face recognition, iris identification and user verification. The three first concern the collection of user biometric factors and then, the fourth stage consists of a combining method of the above, so as to conclude whether the user is granted access or not. The collection of the biometrics is done with the use of the sound and image sensors of the ASSC

The AMANDA high-level modular architecture is visualized in Figure 6. The workflow can be summarized as follows. The utilized environmental sensors wake up from the MCU, gather respective readings including temperature, humidity and CO_2 and the raw data is subsequently delivered to the MCU via the respective serial communication protocol, I^2C or SPI. The readings are gathered at different timings, from 5 to 60 seconds, depending on the specific AMANDA Scenario. Then, the data fusion optimization engine transforms the data to a more usage friendly format, such as floating-point or Q-number format, that will be used as input in the edge intelligence core. The role of the edge intelligence module is to further process the data to ensure that the required accuracy of the decision-making system is reached. As a last step, the data is passed to the cyber security software module to ensure that no sensitive data will be wirelessly transmitted to guarantee the security of the end user's information.

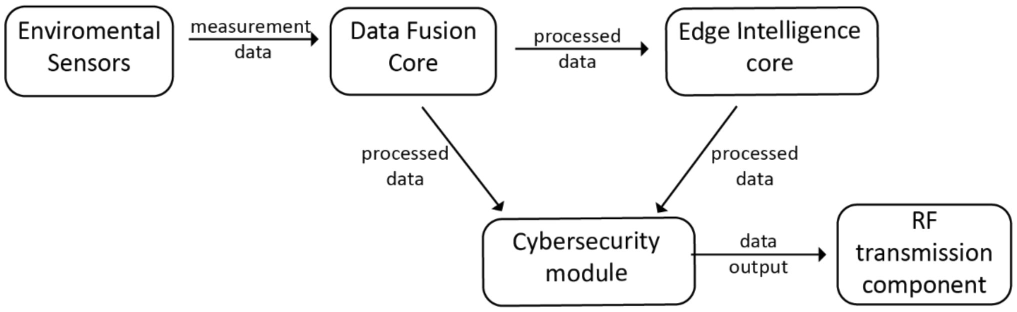


Figure 6 AMANDA High-level modular architecture

The edge intelligence core consists of two elements, as illustrated in Figure 5:

- **Load exported file.** The file is the result of the training phase into AMANDA card. It contains multidimensional arrays with the quantized weights of the trained model and they will be used for the prediction of the new state. The size of the quantized weights depends on the specific AMANDA Scenario with an emphasis on low-power and low-memory requirements
- **Execute ML model.** The second part of the edge intelligence core receives the data from the data fusion component and using the quantized weights from the previous element make useful predictions and decisions for each Scenario of the AMANDA card. The output of this element can be stored in the FRAM and then processing by the cyber security component

2.3 AMANDA Use Cases

This Section describes the architecture of the edge intelligence component from a functional viewpoint, focusing on the way the different algorithms are integrated into each use case Scenario. Such a viewpoint characterizes the fundamental organization of the system, describes the responsibilities of its functional elements, in the form of the edge intelligence mechanisms and defines their primary interactions with other elements.

The edge intelligence component's functional view structure model includes its functional elements, interfaces, connections, and external entities. An extended description of a function view of a software and hardware system is presented in **Deliverable D1.9 – Architecture design of the AMANDA system v2.**

For each of the use case Scenario, a State-of-the-Art analysis is presented, a description of the main functionality of the edge intelligence component is summarised, the dependencies of the edge intelligence core with other external components are detailed and experimental results are provided.

2.3.1 Scenario SC01: Environment and thermal comfort monitoring

2.3.1.1 State-of-the-Art

An IoT system to estimate personal thermal comfort is presented in [26], using a combination of environmental sensors and ML algorithms. The system consisted of an Arduino Mega 2560 connected to a Raspberry Pi, which operated as a gateway for sending the Arduino's collected data to the cloud. Also, different ML techniques were used that included classification methods including Logistic Regression, SVM, Linear Discriminant Analysis, Quadratic Analysis and KNN as well as different regression methods, such as Support Vector Regression and Kernel Ridge Regression. A set of 530 data points were used for the evaluation. It was determined that SVM was the algorithm with the best performance. There is no mention of the memory requirements and power consumption of the system. A low-power and low-cost system was also presented in [27] using a mobile phone and body sensors to monitor physiological parameters such as heart rate, activity, skin temperature, and electrodermal activity. To predict the personal thermal comfort the authors used four groups of ML algorithm that includes linear method such as Logistic regression, non-linear methods such as ANN, SVM, NB and KNN, trees and rules such as DT and Rule-Based Classifier as well Ensembles of Trees such as Random Forest and Stochastic Gradient Boosting. The ensemble algorithms had the highest median prediction power with an accuracy close to 73% while no information is provided on the power consumption of the system.

An ML thermal comfort approach is described in [28] that uses a series of connected sensors to predict a personalized Thermal sensation index by monitoring hand skin temperature, pulse rate, and ambient air variables. Measurements of air velocity, air temperature, relative humidity, skin temperature as well pulse rate were used for the implementation. Also, the suggested technique is data-driven and adaptable, as well as providing real-time data. The system is based on an ATmega 328 and the ML algorithm that was used was the SVM classifier. In [29], a wearable small device the Microsoft Band 2 is used in conjunction with a Hobo Data Logger UX100 and temperature and humidity sensors to model personal thermal comfort. Without the need for a subjective response, ML algorithms were successfully applied to preprocessed data to predict each subject's comfort level. ML algorithms such as AdaBoost classifier, DT, Gradient boosting classifier, Random Forest classifier and SVM were used on three different dataset and the model with the best accuracy was the SVM. However, the proposed thermal comfort model work can be considered limited due to the accuracy of the used sensors, application usability, and small data size.

A system to improve the thermal comfort of occupants is presented in [30] using as a classification model the Random Forest algorithm with an overall accuracy close to 80%. Also, the system was implemented on a Microsoft Band 2 without providing any information on the power consumption. Table 2 provides a summary of the State-of-the-Art.

Papers	Methods	Implementation
[26]	Classification and regression	Arduino mega and Raspberry Pi
[27]	Linear, non-linear, trees and rules and ensembles	mobile phone
[28]	SVM	ATmega 328
[29], [30]	AdaBoost, DT, Gradient boosting classifier, Random Forest classifier and SVM / Random Forest	Microsoft Band 2

Table 2 SC01: Summary of the State-of-the-Art

2.3.1.2 Description

The ASSC is installed outdoors, or in a room or is wearable and can provide measurements related to heating, ventilation and air conditioning. The ASSC can also serve as a weather monitoring station. Moreover, the wearable version is used for thermal comfort monitoring. The ASSC automatically informs a room controlling system to adjust the heating, ventilation or air conditioning in order to achieve the optimal environmental conditions as predefined by the user. An application installed in an end user's smartphone, projects the rooms or outer environmental conditions in real-time as well as the thermal comfort levels of the users. Every 15 minutes, the card measures CO_2, temperature, humidity, atmospheric pressure as well as light intensity. It subsequently transmits the results to a user's smartphone or to the room controller using BLE ADV events. Also, the end-users must define some initial parameters, before the AI module of the AMANDA card is enabled. The clothing insulation is a float value with a range between 0.5 and 1, that describes the amount of clothing required by a resting human to maintain thermal comfort at a room temperature of 21°C, the metabolic rate with a range between 1 and 1.3, the building type (Office, Senior Centre, Others) if the ASSC is installed in a room as well the cooling system of the room (air-conditioned without operable windows, naturally ventilated without mechanical cooling or mixed-mode).

2.3.1.3 Main functionality

The SVM algorithm was chosen for this Scenario as it has a low latency real-time prediction, requires a minimum amount of memory and has the best accuracy.
The process for the implementation of Edge Intelligence for SC01 includes two stages:

- Training on an external node
- Prediction on the AMANDA ASSC

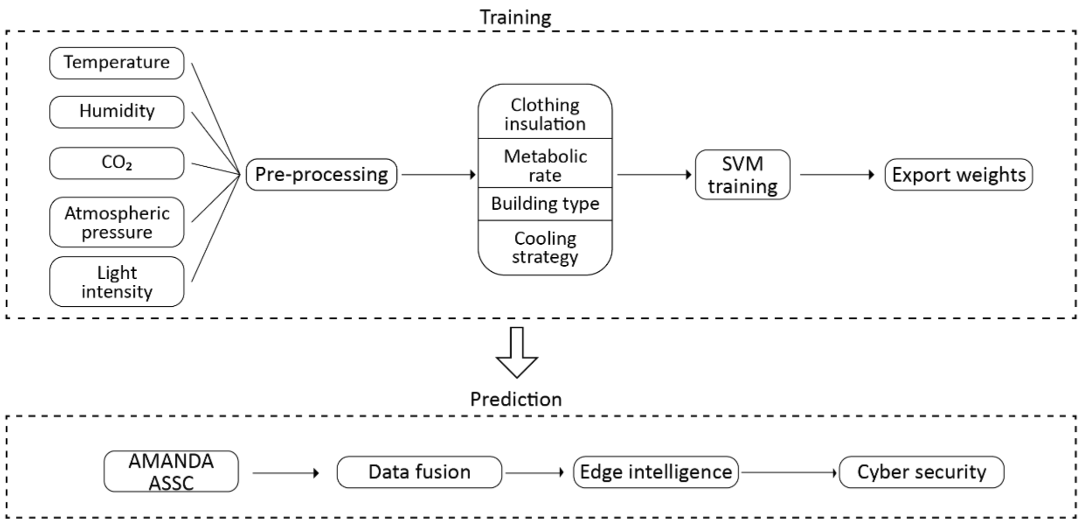


Figure 7 Scenario SC01: Environment and thermal comfort monitoring

Figure 7 presents the system workflow with the first 3 steps (Pre-processing, SVM training, export weights) taking place in an external node while the remaining are executed on the AMANDA ASSC.

- **Pre-processing.** The data used in this Scenario is categorized into numeric (CO_2, temperature, humidity, atmospheric pressure, light intensity) and categorical features (clothing insulation, metabolic rates, cooling strategy). Before the training of algorithms starts, missing values or outliers are removed and a standardization method is applied in the numeric features. Table 3 summarizes the descriptive statistics of the numeric features. Subsequently, with a ratio of 0.2, the input data is divided into two datasets, one for training and the other for testing
- **SVM training.** Before the SVM model can be trained, there are a few parameters that must be set. The kernel is a function that transforms the input data into the required form. There are different types of kernel functions that can be used for different applications but for SC01 the (RBF) was used as it had the best performance combined the numeric and non-numeric features. The C parameter is a feature that can affect the performance of the algorithm. It is a parameter that adds a penalty for each misclassified data point. When C is small, the penalty for misclassified points is minimal, so a decision boundary with a large margin is chosen at the expense of a greater number of misclassifications. While, when c is large, SVM tries to reduce the number of misclassified instances as a result of the high penalty, resulting in a decision boundary with a narrower margin. Thus, value 2 was used for this Scenario. Subsequently, the use of RBF requires configuring one more parameter of the SVM algorithm the gamma value. Gamma is a hyperparameter that decides that how much curvature we want in a decision boundary. A high value of gamma means more curvature while a low value less curvature. Different values were tested and was determined that the value with the best accuracy was 1
- **Export weights.** The SVM model is a lightweight algorithm and so no quantization method needs to be applied to its weights. Thus, the model weights can be used in the next step for prediction

The prediction of SC01 is described by the feature thermal sensation that is a categorical feature with a range of values between -3 and +3 with -3 to mention that the user feels very cold while with 3 that it gets quite warm. Also -2 refers as cool, -1 as slightly cool, 0 as neutral, 1 as slightly warm and 2 as warm. The Edge intelligence module in this Scenario includes the data acquisition as well feature scaling of the input dataset. The numeric input data from the environmental sensors must be on the same scale as the training data. Subsequently, the exported weights of the training phase are loaded from the MCU. The output of the edge intelligence node contains an integer value from -3 to 3.

	Temperature	**Humidity**	**CO_2**	**Atmospheric pressure**	**Light intensity**	**Clothing insulation**	**Metabolic rate**
Type	Float	Float	Integer	Float	Integer	Float	Float
Count	30771	30771	30771	30771	30771	30771	30771
Mean	25.62	49.56	775.45	1020.45	654	0.67	1.20
Std. Deviation	6.13	14.97	87.34	1.25	64.5	0.32	0.18
Min	13.55	10.71	251	1000.1	32	0.00	0.70
25%	21.10	38.43	458	1005.2	540	0.47	1.10
50%	25.64	48.72	543	1020.35	653	0.60	1.20
75%	29.85	60.00	884	1045.12	712	0.78	1.22
Max	45.22	95.30	1.454	1070.4	1302	2.70	3.80

Table 3 SC01: Descriptive statistics of the dataset

2.3.1.4 Dependencies to other components

As can be seen from Figure 7, data are exchanged between the data fusion, cybersecurity and edge intelligence module. The AMANDA system receives raw data from the temperature, humidity, CO_2, atmospheric pressure and light sensor which are used as an input to the data fusion optimization engine. Subsequently, the output is used as input for the edge intelligence module which adapts an SVM model to predict the thermal sensation. Lastly, the data is passed to the cybersecurity software module to ensure that no sensitive data will be wirelessly transmitted to guarantee the security of the end user's information.

- **Data fusion optimization engine**. For SC01 a Probability-based methods in the form of a Bayesian network was used in order to express the dependency between random input variables and construct the relationships between the different data points. Also, a pre-processing of the data was applied transforming the raw data to a float-point format and organizing the input data into data blocks
- **Cybersecurity software module.** The output of the edge intelligence includes a categorical variable with a range of values between -3 and +3 which is encrypted by the cybersecurity module using a PLS-based encryption scheme

2.3.1.5 Experimental Results

The SVM model in SC01 had an accuracy close to 91.4% with an f1 score close to 90.2%. The f1 score is a metric that estimates how well each class is predicted. The dataset used for evaluation contained 6155 instances for all possible values of thermal sensation while the initial dataset used for training contained 24600 instances. Table 4 illustrates the predicted and true labels for each class. As can be observed the instances that have been successfully classified are presented with blue colour while the remainders refer to incorrect predictions.

	-3	1305	11	0	0	5	2	0
	-2	19	970	21	3	2	4	5

Predicted label	**-1**	10	37	518	3	4	4	4
	0	3	13	8	250	11	4	2
	1	0	13	8	5	572	22	15
	2	3	16	3	3	11	965	37
	3	0	12	5	3	5	22	1228
		-3	**-2**	**-1**	**0**	**1**	**2**	**3**
	True label							

Table 4 SC01: Confusion matrix

The SVM model has a computational complexity of 600 multiply-accumulate operations per input and a decision execution time of about 35 milliseconds. The memory requirements of the SVM model for SC01 was about 15.87kB adding the size of each input data. Specifically, input data of temperature, humidity, atmospheric pressure, clothing insulation and metabolic rate are float values and need 4 bytes for every instance. While, input data of CO_2, light intensity, building type as well cooling system are uint16 and it requires 2 bytes per instance.

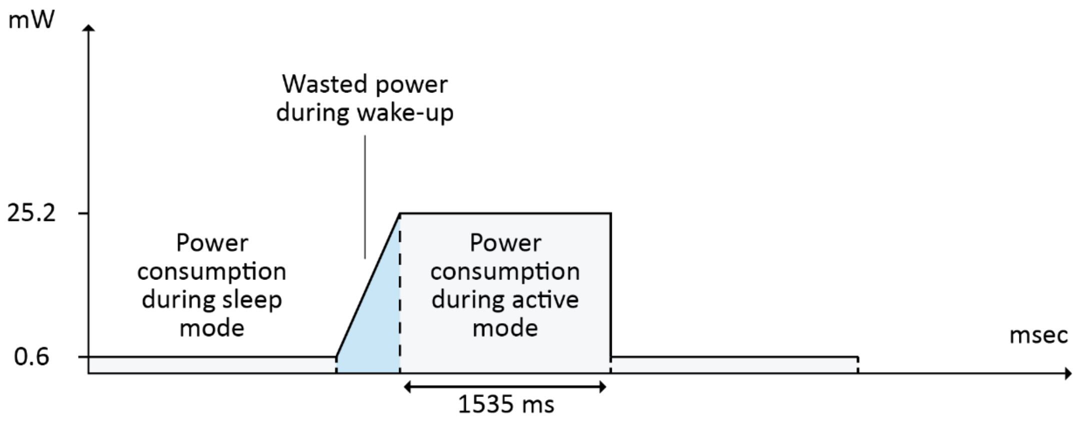


Figure 8 SC01: Power consumption

The amount of power used in active mode (data collection and processing) and standby or sleep mode is the overall power consumption. Figure 8 presents the power consumption of the AMANDA card in active mode as well in sleep mode for SC01. When the card remains in sleep mode the power consumption is about 0.6mW as all the sensors and the MCU has negligible consumption. In active mode, an increase in consumption is observed with a max value of 25.2mW. Also, the execution time was 1535 msec, with 1500 msec for measurements of input data and their transmission to edge intelligence module and 35 msec for the decision of the AI model.

2.3.2 Scenario SC02: Fire monitoring

2.3.2.1 State-of-the-Art

An increasing number of publications on fire monitoring and warning systems have been published in recent years, but literature on their application on embedded systems is still limited. Table 5 provides an overview of different embedded fire detection systems, with comparisons made in terms of the number and type of sensors used, classification algorithms, and implementation platforms. Environmental sensors such as temperature and humidity, as well as image sensors, are among the sensors used. The classification algorithms used are based on the FZ technique, if-then rules, ANN, CNN and the Dempster-Shafer theory (DS). The algorithms have been implemented on the Arduino Uno, Raspberry Pi, Mica Z, Beaglebone and MSP430 platforms.

A FZ-based multisensor fire detection system was described in [31] and CNNs were trained for indoor and outdoor Scenarios. Multiple fire signatures, such as fires, smoke, and heat, as well as photos from surveillance cameras, were used by the system. It was developed and tested on a Beaglebone microprocessor, with a CNN algorithm that was 94% accurate and an FZ algorithm that was 90% accurate. In [32], a device based on FZ was developed to detect the presence of fire using an Arduino Uno. Sensors for flame, temperature and smoke were used to capture data.

In [33] the authors developed and deployed a Wireless Sensor Network (WSN) for fire detection in indoor spaces using multiple sensors. To detect fire, smoke, gas, and temperature sensors were used, with a Raspberry PI operating as the main processing device. The deployed sensors' energy consumption was also calculated, with a minimum of 0.5mW and a maximum of 60mW per hour reported.

A low-rate, low-power sensor node for early detection and monitoring of fire was designed and evaluated in [34]. Using a temperature and humidity sensor, the authors developed two algorithms on an MSP430 microcontroller. The first algorithm was based on a comparison method, while the second was based on DS theory. On a low-power ATmega1281 processor, the same algorithms were used in [35]. The first algorithm used temperature, humidity, and light sensors and was based on a threshold method. The second used DS with the nodes being equipped with temperature and humidity sensors. In [36] presented a method that combined WSN and ANN to detect forest fires. Data was collected using low-cost sensor nodes, such as temperature, light, and smoke sensors, and the collected data was encoded as an input to ANN. For various cases, the accuracy ranged from 87% to 99%, while the system's power consumption was not recorded.

Papers	Sensors	Type of sensors	Classification	Implementation
[33] [36]	3	Environmental and image	FZ	Raspberry Pi
[31]	4	Environmental and image	FZ & CNN	Beaglebone
[32] [35]	3	Environmental	FZ	Arduino Uno
[34]	3	Environmental	DS Theory	MSP430

Table 5 SC02: Summary of the State-of-the-Art

2.3.2.2 Description

The ASSC in SC02 is installed in a room or outdoors and will be able to detect through the use of edge intelligence algorithms the presence of fire and will alert in real-time the occupants and the building's management office or the owner of the outdoors space. The card will transmit details about the incident to the building's control system, via BLE and to the closest fire department, via LoRa. The features that are used for this Scenario includes the CO_2 data with a range of 0 – 1500ppm, the temperature with a range of 0-90°C, the humidity with a range of 0 - 100 and the VOC data with a range of 1 - 3mg/m^3. Until the concentration of CO_2 crosses a pre-set threshold, the device remains in sleep mode with the lowest power consumption. When this occurs, the MCU will wake up from its sleep mode due to the CO_2 sensor interrupt, and input data for temperature, humidity, CO_2, and VOC will be available in order to check for the presence of fire using AI. In case of fire, data is transmitted using several LoRa frames and BLE frames to increase the probability of getting the important message through. The use of AI and CO_2 measurement requires more energy. However, data is transmitted only if there is fire. It means that the wireless part's energy consumption is minimized since no transmission is expected during regular periods when there is no fire. The cycle duration can be increased if several cards are in use, because of the overlap, which leads to an improvement of the energy performance.

2.3.2.3 Main functionality

The ANN was used to check for the presence of fire and specifically the MLP, as a model with low memory requirements, high-accuracy predictions and fast predictions in real-time was needed. Figure 9 illustrates the system workflow with the first 4 steps (Pre-processing, MLP training, quantization, export weights) taking place in an external node while the remaining on the AMANDA ASSC.

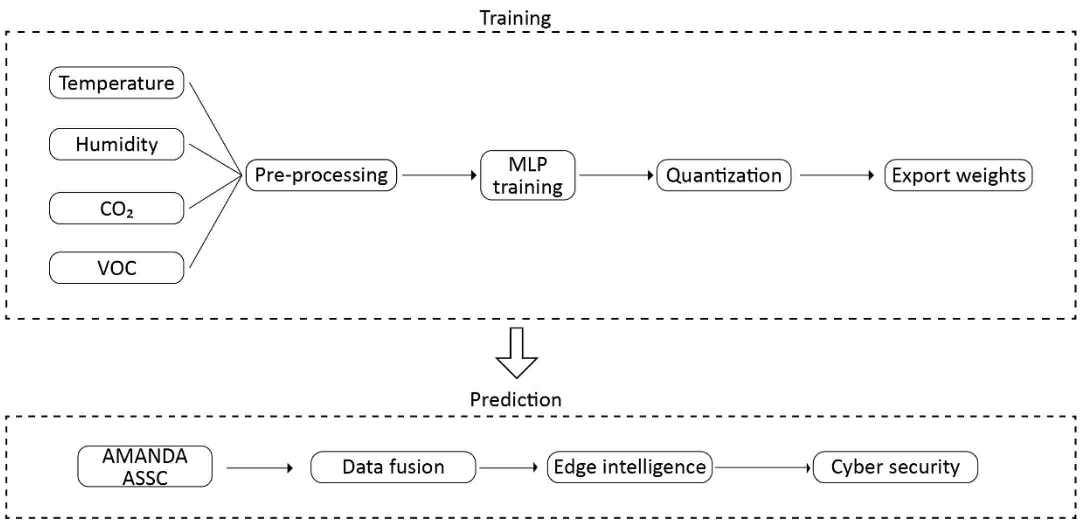


Figure 9 Scenario SC02: Fire monitoring

- **Pre-processing.** Data that contains missing value or outliers are removed. A standardization method is applied shifting the distribution of each attribute to have a mean of zero and standard deviation of 1. The descriptive statistics of the numeric features are shown in Table 6. Also, the input data is divided into two datasets, one for training and the other for testing with a ratio of 0.2
- **MLP training.** Different methods with varying numbers of hidden layers and units were tested, but the accuracy remained constant as the number of layers and units increased. Since the primary goal was to build a model with the smallest memory footprint, lowest power consumption, and maximum accuracy, two hidden layers were used. Also, the rectified linear unit was used as an activation function in both hidden layers because it had the best convergence efficiency. So, when the input values of a model are negative or approach to zero, the network is not able to learn from the training phase because the gradient of the data will be zero. On the other hand, when the output of ReLU is zero, this node will be deactivated. Thus, this method

reduces computational cost as only non-zero neurons will be activated. Therefore, using ReLU allows the network to converge very quickly even with two hidden layers. Since this was a supervised classification problem with only two values, zero for no fire and one for fire, the sigmoid feature was used for the output layer. The MLP was trained using the Adaptive Moment Estimation, an optimization algorithm that updates network weights iteratively based on training data. It used in the training phase to reduce the losses of ANN, modifying a number of attributes such as weights and learning rate

- **Quantization.** Quantization is a technique that reduces the number of bits that are used to store the values of weight and activation functions of predictive algorithms, converting the initial 32-bit floating-point values, to fixed-point integers. For this Scenario, 8 bits for quantization are used as the quantized model had a small memory footprint and a better classification performance
- **Export weights.** The output from the above training phase contains the weights of the model as multidimensional arrays that can be used during the next step for prediction. The size of the exported weights is dependent on the AI algorithm that has been selected as well from the quantization technique followed

The prediction of fire occurrences is a procedure that takes over on the MCU, using the weights of the ANN which have been trained in the previous phase. The Edge intelligence module in this Scenario includes the data acquisition as well feature scaling of the input dataset. The input data from the environmental sensors must be on the same scale as the training data. Thus, a standardization method was applied. Subsequently, the quantized weights of the training phase are loaded from the MCU. The output or prediction of the edge intelligence node contains 1 for fire incidents and 0 for no fire. In order to train and evaluate the MLP model a set of data has been collected using the environmental sensors. The raw data, which describes the two possible events of fire and no fire, was collected at 10-second intervals in an office with a temperature of 23°C and relative humidity of 32%. A small fire incident was simulated. Approximately 4500 data points have been collected in total from the temperature, humidity, CO_2 and VOC sensors, with a precision of two decimals. The dataset was therefore balanced since it had similar numbers of instances from the fire and no fire classes.

	Temperature	**Humidity**	**CO_2**	**VOC**
Type	Float	Float	Integer	Integer
Count	4530	4530	4530	4530
Mean	28.70	31.66	828.84	188.01
Std. Deviation	1.04	3.45	697.81	695.38
Min	26.07	23.46	0	0
25%	28.15	29.61	447	7
50%	28.59	30.76	530	19
75%	29.17	32.84	895	74
Max	45.54	49.66	16400	17023

Table 6 SC02: Descriptive statistics of the dataset

2.3.2.4 Dependencies to other components

The prediction phase in Figure 9 describes the dependencies between the data fusion, edge intelligence and cybersecurity modules. The AMANDA system receives raw data from the temperature, humidity, CO_2 and VOC which are used as the input to the data fusion optimization engine. The output from the data fusion module is used as input for the edge intelligence module. Then, the output from the edge intelligence module and specifically from the MLP model, which contains 1 for fire incidents and 0 for no fire is passed to the cybersecurity software module.

- **Data fusion optimization engine**. For SC02, the Dempster-Shafer theory was used in order to extract new features and in order to increase the need for data completeness of the Scenario since it combines different sensor sources to ensure that the system has enough details to make a decision
- **Cybersecurity software module.** The output of the edge intelligence module includes a binary variable which is encrypted by the cybersecurity module using a PLS-based encryption scheme

2.3.2.5 Experimental Results

The MLP model with an 8-bit representation of weights had an accuracy of 94.2% with an f1 score of 97%. The f1 score was affected by an incorrect classification of class 0 values to class 1 (fire occurrences). In these cases, the system generates false alarms, although this form of mistake has less implications than the reverse situation, where the alarm is not activated in the event of a fire. Also, Figure 10 presents the training and validation accuracy of the model. For the training phase, almost 80% of available input data for the sensors were used while the other 20% for the testing or validation phase. On both the training and evaluation datasets, as can be seen after 80 epochs, exceptional accuracy has been achieved.

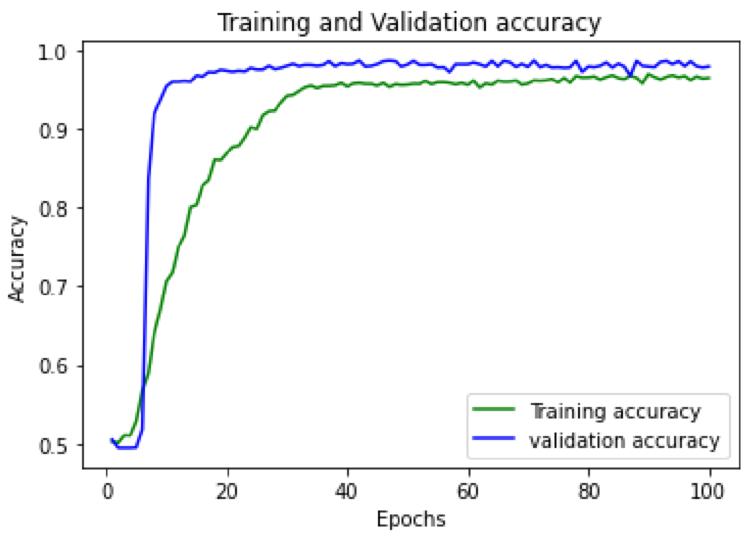


Figure 10 SC02: Accuracy at each epoch for train and validation dataset

To evaluate the performance of the proposed model for SC02, an AUC - Receiver Operating Characteristics (ROC) curve was used. The ROC is represented by a probability curve with the false positive rate on the X axis and the true positive rate on the Y axis. The AUC measures a classifier's ability to correctly divide a dataset into two classes. It's also useful as a ROC curve summary. The range of acceptable values is [0-1], with 1 indicating that the model is a perfect classifier.

Figure 11 presents the AUC - ROC curve. The solid line describes the quantized model with 8 bits, while the dashed line the initial model which is represented with 32 bits. With values of 0.97 and 0.94, respectively, the performance of both variations was very close, with a small decrease for the model with 8bits.

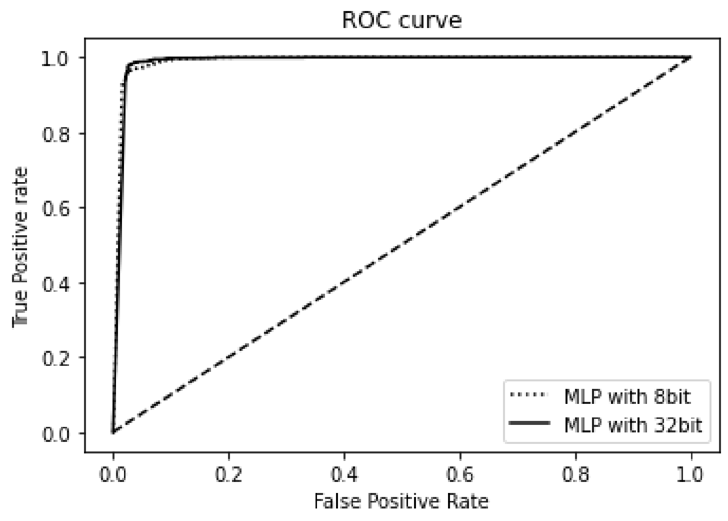


Figure 11 AUC-ROC curve for SC02

The MLP model has a computational complexity of 1.8K multiply-accumulate operations for each input and an execution time of approximately 43msec for each decision. In this Scenario, the quick execution time is particularly critical because, in the event of a fire, there should be an instant warning for immediate action. The memory requirements of the MLP model in the AMANDA card was about 25.65kB adding 4 bytes for every feature, as the input data from the temperature and humidity sensor are float values, while from the CO_2 and VOC are uint16.

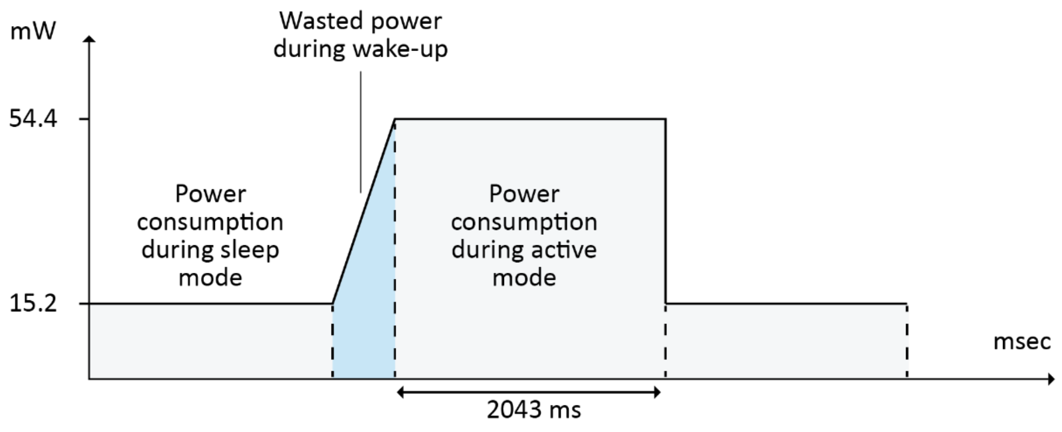

mW
Wasted power during wake-up
54.4
15.2
Power consumption during sleep mode
Power consumption during active mode
msec
2043 ms

Figure 12 SC02: Power consumption

Figure 12 illustrates the power consumption of the AMANDA card in active mode as well in sleep mode. The card remains in sleep mode until the concentration of CO_2 crosses a pre-set threshold or the 10-second interval has elapsed. In this mode, the power consumption was about 15.2mW. When the card “wakes up” from the sleep mode an increase in consumption is observed with a max value of 54.4mW. The execution time was about 2043msec, with 2000msec for the measurements and 43msec for the execution of the edge intelligence module.

2.3.3 Scenario SC03: Continuous occupancy monitoring in a parking lot

2.3.3.1 State-of-the-Art

The number of cars (or personal motor vehicles in general) is ever-increasing in the world, especially in developed countries and regions, where people can afford a car. It is not unusual that a family has more than one private car, and there are also many company cars available for their employees (e.g., for business trips or for personal use as well). The statistics provided by Eurostat indicate that there are about five hundred cars per thousand inhabitants [37]. This causes a huge problem not only for transportation infrastructure, but also for finding available parking spaces. People go anywhere by driving their own car even only for a short time, then they need to park the car. This problem gets even worse in cities, where it sometimes takes a huge effort to find a place to park a vehicle. It increases other traffic problems, such as traffic jams and air pollution in city centres. One of the fundamental problems underlying within the Smart City concept is the reduction of traffic congestion in the city. Knowing that drivers spend an average of 7.8 minutes looking for a free parking place, the development of smart parking systems in order to find free parking spaces more efficiently could contribute to minimizing traffic congestion. Such problems are even more emphasized by knowing that approximately 30% of daily traffic congestion is caused by drivers looking for a free parking space [38]. It has been reported [39] that more than 45% of total traffic congestion is due to drivers looking for available free parking spots. To this end smart and sustainable mobility is, now, one of the central concepts in the vision of the Smart City, where the IoT plays a very important role [40], [41]. A smart parking management system can make the process of finding a parking space quicker and less frustrating and use existing parking spaces more efficient [42]. Smart parking solutions vary with regards to sensing technologies and methods that are used for parking space occupancy prediction and classification.

The majority of articles focused on predicting parking occupancy or availability based on the history of occupancy for a given parking lot which included date-time information and occupancy status. Some researchers do not utilize a particular sensing device, instead focusing on publicly available datasets [43], [44], [45], [46] concentrating the goal of their study in finding the most appropriate ML technique for prediction or classification of a free parking space. They provide a machine learning model-based framework that may be used in future systems, rather than discussing or proposing an entire technical architecture for their approach.

To this end, there are now ITS systems that collect and process traffic data (vehicle presence, vehicle speed, vehicle density and occupancy ratios, etc.) from vehicle detector sensors. These can be intrusive (in-roadway) or nonintrusive (over roadway) [47], [48]. Intrusive detectors are embedded in the pavement, taped to the surface of the roadway, or mounted on the roadway surface. This group includes inductive loops, pneumatic road tubes, piezoelectric cables, and capacitive sensors [48]. Non-intrusive detectors are placed above the surface of the roadway or on poles adjacent to it, so that they do not disrupt traffic flow during installation and maintenance. Some examples are video cameras, acoustic signal processors, radar, and ultrasonic and infrared sensors [48]. All these solutions, however, are power-hungry and expensive to deploy and maintain, hence inadequate for large-scale deployment WSNs, which are preferred over wired sensor networks to cover large areas. Magnetic sensors based on magnetoresistors have recently been proposed for vehicle detection [49], [50] because they are quite sensitive, compact and resistant to external environmental conditions including rain, wind, snow or fog than sensing systems based on video cameras, ultrasound or infrared radiation. WSNs based on magnetoresistors can detect and track moving vehicles to obtain vehicle count or speed statistics [51], [52] but no experimental analyses are known about the use of magnetoresistors to detect static vehicles in spite of its interest, for example, to know where empty parking spots are and for street parking control. Further, published solutions rely on voltage-amplitude based interface circuits, which are more complex, expensive and power-hungry than a direct connection of magnetoresistive sensors (AMR and GMR) to a microcontroller [53]. Recent advances of wireless sensor networks, along with low-cost sensor devices, Web technology and pervasive computing gained momentum to the expansion of the IoT ecosystem [54], [55]. In recent years, a number of solutions have been proposed aimed at resolving the problem of finding free parking spaces aimed at establishing a green environment and increasing the quality of life in Smart cities [56], [57]. The implementation of solutions for the detection of free parking lots in the context of IoT is based on the deployment of sensors that can sense the environment (for example, the presence of vehicles) and transmit that data to a central server for further processing (e.g. via radio channel). Selection of the appropriate sensor device for vehicle detection rather depends on the requirements of a parking lot, focusing on solutions that preserve high accuracy while reducing the overall cost [58].

2.3.3.2 Description

The ASSC in Scenario SC03: Continuous occupancy monitoring in a parking lot is installed on the wall of each parking spot of an indoor or outdoor parking lot. Each ASSC serves as a wireless node and incorporates various data fusion and edge intelligence methods in order to detect whether each parking spot is occupied or not. The AMANDA system receives raw data from the magnetometer which is used as the input to the data fusion optimization engine which incorporates a Kalman filter in order to contain the noise and various inaccuracies that come from such a sensoring system. The output is then used as input for the edge intelligence module which adapts a binary DT to detect the presence of a vehicle in each parking spot.

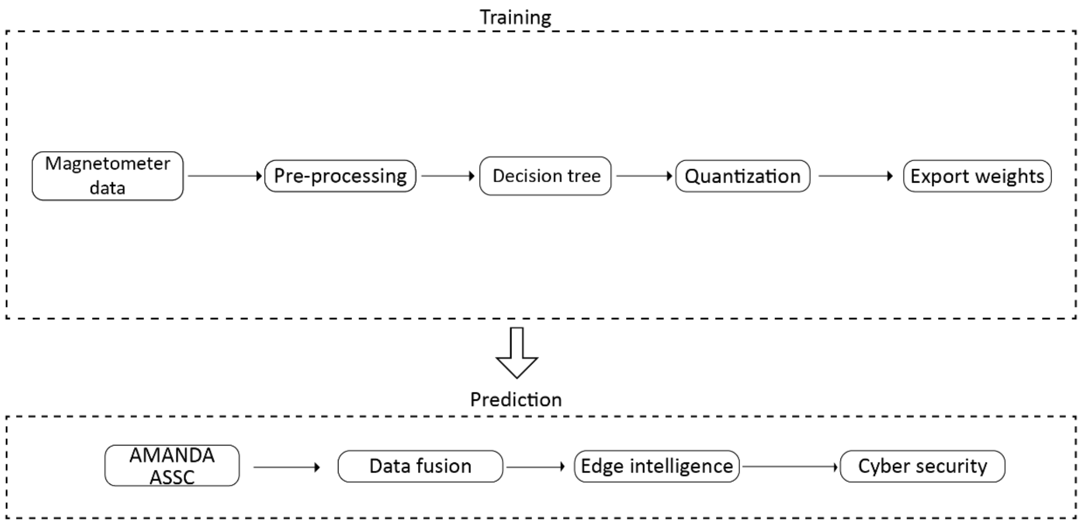


Figure 13 Scenario SC03: Continuous occupancy monitoring in a parking lot

2.3.3.3 Main functionality

Figure 13 presents the system workflow with the first 4 steps (Pre-processing, Decision tree training, quantization, export weights) taking place in an external node while the remaining on the AMANDA ASSC.

- **Pre-processing.** Data collected by the magnetometer sensor may contain missing values or outliers that must remove. The data fusion optimization engine uses a standardization method to adjust the distribution of each attribute in order to have a mean of zero and a standard deviation of one.
- **DT classifier training.** There are several parameters that have to configure before the training of the DT model. Some of them are the criterion, which defines the function to measure the quality of a split. There are two functions available, the Gini criterion for Gini Index and the Entropy for Information Gain. Also, there are two different splitter parameters, which defines the strategy to choose the split at each node. The parameters include the “best” splitter which chooses the best split and the “random” splitter which select randomly the best split. For the AMANDA ASSC, the Entropy splitter was selected combining with the “best” splitter as they had the best result as well the smallest memory footprint. The following equation describes the Entropy, where p_i is the frequentist probability of a class i in the dataset:

$$E(s) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

- **Quantization.** For this Scenario, 16 bits for quantization are used as the quantized model had a small memory footprint and a better classification performance
- **Export weights.** The output from the above training phase contains the weights of the tree model that can be used during the next step for prediction. The different predicted classes and Scenarios are presented in Table 7

Occupied	**Free**
Vehicle right next the ASSC – Clear “Occupied Parking Space” Outcome	No magnetic threshold reached – Clear “Free Parking Space” Outcome
Vehicle above the ASSC – Clear “Occupied Parking Space” Outcome	
Vehicle far from the ASSC - Ambiguous “Occupied Parking Space” Outcome	

Table 7 SC03: Different Scenarios for occupancy and free parking spots

2.3.3.4 Dependencies to other components

Figure 13 shows the relationships between the data fusion, edge intelligence, and cybersecurity modules in the prediction phase. Raw data from magnetic sensor were used from the data fusion module. Then the output data were used from edge intelligence and cybersecurity software module.

- **Data fusion optimization engine**. For SC03 the Kalman filter was used in order to transforms the raw data from magnetic sensor to a more usable format for the edge intelligence module
- **Cybersecurity software module.** The cybersecurity module encrypts the output of the edge intelligence module using a PLS-based encryption method

2.3.3.5 Experimental Results

The binary tree used achieved an accuracy of to 93.3%. In order to evaluate the algorithm a dataset was created with magnetic readings from different vehicles such as cars, bikes and bicycles and different distances from the ASSC with around 30000 data points for the “Occupied Parking Space” and 40000 data points for the “Free Parking Space” classification outcome (Occupied Parking Space or Free Parking Space). Table 8 illustrates the number of true positive, true negative, false positive and false negative for the parking lot occupancy use case.

	Predicted: Occupied Parking Space	**Predicted: Free Parking Space**
Actual: Occupied Parking Space	26151	2279
Actual: Free Parking Space	2187	36038

Table 8 SC03: Confusion matrix

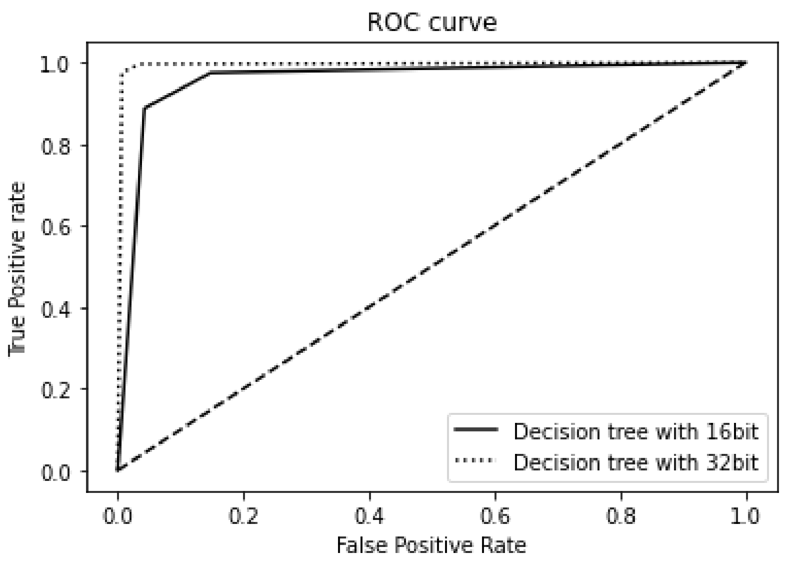


Figure 14 AUC-ROC curve for SC03

Figure 14 presents the AUC - ROC curve for SC03. The solid line describes the quantized model with 16 bits, while the dashed line the initial model which is represented with 32 bits, with values of 0.98 and 0.93, respectively. As can be seen, a small decrease for the model with 16 bits is observed.
The execution of the binary tree has a computational complexity of 750 multiply-accumulate operations per input and a decision execution time of about 45 milliseconds. The memory requirements of the binary tree model were approximately 19.57 kB, with 4 bytes added for input data from the magnetometer as they were float values.

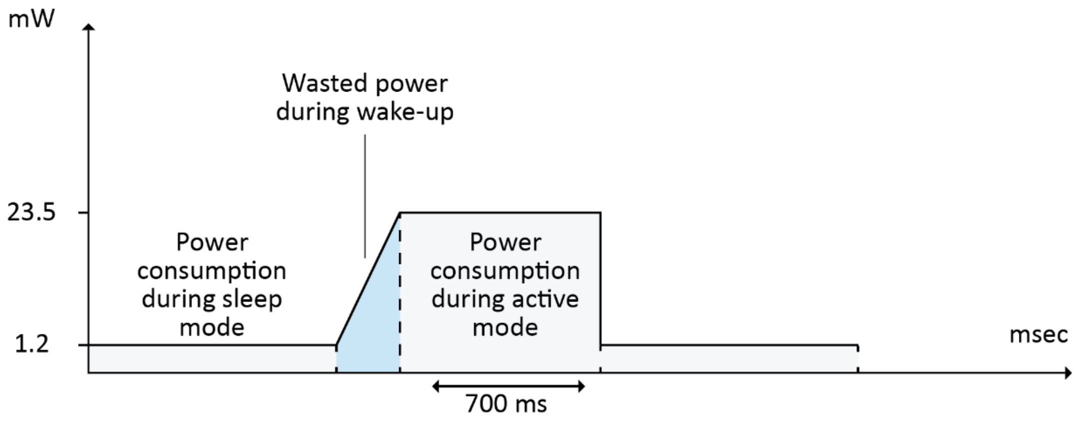

mW
Wasted power during wake-up
23.5
1.2
Power consumption during sleep mode
Power consumption during active mode
msec
700 ms

Figure 15 SC03: Power consumption

The power consumption of the AMANDA card in active and sleep mode for SC03 is presented in Figure 15. When the card is in sleep mode, it consumes approximately 1.2mW of power since all of the sensors and the MCU are in sleep mode. In active mode, an increase in consumption is observed with a max value of 23.5mW. The execution time was about 700 msec, with 670 msec for the measurements and 30 msec for the execution of the edge intelligence module.

2.3.4 Scenario SC04: Asset and people localization with access control

2.3.4.1 State-of-the-Art

A growing number of studies on fall detection systems have been published in recent years, however research on the implementation of such systems on embedded systems is restricted. In [59] presented a smart stick that combined AI with ultrasonic sensors to support visually impaired persons in navigating. The proposed smart stick provided image recognition, collision detection, and obstacle detection. The whole system was implemented on a raspberry pi and the AI algorithm was based on a Computer Vision API of Microsoft Cognitive Services [60]. A system based on a raspberry pi also presented by Pruthvi et al. [61]. For object detection and classification, a deep learning model called YOLO was implemented with an execution time close to 1.22 seconds on average depending on the number of objects in the image [62]. Also, it provided object information by voice for visually impaired people.

In [63] Zhang et al. designed a fall detection system used a cell phone with a tri-axial accelerometer embedded in it. The 1-class SVM is used for data pre-processing, and the wireless channel is used for Internet connection. The KNN technique and kernel fisher discriminant (KFD) analysis are used to classify the data. The system was based on an MCU called PIC18F2455 constructed by the Microchip Technology Inc. with main features of the chip includes 24KByte flash program memory and 2048Byte SRAM. The average accuracy of the implementation was 93 percent, with no mention of power consumption measurements. A fall detection system presented in [64] based on SVM classification algorithm. The portable unit with an embedded microprocessor acquires the tri-axial accelerometer output, and tracking information about the user's motion is transmitted to a local receiver unit. The accuracy of the proposed method was 95.6% and the platform that the system was based was a low power embedded system the MSP430F149.

In [65] an MLP model with three input neurons presented, with the acceleration components collected by a triaxial accelerometer. The MLP included two hidden layers, back-propagation mechanism while the output from this detector can be classified into two possible patterns: fall and not fall. The system was based on a MSP430 and the accuracy on average was 94%. A more complex implementation is described in [66]. The wearable system was based on an ATmega328 and it consisted of an accelerometer, a gyroscope, a thermometer that communicate with an external central node via ZigBee. An MLP model was used as a fall detection algorithm. Furthermore, the system had not evaluated in an experimental testbed. As a result, there are no performance measurements supplied.

Papers	**Methods**	**Implementation**
[59], [61]	Computer Vision API / YOLO	Raspberry Pi
[63]	KNN & KFD	PIC18F2455
[64], [65]	SVM / MLP	MSP430F149 / MSP430
[66]	MLP	ATmega328

Table 9 SC04: Summary of the State-of-the-Art

2.3.4.2 Description

The ASSC is carried by the employees of a company. After of some initial configuration of the ASSC such as defining the user's height and weight and by utilizing the accelerometer along with machine-learning algorithms, the ASSC can detect if the person has fallen and he is unconscious or injured. The ASSC transmits the proper information to a local terminal which projects the status of all employees. Assumption is an average of 1 fall detection every 1 minutes. The active parts of the embedded system are the accelerometer as well the processor running the AI algorithm for fall detection. The data used for Scenario 4 include the value of acceleration on axis x, y, z in meters per second squared as well the g-force in each axis which is calculated by dividing the acceleration of each axis with 9.81, the value which is the force of gravity. The ML algorithm chosen for this Scenario is KNN as it requires a small amount of memory as well has a short execution time. The output of the KNN model can be classified into two possible patterns, fall and not fall and in more detail in each class are contained some activities presented in the Table 10.

Not Fall	**Fall**
Walking slowly/quickly	Fall forward/backward while walking caused by a slip
Jogging slowly/quickly	Lateral fall while walking caused by a slip
Walking upstairs and downstairs slowly/quickly	Fall forward while walking/jogging caused by a trip
Slowly/Quickly sit in a half-height chair and up slowly/quickly	Vertical fall while walking caused by fainting
Slowly/Quickly sit in a low height chair, wait a moment, and up slowly/quickly	Fall while walking, with use of hands in a table to dampen fall, caused by fainting
Sitting for a few moments, attempting to stand, and collapsing into a chair	Fall forward when trying to get up/sit down
Sit for a moment, then slowly/quickly lie down, wait a moment, and then sit again	Lateral fall when trying to get up/sit down
Standing, slowly bending at knees, and getting up	Fall backward when trying to sit down
Standing, slowly bending without bending knees, and getting up	Fall forward/backward while sitting, caused by fainting or falling asleep
Standing, get into a car, remain seated and get out of the car	Lateral fall while sitting, caused by fainting or falling asleep
Stumble while walking	
Gently jump without falling (trying to reach a high object)	

Table 10 SC04: Different scenarios for fall and not fall activities

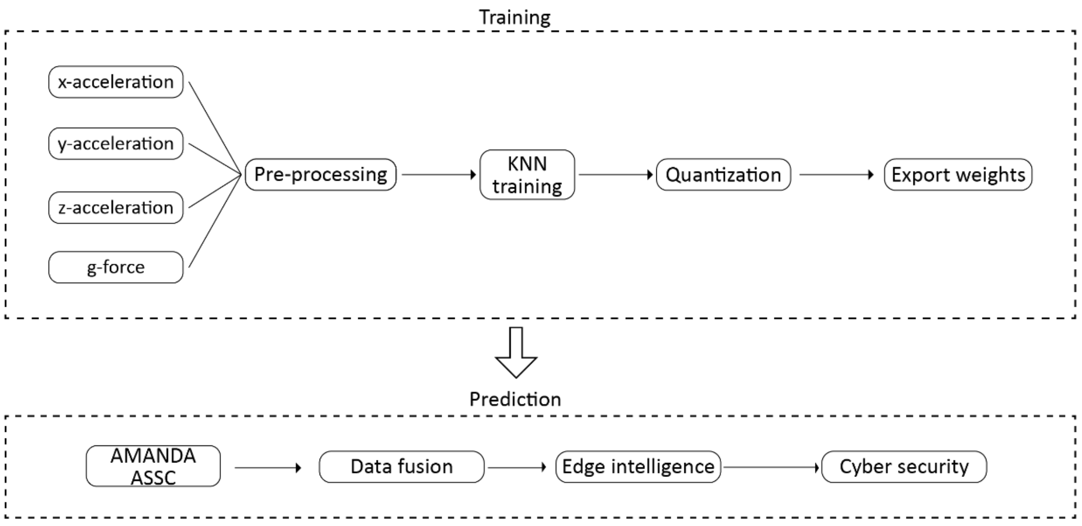


Figure 16 Scenario SC04: Fall detection

2.3.4.3 Main functionality

Figure 16 Scenario SC04: Fall detection presents the system workflow with the first 4 steps (Pre-processing, KNN training, quantization, export weights) taking place in an external node while the remaining on the AMANDA ASSC.

- **Pre-processing**. Data collected by the accelerometer sensor may contain missing values or outliers that must remove. A min-max normalization method is applied for every feature, transforming the minimum value of the feature to 0, the maximum value to 1 and all the other values of the features into a decimal between 0 and 1. Table 11 shows the descriptive statistics for the numeric features. Also, the input data is divided into two datasets, one for training and the other for evaluation with a ratio of 0.2
- **KNN training**. There are several parameters that have to configure before the training of the KNN model. Some of them are the number of k neighbours, the method for the distance calculation between the training data and the leaf size. Different numbers of k neighbours were tested, but the model with 2 k neighbours had the best result as well had the smallest memory footprint and the lowest power consumption. Also, there are 3 distance metrics that are often used are Euclidean Distance, Manhattan Distance, and Minkowski Distance. For the Scenario 4 the metrics that had the best accuracy was the Euclidean Distance. The following equation describes the metric:

$$d_{euclidean} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- Furthermore, the size of the leaf can affect the speed with which the tree is built and queried, as well as the amount of memory required to store it. The optimal value of the leaf size for the minimum memory requirements and the best accuracy of the model was 30
- **Quantization.** Converting the original 32-bit floating-point values to fixed-point integers, quantization can reduce the overall size of the AI model. For this Scenario, 16 bits for quantization are used as the quantized model had a small memory footprint and a better classification performance

- **Export weights.** The output from the above training phase contains the weights of the tree model that can be used during the next step for prediction. The size of the exported weights is described in detail in the next Section

The prediction of the KNN model is a procedure that takes over on the MCU, using the weights which have been trained in the previous phase. The Edge intelligence module in this Scenario includes the data acquisition as well feature scaling of the input dataset. The input data from the accelerometer must be on the same scale as the training data. Thus, a min-max normalization was applied. Subsequently, the quantized weights of the training phase are loaded from the MCU. The output or prediction of the edge intelligence node contains 1 for fall and 0 for not fall.

	x-acceleration	y-acceleration	z-acceleration	g-force
Type	Integer	Integer	Integer	Float
Count	20000	20000	20000	20000
Mean	19.35	-197.42	-43.62	-1.45
Std. Deviation	97.13	191.27	153.97	3.05
Min	-1362	-1903	-1667	-15.44
25%	-9	-263	-105	-1.73
50%	20	-233	-46	-0.05
75%	50	-31	18	-0.01
Max	1930	1553	1295	1.06

Table 11 SC04: Descriptive statistics of the dataset

2.3.4.4 Dependencies to other components

The prediction phase in Figure 16 describes the dependencies between the data fusion, edge intelligence and cybersecurity modules. Raw data from the acceleration sensor are used for the data fusion module, while the fused data are used for edge intelligence or cybersecurity module. Also, the output data from the edge intelligence module is passed to the cybersecurity software before they are shared wirelessly.

- **Data fusion optimization engine**. For SC04, data fusion includes a state estimation in the form of a Kalman filter to extract new features as well as a pre-processing which it combines different sensor sources to ensure that the system has enough details to make a decision
- **Cybersecurity software module.** The output of the edge intelligence module includes a binary variable with values 1 for fall and 0 for not fall which is encrypted by the cybersecurity module using a PLS-based encryption scheme before they are shared wirelessly

2.3.4.5 Experimental Results

The KNN algorithm had an accuracy close to 96.8% with an f1 score of 96.2%. For the evaluation of the algorithm, a test dataset was used, with almost 24000 data for each class. Table 12 illustrates the number of true positive, true negative, false positive and false negative for Scenario 4. As can be observed the algorithm had a good accuracy both in class 0 and class 1.

	Predicted: not fall	Predicted: fall
Actual: not fall	23820	829

Actual: fall	871	24407

Table 12 SC04: Confusion matrix

Figure 17 presents the AUC - ROC curve for SC04. The solid line describes the quantized model with 16 bits, while the dashed line the initial model which is represented with 32 bits. With values of 0.99 and 0.96, respectively, the performance of both variations was very close, with a small decrease for the model with 16 bits.

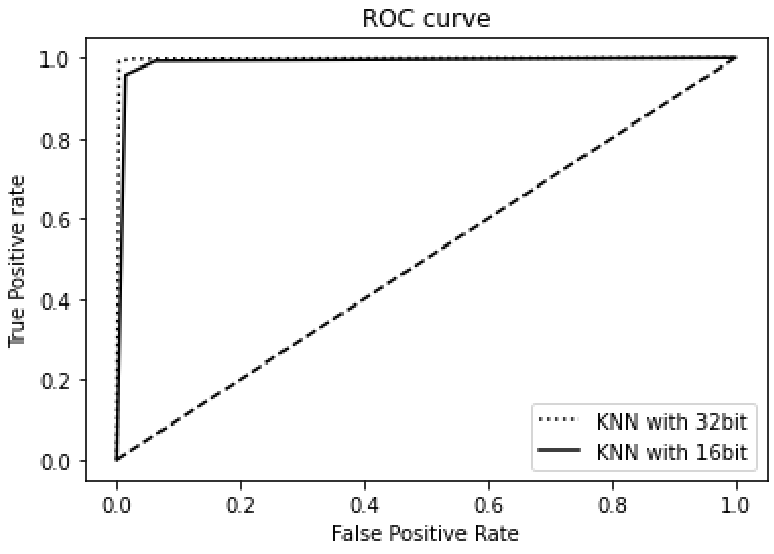


Figure 17 AUC-ROC curve for SC04

The execution of the KNN model has a computational complexity of 650 multiply-accumulate operations per input and a decision execution time of about 40ms. The memory requirements of the KNN model for SC04 were approximate 17.87kB adding 4 bytes for each input variable.

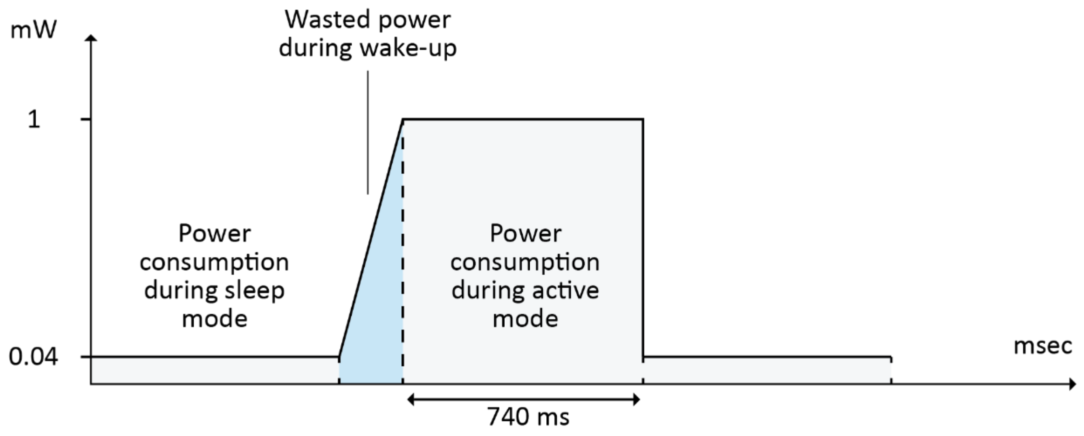

mW
Wasted power during wake-up
1
Power consumption during sleep mode
Power consumption during active mode
0.04
msec
740 ms

Figure 18 SC04: Power consumption

The power consumption of the AMANDA card in active and sleep mode for SC04 is shown in Figure 18. When the card is in sleep mode, it consumes about 0.04mW of power since all of the sensors and the MCU are in sleep mode. In active mode, an increase in consumption is observed with a max value of 1mW. The execution time was about 740 msec, with 700 msec for the measurements and 40 msec for the execution of the edge intelligence module.

2.3.5 Scenario SC05: Monitor transportation conditions of medicines/vaccines

2.3.5.1 State-of-the-Art

The massive increase in parcel transports has necessitated the development of effective and useful intelligent logistics systems. In the field of industrial big data analytics, several prior researches have addressed predictive maintenance problems. In [67] the author suggested a system that it monitoring industrial equipment and predicting when equipment maintenance is necessary, using AI. They provide one of the first examples of performance assessment and prediction tools, which performs preventative maintenance in order to avoid machine failure. The authors of [68] presented a new neural network-based prognostics model to support industrial maintenance decisions. The failure probabilities based on real-world equipment are first calculated using the logistic regression method. The failure probabilities are then used as input into a prognostics model to predict the future value of the failure condition, which is then used to estimate the equipment's remaining useful lifetime. For the evaluation of the system 120 data points were used. Also, the evaluation metrics were the RMSE with a value of 0.18 and the r-squared score with a value of 0.58.

In [69], [70] AI methods and specifically ML models such as ANN, DT, Logistic regression and SVM were used to predict machine failures. The deep learning method with the ANN model had the best accuracy with a value of 96.5% while no information is provided about the power consumption of the system and the memory requirements. A method based on ML algorithms is presented in [71] to track and monitor the shipped packages. The models that were used included the Gradient boosting, SVM, and Logistic Regression classifiers with accuracy 74.2%, 72.8%, and 61.1%, respectively and the evaluation dataset consisted of 22700 data points. Also, the proposed method was implemented on an MSP430 while no information on power consumption is provided.

2.3.5.2 Description

The ASSC is placed in containers that contain sensitive medical supplies such as medicines or vaccines and collects environmental data and the movement profile during delivery. Edge intelligence algorithms and specifically the SVM model in conjunction with the sigmoid function are then used to estimate the supplies' condition during shipping. The shipping evaluation data will be sent to the supply manager's smartphone via BLE. Due to the sigmoid function, the SVM model's output is a continuous number with a range of zero to one. The output value zero means that the parcel is not damaged during transport while value one means that the parcel may have been damaged. During monitoring, the system would mostly consume energy only from the battery. Every 10 minutes the card wakes up and measures temperature, humidity, VOC, light intensity and acceleration. The measurements and time stamps are saved in FRAM. When the shipment is at its destination, the data in the card can be read using NFC or BLE. Required here are low-power sensors and minimal processing. Variations: During transportation in small parcels, the card can be programmed to broadcast an alert to a gateway (say a smartphone or a device built in the transport vehicle), if some of the parameters monitored are outside the safety zone. If the transport box is a hindrance for the communication of the wireless signal, LoRa can be used, with the SF adapted to the transport environment. It can be assumed that there will be little light and that the card will rely on the stored energy. It can also be safely assumed that wireless communication (used only to inform of problems) will not be often activated. An average of 1 LoRa frame (SF11) every 2 hours is assumed. In case BLE is used, 20 ADV events are assumed every 2 hours.

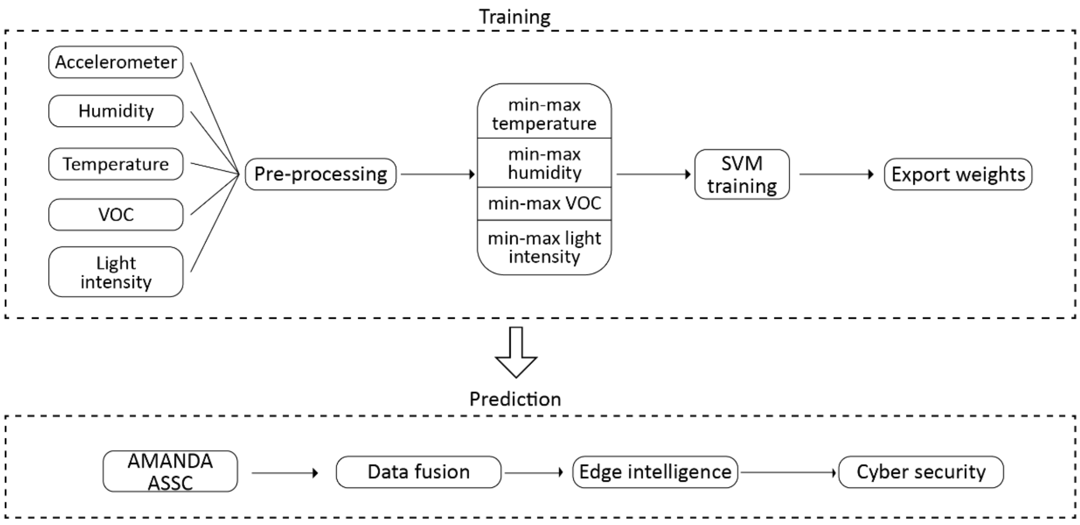


Figure 19 Scenario SC05: Monitor transportation conditions of medicines/vaccines

2.3.5.3 Main functionality

Figure 19 presents the system workflow with the first 3 steps (Pre-processing, SVM training, export weights) taking place in an external node while the remaining on the AMANDA ASSC.

- **Pre-processing**. Data collected by the accelerometer, temperature, humidity as well light sensor may contain missing values or outliers that must remove. Also, some new features such as the min and max value of temperature, humidity, VOC and light intensity set by the supplier are added. A min-max normalization method is applied for every feature, transforming the minimum value of the feature to 0, the maximum value to 1 and all the other values of the features into decimals between 0 and 1
- **SVM training**. There are three parameters for the SVM model that must be specified for SC05. The kernel function, the C parameter as well the gamma value. The Gaussian kernel was used as a kernel function as it had the best accuracy and the less execution time. Also, the number 1 was chosen for the C parameter and gamma value since the model converged faster. Subsequently, the training of the SVM model includes the use of the sigmoid function as the output value should be between [0,1]. The following equation describes the function:

$$y = \frac{1}{(1 + e^{-f(x)})},$$

  Where *f(x)* is the output of the SVM model with values [-inf, inf]
- **Export weights.** The SVM model with the sigmoid function is a lightweight algorithm with a size small enough that it can run on a limited-memory platform such as the AMANDA ASSC thus, no quantization method needs to be applied to its weights

The prediction of transportation conditions takes over on the AMANDA ASSC, using the weights of the SVM which have been trained in the previous phase. The input data from the sensors must be on the same scale as the training data thus a min-max normalization method was applied. The output contains continuous values between [0,1] as before the output of the SVM model a sigmoid function was applied. In order to train and evaluate the SVM model a set of data has been collected using the sensors that utilize the AMANDA ASSC. Approximately 6000 data points have been collected in total from the temperature, humidity and light sensor as well from the accelerometer. Also, the data was divided into two datasets, one for training and the other for evaluation with a ratio of 0.2.

2.3.5.4 Dependencies to other components

In Figure 19, the dependencies between the data fusion, edge intelligence and cybersecurity modules are presented. The data fusion module uses raw data from the acceleration, humidity, temperature, VOC and light sensor while the fused data is utilized for edge intelligence or cybersecurity. Additionally, the edge intelligence module's output data is processed via cybersecurity software, before being transferred wirelessly.

- **Data fusion optimization engine**. For SC05 an AHRS filter with a Semi-Supervised Learning technique of Fuzzy Logic was used in order to correct the problematic data and to improve the data reliability to ensure the required output
- **Cybersecurity software module.** The cybersecurity module encrypts the output of the edge intelligence module using a PLS-based encryption technique

2.3.5.5 Experimental Results

An evaluation dataset was used to calculate the performance of the SVM model and metrics such as MSE, RMSE and R-squared was selected as the prediction were continuous values. The MSE is a metric that indicates how near a fitted line is to the data points. Also, the prediction is better when the MSE is low. The procedure is described by the equation:

$$MSE = \sum_{i=n}^{n} \frac{(\hat{y}_i - y_i)^2}{n}$$

Where *n* is the number of data points, y_i the observed value and $\hat{y_i}$ the predicted values. The RMSE is the square root of the MSE and it is the standard deviation of the residuals (prediction errors). The residuals are a measure of how far the data points are from the regression line. Thus, the RMSE is a measure of how spread out these residuals are. it is defined as follows:

$$RMSE = \sqrt{MSE}$$

MAE is a statistical metric that measures the average magnitude of errors in a set of predictions without taking into account their direction. It's the average of the absolute differences between forecast and actual observation over the test sample, where all individual deviations are given equal weight.

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

The R-squared is a statistical metric that indicates how well a regression model fits the data. The optimal r-square value is 1. The closer the r-square number is to one, the better the model fits. The R-square is a calculation that compares the residual sum of squares (SSres) to the total sum of squares (SStot). The sum of squares of perpendicular distances between data points and the average line is used to determine the total sum of squares. The following equation describes the R square score:

$$R_{square} = 1 - \frac{SSres}{SStot}$$

The MSE, RMSE, MAE and R-square score are presented in Table 13. As can be seen the SVM model has a low value in RMSE for the training dataset as well for the test dataset. It is important to mention that the model didn't overfit on training dataset as the RMSE for the test set is also low. With values of 0.95 and 0.92, R-squared confirms that the model works well for both training and testing data.

	Training	**Testing**
MSE	0.0169	0.0361
RMSE	0.13	0.19
MAE	0.11	0.18
R-square	0.95	0.92

Table 13 SC05: MSE, RMSE, MAE and R-square value

The execution of the SVM model has a computational complexity of 700 multiply-accumulate operations per input and a decision execution time of about 37 milliseconds. The memory requirements of the SVM model for SC05 was about 18.37kB adding the size of each input data. Specifically, input data of temperature, humidity, min and max values of temperature and humidity as well data from accelerometer are float values and need 4 bytes for every instance. While input data of light intensity and VOC are uint16 and it needs 2 bytes per instance.

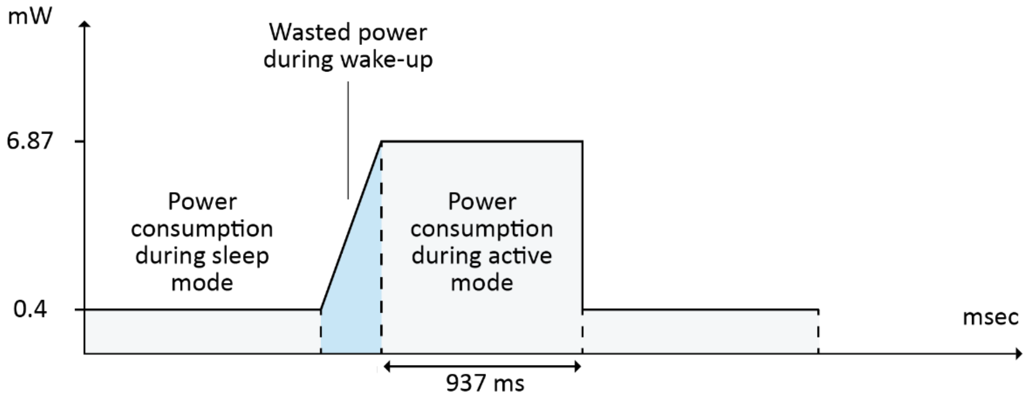


Figure 20 SC05: Power consumption

Figure 20 shows the power consumption of the AMANDA card in active and sleep mode for SC05. When the card remains in sleep mode the power consumption is about 0.4mW as all the sensors and the MCU has negligible consumption. In active mode, consumption increases to a maximum of 6.87mW. The execution time was about 937 msec, with 900 msec for the measurements as well as 37 msec for the execution of the edge intelligence module.

2.3.6 Scenario SC06: Crowd counting for social distancing

2.3.6.1 State-of-the-Art

In the last few years, a growing number of articles have been published for crowd counting systems but research on the implementation of such systems on embedded systems is limited. Methods for counting crowds can be divided into two categories: those that handle counting as an object detection problem and those that use the crowd density estimation based on features and regression analysis. Detection-based methods work by training object detectors to locate each object in an image. As a result, some attributes from an image are extracted and used to train a binary or multiclass classifier. On the other hand, the regression-based models estimate the crowd count using the image features that have extracted and using ML techniques performs a regression between the image features and crowd size.

A lightweight method described in [72] for counting people in indoor spaces based on motion and size criteria using a histogram. The system was tested on the iMote2, a low-power platform with high-performance specs, including an Intel XScale processor, 32MB of flash memory, and 32MB of SDRAM. The research dataset included images of 1, 2, and 3 people, and the above method's accuracy was 89.5%, 82.48%, and 79.8%, respectively. Conti et al. presented two low-power CNN-based methods for estimating classroom occupancy [73]. The first CNN, which has eight layers, is used for head classification and counting. The second approach used the CNN as a regression model which was focused on the calculation of people density. The system was implemented on a Samsung Exynos 5410. A dataset with images from classrooms was used for the evaluation of the system and the root mean square error was 6.42 people with an energy cost of 3.97J/image. Despite the low root mean square error, this work is not appropriate for an embedded device with limited computational power.
Gomez et al. also designed and developed a low power people counting algorithm using a thermal image sensor and a CNN [74]. The CNN had three convolution layers, one pooling layer and for the output was used the softmax function. The system was deployed and validated on a Cortex M4 platform, with an error rate of +- 1 person in 84.4% of the images in the test set, but the execution period to measure the number of people took around 2.3 minutes and consumed 0.48mAh. Another disadvantage of the above approach was the thermal sensor's high cost as compared to an image sensor. In [75] the histogram of gradients and SVM is used for people counting. The proposed system was implemented on an FPGA platform and low-resolution grayscale images with a dimension of 320 x 280 pixels were used for the Scenario. The authors compared the performance of the algorithm using an FPGA platform and a PC with an i5 processor and conclude that their method was almost five times faster than the typical software core system.

Papers	Method	Implementation
[72]	Based on histogram	iMote2
[73]	CNN	Samsung Exynos 5410
[74]	CNN	Cortex M4
[75]	Based on histogram	FPGA

Table 14 SC06: Summary of the State-of-the-Art

2.3.6.2 Description

The ASSC will be distributed to the end-users in order to monitor any potential contact with an infected person. In that case, the end-user will be informed accordingly to perform a test and self-quarantine will be suggested to avoid further spreading of the disease. The BLE module will be used as the main tracking mechanism of the system. The users' anonymity is ensured by using the AMANDA card as a physical token for contact tracing in epidemics. Information will be sent to a gateway, preferably via LoRa SF7, to inform the main controller about areas with possible problems. Regular messages are broadcasted with BLE in ADV mode (train of 10 ADV events per minute) for users that have a smartphone and an appropriate app. These warnings may be used to alert users to potentially dangerously crowded areas. Assumption is an average of 1 frame every 10 minutes. The active parts of the embedded system are the imaging sensor, the processor running the AI algorithm for crowd counting as well the LoRa communication. Communication with LoRa takes place only when critical situations are detected. The detection of people is achieved by splitting the image into smaller images and classified them as people or background. An object detection method was not used as it required long execution time and high computing power systems. Using this method, a CNN with a simpler structure and lower complexity is required as the image detection problem was converted to a binary classification problem. Also, reducing the size of the input image, smaller convolution kernels will be needed, as well a smaller number of weights that represent the CNN model. In order to count the number of people in an indoor room, the AMANDA card is placed on the ceiling.

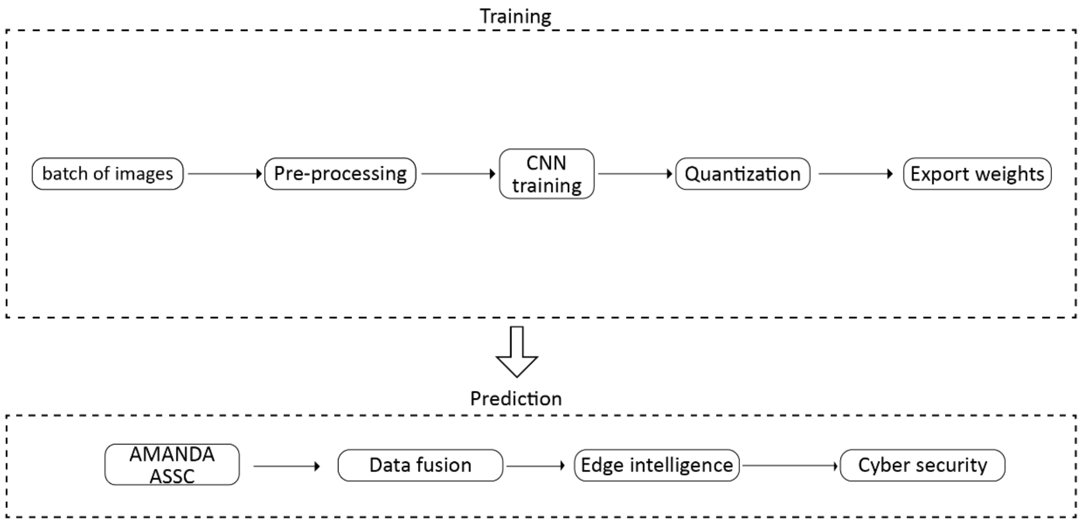


Figure 21 Scenario SC06: Crowd counting for social distancing

2.3.6.3 Main functionality

Figure 21 presents the system workflow with the first 4 steps (Pre-processing, CNN training, quantization, export weights) taking place in an external node while the remaining on the AMANDA ASSC.

- **Pre-processing.** Data from the image sensor includes RGB frames with a resolution of 320*240 pixels. The pixel values in images must be scaled before being used as input to the CNN model. The method that was used for pixel scaling was the pixel normalization, rescaling the pixel values from the range of 0-255 to the range of 0-1 preferred for the CNN model. This can be achieved by dividing all pixels values by the largest pixel value
- **CNN training.** CNN is used in most of computer vision tasks, for image classification and object detection. For image classification a CNN model, small enough that it can run on a limited-memory platform was used. It consists of two convolution layers (Conv) with two pooling layers and one fully-connected. Both of them contain a max-pooling operation with kernel size 2x2 and a padding parameter equal to 'same' as the input image gets fully covered by the filter without losing any of its features. Also, a rectified linear unit is applied at the fully-connected layer and a softmax function for the output layer giving a probability for the classification problem of the input image. The dropout operation is used before the fully-connected layer with a dropout rate of 0.4 to improve the accuracy of CNN and to avoid overfitting
- **Quantization.** Quantization can reduce the total size of the AI model converting the initial 32-bit floating-point values, to fixed-point integers. For this Scenario, 16 bits for quantization are used as the quantized model had a small memory footprint and a better classification performance
- **Export weights.** The output from the above training phase contains the weights of the model as multidimensional arrays that can be used during the next step for prediction.

The size of the exported weights for the CNN model is small enough that it can run on a limited-memory platform such as the AMANDA card

The final count of people is a procedure that takes over on the MCU, using the weights of the CNN model which have been trained in the previous phase. The Edge intelligence module in this Scenario includes the data acquisition as well feature scaling of the input data. The input images from the image sensor must be on the same scale as the training data thus, a pixel normalization was applied. Moreover, the quantized weights of the training phase are loaded from the MCU. The output of Edge intelligence module contains the total number of people that occupies the room which outcomes, comparing the confidence value from the softmax function for each selected image. In order to train and evaluate the CNN model a set of data has been collected using the image sensor. There were images that represent four different behaviours of the employees (wearing hats, covering heads, squatting, putting jackets), different lighting conditions (lights on, lights off) and images with high density with possibly overlapping heads. The training dataset contained almost 2430 images for class person and 1780 images for class background. Data augmentation techniques such as adjusting the brightness, contrast and magnification of the photos were also used to increase the amount of data and the diversity of the training set.

2.3.6.4 Dependencies to other components

As can be seen from Figure 21, data are exchanged between the data fusion, cybersecurity and edge intelligence module. The AMANDA system receives raw data from the image sensor, which are used as the input to the data fusion optimization engine. Then the output is used as input for the edge intelligence module which adapts a CNN model to predict the total count of people. Lastly, the data is passed to the cybersecurity software module to ensure that no sensitive data will be wirelessly transmitted to guarantee the security of the end user's information.

- **Data fusion optimization engine**. For SC06, data fusion is applied using a Kalman filter to extract new features as well as a pre-processing in order to the input images normalized based on the statistical information from the training set
- **Cybersecurity software module.** The output from the edge intelligence module includes the total count of people that occupies a room and it is encrypted by the cybersecurity module using a PLC-based encryption technique

2.3.6.5 Experimental results

The CNN model's classification accuracy is calculated using the validation dataset, which included images for all of the Scenarios described above. Figure 22 and Figure 23 illustrate the training and validation accuracy as well as the training and validation loss of the model for the validation dataset. The learning rate that was used for the model was 0.001 and the total number of epochs was 250. After 100 epochs, as can be seen from the figures, a remarkable precision has been achieved, with a value of around 88.52% and a loss value of around 0.11.

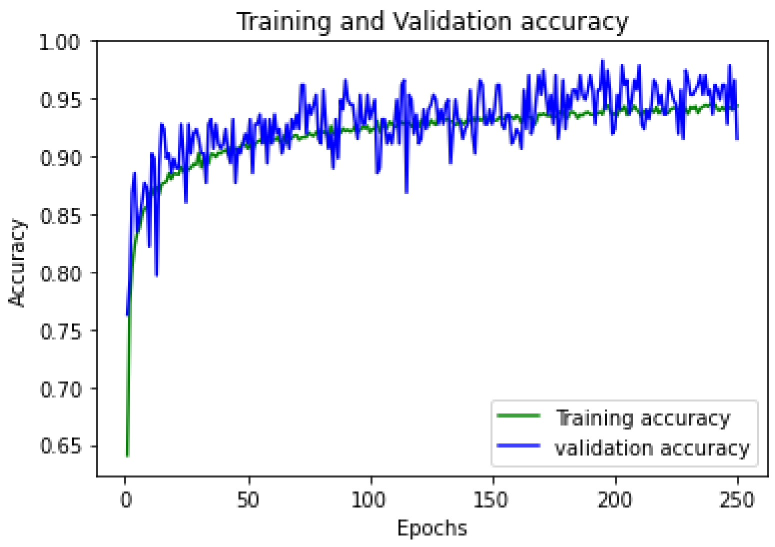


Figure 22 SC06: Accuracy at each epoch for train and validation dataset

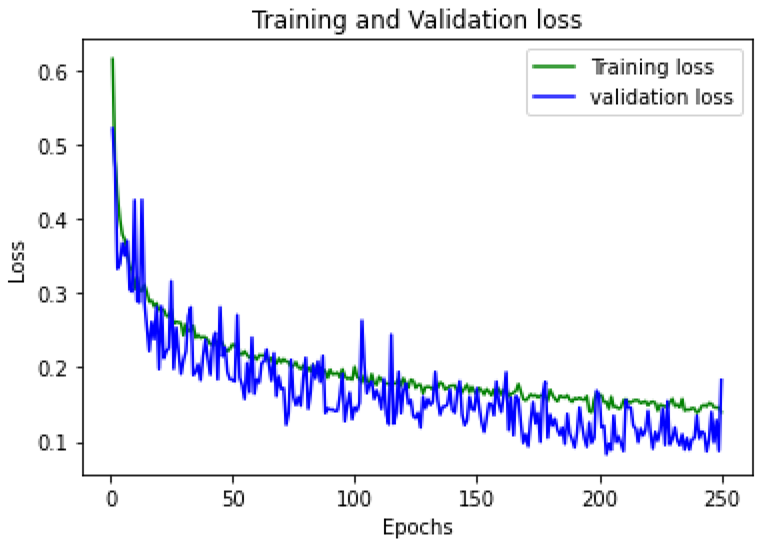


Figure 23 SC06: Loss at each epoch for train and validation dataset

To evaluate the accuracy of the total crowd counting method, five different datasets were used, for five different situations. People wearing a hat, empty room, low light conditions, people very close to each other, with the possibility of overlapping as well images that people are not so close each other. Figure 24 describes the final results. As can be seen, there are four error categories with zero, +-1, +-2, and +-3 errors for a total of ten people inside the room. The most difficult Scenario, as expected, was one in which people were extremely close to each other (high density), with only 40% of the evaluation dataset correctly predicted. However, by increasing the number of training images a better accuracy could be achieved. Furthermore, a good accuracy was observed in all remaining Scenarios.

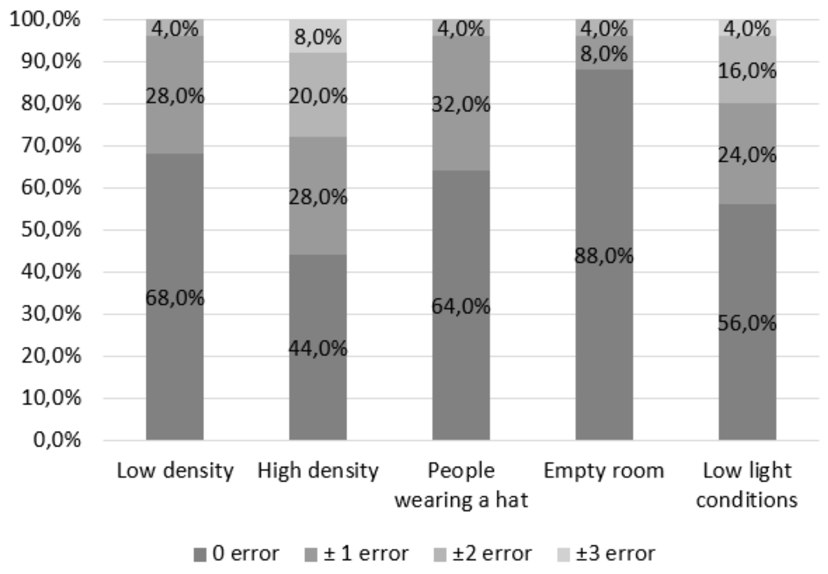


Figure 24 SC06: Percent stacked bar chart of errors for each Scenario

The execution of the CNN model has a computational complexity of almost 43k multiply-accumulate operations for each patch image and an execution time of approximately 3 sec to classify each small image. In total, about 4.8 minutes for almost 100 small images. Also, the memory requirements of the CNN model in the AMANDA card were about 98.51kB adding almost 9.5kB for each frame by the image sensor.

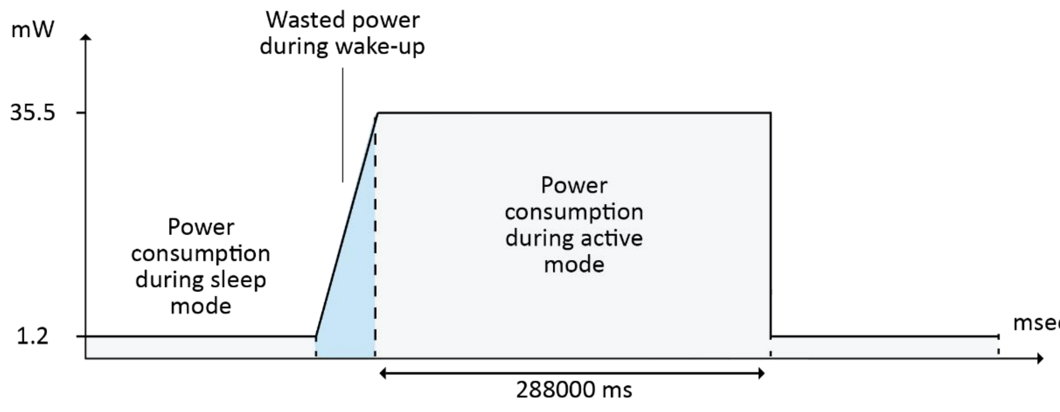

mW
Wasted power during wake-up
35.5
Power consumption during sleep mode
Power consumption during active mode
1.2
msec
288000 ms

Figure 25 SC06: Power consumption

Figure 25 illustrates the AMANDA card's power consumption in both active and sleep modes for SC06. The card's power consumption is around 1.2mW when it is in sleep mode, as the image sensor and the MCU has negligible consumption. In active mode, the consumption increases to a maximum of 35.5mW. The execution time was approximately 288000msec, with about 1000msec for the measurements and 287000msec for the execution of the edge intelligence module.

3 User interface

The AMANDA card does not have a traditional user interface (keyboard, display) to facilitate interaction with the end-users. This is due to constrains such as space and energy. However, the complexity of the system and the different use cases still require ways of interacting with the user in a simple, yet rich and meaningful way. Some of the existing sensors and components can be used for simple interactions. For more complex cases, a smartphone is connected to the AMANDA ASSC through the BLE wireless interface, providing that the card and the smartphone are in communication range. The NFC interface can also be used in some cases to allow a device such as a smartphone to serve as input/output device in very close proximity. In the case of a long distance between the elements, some of the data from the AMANDA Card could be accessed via a cellular link to the web site of the gateway, as can described in Figure 26.

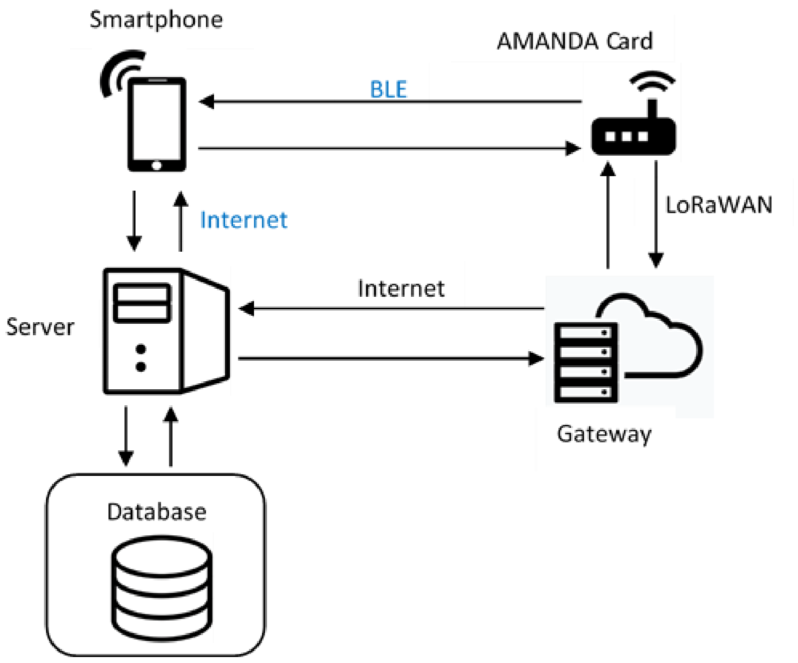


Figure 26 Interfacing options using the smartphone

3.1 Simple interfacing

The main existing elements that can be used to interact with the user are the capacitive sensor and the accelerometer.

3.1.1 The capacitive sensor as an input element

The capacitive sensor is used to detect the touch action of a user. It is the most frequently used sensor for user input. Thanks to its low power consumption, it is constantly active, even in the lowest power mode. One of the most important action is to use the capacitive sensor to move out of the low-power modes, into a state that allows for a more sophisticated interface.

The capacitive sensor can be used in combination modes, depending on how often the user touches the sensor during a given time window.

3.1.2 The accelerometer as an input element

The accelerometer can also be used for a very limited number of input combinations. The card can be shaken in a given way to represent an input parameter, for example by shaking the device once or twice within a 5 seconds window.

The accelerometer can also be read to determine the orientation of the card. That parameter would then be used as input for the application. The use of the orientation method in combination with energy harvesting is patented [76].

3.2 Interfacing over BLE or NFC

A smartphone running an appropriate app can be used to establish a connection with the AMANDA Card. In connected mode, data from the user can be entered using the smartphone. The data will subsequently be transmitted to the AMANDA card via the BLE connection. In the same way, information from the AMANDA card can be sent to the smartphone for visualisation by the user:

- The procedure can be started by a particular sequence on the touch input of the card
- The smartphone app should also be started to allow a BLE connection with the card
- The card responds by entering a new mode whereby a BLE connection is established with the smartphone
- Once the connection is established, frames are exchanged between the smartphone and the AMANDA Card. Those frames may be empty or may contain data (BLE connection mode)
- The frames carry the information from the user to the AMANDA Card and vice versa.
- The appearance and the menu / data to exchange can be designed to fit the application
- An end of interface sequence can be sent at the end to terminate the connection and return the card in its low-power mode

The exchange of parameters during the interface transactions can be based on the use of characteristics as shown on Figure 27 . This is a standard way of exchanging data in BLE. Appropriate text may be added in the app or on the card side. Much text on the card side will lead to extra energy requirements. Therefore, an effort should be made to limit transfer of data or text.

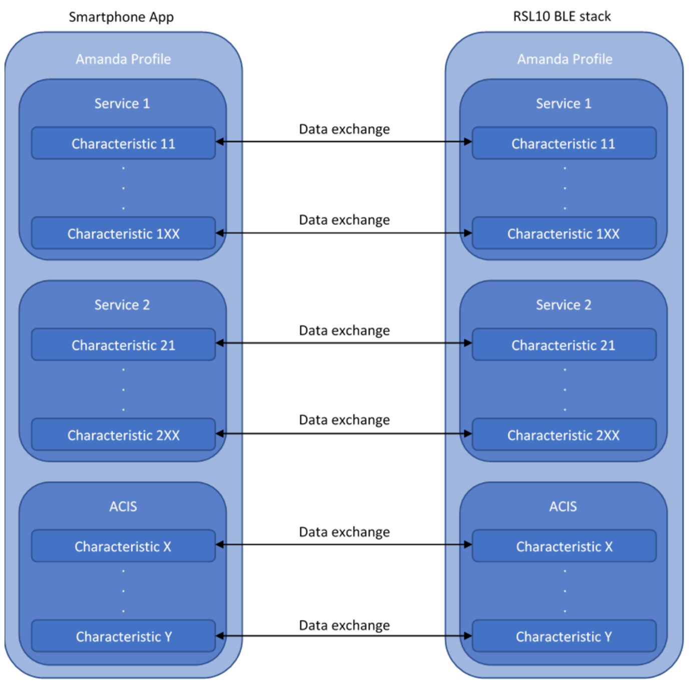


Figure 27 Use of characteristics to exchange data in BLE

In some cases, a simple app that receives and displays the advertised data is sufficient. The smartphone is used as a display element, showing to the user information from the card. That information can be tailored to feed particular use cases. In the case of NFC, the data exchange may be triggered simply by bringing the smartphone near the card. The interrupt signal from the activated NFC link would then start the appropriate user interface routine on the AMANDA card. Obviously, a combination of these methods (BLE interface and sensors) is also possible.

3.3 Mockups

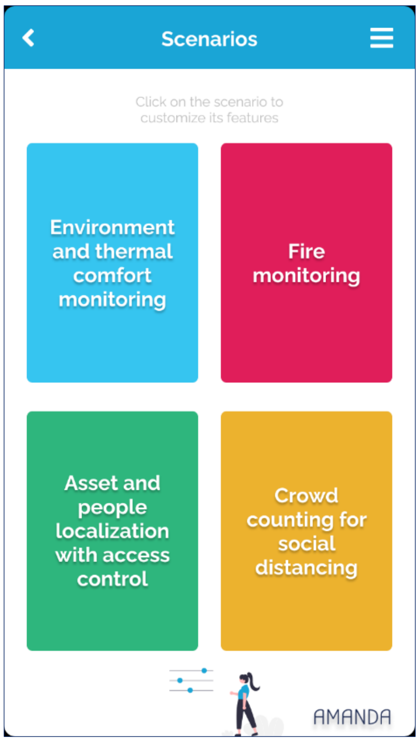
The previous Section described the interaction between the users and some of the existing sensors and components of the ASSC as well as with the mobile application. The AMANDA card also provides a configurable mobile application for developers that can be used to set up and develop upon the core intelligence engine. This Section presents mockups of the mobile application focusing on configurable options that the user can tune in order to maximize the abilities of the edge intelligence module of the card. Table 15 shows the available configurable options for each Scenario.

Scenario	**Features**	**Values**	**Explanation**
SC01	Clothing insulation	0.5 - 1	More than one selection is possible • Pants With Long Legs 0.1 • Sleeveless shirt 0.06 • T-shirt 0.08 • Long-sleeved blouse 0.15 • Short-sleeved shirt 0.09 • Flannel shirt 0.30 • Shorts 0.06 • Trousers (Thin Fabric) 0.15 • Trousers (Thick Fabric) 0.24 • Work coveralls 0.50 • Thin sweater 0.20 • Thick sweater 0.35 • Down jacket 0.55 • Boots 0.05 • Light skirt 15 cm above the knee 0.01 • Heavy knee-length skirt 0.25
	Metabolic rate	0.7 - 4.4met	• Sleeping 0.7 • Seated, reading, writing 1.0 • Typing 1.1 • Standing, relaxed 1.2 • Walking (3.2 km/h) 2.0 • Driving automobile 1.0-2.0 • Cleaning 2.0-3.4 • Handling 50kg bags 4.0 • Dancing, social 2.4-4.4
	Building type	Categories	• Large office (46320m^2, 12 floors) • Medium office (4980m^2, 3 floors) • Small office (510m^2, 1 floor) • Warehouse (5000m^2, 1 floor) • Supermarket (4180m^2, 1 floor) • Hospital (22422m^2, 5 floors)

	Cooling system	Categories	• Air-conditioned without operable windows • Naturally ventilated without mechanical cooling • Mixed-mode
SC02	Threshold value	1200 – 2600ppm	• <1200ppm (with good air exchange and without smoking people) • <1600ppm (with smoking people) • <2000ppm (poor air quality)
SC03	-	-	-
SC04	Initial position	Categories	• Office worker • Employee that works outside of the office • Working as a driver
SC05	Product specifications	Integer	• Minimum - maximum temperature that the package can be maintained • Minimum - maximum humidity that the package can be maintained • Minimum - maximum VOC that the package can be maintained • Minimum – maximum light intensity that the package can be maintained
SC06	Office type	Categories	• Small waiting room (30m^2, often large crowds, high density) • Medium waiting room (50m^2, medium density) • Large waiting room (120m^2, low density) • Small office (2 employees) • Medium office (10 employees) • Large office (30 employees)

Table 15 Different configurations of the edge intelligence core for each Scenario

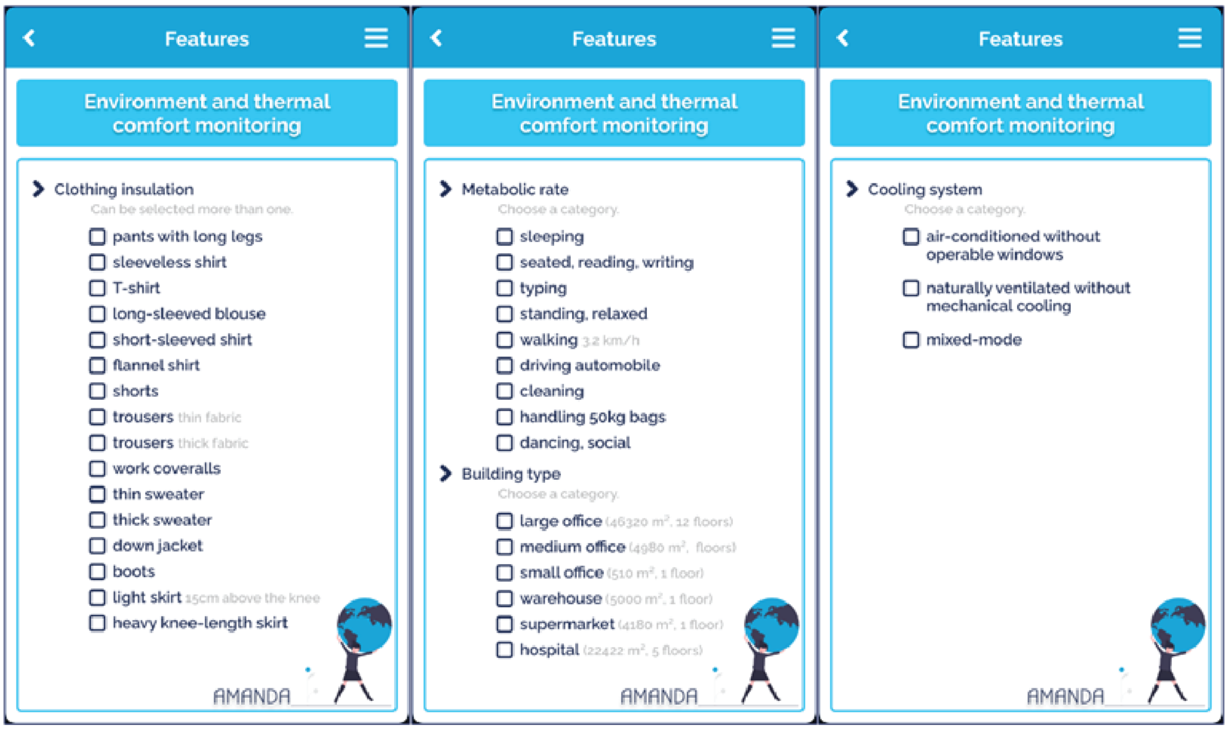
The proposed application contains a home page with 4 Scenarios for which the user can customize its features. It is possible to configure the characteristics of each Scenario only for the cards which have connected with the mobile application via Bluetooth. Figure 28 presents the 4 different Scenarios.


Scenarios
Click on the scenario to customize its features
Environment and thermal comfort monitoring
Fire monitoring
Asset and people localization with access control
Crowd counting for social distancing
AMANDA
Figure 28 Screen with the configurable scenarios

3.3.1 Scenario SC01: Environment and thermal comfort monitoring

For Scenario SC01, environment and thermal comfort monitoring there are three configurable variables that the user can define, as can be seen in Figure 29.

- **Clothing insulation.** It was introduced by [77] and describes the heat transfer characteristics of clothing to wear and takes into account the effect of air movement inside clothing caused by body movement or wind. Moreover, the clothing insulation can be used to evaluate clothing effectiveness
- **Metabolic rate.** The metabolic rate is a ratio of the working metabolic rate to the resting metabolic rate
- **Building type.** There are 6 different building types available that represent the most commercial buildings and are based on the ASHRAE Standard [78]
- **Cooling system.** It describes three different types of building cooling systems. Air-conditioned without operable windows, naturally ventilated without mechanical cooling and mixed mode


Features

Environment and thermal comfort monitoring

Clothing insulation

Can be selected more than one.

- ☐ pants with long legs
- ☐ sleeveless shirt
- ☐ T-shirt
- ☐ long-sleeved blouse
- ☐ short-sleeved shirt
- ☐ flannel shirt
- ☐ shorts
- ☐ trousers thin fabric
- ☐ trousers thick fabric
- ☐ work coveralls
- ☐ thin sweater
- ☐ thick sweater
- ☐ down jacket
- ☐ boots
- ☐ light skirt 15cm above the knee
- ☐ heavy knee-length skirt

Features

Environment and thermal comfort monitoring

Metabolic rate

Choose a category.

- ☐ sleeping
- ☐ seated, reading, writing
- ☐ typing
- ☐ standing, relaxed
- ☐ walking 3.2 km/h
- ☐ driving automobile
- ☐ cleaning
- ☐ handling 50kg bags
- ☐ dancing, social

Building type

Choose a category.

- ☐ large office (46320 m², 12 floors)
- ☐ medium office (4980 m², floors)
- ☐ small office (510 m², 1 floor)
- ☐ warehouse (5000 m², 1 floor)
- ☐ supermarket (4180 m², 1 floor)
- ☐ hospital (22422 m², 5 floors)

Features

Environment and thermal comfort monitoring

Cooling system

Choose a category.

- ☐ air-conditioned without operable windows
- ☐ naturally ventilated without mechanical cooling
- ☐ mixed-mode

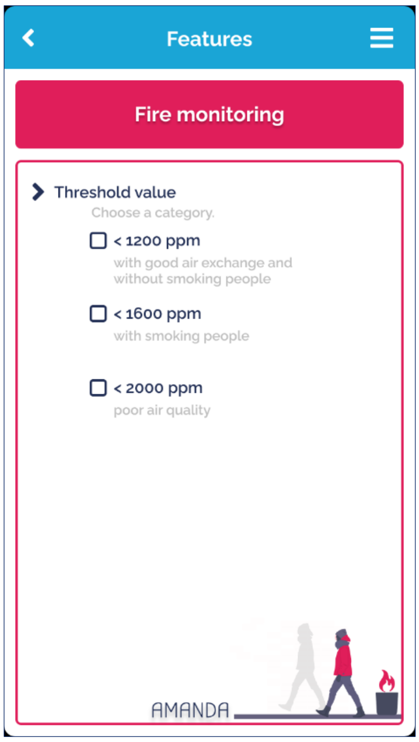
Figure 29 SC01: Configurable UI

3.3.2 Scenario SC02: Fire monitoring

For Scenario SC02 there is a configurable variable. The threshold value is used as a signal that enable the MCU from the sleep mode. The appropriate configuration of this variable is really important as many faults activations from the sleep mode by the MCU may occur using a non-suitable threshold value, increasing the total power consumption. The available threshold values are:

- < 1200ppm, for a room with good air exchange and without smoking people
- <1600ppm, for a room with smoking people
- <2000ppm, for a working area with poor air quality

Figure 30 presents the configurable UI for SC02.



Features

Fire monitoring

Threshold value

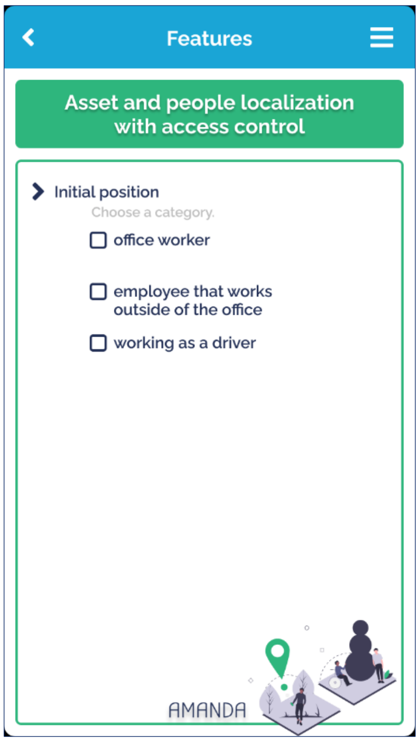
Choose a category.

- [ ] < 1200 ppm
  with good air exchange and without smoking people
- [ ] < 1600 ppm
  with smoking people
- [ ] < 2000 ppm
  poor air quality

Figure 30 SC02: Configurable UI

3.3.3 Scenario SC04: Asset and people localization with access control

For Scenario SC04, after some initial configuration of the ASSC such as defining the user's height and weight and by utilizing the accelerometer along with machine-learning algorithms, the ASSC can detect if the person has fallen. Also, the user can define the working area by choosing one of the below options, as illustrated in Figure 31.



Features

Asset and people localization with access control

Initial position

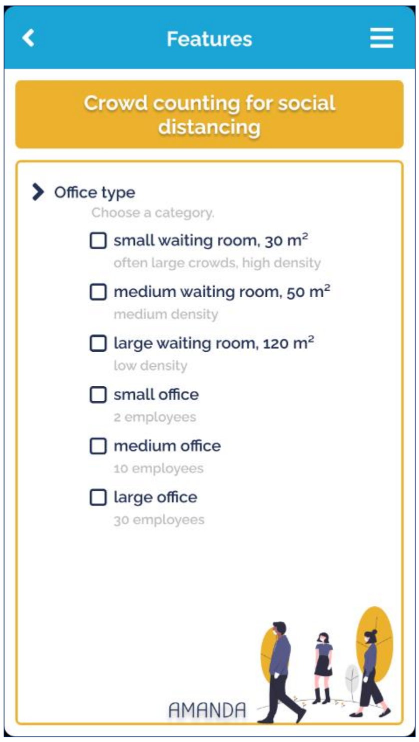
Choose a category.

- [ ] office worker
- [ ] employee that works outside of the office
- [ ] working as a driver

Figure 31 SC04: Configurable UI

3.3.4 Scenario SC06: Crowd counting for social distancing

For Scenario SC06, crowd counting, there are six available options on the office type that the user can choose. For each office type, there is a threshold defined by the edge intelligence core in order to maximize the accuracy of the prediction. The available office types are illustrated in Figure 32.



Office type
Choose a category.

- ☐ small waiting room, 30 m² — often large crowds, high density
- ☐ medium waiting room, 50 m² — medium density
- ☐ large waiting room, 120 m² — low density
- ☐ small office — 2 employees
- ☐ medium office — 10 employees
- ☐ large office — 30 employees

Figure 32 SC06: Configurable UI

4 Evaluation of a specialized low-power edge processor core

Within Task T4.3, additional architectures have been investigated, suitable for low-power edge processing. One such device is the GAP8 of the firm Greenwaves Technologies [79]. It is a device that enables artificial intelligence and signal processing at the very edge of the network [80]. Although it has not been retained for use in the AMANDA project, it is still being evaluated to understand its energy requirements and the way such a device could fit in systems similar to the AMANDA card, in the context of energy harvesting. Most of the results in this Section are from an internal report of ZHAW-InES [81].

4.1 Introduction to the GAP8

The GAP8 is a multicore system, especially designed for battery-powered high-performance applications that require a lot of processing power. It fits well for some audio and video applications. Possible applications are keyword spotting, people counting, surveillance camera. Figure 33 shows the basic architecture of the GAP8. Figure 34 gives a more detailed view of the blocks that are in the device [82].

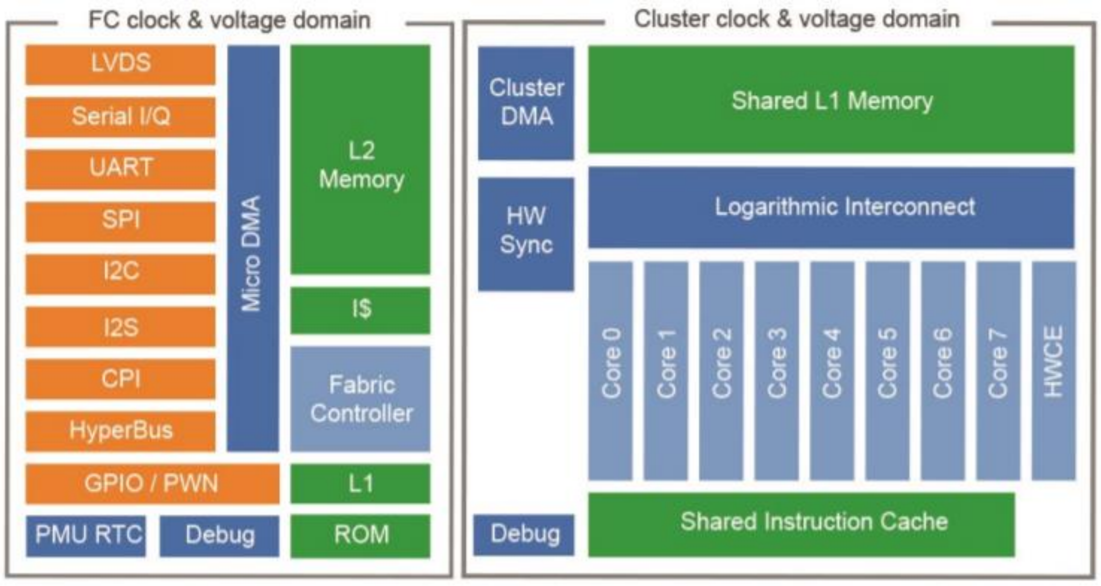


Figure 33 GAP8 Block Diagram [ref product Brief]

The device consists of 9 identical cores based on the RISC-V architecture.

- One core, the Fabric Controller (FC) is used to control the system. It interfaces with the external world, thanks to various communication peripherals. It also controls the Camera Parallel Interface (CPI) and I2S interface for audio applications. The FC can be clocked at up to 250MHz (150MHz @ 1.0V; 250MHz @ 1.2V)
- 8 cores are parts of the compute cluster. They are mainly used for computing tasks and can be clocked as high as 170MHz (87MHz @ 1.0V; 170MHz @ 1.2V)
- Near the 9 cores, there is also a HardWare Convolution Engine (HWCE) which is a Convolutional Neural Network accelerator

The power and frequency of all the cores can be adjusted on demand. Cores and peripherals are also power switchable. This helps the application adjust the energy requirements and thus optimise power consumption.

The GAP8 has several on-chip memory blocks. A ROM that is used at boot time, One Time Programmable (OTP) fuses that allows some parameters to be tuned.

The device does not have an on-chip non-volatile memory. The application must be loaded from external memory in the internal RAM, which leads to a more complex hardware and extra energy at start-up. The RAM resources are as follows [82]:

- A level 2 Memory (512KB) for all the cores
- A level 1 Memory (64KB) shared by all the cores in Cluster (0 wait state memory access)
- A level 1 memory (8KB) owned by FC (0 wait state memory access)
- Memory Protection Unit
- HyperBus Interface to connect external HyperFlash or HyperRAM

The energy performances of the GAP8 are stated by the manufacturer as below [80], [82]:

- Up to 250MHz (FC) 175MHz (Cluster) internal clock
- ~8 GOPS at a few tens of mWs
- 5x5 convolution 16 bit-fixed point in one cycle
- FC delivers 200 MOPS at 10mW @1.2V/250MHz, 4mW @1.0/150MHz
- µA deep sleep current
- 8µA to retain each of the four 128kB banks of L2 memory
- 1.2V down to 1V core VDD supply, 1.8V to 3.3V for I/Os
- 0.5ms cold boot time, 200µs to start 10µs to stop cluster

The power consumption of the GAP8 in some typical applications are provided by the manufacturer as below [80]:

- Tiny Darknet: 0.8fps, 85mW
- QVGA Face Detection: 0.4mW average per fps (3 levels Viola Jones)
- QVGA Pedestrian Detection: 3.4mW average per fps, up to 15fps (6 levels HOG)
- Autonomous Drone Navigation: QVGA 15fps, 84mW (DroNet, derived from ResNet)
- Keyword spotting CNN: 1mW per analysis per second, no accuracy compromise (open source implementation by Google. Can support up to 7 mics with beamforming, echo cancellation and noise reduction.)

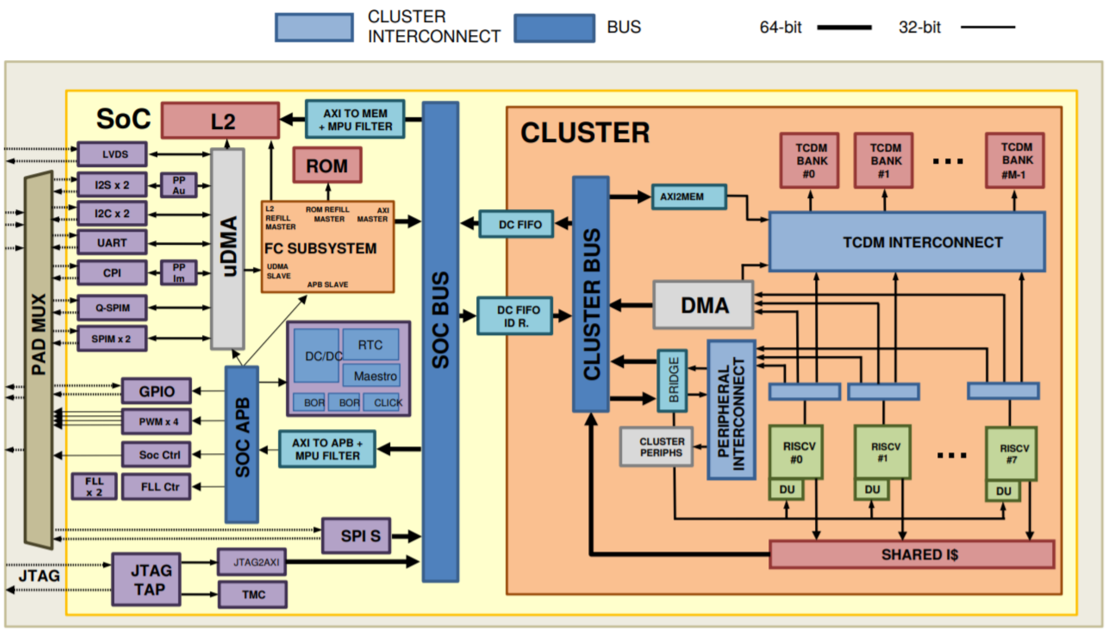


Figure 34 Main functional blocks of the GAP8

4.2 Evaluation of energy requirements

In order to evaluate the energy requirements of the device, an existing kit was used and some extra electronics added. A PCB was designed to allow a more flexible variation of the voltage

input and to introduce appropriate energy measurement points and possibilities to manually discharge the capacitors.
Figure 35 shows a block diagram of the board that was designed for the evaluation. That board is also foreseen for an application powered by a small solar cell.

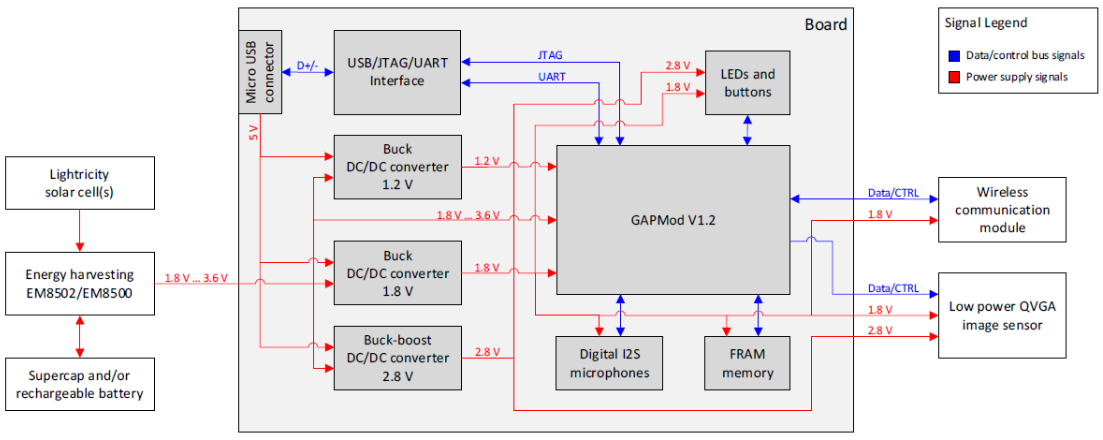


Figure 35 Concept for using the GAP8 as edge processor with harvested energy [81]

The board uses the GAPMod V1.2 from Greenwaves. That module includes basic components such as the GAP8, the external Flash memory and clock circuitry.
For energy measurement, some simple programs where written and loaded on the system. They allow the measurement of the energy requirements for different combinations of cores, at different frequencies and different voltages. The full set of measurements and their interpretation is part of an internal report of ZHAW. Only a few results and general conclusions will be presented here. Further measurements are foreseen, especially with the use of harvested energy. From an early analysis of the characteristics of the GAP8, it was decided to have a closer look at the following points:

- Energy required to boot (especially because program must be copied from an external memory into the on-chip RAM)
- Energy required during lowest power modes
- Energy required for restart from lowest power modes
- Effect of voltage and frequency scaling on power consumption

The first 3 points are important because of the potential use of duty cycling. In such cases, the GAP8 should work for short period of time, efficiently carrying out complex operations in a short time. It will then be put in a low power mode and only restarted when needed, for another step of fast computations. In such cases, the energy requirements in lowest power mode determine the minimum energy that is consumed by the system while doing no useful work. One could also switch off the GAP8 and save the energy consumed in low power mode. However, a cold restart is needed for the next set of operations. For that reason, it is important to know how much energy is needed during a cold start.
The fourth point helps determine how the frequency and voltage can be adjusted to balance the input energy and the energy required for the computations.

4.3 Results of the evaluation

Figure 36 shows the flow chart of a test used to measure energy during start up, low power modes and active mode with various numbers of nodes running.

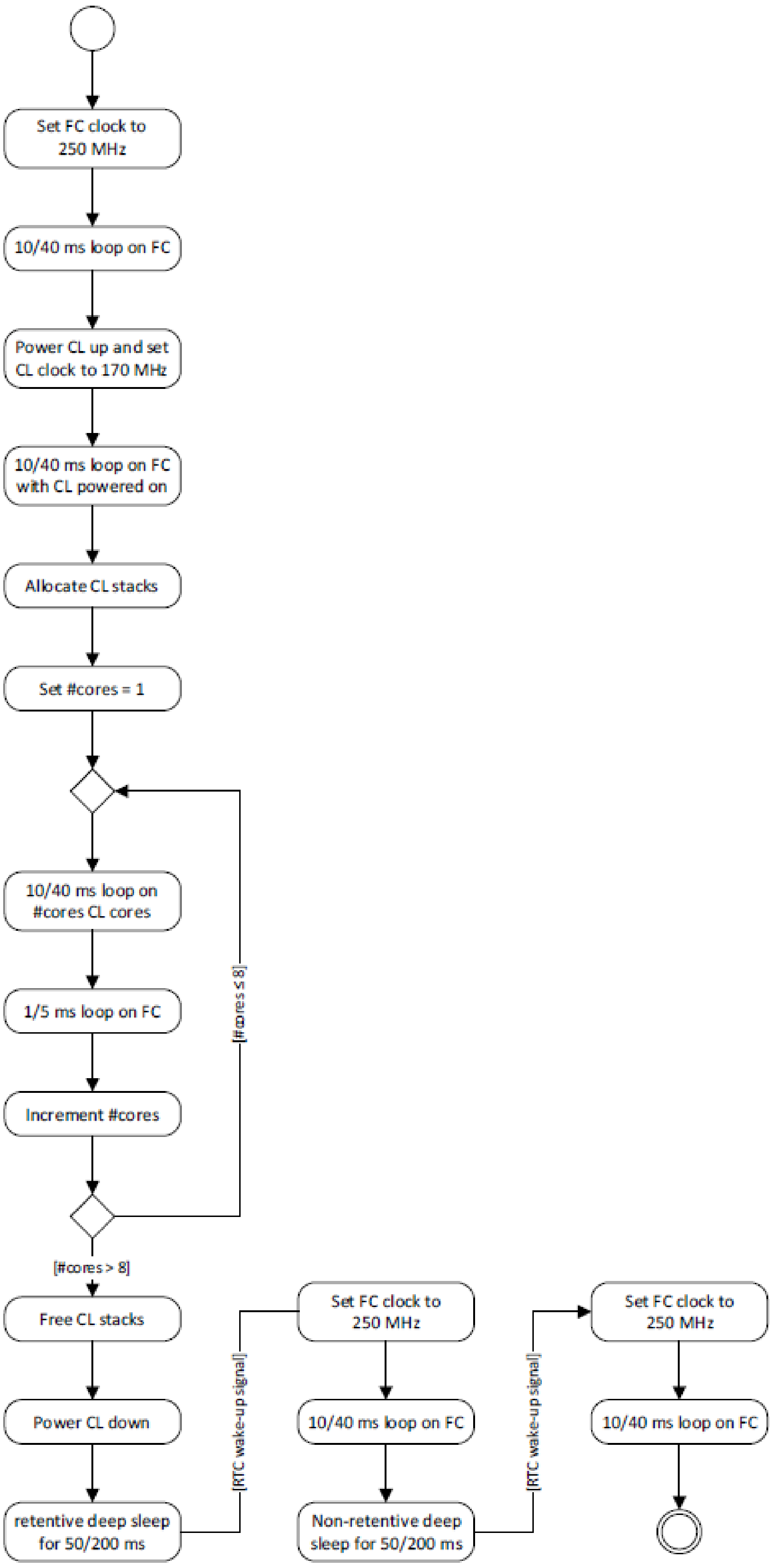


Figure 36 Flow diagram of test with various number of nodes activated [81]

The 3 power supplies of the GAP8 as set as follows: VDD_RAR = 2.6V; VDD_IO = 1.8V; VDD_XTAL = 1.2V. Figure 37 shows the resulting current measurements in sectors.

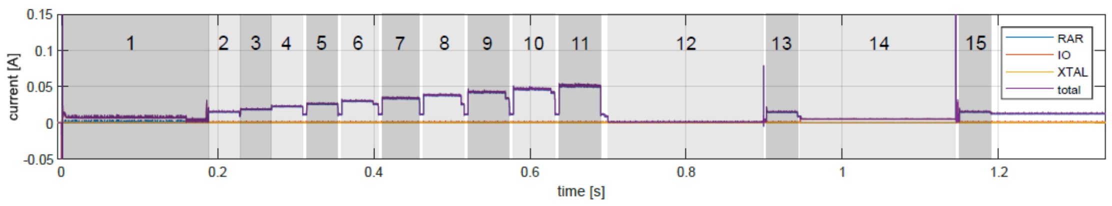


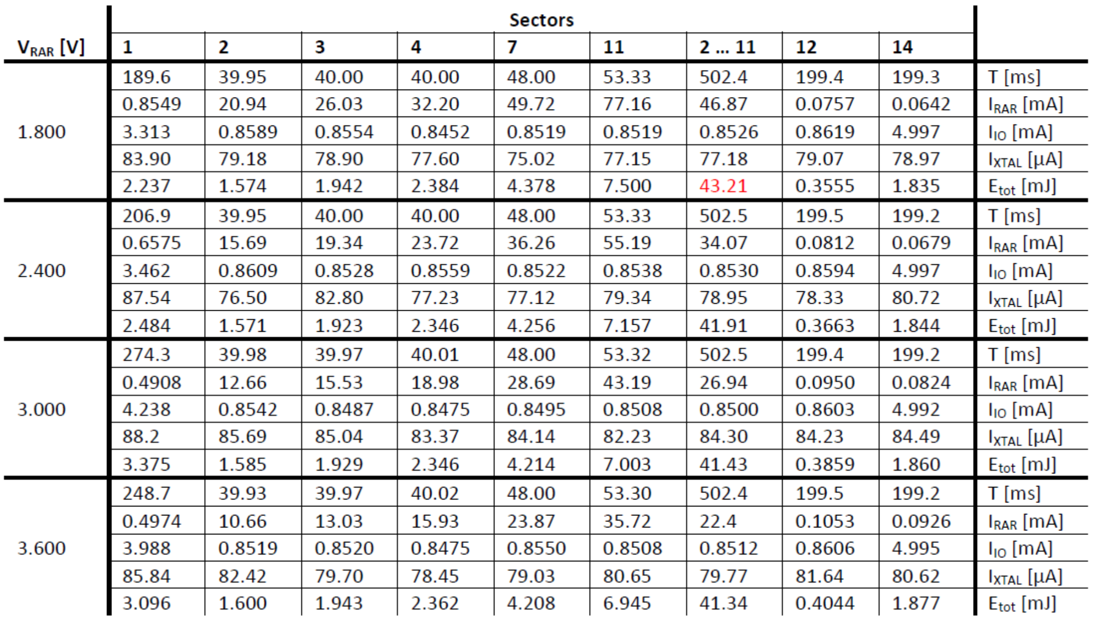
Figure 37 Current measurement arranged in sectors for interpretation [81]

1) Cold boot of the GAP8: Mainly waiting until the oscillator is stable and then loading the firmware from the HyperFlash into the GAP8's RAM (current spike at the end of this sector)
2) 40ms long busy loop executed on the fabric controller
3) Execution of 40ms long busy loop on the fabric controller. The cluster is powered on, but no code is executed on it
4) Execution of 40ms busy loop on 1 cluster core
5) Execution of 40ms busy loops on 2 cluster cores
6) Execution of 40ms busy loops on 3 cluster cores
7) Execution of 40ms busy loops on 4 cluster cores
8) Execution of 40ms busy loops on 5 cluster cores
9) Execution of 40ms busy loops on 6 cluster cores
10) Execution of 40ms busy loops on 7 cluster cores.
11) Execution of 40ms busy loops on all 8 cluster cores
12) Memory-retentive deeps sleep mode for 200ms with wake-up signal from the RTC
13) Execution of 40ms busy loop on the fabric controller after "warm-boot"
14) Non-retentive deep sleep mode for 200ms with wake-up signal from the RTC
15) Execution of 40ms busy loop on the fabric controller

In sector 1, one can observe that the IO current, which includes the HyperFlash supply current, starts high and remains at about 5mA until it goes down very suddenly and remains there until at the very end of this sector where it spikes up. This is where the firmware is read from the HyperFlash into the GAP8's RAM.

When comparing the sector 12 (memory-retentive deep sleep) and the sector 14 (non-retentive deep sleep), one observes that while in sector 12 all currents are very small, in sector 14 the IO current goes up to ca. 5mA and remains at this for the whole duration of the non-retentive deep sleep. After some investigations, it was concluded that this results from the activation of the HyperFlash chips select signal CS1#. The signal is controlled by the boot ROM and can therefore not be changed by the user. The energy costs negatively impact the use of a mode where the program needs to be reloaded.

Figure 38 shows the current consumption and the energy consumption for selected measurement sectors of four measurements with four different RAR voltages. The currents in the table are the average over the whole duration of the respective sector(s). The column labelled "2 … 11" means from the duration of the beginning of sector 2 to the end of sector 11 including the gaps between these sectors. In this time duration, the GAP8 is always executing code and therefore can be used to compare the effects of different RAR voltages. The three other parameters for this comparison are the boot-up, sector 1, and the sleep modes, sectors 12 and 14.



	Sectors									
V_{RAR} [V]	1	2	3	4	7	11	2 … 11	12	14	
1.800	189.6	39.95	40.00	40.00	48.00	53.33	502.4	199.4	199.3	T [ms]
	0.8549	20.94	26.03	32.20	49.72	77.16	46.87	0.0757	0.0642	I_{RAR} [mA]
	3.313	0.8589	0.8554	0.8452	0.8519	0.8519	0.8526	0.8619	4.997	I_{IO} [mA]
	83.90	79.18	78.90	77.60	75.02	77.15	77.18	79.07	78.97	I_{XTAL} [µA]
	2.237	1.574	1.942	2.384	4.378	7.500	43.21	0.3555	1.835	E_{tot} [mJ]
2.400	206.9	39.95	40.00	40.00	48.00	53.33	502.5	199.5	199.2	T [ms]
	0.6575	15.69	19.34	23.72	36.26	55.19	34.07	0.0812	0.0679	I_{RAR} [mA]
	3.462	0.8609	0.8528	0.8559	0.8522	0.8538	0.8530	0.8594	4.997	I_{IO} [mA]
	87.54	76.50	82.80	77.23	77.12	79.34	78.95	78.33	80.72	I_{XTAL} [µA]
	2.484	1.571	1.923	2.346	4.256	7.157	41.91	0.3663	1.844	E_{tot} [mJ]
3.000	274.3	39.98	39.97	40.01	48.00	53.32	502.5	199.4	199.2	T [ms]
	0.4908	12.66	15.53	18.98	28.69	43.19	26.94	0.0950	0.0824	I_{RAR} [mA]
	4.238	0.8542	0.8487	0.8475	0.8495	0.8508	0.8500	0.8603	4.992	I_{IO} [mA]
	88.2	85.69	85.04	83.37	84.14	82.23	84.30	84.23	84.49	I_{XTAL} [µA]
	3.375	1.585	1.929	2.346	4.214	7.003	41.43	0.3859	1.860	E_{tot} [mJ]
3.600	248.7	39.93	39.97	40.02	48.00	53.30	502.4	199.5	199.2	T [ms]
	0.4974	10.66	13.03	15.93	23.87	35.72	22.4	0.1053	0.0926	I_{RAR} [mA]
	3.988	0.8519	0.8520	0.8475	0.8550	0.8508	0.8512	0.8606	4.995	I_{IO} [mA]
	85.84	82.42	79.70	78.45	79.03	80.65	79.77	81.64	80.62	I_{XTAL} [µA]
	3.096	1.600	1.943	2.362	4.208	6.945	41.34	0.4044	1.877	E_{tot} [mJ]

Figure 38 Currents at different VDD_RAR voltage [81]

The following can be concluded from those measurements:

- Sector 1. The duration of the boot-up time varies greatly from ca. 190ms up to ca. 270ms
- Sector "2 … 11". The energy consumptions of the whole GAP8 execution phase remains relatively constant, with exception of the measurement with 1.8V RAR voltage.
- Sector 12 (GAP8 memory-retentive deep sleep mode). With increasing RAR voltage the current and energy consumptions rise. The energy consumed at VRAR = 3.6V is 13.8% higher than at VRAR = 1.8V
- Sector 14 (non-retentive deep sleep mode). High IO current of ca. 5mA. The energy consumed in this sector is accordingly high. The RAR current is about 10µA less than in the memory-retentive deep sleep

To test the impact of the frequency on energy, measurements were carried out at different frequencies, both for the FC and the cluster cores.

- In case of the FC, the system is about 20% more efficient when the maximal frequency of 250MHz is used, compared to a frequency of 50MHz
- In the case of the cluster cores the system is about 10% more efficient when the maximal frequency of 170MHz is used, compared to a frequency of 80MHz

4.4 Conclusions

The measurements taken on the custom-built hardware revealed that the GAP8's cold boot time is very long and takes a lot of energy, especially because of the HyperFlash issue. According to the hardware integration application note, most of this time is spent simply waiting for the GAP8's oscillator to become stable. The wait time can be reduced can be changed by burning the appropriate e-Fuse. The efficiency of the low power mode is also negatively impacted, when the device needs to reload the program (non-retentive low power mode). Those issues slightly limit the use of duty cycling in a system similar to the AMANDA ASSC.

5 Summary

This document is part of **Task T4.3 - Edge Intelligence and User Interfacing** of the AMANDA project. The main purpose of this task is to design and develop a multi-functional and light-weight edge intelligence engine to support the AMANDA card's low-power and intelligent capabilities, to provide a usable and configurable user interface to developers for enabling the setting up and additional development of the core engine and to investigate the embedded storage and processing optimisation of the AMANDA ASSC. This Deliverable details the implementation of different intelligence models that utilise AI techniques to provide predictive and reactive intelligence to the AMANDA ASSC. Supervised learning was used together with classification and regression ML methods to identify patterns, extract and select features, perform predictions and make decisions with minimal human involvement. Furthermore, different optimisation techniques have been applied during the development of edge intelligence strategies and algorithms to further reduce their energy requirements. During the design and evaluation phases, key performance metrics were taken into consideration related to power consumption, memory and computing power requirements.

This Deliverable includes a comprehensive State of the Art examination of edge intelligence architecture and techniques and their applicability to low-power embedded systems. Also, there is an extensive analysis of the AMANDA's edge intelligence core as well as information about the interaction between the different modular software components of the AMANDA ASSC. Following, for each Scenario the implementation of the edge intelligence core was presented with experimental results about the power consumption as well as the accuracy of predictions. Moreover, a usable and configurable interface has been designed as part of Task T4.3, which will be provided for developers to setup and develop upon the core edge intelligence engine of the AMANDA card. Lastly, a comparison between other architectures suitable for low-power edge processing was included.

The research results of this document will provide further directions to optimize the edge intelligence capabilities of the AMANDA platform. Since the individual, partner-developed components of the system are not in their final form and the miniaturized PCB prototype is still under design and development, additional evaluation and optimization of the edge intelligence component will be performed as part of **Task T5.3 - Software optimization for enhanced functionality and autonomy boost.** The results of this process will be reported in **Deliverable D5.3 - Report on Software optimization**.

6 Bibliography

[1] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo and J. Zhang, "Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing," *Proceedings of the IEEE,* vol. 107, no. 8, pp. 1738-1762, 2019.

[2] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge Computing: Vision and Challenges," *IEEE Internet of Things,* vol. 3, no. 5, pp. 637-646, 2016.

[3] H. Khelifi, S. Luo, B. Nour, A. Sellami, H. Moungla, S. H. Ahmed and M. and Guizani, "Bringing deep learning at the edge of information-centric internet of things," *IEEE Communications Letters,* p. 52–55, 2018.

[4] N. Warden and P. Lane, "The deep (learning) transformation of mobile and embedded computing," *Computer,* p. 12– 16, 2018.

[5] G. Leopold, "Edge computing seen transitioning to intelligent edge," *Enterprise AI,* September 2020.

[6] A. Singh, N. Thakur and A. and Sharma, "A review of supervised machine learning algorithms," *3rd International Conference on Computing for Sustainable Global Development (INDIACom),* pp. 1310-1315, 2016.

[7] L. Jiang, H. Zhang and Z. and Cai, "A novel bayes model: hidden naive bayes," *IEEE Transactions on Knowledge and Data Engineering,* p. 1361–1371, 2009.

[8] L. Jiang, Z. Cai, D. Wang and H. and Zhang, "Improving tree augmented naive Bayes for class probability estimation," *Knowledge-Based Systems,* p. 239–245, 2012.

[9] B.-D. S. Shalev-Shwartz, "Understanding Machine Learning: From Theory to Algorithms," *Cambridge University Press,* 2014.

[10] F. Breiman, J. Stone and C. Olshen, "Classification and Regression Trees," *Chapman and Hall/CRC,* 1984.

[11] D. Shakhnarovich, T. Indyk and P., "Nearest-Neighbor Methods in Learning and Vision," *The MIT Press,* 2005.

[12] V. Vapnik and A. Lerner, "Recognition of Patterns with help of Generalized Portraits," *Avtomat. Telemekh.,* p. 774–780, 1963.

[13] I. Goodfellow, Y. Bengio and A. Courville, "Deep Learning," *MIT Press,* 2016.

[14] K. Židek, J. Piteľ and A. and Hošovský, "Machine learning algorithms implementation into embedded systems with web application user interface," *IEEE 21st International Conference on Intelligent Engineering Systems (INES),* pp. 000077-000082, 2017.

[15] I. Čorný, "Possibilities of application of computational intelligence in monitoring of heat production and supply," *Key Engineering Materials,* pp. 560-567, 2016.

[16] I. Čorný, "Overview of progressive evaluation methods for monitoring of heat production and distribution," *Procedia Engineering,* pp. 619-626, 2017.

[17] G. Oliveira, F. Coutinho, R. Campello and M. and Caldi, "Improving k-means through distributed scalable metaheuristics," *Neurocomputing,* pp. 45-57, 2017.

[18] J. Sahoo, L. Mishra and M. and Mohanty, "GA Based Optimization of sign regressor FLANN model for channel equalization," *Recent Developments in Intelligent Systems and Interactive Applications (Iisa2016) Proceedings Paper,* pp. 124-130, 2017.

[19] J. Hou, H. Gao and X. and Li, "DSets-DBSCAN: A parameter-free clustering algorithm," *IEEE Transactions on Image Processing,* 2016.

[20] L. &. M. E. Prabha, "Hardware Architecture of Reinforcement Learning Scheme for Dynamic Power Management in Embedded Systems," *EURASIP Journal on Embedded Systems,* 2007.

[21] B. Prakash, M. Horton, N. Waytowich, W. D. Hairston, T. Oates and a. T. Mohsenin, "On the use of Deep Autoencoders for Efficient Embedded Reinforcement Learning," *In Proceedings of the 2019 on Great Lakes Symposium on VLSI (GLSVLSI '19),* p. 507–512, 2019.

[22] A. R. a. R. Sokolkov, "Learning to Drive Smoothly in Minutes," 2019.

[23] M. Merenda, C. Porcaro and D. Iero, "Edge Machine Learning for AI-Enabled IoT Devices: A Review," *Sensors 20,* 2020.

[24] S. Dhar, J. Guo, J. Liu, S. Tripathi, U. Kurup and M. Shah, "On-Device Machine Learning: An Algorithms and Learning Theory Perspective," 2019.

[25] W. Roth, G. Schindler, M. Zohrer, L. Pfeifenberger, R. Peharz, S. Tschiatschek, H. Froning, F. Pernkopf and Z. Ghahramani, "Resource- Efficient Neural Networks for Embedded Systems," 2020.

[26] E. Laftchiev and D. Nikovski, "An IoT system to estimate personal thermal comfort," pp. 672-677, 2016.

[27] S. Liu, S. Schiavon, H. Prasanna, D. Ming Jin and C. J. Spanos, "Personal thermal comfort models with wearable sensors," *Building and Environment,* 2019.

[28] T. Chaudhuri, Y. C. Soh, H. Li and L. Xie, "Machine learning driven personal comfort prediction by wearable sensing of pulse rate and skin temperature," 2020.

[29] F. Alsaleem, M. Tesfay, M. Rafaie, K. Sinkar, D. Besarla and P. Arunasalam, "An IoT framework for modeling and controlling thermal comfort in buildings Front.," *Built Environ.,* p. 87, 2020.

[30] D. Li, C. C. Menassa and V. R. Kamat, "Personalized human comfort in indoor building environments under diverse conditioning modes," *Building and Environment,* pp. 304-317, 2017.

[31] R. Sowah, K. Apeadu, F. Gatsi, K. Ampadu and B. Mensah, "Hardware Module Design and SoftwareImplementation of Multisensor Fire Detection and Notification System Using Fuzzy Logic and ConvolutionalNeural Networks (CNNs)," *Journal of Engineering,* 2020.

[32] B. Sarwar, I. Bajwa, S. Ramzan, B. Ramzan and M. Kausar, "Design and application of fuzzy logic based firemonitoring and warning systems for smart buildings," *Symmetry 2018,* 2018.

[33] F. Saeed, A. Paul, A. Rehman, W. Hong and H. and Seo, "IoT-BasedIntelligent Modeling of Smart Home Environment for Fire Preventionand Safety," *Journal of Sensor and Actuator Networks 2018,* 2018.

[34] S. Vijayalakshmi and S. Muruganand, "Real Time Monitoring of Wireless Fire Detection Node," *Procedia Technology 2016,* pp. 1113-1119, 2016.

[35] A. Dıaz-Ramırez, L. Tafoya, J. Atempa and P. Mej´ıa-Alvarez, "Wireless Sensor Networks and Fusion Information Methods for Forest Fire Detection," *Procedia Technology 2012,* pp. 69-79, 2012.

[36] H. Soliman, K. Sudan and A. Mishra, "A Smart Forest Fire Early Detection Sensory System, Another Approach of Utilizing Wireless Sensor and Neural Networks," *In Proceedings of the IEEE Sensors 2010 Conference,* 2010.

[37] E. Commision, "Passenger Cars in the EU," 2016. [Online]. Available: http://ec.europa.eu/eurostat/statistics-explained/index.php?title=Passenger_cars_in_the_EU#Overview. [Accessed 2021].

[38] Geng and Cassandras, "New “smart parking” system based on resource allocation and reservations," in *IEEE Trans. Intell. Transport. Syst. 14*, 2013.

[39] Kanan, Sweleh and Elhassan, "A Self-powered and Wireless Parking Sensor with Lux-RSS Correlation for Extra Accuracy," in *International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, 2019.

[40] Perkovic, Damjanovic, Solic, Patrono and Rodrigues, "Meeting challenges in IoT: sensing, energy efficiency and the implementation," in *Fourth International Congress on Information and Communication Technology. Springer. ICICT*, 2019.

[41] Nizetic, Djilali, Papadopoulos and Rodrigues, "Smart technologies for promotion of energy efficiency, utilization of sustainable resources and waste management," in *J. Clean. Prod. 231*, 2019.

[42] M. Cops, "Vehicle Infrastructure Integration (VII) – Bringing Vehicle Adhoc Network Technology on the road," in *Third ACM International Workshop on Vehicular Ad Hoc Networks*, 2006.

[43] G. Ali, T. Ali, M. Irfan, U. Draz, M. Sohail, A. Glowacz, M. Sulowicz, R. Mielnik, Z. Faheem and C. Martis, "IoT Based Smart Parking System Using Deep Long Short Memory Network," in *Electronics 2020*, 2020.

[44] J. Xiao, Y. Lou and J. Frisby, "How likely am I to find parking?—A practical model-based framework for predicting parking availability," in *Transp. Res. Part B Methodol.*, 2018.

[45] A. Camero, J. Toutouh, D. Stolfi and E. Alba, "Evolutionary Deep Learning for Car Park Occupancy Prediction in Smart Cities," in *12th International Conference on Learning and Intelligent Optimization*, 2018.

[46] Y. Zheng, S. Rajasegarar and C. Leckie, "Parking availability prediction for sensor-enabled car parks in smart cities," in *enth IEEE International Conference on Intelligent Sensors, Sensor Networks and Information Processing, ISSNIP 2015*, 2015.

[47] L. A. Klein, "Sensors Technologies and Data Requirements for ITS," in *Norwood, MA: Artech House*, 2001.

[48] L. E. Klein and M. a. L. A., "Summary of vehicle detection and surveillance technologies used in intelligent transportation systems, Handbook," in *U.S. Federal Highway Administration, Intelligent Transportation System*, 2007.

[49] M. H. Kang, B. W. Choi, K. C. Koh, J. H. Lee and G. T. and Park, "Experimental study of a vehicle detector with an AMR sensor," in *Sens. Actuators A*, 2005.

[50] J. Pelegrí, J. Alberola and J. R. and Lajara, "Signal conditioning for GMR magnetic sensors applied to traffic speed monitoring GMR sensors," in *Sens. Actuators A*, 2007.

[51] S. Y. Varaiya and C. a. P., "Traffic surveillance by wireless sensor networks," in *California PATH Res. Rep*, 2007.

[52] A. Haoui, R. Kavaler and P. and Varaiya, "Wireless magnetic sensors for traffic surveillance," in *Transport. Res. Part C, vol. 16*, 2008.

[53] E. Sifuentes, O. Casas and F. a. P.-A. Reverter, "Direct interface circuit to linearize resistive sensor bridges," in *Sens. Actuators A*, 2008.

[54] S. 2. Bibri, "he IoT for smart sustainable cities of the future: an analytical framework for sensor-based big data applications for environmental sustainability," in *Sustain Cities Soc*, 2018.

[55] M. Cruz, J. Rodrigues, G. Gomes, P. Almeida, R. Rabelo, N. Kumar, S. Mumtaz and P. P. W. T. S. R. J. O. M. 2019. An IoT-based solution for smart parking. In: Singh, "International Conference on Computing Communication and Cyber Security.," in *Springer Singapore. IC4S 2019*, 2019.

[56] T. Lin, H. Rivano and F. Le Mouel, "A survey of smart parking solutions," in *IEEE Trans. Intell. Transport. Syst. 18*, 2017.

[57] A. Kotb, Y. Shen, X. Zhu and Y. Huang, "iparkerda new smart car-parking system based on dynamic resource allocation and pricing," in *IEEE Trans. Intell. Transport. Syst. 17*, 2016.

[58] F. Al-Turjman and A. Malekloo, "Smart parking in IoT-enabled cities: a survey," in *Sustain Cities Soc. 49*, 2019.

[59] S. Degaonkar, M. Gupta, P. Hiwarkar, M. Singh and S. A. and Itkar, "A smart walking stick powered by artificial intelligence for the visually impaired," *Int. J. Comput. Appl.,* p. 7–10, 2019.

[60] C. V. version, "https://docs.microsoft.com/en-in/azure/cognitive-services/computer-vision/home," [Online].

[61] S. Pruthvi, P. S. Nihal, R. R. Menon, S. S. Kumar and S. and Tiwari, "Smart blind stick using artificial intelligence," *Int. J. Eng. Adv. Technol.,* p. 19–22, 2019.

[62] J. Redmon, S. Divvala, R. Girshick and A. and Farhadi, "You only look once: Unified, real-time object detection," *in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR),* p. 779–788, 2016.

[63] T. Zhang, J. Wang, P. Liu and J. Hou, "Fall detection by embedding an accelerometer in cellphone and using KFD algorithm," *Int. J. Comput. Sci. Network Secur. (IJCSNS),* 2006.

[64] D. Karantonis, M. Narayanan, M. Mathie, N. Lovell and B. Celler, "Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring," *IEEE Trans. Inf. Technol. Biomed.,* pp. 156-157, 2006.

[65] J.-I. Pan, C.-J. Yung and C. and Liang, "An Intelligent Homecare Emergency Service System for Elder Falling," *Proceedings of ICINCO 2006 - 3rd International Conference on Informatics in Control, Automation and Robotics,* pp. 203-206, 2006.

[66] W.-J. Yi, B. Wang, B. dos Santos, E. Carvalho and J. and Saniie, "Design flow of neural network application for IoT based fall detection system," *Proceedings of the 2018 IEEE International Conference on Electro/information Technology (EIT),* pp. 0578-0582, 2018.

[67] J. Lee, J. Ni, D. Djurdjanovic, H. Qiu and H. and Liao, "Intelligent prognostics tools and e-maintenance," *Computers in industry,* p. 476– 489, 2006.

[68] S. A. Asmai, A. S. H. Basari, A. S. Shibghatullah, N. K. Ibrahim and B. and Hussin, "Neural network prognostics model for industrial equipment maintenance," *11th International Conference on Hybrid Intelligent Systems (HIS),* p. 635–640, 2011.

[69] K. Zhang, S. Ren, Y. Liu and S. and Si, "A big data analytics architecture for cleaner manufacturing and maintenance processes of complex products," *Journal of Cleaner Production,* p. 626 – 641, 2017.

[70] M. Zolotova and a. I., "Comparison between multi-class classifiers and deep learning with focus on industry 4.0," *2016 Cybernetics Informatics (K I),* p. 1–5, 2016.

[71] S. Proto, E. di Corso, D. Apiletti, L. Cagliero, T. Cerquitelli, G. Malnati and D. Mazzucchi, "REDTag: A Predictive Maintenance Framework for Parcel Delivery Services," *IEEE Access,* 2020.

[72] T. Savvides, "Lightweight People Counting and Localizing in Indoor Spaces Using Camera Sensor Nodes," *Distributed Smart Cameras,* pp. 36-43, 2007.

[73] F. Conti, A. Pullini and L. and Benini, "Brain-Inspired Classroom Occupancy Monitoring on a Low-Power Mobile Platform," *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops,* pp. 624-629, 2014.

[74] A. Gomez, F. Conti and L. and Benini, "Thermal image-based CNN's for ultra-low power people recognition," *In Proceedings of the 15th ACM International Conference on Computing Frontiers (CF '18),* p. 326–331, 2018.

[75] I. Ahmad, Z. Ul Islam, F. Ullah, M. Abbas Hussain and S. and Nabi, "An FPGA Based Approach For People Counting Using Image Processing Techniques," *2019 11th International Conference on Knowledge and Smart Technology (KST),* pp. 148-152, 2019.

[76] Koller, Meli, Gysel and Würms., "Wireless self-contained remote control". Patent Patent EP2363845 (B1), 2013.

[77] A. Gagge, A. Burton and B. HC, " A practicla system of units for the description of the heat exchange of man with his enviroment," *Science,* pp. 428-30, 7 Nov 1941.

[78] ASHRAE, "Standard 55 – Thermal Environmental Conditions for Human Occupancy," [Online]. Available: https://www.ashrae.org/technicalresources/.

[79] "GAP processors," GAP processors, [Online]. Available: https://greenwaves-technologies.com/gap8_gap9/.

[80] "GAP8 product brief," [Online]. Available: https://greenwaves-technologies.com/wp-content/uploads/2021/04/Product-Brief-GAP8-V1_9.pdf.

[81] I. Plattner, *Report on VT1 Specialised project (Master class),* ZHAW: supervised by Marcel Meli, 2021.

[82] "Hardware Reference Manual of GAP8," [Online]. Available: https://gwt-website-files.s3.amazonaws.com/gap8_datasheet.pdf. [Accessed 30 05 2021].