Very Efficient Deep Learning in IoT

ICT-56-2020 - Next Generation Internet of Things

D 7.3 Second report on use case development and optimisation

Document information	
Contract number	957197
Project website	www.vedliot.eu
Dissemination Level	PU
Nature	R
Contractual Deadline	31.10.2022
Author	Micha vor dem Berge (CHR)
Contributors	Oliver Brunnegard (Veoneer), Olof Eriksson (Veoneer), Stefan Andersson (Veoneer), Nils Kucza (UNIBI), Franz Meierhöfer (Siemens), Roland Weiss (Siemens), Yufei Mao (Siemens), Andreas Ask (EmbeDL)
Reviewers	Mario Porrmann (UOS), Hans Salomonsson (EmbeDL)

The VEDLIoT project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957197.

Changelog		
v0.1	2022-01-20	Initial draft
v0.2	2022-04-01	Initial input of all partners merged
v0.3	2022-04-05	Summary, Introduction, Conclusion added
v0.4	2022-04-08	Updating all chapters, ready for internal review
v0.5	2022-04-18	Internal review back
v0.6	2022-04-25	Updated all chapters after internal review
v1.0	2022-04-30	Final version (D7.2)
v1.1	2022-08-26	Initial version of D7.3, based on D7.2
v1.2	2022-10-07	Major content of all chapters updated
v1.3	2022-10-18	Ready for internal review
v1.4	2022-10-25	Updated all chapters after internal review
v1.5	2022-10-31	Finalization
v2.0	2022-10-31	Final version (D7.3)

Table of contents

Executive Summary

This deliverable describes the use case developments and optimisations and the transition from traditional algorithms towards machine learning. It covers the work from M7 to M24 and is based on D7.2 [1], the first report on use case development and optimisation.

The two Smart Industrial IoT use cases developed automated testbeds to produce data for training neural networks. For the Motor Condition Classification, the testbed was enhanced to allow for a semi-automatic capturing and labelling of ML learning data. Also, a visually attractive mixed-reality demo software was modified to work as a demonstrator device.

For the Arc Detection use case, also the testbed was enhanced with a variable electronic load to allow a wider range of arcs to be ignited and more valuable ML data to be captured. The training and optimisation of the DL model as well as the integration of a DL accelerator in the demonstrator was the second part worked on.

The Automotive AI use case worked on the simulation of the distributed ML processing of the camera data in the car, in base stations and in the cloud. This effort is the base for the real-world integration and evaluation steps that have been defined and prepared and will be carried out in the next months.

The Smart Home use case achieved a lot of different developments and optimisations, which finally led to a ~2x improvement in performance and a ~1.7x improvement in energy efficiency. Amongst others, the major achievements were that the Smart Mirror was ported to the open source ROS2 middleware [2], first FPGA-accelerated versions for object and gesture detection were developed and benchmarked, the face-recognition was redesigned and achieves a 2x performance improvement, and a secured local voice assistant was developed.

1 Introduction

This deliverable describes the developments and optimisations performed so far in the four selected use cases. They are classified in three sections:

UC1A	Smart Industrial IoT: Motor Condition Classification use case
UC1B	Smart Industrial IoT: Arc Detection use case
UC2	Automotive AI use case
UC3	Smart Home use case

Before reading this deliverable, we recommend first reading D2.3, the definition of the use cases delivered in M9. It describes and defines the use cases in a formalised and homogeneous way. With a slight overlap, the definition of the use cases was completed and delivered, the active development phase started in M7. Since then, the four use cases are in the phase of development and optimisation, from using traditional algorithmic methods to machine learning and energy-efficient hardware, using the VEDLIoT tool flow and methods as shown in Figure 1. These developments and optimisations are described in this deliverable.

It is important to mention that this deliverable is the second of three consecutive deliverables, so it is based on D7.2 [1] and will be updated at the end of the project's lifetime. At the time of writing, all use cases have developed working demonstrators that are the base for further developments, general integrations with other VEDLIoT parts, but especially the integration of ML usage and ML optimisations.

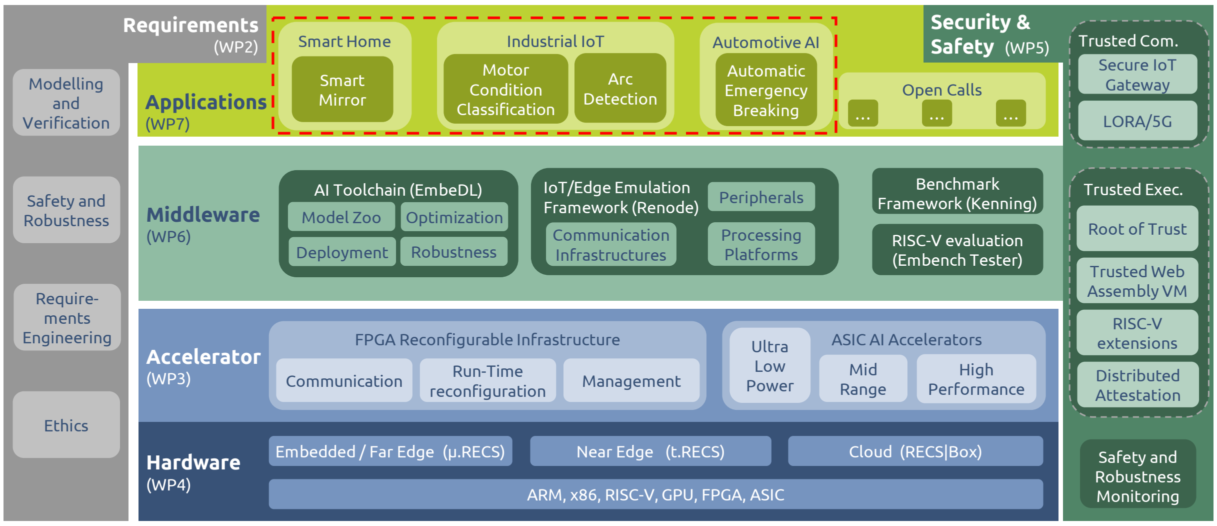


Figure 1: Global picture of the VEDLIoT project, the use cases are outlined in red

2 Smart Industrial IoT: Motor Condition Classification Use Case

The basic concept of the motor condition classification use case is introduced in D2.3 [3]. In general, an increasingly large number of components e.g. motors in industrial sites are equipped with edge devices for condition monitoring. Right now, the function of these edge devices, like the function of many other IoT devices, is limited to sense, pre-process (e.g. state-of-the-art low pass or high pass filtering, data decimation e.g. by averaging and the calculation of KPI's (Key Performance Indicator)), collect (sensors), store and communicate data. In the near future, edge devices will be used in distributed systems to gain a high-level understanding of the pure sensor data by using advanced deep learning methods, thus enabling a more efficient and sustainable operation of the production site. The distributed nature of IoT-systems will also improve the resilience of the industrial site to threats like cyber-attacks. To enable this, we introduce the first deep learning (DL) models for motor state classification.

The already in D7.2 introduced independent DL models are:

- Operational state (ON / OFF)
- Cooling system state
- Mechanical state

Due to the larger complexity, we focused our development in D7.3 on the second model "cooling system state". The development of an electronic device that monitors the correct behaviour of a fan system is in general quite simple, if a large and expensive sensor system consisting of multiple temperature sensors or air flow sensors is used. But due to economic reasons for motors with a nominal power smaller then 100kW such a complex sensor is not feasible: Thus, for the majority of the motors, an attachable sensor device which is installed in a single small case without external temperature, airflow or pressure sensors is the only realistic option. In fact, from a commercial point of view, the sensor devices for the majority of motors should only use temperature sensors which are placed on one of the PCBs that are placed on or close to the surface of the motor within the small cases of the sensor device. The other sensors of the sensor devices which are necessary to detect the operational state and the mechanical state, the vibration and magnetic flux sensors, will also deliver data containing indirect information on the cooling state. So, by using DL, it should be possible to classify the cooling state based on the information of the available sensors without additional external expensive temperature, airflow or pressure sensors.

2.1 Review on achievement in D7.2

The major achievement in D7.2 for the Motor Condition Classification use case was the testbed that allows data collection and data visualization on the iPad. The testbed was assimilated and modified for VEDLIoT based on a pre-existing testbed.

Besides the motor testbed, a conceptual architecture for final motor condition classification demonstrator was already proposed in D7.2. The final demonstrator consists of the following components.

- Motor: SIEMENS SIMOTICS SD 1LE1; 0.75 kW; 2805 RPM
- Inverter type: V20 (Power electronic to drive the motor efficient)
- Sensor device (small edge device)
    - Temperature sensor from -40 to 120°C
    - Vibration sensor with +-16 g and 3.3 kS/s
    - Magnetic field sensor with +-2 mT and 2 kS/s
    - Microcontroller: STM32L4

  - Communication module: integrated WiFi module
- AI-Accelerator: most probably the MAX78000. With its 6.6 mW (2,2mA @ 3V in run mode) the microcontroller-based accelerators are in a completely different power consumption range compared to SMARC modules or Nvidia Jetson modules with a 10 - 20 W range.
- Tablet (iPad) as HMI for the whole demonstrator and for visualisation
- Laptop (Lenovo ThinkPad P15 or better) for validation

Most of the components were physically ready in the testbed (see D7.2) except for the AI-Accelerator. Other than that, software setup was the missing component of the final demonstrator. This includes an AI model and scripts for data validation and visualisation. Therefore, these topics have been the main focus of our developments for D7.3.

2.2 Developments & Optimisation

In this section, we describe the developments and optimisation steps that have been done within this use case since D7.2.

2.2.1 Demo & testbench Development

The demonstrator architecture evolved in the meantime and now contains more details which are necessary to reach the specified goals. The new architecture is depicted in Figure 2. In general, the architecture defines some technical details comparing to the conceptual architecture presented in D7.2.

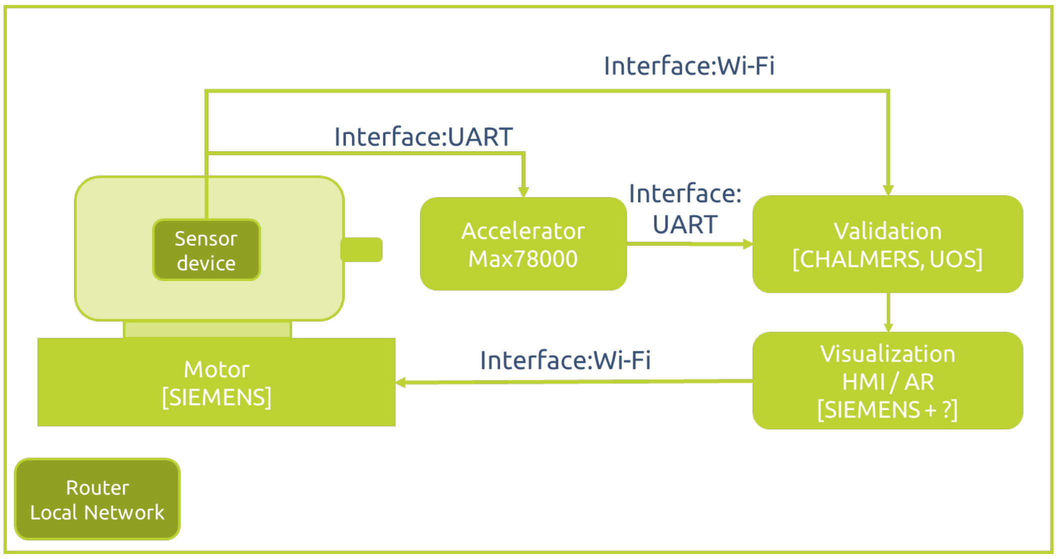


Figure 2: Evolved architecture for motor condition classification

After testing different options, we decided, to start the integration of AI accelerator with the accelerator MAX78000 from Analog (former Maxim). Max78000 is an artificial intelligence microcontroller with convolutional neural network (CNN) accelerator that enables the execution of CNN model at ultra-low power. This makes it a promising candidate for edge devices in IoT use cases. The system architecture of the Max78000, which is a mixture of a micro controller and a CNN engine, can be seen in Figure 3.

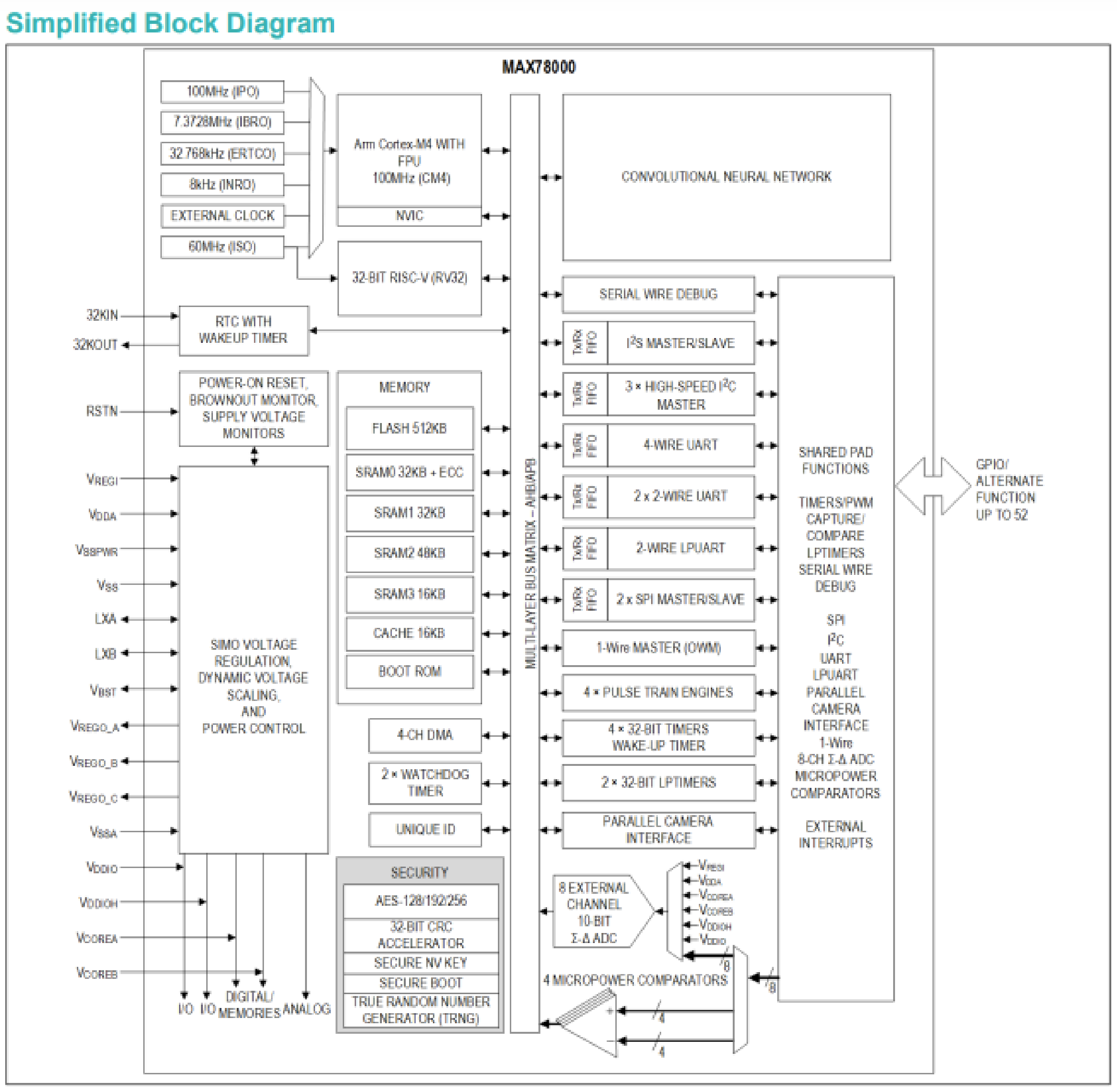


Figure 3: System architecture of the AI accelerator MAX78000

Secondly, we have defined the communication interfaces between different devices based on the specific requirements of the hardware and the firmware. Since the accelerator does not support a Wi-Fi connection, all the data from and to the accelerator are transmitted through UART. The accelerator is receiving the sensor data from the sensor device, and the classification result provided by the accelerator is forwarded via UART for validation. All the other communication channels will use Wi-Fi. A router, currently realised with a Raspberry Pi, creates a local network that connects all the other devices, the sensor device, the laptop and the iPad.

2.2.2 Experiments with the accelerator MAX78000

Before the integration of the new AI accelerator MAX78000, it was necessary to perform several tests to ensure a useful performance in our motor testbed. This included the test of basic functions which are nevertheless important for the demonstration. The different steps of the integration process are shown in Figure 4.

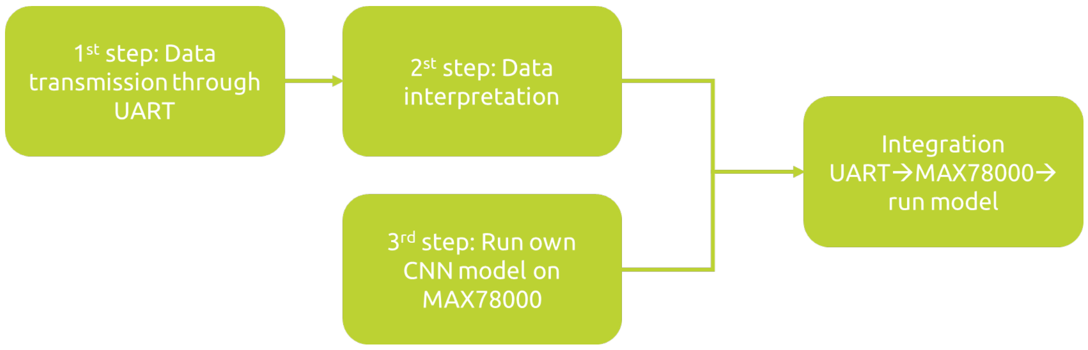

1st step: Data transmission through UART
2st step: Data interpretation
3rd step: Run own CNN model on MAX78000
Integration UART→MAX78000→ run model

Figure 4: Necessary steps for the integration of the accelerator MAX78000

The purpose of the first step, the data transmission step with UART, is to enable data transmission between the sensor device and the accelerator. To achieve this, we have done modification on the sensor device to allow UART connection between the two devices. The function of the second step, the data interpretation step, is to have a code on the accelerator for data extraction as the sensor data is presented as JSON format. The objective of the third step, which could run in parallel, is the execution of the CNN model on MAX78000. By importing existing CNN models on the accelerator, we could gain important experience with the new models. Based in this experience, we were prepared for flashing the motor condition classification model (Section 2.2.3) on the MAX78000.

To conclude, we have already successfully integrated the accelerator MAX78000 in our motor classification testbed. At the current stage, we are improving the data interpretation, the CNN model execution and the integration step. We are also still working on the improvement of our DL-models for the use case.

2.2.3 Deep Learning Model

For a high-performance DL model, a big amount of training data is required. Once the transmission interface was settled, we started the data collection according to the labelling strategy we have already proposed. The main condition that we want to focus on is the condition of the state of the cooling system (cooling state). Therefore, we have designed a shutter that could change the condition of the cooling system by manipulate the amount of air going through the air intake of the motor and thus manipulating the amount of heat which can be transferred from the motor to its surrounding. There are four categories for the air intake conditions as depicted in Figure 5, namely full, medium, small and none air intake. Based on this and different revolutions per minute (rpm), we are labelling the collected data following the categories shown in Table 1. During the data collection, we labeled the data with the name of the air intake category and the motor speed in rpm. However, when training the model, we simplified them as a binary classification problem as indicated in the table – good and bad cooling condition.

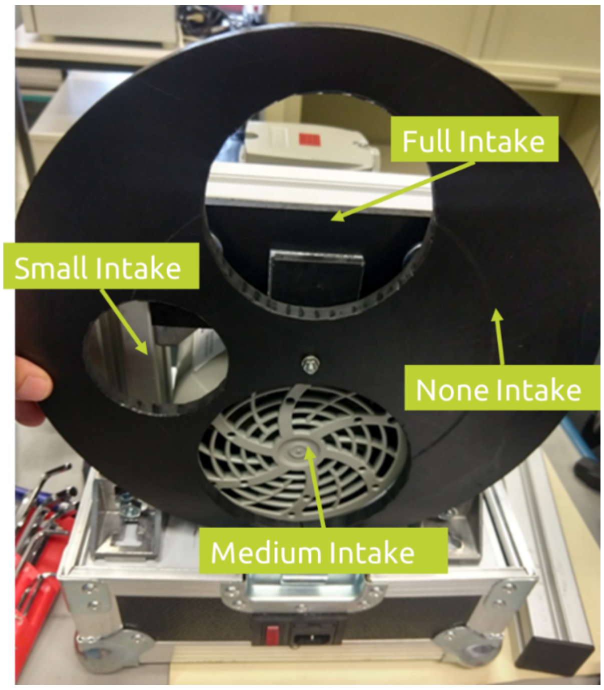


Figure 5: Shutter for condition change of the cooling system

We tried to keep the granularity high in the database so we could gain more flexibility in model training. Besides, detialed labels could also be benefitiary for future classification task on the operational state.

Rpm / Air intake	Full	Medium	Small	None
Off	Good Condition	Bad Condition, air intake maybe blocked		
300				
450				
900				

Table 1: Data collection and labelling strategy

As already mentioned in the introduction, the temperature data would also be an important piece of information for the classification of the cooling system. Figure 6 below shows the temperature change along the time.

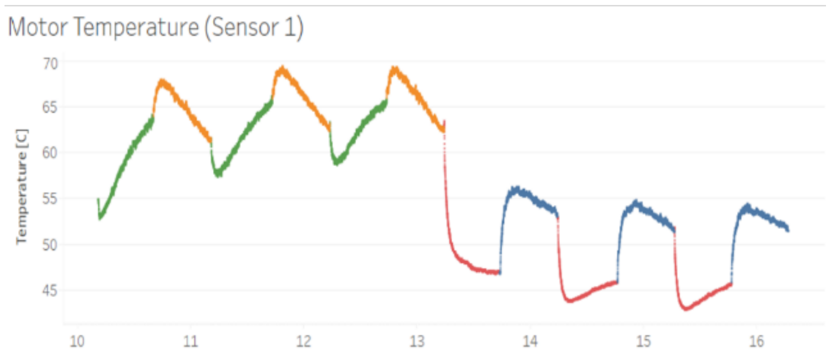


Figure 6: Example of temperature data for the motor use case

At current stage we are accumulating labelled data (based on the labelling strategy describe above) and in parallel we try to keep as much information as possible to ensure the flexibility in the ongoing training process. So, we might be able to increase the accuracy during the last year of the project.

2.2.4 Challenges and future work

We will continuously improve the whole motor condition monitoring testbed. This means that we will improve the different hardware, firmware and software components and their interaction. Currently we intent to work on the following topics:

- DL model training and optimization based on the collected data
- Integration of more advanced DL models on the accelerator
- Integration test and validation
- Hardware design for integrating the accelerator in the PCB-stack of our sensor device
- AR interface for information gained with the DL models

We see the following main challenges:

- Motor type, e.g., motor size and series: Can the same base model or pre-trained model be used for different motor series or types or is an individual model needed for all variations?
- Load: How many operational points (speed vs. load) need to be considered to get a universal model?
- Ambient conditions, e.g., temperature: How is the model affected by the ambient temperature, humidity or height of the operational place?
- Runtime of the motor, e.g., pulsed operation vs. continuous operation: Are there operational patterns that make the model-based classification impossible? What is the minimum run and stop time of the motor?

2.3 Benchmarks

The development and benchmarking have the goal to fulfil these KPI metrics:

Memory:	128 kByte SRAM 512 kByte Flash
Cost:	14 € [controller] / 70 € [System]
Power:	6.6 mW in run for the controller
Energy:	30 Wh/a
Accuracy:	0.1%

The development process of the motor condition classification use case is not yet ready for realistic benchmarking. Nevertheless, we can figure out some acceptance criteria for the system.

Assuming one wrong alert by 35k inferences a year will result in an unrealistic accuracy. If we consider that we can only use time series where we have one operational state transition of the motor, from running to stopped or vice versa, and that we have one transition per working shift, we have to aim at an accuracy of 0.1 %.

Based on the requirement on the power consumption, we could estimate the time limitation of the model inference. The microcontroller has two modes, one is running mode when inference would happen and the other is the idle mode (micro power mode). Energy consumption is different under the two modes. The electrical power of the controller is 6.6mW under running mode. In the running mode, peripherals in the system such as sensors and WiFi module are also working. With this consumption included, the electrical power of the monitoring system is estimated to be 10mW during running mode. In the idle mode, the electrical power of the system is approximate 1mW. In the monitoring system, only the microcontroller is running with micro power mode, modules such as sensors, WiFi and accelerator are shut down to save energy.

The power consumption in idle mode is the minimal requirement for the monitoring system. This makes the power consumption of the system to be at least

$$1mW \times 24\ hours \times 365\ days = 8.76Wh$$

in a year. Therefore, to achieve 30Wh power consumption per year, the system in running mode has only 21.24Wh(30Wh-8.76Wh) to use. Assuming the frequency of checking cooling condition is once every 15 minutes, this would result in 35k inference per year (4 times/hour * 24 hours/day * 365 day/year = 35040 times/year). Each inference can get:

$$\frac{30Wh - 8.76Wh}{35040} = 0.000606Wh = 0.606mWh$$

Since electrical power of running mode is 9 mW more than idle mode, the power for each inference allows an operation time of:

$$\frac{0.606mWh}{9mW} \times 3600 = 242.4s$$

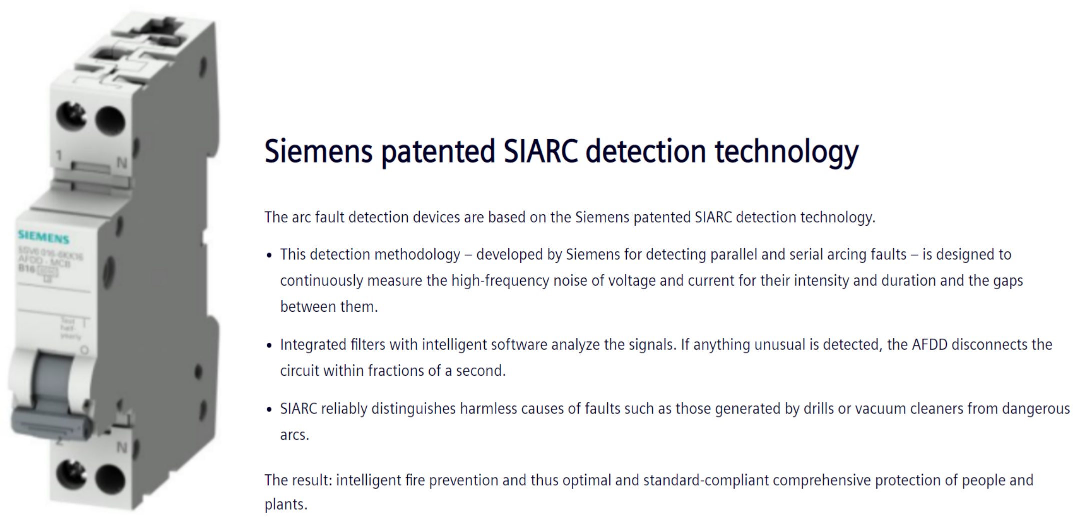
Therefore, to reach the goal of limited energy consumption, time for each inference, including data collection and calculation, should be under 4 minutes.

3 Smart Industrial IoT: Arc Detection Use Case

In the last years, DC distribution systems are gaining ground relative to AC distribution systems. Due to the improved efficiencies and the cost-efficient integration of energy storages, DC distribution systems play an important role in the renewable energy production, the ICT-Industry (Information and Communication Technology) and the electro mobility market. So, in general DC-grids will play an increasing role in the electrical energy production, consumption and distribution in the future [4] [5]. Starting from the well-known AC Arc Fault Detection Devices (AFDD), e.g., the SIARC-Technology from Siemens (see Figure 8), that use classical algorithms (see Figure 7) without deep learning, we move forward to DC arc fault detection devices using DL methods. Especially for DC systems arc detection is quite cumbersome and expensive [6].

Figure 7: Illustration of an electrical distribution system which needs an arc detection due to the complexity

Usually, optical fibres which are fastened next to each DC power line are used to detect the light which is emitted by the plasma of the burning arc. In medium voltage systems (from 1 kV AC to 60 kVAC or from 1500 V DC to 60 kV DC), additionally to the optical detection method, pressure sensors are used to achieve a reliable arc detection. The optical fibres which provide the physical parameters for arc detection are brittle and thus cannot be bent well. So, a lot of space and manual work is needed for the installation of an arc detection system.



Siemens patented SIARC detection technology

The arc fault detection devices are based on the Siemens patented SIARC detection technology.

- This detection methodology – developed by Siemens for detecting parallel and serial arcing faults – is designed to continuously measure the high-frequency noise of voltage and current for their intensity and duration and the gaps between them.
- Integrated filters with intelligent software analyze the signals. If anything unusual is detected, the AFDD disconnects the circuit within fractions of a second.
- SIARC reliably distinguishes harmless causes of faults such as those generated by drills or vacuum cleaners from dangerous arcs.

The result: intelligent fire prevention and thus optimal and standard-compliant comprehensive protection of people and plants.

Figure 8: AFDD (Arc Fault Detection Device) SENTRON 5SV6 from Siemens for AC distribution systems using a classical algorithm (SIARC) patented by Siemens

To avoid the additional costs of the optical and pressure sensor-based arc detection systems, we investigate DL methods for a current pattern-based arc detection system for DC distribution systems. Also, the performance of AC arc fault detection devices could be improved by using DL in addition to the already known classical algorithms, e.g., SIARC (see Figure 8).

In AC-Systems, long lasting and therefore dangerous arcs can very often be detected by certain current patterns which are associated with the system frequency (50 Hz/60 Hz). This typical frequency pattern of the arc current in AC systems can be identified with classical signal analysis methods. But due to the large variety of devices and installation materials in AC systems and their different frequency behaviour, even in AC systems, not all arcs can be detected by the classical algorithms. To further improve the detection rate of arcs in AC systems, we also investigate DL methods for arc detection in AC systems. If this is successful, the method could be used in the product SENTRON 5SV6 (Figure 8).

In order to keep the results comparable, the development was split into two phases:

1. Exchange of the detection method. Comparing the accuracy of the classical algorithms with DL methods
2. Exchange of the electrical distribution network. Comparing DL results of AC and DC arc test data sets

3.1 Recap on the achievements on D7.2

This project aims to develop a DL model that can help detecting arc faults in direct current power system. The main challenge is to ensure the detection accuracy and the critical requirements on the runtime version at the same time, which is also the focus of our development.

On the one side, the detection of arc faults in DC systems is more difficult than in AC systems. Therefore, we would like to experiment with DL models to improve the accuracy on detection but also on wrong positive detection, which seems to be the most critical part. But on the other side, introducing neuronal network technology in industrial detection procedures could cause higher delay times since the execution of the model can take more time. Therefore, with the help of the VEDLIoT toolflow and the integration of a DL accelerator, the execution time of DL models could be cut down and the requirements could be fulfilled.

The setup of the testbed and some experimental work was reported in D7.2. This included:

- Preparation of existing DC datasets for the first DL model.
- Preparation of AC datasets for the DL model, that could be a benchmark for evaluation in the future.
- Setup and modification of the testbench for DC arc generation and data collection. This includes the build-up of an arc generation circuit, the modification of the ADC module for the sensing of arcs and establishing of all necessary tools for data collection and labelling.

3.2 Developments & Optimisation

With the data sets generated with our testbench, we have been focusing on the model training and integration of acceleration in the demonstrator (the majority of the models were provided by EMBEDL). In this section, we describe the developments and optimisation steps that have been done within this use case since D7.2.

3.2.1 Data collection and model training

With our testbench, we have collected over 30,000 data sets for model training and validation. An example of an experimentally generated arc data set is visualized in Figure 9. The curve shows three phases of the arc generation: stable current, arc in the circuit, no current (arc extinguished).

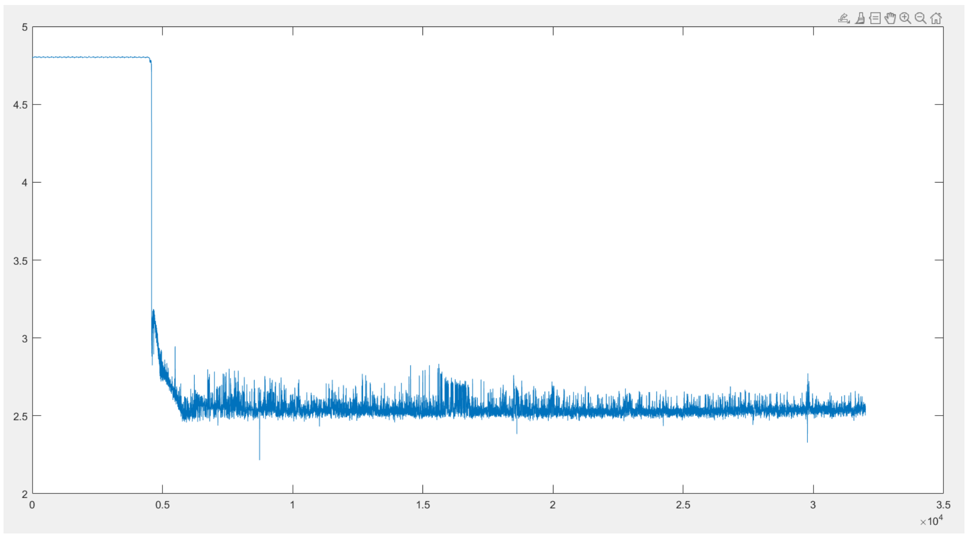


Figure 9: Arc testbench measurement with constant load

Based on the collected data, a model was trained and the tested accuracy usually reached about 97%, which is 2% higher than the accuracy of the model which was trained with the first data sets. We handle this problem as a binary classification problem. Therefore, we eliminate the data with label “unknown” between “no arc” and “arc”. We use 60% of the pre-processed dataset for training, 20% for validation and the remaining 20% for test. Figure 10 below shows the training and validation pattern for the generated DC arc data.

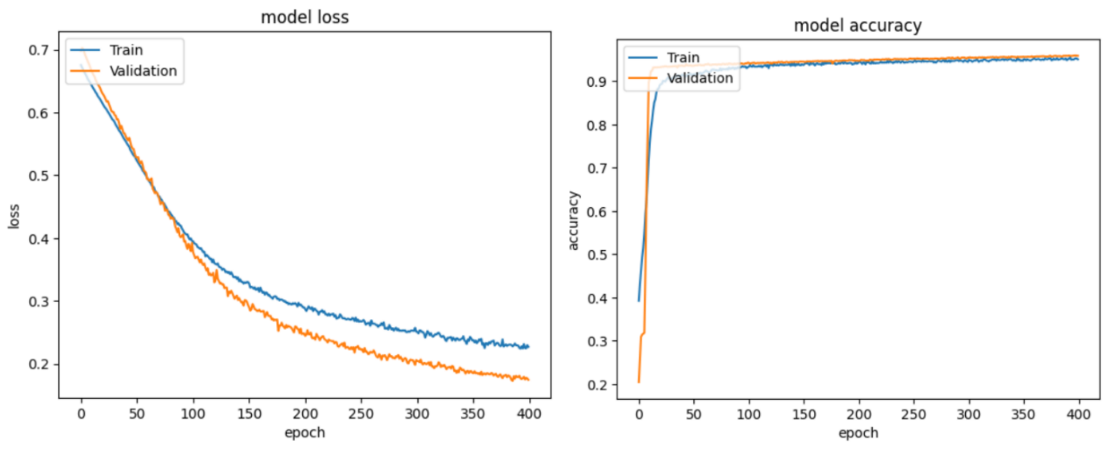


Figure 10: Model loss graph (left) and model accuracy graph (right)

3.2.2 Testbench improvement

Meanwhile, we have improved the arc generation testbench for a higher arc data quality, a faster data collection and an increased variety of the arc data sets.

During the extensive arc experiments for data collection, we have noticed that the datasets we got for the current in our experimental DC-system are too homogeneous. All the data sequences that have been fed to the model are too similar and thus will not cover the variety of current pattern, which can be found in real DC-system. This will inevitably lead to bad performance in real world scenarios. To deal with this problem and generate data with a higher variety, we have integrated an additional variable load in our arc test setup. Figure 11 shows the circuit diagram before and after the integration of our variable load.

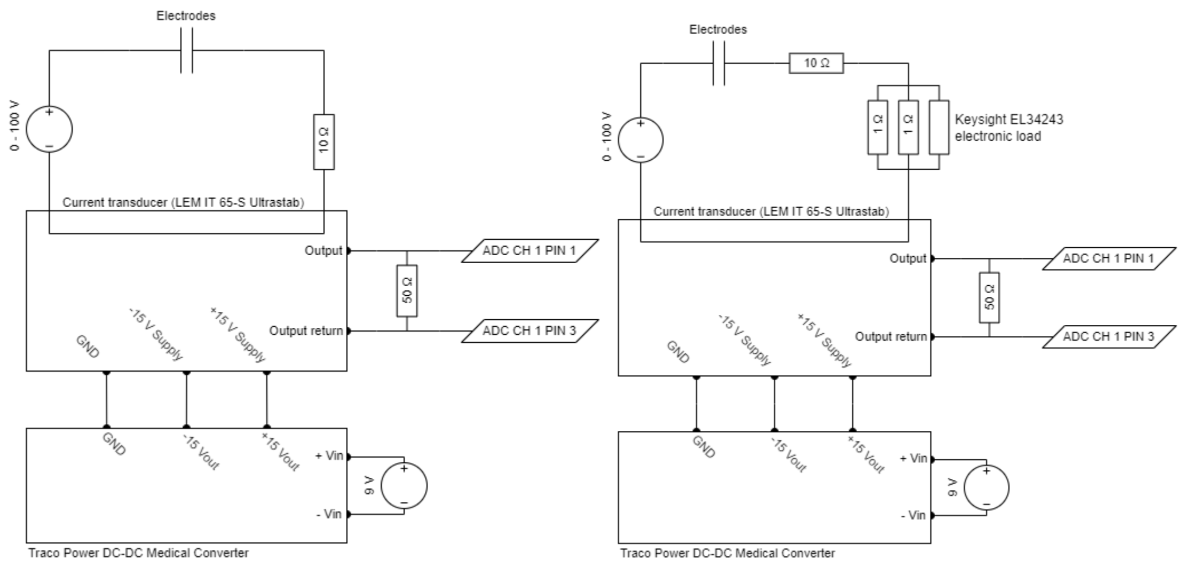


Figure 11: Testbench circuit diagram before (left) and after variable load integration (right)

The variable load used in this setup can be programmed with a list of up to 100 values and a duration for each value. For the purpose of simulating a realistic load, we recorded the load pattern of one of our lab-power-supplies and put the recorded data into that list, which then gets repeated until the end of the measurement. The influence of the variable load can be noticed in Figure 12 at "no arc" stage, which otherwise would be close to constant as shown in Figure 9. The variable load can operate in different modes, up to now we are using the low resistance mode, which allows a resistance range from 0.05 Ω to 30 Ω.

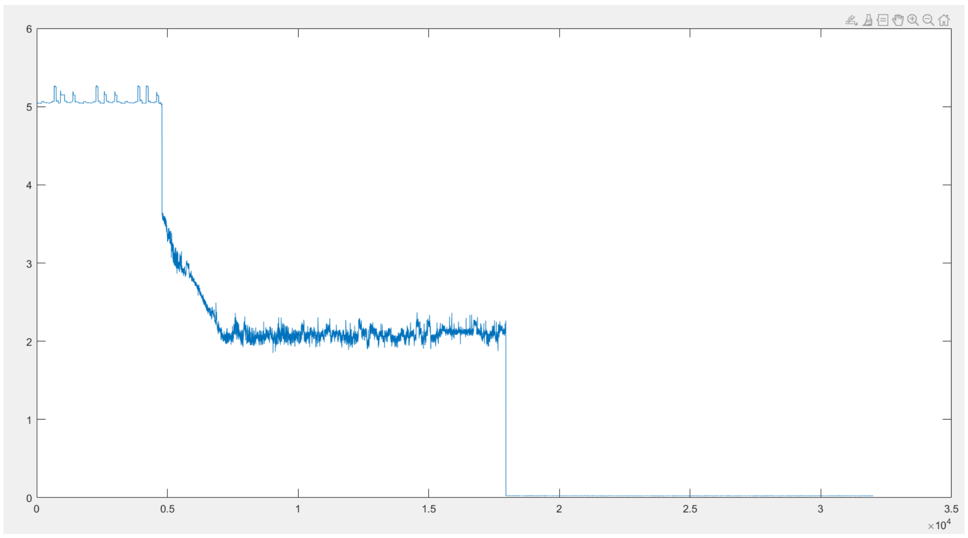


Figure 12: Arc testbench measurement with variable Load

Data labelling was done manually by marking the time point of status transition. It is easy for us to observe status transition in our controlled experimental set up. In a real DC distribution system this information is not directly available, because there not all data of the whole time are present. The deep learning model must draw the right decision by using only a small subset of the date presented in Figure 12.

Before labelling the data, we filtered the datasets. Therefore, we first levelled single-value spikes. They sometimes can be detected among the first 50 values which correspond to a time span of around 3ms. And they are most likely caused by the measuring device, or the wires or the connectors. After that the mean value of each measurement is calculated, the data sequences with a mean value under a certain threshold are excluded. These “useless” datasets are caused for example when the electrodes are too oxidized and thus no arc can be ignited. After pre-processing the data, we can label meaningful data sequences by marking the time point of transition; we can label data in different time span as “no arc”, “unknown” and “arc”.

The pre-processing procedure usually takes some time and makes the data collection relatively time consuming. Therefore, a software-application was created to realize the repeating parts of the pre-processing procedure. The semi-automated data labelling application then only requires a human operator to mark some characteristic time points on the current curve. This makes data collection more efficient and helps to collect 60% more data at the same time compared to the manually labelling in the past.

3.2.3 Accelerator Integration

Another main progress in building the demonstrator is the integration of the DL accelerator Nvidia Jetson NX. The demonstrator now has three parts as depicted in Figure 13: the arc testbench, the DL accelerator and the monitor/HMI as interface for the operator. The arc testbench itself consist of two main parts, the arc generation circuit, and the sensor device. The sensor device is the hardware mentioned in D7.2 that collects current data through a current sensor and an ADC board. The ADC board was modified to fulfil the requirements of the arc detection use case. As progress between D7.2 und D7.3 we enabled the communication between the sensor device and the accelerator through an Ethernet port by using the UDP protocol. We choose Ethernet connection with UDP mainly because this method offers a higher data transmission speed compared to a wireless connection or a TCP IP protocol. For the arc detection use case, timing is a critical factor. We would like to optimize it in every step including the communication, the data pre-processing and the inference.

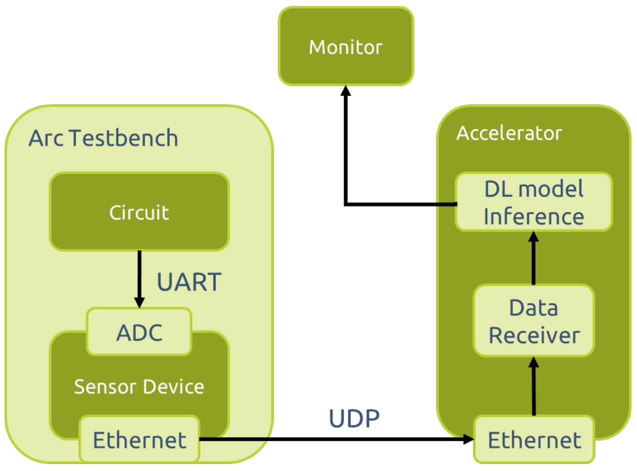


Figure 13: Demonstrator with accelerator integrated

Beside the improved setup of the communication interface, an improve timing requires high quality software that allows a fast data flow into the DL model. Like in the accelerator part, a script for receiving data, data pre-processing and model inference was created to reach the goals of D7.3. First, the raw binary data set, sent by the sensor device is received by the accelerator and later converted to a float format. The data array which consists of time sequences is then pre-processed and sent to the DL-model for inference. The DL-model provides the proper inference on the stat of the DC-system. In the end, the classification result is shown on the monitor/HMI.

The arc use case demonstrator is now at the integration test stage as important components of the demonstrator have passed module tests. The module tests include

- Sensor/ADC test: ensures that current data from ADC is reliable
- Communication interface test: ensures that data is unchanged after transmission
- Model Inference test: ensures that model inference on the accelerator has expected result after model immigration from python to C++

The system has already past main integration tests successfully. We will perform more tests and will do further optimization of all parts of the use case demonstrator to ensure the best quality/accuracy of the classification model.

3.2.4 Challenges and Future Work

First, we would like to improve the performance of the DL model. There are multiple factors that could affect the performance of trained DL models and we will work simultaneously on several of them:

- Enlarge the database. More data that are closer to real scenario will be collected.
- Improvement of the data quality by searching and collecting arc data from real scenarios. How to define and further improve data quality has been a topic of our use case. As an arc only occurs under critical condition, there are naturally less arc data than non-arc data. This leads to an imbalanced dataset, and we will try to generate more arc data to compensate it. Besides, certain data pre-processing technique could also help compensate the influence of the imbalanced datasets.
- Parameter tuning and model comparison to identify the best DL model for our use case.
- Integration in industrial IoT networks. The demonstrator runs currently in a local configuration and the classification result is simply printed on the monitor. In the future, the result should be included in the data stream of our industrial IoT network for post processing (storing, comparing with other data and so on) to realise a real monitoring system.

3.3 Benchmarks

The development and benchmarking of the arc detection use case is focusing on the following KPI metrics which was already described in D7.2:

• Accuracy:	3% improvement referring to AC AFDDs Comparable to DC AFDDs
• Execution time:	20 ms referring to AC AFDDs 11 ms referring to DC AFDDs
• Model size:	smallest and less recourse consuming model that fulfils execution time and accuracy
• Costs:	~100 € referring to AC AFDDs ~500 € referring to DC AFDDs
• Total effort:	50% or less referring to DC AFDDs Comparable to AC AFDDs

The general benchmark for the detection of arcs in AC systems is the Siemens Sentron AFDD 5SV6 (see Figure 8) with the Siemens SIARC-Technology, which is currently the industry gold standard for AC AFDD's. The corresponding standard is IEC/EN-62606.

The general benchmark for DC-AFDD's in low voltage applications is the fibre optic system, e.g., from Dehn (see Figure 14). The installation effort for those optical systems is quite big compared to AC AFDDs that are easily mounted as compact device in the distribution cabinet., which does not need further infrastructure like special sensors. The optical fibre of an optical AFDD needs a direct line of sight to the arc, and thus to as many spots as possible where arcs can occur. So, they are often placed in the remaining "open space" of the distribution cabinet (see Figure 14). Unfortunately, the "open space" is often needed as workspace for service work, change of components and so on. Therefore, the effort for service or modification increases significantly if a fibre optic system is installed.

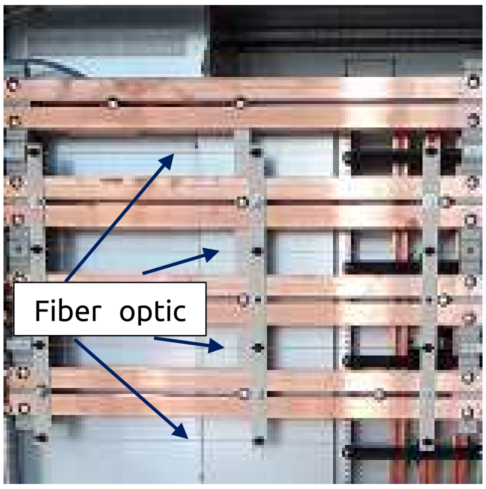
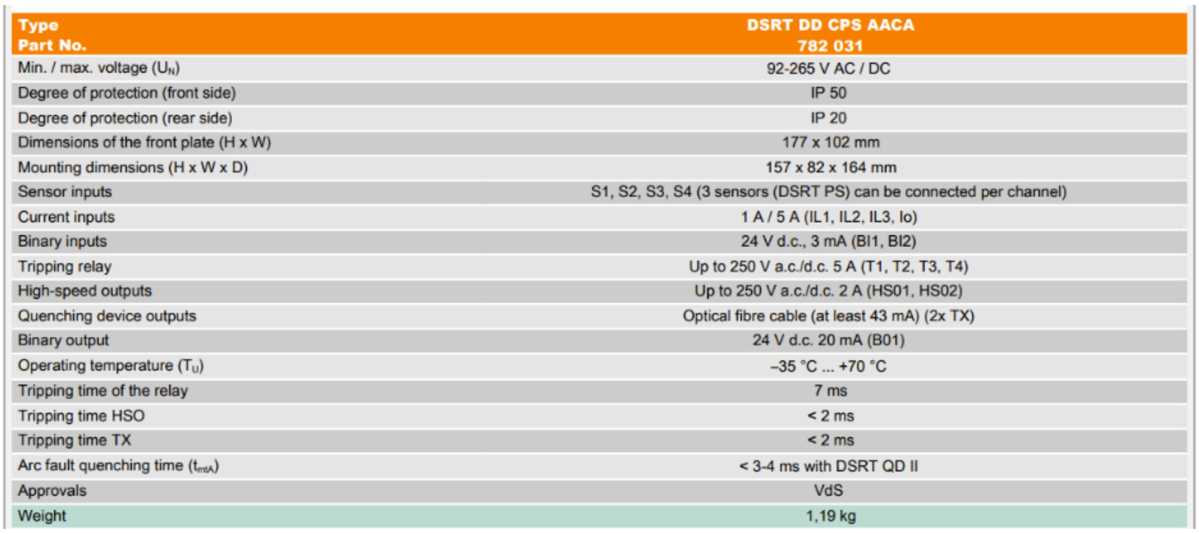


Figure 14: Distribution cabinet with optical fibres for DC arc detection

In Figure 15, the technical data for the DC-AFDD as already given in D7.2 are shown. This technical data can be used as benchmark baseline for the new development. The arc fault quenching time with 3-4 ms and the tripping time of the relay with 7 ms can be added to a reference timing of ~11 ms at least for low voltage applications.



Type **Part No.**	**DSRT DD CPS AACA** **782 031**
Min. / max. voltage (U_N)	92-265 V AC / DC
Degree of protection (front side)	IP 50
Degree of protection (rear side)	IP 20
Dimensions of the front plate (H x W)	177 x 102 mm
Mounting dimensions (H x W x D)	157 x 82 x 164 mm
Sensor inputs	S1, S2, S3, S4 (3 sensors (DSRT PS) can be connected per channel)
Current inputs	1 A / 5 A (IL1, IL2, IL3, Io)
Binary inputs	24 V d.c., 3 mA (BI1, BI2)
Tripping relay	Up to 250 V a.c./d.c. 5 A (T1, T2, T3, T4)
High-speed outputs	Up to 250 V a.c./d.c. 2 A (HS01, HS02)
Quenching device outputs	Optical fibre cable (at least 43 mA) (2x TX)
Binary output	24 V d.c. 20 mA (B01)
Operating temperature (T_U)	–35 °C ... +70 °C
Tripping time of the relay	7 ms
Tripping time HSO	< 2 ms
Tripping time TX	< 2 ms
Arc fault quenching time (t_{mtA})	< 3-4 ms with DSRT QD II
Approvals	VdS
Weight	1,19 kg

Figure 15: Technical data of an DC AFDD from Dehn, used as reference

4 Automotive AI Use Case

Light vehicles contain an abundance of ECUs today, each responsible of a well-defined component in the overall architecture of the vehicle. One example is the camera ECU which serves as the interface towards the out facing camera which is observing the environment ahead of the vehicle. The ECU is tasked with interpreting the scenario ahead of the vehicle, such as detecting actors in the environment.

Figure 16: Vehicle breaking for a pedestrian crossing the street

In the recent years, the trend in the premium segment has shifted towards centralised processing, implementing more powerful processing capabilities that is shared between multiple applications. The hardware resources for central compute is expensive which in turn results in lower end segments not being able to utilize some safety related applications.

The aim for the automotive AI use case is to evaluate how the AI can take advantage of multiple combinations of hardwares by distributing the machine learning model. Some of those resources may be outside of the vehicle, such as in the upcoming 5G NR base stations. The work includes an effort to understand the feasibility of distributing the model both inside as well as outside the vehicle whilst maintaining the integrity of the function. Three configurations of distributed processing will be developed and evaluated in the project, see Table 2.

Configuration	Sensor Adjacent “Thin Edge”	Central Compute “Thick Edge”	Base station compute “Fat Edge”
Baseline	X	-	-
Central Compute Architecture	X	X	-
Low End Vehicle without Central Compute	X	-	X
Best In Class Vehicle	X	X	X

Table 2: The configurations will be benchmarked towards the baseline configuration. The different configurations include processing hardware of different levels of processing capabilities.

The automotive use case is based on an existing vehicle safety feature known as Pedestrian Automatic Emergency braking (Pedestrian AEB). This functionality is described by, for example, the Euro NCAP organisation in [7].

The implementation for the Automotive use case is based on camera images and machine learning. The use case aims to distribute the ML inference over multiple processing nodes. This is described in Chapter 4.2.

4.1 Machine learning training data

Scenario data for the use-case has been collected. As be seen in Figure 17, the automotive use case for VEDLIoT assumes an open road with a pedestrian and possibly other objects.

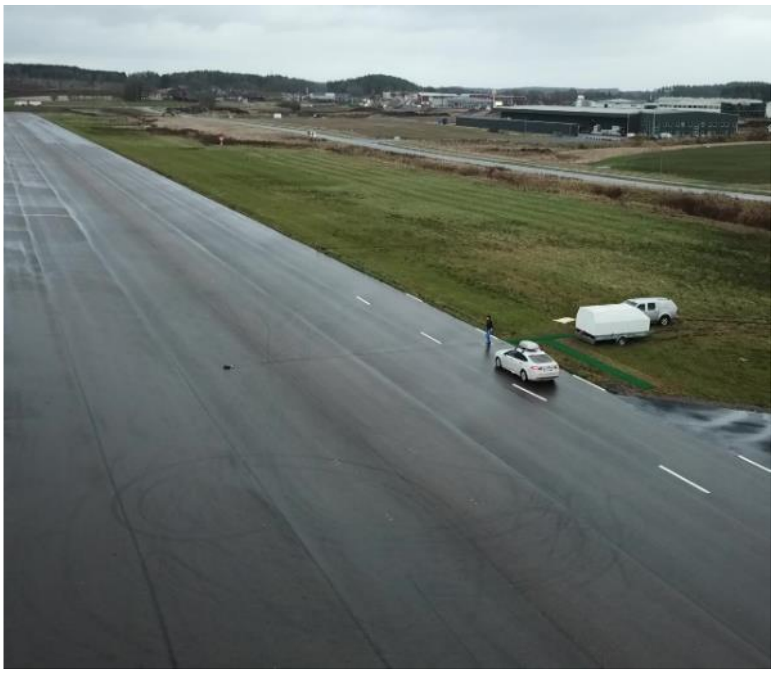


Figure 17: Bird eye view of the data collection environment at the airfield

We identified the following scenario and data variations:

- Target type
  - Pedestrian
  - Other object (trash bin)
  - No target
- Target longitudinal distance
  - Every 1 m over 1 to 100 m (handled by continuous sampling of pictures during each drive)
- Target lateral distance
  - 1, 1.5, 2, 2.5, 3 m
- Target attitude
  - Moving towards
  - Moving away
  - Crossing
- Background type
  - Tarmac
  - Grass
- Illumination
  - Sunlight
  - Cloudy
  - Rain

The data has been collected and labelled for all the combinations as described above. Data for ten runs per combination were collected, resulting in a total of 290 data sets with continuous images. These images make up the training, test and validation data.

The labelling strategy used to create labels for the data sets was to classify one of two conditions:

1. Pedestrian on the road
2. No pedestrian on the road

The classification will lead to an input to the AEB decision algorithm of stopping the vehicle to prevent a collision.

The data collection was performed with the dedicated data collection system depicted in Figure 18. It includes a camera for capturing images of the scene ahead of the vehicle, an RTK GPS to capture a very accurate position of the vehicle at the time of the image capture, and a PC used for managing the incoming data and saving to an external hard drive. This system was designed to collect and store the images used to learn the ML. It is different from the system to be used for validation. The validation system is described in Chapter 4.2.

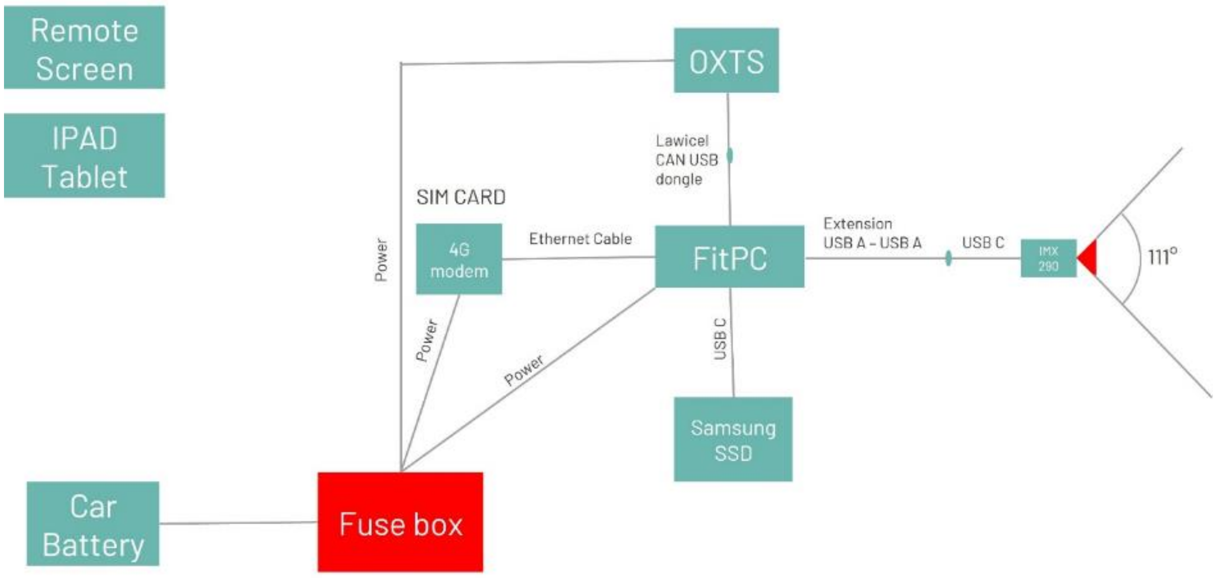


Figure 18: Overview of data collection system

The collected data was stored in a number of folders corresponding to scenario type and variations, thus, each folder only contained information of a specific scenario variation. The images were then manually labelled with one of two labels depending on whether a pedestrian was on the road ahead of the vehicle or not.

Figures demonstrating each scenario can be found in the following pages, where Figure 19 demonstrates the scenario of a pedestrian moving in the same direction as the vehicle on the side of the road. The DL model will classify this scenario as “No pedestrian on road”. Figure 20 demonstrates the scenario of a pedestrian moving towards the vehicle. The DL will classify this as “No pedestrian on road”.

Figure 21 demonstrates a pedestrian crossing the road. The DL will classify the object as “No pedestrian on road” while the pedestrian is outside of the lane markings and “Pedestrian on road” while the pedestrian is inside the lane markings.

Figure 22 demonstrates the scenario of no objects present in the scene. This will always be classified as “No pedestrian on road”.

Figure 23 demonstrates another type of object, in this case a trash bin placed at 3 different positions, present on and off the road. This will be classified as “No pedestrian on road”.

Figure 24 demonstrates the pedestrian moving away from the vehicle. This scenario is much like the one demonstrated in Figure 19 with the change in the environment around the pedestrian as she is walking on grass instead of concrete. The labelling strategy for both scenarios was the same.

The same repetition and change in the environment were made with the scenario demonstrated in Figure 20 and Figure 21.

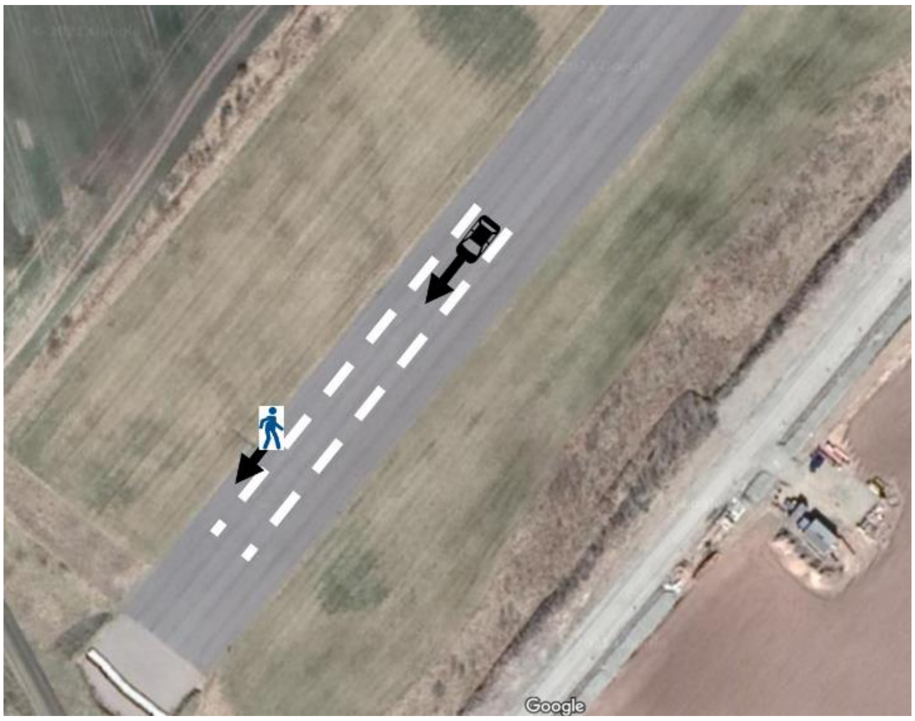


Figure 19: Pedestrian moving away from ego vehicle

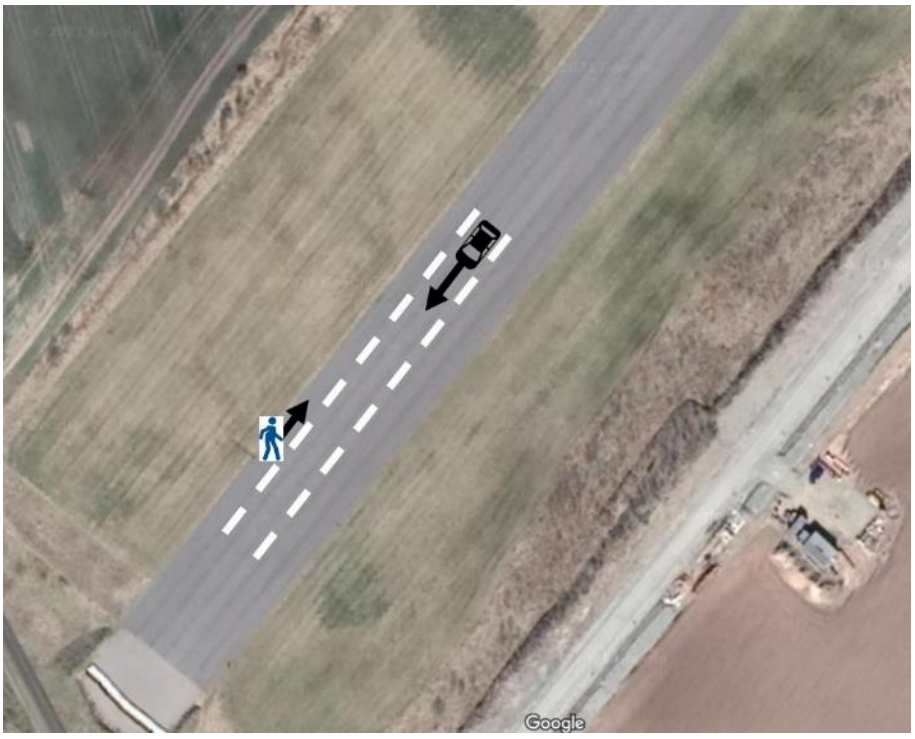


Figure 20: Pedestrian moving towards the ego vehicle

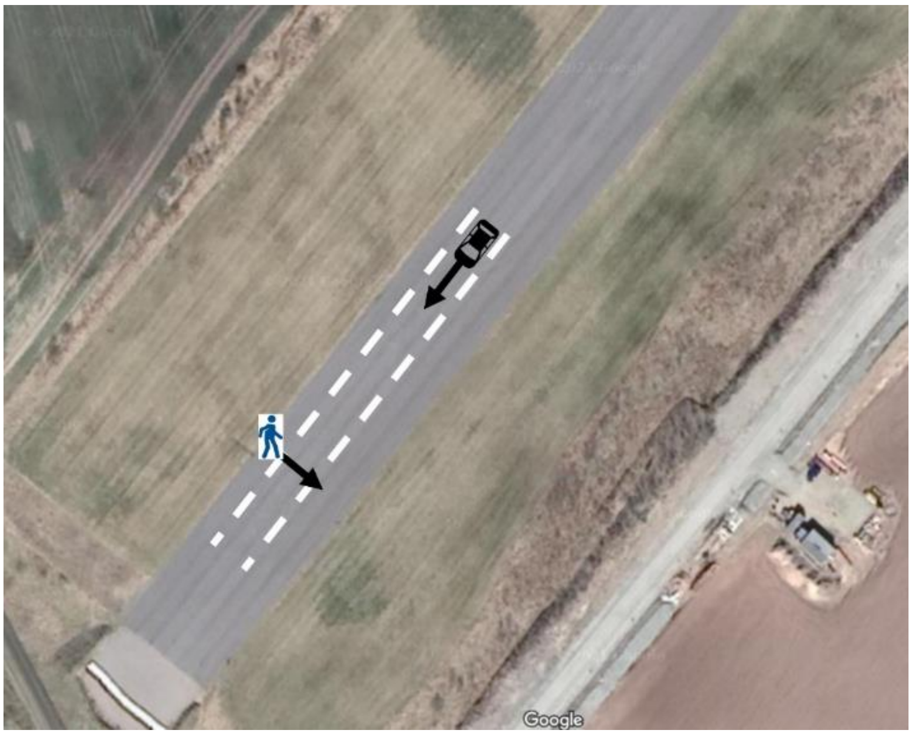


Figure 21: Pedestrian crossing the path of the ego vehicle

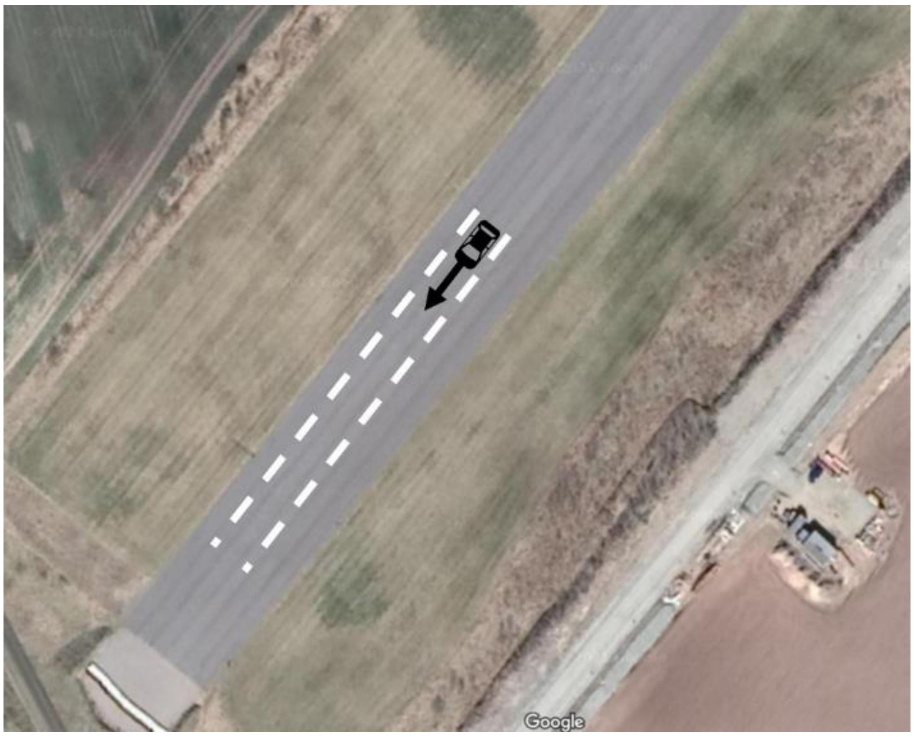


Figure 22: Empty scene

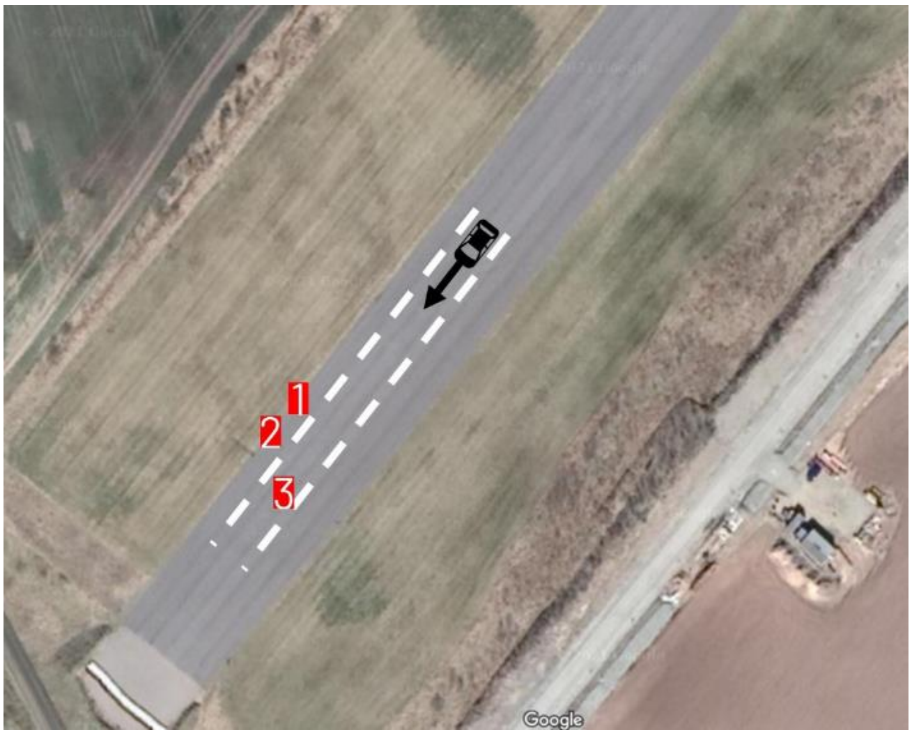


Figure 23: Other objects on and off the path of the ego vehicle

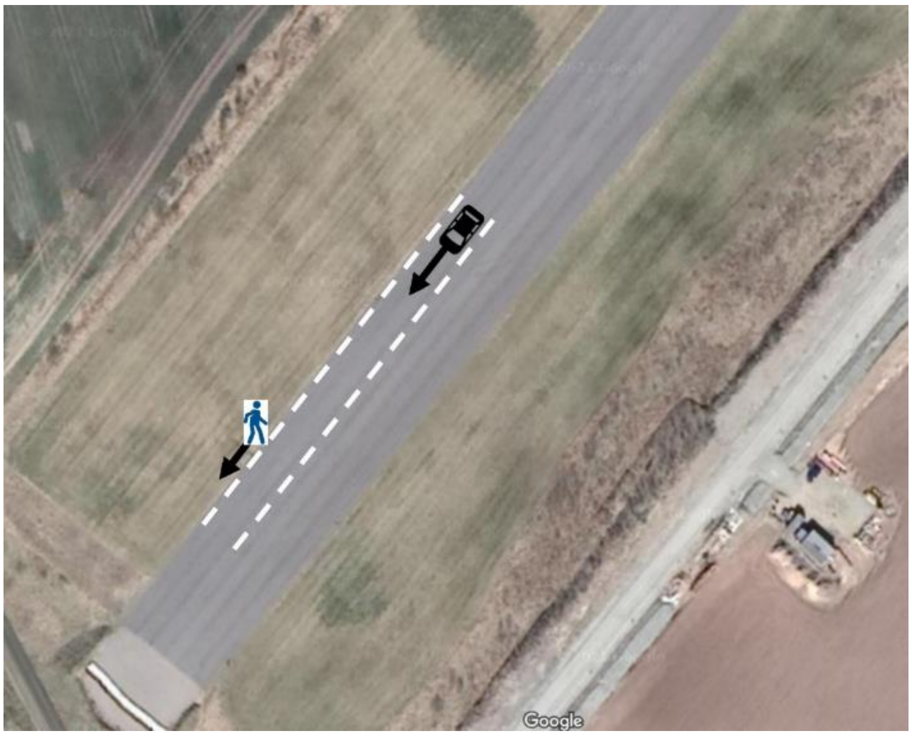


Figure 24: Pedestrian moving away from the ego vehicle on the grass

4.2 Hardware design

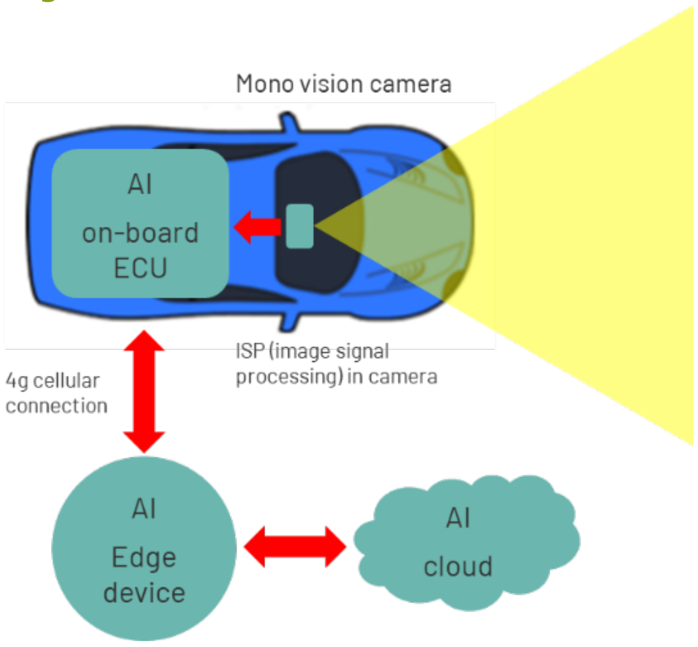


Figure 25: High level overview of the distributed AI system

The hardware design for the automotive use case assumes four possible processing nodes. These are depicted in Figure 25 and below:

- The mono-vision camera processing device
- The vehicle central processing unit (ECU)
- The 5G base station edge processor
- The cloud server

In the VEDLIoT project, we have selected to use the RECS hardware platform [8]: a u.RECS solution, see Figure 26, placed in the car and a t.RECS placed in the base station. The u.RECS contains both a SMARC module, compatible with the existing camera processing device, and a Jetson NX which equals the central computer (ECU) available in cars today. The u.RECS also supports a 5G modem for the communication to the base station, through the mPCIe interface.

The base station will have a t.RECS server installed as processing unit. The actual setup of the different microservers will be decided after local tests with the available microserver architectures.

For the time being, as a first running setup, a local usable evaluation systems will be used to replicate the u.RECS setup in the car. This will be used for local tests, developments and benchmarks. Later, this will be extended step by step and transferred until the final setup is up and running within the automotive environment.

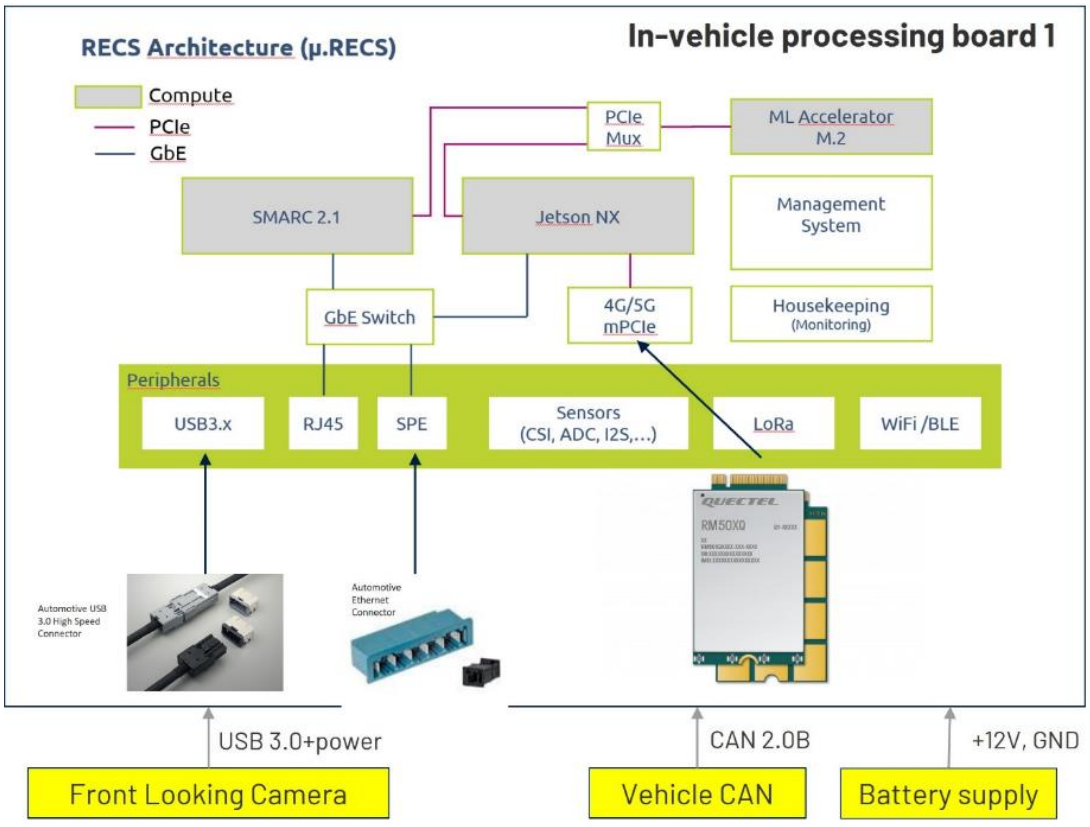


Figure 26: In-vehicle hardware design

4.3 ML model benchmarking

The decision on which ML model to be used is based on previous experience and products from the Veoneer Vision team. We will use EfficientNet. This convolutional neural network architecture has a scaling method that uniformly scales all depth/width/resolution dimensions using a compound coefficient.

The last layer of the EfficientNet-B0 architecture is changed from the 1000 classes used on ImageNet to only 2 (pedestrian on road/ no pedestrian on road). Then the model is trained on the dataset for 50 epochs with the ADAM optimiser following a one-cycle learning rate schedule where the learning rate is first increased from 1e-3 to 1e-2 over the course of 15 epochs, then decreasing to 0 for the remaining 35 epochs. The images are resized to 224x224 pixels in order to fit into the EfficientNet-B0 architecture according to the compound scaling. Furthermore, the images are randomly shifted up to 12 pixels and randomly flipped horizontally in order to counteract potential overfitting. Training is done with a batch size of 64 and weight decay of 1e-5.

The first step of training the ML model has been taken and evaluated. This model was trained on and executed on a laptop, hence differing from the system to be used during the validation later in the project. The evaluation included 7616 images to be classified. The model provided an accuracy of 93.3%, and the result is further detailed in Table 3.

6034 (79.2%) True Negatives	7 (0.1%) False Positives
501 (6.5%) False Negatives	1074 (14.1%) True Positives

Table 3: Confusion matrix of the first evaluation of the ML model

A basic processing reference using the Jetson NX only will be used to identify potential performance and latency issues.

4.4 Communication modelling

For the next step, where ML inference is distributed over multiple processing nodes, it is necessary to understand the limitation imposed by the communication between the devices. A Matlab model has been developed and will be included in the ML model optimisation (Figure 27).

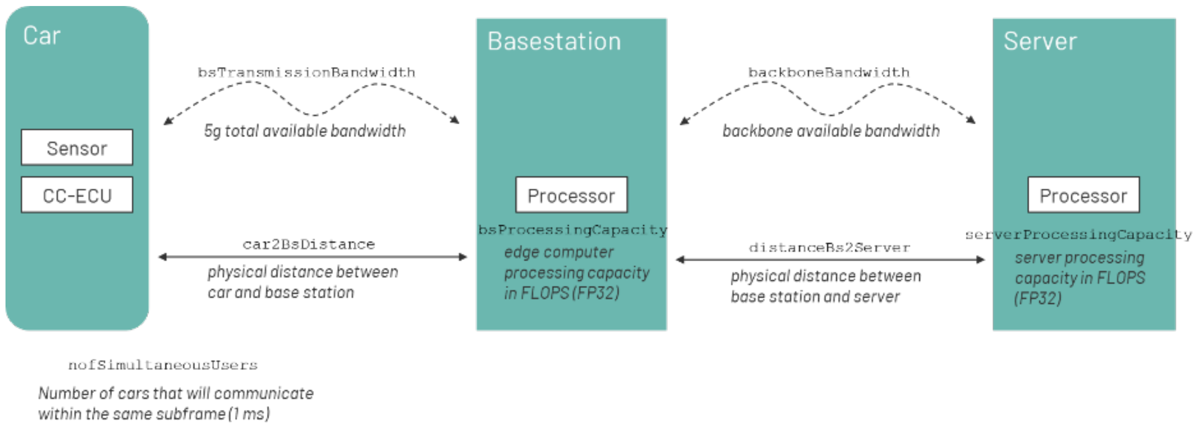

Car
Sensor
CC-ECU
bsTransmissionBandwidth
5g total available bandwidth
car2BsDistance
physical distance between car and base station
Basestation
Processor
bsProcessingCapacity
edge computer processing capacity in FLOPS (FP32)
backboneBandwidth
backbone available bandwidth
distanceBs2Server
physical distance between base station and server
Server
Processor
serverProcessingCapacity
server processing capacity in FLOPS (FP32)
nofSimultaneousUsers
Number of cars that will communicate within the same subframe (1 ms)

Figure 27: Model of communication in a 5G system with distributed systems

The purpose of the model is to provide reasonable effects, e.g., latency and packet loss, to the distributed DL system. The model induces bandwidth occupancy limitations due to an overload of user's connection to the base station. Other effects as for instance those induced on the signal due to distance to the base station can be managed in the model.

4.5 Distributed Processing Simulations

There are multiple scenarios where distributed inference is beneficial. What those benefits are, depends on one hand on the limitations of the distributed compute nodes, but also on external constraints such as data transfer speed or the number of users of a compute node. In general, the idea is to exploit differences in computational power between compute nodes by leveraging the cost of transmitting data against the performance boost of a powerful remote compute node. The objective is often to reduce the overall latency of inference, but other motivations can be driving the need for distribution, such as memory or energy restrictions of a compute node. We have started to investigate the potential of reducing the overall latency of inference.

The execution of CNNs, such as EfficientNet, is highly serialized. The execution of an operation, or block of operations, feeds into the next block of operations. The serialized execution means that the network can be "split" or divided at some point along the network. It is thus possible to perform part of the computation on one compute node and then distribute the rest of the computation to another node, which sends back the prediction. If all compute nodes have equal computational power, it would never be beneficial to distribute, as the extra latency of transmitting the output data of the first network would only increase the total inference latency. In this use case, however, the compute node at the car's sensor, the central compute node, and the compute node in the remote base station, all have different computational power. Additionally, the data size varies at each split point, as a neural network naturally compresses and decompresses the data during inference. If the data size at any potential split point in the network is smaller than the input image, it is possible to reduce the overall latency. The potential latency reduction, however, depends on the speed at which data can be transferred between the nodes.

As a first set of experiments, we have investigated distributed inference in a setup mimicking a weak compute unit at the car's sensor, and a powerful compute unit at a remote base station. Specifically, we have considered a Raspberry Pi 4, and an NVIDIA Xavier. Ultimately, a similar analysis will be performed using the i.MX8+ device, described above, instead of the Raspberry Pi.

In Figure 28, we have calculated the total latency of EfficientNet for a number of different split points. The first part of the computation is performed on the weak Raspberry Pi, the output of that computation is fed as input to the second part of the split network. The extra latency due to data transfer and resource allocation is estimated using the MATLAB model described in section 4.4. For realistic parameters, we conclude that distributed inference has the potential to be highly beneficial. When comparing the overall latency between performing the full computation locally and choosing the optimal split, the latency is reduced from 120 ms to 77 and 66 ms in Figure 29 (a) and (b), respectively. The two parameter regimes we consider are: (a) a car 500 m from the base station, with 1 Gbit/s basestation bandwidth, no other users (cars), and (b) a car 500 m from the base station, with 5 Gbit/s base station bandwidth and 3 other users (cars).

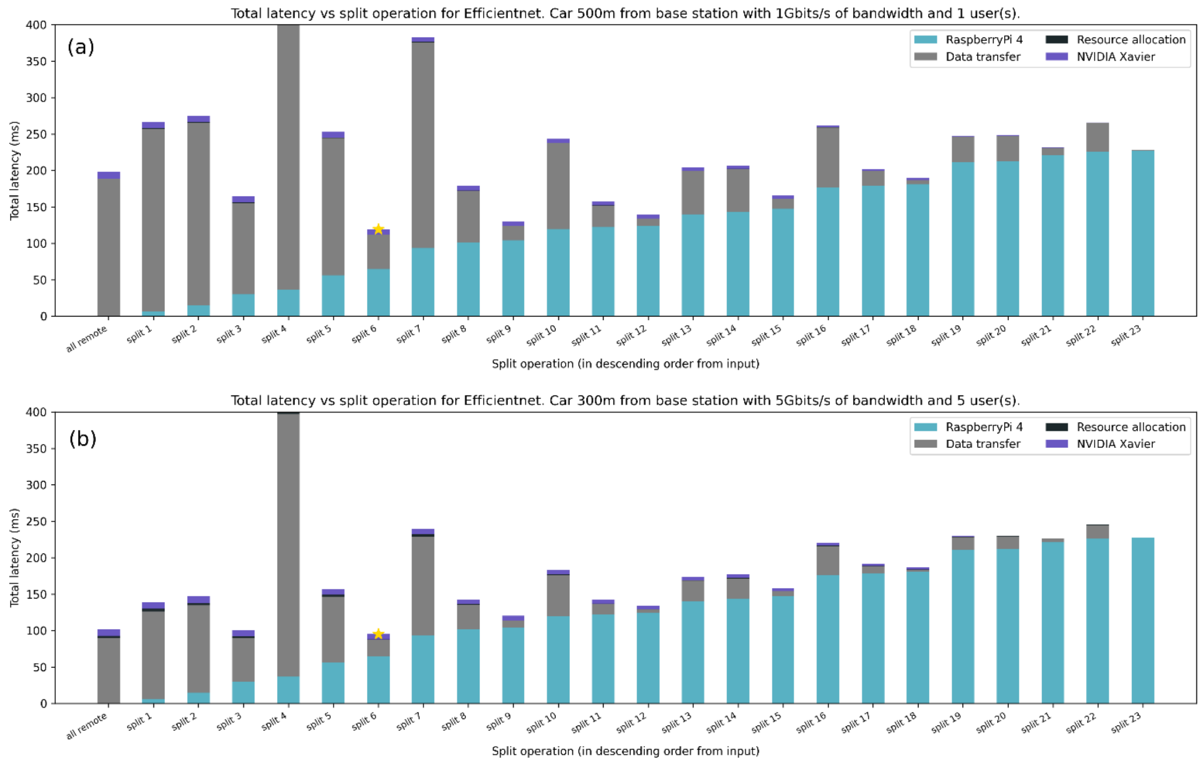


Figure 28 Simulation of total latency when splitting the neural network

4.6 Base station installation

Veoneer is preparing the installation of a 5G base station on the Vårgårda Airfield test facilities. The base station will include a dedicated access for Veoneer test vehicles, aimed to allow for controlled communication testing between vehicles and the systems in the base station. The base station will be installed with two antennas on top of the control room at the airfield. The antennas will be directed to cover the test track within a radius of 250 m.

4.7 Evaluation

Tests will be done during the spring of 2023 to collect the necessary data to capture results on latency, robustness, detection rate and energy consumption. This will be quantified for the configurations presented in Table 2.

- The latency will be monitored and logged on a round-trip basis as the final measurement to be evaluated. It will however need to be logged for the components of the system, e.g. the latency occurred by the inference, wireless transfer and memory operations. The measurement will start when an image is captured from the camera.
- The robustness is a measure of the number of packets that are lost in the communication to and back from the base station.
- Detection rate is the ratio of how often the ML model accurately classifies the scene with “Pedestrian on road” or “No pedestrian on road”.
- The energy consumption will be measured for each used accelerator, two in the u.RECS inside the vehicle and one in the t.RECS in the base station.

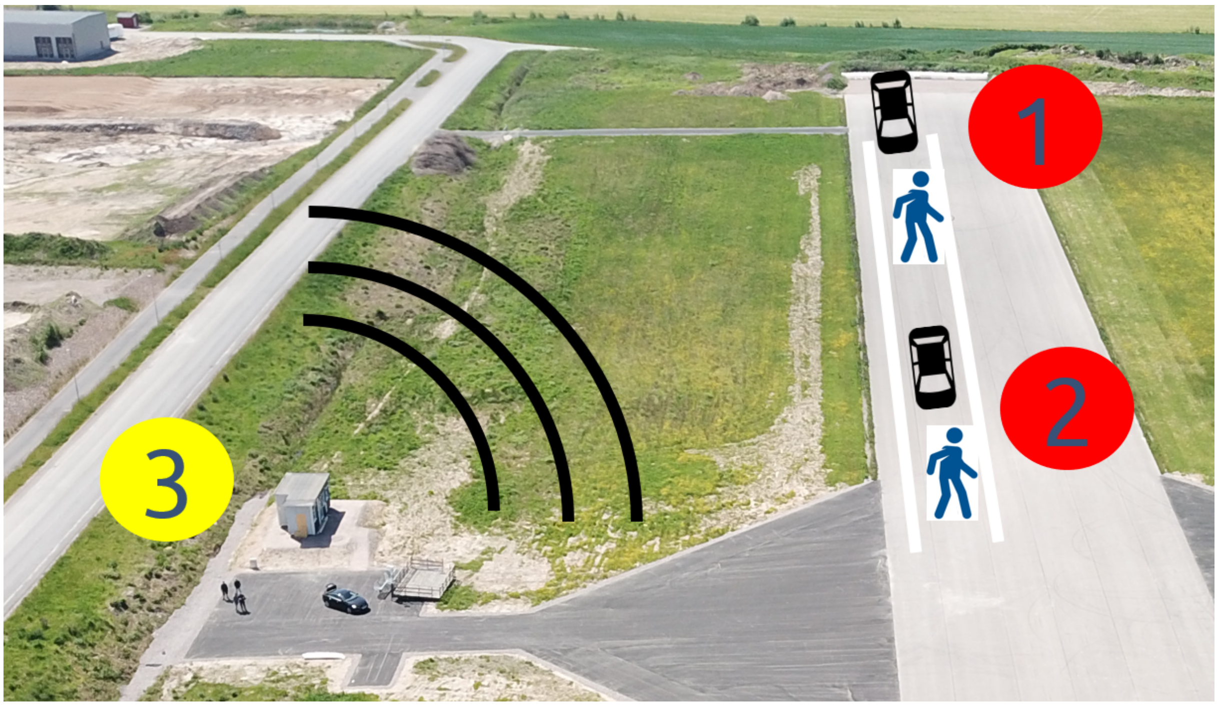


Figure 29: Testing overview. 1: far away, vehicle is outside the range of the base station resulting in poor connection; 2: good connection; 3: control room with base station mounted on the roof

5 Smart Home Use Case

The focus of the smart home use case in this project is primarily on the most computationally intensive and widely used algorithms in smart living environments. These include image processing techniques such as object detection and tracking, and recognition of things such as faces. In addition, speech and natural language processing are important topics for creating voice assistants, which have already found their way into everyday life.

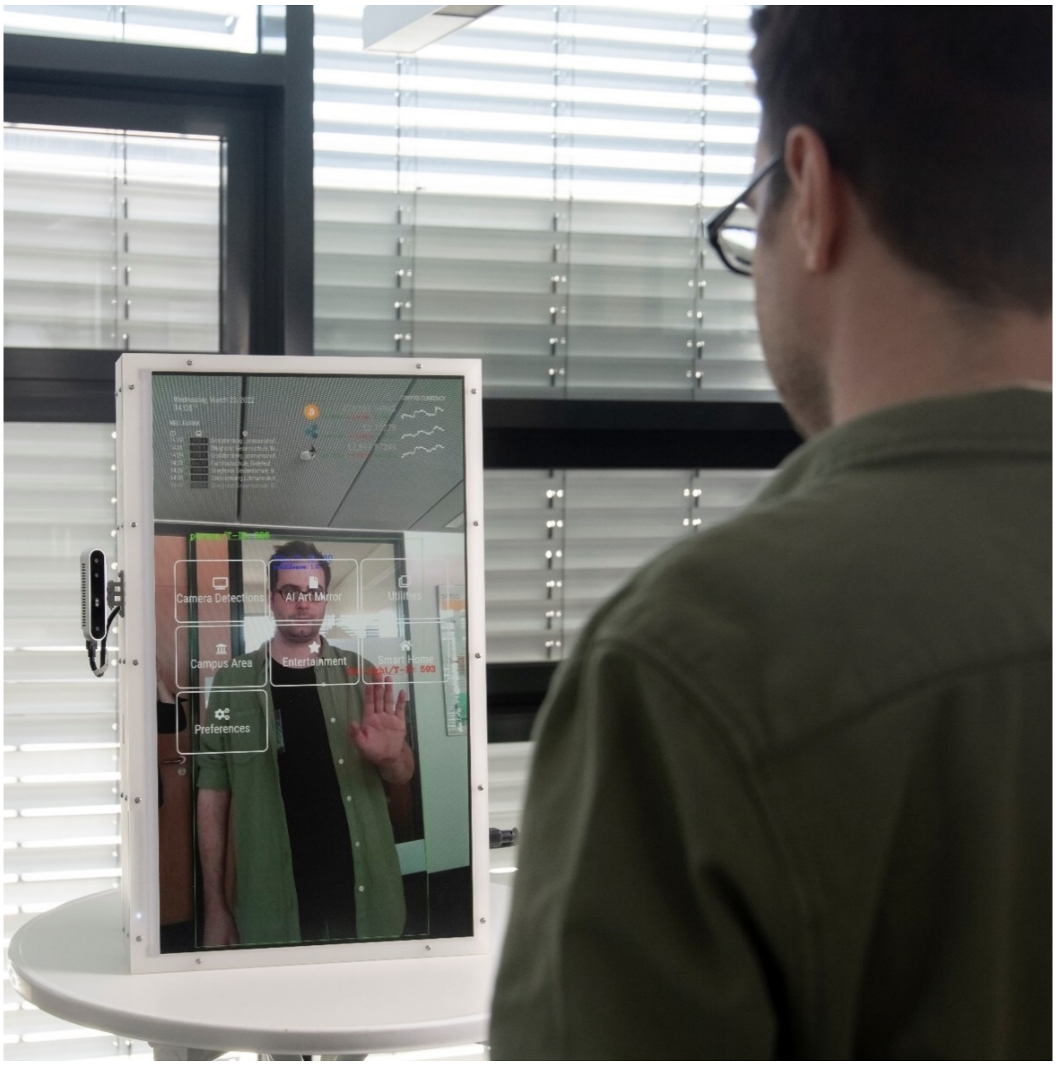


Figure 30: Smart Mirror prototype utilizing a t.RECS edge server in an acrylic case

The most commonly used features are object, gesture and face recognition, which are summarised in this use case in a demonstration case. A smart mirror provides an interaction interface between the resident and the smart home environment (see Figure 30). The residents are recognised by their faces, and personalised information is displayed next to their mirror image. The mirror can be controlled directly by gestures or voice control, for example to obtain further information about the environment or the outside world. The most important aspect for a trustworthy application is a local calculation. This is done on the hardware developed in this project and no image or information leaves the system. Of course, this must not be done at the expense of the electricity bill, which is why efficient AI hardware accelerators and adapted machine learning methods need to be used and implemented. This demonstrator builds on top of results from the previous project LEGaTO [8]; however, in this project, new aspects like the optimisation of the used neural networks, utilising a larger variety of hardware modules and implementing new features like a virtual mirror image, are addressed.

Apart from a simple hotword detection and keyword spotting, which is performed on a generated transcript, also the context of the words and sentences is analysed. As an example, for simple hotword detection, the weather widget could be shown if the words “show” and “weather” are spotted within the transcript. In this project, a speech assistant is created, which is supposed to determine the user's intention based on the spoken word with the help of techniques from the field of natural language processing. This enables various user queries to be linked within a skill system. This way, requests like "show me the weather" and "what will the weather be like" lead to the same weather skill. This is to be implemented on the smallest (most energy and cost-efficient) hardware platform possible, as the intended application is to be found in a variety of living spaces, while the smart mirror will only be found in special locations. However, both systems are based on the common middleware ROS2 [2] and the combination into an interconnected system is intended.

5.1 Developments & Optimisation

The main work in this use case is centred around a smart mirror demonstrator, which is used as a user interaction interface with the smart home environment, see Figure 30. The data path structure of this demonstrator is shown in Figure 31. Different aspects of this flow are addressed in individual tasks and will be integrated into a common demonstrator later in the project. The following subchapters will explain the revised aspects in more detail.

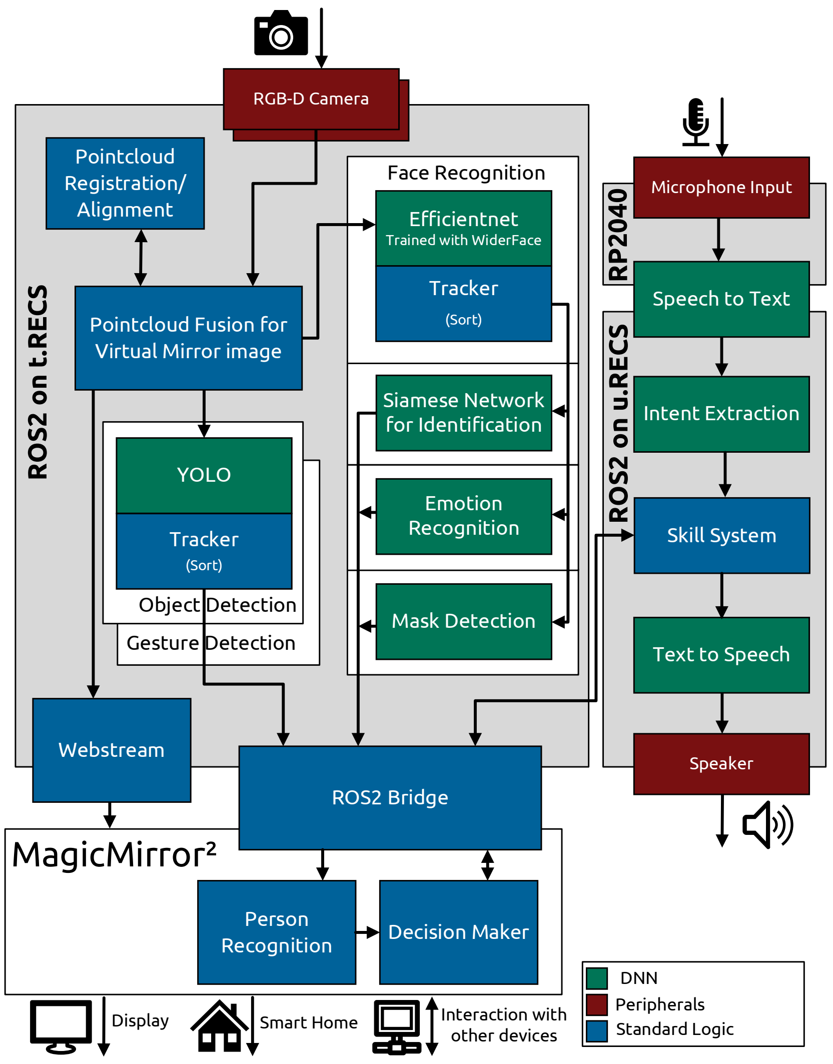


Figure 31: Data flow of the smart mirror demonstrator

5.1.1 Implementation into ROS2 Middleware

In order to have an interchangeable system, everything was implemented into ROS2 nodes since the beginning of the project. This includes image-based algorithms for object, gesture and face recognition as well as the local voice assistant and represented a significant challenge due to the current state of support of ROS2 on different platforms. ROS2 was chosen because of the quality-of-service features and the general structure of the communication. It can be seen as a state-of-the-art standard for intersystem and process communication. Sadly, the support for older versions of Ubuntu was phased out, and some patches and changes were needed to compile a custom version for the different hardware platforms. Even on the NVIDIA AGX Xavier's Ubuntu 18.04, the base system for the mirror, a customized version of ROS2 Foxy (Foxy is a ROS2 distribution) was needed.

With this ROS2 Foxy it was possible to increase the performance of the previous smart mirror from 16 FPS to about 28 FPS with a power consumption of ~100 Watt. Some memory leaks and unexpected behaviour were encountered and fixed, but the middleware used by ROS2 partly overloaded the network interface between the two Xavier systems. This leads to communication speed decreasing slowly and a total collapse after some time. A solution is not yet found, and the reason is unclear. Further investigations will only be conducted after porting to the newer firmware version.

After the release of a newer firmware version for the NVIDIA AGX Xavier and AGX Orin (Jetpack 5.0.2 build on Ubuntu 20.04) the focus switched to implementing support for the newer version of ROS2 Humble Hawksbill as it includes desired quality of service improvements. The first step in porting the smart mirror application to the new firmware version and ROS2 version was executed on the NVIDIA AGX Orin, due to the convenience of handling only the single NVIDIA AGX Orin development kit and not handling the network communication between the two AGX Xavier modules as well. The image processing parts were transferred to the newer version and are showing a performance of 30 FPS on the NVIDIA AGX Orin. Some packages for the communication between ROS2 and NodeJS are currently being worked on. They are needed for the interaction of the ROS2 nodes with the MagicMirror² application. As soon as all needed packages are implemented, the entire smart mirror can be ported to the latest firmware.

5.1.2 Redesign of the Face Recognition Method

The previous face recognition used a support vector machine to distinguish between persons. In this, newly determined feature vectors of faces can be assigned to memorised ones. However, this suffered from performance problems when the number of stored vectors was too large (additionally, more than one vector is required for each person). In the newly redesigned version, this bottleneck was circumvented with a Siamese network. A Siamese network can be trained in such a way that it outputs comparability between two vectors. If this is high enough, the faces in the two images match. For this approach, only one feature vector for each person is needed, and its effectiveness regarding partly covered faces was increased. As Figure 31 shows, facial recognition is also extended by emotion detection and face mask detection. With this, the face of the user is evaluated for the seven base emotions, like anger, happiness, or surprise, as well as if a face mask is worn correctly. This improves the situational awareness of the smart mirror. The newly structured face recognition shows improved performance of 30 FPS instead of the 16 FPS seen before.

5.1.3 Optimization of Object and Gesture Detection using the EmbeDL Toolchain

The currently used object and gesture detections are based on the DNN YoloV4. To improve the performance, the VEDLIoT tools from EmbeDL will be used to optimize it. The starting point for this is the YoloV4 network trained with the COCO dataset, which is also part of the general model zoo of this project. In the smart mirror it is used to detect persons and to obtain a greater situation awareness.

The planned optimizations should lead to an increase in energy efficiency and reduced memory footprint on the hardware. This would free memory on the hardware platform for new features of the smart mirror in the future. It will also lead to a speed-up of the model which will benefit the mirror through shorter response times in the object detection. These benefits will come with a minor decrease in accuracy, however, the lower latency could still make it a better user experience due to lower response times.

In a next step, it will be integrated into the smart mirror prototype for object detection and the network trained for gesture recognition will be improved.

5.1.4 Integration of FPGA Utilization for Object and Gesture Detection.

To evaluate the possibilities of FPGA systems, nodes for the object and gesture detection networks will be implemented on Xilinx boards. Both are based on YoloV4, and this represents the starting point for this development. In a first step, a single YoloV4 network trained with the COCO dataset is implemented, which is also part of the general model zoo of this project. In the smart mirror, it is used to detect persons and to obtain a greater situation awareness. The following steps will combine two YoloV4 networks on a single FPGA. These two networks will consist of one network trained with the coco dataset and one network on the gesture dataset.

A promising candidate for system integration is the new Xilinx Versal architecture. The heterogeneous multicore compute platform integrates ARM processor cores, an FPGA fabric, DSP engines and optional AI engines, thus combining the advantages of FPGAs and dedicated AI accelerators. First implementations on a Versal AI Core Series VCK190 Evaluation Kit [9] have been realized. Utilizing the quite small DPUCVDX8G-C32B1 Deep Learning Processor Unit (DPU) [10], which utilizes 32 AIE Cores, a performance of about 60 FPS has been achieved for a single YoloV4 implementation. The low resource requirements of the realization facilitate its implementation on the new Versal edge devices, which are envisioned for integration into the u.RECS platform. The obtained performance indicates that both required YoloV4 networks can be run in parallel on a single Versal edge device in real-time. Therefore, the architecture will be integrated into the smart mirror prototype for detailed evaluation.

5.1.5 Creation of a Virtual Mirror Image from 3D Point Clouds

In the previous setup of the smart mirror, a semi-transparent mirror film was attached to a display. With the help of this foil, the user could see both, his mirror image and personalized information. The transparency of the foil had to be adapted to the ambient conditions. For example, the shown information could not be read if there was too much light, and the mirror image could not be clearly seen if there was too little light. To bypass the problem, a virtual camera image can be used. In this case, point clouds from different angles are captured, aligned and merged.

The registration of the point clouds is implemented using a voxelized version of the generalized iterative closest point algorithm (VGICP) [11], which provides an affine transformation, estimating the translation and rotation between the two cameras. The recalculation can be executed at a low frequency as the camera does not shift that often. Subsequently, a virtual camera with the merged point cloud can determine a frontal image.

For this purpose, a processing pipeline containing ROS2 modules was implemented, and the processing was split between the two NVIDIA Xavier modules, as shown in Figure 32.

This first implementation achieved a performance of 27 FPS with a latency of 100ms, which is sufficient for real-time operation. The result was not yet used as a mirror due to image artefacts, which are currently being worked on. For integration into the smart mirror, resource utilization optimizations are also needed as it cannot be computed next to the smart mirror application. This is also a topic of current research in the project.

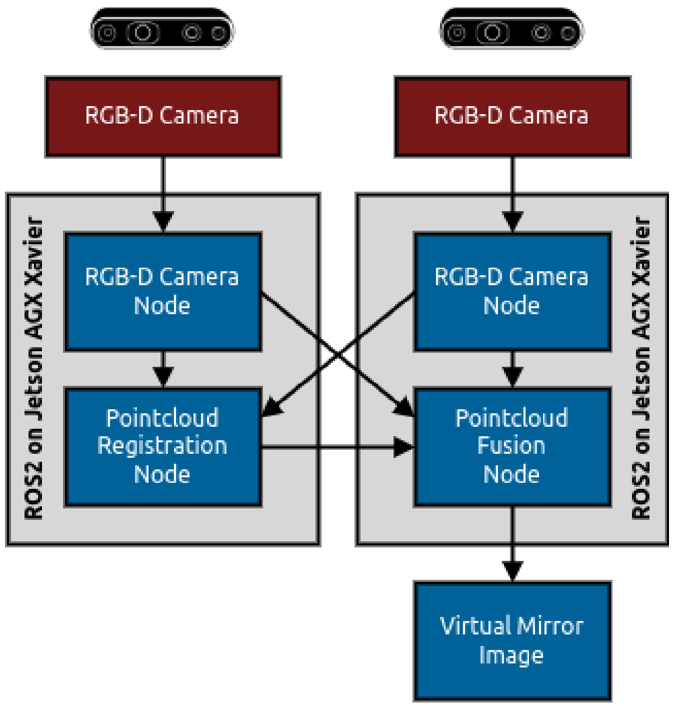


Figure 29: ROS2 node architecture for virtual mirror image creation

5.1.6 Work on Hand Dataset

For the detection of hand gestures, a dataset was created in the preceding project LEGaTO. This dataset was partially relabelled and simultaneously supplemented with the classes for persons and faces. This can be useful for a smaller version of the smart mirror. In this smaller version, the object detection DNN could be dropped as the 80 objects are not strictly necessary, but the detection of persons was covered with it. People are the main point to which both faces and hand gestures are assigned. In the used YoloV4-tiny of the smart mirror, the classes "persons" and "faces" are recognized with a mean average precision of around 90%. For all classes, the mean average precision was increased to 94%. The relabelling is ongoing work as the dataset consists of 400,000 images.

The current work also includes training on the new YoloV7, which promises an increased performance in latency and accuracy. First evaluations show mean average precision of 96% with the gesture dataset. The performance will be evaluated in the next step, due to the needed support for layers in TensorRT [12].

Due to privacy reasons, the current dataset cannot be made publicly available. In order to form a publicly available dataset, the creation of a new dataset based on commonly available sources (like YouTube videos) is evaluated in the next step. More automation is also planned, including complete label creation using different AI methods.

5.1.7 Hardware Evaluation

For the hardware used for the smart mirror, various microservers will be evaluated on the t.RECS edge server. Currently, two Nvidia Xavier modules are used for this demonstrator, and these are planned to be supplemented with an FPGA and exchanged with other modules. The microservers are interconnected via Ethernet and switched PCIe, which offers high bandwidth at a low latency. In order to simplify the exchange of modules, the detection applications of the smart mirror are migrated to the ROS2 middleware.

Evaluations for the smart mirror on the recently released NVIDIA AGX Orin have started in combination with the utilization of FPGA systems. The integration into the t.RECS edge server would be beneficial for this use case.

5.1.8 Local Voice Assistant

The previous smart mirror demonstrator centred significantly on image-processing algorithms. In addition to gesture recognition, a voice assistant is a powerful interaction capability. So far, however, only a rudimentary hotword detection has been used. In this, keywords like "show" and "weather" were detected, for example, to start the weather skill. To enhance this, a voice assistant based only on local processing will be implemented in this project. In order to be able to interact more easily with the existing demonstrator, ROS2 nodes will be implemented that can transform speech-to-text (STT) and text-to-speech (TTS) and extract the user's intention from the transcript. STT nodes with Coqui STT [13] or Vosk [14] are implemented, and the Coqui STT shows the best results. TTS nodes with Coqui TTS or PyTTSx3 [15] have been implemented and evaluated. The Coqui TTS [13] shows a more humanlike voice than PyTTSx3. The Coqui project, a successor of the Mozilla TTS/STT, shows the best results in those two areas and will be integrated into the smart mirror. The natural language processing solutions with SpaCy [16] and RASA [17] are evaluated and show comparable results. The data flow is shown in Figure 32. A simple skill system was implemented based on user intent, and the local voice assistant could present cafeteria and weather information or tell a joke.

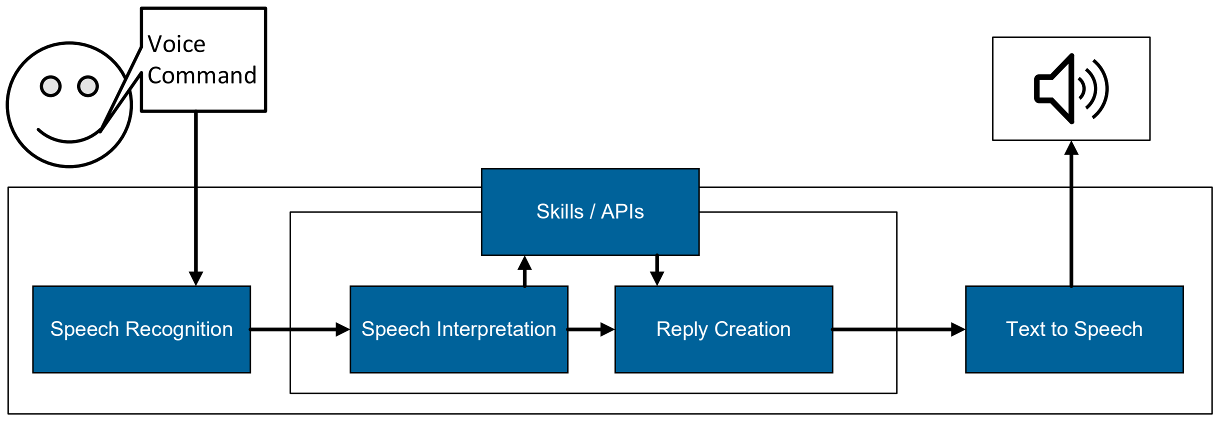


Figure 32: System design of the local voice assistant

A demonstrator using the u.RECS is desired for this speech assistant. The first version is implemented on a Raspberry Pi 4 using 11 Watt and showing a latency of 40 seconds using the Coqui frameworks or 3 seconds using the other two. This high latency is due to the limited performance of the Raspberry and will be improved using more powerful hardware. An implementation using an NVIDIA Xavier NX mounted in the u.RECS will be the next step in combination with securing the recorded voice and encryption methods developed in this project and presented in the next section.

5.1.9 Implementation of a Secure Smart Microphone with Hotword Recognition

As an extension of the integration within a smart home environment, we conceived a wireless smart microphone device as an accessory to the smart mirror demonstrator. The device is based on an Arduino Nano RP2040 Connect [18] microcontroller with a built-in microphone. Spoken keywords are recognized by a small machine-learning model running on the microcontroller. Upon detection, a transcript of the word is sent to a secure service over Wi-Fi. The communication is TLS encrypted and the secure service runs inside a trusted execution environment (TEE) based on Intel Software Guard Extensions (SGX), to ensure the spoken words cannot be spied on. Figure 33 shows an overview of the secure smart microphone architecture.

Currently, the neural network of the smart microphone is trained on a Google spoken word dataset containing 35 words. It can recognize a subset of eight different words: “yes”, “no”, “up”, “down”, “left”, “right”, “on”, and “off”. In a pre-processing step, slices of the recorded audio waveforms are converted from the time domain into the frequency domain using the short-time fourier transform (STFT). This conversion results in spectrograms showing frequency changes over time and are represented as 2D images. This conversion allows the use of established image recognition machine-learning models to classify audio data. A simple model, consisting of just two layers, fits into the memory of the microcontroller, having the drawback of a lower accuracy compared to more advanced voice recognition architectures, but retaining a better latency due to no communication overhead.

Once the hotwords have been identified, the microcontroller establishes a secure connection to an edge computing node using TLS. It dispatches the recognized commands to a program running in a secure enclave, bootstrapped by Intel SGX. An enclave is an isolated area of memory which cannot be subverted by high-privileged software, such as the operating system and the hypervisor. This protection is complemented by a remote attestation mechanism, ensuring that the code executed in the enclave is genuine for the other components. The software running in the enclave receives the voice commands and stores them in an in-memory database. Since software development for TEEs is constraint by many factors (e.g., no POSIX, SDK available for a few programming languages), we leverage one of the state-of-the-art solutions to compile our software into a WebAssembly program and execute it in a trusted runtime [19]. As a result, the edge software source code does not use any proprietary API and can be either compiled to run outside of an enclave as a regular process or shielded by Intel SGX.

In future work, the smart microphone will be extended to send a stream of audio data to the server when certain hotwords are detected. A more advanced speech recognition algorithm can then process the audio stream to generate a transcript of spoken sentences. Moreover, the edge software will evolve into a messages broker to ease the integration of other software and hardware components based on voice recognition. Furthermore, we aim to extend this software to connect to other data sources, such as the images recognized by the camera of the smart mirror. Consequently, this communication bus will handle an important amount of sensitive information, justifying the use of increased safety technology, such as trusted execution environments.

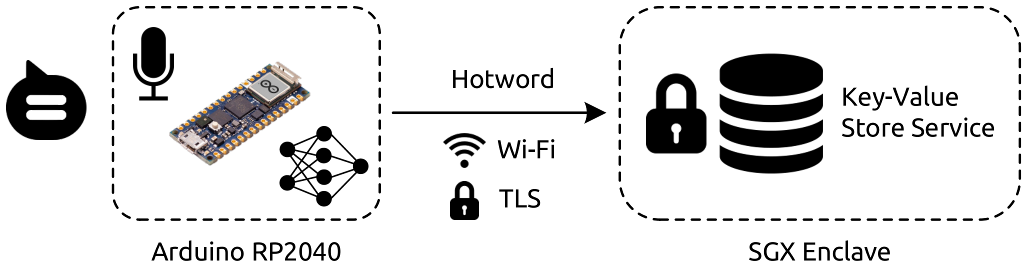


Figure 33: Secure smart microphone architecture overview

5.2 First Benchmarks Results

In the case of the smart mirror demonstrator, the optimisations will be benchmarked by the performance and power consumption of the image processing parts. All detection modules are running simultaneously, as the real-world application would require. The baseline system consists of two NVIDIA AGX Xavier modules on a t.RECS, which originates from the predecessor project LEGaTO. This baseline system delivered a performance of 16 FPS in the object, gesture, and face recognition simultaneously with a power consumption of ~150 Watts. The recent changes with the ROS2 middleware and the restructuring of the face detection have increased the performance to 28 FPS with a power consumption of ~100 Watts. This is an energy efficiency increase of about 2,6x. Running the detections on the NVIDIA AGX Orin shows a performance of 30 FPS with a power consumption of ~55 Watts. The visualisation is left out in this case as some packages are missing, but it shows the performance increase of the NVIDIA AGX Orin compared to the NVIDIA AGX Xavier. An NVIDIA AGX Orin should be sufficient to run the complete smart mirror application. Figure 34 illustrates the energy efficiency increase. A further increase in energy efficiency is still highly desirable and the main goal for this use case.

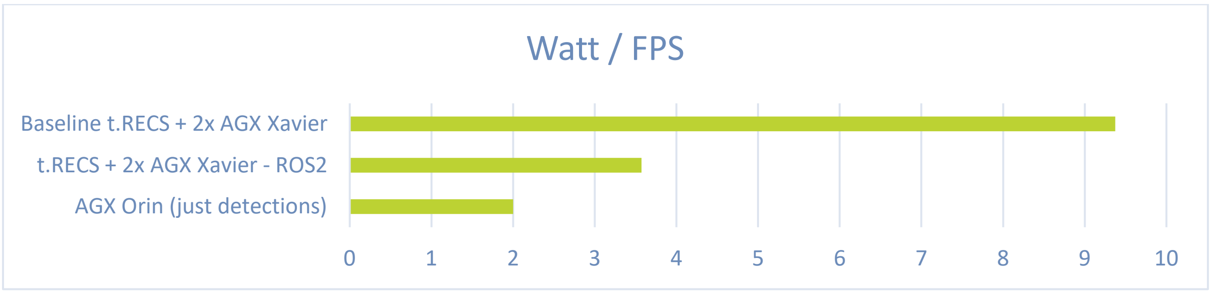


Figure 34: Energy efficiency development of the smart mirror demonstrator

Initially, all new modules will be developed and optimized separately to avoid interference and ensure proper execution. The integration into a new smart mirror demonstrator is planned for the final phase of the project (FPGA utilization, 3D points cloud mirror image).

In the case of the local voice assistant, the first demonstrator consisting of a Raspberry Pi and a microphone is implemented. The benchmark for this use case will be power consumption in relation to latency. The power consumption is around 12 Watt, and the latency for the better sounding version is 40 seconds. Especially for a voice assistant, a long processing time is bothersome but not critical. It should nevertheless be kept below a latency of 5 seconds and will be enhanced in the following steps. Enhancing data privacy and security, the smart microphone demonstrated a possible solution which is also to be integrated.

5.3 First Evaluation of the Key Performance Indicators

The current state of the achieved improvements regarding the KPIs are listed below. Some of them could not yet be evaluated and will be evaluated in the final phase of the project. Integration into the next smart mirror prototype are part of the next steps and optimizations have started but no reliable values have been generated so far.

Performance Metrics:

- **Latency:** The overall latency of the system was decreased, but not accurately measured. The bottleneck for the visualisation is displaying the image in the MagicMirror due to the needed image compressing. The detections are much faster than the currently used JPEG compression for visualization. Therefore, the compression needs to be improved.
- **Achieved performance:** The imaged-based detection nodes for face, gesture and objects are running with a performance of 30 FPS. This is the maximum performance of the camera.

Cost Metrics:

- **Resources:** Can be evaluated as soon as the new hardware combinations are evaluated.
- **Cost:** Can be evaluated at a later state when the new hardware combinations are available for testing and the situation in the global supply chain has normalized.
- **Power/ Energy:** The current smart mirror prototype based on the t.RECS platform is consuming around 100 Watt. The recent development using the newer version of ROS2 on the NVIDIA AGX Orin is showing an improvement to around 60 Watt.

Quality Metrics:

- **Accuracy:** The accuracy for this use case consists of the accuracy of many neural networks and no measurements were yet performed. This is due to the state of the optimizations of the networks which have just started. First results for YoloV4 are evaluated in Deliverable D3.3 [20].

Combined Metrics:

- **(Energy) Efficiency:** The current smart mirror prototype based on the t.RECS platform is showing around 3.57 Watt / FPS. Recent developments using the newer version of ROS2 on the NVIDIA AGX Orin is showing an improvement to around 2 Watt / FPS.
- **Cost Efficiency:** Can be evaluated at a later state when other hardware combinations are available for testing and the situation in the global supply chain has normalized.

5.4 Next Steps

The newly developed modules for the smart mirror will be integrated into the smart mirror in the next step. The performance will be evaluated on the NVIDIA AGX Orin and then on an NVIDIA XAVIER setup. Integrating the optimized version of the gestures and object recognition with the help of the EmbeDL tools or on the FPGA should be easy with the ROS2 middleware. Once stable versions have been implemented, this will be evaluated for performance—subsequently, further evaluations on more hardware configurations of the t.RECS will follow.

With the help of the secure microphone, the first techniques for further securing personal data were demonstrated. This procedure must be adapted and integrated for the developed voice assistant in the next step. Based on NVIDIA Jetson modules or Intel SGX architectures, this voice assistant will then be integrated into the smart mirror.

6 Conclusion

This document is the second of three deliverables about the development of the use cases and the transition from traditional algorithms to a machine learning approach and its optimisation. Therefore, it presents some intermediate results of the use case developments and optimisations, as well as gives an impression of what will be delivered within the last project year.

In the last project months, since the writing of the previous deliverable D7.2, all use cases have achieved to build up a fully functional testbed for a) demonstration purposes to scientific and public audience and b) to general high-quality ML training data. Therefore, the important project milestone 6 "First use cases demos available" is accomplished. Furthermore, all use cases were able to train first neural networks.

The IIoT use case of motor condition classification will continuously improve the whole motor condition monitoring testbed to provide even better training and verification data. Furthermore, the first trained DL models will be enhanced to be more accurate with further training data and generally optimised. A big goal is the integration of the DL models into the testbed for tests, validation and a nice demonstrator with an AR interface.

The next steps of the IIoT arc detection use case will be to enhance the performance as well as the efficiency of the trained DL model. This will be done by enlarging the database with data that is closer to real-world scenarios and improve quality of the data by searching and integrating external data sources of arc data. As for the other use cases, also this use case will be integrated in real-world IIoT networks. The demonstrator runs currently in a local lab environment, but it is planned to include the output of the DL network into the data stream of Siemens' IIoT network for post processing (storing, comparing with other data and so on) to realise a realtime monitoring system.

The automotive AI use case has the goal to integrate all developed components into a car for real-world testing and evaluation. Also, the distributed processing of the captured data will be tested and evaluated in physical scenarios. A major step for this is to build up an own 5G base station and install the project's hardware and software in this edge environment and connect in to the car.

The next steps for the Smart Home use case will be to integrate the newly developed hardware modules into the smart mirror. The overall performance will be evaluated on different hardware setups to find the best suitable hardware. Furthermore, the optimized version of the gestures and object recognition will be integrated and optimized. A next integration step will be to adapt and integrate the secured microphone prototype into the local voice assistant of the smart mirror.

7 References

[1] VEDLIoT project, “D7.2 - First report on use case development and optimisation,” 2022.

[2] ROS project, “ROS - Robot Operating System,” [Online]. Available: https://www.ros.org/. [Accessed 29 04 2022].

[3] VEDLIoT EU project, D2.3 - Pilots/use case specification, 2021.

[4] R. Weiss, L. Ott and U. Boecke, “Energy efficient low-voltage DC-grids for commercial buildings,” *IEEE First International Conference on DC Microgrids (ICDCM),* pp. 154-158, 2015.

[5] B. Wunder, R. Weiss, L. Ott, M. Szpek and U. Boeke, “Energy efficient DC-grids for commercial buildings,” *IEEE 36th International Telecommunications Energy Conference (INTELEC),* pp. 1-8, 2014 .

[6] S. Lu, B. Phung and D. Zhang, “A comprehensive review on DC arc faults and their diagnosis methods in photovoltaic systems,” *Renewable and Sustainable Energy Reviews,* pp. 88-98, Volume 89 2018.

[7] EUROPEAN NEW CAR ASSESSMENT PROGRAMME, “Test protocol - AEB/LSS VRU systems,” 2021.

[8] R. Griessl, “D4.1 First report on cognitive IoT hardware platform and microserver development,” 2022.

[9] Xilinx, “VCK190 Evaluation Board User Guide (UG1366 v1.0),” 2021.

[10] Xilinx, “DPUCVDX8G for Versal ACAPs Product Guide (PG389 v1.1),” 2022.

[11] “github - fast GICP Algorithmen,” [Online]. Available: https://github.com/SMRT-AIST/fast_gicp.

[12] NVIDIA, “Nvidia TensorRT Overview,” [Online]. Available: https://developer.nvidia.com/tensorrt.

[13] coqui, “coqui Text to Speech,” [Online]. Available: https://github.com/coqui-ai/TTS.

[14] alphacephei, “vosk - speech recognition toolkit,” [Online]. Available: https://alphacephei.com/vosk/.

[15] [Online]. Available: https://github.com/IBM/AccDNN.

[16] E. AI, “Spacy website,” [Online]. Available: https://spacy.io/.

[17] “Rasa,” [Online]. Available: https://rasa.com/.

[18] A. Foundation, “Documentation - Nano RP2040 Connect,” [Online]. Available: https://docs.arduino.cc/hardware/nano-rp2040-connect.

[19] J. Ménétrey, M. Pasin, P. Felber and V. Schiavoni, “Twine: An Embedded Trusted Runtime for WebAssembly,” in *ICDE'21: Proceedings of the 37th IEEE International Conference on Data Engineering*, Chania, Greece, 2021.

[20] VEDLIoT EU Project, “Deliverable 3.3 - Evaluation of the DL accelerator designs,” 2022.

[21] A. Jani, “Maxim Showcases Efficient Custom AI,” *Microprocessor Report,* 2021.

[22] L. Gwennap, “XMOS Xcore.ai Adds Vector Unit,” *Microprocessor Report,* 2020.

[23] B. Wheeler, “RISC-V Enables IoT Edge Processor,” *Microprocessor Report,* 2018.

[24] B. Wheeler, “Bitmain SoC Brings AI to the Edge,” *Microprocessor Report,* 2019.

[25] M. Demler, “Syntiant NDP120 Sharpens Its Hearing,” *Microprocessor Report,* 2021.

[26] L. Gwennap, “Intel Gains Myriad Customers,” *Microprocessor Report,* 2018.

[27] M. Demler, “Coherent Logix Configures Edge AI,” *Microprocessor Report,* 2020.

[28] M. Demler, “Hailo Illuminates Low-Power AI Chip,” *Microprocessor Report,* 2019.

[29] M. Demler, “Blaize Ignites Edge-AI Performance,” *Microprocessor Report,* 2020.

[30] M. Demler, “Flex Logix Moves Into Chips,” *Microprocessor Report,* 2019.

[31] N. Developers, “Nvdla primer,” 2021. [Online]. Available: http://nvdla.org/primer.html.

[32] T. Moreau, “A hardware-software blueprint for flexible deep learning specialization,” *arXiv:1807.04188,* 2019.

[33] X. Zhang, “an Automated Tool for Building High-Performance DNN Hardware Accelerators for FPGAs,” 2018.

[34] M. Demler, “Vsora Drives to Deliver Petaflops,” *Microprocessor Report,* 2020.

[35] M. Demler, “Andes plots RISC-V vector heading,” *Microprocessor Report,* 2020.

[36] M. Demler, “Ceva NeuPro accelerates neural nets,” *Microprocessor Report,* 2018.

[37] M. Demler, “Imagination Series4 tiles tensors,” *Microprocessor Report,* 2020.

[38] M. Demler, “Ceva SensPro2 Doubles AI Throughput,” *Microprocessor Report,* 2021.

[39] “DPUCZDX8G for Zynq UltraScale+ MPSoCs,” [Online]. Available: https://www.xilinx.com/support/documentation/ip_documentation/dpu/.

[40] M. Demler, “Think Silicon Spins AI Accelerator,” Microprocessor Report, 2020. [Online].

[41] A. Jani, “SiFive VIU7-256 Takes Vector Lead,” *Microprocessor Report.*

[42] A. Jani, “SiFive Brings Vectors to S-Series,” *Microprocessor Report,* 2021.

[43] M. Demler, “Cortex-M55 Supports Tiny-AI Ethos,” *Microprocessor Report,* 2020.

[44] L. Gwennap, “Cortex-A55 Improves Memory,” *Microprocessor Report,* 2017.

[45] L. Gwennap, “Cortex-A75 Has DynamIQ Debut,” *Microprocessor Report* , 2018.

[46] L. Gwennap, “Arm Dot Products Accelerate CNNs,” *Microprocessor Report* , 2018.

[47] J. Roesch. [Online]. Available: URL: http://dx.doi.org/10.1145/3211346.3211348.

[48] "Apache Software Foundation," [Online]. Available: https://tvm.apache.org/docs/.

[49] G. Developers, "FlatBuffers Documentation," [Online]. Available: https://google.github.io/flatbuffers/index.html.

[50] G. Developers, "TensorFlow Lite Documentation," [Online]. Available: https://www.tensorflow.org/lite/guide?hl=en.

[51] A. Demidovskij, "learning workbench: comprehensive analysis and tuning of neural networks inference.," 2019.

[52] W.-F. Lin, " Onnc: a compilation framework connecting onnx to roprietary deep learning accelerators.," in *International Conference on Artificial Intelligence Circuits and Systems* , 2019.

[53] W.-F. Lin, "Onnc-based software development platform for configurable nvdla designs," in *International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*, 2019.

[54] N. Rotem, "Graph Lowering Compiler Techniques for Neural Networks," 2019.

[55] L. Gwennap, "Kneron Delivers Efficient AI," *Microprocessor Report,* 2020.

[56] L. Gwennap, "Kneron KL720 Boosts Efficiency," *Microprocessor Report,* 2020.

[57] L. Gwennap, "GreenWaves GAP9 Goes Faster," *Microprocessor Report,* 2020.

[58] L. Gwennap, "Kendryte Embeds AI for Surveillance," *Microprocessor Report,* 2019.

[59] M. Demler, "Syntiant NDP120 Sharpens Its Hearing," *Microprocessor Report,* 2021.

[60] L. Gwennap, "LeapMind jumps on binary networks," *Microprocessor Report,* 2020.

[61] Nvidia, "Datasheet of Nvidia Jetson Xavier NX," [Online]. Available: https://developer.download.nvidia.com/assets/embedded/secure/jetson/Xavier%20NX/DG-09693-001_v1.5.pdf. [Accessed 19 01 2022].

[62] SGeT - Standardization Group for Embedded Technologies e.V., "SMARC Standard," [Online]. Available: https://sget.org/standards/smarc/. [Accessed 19 01 2022].

[63] Espressif, "Espressif IoT Development Framework," [Online]. Available: https://github.com/espressif/esp-idf. [Accessed 19 01 2022].

[64] PlatformIO labs, "Platform IO Website," [Online]. Available: https://platformio.org/. [Accessed 19 01 2022].

[65] VEDLIoT EU Project, "Deliverable 2.1 - Intermediate report on the methods for requirement, specification, performance metrics and verification of context limited AI processing systems," 2021.

[66] Ioffe, S., & Szegedy, C., "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *International conference on machine learning,* pp. 448-456, June 2015.

[67] LEGaTO project, “LEGaTO project,” [Online]. Available: https://legato-project.eu/. [Accessed 29 04 2022].

[68] M. Demler, “Ethos-N78 Boosts AI Efficiency,” *Microprocessor Report,* 2020.

[69] O. S. R. Foundation, “ROS - Robot Operating System,” [Online]. Available: https://ros.org/.

[70] coqui, “coqui,” coqui, [Online]. Available: https://coqui.ai/.

[71] coqui, “coqui Speech to Text,” [Online]. Available: https://github.com/coqui-ai/STT.

[72] “pyttsx3,” [Online]. Available: https://github.com/nateshmbhat/pyttsx3.

8 List of Figures