Horizon 2020 Program (2014-2020)

A computing toolkit for building efficient autonomous applications leveraging humanistic intelligence (TEACHING)

D5.3: AI models training Dataset[†]

Contractual Date of Delivery	*30/04/2022*
Actual Date of Delivery	*06/06/2022*
Deliverable Security Class	*Public*
Editor	*Omar VELEDAR (AVL)*
Contributors	*Konstantinos Tserpes (HUA), Iraklis Varlamis (HUA), Massimo Coppola (CNR), Georg Macher (TUG), Juergen Dobaj (TUG), Christoph Kreuzberger (AVL), Markus Posch (AVL), Herbert Danzinger (AVL), Rosaria Potenza (M), Maria Carmela De Gennaro (M), Calogero Calandra (M), Lorenzo Giraudi (I&M), Roberta Peroglio (I&M), Sylvain Girbal (TRT), Siranush Akarmazyan (ITML), Claudio Gallicchio (UNIPI), Vincenzo Lomonaco (UNIPI), Antonio Carta (UNIPI), Valerio De Caro (UNIPI), Alberto Gotta (CNR)*
Quality Assurance	*Davide Bacciu (UNIPI)*

[†] *The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871385.*

The TEACHING Consortium

University of Pisa (UNIPI)	Coordinator	Italy
Harokopio University of Athens (HUA)	Principal Contractor	Greece
Consiglio Nazionale delle Ricerche (CNR)	Principal Contractor	Italy
Graz University of Technology (TUG)	Principal Contractor	Austria
AVL List GmbH (AVL)	Principal Contractor	Austria
Marelli Europe S.p.A. (M)	Principal Contractor	Italy
Ideas & Motion (I&M)	Principal Contractor	Italy
Thales Research & Technology (TRT)	Principal Contractor	France
Information Technology for Market Leadership (ITML)	Principal Contractor	Greece
Infineon Technologies AG (IFAG)	Principal Contractor	Germany

Document Revisions & Quality Assurance

Internal Reviewers

- *Reviewer: Georg Macher, (TUG)*

Revisions

Version	Date	By	Overview
1.0	06/06/2022	Bacciu (UNIPI)	QC version
0.5	03/06/2022	Macher (TUG)	Review version
0.4	02/06/2022	Veledar (AVL)	Edited version
0.3	27/05/2022	All contributors	Draft
0.2	24/05/2022	All contributors	Partner Contributions
0.1	04/04/2022	All contributors	Revised structure
0.0	01/10/2021	Veledar (AVL)	ToC

Table of Contents

LIST OF TABLES 5
LIST OF FIGURES 6
LIST OF ABBREVIATIONS........ 7
EXECUTIVE SUMMARY........ 9
1 INTRODUCTION 10
2 THE USE CASES – AN OVERVIEW AND METHOD 12
2.1 THE AVIONICS USE CASE 12
2.1.1 Use-case description 12
2.1.2 Objectives of the avionics use-case 14
2.1.3 Tailoring of the avionics use-case for the selected hardware platform and general-purpose operating system 16
2.1.4 Stress measurement concept & methodology to collect training datasets........ 18
2.1.5 Stress measurement implementation on the iMx8 board 19
2.1.6 Description of the avionics training datasets passed to WP4 to perform anomaly detection 21
2.1.7 Statistical Analysis of collected datasets 23
2.2 THE AUTOMOTIVE USE CASE 26
2.2.1 Technology tailoring for the automotive use case 26
2.2.2 Driving simulator setup and procedure 28
2.2.3 Driving scenarios 29
2.2.4 Simulator study data 33
2.3 EXTERNAL DATA (OPEN ACCESS)........ 40
2.3.1 Datasets for Human state and activity monitoring........ 40
2.3.2 Datasets for anomaly detection in aerospace applications 41
2.3.3 Datasets for anomaly detection in cybersecurity applications........ 42
2.3.4 Datasets for human-centric personalization 42
3 CONTRIBUTION TO MILESTONES 43
3.1 MILESTONE 3: DATASET FOR AI MODELS TRAINING AVAILABLE........ 43
3.2 MILESTONE 4: TEACHING CORE TECHNOLOGY INTEGRATED IN USE CASES SETUP 43
3.2.1 High-Performance Computing and Communication Infrastructure (HPC2I) as integrated within the use cases. 44
3.2.2 AI-as-a-Service core technology integrated into the use cases setup........ 46
BASE PYTHON CLASSES 46
CURRENTLY AVAILABLE NODES........ 46
HOW TO USE NODES 47
HOW TO IMPLEMENT CUSTOM NODES 47
DEBUG/TEST MODE 47
4 DISCUSSION AND CONCLUSION 48
5 REFERENCES 49

List of Tables

Table 1 Driving scenario list 29

List of Figures

Figure 1 – The avionic use case composed of the CBB (in blue) and the FMS (in yellow) 12
Figure 2 - Flight Management System: functional overview 13
Figure 3 - FMS Display 14
Figure 4 - iMx8 socket on a module 15
Figure 5 - Convergence of safety and security monitoring challenges 16
Figure 6 - Avionics use-case: a functional view 17
Figure 7 - Mapping of the avionics use-case to the iMx8 board 17
Figure 8 - Generation of a cyberprint representative of the nominal/normal behaviour of the critical software on the hardware 18
Figure 9 – Multi-layer / multi-source monitoring 19
Figure 10 - First lines of a raw.csv file 22
Figure 11 - Retrieving data semantic information from generated traces 23
Figure 12 - Respective instruction profile of the FMS application (on the left) and the FFT application (on the right) 24
Figure 13 - Instruction profile of a cache-miss DoS attack on the L2 cache 24
Figure 14 - Runtime profiling of the FMS application without anomalies 25
Figure 15 - Runtime profiling of the FMS application with anomalies 25
Figure 16 - Temperature (left) and power (right) profiling of the FMS application 26
Figure 17 EECACC - ECU vs FMU comparison 27
Figure 18 EECACC input - output variables 27
Figure 19 EECACC using LTNNG tracing toolkit 28
Figure 20 Scenarios 1 and 2 - sharp cut-in 30
Figure 21 Scenario 3 - joining a motorway with a formed platoon 31
Figure 22 Scenario 3 - platoon cut out 31
Figure 23 Scenario 3 vs Scenario 4 - Adverse Weather Conditions 32
Figure 24 Scenario 5 - phantom breaking in city driving conditions 32
Figure 25 Scenario Data 34
Figure 26 Raw ECG Sensor Data 34
Figure 27 Raw GSR Sensor Data 34
Figure 28 Scenario 1, participant 2111 35
Figure 29 Scenario 1, Participant 2180 36
Figure 30 Scenario 2, Participant 2111 36
Figure 31 Scenario 3, Participant 2022 37
Figure 32 Scenario 4, Participant 2180 37
Figure 33 Scenario 5, Participant 2111 38
Figure 34 Scenario 5, Participant 2111 - Zoomed-in 38
Figure 35 Scenario 6, Participant 2111 39
Figure 36 Scenario 6, Participant 2111 - Data Outage 39
Figure 37: Main logical components design of the TEACHING platform 43
Figure 38: TEACHING demo Application 44

List of Abbreviations

AD	Autonomous Driving
AI	Artificial Intelligence
BVP	Blood Volume Pulse
CBB	Cyber BlackBox
CPS	Cyber-Physical System
DL	Deep Learning
DTM	Dynamic Thermal Management
DVFS	Dynamic Voltage and Frequency Scaling
Dx.y	Project deliverable (x = WP number, y = deliverable order)
EC	European Commission
ECG	Electrocardiogram
EDA	Electrodermal Activity
EECACC	Energy-Efficient Cooperative Adaptive Cruise Control
ESN	Echo State Network
FFT	Fast Fourier Transform
FL	Federated Learning
FMS	Flight Management System
FMU	Functional Mock-Up
GDPR	General Data Protection Regulation
GSR	Galvanic Skin Response
HHAR	Heterogeneity Human Activity Recognition
HIDS	Host-based Intrusion Detection Systems
HPC2I	High-Performance Computing and Communication Infrastructure
HR	Heart Rate
HRV	Heart Rate Variability
HUMS	Health Usage and Monitoring System
MAS	Model Aggregation Service
MEC	Mobile Edge Computing
METrICS	Measurement Environment for Multi-Core Time-Critical Systems
MHU	Message Handling Unit
ML	Machine Learning
MSL	Mars Science Laboratory
MSx	Milestone x (x = milestone ID)

MTS	Model Transfer Service
PMC	Performance Monitor Counter
RL	Reinforcement Learning
RTOS	Real-Time Operating System
SCP	System Control Processor
SMAP	Soil Moisture Active Passive
SoC	System on Chip
SoM	Socket on Module
THErM	Temperature Health & Energy Monitoring framework
WCET	Worst-Case Execution Time
WP	Work Package

Executive Summary

This document describes the data collection activities and the acquired data from the TEACHING project’s use cases. One use case is focusing on the avionics domain, which is likely to implement TEACHING outcomes, the industrial exploitation of which is likely to be delayed due to the stringent safety-related restrictions. The other use case is aimed at earlier exploitation of TEACHING assets in one of the current key automotive trends, autonomous driving. In both cases, the focus of this work is on the collection and management of adequate real-world data, that is to be exploited for the development and later validation of the Artificial Intelligence (AI) and learning models as a part of the TEACHING AIaaS system. This document also describes the tailoring performed to align the TEACHING AI activities with the use case, especially in terms of generating the appropriate AI models through the utilisation of the collected data. The models will be incrementally refined and tailored to the necessity of both use cases.

The avionic use-case includes safety and security-critical applications, the Flight Management System (FMS) and the Cyber-BlackBox (CBB). The work advocates acceptance of AI in avionics, targets benchmarking of the automotive platform for adoption into avionics and monitors requirements and performance costs of future safety-critical systems.

The automotive use case relies on a driving simulator study and a set of TEACHING-specific driving scenarios to acquire relevant subjective and objective driver data to quantify, understand and predict human behaviour in different driving scenarios. The focus goes beyond the safety-critical scenarios, as it also touches upon the human response to driving events in general, as a means of determining the human state at any given time and evaluating their ability to interact with the machine if and when needed.

1 Introduction

Considering the expectations and the consequent need to integrate the human-centric aspect into the progressive automation of CPSoS in a holistic manner, it is crucial to integrate the human comfort and distress throughout system operation. The realisation of such a holistic environment that enables the cooperation between the human and the cyber-physical entities relies on AI to handle the inherently dynamic, connected and interacting operation. If successful, AI will prove to be the key component of human-to-machine cooperation. To achieve such a prominent function, AI components and applications must be enabled to abstract from the details of the low-level sensor data. That implies access to relevant data to drive the advanced analytics, adaptivity and prediction mechanisms with the potential to be integrated into the edge resources of the future.

To that aim, as envisaged by TEACHING overall objective 7, "*Evaluate the computing, adaptation and dependability functionalities of the CPSoS on safety-critical industrial applications through autonomous driving and aviation demonstration environments*", the industry-relevant demonstrators must provide training data sets for development, refinement and assessment of proposed algorithms. Hence, this document describes the creation and format of the datasets for AI training, which were generated through targeted and tailored use case designs and initial implementation.

The avionics use case focuses on data that evaluates software behaviour on the hardware SoC (e.g., temperature or power consumption information). The aggregated data is used to deduce the traces of hardware events that are used for AI model training (WP4). The aim is to perform offline learning capable of delivering online detection of anomalies with respect to the expected application behaviour. The detection is to guarantee the dependable operation of autonomous in-flight plane guidance through the introduction of AI into a domain that resists such algorithms due to the need for certification of autonomous piloting and the associated industrial processes. The application is also to be visualised using a display GUI for the FMS that runs on TEACHING-specific hardware.

On the other hand, the automotive use case targets the usage of the same TEACHING platform, but in an environment that improves driving automation through the integration of vehicle occupants into the control loop. While adequate human-machine interaction is the prerequisite for intuitive usage and acceptance of driving automation, it is also a viable channel for obtaining real-time user feedback for the optimisation of vehicle controls. The feedback enables AI training to understand user inner state and reactions and inform vehicle controls of user expectations within the legal road limitation. In addition, it also tracks user satisfaction with and trust in the system, hence improving the mobility solutions and increasing road safety. Trust and acceptance are potentially the crucial parameters for determining the success of autonomous driving deployment in wider society. Hence, this data taking campaign seeks to determine the appropriate and measurable parameters to be able to quantify trust and acceptance in a physically safe environment using dependable methods. It also delivers data for the AI training activities (WP4), just as is the case with the avionics use case. The data acquisition campaign is based on a recent TEACHING-specific study with a set of tailor-made driving scenarios to gather user subjective (questionnaire) and objective (psychophysiological) feedback.

The rest of this document describes the activities related to the data taking for AI training. To that aim, the core activities are described per use case in chapter 2 and are placed in the context of the use cases. Chapter 3 maps the outcomes of those activities to the overall TEACHING milestones 3 (*Dataset for AI models training available*) and 4 (*Teaching core technology integrated in use cases setup*), which is also followed by a discussion in chapter 4.

2 The Use Cases – an overview and method

This section provides an overview of both use cases and recaps the objectives of each use case within the scope of the TEACHING project. It also describes the tailoring of each use-case to fit with TEACHING AI activities, and details how the input dataset for the AI models of WP4 are collected.

2.1 The Avionics Use Case

2.1.1 Use-case description

The avionic use-case, already presented in Deliverable D5.1 and D5.2, is composed of a safety and security-critical application, the **FMS** and a HIDS / HUMS monitoring system, the **CBB**, that is to ensure the correct behaviour of the safety-critical software on the target embedded hardware board, as depicted in Figure 1.

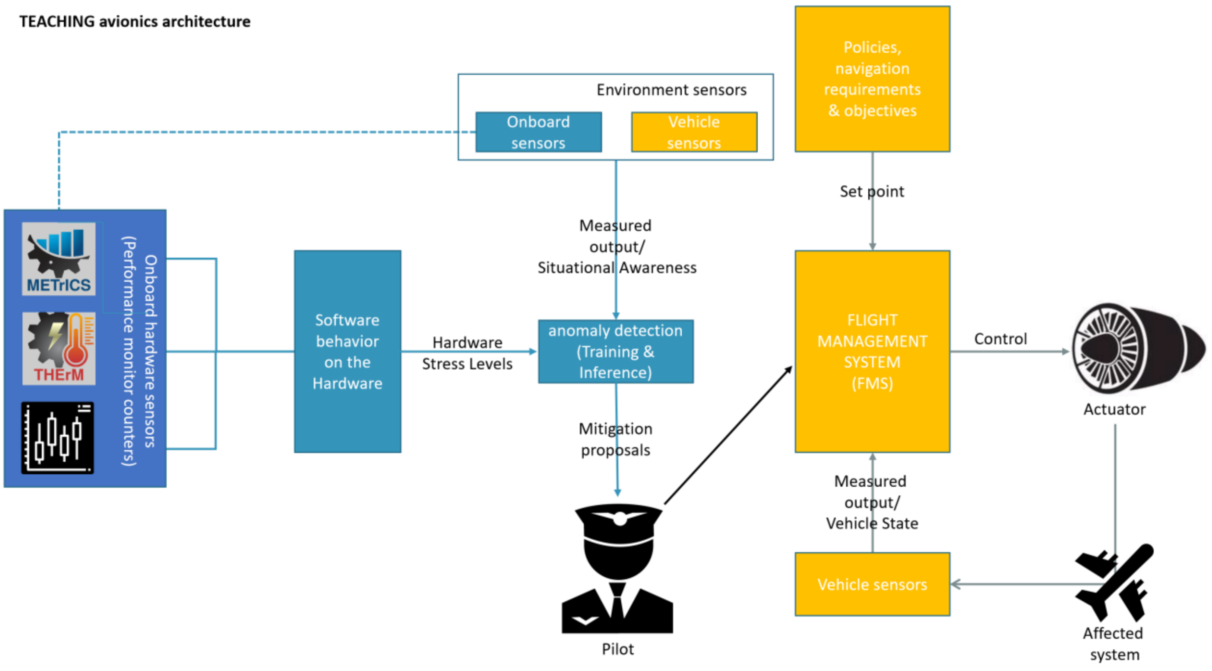

TEACHING avionics architecture
Environment sensors
Onboard sensors
Vehicle sensors
Policies, navigation requirements & objectives
Set point
Measured output/ Situational Awareness
METrICS
THErM
Onboard hardware sensors (Performance monitor counters)
Software behavior on the Hardware
Hardware Stress Levels
anomaly detection (Training & Inference)
FLIGHT MANAGEMENT SYSTEM (FMS)
Control
Actuator
Mitigation proposals
Measured output/ Vehicle State
Vehicle sensors
Pilot
Affected system

Figure 1 – The avionic use case composed of the CBB (in blue) and the FMS (in yellow)

The CBB collects information about the software behaviour on the hardware SoC thanks to dedicated **probes** detailed in Section 3 of Deliverable D5.2, such as METrICS [1], relying on performance monitoring counters and THErM relying on SoC-level specific probes from the SCP coprocessor to collect temperature or power consumption information.

This collected information is then combined in an **aggregation** phase that generates traces of hardware events that will serve as learning material for the AI models of WP4, detecting anomalies with respect to the expected application behaviour. While this learning phase occurs offline, the inference part is expected to be performed online, on the embedded device.

The critical **FMS application** includes time-critical tasks that are regrouped in task groups (Sensors, Localisation, Flightplan, Trajectory, Nearest Airports and Guidance) as presented in Figure 2, that allow autonomous in-flight guidance of the plane, following a pre-set Flightplan from the take-off airport to the landing airport.

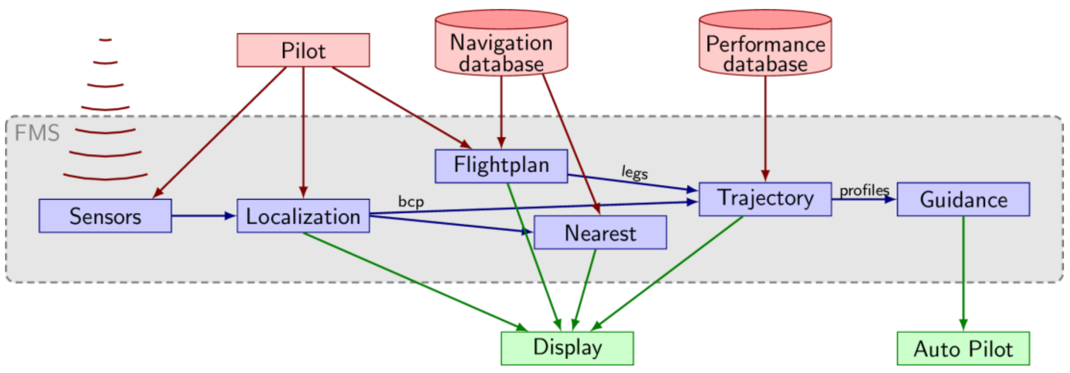


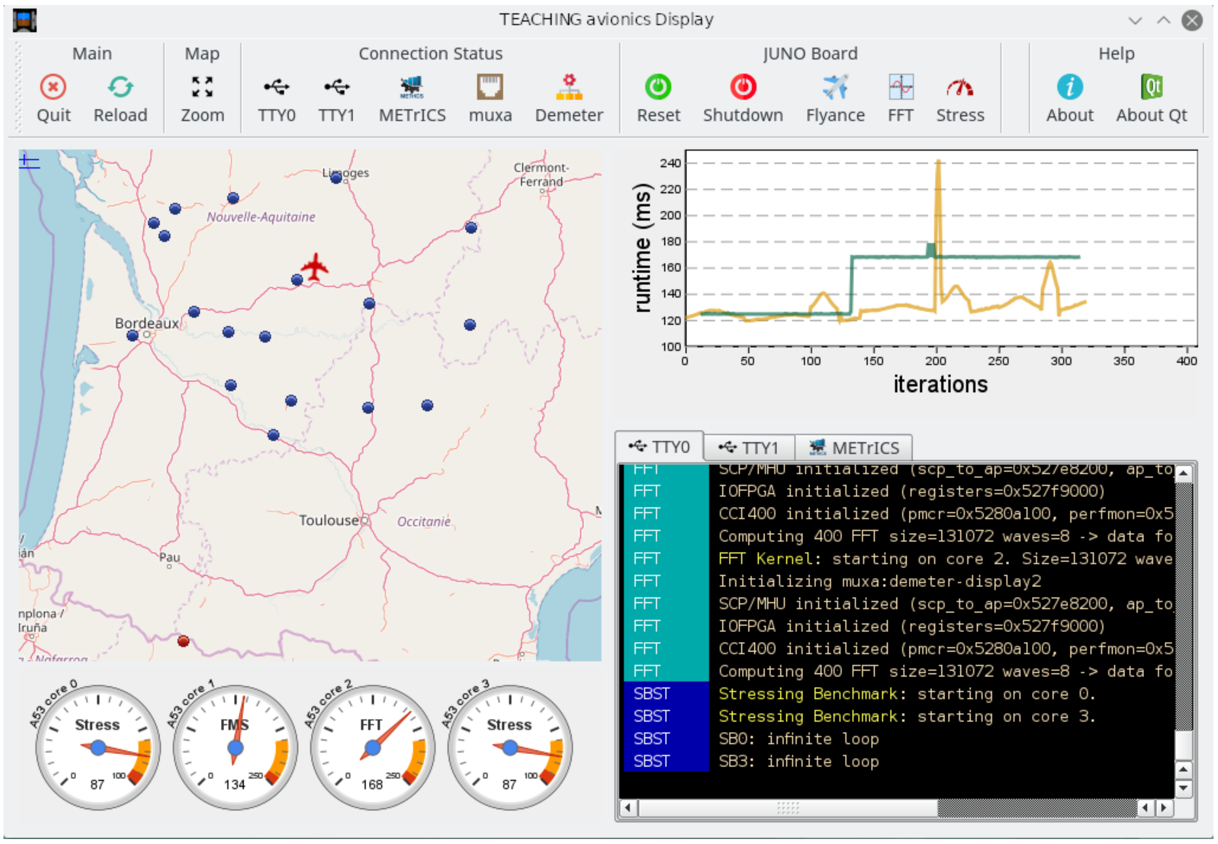
Figure 2 - Flight Management System: functional overview

The tasks from the FMS application have stringent real-time requirements already detailed in Section 2.1 of Deliverable D5.2, most of them being periodic, meaning they do not start when their input data is ready as with regular dataflow applications, but are activated by the clock, even if the latest set of input data is not ready yet. Such hard-real time behaviour should not be altered by the HUMS / HIDS system ensuring the correct behaviour of the FMS.

The monitoring **CBB application** has to ensure the correct behaviour of the autonomous piloting software (the FMS) on the hardware target, an iMx8 board provided by I&M. The idea behind the CBB is to learn the normal/nominal behaviour or the critical software with regards to hardware resource usage, to later be able to detect safety failures or security attacks as deviation from this expected behaviour.

This monitoring activity, relying on an AI-based algorithm, is a way to introduce AI in avionics, without impacting the ability to certify autonomous piloting and the associated industrial processes, focusing only on monitoring, and proposing mitigation solutions to the pilot, as introduced in Figure 1. As a consequence, the CBB application is responsible for collecting traces of software-induced hardware events that will serve as training datasets for the AI models.

Finally, the **display application** will provide both a GUI for the FMS, as well as TEACHING related KPI, such as the ability to detect safety hazards or security issues including specific attacks. The current pre-release of the display application runs on an external PC hardware. The final version will run onto the embedded system, on a separated Ultrascale+ board provided by I&M. A screenshot of the current version appears in Figure 3.



```
FFT   SCP/MHU initialized (scp_to_ap=0x527e8200, ap_to
FFT   IOFPGA initialized (registers=0x527f9000)
FFT   CCI400 initialized (pmcr=0x5280a100, perfmon=0x5
FFT   Computing 400 FFT size=131072 waves=8 -> data fo
FFT   FFT Kernel: starting on core 2. Size=131072 wave
FFT   Initializing muxa:demeter-display2
FFT   SCP/MHU initialized (scp_to_ap=0x527e8200, ap_to
FFT   IOFPGA initialized (registers=0x527f9000)
FFT   CCI400 initialized (pmcr=0x5280a100, perfmon=0x5
FFT   Computing 400 FFT size=131072 waves=8 -> data fo
SBST  Stressing Benchmark: starting on core 0.
SBST  Stressing Benchmark: starting on core 3.
SBST  SB0: infinite loop
SBST  SB3: infinite loop
```

Figure 3 - FMS Display

2.1.2 Objectives of the avionics use-case

The objective of the avionics use case is threefold. **First**, it advocates the introduction of AI in avionics. Autonomous piloting was introduced to the avionics domain decades ago to reduce the piloting crew size, optimise fuel consumption and reduce safety hazards. The avionics sector, which involves stringent safety processes, is therefore very reluctant to introduce AI to such systems, as it is not yet part of the industrial process, and is facing a large set of challenges concerning explainability and reproducibility. As a consequence, we foresee the introduction of AI in autonomous piloting for commercial flights to happen once autonomous driving has become the DeFacto standard. However, avionics is also facing safety and security challenges, especially with the SESAR joint undertaking European partnership that aims at densification of the avionics traffic, with digital and connected aircraft.

The common avionics practice of dealing with safety hazards is to rely on Heath Usage Monitoring Systems (HUMS). The new security challenges are beginning to be dealt with by Host Intrusion Detection Systems (HIDS). Both of those systems rely on monitoring techniques for the detection phase of the “Detect, Respond, Prevent, Recover” process. Such monitoring systems present a potential entry point for AI systems. The monitoring function is usually DAL-D or DAL-C, running at a lower criticality level than the autonomous piloting. It involves large traces of events that need to be classified to detect dreaded events.

Section 2.1.3 describes the training dataset collection process we use to collect traces to be used as input by the AI models.

The **second objective** is to consider the usability of an automotive platform for avionics products. Avionics products usually rely on dedicated hardware following certification standards [2] running dedicated software with their own certification standards [3], [4] scheduled through a specific real-time operating system (RTOS) such as MACS2 or PikeOS. The selected hardware platform to run the critical software is provided by I&M and based on the iMx8 Socket on Module (SoM) appearing in Figure 4.

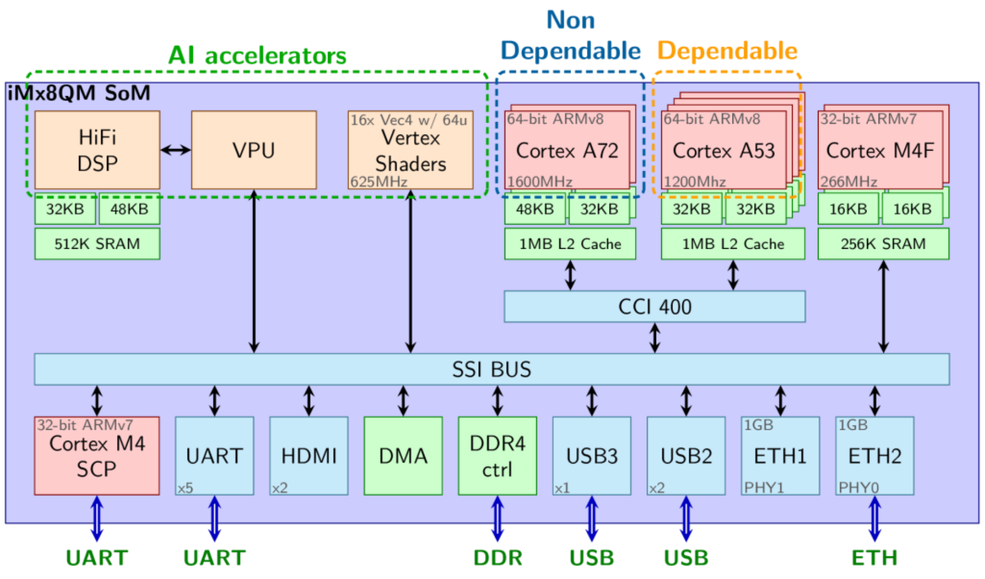


Figure 4 - iMx8 socket on a module

The module integrates Cortex A53 in-order core that is expected to fits the dependability requirements of hard-realtime applications; High-performance Cortex A72 cores that have the potential to run the monitoring services and the inference/anomaly detection processes also possibly exploiting dedicated GPGPU or AI-oriented accelerators.

The hardware platform also runs a general-purpose Linux operating system and not a dedicated RTOS. The Linux kernel has been patched with the LinuxRT (real-time Linux) extension. However, LinuxRT is not real-time as in “real-time operating systems”. RTOS usually rely on static scheduling, effectively controlling and minimizing process migration and context switches. On the contrary, LinuxRT is maximizing context switches to make sure that real-time interrupts/exceptions can always take over less critical processes, and is based on dynamic priority scheduling. As a consequence, the fitness of such a hardware platform coupled with a very dynamic operating system versus the avionic requirements is an interesting alternative to study, concerning hard real-time constraints.

The **third objective** is related to the monitoring requirements of future HUMS and HIDS systems and their associated performance costs. Figure 5 foresees the future trends in safety-critical and security-critical systems with regards to monitoring, presenting the usual V-shaped system development life-cycle.

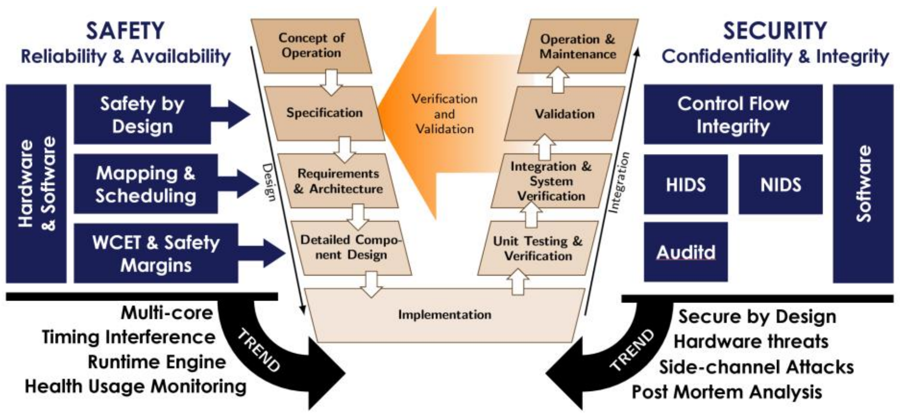


Figure 5 - Convergence of safety and security monitoring challenges

Regulation standards [2], [3], [4] led the safety-critical industry to focus mainly on the design phases [5] phases to ensure reliability and availability by design [6], both at the software and the hardware levels. Security, on the other hand, has typically been provided as specialized software executed during operation, to overcome critical application weaknesses, especially software vulnerabilities with intrusion detection systems [7]. The introduction of multi-core architecture in safe or secure systems already led to a couple of common solutions like memory-space partitioning [8]. It also leads to some converging trends between safety and security practices appearing in Figure 5.

On the safety side, the timing interference problem on time-critical systems for instance has led to a set of control-based or regulation-based solutions [9], [10] solving the issue during the operational phase. On the security side, secure-by-design [11] has become a hot topic with a particular focus on the design phases of the V-shaped model. Also, the hardware should rather not be considered as reliable anymore, but as a potential source for side-channel attacks instead [11], [12]. A common trend for both safety and security-critical systems is therefore to focus on all phases of the life-cycle from conception to operation. Furthermore, the foreseen convergence of safety-critical and security-critical systems would benefit from the gathering of monitoring resources for HUMS and HIDS systems.

2.1.3 Tailoring of the avionics use-case for the selected hardware platform and general-purpose operating system

The avionics use-case components are presented in Figure 6. Beyond the critical FMS running on the iMx8 platform and its associated display aiming at running on the Ultrascale+ platform (but now running on an external PC host), METrICS is also running on the iMx8 platform in the form of a Linux kernel driver and a METrICS collector process.

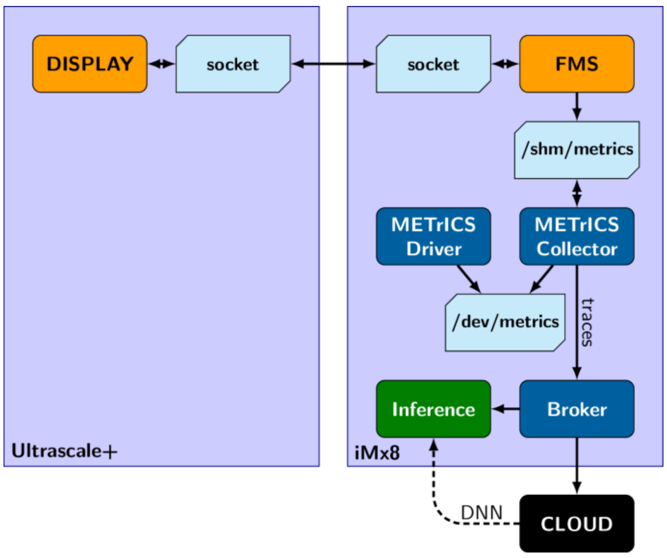


Figure 6 - Avionics use-case: a functional view

The METrICS driver is necessary to perform the configuration of the Performance Monitor Counters (PMCs) that requires privilege kernel mode. The driver is setting up the /dev/metrics device driver upon which CTLIO calls allow to trigger the driver services, such as configuring a counter to count a specific hardware event, starting or stopping the counters, and making the counter being available in user-mode.

The metrics collector process uses these services to set up a particular counter configuration and prepares a shared memory together with the associated /shm/metrics file handle to access such inter-process memory. Such memory is used during the execution of the critical application to store the collected metrics before they get dumped as a set of CSV files as described in Section 2.1.6.

Figure 7 presents the mapping of the use-case applicative processes to the target iMx8 board:

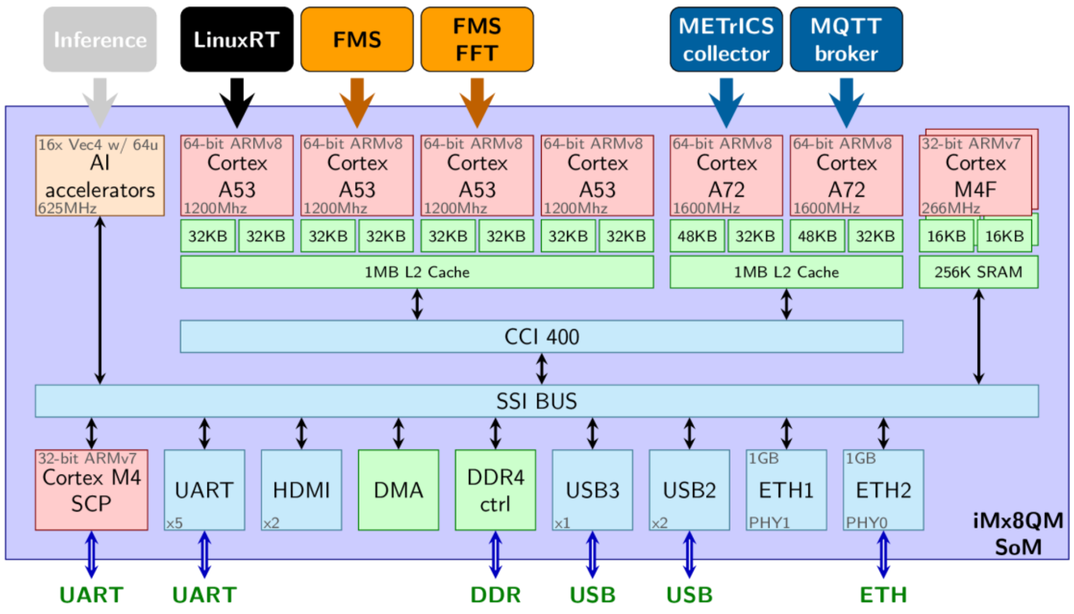


Figure 7 - Mapping of the avionics use-case to the iMx8 board

Cores 0 to3 are the dependable Cortex A53 cores, while cores 5 to 6 are the high-performance Cortex A72 cores. As with every Linux, LinuxRT is mostly running on the first core, while being allowed to use the other cores. We explicitly prevented, as a kernel boot option (See Section 2.1.1 of Deliverable D5.2) the LinuxRT scheduler from using cores 1 and 2 that are dedicated to the time-critical applications. The critical FMS application is forced by affinity to run standalone on core 1, while spawning an FFT thread to run standalone on core 2 to filter out avionics' sensors data.

The METrICS collector process, which specific tailoring for the iMx8 board is presented in Section 4.1.3 of Deliverable D5.2, and the new MQTT broker are respectively running on cores 4 and 5, the high-performance cores. Later, the inference will also run either on these high-performance cores, or on the AI accelerators of the board.

2.1.4 Stress measurement concept & methodology to collect training datasets

As already stated, the purpose of the CBB is twofold: 1) learn the expected normal and nominal behaviour of the critical application, and 2) detect safety failures or security attacks as deviation from this expected behaviour.

The expected behaviour of the software on the hardware is collected through METrICS and the performance monitor counters capturing occurrences of hardware events among hundreds of available events. However, on the IMx8 board, only 6 counters are available per core, only allowing us to monitor 6 hardware events at a given time. As a consequence, capturing the overall expected behaviour requires us to perform many full runs of the critical FMS application, following the process described in Figure 8.

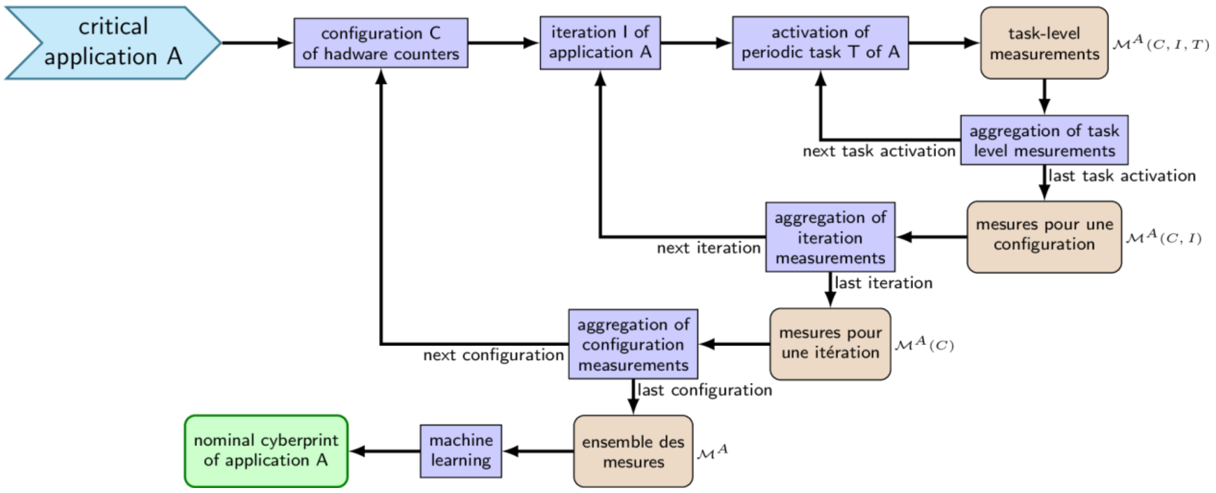

critical application A
configuration C of hadware counters
iteration I of application A
activation of periodic task T of A
task-level measurements
$\mathcal{M}^A(C, I, T)$
aggregation of task level mesurements
next task activation
last task activation
mesures pour une configuration
$\mathcal{M}^A(C, I)$
aggregation of iteration measurements
next iteration
last iteration
mesures pour une itération
$\mathcal{M}^A(C)$
aggregation of configuration measurements
next configuration
last configuration
ensemble des mesures
$\mathcal{M}^A$
machine learning
nominal cyberprint of application A

Figure 8 - Generation of a cyberprint representative of the nominal/normal behaviour of the critical software on the hardware

Beyond iterating on the available counter configuration to cover all the available hardware events, the critical application is fully run several times (application iterations) for statistical coverage purposes, and METrICS measurements are performed at the periodic task level where the real-time constraints are located.

Along the process, aggregating the measurements corresponds to aggregating traces of monitored hardware events, which will serve as an input dataset for the AI learning model capturing the expected normal/nominal behaviour in the form of a cyberprint represented by a deep neural network.

The implementation of the campaign collection and the associated dataset trace formats are presented respectively in Section 2.1.5 and Section 2.1.6.

Beyond METrICS with the timing and performance counter information, we expect after the METrICS project to make the CBB gather monitoring information from different levels and different sources to communalize the monitoring features, reducing the overall performance impact of monitoring, as presented in the [13] survey and position paper; and illustrated by Figure 9.

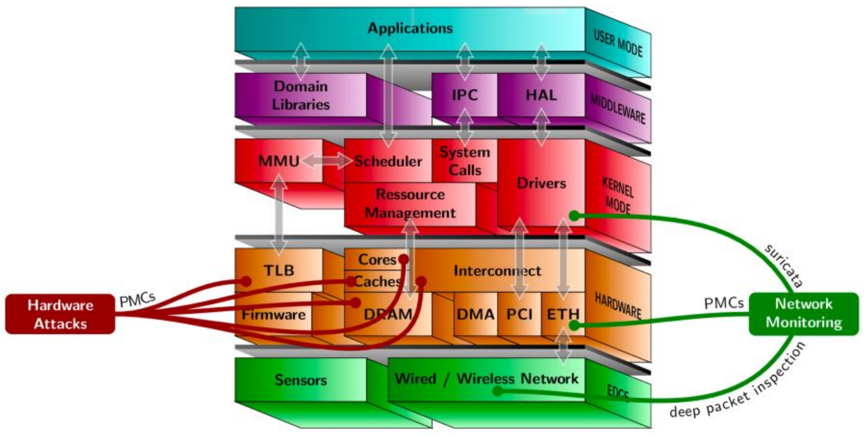


Figure 9 – Multi-layer / multi-source monitoring

Figure 9 shows the different layers composing the technology stack. This includes the user-mode applications, the domain-specific middleware, the operating system layer, the embedded hardware SoC and the physical communication layer. The intra-layer and inter-layer arrows show regular component interactions, for instance, the operating system scheduler sets which application should be running, sets up the proper MMU entries for logical to physical address translation and flushes the hardware TLB caching the MMU.

Most monitoring techniques in HUMS or HIDS are already multi-level implying several layers. METrICS, for instance, collects information from the hardware layer, inserting probes in the application layer and capturing context switches from Kernel / OS layer. In fact, each layer has only limited information and lacks the semantics of the other layers: the hardware layer has efficient and immediate pipeline-related information from the Performance Monitor Counters (PMCs) such as branches or even cache misses but does not know which application, task or thread is currently running. This information is only available either directly from the application layer or from the scheduler of the operating system layer. Therefore gathering information from multiple layers is necessary to make monitoring efficient. Pushing this concept further, we might also benefit from gathering information from multiple sources, communalising the monitoring information from different subsystems. It will reduce the overall performance costs, as many HIDS/HUMS are accessing the same information, and provide an opportunity to identify, thanks to AI and machine learning, new correlations for anomaly detection.

2.1.5 Stress measurement implementation on the iMx8 board

To implement the process described in Figure 1 for a specific flight plan of the FMS application, we instrumented the FMS with METrICS to collect timing and performance counter values at a periodic task level, using the following pseudo-code implemented in python:

```
$N=0
for config in [0,nb_metrics_config]
  for it in [0,nb_fms_iterations]
    run the FMS instrumented with METrICS
    have METrICS collector process dumps collected info
    as run_$N.csv, raw_$N.csv and log_$N.csv files
    $N++
```

Thanks to METrICS, each execution of the FMS generates a set of output trace files that could later be used as an input dataset for the AI models:

- run_##.csv: Information about the parameters of the current execution, such as which hardware events are collected by METrICS, the total number of hardware events being collected, and so on...
- raw_##.csv: Raw collected trace data. Each row of the file corresponds to a time series associated to a pair of probes in the source code. These probes are positioned around the software's periodic tasks to collect behavioural information about these tasks.
- log_##.csv: user and application-level trace information. Currently used to dump deployment information (on which core run which application or stressing benchmark), FMS related information such as which trajectory is currently considered. Could be later completed with TEACHING KPI related information.

These files are further detailed in Section 2.1.6.

As for input parameters,

- **nb_fms_iteration**: The number of FMS iterations (used for the statistical purpose) is currently 10, and traces of different iterations of the same METrICS setting should provide similar results.
- **nb_metrics_config**: Number of different possible configurations of METrICS. Each of these configurations corresponds to a different set of hardware events being collected (and hence a different meaning of the PMC columns in the raw.csv file). As each core supports 6 performance monitor counter, each configuration corresponds to a set of 6 hardware events.

In the trace files generated so far, 5 different configurations have been used:

- { L1D_CACHE, L1D_CACHE_REFILL, L1D_CACHE_WB, L2D_CACHE, L2D_CACHE_REFILL, L2D_CACHE_WB} that focuses on the memory data path and caches
- { INST_RETIRED, LD_RETIRED, ST_RETIRED, BR_PRED, BR_MIS_PRED, PREFETCH} that focuses on application characterization in terms of instruction type
- { LD_SPEC, ST_SPEC, BR_PRED, DP_SPEC, VFP_SPEC, L1I_CACHE_REFILL } that focuses on the speculative aspects of the architecture, especially for the A72 core.
- { LD_SPEC, ST_SPEC, L1D_CACHE, INST_SPEC, L1I_CACHE, L1I_CACHE_REFILL } that focuses on speculative memory and control flow accesses.
- { L1D_TLB_REFILL, L1I_TLB_REFILL, L1D_CACHE, L1I_CACHE, BUS_ACCESS, MEM_ACCESS } that focuses on the TLB / MMU aspects.

The hardware performance events are further detailed in the official documentation, Section 12.4.2 of [14] and Section 11.8 of [15]. More hardware events could be taken into account.

Defining the meaningful event set is part of the discussion between WP5 and WP4, and could also be AI-driven.

2.1.6 Description of the avionics training datasets passed to WP4 to perform anomaly detection

Trace folder organization: For the avionics use case, collected traces are stored on the official TEACHING SharePoint, in **WP5 (AVL) / Avionics UC / traces** in the form of subfolders of tar.gz files. Each subfolder corresponds to a full experimental campaign with the Flight Management System (FMS) application following a specific flight plan, indicated by the folder name using ICAO codes. For instance, LFBT_LFBL corresponds to a fixed flight plan from TARBES Lourdes-Pyrénées (LFBT) to LIMOGES Bellegarde (LFBL).

Each such subfolder corresponding to a specific flight plan contains a set of tar.gz files corresponding to the different mapping of the avionics application as well as the stressing benchmarks / cyber-attack deployments.

Each tar.gz file has a name in this format: abcd.ef.tar.gz. The 4 first letters indicate which applications are deployed on the 4 dependable Cortex A53 cores of the iMx8 platform, and the last two letters on the application deployed on the non-dependable Cortex A72 cores.

Each letter indicates which application is deployed:

- x: Nothing is deployed on this core.
- F: The FMS is deployed on this core.
- T: The FFT computation is deployed on this core.
- S: A continuous stressing benchmark is deployed on this core
- Z: An intermittent stressing benchmark is deployed on this core.

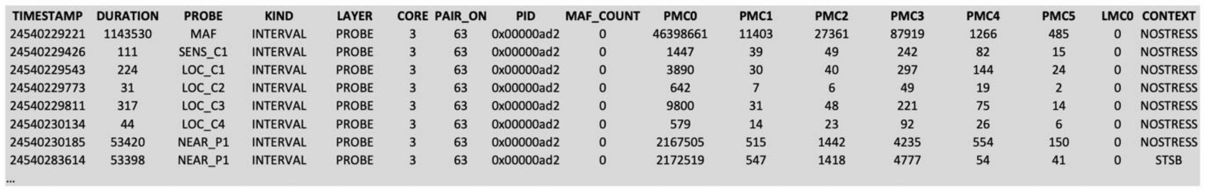
So, for instance;

- xFTx.xx.tar.gz corresponds to the avionic use-case running standalone in the system, serving as a nominal/normal reference, without any security-related anomalies.
- SFTS.xx.tar.gz and SFTS.SS.tar.gz correspond to maximum stress scenarios with permanent anomalies due to DoS attack on the memory resource. The first tarball only issues stress on the neighbouring Cortex A53 cores, whereas the second tarball extends this to the Cortex A72 cores.
- ZFTZ.xx.tar.gz and ZFTZ.ZZ.tar.gz correspond to the same scenario as above, but replacing the continuous stress by intermittent stress pausing and restarting every 15 seconds approximately.

As of April 2022, a total of 531MB of input training dataset is being generated for a single flight plan. It will be later completed by training dataset involving further flight plans, as well as with interactive flight plans where the pilot is allowed to change the flight plan online. Those traces only reflect the trace level of the hardware by the running software. There are no ethical issues related to those traces. Once stabilized such traces could also be provided as open access data.

Training dataset file formats: Section 2.1.5 states that the avionic training datasets are generated as a set of commas separated values files: raw.csv, run.csv and log.csv. A last CSV file hw_events.csv is common to all the FMS runs, being purely hardware dependant and listing all the hardware events that could be probed by the iMx8 platform as documented in Section 12.9 [14].

The collected time-series data to be used as training material are contained in the **raw.csv** files, as shown in Figure 10.



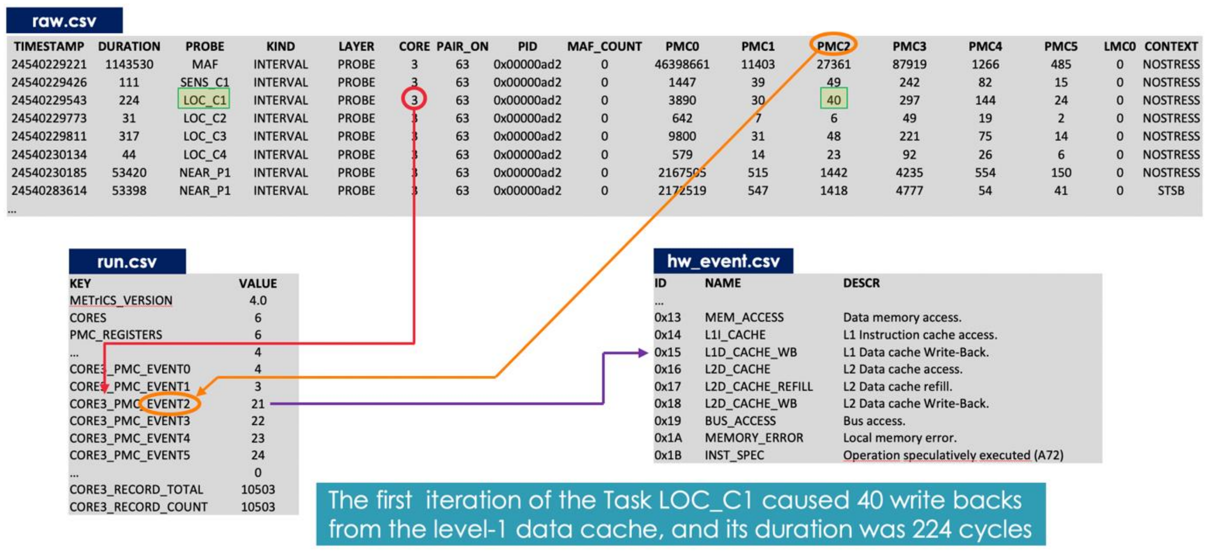
TIMESTAMP	DURATION	PROBE	KIND	LAYER	CORE	PAIR_ON	PID	MAF_COUNT	PMC0	PMC1	PMC2	PMC3	PMC4	PMC5	LMC0	CONTEXT
24540229221	1143530	MAF	INTERVAL	PROBE	3	63	0x00000ad2	0	46398661	11403	27361	87919	1266	485	0	NOSTRESS
24540229426	111	SENS_C1	INTERVAL	PROBE	3	63	0x00000ad2	0	1447	39	49	242	82	15	0	NOSTRESS
24540229543	224	LOC_C1	INTERVAL	PROBE	3	63	0x00000ad2	0	3890	30	40	297	144	24	0	NOSTRESS
24540229773	31	LOC_C2	INTERVAL	PROBE	3	63	0x00000ad2	0	642	7	6	49	19	2	0	NOSTRESS
24540229811	317	LOC_C3	INTERVAL	PROBE	3	63	0x00000ad2	0	9800	31	48	221	75	14	0	NOSTRESS
24540230134	44	LOC_C4	INTERVAL	PROBE	3	63	0x00000ad2	0	579	14	23	92	26	6	0	NOSTRESS
24540230185	53420	NEAR_P1	INTERVAL	PROBE	3	63	0x00000ad2	0	2167505	515	1442	4235	554	150	0	NOSTRESS
24540283614	53398	NEAR_P1	INTERVAL	PROBE	3	63	0x00000ad2	0	2172519	547	1418	4777	54	41	0	STSB
…																

Figure 10 - First lines of a raw.csv file

Each periodic task in the critical application is surrounded by a pair of probes allowing us to capture information on the particular task activation. As a consequence, each row in the raw.csv file corresponds to a task activation, whereas the columns correspond to:

- Timestamp: Timestamp of the task activation (in #cycle since boot)
- Duration: Duration of the task (in #cycle)
- Probe: Unique name of the software task if LAYER=probe
- Kind: probe kind: Interval, Begin or End (here always interval for periodic tasks)
- Layer: Where the METrICS probe is inserted (here always 'probe' for application level probe. METrICS could also retrieve information from the OS kernel and drivers).
- Core: Number of the core the probe was executed on (should be constant for a particular run as the FMS application is stuck by affinity to a particular dependable core)
- Pair_on: (please ignore, internal usage while pairing begin and end probes, and this pairing is already performed)
- Pid: process ID of the monitored application in the underlying operating system
- Maf_count: (Major frame number of the avionic application corresponds to the operational cycle iteration number of the application.)
- PMC#: Collected performance monitor counter (run.csv provides the information on which PMC corresponds to which hardware events)
- LMC#: Collected monitor counter from the software (could be either application or user defined. might later be used to store power, current, temperature information). More LMC counters could be provided at the cost of extra storage space and more memory access during collection.
- Context: Information about co-running contexts such as co-running stressing benchmarks or cyber-attacks. This information should not be used as training material, as the purpose is to detect these anomalies. But WP4 required this information to be added as a way to assess the performance of the detection models.
  - NoStress indicates a standalone execution without anomalies
  - STSB: Indicates the anomaly of co-running with an application performing a DoS attack on memory resource producing a large number of store requests inducing L1 & L2 cache misses.
  - LDSB: Indicates the anomaly of co-running with an application performing a DoS attack on memory resource producing a large number of load requests inducing L1 & L2 cache misses.

As stated, to exploit the Performance Monitor Counter-information (PMC), it is necessary to refer to the **run.csv** file that details which hardware event id is monitored by each PMC, and finally to refer to hw_event.csv to pair this hardware event id with the actual hardware event as shown in Figure 11.



raw.csv

TIMESTAMP	DURATION	PROBE	KIND	LAYER	CORE	PAIR_ON	PID	MAF_COUNT	PMC0	PMC1	PMC2	PMC3	PMC4	PMC5	LMC0	CONTEXT
24540229221	1143530	MAF	INTERVAL	PROBE	3	63	0x00000ad2	0	46398661	11403	27361	87919	1266	485	0	NOSTRESS
24540229426	111	SENS_C1	INTERVAL	PROBE	3	63	0x00000ad2	0	1447	39	49	242	82	15	0	NOSTRESS
24540229543	224	LOC_C1	INTERVAL	PROBE	3	63	0x00000ad2	0	3890	30	40	297	144	24	0	NOSTRESS
24540229773	31	LOC_C2	INTERVAL	PROBE	3	63	0x00000ad2	0	642	7	6	49	19	2	0	NOSTRESS
24540229811	317	LOC_C3	INTERVAL	PROBE	3	63	0x00000ad2	0	9800	31	48	221	75	14	0	NOSTRESS
24540230134	44	LOC_C4	INTERVAL	PROBE	3	63	0x00000ad2	0	579	14	23	92	26	6	0	NOSTRESS
24540230185	53420	NEAR_P1	INTERVAL	PROBE	3	63	0x00000ad2	0	2167505	515	1442	4235	554	150	0	NOSTRESS
24540283614	53398	NEAR_P1	INTERVAL	PROBE	3	63	0x00000ad2	0	2172519	547	1418	4777	54	41	0	STSB
...																

run.csv

KEY	VALUE
METrICS_VERSION	4.0
CORES	6
PMC_REGISTERS	6
...	4
CORE3_PMC_EVENT0	4
CORE3_PMC_EVENT1	3
CORE3_PMC_EVENT2	21
CORE3_PMC_EVENT3	22
CORE3_PMC_EVENT4	23
CORE3_PMC_EVENT5	24
...	0
CORE3_RECORD_TOTAL	10503
CORE3_RECORD_COUNT	10503

hw_event.csv

ID	NAME	DESCR
...		
0x13	MEM_ACCESS	Data memory access.
0x14	L1I_CACHE	L1 Instruction cache access.
0x15	L1D_CACHE_WB	L1 Data cache Write-Back.
0x16	L2D_CACHE	L2 Data cache access.
0x17	L2D_CACHE_REFILL	L2 Data cache refill.
0x18	L2D_CACHE_WB	L2 Data cache Write-Back.
0x19	BUS_ACCESS	Bus access.
0x1A	MEMORY_ERROR	Local memory error.
0x1B	INST_SPEC	Operation speculatively executed (A72)

The first iteration of the Task LOC_C1 caused 40 write backs from the level-1 data cache, and its duration was 224 cycles

Figure 11 - Retrieving data semantic information from generated traces

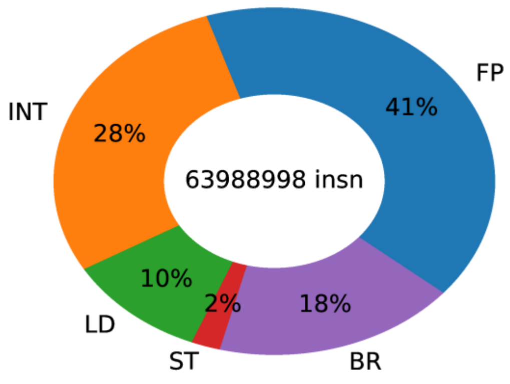
A full data collection campaign for a single trajectory totalises 531MB of CSV dataset files, with 5.252.000 total rows of time series in all the raw.csv files.

2.1.7 Statistical Analysis of collected datasets

Prior to being used in an AI framework, METrICS [1] was relying on post-processing statistical analysis to characterize both the safety-critical software and hardware. This statistical analysis relies on the same traces of hardware events as the AI models from WP4, but requires to be guided by an expert in the domain to extract meaningful information.

Even though this statistical information is not strictly necessary for learning-oriented AI models, such analysis is still of importance to check for the correctness of the input dataset provided to the AI models, to extract some semantics usable by hybrid AI approaches, and to provide human-readable information about the running hardware and software with statistical visualizations.

Software profiling related information: A first characterization of the software consists in providing some profiling information in terms of instruction type ratio, such as integer instructions, floating-point instructions, branch instructions, load and store instructions. Figure 12 shows the profile of two critical applications composing the TEACHING avionics use case, the FMS guiding the aircraft, and the fast Fourier transform filtering the aircraft sensor information.



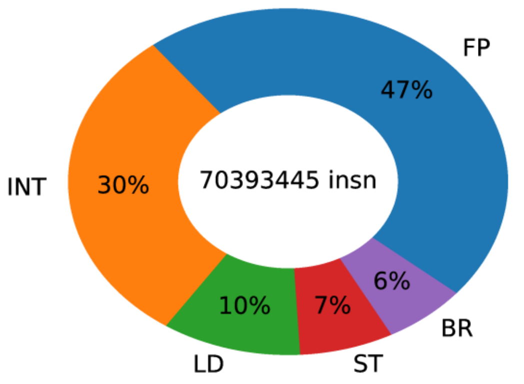


Figure 12 - Respective instruction profile of the FMS application (on the left) and the FFT application (on the right)

Even though both profiles look the same, with a similar total number of instructions being executed every 200ms, and similar ratios in terms of integer versus floating-point instructions, both applications are impacted very differently by potential safety hazards or cyber-attacks, especially from stress-based failures or DoS attacks on the application memory paths. Figure 13 shows the profile of such a DoS attack.

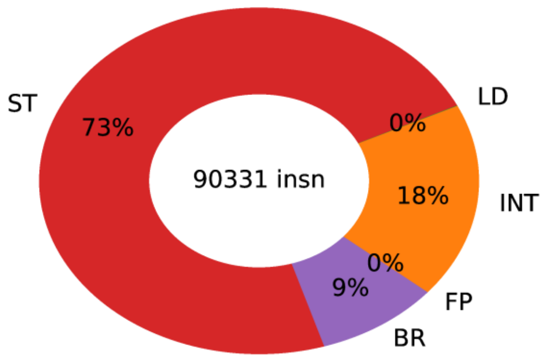


Figure 13 - Instruction profile of a cache-miss DoS attack on the L2 cache

As expected, such a DoS attack is mainly focusing on memory access instructions. Here the stressing benchmarks if performing out of order store memory accesses trying to maximize the number of L2 cache misses, hence maximizing the number of main memory accesses, while having an access behaviour not predictable by the cache prefetchers.

Such a malicious co-running application can have a very negative impact on the critical applications, as their data is evicted from the cache by the stressing benchmark, causing more cache misses at the level of the critical application, and therefore longer memory access slowing down the hard-realtime functions, endangering the use-case deadlines.

Runtime profiling information: An important characterization aspect for time-critical applications, is runtime profiling, capturing the running times of every periodic task, and checking them against the task deadlines.

Figure 14 presents the duration of each operational iteration of the FMS application that is run every 200ms, while the FMS application is running standalone on the target platform. It corresponds to the nominal/normal execution of the application without any anomalies.

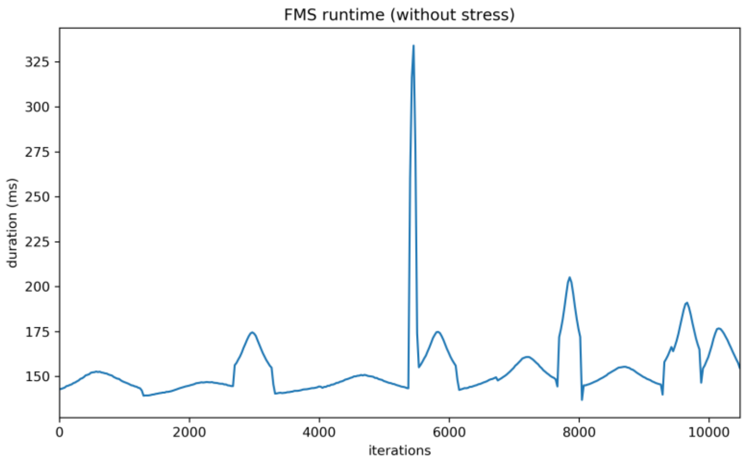


Figure 14 - Runtime profiling of the FMS application without anomalies

The figure shows some runtime variation between iterations which duration varies with the complexity of the trajectory (less computation for straight line trajectories, and more computation for curves and when sensor data is refreshed).

Figure 15 shows the same FMS runtime information, for the same flight plan, but this time while co-running with the DoS attack presented in Figure 13.

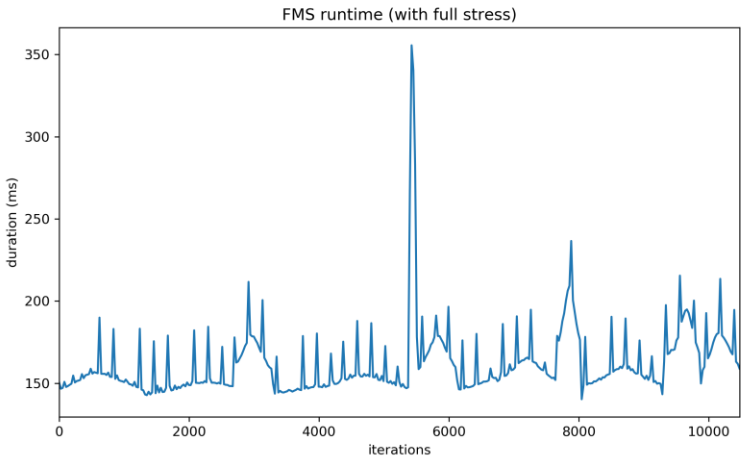


Figure 15 - Runtime profiling of the FMS application with anomalies

Peak slowdown can now be observed, regularly endangering the 200ms deadline associated with each operational iteration. AI models have the opportunity to anticipate these peak slowdowns, observing the increase of cache misses beforehand, and therefore detecting such a DoS attack.

Temperature and power consumption profiling: Beyond the performance monitor information collected with METrICS, THErM allows us to collect power and voltage information from the SCP processor through its MHU interface as described in Deliverable D5.2 Section 4.2.

The left chart of Figure 16 relates the temperature of the dependable core of the iMx8 platform while running the FMS application consecutively 50000 times. The first darker part of the curve corresponds to having the board running with air conditioning turned on to keep an external temperature at 20°c. The air conditioning is turned off for the lighter part of the right with significative impact on the runtime, as the Dynamic Thermal Management (DTM) mechanisms enter into action to protect the hardware, affecting the frequency scaling, and therefore the application runtimes.

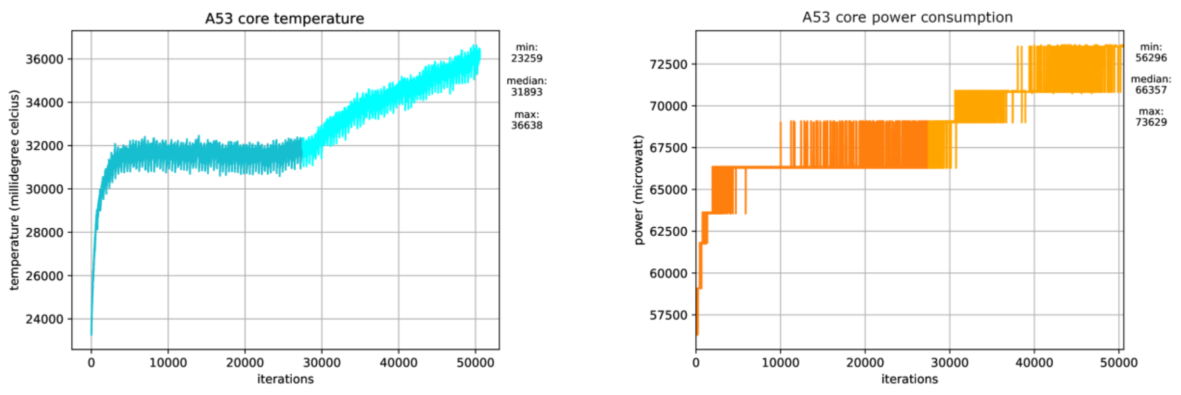


Figure 16 - Temperature (left) and power (right) profiling of the FMS application

The right chart of Figure 16 shows power consumption during the same scenario, again with extra consumption when the external temperature is above a given threshold.

2.2 The Automotive Use Case

As stated in Section 1 of this document and described in detail in deliverables D5.1 (*Initial use case specifications*) and D5.2 (*Preliminary use case deployment, implementation and integration report*), the automotive use case supports the development of driving automation by integrating humans into the control loop. Human integration is only meaningful if there is an improved understanding of human psychophysiological factors. That implies the detection of the human state for training and modelling purposes and then later integration of those models into real vehicles. Once implemented in reality, vehicle occupant state will have to be monitored and adequately responded to e.g., by adopting vehicle controls to alter the driving style to respond to the individual needs of vehicle occupants or to respond to the changing environmental conditions. The implementation in real vehicles is to rely on the tailored TEACHING platform. This whole process relies on the acquisition of relevant data. For that reason, a dedicated TEACHING driving simulation study is performed, as described in the rest of section 2.2.

2.2.1 Technology tailoring for the automotive use case

Activities are ongoing to accommodate the usage of necessary algorithms in the TEACHING platform and integrate the same into the automotive use case. The following set of tailoring actions has been undertaken to achieve the current state of the automotive use case at the time of the data acquisition campaign.

To enable the human-driven adaptation of the driving automation the Energy-Efficient Cooperative Adaptive Cruise Control (EECACC) is optimised for the TEACHING project to demonstrate the usability of I&M's real-time Linux-based platform as a control device with embedded algorithms. EECACC is tested in both, open and closed-loop formats. The behaviour is compliant with the acceptance criteria, as the platform and a Functional Mock-Up (FMU) that run the same EECACC algorithms also produce equally comparable outputs. While there are some instabilities associated with the integration of the platform into the driving simulator, the work is ongoing to resolve the issues and secure the platform's integration in the final simulator. Figure 17 depicts the comparison between the signals generated using two different environments i.e. velocity (red), acceleration (blue) and distance covered (green) signals from the figure's top are generated using the platform, while the bottom part of the figure depicts the same signals generated in a simulated (FMU) environment.

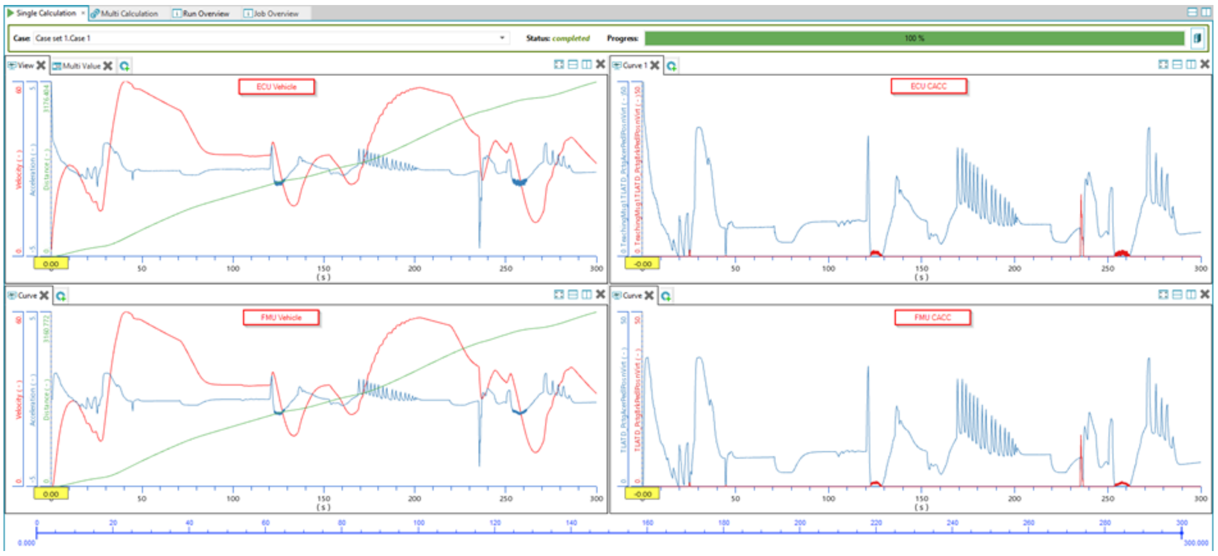


Figure 17 EECACC - ECU vs FMU comparison

Generally, as detailed in section 4.3 of D5.2, EECACC is informed of the vehicle's external environment and subsequently concludes the demand speed per the commanded driving aggressiveness and fuel consumption, as depicted in Figure 18. In this case, the key tailoring activities include the selection of appropriate driving modes by the EECACC subject to the scenario requirements. The tailoring will continue in direction of EECACC responding to the specific responses of the vehicle occupants.

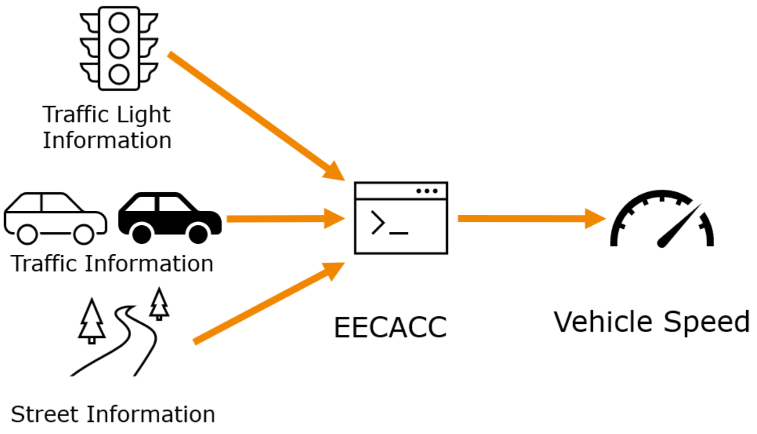


Figure 18 EECACC input - output variables

The EECACC and the other software have been instrumented using the LTTNG tracing toolkit. This is performed to allow the generation of execution traces to be later analysed, including both, user space and kernel events. The expected outcome is the collection of the latencies of all function calls. This analysis will give a measure of the stability of the execution of software. This insight is needed in order to detect where optimizations are needed and drive the work in an efficient manner.

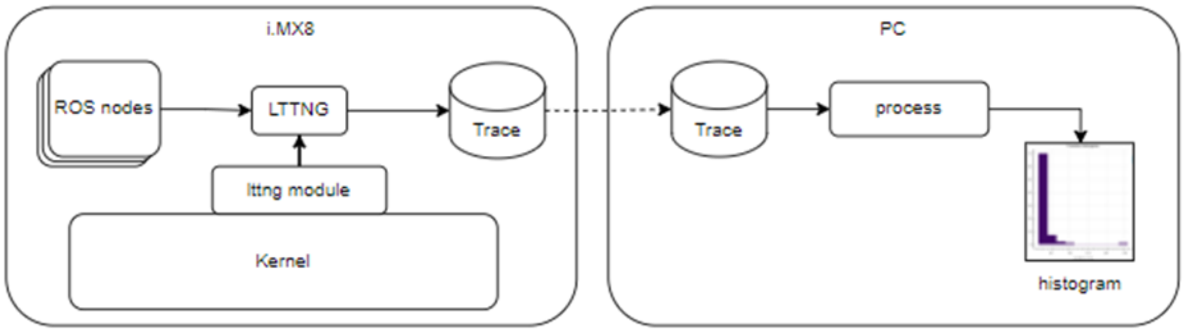


Figure 19 EECACC using LTNNG tracing toolkit

While the traces are already being generated, the analysis is in the process of development. The resulting insights will enable optimisations of the software to solve the instability issues before the next driving simulator demonstrator session, which is planned to have the platform physically integrated into the driving simulator.

2.2.2 Driving simulator setup and procedure

The starting point for the driving simulator study is detailed in [16] and [17]. This initial setup is considerably altered to fit the needs of the TEACHING project described in D5.1. The preliminary phase of the study (Controlled laboratory environment test) is detailed in section 4.4 of D5.2. The implementation of the study continues from that preliminary implementation and combines with an altered version of the previous study to advance from the previous primary aim of determining if the usage of an advanced driving simulator can improve consumer trust and acceptance of driving automation. The current study is focusing on understanding the key human factors, with a focus on stress and comfort in the context of driving automation safety.

The driving simulator combines support hardware (e.g., control PCs, body and vehicle sensors, etc.) and the software that integrates the hardware components into the test environment to ease communication and data synchronisation. Due to the range of often incompatible equipment being used, data synchronisation is a complex issue that must be resolved should the collected data offer an opportunity for a meaningful analysis. The actual simulator is a dynamic hexapod platform that carries a cockpit, which incorporates a real vehicle cut out containing the driver seat and the complete driver environment. The virtualisation is achieved by projecting a high frame rate (100Hz, 4K) on a 180° view canvas for realistic simulation and reduction of motion sickness. The simulator motion control allows six degrees of freedom as needed for three-dimension translational and pivoting around each axis. The maximum available acceleration is 6m/s^2 with one meter of manoeuvring possibility in all three dimensions (x/y/z) and a rotational capability.

For the TEACHING study, the objective human measurements are done using a chest belt for the measurement of Heart Rate (HR) and the Heart Rate Variability (HRV), and a fingertip sensor for the measurement of Galvanic Skin Response (GSR) as non-invasive electrodermal activity sensor that determines the intensity of human emotional state (emotional arousal). This specific sensor combination targets the TEACHING intent to quantify and estimate human stress and comfort within the context of driving safety.

The subjective perception of the driving situation is collected through a set of pre-simulation, interim and post-simulation questionnaires. A total of 23 pre-simulation questionnaires are presented to the study subjects to respond to. These are gathered into two groups. The first group of questions collects the data on **social demographics** and **automated driving experience** to enable the grouping of participants based on their background and experience with automation. These questions also offer an insight into the potential increase in user **acceptance** across different demographic groups. The second set of questions seeks to receive a Lickert scale response to several statements that measure the extent to which users find automated driving systems **useful** and **trustworthy**. These questions consider the user's trust and perception of safety at the start of the process before entering the driving simulator. As such, the questions focus on the subject's understanding of high-level understanding of driving automation and its relation to safety.

The interim questionnaires are presented to the subjects during the simulations, in between the driving scenarios at the driving simulator. The subjects encounter seven driving scenarios in the driving simulator with automated driving simulated at SAE level 4+ AD [18]. That means that the driving simulator drives automatically with no need for the subjects to control the vehicle and the subjects are requested not to intervene in the course of the simulator study. The focus of the study is on their psychophysiological status and not on their physical responses. In real driving situations (for SAE 4), the subjects would have an opportunity to actively intervene in the situation if they feel that they need to do so.

Each scenario is followed by a set of questions about the individual perception of driving situations. The questions are grouped to address the following human factors: **comfort/discomfort**, **workload** (e.g., mental or physical demand), **safety**, **satisfaction** with the system, **trust**, and **acceptance**.

Upon the driving simulation, the subjects leave the cockpit and all body sensors are removed. Upon a brief check of their mental and physical state, the subjects respond to the post-simulation questionnaire. The procedure guarantees subject engagement to minimise the possibility of post-test motion sickness outside the controlled test environment. The posed questions focus on the quantification of the expression of **trust** as a way of measuring trust improvement as a result of driving simulation, **cognitive safety** specifically related to the experienced driving automation, the general system **usability** unrelated to the particular scenarios, and user **acceptance**.

Other than the initial demographic set of questions, all other questions are answered using the Lickert scale.

2.2.3 Driving scenarios

Seven driving scenarios are deployed in the first study campaign. The initial scenario is used for familiarisation with the environment. The following six scenarios offer an opportunity to experience different driving traffic situations. Those six scenarios are three pairs of scenarios; each scenario is replayed the second time under different conditions. The focus of each scenario pair is overtaking (pair 1), platooning (pair 2), as defined in the TEACHING proposal, and unexpected breaking (pair 3), as listed in Table 1.

Table 1 Driving scenario list

ID	LVL 5 - scenarios	environment/adaptation
1	EECACC with a sharp cut in after overtaking the ego-vehicle	**comfort** driving mode
2		**sport** driving mode
3		comfort mode in **good weather**

4	Motorway platooning with vehicle cutting out	comfort mode in **bad weather**
5	Dysfunctional EECACC responding to phantom breaking	**maintain mode** after the breaking
6		**different mode** after the breaking

A brief description of driving scenarios is provided in the rest of this subsection. All scenarios are performed using a left-hand driven passenger ego vehicle.

Scenario 1 (duration ~ 2:35)

- Ego-vehicle moves on a relatively straight country road with several gentle bends (allow acclimatisation in the anticipation phase)
- A few seconds after 1:00, the single lane country road turns into a motorway
- At approximately 1:32, another passenger vehicle appears in the left lane (overtaking the ego vehicle) and cuts in front of the ego vehicle
- The overtaking vehicle indicates with a right-hand indicator and cuts in quite sharply in front of the ego vehicle
- **Ego vehicle brakes rather sharply to avoid a collision at 1:42 (expecting vehicle occupant response)**, as depicted in Figure 20
- The overtaking vehicle moves rapidly ahead creating the relaxation phase till the end of the scenario

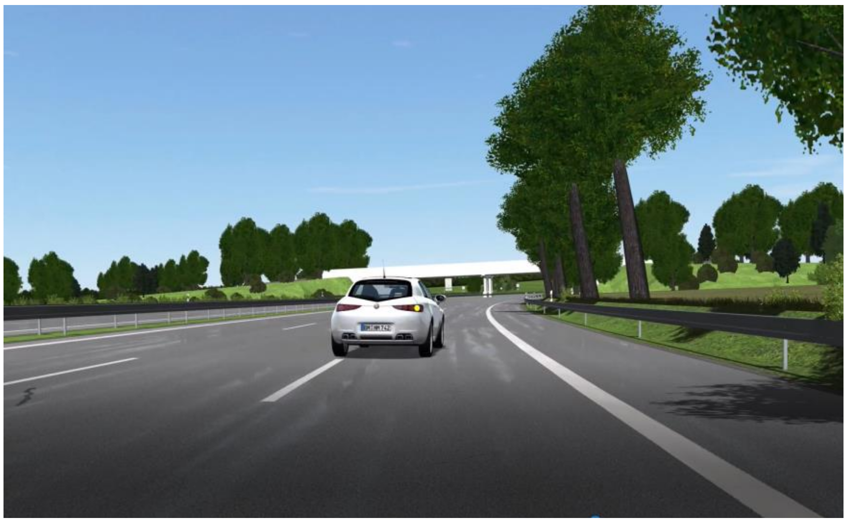


Figure 20 Scenarios 1 and 2 - sharp cut-in

Scenario 2 (duration ~ 2:35)

- **Scenario 1 is repeated**, but this time, the 'comfort' driving mode is replaced with the **'sporty' driving mode**, hence allowing a shorter distance between the ego vehicle and the vehicle in front

Scenario 3 (duration ~ 2:37)

- The scenario begins with the ego vehicle departing a city road and merging into a slip road towards a motorway
- As the vehicle approaches the motorway, a formed platoon of four vehicles passes past the ego vehicle at a speed, as depicted in Figure 21

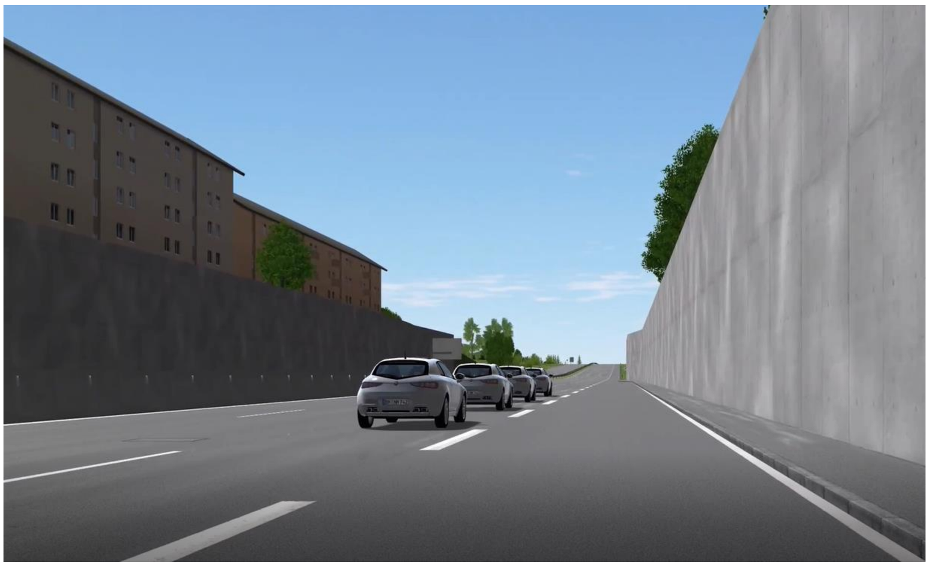


Figure 21 Scenario 3 - joining a motorway with a formed platoon

- As the ego vehicle joins the motorway (at a speed considerably lower than that of the platoon), it accelerates to merge with the platoon at its end
- At approximately 1:27, **the ego vehicle becomes a part of the platoon** i.e., its last (5th) member
- Approximately 5 seconds later, the fourth **vehicle in the platoon (the one in front of the ego vehicle) cuts out of the platoon** overtaking the 3 vehicles in front of it, as depicted in Figure 22

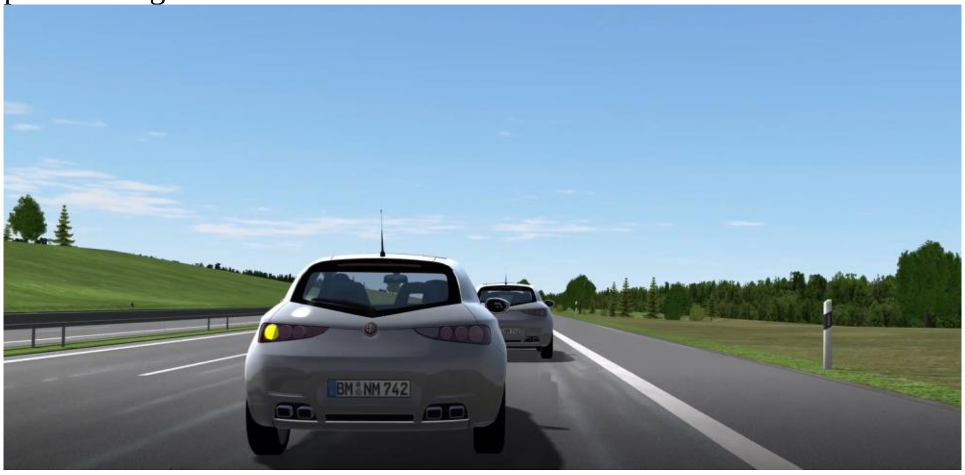


Figure 22 Scenario 3 - platoon cut out

- **Ego vehicle catches up with the platoon** (i.e., reduces the distance with its 3rd member), but this time at a lower acceleration rate than was the case when it joined the platoon
- The remaining 50 seconds of the scenario are the relaxation phase i.e., the ego vehicle continues following the other members of the platoon

Scenario 4 (duration ~ 2:37)

- Scenario 4 is a **repeat of scenario 3 in less favourable environmental conditions** with reduced visibility, as depicted in Figure 23, with the expectation to entice different responses from the study subjects.

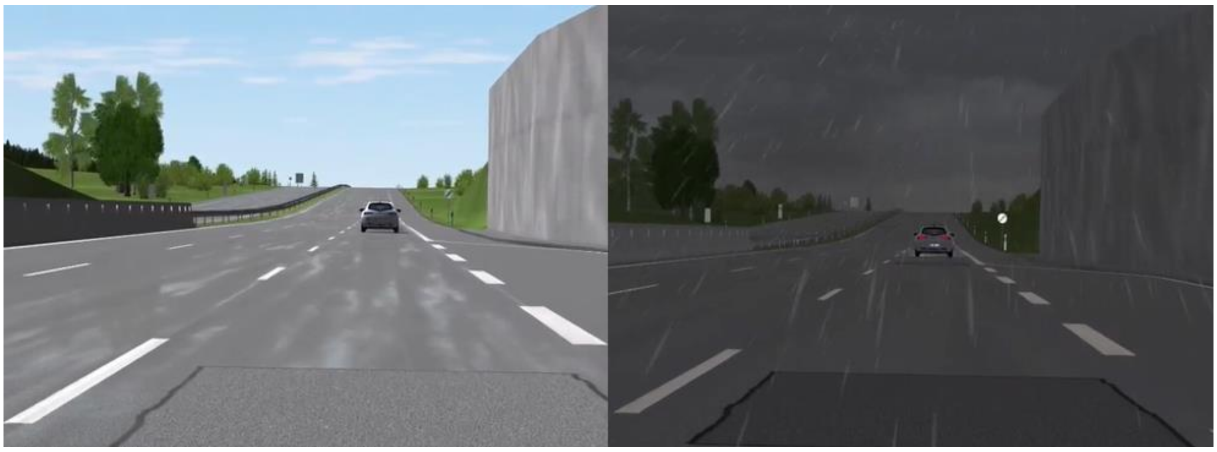


Figure 23 Scenario 3 vs Scenario 4 - Adverse Weather Conditions

<u>Scenario 5 (duration ~ 2:35)</u>

- ➢ The scenario focuses on the city road driving
- ➢ The initial acclimatisation phase is followed by a relatively sharp breaking at the traffic light by a petrol station at approximately 00:45
- ➢ The second event of significant interest occurs as the road bends near a side-parked delivery truck at around 01:10 (depicted in) – the deliberately introduced **phantom breaking** could occur in the real driving scenario due to the optical reflection from the truck surface or sensor inaccuracy (i.e., the truck could be interpreted as a vehicle on the road, despite being parked on the side of the road)

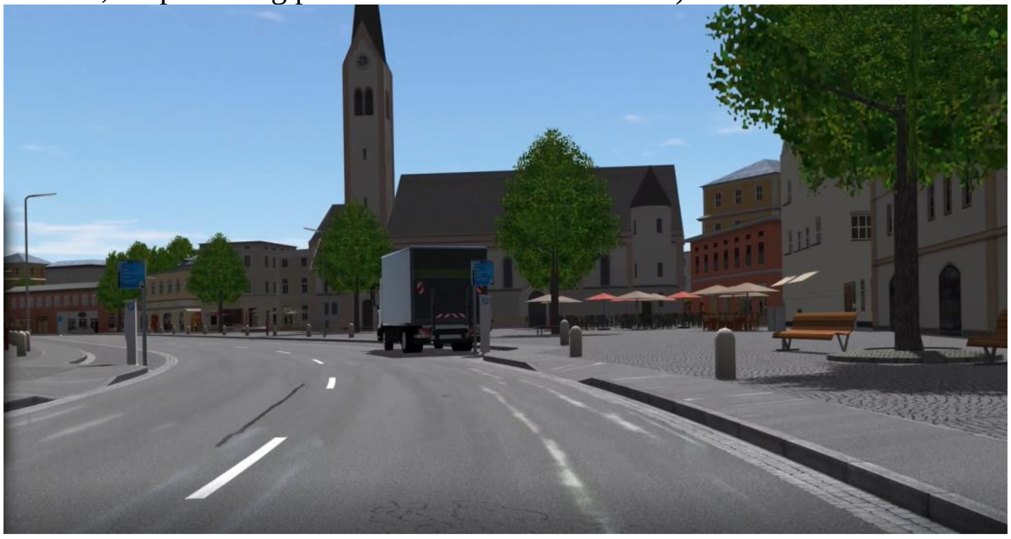


Figure 24 Scenario 5 - phantom breaking in city driving conditions

- ➢ The third, less significant event occurs at the road crossing at 01:47 – the reaction could be possible in anticipation of the ego vehicle breaking at the traffic light, which does not occur due to the green light being displayed at the traffic light

<u>Scenario 6 (duration ~ 2:35)</u>

- ➢ While the ego vehicle starts driving in 'sporty' mode in scenarios 5 and 6, the phantom breaking of scenario 5 causes the vehicle to continue driving in 'comfort' mode while the **scenario 6 switches back to the 'sporty' mode**

2.2.4 Simulator study data

While adequate human-machine interaction is the prerequisite for intuitive usage and acceptance of driving automation, it is also a viable channel for obtaining real-time user feedback for the optimisation of vehicle controls and prediction of human needs. The feedback enables AI training to understand user behaviour and informs vehicle controls of user expectations within the legal road limitation. In addition, it tracks user satisfaction and trust in the system, hence improving the mobility solutions and increasing road safety. Trust and acceptance are potentially the crucial parameters for determining the success of autonomous driving deployment in wider society. Hence, this data taking campaign seeks to determine the appropriate and measurable parameters to be able to quantify trust and acceptance in a physically safe environment using dependable methods. It also delivers data for the AI training activities (WP4), just as is the case with the avionics use case.

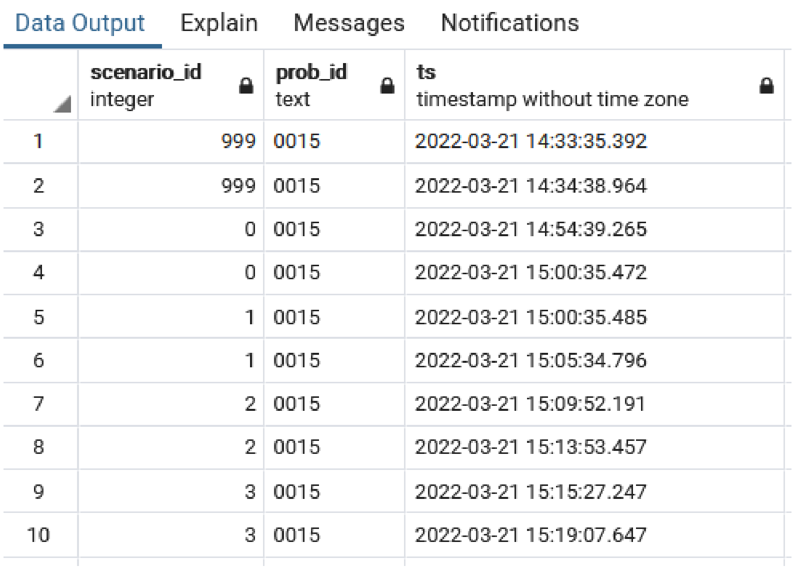
The data acquisition campaign is based on a recent TEACHING-specific study with a set of tailor-made driving scenarios (described in Section 2.2.3) to gather user subjective (questionnaire) and objective (psychophysiological) feedback, as described in the simulator setup and procedure of Section 2.2.2. The human state is estimated from objective human measurements of HR, HRV and GSR. The subjective perception of the driving automation is yielded from the questionnaires, as described in Section 2.2.2.

The subjective data were collected anonymously. There is no possibility to assign the data to the respective participating person to safeguard the personal data per article 9 of the General Data Protection Regulation (EU) (GDPR). The following describes the key consideration at the initial stage of data analysis:

- The study sample consists of 40 subjects with an average age of 35.5 years (sd = 8.5). A total of 16 women and 24 men took part in the study. A driving licence is held by 39 participants. On average, the participants have held their driving license for 16.7 years (sd = 7.8).
- Scales are created and evaluated regarding their quality criteria (internal consistency, reliability etc.).
- Depending on the quality criteria, individual items were excluded from the data analysis for specific scales.
- Data is then evaluated using multivariate statistical analysis methods.
- The difference in the dependent variables (comfort, workload, etc.) concerning different driving modes or environmental factors within a driving scenario is calculated using variance analysis with repeated measurements. Post-tests are determined and graphics are created. Sociodemographic variables are included in the analyses.
- In addition, correlations were determined between the comfort, discomfort scale, workload, and relevant psychological aspects such as trust, safety, acceptance and satisfaction for external validation.

When it comes to the objective data, the following are the key data characteristics:

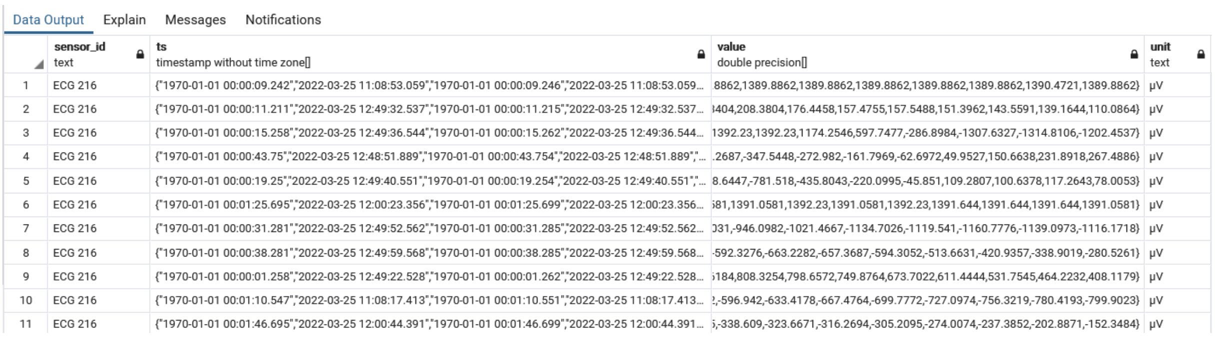
- Data is stored in the database
- The scenario data is associated with the following (Figure 25 Scenario DataFigure 25):
    - Scenario ID
    - Participant ID (‘*prob_id*’)
    - Start and stop timestamps, both are which are used to extract appropriate sensor data for each participant



	scenario_id integer	prob_id text	ts timestamp without time zone
1	999	0015	2022-03-21 14:33:35.392
2	999	0015	2022-03-21 14:34:38.964
3	0	0015	2022-03-21 14:54:39.265
4	0	0015	2022-03-21 15:00:35.472
5	1	0015	2022-03-21 15:00:35.485
6	1	0015	2022-03-21 15:05:34.796
7	2	0015	2022-03-21 15:09:52.191
8	2	0015	2022-03-21 15:13:53.457
9	3	0015	2022-03-21 15:15:27.247
10	3	0015	2022-03-21 15:19:07.647

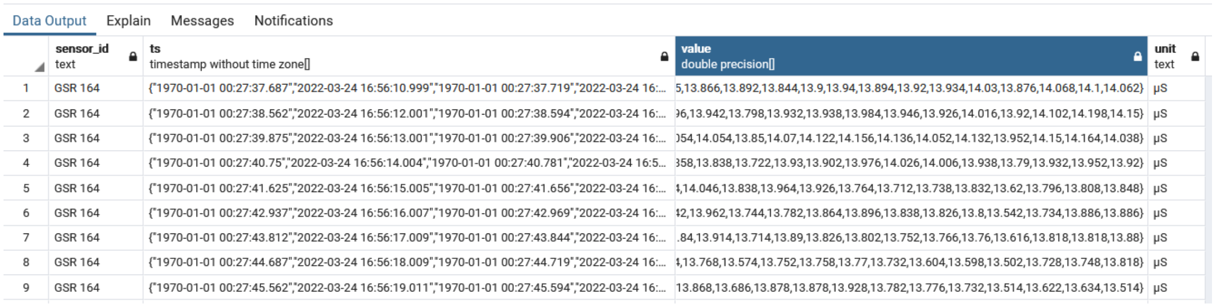
Figure 25 Scenario Data

- Sensor data:
    - Pre-processed sensor (ECG and GSR) data - timestamps and values, as depicted in Figure 26 and Figure 27.
    - The data is extracted from the socket in batches and then reconstructed using the relative time stamps.
    - The processing is performed to achieve a more readable format. Some additional processing is performed for some participants to correct the timestamps.



	sensor_id text	ts timestamp without time zone[]	value double precision[]	unit text
1	ECG 216	{"1970-01-01 00:00:09.242","2022-03-25 11:08:53.059","1970-01-01 00:00:09.246","2022-03-25 11:08:53.059...	.8862,1389.8862,1389.8862,1389.8862,1389.8862,1389.8862,1390.4721,1389.8862}	µV
2	ECG 216	{"1970-01-01 00:00:11.211","2022-03-25 12:49:32.537","1970-01-01 00:00:11.215","2022-03-25 12:49:32.537...	404,208.3804,176.4458,157.4755,157.5488,151.3962,143.5591,139.1644,110.0864}	µV
3	ECG 216	{"1970-01-01 00:00:15.258","2022-03-25 12:49:36.544","1970-01-01 00:00:15.262","2022-03-25 12:49:36.544...	1392.23,1392.23,1174.2546,597.7477,-286.8984,-1307.6327,-1314.8106,-1202.4537}	µV
4	ECG 216	{"1970-01-01 00:00:43.75","2022-03-25 12:48:51.889","1970-01-01 00:00:43.754","2022-03-25 12:48:51.889","...	.2687,-347.5448,-272.982,-161.7969,-62.6972,49.9527,150.6638,231.8918,267.4886}	µV
5	ECG 216	{"1970-01-01 00:00:19.25","2022-03-25 12:49:40.551","1970-01-01 00:00:19.254","2022-03-25 12:49:40.551","...	8.6447,-781.518,-435.8043,-220.0995,-45.851,109.2807,100.6378,117.2643,78.0053}	µV
6	ECG 216	{"1970-01-01 00:01:25.695","2022-03-25 12:00:23.356","1970-01-01 00:01:25.699","2022-03-25 12:00:23.356...	81,1391.0581,1392.23,1391.0581,1392.23,1391.644,1391.644,1391.644,1391.0581}	µV
7	ECG 216	{"1970-01-01 00:00:31.281","2022-03-25 12:49:52.562","1970-01-01 00:00:31.285","2022-03-25 12:49:52.562...	31,-946.0982,-1021.4667,-1134.7026,-1119.541,-1160.7776,-1139.0973,-1116.1718}	µV
8	ECG 216	{"1970-01-01 00:00:38.281","2022-03-25 12:49:59.568","1970-01-01 00:00:38.285","2022-03-25 12:49:59.568...	-592.3276,-663.2282,-657.3687,-594.3052,-513.6631,-420.9357,-338.9019,-280.5261}	µV
9	ECG 216	{"1970-01-01 00:00:01.258","2022-03-25 12:49:22.528","1970-01-01 00:00:01.262","2022-03-25 12:49:22.528...	184,808.3254,798.6572,749.8764,673.7022,611.4444,531.7545,464.2232,408.1179}	µV
10	ECG 216	{"1970-01-01 00:01:10.547","2022-03-25 11:08:17.413","1970-01-01 00:01:10.551","2022-03-25 11:08:17.413...	,-596.942,-633.4178,-667.4764,-699.7772,-727.0974,-756.3219,-780.4193,-799.9023}	µV
11	ECG 216	{"1970-01-01 00:01:46.695","2022-03-25 12:00:44.391","1970-01-01 00:01:46.699","2022-03-25 12:00:44.391...	,-338.609,-323.6671,-316.2694,-305.2095,-274.0074,-237.3852,-202.8871,-152.3484}	µV

Figure 26 Raw ECG Sensor Data



	sensor_id text	ts timestamp without time zone[]	value double precision[]	unit text
1	GSR 164	{"1970-01-01 00:27:37.687","2022-03-24 16:56:10.999","1970-01-01 00:27:37.719","2022-03-24 16:...	5,13.866,13.892,13.844,13.9,13.94,13.894,13.92,13.934,14.03,13.876,14.068,14.1,14.062}	µS
2	GSR 164	{"1970-01-01 00:27:38.562","2022-03-24 16:56:12.001","1970-01-01 00:27:38.594","2022-03-24 16:...	6,13.942,13.798,13.932,13.938,13.984,13.946,13.926,14.016,13.92,14.102,14.198,14.15}	µS
3	GSR 164	{"1970-01-01 00:27:39.875","2022-03-24 16:56:13.001","1970-01-01 00:27:39.906","2022-03-24 16:...	54,14.054,13.85,14.07,14.122,14.156,14.136,14.052,14.132,13.952,14.15,14.164,14.038}	µS
4	GSR 164	{"1970-01-01 00:27:40.75","2022-03-24 16:56:14.004","1970-01-01 00:27:40.781","2022-03-24 16:5...	58,13.838,13.722,13.93,13.902,13.976,14.026,14.006,13.938,13.79,13.932,13.952,13.92}	µS
5	GSR 164	{"1970-01-01 00:27:41.625","2022-03-24 16:56:15.005","1970-01-01 00:27:41.656","2022-03-24 16:...	4,14.046,13.838,13.964,13.926,13.764,13.712,13.738,13.832,13.62,13.796,13.808,13.848}	µS
6	GSR 164	{"1970-01-01 00:27:42.937","2022-03-24 16:56:16.007","1970-01-01 00:27:42.969","2022-03-24 16:...	42,13.962,13.744,13.782,13.864,13.896,13.838,13.826,13.8,13.542,13.734,13.886,13.886}	µS
7	GSR 164	{"1970-01-01 00:27:43.812","2022-03-24 16:56:17.009","1970-01-01 00:27:43.844","2022-03-24 16:...	.84,13.914,13.714,13.89,13.826,13.802,13.752,13.766,13.76,13.616,13.818,13.818,13.88}	µS
8	GSR 164	{"1970-01-01 00:27:44.687","2022-03-24 16:56:18.009","1970-01-01 00:27:44.719","2022-03-24 16:...	4,13.768,13.574,13.752,13.758,13.77,13.732,13.604,13.598,13.502,13.728,13.748,13.818}	µS
9	GSR 164	{"1970-01-01 00:27:45.562","2022-03-24 16:56:19.011","1970-01-01 00:27:45.594","2022-03-24 16:...	13.868,13.686,13.878,13.878,13.928,13.782,13.776,13.732,13.514,13.622,13.634,13.514}	µS

Figure 27 Raw GSR Sensor Data

The following set of figures is drawn for specific scenarios as examples for data representation. All figures show the ECH, Heart Rate and GSR signals. The red lines in the bottom graph of each figure indicates the anticipated key points of interest where specific attention is paid to data analysis. Each red line is preceded by 10 seconds of anticipation phase, superseded by 10 seconds of reaction phase and then 10 seconds of relaxation phase.

- Figure 28: scenario 1, with the key anticipated reaction (post-red line of the the bottom graph) when the overtaking vehicle appears on the left hand side of the ego vehicle
- Figure 29: scenario 1, but different subject to demonstrate varying reactions between different vehicle occupants
- Figure 30: scenario 2 wiht a similar timing expectation,yet a different reaction due to a different driving mode as for the scenario 1
- Figure 31: scenario 3 with anticipated reaction as the ego vehicle is joining the motorway and then again as it interacts with the platoon
- Figure 32: scenario 4 with less dynamic response, as the difference from the previous scenario is only the environmental conditions i.e. the visibility
- Figure 33: scenario 5 with the anticipated reaction at the sharp breaking by the traffic light, phantom breaking, and an approach to the final cross-road
- Figure 34: zoomed in scenario 5 to show occasional issue with the heart-beat peak detection (hence, the ned for pre-processing)
- Figure 35: scenario 6 with anticipated reactions, just as in scenario 5
- Figure 36: data outage demonstrating the need for pre-processing

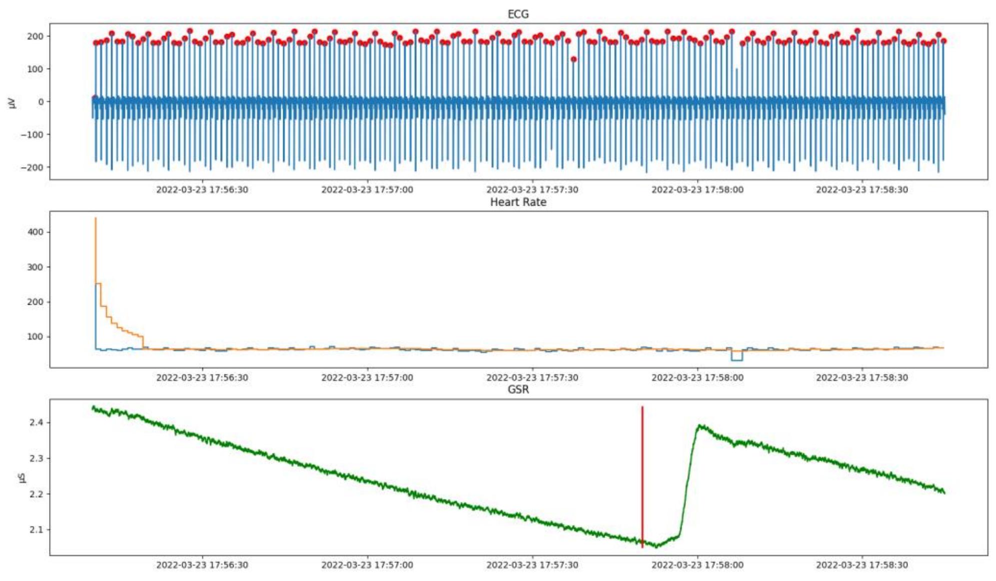


Figure 28 Scenario 1, participant 2111

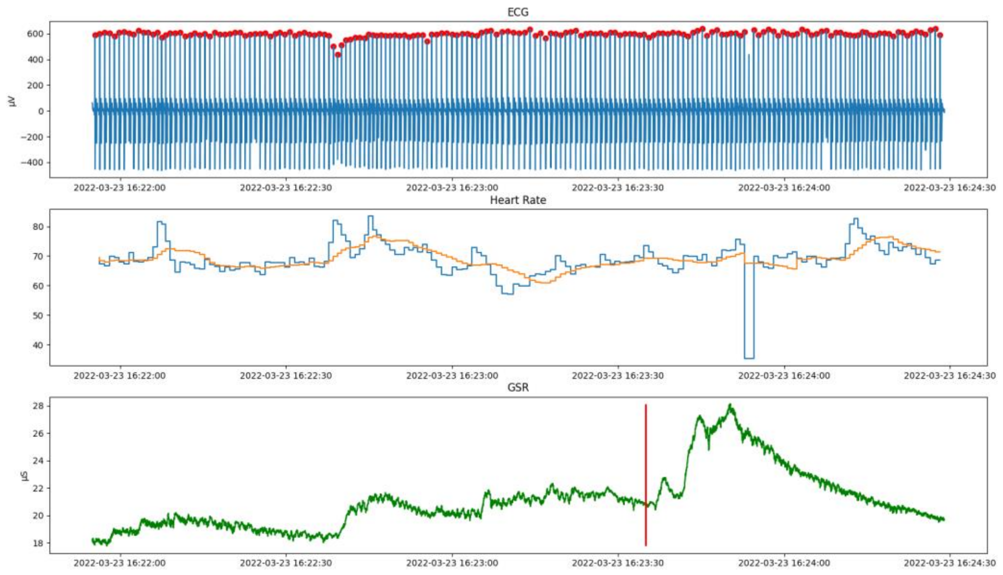


Figure 29 Scenario 1, Participant 2180

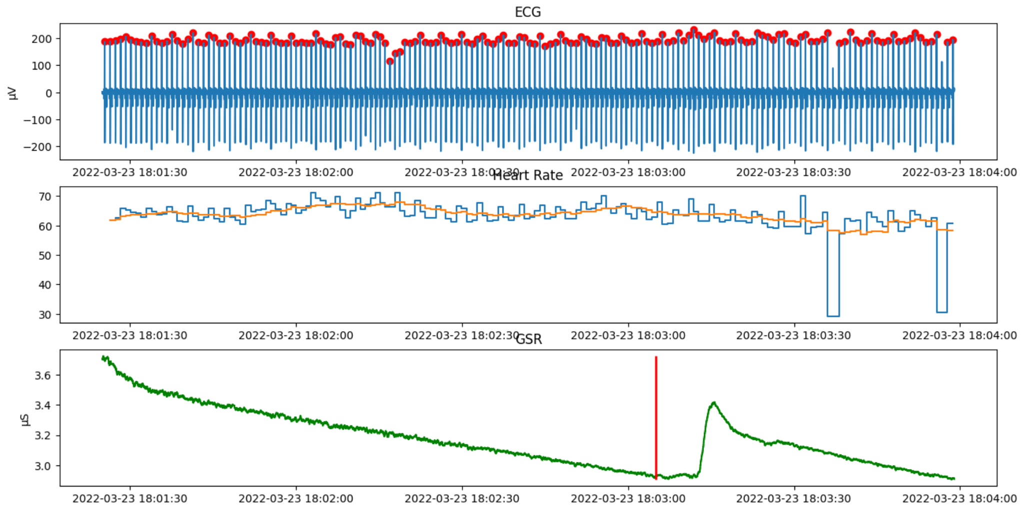


Figure 30 Scenario 2, Participant 2111

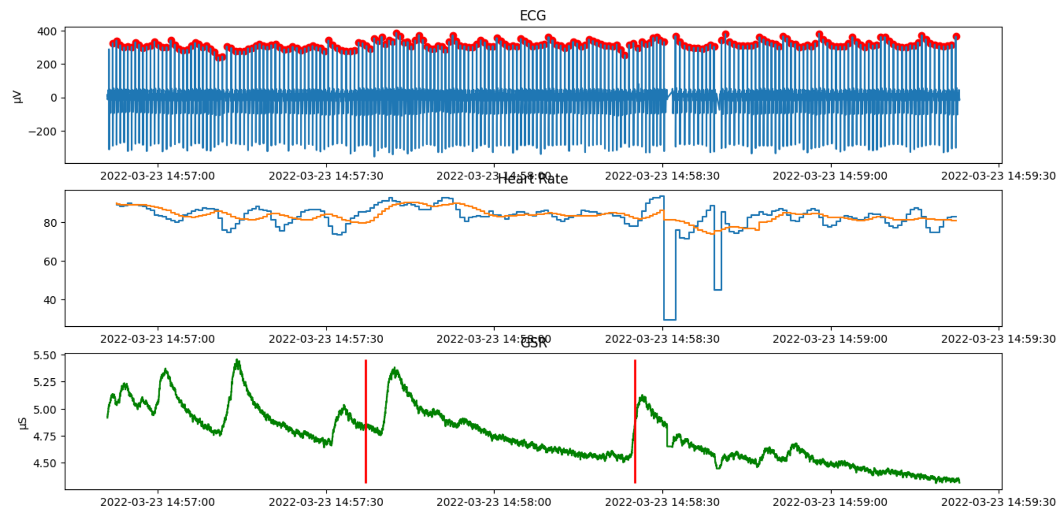


Figure 31 Scenario 3, Participant 2022

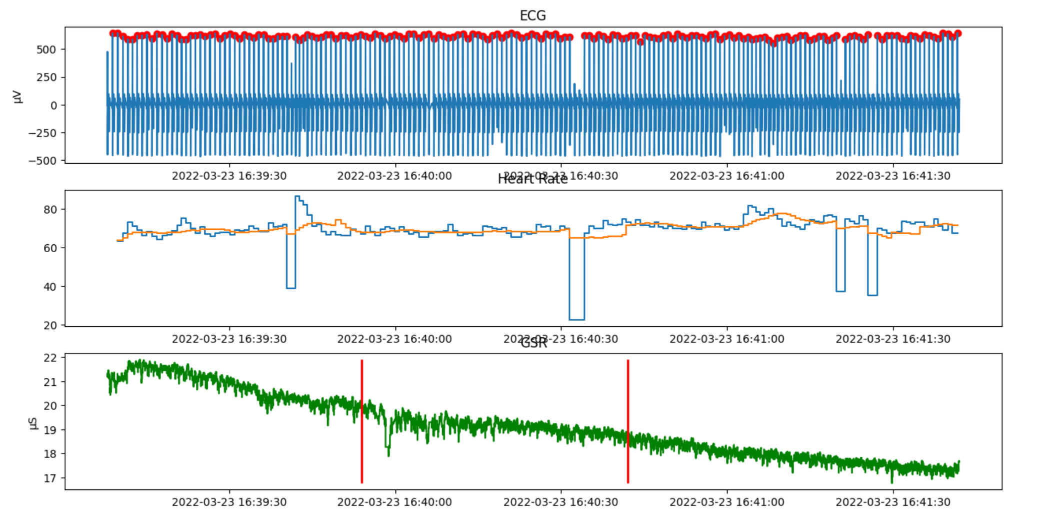


Figure 32 Scenario 4, Participant 2180

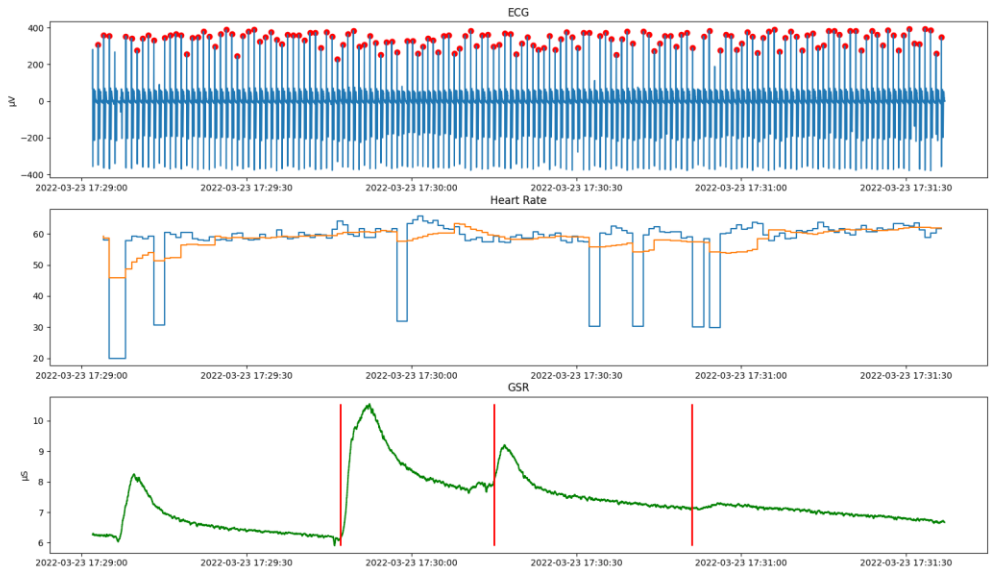


Figure 33 Scenario 5, Participant 2111

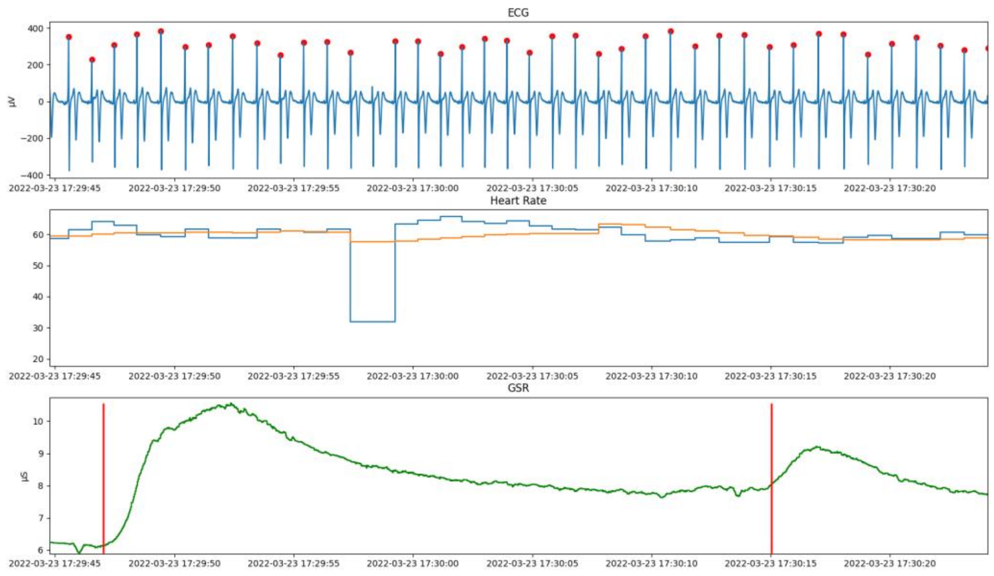


Figure 34 Scenario 5, Participant 2111 - Zoomed-in

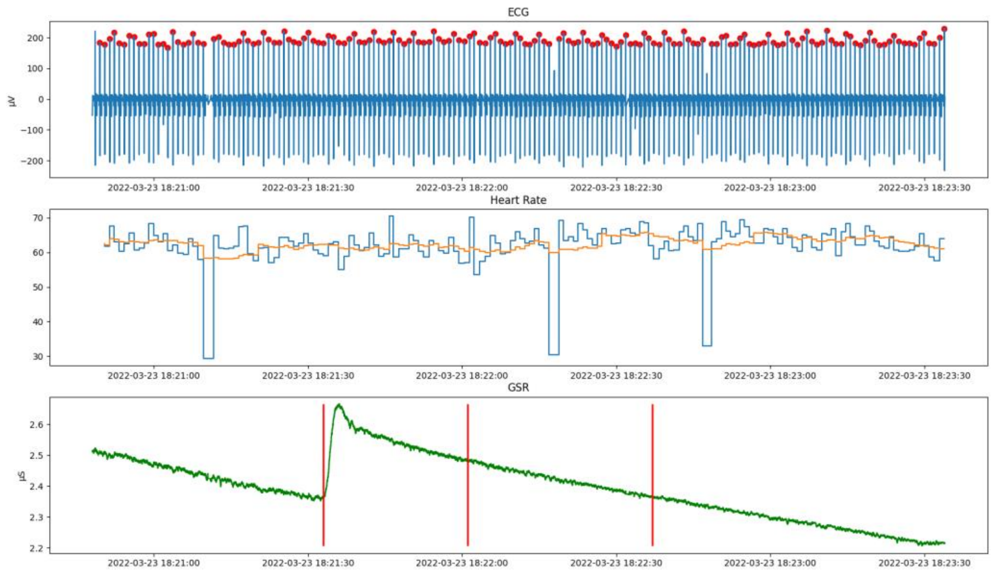


Figure 35 Scenario 6, Participant 2111

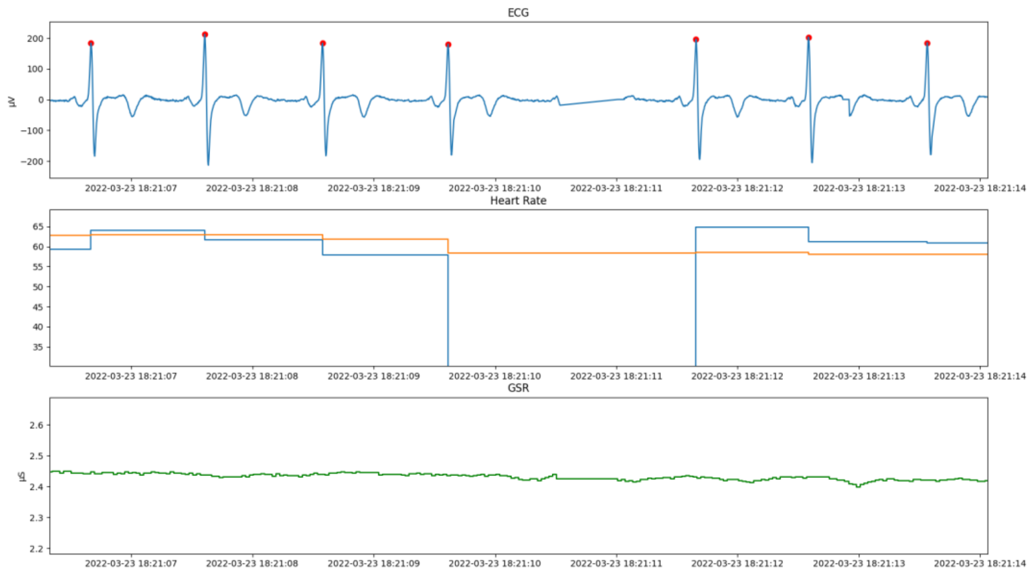


Figure 36 Scenario 6, Participant 2111 - Data Outage

As two different data groups are collected (subjective and objective), the key anticipated action is that of correlating the two, finding out what the key human parameters are (starting from stress and comfort) and then generating a human model that would allow prediction of human behaviour in specific driving scenarios. Such an outcome would on one hand enable improvements of the general vehicle control strategies and on other hand enable personalisation.

2.3 External data (open access)

In agreement with the intended preliminary design and development of the AI models, several publicly accessible datasets on learning problems are used. These have similar nature to those gathered in TEACHING use case scenarios. In general, they enable the execution of preliminary experiments useful for the following developments of the different AI algorithms. We report the main information on these datasets in the remainder of this section.

2.3.1 Datasets for Human state and activity monitoring

Concerning Human Activity Recognition and Monitoring five main datasets are employed. Each of those is extensively described below.

The main dataset of choice for the benchmarks is the Wearable Stress and Affection Detection (WESAD) dataset [19]. The data are gathered in a study on 15 subjects, which along with 36-minute sessions, performed activities depending on a desired cognitive state to be induced. Such state could be one of the following: *"baseline"*, denoting the resting conditions of the subject; *"amusement"; "stress";* and *"meditation"*, which is used in-between the amusement and stress conditions to reach a near-to-baseline state before engaging the next activity. The data were collected by two main devices worn by the subjects during the activities: a RespiBAN and Empatica E4 devices. The former is a chest-worn device which samples data for the electrocardiogram (ECG), electrodermal activity (EDA), electromyogram (EMG), respiration, body temperature and three-axis acceleration at 700Hz. The latter samples the Blood Volume Pulse (BVP) at 64Hz, the electrodermal activity (EDA) and the body temperature at 4Hz and the three-axis acceleration ad 32Hz. Each sample is labelled with the expected cognitive state corresponding to the activity performed by the subject at that time. As a result, the dataset is provided as a set of 15 disjoint time series, one for each subject, each of which presents the samples from the devices as features, and the expected cognitive state as a label. The data divided by the user makes it also suitable for simulating a federated learning scenario. From this dataset, we defined a binary classification task, consisting in predicting the stresses/unstressed user state based only on the RespiBAN data. In particular, a subset of 9 subjects is used as a training set, another subset of 3 subjects as a validation set, and the last 3 users as a test set. Two data perspectives were employed, depending on the adopted kind of data processing. In the first case, for each user, all the data is standardised. They are split into subsequences in a sliding window fashion. Then, for each user, the binary label is predicted for each of the subsequences. The second case uses data normalisation by computing the mean and standard deviation of the features of each subject only on the samples labelled as the *baseline*. The increase in performance in the second case validated the assumption that calibrating the standardisation only on the baseline condition allows better generalisation of the variations which happen in the other cognitive states, making the final model more independent of the user on which it was trained on.

The Heterogeneity Human Activity Recognition (HHAR) dataset contains inertial measurements recorded from 9 subjects performing physical activities [20]. The sensors used (gyroscopes and accelerometers) are part of smartphones or smartwatches which are carried or worn by the subjects (8 smartphones and 4 smartwatches). Each subject is requested to perform 6 different physical activities (without a particular order): standing, sitting, walking, waking up the stairs, walking down the stairs, and biking. The sensor data is labelled with the associated activity: these labels are employed to perform the classification.

The Physical Activity Monitoring (PAMAP2) dataset [21] contains physiological and inertial measurements recorded from a heart rate monitor and 3 inertial measurement units. Each of the 9 subjects is requested to perform 18 activities (of which 6 are optional) such as running, watching TV, car driving, and so on. For TEACHING purposes, the selected activities are the top 4 in terms of the corresponding number of participants. These are lying, sitting, standing, and walking, which have been used for a classification task.

The OPPORTUNITY Activity Recognition dataset [22] includes measurements from body-worn sensors (7 inertial measurement units, 12 3D acceleration sensors, 4 3D localization information) and other sensors associated with objects in a controlled environment. For TEACHING purposes, only the body-worn sensors are considered, in particular the 3D acceleration, 3D rate of turn, 3D magnetic field, and orientation of the sensors. The 4 participants are requested to perform several activities in different contexts (e.g., making coffee, eating a sandwich, cleaning a room), and the data are labelled according to four basic motor activities (standing, walking, lying, and sitting). These basic motor activities are employed in a classification task.

The ASCERTAIN dataset [23] has been built to support tasks of emotion and personality recognition from commercially available sensors. Signals include electrocardiogram, electroencephalogram, electrodermal activity, and facial activity, which were recorded while the subjects were watching affective movie clips (36 clips per subject). After each clip, the 58 subjects are required to give a rating of their state of arousal (indicating a value between 0 and 6) and valence (–3 to 3). These values are considered axes of the circumplex model and the quadrant is classified. This yields the following four classes: high arousal - high valence, high arousal - low valence, low arousal - high valence, and low arousal - low valence.

2.3.2 Datasets for anomaly detection in aerospace applications

Regarding anomaly detection in avionics applications, it was found useful to start the development of the AI models on a related problem concerning the aerospace application. To this purpose, two open-access datasets are used for anomaly detection. These are collected from *Incident Surprise, Anomaly* (ISA) reports from NASA, in which experts annotate unexpected events (anomalies) which could put at risk the operations of a spacecraft. Reports that are given particular attention are the Soil Moisture Active Passive (SMAP) satellite[1], and the Mars Science Laboratory (MSL) rover (known as Curiosity)[2].

Both spacecrafts gather telemetry data for two different purposes: the former measures the amount of water on Earth; the latter is seeking to find out if Mars has ever met the conditions to host small life forms. The data comes from real-life incident reports that are annotated with anomaly labels by domain experts. These reports are used to study unexpected events that can put at risk the operations that a spacecraft must perform.

The dataset consists of 82 telemetry channels identified with an alphanumerical ID which determines the channel type (e.g., P-1= First Power channel). Each channel is a matrix where the first dimension indicates the timesteps and the second dimension indicates the input features. In particular, the first feature represents the values of the telemetry, while the others are commands sent or received by the module delegated for gathering such data. All the numerical values are normalized to lie in the range (-1, 1), while commands are one-hot

[1] https://smap.jpl.nasa.gov/mission/description/

[2] https://mars.nasa.gov/msl/mission/overview/

encoded. There is neither information about the sampling frequency of the channel, nor the nature of the commands.

The dataset is provided with a training and test split, where the training set contains only a non-anomalous sequence, while the test set contains anomalous subsequences for evaluating anomaly detection methods.

2.3.3 Datasets for anomaly detection in cybersecurity applications

For the cybersecurity application, TEACHING provides an anomaly detection module based on Long Short-Term Memory Autoencoders. For the development and testing of the module, the UNSW-NB15 dataset is used[3], which is a computer network security dataset released in 2015. The dataset contains streaming data related to more than 1.5 million realistic network activities tagged as either normal or abnormal (attack). It comprises 49 features split into four categories: flow features (e.g., source and destination IP, port), basic features (e.g., packet count, protocols, duration), content features (e.g., packet/content size), time features (e.g., inter-packet arrival time), additional features engineered from all the previous ones, and finally the ground truth features declaring the attack category and a Boolean (0 for normal, 1 for attack). The attack entries are classified into nine categories. Our testing setup involves data preparation pipelines including label encoding, one-hot encoding, and normalization with min-max scaling.

2.3.4 Datasets for human-centric personalization

For the purposes of human-centric personalisation, an RL model is implemented. The model is trained and tested in a simulation environment, the Carla simulator in this case. For the TEACHING project, it is crucial to have a method that generates synthetic data for the driver state. The term driver state refers to the driver's feeling of comfort while experiencing a simulated route in the autonomous vehicle. The metrics that are used to quantify the driver's comfort levels are stress and excitement. The Carla simulator provides the whole world state needed for the RL scenario, which comprises metrics about the vehicle state such as velocity, angular velocity, acceleration, etc., but does not provide anything related to the driver state. To tackle that issue, the normal skew and logistic functions are used to create three types of drivers that describe their preferred way of driving: (i) cautious, (ii) normal, and (iii) aggressive. The functions implemented are using as input the linear and angular velocity, which are closely correlated with how a driver perceives a route in the autonomous vehicle. To achieve diversity among the different drivers, the special parameters of the two functions had to be tweaked. Finally, a black and white image of the future route of the vehicle in the next 100 meters is employed as the third input to describe the forthcoming route state. This image was generated using the route planner of Carla but in a real scenario, and can be easily reproduced using the navigation system of the vehicle. The set of the world and driver states along with the image are the inputs of the RL model. The advantage actor-critic algorithm is used in TEACHING experiments, which has been commonly used in the last few years in RL research and offers very reliable results. Also, a modified version of the default Carla autopilot system is utilised to enable the control of how the autopilot chooses the target speed of the vehicle. The output of the RL was the decision for speeding up or down by 1 km/hour and that was transformed into commands sent back to the autopilot.

[3] https://research.unsw.edu.au/projects/unsw-nb15-dataset

3 Contribution to Milestones

This deliverable is the only one that is planned as a verification means for milestones M3 (*First dataset for AI training and use case design available*) and M4 (*First release of the TEACHING platform with core functionalities*).

3.1 Milestone 3: Dataset for AI models training available

Section 2 of this document describes the data generation for both use-cases and as such is used as a verification means for milestone 3.

It must be noted that the automotive data set (that is generated using the driving simulation study of Section 2.2) will be provided to the partners under the condition that GDPR is satisfied, but will not be made openly available to safeguard the personal data per article nine of the GDPR, as stated in Section 2.3.4. However, this data set is destined for WP4, the outcome of which is to be returned to WP5 for verification of the generated algorithms.

3.2 Milestone 4: Teaching core technology integrated in use cases setup

The TEACHING platform[4], design and principles of which will be detailed in D1.3 and D4.3., consists of a parallel and distributed micro-services architecture based on docker-compose following the general design presented in D4.2 and D1.2 and presented in Figure 37. A TEACHING application can be defined as a docker-compose configuration file. More user-friendly graphical user interfaces may be constructed to make such a process easier for users without any knowledge of software programming.

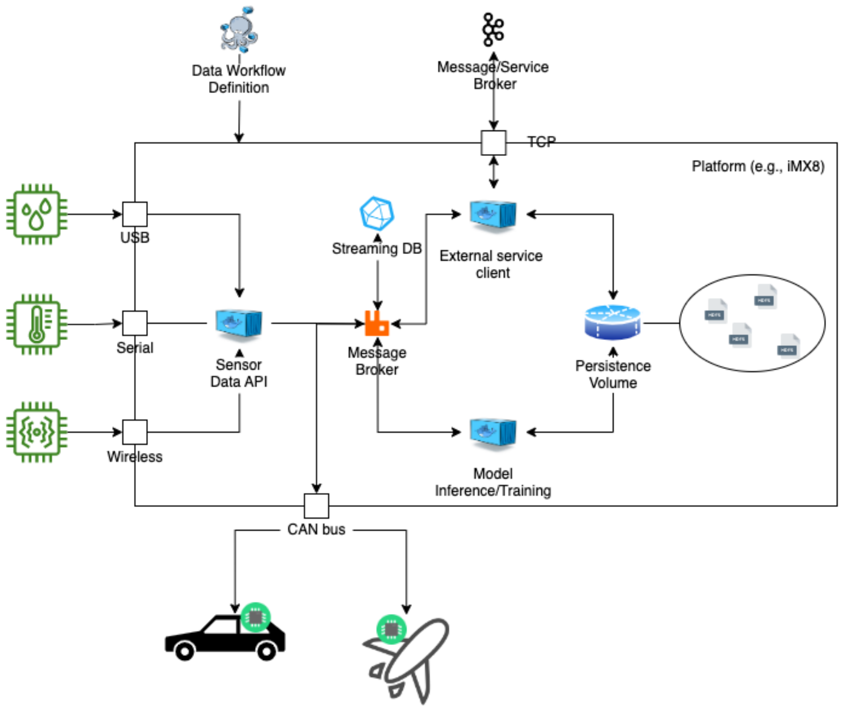


Figure 37: Main logical components design of the TEACHING platform

[4]The platform is currently being developed as 7 different private repositories within a GitHub organization related to EU TEACHING 2020. Such organization is providing faster release cycles and will open the path to a general open and accessible development of the library in the future.

The application logic is defined at the docker-compose level with a computational graph connecting several nodes (docker images) and arcs (communications link). Nodes can be producers, consumers or both and communicate via RabbitMQ. Following the INPUT_TOPIC(s) and OUTPUT_TOPIC(s) variables (often hardcoded within each node) is possible to understand the graph structure and information flow.

As a special case, the workflow can include external services through a TCP/IP and/or a CAN bus port (or similar). These ports facilitate the interaction with web-based services and external co-located systems respectively.

Nevertheless, every node is agnostic with respect to the graph composition and technology used in other nodes. Every node only implements a single function and processes each *DataPacket* (a JSON-based object) made available through the INPUT_TOPIC(s).

An example TEACHING application is depicted in Figure 38.

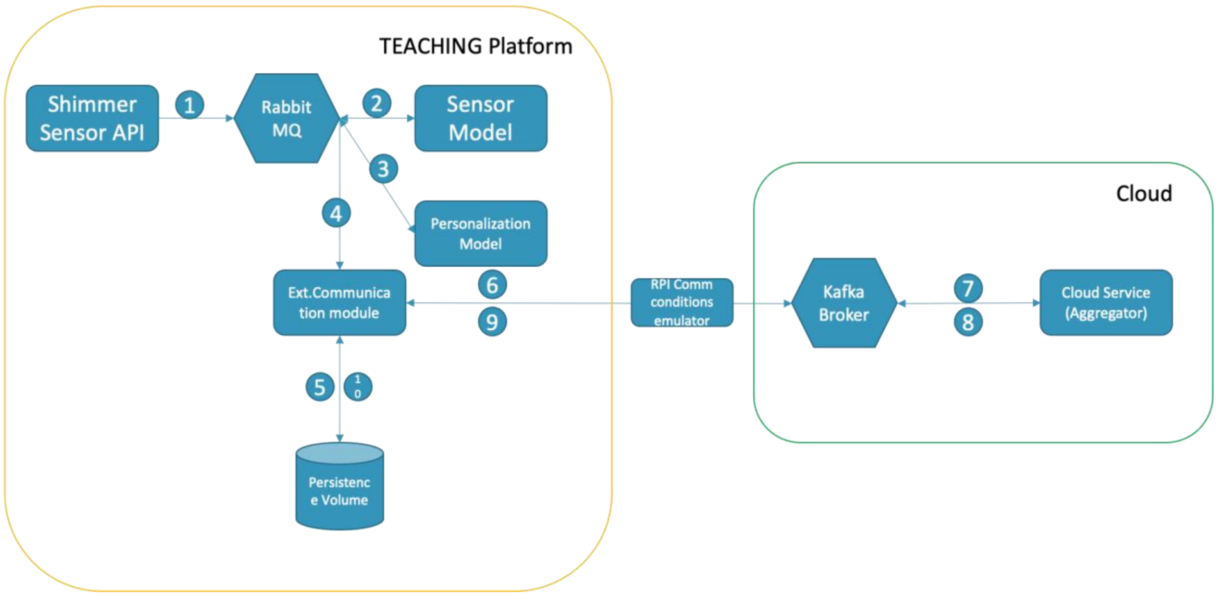


Figure 38: TEACHING demo Application

The workflow depicts an application that receives data from a wearable sensor and uses it to adapt a vehicle's driving style to the preferences of the human passenger. Since the model that personalises the driving experience is continuously learning, the application employs the knowledge as it has been accumulated in the personalization models of all connected vehicles' passengers and aggregates the model weights to achieve some sort of "transfer learning".

3.2.1 High-Performance Computing and Communication Infrastructure (HPC2I) as integrated within the use cases.

The different architectural layers of the SW platform described so far support TEACHING applications and are implemented by the project WPs. Within WP2, the focus of the design, research and implementation effort is on providing the adaptation and management layer for the distributed learning platform, allowing the components of the learning platform to exploit both custom HW resources and Cloud services. The conflation of the resource layers and the software artefacts that provide them interoperation constitutes the High-Performance Computing and Communication Infrastructure (HPC2I) platform. The abstract view of the platform and the corresponding activities of specific WP2 tasks have been already described in previous deliverables, specifically D2.2[5] sections 3 and 4 provide a full description, including the deployment of pub/sub overlay networks, the porting of computation-oriented and performance measuring frameworks. For the sake of brevity and relevance to the present document, here we only discuss recent contributions of WP2 supporting the use case implementation in WP5. The HPC2I platform can be seen as supporting both sides of the example application shown in Figure 38, where the left-hand part shows components deployed on the vehicle, while the right-hand part of the figure (the "Cloud" square) depicts some of the services deployed and provided by HPC2I on top of the whole set of its devices.

Computation and performance. Concerning the computational capabilities, substantial work has been put into the exploitation of specific device features in the different resource layers of the HPC2I platform, specifically of heterogeneous accelerators (CPU, GPU and FPGA) that support high performance streaming content analysis on high-end Cloud resources as well as on low-power, embedded edge devices. The Windflow[6] library is ported to Edge-class resources (e.g. NVIDIA Jetsons devices) and validated on common application schemes to vertically provide stream mining capabilities on the whole HPC2I platform. The significant effort there went into optimising the memory usage (zero-copy operations, buffer reuse) on the resource- and power-constrained devices of the Edge layers. Windflow has also been extended with new operators as well as with source and sink capabilities that allow connection to the pub/sub overlays adopted within the HPC2I (rabbitMQ and Kafka), paving the way for its integration within the use cases. The contributions related to the aspect of measuring performance metrics on both real-time OSes and Linux systems (the METrICS library), which are a key necessity for the use cases, were already reported in D2.2.

Networking and communication. From the viewpoint of the communication support, the first protocol for Federated Learning (FL) suitable for the network infrastructure is developed within WP2, following the Mobile Edge Computing (MEC) overall model and addressing the problem of client selection. In the FL process, a balance needs to be achieved between higher accuracy (a larger number of clients can generate a more accurate shared global model) and the amount of data required to update the global model. Overflows and instability of the data queues may result at the edge of the network infrastructure, but they can be prevented by selecting a subset of the clients to send their models. A software component for the HPC2I platform is implemented to allow the simulation of vehicular communication scenarios and numerically analyse the above protocols. Different communication setup and vehicular parameters can be simulated, such as packet loss, latency, vehicle speed and vehicle density.

Security. As a contribution to the security aspects of networking in the use cases, the Model Transfer Service (MTS) is developed to secure the transfer of the serialised knowledge models between the edge and the cloud. This is a bidirectional process needed to support FL methods, and interfaces on the edge, where serialised models (HDF5 files) are received via a RabbitMQ file path, encrypted, and sent over Kafka. On the cloud side, messages received by the Kafka broker are decrypted to local storage to be further processed. The component also provides secure, encrypted transmission in the opposite direction, from the Cloud-based services back to the Edge node local RabbitMQ service. The MTS microservice is deployed in the iMx8 board at the edge as well as on near-Edge and Cloud computing nodes.

Scalability. Exploiting the MTS and the existing pub/sub-networks an improved generic system performing model aggregation and distributed computation, the Model Aggregation

[5] D2.2: Refined requirement specifications and preliminary release of the computing and communication platform – August'21

[6] https://github.com/ParaGroup/WindFlow

Service (MAS)[7], is developed. The MAS can provide greater platform scalability as well as increased flexibility by interfacing the HPC2I services with the AIaaS technology, the stream mining technology and other custom software modules.

Orchestration and coordination. The HPC2I platform is responsible for linking Far-Edge CPS on vehicles with a hierarchy of services deployed on Near Edge and Cloud resources, as well as coordinating application component instances over the whole platform. While the applied industry-standard techniques mentioned in D2.1 and D2.2 (e.g., Docker and Kubernetes containers) are a sound choice for the implementation of the platform components, they do not exhaust the issues posed by the overall TEACHING settings. In-vehicle deployment is likely to remain quite static, being constrained by security and privacy considerations. Service deployment on all other layers of the platform on the other hand needs to consider network features (latency, bandwidth, reliability) as one end of the placement trade-off with computation cost and performance. Techniques for minimizing specific metrics of the distributed CPSoS are being applied to the deployment and orchestration of services within the HPC2I platform.

3.2.2 AI-as-a-Service core technology integrated into the use cases setup

The AI-Toolkit[8] collects and implements the AI nodes (also called *learning modules*) already made available to define an arbitrary complex TEACHING application. Such processing modules can be used for model inference (prediction) and training (or finetuning) or just for pre-processing/synchronising data. The AI-Toolkit is implemented to be agnostic with respect to the AI framework used as long as it offers a Python API interface. However, using a different programming language or dependency is definitely possible based on the strong dockerisation imposed by the platform.

The basic documentation with the current features is reported below. This is already implemented and available within the TEACHING platform, together with how to use them within a demo application.

Base Python Classes

At the moment the AI-Toolkit offers the following base classes:

- **LearningModule**: a Python class every learning module node can inherit from. It implements the basic API of a learning module node (both for inference/eval and training).
- **ESN**: general Echo State Network (ESN) for sequential classification implemented in Tensorflow.

Currently Available Nodes

At the moment the AI-Toolkit offers:

- **StressModule**: learning module for the stress prediction based on electrodermal activity, implemented in Tensorflow.
  - *input topics*: *defined at the docker.compose level*
  - *output topics*: "prediction.stress.value"
  - *assumptions*: it assumes each DataPacket processed has "eda" in its JSON

[7] https://github.com/EU-TEACHING/teaching-model-aggregator

[8] https://github.com/EU-TEACHING/teaching-ai-toolkit

How to Use Nodes

Nodes can be instantiated at the docker.compose level (i.e. defining a new yaml file with the appropriate syntax: see the teaching-app repository for examples and further instructions). When adding a new node to the app, the *input topic* (i.e. the communication channel the node will listen to) needs to be defined there. The *output topic* is hardcoded into the node logic (and listed in the section above for easy reference).

When the TEACHING app starts, the node `__init__` method will be run, followed by the `_build` method. The `__call__` implements a Python *generator*: this means that if the node is a producer, it will *yield* DataPackage downstream; if it is a consumer it will loop over an *input_fn* (another generator) defined upstream in the computational graph.

How to Implement Custom Nodes

In order to implement a new learning/eval module, it is possible to take the *StressModule* as a reference. Overall, it is needed to define the three functions mentioned before: `__init__`, `_build` (can be void) and `__call__`. The only aspect that deserves a little attention is how to implement the `__call__` method. First, it is important to think well if the node is a producer, a consumer, or both. This should be reflected in the decorator `@TEACHINGNode(produce=True, consume=True)`. If `consume=True` then the `__call__` method should have an input parameter `input_fn` that we can loop over to get new DataPackets from nodes upstream. If `produce=True` then the `__call__` method needs to *yield* a DataPacket: this will be downstreamed automatically to the (eventual) consumer nodes.

Debug/Test Mode

First of all, it is needed to set up the environment. If using Windows, it is possible to install WSL2 and Docker Desktop. Then, within Docker settings, it is necessary to make sure to check the WSL integration tab and connect it to the virtual image. Once this is done, it is possible to run docker-compose directly within the WSL image (e.g. Ubuntu based). Alternatively, it is possible to install and run everything from Windows powershell directly.

In order to debug the learning module, it is suggested to implement a `__main__` method within the learning module script and to momentarily disable the `@TEACHINGNode(produce=True, consume=True)` decorator (again, you can take the *StressModule* as a reference). This way it is possible to see if the method is working as expected by implementing a debug script (see for example debug.py). After that, it is possible to restore the decorator of the `__call__` method and implement a simple scenario to test if the node is working within the teaching platform (see scenario_1.yaml as a reference).

Once **myscenario.yaml** has been defined, it is possible to run it simply by running the following lines of code:

```
git clone --recurse-submodules https://github.com/EU-TEACHING/teaching-app
cd teaching-app
mv myscenario.yaml scenarios/
docker-compose -f scenarios/myscenario.yaml up
```

4 Discussion and Conclusion

Section 2.1 presented the avionics use-case composed of the safety and security-critical Flight Management System (FMS) and the Cyber-BlackBox (CBB) monitoring software that should provide input trace of PMC values representative of the behaviour of the critical software on the hardware to the AI models of WP4. The objective of the AI models is twofold: 1) To detect deviation from expected/learnt behaviour and use this information to detect either safety issues or security threats. 2) provide a smart way/order to scan the PMC while performing online detection as we can only capture 6 counters at a given time on the target hardware architecture.

Section 2.2 showcases the data taking activities for the automotive use case. While the technology blocks of TEACHING are being tailored for integration, it is required to further optimise these to ease their full integration into the system. As the concepts are solidified, the path to exploitation will be eased and sped up. The key at this stage is to allow customisation and adequate responses to the needs of vehicle occupants.

While the driving simulator studies are performed on regular basis, this dynamic advanced driving simulator is deployed using a specific set of driving scenarios to satisfy the needs of the TEACHING project. Two points could always be argued and the first, a more general one, concerns the usage of driving simulators and their deviations from the real driving situation. While this is a valid point, efforts are made to bring the driving simulator experience as close as possible to the real driving while continuing to safeguard the safety of the vehicle occupant. There is no better option at the moment to perform this kind of work. The second point about the validity of the study is related to the choice of driving scenarios. Scenarios could always be improved. The target of the scenarios that are specifically built for this study is to entice user responses. The initial data checks suggest that TEACHING is on the right course to achieve that target. Further detailed data analysis is expected to confirm that view.

Ultimately, this deliverable makes available several important datasets for the next stages of development, training, and evaluation of the AI models designed within the activities of WP4. According to the risk management plan, these activities so far have been based on publicly available benchmark datasets from literature (also briefly summarized in this document), which drove the design and development of the first integrated release of the AI toolkit (Milestone MS4). In this regard, the outcomes of the data gathering process described in this deliverable represent a decisive step through the full integration and validation of the AI toolkit in the use cases setup (Phase 3 of the project, towards Deliverable D4.3 and Milestones MS5-6).

While the progress is made and the data is being made available for further scrutiny and improved modelling, the further project outcomes are to be used to confirm the initial plans for development and are to also pave the path for sustainable exploitation of the TEACHING assets.

5 References

[1] S. Girbal, J. Le Rhun and H. Saoud, "METrICS: a Measurement Environment for Multi-Core Time Critical Systems," in *Embedded Real Time Software and Systems (ERTS'18)*, Toulouse France, 2018.

[2] Radio Technical Commission for Aeronautics (RTCA) and EURopean Organisation for Civil Aviation Equipment (EUROCAE), *DO-254: Hardware Considerations in Airborne Systems and Equipment Certification,* 1992.

[3] Radio Technical Commission for Aeronautics (RTCA) and EURopean Organisation for Civil Aviation Equipment (EUROCAE), *DO-178B: Software Considerations in Airborne Systems and Equipment Certification,* 1992.

[4] Radio Technical Commission for Aeronautics (RTCA) and EURopean Organisation for Civil Aviation Equipment (EUROCAE), *DO-297: Software, Electronic, Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations.*

[5] R. Fuchsec, "PikeOS Multi-Core Features and CAST-32A Compliance," SYSGO. Whitepaper, 2019.

[6] C. Hobbs, Embedded software development for safety-critical systems, CRC Press, 2019.

[7] E. Biermann, E. Cloete and L. M. Venter, "A comparison of Intrusion Detection Systems," *Journal on Computers & Security, Elsevier,* 2001.

[8] R. Strackx, F. Piessens and B. Preneel, "Efficient isolation of trusted subsystems in embedded systems," in *International Conference on Security and Privacy in Communication Systems,* 2010.

[9] S. Girbal, X. Jean, J. Le Rhun, D. G. Pérez and M. Gatti, "Deterministic Platform Software for Hard Real-Time systems using multi-core COTS," in *Proceedings of the 34th Digital Avionics Systems Conference (DASC),* Prague, 2015.

[10] S. Mekid, "IoT for health and usage monitoring systems: mitigating consequences in manufacturing under CBM," in *18th IEEE International Multi-Conference on Systems, Signals & Devices (SSD),* 2021.

[11] S. C. A. Longari, M. Carminati and S. Zanero, "A secure-by-design framework for automotive on-board network risk analysis," in *IEEE Vehicular Networking Conference (VNC),* 2019.

[12] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, A. Fogh, J. Horn and S. Mangar, "Meltdown: Reading Kernel Memory from User Space," in *27th USENIX Security Symposium,* Baltimore, 2018.

[13] S. Girbal, J. Le Rhun, D. G. P´erez and D. Faura, "Safety & Security monitoring convergence at the dawn of open hardware," in *Embedded Real Time Software and Systems (ERTS'22),* Toulouse France, 2022.

[14] ARM corporation, *ARM® Cortex®-A53 MPCore Processor - Technical Reference Manual revision R0P4,* 2016.

[15] ARM corporation, *ARM® Cortex®-A53 MPCore Processor - Technical Reference Manual revision R0P3,* 2016.

[16] P. Clement, H. Danzinger, O. Veledar, C. Koenczoel, G. Macher and A. Eichberger, “Measuring trust in automated driving using amulti-level approach to human factors,” in *In Print (Euromicro DSD/SEAA Conference 2021),* Palermo, 2021.

[17] P. Clement, O. Veledar, C. Könczöl, H. Danzinger, M. Posch, A. Eichberger and G. Macher, “Enhancing Acceptance and Trust in Automated Driving trough Virtual Experience on a Driving Simulator,” *Energies,* vol. 15, no. 3, p. 781, 2022.

[18] “Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles J3016_202104,” SAE international, 2021.

[19] P. Schmidt, A. Reiss, R. Duerichen, C. Marberger and K. Van Laerhoven, “Introducing WESAD, a Multimodal Dataset for Wearable Stress and Affect Detection,” Association for Computing Machinery, Boulder, CO, USA, 2018.

[20] A. Stisen, H. Blunck, S. Bhattacharya, T. S. Prentow, M. B. Kjaergaard, A. Dey, T. Sonne and M. M. Jensen, “Smart devices are different: assessing and mitigatingmobile sensing heterogeneities for activity recognition,” in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems, pp. 127–140,* New York, 2015.

[21] A. Reiss and D. Stricker, “Introducing a new benchmarked dataset for activity monitoring,” IEEE Computer Society, ISWC, pp. 108–109, 2012.

[22] D. Roggen, A. Calatroni, L.-V. Nguyen-Dinh, R. Chavarriaga, H. Sagha and S. T. Digumarti, “Collecting complex activity data sets in highly rich networked sensor environments,” Seventh International Conference on Networked Sensing Systems, Kassel, Germany, 2010.

[23] R. Subramanian, J. Wache, M. K. Abadi, R. L. Vieriu, S. Winkler and N. Sebe, “ASCERTAIN: Emotion and Personality Recognition Using Commercial Sensors,” *IEEE Transactions on Affective Computing,* vol. 9, no. 2, pp. 147-160, 2018.

[24] P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp and S. Mangard, “Spectre Attacks: Exploiting Speculative Execution,” in *40th IEEE Symposium on Security and Privacy (S&P'19),* 2019.