Project Report

Autonomous Trustworthy Monitoring and Diagnosis of CubeSat Health (AtMonSat)

University of Luxembourg

Faculty of Science, Technology, and Medicine (FSTM)

November 17, 2022

List of contributors

Ross Horne

Department of Computer Science
Faculty of Science, Technology, and Medicine (FSTM)
University of Luxembourg

Sjouke Mauw

Department of Computer Science
Faculty of Science, Technology, and Medicine (FSTM)
University of Luxembourg

Andrzej Mizera *(Principal Investigator)*

Department of Computer Science
Faculty of Science, Technology, and Medicine (FSTM)
University of Luxembourg

André Stemper

Faculty of Science, Technology, and Medicine (FSTM)
University of Luxembourg

Jan Thoemel

CubeSat Laboratory
Interdisciplinary Centre for Security, Reliability and Trust (SnT)
University of Luxembourg

Abstract

This document is the final report concluding the execution of the AtMonSat project co-funded by the European Space Agency (ESA) under the Open Space Innovation Paltform (OSIP) and the University of Luxembourg. AtMonSat concerns on-board fault detection using artificial neural networks for CubeSat systems and related spacecraft where computing resources are limited. In particular, the concrete problem scenario of malfunctioning of CubeSat board elements is considered. The AtMonSat final report provides the problem statement, discusses the performed experiments designed to generate proper sets of data, and presents the details of the proposed solution. The report shows the devised framework to be both effective and suitable for implementation on a CubeSat.

Acknowledgements

We thank Luis Mansilla and Christophe Honvault (the Technical Representative of ESA) at the Software Technology Section (TEC-SWT) of the ESA European Space Research and Technology Centre (ESTEC). Their monthly feedback, has helped guide the AtMonSat project towards relevant outcomes. ESTEC is located at Keplerlaan 1, 2201 AZ Noordwijk, The Netherlands.

The AtMonSat project was partly funded by the European Space Agency (ESA) under the Open Space Innovation Platform (OSIP), under which ESA organizes and launches Campaigns and Channels to seek ideas related to space research, otherwise known as ESAIdeas. For reference, this project relates to ESA Contract No. 400134792/21/NL/GLC/my, and ESA activity description No. I-2020-03332. More than 50% of funding was provided in kind by the Department of Computer Science in the Faculty of Science Technology and Medicine at the University of Luxembourg.

We thank Ines Crisostomo for overseeing contractual and administrative matters at the University of Luxembourg. We thank Gian Lorenzo Casini and Miguel Yagues Palazon for assisting with contractual and administrative matters on the side of ESA. We thank Patricia Conti, Director for Economic Affairs at the Luxembourg Space Agency, who handled the ESA OSIP budget for Luxembourg.

Contents

1 Introduction

This project concerns on-board fault detection using artificial neural networks for CubeSat systems and related spacecraft where computing resources are limited. CubeSat is a term coined to refer to a small satellite, which in its most basic form is in the shape of a 10 cm cube, namely 1U, or a small multiples thereof, e.g., 2U, 3U, and 6U. CubeSats are recommended to comply with the ISO standards defined within the CubeSat Design Specification [1, 2]. CubeSats play a notable role in the New Space Economy. These spacecrafts are systems of great interest to the industry, scientific community, and government agencies as affordable facilities providing capabilities for a broad set of activities. The great success of CubeSats can be largely attributed to the *"low cost and fast delivery"* paradigm they introduced to the space research [3]. Unfortunately, the new paradigm is also to be considered accountable for the high percentage of failed missions, due to the fact that CubeSat critical components are built with cheap materials and to the low-cost processes of production and verification. Furthermore, CubeSats either implement very limited Fault Detection, Isolation, and Recovery (FDIR) functionality or lack it completely.

The focus of the AtMonSat project is to improve fault detection on-board CubeSats, as a prime example of a low-cost mission in Low Earth Orbit (LEO). This focus is motivated by the following regulatory reasons.

Firstly, due to launch trends, there is increasing concern from Launching States about liability issues stemming from conjunctions in LEO. Since, under current international treaties, the Launching State itself is liable if a satellite originating from their territory is deemed to be at fault in a conjunction, there is a trend towards states limiting their liability by requiring that operators are covered by insurance. Indications are that this trend will grow.

Secondly, while the developments of AtMonSat are relevant to spacecraft in general, we prioritise CubeSats, since there is an explicit ISO standard [4] for CubeSats, which imposes fewer reliability requirements than normal for spacecraft. In fact, the ISO standard does not impose any requirements with respect to critical software reliability. This leaves a gap between the deregulation advocated by the engineering standard, intended to stimulate innovation, and the demands of an insurer. Insurers will increasingly require proof that steps have been taken to reduce the possibility of CubeSat failure resulting in creating a dangerous object in LEO, hence auditable verification and system health monitoring of CubeSats will likely become essential. By making the results of this project available in the public domain, we aim to stimulate innovation in the New Space industry by exploring the feasibility and effectiveness of introducing on-board artificial neural networks that may facilitate the regulatory compliance of CubeSat missions.

Thus the high-level objective of AtMonSat is the development of a framework for explainable on-board anomaly detection and system health monitoring of a CubeSat. Notice that in the context of the AtMonSat project, explainable is understood differently from the classical notion used in the field of Explainable Artificial Intelligence (XAI). Here by explainable we mean that the proposed solution provides evidence that a problem of a certain type occurred, i.e., one which induces disturbance to the pattern of

interrelations between thermal telemetry data captured by different sensors as explained in the description of the concrete problem scenario below. The explainability of our solution lies in the choice of the bearer of information on system health, i.e., the complex thermal pattern. A disturbance in this pattern provides evidence on the occurrence of an additional heat source due to malfunction of some component of a specific subsystem of the CubeSat.

The aim of our framework is to make CubeSats more reliable and auditable, thereby facilitating compliance with regulation imposed by Launching States. Importantly, the proposed solution does not compromise the low-cost and fast-delivery characteristics of small spacecrafts. There is also a practical reason for considering CubeSats, specifically that the project team had direct physical access to a CubeSat during the project, i.e., the EduSat of the CubeSat Lab at the University of Luxembourg. Thus insight gleaned may be translated to other space missions with similar resources and regulatory requirements. To ensure that such missions are targeted a requirement of this project is to ensure that solutions are implementable on a typical microcontroller, respecting their limited memory and computational power, while leaving space for control software.

More specifically, within the above broader objective, we have focused on a case study, explained next, that involves anomalies in temperature readings that fall out of the scope of simple hardwired threshold triggers, and instead demand multivariate analysis of multiple components. Such a multivariate analysis is suited to artificial neural networks.

Concrete problem scenario: Malfunctioning of CubeSat board elements. For this investigation, we have selected a narrow case study in order to obtain datasets for training, testing and evaluating possible on-board artificial neural network solutions. The scenario we investigate aims to capture potential malfunctioning of CubeSat board component(s) manifested in subtle distortions of the temperature pattern inherent to the operation of the satellite in the LEO. On the CubeSat that we used for the experiments, various board components are equipped with temperature sensors, i.e., four Maximum Power Point Trackers (MPPTs), four voltage converters, and one battery monitor, which are spread across the board and provide real-time measurements. The placements of the nine sensors are schematically shown in Figure 1.

We consider scenarios under which the malfunctioning of any element on the board results in an increase of its temperature. Although the malfunctioning element itself may not be equipped with a temperature sensor, its abnormal state generates additional heat that is transferred via the mechanisms of thermal conduction (mainly) and thermal radiation (to some limited extent) across the board. Since our solution is centred on analysing the complex thermal pattern, it is characterised by a comprehensive health monitoring capability in the sense that it is not limited to a particular CubeSat board component, but aimed at detecting anomalies in the operation of any of them. Yet, by focusing on such a specific scenario, we aim to be able to explain the occurrence of anomalies as appearance of additional heat sources on the CubeSat board.

Under normal conditions, a CubeSat orbits in LEO while executing tasks related to its nominal functioning. We can assume that these tasks are performed either constantly or

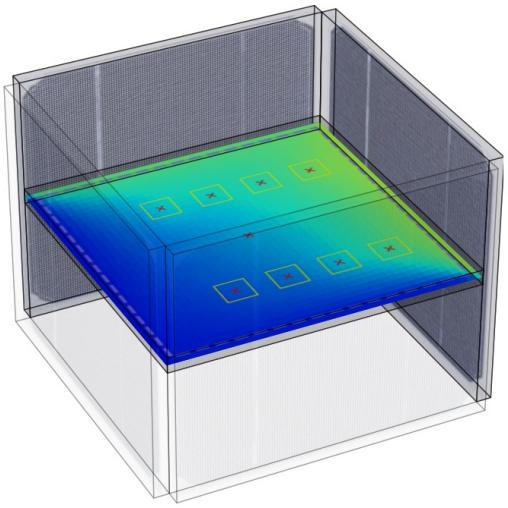


Figure 1: Schematic illustration of the placement of nine temperature sensors on the CubeSat board. The temperature sensors are indicated with red crosses framed with yellow squares.

periodically due to the immanent orbiting movement of the CubeSat around the Earth. Moreover, due to orbiting, the spacecraft is periodically illuminated by the sun light which introduces cyclic changes to the temperature values recorded by the sensors. Put together, it is justifiable to assume that there exists a periodic thermal pattern manifested in the relative timings of temperature value changes measured by individual sensors on the board. Under abnormal conditions, the additional heat is introducing disruptions to the characteristic thermal pattern. The solution proposed within AtMonSat is identifying subtle changes in the thermal pattern and reports them as plausible indicators of the malfunctioning of elements on the CubeSat board.

2 Project summary

Before providing details of the project, we present, in this section, an overview of the agreed milestones achieved.

2.1 Agreed milestones with completion dates

Task	**Time**	**Contrib.**	**Description**
Develop OS technology for the hardware-isolated interception of I/O data.	[M1-M3]	Mizera, Stemper, Horne	Done. Developed solutions for consuming on-board stream of telemetry from CAN bus of the EduSat. Telemetry assessed to identify interesting data for experiments. OS solution allowing monitor and control software to co-exist identified.

Formulation of the state-of-the-art AI techniques for anomaly detection and system health monitoring.	[M1-M6]	Mizera, Stemper	Done. Assembled candidate artificial neural networks for anomaly detection and system health monitoring, in particular LSTM recurrent neural networks, autoencoders, and CNNs. Models were trained with preliminary data and an idealised physics model of temperature variations during normal and failure modes. Induced anomalies were evaluated for their explainability. PCA was explored for feature extraction.
Construction and verification of adaptable and explainable solutions.	[M6-M15]	Mizera, Stemper	Done. Evaluations showed CNNs detected anomalies most effectively in the case of real lab data. A prototype was built and verified against different datasets and parametrisations. Higher quality data was extracted by experimenting with EduSat.
Evaluation and fine tuning with respect to the tumbling of CubeSats.	[M12-M15]	Mizera, Stemper	Done. The solution was benchmarked against a rule based approach, typically used. The model detects anomalous variations in temperature whilst orbiting. Quantisation of the model was found to be effective on synthetic data only.
Identification of recommended solutions for given types of missions.	[M16-M18]	Mizera, Stemper, Horne	Done. The on-board stream processing from M1-M3, and the trained CNN from M12-M15 were assembled for deployment. Memory usage, execution time, and power consumption were measured. Solution deployed using FreeRTOS suited to the on-board microcontrollers of many spacecrafts.

Notable deviations: We proposed to work on ESA PhiSat2 as the CubeSat. However, ESTEC took over from PhiLab as supervisors, so we used the EduSat of the University of Luxembourg. A refinement is that we considered finer temperature anomalies due to component failure rather than tumbling, since there exist established mechanisms for stabilising during tumbling. This provided anomalies that were clearly out-of-scope of rule-based AI, while being in scope of NN-based AI and possible to simulate in our lab environment.

3 Experimental data

We explain our methodology for generating data. Two types of data is considered. Firstly, there is the synthetic data we generated from a simulation of the diffusion of heat in a model of the CubeSat. Secondly, we generated data from an experimental setup using a

real CubeSat in a lab environment where environmental factors were minimised in order to simulate conditions in space.

3.1 Synthetic data generation

We explain the equations used to generate synthetic data. We also cover how the model is compared to the experimental setup, do justify its correctness, and how it was used to generate a dataset.

3.1.1 Physical model

We performed thermal simulation of a green printed circuit board (PCB). Since detailed modelling of a power system motherboard of a CubeSat would be challenging in itself, a PCB was considered as a largely simplified imitation of the actual motherboard. The purpose was to relatively effortlessly generate reproducible temperature sensor data with and without anomalies to allow us to train and evaluate anomaly detection models while experimental data from a CubeSat was still not available.

For synthetic data generation, we introduced a physical model of the PCB based on the diffusion equation

$$\frac{\partial T(x,y,t)}{\partial t} = \alpha \frac{\partial^2 T(x,y,t)}{\partial x^2} + \alpha \frac{\partial^2 T(x,y,t)}{\partial y^2} + \frac{g(x,y,t)}{\rho \cdot c_p} \tag{1}$$

with

$$\alpha = \frac{\kappa}{\rho \cdot c_p},$$

where κ is thermal conductivity, ρ is the density, and c_p is the specific heat capacity.

The simulation of the heat transfer across the PCB, taking into account the specific placement of heat sources as depicted in Figure 2, was based on solving the diffusion equation (Eq. 1) in 2D using the Finite-Difference Time-Domain (FDTD) method.

The synthetic data were to resemble the future data obtained with lab experiments and not in space environment. The rationale behind this decision was the fact that we did not have the possibility to acquire real-life, in-orbit data and all considerations of the AtMonSat project were necessarily limited to lab conditions. Therefore, besides radiation, the thermal simulation model additionally included surface convection, a phenomenon that does not occur in space.

For the solar panels, the simulation considered irradiance (angle, intensity) and radiative loss. The simulation allows specifying heat sources (yellow rectangles in Figure 2) that can be key-frame controlled or controlled by an external CSV file. Key-frame based control allows moving the position of the sources. The latent data for the different temperature sensors are sampled at the locations indicted by red crosses in Figure 2. These locations have been coarsely estimated to correspond to the positions of the temperature sensors of the MPPT converters, voltage converters, and the battery monitor

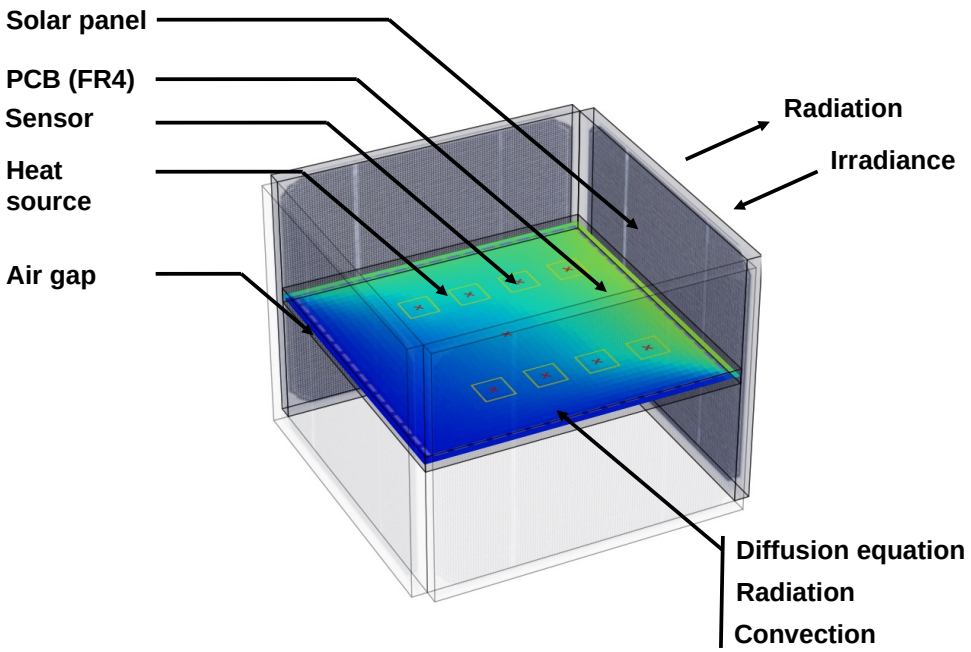


Figure 2: Simplified thermal simulation of an electrical power supply system of a Cubesat to generate quantized temperature sensor data.

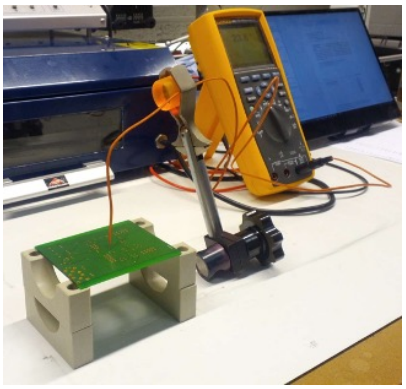
on the EduSat's electrical power supply board (specifications for the EduSat appear in Section 3.2.1). The latent data is then quantized to 1°C resolution. The source code of the simplified thermal simulator of an electrical power supply system of a CubeSat (EPSThermalSimulator) is available in a GitHub repository [5].

3.1.2 Experimental verification

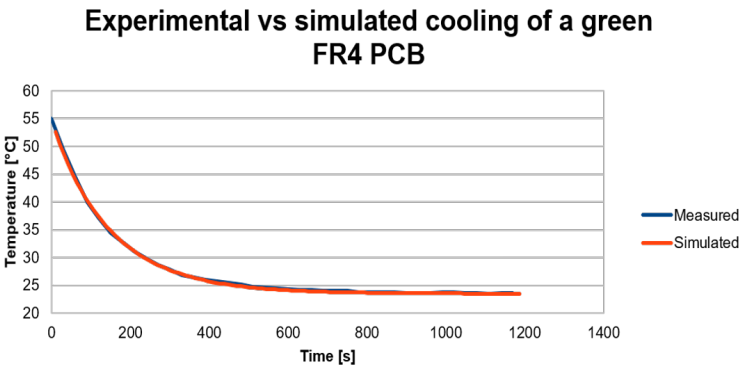
The parameter settings of the physical model were verified in two experiments: 1) Cooling of a PCB (FR4) and 2) Resistive heating. In order to check the correctness of the parameters by comparing the model outcomes with the reality, it was necessary to take the air gap into account, see Figure 2 for illustration.

In the Cooling of a PCB (FR4) experiment, the board was heated up to 55°C and then left for cooling down to room temperature, i.e., 23.5°C. The PCB was of dimension 0.08×0.065 [m^2]. Both radiation and convection were considered for the simulation with a 20×25 grid. The experimental setup and the obtained validation results are shown in Figure 3.

In the Resistive heating experiment, the FR4 PCB board was heated up by a 120 Ohms resistor powered by 100 mW (3.3 V, 30 mA). A temperature sensor was located next to the resistor. Both radiation and convection were considered for the simulation with a 30×35 grid. During the experiment, a strong dependence of the measurements on sensor contact (thermal paste) was observed. The experimental setup and the obtained validation results are shown in Figure 4.

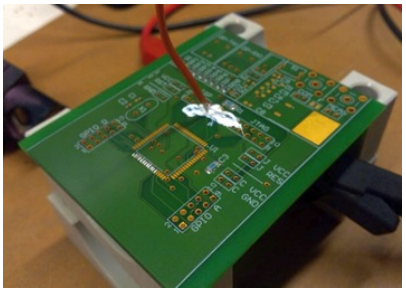


(a) Setup of the PCB cooling experiment.

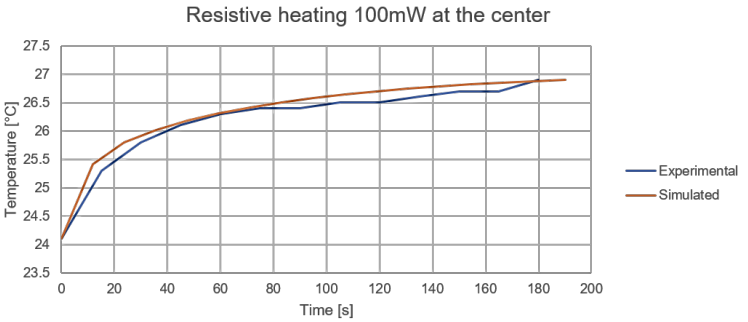


(b) Measured and simulated temperatures during FR4 PCB cooling.

Figure 3: Setup (a) and results (b) of the FR4 PCB cooling experiment.



(a) Setup of the PCB resistive heating experiment.



(b) Measured and simulated temperatures during FR4 PCB resistive heating.

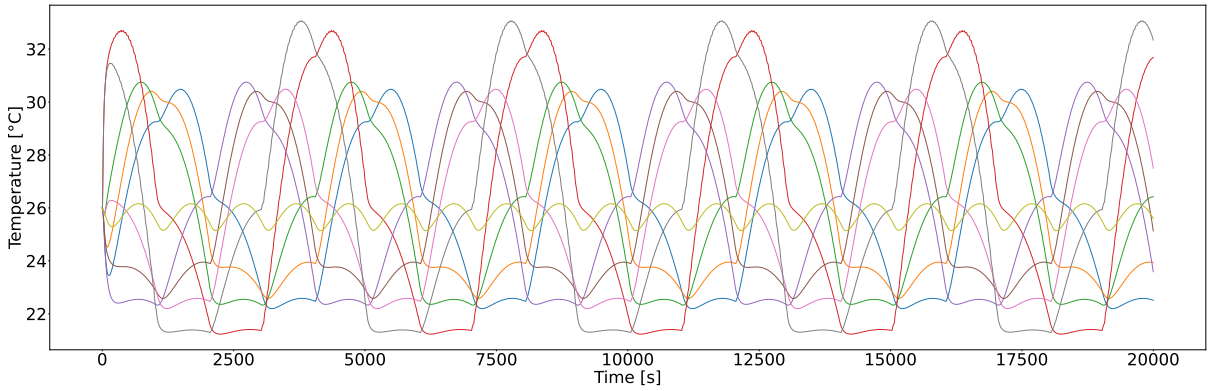
Figure 4: Setup (a) and results (b) of the FR4 PCB resistive heating experiment.

3.1.3 Simulated datasets

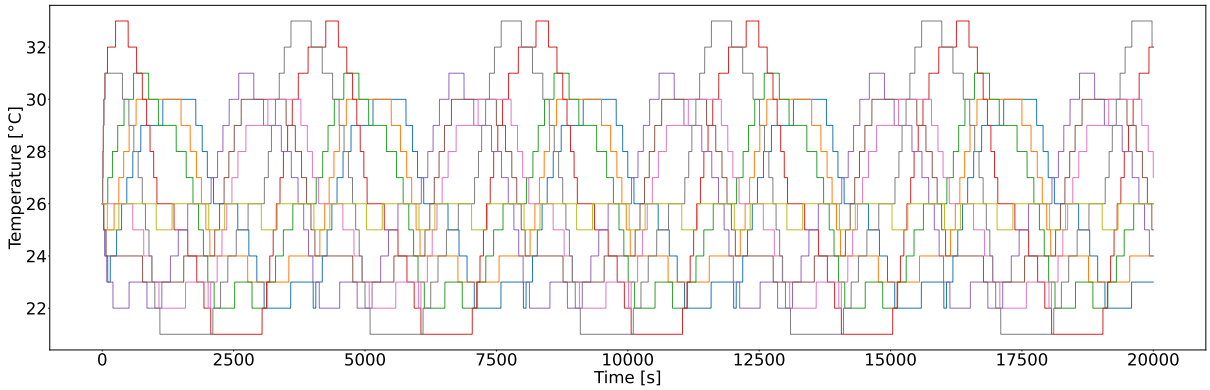
The simulator was used to generate temperature sensor data both for normal and abnormal conditions. Unfortunately, FDTD simulations turned out to be computationally heavy and, in order to generate longer temperature time series data within reasonable time frames, the air gap was excluded from consideration. Although this compromised the match with the reality, we decided to accept this fact since the important factor was to have some clean and regular data that would allow us to perform an initial pre-selection of candidate solutions for the detection of subtle anomalies. Anyway, the simulations with the air gap considered, were adequate for the PCB and not for the actual motherboard. Furthermore, the actual development of the proposed solution was conducted with lab data.

In normal conditions, the PCB is illuminated by a source circling around the PCB. Simulated temperature sensor data in normal conditions, both latent and discretised, are presented in Figure 5.

In the abnormal conditions, an additional heat source placed on the PCB of power 40 mW is turned on for 500 s. The simulated temperature sensor data in abnormal conditions with indicated regions of additional heating are shown in Figure 6.



(a) Latent simulated temperature sensor data in normal conditions.



(b) Discretised simulated temperature sensor data in normal conditions.

Figure 5: Simulated temperature sensor data in normal conditions.

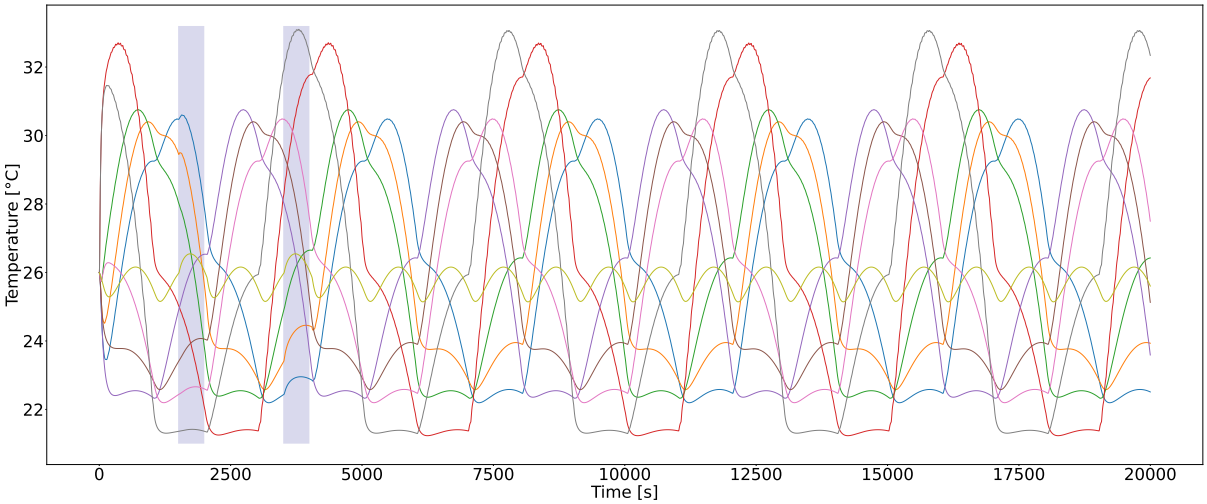
The simulated temperature sensor datasets were used in the initial phase of the project to allow us to train, evaluate, and test various anomaly detection candidate deep-learning models while experimental data from a CubeSat was still not available. Nevertheless, the synthetic data was characterised by perfect periodicity and no noise. This ideal regularity presented a representativeness gap that was noticed first when using the "real" lab generated data discussed next.

3.2 Lab data acquisition

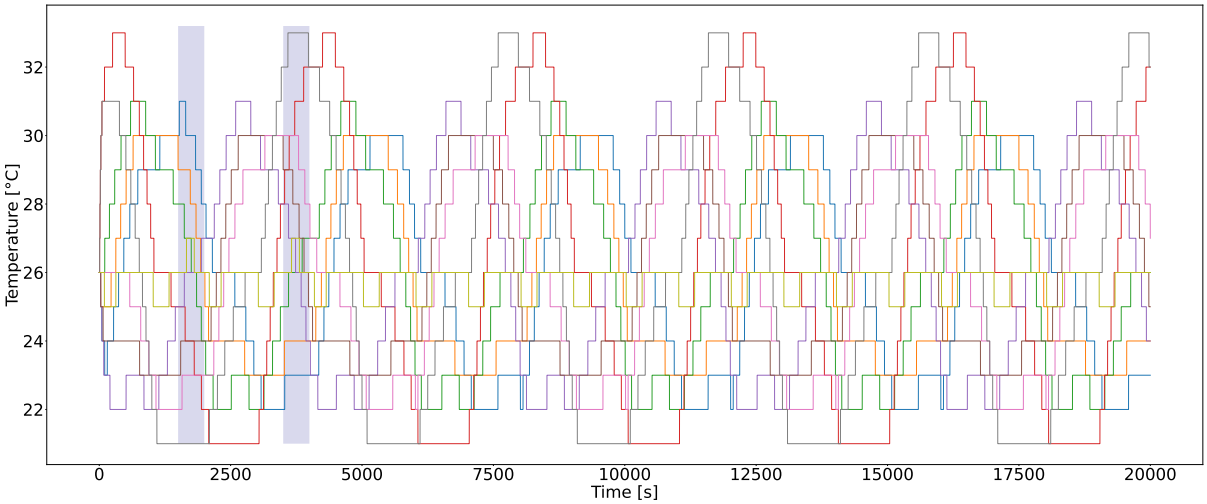
We explain in this section our experimental setup and how we gathered data using the CubeSat prototype supplied by NanoAvionics. We explain also how we limited the impact of environmental factors on the dataset so as to focus on the anomaly in question.

3.2.1 Experimental setup

Within AtMonSat an engineering model (EM) was used to generate telemetry data for analysis. It consists of a NanoAvionics near-flight prototype containing an UHF commu-



(a) Latent simulated temperature sensor data in abnormal conditions.



(b) Discretised simulated temperature sensor data in abnormal conditions.

Figure 6: Simulated temperature sensor data in abnormal conditions with two anomalies. The time of additional heating source being on is indicated with light-violet shading bars.

nications subsystem (COM), an electrical power supply system (EPS), attitude control sensor and actuators (ADC), and an on-board computer (CDH) with an attitude control algorithm. The EM, i.e., the EduSat of the CubeSat Lab at the University of Luxembourg, is depicted in Figure 7. It is complemented with a laboratory ground segment consisting of a radio and a personal computer. The system utilises the CubeSat protocol [6], which supports a distributed architecture where subsystem are addressable nodes. The EM is near-flight ready and hence features a high number of typical CubeSat sensors. Yet, the overall number of sensors and actuators, such as the number of solar panels, is reduced to enable cost-efficient education and research.

In order to simulate the in-orbit conditions of a CubeSat in LEO, the following experimental setup was devised. The EduSat was placed on a *rotation table*, which was configured to make one full rotation per 90 min. That period of the rotation represents a typical time for a satellite in LEO to complete a full orbit around the Earth. Dedicated

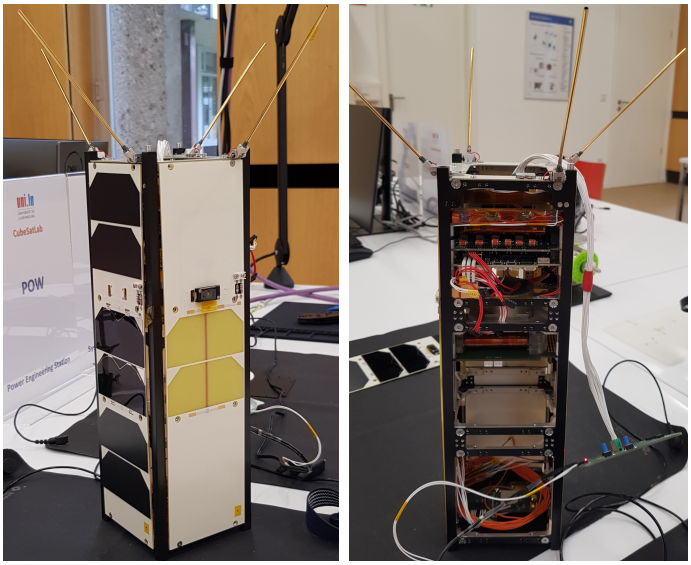


Figure 7: The EduSat (NanoAvionics EM) of the CubeSat Lab at the University of Luxembourg. The photo on the right presents the interior of the EduSat.

software was controlling the rotation table engine and logging the time and actual angular position of the EduSat.

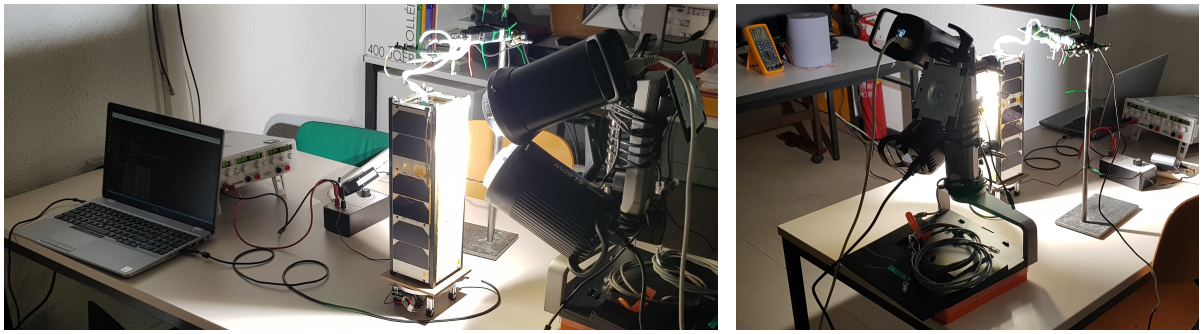
To mimic the periodical illumination of a CubeSat by sun light, EduSat was illuminated by two continuous, monolite-style light sources, i.e., Godox SL100Bi and Neewer SL-60W set to 80% brightness, 6500K and 100% brightness, 5600K, respectively. The light sources were focused at the height of the EduSat board. In order to keep the room temperature stable, the experiments were conducted in a basement room with highly limited sun light access and no other light sources. The experimental setup is shown in Figure 8. The room temperature was constantly monitored during experiments (see Figure 8b) and the temperature logs were saved.

For data acquisition, the EduSat was connected to a laptop via a USB-CAN converter attached to the EduSat's break-out board for data download. Since the board was linked to the EduSat with a cable, see Figure 8a, the rotation was causing twisting of the cable. To avoid any damage, the rotation table was quickly revolving back 360° after completing each full rotation and in this way untwisting the cable.

3.2.2 Datasets acquisition

A number of lab experiments mimicking the in-orbit conditions was performed with the EduSat to generate relevant telemetry data. For each experiment, telemetry data logged by the EduSat was collected. The data acquisition frequency was set either to 0.1 Hz or 0.2 Hz, corresponding to the recording of one telemetry datapoint per 10 s or 5 s, respectively. The EduSat buffer capacity was 1340 telemetry entries, so partial data was systematically downloaded and stored on a computer and later merged into a single telemetry data time series. The experiments consisted of three phases.

1. Phase I: during this initial phase the light sources were heating up the EduSat while the rotation was off in order for the EduSat to reach a stable experiment base temperature.



(a) (b)

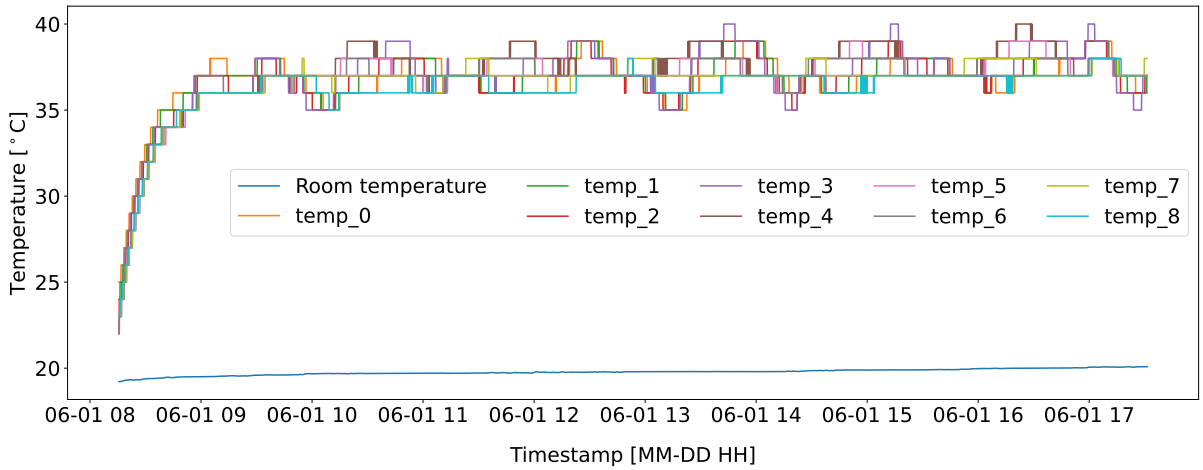
Figure 8: The experimental setup for data acquisition. (a) The EduSat is placed on the rotation table, which rotation speed is controlled by dedicated software run on the laptop. The two light sources illuminate the EduSat at the height of the EduSat board. The break-out board is connected on one end to the top of the EduSat with a white cable and on the other end via a CAN-USB converter to a laptop, the latter not visible in the photo. (b) The room temperature in monitored and logged by a Fluke digital multimeter visible in the top left corner.

2. Phase II: during it the EduSat was rotating while illuminated by the light sources and the normal conditions data were collected for a few rotations.
3. Phase III: during this final phase the anomalies were being introduced by turning on battery heating of specified power and duration which was generating abnormal conditions data.

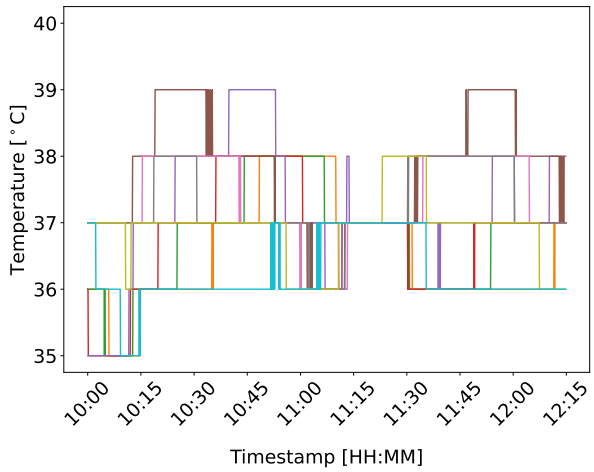
In order to simulate anomalies in Phase III, battery heating was used to imitate additional heating source(s) on the board which could appear due to malfunctioning of some board component(s). The power and duration of the battery heating was chosen in such a way that only subtle distortions to the thermal pattern were introduced which could not be spotted by visual inspection of the plots of the gathered time series data. Sufficient time was assured after each introduced anomaly for the EduSat to revert to the normal conditions before a subsequent anomaly was introduced or the experiment was ended.

For each experiment, the raw telemetry data portions downloaded from the EduSat were merged into a single time series. Next, data corresponding to Phase I were excluded and the remaining time series was split into to datasets: one containing the normal condition (nominal) data of Phase II and the other containing the abnormal (also refereed to as anomalous) data of Phase III. For some experiments Phase II or Phase III could be missing, leading to generation of only abnormal or normal dataset, respectively. Examples of normal and abnormal datasets from an experiment are presented in Figure 9, where each plot shows the readings of the nine EduSat board temperature sensors in time. The resolution of the temperature sensors was 1°C. Table 2 provides summary information on the individual experiments, which datasets were used for the training, evaluation, and test of the final model of the AtMonSat project.

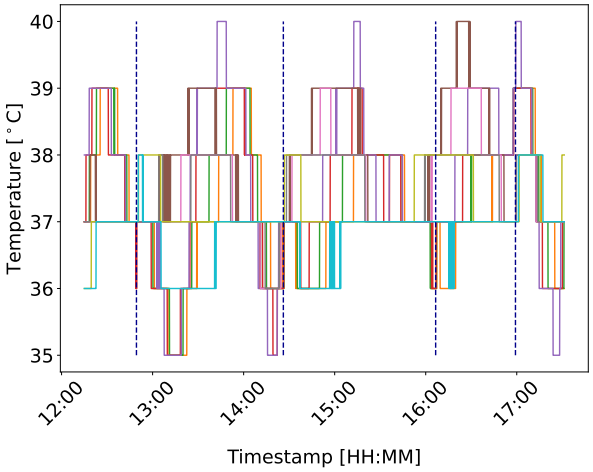
One additional experiment, not shown in Table 2, was conducted in order to collect a normal dataset for model validation purposes. We refer to this dataset as the *validation dataset*.



(a) Full dataset containing temperature values from the nine EduSat board sensors for Phase I, Phase II, and Phase III of the experiment. The blue line in the bottom of the plot presents the logged room temperature.



(b) Normal dataset of the experiment.



(c) Abnormal dataset of the experiment. The moments of the introduction of anomalies are indicated with dashed dark blue vertical lines.

Figure 9: Example of datasets generated in a single experiment. The respective colours of the lines presenting the values recorded by the nine EduSat board temperature sensors are kept the same in all plots.

Exp. date	No. anom.	Battery heating power & time	Freq.	Normal
2022/04/06	3	100%, 60s | 25% 60s | 100%, 10s	0.1 Hz	✓
2022/05/18	3	100%, 15s | 100%, 15s | 100%, 15s	0.1 Hz	✓
2022/05/20	1	100%, 15s	0.2 Hz	✓
2022/05/30	1	100%, 20s	0.2 Hz	✓
2022/06/01	4	100%, 20s | 100%, 20s | 100%, 20s | 100%, 20s	0.2 Hz	✓
2022/06/03	1	100%, 30s	0.2 Hz	✓
2022/06/08	2	100%, 30s | 100%, 30s	0.2 Hz	✓
2022/06/15	2	100%, 60s | 100%, 60s	0.2 Hz	✓
2022/06/22	2	100%, 60s | 100%, 60s	0.2 Hz	✓
2022/07/20	4	100%, 60s | 100%, 60s | 100%, 60s | 100%, 60s	0.2 Hz	✗

Table 2: Information on individual data acquisition experiments. For each anomaly, the battery heating power and heating duration time are provided in the order of appearance separated by the | delimiter. The 'Freq.' column presents the data acquisition frequency used in each experiment. The 'Normal' column indicates whether normal condition dataset was generated during the respective experiment: ✓ - yes, ✗ - no.

4 Methodology

Health monitoring and real-time detection of any symptoms of anomalous behaviour in the multivariate telemetry data is a very important task in the operation of artificial satellites [7]. The traditional and commonly used method for this purpose is the Out-Of-Limit (OOL) technique, in which the sensor measurements are checked as to whether they are within predefined ranges [8–10]. However, even the most sophisticated OOL methods fail to detect complex patterns in spacecraft flight data generated by variations in the state of components during the nominal functioning of a satellite, and are therefore bound to miss many anomalies [10]. Moreover, OOL approaches are not capable of detecting *novel behaviours*, i.e., events that are novel with respect to a set of behaviours known to be nominal, for which on-board data measurements (or their differences) are within the defined OOL thresholds [11, 12]. However, novel behaviours are often early indicators of upcoming anomalies and failures. In this sense, OOL techniques do not allow forthcoming problems to be to anticipated [12]. Furthermore, it is very costly to develop and maintain the sets of rules of OOL anomaly detection systems with the use of expert knowledge [7].

In recent years, data-driven or learning-based methods that mine relevant information from large datasets have provided breakthroughs in numerous fields. In the context of spacecraft operation, methods based on machine learning (ML) or artificial intelligence (AI) techniques offer some effective approaches in the way telemetry data is exploited in the context of anomaly root cause analysis and novelty detection [12]. In particular, AI-based approaches are effective in extracting patterns and correlations in intertwined streams of telemetry data [13]; data, which include many aspects, such as high dimensionality, multimodality, and heterogeneity [7]. In fact, some experts anticipate that spacecraft operations is an area where AI-based methods can provide the highest benefits in the space engineering domain [12].

Within AtMonSat , an AI-based solution for the problem scenario described in Section 1 has been devised. Several resource constraints are imposed on the proposed

anomaly detection algorithm since deployment to a microcontroller is targeted. In particular, it was necessary for the algorithm to fit into limited memory available on the microcontroller, to be capable of performing real-time inference with the restricted target computational resources, and to operate in the allowed energy consumption range. Memory is particularly pertinent when neural networks are deployed, due to the large number of parameters for each layer. Details on the available microcontroller's resources are provided in Section 5.5 and results of the measurements of actual usage of the resources by the microcontroller implementation of our anomaly detection solution are presented in Section 6.3.

4.1 Feature selection and engineering

Feature selection is a crucial step in the development of any ML- or AI-based computational framework. For the problem scenario of AtMonSat (see Section 1), 9 out of all 500 general telemetry attributes have been chosen. Those selected are the EPS general telemetry attributes corresponding to temperature sensors of the battery monitor, four MPPTs, and four voltage converters on the EduSat board. These telemetry attributes form the first set of features, i.e., the *raw temperature sensors* features.

The raw readings from the nine temperature sensors were used to engineer another set of nine features, one for each sensor. The engineered features are nine counters for each datapoint (i.e., readout from the nine sensors), that each record the number of datapoints since the last temperature change for the respective sensor. Due to the fact that the arrival of each datapoint initiates the next iteration of the anomaly detection algorithm in the target implementation on the microcontroller, see Section 5, datapoints are also referred to as *iterations* and the set of engineered features is named as the *Iterations Since the Last Change* (ISLC) features. Formally, we define $islc_{T_j}[0] = 0$ and for $i = 1, 2, \ldots$ which iterates over the indices of subsequent datapoints (iterations), we have

$$islc_{T_j}[i] = \begin{cases} islc_{T_j}[i-1] + 1, & \text{if } T_j[i] = T_j[i-1] \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

where $j \in [1..4]$. For example, in Table 3, the raw temperature values recorded by four imaginary sensors T_{1-4} are presented. The respective ISLC features are presented in columns $islc_{T_{1-4}}$, where a zero value entry indicates the event of temperature change in comparison to the previous temperature value for the corresponding temperature sensor.

Nevertheless, the research conducted within the AtMonSat project revealed that in order to obtain meaningful results, certain ISLC subsequences still need to be replaced with linearly interpolated values as explained next. If $islc_{T_j}[i] = 0$ for some $i > 0$, then let k be the number of iterations to previous temperature change event amongst all the sensors, i.e.,

$$k = \min(\{islc_{T_j}[i-1] + 1\} \cup \{islc_{T_l}[i] \mid l \neq j\}). \tag{3}$$

Then values $islc_{T_j}[i-k+l]$ for $1 \leq l < k$ are replaced with linearly interpolated values between $islc_{T_j}[i-k]$ and 0, i.e,

$$islc^{int}_{T_j}[i-k+l] = islc_{T_j}[i-k] - l * (islc_{T_j}[i-k]/k) \quad \text{for } 1 \leq l < k. \tag{4}$$

For an example, see columns $islc^{int}_{T_{1-4}}$ in Table 3.

The interpolation gives rise to *interpolated ISLC* features. As observed in our experiments, the interpolation is a crucial step for the considered deep-learning models to properly train and perform on the anomaly detection task.

Notice that the interpolation requirement introduces an inherent delay to the real-time anomaly detection. The classification of whether the next datapoint is anomalous or not cannot be provided immediately upon arrival of the datapoint. In order to compute the corresponding interpolated ISLC values, the new datapoint needs to be stored together with subsequent datapoints in a dedicated queue until the moment when a datapoint with a temperature change on one of the sensors is received. At that point, the interpolated ISLC values can be computed, the anomaly detection algorithm can be run for all the datapoints in the buffer, and finally the buffer can be freed.

	T_1	T_2	T_3	T_4	$islc_{T_1}$	$islc_{T_2}$	$islc_{T_3}$	$islc_{T_4}$	$islc^{int}_{T_1}$	$islc^{int}_{T_2}$	$islc^{int}_{T_3}$	$islc^{int}_{T_4}$
0	37	38	38	39	0	0	0	0	0.00	0	0	0
1	37	38	38	39	1	1	1	1	0.00	1	1	1
2	38	38	38	39	0	2	2	2	0.00	2	2	2
3	38	38	38	39	1	3	3	3	1.00	1	1	3
4	38	39	39	39	2	0	0	4	2.00	0	0	4
5	38	39	39	39	3	1	1	5	1.50	1	1	5
6	38	39	39	39	4	2	2	6	1.00	2	2	6
7	38	39	39	39	5	3	3	7	0.50	3	3	7
8	39	39	39	39	0	4	4	8	0.00	4	4	8
9	39	39	39	39	1	5	5	9	1.00	5	5	4
10	39	39	39	38	2	6	6	0	2.00	6	6	0
11	39	39	39	38	3	7	7	1	1.33	7	7	1
12	39	39	39	38	4	8	8	2	0.67	8	8	2
13	40	39	39	38	0	9	9	3	0.00	9	9	3
14	40	39	39	38	1	10	10	4	1.00	10	10	4

Table 3: Example of ISLC features engineering. Columns T_{1-4} contain raw values as recorded by four temperature sensors; columns $islc_{T_{1-4}}$ contain the corresponding ISLC values before applying interpolation; columns $islc^{int}_{T_{1-4}}$ contain the interpolated ISLC features, which are used for model training and inference. Red temperature values in columns T_{1-4} indicate the events of temperature change, while interpolated values in columns $islc^{int}_{T_{1-4}}$ are highlighted with the cyan colour.

For example, let us consider the entries in columns T_{1-4} in Table 3 with index 5 as the next datapoint that is available to the anomaly detection algorithm. No temperature change takes place with respect to the datapoint indexed 4. Therefore, the execution of the anomaly detection algorithm for this datapoint is postponed. Upon arrival of the datapoint with index 8, the values in columns $islc^{int}_{T_{1-4}}$ can be computed and the anomaly detection algorithm run for points with indices 5–8. Nevertheless, our experiments show that usually the numbers of subsequent datapoints buffered is not large. A histogram

showing the distribution of these numbers obtained when computing the interpolated ISLC features for all normal datasets of the experiments in Table 2 is shown in Figure 10. The maximum value is 133.

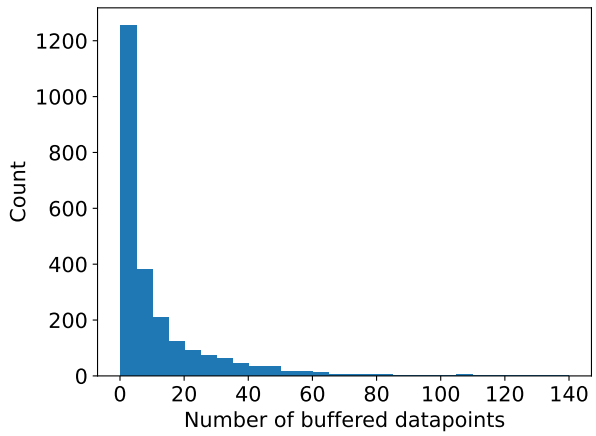
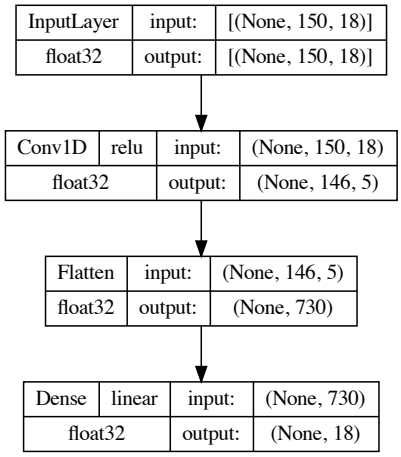


Figure 10: Histogram of the number of datapoints queued before the anomaly detection algorithm is run. The histogram is generated from buffering data collected while computing interpolated ISLC features for all normal datasets of the experiments in Table 2.

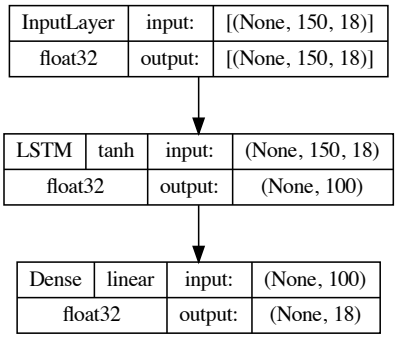
At first sight, one could argue that the delay in anomaly detection caused by the need to buffer 100 datapoints is too long: with data acquisition frequency of 0.2 Hz this would result in a delay of 500 s, i.e., 8 min. 20 s. However, for our particular scenario, the anomalies are assumed to manifest themselves via temperature changes. Therefore, if an anomaly fitting the scenario occurs, it will cause a temperature change, which in turn will cause the interpolated ISLC values to be computed and the anomaly detection algorithm to be triggered to detect the abnormal behaviour of the system. Therefore, we consider this inherent delay to be a minor disadvantage of the proposed solution. Details on how the ISLC interpolation is implemented in the real-time analysis scenario on the microcontroller are discussed in Section 5.

We also experimented with two other sets of features. In particular, we considered the two highest-order components obtained with the Principal Components Analysis (PCA) applied either 1) to the raw temperature data from the nine sensors, or 2) to the nine interpolated ISLC values. For anomaly detection, PCA is known to be effective due to its ability to identify points that violate inter-attribute dependencies [14] and the violations are most pronounced in the highest-order components, i.e., ones with the lowest variance. However, our experiments both with synthetic datasets obtained from the simplified thermal simulator and the real lab datasets found that that taking the feature set to be the two highest-order PCA components of the nine raw temperature sensor values or of the nine interpolated ISLC values was not sufficient.

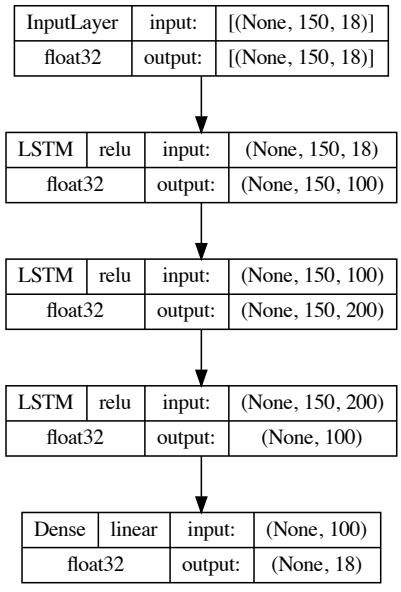
To summarise, since the synthetic datasets are exceptionally clean – they do not contain any noise, and are regular – they are perfectly periodic, only the nine interpolated ISLC features were enough to consider for the anomaly detection task. For the noisy and irregular real lab datasets, both the nine raw temperature sensors features and the nine interpolated ISLC features were used.



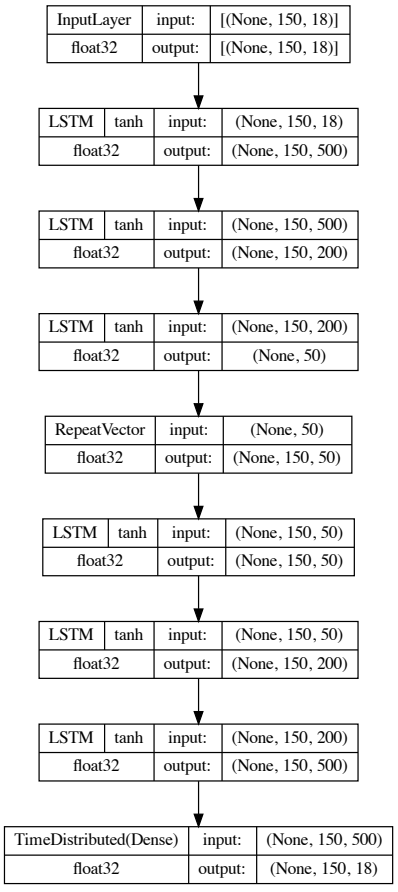
(a) CNN-based model (13,613)



(b) LSTM-based model ver. 1 (49,418)



(c) LSTM-based model ver. 2 (410,618)



(d) AutoEncoder (3,281,018)

Figure 11: Main types of deep-learning model architectures considered in the AtMonSat project with example configurations. The total numbers of parameters are provided in parentheses in the captions of the individual architectures. The graphs were produced with the tf.keras.utils.plot_model function of TensorFlow.

4.2 Deep-learning model architectures

We considered different deep-learning model architectures: CNN-based, LSTM-based, and an AutoEncoder with different configurations of activation functions and individual layers. The architectures with example configurations are presented in Figure 11.

We selected the approach of training deep-learning models exclusively with the normal datasets (i.e., regression training), as an alternative version of training the models using both normal and abnormal datasets (i.e., classification type of training). All models were trained with the "Mean Squared Error" (MSE) loss function and the Adam optimiser with TensorFlow default settings. Early stopping callback was used with the following configuration: monitor='loss', min_delta=$1e^{-2}$, patience=10, and mode ='auto'. The number of epochs was set to 500. The input to the models consisted of batches of input tensors of shape $(window\_length, number\_of\_features)$, where the $window\_length$ defines the size of the number of consecutive (historical) datapoints used to make the inference. For the target model which is presented in the continuation, we set $window\_size = 150$ and $number\_of\_features = 18$. The output of the models are tensors of shape $(1, number\_of\_features)$, i.e., the models infer a single next datapoint in terms of interpolated ISLC values and raw temperature values. During training, the inferred outputs where compared to the actual datapoints using the MSE loss function, which was optimised with the backpropagation algorithm.

4.3 Anomaly detection algorithm

We explain now the anomaly detection algorithm we developed. The algorithm makes use of a metric for calculating errors, which turns out to be an important design decision. The two metrics we considered were: 1) the Euclidean distance and 2) the Mahalanobis distance between the two vectors of length $number\_of\_features$. In the case of the Mahalanobis distance, the trained model is run on the training data to estimate the mean and the inverse standard deviation matrix based on the normal condition error vectors obtained by subtracting the prediction from the reference. The estimated mean and inverse of the covariance matrix are used to define the Mahalanobis distance function for the analysis of the abnormal datasets.

The anomaly detection algorithm developed within AtMonSat is devised as follows.

Step 1: A deep-learning model is trained on data from all the normal datasets generated with the experiments presented in Table 2 with the settings presented in Section 4.2.

Step 2: The trained model is run on an anomalous dataset. For each inferred point an error with respect to the actual datapoint is computed with respect to the chosen metric (the Euclidean or Mahalanobis distance). The errors, computed based on the model outputs and the ground-truth feature vectors, are referred to as the *raw anomaly detection signal*.

Step 3: The raw anomaly detection signal is post-processed by considering a threshold value and a so-called *hold-off window*. The threshold value is determined with

a separate procedure described in Section 4.4, while the *hold-off window* is given by a duration specifying the number of timepoints. The raw anomaly detection signal is scanned in the chronological order of the corresponding datapoints one point after the other. If an error value greater or equal to the threshold value is observed, the corresponding datapoint is classified as anomalous and all consecutive datapoints within the hold-off window starting at the anomalous datapoint are classified as normal. Scanning then continues from the first subsequent error outside the hold-off window.

The post-processing is introduced due to the fact that the information on abnormal behaviour is carried in the change of the temperature pattern. However, temperature changes are inherently slow and continuous. Furthermore, there is a delay between the occurrence of a board element malfunction and the time the disturbance in the temperature pattern is registered by the sensors scattered across the board. The delay is determined by the thermal conduction characteristics of the board. Given this, the rationale behind introducing the hold-off window for the post-processing of the raw anomaly signal is twofold. First, due to continuity of thermal changes, an anomaly is usually indicated by consecutive raw signal values exceeding the threshold. Second, an anomaly can be demonstrated by the occurrence of multiple anomaly signal peaks one shortly after the other as observed in some of the generated datasets. The introduction of the hold-off window post-processing allows all datapoints exceeding the threshold in a window to be considered as a single anomaly. This cleans the raw anomaly signal for subsequent quantitative evaluation of the performance of the models with respect to appropriate metrics described in Section 6. In our case, we set the hold-off window to 60 timepoints.

4.4 Determination of the anomaly detection threshold

Both the Euclidean and the Mahalanobis error thresholds are determined with the same approach which mimics the statistical leave-one-out cross-validation (a.k.a. out-of-sample testing) procedure. One of the normal datasets of the experiments presented in Table 2 is kept aside and all the others are used to train a model with the settings provided in Section 4.2. Then, the fitted model is run on the kept-aside normal dataset and the errors are computed and stored. For the Mahalanobis distance, the mean and the inverse covariance matrix are estimated from the errors obtained by running inference on all the normal datasets except the one kept aside. The procedure is repeated for all normal datasets kept aside. Finally, all the errors are used to generate a histogram, i.e., an empirical distribution, and the significance threshold value corresponding to a chosen statistical significance level (e.g., 0.05 or 0.01%) is used as the respective Euclidean or Mahalanobis error threshold, see Figure 12.

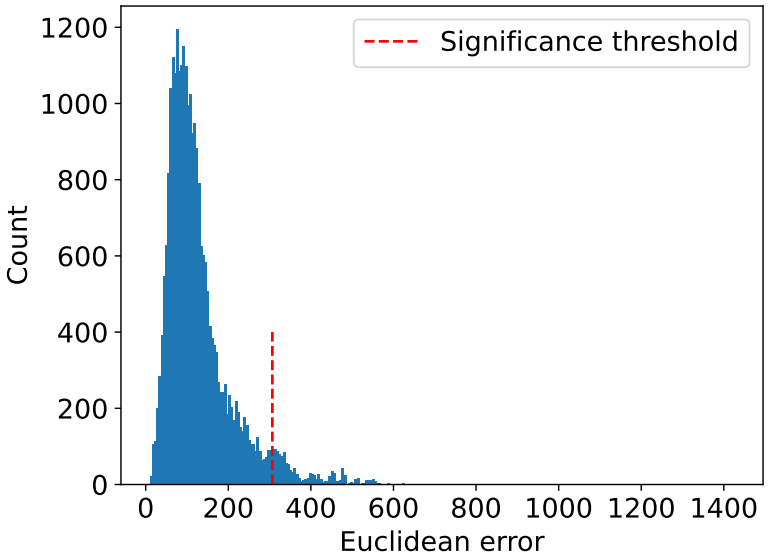


Figure 12: An empirical distribution of the Euclidean error of values inferred by the CNN-based model exploited by the anomaly detection algorithm. A significance threshold value is indicated by the vertical red dashed line corresponding to a statistical significance level of 0.05.

5 Implementation

The anomaly-detection algorithm devised within the AtMonSat project is implemented in C++ in a generic way that facilitates the compilation for a PC and targeting to a microcontroller. The algorithm is embedded in a testbed to allow verification of the results and performance. Only a small part (e.g., the data link layer and the time metric) of this testbed is hardware dependent and must be exchanged for the PC and microcontroller versions. Figure 13 shows a general overview and the differences of both implementations. All source codes are made available via the AtMonSat GitHub repository [15].

5.1 Testbed

The testbed consists of a frontend and a backend. The frontend is running on the host, e.g., a PC, and the backend is part of the target.

Figure 14 presents a block diagram of the testbed, shown on the left-hand side, which is calling the anomaly detection algorithm, shown on the right-hand side, upon reception of a datapoint. The anomaly detection algorithm itself is discussed in Section 5.3.

5.1.1 Frontend

The testbed frontend consists of a Jupyter notebook (*atmonsat_testbench.ipynb*), which is running on top of Jupyter and iPython, to control the experiment and to record and to visualize the results. This notebook allows:

- selecting the connection target, i.e., local or remote experiment;
- selecting the dataset and dataset range ('experiment', 'normal', 'anomaly') to be send to the target; and
- changing the algorithm settings.

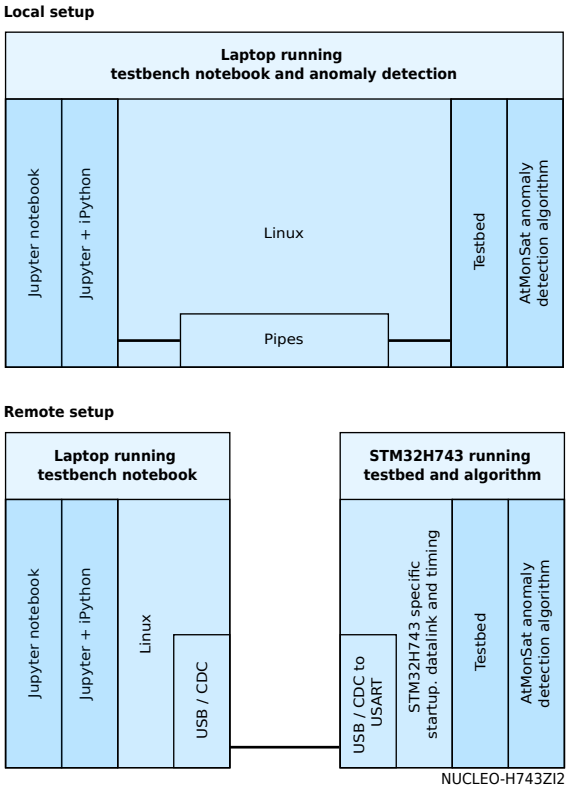


Figure 13: Overview of the testbed with a local (top) and a remote connection (bottom). Both versions only differ in the data link layer and the timing functions.

All results and associated settings are stored in a unique (timestamped) directory in the results subdirectory.

The testbench relies on the dataset notebook (*dataset.ipynb*) to provide the data and meta-data for each experiment, and the connection notebook (*connection.ipynb*) to provide a reliable connection towards the target.

All experimental data are stored in the *datasets* directory with a dedicated subdirectory for each experiment. Each subdirectory contains subfolders with raw experimental data and a meta file (*meta.json*) to explain and correct the data. The individual datasets of the experiments can be accessed using the Dataset class of the dataset notebook (*dataset.ipynb*).

5.1.2 Backend

The backend consists of a hardware specific data link layer to accept data from the communication byte by byte and to forward it to the hardware independent communication protocol.

The communication protocol reassembles the transmitted data into objects (e.g., datapoint, inverse covariance matrix for Mahalanobis error) and calls a proper callback han-

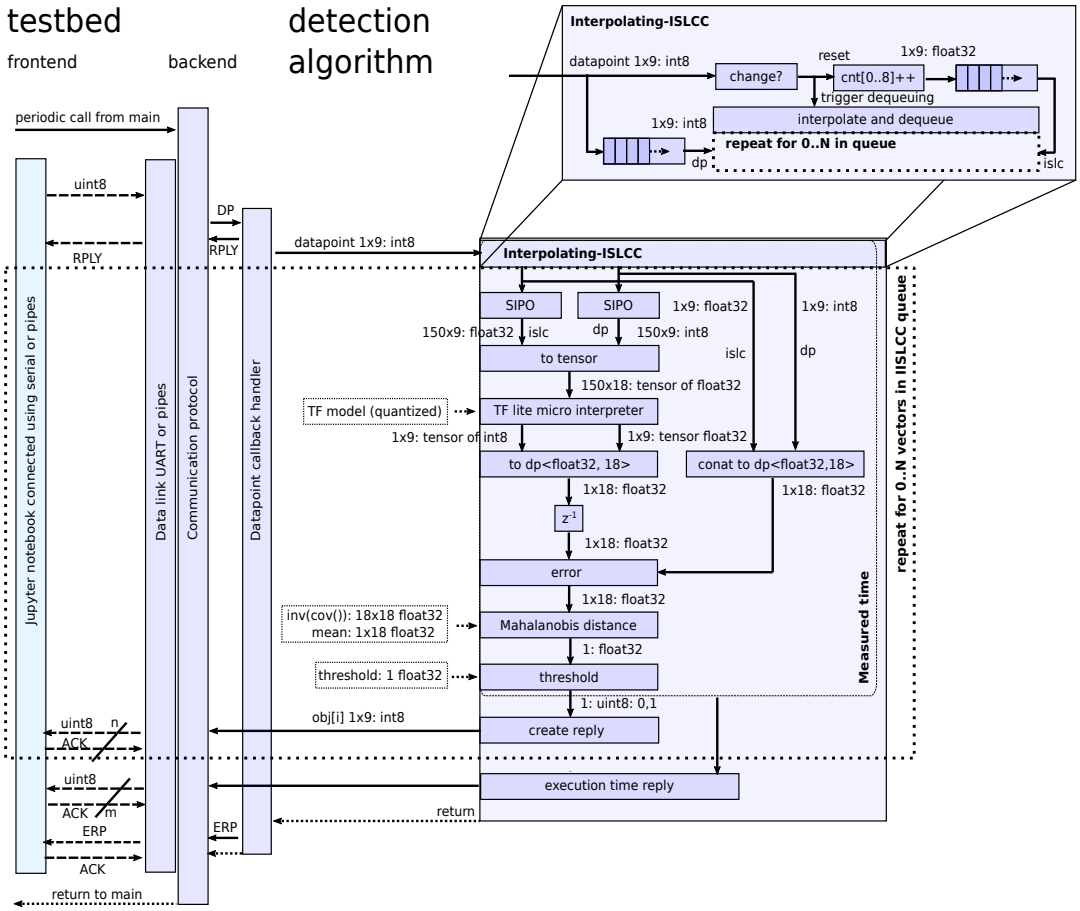


Figure 14: Flow diagram of the STM32H743 microcontroller implementation.

dler, such as the datapoint callback handler depicted on the left-hand side of the diagram in Figure 14, once the object is updated. The diagram only shows the datapoint callback handler. Other, simpler callback handlers (e.g., initialization, inverse covariance matrix update) are omitted for clarity, since they are handled the same way as the datapoint callback handler in the diagram.

The datapoint callback handler's main purpose is to call one iteration of the the anomaly detection algorithm upon reception of a datapoint. Moreover, the handler starts the timer for the execution time measurement prior to the call of the anomaly detection algorithm iteration and stops the timer after the completion of the iteration. The datapoint callback handler then replies the execution time to the host.

The datapoint callback handler also supplies the anomaly detection algorithm with the necessary callbacks to reply, e.g., anomaly detection outcome, Mahalanobis distance, error messages, to the host. These callbacks are exempt from the time measurement as they are part of the protocol and not of the anomaly detection algorithm.

5.1.3 Communication

Communication direction is by default from host to target. It is a half duplex communication where every received byte is acknowledged by the peer. The design considerations here are that one USART (Rx+Tx) is sufficient and no multi-threading is required.

The communication transports a stream of data containers of different types. This allows different content, which is identified by a *cmd id*, to be transmitted without modifying the protocol. A dedicated callback handler is called after the reception of each container type.

Every received byte is acknowledged to confirm the reception and to delay the communication until processing on the target, which is slower, is completed. This allows working without buffers and interrupts which otherwise would disturb timing measurements.

Sending an RPLY (reply) instead of ACK (acknowledge) upon reception of a byte allows the target to change the communication direction from target to host until an ERP (End of RePly) command is send. Reversing the communication direction allows the target to send one or multiple reply objects, e.g., detection results and timing information, to the host.

5.2 Conversion of a TensorFlow pre-trained model to a TensorFlow Lite Micro version

A deep-learning model was implemented and pre-trained using the TensorFlow framework. The model was then converted to its TensorFlow Lite Micro using the provided converter by TensorFlow Lite, i.e., tf.lite.TFLiteConverter. In the TensorFlow model we restricted ourselves to just use operators that are implemented in the TensorFlow Lite Micro and listed in the *builtin_ops.h* file. This allowed straightforward model conversion. The resulting TensorFlow Lite Micro model was then saved as an array in C++ source file. The details on the conversion can be found in the *firmwares/atmonsat_common_sources/atmonsat_model_quantize.ipynb* Jupyter notebook.

Attempts to further quantize the TensorFlow Lite Micro model resulted in significant performance drop. Therefore, we decided not to quantize the target model. Due to the time limitations of the AtMonSat project, we decided to leave quantization for future research as explained in Section 7.

5.3 Anomaly Detection Algorithm

The C++ implementation of the anomaly detection algorithm is build around a TensorFlow Lite Micro model which is obtained by converting a pre-trained deep-learning TensorFlow model as described in Section 5.2. The algorithm is evaluated on the arrival of each datapoint. For the computation of the interpolated ISLC features, datapoints are entered into the *Interpolating Iterations Since Last Change Counter* module from where they are forwarded into two Serial-In-Parallel-Out (SIPO) blocks. The outputs of the

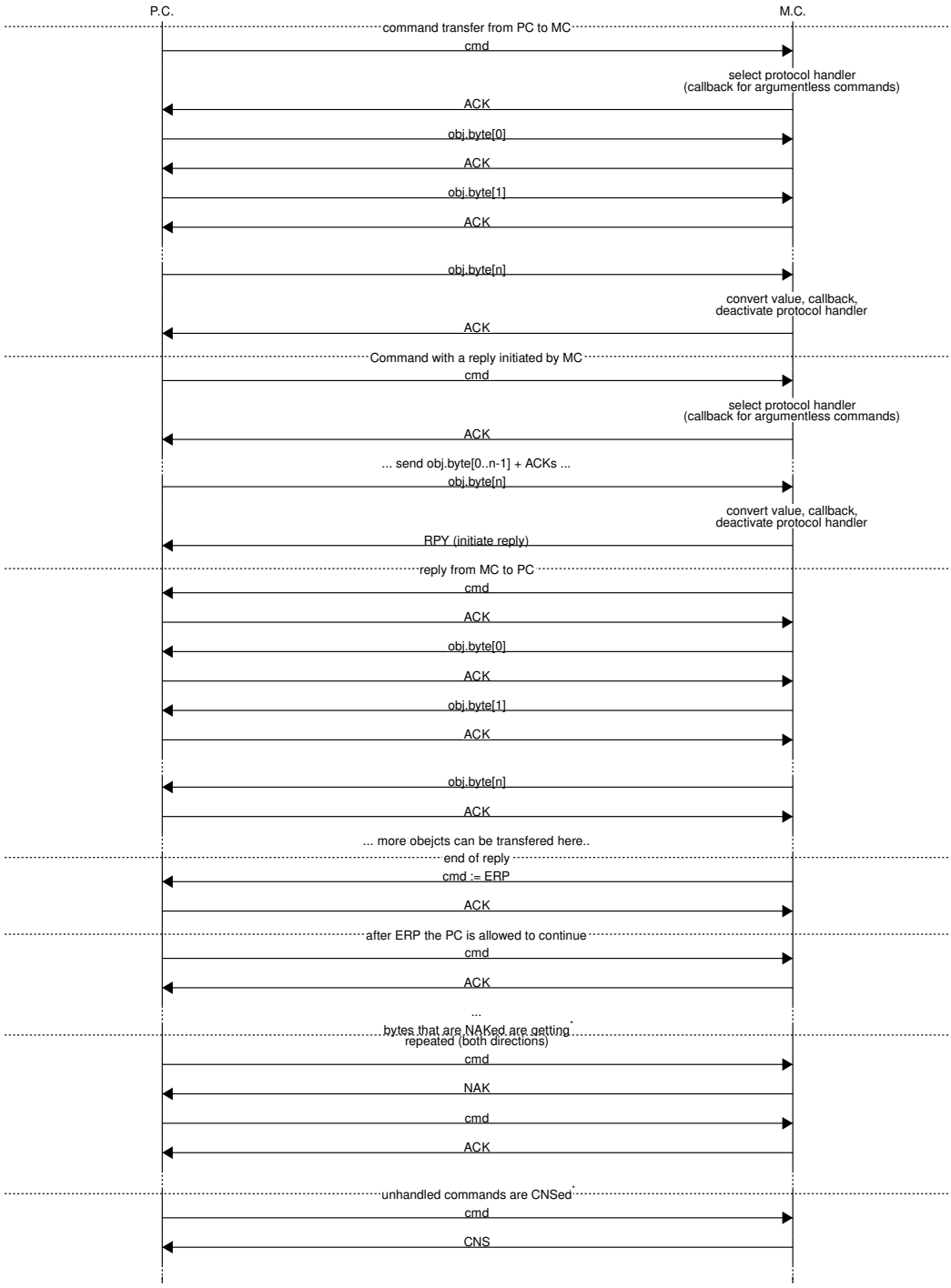


Figure 15: A diagram providing an overview of the communication protocol.

two SIPOs are two windows (e.g., of width 150) of datapoints that are given as input to the TensorFlow Lite Micro model after conversion to a float32 tensor, see Figure 14.

The output of the deep-learning model is delayed by one iteration before getting compared against the new window values of the next iteration. The result of the comparison is the error that is measured with a Mahalanobis distance against a predefined mean. The distance is reported to the host. Next, the Mahalanobis distance is compared against a pre-determined threshold value and processed with a hold-off window (e.g., of the length of 60 samples) as described in Section 4.3. This results in the anomaly detection decision, which is next reported back to the host.

5.3.1 Interpolating Iterations Since Last Change Counter module

The Interpolating Iterations Since Last Change Counter (IISLCC) module implements the computation of the interpolated ISLC features described in Section 4.1. Upon reception of a new datapoint, i.e., a vector with the raw temperature sensors features of length $number\_of\_features$, the IISLCC increases by 1 its internal counters – one for each of the $number\_of\_features$ raw temperature sensors features. Then, the new datapoint is compared against the last datapoint for changes detection.

If no change took place, then the new datapoint and a vector of the IISLCC internal counters' current values are enqueued in their corresponding queues of fixed size, which was determined based on the results of the study of datapoints buffering data presented in Section 4.1. If the queues are full, one anomaly detection callback is forced to free storage space for the new datapoint.

If a change is detected, the algorithm finds the index (or indices in the case of multiple simultaneous changes) of the raw temperature sensors feature(s) for which the change(s) occurred and resets the corresponding counter(s) to 0. Next, for each of the counters, all queued values are re-visited and replaced with linearly interpolated values between the value of the first enqueued value of the particular counter and the just reset zero value, see Section 4.1 for the details on the interpolation. Then, the inference callback handler is called for each enqueued datapoint and its corresponding vector of all IISLCC internal counters' values, emptying both queues. Finally, the current datapoint and its corresponding vector of all IISLCC internal counters' current values are enqueued for the next iteration.

Further details on the IISLCC Implementation can be found in *islcc_interpolator.h*.

5.4 PC implementation (native)

The PC implementation is targeted for a Linux (e.g., Ubuntu) platform. It consists of the testbed and the anomaly detection algorithm, see the top part of Figure 13. The data link layer of the testbed uses pipes (stdin, stdout, stderr) for communication with the controlling Jupyter notebook, where stderr can be used for debugging purposes but is

not used by the testbench itself. The time metric is able to measure nanoseconds based on a Linux Kernel API.

Building the PC implementation can be done by calling the Makefile *make*. The Placebo version (version without algorithm) can be build with *make PLACEBO=1*.

The path to this binary can be setup as one of the settings of the testbench Jupyter notebook.

5.5 Microcontroller implementation

The microcontroller implementation is targeted towards an STM32H743 32bit ARM® Cortex®-M7 microcontroller. This is the same microcontroller as the one used in the EduSat. The STM32H743 microcontroller is a 32-bit Arm® Cortex®-M7 core with double-precision FPU and L1 cache (16 kB of data and 16 kB of instruction cache). The complete datasheet can be found at [16].

The implementation was tested on a NUCLEO-H743ZI board, see Figure 16. The code can be built and uploaded using the STM32CubeIDE from STMicroelectronics.

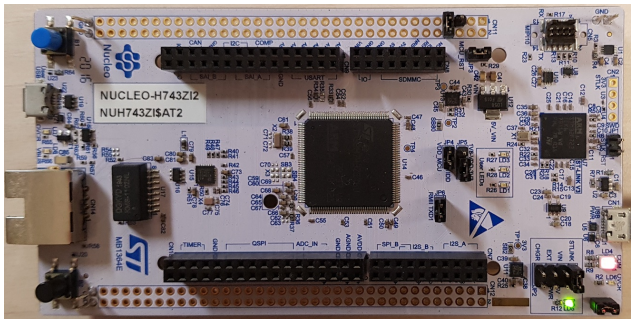


Figure 16: NUCLEO-H743ZI development board with the STM32H743ZI microcontroller unit (MCU).

Clock configuration functions are provided for four frequencies, i.e., 39 MHz, 78 MHz, 146 MHz, and 298 MHz. These values are selected to get a millisecond timer for timing measurements while fulfilling all core internal timing constraints. The active clock configuration can be selected in the *main_stm32.h* file. This also adapts the communication baudrate. The clock frequency is one of the settings that must be changed in the testbench Jupyter notebook before the communication. This automatically adapts the communication baudrate for that clock frequency on the host side.

The POWER_MEASUREMENT definition must be declared while making power measurements. This disables all unnecessary consumers (e.g., LEDs).

5.5.1 Baremetal implementation

The baremetal implementation is targeted for an STM32H743 microcontroller. It consists of the testbed and the anomaly detection algorithm, see the top part of Figure 13. The data link layer of the testbed uses USB/CDC protocol for communication with the controlling Jupyter notebook. The time metric is able to measure milliseconds based on a 32-bit timer. The main entry of the baremetal implementation can be found in

main_stm32.cc. The data link layer is defined in *protocol_datalink_stm32_usart.∗* files and the time metric is declared in *time_metric_stm32.∗* files.

5.5.2 FreeRTOS implementation

The AtMonSat anomaly detection code was also tested in conjunction with a preemptive FreeRTOS kernel. The anomaly detection algorithm controlled by the testbed is implemented as one task. The code of this task is close to the baremetal implementation, see Section 5.5.1. Another dummy task is created to demonstrate multitasking. All memory used by the kernel and tasks where allocated statically. The main entry of the FreeRTOS implementation can be found in *main_stm32_freertos.cc*. The data link layer is defined in *protocol_datalink_stm32_usart.∗* files and the time metric is declared in *time_metric_stm32_freertos.∗* files.

5.6 Power measurement setup

Increase in power consumption of the microcontroller due to the execution of the anomaly detection algorithm is estimated as the difference in power consumption of the placebo version of the firmware and the firmware including the anomaly detection algorithm. Measuring the difference cancels the power consumption due to the testbed and other components on the development board.

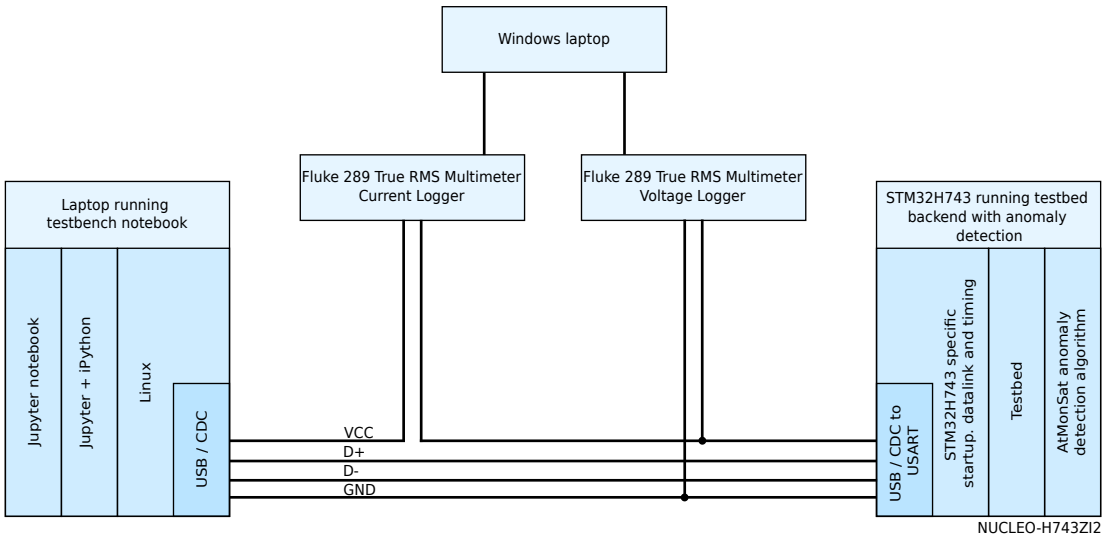


Figure 17: Overview of the setup for measuring current and voltage during execution of the testbed with the anomaly detection algorithm on the STM32H743 microcontroller.

An overview of the setup for measuring power consumption is presented in Figure 17, while a photo of the actual lab setup is shown in Figure 18 shows a photo of the setup in the lab. Power is measured by recording the voltage and current with two Fluke 289

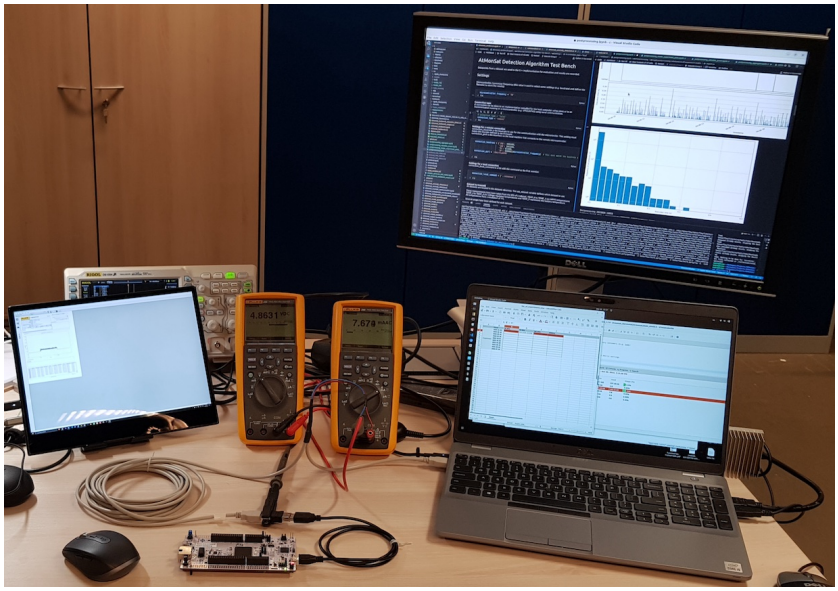


Figure 18: Setup for measuring the current and the voltage during execution of the testbed with the anomaly detection algorithm on the STM32H743 microcontroller.

True RMS multimeters with logging capability. Samples are recorded every second and they consist of the RMS average value of the voltage or current during that second.

During post processing, the voltage and current logs are realigned. Next, the voltage and the current values are multiplied to give a power trace of the experiment. Mean and standard deviation of the power trace are calculated and reported in the *power_statistics.json* file. The raw data is saved in CSV files.

5.7 Post-processing

All the data collected during the execution time and power consumption measurements is stored in raw format. To visualize and aggregate the information, a number of post-processing Jupyter notebooks is provided.

It is sufficient to run the main post-processing notebook (*postprocessing.ipynb*) which applies all other post-processing notebooks to the data.

Notice that the post-processing notebook works on a directory named *measurements*, whereas testbench stores the data in the *results* directory. The data has to be copied by hand or the measurement_root variable in the post-processing notebook needs to be set with the path to the directory. This is done on purpose to prevent automatically adding results from a test run into the measurements directory.

Aggregated data is stored inside the *measurements* directory. Generated plots and data for each measurement can be found in its subdirectory of the *measurements* directory.

6 Results

6.1 Anomaly detection on synthetic datasets

We run the raw anomaly detection algorithm, i.e., without post-processing of the raw anomaly detection signal (Step 3), on the synthetic datasets. We considered three different deep-learning models with architectures presented in Figures 11a, 11c, and 11d, i.e., a CNN-based model, an LSTM-based model, and an AutoEncoder. The models were trained with the discretised simulated temperature sensor data in normal conditions presented in Figure 5. However, in the case of the synthetic data analysis, we introduced the following modifications:

- the window size parameter was set to 20;
- as input we only used the nine interpolated ISLC features without the nine corresponding raw temperature features;
- the number of filters in the Conv1D layer of the CNN-based model in Figure 11a was changed from 5 to 64.

Next, the trained models were applied to the discretised simulated temperature sensor data in abnormal conditions presented in Figure 6. Our experiments revealed that only the use of the Mahalanobis distance metric was producing meaningful results. The raw anomaly detection signals, i.e., the Mahalanobis errors, for all the three deep-learning models are presented in Figures 19, 20, and 21. Clearly all models are capable of correctly identifying the anomalies.

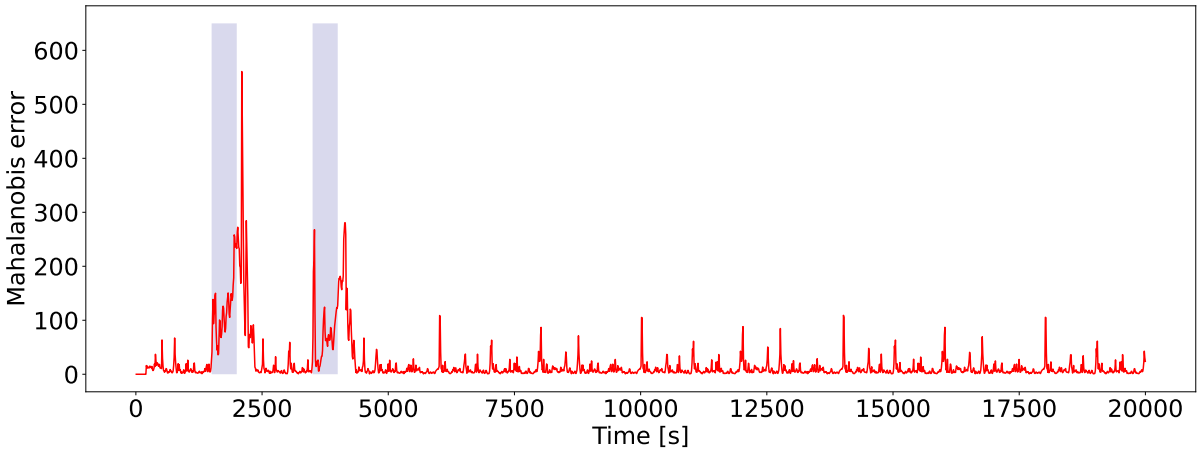


Figure 19: Raw anomaly detection signal output by the anomaly detection algorithm with the CNN-based model in Figure 11a for the discretised simulated temperature sensor data in abnormal conditions presented in Figure 6. With respect to the original architecture in Figure 11a, the number of filters of the Conv1D layer was increased to 64. The window size parameter was set to 20. The times when the anomalous battery heating is on are indicated with the light-violet bars.

Nevertheless, since the synthetic datasets are ideally regular, i.e., perfectly periodic and without any noise, deep-learning models with a relatively small number of parameters

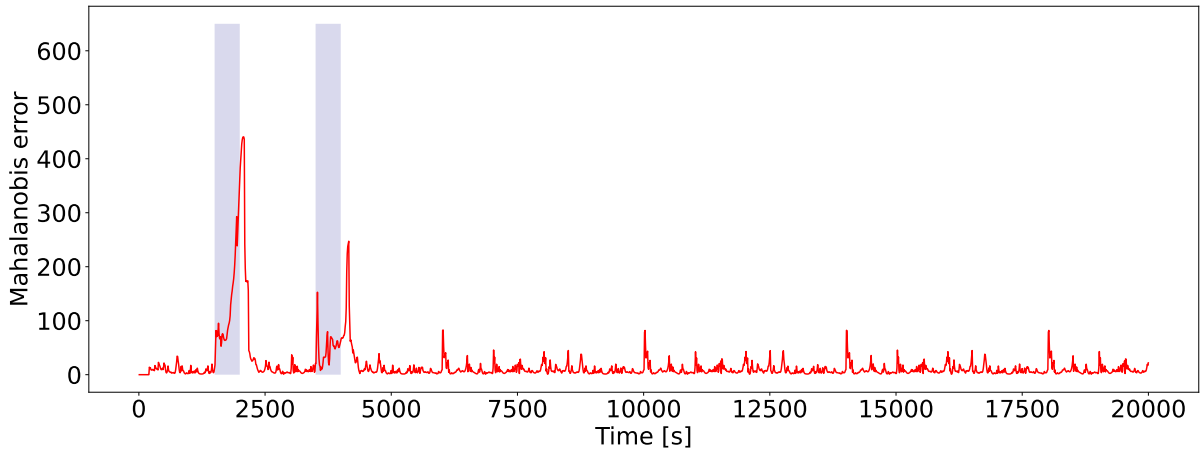


Figure 20: Raw anomaly detection signal output by the anomaly detection algorithm with the LSTM-based model in Figure 11c for the discretised simulated temperature sensor data in abnormal conditions presented in Figure 6. The window size parameter was set to 20. The times when the anomalous battery heating is on are indicated with the light-violet bars.

were capable of capturing the pattern of inter-attribute dependencies during training, and also capable of identifying anomalies during inference. Thus, we did not perform any fine-tuning or optimisation of the deep-learning models, but rather decided to focus on the much more challenging real lab data.

We just mention here that an initial quantization of the models run on a PC did not result in a significant drop in anomaly detection performance, contrary to what we observed in the case of real lab data.

6.2 Anomaly detection on lab datasets

The anomaly detection algorithm employing trained models was applied to the anomalous datasets of the experiments presented in Table 2. As in the case of the synthetic datasets, our experiments revealed that only the use of the Mahalanobis distance metric was generating meaningful results. Furthermore, only the CNN-based model outlined in Figure 11a was capable of providing meaningful results for the noisy real-life data in contrast to synthetic data in which case all types of architectures in Figure 11 are valid. For example, the raw anomaly detection signal output by the anomaly detection algorithm with the LSTM-based model in Figure 11b for the 2022.06.03 abnormal dataset is shown in Figure 22. The LSTM-based model failed to identify the anomaly due to the too high Mahalanobis error threshold indicated with the dashed green line. The high value of the threshold originated from poor performance of the model on the out-of-sample test with the 2022.05.30 normal dataset kept aside (see Section 4.4 for details on the threshold determination). Given this threshold, the anomaly detection algorithm with the LSTM-based model performed poorly also on other abnormal datasets.

Nevertheless, one could argue that lowering the threshold in some way could lead to better performance. To verify this, we computed the areas under the precision-recall

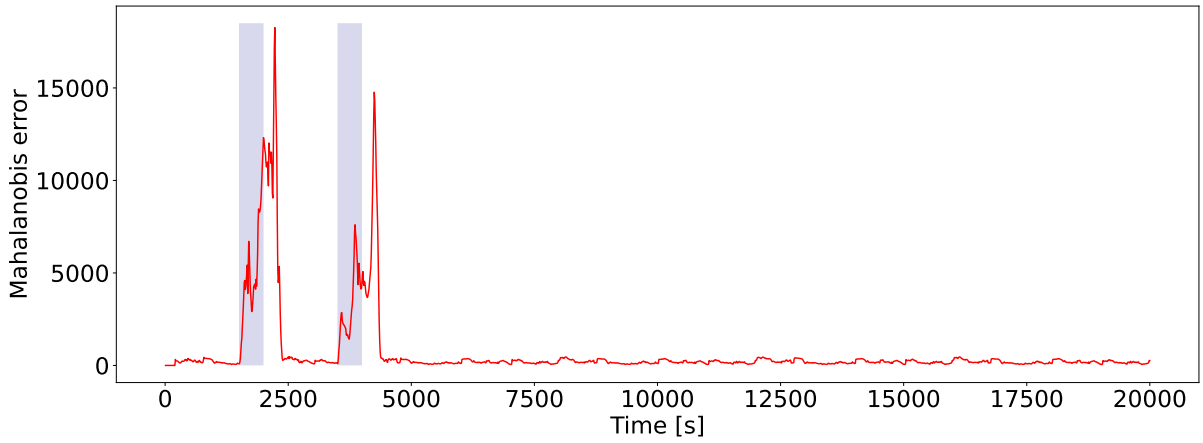


Figure 21: Raw anomaly detection signal output by the anomaly detection algorithm with the AutoEncoder model in Figure 11d for the discretised simulated temperature sensor data in abnormal conditions presented in Figure 6. The window size parameter was set to 20. The times when the anomalous battery heating is on are indicated with the light-violet bars.

curves generated with the anomaly detection algorithm with the LSTM-based model for individual abnormal datasets. The areas are presented in Table 4.

Exp. date	04/06	05/18	05/20	05/30	06/01	06/03	06/08	06/15	06/22	07/20
PRC area	0.90	0.57	1.00	1.00	0.04	0.25	0.16	1.00	1.00	0.30

Table 4: Areas under the precision-recall curves (PRC) of the anomaly detection algorithm with the LSTM-based model in Figure 11b run on individual abnormal datasets. A shorter notation is used to denote the datasets, e.g., 04/06 stands for 2022/04/06.

By comparing the areas of the LSTM-based model with the respective areas obtained with the CNN-based model, which are going to be discussed in Section 6.2.1 and shown in Figure 26, it was concluded that the CNN-based model is preferable. The conclusion remained valid for the LSTM-based model in Figure 11c and for other considered variants with different numbers, shapes, and activation functions of LSTM layers, which are not shown in this report.

Given the above observations, we henceforth focus on the CNN-based model architecture. Our experiments with different sizes of the window with past datapoints for inference, i.e., 100, 150, 200, revealed that the best performance was achieved with the window size of 150. The CNN-based model is trained as described in Section 4. Additionally, the trained model is validated with the validation dataset. The results of the anomaly detection algorithm run on the validation dataset, which is a normal condition dataset (see Section 3.2), are presented in Figure 23.

Then, the anomaly detection algorithm is run on the anomalous datasets of the experiments in Table 2. The obtained results in terms of the raw anomaly detection signal and the post-processed anomaly detection signal, which is the final output of the the proposed anomaly detection algorithm, are presented in Figure 25.

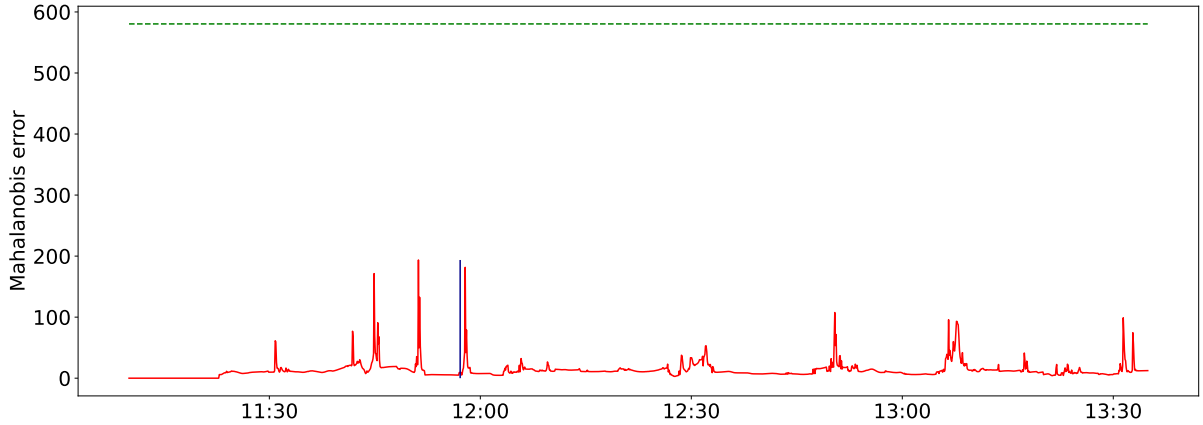


Figure 22: Raw anomaly detection signal output by the anomaly detection algorithm with the LSTM-based model in Figure 11b for the 2022/06/03 abnormal dataset. The Mahalanobis threshold value, determined as described in Section 4.4, is indicated with dashed green line. The blue vertical line shows the anomaly of the dataset.

6.2.1 Quantitative evaluation of the algorithm performance

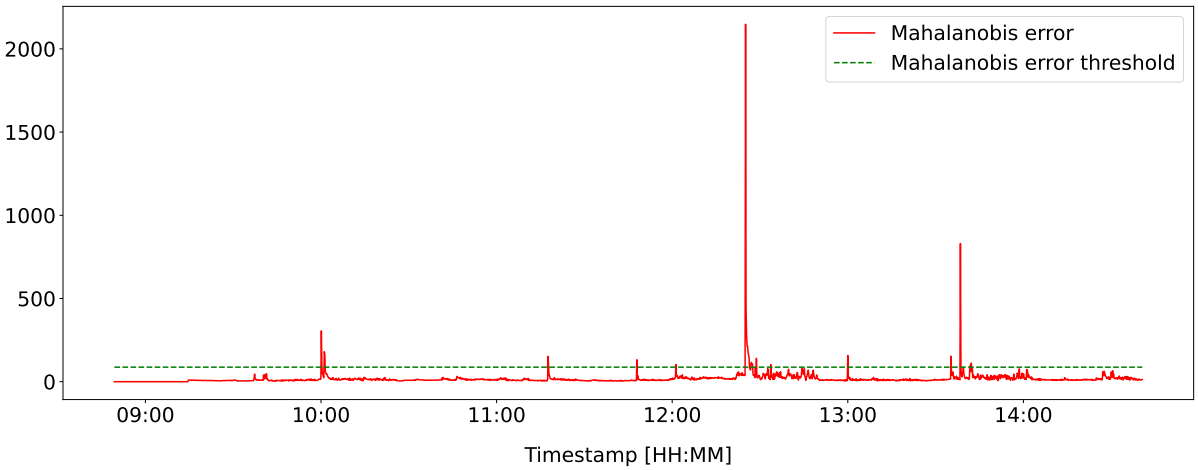
The confusion matrix values, i.e., the true positives (TPs), false positives (FPs), and false negatives (FNs), and the values of three standard metrics for classification evaluation, i.e., precision, recall, and the F1-score, where

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}},$$

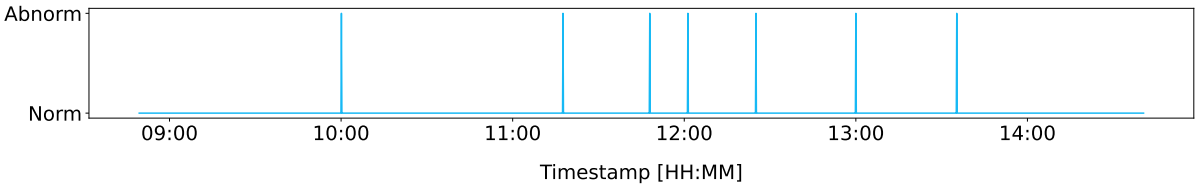
$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \text{ and} \tag{5}$$

$$\text{F1-score} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}},$$

are calculated as explained next. In order to count the TPs, we introduce the notion of a TP-interval. Let a_i be the index of the first datapoint recorded by the temperature sensors after (or at) the anomaly start time, i.e., the moment of battery heating being turned on. Since heating anomaly requires time for the temperature change to reach the sensors, there is an inherent delay, or inertia, before the anomaly is detectable. We therefore allow the anomaly to be detected within a window of datapoints, referred to as the *anomaly inertia window*. In our experiments, we set the length of this window, denoted l_{aiw}, to 60. If an anomaly signal is output by the anomaly detection algorithm for any of the datapoints with indices in the range $[a_i, a_i + l_{aiw}]$, the anomaly signal is considered as a TP. However, due to the interpolation of the ISLCs, there is a possibility for an anomaly to be detected and identified before the datapoint with index a_i as the information about the near future is conveyed in the interpolated subsequence of ISLC(s). For example, the subsequence $islc_{T_1}^{int}[4-6]$ in Table 3 contains interpolated values. By observing it, one can deduce that a temperature change will be captured by the temperature sensor T_1 in the 8th datapoint. To take this possibility into account, the



(a) The Mahalanobis error, i.e., the inference error measured with the Mahalanobis distance, on the validation dataset.



(b) The output of the anomaly detection algorithm on the validation dataset, i.e., the post-processed raw anomaly detection signal with hold-off window size of 60 datapoints. The confusion matrix values are as follows: False Positives: 7, False Negatives: 0, True Positives: 0, and True Negatives: 1925.

Figure 23: Validation results of the anomaly detection algorithm employing the CNN-based model in Figure 11a.

TP-interval is expanded to the left as follows. It is checked whether any of the nine interpolated ISLC features corresponding to a_i-th datapoint contain an interpolated value. If this is the case, let int_{start} and int_{end} be the start and the end indices of the interpolated subsequence, respectively. In our example, $int_{start} = 5$ and $int_{end} = 7$. Then, the TP-interval is given by $[int_{start}, a_i + l_{aiw}]$. All anomaly detection signals within this interval are considered as TPs. In Figure 25, each anomaly is plotted with its TP-interval visualised with light-violet rectangles in the plots with the final output of the anomaly detection algorithm.

Once the TPs are calculated, all the remaining anomaly signals are counted as FPs. Anomalies that are not identified contribute to the number of FNs. Finally, the number of TNs is obtained with the following expression:

$$\text{TN} = \text{total number of datapoints} - (\text{TP} + \text{FPs} + \text{FN}).$$

Next, the precision, recall, and F1-score are calculated in accordance with Equation 5. The precision and the recall make it possible to assess the performance of a classifier on the minority class of an imbalanced dataset [17], as both metrics are unconcerned with the majority class and only focus on the minority class. In our case the minority class is 'abnormal', i.e., TPs, while the majority class is 'normal', i.e., TNs. The quantita-

tive evaluation of the anomaly detection algorithm employing the CNN-based model is provided in Table 5.

CNN-based model, window size=150						
Exp. date	**TP**	**FP**	**FN**	**Precision**	**Recall**	**F1-score**
2022/04/06	2	0	1	1.00	0.67	0.80
2022/05/18	1	2	2	0.33	0.33	0.33
2022/05/20	1	0	0	1.00	1.00	1.00
2022/05/30	1	0	0	1.00	1.00	1.00
2022/06/01	3	9	1	0.25	0.75	0.38
2022/06/03	1	2	0	0.33	1.00	0.50
2022/06/08	2	2	0	0.50	1.00	0.67
2022/06/15	2	1	0	0.67	1.00	0.80
2022/06/22	1	0	0	1.00	1.00	1.00
2022/07/20	4	12	0	0.25	1.00	0.40

Table 5: Confusion matrix values (TP, FP, and FN) and the values of three model performance metrics (precision, recall, and F1-score) for the CNN-based model (see Figure 11a) with window size of 150 for the individual abnormal datasets.

There is a significant variance in the values of the precision, recall, and F1-score metrics for different datasets. This is mainly due to the high level of noise and the fact that the individual experiments were conducted on different days in the lab. Although we made all the effort to eliminate any external factors that could distort the generated data, given our highly limited experimental lab resources and conditions, the external factors still impacted the data. For example, as can be observed by comparing the room temperature logs from different experiments, the base room temperature was changing from one day to another. Because of our efforts, the differences were not very large. Still, they were inevitably impacting the temperature patterns recorded by the EduSat board temperature sensors. Given the 1°C resolution of the sensors, a change in the base room temperature generated differences in the temperature ranges of the profiles recorded by the individual sensors, and, even more importantly in the context of our solution, in the timings between temperature changes registered by the individual sensors. Specifically, the latter had a direct and significant impact on the values of the interpolated ISLC features.

The precision-recall curves with computed areas under the curves for individual abnormal datasets are presented in Figure 26. Since the anomaly detection domain is highly skewed, precision-recall curves are more informative than ROC (receiver operating characteristic) curves, since the latter may provide an excessively optimistic evaluation of the performance [18]. Indeed, due to the highly dominating numbers of TNs, the areas under the ROC curves are very close to 1 for all abnormal datasets. Therefore, ROC curves in this case do not reveal the actual performance of our anomaly detection algorithm.

6.2.2 Comparison with a rule-based benchmark approach

We compare the performance of our proposed AI-based solution with a classical approach to health monitoring in the space industry, i.e., a rule-based approach. For this purpose,

we develop the following benchmark procedure. As argued in Section 6.2.1, there is an inherent inertia before the excessive heat due to malfunctioning of a component reaches the sensors. Furthermore, the delay may vary for individual sensors on the board, which is caused by the differences in their distance to the malfunctioning component. With this in mind, as in the case of TP-intervals, we consider the same anomaly inertia window of length l_{aiw}. For a given datapoint at index i, if all nine sensors record a change in temperature within datapoints of indices in $[i, i + l_{aiw}]$, then this datapoint is considered positive; otherwise, it is considered negative. This classification is performed for all datapoints in the abnormal dataset. Benchmark approach raw classification of datapoints of an abnormal dataset is shown in Figure 24.

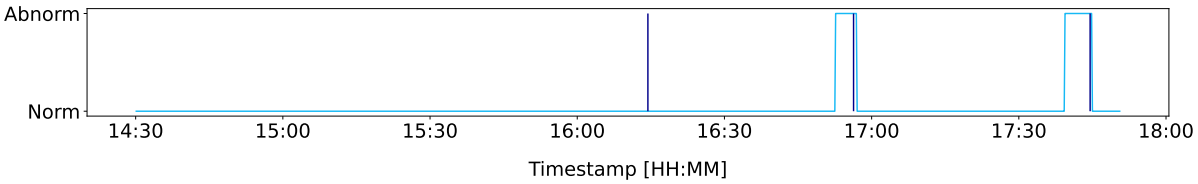
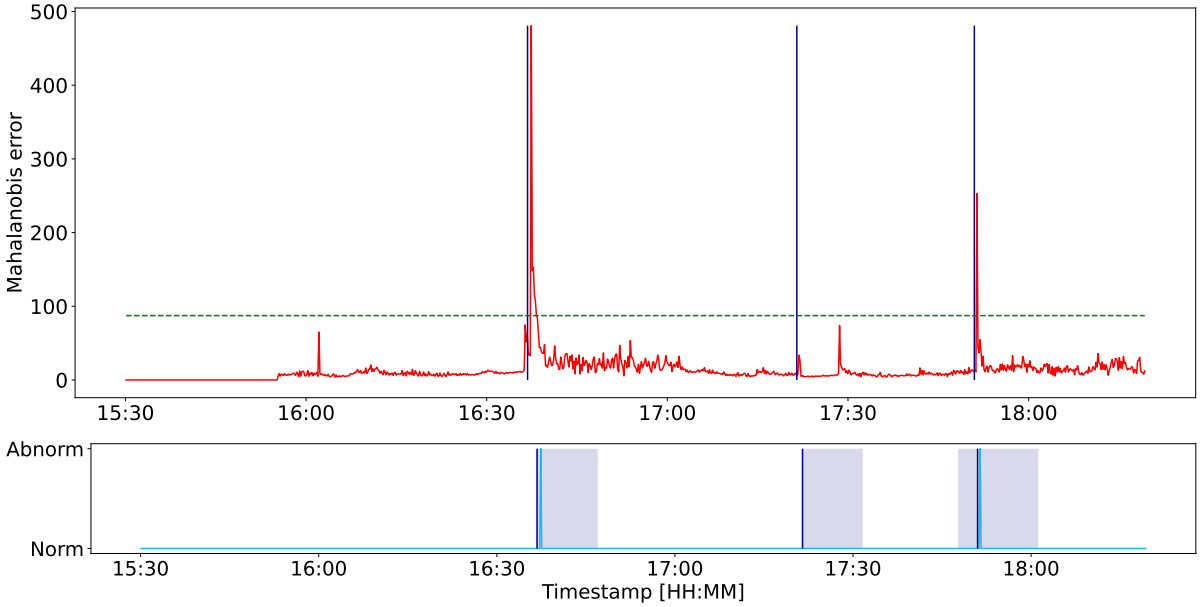


Figure 24: Raw classification of the datapoints of the abnormal dataset of 2022/05/18 by the rule-based benchmark approach. The anomaly start times are indicated with vertical dark blue lines and the classification results are shown with the cyan line.

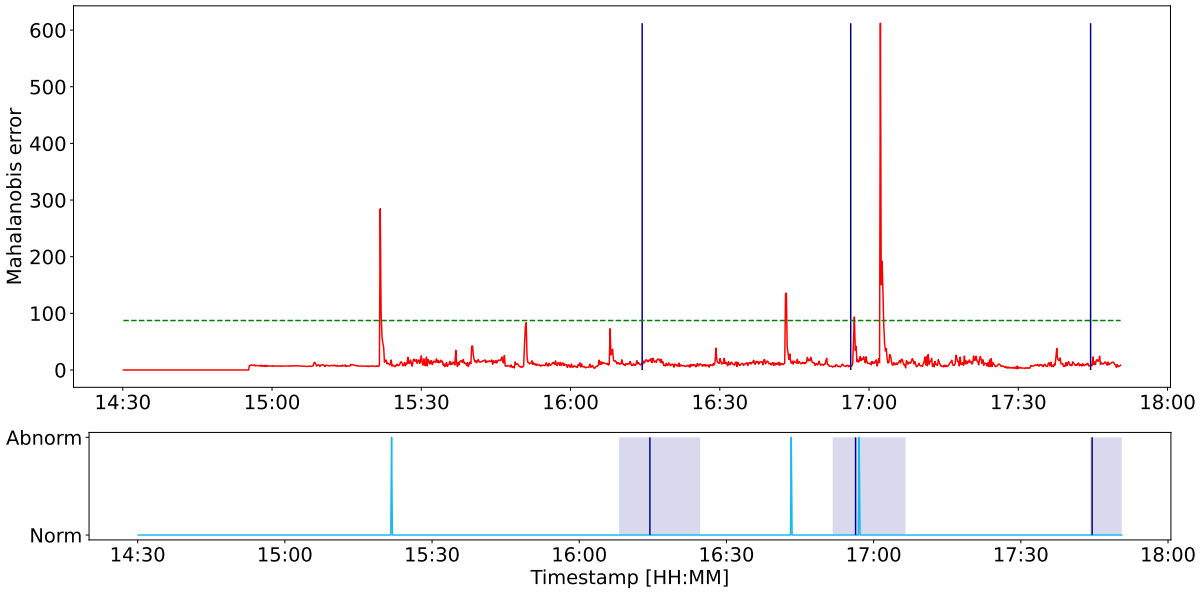
Next, contiguous subsequences of positive datapoints are considered, referred to as *positive intervals*. If a subsequence is longer than $2\,l_{aiw}$, then it is split into consecutive positive intervals of length $2\,l_{aiw}$, with the last one possibly being shorter. In the benchmark approach, we define an anomaly to be contained within a positive interval if and only if the start time of the anomaly is within the time range of the positive interval given by timestamps of its datapoints. An anomaly is considered to be correctly detected and increases the counter of TPs, if it is contained within some positive interval. If the anomaly is not contained in any of the positive intervals, the anomaly increases the counter of FNs. Finally, all positive intervals within which no anomaly is contained increase the counter of FPs. The results of applying the benchmark approach to the abnormal datasets of the experiments in Table 2 are shown in Table 6. By comparing the results to the ones in Table 5, one can clearly see that our anomaly detection algorithm outperforms the benchmark solution in most of the cases. These results also justify to some extent the need of constructing more complex, AI-based frameworks for health monitoring of CubeSats.

6.3 Performance evaluation of the microcontroller implementation

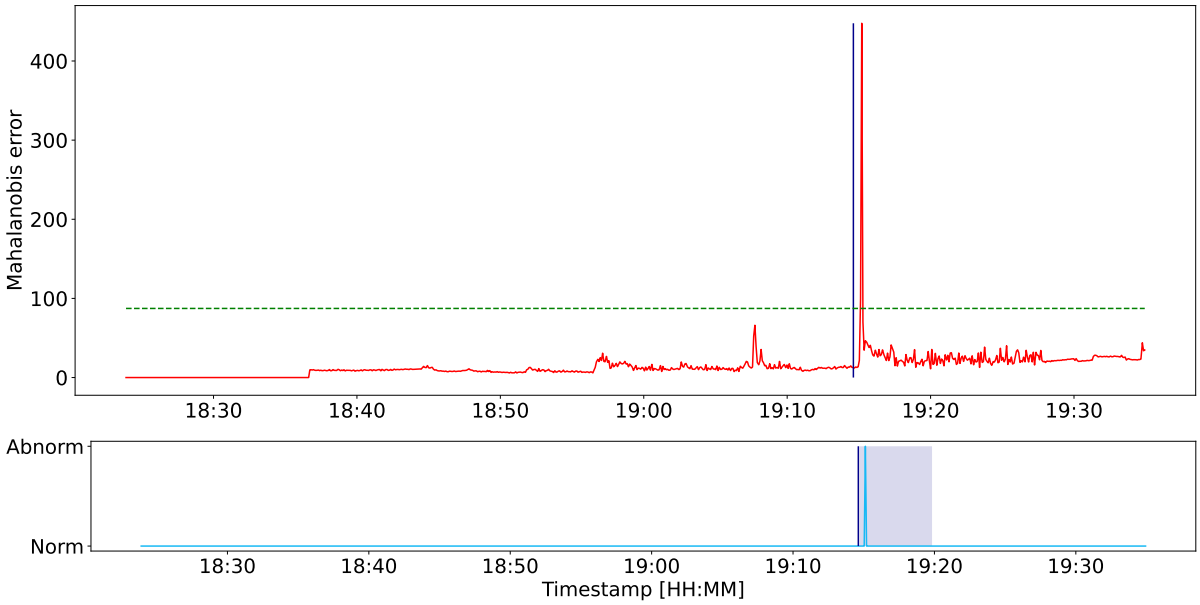
The anomaly detection algorithm employing a pre-trained CNN-based model in Figure 11a is evaluated in terms of the following three criteria: execution time, memory usage, and energy consumption. The microcontroller implementation of the deep-learning model is not quantized. Attempts to perform post-training quantization or to run quantization-aware training resulted in poor performance of the models. Therefore, all the results presented here are obtained with the deep-learning model implemented with float32.



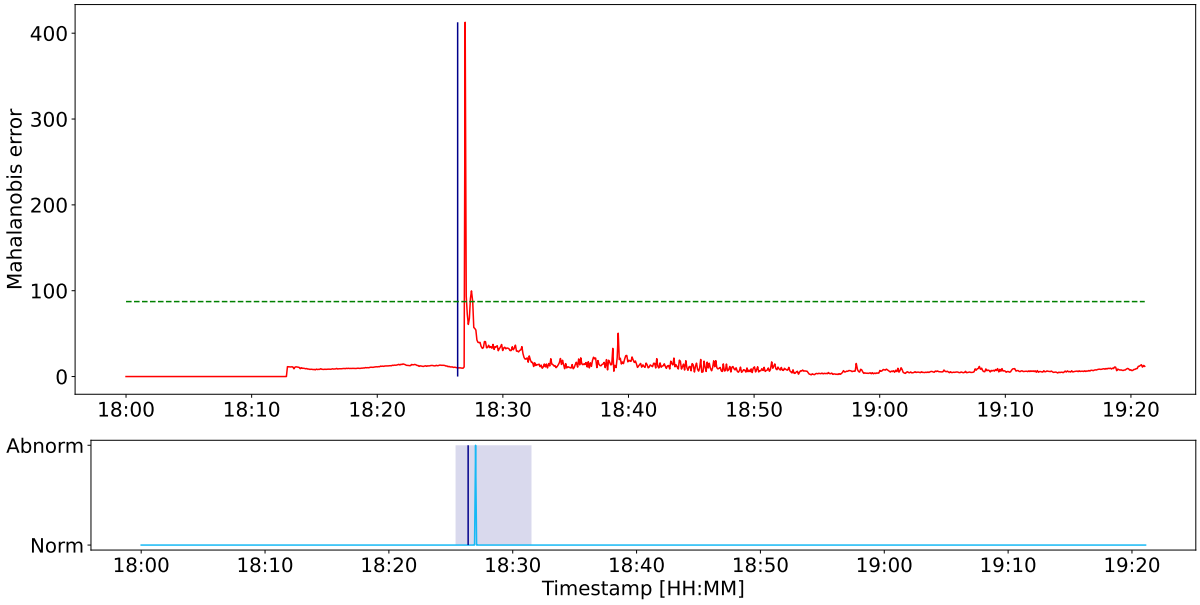
(a) Anomaly detection results for the abnormal dataset of the 2022.04.06 experiment.



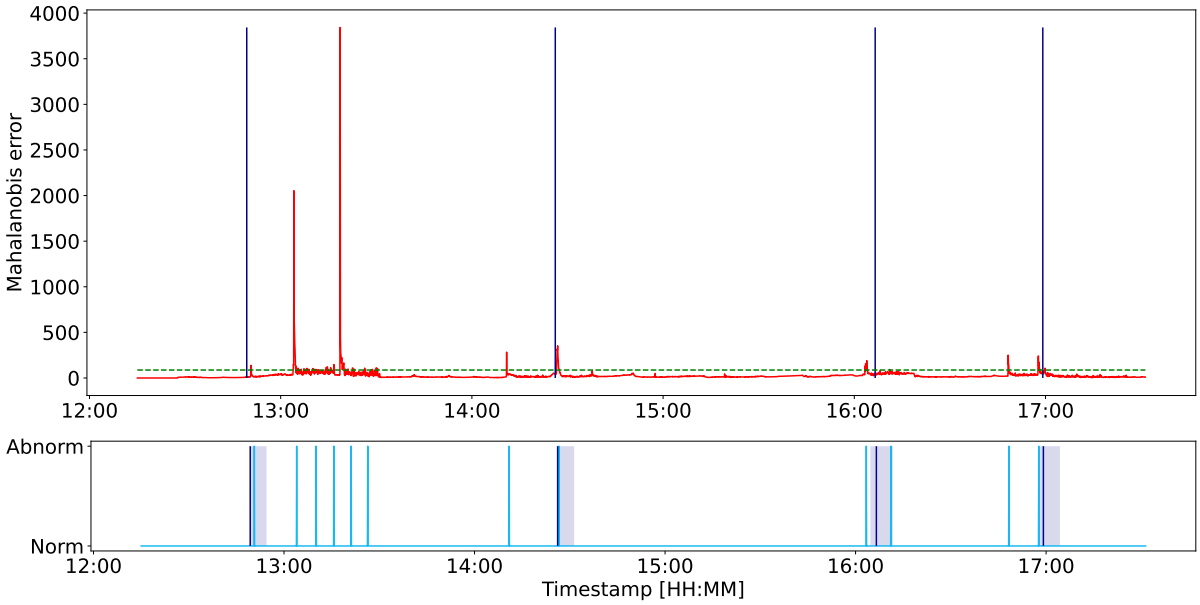
(b) Anomaly detection results for the abnormal dataset of the 2022.05.18 experiment.



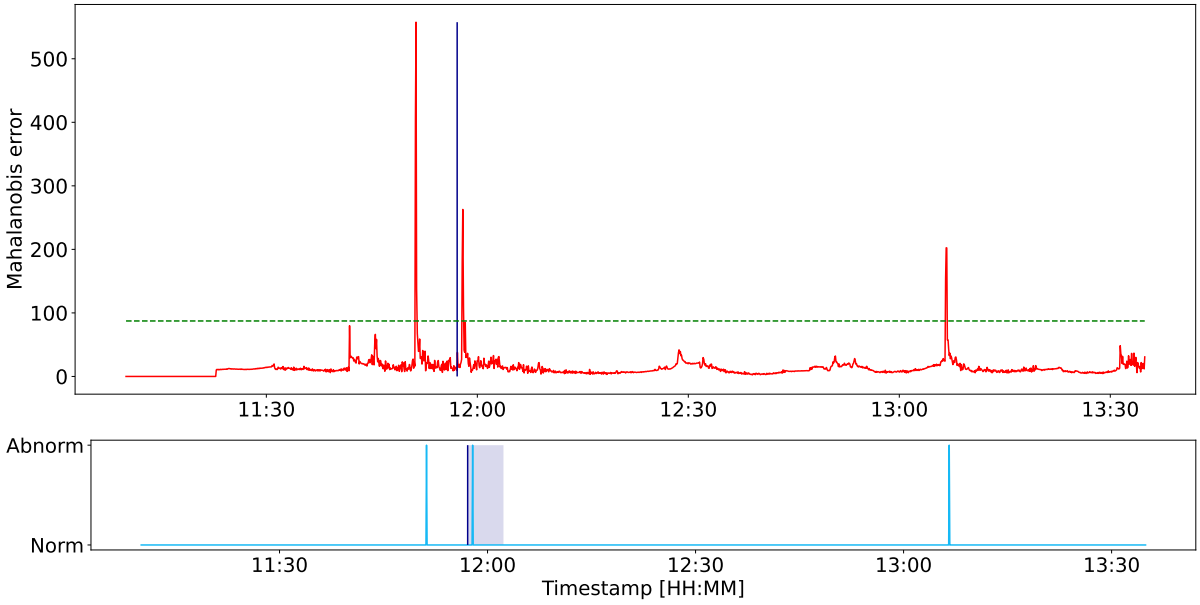
(c) Anomaly detection results for the abnormal dataset of the 2022.05.20 experiment.



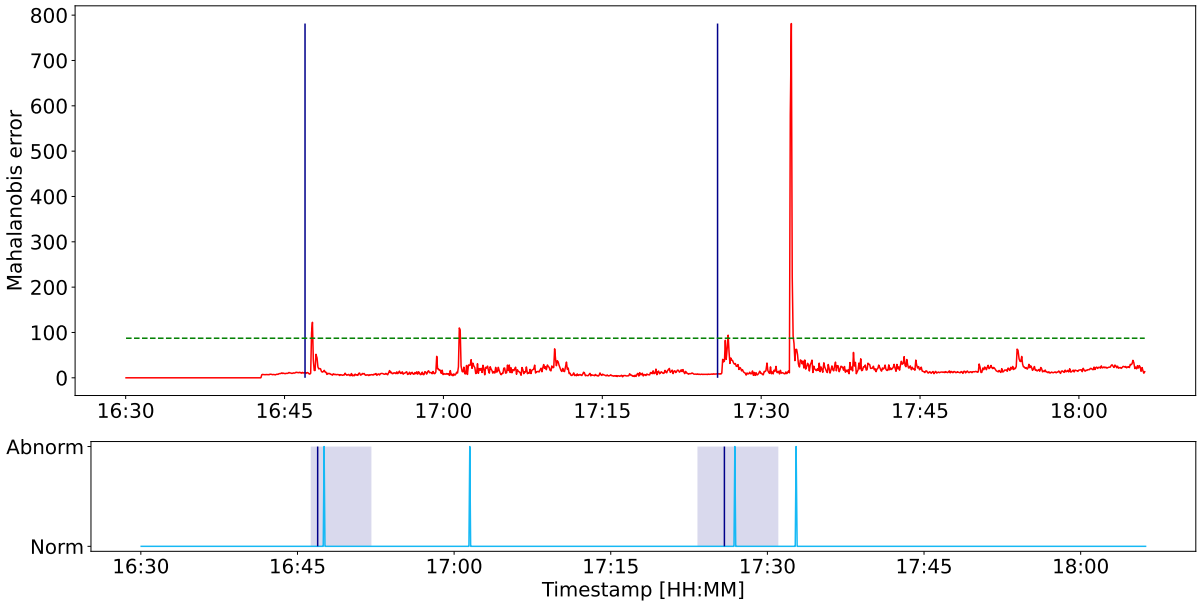
(d) Anomaly detection results for the abnormal dataset of the 2022.05.30 experiment.



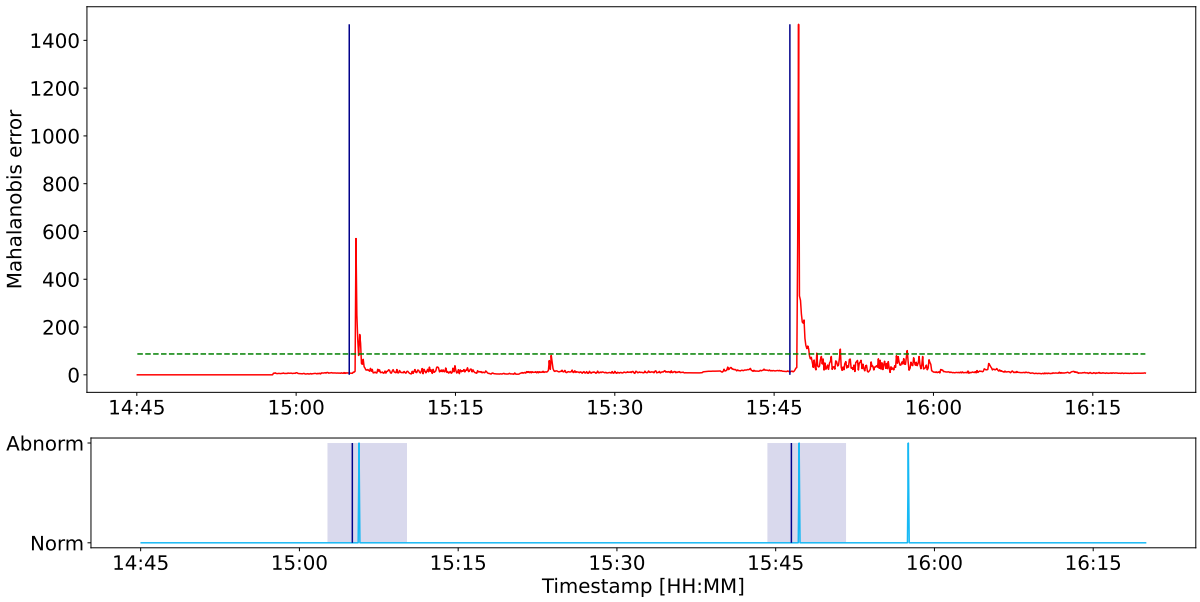
(e) Anomaly detection results for the abnormal dataset of the 2022.06.01 experiment.



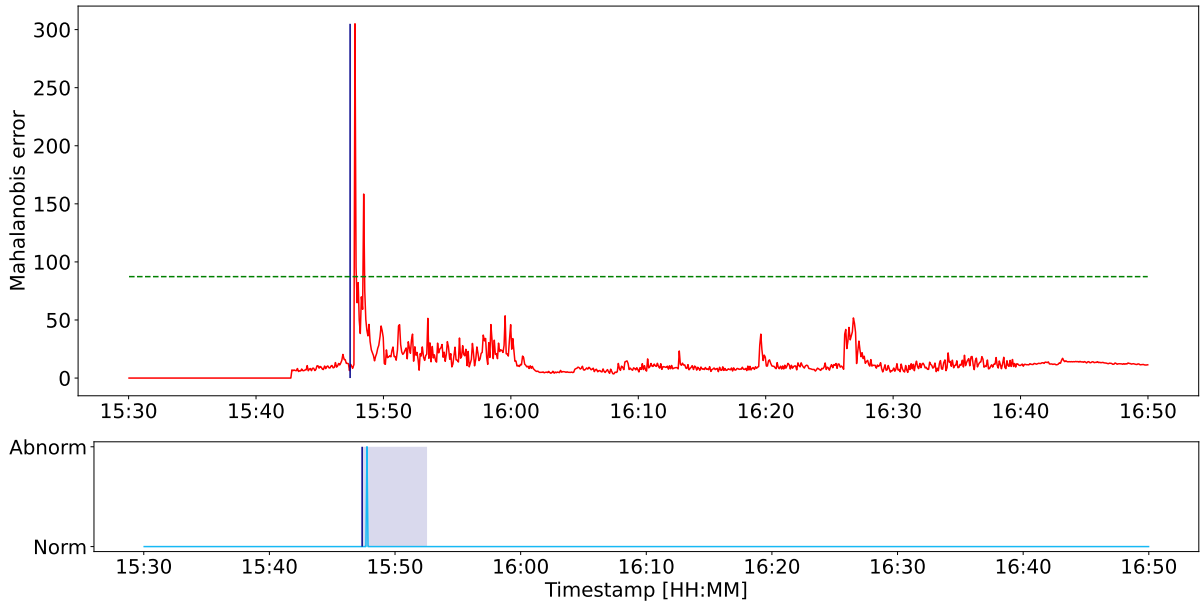
(f) Anomaly detection results for the abnormal dataset of the 2022.06.03 experiment.



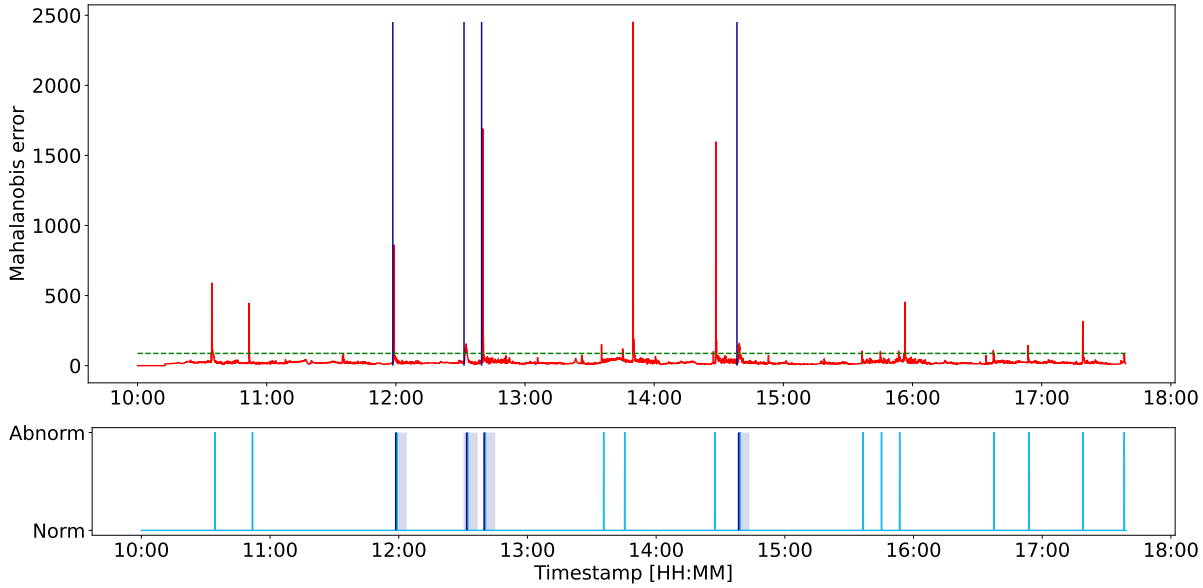
(g) Anomaly detection results for the abnormal dataset of the 2022.06.08 experiment.



(h) Anomaly detection results for the abnormal dataset of the 2022.06.15 experiment.

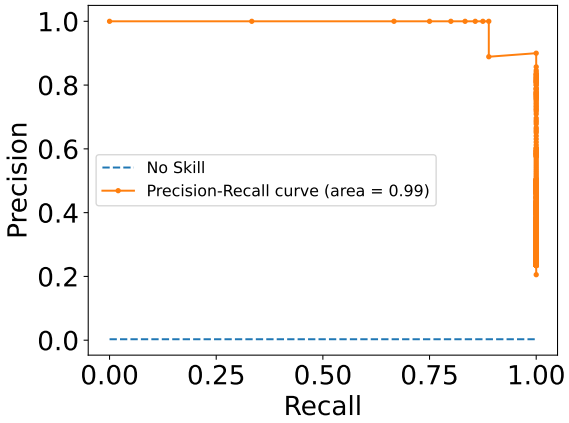


(i) Anomaly detection results for the abnormal dataset of the 2022.06.22 experiment.

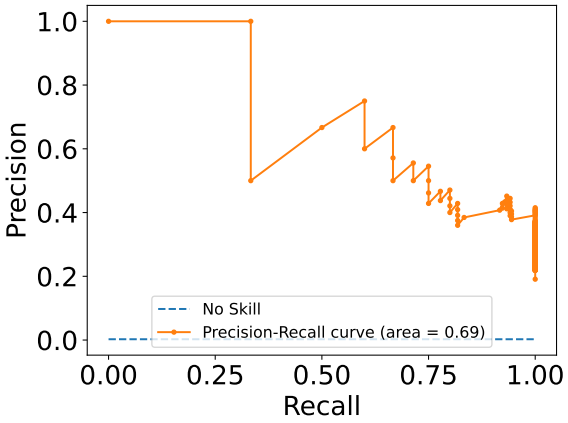


(j) Anomaly detection results for the abnormal dataset of the 2022.07.20 experiment.

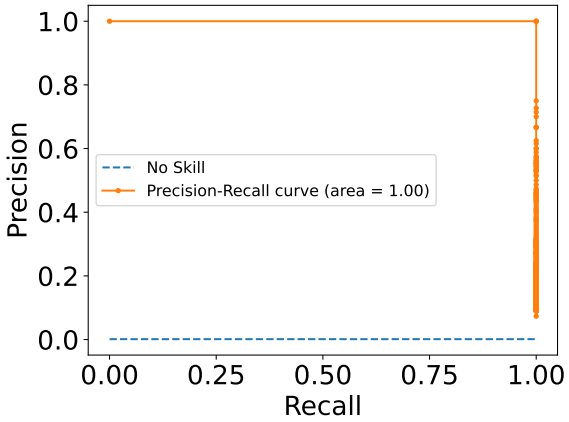
Figure 25: Anomaly detection results of the anomaly detection algorithm employing the CNN-based model in Figure 11a. For each experiment two plots are shown: the raw anomaly detection signal, i.e., the Mahalanobis error (above) and the raw signal after post-processing, i.e., the final output of the anomaly detection algorithm (below). The plots contain the Mahalanobis error (red), the Mahalanobis error threshold corresponding to the 0.05 significance level (horizontal dashed green line), the anomaly start times (vertical dark blue lines), the True Positive intervals (light-violet rectangles), and the anomaly detection signal (cyan).



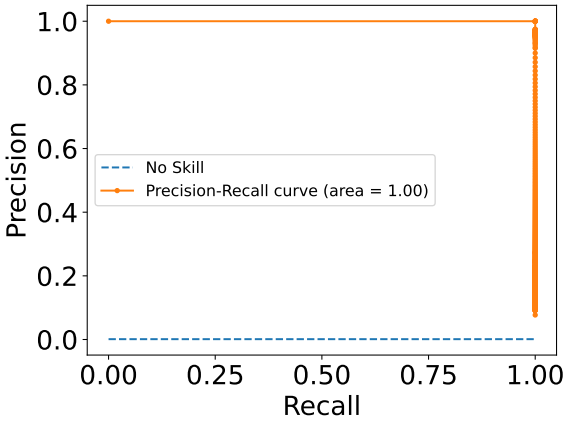
(a) Abnormal dataset of 2022.04.06.



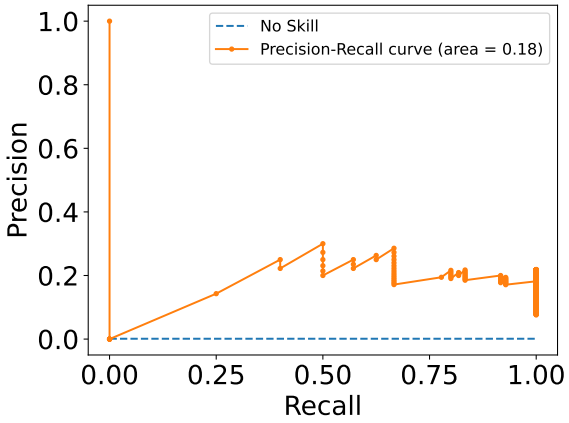
(b) Abnormal dataset of 2022.05.18.



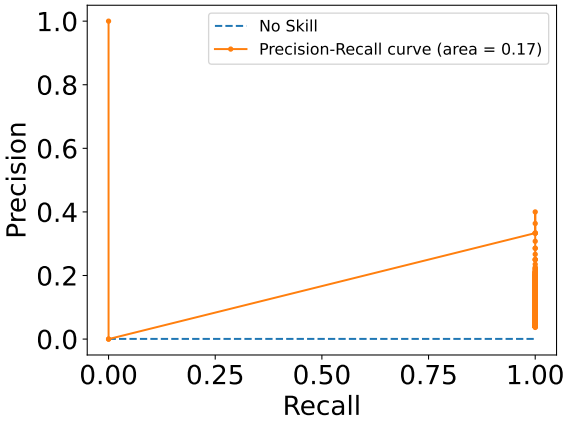
(c) Abnormal dataset of 2022.05.20.



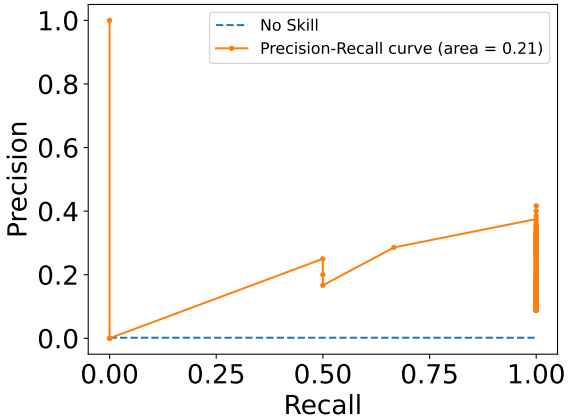
(d) Abnormal dataset of 2022.05.30.



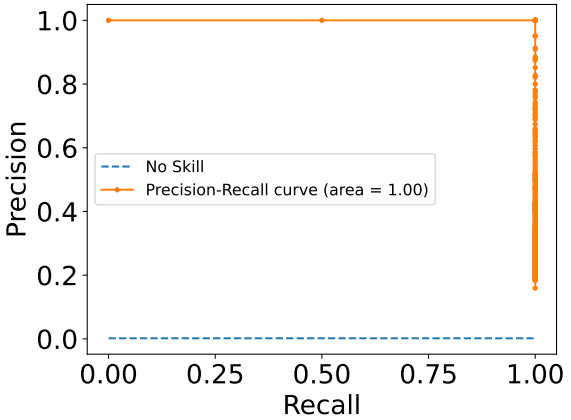
(e) Abnormal dataset of 2022.06.01.



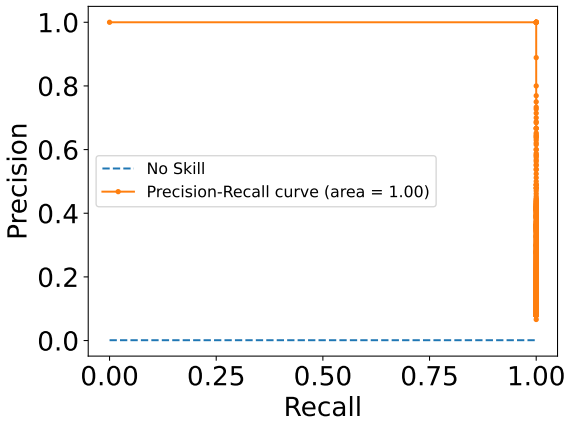
(f) Abnormal dataset of 2022.06.03.



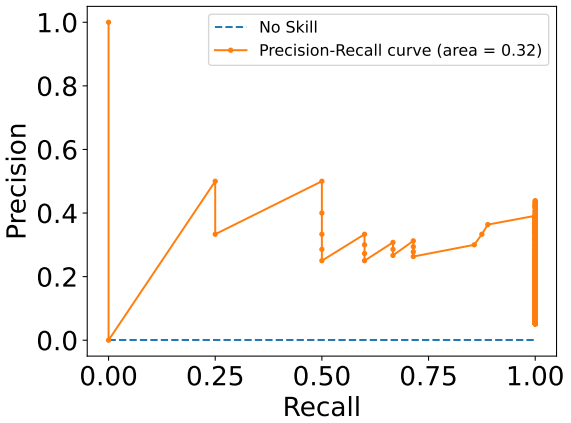
(g) Abnormal dataset of 2022.06.08.



(h) Abnormal dataset of 2022.06.15.



(i) Abnormal dataset of 2022.06.22.



(j) Abnormal dataset of 2022.07.20.

Figure 26: Precision-recall curves (orange) obtained for the anomaly detection algorithm with the CNN-based model (Figure 11a) for individual abnormal datasets. The No Skill lines (dashed blue) indicate the proportion of the number of anomalies to the total number of datapoints in the anomalous datasets. The areas under the precision-recall curves for the CNN-based model are provided in the legends.

Benchmark rule-based solution						
Exp. date	**TP**	**FP**	**FN**	**Precision**	**Recall**	**F1-score**
2022/04/06	1	0	2	1.00	0.33	0.50
2022/05/18	2	0	1	1.00	0.67	0.80
2022/05/20	0	0	1	0.00	0.00	0.00
2022/05/30	0	0	1	0.00	0.00	0.00
2022/06/01	0	1	4	0.00	0.00	0.00
2022/06/03	0	0	1	0.00	0.00	0.00
2022/06/08	0	0	2	0.00	0.00	0.00
2022/06/15	2	0	0	1.00	1.00	1.00
2022/06/22	1	0	0	1.00	1.00	1.00
2022/07/20	2	0	2	1.00	0.50	0.67

Table 6: Confusion matrix values (TP, FP, and FN) and the values of three model performance metrics (precision, recall, and F1-score) for the rule-based benchmark solution for the individual abnormal datasets.

In order to collect larger samples of measurements, we run the anomaly detection algorithm on the microcontroller on merged both normal and abnormal datasets of the individual experiments. We refer to the merged datasets as the *experimental dataset* of the given experiment.

6.3.1 Anomaly detection algorithm execution time measurements

We performed detailed measurements of the execution times of the anomaly detection algorithm run on the microcontroller with different settings of the clock frequency. The exact part of the anomaly detection algorithm which execution was considered in the measurements is indicated with the *'Measured time'* frame in Figure 14. The execution times were determined for the processing of each datapoint of the individual experimental datasets. For example, the execution times for each datapoint of the 2022/04/06 experimental dataset processed on the microcontroller with clock frequency set to 298 MHz are shown in Figure 27.

Notice that for some datapoints the execution times are close to 0. These times are just needed for queuing the datapoints without computation of the interpolated ISLC features and without the execution of the core part of the anomaly detection algorithm, i.e., the parts shown inside the *'repeat for 0..N vectors in IISLCC queue'* and *'repeat for 0..N in queue'* frames in Figure 14. The processing of these points is postponed till the moment when interpolated ISLC feature values can be computed, i.e., when a temperature change by any of the temperature sensors is detected at the arrival of some future datapoint. Once temperature change is observed, the interpolated ISLC features are computed and the anomaly detection algorithm is run for all the queued datapoints, see Section 5.3.1 for details. Therefore, the subsequences of near 0 execution times are followed by execution time peaks associated with datapoints which resulted in change detection and dequeuing. The peaks reflect the processing of all the queued datapoints.

In Table 7, histograms with anomaly detection algorithm execution times for individual experiments are shown for four different settings of the microcontroller's clock

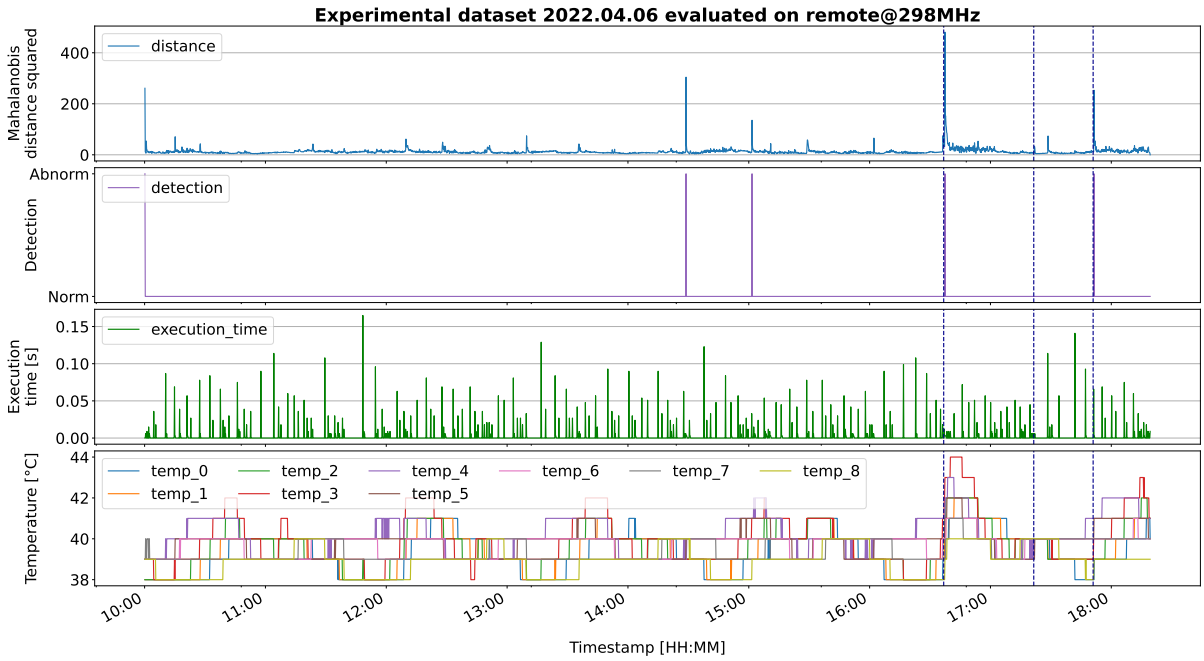
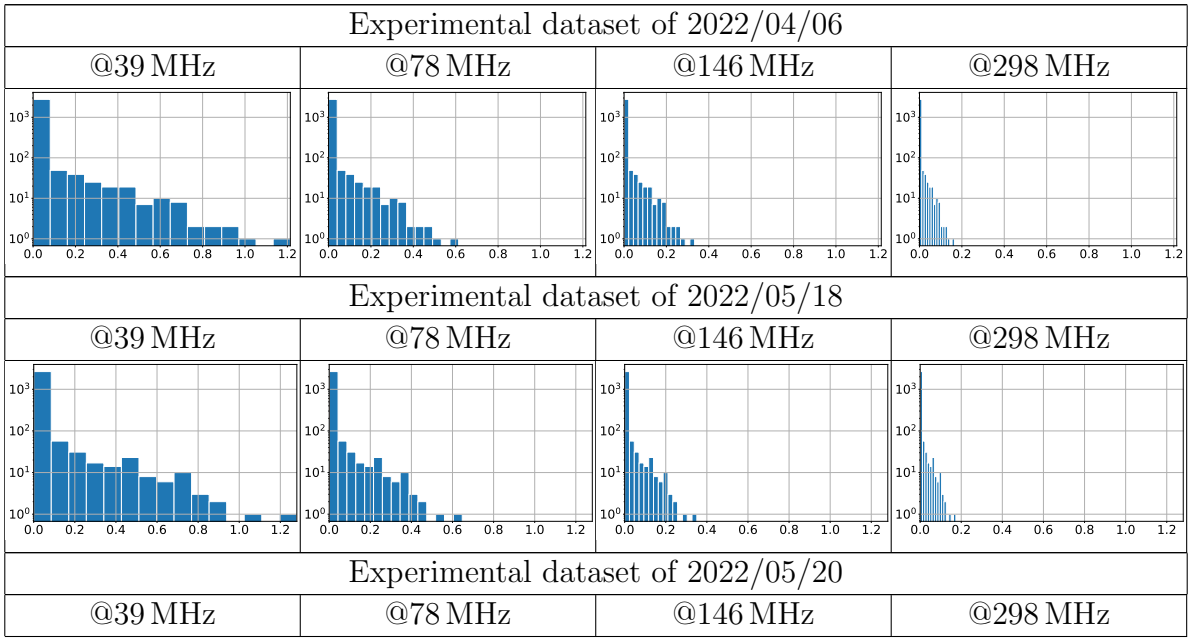
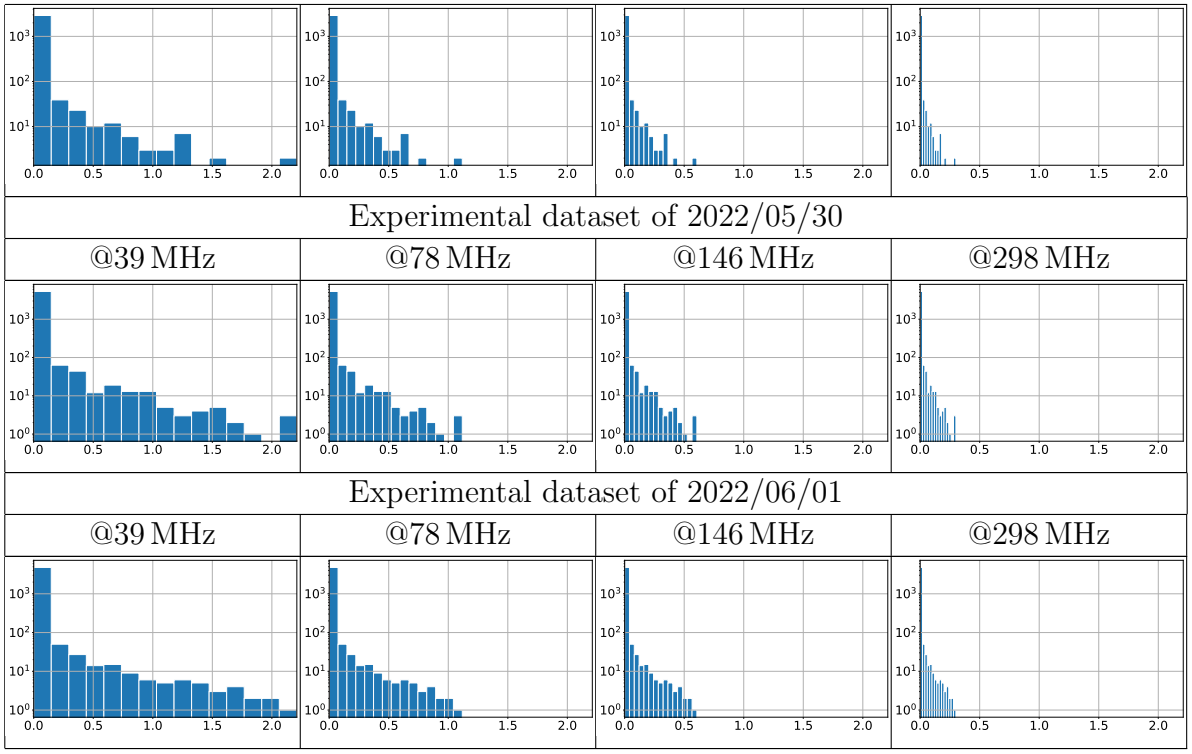


Figure 27: Execution time measurements in seconds for the anomaly detection algorithm run on the microcontroller with clock frequency set to 298 MHz while analysing the 2022/04/06 experimental dataset. From top to bottom: raw Mahalanobis error of the inference, output of the anomaly detection algorithm, execution time for each datapoint, and temperature values recorded by the nine EduSat board temperature sensors.

frequency, i.e., 39 MHz, 78 MHz, 146 MHz, and 298 MHz. The mean execution times are shown in Figures 28 and 29.




Experimental dataset of 2022/05/30
@39 MHz
@78 MHz
@146 MHz
@298 MHz
Experimental dataset of 2022/06/01
@39 MHz
@78 MHz
@146 MHz
@298 MHz

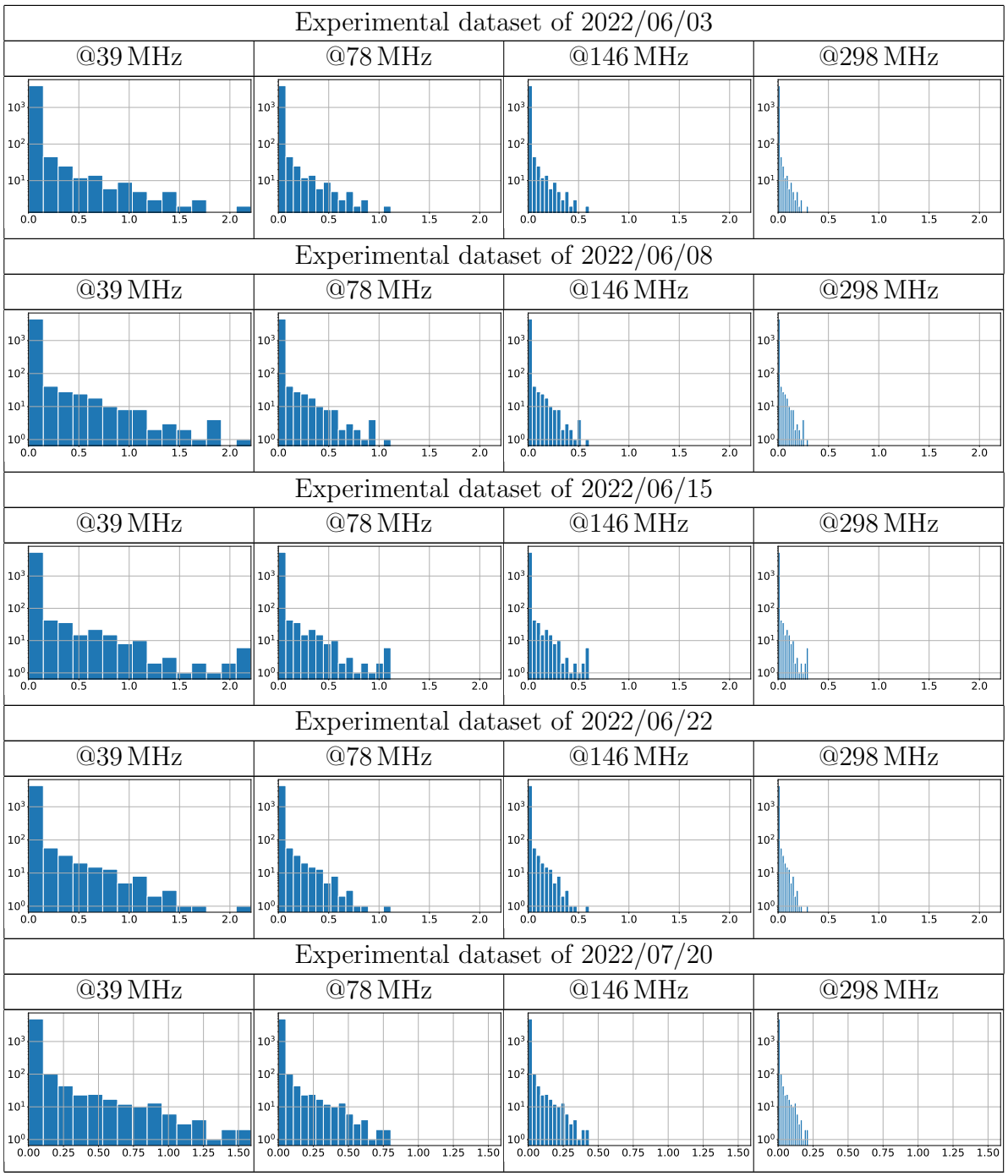


Table 7: Histograms of the anomaly detection algorithm execution times for each datapoint of the individual experimental datasets with different settings of the microcontroller's clock frequency. For all histograms, the x-axis is time in seconds and the y-axis is the logarithm of the count of datapoints processed with times in ranges associated with the respective histogram bars.

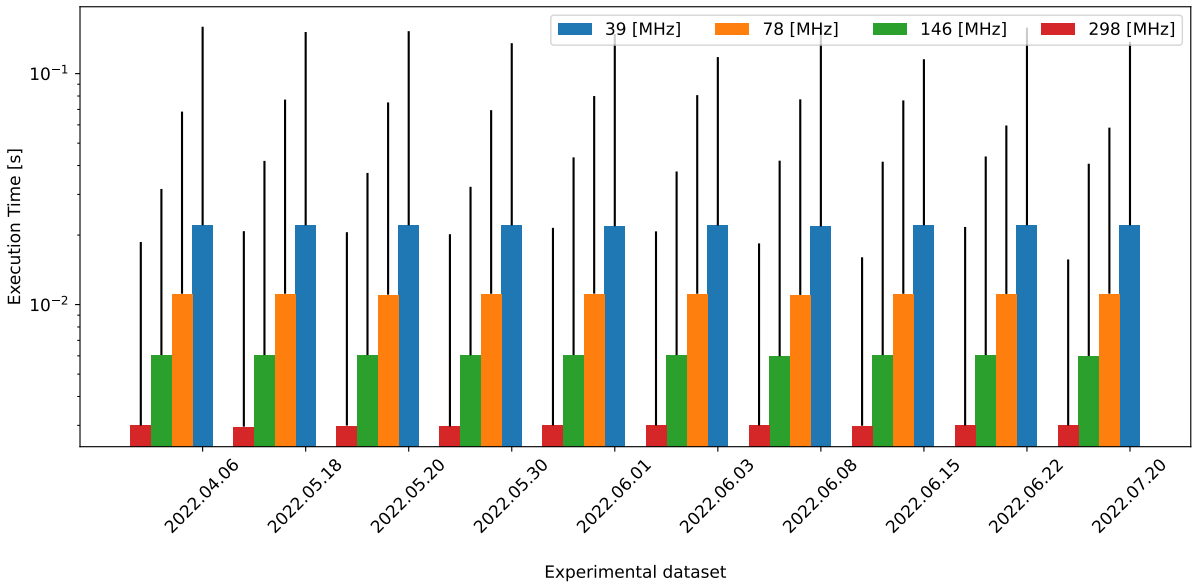


Figure 28: Mean execution times of the anomaly detection algorithm run on individual experimental datasets for four different settings of the microcontroller's clock frequency, i.e., 39 MHz, 78 MHz, 146 MHz, and 298 MHz. Standard deviations are shown with black vertical lines.

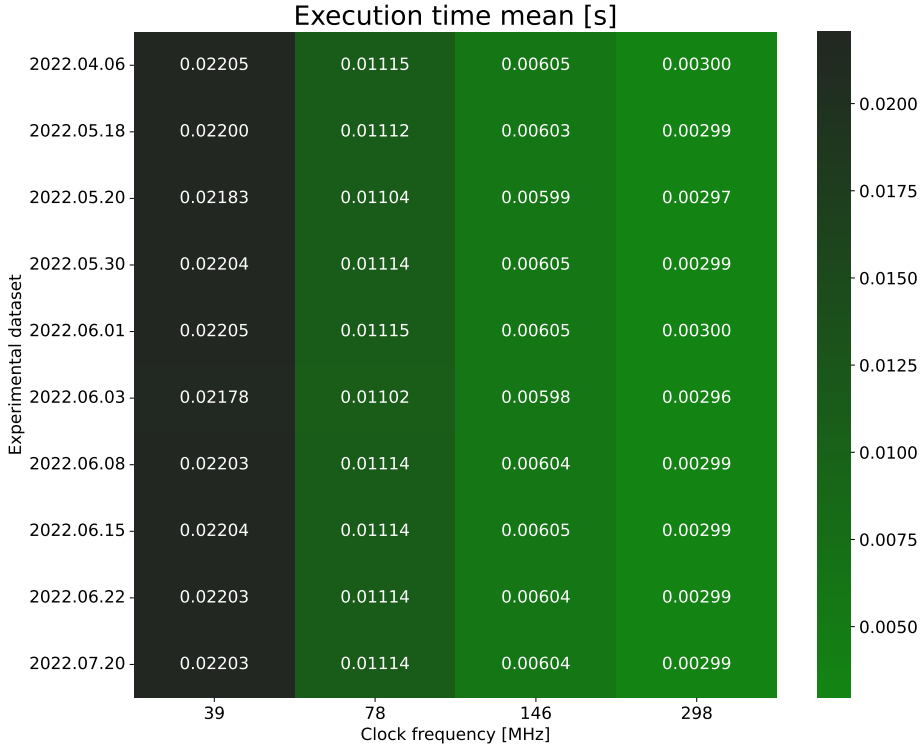
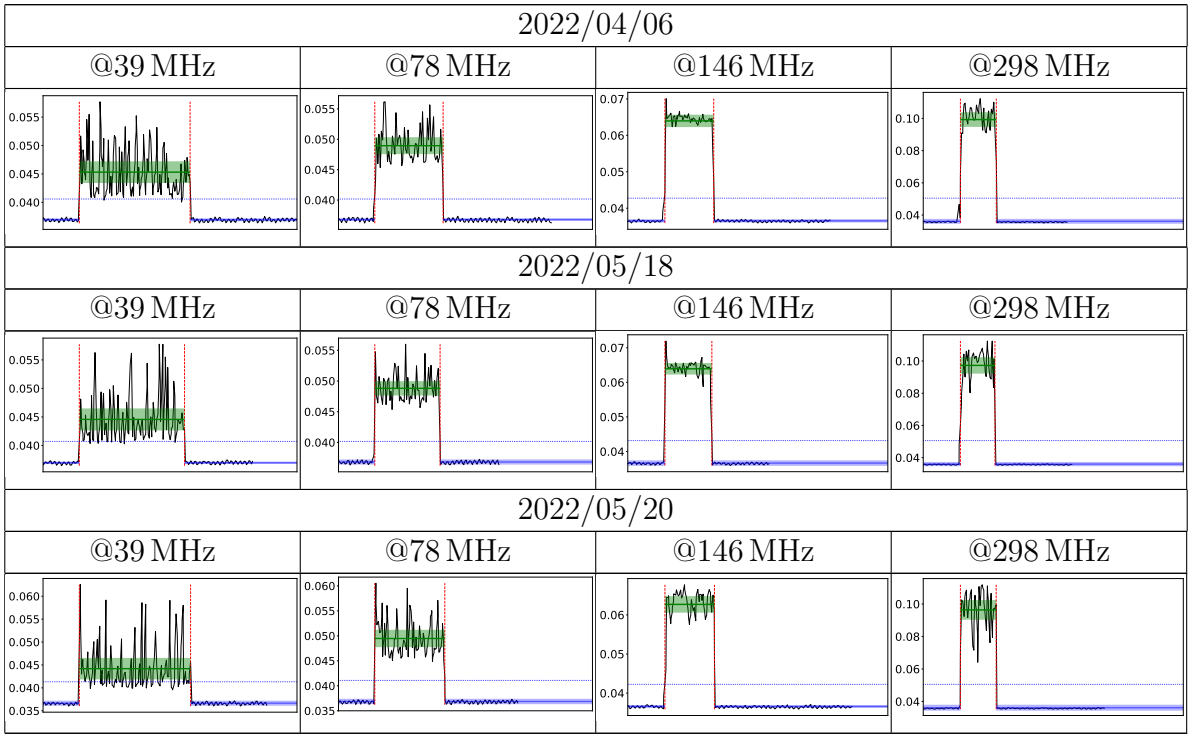


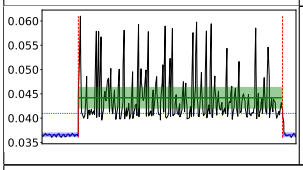
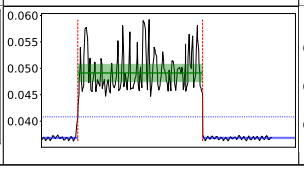
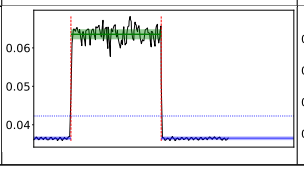
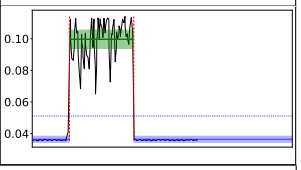
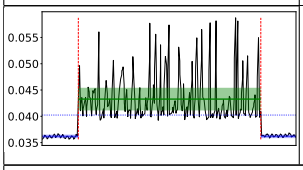
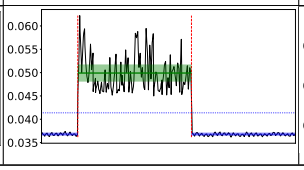
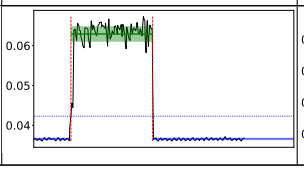
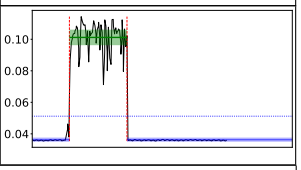
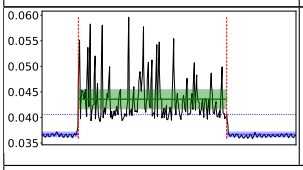
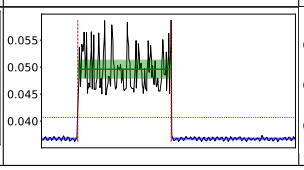
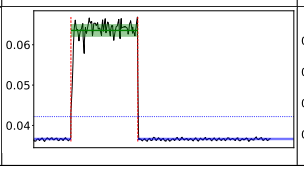
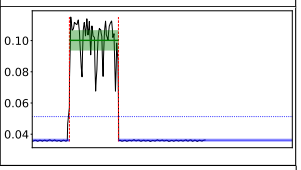
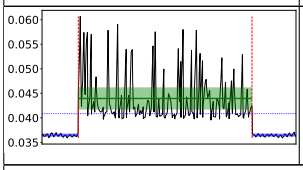
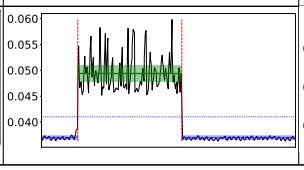
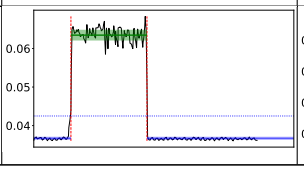
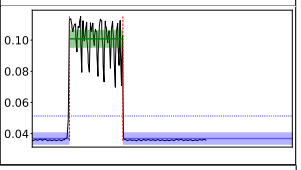
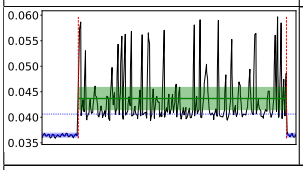
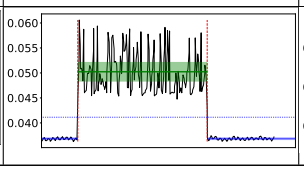
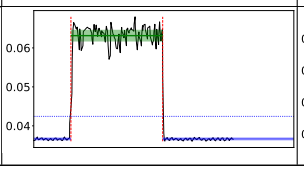
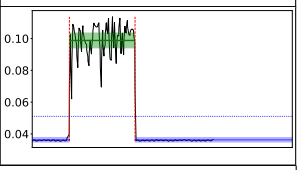
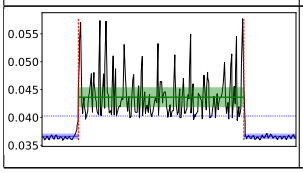
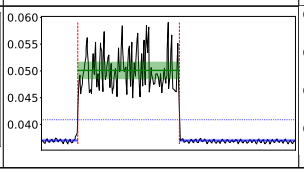
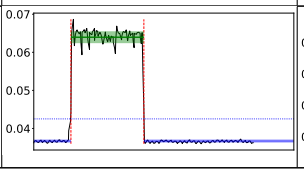
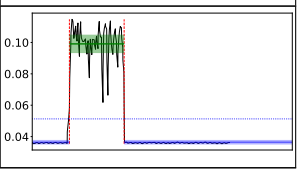
Figure 29: Mean execution times of the anomaly detection algorithm run on individual experimental datasets with four different settings of the microcontroller's clock frequency.

6.3.2 Power consumption measurements

Power consumption of the microcontroller while running the testbed with the anomaly detection algorithm on individual experimental datasets under different settings of clock frequency is presented in Table 8. Each measurement consists of three phases: 1) the microcontroller running in idle state, 2) the microcontroller running the testbed with the anomaly detection algorithm on an experimental dataset, and 3) the microcontroller again in idle state. The average of the instantaneous power over the processing of one full experimental dataset, i.e., over the second phase, is referred to as *experiment mean power*. The numerical values of experiment mean powers for individual experimental datasets under different settings of clock frequency are provided in Figure 30.

Measurements of *background power*, i.e., the idle state instantaneous power consumption, are included for reference and for the calculation of *net experiment mean power* consumed by the testbed with the anomaly detection algorithm. The net experiment mean power is calculated as the experiment mean power minus the background mean power, i.e., the average of background power measurements. The net experiment mean power consumptions with the standard deviations calculated in accordance with the expression for the standard deviation of a sum/difference of two uncorrelated random variables, i.e., $\sigma_{X-Y} = \sqrt{(\sigma_X^2 + \sigma_Y^2)}$, are shown in Figure 31.



2022/05/30			
@39 MHz	@78 MHz	@146 MHz	@298 MHz
			
2022/06/01			
@39 MHz	@78 MHz	@146 MHz	@298 MHz
			
2022/06/03			
@39 MHz	@78 MHz	@146 MHz	@298 MHz
			
2022/06/08			
@39 MHz	@78 MHz	@146 MHz	@298 MHz
			
2022/06/15			
@39 MHz	@78 MHz	@146 MHz	@298 MHz
			
2022/06/22			
@39 MHz	@78 MHz	@146 MHz	@298 MHz
			

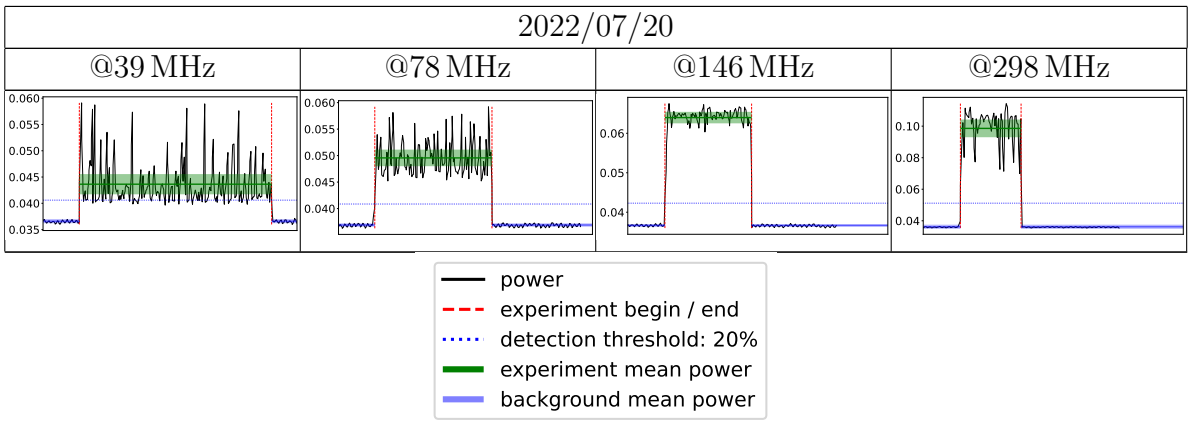


Table 8: Plots of the microcontroller's power consumption for different settings of the clock frequency for individual experimental datasets. Power (black lines) is calculated from current and voltage values recorded by two multimeters every 1 s. For all plots, the y-axis is power in watts [W]. The x-axis is the time of current and voltage measurements, which is unrelated to the presented results and therefore not shown for better readability. However, for comparison purposes, the x-axis range spans the same time interval for all plots: starts 30 s prior to the experiment begin (left vertical red dashed line) and ends 180 s after the experiment begin line. The numerical values corresponding to the heights of the green lines representing experiment mean power are provided in Figure 30.

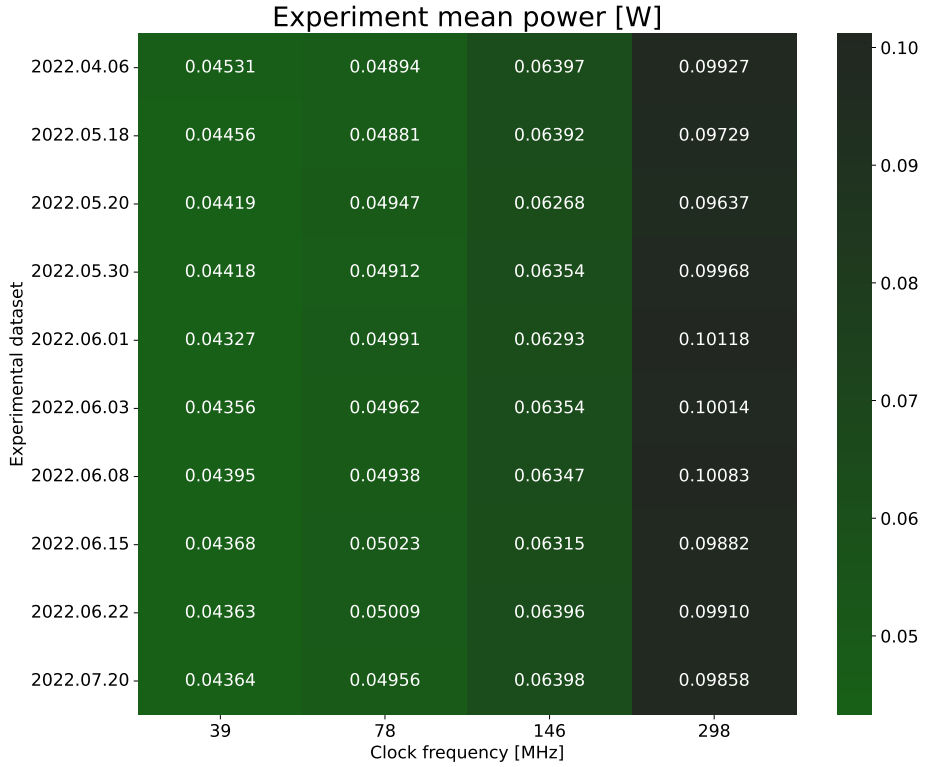


Figure 30: Experiment mean power consumed by the microcontroller while running the testbed with the anomaly detection algorithm on individual experimental datasets with different clock frequencies.

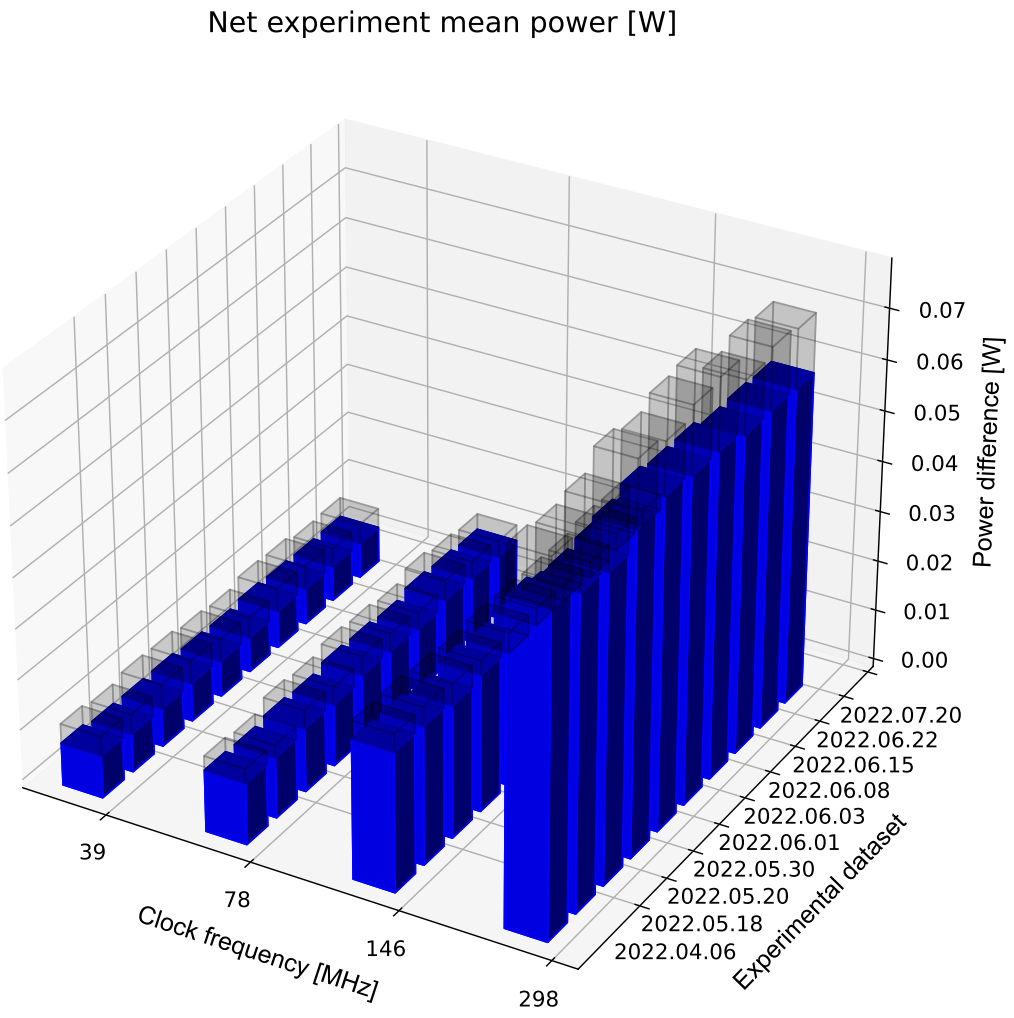


Figure 31: Net experiment mean power consumed by the testbed with the anomaly detection algorithm while running on the microcontroller and processing individual experimental datasets with different clock frequencies. The net experiment mean power is given by the blue bars. The grey bars represent the means plus standard deviations.

6.3.3 Memory footprint

The implementation of the anomaly detection algorithm is using both static memory and stack. To prevent fragmentation, neither dynamic memory nor heap are used.

To measure the stack and static memory used by the anomaly detection algorithm, a placebo version of the firmware is implemented. This version contains the complete testbed without the algorithm itself. The placebo version can be compiled using *make PLACEBO=1* for the PC version or by selecting the PLACEBO profile in the microcontroller firmware.

To find the net stack and static memory usage by the algorithm alone, the stack and static memory used by the placebo version are subtracted from the stack and static memory used by the complete version, i.e., testbed with the algorithm, respectively.

Stack

To measure the stack of either version, the relevant memory range is initialised by overwriting it with a commonly used magic value (*0xDEADBEEF*) prior to running the firmware and checking the amount of modified bytes afterwards. The initialisation of the memory range is done in the startup assembly code available in the *Core/Startup/ startup_stm32h743zitx.s* subdirectory of the firmware directory. Checking stack usage is done using a debugger. Results of the measurements are provided in Table 9.

uC firmware	**stack end address**	**stack dirty begin address**	**stack used [bytes]**
uC::complete	0x24080000	0x2407f4f0	2832
uC::placebo	0x24080000	0x2407fdb0	592
algorithm = complete - placebo	–	0x8C0	**2240**

Table 9: Stack usage of the complete and the placebo uC firmwares. The third line shows the stack used by the AtMonSat anomaly detection algorithm.

Static memory

Static memory usage for each firmware section is measured using the GNU size utility (e.g., *size -d atmonsat_firmware*) and listed in Table 10 for different firmware versions and targets.

On the microcontroller the data and bss sections are stored in the memory RAM_D1 (512kB). The text section is stored in flash (2MB). All other memory blocks are unused. This partitioning is defined in the proper linker script. The percentage values give in Table 10 are calculated based on the respective sizes of the corresponding memory.

7 Conclusions

Within the AtMonSat project the problem of on-board fault detection using artificial neural networks for CubeSat systems and related spacecraft with limited computing resources

firmware	text	data	bss	flash		RAM_D1	
PC::complete	1096106	6256	58184	–	–	–	–
PC::placebo	77562	1968	2792	–	–	–	–
PC::algorithm = complete - placebo	1018544	4288	55392	–	–	–	–
uC::complete	303948	864	51488	297.67kB	14.53%	51.06kB	9.97%
uC::placebo	176140	840	15944	127.83kB	8.44%	16.32kB	3.19%
uC::algorithm = complete - placebo	127808	24	35544	168.84kB	6.09%	34.74kB	6.78%

Table 10: Static memory usage of the complete and placebo firmwares compiled for PC and the microcontroller (uC). The third and the sixth line show the static memory used by the AtMonSat anomaly detection algorithm for PC and the uC versions, respectively.

was considered. The concrete problem scenario of malfunctioning of CubeSat board elements was investigated. Experiments for generating telemetry data for this particular scenario were devised and performed with EduSat – a near-flight engineering model of the CubeSat Lab at the University of Luxembourg. Next, an artificial neural network-based anomaly detection algorithm for the considered concrete problem scenario was proposed. Various deep-learning architectures were investigated and the one providing the best performance results, yet compliant with CubeSats' limited computational resources, was selected. The anomaly detection algorithm employing the chosen CNN-based deep-learning model was implemented on a development board with the STM32H743ZI microcontroller unit which was the same as the one in EduSat. Measurements of power consumption, anomaly detection algorithm execution time, and memory usage were performed. The AtMonSat anomaly detection algorithm was also tested in conjunction with a preemptive FreeRTOS kernel. The presented results allowed us to draw the conclusion that the proposed solution is both effective and suitable for implementation on a CubeSat system.

Nevertheless, the simulation of the in-orbit environment for acquisition of relevant telemetry data posed several considerable challenges given our limited lab resources and conditions. In particular, due to the nature of the considered subset of telemetry data, i.e., thermal data, the experiments were time consuming. First, they required the EduSat's temperature to stabilise under illumination. Second, to simulate normal LEO conditions, the satellite was rotating at the velocity of one full turn per 90 min. The generation of a normal dataset required at least few full rotations. Finally, the abnormal conditions were simulated by turning on the EduSat's battery heater. Each such anomaly was followed by a necessary period of normal conditions in order for the thermal pattern to get back to normal, which was a rather slow process. Therefore, the number of anomalies that could be produced in a one-day experimental session was highly limited.

The lab generated data were noisy. Given our limited resources, we were not able to eliminate all external factors that were distorting the temperature data. Although we tried to reduce the impact of theses factors as much as possible, we were not able to eliminate variations in the room temperature between different days on which experiments were conducted. These fluctuations had impact both on the range and the timings between changes of the temperatures recorded by the individual sensors of the EduSat's board.

Another issue was related to the way anomalies were introduced. Having in mind the protection of the valuable EduSat, we were reluctant to introduce any kind of additional elements to the EduSat's board that would simulate faulty components since this would constitute a significant risk to the satellite's hardware. In consequence, we could not simulate failures at different locations on the EduSat's board.

Given the challenges related to data generation, we were not able to obtain more clean lab-generated datasets for model training, evaluation, and testing. Nevertheless, our solution proved effective even with the noisy data. As future work, one could consider the acquisition of higher quality lab data, or even real in-orbit data, and the fine-tuning of our current solution based on them.

It is worth emphasising that the two types of data considered within the AtMonSat project, i.e., the synthetic data and lab data, represent two extremes: ideally regular, clean data and irregular, highly noisy data, respectively. As shown in this report, our solution is effective on both types. Since it is justifiable to assume that target in-orbit data would be positioned between the two opposites, our anomaly detection algorithm would presumably generate less False Positive outcomes for real in-orbit data than in the case of the lab data. Verification of this would require access to real-life CubeSat mission telemetry data.

Another potential future research direction would be to try to further reduce the number of False Positive cases output by the AtMonSat anomaly detection algorithm. For example, this could be achieved by considering an ensemble of independent light-weighted models which outputs would be aggregated to provide the final classification decisions.

The problem of model quantization is another issue that could be further investigated in the future. In our case, the quantization did introduce a significant drop in performance for the real lab datasets. Therefore, we decided to implement the final solution using TensorFlow Lite Micro without quantization. However, one could search for other model architectures which would provide good results after applying quantization. This would result in a solution requiring even less memory and computational power.

Finally, our solution could be extended with additional mechanisms for the anomaly root cause identification. Currently, the detection of an anomaly provides evidence that an additional heat source stemming from faulty operation of some component on the EduSat's board. However, once an anomaly is identified, additional mechanisms could be triggered which, based on data for other telemetry attributes, would indicate the malfunctioning component. Development of such mechanisms could be considered as a follow-up of the AtMonSat project.

References

[1] The CubeSat Program, Cal Poly SLO, "CubeSat Design Specification (CDS) Rev. 13," tech. rep., California Polytechnic State University, 2015.

[2] The CubeSat Program, Cal Poly SLO, "6U CubeSat Design Specification Revision 1.0," tech. rep., California Polytechnic State University, 2016.

[3] F. Stesina and S. Corpino, "Investigation of a CubeSat in Orbit Anomaly through Verification on Ground," *MDPI Aerospace*, vol. 7, no. 4, p. 38, 2020.

[4] ISO 17770:2017, "Space systems – Cube satellites (CubeSats)," Standard, International Organization for Standardization, Geneva, Switzerland, 2017.

[5] A. Stemper, "Simplified Thermal Simulator of an electrical power supply system of a Cubesat (EPSThermalSimulator) GitHub Repository." https://github.com/andre-stemper/EPSThermalSimulator.git, 2022.

[6] J. De Claville Christiansen, "CubeSat Space Protocol (CSP): Network-Layer delivery protocol for CubeSats and embedded systems," Tech. Rep. GS-CSP-1.1, GOMSpace ApS, 2011.

[7] T. Yairi, N. Takeishi, T. Oda, Y. Nakajima, N. Nishimura, and N. Takata, "A Data-Driven Health Monitoring Method for Satellite Housekeeping Data Based on Probabilistic Clustering and Dimensionality Reduction," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 53, no. 3, pp. 1384–1401, 2017.

[8] J.-A. Martínez-Heras, A. Donati, M. G. F. Kirsch, and F. Schmidt, "New Telemetry Monitoring Paradigm with Novelty Detection," in *Proceedings of the SpaceOps 2012 Conference*, American Institute of Aeronautics and Astronautics, Inc., 2013.

[9] European Cooperation for Space Standardization, "ECSS-E-ST-70-41C: Space engineering – Telemetry and telecommand packet utilization," 2016.

[10] J.-G. Meß, F. Dannemann, and F. Greif, "Techniques of Artificial Intelligence for Space Applications – A Survey," in *European Workshop on On-Board Data Processing (OBDP2019)*, European Space Agency, 2019.

[11] J. Martínez-Heras and A. Donati, "Enhanced telemetry monitoring with novelty detection," *AI Magazine*, vol. 35, no. 4, pp. 37–46, 2014.

[12] M. Tipaldi, L. Feruglio, P. Denis, and G. D'Angelo, "On applying AI-driven flight data analysis for operational spacecraft model-based diagnostics," *Annual Reviews in Control*, vol. 49, pp. 197–211, 2020.

[13] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. Gao, "Deep learning and its applications to machine health monitoring," *Mechanical Systems and Signal Processing*, vol. 115, 2019.

[14] C. C. Aggarwal, *Outlier Analysis*. Cham, Switzerland: Springer, 2nd ed., 2017.

[15] A. Stemper, "The AtMonSat Project GitHub Repository." https://github.com/andre-stemper/ATMonSAT.git, 2022.

[16] STMicroelectronics N.V., "STM32H742xI/G STM32H743xI/G." https://www.st.com/resource/en/datasheet/stm32h743vi.pdf, 2022. Last accessed: [25 October 2022].

[17] G. M. Weiss, "Foundations of imbalanced learning," in *Imbalanced Learning: Foundations, Algorithms, and Applications* (Y. Ma and H. He, eds.), ch. 2, pp. 13–41, Hoboken, New Jersey: John Wiley & Sons, Ltd, 1st ed., 2013.

[18] P. Branco, L. Torgo, and R. P. Ribeiro, "A Survey of Predictive Modeling on Imbalanced Domains," *ACM Computing Surveys*, vol. 49, no. 2, pp. Article No. 31:1–50, 2017.