Design and validation of scalable PHM solutions for aerospace on-board systems

Fabio Federici[1], Cecilia Tonelli[1], Mathieu Le Cam[2], Marcello Torchio[2], and David Larsen[3]

Collins Aerospace, [1] Rome, Italy, [2] Cork, Ireland, [3] Vergennes, VT, United States of America

{name.surname}@collins.com

ABSTRACT

In recent years, Prognostic & Health Management (PHM) has become a topic of strong interest in the aerospace domain. Health assessment and remaining useful life estimation for on-board systems provide several advantages, mainly related to the increased analysis capabilities and the reduction of maintenance interventions (and, consequently, of operating costs). For this reason, it is of interest for the aerospace industry to identify and define efficient strategies both for the introduction of native PHM capabilities in new generation on-board systems and for the retrofit of existing ones. This paper proposes a strategy for the scalable deployment of PHM techniques for on-board systems, with particular focus on edge computing capabilities. Different reference scenarios (ranging from cloud-based processing to local-only processing) are presented, and an edge-focused PHM architecture is discussed in detail, with the relative challenges addressed. The design and validation of proposed edge-based solution is described, with specific reference to its support for an existing data analytics framework. The solution is then assessed against a reference aerospace use case involving a representative aircraft braking system, focusing on computational aspects to highlight the compatibility of the proposed deployment strategy with efficient on-board computations.

1. INTRODUCTION

Prognostics and condition-based maintenance (CBM) have attracted significant interest of the aerospace sector in the recent years. The goal of prognosis is to track degrading aspects of the overall design to predict deviation with respect to a reference baseline (e.g., healthy condition). Generally, we define prognostic systems as ones that compute remaining useful life (RUL), performance life remaining (PLR) or state of health (SOH) with sufficient fidelity and sufficient advance notice to allow a maintenance action well before an operational failure (SAE JA6268). Prognostics may provide significant benefits when applied to complex aerospace systems, potentially reducing maintenance-related downtime and costs, and improving the overall efficiency of the systems.

Traditional approaches to prognostics relied on physics-based methods (Cadini, Zio, & Avram, 2009) often involving fault propagation and reliability models for the component or system under consideration. Such methods require a thorough understanding of the system, and they are usually specific to a component and not generalizable for a broad variety of applications. More recently, data-driven approaches emerged as a powerful alternative. These approaches perform RUL prediction from the operational run-to-failure raw time series data, collected from sensors mounted on the components or systems under consideration. There are two types of data-driven approaches in the literature, *direct* and *indirect*.

Direct approaches rely on training a neural network to learn the RUL directly from the run-to-failure time series data (Zhang, Wang, Li, Cui, Liu, Yang, & Hu, 2018).

Indirect approaches rely instead on the so-called health monitoring, i.e., mapping the time series data into a one-dimensional Health Index (HI), which decreases monotonically and proportionally to the time series degradation (Mosallam, Medjaher, & Zerhouni, 2015). Deep learning methods are used frequently (Reddy, Venugopalan, & Giering, 2016) for health monitoring purposes. Once the health index is computed, RUL can be estimated, for instance, as a weighted average of RULs of matching HI curves (Wang, 2010) of all the time series in the training dataset.

One of the more relevant problems related to the computation of RUL for aerospace systems is the reduced availability of operational data from the systems under consideration. An efficient prognostics framework should be able to acquire data from sensors during relevant periods, collecting extended time series and provide them as input to described algorithms, which should be in turn deployed over proper computation platforms. Those capabilities are generally framed in the wider Integrated Vehicle Health Management

(IVHM) capability, which could be intended as the supporting platform enabling CBM of complex aircrafts.

Airborne systems include a variety of hardware and software components, complicating the definition of standardized mechanisms and infrastructures supporting CBM. Over the years, several standards (e.g., ISO 13374 or SAE JA6268) and architectural specifications (e.g., Open System Architecture OSA-CBM) tried to provide guidance for the implementation of generic, interoperable architectures (Chang, Gao, & Wang, 2018) (Goebel & Rajamani, 2021).

Several authors presented implementations compliant, or inspired to cited reference standards, providing mappings over specific supporting technologies. Tambe (2013) presented a distributed architecture for avionics sensor health assessment compliant with OSA-CBM and using the Data Distribution Service (DDS), a standard for real-time distributed systems supporting the publish/subscribe paradigm. Ezhilarasu and Jennions (2021) developed an architecture of a Framework for Aerospace Vehicle Reasoning (FAVER), a system-agnostic framework inspired by OSA-CBM and developed to isolate propagating faults by incorporating Digital Twins (DTs) and reasoning techniques. In other cases, the reference standards have not been considered, opting for alternative architectural proposals. Wang, Pan, Xiong, Fang, and Wang (2017) presented a software architecture based on Service-Oriented Architecture (SOA) and dual bus technology to share the information from on-board systems. Chen, Hu, & Hou (2021) proposed a general, time-variant architecture model usable to simulate PHM systems. Li, Verhagen, & Curran (2020) discussed a generic architecture along with a systematic methodology to support the design of PHM systems.

Only a limited number of works considered the integration of high-performance computing unit on-board the aircraft. For example, Chen, Liu, & Zhou (2020) presented a VPX-based computing platform for aircrafts with an artificial intelligence module built-in.

From a design perspective, several choices must be addressed to define a system supporting outlined functionalities (Li et al., 2020).

Differently from the literature presented above, this paper proposes a strategy for the scalable deployment of PHM techniques for on-board systems, with particular focus on edge computing capabilities. Different reference scenarios (ranging from cloud-based processing to local-only processing) are presented, and an edge-focused architecture, along with the challenges related to its implementation, is discussed. The selected solution is then detailed and evaluated with specific reference to the possibility of supporting an existing, mature data analytics framework. This integrated solution is then assessed against a reference aerospace use case involving a representative aircraft braking system, detailing its initial validation, and analyzing the feasibility of proposed system for a real-world deployment.

2. PHM Support for on-board systems

A CBM support system usually includes multiple functions. As a reference, the OSA-CBM functional model (MIMOSA, 2022) is represented in Figure 1.

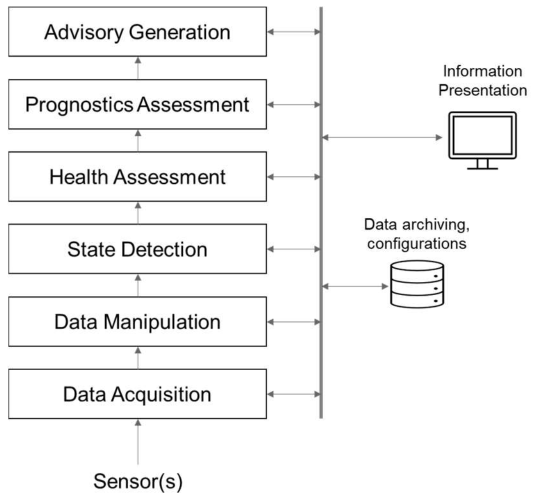


Figure 1 - OSA-CBM reference functions

The basic functionalities required by a CBM support system are two: (*i*) Data Acquisition (DA), and (*ii*) Data Manipulation (DM). The DA functionality provides access to the raw sensors' data collected from a Target System (TS) and may additionally include simple sensing calibration capabilities. The DM functionality, instead, implements mainly signal processing functions, together with sensor fusion and feature extraction algorithms. The State Detection (SD) function is used to estimate the current condition of the TS. It usually includes various functions ranging from Built-in Test (BIT) to more advanced components oriented to fault-detection. The Health Assessment (HA) function provides a SOH estimate of the TS, usually considering its operating conditions as well as the results of previous assessments. The Prognostics Assessment (PA) function estimates the RUL of the TS, either directly or indirectly (i.e., relying on the SOH estimate provided by the HA function).

This work mainly focuses on DA and DM, along with the integration of the HA and prognostic functions. The Advisory Generation (AG) function, that usually provides reports and recommendations for maintenance actions based on the results of the HA and PA, is not considered in this work.

The described functions shall be deployed onto suitable computing platforms and integrated with the existing aircraft systems, such as the TS (to ensure DA from sensors), but also other systems (or sub systems) that enable data transmission, aggregation, and processing. Ideally, the integration of the highlighted functions should have minimal impact on the overall aircraft design, especially in terms of Size, Weight, And Power (SWAP) consumption. The overall CBM support system should also be designed to support modularity and versatility in terms of integration with other systems (Finda & Hédl, 2014). CBM functions are frequently delivered through a distributed architecture, that can potentially involve both on-board and ground-based operations. Figure 2 represents a high-level view of a possible deployment for CBM related functions.

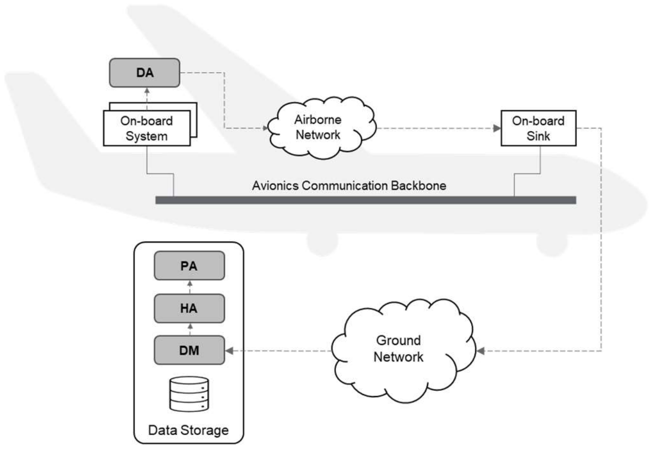


Figure 2 – Schematization of a ground-based CBM supporting architecture

In particular, the setup in Figure 2 represents a generic architecture that enables ground-based CBM operations through a mix of on-board and ground-based systems. The on-board system in only responsible for providing DA functions that collect measurement from the TSs. The TSs' data are acquired and collected during flight, continuously or only during relevant flight phases. The collected data are transmitted towards an on-board aggregation unit (or sink) through an airborne network. It should be noted that this network is generally a dedicated communication infrastructure, distinct from the traditional avionics' communication backbones. Finally, the collected data are transferred to a ground-based infrastructure through a ground network. Data may be streamed continuously (e.g., satellite connections), or stored for the flight duration and transferred once the aircraft has landed. The ground-based infrastructure is responsible to carry out all the remaining functions of the OSA-CBM reference model in an offline fashion (i.e., without processing the data as they are received). The data may be permanently stored and analysed according to specific HA and prognostics techniques. Modern deployments may leverage private cloud infrastructure to support the ground-based operations (Terrissa, Meraghni, Bouzidi, & Zerhouni, 2016).

This approach presents several advantages in terms of implementation: it requires a limited number of functions to be deployed onto on-board systems, and the ground-based offline computation is not constrained by limited resources in terms of computational power and storage. On the other hand, this architecture requires the collection of extended time-series data, which must be transmitted and temporarily stored on-board (e.g., by the aggregation unit), and then transmitted to the ground infrastructure. Moreover, the availability of ground-based facilities to support the required computations is generally not guaranteed for every type of aircraft.

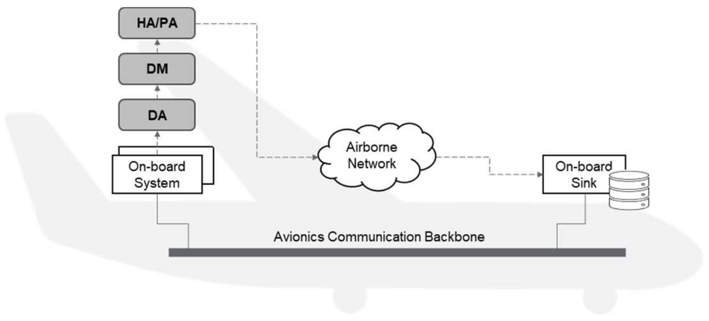


Figure 3 - Schematization of an edge-oriented CBM supporting architecture

A possible way to overcome the aforementioned limitations could be through the adoption of edge-oriented architectures. Figure 3 provides a representation of this scenario. This setup supports the on-board execution of target CBM strategies. Indeed, the on-board system would not only support DA functionalities, but would also provide extended computing capabilities to host other, more advanced functions, as DM, HA, and PA. Data are acquired and processed during flight, continuously or only during relevant flight phases, in order to produce relevant results related to RUL or HA. Only the obtained results are transmitted toward the on-board aggregation unit. This setup allows to overcome several disadvantages of a ground-oriented infrastructure. The majority of CBM related functions could potentially be hosted onto the on-board system, without the need for ground-based support. The acquired data can be processed close to the source, without requiring the storage or transmission of large-size data sets, neither to an on-board sink nor to a ground-based station. As a drawback, this approach requires extended capabilities (e.g., increased computational power) to be included among on-board systems.

Intermediate scenarios could be also considered. As an example, the edge processing capability could be limited to DA and DM, with DM delivering only feature extraction functionalities. This configuration has the advantage of requiring reduced processing resources, while still allowing to support significant data reduction, as time series are processed to extract only relevant indicators.

The introduction of CBM support at the edge, from basic DA functionalities to full-fledged support for data analytics, poses multiple challenges in terms of overall system design. The following section analyses the most relevant challenges and introduces a reference architecture suitable to support the reviewed scenarios.

2.1. PHM for on-board systems: proposed approach and related challenges

All the architectures reviewed in the previous section, require the baseline capability of extracting data from the TS. This may represent a significant challenge in aerospace systems, both in the case of legacy equipment and in the case of newly designed platforms. Legacy equipment is usually not designed to support expansion for adding the additional functionalities required to support data collection, nor it provides the communication interfaces required to share collected data with other on-board systems. New equipment can be designed considering native support for CBM related functions, like DA, storage, and communication, with limited impact on overall size, weight, and power. Adding CBM-related function to safety-critical equipment (for example, including health-monitoring related items in an Engine Control Unit) can also be challenging from the point of view of safety-related certification, as these functions are typically non-critical (e.g., classified at DAL-E according to RTCA DO-178C). A system including both safety-relevant functionalities and CBM related functionalities on the same platform would be considered a mixed-criticality system and should provide guarantees of strong isolation between the different DAL levels to be certifiable.

An intermediate solution is the addition of external dedicated unit, specifically designed to support CBM related functions. This option would be potentially able to fit both the use case of legacy equipment, and that of newly developed systems, with limited impact on certification issues. The CBM functionalities would be in fact allocated to a separate platform, with proper interlock mechanisms in place to guarantee the absence of undesired interference at the communication interfaces. The platform could not only implement DA (passively receiving data transmitted by the TS over a dedicated interface, if this is supported by the design, or actively interrogating the TS to extract relevant data using available access mechanisms, such as dedicated test interfaces), but also support local data processing (from DM for feature extraction, to local HA). The obvious drawback of an external unit is the impact in terms of additional SWAP, which could be limited at the expense of reduced computational resources. Interestingly, an external dedicated unit can deliver more flexibility from the point of view of on-board communication, providing support for multiple interfaces, and allowing to run dedicated communication stacks and middleware.

This work considers the use of a dedicated unit to support edge-based CBM-related functions. Figure 4 provides an overview of the reference architecture used in the context of this work, with specific focus on the interaction among the different systems involved.

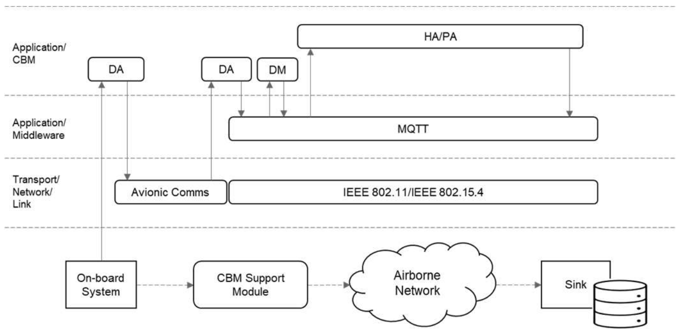


Figure 4 - High-level view of proposed system architecture

The proposed schematization identifies several reference layers, mainly related to the actual technologies supporting the distributed deployment. The lowest level includes relevant physical systems installed on-board, and related communication means. On top of this level, networking technologies and protocol stacks are considered. The application level is then split into two different layers: the middleware layer, including all components responsible for communication abstraction and the CBM layer, including the different CBM functions detailed in section 2.

As anticipated, a dedicated unit, the CBM Support Module (CSM), is connected to the TS, and it can collect relevant data from it.

The CSM communicates with the TS using a dedicated avionic interface. Data exchange is usually managed by means of a standard avionic protocol (e.g., ARINC 429), or by means of a custom protocol in case of proprietary interfaces.

The CSM also includes an external network interface, used to access the dedicated CBM support network. Different approaches can support the communication, ranging from traditional wired networks to wireless-based solutions. To reduce the overall impact of the CSM, the use of wireless technologies (e.g., IEEE 802.11, IEEE 802.15.4) may be considered.

The external network is used to communicate to a network sink module, intended as an on-board data collection unit.

The Message Queuing Telemetry Transport (MQTT) middleware provides an abstraction over the low-level communication method and allows to implement a publish-subscribe message transport between multiple on-board units. The MQTT model requires a message broker, i.e., a server able to receive messages from a sender and route it to the intended receiver. Multiple clients can connect to the broker over a network to exchange messages. In the presented approach, the CSM can run one or more MQTT clients, each one implementing a specific function. In a basic setup, only one message broker is included, running on the (e.g.) network sink module. In more advanced configurations, the CSM may host multiple functions, communicating by means of the MQTT middleware. In that case, a local broker runs on the

module, leveraging the possibility offering by MQTT of bridging multiple brokers in a network.
In the MQTT model, a hierarchy of topics support the exchange of data along the network. Clients publish new data over a certain topic and may subscribe to other topics to receive data. The broker oversees managing data distribution among the client that have subscribed a certain topic. The proposed model does not rely on statically configured clients, assuming instead that each client requires a configuration at startup. The configuration specifies which data shall be acquired by the client, the origin of this data and the means supporting data acquisition. The configuration also specifies the destination of data produced by the client (e.g., the topic over which the data shall be published). The use of configurations allows to implement re-usable modules, with a fixed functionality potentially configurable to accommodate the needs of different application scenarios.
Another advantage of the abstraction provided by the MQTT middleware is the possibility of relocating functions on different nodes of the network. For example, in the approach under analysis the DM function and the prognostic function are intended to receive and send data using the MQTT publish-subscribe mechanism. This means that those functionalities can be easily moved between the CSM and the sink module. The mapping is usually done according to specific non-functional requirements, as discussed in the beginning of this section.
Functions with specific dependencies on the underlying physical equipment have less flexibility. For example, on the CSM the data acquisition function only relies on MQTT mechanisms to share acquired data, however it requires physical connection to the TS.

3. A REFERENCE USE CASE: THE TRAJECNETS CBM FRAMEWORK

The general strategies described in section 2 can be used to map an actual CBM framework to a proposed high-level architecture and drive the detailed design of some of its reference modules. This section provides an overview of the reference framework adopted in the context of this work from the point of view of data analytics. The different CBM - related functions will then be identified and mapped onto the reference architecture in Figure 4, also detailing their specific implementation in hardware and software.

3.1. Overview of target Analytics

The proposed CBM framework is built on top of the TrajecNets approach (Shahid & Ghosh, 2019), and leverages a Recurrent Neural Network (RNN) based autoencoder for embedding the run-to-failure time series sensor data in a 2D feature space. The embedding is in the form of a trajectory representing the temporal evolution of data from healthy to failure states. This trajectory can be used for health monitoring, which can in turn be used for RUL estimation.

Figure 5 provides a high-level schematization of the data analytics pipeline for the proposed RUL estimation.

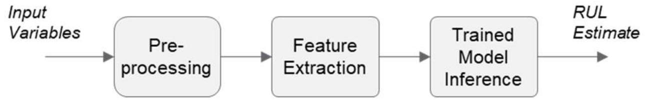


Figure 5 - Schematization of a reference data analytics pipeline

Input variables are acquired and may be pre-processed in different ways depending on the specific flight phase. The selection of the relevant flight phases and the related pre-processing strategies usually depend on domain knowledge and preliminary data exploration.
For each pre-processed input variable, up to 13 features may be computed: mean, standard deviation, variance, minimum, maximum, median, sum, mean absolute deviation, skewness, kurtosis, number of null values, 10th percentile, and 90th percentile. These features are extracted to statistically describe the variables' range of variation and helps in providing an efficient representation of the TS by reducing the amount of raw data.
The data-driven approach developed for the RUL estimation of a TS is based on an Auto-Encoder (AE) architecture such as the one presented in Figure 6. Starting from high-dimensional inputs set (i.e., up to 13 dimensions), the AE is capable of providing a representation of the TS SOH/RUL in a smaller-dimensional latent space (2D for the TrajecNets approach). Indeed, the RUL is estimated based on the smaller-dimensional representation that stems from the computation of current and past data. Due to its complexity, this model may involve more than 30,000 weights to be trained. During training, the resulting TrajecNets' RNN learns the neurons' weights with the objective of minimizing the reconstruction error of the AE and the exponentially weighted mean absolute error of the estimated RUL. For more details about the training procedure, and inner details of TrajecNets, please refer to (Shahid & Ghosh, 2019). The described CBM framework has been implemented using the widely adopted TensorFlow library, and it has been previously validated using publicly available datasets (Shahid et al., 2019).

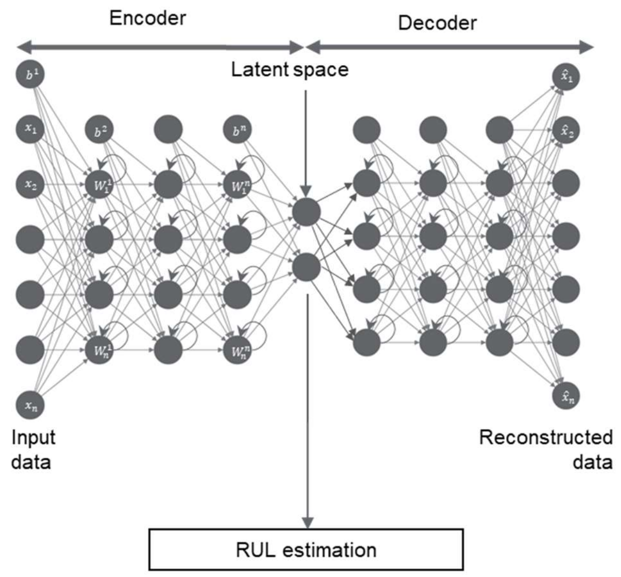


Figure 6 – TrajecNets architecture for data-driven RUL estimation (Shahid & Ghosh, 2019).

3.2. Mapping to the reference architecture

The implementation of the CBM framework described in section 3.1 requires support for the following high-level functions:

- DA: the input data to the TrajecNets RNN must be collected from the TS.
- DM: the proposed algorithm assumes that the input data are collected during specific flight phases, with different pre-processing requirements. A set of relevant features must be extracted from the acquired time-series.
- HA and PA: as described, the data-driven model developed and trained using the collected data can support in providing TS's HA and RUL estimation.

The SD function is not considered in the context of this work, but it can be potentially integrated in the CSM.
The DA function shall be allocated both on the TS and on the CSM, according to the strategies already detailed in section 2.1. Considering the low computational complexity, it is possible to deploy the DM functions directly on the CSM. One possible limitation related to the feature extraction could be related to the need of storing the acquired time series on CSM's memory, which can be in general limited. Implementing online calculation of target features (e.g., statistical indices) may help in mitigating this risk.
As discussed, the integration of the prognostics function on the on-board CSM represents an interesting, but challenging, aspect that requires a reduction in size for the TrajecNets-based RUL model in order to fit the constrained resources of the on-board system.
There are different approaches to reduce the size of a model, such as quantization, pruning and clustering. Quantization can reduce the size of a model by mapping continuous variables onto discrete values, potentially at the expense of some accuracy (such as truncating or rounding). Pruning and clustering can reduce the size of a model by making it more compressible. Pruning works by removing parameters within a model that have only a minor impact on its predictions. Pruned models are the same size on disk, and have the same runtime latency, but can be compressed more efficiently. Clustering works by grouping the weights of each layer in a model into a predefined number of clusters, then sharing the centroid values for the weights belonging to each individual cluster. This reduces the number of unique weight values in a model, thus reducing its complexity. Clustered models can also be compressed more effectively, providing some deployment benefits.
As anticipated, the TrajecNets-based RUL model developed in this work relied on the usage of the TensorFlow framework; for this reason, the TensorFlow Lite toolset (Google, 2022) appeared to be a natural option to support the RUL's model size reduction.

3.3. Edge platform selection

Different classes of devices can be considered for the implementation of the on-board CSM, ranging from microcontroller devices to more capable single board computers. Considering the need to host middleware-related functionalities, and potentially supporting advanced frameworks like TensorFlow, a mid-range single board computer has been identified.
Specifically, the target platform considered in this work is a commercial single-board computer based on a Freescale i.MX6 System on Chip (SoC), including an Arm® Cortex®-A9 single core, operating at a clock frequency of 800 MHz. The SoC also include a Graphics Processing Unit (GPU), providing hardware acceleration for 3D graphics but not suitable for more general-purpose computation. The platform manufacturer provides support for several GNU/Linux OS distributions. The TensorFlow framework has been fine-tuned to run on top of the standard Ubuntu Linux distribution. The platform has a base storage capability of 4GB, with support for external expansion. It supports a variety of communication interfaces, including wired and wireless networking.

3.3.1. Software architecture overview

The proposed hardware platform provides a variety of I/O interfaces, suitable for interfacing with the TS and for communicating over an airborne network supporting the overall PHM functions.
The software architecture of the proposed system is represented in Figure 7.

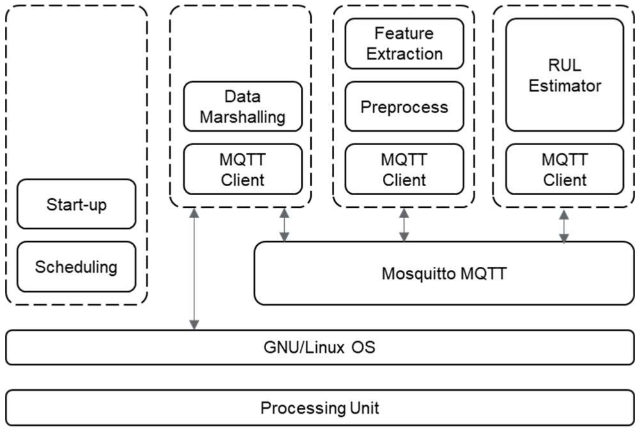


Figure 7 – Proposed CSM software architecture

The DA component supports communication over external interfaces, data marshalling, and interaction over the MQTT middleware, which, as anticipated, provides an abstraction for network communications.

The feature extraction component supports the online calculation of target features set and handles the interactions with the MQTT middleware. The component receives as input the information on the current flight phase and relies on it to apply the correct pre-processing approach to the input data.

Finally, the CSM supports PA and HA, based on the algorithm described in section 3.1. This component relies on the TensorFlow Lite runtime and integrates an MQTT client. Besides the described function, the platform includes a standard generic component supporting the initial start-up of the platform (initialization of networking services and of MQTT middleware support) and scheduling of other components.

4. VALIDATION SETUP: SUPPORTING CBM FOR AN AIRCRAFT BRAKING SYSTEM

To validate the on-board PHM architecture introduced in the previous sections, in the following part of the paper, a representative braking system is presented as TS.

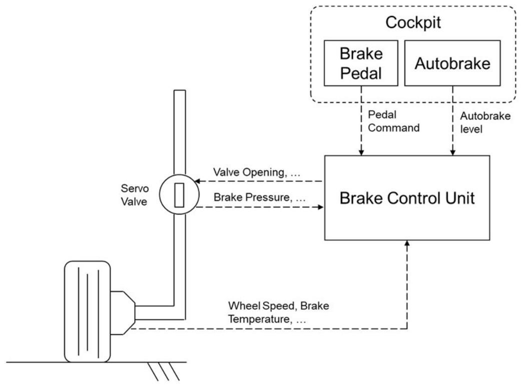


Figure 8 - High-level view of an Aircraft Braking System

Figure 8 shows a simplified control architecture for an aircraft electro-hydraulic/electro-hydrostatic braking system (SAE International, 2006).

Common commercial aircraft brakes are made of Carbon/Carbon composites. The brake is composed of a stack of alternated stator and rotor discs. A servo valve regulates the hydraulic fluid directed to the pistons in the brake assembly. An increase of the hydraulic pressure causes a compression in the stack of stators and rotors, resulting in an increase of friction that slows down the wheel rotation. During the braking action, most of the aircraft's kinetic energy is transformed into heat and absorbed by the brakes (Daidzic, 2017).

A Brake Control Unit (BCU) hosting the control system is responsible for the regulation of the servo valve. The valve opening is computed according to the reference signals received from the cockpit, namely the pilot (and/or co-pilot) brake pedal command and/or the level of required autobrake. The BCU can collect data from several sensors deployed along the system (wheel speed, brake temperature, brake pressure, etc.), and uses them as input for the different control algorithms hosted on the unit.

After each brake application, the brake's discs tend to wear (due to the loss of material induced by friction). Moreover, wear dynamics are impacted by the thermal behavior of the brake (Di Santo, 2005). When the level of wear reaches certain limits, the brake needs to be replaced with a new one. It is therefore possible to relate a brake's wear to its RUL.

The following subsections provide an overview of a possible application of the generic framework described in section 3 for brake RUL estimation, along with its actual deployment of an on-board setup that follows the architectural approach presented in section 2.1. Due to the confidentiality of the data presented, normalization procedures were carried out. Despite the normalization process, the results presented in the following section highlights the efficacy of the proposed approach in supporting the scalable deployment of complex PHM functions onto on-board components with limited resources.

4.1. Data generation, acquisition, and manipulation

To generate data and support the training phase of the TrajecNets-based RUL estimator for brakes, a MATLAB-based simulator was built to emulate the behaviour of both aircraft brakes' wear and thermal dynamics.

A representative set of variables collected from the simulator were used to train TrajecNets (example subset shown in Table 1).

The simulated data are sampled with a frequency of 1 Hz and pre-processed applying a moving average. The acquisition and pre-processing of input data leads to a total amount of 1820 data items per flight, for a total size of ~7 KBs.

Table 1 – Representative set of input variables

Variable	Description
Wear level	Indicator that provides a quantification of the current brake's wear level.
Brake temperature	Indicator that provides the current temperature of the brake.
Flight phase	Flag that indicates in which flight phase the system is working in.

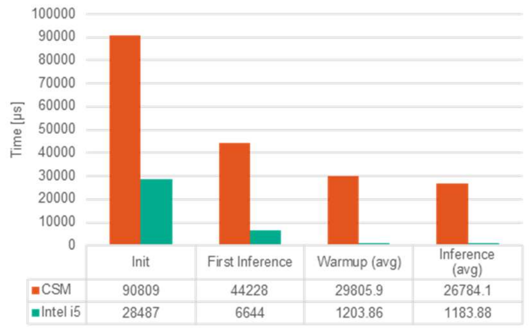
As discussed in section 3.1, a set of features was computed to support the brake RUL estimation. These features are computed for each input variable, per each flight phase of a given flight. It should be noted the features could be computed online, leading to a potential further reduction of memory footprint.
The outcome of the training step provided a TrajecNets-based model (original model) trained with a high-fidelity parameterization of its internal structures (~30,000 32bit float parameters). However, due to its complexity and dimensionality, the online execution of such model onto the edge computing device defined in section 3.3 resulted to be prohibitive.

4.2. Model reduction, integration & assessment

To overcome the edge deployment limitations, a model reduction and integration process was carried out. The original model was reduced to a quantized version through TensorFlow Lite, where a quantization of the ~30,000 parameters from float (32bits) to integer (16bits) has been adopted.
The overall reduction process required two steps:

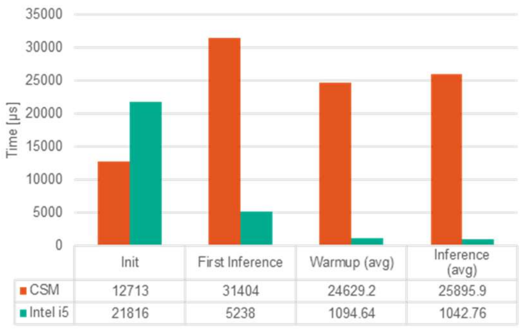
(i) the original model with a size of 340 KBs, was initially *reduced* to a TensorFlow Lite float model of 211 KBs in size, and
(ii) the float model was finally *reduced* into the quantized version which size was 164 KBs (about 77% of the float model size).

The impact of the model size reduction through quantization on the model accuracy has been evaluated through simulation. The accuracy of the model is defined as the Root Mean Squared Error (RMSE) between the RUL prediction and a target value, over a statistically meaningful number of unseen simulated flights.
From the comparison results, it emerged that both the quantized and float models provided RMSE indices ~5% higher with respect to the original model (considered our best-case scenario). The accuracy of the reduced models is slightly impacted but not in a significant manner for the application under analysis.
The performance of the reduced models has been characterized on the reference platform using the standard TensorFlow Lite Benchmark tool. The tools allow to measure relevant execution times (initialization time, inference time of warmup state, inference time of steady state) and memory usage (memory usage at initialization time, overall memory usage during execution). As a reference, the same characterization for the reduced models executed on a high-end processor (Intel Core i5-8365U with a clock frequency of 1.60 GHz) is included.
The inference timings of the reduced models are summarized in Figure 9 and Figure 10, respectively for the quantized version and the float version.



	Init	First Inference	Warmup (avg)	Inference (avg)
■ CSM	90809	44228	29805.9	26784.1
■ Intel i5	28487	6644	1203.86	1183.88

Figure 9 – Inference times, quantized model



	Init	First Inference	Warmup (avg)	Inference (avg)
■ CSM	12713	31404	24629.2	25895.9
■ Intel i5	21816	5238	1094.64	1042.76

Figure 10 – Inference times, float model

In both cases, estimated time figures resulted acceptable with respect to target performance requirements, especially considering the non-real time nature of the execution for this part of the application.
The peak memory footprint for the reduced models is detailed in Table 2 and Table 3, respectively for the quantized version and the float version.

Table 2 - Quantized model, peak memory footprint [MB]

	CSM	Intel i5
Init	2.74	3.24
Overall	3.52	3.99

Table 3 - Float model, peak memory footprint [MB]

	CSM	Intel i5
Init	2.93	4.16
Overall	3.64	4.78

Both in the case of the quantized model and the float model, the peak memory footprint proved to be compatible with the memory capabilities of the described processing platform.

5. Conclusion

This paper presented a strategy for the scalable deployment of PHM techniques for on-board systems, with particular focus on edge computing capabilities.

The relevant scenario for deployment of CBM support for on-board systems have been presented, and a system level architecture able to address significant use-cases has been introduced. The proposed approach allows flexibility both in the on-board deployment and in targeting different on-board systems.

In the proposed approach, the TS is extended with an external support module, able to host relevant CBM related functions.

The application of the proposed approach to an actual aerospace use case has been discussed, first introducing a generic CBM framework, previously validated in a traditional offline setup, and then mapping this framework over the proposed reference architecture.

The proposed implementation proved to be viable in a real use case, a representative aircraft braking system, used for the overall validation of the proposed approach. The described methodology is in any case generalizable and can be applied to different aircraft systems.

Based on proposed analysis, it was shown that the TrajecNets-based model trained off-line using simulation data (and based on the full TensorFlow library) can be converted and reduced in size (and complexity) for an embedded application using TensorFlow Lite toolset. A non-significant impact on the model accuracy for RUL estimation related to brake wear was evaluated for this use case.

Acknowledgement

This research has been funded by the European Union's CleanSky2 Research and Innovation program ICO-Brake under grant No. 945535. The authors would like to thank Jeffrey Schmidt, Rhonda Walthall, and Marc Georgin for their support during the execution of the activities described in this paper.

References

Brady, C. (2022). Brake Wear Pin. Retrieved from B737: http://www.b737.org.uk/brakepin.htm

Cadini, F., Zio, E., & Avram, D. (2009). Model-based Monte Carlo state estimation for condition-based component replacement. *Reliability Engineering & System Safety*, 94 (3), pp. 752–758.

Chang, S., Gao, L., & Wang, Y. (2018). A review of integrated vehicle health management and prognostics and health management standards. *International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC)*, (pp. 476-481)

Chen, R., Hu, Y., & Hou, Y. (2021). Architecture analysis for avionics prognostics and health management system. *CAA Symposium on Fault Detection, Supervision, and Safety for Technical Processes (SAFEPROCESS)*, (pp. 1-5)

Chen, Z., Liu, J., & Zhou, X. (2020). An Embedded AI Computing Platform for Aircrafts Based on VPX Bus. *4th CAA International Conference on Vehicular Control and Intelligence (CVCI)*, (pp. 602-606)

Daidzic, N. E. (2017). Modeling and Computation of the Maximum Braking Energy Speed for Transport Category Airplanes. *Journal of Aviation Technology and Engineering*, 2-25

Di Santo, G. (2005). Proper Operation of Carbon Brakes. *11th Performance and Operation Conference*

ISO. (2003). Condition monitoring and diagnostics of machines - Data processing, communication and presentation - Part 1: General guidelines, ISO 13374-1:2003

Ezhilarasu, C. M., & Jennions, I. K. (2021). Development and Implementation of a Framework for Aerospace Vehicle Reasoning (FAVER). *IEEE Access*, 9, 108028-108048.

Finda, J., & Hédl, R. (2014). On-board SHM System Architecture and Operational Concept for Small Commuter Aircraft. *PHM Society European Conference*, 2 (1)

Goebel, K., & Rajamani, R. (2021). Policy, regulations and standards in prognostics and health management. *International Journal of Prognostics and Health Management*, 12(1).

Google LLC. (2022). TensorFlow Lite: Deploy machine learning models on mobile and IoT devices. Retrieved from https://www.tensorflow.org/lite/

Han, D. Y. (2019). A distributed autonomic logistics system with parallel-computing diagnostic algorithm for aircrafts. *2019 IEEE AUTOTESTCON*, (pp. 1-8)

Li, R., Verhagen, W. J., & Curran, R. (2020). A systematic methodology for Prognostic and Health Management system architecture definition. *Reliability Engineering & System Safety*.

MIMOSA. (2022). Open System Architecture for Condition Based Monitoring (OSA-CBM). Retrieved from: https://www.mimosa.org/mimosa-osa-cbm/

Mosallam, A., Medjaher, K., & Zerhouni, N. (2015). Component based data-driven prognostics for complex systems: Methodology and applications. *First International Conference on Reliability Systems Engineering*, (pp. 1–7)

Reddy, K. K., Venugopalan, V., & Giering, M. J. (2016). Applying deep learning for prognostic health monitoring of aerospace and building systems. *1st ACM SIGKDD Workshop on ML for PHM*

SAE International. (2006). Braking System Dynamics, AIR1064D

SAE International. (2018). Design & Run-Time Information Exchange for Health-Ready Components, SAE JA6268

Shahid, N., & Ghosh, A. (2019). TrajecNets: Online failure evolution analysis in 2D space. *International Journal of Prognostics and Health Management*, 10(4)

Tambe, S. U. (2013). An extensible architecture for avionics sensor health assessment using data distribution service. *AIAA Infotech@Aerospace (I@A) Conference*, (p. 5139)

Terrissa, L. S., Meraghni, S., Bouzidi, Z., & Zerhouni, N. (2016). A new approach of PHM as a service in cloud computing. *4th IEEE International Colloquium on Information Science and Technology (CiSt)*, (pp. 610-614)

Wang, F., Pan, S., Xiong, Y., Fang, H., & Wang, D. (2017). Research on software architecture of prognostics and health management system for civil aircraft. *International Conference on Sensing, Diagnostics, Prognostics, and Control*

Wang, T. (2010). *Trajectory similarity-based prediction for remaining useful life estimation*. Doctoral dissertation, University of Cincinnati.

Zhang, A., Wang, H., Li, S., Cui, Y., Liu, Z., Yang, G., & & Hu, J. (2018). Transfer learning with deep recurrent neural networks for remaining useful life estimation. *Applied Sciences*, 8 (12), p. 2416

BIOGRAPHIES

Fabio Federici is a Principal Engineer at Collins Aerospace, a Raytheon Technologies company. At Collins, he works as part of the Applied Research and Technology organization, mainly focusing on dependable embedded HW/SW architectures. He originally joined the United Technologies Corporation (now part of Raytheon Technologies) in 2017, as part of the corporate research center. Before that, he worked as a Data Handling and Research Engineer at Thales Alenia Space Italy. Fabio holds a Ph.D. in Electrical and Information Engineering from the University of L'Aquila (Italy). He has been a post-doctoral researcher and a contract professor at the same university.

Cecilia Tonelli earned a M.S. in Mathematics, with a Computational Algebra thesis in 2011 and one year later she participated the 2012 EACA (Meetings on Computer Algebra and Applications). She worked as Junior Software Developer in automotive sector for 3 years, then she joined the United Technologies Corporation (now part of Raytheon Technologies) in 2015 as Senior Engineer, working for Collins, in the Applied Research and Technology organization. She was member of the Embedded Technologies group, now she is member of the Digital Thread & Twin Technologies.

Mathieu Le Cam is a senior researcher with Collins Aerospace Ireland. He received his PhD in Building Engineering from Concordia University, Canada in 2016 and his M. Eng. in Mechanical Engineering from Ecole Polytechnique de Montreal in 2012. His academic research focused on data-driven and physics-based modeling for the forecast of the electric demand of HVAC systems, and their application in building energy management. Since he joined Collins Aerospace Ireland, formerly United Technologies Research Centre, in 2017, he has been involved in different projects leveraging his skills in machine learning and mechanical engineering for the Prognostics and Health Managements of different aircraft systems.

Marcello Torchio obtained the master's degree in computer engineering from the University of Pavia in 2012. In 2016 he received the PhD in Automatic Controls at the University of Pavia with the Thesis entitled "Model Predictive Control Strategies for Advanced Battery Management Systems". In 2016 he joined United Technologies Research Centre Ireland, ltd as a Senior Research Scientist in the Controls and Decision Support group providing technical and management contributions in the execution of R&D projects. From 2021 he covers the role of Sr. Principal Engineer at the Advanced Research & Technology center of Collins Aerospace, within the Autonomous Systems team. He is author of several international journal and conference papers on topics of model predictive control, optimization, and electrochemical modelling, as well as author of different patents.

David Larsen is a Technical Fellow, PHM Systems at Collins Aerospace. Dave is a leader in the advancement of Prognostics and Health Management (PHM) Systems for Collins, authoring Collins' common PHM standard work methods, driving PHM collaboration, and leading the design and development of PHM-enabling systems and components, specializing in health data provisioning. Dave teaches the Introduction to PHM course and the PHM Fundamentals and Use Cases course through the Collins Aerospace Technical University. Dave is also the Dean of the CATU Systems Engineering College. Dave holds a BSEE from Rensselaer Polytechnic Institute. Dave is a Fellow of the PHM Society.