Ref. Ares(2026)8240339 - 28/08/2026

Funded by the European Union

UK Research and Innovation



Report of Communication Protocol Framework

Document ID: AVATAR_D3.1

Date : 21/08/2026

Dissemination level: Public

Funding

This project has received funding from the European Climate, Infrastructure and Environment Executive Agency (CINEA) under Grant agreement no 101096073, and from United Kingdom Research and Innovation (UKRI) under Grant agreement no 10065739.

Disclaimer

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or CINEA. Neither the European Union nor the granting authority can be held responsible for them.

Abstract: This deliverable summarizes the development of a lightweight, energy-efficient communication framework for the AVATAR project. It includes event-triggered data acquisition, wireless intra-aircraft communication via Bluetooth Low Energy (BLE), and RF-based transmission for drones to minimize onboard weight. The system enables real-time sensor data transfer to the edge device, for later transmission to cloud platform data analysis. Each technology choice is tailored to the operational needs and constraints of aircraft and drones, ensuring reliable and scalable integration.

Table 1 : Table of versions

Revision History Table		
Version n°	**Issue Date**	**Reason for change**
A01	21/07/2024	First draft with Table of Contents
A02	07/03/2025	Validated Table of Contents, Consolidated document
A03	04/04/2025	Document reviewed and ready to be published

Table 2 : Table of parties involved in the compilation of the deliverable.

Nb	Member name	Short name	Check if involved
1	LOUGHBOROUGH UNIVERSITY	LBORO	**X**
2	IMPERIAL COLLEGE LONDON	ICL	**X**
3	DANISH TEKNOLOGISK INSTITUT	DTI	
4	UNIVERSITE DU LUXEMBOURG	UL	**X**
5	NTT LUXEMBOURG PSF SA	NTT	**X**
6	EVEKTOR SRO	EVE	**X**
7	NORDIC WING APS	NW	**X**

Content

1 Introduction 5
1.1 Purpose of the Document 5
1.2 Structure of the Document 5
1.3 Relation to other Work Packages and Deliverables 5
2 Overview of D3.1 Objectives 7
2.1 Event-Triggered Multi-Mode Data Acquisition (WP3.1) 7
2.2 Wireless Avionic Intra-communications (WP3.2) 7
2.3 Aircraft to Ground System Communication (WP3.4) 7
3 EVENT-TRIGGERED MULTI-MODE DATA ACQUISITION 8
3.1 Objectives 8
3.2 Implemented Mechanisms 8
3.3 Developed Algorithms 9
3.4 EXPERIMENTAL VALIDATION 10
4 WIRELESS AVIONIC INTRA-COMMUNICATIONS 12
4.1 Objectives 12
4.2 Network Architecture 13
4.2.1 Communication platform 14
4.3 Implemented Solutions 20
4.3.1 Gateway Implementation 20
4.3.2 Sensor Impelementaion 21
4.3.3 Limitations 23
4.4 Testing and Validation 26
4.4.1 BLE performance testing methodology 26
5 AIRCRAFT TO GROUND SYSTEM COMMUNICATION 29
5.1 Objectives 29
5.2 Data COLLECTION AND TRANSMISSION METHODS 29
5.3 Technology Comparison 32
5.3.1 WIRELESS AND WIRED COMMUNICATION TECHNOLOGIES 32
5.4 TESTING AND VALIDATION 32
5.4.1 TRL TESTING STAGES 32
5.4.2 END-TO-END TESTING 32
5.5 CLOUD INTEGRATION AND DATA PROCESSING 32
5.6 IMPLEMENTATION ROADMAP 34
6 Conclusion 35
7 References 35

Figures

Figure 1 : Flowchart of event-triggered DAQ 10
Figure 2 : Experimental results for event-triggered DAQ mechanism 11
Figure 3 : AVATAR Wireless Data Transmission Overall Structure 13
Figure 4 : RPI 5 17
Figure 5 : STM32L475VGT6 board 18
Figure 6 : Counter Diff Chart for 10ms Interval Data Transmission 27
Figure 7 : Counter Diff Chart for 10ms Interval Data Transmission 28
Figure 8 : Shows high level data transmission from the Aircraft sensors to the cloud visual representation in a Digital Twin and Data Analysis. 29
Figure 9 : Technical High Level Data Architecture: Astero aircraft 30
Figure 10 : Technical High Level Data Architecture: Astero aircraft 30
Figure 11 : Grafana board representing a Digital Twin of Evektor Cobra. 33
Figure 12 : Grafana board representing a Digital Twin of Nordic Wing Astero drone. 33

Tables

Table 1 : Table of versions 2
Table 2 : Table of parties involved in the compilation of the deliverable. 2
Table 3 : BLE Types of Operations Comparison 16
Table 4 : Research Transmission Methods 31
Table 5 : Communication Technologies Comparison 32

1 INTRODUCTION

1.1 PURPOSE OF THE DOCUMENT

The purpose of this document is to provide a comprehensive report on the communication protocol framework developed under Work Package 3 (WP3) of the AVATAR project. In particular, the work described in this deliverable, D3.1, account for the work performed during the sub-WPs:

- WP 3.1 Event-Triggered Multi-Mode Data Acquisition
- WP 3.2 Wireless Avionic Intra-communications
- WP 3.4 Aircraft to ground system communication

This deliverable covers the design, validation, and implementation of data communication, storage, and processing infrastructure necessary for the seamless operation of the digital twin platform. It aims to ensure efficient data acquisition, transmission, and processing from the smart skin to the aircraft onboard gateway and from the aircraft to ground systems. The document also outlines the methodologies adopted for event-triggered data acquisition, intra-communications within the aircraft, and communication between the aircraft and ground systems.

1.2 STRUCTURE OF THE DOCUMENT

This document is structured into several sections to systematically present the information:

1. Introduction: Outlines the purpose, structure, and relevance of the document to other work packages and deliverables.
2. Overview of D3.1 Objectives: Provides a summary of the objectives for WP3.1, WP3.2, and WP3.4.
3. Event-Triggered Multi-Mode Data Acquisition: This section delves into the specifics of WP3.1, covering its objectives, the event-triggered mechanisms implemented, the algorithms developed for data acquisition, and the optimization and evaluation processes undertaken to minimize power consumption while ensuring high performance.
4. Wireless Avionic Intra-communications: This section presents the detailed objectives, network architecture, implemented solutions, and the testing and validation procedures for WP3.2, focusing on creating a robust intra-aircraft communication system that supports the digital twin framework efficiently and effectively.
5. Aircraft to ground system communication: This section provides an in-depth look at WP3.4, outlining the objectives, the process of data link selection, technology comparison, and the implementation roadmap. It highlights the steps necessary to establish effective communication links between the aircraft and ground systems, ensuring seamless data transmission and integration.
6. Conclusions: Provides a summary of findings and future work.
7. References: Lists all references used in the document.
8. Appendices: Contains additional supporting information or data

1.3 RELATION TO OTHER WORK PACKAGES AND DELIVERABLES

This deliverable, D3.1, is closely related to several other work packages and deliverables within the AVATAR project:

- WP1: Provides the requirements and constraints for the data communication and processing infrastructure.

- WP2: Supplies the IoT sensing skin whose data needs to be communicated and processed.
- WP4: Utilizes the processed data for health monitoring and high-fidelity simulations.
- WP5: Relies on the communication framework for advanced analytics and predictive maintenance.
- WP6: Integrates and tests the communication protocols within the digital twin platform and aircraft demonstrators.

2 OVERVIEW OF D3.1 OBJECTIVES

2.1 EVENT-TRIGGERED MULTI-MODE DATA ACQUISITION (WP3.1)

The primary goal of WP3.1 is to implement an event-triggered data acquisition mechanism with multiple operation modes to optimize power consumption while maintaining high sensing performance. The objectives include:

- Implementing Event-Triggered Mechanisms: Develop mechanisms to trigger data acquisition based on specific events, reducing unnecessary data collection and power consumption.
- Developing Data Acquisition Algorithms: Create algorithms that balance the need for sufficient sensing data with the goal of minimizing power usage.
- Optimizing Data Acquisition: Fine-tune the mechanisms to achieve minimal average power consumption without compromising data quality.
- Evaluating Performance: Assess the performance of the data acquisition system on a component level to ensure it meets the desired specifications.

2.2 WIRELESS AVIONIC INTRA-COMMUNICATIONS (WP3.2)

WP3.2 aims to design and implement an optimal intra-aircraft communication system to support the digital twin framework. The objectives include:

- Selecting Communication Technology: Choose the most suitable technology for intra-aircraft communications based on the requirements and constraints identified in previous work packages.
- Developing Network Architecture: Create a robust network architecture that accommodates the dispersed nature of aircraft subsystems.
- Reviewing and Comparing Protocols: Analyze various wireless sensor network (WSN) protocols to determine the optimal one for implementation.
- Implementing and Validating Solutions: Test and validate the chosen communication protocol and system with prototype IoT skin and aircraft demonstrators.

2.3 AIRCRAFT TO GROUND SYSTEM COMMUNICATION (WP3.4)

WP3.4 focuses on establishing effective communication links between the aircraft and ground systems. The objectives include:

- Selecting Data Links: Identify and choose appropriate data links for transmitting sensing data from the aircraft to ground systems.
- Identifying Standards and Technologies: Explore and compare various standards and technologies for air-to-ground communication.
- Technology Comparison: Assess the advantages and disadvantages of each technology in relation to the requirements.
- Implementation Roadmap: Develop a roadmap for implementing the selected technologies and addressing any gaps in current systems.

3 EVENT-TRIGGERED MULTI-MODE DATA ACQUISITION

3.1 OBJECTIVES

The primary objectives of WP3.1 are to implement an event-triggered data acquisition mechanism with multiple operation modes to optimize power consumption while maintaining high sensing performance. This involves developing data acquisition algorithms, optimizing the mechanisms to minimize power consumption, and evaluating the performance of the system on a component level.

3.2 IMPLEMENTED MECHANISMS

The event-triggered mechanisms implemented in WP3.1 aim to reduce unnecessary data collection and power consumption. These mechanisms are designed to activate data acquisition only when specific events or conditions are detected, thereby conserving energy while ensuring that critical data is captured.

In this event-triggered multi-mode data acquisition system, various sampling frequencies are employed for acquiring sensor data. Referring to the design scheme outlined in Report D2.2, each sensor node incorporates strain gauges, accelerometers, and temperature sensors, which are deployed at critical locations such as the wings, tail, engine cover, and center of gravity. Under normal flight conditions, where the aircraft is not subjected to significant external excitations—such as collisions or high-frequency vibrations—the MCU's multiple ADC channels acquire low-frequency sampled data from strain sensors, accelerometers, and temperature sensors. However, when an external impact occurs, the sensor data from the impacted region and its surrounding area are likely to contain crucial information for assessing the aircraft's overall health and remaining lifespan. Consequently, to ensure the capture of a more comprehensive dataset, the sampling frequency is increased to the kilohertz (kHz) or even hundreds of kilohertz range. Low-frequency sampling minimizes MCU power consumption and simplifies data processing, whereas high-frequency sampling provides richer sensor data to enhance the real-time health assessment of the aircraft. To reduce unnecessary data processing and power consumption, and considering the memory constraints of the MCU, an event-triggered mechanism is implemented to switch between different sampling frequencies. By setting specific threshold values for the sensor data, the system determines whether an event has been triggered, subsequently adjusting the sensor sampling frequency. These threshold values are defined based on the comparison of sensor data under strong external excitations and minor disturbances. For instance, when the instantaneous acceleration of a specific axis exceeds a predefined threshold, an event is triggered, prompting the DAQ system to adjust the sampling frequency parameters and increase the sampling rate. After conducting high-frequency sampling for a predefined period, the system automatically reconfigures the sampling parameters, reverting the DAQ system to its standard low-frequency sampling mode. When such an event is triggered, the DAQ system follows this temporary frequency adjustment protocol, ensuring that data acquisition transitions seamlessly between different sampling modes.

3.3 DEVELOPED ALGORITHMS

In the event-triggered DAQ system, the sampling frequency of ADC channels is determined by multiple parameters within the MCU, including the ADC clock, ADC prescaler, the number of active ADC channels, sampling period, sampling time, and oversampling factor. Consequently, different ADC configurations are set for different sampling frequency modes. According to Reports D2.1 and D2.2, the DMA circular mode is employed to facilitate continuous sensor data acquisition. In this mode, the sampling frequency parameters are pre-defined in the adc.c file. When enabling ADC DMA circular mode, the buffer size must be pre-allocated to store the raw sensor data effectively. To accommodate different sampling frequencies, multiple sets of sampling frequency parameters need to be defined. For instance, by selecting different sampling periods during ADC initialization, the system can achieve varying sampling frequencies. The ADC channel sampling frequency *fsa* is expressed as:

$$f_{sa} = \frac{F_{ADCCLK}}{PSC * N * (T_{SC} + T_{ST}) * R_{OS}}$$

- F_{ADCCLK}, ADC Clock (ADCCLK): Derived from the system clock (MHz), where the minimum and the maximum values are 2 MHz and 80 MHz, respectively. In addition, only integer values are allowed.
- ADC Prescaler (PSC): Can be selected from {1, 2, 4, 6, 8, 10, 12, 16, 32, 64, 128, 256}.
- N: This equals to the total number of ADC channels used in the selected ADC module of the MCU board, which should be no more than 16.
- T_{ST}, Sampling Time: Each conversion requires 12.5 cycles for a 12-bit ADC resolution.
- T_{SC}, Sampling Cycles: Configurable values, can be selected from {2.5, 6.5, 12.5, 24.5, 47.5, 92.5, 247.5, 640.5}.
- R_{Os}, Oversampling Ratio: Hardware oversampling is defined as follows:
  - 12-bit resolution → 1×
  - 13-bit resolution → 4×
  - 14-bit resolution → 16×
  - 15-bit resolution → 64×
  - 16-bit resolution → 256×

For example, a sampling rate of 1056 Hz for 16-bit 8 ADC channels can be achieved by:

- 1056≈80*1000/[1×8×(12.5+24.5)×256].

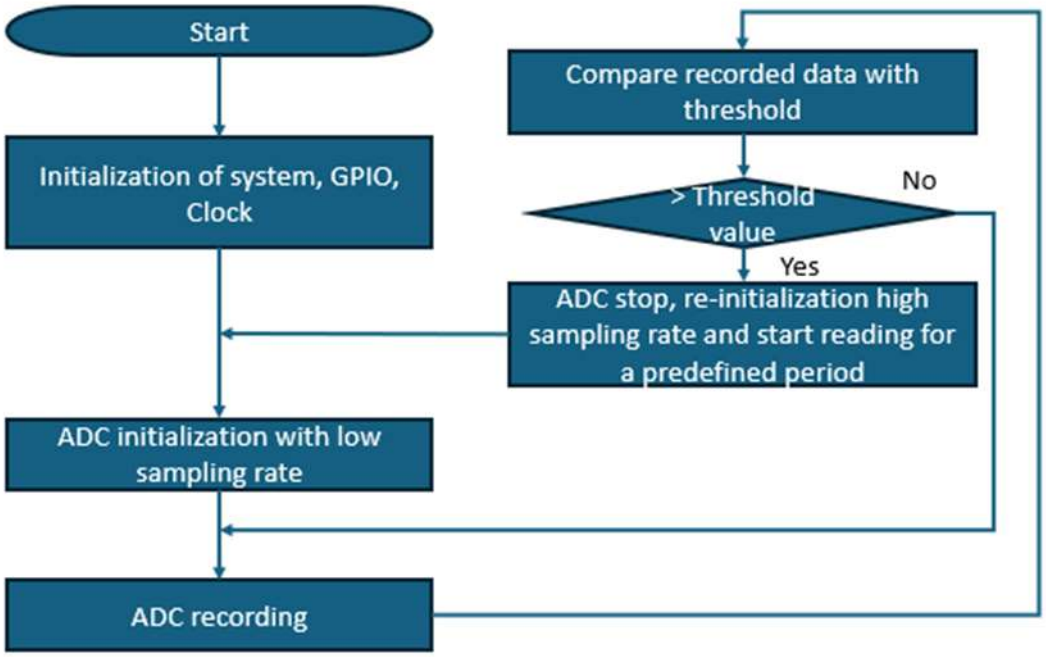


Figure 1 : Flowchart of event-triggered DAQ

The flowchart of the developed event-triggered data acquisition system (DAQ) is illustrated in Figure 3.3. It starts with system initialization, which includes the configuration of GPIO, system clocks, and other essential peripherals. Subsequently, the ADC is initialized and calibrated using low-frequency sampling parameters, and DMA circular mode is employed to enable continuous sampling and data output. According to the methodology outlined in Reports D2.1 and D2.2, a ping-pong buffer mechanism is used to alternately utilize the ADC buffers. The acquired sensor data is continuously compared against a predefined threshold value. When the ADC data exceeds the threshold, the system immediately halts ADC sampling, reconfigures the ADC with high-frequency sampling parameters, and initiates high-frequency sampling for a predefined duration while simultaneously outputting data. The duration of high-frequency sampling is determined by the sampling frequency and buffer size and is typically less than two seconds. For instance, during an impact event, the acceleration or strain data exhibits significant fluctuations only for a short period, necessitating a buffer size sufficient to capture data exclusively during the high-frequency sampling phase. Once the predefined duration expires, the ADC automatically stops and reinitializes using low-frequency sampling parameters. The system then resumes continuous low-frequency sampling using DMA circular mode, ensuring continuous data acquisition and output.

3.4 EXPERIMENTAL VALIDATION

To validate the event-triggered DAQ sampling mechanism, a single ADC channel was employed, with two distinct ADC sampling frequency configurations: Low-frequency sampling at 1531 Hz, and High-frequency sampling at 9523 Hz. An event condition was defined such that the sensor voltage exceeding 1.4V would trigger a sampling frequency transition. Specifically, when the ADC detects a sensor voltage greater than 1.4V, the system immediately switches to high-frequency sampling mode. Based on the predefined ADC buffer size, the high-frequency sampling persists for approximately 1.6 seconds before the DAQ automatically stops high-frequency sampling and reverts to the default low-frequency mode. During the DAQ sampling process, an oscilloscope was used to simultaneously record the sensor data to serve as a reference for comparative analysis. As illustrated in Figure 3.4, the event-triggered DAQ

system captures sensor data across three distinct phases. The recorded results—represented by the green and black curve for low-frequency sampling durations while the blue curve for high-frequency sampling periods—are compared against the reference values, depicted by the red curve.

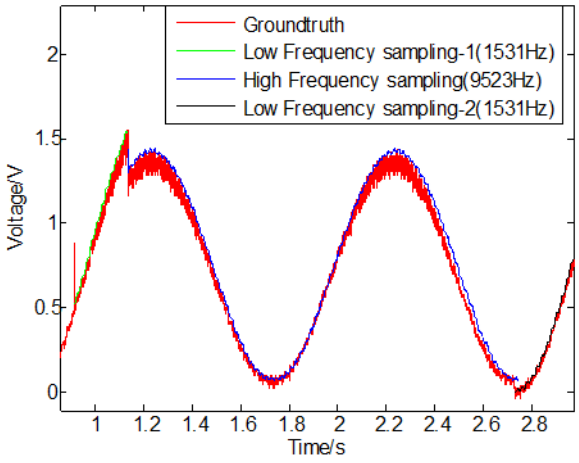


Figure 2 : Experimental results for event-triggered DAQ mechanism

4 WIRELESS AVIONIC INTRA-COMMUNICATIONS

4.1 OBJECTIVES

The primary objective of this project is to design, develop, and implement a robust infrastructure that enables efficient and reliable communication between a network of sensors and a central gateway, ensuring seamless data transmission. This infrastructure must support real-time data acquisition, aggregation, and forwarding to a cloud-based system for further processing, analysis, and long-term storage.

The system architecture will comprise multiple sensors, each responsible for collecting specific environmental or operational data, such as temperature, motion, or other relevant metrics, depending on the application. These sensors will be strategically deployed to ensure comprehensive data collection within the designated area or system. Each sensor node will be designed to operate with minimal power consumption while maintaining consistent data transmission rates to optimize efficiency and prolong battery life in cases where wireless or battery-powered sensors are used.

The gateway will act as a centralized hub that facilitates the communication between the sensors and the cloud infrastructure. It will be responsible for aggregating data received from multiple sensors, performing initial filtering or pre-processing where necessary to remove redundant or erroneous data, and ensuring data integrity before transmission. The gateway must support multiple communication protocols, such as Wi-Fi, Bluetooth, Zigbee, LoRa, or other relevant IoT communication standards, to accommodate various sensor types and deployment scenarios.

Once the gateway has successfully received and processed the sensor data, it will prepare and save the received data so that it will be sent to a cloud-based platform, where the data will be further processed, analyzed, and stored. The cloud infrastructure will provide scalable storage solutions and computational capabilities to perform advanced data analytics, such as trend detection, predictive modeling, and anomaly detection. The processed data can then be accessed via user interfaces, dashboards, or API endpoints for real-time monitoring, reporting, and decision-making.

To ensure the system operates effectively in diverse environments, considerations such as network reliability, latency, fault tolerance, and power management will be incorporated into the design. Redundancy mechanisms will be introduced to minimize data loss in case of network failures, and adaptive transmission techniques will be implemented to optimize bandwidth usage.

Overall, this infrastructure aims to provide a scalable, high-performance solution for real-time sensor data acquisition, processing, and cloud integration, catering to various industrial, environmental, and smart automation applications.

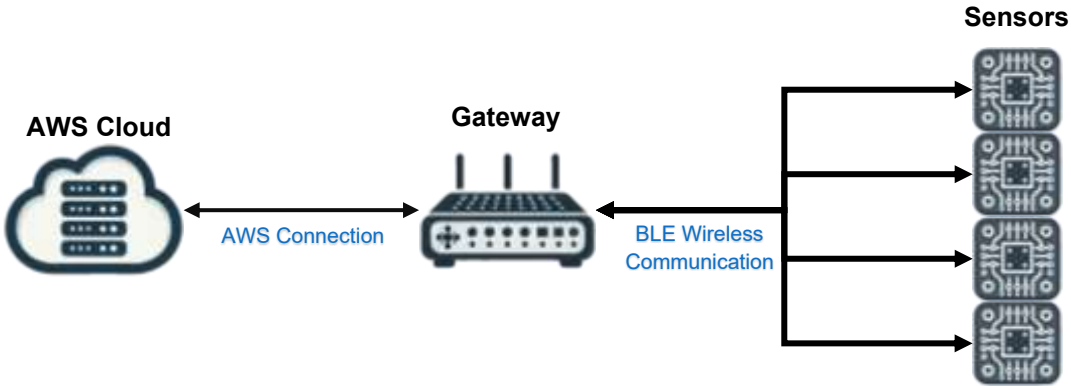


Figure 3 : AVATAR Wireless Data Transmission Overall Structure

4.2 NETWORK ARCHITECTURE

The network architecture will adopt a client-server model to facilitate structured and efficient communication between the gateway and the connected sensors. In this architecture, the gateway will function as the client, while the sensors will act as servers. This approach ensures a well-organized data exchange mechanism where the gateway actively requests or receives data from the sensors in a controlled manner.

The communication platform has been designed to function as a transparent data pipeline, ensuring that it operates independently of the type of data being transmitted. This design principle allows seamless scalability, flexibility, and modularity, making the system adaptable to various sensor configurations and data types.

Each sensor, operating as a server, will be responsible for collecting real-time data from its environment, such as temperature, motion, or other relevant metrics. The sensors will be designed to respond to data requests initiated by the gateway or periodically transmit data based on pre-defined intervals, event triggers, or threshold-based alerts. Depending on the communication protocol used, sensors may support a request-response model, a publish-subscribe mechanism, or a hybrid approach to optimize data flow and minimize latency.

The gateway, acting as the client, will establish and maintain communication sessions with multiple sensors, ensuring seamless data aggregation and processing. It will continuously receive data streams through "notify" mechanisms.

By implementing this client-server architecture, the system will achieve reliable and scalable sensor-to-gateway communication, providing a robust foundation for real-time data collection, processing, and cloud integration.

4.2.1 COMMUNICATION PLATFORM

In this project, we aim to establish a wireless communication link between two embedded systems using Bluetooth Low Energy technology. The setup consists of two primary components:

- A Raspberry Pi 5 (RPI 5) serving as the gateway node.
- A STM32L475VGT6 Discovery Kit functioning as the sensor node.

The gateway node (RPI 5) will be responsible for receiving and storing data transmitted from the sensor node. The sensor node (STM32L475VGT6) will acquire sensor data and send it wirelessly to the gateway via Bluetooth. To achieve reliable communication, both devices will utilize their integrated Bluetooth modules and appropriate software stacks.

4.2.1.1 BLE COMMUNICATION

Bluetooth Low Energy (BLE) operates using the Generic Attribute Profile (GATT), which defines how data is structured and exchanged between a client (typically a central device) and a server (typically a peripheral device). GATT defines three primary operations for data exchange: Write, Read, and Notify. These operations dictate how data is transferred between devices, influencing communication efficiency, latency, and power consumption. In this setup we're considering sensor node to act as the "server" and gateway to act as the "client".

4.2.1.1.1 WRITE OPERATION

The Write operation allows the client device to send data directly to a characteristic on the server device. This operation is commonly used when the client needs to configure the server, issue commands, or update stored data.

4.2.1.1.1.1 USE CASES

- Sending control commands from a client to a BLE-enabled device (server), such as toggling an LED or adjusting a motor speed.
- Updating configuration parameters on an embedded sensor node.
- Writing data logs to a BLE-enabled storage device.

4.2.1.1.1.2 IMPLEMENTATION DETAILS

There are two types of write operations in BLE:

1. Write Without Response (GATT Write Command)
    - The client writes data to the server without requiring an acknowledgment.
    - This method is faster and reduces latency but does not guarantee data reception.
    - Suitable for applications where occasional packet loss is acceptable (e.g., sending frequent sensor updates).
2. Write With Response (GATT Write Request)
    - The client writes data to the server and waits for an acknowledgment.
    - Ensures data integrity and confirms successful transmission.
    - Involves additional communication overhead, which may introduce slight latency.
    - Used for critical commands where reliability is essential.

4.2.1.1.1.3 ADVANTAGES AND CONSIDERATIONS

- ✓ Ensures reliable command execution when using Write with Response.
- ✓ Allows real-time control and configuration updates.
- ❖ May introduce latency due to acknowledgment requirements.
- ❖ Increased power consumption when sending frequent updates.

4.2.1.1.2 READ OPERATION

The Read operation allows the client device to request data from a characteristic on the server device. The server responds with the requested data, making it ideal for retrieving stored values or static configuration settings.

4.2.1.1.2.1 USE CASES

- Reading sensor calibration values or device firmware version.
- Retrieving the battery level of a BLE peripheral.
- Accessing a device's stored logs or configuration settings.

4.2.1.1.2.2 IMPLEMENTATION DETAILS

1- Synchronous Request-Response Model
   - The client sends a read request to the server.
   - The server responds with the requested data.

2- On-Demand Data Retrieval
   - The client reads data only when needed, rather than continuously polling.
   - Reduces unnecessary communication and power consumption.

4.2.1.1.2.3 ADVANTAGES AND CONSIDERATIONS

- ✓ Suitable for infrequently changing data.
- ✓ Ensures accurate data retrieval at the time of request.
- ❖ Increases communication overhead if used too frequently.
- ❖ Introduces slight latency since the client must explicitly request data.

4.2.1.1.3 NOTIFY OPERATION

The Notify operation enables the server device to automatically send data updates to the client device without requiring an explicit request. This event-driven approach is highly efficient for applications requiring real-time data streaming.

4.2.1.1.3.1 USE CASES

- Continuously sending real-time sensor data, such as temperature or heart rate readings.
- Sending alerts or status updates when an event occurs (e.g., motion detected).
- Reducing power consumption in battery-operated BLE peripherals by eliminating frequent polling.

4.2.1.1.3.2 IMPLEMENTATION DETAILS

1- Notification Subscription
   - The client subscribes to notifications by enabling the Notify property of a characteristic.

- The server then automatically sends data updates whenever the characteristic value changes.

2- Unacknowledged Transmission
   - Unlike Write with Response, notifications do not require acknowledgment from the client.
   - This allows fast, low-latency communication with minimal overhead.

4.2.1.1.3.3 ADVANTAGES AND CONSIDERATIONS

- ✓ Reduces power consumption by eliminating unnecessary polling.
- ✓ Minimizes latency since data is pushed instantly when changes occur.
- ✓ Ideal for real-time sensor streaming applications.
- ❖ No built-in acknowledgment mechanism, so packet loss is possible.
- ❖ Requires the client to actively listen for incoming notifications.

4.2.1.1.4 COMPARISION OF BLE OPERATIONS

Table 3 : BLE Types of Operations Comparison

Operation	Direction	Acknowledgment	Latency	Power Consumption	Use Case Example
Write	Client -> Server	Optional (depends on type)	Moderate	Medium	Sending commands, configuration updates
Read	Client -> Server	Required	High	Low (if infrequent)	Retrieving stored values, battery status
Notify	Server -> Client	No	Low	Very Low	Real-time sensor updates, alerts

4.2.1.1.5 CHOOSING THE OPERATION TYPE

Choosing the appropriate BLE operation depends on the specific requirements of the application:

- Write operations are best for sending configuration updates or commands from a client to a server.
- Read operations are useful for retrieving stored values on demand.
- Notify operations are optimal for real-time data streaming with minimal power consumption.

Since the primary objective of our communication system is to transmit real-time sensor data from the STM32L475VGT6 Discovery Kit (sensor node) to the Raspberry Pi 5 (gateway node) with minimal communication overhead while maintaining the lowest possible power consumption, we have carefully selected the Notify method as the preferred BLE operation for data transmission.

The Notify operation in Bluetooth Low Energy (BLE) communication allows the sensor node to send data to the gateway asynchronously and efficiently, without requiring an explicit request from the receiver. Unlike the Read operation, which involves periodic polling from the gateway and introduces additional communication latency, the Notify method ensures that data is immediately pushed from the sensor node only when an update occurs. This event-driven mechanism eliminates the need for continuous requests, thereby reducing unnecessary power consumption and optimizing bandwidth utilization.

Additionally, in contrast to the Write operation, which typically requires bidirectional communication and acknowledgment, the Notify method operates in a unidirectional, non-acknowledged manner, further minimizing latency and energy consumption. This is particularly beneficial in battery-operated embedded systems, where power efficiency is a critical design consideration.

By leveraging the Notify method, we can ensure low-latency, energy-efficient, and highly responsive data transmission, making it the ideal choice for real-time sensor applications where rapid updates and minimal power drain are essential.

4.2.1.2 RASPBERRY PI 5 BLUETOOTH MODULE

The Raspberry Pi 5 is a single-board computer equipped with built-in Bluetooth functionality. This allows it to act as a central device or a peripheral in a Bluetooth network. You can see a picture of the board without any hardcases in Figure 4.



Figure 4 : RPI 5

4.2.1.2.1 HARDWARE ASPECTS OF RPI 5 BLUETOOTH

The Raspberry Pi 5 integrates Bluetooth technology within its onboard wireless communication module:

- **Chipset**: The RPI 5 uses a “Cypress CYW55572” module, which supports Bluetooth 5.0.

- **Antenna**: The module features an onboard PCB antenna for wireless connectivity.
- **Communication Interface**: The Bluetooth module interfaces with the system through UART or HCI (Host Controller Interface).
- **Power Management**: Integrated low-power operation modes allow efficient energy usage, which is essential for embedded systems.

4.2.1.2.2 SOFTWARE ASPECTS OF RPI 5 BLUETOOTH

To establish Bluetooth communication, the Raspberry Pi 5 relies on Linux-based software stacks and utilities.BLE supports three primary methods for data transfer, each serving different communication needs within a system:

- **Operating System**: Raspberry Pi OS (Debian-based)
- **Drivers & Firmware**: The “brcmfmac” driver is responsible for managing wireless communications, including Bluetooth operations.
- **Configuration & Utilities**: “bluetoothctl” for command-line Bluetooth device management.
- **Programming Interfaces**: C/C++ libraries for low-level Bluetooth communication like “simpleBLE”.

By leveraging these capabilities, the Raspberry Pi 5 can establish and manage Bluetooth Low Energy (BLE) or classic Bluetooth communication with the sensor node.

4.2.1.3 STM32L475VGT6 DISCOVERY KIT

The STM32L475VGT6 Discovery Kit is an evaluation board designed for IoT applications, featuring an STM32L4-series microcontroller with ultra-low power consumption. This board includes an integrated Bluetooth module, enabling it to function as a wireless sensor node.

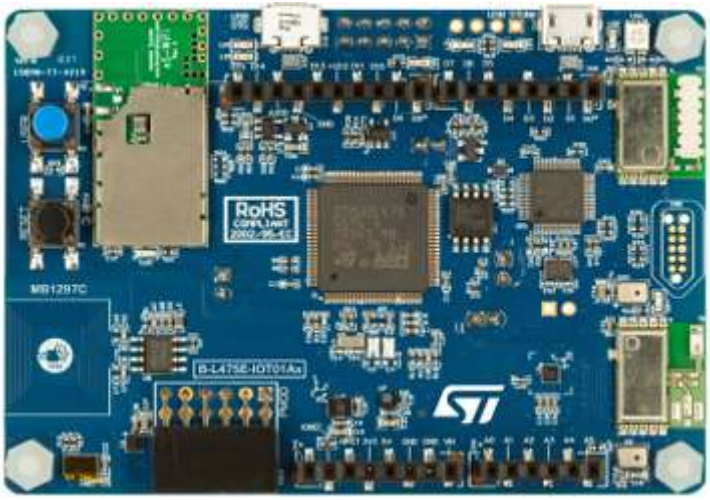


Figure 5 : STM32L475VGT6 board

4.2.1.3.1 HARDWARE ASPECTS OF STM32L475VGT6 BLUETOOTH

- **Bluetooth Module**: SPBTLE-RF, based on the BlueNRG_MS SoC.
- **Communication Interface**: The Bluetooth module communicates with the STM32 microcontroller via SPI.

- **Antenna**: Integrated ceramic antenna for wireless communication.
- **Power Efficiency**: The module operates at an ultra-low power level, which is critical for battery-operated IoT devices.

4.2.1.3.2 SOFTWARE ASPECTS OF STM32L475VGT6 BLUETOOTH

The Bluetooth software stack for the STM32L475VGT6 Discovery Kit is primarily based on the **ST** BlueNRG_MS firmware and the STM32Cube ecosystem.

- **Bluetooth Stack**:
  - **ST BlueNRG_MS SDK**: Implements BLE communication, including GAP (Generic Access Profile) and GATT (Generic Attribute Profile) support. In this setup we've used GATT for the comm
  - **BLE Profiles**: Supports standard BLE profiles like Heart Rate**,** Environmental Sensor**,** and Custom Services.
- **Programming & Development Tools**:
  - **STM32CubeIDE**: Integrated development environment for firmware development.
  - **STM32CubeMX**: Used for peripheral configuration and middleware setup.
  - **HAL (Hardware Abstraction Layer)**: Provides APIs for SPI communication with the Bluetooth module.
- **Firmware & Libraries**:
  - **BlueNRG_MS Firmware**: Runs on the Bluetooth module to handle low-level BLE operations.
- **Debugging & Monitoring**:
  - **ST-Link Debugger**: Used for real-time debugging of firmware.
  - **BLE Scanner Apps**: Android/iOS applications can be used to test and verify Bluetooth communication.

4.2.1.4 INTERFACING THE SENSOR NODE WITH THE GATEWAY

To establish a communication link between the STM32L475VGT6 Discovery Kit and the Raspberry Pi 5, the following steps need to be implemented:

- Configure the STM32 board as a Bluetooth Peripheral (advertising sensor data over BLE).
- Enable the Raspberry Pi 5 as a Bluetooth Central Node (scanning and connecting to the STM32 board).
- Establish a GATT (Generic Attribute Profile) Connection to read sensor values from the STM32 device.
- Process & Forward Data from the Raspberry Pi 5 to other network interfaces if required (e.g., MQTT broker, database, or cloud services).

In this particular communication platform, the notify operation is the most appropriate method for transferring data from the sensors to the gateway. Sensors continuously collect environmental or operational data, such as temperature or other parameters, and must transmit this information efficiently

to the gateway for further processing. Using the notify method allows the sensors to push updates only when new data is available, optimizing power usage and reducing unnecessary communication overhead. Additionally, since notifications do not require acknowledgment, data transmission occurs with minimal delay, ensuring near-instantaneous updates in real-time monitoring applications.

By leveraging BLE notifications, the communication platform achieves a balance between energy efficiency and responsiveness, making it well-suited for IoT deployments where power conservation and real-time data transmission are critical requirements.

4.3 IMPLEMENTED SOLUTIONS

To achieve the desired functionality, two distinct implementations must be developed—one for the sensor and one for the gateway. Each implementation serves a unique role in the communication process, ensuring seamless data transmission and efficient system operation.

4.3.1 GATEWAY IMPLEMENTATION

The gateway application follows a structured workflow consisting of multiple key steps to ensure reliable and efficient data exchange with the sensor nodes:

1- **Device Scanning and Identification**

- Initially, the gateway application scans the surrounding area for available BLE devices. This process is necessary to detect all nearby devices that are advertising their presence.
- Since each sensor node has a unique Hardware Address (MAC address), the gateway does not attempt to connect to all available devices. Instead, it searches specifically for the predefined hardware addresses of the sensor nodes. This filtering mechanism ensures that only the designated sensor devices are discovered and connected to the gateway.

2- **Establishing a BLE Connection**

- Once the sensor nodes are identified during the scanning phase, the gateway initiates a BLE connection to each detected device.
- Establishing a stable and reliable connection is essential, as this enables the exchange of timestamp synchronization data and real-time sensor readings.

3- **GATT Services**

- Each sensor node provides two distinct GATT (Generic Attribute Profile) services:
  - **Timestamp Reference Service** – Used for synchronizing time between the gateway and the sensor node.
  - **Data Streaming Service** – Used for transmitting real-time sensor data to the gateway.

4- **Timestamp Synchronization**

- For the timestamp, we're using the Epoch timestamp.
  - Epoch time, also known as Unix time or POSIX time, is a widely used system for tracking time in computing and embedded systems. It represents the number of seconds that have elapsed since a fixed reference point: January 1, 1970, 00:00:00 UTC (Coordinated Universal Time). This moment, known as the Unix Epoch,

serves as the universal zero-point for timekeeping in Unix-based operating systems and many modern programming environments.
    - Epoch time is typically represented as an integer count of seconds (for standard timestamps) or as a floating-point value (for sub-second precision, such as milliseconds or microseconds).
    - Example: 1707052800 (Represents February 4, 2024, 00:00:00 UTC)
- At the beginning of the connection, the gateway sends a timestamp to the sensor node's GATT Timestamp Reference Service. This timestamp serves as a reference for all subsequent data transmissions.
- From this point forward, the sensor node should use this reference timestamp for all its outgoing data packets, ensuring accurate and synchronized time representation across all measurements.

5- Data Reception via BLE Notifications

- After the initial timestamp synchronization, the gateway subscribes to BLE notifications from the sensor node's Data Streaming Service.
- The sensor node continuously sends real-time data to the gateway using the Notify method. This event-driven approach allows efficient data transmission without the need for continuous polling, reducing latency and power consumption.

6- Data Logging and Storage

- Each incoming data packet from a sensor node is logged into a separate file, ensuring organized data storage.
- A dedicated file is maintained for each sensor node, allowing the system to handle multiple nodes efficiently.
- These data files serve as a local buffer, ensuring that all sensor readings are stored reliably before further processing.

7- Cloud Integration and Data Forwarding

- The logged sensor data is made available for another software module responsible for reading the stored files and transmitting the data to the cloud.
- This approach allows for scalability, as the system can be easily extended to support cloud-based storage, remote monitoring, and data analytics.

4.3.2 SENSOR IMPELEMENTAION

The sensor node is implemented using the B-L475E-IOT01A2 Discovery Kit, an advanced IoT development board based on the STM32L475VGT6 microcontroller. This board includes an onboard BlueNRG_MS Bluetooth Low Energy (BLE) module, which enables wireless communication with the Raspberry Pi 5 gateway.

The firmware development for the sensor node is carried out using STM32CubeIDE (version 1.16.1), an integrated development environment that provides a comprehensive toolset for STM32 microcontroller programming, peripheral configuration, and BLE stack implementation. The BLE functionality is implemented using the STM32Cube firmware package, which provides libraries and middleware for seamless Bluetooth communication.

4.3.2.1 BLE SERVICES AND CHARACTERISTICS CONFIGURATION

Bluetooth Low Energy (BLE) devices are structured using the Generic Attribute Profile (GATT), which organizes data exchange through services and characteristics. In this implementation, the sensor node is configured with the following GATT services:

1. **Sensor Data Transmission Service**
    - Contains a characteristic for sending real-time sensor data from the sensor node to the gateway.
    - Configured to use the Notify method for efficient, low-latency data transmission with minimal power consumption.
2. **Timestamp Synchronization Service**
    - Contains a characteristic for receiving a timestamp from the gateway.
    - Used to establish a synchronized reference time for all subsequent sensor data packets.

Each of these services and characteristics is defined during the initialization phase of the BLE module.

4.3.2.2 BLE INITIALIZATION AND ADVERTISING

The BLE module initialization process consists of:

- Creating the required GATT services and characteristics within the BlueNRG_MS module.
- Setting the sensor node to operate as a BLE Peripheral, enabling it to advertise its presence.
- Broadcasting an advertising packet that allows the Raspberry Pi 5 gateway to detect and establish a connection with the sensor node.

Once initialized, the sensor node continuously advertises its availability, allowing the gateway to discover and initiate a BLE connection.

4.3.2.3 TIMESTAMP SYNCHRONIZATION AND REFERENCE STORAGE

When the gateway establishes a connection with the sensor node, it transmits a timestamp value representing the current Epoch time. The sensor node receives this timestamp via the Timestamp Synchronization Service and stores it in a dedicated variable called "Ref_EpochTime". This reference timestamp is used to generate accurate, synchronized timestamps for all subsequent sensor data packets transmitted to the gateway.

This mechanism ensures that all recorded sensor data is time-aligned, allowing for precise data correlation when it is processed by the gateway.

4.3.2.4 DATA BUFFERING AND TRANSMISSION MECHANISM

To ensure efficient and reliable data transmission, a buffering system is implemented to manage outgoing data:

1. **Data Buffering**
    - A circular buffer is implemented to temporarily store sensor data before transmission.

- This buffer enables efficient handling of continuous real-time data streams, preventing data loss due to transmission delays.

2. **Data Transmission Function**
   - A dedicated function is responsible for accepting sensor data as an input parameter.
   - The function stores the received data in the circular buffer and schedules it for transmission.
3. **Transmission Process**
   - The buffer is checked every 10 milliseconds to determine if data is available for transmission.
   - Data is sent in 20-byte packets (the Maximum Transferable Unit size because of BLE limitation) at the same 10 milliseconds interval.
   - The process continues until the buffer is completely emptied, ensuring that all sensor data is successfully transmitted to the gateway.

This approach ensures that data is transmitted efficiently without overloading the BLE connection, maintaining a balance between real-time performance and low power consumption.

4.3.3 LIMITATIONS

The current implementation is designed to support a data transfer rate of approximately 135 Kbps, which is required to handle 8 sensor channels, each transmitting 2-byte samples at a sampling rate of about 1 KHz. However, the BlueNRG_MS Bluetooth Low Energy (BLE) module, used in the sensor nodes, imposes certain constraints that limit the achievable data throughput and affect real-time communication performance.

4.3.3.1 BLE LIMITATIONS IN THE CURRENT IMPLEMENTATION

1- **Connection Interval Constraints**
   - The notify mechanism in BLE is used to transmit sensor data from the sensor nodes to the gateway.
   - While the theoretical minimum connection interval for the BlueNRG_MS module is 7.5 milliseconds, practical experimental testing has shown that a stable data stream can only be maintained with a 10–15 millisecond connection interval.
   - A longer connection interval reduces the number of packets that can be transmitted per second, limiting overall data throughput.

2- **MTU Size Restriction**
   - The Maximum Transferable Unit (MTU) size determines the amount of data that can be sent in a single BLE packet.
   - The BlueNRG_MS module enforces an MTU size limit of 20 bytes when operating in notify mode.

- Since each data packet contains sensor samples, the 20-byte limit means that only 10 samples (2 bytes each) can be transmitted per packet.
- This restriction increases the number of packets required to transmit all data, further limiting throughput.

3- **Combined Impact on Data Throughput**
   - Given the MTU limitation (20 bytes per packet) and the practical connection interval (10–15 milliseconds), the maximum achievable data rate is significantly lower than the required 135 Kbps (about **10Kbps**).
   - These BLE constraints are inherent limitations of the BlueNRG_MS hardware and cannot be further optimized through firmware improvements.

4.3.3.2 POSSIBLE SOLUTIONS

To overcome the limitations of the current BLE implementation, different solutions can be applied based on the specific requirements of the project. These solutions primarily focus on either hardware upgrades to improve BLE communication efficiency or software-based optimizations to reduce bandwidth requirements.

4.3.3.2.1 PROJECT REQUIREMENTS

The current system is designed to transmit ADC data and timestamp information from the sensor node (MCU) to the gateway. The required data rate must accommodate:

1- 8 ADC channels
   a. Each channel generates 2-byte samples.
   b. The sampling frequency is 1056 Hz.
2- 1 additional ADC channel
   a. Generates 2-byte samples.
   b. The sampling frequency is 1 Hz.
3- Timestamp Transmission
   a. A 4-byte Epoch timestamp is transmitted once per second to synchronize the data batches.
4- Batch Transmission
   a. The MCU sends a new data batch every second to the gateway.

The required bandwidth is calculated as follows:

$$(8 \times 2 \times 1056 \times 8) + (1 \times 2 \times 1 \times 8) + (1 \times 4 \times 8)\, bits\ per\ second$$

Breaking it down:

- **8 ADC channels**:

$$(8 \times 2\ bytes \times 1056\ Hz \times 8) = 135.17\ Kbps$$

- **1 ADC channel**:

$$(1 \times 2\ bytes \times 1\ Hz \times 8) = 16\ bps$$

- **Timestamp (Epoch Time, 4 bytes, 1 Hz)**:

$$(1 \times 4\ bytes \times 8) = 32\ bps$$

- **Total required bandwidth: ≈ 135.2 Kbps**

With the current BlueNRG_MS BLE module, achieving this data rate is challenging due to MTU size limitations and connection interval constraints. Therefore, to ensure stable and efficient data transmission, the following solutions are proposed.

4.3.3.2.2 HARDWARE RELATED SOLUTION

To address the hardware constraints of the BlueNRG_MS BLE module, an upgrade to the BlueNRG-2 module is recommended.

- **Advantages of Upgrading to BlueNRG-2**
  - Increased MTU Size – The MTU size increases from 20 bytes (BlueNRG_MS) to 158 bytes (BlueNRG-2), enabling larger data packets per BLE transmission and reducing protocol overhead.
  - Improved Data Throughput – The enhanced BLE stack in BlueNRG-2 allows for higher sustained data rates, making it possible to meet the 135.2 Kbps requirement.
  - Optimized Power Consumption – The BlueNRG-2 module includes power-saving features, which improve energy efficiency compared to BlueNRG_MS.
- **Advantages of Upgrading to BlueNRG-2**
  - The increased MTU size allows larger packets per connection event, reducing the number of required packets and improving throughput.
  - The higher efficiency of BlueNRG-2 can enable the system to achieve a stable data rate that meets project requirements.

4.3.3.2.3 NON-HARDWARE RELATED SOLUTION

If upgrading to BlueNRG-2 is not feasible, an alternative approach is to optimize the data transmission strategy by reducing the ADC sampling frequency.

- **Proposed Software Optimization**
  - Reduce the sampling frequency of the 8 ADC channels from 1056 Hz to 50 Hz
- **Revised Bandwidth Calculation (After Frequency Reduction)**
  - If the sampling frequency is reduced to 50 Hz, the new bandwidth requirement is: $(8 \times 2 \times 50 \times 8) + (1 \times 2 \times 1 \times 8) + (1 \times 4 \times 8)bits\ per\ second$
- **Breaking it down**:
  - 8 ADC Channels (50 Hz):

$$(8 \times 2\ bytes \times 50\ Hz \times 8) = 6.4\ Kbps$$

  - 1 ADC Channel (1 Hz):

$$(1 \times 2\ bytes \times 1\ Hz \times 8) = 16\ bps$$

  - Timestamp (Epoch Time, 4 bytes, 1Hz):

$$(1 \times 4\ bytes \times 8) = 32\ bps$$

- **New required bandwidth**: **≈ 6.5 Kbps**
- **Impact of Reducing Sampling Frequency**
  - Drastically lowers bandwidth requirements from 135.2 Kbps to ≈6.5 Kbps, making it fully compatible with the current BlueNRG_MS module.
  - Ensures stable data transmission without packet loss by reducing transmission frequency.

- Minimizes BLE overhead and power consumption, extending battery life for sensor nodes.

4.4 TESTING AND VALIDATION

The testing procedure was designed to evaluate the BLE communication performance between the sensor nodes and the gateway under real-world operating conditions. The primary objective was to determine the maximum achievable data rate, identify potential data losses, and assess the impact of adding multiple BLE services on the reliability and stability of the transmission.

4.4.1 BLE PERFORMANCE TESTING METHODOLOGY

To assess the BLE module's capabilities, a test project was developed that integrates the BlueNRG_MS module on the sensor side to transmit 1-byte counter data at the maximum possible rate to the gateway. This setup allowed for a controlled and repeatable evaluation of data transmission efficiency.

1- **Testing Setup**
   - The test environment consisted of two sensor nodes and one gateway.
   - The sensor nodes were programmed to send 1-byte counter values continuously at the highest achievable rate.
   - The gateway device was responsible for receiving the counter values and logging the received data into a file for analysis.
   - The logs were examined for packet loss, transmission stability, and timing consistency.
2- **Initial Performance Results (Single BLE Service)**
   - Before introducing the Epoch timestamp synchronization service, the sensor nodes were able to transmit counter data with no packet loss when using a 10-millisecond send interval.
   - This confirmed that, under optimal conditions, the BLE communication was stable and reliable at this interval.
3- **Impact of Adding a Second BLE Service**
   - After introducing the second BLE service for transmitting the Epoch timestamp, occasional packet losses were observed at the same 10-millisecond send interval.
   - The observed losses were due to hardware limitations of the BlueNRG_MS module, which restricted the number of packets that could be sent within a given connection interval.

- The simultaneous transmission of two different types of data exceeded the maximum allowable BLE throughput, leading to dropped packets.

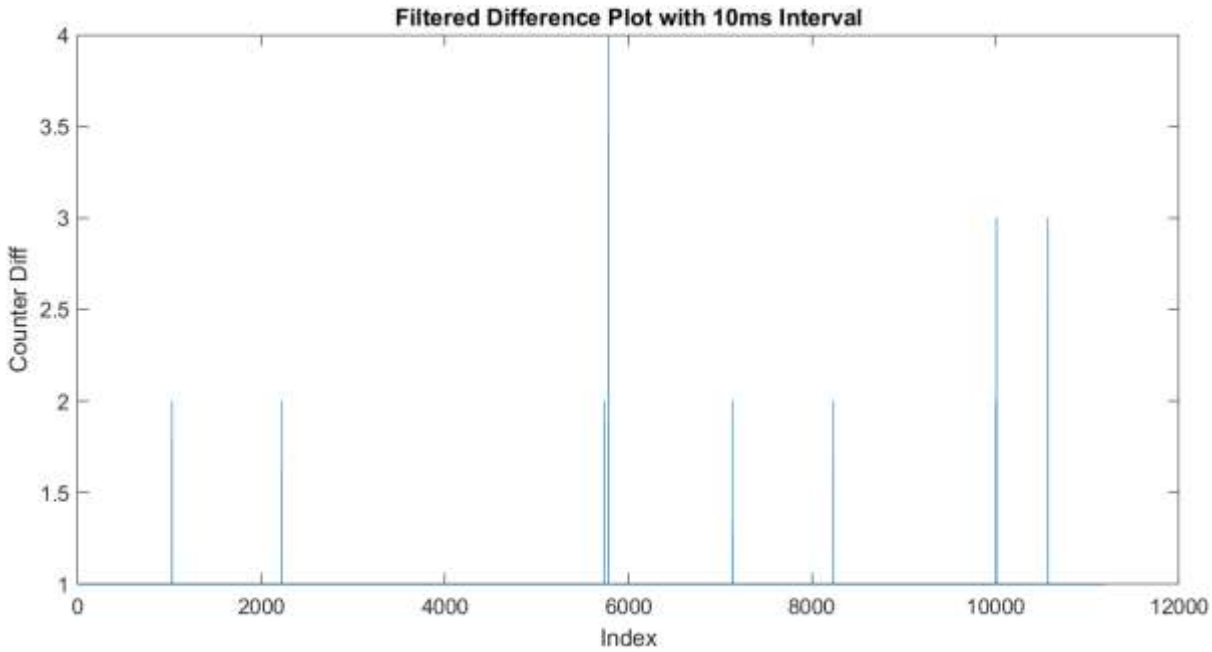


Figure 6 : Counter Diff Chart for 10ms Interval Data Transmission

4- Optimizing the Transmission Interval

- To eliminate data loss, the send interval was increased to 15 milliseconds.
- At this new interval, packet losses were no longer observed, and data transmission remained stable and consistent.
- The increased interval reduced the packet collision probability and allowed the BLE module to process and transmit both services without exceeding its data handling capabilities.

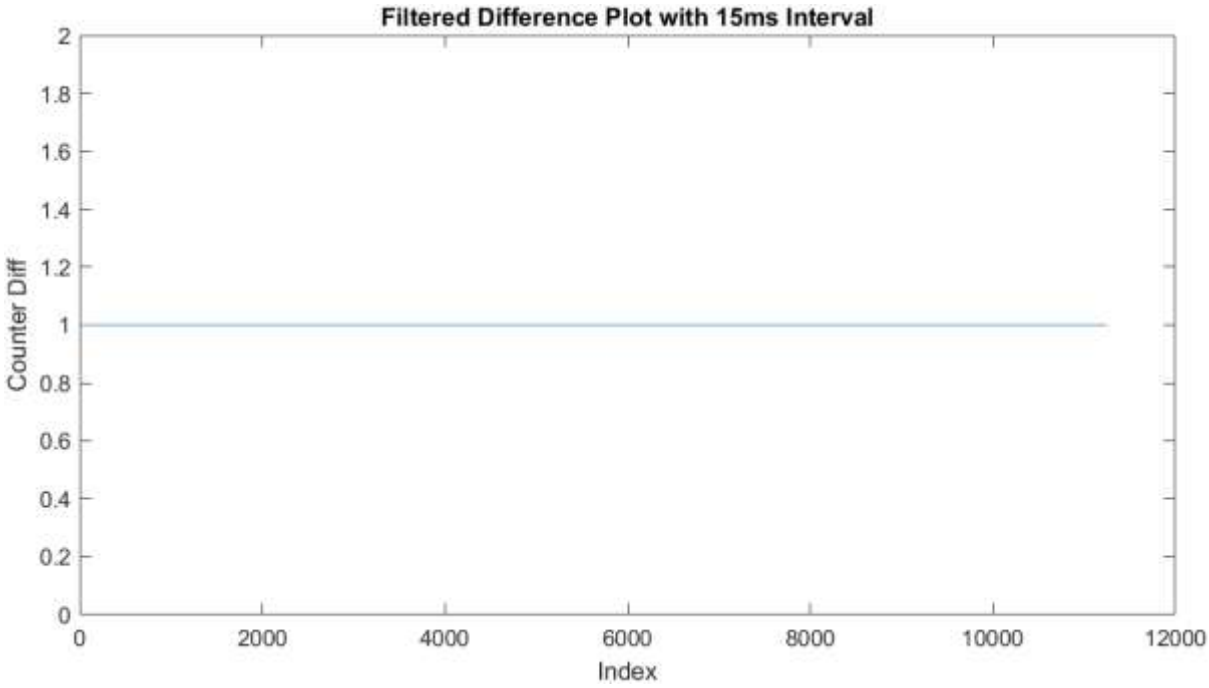


Figure 7 : Counter Diff Chart for 10ms Interval Data Transmission

5 AIRCRAFT TO GROUND SYSTEM COMMUNICATION

5.1 OBJECTIVES

The primary objective of WP3.4 is to establish a robust and efficient communication link between aircraft, drones, edge devices (data storage and pre-processing), and cloud systems to facilitate data research and analysis. This ensures seamless transmission of real-time sensor data, to support predictive maintenance, operational monitoring, and digital twin integration for data visualization. The key goals include:

- Selecting appropriate communication technologies for reliable and secure data transmission.
- Ensuring minimal latency while maintaining high data integrity.
- Developing an optimized implementation roadmap for integrating aircraft-cloud communication systems within the AVATAR project.



Figure 8 : Shows high level data transmission from the Aircraft sensors to the cloud visual representation in a Digital Twin and Data Analysis.

5.2 DATA COLLECTION AND TRANSMISSION METHODS

This section explores multiple aircraft-to-cloud communication technologies, evaluating their feasibility for seamless data transfer in aerospace environments. The selection process considered key factors such as:

- Latency: Minimizing delays in transmitting critical operational data.
- Bandwidth: Accommodating large volumes of sensor and telemetry data.
- Security: Implementing encryption and authentication mechanisms to prevent data breaches.
- Scalability: Supporting future enhancements and integration with additional aircraft systems.

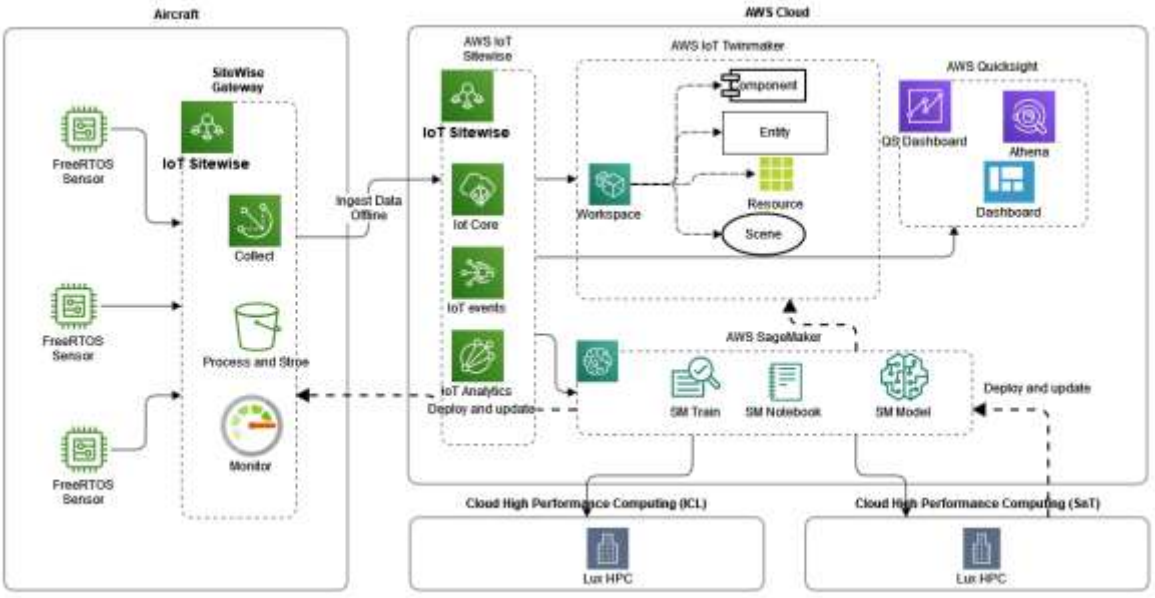


Figure 9 : Technical High Level Data Architecture: Astero aircraft

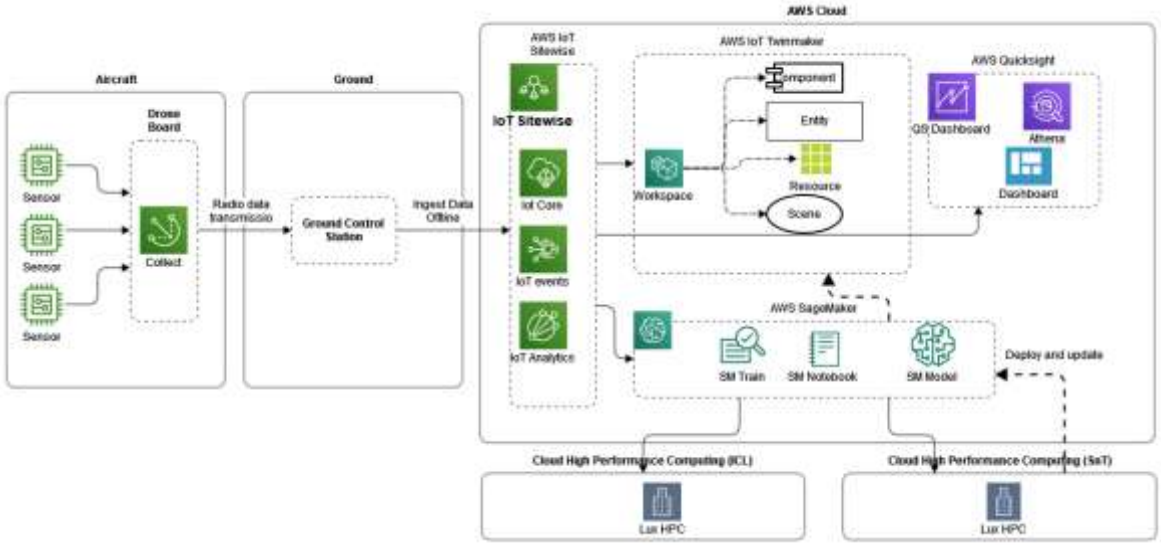


Figure 10 : Technical High Level Data Architecture: Astero aircraft

The data collection and transmission workflow involve multiple stages:

1. **Data Acquisition from Sensors:** Various sensors embedded within the aircraft and drones monitor key parameters such as temperature, pressure, vibration, structural integrity, and environmental conditions. These sensors generate continuous streams of raw data, which are crucial for predictive maintenance and operational monitoring.
2. **Microcontroller Unit (MCU) Processing:** The collected data is transmitted to a Microcontroller Unit (MCU), which serves as the first processing point. The MCU organizes and timestamps the raw sensor data before forwarding it to an Edge Device.

3. **Edge Device Data Aggregation:** A Raspberry Pi-based Edge Device is deployed onboard in the case of EVEKTOR Aircrafts or attached to the Radio Frequency Controller in case of the Drone of Nordic Wing, acting as an intermediary system for data collection.

   The Edge Device collects data through one of the three communication options:

Table 4 : Research Transmission Methods

Transmission Method	Developer	Key Features
Bluetooth Low Energy (BLE)	University of Luxembourg (UL) Loughborough University (LBORO)	Secure, low-energy data transmission from MCU to Edge Device
Wired Transmission	Loughborough University (LBORO)	High reliability, enhanced bandwidth, minimal latency, reduced data loss risks
Radio Frequency (RF)	Nordic Wing (Drone Application)	Real-time data transmission from the drone's Edge Device to a Ground Control Station (GCS), ensuring low-latency and continuous data collection

4. **Local Storage on the Edge Device:** The Edge Device temporarily stores the collected data, allowing it to be retrieved and transferred operational flight.

5. **Data Processing:** Once the operational flight is completed, the aircraft or drone lands and the data collected is ready, **a specialized service, developed by NTT Luxembourg,** is triggered within the Edge Device to **process, clean, and structure the data** before transmission to the Cloud. This ensures:

   I. **High-quality, structured data** is stored, improving analytics accuracy.
   II. **Efficient retrieval and cloud transfer** after the operational flight.

6. **Cloud Transmission via AWS IoT SiteWise Edge:** NTT Service uses **AWS IoT SiteWise Edge** with **AWS Greengrass v2 Stream Manager** to facilitate the secure and structured transfer of data from the Edge Device to the AWS Cloud. This system enables:

   I. **Real-time data streaming** to AWS SiteWise for monitoring and analysis.
   II. **Efficient data compression and batching** for optimal cloud transfer.
   III. **Secure authentication and encryption** for data integrity and protection.
   IV. **Remote monitoring and control capabilities** for seamless integration with the AVATAR digital twin platform.

Note: Alternatively, a **cloud transmission method using MQTT-enabled SiteWise gateways** was tested. This approach offers a **scalable and lightweight data transfer solution**, ensuring **high interoperability and flexibility** for future cloud integration and expansion requirements.

5.3 TECHNOLOGY COMPARISON

A comparative analysis of potential communication technologies was conducted to determine the most suitable approach. The evaluation focused on:

5.3.1 WIRELESS AND WIRED COMMUNICATION TECHNOLOGIES

Table 5 : Communication Technologies Comparison

Technology	Latency	Bandwidth	Security	Use Case
BLE (UL, LBORO)	Low	Moderate	High	Short-range aircraft communication
Wired (LBORO)	Very Low	High	High	Aircraft wired sensor integration
RF (NW Drone)	Low	Moderate	Moderate	Long-range drone communication

Considering the AVATAR project's requirements for **high availability, security, and low-latency transmission**, the combination of **BLE (UL, LBORO) and wired transmission (LBORO) for aircraft, and RF for drone-cloud transmission** has been selected as the optimal solution.

5.4 TESTING AND VALIDATION

To ensure the reliability of the data transmission and processing framework, Technology Readiness Level (TRL) testing was conducted at multiple stages:

5.4.1 TRL TESTING STAGES

- TRL 1 – Basic Principles Observed: Theoretical studies and initial simulations were conducted to evaluate the feasibility of sensor data collection and cloud integration.
- TRL 2 – Technology Concept Formulated: The proposed system architecture was developed, and a preliminary prototype was tested in a controlled lab environment.
- TRL 3 – Experimental Proof of Concept: A working prototype was deployed on a test aircraft to validate real-time data collection, processing, and transmission capabilities.

Throughout the testing phases, **data standardization was implemented** to ensure consistency in the structure, format, and transmission of data across various test environments.

5.4.2 END-TO-END TESTING

The final test was a full end-to-end validation, confirming successful data transmission from sensors to the cloud. This test validated:

- Seamless integration between hardware (sensors, MCU, Edge Device) and software (data processing, AWS SiteWise Edge).
- Data accuracy and completeness after transmission to the cloud.
- System resilience in various operational scenarios.

5.5 CLOUD INTEGRATION AND DATA PROCESSING

Once the data reaches the AWS Cloud, it undergoes a structured transformation process to ensure seamless integration with advanced analytics, machine learning models, and digital twin simulations. The cloud-based processing includes:

- Data Structuring and Tagging: All collected data is assigned metadata such as timestamps, sensor IDs, aircraft/drone identifiers, and operational context for precise tracking and historical analysis.

- Machine Learning and Predictive Analytics: The AWS IoT SiteWise platform leverages advanced algorithms to identify potential failures, anomalies, and inefficiencies, enabling proactive maintenance and reducing operational risks.
- Real-Time Dashboards and Visualizations: AWS SiteWise and Grafana-based dashboards provide interactive monitoring tools that display real-time sensor data and predictive insights.
- Historical Data Storage and Research Analysis: The cloud infrastructure supports long-term storage and data retrieval for research, trend analysis, and cross-aircraft performance benchmarking.

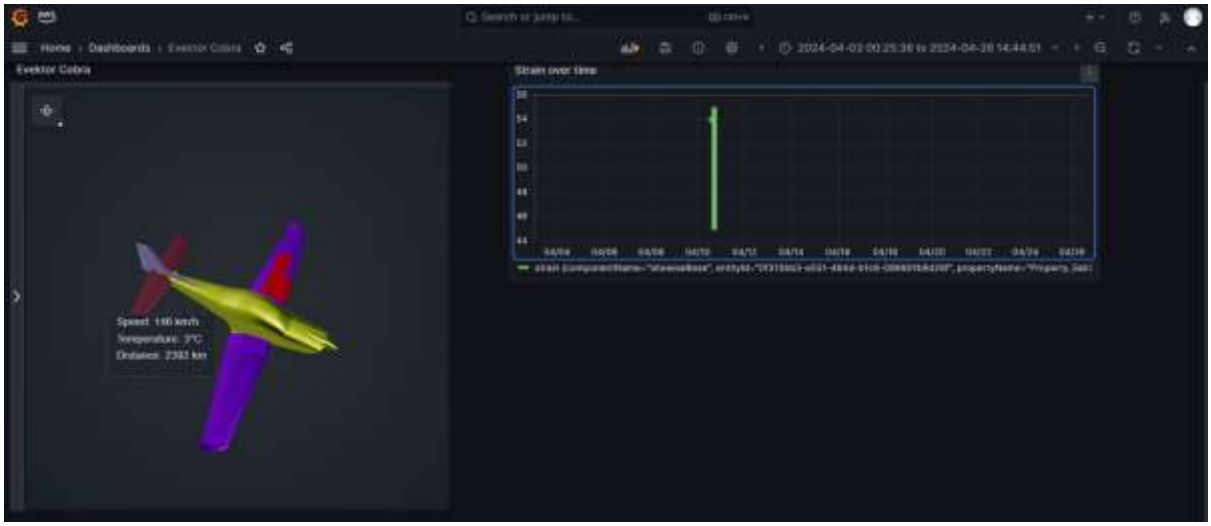


Figure 11 : Grafana board representing a Digital Twin of Evektor Cobra.

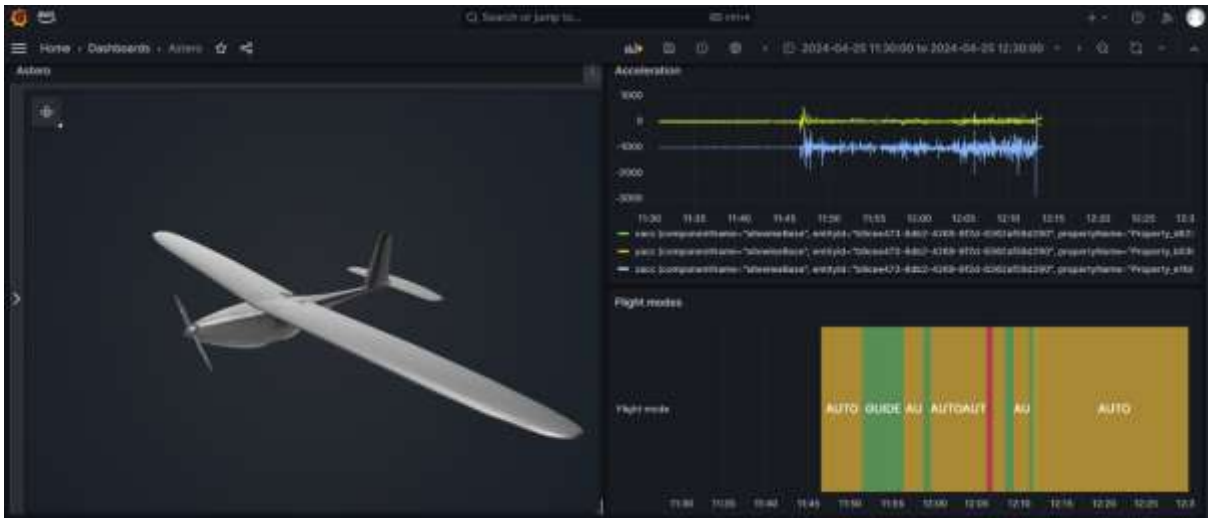


Figure 12 : Grafana board representing a Digital Twin of Nordic Wing Astero drone.

5.6 IMPLEMENTATION ROADMAP

The implementation of the aircraft-to-cloud system communication framework follows a phased approach:

Phase 1: Technology Validation

- Conduct real-world testing of BLE, wired transmission, and RF communication under various conditions.
- Validate integration with AWS IoT SiteWise and predictive maintenance frameworks.

Phase 2: System Integration

- Deploy BLE and wired transmission modules within EVEKTOR aircraft.
- Implement RF-based communication for Nordic Wing drones.
- Establish secure data pipelines to transfer collected sensor data to AWS Cloud upon aircraft landing.

Phase 3: Optimization & Scalability

- Optimize data compression and transmission scheduling to reduce bandwidth usage.
- Expand the communication infrastructure to support additional aircraft and drones in the AVATAR ecosystem.

By leveraging **edge computing for data preprocessing, encryption for security, and a hybrid transmission model**, this section ensures that the AVATAR project achieves reliable and efficient aircraft-to-cloud communication, supporting enhanced operational monitoring and predictive maintenance.

6 CONCLUSION

Within the AVATAR communication framework, the strategic selection of Bluetooth Low Energy (BLE) for aircraft and Radio Frequency (RF) transmission for drones reflects a deliberate balance between performance, power efficiency, and system integration constraints.

For manned aircraft platforms like EVEKTOR, BLE provides a compact, energy-efficient, and reliable wireless solution for intra-aircraft communication. BLE's low overhead and notify-based data streaming enable real-time sensor updates with minimal power consumption, making it ideal for embedded avionics without introducing additional cabling or excessive weight. BLE's seamless integration with existing microcontroller platforms, such as STM32, and gateway devices like the Raspberry Pi, ensures a lightweight and scalable solution for aircraft systems where weight and modularity are critical.

In contrast, drones such as the Nordic Wing Astero face more stringent weight and energy constraints, particularly regarding payload capacity and flight autonomy. Introducing BLE gateways or additional microcontroller infrastructure would increase onboard complexity and weight. Instead, direct RF transmission from the drone's embedded systems to ground control stations offers a more suitable alternative. RF solutions are optimized for long-range, low-latency communication, and their lightweight footprint and direct-to-ground architecture eliminate the need for intermediary gateways, aligning with the minimalistic and aerodynamic design of UAVs.

By tailoring the communication technologies—BLE for aircraft and RF for drones—the AVATAR project ensures optimal integration with minimal impact on aerodynamic performance, power budgets, and system weight, supporting scalable, real-time data storing, and mission-critical data exchange across platforms.

As outlined in Section NON-Hardware related solution4.3.3.2.3, the sampling frequency of the ADC channels needs to be reduced from 1056 Hz to 50 Hz in order to accommodate the current hardware constraints. Concurrently, efforts are underway to develop an alternative wired solution, which allows us to transmit the original full signal to the Edge device (and to the cloud). Further improvements allowing the full signal transmission should be pursued once the project advances to a higher Technology Readiness Level (TRL).

7 REFERENCES

1. Bekas, D.G., Z. Sharif-Khodaei, and M.F. Aliabadi, *An innovative diagnostic film for structural health monitoring of metallic and composite structures.* Sensors, 2018. **18**(7): p. 2084.
2. Salmanpour, M.S., Z. Sharif Khodaei, and M.H. Aliabadi, *Airborne transducer integrity under operational environment for structural health monitoring.* Sensors, 2016. **16**(12): p. 2110.
3. STM32, D.k.f.I.n., multi-channel communication with STM32L4," um2153, October 2019.