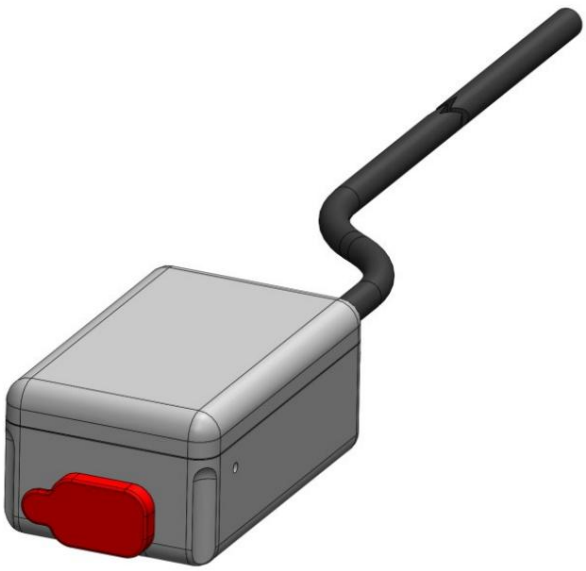
MICROLOG-01



Introduction

MICROLOG-01 is a CAN data logger designed for use in various fields (such as laboratory analysis and monitoring, machinery monitoring, precision agriculture, industrial monitoring, process analysis, automotive, motorsport, etc.) thanks to its data logging capability via CAN bus (specifically designed for sectors such as automotive and motorsport) and its ability to support CAN OBDII and J1939 protocols. The data logger is designed to acquire up to 16 signals.

CAN (which can be either open CAN, OBDII or J1939) simultaneously, with frequencies that can vary from 0.01Hz to 100Hz, making it an excellent data logging system.

CAN Bus Interface

The system integrates a CAN Bus capable of extracting up to 16 channels simultaneously.

- **Transceiver:** High-Speed CAN 2.0B.
- **Protocols:** Using the GUBELLINI DataStudio software it is possible to configure the CAN bus according to the OBD II, SAE J1939, ISOBUS (ISO 11783), OpenCAN (EN-50325-4) protocols.
- **Termination:** The 120 Ohm termination resistor is present internally in the datalogger.

Isolation and work environment

The datalogger guarantees an IPX7 insulation level, allowing use in harsh environments. The guaranteed operating temperature range is from -20°C to +85°C.

Internal memory

Logged data is saved to an unformatted micro SD card (RAW writing). Supported micro SD card formats are 16/32/64/128GB (larger sizes are supported but cannot be used to their full capacity). The micro SD card can be removed and connected directly to a PC to retrieve data (organized into datasets) using the dedicated GUBELLINI DataStudio function.

Auto-Start Conditions

The data logger can manage the start of recording in two modes: "Always On" and "Auto-start Condition". In "Always On" mode, the data logger generates a new dataset and begins recording the data of the active inputs immediately after startup. The dataset closes when the device is turned off. In "Auto-Start Condition" mode, the data logger generates a new dataset and begins recording the data of the active inputs immediately after startup.
Record data only after a condition occurs (linked to the value of one of the active inputs). The "auto-Start" condition is software-programmable. The dataset is closed and recording stops when the device is turned off.

Analysis software

The MICROLOG data logger comes with the free GUBELLINI DataStudio suite, the software that allows for complete device management. In addition to standard functions (such as device configuration, data downloading, and external module management), DataStudio includes the innovative AI Analyst (an analysis system incorporating artificial intelligence). This multi-agent system integrates with cutting-edge models such as Google Gemini and OpenAI to automate anomaly detection, perform complex mathematical transformations (such as FFTs or integrals), and generate professional technical reports in seconds.

Dataset comparison

The analytics suite offers high-performance visualizations through dynamic graphs that allow you to upload one or two data sets and compare them. The analytics software allows you to compare different laps, sessions, or devices.

Layout

The analysis suite offers the ability to save layouts with a specific choice of channels, upper and lower limits, colors, line thickness, etc., so they are always available. You can switch between layouts to perform different types of analysis. You can always display only the most significant graphs, depending on what you are looking for.

Virtual channels

The Virtual Channels module allows you to create new telemetry channels (e.g., velocity, spacing, power, filters) by mathematically processing the data recorded by the Datalogger. The heart of the system is a high-performance vector calculation engine (via C# scripting). You write the "recipe" (the script) for a single point in time, and the software will automatically execute it at full speed for all the hundreds of thousands of samples recorded in the dataset.

Artificial intelligence

The GUBELLINI DataStudio data logger management software is equipped with an AI module called **AI Analyst. AI Analyst is** an innovative way to analyze data recorded by the data logger.
It's not just a simple AI "chatbot" you can ask generic questions to, but a true **multi-agent artificial intelligence system** integrated into the software. It's designed to assist you in analyzing acquired data, automating the search for anomalies, the creation of complex graphs, and the preparation of professional reports.

Diet

The device can be powered by a DC voltage from 7 to 24V. The main connector provides a 5V supply voltage for the connected sensors.

USB-C connection

You can connect to the data logger via a USB-C connector located on the side of the enclosure and protected by a rubber gasket. The USB connection allows you to send and receive configuration data. To download the recorded data, you can remove the microSD card and connect it to your computer. Using the included software, you can download the datasets to your PC and export them in CSV format or analyze them.
directly via the included data analysis module

Main Connector: (Automotive Grade)

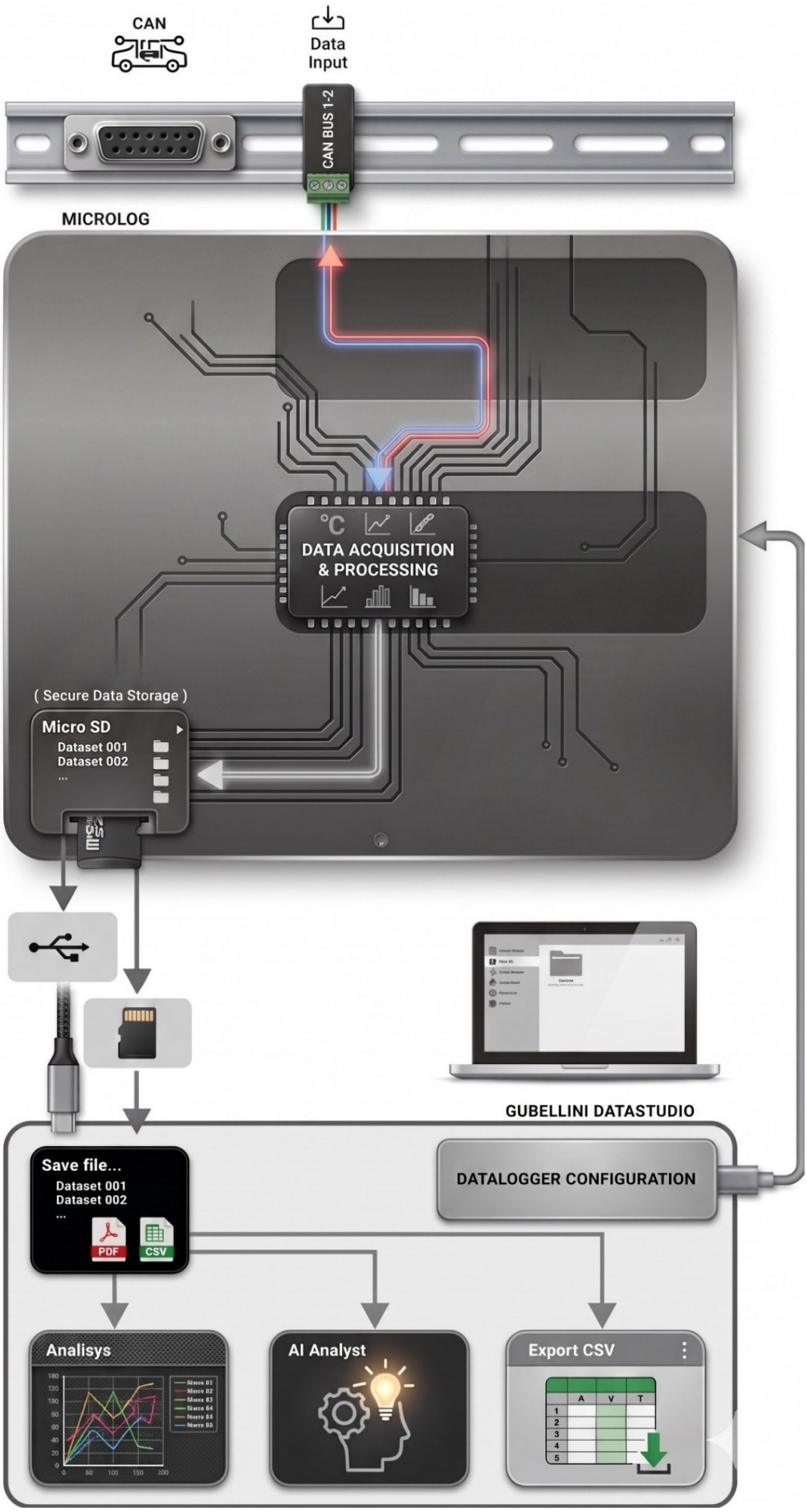
To ensure maximum reliability in signal transmission and maintain the IPX7 waterproof certification, the physical interface of the CAN bus is via a 4-pin AMP Super Seal 1.5 series connector (code 282106-1).

PART 1: Software

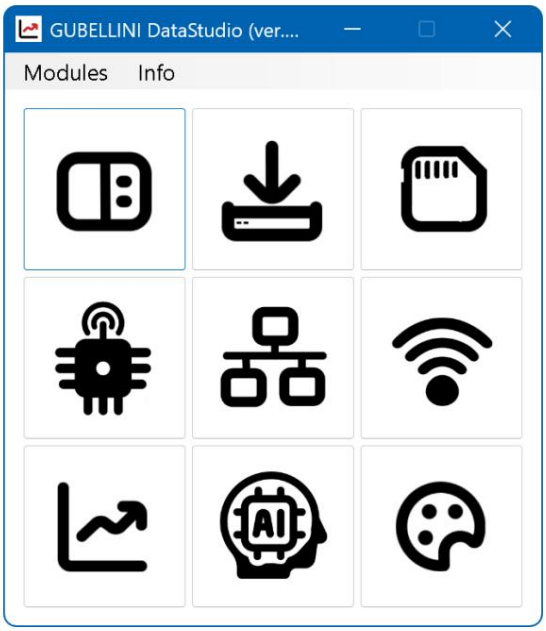
Chapter 1: Introduction and Main Interface

The MICROLOG data logger is managed entirely through the GUBELLINI DataStudio software. Using GUBELLINI DataStudio, you can configure the logger (setting the inputs, CAN bus, sampling rates, and more), download the recorded data (saved in sessions called "datasets"), and extract them.
Open datasets directly from the micro SD card to view graphs and statistics. The software also includes an "agentic" AI system that lets you leverage the power of artificial intelligence to perform analyses and compile reports.

Machine Translated by Google



1.1 The Home Screen (Hub)



When you launch the software, you'll be presented with the main screen, which acts as a true "command center" (Hub). From here, you can quickly access all the program's modules via 8 large buttons or the top drop-down menu.

Here's an overview of the tools available to you regarding MICROLOG:

Datalogger Management

- **MICROLOG** By clicking on this button you access the main management panel of the Datalogger. From here you can configure input parameters (activation, sampling rates, conversion formulas, etc.), configure the CAN bus, define when and how to start recording, monitor connection status, and view data in real time.

- **Channels:** Opens the channel configuration form for analysis. Here you can customize the names, units of measurement, display colors on graphs, maximum/minimum values and line thicknesses (of the 16 CAN inputs).

- **Analysis:** Opens the main data analysis suite (Time Charts) to view
  the progress of the recordings (the datasets) and examine the data through functions such as Scatter charts, histograms, FFT and more.

- **AI Analyst:** Launch the built-in virtual AI assistant powered by Artificial Intelligence. This advanced tool will help you interpret the data you acquire by providing insights and suggestions. Note: It requires a personal Gemini or OpenAI API key to function.

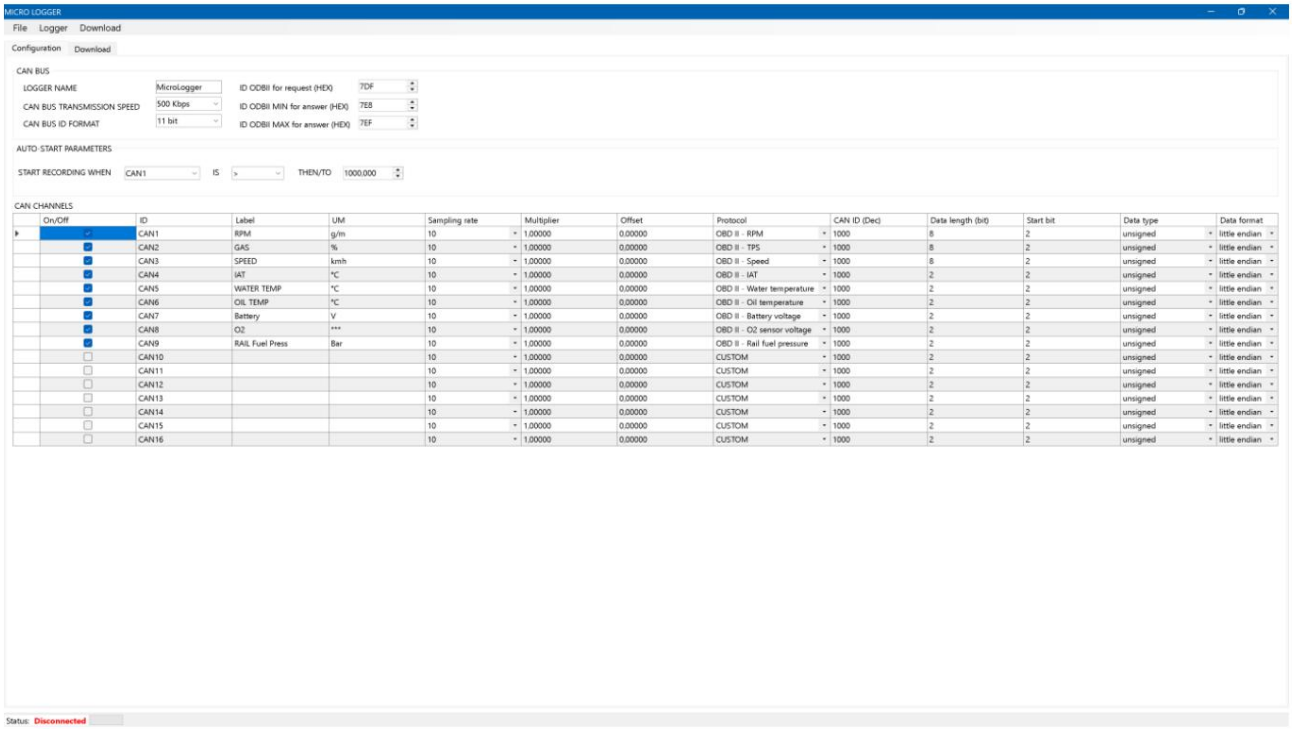
1.2 The Top Menu Bar

In addition to the buttons, there is a classic menu bar at the top of the window:

- **"Modules" menu:** Contains a drop-down list that exactly replicates the functions of the main buttons (Logger, Download, Micro SD, Channels, Analysis, AI Analyst, WiFi Monitor, CAN Bridge), giving you an alternative way to navigate between windows.
- **"Info" menu -> "Software version":** Clicking this item will display a summary window containing program information. You can view the software name, current version, author, copyright information, and the version of the operating system currently in use on your PC.
- **"Info" menu -> "Update Software":** Clicking this item will check the server for the latest version of the software. You can decide whether or not to update it.

Chapter 2: MICROLOG (Datalogger Configuration)

This section is accessible by clicking on the MICROLOG from the main screen, it is the heart of the management software. From here, you can configure all the datalogger parameters and download the data stored in the datalogger (grouped into datasets) directly from the microSD card. The screen is divided into two folders: **CONFIGURATION and DOWNLOAD.**



2.1 Connection and Status

At the bottom of the window, you can monitor the connection status. If the USB connection is successful, the status will change from ***DISCONNECTED*** (red) to ***CONNECTED.***
(green) and the COM port used will be shown.

2.2 CAN BUS Settings

In the **CAN BUS** box you can define the general communication parameters of the vehicle line:

- **CAN BUS TRANSMISSION SPEED:** Choose the network speed (e.g. 1 Mbps, 500 Kbps or 250 Kbps).
- **CAN BUS ID FORMAT:** Selects the standard (11 bit) or extended (29 bit) identifier. In the Under the CAN channel-specific settings below you can also set the protocol type (e.g. *CUSTOM, OBD II - RPM, OBD II - Speed,* etc.), the ID, the message length and the byte order (Little/Big Endian).
- **OBD-II parameters:** The default parameters are those normally used in the automotive sector. Do not change these values unless specifically needed.

2.3 LOGGER IDENTIFICATION

You can assign a name to each logger so that when you download the dataset, the file will automatically be named with that specific name. The data logger name will also remain saved in the data file.

2.4 Auto-Start Parameters

To avoid starting recording simply by turning it on, you can instruct the data logger to begin recording data only after a specific physical event occurs. In the dedicated box, you can set a logic such as:

- *START RECORDING WHEN* **[Channel]** *IS* **[Condition: >, <, =]** *THEN/ TO* **[Numeric value]** *(For example: Start when GPS speed or engine rpm exceeds a certain threshold).*

2.5 CAN Channels - Channel Configuration

The main screen of the Logger presents a table that allows you to configure in detail the behavior and acquisition logic of each signal you want to record via the CAN bus.

CAN channels

The CAN inputs are considered acquisition "channels." On the single MICROLOG CAN bus, you can "sniff," extract, and translate digital messages from the vehicle's data network (e.g., the ECU).

- **On/Off, Label, UM, Sampling rate, Multiplier, Offset:** They regulate the activation, name, unit of measurement, sampling and mathematical conversion of the data extracted from the CAN bus.
- **Protocol:** Allows you to choose the data extraction logic from the CAN bus. By selecting *CUSTOM,* you will have to manually fill in the subsequent entries (to intercept the OPENCan protocol); alternatively, you can choose pre-filled OBD-II channels (such as *OBD II - Speed, OBD II - RPM, OBD II - TPS,* etc.) or J1939 channels (used in agricultural machinery, heavy vehicles, and industry).

  By selecting one of the OBDII options, the datalogger will send a specific request on the CAN bus to obtain the relevant information in response (usually from the vehicle control unit).
  By selecting a J1939 channel instead, the logger will “sniff” all messages on the CAN bus and filter the messages using the PNG included in the 29bit ID.
- **CAN ID:** The unique identifier (in decimal format) of the CAN message you want to extract from the information.
- **Data length:** Sets the number of bits to consider when extracting data from the CAN packet (from 1 (32-bit)
- **Start bit:** Sets the starting point (byte number) within the CAN packet from which the system must start reading your data (0 to 63).
- **Data type:** Allows you to define the ***“unsigned”*** or ***“signed”*** format
- **Data format:** Allows you to define the “endianess” of the data: little endian or big endian

2.6 Save or Send Configuration

Once you have set the datalogger parameters according to your needs, you can save the configuration for subsequent software launches and send the configuration to the datalogger:

- **File -> Save configuration:** Save an .xml file on your PC.
- **Logger -> Send configuration to logger:** Physically transfers the newly set parameters to the internal memory of the connected datalogger.

2.6 Datalogger operating logic

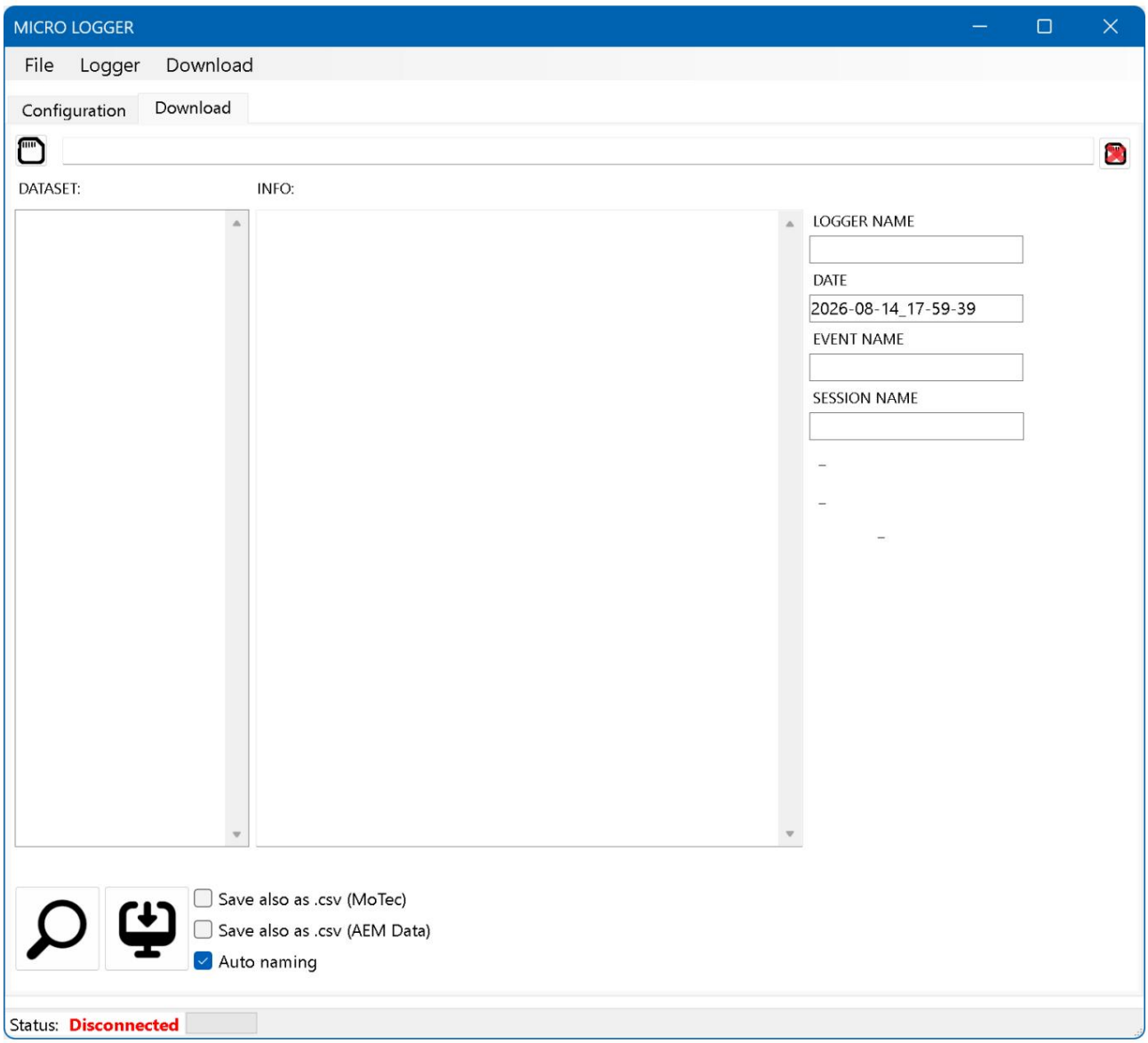
At each start-up, the datalogger reads the configuration metadata from the micro-SD memory (relating to the last programming sent via USB) and decides whether to start storing the input values from the first available moment or wait for the event defined by the **AUTO-START parameters.** When recording starts, the datalogger's LED will change its flashing pattern. When recording begins, a new dataset is defined (by assigning a unique serial number). The dataset will be closed when the datalogger is turned off.

If the AUTO-START condition is set to “Always-on”, a new dataset identified with a sequential number is generated at each power-on. If the AUTO-START condition is set to detect a physical event (e.g., analog input 1 pressure greater than 4 bar), the datalogger will wait to generate a new dataset. When the AUTO-START condition is met, a new dataset identified with a sequential number is generated and the values begin to be stored in the micro-SD memory. All datasets remain saved in the micro-SD.

present in the datalogger, even after switching off.

Chapter 3: Download - Downloading data from the SD card

By selecting the **Download folder,** you enter the interface for downloading the data stored in the device's internal memory.
Important: You must disconnect the Micro-SD card from the data logger and insert it directly into your PC's card reader.



3.1 Scanning and Viewing Datasets

1. **Disk Selection:** Click on the button next to the *SD PATH* line ( ). A window will open asking you to select the physical drive to which the memory card is connected. Micro-SD. MicroSD is usually the last line listed.
2. **Scan:** Using the appropriate button (Dataset list in the list), the software will read the Micro-SD populating a on the left of the window (instantly).
3. **Download:** Select the dataset you want to download to your PC and fill in the "Event Name" or "Event Name" fields. check the extraction in .csv. By clicking the download button (downloaded and), the selected dataset will be saved.

3.2 Nomenclature and Export

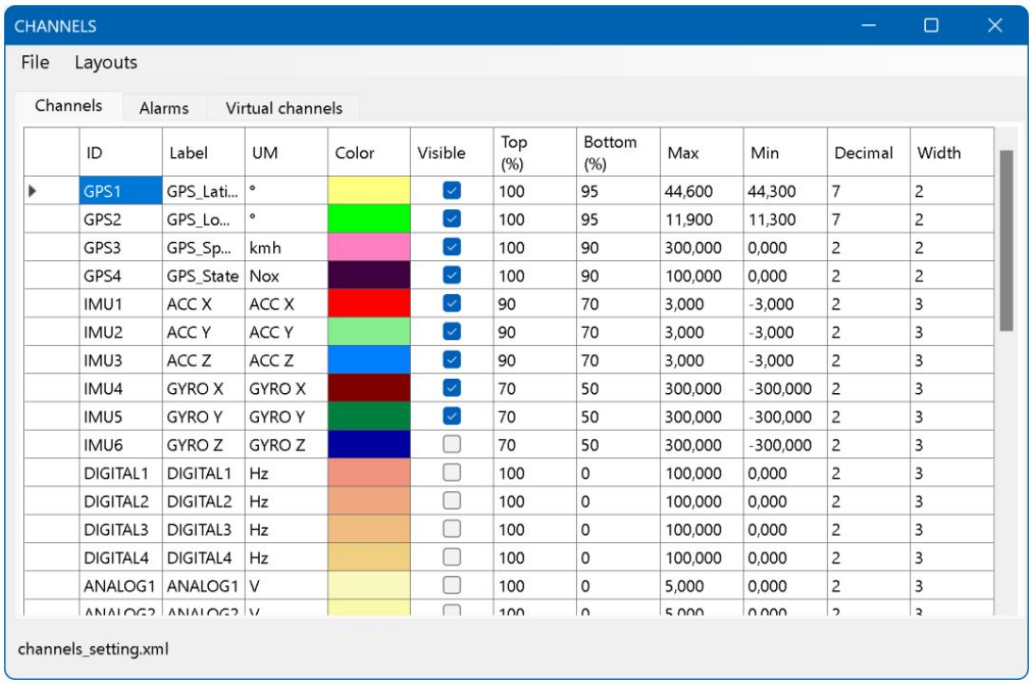
Below, you can decide how to name the exported files:

- **Auto naming:** If activated, the program will name the file by automatically combining Name Logger, Date, Event and Session.
- **Advanced Saving (CSV):** In addition to saving data in raw format (.dat file), you can check the **Save also as .csv (MoTec)** or .csv **(AEM Data)** options to directly export formats compatible with third-party analysis software.

Chapter 4: Channels (Graphical Channel Configuration)

The **Channels** module is the tool dedicated to visually preparing your data. Accessible both from the main menu (Hub) and directly from within the analysis screen, this panel allows you to decide *how* each individual acquired input should be plotted on the graph.

NOTE: The CHANNELS and GRAPHS modules are common to several devices and therefore include channels that are not present on all devices. In this case, MICROLOG only manages the 16 CAN inputs, while the GPS, IMU, DIGITAL, and ANALOG inputs are not supported. If you use MICROLOG, you can simply deactivate all these input groups (by checking the "Visible" column).



	ID	Label	UM	Color	Visible	Top (%)	Bottom (%)	Max	Min	Decimal	Width
▶	GPS1	GPS_Lati...	°		☑	100	95	44,600	44,300	7	2
	GPS2	GPS_Lo...	°		☑	100	95	11,900	11,300	7	2
	GPS3	GPS_Sp...	kmh		☑	100	90	300,000	0,000	2	2
	GPS4	GPS_State	Nox		☑	100	90	100,000	0,000	2	2
	IMU1	ACC X	ACC X		☑	90	70	3,000	-3,000	2	3
	IMU2	ACC Y	ACC Y		☑	90	70	3,000	-3,000	2	3
	IMU3	ACC Z	ACC Z		☑	90	70	3,000	-3,000	2	3
	IMU4	GYRO X	GYRO X		☑	70	50	300,000	-300,000	2	3
	IMU5	GYRO Y	GYRO Y		☑	70	50	300,000	-300,000	2	3
	IMU6	GYRO Z	GYRO Z		☐	70	50	300,000	-300,000	2	3
	DIGITAL1	DIGITAL1	Hz		☐	100	0	100,000	0,000	2	3
	DIGITAL2	DIGITAL2	Hz		☐	100	0	100,000	0,000	2	3
	DIGITAL3	DIGITAL3	Hz		☐	100	0	100,000	0,000	2	3
	DIGITAL4	DIGITAL4	Hz		☐	100	0	100,000	0,000	2	3
	ANALOG1	ANALOG1	V		☐	100	0	5,000	0,000	2	3

6.1 Configuration Parameters (The Table)

The interface appears as a large table. Each row represents a channel in your system.
Here's what each column means and how you can change it:

- **ID (Read-only):** The original hardware name of the channel (e.g. GPS1, ANALOG4). It cannot be modified.
- **Label:** The custom name you want to give to the sensor (e.g. "Lean Angle", "Brake Pressure"). This will be the name that will appear in the legend of the ANALYSIS screen (where the graphs relating to the datasets are displayed.
- **UM:** is the unit of measurement that will appear in the legend of the ANALYSIS screen (where the displayed graphs related to the datasets.
- **Color:** The color the line will be drawn in on the graph. **To change it, double-click the color cell:** a palette will open from which you can choose your preferred hue.
- **Visible:** A checkbox. If unchecked, the channel will be shown in the values table. but it will *not* be drawn in the chart area to avoid visual confusion.
- **Top (%) and Bottom (%):** These values (from 0 to 100) allow you to confine the graph of a sensor to a specific horizontal "portion" of the screen, avoiding that the lines

overlap chaotically. For example, setting Top to 100 and Bottom to 50 will draw the channel only in the top half of the screen.

- **Max and Min:** Represent the physical limits of the vertical axis (Y) when the graph is in plot mode. "Manual". For example, if it's a water temperature sensor, you might set Min = 0 and Max = 120. If the graph zoom (in the Graphs window) is set to “Auto” these values will be ignored.

- **Decimal:** Determines the number of decimal places with which the sensor value will be displayed in the table of values.

- **Width:** The thickness (in pixels) of the line drawn on the graph. Increase it to emphasize the most important channels.

6.1.1 Tabs Channels, Alarms, Virtual channels

In the Channels window, you'll find two other tabs: Alarms and Virtual Channels. The first contains the configuration of alarms that MICROLOG doesn't manage. Ignore this folder. The second, however, contains all the virtual math channels you have created using the Graphs module (the analysis module that allows you to analyze data recorded via CAN).

6.2 Managing Configuration Files

Using the menu above, you can:

- **Save As GLOBAL:** Saves the current changes to the channels_setting.xml file. This file is the configuration file that will be automatically opened when you open a dataset in the Graphs window.

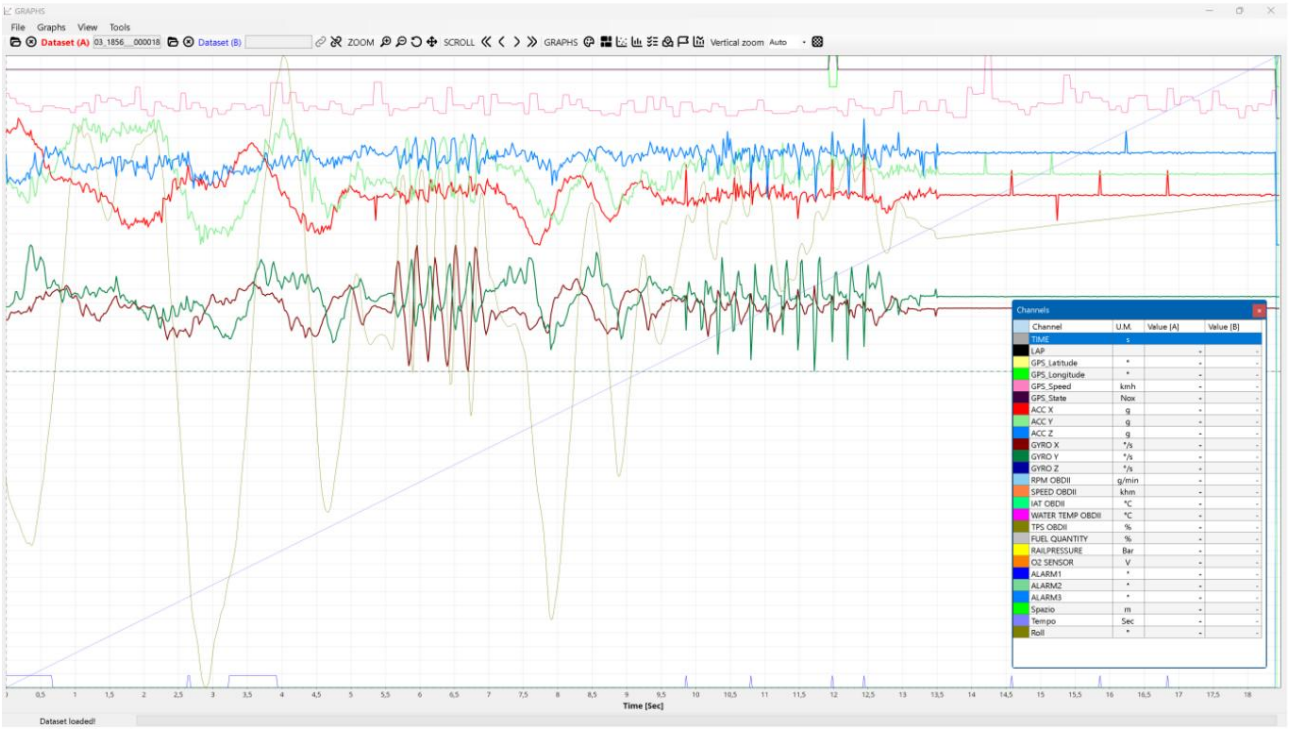
- **Save Layout:** Create a new style file (template) to be reused in the future for other datasets (in the Layouts folder).

- **Open Layout:** Allows you to open a previously saved “Layout” file.

Layouts allow you to quickly switch between display styles without having to re-set all the signal parameters. You can save a layout file with only the signals from the IMU, only the engine signals, or with specific virtual channels. Each layout file includes the list of channels to display, the color, the max and min range, and all the parameters you can modify via the **Channels window.**

Chapter 7: Analysis (Data Visualization and Analysis)

By clicking the **Analysis** button , you'll access the software's analytical core. When the window opens, the graph area will be empty and will only populate if a dataset is opened/loaded.



7.1 Loading a Dataset

To view recorded data:

1. Click on the folder icon in the top left of the toolbar (or go to **File -> Open dataset).**
2. Select the .dat file downloaded from the Datalogger.
3. The system will decode the binary file, extract the sample rates and display the entire trace on the screen, populating the side table with the list of active channels.
4. The first dataset will be loaded as dataset A. You can also load a second dataset (dataset B) to compare the values and trends of the various signals

7.2 Chart Navigation (Zoom and Pan)

The top toolbar offers advanced controls for moving fluidly along the time (X) axis:

- **Zoom + and Zoom -:** Enlarge or reduce the area of the graph centered on the screen horizontally.
- **Zoom Undo:** Cancels the last zoom level applied, allowing you to go back to the steps previous (the software stores the history of your movements!).
- **Zoom Max:** Restores the global view, showing the acquisition from second 0 up to in the end.

- **Scroll Arrows (Pan):** The <<, <, >, >> buttons allow you to scroll the graph to the right or left (forwards and backwards in time) in small (20% of the view) or large (80% of the view) jumps.

7.3 "Vertical Zoom" Mode

In the drop-down menu at the top right, you will find the **Vertical zoom** item which adjusts the behavior of the Y-axis (the height of the lines):

- **Auto:** The software automatically scales each curve so that its recorded minimum and maximum values perfectly fill the screen. Ideal for quick visual inspection.
- **Manual:** The graph strictly adheres to the *Min* and *Max* limits you set in the *Channels submenu.* Ideal for comparing different graphs while maintaining the same magnitude scale.

7.4 The Interactive Cursor and the Data Table

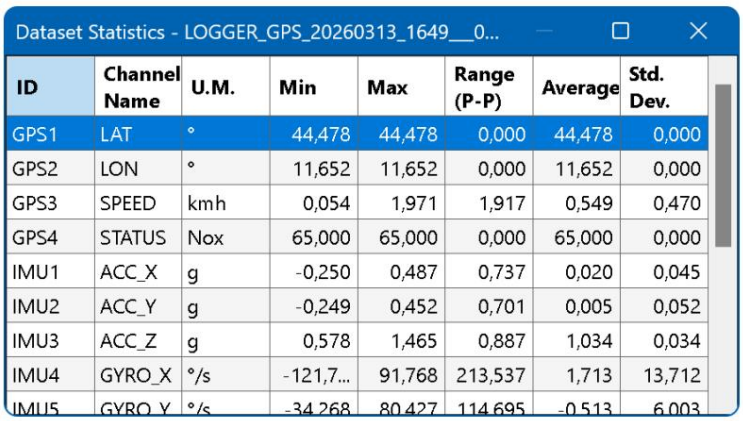
By clicking the mouse on the graph area, you'll notice a **dotted crosshair** that follows your movements. On the right of the screen is a data grid **(Channels):**

- When you click the mouse, **the grid on the right updates instantly** showing the value recorded by each individual sensor at that specific moment. Alarms and virtual channels are also displayed. The visible channels are those activated from the Channels window.
- By clicking on a row of the values table (to the right of the chart area), the axis
  The selected input will be highlighted, displaying its graduated scale on the left side of the screen (the same color as the line). The selected graph line will be doubled in thickness (so it stands out from the others).

7.5 Advanced Analysis Tools

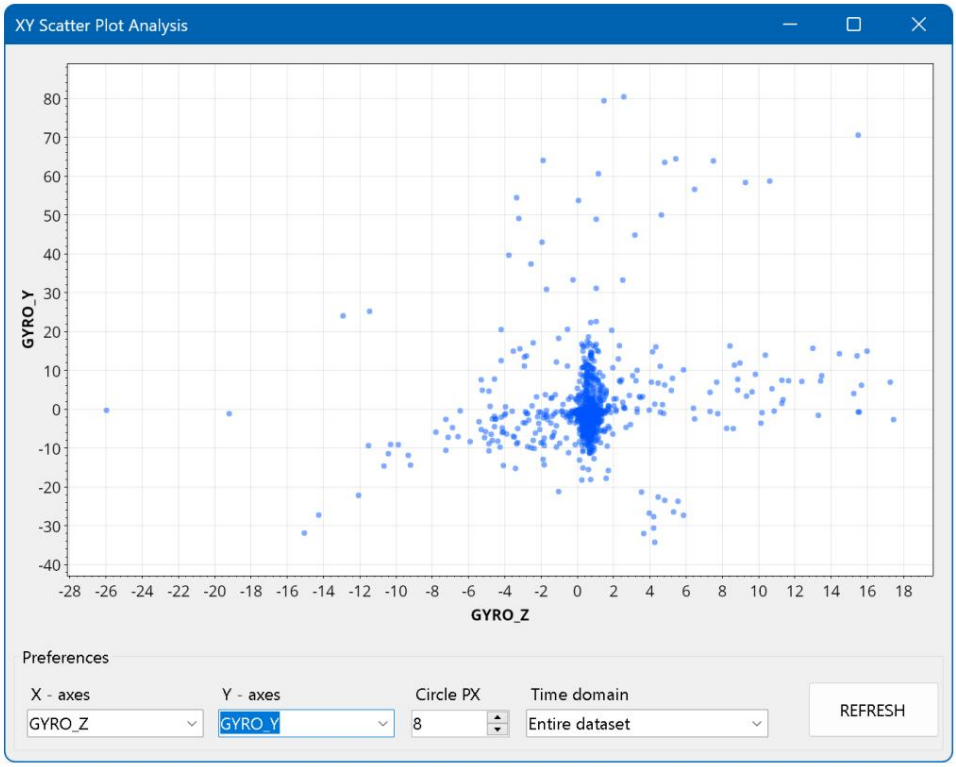
From the toolbar (or the **Graphs menu)** you can launch additional analysis modules based on the currently open dataset:

- **Statistics** [Calculator/Statistics Icon]: Generates a summary table that calculates
  Instantly *Minimum Value, Maximum Value, Range, Average* and *Standard Deviation* for each active channel in the entire session.

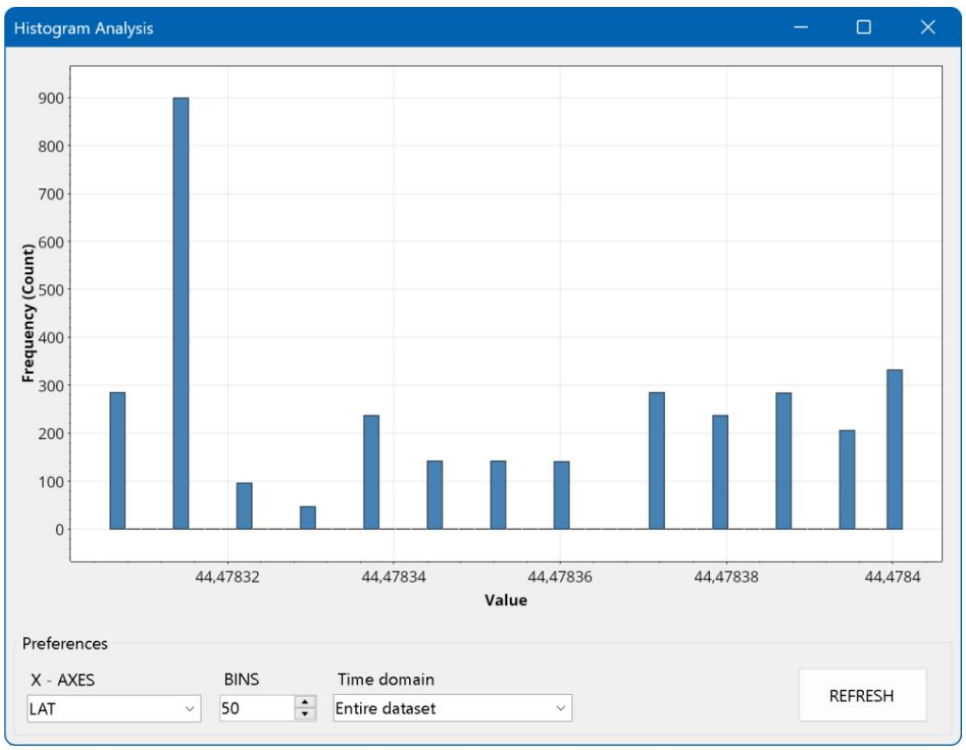


ID	Channel Name	U.M.	Min	Max	Range (P-P)	Average	Std. Dev.
GPS1	LAT	°	44,478	44,478	0,000	44,478	0,000
GPS2	LON	°	11,652	11,652	0,000	11,652	0,000
GPS3	SPEED	kmh	0,054	1,971	1,917	0,549	0,470
GPS4	STATUS	Nox	65,000	65,000	0,000	65,000	0,000
IMU1	ACC_X	g	-0,250	0,487	0,737	0,020	0,045
IMU2	ACC_Y	g	-0,249	0,452	0,701	0,005	0,052
IMU3	ACC_Z	g	0,578	1,465	0,887	1,034	0,034
IMU4	GYRO_X	°/s	-121,7...	91,768	213,537	1,713	13,712
IMU5	GYRO_Y	°/s	-34,268	80,427	114,695	-0,513	6,003

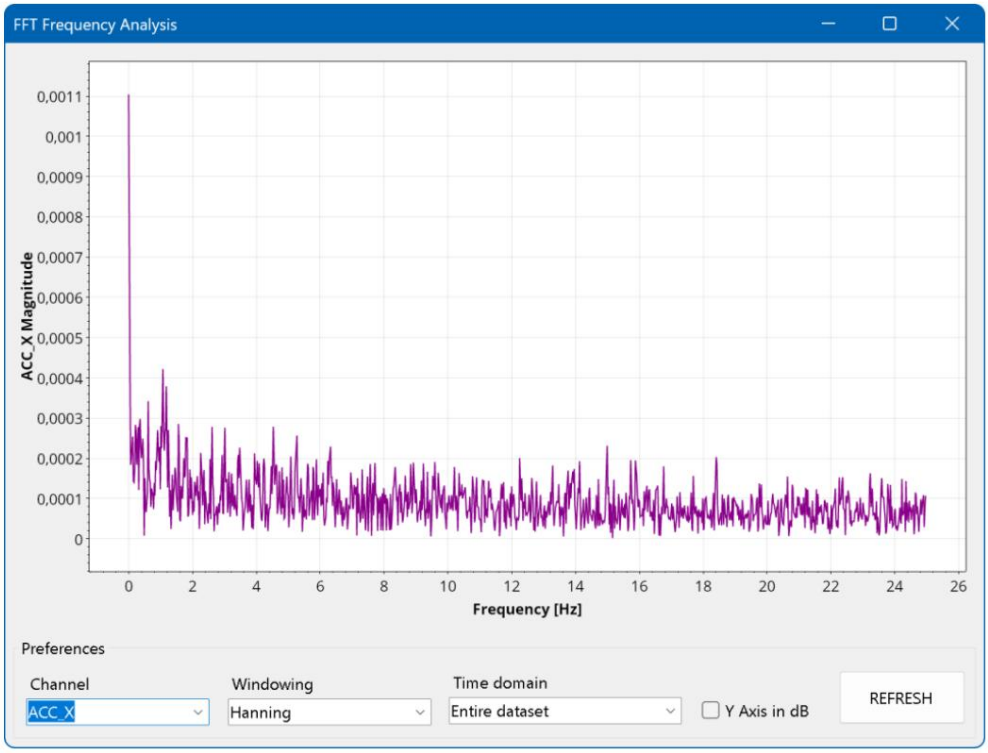
• **Scatter XY:** Opens the module to plot one channel as a function of another (e.g. temperature and pressure).



• **Histogram:** Opens the histogram module to understand how long (statistical frequency) a sensor remained within a certain range of values (e.g. temperature map).



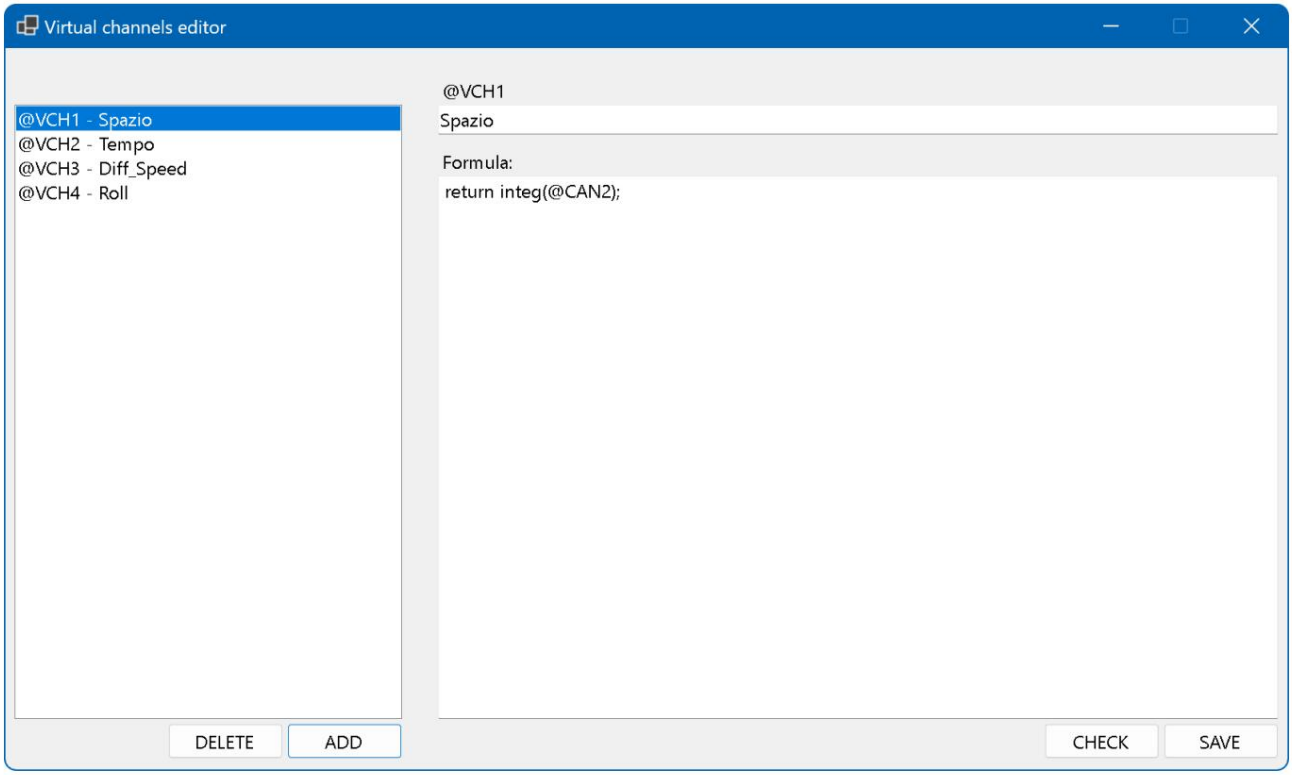
• **FFT (Fast Fourier Transform):** Starts the spectrum analyzer for frequency analysis and vibrations (e.g. chattering or engine speed analysis).



•

7.6 Virtual Channels

Via the menu in other Tools > Virtual channels editor, you can open the **Virtual Channels Editor** module
To generate up to 200 virtual channels. The **Virtual Channels Editor** module allows you to create new telemetry channels (e.g., speed, spacing, power, filters) by mathematically processing the data recorded by the datalogger. The heart of the system is a high-performance vector calculation engine. You write the "recipe" (the script) for a single instant in time, and the software will automatically execute it at maximum speed for all the hundreds of thousands of samples recorded in the dataset.



7.6.1 The Graphical Interface (Virtual Channels Editor)

The editor window is divided into two main sections: on the left is the list management, on the right is the formula writing area.

- **ADD button:** Creates a new virtual channel at the bottom of the list. It will be generated automatically a unique ID (e.g. @VCH1).
- **DELETE button:** Removes the currently selected virtual channel from the list (it will be confirmation required).
- **Channel List:** Displays a list of all virtual channels. **Important:** The order in this list is the order of execution. If the @VCH2 channel uses the @VCH1 channel in its formula, the @VCH1 channel must be higher in the list to be calculated first.
- **"Name" box:** Allows you to assign a readable name to the channel (e.g. "Path Space" or "RPM Filter").
- **"Formula" box:** The text area where you type the mathematical script.

- **CHECK button:** Performs instant code validation. Checks for the presence of typos, missing parentheses, or non-existent variables *before* performing the actual calculation. Always use it before saving.
- **SAVE button:** Permanently saves the entire configuration to a file on your computer, so you can find it the next time you start the program.

7.6.2: Advanced Manual: Scripting and Mathematics of Virtual Channels

This document comprehensively defines the syntax, data types, and all the mathematical, algebraic, and logical functions available within the Virtual Channels Editor. The underlying language used is **C# (C-Sharp).** Each expression is calculated instant by instant for each sample in the dataset.

1. The Semicolon (;)

In C#, **every single statement must end with a semicolon.** It tells the engine where one command ends and the next begins.

- *Incorrect:* float x = 10
- *Correct:* float x = 10;

2. Case-Sensitive (Uppercase and Lowercase)

The language strictly distinguishes between uppercase and lowercase letters.

- Writing math.sin, MATH.SIN or Math.Sin produces different results (only the third is correct).
- A variable called velocity is different from Velocity.

3. The mandatory "return" clause

The script must always conclude by telling the program **which number to save in the graph.** The return keyword is used.

C#

return @GPS1 * 2;

7.6.3: Data References (@ Tags)

To insert the sensor values relating to that specific moment in time into the formula, use tags preceded by the @ symbol.

- **Absolute Time:** @TIME (returns the elapsed time in seconds, e.g. 12.045).
- **Original Datalogger Channels:**
  - GPS (1-4): @GPS1, @GPS2, @GPS3, @GPS4
  - or IMU (1-6): @IMU1 ... @IMU6
  - or DIGITAL (1-4): @DIGITAL1 ... @DIGITAL4

or ANALOG (1-8): @ANALOG1 ... @ANALOG8

or CAN (1-16): @CAN1 ... @CAN16

- **Virtual Channels (already calculated):** @VCH1, @VCH2, @VCH3...

Example: Adding the analog accelerator to the brake on the CAN:

C#

return @ANALOG1 + @CAN5;

7.6.4: Data Types and Variables

Before using a value within the temporary script, it is useful to declare it by specifying its data type.

- **float (Single-precision decimal):** The standard format for telemetry. When you write a fixed number, add an f at the end to indicate it's a float.
  - *Example:* float radius = 12.5f;
- **int (Integer):** Whole numbers without commas. Useful for counters or loops.
  - *Example:* int samples = 100;
- **bool (Boolean):** Can only assume the values TRUE or FALSE. Useful for flags of state.
  - *Example:* bool inBraking = true;
- **double (Double precision decimal):** Used for astronomical or very high precision calculations precision. Usually, float is sufficient.
  - *Example:* double constant = 0.123456789;

7.6.5: Algebraic and Logical Operators

Operators allow you to combine signals (recalled with the @ symbol, e.g. @GPS1).

Mathematical Operators

- **Addition (+):** return @ANALOG1 + 5.0f;
- **Subtraction (-):** return @GPS3 - @GPS2;
- **Multiplication (*):** return @CAN1 * 3.6f;
- **Division (/):** return @IMU1 / 9.81f;
- **Modulus (%):** Returns the remainder of a division (useful for continuous loops).
  - *Example:* return @TIME % 60.0f; (Resets the time every 60 seconds.)

Logical and Relational Operators (Comparison)

They are used within conditions (If/While).

- **Same as (==):** @DIGITAL1 == 1.0f
- **Different from (!=):** @CAN1 != 0.0f
- **Greater / Less (>, <):** @GPS1 > 100.0f
- **Greater than or equal to / Less than or equal to (>=, <=):** @IMU2 <= 0.5f
- **AND (Logical AND - &&):** True if *both* conditions are true.
    - or *Example:* if (@GPS1 > 50.0f && @DIGITAL1 == 1.0f)
- **OR (Logical OR - ||):** True if *at least one* of the conditions is true.
    - o *Example:* if (@CAN1 > 100.0f || @CAN2 > 100.0f)

7.6.6: Conditional Structures

They allow you to perform different calculations based on the state of the data.

1. The If ... Else block If ... Else

Executes a cascading block of code.

C#

```
if (@GPS1 > 200.0f) {
    return 3.0f; // Very fast vehicle
}
else if (@GPS1 > 100.0f) {
    return 2.0f; // Moderate vehicle
}
else {
    return 1.0f; // Vehicle slow or stopped
}
```

2. The Ternary Operator (? :)

An If-Else condensed to a single line. Syntax: (Condition) ? ResultIfTrue : ResultIfFalse;

C#

```
// If CAN1 is negative, set it to 0, otherwise leave the value unchanged
return (@CAN1 < 0.0f) ? 0.0f : @CAN1;
```

3. The switch block

Useful when a variable (for example the state of a control unit) can take on many well-defined integer values.

C#

```
int mappaMotore = (int)@CAN5;
switch (mapMotor) {
    case 1:
        return @ANALOG1 * 1.0f; // Rain Map
    case 2:
        return @ANALOG1 * 1.5f; // Sports Map
    default:
        return 0.0f; // Unrecognized case
}
```

7.6.7: Iterative Loops

They are used to repeat complex operations at the same instant in time.

1. For loop

Repeats an operation a predefined number of times. Requires a starting index, an end condition, and an increment (i++).

C#

```
float sumSeries = 0.0f;
for (int i = 1; i <= 5; i++) {
    sumSeries = sumSeries + (@CAN1 / i);
}
return sumSeries;
```

2. While loop

Repeats the block *as long as* the condition remains true. It is evaluated at the beginning of the loop.

C#

```
float threshold = @GPS1;
int iterations = 0;
while (threshold > 10.0f) {
    threshold = threshold - 5.0f;
    iterations++;
}
return iterations;
```

3. Do ... while loop

Similar to the while statement, but guarantees that the code will execute *at least once,* because the condition is evaluated at the end.

C#

```
float x = @ANALOG1;
do {
   x = x * 0.9f; // Reduce by 10%
} while (x > 100.0f);
return x;
```

7.6.8: Standard Mathematical Functions

Callable by prefixing Math. (make sure to use capital letters).

- **Absolute Value (Math.Abs):** Removes the negative sign.
  - or *Example:* return Math.Abs(@IMU1);
- **Standard Rounding (Math.Round):** Rounds to the nearest value.
  - o *Example:* return Math.Round(@GPS1, 2); *(Round to 2 decimal places).*
- **Round Up (Math.Ceiling):** Rounds up to the next integer.
  - o *Example:* return Math.Ceiling(4.2f); *(Returns 5).*
- **Round Down (Math.Floor):** Rounds down to the nearest integer.
  - o *Example:* return Math.Floor(4.8f); *(Returns 4).*
- **Truncate (Math.Truncate):** Removes the decimal part without rounding.
  - o *Example:* return Math.Truncate(-5.9f); *(Returns -5).*
- **Exponential (Math.Exp):** Calculates e^x.
  - o *Example:* return Math.Exp(@CAN1);
- **Natural Logarithm (Math.Log):** Calculates the logarithm to base e.
  - o *Example:* return Math.Log(@ANALOG3);
- **Logarithm base 10 (Math.Log10):**
  - or *Example:* return Math.Log10(@ANALOG3);
- **Power (Math.Pow):** Raises a number to a specific exponent.
  - o *Example:* return Math.Pow(@GPS1, 2.0f); *(Calculate the square of GPS1).*
- **Square Root (Math.Sqrt):**
  - o *Example:* return Math.Sqrt(@GPS2);
- **Sign (Math.Sign):** Returns 1 if positive, -1 if negative, 0 if zero.

or *Example:* return Math.Sign(@IMU3);

7.6.9: Trigonometric Functions

All trigonometric functions require angles expressed in **radians.** To convert degrees to radians, multiply by (Math.PI / 180).

- **Constant Pi:** Math.PI *(It is approximately 3.14159).*
- **Sine (Math.Sin):** return Math.Sin(@IMU1 * Math.PI / 180.0f);
- **Cosine (Math.Cos):** return Math.Cos(@IMU1);
- **Tangent (Math.Tan):** return Math.Tan(@IMU1);
- **Arcsine (Math.Asin):** Returns the angle whose sine is the specified value. return Math.Asin(@CAN1);
- **Arcosine (Math.Acos):** return Math.Acos(@CAN1);
- **Arctangent (Math.Atan):** return Math.Atan(@CAN1);
- **Arctangent with 2 arguments (Math.Atan2):** Calculates the angle based on Y and X coordinates, useful for calculating trajectories. return Math.Atan2(@GPS_Y, @GPS_X);
- **Hyperbolic Sine / Cosine / Tangent:** Math.Sinh(x), Math.Cosh(x), Math.Tanh(x).

7.6.10: Statistical Functions (Max, Min, Avg)

- **Maximum Value (Math.Max):** Returns the greater of two numbers.

  or *Example:* return Math.Max(@GPS1, @GPS2);

- **Minimum Value (Math.Min):** Returns the smaller of two numbers.

  o *Example:* return Math.Min(@ANALOG1, 5.0f); *(Keeps the signal below 5V).*

- **Simple Average (Avg):** It is calculated using algebraic operators.

  o *Example:* return (@IMU1 + @IMU2 + @IMU3) / 3.0f;

7.6.11: Telemetry Analysis and Signal Filters

These are special functions designed to process arrays of data in the time domain, analyzing how the signal behaves compared to previous samples.

Basic Dynamic Functions

- **Integral (Space from Time/Velocity):** Accumulates the area of the signal taking into account the sampling rate.

  o *Syntax:* integral(value)

  o *Example:* return integral(@GPS3);

- **Derivative (Acceleration from Velocity):** Calculates the rate of change with respect to the sample previous.
    - o *Syntax:* derivative(value)
    - o *Example:* return derivative(@GPS3);

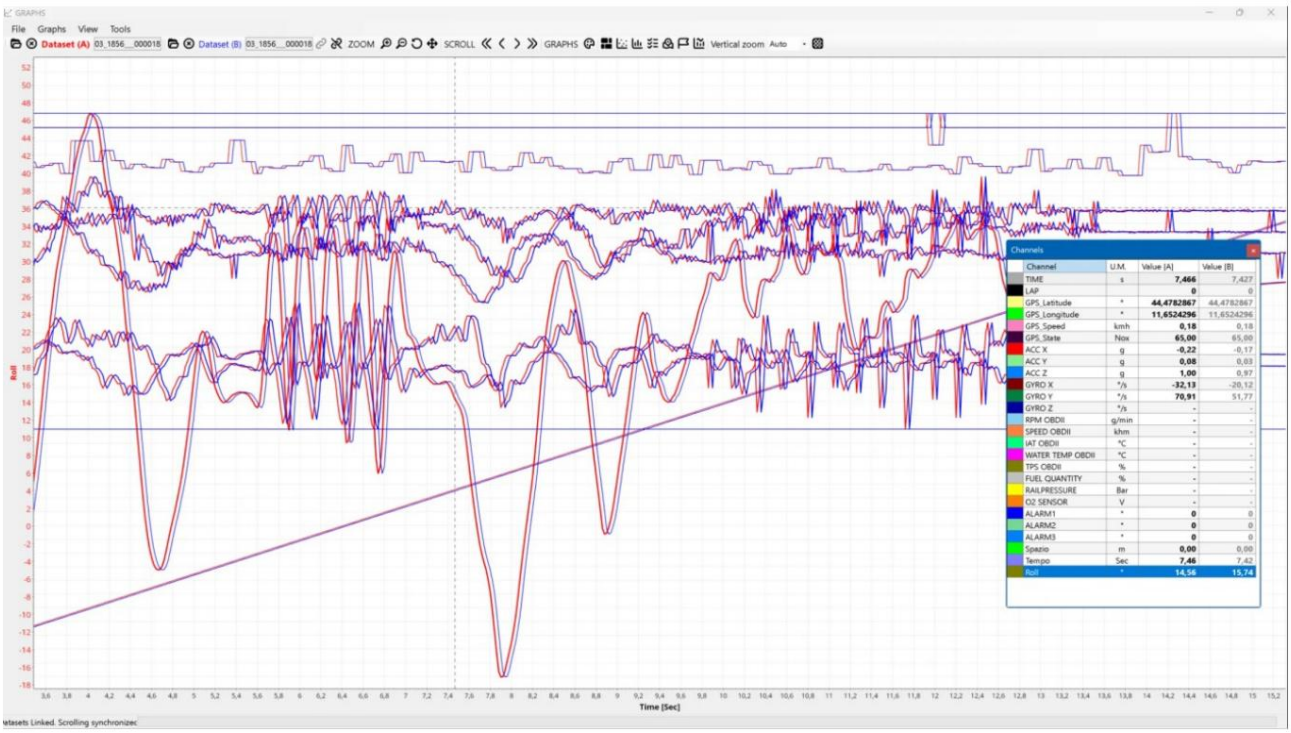
Filters (Signal Conditioning)

They allow you to clean up "noisy" signals (such as accelerometers or linear potentiometers).

- **Moving Average:** Smooths the signal by averaging the last N samples. Introduces a slight delay.
    - o *Syntax:* MovingAverage(value, number_of_samples)
    - o *Example:* return MovingAverage(@IMU1, 10); *(Average over the last 10 samples).*
- **Low Pass Filter (LP - Low Pass):** Cuts high frequencies (vibrations) and lets the low frequencies (real movements).
    - o *Syntax:* LPFilter(value, cutoff_frequency_Hz)
    - o *Example:* return FilterLP(@ANALOG3, 5.0f); *(Filters out anything that oscillates at more than 5 Hz.)*
- **High Pass Filter (HP - High Pass):** Cuts continuous or slow frequencies and displays only the rapid changes.
    - o *Syntax:* FilterHP(value, cutoff_frequency_Hz)
    - o *Example:* return FilterHP(@IMU2, 2.0f);
- **Band Pass Filter (BP):** Allows only a specific window of frequencies to pass.
    - o *Syntax:* FilterBP(value, frequency_min_Hz, frequency_max_Hz)
    - o *Example:* return FiltroBP(@CAN5, 10.0f, 20.0f);
- **IIR (Infinite Impulse Response) filter:** Autoregressive filter based on a damping coefficient (from 0.0 to 1.0). The closer it is to 1, the greater the damping.
    - o *Syntax:* IIRFilter(value, damping_factor)
    - o *Example:* return IIRFilter(@GPS1, 0.85f);
- **FIR (Finite Impulse Response) filter:** Filter calculated on the basis of a fixed window without feedback, excellent for phase stability.
    - o *Syntax:* Since it requires a custom weight array, in telemetry it is Usually approximated by weighted combinations of time-based Moving Averages or explicitly stated. In this module, using MovingAverage represents the purest and most applied form of rectangular FIR filtering.

7.7 Dataset comparison

You can upload up to 2 datasets: dataset A and dataset B. When you upload the second dataset (B), the graphs on the screen will automatically all be colored in two colors: one color for each dataset.

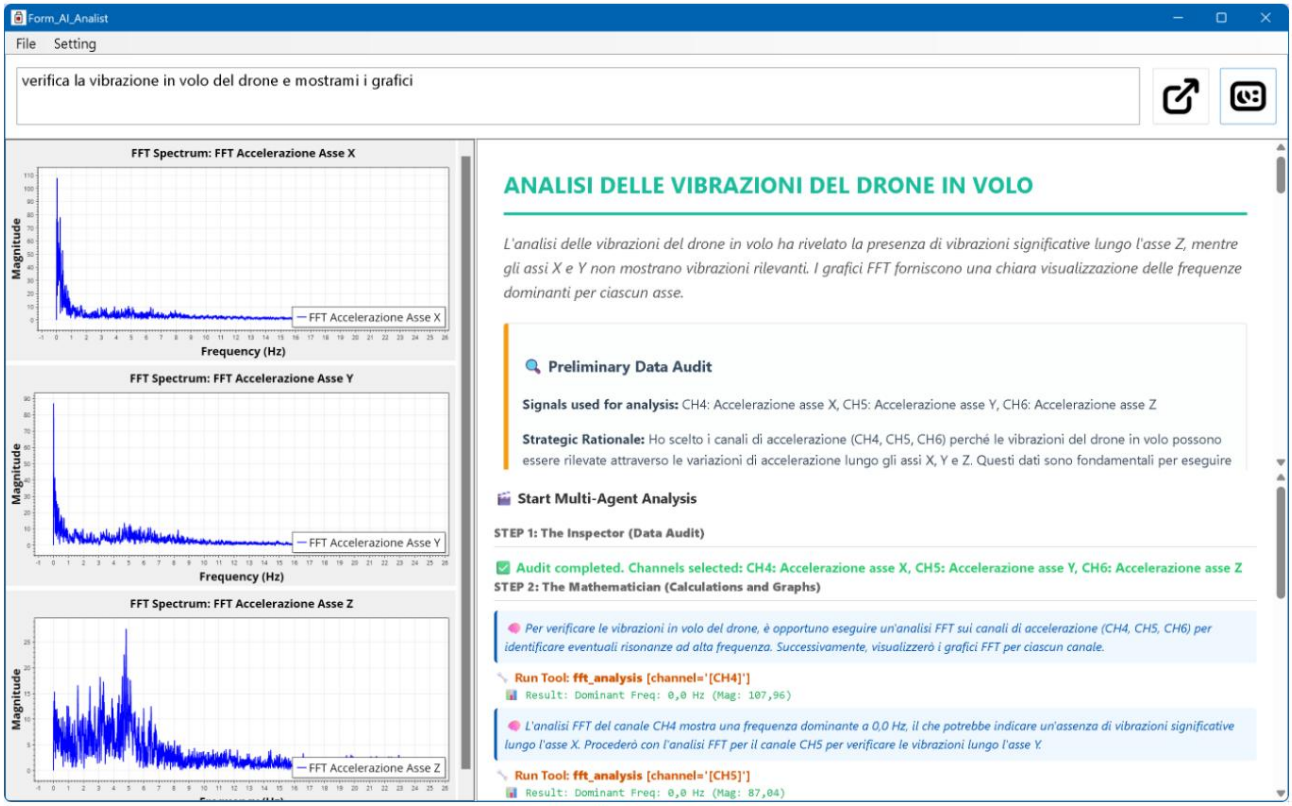


In the channel window, two columns are displayed: Value A and Value B, where the values pointed by the mouse in the graph are indicated.

Using the “link” and “unlink” buttons ( ) you can connect and disconnect the two datasets so as to translate dataset B onto dataset A. For example, if you want to synchronize an event that happened at two different times, you can press unlink, scroll through dataset B while leaving dataset A still and finally press link to reconnect the two datasets together.

Chapter 8: AI Analyst

AI Analyst is an innovative way to analyze data recorded by Datalogger. In the AI Analyst window, you'll have a field for entering queries, a menu for uploading a dataset, and an area displaying the AI's reasoning, results, and graphs. **AI Analyst** isn't just a chatbot for asking generic questions, but a true **multi-agent artificial intelligence system** integrated into the software. It's designed to assist you in data analysis.
telemetrics, automating the search for anomalies, the creation of complex graphs and the drafting of professional reports.



1. What does an AI Analyst do?

When you enter a request (e.g., "Analyze vehicle cornering stability"), AI Analyst takes over the entire workflow typical of a data engineer:

- **Select sensors:** It understands autonomously which logger channels are needed for answer your question.
- **Apply mathematics:** Use an advanced calculation engine to perform complex operations (Filters, Fast Fourier Transform (FFT), Integrals, Derivatives, Statistics).
- **Generate graphs:** Create dedicated visualizations (timelines, frequency spectra, histograms, XY scatterplots) to prove his points.
- **Write a Report:** Layout the results in a readable document, inserting the formulas used, the generated graphs and the technical conclusions.

2. How it works (Behind the scenes)

To ensure maximum precision and minimize errors, the system divides the work into three sequential phases (visible in real time in the Debug panel):

1. **The Inspector (Data Audit):** Checks the metadata of the uploaded dataset. It verifies which signals are available, discards unnecessary ones, and alerts you if any sensors that would have made the analysis more accurate are missing.

2. **The Mathematician (Processing):** Query real-world numerical data. Use mathematical tools to find peaks, calculate correlations, and draw graphs in the sidebar.

3. **The Editor (Drafting):** Collects the audit results, the mathematical results, and the graph images to draft the final document, writing it **exactly in the language you used** to submit the application.

3. How to make the most of it (Best Practices)

Artificial intelligence is powerful, but the best results are achieved by providing the right context. Here are the golden rules for achieving perfect analyses:

- **Configure Context (Essential):** Before using the AI, use the Settings > AI Configuration (Form_AI_Contest). Give your channels clear names (e.g., "Acc Y" instead of "CH5") and add short descriptions (e.g., "Accelerometer located on the left arm"). The more information you provide, the better the AI will be at contextualizing the data.
- **Be specific in your requests:** Avoid overly vague questions like "What's wrong?" Prefer direct prompts like: "Calculate the roll angle using the accelerometers and gyroscope. Look for any abnormal peaks above 2g."
- **Use your language:** You don't have to write in English. If you write in Italian, the AI will lead the whole reasoning and will write the report in Italian.

4. How to use Reports and Alarms

At the end of each run, the AI Analyst will present you with a hybrid HTML document in the center of the screen.

- **Export to PDF:** Use the "Save Report" button to export the document to PDF. The generated graphs will automatically be pasted into the document. It's the perfect tool for quickly sharing analyses with colleagues, pilots, or clients.

- **Automation (Alarm Suggestions):** At the end of each report, the AI will provide you with a section "Automation Recommendations." Here, it will suggest mathematical thresholds (e.g., If GyroZ > 150 deg/s for 0.5s). Use these suggestions to configure alarms in your data logger, which will automate problem detection for future logs without having to request AI intervention again.

To make this "virtual engineer" work at its best, some initial configuration is required.

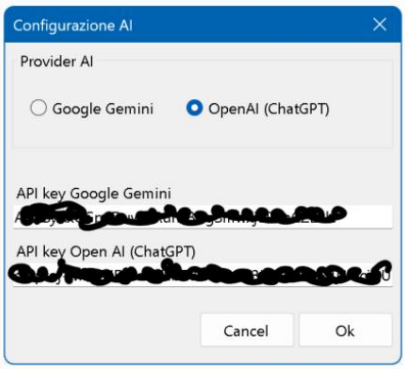
8.1 Initial Configuration and API Keys (Settings)

AI Analyst relies on top-notch generative AI models (such as **Google Gemini** or **OpenAI ChatGPT).** To use them, you must provide the software with your own access key (API Key).

1. From the AI Analyst module menu, open the **Settings item.**

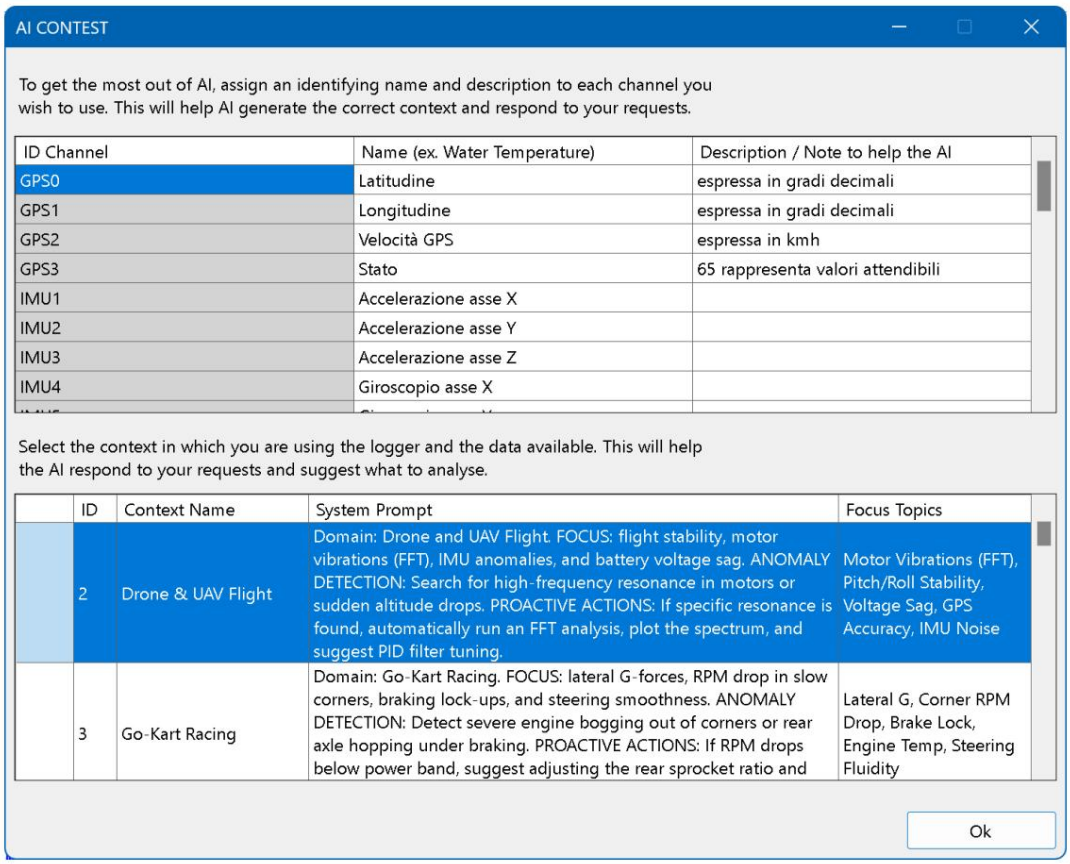
2. Choose which "brain" you want to use (Gemini is often recommended for its speed of reasoning, but you can opt for OpenAI).

3. Enter your **API Key.**



How to Obtain an API Key and Costs: API keys are generated by registering on the respective vendors' developer portals (for example, searching for *Google AI Studio* for Gemini or *OpenAI Developer Platform* for ChatGPT). Since these companies' web interfaces change frequently, simply follow their official online guides to generate the key. *Note on costs:* Using these APIs via the Datalogger incurs costs based on the amount of text processed. However, for typical telemetry analysis use, these costs are negligible (often fractions of a cent per analysis), and many vendors offer generous free monthly credits.

8.2 Setting the Stage: Channel Naming, Contexts, and URL Links



AI is incredibly powerful, but it needs to know *what* it's looking at. Telling the AI to "analyze CAN1" won't yield great results. If you tell it that CAN1 is "Brake Pressure," the AI will understand exactly what to look for.

NOTE: in this window there are also different groups of “channels”: digital, GPS, IMU etc.
MICROLOG manages CAN channels only. Enter descriptions only for these channels and leave the others blank.

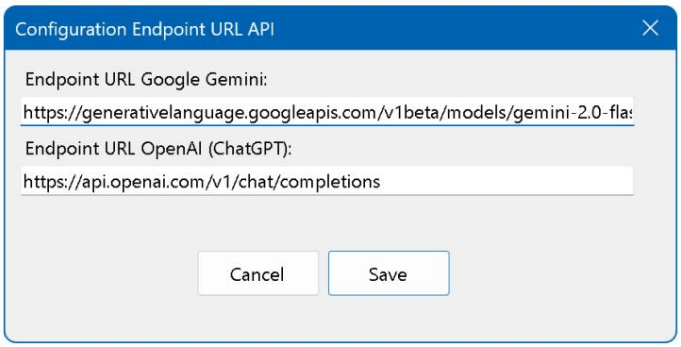
From the top menu, open **Settings -> AI Channel Info and Context.** This will open a table that's essential for training the AI:

- **Name (Channel Name):** Write the real name of the sensor (e.g. *Water Temperature, Suspension Travel, Wheel Speed).*
- **Description:** Add a text note explaining to the AI what that item is for. channel. This is **crucial** for the mathematical algorithm to work properly. (Ex: *"It helps us understand how hard the rider brakes,"* or *"It helps us identify front-wheel jolts.")*

Context (Contexts): On the same screen, you can select the "Context" you're working in. Contexts pre-set the AI's "focus." Here are some built-in examples:

- **Drone & UAV Flight:** Focus on flight stability, motor vibrations (FFT) and voltage drops. Looks for high frequency resonances and suggests PID filter calibrations.
- **Precision Agriculture Tractor:** Focuses on GPS accuracy, wheel slip percentage, and power take-off (PTO) rpm stability. Suggests tire pressure adjustments based on tractive effort.
- **Industrial Machinery CMS:** Machine condition monitoring. It searches for bearing resonance frequencies and abnormal vibration peaks to prevent thermal breakdowns.
- **Engine Dynamometer:** Engine dyno testing. Focus on torque/power curves, Stoichiometric ratio (AFR) and knock peaks. Analyzes and suggests ignition delays in case of anomalies.

From the top menu, open **Settings -> AI Link.** This will open a window where you can enter links to access the Gemini and OpenAI model APIs.



Use this link for Gemini:
https://generativelanguage.googleapis.com/v1beta/models/gemini-2.0-flash:generateContent?key=

Use this link for OpenAI:
https://api.openai.com/v1/chat/completions

8.3 How to query the AI (Best Practices)

Once you've uploaded a dataset and defined the context, you're ready to query the AI by typing in the search bar. For best results, **ask technical, targeted questions that require calculations.**

- *Weak question:* "Look at the graph and tell me how the vehicle is doing." (The AI has no eyes and will give you a generic answer).
- *Excellent question:* "Analyze the engine vibration channels. Do an FFT to find the dominant frequency and tell me if there are any anomalous resonances. Plot the spectrum for me."

Examples of Effective Prompts:

- *"Show me the frequency spectrum (FFT) of the engine vibration. There are frequencies dominants that indicate an imbalance?"*
- *"Is there a correlation between the sagging battery voltage and the analog 1 signal drain spikes? Show me a scatter plot."*
- *"Calculate the integral of the acceleration X to estimate the velocity and compare it to the GPS velocity."*

8.4 The Analysis Interface (Reports, Graphs and Debugging)

When you click on **Analyze,** the interface comes to life and is divided into three sections:

1. **The Debug Panel (Bottom Right):** This is your window into the AI's "brain." Here you'll see the machine's reasoning unfold in real time *("Looking for peaks...", "Running the correlation tool...")* and the raw mathematical results it extracts from the dataset.

2. **The Text Report (Top right):** A readably formatted page where the AI gives you its final answer, its diagnostic conclusions and advice on preventative actions to take.

3. **Generated Graphs (left):** Based on its inferences, the AI will automatically draw new graphs below the text to support its thesis. The AI can generate complex time graphs (mathematically crossing multiple channels), frequency histograms, XY scatterplots, or frequency-based spectrum analysis (FFT).

8.5 Exporting the Report (Export PDF)

If you've performed an analysis that's particularly useful for a diagnosis (for example, to justify maintenance on an industrial machine or a setup for a customer), you can click the **Export PDF button.** The software will generate a professionally formatted document containing the dataset information, your initial question, the AI's step-by-step reasoning, the graphs it generated, and its final technical conclusion. A great tool to include in your test documentation!

PART 2: Hardware

Chapter 10: Hardware Architecture and Inputs

The Datalogger is designed to acquire values from the CAN bus, using different protocols: OBDII, J1939 or Open CAN.

10.1 CAN Bus Interface

The system integrates a CAN Bus capable of extracting up to 16 channels simultaneously.

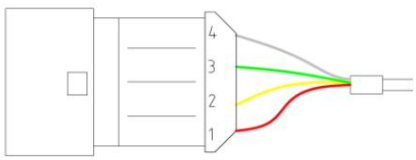
- **Transceiver:** High-Speed CAN 2.0B.
- **Protocols:** Using the GUBELLINI DataStudio software it is possible to configure the CAN bus according to the OBD II, SAE J1939, ISOBUS (ISO 11783), OpenCAN (EN-50325-4) protocols.
- **Termination:** The 120 Ohm termination resistor is present internally in the data logger..

10.2 Main Connector: AMP SuperSeal Series 1.5 4-pin (Automotive Grade)

To ensure maximum reliability in signal transmission and maintain the IPX7 waterproof certification, the physical interface is entrusted to a single automotive-grade connector.

Pin-out Table (Connector Mapping)

Below is the function of each pin on the connector.



AMP SuperSeal 1.5
Male (4PIN) **282106-1**

Pin	Name	Description	I
1	VBatt	Diet	POWER SUPPLY
2	CAN H	CAN line	CAN BUS
3	CAN L	CAN line	CAN BUS
4	GND	Mass	POWER SUPPLY

Chapter 11: Dimensions and Mechanical Installation

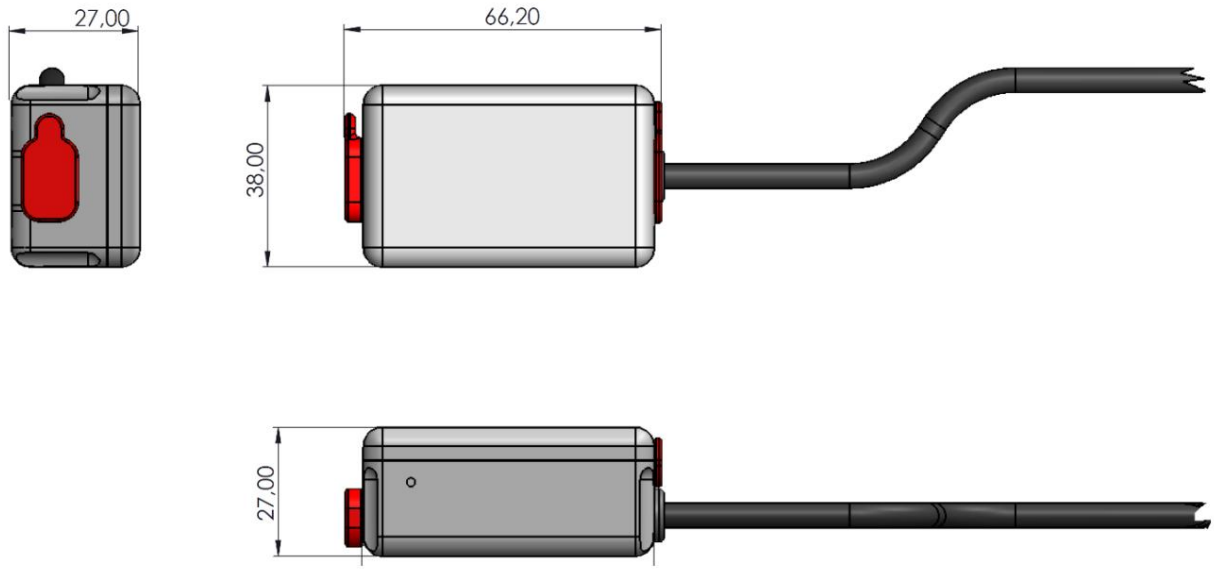
To ensure system reliability (IPX7 certified against immersion) and correct reading of the inertial sensors (IMU), the data logger must be installed respecting the dimensions and tolerances reported below.

11.1 Mechanical Specifications

- **Casing Material:** Nylon PA12H.
- **External Dimensions (L x W x H):** E.g. 93 mm x 63 mm x 30 mm.
- **Weight:** 180 grams.
- **Protection rating:** IPX7.
- **Operating Temperatures:** From -20°C to +85°C.

11.2 Technical Drawing and Dimensions

Note: The dimensions shown are expressed in millimeters (mm).



11.3 Connection diagram

GUBELLINI sas by Diego Gubellini & C.

Via Euridia Bergianti 10B 40059 Medicina BO Italy | VAT IT 03466001207

URL. https://www.gubellinielectronics.com | https://www.gripone.com

EMAIL. info@gubellinielectronics.com | info@gripone.com