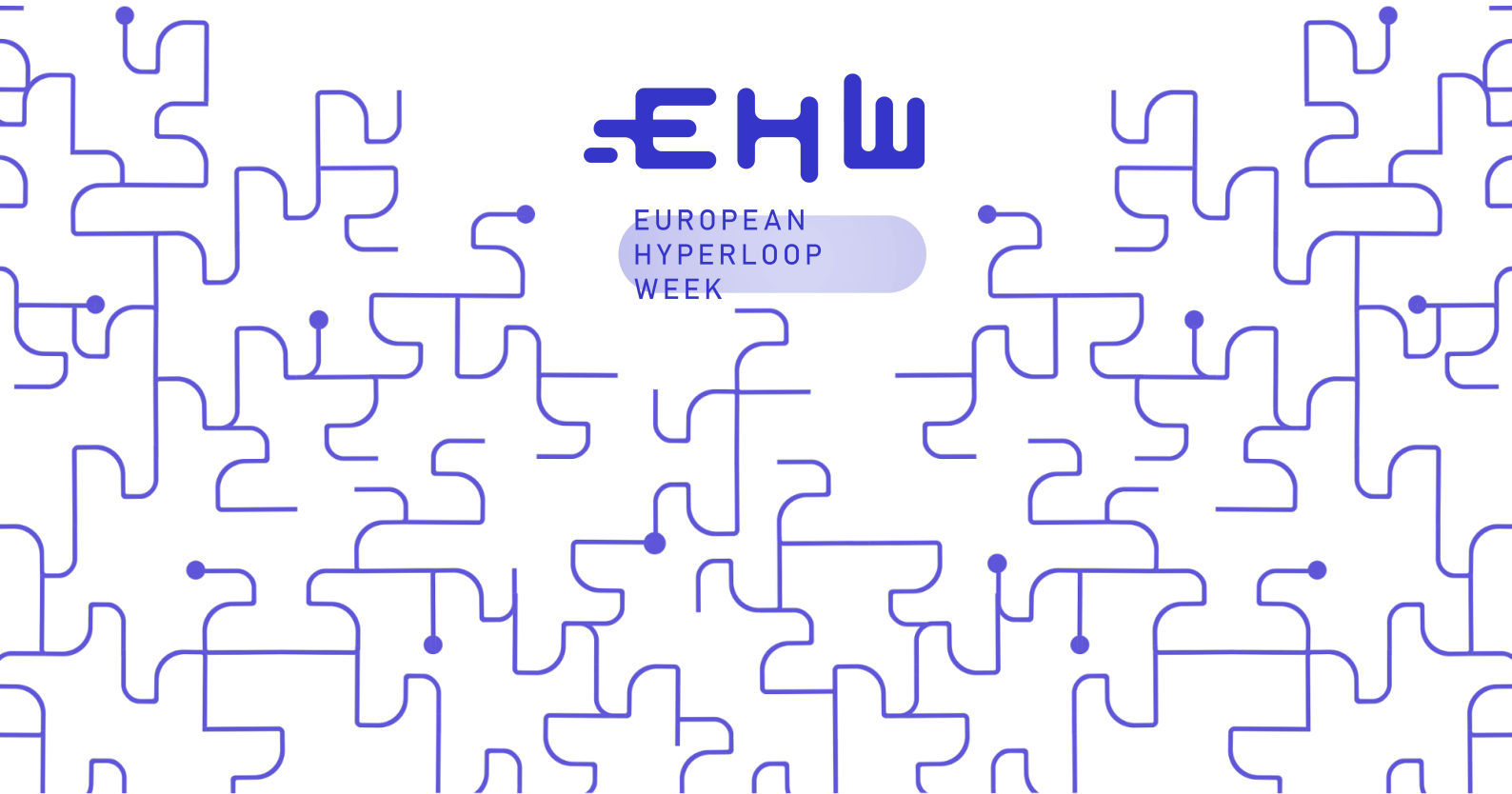
zürich 2024

Standard Software Proposal

European Hyperloop Week

Tech Committee

November 20, 2023



This page has been left intentionally blank.

Foreword

Dear Participants,

As seen in previous editions, data logging can become a complex process if not designed correctly. For this reason, the EHW presents a simple data logging proposal that can be used in two ways.

- It can be copied and used in a system. Note that in the case of a complete system this will not award teams any points on innovation, but will reduce the overall workload and allow a novel team to ensure a more reliable demonstration. In the case of a subsystem demonstration that is not Sense & Control it will still need to use data logging. For this reason, we recommend adhering to this proposal as there is no real penalty and we encourage teams to not overcomplicate it.
- Second, it can be used for inspiration and developed further depending on the team necessities. This is probably the most useful use case, as most teams will benefit from adapting it to their needs acquiring some technical knowledge in the process.

Note that in case of an hypothetical Sense & Control proposal the team is responsible for developing the proposal up to a point where it adds any value to the competition and can be therefore accepted.

This proposal serves as a suggestion for teams and is in no way mandatory.

Best of regards and good luck!

The EHW Technical Team

This page has been left intentionally blank.

Contents

November 20, 2023, the EHW Committee

This page has been left intentionally blank.

1 System Architecture

Every system needs a telemetry architecture to validate its behaviour with real time data. For this reason we present our own proposal following standards of the industry and the likes of successful teams in previous editions.

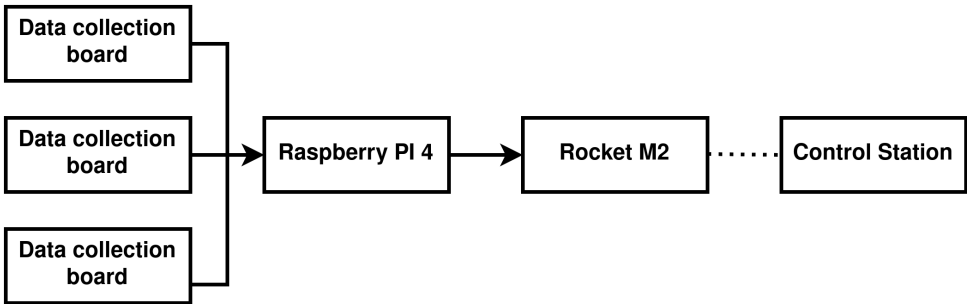


Figure 1: System Architecture

At the start of the chain we find a sensor that collects data such as position, temperature or current of the system. This sensor will normally collect data and send it in SPI or I2C synchronous communication protocols. These protocols can be interpreted by any microcontroller with digital input capabilities, like the one in a Raspberry PI or the STM32 family.

Here are some examples of commercial sensors:

- Hall-Effect current sensor
- IMU Sensor

If the workload of your receiving board is low enough, you could send this data directly to a Network Access Point (NAP) which would then send it to a wirelessly connected laptop. As this is not normally the case, you can use an auxiliary board that sniffs the information of all the other boards using a switch in the case of Ethernet or a direct connection to the bus in the case you use CAN. In both cases, the board must still have Ethernet connection in order to connect to the NAP.

Once at the NAP, the information can be accessed via WiFi from your laptop using either TCP or UDP connections. Finally the information can be processed by a backend and displayed in a clear and safe manner using a frontend framework.

2 Data Collection

Sensors normally send data using synchronous communication protocols where the sensor takes the role of a slave/peripheral and the board acts as a master/controller. This is done to prevent any clock synchronization errors and to simplify all-to-one communication.
The most important communication protocols are:

- I2C
- SPI

SPI and I2C are communication protocols with unique strengths. SPI provides high-speed, simultaneous data transfer (full-duplex) and supports multiple devices, ideal for fast applications. However, it requires more pins, has limited cable length, and is less power-efficient.

In contrast, I2C offers a low pin count, unique addressing, and built-in acknowledgment ensuring reliable data transmission, excelling in multi-master scenarios. Yet, it has lower speeds, operates half-duplex, and faces cable length limitations. The choice you make depends on your application needs like speed, complexity, and power efficiency.

Boards then need to communicate data between them to ensure safe operation and synchronize/update information on distributed architectures. For this two protocols have proven successful in previous editions: CAN and Ethernet.

2.1 CAN

For communication between different boards we propose the use of a CAN (Control Area Network) bus instead of Ethernet. Ethernet may be a lot faster than a CAN bus, but the process of configuring it can be a lot more complex.

Further advantages of a CAN include but are not limited to:

- The technical features of CAN have made it particularly effective in the vehicular environment, starting with its high tolerance for noise, supported by CAN's physical layer and protocol.
- CAN supports native multicast and broadcast, provides built-in frame priorities, offers non-destructive collision resolution and easily supports long single buses.

So, while CAN is a reliable choice for many applications, it's important to consider these drawbacks when weighing it against alternatives like Ethernet:

- A single CAN bus is limited to a maximum data-transmission rate of 8 to 10 Mb/s and therefore struggles to keep up with the demands of today's connected vehicles.
- As more devices are connected to the same bus, the bus' performance is greatly decreased.

2.2 Ethernet

On the other side, Ethernet can be used for obtaining greater bandwith overall.

Further advantages of Ethernet include but are not limited to:

- Devices can be connected and disconnected in real time, with zero downtime, marking a significant advantage over CAN buses.

On the other side, here are the main drawbacks of using Ethernet:

- More expensive physical-layer interface
- costs associated with required switches
- complications surrounding EMI and EMC issues with two-wire unshielded twisted-pair (UTP) Ethernet.
- Ethernet communication is non-real-time and non-deterministic.

2.3 Raspberry Pi 4

As an alternative to custom designed control boards, we recommend using the Raspberry Pi 4. It offers solid performance, extensive documentation and is also energy efficient.
In addition, with the release of the Raspberry Pi 5 it will become more readily available and possibly cheaper. While the newer variant boasts several improvements, we believe that the predecessor will be sufficient for the needs of most teams.
Some key features of the Raspberry Pi 4 relevant to embedded systems are:

- Quad-core processor running at 1.8GHz with RAM options ranging from 1GB to 8GB.
- 5V DC supply via USB-C or GPIO, with optional power over Ethernet
- Operative temperature range of 0 to 50°C.
- High-speed wireless, Bluetooth (5.0 and BLE), and Gigabit Ethernet.

More documentation can be found on their official website.

2.4 Rocket M2

In previous editions, the Rocket M2 has been the preferred option by most teams. It is a rugged, high-power, linear 2x2 MIMO radio which achieves enhanced receiver performance. Its speed can achieve 150+Mbps real TCP/IP.

For more information, you can refer to the official documentation of the Rocket M2.

3 Control Station / Graphical User Interface (GUI)

A GUI is a user interface that allows the user to interact with an electronic device through the use of interactive elements such as images, icons, buttons and possibly audio instead of text based commands.

The interactive elements in the GUI let the user trigger certain actions or input data in order to make a change in the system. Essentially, it improves the ease in navigating and interacting with information through visual presentation and interactive elements.

The GUI consists of a frontend and a backend. The frontend allows the user to see and interact with the vehicle and includes all that is visible by the user. The backend on the other side is responsible for the storing, processing and retrieval of data, managing logic, and the overall functionality of the system.

3.1 Backend

Backend development is usually done using languages such as Python, Ruby, or Java, along with frameworks like Django, Ruby on Rails, or Node.js.

Here are some critical features that a backend must have in order to compliment with a real time critical system standards:

- Redundant systems to ensure continuous operation even in the event of component failures.
- Fault-tolerant architectures recover from unexpected issues without compromising functionality.
- Real-time processing of data from sensors, cameras, and other inputs to provide instant feedback to the frontend.
- Algorithms for quick decision-making in response to changing conditions.
- Over-the-air (OTA) updates to ensure that vehicle software is always up-to-date.
- Ability to roll back updates in case of unforeseen issues.

Below are two open source examples of the backend provided by UPV and Swissloop that are free to use and adapt.

- UPV 2023 Backend
- Swissloop GUI

3.2 Frontend

Frontend development uses technologies like HTML, CSS, and JavaScript to create the visual elements that the user interacts with.

The frontend is not only important for aesthetic purposes, but it is a crucial part of a system's safety. Here are some critical features that a frontend must have in order to compliment with EHW standards: Use of intuitive symbols and colors to quickly convey information without causing distraction.

- Immediate alerts for critical issues like overcurrent or a faulty board.
- Visual and auditory warnings for potential hazards or emergency situations.
- Integration with sensors and cameras to provide a real-time view of the vehicle's surroundings.
- Instant and prominent display of emergency braking activation.
- Minimization of complex menus and options to reduce distraction.
- Allow to personalize the information displayed based on the demonstration needs.
- Ensure that customization does not compromise critical safety information.

Below is an open source example of the frontend provided by UPV and Swissloop.

- UPV 2023 Frontend
- Swissloop GUI

3.3 OpenMCT

OpenMCT is an open-source real time data visualization tool created by NASA. Some of it's most common use cases are:

- Monitoring of IOT devices
- Drones
- Robotics
- Electronic health monitoring
- Computer and network performance monitoring
- Process control monitoring

For this reason we believe that it OpenMCT is an ideal tool for the development of control stations for a hyperloop system.

For more information and projects made with openMCT you can refer to the official NASA website.