

eFMI-based battery management system demonstrator

Industry	General
Domain	Control, batteries
Use case	Test of battery management system (BMS) control logic in combination with a virtual battery cell plant model on a μ C board (e.g., Arduino® Uno Rev3)
Value proposition	<ul style="list-style-type: none"> Integration of multi-domain physics system models within electronic controls development From equation-based physics to embedded code

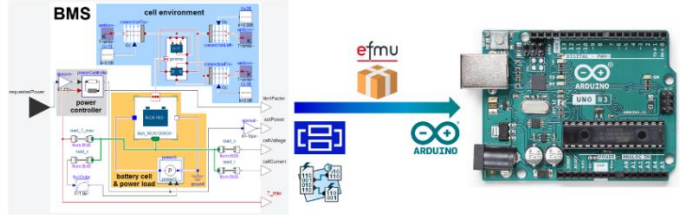


Figure 1: Battery management system (BMS) demonstrator – Modelica model, Arduino® Uno Rev3 target and DS tooling Dymola & Software Production Engineering (SOP-OC).

In this demonstrator – showcased at the Dassault Systèmes booth of the [16th International Modelica & FMI Conference](#) (September 8-10, 2025, Lucerne, Switzerland) – a simple battery management system (BMS) modeled by means of first order principles in Modelica (a-causal equations) is deployed on a real-time μ C-board (Arduino® Uno Rev3) with the help of Dassault Systèmes eFMI toolchain: Dymola and Software Production Engineering (SOP-OC). The BMS operates a passive-cooled battery to meet power requests as good as possible without endangering the battery due to overheating, whereas the cell core temperature, power load and cooling are real-time simulated. The virtual temperature sensor of the simulation enables to avoid deployment of a real temperature sensor in the battery cell, an in practice typical situation.

Objective of the demonstrator is to show the added value of the eFMI technology for advanced controller design and the maturity of Dassault Systèmes eFMI tooling.

eFMI overview

The open eFMI Standard (Functional Mock-up Interface for embedded systems) enhances embedded software development by integrating high-level simulation models, like multi-domain physics models in Modelica, into the design process. The [Modelica Association Project website for eFMI](#) summarizes the objective pretty well:

“The eFMI® Standard is an open standard for the stepwise, model-transformation-based development of advanced control functions suited for safety-critical and real time targets. Its container architecture defines the common ground for collaboration among the stakeholders and toolchains along the various abstraction levels from high-level modeling and simulation – e.g., a-causal, equation-based physics in Modelica – down to actual embedded code.”

By leveraging physics-based models, control development can be extended beyond conventional approaches, enabling:

- **Virtual sensors** (observers)
- **Model-based diagnostics**
- **Inverse physics models** as feedforward components in control structures
- **Model predictive control (MPC)**
- **Advanced operating strategies**
- ...and much more

The integration of eFMI thus allows a seamless transition from high-level physics modeling to production-ready embedded code, reducing development effort, improving accuracy, and supporting advanced control strategies in a safe and efficient manner. An exported model that applies the eFMI Standard is named eFMU (Functional Mock-up Unit for embedded systems).

For our tooling ecosystem, the eFMI support of Dymola for example enables Modelica users to reuse their equation-based designs in the safety-critical, embedded real-time domain, adding significant value to their existing Modelica assets and opens new markets for Dymola and our Modelica library portfolio when not “just” simulation is required but also embedded code generation capability.

For more details about the eFMI Standard, a nice [overview video at YouTube](#) from the tutorial given at the International Modelica Conference 2023 can be consulted (≈35 min). A very [brief summary about the eFMI Standard](#) also exists (≈4 min).

Demonstrator application scenario

Figure 2 illustrates the setup for the eFMI BMS demonstrator in a hard real-time simulation configuration. The BMS consists of the virtual battery cell, power load and cell environment plant models combined with a power controller model, all deployed and executed on a μ C-board (Arduino® Uno Rev3). Two externally connected potentiometers (analog input signals) allow users to define the requested power from the battery cell and to recalibrate the ambient temperature of the system model. Based on these analog inputs, the core temperature of the battery cell is computed. According to the computed core temperature, the power controller limits the actual applied power at the cell to ensure safe operation (i.e., avoid damage due to overheating). The model currently focuses mainly on the electro-thermal behavior of the cell while neglecting aging effects which are usually part of long-time offline simulations, not online control.

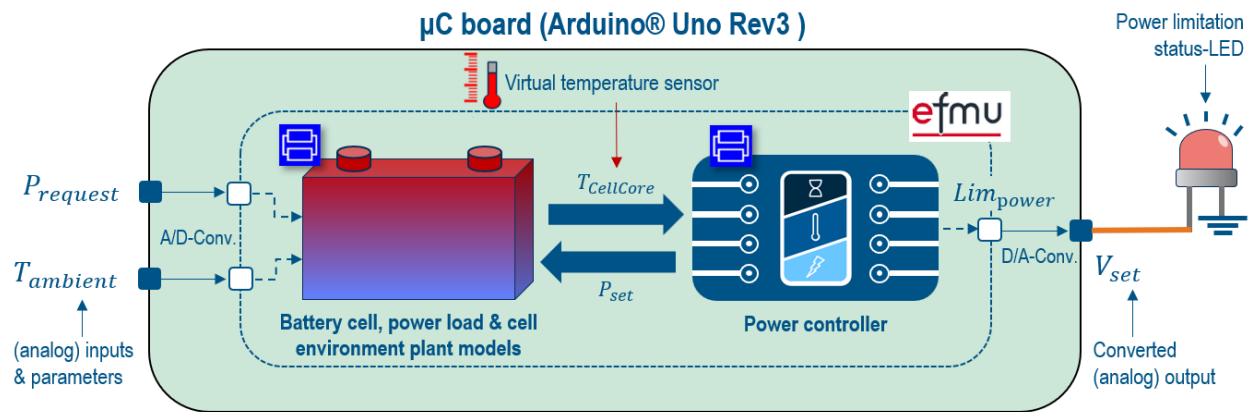


Figure 2: Overview of eFMI demonstrator setup.

In this scenario, the cell's core temperature is of particular interest. Measuring it directly in a real-world application, for example by adding sensors during cell production, is challenging and costly. By integrating multi-physics models into the embedded code, more sophisticated virtual sensors can be supported. For demonstration, the setup uses only a fully virtual cell model. In a real-world application, however, this virtual model would run in parallel to a physical cell and combine its results with additional sensor inputs, such as surface temperature measurements and load profiles measured at the terminal pins. This hybrid approach increases the accuracy of the core temperature estimation, since the virtual sensor benefits from both the predictive power of the model and the corrective influence of real measurement data.

Implemented Dymola model

Figure 3 shows the implemented Dymola model of the demonstrator. The model can be divided into four parts:

1. Battery cell & power load

Model of an NCA-type battery cell from the Dassault Systèmes Battery Library. The parameters of the electrical and thermal sub-models are derived from the cell's data sheet. Material properties for the thermal model are adapted from literature sources. A connected power source component at the cell's terminal pins allows the definition of changing load profiles during simulation.

2. Power controller

Controller model limiting the power linear depending on the cell core temperature and the maximal permitted temperature to avoid overheating.

3. Cell Environment

Convective heat exchange between the cell and the environment. The ambient temperature can be (re)calibrated during execution (eFMI tunable parameter).

4. In-, outputs and tunable parameters

Input is the requested power `requestedPower`. The ambient temperature used for passive cooling/heating is provided as tunable parameter. The outputs are the limiting

factor `limitFactor` of the power load (values between 1 and 0), the actually set – i.e., by the power controller limited – power `setPower`, as well as the cell's core temperature `T_max`, terminal voltage `cellVoltage`, and current `cellCurrent`.

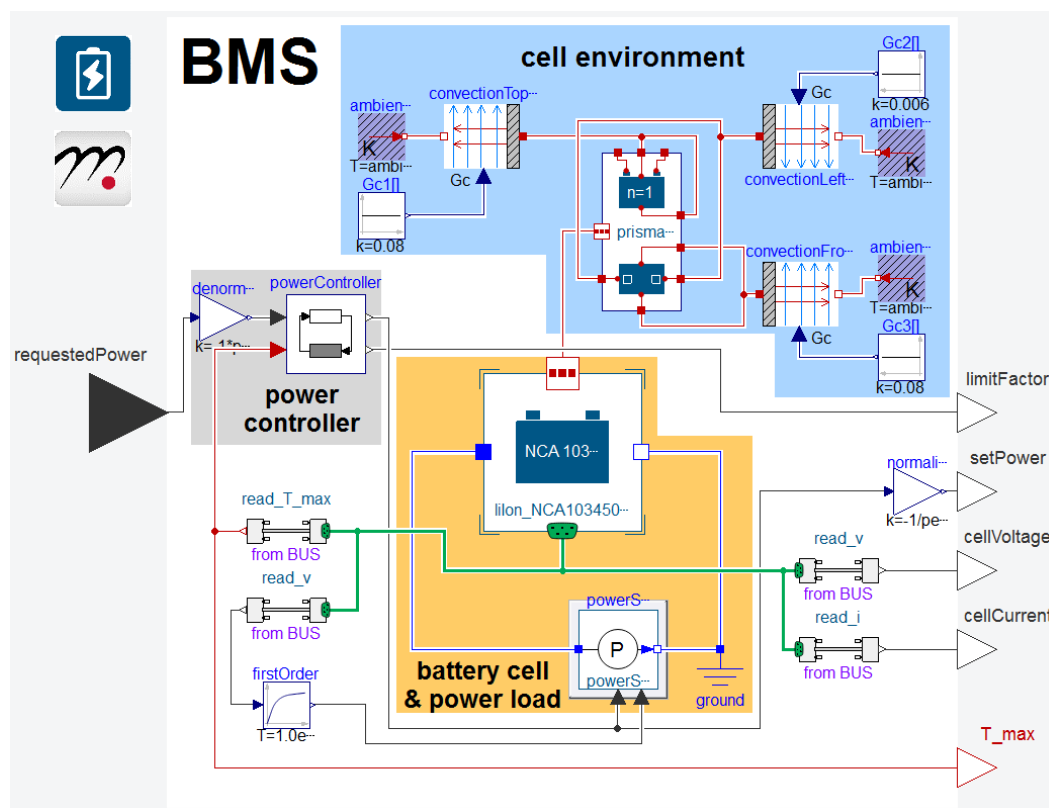


Figure 3: BMS Dymola model. The `setPower` output is the requested power limited by the power controller based on the simulated cell core temperature T_{\max} . The cell environment models the thermal conduction of the battery with the ambient, with the ambient temperature set by a tunable parameter that can be recalibrated at runtime.

The battery cell & power load models are out-of-stock models from the Dassault Systèmes Battery Library, with cell aging disabled. No additional modifications, besides using the MSL table adapters from the eFMI library, have been necessary to enable eFMI code generation.

Embedded setup and deployment


The demonstrator is tested and deployed on an [Arduino®](#) Uno Rev3 board:

- 32 KB flash memory
- 2 KB SRAM
- 16 MHz

The eFMU 32 Bit floating-point precision production code generated by Software Production Engineering requires $\approx 50\%$ of program memory; execution of a sampling step (`eFMI DoStep()`) requires $\approx 50\%$ of data memory and takes ≈ 2 ms. Hence, the used sampling period of 10 ms can be easily satisfied. The generated C production code is portable and self-contained; any other similar μ C-board would be feasible too. Dymola supports the export of eFMI production code as

Arduino® sketch however, enabling seamless further system integration and deployment from the Arduino IDE (upcoming feature of Dymola 2026x).



Figure 4: BMS live demo at the Dassault Systèmes booth of the 16th International Modelica & FMI Conference, September 8-10, Lucerne, Switzerland ().

The attached video (Figure 4) shows the execution of the deployed eFMU. For visualization purposes a python-based user interface for plotting signals of interest during simulation has been developed. As shown in Figure 2, the power limitation is additionally indicated by a Status LED using the following color coding:

- **Blue:** Successful eFMU initialization/reinitialization waiting for execution
- **Green:** Cell temperature is below defined limits → **no power limitation**
- **Purple:** Cell temperature exceeds lower boundary → **power limitation** (between 1 and 0)
- **Red:** Cell temperature exceeds upper boundary → **complete power limitation** (0 W)

Note, that the used cell (Panasonic NCA 103450 2350 mAh), power requests (max. 25 W) and resulting thermal and electric behavior are for demonstration purposes of eFMI only. A realistic setup would avoid any complete power limitation, never request such high power from such a household consumer AA cell or use a different cell model, likely provide active cooling, etc. – i.e., use a different parametrization. This does not change the feasibility of the Dassault Systèmes eFMI tooling. The setup is good for interactive plotting of the modeled effects while users play with the power request and ambient temperature potentiometers of the live demonstrator.

Conclusion

The developed eFMI demonstrator clearly shows the added value of integrating multi-physics Modelica models into embedded code for controller designs. By means of the eFMI Standard, conventional design approaches are extended by virtual sensors, model-based diagnostics,

inverse physics models, and model predictive control (MPC), thereby enabling more advanced operating strategies. Furthermore, the demonstrator highlights the potential for reducing development time and cost by reusing high-fidelity models throughout the entire toolchain, from simulation to embedded deployment. This facilitates a more seamless transition from system design to implementation and increases transparency in controller behavior. In addition, the approach paves the way for improved robustness and adaptability of control systems in real-world applications, making eFMI a promising enabler for future intelligent and efficient embedded solutions. Last but not least, the demonstrator shows the maturity of the eFMI toolchain – Dymola and Software Production Engineering (SOP-OC) – of Dassault Systèmes.

Acknowledgements

Demonstrator development team:

Christoff Bürger (Christoff.Buerger@3ds.com)

- eFMI support, article author, documentation

Christopher Stromberger (Christopher.Stromberger@3ds.com)

- Modelica model, interactive user-interface & plotting, live demo video (Figure 4)

Torsten Sommer (Torsten.Sommer@3ds.com)

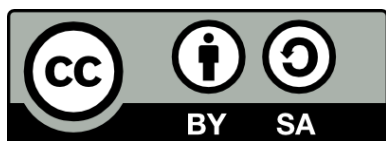
- Arduino deployment, physical setup

We like to thank Nils Modrow (Nils.Modrow@3ds.com) for consultancy regarding the Dassault Systèmes Battery library.

Copyright and licensing



© 2025, Dassault Systèmes, Modelica Association and contributors.



This work is licensed under a [CC BY-SA 4.0 license](https://creativecommons.org/licenses/by-sa/4.0/).

Modelica® is a registered trademark of the Modelica Association.

eFMI® is a registered trademark of the Modelica Association.

FMI® is a registered trademark of the Modelica Association.

Third party marks and brands are the property of their respective holders.