UNIVERSITÀ DI PISA

2018

# Hardware in the Loop Simulation Platform

REPORT

ROCCO MORELLO

08/07/2018

# Contents

# 1 Introduction

The aim of this work is to provide a Hardware in the Loop (HiL) simulation platform that allows a hardware battery state estimator, implemented by using a field-programmable gate array (FPGA) or a micro controller, to be tested under a wide range of operating conditions representative of the EV usage. In particular, the implemented platform is able to simulate an EV in many driving scenarios. The EV and its battery parameters are fully configurable. In this way, the battery behaviour is reproduced considering various real conditions, in which the unbalancing among the cells, their ageing and temperature variations can be simulated. Another important characteristic of the HiL simulator is the absence of power flow between it and real components. In fact, in order to test the most of the Battery Management System (BMS) functions, including the battery state estimation, only the values of the acquired quantities are needed. For this purpose, the platform also models the analog front-end, composed by the current and voltage sensors. The signals are sent to the estimators by using a digital interface.

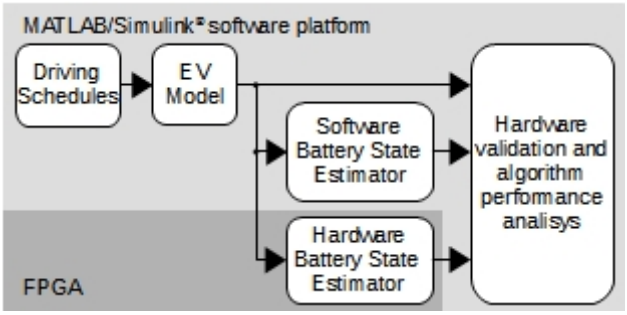


*Figure 1 Block diagram of the developed hardware-in-the-loop simulation platform.*

The software platform (Figure 1) has been entirely implemented in the MATLAB/Simulink® environment. The driving schedule block allows us to select a standard speed profile $v$, which is the input of the EV model block. The latter generates the input signals for the algorithms and the signals used to verify the obtained results. In this way the algorithms' performance can be carefully evaluated. Furthermore, the hardware algorithm implementation is validated by comparing its outputs to the results obtained by the software implementation of the battery state estimator.

## 1.1 Reference design details

The provided reference design consists of the following main components:

- The Battery Unipi library that contains the Simulink blocks used to build the HiL platform.
- The Simulink model that describes the HiL platform.
- The file to program the FPGA in which the hardware estimator under test will be instantiated.

These components are described in details in the following sections.

## 1.2 Add the Battery Unipi library to the Simulink Library Browser

To be able to access to the developed blocks directly from the Simulink Library Browser, the user has to follow these steps:

1. Copy the Model folder into a desired folder.
2. Add this directory with the custom libraries to the MATLAB path and save it, by executing the following commands:

```
addpath('directory_path')
savepath
```

3. Make sure that the MATLAB paths include the path to this directory. You can execute the following command in the MATLAB command prompt to find out which directories are in your paths:

```
path
```

4. Refresh the Simulink Library Browser.

# 2 Driving Schedules

The implemented platform is able to simulate various driving situations by using the "Speed_Profile" block. In order to implement this capability, an heterogeneous collection of standard driving cycles has been created by putting together 18 different profiles. These vehicle dynamometer test cycles are used for emission and fuel economy measurement, so they are representative of some real driving situations. The duration, distance and average speed of each cycle are reported in Table 1. The various driving schedules differ a lot in the average speed and, thus, in the electric power required from the traction battery.

| World areas | Driving schedule | Duration (min) | Distance (km) | Average speed (km/h) |
|---|---|---|---|---|
| Worldwide | WLTP class 3 | 30 | 23.3 | 46.5 |
| United States | UDDS | 23 | 12.1 | 31.5 |
| | HWFET | 13 | 16.8 | 77.5 |
| | FTP | 31 | 17.6 | 34.1 |
| | IM240 | 4 | 3.1 | 47.1 |
| | SFTP US06 | 10 | 12.9 | 77.2 |
| | SFTP SC03 | 10 | 5.8 | 34.5 |
| | NYCC | 10 | 1.9 | 11.4 |
| | LA92 | 24 | 15.8 | 39.6 |
| | LA92 short | 16 | 11.1 | 41.8 |
| European Union | EUDC | 7 | 6.8 | 58.6 |
| | NEDC | 20 | 8.5 | 25.4 |
| | ECE R15 | 3 | 0.8 | 16.5 |
| | ArtUrban | 17 | 5.0 | 17.6 |
| | ArtRoad | 18 | 17.2 | 57.4 |
| | ArtMw130 | 18 | 29.0 | 96.8 |
| | ArtMw150 | 18 | 29.8 | 99.5 |
| Japan | J1015 | 15 | 6.4 | 25.6 |

*Table 1 Driving schedules details*

These cycles are defined by various specialized worldwide organizations, therefore they can be classified on the base of the world areas in which they are used:

1. *International*: the Worldwide harmonized Light vehicles Test Procedures (WLTP) is maintained by the UNECE World Forum for Harmonization of Vehicle Regulations. The class determines the power-to-mass ratio of the vehicles. In particular, class 3 is representative of European and Japanese vehicles.
2. *United States*: the Urban Dynamometer Driving Schedule (UDDS), the Highway Fuel Economy Test (HWFET), the Federal Test Procedure (FTP), the two Supplemental Federal Test Procedure (SFTP US06 and SC03), the New York City Cycle (NYCC) and the Inspection and Maintenance (IM240) driving cycle are defined by the U.S. Environmental Protection Agency. The UDDS has been developed to test

heavy-duty vehicles. All the FTP cycles are variants of the UDDS one, used to test light-duty vehicles in urban scenarios. In particular, the SFTP US06 is a cycle with high speed and acceleration variations, instead the SFTP SC03 take in consideration the engine load due to the use of air conditioning. Another cycle useful to test urban profile is the NYCC, instead the HWFET is used for simulating highway scenarios. The IM240 is usually used for emission testing. The LA92 and the LA92 short (which includes the first 969 seconds of the LA92) are developed by the California Air Resources Board. They are used for urban scenarios, but are more aggressive than the FTP cycles.

3. *European Union*: the New European Driving Cycle (NEDC), the Extra-Urban Driving Cycle (EUDC) and the Economic Commission for Europe urban driving cycle (ECE R15) are maintained by the United Nations Economic Commission for Europe (UNECE). The Urban cycle (ArtUrban), the Rural road cycle (ArtRoad) and the Motorway cycles (ArtMw130 and ArtMw150, with a maximum speed of 130 and 150 km/h, respectively) are included in the Common Artemis Driving Cycles (CADC), developed within the European Artemis (Assessment and Reliability of Transport Emission Models and Inventory Systems) project.
4. *Japan*: the 10-15 mode cycle (J1015) had been used in Japan for emission and fuel economy testing for light duty vehicles.

## 2.1 How to use "Speed_Profile" block

The Speed_profile block is located in the Battery Unipi library. Once the block is placed in the model, the parameters configuration can be opened by double-click. The only parameter to set allows us to choose the

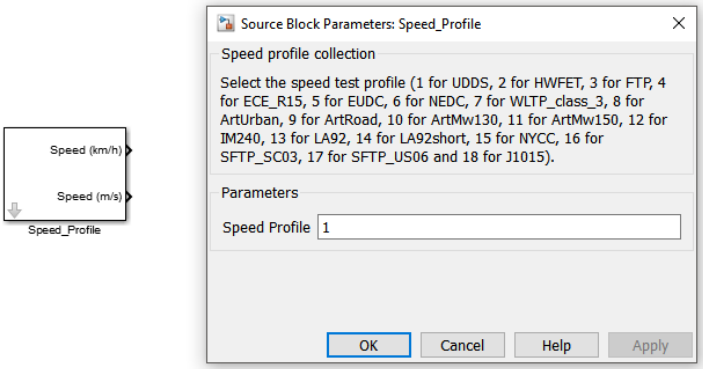


*Figure 2 "Speed_Profile" block and its parameters configuration window.*

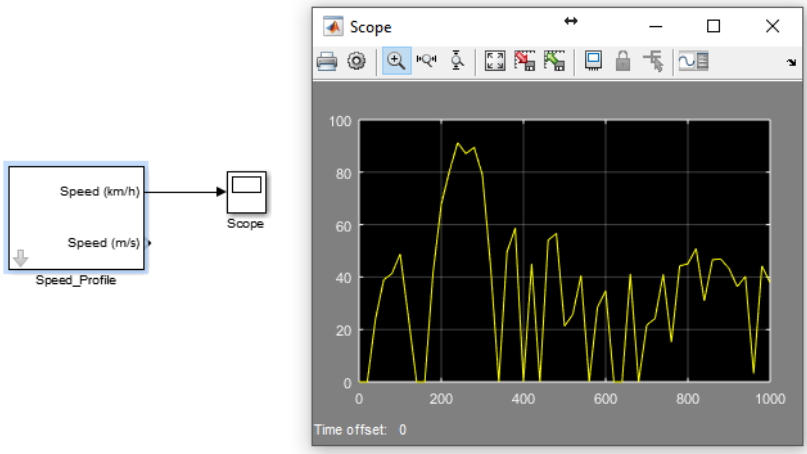


*Figure 3 "Speed_Profile" block in action.*

speed profile (follow the instruction shown in the parameters window). The profiles are stored in the folder …/Model/cycledata. They must be loaded in the workspace:

```
load ./Model/cycledata/cycles_artemis
load ./Model/cycledata/cycles_epa
load ./Model/cycledata/cycles_nedc
load ./Model/cycledata/cycles_wltp
load ./Model/cycledata/Japan_Light_Duty_Vehicles
load ./Model/cycledata/United_States_Light-Duty_Vehicles
```

Every sample time the block provides the value of the speed in km/h and in m/s. The selected profile is repeated until the simulation is stopped.

Figure 3 shows the functioning of this block, using the UDDS profile.

# 3 Electric Vehicle Model

The block diagram of the EV model is shown in Figure 4. The battery and the electric vehicle behaviour is simulated by mathematical models, executed with a tuneable integration time step (in our examples it is set to 100 ms, which is suitable for capturing the system dynamics of interest). The model outputs consist of the battery current $I_b$ and the cell voltages $V$ and the corresponding versions with the measurement noise $\hat{I}_b$ and $\hat{V}$. They form the input of the battery state estimator, which in turn computes the SOC estimation as well as the vector of the ECM parameters $p$. The traction battery, simulated by the application, and the battery state estimator, implemented in a FPGA, interact by using digital signals only. Consequently, the interface between the HiL model and the hardware can be implemented as a digital communication layer mapped over the JTAG link, without the need of reproducing the power interface of the battery, as instead required for validating other BMS functions as cell balancing.

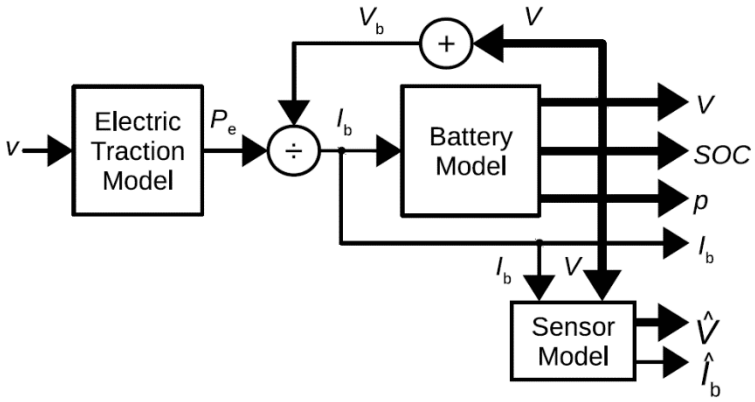


*Figure 4 Block diagram of the developed MATLAB/Simulink® EV Model for traction battery simulation.*

## 3.1 Battery Model

The battery model is able to simulate a battery composed by a given number N of series-connected cells. The only input is the battery current $I_b$ which is the same for all the series-connected cells, as shown in Figure 5.

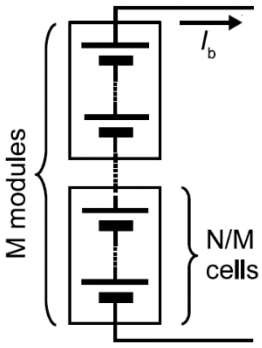


*Figure 5 Electrical organization of the simulated battery.*

At each time step, the model generates the arrays of the cell voltages $V$, temperature and SOC, as well as the current values of the model parameters. Each cell temperature is determined on runtime by a simple cell thermal model (CTM) that allows the simulator to provide an approximated estimation of the temperature changing. In the battery pack the cells are organized in M modules thermally isolated each other (see Figure 6). A module contains N/M cells that are side by side, so each CTM interacts with the nearest two in order to simulate the cell-to-cell heat exchange.

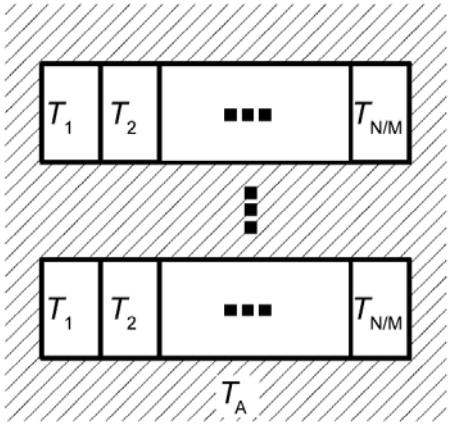


*Figure 6 Thermal organization of the simulated battery.*

The equivalent circuit model and the thermal model at cell at battery pack level will be described in Sections 5 and 6, respectively.

# 4 Electric Traction Model

The electric traction model block, shown in Figure 7, simulates the behaviour of an electric vehicle on a zero grade road by using a simple dynamic model.

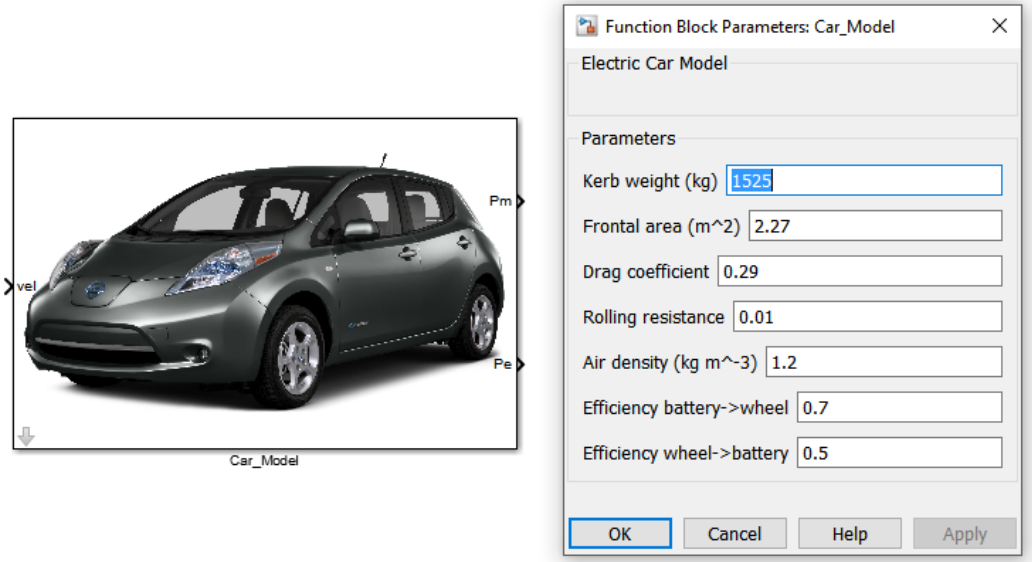


*Figure 7 "Car_Model" block and its parameters configuration window.*

The mechanical power $P_m$ is calculated starting from the speed $v$, as the sum of three contributions: one linked to the acceleration, one to the air resistance and the other to the rolling resistance:

$$P_m = Fv = \left(M\dot{v} + \frac{1}{2}\rho_{air} S\, C_X v^2 + \alpha_R M g\right) v$$

Therefore, the electric power $P_e$ is obtained from $P_m$ by using the following equation:

$$P_e = \left(\frac{1}{\eta_{wheel}} \frac{1 + sgn(P_m)}{2} + \eta_{reg} \frac{1 - sgn(P_m)}{2}\right) P_m$$

in which two different energy efficiencies are took in consideration, one for the traction ($\eta_{wheel}$) and the other one during the regenerative braking ($\eta_{reg}$). This is the model contained in the "Car_Model" block, as shown in Figure 8.

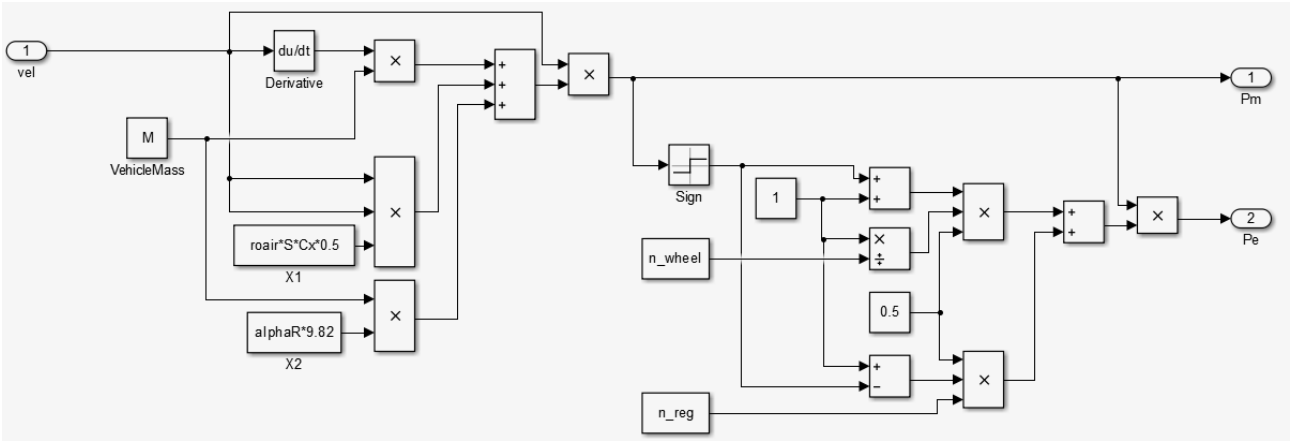


*Figure 8 Developed Simulink electric traction model.*

The model parameters used in the previous equations can be chosen in order to resemble a commercial electric car. Their description are reported in Table 2, in which there are also their values for the Nissan Leaf, the reference car used in this work.

| Symbol | Description | Value |
|---|---|---|
| $M$ | Kerb weight | 1525 kg |
| $S$ | Frontal area | 2.27 $m^2$ |
| $C_X$ | Drag coefficient | 0.29 |
| $\alpha_R$ | Rolling resistance | 0.01 |
| $\rho_{air}$ | Air density | 1.2 $kg/m^3$ |
| $g$ | Gravity acceleration | 9.82 $m/s^2$ |
| $\eta_{wheel}$ | Efficiency from battery to wheels | 0.7 |
| $\eta_{reg}$ | Efficiency from wheels to battery | 0.5 |

*Table 2 Parameters of Nissan Leaf*

Note that the input speed value must be provided in m/s, as shown in Figure 9. The block outputs are the electric and the mechanical power.

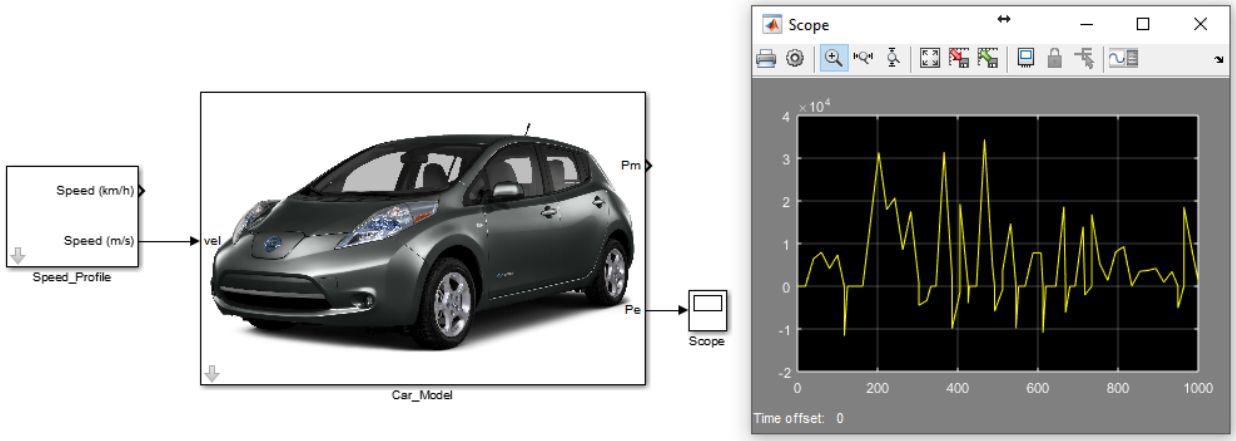


*Figure 9 "Car_Model" block in action.*

The battery current during a test is the results of the division between the electric power $P_e$ and the battery voltage $V_b$, as shown in Figure 4. $V_b$ is obtained as sum of the cells voltage calculated by the Battery Model and contained in $V$.

# 5 Electro-Thermal Cell Model

The cell model is composed by two main parts: the electric model and the thermal equivalent model. The interaction between the electric and the thermal models is shown in Figure 10.

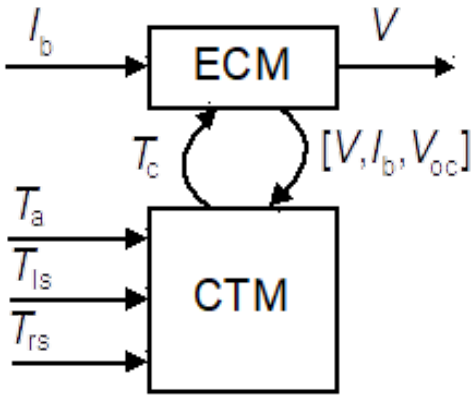


*Figure 10 Integration of the electric model with the thermal one.*

The inputs of the CTM are the temperature of the two largest surfaces $T_{ls}$ and $T_{rs}$, the ambiance one $T_a$, the cells' current, terminal voltage and open circuit voltage, provided by the ECM model and used to obtain the generated heat $\dot{Q}$. The output is the vector of the cell internal temperatures $T_c$.

## 5.1 Equivalent Circuit Model (ECM)

The electric model adopted is the ECM shown in Figure 11. This is a very common choice to simulate a Li-ion battery with high accuracy in a HiL platform. The left hand side models the cell capacity $Q_n$ and evaluates the SOC as the voltage across a linear capacitor with a capacity equal to $Q_n$ (expressed in Coulomb) divided by 1V.

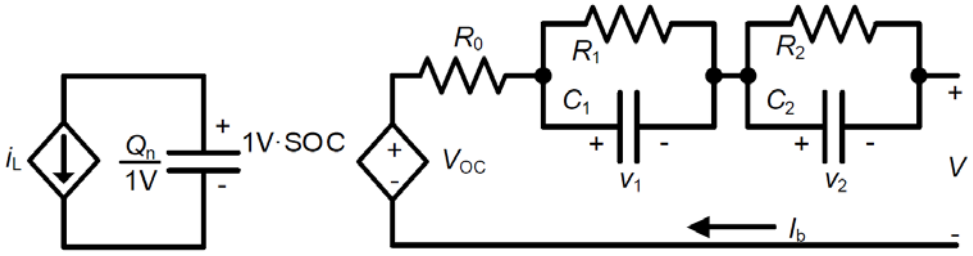


*Figure 11 Electric circuit model with two RC branches of the battery cells.*

The cell voltage $V$ is obtained by the sum of the open-circuit voltage $V_{OC}$ and a dynamic term, which accounts for the internal ohmic resistance $R_0$ and the double layer ($V_1$) and diffusion ($V_2$) effects of the Li-ion battery during charging and discharging.

The model parameters change with manufacturing variations, ageing and operating conditions, such as temperature, state of charge and current rate. In order to model the dependency of the parameters [$R_0$; $R_1$; $R_2$; $C_1$; $C_2$] on temperature, current and SOC, their values are stored in 3D LUTs. These dependencies are shown in Figure 12, Figure 13, Figure 14, Figure 15 and Figure 16. In this work, the LUTs have been populated with the values extracted from pulsed current tests (PCT) performed at different temperatures and with different pulse amplitudes on a 1.5Ah NMC cell. This test procedure consists of three phases:

1. Initial phase: the cell is completely charged (CC-CV mode) and then discharged, with the rated current, at the temperature of 25°C. This phase ensures that the following test will start from a well-known status.

2. 1 h pause: we let the cell to settle down ensuring that all the transients are expired before starting the real test.
3. Test phase: the charge/discharge cycle is then repeated a second time at the temperature and current of interest. The discharging and charging processes are stopped for 5 min at 1% SOC and then every 9% SOC in this second cycle. A pause of one hour is applied between the charging and the discharging cycles.

The model is then generalized to simulate a cell with the same technology but different capacity by proportionally scaling the LUT values with the capacity, directly for the resistive elements and inversely for the capacitive ones.

Regarding the open-circuit voltage $V_{OC}$ only the dependency on SOC is considered, while the capacity $Q_n$ is kept constant during the tests. The variability of the cell behaviour is considered by setting the initial SOC, the model parameters and the capacity of each cell individually.

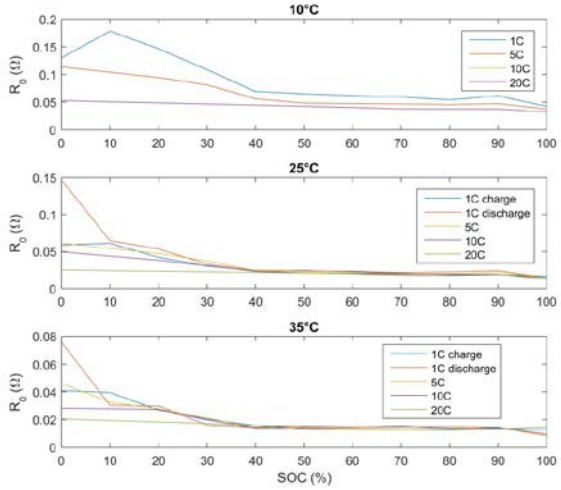


*Figure 12 Variation of the $R_0$ value, function of SOC, temperature and current rate (for a 1.5Ah Kokam cell).*

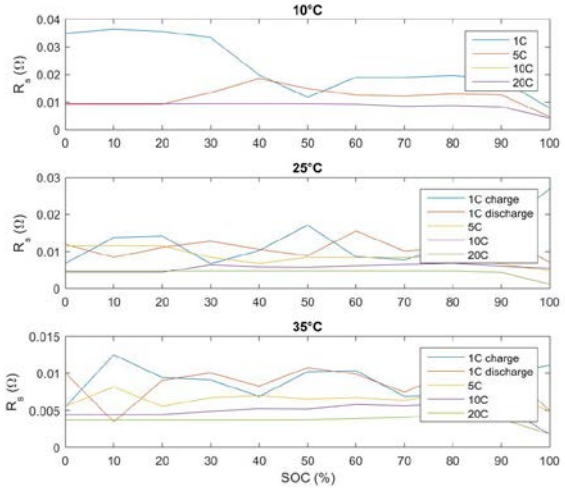


*Figure 13 Variation of the $R_s$ value, function of SOC, temperature and current rate (for a 1.5Ah Kokam cell).*

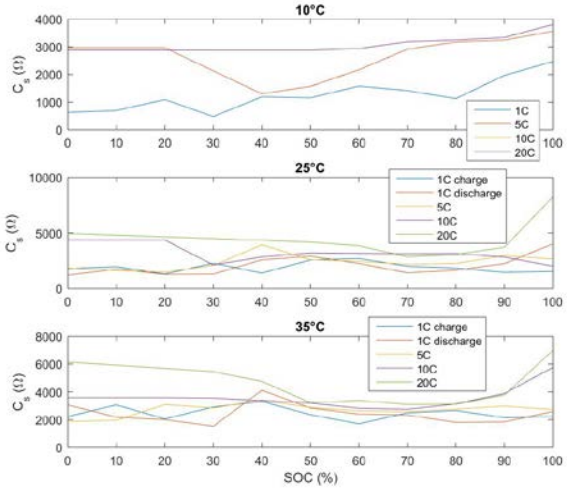


*Figure 14 Variation of the $C_s$ value, function of SOC, temperature and current rate (for a 1.5Ah Kokam cell).*

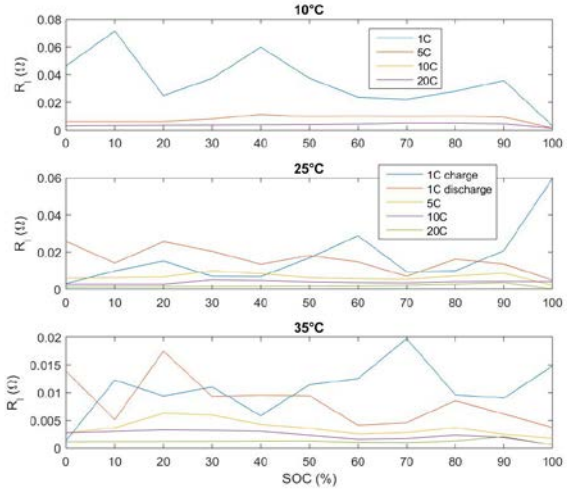


*Figure 15 Variation of the $R_l$ value, function of SOC, temperature and current rate (for a 1.5Ah Kokam cell).*

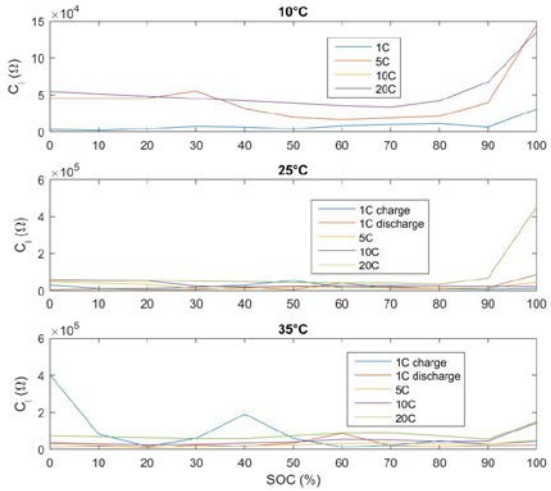


*Figure 16 Variation of the $C_l$ value, function of SOC, temperature and current rate (for a 1.5Ah Kokam cell).*

## 5.2 Cell Thermal Model (CTM)

The temperature of each cell is determined by a simple cell thermal model (CTM). This model uses the electrical equivalent model of the thermal system, as shown in Figure 17. It considers the heat exchange between cells and the external environment and also the cell-to-cell exchange in the battery pack.

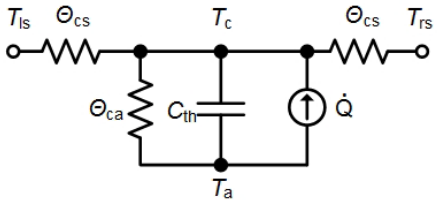


*Figure 17 Electrical equivalent thermal model of the battery cells.*

All the thermal resistances $\Theta_{cs}\ [K/W]$ which model the core-to-surface resistances have the same value. $T_{ls}$ and $T_{rs}$ are the temperatures between the surface of the considered cell and the nearest two cells' surfaces. Therefore, these refer to the two largest surfaces. The cell-to-air contact, which refers to the others surfaces and pads, is modelled by the thermal resistance $\Theta_{ca}\ [K/W]$. The temperature of the cell core $T_c$ is the voltage at the top terminal of the thermal capacitance $C_{th}\ [J/K]$:

$$T_c = \frac{1}{sC_{th}}\left(\dot{Q} - \frac{2T_c - T_{rs} - T_{ls}}{\theta_{cs}} - \frac{T_c - T_a}{\theta_{ca}}\right)$$

The current generator $\dot{Q}\ [W]$ models the cell self-heating. It is the sum of a reversible contribution, the entropic heat flow, and an irreversible part, due to ohmic losses:

$$\dot{Q} = \dot{Q}_{rev} + \dot{Q}_{irr}$$

The irreversible heat can be obtained by multiplying the cell overvoltage with its current (positive in the charging direction):

$$\dot{Q}_{irr} = I(V_{OC} - V)$$

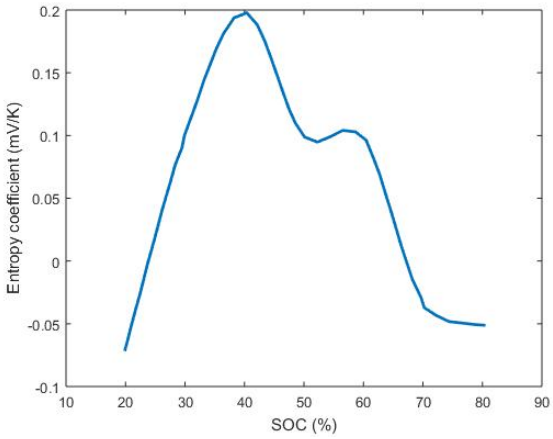


*Figure 18 Entropy variation vs. SOC for a NMC (Kokam) cell[1].*

The reversible heat is modelled as follow:

$$\dot{Q}_{rev} = IT_c\frac{\partial V_{OC}}{\partial T_c}$$

[1] Eddahech, A., Briat, O., & Vinassa, J.-M. (2013). Thermal characterization of a high-power lithium-ion battery: Potentiometric and calorimetric measurement of entropy changes. Energy, 61(0), 432–439.

where the value of the entropy variation $\partial V_{OC}/\partial T_c$ depends on the cell chemistry. In this work, we use a NMC Kokam reference cell with the entropy variation shown in Figure 18.

This model is used to build the battery thermal model in a modular way. The battery thermal model considers also the heat exchange with the environment and the nearest two cells in the battery pack. This phenomena is modelled by the thermal resistances $\Theta_{cs}$ and $\Theta_{ca}$. These include all the heat exchange phenomena, *i.e.*, conductive, convective and radiative heat transfer. The values of the thermal parameters are reported in Table 3.

| $\Theta_{ca}\ [K/W]$ | $\Theta_{cs}\ [K/W]$ | $C_{th}\ [J/K]$ |
|---|---|---|
| 67.4875 | 29.6123 | 21.4444 |

*Table 3 Thermal parameters for the reference cell.*

An initial value of these parameters have been found considering the cell chemistry and dimension (see Table 4). Then, the final values have been obtained by an optimization phase executed in Matlab/Simulink® environment.

| | | |
|---|---|---|
| Dimension | Thickness (mm) | 7.2 |
| | Width (mm) | 37.5 |
| | Length (mm) | 70.0 |
| Weight (g) | | 37.0 |

*Table 4 Reference cell dimension and weight specifications.*

In particular, the reference cell has been tested in a thermal chamber, set to a temperature of 24.833 °C, by using a stepwise current profile (shown in Figure 20). During the test, the surface temperature has been acquired. In this way, we can optimize the value of the parameters $C_{th}$, $\Theta_{cs}$ and $\Theta_{ca}$ by reducing the difference between the acquired and the computed surface temperature. To do that, a single cell thermal model has been used (see Figure 19).

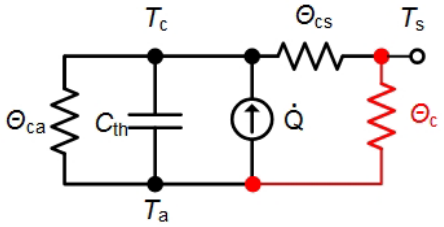


*Figure 19 Standalone cell thermal model.*

The thermal resistance $\Theta_c$ models the heat exchange between the two biggest cell surfaces and the environment. It takes in consideration only the convective phenomenon:

$$\Theta_c = \frac{1}{hA}$$

Where $h$ has been initially set to 10. Also this parameter is optimized by the optimization process, reaching a value of 12.2308 $W/Km^2$. The surface temperature is computed as follows:

$$T_s = \frac{T_a\Theta_{cs} + T_c\Theta_c}{\Theta_c + \Theta_{cs}}$$

with

$$T_c = \frac{1}{sC_{th}}\left(\dot{Q} - \frac{T_c - T_a}{\theta_{cs} + \Theta_c} - \frac{T_c - T_a}{\theta_{ca}}\right)$$

In order to allow cell with different dimensions, weight and capacity to be simulated, we can obtain the thermal conductivity

$$h = \frac{1}{\Theta A}\left[\frac{W}{Kmm^2}\right]$$

where $A$ is the surface. The thermal conductivity for the surfaces which are usually in contact with the other cells is $h_{cs} = 12.87\ W/Km^2$, instead of $h_{ca} = 9.57\ K/Wm^2$ of the lateral surfaces. Regarding the thermal capacity, we can easily obtain the specific heat:

$$c_p = \frac{C_{th}}{m} = 579.6\frac{J}{Kkg}$$

where $m$ is the cell mass in $[kg]$.

## 5.3 Cell Model Validation

The cell model has been validated by using a stepwise current test, as shown in Figure 20. The measured current of the real test is used as input of the implemented model, configured to simulate only one cell (see Figure 21). From Figure 20, we note that the predicted cell voltage is in very good agreement with the measured one, resulting in a maximum and an rms error of 238.6 mV and 15.1 mV, respectively. Good results are also achieved for the temperature prediction by using the implemented simple thermal model, obtaining an rms error of 0.41 °C.

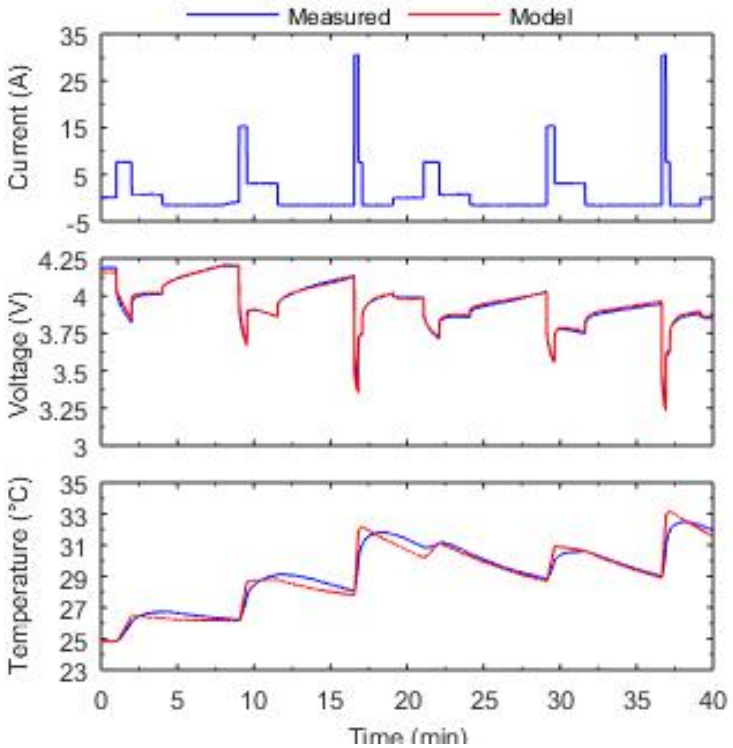


*Figure 20 Model and measured voltages and temperatures comparison.*

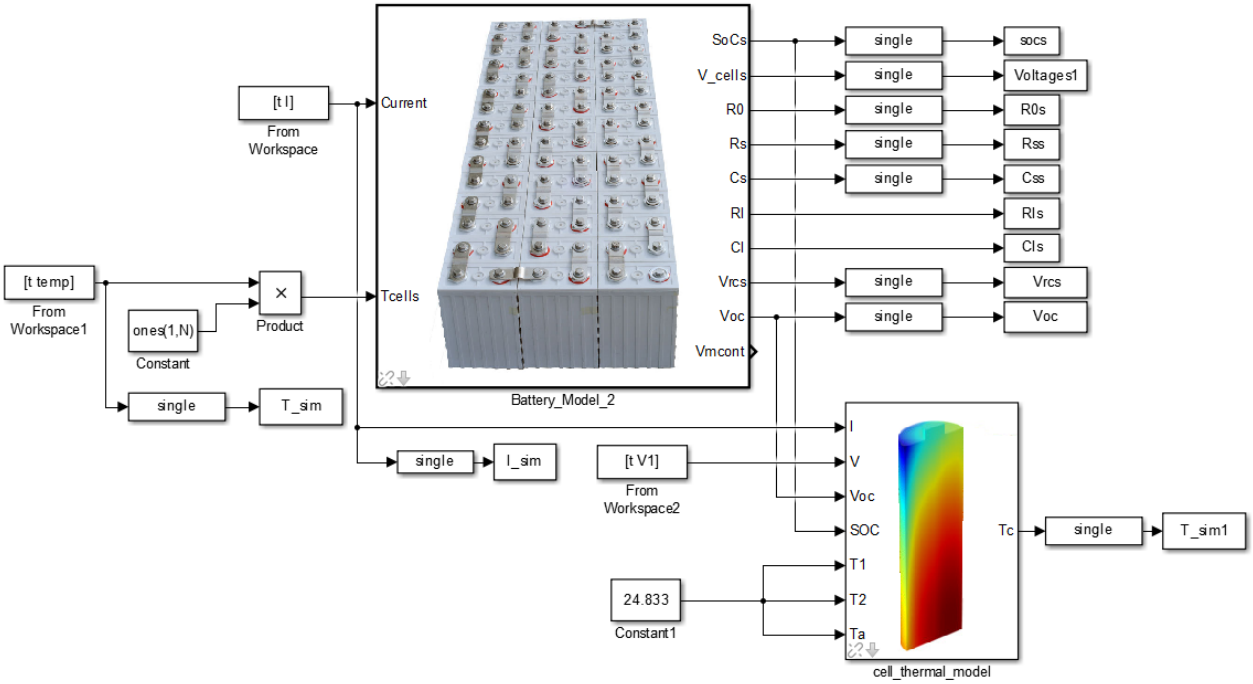


*Figure 21 Single cell electro-thermal Simulink model.*

### 5.3.1 Considerations

In order to reduce the model's complexity, we can analyse its behaviour in simplified cases. We analysed some modification with the aim to improve the accuracy of the model. These are summarized in Table 5. The best choice is to use a model that considers all the parameters dependent on the C-rate and which use the cubic spline interpolation method. This kind of model spends 17.13 seconds to execute the simulation reported in Figure 20. A good trade –off between accuracy and execution time could be to use the linear interpolation method. In this case, the model spends 14.86 seconds.

| Case description | Voltage Max error (mV) | Voltage rms error (mV) |
|---|---|---|
| C-rate dependency for all parameters, Cubic spline interpolation method | 264.7 | 16.0 |
| C-rate dependency for all parameters, modification on the RC groups, Cubic spline interpolation method | 265.1 | 16.3 |
| C-rate dependency for all parameters, Linear interpolation method | 275.9 | 16.4 |
| C-rate dependency for all parameters, modification on the RC groups, Linear interpolation method | 276.2 | 16.6 |
| No C-rate dependency for $R_0$ | 280.9 | 17.7 |
| No C-rate dependency for $R_0$, modification on the RC groups | 281.1 | 17.9 |

*Table 5 Model modifications analysis.*

# 6 Electro-Thermal Battery Model

The battery model is also composed by two parts: the electrical and the thermal model. The latter has a hierarchical structure, as shown in Figure 22. This structure will be described in details in the following sections.

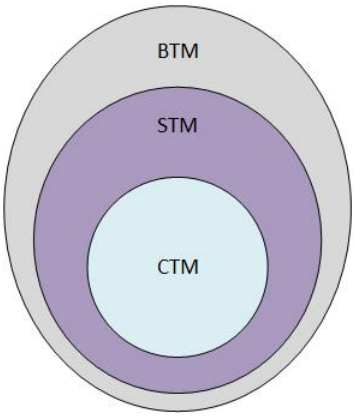


*Figure 22 Hierarchical structure of the battery thermal model.*

## 6.1 Electrical Battery Model

The electrical battery model is based on the replication of the ECM (Figure 23). The Battery Electrical Model is a Simulink "Each subsystem" block, set to simulate a series of N cells. All the battery parameters can be set by using the block setup window.

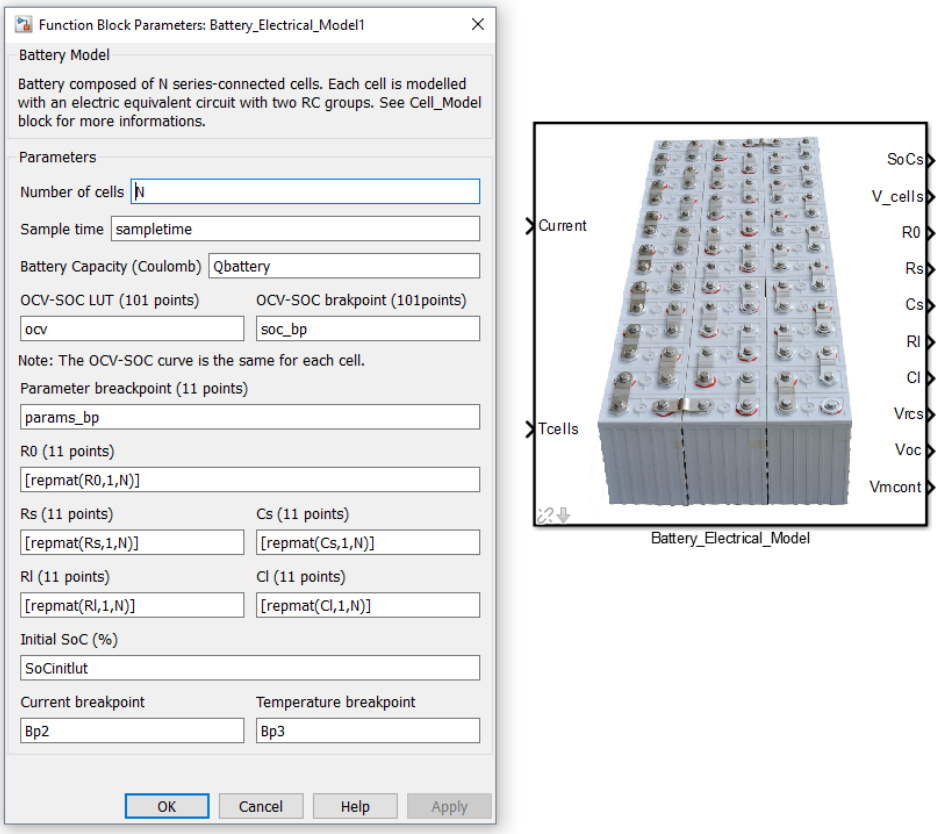


*Figure 23 Battery electrical model and its setup window.*

The following considerations must be done:

- Qbattery and SoCinitlut are vectors that contains all the N cells' capacity and initial state-of-charge, respectively;
- ocv and soc_bp must have the same dimension;
- R0, Rs, Cs, Rl and Cl are three-dimensional vectors. They have the same dimension of params_bp, Bp2 and Bp3 for SOC, current and temperature dependency, respectively.

All the battery's cells behave in the same manner if their parameters are the same. The parameter extracted from the Kokam cell are stored in the cell_model_params.mat file. Therefore, before simulating the electrical model, you usually have to execute the following commands (for a battery composed by 24 cells):

```
load cell_model_params
N = 24;
Qbattery = 1.5*3600*ones(1,N);
SoCinitlut = 99.5*ones(1,N);
sampletime = 0.25;
```

## 6.2 String Thermal Model (STM)

The cells in the battery pack are organised in modules (or strings). This organization is also used in the thermal model, as shown in Figure 6. Each module is thermally isolated from the others, but the cell-to-cell heat exchange is taken into account in the module. Therefore, the String Thermal Model (STM) must be able to simulate this phenomenon. The current and the room temperature inputs are the same for every cell. The terminal voltage and the open circuit voltage inputs refer to the corresponding cell. As $T_{lc}$ and $T_{rc}$ inputs the cells receive a mean value of the temperatures of the two cell put in touch. The situation is different for the cells that are at the ends of the module. In fact, they have as an input the room temperature. This model can be seen in Figure 24.

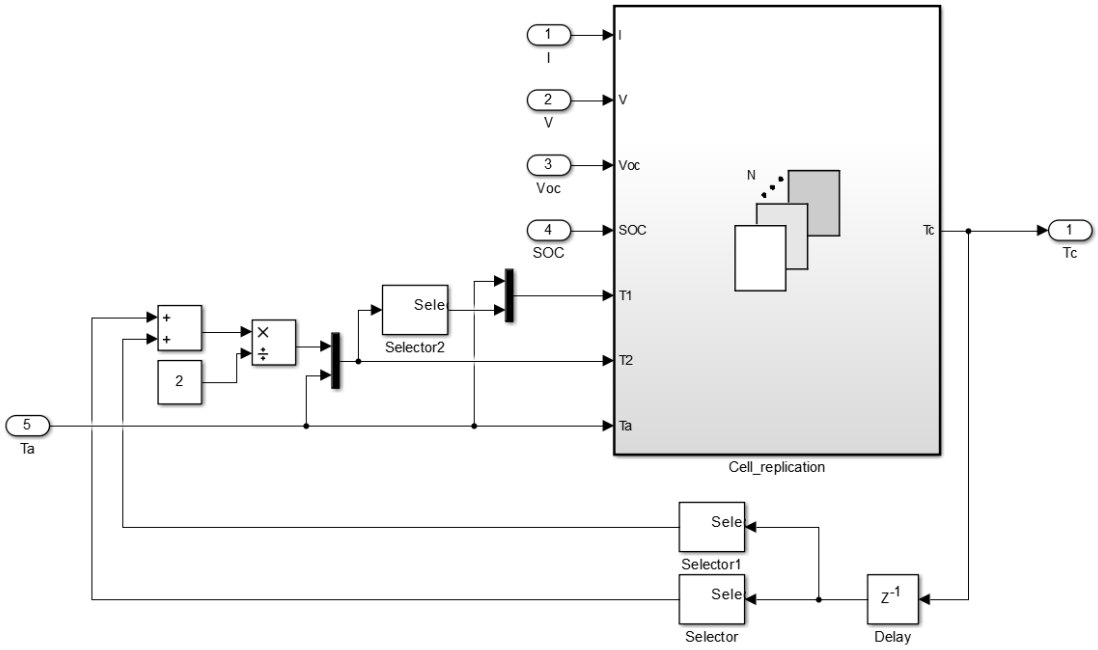


*Figure 24 Simulink string thermal model.*

In order to test the STM, we tried to simulate a 12 series-connected cells battery, by using as input the same current profile seen in Figure 20. The used model is shown in Figure 25.

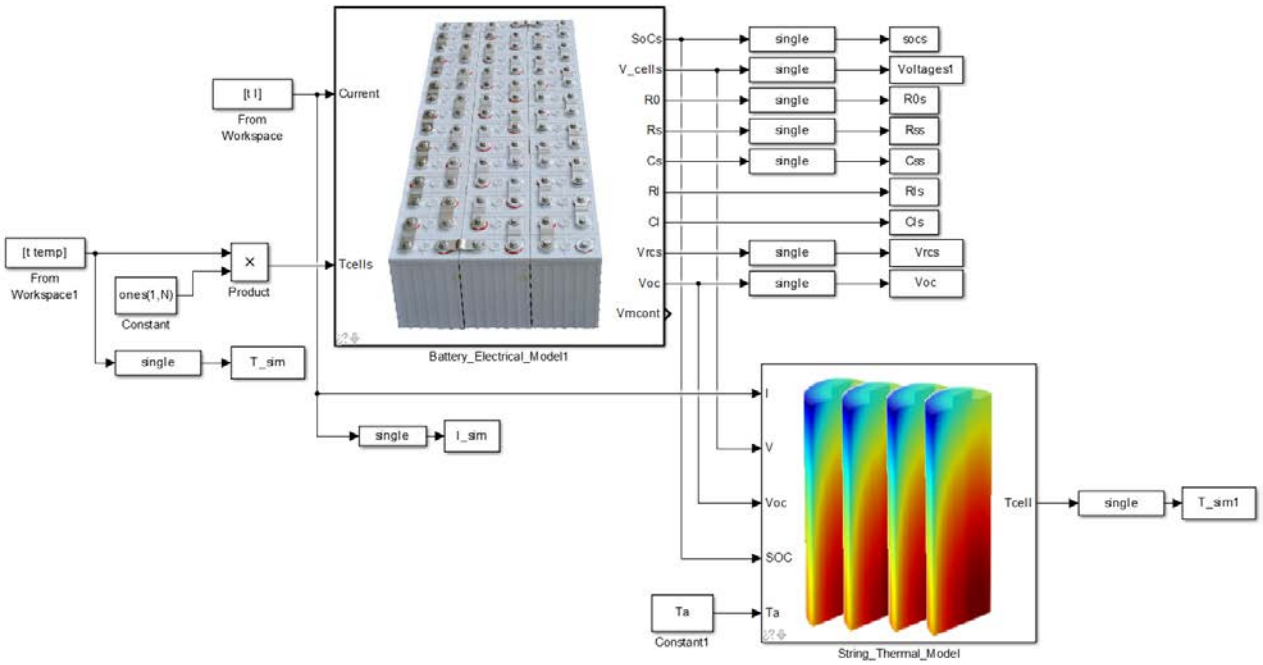


*Figure 25 Simulink model of a 12 cells battery.*

The simulation results are shown in Figure 26 and in Figure 27. The used thermal parameters are the same of those shown in section 5.2. We can see that the first six cells behave in the same manner of the other six, *i.e.*, the temperature distribution in a module is symmetrical if the cells are equal. Inside the module, the temperature is higher than in the ends because of the lower heat exchange with the environment.

The software spends 138.5 seconds to simulate this battery model. Therefore, the simulation time is 9.32 times greater than a single cell model simulation.

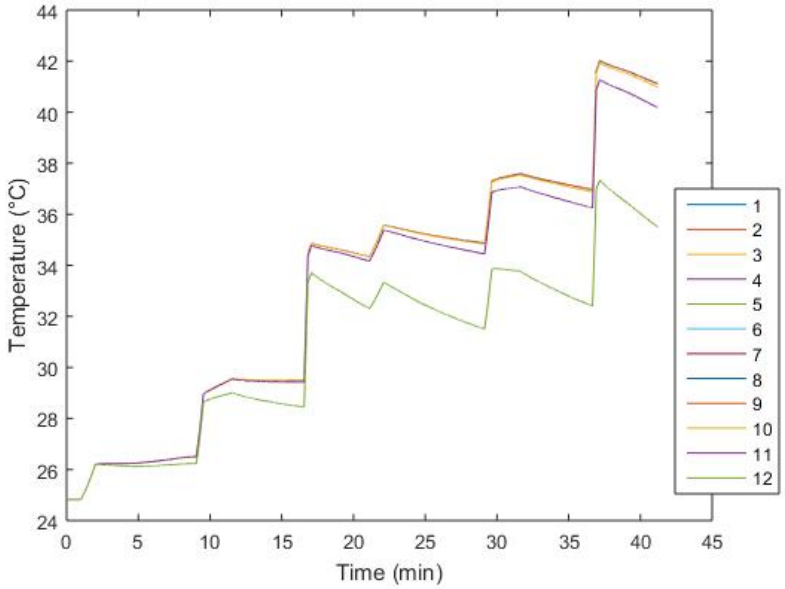


*Figure 26 Temperature of the 12 cells in the simulated battery.*

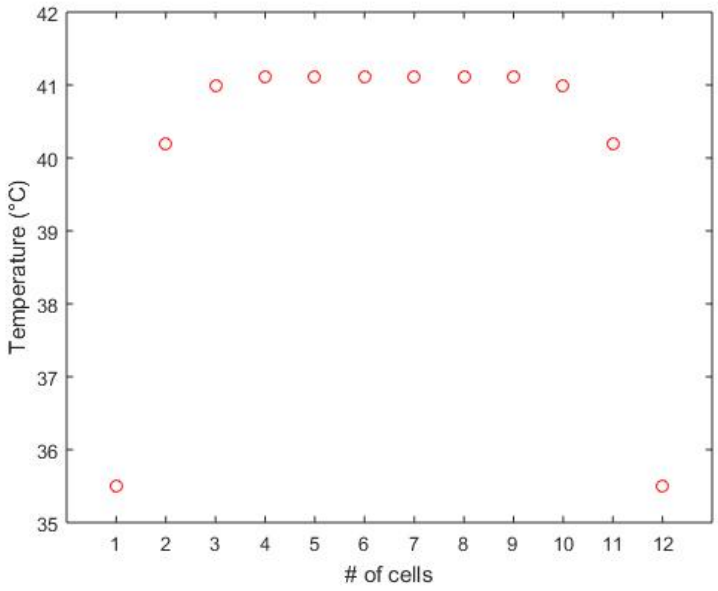


*Figure 27 Temperatures distribution inside the module.*

## 6.3 Battery Thermal Model (BTM)

The Battery Thermal Model (BTM) works in the same way of the Electrical Battery Model. In fact, it replies the STM in order to simulate the behaviour of the battery.

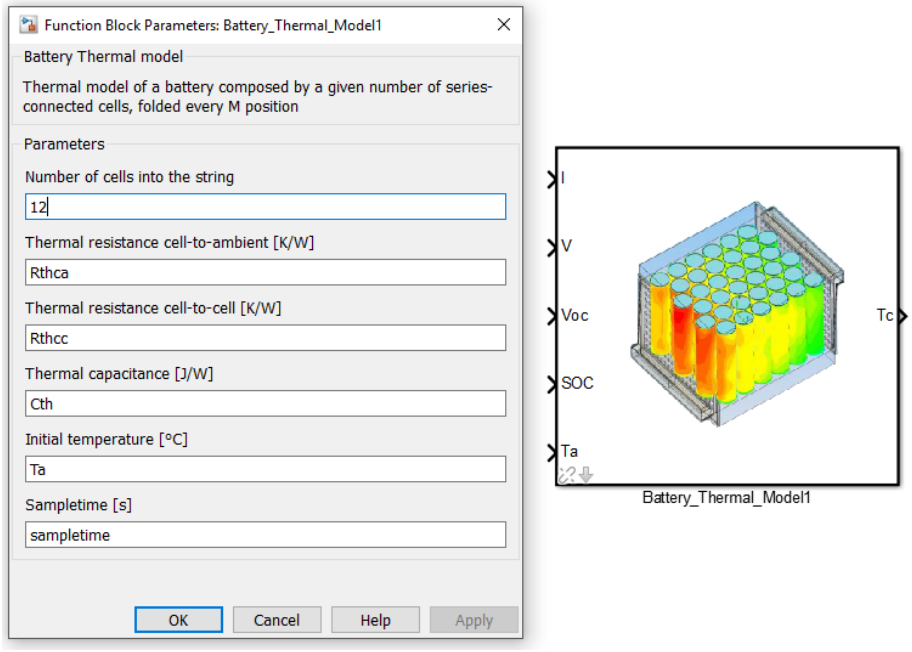


*Figure 28 Battery thermal model and its setup window.*

Over the thermal parameters, you have to specify the number of cells in each string (Figure 28). For instance, for a 24-cells battery, it will be composed by two strings if we put this parameter equal to 12. The thermal parameters have to be set before the simulation. Remember also to load the file that contains the LUT where the cell's entropy is stored:

```
load entropy
Rthca = 70.7228;
```

```
Rthcc = 31.0949;
Cth = 59.4476;
Ta = 24.833;
```

This model has been simulated and the results are shown in Figure 29, Figure 30 and Figure 31. We can see that the two modules behave in the same way. This because of the separation of the modules in the battery pack. In fact, modules are thermally isolated each other.

The simulation spends 273.11 seconds. Therefore, the simulation time is 18.38 times greater than a single cell model simulation.

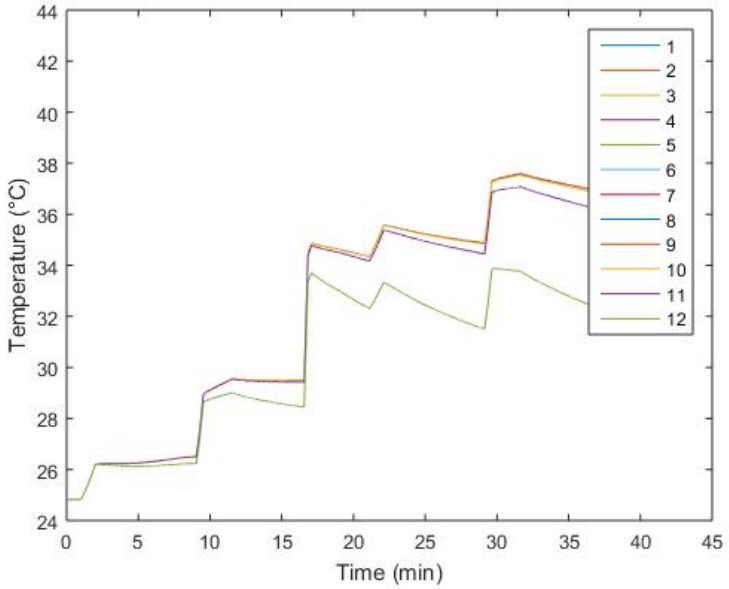


*Figure 29 Temperature of the 12 cells in the first string.*

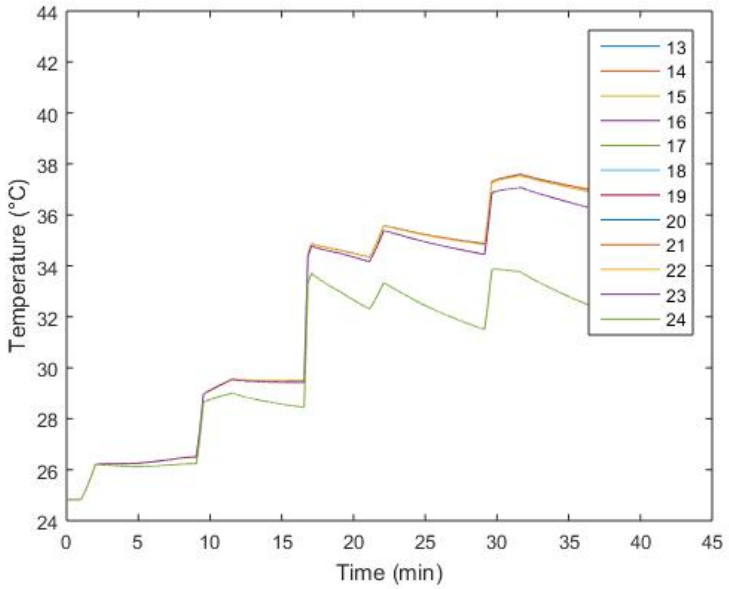


*Figure 30Temperature of the 12 cells in the second string.*

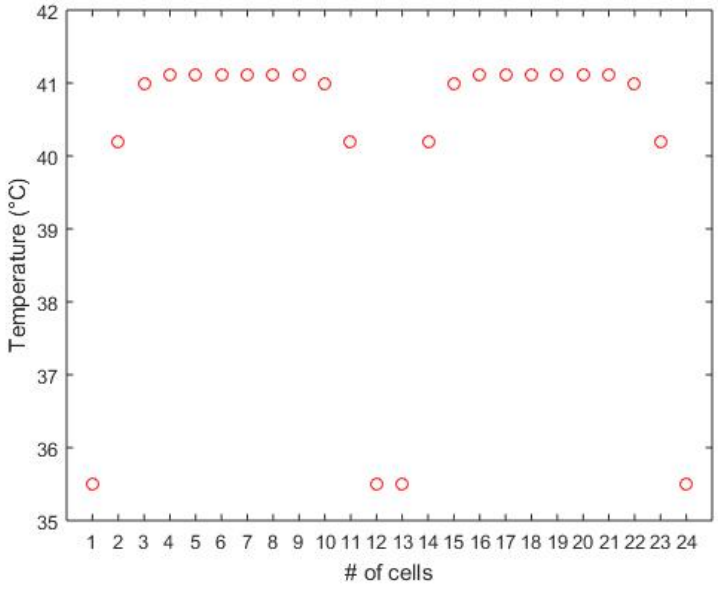


*Figure 31 Temperatures distribution inside the two modules.*

# 7 Sensor model

The estimators use the measurable physical quantities, *i.e.*, current and terminals voltage, of the cell in order to give an estimation of its internal state. In a real application, the measurement are performed by an acquisition system composed by current and voltage sensors and one or more analog-to-digital converters (ADC) that convert the acquired analog signals in digital values. This process introduces a measurements noise. The estimators must be able to execute a correct estimation even in presence of these perturbations. In order to develop reliable algorithms and to test their capability to work in a real system, a model of the sensors has been developed (Figure 32).

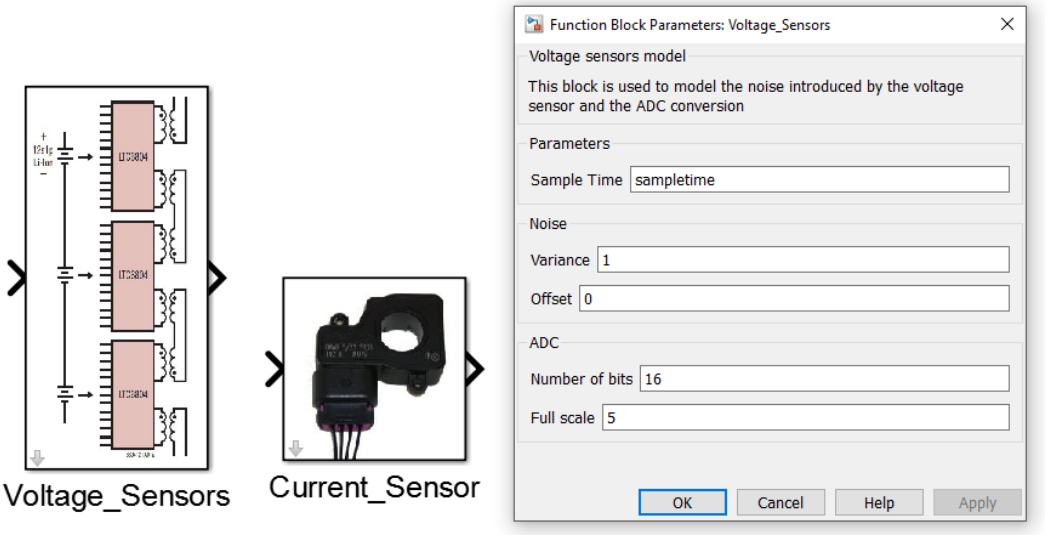


*Figure 32 Simulink model of the voltage and current sensors, together to the voltage sensor's setup window (equal to the one of the current sensor).*

This model uses as input the current $I$ and the cell voltages $V_i$ generated by the battery model and provides as output the measured current $\hat{I}$ and voltages $\widehat{V}_i$, as shown in Figure 4. Both current and voltage sensors are modelled in the same way, as shown in Figure 33 and in Figure 34. They introduce measurement noise and offset errors. The value of these errors are settable to simulate systems with different specifications. Furthermore, the acquisition system block is able to model the quantization error, *i.e.*, the difference between the input signal value and its quantized value. This error depends on the number of bits of the ADC and on its input range. These are editable parameter of the acquisition system. In particular, the current sensor input range is different from the voltage one. In fact, in this application the voltage is a quantity included in the cell operating voltage range, which is always greater than zero. On the contrary, the current ADC has a large bipolar range.

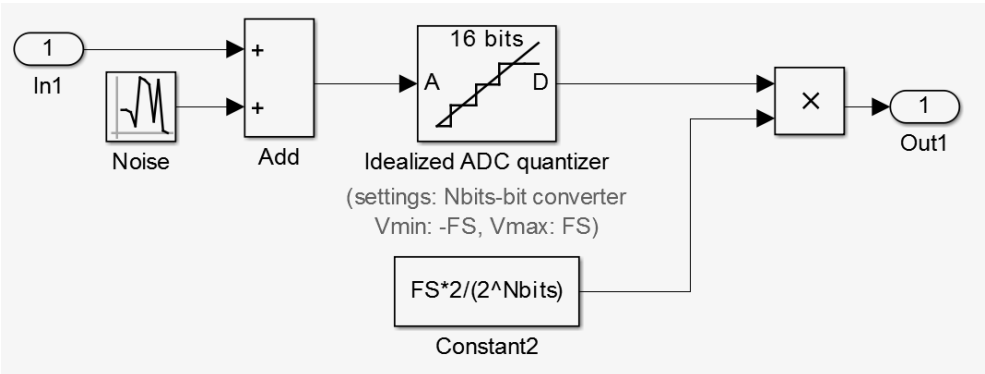


*Figure 33 Current sensor model.*

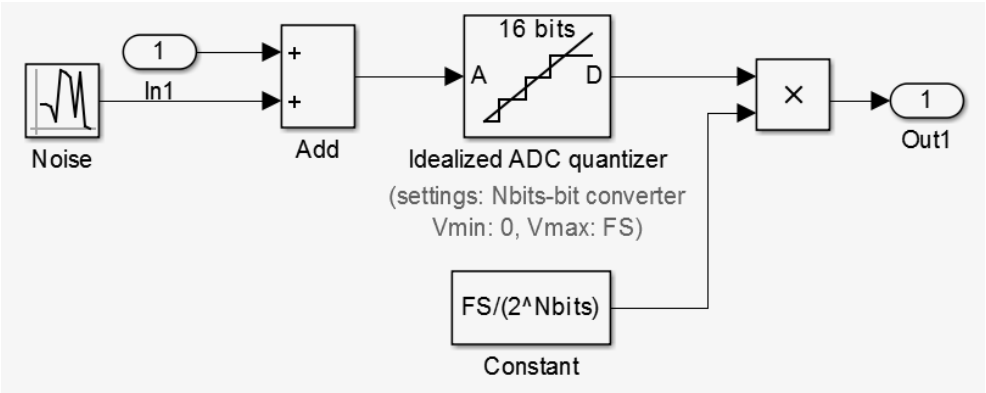


*Figure 34 Voltages sensor model.*

The functioning of these two sensors is shown in Figure 35 and Figure 36. The parameter used in this simulation are the following:

```
% Current sensor
CurrNbits = 16; % Number of bits of the ADC
CurrFS = 100; % Full scale
CurrVar = 0.001; % Noise Variance
CurrOff = 0.2; % Offset

% Voltage sensor
VoltNbits = 16; % Number of bits of the ADC
VoltFS = 5; % Full scale
VoltVar = 0.0001; % Noise Variance
VoltOff = 0; % Offset
```

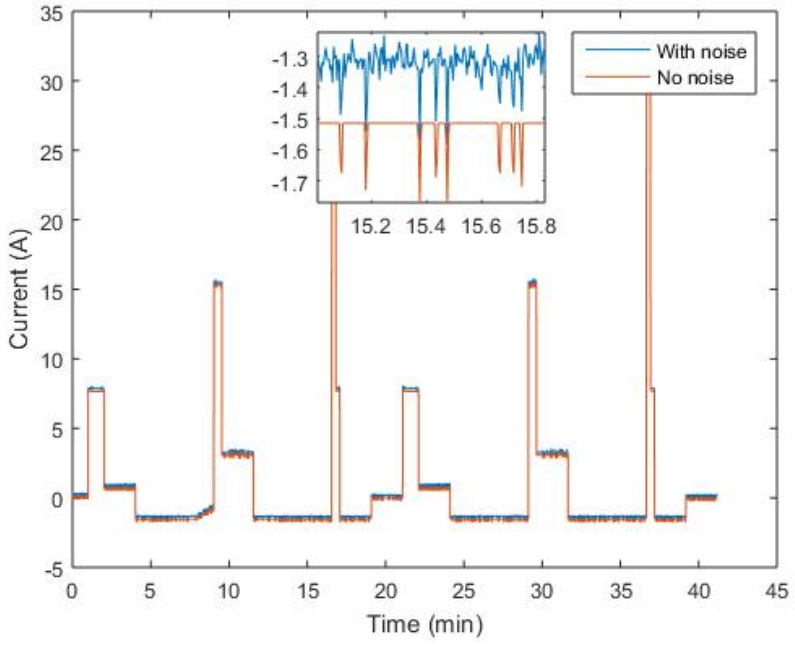


*Figure 35 Comparison between the current signal with and without noise.*

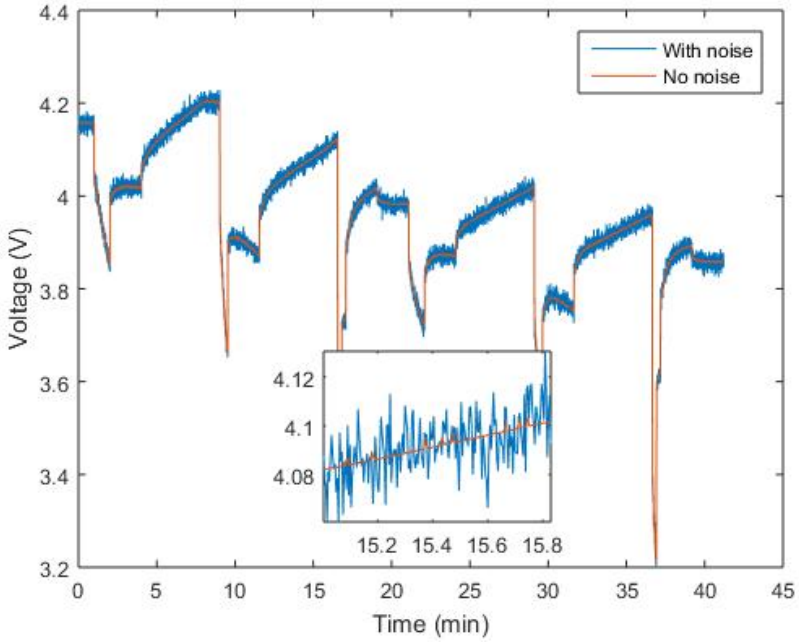


*Figure 36 Comparison between the voltage signal with and without noise.*

# 8 Hardware validation and algorithm performance analysis

The implemented simulation platform is used to check the performance of an algorithm for SOC estimation. However, it can be also used to test many BMS functions.

The algorithm performance regarding SOC estimation can be analysed by comparing the SOC provided as output from the battery model and the one estimated by the algorithm.

The output of the hardware-implemented algorithm can be also compared with the results of the software implementation in order to validate the hardware modules. The timing of both implementations must be the same in order to compare their behaviour.

In Figure 37, we can see the implementation of the Adaptive Mix Algorithm (AMA), for SOC estimation, in software (AMA-CPU) and in hardware (AMA-FPGA). The interfaces with the two implementations are the same and the signals timing is managed by the InputBlock. The results are compared and visualized by the Scopes block.

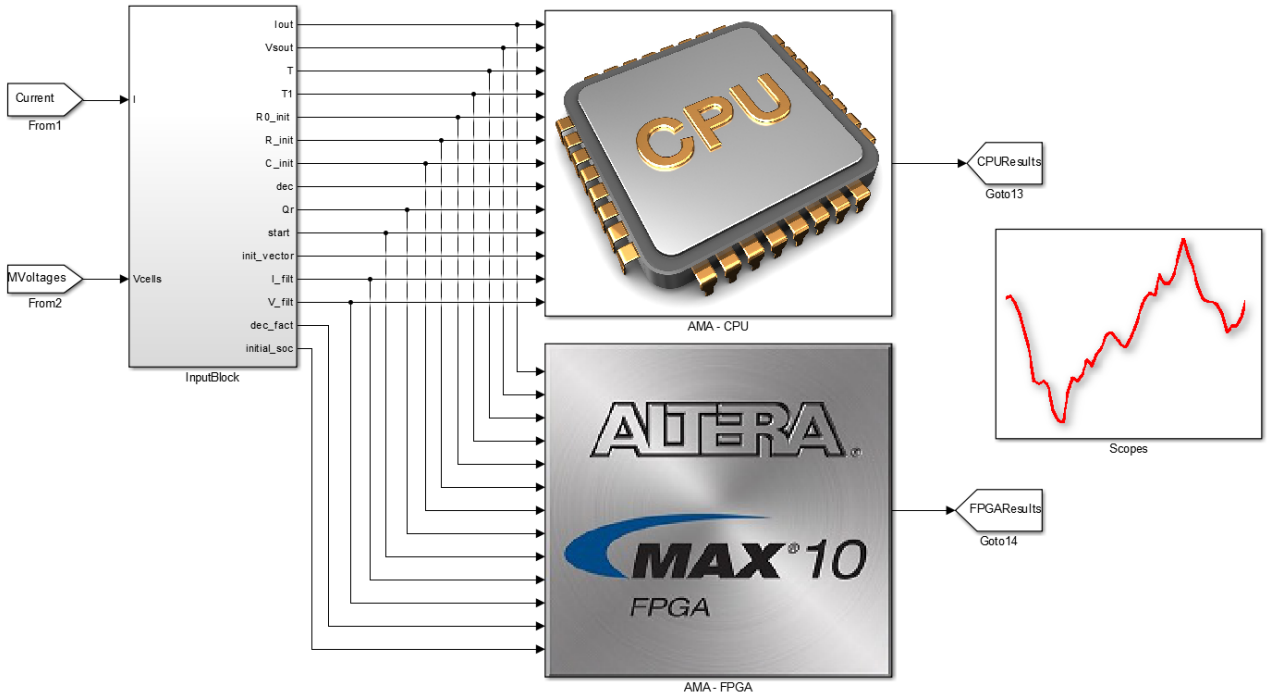


*Figure 37 Hardware validation Simulink blocks.*

The two main blocks have almost the same inputs and outputs. They are based on the Simulink S-function block, which is used to execute Matlab functions. Please refer to the guide "Write Level-2 MATLAB S-Functions" to understand how to write an S-Function.

In the following sections, we describe how to simulate the same behaviour of the hardware algorithm in the Simulink environment and how the PC can send and receive data from the FPGA.

## 8.1 Hardware-in-the-Loop Platform files

The simulation platform main folder contains the following files and subfolder:

```
|HiL main folder → |Model → |Cycledata
|                  |        Battery_unipi.slx
|                  |        cell_model_params.mat
|                  |        entropy.mat
|                  |Sim_Model → CompleteModel.slx
```

```
|                    |                CompleteModelHiL.slx
|                    |Functions
|                    |FPGA_files
|                     ArchData.m
|                     Launcher.m
|                     SetCreator.m
|                     OffilineEstimation.m
```

Launcher.m is the main script. It executes the main operations that allow the simulation to be run. These operations are the following:

1. Loading of the workspace.
2. Selection of the driving cycle.
3. Execution of the SetCreator.m script. It creates the Settings structure, in which there are all the parameters which are needed for simulating the model.
4. Modification of some Setting's fields in order to set the simulation constrains and save the structure.
5. Simulation of the model. The user can decide if using the standalone model or the HiL platform by connecting the FPGA to the PC. The standalone model can be used to analyse the battery behaviour in different conditions.
6. Saving of the simulation results structure.
7. Execution of the ArchData.m script. It creates a new folder and save the simulation structure in it.

## 8.2 The Adaptive Mix Algorithm

The Adaptive Mix Algorithm (AMA) is a technique based on the Mix Algorithm for SOC estimation[2]. It is a model-based method (see Figure 38) that compares the model output voltage to the measured cell voltage in order to correct the estimation of the SOC state variable, computed integrating the battery current[3].

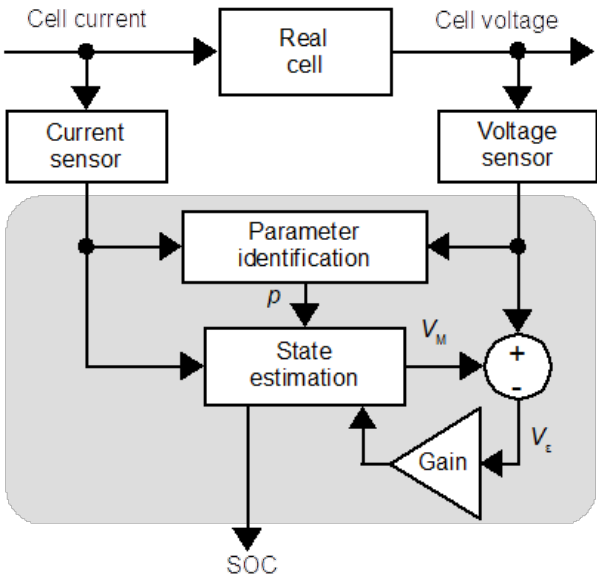


*Figure 38 Functioning diagram of a model-based algorithm.*

[2] F. Codeca, S. M. Savaresi, and G. Rizzoni, "On battery State of Charge estimation: A new mixed algorithm," in 2008 IEEE Int. Conf. Control Appl. IEEE, Sep 2008, pp. 102–107.
[3] F. Baronti, R. Roncella, R. Saletti, and W. Zamboni, "FPGA Implementation of the Mix Algorithm for State-of-Charge Estimation of Lithium-Ion Batteries," in IECON 2014 - 40th Annu. Conf. IEEE Ind. Electron. Soc., 2014, pp. 5641–5646.

The ECM used in the algorithm is the same of the one in Figure 11 with only one RC group. This reduces the computational complexity, while preserving a good accuracy, especially in applications with fast transients. The ECM parameters $[R_0; R_1; C_1]$ are identified online by using the Moving Window Least Squares (MWLS) method, applied to the AutoRegressive eXogenous (ARX) representation of the ECM[4]. It applies the LS method to a set of current and voltage samples in a given time window, which is periodically shifted in time[5].

## 8.3 Software battery state estimator

The AMA-CPU block uses an S-function to run the AMA algorithm. In this way, we can define the algorithm sample time, the inputs, the outputs and others parameters of a generic Simulink block. The outputs are computed by running the AMA developed using the MATLAB script language. The execution time of a single sample computation is of 0.106 seconds. This is obtained by simulating only the model and AMA-CPU block for 601 samples (complete simulation time equal to 63.97 seconds).

## 8.4 Hardware battery state estimator

This block uses an S-function to establish a communication with an Intel FPGA device, which belongs to the Max 10 family[6] in this work.

The communication is based on the System Console package, provided by Intel (you have to run Matlab by using the DSP Builder tool). This API is a simple set of commands which allows real-time interaction with the FPGA design running on your target platform. The interaction is done using memory mapped transactions initiated from the MATLAB environment. These transactions are captured by a dedicated Avalon® Memory-Mapped (Avalon-MM) Master interface connected to an Avalon-MM Slave interface inside your design, and compiled into an FPGA. The MATLAB API can be used in multiple tasks while designing, evaluating, and verifying designs. It can be used to configure and tune an FPGA design parameter in runtime similar to tuning data pipe gains and modifying filter parameters. It can also be used to monitor different statuses by reading specific registers in runtime. The most powerful capability of the MATLAB API can be seen in the HiL approach. It allows you to generate complex stimulus datasets on the host PC and download to the FPGA target in real time. This is convenient because it is rarely possible to create such a flexible and sophisticated stimulus on a target platform with minimal programming effort. A script example is shown in Table 6.

```
SystemConsole.refreshMasters; %Investigate available targets
M = SystemConsole.openMaster(1); %Creates connection with FPGA target
%%%%%%%% User Application %%%%%%%%%%%%
. . . .
M.write('uint32',write_address,data); %Send data to FPGA target
. . . .
data = M.read('uint32',read_address,size); %Read data from FPGA target
. . . .
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
M.close; %Terminates connection to FPGA target
```

*Table 6 MATLAB API with System Console Script Example.*

The battery estimator is implemented using the Intel design flow as illustrated in Figure 39. Hardware design starts in DSP Builder where the algorithm is described in Simulink models and synthesized to low-level

[4] H. Rahimi-Eichi, F. Baronti, and M.-Y. Chow, "Modeling and online parameter identification of Li-Polymer battery cells for SOC estimation," in 2012 IEEE Int. Symp. Ind. Electron. IEEE, May 2012, pp. 1336–1341.

[5] F. Baronti, W. Zamboni, N. Femia, H. Rahimi-Eichi, R. Roncella, S. Rosi, R. Saletti, and M.-Y. Chow, "Parameter identification of Li-Po batteries in electric vehicles: A comparative study," in 2013 IEEE Int. Symp. Ind. Electron. IEEE, May 2013, pp. 1–7.

[6] Note: A common error with this device family is "Error (14703): Invalid internal configuration mode for design with pre-initialized eram". To fix this issue, set the following option: Assignments -> Device -> Device and Pin Options -> Configuration -> Configuration Mode: Single uncompressed image with Memory Initialization.

hardware description. The design is optimized for performance and resource by applying pipelining, time division multiplexing/folding and customizing precision. In the Qsys system integration tool, the generated hardware components are connected to other components in the system, including a Nios II 32-bit soft processor, JTAG and memory. A complete design is synthesized and programmed for the target FPGA using Quartus II design software.

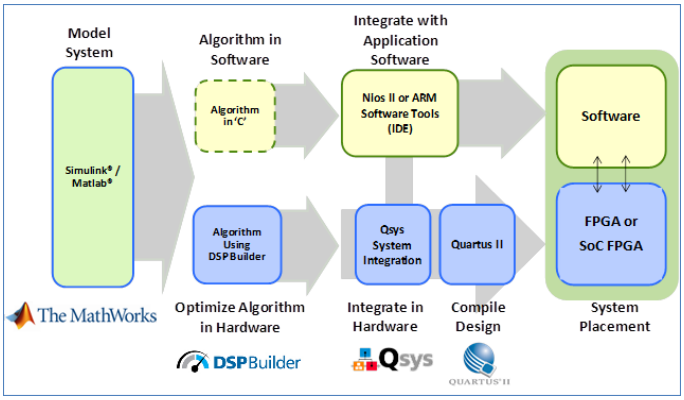


*Figure 39 Intel FPGA design flow.*

The hardware system is composed by two parts: the MM master unit and the hardware estimator, which is a MM slave unit. In the implemented design, the MM master interface is a JTAG Master unit connected to the host pc via an USB cable. It is used to access the registers of the hardware estimator which uses the AMA to predict the SOC of twelve series-connected cells. The interconnections between the JTAG master and the developed estimator are shown in Figure 40.

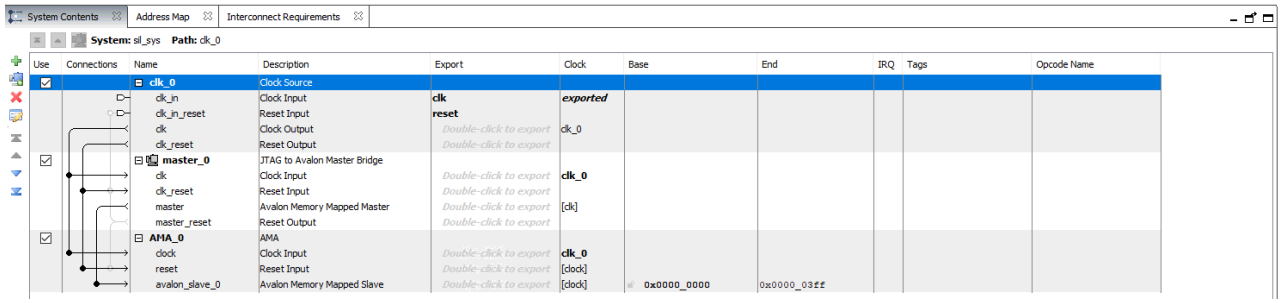


*Figure 40 System built in the Qsys tool.*

The AMA and hardware module has been implemented for a low cost Intel MAX® 10 FPGA (10M50DAF484C6GES device). The FPGA resource usage and performance are shown in Table 7.

| | |
|---|---|
| Logic Elements | 38 k/50 k (76%) |
| 9-bit Multipliers | 219/288 (76%) |
| Memory bits | 170 Kb/1638 Kb (10%) |
| Execution time | 34 µs (@ 100 MHz) |

*Table 7 Estimator resource usage and performance.*

The estimator fits in the chosen device. The very short execution time allows the same module to be used for estimating a large number of cells in a time multiplexing fashion. The number of cells affects the required memory inside the AMA module (the memory bits figure reported in the table refers to 12 cells).

For the system integrated in the HiL platform, the execution time of a single sample computation is of 0.106 seconds. This is obtained by simulating only the model and AMA-FPGA block for 601 samples (complete simulation time equal to 63.56 seconds which takes also into account all the operation of data management). A more accurate estimation has been computed by measuring the time of each iteration; this analysis brings to an execution time of **about 0.09 seconds**.

The user can program the MAX® 10 FPGA by using the provided *sof* file and the programmer that can be downloaded from the official site (www.altera.com).

## 8.5 Validation of the hardware implementation

The developed HiL platform, whose real photo is shown in Figure 41, has been used to validate the hardware implementations of the AMA estimator and to assess their performance. To this end, the Nissan Leaf traction battery has been simulated. It consists of 96 series-connected NMC cells with a capacity of 66.2Ah, reaching a nominal voltage of 355.2V. From a thermal point of view, there are 8 modules of 12 series-connected cells.

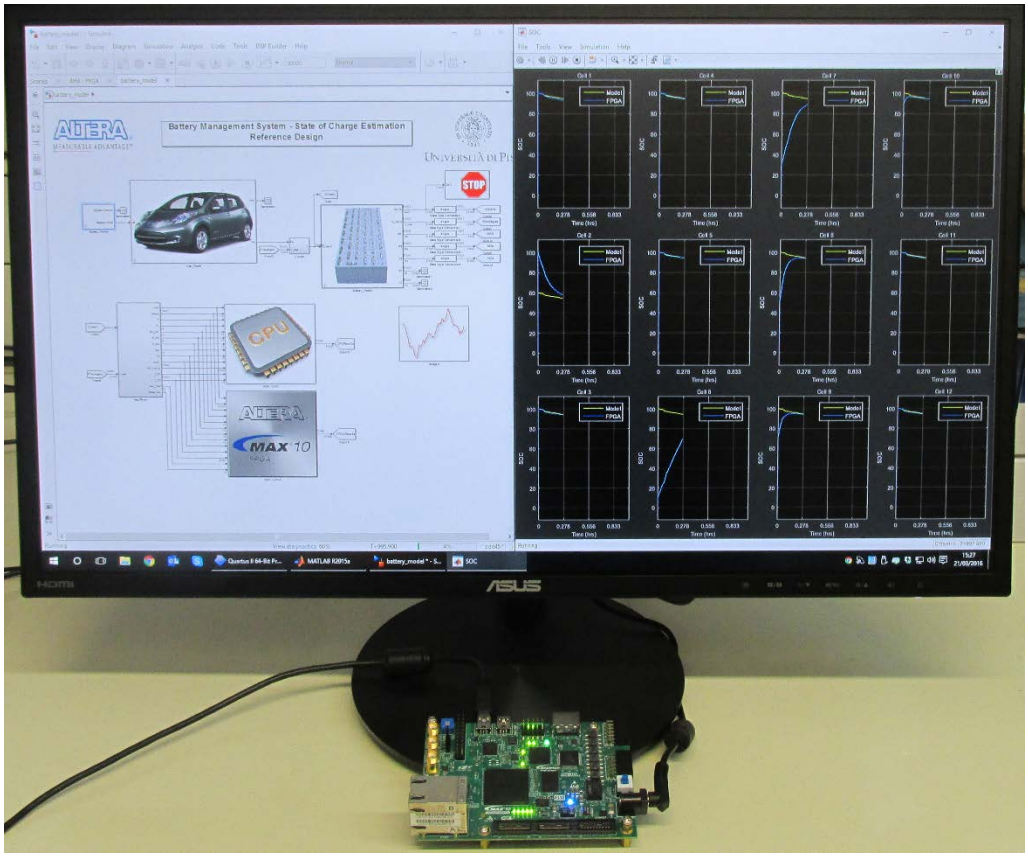


*Figure 41 Hardware-in-the-Loop platform in action.*

The length of the moving window in the AMA is set to 90 seconds.

The validation of the estimator has been executed by comparing the FPGA results and those obtained from the software-implemented algorithms.

## 8.6 Performance assessing

Various scenarios has been simulated in order to assess the algorithm performance. All the executed simulations consist of the repetition of a driving schedule, listed in Table 1, until the battery reaches a state of charge equal to 20%, starting from a SOC of 80%.

The algorithm has been tested in case of noiseless sensors, in which the acquisition system introduces only the quantization error. Here, the simulations have been carried out with identical cells, all starting from the

same SOC. For this reason, all the battery modules behave in exactly the same way. Consequently the estimators, which are capable of handling up to 12 cells, are used to analyse all the cells of a single module.

The maximum and rms SOC errors for all driving schedules are reported in Table 8. The temperature of all the cells has been calculated by the thermal model, starting from 10 °C, which is also the temperature of the ambience.

| Driving Schedule | SOC Max error (%) | SOC rms error (%) |
|---|---|---|
| WLTP class 3 | 6.0 | 3.0 |
| UDDS | 4.4 | 2.2 |
| HWFET | 5.8 | 3.4 |
| FTP | 4.8 | 2.3 |
| IM240 | 5.3 | 2.8 |
| SFTP US06 | 9.0 | 5.0 |
| SFTP SC03 | 5.0 | 2.5 |
| NYCC | 3.2 | 1.3 |
| LA92 | 6.0 | 3.1 |
| LA92 short | 6.0 | 3.1 |
| EUDC | 5.7 | 3.1 |
| NEDC | 4.3 | 2.2 |
| ECE R15 | 2.8 | 1.3 |
| ArtUrban | 4.1 | 1.9 |
| ArtRoad | 6.1 | 3.4 |
| ArtMw130 | 7.6 | 4.3 |
| ArtMw150 | 8.3 | 5.1 |
| J1015 | 3.8 | 1.8 |

*Table 8 SOC estimation error with noiseless measurement.*

In real applications, the algorithms' inputs are measured by an acquisition system that introduces measurement noise. The developed HiL platform can be also used to analyse how the measurement precision influences SOC estimation, in order to allow the algorithms to be suitably tuned. The application is able to simulate independently the current and voltages signals acquisition. In fact, the two quantities have a different acquisition system that causes different measurement accuracy. The voltage can be directly acquired from an ADC with a small input range (*e.g.*, from 0 to 5V). This allow the system to obtain a very low error. On the contrary, the current is firstly converted in voltage by a current sensor, such as Hall effect sensor or resistive shunt, and then acquired by an ADC. The input range must cover a large interval of current values (*e.g.*, from 0 to hundreds of ampere). These facts introduce relevant errors.

To tests the algorithms performance in these conditions we reproduced an acquisition system that introduce a Gaussian noise with a variance of 4 $A^2$ on current and with a variance of 2.5 x $10^{-5}$ $V^2$ on voltage measurement. Both ADCs have a resolution of 16 bits. The estimation errors obtained in these conditions for every driving schedules are reported in Table 9.

| Driving Schedule | SOC Max error (%) | SOC rms error (%) |
|---|---|---|
| WLTP class 3 | 7.1 | 3.8 |
| UDDS | 5.7 | 3.0 |
| HWFET | 8.8 | 5.2 |
| FTP | 6.0 | 3.0 |
| IM240 | 7.2 | 3.9 |
| SFTP US06 | 9.8 | 5.5 |
| SFTP SC03 | 6.5 | 3.3 |
| NYCC | 4.0 | 1.7 |
| LA92 | 7.1 | 3.8 |
| LA92 short | 7.2 | 3.9 |
| EUDC | 8.3 | 4.6 |
| NEDC | 6.0 | 3.1 |
| ECE R15 | 3.9 | 1.8 |
| ArtUrban | 5.4 | 2.5 |
| ArtRoad | 7.6 | 4.2 |
| ArtMw130 | 10.7 | 6.1 |
| ArtMw150 | 10.6 | 6.6 |
| J1015 | 5.2 | 2.5 |

*Table 9 SOC estimation error with noisy measurement.*

# 9 Revision History

| Date | Changes |
|---|---|
| | Initial Relase |