Xavier Pérez Simó

Master's final thesis

Li-ion battery cell emulator based on a buck converter

Master's degree on Electric Vehicle Technologies

UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2024

Supervisor: Carlos Olalla Martínez

Outline

Figures list

Table list

1 Introduction

Throughout human history the mobility between towns, big cities or even among countries is being evolved constantly. The fuel reserves as well as the polluting gases from the combustions engines is a present problem. The automation manufactures have been innovating to get ahead a next vehicle generation. The electrical vehicles have better efficiency and zero emissions in comparison of the combustion engine.

The electric vehicle quality is focused on design a suitable drivetrain model according to the vehicle's specifications. The electrical drivetrain is based on the Energy Storage System (ESS) whose architecture has the battery cells, Battery Management Systems (BMS) and DC-DC bus converter. The following figure is an example where that components are in the vehicle.

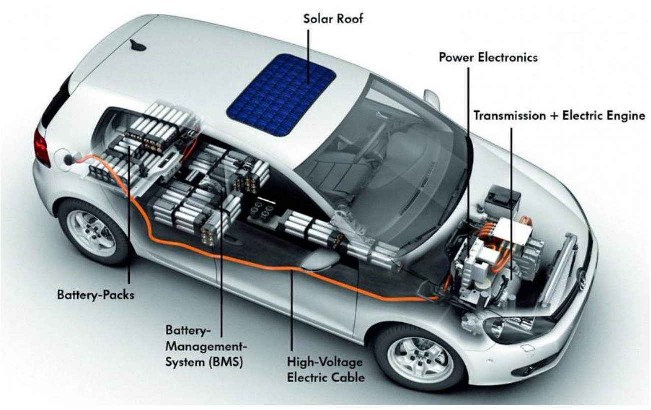


Figure 1. Electric vehicle key components [15]

The battery technology is one of the main electrical components which must be well selected to achieve an optimum vehicle model. Going back to the origins has been considered interesting to understand the principles of battery cells. Alessandro Volta invented the first battery cell in 1799. This experiment served to discover the main parts that compose a battery cell. Firstly, there are the electrodes whose function is to flow current which leaves or returns to the electrolyte. In that case, there was carbon and zinc strips that were placed inside the electrolyte solution. Secondly, the electrolyte is a solution that acted as a path between the electrodes. The electrolyte could be a salt, an acid, or an alkaline solution. In that case, the electrolyte was a liquid form. Thirdly, there was the container which could be constructed for different materials to hold or to contain the electrolyte. The container was also used to mount the electrodes. In the galvanic cell the container must be constructed of a material that will not be affected by the electrolyte. The parts described above are represented on the following figure schematic.

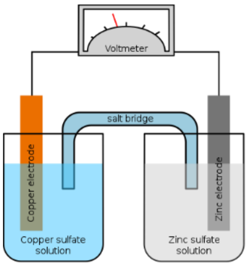


Figure 2. Schematic diagram of a Galvanic cell [18]

The load connected externally provokes that electrons flow through a difference in potential across the electrodes from the cathode (negative electrode), through the external conductor to the anode (positive electrode). The voltage levels depend on multiple factors such as, the electrode materials or the electrolyte composition. The current delivered depends on the circuit resistance, also including the cell resistance itself. This internal cell resistance depends on the electrodes size, the distance between them in the electrolyte, and the electrolyte resistance. To define a standard battery cell voltage which depending on the composition materials, a hydrogen electrode has been taken as a standard reference potential electrode.

The battery cell trajectory from Alessandro Volta has been managed to nowadays different manufactures which have reached different capacities and materials. The typical batteries used in electric vehicles are lithium-ion batteries because they have high energy density, longer cycle life in comparison of other battery chemistries. The battery cells could also be modeled to emulate their behavior on a prototype representation. Battery emulation is a solution to model and define the suitable battery parameters for each electric vehicle model. This Battery emulation tooling is used for a BMS development, to determine the battery life, or even for production test validations. One of the advantages to emulate batteries is that instantly simulate any state of charge (SOC) versus manually draining a battery to the desired level.

The objective of this master thesis is to develop a battery cell emulator from the design to the prototype implementation. The prototype designed in this master thesis consists of a DC-DC converter which emulates the behavior of a battery cell. This first release is a unidirectional converter which only allows to discharge the battery due to simplified design. Firstly, the battery cell model has been defined under a low current profile. Secondly, the converter has been designed to be used as a cell emulator. Finally, the prototype has been implemented on a customized PCB for the power part, and demo studio board has been used for de control part.

The topic inspiration comes from commercial battery cell emulator used to test the BMS in my work environment. This is COMENSO which offers a battery pack simulation for testing purpose. The goal of this master thesis is based on create an equivalent prototype from that supplier to emulate a battery cell.

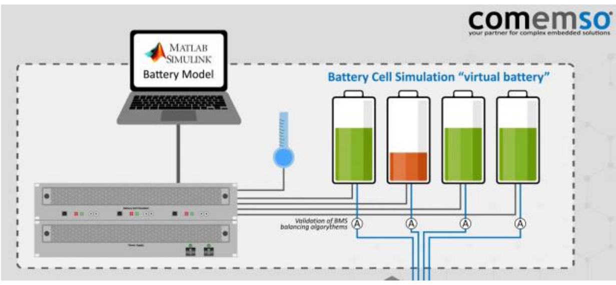


Figure 3. Battery cell simulator block diagram from COMENSO [17]

The block diagram above is the commercial tool model which is ready to connect multiple cells in parallel. However, the idea of this thesis is to achieve the implementation of only one battery cell in unidirectional site. These cells are controlled through Matlab or Canoe tooling, but the prototype idea here is to control the cells at C language level for the microcontroller.

2 Battery cell Model

The purpose of this chapter is to define first a method to model a battery cell for SOC estimation. This electrical cell model can be applied to any cell type and chemistry. Then, a battery cell has been selected for this project. Finally, a buck converter has been designed to emulate the battery cell behavior.

2.1 Electrical cell model

The SOC is the charge level expressed in percentage of an electric battery relative to its capacity. To define the SOC estimation a physical model is needed to describe the battery cell behavior. Although, the electrochemical model would perfectly describe the cell behavior, the model has not been used in this project due to difficulty to define and perform the test. The cell behavior has been described in terms of electrical elements, defining the electrical magnitudes, for instance, a capacitor has been used to represent the total capacity. The SOC in this model is represented through a capacitor voltage (V_{SOC}). The V_{SOC} range is between 0 V (0% SOC) until 1 V (100% SOC).

The dynamic elements have not been considered on this first model approximation. So, the battery voltage will be equal to the Open Circuit Voltage (V_{OCV}). The relation between V_{OCV} and V_{SOC} is a particular characteristic of each type of cell which represents the battery's performance under specific discharge conditions and operating modes. The following schematic represents this first model approximation.

$$V_{SOC} = -\frac{1}{C_{SOC}} \cdot \int_0^T ibat(t)dt$$

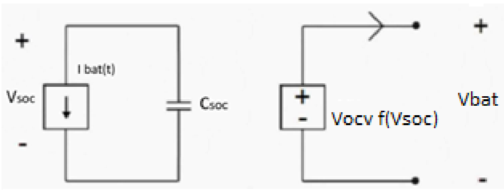


Figure 4. First approximation battery cell model

The charge [Q] defined in Coulombs is a physical magnitude to define the cell capacity, but the manufacturers usually used mAh or Ah. Assuming the SAMSUNG INR 18650-20R has a capacity of 2000mAh, and the maximum V_{SOC} is 1 V, the C_{SOC} has been calculated as follow:

$$\int_0^{T_{end}} ibat(t)dt = Q_{total} = C_{SOC} = 2000\, mAh \cdot \frac{3600s}{1h} = 7200F$$

The static profile with low current has been considered in the first approximation. Then, dynamic elements have been considered for the second approximation. The series drop resistor has been modeled to achieve a more accurate battery cell model. This series resistor represents the voltage drop on cell terminals when the current is flowing to input the cell or output. The drop voltage to discharge o charge the battery cell is different, but in this project only discharge voltage has been considered. The resistor effect is the energy dissipated as heat inside the cell. The following schematic represents the evolution of that battery cell model considering this dynamic effect.

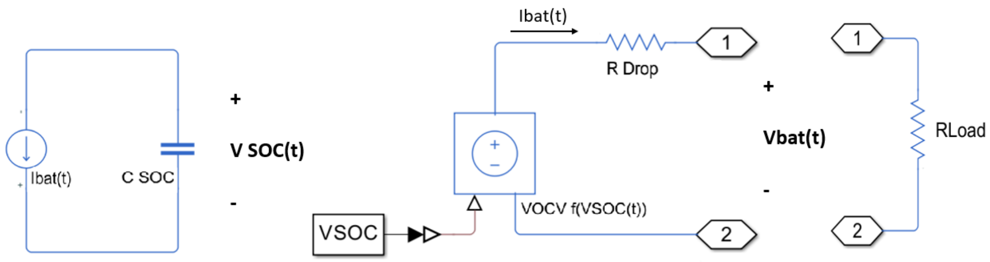


Figure 5. Battery cell theoretical model used in this project.

There are more dynamic elements to define the battery cell model, but they have not modeled in this project. Some examples could be the hysteresis or differential voltage which have been discarded due to a simplified model version.

2.2 Samsung battery cell

A SAMSUNG battery cell, INR 18650-20R, has been modeled in this project. It is a typical cylindrical Li-ion cell characterized for its high-power density with nominal voltage of 3.6 V and capacity of 2 Ah. Looking into the datasheet, some more interesting parameters have been extracted [4].

Battery Cell INR 18650-20R specifications	
Nominal discharge capacity	2000 mAh (0.2 C, 2.5 V discharge cut-off)
Nominal voltage	3.6 V
Max. continuous discharge	22 A(at 25ºC), 60% at 250cycle (Continuous)
Discharge cut-off voltage	2.5 V (End of discharge)
Cell chemistry	LNMC/Graphite

Table 1. Samsung battery cell characteristics

The cell shall be discharged continuously under certain condition to maintain the maximum capacity. This capacity has been measured with discharge current of 400 mA (0.2 C) with 2.5 V cut-off within 1 hour after standard charge, which complying to the minimum capacity of IEC61960 standard. However, the discharge capacity is also measured with the maximum discharge current, for that reason, the cell capacity is reduced to 95% from its total capacity.

	Discharge condition	
Current	0.40 A	20 A
Relative Capacity	100% (2000 mAh)	95%(1900 mAh)

Table 2. Discharge condition limits

Misuse of the battery cell could also provoke its consequences. For instance, the over discharging exceeding the current limit may cause loss of battery performance characteristics. Over discharging also may occur by self-discharge if the battery is left for a very long time without any use.

The Lithium Nickel Manganese Cobalt Oxides (LNMC) is the cathode material of that cell while the graphite is the anode material. These materials characterize the cell as very high specific energy but limited specific power. The battery cell cost is high, because nowadays the cobalt is expensive material. The following figure is the real Samsung battery cell.

Figure 6. Samsung INR 18650-20R

A discharge comparison between two different cells from SAMSUNG has been showed on the next figure. The battery cell used in this project (INR18650-20R) is represented through the red curve, and the other similar battery cell also from SAMSUNG INR18650-20S is represented through the blue curve. The different OCV behavior has been appreciated under the same discharge current conditions of 200mA. Although, both battery cells analyzed have the same capacity of 2000 mAH, they have different OCV discharge curve under the same discharge conditions.

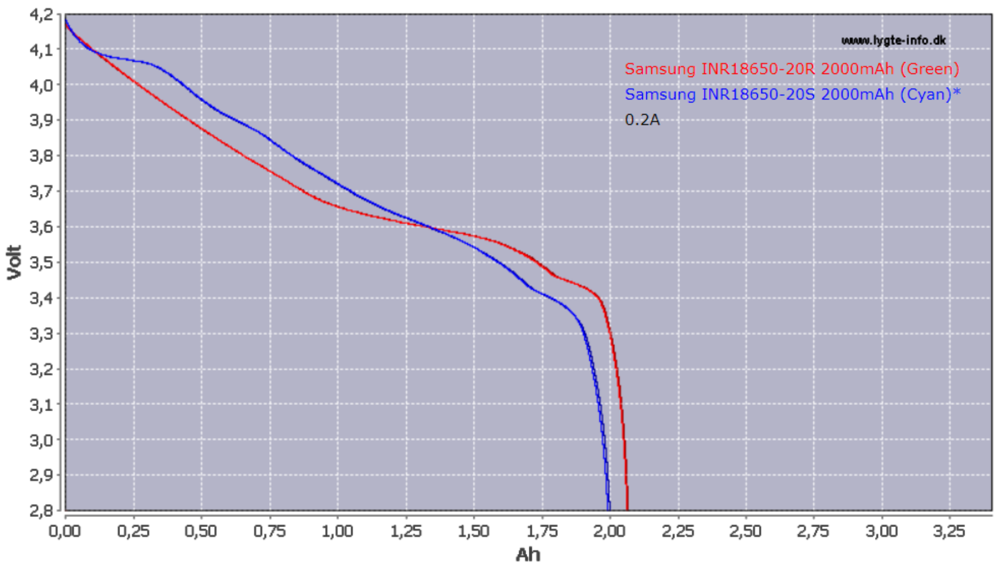


Figure 7. Battery cell discharge comparison [16]

The voltage cell is decreasing during discharge following a nonlinear curve from the maximum voltage cell at 100% of SOC. The SAMSUNG battery cell capacity modeled is 2000 mAh which corresponds at 100% of SOC. The discharge current value determines different discharge profiles. The defined profile with the C-rate which is specified in the following formula:

$$C_{rate} = \frac{discharge\ current\ [A]}{nominal\ capacity\ [Ah]} = \frac{200\ mA}{2000\ mAh} = 0.1C$$

The discharge 0.1 C-rate corresponds a relation between discharging the SAMSUNG battery cell using a low current of 100 mA, and the total capacity of 2000 mAh. So, the C-rate defines the speed at which the cell goes from 100% to 0% under specific constant current. To obtain that profile, the cell must be set in open circuit during long time to obtain an ideal voltage source without any disturbances. The low current 0.1 C used to discharge the cell achieves the cell in equilibrium conditions to obtain the Open Circuit Voltage (OCV).

The "calce" battery research group [5] tested the battery cell INR 18650-20R using several profiles under different conditions. Parameters that affect the battery behavior could be temperature or current profiles which affect the discharging battery in different conditions. So, the battery profile defines the specific conditions to discharge the battery. A standard profile defines the parameters like the amount of energy that battery can store or, the supply energy that battery can discharge over time.

The battery profile "11_5_2015_low current OCV test_SP20-1.xlsx" at 23 ºC [5] has been used in this project. This profile is characterized by low current OCV at 23 ºC. The following figure represents the profile explained before. This profile takes a long time to discharge the battery due to low current profile.

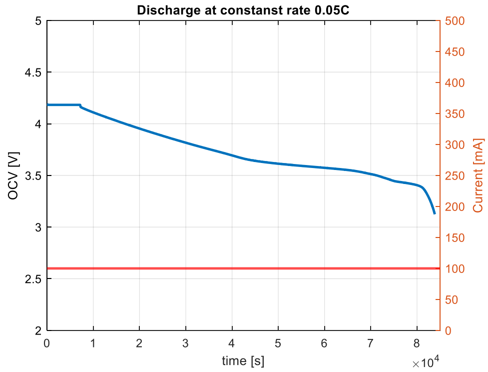


Figure 8. Discharge the battery cell at very low constant current.

The relation between V_{OCV} and V_{SOC} is no linear even though the discharge current is constant. The relation between V_{SOC} and V_{OCV} for that battery cell INR 18650-20R has been shown on the next figure.

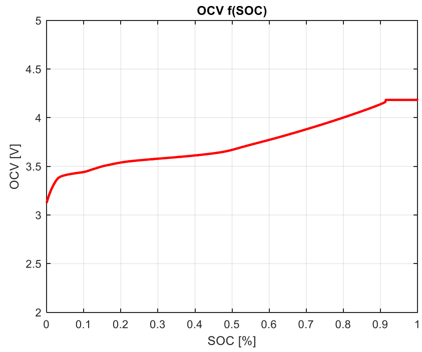


Figure 9. Open Circuit Voltage $V_{OCV}=f(V_{SOC})$

This battery cell relation between V_{SOC} and V_{OCV} is going to be used then on prototype implementation. The values are going to be load on a lookup table to use on Simulink simulations and to program the microcontroller for implementation part.

2.3 Buck converter design

The buck converter is the topology chosen to emulate the INR cell battery. The bidirectional converter is the most appropriate topology to emulate a real cell because it would be able to charge and discharge the cell. However, to simplify the prototype design, this buck converter is only unidirectional to achieve only discharge mode. The output voltage range expected based on battery cell is between 2.5 V to 4.5 V. The following table describes the converter specifications that would be designed in this chapter:

DC-DC Converter specifications	
Input Voltage	V_{IN} = 12 V
Output Voltage	V_O = [2 V…4.5 V]
Output Power	P_{Omax} = 5 W
Theoretical Output Load	5 Ω
Theoretical duty cycle	$D_a=V_{O1}/V_G=2/12=0.16\%$ $D_b=V_{O2}/V_G=5/12=0.41$
Working mode	CCM -> Continuous Conduction Mode
Max current	2 A
Output voltage ripple	ΔV_O = 5 mV
Switching frequency	100 kHz

Table 3. DC-DC Converter specifications

This converter, as defined above, operates in continuous conduction mode (CCM) which implies two static subintervals. The electrical converter equations consider the inductor current $I_L(t)$ and voltage capacitor $V_c(t)$ which are not linear variables. So, the following table summarizes the differential equations used to represent the buck converter behavior. These equations are going to be used later for the linearized model.

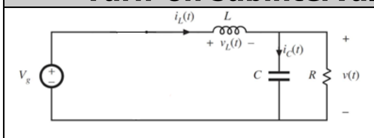
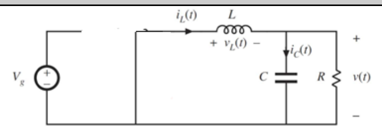
<table>
<tr><td></td><td></td><th>Turn-on subinterval</th><th>Turn-off subinterval</th></tr>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td rowspan="4">Differential equations</td><td>Inductor current subintervals</td><td>$\frac{diL(t)}{dt}=\frac{V_G}{L}-\frac{V_C}{L}$</td><td>$\frac{diL(t)}{dt}=-\frac{V_0}{L}$</td></tr>
<tr><td>Average current</td><td colspan="2">$\frac{diL(t)}{dt}=\frac{-V_0-dV_G}{L}$</td></tr>
<tr><td>Output voltage Subinterval</td><td>$\frac{dVc(t)}{dt}=\frac{I_L}{C}-\frac{V_o}{R\cdot C}-\frac{I_o}{C}$</td><td>$\frac{dVc(t)}{dt}=\frac{I_L}{C}-\frac{V_o}{R\cdot C}-\frac{I_o}{C}$</td></tr>
<tr><td>Average voltage</td><td colspan="2">$\frac{dVc(t)}{dt}=\frac{-V_0-RI_L-RI_O}{R\cdot C}$</td></tr>
</table>

Table 4. Converter differential equations

To proceed on the converter dynamic analysis a derivative model definition is need. The approximation must be linearizable by constructing a small-signal converter model which lose the high frequency information. A matrix method is used to define the state-space average model which its equation in equilibrium is:

0 = A · X + B · U

$$0 = \begin{pmatrix} 0 & -\frac{1}{L} \\ \frac{1}{C} & -\frac{1}{C \cdot R} \end{pmatrix} \cdot \begin{pmatrix} I_L \\ V_0 \end{pmatrix} + \begin{pmatrix} \frac{d}{L} & \frac{V_g}{L} & 0 \\ 0 & 0 & -\frac{1}{C} \end{pmatrix} \cdot \begin{pmatrix} V_g \\ d \\ I_0 \end{pmatrix}$$

Figure 10. State-space average model in equilibrium

The equilibrium State vector "X" is composed by sate variables which are the inductor current and voltage capacitor. These electrical state variables whose values in t=0 (equilibrium) are necessary to know any value when t>0. The state variable determines the energy stored during a specific period. The input vector "U" is considered the input voltage, the duty cycle, and current perturbations. The details are in the annex section [A.2]

This linearized circuit model is introduced in MATLAB to continue the analysis. To do that is necessary to dimension the electrical state variables. The inductor value calculated based on the worst-case scenario is:

$$L > \frac{V_o}{f \cdot \triangle I_{min}} = \frac{5}{100 \cdot 10^3 \cdot 0.8} = 62.5\ \mu H \rightarrow L = 68\ \mu H$$

The capacitor calculated to achieve the minimum output voltage ripple is:

$$C > \frac{(1-D) \cdot V_o \cdot T_s^2}{8 \cdot L \cdot \triangle V_o} = \frac{(1-0.41) \cdot 5 \cdot (10\mu s)^2}{8 \cdot 68\mu H \cdot 5mV} = 108uF \rightarrow C > 108\mu F$$

The output load of 5 Ω has been considered to perform the dynamic analysis. So, the next step is to validate the plant model under Laplace (S) domain. The control to output function on the output voltage limits are:

$$G_{vd\,A}(s) = \frac{v(s)}{d(s)} \rightarrow (vg(s) = 0) = \frac{8.0214e+08}{s^2 + 5862s + 6.684e07} = G_{vd\,A}(s)$$

$$G_{vd\,B}(s) = \frac{v(s)}{d(s)} \rightarrow (vg(s) = 0) = \frac{8.0214e+08}{s^2 + 909s + 6.684e07} = G_{vd\,B}(s)$$

The next bode representation demonstrates that control to output function is stable, but the phase are in the limit, because the crossover frequency approaches the 180º.

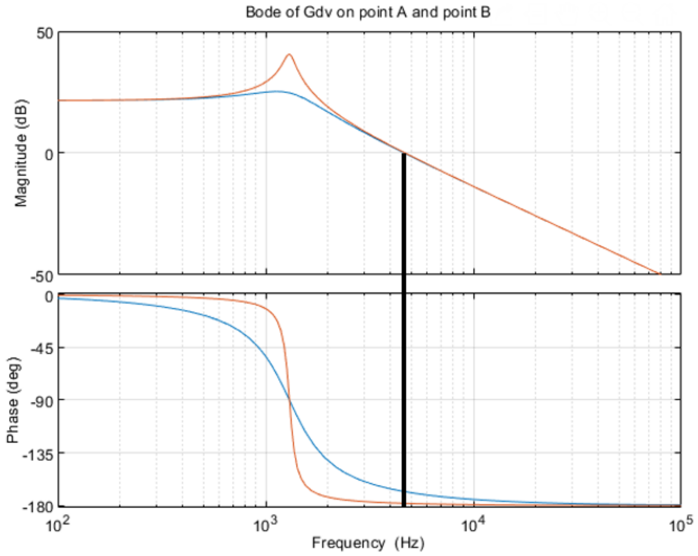


Figure 11. The bode open loop gain for point A and B

One of the main criterial to define the compensator must consider improving the phase margin, because the phase margin is in the limit of stability following the special case of the Nyquist stability theorem. The system step response does not oscillate but takes so time to be stable, in special one of the equilibrium point condition. So, the step response also confirms that the stability must be improved during the close loop design. The details are in the annex section [A.2].

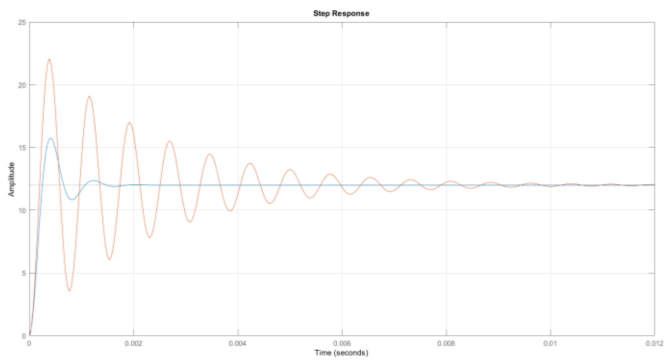


Figure 12. Open loop system step response on the two limit points.

The quality factor calculated is too high for one limit point. Considering that the optimum qualify factor is between 0.5 to 1, the ones calculated in open loop must be adapted to enter in the optimum range values.

$$Q = \frac{R \cdot \sqrt{C}}{\sqrt{L}} = \frac{5 \cdot \sqrt{220uF}}{\sqrt{68uH}} = 9$$

$$Q_A = \frac{1}{2 \cdot \xi} = \frac{1}{2 \cdot 3.47e-1} = 1.44$$

$$Q_B = \frac{1}{2 \cdot \xi} = \frac{1}{2 \cdot 5.56e-2} = 9$$

The system has been validated under different sinusoidal waves frequencies. The frequency response estimator has been used to analyze the converter response under frequency variations. The average model has been used to simplify the system.

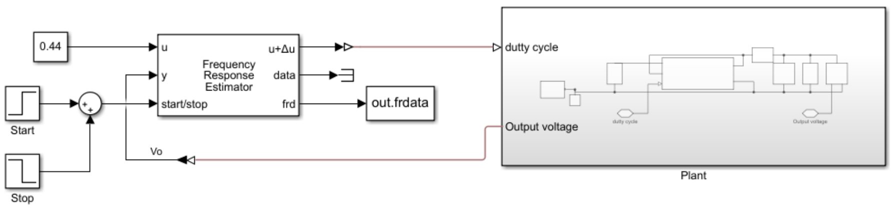


Figure 13. Frequency Response Estimator schematic

The low frequency interval response is the expected one, but the system does not answer as expected during the high frequency interval, because that interval is approximately the switching frequency (f=100 kHz).

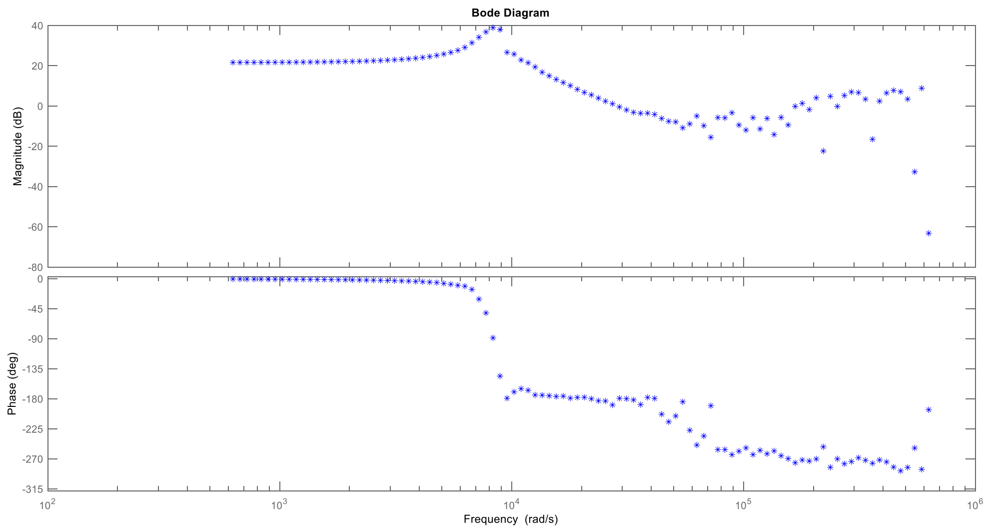


Figure 14. Frequency Response Estimator result

2.3.1 Digital compensator

A proportional integrator compensator (PI) has been designed for this converter to achieve the prototype requirements. The PI proposed has a zero to maintain adequate phase margin. The crossover frequency must be sufficiently below the switching frequency.

$$G_C(s) = \frac{Vc(s)}{Ve(s)} = \frac{0.03 \cdot (s + 5300)}{s}$$

The bode plot shown below compares the Gvd transfer function and the close loop transfer function. The close loop system compensator keeps the system stable because the phase margin is 90º at the crossover frequency of 30Hz.

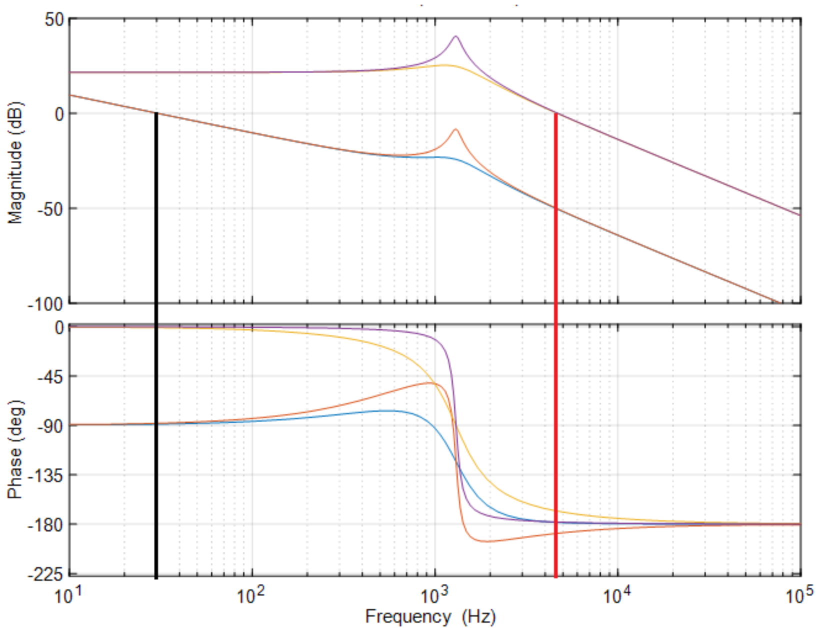


Figure 15. Bode comparison between Gvd and close loop T.

The low crossover frequency causes slow response time. Although, this project accepts this constrains, the cell emulator presents a low accuracy of fast changes in the discharge current, as for example in the case of load steps.

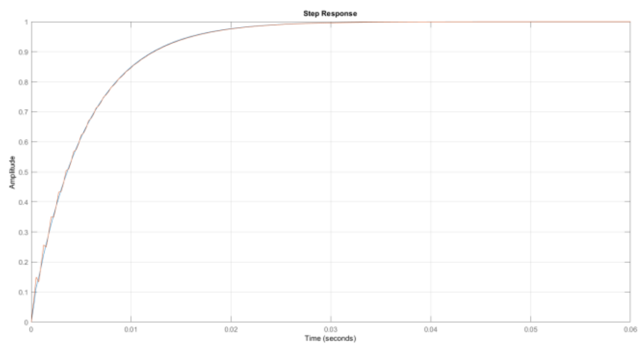


Figure 16. Step system response.

A system discretization has been done to implement the digital control. Tustin transformation has been used to obtain the discretized compensator. The same switching frequency is used for the sampling frequency which is 100 kHz. The sampling frequency is many times higher than the crossover frequency for that reason, the direct transformation could be implemented without considering the delays associated to discrete-time control. The details are specified on the annex section [A.2].

$$G_{cd}(z) = \frac{d(z)}{e(z)} = 0.00307 \cdot \frac{(z - 0.9484)}{(z - 1)}$$

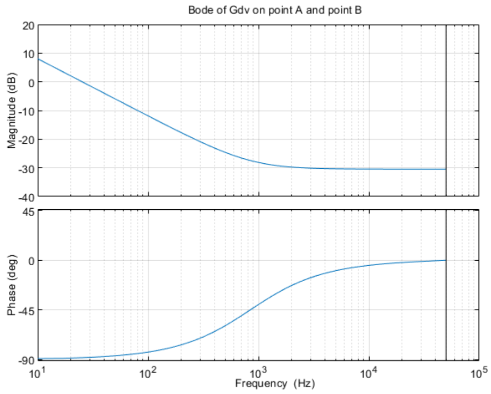


Figure 17. The $G_{cd}(z)$ obtained using a Tustin Transformation

Some mathematical operations based on the Tustin transformation result has been performed to achieve a standard equation.

$$G_{cd}(z) = \frac{d(z)}{e(z)} = 0.00307 \cdot \frac{(z - 0.9484)}{(z - 1)} \cdot \frac{\frac{1}{z}}{\frac{1}{z}}$$

$$G_{cd}(z) = \frac{d(z)}{e(z)} = 0.00307 \cdot \frac{(1 - 0.9484z^{-1})}{(1 - z^{-1})}$$

$$d(z) = 0.0307 \cdot \frac{(1 - 0.9484z^{-1})}{(1 - z^{-1})} \cdot e(z)$$

$$d(z) = \frac{0.00307 - 0.03z^{-1}}{(1 - z^{-1})} \cdot e(z)$$

The equation obtained above has been used to compare with the following standard equation. The compensator constants have been obtained based on this comparison between both equations.

$$d(z) = \frac{K_p + \ K_i\,T_s \ \ + K_p\,z^{-1}}{(1 - z^{-1})} \cdot e(z)$$

$$\frac{d(z)}{e(z)} = \frac{0.00307 - 0.03z^{-1}}{(1 - z^{-1})} = \frac{K_p + \ K_i\,T_s \ \ + K_p\,z^{-1}}{(1 - z^{-1})}$$

$$0.00307 = \ K_p + \ \ K_i\,T_s \qquad K_p = -0.03$$

$$T_s = \frac{1}{100kHz} = 10us$$

$$0.00307 = \ -0.03 + \ \ K_i\,10us$$

$$K_i = \frac{0.00307 + 0.03}{10us} = 3307$$

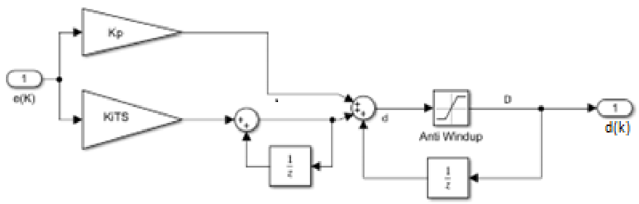


Figure 18. Simulink representation of compensator Gc(z)

The Simulink representation above is the difference compensator equations. It is composed by the integrator block, the gain block, and the anti-windup structure. This structure is necessary in a close loop system to avoid the integrator action overload. The discretized transfer function is very sensitive, for that reason all decimals in the gains blocks has been considered in Simulink otherwise the compensator could be affected on its results.

2.3.2 Simulations

Firstly, the converter has been simulated in open loop for two static points which are the cell level limits. The lowest cell level is represented with a value of 2 V, and 5 V corresponds to the highest cell level. The inductor current has an appropriate curve avoiding the pass for 0 which keeps the convert in continuous conduction mode (CCM).

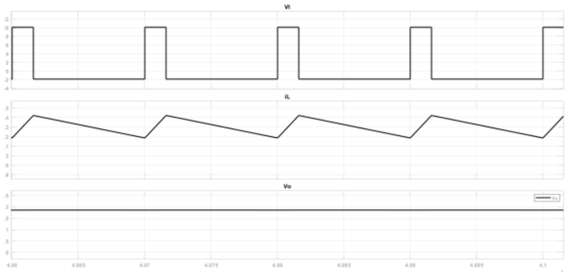


Figure 19. Point A(Vout = 2 V Dutty = 0.16%)

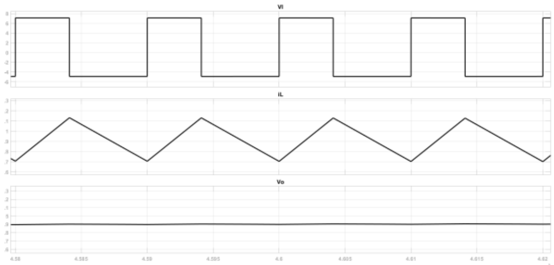


Figure 20. Point B(Vout = 5 V Dutty = 0.41%)

Secondly, the converter has been simulated in closed loop with a resistor output perturbation. The model considers the ADC 12 bits idealized quantifier, and the acquisition module to recover the original analog value. The compensator part is composed by the difference model equation extracted from the previous section. The anti-windup structure has been added to be assure the close loop delivers appropriate values. The reference of 3V has been set for this first test.

The output voltage converter has been quantized in this model to design a model as close as possible to the final prototype. Then, the acquisition stage has been implemented to recover the original signal. To do that, the formula described on microcontroller reference manual has been used. [10]

$$Voltage\ value = \frac{Digital\ value \cdot (V_{LMax} - V_{LO})}{2^{adc\ bits}} + V_{LO}$$

The ADC maximum input voltage restriction is 3 V, for that reason a resistor divider has been introduced after output voltage converter to adapt the value for the ADC input stage. A gain block at the end has been used to recover the original value.

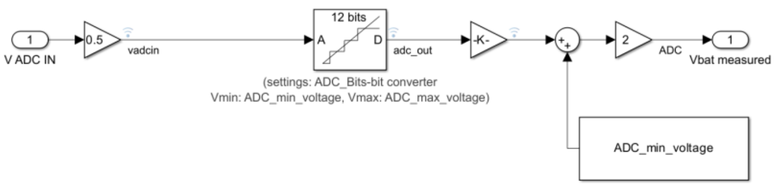


Figure 21. Voltage ADC acquisition.

The inductor current and output voltage under perturbations have been shown on the figure. Although, there are perturbation effects on output voltage, the converter keeps the reference value of 3 V. As a conclusion, the converter works successfully at simulation level.

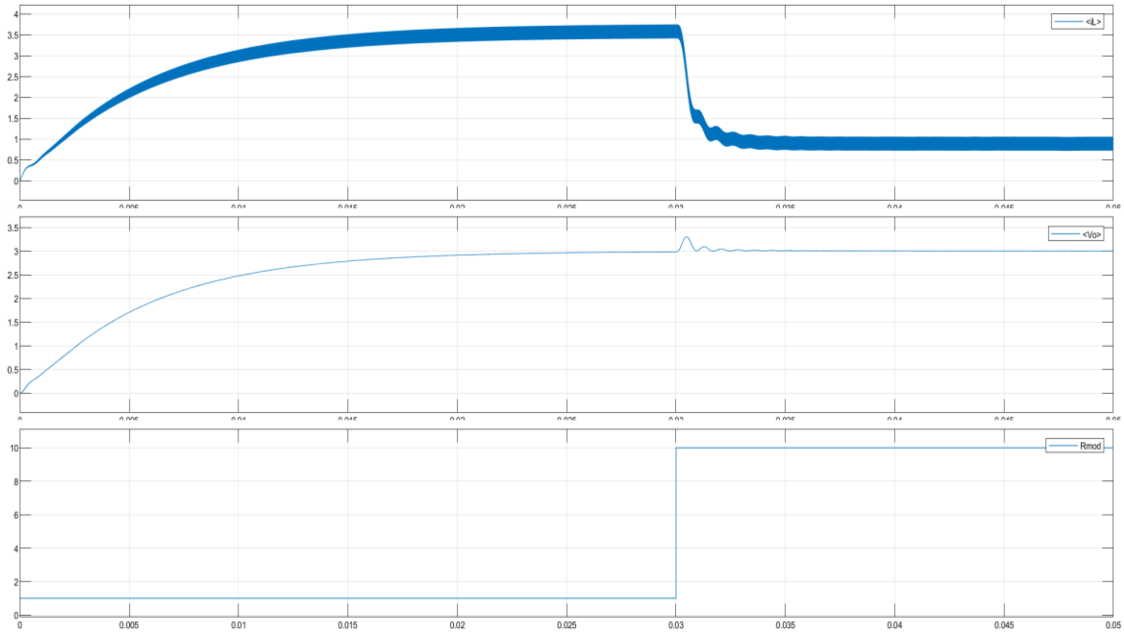


Figure 22. The control effect under perturbation applied.

2.4 Battery cell integration

The previous chapters define a method to model a battery cell and a buck converter design. The next step, which will be taken in this chapter, is to integrate the battery cell model using a buck converter.

The figure below is the whole system schematic representation. The battery cell control updates the SOC periodically to obtain a constant voltage reference for the DC-DC. The closed loop adjusts the voltage reference to regulate the converter and to avoid fluctuations in case there are some load perturbations. The voltage reference range for the DC-DC are the battery cell limits. The model has been designed to discharge the battery for that reason the SOC always decrease its values.

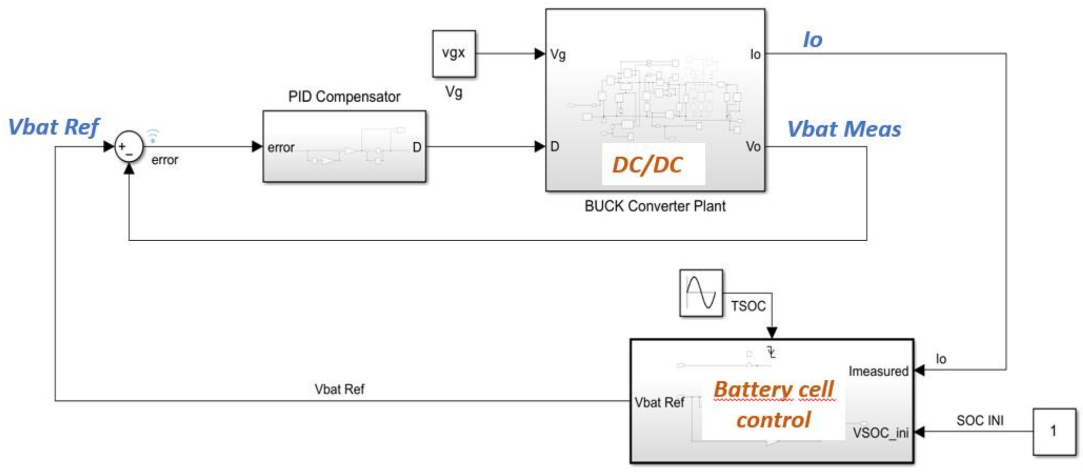


Figure 23. Prototype block diagram

The "Buck converter Plant" block as well as the "PID compensator" block has been designed and explained accurately on previous chapter. However, the battery cell model has been developed on this chapter.

The state of charge voltage (V_{SOC}) has been updated following the next equation which represents the basic cell model without considers the dynamic effects.

$$V_{SOC}(t) = \ V_{SOC}(t-1) - \frac{1}{C_{SOC}} \int_0^{T_{end}} ibat(t)dt$$

The state of charge voltage (V_{SOC}) obtained from the equation above will be used to obtain the open circuit voltage (V_{OCV}). A lookup table has been used to relation between V_{SOC} values and the V_{OCV} values. The values updated on the lookup table has been extracted from CALCE group battery profile. The battery profile used represents the behavior of battery cell INR18650-20R. The profile details are on the CALCE website [5]. The only dynamic effect considered is the resistive serial drop. The next figure represents the battery cell model.

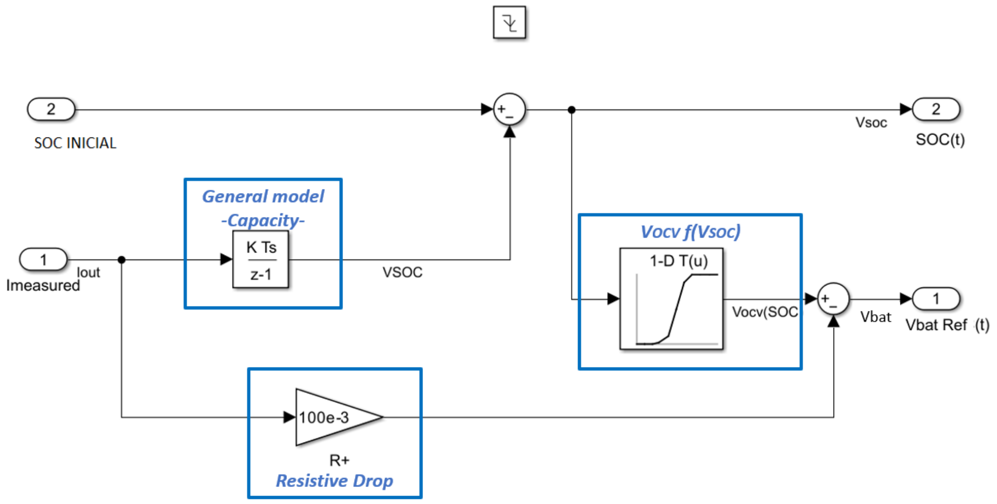


Figure 24. Battery cell model implemented

The following graph is the lookup table values loaded on the model to perform a discharge cell as close as possible to the real SAMSUNG battery cell.

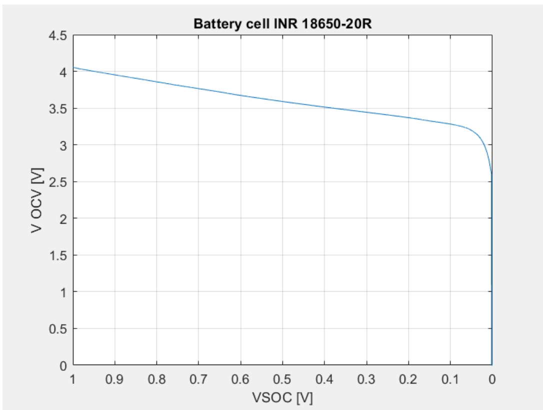


Figure 25. Battery cell INR 18650-20R

Several load variations have been added on the output buck converter to simulate different discharge conditions during different time slots. The cell could be discharged faster as long as the output current increases.

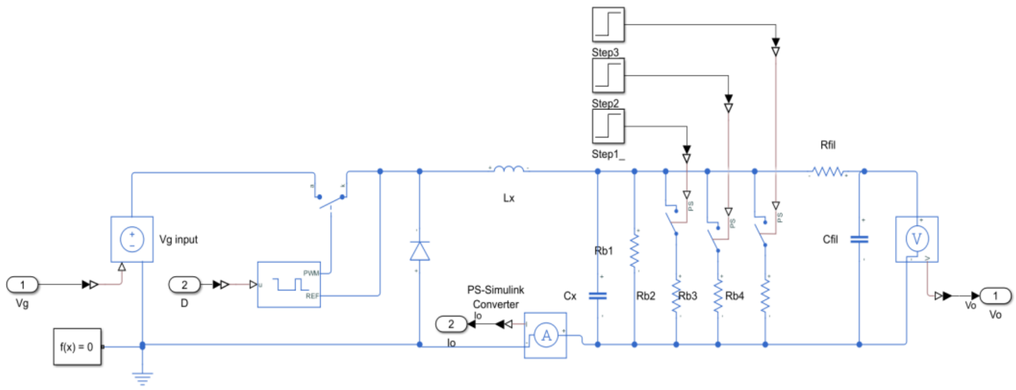


Figure 26. The buck converter with different load discharge conditions

The next graph will be the result of that load variations. The simulation time is short, because the complete model takes so time to be compiled. For that reason, the state of charge will be little downloaded.

The next graphs represent the discharge cell which works on different current conditions based on output load variations. The dynamic effect on voltage drop is appreciated on the Vbat measured with different current conditions. Although, the SOC discharged is so low due to short time simulation, the different soc slopes could be appreciated depending on the output current applied.

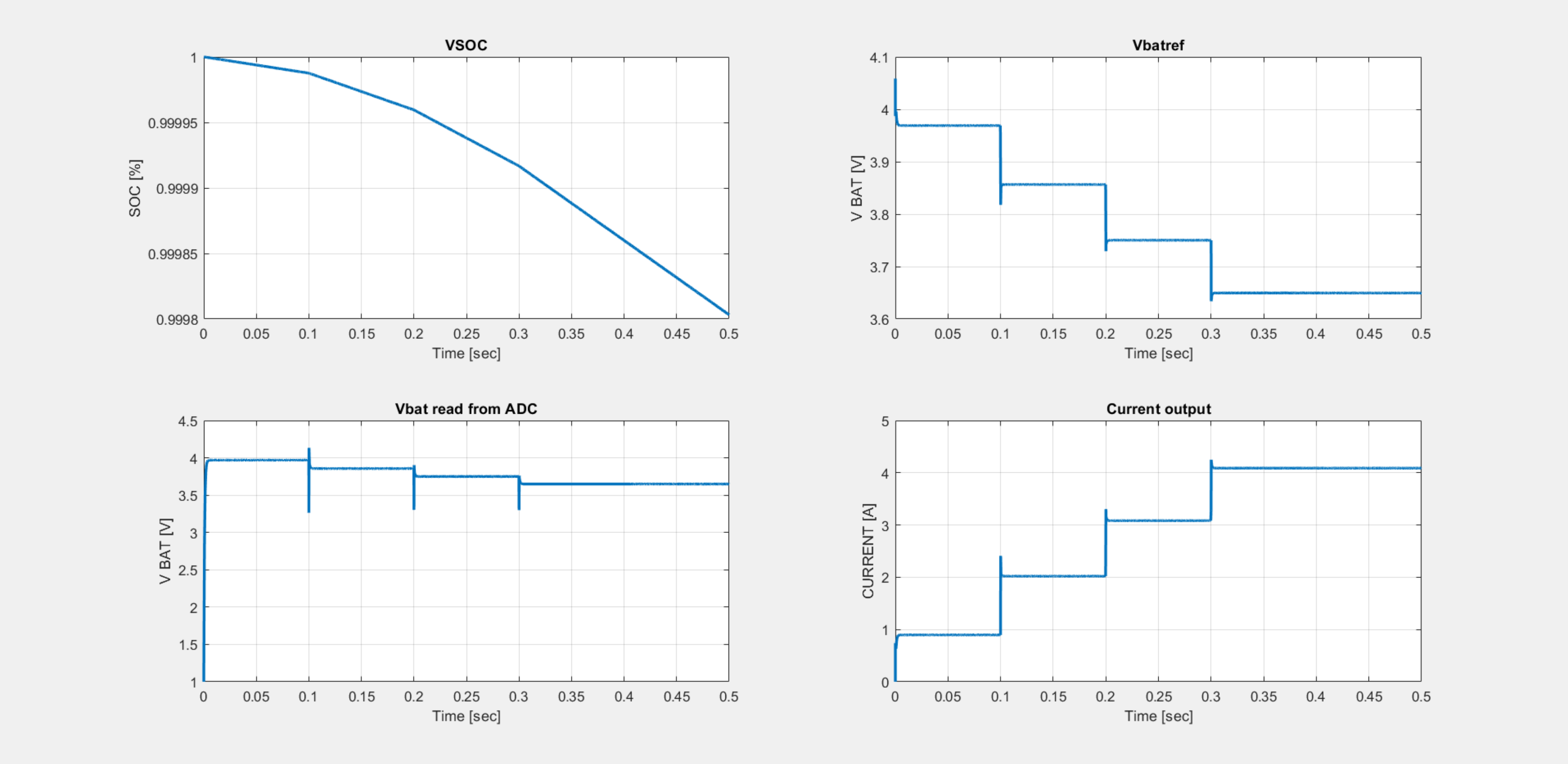


Figure 27. Battery discharge cell under output variations

The "Vbat Meas" is the output signal converter which matches with the "Vbat ref" which is the reference converter voltage. So, the buck converter delivers the output voltage expected according to each reference point even under load variations. The next figure only shows the voltage measured which some voltage spikes could be observed. The reason of that comes from dynamic converter response. Although, there are some voltage peaks on load variations, the battery cell variations are slow enough to obviate these voltage peaks.

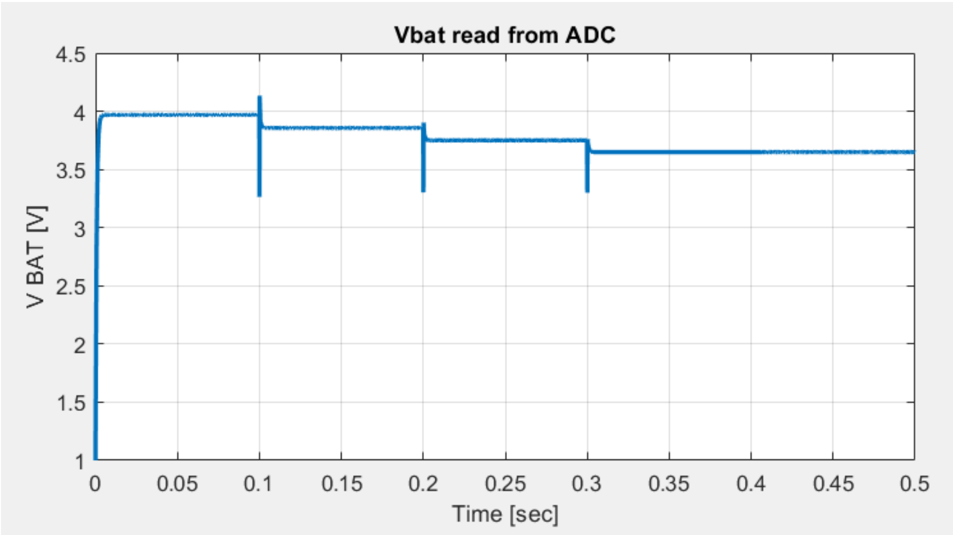


Figure 28. Voltage peaks on load variations due to slow response converter

The series resistor effect is the only dynamic effect contemplated in this model. Other dynamic effects like hysteresis or diffusion effects would be contemplated to get a more accurate representation of real battery, but these effects complicate the prototype design and they have been discarded.

The next simulations are going to be implemented with average model due to reduced the compilation resources. The system has been evaluated with different load conditions. Firstly, the R=5 Ω constant load has been used to discharge the battery cell. The C-rate obtained in that conditions is approximately of 0.55 C. The total time to discharge the battery cell with that conditions will be approximately 2 hours.

$$C_{rate} = \frac{discharge\ current\ [A]}{nominal\ capacity\ [Ah]} = \frac{1117mA}{2000mAh} = 0.55\ C$$

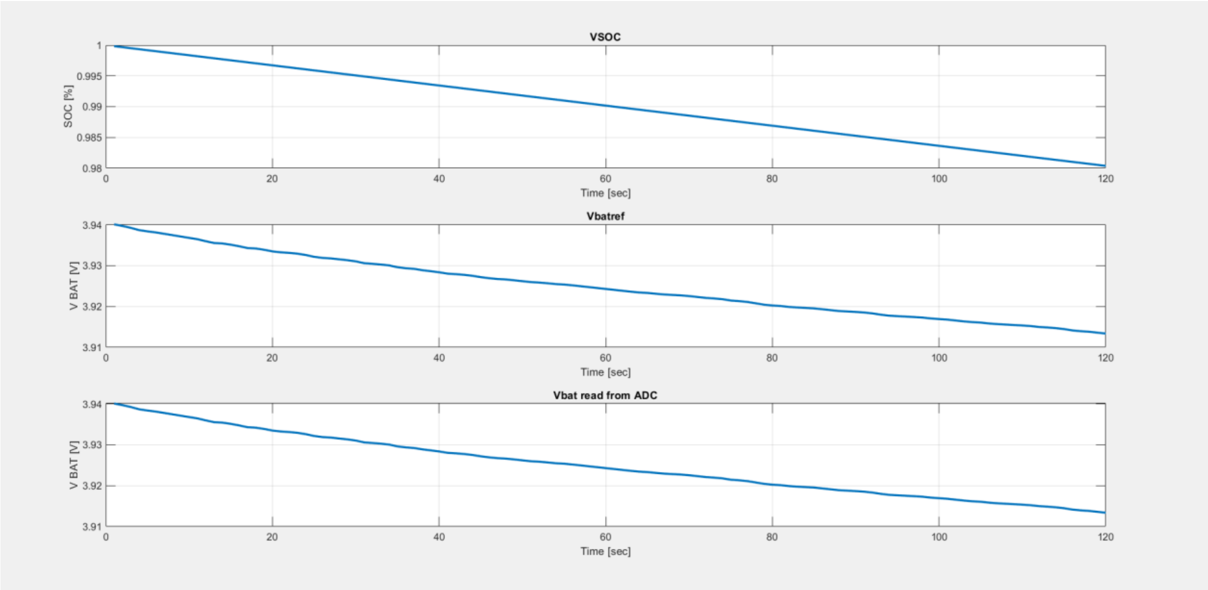


Figure 29. Battery discharge 1% with constant output load of R=5 Ω

$$V_{SOC}\ (\mathrm{t} = 120) = 1 - \frac{1}{C_{SOC}} \cdot \int_0^T ibat(t)dt = 1 - (\frac{1}{7200} \cdot 1.17 \cdot 120) = 0.9805$$

$$Total\ discharge\ time = \frac{2000\ mAh}{1117\ mA} = 1.79H = 1\ hora\ 47min$$

The second case analyzed uses a lower resistor load, in this case Rout = 0.8 Ω. The lower resistor value, which reaches a highest output current, get the new C-rate value of 30 C.

$$C_{rate} = \frac{discharge\ current\ [A]}{nominal\ capacity\ [Ah]} = \frac{6\ A}{2000\ mAh} = 30\ C$$

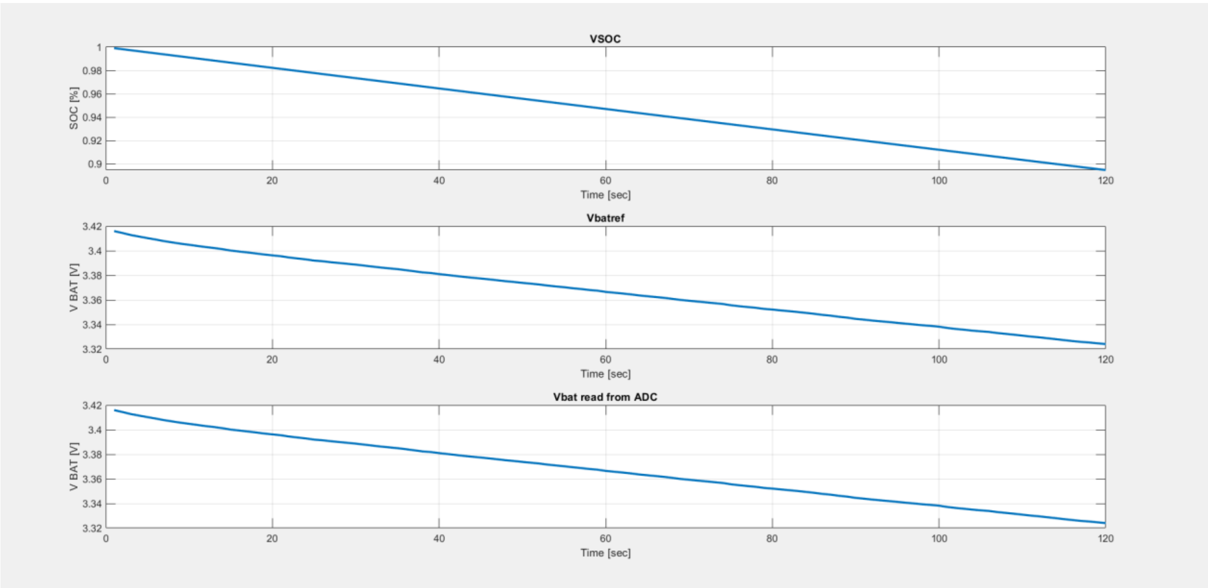


Figure 30. Battery discharge with constant output load of R=0.8 Ω

The time analyzed of 120 seconds, which is the same as previous test, discharge the battery until the 90% of the SOC.

$$V_{SOC}\,(\mathrm{t}=120) = 1 - \frac{1}{C_{SOC}} \cdot \int_0^T ibat(t)dt = 1 - (\frac{1}{7200} \cdot 6 \cdot 120) = 0.90$$

The third case analyzed uses a resistor load of Rout = 100 Ω. This case now obtains the opposite effect due to low current output. The SOC obtained after the same timing test is 99%, so the cell would need 33 hours to be totally discharge. The C-rate value in this case is 0.3 C.

$$C_{rate} = \frac{discharge\ current\ [A]}{nominal\ capacity\ [Ah]} = \frac{600mA}{2000mAh} = 0.3\ C$$

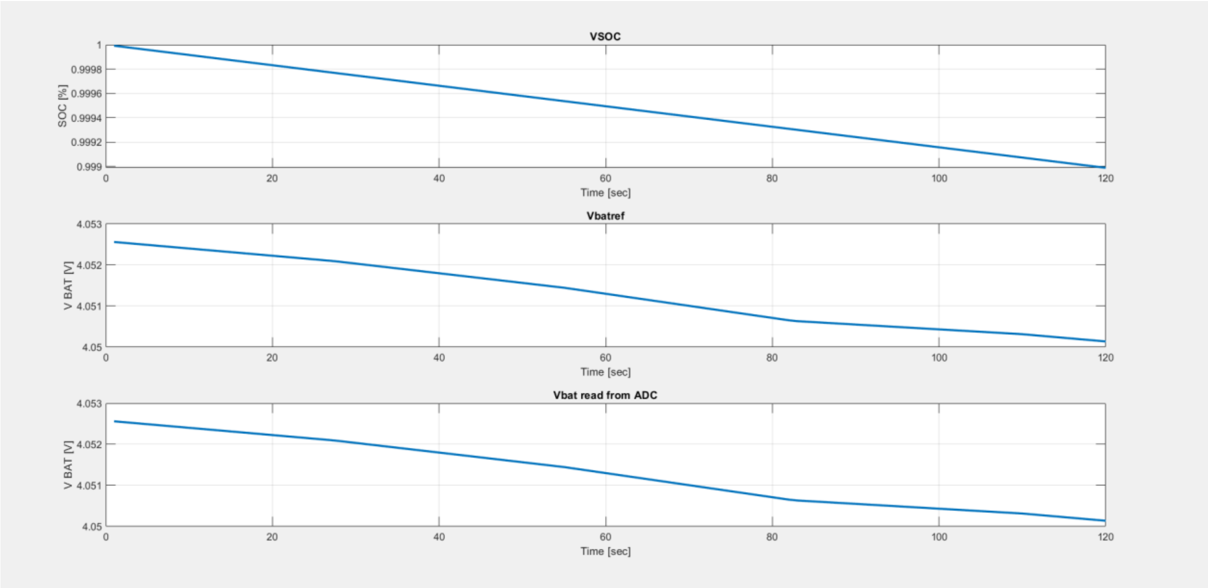


Figure 31. Battery discharge with constant output load of R=100 Ω

$$V_{SOC}\,(\mathrm{t}=120) = 1 - \frac{1}{C_{SOC}} \cdot \int_0^T ibat(t)dt = 1 - (\frac{1}{7200} \cdot 0.6 \cdot 120) = 0.99$$

The next graph is the relation of the SOC in function of Vbat during the time test of 120 sec.

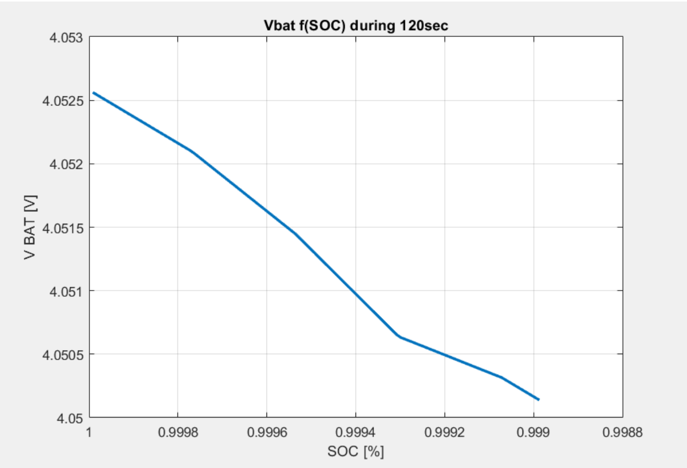


Figure 32. Vbat f(VSOC) for the low current profile.

Next graph shows the battery voltages which corresponds on the scenarios described above. The slopes are different depend on the load conditions.

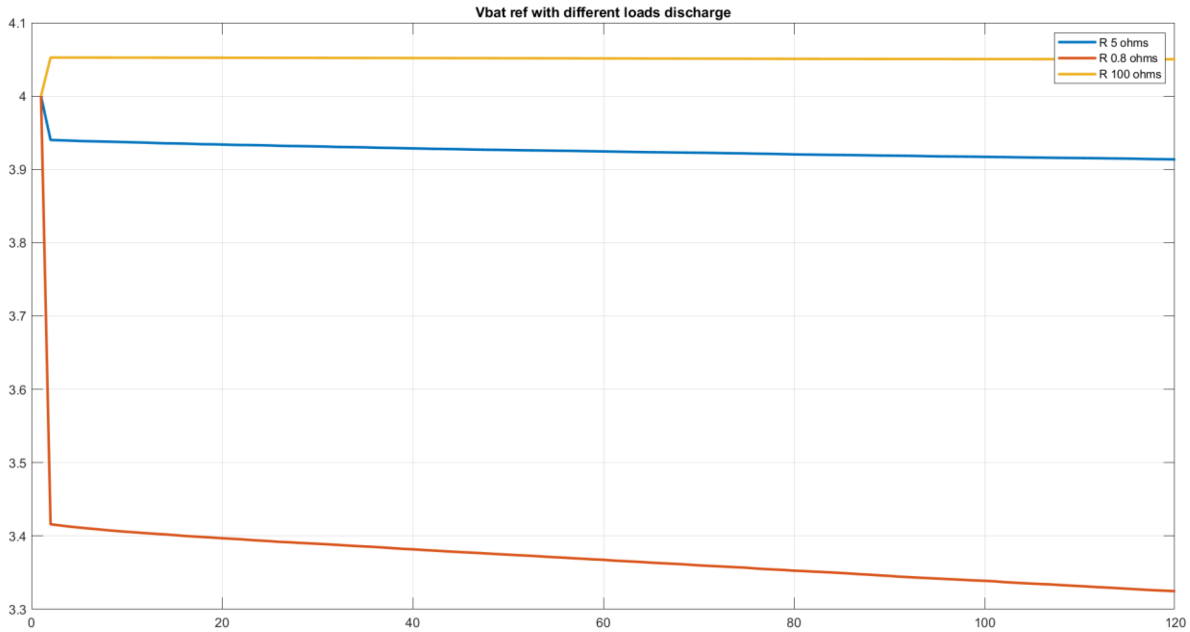


Figure 33. Battery discharge comparison between different load values

To sum up, there is the following table which synthesize the different test cases which the average current output and time expected to discharge the battery.

Case	Resistor load [Ω]	Average current [A]	Time expected discharge
1	0.8 Ω	1.16 A	103 min
2	5 Ω	6.62 A	18 min
3	100 Ω	0.06 A	33 hours

Table 5. Comparison between different discharge scenarios

3 Battery cell prototype

The physical prototype implementation is going to be described in this chapter. The purpose is to obtain a similar cell behavior achieved on the model design, but now a PCB prototype with microcontroller programed. A Launchpad from Texas Instruments has been used on this this prototype for the control part, and a customized PCB has been used for the power part which the buck converter is implemented.

The cell discharge is constant on this prototype, for that reason the load resistors has been mounted on the PCB to keep the constant load. Despite of the embedded resistors, two output bananas have been mounted to add extra loads in case to change the discharge current. The close loop PI regulation has been implemented on C code to maintain a successful regulation based on voltage cell references and avoiding possible load disturbances. The PCB track width is 2mm to assure the power current output without losses, and the signal tracks width is 1mm. Almost all tracks has been distributed on bottom side to simplify the design. The PCB layout which illustrated the specifications above has been included on the annex part [A.8].

The next figure shows the PCB prototype implemented with all components mounted. The components used are through hole (THT) to facilitate the assembly design. If the components had been surface mounted devices (SMD), the PCB cell emulator would have been reduced its dimensions. The component list used to build the customized PCB for the buck are described on the annex part [A.4].

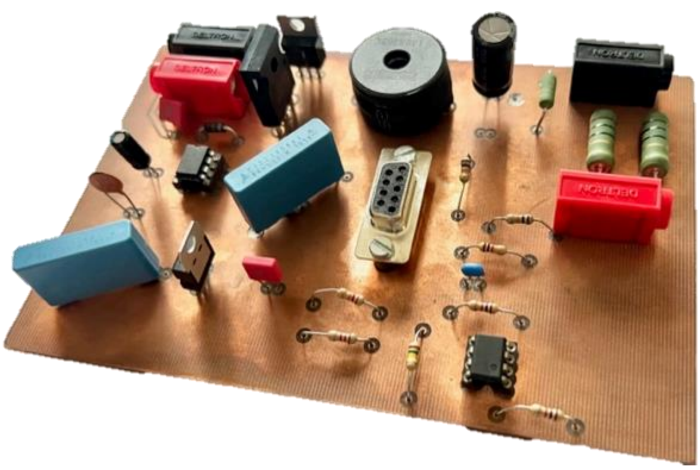


Figure 34. Customized PCB for DCDC buck converter implementation.

The control part has been implemented with a Launchpad which had been used on the master laboratory courses. The DIN-9 connector has been mounted to use as an interface between customized PCB and Launchpad. The main purpose is to implement the buck converter on the PCB, and the control part programed using Launchpad. Additional components have been mounted on the customized PCB. For instance, the Mosfet driver, the operational amplifier to adjust the measure for ADC and a linear regulator. The 5 V lineal regulator was used to supply the control part on the previous prototype release; however, this is not necessary using Launchpad, because it does not need external supplier. The Schematic PCB has been included on the annex part [A.7]

The next figure is the Launchpad used to program the control loop gain. The microcontroller used on this platform is TMS320F28379D, the main characteristics are described below.

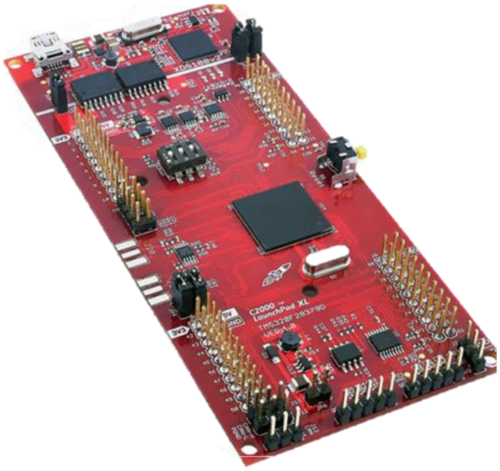


Figure 35. Launchpad Platform studio board used to develop the control block.

This Launchpad has embedded the USB connected isolated XDS100v2 JTAG debug probe for real-time debug and flash programming. The most notable microcontroller features are described on the following table:

Main MCU parameters	
CPU	Dual 32-bit Floating-Point C28x CPUs,
Frequency (MHz)	200MHz
Memory	1MB embedded Flash, and 204KB RAM
ADC resolution (Bps)	12 or 16
PWM	24 PWM channels with resolution of 150 ps.
Communication interface	CAN, I2C, SPI, UART

Table 6. Key MCU features (TMS320F28379D)

3.1 Mosfet driver configuration

A high side driver has been used to control the Mosfet in bootstrap configuration. The IR2125 has been used to control the Mosfet IRFP4110PBF. The minimum voltage applied on V_{GS} terminals is 5 V according with the next figure extracted from datasheet.

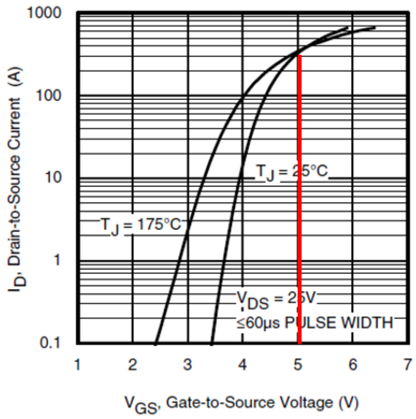


Figure 36. Typical transfer characteristics [12]

The maximum system output is approximately 6.6 V, so the voltage to be applied on the gate terminal is 11.6 V. According to the capacity datasheet specifications and the voltage circuit conditions, the bootstrap capacitor has been calculated.

$$V_{GS} > 5V \rightarrow V_{GS} = V_G - V_S \rightarrow V_G = V_{GS} + V_S = 5 + 6.6 = 11.6V$$

$$C_B > \frac{2 \cdot [2 \cdot Q_g + \frac{I_{qbs(max)}}{f} + Q_{ls}]}{V_{cc} - V_f - V_{LS} - V_{min}} = \frac{2 \cdot [2 \cdot 150nC + \frac{100nA}{100kHz} + 1nC]}{12 - 0.7 - 5 - 5} = 463nF$$

3.2 Hardware sensing design

The output voltage and current has been captured by microcontroller ADC. There is necessary to adapt the analog output converter values to appropriate signals values for the ADC. The current and voltage signal has been processed with different way which are explained below.

The output current is measured by shunt resistor of 100 mΩ . The maximum current expected is too low for ADC input signal, for that reason, the operational amplifier block has been implemented. With this way the ADC has an accurate current measuring.

	Output current	Vshunt	Vin ADC	Identifier
Min	0 A	0 V	0.5 V	Point A
Max	2 A	0.2 V	3.3 V	Point B

Table 7. Acquisition current measurement

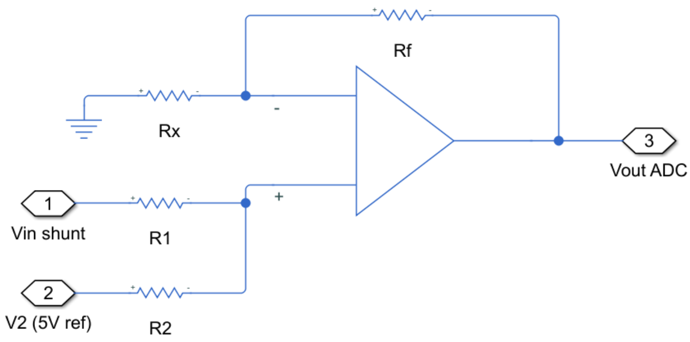


Figure 37. Current stage adaptation for an optimum ADC measurement

The following equation represents the mathematical point of view of that amplifier stage.

$$V_O = \frac{R_2 \cdot (R_x + R_f)}{R_x \cdot (R_1 + R_2)} \cdot V_{in} + \frac{R_1 \cdot (R_x + R_f)}{R_x \cdot (R_1 + R_2)} \cdot V_2$$

Using that generic equation, the point A and point B are defined as follow:

$$Point\ A \rightarrow 0.5 = \frac{R_1 \cdot (R_x + R_f)}{R_x \cdot (R_1 + R_2)} \cdot 5 \rightarrow 2.5 = \frac{R_1 \cdot (R_x + R_f)}{R_x \cdot (R_1 + R_2)}$$

$$Point\ B \rightarrow 3.3 = \frac{R_2 \cdot (R_x + R_f)}{R_x \cdot (R_1 + R_2)} \cdot 0.2 + \frac{R_1 \cdot (R_x + R_f)}{R_x \cdot (R_1 + R_2)} \cdot 5$$

The value of two resistors is assumed to be known to facilitate the solving equations.

$$\boldsymbol{R_x = 1k}\Omega \quad \boldsymbol{R_2 = 150k}\Omega$$

Finally, the rest of resistors values has been calculated.

$$Point\ A \rightarrow 2.5 \cdot (1kR_1 + 150k) = 1kR_1 + R_1 \cdot R_f$$

$$Point\ B \rightarrow 3.3 = \frac{150k \cdot (1k + R_f)}{1k \cdot (R_1 + 150k)} \cdot 0.2 + \frac{R_1 \cdot (1k + R_f)}{1k \cdot (R_1 + 150k)} \cdot 5$$

$$\boldsymbol{R_f = 13100}\Omega \quad \boldsymbol{R_1 = 1k}\Omega$$

After this calculated step above, the current stage acquisition can be finalized. The next step is to implement the voltage acquisition step, which is designed with a resistor divided, low pass filter to avoid high frequency noise, and the calibration step to acquire more precision real values.

The output voltage range converter values overcome the ADC limits, for that reason a resistor divider has been used to reduce the input voltage values for the ADC. Moreover, the capacitor has been added to filter the possible high frequency variations. The following schematic is the acquisition designed to adapt the voltage input range for ADC:

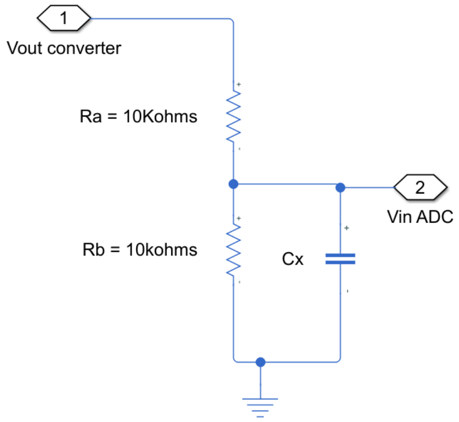


Figure 38. Adaptation stage for input ADC margins

The crossover frequency applied for the low pass filter is 1 kHz. The following bode shows the crossover frequency as well as the attenuation due to resistor divider.

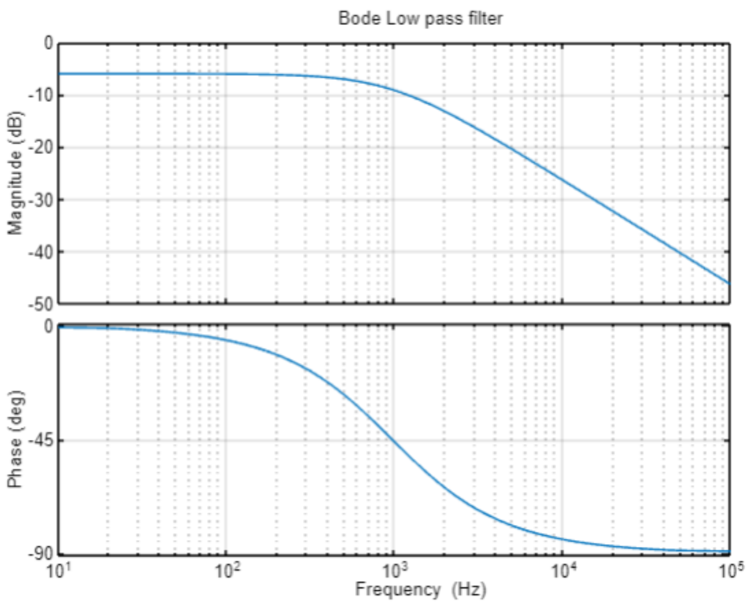


Figure 39. Bode low pass filter transfer function representation

$$H(s) = \frac{VinADC(s)}{Voutconverter\ (s)} = \frac{1}{2 + RCs}$$

The capacitor has been calculated based on the transfer function equation and the filter crossover frequency.

$$W_o = 2 \cdot \pi \cdot 1000Hz = 6283.18 \frac{rad}{s}$$

$$H(s) = \frac{1}{2 + RCs} = \frac{\frac{1}{2}}{1 + \frac{RCs}{2}} = \frac{1}{2} \cdot \frac{1}{1 + \frac{RCs}{2}} = \frac{1}{2} \cdot \frac{1}{1 + \frac{s}{w}}$$

$$\frac{RC}{2} = \frac{1}{W_o} \rightarrow W_o RC = 2 \rightarrow C = \frac{2}{W_o \cdot R}$$

$$C = \frac{2}{2 \cdot \pi \cdot 10000 \cdot R} = 30\, nF = C$$

The ADC must be calibrated to read the right values. Two calibrations points has been used to calculate the linearization coefficients.

$$a + bx_1 = V_1 \rightarrow a + 1780b = 1.92$$

$$a + bx_2 = V_2 \rightarrow a + 2330b = 3.55$$

$$b = \frac{3.55 - 1.92}{2330 - 1780} = 0.003$$

$$a = 1.92 - 1780 \cdot 0.003 = -3.335$$

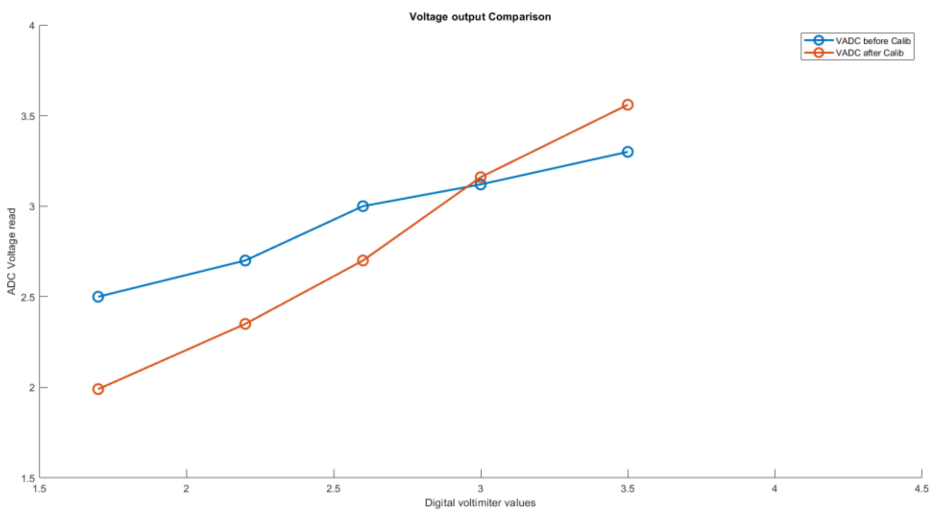


Figure 40. Comparison between values before and after calibration

The graph above demonstrate that the calibration effect based on coefficients calculated returns a lineal relation between the real value and value read from ADC.

3.3 Software development

The main software concept is to capture periodically the voltage output converter as well as its current output. Based on that input signals, a PI controller is implemented with a voltage close loop to adjust the PWM signal for the gate Mosftet. Moreover, there is a function which updates the battery State of Charge (SOC).

The timer implemented has been used to service the routine for updating the SOC every two seconds. This routine implements several operations like to integrate the current read for calculating the voltage capacitor of V_{SOC} and to operate a lookup table to get the V_{OCV} value.

The code also contains interrupt functions which was used to capture the analog ADC counts each time that the PWM signal counter through zero. Peripherals details are described in detail on the next section.

The prototype purpose consists of controlling the output voltage converter to emulate a cell behavior. The algorithm is based on the prototype simulation which was explained on previous chapters. To simplify the code implementation the lookup table which links the V_{SOC} with V_{OCV} has been reduced to arrays of 95 values. So, in this case the battery cell is a representation of total values that lookup table originally have from CALCE group. The simplified lookup table has been represented on the following figure:

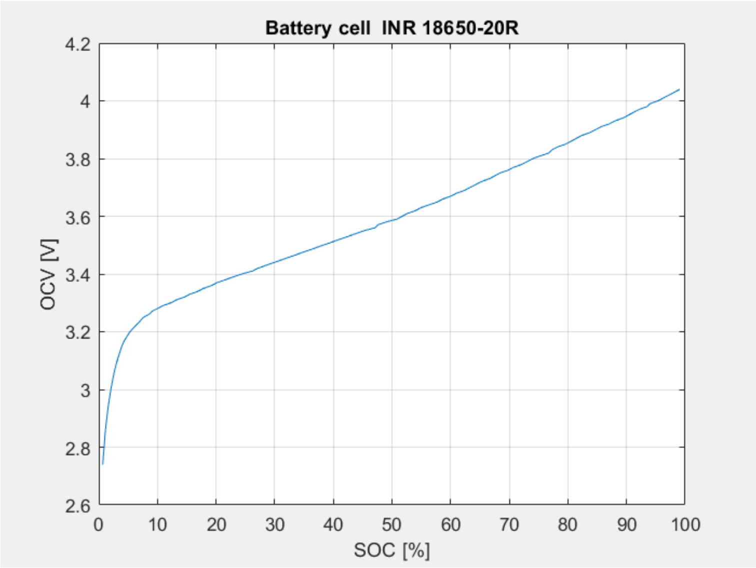


Figure 41. Lookup table loaded on the code which obtains the OCV based on SOC.

The variable "`socini`" defines the initial SOC value, and the algorithm decrease that SOC values updating the state of charge each time that this is evaluated on timer 1, considering the real current.

The main loop has several functions. It processes the input ADC values, it adjusts the voltage reference, coming from timer 1, for the proportional integrator controller (PI), and finally, it applies the duty cycle update to control the mosfet gate of buck converter.

The function used to implement the controller has been extracted from Texas instruments library. Concretely, the function used is "DCL_runPI_C3" which its diagram is showed on the next figure.

The proportional constant "Kp" and "ki" has been used the same values as the simulation level. The microcontroller code function also considers the anti-windup up operation to limit the integration maximum values.

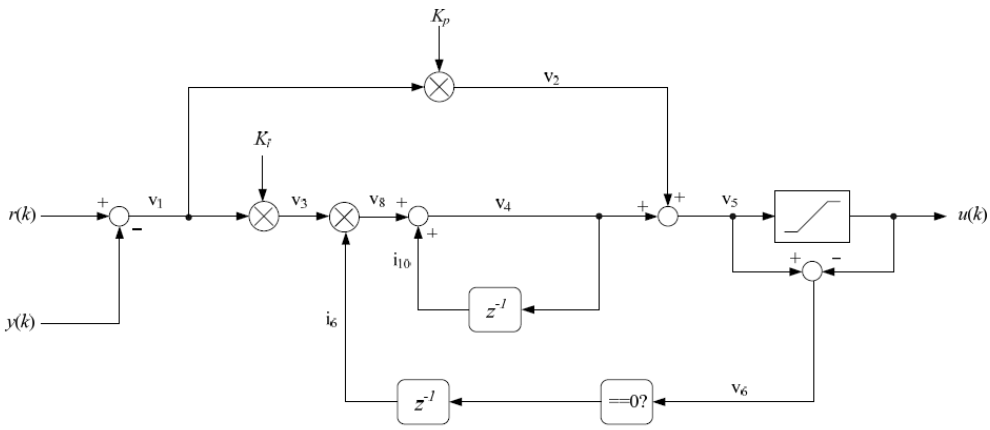


Figure 42. DCL_runPI_C3 architecture extracted from DCL [19]

Timer, analog to digital converters (ADC) and PWM are the peripherals used to develop this program. The predefined libraries from Texas Instruments facilitates these peripherals configurations.

Two ADC have been used to read the output current converter and voltage output simultaneously. The ADCCLK has been divided to 4 using the prescaler due to high device system frequency. The ADC trigger is synchronized with the output PWM which sets an interruption each time PWM through cero. This interruption captures the ADC counts, and it saves them to an array due to noise reduction. The ADC resolution used is 12 bits which is enough to read the output voltage and current. Different acquisition methods have been used to capture the original analog values. A resistor voltage divider has been used to capture the output voltage, with that way the maximum ADC input value within allowed range has been assured. The output current has been captured through shunt resistor. The operational amplifier circuit has been used to adapt the signal to the ADC. The ADC counts have been captured simultaneously through an interrupt which is triggered each time that PWM counter through zero.

The PWM is used to control the gate for the Mosfet buck converter. The desired frequency is 100kHz due to frequency commutation converter. The counter mode used is the counter up which is the default one. Although, there are two PWM set for each channel, only the PWM A is used, because the converter is unidirectional with only one Mosfet to be controlled. The PWM module has 12 bits with a prescaler option in case the period requested does not has the possibility to set in its registers.

The following figure shows the software diagram which represents the code implemented. The focus blocks are the timer which represents the battery cell control. Then, the main block manages the compensator voltage to convert an appropriate duty cycle.

Thanks to Texas instruments preconfigured peripherals, there was not necessary to set values at register level. For instance, the register TBPRD (Time-Base Period Register) was necessary to configure the PWM, but at Texas Instruments function level was not necessary to set that register from the user. The code used for this prototype implementation has been included on the annex section with all details [A.1]

The next diagram is a graphical representation of the software code implemented. The main blocks highlighted are the interruptions, the timer peripheral, and the main block function. All those blocks would be nothing without a right initialization of that modules.

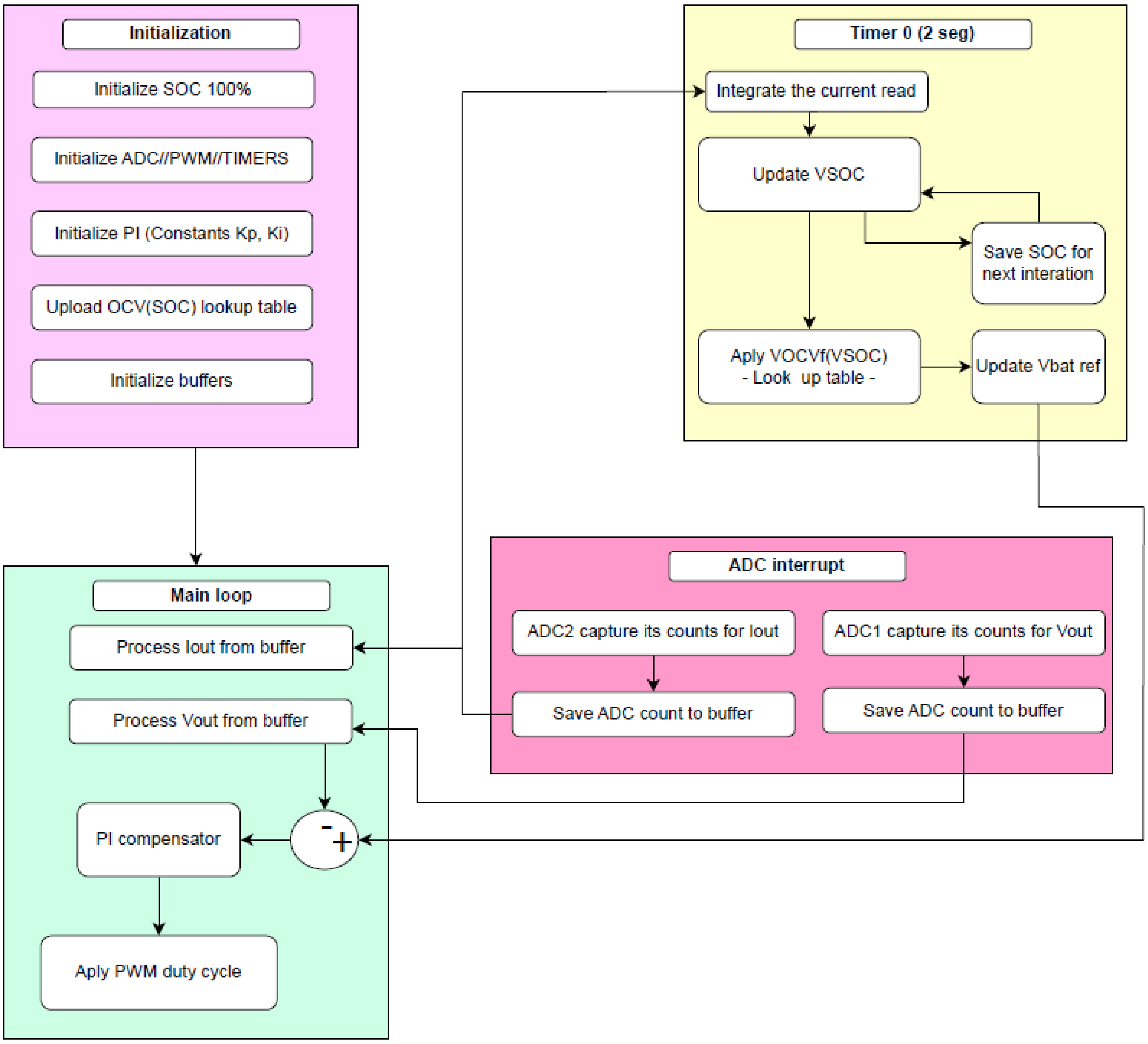


Figure 43. Software block diagram to emulate the cell.

3.4 Experimental results

Firstly, the PWM signal has been monitored to validate the right DC-DC converter behavior. The test has been performed on two duty cycle points configured directly on the micro. The desired voltages have been obtained on the figures below according to its duty cycle.

$$V_{out} = d \cdot V_{in} = 0.2 \cdot 12 = 2.4\ V$$

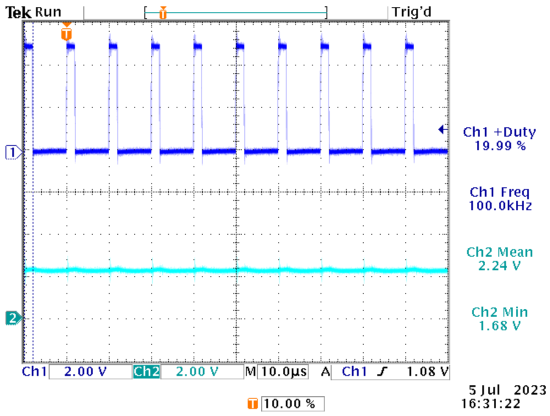


Figure 44. Output converter voltage with a duty cycle of 20%

$$V_{out} = d \cdot V_{in} = 0.3 \cdot 12 = 3.6\ V$$

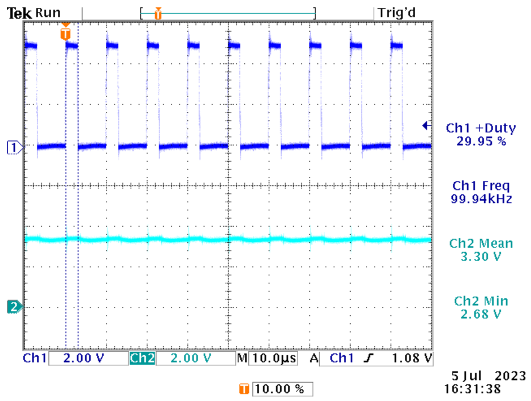


Figure 45. Output converter voltage with a duty cycle of 30%

The battery cell prototype has been tested discharging the battery from 100% SOC until 0.67% SOC. The total time that the prototype battery cell has been tested is approximately 3 hours. The data has been extracted during debug period from Code Composer Studio (CCS). This program has the option to capture the values exporting the JSON files. This experiment has been exported these JSON files every 3 minutes. The annex section includes one JSON example file. [A.3]

The signals captured are the state of charge, the current read through a shunt resistor, the voltage output read from ADC, and the expected voltage converter reference. All these signals have been graphed and analyzed below.

The first graph reported is the relation between V_{SOC} and V_{BAT}. The buck converter meets the requirements assigned for the voltage reference even though there are small fluctuations due to existing noise.

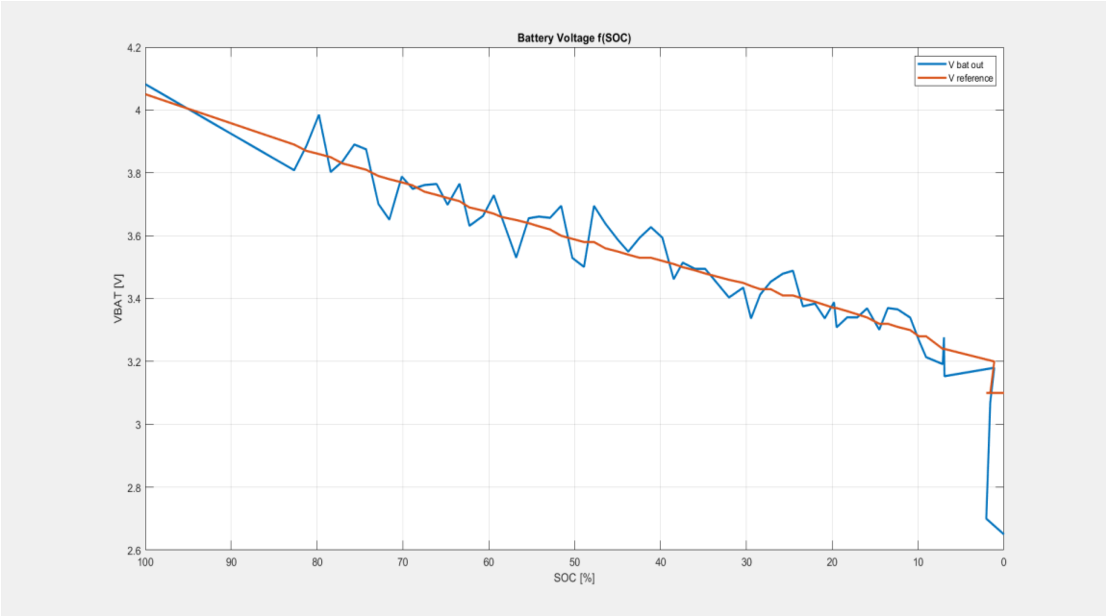


Figure 46. Battery cell discharge voltage V_{BAT} in function of SOC

The test has been performed with constant load of 3.3 Ω. The following graph shows a current reported which is approximately constant of 1 despite off existing noise.

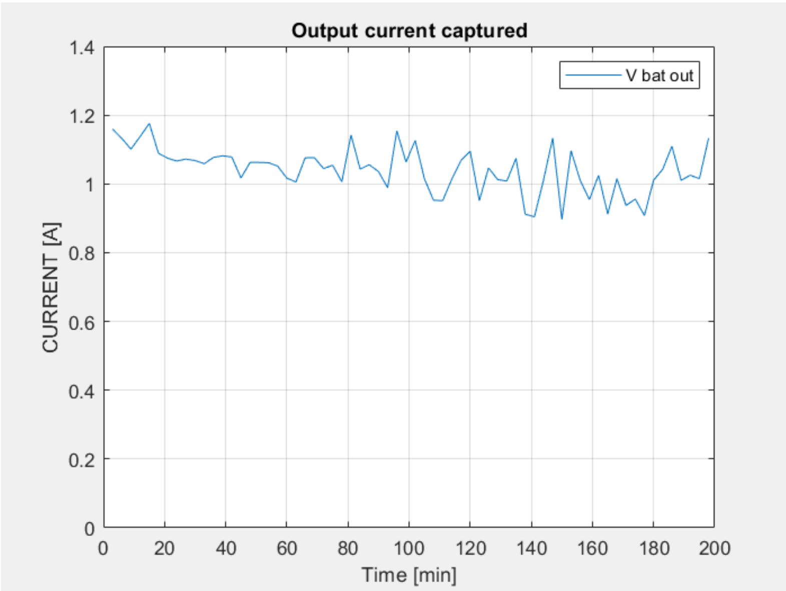


Figure 47. Output current captures which is approximately constant during all test

Finally, the next graph is the result of discharging the battery cell as expected for 3 hours approximately. So, the experimental results can be concluded that the battery cell prototype works successfully.

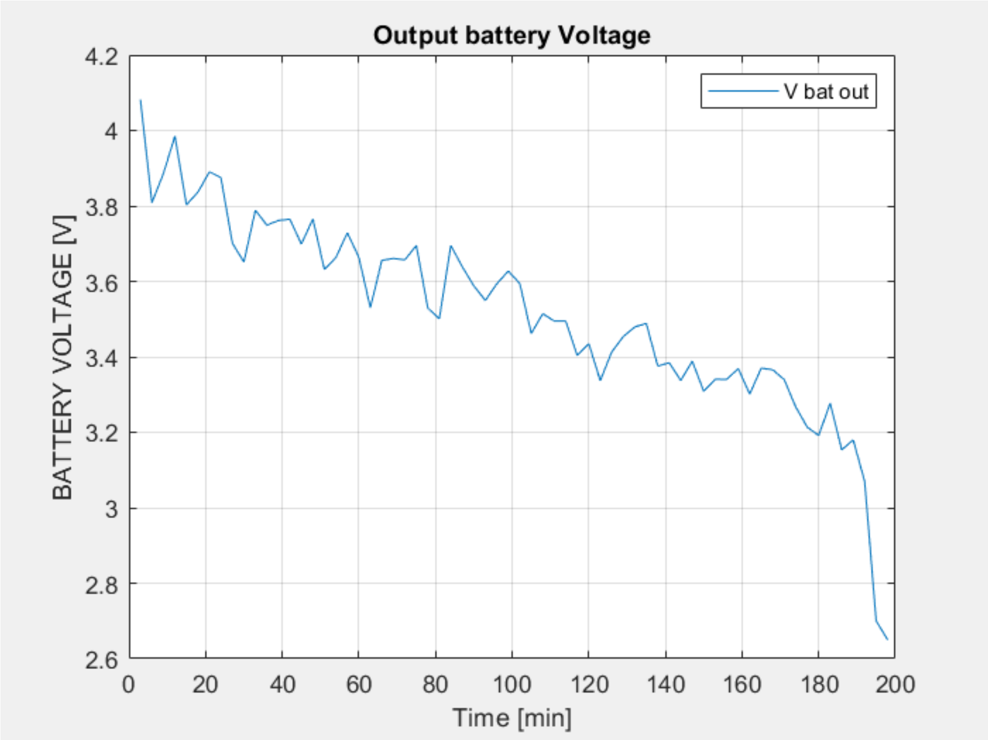


Figure 48. Battery cell voltage discharge

The most appropriate experimental method to capture all battery cell was capturing the points every 3 minutes through json files. However, the oscilloscope has also used to captured one specific point to verify the right behavior. Although, the output signal is not totally clean due to output capacitor filter, the output voltage reaches the expected value of 3.9 V in that case. The shunt value has been captured as well to verify the right current measured which reports 1.58 A.

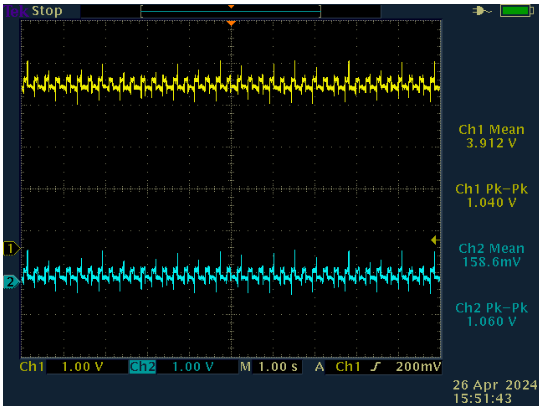


Figure 49. The output voltage and shunt voltage signal

4 Next steps conclusions

The first requirements proposed has been achieved for this preliminary prototype release. From academic point of view, different topics have been developed in this project. The model designed under Simulink, the real implementation on a customized PCB, and code programed. Moreover, the converter design concepts as well as energy storage concepts has been applied on this project. It is a suitable project for this master, I can say that I have learned a lot from an academic point of view.

Although, the results achieved fulfill the expectations proposed, there are more possible requirements that this project could have been implemented. There are described below several proposals to improve the next prototype releases.

Firstly, not all cell model parameters have been modeled. The SOC vs OCV curve would have been more accurately if the dynamics behaviors have been considered, and more points have been represented on the MCU look up table. Parameters such as the hysteresis or diffusion effects could be considered as well to improve the model representation. Thanks to these improvements, the emulated cells INR 18650-20R would be more accurately to the reality reducing the differences between real cell and the emulated one.

Secondly, it should be considered to escalate that prototype for an entire battery pack model which emulates a battery model to be applied for an electric vehicle. Whit this proposed topology the cells should be balanced to equalize the complete battery cells.

Thirdly, The PCB dimensions could be reduced for new cell design. The components could be smaller as well the distance between them. The output voltage could be better filtered if more capacitors in parallel would be considered.

Fourthly, second close loop which controls the output current could be also considered. The SOC should be updated more accurately with this second loop. Additionally, bidirectional converter which allows to charge and discharge the battery cell model could be also designed.

Fifth, estimations parameters could be considered for the next prototype releases. The Sate Of Health (SOH), Deep Of Discharge (DOD), State Of Power (SOP), among others estimation parameters which performed a better battery emulation model. The SOH consider that the cell will be aging during their life, so the parameters of the cell are degraded. The SOH can estimate these new parameters and provide a quantifiable measurement of the health cell, which are based in their parameters and their capacity compared to original ones. Furthermore, the temperature dependences should be considered in the model which will improve the SOC profile in different circumstances.

Finally, I would like to acknowledge to Carlos Olalla, professor at Universitat Rovira I Virgili (URV) for the guidance during the project development as well as to keep me motivated to achieve this actual prototype results.

5 References

[1] Extended Kalman filtering for battery management systems of LiPB-based HEV battery packs Part 1. Background – Gregory L. Plett - Journal of Power Sources 134 (2004) 252–261.

[2] Extended Kalman filtering for battery management systems of LiPB-based HEV battery packs Part 2. Modelling and identification– Gregory L. Plett - Journal of Power Sources 134 (2004) 262–276.

[3] Extended Kalman filtering for battery management systems of LiPB-based HEV battery packs Part 3. State and parameter estimation – Gregory L. Plett - Journal of Power Sources 134 (2004) 277–292.

[4] Datasheet cell battery INR18650-20R

[5] Battery cell profile from Calce group

[6] Datasheet driver MOSFET IR2125.

[7] Launchpad F28379D description

[8] TMS320F28379D Datasheet

[9] National Instruments product description TMS320F28379D

[10] Reference Manual TMS320F28379D

[11] Datasheet Inductor Murata

[12] Datasheet Mosfet IRFP4110PbF

[13] Datasheet Schottky diode, 30A, 100V, ref MBR30H100CTG

[14] Datasheet voltage regulator L7805CV, 1.5A

[15] Electric vehicle architecture

[16] Battery comparison discharge

[17] Battery cell simulation from COMENSO

[18] Voltaic Pile

[19] C2000 Digital Control Library

[20] Energy conversion and Storage [ECE] lectures

[21] Modeling and Control of Switching Converters [MCC] lectures.

6 Annexes

6.1 Code implemented

```
//#############################################################
// FILE:   Main.c
//
// TITLE:  Li-ion battery cell emulator based on a buck converter
//
//###########################################################################
//
// Included Files
//
#include "driverlib.h"
#include "device.h"
#include "board.h"
#include "DCLF32.h"
#include "DCL.h"
// Defines
#define RESULTS_BUFFER_SIZE     256
#define EX_ADC_RESOLUTION       12
#define TABLE_SIZE 165
// Globals
volatile uint16_t bufferFull;                // Flag to indicate buffer is full
volatile uint16_t cbufferFull;                       // Flag to indicate buffer is full
uint16_t adcAResults[RESULTS_BUFFER_SIZE];   // Buffer for results
uint16_t adccurrent[RESULTS_BUFFER_SIZE];    // Buffer for results
uint16_t index,cindex;                         // Index into result buffer
uint16_t cpuTimer0IntCount,cpuTimer1IntCount;
float32_t voltageread,currentread,currentreadadc;
float32_t promedio,promediocurr;
float32_t dutty,dk,ref,refpi,refs,t;
uint16_t id;
float32_t soc,socini,csoc,intcurr,vsoc, intcurrbef;

//define the SOC lookup table
uint16_t lookup_vocv[TABLE_SIZE] =
{260,261,262,263,264,265,266,267,268,269,270,271,272,273,274,275,276,277,278,279,280,281,282,2
83,284,285,286,287,288,289,290,291,292,293,294,295,296,297,298,299,300,301,302,303,304,305,306
,307,308,309,310,311,312,313,314,315,316,317,318,319,320,321,322,323,324,324,325,326,327,328,3
29,330,331,331,332,333,334,335,336,337,338,339,339,340,341,341,342,343,343,344,345,345,346,347
,348,349,349,350,351,352,352,353,354,355,356,357,358,359,360,360,361,362,363,364,364,365,366,3
67,368,368,369,370,370,371,372,373,374,374,375,376,377,378,379,379,380,381,381,382,383,384,385
,386,387,388,389,390,391,392,393,393,394,395,396,397,398,399,399,400,401,402,403,403,404,405,4
06};

uint16_t lookup_vsoc[TABLE_SIZE] =
{67,113,160,206,253,299,345,392,438,531,577,624,670,717,763,856,902,995,1088,1227,1320,1460,15
52,1692,1785,1924,2017,2156,2295,2435,2620,2713,2852,2992,3131,3270,3409,3549,3688,3827,3967,4
106,4245,4384,4524,4709,4756,4895,5081,5174,5266,5406,5499,5638,5777,5870,6009,6102,6241,6334,
6427,6520,6659,6752,6845,6984,7077,7216,7309,7402,7541,7681,7727,7820,7959,8052,8145,8238,8377
,8470,8563,8702,8795,8934,9027,9120,9213,9352,9398,9538,9630,9723,9816,10000};

EPWM_SignalParams pwmSignal ={50000, 0.1f, 0.7f, true, DEVICE_LSPCLK_FREQ,
SYSCTL_EPWMCLK_DIV_1,EPWM_COUNTER_MODE_UP, EPWM_CLOCK_DIVIDER_1,
                EPWM_HSCLOCK_DIVIDER_1};

//
//  Functions defined
//
void initADC(void);
void initEPWM(void);
void initADCSOC(void);
void applydutty(float vc);
__interrupt void adcA1ISR(void);
__interrupt void cpuTimer0ISR(void);
__interrupt void cpuTimer1ISR(void);
void initCPUTimers(void);
void configCPUTimer(uint32_t, float, float);
DCL_PI pid1 = PI_DEFAULTS;
```

```
//
// Main
//
void main(void)
{
    // Initialize device clock and peripherals
    Device_init();

    // Disable pin locks and enable internal pull-ups.
    Device_initGPIO();
    // Initialize PIE and clear PIE registers. Disables CPU interrupts.
    Interrupt_initModule();

    // Initialize the PIE vector table with pointers to the shell Interrupt
    // Service Routines (ISR).
    Interrupt_initVectorTable();

    // For this case just init GPIO pins for ePWM1
    Board_init();

    // ISRs for each CPU Timer interrupt
    Interrupt_register(INT_TIMER0, &cpuTimer0ISR);
    Interrupt_register(INT_TIMER1, &cpuTimer1ISR);

    //
    // Initializes the Device Peripheral. For this example, only initialize the
    // Cpu Timers.
    //
    initCPUTimers();

    //
    // Configure CPU-Timer 0, 1, and 2 to interrupt every second:
    //
    configCPUTimer(CPUTIMER0_BASE, DEVICE_SYSCLK_FREQ, 1000000);
    configCPUTimer(CPUTIMER1_BASE, DEVICE_SYSCLK_FREQ, 2000000);
    CPUTimer_enableInterrupt(CPUTIMER0_BASE);
    CPUTimer_enableInterrupt(CPUTIMER1_BASE);
    Interrupt_enable(INT_TIMER0);
    Interrupt_enable(INT_TIMER1);
    CPUTimer_startTimer(CPUTIMER0_BASE);
    CPUTimer_startTimer(CPUTIMER1_BASE);
    // Interrupts that are used  re-mapped to ISR functions
    Interrupt_register(INT_ADCA1, &adcA1ISR);
    // Set up the ADC and the ePWM and initialize the SOC
    initADC();
    initEPWM();
    initADCSOC();
    //
    // Enable ADC interrupt
    //
    Interrupt_enable(INT_ADCA1);
    //
    // Enable global Interrupts and higher priority real-time debug events:
    //
    EINT;  // Enable Global interrupt INTM
    ERTM;  // Enable Global realtime interrupt DBGM

    // Start ePWM1, enabling SOCA and putting the counter in up-count mode
    EPWM_enableADCTrigger(EPWM1_BASE, EPWM_SOC_A);

    //Inizialize the PI Controller
    /* initialise controller variables */

    pid1.Kp = 0.3f;
    pid1.Ki = 0.27f;
    pid1.Umax = 1.0f;
    pid1.Umin = 0;
    pid1.i6 = 0;
    pid1.i10 = 0;
    pid1.i11 = 0;

    ref = 4;
    socini = 1;
    intcurrbef = 0.0;
```

```
// Initialize results buffer
    for(index = 0; index < RESULTS_BUFFER_SIZE; index++)
    {
        adcAResults[index] = 0;
    }
    for(cindex = 0; cindex < RESULTS_BUFFER_SIZE; cindex++)
    {
        adccurrent[cindex] = 0;
    }

    index = 0;
    bufferFull = 0;
    cindex = 0;
    cbufferFull = 0;
    applydutty(ref);

    //
    // IDLE loop. Just sit and loop forever (optional):
    //
    for(;;)
    {
        asm ("  NOP");
        promedio = 0;
        suma = 0;
         for (i=0; i<RESULTS_BUFFER_SIZE; i++)
         {
                suma += adcAResults[i];
         }

         promedio = suma / RESULTS_BUFFER_SIZE;
         voltageread = ((promedio*0.003F) - 3.35F);
         promediocurr = 0;
         suma = 0;

         for (i=0; i<RESULTS_BUFFER_SIZE; i++)
         {
                suma += adccurrent[i];
         }

         promediocurr = suma / RESULTS_BUFFER_SIZE;
         currentreadadc =  (((3-0.01)/4096)*promediocurr)+ 0.01;
         currentread = (((currentreadadc-0.46)/13.90)*10);

         //RUN PI Controller
         dk = DCL_runPI_C3(&pid1, ref, voltageread);
         refpi = dk+ref;
         applydutty(refpi);
         }
}

void initEPWM (void)
{
    //
    // Configuring ePWM module for desired frequency and duty
    //
    EPWM_configureSignal(myEPWM1_BASE, &pwmSignal);

    //
    // ePWM1 SYNCO is generated on CTR=0
    //
    EPWM_setSyncOutPulseMode(myEPWM1_BASE, EPWM_SYNC_OUT_PULSE_ON_COUNTER_ZERO);

    //
    // Configure the SOC to occur on the first up-count event
    //
    EPWM_setADCTriggerSource(myEPWM1_BASE, EPWM_SOC_A, EPWM_SOC_TBCTR_U_CMPA);
    EPWM_setADCTriggerEventPrescale(myEPWM1_BASE, EPWM_SOC_A, 1);
}

void initADC(void)
{
   //
    // Set ADCDLK divider to /4
    //
    ADC_setPrescaler(ADCA_BASE, ADC_CLK_DIV_4_0);
    //
```

```
    // Set resolution and signal mode (see #defines above) and load
    // corresponding trims.
    //
#if(EX_ADC_RESOLUTION == 12)
    ADC_setMode(ADCA_BASE, ADC_RESOLUTION_12BIT, ADC_MODE_SINGLE_ENDED);
#elif(EX_ADC_RESOLUTION == 16)
    ADC_setMode(ADCA_BASE, ADC_RESOLUTION_16BIT, ADC_MODE_DIFFERENTIAL);
#endif

    //
    // Set pulse positions to late
    //
    ADC_setInterruptPulseMode(ADCA_BASE, ADC_PULSE_END_OF_CONV);
    //
    // Power up the ADC and then delay for 1 ms
    //
    ADC_enableConverter(ADCA_BASE);
    DEVICE_DELAY_US(1000);
}

void initADCSOC(void)
{
    //
    // Configure SOC0 of ADCA to convert pin A0. The EPWM1SOCA signal will be
    // the trigger.
    // - For 12-bit resolution, a sampling window of 15 (75 ns at a 200MHz
    //   SYSCLK rate) will be used.  For 16-bit resolution, a sampling window
    //   of 64 (320 ns at a 200MHz SYSCLK rate) will be used.
    // - NOTE: A longer sampling window will be required if the ADC driving
    //   source is less than ideal (an ideal source would be a high bandwidth
    //   op-amp with a small series resistance). See TI application report
    //   SPRACT6 for guidance on ADC driver design.
    //


#if(EX_ADC_RESOLUTION == 12)
    ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER0, ADC_TRIGGER_EPWM1_SOCA,
                 ADC_CH_ADCIN0, 15);
    ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER2, ADC_TRIGGER_EPWM1_SOCA,ADC_CH_ADCIN2, 15);
#elif(EX_ADC_RESOLUTION == 16)
    ADC_setupSOC(ADCA_BASE, ADC_SOC_NUMBER0, ADC_TRIGGER_EPWM1_SOCA,
                 ADC_CH_ADCIN0, 64);
#endif

    //
    // Set SOC0 to set the interrupt 1 flag. Enable the interrupt and make
    // sure its flag is cleared.
    //
    ADC_setInterruptSource(ADCA_BASE, ADC_INT_NUMBER1, ADC_SOC_NUMBER0);
    ADC_enableInterrupt(ADCA_BASE, ADC_INT_NUMBER1);
    ADC_clearInterruptStatus(ADCA_BASE, ADC_INT_NUMBER1);
}

//
// ADC A Interrupt 1 ISR
//
__interrupt void adcA1ISR(void)
{
    //
    // Add the latest result to the buffer
    //
    adcAResults[index++] = ADC_readResult(ADCARESULT_BASE, ADC_SOC_NUMBER0);
    adccurrent[cindex++] = ADC_readResult(ADCARESULT_BASE, ADC_SOC_NUMBER2);
    //
    // Set the bufferFull flag if the buffer is full
    //
    if(RESULTS_BUFFER_SIZE <= index)
    {
        index = 0;
        bufferFull = 1;
    }
    if(RESULTS_BUFFER_SIZE <= cindex)
    {
        cindex = 0;
        cbufferFull = 1;
    }
```

```
    //
    // Clear the interrupt flag
    //
    ADC_clearInterruptStatus(ADCA_BASE, ADC_INT_NUMBER1);

    //
    // Check if overflow has occurred
    //
    if(true == ADC_getInterruptOverflowStatus(ADCA_BASE, ADC_INT_NUMBER1))
    {
        ADC_clearInterruptOverflowStatus(ADCA_BASE, ADC_INT_NUMBER1);
        ADC_clearInterruptStatus(ADCA_BASE, ADC_INT_NUMBER1);
    }

    //
    // Acknowledge the interrupt
    //
    Interrupt_clearACKGroup(INTERRUPT_ACK_GROUP1);
}


void configCPUTimer(uint32_t cpuTimer, float freq, float period)
{
    uint32_t temp;

    //
    // Initialize timer period:
    //
    temp = (uint32_t)(freq / 1000000 * period);
    CPUTimer_setPeriod(cpuTimer, temp);

    CPUTimer_setPreScaler(cpuTimer, 0); //Set pre-scale counter to divide by 1
(SYSCLKOUT):
    CPUTimer_stopTimer(cpuTimer);
    CPUTimer_reloadTimerCounter(cpuTimer);
    CPUTimer_setEmulationMode(cpuTimer,CPUTIMER_EMULATIONMODE_STOPAFTERNEXTDECREMENT);
    CPUTimer_enableInterrupt(cpuTimer);

    //
    // Resets interrupt counters for the three cpuTimers
    //
    if (cpuTimer == CPUTIMER0_BASE)
    {
        cpuTimer0IntCount = 0;
    }
    else if(cpuTimer == CPUTIMER1_BASE)
    {
        cpuTimer1IntCount = 0;
    }
}

void initCPUTimers(void)
{
    //
    // Initialize timer period to maximum
    //
    CPUTimer_setPeriod(CPUTIMER0_BASE, 0xFFFFFFFF);
    CPUTimer_setPeriod(CPUTIMER1_BASE, 0xFFFFFFFF);

    //
    // Initialize pre-scale counter to divide by 1 (SYSCLKOUT)
    //
    CPUTimer_setPreScaler(CPUTIMER0_BASE, 0);
    CPUTimer_setPreScaler(CPUTIMER1_BASE, 0);

    //
    // Make sure timer is stopped
    //
    CPUTimer_stopTimer(CPUTIMER0_BASE);
    CPUTimer_stopTimer(CPUTIMER1_BASE);

    //
    // Reload all counter register with period value
    //
```

```
    CPUTimer_reloadTimerCounter(CPUTIMER0_BASE);
    CPUTimer_reloadTimerCounter(CPUTIMER1_BASE);

    //
    // Reset interrupt counter
    //
    cpuTimer0IntCount = 0;
    cpuTimer1IntCount = 0;
}

//
// cpuTimer0ISR - Counter for CpuTimer0
//
__interrupt void cpuTimer0ISR(void)
{
    cpuTimer0IntCount++;

   //
    // Acknowledge this interrupt to receive more interrupts from group 1
    //
    Interrupt_clearACKGroup(INTERRUPT_ACK_GROUP1);
}

//
// cpuTimer1ISR - Counter for CpuTimer1
//
__interrupt void cpuTimer1ISR(void)
{
         csoc = 7200;
         intcurr = (currentread*2) + intcurrbef;
         intcurrbef = intcurr;
         vsoct = intcurr/csoc;
         vsoc = socini - vsoct;
         for (i = TABLE_SIZE; i >= 1; i--) {
                max = ((float)lookup_vsoc[i-1])/10000;
                min = ((float)lookup_vsoc[i-2])/10000;
                if ((vsoc <= max)&(vsoc >= min)) {
                       id = i-1; // Guardamos la posición
                       break; // Salimos del bucle
                }
         }
         refs = (float)lookup_vocv[id]/100;
         ref = refs - (currentread*0.1);    //
    // The CPU acknowledges the interrupt.
    //
         cpuTimer1IntCount++;
  }

void applydutty(float vc) {
    dutty = vc /12;
    if (dutty>0.5 || dutty==0) {
        dutty = 0.5;
    }
    pwmSignal.dutyValA = dutty;
    EPWM_configureSignal(myEPWM1_BASE, &pwmSignal);
}
//
// End of file
//
```

6.2 Matlab live script

```
clearvars
clc
syms vo il; %define state vars vector
syms d vg io; %define control vector
syms R L C Ts; %define vars
x=[il;vo]
```

x =

$$\begin{pmatrix} \mathrm{il} \\ \mathrm{vo} \end{pmatrix}$$

```
u=[vg;d;io] %input vector
```

u =

$$\begin{pmatrix} \mathrm{vg} \\ d \\ \mathrm{io} \end{pmatrix}$$

Static equations definition

```
f1 = [d*((vg/L)-(vo/L));d*((il/C)-(vo/(R*C))-(io/C))]
```

f1 =

$$\begin{pmatrix} d\,\left(\frac{\mathrm{vg}}{L} - \frac{\mathrm{vo}}{L}\right) \\ -d\,\left(\frac{\mathrm{io}}{C} - \frac{\mathrm{il}}{C} + \frac{\mathrm{vo}}{C\,R}\right) \end{pmatrix}$$

```
f2 = [(1-d)*(-vo/L);(1-d)*((il/C)-(vo/(R*C))-(io/C))]
```

f2 =

$$\begin{pmatrix} \frac{\mathrm{vo}\,(d-1)}{L} \\ (d-1)\,\left(\frac{\mathrm{io}}{C} - \frac{\mathrm{il}}{C} + \frac{\mathrm{vo}}{C\,R}\right) \end{pmatrix}$$

```
fav = f1+f2;
fav = simplify(fav)
```

fav =

$$\begin{pmatrix} -\frac{\mathrm{vo} - d\,\mathrm{vg}}{L} \\ -\frac{\mathrm{vo} - R\,\mathrm{il} + R\,\mathrm{io}}{C\,R} \end{pmatrix}$$

State space model

```
A= jacobian(fav,x);
A = simplify(A)
```

A =

$$\begin{pmatrix} 0 & -\frac{1}{L} \\ \frac{1}{C} & -\frac{1}{C\,R} \end{pmatrix}$$

```
B= jacobian(fav,u);
B = simplify(B)
```

B =

$$\begin{pmatrix} \frac{d}{L} & \frac{vg}{L} & 0 \\ 0 & 0 & -\frac{1}{C} \end{pmatrix}$$

Values definition for two static points A and B

```
Lx = 68e-6;
vgx=12;
Cx=220e-6;
Tsx=1e-5;
fs = 1/Tsx; %fs =100kHz
Ra=0.8;
voa=2;
Da=0.16;
Rb= 5;
vob=5;
Db=0.44;
```

Transfer function definition

```
% limit inferior de cela - PUNT A
A_num_a = subs(A, [L C R d], [Lx Cx Ra Da]);
B_num_a = subs(B, [L vg C d vo], [Lx vgx Cx Da voa]);
Ga = ss(double(subs(A_num_a)), double(subs(B_num_a)), eye(2),0);
Ga = zpk(Ga);
Gvd_a = Ga(2,2)
```

```
Gvd_a =


        8.0214e+08
  ------------------------
  (s^2 + 5682s + 6.684e07)


Continuous-time zero/pole/gain model.
```

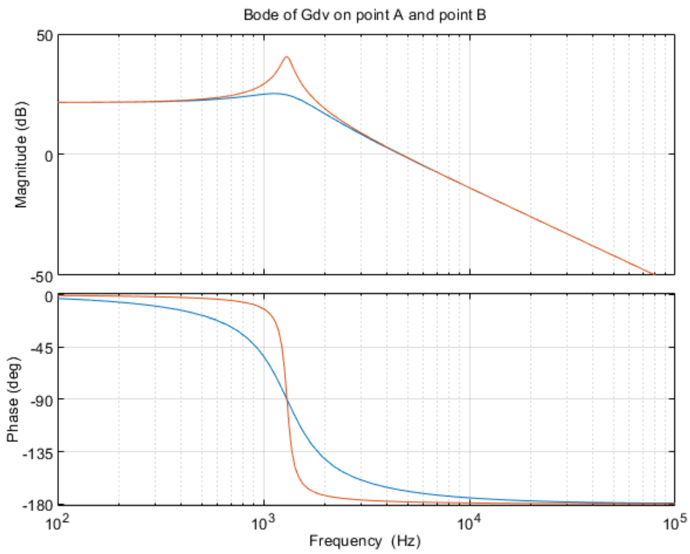
```
% limit superior de cela - PUNT
A_num_b = subs(A, [L C R d], [Lx Cx Rb Db]);
B_num_b = subs(B, [L vg C d vo], [Lx vgx Cx Db vob]);
Gb = ss(double(subs(A_num_b)), double(subs(B_num_b)), eye(2),0);
```

```
Gb = zpk(Gb);
Gvd_b = Gb(2,2)
```

```
Gvd_b =

 
          8.0214e+08
  -------------------------
  (s^2 + 909.1s + 6.684e07)
 
 Continuous-time zero/pole/gain model.
```

Bode representation

```
options = bodeoptions;
options.Grid = 'on';
options.FreqUnits = 'Hz';
options.Title.String = 'Bode of Gdv on point A and point B';
bode(Gvd_a,Gvd_b,options)
```



Comepensator Gc design

```
Gc = zpk(-5300,0,0.003)
```

```
Gc =
 
  0.003 (s+5300)
  --------------
        s
 
Continuous-time zero/pole/gain model.
```

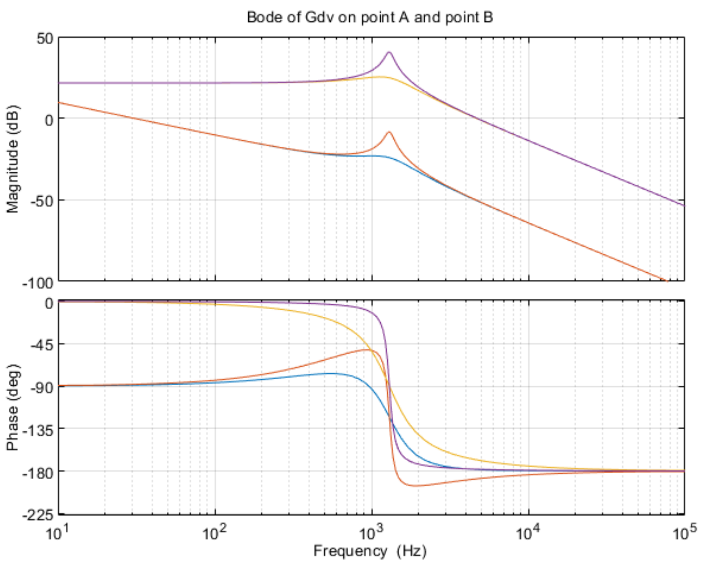
```
Ta= Gc*Gvd_a
```

```
Ta =

      2.4064e06 (s+5300)
  --------------------------
  s (s^2 + 5682s + 6.684e07)

Continuous-time zero/pole/gain model.
```

```
Tb = Gc*Gvd_b
```

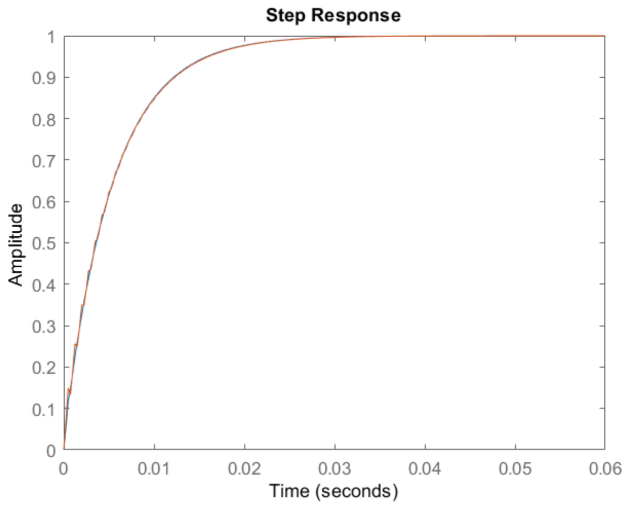
```
Tb =

      2.4064e06 (s+5300)
  ---------------------------
  s (s^2 + 909.1s + 6.684e07)

Continuous-time zero/pole/gain model.
```

```
bode(Ta,Tb,Gvd_a,Gvd_b,options)
```




```
step((Ta/(1+Ta)),(Tb/(1+Tb)))
```

```
Warning: Error creating or updating Line
Error in value of one or more of the following properties: XData YData
Array is wrong shape or size
```



Z Domain

```
f_sampling = 1*fs;
Gc_z = c2d(Gc, 1/f_sampling, 'tustin')
```

```
Gc_z =


  0.0030795 (z-0.9484)
  --------------------
         (z-1)


Sample time: 1e-05 seconds
Discrete-time zero/pole/gain model.
```

```
%prepare the parameters to put into simulink transfer function block
Gc__Z = get(Gc_z,'Z');
Gc__P = get(Gc_z,'P');
Gc__K = get(Gc_z,'K');
```

FREQUENCY RESPONSE ESTIMATOR

```
f = logspace (2, 5,100)
```

```
f = 1×100
10^5 ×
 0.0010    0.0011    0.0011    0.0012    0.0013    0.0014    0.0015 ⋯
```

defining digital parameters

```
DPWM_Bits = 10;
DPWM_max_voltage = 2.4;
DPWM_min_voltage = 0.4;
ADC_max_voltage = 3.3;
ADC_min_voltage = 0.5;
ADC_Bits = 12;
```

```
fclock = 60e6;
```

Battery cell definition

```
data=xlsread('SOC_OCV_Lookuptable_V5.xlsx');
SOC = data(:,1);
OCV = data(:,2);
SOCIN = linspace(1,0,8500)';
```

6.3 JSON files

The experimental values has been captured using a specific format that CCCS could export. This is an example of one capture exported to json files.


```
[
  {
        "module": "xdc.rov.runtime.Monitor",
        "view": "Variable",
        "args": {
               "name/addr": "cpuTimer0IntCount",
               "check": true}
  },
  {
        "address": "0xaed0",
        "name": "cpuTimer0IntCount",
        "character": "?",
        "int8_t": "3539",
        "int16_t": "3795",
        "int32_t": "124325587",
        "uint8_t": "3795",
        "uint16_t": "3795",
        "uint32_t": "124325587",
        "float32": "1.7533328205810946e-34"}
]
[
  {     "module": "xdc.rov.runtime.Monitor",
        "view": "Variable",
        "args": {
               "name/addr": "voltageread",
               "check": true}
  },
  {     "address": "0xaed4",
        "name": "voltageread",
        "character": "?",
        "int8_t": "4375",
        "int16_t": "4375",
        "int32_t": "1078792471",
        "uint8_t": "4375",
        "uint16_t": "4375",
        "uint32_t": "1078792471",
        "float32": "3.2041680812835693"}
]
[
  {     "module": "xdc.rov.runtime.Monitor",
        "view": "Variable",
        "args": {
               "name/addr": "currentread",
               "check": true}
  },
  {     "address": "0xaed6",
        "name": "currentread",
        "character": "?",
        "int8_t": "33201",
        "int16_t": "-32079",
        "int32_t": "1065648817",
        "uint8_t": "33457",
        "uint16_t": "33457",
        "uint32_t": "1065648817",
        "float32": "1.0352383852005005"}
]
[
  {     "module": "xdc.rov.runtime.Monitor",
        "view": "Variable",
        "args": {
               "name/addr": "ref",
               "check": true}
  },
  {     "address": "0xaee2",
        "name": "ref",
        "character": "?",
        "int8_t": "39179",
        "int16_t": "-26357",
```

```
          "int32_t": "1078761739",
          "uint8_t": "39179",
          "uint16_t": "39179",
          "uint32_t": "1078761739",
          "float32": "3.19684100151062" }
    ]
    [
      {       "module": "xdc.rov.runtime.Monitor",
              "view": "Variable",
              "args": {
                     "name/addr": "soc",
                     "check": true}
      },
      {       "address": "0xaeea",
              "name": "vsoc",
              "character": "?",
              "int8_t": "19980",
              "int16_t": "19980",
              "int32_t": "1054821900",
              "uint8_t": "19980",
              "uint16_t": "19980",
              "uint32_t": "1054821900",
              "float32": "0.43614232540130615"}
    ]
    [
      {       "module": "xdc.rov.runtime.Monitor",
              "view": "Variable",
              "args": {
                     "name/addr": "refs",
                     "check": true}
      },
      {       "address": "0xaee6",
              "name": "refs",
              "character": "?",
              "int8_t": "13107",
              "int16_t": "13107",
              "int32_t": "1079194419",
              "uint8_t": "13107",
              "uint16_t": "13107",
              "uint32_t": "1079194419",
              "float32": "3.299999952316284"}
    ]
```

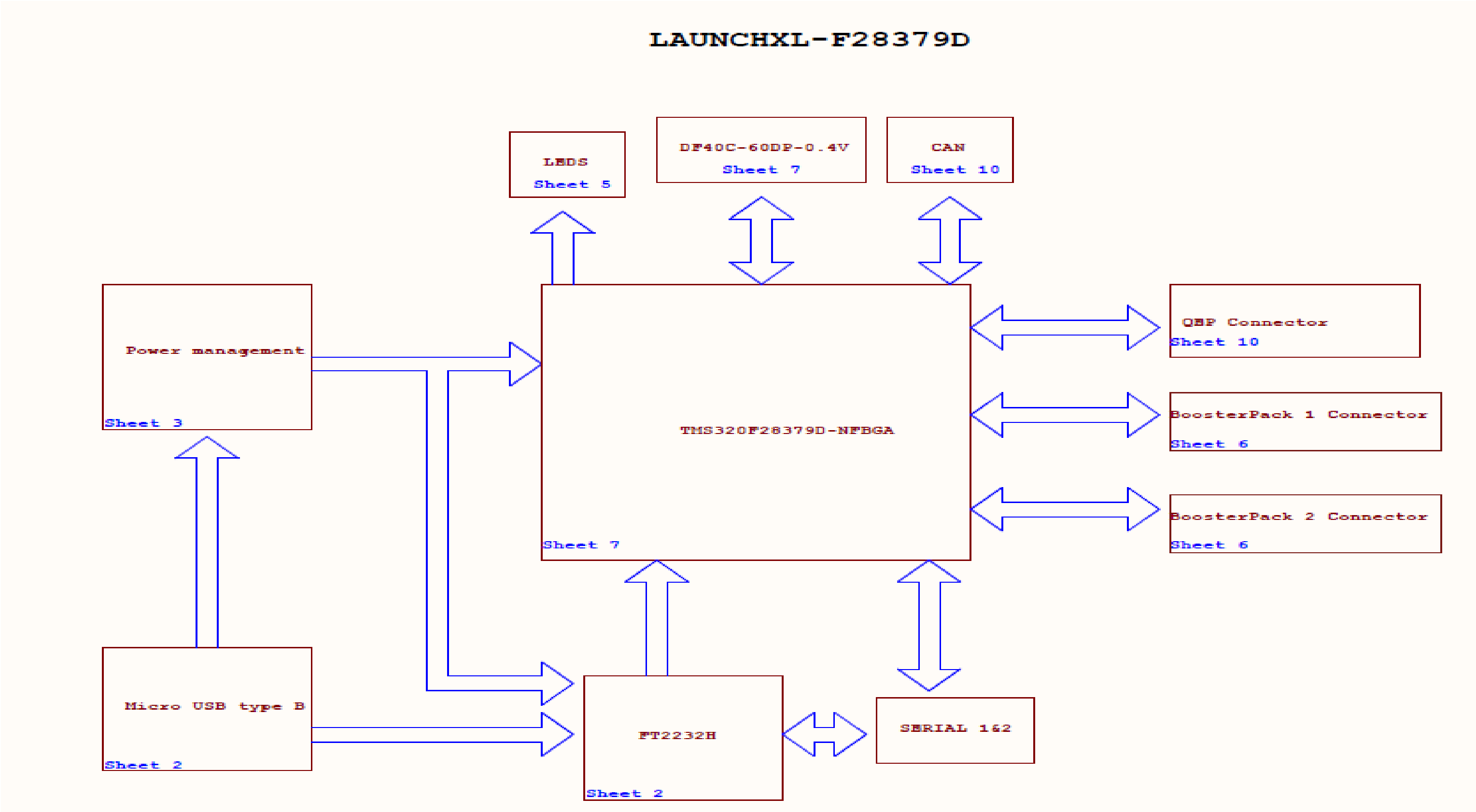
6.4 Component list

This table is the component list build for the custom PCB:

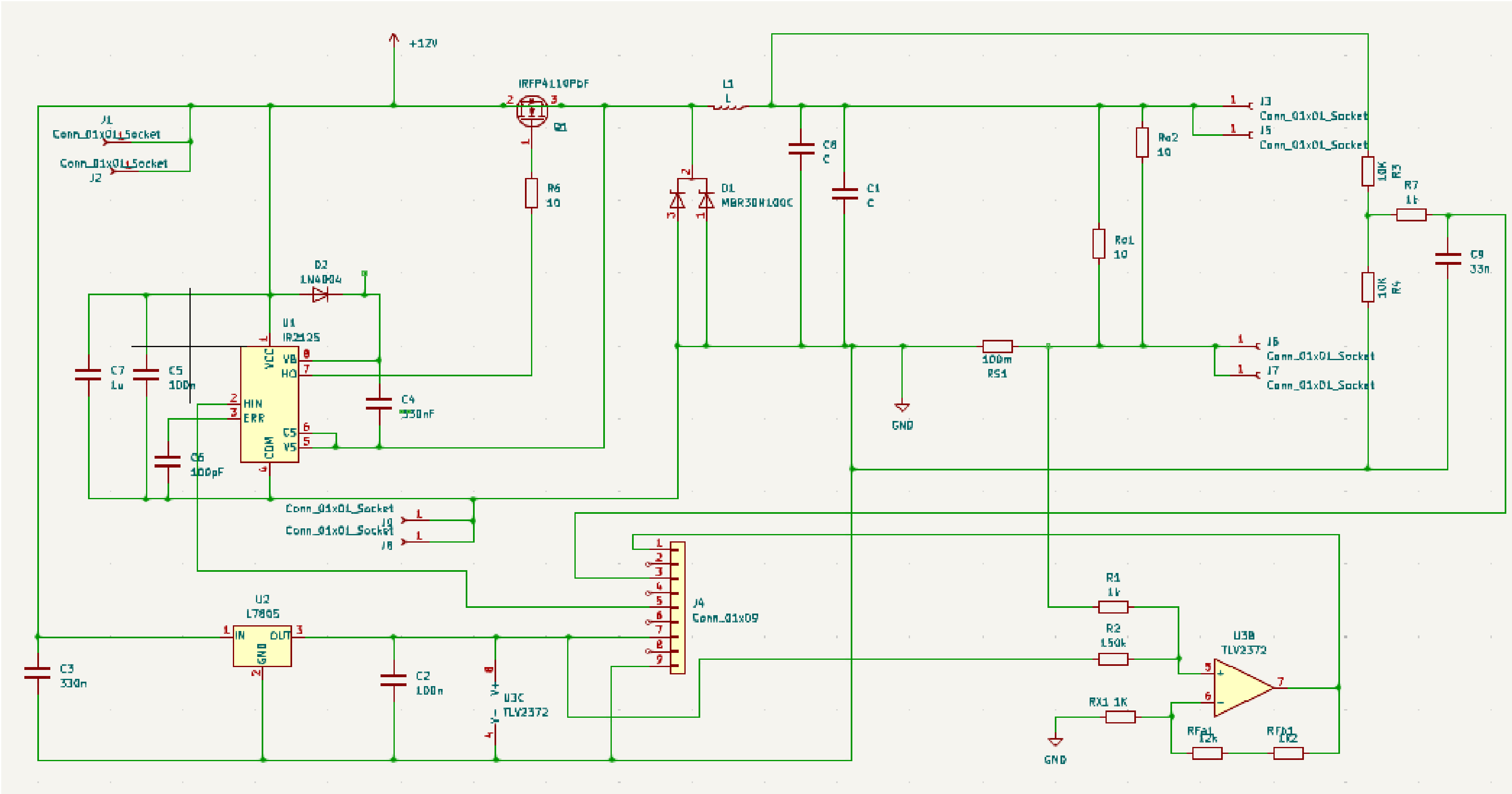
Component	Description	RS Code
U2	Regulador de tensión lineal, L7805CV, 1.5A 5 V TO-220 3 pines	298-8514
U1	Driver de puerta MOSFET CMOS, IR2125PBF	260-5185
U3	Amplificador operacional TLV2372IP Precisión	661-2066
N/A	Zócalo DIL Preci-Dip, paso de 2.54mm	702-0654
C5	Condensador de película WIMA AEC-Q200, 100nF	108-2700
C2	Condensador de película WIMA AEC-Q200, 100nF	108-2700
C3	Condensador de película EPCOS, 330nF,	874-1443
C4	Condensador de película EPCOS, 330nF,	874-1443
D2	Diode, 1N4004-E3/54, 1A	628-9029
Q1	Infineon HEXFET IRFP4110PBF N-Kanal,	688-6995
D1	Diodo, MBR30H100CTG, Rectificador Schottky, 30A	781-5679
C1	Condensador electrolítico Rubycon serie BXF, 220µF, ±20%	725-8990
R01	Resistencia Bourns, de 10Ω ±5%, 5W, Serie FW	775-7560
R02	Resistencia Bourns, de 10Ω ±5%, 5W, Serie FW	775-7560
RS1	Resistencia de Agujero Pasante, 0.1 ohm, PAC, 1 W, ± 1%, Axial	3547136
R1	Resistencia Bourns, de 1KΩ ±5%, 5W, Serie FW	707-7666
R2	Resistencia Bourns, de 150KΩ ±5%, 5W, Serie FW	151-094
RX1	Resistencia Bourns, de 1KΩ ±5%, 5W, Serie FW	151-094
RFA1	Resistencia Bourns, de 12KΩ ±5%, 5W, Serie FW	151-094
RBF1	Resistencia Bourns, de 1K2Ω ±5%, 5W, Serie FW	151-094
L1	Murata, 68 µH ±15%, Núcleo de bobina de ferrita, SRF máxima	170-9234
J4	Connector D-sub RS PRO, paso 2.77mm	528-3597
J1/J2	Conector hembra para pruebas Hembra, 10A, 50V, Contacto Plata, Rojo	103-6511
J3/J5	Conector hembra para pruebas Hembra, 10A, 50V, Contacto Plata, Rojo	103-6511
J6/J7	Conector hembra para pruebas Hembra, 10A, 50V, Contacto Plata, Negro	103-6527
J8/J9	Conector hembra para pruebas Hembra, 10A, 50V, Contacto Plata, Negro	103-6527

Table 8. Boom of material used for the custom PCB

6.5 Launchpad F28379D block diagram [7]



6.6 Power board schematic



6.7 PCB layout