INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Université Fédérale Toulouse Midi-Pyrénées

LAAS

Electrical & Computer Engineering Department

5th year - Automatic Electronic - ESPE

# INTERDISCIPLINARY PROJECT

# Scientific report

January 2021

Jérémy Aubert, *Student, INSA Toulouse,* Xavier Bourlot, *Student, INSA Toulouse,*
Clarisse Calmot, *Student, INSA Toulouse,* Pauline Combes, *Student, INSA Toulouse,*
Emma Devaud, *Student, INSA Toulouse,* Romain Gary, *Student, INSA Toulouse,*
Thibault Jean, *Student, INSA Toulouse,* Mélissa Malischewski, *Student, INSA Toulouse,*
Xiaohu Zhang, *Student, INSA Toulouse*

**Tutors :**

| | |
|---|---|
| Mr. Alexandre | BOYER |
| Mr. Thierry | ROCACHER |
| Mr. Christophe | ESCRIBA |

***Abstract*—The goal of our project was to build a system that ensures "intelligent" management of the energy produced by a photovoltaic panel. The system selects the source of energy: the main network or the photovoltaic production system.**

**Others significant purposes are: managing photovoltaic production, shaping the signal produced and providing real-time information on photovoltaic production and the system state. We designed and implemented a rectifying circuit (DC-to-DC converter) to control the voltage within a certain range. We used batteries as an electrical energy storage device. During experiments, our devices generated enough energy to support screen equipment. Our results show that there is a lot of energy that can be recovered in daily life.**

***Index Terms*—photovoltaic, harvesting energy system, battery management, MPPT, buck converter**

## 1. Acknowledgements

We would like to thank our three tutors M. Thierry Rocacher, M. Alexandre Boyer and M. Christophe Escriba for their unfailing support during these months of preparation. We would not have achieved such a level of technical complexity without the wise advice of M.Rocacher and its constant questioning to challenge us.
We warmly thank Mr.Shea for his support, his wise advice and his reviews.
We also wanted to thank Mr. Boyer for motivating us when the end of the semester and the tiredness took over , he was a great support to us in this adventure. In addition to that, his technical knowledge around EMC and PCB routing were very valuable.
We also thank M.Escriba for his technical skills on various subjects such as the PCB manufacture and mounting. Also, a special thanks to M.José Martin and M.Sebastien Di Mercurio for their technical support. Of course, we would like to thank the whole management team of the GEI for allowing us to buy all the material we needed and for putting GEI rooms at our disposal whenever we needed it.

## 2. Introduction

Over the last decades, the quantity of carbon dioxide and the depletion of natural resources have been threatening more and more the future of our planet. The development of renewable energy production has a lot of potential to replace many traditional and more polluting energy sources like coal and oil.
As part of the 5 ESPE class project, our objective is to create a system that ensures "intelligent" management of the energy produced by a photovoltaic panel specifically for our department at INSA Toulouse.
To favor the renewable energy source while guaranteeing no interruption of the electricity supply, the system selects the most relevant source of energy: the main network or a photovoltaic production system; to power screens and other

devices.
To address this research we propose to build two parts. The first one is a switching converter that controls the solar energy production and regulates the voltage for the different loads downstream, such as batteries. Secondly, we will demonstrate why a battery management system is vital to manage the batteries configuration, their charges and to ensure the system security. We will also present programmed software part to provide real-time information (current, voltage) on the battery system and its state. This article will detail the hardware and software design of our system.

## 3. Buck converter design

First, we needed to build a buck for the system. A buck converter is a DC-to-DC power converter which steps down voltage (while stepping up current) from its input (supply) to its output (load). In our system, a buck allows us to convert the high voltage supply (here, the solar panel, 24V-50V) to the charging voltage for the batteries(12V or 24V).

Furthermore, the control modes of the converter allows operation in both constant voltage (CV), constant current (CC) output, or maximum power point tracking (MPPT) which is a special mode dedicated to extract the maximum amount of power from the solar panel.

### *A. Hardware architecture and design choices*

Typically, the model of a buck circuit is shown below:

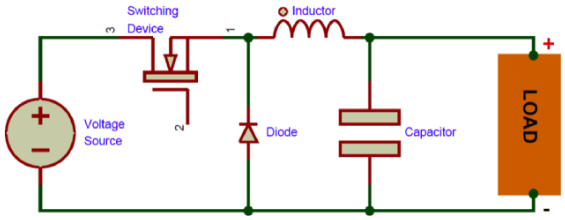


Figure 1: Typical architecture of a buck converter

To create this model and to adapt it to our system properly, we divided the buck into several parts.

*1) Charge Pump:* The first step is to chose the way to control the buck. We chose a MOSFET to realise this. This MOSFET can be consider as a switch which controls the buck input. The MOSFET output is a PWM signal. Thanks to the low cost and small size of MOSFET, we can easily integrate it into a PCB circuit. The MOSFET can not be driven directly because of the low voltage, so we used a driver, which provides the voltage difference between the pole Gate and the pole Source. In this part, it forms a charge pump, which means that the circuit can offer a higher voltage to drive the MOSFET and eliminates another independent power supply. When MOSFET is closed, the inductance and the capacitance will be charged. Once the MOSFET is open, the inductance and the capacitance will discharge and supply the load. The schematic is shown below:

*2) Asynchronous:* The buck we have designed is asynchronous. An asynchronous buck is cheaper than synchronous one. Even though its efficiency is lower (but in our system, the output voltage is relatively high), the diode voltage drop accounts for a small proportion of the overall voltage.

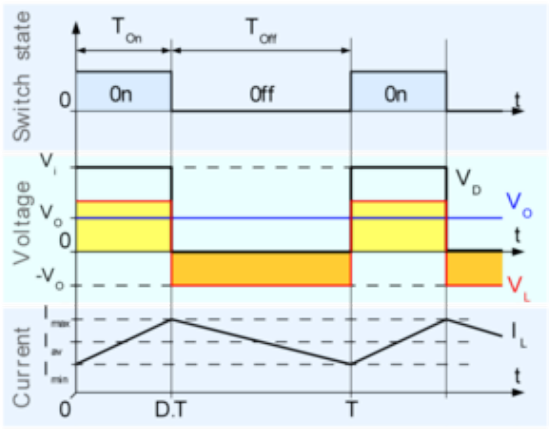


Figure 2: Evolution of the voltages and currents with time in an ideal buck converter operating in continuous mode

*3) Design trade-offs:* Once the overall architecture of the converter was selected, some key variables had to be defined in order to be able to properly size the power components. In particular, the switching frequency is an important parameter, as increasing it will also increase the dynamic losses, but reduce component sizes. Especially considering the inductor selection, both the switching frequency and the allowed current ripple can greatly impact the system footprint by several orders of magnitude.

We settled for a compromise solution, taking into account the components already available in stock, and selected a frequency of 150kHz and a current ripple of approximately 25mA. While the %ripple is low compared to usual values (around 20-30%), this allowed us to relax some constraints on the output filtering section, reducing output capacitance and battery ripple current overall. The choice of a secondary filter inductor has been made to further smooth out the output.

### B. Hardware schematic

The complete schematic of the converter is available in appendix. Here will be described some of the choices and considerations regarding component selection, placement and routing.

The current sensing has been implemented on the high side of the load, through a shunt resistor. The high common mode voltage required us to use a specialized current sense amplifier. The main advantage of such a technique is to provide a common ground across all the PCB, avoiding any ground loop issue we may encounter when integrating in the whole system.

The MOS and Schottky diode static and dynamic losses have been estimated, to validate the thermal side of the design. Low rds_on and Vf are essential to guarantee that devices will run cool, improving the system efficiency and lifetime, especially since electrolytic capacitors will be placed nearby. The required area of copper to maintain the junction of the diode under an acceptable temperature was calculated, and checked on the final PCB layout. This was another compromise between running the diode cooler (by increasing the area of the switching node) and increasing the amount of EM radiation, due to the large switching plane.

Filtering capacitors have been selected to have low ESR, in order to minimize self-heating (which shortens their lifespan) and to improve filtering quality. The RMS current requirement for both the input and output were quite stringent, as it is difficult to find low value capacitors with high ripple current rating.

### C. PCB routing

A proper routing of the converter is essential to obtain good performances, especially regarding EMC considerations. Care has been taken to follow good routing practices, ensuring sufficient track width to handle the current at stake, and minimizing parasitic elements by a good track spacing and the use of power places. Especially in the PCB area where high $dV/dt$ and $di/dt$ occur, the current loop should be minimized to limit its radiating capabilities. Sensitive low-level analog feedback signals have been routed to avoid coupling with noisy planes.

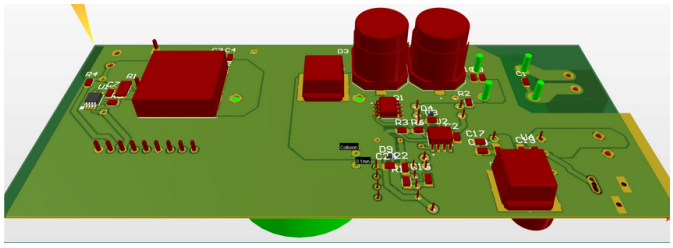

Figure 3: PCB 3D preview, bottom

For the low ohmic value shunt resistor, Kelvin connections ensured an accurate measurement of the output current. Extra footprints were laid out on the design to open the possibility for modding the board, and improve the EMI filters later on, at the prototyping stage. Coupling between the two outputs filtering inductors was also taken into consideration, as unwanted coupling due to poor placement can drastically reduce the filter effectiveness. Finally, debug functions, such as numerous test points, debug LED and switch, cutoff points, etc, were included to ease development.

### D. Software Analysis and Architecture

The software plays an major role in the system because it enables the final user to choose the desired configuration for the buck regulation. The goal is to develop an API (Application Programming Interface) to control the converter bloc of supply system.

*1) Functionalities:* The software functions associated to the specification functionalities are described below.

| Functional requirements | Degree of consideration | Software function concerned |
|---|---|---|
| F1a_1 | partial | GetMeasurement |
| F1a_2 | partial | GetMeasurement |
| F1a_3 | partial | GetMeasurement |
| $F1a\_4$ | partial | RegulateCV<br>RegulateCC<br>RegulateMPPT |
| F2_1 | total | GetMeasurement |
| F2_2 | total | RegulateCV<br>RegulateCC<br>RegulateMPPT |
| F4_2 | partial | getInputVoltage<br>getOutputVoltage<br>getOutputPower<br>getInputPower<br>getOutputCurrent |

*2) Software architecture:* Following the different functionalities, we have established a class diagram to be able to answer them, as seen below.

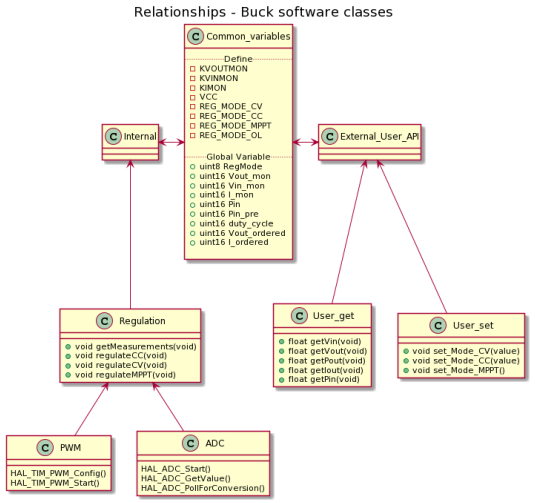


Figure 4: Class Diagram of the Buck software architecture

In this diagram, two main parts refer to the internal processes and the external API. On the left, the functions are used for regulation computation whereas on the right, the functions are dedicated to the user. By using this library, the user will be able to chose a specific regulation mode, or recover a regulation characteristic value, such as voltage, current or power.

## E. Regulation

*1) Dynamic system modeling:* For the model of our system, we use Matlab and Simulink to obtain our regulation coefficients for CV regulation and CC regulation. Is shown the Simulink diagram below.

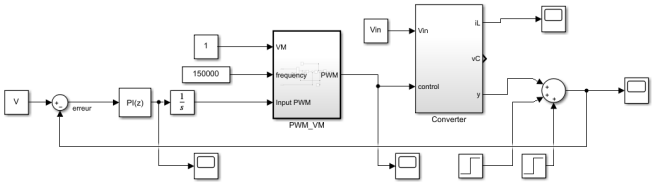


Figure 5: Regulation modelisation with a PI corrector

With this Simulink we have the Proportional and Integrator (PI) coefficients (Kp and Ki, used below).

*2) Regulation for CV/CC:* In this part, we describe the computation of the current and voltage regulation by using a recurrence equation. Starting from the Simulink PI equation previously identified, we calculated the following reccurence equation:

$$Kp * e_n - Kp * e_{n-1} + Ki * Te * e_{n-1} + s_{n-1} = s_n \quad (1)$$

We have:
- $e_n$ which is the current error between the value and the target.
- $e_{n-1}$ which is the previous error.
- $s_n$ which is the output of our system, here the duty cycle.
- $s_{n-1}$ which is the previous value of our output.

*3) Algorithm for MPPT:* The MPPT needs a different regulation because contrary to the previous regulations where we set a target value, we are constantly in search of the maximum value point. A solar panel works with a voltage/current characteristic curve, a parabolic shape. To reach the parabola extremum, we vary the output current while monitoring the input power to obtain the maximum power. This is how we built our algorithm for MPPT regulation, taking into account noise consideration in our measurements.

## F. Test results, performance

Once the PCB manufactured, and the regulation parameters computed, the last part of our work was testing the effective behaviour of the buck, and validate its performance.

*1) Test plan:* A test protocol has been elaborated in order to check the robustness and the reliability of the buck, regarding the PCB and the software program. Here are the different elements of the test plan :
-Progressive assembly and testing the PCB parameters.
-Run some open loop tests such as voltages and bootstrap supply operation and microcontroller feedbacks and readings.
-Run some tests in closed loop such as navigate through user interface and evaluate the different regulation modes.
-Move from using lab power supplies as sources to the

actual solar panel.
-Examine the electromagnetic compatibility aspects.

*2) Test results:*

*a) Open-loop tests:* During the first open-loop tests, several minor mistakes were discovered in the PCB, but those were easily fixable and did not impact the correct operation of the converter. After verifying the appearance and values of the switching waveforms, and the operation of the bootstrap supply at low power, low Vin, we decided to step up the power. Tests were conducted on dummy resistive loads, up to 230W output, 28V out, 8.3A, 45V in. The temperature of the critical components (MOS, diode, inductor, ...) was carefully monitored, and no component exceeded 40°C at steady state, which is better thermal performance than expected. The overall efficiency of the system was measured, by paying attention to ensure accurate measurements. By example, the voltages should be sampled as close as possible to the DUT, and input voltage should be adjusted when increasing the load current to compensate the voltage drop in the cables.

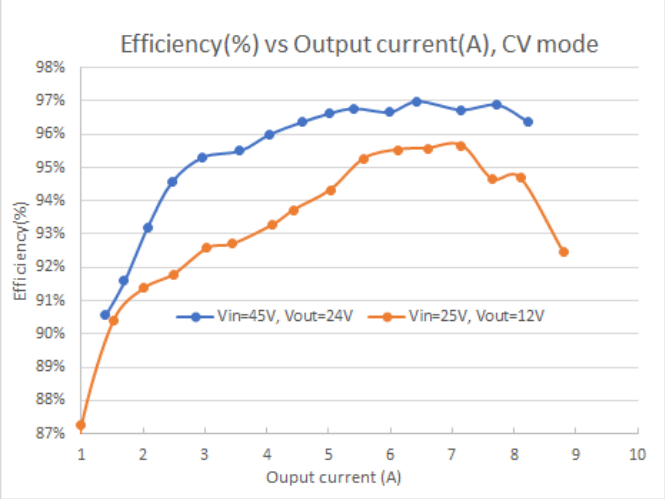


Figure 6: Buck efficiency vs output current, 2 Vin values

The consumption of the auxiliary 12V supply and the microcontroller (powered from the auxiliary supply) has been taken into account as well for this figure. Although it has a negligible effect at high loads, its fixed losses greatly degrade the system efficiency for output currents less than 1.5A.

*b) Closed loop tests:* During closed loop testing, the three mode of operation (CV, CC, MPPT) were individually tested, using a simple uart debugging interface. This interface allowed us to set the operating mode, change the set point value (either target voltage or current) and provide feedback on the controller measured values and output command. The stability of the closed loop was tested, as well as the accuracy of the set point across all of the output range, for various loads configurations. The high frequency ripple was measured to be less than 160mV, or 22mA in current, and proved to be mostly constant for different input voltage, output voltage and current configurations. Load tests on the solar panel were performed, as long as solar power was available. The MPPT regulation proved to be more difficult to achieve, as our power measurement were too noisy for our simple algorithm to converge. However, it should be possible implement this mode in software given enough time to debug it, as the sensing hardware is functional.

*c) EMC testing:* Once the functionality of the converter was verified, we had the opportunity to take some pre-compliance electromagnetic tests. We did not examine susceptibility compliance aspects, nor radiated emissions, but conducted emissions. The device was set up in an anechoic chamber, put under load (Vin=45V, Vout=24V, 4A) and powered through approx 1.5m cable laid 5cm above the ground plane. Common mode current measurements were taken on the input, as well as voltage noise measurements, on both the positive and negative wires. Those results are shown here :

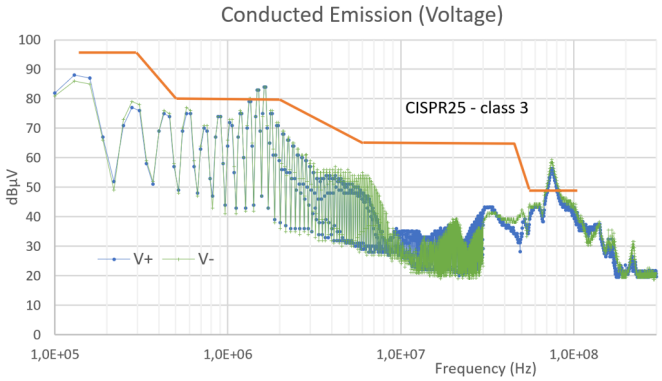


Figure 7: Conducted emission test from 100kHz to 300MHz

A comparison is made with a CISPR norm, which is an automotive standard, shows that the low frequency portion of the spectrum is mostly well filtered, apart from a small 5dB peak at 1.5MHz, which could be attenuated by tweaking the filter values. On the high end of the spectrum, another peak is present at 72MHz. This could be due to cable resonances. Overall, the performance of the device is quite good considering that it is only a first prototype, and nothing was specifically done to improve the filter response.

## 4. Battery Management System

This section presents all the design aspects of the Battery Management System (BMS). The different parts of the hardware design are first addressed: the high level architecture, the technological choices, the routing and finally the validation tests. Then, the software aspects of the project are presented: the analysis of the functionalities, the class diagram structuring and finally the implementation of the CPLD is detailed.
The main objective here is to design a system to easily and safely change the configuration of the batteries and to produce a API to control the system according to the state of charge of the batteries.

### A. Hardware architecture and technology choices

First, the system has to be able to switch between 4 distinct modes: series (when the system acts like a power supply to the load), charge battery 1, charge battery 2 and parallel (charge both battery 1 and battery 2). To do so, we will need only three switches (K2, K3 and K4 in fig 8.
Second, the system has to be able to disconnect the load, the buck or both of them. Therefore, two additional switches upstream and downstream of the system (K1 and K5 in fig 8) need to be added. The topology with the 5 switches can be found on Figure 9 with highlights for each configuration. Table I describes each switch behaviour according to the mode.

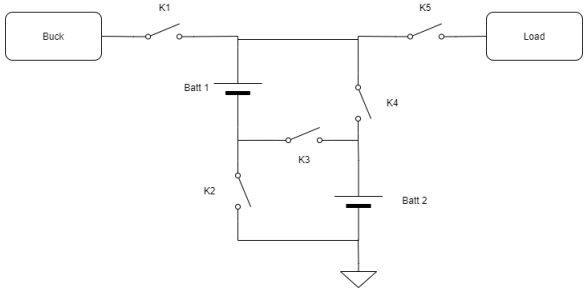


Figure 8: General topology of the switches

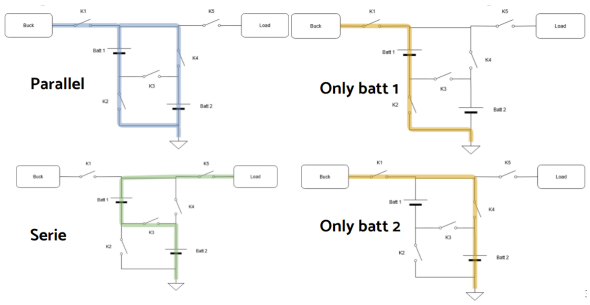


Figure 9: The 4 switch configurations

| Configuration | Switches open | Switches closed |
|---|---|---|
| **Parallel** | K3 K5 | K1 K2 K4 |
| **Series** | K2 K4 | K1 K3 K5 |
| **Batt1** | K3 K4 K5 | K1 K2 |
| **Batt2** | K2 K3 K5 | K1 K4 |

Table I: Status of the switches according to the mode

The system monitors the batteries thanks to two gauges LTC2944, throught I2C communication. Each gauge needs to be placed around a shunt resistance on a Kelvin connection. Finally, we place a CPLD between the microcontroller and the switches command to have a better level of security here. Actually, this technological choice will be more detailed in a future part of this report. The system described here is represented on fig 10.

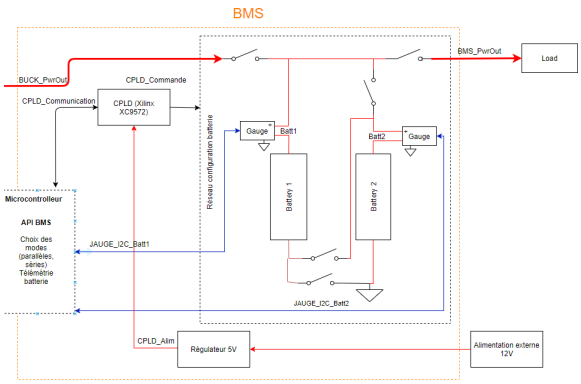


Figure 10: High Level Schematic

*1) Technological choices:* The next step is to choose how the different switches will be controlled. After a brief comparison between MOSFET and relay characteristics, we chose to use MOSFET, since this technology is cheaper and requires less current. We also studied the solution of relay MOSFETS: to have a sufficiently small $R_{DSon}$, the unit price increased too much for our specifications. The MOSFET solution seemed the best solution.

*2) Topology choices:* In Figure 12, we can visualize each switch and its topology[1].

Switches K1, K4 and K5 work in the same way: the two head-to-tail PMOS ensure that the switch is fully open when the gate voltage is at 0V and avoid current feedback. Moreover, if we send a logical '1', a voltage will be bring to the gate thanks to the bulk diode, closing the MOSFET, as shown in fig 11. Finally, the Zener allows us to stabilize the control.

[1] You can find the schematic in better quality on Annex

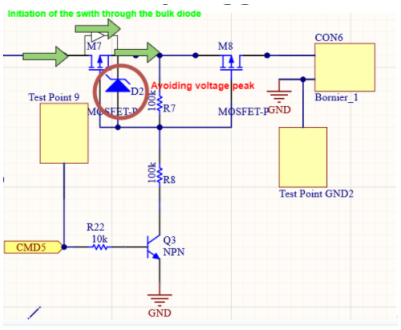


Figure 11: K1 K4 and K5 principle

Switch K2 has the advantage of being grounded on one side. Thus NMOS can be used to open the switch. This topology and its control is based on the fact that at the source of M3 there is always 12V or 24V regardless of the configuration. Finally, the two bipolar cascaded poles allow us to control the switch more easily.
Finally, the K3 switch works with the same technology as K1, K4 and K5 except that it relies on battery voltage levels to operate, like K2.

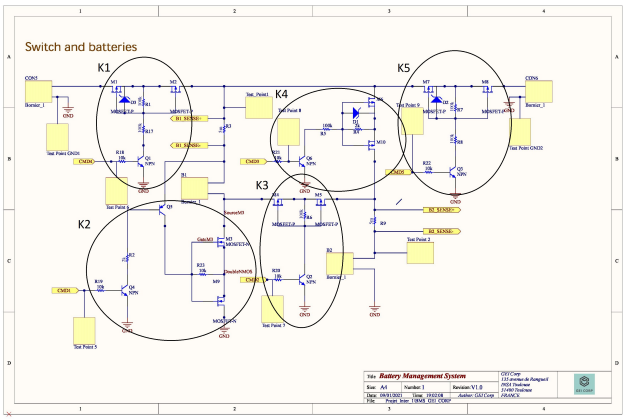


Figure 12: Switch schematic

*3) Routing of the PCB (Printed Circuit Board):* We have seen through the preceding section our choices concerning the schematic. In this section, we will focus on how we achieved the routing of the PCB (Printed Circuit Board) by considering two main issues: the high amperage of some of the signals and the EMC (electromagnetic compatibility). To do so, we organised the PCB in three sections as seen in Figure 13: the power section where a high current will flow; the control section composed by the bipolar transistors controlling the switches and finally the microcontroller interface section with the CPLD and the different connections with the STM32.
For the power section, if we have small tracks, the risk is an unwanted temperature elevation. Thus, if we want a maximum temperature elevation of 10°C, we can compute

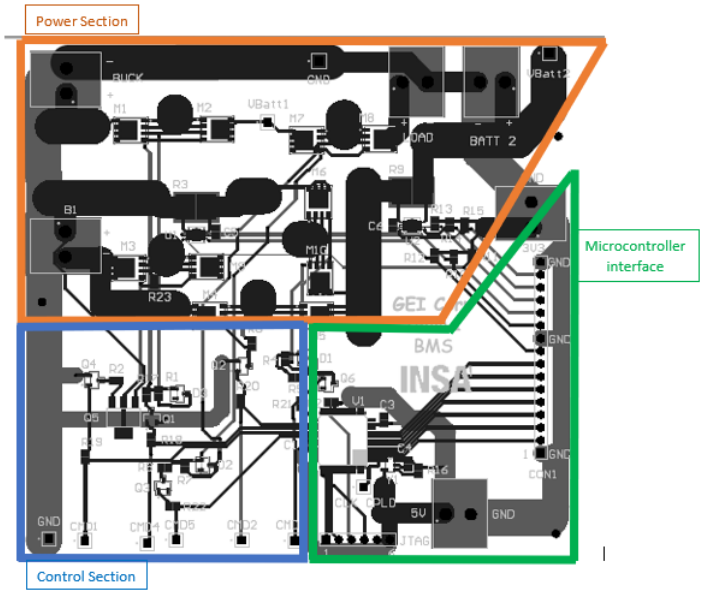


Figure 13: Organisation of the routing of the PCB

the minimum width of the tracks by using the following formula :

$$w_{min} =^{0.725} \sqrt{\frac{I_{max}}{\Delta T^{0.44} t^{0.725} k}}$$

where t=35 $\mu$ m, k=9,87, $\Delta$ T = 10 and $I_{max} = 10$.
Thanks to this, we know our minimum width is 7mm for the tracks located in the power section.

Finally, concerning the EMC specifications, since we do not have a lot of commutations, our system will not generate a lot of electromagnetic interferences. However, the SCL signal (used by the $I^2C$ bus) can generate some perturbations since it is a clock with a medium frequency. That is why we decided to add a ground connection just below the SCL signal to isolate and protect the other inputs and outputs of the system.

*4) PCB testing and validation:* The PCB we have manufactured requires validation tests for each sub-function of the board. Indeed, a faulty switch would make the board unusable and above all dangerous. Our test plan was organized as follows:

1) Switch K1;
   - Connect a 24V power supply to CON5;
   - Connect a 3V3 power supply to test point 6;
   - On a breadboard, add a 1k resistance connected to test point 1 and GND;
   - Measure with a multimeter if the switch is open or closed following the command (0V or 3V3 on TP6);
   - Run the power tests with the appropriate equipment (radiator).
2) Switch K4;

- Validate K1 and close K1 (3V3 on TP6);
- Connect a 24V power supply to CON5;
- Connect a 3V3 power supply to test point 9;
- On a breadboard, add a 1k resistance connected to CON6 and GND;
- Measure with a multimeter if the switch is open or closed following the command (0V or 3V3 on TP9);
- Run the power tests with the appropriate equipment (radiator).

3) Switch K2;
   - Validate K1 and close K1 (3V3 on TP6);
   - Connect discharged battery 1
   - Connect a 15V power supply to CON5;
   - If we put 0V on test point 5, we should measure 3V between M3 and B1;
   - If we put 3V3 on test point 5, we should measure around 0V between M3 and B1, and the power supply must deliver current (charge the battery);
   - Run the power tests with the appropriate equipment (radiator)
4) Switch K4;
   - Validate K1 and close K1 (3V3 on TP6);
   - Connect discharged battery 2;
   - Connect a 15V power supply to CON5;
   - If we put 0V on test point 8, we should measure 3V between M10 and B2;
   - If we put 3V3 on test point 8, we should measure around 0V between M10 and B2, and the power supply must deliver current (charge the battery);
   - Run the power tests with the appropriate equipment (radiator)
5) Switch K3;
   - Close K1 and K3;
   - Open K2 and K4;
   - Connect both batteries;
   - Put 3V3 on test point 7;
   - Connect a 15V power supply to CON5;
   - The power supply must deliver current (charge the batteries) ;
   - Open K1, K2 and K4;
   - Close K3 and K5;
   - Put a load on CON6;
   - We should measure a voltage drop before the load.
   - Put 0V on test point 7 and repeat the experiment, the result should be reversed.
6) Gauges B1 and B2
   - Branch both batteries and monitor the charge with the gauges (special software for testing)
7) CPLD
   - Check the oscillator on test point 4;
   - Do not connect any alimentation or batteries;
   - Run special software for testing and check the CPLD's output. If it matches the command, then it is validated and resistance should be connected. Otherwise, overcome the CPLD with wires.

### B. Software Analysis and Architecture

In the same time, our system has to be managed by software to collect information or to change its configurations. The team was only in charge of the software layer and not the client layer.

*1) Functionalities:* In order to built a functional and secure device, the user needs to have access to the voltage, current or state of charge of the battery to take decisions. When knowing the states of the battery, the user has to be able to set the different modes (parallel, serie, etc) or to disconnect everything when an issue appears.

In the table II, we defined functions to answer specifications[2] given by our client and specified if the need was entirely considered or not :

| **Functional requirements** | **Degree of consideration** | **Software function concerned** |
|---|---|---|
| F1a_1 | partial<br>partial | setMode<br>DisconnectSource |
| F1a_2 | partial<br>partial | setMode<br>DisconnectSource |
| $F1a\_3$ | partial | resetMode<br>getVoltage<br>getCurrent<br>getSoc |
| $F1a\_4$ | partial | setModeSeries<br>setModeParallel<br>setModeLoadBatt1<br>setModeLoadBatt2<br>InfoBatt - getSoc |
| F2_1 | total<br>total | setMode<br>DisconnectLoad |
| F2_2 | total | setModeLoadBatt1<br>setModeLoadBatt2<br>setModeParallel |
| F2_3 | total | getVoltage<br>getCurrent<br>getSoc |
| F4_1 | partial | getVoltage<br>getCurrent<br>getSoc |

Table II: Functional requirements and software functions associated

[2]Please find the written functionalities in annex

*2) Software architecture:* With the functions above defined, we designed and built the architecture of the project.

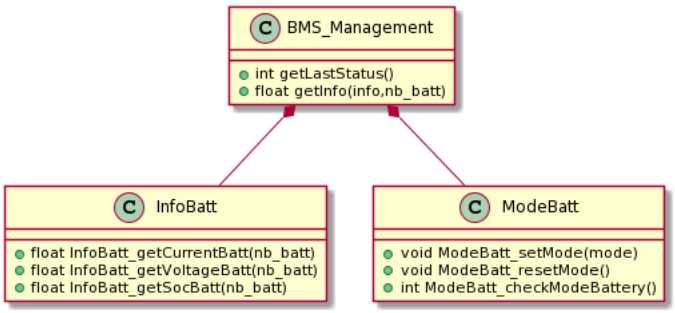


Figure 14: Class Diagram of the BMS software architecture

The user can now access information with the "get..." functions and change the system configuration with the "set..." functions.

### C. CPLD architecture and communication

Regarding batteries configurations, we can identify some dangerous ones we want to avoid, for example to short circuit a battery. Since we want a safe system, a Complex Programmable Logic Device (CPLD) will be placed downstream of the switches in order to avoid these dangerous configurations. The CPLD acts as a protection: no matter what commands it receives from the software, it will ensure the system integrity.

*1) Technology choices:* The CPLD used is from **Xilinx** its reference is: `XC9572XL` .
The datasheet is available here: https://www.mouser.fr/datasheet/2/903/ds057-1595690.pdf

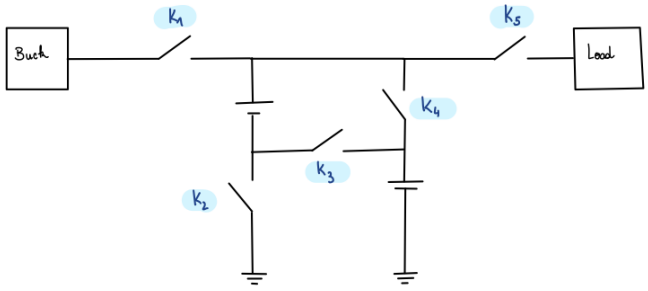


Figure 15: Switches numbering

*2) Inputs and Outputs:* 4 inputs are used in the CPLD. The microcontroller sends logic level signals corresponding to a '1' or a '0'. Their names and roles are described in the list below.

- `LOAD`: Enables the user to load a new configuration (batteries in parallel, in series, only battery 1 or 2).
- `MODE_0`: Allows the user to choose a configuration.(See the table below )

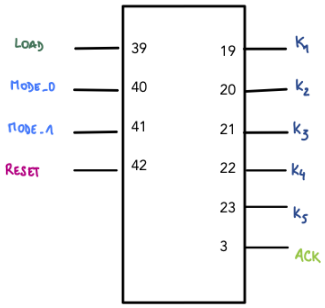


Figure 16: CPLD inputs/outputs

| | MODE_0 | MODE_1 |
|---|---|---|
| **PARALLEL** | 0 | 0 |
| **SERIES** | 1 | 1 |
| **BATT1** | 0 | 1 |
| **BATT2** | 1 | 0 |

Table III: **Configuration mode encoding**

- `MODE_1`: Allows the user to choose a configuration.(See the table below )
- `RESET`: Enables the reset configuration where all switches are open.

6 logic level outputs are generated by the CPLD:

- `K1, K2, K3, K4, K5`: correspond to each switch command (OPEN='0' or CLOSE='1').
- `ACK`: raised when the configuration change has succeeded.

*3) State Machine Diagram:* In order to represent the interaction between each configuration we have conceived a state machine diagram. One configuration correspond to one state.

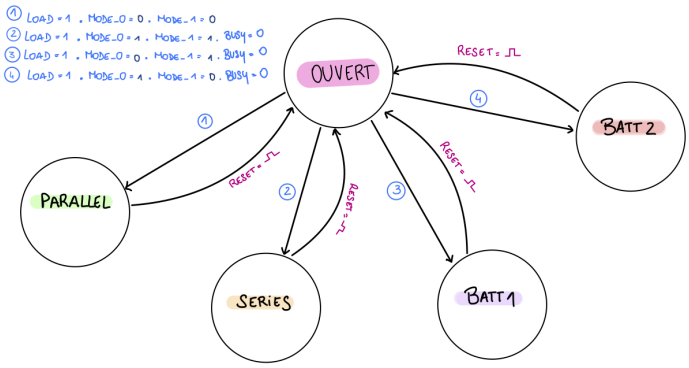


Figure 17: State machine diagram of switches control

A significant thing to note is that PARALLEL, SERIES, BATT1 and BATT2 states can only be reached if the previous state is OUVERT.

The oscillator which runs the clock is set to 33kHz. Each state is composed by a sequence of actions, so switches will

be commanded in a certain order to avoid any dangerous states. Between each action we have chosen to set a 1ms delay, this delay has to be higher than MOSFET transistor switching times. In our case, the longest switching time is due to the PMOS (SI7155DP-T1-GE3) with a maximal rise time equals to 370 ns.
Datasheet available here: http://www.farnell.com/datasheets/2606255.pdf

Following, sequence of switches commands to reach each configuration:

- OUVERT
  1) K1=0
  2) K5=0
  3) K3=0
  4) K2=0 and K4=0
- PARALLEL
  1) K2=1 and K4=1
  2) K1=1
- SERIES
  1) K3=1
  2) K1=1
  3) K5=1
- BATT1
  1) K2=1
  2) K1=1
- BATT2
  1) K4=1
  2) K1=1

*4) VHDL Code:* To code and to implement our program on the CPLD we used:

- **VHDL** language
- **Xilinx ISE**. This software has been discontinued since 2013, therefore to use it with Windows 10 a virtual machine is required.

BMS.vhd: Contains the code of the state machine

To create delays the solution in VHDL is to code a prescaler which divides the main clock. In our case we have two distinct synchronous processes so we needed two new clocks (around 1ms and 0,5ms).

A crucial point is the initial state when the CPLD is turned on. We had to make sure that it is not randomly initialized.
The solution implemented is available on this site:
https://www.xilinx.com/support/documentation/sw_manuals/help/iseguide/mergedProjects/destech/html/cd_fsm.html.

Before any change between configurations, the function ModeBatt_resetMode will be called by the µC in order to open all switches (to reach the OUVERT state). Once this step is done and if the CPDL has sent the acknowledgment (ACK output), then the µC will call ModeBatt_setMode to enter the new configuration.

*5) Simulation and material tests:* Xilinx ISE offers the possibility to run simulation.
testcpt.vhd: Contains the code for simulations
BMS.ucf: Contains pinlock information. **In this file the pins correspond to the card test** (schematic sch_CPLD.pdfavailable on Dropbox https://www.dropbox.com/home/Projet_5ESPE_2020_2021/Team_1/BMS/SoftwareDesign/CPLD

To carry out tests and to implement our program a test card was used. This card contains an another CPLD (XC9572XL-10VQG44C), the clock is set to 50MHz. This led to a minor change in our program (BMS.vhd), the value for our first prescaler (Q_div) has to be modified in order to set the new clock at 2000KHz.
The following simulations are implemented:

1) RESET test lead to OUVERT state (all switches are open)
2) PARALLEL-OUVERT
3) PARALLEL-OUVERT-SERIES
4) PARALLEL-OUVERT-BATT1
5) PARALLEL-OUVERT-BATT2
6) OUVERT is primary, nothing can happened until all switches are open.

All these simulations were successful and two examples can be found on Figure 6 and 7 in the annex part.

Then some material tests were carried out with a test card. Outputs were observed with the Analog Discovery 2 logic analyzer. Some results are visible on Figure 8 and 9 in annex. The system worked as expected. The next step will be to test it on a PCB with the switches.

### *D. Battery gauges communication*

*1) Presentation of the gauges:* Gauges use a LTC2944 sensor to measure three parameters: battery terminal voltage, current through the battery, state of charge of the battery. It will be placed above each battery (on the positive terminal).

*2) Communication between the gauges and the µC:* We use the $I^2C$ protocol to communicate with the LTC2944. At initialization, we make sure the communication between the sensor and our microcontroller is enabled. Then, the sensor is configured to perform an automatic and continuous measure so we can extract the information at any time. As the slave address is hard-fixed, we cannot change it, and as

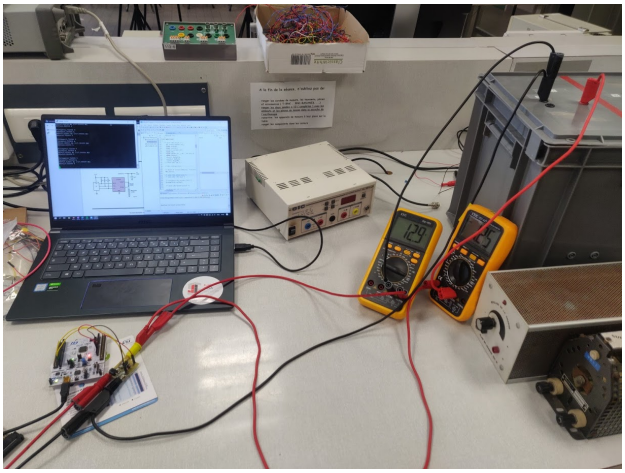

Figure 18: Test *in-situ* of the gauge connected to the battery

we use two LTC2944, we need two distinct $I^2C$ channels (thereafter hi2c1 and hi2c2).

*3) Data processing:* As we need to extract the 2-byte-values ("MSB" byte and "LSB" byte) from the sensor, we check on the datasheet where to find them, *i.e.* what are their addresses. We have the following ones :

- Voltage register MSB: `0x08`,
- Current register MSB: `0x0E`,
- State of Charge register MSB: `0x02`.

We only need the MSB address to get all the information since the two bytes follow each other. Thus, we will only use in our code the addresses gave hereinabove. After getting the data through the $I^2C$ channel, we use the calculation on the datasheet to get the real value:

- For the voltage, we use the formula: $V = 70.8V * \frac{RESULT_{dec}}{65535}$
- For the current: $I = \frac{64mV}{R_{sense}} * \frac{RESULT_{dec}-32767}{32767}$ - where $R_{sense} = 6.08m\Omega$ according to a measure of it *in-situ* during the battery test.
- For the SOC (after initialization): $SOC = RESULT_{dec} * \delta Q$ - where $\delta Q[Ah] = 0.34 * \frac{50m\Omega}{R_{sense}} * \frac{M}{4096}$ and where M is the prescaler set at 1024.

*4) Test:* For this test (*c.f.* Fig. 18), we will be using:

- one 12V-battery and one load,
- the LTC2944 and our µC (with the code `I2Ctest.c`),
- one voltmeter and one ammeter.

The test will be divided in two parts: the first one will consist in discharging the battery through a load, the second one in charging the battery thanks to a power supply. With our multimeters, we validate the results obtained on with the gauge for the voltage and the current. The feedback we have from our microcontroller is given hereinafter (*c.f* Fig.19). After some issues, especially with the SOC value, we found out that we inverted the MSB and the LSB byte. If it had a low result on the voltage and current read, it gave the impression of a big dynamic on the SOC read.

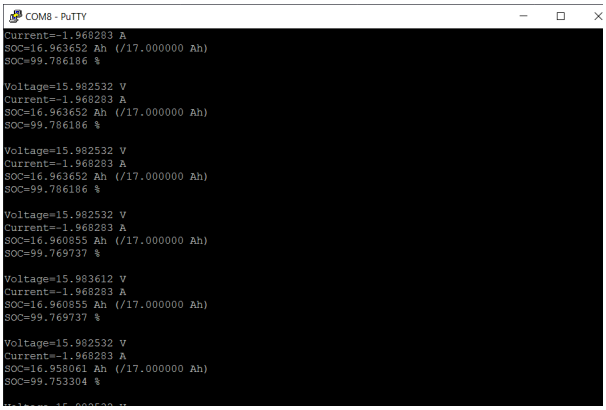


Figure 19: Feedback of LTC2944 from the micrcontroller

## 5. System integration

The integration is the final step of the project, where all the subsystems are put together to form the Smart Power Management System. This involves cascading the several elements in this order:
- The solar panel is connected to the buck to first transmit the raw power.
- The buck converts the energy and then is connected on the BMS which handles connection between the buck, gauges and the batteries.
- The load is the final element of the chain, connected to the BMS which regulates power transfer to the charge as explained is the previous section.

The buck and the BMS have been produced on two separate boards. As a practical matter, we managed the GPIO sharing to use the same microcontroller, connected by wires to the buck and BMS. Regarding the software integration, both API will be combined so that the user can use needed functions with ease.

Furthermore, we defined some use cases to have a better understanding of the whole systems behaviour. The following situations describe the buck and BMS interactions in a normal operating mode, and also when some errors come up.

1) **Batteries 1 and 2 in series** : **buck in CV mode or MPPT**. Batteries use the energy transmitted by the buck to charge. If there is too much then the energy is directly sent to the load.
2) **Battery 1, battery discharged (charge battery 1 mode )**: To begin the battery charge when it is empty (bulk and absorption phase) the **buck is in CC mode**. To end the charge (floating phase) the **buck has to switch into the CV mode** with a voltage control at 12V.
3) **Battery 2, battery discharged (charge battery 2 mode)**: Same as above.
4) **Batteries 1 and 2 in parallel (floating end of charge)**: The buck is in **CV mode** in order to complete the charge.
5) **Batteries 1 and 2 disconnected**: The power coming from the buck is directly transmitted to the load.

In addition to those batteries configurations, the two switches upstream and downstream allow the BMS to be disconnected from the buck (in case of an overvoltage) or from the load.

## Conclusion

Throughout this project we were divided into subgroups in order to work more efficiently on different tasks. The regular working session on Tuesdays also played an important role in our productivity and our efficiency. With dedicated explicit tasks, everyone could fully use their potential and make the project improve little by little. This project is a first step toward industrialisation process, from sizing and design to test on the actual board. Then working on all these steps gave us a better understanding of project development as a whole. Moreover, we were able to fully apply theoretical concepts taught in several classes.
Going through this project all along this semester is a great experience even if due to covid-19 situation, working conditions and physical time spent in the GEI were not optimum. But as engineers, we managed to adapt ourselves and finally deliver usable bricks for our successors, in line with the specifications.