POLITECNICO DI TORINO

Master Degree Course in Electronic Engineering

Master Degree Thesis

BAT-MAN

Current and Voltage Sensing Circuit for Automotive Batteries SoH and SoC Determination

Advisor
Prof. Eros Pasero
Co - Advisors
Eng. Jacopo Ferretti
Eng. Vincenzo Randazzo
Internship Tutor
Eng. Giovanni Guida

Candidates
Federico Di Fazio
Davide Fazi

December 2019

Summary

Nowadays, batteries are always more important in our lives especially in automotive field where they are the key point of innovation. In this scenario it is necessary to check how good the battery is.

The aim of this project is to evaluate the State of Charge (SoC) and the State of Health (SoH) of a lead acid 12V battery: the first one indicates the remaining charge of a battery while the second is a measure of its ageing. The two parameters are strictly related: an old battery has a low SoH, this implies a derating of its capacity and a consequent different behaviour of SoC.
In order to compute these figures of merit most commercial devices measure the voltage of the battery during the different operative stages; this technique requires the knowledge of the discharge curve which is different for every manufacturer and its behaviour is also modified by many external parameters such as temperature and battery ageing.

In the developed device both battery voltage and sink/charge current are measured to obtain a better estimation of the desired parameters. This introduces huge problems related to the high currents involved; the adopted solution is the resistive current sensing since it is robust to EM disturbances, low noisy, and ensures high temperature ratings and stability. This solution was compared with an hall effect sensor that is intrinsically insulated and allows to avoid series insertion. On the other hand, high precision sensors are expensive and require many foresights such as modulation.

The current to be measured is as high as 1000 A, therefore a shunt resistor with a value of $50\mu\Omega$ was chosen to reduce both power dissipation and voltage drop. Here the need of an accurate sensing circuit, capable of appreciating voltages as low as $1\mu V$, arises. Moreover, since the device must be able to appreciate tens of mA, the involved ADC must have a high resolution. These two requirements are accomplished by a commercial integrated circuit - the INA226Q1 - which is equipped with a $\Delta\Sigma$ ADC with a 16 bits resolution. The same IC is also capable to acquire the battery voltage with the same resolution used for the current.

PCB and hood temperature are acquired too: the first one allows to compensate the thermal diffusion of the components - especially of the input protection diode - while the second one is taken into account by the SoH and SoC algorithm. This algorithm - patented by *brainTechnologies* - is based on a Kalman Filter but will not be object of this thesis work. The novelty of the involved approach consists of a multi-model description of the physical behaviour of the battery. This approach allows to manage multi-modal disturbances providing a high level of robustness to the inference and reduces computational cost in order to implement it on a microcontroller.

In fact, the core of BAT-MAN is a 32 bit microcontroller - programmed using a real time operating system - based on an ARM® Cortex® M3 and equipped with an internal Bluetooth® module used to transmit acquired data and predictions to a smartphone where they are stored and plotted by and Android™ App. The use of a RTOS allows to take advantage of multitasking computation, which plays a primary importance role in an application where different tasks have to be synchronised.

Acquired data, before being sent to the smartphone app, can be stored in an external flash memory. This feature could allow to monitor the battery also when the driver is not inside the car, for example understanding if - when the car is parked - there is some equipment that is sinking current.

The project consists of all the steps regarding the designing of a PCB; choice of the components, design of the schematic, layout routing and real implementation with both reflow oven and manual soldering. Firmware and Java Android app were part of the performed work too.

An interesting activity carried out during the development of this project was the simulation of the antenna matching circuit provided by the manufacturer of the microntroller (Texas Instruments). This study highlighted that the suggested components values are not suitable for the routed layout; in fact, at 2.4GHz the traces act as microstrips. By using *Optimization Cockpit* by Keysight ADS, it was possible to select components values such that antenna and μC RF pins are properly matched.

Several tests were carried out on BAT-MAN to debug and guarantee its correct working. A verification & validation process was performed to ensure the meeting of the requirements and specifications intended to be reached. This procedure was accomplished by mean of many different laboratory test. Unfortunately, due to the lack of specific equipment, many of the measurements provided just qualitative

results. In order to achieve the best reliability in all conditions, sensors must be calibrated using certified instrumentation.

The Battery Management system has also been field-tested by connecting it to the battery installed in the hood of a car. This was the only way to properly validate its functioning. The car used for the tests - during the cranking phase - required hundreds of amperes to the battery, ensuring not only the capability to survive to these high currents, but also to accurately measure them. By doing this it was also possible to verify the correctness of the SoC and SoH algorithm developed by the partner company.

The main BAT-MAN's future perspective consists in Cloud Computing which allows to put into communication many devices - therefore many users - in order to achieve better SoC and SoH estimations. In fact the battery model could become every day more accurate by creating a worldwide database where data from batteries, cars and alternators are stored.

The new era of automotive is certainly represented by Hybrid and Fully-Electric vehicles, therefore BAT-MAN should be able to face this huge challenge. In order to exploit the developed technology for this purpose, the design choices should be adapted to meet these new requirements.
Electric vehicles are supplied by LiPo batteries composed of a big cell's array reaching voltages as high as 400 V; for this sake a conditioning circuit must be adopted.

The whole project has been sponsored by *MESAP* and realized in cooperation with *brainTechnologies* - as project leader - and *Sive*.

Acknowledgements

Ringraziamo il Prof Eros Pasero per la disponibilità dimostrata nei nostri confronti e per averci offerto di poter lavorare ad un progetto così interessante e motivante. Questi mesi ci hanno permesso di acquisire nuove competenze mettendoci alla prova con una realtà innovativa e non esclusivamente didattica.
Grazie a Jacopo e Vincenzo per averci supportato durante questo percorso rendenendosi sempre disponibili nel momento del bisogno. Il loro aiuto è stato prezioso ai fini del raggiungimento del nostro obiettivo.
Vorremmo ringraziare *brainTechnologies*, in particolar modo il referente del progetto BAT-MAN Giovanni Guida, per averci dato l'opportunità di metterci in gioco su tematiche stimolanti e all'avanguardia. Giovanni si è sempre mostrato disponibile a chiarire ogni aspetto inerente l'attività coinvolgendoci nelle fondamentali prove su vettura.
Un saluto e un grande in bocca al lupo al nostro amico e compagno di laboratorio Domenico per il proseguimento del suo percorso.

Contents

List of Tables

List of Figures

Part I

Introduction

Chapter 1

Introduction to the problem

1.1 Battery Technology

A lead-acid battery consists of mold lead plates inserted in liquid concentrated sulfuric acid and enclosed in a robust polymer casing equipped with a vent. The thickness of the plates varies according to the purpose of the battery; deep-cycle or starting purpose.
These devices are composed of many cells, whose voltage must be between 1.5V and 2.7V. This is the oldest kind of battery but is still used and developed thanks to its very low cost, recyclability, and good longevity in full charged state.
On the other hand, when these batteries are fully discharged they must be immediately recharged to avoid sulfation; a creation of large crystals which can be partially repaired only charging at elevate voltages for several days.
Another problem is plate corrosion, avoided keeping the battery charged without exceeding 2.4V. All these phenomena cause a degradation of the performances reducing the SoH and then the maximum charge.
The futuristic developments to avoid these problems, is to use graphite foam electrodes thus improving contact between electrodes and active materials due to high surface area. In such a way electrode corrosion and sulfatation are drastically reduced while energy and power densities are increased. [1]

1.2 Battery Model

A battery is a complex electrochemical system, therefore it is not simple to obtain a precise model and there is not an unique way to model it.
For this reason, many different approaches can be used to model battery behaviour, each one able to characterise a different detail level of reality directly reflecting on the computational effort. Actually, each model has different characteristics and the choice must be done according to a trade-off between accuracy and complexity. [2]

The main possibilities are:

- **Analytical modelling of electrochemical phenomena**
  By solving complex non-linear systems it is possible to describe the macroscopic behaviour of the chemical processes involved in a battery. This method requires a big experimental effort to evaluate the parameters and is also characterised by an heavy computational cost. An example for this approach is the *Kinetic Battery Model*

- **Data-Driven modelling (Black-Box)**
  This technique doesn't require a deep knowledge of the physical processes since it is based on an heuristic approach to forecast behaviours and battery conditions. Thence, an artificial intelligence such as a neural network is needed, obtaining at the end a non-linear function able to describe the battery variables of interest.

- **Physical modelling (Electrical equivalent circuit)**
  It is possible to represent a battery as an electrical equivalent circuit. However, also to do this, a deep knowledge of the internal chemical processes is needed in order to explain the non-linearity observed during tests. It is possible to choose different electrical models depending on the degree of accuracy and consequently on the complexity of the model.

- **Hybrid modelling**
  Is a combination of the electrochemical and the RC electric model. The first one allows to estimate the SoC in a robust manner including non-linear variations and self-recovery, while the second reproduces a wide spectrum of dynamics.

1.2.1 Electrical Equivalent Circuit Model

The most interesting family of models to be considered is the one based on electrical equivalent circuit since this is used by *brainTechnologies S.r.l.* to estimate SoC and SoH.
The main electrical models are:

- **Ohmic Model**

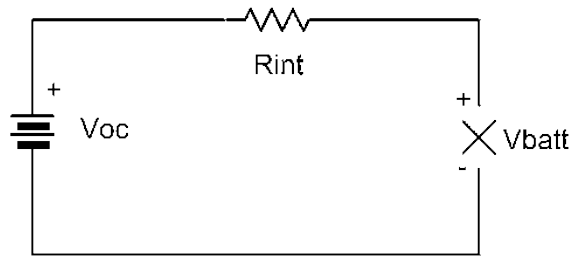


Figure 1.1. Battery Ohmic Model

Is the easiest model since it consist in just one resistor to model the internal impedance of the battery. To have a dynamic model it is possible to consider the resistance value dependent on the SoC.

- **Thevenin Model**

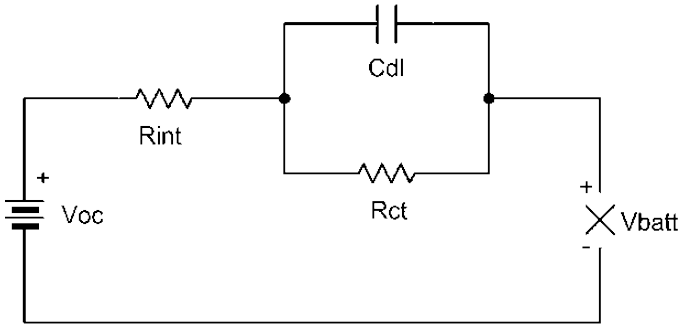


Figure 1.2. Battery Thevenin Model

In the previous model an R-C component has been added to take into consideration also the true capacitance of the battery. The delay due to the RC constant allows to consider the transient response, really useful in car systems where high currents are absorbed in a relatively short time.

- **Non-Linear Dynamic Model**

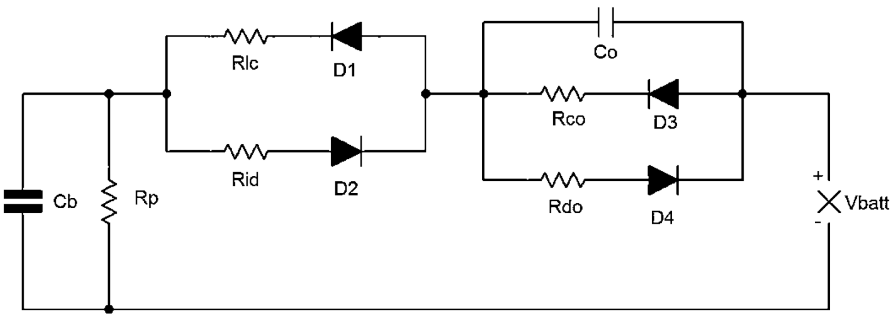


Figure 1.3. Battery Non-Linear Model

This circuit intorduces non-linear components (diodes) to model the non-linearities of the battery behaviour, making it even more realistic.

- **Finite Warburg Impedance Model**

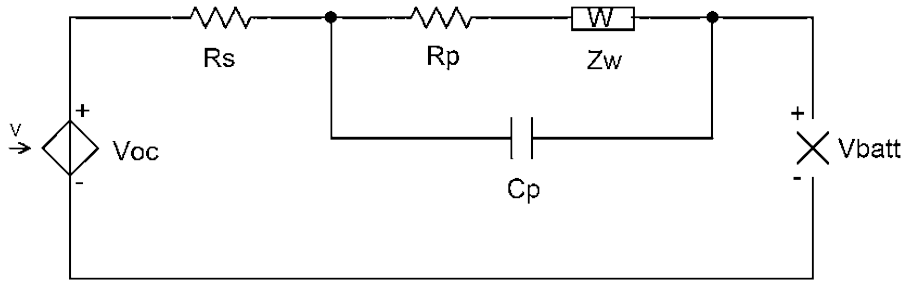


Figure 1.4. Battery Finite Warburg Impedance Model

Introduces an impedance (W) - heavily dependent on frequency - with the aim of taking into account the diffusion rate of the charges into the battery.

1.3 State of Art

In this section are reported some of the devices available on the market, each one is representative of a different market segment as it will be clear observing price and features.

- **Cadex C8000 Battery Testing System**

  Is a device capable to monitor battery cell balance using an ad-hoc load protocol. It is capable to test batteries with different technologies: Lead Acid, NiMH, NiCD, Li-ION, and Li-Phosphate.
  It can evaluate the whole battery state or the conditions of a single cell, taking into account up to five of them at a time.

  The measurement system works by charging and discharging the cells according to different loading tests such as GSM and CDMA; in such a way it is possible to define the lifecycle of the battery. It is also capable to measure the self discharge and to evaluate battery resistance with DC pulses (IEC61436) and impedance with 1kHz signal.
  In order to perform discharges with high currents - up to 240 A - it needs an external load.

  Another interesting feature is the capability to activate seemingly dead batteries.

  This device is a laboratory measurement system that can be used by battery producers to test and characterise the battery behaviour, therefore is really expensive and accurate and **costs about 11k€**.

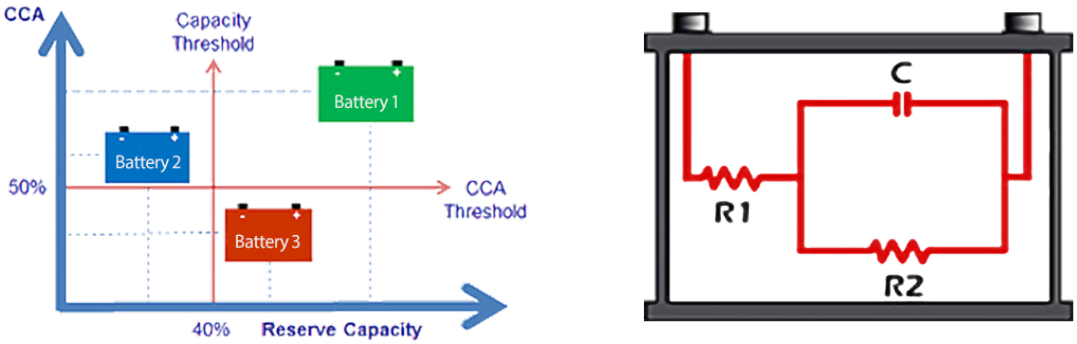
Figure 1.5. Cadex C8000 [3]

- **Cadex SPECTRO CA-12 Battery Rapid-Tester**

  This measurement system uses Electrochemical Impedance Spectroscopoy (EIS) to measure battery performances. It is specifically designed to evaluate the behaviour of lead-acid starting batteries. These are mainly defined by three parameters:

  - State of Charge
  - Cold Cranking Amp (CCA)
  - Capacity

  The second parameter defines the ability of a battery to start an engine at cold temperatures.



(a) Battery performance based on CCA and capacity

(b) Randle's model. R1 reflects resistance (CCA); R2 and C evaluate capacity.

Figure 1.6. SPECTRO CA-12 functioning

Thanks to these parameters, it is capable to define the State of Health of the battery, with respect to Figure 1.6b *Battery 1* is a good battery because both CCA and capacity are high. Battery 2 and 3 are two different examples of an unhealthy battery. Capacity is the most important indicator since its behaviour is more predictable than CCA one.

For example, due to capacity-fade, battery capacity drops unnoticed until the vehicle won't start for lack of energy. Capacity and CCA readings don't correlate; CCA can remain high while the capacity drops low gone.

Cadex sustains that Multi-model electro-chemical impedance spectroscopy (Spectro™) is more accurate than the common Randel's model in Figure 1.6a.

Figure 1.7. Cadex SPECTRO CA-12 [4]

- **BOSCH Electronic Battery Sensor**

  The electronic battery sensor (EBS) provides reliable and precise information on the status of 12-V lead-acid batteries while taking battery ageing effects into account. By providing this relevant information, the EBS allows for the implementation of an optimized electrical-power management (EEM) system in the vehicle and supports fuel- and CO2-saving technologies such as start/stop, coasting or recuperation.
  The electronic battery sensor (EBS) is attached to the negative terminal of a 12-V lead-acid battery with the terminal clamp and connected to the vehicle's body by a screw-on ground cable. The EBS measures the current using a shunt and determines the battery's voltage and temperature. These base parameters are required as input parameters for the integrated battery status tracking system (BZE) among other things. On that basis, the BZE algorithm predicts the battery's state of charge (SoC), the state of function (SOF) and the state of health (SoH).

  The SoC indicates how much power is available. Using the SoC, the electrical-power management system (EEM) in the main control unit regulates ancillary load to optimize battery life and fuel consumption.

  The SOF predicts the influence of the load profile on the voltage curve. Based on the SoH value, battery aging effects and their influence on the battery's capacity to store energy and its power output are calculated.

Figure 1.8. BOSCH Electronic Battery Sensor [5]

This device is intended for internal market; Bosch sells them to automotive manufactures to be installed on new cars, therefore there are no informations about specifications and prices.
Bosch sensor is the direct competitor of the device developed in these thesis since it has the same market segment, dimensions and functionalities.

- **Midac E-MOTION**

This device is intended for after-market and proposes to evaluate SoC, SoH and alternator health only by measuring battery voltage. It is sold in two versions: battery with integrated device, or kit. In both the cases the device connects via Bluetooth to a smartphone and shows all the data in an application.
The device is easy to install - since it does not require to measure current - it just needs to be connected in parallel to the battery, therefore it can also be installed by inexperienced people. On the other hand, by only measuring voltage, it can not be as accurate as the Bosch one though this is not mentioned by the manufacturer.

(a) (b)

Figure 1.9. Midac E-MOTION Kit (a) and Battery (b) with smartphone app [6]

1.4 SoC and SoH

Starting from the different models explained at 1.2.1, many algorithms to determine the SoC and SoH can be used.
Since the aim of this thesis is not to explain the different algorithms only the relevant ones will be appointed.

1.4.1 Voltage Method

This method is the most basic and allows to evaluate the SoC by simply measuring the voltge. By reading the discharge curve, that plots SoC as a function of the voltage, it is possible to detect the SoC corresponding to the acquired voltage.
This is an offline method and requires a stable voltage range, moreover the discharge test to obtain the homonym curve requires also a recharge period making this method too time consuming.
Finally, the discharge curve is different for every manufacturer and its behaviour is also modified by many external parameters such as temperature and battery ageing.[7]

In Figure 1.12 are reported some discharge characteristics evaluated using a discharge path at different frequencies. It is easy to observe that batteries from a different manufacturer have a different behaviour and also that the same battery has different characteristics during its life cycle.

1.4.2 Enhanced Coulomb Counting Algorithm

In the literature the releasable capacity $C_{releasable}$ of an operating battery is defined as the capacity released when it is completely discharged.

According to this, the State of Charged (SoC) is defined as the ratio between the releasable capacity and the rated capacity (C_{rated}), expressed as a percentage.

$$SoC = \frac{C_{releasable}}{C_{rated}} 100\% \tag{1.1}$$

In an old battery the rated capacity does not correspond to the maximum releasable capacity C_{max} due to defects, sulfation or others degration processes. For this reason the State of Health must be defined as the ratio between the maximal releasable and the rated capacity.

$$SoH = \frac{C_{MAX}}{C_{rated}} 100\% \tag{1.2}$$

When a battery discharges, the Depth of Discharge (DOD) is defined as the percentage of the capacity that has been discharged relative to C_{rated}.

$$DOD = \frac{C_{released}}{C_{rated}} 100\% \tag{1.3}$$

$C_{released}$ is the capacity discharged by an amount of current. By measuring a charging and discharging current (I_b), the DOD during a defined operating period τ can be evaluated as

$$\Delta DOD = \frac{\int_{t_0}^{t_0+\tau} I_b(t)dt}{C_{rated}} 100\% \tag{1.4}$$

By mean of this new metric, it is possible to find a relation between SoC and SoH as:

$$SoC(t) = SoH(t) - DOD(t) \tag{1.5}$$

In order to obtain a better model, taking into account the operation efficiency, DOD can be avaluated as:

$$DOD(t) = DOD(t_0) + \eta \Delta DOD \tag{1.6}$$

Considering the accumulation of DOD with time elapsing.

This is an online method, therefore the parameters are update during the lifetime of the battery but when considering a new battery SoH must be supposed equal to 100% and SoC must be evaluated with Voltage Method subsection 1.4.1.

To estimate the previously defined parameters voltage battery (V_b) and I_b must be measured; charging or discharging state can be discerned by observing the current sign. In fact, by knowing I_b it is possible to evaluate the ΔDOD value as in Equation 1.4, obtaining DOD (Equation 1.6) obtaining SoC as in Equation 1.5.
Every time that the battery is either fully charged or discharged, SoH must be evaluated respectively according to the SoC or DOD.
Everything explained before is schematized in flowchart Figure 1.10.

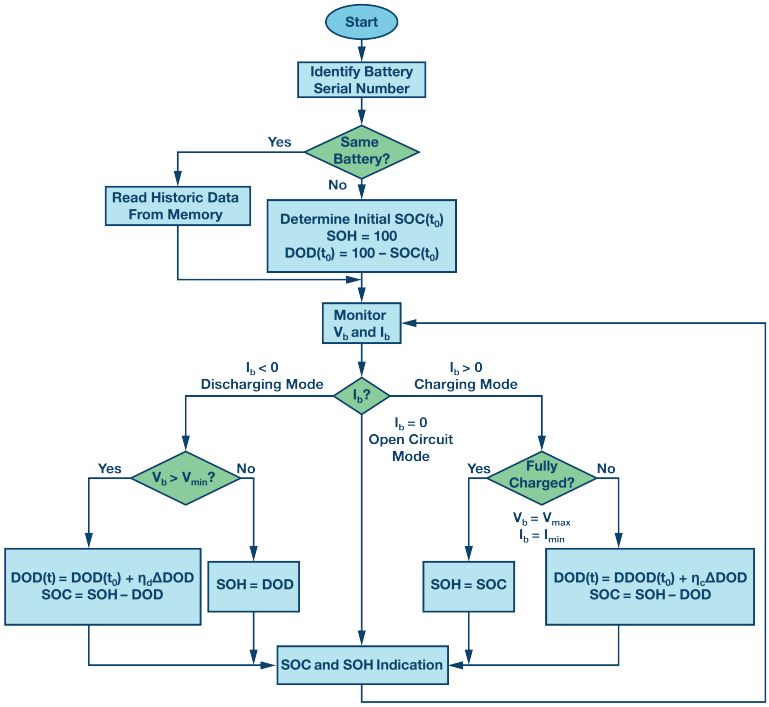


Figure 1.10. Flowchart of the enhanced coulomb counting algorithm [7]

This is an algorithm with e relatively low computational cost that can be implemented also in simple hardware. The drawback is the need of defining the initial conditions for a new battery.

1.4.3 Kalman Filter

Kalman filter is an algorithm that allows to estimate the inner state of a dynamic system, therefore can also be used to evaluate battery SoC. It is based on a recursive solution ensuring both state observation and prediction problems, its strength is related to the capability to provide dynamic error bounds related to state estimates. By using the Extended Kalman Filter, it is possible to take into account also the non-linearities of the system, adding the possibility of estimating SoH in real-time.
This method needs a precise model for the battery - usually the one in Figure 1.2 - and a precise identification of its parameters. Accurate initialization and large computing capacity are needed too. [7]

The algorithm to be implemented in this project - developed by *brainTechnologies S.r.l.* therefore not object of this elaborate - starts from this method with the aim of obtaining more reliable results by using a novel modelling approach, based on multi-model description of the physical behaviour of the battery. This approach allows to manage multi-modal disturbances providing a high level of robustness to the inference and reduces computational cost in order to implement it on a microcontroller. More informations about this topic are available at [2] and [9]

1.4.4 Impedance Spectroscopy

This method consists in injecting a small amplitude AC signal ($\approx 10mV$) in the battery observing its behaviour. By varying the frequency of the signal it is possible to observe the complex response of the signal getting a model like the one in Figure 1.4.[10].
This method, used by the device in Figure 1.7 need an expensive equipment, therefore is not suitable for a device to be installed on cars.

1.5 Starting Point - Measurement Campaign

It is possible to consider as a starting point the work carried out by C. Radici and G. Di Simone in their Master Thesis work.[8].
They set up a data acquisition system capable to discharge lead-acid batteries in a controlled way by using NI LabView.

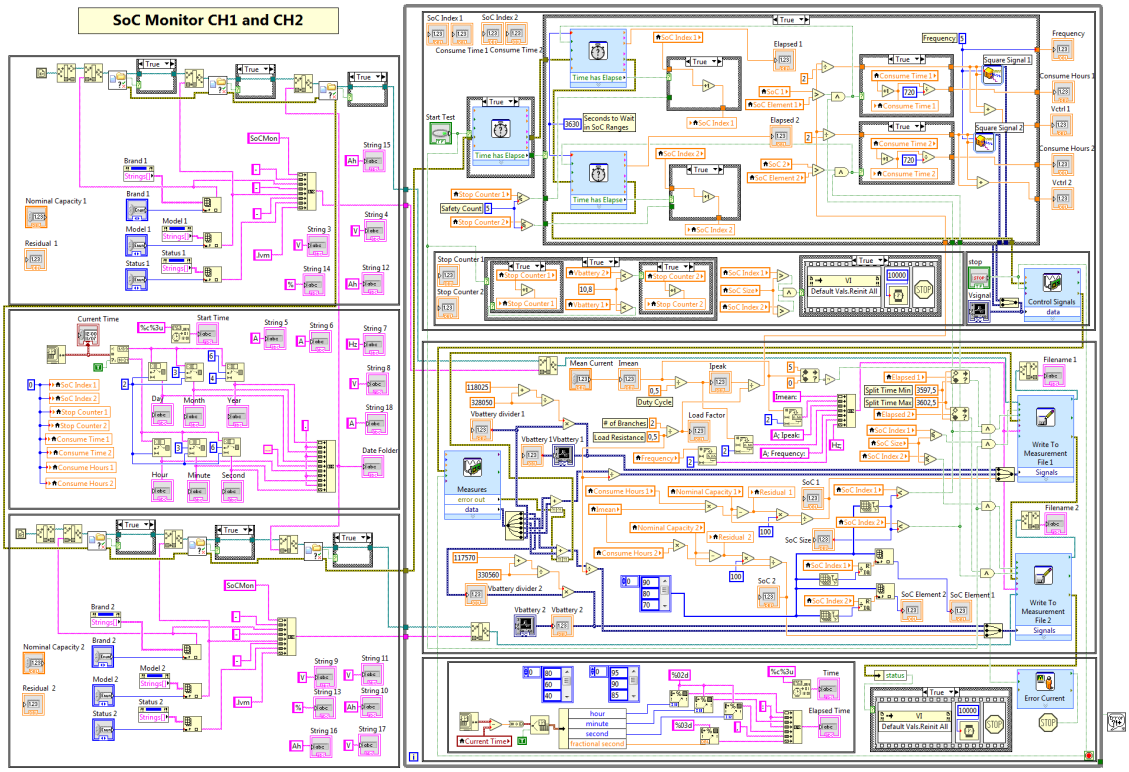
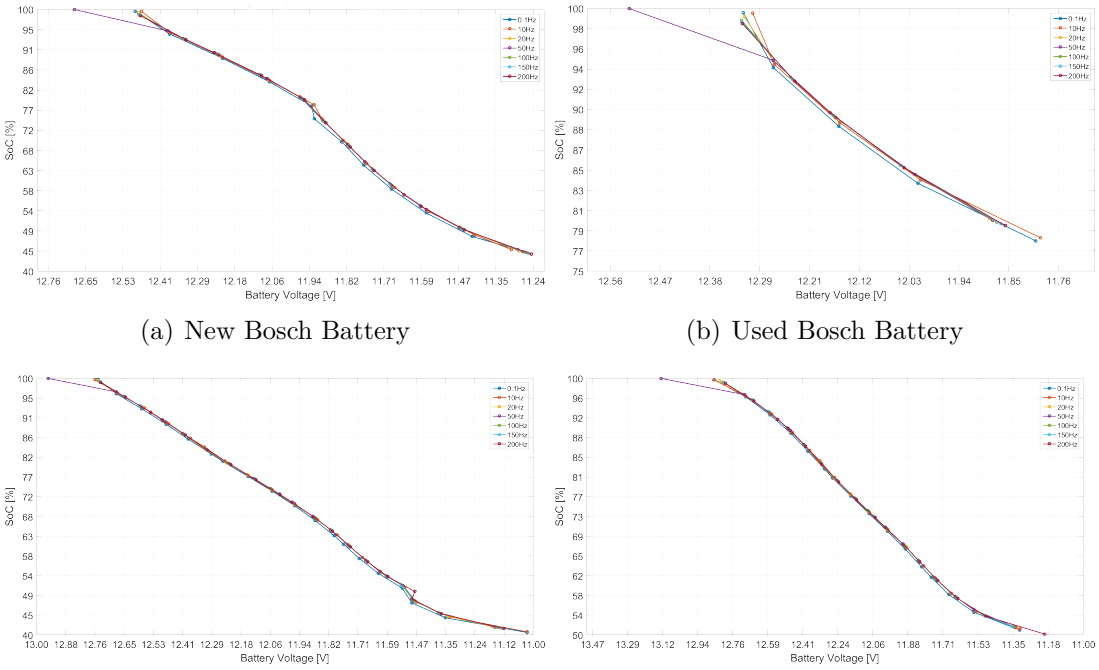


Figure 1.11. Data Acquisition System set up by C. Radici and G. Di Simone [8]

The measurement systems has the task to discharge a battery in a controlled way in order to obtain SoC and internal impedance. The discharge is performed by applying a load in such a way to obtain a current with a square wave shape. The maximum amplitude is fixed at 8A while the frequency varies between 0.1Hz and 200 Hz. Some of the obtained graphics are reported here.



(a) New Bosch Battery

(b) Used Bosch Battery

(c) New Energeco Battery

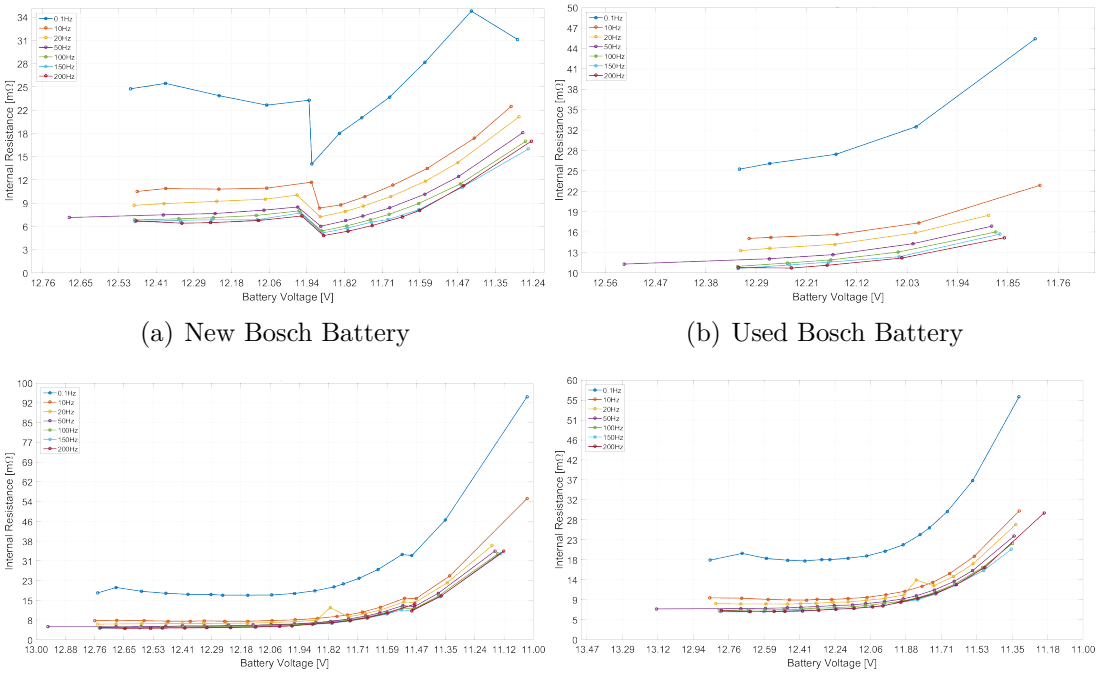
(d) Used Energeco Battery

Figure 1.12. SoC vs Battery Voltage at different control frequencies [8]

From Figure 1.12 it is clearly visible that the discharge curves associated to different batteries - Bosch and Energeco brands in this case - are really different. For new batteries Bosch has a voltage corresponding to SoC 100% of about 12.45V while Energeco in the same conditions has 12.76V. By considering the same batteries after a period of ageing, their curves are really different. This is the reason why the "Voltage Method" to obtain SoC described in subsection 1.4.1 is not a stand-alone algorithm but needs many informations from the battery datasheet.
It is interesting to observe that SoC curve does not strongly depend on the frequency of the discharge square wave.

On the other hand, for what concerns internal impedance with respect to battery voltage there is a strong dependence on the control frequency. This result is interesting because endorses the inductive behaviour of the battery, representable as in Figure 1.2.
This could allow to obtain values to be inserted in the equivalent circuit; unfortunately as usual the behaviour is different during the various life-stages of the battery.

From this awarenesses the need of an online estimator like the Kalman Filter arises.



(a) New Bosch Battery

(b) Used Bosch Battery

(c) New Energeco Battery

(d) Used Energeco Battery

Figure 1.13. SoC vs Battery Voltage at different control frequencies [8]

Part II

Implementation

Chapter 2

Analysis of the task

"Designed by Freepik"

In this chapter the requirements and main issues related to the implementation are going to be analysed taking into account the typical behaviour of a battery-car system.

2.1 Executive Summary

BAT-MAN objective is the realization of a new low cost product idea able to evaluate State of Charge (SoC) and State of Health (SoH) of a Lead acid 12V battery; therefore it should:

- Recognize critical states, thus permitting to avoid unexpected service interruptions
- Inform the user or a supervision system of the actual SoC and SoH
- Collect data and elaborate useful battery stats

The developed technology should be extendable to Lithium Battery Management Systems so to overcome tomorrow's challenges.
Moreover the device should be included in both in-market and after-market production lines. The device must be composed of:

- PCB to be connected to the battery terminals with a microcontroller as its core
- Bluetooth Low Energy (BLE) communication system
- Human-Machine Interface (HMI) through smartphone app downloadable from major app stores

2.2 Targets

The commitments assigned by the customers - embodied by *Politecnico di Torino* and *brainTechnologies* upon request of *MESAP* - impose that the device to be developed should withstand the following requirements:

- Focus on Lead-acid technology since it is the most commonly used in automotive field
- Connectable to battery terminals, therefore battery supplied
- BLE protocol capabilities
- Current and Voltage measurement for a better estimation
- PCB solution with limited size
- Controlled by a microcontroller
- Automotive certifications compliant

In order to evaluate state of charge and state of health it is necessary to collect the electrical parameters of the battery.
As explained in section 1.3 the most part of the devices available on the market can only measure voltage, which however cannot provide an accurate estimation of SoH and SoC. Hence the decision to add an important feature: current measurement allows to use more sophisticated algorithm.
The work done until now has to face with issues related to the high currents involved during the different operating conditions of a car.

This is the keypoint on which the implementation choices are going to be based; therefore a study has been carried out evaluating the typical values of current for every single part of the equipment as shown in Table 2.1.

Equipment	Current Sank
Ignition	2-9A
Radio	0.5-5A
Windshield Wipers	7.5A
Headlamps (Low Beam, Dim)	17-18A
Headlamps (High Beam, Bright)	19-20A
Parking lights	4-10A
Brake lights	6-11A
Interior lights	2-4A
Bonnet Light	0.5-1A
Horn	4A
Power Window (One window)	5A
ABS Brakes (Max)	14A
Boot Light	0.5-1A
Blower (Heater, Air Conditioner)	10-14A
Heated Rear Window Defogger	13-28A
Heated Seat	4-5A
Power Seat Motor	10-13A
Summer Starting (Petrol)	150-200A
Summer Starting (Diesel)	450-550A
Winter Starting (Petrol)	250-350A
Winter Starting (Diesel)	700-800A

Table 2.1. Typical current loads of passenger cars

Considering the worst case, maximum total current load, referring to a diesel passenger car, of about 985A was estimated.
The whole project started from this point because it is not easy to handle such a high current on a PCB. Moreover, since the battery is discharged when the equipment parts are turned on, and charged when the alternator is working, the current must be measured in both directions.

2.3 Current Sensing Topologies

To evaluate a current there are mainly two measurement topologies:

- Hall effect sensor
- Shunt resistor

2.3.1 Resistive Current Sensing

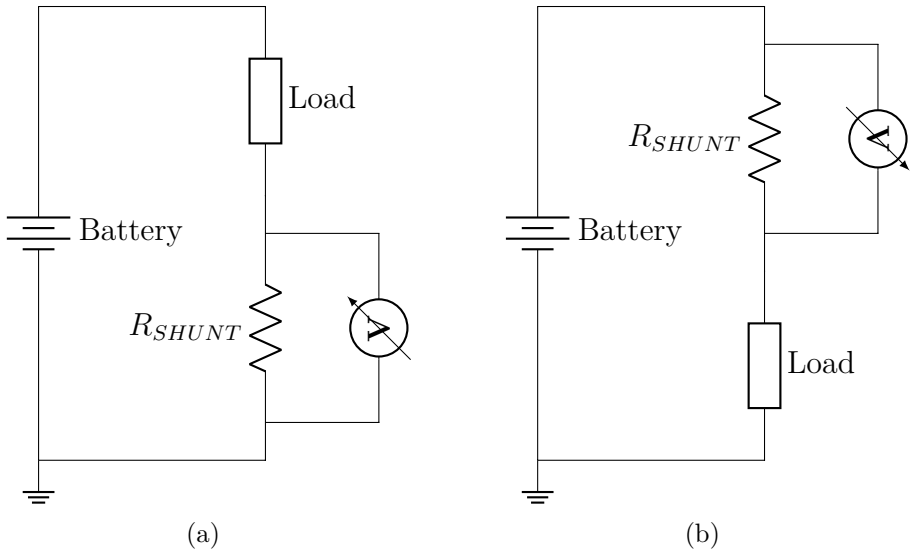


Figure 2.1. Shunt measurement topologies

Shunt resistor insertion is the easiest and most commonly used current sensing topology. It exploits the voltage drop associated with an electrical current flowing through a resistor placed in series between a source and its load.
Simply referring to Ohm's Law, by knowing the resistance value, it's possible to evaluate the current flow having

$$I = \frac{V}{R}$$

The main reason to choose this approach is that resistors are inexpensive and available in many formats, values, power ratings, and accuracies. However, this method implies the necessity to insert it in series so it could be difficult to be integrated when applied to existing devices like in this analysis. This insertion also means that the measuring and measured device are not electrically insulated. However, the lack of electrical isolation, is only relevant for voltages which could be dangerous for human safety.

There is an important trade off between having a high current measurement precision and a low voltage drop across the resistor. In fact, the voltage delivered to the load should be as similar as possible to the source one to guarantee the proper behaviour of the system. On the other hand, an higher drop is obviously much easier to be measured.

Another relevant problem happens when both the terminals of the shunt are at high voltage, thus causing a large common mode voltage. This non preferred configuration is defined as *high side* and shown in Figure 2.1, in contrast to it the *low side* topology (Figure 2.1) should be preferred. In this way, since the voltage drop must be low, both the potentials are approximately zero reducing drastically the common mode component evaluated as:

$$V_{CM} = \frac{V_+ + V_-}{2}$$

Moreover, the current flow causes a power dissipation by Joule Effect that not only implies an efficiency derating but also a temperature spread that may negatively affect the measurements; the need of cooling the resistor arises.

Pros:

- Available in many formats, values, power ratings, accuracy, material
- Affordable for High precision
- Robust to EM disturbances
- High Temperature Ratings
- High Temperature stability
- Low noise
- Wide dynamic range

Cons:

- Diffuclt to be integrated in existing devices
- Power Dissipation
- Trade-off between
- Lack of electrical insulation
- Possible common-mode component
- Possible High Voltage ratings
- Parasitic Inductance

2.3.2 Hall Effect Sensor

This kind of sensors are used for indirect measurements as they exploit the magnetic field induced by a current flowing into a wire. There is a wide variety of methods that employ Hall Effect sensing, starting from the *Free-Space Current Sensing*[11] up to *Toroidal Current Sensors*.

Free-Space Current Sensing

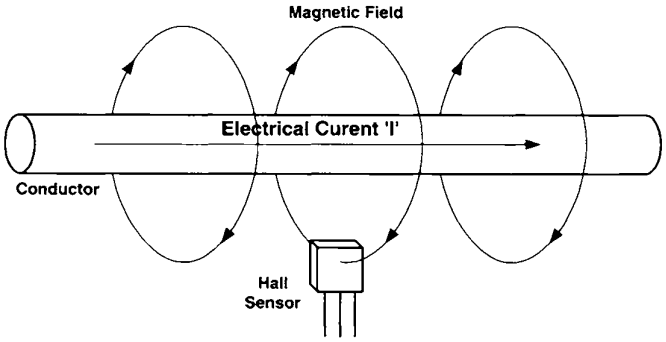


Figure 2.2. Current sensing in free space around conductor

The first one is a linear Hall Effect sensor that must be place in proximity of the conductor under test. This method can hardly ever be applied because is dependent on the distance from the wire

$$B = \frac{\mu_0 I}{2\pi r}$$

Where:

B is the induced magnetic field

I is the flowing current

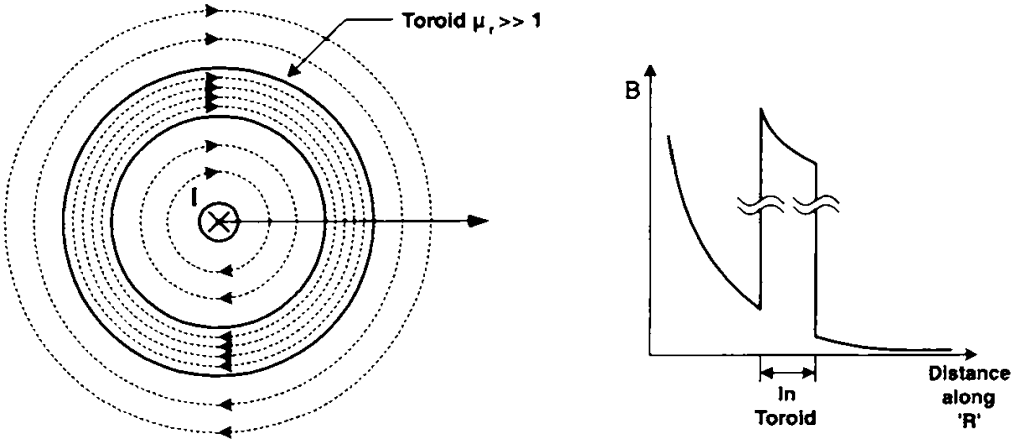
$\boldsymbol{\mu_0}$ is the magnetic permeability of the air

r is the distance from the centerline of the conductor to the sensor

This device is also heavily influenced by the Earth magnetic field: the two fields are added as vectors and therefore the result depends on the spatial orientation of the sensor.

Toroidal Current Sensor

The free-space current sensor lacks of sensitivity and susceptibility to outside interferences, therefore we should move to another topology obtaining an highly sensitive current sensor which is also immune to external interferences. This result is achieved by using a high permeability toroid around a current carrying conductor. When the wire is centred in the toroid, the toroid's presence will not affect the overall shape of the field, on the contrary it will cause significant increase in flux density as visible in Figure 2.3.



(a) (b)

Figure 2.3. Magnetic field in and near toroid (a) and intensity profile (b) [11]

The new equation of the field is the following:

$$B = \frac{\mu_0 \mu_r I}{2\pi r}$$

Where $\boldsymbol{\mu_r}$ is the relative permeability of the toroid material which is a high value (>100-1000). As a consequence of this equation the field is much less dependent on the exact placement of the conductor.

Hall effect sensors are non linear, especially at zero crossing and near the magnetic core saturation region. Moreover it's not suggested to use them in an environment where electromagnetic disturbances are present.
They have a limited operating temperature range, typically up to $85°C$, too low for automotive specification. This behaviour is due to the ferrite material characteristic B-H curve; at the *Curie temperature* the core permeability sharply disappears and the ferrite material loses its magnetic properties, thus reducing flux density in the core material.[13]

Pros:

- Affordable for Low precision
- Intrinsically insulated
- Not relevant power dissipation
- Magnetic offset can be removed
- Easy to post-install, especially open-loop

Cons:

- Limited Temperature rating
- Susceptible to EM disturbances
- Need of modulation to avoid noise problems
- Electrical offset affect by temperature
- Magnetic offset due to core magnetization
- Nonlinear behaviour wrt temperature
- Difficult to be calibrated

2.3.3 Current Sensors Comparison

CATEGORY	SHUNT-BASED	HALL-BASED
Solution size	Similar	Similar
Offset	Very Low	Medium
Offset drift over temperature	Low	Medium
Accuracy	<0.5% after calibration	<2% after calibration
Noise	Very Low	High
Bandwidth	Similar	Similar
Latency	Similar	Similar
Nonlinearity	Very Low	High
Long-term stability	Very High	Medium
Cost	Similar	Similar
Vibration Impact	Very Low	Low
Power dissipation	Low	Very Low
Customization	Flexible	Limited

Table 2.2. Difference between Shunt- and Hall-based isolated Current Sensing [12]

The pros and cons exposed above have then to face the requirements of the project. The main reason why an Hall-based sensor is not suitable for an automotive device is that it is too susceptible to external EM disturbances and temperature variations. Moreover, the maximum temperature is too low ($85°C$) for automotive standards which state a maximum rating of not less than $105°C$. The Hall sensor is also less accurate and suffers of non-linearity problems.
On the other hand, shunt topology is linear, low noise and robust to EM disturbances which are important tasks for the application in analysis. For these reasons the latter topology has been chosen; this implies to deal with the insertion problems that will be deeply analysed in the following chapters.

Finally, it is important to take into consideration that due to the high currents involved the resistance to be chosen must have a very low value both to avoid excessive power dissipation and high voltages that could be tricky to be measured.

2.4 Conditioning Circuit Concept

The voltage across the shunt resistor must be acquired through an AD converter; before doing this it is necessary to properly amplify the signal since it is a very low value. In order to do this a starting point circuit could be the one in Figure 2.4.

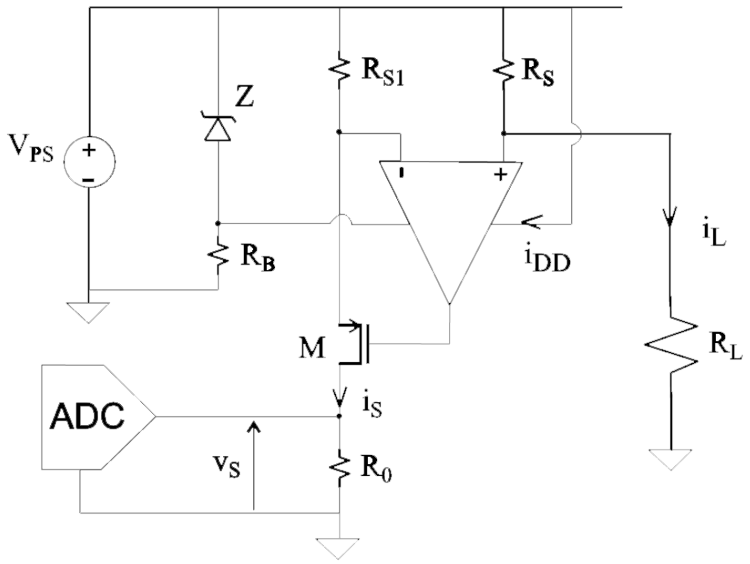


Figure 2.4. Concept of a conditioning sensing circuit for a shunt resistor

It is a negative feedback circuit where the sank current i_L causes a voltage drop on R_L. This, thanks to the reaction, causes a proportional variation of the current i_s that passes on the output resistor R_0.

Instead of implementing a discrete components circuit like this, we chose to adopt a commercial integrated solution as described in section 3.1 to obtain both a smaller layout and more precise results.

Chapter 3

Implementation

Once the targets to be reached have been set, the following step is the components choice according to the requirements. These will then have to be represented and connected in a schematic before placing them into the layout which is the final designing step before the physical building of the device. [1]

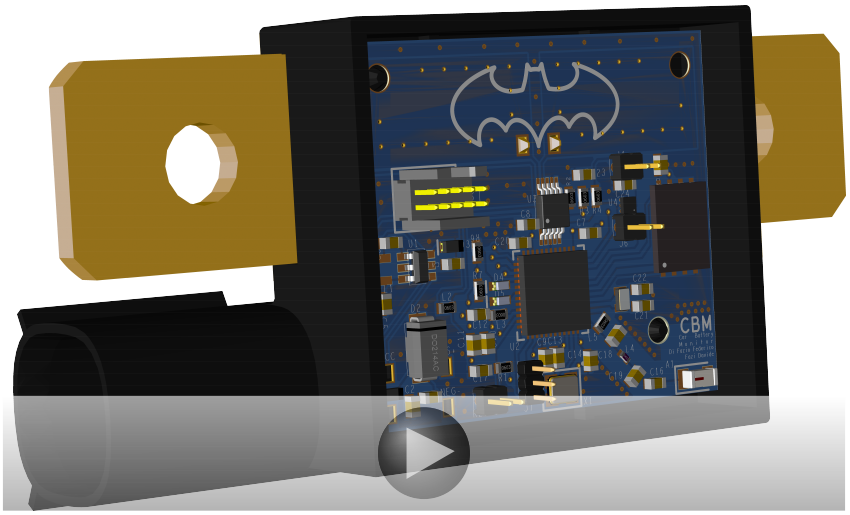


Figure 3.1. A 3D model of the complete physical product

[1] If the Figure does not appear interactive, please enable this function and click on it or use a recent version of Adobe Reader

3.1 Components choice

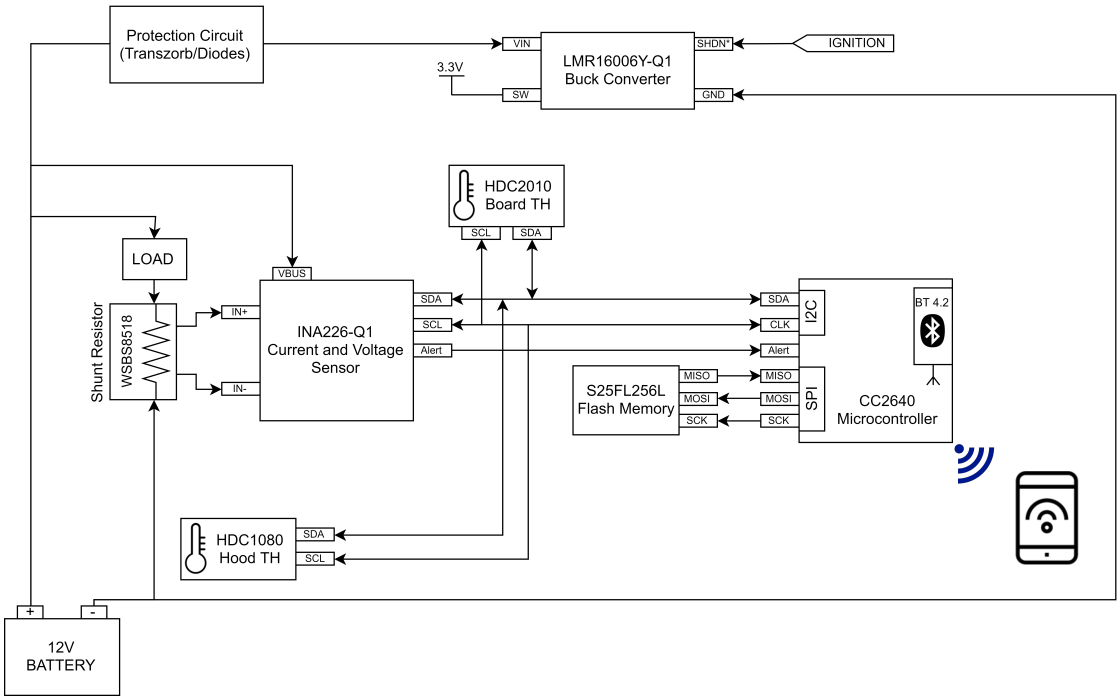


Figure 3.2. Block diagram

This project is mainly composed by these blocks:

- Protection Circuit based on transzorb: used to protect the whole system from transient voltage
- LMR16006Y-Q1: PWM DC/DC buck regulator used to step-down the 12V supply voltage to 3.3V
- WSBM8518: 50$\mu\Omega$ shunt resistor
- INA226-Q1: used to amplify the voltage across the current shunt and transmit it to the μC
- CC2640R2F-Q1: BT low energy Wireless MCU used to manage data acquisition and transmission

3.1.1 Voltage Regulator

The voltage source is represented by a 12V battery, therefore a step-down regulator is needed in order to properly supply the logic. In order to reduce the board occupation, we opted for an integrated solution instead of a discrete one that would have needed many components.
There are mainly two possible choices: [14]

- **LDO converter**

  Pros

  - Low Drop Out
  - Stable output during current load transients
  - Low complexity
  - Low noise
  - Low price

  Cons

  - Low efficiency (dependent on voltage drop)
  - High area occupation
  - High power dissipation (possible need of heat sink)
  - High quiescent current

- **Switching regulator**

  Pros

  - High efficiency (typical >80%)
  - Low area occupation
  - Low power loss
  - Very low quiescent current

  Cons

  - Spikes during current load transients
  - High drop out
  - High cost
  - High noise
  - High complexity

A Chopper-Buck converter solution has been chosen because of its better efficiency and lower board area occupation compared to an LDO. On the other hand, this solution will introduce noises that will have to be studied during the testing procedures.
Moreover, at the first stages of this project there were not components having a high transient current. Only later we added a Flash memory which has made us aware of this issue.

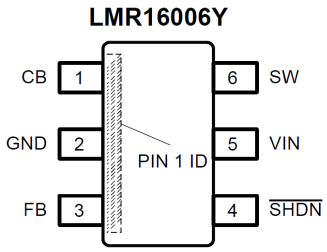
The choice has been narrowed down to two devices selecting them by automotive grade, maximum input rating voltage, enable pin presence and size. The enable pin is an important feature because it allows the device to be automatically switched on at car ignition.

PARAMETER	**TPS560200-Q1**	**LMR16006Y-Q1**
Automotive grade	AEC-Q100	AEC-Q100
$V_{in,MAX}$	17V	40V
Fixed output	no	yes
Adjustable output	yes	yes
Switching frequency	600 kHz	2.1MHz
Quiescent current	0.06 mA	0.028 mA
Maximum EN voltage	7 V	65 V
Size	3.00 x 3.00 mm^2	1.60 x 2.90 mm^2
Thermal protection	yes	yes
Over current protection	yes	yes

Table 3.1. Comparison between the buck converters in exam [15], [16].

From Table 3.1 it can be observed that the main electrical feature to prefer LMR16006Y-Q1 to TPS560200-Q1 is the maximum enable voltage; it is possible to provide the enabling 12V signal without the need of a conditioning circuit. Moreover, an higher switching frequency implies both a reduction in size of the IC and lower values of the external passive components (C and L), presumably lowering their packages dimension.
An high switching frequency allows to improve transient response and to avoid frequency bands in which noise would be disruptive such as in vehicle motion or AM communication.
As a drawback, efficiency is worse due to the greater number or constant energy switching events per time. [18]

LMR16006Y-Q1



Pin Functions

PIN		I/O	DESCRIPTION
NAME	NUMBER		
CB	1	I	Switch FET gate bias voltage. Connect C_{boot} capacitor between CB and SW.
GND	2	G	Ground connection.
FB	3	I	Feedback Input. Set feedback voltage divider ratio with $V_{OUT} = V_{FB}$ (1 + (R1/R2)).
$\overline{SHDN}$	4	I	Enable and disable input (high voltage tolerant). Internal pull-up current source. Pull below 1.25 V to disable. Float to enable. Establish input undervoltage lockout with two resistor divider.
VIN	5	I	Power input voltage pin. Input for internal supply and drain node input for internal high-side MOSFET.
SW	6	O	Switch node. Connect to inductor, diode, and C_{boot} capacitor.

Figure 3.3. LMR16006Y-Q1 pinout [15]

The main features of the device are as follows:

- Grade 1 qualified for automotive applications
- Input voltage range 4V to 40V
- $28\mu A$ standby current in ECO mode
- Overcurrent and temperature protection
- Available in fixed output 3.3V
- High voltage enable input (up to 65 V)
- Maximum output current 600mA
- Switching frequency 2.1MHz
- Fixed 3.3V output
- High Efficiency

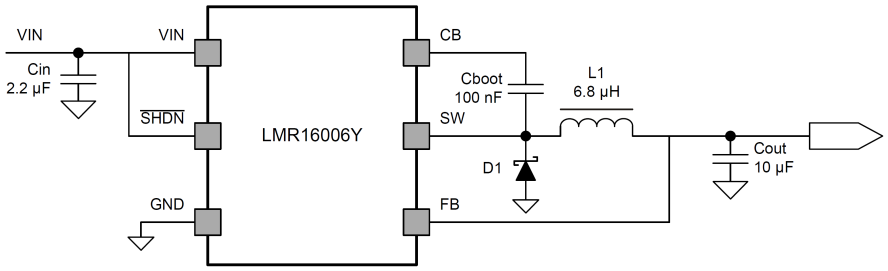


Figure 3.4. Typical fixed output configuration

The LMR16006Y-Q1 is a PWM DC-DC buck (step-down) regulator suitable for battery operating circuits, that allows an input voltage of 4 to 40 V . There is no need of feedback resistors because the chosen model has an internal voltage divider which provides a fixed output of $3.3V \pm 3\%$.

Input capacitor Referring to Figure 3.4 Cin is a bulk capacitor which prevents large switching voltage transients at the input of the converter. The capacitor to be chosen must have a ripple current rating greater than the maximum input current ripple which is calculated as:

$$\begin{aligned} I_{cirms} &= I_{out}\sqrt{\frac{V_{out}}{V_{in,min}}\frac{V_{in,min}-Vout}{V_{in,min}}} = \\ &= 73mA \cdot \sqrt{\frac{3.3V}{7V}\cdot\frac{7V-3.3V}{7V}} = 36.44mA \end{aligned} \tag{3.1}$$

The input capacitance value - that should be chosen between $1\mu F$ and $10\mu F$ - determines the maximum input ripple voltage as:

$$\Delta V_{in} = \frac{0.25\ I_{out,max}}{f_{sw}\ C_{in}} = \frac{0.25 \cdot 73mA}{2.1MHz\ 10\mu F} = 869\mu V \tag{3.2}$$

We chose $10\mu F$ to have the minimum ripple since the input voltage must be measured.

Bootstrap Capacitor C_{BOOT} is needed to perform a charge pump operation. The value must be chosen between $0.1\mu F$ and $1\mu F$ depending on the difference between input and output voltage. The lowest value has been selected since output voltage is about 20% of the input one.

Output Inductor When selecting the output inductor, inductance, peak current and DC-resistance should be properly chosen. For this sake, peak-to-peak inductor ripple current and input-output voltages have to be evaluated.

$$\begin{aligned} L_{o,min} &= \frac{V_{in,max} - V_{out}}{0.2\ I_o}\ \frac{V_{out}}{f_{sw}\ V_{in,max}} = \\ &= \frac{16V - 3.3V}{0.2 \cdot 73mA} \cdot \frac{3.3V}{2.1MHz \cdot 16V} = 85\mu H \end{aligned} \tag{3.3}$$

Therefore a $100\mu H$ value was chosen.

$$\begin{aligned} I_{ripple} &= \frac{V_{out}\ (V_{in,max} - V_{out})}{f_{sw}\ V_{in,max}\ L_o} = \\ &= \frac{3.3V \cdot (16V - 3.3V)}{2.1MHz \cdot 16V \cdot 100\mu H} = 12.5mA \end{aligned} \tag{3.4}$$

$$\begin{aligned} I_{L-RMS} &= \sqrt{I_o^2 + \frac{1}{12}\ I_{ripple}^2} = \\ &= \sqrt{(73mA)^2 + \frac{1}{12}(12.5mA)^2} = 73.1mA \end{aligned} \tag{3.5}$$

$$\begin{aligned} I_{L-peak} &= I_o + \frac{I_{ripple}}{2} = \\ &= 73mA + \frac{12.5mA}{2} = 79.25mA \end{aligned} \tag{3.6}$$

Output Capacitor In order to select the proper value of C_{out} it must be taken into account that it influences modulator pole, output voltage ripple and how the regulator responds to a large change in load current.
The main issue to be faced is the one regarding the output voltage ripple since, in a measurement system, source fluctuations may compromise the acquisitions reliability. Furthermore, the presence of a flash memory - sinking high currents during erase and program phases - implies a special care to this aspect.

$$C_{out} > \frac{2\ \Delta I_{out}}{f_{sw}\ \Delta V_{out}} = \frac{2 \cdot 73mA}{2.1MHz \cdot 10mV} = 6.95\mu F \tag{3.7}$$

$$C_{out} > L_o\ \frac{I_{oh}^2 - I_{ol}^2}{V_f^2 - V_i^2} = 100\mu H \cdot \frac{(73mA)^2 - (0.744mA)^2}{(3.305V)^2 - (3.295V)^2} = 8.07\mu F \tag{3.8}$$

$$C_{out} > \frac{1}{8\ f_{sw}}\ \left(\frac{V_{o,ripple}}{I_{L,ripple}}\right)^{-1} = \frac{1}{8 \cdot 2.1MHz}\left(\frac{10mV}{12.5mA}\right)^{-1} = 74.25nF \tag{3.9}$$

Since the constraint is represented by the second equation stating $C_{out} > 8.07\mu F$, the value of the output capacitor has been chosen equal to $10\mu F$. The last thing to be considered is the ESR resistor of the capacitor that can be calculated as:

$$R_{ESR} < \frac{V_{o,ripple}}{I_{L,ripple}} = \frac{10mV}{12.5mA} = 0.8\Omega \tag{3.10}$$

3.1.2 Protection Circuit

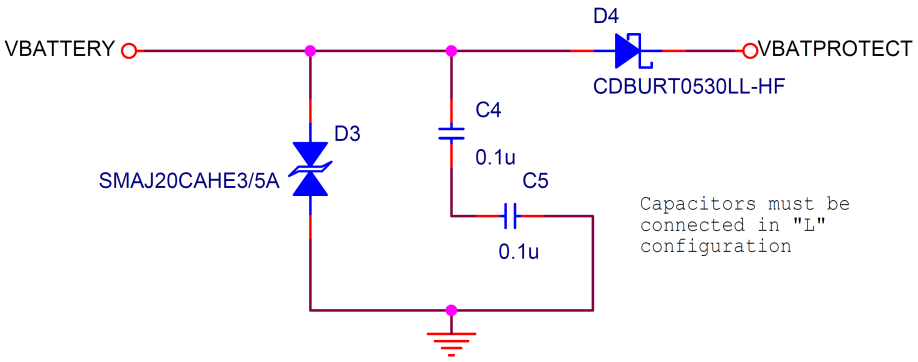
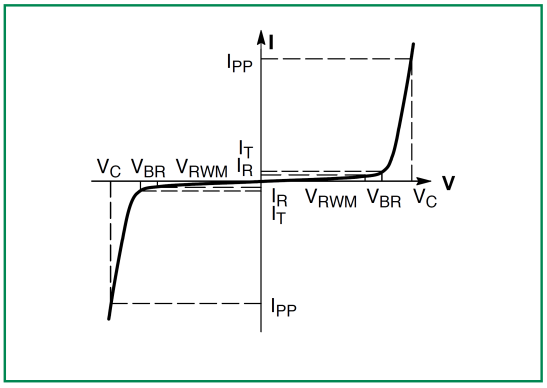


Figure 3.5. Circuit for Electrical Transient Protection

The transient voltage suppressor (TVS) is placed to suppress coupling transients according the ISO7637-3 standard upon which the whole circuit relies. This standard takes care of the two main types of coupling effects on supply lines: capacitive or inductive. The strength of pulses - according to level IV - would be in magnitudes of –60 V and 40 V.
To do this we used a Transzorb (TVS diode) with a working peak reverse voltage of 20V, enough to protect the buck converter from overvoltages which could damage the device. The breakdown voltage is between 22.2V and 24.5V (test current 1mA) and the Maximum Clamping Voltage is 32.4V with a Maximum Peak Pulse Surge Current of 12.4A, a value that we will never reach.
Graph in Figure 3.6 explains the TVS diode working zones.



Symbol	Parameter
I_{PP}	Maximum Reverse Peak Pulse Current
V_C	Clamping Voltage @ I_{PP}
V_{RWM}	Working Peak Reverse Voltage
I_R	Maximum Reverse Leakage Current @ V_{RWM}
V_{BR}	Breakdown Voltage @ I_T
I_T	On-State Current
I_F	Maximum Temperature Coefficient of V_{BR}
V_F	Forward Voltage @ I_F
VF	Forward Voltage @ I_F

Figure 3.6. Transzorb Transcharacteristic [17]

Another noteworthy parameter of the TVS diode is the peak power rating, important because it is proportional to the package size of the diode. This rating can be evaluated considering the worst case of a Pulse B from ISO 7637-3:

- $V_{pulse} = 40V$
- $R_{source} = 50\Omega$
- $V_{clamp} = 32.4V$ (for $10/1000\mu s$)

$$I_{source} = \frac{V_{pulse} - V_{clamp}}{R_{source}} = \frac{40.0V - 32.4V}{50\Omega} = 0.152A \tag{3.11}$$

So the peak power on the TVS is

$$P_{TVSpk} = I_{source} \cdot V_{clamp} = 0.152A \cdot 32.4V = 4.9248W \tag{3.12}$$

Therefore, the selected device is compliant with this specification since it's limit is $400W$.

The Schottky diode is used to protect the Buck from polarity inversions that could take place when the TVS diode goes into breakdown or when a negative voltage is applied. The choice to use a Schottky was made according to its very low threshold voltage V_γ allowing to get a measure of the battery voltage much closer than the original one.
The tank capacitors have the role to supply the buck when the transzorb is conducting. They are placed in L configuration to be compliant with automotive vibrational standards.

3.1.3 Current and Voltage Sense Amplifier

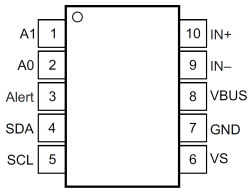
Shunt voltage must be negligible with respect to the supply voltage in order to not affect the correct behaviour of the car electric equipment. This voltage - which can go down to some μV - has to be amplified in order to be acquired by an ADC converter.
Many solutions have been analysed having as common features Q1 Automotive grade and bidirectional capability.

PARAMETERS	**INA226-Q1**	**INA225-Q1**	**INA220-Q1**	**LMP860x-Q1**
Shunt V measurement	Yes	Yes	Yes	Yes
Bus V Measurement	Yes	No	Yes	No
Output Type	I2C	Analog	I2C	Analog
Resolution	16	-	12	-
Shunt Accuracy	±0.02%	±0.1%	±0.2%	±0.2%
Bus Accuracy	±0.02%	-	±0.2%	-
Alert function	Yes	No	No	No
CMRR	126 dB	95 dB	120 dB	90 dB
Bandwidth	3.5 kHz	200 kHz	5.5 kHz	60 kHz
Gain	1	200 100 50 25	1 0.5 0.25 0.125	100 50 20
Programmable gain	No	Yes	Yes	No
Gain Error	0.1	0.15	0.5	0.5
Offset Voltage	$10\mu V$	$150\mu V$	$50\mu V$	$1000\mu V$

Table 3.2. Comparison Among different sense amplifier [19], [21], [20], [22]

Among the different solutions, at a first glance, the digital ones have been preferred since I2C communication is more noise-resistant than analog one.
Between the remaining components INA226-Q1 has been chosen relying on its higher resolution and CMRR, lower offset and the presence of an alert pin. It should be noticed that - although its bandwidth is the lowest one - this is not a problem because current variations in a car-battery system are pretty slow. Actually, this drawback is a strength since all the components at higher frequencies are just noise or disturbances.

INA226-Q1



Pin Functions

PIN		I/O	DESCRIPTION
NAME	NO.		
A0	2	Digital input	Address pin. Connect to GND, SCL, SDA, or VS. Table 2 shows pin settings and corresponding addresses.
A1	1	Digital input	Address pin. Connect to GND, SCL, SDA, or VS. Table 2 shows pin settings and corresponding addresses.
Alert	3	Digital output	Multi-functional alert, open-drain output.
GND	7	Analog	Ground.
IN+	10	Analog input	Connect to supply side of shunt resistor.
IN–	9	Analog input	Connect to load side of shunt resistor.
SCL	5	Digital input	Serial bus clock line, open-drain input.
SDA	4	Digital I/O	Serial bus data line, open-drain input/output.
VBUS	8	Analog input	Bus voltage input.
VS	6	Analog	Power supply, 2.7 V to 5.5 V.

Figure 3.7. INA226-Q1 pinout [19]

The main features of the device are:

- Grade 1 qualified for automotive applications
- High accuracy (0.1% Max gain error, $10\mu V$ Max offset)
- 16 bit resolution with internal $\Delta\Sigma$ ADC
- Bus voltage sensing from 0 to 36 V
- I^2C or SMBUS interface
- Programmable address
- Supply voltage between 2.7V and 5.5V
- Multiplexed voltage and current reads
- Programmable alert pin

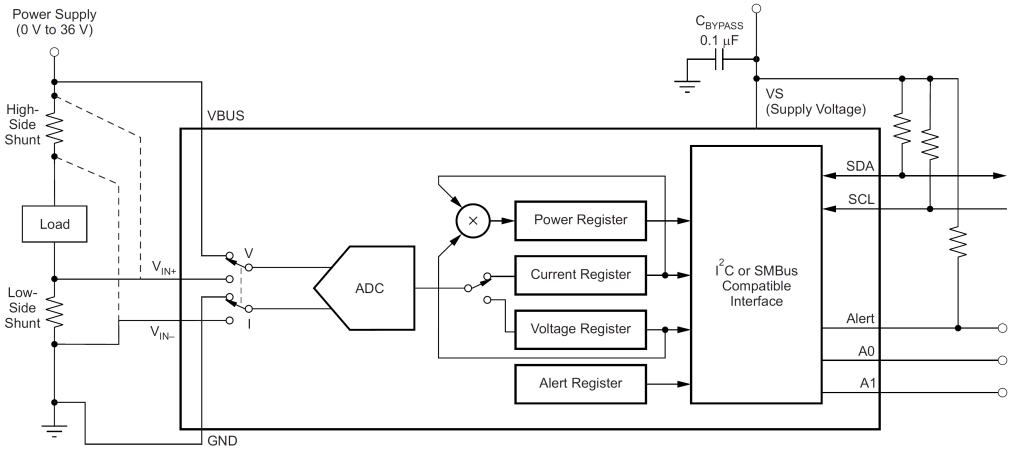


Figure 3.8. INA226Q1 Internal block diagram

The INA226-Q1 is an instrumentation amplifier circuit capable to acquire both a voltage across a shunt resistor and a bus voltage. The idea is to measure the voltage across a battery and, at the same time, the current sank by a load. It provides a digital I^2C output which allows to program its internal registers and read measured values. The address can be selected by stacking A0 and A1 pins to GND,SCL,SDA or VS; in our application they have both been stacked to VDD, obtaining the address 1000101.

Digital Communication

By using the digital interface we can perform two kind of operations:

- **Write on a register**
  The device can be programmed by writing on specific registers. The master starts the communication by sending the INA226-Q1 address with the $R/\overline{W}$ bit low. After the acknowledge reception the master transmits the register address followed by the two bytes of data (16 bit registers). The communication is ended by the master generating a start or stop condition

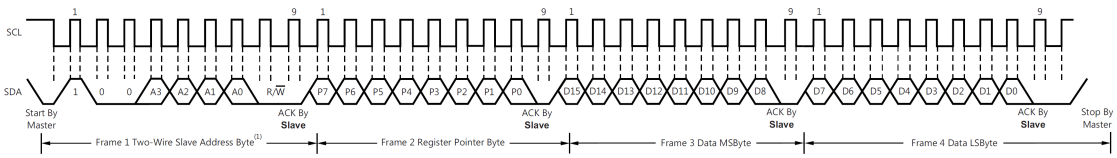


Figure 3.9. Timing diagram for write format

- **Read from a register**
  This is useful to acquire current, voltage and power values from the respective registers. When reading from the device, the last value stored in the register pointer by a write operation determines which register is read during a read operation. To change the register pointer for a read operation, a new value must be written to the register pointer. The master then generates a start condition and sends the slave address byte with the $R/\bar{W}$ bit high to initiate the read command. The next byte is transmitted by the slave and is the most significant byte of the register indicated by the register pointer. This byte is followed by an Acknowledge from the master; then the slave transmits the least significant byte. The master can terminate the communication by sending a not acknowledge or a start or stop condition.
  It is important to mention that all data bytes are transmitted MSB first.

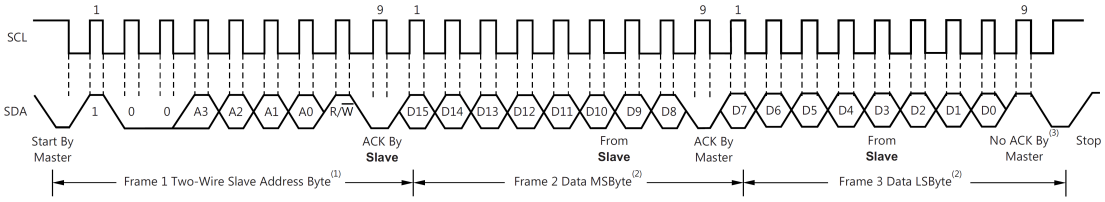


Figure 3.10. Timing diagram for read word format

Conversion time

It is possible to select many conversion times: by selecting longer conversion times the accuracy is higher. This is because it is possible to select an average mode that allows to mediate the value on measurement. It is immediate to understand that using an higher conversion rate it is possible to mediate the result across many values. For our purpose a good compromise can be 1.1ms, this value corresponds to a medium noise of $18\mu V$.

Resolution and calibration

The ADC has a resolution of 16 bit so that the LSB step size is $1.25mV$ for bus voltage beacuse the full scale range of the ADC is 40.96V although is not possible to apply more than 36V. Quantum is $2.5\mu V$ for the shunt voltage since the maximum range is $[-81.9175mV, 81.92mV]$.
For the bus voltage there are no further regulations and the MSB will always be 0 because negative voltages are not allowed.
For the shunt voltage the issue is a bit trickier because we need to consider the shunt resistor value.

First of all we can calculate the effective LSB as follows:

$$I_{LSB} = \frac{Maximum\ Expected\ Current}{2^{15}} = \frac{1600A}{2^{15}} = 48.83mA \tag{3.13}$$

So that we can estimate the calibration register value:

$$\begin{aligned} CAL = \frac{0.00512}{I_{LSB} \cdot R_{shunt}} = \frac{0.00512}{\frac{1600A}{2^{15}}\ 50\mu\Omega} = 2097.152 \\ \approx 2097 \rightarrow b\,0000\,1000\,0011\,0001 \end{aligned} \tag{3.14}$$

The Current register value is so calculated as:

$$I = \frac{ShuntVoltage \cdot CAL}{2048} \tag{3.15}$$

The calibration register can also be used to adapt the current scale to the full range scale (no need in this application because we are currently using the full range scale) and to calibrate the device after an accurate measurement done by using an external multimeter.
Actually this register is not used in BAT-MAN because the conversion from shunt voltage to current is done on SW side.

Alert function

The Alert pin has the capability to be programmed to respond both to a single user-defined event or to a Conversion Ready notification if desired.

In the first case, it is asserted when the value of current or voltage indicated in the right registers has been reached. This can be due to a maximum or minimum current or voltage value or to many other parameters. When a maximum or minimum value is exceeded the corresponding number is stored in a register so that it can be read from the master.

The same pin can be used as an interrupt notifying the end of the conversions. In order to understand the triggering event, it is necessary to interrogate the *Mask/Enable* register looking for an asserted flag.
The same register can also be read in polling mode when alert pin use is inhibited.

3.1.4 Shunt Resistor

The shunt resistor has to be selected according to its power rating and resistance value. The resistance must be such that the voltage drop across its terminals is negligible with respect to the voltage of the source.
The maximum current calculated from Table 2.1 is 985 A, approximated to 1000 A, and the voltage source is around 12V. According to the selected conditioning circuit, shunt voltage must be included in the range $-81.9175mV \div 81.92mV$.
Starting from this consideration the maximum resistance value has been calculated as:

$$R_{shunt} < \frac{V_{max}}{I_{max}} = \frac{81.92mV}{1000A} = 81.92\mu\Omega \tag{3.16}$$

By fixing this value, the required maximum power rating is:

$$P_{max} = R_{shunt}\, I_{max} = 81.92\mu\Omega \cdot (1000A)^2 \approx 80W \tag{3.17}$$

The closest available commercial value is $50\mu\Omega$ with a continuous 36W power rating.

WSBM8518

Figure 3.11. WSBM8518 Resistor with its Package and Molex Connector [23]

The main features of the device are as follows:

- Resistance of $50\mu\Omega \pm 5\%$ with four terminals
- Continuous power rating of 36 W
- Peak power rating of 180W for 5s
- Includes 4-pin male connector that mates with a Molex type MX150
- low TCR ($200ppm/°C$)

This shunt resistor has four terminals, in such a way the voltage across it can be measured as near as possible without the influence of the voltage drop across the copper bars which could be comparable with that of the leads of the resistor.
The component is built in a plastic package ($40.1 \times 35.1mm$)so that the whole PCB can be contained in it.
In order to provide the supply and communicate with the board, the MX150 connector can be exploited by clamping it with its corresponding sealed male connector.

With this specifications it is possible to calculate the actual maximum measurable current:

$$I_{MAX} = \sqrt{\frac{P_{RATING}}{R_{SHUNT}}} = \sqrt{\frac{36W}{50\mu\Omega}} \approx 848.5A \tag{3.18}$$

However - since the maximum current is absorbed for a limited amount of time - considering the peak power rating we can evaluate:

$$I_{MAX,5s} = \sqrt{\frac{P_{RATING,5s}}{R_{SHUNT}}} = \sqrt{\frac{180W}{50\mu\Omega}} \approx 1900A \tag{3.19}$$

Actually the upper bound is set by the INA226 which can stand at its input no more than 81.92mV, thus obtaining:

$$I_{MAX,5s} = \frac{V_{MAX,INA226}}{R_{SHUNT}} = \frac{81.92mV}{50\mu\Omega} \approx 1638A \tag{3.20}$$

3.1.5 Flash Memory

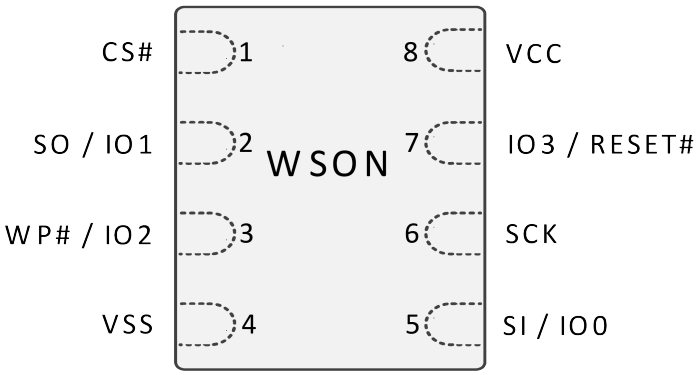


Figure 3.12. S25FL256L - Flash Memory Pinout [24]

A Flash memory has been added to meet the need of storing data also when the smartphone is not connected to the device.

The choice reflected a trade off between package size and array dimension. The main features are:

- AEC-Q100 Automotive Grade Qualified
- SPI protocol communication
- 256 Mbit density
- 65nm NOR Flash Memory
- Min Erase Sector Size 4KB
- Single byte programming
- Normal Read Speed 50 MHz

Digital Communication

SPI protocol allows to manage the Flash operating mode:

- **Erase Operation**
  Four erasing mode are available depending on the erasing size: sector, half block, block, chip; in all the cases a *Write Enable* (WREN) command must be sent in order to allow this procedures' execution.
  Chip Erase (CE) command sets all bit to 1 inside the entire flash memory array, therefore there is no need to specify the initial erasing address like in the other cases.

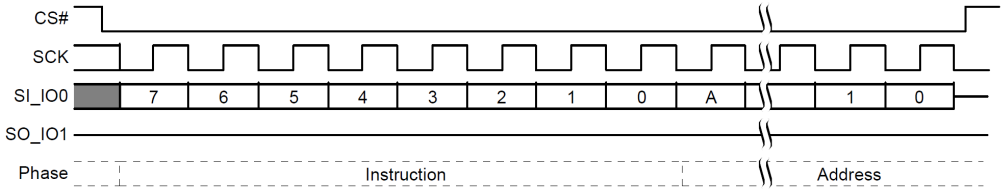


Figure 3.13. Erase Operation Command Sequence

The remaining three operation have the same behaviour; by specifying the proper command and an address, all the N following bytes are deleted, where N depends on the portion of array to be erased.

	N
Sector	4KB
Half-Block	32KB
Block	64KB
Chip	32MB

Table 3.3. Individually erasable portions of array.

- **Array Program Operation**
  This is the way used to write data in the NOR Flash array. To implement this feature the page program command - preceded by a WREN - must be used.
  The command sequence is composed of an 8-bit instruction, 32-bit address and the data stream as in Figure 3.14; the minimum program dimension is represented by a single byte.
  This operation is only able to program by deasserting logic 1s (1→0 but not 0→1.)

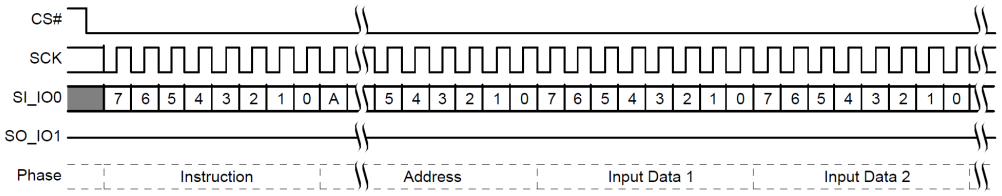


Figure 3.14. Array Program Operation Command Sequence

- **Array Read Operation**
  The memory content can be read by using the Read command; the address can start at any location of the array and it is automatically incremented to next higher one after each byte of data is shifted out. Both one byte or the entire array can be read of this instruction.

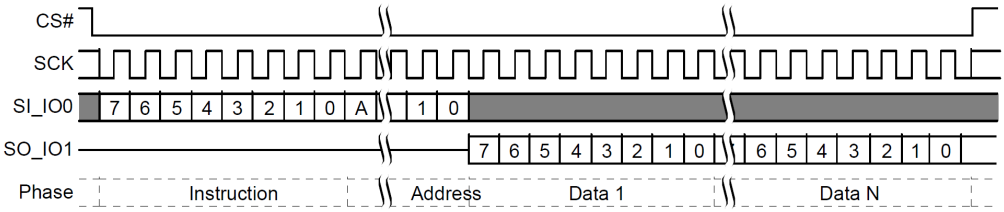


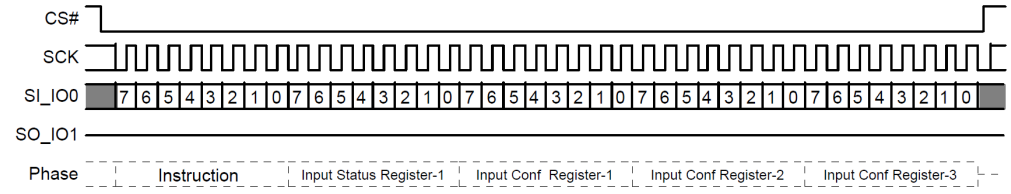
Figure 3.15. Array Read Operation Command Sequence

- **Register Read/Write**
  The flash memory provides numerous registers reporting configuration options and operation status. The access in both read and write modes are performed by using specific commands.

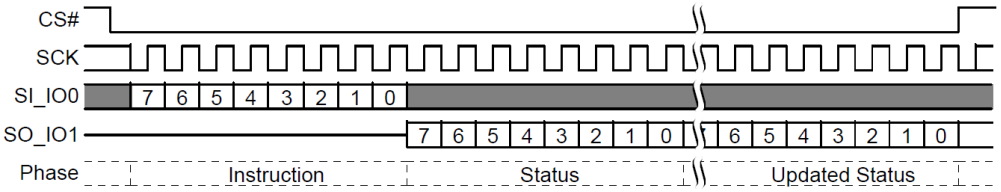
	Bits							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Status Register 1	SRP0_NV	SEC_NV	TBPROT_NV	BP_NV2	BP_NV1	BP_NV0	WEL_D	WIP_D
	Status Register Protect 0	Top or Bottom Relative Protection	Legacy Block Protection Volatile				Write Enable Latch	Write in Progress
Status Register 2	RFU	E_ERR	P_ERR	RFU	RFU	RFU	ES	PS
	Reserved	Erase Error Occurred	Programming Error Occurred	Reserved	Reserved	Reserved	Erase Suspend	Program Suspend
Config Register 1	SUS_D	CMP_NV	LB3	LB2	LB1	LB0	QUAD_NV	SRP1_D
	Suspend Status Default	Complement Protection Default	Security Region Lock Bits				Quad Default	Status Register Protect 1 Default
Config Register 2	SUS	CMP	LB3	LB2	LB1	LB0	QUAD	SRP1
	Suspend Status	Complement Protection	Volatile copy of Security RegionLock Bits				Quad I/O mode	Status register Protect 1
Config Register 3	IO3R_NV	OI_NV		RFU	QPI_NV	WPS_NV	ADP_NV	RFU
	IO3_Reset	Output Impedance		Reserved	QPI	Write Protect Selection	Address Length at Power-up	Reserved

Table 3.4. Flash Registers mapping

While every register can be read separately, the programming is accomplished by the *Write Register* (WRR) command; the only registers that can be written are Status-1 - for protection purpose - and Configuration-1,2,3 - allowing to change the behaviour of the memory according to the user needs. WRR command permits to set all the above register at a time (Table 3.4).



(a) Write Register command



(b) Status1 Read command

Figure 3.16. Register Write (a) and Read (b) sequences.

3.1.6 Temperature and Humidity Sensors

The need of measuring the temperature arose mainly for two reasons:

- Compensate the thermal drift of the components
- Measure hood's temperature

The first reason is due to the fact that in a measurement system - to obtain precise acquisitions - the effects due to thermal drift of some components can not be neglected. This is mainly true for the protection diode *D4* in Figure 3.5 since its forwarding bias voltage drop reduces the value of the measured supply voltage. This reduction is dependent on the temperature and, since in a car's hood the temperature's gradient is of dozens of degrees, it could not be negligible.

The second reason is due to the SoC and SoH algorithm which needs temperature as an input parameter.

In order to accomplish these tasks, the choice has been to use two different sensors. The first one will be directly soldered on the board and its thermal pad is on the bottom of the component. The second one will be connected on the case of the measurement system having the thermal pad facing to the hood.

The two components should also be able to measure relative humidity for future development purposes.

Board Temperature and Humidity Measurement - HDC2010

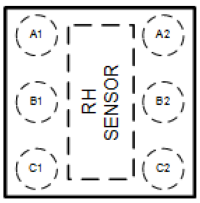


Figure 3.17. HDC2010 Pinout [29]

The main features are:

- Temperature accuracy $\pm 0.2°C$
- Humidity accuracy $\pm 2\%$
- I2C Interface
- Maximum resolution 14 bit
- Capacitive-based sensor
- Factory calibrated
- Heating element to burn away condensation and moisture for increased reliability
- Bottom sensing

I2C read operation is performed accessing the desired register or reading in burst mode the whole array. For further informations about I2C refer to subsubsection 4.2.1

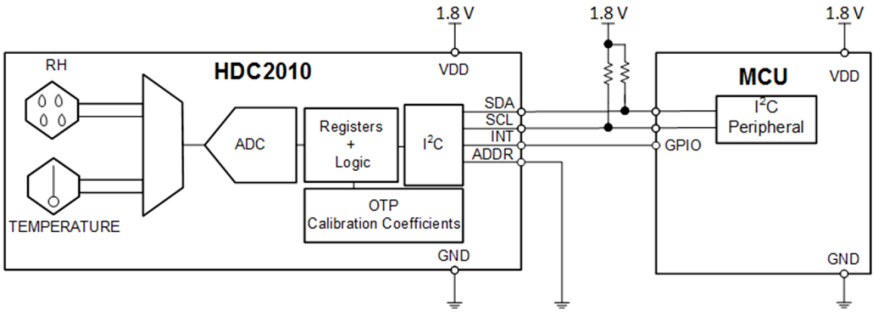


Figure 3.18. HDC2010 Typical Application [29]

Hood Temperature and Humidity Measurement - HDC1080

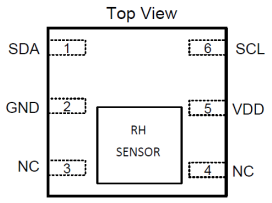


Figure 3.19. HDC1080 Pinout [30]

The main features are:

- Temperature accuracy $\pm 0.2°C$
- Humidity accuracy $\pm 2\%$
- I2C Interface
- Maximum resolution 14 bit
- Factory calibrated
- Top sensing

Temperature and humidity readings are performed by sending a proper command via I2C, not by directly accessing the registers.

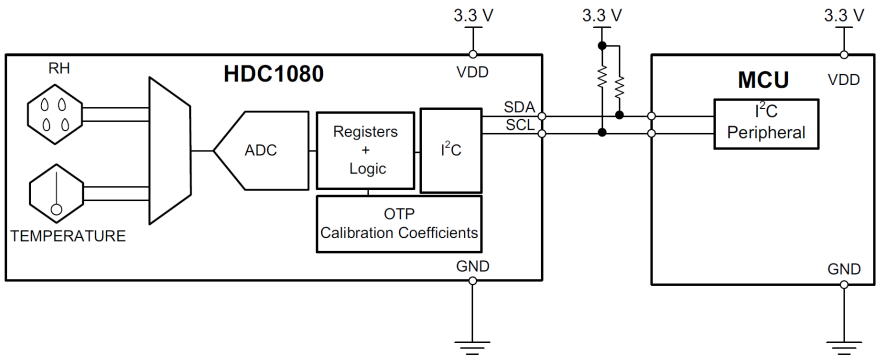


Figure 3.20. HDC1080 Typical Application [30]

3.1.7 CC2640R2F-Q1

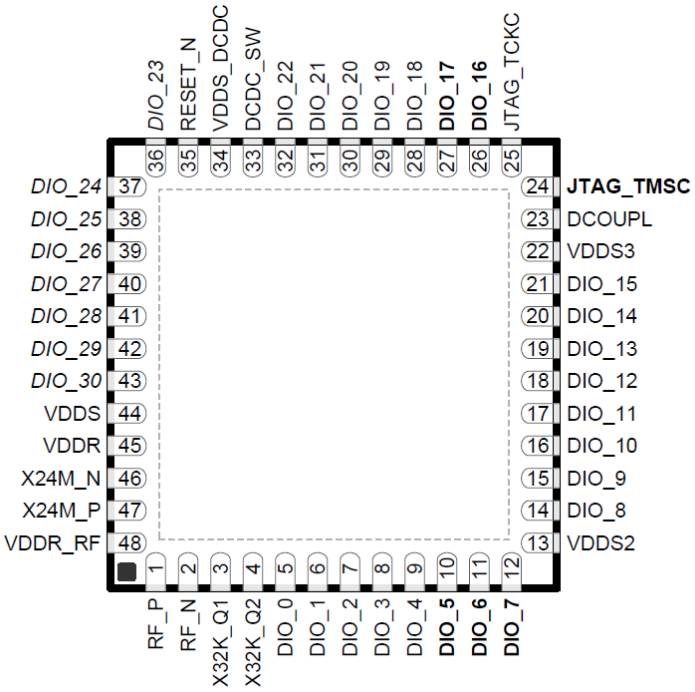


Figure 3.21. CC2640R2F-Q1 Pinout [25]

The main features are:

- Qualified for automotive applications
- ARM Cortex-M3 with 32 bit architecture
- 275kB nonvolatile memory and 8kB SRAM
- I^2C and SPI compatible
- Supply voltage range 1.8 to 3.8V
- 2.4GHz transceiver compatible with BLE 4.2 and 5.0
- Sensor Controller 16 bit Coprocessor

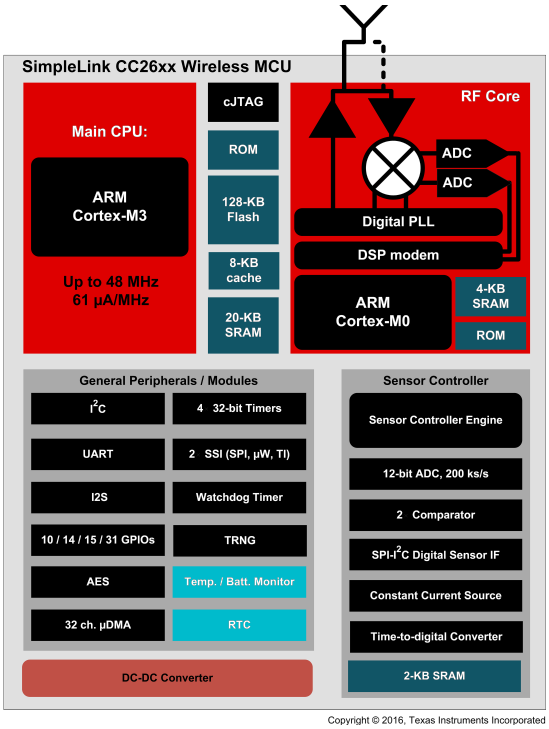


Figure 3.22. CC2640R2F Block Diagram

This microcontroller is the core of the device, its functions are:

- communicating with the INA226-Q1 in order to acquire current and voltage values through the I^2C BUS
- communicating with the HDC2010 and HDC1080 in order to acquire both temperature and humidity of the board and hood through the I^2C BUS
- storing and reading data from the S25FL256 flash memory through the SPI BUS
- evaluating SoC and SoH through specific algorithms
- controlling the proper working of the device signalling states by mean of LEDs
- sending the receiving data through BLE to a smartphone

One of the main features is the possibility to use BLE protocol by connecting an antenna; in order to do this it is necessary to match the impedance with one of the following front-ends:

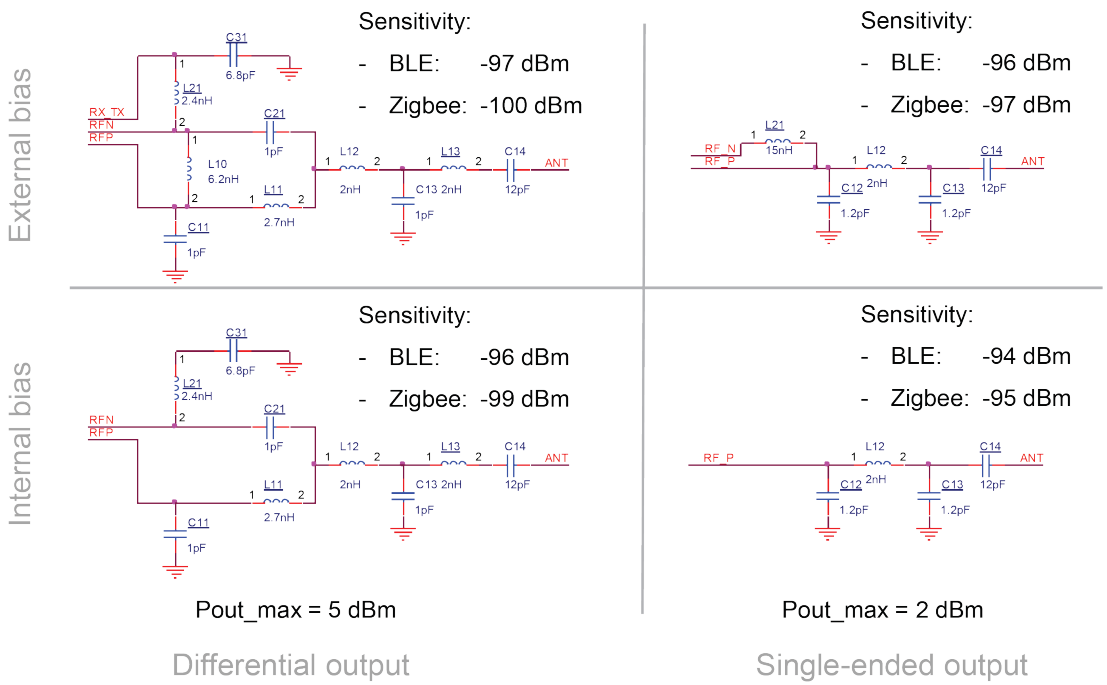


Figure 3.23. RF front ends

External bias with single-ended output front-end was chosen in order to achieve a good sensitivity without using too many components. This topic will be deepened in section 3.2.
The Front End needs to be connected to an antenna, which can be a directly designed on the PCB (patch) one or a built-in discrete component.

At the beginning a patch antenna solution has been considered, the DN007, whose design parameters are provided by the manufacturer of the μC (Texas Instruments) [26].
This solution implies to have a big PCB area available because it occupies $192mm^2$, this dimension is not suitable since the WSBM8518 case imposes strict constraints on PCB size.

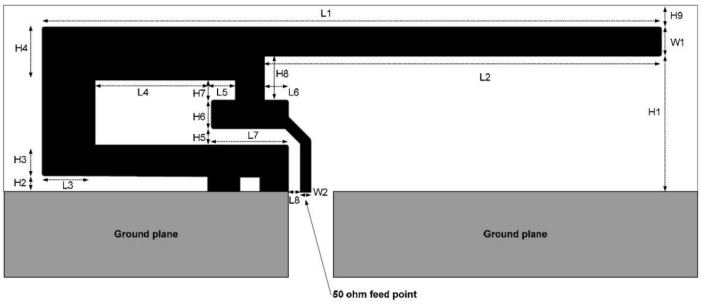


Figure 1. IFA Dimensions

Table 1. IFA Dimensions

H1	5.70 mm	W2	0.46 mm
H2	0.74 mm	L1	25.58 mm
H3	1.29 mm	L2	16.40 mm
H4	2.21 mm	L3	2.18 mm
H5	0.66 mm	L4	4.80 mm
H6	1.21 mm	L5	1.00 mm
H7	0.80 mm	L6	1.00 mm
H8	1.80 mm	L7	3.20 mm
H9	0.61 mm	L8	0.45 mm
W1	1.21 mm		

Figure 3.24. Design Parameters for the DN007 Patch Antenna

For this reason the choice fell on a chip antenna characterized by a small package. The selected commercial device is the 2450AT18A100, manufactured by Johanson Technology.

Terminal Configuration

No.	Function
1	FEED
2	NC

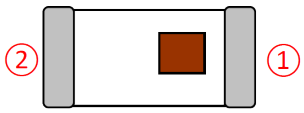


Figure 3.25. RF front ends

3.2 Antenna Matching

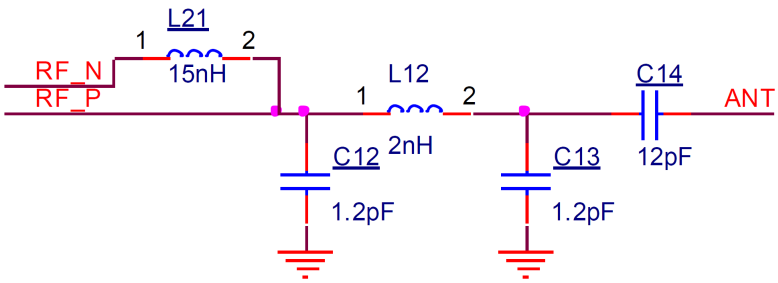


Figure 3.26. Single Ended External Biased matching circuit

The matching circuit in Figure 3.26 can be studied as three separate parts:

- **Balun**: a network that transforms from a balanced (differential) to an unbalanced (single-ended) signal. The balun has a $\pm 90°$ phase shift implemented by using a low pass filter and a high pass filter. In this case it is sufficient to use one inductor connected in parallel to the differential pins of the μC.
  The important part is to keep the balun as symmetrical as possible, since an unbalance causes higher harmonic level, especially at the 2nd and 4th harmonic. Furthermore, it results in a reduced output power at the single ended side of the balun.

- **LC Filter** should be laid out so that Xtalk between the shunt components is minimized.
  Figure 3.27 shows three different layouts from the worst to the best. The advantage with the layout on the right is that the parasitic inductance in the PCB track between the shunt capacitor and the series inductor is in series with this last.
  LC network also performs a first rough matching.

- **Matching Capacitor**: is an optional component needed to fine-tune the matching, depending on the antenna characteristic impedance.

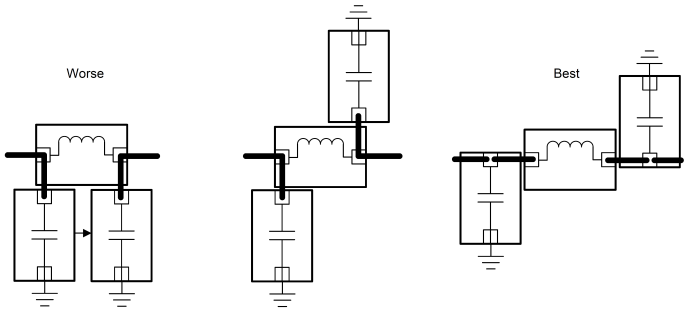
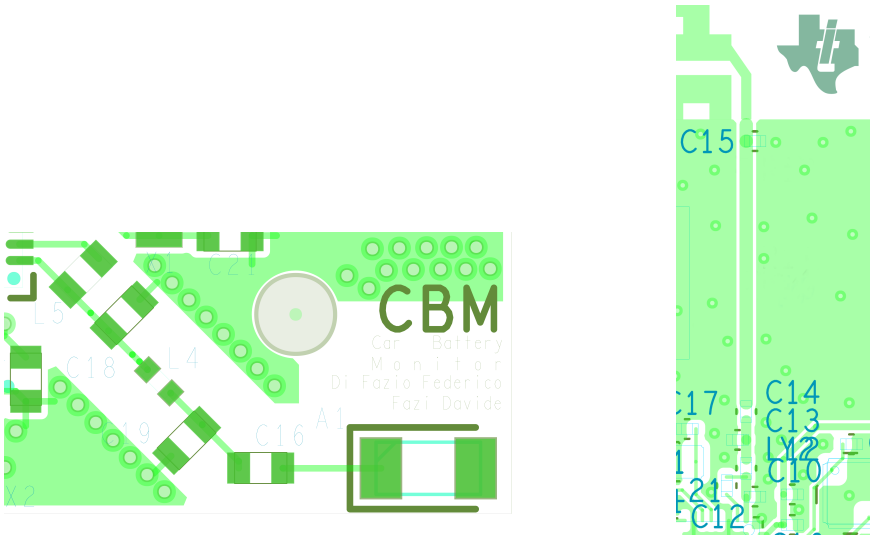


Figure 3.27. Different layouts of LC filters in π configuration

Unfortunately, as shown in Figure 3.28, due to the limited space available for the layout of the PCB, the Texas Instruments' layout guidelines could not accomplish. Therefore, a tuning on the component value could be required to obtain the desired performance.
In order to do this, a possible solution is to use Advanced Design System (ADS) to simulate and then compare the impedances and S-parameters of the custom design with the reference one.

It is important to remark that no traces should be routed on the part of the bottom layer beneath the matching circuit and the antenna.



(a) Used layout

(b) TI reference layout

Figure 3.28. Magnification of the layouts.

3.2.1 Single-Ended Internal Bias matching circuit

For sake of simplicity, we decided to start the analysis with the internal biased circuit (no balun needed).

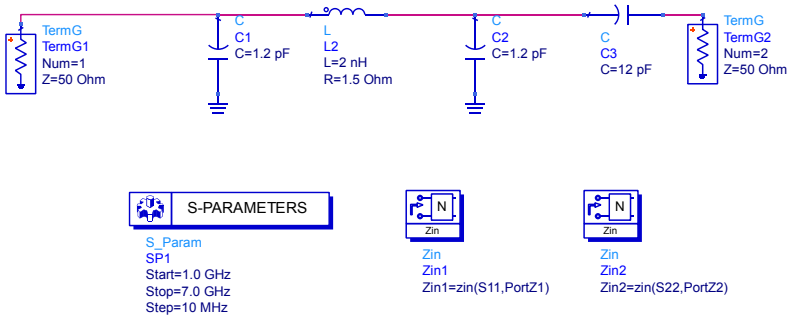


Figure 3.29. Single Ended Internal Bias matching network

To analyse the behaviour of the circuit a 50Ω termination on the output was put to emulate the antenna characteristic impedance.
The characteristic impedance of the RF pin of the μC is not provided by the manufacturer, therefore it was assumed to be equal to 50Ω too.

In this configuration it was possible to simulate the behaviour of the line by plotting its S-parameters varying the frequency from $1GHz$ to $7GHz$ with a step of $10MHz$, obtaining the graph in Figure 3.30.

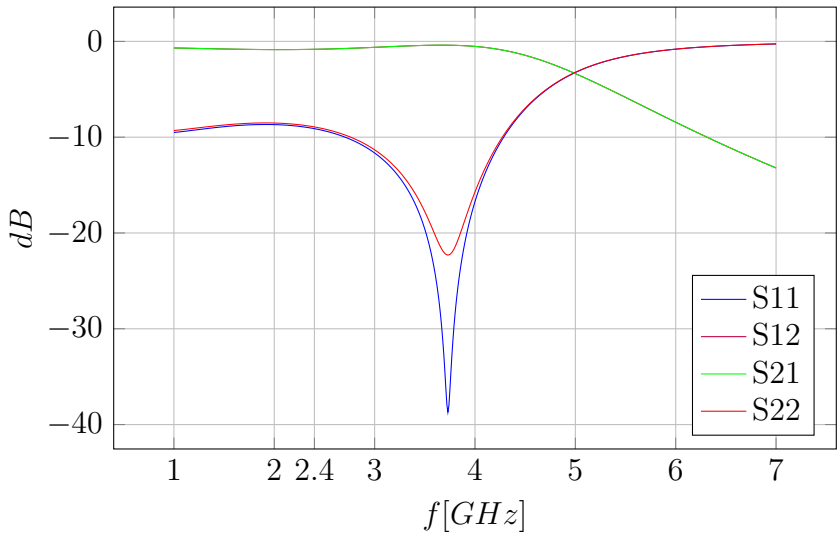


Figure 3.30. S-parameter with 50Ω input impedance

The two parameters S11 and S22 were expected to have an absolute minimum at the resonant frequency of the antenna (2.4GHz).
As it is evident from the graph, this is not confirmed by the simulation which presents an absolute minimum @ $\sim 3.8GHz$. This is due to an erroneous matching on the input.

Texas Instruments Customer Care At first, to recover this error, Texas Instruments was contacted, asking for the actual characteristic impedance value of both the RF pins, since this value is not reported at all in the datasheet.
Their answer was:

> *We only provide the information found in the document you are referring to: "Output impedance is normally used to design complex conjugate impedance matching between amplifier and load. For linear amplifiers this is sufficient to secure optimum power transfer. This method is thus not valid for the CC13xx/CC26xx series."*
> *[...]Normally a different length of the traces in your layout vs the reference design have limited impact. If you want to be sure, you should simulate the reference design layout with component models and compare with your layout. Then the impedance of the terminals have no impact since you are comparing.*

Proper terminations simulation Considering this, two more simulation parameters have been set (Zin1 and Zin2 in Figure 3.29) in order to find the proper value of the termination to be used. The result is the following chart.

Frequency [GHz]	Zin [Ω]	Zout [Ω]
1.000	30.156 - j19.156	60.873 - j23.492
1.500	26.508 - j15.209	53.511 - j15.524
2.000	24.587 - j11.118	49.633 - j7.266
2.400	24.769 - j7.519	50.000 + j3.232E-4
2.500	25.112 - j6.579	50.691 + j1.897
2.600	25.595 - j5.628	51.668 + j3.817
3.000	29.333 - j1.838	59.214 + j11.467
3.500	40.572 + j1.000	81.901 + j17.196
4.000	61.709 - j11.434	124.569 - j7.903
4.500	54.426 - j51.405	109.866 - j88.590
5.000	21.266 - j54.721	42.930 - j95.285
5.500	7.962 - j43.464	16.073 - j72.560
6.000	3.492 - j35.115	7.050 - j55.708
6.500	1.750 - j29.535	3.533 - j44.443
7.000	0.967 - j25.629	1.953 - j36.559

Table 3.5. Impedances for Single Ended Internal Bias matching network

The row of interest in the table is the one referred to 2.4GHz: as expected Zout is approximately 50Ω but, the most important result is Zin value equal to $24.769 - j7.519$. The latter can be used to obtain the input termination value by considering conjugate matching.
In Figure 3.31 the updated schematic is reported.

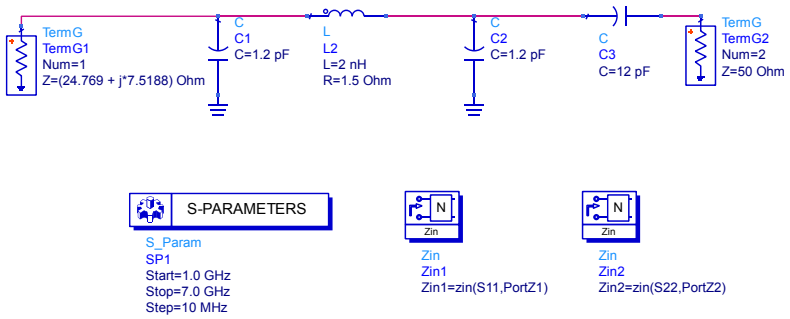


Figure 3.31. Single Ended Internal Bias matching network with matched impedance

As expected, since both the input and the output are matched, S11 and S22 have an absolute minimum value for 2.4GHz. This means that no reflections are present neither at the input nor at the output port.

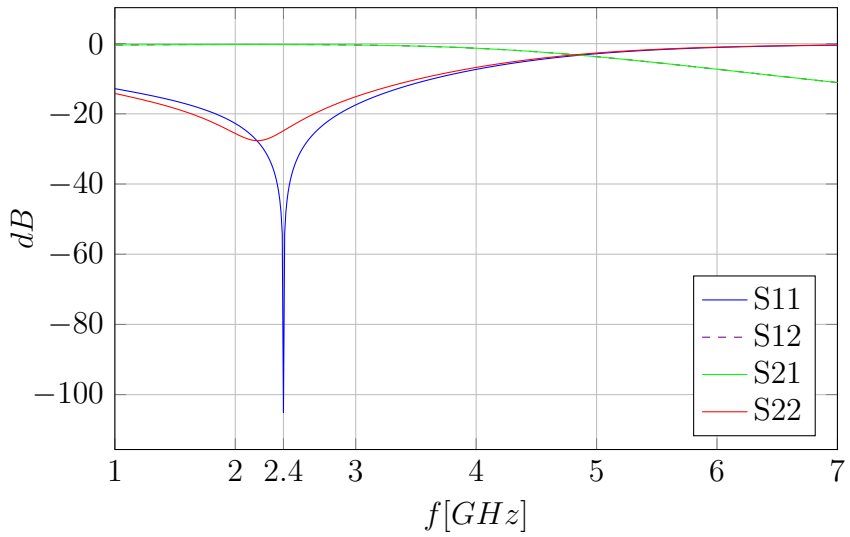


Figure 3.32. S-parameter with $24.769 + j7.519$ output impedance of the RF uC pins

3.2.2 Single-Ended External Bias matching circuit

After the previous analysis, it is now possible to evaluate the behaviour of the real implemented circuit and compare it with the reference one, keeping into consideration the layout.

Ideal case Firstly, the ideal case was simulated to have a rough idea of the external biased circuit behaviour.

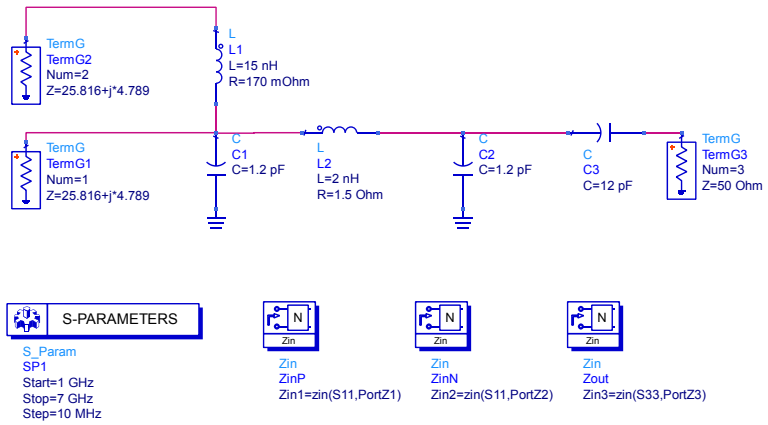


Figure 3.33. Single Ended External Bias matching network with matched impedance

The starting point is to evaluate, as done before, the input and output impedances reported in Table 3.6.
There is a slight variation in the output impedance which, however, does not affect considerably the performance. Instead, the input impedances' values, which are now two, vary; it was expectable as a balun (an inductor) was inserted to have an external bias that improves the sensitivity of 1dB.
In Figure 3.33, a 3-port network can be recognized but, actually, port two has only the function of inserting an external bias. In fact, by calculating its impedance referring to S22 parameter @2.4GHz and by using its complex conjugate as the termination value, the resulting S parameters is totally erroneous as shown in Figure 3.34.

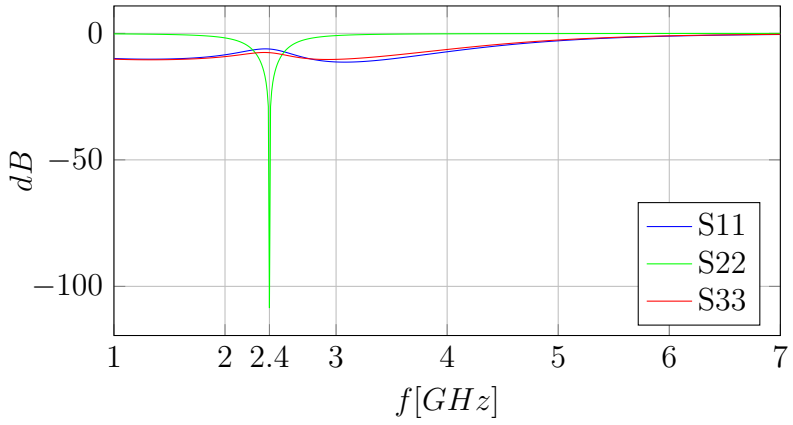


Figure 3.34. S-parameters with wrong impedance at port 2

In the above graph both the reflection coefficients S11 and S33 have not a minimum at the working frequency, this means that the whole energy, both in transmission and in reception, is reflected. Only the reflection coefficient on *PortZ2* has a minimum @ 2.4GHz, which is not useful at all, because this pin is not transmitting anything.
The impedance in question is $13.564 + j225.425\Omega$: a so high positive imaginary part is for sure due to the inductor used as a balun.
In light of this, it can be interesting to observe the value of S22 parameter at low frequencies: since the balun is a bypass inductor, the reflection coefficient should have an high pass trend. This theory is confirmed by the following graph, obtained by using the right values of impedance at all the ports.
At low frequencies, S22 and S11 have the same behaviour: port 1 and 2 are connected by an inductor which is a short circuit at low frequencies.

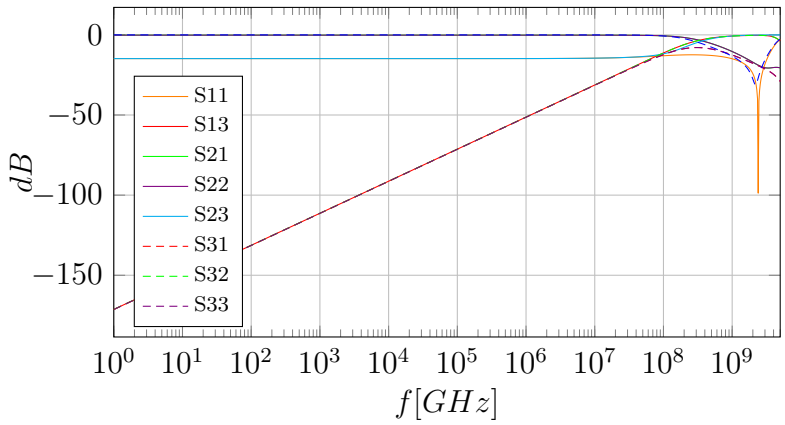


Figure 3.35. S-parameter over all the frequencies

In the following chart the characteristic impedance of the ports are reported; for port 2 both the right and wrong values are reported.

Frequency [GHz]	ZinP [Ω]	ZinN, wrong [Ω]	ZinN, good [Ω]	Zout [Ω]
1.00	36.642 - j7.812	16.273 + j92.132	36.642 - j7.812	32.328 + j0.265
1.50	30.569 - j9.983	15.070 + j139.267	30.569 - j9.983	39.743 + j1.708
2.00	26.693 - j7.894	14.030 + j186.884	26.693 - j7.894	46.110 - j0.660
2.30	25.837 - j5.646	13.638 + j215.761	25.837 - j5.646	49.890 - j3.412
2.40	25.816 - j4.788	13.570 + j225.432	25.816 - j4.788	51.172 - j4.597
2.50	25.936 - j3.888	13.542 + j235.123	25.936 - j3.888	52.462 - j5.945
3.00	29.137 + j1.123	14.116 + j283.693	29.137 + j1.123	58.386 - j15.985
3.50	39.405 + j5.669	16.048 + j331.811	39.405 + j5.669	58.920 - j32.655
4.00	63.091 - j1.178	18.897 + j378.433	63.091 - j1.178	46.657 - j49.021
4.50	68.431 - j47.575	21.343 + j423.135	68.431 - j47.575	27.922 - j53.051
5.00	27.390 - j60.067	22.196 + j466.748	27.390 - j60.067	14.555 - j47.787
5.50	9.666 - j47.149	21.394 + j510.552	9.666 - j47.149	7.526 - j40.859
6.00	4.075 - j37.365	19.623 + j555.229	4.075 - j37.365	4.061 - j34.998
6.50	1.995 - j31.002	17.540 + j600.823	1.995 - j31.002	2.311 - j30.456
7.00	1.088 - j26.651	15.513 + j647.120	1.088 - j26.651	1.383 - j26.956

Table 3.6. Impedances for Single Ended External Bias matching network

For the reasons explained above, it is not useful to show S parameters relative to port 2, so they will not appear in the following pages.
The obtained graph is somewhat similar to the one obtained for the Internal Bias in Figure 3.31.

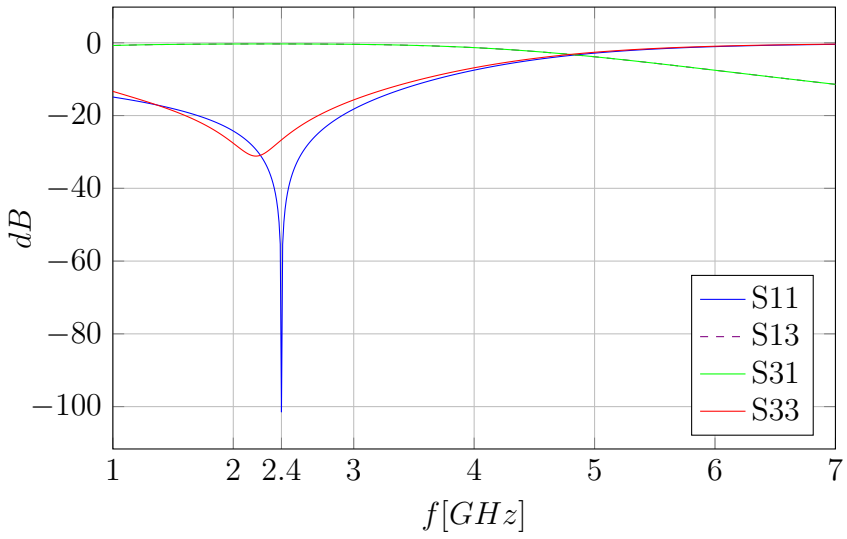
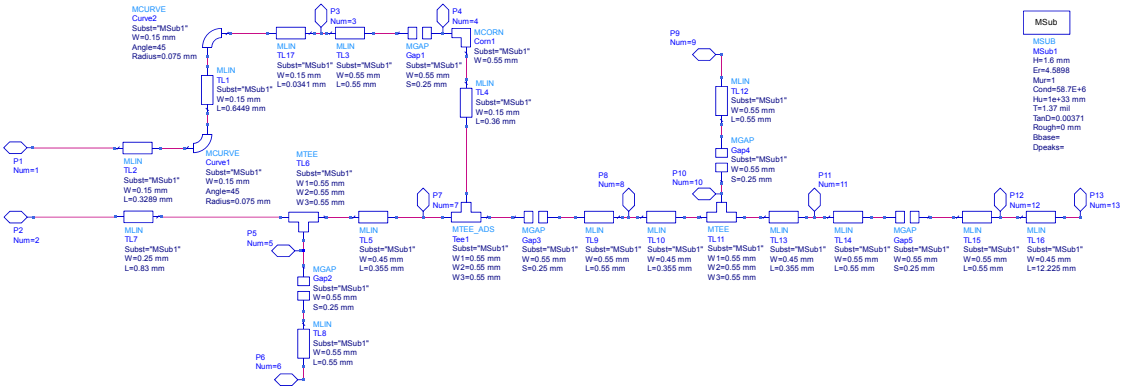


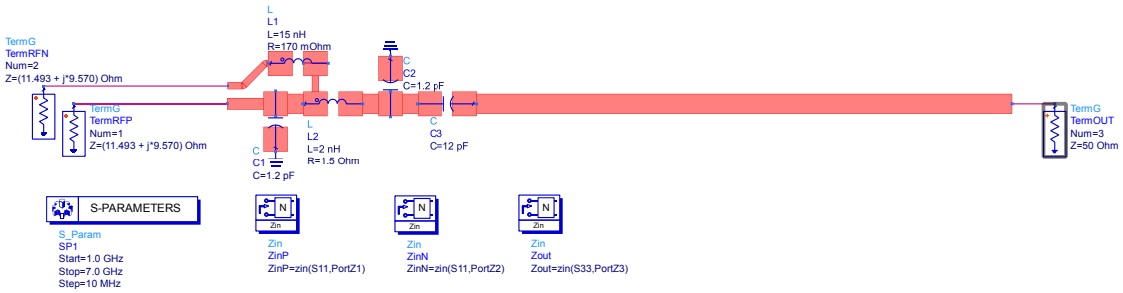
Figure 3.36. S-parameter with $25.816 + j4.788$ input impedance at both port 1 and 2

Texas Instruments Reference Layout The successive step is to associate a layout to the schematic, so to extrapolate the real behaviour of the matching circuit on a PCB.
First thing was to redraw in ADS the layout given by Texas Instruments in order to have a reference sample to be compared with BAT-MAN design.



(a) Schematic View

(b) Layout View



(c) Mixed View

Figure 3.37. Reference Layout ADS microstrip design

Three different ways to do this:

- **Traces On Layout → Schematic**: Importing the layout designed in OrCAD to *ADS Layout Editor* resulted in a draw made of *traces* connected in the right shape.
  After defining the substrate and the ports where the real components have to be connected, the design has to be converted in a *component* containing the EM parameters. Then the *schematic* relative to the component has been *generate*d.
  Unfortunately this method **did not work**: in the generated schematic there was just a bunch of unconnected ports. Therefore components were added to the layout by using *C_pad* and *L_pad* and setting the proper values. This resulted in having just a bunch of unconnected ports and components.
- **Microstrip On Layout → Schematic**:As a first try the traces were converted into microstrips but, once again, it did not work. Therefore, the microstrips were drawn from scratch in ADS measuring the OrCAD design and added the ports. The generated schematic was garbled and it was **impossible to work with it**.
- **Microstrip On Schematic → Layout**: Finally, **it worked**! By inverting the workflow, drawing the microstrips on the schematic as in Figure 3.37a and exporting the layout, it was possible to obtain a layout consistent with the schematic. After inserting the ports, the component (creating also the EM view) were generated. It is important to notice that the schematic has to be reimported from the layout to synchronize the placement of the ports.

These steps allowed to obtain the objects represented in Figure 3.37a and in Figure 3.37b. It is important to notice that the pads of the components (in SMD package 0603 or 0402, depending on the components) have been represented as microstrips too.
In order to make the microstrip work, it is necessary to define the substrate where it is laid. In this case it is a 1 oz copper layer over an FR4 insulator, so the corresponding parameters as in the table below were inserted:

Substrate Thickness	H	1.6 mm
Relative Dielectric Constant	ϵ_r	4.5898
Relative Permeability	μ_r	1
Conductor Conductivity	k	58.7e6 S/m
Cover height	Hu	inf
Conductor Thickness	T	1.37 mil (1oz)
Dielectric Loss Tangent	TanD	0.00371
Frequency at which ϵ_r and TanD are specified	FreqForEpsrTanD	2.52GHz

Table 3.7. Substarte Parameters

Finally, a new schematic was created, importing the layout as a component and connecting all the discrete components and the terminations as in Figure 3.37c. With the steps followed in subsection 3.2.1 the following results were obtained.

Frequency [GHz]	ZinP [Ω]	ZinN [Ω]	Zout [Ω]
1.00	68.915 + j13.844	68.915 + j13.844	31.223 + j79.002
1.50	51.230 - j39.025	51.230 - j39.025	141.892 + j131.653
2.00	18.137 - j24.504	18.137 - j24.504	124.159 - j35.353
2.40	11.493 - j9.570	11.493 - j9.570	54.709 + j7.833
2.50	10.828 - j6.119	10.828 - j6.119	48.268 + j21.308
2.60	10.413 - j2.696	10.413 - j2.696	43.874 + j34.818
3.00	10.940 + j11.854	10.940 + j11.854	39.264 + j93.215
3.50	19.883 + j39.159	19.883 + j39.159	63.765 + j208.741
4.00	109.340 + j100.199	109.340 + j100.199	355.529 + j568.133
4.50	59.301 - j115.409	59.301 - j115.409	296.578 - j597.477
5.00	10.267 - j55.171	10.267 - j55.171	45.142 - j241.361
5.50	4.301 - j33.370	4.301 - j33.370	20.727 - j135.576
6.00	2.722 - j21.898	2.722 - j21.898	14.804 - j82.738
6.50	2.242 - j14.420	2.242 - j14.420	12.274 - j48.807
7.00	2.104 - j9.069	2.104 - j9.069	8.894 - j21.709

Table 3.8. Impedances for Texas Instruments layout matching network

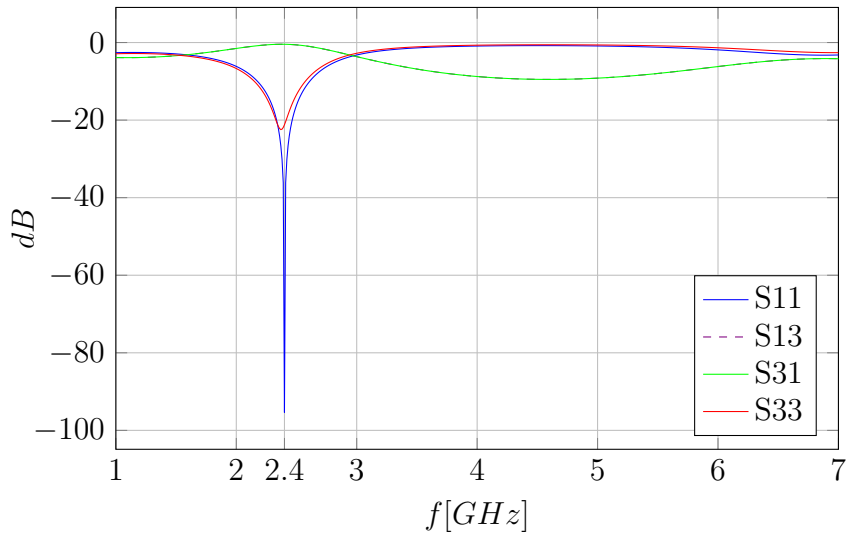
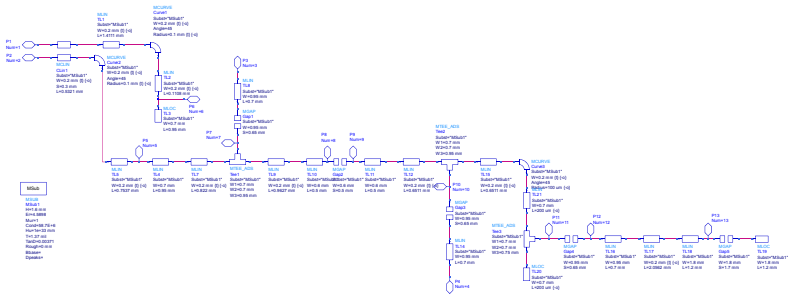


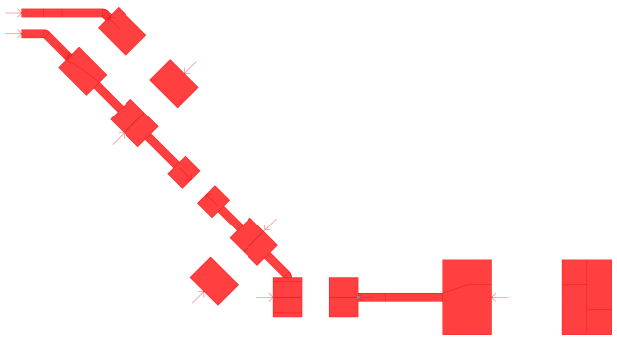
Figure 3.38. S-parameters for Texas Instruments layout

The extracted impedances, from Table 3.8, reveal that the output is far different from the ideal 50Ω. Despite this, the behaviour in Figure 3.38 is quite similar to the ideal one; it is even better for what concerns S33 which has a minimum closer to the frequency of interest (cf. Figure 3.26).
The RF pins' impedances in Table 3.8 are much different from those of the ideal case (cf. Table 3.6), in fact the layout, modelled as a microstrip line, influences the matching network characteristic impedances. Therefore it can be assumed the complex conjugate of these values as the one to be used at the terminations hereafter.

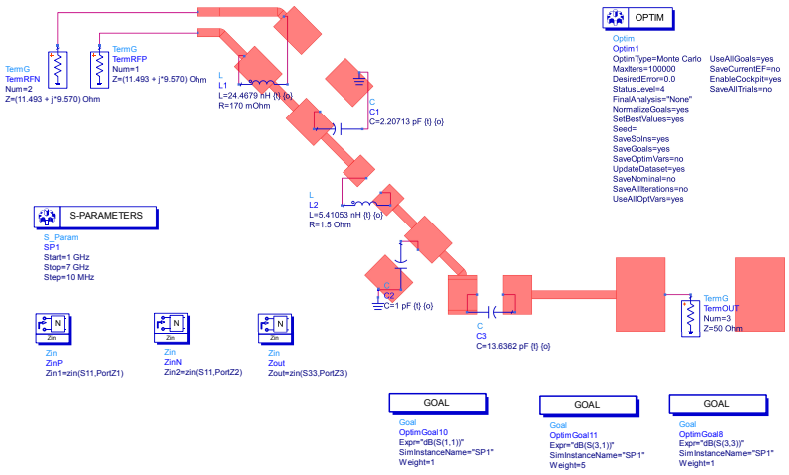
Actual PCB Layout Finally, the previous procedure was repeated for the design obtaining the Figure 3.39. The values extrapolated were used as terminations from the analysis of the Texas Layout assuming they are the closest values to the real ones as suggested by Texas Instruments (cf. paragraph 3.2.1) .
The first was performed keeping the discrete components values used up to now.



(a) Schematic View



(b) Layout View



(c) Mixed View

Figure 3.39. Actual PCB Layout ADS microstrip design

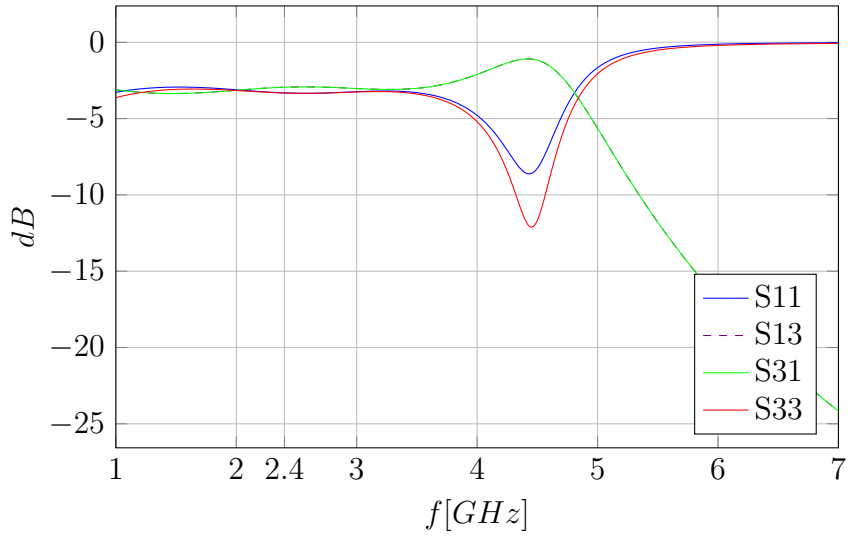


Figure 3.40. S-parameters for actual layout with original discrete components values

The graph reports a completely wrong behaviour, since the minimum of S11 and S33 are at a higher frequency and are too high.
Therefore the need to tune the components' values arises. In order to do this in an automated way, the choice was to use the *Optimization Cockpit* tool leaning on the *Goal* components. In fact, by setting the proper goals and by fixing a variation range for the capacitances and inductances, it is possible to obtain the values that better approximate the reference curves in Figure 3.38.
Obviously, no goals were set about the S parameters regarding the second port since it is not relevant for this study case.

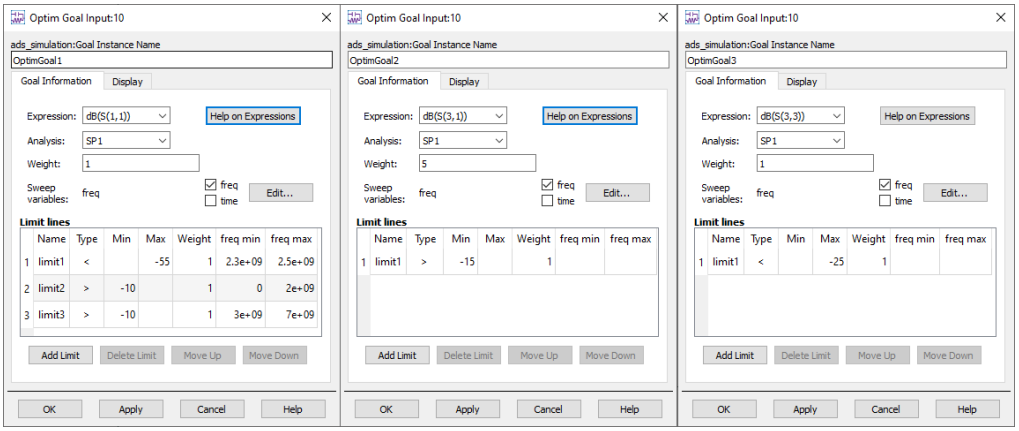


Figure 3.41. Goals parameters settings

In the Optimization Cockpit it is possible to: follow the variation of the curves (solid) with respect to the initial shape (dashed) and that of the components, choose the optimization algorithm and manually tune the components.
At first a random algorithm was used, but it revealed to be inefficient in terms of time and results. Therefore, after trying the whole possibilities, Montecarlo method appeared to be the best one.

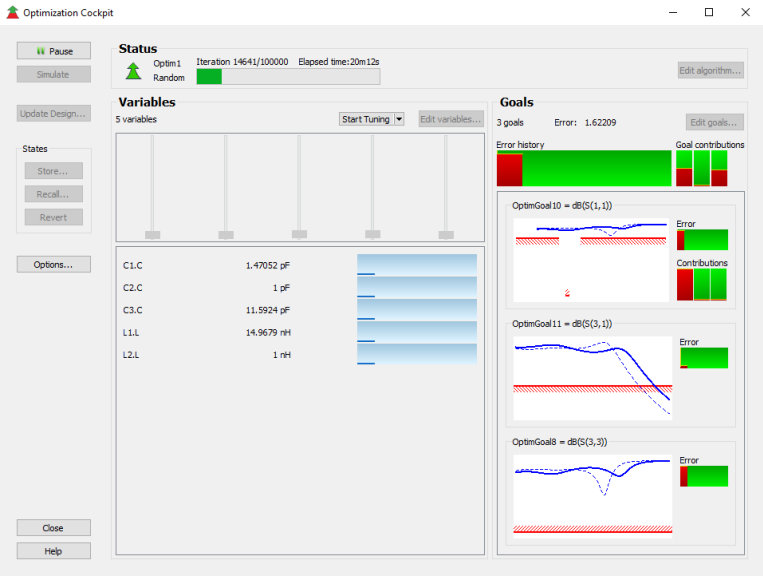


Figure 3.42. Optimization tool at work

The optimization gave as result the scattering parameters in Figure 3.43.
The shape is closer to the original one than before; however, unfortunately, the goals were not fully satisfied: although the minimum of S11 is at 2.4GHz, it is much higher than it should be, on the other hand S33 is lower than before and also below S11.

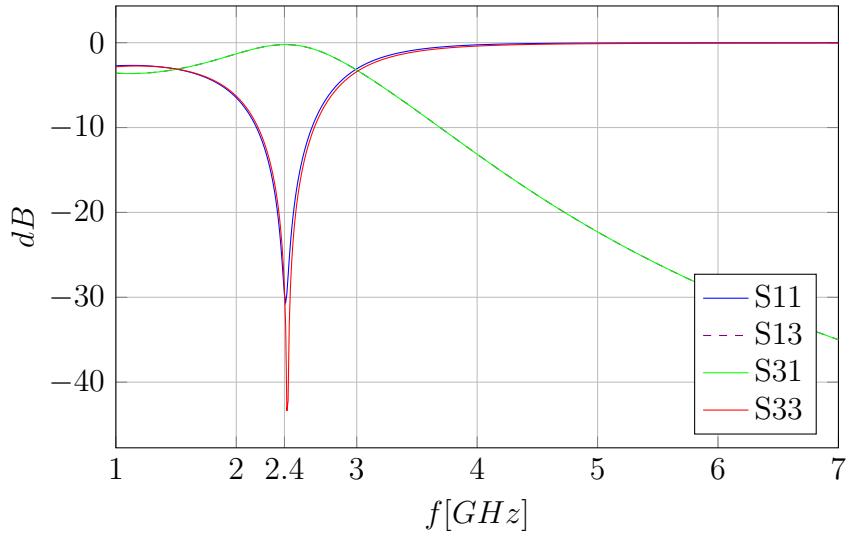


Figure 3.43. S-parameters for actual layout with optimized components values

Microstrips Tuning The *Optimization Variable Setup* allows to set the microstrip width too as a tunable element by the optimization cockpit. This method could be useful in order to avoid changing the discrete component values, by the way it seemed not so useful in this case as it needs to change too many variables, thus resulting in a behaviour not so different from the starting one.

3.2.3 Final Considerations

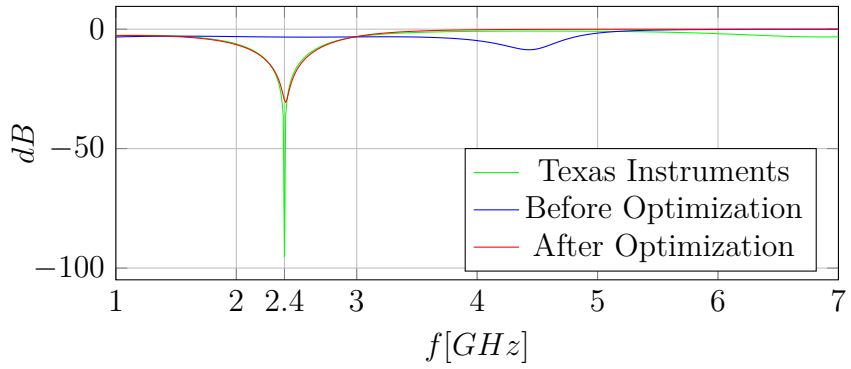


Figure 3.44. S11 Parameters for the 3 Different Implementations

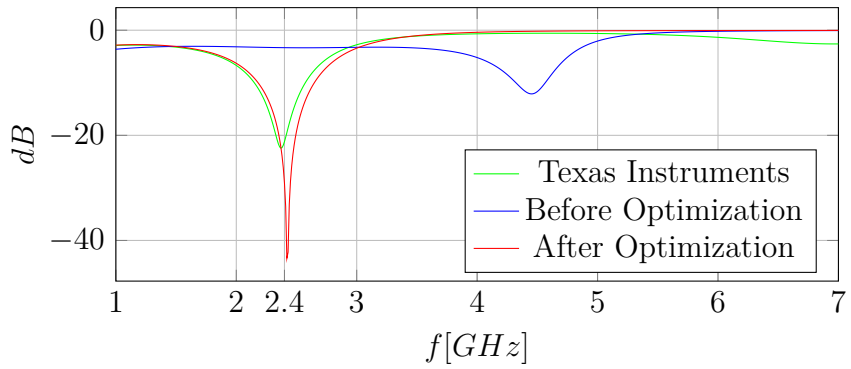


Figure 3.45. S33 Parameters for the 3 Different Implementations

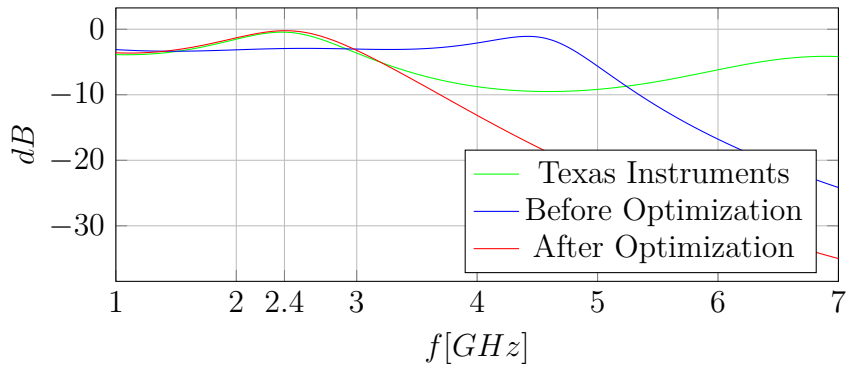


Figure 3.46. S13 (S31) Parameters for the 3 Different Implementations

In Figure 3.44, 3.45 and 3.46, a comparison between all the analysed implementations is shown to put in evidence each characteristic's trend in the three designs.
To sum up, a good layout is really important to get a properly matched antenna. When it is not possible to follow the reference layout, as in this case, it is appropriate to use a specific tool to find the right values to be used.
The following is a table showing the differences between the values suggested by TI and those obtained with the optimization considering our layout. These values don't correspond to commercial ones, the nearest are reported in the table too.

	Original Values	Optimized Values	Commercial values
C_1 [pF]	1.2	2.20713	2.2
C_2 [pF]	1.2	1.0000	1
C_3 [pF]	12	13.6362	14
L_1 [nH]	15	24.4679	24
L_2 [nH]	2	5.41053	5

Table 3.9. Discrete Components Values

Fine tuning By using commercial values, the S-parameters slightly changed, this causes a little performance loss at the frequency of interest.
To overcome this defect, it is now possible to do a fine tuning by optimizing the microstrips' width (cf. paragraph 3.2.2) because the target is not to change the frequency of the minimum but to make the curve steeper in that neighbourhood.
By doing this S11 gets even better than before, having its minimum equal to $\sim -40dB$; S33 gets a bit worse, but it is not so relevant since it was already better than the target.

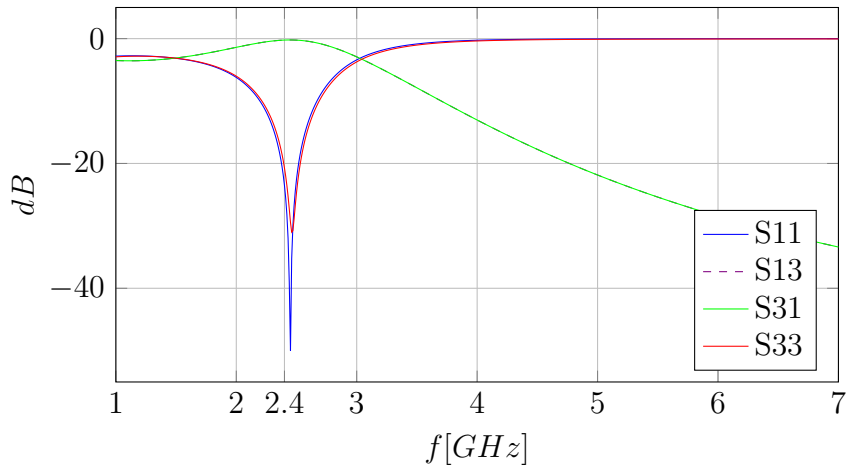
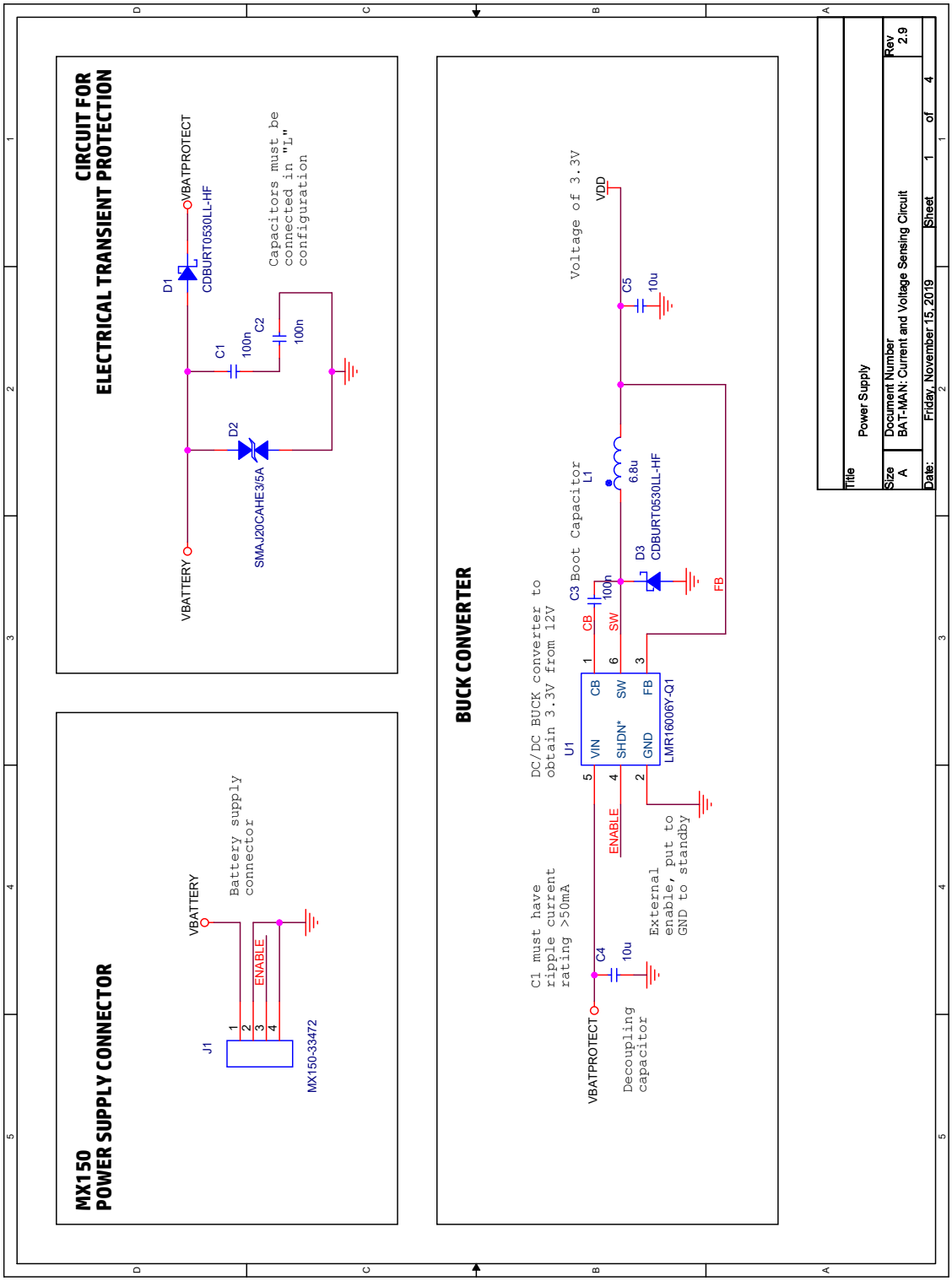
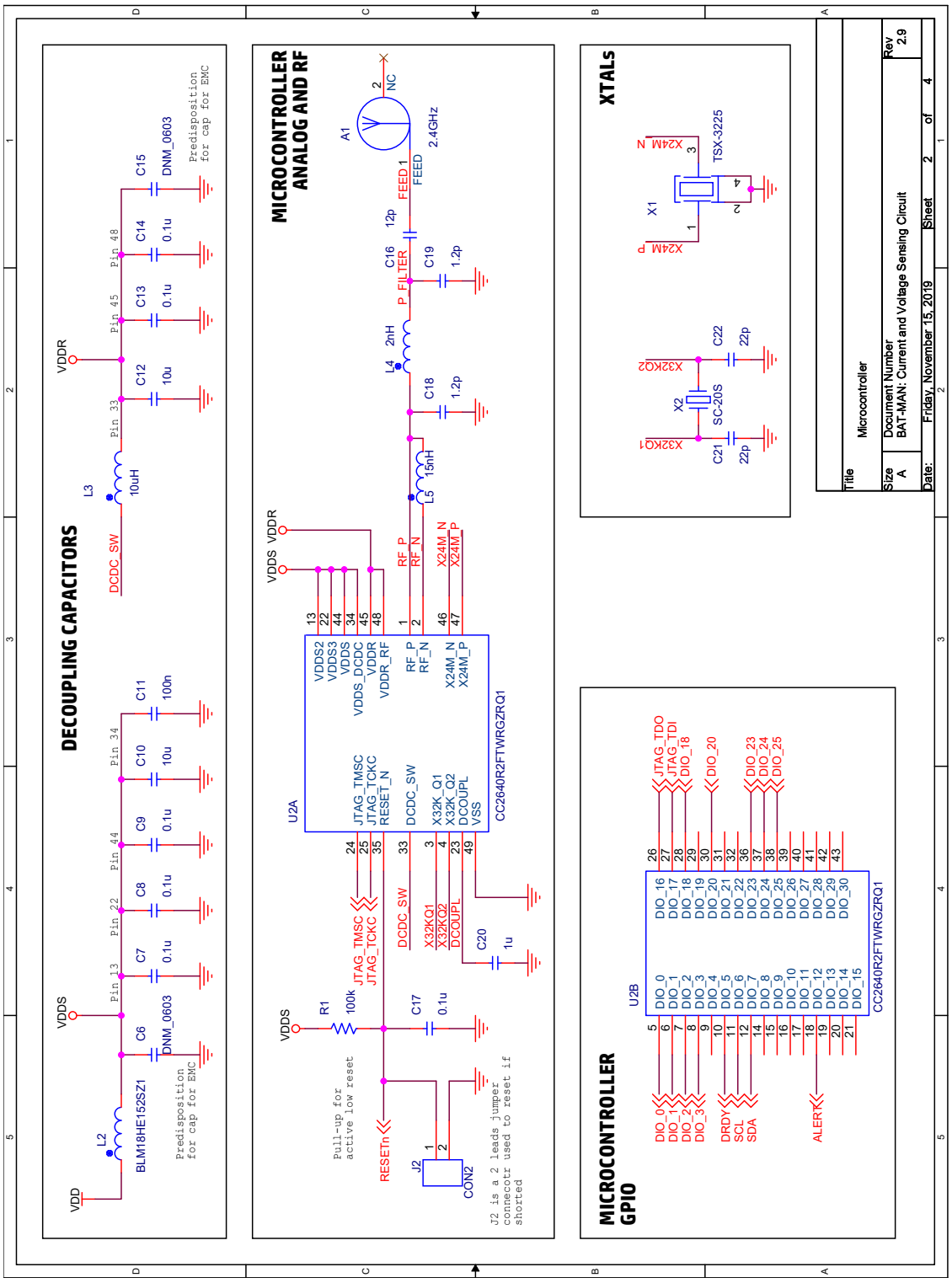
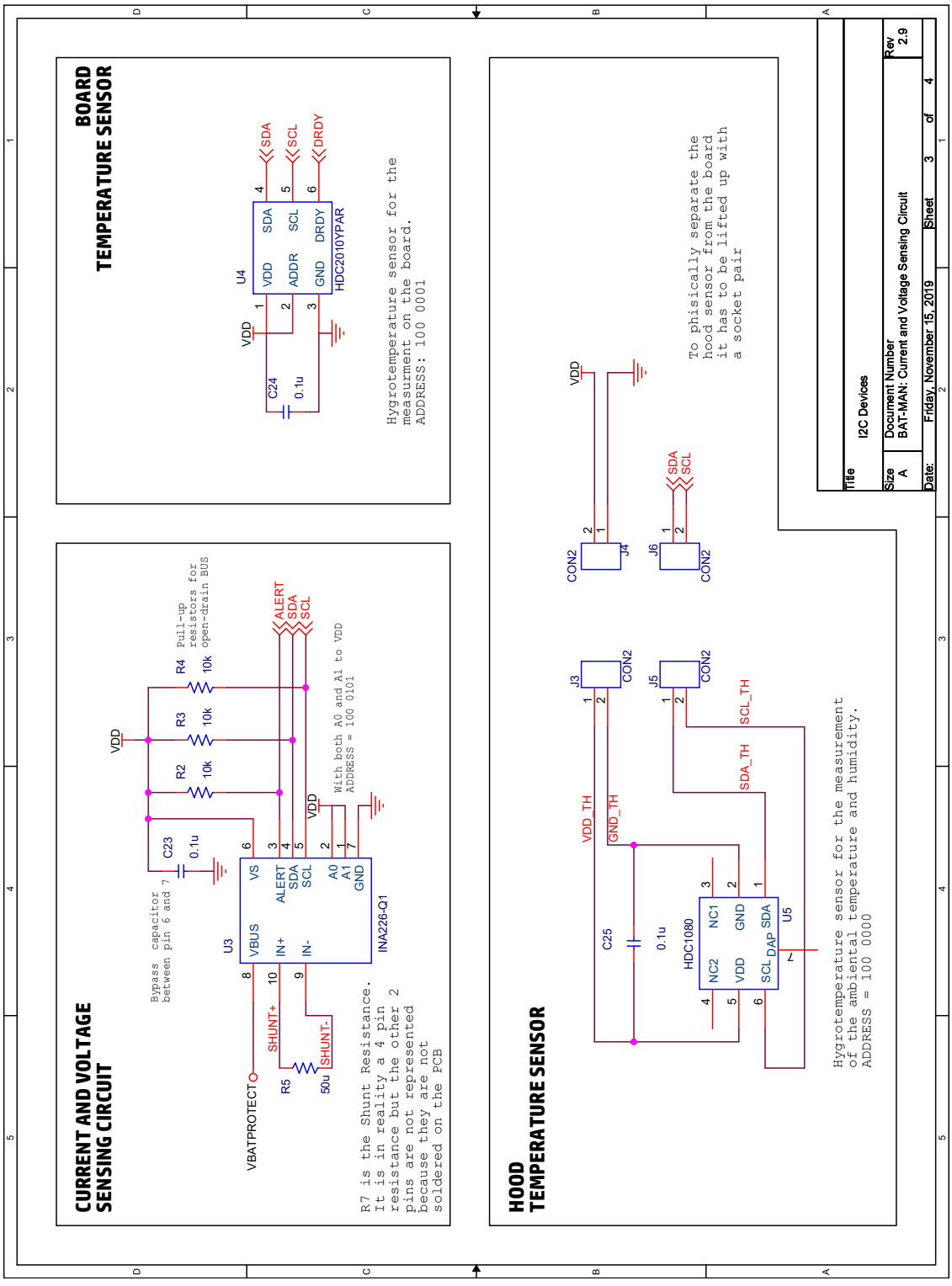


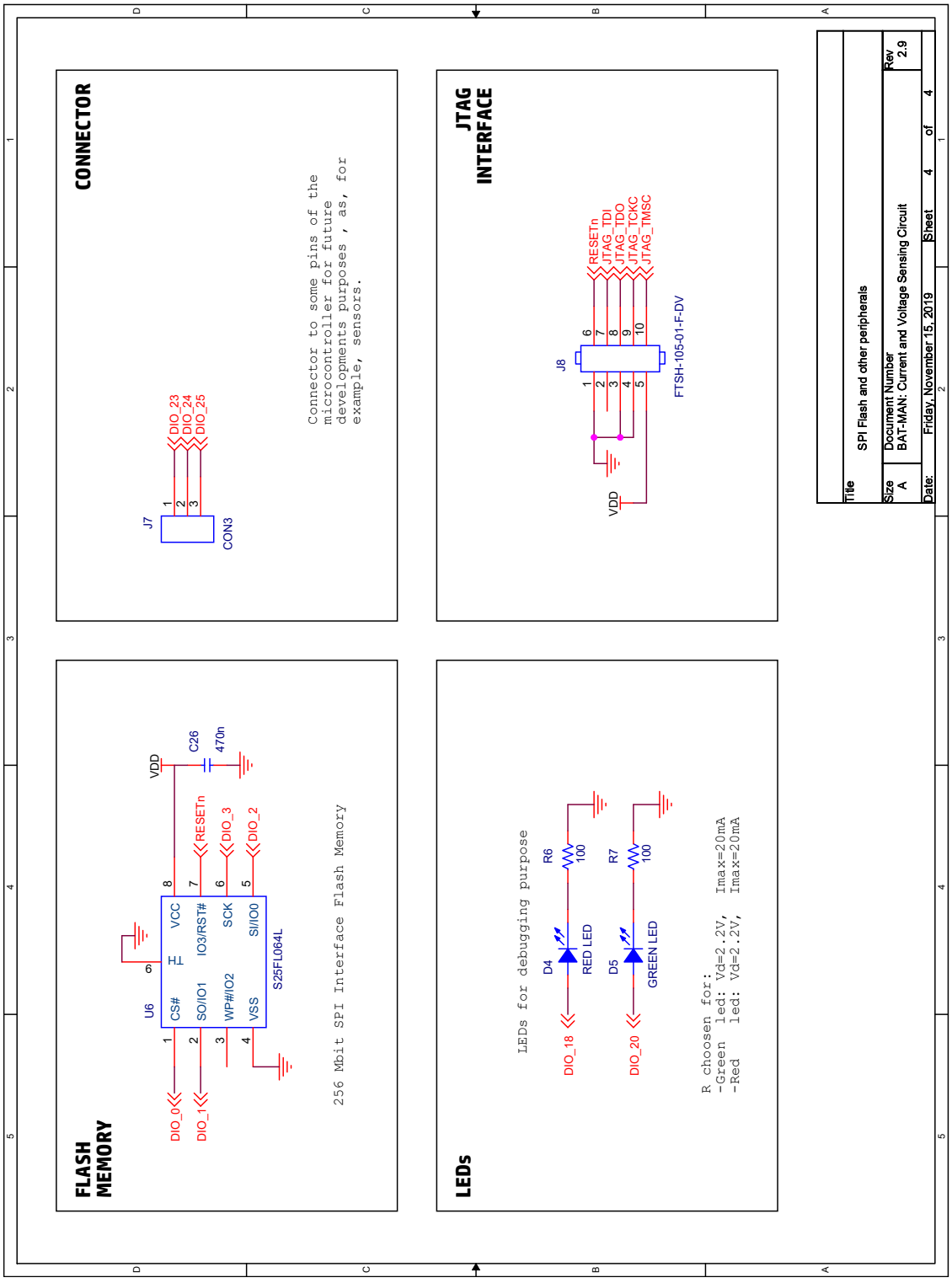
Figure 3.47. S-parameters for actual layout with optimized components values

3.3 Schematic explanation




DECOUPLING CAPACITORS
MICROCONTROLLER ANALOG AND RF
XTALs
MICROCONTROLLER GPIO
CC2640R2FTWRGZRQ1
TSX-3225
Title
Microcontroller
Document Number
BAT-MAN: Current and Voltage Sensing Circuit
Rev
2.9
Friday, November 15, 2019
Sheet
2
of
4


BOARD
TEMPERATURE SENSOR
U4
HDC2010YPAR
VDD
SDA
SCL
DRDY
ADDR
GND
C24
0.1u
Hygrotemperature sensor for the
measurment on the board.
ADDRESS: 100 0001
CURRENT AND VOLTAGE
SENSING CIRCUIT
U3
INA226-Q1
VBUS
VS
IN+
IN-
ALERT
SDA
SCL
A0
A1
GND
R2
R3
R4
10k
Pull-up
resistors for
open-drain BUS
C23
0.1u
Bypass capacitor
between pin 6 and 7
With both A0 and A1 to VDD
ADDRESS = 100 0101
R5
50u
SHUNT+
SHUNT-
VBATPROTECT
R7 is the Shunt Resistance.
It is in reality a 4 pin
resistance but the other 2
pins are not represented
because they are not
soldered on the PCB
HOOD
TEMPERATURE SENSOR
J4
J6
J3
J5
CON2
To phisically separate the
hood sensor from the board
it has to be lifted up with
a socket pair
C25
0.1u
HDC1080
NC1
NC2
GND
SCL_DAP
SDA
U5
VDD_TH
GND_TH
SDA_TH
SCL_TH
Hygrotemperature sensor for the measurement
of the ambiental temperature and humidity.
ADDRESS = 100 0000
Title
I2C Devices
Size
A
Document Number
BAT-MAN: Current and Voltage Sensing Circuit
Rev
2.9
Date:
Friday, November 15, 2019
Sheet
3
of
4


CONNECTOR
J7
CON3
DIO_23
DIO_24
DIO_25
Connector to some pins of the microcontroller for future developments purposes , as, for example, sensors.
JTAG INTERFACE
J8
FTSH-105-01-F-DV
VDD
RESETn
JTAG_TDI
JTAG_TDO
JTAG_TCKC
JTAG_TMSC
FLASH MEMORY
U6
S25FL064L
CS#
SO/IO1
WP#/IO2
VSS
VCC
IO3/RST#
SCK
SI/IO0
VDD
C26
470n
RESETn
DIO_3
DIO_2
DIO_0
DIO_1
256 Mbit SPI Interface Flash Memory
LEDs
LEDs for debugging purpose
D4
RED LED
D5
GREEN LED
R6
R7
100
DIO_18
DIO_20
R choosen for:
-Green led: Vd=2.2V, Imax=20mA
-Red led: Vd=2.2V, Imax=20mA
Title
SPI Flash and other peripherals
Size A
Document Number
BAT-MAN: Current and Voltage Sensing Circuit
Rev 2.9
Date: Friday, November 15, 2019
Sheet 4 of 4

3.3.1 Circuit For Electrical Transient Protection

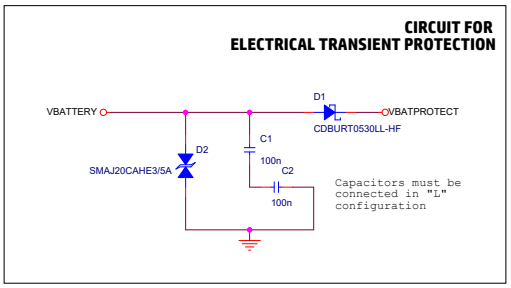


Figure 3.48. Transient Protection Circuit

For further explanation see subsection 3.1.2.

3.3.2 Buck Converter

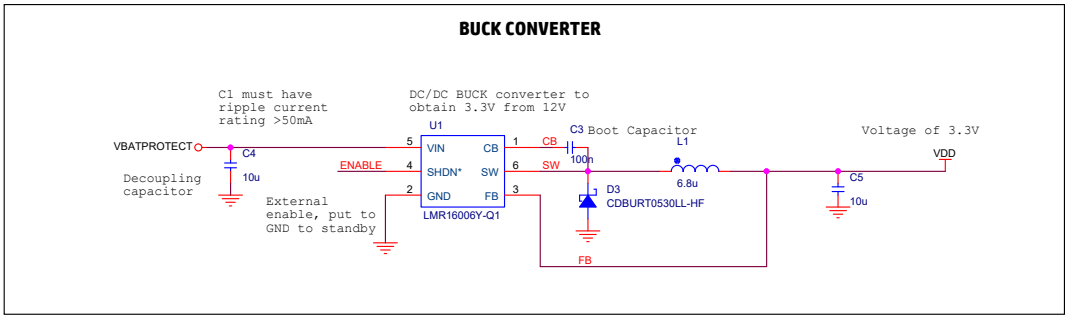


Figure 3.49. Buck Converter Circuit

This block represents the supply for the entire circuit, converting 12V battery voltage to 3.3V.

- C4 is the input bypass capacitor
- C3 is the bootstrap capacitor
- D3 is the flyback diode
- L1 is the output inductor
- C5 is the output capacitor

3.3.3 Decoupling Capacitors

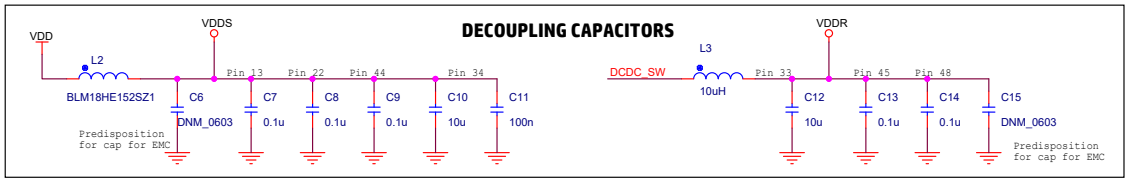


Figure 3.50. Microcontroller decoupling capacitor

In the left part:

- C6 is an additional capacitor which will be placed after built to overcome possible EMC problems
- C7 → C11 are decoupling capacitors needed to make it work properly
- L2 is a ferrite core for EMC compliance

In the right part:

- L3 and C12 are external components needed for the internal DCDC switching voltage regulator
- C13 and C14 are decoupling capacitor
- C15 is an additional capacitor which will be placed after built to overcome possible EMC problems

3.3.4 Microcontroller Analog and RF

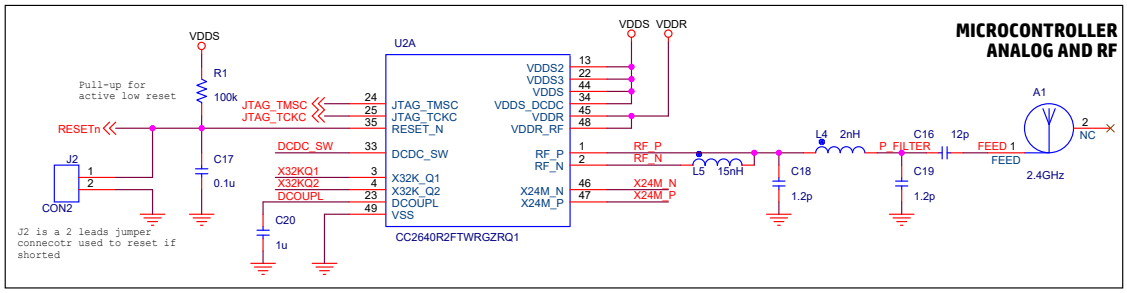


Figure 3.51. Microcontroller Analog and RF

- R1, C17 and CON2 are used to generate an active low Reset by mean of a jumper
- C20 is a decoupling capacitor
- L4, L5, C16, C18, C19 represents the matching network for the BLE chip antenna A1

3.3.5 Quartz Oscillators

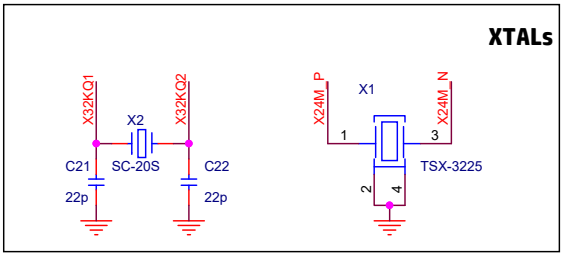


Figure 3.52. Quartz Oscillators

- X1 is the 24MHz XTAL needed to make the μC main clock
- X2 is the 32.768kHz XTAL needed to make a precise low frequency oscillator

Both the Oscillators are needed to ensure the correct behaviour of the BLE connection.

3.3.6 Current and Voltage Sensing Circuit

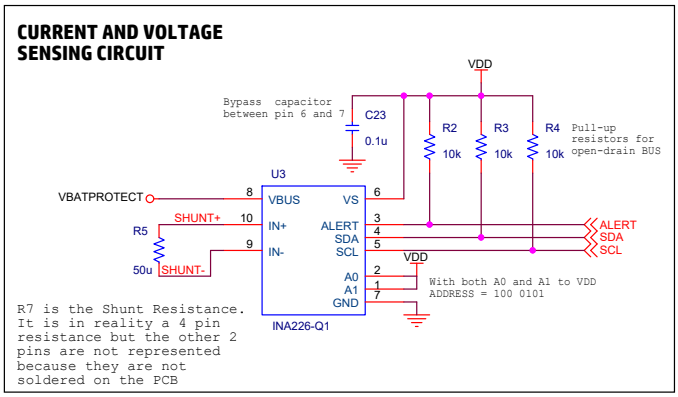


Figure 3.53. Current and Voltage Sensing Circuit

- R5 is the shunt resistor, the core of the project
- R2, R3 and R4 are the pull up resistors for I2C communication and ALERT function
- C23 is the decoupling capacitor needed for INA226

A0 and A1 pins of the INA226 are both connected to VDD so to get a specific I2C address (100 0101).

3.3.7 Flash Memory

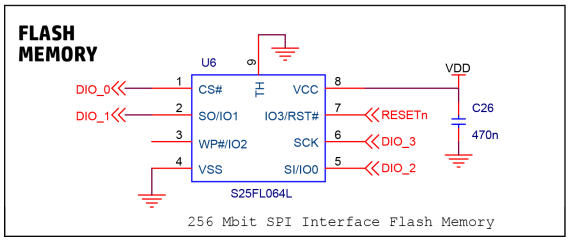


Figure 3.54. NOR Flash Memory

- C26 is the decoupling capacitor which should have a greater value than the usual 100nF - here it is 470nF - because it has to overcome the significant current absorption during erasing/programming operation

3.3.8 Board Temperature Sensing

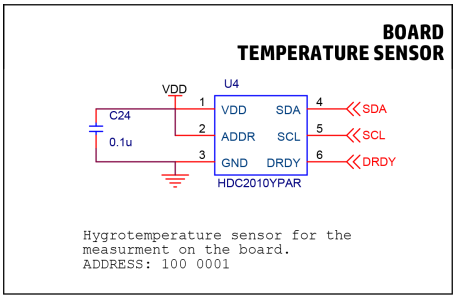


Figure 3.55. Board Temperature Sensor

- C24 is the decoupling capacitor needed for HDC2010
- Address pin is stacked to VDD results in I2C address 100 0001

3.3.9 Hood Temperature Sensing

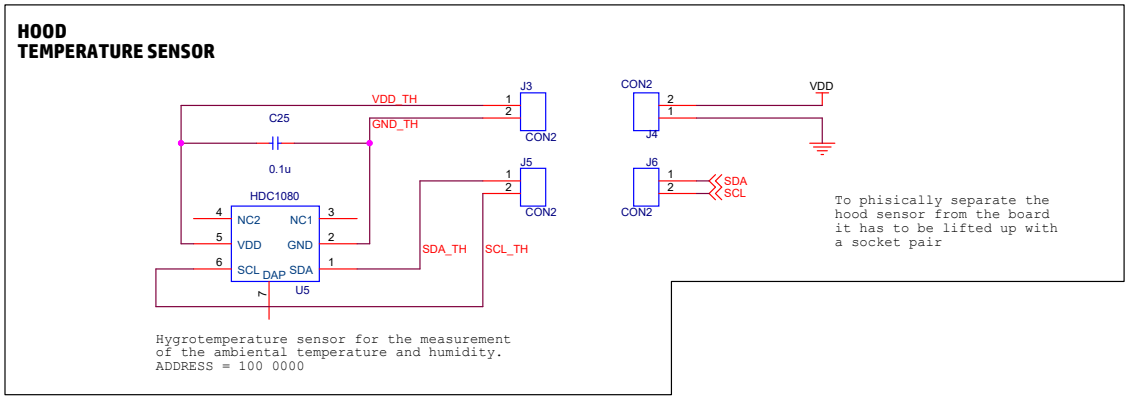


Figure 3.56. Hood Temperature Sensor

- C25 is the decoupling capacitor needed for HDC1080
- J3 is a female socket (on HDC1080 board) pairing with J2 male socket (on the main board), there to supply the sensor
- J5 is a female socket (on HDC1080 board) pairing with J6 male socket (on the main board), which provides I2C connection

3.4 Layout explanation

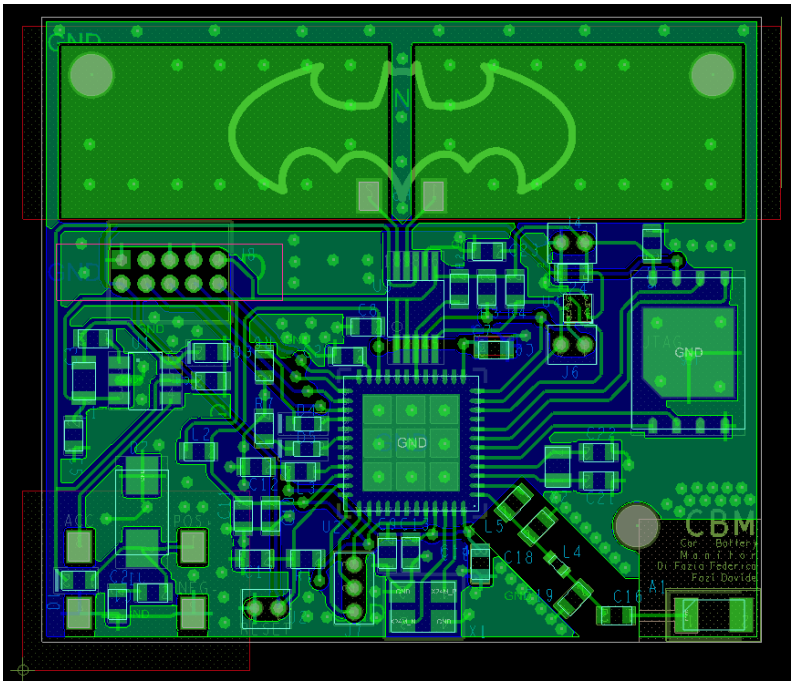


Figure 3.57. PCB Layout from OrCAD PCB Designer

The following step consists in the layout design; in order to do this it is necessary to export the netlist from *Capture CIS* tool and import it back in the *OrCAD PCB Designer* tool.
Once the components have been imported they must be properly placed, according to the following rules:

- Place bypass capacitor as near as possible to the related supply pin, avoiding to put them on the other layer
- Separate analog components from the digital ones. In this project the only analog component is the INA226, whose analog pins are placed on the upper part of the board
- Use big pads for the shunt resistor to dissipate heat
- Place the antenna and its matching circuit as far as possible from the sensing circuit in order to reduce acquisition disturbances. Crystals must be far from analog part too.

- Place antenna matching components in a straight line as in Figure 3.58 to avoid impedance mismatching due to the traces behaviour; at high frequencies they act as microstrips
- Antenna matching filter components must be placed as in Figure 3.27 to reduce crosstalk interferences.
- Place all the components on the TOP layer: shunt resistor is included in a plastic case (see Figure 3.11) otherwise it will be not be possible to replace them in case of failure
- Shunt resistor and MX150 connector position are fixed since they are included in the case
- Connectors must be accessible from the outside

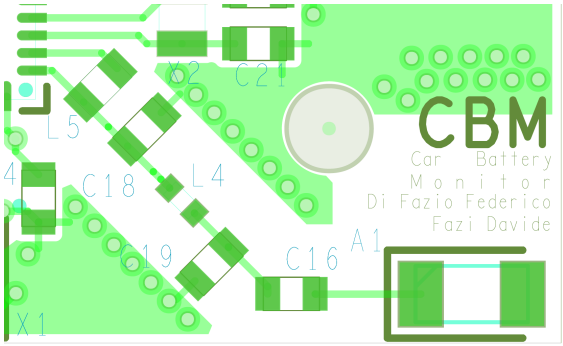


Figure 3.58. Zoom of the Antenna Matching

After the placing process, components must be connected through traces (Routing Process) according to the following rules:

- Avoid long traces when possible
- Avoid ground loops
- Place the ground plane on both sides of the PCB connecting them with many vias in order to be equipotential
- Avoid unconnected islands on the ground plane and traces that divide the ground plane
- Let the area under the antenna and the matching circuit free from ground plane to avoid unwanted shielding

- Use via stitching technique - which consists in bounding an area with a via fence - around RF zone to ensure shielding with the remaining part of the board

- Connect the pins of the shunt resistor to the differential pair of the INA226 through a Kelvin Sense connection with symmetrical traces as in Figure 3.59

- For the components that had a recommended layout (reported in the datasheet) the placing and routing process should be done according to it

- Components have to be connected to ground through a 3-point connection. By using a full-contact connection manual soldering process could be tricky because heat would be dissipated by the whole ground plane

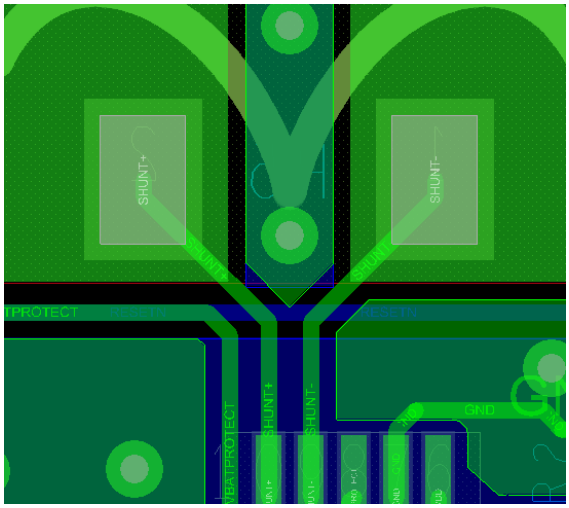


Figure 3.59. Zoom of the Kelvin Sense

After the whole designing process is done, gerber files must be generated in order to send them to the manufacturer to be printed.

Gerber files of both layers are merged in Figure 3.60 and Figure 3.61.

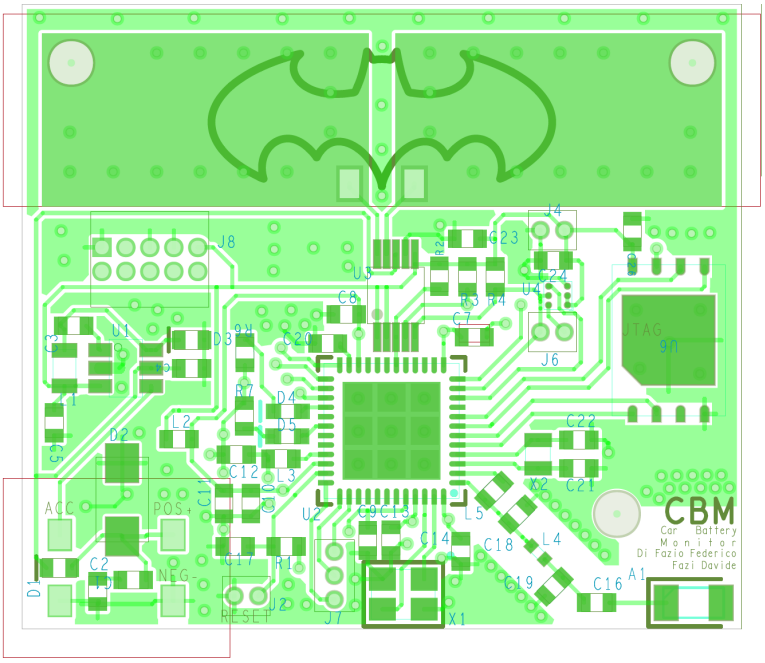


Figure 3.60. PCB Gerber Top

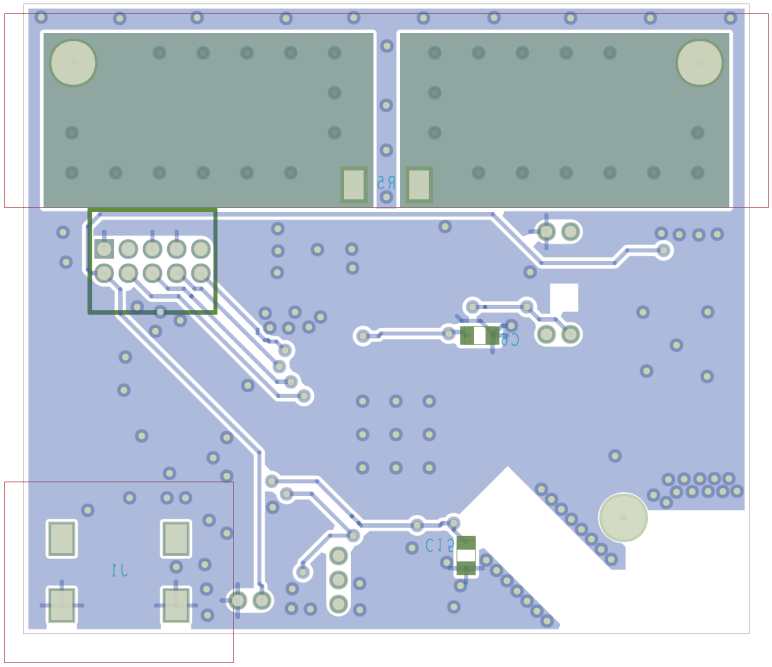


Figure 3.61. PCB Gerber Bottom

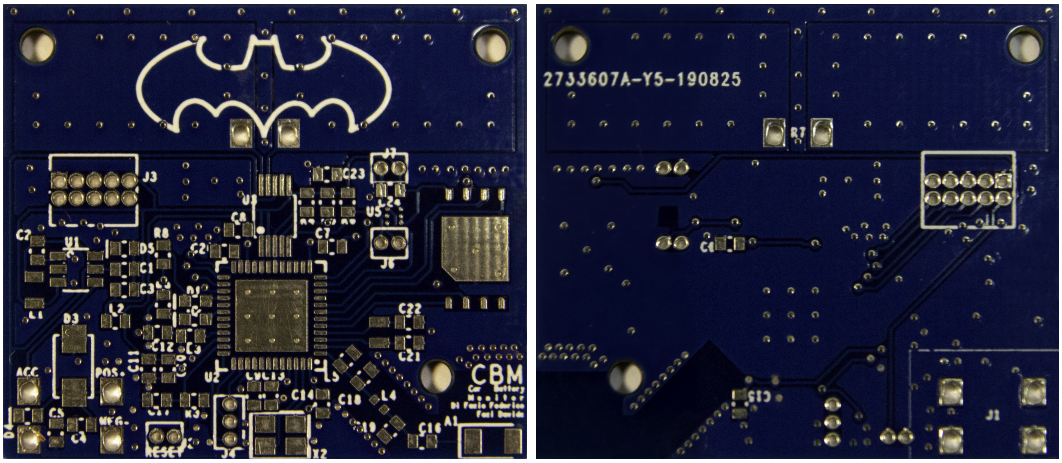
3.5 Bill of Material

Idx	#	Manufacturer N.	Manufacturer	Reference	Value	Price @ 1u	Price @ 10ku
1	1	2450AT18A100E	Johanson	A1	Antenna @ 2.4GHz	0.56 €	1 760.00 €
2	13	GCM188R91E104KA37J	Murata	C1,C2,C3,C7,C8, C9,C11,C13,C14, C17,C23,C24,C25	100nF Capacitor	1.68 €	3 250.00 €
3	1	C0603X120J5GACAUTO	KEMET	C16	12pF Capacitor	0.34 €	670.00 €
4	2	06035A1R2K4T2A	AVX	C18,C19	1.2pF Capacitor	0.50 €	760.00 €
5	1	CGA3E2X5R1A105M080AA	TDK	C20	$1\mu F$ Capacitor	0.09 €	130.00 €
6	2	AC0603JRNPO9BN220	Yageo	C21,C22	22pF Capacitor	0.22 €	280.00 €
7	1	CGA3E1X7R1C474M080AC	TDK	C26	470nF Capacitor	0.14 €	190.00 €
8	4	GRT188C81C106ME13D	Murata	C4,C5,C10,C12	$10\mu F$ Capacitor	1.66 €	3 880.00 €
9	2	CDBURT0530LL-HF	Comchip Technology	D1,D3	Schottky Diode	0.72 €	1 560.00 €
10	1	SMAJ20CAHE3/61	Vishay	D2	TVS Diode	0.43 €	1 040.00 €
11	1	SML-D12U1WT86	ROHM Semiconductor	D4	Red LED	0.19 €	240.00 €
12	1	SML-D12M8WT86	ROHM Semiconductor	D5	Green LED	0.29 €	370.00 €
13	1	33472-4001	Molex	J1.1	RCPT Connector	2.04 €	10 700.00 €
14	4	33012-2001	Molex	J1.2	14/16AWG Connector	0.58 €	3 360.00 €
15	1	M50-1930005	Harwin	J2.1	Jumper	0.29 €	1 120.00 €
16	5	M50-3530242	Harwin	J2,J4,J6	2 Pin Male	0.50 €	1 850.00 €
17	2	M50-3030242	Harwin	J3,J5	2 Pin Female	1.06 €	4 500.00 €
18	1	M50-3530342	Harwin	J7	3 Pin Male	0.13 €	450.00 €
19	1	FTSH-105-01-L-D-K	Samtec	J8	JTAG Connector	2.79 €	13 900.00 €
20	1	KLZ2012NHR101LTD25	TDK	L1	$100\mu H$ Inductor	0.28 €	840.00 €
21	1	BLM18HE152SZ1D	Murata	L2	Ferrite	0.18 €	490.00 €
22	1	KLZ1608MHR100WTD25	TDK	L3	$10\mu H$ Inductor	0.23 €	690.00 €
23	1	MLG1005S2N0BTD25	TDK	L4	2.0nH Inductor	0.09 €	240.00 €
24	1	LQW18AS15NG0ZD	Murata	L5	15nH Inductor	0.22 €	650.00 €
25	1	AC0603FR-07100KL	Yageo	R1	100KΩ Resistor	0.09 €	40.00 €
26	3	AC0603FR-0710KL	Yageo	R2,R3,R4	10KΩ Resistor	0.27 €	90.00 €
27	1	WSBM8518L0500JK	Vishay	R5	50uΩ Resistor	15.26 €	76 100.00 €
28	2	AC0603FR-07100RL	Yageo	R6,R7	100Ω resistor	0.18 €	80.00 €
29	1	LMR16006YQ3DDCRQ1	Texas Instruments	U1	Buck Regulator	2.97 €	14 600.00 €
30	1	CC2640R2FTWRGZRQ1	Texas Instruments	U2	Microcontroller	7.44 €	40 700.00 €
31	1	INA226AQDGSRQ1	Texas Instruments	U3	Sensing Circuit	2.97 €	14 600.00 €
32	1	S25FL256LAGNFM010	Cypress Semiconductor	U4	NOR Flash	8.53 €	46 300.00 €
33	1	HDC2010YPAR	Texas Instruments	U5	TH Sensor	2.31 €	10 200.00 €
34	1	HDC1080DMBR	Texas Instruments	U7	TH Sensor	2.52 €	12 400.00 €
35	1	SC20S-7PF20PPM	Seiko Instruments	X1	32.768kHz XTAL	0.71 €	3 530.00 €
36	1	TSX-3225 24.0000MF15X-AC3	Epson	X2	24MHz XTAL	0.32 €	1 390.00 €
					Total:	58.78 €	272 950.00 €

Table 3.10. Bill of Material with prices from Mouser Electronics Inc.

3.6 Mounting Process

3.6.1 Soldering Paste Spreading



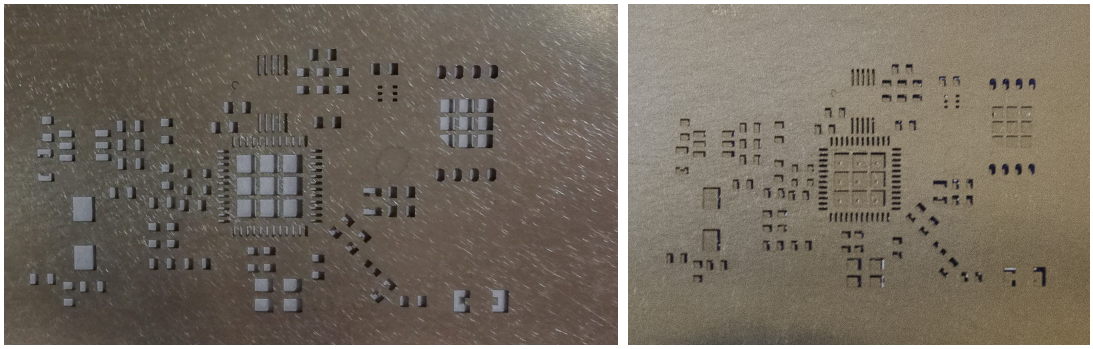
(a) Top View (b) Bottomo View

Figure 3.62. PCB Received by the Manufacturer

After obtaining the Gerber files, they have been sent to a manufacturer to obtain printed PCBs and stencils. The following phase consists in soldering the components on the PCB in the proper way. The process consists in the following steps:

- Soldering paste spreading
- Components pick and placing
- Reflow oven soldering

3.6.2 Soldering Paste Spreading



(a) Stencil

(b) Stencil Properly Placed Over the PCB

Figure 3.63. Stencil of the BAT-MAN Circuit

Stencil is laser cut foil, each hole corresponds to the padstack of an SMD component. It is a mask that must be applied to the PCB in order to spread the soldering paste only on the padstacks of the components to be soldered with the reflow oven. In Figure 3.63 is reported the stencil used for the circuit.

The stencil must be inserted in the Stencil Mate machine, this operation is done by tightening the screws. After the stencil is properly fixed it must be tended to obtain a smooth surface; no air must be present between stencil and PCB to avoid leaks of paste in undesired regions.

Figure 3.64. Stencil Tended in the Stencil Mate

The trickiest part is to properly place the PCB - thanks to magnetic supports - under the stencil, so to align padstacks and stencil cuts. When this condition is satisfied the PCB can be lifted up to be strictly in contact with the foil.

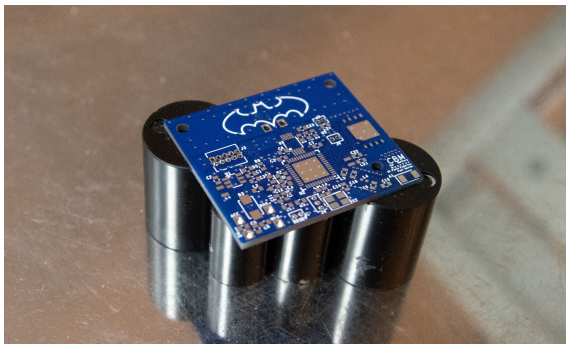


Figure 3.65. PCB Positioned in the Magnetic Supports

The next step consists in spreading the soldering paste on the PCB by mean of a spatula. The deposed layer must be thin, otherwise an excess of it could imply the creation of short circuits during the melting of the solder.

3.6.3 Components Pick and Placing

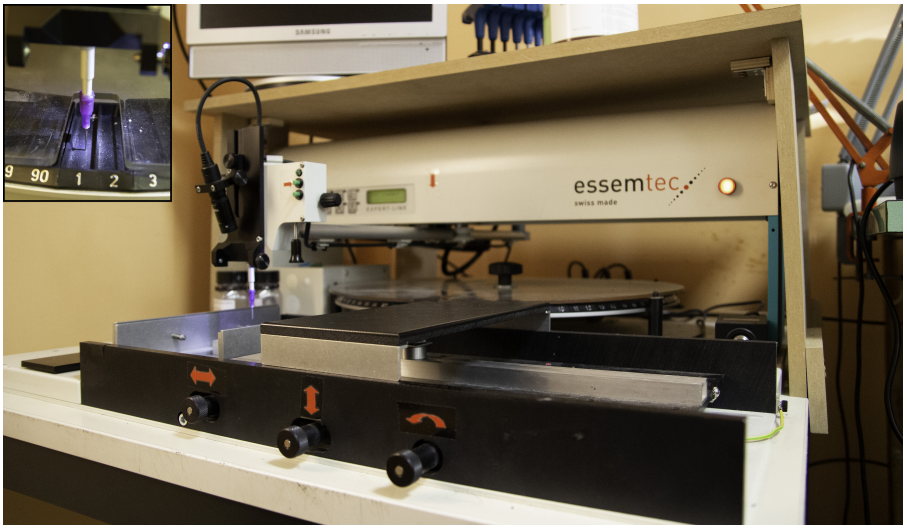
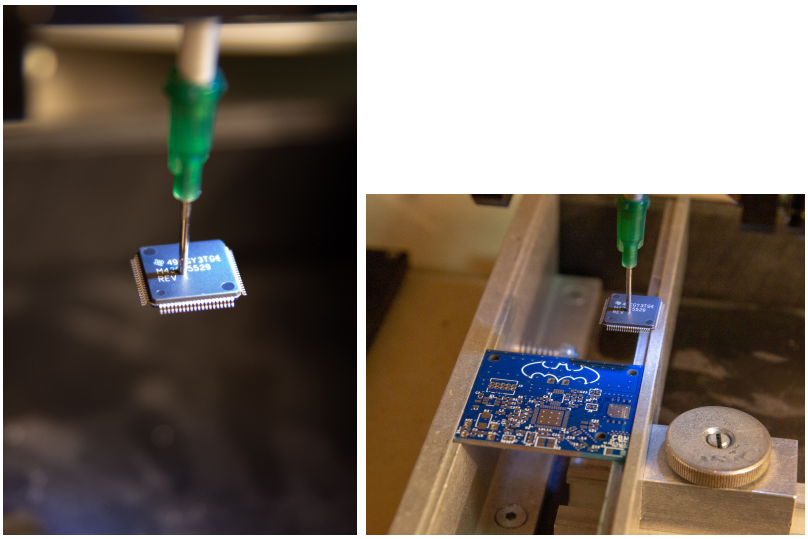


Figure 3.66. Pick and Place Machine ()zoom of the components holder on the left)

The SMD components must be placed on the PCB in the correct position by mean of a pick and place. This machine consists in a needle that lifts up components thanks to a vacuum, the component can be moved using four degree of freedom (x,y,z axis and rotation).

When the component is in the proper position it can be placed and, when it touches the surface, the vacuum turns off leaving it on the PCB. During this phase the component is not soldered but the paste has a mild glueing effect that avoids problems related to unwanted movement due to PCB shaking.
In Figure 3.67b, the pads are not covered by soldering paste because the photo has not been taken during the real mounting process. An example of pads covered by solder is in Figure 3.68



(a) Needle Magnification (b) Placing Process

Figure 3.67. Pick and Place Process

The described process has to be repeated for all the components, to be sure to have correctly placed all of them, a microscope can be used; magnification is needed for very little components.

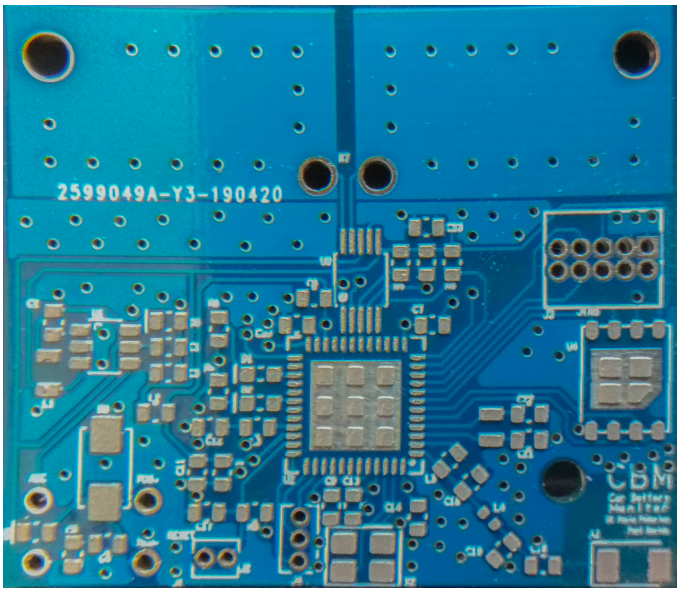


Figure 3.68. Example of Pads Covered by Paste

3.6.4 Reflow Oven Soldering

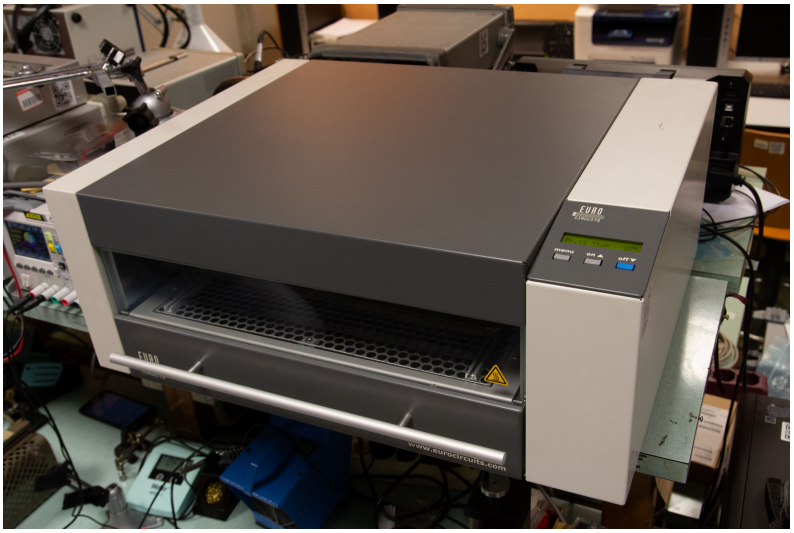


Figure 3.69. Reflow Oven

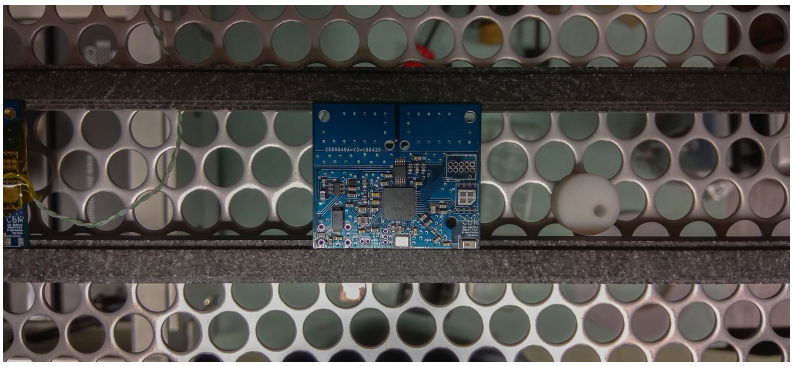


Figure 3.70. Populated PCB Placed in the Reflow Oven's Supports

The PCB populated with all the components must be placed in the reflow oven to melt down the solder. Reflow oven is a special furnace engineered for this purpose, it can be programmed to obtain a specific soldering profile. This must be complaint with the datasheet of the soldering paste, moreover the peak temperature must be lower than the maximum temperature rating of all the involved components.

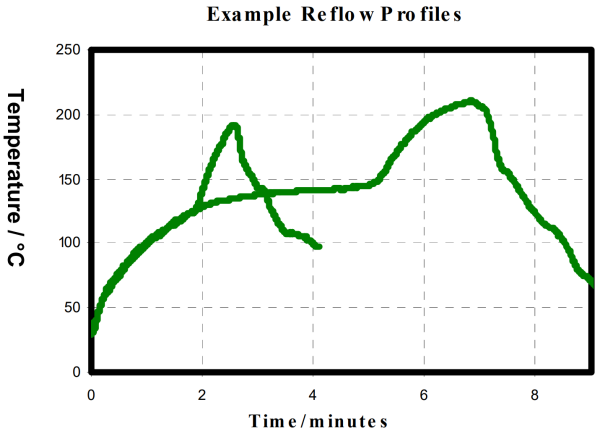


Figure 3.71. Soldering Paste Profile From Datasheet [27]

In general the soldering profile consists in a relative slow heating until the reaching of the melting temperature, followed by a rapid cooling obtained by opening the oven's door. Inside the furnace two thermocouples allow to monitor both the temperature of the chamber and of the PCB. The second sensor is usually placed on a dummy PCB produced with the same materials of the one to be soldered.

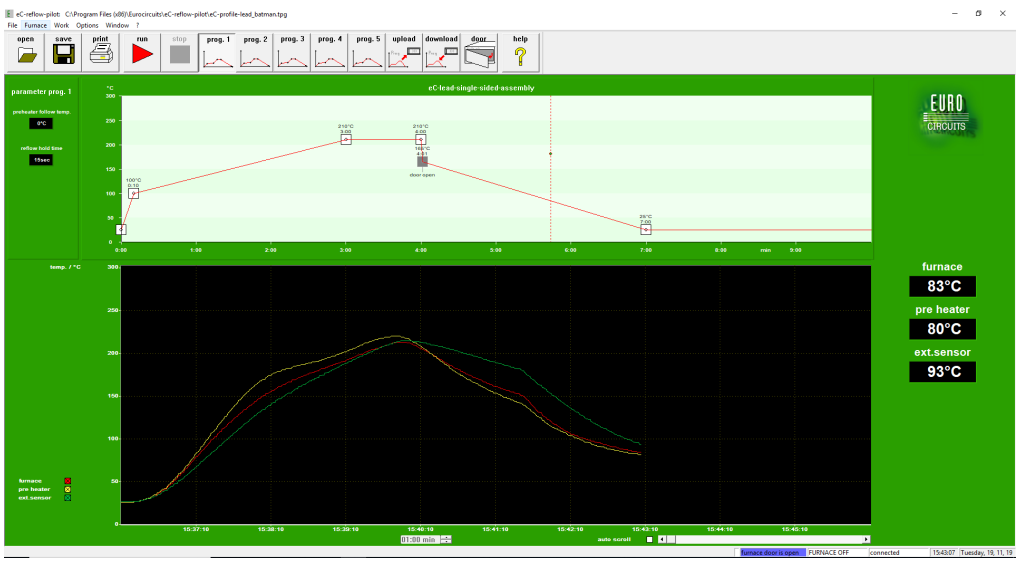


Figure 3.72. Reflow Oven Soldering Profile for Lead Solder Paste

3.6.5 Final Result

Since the through hole components can not be inserted in the reflow oven they must be hand-soldered. Especially the shunt resistor is the last component to be connected since it is die in a plastic case.

Over the main PCB a little board is connected through sockets. It contains the HDC1080 whose purpose is to evaluate hood's temperature, therefore it is lifted to be far from the device itself.

The final result is in Figure 3.73 where it is equipped with supply connector - conneted to the molex MX-150 pin - and both female and male battery plugs. The first one is used to connect the device to the battery, the second one will be attached to the negative cable connector of the car.

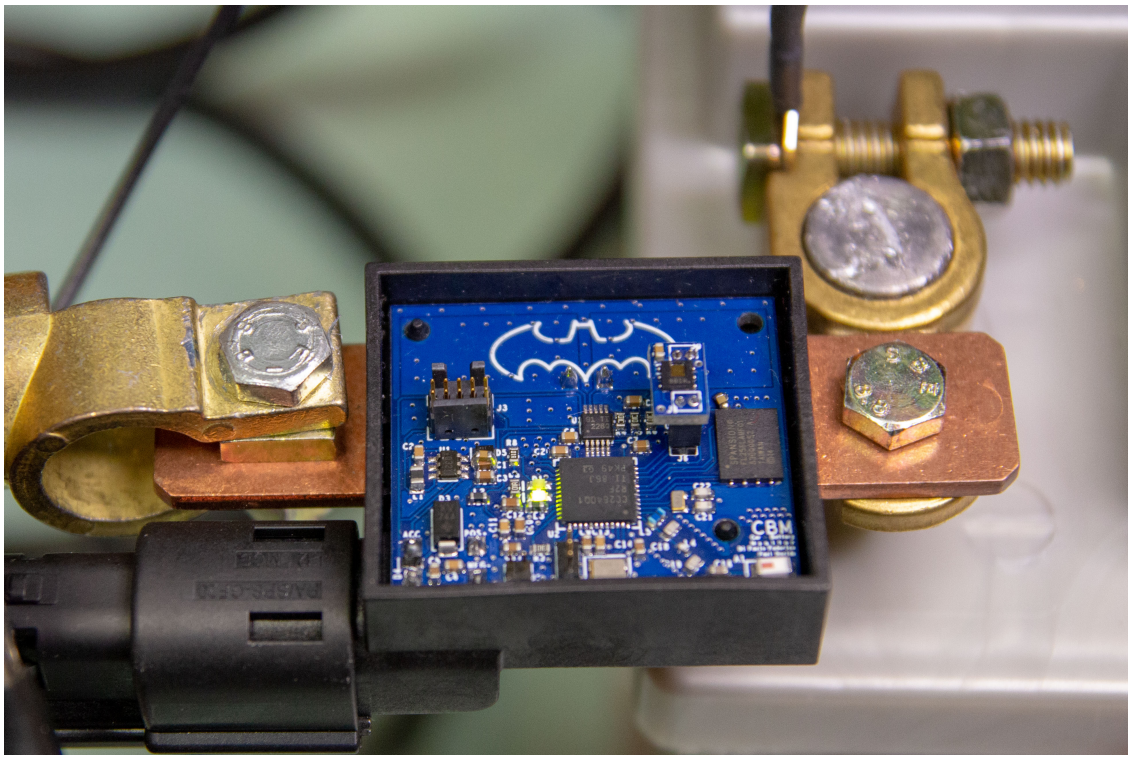


Figure 3.73. Device Completely Mounted

Chapter 4

Firmware

4.1 RTOS

> *Real time systems are defined as those systems in which the correctness of the system depends not only on the logical result of the computation, but also on the time at which the results are produced.*
>
> J. Stankovic, "Misconceptions About Real Time Computing,"
> IEEE Computer, 21(10), October 1988.

A real-time operating system (RTOS) is an operating system intended to serve real time applications that process data as it comes in. It is a time bound system which has well defined fixed time constraints, if these are not satisfied the system will fail.

The advantage of having a RTOS is the possibility to run several threads at the same time, thus incrementing the speed of execution and the optimization of the code. This behaviour can present code critical sections, avoidable by using mutex and semaphores. To achieve a right execution of the program, a scheduling algorithm must be adopted.

Scheduling

The objective of multiprogramming is to have some process running at all times, to maximize the CPU utilization. The target of time sharing is to switch the CPU among processes so frequently that users can interact with each program while it is running. To meet these objectives, the process scheduler selects an available process for program execution on the CPU.

Since program is executed concurrently, scheduler has to determine which thread

executes next exploiting a scheduling mechanism; this provides the tools and environment for controlling the flow of the thread through the various queues by mean of *Context Switching* and *Queueing Capabilities*.
A scheduling policy is needed to define which ready thread has to be chosen from the ready list.[31]

4.1.1 TI-RTOS

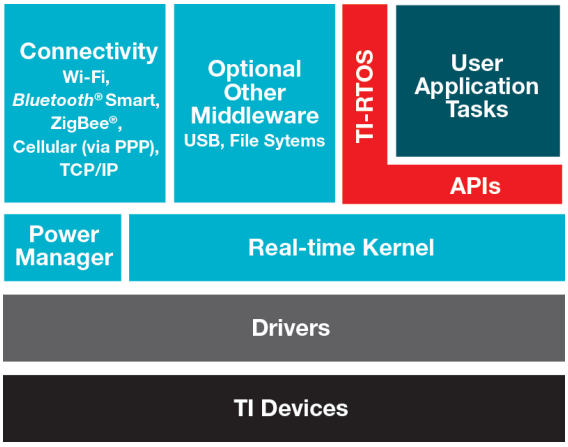


Figure 4.1. TI-RTOS Hierarchical Structure

Texas Instruments has developed its own RTOS necessary to make the BLE GATT stack working.

By default, it uses *Preemptive Scheduling* that is the most common type of RTOS scheduler. With a preemptive scheduler, a running thread continues until it

- finishes, as when an ISR (Interrupt Service Routine) completes
- a higher priority thread becomes ready - in this case the higher priority thread preempts the lower priority thread
- the thread gives up the processor while waiting for a resource (e.g. a task calls sleep()).

4.2 Sensor Controller - Peripherals Management

Sensor Controller is a coprocessor specifically designed to manage sensors with both ananlog and digital interface.
It can be programmed using the ad-hoc created IDE called Sensor Controller Studio. This 16-bit architecture allows to implement an asymmetric multicore-processing; while it is acquiring data from sensors the main core can either perform other tasks or be in idle mode to reduce power consumption.

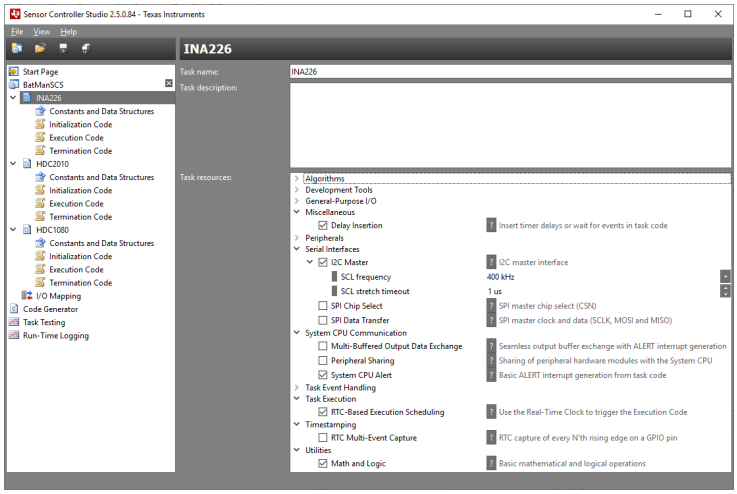


Figure 4.2. SCS Screen for INA226 Peripherals Selection

Digital communications - such as I2C and SPI - are bit-banged; software is used to generate and process signals instead of dedicated hardware.

In this project SC has been used to communicate via I2C protocol with Hygro-Thermometers and Current/Voltage sensing circuit.
At a first glance, the intention was to use it also to manage the external Flash Memory; actually it was not possible due to limited resources and to the impossibility to create a library inside the provided IDE.

Therefore this useful tool has a reduced application field restricted to simple and slow peripherals.

The program can be organized in different tasks which are not thought to communicate with each other. Every task is divided into an *Inizialition Code*, ran only once, an *Execution Code*, repeated programmatically or on request, and a *Termination Code* in which the code to shut down a device should be placed, actually it is not used as most sensors don't need to be terminated in a specific way.

4.2.1 Sensor Controller Tasks

For this project the involved resources are:

- **Delay Insertion** to obtain FW delays required by the sensors
- **I2C Master** to set the I2C peripheral in master mode and select frequency and clock stretch timeout
- **System CPU Alert** generates an interrupt to be recognized by the main CPU
- **RTC-Based Execution Scheduling** to trigger the execution code using RTC according to settings in main CPU
- **Math and Logic** implements basic mathematical and logical operations

For every task, there is the possibility to set the I/O mapping relative to the used peripherals shown in Figure 4.3. Exploiting I2C protocol the three tasks use the same DIOs.

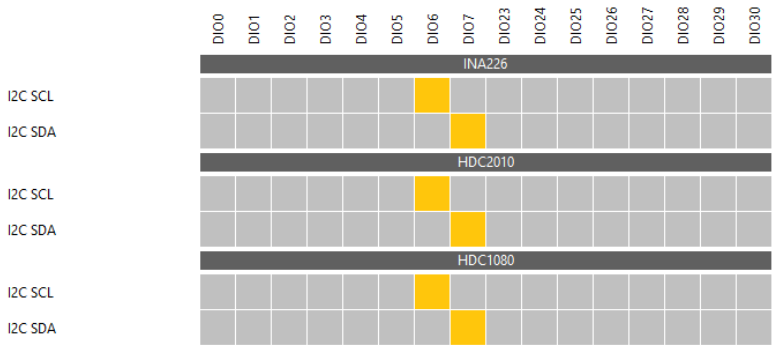


Figure 4.3. Sensor Controller I/O Mapping for the Tasks

INA226

Initialization Code

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
RST				AVG2	AVG1	AVG0	VBUSCT2	VBUSCT1	VBUSCT0	VSHCT2	VSHCT1	VSHCT0	MODE3	MODE2	MODE1
0	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	RESERVED			4 AVERAGE			8.244ms BUS CONVERSION			8.244ms SHUNT CONVERSION			SHUNT+BUS, CONTINUOUS		

Table 4.1. INA226 Configuration Register

The initialization code is used to setup the parameters in the configuration register in order to select the conversion times and the number or averages; according to this value, n acquisitions are averaged out to reduce noise influence.

```
1  /* -------------------------SETTING CONFIGURATION REGISTER------------------------- */
2  i2cStart();
3  i2cTx(INA226_I2C_ADDR | I2C_OP_WRITE);
4  i2cTx(CONFIG_REG);
5  i2cTx(cfg.config >>8);
6  i2cTx(cfg.config & 0xFF);
7  i2cStop();
8
9  fwDelayUs(100000,FW_DELAY_RANGE_100_MS);
10
11 /* ------------------CHECKING CORRECTNESS OF CONFIGURATION REGISTER------------------ */
12 i2cStart();
13 i2cTx(INA226_I2C_ADDR | I2C_OP_WRITE);
14 i2cTx(CONFIG_REG);
15 i2cRepeatedStart();
16 i2cTx(INA226_I2C_ADDR | I2C_OP_READ);
17 i2cRxAck(output.configH);
18 i2cRxAck(output.configL);
19
20 i2cStop();
21
22 fwGenAlertInterrupt();
```

The code ends with the callback *fwGenAlertInterrupt()* to generate an interrupt useful to control the execution flow in the main program.

Execution Code

```
U16 CNVR;
U16 MaskEnH;
U16 MaskEnL;
CNVR=0;
MaskEnH=0;
MaskEnL=0;

if (state.i2cStatus == 0x0000) {
  for(U16 n=0; n<10; n++){  // acquire 10 consecutive samples
    CNVR=0;
/* ----------------------------------START ACQUISITIONS---------------------------------- */
    i2cStart();
    i2cTx(INA226_I2C_ADDR | I2C_OP_WRITE);
    i2cTx(SHUNT_V_REG);
    i2cRepeatedStart();
    i2cTx(INA226_I2C_ADDR | I2C_OP_READ);
    i2cRxAck(output.shuntH);
    i2cRxAck(output.shuntL);
    i2cStop();

    i2cStart();
    i2cTx(INA226_I2C_ADDR | I2C_OP_WRITE);
    i2cTx(BUS_V_REG);
    i2cRepeatedStart();
    i2cTx(INA226_I2C_ADDR | I2C_OP_READ);
    i2cRxAck(output.busH);
    i2cRxAck(output.busL);
    i2cStop();

    output.mVBusVoltage[n] = (output.busH<<8) | output.busL;
    output.mVShuntVoltage[n] = (output.shuntH<<8) | output.shuntL;
/* ----------------------------CHECK IF NEXT CONVERSION READY---------------------------- */
    while(CNVR==0){
      i2cStart();
      i2cTx(INA226_I2C_ADDR | I2C_OP_WRITE);
      i2cTx(MASK_EN_REG);
      i2cRepeatedStart();
      i2cTx(INA226_I2C_ADDR | I2C_OP_READ);
      i2cRxAck(MaskEnH);
      i2cRxAck(MaskEnL);
      i2cStop();

      CNVR=MaskEnL & 00001000;
    }
  }
} else {
  i2cStop();
}

fwGenAlertInterrupt();
```

The core of the entire program is the INA226 task *Execution Code* where Bus and Shunt Voltages are acquired by reading the proper register through two I2C communications.
I2C protocol allows to only "transfer" 8bit at a time while the sensor has a 16bit ADC, therefore - after the acquisition - the most significant byte and the least significant byte must be combined into a 16bit value by using a left-shift and an OR operation (Value = (MSB«8) | LSB).
INA226 has an ALERT pin which can be used to generate an interrupt when the conversion is ready; unfortunately in this project it could only be connected to a pin not mapped in Sensor Controller. For this reason the solution came by analysing the *MaskEnable* register which contains the bit *CNVR* being asserted as well as the ALERT pin.
However, in this way the register has to be read in polling using I2C bus, thus wasting time and resources. Actually this is not a relevant problem as it is acquiring at a very slow rate.

HDC2010

Execution Code

```
if (state.i2cStatus == 0) {
/* ---------------------------------TRIGGER ACQUISITIONS-------------------------------- */
  i2cStart();
  i2cTx(HDC2010_I2C_ADDR | I2C_OP_WRITE);
  i2cTx(MEAS_CONFIG);
  i2cTx(0x01);
  i2cStop();
/* -------------------HUMIDITY IS AUTOMATICALLY READ AFTER TEMPERATURE------------------ */
  i2cStart();
  i2cTx(HDC2010_I2C_ADDR | I2C_OP_WRITE);
  i2cTx(TEMPL);
  i2cRepeatedStart();
  i2cTx(HDC2010_I2C_ADDR | I2C_OP_READ);
  i2cRxAck(output.TempL);
  i2cRxAck(output.TempH);
  i2cRxAck(output.HumL);
  i2cRxNack(output.HumH);
  i2cStop();
  output.temperature2010 = (output.TempH<<8) | output.TempL;
  output.humidity2010    = (output.HumH <<8) | output.HumL;
} else {
  i2cStop();
}
fwGenAlertInterrupt();
```

HDC1080

RST	[15]	Software reset bit	0	Normal Operation, this bit self clears
			1	Software Reset
Reserved	[14]	Reserved	0	Reserved, must be 0
HEAT	[13]	Heater	0	Heater Disabled
			1	Heater Enabled
MODE	[12]	Mode of acquisition	0	Temperature or Humidity is acquired.
			1	Temperature and Humidity are acquired in sequence, Temperature first.
BTST	[11]	Battery Status	0	Battery voltage >2.8V (read only)
			1	Battery voltage <2.8V (read only)
TRES	[10]	Temperature Resolution	0	14 bit
			1	11 bit
HRES	[9:8]	Humidity Resolution	0	14 bit
			1	11 bit
			10	8 bit
Reserved	[7:0]	Reserved	0	Reserved, must be 0

Table 4.2. HDC1080 Configuration Register detail.

Initialization Code

```
i2cStart();
i2cTx(DEV_ADD | I2C_OP_WRITE);
i2cTx(CONFIG);
i2cTx(cfg.config >>8);
i2cTx(cfg.config & 0xFF);
i2cStop();

fwDelayUs(20000,FW_DELAY_RANGE_100_MS); // wait for 20ms
```

Execution Code

```
if (state.i2cStatus == 0x0000) {
/* --------------SEND COMMAND TO READ TEMPERATURE AND HUMIDITY CONSEQUENTLY-------------- */
  i2cStart();
  i2cTx(I2C_OP_WRITE | DEV_ADD);
  i2cTx(TEMPERATURE);
  i2cRepeatedStart();
  i2cTx(I2C_OP_READ | DEV_ADD);
  i2cRxAck(output.TempH);
  i2cRxAck(output.TempL);
  i2cRxAck(output.HumH);
  i2cRxNack(output.HumL);
  i2cStop();
  output.temperature = (output.TempH <<8)+output.TempL;
  output.humidity = (output.HumH <<8)+output.HumL;
} else {
  i2cStop();
}
fwGenAlertInterrupt();
```

The code above, related to HDC2010 and HDC1080 TH sensors, simply describes I2C communications with the only difference that in HDC2010, when triggering a measure, it has to be read from the register while, in HDC1080, the measured value is transferred automatically after sending the triggering command.

4.3 Code Composer Studio - Main Program development

In this section the development of the main program will be explained.
The complexity of the integration of the BLE stack made it impossible to create a new project "from scratch", therefore the most desirable way to make it was to start from *ProjectZero*, an example provided by TI which includes BLE stack and task organization.
The resulting code had then to be massively cleaned up from all the unnecessary services and GPIO initialization so to make it compliant to the needs.
The program, too large to be analysed entirely, will be partitioned by functional blocks in the following part.

4.3.1 ProjectZero - Main Procedures

ProjectZero_taskFxn()

The application task entry point. Invoked by TI-RTOS when BIOS_start is called. Calls an init function and enters an infinite loop waiting for messages.

Messages can be sent either directly from the BLE stack or from user code like Hardware Interrupt (Hwi) or a callback function.
The reason for sending messages to this task is that some RTOS and Stack APIs are not available in callbacks and so the actions that may need to be taken is dispatched to this Task.

ProjectZero_init()

Called before the task loop and contains application-specific initialization of the BLE stack, hardware setup, power-state notification (if used), and BLE profile/service initialization.
GPIO initialization follows the specification declared in *"Board.h"*, where the PINs intended to be used with I2C, SPI or simply as LED should be defined.

user_processApplicationMessage()

Handle application messages that belong to the application itself, not to the BLE stack.
For example, in a Software Interrupt (Swi) it is not possible to call any BLE APIs, so the Swi function must send a message to the application Task for processing in Task context.

4.3.2 Sensor Controller interfacing

The code shown in subsection 4.2.1 must be included in the main program in order to make it work.
To do this SCIF (Sensor Controller InterFace) drivers - generated by the Sensor Controller Project - have to be integrated in the *project_zero.c* source file.
Most of the sensor controller registers are memory-mapped and are available for the system CPU to read or write. These are found in the `AUX_SCE:FETCHSTAT` and the `AUX_SCE:CPUSTAT` registers and can be easily accessed by scifTaskData structure.

The first step is to include *"scif.h"* generated source file where all the constants and variables created in SCS are defined.
Next SCIF driver has to be initialized by calling the three functions below:

```
1 scifOsalInit();
2 scifOsalRegisterCtrlReadyCallback(scCtrlReadyCallback);
3 scifOsalRegisterTaskAlertCallback(scTaskAlertCallback);
```

Where `scCtrlReadyCallback` and `scTaskAlertCallback` are two callbacks that must be defined in *project_zero.c* to manage data transfer between coprocessor and main core.

```
1 scifStartRtcTicksNow(0x00010000 / N);
```

The function above is used to wake up a Sensor Controller task with a frequency equal to NHz. In such a way the task is repeated continuously every $1/N$ seconds. Actually this command is related to SCS routine `fwScheduleTask(M)`, so that the specified task is ran only after M pulses, corresponding to M/N seconds.

In order to trigger the *Initialization Code* the function `scifStartTasksNbl` must be used, giving as a parameter a 32bit code where the i-th bit is asserted according to the involved task.

`scifTaskData` is a data structure containing all the task's variables easily accessible using `scifTaskData.task_name.var_type.var_name`. Depending on the variable type it can have different R/W permissions.

When SC generates the interrupt for the main core, it enters in a function - in this case it is *processTaskAlert* - where it can be checked which task has sent an ALERT (calling in SCS `fwGenAlertInterrupt`) by decoding *bvAlertEvents*, a 32bit array structured as the one explained above.

In this project the execution of each task loop has been managed by running from *project_zero.c* the `scifSwTriggerExecutionCodeNbl`function, having as a parameter the usual 32bit code to identify a task.

4.3.3 Bluetooth Services

Bluetooth Introduction

BLE communication works over-the-air according to *Attirbute Protocol (ATT)*, actually it is more common to talk about *Generic Attibute Protocol (GATT)* which can be seen as a meta-layer of ATT.

Attribute is the smallest addressable unit of data used by ATT and GATT and consists of:

- Handle – The 'address' of the Attribute when accessed via the Attribute Protocol (16 bits)
- UUID – The 'type' of the Attribute (16 or 128 bits)
- Value – Array of bytes interpreted differently depending on the UUID (1 to 512 bytes).

Even though the max length of an attribute value is logically 512 bytes, the Maximum Transmission Unit (MTU) size for ATT can be lower - minimum 27 bytes, not accounting for L2CAP and ATT command header. If the length exceeds this, a GATT *ReadLongCharValue* or GATT *WriteLongCharValue*, which splits the data up, must be performed.

In order to use this Protocols, *Bluetooth SIG* has defined several **Profiles**, text documents specifying how to use these protocols to achieve a certain behaviour.

The Hierarchy of BLE protocol then is:

$$Profile \rightarrow Service \rightarrow Characteristic \rightarrow Value$$

- **Characteristic** has at least one *Value* and a *Declaration* which contains R/W permissions of the value attribute, the *UUID* of the characteristic and its *Handle*. A *Descriptor* is provided to describe other attributes of the characteristic such as Notifications about value changes.
  The UUID is needed, from the smartphone side, in order to know how the value of the Attribute should be interpreted. It must be 128 bit long in order to avoid collisions with SIG patented characteristics.

- **Service** is a collection of characteristics. It contains every attribute from the declaration of a service until the declaration of another service, ordered by attribute handle.
  In light of this, the Profile defines a collection of one or more services and how services can be used to enable an application or use case. Each Service is identified by a UUID, which can be 16 bit or 128 bit long with no restrictions.

Profiles Creation

In order to make the application run over Bluetooth, a new ad-hoc profile has to be generated.
Bluetooth SIG used to provide a software called *"Bluetooth Developer Studio"* which, however, is now discontinued. Therefore, Texas Instruments created an online tool including a light version of that program represented in Figure 4.4.

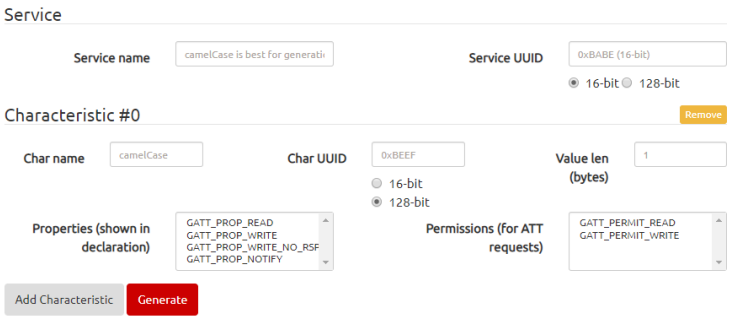


Figure 4.4. BLE Profile Generator

In this project a service containing 5 characteristics has been created:

DESCRIPTION	NAME	UUID	LENGTH	PROPERTIES
Configuration	BATMAN_CFGCHAR	0xCFCA	2	GATT_PROP_READ | GATT_PROP_WRITE
Voltage	BATMAN_VOLTAGECHAR	0xDEAD	60	GATT_PROP_READ | GATT_PROP_NOTIFY
Current	BATMAN_SHUNTCHAR	0xBEEF	60	GATT_PROP_READ | GATT_PROP_NOTIFY
Hood TH	BATMAN_HOODTHCHAR	0x00DC	4	GATT_PROP_READ | GATT_PROP_NOTIFY
Board TH	BATMAN_BOARDTHCHAR	0xB0AD	4	GATT_PROP_READ | GATT_PROP_NOTIFY

Table 4.3. Characteristic of BatMan Service (UUID: 0xFEDA)

Here the Configuration Characteristic is used in both directions: the application, on its side, can set the Configuration of the whole device and send an acknowledge of successful connection, on the other side the microcontroller can send messages to the paired device through this characteristic.
All the other characteristics are simply needed to exchange data between the device and a connected smartphone, therefore only READ property should be set. Actually it is very useful, in this case, to also activate NOTIFY property in order to wake up the Android App when a new value is being sent.

The tool in Figure 4.4 then generates two C files - corresponding to header (.h) and implementation (.c) files - to be included in the final project. Characteristic length and UUID can be modified at any time by simply replacing the value in the header file with the desired one.

BLE Integration

The first step is to initialize the readable characteristic by sending a placeholder variable through the SetParameter routine which gets as parameters Characteristic name, length and the pointer to the value to be sent.

```
1
2 uint8_t someVal[20] = {0};
3
4 // Initalization of characteristics in BatMan that are readable.
5 BatMan_SetParameter(BATMAN_CFGCHAR, BATMAN_CFGCHAR_LEN, &someVal);
6 BatMan_SetParameter(BATMAN_VOLTAGECHAR, BATMAN_VOLTAGECHAR_LEN, &someVal);
7 BatMan_SetParameter(BATMAN_SHUNTCHAR, BATMAN_SHUNTCHAR_LEN, &someVal);
8 BatMan_SetParameter(BATMAN_HOODTHCHAR, BATMAN_HOODTHCHAR_LEN, &someVal);
9 BatMan_SetParameter(BATMAN_BOARDTHCHAR, BATMAN_BOARDTHCHAR_LEN, &someVal);
```

The same function should be called when a transmission has to be performed; in this project the data to be sent are packed in order to send more than a single shunt sample and both humidity and temperature as a single characteristic value as in the code shown below.

```
1 uint16_t hood_temp, hood_RH, hood[2];
2 uint16_t ShuntVoltage[BATMAN_SHUNTCHAR_LEN/2];
3
4 /* ----------------------ACQUIRE N/2 SHUNT VOLTAGE BYTES TO BE SENT---------------------- */
5 for(n=0; n<BATMAN_SHUNTCHAR_LEN/2; n++){
6   ShuntVoltage[n]   = scifTaskData.ina226.output.mVShuntVoltage[n];
7 }
8
9 BatMan_SetParameter(BATMAN_SHUNTCHAR, BATMAN_SHUNTCHAR_LEN, ShuntVoltage);
10
11 /* ----------------------ACQUIRE TH DATA AND COMBINE INTO AN ARRAY---------------------- */
12 hood_RH = scifTaskData.hdc1080.output.humidity;
13 hood_temp = scifTaskData.hdc1080.output.temperature;
14
15 hood[0] = hood_RH;//   & 0xFF;
16 hood[1] = hood_temp;// & 0xFF;
17
18 BatMan_SetParameter(BATMAN_HOODTHCHAR, BATMAN_HOODTHCHAR_LEN, hood);
```

The need to transmit large packets of data is mostly due to BLE power consumption; every time the μC performs a Bluetooth operation it sinks $\approx 10mA$, which makes the device having a doubled consumption. Moreover the smartphone battery is highly stressed when receiving data over Bluetooth so it is better to exploit the whole payload size (512 bytes) of the GATT in order to minimize the number of connections.

4.3.4 Flash Memory

SPI Communication

As explained in subsection 3.1.5 the need of using a flash memory arose with the aim of storing data when the Bluetooth connection is not established.
Initially, the idea was to use Sensor Controller to handle this SPI peripheral; unfortunately it was not possible due to the limited resources of this coprocessor. Moreover SCS doesn't allow to handle *C* libraries that are necessary because a flash memory needs many commands to work properly.

To handle an SPI peripheral directly from the main core, it is necessary to make use of the drivers needed to interface with the proper hardware peripheral of the microcontroller.
The following code allows to initialize the driver and must be performed before every SPI command is called.

```
//Initialize SPI Communication
SPI_init();
```

To open an SPI communication it is necessary to define the parameters that are the bitrate, polarization and phase, functioning mode (master or slave) and the size of the data.

```
//open SPI communication
SPI_Handle MEM_openSPI(){
SPI_Params params;
SPI_Params_init(&params); //initialization with some default values
params.bitRate      = 1000000; //12MHz is the maximum value as Master
params.frameFormat  = SPI_POL1_PHA1;
params.mode         = SPI_MASTER;
params.dataSize     = 8;
}
```

When the communication is established it is possible to send/receive data; since SPI is a full-duplex communication protocol the only command allowed is *SPI_transfer()* which can write and read simultaneously.

```
SPI_Transaction transaction;
transaction.count   = txBufLength;
transaction.txBuf   = txBuf;
transaction.rxBuf   = rxBuf;

//Write and Read
SPI_transfer(handle, &transaction);
```

After the transmission has been performed it is necessary to close the communication.

```
SPI_close(handle);
```

The described drivers were used to implement a library that allows to control the memory, the main features are:

- Read from a specific register
- Write in a specific register
- Enable erasing/writing
- Erase a section/all the main array
- Write in the main array
- Read from the main array

The full library is reported in appendix to this elaborate.

Storing Algorithm

Reading and writing from the flash memory must be handled in a proper way; when n values should be written there must be at least $n*4$ free bytes. Therefore an algorithm to erase a sector only when needed has been implemented.

However this is not sufficient to use the flash in an efficient way, condition that would imply writing data on it when a smartphone is not connected and transferring them all to the app when pairing is established.
This algorithm was not implemented due to time reasons, therefore represents a future perspective (section 8.2).

Chapter 5

Android App

The final step of the Implementation part is the Android App development, to let the user interface with the device via Bluetooth. The application main tasks are to graph Voltage, Current and both Hood and Board TH data and simultaneously save them into a properly formatted file.

The application has been entirely developed using Android Studio IDE to achieve an improved reliability of the dataflow.

5.1 Start Activity

The activity's layout was written in XML in the file *"activity_start.xml"* which is linked to the Java management code in *"StartActivity.java"* file through the command `setContentView(R.layout.activity_start);`.

The Start screen in Figure 5.1 appears when the device is ready to be connected but the phone is still unpaired. The first step before connecting to the device and start the acquisition is to fill the blank spaces in order to keep trace of the battery and car used for that sampling campaign, thus having an unambiguous reference.

From this window it is also possible to specify the name of the folder where the files have to be created and the path to that folder.

When everything is properly set up, the acquisition begins by hitting the START button at the bottom.

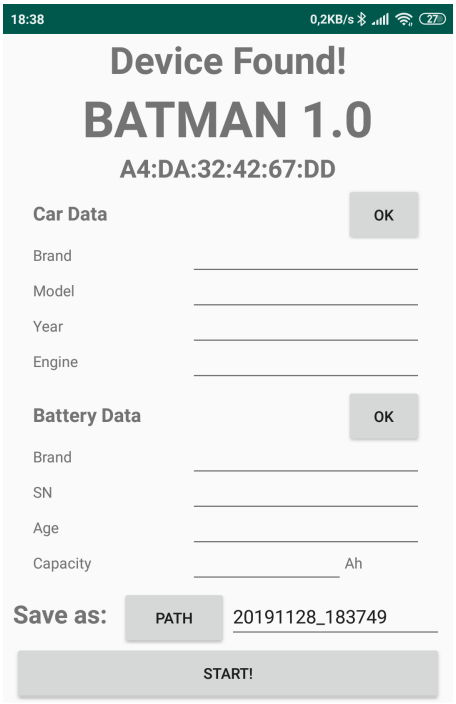


Figure 5.1. Android App Start Screen

5.1.1 Ask for Permission

In order to use Bluetooth and store data in a file, the app should be allowed by the user to use some specific resources.

The first thing to do is to check whether BLE is supported or not by the target device by simply interrogating it:

```
1 if (!getPackageManager().hasSystemFeature(PackageManager.FEATURE_BLUETOOTH_LE)) {
2   Toast.makeText(this, "BLE not supported", Toast.LENGTH_SHORT).show();
3   finish();
4 }
```

Then, by creating a *BluetoothManager* object, the user has to be requested to activate Bluetooth - if not already active - and to allows access to its location in order to be able to use Bluetooth connection. To do this a banner, corresponding to the `StartActivityForResult`code, asking to give the permission will pop-up.

```
if(ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_COARSE_LOCATION) !=
    PackageManager.PERMISSION_GRANTED) {
  Log.w("BleActivity", "Location access not granted!");
  ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.ACCESS_COARSE_LOCATION},
      MY_PERMISSION_RESPONSE);
}

// Initializes a Bluetooth adapter.  For API level 18 and above, get a reference to
// BluetoothAdapter through BluetoothManager.
BluetoothManager bluetoothManager = (BluetoothManager) getSystemService(Context.BLUETOOTH_SERVICE);
mBluetoothAdapter = bluetoothManager.getAdapter();
if (mBluetoothAdapter == null || !mBluetoothAdapter.isEnabled()) {
  Intent intent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
  startActivityForResult(intent, REQUEST_ENABLE_BT);
}
```

The key feature of the app is the capability of saving the whole collected data into a file, in such a way a database can be created with all the information from a certain battery. Actually, to do this, the user must grant the permission to write on the device; this is done by the following code

```
if (ContextCompat.checkSelfPermission(StartActivity.this, Manifest.permission.
    WRITE_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED)
{
  ActivityCompat.requestPermissions(StartActivity.this,
  new String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE}, MY_PERMISSIONS_REQUEST_WRITE_STORAGE);
}
mExternalStorageAvailable = false;
mExternalStorageWriteable = false;
String state = Environment.getExternalStorageState();

if (Environment.MEDIA_MOUNTED.equals(state)) {
  // We can read and write the media
  mExternalStorageAvailable = mExternalStorageWriteable = true;
} else if (Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
  // We can only read the media
  mExternalStorageAvailable = true;
  mExternalStorageWriteable = false;
} else {
  // Something else is wrong. It may be one of many other states, but all we need
  //  to know is we can neither read nor write
  mExternalStorageAvailable = mExternalStorageWriteable = false;
}
```

5.1.2 Wait for Available Device

After all the permissions have been granted, BLE scan can be started by the function `startBleScan()` which looks for the device for a `SCAN_PERIOD` equal to $10000ms$ ($10s$), then it stops searching.

By analysing the `onScanResult` callback, it is possible to check if the device to be connected is available. In this case, when the device name contains the string `"BATMAN"`, the page will show the user the guided procedure to start the acquisition.

```
private void startBleScan() {
scanner=mBluetoothAdapter.getBluetoothLeScanner();
mScanning = true;
AsyncTask.execute( scanner.startScan(leScanCallback); );
mHandler.postDelayed({
  scanner.stopScan(leScanCallback);
  mScanning = false;
  }, SCAN_PERIOD);
}

public ScanCallback leScanCallback = onScanResult(int callbackType, result) {
  mDevice = result.getDevice();
  if (mDevice.getName() != null && mDevice.getName().contains("BATMAN") {
    if(scanner != null) {
      scanner.stopScan(leScanCallback);
    }
    if (mDevice == null) {
      Log.d(TAG, "onScanResult: device not found");
    } else {
      mDeviceAddress = mDevice.getAddress();
      mDeviceName = mDevice.getName();
      devices.setText("Device Found!");
      devName.setText(mDeviceName);
      devAdd.setText(mDeviceAddress);
      devLayout.setVisibility(View.VISIBLE);
      carLayout.setVisibility(View.VISIBLE);
    }
  }
};
```

5.1.3 Connect

When all the parameters are set, by hitting the start button the following code will be executed:

```
startButton.setOnClickListener({
  final boolean err_solve = mBluetoothLeService.connect(mDeviceAddress);
  Log.d(TAG, "Connect request result=" + err_solve);
  Toast.makeText(StartActivity.this, "CONNECTED TO " + mDeviceName, Toast.LENGTH_SHORT).show();
  String cardesc = "CAR "+ carBrand.getText() + " " + carModel.getText() + " " + carYear.getText()
     + " " + carEngine.getText() + "\n";
  String batdesc = "BATTERY "+ batBrand.getText() + " " + batModel.getText() + " " + batAge.getText
     () + "y " + batCap.getText() + "Ah\n";
  SaveText("BatManApp/", fileName+".txt", cardesc);
  SaveText("BatManApp/", fileName+".txt", batdesc);
  startActivity(new Intent(StartActivity.this, GraphActivity.class));
  }
});
```

The device tries to connect to *BAT-MAN* by mean of its MAC address calling the `connect` routine from *BluetoothLeService* service which interacts with the BLE device via the Android BLE API.

The car and battery parameter are made up into `cardesc` and `batdesc` and printed to the first two lines of the *"filename.txt"* file.

By using the `startActivity(callingActivity, destinationActivity)` command, the page (android activity) will switch to the one able to graph the collected data.

5.2 Graph Activity

As for the previous activity, this one is structured into two main files; a layout one *"activity_graph.xml"* and a management one *"GraphActivity.java"*

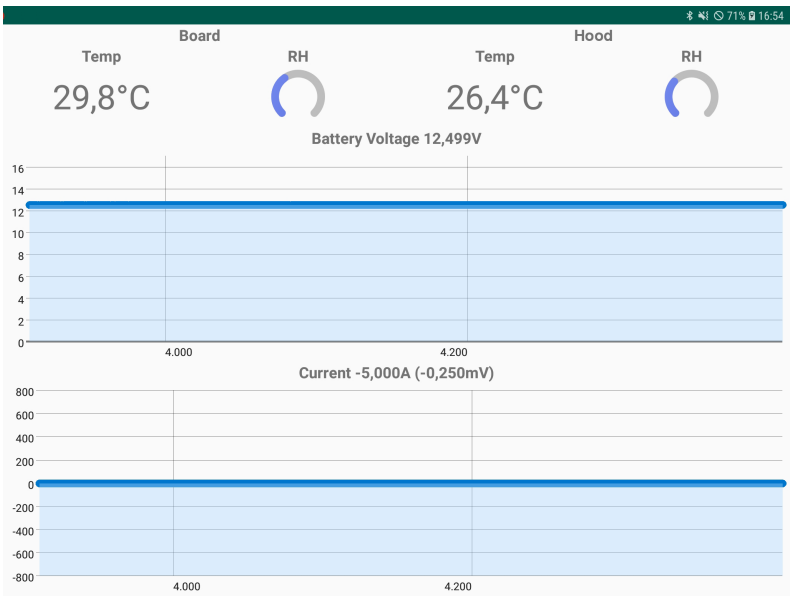


Figure 5.2. Android App Graph Screen (charging)

5.2.1 Graphs

In order to graph the collected data an external class - **GraphView** by *jjoe64* - has been used.

As it can be seen from Figure 5.2, there are two graphs, the upper plots the battery voltage on *"series1"*, the bottom one plots the current sank or supplied by the battery on *"series2"*.
The graph should have a different layout depending on the type of data; Voltage range to be plotted is between 0V and 17V (a value which cannot be reached) while current should have a greater range because, during cranking, it can go up to 800A, therefore a range [-800A;800A] has been chosen.

```
protected void GraphSettings( ) {
  Vbus_tw.setText("Battery Voltage");
  viewport1 = VbusGraph.getViewport();
  viewport1.setYAxisBoundsManual(true);
  viewport1.setMinY(0);
  viewport1.setMaxY(17);
  viewport1.setXAxisBoundsManual(true);
```

```
  viewport1.setMinX(0);
  viewport1.setMaxX(500);
  series1.setDrawBackground(true);
  series1.setDrawDataPoints(true);

  Vshunt_tw.setText("Shunt Current");
  viewport2 = VShuntGraph.getViewport();
  viewport2.setYAxisBoundsManual(true);
  viewport2.setMinY(-800);
  viewport2.setMaxY(800);
  viewport2.setXAxisBoundsManual(true);
  viewport2.setMinX(0);
  viewport2.setMaxX(500);
  series2.setDrawBackground(true);
  series2.setDrawDataPoints(true);
}
```

5.2.2 Wait for Characteristic Notification

When the smartphone receive a notification by a certain characteristic, the app calls the `onCharacteristicChanged` routine that executes `broadcastUpdate` specifying the characteristic and `ACTION_DATA_AVAILABLE` as the action that will be passed to the newly created *Intent*.

By checking characteristic UUID and comparing with the known ones, it is possible to manage the different notifications with a specific code; in the code below it is only reported the current management as an example:

```
if (UUID_VSHUNT_CHAR.equals(characteristic.getUuid())) {
  int Vshunt_sample;;
  Calendar now = Calendar.getInstance();
  SimpleDateFormat dateFormat = new SimpleDateFormat("HHmmssSSS", Locale.US);
  Date date = new Date();
  String timestamp = dateFormat.format(date);

  for (int i=0; i<SampleGattAttributes.PAYLOAD_SIZE; i+=2) {
    GraphActivity.vVshunt[i/2].timestamp = new Date();
    GraphActivity.vVshunt[i/2].value = (int) characteristic.getIntValue(BluetoothGattCharacteristic
     .FORMAT_SINT16, i);
    String sVshunt = dateFormat.format(GraphActivity.vVshunt[i/2].timestamp) + "\t" + String.format
     ("%X", (int) GraphActivity.vVshunt[i/2].value) + "\r";
    SaveText("BatManApp/", StartActivity.fileName+".txt", "S "+sVshunt);
  }
  acq_char++;
}

...

if(acq_char==4) {         // all 4 characteristic have been read
  acq_char=0;
  sendBroadcast(intent);
}
```

The transmitted value is stored through the `charcateristic.getIntValue` procedure specifying the number format which, in this case, is a 16bit signed integer. When dealing with the Voltage and Current characteristics, the need to iterate the

characteristic read rises, therefore an offset equal to the double of the iteration step must be specified in order to scroll the array.

The generated text file rows have the following format:

- **Type**: B for battery Voltage, S for the battery Current, H for the hood temperature and P for the PCB temperature
- **Date**: in the format "HHmmssSSS"
- **Value**: in HEX format in order to reduce the file size; the maximum representable signed number on 16bit is 32768 and - if written on a file - its size is 5byte because every ASCII character is coded on 8bit, looking at the negative number -32767 it is even worse being written on 6byte. If an HEX formatting is used a maximum of 4byte will be occupied if a 16bit value is written.

Once all four characteristics are stored, the *"acq_char"* variable is set to 0 and a new broadcast is sent to the *GraphActivity* together with the created Intent.

5.2.3 Graph Update

Data from characteristic are stored in global variables inside *GraphActivity* class.

The broadcast sent by the Service in the above steps is picked up by this class and the corresponding action is extracted. If it corresponds to the previously declared `ACTION_DATA_AVAILABLE`, then the layout must be updated as shown below.

```
1  private final BroadcastReceiver mGattUpdateReceiver = new BroadcastReceiver(Context context, Intent
       intent) {
2    final String action = intent.getAction();
3    if (BluetoothLeService.ACTION_DATA_AVAILABLE.equals(action)) {
4      updateTH();
5      updateGraphs();
6      tx_packets++;
7    }
8  };
9
10 private void updateTH() {
11   double temp = (vHoodT.value*165)/Math.pow(2,16) - 40;
12   double hum  = (vHoodH.value*100)/Math.pow(2,16);
13
14   HoodHGauge.setValue((int) hum);                // update Humidity Gauge
15   HoodT_tw.setText(String.format("\%.1f", temp)+"°C");   // update Temperature
16
17   temp = (vBoardT.value*165)/Math.pow(2,16) - 40;
18   hum  = (vBoardH.value*100)/Math.pow(2,16);
19   BoardHGauge.setValue((int) hum);               // update Humidity Gauge
20   BoardT_tw.setText(String.format("\%.1f", temp)+"°C");  // update Temperature
21 }
22
23 private void updateGraphs() {
24   double Current;
```

```
25    double Vbus;
26    double Vshunt;
27
28    for (int i=0; i<SampleGattAttributes.PAYLOAD_SIZE/2; i++) {
29      Vbus = vVbus[i].value*1.25/1000 + 0.15;
30      Vshunt = vVshunt[i].value*2.5/1000;
31      Current = vVshunt[i].value/ 20.0;
32
33      Vbus_tw.setText("Battery Voltage "+ String.format("\%.3f", Vbus)+"V");
34      Vshunt_tw.setText("Current " + String.format("\%.3f", Current) + "A (" +  String.format("\%.3f"
         , Vshunt)+"mV)");
35      series1.appendData(new DataPoint(i+tx_packets*SampleGattAttributes.PAYLOAD_SIZE/2, Vbus), true,
          500);
36      series2.appendData(new DataPoint(i+tx_packets*SampleGattAttributes.PAYLOAD_SIZE/2, Current),
         true, 500);
37    }
38  }
```

Part III

Testing

Chapter 6

Testing

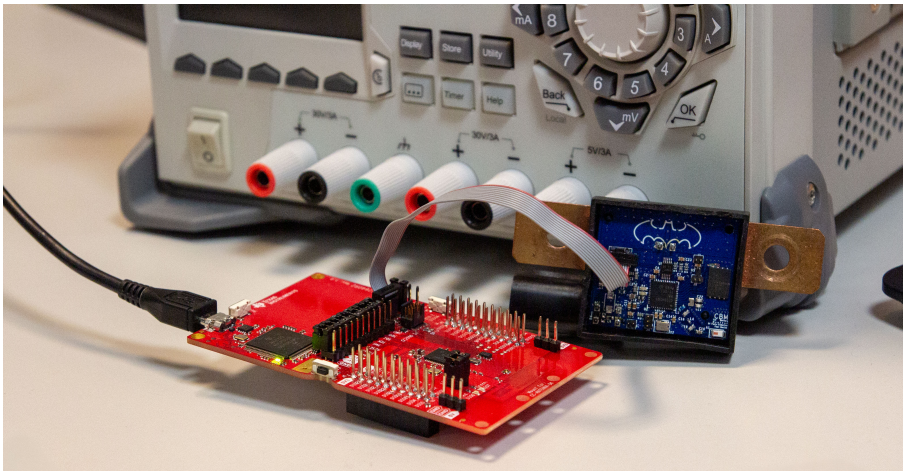


Figure 6.1. Flash/Debug connection to the Launchpad

To understand if both hardware and firmware work properly it was necessary to perform many tests. In order to flash the firmware and to debug the code, a *LaunchPad* board was used.
This device also includes a development board with a CC2640R2F microcontroller installed on it, allowing to test the firmware on it. This board allowed to test the firmware before the production of the PCB, accelerating its development and making the isolation of firmware and hardware failures possible.

All the data used for the plots in this chapter have been acquired by an oscilloscope, the *Tektronix MDO3104*.

6.1 Buck voltage

Ideally, it is expected to detect a continuous voltage with the value of 3.3V. Actually the supply voltage presents some spikes because the adopted regulator is a buck converter, implying an intrinsic fluctuation due to its switching structure.

The worst possible case is represented by the maximum current absorption, condition only reached when erasing or programming the external Flash Memory sinking $40mA$ for a short time causing a current impulse. Therefore, by using AC coupling mode, a magnification of this effect has been represented in Figure 6.2. However, the maximum amplitude of this fluctuations is of 60mV in positive and 30mV beyond the nominal value. This behaviour appears not to be so relevant, extending the buck output range from 3.27V to 3.36V.

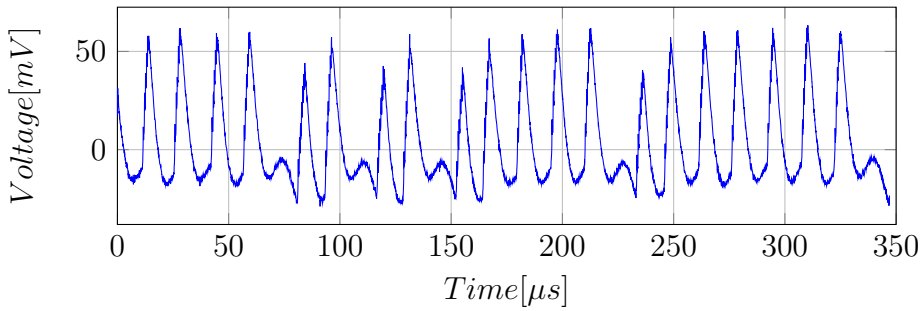


Figure 6.2. Supply Voltage fluctuations when sinking impulsive 40mA.

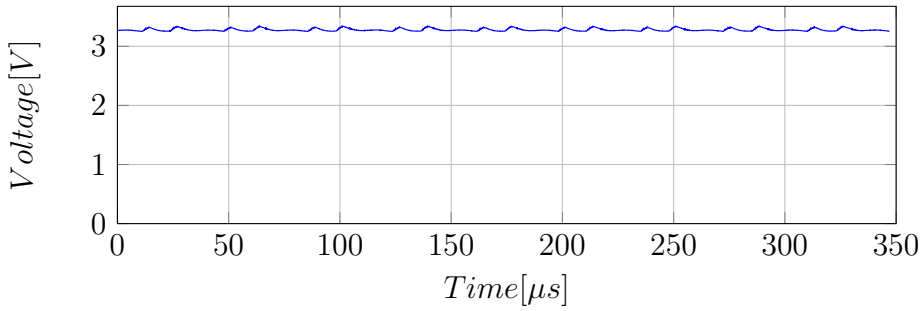


Figure 6.3. Supply Voltage behaviour when sinking impulsive 40mA.

6.2 XTAL

The main way to detect if the microcontroller is properly working or not is to observe the oscillations at 24MHz and at 32.768kHz. In fact, in debug mode, the microcontroller executes the program also if the clocks are not working; probably the timing is provided by the LaunchPad board.

When everything works properly it is expected to observe two sinusoids like in the following plots.

Observing the clock is also a possible way to understand if the μC enters in shutdown mode, however this mode is not currently implemented in the firmware.

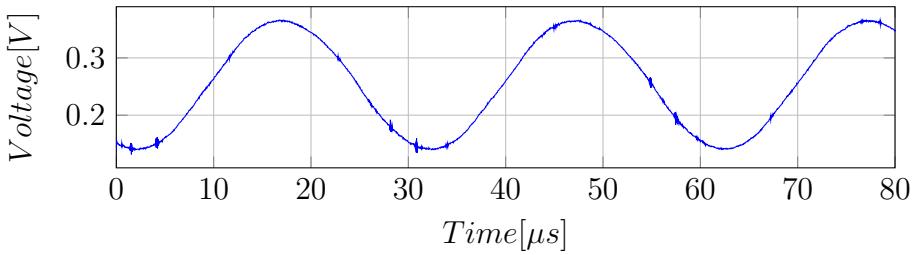


Figure 6.4. 32kHz XTAL

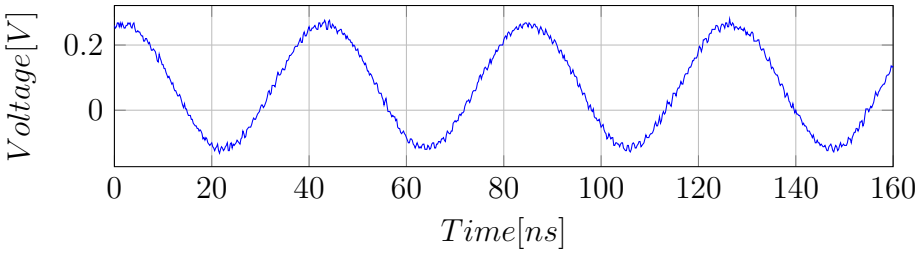


Figure 6.5. 24MHz XTAL

6.3 I2C debugging

In all of the plots in Figure 6.6, 6.7 and 6.8 there are several high spikes on SDA signal; these are due to the acknowledge made by the slave.
Taking into consideration Figure 6.6, the transmission starts by putting low SDA when SCL is active in the point marked as ST. This is the first part of the execution code, therefore Sensor Controller sends INA226 address - equal to 0x45 (1000 101) - shifted left by 1 position and followed by the $R/\overline{W}$ low, and then the slave answers by sending an acknowledgement to confirm a successful communication. When the line owner changes from the microcontroller to the slave device, SDA line is temporarily not driven by anyone therefore the pull-up resistor tends to pull SDA signal to V_{DD} until the successive "owner" will not drive the line again.

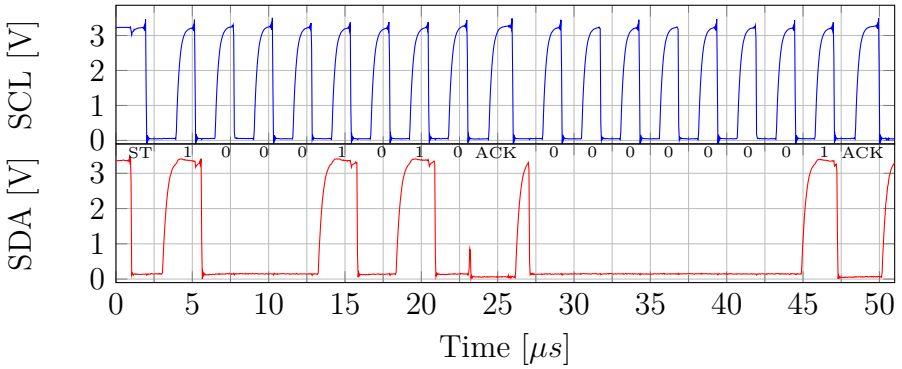


Figure 6.6. I2C communication - INA226

In Figure 6.7 the device address of HDC2010 - now equal to 0x41 - is sent and, after it, the command 0x0F which triggers the measurement is given.
In the top plot - the one representing SCL line - the clock appears to be smoother on the positive edges. In fact, by using three I2C devices on the same bus, the capacitance of the line is higher than using only one of them; the time constant of the line is $40ns$ and the clock period is equal to $2.5\mu s$. Therefore the delay is visible but it doesn't negatively affect data recognition. The actual value of the I2C pull-up resistors is $10k\Omega$, which is lower than the maximum one evaluated below:

$$R_{P,MAX} = \frac{t_{rise}}{0.8473\ C_{line}} = \frac{300ns}{0.8473 \cdot 4pF} \approx 88.5k\Omega \tag{6.1}$$

Where:

- t_{rise} is the maximum rise time of the clock when in fast mode (up to 400kHz)
- C_{line} is the capacitance value of the line which is the sum of $C_{INA226} = 3pF$, $C_{HDC2010} = 0.5pF$ and $C_{HDC1080} = 0.5pF$

- $0.8473 = \ln \frac{0.7}{0.3}$ is obtained by the exponential behaviour considering an high level of $0.7V_{DD}$ and a low level of $0.3V_{DD}$

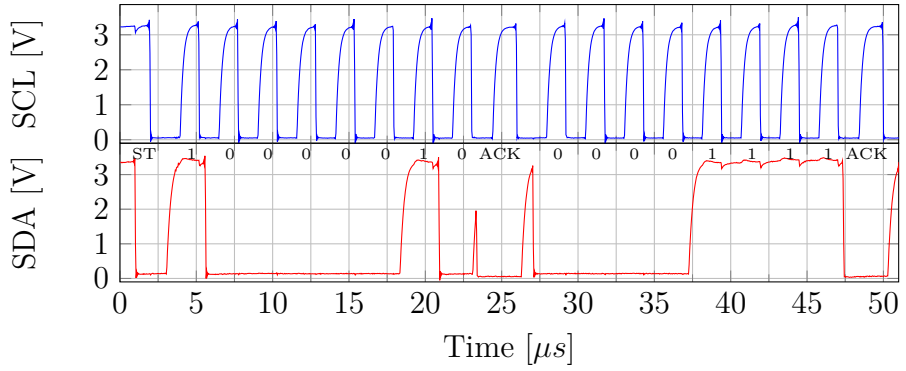


Figure 6.7. I2C communication - HDC2010

The device address of HDC1080 is 0x40, it is left shifted 1 position and combined with the $R/\overline{W}$ bit low as visible in Figure 6.8.
The first bit of the address - the first transmitted logic 1 - presents a negative spike in correspondence with the falling edge of the clock which shows the same spike. This phenomenon is due to cross-talk between SCL and SDA lines which is visible in all the reported figures.

It is also possible to recognize a different zero voltage level between the address transmission and the acknowledge bit. It seems that the three devices can pull down the line to a value closer to 0V with respect to Sensor Controller, thus suggesting a worse hardware driver.

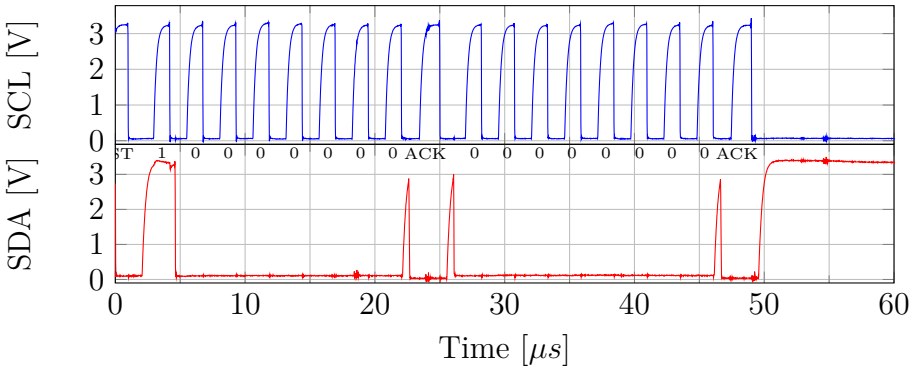


Figure 6.8. I2C communication - HDC1080

6.4 SPI debugging

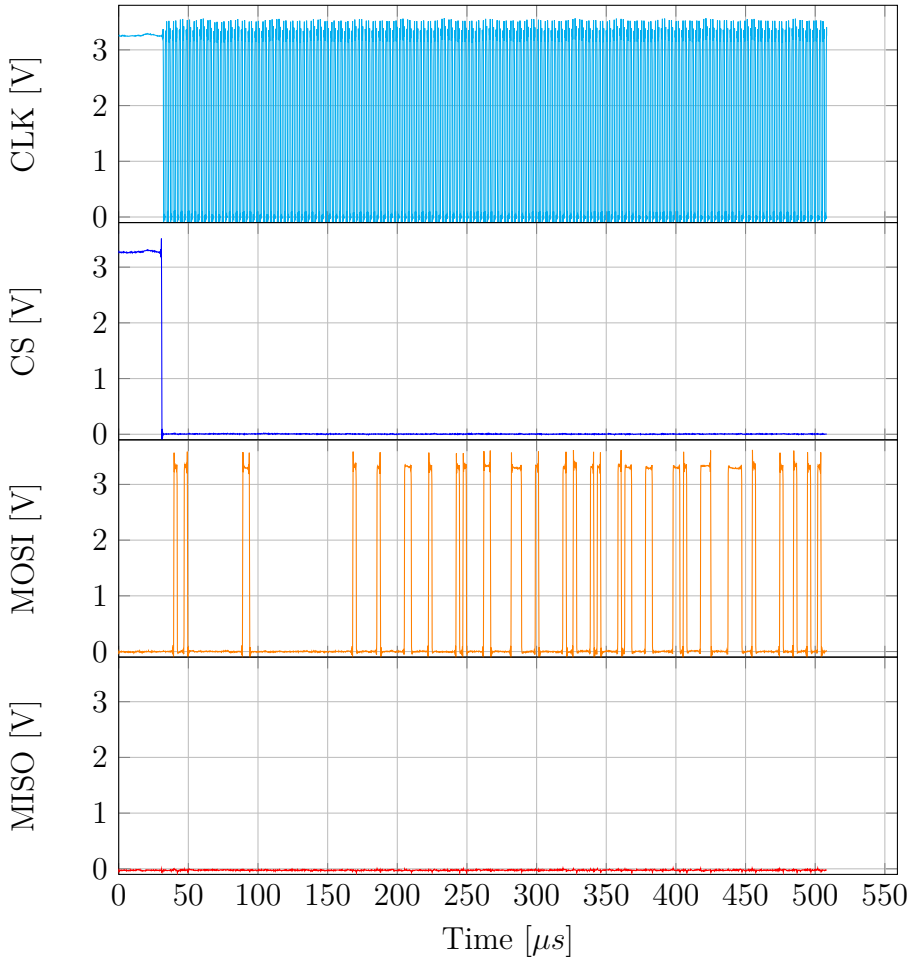


Figure 6.9. SPI communication - Flash Memory 0 to 255 write

In Figure 6.9, a write operation to the flash memory is reported. The first 8bit, corresponding to the write page (`MEM_CMD_4PP`) command, are followed by the 4 bytes initial writing address 0x00018000. Later a sequence of number (0,1,2,3...) is transmitted.

The same sequence is then sent back to the microcontroller (Figure 6.10) to check the correctness of both writing and reading operation.
In this plot the read operation is defined by sending `MEM_4READ` command (0x13) followed by the same starting address of the previous transaction.

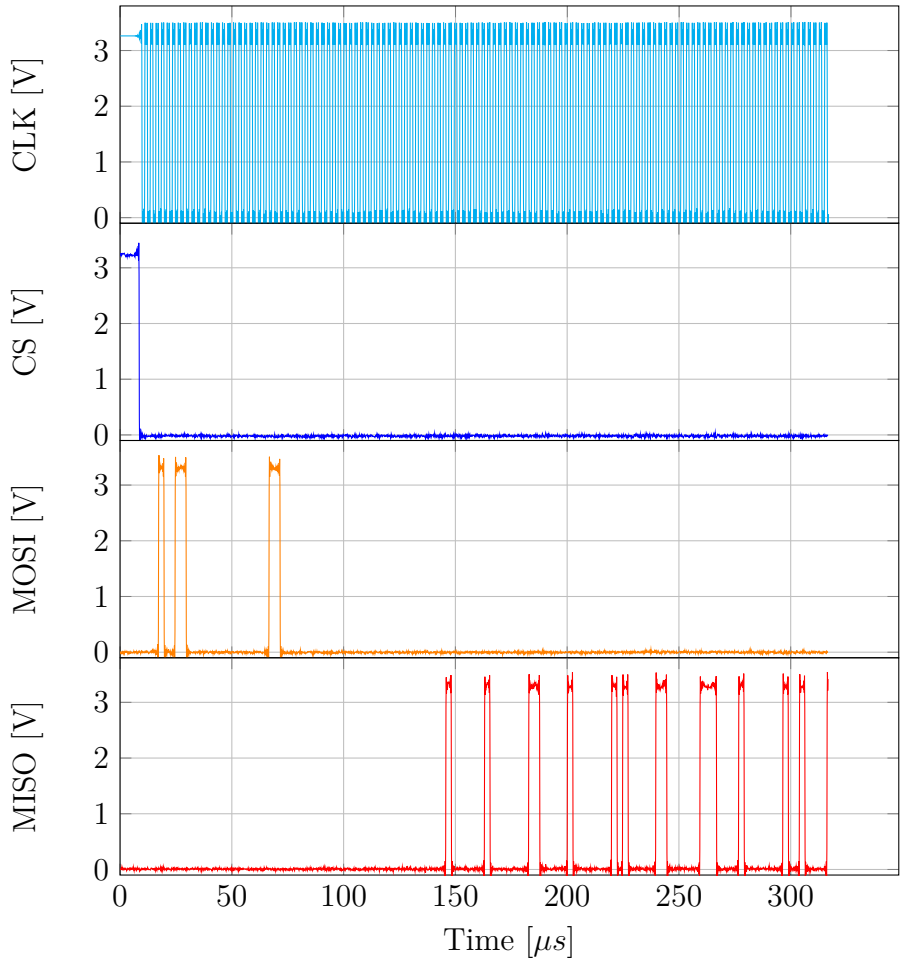
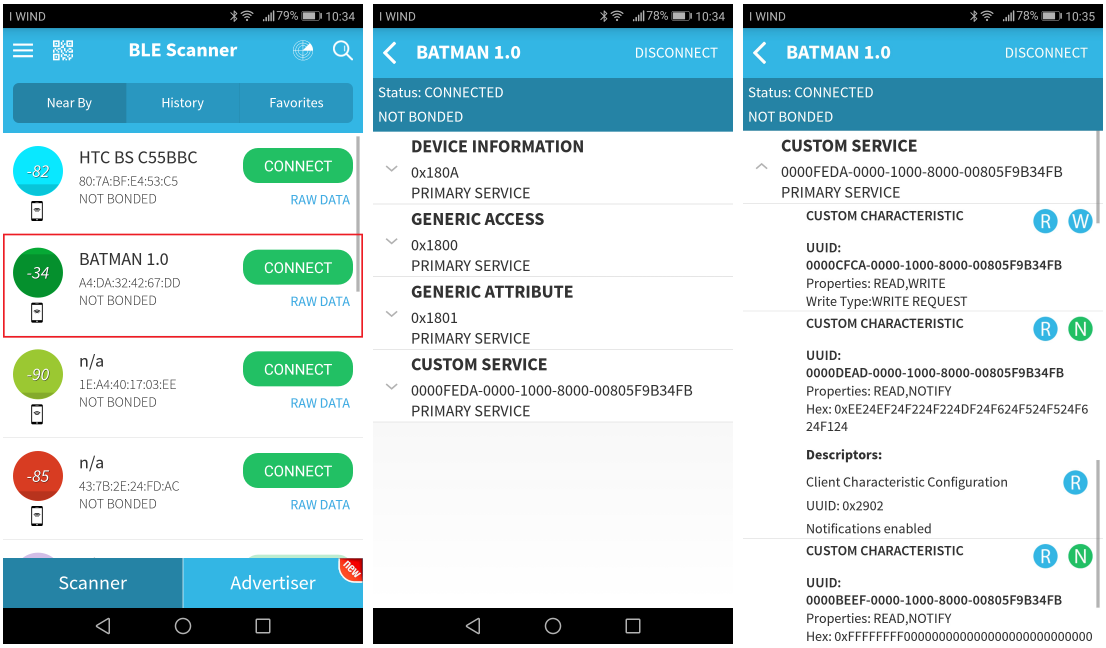


Figure 6.10. SPI communication - Flash Memory read

6.5 BLE (BLE scanner)

As suggested in Texas Instruments documentation, a possible way to communicate via Bluetooth with the board without the need of a dedicated application, is to use *BLE Scanner* app available on *Google Play Store*.
This app was used to verify proper communication establishment and data transmission at the beginning of the project when the dedicated app was not ready. Another reason to use it is to understand if an hypothetical error is due to the HW/FW or to the Android Application.

BLE Scanner allows to detect all the close Bluetooth devices giving also an indication on the signal power. By pressing *CONNECT* a connection is established and the available services are shown. It is possible to observe the characteristics inside a service having an information about read and written data shown in hexadecimal representation.



(a) Available Devices

(b) Available Services in BAT-MAN

(c) Services of Interest

Figure 6.11. BLE Scanner

Chapter 7

Verification and Validation

7.1 Laboratory Tests

In order to validate the built device several tests were performed, both in laboratory and on field, connecting the device to a battery inserted in a real car, thus evaluating its behaviour in a non-ideal environment.

Unfortunately the available measurement instruments have a maximum rating not even comparable to *BAT-MAN* 800A (1600A for 5s) range. In fact the usable multimeter - *Rigol DM3051* - can only stand up to 10A. Moreover, there were not loads capable of sinking high current without damages, so we had to sharpen our minds and find a way to do it.

Only at the end of the thesis work, an active load - TTi LD400P - was lent to the purpose.

7.1.1 Power supply current sinking

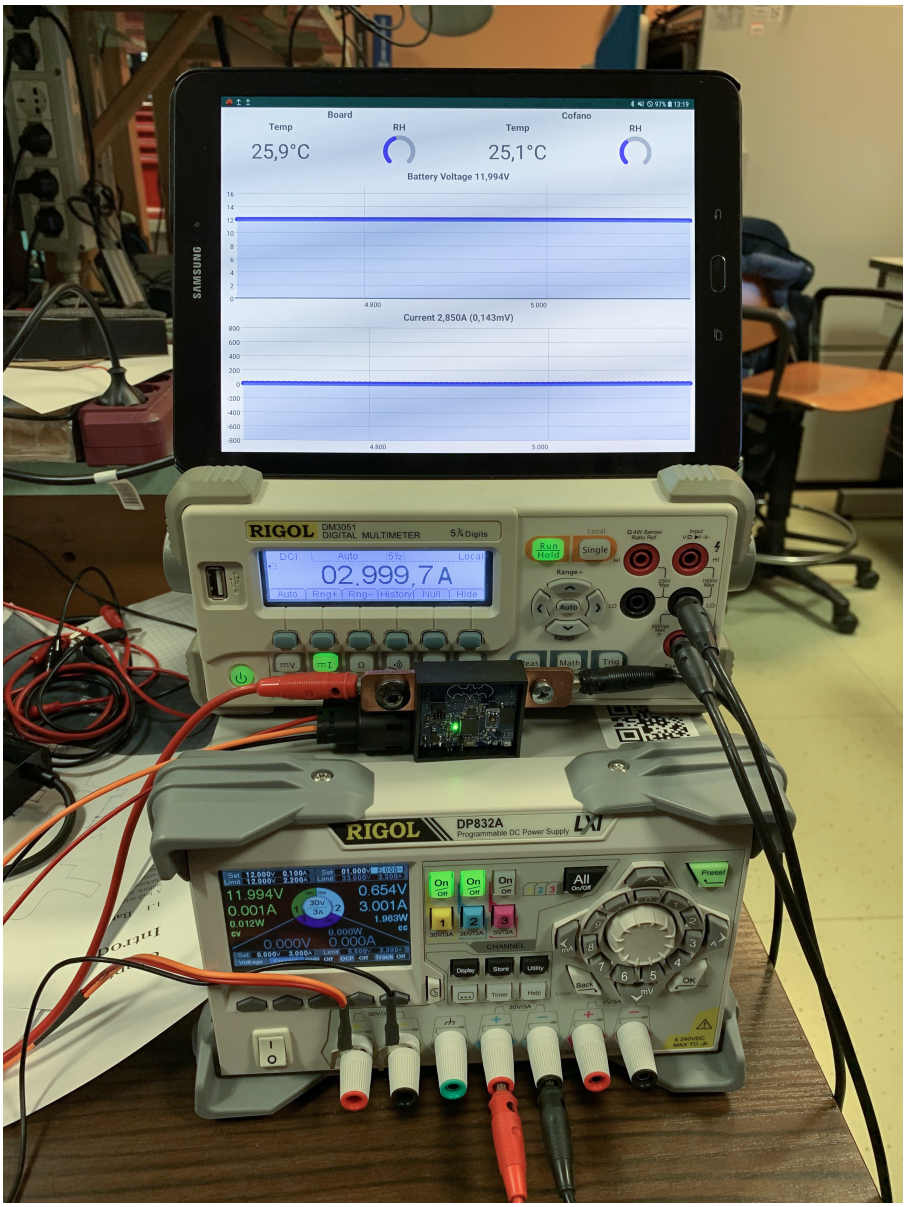
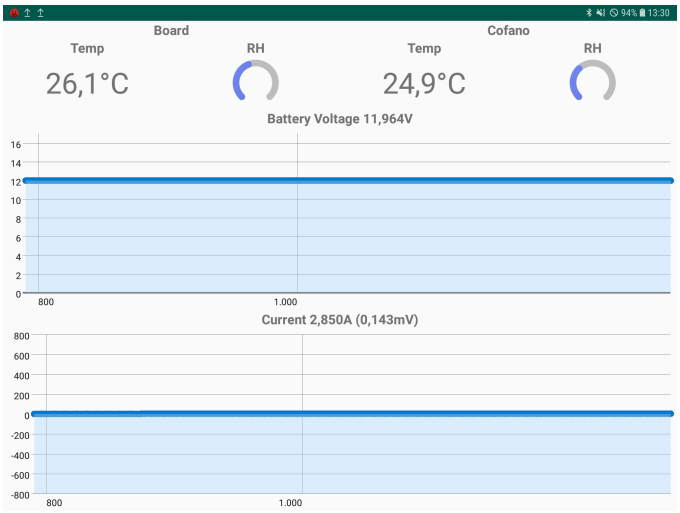


Figure 7.1. First Measurement setup

Initially, the device has been supplied with the *Rigol DP832A* using two channels; a voltage controlled one set to 12V to emulate the battery behaviour, the other one in fixed current mode at 3A.
To achieve the current generator configuration from the power supply, the shunt

resistor leads were directly connected to positive and negative terminals of the generator without inserting a load in between. In such a way - since the resistance is $50\mu\Omega$ - independently on the voltage setting, the current depends on the selected current threshold. The results are shown in Figure 7.2 where the current and voltage data are very close to the defined ones.



(a) BAT-MAN application

(b) Multimeter Screen

Figure 7.2. Rigol Power Supply Current Sinking

Nominal Value	BAT-MAN	Rigol DM3051
100 mA	100 mA	101.615 mA
500 mA	450 mA	502.20 mA
1.0 mA	900 mA	1002.16 mA
1.5 mA	1400 mA	1499.6 mA
2.0 mA	1850 mA	2000.7 mA
2500 mA	2400 mA	2500.1 mA
3000 mA	2850 mA	2999.1mA

Table 7.1. Comparison Between Current Values of the DUT and Multimeter

7.1.2 Power supply voltage source

This test has been carried out using *Rigol DP832A* DC power supply as the voltage source and comparing BAT-MAN acquisition with the values measured with *Rigol DM3051* digital multimeter. Table 7.2 shows the results for a set of voltages from 5V to 16V with a step of 0.5V.

To carry out this test the same setup in Figure 7.1 was adopted, except for the multimeter measuring voltage instead of current.

In Figure 7.3 it is possible to appreciate the residuals between the two measurement tools while on X-axis the nominal values are reported. The residuals are due to the direct voltage drop on the protection Schottky diode which is dependent on the current flowing in it and from the temperature (Figure 7.4). This logarithmic behaviour can be approximated with a straight line since just a limited current variation is involved; with lower supply voltages the current is higher in order to achieve the same power on the buck converter.
The straight line crosses the zero around 12V because the voltage to compensate the drop on the diode was measured at 12V 25°C.
By keeping into consideration this trend, it becomes possible to dynamically compensate the error thus having more realistic results.

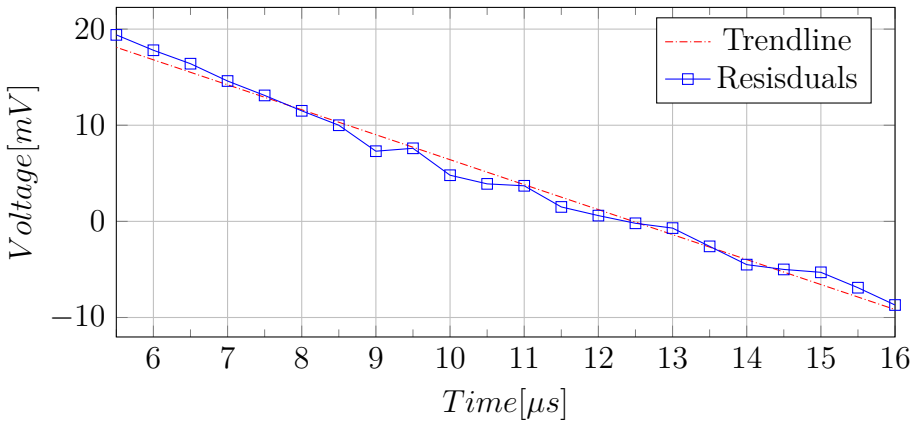


Figure 7.3. Residuals between BAT-MAN and Rigol DM3051 voltage values

Nominal Value [V]	BAT-MAN [V]	Rigol DM3051 [V]
5.5	5.482	5.5014
6	5.981	5.9988
6.5	6.481	6.4974
7	6.984	6.9986
7.5	7.487	7.5001
8	7.989	8.0005
8.5	8.488	8.498
9	8.991	8.9983
9.5	9.489	9.4966
10	9.993	9.9978
10.5	10.494	10.4979
11	10.994	10.9977
11.5	11.496	11.4975
12	11.997	11.9976
12.5	12.497	12.4968
13	13.002	13.0013
13.5	13.501	13.4984
14	14.001	13.9965
14.5	14.502	14.497
15	15.003	14.9977
15.5	15.506	15.4991
16	16.006	15.9973

Table 7.2. Comparison between BAT-MAN and Rigol DM3051 voltage values

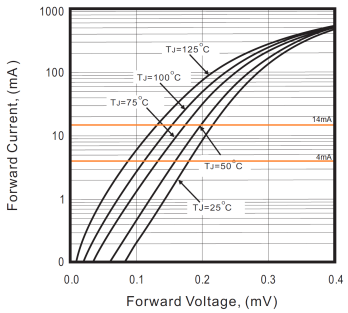


Figure 7.4. Forward Current vs Forward Voltage of the Schottky diode [33]

7.1.3 Inverter & Battery

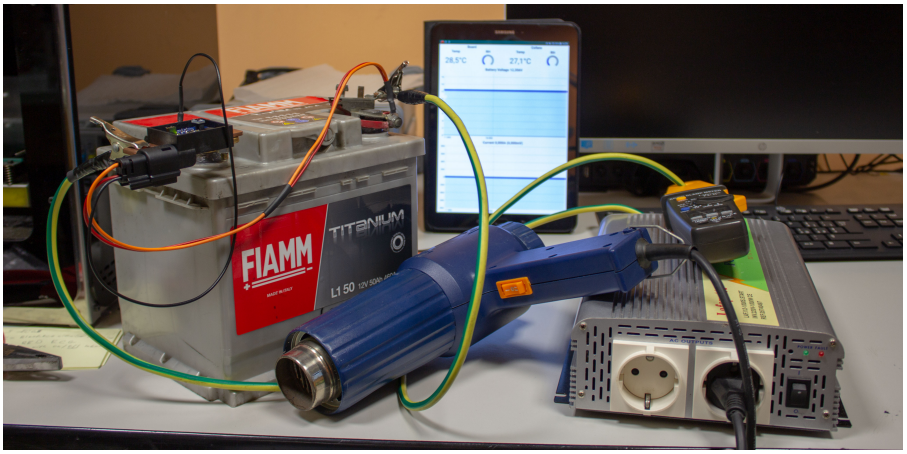


Figure 7.5. Second Measurement setup

In order to provide a relatively high current without an appropriate instrumentation, the setup in Figure 7.5 was used. It is composed of an Low-cost modified squarewave inverter (*LAF.I12-1000S.START*) and a 12V lead-acid battery.

Both the inverter and the DUT are supplied by the battery and the inverter allows to use a 220V AC hot air gun to modulate the absorbed current. In this configuration when the gun is at the maximum power, the inverter is able to only supply it for a short time before going into protection mode, probably because battery voltage drops under a internally fixed threshold.
To compare the results, a current clamp with a full-scale of 80A (*Lafayette PA-37*) was used.

From this qualitative test it is possible to observe that the device can stand without any problems current spikes as high as 120A. The measured values are coherent with those of the reference multimeter.

7.1.4 High Current Power supply & battery

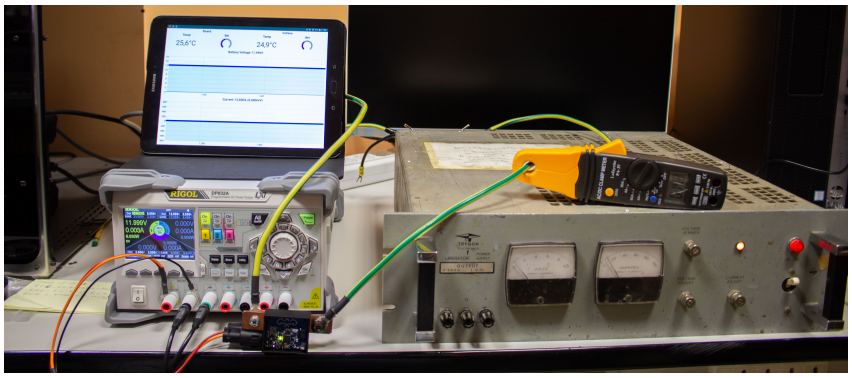
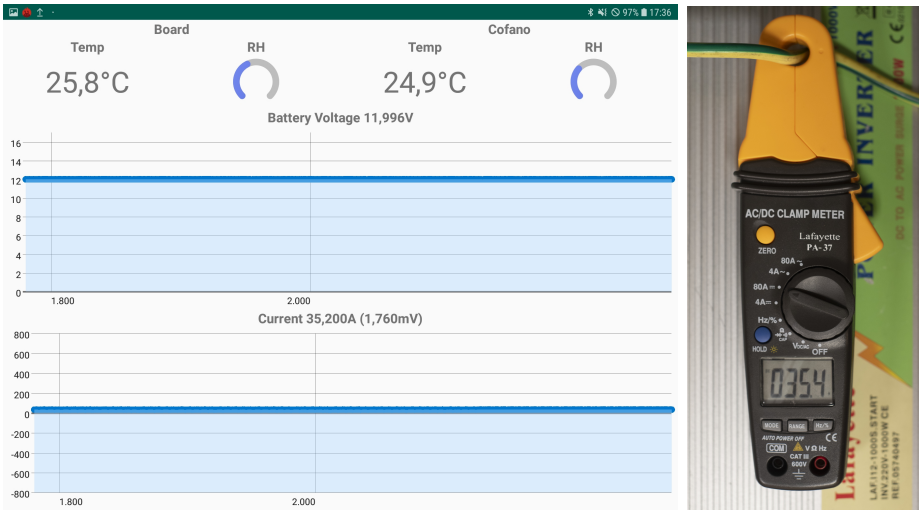


Figure 7.6. Third Measurement setup

The device used in this step is the *Trygon Electronics Liberator Power Supply 4.8V-6.8V, 0A-70A*. It is an old power supply no longer in use with the capability of providing a maximum current of 70A, actually - due to aging - it could only give as output 30A.
To use this power supply, the same configuration explained in subsection 7.1.1 was adopted.
As before the results obtained were compared with those of the Lafayette current clamp obtaining a confirmation of BAT-MAN reliability when dealing with dozens of amperes.
In Figure 7.7 the two measurements differ by 200mA, corresponding to 4 LSBs in BAT-MAN device; this result must be taken with a grain of salt because the uncertainty of the clamp meter is not provided.



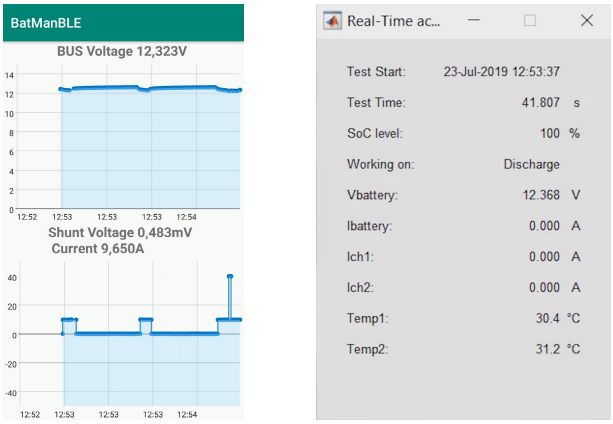
(a) BAT-MAN application (b) Clamp Screen

Figure 7.7. Trygon Power Supply Current Sinking

7.1.5 DIY Programmable Active Load

Figure 7.8. Fourth Measurement setup

For this test a programmable active load entirely designed and built by *C.Radici* and *G.Di Simone* has been used [8]. This device is able to perform a discharge sinking current pulses of 10A/20A/40A by using a MATLAB script. The heavier drawback of this device is that it can't absorb a large amount of current continuously.
However, also in this case, the obtained results can be compared with those evaluated directly by MATLAB as shown in Figure 7.10.



(a) BAT-MAN application

(b) Matlab Screen

Figure 7.9. BAT-MAN app vs DIY Active Load

In Figure 7.9 on the left, BAT-MAN app real time values and graphs are shown, on the right the MATLAB interface of the active Load.
The latter also shows temperature information not to be confused with BAT-MAN temperature. In fact the values represented by MATLAB are referred to the internal temperature of the active load which should be monitored in order to control the fan cooling system.

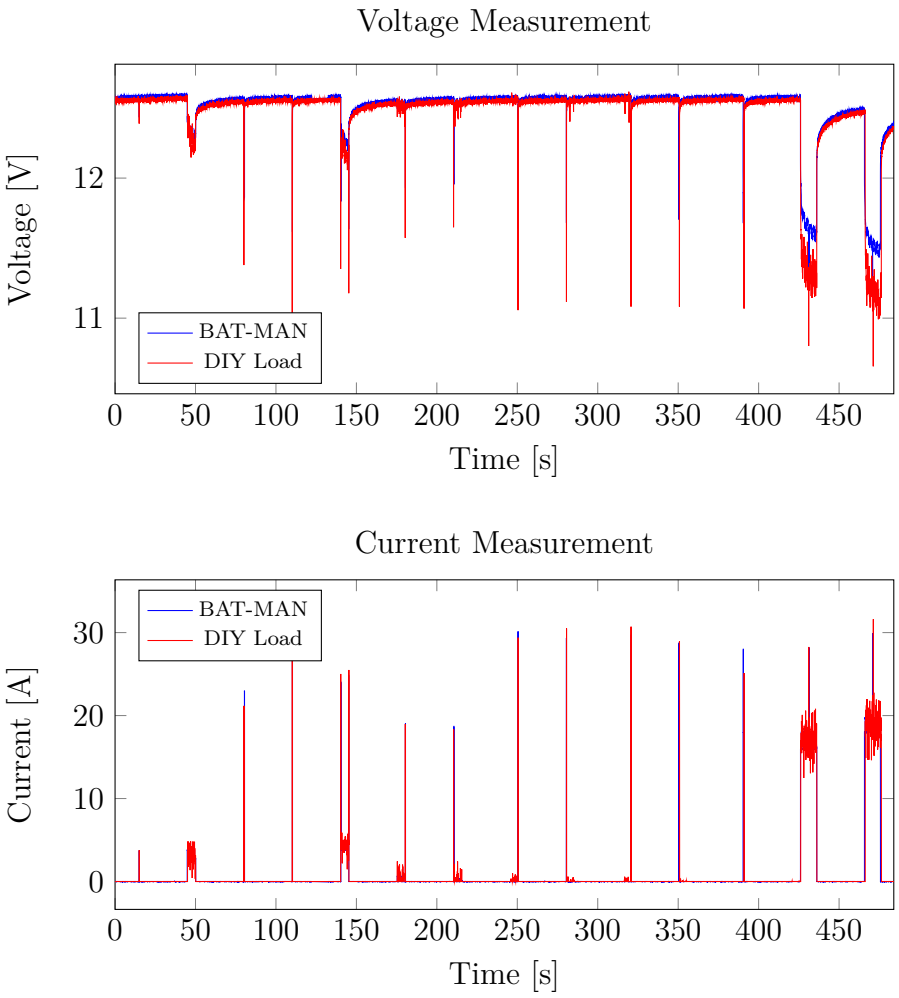


Figure 7.10. Comparison between BAT-MAN acquisition and DIY Programmable Load

In the graphs in Figure 7.10, a discharge pattern has been applied providing different current pulses to analyse the voltage response. The difference between the two plots is acceptable, but it has to be noted that the *DIY Active Load* has never been professionally calibrated so this comparison is just qualitative.

7.2 EM Emissions

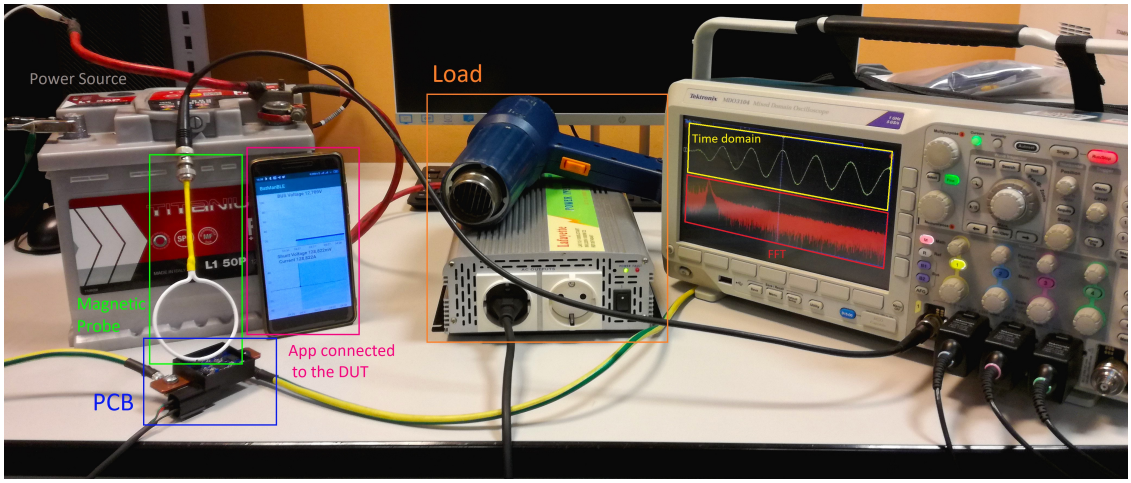


Figure 7.11. First workbench for magnetic field relevation

The aim of this part is to discover if the analysed circuit generates magnetic disturbances. This problem has been issued because the circuit measures currents as high as 800 A for 10-20 ms.
Ideally, to perform this test, a passive load should be used in order to have no disturbances deriving from the current sinker. Unfortunately, this was not possible because due tot the lack of a resistor with such a low value and high dissipation capability; so test were performed with different loads.
Since the employed active loads introduce noise, the analysis was carried out by comparing the generated magnetic field with and without the DUT.

7.2.1 First setup

The first setup is shown in Figure 7.11 and consists in:

- **Battery**, used to power on the DUT and as current source
- **Inverter and Hot Air Gun**, used as a load
- **Magnetic Probe**, connected to a 50Ω coaxial cable
- **Oscilloscope**, MDO3104 set with both time domain and FFT
- **Android App** receiving battery voltage and current values from the PCB

The probe is a single coil circular solenoid: when it is permeated by a magnetic field, a current is induced in the loop. This current, terminated on a 50Ω cap, is transduced into a voltage detectable by an oscilloscope. In order to maximise this effect the loop probe should be normal to the current flow as in Figure 7.12.

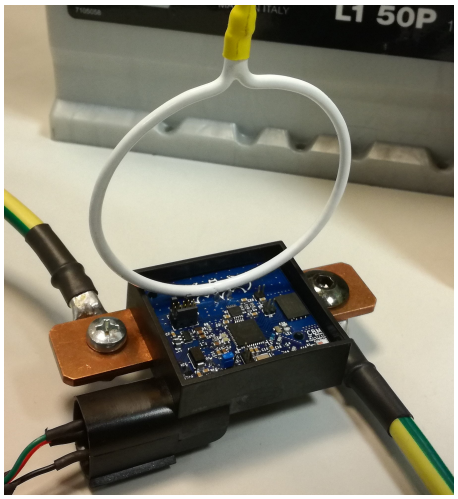


Figure 7.12. Position of the Magnetic Probe on the DUT

To simulate the current cranking due to the ignition of an internal combustion engine, the oscilloscope was triggered at the start up of the hot air gun. During this phase the load sinks about 115 A. The results with and without the PCB are shown in the following graphs.

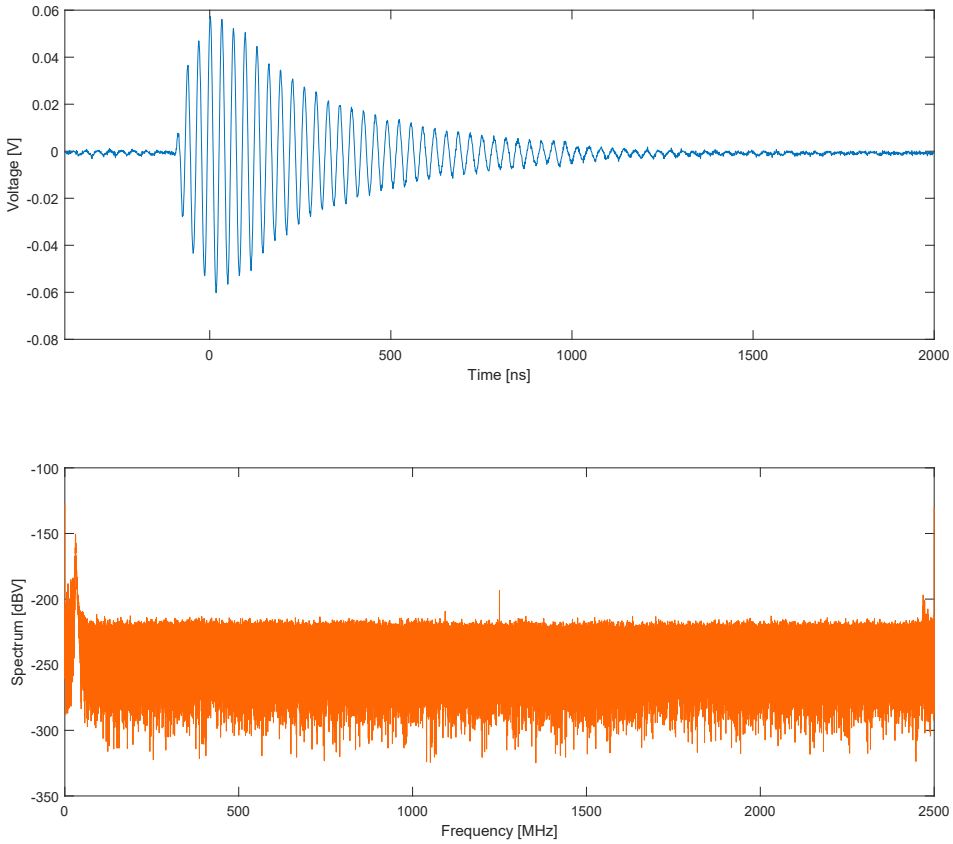


Figure 7.13. DUT connected to battery and inverter

In the time domain graph (the upper one) the disturbances due to the switch on of the hot air gun are shown: it's easy to distinguish a sinusoid at a certain frequency being attenuated after the current pulse.
For this reason an FFT analysis - represented in the bottom graph - has been performed.
There a peak @ $\sim 30MHz$ can be observed which is the fundamental frequency we cited above.
It is noticeable a not so relevant peak at 1.25GHz and a much more relevant one at $\sim 2.4GHz$ as expected. The latter is not a disturbance in this case: this test has been carried out with the device transmitting data to the smartphone over bluetooth operating at this frequency.

Now it is interesting to compare these results with those obtained with the same

workbench, but without our PCB (Figure 7.14).
In this case the measurements have been done placing the loop probe normally to the wire going from the battery directly to the inverter.
As expected in this case there is not the 2.4GHz component, since there is no object transmitting bluetooth data. The remaining part of the spectrum is pretty similar to the previous one: this suggests that the disturbances are originated by the load and the external environment.
In order to demonstrate this thesis, the experiment was repeated with a different load.

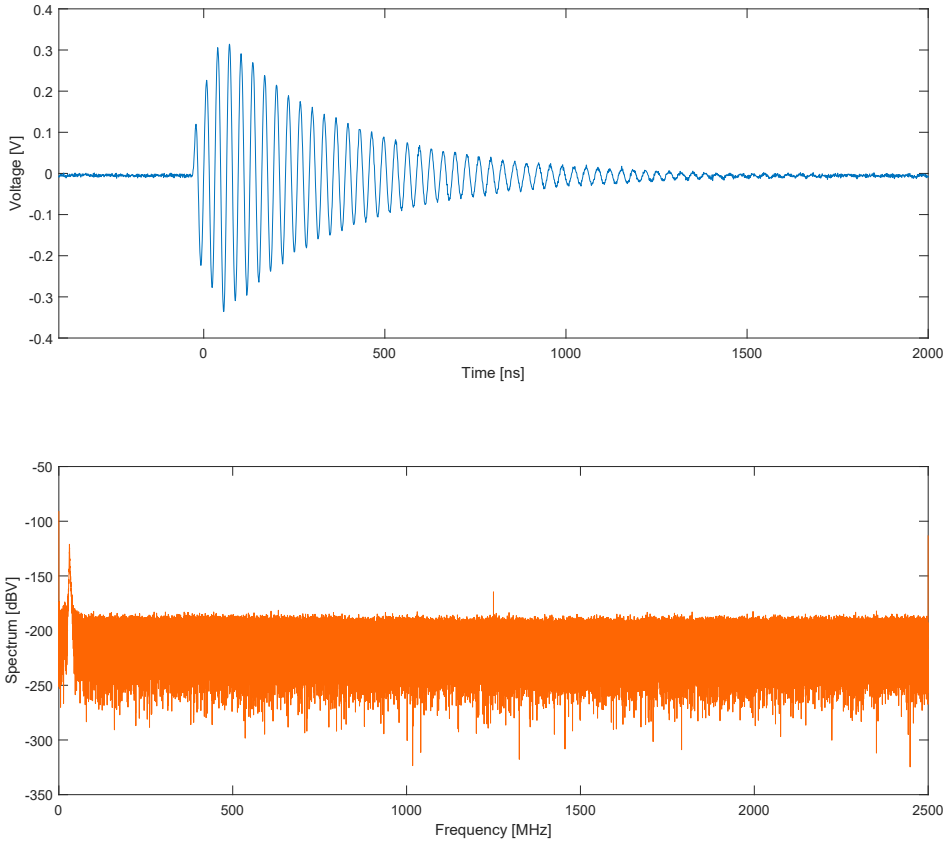


Figure 7.14. Configuration in Figure 7.13, without DUT

7.2.2 Second setup

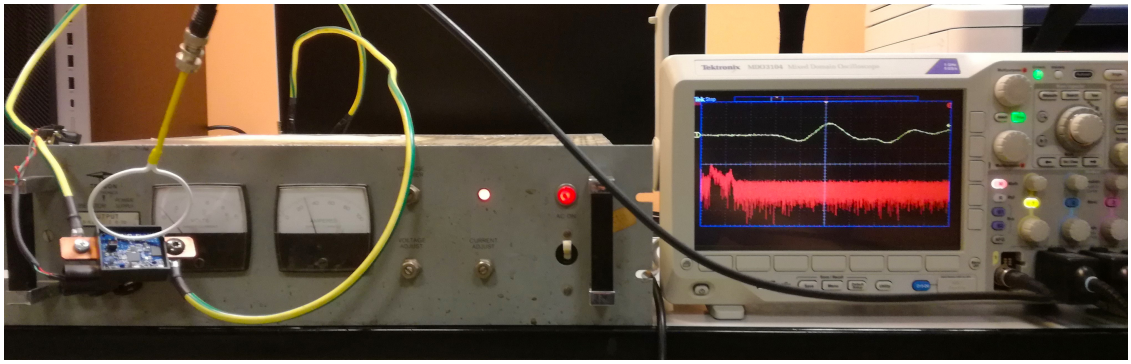


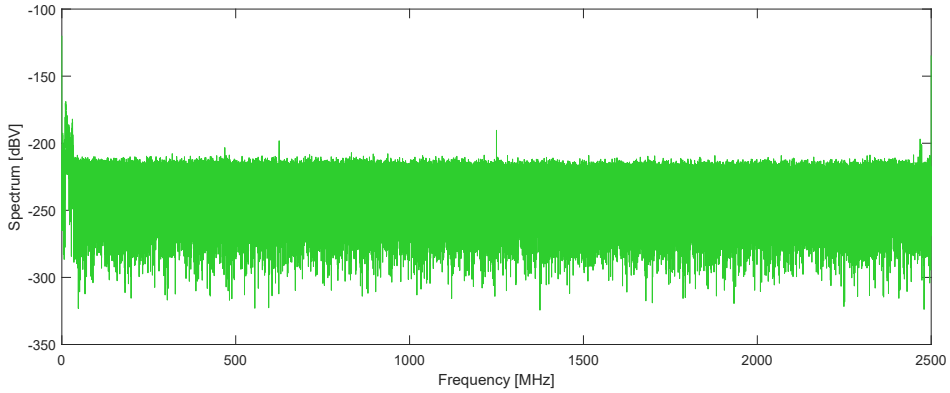
Figure 7.15. Second workbench for magnetic field relevation

The second setup is shown in Figure 7.15 and consist in:

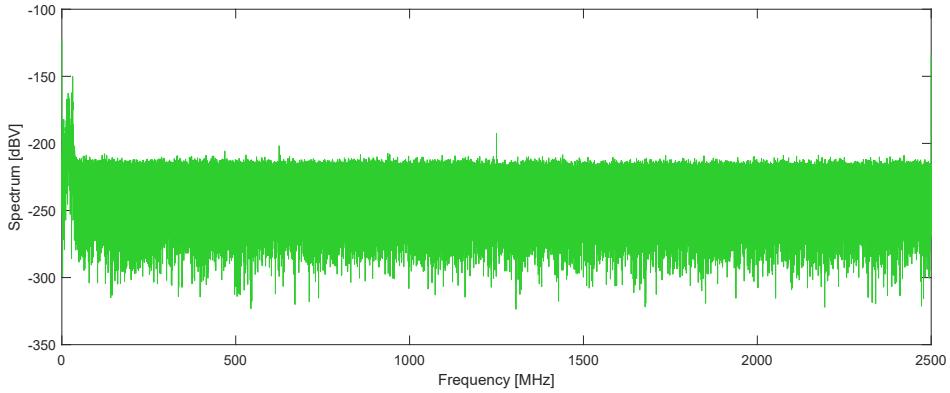
- **Power Supply**, Trygon Electronics Liberator Power Supply 4.8V-6.8V, 0A-70A
- **Magnetic Probe**, connected to a 50Ω coaxial cable
- **Oscilloscope**, MDO3104 set with both time domain and FFT
- **Android App** receiving battery voltage and current values from the PCB

This new setup is composed of an old current generator capable to output, in theory, up to 70A ranging from 4.8V to 6.8V. By connecting the shunt resistor between the two terminal of the generator, it goes into protection mode because it cannot output a current corresponding to a voltage drop of 4.8V (10e5 A) and, therefore, it gives the maximum current it can, limiting the voltage respecting Ohm's law. By the way it was not possible to get 70A but only 30A, maybe this is due to the aging of the components or to internal unknown modifications.

From the Figure 7.16 it is visible that also in this case there is a component at 2.4GHz only in configuration (a), as expected. The remaining part has the same behaviour in both (a) and (b) and is much more noisy than in configuration one.



(a) With DUT



(b) Without DUT

Figure 7.16. FFT for the second setup in the two configurations

7.2.3 Conclusions

From the tests carried out, it can be asserted that BAT-MAN doesn't generate appreciable disturbances with respect to a laboratory environment full of noise.
In order to obtain reliable results these test should be performed in a totally different environment: an anechoic chamber is needed to be totally shielded from undesired noise sources and a passive load should be used too.

7.3 On Field Measurements

Figure 7.17. BAT-MAN mounted on a FIAT Bravo

After the tests performed in laboratory the device was ready to be mounted in a real battery-car system as shown in Figure 7.17.

Acquisitions were stored in a *.txt* file to be post-processed using MATLAB with the possibility to apply the algorithm provided by *brainTechnologies* to predict the SoH and SoC of the battery under test.
From Figure 7.18 it is interesting to observe cranking which lasts for more or less 1 second; current reaches peaks as high as 600A with a corresponding voltage sharply sloping down to 8V.

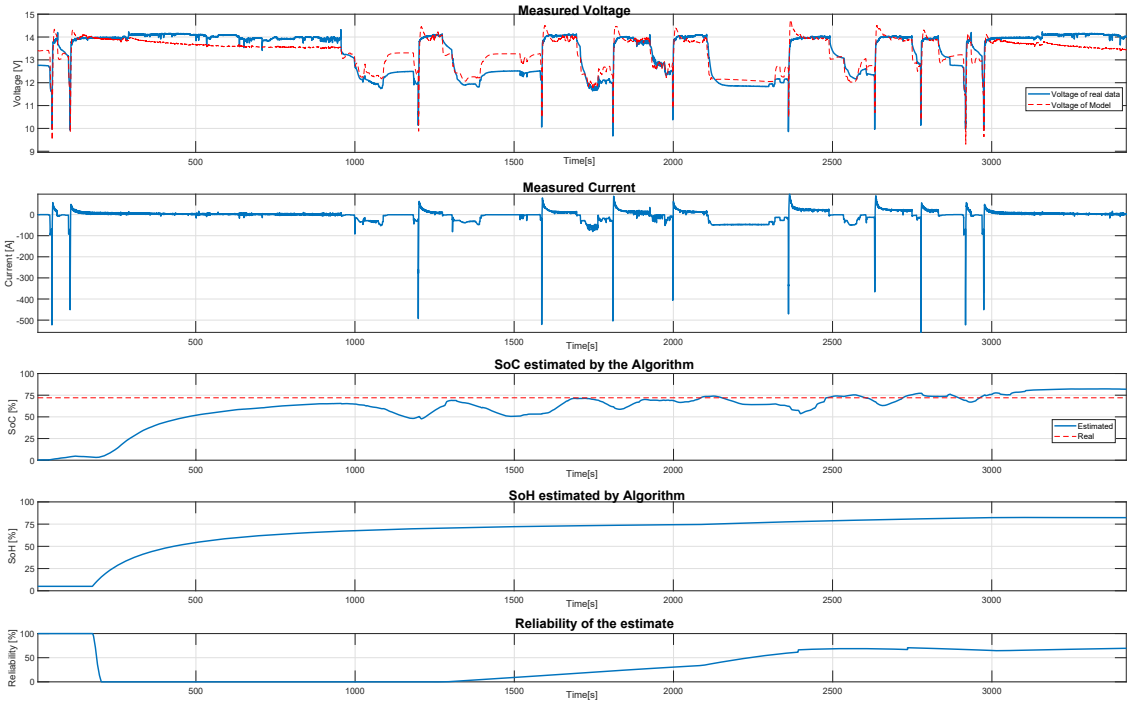


Figure 7.18. BAT-MAN results on an Alfa 159 Diesel

The Kalman filter based algorithm is capable of evaluating the SoC and SoH in real-time according to voltage, current and temperature values. As shown in the fifth plot, when the acquisition starts the two figure of merits are not reliable at all since the filter doesn't have enough past information to create a veridical model.

7.4 Measurement Campaign

7.4.1 Slow Discharge

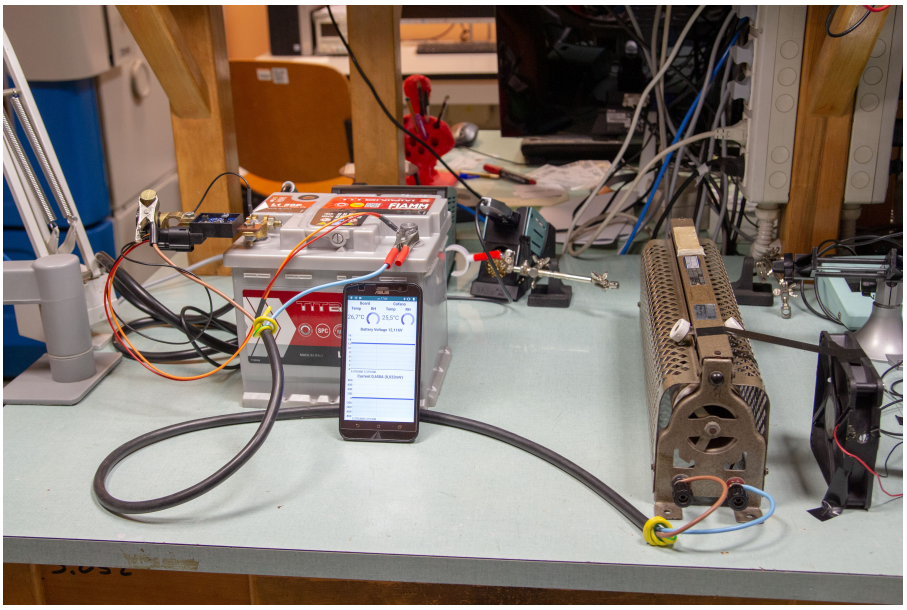


Figure 7.19. Discharge Setup with Passive Load

This test was carried out using a 100Ω rheostat set to ≈ 18.5Ω in order to sink $0.65A$, this value had to be chosen to avoid overheating of the load. The battery under test - a FIAMM L150P - has a capacity of $50Ah$, therefore the estimated time to complete discharge it is of 77 $hours$.

In Figure 7.20 the acquired data are reported. It is possible to observe that after approximately 50 hours the battery was not able to provide the requested current anymore, this causes a sharp voltage reduction to about 7V. After 6 hours another fast decreasing of the current take place, bringing the voltage under 5V; then, after 8 hours, it goes down till 3V and the battery cannot provide current at all. At this point it is possible to consider the battery SoC equal to 0

In the third graph the discharge rate is plotted showing the degradation of the SoC over time, going from a fully charged state to a low one. To achieve this result it was necessary to keep into account the SoH of the battery; due to aging and stress, the rated capacity is lower than the nominal one, in fact this discharge cycle lasted for 10 hours less than expected. The total DOD, obtained by using Coulomb Counting method with a MATLAB script, was of about 38Ah instead of the nominal 50Ah capacity.

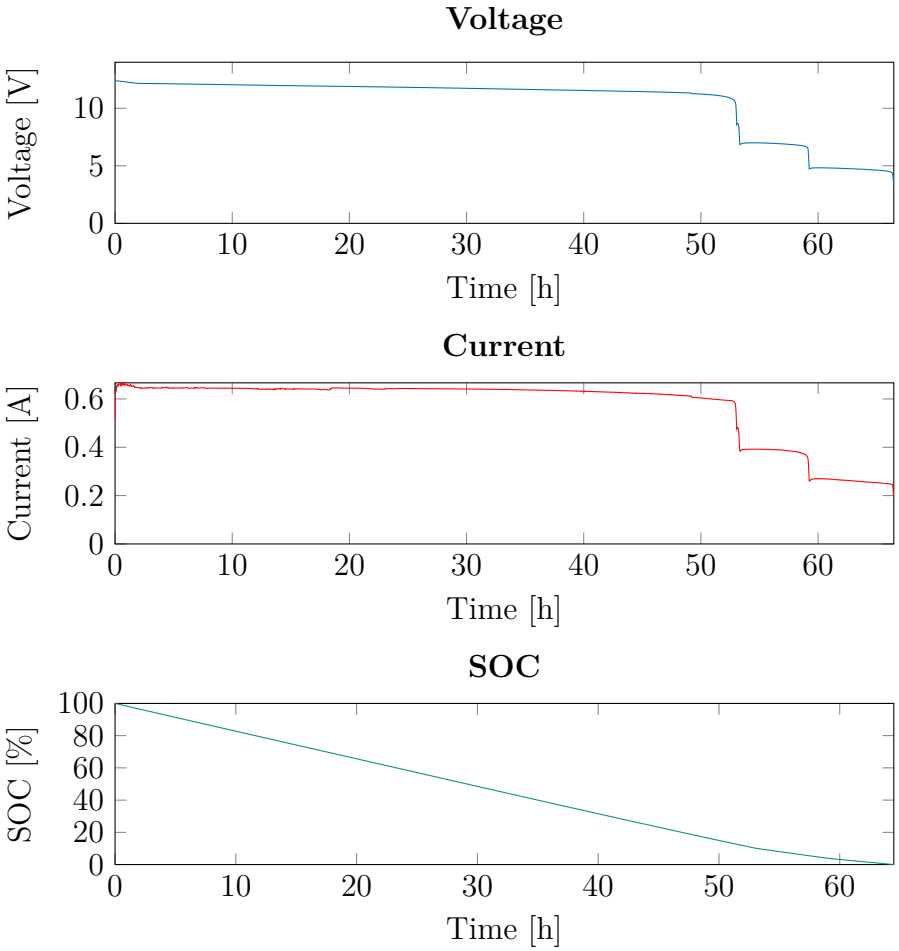


Figure 7.20. Discharge cycle - BOSCH S3 001

7.4.2 Fast Discharge

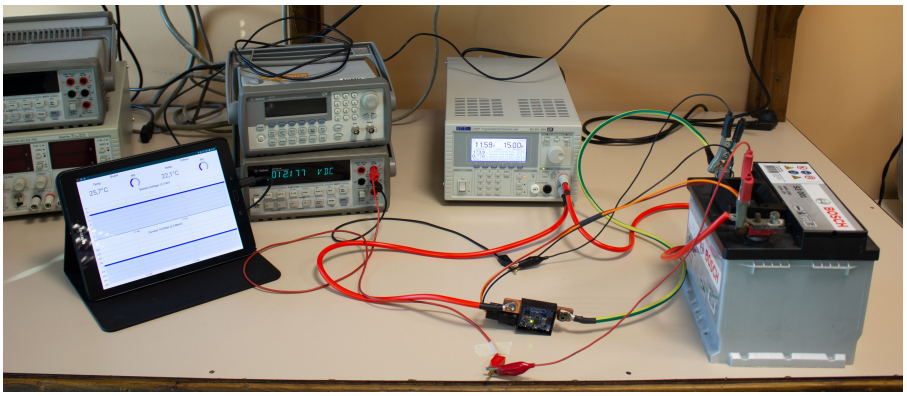


Figure 7.21. Discharge Setup with Active Load

For this test it was possible for the first time to use an active load - the TTi LD400P - lent by another laboratory. The aim of this test is to fully discharge a battery at a constant current of 10 A obtaining the discharge characteristic of the battery under test.

The battery involved is the *BOSCH S3 001 - 12V 41Ah*. It is not a new battery since it has been subjected to many full charge/discharge cycles by other students previously involved in the project.

The graph in Figure 7.22 shows a partial discharge cycle, in fact the final SoC is not equal to 0. When the load is disconnected the voltage rapidly increases up to 11V, then it slowly gets back to its resting voltage.
By mean of this phenomena it is clear that an algorithm only based on voltage measurement can be misleading.

The battery was then charged (subsection 7.4.3) and underwent another full discharge cycle in Figure 7.23. In this case, when disconnecting the load, the voltage remains stacked at 3V proving the battery was completely exhausted.

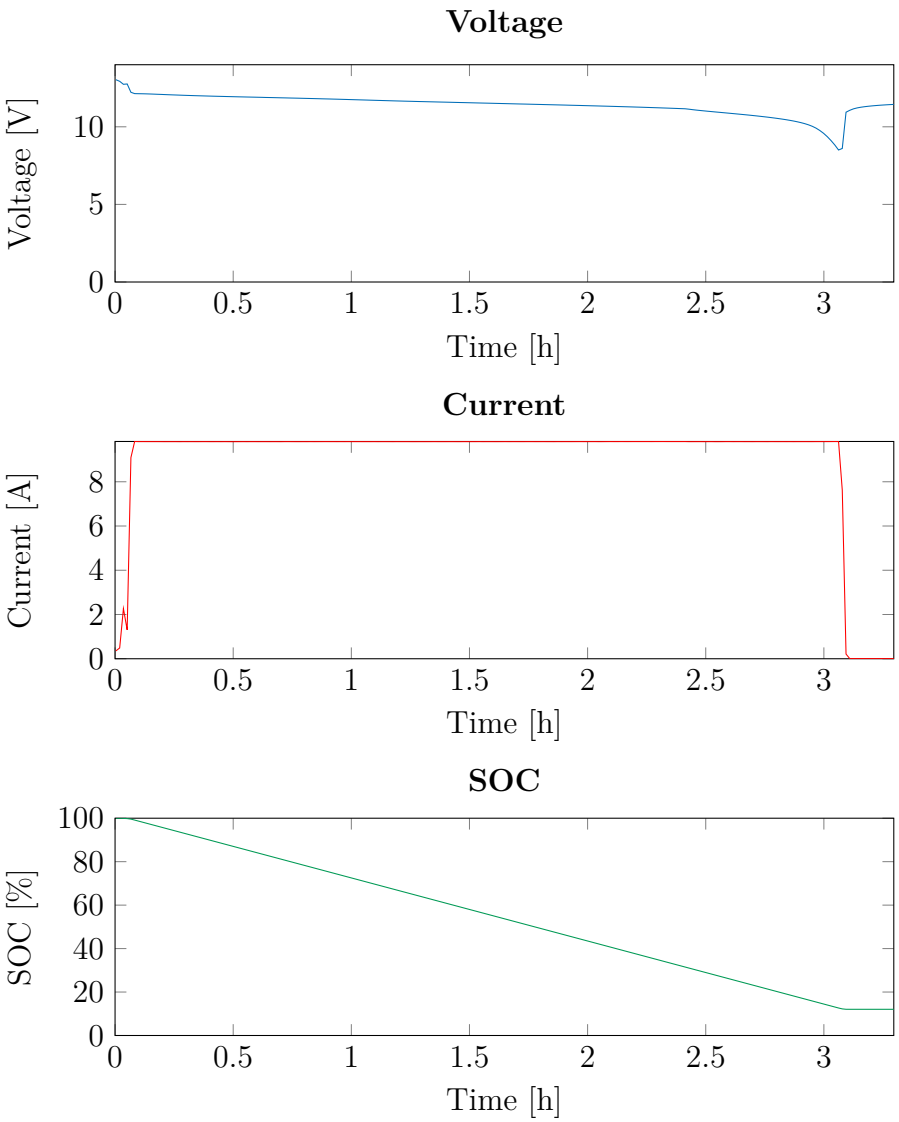


Figure 7.22. Discharge cycle - BOSCH S3 001

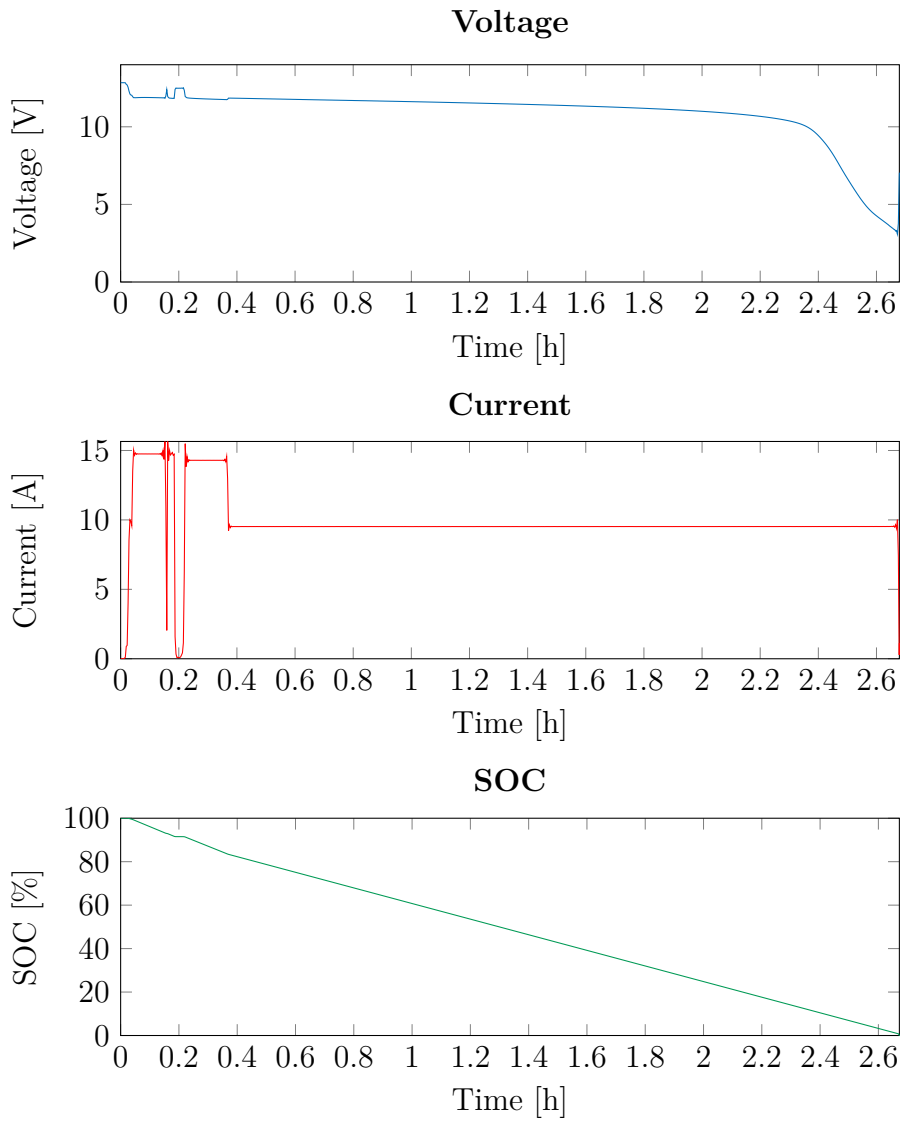


Figure 7.23. Discharge cycle - BOSCH S3 001

7.4.3 Charge

Figure 7.24. Charge Setup with Car Battery Charger

This test was carried out using a *CTEK MXS 5.0* lead-acid batteries charger. The graph in Figure 7.24 shows how the battery had been recharged at first providing a constant 5A current then exponentially decreasing it until the fully charged state has not been reached. In this moment a maintenance current - measured to be 0.1A - is provided to prevent the battery from self-discharging while not in use.

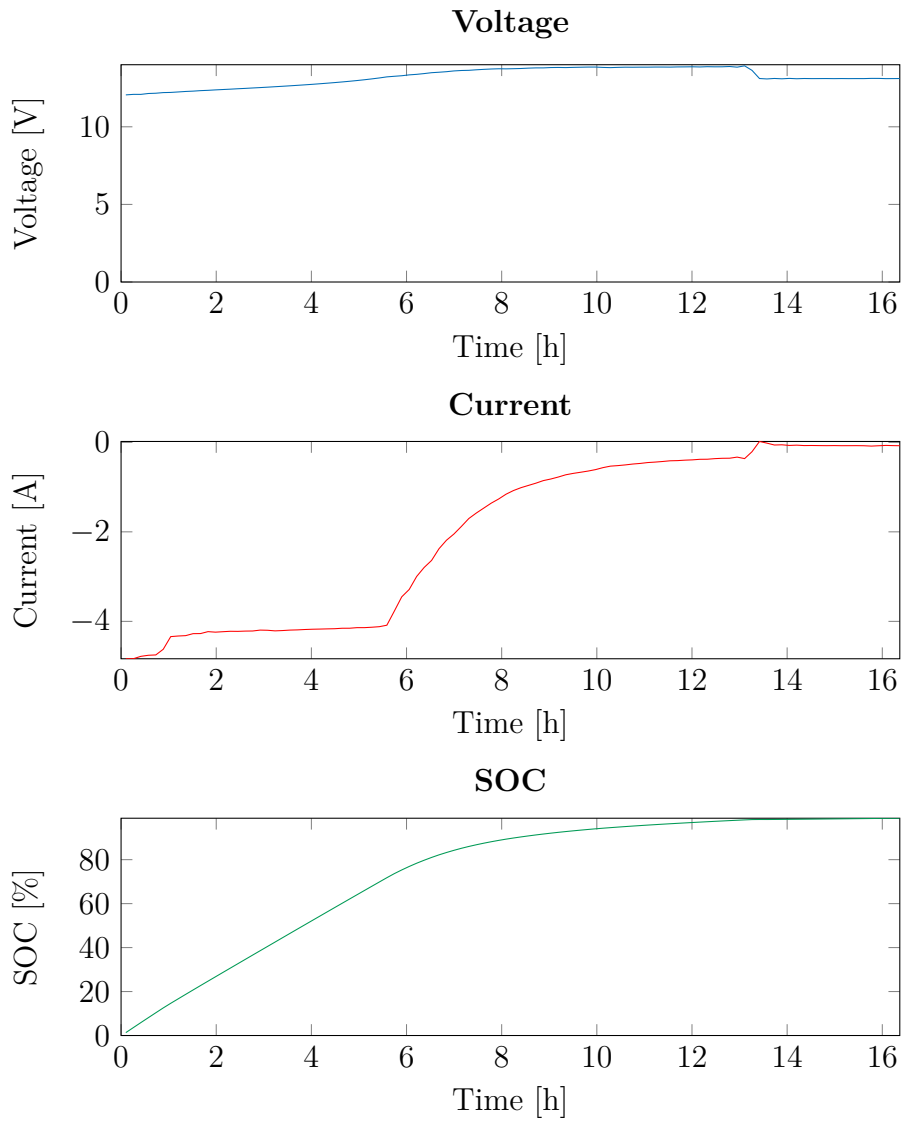


Figure 7.25. Charge cycle - BOSCH S3 001

7.4.4 SoC vs Voltage

From the previous discharge cycles it is possible to trace a plot of the SoC level with respect to the voltage of the battery obtaining the so called *Discharge Curve*. It is noticeable that the three curves are extremely different each other, though Red and Green ones are similar since are obtained by the same battery.

This fact validate the fallacious results which can be obtained by using *Voltage Method* (subsection 1.4.1) to get information about SoC and SoH.

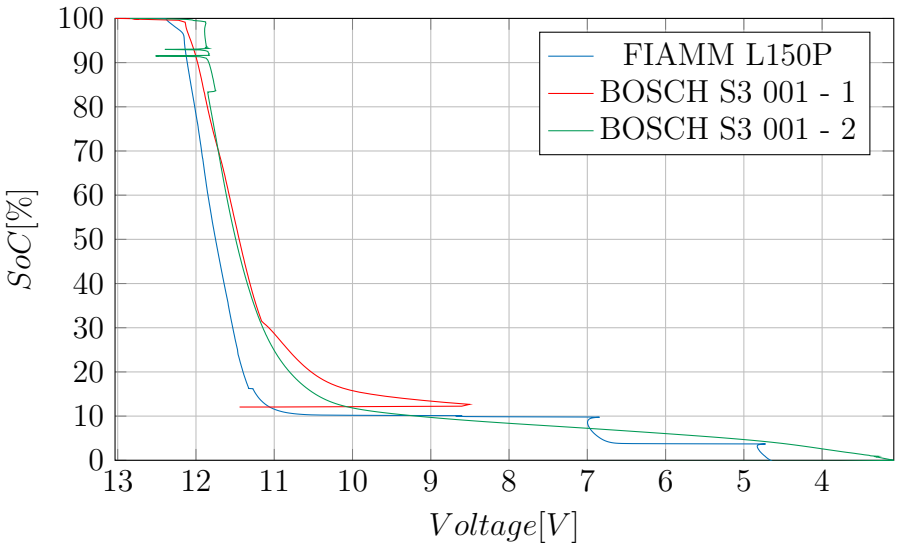


Figure 7.26. SoC vs Voltage characteristic.

Chapter 8

Future perspectives

8.1 Future HW developments

An external protection circuit can be implemented to provide surge and fast-transient protection and demonstrate the different immunity levels to IEC61000-4-4 and IEC61000-4-5.

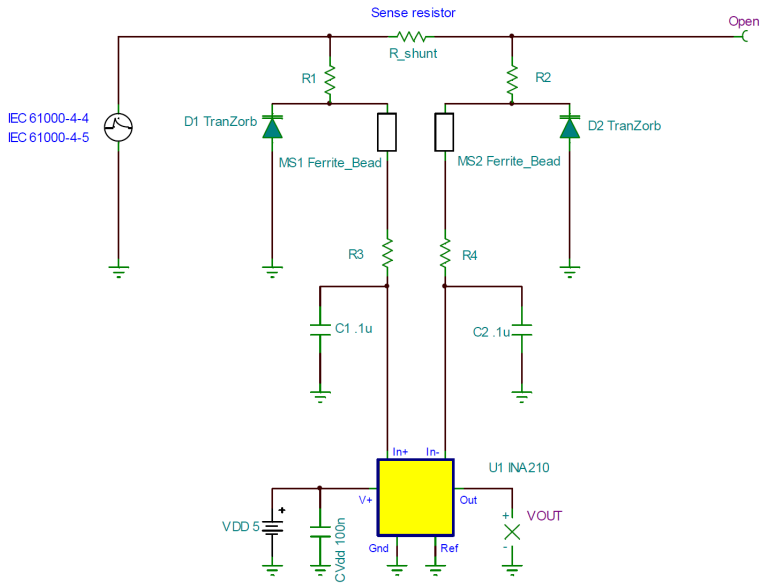
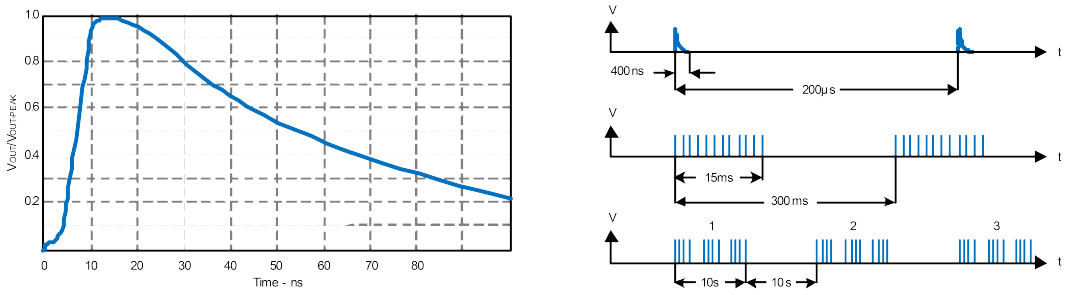
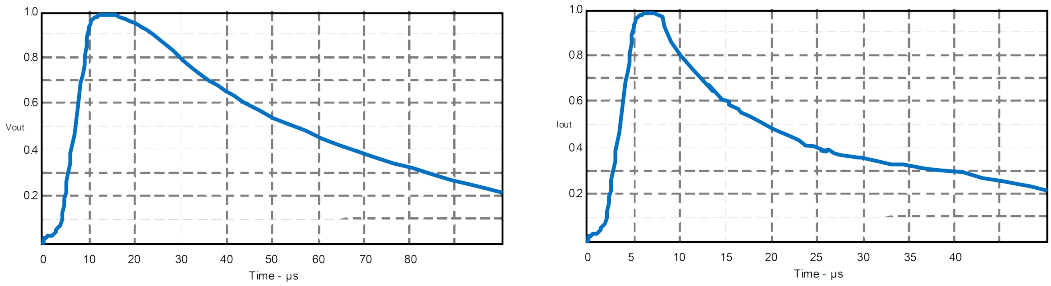


Figure 8.1. Analog Front End Protection Circuit [32]



(a) IEC-61000-4



(b) IEC-61000-5 Voltage Impulse and Current Impulse

Figure 8.2. IEC Transient Pulses [32]

Transient signals or radiated emissions can cause electrical damages, overstresses or other disruptions. With this protection circuit the device is protected by IEC61000-4-4 and IEC61000-4-5 voltages represented in Figure 8.2.

The components in the Figure 8.1 have the following purposes:

- TVS diodes provide a diversion path and their clamping voltage must be selected according to the maximum ratings of the INA226. They are used because are capable to clamp to ground very fast transient.
- R1 and R2 are need to limit current on the tranzorbs
- C4 and C5 provide another diversion path
- ferrite beads attenuate the high frequency signals through C4 and C5
- R3 and R4 attenuate ringing between the ferrite beads and the capacitors

The components value can be selected according to the measurement performed by Texas Instruments in the proper application note [32]

8.2 Future FW Development

To perform acquisitions when the device is offline, it is useful to implement an algorithm that allows to store data in the flash when no connection is established. These samples must be then sent to the smartphone as soon as it connects.

This feature could allow to monitor the battery also when the driver is not inside the car for example understanding if - when the car is parked - there is some equipment that sinks current.

It can be interesting to understand how long the offline acquisition can be. With the settings selected each voltage or current value is acquired every 16.488 ms and, after 30 acquisions, two temperature values are sent. Every sample is a 16 bit variable, so:

$$acquisition_{time} = 30 \cdot 16.488ms = 494.640ms \tag{8.1}$$

$$acqusition_{size} = 2 \cdot (30 \cdot 16bit) + (2 \cdot 16) = 992bit = 124byte \tag{8.2}$$

Having a flash with a size of 32 MByte:

$$\#acquisitions = \frac{32MByte}{124Byte} = 258060 \tag{8.3}$$

$$acquisition_{duration} = 258060 \cdot 494.640ms \approx 36h \tag{8.4}$$

This is a considerable time that can be enough for most of the applications.

However, in order to increase this value or to reduce the files transmitted to the app, it is possible to reduce the sampling frequency when the current involved is stable. The best idea could be to differentiate the samples to understand if there is a sensible variation and, otherwise, average them. By doing this, different sampling frequency must be handled and this information must be stored and given to the app in order to reconstruct the original signal.

8.3 Cloud computing

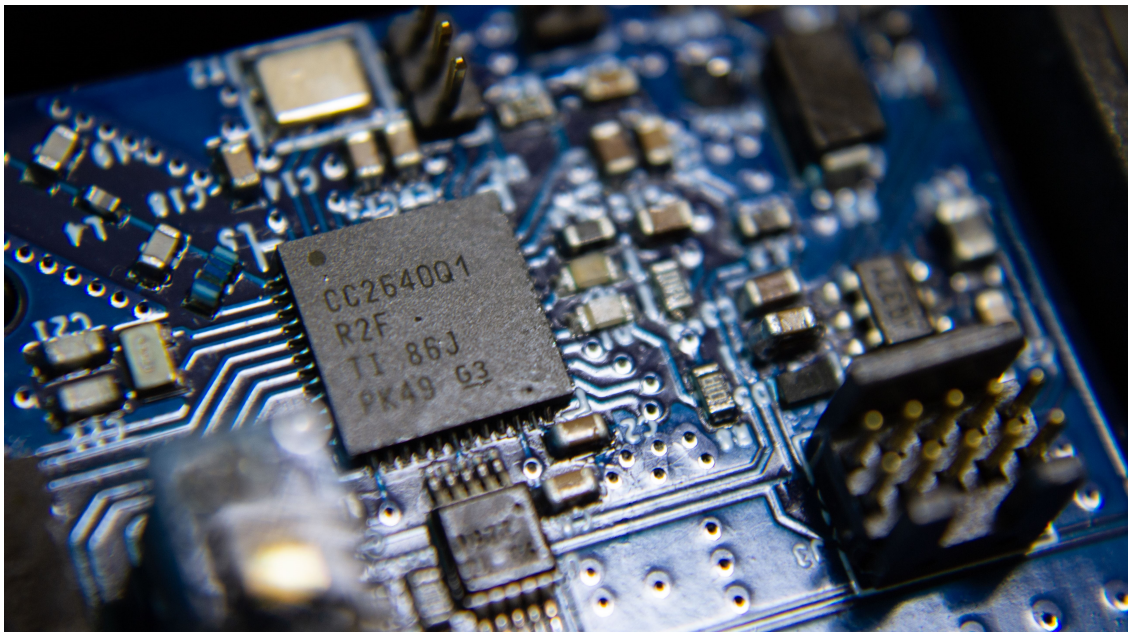
An interesting development consist in cloud computing; an on-demand availability of computer resources without the need of a direct management by the user. This mainly consists in data storage and computational power capabilities.

This technology would allow to put into communication many devices - therefore many users - in order to achieve better SoC and SoH estimations. By mean of servers capable to communicate via internet with the smartphones paired with BAT-MAN, it is possible to collect all the data referred to a huge amount of different batteries, alternators and cars.
Therefore, by mean of a neural network, it is possible to implement a *Data-Driven* model also called as Black-Box (see subsection 1.2.1) ensuring a big reliability. The strength of this technique is that the model becomes every day more accurate thanks to machine learning. Moreover, this solution overcomes all the problems related the lack of resources of a microcontroller that can be a big problem when dealing with solutions like *Kalman Filter*.

An intermediate solution could involve the so called *Edge Computing*, moving the computation to the edges, represented in this case by the smartphone. This solution allows to reduce response time and improves bandwith, the latter is a growing problem since IoT devices are always more diffused and greedy.
Also in this case the cloud architecture would remain but keeping data elaboration at a lower level, allowing the algorithm to work also when no internet connection is available.

Chapter 9

Conclusions



The objective of this work was to develop an automotive device capable of measuring Voltage, Current and Temperature with the aim of obtaining the state of charge (SoC) and state of health (SoH) of a 12V lead-acid battery.

To achieve this results - starting from the requirements specified by the project creators - it was necessary to follow the whole production cycle from the components choice to the verification and validation process. Through this cycle, it was possible to develop a fully working prototype accomplishing all the desired tasks.

According to the involved components ratings and to the validation process, a datasheet has been generated (section 9.1).

Due to the lack of specific equipment, many of the measurements provide just qualitative results. In order to achieve the best results in all conditions, sensors must be calibrated using certified instrumentation.

Although the keypoint of this project was to design a device capable to measure the parameters of interest, it would not be able to generate SoH and SoC estimations without the use of proper prediction algorithms. The estimator designed by BrainTechnologies can be thoroughly implemented in BAT-MAN device directly in its firmware. By mean of cloud computing the possibility of increasing the number of battery models arises; this would increase the reliability of the estimation algorithm.

BAT-MAN can find its principle application in automotive industry to be mounted on board providing useful informations to the driver and to insiders as car and battery manufacturers or mechanical workshops.

An efficient battery manager will become essential in the near future for hybrid and fully electric vehicles where the main fuel consists in the electricity provided by the battery. This application field is so critical that single cells must be monitored too in order to avoid charge unbalance between them.

However, to port this technology to lithium-ion batteries voltage range must be extended by adding a proper conditioning system because these technologies reach voltages up to 400 V. For what concerns current the actual device is already capable to be used without need of modifications; the actual range is also too high, a shunt resistor of $100\mu\Omega$ could be used instead. This is because using voltages of hundreds of volts, it is possible to obtain huge powers limiting maximum currents.

To sum up, the specifications required by the Regional Project commitments have been satisfied, opening a window of possibilities both on the actual market leader technologies and on the future ones.

9.1 Datasheet

Features

- Automotive compliant components
- Lead-acid batteries specific
- Supplied by 12V car battery
- Voltage measurement up to 17V
- Current sense using a shunt resistor
- Low-side or High-side current sensing
- Hood humidity and temperature sensing
- Board humidity and temperature sensing for thermal derating compensation
- Protection against transient overvoltage according to ISO 7637-3
- Protection against reverse polarity
- SoC and SoH embedded algorithm
- Bluetooth 4.2 capability
- Offline mode supported
- Ad hoc Android application
- Flexible technology (LiPo batteries oriented)

Description

The battery manager (BAT-MAN) measures voltage, current and temperature of a 12V Pb-acid batteries ensuring an optimum accuracy. The *brainTechnologies* patented algorithm predicts the SoH and SoC of the battery providing real-time informations about the ageing effects. Acquired data are transmitted through a Bluetooth® connection to the user's smartphone giving the possibility to be up to date on the battery status. The device has to be connected in series between battery and load, attached to the negative pole.
With BAT-MAN the drivers can avoid breakdowns resulting from a discharged battery.

Application

- On Board SoH and SoC monitor
- Car alternator health monitor
- Abnormal battery behaviour alert
- Low cost laboratory battery charge/discharge monitor

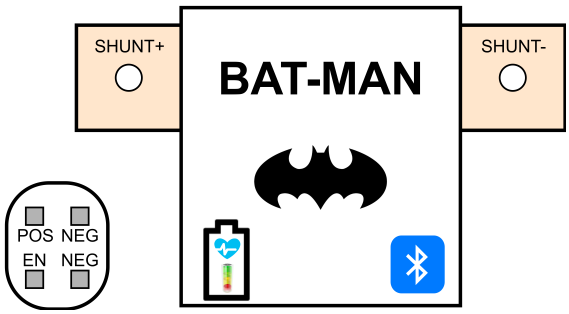


Figure 9.1. BAT-MAN Pinout

	MIN	TYP	MAX	UNIT
RATINGS				
Operating Supply Voltage	4		17	V
Supply Current			80	mA
Operating Temperature	-40		105	$°C$
CURRENT AND VOLTAGE DC ACCURACY				
ADC native resolution		16		Bits
Current resolution		50		mA
Current accuracy			150	mA
Voltage resolution		1.25		mV
Voltage accuracy (no compensation)			20	mV
ADC conversion time	140		8244	μs
HOOD T/H DC ACCURACY				
ADC native resolution		14		Bits
RH range	0		100	% RH
RH resolution		0.1		% RH
RH accuracy		2		% RH
ADC RH conversion time	2.5		6.5	ms
Temperature range	-40		125	$°C$
Temperature resolution		0.1		$°C$
Temperature accuracy		0.2	0.4	$°C$
ADC temperature conversion time	3.65		6.35	ms
BOARD T/H DC ACCURACY				
ADC native resolution		14		Bits
RH range	0		100	% RH
RH resolution		0.1		% RH
RH accuracy		2	3	% RH
ADC RH conversion time	275		660	μs
Temperature range	-40		85	$°C$
Temperature resolution		0.1		$°C$
Temperature accuracy		0.2	0.4	$°C$
ADC temperature conversion time	225		610	μs

Appendix A

Flash Memory Library

S25FL256L.h

```
#include <stdint.h>
#include <stddef.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <ti/drivers/GPIO.h>
#include <ti/drivers/SPI.h>
#include <ti/drivers/spi/SPICC26XXDMA.h>
#include <ti/drivers/dma/UDMACC26XX.h>
#include <ti/drivers/GPIO.h>
#include "Board.h"

#ifndef S25FL256L_H
#define S25FL256L_H

/*----------------------------------------
Numeric constants
----------------------------------------*/
// Commands
// Read Device Identification
#define   MEM_CMD_RDID      0x9F
#define   MEM_CMD_RSFDP     0x5A
#define   MEM_CMD_RDQID     0xAF
#define   MEM_CMD_RUID      0x4B
// Register Access
#define   MEM_CMD_RDSR1     0x05
#define   MEM_CMD_RDSR2     0x07
#define   MEM_CMD_RDCR1     0x35
#define   MEM_CMD_RDCR2     0x15
#define   MEM_CMD_RDCR3     0x33
#define   MEM_CMD_RDAR      0x65
#define   MEM_CMD_WRR       0x01
#define   MEM_CMD_WRDI      0x04
#define   MEM_CMD_WREN      0x06
#define   MEM_CMD_WRENV     0x50
#define   MEM_CMD_WRAR      0x71
#define   MEM_CMD_CLSR      0x30
#define   MEM_CMD_4BEN      0xB7
#define   MEM_CMD_4BEX      0xE9
#define   MEM_CMD_SBL       0x77
```

```
41 #define   MEM_CMD_QPIEN     0x38
42 #define   MEM_CMD_QPIEX     0xF5
43 #define   MEM_CMD_DLPRD     0x41
44 #define   MEM_CMD_PNVDLR      0x43
45 #define   MEM_CMD_WVDLR     0x4A
46 // Read Flash Array
47 #define   MEM_CMD_READ      0x03
48 #define   MEM_CMD_4READ     0x13
49 #define   MEM_CMD_FAST_READ   0x0B
50 #define   MEM_CMD_4FAST_READ    0x0C
51 #define   MEM_CMD_DDRFR     0x0D
52 #define   MEM_CMD_4DDRFR      0x0E
53 #define   MEM_CMD_DOR       0x3B
54 #define   MEM_CMD_4DOR      0x3C
55 #define   MEM_CMD_QOR       0x6B
56 #define   MEM_CMD_4QOR      0x6C
57 #define   MEM_CMD_DIOR      0xBB
58 #define   MEM_CMD_4DIOR     0xBC
59 #define   MEM_CMD_QIOR      0xEB
60 #define   MEM_CMD_4QIOR     0xEC
61 #define   MEM_CMD_DDRQIOR     0xED
62 #define   MEM_CMD_4DDRQIOR    0xEE
63 // Program Flash Array
64 #define   MEM_CMD_PP        0x02
65 #define   MEM_CMD_4PP       0x12
66 #define   MEM_CMD_QPP       0x32
67 #define   MEM_CMD_4QPP      0x34
68 // Erase Flash Array
69 #define   MEM_CMD_SE        0x20
70 #define   MEM_CMD_4SE       0x21
71 #define   MEM_CMD_HBE       0x52
72 #define   MEM_CMD_4HBE      0x53
73 #define   MEM_CMD_BE        0xD8
74 #define   MEM_CMD_4BE       0xDC
75 #define   MEM_CMD_CE        0x60
76 // Erase/Program Suspend/Resume
77 #define   MEM_CMD_EPS   0x75
78 #define   MEM_CMD_EPR   0x7A
79 // Security Region Array
80 #define   MEM_CMD_SECRE     0x44
81 #define   MEM_CMD_SECRP     0x42
82 #define   MEM_CMD_SECRR     0x48
83 //Array Protection
84 #define   MEM_CMD_IBLRD     0x3D
85 #define   MEM_CMD_4IBLRD      0xE0
86 #define   MEM_CMD_IBL       0x36
87 #define   MEM_CMD_4IBL      0xE1
88 #define   MEM_CMD_IBUL      0x39
89 #define   MEM_CMD_4IBUL     0xE2
90 #define   MEM_CMD_GBL       0x7E
91 #define   MEM_CMD_GBUL      0x98
92 #define   MEM_CMD_SPRP      0xFB
93 #define   MEM_CMD_4SPRP     0xE3
94 //Individual and Region Protection
95 #define   MEM_CMD_IRPRD     0x2B
96 #define   MEM_CMD_IRPP      0x2F
97 #define   MEM_CMD_PRRD      0xA7
98 #define   MEM_CMD_PRL       0xA6
99 #define   MEM_CMD_PASSRD      0xE7
100 #define   MEM_CMD_PASSP     0xA8
101 #define   MEM_CMD_PASSU     0xEA
102 // Reset
103 #define   MEM_CMD_RSTEN     0x66
```

```
104 #define   MEM_CMD_RST       0x99
105 #define   MEM_CMD_MBR       0xFF
106 // Deep Power Down
107 #define   MEM_CMD_DPD       0xB9
108 #define   MEM_CMD_RES       0xAB
109
110 /*---------------------------------------
111 Functions prototypes
112 ---------------------------------------*/
113
114 /************* BASIC SPI FUNCTIONS *********/
115
116 //write "txBufLength" words over SPI communication
117 void SPI_write(uint8_t    * txBuf, uint16_t txBufLength, SPI_Handle handle);
118
119 //Read "rxBufLength"words over SPI communication
120 void SPI_read(uint8_t   * rxBuf, uint16_t rxBufLength, SPI_Handle handle);
121
122 //write "txBufLength" words over SPI communication and then read "rxBufLength"
123 //The transaction must be as long as the transmission during reading we transmit NULL
124 void SPI_readWrite(uint8_t    * rxBuf,uint8_t   * txBuf, uint16_t txBufLength, SPI_Handle handle);
125
126 /************** BASIC MEMORY FUNCTIONS ********************/
127
128 //open SPI communication for the memory
129 SPI_Handle MEM_openSPI();
130
131 //read a statsus or control register of the memory
132 //this function is called by the following ones
133 uint8_t MEM_readRegister(uint8_t cmd );
134
135 //write a command followed by an address
136 //this function is called by the following ones
137 void MEM_writeCommandAddress(uint8_t cmd, uint32_t address);
138
139 //write a single command
140 //this function is called by the following ones
141 void MEM_writeCommand(uint8_t cmd);
142
143 /********* READ REGISTER FUNCTIONS    ***************/
144
145 //read status register 1
146 uint8_t MEM_readStatus1();
147
148 //read status register 2
149 uint8_t MEM_readStatus2();
150
151 //read configuration register 1
152 uint8_t   MEM_readConfiguration1();
153
154 //read configuration register 2
155 uint8_t   MEM_readConfiguration2();
156
157 //read configuration register 3
158 uint8_t   MEM_readConfiguration3();
159
160 /************** REGISTER WRITE FUNCTIONS ************/
161
162 //write enable
163 void MEM_writeEnable();
164
165 //write disable
166 void MEM_writeDisable();
```

```
167
168 //write enable volatile
169 void MEM_writeEnableVolatile();
170
171 //clear status register WIP, WEL, P_ERR, E_ERR
172 void MEM_clearStatus();
173
174 //enter 4 byte address mode (4byte instructions are not affected)
175 void MEM_4byteEn();
176
177 //exit 4 byte address mode (4byte instructions are not affected)
178 void MEM_4byteEx();
179
180 /*********** READ FLASH FUNCTIONS ***************/
181
182 //read memory array
183 void MEM_read(uint32_t address, uint16_t n_read, uint8_t * data);
184
185 /*************** WRITE FLASH FUNCTIONS *************/
186
187 //Generic size writing
188 //This function will be called by the following ones
189 void MEM_program(uint32_t address, uint8_t * data, uint16_t n_write);
190
191 //write one page (256 byte) of the memory array
192 //before calling this operation we need to enable writing
193 void MEM_programPage(uint32_t address, uint8_t * data);
194
195 /************ ERASE COMMANDS *************/
196
197 //sector erase
198 void MEM_eraseSector(uint32_t address);
199
200 //half block erase
201 void MEM_eraseHalfBlock(uint32_t address);
202
203 //block erase
204 void MEM_eraseBlock(uint32_t address);
205
206 //chip erase
207 void MEM_eraseChip();
208
209 /******** PROGRAM/ERASE SUSPEND/RESUME ************/
210
211 //program suspend
212 void MEM_programEraseSuspend();
213
214 //program resume
215 void MEM_programEraseResume();
216
217 /********** SOFTWARE RESET **************/
218
219 // software reset
220 void MEM_softwareReset();
221
222 /********* POWER DOWN ***************/
223 //enter in deep power down mode
224 void MEM_deepPowerDown();
225
226 //release from deep power down mode
227 void MEM_deepPowerDownRelease();
228 #endif
```

S25FL256L.c

```c
#include <stdint.h>
#include <stddef.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <ti/drivers/GPIO.h>
#include <ti/drivers/SPI.h>
#include <ti/drivers/spi/SPICC26XXDMA.h>
#include <ti/drivers/dma/UDMACC26XX.h>
#include <ti/drivers/GPIO.h>
#include "Board.h"
# include "S25FL256L.h"

/************* BASIC SPI FUNCTIONS *********/

//write "txBufLength" words over SPI communication
void SPI_write(uint8_t  * txBuf, uint16_t txBufLength, SPI_Handle handle){
  SPI_Transaction transaction;
  transaction.count   = txBufLength;
  transaction.txBuf   = txBuf;
  transaction.rxBuf   = NULL;

  //Write
  SPI_transfer(handle, &transaction);
}

//Read "rxBufLength"words over SPI communication
void SPI_read(uint8_t  * rxBuf, uint16_t rxBufLength, SPI_Handle handle){
  SPI_Transaction transaction;
  transaction.count   = rxBufLength;
  transaction.txBuf   = NULL;
  transaction.rxBuf   = rxBuf;

  //Read
  SPI_transfer(handle, &transaction);
}

//write "txBufLength" words over SPI communication and then read "rxBufLength"
//The transaction must be as long as the transmission during reading we transmit NULL
void SPI_readWrite(uint8_t  * rxBuf,uint8_t  * txBuf, uint16_t txBufLength, SPI_Handle handle){
  SPI_Transaction transaction;
  transaction.count   = txBufLength;
  transaction.txBuf   = txBuf;
  transaction.rxBuf   = rxBuf;
  //Write then Read
  SPI_transfer(handle, &transaction);
}
/************** BASIC MEMORY FUNCTIONS ********************/

//open SPI communication for the memory
SPI_Handle MEM_openSPI(){
  SPI_Params params;
  SPI_Params_init(&params); //initialization with some default values
  params.bitRate      = 100000; //12MHz is the maximum value as Master
  params.frameFormat  = SPI_POL1_PHA1;
  params.mode         = SPI_MASTER;
  params.dataSize     = 8;

  // Open SPI
  return SPI_open(Board_SPI_MASTER, &params);
```

```
61 }
62
63 //read a status or control register of the memory
64 //this function is called by the following ones
65 uint8_t MEM_readRegister(uint8_t cmd ){
66   uint8_t txBuf[2];
67   uint8_t rxBuf[2];
68   SPI_Handle handle;
69
70   uint16_t txBufLength= sizeof(txBuf);
71
72   // Open SPI
73   handle=MEM_openSPI();
74
75   // Associate command to buffers
76   txBuf[0]= cmd;
77   txBuf[1]= NULL;
78
79   //Write command then read value
80   SPI_readWrite(rxBuf, txBuf, txBufLength, handle);
81
82   // Close SPI
83   SPI_close(handle);
84
85   return rxBuf[1];
86 }
87
88 //write a command followed by an address
89 //this function is called by the following ones
90 void MEM_writeCommandAddress(uint8_t cmd, uint32_t address){
91   uint8_t txBuf[5];
92   //uint8_t rxBuf[5];
93   SPI_Handle handle;
94
95   uint16_t txBufLength= sizeof(txBuf);
96
97   // Open SPI
98   handle=MEM_openSPI();
99
100  // Command
101  txBuf[0]=cmd;
102  txBuf[1]= address>>24;
103  txBuf[2]= address>>16;
104  txBuf[3]= address>>8;
105  txBuf[4]= address;
106
107  //Associate command to buffer
108  SPI_write(txBuf, txBufLength, handle);
109
110  // Close SPI
111  SPI_close(handle);
112 }
113
114 //write a single command
115 //this function is called by the following ones
116 void MEM_writeCommand(uint8_t cmd){
117   uint8_t txBuf[1];
118   //uint8_t rxBuf[1];
119   SPI_Handle handle;
120
121   uint16_t txBufLength= sizeof(txBuf);
122
123   // Open SPI
```

```
124   handle=MEM_openSPI();
125
126   // Command
127   txBuf[0]= cmd;
128
129   //Associate command to buffer
130   SPI_write(txBuf, txBufLength, handle);
131
132   // Close SPI
133   SPI_close(handle);
134 }
135
136 /********* READ REGISTER FUNCTIONS  ***************/
137
138 //read status register 1 (Volatile)
139 uint8_t MEM_readStatus1(){
140   uint8_t status1;
141   status1=MEM_readRegister(MEM_CMD_RDSR1);
142   return status1;
143 }
144
145 //read status register 2
146 uint8_t MEM_readStatus2(){
147   uint8_t status2;
148   status2=MEM_readRegister(MEM_CMD_RDSR2);
149   return status2;
150 }
151
152 //read configuration register 1
153 uint8_t MEM_readConfiguration1(){
154   uint8_t configuration1;
155   configuration1=MEM_readRegister(MEM_CMD_RDCR1);
156   return configuration1;
157 }
158
159 //read configuration register 2
160 uint8_t MEM_readConfiguration2(){
161   uint8_t configuration2;
162   configuration2=MEM_readRegister(MEM_CMD_RDCR2);
163   return configuration2;
164 }
165
166 //read configuration register 3
167 uint8_t MEM_readConfiguration3(){
168   uint8_t configuration3;
169   configuration3=MEM_readRegister(MEM_CMD_RDCR3);
170   return configuration3;
171 }
172
173
174 /************** REGISTER WRITE FUNCTIONS ************/
175
176 //write enable
177 void MEM_writeEnable(){
178   MEM_writeCommand(MEM_CMD_WREN);
179 }
180
181 //write disable
182 void MEM_writeDisable(){
183   MEM_writeCommand(MEM_CMD_WRDI);
184 }
185
186 //write enable volatile
```

```
187 void MEM_writeEnableVolatile(){
188   MEM_writeCommand(MEM_CMD_WRENV);
189 }
190 
191 //clear status register WIP, WEL, P_ERR, E_ERR
192 void MEM_clearStatus(){
193   MEM_writeCommand(MEM_CMD_CLSR);
194 }
195 
196 //enter 4 byte address mode (4byte instructions are not affected)
197 void MEM_4byteEn(){
198   MEM_writeCommand(MEM_CMD_4BEN);
199 }
200 
201 //exit 4 byte address mode (4byte instructions are not affected)
202 void MEM_4byteEx(){
203   MEM_writeCommand(MEM_CMD_4BEX);
204 }
205 
206 /*********** READ FLASH FUNCTIONS ***************/
207 
208 //read n_read bytes from the memory array
209 void MEM_read(uint32_t address, uint16_t n_read, uint8_t* data){
210   uint16_t n = n_read+5;  //number of transmissions equal to n_read + 4 address byte +command byte
211   uint8_t txBuf[261]={0};
212   uint8_t rxBuf[261]={0};   //useful received datas start from position 5
213   SPI_Handle handle;
214 
215   uint16_t txBufLength= sizeof(txBuf);
216 
217   // Open SPI
218   handle=MEM_openSPI();
219 
220   // Associate command to buffers
221   txBuf[0]=MEM_CMD_4READ;
222   txBuf[1]= address>>24;
223   txBuf[2]= address>>16;
224   txBuf[3]= address>>8;
225   txBuf[4]= address;
226 
227   //Write command then read value
228   SPI_readWrite(rxBuf, txBuf, txBufLength, handle);
229 
230   // Close SPI
231   SPI_close(handle);
232 
233   for (int i=0; i<n_read;i++) {
234     //rxBuf[i+5]=i;
235     data[i]=rxBuf[i+5];
236   }
237   //data=rxBuf;
238   rxBuf[0]=0;
239 
240 
241 }
242 /*************** WRITE FLASH FUNCTIONS **************/
243 
244 //Generic size writing
245 //This function will be called by the following ones
246 void MEM_program(uint32_t address, uint8_t * data, uint16_t n_write){
247   uint16_t n = n_write+5;  //number of transmissions equal number of bytes to write + 4 byte
        address + command
248   uint8_t txBuf[261];
```

```
  uint8_t rxBuf[261];
  SPI_Handle handle;

  uint16_t txBufLength= sizeof(txBuf);

  // Open SPI
  handle=MEM_openSPI();

  // Associate command to buffers
  txBuf[0]=MEM_CMD_4PP;
  txBuf[1]= address>>24;
  txBuf[2]= address>>16;
  txBuf[3]= address>>8;
  txBuf[4]= address;

  //Associate data to buffer
  for (int i=5; i<=n; i++){
    txBuf[i]=data[i-5];
  }

  //Write command then write values
  SPI_readWrite(rxBuf, txBuf, txBufLength, handle);

  // Close SPI
  SPI_close(handle);
}

//write one page (256 byte) of the memory array
//before calling this operation we need to enable writing
void MEM_programPage(uint32_t address, uint8_t * data){
  uint16_t n_write =256;
  MEM_program(address, data, n_write);
}

/************ ERASE COMMANDS **************/

//sector erase
void MEM_eraseSector(uint32_t address){
  MEM_writeCommandAddress(MEM_CMD_4SE, address);
}

//half block erase
void MEM_eraseHalfBlock(uint32_t address){
  MEM_writeCommandAddress(MEM_CMD_4HBE, address);
}

//block erase
void MEM_eraseBlock(uint32_t address){
  MEM_writeCommandAddress(MEM_CMD_4BE, address);
}

//chip erase
void MEM_eraseChip(){
  MEM_writeCommand(MEM_CMD_CE);
}

/******** PROGRAM/ERASE SUSPEND/RESUME ************/

//program suspend
void MEM_programEraseSuspend(){
  MEM_writeCommand(MEM_CMD_EPS);
}

```

```
312 //program resume
313 void MEM_programEraseResume(){
314   MEM_writeCommand(MEM_CMD_EPR);
315 }
316 
317 /********** SOFTWARE RESET **************/
318 
319 // software reset
320 void MEM_softwareReset(){
321   MEM_writeCommand(MEM_CMD_RSTEN); //reset enable
322   MEM_writeCommand(MEM_CMD_RST);  // reset
323 }
324 
325 
326 /********* POWER DOWN ****************/
327 
328 //enter in deep power down mode
329 void MEM_deepPowerDown(){
330   MEM_writeCommand(MEM_CMD_DPD);
331 }
332 
333 //release from deep power down mode
334 void MEM_deepPowerDownRelease(){
335   MEM_writeCommand(MEM_CMD_RES);
336 }
```

Bibliography

[1] Y. Barsukov, J. Qian, *Battery Power Management for Portable Devices*, Artech House, 2013. Chapter 1

[2] S.Pregnolato "Real-Time Battery Conditions Estimation - Energetic framework definition and algorithm implementation for real-time determination of the batteries' SoC and SoH", M.S. Thesis, Politecnico di Torino, Torino, Oct 2018

[3] Cadex, "C8000 Battery Testing System. Affordable Lab-Grade Accurcy and Power"[Online]. Available: cadex.com/en/products/c8000-battery-testing-system [Accessed Nov. 24, 2019]

[4] Cadex, "SPECTRO CA-12 Battery Rapid-Tester"[Online]. Available cadex.com/en/products/spectro-ca-12-battery-rapid-tester [Accessed Nov. 24, 2019]

[5] Bosch, "Electronic Battery Sensor"[Online]. Available bosch-mobility/electronic-battery-sensor/[Accessed Nov. 24, 2019]

[6] Midac, "E-MOTION"[Online]. Available: midacbatteries.com/it/news/Midac-lancia-il-kit-e-la-batteria-E-MOTION-41 [Accessed Nov. 24, 2019]

[7] M.Murnane, A Ghazel, *A Closer Look at State of Charge (SOC) and State of Health (SOC) Estimation Technique*, Analog Devices, 2017.[Online Techinal Article] Available: analog.com/A-Closer-Look-at-State-Of-Charge-and-State-Health-Estimation-Techniques

[8] G. Di Simone, C. Radici, "Study of Lead Acid Battery Life Cycle for Automotive", M.S. Thesis, Politecnico di Torino, Torino, Mar 2019

[9] M. Colabella, "Identification and Predictive Analysis of Storage Systems", M.S. Thesis, Politecnic di Torino, Torino, Dec 2019

[10] C Reece, "An Introduction to Electrochemiccal Impedance Spectroscopy (EIS)", Jefferson Lab.[Online] Available: jlab.org/%20intro%20Reece.pdf

[11] Ed. Ramsden, *Hall-Effect Sensors: Theory and Application*, Newnes, 2006. Web, Chapter 7

[12] Maniar, Krunal *Comparing shunt- and hall-based isolated current-sensing solutions in HEV/EV*, Texas Instruments, Jan 2019. [Online paper]. Available: ti.com/lit/an/sbaa293b/sbaa293b.pdf [Accessed November 8, 2019]

[13] Maniar, Krunal *Comparing shunt- and Hall-based current-sensing solutions in onboard chargers and DC/DC converters*, Texas Instruments, Jan 2019. [Online

paper]. Available: ti.com/lit/wp/sbaa318a/sbaa318a.pdf [Accessed November 8, 2019]

[14] R. Nowakowski and R. Taylor, "Linear versus switching regulators in industrial application with a 24-V bus", *Analog Applications Journal*, High-Performance Analog Products, 3Q 2013. [Online]. Available: ti.com/lit/an/slyt527/slyt527.pdf

[15] Texas Instruments, "SIMPLE SWITCHER® Buck Regulators With High-Efficiency ECO Mode", LMR16006Y-Q1 Datasheet, June 2015.

[16] Texas Instruments, "4.5-V to 17-V Input, 500-mA Synchronous Step-Down SWIFT™ Converter With Advanced Eco-mode™", TPS560200-Q1 Datasheet, April 2016 [Revised May 2016]

[17] Texas Instruments, "Automotive Shunt-Based 500A Precision Current Sensing Reference Design", TIDA-03040, Feb 2017 [Revised Jul. 2017].

[18] On Semiconductors, "Effects of High Switching Frequency on Buck Regulators", Dec 2009. [Online presentation]. Available: onsemi.com/pub/Collateral/TND388-D.PDF [Accessed November 12, 2019]

[19] Texas Instrument, "Automotive Grade, 36-V, Bi-Directional, Zero-Drift, High Accuracy, Low- or High-Side, I2C Compatible Current/Power Monitor with Programmable Over-Current Alert", INA226-Q1 Datasheet, Jul 2015.

[20] Texas Instruments, "Automotive-Grade, 36-V Programmable-Gain, Voltage-Output, Bidirectional, Zero-Drift, High-Speed Current-Shunt Monitor", INA225-Q1 Datasheet, Feb 2015.

[21] Texas Instruments, "Automotive Grade, 26-V, Bi-Directional, Zero-Drift, Low- or High-Side, I2C-Compatible Current/Power Monitor", INA220-Q1 Datasheet, Jun 2012 [Rev. Mar 2016].

[22] Texas Instruments, "60-V, Bidirectional, Low- or High-Side, Voltage-Output, Current-Sensing Amplifiers", LMP860x-Q1 Datasheet, Sept 2008 [Rev. Apr 2016].

[23] Vishay, "Power Metal Strip® Battery Shunt Resistor W/Molded Enclosure Very Low Value (50$\mu\Omega$, 100$\mu\Omega$, 125$\mu\Omega$, 500$\mu\Omega$)", WSBM8518 Datasheet, Feb. 2017.

[24] Cypress Semiconductor, "256-Mb (32-MB)/128-Mb (16-MB), 3.0 V FL-L Flash Memory", S25FL256L Datasheet, Jul 2018.

[25] Texas Instruments, "SimpleLink™ Bluetooth® low energy Wireless MCU for Automotive", CC2640R2F-Q1 Datasheet, Jan 2017 [Rev. Aug 2017].

[26] Texas Instruments, "2.4-GHz Inverted F Antenna", Appl. Report SWRU120C, Apr. 2007 [Rev. Feb 2017].

[27] Multicore, "MP218 Solder Paste", MP218 Datasheet, June 2007.

[28] Johanson Technology, "2.4GHz Mini Antenna, SMT", 2450AT18A100 Datasheet, Sept 2018

[29] Texas Instruments, "Low-Power Humidity and Temperature Digital Sensors", HDC2010 Datasheet, Jul 2017 [Rev. May 2019].

[30] Texas Instrument, "Low Power, High Accuracy Digital Humidity Sensor with Temperature Sensor", HDC1080 Datasheet, Nov. 2015 [Rev. Jan 2016].

[31] A. Silberschatz, P.B. Galvin, and G. Gagne, *Operating System Concepts, 8th Edition*, Wiley, 2009, Chapt. 6.

[32] J. Wardall, "Transient Robustness for Current Shunt Monitor", Texas Instruments, TIDA-00302, 2013 [Revised Nov 2014].

[33] Comchip, "Low Profile SMD Schottky Barrier Diode", CDBURT0530LL-HF Datasheet [Rev. B].