POLITECNICO DI TORINO

Master's Degree in Mechatronic Engineering

Master's Degree Thesis

Firmware design of an Electric Vehicle Supply Equipment compliant with UL standards for North-American market

Supervisors

Prof. RADU BOJOI

Ing. MAICOL TONDELLI

Candidate

STEFANO ZICHI

December 2023

Summary

In recent years, the transition to more sustainable mobility has become a topic of public debate. In this context, the significant diffusion of electric vehicles (EVs) has gone hand in hand with the continuous development and research of charging infrastructure. Among the various charging devices, AC wallboxes represent one of the most popular infrastructures due to their versatility and maintenance of high efficiency of charging in both domestic and commercial settings. The thesis work is part of an internal development project of the Research and Development department of the electronics division of Bitron Group S.p.A and the goal is to develop the firmware of an AC wallbox compliant with UL standard.
In the first stage, the work focus was on firmware development, subsequently, the operations were verified with laboratory tests

Table of Contents

List of Tables

List of Figures

Acronyms

AC

Alternating Current

ADC

Analog to Digital Converter

APB

Advanced Peripheral Bus

ARR

AutoReload Register

DAC

Digital to Analog Converter

DC

Direct Current

EV

Electric Vehicle

EVSE

Electric Vehicle Supply Equipment

GPIO

General Purpose InputOutput

IC

Input Capture

IDE

Integrated Development Environment

ISR

Interrupt Service Routine

MCU

Microcontroller Unit

NTC

Negative Temperature Coefficient

PWM

Pulse-Width Modulation

RCM

Residual Current Monitoring

RNG

Random Number Generator

RTC

Real Time Clock

UL

Underwriters Laboratories

Chapter 1

Thesis objective

The thesis work is part of an internal project developed by the Research and Development department of the electronics division of Bitron Group S.p.A.
The globally operating Bitron Group stands out as a leading supplier of electromechanical components for various market sectors. These sectors include Home Appliances, Automotive, HVAC, EV charging and Energy. In each of these areas, Bitron has a solid reputation in the research, development and production of mechatronic and electronic components.
The company has 35 factories scattered around the world and among them, there is the company's headquarters in Grugliasco, the place where I did my internship.

Figure 1.1: Bitron Headquarters, Grugliasco

Specifically, the project consists in design and development of an AC charging wallbox intended for home use, named in the design phase "Wallbox Mini" (shown in Figure 1.2), particular because of its small dimensions.

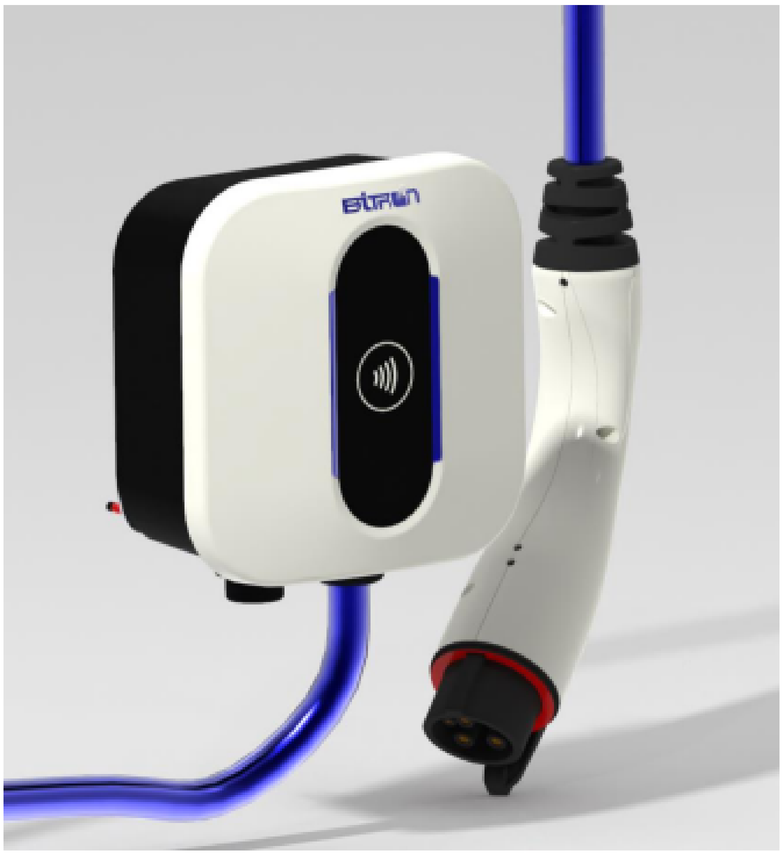


Figure 1.2: "Wallbox Mini" with connector [1]

The product is designed to be sold in the North American market, where safety regulations regarding programmable devices are different than in the European market.
In addition, in the U.S. the household electrical system is two-phase, with a voltage of 120 V and a mains frequency of 60 Hz. Specifically, the design must comply with "UL 2594," which in turn echoes "UL 1998," as will be explained in detail below.
In my thesis work, I contributed to the firmware writing of some applications, specifically those related to:

- Charging state model;
- Temperature sensor;

- Residual Current Monitoring (RCM);
- Relay handler

Chapter 2

Wallbox AC functionality introduction

In recent years, the transition to more sustainable mobility has become a topic of debate involving public opinion.
In this context, the significant spread of electric vehicles (EVs) has gone hand in hand with the continuous development and research of charging infrastructures.
Among the various charging devices, AC wallboxes represent one of the most popular infrastructures due to their versatility and maintenance of high charging efficiency in both domestic and commercial settings.
AC wallboxes allow charging power ranging from 3.7 kW to 22 kW, generally lower values than DC wallboxes. The lower power values and less installation complexity make the prices of AC wallboxes are significantly lower than those of 'competing' DC wallboxes.

2.1 AC Charging Levels

Two levels of AC charging are defined:

- AC Level 1: the charging is single-phase, the voltage is 120 V and the maximum current is 16 A. The rated power delivered is in the range of 1,4 kW to 2,4 kW, due to the limited power, the charging is slow. The AC level 1 can be done using a common household outlet and requires only a charging cable

- AC Level 2: charging is two-phase, the voltage is 240 V and the maximum current is 80 A. The rated power delivered is in the range of 3,7 kW to 22 kW. The AC level 2 can be done with a wallbox and a dedicated installation.

In figure 2.1, is shown the conductive coupler contact interface.In the same figure we can also appreciate:

- Equipment-Chassis Ground: both are aimed at ensuring the safety of the user by connecting the metal parts of the vehicle to the ground;
- Control Pilot: is responsible for managing the various charging phases of the vehicle and regulates the current supplied by the wallbox. An explanation of its functionality can be found in chapter 5;
- Proximity Detection: is responsible for signalling the status of the connection between EVSE and EV, an explanation of its functionality can be found in section 2.3.

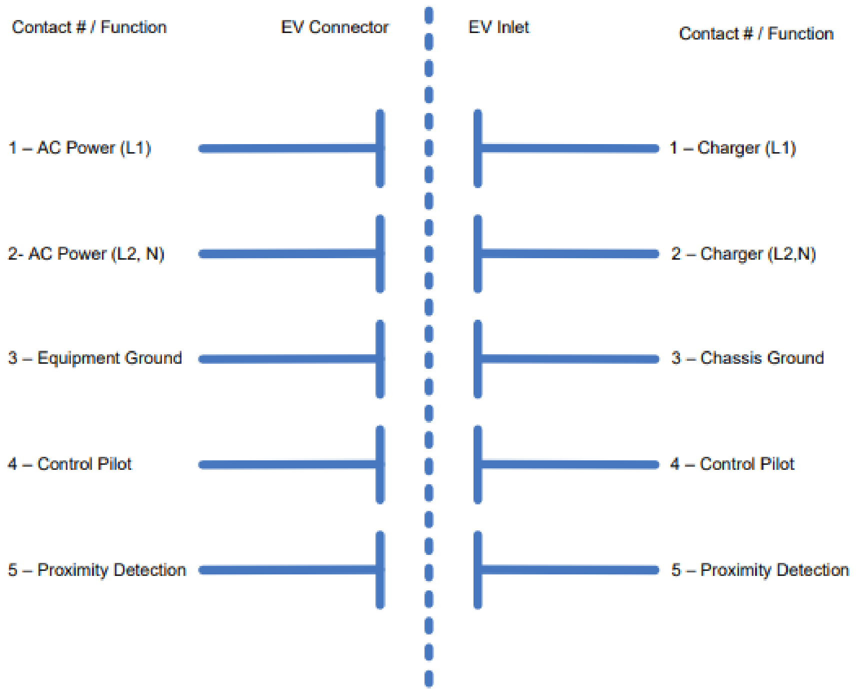


Figure 2.1: On the left are shown the EV connector coupler contact while on the right side the EV inlet coupler contact. The contact number 2 is "N" in AC Level 1 while is L2 in AC Level 2 [2]

2.2 Types of charging connectors

An essential element in the EV charging process is the connector, which is the tool that connects the EVSE to the EV enabling the transfer of energy safely and

efficiently.
There are currently two globally recognized connector standards:

- Type 1: provided with 5 contacts: 3 represent power contacts (L1, L2/N and Protective Earth (PE)) and 2 represent communication contacts (Proximity Pilot (PP) and Control Pilot (CP)). Type 1 connector can be used for single-phase or two-phase charging, it represents the North American and Japanese standards for AC charging stations;

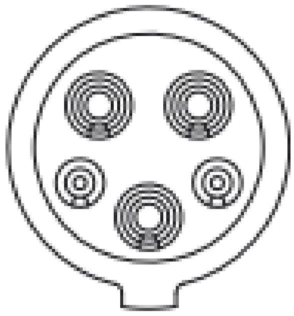


Figure 2.2: Connector Type 1 with 5 contacts [3]

- Type 2: provided with 7 contacts: 5 represent power contacts (L1, L2, L3, N and Protective Earth) and 2 represent communication contacts (Proximity Pilot (PP) and Control Pilot (CP)). Type 2 connector can be used for single-phase or three-phase recharging, it is the European standard for AC charging stations.,

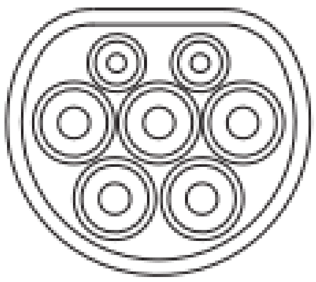


Figure 2.3: Connector Type 2 with 7 contacts [3]

2.3 Proximity Detection

The Proximity Detection, o Proximity Pilot, is a signal referred to chassis ground and indicates the connection status between EVSE and EV.
In figure 2.4 is shown the proximity pilot circuit. The voltage has a nominal value equal to 4.46 V between pins 5 e 3 when the EVSE connector is not connected to the vehicle inlet.
The voltage value is regulated by switch S3:

- when S3 is opened, the voltage between pins 3 and 5 has a nominal value equal to 2.77 V;
- when S3 is closed, the voltage between pins 3 and 5 has a nominal value equal to 1.53 V.

The voltage value decreases because when S3 is open, the value of the parallel resistance in the common part is larger.
Also in figure 2.4, we see that a common ground reference between EVSE and EV is ensured during the connection.

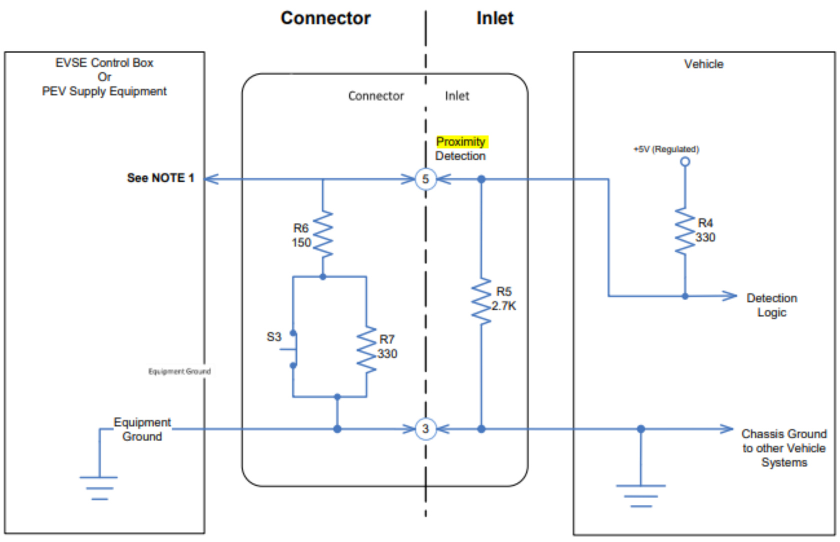


Figure 2.4: Detection Pilot Circuit [2]

Chapter 3

Wallbox Mini description

The "Wallbox Mini" project pursued by Bitron has the objective to develop a small-sized AC charging wallbox (as shown in the figure 3.1), capable of delivering rated power up to 11.5 kW depending on the variant installed.
Other technical specifications, such as voltage-current ratings and possible communication technologies, are shown in table 3.1.

Power	9,6 kW	11,5 kW
AC voltage	208-240 V	208-240 V
AC current	0-40 A	0-48 A
Frequency	60 Hz	60 Hz
Voltage tolerance	±10%	±10%
Connectivity	Wi-fi, Bluetooth, LTE 4G	Wi-fi, Bluetooth, LTE 4G

Table 3.1: "Wallbox mini" technical specifications, referred to North-America rated parameters and for two different power variants

The "Wallbox Mini" is an AC Level 2 EVSE, compliant with SAE-J1772 standard. As a product for sale in the North American market, the power line is two-phase and requires a supply voltage of 240 V.
The connector chosen for current delivery is a Type 1 connector, compliant with SAE J1772 and UL standards and already described in the section 2.2.
The connector chosen for the wallbox is shown in figure 3.2.

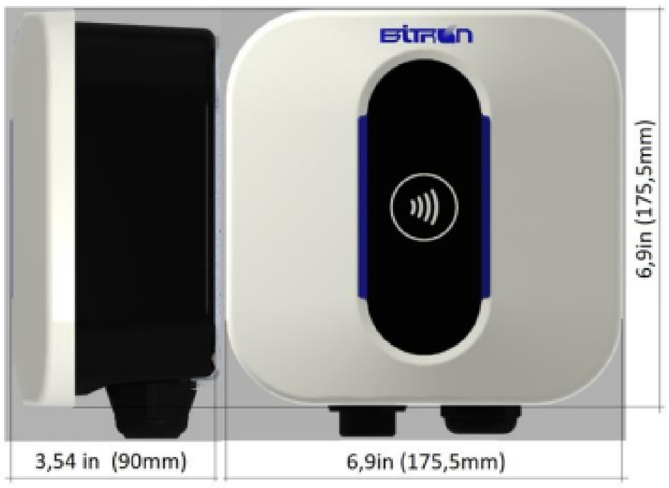


Figure 3.1: On the left is shown the wallbox side view and on the right the wallbox front part, both with related measurements [1]

Figure 3.2: Photo of "Wallbox Mini" connector

3.1 STM32MP15x microprocessor family

A microprocessor from the STM32MP15x family made by STMicroelectronics was chosen for the development of the project.
The STM32MP15x line is based on the flexible architecture of an Arm Cortex-A7 processor and Cortex-M4 processor.
The use of this device is very common in a variety of applications because of the low power consumption and low cost, as well as advanced security features.

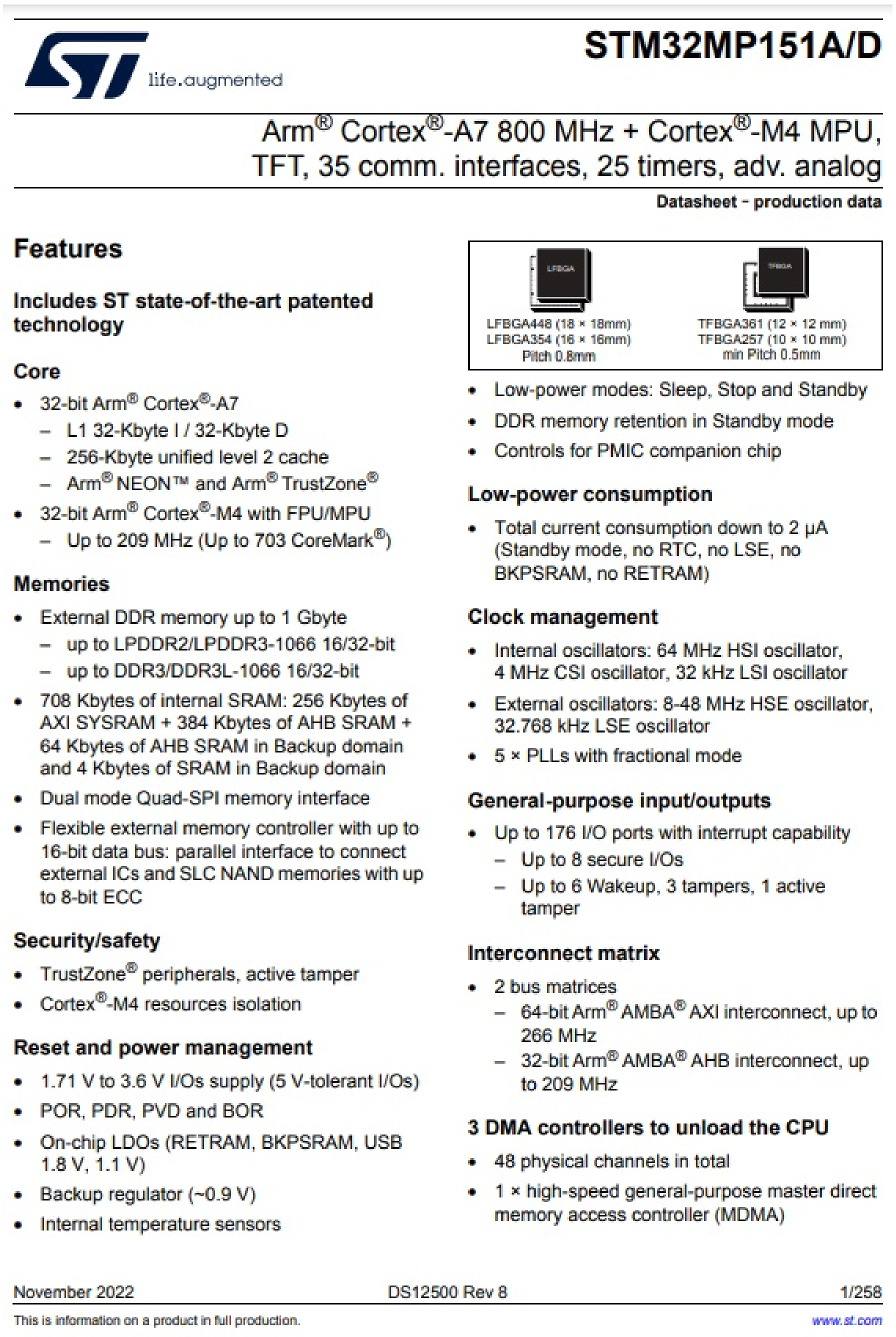
In more in details, the STM32MP151AAA variant used in the current project consists of a single core Arm Cortex-A7 650 MHz combined with Cortex-M4 at 209 MHz and includes:

- the OpenSTLinux distribution, a Linux distribution, running on the Arm Cortex-A7 processor. This feature gives benefits in terms of speed of development and ductility to develop the high-level programming parts, which are related to the communications (Wi-fi, LTE, Bluethoot);
- the STM32CubeMPU Package, running on the Arm CortexM4 processor. In Arm CortexM4 is running the firmware without an operating system and implemented the real timing function.

This kind of microprocessor is chosen due to the possibility of having the parallel execution of the two cores. The microprocessor is equipped with external SRAM, embedded SDRAM and DDR memories and some of the features are listed below:

- two ADCs;
- two DACs;
- a low power RTC;
- 12 general purpose 16-bit timer;
- two PWM timers for motor control;
- five low power timers;
- 176 GPIOs;

This microprocessor family support also communication interfaces such as UART, USART, SPI and I2C.



STM32MP151A/D

life.augmented

Arm® Cortex®-A7 800 MHz + Cortex®-M4 MPU,
TFT, 35 comm. interfaces, 25 timers, adv. analog

Datasheet - production data

Features

Includes ST state-of-the-art patented technology

Core

- 32-bit Arm® Cortex®-A7
  - L1 32-Kbyte I / 32-Kbyte D
  - 256-Kbyte unified level 2 cache
  - Arm® NEON™ and Arm® TrustZone®
- 32-bit Arm® Cortex®-M4 with FPU/MPU
  - Up to 209 MHz (Up to 703 CoreMark®)

Memories

- External DDR memory up to 1 Gbyte
  - up to LPDDR2/LPDDR3-1066 16/32-bit
  - up to DDR3/DDR3L-1066 16/32-bit
- 708 Kbytes of internal SRAM: 256 Kbytes of AXI SYSRAM + 384 Kbytes of AHB SRAM + 64 Kbytes of AHB SRAM in Backup domain and 4 Kbytes of SRAM in Backup domain
- Dual mode Quad-SPI memory interface
- Flexible external memory controller with up to 16-bit data bus: parallel interface to connect external ICs and SLC NAND memories with up to 8-bit ECC

Security/safety

- TrustZone® peripherals, active tamper
- Cortex®-M4 resources isolation

Reset and power management

- 1.71 V to 3.6 V I/Os supply (5 V-tolerant I/Os)
- POR, PDR, PVD and BOR
- On-chip LDOs (RETRAM, BKPSRAM, USB 1.8 V, 1.1 V)
- Backup regulator (~0.9 V)
- Internal temperature sensors

LFBGA448 (18 × 18mm)
LFBGA354 (16 × 16mm)
Pitch 0.8mm

TFBGA361 (12 × 12 mm)
TFBGA257 (10 × 10 mm)
min Pitch 0.5mm

- Low-power modes: Sleep, Stop and Standby
- DDR memory retention in Standby mode
- Controls for PMIC companion chip

Low-power consumption

- Total current consumption down to 2 µA (Standby mode, no RTC, no LSE, no BKPSRAM, no RETRAM)

Clock management

- Internal oscillators: 64 MHz HSI oscillator, 4 MHz CSI oscillator, 32 kHz LSI oscillator
- External oscillators: 8-48 MHz HSE oscillator, 32.768 kHz LSE oscillator
- 5 × PLLs with fractional mode

General-purpose input/outputs

- Up to 176 I/O ports with interrupt capability
  - Up to 8 secure I/Os
  - Up to 6 Wakeup, 3 tampers, 1 active tamper

Interconnect matrix

- 2 bus matrices
  - 64-bit Arm® AMBA® AXI interconnect, up to 266 MHz
  - 32-bit Arm® AMBA® AHB interconnect, up to 209 MHz

3 DMA controllers to unload the CPU

- 48 physical channels in total
- 1 × high-speed general-purpose master direct memory access controller (MDMA)

November 2022 DS12500 Rev 8 1/258

This is information on a product in full production. www.st.com

Figure 3.3: STM32MP151AAA datasheet [4]

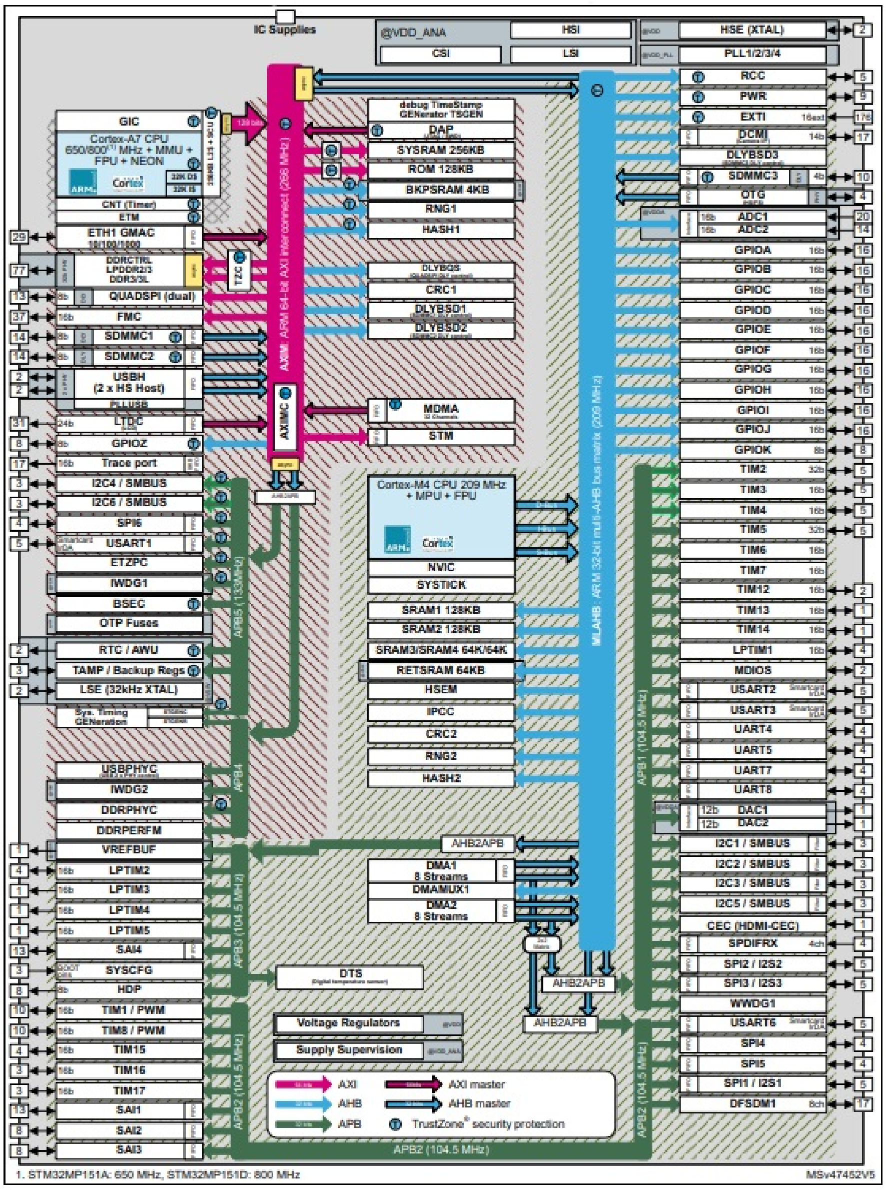

IC Supplies
@VDD_ANA
HSI
HSE (XTAL)
CSI
LSI
PLL1/2/3/4
GIC
Cortex-A7 CPU 650/800(1) MHz + MMU + FPU + NEON
AXIM: ARM 64-bit AXI interconnect (266 MHz)
MLAHB: ARM 32-bit multi-AHB bus matrix (209 MHz)
Cortex-M4 CPU 209 MHz + MPU + FPU
APB1 (104.5 MHz)
APB2 (104.5 MHz)
APB3 (104.5 MHz)
APB4
APB5 (133MHz)
AXI
AHB
APB
AXI master
AHB master
TrustZone® security protection
1. STM32MP151A: 650 MHz, STM32MP151D: 800 MHz
MSv47452V5

Figure 3.4: STM32MP151AAA block diagram [4]

3.2 Cortex-M4

Cortex-M4 is one of the processor cores in the ARM Cortex-M family.
Because of 'high efficiency and low power consumption, it is one of the most popular cores in various applications, such as automotive, industrial automation, or power management. Based on the ARMv7E-M architecture, it provides high performance

in embedded applications.
With an internal clock frequency of 209 MHz, it has two hardware components such as the FPU and MPU, which specifically are:

- Floating Point Unit (FPU): enables complex mathematical operations in which decimal numbers are involved by combining high accuracy and high speed;
- Memory Protection Unit (MPU): ensures the protection of memory and manages its' access in different areas.

3.2.1 Firmware architecture

For firmware development, it was decided to define a layer software architecture consisting of modules dependent on each other according to a certain logic and hierarchy.
The purpose is to obtain:

- Portability: Application-level modules can be ported from one platform to another without any changes on the code;
- Debugging: in the presence of a bug, it will be easier to isolate the problem and take action to fix it without interfering with other functionality;
- Software process certifiability: having a software architecture is the basis for having a software development process that can eventually also be certified (see UL standards case);
- Testability: each module can be tested independently;
- Block development: as the needs of different wallboxes change, only certain modules that interface with the device will be modified. In addition, interventions can be divided among resources in a clearer and more defined manner;
- Readability: a module division allows better interpretation and understanding for those who will have to work with firmware in the future.

There are 3 layers identified:

- Application layer: this layer contains the modules that invoke the functions of the BSP layer, handling its scheduling. They are portable modules, since they are not developed to act directly on hardware and they can be ported from one platform to another;

- Board Support Package (BSP) layer: this layer contains the modules that interface directly to the hardware, commanding its operations. The modules must be adapted according to the hardware specifications of the device;
- HAL layer: this layer contains the functions provided by STMicroelectronics libraries. The HAL functions allow direct access to microcontroller memory, manage peripherals, interrupts, etc.... .

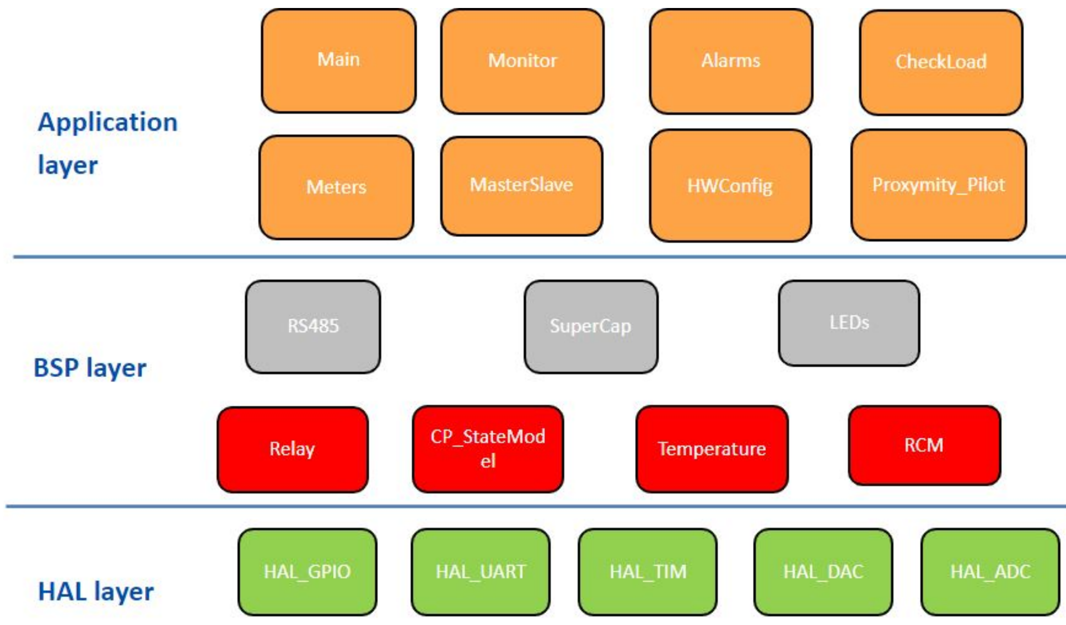


Figure 3.5: Layer division of the firmware architecture with some of the modules

A subdivision such as the one described in figure 3.5 made it easier to manage the work and divide it among the various resources.
In the BSP layer, the modules I personally took care of are highlighted in red.

3.3 Standards for Safety

The final product "Wallbox Mini" must satisfies the requirements of UL standards. Underwriters Laboratories (UL) is a leading organization in the field of safety evaluation and certification based in the United States.
UL's goal is to enhance the safety of people through the evaluation and certification of products before they enter in North-America market.

Figure 3.6: UL logo [5]

3.3.1 UL 2594

The "UL 2594" is the standard for Safety for Electric Vehicle Supply Equipment. This standard aims to provide safety standards for all supply equipment with a primary voltage of 600 V or less that intend to provide AC power to an electric vehicle with an on-board charging unit.
The "UL 2594" states [5] that "When such circuits employ a microprocessor executing software to perform the safety-related function, the software shall comply with the requirements in Annex A, Ref. No. 54".
The "Annex A, Ref. No. 54" is referred to another UL standard, the "UL 1998". Since the project's safety functions are executed by a programmable component, the requirements specified in "UL 1998" must be satisfied.

3.3.2 UL 1998

The "UL 1998" is the standard for Safety for Software in Programmable Components. The standard identifies two classes of software:

- class 1: Sections of software intended to control function to reduce the likelihood of a risk associated with the equipment [6];

- class 2: – Sections of software intended to control functions to reduce the likelihood of special risks (e.g., explosion) associated with the equipment [6].

The "UL 1998" requires risk analysis, process definition, software analysis and among this requirements there is the [6]: "Measures To Address Microelectronic Hardware Failure Modes". This part of standard states that the following physical failures must be considered [6]:

- CPU registers, instruction decoding and execution, program counter, addressing and data paths;
- Interrupt handling and execution;
- Clock;
- Non-volatile and volatile memory and memory addressing;
- Internal data path and data addressing;
- External communication and data, addressing, and timing;
- Input/output devices such as analog I/O, D/A and A/D converters, and analog multiplexers;
- Monitoring devices and comparitors;
- Application-Specific Integrated Circuits (ASICs), Gate Array Logics (GALs), Programmable Logic Arrays (PLAs), and Programmable Gate Arrays (PGAs) hardware

For each physical failure are specified examples of acceptable measures that can be taken in order to avoid the failure.
In order to satisfy the requirements imposed by UL, a "self-test library" from STMicroelectronics certified by UL itself was used, which takes all sufficient measures to account for the previously listed faults.
Furthermore, each security feature was documented for certification purposes. In this context, I was responsible for the firmware development of two applications related to two programmable safety devices on the wallbox: the RCM and the Relay.

Chapter 4

Development software for STMicroprocessor

For code development and peripheral configuration were used two software developed by STMicroelectronics for applications based on STM32 family microcontrollers, STM32CubeMX and STM32CubeIDE.

4.1 STM32CubeMX

The STM32CubeMX software is a tool used for generating the initialization code. The software, through its intuitive graphical interface, allows easy configuration of peripheral devices via the pinout view (fig.4.1), giving to the user the possibility to choose which functionality to allocate to each GPIO pins.

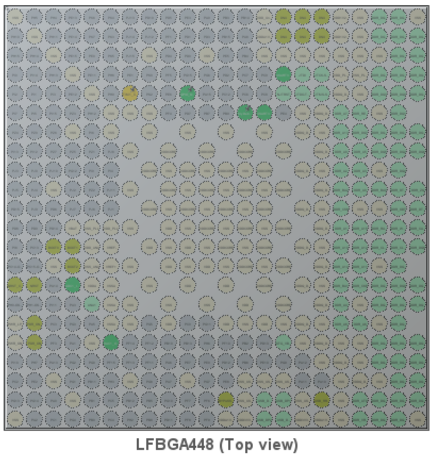
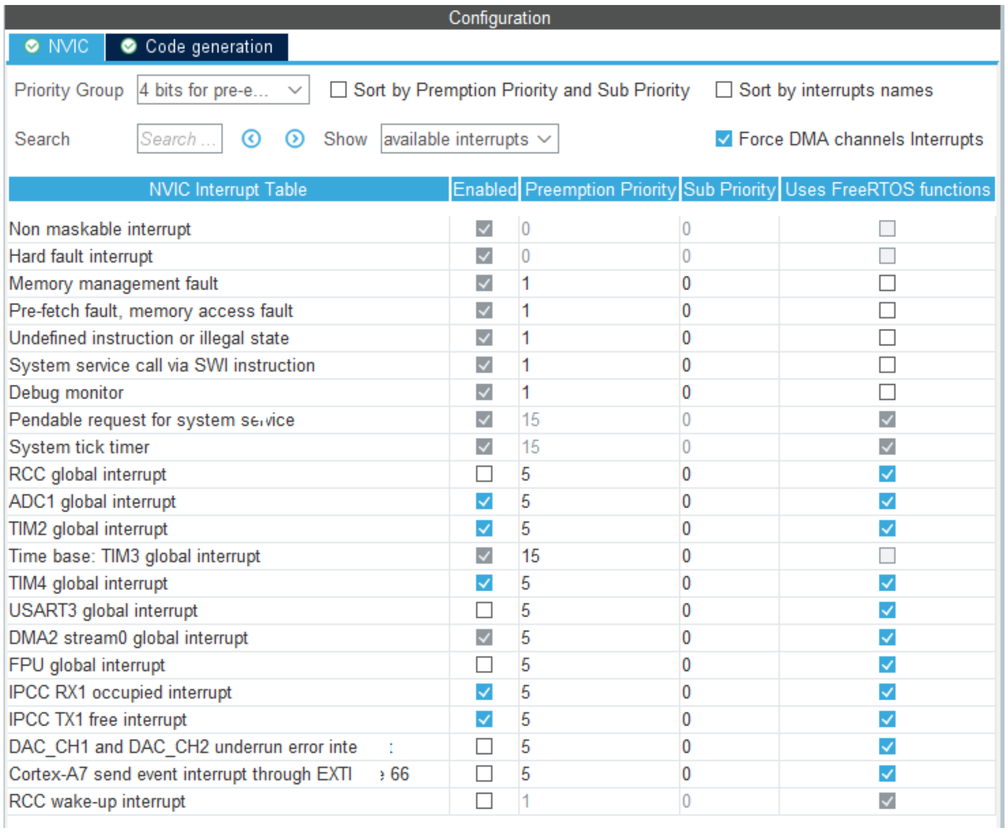


Figure 4.1: STM32CubeMX Main Screen

Among the various functions that can be assigned to the pins, those of the Timer and ADC converter are the ones that have been used for the work done. In addition, each pin can be assigned in the design phase to either Cortex-A7 or Cortex-M4.
Once the function that is due to each pin has been chosen, it is evaluated whether and which peripherals should be activated in interrupt mode.In this way, when a specific event occurs (such as a change of state on a GPIO), an interrupt signal is generated that interrupts normal program execution to carry out the Interrupt Service Routine (ISR). An example of an interrupt table is shown in figure fig.4.1.



Priority Group: 4 bits for pre-e...
☐ Sort by Premption Priority and Sub Priority
☐ Sort by interrupts names
Search: Search ...
Show: available interrupts
✓ Force DMA channels Interrupts

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority	Uses FreeRTOS functions
Non maskable interrupt	✓	0	0	☐
Hard fault interrupt	✓	0	0	☐
Memory management fault	✓	1	0	☐
Pre-fetch fault, memory access fault	✓	1	0	☐
Undefined instruction or illegal state	✓	1	0	☐
System service call via SWI instruction	✓	1	0	☐
Debug monitor	✓	1	0	☐
Pendable request for system service	✓	15	0	✓
System tick timer	✓	15	0	✓
RCC global interrupt	☐	5	0	✓
ADC1 global interrupt	✓	5	0	✓
TIM2 global interrupt	✓	5	0	✓
Time base: TIM3 global interrupt	✓	15	0	☐
TIM4 global interrupt	✓	5	0	✓
USART3 global interrupt	☐	5	0	✓
DMA2 stream0 global interrupt	✓	5	0	✓
FPU global interrupt	☐	5	0	✓
IPCC RX1 occupied interrupt	✓	5	0	✓
IPCC TX1 free interrupt	✓	5	0	✓
DAC_CH1 and DAC_CH2 underrun error inte :	☐	5	0	✓
Cortex-A7 send event interrupt through EXTI 66	☐	5	0	✓
RCC wake-up interrupt	☐	1	0	✓

Figure 4.2: STM32CubeMX Interrupt table

Once the peripherals are configured, the STM32CubeMX generates the initialization code containing all the configurations set in the Grafical User Interface (GUI). In the specific case of the STM32MP15x microcontroller family, two projects are generated: one for the Cortex-A7 and one for the Cortex-M4. As shown in figure 4.3, three folders are also generated.:

- Common: contains common code that can be used by the various projects, such as the libraries;
- Drivers: contains source files for drivers used by activated peripherals;
- Middlewares: contains additional libraries used to extend the functionality of the microcontroller, such as communication protocols or operating systems.

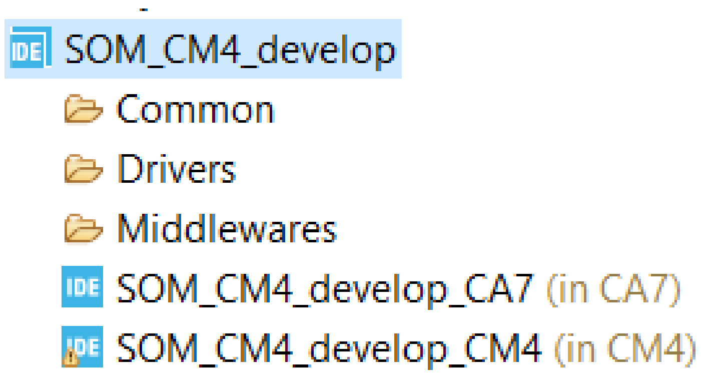


Figure 4.3: Project folders created by STM32CubeMX after code generation

4.1.1 Timer configuration

The timers I used are general-purpose timers, each of which has 4 channels that can be set in different modes, including:

- Input capture mode;
- Output compare no output;
- Output compare;
- PWM generation no output;
- PWM generation;

Each timer has a 32-bit or 16-bit counter, and the maximum count value can be set by changing the value of ARR. In addition, they have a clock frequency that varies according to the bus they belong to and can be changed in the clock interface configuration (figure 4.4) .

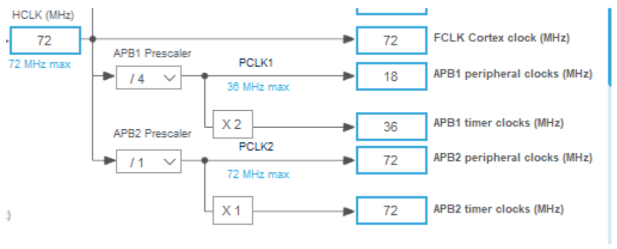


Figure 4.4: A screen of clock configuration settings. The values in the cells represent the internal clock for the peripheral that stand in that specific APB.

The count rate can be changed by entering a prescaler value, which will divide the clock frequency by the entered value minus one unit. The prescaler is based on a 16 bits counter.

$$f_{timer} = \frac{f_{clock}}{(Prescaler - 1)} \tag{4.1}$$

Even in GUI, there be the possibility to choose the mode of counting:

- Up counting: from zero to the ARR value;
- Down Counting: from ARR value to zero.

4.2 STM32CubeIDE

The STM32CubeIDE software is an Integrated Development Environment (IDE) dedicated to programming and debugging STM32 microcontrollers and microprocessors.
The software integrates the configuration and functionality of the project generated through STM32CubeMX, these can be changed at any time by the 'user through code regeneration, without losing the parts of private code developed so far. The files are divided into areas where there is generated code and those where the user has to write his part, as shown in the code part below.

```
/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_USART2_UART_Init();
MX_TIM2_Init();
MX_TIM3_Init();
MX_TIM4_Init();
/* USER CODE BEGIN 2 */

/* USER CODE END 2 */
```

Chapter 5

Control Pilot module

The control pilot is the primary control means to ensure proper operation when connecting an EV to the EVSE and is referred to EVSE.
The control pilot module has the task of:

- manage the state machine of charging and thus the various phases: to each state corresponds a different maximum voltage level of PWM;
- adjust the current that the EVSE delivers to the vehicle.

According to SAE J1772, in the charging phase, each state corresponds to a voltage level, and power is transferred via a PWM signal.

5.1 Charging state model

The charging state model has the following states [2]:

- State A: the vehicle is not connected. In this state the signal is a static voltage with a nominal value of 12 V;
- State B1: the vehicle is connected but is not ready to accept energy, while the EVSE is not ready to supply energy. In this state the signal is a static voltage with a nominal value of 9 V;
- State B2: the vehicle is connected but is not ready to accept energy, while the EVSE is ready to supply energy. In this state the signal is a PWM at 1 kHz, with a maximum value equal to 9 V and a minimun value equal to -12 V;
- State C: the vehicle is connected but and is ready to accept energy, also the EVSE is ready to supply energy. In this state the signal is a PWM at 1 kHz, with a maximum value equal to 6 V and a minimun value equal to -12 V;

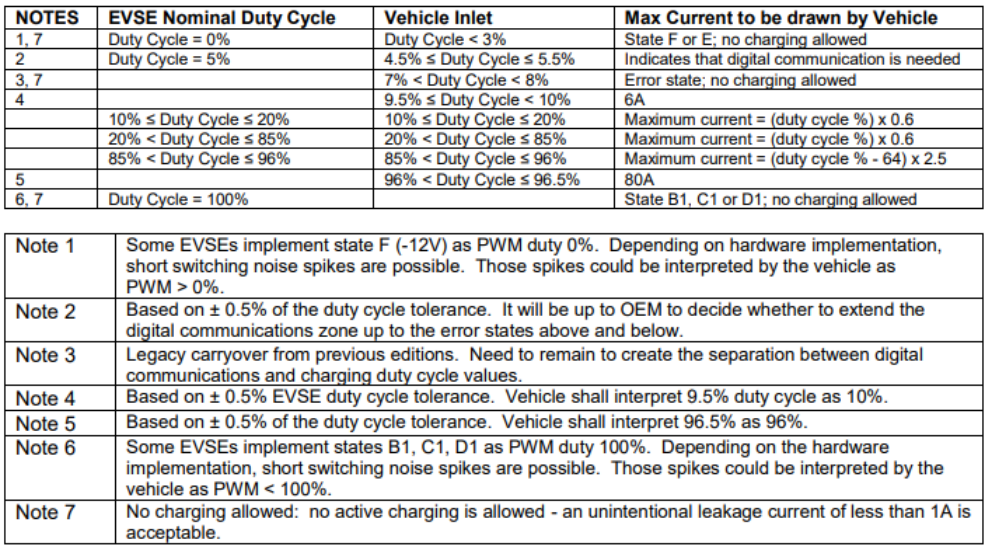
- State D: this state is equal to state C with the difference that the EV requires indoor ventilation in charging area. In this state the signal is a PWM at 1 kHz, with a maximum value equal to 3 V and a minimun value equal to -12 V;
- State E: is an error state, it occurs when the EVSE is diconnected from vehicle and there is a loss of electric power.In this state the signal is a static voltage with a nominal value of 0 V;
- State F: is a generic error state and the EVSE is not capable to supply energy due to different faults.In this state the signal is a static voltage with a nominal value of -12 V;

In tab 5.1 are shown the values of state voltages with the corrispective minimum and maximum value that they can have during normal operations:

STATE	MIN. VOLTAGE [V]	NOM. VOLTAGE [V]	MAX. VOLTAGE [V]
State A	11.40	12.00	12.60
State B1	8.36	9.00	9.59
State B2	8.36	9.00	9.59
State C	5.47	6.00	6.53
State D	2.58	3.00	3.28
State F	-12.60	-12.00	-11.40

Table 5.1: Range value of state voltages [2]

As described earlier, the control pilot is also responsible for managing the amount of current to be delivered to the vehicle. This is done by changing the duty cycle value of the PWM to the states corresponding to the charging phase, i.e., state C (if no ventilation is required) or D (if ventilation is required).
The duty cycle value is modulated according to the conversion formulas shown in figure 5.1.



NOTES	EVSE Nominal Duty Cycle	Vehicle Inlet	Max Current to be drawn by Vehicle
1, 7	Duty Cycle = 0%	Duty Cycle < 3%	State F or E; no charging allowed
2	Duty Cycle = 5%	4.5% ≤ Duty Cycle ≤ 5.5%	Indicates that digital communication is needed
3, 7		7% < Duty Cycle < 8%	Error state; no charging allowed
4		9.5% ≤ Duty Cycle < 10%	6A
	10% ≤ Duty Cycle ≤ 20%	10% ≤ Duty Cycle ≤ 20%	Maximum current = (duty cycle %) x 0.6
	20% < Duty Cycle ≤ 85%	20% < Duty Cycle ≤ 85%	Maximum current = (duty cycle %) x 0.6
	85% < Duty Cycle ≤ 96%	85% < Duty Cycle ≤ 96%	Maximum current = (duty cycle % - 64) x 2.5
5		96% < Duty Cycle ≤ 96.5%	80A
6, 7	Duty Cycle = 100%		State B1, C1 or D1; no charging allowed

Note 1	Some EVSEs implement state F (-12V) as PWM duty 0%. Depending on hardware implementation, short switching noise spikes are possible. Those spikes could be interpreted by the vehicle as PWM > 0%.
Note 2	Based on ± 0.5% of the duty cycle tolerance. It will be up to OEM to decide whether to extend the digital communications zone up to the error states above and below.
Note 3	Legacy carryover from previous editions. Need to remain to create the separation between digital communications and charging duty cycle values.
Note 4	Based on ± 0.5% EVSE duty cycle tolerance. Vehicle shall interpret 9.5% duty cycle as 10%.
Note 5	Based on ± 0.5% of the duty cycle tolerance. Vehicle shall interpret 96.5% as 96%.
Note 6	Some EVSEs implement states B1, C1, D1 as PWM duty 100%. Depending on the hardware implementation, short switching noise spikes are possible. Those spikes could be interpreted by the vehicle as PWM < 100%.
Note 7	No charging allowed: no active charging is allowed - an unintentional leakage current of less than 1A is acceptable.

Figure 5.1: All possible duty cycle ranges with relative indication of current that can be drawn by the vehicle [2]

In figure 5.2 is shown the value maximum value of current that can be drawn by the vehicle in function of duty cycle value.
Is notable that the minimum value of current is equal to 6 A at 9.5% of duty cycle, while the maximum value is 80 A at 96.5% of duty cycle.

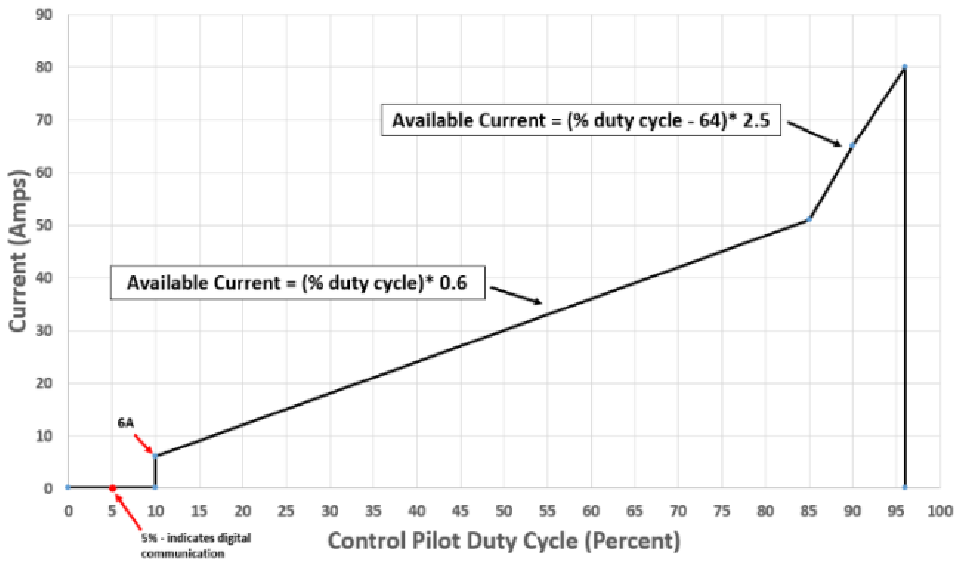
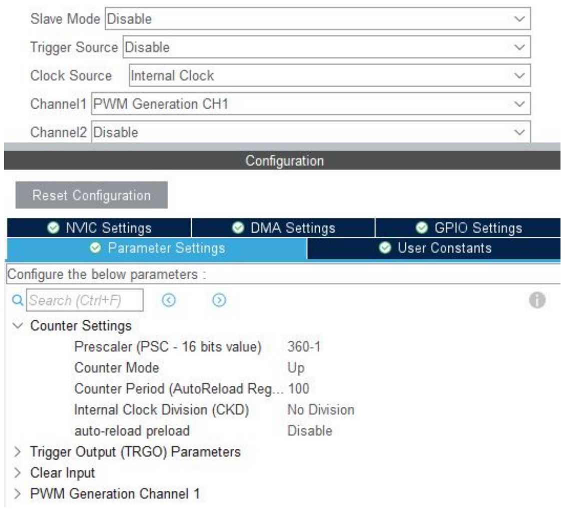


Figure 5.2: Characteristic of current in function of duty cycle[2]

5.2 Peripheral configuration

To manage this module, I used a timer with a clock frequency of 36 MHz.
I configured one channel in PWM generation, inserted a prescaler of (360-1) and an ARR of 100, in order to obtain a PWM output with has a frequency of 1 KHz. The settings are shown in figure 5.3.



Slave Mode: Disable
Trigger Source: Disable
Clock Source: Internal Clock
Channel1: PWM Generation CH1
Channel2: Disable

Counter Settings

Prescaler (PSC - 16 bits value)	360-1
Counter Mode	Up
Counter Period (AutoReload Reg...	100
Internal Clock Division (CKD)	No Division
auto-reload preload	Disable

Figure 5.3: Timer configuration settings

5.3 Module description

As described in 5.1, the control pilot manages the transition between charging states and to adjust the current la to be delivered to the EV.
The following is an implementation of a part of the function *CpStateMachine_Task()*, which is the function that manages the state machine. These lines handle the startup of the state machine and the states transition until state B2.
As can be seen, firstly the function checks that the state machine is not in a fault state (corresponding to state F).
If the state machine has no problems, inputs and outputs are initialized and the voltage level received from the vehicle is evaluated, which is acquired from another module via an ADC peripheral.
Next, the starting state is set and for each of these and it is checked that the acquired voltage value is within the upper and lower limits of the considered state.
If this condition is not met, the next or previous state is passed to.
Within each state, it is evaluated whether the voltage takes undefined values.
If undefined state occurs, the state machine is reinitialized.

At the end of each state, the PWM is triggered and a certain duty cycle value is assigned (coming from another firmware module).
The value of the duty cycle that adjusts the amount the current delivered by the EVSE to the vehicle.

```
void CPStateMachine_Task( void )
 {
    if( CpSm.cpsm_input.cmd[IN_CMD_FAULT] == true   )
    {
        CpSm.cpsm_state = CPSM_STATE_F;
        CpSm.SetCPHighLvlFunct(-12000);
        __HAL_TIM_SET_AUTORELOAD(&htim2, 100);
        CpSm.cpsm_output.ind[OUT_IND_PWM_ENABLED] = false;
    }
    switch( CpSm.cpsm_state )
    {
        case CPSM_STATE_INIT:
            /* init inputs */
            CpSm.cpsm_input.cmd[IN_CMD_READY] = false;
            CpSm.cpsm_input.cmd[IN_CMD_FAULT] = false;
            CpSm.cpsm_input.cmd[IN_CMD_RESET] = false;

            /* init outputs */
            CpSm.cpsm_output.ind[OUT_IND_SIMPLE_CIRCUIT] = false;
            CpSm.cpsm_output.ind[OUT_IND_PWM_ENABLED] = false;

            CpSm.cpsm_state = CPSM_CHECK_STATE;
        break;

        case CPSM_CHECK_STATE:
    if( cp_voltage_mV > CpSm_Limits_mV[LIMIT_HIGH][CPSM_LIMITS_A] )
            { /* voltage too high...*/
                /* cpsm remains in this state*/
            }
            else if( cp_voltage_mV < CpSm_Limits_mV[LIMIT_HIGH][
    CPSM_LIMITS_BX]   )
            {
                if( cp_voltage_mV < CpSm_Limits_mV[LIMIT_HIGH][
    CPSM_LIMITS_C] )
                {
                    if( cp_voltage_mV < CpSm_Limits_mV[LIMIT_HIGH][
    CPSM_LIMITS_D] )
                    {
                        if( ( cp_voltage_mV < CpSm_Limits_mV[
    LIMIT_HIGH][CPSM_LIMITS_E] ) &&
                            ( cp_voltage_mV > CpSm_Limits_mV[
    LIMIT_LOW][CPSM_LIMITS_E] )      )
```

```
                        {
                            CpSm.cpsm_state = CPSM_STATE_E;
                        }
                        else
                        {
                            CpSm.cpsm_state = CPSM_STATE_D;
                        }
                    }
                    else
                    {
                        CpSm.cpsm_state = CPSM_STATE_C;
                    }
                }
                else
                {
                    CpSm.cpsm_state = CPSM_STATE_B1;
                }
            }
                    else
                    {
                        CpSm.cpsm_state = CPSM_STATE_A;
                    }
        break;

        case CPSM_STATE_A:
        if( cp_voltage_mV > CpSm_Limits_mV[LIMIT_HIGH][CPSM_LIMITS_A]
    )
        { /* voltage too high...*/
            CpSm.cpsm_state = CPSM_STATE_UNDEF;
        }
        else if( cp_voltage_mV < CpSm_Limits_mV[LIMIT_HIGH][
   CPSM_LIMITS_BX]   )
        {
            if( cp_voltage_mV < CpSm_Limits_mV[LIMIT_HIGH][
   CPSM_LIMITS_C]  )
            {
                if( cp_voltage_mV < CpSm_Limits_mV[LIMIT_HIGH][
   CPSM_LIMITS_D]  )
                {
                    if( ( cp_voltage_mV < CpSm_Limits_mV[LIMIT_HIGH][
   CPSM_LIMITS_E]  ) &&
                        ( cp_voltage_mV > CpSm_Limits_mV[LIMIT_LOW][
   CPSM_LIMITS_E]  )     )
                    {
                        CpSm.cpsm_state = CPSM_STATE_E;
                    }
                    else
                    {
                        CpSm.cpsm_state = CPSM_STATE_D;
                    }
                }
                else
                {
                    CpSm.cpsm_state = CPSM_STATE_C;
                }
            }
            else
            {
                CpSm.cpsm_state = CPSM_STATE_B1;
            }
        }
        else
        {
            /* cpsm remains in this state*/
        }

        if( CpSm.cpsm_output.ind[OUT_IND_PWM_ENABLED] == true )
        {
            __HAL_TIM_SET_AUTORELOAD(&htim2, 100);
            CpSm.cpsm_output.ind[OUT_IND_PWM_ENABLED] = false;
        }

        break;

        case CPSM_STATE_B:
         if( cp_voltage_mV > CpSm_Limits_mV[LIMIT_LOW][CPSM_LIMITS_A]
     )
        {
            if( cp_voltage_mV > CpSm_Limits_mV[LIMIT_HIGH][
    CPSM_LIMITS_A] )
            {
                /* voltage too high...*/
                CpSm.cpsm_state = CPSM_STATE_UNDEF;
            }
            else
            {
                CpSm.cpsm_state = CPSM_STATE_A;
            }
        }
        else if( cp_voltage_mV < CpSm_Limits_mV[LIMIT_HIGH][
    CPSM_LIMITS_C]  )
        {
            if(  cp_voltage_mV < CpSm_Limits_mV[LIMIT_HIGH][
    CPSM_LIMITS_D] )
            {
                if( cp_voltage_mV < CpSm_Limits_mV[LIMIT_HIGH][
    CPSM_LIMITS_E] )
                {
                    if( cp_voltage_mV > CpSm_Limits_mV[LIMIT_LOW][
    CPSM_LIMITS_E] )
                    {
                        CpSm.cpsm_state = CPSM_STATE_E;
                    }
                    else
                    { /* voltage below minimum E state */
                        CpSm.cpsm_state = CPSM_STATE_UNDEF;
                    }
                }
                else
                {
                    CpSm.cpsm_state = CPSM_STATE_D;
                }
            }
            else
            {
                CpSm.cpsm_state = CPSM_STATE_C;
            }
        }
        else
        {
            /* remains in this state */
        }

        if( CpSm.cpsm_input.cmd[IN_CMD_READY] == true ) // STATE B2
        {
            __HAL_TIM_SET_AUTORELOAD(&htim2, cp_duty_target);
            CpSm.cpsm_output.ind[OUT_IND_PWM_ENABLED] = true;
        }
        else //STATE B1
        {
            __HAL_TIM_SET_AUTORELOAD(&htim2, 100);
            CpSm.cpsm_output.ind[OUT_IND_PWM_ENABLED] = false;
        }
        break;
    }
}
```

5.4 Functionality test

In order to verify the wallbox behavior with respect to the vehicle, the test was performed with a car simulator. The car simulator is a device that can change the state of the charging operation by using a switch.
The objective is to verify if the firmware responds correctly and meets the vehicle's demands
In figure 5.4 is shown the B2 state. The PWM has a maximum value equal to 9 V, a minimum value equal to -12 V and a period of 1 kHz. In this state EVSE and EV are connected but the vehicle is not ready to accept energy.

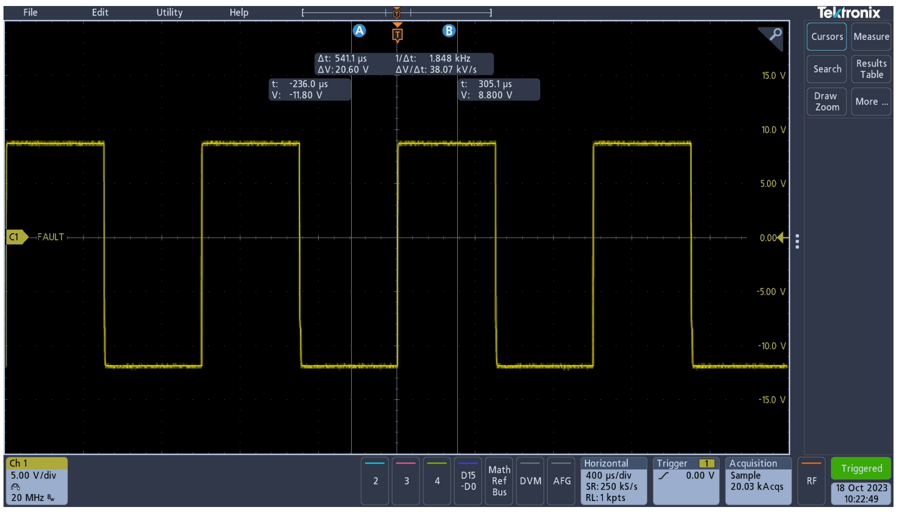


Figure 5.4: Channel 1 (yellow line) shows the PWM generated by EVSE in B2 state.

In figure 5.5 is shown the C state. The PWM has a maximum value equal to 6 V, a minimum value equal to -12 V and a period of 1 kHz.In this state vehicle and EVSE are connected and are ready to exchange power. In figure 5.6 is shown the

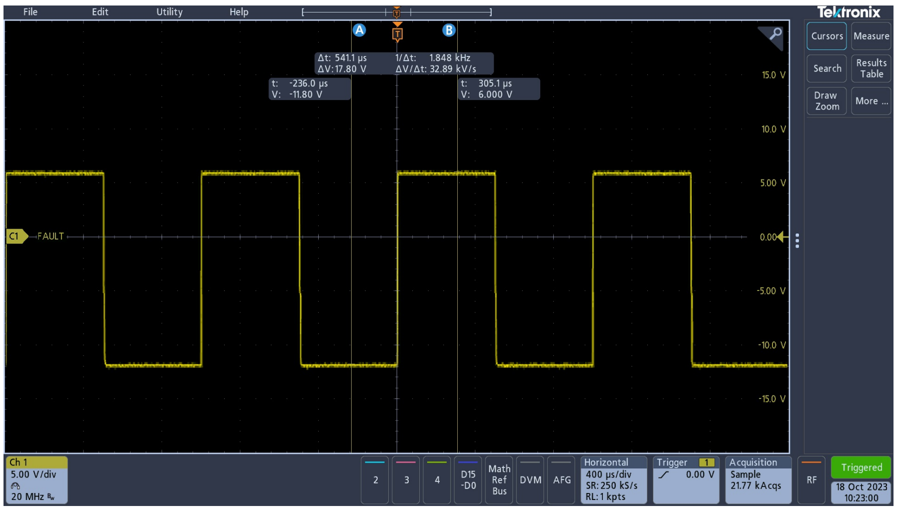


Figure 5.5: Channel 1 (yellow line) shows the PWM generated by EVSE in C state.

D state. The PWM has a maximum value equal to 3 V, a minimum value equal to -12 V and a period of 1 kHz.This state is equal to C state but the vehicle requires indoor ventilation .

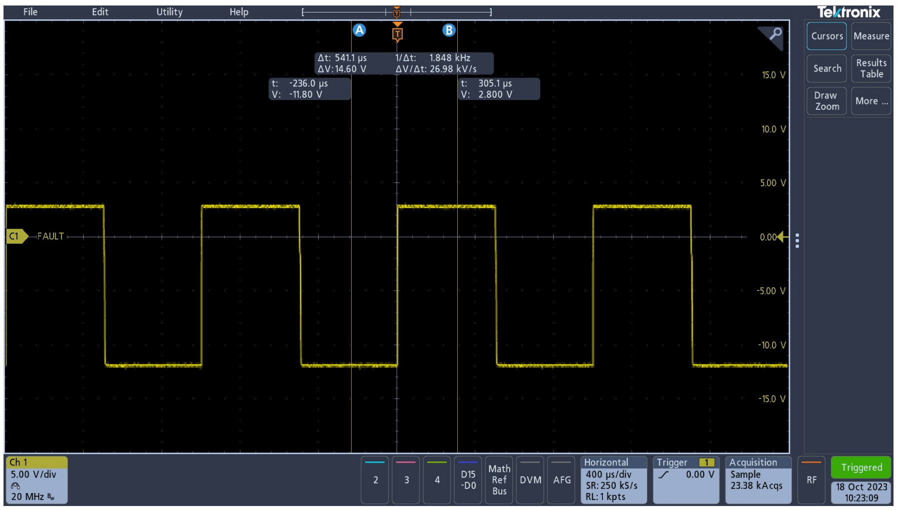


Figure 5.6: Channel 1 (yellow line) shows the PWM generated by EVSE in D state.

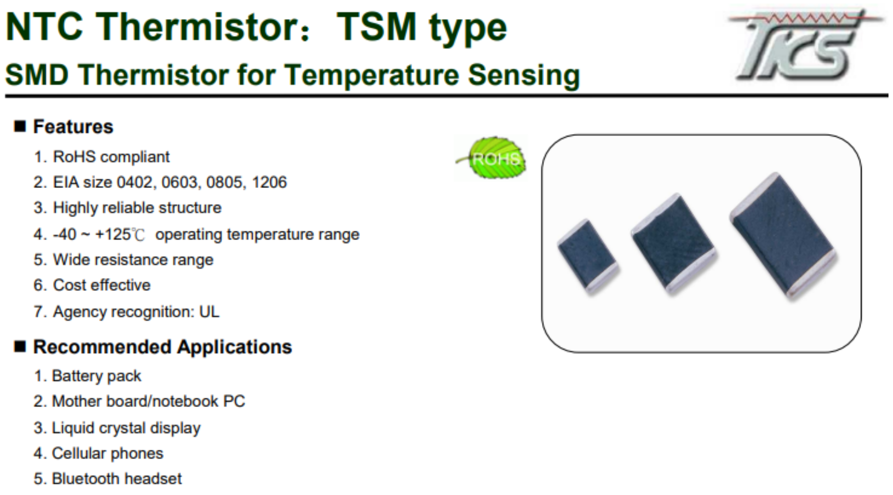
Chapter 6

Temperature sensor module

The continuous monitoring of the temperature has safety means in the project. The temperature is measured through a sensor mounted on the wallbox's power board.
The measure of interest is the internal temperature of the device: a high temperature can be dangerous and ruin the hardware components on the board, affecting the working efficiency.

6.1 Hardware description

The temperature sensor used for the project is an NTC from TKS (figure 6.1).



NTC Thermistor: TSM type
SMD Thermistor for Temperature Sensing

■ **Features**
1. RoHS compliant
2. EIA size 0402, 0603, 0805, 1206
3. Highly reliable structure
4. -40 ~ +125℃ operating temperature range
5. Wide resistance range
6. Cost effective
7. Agency recognition: UL

■ **Recommended Applications**
1. Battery pack
2. Mother board/notebook PC
3. Liquid crystal display
4. Cellular phones
5. Bluetooth headset

Figure 6.1: NTC datasheet [7]

An NTC, or "Negative Temperature Coefficient," is a type of thermistor, which is a temperature-sensitive electronic device, whose resistance value decreases with the temperature's increasing .
The dependence between resistance and temperature has a nonlinear trend, as shown in figure 6.2, dwhere on the x-axis are the temperature values in $[°C]$ and on the y-axis are the resistance values in $[KOhm]$.
The curve to which reference should be made is that of TSM0A103.

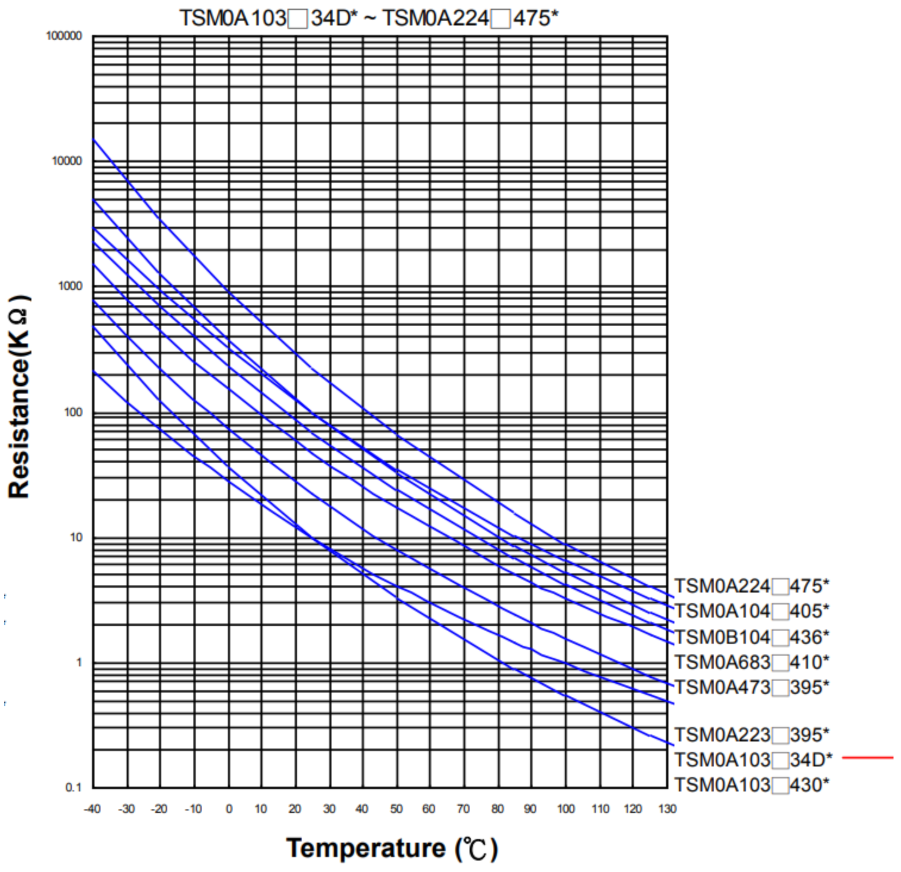


Figure 6.2: NTC Characteristic Curve [7]

In figure 6.3 is shown circuit for the acquisition of the temperature from the analog sensor NTC.
Temperature sensor is built using "Wheatstone Bridge" circuit. Since sensor output dynamics does not match the microcontroller input dynamics, it is necessary to introduce a circuit that matches those values of voltage The microcontroller acquire the analog signal in the point called *"Temp_RL1"*.

In this point the voltage value is the output of operational IC6, which is the difference between the inputs in positive (pin 3) pin and negative pin (pin 4) amplified by 20%.
The analog value at pin 4 is constant while at pin 3 the voltage depends by the NTC's resistance value which, as described above, changes in accordance with temperature.

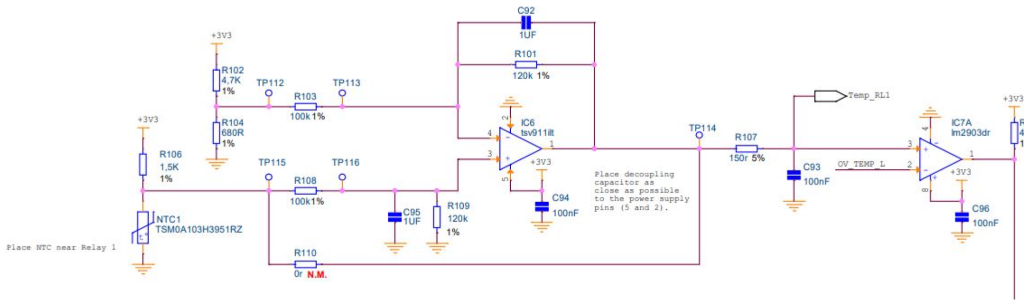
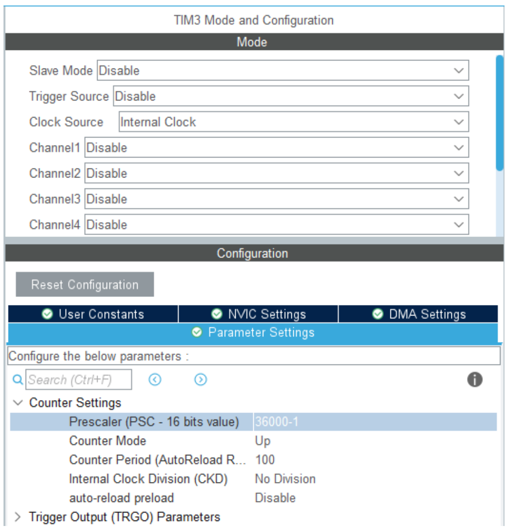


Figure 6.3: NTC Hardware scheme

6.2 Peripheral configuration

For the management of this module I used:

- Timer: In interrupt mode, I used this timer as a counter. Given an internal clock frequency of 36 MHz, I inserted a prescaler equal to (36000-1) to have a counter frequency of 1KHz. With an ARR equal to 100, I obtained a timer that triggers an interrupt every 100 ms.


TIM3 Mode and Configuration

Mode

Slave Mode Disable
Trigger Source Disable
Clock Source Internal Clock
Channel1 Disable
Channel2 Disable
Channel3 Disable
Channel4 Disable

Configuration

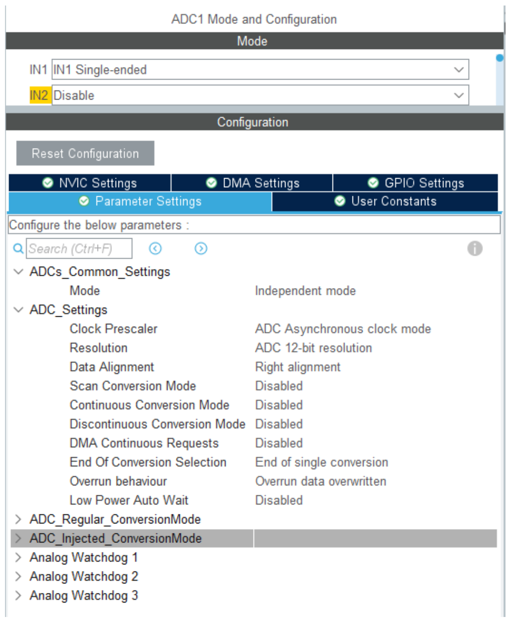
Counter Settings

Prescaler (PSC - 16 bits value)	36000-1
Counter Mode	Up
Counter Period (AutoReload R...	100
Internal Clock Division (CKD)	No Division
auto-reload preload	Disable

Trigger Output (TRGO) Parameters

Figure 6.4: Temperature timer settings

- ADC: In interrupt mode, I used the ADC converter with a resolution of 12 bits. Every time the ADC acquires a value an interrupt is triggered, so I can be sure that the converter has acquired new data.



ADC1 Mode and Configuration

Mode

IN1 IN1 Single-ended

IN2 Disable

Configuration

ADCs_Common_Settings	
Mode	Independent mode
ADC_Settings	
Clock Prescaler	ADC Asynchronous clock mode
Resolution	ADC 12-bit resolution
Data Alignment	Right alignment
Scan Conversion Mode	Disabled
Continuous Conversion Mode	Disabled
Discontinuous Conversion Mode	Disabled
DMA Continuous Requests	Disabled
End Of Conversion Selection	End of single conversion
Overrun behaviour	Overrun data overwritten
Low Power Auto Wait	Disabled
ADC_Regular_ConversionMode	
ADC_Injected_ConversionMode	
Analog Watchdog 1	
Analog Watchdog 2	
Analog Watchdog 3	

Figure 6.5: Temperature ADC settings

6.3 Module description

The purpose of the module is to handle the bit acquisition of the ADC in order to get back the temperature through the conversion formulas.

6.3.1 Piecewise linearization

The first thing to do was to convert the characteristic of the NTC in figure 6.2 to that of figure 6.6, where on the vertical axis there is the temperature while on the horizontal axis there is the corresponding value in bits.
Next, I proceeded with stroke linearization of the curve: I divided the curve into 5 zones in which each of these is characterized by a linear equation that has temperature as the unknown variable.
With this technique, depending on the bit value read and the zone to which it belongs, I have 5 different conversion formulas.

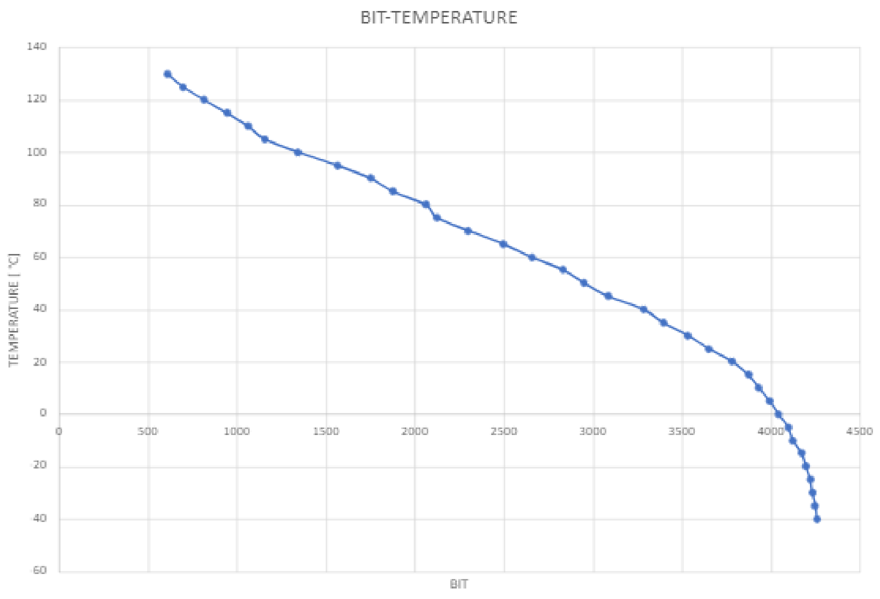


Figure 6.6: Bit-Temperature Characteristic

6.3.2 Acquisition handler

For signal acquisition, I used a timer and an ADC in interrupt mode.
When the timer reaches 100 ms an interrupt is triggered that starts the acquisition in the ADC port; once the data is received the converter interrupt is triggered. A callback is then called that saves the data and stops the acquisition: this way I have a periodic acquisition every 100 ms.
The conversion of bits to temperature is handled with a state machine: each state corresponds to a linearization zone, characterized by a specific conversion formula. The transition from one zone to the other is handled with hysteresis to avoid having an oscillation between states in case the acquired value fluctuates around the threshold.
The thresholds of the various zones are changed by a constant tolerance value, so I have lower values for low limits and higher values for high limits.
The received bit is converted to temperature depending on the linearization region it belongs.
The developed code part is shown below.

```c
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if ( htim->Instance == TIM3 )
    {
        __HAL_TIM_SET_COUNTER(&htim3, START_COUNTER_VALUE);
        HAL_ADC_Start_IT(&hadc1);
    }
}

void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc)
{
    if (hadc->Instance == ADC1)
    {
        BitValue = HAL_ADC_GetValue(&hadc1);
        ConversionStatus = RUN;
    }

}

void TPT_State_Machine()
{
    switch(BitZone)
    {
        case ZONE1:
             if ( BitValue >= (BOUND_HIGH_BIT_1 + BIT_TOLLERANCE))
            {
                BitZone = ZONE2;
            }
            else
            {
                TPTValue = ( M_GRADE_1 * BitValue) + OFFSET_1;
            }
            break;

        case ZONE2:
            if  ( BitValue >= (BOUND_HIGH_BIT_2 + BIT_TOLLERANCE)  )
            {
                BitZone = ZONE3;
            }
            else if ( BitValue < (BOUND_LOW_BIT_2 - BIT_TOLLERANCE)  )
            {
                BitZone = ZONE1;
            }
            else
            {
                TPTValue = ( M_GRADE_2 * BitValue) + OFFSET_2;
            }
            break;

        case ZONE3:
            if ( BitValue >= (BOUND_HIGH_BIT_3 + BIT_TOLLERANCE) )
            {
                BitZone = ZONE4;
            }
            else if ( BitValue < (BOUND_LOW_BIT_3 - BIT_TOLLERANCE) )
            {
                BitZone = ZONE2;
            }
            else
            {
                TPTValue = ( M_GRADE_3 * BitValue) + OFFSET_3;
            }
            break;

        case ZONE4:
            if ( BitValue > (BOUND_HIGH_BIT_4 + BIT_TOLLERANCE) )
            {
                BitZone = ZONE5;
            }
            else if ( BitValue < (BOUND_LOW_BIT_4 - BIT_TOLLERANCE) )
            {
                BitZone = ZONE3;
            }
            else
            {
                TPTValue = ( M_GRADE_4 * BitValue) + OFFSET_4;
            }
            break;

        case ZONE5:
            if ( BitValue < (BOUND_LOW_BIT_5 - BIT_TOLLERANCE) )
            {
                BitZone = ZONE4;
            }
            else
            {
                TPTValue = ( M_GRADE_5 * BitValue) + OFFSET_5;
            }

            break;
    }

}
```

Next, a "Moving Average Filter" is applied in order to reduce any noise. The filter is a 10 point filter and the result is a temperature value every second, which is enough time to measure a quantity like temperature that has a very slow dynamics of change.

6.4 Functionality test

To verify the correct operation of the module, it was sufficient to make a comparison between the temperature obtained from the measurement and the ambient temperature.
In figure 6.7 is shown the oscilloscope measurement of the input voltage value to the ADC. The measured voltage value is 2.84 V, corresponding to 3589 bits. This bit value corresponds to a temperature of 27.2 °C (as shown in the characteristic in figure 6.6), which corresponds with the ambient temperature of the test.

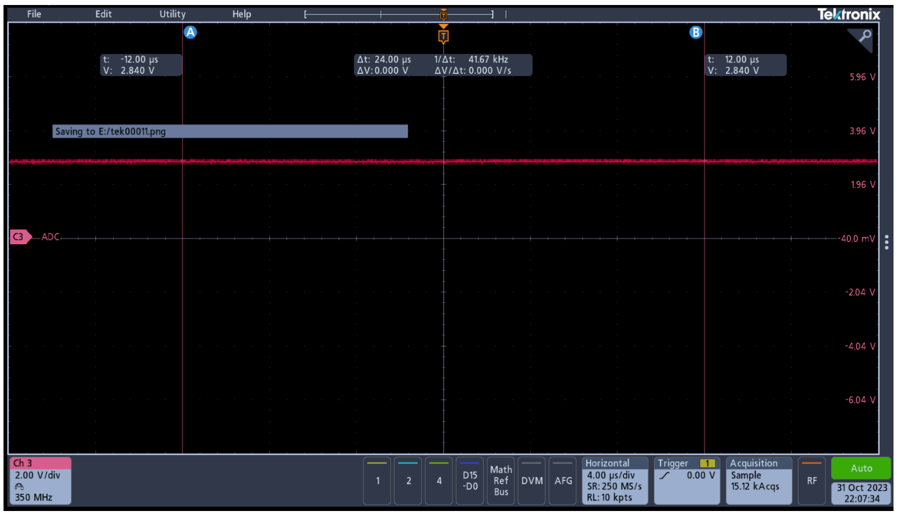


Figure 6.7: Channel 3 (red line) shows the signal level at point *"Temp_RL1"*, previously shown in figure 6.3

Next, I approached a heat source to NTC to check the correct operation of the module after a change of condition. The temperature values obtained were higher and higher in the time, until they stopped around the temperature value of the heat source.
Once the source that triggered the temperature rise was removed, the NTC returned to measure the ambient temperature value in acceptable times.

Chapter 7

Residual Current Monitoring (RCM) module

Residual current monitoring (RCM) is a device used for safety reasons in electrical circuits and has the role of continuously monitoring the presence of current dispersion towards earth.
The RCM is a toroidal ferrite magnetic core that performs the function of measuring the difference between the input and output current. When the current dispersion is higher than a certain threshold, the device instantaneously stops the operation of the wallbox by opening the relays.
The wires pass through the magnetic core and what happens is that any changes in current generate a magnetic field that goes to alter the current flowing inside the device.
Fault can be of two types:

- AC Fault: there is a difference between the current directed to the load (L_1) and current that return from the load (L_2) higher than 20 mA, it means that there was a leakage to ground;

- DC Fault: occurs when one of the phases loses insulation to ground and there is a current dispersion higher than 6 mA.

The standard "UL 2594" requires that the safety devices (RCM and Relays) must have a function that tests if they are working properly. For this reason, the RCM has a TEST pin that gives the possibility to implement the standard required test function. When the test is done, the RCM must open the relay to have a good execution of safety operations.
The device chosen for the project is the Kemet FG-R01-4A (shown in figure 7.1). The device is compliant with UL standards and is a type A, which means that is

fault-sensitive in both DC and AC case.

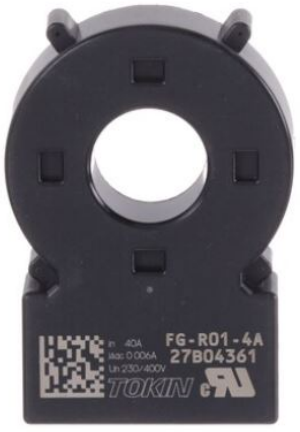


Figure 7.1: RCM mounted on Wallbox Mini

7.1 Hardware description

The device has 14 pins as shown in figure 7.2, below is listed the pin description :

- PIN 1: VDD, used to provide the 5 V power supply to the system;
- PIN 2: GND, used to connect the device to ground;
- PIN 3: AOUT, analog output pin, is not used;
- PIN 4: DC Fault, highlights any DC leakage current in the circuit by going from 0 to 1;
- PIN 5: AC Fault, highlights any AC leakage current in the circuit by going from 0 to 1;
- PIN 6: TEST, when it receives a pulse, the device starts operation of general test;
- PIN 7-10: Output Power Wires, pin 7 and pin 8 shorted respectively with pins 10 and 9, to create two unique lines;
- PIN 11-14: Input Power Wires, pin 11 and pin 12 shorted respectively with pins 14 and 13 , to create two unique lines;

From these pins in fact come out the lines called L1_Relay and L2_Relay that control their opening in case of fault.
Also in figure 7.2 can be appreciated the hardware connection of the other pins: it can be seen that the AOUT pin is actually disconnected while the signals uC_RCM_AC_FAULT and the uC_RCM_DC_FAULT represent the output of the RCM and will be the signals that determine the relays state.

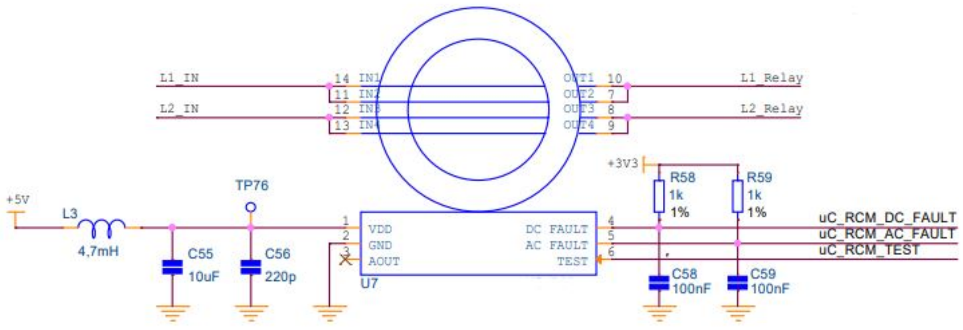


Figure 7.2: RCM hardware scheme

7.2 Peripheral configuration

For the management of this module I use:

- Timers: the timers used all stay on APB1 and receive an internal clock frequency of 36 MHz

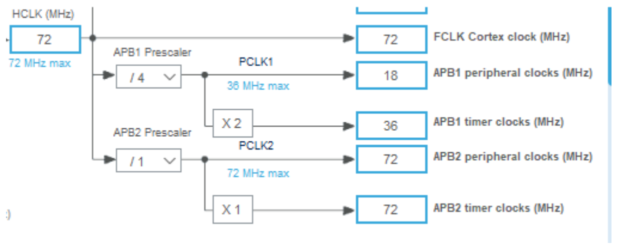
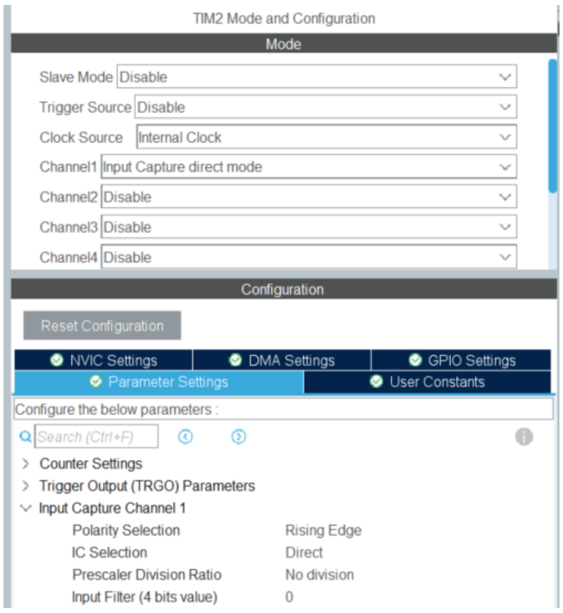

HCLK (MHz) 72
72 MHz max
APB1 Prescaler /4
PCLK1
36 MHz max
X 2
APB2 Prescaler /1
PCLK2
72 MHz max
X 1
72 FCLK Cortex clock (MHz)
18 APB1 peripheral clocks (MHz)
36 APB1 timer clocks (MHz)
72 APB2 peripheral clocks (MHz)
72 APB2 timer clocks (MHz)

Figure 7.3: A screen of clock configuration settings. The timers used are in APB1 timer clocks cell.

– TIM 1: with a default clock frequency of 36MHz and in interrupt mode, I used channel 1 in Input Capture mode: that is, whenever the pin detects an input signal, it triggers the interrupt that causes a routine to run. The settings are shown in figure 7.4;


TIM2 Mode and Configuration

Mode

Slave Mode: Disable
Trigger Source: Disable
Clock Source: Internal Clock
Channel1: Input Capture direct mode
Channel2: Disable
Channel3: Disable
Channel4: Disable

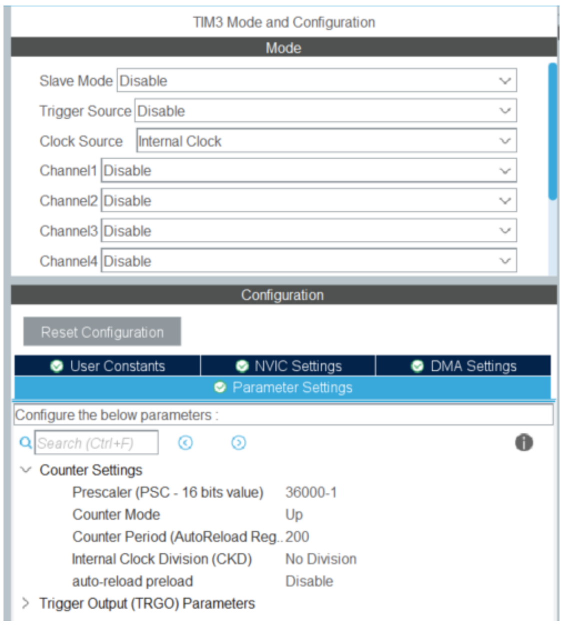
Configuration

Configure the below parameters :

Input Capture Channel 1
Polarity Selection: Rising Edge
IC Selection: Direct
Prescaler Division Ratio: No division
Input Filter (4 bits value): 0

Figure 7.4: Timer 1 configuration settings

– TIM 2: In interrupt mode, I used this timer as a counter. I set the prescaler equal to (36000-1) to have a clock frequency of 1 KHz, I then entered AutoReload Register(ARR) value equal to 200 to have a counter that every 200 ms and would trigger a routine. The settings are shown in figure 7.5;



Slave Mode: Disable
Trigger Source: Disable
Clock Source: Internal Clock
Channel1: Disable
Channel2: Disable
Channel3: Disable
Channel4: Disable

Configure the below parameters :

Counter Settings

Prescaler (PSC - 16 bits value)	36000-1
Counter Mode	Up
Counter Period (AutoReload Reg..	200
Internal Clock Division (CKD)	No Division
auto-reload preload	Disable

Trigger Output (TRGO) Parameters

Figure 7.5: Timer 2 configuration settings

- GPIO
  - PA5: used as output GPIO to generetate a pulse.

7.3 Module description

The module has a dual purpose:

- to continuously monitor the fault pins of the device (AC and DC) in order to report any malfunctions;
- to carry out the function test of the RCM.

The Fault pins are connected to a GPIO which is configured as a timer in Input Capture mode. Whenever the pin detects a rising edge, a variable that follows the trend of the signal is set to 1 and the polarity of the IC is changed. In this way I can handle short-duration faults that do not result in a system crash. At the end, a callback that starts the counting timer 2 is called: if the fault state lasts longer than 200 ms the fault state is signaled to the system.

```
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
    if((htim == &htim2) && (htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1
  ))
    {
        if(EdgePolarity == RISING)
        {
            HAL_TIM_Base_Start_IT(&htim3);
            EdgePolarity = FALLING;
            RCM.GeneralFault = true;
            TIM_RESET_CAPTUREPOLARITY(&htim2, TIM_CHANNEL_1);
            TIM_SET_CAPTUREPOLARITY(&htim2, TIM_CHANNEL_1,
  TIM_INPUTCHANNELPOLARITY_FALLING);
        }

        else
        {
            HAL_TIM_Base_Stop_IT(&htim3);
            __HAL_TIM_SET_COUNTER(&htim3, START_COUNTER_VALUE);

                EdgePolarity = RISING;
            RCM.GeneralFault = false;
            TIM_RESET_CAPTUREPOLARITY(&htim2, TIM_CHANNEL_1);
            TIM_SET_CAPTUREPOLARITY(&htim2, TIM_CHANNEL_1,
  TIM_INPUTCHANNELPOLARITY_RISING);
        }
    }
}
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if( htim->Instance == TIM3 )
    {
        RCM.AllarmFault = true;
    }
}
```

To carry out the test, a pulse must be sent to the TEST pin of a duration of at least 120 us.The pulse is sent with GPIO.
Following the datasheet of the component, shown in figure 7.6, it should happen that:

- DC Fault pin is activated at least 0.12 s after the falling edge of the 'pulse and must stay high for a time ranging from 0.2 s to 1.18 s;
- AC Fault pin is activated at least 0.70 s after the falling edge of the 'pulse and must stay high for a time ranging from 0.2 s to 1.40 s.

The test execution status is monitored continuously and if something goes wrong another module handles the fault.

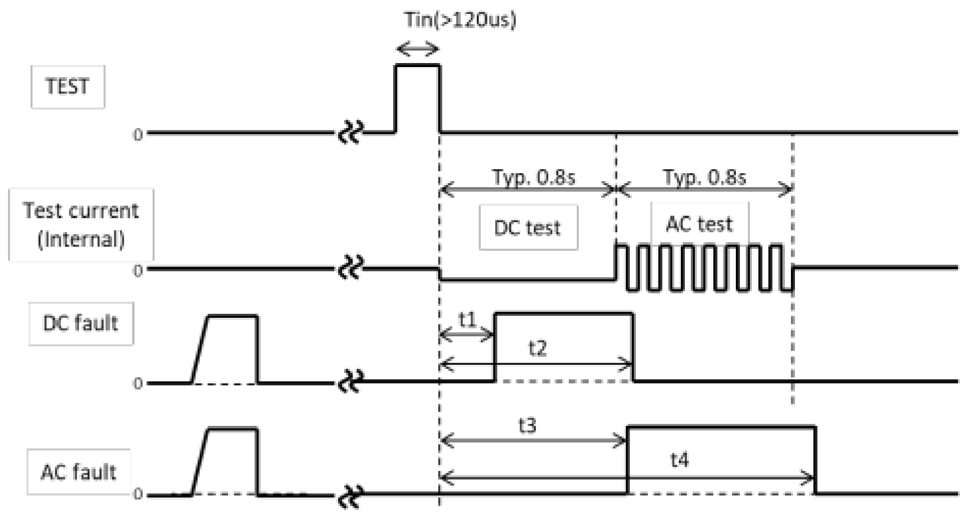


Figure 7.6: In order are shown: impulse test; internal current generated after impulse test to simulate AC and DC faults; DC fault signal behavior; AC fault signal behavior [8]

PARAMETER	MINIMUM	MAXIMUM
t1	0.12	0.60
t2	0.80	1.30
t3	0.70	1.20
t2	1.40	2.10

Table 7.1: Maximum and minimum parameters test times referred to figure 7.6

7.4 Functionality tests

7.4.1 DC fault test

To simulate the fault of the system, it is sufficient to pass direct current inside the "hole" of the RCM.
This will create an induced magnetic field on the device coils inducing a difference between input and output current.
As shown in figure 7.7, the signal throughout the duration of the fault remains high. In the case on the left, where the fault is less than 2 s, no alarm flag is generated. On the contrary in the right side, a fault lasting more than 2 s generates an alarm.

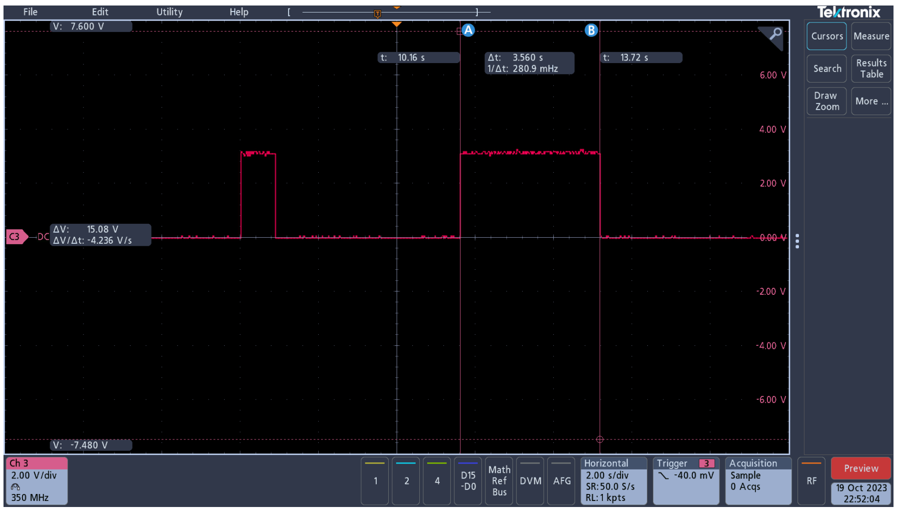


Figure 7.7: Channel 3 (Red line) is the DC Fault signal. On the left side, the fault time is lower than 2 s, while on the right the time is measured and is equal to 3.56 s

7.4.2 AC fault test

To simulate an AC fault, I altered one of the two phases by connecting a load between the chosen phase and ground to create a dispersion towards earth.
In the figure 7.8, the blue signal corresponds to the output of the AC Fault pin, which remains high throughout the duration of the fault.

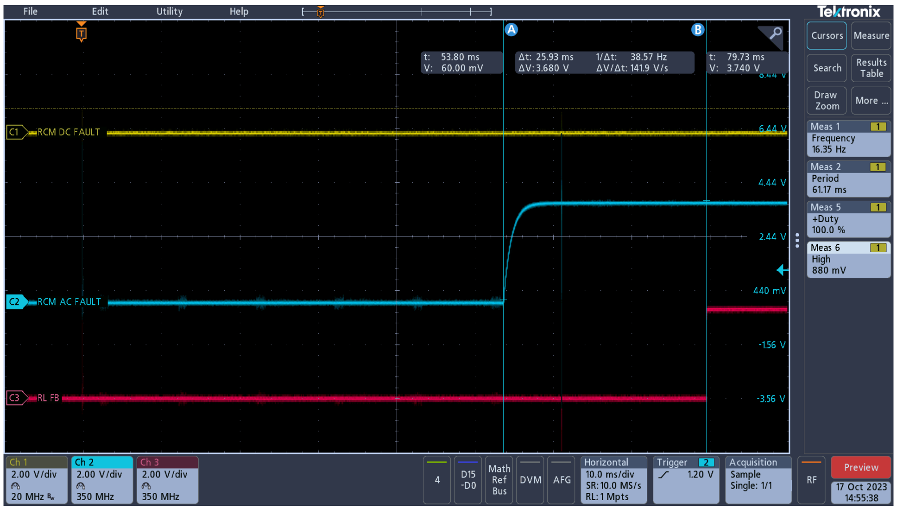


Figure 7.8: Channel 1 (yellow line) shows the DC Fault signal not considered in this test; Channel 2 (blue line) shows the AC Fault signal; Channel 3 (red line) shows the relay feedback not considered in this test

7.4.3 Self test

To test the self-test conditions, it was sufficient to send a 5 ms pulse to the TEST pin of the RCM with a GPIO.
Figure 7.9 shows the output signals from the AC Fault and DC Fault pins, with the measurement on the DC Fault of 700 ms highlighted, a parameter that stands within the among the values shown in table 7.1.
Instead, figure 7.10 shows the measurement with the oscilloscope of the pulse time.

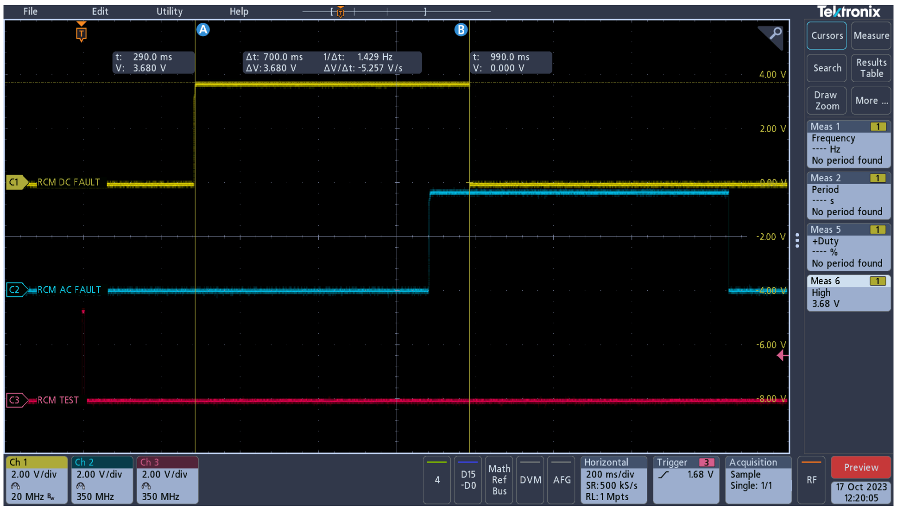


Figure 7.9: Channel 1 (yellow line) shows the DC Fault signal; Channel 2 (blue line) shows the AC Fault signal; Channel 3 (red line) shows the input test signal

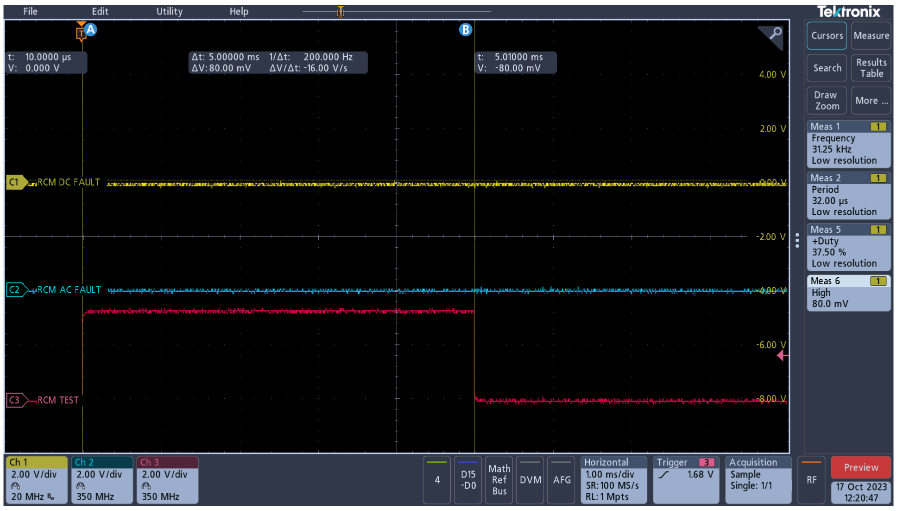


Figure 7.10: Channel 3 (red line) shows the time measurement of the test pulse

Chapter 8

Relay handler module

The relay is a device that performs the safety function within the circuit.
On the wallbox are mounted two relays, and with RCM are the protection devices that block the exchange of power between vehicle and EVSE in case of fault.
Specifically, it consists of a switch that is driven by an external signal to open or close it.
The relay has an electromagnetic winding and a contact: the drive signal determines a magnetic field that attracts or repels the contacts.
When the relays open, the operation of the wallbox is instantaneously blocked and the current is no longer delivered; conversely, when the wallbox is in normal operation, the relays are closed.
Two relays are mounted on the wallbox, and the model is shown in figure 8.1.

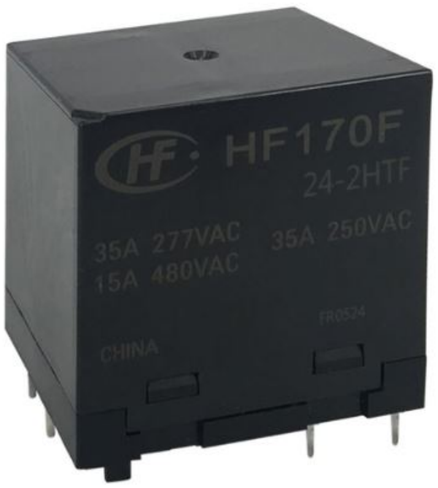


Figure 8.1: Relay mounted on Wallbox Mini

8.1 Hardware description

For bitron privacy reasons, it was not possible to include the entire hardware implementation that controls the relays, but only an extract shown in figure 8.2.

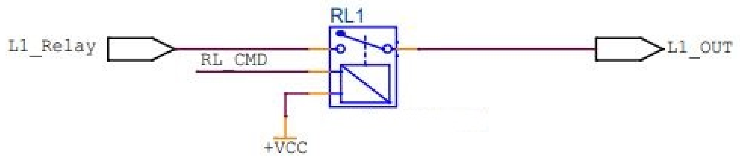


Figure 8.2: Relay with input and output signals

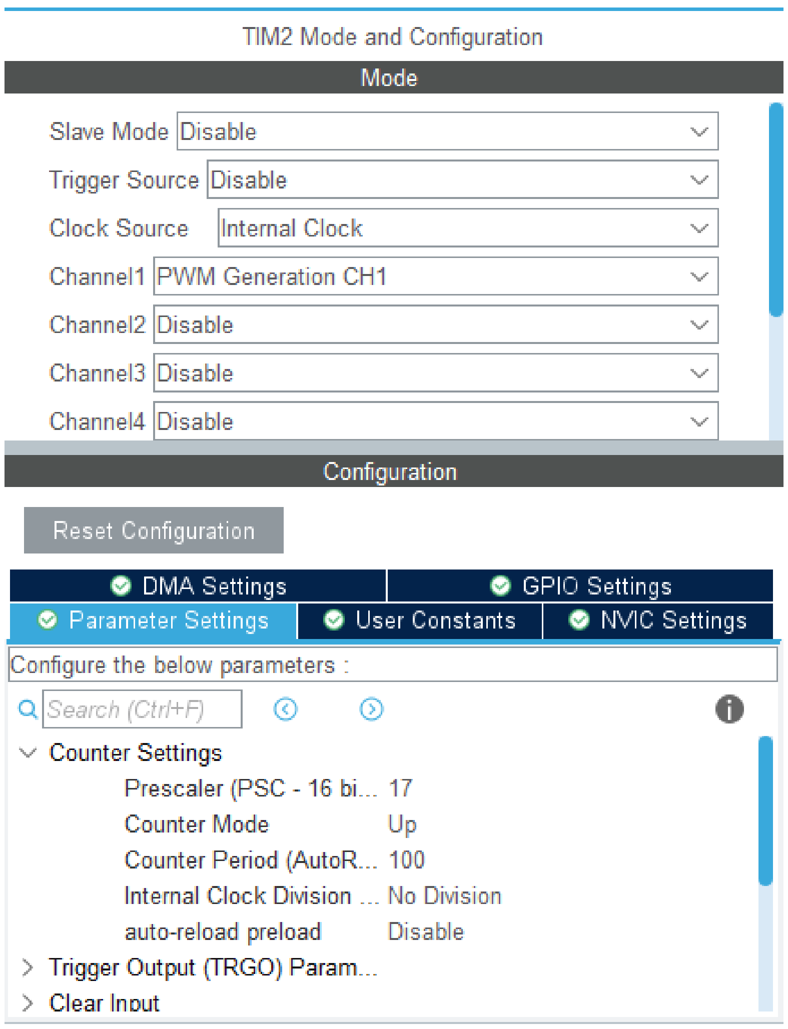
What is relevant to know is that each relay has three input:

- one of the wallbox's power lines;
- the command that determines the relay status;
- the 12 V power supply.

The output of the device is the power line, which will be 0 if the relay is open.

8.2 Peripheral configuration

To manage this module, I used a timer with a clock frequency of 36 MHz.
I configured one channel in PWM generation, inserted a prescaler of (18-1) and an ARR of 100 to get an output PWM with a frequency of 20 KHz.
The settings are shown in figure 8.3 .


TIM2 Mode and Configuration

Mode

Slave Mode: Disable
Trigger Source: Disable
Clock Source: Internal Clock
Channel1: PWM Generation CH1
Channel2: Disable
Channel3: Disable
Channel4: Disable

Configuration

Counter Settings

Prescaler (PSC - 16 bi...	17
Counter Mode	Up
Counter Period (AutoR...	100
Internal Clock Division ...	No Division
auto-reload preload	Disable

Trigger Output (TRGO) Param...
Clear Input

Figure 8.3: Timer settings to generate a PWM signal

8.3 Module description

The opening or closing of relays is handled by a module in the application layer that calls two functions of the BSP layer, defined as:

- *open_relay()* to control the opening of relays;
- *close_relay()* to control the closing of relays.

When the wallbox is started the relays are closed and maintain this state during normal operation of the device. However, when the RCM detects a fault, the function *openrelay()* opens the relays, blocking the wallbox. The closing command is a PWM with variable duty cycle generated with the timer and is handled with the peak and hold technique:

- the maximum current sufficient to close the contacts is supplied for a certain time, specifically for 100 ms. At this time, the duty cycle of the PWM is 100%;
- once the relays are closed, the duty cycle is lowered to 60%, which keeps the device contacts closed.

The peak and hold technique ensures the proper functioning of the device with significant energy savings. In addition, since there is less current that flows in the devices, the phenomenon of self-heating is reduced.
The peak and hold trend is shown in figure 8.4 while the period measurement is shown in detail in figure 8.5.

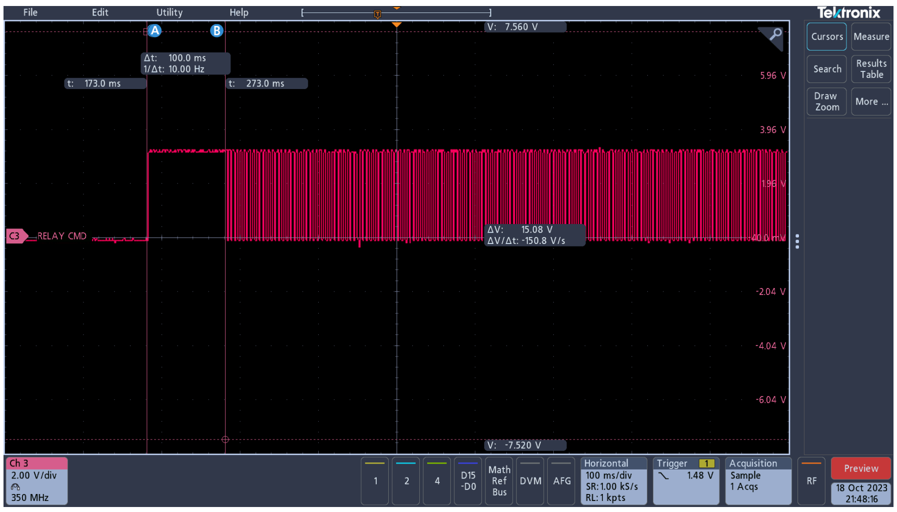


Figure 8.4: Channel 3 shows the peak and hold signal

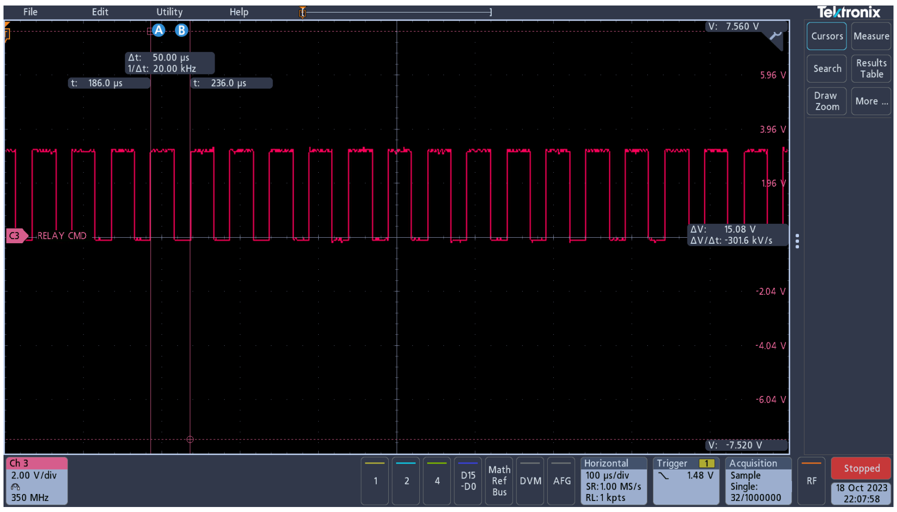


Figure 8.5: Channel 3 shows the period measurement of the PWM

Below is shown the code for the function that controls the closing of the relays.

```
void close_relay(void)
{
    switch (RelayCloseStatus)
      {
      case INIT:

        __HAL_TIM_SET_AUTORELOAD(&htim2, AUTORELOAD);
        __HAL_TIM_SET_COUNTER(&htim2, START_COUNTER_VALUE);
          HAL_TIM_PWM_Start(&htim2, TIM_CHANNEL_1);
            TimerSw_Set(&MyTimer, PEAK_TIME);
            RelayCloseStatus = PEAK;
          break;

      case PEAK:

          if ( TimerSw_IsExpired(&MyTimer) == true  )
          {
              RelayCloseStatus = HOLD;
          }
          break;

      case HOLD:

          __HAL_TIM_SET_AUTORELOAD(&htim2, PWM_MODULATION);
          RelayCloseStatus = IDLE;
          IsRelayClosed = true;
```

```
            break;

        case IDLE:

            break;
        }
}
```

8.4 Functionality test

The relay function test is closely related to the test performed on the RCM explained on 7.4, keeping the same setup and using the output signal from the RCM to drive the relay.
I measured the signal RL_FB, which represents a voltage level according to the state of the relay: high when the relays are closed, and low when they are open. In figure 8.6, the closing of the relays by the signal transition is shown. Furthermore, the time between the instant when the RCM perceives the fault and when the voltage level on the relay changes is measured.

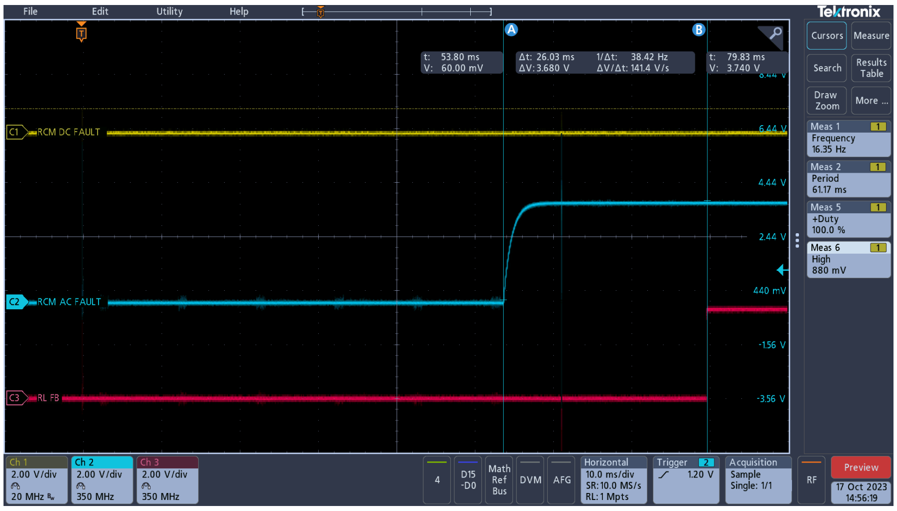


Figure 8.6: Channel 1 (yellow line) shows the DC Fault signal; Channel 2 (blue line) shows the AC Fault signal; Channel 3 (red line) shows the relay feedback signal. The measurement shows the time difference between start of failure and relay feedback signal equal to 26.03 ms

The measured time, about 26 ms, was too high. The reason is that when the feedback signal of the relays is measured, the mechanical closing time of the relays

must be taken into account.
To get an accurate measure of the relay reaction time, it is more correct to consider the command pin
As shown in figure 8.7, the actual closing time of the safety device is very small, around 680 us.

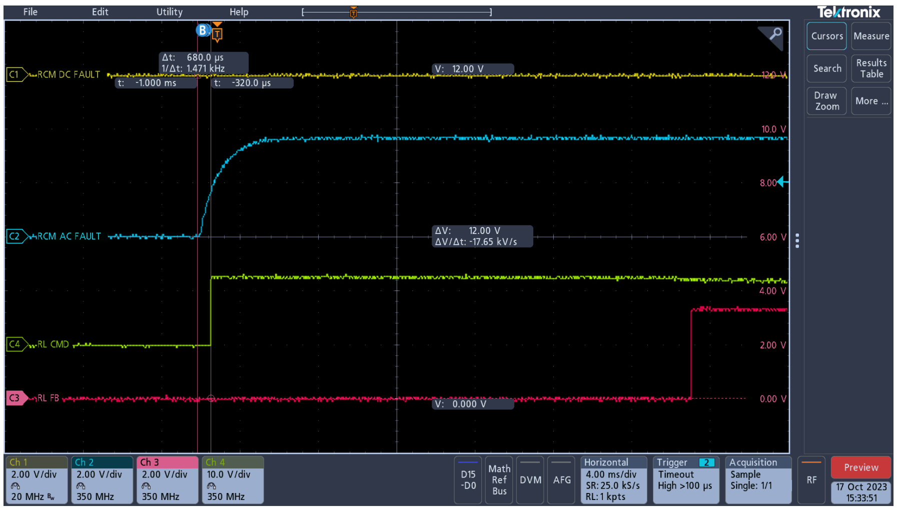


Figure 8.7: Channel 1 (yellow line) shows the DC Fault signal; Channel 2 (blue line) shows the AC Fault signal; Channel 3 (red line) shows the relay feedback signal; Channel 4 (green line) shows the relay command signal. The measurement shows the time difference between start of failure and relay reaction time

Chapter 9

Conclusions

My thesis work has given me the opportunity to study and increase my knowledge about vehicle's AC charging world.
In the first stage, has been important understand the charging levels and how they work. Later, the most important aspects has been analyzed, like the types of connectors and the coupler contact interface between EVSE and vehicle.
The thesis work has follow some important phases before starting the firmware developing.
A crucial point for the work has been the drafting of firmware architecture and once all the resources have been clear the work to be done, is started the phase of code developing.
As may become clear, a project like the one treated in this thesis which concerns mechanical, hardware and software development, requires a suitable time to be completed and is subject to continuous changes.
About possible future developments, there are two macrothemes to consider:

- for the functionality purpose, it will be important testing hardware and firmware once both will be at the final development stage. In addition, will be important to verify the compatibility of both when the wallbox is in steady-state work condition;
- for the certification purpose, they shall draw up the entire documentation requested by the certification body which will make then all the necessary tests on the device in order to evaluate the compliance with UL standards.

In conclusion, and for what concern about my contribute, the modules which I develop during my internship can be adapted for future company's project with a different hardware structure due to the architecture description explained in 3.2.1.

Bibliography

[1] *Wallbox Mini specifications*. Bitron (cit. on pp. 2, 9).

[2] *J1772- Surface Vehicle Standard*. SAE International (cit. on pp. 5, 7, 22–24).

[3] *Guida alla ricarica*. eStation (cit. on p. 6).

[4] *STM32MP151A/D*. STMicroelectronics (cit. on pp. 11, 12).

[5] *UL Standard for Safety for Electric Vehicle Supply Equipment, UL 2594*. Underwriters Laboratories (UL) (cit. on p. 15).

[6] *UL Standard for Safety for Software in Programmable Components, UL 1998*. Underwriters Laboratories (UL) (cit. on pp. 15, 16).

[7] *SMD Thermistor for Temperature Sensing*. TKS (cit. on pp. 32, 33).

[8] *Fluxgate-Based Residual Current Sensor*. KEMET (cit. on p. 47).