Motorvehicle University
of Emilia-Romagna

Advanced Automotive Electronic Engineering

Group Four

Project report
of
Automotive Electronics

F1 Smart Controller

Authors:
908759 Biguzzi Annachiara
889841 Di Loro Giorgio
890427 Mancini Luca
889711 Ravagli Giacomo
877990 Villar Tovar Armando Ruben
267843 Zhou Tong

Professor:
Pavan Paolo

Stakeholder:
Silenzi Claudio

Abstract

The paper follows the complete development of the smart controller commissioned by Ferrari, starting from project planning and group organization, going through the initial selection of the components, down to the actual realization of the final product. This document covers various topics related to electrical, electronic, software, mechanical and management engineering, thus it could be regarded as a useful reference manual for those that are required to start working on a new project from scratch for their first time.

Keywords: Ferrari, management, planning, electronics, automotive, design.

Contents

1 Project Overview **13**
- 1.1 Project Description . . . 13
- 1.2 Scope and Objectives . . . 13
- 1.3 Use case . . . 13
- 1.4 Project Approach . . . 14
- 1.5 Software tools . . . 15
- 1.6 Cost analysis . . . 16
- 1.7 Risk Analysis . . . 17

2 Project Management Plan **19**
- 2.0.1 Project Scheduling . . . 19
- 2.1 Work Breakdown Strategy . . . 19
  - 2.1.1 Macro-Planning . . . 20
  - 2.1.2 Reports . . . 21
- 2.2 Major Project Milestones . . . 22
- 2.3 Assignment of the Roles and Responsibilities . . . 22
- 2.4 Documentation Required . . . 24

3 Requirements Specification **25**

4 Analysis **27**
- 4.1 Hardware Goals . . . 27
- 4.2 Firmware Goals . . . 27
- 4.3 Concept . . . 27
  - 4.3.1 First Solution . . . 27
  - 4.3.2 Update to Wireless Solution . . . 27
  - 4.3.3 CL1 - Update to Final Modular Solution . . . 29
- 4.4 General Overview of the Decisional Process . . . 31
- 4.5 MCU . . . 31
  - 4.5.1 Evaluated MCU . . . 32
- 4.6 Sensors . . . 33
  - 4.6.1 Barometric Pressure Sensor . . . 33
  - 4.6.2 Accelerometer . . . 36
- 4.7 Evaluation Board . . . 38
- 4.8 Power Supply . . . 39
  - 4.8.1 Power Converter Choice . . . 40
  - 4.8.2 LiPo Battery 3.7V 150 mAh . . . 42
- 4.9 Wireless connection . . . 43
  - 4.9.1 Main Wireless Features . . . 43
  - 4.9.2 Proposed Solutions . . . 44
- 4.10 Wired Connection - Output . . . 45

4.11 Bill of Materials and Cost Evaluation . . . 47
4.11.1 Evaluation Board . . . 47
4.11.2 Custom Board . . . 48
4.11.3 Custom Board with Sensors on both sides . . . 48
4.11.4 Custom Board Reduced Size . . . 49
4.11.5 Cost Resume . . . 49
4.11.6 Conclusion . . . 50
4.12 Overall Expenses . . . 53

5 Hardware Design **54**
5.1 Design Objectives and Specification . . . 54
5.2 Evaluation Set Up . . . 54
5.3 Design Steps . . . 54
5.4 MCU . . . 55
5.5 Sensors . . . 60
5.6 CAN . . . 61
5.7 Power Converter . . . 63
5.8 Connectors . . . 66
5.9 SEPIC Converter Simulations . . . 67
5.9.1 Rload estimation . . . 70
5.9.2 Simulation Set-Up . . . 70
5.9.3 Constant Input Voltage V_IN = 12 V . . . 71
5.9.4 Battery Discharge - from V_IN = 3.7 V at t = 0 s to V_IN = 3 V 71
5.9.5 Constant Overvoltage Input V_IN = 14 V . . . 71
5.9.6 Turning off and on . . . 71
5.10 Polarity Inversion and Over-current Protection . . . 74
5.10.1 Overcurrent protection . . . 74
5.10.2 Simulation Set-Up . . . 76
5.10.3 Simulations . . . 76
5.11 PCB Features . . . 79
5.11.1 PCB Goals . . . 79
5.11.2 PCB Main Features . . . 79
5.12 PCB layout . . . 79
5.12.1 PCB 2D . . . 79
5.12.2 Eagle Settings . . . 79
5.12.3 PCB 3D . . . 82
5.12.4 Dimensions and Weight . . . 83
5.13 Estimated Timeline for Production . . . 83
5.14 Soldering . . . 84
5.15 Future Developments . . . 84
5.15.1 High Side Output Implementation . . . 84
5.15.2 Reduced Size Components Upgrade . . . 85
5.15.3 Reduce Size CL1 Board Design . . . 86

5.15.4 Connector 87

6 Firmware Design **89**
6.1 Firmware Goals 89
6.2 Selection of the IDE 89
6.3 Unified Code - Evaluation and Custom Board Characteristics 90
6.4 Main Characteristics 91
6.5 Evaluation Board Set-Up 92
6.5.1 Sampling Frequency Considerations 92
6.5.2 LIS3DH and BMP280 Configuration 93
6.5.3 NRF24L01+ Configuration 93
6.6 Code Overview - Sensor Board 93
6.7 Code Overview - Controller board 97
6.8 Code Overview - Wired Sensor 100
6.9 Future development 103

7 Software Design - User Interface **105**
7.1 Software Goals 105
7.2 User Interface Architecture 105
7.2.1 STM32F407G-DISC1 Evaluation Board 105
7.3 Firmware Development 106
7.3.1 CAN configuration 106
7.3.2 USART/USB configuration 107
7.3.3 Functional flow 107
7.4 Software Development 109
7.4.1 LabView Block Diagram 109
7.4.2 LabView Front Panel 112
7.5 User interface Future Development 113

8 Mechanical Design **114**
8.1 Goals 114
8.2 CL1 Box 114
8.2.1 Main Box 114
8.2.2 Cover 119
8.2.3 Assembly of the box 120
8.2.4 Box dimensions resume 121
8.3 3D Printing 121
8.3.1 3D Printing Advantages 121
8.4 Material Evaluation 122
8.4.1 PLA 122
8.4.2 ABS 123
8.4.3 PETg 123
8.5 Mechanical Future Developments 124

9 Hardware Validation and Test **125**
9.1 Validation Procedures Set-Up . . . 125
9.2 Validation Test Summary . . . 125
9.3 Electrical testing . . . 125
9.3.1 Test 1_ELECTRICAL . . . 125
9.3.2 Test 2_ELECTRICAL . . . 126
9.3.3 Test 3_ELECTRICAL . . . 126
9.4 Functional testing - Signals . . . 126
9.4.1 Test 1_SIGNAL . . . 126
9.4.2 Test 2_SIGNAL . . . 126
9.5 Functional testing - SEPIC . . . 127
9.5.1 Test 1_SEPIC . . . 127
9.5.2 Test 2_SEPIC . . . 127
9.5.3 Test 3_SEPIC . . . 128
9.5.4 Test 4_SEPIC . . . 129
9.6 Functional testing - MCU . . . 130
9.6.1 Test 1_MCU . . . 130
9.6.2 Test 2_MCU . . . 130
9.7 Functional testing - CAN Line . . . 130
9.7.1 Test 1_CAN . . . 131
9.7.2 Test 2_CAN . . . 131
9.8 Functional testing - Over-current and Polarity Inversion Protection . . . 131
9.9 Functional testing - Additional Tests . . . 132
9.9.1 Text 1_RES - Temperature Test . . . 132
9.9.2 Text 2_RES - Climatic Test . . . 132
9.9.3 Text 3_RES - Vibration Test . . . 133
9.9.4 Test 4_RES - Accelerated Lifecycle test . . . 134
9.9.5 Text 5_RES - EMC Test . . . 134
9.9.6 Text 6_RES – Conformance Test . . . 134
9.10 Custom Board Current Consumption . . . 135
9.10.1 RX Configuration Theoretical Consumption . . . 135
9.10.2 TX Configuration Theoretical Consumption . . . 135
9.10.3 Sleep Mode Configuration Theoretical Consumption . . . 136
9.10.4 Real Current Consumption . . . 136
9.11 Thermal Analysis . . . 137
9.12 Battery Life Estimation . . . 138
9.12.1 Battery Objectives . . . 138
9.12.2 Assumptions on Operation Mode and Battery Life . . . 138
9.13 FMEA . . . 139
9.14 Hardware Tests Performed Resume . . . 140

10 Firmware Validation **141**
10.1 Firmware Tests . . . 141
10.1.1 Test 1_CL1_FUNCTION . . . 141
10.1.2 Test 2_CL1_FUNCTION . . . 142
10.2 User Interface Tests . . . 143
10.2.1 Test 1_USER_FUNCTION . . . 143
10.2.2 Test 2_USER_FUNCTION . . . 144
10.3 Firmware performances . . . 145
10.3.1 Test1_PERFORMANCE . . . 145
10.3.2 Test2_PERFORMANCE . . . 146
10.4 Impact on CAN Bus load . . . 147
10.5 Firmware Tests Performed Resume . . . 149

11 Mechanical Test **150**
11.1 Static Force Analysis . . . 150
11.1.1 Force Simulation Goals . . . 150
11.1.2 Simulation Set-Up . . . 150
11.1.3 Force Application Points . . . 151
11.1.4 Simulation Results . . . 152
11.1.5 Conclusions . . . 160
11.2 Air Flow Simulations . . . 161
11.2.1 Simulation Set-Up . . . 162
11.2.2 Simulation Results . . . 164

12 Automotive Compliance Testing **166**

Appendix A: Version Index **167**

Appendix B: Board Schematic **168**

Appendix C: Source Code **176**
.I Evaluation Source Code - STM8AF5288T . . . 176
.II Board Source Code - CL1 . . . 187
.III CAN Sniffer Source Code - STM32F407G-DISC1 . . . 196

Appendix D: Datasheets **202**

Appendix E: Validation Documents **214**

Appendix F: Weekly Reports **219**

Appendix G: Additional Documents **230**
.IV Open Project Macro-activities Report . . . 230
.V Software Documentation . . . 236

.VI MCU - STM8AF5288T STM8 CubeMX Configuration 258
.VII MCU - STM32F407G_DISC1 STM32 CubeMX Configuration 264

I Evaluation Document **277**
I.I Encountered Problems 277
I.II Project Working Hours 277

II User Manual **279**
II.I CL1 Custom 279

Bibliography **282**

List of Figures

1 System Positioning . . . 14
2 V-Model . . . 14
3 Modified V-Model . . . 15
4 Work Breakdown Strategy . . . 19
5 Gantt chart . . . 20
6 Block diagram that shows the defined profiles . . . 23
7 Description of the first high level architecture of the system . . . 28
8 Description of the wireless solution architecture . . . 28
9 CL1 solution . . . 30
10 General Decisional Process . . . 31
11 STM8AF Board . . . 39
12 Nucleo208rb . . . 39
13 Adafruit BMP388 . . . 40
14 Adafruit LIS3DH . . . 40
15 LiPo battery cell 150 mAh . . . 43
16 Ethernet physical description . . . 47
17 CAN physical description . . . 47
18 Evaluation board BOM . . . 48
19 Custom board BOM . . . 49
20 Custom board BOM - considering sensors on both sides . . . 50
21 Custom board reduced size BOM . . . 51
22 Cost report . . . 53
23 Evaluation set-up . . . 54
24 MCU circuit . . . 55
25 Pin configuration through Cube MX . . . 56
26 MCU - Sensors connection . . . 56
27 Clock through Cube MX . . . 57
28 SWIM pinout . . . 57
29 Sensors circuit . . . 60
30 CAN circuit . . . 61
31 SEPIC converter . . . 63
32 Ripple on output voltage with PMEG6030EP . . . 68
33 Ripple on output current with PMEG6030EP . . . 68
34 Ripple on output voltage with RB550VM-30 . . . 69
35 Ripple on output current with RB550VM-30 . . . 69
36 Max current consumption RX mode . . . 70
37 LTSpice SEPIC converter circuit . . . 70
38 V_IN = 12 V . . . 71
39 Battery discharge - from V_IN = 3.7 V at t = 0 s to V_IN = 3 V . . . 72
40 Battery discharge - V_IN = 3.09 V the SEPIC converter is switched off . . . 72
41 Overvoltage condition - V_IN = 14 V . . . 73

42 Turn off - turn on . . . 73
43 Polarity inversion and over-current protection circuit . . . 74
44 PTC characteristics . . . 75
45 PTC characteristics . . . 75
46 LTSpice Set-up . . . 76
47 Behaviour of the protection in case of polarity inversion . . . 77
48 LTSpice Set-up . . . 78
49 PCB Final layout . . . 80
50 PCB Final layout . . . 81
51 SEPIC recommended layout . . . 81
52 SEPIC layout . . . 82
53 3D view . . . 83
54 High Side intended design . . . 86
55 Reduced size board design . . . 87
56 Nucleo-208RB . . . 90
57 Proposed solution architecture highlighting main peripherals used . . . 92
58 User interface architecture . . . 105
59 STM32F407G-DISC1 . . . 106
60 Opening of the USB port . . . 110
61 Data extrapolation from th received packet . . . 110
62 Closing of the USB port . . . 111
63 Front Panel for data visualization . . . 112
64 Bi-directional architecture block scheme . . . 113
65 Main box . . . 115
66 Top view of the main box . . . 115
67 Top view of the modified main box . . . 116
68 Side view of the main box . . . 116
69 Top view of the battery box . . . 117
70 Side view of the main box . . . 117
71 Side view of the main box . . . 118
72 Assembly of the box . . . 119
73 Assembly of the box . . . 120
74 Side view of the assembly . . . 120
75 Box dimensions . . . 121
76 Material mechanical properties . . . 122
77 Test procedure summary . . . 125
78 Load set-up . . . 127
79 Board connected with the 70 Ω . . . 128
80 Load Test . . . 129
81 CAN Test . . . 131
82 RX Current consumption . . . 135
83 TX Current consumption . . . 135
84 Qualitative results of thermal simulation . . . 137

85 CL1 system set-up . . . 141
86 Evaluation Board Sensor Transmitter . . . 143
87 CL1 Control Board Receiver . . . 144
88 Cluster simulator used to evaluate the CAN bus load . . . 148
89 Configuration of ABS material properties . . . 150
90 Configuration of PET material properties . . . 150
91 Constraints definition . . . 151
92 Force applied on the top . . . 151
93 Constraints definition . . . 152
94 Constraints definition . . . 152
95 Top stress . . . 153
96 Length direction stress . . . 153
97 Width direction stress . . . 154
98 Top stress . . . 154
99 Length direction stress . . . 155
100 Width direction stress . . . 155
101 Top stress . . . 156
102 Length direction stress . . . 156
103 Width direction stress . . . 157
104 Top stress . . . 158
105 Length direction stress . . . 158
106 Width direction stress . . . 159
107 Stress and deformation data comparison . . . 159
108 Side view of the wing . . . 161
109 Rear wing model without the box . . . 161
110 Rear wing model with the box . . . 162
111 Box on the wing zoomed . . . 162
112 Domain selection . . . 163
113 Cell example . . . 163
114 Cell set-up . . . 163
115 Velocity field without the box . . . 164
116 Velocity field with the box . . . 164
117 Pressure field without the box . . . 165
118 Pressure field with the box . . . 165
119 MCU and Connectors . . . 169
120 Power Supply . . . 170
121 Sensors . . . 171
122 MCU and Connectors . . . 173
123 Power Supply . . . 174
124 Sensors . . . 175
125 Group work hours count table . . . 278
126 Connection of the ST_LINK for STM8 . . . 279
127 Connection for programming . . . 280

128 CL1 front panel 281

List of Tables

1 Cost evaluation . . . 17
2 Description of the main risk factors . . . 17
3 Defined milestones and description of the main objectives. . . . 22
4 Description of main responsibilities of each profile . . . 23
5 MCUs features . . . 32
6 Barometric Pressure sensors overview . . . 33
7 Accelerometer overview . . . 36
8 AIS1120SX accelerometer . . . 37
9 AIS2120SX accelerometer . . . 38
10 LT8330 overview . . . 41
11 Wireless technologies . . . 44
12 Cost evaluation . . . 52
13 List of the used MCU pins . . . 58
14 CSTCE16M0V53 characteristics . . . 59
15 List of capacitors . . . 59
16 List of capacitors . . . 60
17 List of CAN components . . . 61
18 List of SEPIC components . . . 65
19 Diodes comparison . . . 67
20 PCB settings . . . 79
21 Main geometrical characteristics board . . . 83
22 Production timeline . . . 84
23 MCUs features . . . 85
24 Main geometrical characteristics board . . . 86
25 Tests resume . . . 140
26 Error codes transmitted on the CAN bus . . . 145
27 Cycle time evaluation . . . 146
28 Error rate in comparison . . . 146
29 Tests resume . . . 149
30 Documentation versions . . . 167
31 Main problems encountered . . . 277
32 Error codes transmitted on the CAN bus . . . 280

1 Project Overview

1.1 Project Description

The aim of the project is to realize a system that is able to acquire inputs from two different sensors and to display outputs. The main specifications are given by the customer, whereas the features related to the design of the system are left for the team to decide. This project is divided in two main parts:

- Project planning
- Technical design

Project management and all the topics related to planning are key aspects for the successful completion of the project.
In the following section the objectives of the project and the design approach adopted for the technical realization are described.

1.2 Scope and Objectives

The project aims to achieve the following goals:

- Realize a robust project planning, in order to have a faster development process;
- Identify the role that each member has inside the team;
- Define the architecture and make considerations on the selection of the components to be used;
- Realize a fully functional prototype using an evaluation board;
- Realize a custom board using properly chosen components that satisfy the requirements given by the customer;
- Validation of both the hardware and the firmware, focusing on the procedures used;
- Analysis of the main causes of failure.

1.3 Use case

The positioning of the entire system (controller board + sensor board) is illustrated in the following figure that has been provided by the company as a specification. The sensor board has to be mounted on the DRS wing in order to measure the air flow during tests. The controller can be mounted in the lateral side of the rear wing in order to be easily wired, avoiding problems related to the stress of the cables.

Figure 1: System Positioning

1.4 Project Approach

In the first phase of the project the V-Model development strategy has been chosen. The V-Model is the most widely used approach for project development; it is described in the following figure.

Due to the lower complexity of the project compared to a much more complex company

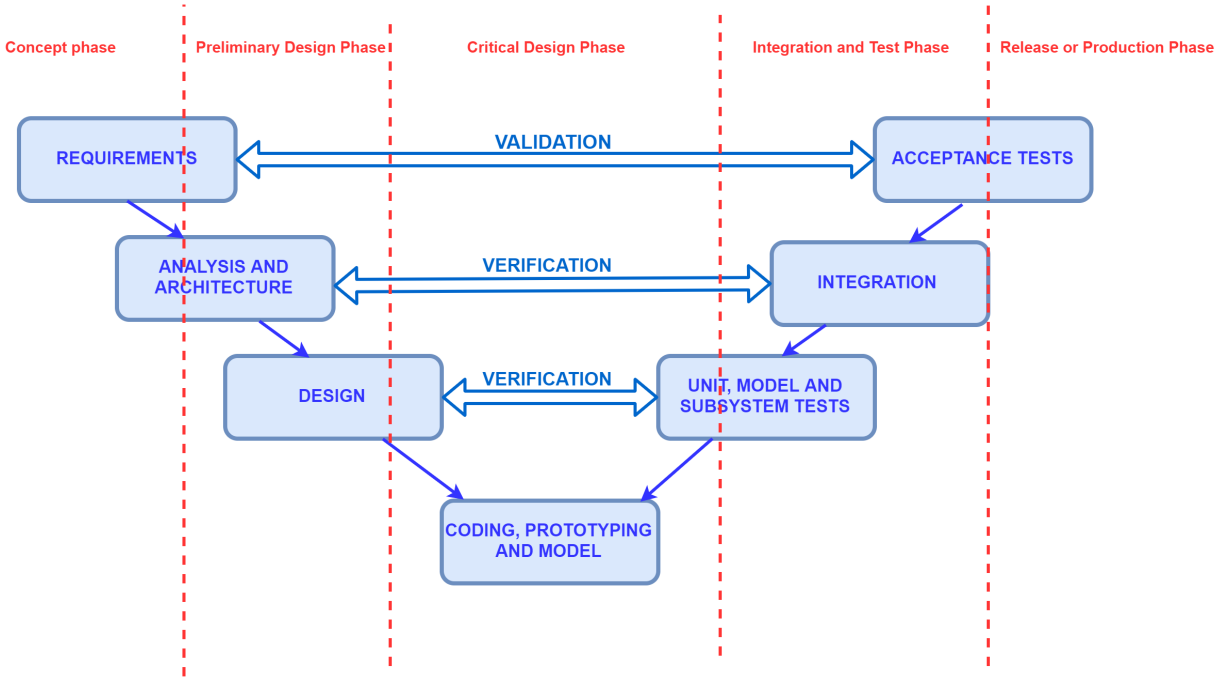


Figure 2: V-Model

project and due to the time constraint, our approach is based on a modified version of the V-Model, in order to make the development of the project as flexible and quick as possible. The modified V-model approach is described in the following image. The resulting

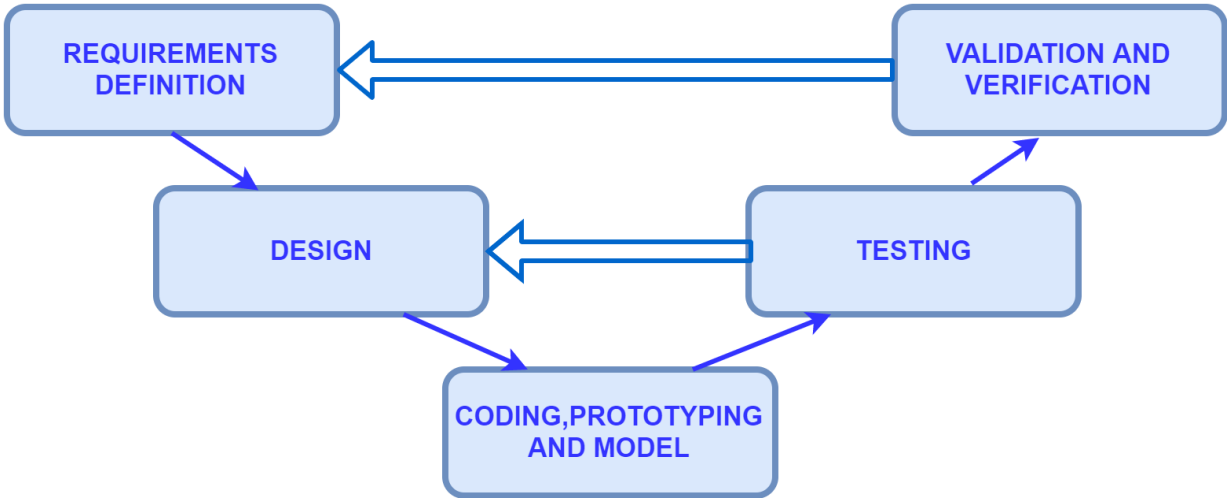


Figure 3: Modified V-Model

methodology is obtained both from the V-Model and from the waterfall approach. The reason behind this choice is related to the high number of short term decisions that have to be taken and to the fact that due to the strict timing constraints it is not possible to spend too much time looping on a phase of the project if a design error arises.

1.5 Software tools

In this section the software tools chosen as support for planning and design activities are listed and a brief description is provided.

- **Open Project**: a web-based open source project management software that supports teams along the entire project life-cycle. It allows the creation of Gantt charts, the assignment of tasks to specific team members, the tracking of meetings and work sessions;
- **GitHub**: an implementation of a Version Control System that allows to store data while keeping track of its modifications and to parallelize the execution of tasks between different team members;
- **TeXnik Center**: LaTeX text editor used for the redaction of the documentation;
- **One Note**: a note management software used to redact meeting reports and to write down ideas and useful information in a fast but organized way;
- **Telegram**: a messaging software used to easily communicate and discuss non-critical topics;

- **STM8 Cube MX**: a graphical tool that allows configuring STM8 MCUs easily, also generating the corresponding configuration reports;
- **ST Visual Develop**: IDE that provides debug and diagnostic features for STM8 applications while they are executing on the device by reading and displaying variables in real-time;
- **LTSpice**: software used for electrical simulation of circuits;
- **Autodesk Eagle CAD**: an electronic design automation software enabling printed circuit board (PCB) design, connection of schematic diagrams, component placement and PCB routing. Used for the design of the custom board;
- **Autodesk Fusion 360**: a 3D modeling software for mechanical design. Used for the design of the custom board's enclosure.
- **Drawio**: diagram software used to produce flow charts and block schemes.
- **Doxygen**: software used for the automatic generation of documentation from software;
- **System Workbench for STM32**: IDE for STM32 programming, used for the user interface firmware;
- **STM32 Cube MX**: a graphical tool that allows configuring STM32 MCUs easily, also generating the corresponding configuration reports .
- **NI LabView**: software from National Instruments that provides a wide range of functions. Used for user-friendly data visualization.
- **Lightworks**: video editing software used for the redaction of the project presentation;
- **Arduino IDE**: IDE used to write, compile and load code for the MCUs;
- **SolidWorks**: CAD software used for mechanical design and air flow simulation;
- **Power Point**: software used in order to prepare the presentation.

1.6 Cost analysis

In this section an estimation of the costs performed during the requirement evaluation phase is presented considering the main cost factors. Constraints related to the cost:

- The cost must be kept as low as possible and the budget of 100 € provided by University of Bologna must not be exceeded.

Cost factor	**Estimated cost in Euro**
Evaluation board	30
Evaluation sensors	20
Components, sensors and connectors	45
Custom board production	5
Total	**100**

Table 1: Cost evaluation

- The realization of the custom board must lead to a cost reduction with respect to the realization of the prototype.

In this preliminary assessment the total cost is compliant to the budget requirement and it is divided as follows.

Prototype realization includes the costs of the evaluation board and of the sensors with which it must interface.

$$\textbf{Prototype Cost} = 30 + 20 = 50€$$

Custom realization includes the components needed to assemble the board (sensors and connectors included), plus production costs.

$$\textbf{Custom Cost} = 45 + 5 = 50€$$

1.7 Risk Analysis

In the following table a risk matrix that considers all the parts of the project that can be affected by problems and their related causes are shown. The most critical parts of

Risks / Project Phase	**Timing**	**Functional objectives**	**Supplier sourcing**	**Design**	**Production**
Management					
Hardware					
Software					
Documentation					
Testing					

Table 2: Description of the main risk factors

the project are mainly two. Firstly, the production of the custom board could face many issues because this phase of the project is strictly related to the suppliers. In fact, several problems that could arise have been identified as supplier related.

- Component shipping delay;
- Wrong components are shipped;
- PCB production delay;
- PCB shipping delay.

On the other hand all the design phases, both hardware and software, may encounter problems related with

- Wrong evaluation of the requirements;
- Wrong choice of the components due to lack of time in the component definition phase;
- Software related problems;
- Hardware related problems.

The phases related to project planning and documentation redaction have been assessed as less critical.

2 Project Management Plan

An accurate planning of the design phases is required in order to complete the project successfully. This planning allows to divide tasks among team members and to analyze all the critical factors that can cause delays. Each member of the team has a defined role and is in charge of controlling and managing a specific subset of the work. The following sections illustrate the main activities related to project management and organization:

- Work Breakdown Strategy
- Macro-Planning
- Weekly reports

2.0.1 Project Scheduling

The scheduling of the activities has been performed through **Open Project**, an open source management software that is freely available in the hosted version. This software was used to assign tasks, with their specified deadline and description, to each member of the team. Moreover, it allowed the subdivision of the work in macro-activities and the generation of the Gantt chart.

2.1 Work Breakdown Strategy

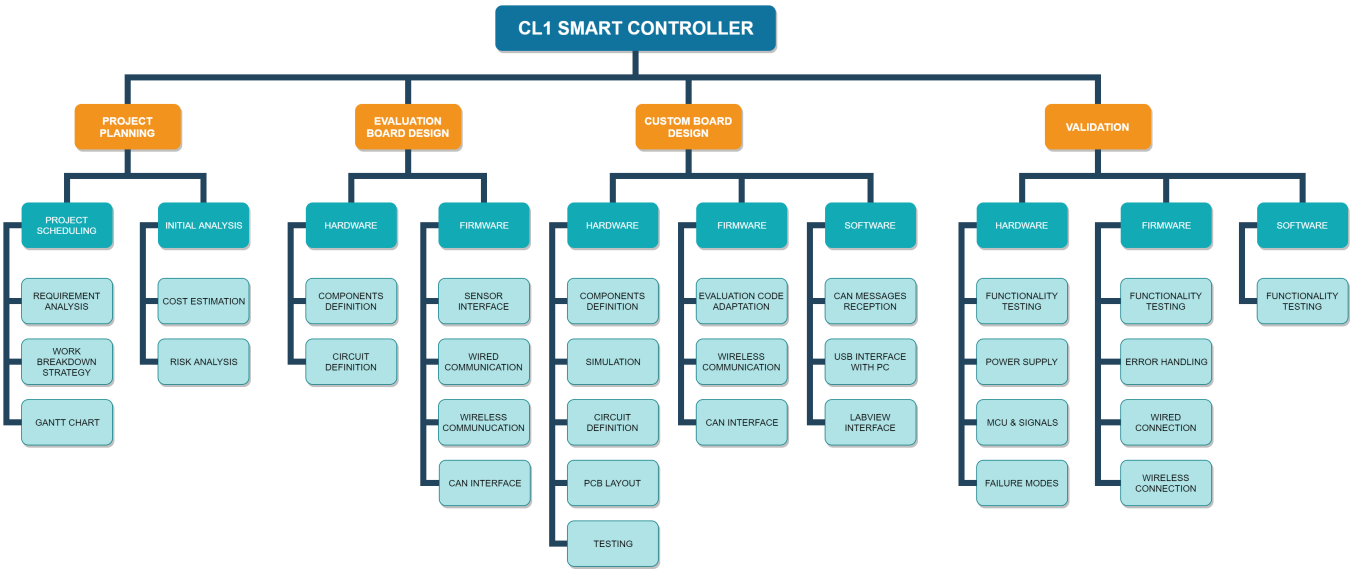


Figure 4: Work Breakdown Strategy

The main deliverables that must be provided are listed in the Work Breakdown Strategy and the related activities are scheduled in the Gantt chart shown in the next section.

2.1.1 Macro-Planning

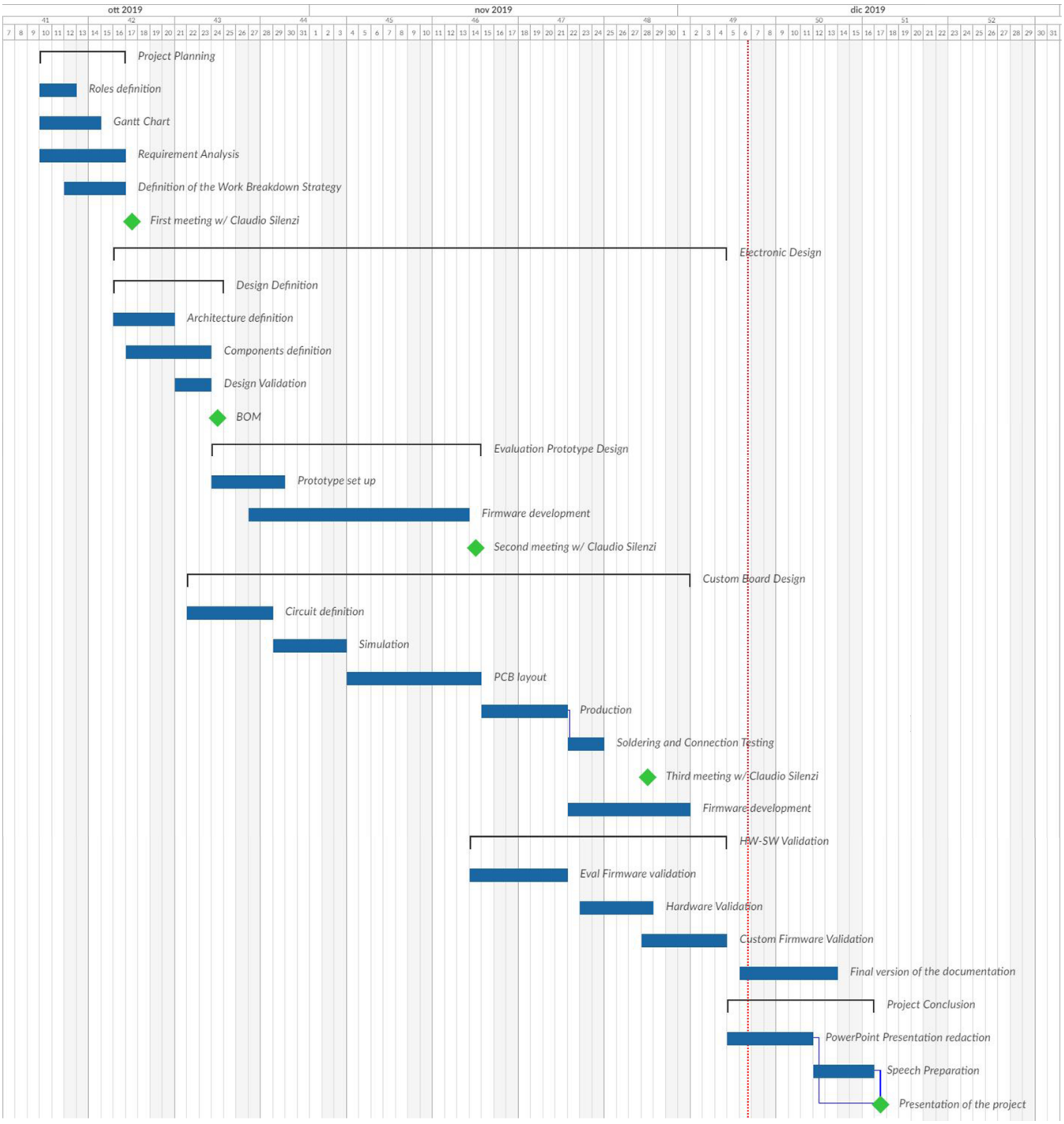


Figure 5: Gantt chart

The Gantt chart illustrates the main phases of the project. All minor tasks are scheduled

using Open Project in order to make the tracking of each member's work more efficient and the evaluation of the current state of the project easier.

2.1.2 Reports

A file with the description of the main goals achieved and of the most critical problems encountered by the whole team during the past week was written down by one team member each week. The following elements were highlighted in each report:

- Description of the main goals of the current section of the project;
- Description of the main choices made;
- Brainstorming on the current and future phases of the project.

The aim of these reports is to keep track of work status, of new ideas and of the overall structure of the design, in order to make each design or development phase more readable and comprehensible. Weekly reports are listed in Appendix F.

2.2 Major Project Milestones

The major milestones defined during project planning, together with the main objectives that have to be reached, are listed below.

Milestone	**Completion date**	**Main objectives**
First Progress Meeting	October 17th, 2019	- Definition of role and responsibilities of each team member; - Evaluation of the specifications; - Project scheduling; - Project charter realization.
BOM completion	October 24th, 2019	- Definition of all the components; - Simulation of the behaviour of the components that need to be simulated;
Second Progress Meeting	November 14th, 2019	- Completion of the software development for the evaluation board; - Completion of the realization, simulation and layout of the custom board.
Third Progress Meeting	November 28th, 2019	- Validation of the software of the evaluation board; - Soldering and testing of the custom board;
Presentation of the project	December 19th, 2019	- Completion of the validation phase; - Finalization of the documentation; - Realization of the final presentation.

Table 3: Defined milestones and description of the main objectives.

2.3 Assignment of the Roles and Responsibilities

The team is composed by 6 members, each one being in charge of managing a different part of the project and having specific responsibilities. The main roles assigned to each team member are listed below.

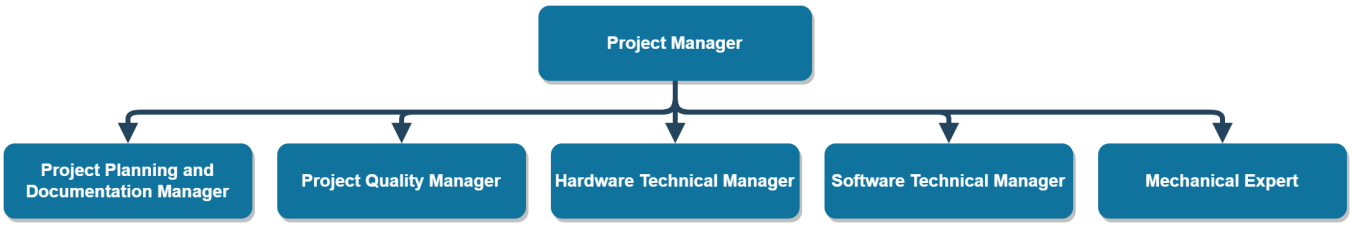


Figure 6: Block diagram that shows the defined profiles

Role	**Team Member**	**Main responsibilities**
Project Manager	Di Loro Giorgio	Responsible for communication, cooperation and interfacing between different managers and team members. Ensures the coordination between different compartments and provides solutions in case of conflicts.
Project Planning and Documentation Manager	Mancini Luca	Responsible for the compliant redaction of the deliverable documentation required, for activity planning and for the overall organization.
Project Quality Manager	Villar Tovar Armando Ruben	Responsible for the validation processes at hardware, software and general levels, ensures the compliance of the adopted solutions and their correct design and implementation from a quality point of view.
HW Technical Manager	Biguzzi Annachiara	Responsible for the hardware design and implementation, head and active member of the hardware development section of the project.
SW Technical Manager	Ravagli Giacomo	Responsible for the software design and implementation, head and active member of the software development section of the project.
Mechanical Expert	Zhou Tong	Responsible for mechanical study and realization of the enclosure for the custom board and active member of the quality and validation section of the project.

Table 4: Description of main responsibilities of each profile

2.4 Documentation Required

In the first phase of the project an accurate analysis of the documentation that is needed in order to fulfill the requirements has been performed.

- Project charter: it includes the main objectives of the project, the work breakdown strategy, project planning documentation (reports, Gantt chart) and the definition of roles and responsibilities;
- Accurate description of all the choices made for both evaluation board and custom board implementations, with advantages and disadvantages;
- Description of the design process at hardware, firmware and software level;
- Description of both performed tests and theoretical tests required for validation;
- User manual;
- Files (datasheets, code, schematics) attached to the documentation;
- Document for work evaluation written by each team member.

3 Requirements Specification

In the following section all the requirements set by the customer are listed.

Inputs:

- 1 analogue single ended/digital Barometric pressure sensor
  - Range 800-1200 mBar;
  - Operating temperature from -40 to 85 °C;
  - Absolute accuracy 250 Pa.
  - Relative accuracy 25 Pa.
- 1 analogue single ended/digital Accelerometer one/three axis:
  - Range up to $\pm 15G$;
  - Resolution up to 10 bit at full scale;
- Optionally the sensors can be located on a separate board that transmits wireless to a main board.

Outputs:

- 1 CAN/Ethernet line to be linked to a PC for data monitoring;
- Optionally: 1 HiSide 1 A producing a PWM that is a function of an input.

Power supply:

- Range 10-16 V
- Protection towards over-voltage, polarity inversion;
- Max consumption typical <100 mA (wireless devices not included).

Other specifications:

- Operating temperature range 0-105°C ;
- Volume and weight: as small as possible (the physical dimension will be part of the evaluation) (smart IC packages can be just only evaluated);
- Connectors: free;

- Heat rejection: detailed data must be provided;
- IPX protection: not required.

In addition the final custom board must provide:

- Automotive compliance;
- Reduced size with respect to the evaluation version;
- Protection, recoveries, diagnostics;
- Self check systems;
- Cost reduction if possible compared to the evaluation version;
- Qualification/validation processes.

4 Analysis

4.1 Hardware Goals

The three main goals are

- **Evaluation Board**: creation of a working prototype by using an evaluation board. This board should be a proof of concept for the custom board, hence it must perform its intended functionality as required.
- **Custom Board**: design of a custom board, which must be smaller and cheaper than the evaluation prototype. The goal that has been set by the team is to actually realize, solder and test the PCB in order to have a working custom board.
- **Reduced Size Custom Board**: design of a custom board which is smaller than the actually realized custom board.

4.2 Firmware Goals

The two main goals are

- **Evaluation Board**: creation of correct firmware able to work according to specifications on the evaluation board.
- **Custom Board**: adaptation of the code developed in the previous phase in order to obtain a working custom prototype.

4.3 Concept

4.3.1 First Solution

The initial idea was to realize a system composed by a single board connected to the PC through a CAN line for data monitoring. The main components needed on the controller, as described in the specifications, are a MCU, a barometric pressure sensor and an accelerometer. Additionally, a step down converter is used for converting the 10-16 V power supply to the correct supply voltage required by the MCU and a CAN transceiver is required for implementing the physical layer of the CAN protocol.
In the following figure the high level definition of this first concept is shown.

4.3.2 Update to Wireless Solution

Due to the update of the specifications, the wired solution has been revised because of the difficulty of realizing wired connections that could reach the DRS wing (for example), and the result is shown in the following figure.

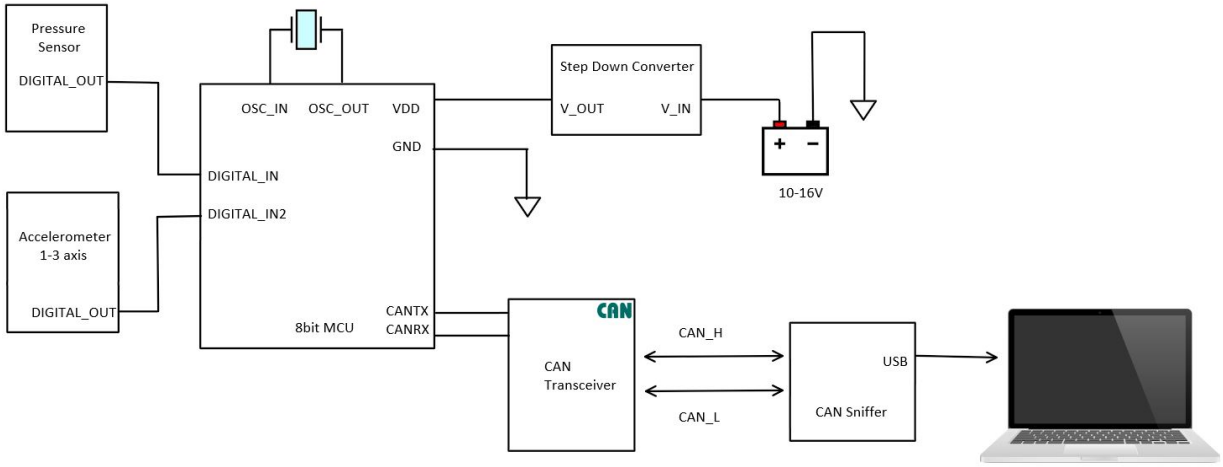


Figure 7: Description of the first high level architecture of the system

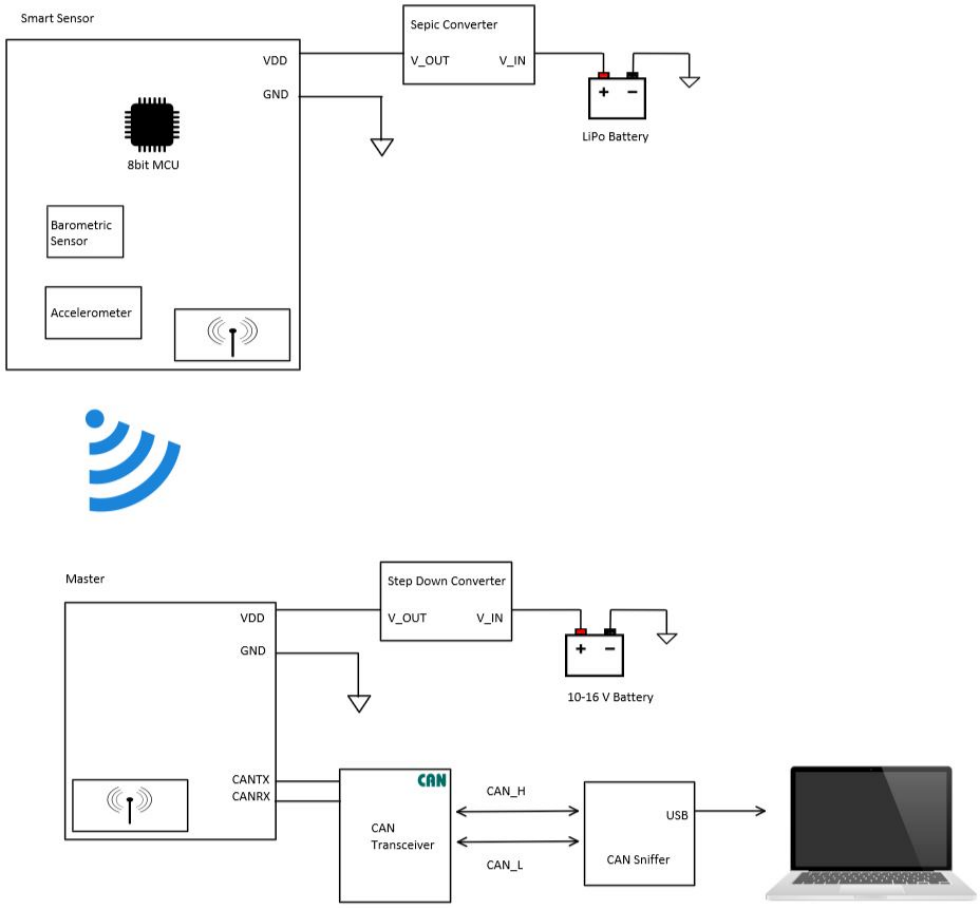


Figure 8: Description of the wireless solution architecture

4.3.3 CL1 - Update to Final Modular Solution

The final architecture of the system is based, as before, on the use of a wireless connection between the two boards and of a CAN interface for connection to the PC. What has been improved is the modularity of the system. The idea is to realize two identical boards:

- **Board 1 - Controller Board** The controller board is supplied by the battery of the car but could be also connected to a LiPo cell in order to use it as a sensor board. It can be equipped with sensors in order to add a measurement point in the vehicle.
- **Board 2 - Sensor Board** The sensor board is supplied by a LiPo battery cell and it interfaces with the controller board via wireless communication. In case of problems affecting the controller board, the sensor board could be easily reconfigured and wired directly to the CAN line of the vehicle (taking care of possible wiring issues related to the position in which the sensor board is mounted).

The desired operation mode *controller board* or *sensor board* can be selected with a simple firmware change.
This particular design strategy leads to multiple advantages:

- The entire system can be implemented with just a single board design;
- Easy reconfiguration via firmware;
- Cost and design time reduction;
- Possibility to use the controller board as an additional measurement point.

Disadvantages:

- Increase in board size due to the space reserved for additional sensors;
- If sensors are mounted on both boards the cost of the final system increases;

Due to the modular nature of the final prototype the product has been called CL1 ("*Clone*").

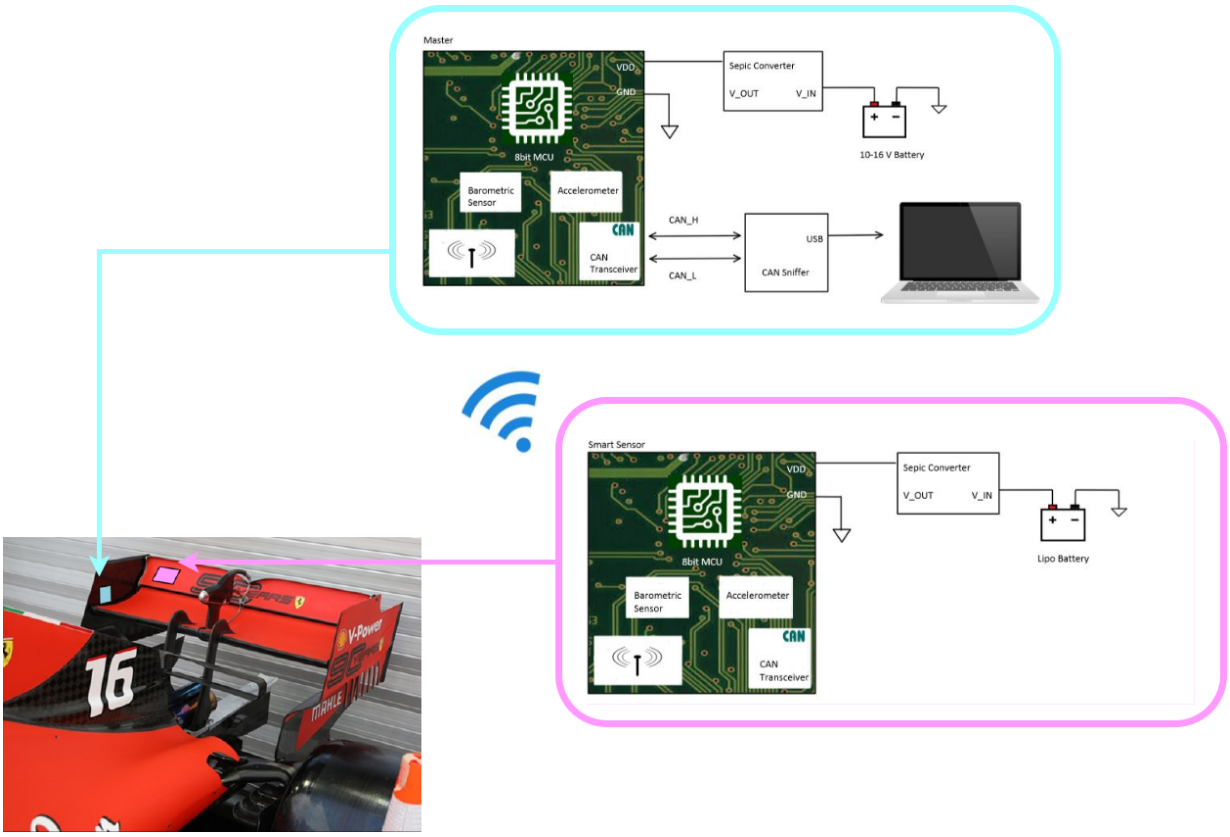


Figure 9: CL1 solution

4.4 General Overview of the Decisional Process

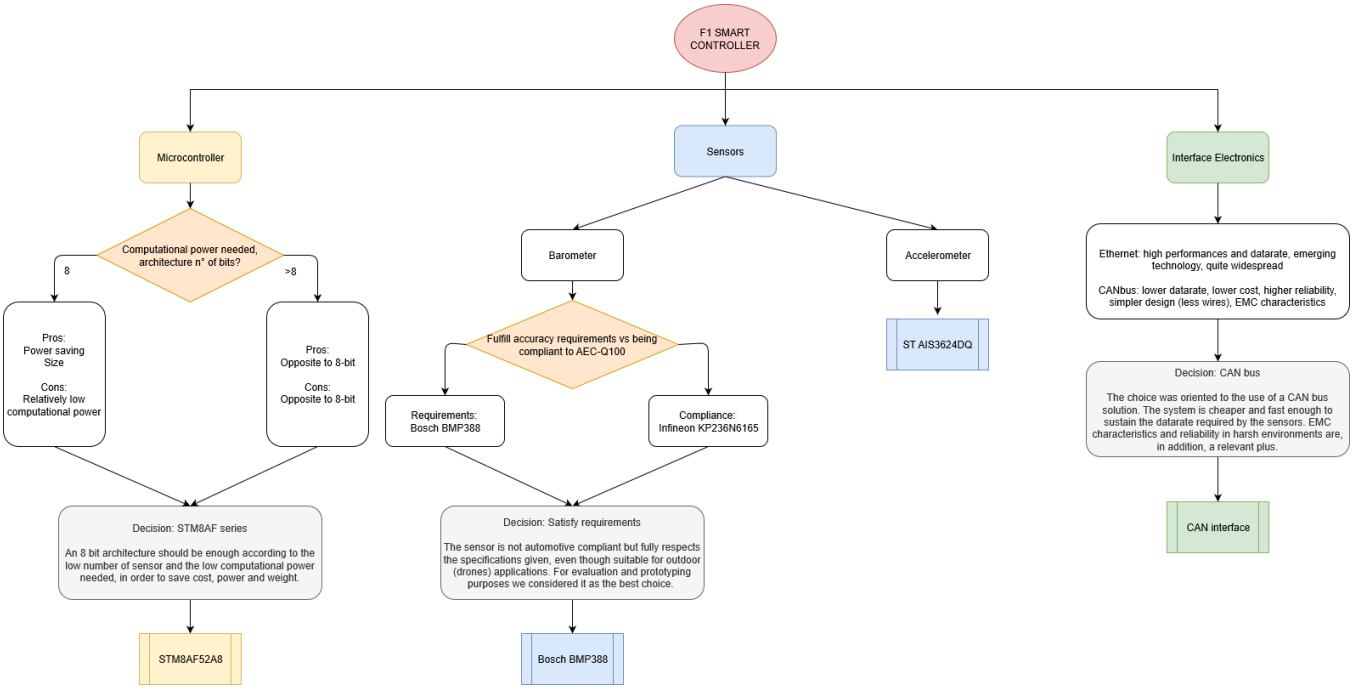


Figure 10: General Decisional Process

4.5 MCU

Several solutions for the choice of the microcontroller were evaluated by analyzing both advantages and disadvantages of each one of them.

Required features
Main microcontroller features needed:

- AEC-Q10x qualified, compliant for automotive applications.
- 8-bit MCU
  In order not to oversize the microprocessor with respect to the application (considering the small amount of computational power needed by the system), an 8-bit architecture was considered suitable.
- SPI and I^2C interfaces
  For digital sensors and wireless module connection.
- CAN interface
  For data monitoring from the PC.

4.5.1 Evaluated MCU

The main features of the MCUs that have been evaluated are displayed in the following table.

	ATM328P	**STM8AF5288T**
Manufacturer	Microchip	STMicroelectronics
Architecture	8-bit	8-bit
Interfaces	USART, SPI, I^2C	USART, SPI, I^2C
CAN support	Yes	Yes
RAM memory	4 KBytes	6 KBytes
FLASH memory	64 KBytes	128 KBytes
AEC-Q100	Yes	Yes
Package	64-lead TQFP	LQPF48

Table 5: MCUs features

ATM328P
Advantages

- Compatible with Arduino code;
- All the specifications are met;

Disadvantages

- Both RAM and FLASH memory are smaller than STM8's memory;

STM8AF5288T
Advantages

- Availability of a small 32 pin VFQFP32 package that could be used to design a reduced size board;
- All the specifications are met;
- More FLASH memory with respect to the other solution;
- Previous experience of some team members with STMicroelectronics MCUs;

Disadvantages

- Availability of a small 32 pin LQFP32 package that is solderable but does not support the CAN interface;

Also *Texas Instruments* MCUs have been considered but during brainstorming the following issues came out:

- Difficult to program;
- MCUs are more expensive than STM-based solutions;
- Evaluation boards are more expensive than STM-based solutions;

The number of pins needed for the configuration of the custom solution in our application is small (about 20 pins used for supply, SPI interface, CAN interface and eventually an analog input for ADC). ATM328P is available in 64-lead TQFP, this means that the majority of the MCU pins would be disconnected. The STM8AF5288T has 48 pins, thus it allows to avoid leaving a high number of disconnected pins.
Moreover, using its VFQFP32 version it is possible to further reduce the number of disconnected pins and the overall size of the MCU.

4.6 Sensors

4.6.1 Barometric Pressure Sensor

The pressure sensors analyzed are described in the following table.
The advantages and disadvantages of all these sensors are listed below.

	BMP388	**SMP580**	**KP236**	**LSP22HH**
Manufacturer	Bosch Sensortec	Bosch Sensortec	Infineon	STMicroelectronics
Interfaces	SPI, I^2C	SPI	Analog	SPI, I^2C
Measurement Range	30-125 kPa	40-115 kPa	60-165 kPa	26-126 kPa
Absolute Accuracy	50 Pa	1 kPa	1 kPa	100 Pa
Relative Accuracy	8 Pa	Not provided	Not provided	10 Pa
Temperature Range	-40 °C - +85 °C	-40 °C - +125 °C	-40 °C - +125 °C	-40 °C - +85 °C
AEC-Q100	No	Yes	Yes	No
Package	LGA	SOIC 8	PG-DSOF-8-16	LGA

Table 6: Barometric Pressure sensors overview

BMP388
Advantages

- Very high accuracy: all the specifications are met;
- Small size: reduced area required on the board;

- Digital interface;
- Availability of libraries: due to the strict deadlines the presence of libraries that allow to easily interface with the sensors is key in order to save time;
- High availability of sensor evaluation boards.

Disadvantages

- It is not automotive compliant;
- It is difficult to solder because of its size and package (LGA);
- Smaller temperature range compared to other sensors considered.

SMP580
Advantages

- It is automotive compliant;
- Due to its SOIC 8 package it is easy to solder;
- Signal conditioning is integrated in the ASIC.

Disadvantages

- Relative accuracy is not provided by the datasheet (which provides poor overall data): due to this problem it is difficult to state if the specifications are met. In order to have more information, the manufacturer was contacted but no clear response has been obtained;
- Absolute accuracy of 1 kPa is too high;
- Maximum detectable value of 115 kPa is too low;
- Difficult to find it on the most popular electronic components vendors' websites;
- Sensor evaluation board not found.

KP236
Advantages

- It is automotive compliant;
- Easy to solder;
- Range meets the specifications.

Disadvantages

- Relative accuracy not provided by the datasheet;
- Absolute accuracy of 1 kPa is too high;
- Bigger size with respect to other sensors considered;
- The analog interface requires signal conditioning and amplification: this adds complexity to the circuit.
- Sensor evaluation board not found.

LPS22HH
Advantages

- Very high accuracy: all the specifications are met;
- Small size: reduced area required on the board;
- Digital interface;
- Availability of libraries: due to the strict deadlines the presence of libraries that allow to easily interface with the sensors is key in order to save time;

Disadvantages

- It is not automotive compliant;
- It is difficult to solder because of its size and package (LGA);
- Smaller temperature range compared to other sensors considered;
- Evaluation board not available.

After the evaluation of pros and cons of these solutions, the choice fell on the BMP388 barometric pressure sensor. This sensor perfectly meets the specifications related to measurement range and accuracy.
Moreover, even if the operating temperature range is smaller than the ones of other two

proposed solutions, the sensor will not be subject to high temperatures because of the place in which the board will be mounted and because of the specific application it is meant for. Another reason why this sensor has been chosen is the availability of sensor evaluation boards: thanks to these it is possible to use the same sensor in both the evaluation and the custom board, leading to a complete reuse of the code.
Since this type of application requires very high accuracy which cannot be achieved with the automotive compliant sensors that have been found on the market, the BMP388 is suitable. BMP388 is typically used on drones, this means that it is typically subject to stress and vibration. In the following sections a complete analysis of the kind of tests that can be performed in order to prove the compliance for automotive applications will be provided. It was concluded that the BMP388 and the LPS22HH have comparable characteristics. The reason why BMP388 has been chosen is that an evaluation board is not available for LPS22HH (there is an adapter board for an STM32 Evaluation Kit but its compatibilities are not clear).

4.6.2 Accelerometer

	AIS3624DQ	**IAM-20381**
Manufacturer	STMicroelectronics	TDK
Interfaces	SPI, I^2C	SPI, I^2C
Measurement Range	$\pm 6g, \pm 12g, \pm 24g$	$\pm 2g, \pm 4g, \pm 8g, \pm 16g$
Axis	3-axis	3-axis
Output resolution	12 bit	16 bit
Temperature Range	-40 °C - +105 °C	-40 °C - +85 °C
AEC-Q100	Yes	Yes
Package	QFN24	LGA 16 pin

Table 7: Accelerometer overview

AIS3624DQ
Advantages

- It is automotive compliant;
- Specifications are met in terms of resolution and range;
- The manufacturer provides libraries for interfacing with the sensor;
- Previous experience of a team member with STMicroelectronics accelerometers: this allows to reduce the time needed to learn how to use libraries and software tools.

Disadvantages

- Difficult to solder;
- The only evaluation board for this accelerometer that has been found is an adapter for a STM32 Evaluation Kit: it is not clear if it is compatible with STM8A-DISCOVERY.

IAM-20381
Advantages

- It is automotive compliant;
- Specifications are met in terms of resolution and range.

Disadvantages

- Libraries are not available;
- Sensor evaluation boards were not found.

To overcome the problem of soldering, the sensor selected for the custom board is the one-axis version of AIS3624DQ (AIS1120SX) that has the SOIC 8 package. The characteristics of the AIS1120SX are shown in the following table:
It is important to notice that the AIS1120SX still complies to specifications.

	AIS1120SX
Manufacturer	STMicroelectronics
Interfaces	SPI
Measurement Range	$\pm 120g$
Axis	1-axis
Output resolution	12 bit
Temperature Range	-40 °C - +105 °C
AEC-Q100	Yes
Package	SOIC 8

Table 8: AIS1120SX accelerometer

This accelerometer is used for the first version of the custom board because it is solderable without an automatic machine; the AIS3624DQ is then used for the reduced size design .

Solution adopted due to problems with the supplier
In order to overcome the problem related with the supplier (the accelerometer was missing when the components were delivered), another accelerometer (equivalent to the one previously chosen for the custom board) has been selected.
In the following table the updated features for the custom board accelerometer:

	AIS2120SX
Manufacturer	STMicroelectronics
Interfaces	SPI
Measurement Range	$\pm 120g$
Axis	2-axis
Output resolution	12 bit
Temperature Range	-40 °C - +105 °C
AEC-Q100	Yes
Package	SOIC 8

Table 9: AIS2120SX accelerometer

4.7 Evaluation Board

The choice of the evaluation board is directly related to the choice of the microcontroller. In fact, the same micro-controller is used both in the prototyping phase with the evaluation board and in the custom board, in order to allow an almost complete reuse of the code.
The STM8A-DISCOVERY is an evaluation kit containing two different evaluation boards mounting a STM8AF and a STM8AL microcontroller respectively. Due to the goal of complete reuse of the code, an additional NUCLEO2088RB board, which provides compatibility with the same libraries used for the STM8AF, has been used. The NUCLEO board deploys a STM8S MCU. The idea is to use the two STM8A boards to implement the system:

- **STM8AF board** is used as controller board because the MCU supports the CAN interface. The CAN transceiver is not needed because it is already embedded in the board.
- **STM8AL board** is used as sensor board, both the evaluation boards of the pressure sensor and of the accelerometer are connected to the pins of the STM8AL board.

Barometric Pressure Sensor
Adafruit BMP388 has been chosen in order to realize the evaluation board prototype using the same hardware components that will be deployed on the custom board. Because of this choice, the code implemented for this component can be easily adapted for the second part of the project.

Accelerometer
Adafruit LIS3DH has been chosen in order to realize the evaluation board prototype even if it is not the same sensor that will be used for the second part of the project. The accelerometer chosen for the custom board is the AEC-Q100 compliant version of LIS3DH.

Figure 11: STM8AF Board

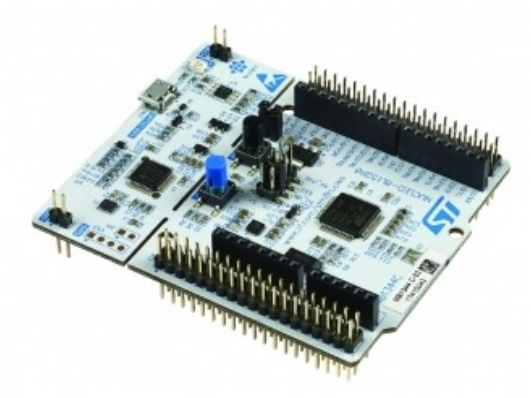


Figure 12: Nucleo208rb

4.8 Power Supply

In order to realize a fully modular system, both the controller and the sensor board must be capable of being supplied by

- LiPo Battery 3.7 V

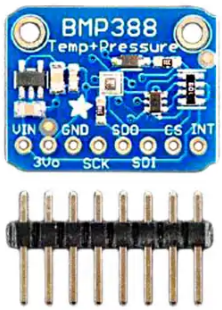


Figure 13: Adafruit BMP388

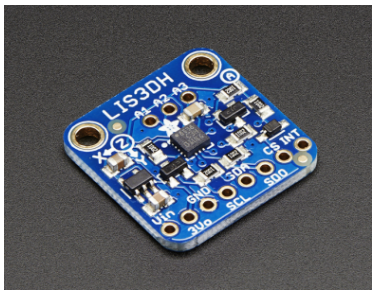


Figure 14: Adafruit LIS3DH

- Vehicle Battery 12 V

Because of this, the power converter is required to obtain the supply voltage needed by our system (MCU and sensors), 3.3 V, both from the 12 V of the vehicle battery and from the LiPo battery. In the worst case, when it is completely discharged, the LiPo provides a voltage equal to 3 V.

4.8.1 Power Converter Choice

LDO
Advantages

- Easy design on the PCB, only few capacitors are needed;
- Low cost;
- Low noise.

Disadvantages

- Low efficiency;

- The LDO is a step-down converter, therefore it is only suitable for obtaining 3.3 V from the 12 V vehicle battery.

SEPIC
Advantages

- High efficiency;
- The SEPIC can both step-down and step-up the voltage;
- Low thermal dissipation;
- It can handle large output current.

Disadvantages

- High cost;
- Complex design on the PCB, a lot of components with specific values are needed;
- Affected by switching noise.

In order to have a fully modular system that can be supplied by either the 12 V vehicle battery or the 3 V (worst case) LiPo cell, the SEPIC solution has been chosen. A SEPIC solution is described in the following table:

	LT8330
Manufacturer	Analog Devices
Operation modalities	Boost, SEPIC, Inverting converter
Vin range	3-40V
Vout range	Positive or Negative Output Voltage Programming with a Single Feedback Pin
Max output current	1 A
Temperature Range	-40 °C - +125 °C
AEC-Q100	Yes
Package	6-LEAD TSOT-23

Table 10: LT8330 overview

LT8330
Advantages

- Perfectly matches with our supply requirements;
- Wide input and output voltage ranges;
- The manufacturer provides a detailed description of the layout and advised components;
- The package is easy to be soldered.

Disadvantages

- High output current that is not needed for our application;
- High cost.

More cost efficient integrated circuit for this kind of application probably exist, but due to the small amount of time provided for the definition of the components this solution has been evaluated as suitable. In fact, before ordering the component, simulations were made in order to prove that the LT8330 behaves as expected.

4.8.2 LiPo Battery 3.7V 150 mAh

Multiple solutions have been analyzed, due to the different kinds of batteries' technologies and shapes.
Requirements

- Enough energy to last 2/3 hours;
- Small size, the package should be as small as possible;
- Low weight, it must not impact on the weight of the system too much.

For this reason, cylindrical and prismatic cells were not considered. The best solution, that perfectly fits with these strict requirements on size and shape, is a pouch cell. Pouch cells (30x20 mm) are available on the market at a really low price.
Advantages

- Small: 30x20 mm;
- In drones, the current requested to the battery is very high (drones' batteries must handle 20/30C discharges) but this is not the case of our application. For this reason the discharge rate is lower (1C) and this leads to a lower weight of the battery.
- The shape allows a simple design for the box.

Disadvantages

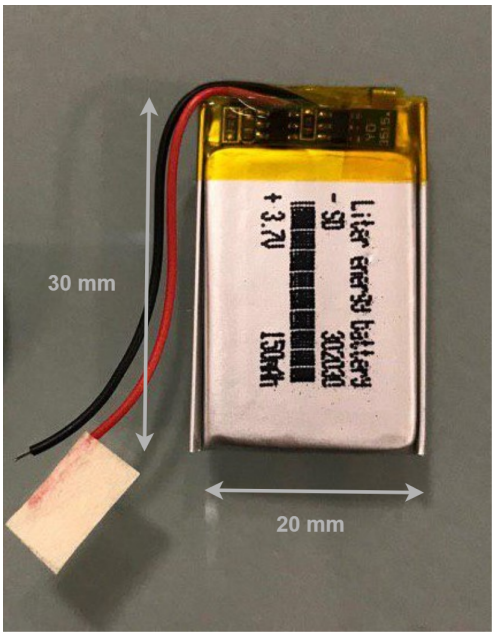


Figure 15: LiPo battery cell 150 mAh

- At least 30 days are needed for it to be delivered from china. This leads to a dramatic increase in risk related to the supplier.

The choice comes from a trade-off between the duration of the battery (that is estimated to be of around 2 or 3 hours due to the fact that the system has to be used during tests), its size and its weight. In order to avoid damage to the cell due to collisions and stress, a proper box, in which the cell will be inserted, has been designed.

4.9 Wireless connection

4.9.1 Main Wireless Features

Main specifications for wireless connection:

- Automotive compliant if available;
- 2 Mbps data rate;
- Low power consumption;
- Easy to interface with the MCU;
- The communication distance is less than 1 m;
- Simple architecture: no need of a more powerful MCU or of an MCU embedded in the module;

- Availability of an evaluation module to be used for the evaluation board implementation;
- Short range needed for this application: long range devices are useless.

4.9.2 Proposed Solutions

The pros and cons of the different wireless technologies that have been taken into account are listed in the following table. It has been decided to discard solutions with a complex architecture, which are hard to manage, with high power consumption or designed for long-range communications, such as ANT, Lora and Wi-Fi. The 3 evaluated technologies are:

- **Bluetooth 4.0 Low Energy**, one of the versions of Bluetooth that are available on the market. It sacrifices range and data throughput for significant improvements in power consumption with respect to Bluetooth V2.1+EDR (Classic) and Bluetooth v3.0+HS (High speed).
- **ZigBee**, a communication protocol used for very dense wireless sensor networks.
- **NRF24L01+**, device that allows to communicate at a frequency of 2.4 GHz (ISM band).

	Bluetooth 4.0 Low Energy	**ZigBee**	**RF 2.4 GHz**
Frequency	2.4 GHz	2.4 GHz	2.4 GHz
Max Raw Bit Rate	1 Mbps	0.25 Mbps	2 Mbps
Typical Data Thoughput	0.27	0.2	N/A
Max Outdoor Range	50 m	100 m	10 m (printed antenna)
Relative Power Consumption	Very low	Very Low	Low
Network size	Undefined	64000	125

Table 11: Wireless technologies

Bluetooth
Advantages

- High range;
- High data rate;
- Low power consumption;
- Widespread, well-built protocol.

Disadvantages

- Time overhead for establishing the connection;
- Overall protocol overheads on messages;
- Uses the full frequency spectrum regardless of the specific system setup.

NRF24L01+
Advantages

- High data rate;
- Suitable for short range communication.

Disadvantages

- Low range with respect to BLE;
- Not as widespread as other protocols.

4.10 Wired Connection - Output

According to the specification given by the customer, two different solutions to stream data to the PC can be adopted:

- CAN interface;
- Ethernet.

CAN
Advantages

- Widespread in automotive applications;
- Simpler design on the PCB;
- Supported by the MCU;
- Provides reliable communication;
- Only two wires needed;

Disadvantages

- Lower data rate than Ethernet;
- A device that acts as a CAN sniffer is needed;

Ethernet
Advantages

- Increasingly used in automotive applications;
- Simpler connection to the PC (no need for a device that acts as sniffer);
- High throughput.

Disadvantages

- Does not provide a secure communication;
- Low protocol efficiency;
- Eight wires needed;
- Big connector needed;
- More components needed compared to CAN;
- More precautions required for physical design on the PCB (traces length, width and positioning) compared to CAN;

The circuits that have to be realized for both CAN and Ethernet interfaces are shown below. CAN needs less components with respect to Ethernet and no connector is needed on the board (the RJ45 connector is bulky).
Considering the idea of realizing a modular solution, the use of CAN allows to expand the system in a simple way because CAN bus' nature gives the possibility to easily add nodes to the network. The main reasons for choosing CAN communication are listed below:

- The simpler design leads to the reduction of the size of the custom board;
- Reduction of components' costs;
- This bus is already widespread in the automotive industry;
- The application doesn't require high data rates;

Connectors
It has been decided to avoid inserting a connector in the first realization of the custom board, mainly for the following reasons:

- A connector that can satisfy the requirement set was not found before the components' choice deadline;
- A board connector is preferable but the connectors analyzed provide impractical connection and disconnection.

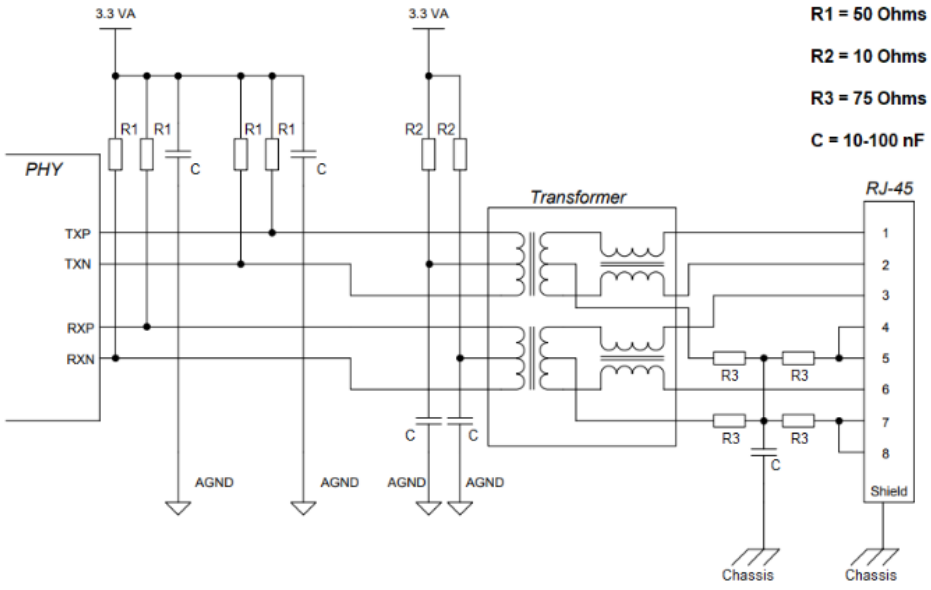


Figure 16: Ethernet physical description

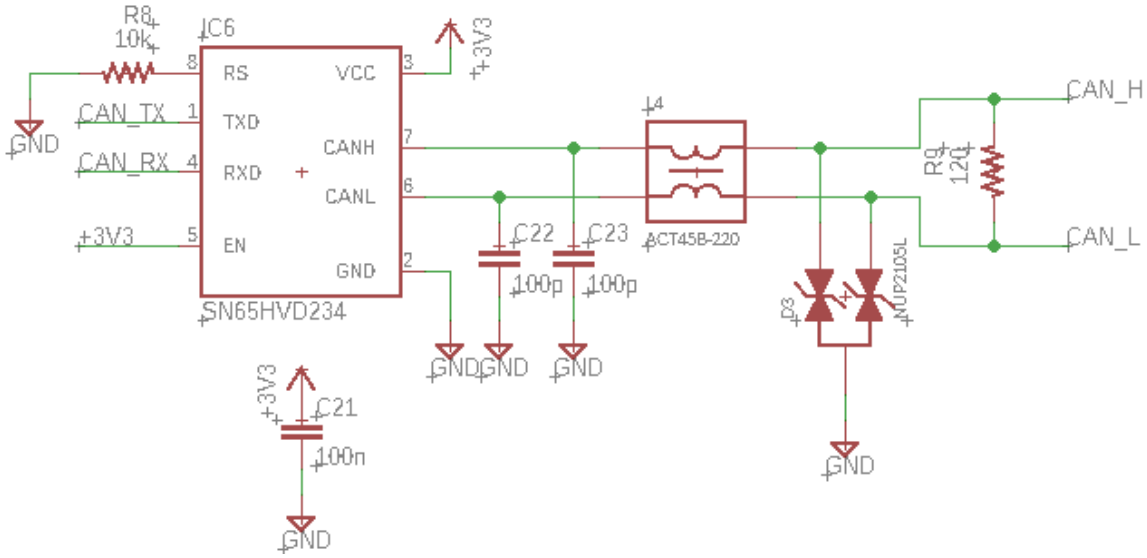


Figure 17: CAN physical description

4.11 Bill of Materials and Cost Evaluation

4.11.1 Evaluation Board

The components used for the evaluation board are shown below.

DigiKey code	Manufacturer code	Manufacturer	Description	Number of pie	Price €	Total Price €
497-13747-ND	STM8A-DISCOVERY	STMicroelectronics	DISCOVERY STM8A EVAL BRD	1	22,75	22,75
1568-1513-ND	PRT-12796	SparkFun Electronics	JUMPER WIRE F/F 6" 20PCS	2	1,77	3,54
1568-1511-ND	PRT-12794	SparkFun Electronics	JUMPER WIRE M/F 6" 20PCS	1	1,77	1,77
1568-1512-ND	PRT-12795	SparkFun Electronics	JUMPER WIRE M/M 6" 20PCS	1	1,77	1,77
1528-1516-ND	2809	Adafruit Industries LLC	ADAFRUIT LIS3DH TRIPLE-AXIS ACC	1	4,5	4,5
1528-2733-ND	3966	Adafruit Industries LLC	BMP388 - PRECISION BAROMETRIC PR	1	9,05	9,05
	NRF24L01	Nordic Semiconductor	Wireless Module	2	0,77	1,54
					Totale	44,92

Figure 18: Evaluation board BOM

4.11.2 Custom Board

The component used for the custom board are listed below. What is considered in the total price, in order to fulfill the specifications, is:

- 2x Power supply (one for each of the two boards);
- 2x MCU;
- 2x NRF24L01+ module;
- 1x Accelerometer (to be mounted only on the sensor board);
- 1x Barometric pressure sensor (to be mounted only on the sensor board);
- 1x CAN transceiver.

4.11.3 Custom Board with Sensors on both sides

The components used for the custom board are listed below. What is considered in the total price, in order to obtain two measuring points and a reconfigurable system, is:

- 2x Power supply (one for each of the two boards);
- 2x MCU;
- 2x NRF24L01+ module;
- 2x Accelerometer;
- 2x Barometric pressure sensor;
- 2x CAN transceiver.

DigiKey code	Manufacturer code	Manufacturer	Description	Number of pieces	Price €	Total price €
497-18610-ND	STM8AF5288TCY	STMicroelectronics	IC MCU 8BIT 64KB FLASH 48LQFP	2	3,02	6,04
296-15380-1-ND	SN65HVD231QDRG4Q1	Texas Instruments	IC TRANSCEIVER HALF 1/1 8SOIC	1	2,56	2,56
497-17054-1-ND	AIS1120SXTR	STMicroelectronics	ACCELEROMETER 120G SPI 8SOIC	1	5,71	5,71
490-17959-1-ND	CSTNE8M00G55A000R0	Murata Electronics	CERAMIC RES 8.0000MHZ 33PF SMD	2	0,28	0,56
445-3957-1-ND	ACT45B-220-2P-TL003	TDK corporation	CMC 22UH 200MA 2LN SMD AEC-Q200	1	2,04	2,04
SZNUP2105LT1GOSCT-ND	SZNUP2105LT1G	ON Semiconductor	TVS DIODE 24V 44V SOT23-3	1	0,36	0,36
495-2416-1-ND	B59606A0110A062	EPCOS	PTC RESET FUSE 30V 90MA 1210	2	1,09	2,18
MMSZ5247BT1GOSCT-ND	MMSZ5247BT1G	ON Semiconductor	DIODE ZENER 17V 500MW SOD123	2	0,18	0,36
LT8330ES6#TRMPBFCT-ND	LT8330ES6#TRMPBF	Analog Devices	IC REG BST SEPIC ADJ 1A TSOT23-6	2	5,79	11,58
828-1079-1-ND	BMP388	Bosch sensortec	SENSOR PRESSURE ABS	1	3,47	3,47
P100KDACT-ND	ERA-6AEB104V	Panasonic Electronic	RES 100K OHM 0.1% 1/8W 0805	2	0,33	0,66
P115KDACT-ND	ERA-6AEB1153V	Panasonic Electronic	RES 115K OHM 0.1% 1/8W 0805	2	0,33	0,66
490-5899-1-ND	GRM219R71E105KA88D	Murata Electronics	CAP CER 1UF 25V X7R 0805	2	0,24	0,48
732-2304-1-ND	744878100	Wurth Electronik	INDUCT ARRAY 2 COIL 10UH SMD	2	2,32	4,64
P576KCCT-ND	ERJ-6ENF5763V	Panasonic Electronic	RES 576K OHM 1% 1/8W 0805	2	0,09	0,18
RB550VM-30FHTE-17CT-ND	RB550VM-30FHTE-17CT-ND	Rohm Semiconductors	RB550VM-30FH IS LOW V F	2	0,34	0,68
P510KCCT-ND	ERJ-6ENF5103V	Panasonic Electronic	RES 510K OHM 1% 1/8W 0805	2	0,09	0,18
399-19474-1-ND	C0805X479C5GACAUTO	KEMET	CAP CER 4.7PF 50V C0G/NP0 0805	2	0,5	1
490-14361-1-ND	GCM21BR70J106KE22K	Murata Electronics	CAP CER 10UF 6.3V X7R 0805	2	0,38	0,76
P21326CT-ND	ERJ-PB6D4993V	Panasonic Electronic	RES SMD 499K OHM 0.5% 1/4W 0805	2	0,23	0,46
P536KCCT-ND	ERJ-6ENF5363V	Panasonic Electronic	RES SMD 536K OHM 1% 1/8W 0805	2	0,09	0,18
399-7999-1-ND	C0805C104K8RACTU	KEMET	CAP CER 0.1UF 10V X7R 0805	4	0,19	0,76
399-6931-1-ND	C0805C105K8RACAUTO	KEMET	CAP CER 1UF 10V X7R 0805	6	0,23	1,38
A129738CT-ND	CRGCQ0805F120R	TE connectivity	CRGCQ 0805 120R 1%	1	0,09	0,09
399-6918-1-ND	C0805C101K5GACAUTO	KEMET	CAP CER 100PF 50V C0G/NP0 0805	2	0,15	0,3
RNCF0805DTE10K0CT-ND	RNCF0805DTE10K0	Stackpole electronics	RES 10K OHM 0.5% 1/8W 0805	2	0,14	0,28
	BATTERY		LiPO Battery 3.7V 400mAh	1	0,01	0,01
	NRF24L01	Nordic Semiconductor	Wireless Module SMD	2	0,77	1,54
			Totale			49,1

Figure 19: Custom board BOM

4.11.4 Custom Board Reduced Size

The components used for the custom board with the goal of size reduction are listed below. What is considered in the total price, in order to fulfill the specifications, is:

- 2x Power supply (one for each of the two boards);
- 2x MCU;
- 2x NRF24L01+;
- 1x Accelerometer (to be mounted only on the sensor board);
- 1x Barometric pressure sensor (to be mounted only on the sensor board);
- 1x CAN transceiver.

4.11.5 Cost Resume

The total cost of the project is within the initial budget. With less than 100 € it was possible to realize both the prototype for the evaluation phase and the custom board. The

DigiKey code	Manufacturer code	Manufacturer	Description	# of pieces	Price €	Total Price €
497-18610-ND	STM8AF5288TCY	STMicroelectronics	IC MCU 8BIT 64KB FLASH 48LQFP	2	3,02	6,04
296-15380-1-ND	SN65HVD231QDRG4Q1	Texas Instruments	IC TRANSCEIVER HALF 1/1 8SOIC	2	2,56	5,12
497-17054-1-ND	AIS1120SXTR	STMicroelectronics	ACCELEROMETER 120G SPI 8SOIC	2	5,71	11,42
490-17959-1-ND	CSTNE8M00G55A000R0	Murata Electronics	CERAMIC RES 8.0000MHZ 33PF SMD	2	0,28	0,56
445-3957-1-ND	ACT45B-220-2P-TL003	TDK corporation	CMC 22UH 200MA 2LN SMD AEC-Q200	2	2,04	4,08
SZNUP2105LT1GOSCT-ND	SZNUP2105LT1G	ON Semiconductor	TVS DIODE 24V 44V SOT23-3	2	0,36	0,72
495-2416-1-ND	B59606A0110A062	EPCOS	PTC RESET FUSE 30V 90MA 1210	2	1,09	2,18
MMSZ5247BT1GOSCT-ND	MMSZ5247BT1G	ON Semiconductor	DIODE ZENER 17V 500MW SOD123	2	0,18	0,36
LT8330ES6#TRMPBFCT-ND	LT8330ES6#TRMPBF	Analog Devices	IC REG BST SEPIC ADJ 1A TSOT23-6	2	5,79	11,58
828-1079-1-ND	BMP388	Bosch sensortec	SENSOR PRESSURE ABS	2	3,47	6,94
P100KDACT-ND	ERA-6AEB104V	Panasonic Electronic	RES 100K OHM 0.1% 1/8W 0805	2	0,33	0,66
P115KDACT-ND	ERA-6AEB1153V	Panasonic Electronic	RES 115K OHM 0.1% 1/8W 0805	2	0,33	0,66
490-14532-1-ND	GCM21BR71C475KA73K	Murata Electronics	CAP CER 4.7UF 16V X7R 0805	6	0,4	2,4
490-5899-1-ND	GRM219R71E105KA88D	Murata Electronics	CAP CER 1UF 25V X7R 0805	2	0,24	0,48
1727-5218-1-ND	PMEG6030EP,115	NXP	DIODE SCHOTTKY 60V 3A SOD128	2	0,47	0,94
732-2304-1-ND	744878100	Wurth Electronik	INDUCT ARRAY 2 COIL 10UH SMD	2	2,32	4,64
P576KCCT-ND	ERJ-6ENF5763V	Panasonic Electronic	RES 576K OHM 1% 1/8W 0805	2	0,09	0,18
P510KCCT-ND	ERJ-6ENF5103V	Panasonic Electronic	RES 510K OHM 1% 1/8W 0805	2	0,09	0,18
399-19474-1-ND	C0805X479C5GACAUTO	KEMET	CAP CER 4.7PF 50V C0G/NP0 0805	2	0,5	1
490-14361-1-ND	GCM21BR70J106KE22K	Murata Electronics	CAP CER 10UF 6.3V X7R 0805	2	0,38	0,76
P21326CT-ND	ERJ-PB6D4993V	Panasonic Electronic	RES SMD 499K OHM 0.5% 1/4W 0805	2	0,23	0,46
P536KCCT-ND	ERJ-6ENF5363V	Panasonic Electronic	RES SMD 536K OHM 1% 1/8W 0805	2	0,09	0,18
399-7999-1-ND	C0805C104K8RACTU	KEMET	CAP CER 0.1UF 10V X7R 0805	4	0,19	0,76
399-6931-1-ND	C0805C105K8RACAUTO	KEMET	CAP CER 1UF 10V X7R 0805	2	0,23	0,46
A129738CT-ND	CRGCQ0805F120R	TE connectivity	CRGCQ 0805 120R 1%	2	0,09	0,18
399-6918-1-ND	C0805C101K5GACAUTO	KEMET	CAP CER 100PF 50V C0G/NP0 0805	2	0,15	0,3
RNCF0805DTE10K0CT-ND	RNCF0805DTE10K0	Stackpole electronics	RES 10K OHM 0.5% 1/8W 0805	2	0,14	0,28
	NRF24L01	Nordic Semiconductor	Wireless Module	2	0,77	1,54
					Totale	65,06

Figure 20: Custom board BOM - considering sensors on both sides

cost also includes the production of the custom board. Since the production has been done in China, the cost of production is very low (about 1.8 €).

Total price = Evaluation + Custom + Production cost = 44.92 € + 49.10 € + 2*(0.48) €= **€ 94.98**

4.11.6 Conclusion

Goals achieved from a cost point of view:

- Total cost within the budget;
- Overall cost within the budget also for the smaller board;
- Reduction with respect to the cost assessed in the first estimation phase for the evaluation board;
- Reduction with respect to the overall estimated costs.

Goals not achieved:

- Cost reduction in the custom board with respect to the evaluation version;

DigiKey code	Manufacturer code	Manufacturer	Description	# of pieces	Price €	Total Price €
497-19213-ND	STM8AF5286UDY	STMicroelectronics	IC MCU 8BIT 64KB FLASH 32VFQFPN	2	2,89	5,78
497-16268-1-ND	AIS3624DQTR	STMicroelectronics	ACCEL 6-24G I2C/SPI 24QFN	1	9,52	9,52
490-17959-1-ND	CSTNE8M00G55A000R0	Murata Electronics	CERAMIC RES 8.0000MHZ 33PF SMD	2	0,28	0,56
732-5621-1-ND	7442335600	Wurth Electronik	CMC 600MA 2LN 60 OHM SMD	2	1,54	3,08
495-2416-1-ND	B59606A0110A062	EPCOS	PTC RESET FUSE 30V 90MA 1210	2	1,09	2,18
MMSZ5247BT1GOSCT-ND	MMSZ5247BT1G	ON Semiconductor	DIODE ZENER 17V 500MW SOD123	2	0,18	0,36
NCV7342MW3R2GOSCT-ND	NCV7342MW3R2G	ON Semiconductor	IC TRANSCEIVER HALF 1/1 8SOIC	1	1,03	1,03
828-1079-1-ND	BMP388	Bosch sensortec	SENSOR PRESSURE ABS	1	3,47	3,47
LT8330EDDB#TRMPBFCT-ND	LT8330EDDB#TRMPBF	Analog Devices	IC REG BST SEPIC ADJ 1A 8DFN	1	5,79	5,79
SZNUP2105LT1GOSCT-ND	SZNUP2105LT1G	ON Semiconductor	TVS DIODE 24V 44V SOT23-3	2	0,36	0,72
732-11688-1-ND	7448845100	Wurth Electronik	INDUCTOR ARRAY 2 COIL 10UH SMD	2	2,13	4,26
A129616CT-ND	CRGCQ0402F120R	TE Connectivity	CRGCQ 0402 120R 1%	1	0,09	0,09
A129616CT-ND	CRGCQ0402F120R	TE Connectivity	CRGCQ 0402 120R 1%	2	0,09	0,18
P499KLCT-ND	ERJ-2RKF4993X	Panasonic	RES SMD 499K OHM 1% 1/10W 0402	2	0,09	0,18
RMCF0402JT510KCT-ND	RMCF0402JT510K	Stackpole Electronics	RES 510K OHM 5% 1/16W 0402	2	0,09	0,18
RMCF0402FT576KCT-ND	RMCF0402FT576K	Stackpole Electronics	RES 576K OHM 1% 1/16W 0402	2	0,09	0,18
1727-5218-1-ND	PMEG6030EP,115	NXP	DIODE SCHOTTKY 60V 3A SOD128	2	0,47	0,94
490-16434-1-ND	GCM155R71A104KA55D	Murata Electronics	CAP CER 0.1UF 10V X7R 0402	4	0,09	0,36
399-17545-1-ND	C0402C101J5RACAUTO	KEMET	CAP CER SMD 0402 100PF 5% X7R 50	2	0,1	0,2
490-18216-1-ND	GRT155R70J105KE01D	Murata Electronics	CAP CERAMIC 1UF 6.3V X7R 10% PAD	2	0,13	0,26
490-14361-1-ND	GCM21BR70J106KE22K	Murata Electronics	CAP CER 10UF 6.3V X7R 0805	2	0,38	0,76
1490-1033-1-ND	NRF24L01P-R7	Nordic Semiconductors	IC RF TXRX ISM>1GHZ 20VFQFN	2	3,21	6,42
P10KDECT-ND	ERA-2AED103X	Panasonic	RES SMD 10K OHM 0.5% 1/16W 0402	2	0,2	0,4
490-15586-2-ND	LQW15AN7N9J8ZD	Murata Electronics	FIXED IND 7.9NH 1.7A 50 MOHM	2	0,09	0,18
311-1.00MLRTR-ND	RC0402FR-071ML	Yageo	RES SMD 1M OHM 1% 1/16W 0402	2	0,09	0,18
490-6767-2-ND	LQW15AN12NG00D	Murata Electronics	FIXED IND 12NH 500MA 140 MOHM	2	0,09	0,18
311-22KGRCT-ND	RC0603JR-0722KL	Yageo	RES SMD 22K OHM 5% 1/10W 0402	2	0,09	0,18
311-1017-1-ND	CC0402JRNPO9BN150	Yageo	CAP CER 15PF 50V C0G/NPO 0402	2	0,09	0,18
311-1378-1-ND	CC0402KRX7R7BB223	Yageo	CAP CER 0.022UF 16V X7R 0402	2	0,09	0,18
311-1367-1-ND	CC0402CRNPO9BN4R7	Yageo	CAP CER 4.7PF 50V C0G/NPO 0402	2	0,09	0,18
311-1042-1-ND	CC0402KRX7R7BB103	Yageo	CAP CER 10000PF 16V X7R 0402	2	0,09	0,18
311-1018-1-ND	CC0402JRNPO9BN220	Yageo	CAP CER 22PF 50V C0G/NPO 0402	4	0,09	0,36
311-1003-1-ND	CC0402CRNPO9BN1R5	Yageo	CAP CER 1.5PF 50V C0G/NPO 0402	2	0,09	0,18
311-1410-1-ND	CC0201KRX5R5BB333	Yageo	CAP CER 0.033UF 6.3V X5R 0402	2	0,09	0,18
490-12967-2-ND	LQP03TN2N3B02D	Murata Electronics	FIXED IND 2.3NH 500MA 200 MOHM	2	0,09	0,18
	BATTERY		LiPO Battery 3.7V 400mAh	1	0,99	0,99
					Totale	50,23

Figure 21: Custom board reduced size BOM

- Reduction with respect to the cost assessed in the first estimation phase for the custom board.

	Estimated	**Final cost**	**Cost reduction**
Evaluation board	50 €	44.92 €	- 5.08 €
Custom board	45 €	49.36 €	+ 4.36 €
Custom board sensor x2	-	65.06 €	-
Custom board reduced size	45 €	50.99 €	+ 5.99 €

Table 12: Cost evaluation

4.12 Overall Expenses

The resume of all the effective expenses is shown in Figure 22.

Costs Report CL1

				EXPENSE REPORT TOTAL	**97,60 €**
Name:	Group 4	Purpose:	CL1 project cost report	**Evaluation board cost**	**44,92 €**
Dept:	Sales	Start Date:	08/10/2019	**CL1 system cost**	**49,10 €**
		End Date:	19/12/2019	**CL1 cost/complete board**	**24,55 €**
				EVAL + CUSTOM TOTAL	**94,02 €**

Date	Account	Description	Distributor	Quantity	Unit Price	Total €	Notes
24/10/2019	Components Purchasing	Components for evaluation and custom board	Digikey	1	91,80 €	91,80 €	Additional components purchased as back-up
30/10/2019	Battery cell purchasing	Battery for sensor board supply	AliExpress	1	0,01 €	0,01 €	
11/11/2019	PCB Production	Custom board production	JLCPBC	10	0,48 €	4,80 €	
22/11/2019	Battery cell purchasing	Battery for sensor board supply	Ebay	1	0,99 €	0,99 €	First order from China not delivered - back up solution

Figure 22: Cost report

5 Hardware Design

5.1 Design Objectives and Specification

The main objectives related to the hardware design are:

- Create a single modular design that can be used both as control and as sensor board;
- Make the system work properly;
- Size reduction with respect to the evaluation prototype;
- Compact design.

5.2 Evaluation Set Up

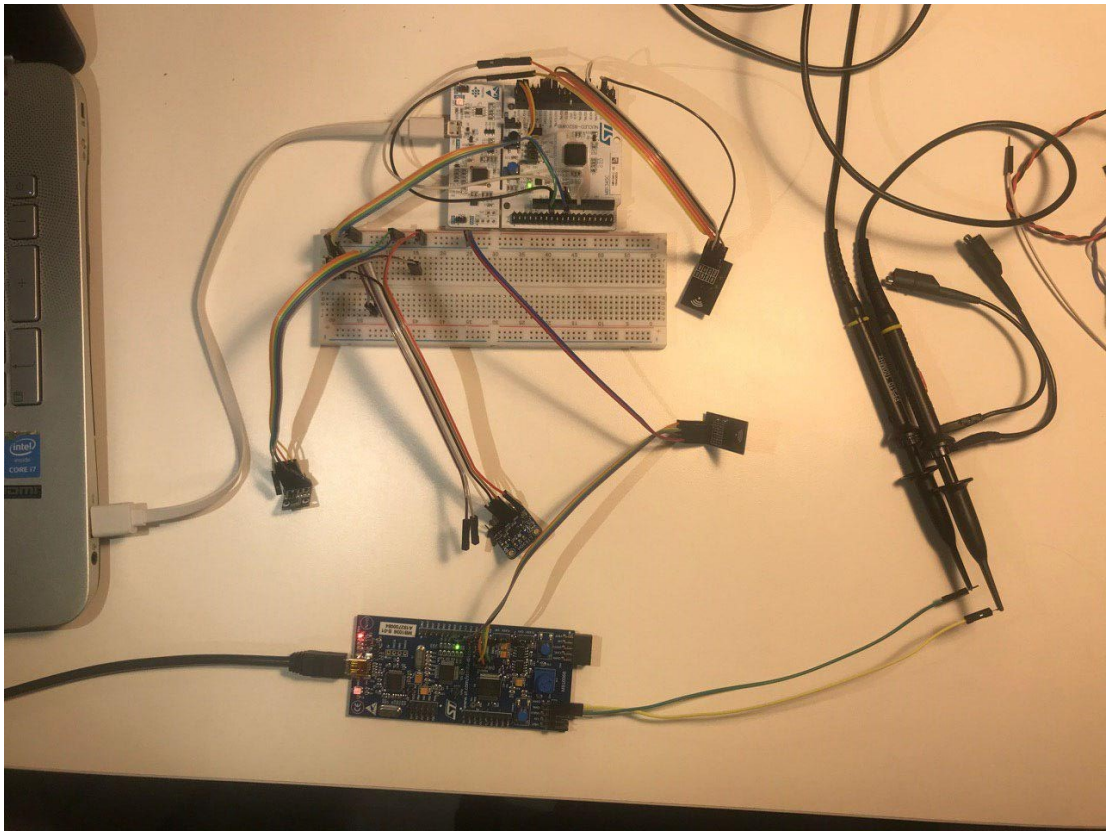


Figure 23: Evaluation set-up

5.3 Design Steps

The component choice has been described in the previous section. The components and the design of the circuit are considered in the following paragraph.

5.4 MCU

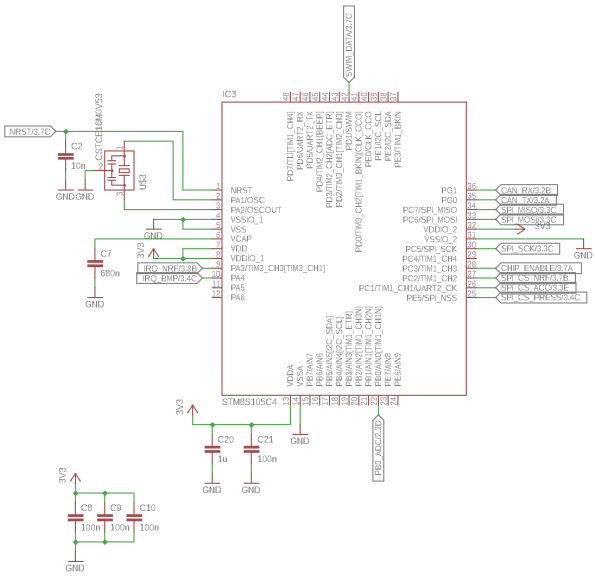


Figure 24: MCU circuit

The configuration of the MCU pins is done with STM8 Cube MX, that allows the proper configuration of STM7/STM8 MCUs and the generation of the code to initialize peripherals. In the following table all the functions related to the used pins are listed.

- The connection with the sensors is establish via SPI, the sensor is selected using a MCU pin configured as *Chip Select*. The block scheme shows the way in which the sensors are connected.
- Cube MX also allows to set the frequency of the external clock, as the figure shows. The custom board is equipped with a 16 MHz external oscillator.
  According to the STM8AF5288T datasheet, the crystal characteristics have been checked with the following formula:

$$g_m >> g_{crit}$$
$$g_{crit} = (2\pi f_{HSE})^2 R_m (2C_0 + C)^2 = (2\pi \ \ 16 \ \ 10^6)^2 (40)(2C_0 + 15 \ \ 10^{-12})^2 << 5\frac{mA}{V}$$

  The maximum allowable value for built-in load capacitance is 20 pF.
- As stated in the MCU datasheet, all the supply pins have one or more decoupling capacitors. the capacitor values and the MCU pins to which they are connected are shown in the following table. The capacitor number refers to the number in the Eagle schematic.

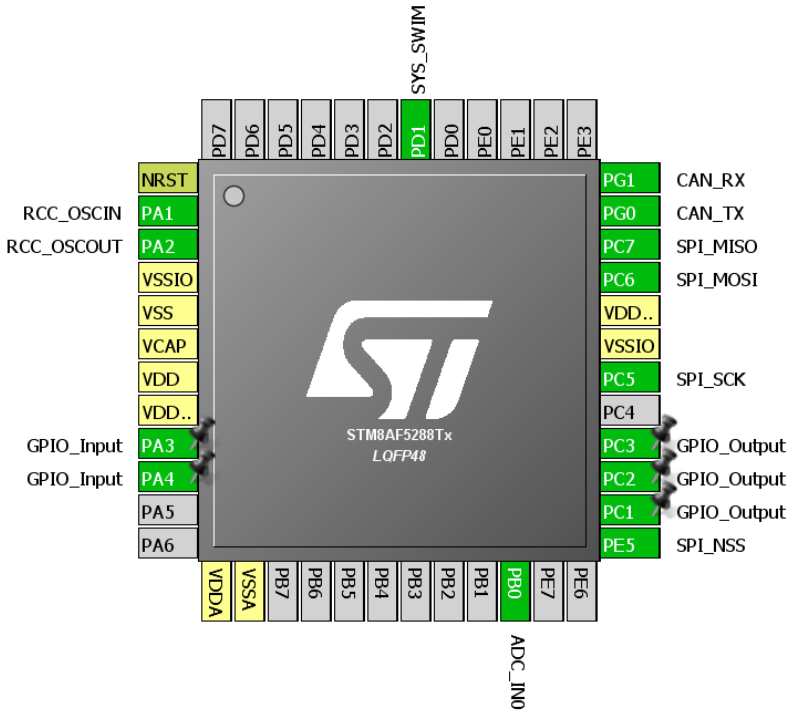


Figure 25: Pin configuration through Cube MX

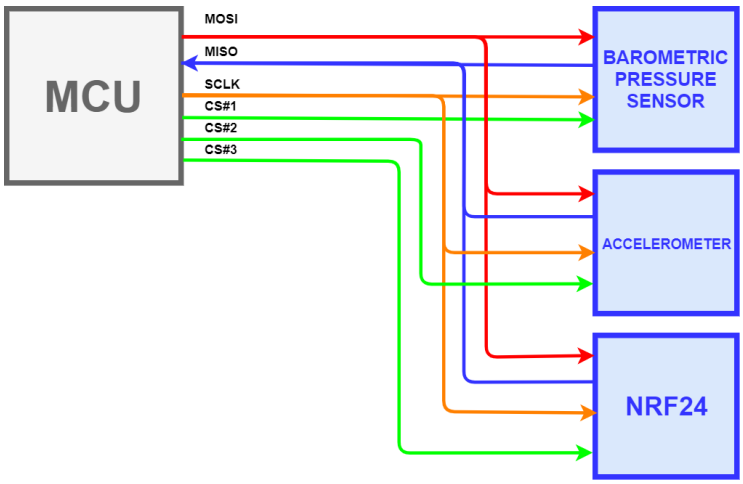


Figure 26: MCU - Sensors connection

- 4 headers have been inserted to program the MCU. They will be soldered for programming and then desoldered in order to reduce the space occupation.

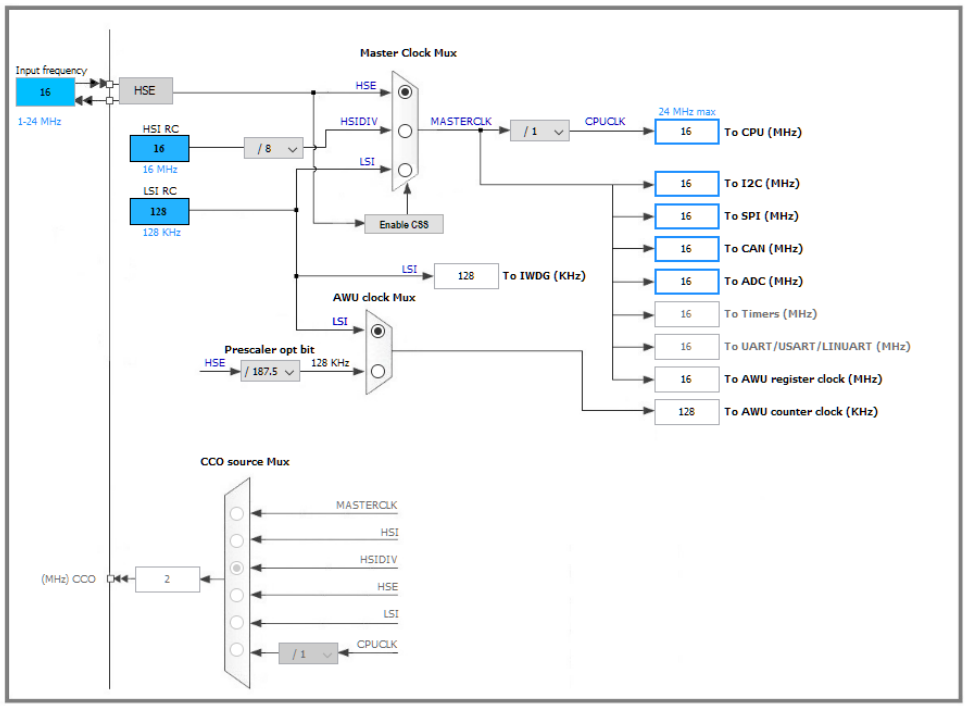


Figure 27: Clock through Cube MX

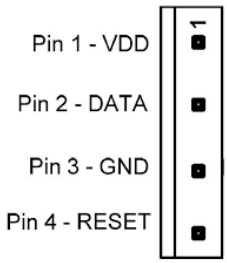


Figure 28: SWIM pinout

Pin Number	Pin Name	Activated as	Description
1	NRST	MCU reset	Connected to the programming interface
2	PA1	*RCC_OSCIN*	External oscillator input
3	PA2	*RCC_OSCOUT*	External oscillator output
4	VSSIO	Digital ground	Connected to GND
5	VSS	Ground	Connected to GND
6	VCAP		
7	VDD	Supply Voltage	Connected to 3.3V
8	VDDIO	Digital supply voltage	Connected to 3.3V
9	PA3	*GPIO_Input*	External interrupt NRF24L01+
10	PA4	*GPIO_Input*	External interrupt BMP388
13	VDDA	Analog supply voltage	Connected to 3.3V
14	VSSA	Analog ground	Connected to GND
22	PB0	*ADC_IN*0	ADC channel for battery voltage monitoring
25	PE5	*SPI_NSS*	Default SPI chip select - BMP388
26	PC1	*GPIO_Output*	SPI chip select - AIS2120SX
27	PC2	*GPIO_Output*	SPI chip select - NRF24L01+
28	PC3	*GPIO_Output*	Chip enable NRF24L01+
30	PC5	*SPI_SCK*	SPI Clock
31	VSSIO	Digital ground	Connected to GND
32	VDDIO	Digital supply voltage	Connected to 3.3V
33	PC6	*SPI_MOSI*	SPI Master Out Slave In
34	PC7	*SPI_MISO*	SPI Master In Slave Out
35	PG0	*CAN_TX*	CAN transmitter
36	PG1	*CAN_RX*	CAN receiver
42	PD1	*SYS_SWIM*	SWIM Data for programming

Table 13: List of the used MCU pins

	CSTCE16M0V53
Manufacturer	Murata Electronics
Frequency	16 MHz
Built-in Load Capacitance	15 pF
Temperature Range	-40 °C - +125 °C
AEC-Q100	Yes
Dimensions	3.2x1.3mm

Table 14: CSTCE16M0V53 characteristics

Component Number	**Pin Name**	**Capacitor Value**
C2	NRST	10 nF
C7	VCAP	680 nF
C8	VDD	100 nF
C9	VDD	100 nF
C10	VDD	100 nF
C20	VDDA	1 uF
C21	VDDA	100 nF

Table 15: List of capacitors

5.5 Sensors

The connection of the sensors requires capacitors on the supply line, as follows.

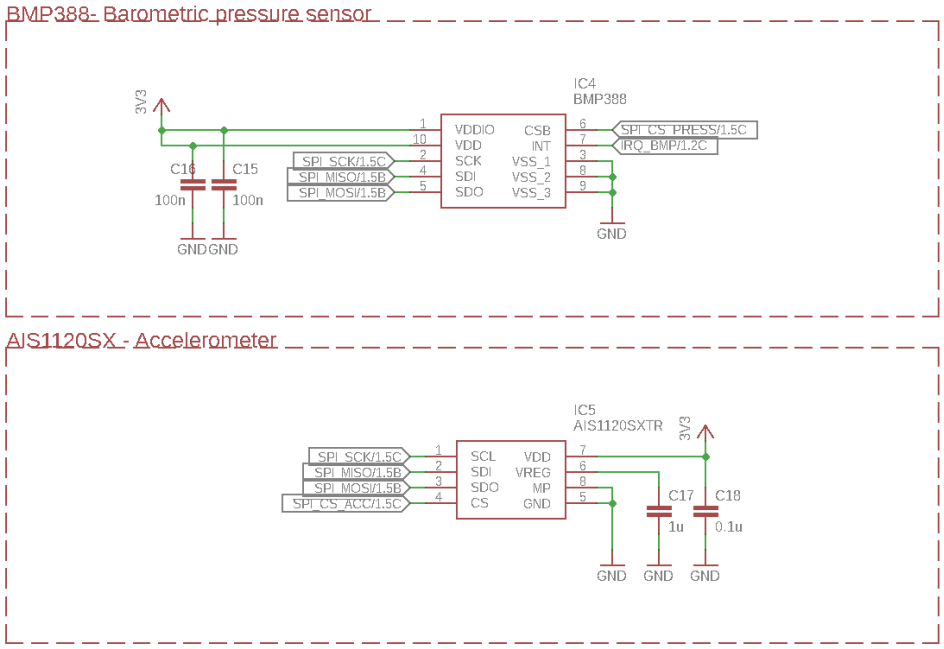


Figure 29: Sensors circuit

Component Number	Pin Name	Capacitor Value	Sensor
C15	VDD	100 nF	BMP388
C16	VDDIO	100 nF	BMP388
C17	VREG	1 uF	AIS2120SX
C19	VDD	100 nF	AIS2120SX

Table 16: List of capacitors

5.6 CAN

In order to translate messages from CAN_TX/CAN_RX MCU pins to CAN_H/CAN_L lines a CAN transceiver is needed. The chosen one is the Texas Instruments SN65HVD231Q-Q1 CAN transceiver. In the following, the CAN line circuit and all its components are shown. In order to increase reliability by reducing the effect of electromagnetic inter-

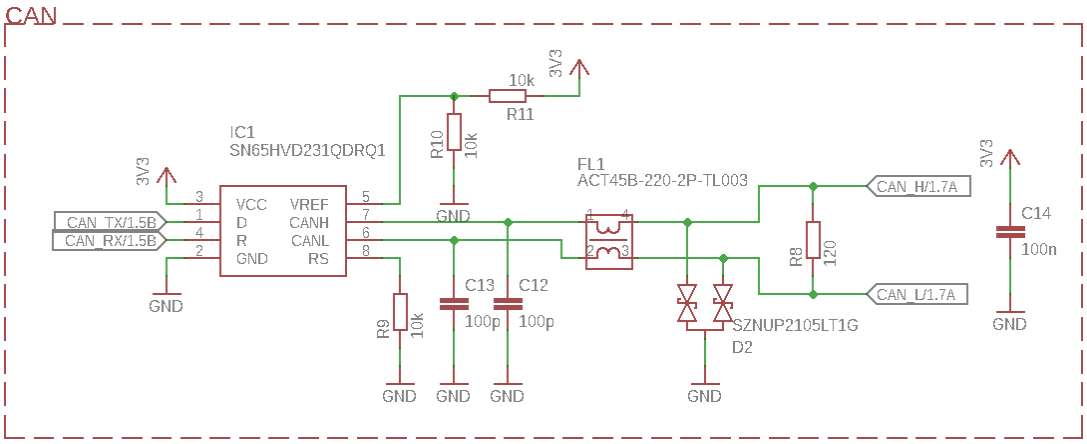


Figure 30: CAN circuit

Component Number	Value	Description
C12	100 pF	
C13	100 pF	
R8	120 kΩ	CAN bus termination
R9	10 kΩ	Used for CAN transceiver slope adjustment
R10	10 kΩ	Together with R11 it implements a voltage divider in order to obtain VDD/2 on the VREF pin
R11	10 kΩ	
D2	-	TVS Diode
FL1	-	Common Mode Choke
IC1	-	CAN transceiver

Table 17: List of CAN components

ference and electrostatic discharge noise, the CAN bus has a protection circuit composed by:

- Common mode choke, which is used to attenuate the common mode noise that affects both the transceiver bus lines;

- TVS Diode, which is used for electrostatic discharge protection and to limit the voltage levels on the line (the limit depends on the rating of the TVS device).

5.7 Power Converter

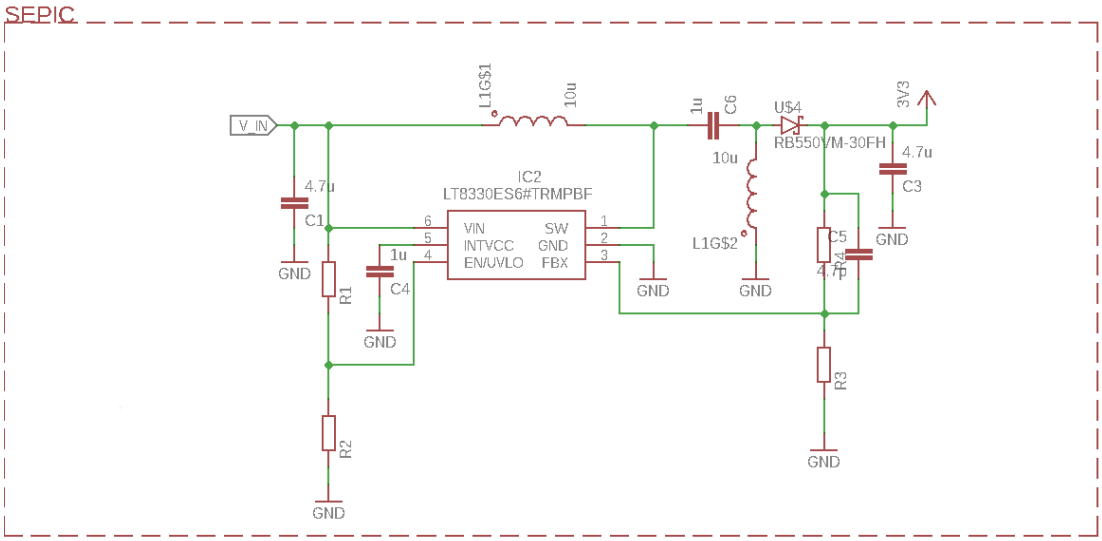


Figure 31: SEPIC converter

As described in the previous section, a SEPIC converter has been used to regulate the supply voltage. The design of its part of the circuit was by far the most complex. The following steps have been followed for the design phase:

- Calculation of the values of the components according to the LT8330 datasheet;
- Choice of the proper components taking into account what is suggested by the datasheet;
- Simulation of the behaviour of the converter;
- Modifications in the components' choice, based on the simulation.

LT8330 Components

Set as $V_{IN(MIN)} = 3V$ and $V_{IN(MAX)} = 12V$, with $V_D = 0.59V$ (forward voltage of the RB550VM-30FHTE Schottky diode according to the datasheet).

$$D_{max} = \frac{V_{OUT}+V_D}{V_{IN(MIN)}+V_{OUT}+V_D} = \frac{3.3+0.59}{3+3.3+0.59} = 0.5645$$
$$D_{min} = \frac{V_{OUT}+V_D}{V_{IN(MAX)}+V_{OUT}+V_D} = \frac{3.3+0.59}{12+3.3+0.59} = 0.2448$$

The obtained values are the maximum and minimum duty cycles. The conditions on the duty cycle that have to be satisfied are the following:

$$D_{max} < 1-(Minimum\,Off-Time_{MAX} \quad f_{OSC(MAX)}) = 1-(65\ \ 10^{-9})(2.15\ \ 10^{6}) = 0.86025$$
$$D_{min} > (Minimum\,On-Time_{MAX} \quad f_{OSC(MAX)}) = (105\ \ 10^{-9})(2.15\ \ 10^{6}) = 0.22575$$
$$D_{MAX} = 0.5645 < 0.86025 \text{ and } D_{MIN} = 0.2448 > 0.22575$$

Both the conditions on the duty cycle are satisfied.
The values for $MinimumOff-Time_{MAX} = 65ns$, $MinimumOn-Time_{MAX} = 105ns$ and $f_{OSC(MAX)} = 2.15MHz$ are specified in the LT8330 datasheet.

The inductor selection is done based on the maximum output current capability.

$$I_{L1(MAX)(AVE)} = I_{IN(MAX)(AVE)} = I_{OUT(MAX)}\frac{D_{MAX}}{1-D_{MAX}} = (0.180)\frac{0.5645}{1-0.5645} = 0.2333\ A$$
$$I_{L2(MAX)(AVE)} = I_{OUT(MAX)} = 0.180\ A$$
$$I_{SW(MAX)(AVE)} = I_{L1(MAX)(AVE)} + I_{L2(MAX)(AVE)} = 0.4133\ A$$
$$I_{SW(PEAK)} = (1+\tfrac{\chi}{2})(I_{OUT(MAX)})(\tfrac{1}{1-D_{MAX}}) = (1-0.2)(0.180)\tfrac{1}{1-0.5645} = 0.3307\ A$$

The value of 180 mA has been chosen for $I_{OUT(MAX)}$. This was done taking into account the worst case maximum current consumption, which refers to a scenario in which the MCU is driving all its I/O pins at the same time. This is not the case for the specific application of the board, but this design choice provides robustness even in the case in which other functionalities have to be implemented on the board. Thus, expansions and upgrades of the system do not require to redesign the power delivery circuitry from scratch, because this strategy allows complete reuse in different scenarios.
It is recommended that χ falls between 0.2 to 0.6. The choice of smaller values of ΔI_L) requires large inductances. Accepting larger values of ΔI_L) allows the use of low inductances, but results in higher input current ripple and core losses.

$$\Delta I_{SW} = \chi\, I_{SW(MAX)(AVE)} = (0.4)(0.3307) = 0.1323\ A$$
$$\Delta I_{L1} = \Delta I_{L2} = 0.5\Delta I_{SW} = 0.0661\ A$$

Due to the current limit of its internal power switch, the LT8330 should be used in a SEPIC converter whose maximum output current ($I_{O(MAX)}$) is less than the output current capability by a sufficient margin (10% or higher is recommended).

$$I_{O(MAX)} = (1-D_{MAX})(1-0.5(\Delta I_{SW}))(0.9) = (1-0.5645)(1-0.5(0.1323))(0.9) = 0.366\ A$$
$$L1 = L2 = \frac{V_{IN(MIN)}}{(0.5)(\Delta I_{SW})(f_{OSC})}D_{MAX} = \frac{3}{(0.5)(0.1323)(2\ \ 10^{6})}(0.5645) = 15.6\ uH$$

As a result of the calculation The values of R3 and R4 are calculated as follows

$$R4 = R3(\tfrac{V_{OUT}}{1.6} - 1) = (1.0625)R3 \longrightarrow R3 = 499\ k\Omega \text{ and } R4 = 576\ k\Omega$$

The values are chosen according to the resistors available on the market.
In the simulation section all the simulations are described.

Component Number	Value	Description
C1	4.7 uF	Input capacitor
C3	4.7 uF	Output capacitor
C4	1 uF	
C5	4.7 pF	
R1	kΩ	Voltage divider to select when the SEPIC has to turn-off
R2	kΩ	Voltage divider to select when the SEPIC has to turn-off
R3	499 kΩ	Voltage divider to select the output voltage
R4	576 kΩ	Voltage divider to select the output voltage
U4	-	Schottky diode
L1	10 uH	744878100 Wurth Electronics, advised by the datasheet
L2	-	
IC2	-	SEPIC controller LT8330

Table 18: List of SEPIC components

5.8 Connectors

As external connectors there are

- 4x header for MCU programming;
- 4x header for supply and CAN interface.

The reasons why a board mounted connector is not present are:

- Automotive compliant connectors, suitable for the application, are not present on the supplier's website. Automotive compliance is an important characteristic for the connector because it needs to have high resistance to stress and vibrations;
- Lack of time before the order of the components did not give the opportunity to evaluate all possible solutions.
  After some weeks a connector has been found on another supplier's website.

5.9 SEPIC Converter Simulations

In order to make sure that the SEPIC converter was being designed properly, simulations have been performed. The aims of the LTSpice simulations are

- Verify that the SEPIC converter behaves as expected: verify the proper switch on and off transitions.
- Evaluate which is the best choice for the diode;
- Verify behaviour considering overvoltage condition (with respect to the voltage level for which the SEPIC has been designed).

In order to set up the simulation the following components have been used:

- Resistors as calculated in the previous section;
- Inductors as calculated in the previous section;
- Capacitors as recommended by the LT8330 datasheet;
- 2 different diodes have been considered.

Diode selection
The main reasons for which the RB550VM-30 has been chosen are:

	PMEG6030EP	**RB550VM-30**
Manufacturer	NXP	Rohm Semiconductor
Typ. forward voltage	0.46 V	0.59 V
Average forward current	3 A	0.5 A
Reverse voltage	60 V	30 V
Typ. reverse current	80 μA	35 μA
Package	SOD128	SOD323-FL
Notes	Recommended by the datasheet	Selected based on forward current and voltage

Table 19: Diodes comparison

- PMEG6030EP has a forward current that is too high with respect to what is needed by the application;
- SOD323-Fl is considerably smaller than SOD128. This leads to space saving;

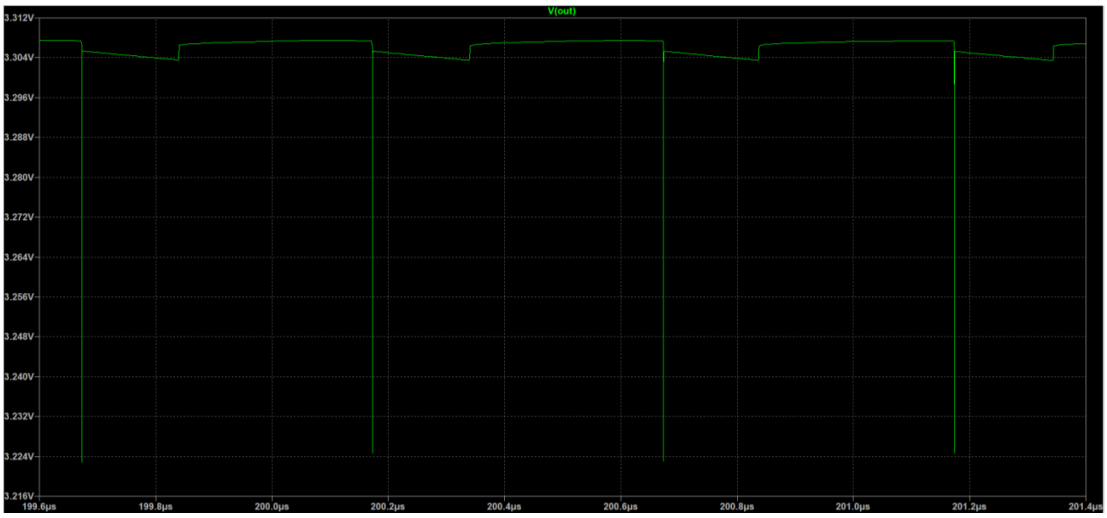


Figure 32: Ripple on output voltage with PMEG6030EP

- As it is shown in the following figures, the behaviour of RB550VM-30 is considerably improved with respect to PMEG6030EP.
  In Figure 32 the output voltage behaviour with the PMEG6030P Schottky diode is displayed, showing a 90 mV drop. The presence of this kind of ripple can cause noise on the supply line.
  In Figure 33, the output current using PMEG6030P is considered. The current is

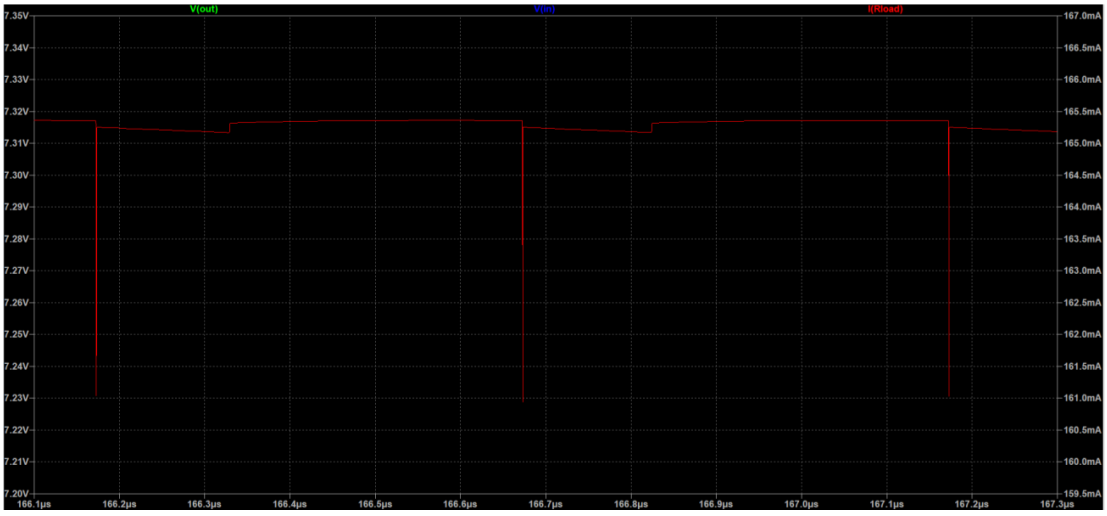


Figure 33: Ripple on output current with PMEG6030EP

almost flat, with just a 4 mA ripple.
In Figure 34 the output voltage behaviour with the RB550VM-30 Schottky diode is displayed, showing a 15 mV drop. There is a ripple reduction of about 6 times with respect to the previous case. In Figure 35, the output current using RB550VM-30 is

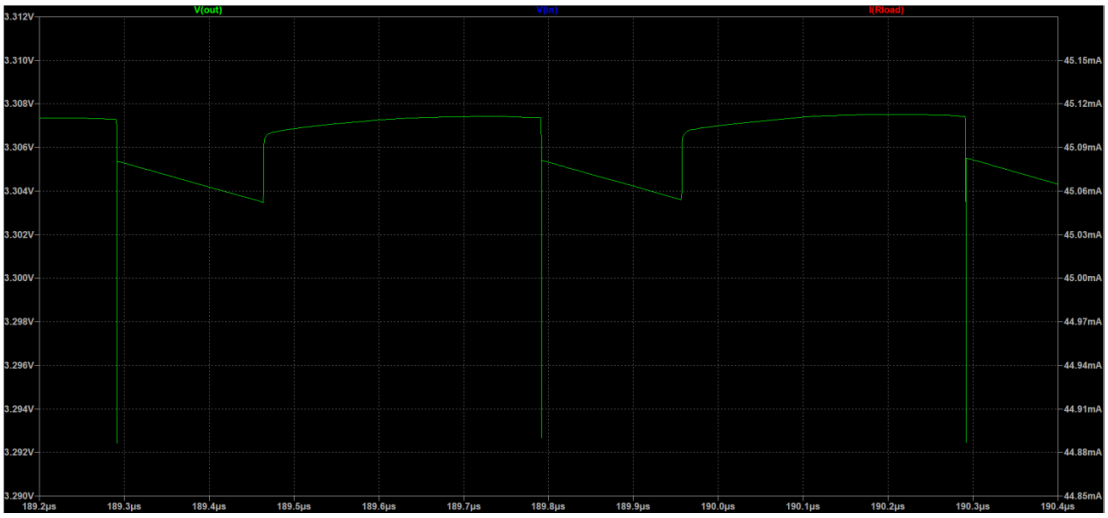


Figure 34: Ripple on output voltage with RB550VM-30

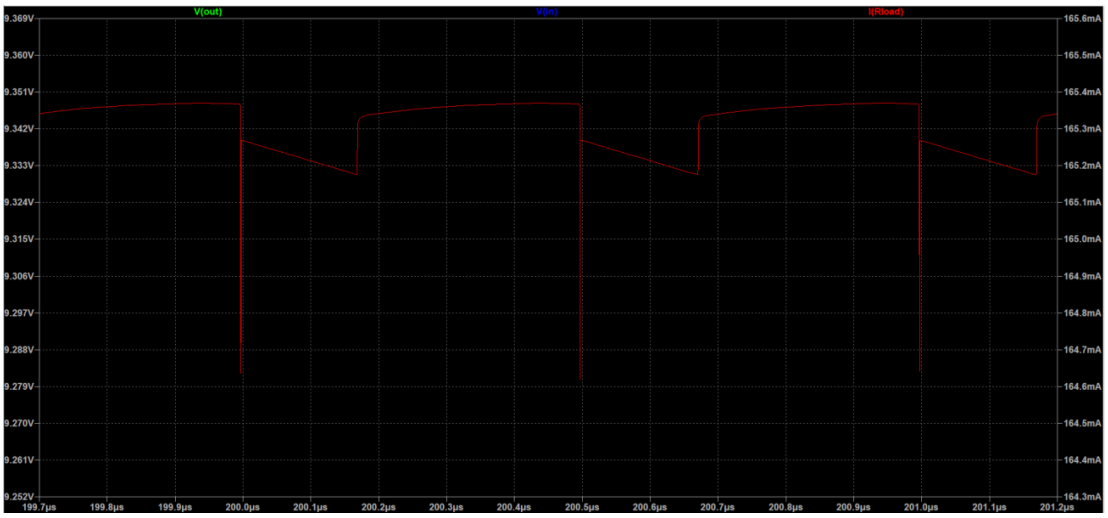


Figure 35: Ripple on output current with RB550VM-30

considered. The current has a 0.8 mA ripple. There is a ripple reduction of about 5 times with respect to the previous case.
Due to the improved behaviour with the RB550VM-30 and to the fact that it perfectly satisfies the requirements for the realization of the SEPIC, this component has been considered the most appropriate choice.

5.9.1 Rload estimation

Component	Power Mode	Imax [mA]
STMAF5288TCY - MCU	Run - 16 MHz - FLASH MEM	14
BMP388 - Pressure Sensor	Forced Mode - High Resolution @50Hz	0,975
AIS1120SXTR - 1 Axis Accelerometer	Not Available	6
NRF24L01 - Wireless Module	RX 1Mbps	13,1
SN65HVD231QDRG4Q1 - CAN Transceiver	RECEIVER - All Devices Mode	17
	TOTAL	51,075

Figure 36: Max current consumption RX mode

Rload has been selected considering that the maximum component current consumption. $R_{MCU} = \frac{V_{DD}}{I_{supply(MAX)}} = \frac{3.3\ V}{0.014\ A} = 235.7\ \Omega$

$$R_{NRF} = \frac{V_{DD}}{I_{supply(MAX)}} = \frac{3.3\ V}{0.0135\ A} = 244.4\ \Omega \text{ (RX mode)}$$

$$R_{Acc} = \frac{V_{DD}}{I_{supply(MAX)}} = \frac{3.3\ V}{0.006\ A} = 550\ \Omega$$

$$R_{Pres} = \frac{V_{DD}}{I_{supply(MAX)}} = \frac{3.3\ V}{0.000975\ A} = 3384.6\ \Omega$$

$$R_{CAN} = \frac{V_{DD}}{I_{supply(MAX)}} = \frac{3.3\ V}{0.017\ A} = 194.1\ \Omega$$

$$\frac{1}{R_{LOAD}} = \frac{1}{R_{MCU}} + \frac{1}{R_{NRF}} + \frac{1}{R_{Acc}} + \frac{1}{R_{Pres}} + \frac{1}{R_{CAN}} = \frac{1}{235.7} + \frac{1}{244.4} + \frac{1}{550} + \frac{1}{3384.6} + \frac{1}{194.1} = 0.0156 \rightarrow$$

$$R_{LOAD} = 64.1\ \Omega$$

5.9.2 Simulation Set-Up

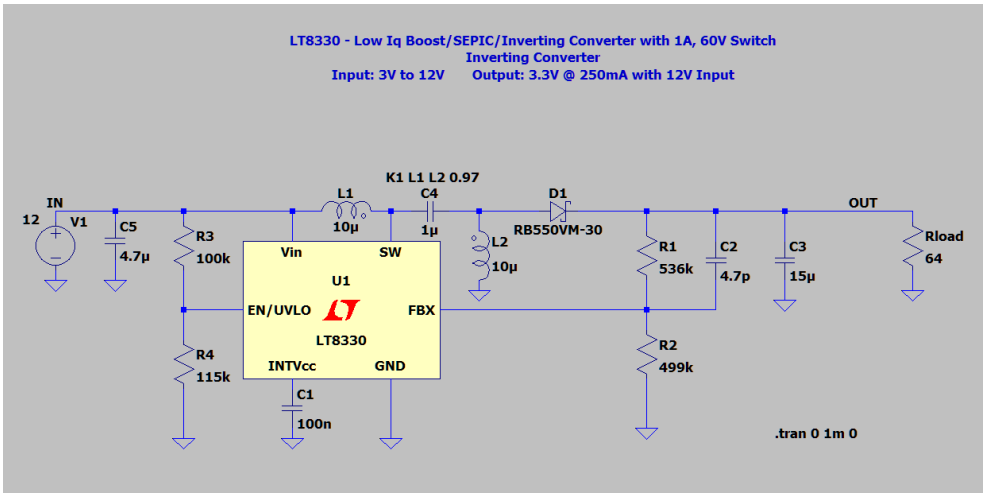


Figure 37: LTSpice SEPIC converter circuit

5.9.3 Constant Input Voltage V_IN = 12 V

Simulation of the SEPIC behaviour when the system is supplied by the vehicle battery (controller side).

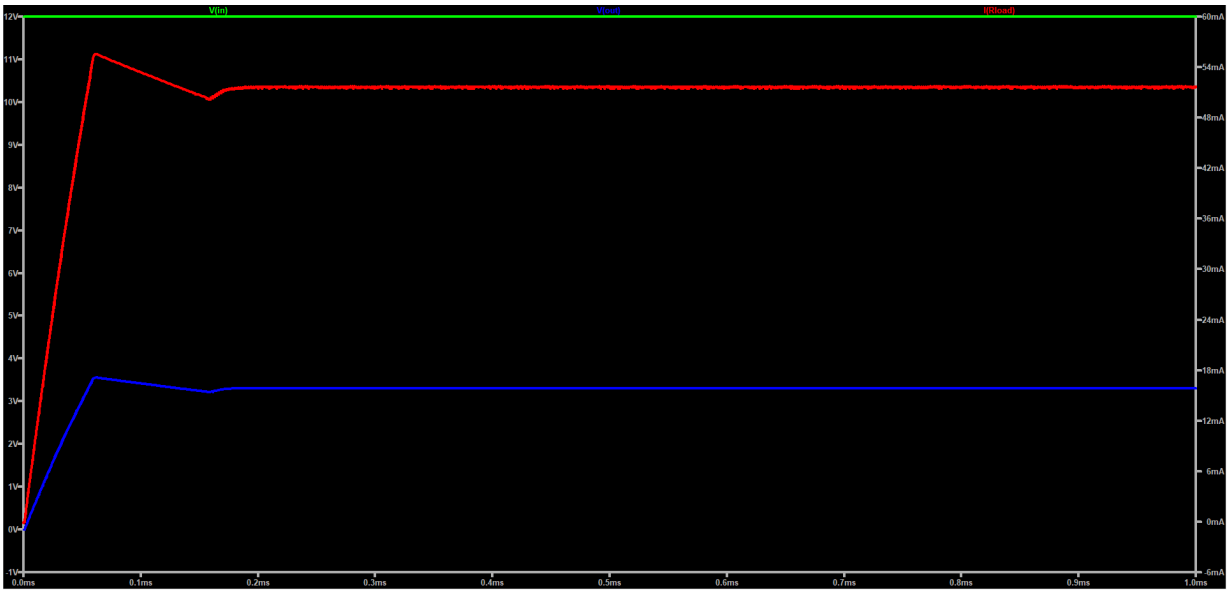


Figure 38: V_IN = 12 V

5.9.4 Battery Discharge - from V_IN = 3.7 V at t = 0 s to V_IN = 3 V

Simulation of the SEPIC behaviour when the system is supplied by the LiPo battery cell (sensor side).

When the cell voltage is equal to 3.09 V the SEPIC converter is switched off. In order to obtain this behaviour the voltage on the EN pin of LT8330 has to be lower than 1.6 V. In Figure 40 is shown using cursors the input voltage level at which the SEPIC converter is turned off.

$$V_{EN} = 1.6 = 3\frac{R4}{R3+R4}$$

5.9.5 Constant Overvoltage Input V_IN = 14 V

Simulation of the SEPIC behaviour when the system is supplied by the vehicle battery (controller side). In case of a voltage that is slightly higher that the rated one, SEPIC converter maintain the correct behaviour.

5.9.6 Turning off and on

It has been also analized the condition in which the the SEPIC converter switch completely off and then turns on. This kind of situation is not expected to occur during the operation

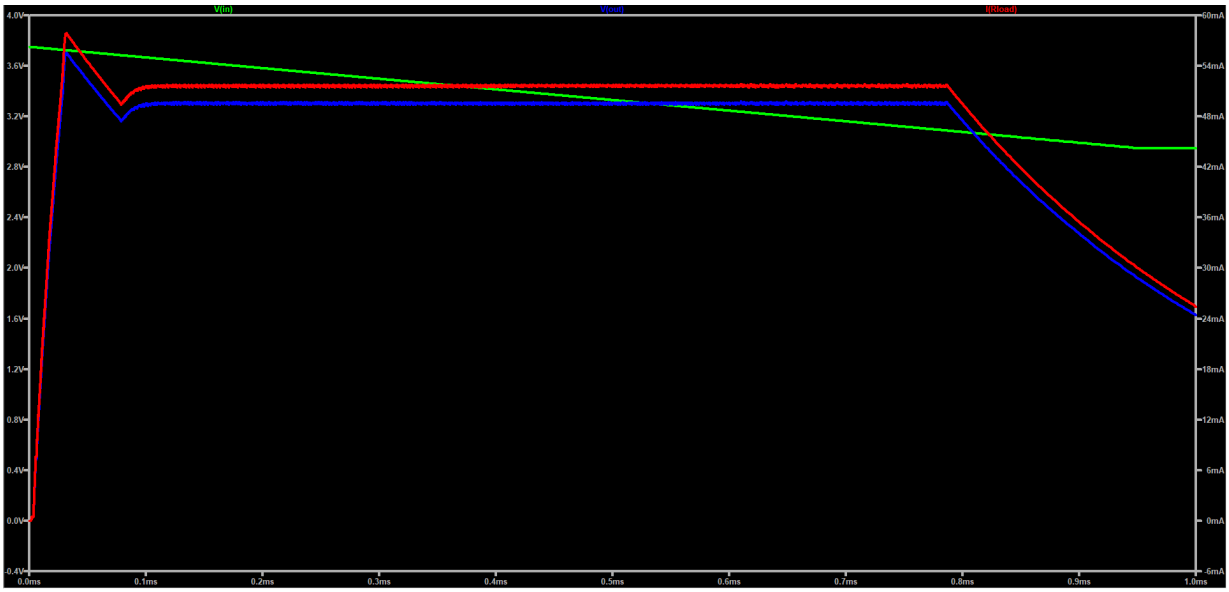


Figure 39: Battery discharge - from V_IN = 3.7 V at t = 0 s to V_IN = 3 V

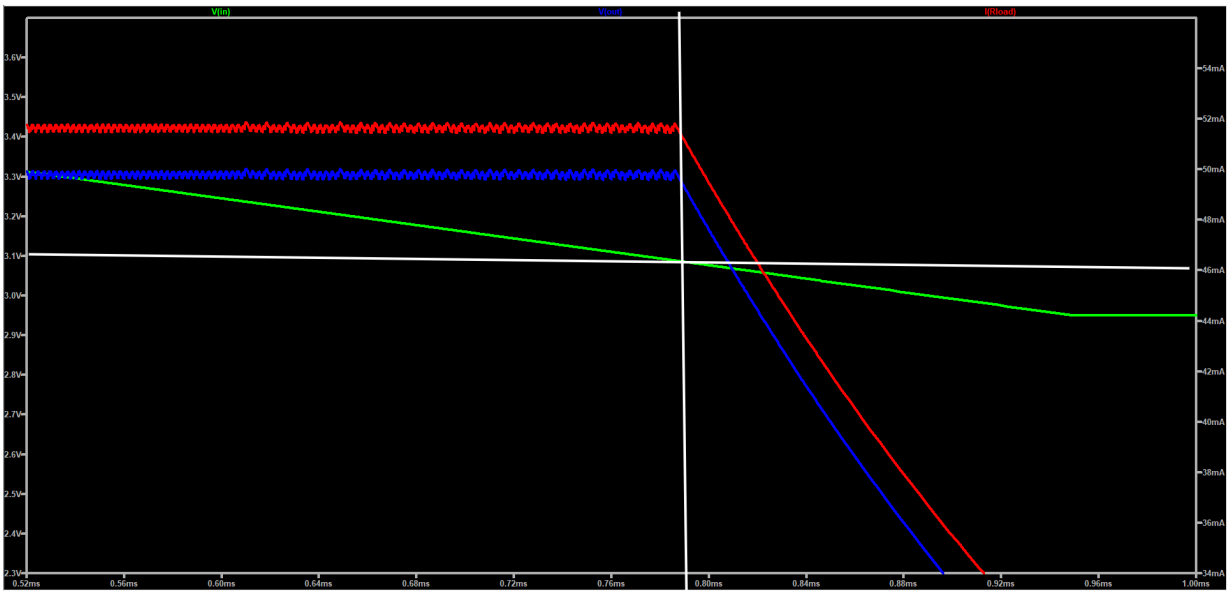


Figure 40: Battery discharge - V_IN = 3.09 V the SEPIC converter is switched off

of our system. The test is done in order to see if after turning off the SEPIC is able to turn on. It can be noticed from the figure that

- When the SEPIC turns on the ripple result smaller. In fact, when the converter turns on at the beginningthere is a ripple of 0.5 V compared to the negligible ripple of the second one (0.07 V).

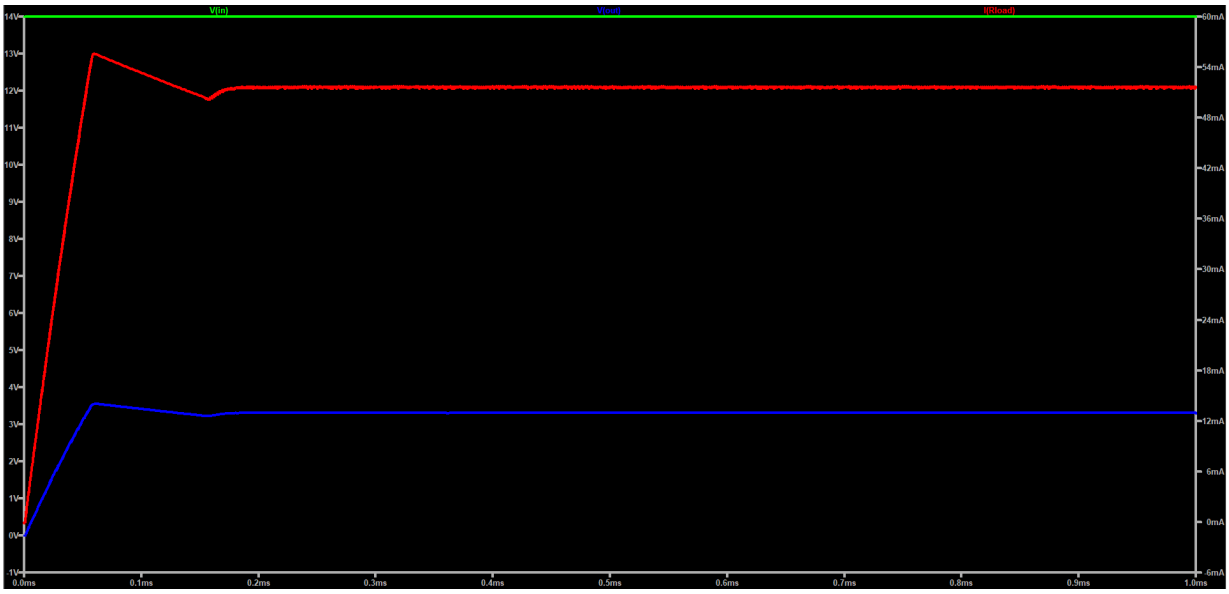


Figure 41: Overvoltage condition - V_IN = 14 V

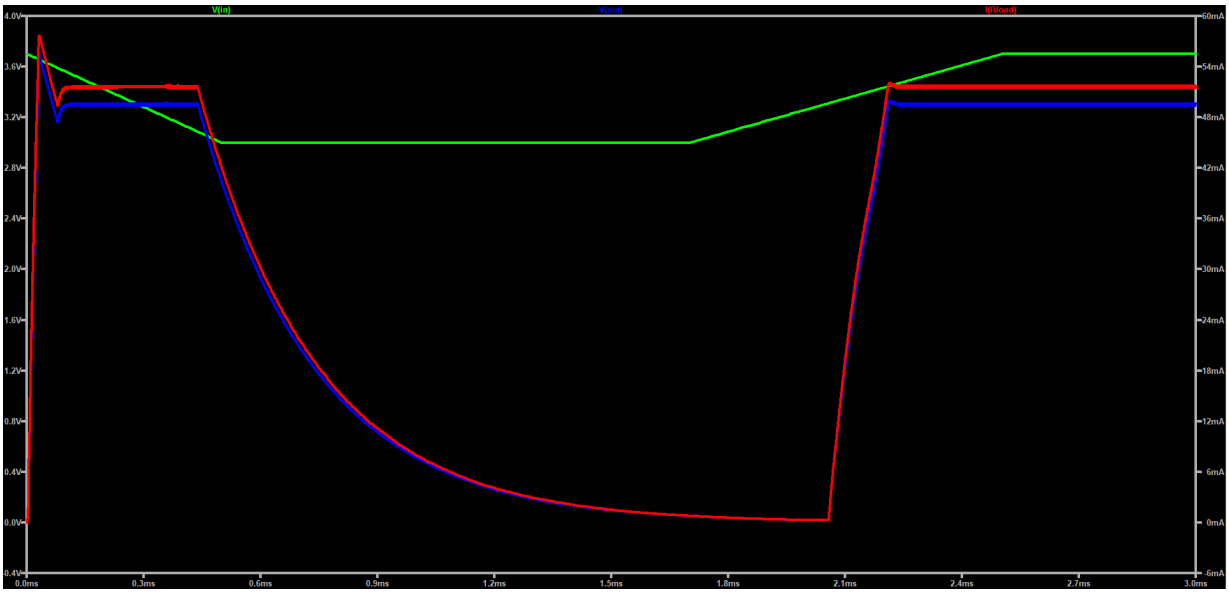


Figure 42: Turn off - turn on

- The turning on time is the same in both the turning on and it is equal to 0.1 ms.

In Figure 42, the output voltage is represented in blue, the output current in red and the input voltage in green.

5.10 Polarity Inversion and Over-current Protection

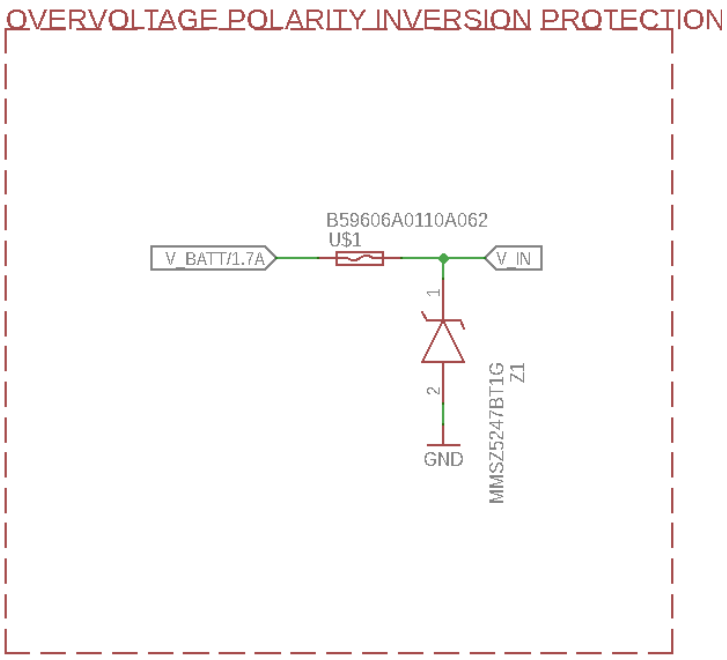


Figure 43: Polarity inversion and over-current protection circuit

Different kind of circuit can be used for the implementation of a polarity inversion and over-current protection, this circuit is made by:

- PTC fuse 180 mA.
- Zener diode 17 V.

Main advantages

- Easy to realize, low number of components needed;
- Low area occupation on the PCB;
- Low realization cost;
- Low number of simulation needed to prove the functioning.

5.10.1 Overcurrent protection

In this case we will analyze individually their features and why they are important for the CL1 design. First, in purple we have the overcurrent protection, which is connected in series to protect the circuit connected to the Thevenin equivalence for V_IN. In the first

instance the circuit is operating under normal conditions, therefore the device remains at an extremely low resistance, and allows the electrical current to flow along the path without any resistance. However, when there is an over current condition, the Polymeric PTC material of the device heats up and its resistance increases steeply. Therefore, when a short circuit occurs, the component heats up and changes rapidly from and easily current flowing channel to a nearly impossible conductor, only conducts again the fuse finds itself in a cooldown and there are no overcurrent's. At a high temperature caused by the high

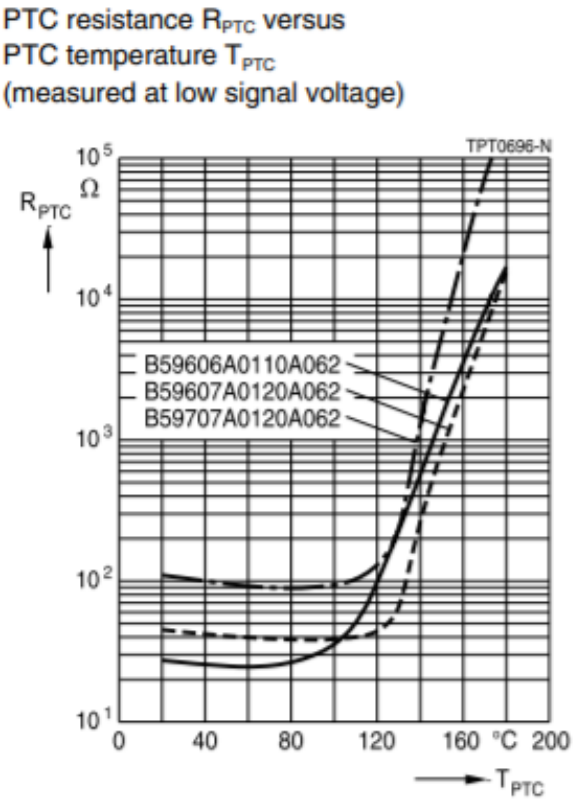


Figure 44: PTC characteristics

level of current, the PTC opens the circuit by representing an exponentially increasing resistance where eventually zero current will flow, until cooldown causing the following effect. A decrease in current and consequently in resistance. The B59606A0110A062 is a

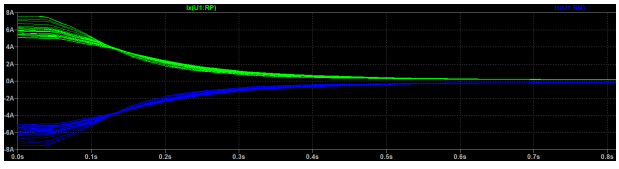


Figure 45: PTC characteristics

PTC resettable fuse, meaning an improvement of generic one-time fuses. In this case the maximum current that can be pulled by the component is 0.5A reflected by a minimum Thevenin resistance of 17 ohms(below this value the circuit will short and the PTC fuse will heat up) and within an operating tension of 30V.

5.10.2 Simulation Set-Up

On this next analysis will compare the datasheet behavior of the PTC against a simulation of the same but under a different input to test the appropriate switching of the PTC fuse with the following circuit with varying current.

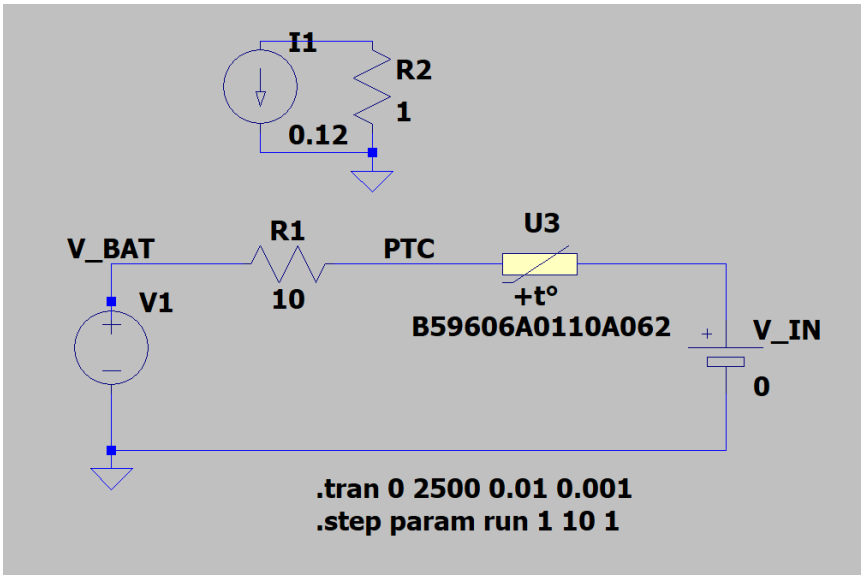


Figure 46: LTSpice Set-up

5.10.3 Simulations

In Figure 47, it is simulated that the supply is connected in the wrong way. This condition has been simulated with LTSpice through the use of a voltage generator reverse connected. It can be seen that the voltage V_In = - 12 V (Green line). In case of polarity inversion the Zener diode starts to conduct in order to protect the circuit. The next analysis corresponds to the polarity inversion protection. The initial circuit designed was the following with a Zener diode connected with its cathode to V_IN, in such a way it works as a reverse diode and shorts the output to ground in case there is an inverse voltage sensed. F1 being the variation of the DC voltage the Zener will see.
If we examine this simple circuit, it clearly does its job; however, in the case there is an overvoltage the Zener will dissipate more power than the fuse itself, until eventually the fuse blows(in CL1 case the limitation of current and create an open circuit with a very high resistance). In the reverse case the Zener will behave as a common diode. Considering

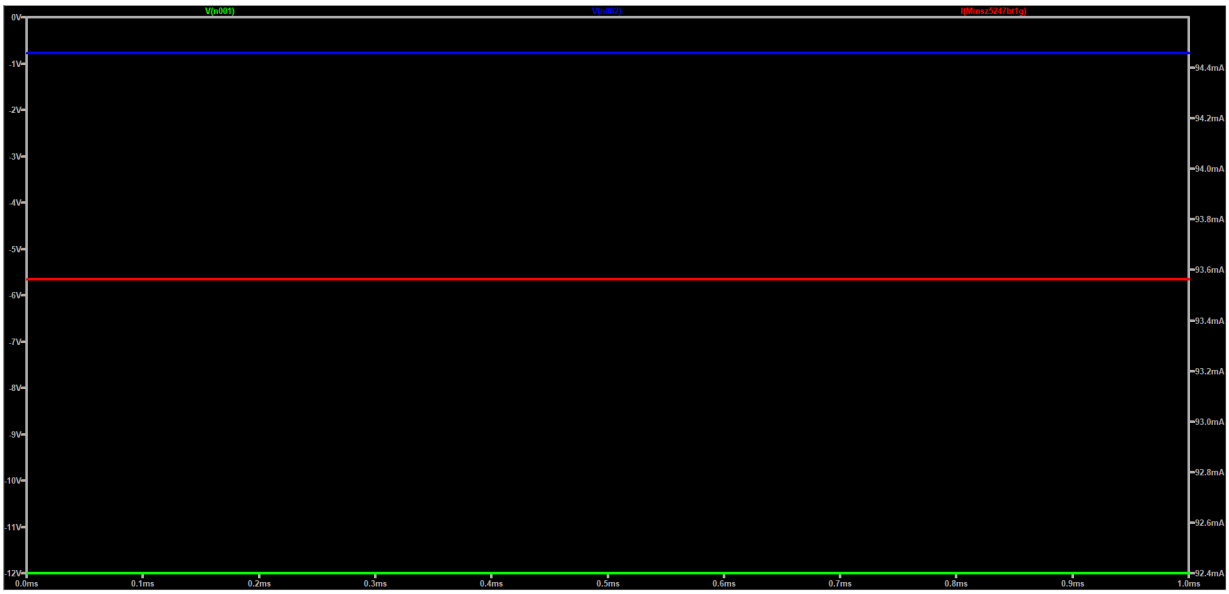


Figure 47: Behaviour of the protection in case of polarity inversion

that we want to avoid any damage on the circuit or any component that may raise the costs and avoid replacing either the Zener. Consequently, a MOSFET may solve this issue, because when reverse biased, the Zener is placed under a high finite resistance, which can be assimilated to the objective of a MOSFET. Another design recommendation is to choose a MOSFET with a low Rdson, because this can reduce the power losses. The circuit to achieve a robust polarity inversion protection we can consider the following.

The simulation below corresponds to a variable voltage as we could expect under the change of the level of the battery V_BAT, where any peak voltage can be expected. In this case The circuit is operating in an operating range of current in this case 120 mA. Next the fuse heats up and switches to a high resistance(open circuit) over time depending on the electrical resistance. In 1.3s, the voltage increases again to a high value and the resistance remains extremely high. Finally, at 1.8s the voltage is decreased to an operating range. In this example the V_IN is a Thevenin equivalent of a battery with its internal resistance.

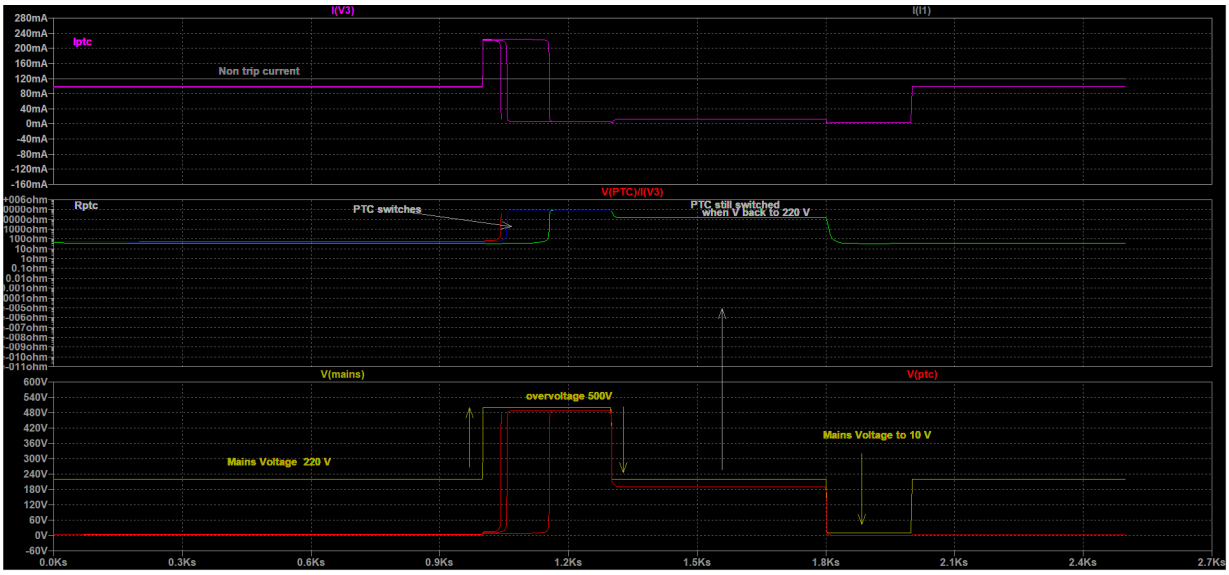


Figure 48: LTSpice Set-up

5.11 PCB Features

The idea is to realize a small PCB, ideally of dimensions equal to the LiPo battery in order to design a really small box for both of them.

5.11.1 PCB Goals

- Realize a PCB as smaller as possible;
- Realize a solution with all the functionalities;
- Realize a PCB that meet all the requirements set by the customer.
- Estimated dimension 50x50 mm;

5.11.2 PCB Main Features

- 2 layers PCB: despite a not complex design and soldering allows to reduce the size;
- GND plane has been inserted in both the layers;
- 2 mounting holes in 2 corners;
- SEPIC converter requires a particular layout in order to avoid performance degradation: planes for each signal are insereted.

5.12 PCB layout

5.12.1 PCB 2D

5.12.2 Eagle Settings

Component	Width	Drill	Isolate
GND plane	-	-	0.3 mm
VDD and VIN Traces	0.4/0.6 mm	0.35 (Eagle default)	-
Signal Traces	0.2 mm	0.35 (Eagle default)	-
Vias	Automatic diameter	0.35 (Eagle default)	-

Table 20: PCB settings

During the layout definition the following factors has been taken into account:

- **Supply line**: In order to keep the VIN (battery voltage) and VDD (components supply voltage) supply line as short as possible the connector has been positioned as close as possible to the SEPIC converter input and the SEPIC converter output (from which starts the VDD line) is located in order to easily reach the components supply pins;

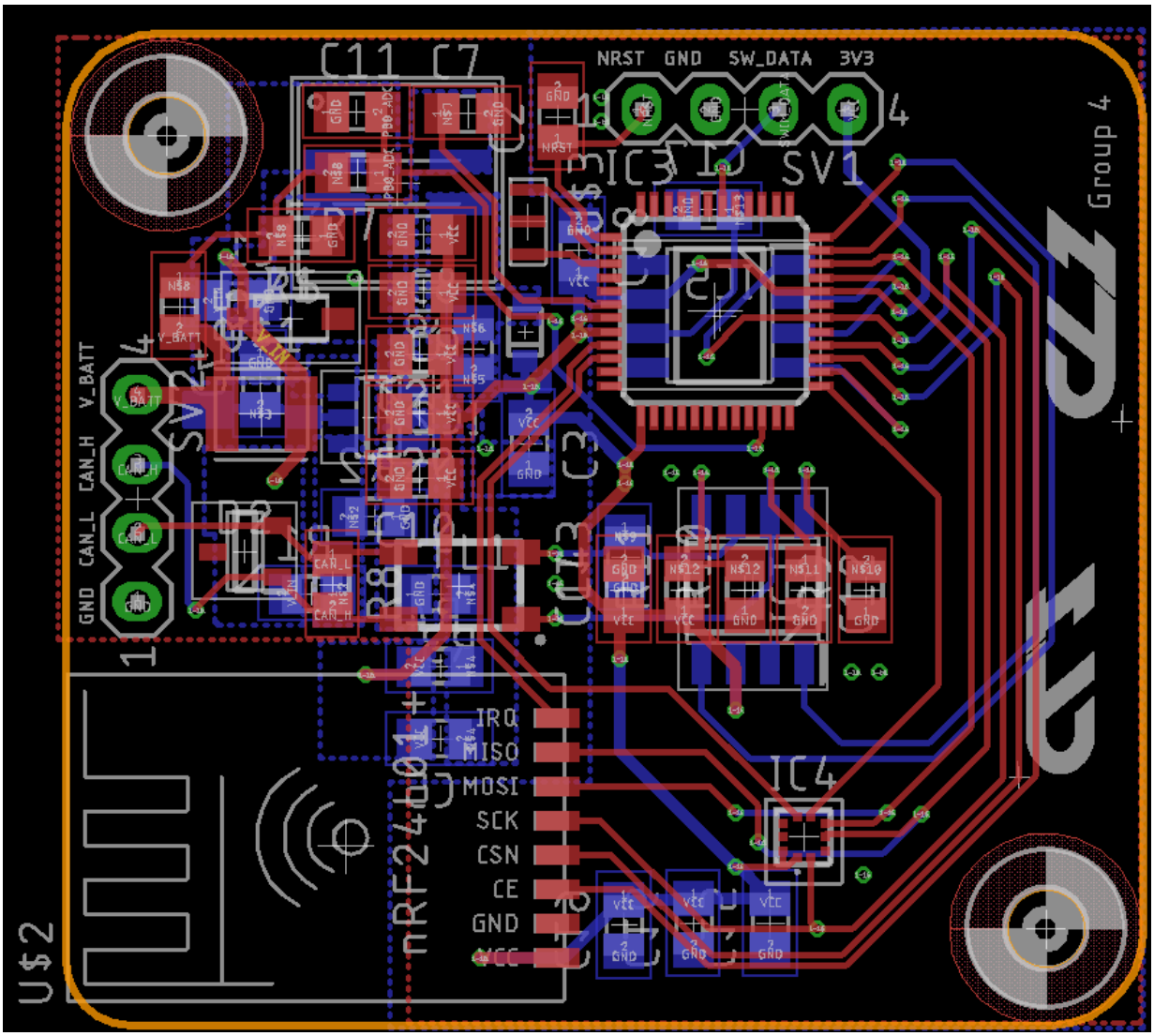


Figure 49: PCB Final layout

- SEPIC requires a particular layout that is recommended by the manufacturer. Due to the fact that it is a switching circuit a wrong positioning can lead to performance degradation. The recommended layout include both the positioning of the components and planes required in order to facilitate the dissipation. The only modification made is related to the position of the C2 capacitor, in fact according to the recommended layout the V_IN plane has to go through the capacitor.

  Issues

  The plane that connect the SEPIC controller input and the inductance L1 has a point that result weak. In fact the plane go through the two soldering pads of the capacitor with a width of less than 0.65 mm (0.65 mm is the overall spacing between the soldering pads of the 0805 capacitor).

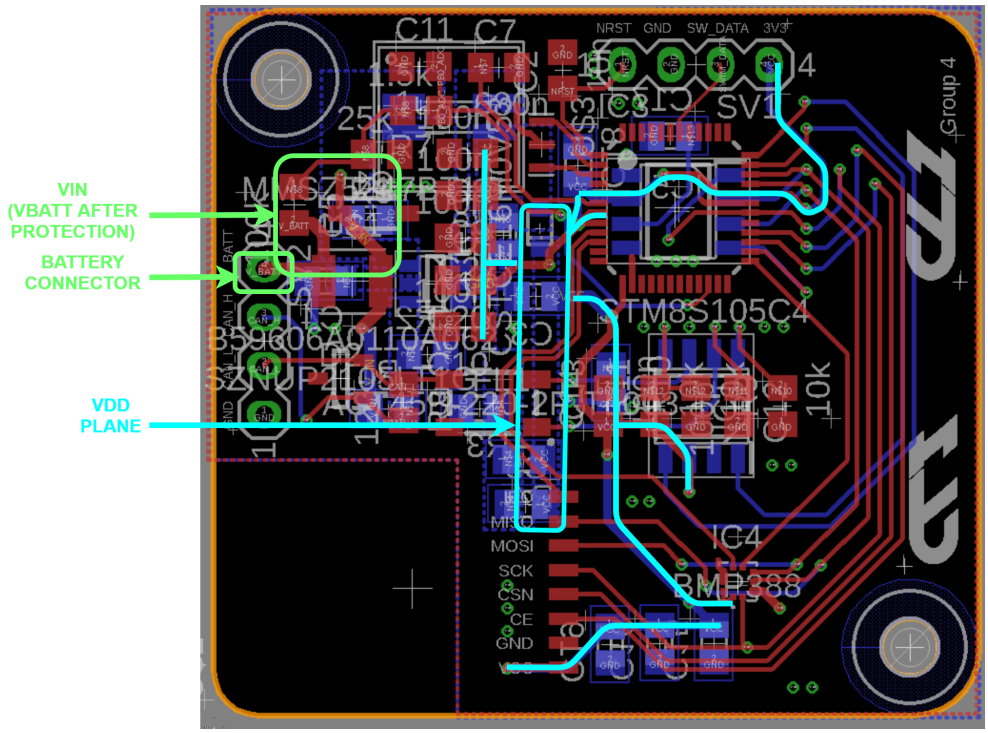


Figure 50: PCB Final layout

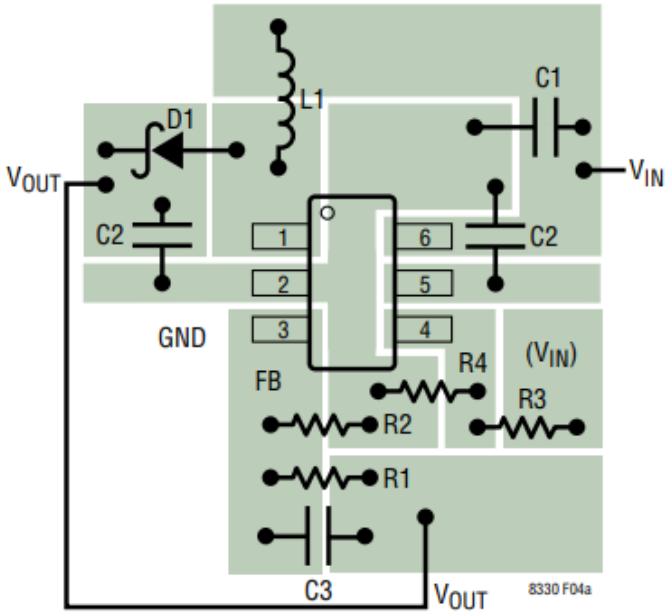


Figure 51: SEPIC recommended layout

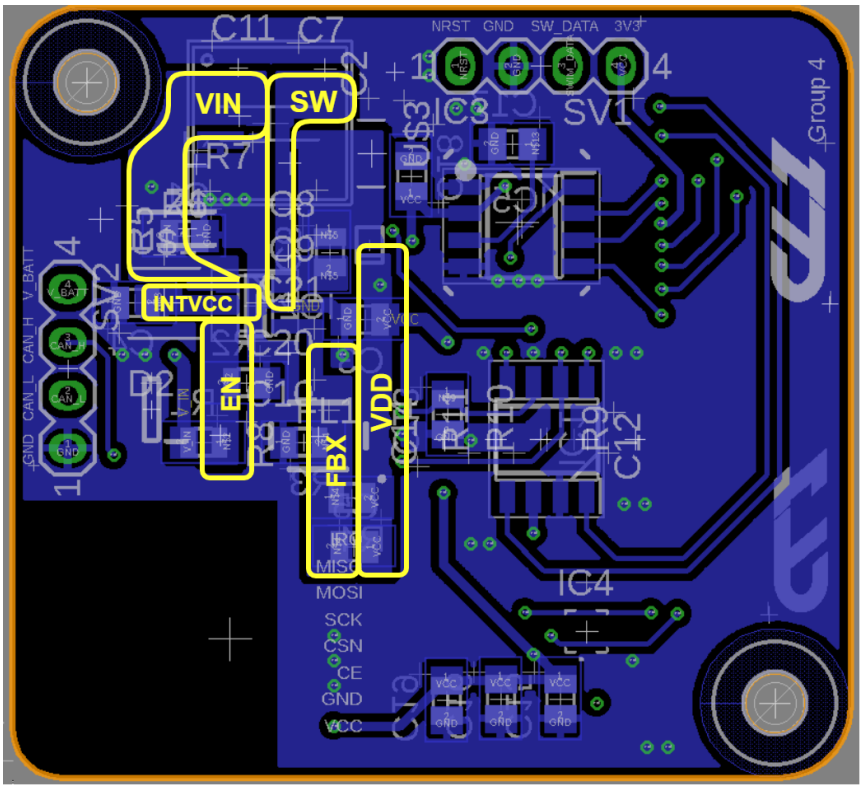


Figure 52: SEPIC layout

Solution

The capacitor has been rotated in order to create a more robust plane and avoiding to go though the capacitor pads.

- The most difficult routing is related to SPI connection. Due to the fact that all the devices (pressure sensor, accelerometer and wireless module) has to be connected to SPI interface, it leads to routing problems. As rule of thumb what it has been done is to limit as possible the presence of more than one via in each trace.
- The GND plane has been positioned in all the PCB surface. GND plane has rank 2. It covers all the PCB area and when another plane is positioned with rank 1 it overcomes the GND plane (as in the SEPIC happens, in fact SEPIC planes are configured with rank 1). The only part in which it has been cut is under the antenna printed on the module, because it can cause transmission problems.

5.12.3 PCB 3D

In the following figure the 3D view of the PCB generated with Autodesk Fusion 360.

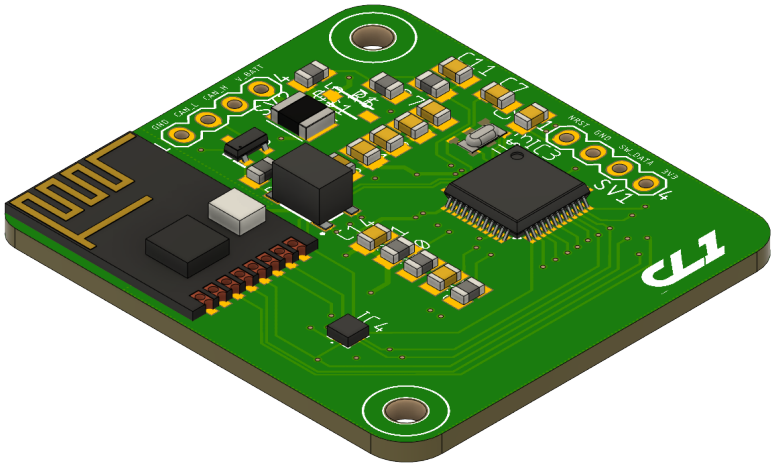


Figure 53: 3D view

5.12.4 Dimensions and Weight

Length	40 mm
Width	36.66 mm
Area	1466.4 mm^2
Weight	7 g

Table 21: Main geometrical characteristics board

The weight is intended with all the components soldered and not considering the wires.

5.13 Estimated Timeline for Production

In the following table are listed both the expected production date and the real one. The main goal reached is to have the PCB design ready for production a week before the estimated date. This gives the possibility to have more time for testing and validation phase after production.

	Estimated Date	Effective Date
PCB layout completion	November 14th, 2019	November 6th, 2019
Production	November 18th, 2019	November 11th, 2019
Expected delivery	November 25th, 2019	November 18th, 2019
Soldering components	November 26th, 2019	November 19th, 2019
Start testing	November 27th, 2019	November 20th, 2019

Table 22: Production timeline

5.14 Soldering

In order to properly solder the entire board, the following equipment is used:

- Soldering station;
- Hot air station;
- Flux: to avoid MCU pin short circuited one with the other.

PROBLEM ENCOUNTERED: The pressure sensor is too small to be soldered. The choice not to solder this component came from the fact that it is preferable to avoid short circuit during components soldering, in particular between VDD and GND. All the details related to the circuit and functional testing are described in the validation section.

5.15 Future Developments

5.15.1 High Side Output Implementation

Main reasons for which High Side Output is not implemented at the current stage of the project are:

- Complex design of the system;
- Functional simulations are needed;
- Low knowledge of this kind of circuit by team members contributes to the choice;
- It has been evaluated too complex to be designed and tested until the project deadline. Moreover, the deadline set for PCB production is November 14th, 2019.
- Due to the presence of the MOSFET that has to withstand a current of 1 A, more accurate thermal consideration are needed and eventually an heat sink is needed;
- Higher power consumption respect to the realized solution.

For all this reasons the it is not implemented in the realized solution.
In the following part the requirements and a brief description of the High Side design is conducted.

- Choice of a 1 A NMOS or PMOS transistor;
- Gate-driver to avoid that high current flows on the MCU pin used for PWM generation.

In the following table are listed the main characteristics of MCP14A015 that has been selected as suitable for High-Side output design.

	MCP14A0152	**TC4431**
Manufacturer	Microchip	Microchip
Input supply voltage	4.5 V to 18 V	4.5 V to 30 V
Peak output current	1.5 A	1.5 A
AEC-Q100	No	No
Temperature range	-40 to +125 °C	-40 to +125 °C
Packages available	6-lead SOT-23 / DFN-6	SOIC-8 / PDIP-8/ CERDIP-8

Table 23: MCUs features

In Figure 54, it is shown an example of the circuit configuration for high side output. The high side gate driver drives the gate of the transistor. Pin CTL of the gate driver takes as input the PWM signal generated by the MCU, which duty cycle is a function of the pressure value.

5.15.2 Reduced Size Components Upgrade

In the design of a PCB of smaller dimensions some components has been modified respect to the realized solution.

- All the 0805 resistor and capacitors has been substituted with 0403 package.
- It is not possible to reduce the MCU size because the STM8AF62x (VQFN package) has no support for SPI communication needed for NRF24L01+ communication;
- AIS3624DH 3-axis accelerometer has been used to substitute AIS2120SX;
- Due to the fact that the wireless module is no more present antenna design on the PCB and components for the functioning of NRF24l01+ are needed;
- 1210 PTC fuse for over-current protection is substituted with 0603 package;
- All the 0805 resistor and capacitors has been substituted with 0403 package.
- Smaller CAN transceiver and common mode choke.

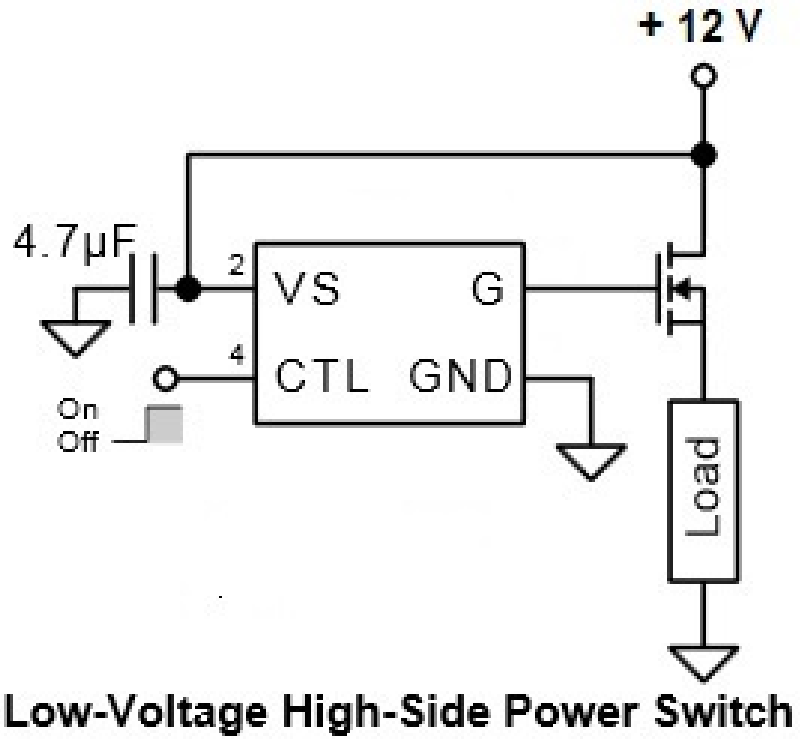


Figure 54: High Side intended design

5.15.3 Reduce Size CL1 Board Design

After that all the components has been substituted, what has been done is to slightly change the schematic in order to add capacitors and resistors for the new components proper functioning.
In the Appendix B is attached the schematic of CL1 board reduced size.
In Figure 55 the design of the board.

Length	33.91 mm
Width	21.50 mm
Area	729.07 mm^2
Expected weight	3.5 g

Table 24: Main geometrical characteristics board

Achieved Goals

- Size reduction of 50.3% respect the realize CL1 custom board;

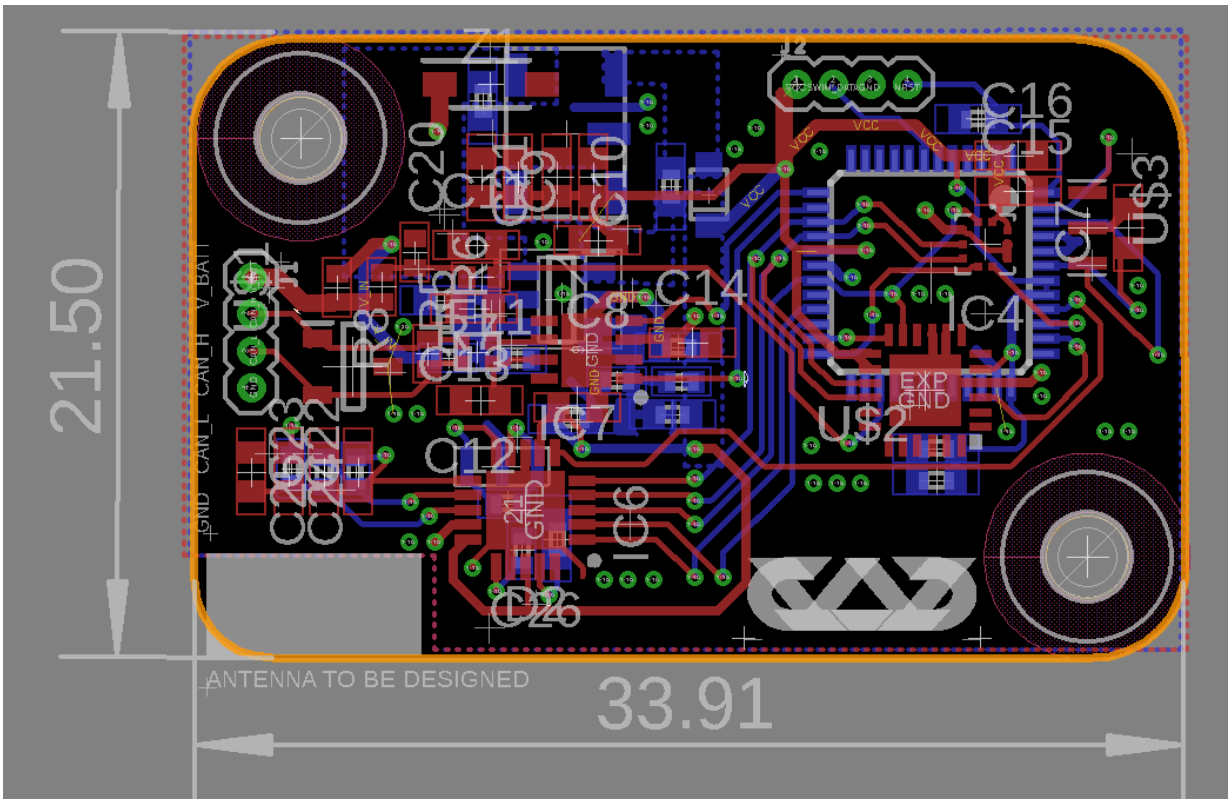


Figure 55: Reduced size board design

- Size reduction has been achieved without changing too much the previous design, design blocks has been reused;
- A BOM for the reduced size board has been compiled and the resulting cost is equal to the realized board.

Next Steps

- Further studies on the antenna design, in order to print the antenna on the PCB studies on the proper shape of the antenna, on how the antenna radiates and on possible EMI problems has to be conducted;
- In order to further reduce the dimension, the MCU has to be positioned on the top layer above the SEPIC converter. Due to the presence of the switching converter can lead to interference problems so an accurate analysis is needed.

5.15.4 Connector

In the realized custom board the connector has been not used because time before the order completion was not enough to find a suitable connector that is automotive compliant, board surface mounted, small size and low weight. The use of a connector has not to impact too

much on the dimensions and on the weight of the system, in particular the dimension that it necessary to be kept as low as possible is the height, that is what impact on the air flow on the wing.
As a future development there is the choice of a proper connector, below are listed the requirements of the connector that has been identified:

- The height has to be less than 2.5 mm that is the height of the highest component, in order to do not affect the design of box and its height;
- Automotive compliant;
- In order to obtain a system that can also be exposed to water, dust and other environmental factors, the connector must be sealed. Further studies has to be conduction in order to seal also the board.

6 Firmware Design

This Section focuses on the description of the firmware. The section aims at explaining what are the operation performed, and why some design choices have been made. On this latter point, it is worth to note that the software has to run on an 8-bit MCU, hence the designer cannot always adopt fancy solutions.
The section is organized as follows:

- A short introduction to the IDE adopted is presented;
- Firmware designed for the evaluation description;
- Firmware designed for the *CL1* custom board description;

6.1 Firmware Goals

- Realize a firmware solution that works with the chosen hardware;
- Allow an almost complete reuse of the code from the evaluation prototype to the CL1 custom board;
- Consider solution that can improve and optimize the firmware for future developments.

6.2 Selection of the IDE

As any software designer knows, a user-friendly IDE, coupled with a powerful compiler, greatly reduces the effort required to develop the source code.

With that in mind, the team explored the available solutions proposed by ST Microelectronics. The first one was ST Visual Develop which, as the name suggests, is provided directly by ST Microelectronics. The main drawback of this solution was that look (and actually, is) very aged; the compatible compiler, Cosmic for C, supports only C89, a version of the C programming language which dates back to 1983. It comes without saying that it introduced tight constraint in the syntax of the code.

The second solution proposed was the use of IAR Embedded Workbench for STM8, which is not a free software but it provides a 1-month trial version. The IAR compiler has somewhat the same limitations faced with the use of Cosmic compiler, however in this case the IDE is more up-to-date and comes with a superior debugger.

The third solution that was considered, which in turn is also the one adopted for the development of the firmware, follows a somewhat unconventional path. The firmware development team noticed that ST Microelectronic provides a package that makes the STM8S MCU compatible with the Arduino IDE. This was regarded by the team as a positive news since the IDE is very easy to be used and offers a lot of handy features. This compatibility is originally meant for the Nucleo-208RB (Figure 56). However, the MCU coded STM8S208RB is almost identical to the MCU adopted for this project, namely

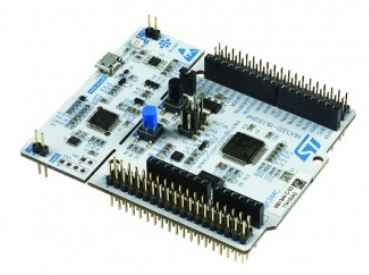


Figure 56: Nucleo-208RB

STM8AF5822; both even adopt the same libraries (again, officially provided by ST Microelectronics). The main difference between the two MCUs is the fact that the latter is automotive compliant, and that the former one has a greater amount of PINs (64 vs. 48). However, this turned out not being an issue, since the PIN mapping was exactly the same one. After all, it was understood that even the adopted MCU was fully supported by Arduino IDE without the need of any modification.

At this point, it is worth nothing that no one of the three proposed solutions was able to fully solve a problem that was faced. The problem is the following one: the barometric sensor selected (Bosch BMP388) uses 64-bit floating point data (stored in an internal register) for calibration purposes. This would not usually be a problem, since most of the compiler are compatible with this data type. However, all the compilers compatible with STM8 MCUs (namely Cosmic for C, Cosmic for C++, IAR and Raisonance) do not support 64-bit data. So the group was forced to abandon the BMP388 sensor. For the evaluation part it was replaced with the older BMP180, which however is not compliant to the requirements (works only up to 110 hPa), while for the *CL1* board it was replaced with the LTS22HH ST. This latter sensor had been taken into account even during the initial selection of components, however it was discarded since the cost of the expansion board (useful for evaluation purposes) was almost the double with respect to the breakout board mounting the BMP388, and its performances are slightly worse than BMP388. However, it is fully compatible with the adopted MCU family, since it requires no 64-bit data. This issue was not noticed during the initial selection of components.

6.3 Unified Code - Evaluation and Custom Board Characteristics

Evaluation board's firmware has to be designed having in mind which are the requirements for the final custom board, since the code has to be maintained the same as much as possible.

The list presents the section of the code that has been implemented in the evaluation board and are reused *as is* for the final custom board.

- MCU configuration;

- CAN message transmission;
- Wireless transmission;

While the part that change is the part related to sensor acquisition, since the sensor used for the evaluation part are different from the ones adopted on the *CL1* board. Thus, once the evaluation board's firmware is ready and tested, obtaining the code for the *CL1* board it is a breeze.

6.4 Main Characteristics

The main peculiarity of the *CL1* board is its capability of working in different modes.

- **Controller board** for data reception and routing on the CAN line.
- **Sensor board** for data acquisition and transmission via wireless.
- **Controller + Sensor board**: data acquisition and transmission on CAN line; this configuration can be used in case a wired solution is desired or if the user wants to add another measurement point in the controller.

For this reason the firmware to be flashed into the two board is not exactly the same; different modules have to be configured according to the application, as reported in the following list.

- **Controller board**
  - Configuration as receiver.
  - CAN interface enabled.
  - Sensor acquisition disabled.
- **Sensor board**.
  - Configuration as transmitter.
  - CAN interface disabled.
  - Sensor acquisition enabled.
- **Controller + Sensor board**.
  - Configuration as receiver or wireless transmission disabled.
  - CAN interface enabled.
  - Sensor acquisition enabled.

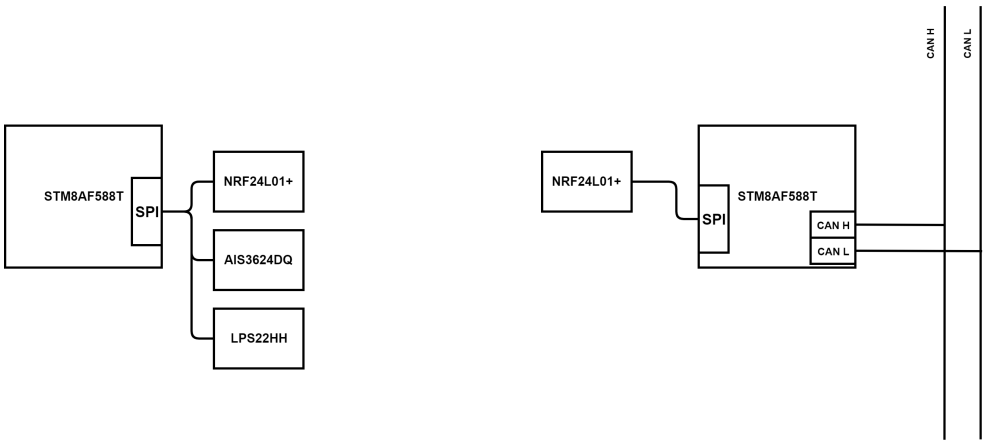


Figure 57: Proposed solution architecture highlighting main peripherals used

The firmware is provided to the user as an unified version of the code, which allows to easily program the board to work in one of the three foreseen modes. The user has simply to select the desired working mode, compile and flash the code to the board. In alternative, it is also possible to directly provide three different versions of the firmware already compiled, ready to be flashed on the board.

It is worth to note that despite implementing three different working modes within the same source file, there is no waste of resources since only relevant parts are compiled; this is made possible exploiting pre-processor directives.

The reader can refer to Appendix 12 to have a deeper view on the implementation.

The code begins with the selection of the functionality required; the user is asked to select just one functionality, otherwise the code produced won't make sense. No particular lockings to avoid inconsistencies are implemented since the firmware is provided yet compiled, or it is provided as source code to expert users.

```
24  /* Uncomment according to what is the functionality required
25  * PAY ATTENTION: JUST UNCOMMENT ONE AT A TIME!
26  */
27  #define SLAVE_MODE
28  //#define MASTER_MODE
29  //#define WIRED_SENSOR
```

Then, the preprocessor tailors the code accordingly.

6.5 Evaluation Board Set-Up

6.5.1 Sampling Frequency Considerations

The sampling frequency of the LIS3DH (accelerometer) could be selected. The user has the possibility to select the required sampling frequency from one in the following list: 1, 10, 25, 50, 100, 200, 400, 1600, 5000 Hz. The selection *a priori* is performed considering that at each cycle, the accelerometer has to provide a fresh data. Thus, it has been estimated that a cycle (data acquisition from the two sensors - wireless data transmission), even in

the best case, won't be shorter than 10 ms. This suggests to select a frequency of 100 Hz. This frequency can be modified in a future release of the firmware, after the assessment of the real performances.

6.5.2 LIS3DH and BMP280 Configuration

The application requires a data range $\pm 15g$, and accelerometer is mainly required to measure vibrations; barometer instead is required to acquire the actual atmospheric pressure. This suggest to use no filtering of the data, since the application requires getting even the minimum pressure variation and all the acceleration peaks.

6.5.3 NRF24L01+ Configuration

The wireless module transmit the data in the 2.4 GHz ISM band. The band is divided into 126 channels, and the user has the possibility to select any of them in order to perform the communication. Of course, both transmitter and receiver should be tuned on the same band to maximize the probability of successful transmission.

Settings

- Transmitter radio ID = 1
- Receiver radio ID = 0
- Bitrate = 2 Mbps (alternatives: 250 Kbps, 1 Mbps)
- Channel = 100 (center frequency: 2.48 GHz)

6.6 Code Overview - Sensor Board

To avoid waste of resources, just the needed libraries are included.

```
36        /* Include the libraries, according to what is the functionality required */
37        #include "Wire.h"
38        #include <SPI.h>

42        #if defined(SLAVE_MODE) || defined(MASTER_MODE)
43        #include <NRFLite.h>
44        #endif
45        #if defined(SLAVE_MODE) || defined(WIRED_SENSOR)
46        #include <Adafruit_BMP085.h>
47        #include "lis3dh-motion-detection.h"
48        #endif
```

Required variables are defined.

```
63        #if defined(SLAVE_MODE)
64        /* Variables name for the Wireless transmission */
65        const static uint8_t RADIO_ID = 1;             // Transmitter radio's id.
66        const static uint8_t DESTINATION_RADIO_ID = 0; // Id of the radio we will transmit
              to.
```

```
67          #endif
68          #if defined(SLAVE_MODE) || defined(MASTER_MODE)
69          const static uint8_t PIN_RADIO_CE = 2;          // 2 = PE0
70          const static uint8_t PIN_RADIO_CSN = 5;         // 5 = PC2
```

The adopted microcontroller can handle data up to 32 bits; therefore, any data up to 32 bits can be sent. The adopted wireless module performs the transmission one data packet at a time, each packet consisting of 8 bits. Thus at the receiver side the controller has to recombine the packet received into proper size data.

```
72          /* Definition of the radio packet */
73          struct RadioPacket // Any packet up to 32 bytes can be sent.
74          {
75          uint8_t FromRadioId;
76          int32_t PressureTx;
77          int32_t AccelerationTx;
78          uint8_t FailedTxCount;
79          };
80
81          NRFLite _radio;
82          RadioPacket _radioData;
83          #endif
```

Sensors are to be initialized appropriately. The application requires a data range $\pm 15g$, and accelerometer is mainly required to measure vibrations; barometer instead is required to acquire the actual atmospheric pressure. This suggest to use no filtering of the data, since the application requires getting even the minimum pressure variation. For what concerns the sensor's sample rate, we need to have one new sample each loop iteration. Thus, the sample rate depends on the cycle time, which is not known *a priori*. Thus the initial selection has been 100 Hz, which is correct if the cycle time is greater than 10 ms. After assessment of the performance is it possible to reduce it to 50 Hz, thus reducing the power consumption.

```
87          #if defined(SLAVE_MODE) || defined(WIRED_SENSOR)
88          /* Here we have the configuration parameters for the accelerometer  */
89          uint16_t sampleRate = 100;  // HZ - Samples per second - 1, 10, 25, 50, 100, 200,
                400, 1600, 5000
90          uint8_t accelRange = 16;    // Accelerometer range = 2, 4, 8, 16g
91          LIS3DH myIMU(0x18);         // Default address is 0x18.
92          int16_t dataHighres = 0;    // Save the raw data before conversion to float
93
94
95          /* Configuration parameter for the barometer */
96          Adafruit_BMP085 bmp;
97
98          float AccAsseZ = 0;         // Measured acceleration
99          int32_t AccAsseZ_mg = 0;    // Acceleration in milli-g
100         float Temp_Meas = 0;        // Measured temperature
101         int32_t Press_Meas = 0;     // Measured pressure
102         #endif
```

Then, the devices attached to the wireless sensor are set-up. If devices are not initialized correctly, the error is displayed and the execution is trapped into an infinite dummy loop.

```
105         #if defined(SLAVE_MODE)
106         /* Initialization of the SLAVE node */
107         void setup()
108         {
109         Serial.begin(115200);
```

```
110         delay(1000); //wait until serial is open...
111
112         /* Initialization of the accelerometer */
113         if( myIMU.begin(sampleRate, 1, 1, 1, accelRange) != 0 )
114         {
115         Serial.print("Failed to initialize IMU.\n");
116         }
117         else
118         {
119         Serial.print("IMU initialized.\n");
120         }
121
122         /* Detection threshold can be from 1 to 127 and depends on the Range
123         * chosen above, change it and test accordingly to your application
124         * Duration = timeDur x Seconds / sampleRate */
125         myIMU.intConf(INT_1, DET_MOVE, 13, 2);
126         myIMU.intConf(INT_2, DET_STOP, 13, 10);
127
128         uint8_t readData = 0;
129
130         // Confirm configuration:
131         myIMU.readRegister(&readData, LIS3DH_INT1_CFG);
132         myIMU.readRegister(&readData, LIS3DH_INT2_CFG);
133
134         // Get the ID:
135         myIMU.readRegister(&readData, LIS3DH_WHO_AM_I);
136         Serial.print("Who am I? 0x");
137         Serial.println(readData, HEX);
138
139
140         /* Initialization of the barometer */
141         if (!bmp.begin())
142         {
143         Serial.println("Could not find a valid BMP085 sensor, check wiring!");
144         while (1);
145         }
146
147
148         /* Initialization of the nRF24 */
149         // By default, 'init' configures the radio to use a 2MBPS bitrate on channel 100 (
                channels 0-125 are valid).
150         // Both the RX and TX radios must have the same bitrate and channel to communicate
                with each other.
151         // You can run the 'ChannelScanner' example to help select the best channel for
                your environment.
152         // You can assign a different bitrate and channel as shown below.
153         //   _radio.init(RADIO_ID, PIN_RADIO_CE, PIN_RADIO_CSN, NRFLite::BITRATE250KBPS,
                0)
154         //   _radio.init(RADIO_ID, PIN_RADIO_CE, PIN_RADIO_CSN, NRFLite::BITRATE1MBPS, 75)
155         //   _radio.init(RADIO_ID, PIN_RADIO_CE, PIN_RADIO_CSN, NRFLite::BITRATE2MBPS,
                100) // THE DEFAULT
156
157         if (!_radio.init(RADIO_ID, PIN_RADIO_CE, PIN_RADIO_CSN))
158         {
159         SOS_blinking_LED();
160
161         Serial.println("Cannot communicate with radio");
162         while (1); // Wait here forever.
163         }
164
165         _radioData.FromRadioId = RADIO_ID;
166         }
167         #endif
```

Finally, the execution of the main program starts. As it can be seen, it consists of an infinite

loop which continuously samples the two sensors, prepare the data packet for wireless transmission, and finally sends the data, waiting (optionally) for an acknowledgment to be recived before starting the loop again. Since the code has been developed to run on an evaluation board, some convenience functions are introduced, namely printing to the serial monitor which are the data the board is sending over the radio channel, and whether the packet has been received or not. These lines will be omitted in the code for the custom board, thus saving memory and execution time.

```
245         #if defined(SLAVE_MODE)
246         /* Execution of the main program */
247         void loop()
248         {
249         /* Here perform the acquisition from the accelerometer */
250         dataHighres = 0;
251         myIMU.readRegisterInt16( &dataHighres, LIS3DH_OUT_X_L );
252         myIMU.readRegisterInt16( &dataHighres, LIS3DH_OUT_Z_L );
253         AccAsseZ = myIMU.axisAccel( Z );     // store the data in floating point
254         AccAsseZ_mg = (int32_t)(AccAsseZ * 1000);
255         // Print the result, i.e. acceleration in mg
256         Serial.print("Acc. axis Z in mg = ");
257         Serial.println(AccAsseZ_mg); // print with 4 digit after comma
258
259         /* Perform the acquisition from the barometer */
260         Temp_Meas = bmp.readTemperature();
261         Press_Meas = bmp.readPressure();
262         // Print the result: Temperature and Pressure reading
263         Serial.print("Temperature = ");
264         Serial.print(Temp_Meas);
265         Serial.println(" *C");
266         Serial.print("Pressure = ");
267         Serial.print(Press_Meas);
268         Serial.println(" Pa");
269         Serial.print("\n");
270
271         /* Send the data */
272         _radioData.PressureTx = Press_Meas;
273         _radioData.AccelerationTx = AccAsseZ_mg;
274         Serial.print("Sending ");
275         Serial.print(_radioData.PressureTx);
276         Serial.print(" Pa, and ");
277         Serial.print(_radioData.AccelerationTx);
278         Serial.print(" mg");
279
280         // By default, 'send' transmits data and waits for an acknowledgement.  If no
                acknowledgement is received,
281         // it will try again up to 16 times.  You can also perform a NO_ACK send that does
                not request an acknowledgement.
282         // The data packet will only be transmitted a single time so there is no guarantee
                it will be successful.  Any random
283         // electromagnetic interference can sporatically cause packets to be lost, so
                NO_ACK sends are only suited for certain
284         // types of situations, such as streaming real-time data where performance is more
                important than reliability.
285         //   _radio.send(DESTINATION_RADIO_ID, &_radioData, sizeof(_radioData), NRFLite::
                NO_ACK)
286         //   _radio.send(DESTINATION_RADIO_ID, &_radioData, sizeof(_radioData), NRFLite::
                REQUIRE_ACK) // THE DEFAULT
287
288         if (_radio.send(DESTINATION_RADIO_ID, &_radioData, sizeof(_radioData))) // Note
                how '&' must be placed in front of the variable name.
289         {
290         Serial.println("...Success");
```

```
291         }
292         else
293         {
294         Serial.println("...Failed");
295         _radioData.FailedTxCount++;
296         }
297
298         Serial.print("Failed␣transmission:␣");
299         Serial.print(_radioData.FailedTxCount);
300         Serial.print("\n");
301         Serial.print("\n");
302         Serial.print("\n");
303         Serial.print("\n");
304
305         /* At the end, restart the cycle after a certain amount of time (ms) */
306         delay(1);
307         }
308         #endif
```

6.7 Code Overview - Controller board

The wireless receiver node needs to acquire the data coming from the wireless sensor, and transmit them over the CAN bus. Again, just the required libraries are included.

```
36          /* Include the libraries, according to what is the functionality required */
37          #include "Wire.h"
38          #include <SPI.h>
39          #if defined(MASTER_MODE) || defined(WIRED_SENSOR)
40          #include <stm8s_can.h>
41          #endif
42          #if defined(SLAVE_MODE) || defined(MASTER_MODE)
43          #include <NRFLite.h>
44          #endif
```

Then, parameters needed by the application are defined.

```
51          #if defined(MASTER_MODE) || defined(WIRED_SENSOR)
52          /* Initialization of the parameters for CAN transmission */
53          unsigned long ID = 0;
54          unsigned char tx_buffer[0x08] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
                // payload = 8 bytes
55          #endif

59          #if defined(MASTER_MODE)
60          /* Variables name for the Wireless transmission */
61          const static uint8_t RADIO_ID = 0;              // Receiver radio's id.  The
                transmitter will send to this id.
62          #endif

59          #if defined(SLAVE_MODE) || defined(MASTER_MODE)
60          const static uint8_t PIN_RADIO_CE = 2;         // 2 = PE0 in the STM8AF board -->
                to be changed to 6 = PC3 in final board
61          const static uint8_t PIN_RADIO_CSN = 5;        // 5 = PC2
62
63          /* Definition of the radio packet */
64          struct RadioPacket // Any packet up to 32 bytes can be sent.
65          {
66          uint8_t FromRadioId;
67          int32_t PressureTx;
68          int32_t AccelerationTx;
69          uint8_t FailedTxCount;
70          };
```

```
71
72          NRFLite _radio;
73          RadioPacket _radioData;
74          #endif
```

The communication protocols are set-up.

```
170         #if defined(MASTER_MODE)
171         /* Initialization of the MASTER node */
172         void setup()
173         {
174         Serial.begin(115200);
175         delay(1000);
176
177         pinMode(PC3, OUTPUT);
178
179
180         // By default, 'init' configures the radio to use a 2MBPS bitrate on channel 100 (
                channels 0-125 are valid).
181         // Both the RX and TX radios must have the same bitrate and channel to communicate
                with each other.
182         // You can run the 'ChannelScanner' example to help select the best channel for
                your environment.
183         // You can assign a different bitrate and channel as shown below.
184         //   _radio.init(RADIO_ID, PIN_RADIO_CE, PIN_RADIO_CSN, NRFLite::BITRATE250KBPS,
                0);
185         //   _radio.init(RADIO_ID, PIN_RADIO_CE, PIN_RADIO_CSN, NRFLite::BITRATE1MBPS, 75)
                ;
186         //   _radio.init(RADIO_ID, PIN_RADIO_CE, PIN_RADIO_CSN, NRFLite::BITRATE2MBPS,
                100); // THE DEFAULT
187
188         if (!_radio.init(RADIO_ID, PIN_RADIO_CE, PIN_RADIO_CSN))
189         {
190         SOS_blinking_LED();
191
192         //Serial.println("Cannot communicate with radio");
193         while (1); // Wait here forever.
194         }
195
196         setup_CAN();
197         }
198         #endif
```

Since the initialization of the CAN consists of several lines of code, it has been encapsulated into a function.

```
214         #if defined(MASTER_MODE) || defined(WIRED_SENSOR)
215         void setup_CAN()
216         {
217         CAN_DeInit();
218
219         CAN_Init(CAN_MasterCtrl_NoAutoReTx,
220         CAN_Mode_Normal,
221         CAN_SynJumpWidth_1TimeQuantum,
222         CAN_BitSeg1_8TimeQuantum,
223         CAN_BitSeg2_7TimeQuantum,
224         4);
225
226         CAN_FilterInit(CAN_FilterNumber_0,
227         ENABLE,
228         CAN_FilterMode_IdMask/*CAN_FilterMode_IdList*/,
229         CAN_FilterScale_32Bit,
230         0x00,
231         0x00/*0x08*/,
232         0x00/*0x24*/,
233         0x00/*0x68*/,
```

```
234         0x00/*0x00*/,
235         0x00/*0x08*/,
236         0x00/*0x24*/,
237         0x00/*0x68*/);      //Check messages from ID: 0x1234  // Check Filter Register
                Table
238
239         CAN_ITConfig(CAN_IT_FMP,ENABLE);
240         }
241         #endif
```

Finally, the main program is executed: it consists of an infinite loop where the sensor continuously polls on a flag that indicates that the wireless module has received a new data; then the data is decomposed into 8-bit packets that are sent on the CAN bus. The CAN protocol allows each message to carry 8 byte payload; luckily, the application need to send one pressure and one acceleration information, each one being a 32-bit message. Thus, a single CAN message is sufficient to send all the information related to one acquisition performed by the wireless sensor.

```
311         #if defined(MASTER_MODE)
312         /* Execution of the main program */
313         void loop()
314         {
315
316         digitalWrite(PC3, HIGH);   // just used as a simple debug, to see if the code has
                arrived there
317         while (_radio.hasData())
318         {
319         digitalWrite(PD0, HIGH);
320
321         _radio.readData(&_radioData); // Note how '&' must be placed in front of the
                variable name
322
323
324         String msg = "Radio␣";
325         msg += _radioData.FromRadioId;
326         msg += ",␣pressure:";
327         msg += _radioData.PressureTx;
328         msg += "␣Pa,␣acc:";
329         msg += _radioData.AccelerationTx;
330         msg += "␣mg.␣";
331         msg += _radioData.FailedTxCount;
332         msg += "␣Failed␣TX";
333
334         Serial.println(msg);
335
336         digitalWrite(PD0, LOW);
337
338
339         /* Create the bit mask for the PRESSURE data to be transmitted on the CAN bus
340         * int32_t --> 4 x uint8_t  (then, at the receiver size we need to do the same
341         */
342         tx_buffer[0] = (uint8_t)(_radioData.PressureTx);
343         tx_buffer[1] = (uint8_t)(_radioData.PressureTx>>8);
344         tx_buffer[2] = (uint8_t)(_radioData.PressureTx>>16);
345         tx_buffer[3] = (uint8_t)(_radioData.PressureTx>>24);
346         /* Create the bit mask for the ACCELEROMETER data to be transmitted on the CAN bus
347         * int32_t --> 4 x uint8_t  (then, at the receiver size we need to do the same
348         */
349         tx_buffer[4] = (uint8_t)(_radioData.AccelerationTx);
350         tx_buffer[5] = (uint8_t)(_radioData.AccelerationTx>>8);
351         tx_buffer[6] = (uint8_t)(_radioData.AccelerationTx>>16);
352         tx_buffer[7] = (uint8_t)(_radioData.AccelerationTx>>24);
353
354
```

```
355         send_data_to_CAN_bus();
356         }
357         }
358         #endif
```

In this application it was designed to use the extended CAN identifier since it is not known *a priori* which is the ID format adopted by the nodes connected to the bus, and which are the IDs which have been used yet. The use of the extended CAN identifier is profitable in such a situation since it won't generate issues in more possible situations. As an example, assume that a node adopting ID = 1 (the one selected for that application) is connected to the bus yet. If the node adopts the standard ID format, there will not be collision since the already present node win the arbitration. This happens since the SRR bit, which comes after the first 11 bits of ID, is dominant if the identifier is standard.

```
417         #if defined(MASTER_MODE) || defined(WIRED_SENSOR)
418         void send_data_to_CAN_bus()
419         {
420         digitalWrite(PA3, HIGH);
421         CAN_Transmit(0x0001,  CAN_Id_Extended, CAN_RTR_Data, 8, tx_buffer);  /* ID: 0b0000
                0100 1001 0110 */
422         delay(1);
423         digitalWrite(PA3, LOW);
424         }
425         #endif
```

6.8 Code Overview - Wired Sensor

This functionality is, for what concerns the code, a mix between the wireless sensor and the wireless receiver. Thus, refer to previous sections for considerations regarding the settings' selection.

Again, just the needed libraries are included.

```
36          /* Include the libraries, according to what is the functionality required */
37          #include "Wire.h"
38          #include <SPI.h>
39          #if defined(MASTER_MODE) || defined(WIRED_SENSOR)
40          #include <stm8s_can.h>
41          #endif

45          #if defined(SLAVE_MODE) || defined(WIRED_SENSOR)
46          #include <Adafruit_BMP085.h>
47          #include "lis3dh-motion-detection.h"
48          #endif
```

Then, parameters needed by the application are defined.

```
51          #if defined(MASTER_MODE) || defined(WIRED_SENSOR)
52          /* Initialization of the parameters for CAN transmission */
53          unsigned long ID = 0;
54          unsigned char tx_buffer[0x08] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
                // payload = 8 bytes
55          #endif

86          #if defined(SLAVE_MODE) || defined(WIRED_SENSOR)
87          /* Here we have the configuration parameters for the accelerometer  */
88          uint16_t sampleRate = 100;  // HZ - Samples per second - 1, 10, 25, 50, 100, 200,
                400, 1600, 5000
89          uint8_t accelRange = 16;    // Accelerometer range = 2, 4, 8, 16g
```

```
90          LIS3DH myIMU(0x18);          // Default address is 0x18.
91          int16_t dataHighres = 0;     // Save the raw data before conversion to float
92
93
94          /* Configuration parameter for the barometer */
95          Adafruit_BMP085 bmp;
96
97          float AccAsseZ = 0;          // Measured acceleration
98          int32_t AccAsseZ_mg = 0;     // Acceleration in milli-g
99          float Temp_Meas = 0;         // Measured temperature (BMP180)
100         int32_t Press_Meas = 0;      // Measured pressure
101         #endif
```

Sensors and CAN transceiver are set-up; if an initialization fails, the execution is trapped within an infinite dummy loop.

```
201         #if defined(WIRED_SENSOR)
202         /* Initialization of the WIRED SENSOR */
203         void setup()
204         {
205         Serial.begin(115200);
206         delay(1000);
207
208         setup_CAN();
209
210         /* Initialization of the accelerometer */
211         if( myIMU.begin(sampleRate, 1, 1, 1, accelRange) != 0 )
212         {
213         Serial.print("Failed␣to␣initialize␣IMU.\n");
214         ErrorSignal_LIS_CAN();
215         }
216         else
217         {
218         Serial.print("IMU␣initialized.\n");
219         }
220
221         /* Detection threshold can be from 1 to 127 and depends on the Range
222         * chosen above, change it and test accordingly to your application
223         * Duration = timeDur x Seconds / sampleRate */
224         myIMU.intConf(INT_1, DET_MOVE, 13, 2);
225         myIMU.intConf(INT_2, DET_STOP, 13, 10);
226
227         uint8_t readData = 0;
228
229         // Confirm configuration:
230         myIMU.readRegister(&readData, LIS3DH_INT1_CFG);
231         myIMU.readRegister(&readData, LIS3DH_INT2_CFG);
232
233         // Get the ID:
234         myIMU.readRegister(&readData, LIS3DH_WHO_AM_I);
235         Serial.print("Who␣am␣I?␣0x");
236         Serial.println(readData, HEX);
237
238
239         /* Initialization of the barometer */
240         if (!bmp.begin())
241         {
242         Serial.println("Could␣not␣find␣a␣valid␣BMP180␣sensor,␣check␣wiring!");
243         ErrorSignal_BMP_CAN();
244         while (1);
245         }
246         }
247         #endif
248
249
```

```
250
251         #if defined(MASTER_MODE) || defined(WIRED_SENSOR)
252         void setup_CAN()
253         {
254         CAN_DeInit();
255
256         CAN_Init(CAN_MasterCtrl_NoAutoReTx,
257         CAN_Mode_Normal,
258         CAN_SynJumpWidth_1TimeQuantum,
259         CAN_BitSeg1_8TimeQuantum,
260         CAN_BitSeg2_7TimeQuantum,
261         4);
262
263         CAN_FilterInit(CAN_FilterNumber_0,
264         ENABLE,
265         CAN_FilterMode_IdMask/*CAN_FilterMode_IdList*/,
266         CAN_FilterScale_32Bit,
267         0x00,
268         0x00/*0x08*/,
269         0x00/*0x24*/,
270         0x00/*0x68*/,
271         0x00/*0x00*/,
272         0x00/*0x08*/,
273         0x00/*0x24*/,
274         0x00/*0x68*/);      //Check messages from ID: 0x1234  // Check Filter Register
              Table
275
276         CAN_ITConfig(CAN_IT_FMP,ENABLE);
277         }
278         #endif
```

Finally, the main program is executed. It consists of an infinite loop where the microcontroller acquires data from the barometric sensor and accelerometer, split them into 8-bit packets that are transmitted on the CAN bus. Each CAN message provides information related to a cycle of acquisition of data from sensors: half message contains the barometer data, the other half contains accelerometer data.

```
407         #if defined(WIRED_SENSOR)
408         /* Execution of the main program */
409         void loop()
410         {
411         /* Here perform the acquisition from the accelerometer */
412         dataHighres = 0;
413         myIMU.readRegisterInt16( &dataHighres, LIS3DH_OUT_X_L );
414         myIMU.readRegisterInt16( &dataHighres, LIS3DH_OUT_Z_L );
415         AccAsseZ = myIMU.axisAccel( Z );     // store the data in floating point
416         AccAsseZ_mg = (int32_t)(AccAsseZ * 1000);
417         // Print the result, i.e. acceleration in mg
418         Serial.print("Acc. axis Z in mg = ");
419         Serial.println(AccAsseZ_mg); // print with 4 digit after comma
420
421         /* Perform the acquisition from the barometer */
422         Temp_Meas = bmp.readTemperature();
423         Press_Meas = bmp.readPressure();
424         // Print the result: Temperature and Pressure reading
425         Serial.print("Temperature = ");
426         Serial.print(Temp_Meas);
427         Serial.println(" *C");
428         Serial.print("Pressure = ");
429         Serial.print(Press_Meas);
430         Serial.println(" Pa");
431         Serial.print("\n");
432
433         /* Create the bit mask for the PRESSURE data to be transmitted on the CAN bus
434         * int32_t --> 4 x uint8_t  (then, at the receiver size we need to do the same
```

```
435         */
436         tx_buffer[0] = (uint8_t)(Press_Meas);
437         tx_buffer[1] = (uint8_t)(Press_Meas>>8);
438         tx_buffer[2] = (uint8_t)(Press_Meas>>16);
439         tx_buffer[3] = (uint8_t)(Press_Meas>>24);
440         /* Create the bit mask for the ACCELEROMETER data to be transmitted on the CAN bus
441         * int32_t --> 4 x uint8_t  (then, at the receiver size we need to do the same
442         */
443         tx_buffer[4] = (uint8_t)(AccAsseZ_mg);
444         tx_buffer[5] = (uint8_t)(AccAsseZ_mg>>8);
445         tx_buffer[6] = (uint8_t)(AccAsseZ_mg>>16);
446         tx_buffer[7] = (uint8_t)(AccAsseZ_mg>>24);
447
448         send_data_to_CAN_bus();
449         }
450         #endif
451
452
453
454         #if defined(MASTER_MODE) || defined(WIRED_SENSOR)
455         void send_data_to_CAN_bus()
456         {
457         digitalWrite(PA3, HIGH);
458         CAN_Transmit(0x0001,  CAN_Id_Extended, CAN_RTR_Data, 8, tx_buffer);  /* ID: 0b0000
                0100 1001 0110 */
459         delay(1);
460         digitalWrite(PA3, LOW);
461         }
462         #endif
```

6.9 Future development

The code provided is working as expected, and has been extensively tested. However, some functionalities can be refined a bit in order to get more elegant solutions, or increase performances or data secrecy.

The Section contains some suggestion related to possible future implementations. Due to delivery requirements, the proposed solutions have not being implemented. Some of them would require a very small implementation effort, but it was decided to abort them since the delivery milestone did not allow to perform the required tests; instead, other may take more time.

- In wireless receiver, use interrupts instead of polling the flag to see if a new data has arrived.
- In wireless sensor, avoid to reconstruct the sensor data at the transmitter side, just send raw data and recombine them at the receiver side; in this way, the cycle time at transmitter side is speed up (few ms), and more data secrecy is obtained since if anyone is able to sniff the wireless data, he will get row data, and not an actual number
- Allow the wireless node to receive command on the CAN bus; when a certain message (to be decided) is received, we want to stop the data transmission. The wireless

receiver has to send a command over the radio channel to the wireless sensor, to stop it. The wireless sensor must go to a low power mode, thus saving power

7 Software Design - User Interface

7.1 Software Goals

The main goals for software design consist in:

- Providing a system for data monitoring that allows the user to easily visualize the acquired signals.
- Realizing an executable program that can be easily installed.

7.2 User Interface Architecture

The proposed solution consists in the realization of a user interface allowing the user to acquire CAN messages, unpack the data from the message, reconstruct the data and send it to the PC via USB. This is performed exploiting a 32-bit evaluation board available in our warehouse (STM32F407G-DISC1), entirely programmed by the team in order to act as a CAN sniffer. Received data are formatted in a string that is read through a NI LabView VI that allows to plot sensor data, battery voltage and error messages. NI LabView executable is employed to realize a graph, showing the fluctuation of acceleration and pressure as the time goes by. This solution requires a bit of implementation effort, but it is extremely cheaper than buying a commercial CAN sniffer (which costs up to 100£) and allows to have a fancy representation of data, much more readable than data streamed to the serial monitor as numbers.

In Figure 58 it is reported the block scheme of the user interface.

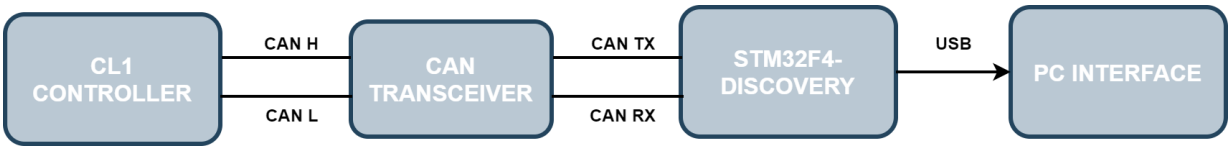


Figure 58: User interface architecture

7.2.1 STM32F407G-DISC1 Evaluation Board

STM32F407G-DISC1 has been chosen as platform to interface CL1 system with the PC for data monitoring.

The main reasons behind the selection of this platform:

- STM32F407G-DISC1 evaluation board has been used for another university project. No need to purchase it, leads to cost reduction;
- It has CAN interface support (only the CAN transceiver is needed);
- Easily configurable using STM32 CubeMX with HAL libraries.

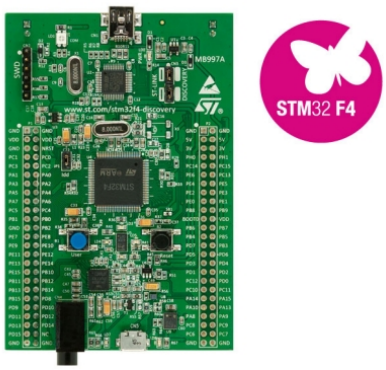


Figure 59: STM32F407G-DISC1

7.3 Firmware Development

7.3.1 CAN configuration

The function *static void MX_CAN1_Init(void)* is automatically generated by STM32 CubeMX based on the configuration set by the user.

```
/**
  * @brief CAN1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_CAN1_Init(void)
{

  hcan1.Instance = CAN1;
  hcan1.Init.Prescaler = 9;
  hcan1.Init.Mode = CAN_MODE_NORMAL;
  hcan1.Init.SyncJumpWidth = CAN_SJW_1TQ;
  hcan1.Init.TimeSeg1 = CAN_BS1_8TQ;
  hcan1.Init.TimeSeg2 = CAN_BS2_7TQ;
  hcan1.Init.TimeTriggeredMode = DISABLE;
  hcan1.Init.AutoBusOff = DISABLE;
  hcan1.Init.AutoWakeUp = DISABLE;
  hcan1.Init.AutoRetransmission = DISABLE;
  hcan1.Init.ReceiveFifoLocked = DISABLE;
  hcan1.Init.TransmitFifoPriority = DISABLE;

  if (HAL_CAN_Init(&hcan1) != HAL_OK)
  {
    Error_Handler();
  }

}
```

The main configuration is done in according to the structure of the CAN message used in the CL1 firmware. In order to be able to both transmit and receive data using the CAN line, the operation mode is CAN_MODE_NORMAL. In addition, the prescaler has been set to a value equal to 9, in order to match the bit timing selected in the CL1 board settings.

CAN messages filtering
Due to the fact that the CAN network is not complex and only a node is connected,

all the CAN messages have to be received. So, a message filtering is not needed in this case. Eventually, for future applications the sniffer could be configured to filter unwanted messages and accept only the messages coming from the nodes we are interested in.

```
        /* Filter configuration - accept all data*/
        CAN_FilterConfStructure.FilterFIFOAssignment = 0;
        CAN_FilterConfStructure.FilterMode = CAN_FILTERMODE_IDMASK;
        CAN_FilterConfStructure.FilterScale = CAN_FILTERSCALE_32BIT;
        CAN_FilterConfStructure.FilterIdHigh = 0x0000;
        CAN_FilterConfStructure.FilterIdLow = 0x0000;
        CAN_FilterConfStructure.FilterMaskIdHigh = 0x0000;
        CAN_FilterConfStructure.FilterMaskIdLow = 0x0000;
        CAN_FilterConfStructure.FilterActivation = ENABLE;
```

7.3.2 USART/USB configuration

The discovery board STM32F407G-DISC1 offers both a simple serial USART peripheral to have communication with external devices, as well as a direct micro-USB port. The direct USB-to-USB communication has been considered more straightforward and easy to implement, thus after several testing it has been found to be not suitable. In fact, the activation of the CAN peripheral caused problems to the USB one, resulting in an impossibility of them to stream data at the same time. This has been probably caused by a shared memory area between the drivers of the peripherals. Consequently, the decision has converged to a USART communication, requiring then a simple USART-to-USB adapter, already belonging to the team, without any additional cost.

```
 /**
  * @brief USART2 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART2_UART_Init(void)
{

  huart2.Instance = USART2;
  huart2.Init.BaudRate = 115200;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;

  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }

}
```

7.3.3 Functional flow

- Data from sensors are sent from the CL1 board through the CAN line;
- STM32F407G-DISC1 receives the CAN message through a polling mechanism. When a CAN message arrives what happens is that the FMR0 flag is set to 1 (by hardware) and data can be read from the input buffer.

CAN message reception

```
1
2                   /* Receiving the message from the input FIFO */
3                   HAL_CAN_GetRxMessage(&hcan1, CAN_RX_FIFO0, &CAN_RxMessage,
                        receiveBuffer);
4
5                   /* Decoding the data and copying it into the corresponding variables
                         */
6                   pressureDataTemp = (int32_t) (receiveBuffer[0] + (((int32_t)
                        receiveBuffer[1]) << 8) + (((int32_t) receiveBuffer[2]) << 16) +
                         (((int32_t) receiveBuffer[3]) << 24));
7                   accelerometerData = (int32_t) (receiveBuffer[4] + (((int32_t)
                        receiveBuffer[5]) << 8) + (((int32_t) receiveBuffer[6]) << 16) +
                         (((int32_t) receiveBuffer[7]) << 24));
```

- CAN message is read and saved in *receiveBuffer*.
- *pressureDataTemp* and *accelerometerData* variables are use to store sensor data contained in the CAN message.

Data extracted from the CAN message are then formatted and sent to the PC in the following way:

```
1                   if(pressureDataTemp != pressureData)
2                   {
3                           pressureData = pressureDataTemp;
4                           itoa(accelerometerData, accString, 10);
5                           itoa(pressureData, pressureString, 10);
6                           debugPrint(&huart2, accString);
7                           debugPrint(&huart2, ",␣");
8                           debugPrint(&huart2, pressureString);
9                           debugPrint(&huart2, ";\n");
10                  }
```

The *if* condition allows to print to send only messages whose barometric pressure value is different from the previous sent one. This allowed to lighten a bit the data stream, thus considering that the pressure data has very high resolution, avoiding then data loss.
The function *debugPrint* has been defined and used in order to redirect the USART transmission to a string "printf-similar" one.

```
1   /**
2     * @brief Serial monitor print redirect from USART function
3     * @param None
4     * @retval None
5     */
6   void debugPrint(UART_HandleTypeDef *huart, char out[]){
7
8           HAL_UART_Transmit(huart, (uint8_t*) out, strlen(out), 10);
9
10  }
```

All the STM32F407G-DISC1 firmware can be find in Appendix 12.

7.4 Software Development

The executable program for data visualization has been realized using NI Labview. The main reason behind this choice is due to previous experience with this software, that allows to obtain a way to visualize the data simply realizing a program that:

- Open a connection with the USB port.
- Read data from the USB board.
- Data is manipulated to prepare for visualization.
- Generate a graph with accelerometer and barometric pressure sensor data.

7.4.1 LabView Block Diagram

The software performs the following actions:

- Open the USB communication port.
  Serial USB communication is configured before the opening. A cluster containing all configuration data (baud rate, data bits, parity, stop bits and flow control) is created and put as input to VISA Configure Serial Port block. Then the communication start with the block VISA Flush I/O Buffer.
- Read the received string and extrapolate acceleration and pressure data;
  The block VISA Read read from the serial port a string of a given format, *%lu,%lu,%ld,%ld;*
- Close the USB communication port;
  Before the program stops the serial port is closed using the block VISA Close.

Flat sequences for configuration and close phase are used in order to ensure that the part is executed in the desired moment.

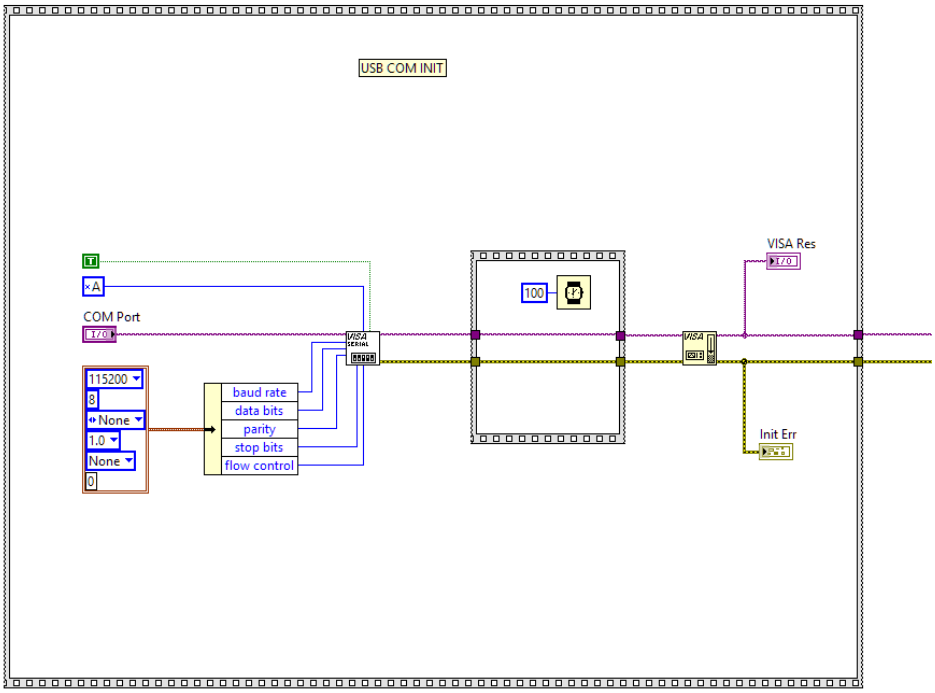


Figure 60: Opening of the USB port

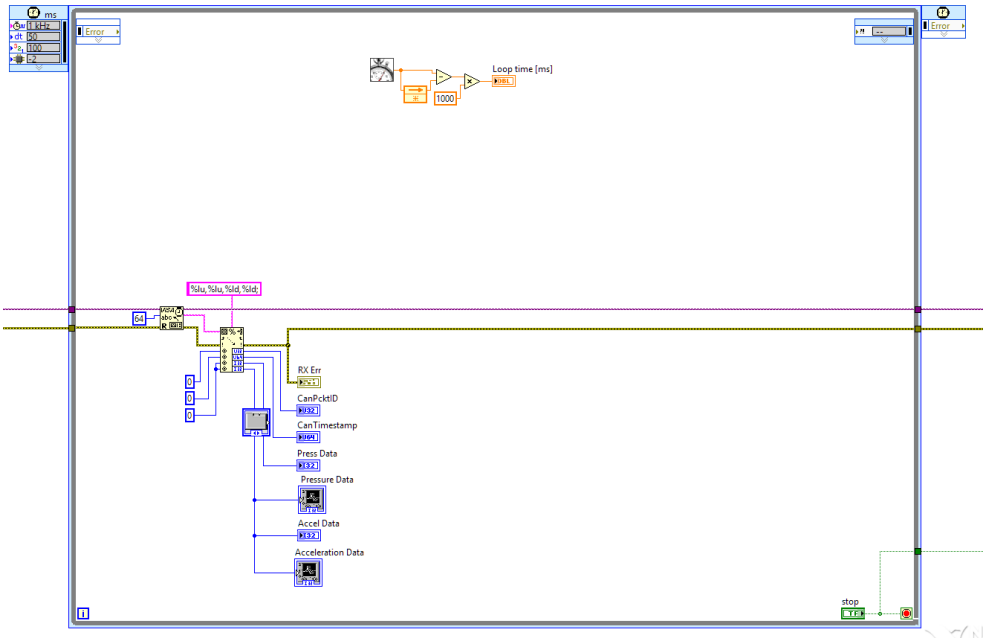


Figure 61: Data extrapolation from th received packet

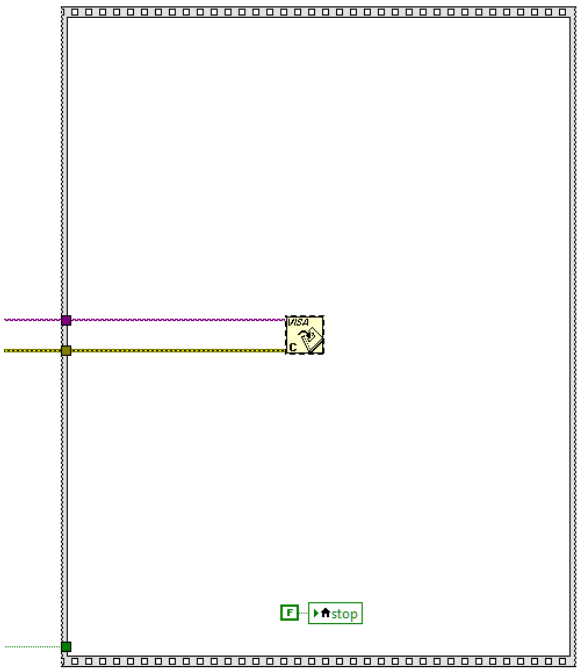


Figure 62: Closing of the USB port

7.4.2 LabView Front Panel

The front panel visualization allows to plot acceleration and pressure data.

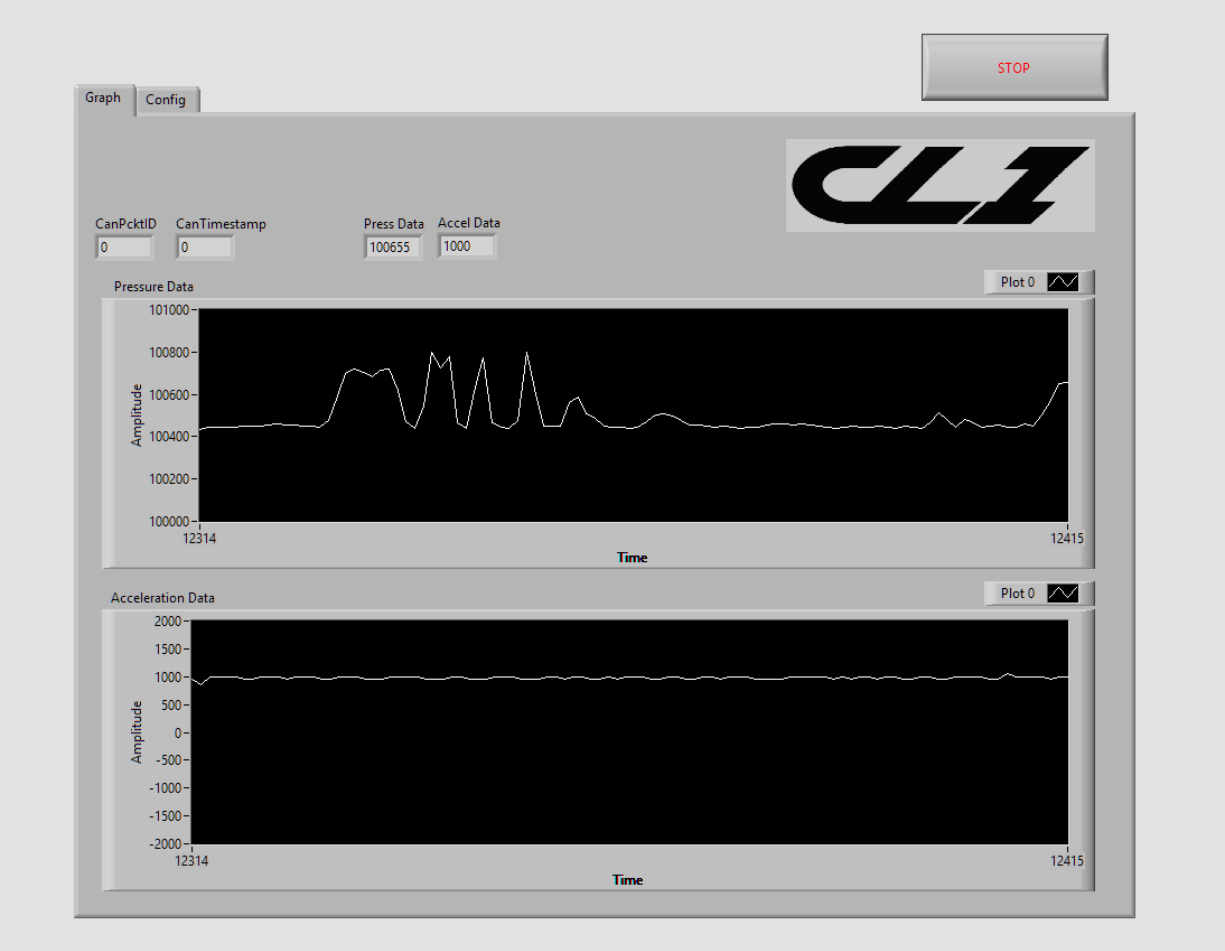


Figure 63: Front Panel for data visualization

7.5 User interface Future Development

- Add to the CAN message also the battery level for battery charge monitoring.
- In order to make the system more usable, an important development is related to bi-directional communication. If the executable program is able to start the data acquisition of the system, it will be possible to start and stop the system as the user wants.
  The high-level schematics of the required operations to pursue the goal is reported.

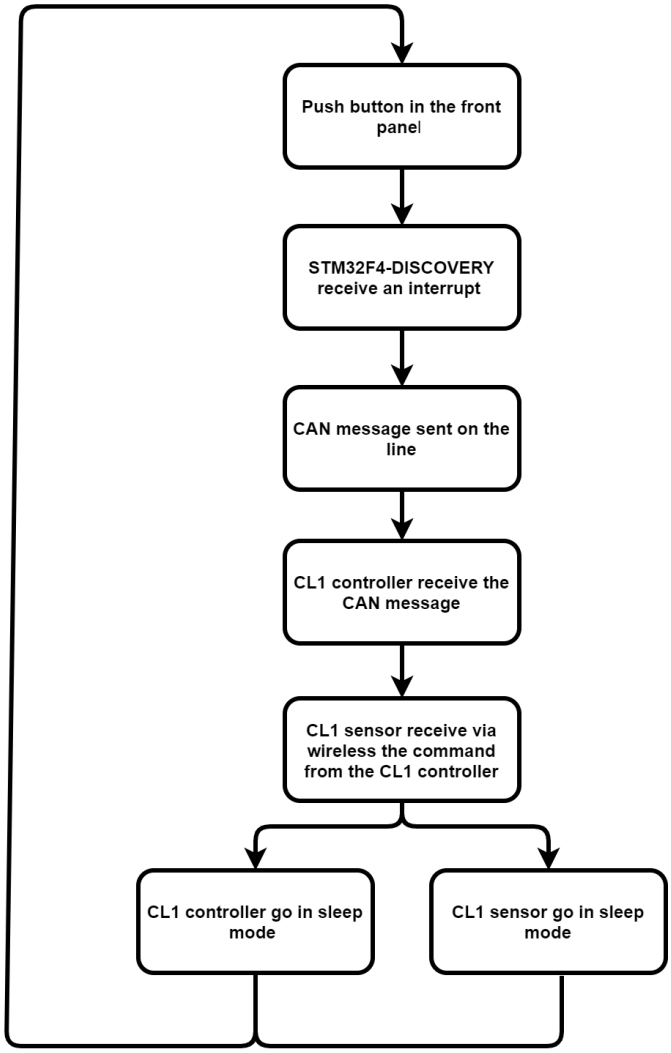


Figure 64: Bi-directional architecture block scheme

8 Mechanical Design

8.1 Goals

The main goals for mechanical design are listed in the following lines.

- Realize an enclosure for the PCB and the battery;
- Obtain an overall robust enclosure.
- Make the enclosure's impact on the wing's profile as small as possible.

8.2 CL1 Box

The design of the box has been done in order to contain the PCB and the battery cell. The dimension has been kept as small as possible, in particular the height is the most critical dimension because it can influence the air flow on the wing.

- Low weight;
- Small dimension;
- Opening on both the sides to let the air flowing;
- Opening on the bottom for battery connection;
- IPx protection is not provided;

The box for CL1 system is made by 3 parts:

- Main box for the PCB with holes for screws and openings for air flow;
- Additional box for the battery to be positioned below the main box;
- Cover for the box with openings for air flow.

8.2.1 Main Box

Problems Encountered
After the 3D printing of the box, it has been notice that there is a problem when the board is put into the box because of the dimension if the inductance. To solve this problem, part of the material around the screw hole has been removed.

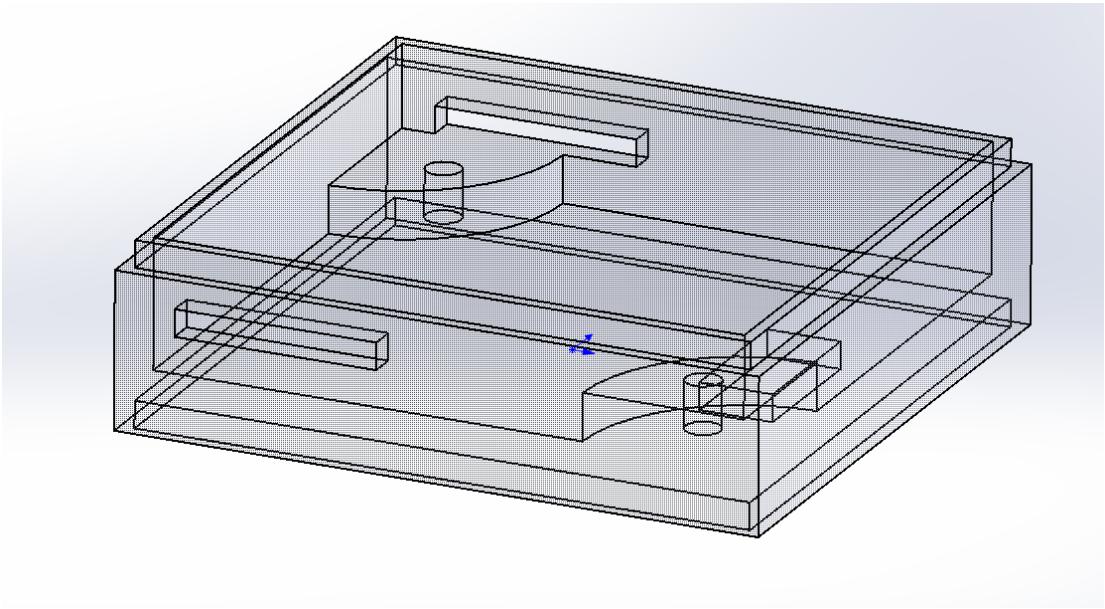


Figure 65: Main box

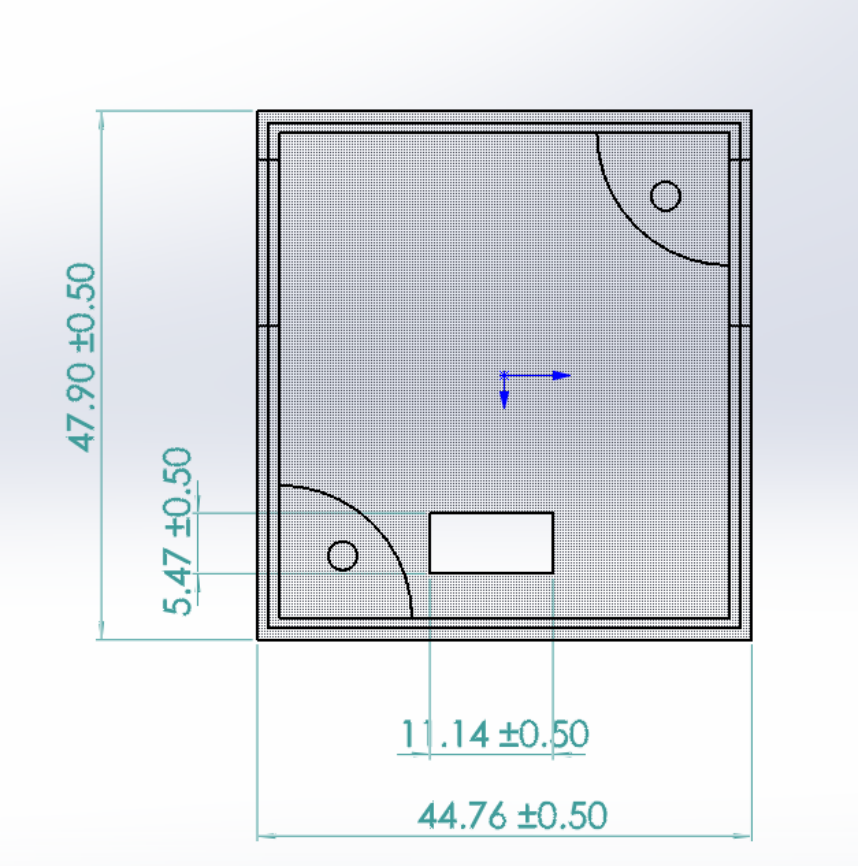


Figure 66: Top view of the main box

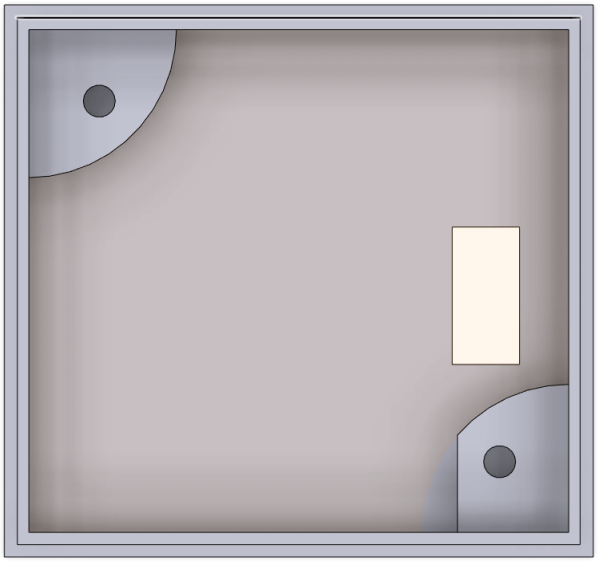


Figure 67: Top view of the modified main box

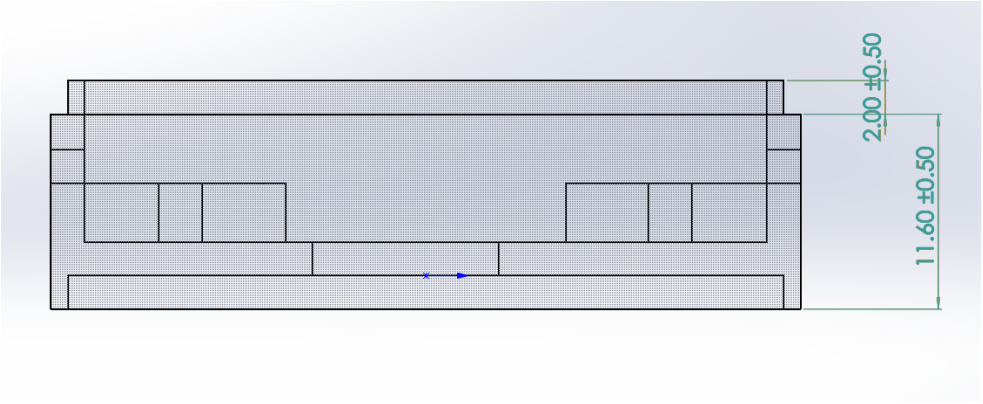


Figure 68: Side view of the main box

Battery box
The battery box is designed to hold the battery. It has no cover and it is directly connected to the main box.

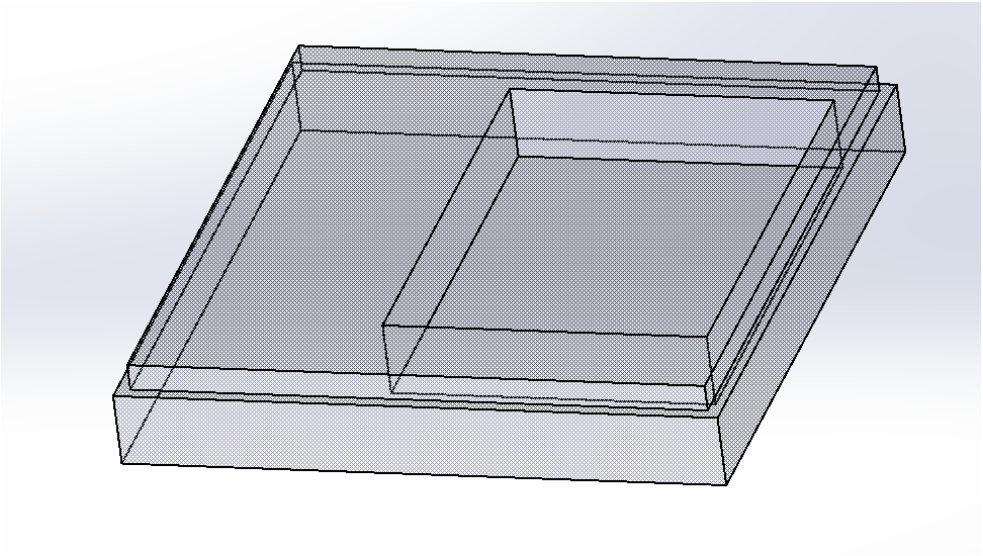


Figure 69: Top view of the battery box

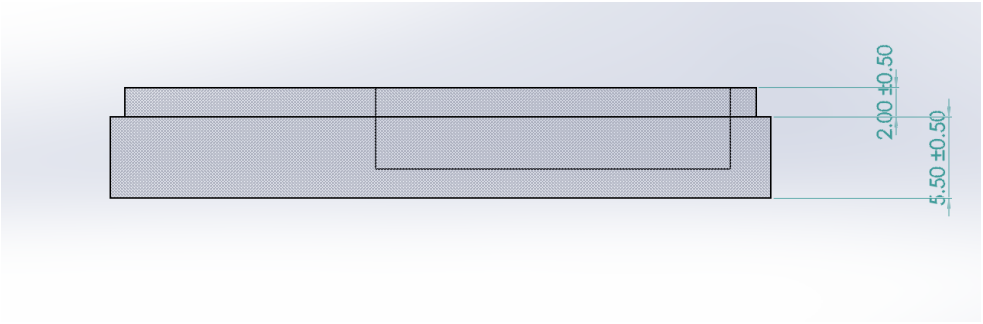


Figure 70: Side view of the main box

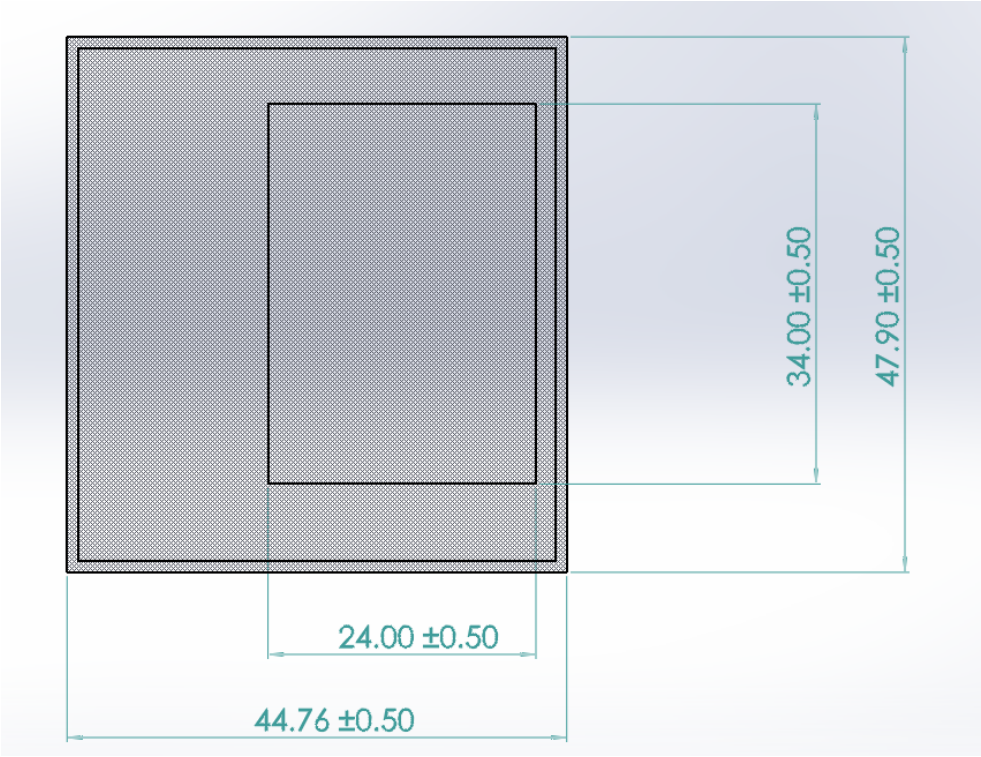


Figure 71: Side view of the main box

8.2.2 Cover

The cover is designed in order to have

- Aperture for air flow and for board programming;
- Easy connection with the rest of the box.

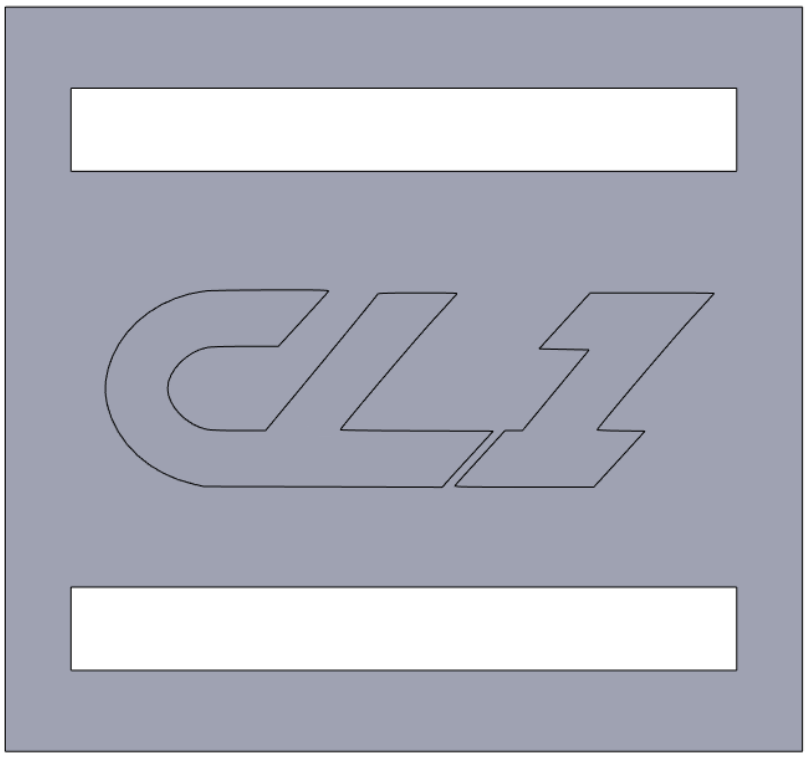


Figure 72: Assembly of the box

8.2.3 Assembly of the box

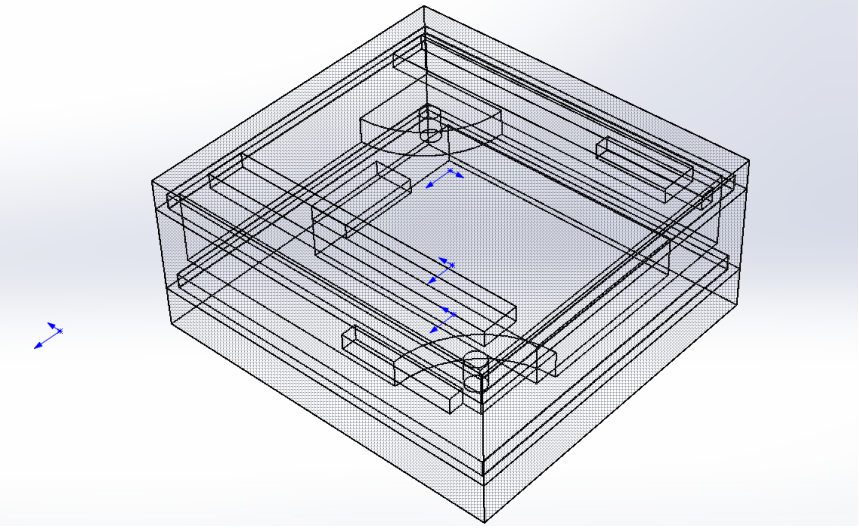


Figure 73: Assembly of the box

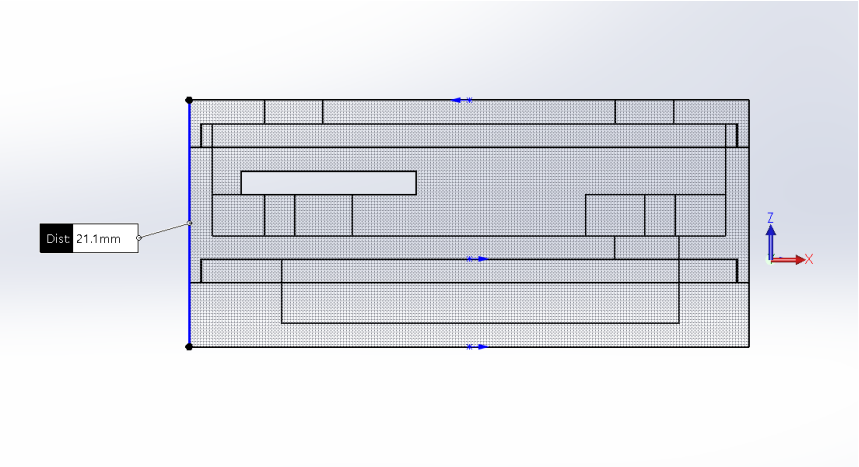


Figure 74: Side view of the assembly

Design Modification

When the battery has been delivered it result more thin than stated in the datasheet. For this reason it was possible to reduce the height of the box assembly by 2mm. The final height is 19.1 mm.

8.2.4 Box dimensions resume

Box	Dimension [mm]	Notes
Length of the box	39.9+4+4	4mm is the 2*thickness of the box
Width of the box	36.76+4+4	4mm is the 2*safety margin
High of the box	7.6+4+2	2mm is the 2*safety margin

Battery Box	Dimension [mm]	Notes
Length	39+4	4mm is the 2*thickness of the box
Width	19+4	4mm is the 2*safety margin
High	5+4	no cover

Corner hole(right)	Dimension [mm]
to right	3,617
to below	3,726
High of the corner hole	3
Radius of the corner hole	10
Radius of the screws hole	1,4

Corner hole(left)	Dimension [mm]
to left	3,726
to up	3,797
High of the corner hole	3
Radius of the corner hole	10
Radius of the screws hole	1,4

Battery hole	Dimension [mm]	Notes
to right	2,151	
to up	3,723	
length of the battery hole	7.798+4	4mm is the 2*safety margin
width of the battery hole	2.528+4	

Figure 75: Box dimensions

8.3 3D Printing

The designed box has been 3D printed, the reason of this choice is related to the availability of a 3D printer and the low cost of this solution respect others.

8.3.1 3D Printing Advantages

- Light weight parts;
- Reduction in costs;
- Realization of complex structures difficult to be realized with traditional methods;

- Reduction in waste material;
- Quite fast realization;

8.4 Material Evaluation

In order to evaluate the material that is suitable for the realization of the box, it has been considered different material that can be easily 3d printed at a really low cost. The main searched characteristics are resistance to high temperatures and robustness The battery cell is the most critical part of the system from the safety point of view, for this reason the realization of a box that is robust and that contain the battery without an excessive moving inside and that avoids stress on it. In the following figure are shown the mechanical properties of the considered materials. Both the one of which the first prototype is made (PLA) and the two material evaluated as suitable for a future realization (ABS and PET). In the following part different materials have been considered with their main advantages

	PLA	ABS	PETg
Recycled Content	55%	65%	67%
E-modulus	3120 MPa	1900 MPa	2020 MPa
Tensile Strength at Yield	38 MPa	44 MPa	50.4 MPa
Strain at Break	19.5%	9%	22.7%
Yield Strain	4.8%	2.8%	5.9%
Flexural Strength	N/A	56 MPa	69 MPa

Figure 76: Material mechanical properties

and disadvantages.

8.4.1 PLA

The material in which the prototype of the box is realized is PLA, for the reasons listed below

- Low cost material, suitable for a prototype box realization that is re-printed several times;
- No need of particular conditions during printing;

Advantages

- Strength and stiffness higher than other materials;
- Easy to be printed;
- Low cost.

Disadvantages

- It is not heat resistant, above 50 °C start melting and lose strength and stiffness properties.

This kind of material is not suitable for any kind of part that need robustness and in general is used in a hobbyist use case.

8.4.2 ABS

Advantages

- High heat eat resistant, above 200 °C;
- Easy to be printed but in a controlled environment;
- Low cost.

Disadvantages

- Strength and stiffness lower than PLA;
- Poor weathering resistance;
- Ordinary grades burn easily and continue to burn once the flame is removed;
- Scratches easily;
- Poor solvent resistance, particularly aromatic, ketones and esters;
- Can suffer from stress cracking in the presence of some greases.

8.4.3 PETg

Advantages

- Heat resistant: 90 °C;
- Durable and tough;
- Good chemical resistance;

- High impact strength;
- Difficult to scratch.
- Low cost.

Disadvantages

- Not easy to be printed, difficult set the 3D printer in order to produce a both durable and detailed product;
- Can be weakened by UV light;
- For constant outdoor exposure products, it can be expect to a strength decrease after a certain time;

Taking into account the materials characteristic, the material that seems more suitable for the intended application can be ABS or PETg. In section related to mechanical consideration, it can be found a static pressure analysis on the two materials, based on the result on the simulation further considerations can be done.

8.5 Mechanical Future Developments

The main goals for mechanical design are related with the improvement of the behaviour of the box.

- Modification in the box shape in order to obtain a better response to aerodynamic simulations and reduce the impact of the presence of the box on the wing, this can be eventually done reducing as possible the height of the box. Considering the height of the components chosen for the CL1 reduced size board it is possible to reduce the height of the new box of 2 mm, obtaining a total height of 17.1 mm for the reduced size box;
- In this table it is possible to find the possible dimension of the box for the reduced size board designed. It can be clearly noticed that the dimension of the new box is reduced, in fact the volume is 16569.9 mm^3 rather than the 32701.11 mm^3 of the CL1 box, with a 49.33% reduction in volume.
- Further studies on the behaviour of the box when forces are applied and how much force can the box withstand.
- Modification in the shape to increase the resistance to force of the cover of the box.
- Implement a strategy (kind of material use, shape of the box, additional material used as coating ecc) to achieve IPx protection, related to water, dust and other environmental factors.

9 Hardware Validation and Test

9.1 Validation Procedures Set-Up

In the following section are described all the tests performed or considered suitable to test the correct operation of the CL1 custom board.

- Each test procedure starts with "Test X_section" written in bold, where X indicates the number of the test related with that section of the circuit;
- All the test phases are listed in a bulleted list.

9.2 Validation Test Summary

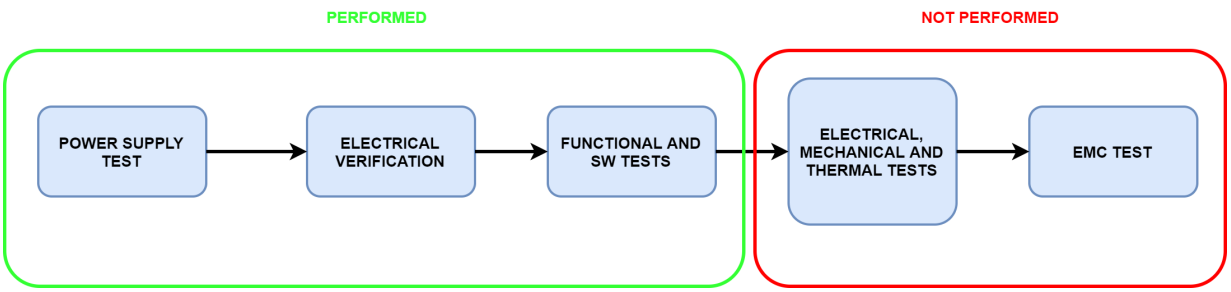


Figure 77: Test procedure summary

9.3 Electrical testing

Firstly, all the circuit is tested with the multimeter in order to see if all the components are soldered properly.

9.3.1 Test 1_ELECTRICAL

- Check the electrical connection using a multimeter;

Verify continuity. The test consists on verifying the proper connection between each component, each connected pad and each netlist. The test has to be repeated for all the components in the board. In case of small components, which pin are too close one to each other to be correctly singularly touched with the multimeter, it has to be used a different way of testing because of the dimension of the multimeter probe.

9.3.2 Test 2_ELECTRICAL

- Check the electrical connection using a probe of smaller dimension connected to the multimeter;

In the first phase this test is not performed, because of the need of a structure able to apply a pressure on the MCU pins without moving, avoiding the contact with other pins. For this reason, problems related to the uncorrect soldering of the MCU will arise in the next testing phases.

9.3.3 Test 3_ELECTRICAL

- Check the electrical connection using a probe of smaller dimension connected to the multimeter;

Verify the polarization of the components. The test consists on verifying the proper power supply of each individual component. The test must be repeated for all the components in the board. In case of small components, which pin are too close one to each other to be correctly singularly touched with the multimeter, it must be used a different way of testing because of the dimension of the multimeter probe.

9.4 Functional testing - Signals

Firstly, all the circuit is tested with the oscilloscope in order to see if all the components are functioning properly.

9.4.1 Test 1_SIGNAL

- Check the signal output connection using a probe of smaller dimension connected to the oscilloscope;

Validate the conformance of the component under test. Test each sub-circuit to determine whether the process complies with the requirement of the specification, technical standard and falls within the regulations in laboratory conditions.

9.4.2 Test 2_SIGNAL

- Check the signal output connection using a probe of smaller dimension connected to the oscilloscope and the FFT mode enabled;

Validate the signal quality. The test consists on verifying the quality of the small signal flow and flow throw the different layers before being processed. The test is realized by performing an FFT(Fast Fourier Transform) signal analysis to understand the distortion, signal strength, and ground noise. This information can later be taken into consideration to improve the design by applying different filters in different stages of the design.

9.5 Functional testing - SEPIC

Before the soldering of main components as MCU and sensors, the SEPIC converter correct operation has been tested.

9.5.1 Test 1_SEPIC

- Realize a load that simulates the presence of the other components connected to the supply line.
  The load is made by 3 through holes 220 Ω resistors connected in parallel, obtaining

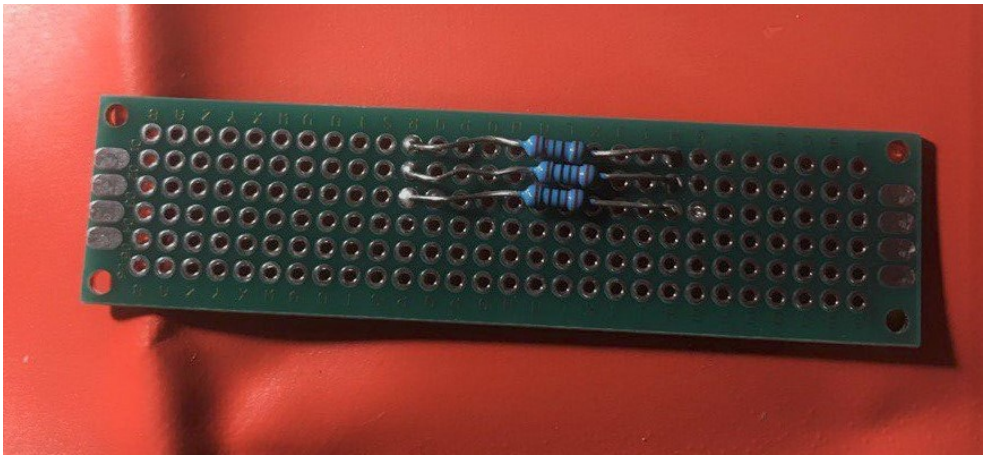


Figure 78: Load set-up

  an equivalent resistance of 73 Ω. It simulates the presence of the rest of the circuit in the worst current consumption condition.

- Connect the 73 Ω load to the board exploiting VDD and GND pin in the programming connector.

- Check the output voltage of the SEPIC converter with the multimeter.

The converter output functioning is verified because the output voltage measured is correct.
Now that the behaviour of the converter has been verified, all the components can be soldered.

9.5.2 Test 2_SEPIC

- Solder all the missing components.

- Test again the output voltage with the multimeter.

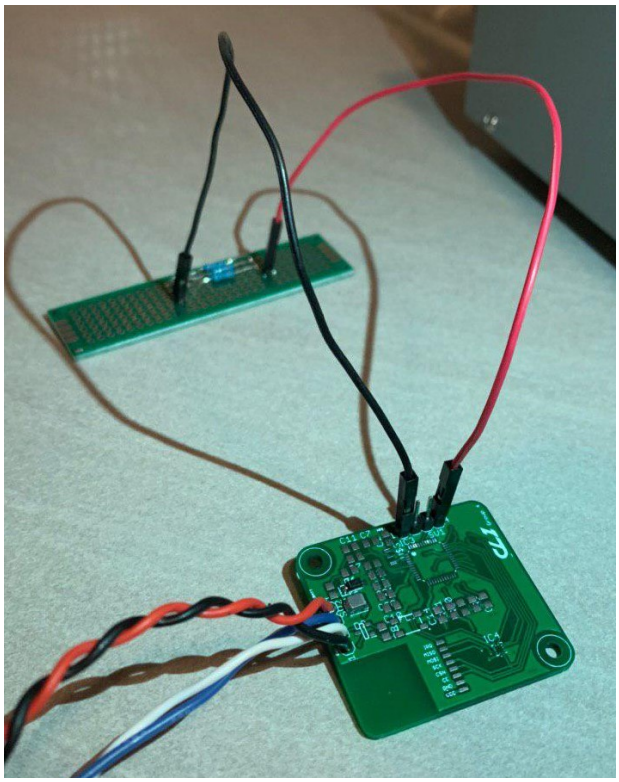


Figure 79: Board connected with the 70 Ω

Under this test condition with the power supply set at 12.3 V, the total current absorption is 0.01 A. The SEPIC output voltage measured with the multimeter is 3.309 (with a slight variation of ±0.02 V).

Encountered Problems During SEPIC Testing
It has been notice than the SEPIC converter turn off when the supply voltage is equal to 4.1 V, this is a problem when the battery is connected because of the fact that at 4.1 V the cell is around 90% SOC, so almost completely charged.
In order to understand which is the nature of the problem test with the multimeter where performed.

9.5.3 Test 3_SEPIC

- Decreasing the supply voltage from the power supply check at which supply voltage the SEPIC converter turns off;
- After the previous verification, the probe of the multimeter is positioned after the PTC fuse;

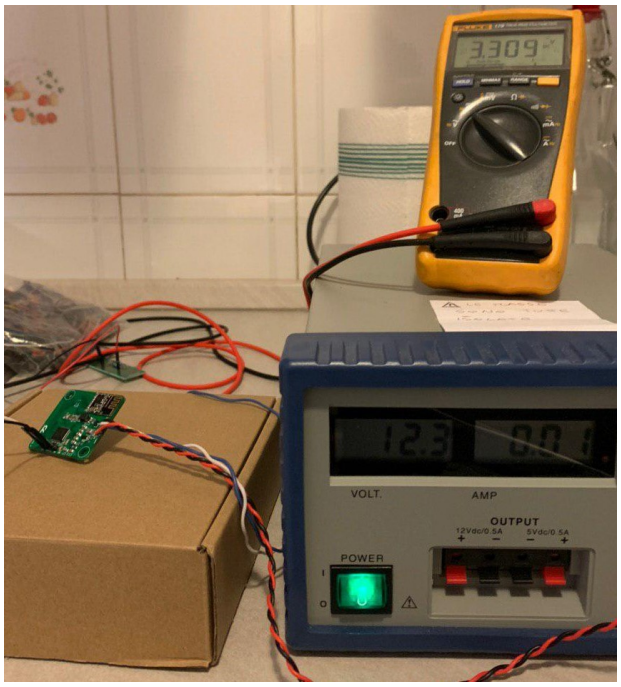


Figure 80: Load Test

What it has bees notice is that the voltage measured at the SEPIC converter input (after the PTC fuse) when the converter turns off is 3.06 V. This means that the mismatching recorded at the power supply side is due to a non zero-resistance of the PTC fuse.
In conclusion, the PTC fuse need to be replaced with a component that has a resistance close to zero, in order to not impact on the SEPIC turning off and make the system usable also when powered by a battery cell.

9.5.4 Test 4_SEPIC

- Set the power supply to 12 V;
- Connect the probe to the oscilloscope;
- Connect the probe to the V_DD pin in the programming connector;
- Decrease the voltage of the power supply until the SEPIC converter turning off;

This test is performed in order to check the real behaviour of the SEPIC converter during turning off.

9.6 Functional testing - MCU

In order to test the correct operation of the MCU, it has been done the simpler thing that can shown that the MCU is working.

9.6.1 Test 1_MCU

- Connection of the ST-LINK for STM8 to the programming header connector;
- Try to read the memory of the device using ST Visual Programmer.

The memory is accessible by the ST Visual Programmer, this means that the MCU is working.

9.6.2 Test 2_MCU

- After Test 1_MCU steps, flash into the memory a simple program;
- Debug the code and verify that the data read by internal ADC is correct.

In order to see if the MCU is programmable, it has been tried to read the supply voltage using an analog input pin. The test program can be find in Appendix 12.

Things that could be improved in the design : The connection of a LED to a GPIO output pin of the MCU allow to easily evaluate the correct operation by LED blinking.

Encountered Problems During MCU Testing
The voltage value on the analog pin in the sensor board was not correct. The value read in the variable ADC_converted_value was equal to 1.71 V instead of the voltage set from the power supply of 12 V.
Because of the reduced dimension of the MCU and the closeness between its pins that make it difficult to correctly solder by hands, the problem has been attributed to the soldering.
Problem Handling

- Use hot air on the MCU to allow the soldering of the pins.
- Add tin and flux on the MCU pins if it is needed.

9.7 Functional testing - CAN Line

In order to test the correct operation of the CAN line, a dummy message has been sent.

9.7.1 Test 1_CAN

- Check the voltage level at which is the line when nothing is transmitted.

The line, when no transmission is present, has to stay at 3.3 V/2 .

9.7.2 Test 2_CAN

- Send on the CAN line 0x0000000000000000;
- Using the oscilloscope visualize the CAN message.

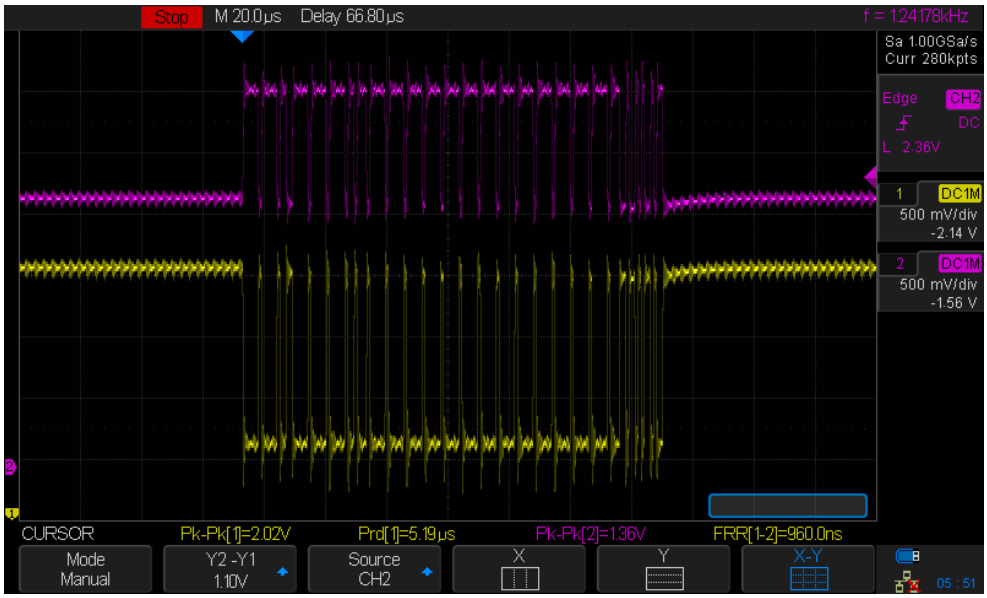


Figure 81: CAN Test

In Figure 81 is shown the message read using the oscilloscope.
The payload contain all zeros, every 5 zeros a one is present for bit stuffing.

9.8 Functional testing - Over-current and Polarity Inversion Protection

In order to test the correct operation of the protection circuit different test can be performed. This kind of test are only listed in this section but they have not been conducted because testing the system in limiting operating condition can determine the bad functioning or the damage of the components and additional items to substitute are not available.

- **Test 1_PROT - Overcurrent protection** If from the power supply a current is set, it is possible to simulate the behaviour when in the circuit flows to much current. The PTC has a exponential behaviour, when the current increases the resistance increase because of the temperature variation.
- **Test 2_PROT - Polarity inversion** To test the correct functioning of the polarity inversion protection the V_BATT and GND has to be swapped. As stated in the simulations in case of inversion of the two wires the Zener diode connected to the line goes in conduction and avoid the damage of the circuit components.

9.9 Functional testing - Additional Tests

9.9.1 Text 1_RES - Temperature Test

All the Electrical testing's including Test 1_Electrical, Test 2_Electrical, Test 3_Electrical and Test 4_Electrical and Signal Testing's including Test 1_Signal and Test 2_Signal, MUST be performed under stressed device to assess the performance and the failed sections of the circuit.

Test 1_Temperature high

- Check the circuit operation using a probe of smaller dimension connected to the oscilloscope;

To simulate the temperature test mand verify the proper component operation within the required high temperate requirement, we will take into consideration an upper with a hair blow dryer. Both tests require precise temperature measuring equipment, in this case a multimeter with a temperature probe.

Test 1_Temperature low

- Check the circuit operation using a probe of smaller dimension connected to the oscilloscope;

To simulate the temperature test mand verify the proper component operation within the required low temperate requirement, we will take into consideration and lower extreme temperature inside a freezer. Both tests require precise temperature measuring equipment, in this case a multimeter with a temperature probe.

9.9.2 Text 2_RES - Climatic Test

This test is an extended alternative to the temperature test yet adding the humidity and wind variables. To perform this test, it is compulsory to test in a climatic chamber that can vary different temperatures, humidity conditions and wind forces, recreated to different climatic profiles. These chambers also check the suitability of the CL1 under seals and adhesives. All the Electrical testing's including Test 1_Electrical, Test 2_Electrical, Test 3_Electrical and Test 4_Electrical and Signal Testing's including Test 1_Signal and Test

2_Signal, MUST be performed under stressed device to assess the performance and the failed sections of the circuit.

Test 1_Temperature high and low

- Check the circuit operation using a probe of smaller dimension connected to the oscilloscope;

To simulate the temperature test mand verify the proper component operation within the requirements the circuit must undergo a climatic chamber, to withstand temperatures from -65 °C to 150 °C, simultaneously verify the performance of the electronics at these extreme temperatures. This test must be done on the output of each component and each sub circuit block. Both tests require precise temperature measuring equipment, in this case a multimeter with a temperature probe.

Test 2_Humidity

- Check the circuit operation using a probe of smaller dimension connected to the oscilloscope;

This test analyzes the performance of the circuit under different stressed humidity percentages through levels water in the chamber. To perform this test, it is compulsory to test in a climatic chamber that can vary different humidity conditions recreated to different climatic profiles. These chambers also check the suitability of the CL1 under seals and adhesives.

Test 3_Wind

- Check the circuit operation using a probe of smaller dimension connected to the oscilloscope;

This test analyzes the performance of the circuit under different stressed wind conditions through fans. To perform this test, it is compulsory to test in a climatic chamber that can vary different wind forces conditions recreated to different climatic profiles. These chambers also check the suitability of the CL1 under seals and adhesives.

9.9.3 Text 3_RES - Vibration Test

All the Electrical testing's including Test 1_Electrical, Test 2_Electrical, Test 3_Electrical and Test 4_Electrical and Signal Testing's including Test 1_Signal and Test 2_Signal, must be performed under stressed device to assess the performance and the failed sections of the circuit.

Test 1_Vibration

- Check the circuit operation using a probe of smaller dimension connected to the oscilloscope;

The vibration test to be performed is done with a electro-acoustic shaker, that can shock the complete structure up to 50G a and bump testing in the case of automotive applications.

9.9.4 Test 4_RES - Accelerated Lifecycle test

The test of the material and structure can be performed on the CL1 with case included with a HALT test(Highly Accelerated Life Test), which works very efficiently to determine weaknesses in the design or in the manufacture by applying rapid 3D vibrations and thermal stress to the prototype using a specific testing equipment.

9.9.5 Text 5_RES - EMC Test

The pre-scan for EMI on the circuit consists on scanning for any radio frequency emission with the electronics hardware, not including the antenna and compare these measurements with the EMC regulations limits and the FIA regulations for EMI. If there is a noticeable disturbance that degrades the performance of the circuit or even if it may cause it to stop from functioning. Moreover, since there is a data transmission path, this may have an impact on the error rate, due to data loss. All the Electrical testing's including Test 1_Electrical, Test 2_Electrical, Test 3_Electrical and Test 4_Electrical and Signal Testing's including Test 1_Signal and Test 2_Signal, MUST be performed under stressed device to assess the performance and the failed sections of the circuit.
Test 1_EMC

- Check the circuit operation using a probe of smaller dimension connected to the oscilloscope;

This analysis is quite broad, taking into account that the complete design has to be separated into different possible failure sectors (software used, product system architecture, PCB design, layers of the circuit board, currents flowing in ground plane, parasitic inductances due to close via channels, switching frequency and many other factors. Designs in the automotive field tend to have a high percentage of failure and have slightly higher EMI test yields, and perhaps due to the deep inspection that goes into electronics for competition vehicles with their wide levels of design and features. Simultaneous to this a signal quality test must be performed. The test consists on verifying the quality of the small signal flow and flow throw the different layers before being processed. The test is realized by performing a FFT(Fast Fourier Transform) signal analysis to understand the distortion, signal strength, and ground noise.

9.9.6 Text 6_RES – Conformance Test

- Check the circuit operation using a probe of smaller dimension connected to the oscilloscope;

Test the circuit under real life conditions the complete CL1 must be placed in an operational vehicle and test its working performance. Validate the conformance of the component under test. Test each sub-circuit to determine whether the process complies with the requirement of the specification, technical standard and falls within the regulations in laboratory conditions.

9.10 Custom Board Current Consumption

9.10.1 RX Configuration Theoretical Consumption

Data for the theoretical analysis has been extrapolated from the datasheets current consumption informations.
Considering that the controller board is equipped in the following way:

Component	Power Mode	I_{max} [mA]	I_{typ} [mA]
STMAF5288TCY - MCU	Run - 16 MHz - FLASH MEM	14	7,4
BMP388 - Pressure Sensor	Forced Mode - High Resolution @50Hz	0,975	0,69
AIS1120SXTR - 1 Axis Accelerometer	Not Available	6	6
NRF24L01 - Wireless Module	RX 2Mbps	13,5	13,5
SN65HVD231QDRG4Q1 - CAN Transceiver	RECEIVER - All Devices Mode	17	10
	TOTAL	51,475	37,59

Figure 82: RX Current consumption

- NRF24L01+ module configured as receiver;
- CAN transceiver;
- No sensors;

So not considering in the calculation the current consumption due to the sensor, the result is

$$\text{RX\_current\_typ} = (37.59 - 0.69 - 6)\ \text{mA} = 30.9\ \text{mA}$$
$$\text{RX\_current\_max} = (51.475 - 0.975 - 6)\ \text{mA} = 44.5\ \text{mA}$$

9.10.2 TX Configuration Theoretical Consumption

Component	Power Mode	I_{max} [mA]	I_{typ} [mA]
STMAF5288TCY - MCU	Run - 16 MHz - FLASH MEM	14	7,4
BMP388 - Pressure Sensor	Forced Mode - High Resolution @50Hz	0,975	0,69
AIS1120SXTR - 1 Axis Accelerometer	Not Available	6	6
NRF24L01 - Wireless Module	TX @0dBm Output Power	11,3	11,3
SN65HVD231QDRG4Q1 - CAN Transceiver	RECEIVER - All Devices Mode	17	10
	TOTAL	49,275	35,39

Figure 83: TX Current consumption

Considering that the sensor board is equipped in the following way:

- Barometric pressure sensor and accelerometer;
- NRF24L01+ module configured as transmitter;

- No CAN transceiver.

So not considering in the calculation the current consumption due to CAN transceiver, the result is

$$\text{TX\_current\_typ} = (35.39 - 10)\ \text{mA} = 25.39\ \text{mA}$$
$$\text{TX\_current\_max} = (51.475 - 17)\ \text{mA} = 34.475\ \text{mA}$$

In case low power consumption strategies are not implemented during normal operation conditions this is the current consumption of the CL1 custom board. In the following paragraph is illustrated the current consumption in case all devices are put in sleep mode.

9.10.3 Sleep Mode Configuration Theoretical Consumption

- Barometric pressure sensor and accelerometer in sleep mode;
- NRF24L01+ module in sleep mode;
- CAN transceiver in sleep mode.
- MCU in two different low power modes (wait and halt modes).

So considering in the calculation the current consumption of all the devices in sleep mode, the result is

$$\text{Sleep\_current\_typ} = 1.1781\ \text{mA}$$
$$\text{Sleep\_current\_max} = 1.9281\ \text{mA} \quad \text{(with MCU in Wait Mode)}$$

The main contribution in the consumption is due to the MCU that if it is in wait mode, that means the CPU is stopped but peripherals are kept running, maintain a consumption higher than 1 mA.
While, if it is in Halt mode, that means CPU and peripheral clocks are stopped and the main voltage regulator is powered off, the overall consumption of the system decrease.

$$\text{Sleep\_current\_typ} = 0.0331\ \text{mA}$$
$$\text{Sleep\_current\_max} = 0.0631\ \text{mA} \quad \text{(with MCU in Halt Mode)}$$

9.10.4 Real Current Consumption

No strategies to measure the current consumption on the board are implemented. An estimation of the real current consumption derive from the current absorption when the board is supplied by the power supply. In this case due to the low accuracy of the power supply it is not provided a precise indication of the absorbed current.
What can be read from the power supply is 0.01 A at 12 V.

$$\text{current\_12} = 0.01\ \text{A} = 10\ \text{mA}$$
$$\text{power} = (0.01\ \text{A})*(12\ \text{V}) = 0.12\ \text{W}$$
$$\text{current\_3\_3} = \frac{0.12W}{3.3V} = 0.0367\ \text{A} = 36.7\ \text{mA}$$

The real value depends on how the rounding is performed in the instrument. But the value obtained from this simple calculation is in accordance with the theoretical one (for the controller board, RX configuration).

9.11 Thermal Analysis

Thermal simulation of the circuits has been performed in order to evaluate the heat rejection of the system. In order to do this, a simplified model of the CL1 custom board has been developed and deployed on a CAD software (Fusion 360).
The features of this simplified model are:

- Reduced number of components, just active components (SEPIC, MCU, sensors, CAN transceiver and wireless module) have been considered;
- Dummy parameters in order to make the computation easy, even if not exact they give good qualitative results;
- Both internal heat generation and convection are simulated.

The qualitative result of the simulation is shown in Figure 84. The blue color is used to indicate the lowest temperature, whereas the highest temperature is represented by the red. As the figure shows, the component that heats up the most, as expected, is the SEPIC converter, which is in charge of regulating the supply voltage. Heat rejection strategies have

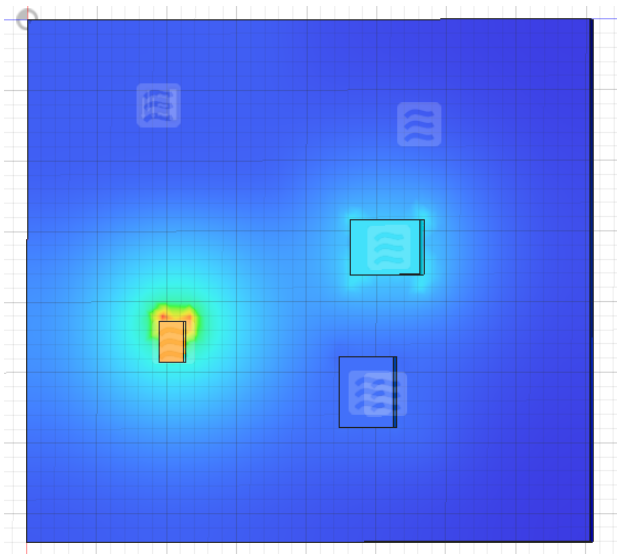


Figure 84: Qualitative results of thermal simulation

been applied in order to reduce the temperature of the SEPIC: according to the datasheet (that can be found in Appendix D, also see Figure 51) this component requires to be connected to metallic planes of specific sizes and shapes in order to increase its contact surface

for heat exchange and to dissipate in a more effective way. This is why these planes have been included in the PCB design, as shown in Figure 52.
This solution allows to effectively reduce the temperature of the SEPIC and of its surroundings, making the CL1 compliant with the specifications given by the customer. Heat sinks are not required in this particular application because of the cooling action provided by the air flowing inside the enclosure of the board, however, should they be needed because of possible project upgrades or expansions, the datasheet of the component provides information on how to dimension them and on where to place them.

9.12 Battery Life Estimation

The choice of the battery capacity derived from a trade-off, considering

- Battery cell size;
- Battery cell thickness;
- Battery cell weight;
- Battery cell capacity.

9.12.1 Battery Objectives

- Considering that the system is designed to be used during testing, the duration of the battery is not necessary to last for long time;
- The entire system has to be as lighter as possible, in order to not add too much weight on the wing;
- The maximum target dimension for the battery is equal to the size of the CL1 custom board, in order to do not increase the dimension of the box.
- The dimension that has been taken into account most is the height, because the will of reducing as much as possible the heght of the overall box.

9.12.2 Assumptions on Operation Mode and Battery Life

150 mAh is the total capacity of the battery a safety margin of 20% is considered. So a 120 mAh capacity can be used.
Considering the system operating in **normal operation mode**, so low power states are not used. The consumption of the sensor board (that is the one that has to be supplied by the battery) is in the worst case 34.475 mA.

$$\text{battery\_life\_h} = \frac{120mAh}{34.475mA} = 3.481 \text{ h} = 208.8 \text{ minutes}$$

The battery duration has been considered suitable for the application, because the operation time is small and when the battery is discharged it can be easily replaced.
While considering the system operating in **low power operation mode** for an estimated period of time equal to the 20% of the total operation time. The consumption of the sensor board is in the worst case 34.475 mA while operating and 0.0631 mA while it is in sleep.

9.13 FMEA

In the Failure Mode and Effect Analysis are taken into consideration the cause hat can lead to a system failure. Highlighting:

- Cause Factor;
- Failure Mode;
- Failure Cause;
- Effect on the system;
- How the failure can be detected;
- Difficulty of the detection;
- How the system can be recovered.

The document can be found in Appendix .III.

9.14 Hardware Tests Performed Resume

Test Name	Status	Notes
Test1_ELECTRICAL	Done	
Test2_ELECTRICAL	Done	
Test3_ELECTRICAL	Done	
Test1_SIGNAL	Not performed	
Test2_SIGNAL	Not performed	
Test1_SEPIC	Done	
Test2_SEPIC	Done	
Test3_SEPIC	Done	
Test4_SEPIC	Done	
Test1_MCU	Done	
Test2_MCU	Done	
Test1_CAN	Done	
Test2_CAN	Done	
Test1_PROT	Not performed	
Test2_PROT	Not performed	
Test1_RES	Not performed	
Test2_RES	Not performed	
Test3_RES	Not performed	
Test4_RES	Not performed	
Test5_RES	Not performed	

Table 25: Tests resume

All the tests that have been conducted are only used to verified the correct operation of each part of the board, only related to electrical connection. The tests for firmware functionalities are listed in the next section.

10 Firmware Validation

10.1 Firmware Tests

10.1.1 Test 1_CL1_FUNCTION

Due to the impossibility of acquiring data from the sensor in the CL1 custom board, in order to check the correct operation of the CL1 system the following procedure is performed.

- Save a known waveform in the sensor board MCU;
- Transmit the data to the controller board;
- Visualize the waveform using the user interface;
- Check the correctness of the data.

Test set-up
For this test the board are powered by the STM32F407G-DISC1 board 5V pin because is

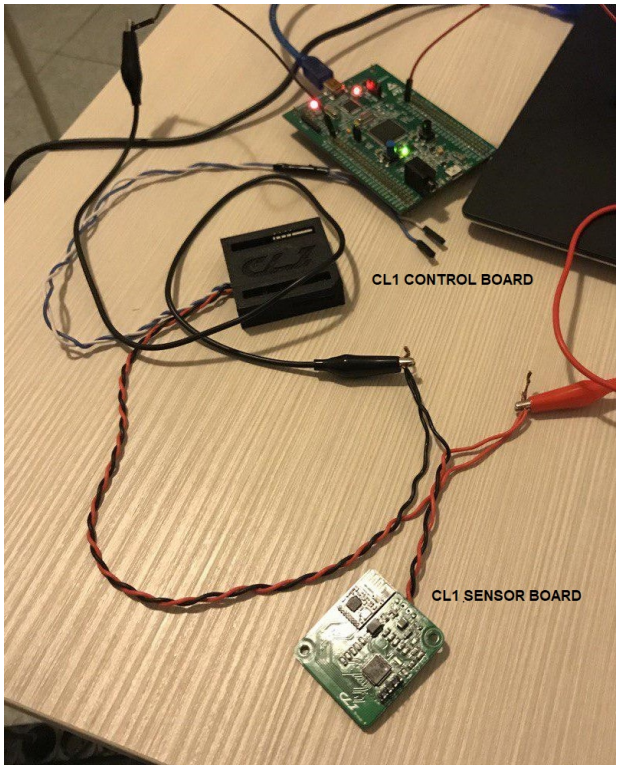


Figure 85: CL1 system set-up

the only supply source available when the test is performed.

Test result
A hardware problem is affecting the CL1 sensor board, it is not sure if it is regarding the MCU soldering (the MCU is working because it is possible to program it correctly and read the supply voltage) or the NRF24L01+ uncorrect functioning. Due to this problem the CL1 sensor board do not work neither if programmed as transmitter or as receiver.

Proposed Solutions
In order to solve the problem arised, the following solution is proposed.

- Substitute the NRF24L01+;
- Try if the the system is working.

Not performed, because of the need of additional components.

10.1.2 Test 2_CL1_FUNCTION

- Use CL1 controller as a receiver and for data transmission on the CAN line;
- Using a specific CL1 firmware able to count both the number of packet successfully received and transmissions failed;
- User interface used for acceleration and pressure data visualization;
- Test performed in both static and dynamic conditions.

Dynamic conditions means that the system is used on a car.

Test set-up

- Evaluation board as transmitter and data acquisition; CL1 sensor board is not used because of the problem encountered in the previous test and the lack of sensor mounted.
- CL1 control board used as wireless receiver and for CAN line transmission.
- CL1 Labview interface for data visualization;

Test result
The result in terms of performances is provided in the performance section. The test has shown that the system is correctly working also in non static conditions, considering all the environmental and stress factors that are present in a real situation.

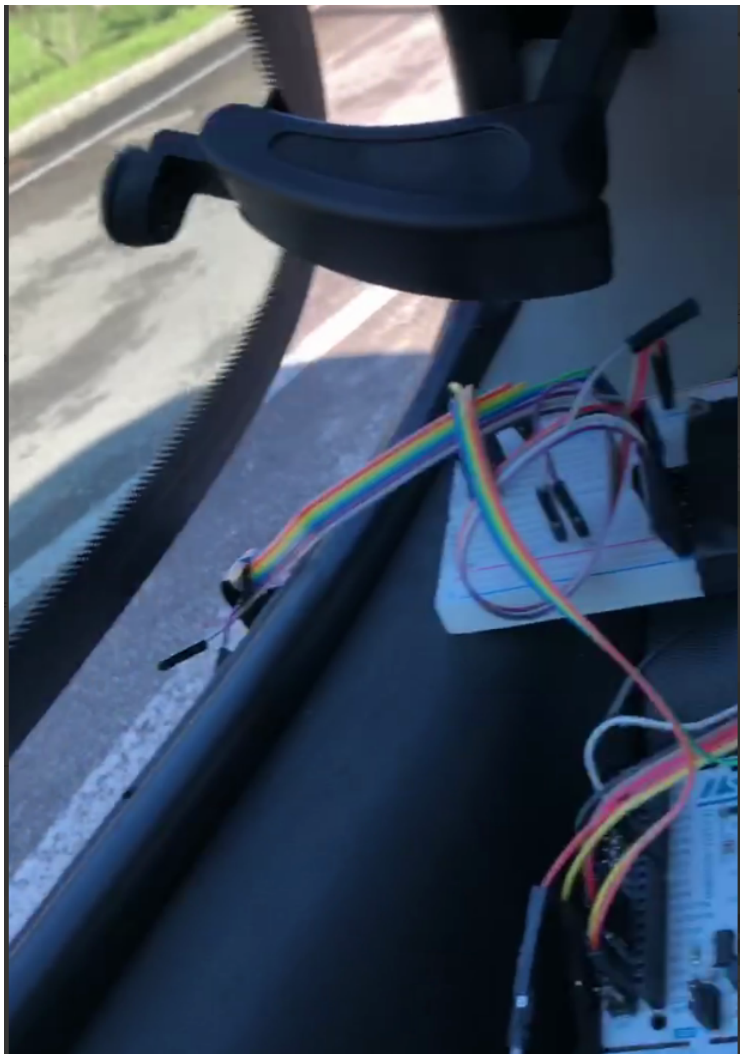


Figure 86: Evaluation Board Sensor Transmitter

10.2 User Interface Tests

10.2.1 Test 1_USER_FUNCTION

- Open LabView CL1 executable;
- Start the execution;
- Check the data correctness;
- Stop the execution and check if the program is correctly stopped without errors.

The data visualization is performed correctly, all the data are received. The reception take place every 50 ms (the value is counted by the LabView interface).

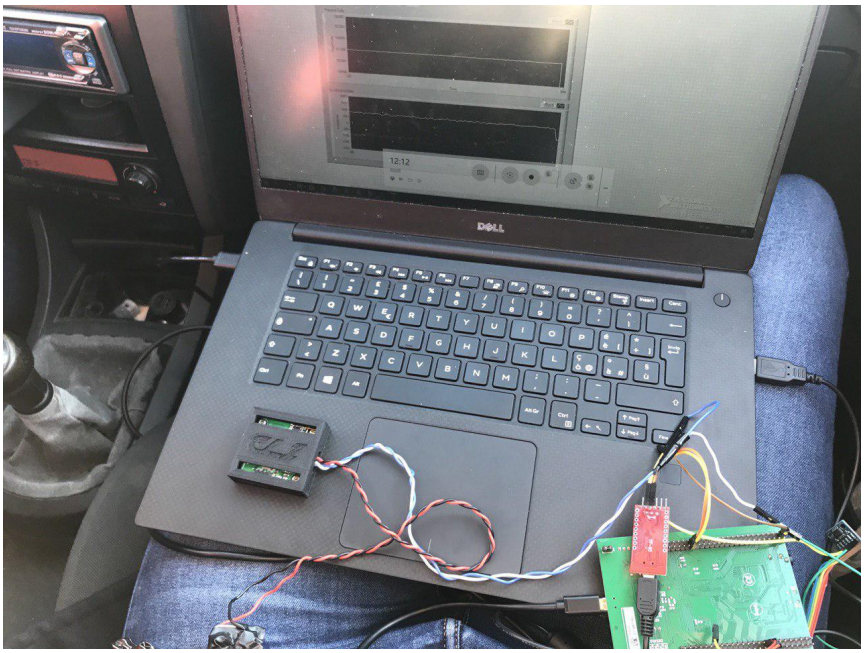


Figure 87: CL1 Control Board Receiver

NOTES: There is a problem when both LabView and Putty try to access to the serial port. When Putty is open the CL1 interface fail to read the data from the serial port and viceversa, when CL1 interface is open, Putty connection is not possible.

10.2.2 Test 2_USER_FUNCTION

- Open LabView CL1 executable;
- Start the execution;
- Simulate by modifying the CL1 firmware the presence of an error in the configuration;
- Check if the correct error CAN message is received.
- Stop the execution .

The table of messages containing a configuration error is the following.
If the initialization of some the components fails, it is possible to exploit the connection to the CAN bus to send an error message to privide a warning. Of course, this require the proper initialization of the CAN module. This error signaling strategy can be implemented in Wireless Receiver board and in Wired Sensor.

Error	Error Message on CAN bus	Applies to:
nRF initialization fail	0x0000000000000001	Wireless Receiver
Barometer initialization fail	0x0000000000000002	Wired Sensor
Accelerometer initialization fail	0x0000000000000003	Wired Sensor

Table 26: Error codes transmitted on the CAN bus

10.3 Firmware performances

The section is concerned with the validation of the firmware performances. In particular, the assessment focuses on the computation time, the average number of clock cycle needed to perform a full cycle of the main program, and the number of packet (transmitted over the wireless channel) that are lost during the working time.

The main program consists of

- A set-up phase that is performed only once;
- the controller runs an infinite loop performing sequentially the following operations:

  - Data acquisition from the accelerometer;

  - Data acquisition from the barometer;

  - Wireless transmission of the acquired values to the wireless node that is connected to the CAN bus.

Thus the time required to perform a full cycle is tightly related to the data reception rate at user's side.

10.3.1 Test1_PERFORMANCE

The assessment of the performance have been performed using the NUCLEO STM8S208RB board. The reason behind this choice is purely a practical one: the use of the evaluation board allow to get information directly using the serial monitor, while the custom *CL1* board, that is designed to be a final product, does not have PIN header for UART communication, nor a USB port, since are useless for the application and only add weight and cost unnecessarily. It is worth to point out that the result obtained would be exactly the same one could obtain using the *CL1* board, since the microcontroller architecture and the clock frequency adopted are identical.

Evaluation results obtained performing a static test on a test bench are reported in Table 27. The following paragraph contains a brief explanation of the test set-up, as well as some considerations on the obtained results.

The first test performed focus on the code developed for the evaluation board (refer to Appendix 12). A significant difference can be noticed between the two cases of failed

Code tested	Loop Execution Time	Clock cycles
Evaluation board code		
Successful transmission	54 ms	864000
Failed transmission	66 ms	1056000
CL1 board code		
Successful transmission	38 ms	608000
Failed transmission	49 ms	784000
No ACK	38 ms	608000

Table 27: Cycle time evaluation

transmission and successful transmission. In case the transmission is unsuccessful, a situation which was tested disconnecting the receiver node from the supply, the loop requires 12 ms more to execute. The reason is due to the fact that the code implemented requires an acknowledgement upon successful transmission; if no acknowledgement is received, the transmitter tries to send the same packet again up to 16 times, then drop the packet. The request for acknowledgement can as well be disabled; in this case, the time required by the application is almost the same one required in the case the transmission is successful at the first time.

10.3.2 Test2_PERFORMANCE

The second test was focused on assessing the performances of the code which is to be flashed to the *CL1* board. As it was explained in previous sections, the code is very similar to the one implemented on the evaluation board; the main difference resides in the fact that all unnecessary operations, mainly sending data to serial monitor, have been eliminated. As it can be seen, eliminating those operations 16 ms cycle time could be saved. The loop is executed once every 38 ms (if the transmission is successful at a first trial, or if no acknowledgement is required). Thus the data reception frequency is more than 26 Hz.

It is wort to point out that the results obtained have a resolution of 1 ms, thus the actual cycle time could be a little bit smaller or bigger than the reported 38 ms. Also the number of clock cycles is indicative, since it has been obtained multiplying the cycle time by the clock frequency (16 MHz). Table 28 reports the test results related to the packet

	Missed packets	Tot. num. of packets	Error rate
Static test (dist. 1.20 m)	0	47358	0.000%
Dynamic test	1454	18476	7.87%

Table 28: Error rate in comparison

reception rate. Two tests have been performed: a static test with the evaluation board on a test bench, and a dynamic test where the *CL1* boards were installed on a vehicle and tested in an urban environment, with normal traffic conditions.

The former test have been performed for a time span of 30 minutes, with the evaluation boards correctly configured and laid on the table, at a reciprocal distance of 1.20 meters. It is not surprisingly that no packets were lost in this situation: the test bench is not being exposed to particular interferences, except for the Wi-Fi router which transmit the same ISM band, thus showing that the solution is able to work despite to the presence of Wi-Fi networks in the test area. Another reason behind the 100% reception rate is the fact that the transmitter requires an acknowledgement from the receiver, and if the acknowledgement is not received the same packet is re-transmitted up to 16 times before dropping the packet; thus, if a temporary disturbance affects the radio channel, the packet is re-transmitted thus possibly avoiding a loss of information. The acknowledgement request can even be disabled if speed of execution is to be preferred over data reception rate.

The dynamic test was performed mounting the test setup on a mule vehicle, namely a Seat Ibiza. The evaluation board was used in the Wireless Sensor configuration, while the reception and the transmission on the CAN bus is performed using the custom *CL1* board. As it can be seen from Table 28, the error rate is not negligible in this situation. However, the high number of failed transmission is mainly due to a not reliable connection between the *CL1* board and the power supply: in some driving condition, i.e. upon crossing a road bump, a disconnection happened and several packets were lost until a human operator were able to restore the connection. A more reliable connection, e.g. exploiting nuts, would dramatically reduce the number of lost packets, thus the error rate.

10.4 Impact on CAN Bus load

This section briefly focuses on the possible increase in the CAN bus load due to the insertion of the packet used to transmit sensors' data. In fact, if the load on the bus overcome a certain threshold, collision starts happening and communication is seriously affected.

In order to estimate which could be the impact of the addition of a new packet transmitted every 35 ms, an instrument cluster emulator and the `can-util` software has been used. The working principle is straightforward: a virtual CAN line is set-up within the Ubuntu operating system; the instrument cluster generates traffic that flows on the virtual CAN line. Normally, the load on the bus in this condition is 47% (bitrate 500 kHz). Then, a *reasonable* data packet is injected every 35 ms: the chosen ID is not too small, and the data is composed of 64 bits, as the data it would be sent on the bus by the *CL1* board in a real-life situation. It has been observed that the load seldomly rise up to 48%.

Thus, it is possible to conclude that the addition of the packet on a moderately loaded bus is not a cause of overloading.

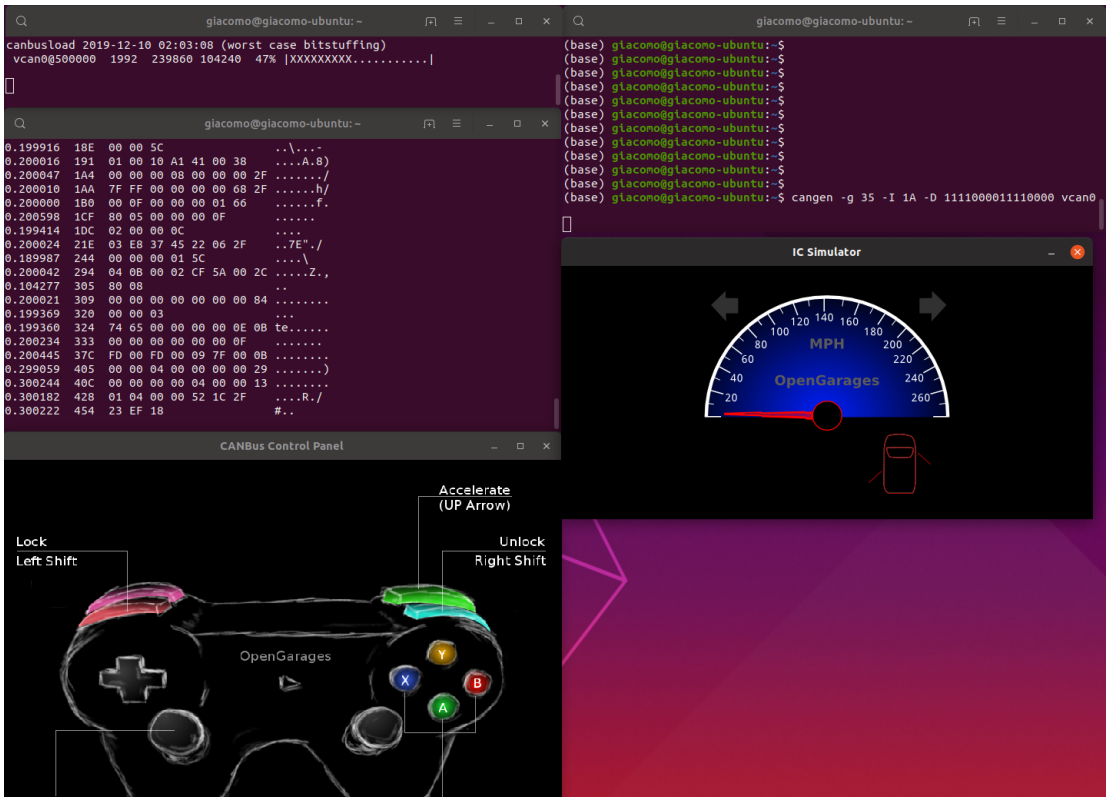


Figure 88: Cluster simulator used to evaluate the CAN bus load

10.5 Firmware Tests Performed Resume

Test Name	Status	Notes
Test1_FUNCTION	Done	Problems in CL1 sensor board
Test2_FUNCTION	Done	
Test1_USER_FUNCTION	Done	
Test2_USER_FUNCTION	Not performed	
Test1_PERFORMANCE	Done	
Test2_PERFORMANCE	Done	

Table 29: Tests resume

11 Mechanical Test

11.1 Static Force Analysis

11.1.1 Force Simulation Goals

- Understand which material is suitable for the box realization;
- Evaluate the robustness of the battery box in order to know if it's suitable for holding a battery cell.

11.1.2 Simulation Set-Up

In the following table are illustrated the mechanical properties of ABS plastic (**??**). While

Property	Value	Units
Elastic Modulus	2410000000	N/m^2
Poisson's Ratio	0.3897	N/A
Shear Modulus	862200000	N/m^2
Mass Density	1070	kg/m^3
Tensile Strength	40000000	N/m^2
Compressive Strength		N/m^2
Yield Strength		N/m^2
Thermal Expansion Coefficient		/K
Thermal Conductivity	0.2618	W/(m·K)

Figure 89: Configuration of ABS material properties

in Figure **??** the properties of PEG material.

Before conducting the simulation, constraints has to be defined. It has been supposed

Property	Value	Units
Elastic Modulus	2960000000	N/m^2
Poisson's Ratio	0.37	N/A
Shear Modulus		N/m^2
Mass Density	1420	kg/m^3
Tensile Strength	57300000	N/m^2
Compressive Strength	92900000	N/m^2
Yield Strength		N/m^2
Thermal Expansion Coefficient		/K
Thermal Conductivity	0.261	W/(m·K)

Figure 90: Configuration of PET material properties

that the box (as in the intended use positioning) is fixed in the bottom side. From the simulation point of view, this means that the reaction of the medium on which the box is fixed is considered.
In Figure 91 is represented the box with the constraint applied. The simulation is simplified

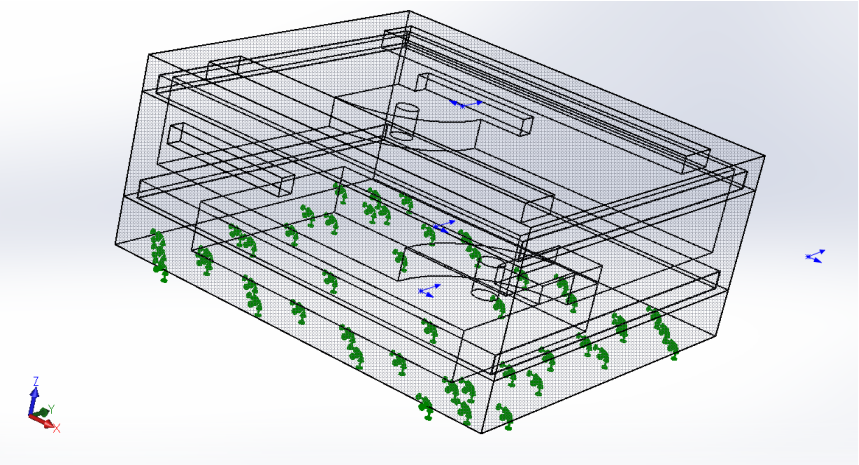


Figure 91: Constraints definition

so this is the only constraint that is consider.

11.1.3 Force Application Points

- Apply a uniformly distributed force to the top, the size of the force is 20 newtons;

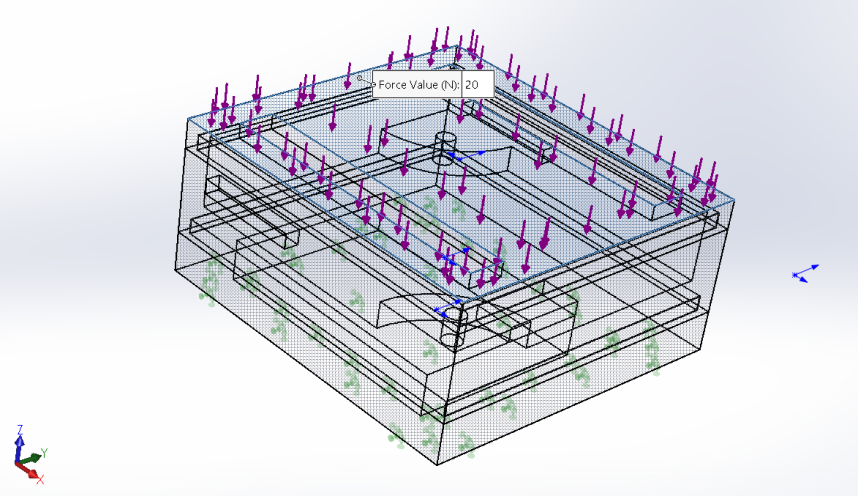


Figure 92: Force applied on the top

- Apply a uniformly distributed force on the side of the length direction, the size of the force is 100 newtons;

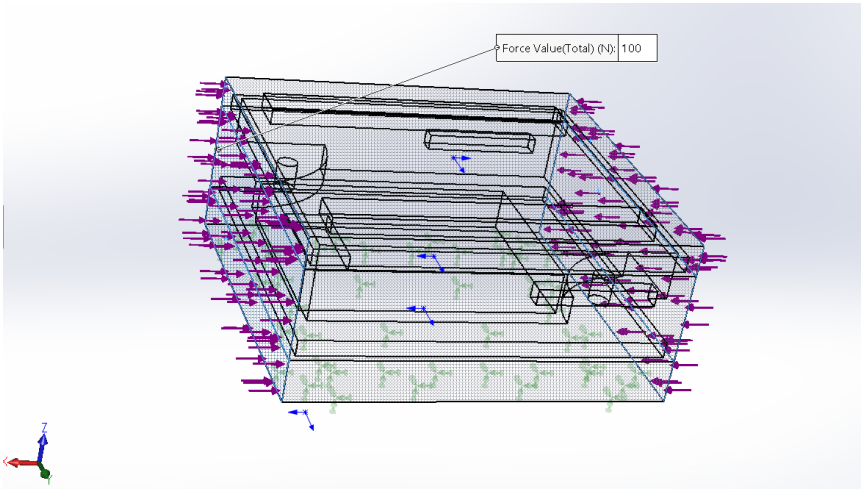


Figure 93: Constraints definition

- Apply a uniformly distributed force on the side of the width direction, the size of the force is 100 newtons.

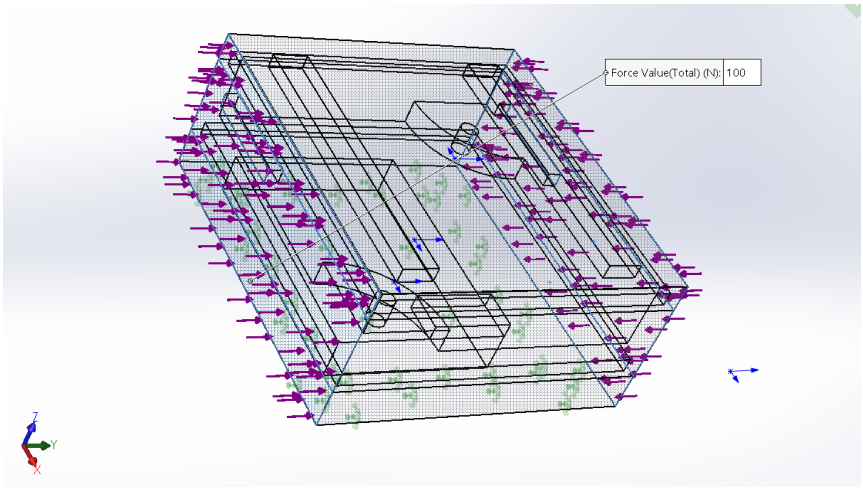


Figure 94: Constraints definition

11.1.4 Simulation Results

In this section are shown the simulation results making a comparison between the ABS plastic and PET plastic. What is highlighted are both stress and deformation, in case a 100 N force is applied.

- **ABS stress**

  Stress when a force on the top is applied.

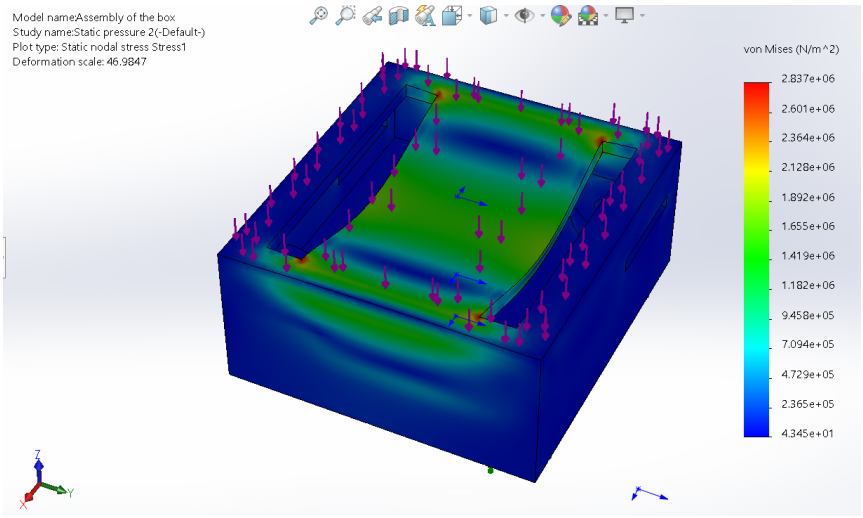


Figure 95: Top stress

Stress when a force on the length direction.

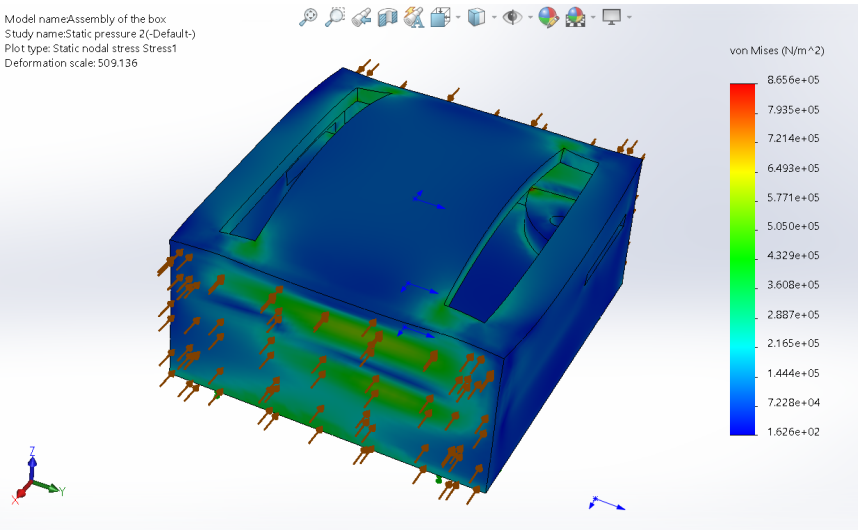


Figure 96: Length direction stress

Stress when a force on the width direction.

- **PET stress**

  Stress when a force on the top is applied.

  Stress when a force on the length direction.

  Stress when a force on the width direction.

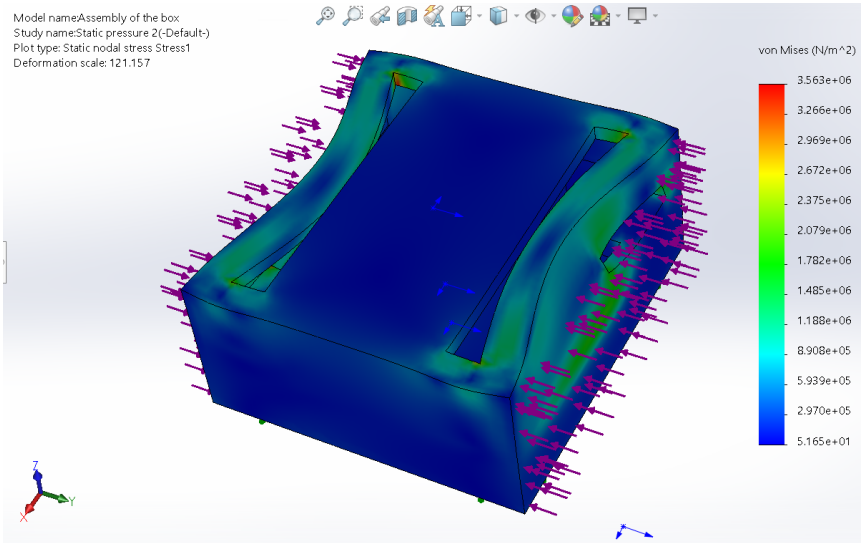


Figure 97: Width direction stress

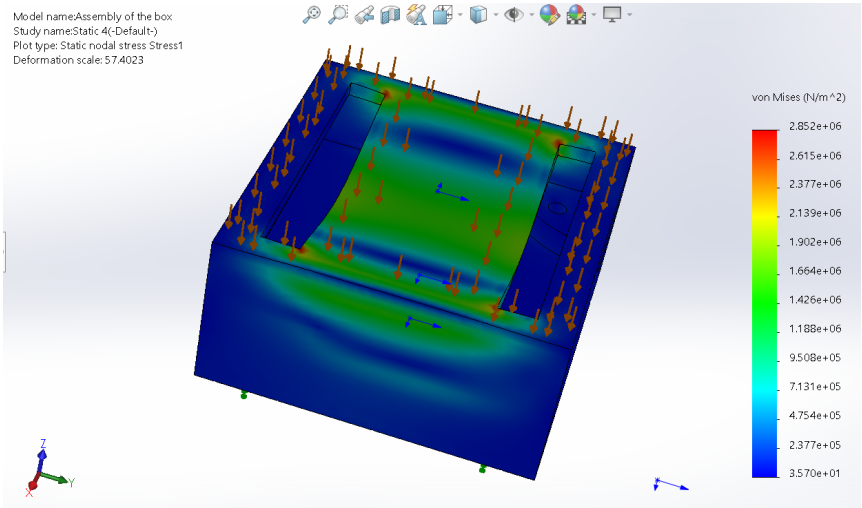


Figure 98: Top stress

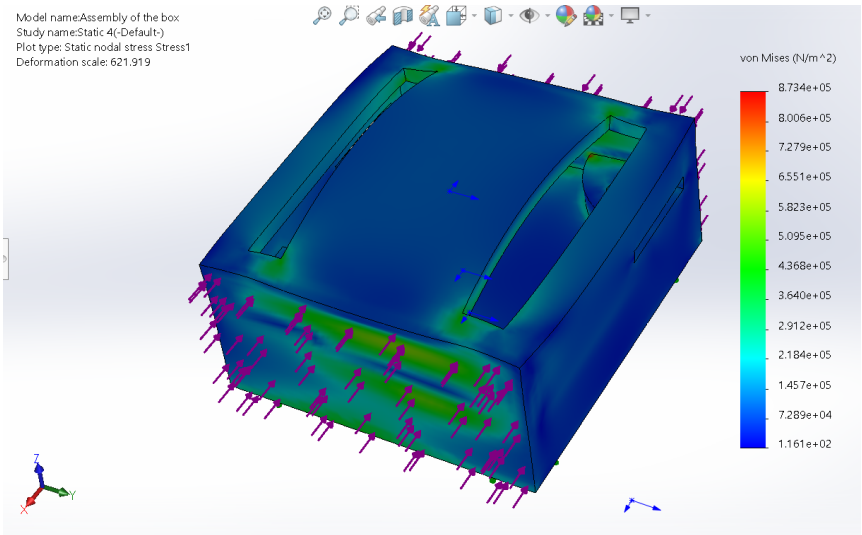


Figure 99: Length direction stress

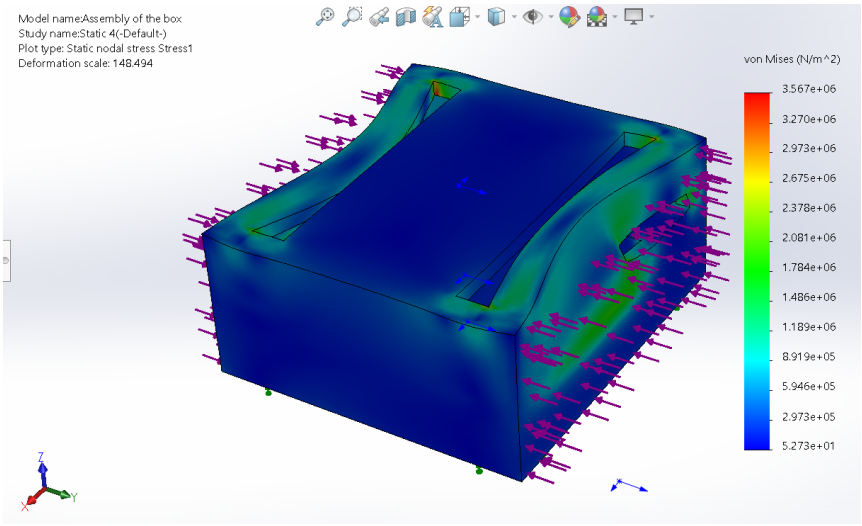


Figure 100: Width direction stress

- **ABS deformation**

  Deformation when a force on the top is applied.

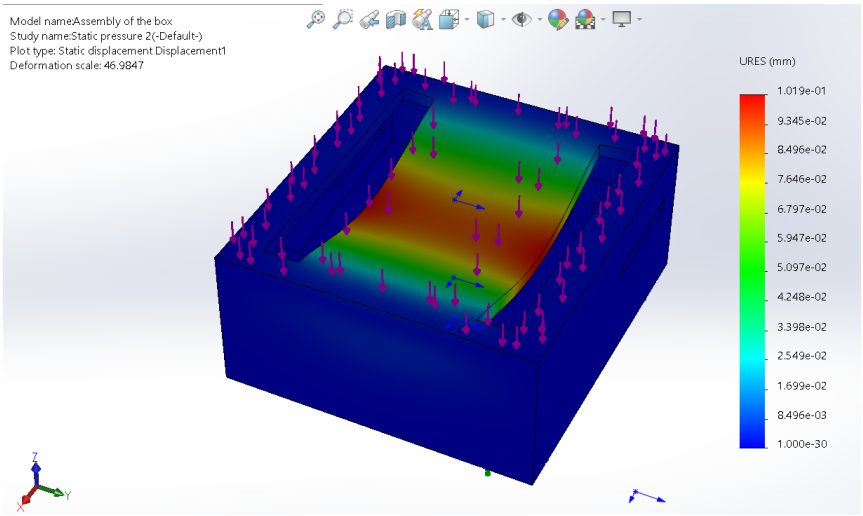


Figure 101: Top stress

Deformation when a force on the length direction.

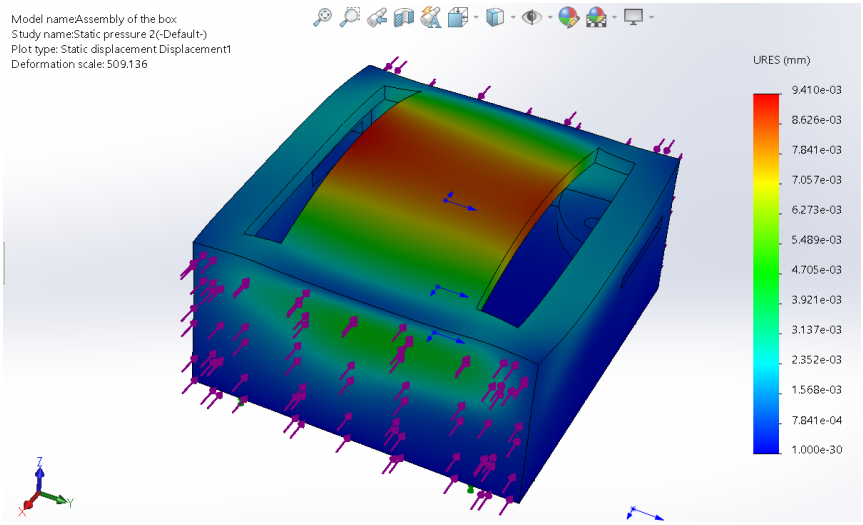


Figure 102: Length direction stress

Deformation when a force on the width direction.

- **PET deformation**

  Deformation when a force on the top is applied.

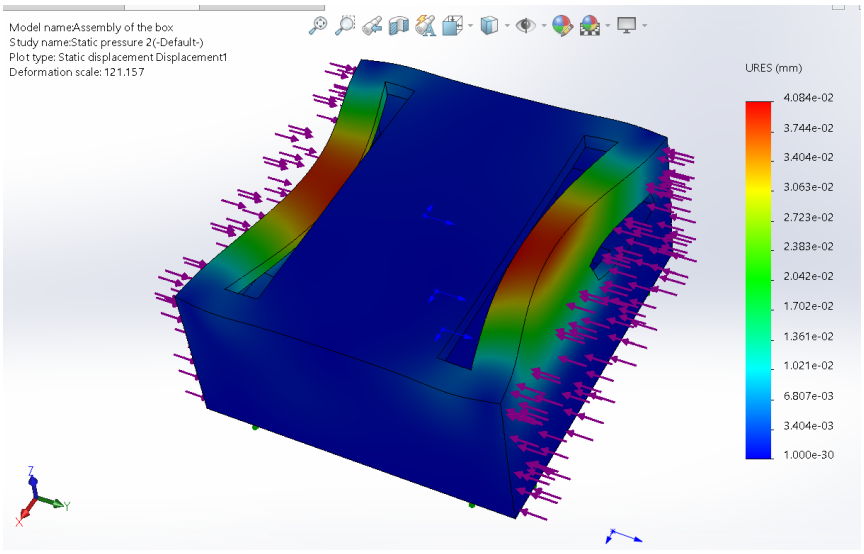


Figure 103: Width direction stress

Deformation when a force on the length direction.

Deformation when a force on the width direction.

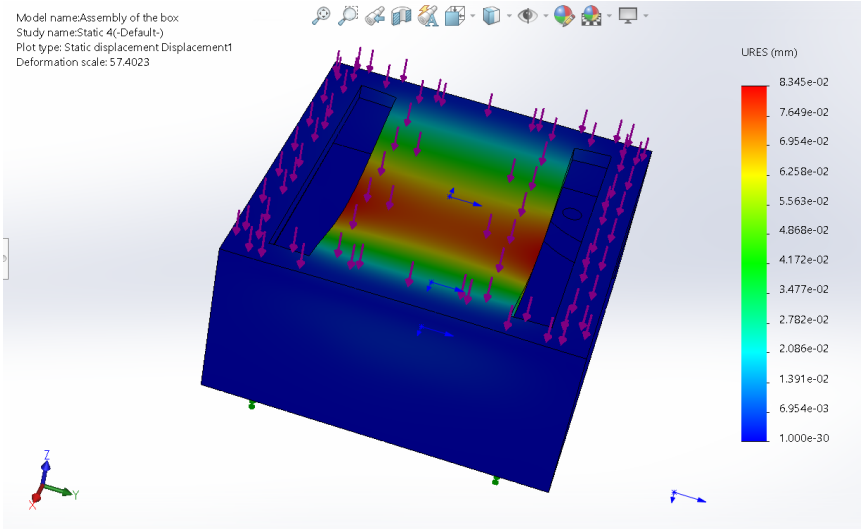


Figure 104: Top stress

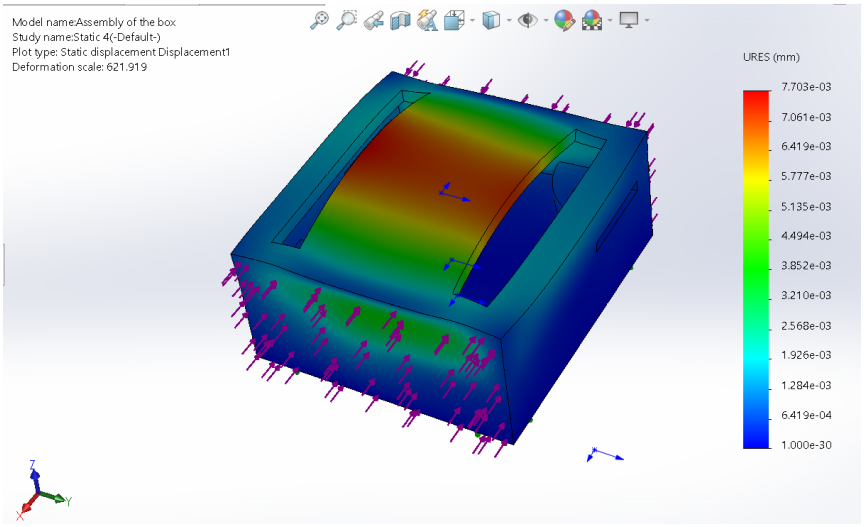


Figure 105: Length direction stress

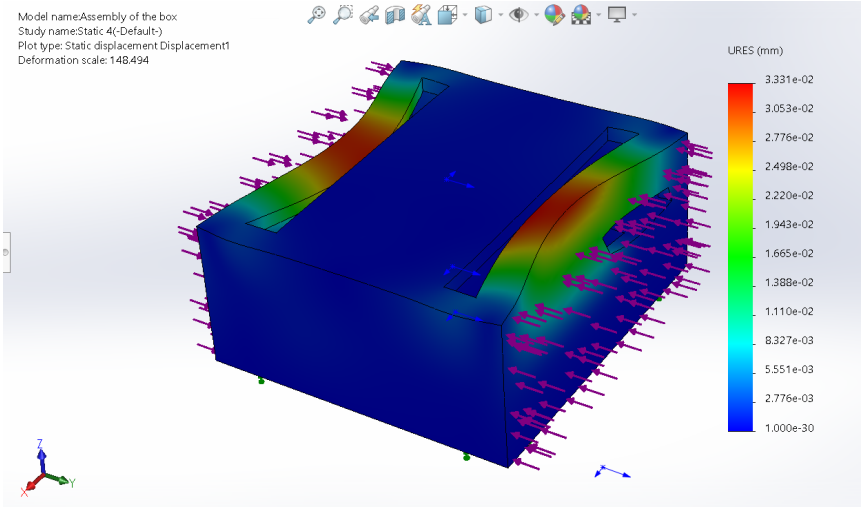


Figure 106: Width direction stress

Material		ABS PC		PET	
Force on the top					
	average of stress	7,84E+05	N/m^2	7,87E+05	N/m^2
	maximal of stress	2,84E+06	N/m^2	2,85E+06	N/m^2
	minimum of stress	9,57E+03	N/m^2	9,73E+03	N/m^2
	average of displacement	3,56E-02	mm	2,93E-02	mm
	maximal of displacement	1,02E-01	mm	8,34E-02	mm
	minimum of displacement	7,44E-05	mm	7,31E-05	mm
Force on the side of the length direction					
	average of stress	2,33E+05	N/m^2	2,34E+05	N/m^2
	maximal of stress	5,65E+05	N/m^2	5,66E+05	N/m^2
	minimum of stress	4,74E+04	N/m^2	4,77E+04	N/m^2
	average of displacement	1,66E-03	mm	1,36E-03	mm
	maximal of displacement	4,63E-03	mm	3,81E-03	mm
	minimum of displacement	1,00E-30	mm	1,00E-30	mm
Force on the side of the width direction					
	average of stress	5,51E+05	N/m^2	5,53E+05	N/m^2
	maximal of stress	1,73E+06	N/m^2	1,74E+06	N/m^2
	minimum of stress	4,94E+04	N/m^2	5,08E+04	N/m^2
	average of displacement	7,45E-03	mm	6,09E-03	mm
	maximal of displacement	3,90E-02	mm	3,18E-02	mm
	minimum of displacement	1,00E-30	mm	1,00E-30	mm

	Value	
Avg	6,09E-03	mm
Max	3,18E-02	mm
Min	1,00E-30	mm

Figure 107: Stress and deformation data comparison

11.1.5 Conclusions

- The stress related to the two material is almost comparable, because the stress mostly depend on the structure of the box. The displacement in case of PET material is smaller because the elastic module of the material is higher.
- The part of the box that is less robust is the cover, this is because of the presence of aperture for air flowing. After further studies on the proper shape of the box, the need of apertures and their positioning, it can be evaluated also a structure that has a more robust cover.
  This limitation subject also the effect of the force applied on the box side.
- The aim of this test is to ensure that the battery is safe from damaging. The force impact on the battery box is limited and so the battery box result robust.

11.2 Air Flow Simulations

In order to evaluate the behaviour of the wing while the system is mounted a simplified simulation has been set up. The aim is to see if the presence of the box influences the air flows on the rear wing, the model of the rear wing has been realized using dimension found on the internet that are compatible with the real ones.
In the Figure 109 and Figure 110, it is shown the rear wing both without the box and with the box mounted on it.

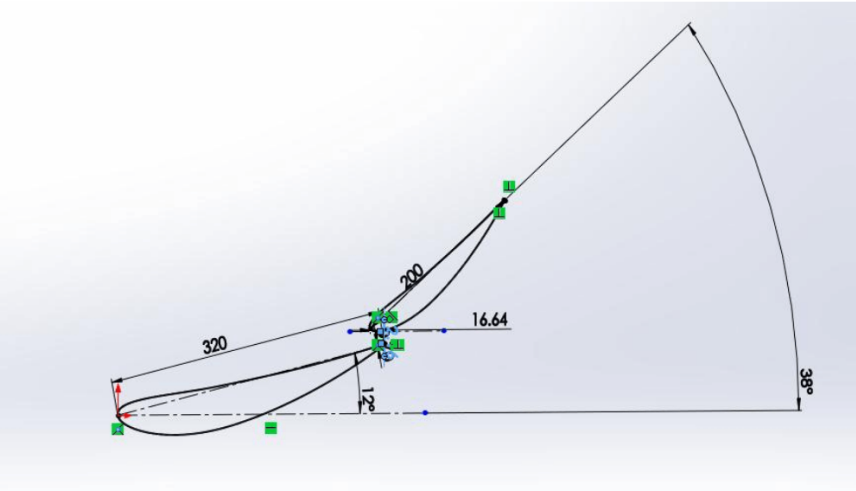


Figure 108: Side view of the wing

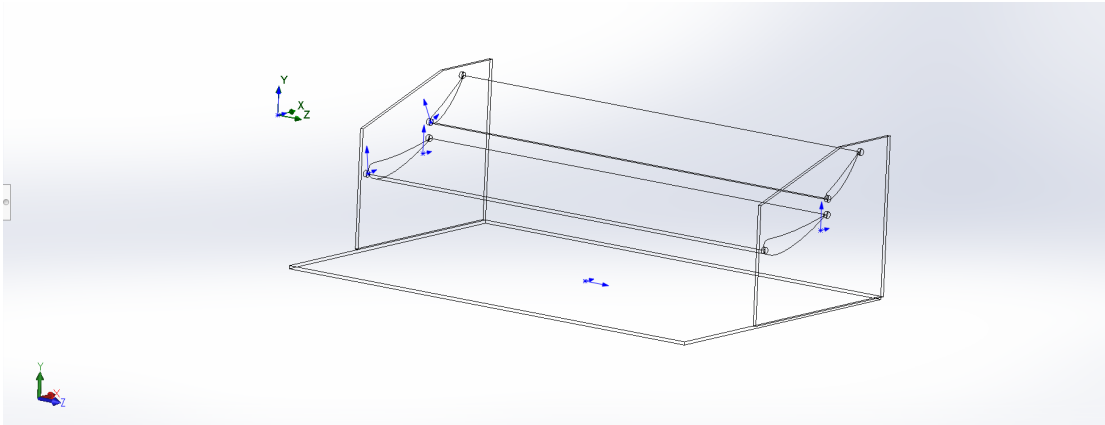


Figure 109: Rear wing model without the box

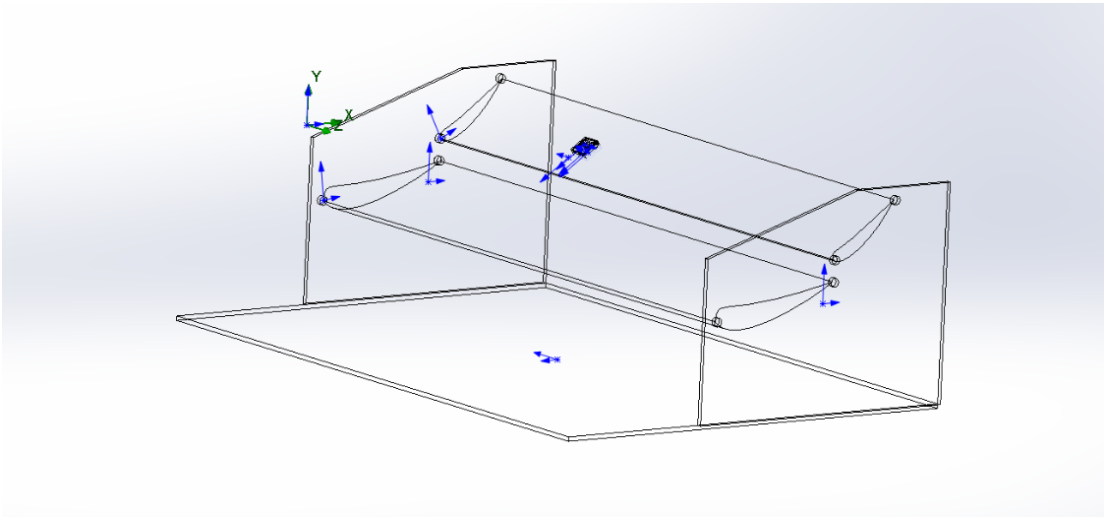


Figure 110: Rear wing model with the box

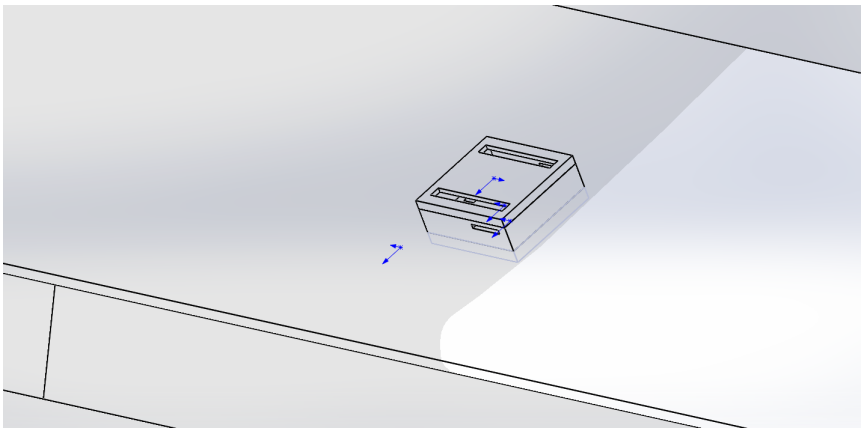


Figure 111: Box on the wing zoomed

11.2.1 Simulation Set-Up

The first step is to select the computational domain and divide cells, as shown in Figure 112.

The finer the cell is, the more accurate the simulation results are, but the amount of the calculation is bigger, so the chosen level of the cell is 6. In the Figure 113 is provided an example of the cells. The amount of the cells is about 1.5 million, and the iterations are more than 400.

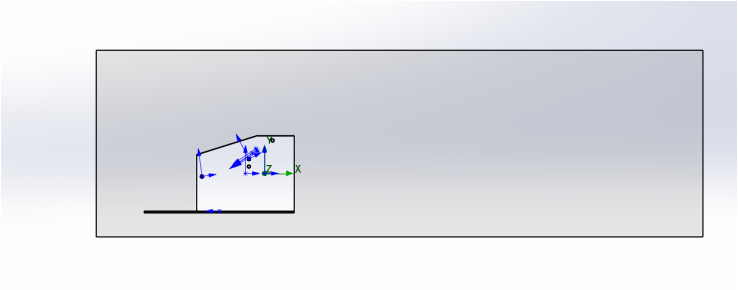


Figure 112: Domain selection

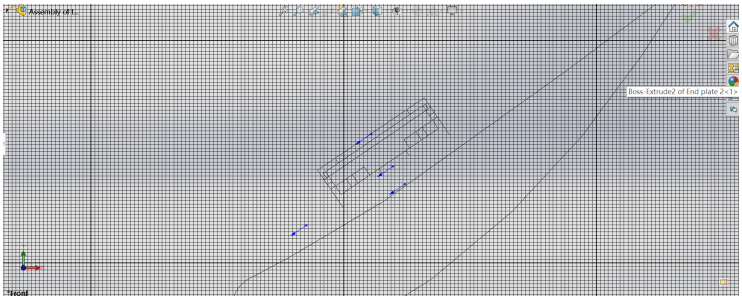


Figure 113: Cell example

Parameter	Value
Status	Preparing for calculation...
Total cells	1,487,547
Fluid cells	1,487,547
Fluid cells contacting solids	210,064
Iterations	408
Last iteration finished	11:30:16
CPU time per last iteration	00:00:15
Travels	3.00313
Iterations per 1 travel	163

Figure 114: Cell set-up

11.2.2 Simulation Results

In order to evaluate the effect that the box has on the wing, a simulation is conducted in order to see if it behaves as expected.

- **Without box**

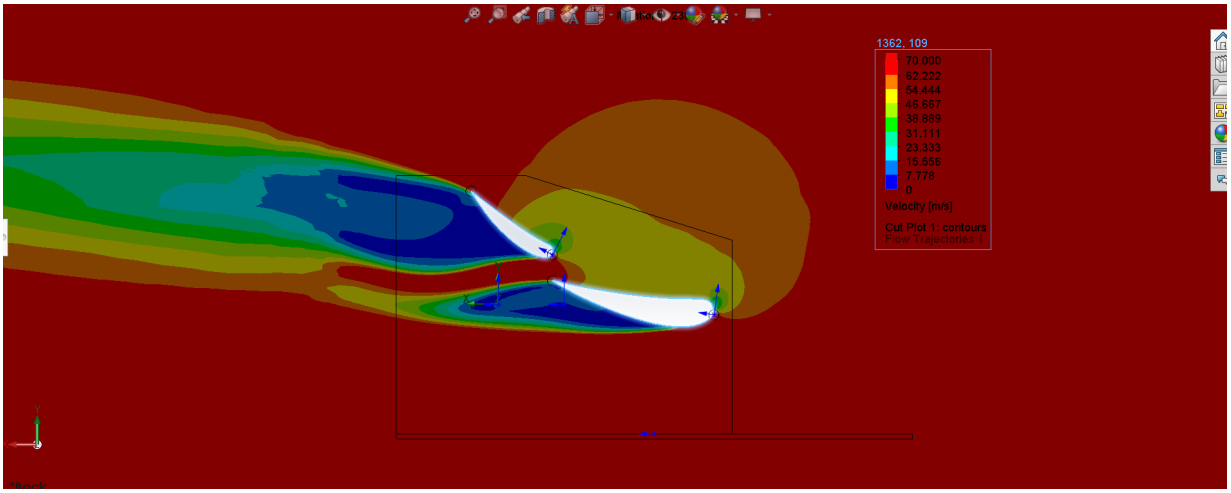


Figure 115: Velocity field without the box

- **With box** It can be noticed that the presence of the box create an area in which

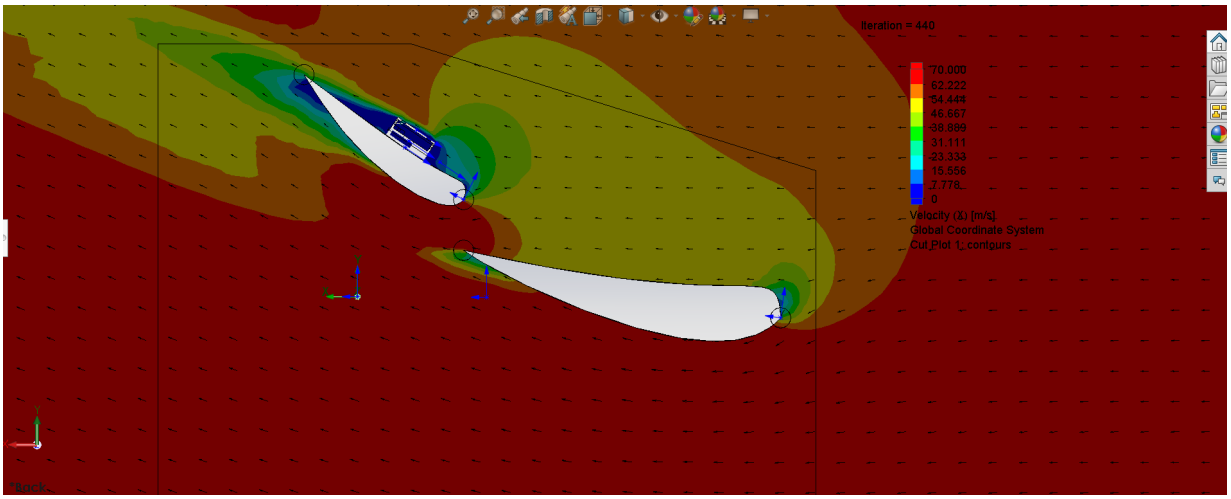


Figure 116: Velocity field with the box

the air flow decrease its velocity close by the box.

In Figure 117 and Figure 118 is shown the behaviour of the pressure in the two cases.

- **Without box**

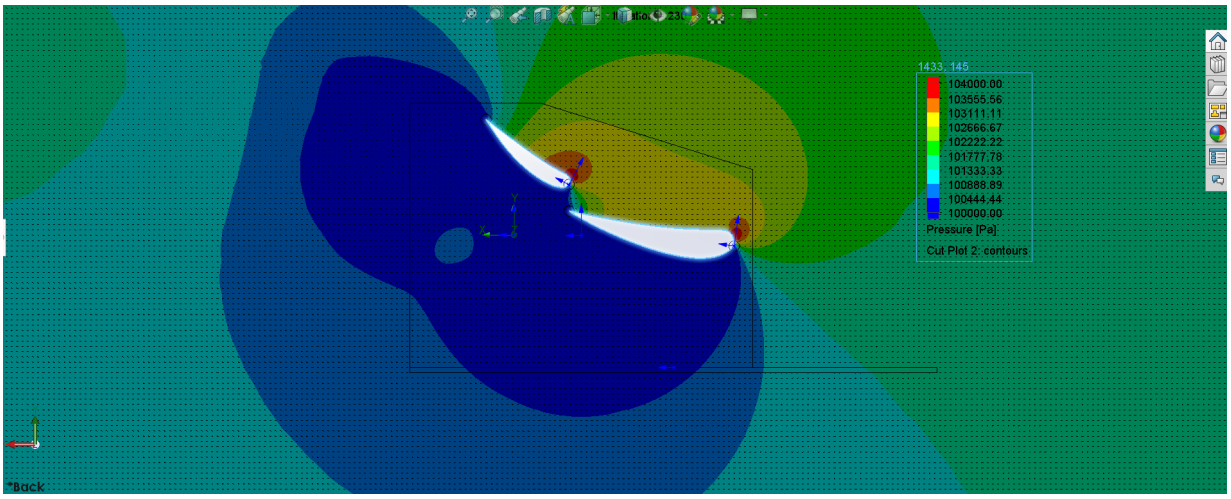


Figure 117: Pressure field without the box

- **With box**

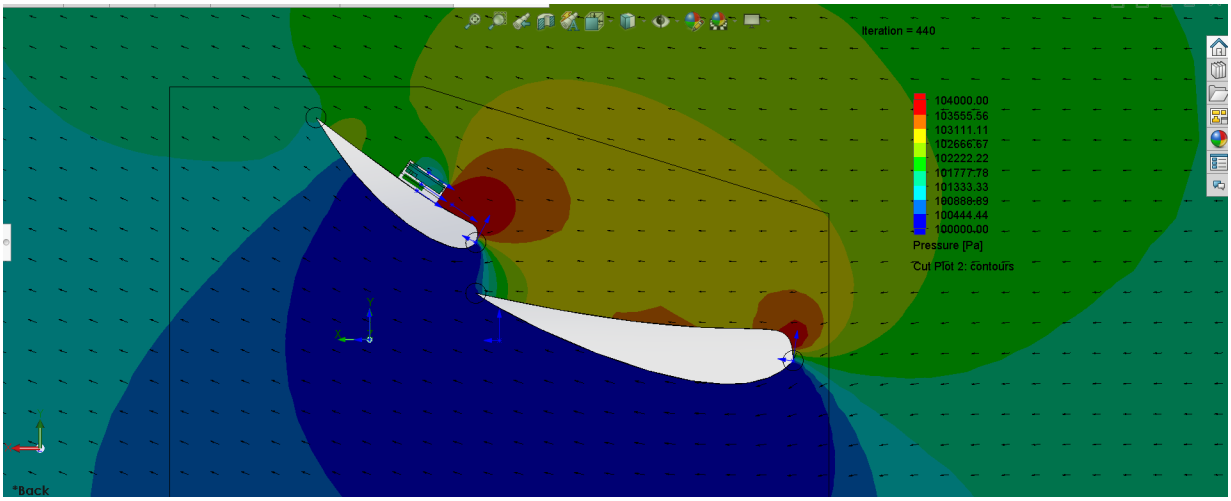


Figure 118: Pressure field with the box

12 Automotive Compliance Testing

For the components that are not automotive compliant a list of the tests, that have to be performed in order to be used for automotive applications, is provided.
On the components used for automotive application additional tests has to be performed to ensure safety. In the following section the test that needs to be performed on the components that are not on the market with AEC-Q100 certification. The components that needs to be tested is the barometric pressure sensor **LPS22DH** (or **BMP388**) and **NRF24L01+**.

EMC and Electrical Tests

- Conducted and radiated emission testing;
- Voltage surge immunity testing;
- Magnetic field immunity testing;
- Electrostatic Discharge (ESD) testing;
- Insulation test;
- Dielectric strength;

Environmental Tests

- Mechanical testing, in order to test the component behaviour in case of vibration, impact, shock, combined temperature cycling/vibration test;
- Climatic testing in order to test the component behaviour in case of temperature and humidity, thermal shock, fatigue and endurance and corrosion testing;
- Weathering testing, against solar and UV ageing;

Appendix A: Version Index

Version Num	Version	Date	Name
0.0	First draft documentation	October 12th, 2019	Annachiara Biguzzi
0.1	Project management strategy	October 14th, 2019	Giorgio Di Loro
0.2	Components definition	October 20th, 2019	Annachiara Biguzzi
0.3	HW description	November 5th, 2019	Annachiara Biguzzi
0.4	FW description	November 30th, 2019	Giacomo Ravagli
0.5	FW user interface description	December 4th, 2019	Luca Mancini
0.6	Mechanical description and validation	December 6th, 2019	Zhou Tong
0.7	SW user interface description	December 7th, 2019	Luca Mancini
0.8	HW validation description	December 7th, 2019	Annachiara Biguzzi
0.9	HW validation description	December 7th, 2019	Armando Villar
1.0	FW validation description	December 10th, 2019	Giacomo Ravagli

Table 30: Documentation versions

Appendix B: Board Schematic

CL1 Realized Board Schematic

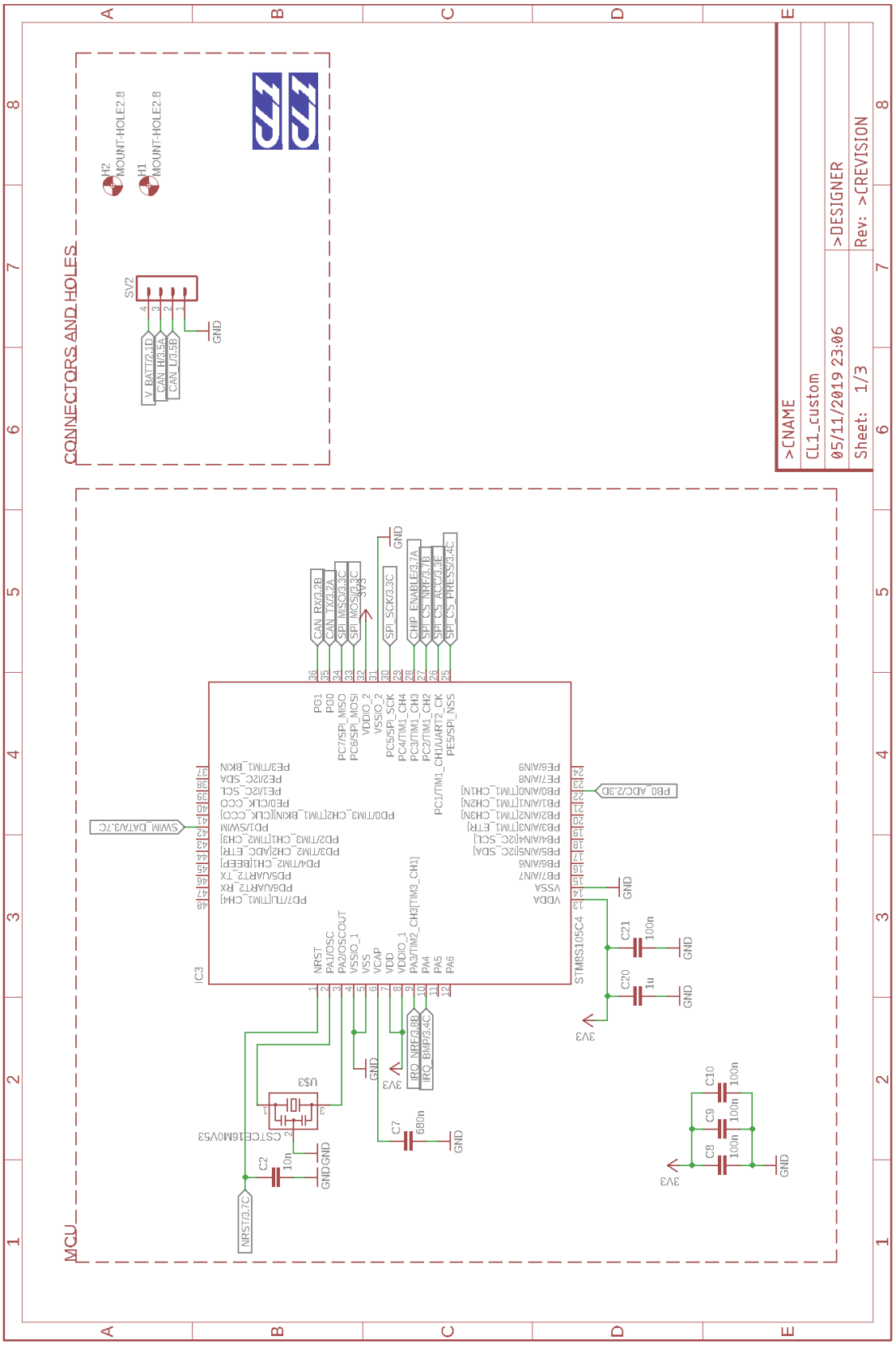


Figure 119: MCU and Connectors

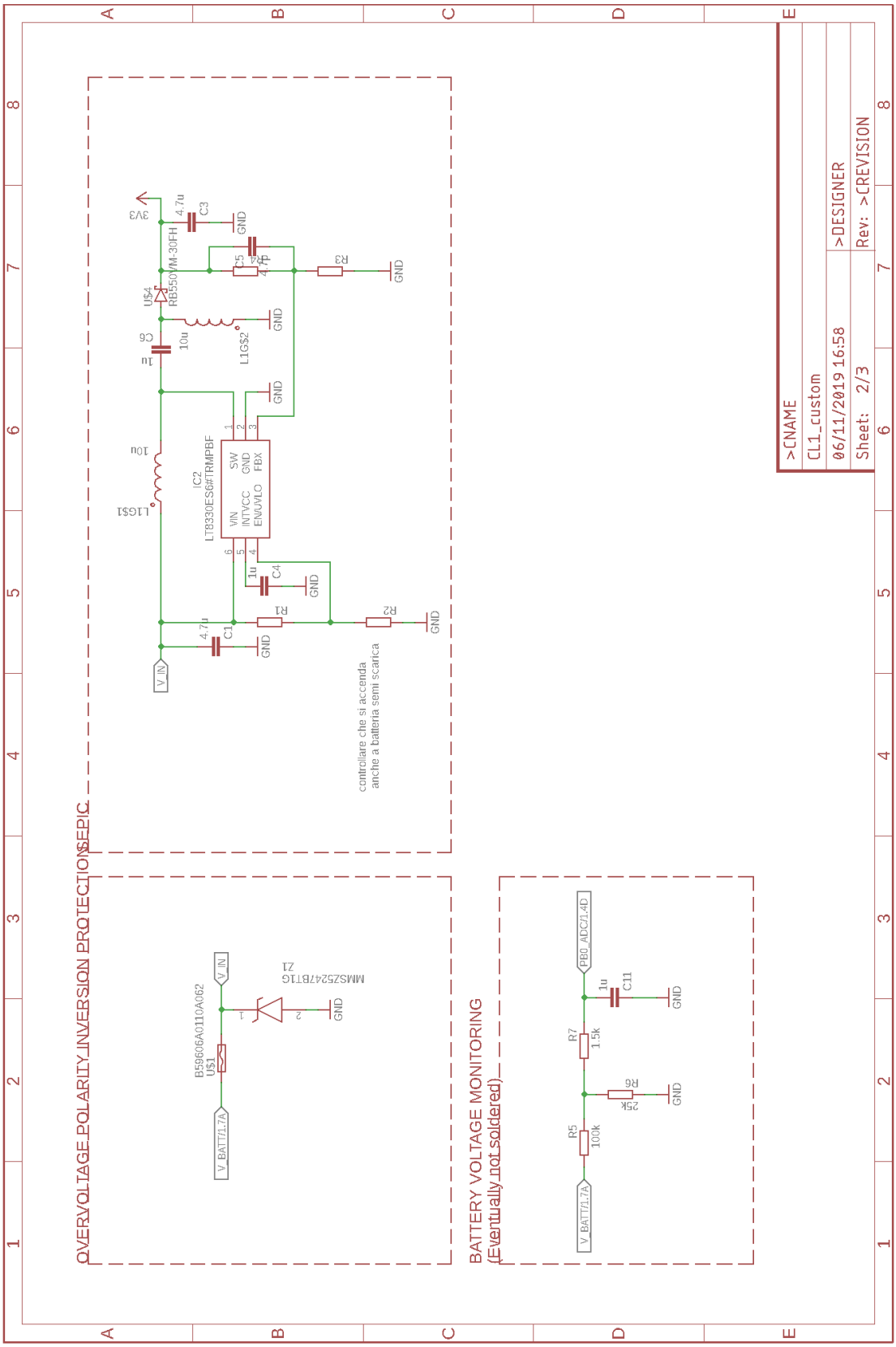


Figure 120: Power Supply

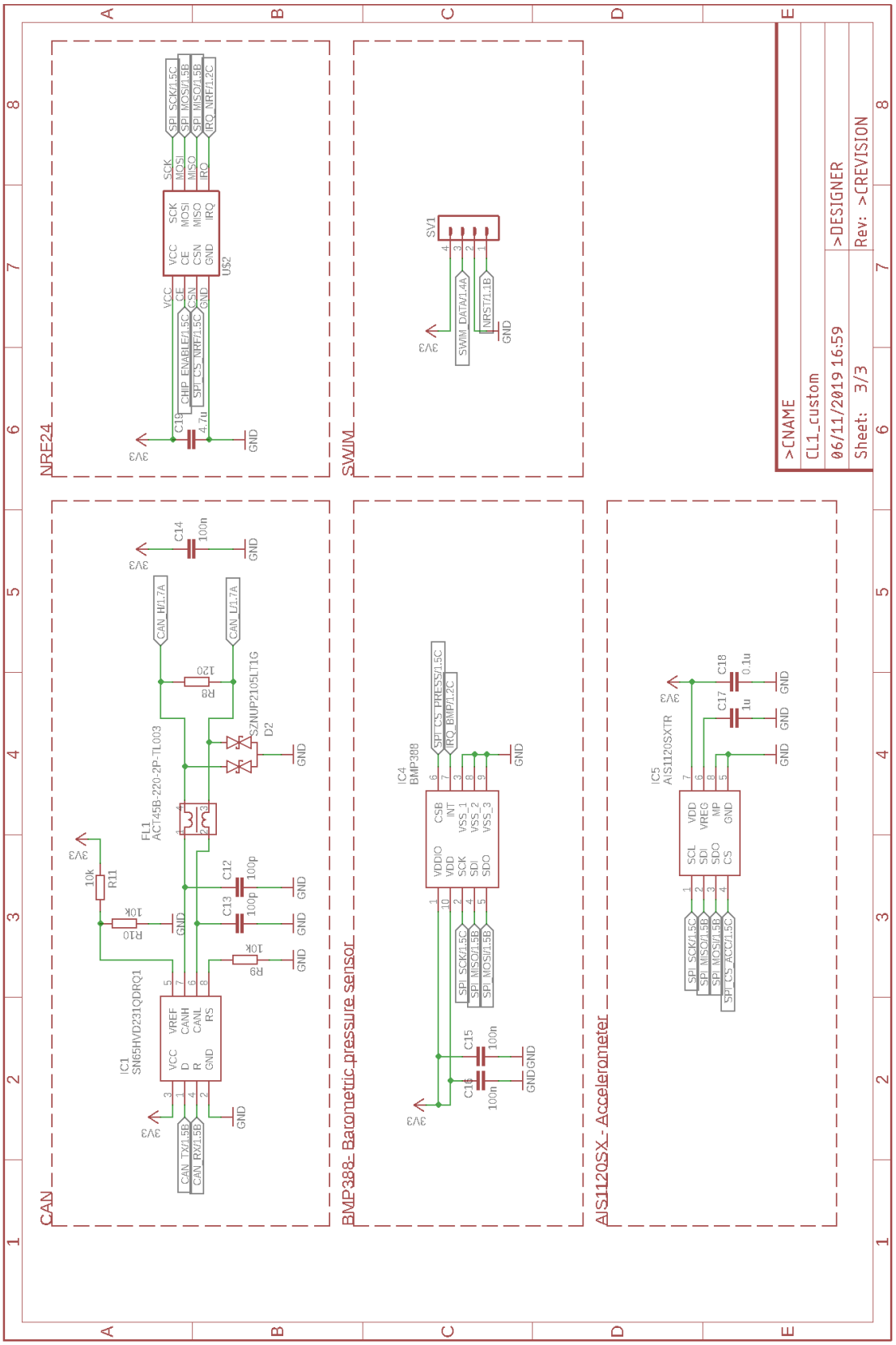


Figure 121: Sensors

CL1 Reduced Size Board Schematic

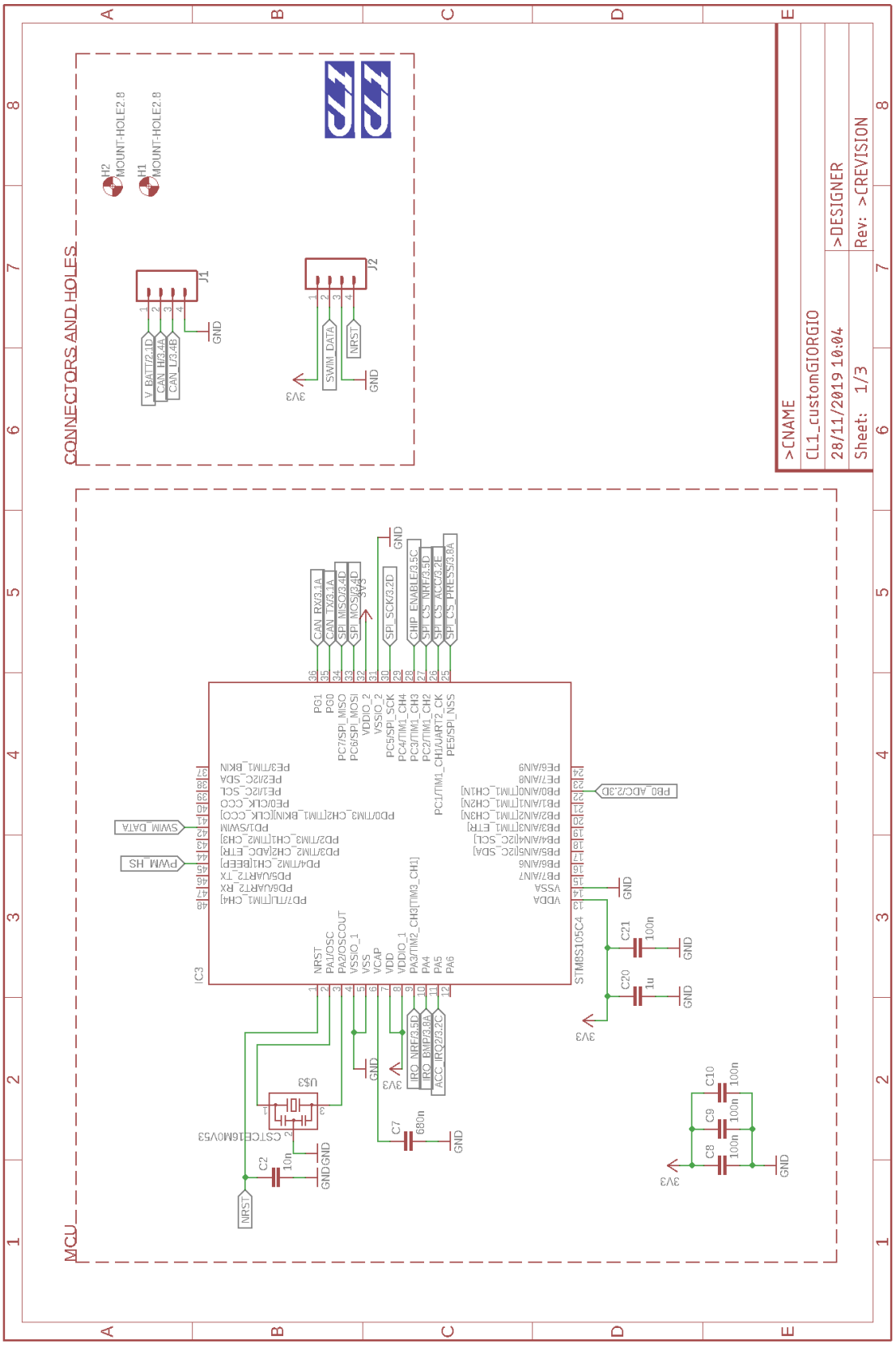


Figure 122: MCU and Connectors

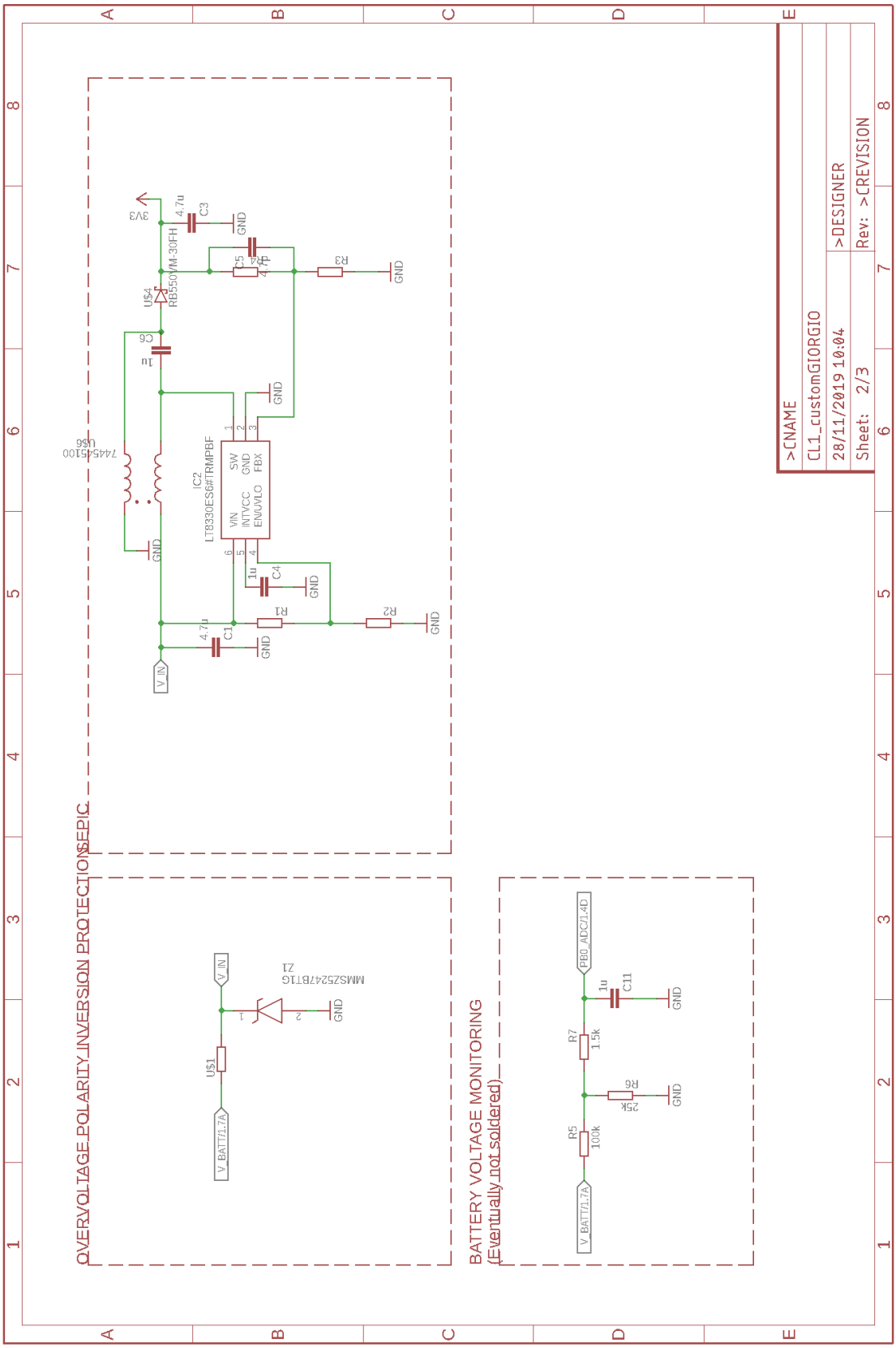


Figure 123: Power Supply

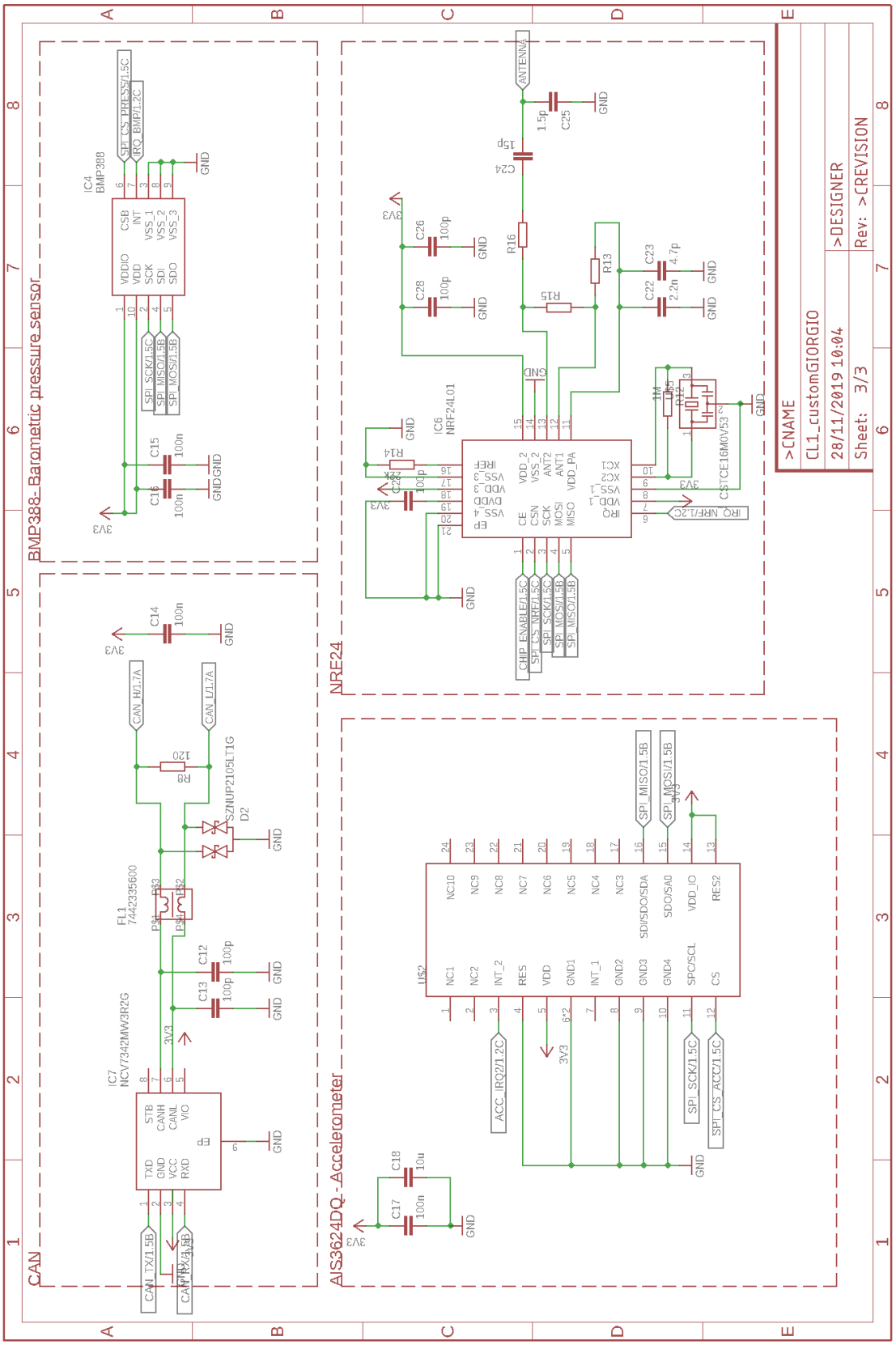


Figure 124: Sensors

Appendix C: Source Code

.I Evaluation Source Code - STM8AF5288T

```
/**
  ******************************************************************************
  * @file    unified_multimode.c
  * @author  CL1 Team
  * @version V3.0
  * @date    09-December-2019
  * @brief   This file is aimed at providing an handy tool to the user of the CL1 board.
  *          It allows an easy reconfiguration of the functionality of the board according to the use case,
  *          without wasting resources: only the relevant portion of code will be compiled, exploiting
  *          preprocessor's directives.
  *
  *
  *          Three OPERATING MODES are allowed:
  *          1) Slave mode (Smart Sensor): the CL1 board acquires the pressure and the acceleration, and send
  *          it to the master via radio packet
  *          2) Master mode: the CL1 board has no sensor; acquires radio packet sent by a slave node, and
  *          transmit the packet on the CAN bus
  *          3) Wired Sensor: the CL1 board acquires pressure and acceleration from sensor installed on it, and
  *          transmit the data directly on the CAN bus (no wireless communication in this case)
  *
  *          Pay attention: it is also needed to reconfigure the CE and CSN pin for the nRF24 module, according
  *          to the board where it is used --> change if used on the CL1 board
  *
  *          This code works on EVALUATION BOARD: to be used on CL1, we have to change the instructions for the sensors
  ******************************************************************************
  */


/** Uncomment according to what is the functionality required
  * ATTENTION: JUST UNCOMMENT ONE AT A TIME!
  */
#define SLAVE_MODE
/* #define MASTER_MODE */
/* #define WIRED_SENSOR */


#define HIGH_RESOLUTION
#define LIS3DH_DEBUG Serial  /* Enable Serial debbug on Serial UART to see registers wrote */


/* Includes ------------------------------------------------------------------*/
/* Include the libraries, according to what is the functionality required */
#include "Wire.h"
#include <SPI.h>
#if defined(MASTER_MODE) || defined(WIRED_SENSOR)
#include <stm8s_can.h>
#endif
#if defined(SLAVE_MODE) || defined(MASTER_MODE)
#include <NRFLite.h>
#endif
```

```
51  #if defined(SLAVE_MODE) || defined(WIRED_SENSOR)
52  #include <Adafruit_BMP085.h>
53  #include "lis3dh-motion-detection.h"
54  #endif
55
56
57  /* Private variables ---------------------------------------------------------*/
58  #if defined(MASTER_MODE) || defined(WIRED_SENSOR)
59  /* Initialization of the parameters for CAN transmission */
60  unsigned long ID = 0;
61  unsigned char tx_buffer[0x08] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}; //
        payload = 8 bytes
62  #endif
63
64  #if defined(MASTER_MODE)
65  /* Variables name for the Wireless transmission */
66  const static uint8_t RADIO_ID = 0;             // Receiver radio's id.  The transmitter
        will send to this id.
67  #endif
68
69  #if defined(SLAVE_MODE)
70  /* Variables name for the Wireless transmission */
71  const static uint8_t RADIO_ID = 1;             // Transmitter radio's id.
72  const static uint8_t DESTINATION_RADIO_ID = 0; // Id of the radio we will transmit to.
73  #endif
74
75  #if defined(SLAVE_MODE) || defined(MASTER_MODE)
76  const static uint8_t PIN_RADIO_CE = 2;         // 2 = PE0 in the STM8AF board --> to be
        changed to 6 = PC3 in final board
77  const static uint8_t PIN_RADIO_CSN = 5;        // 5 = PC2
78
79  /* Definition of the radio packet */
80  struct RadioPacket // Any packet up to 32 bytes can be sent.
81  {
82      uint8_t FromRadioId;
83      int32_t PressureTx;
84      int32_t AccelerationTx;
85      uint8_t FailedTxCount;
86  };
87
88  NRFLite _radio;
89  RadioPacket _radioData;
90  #endif
91
92  #if defined(SLAVE_MODE) || defined(WIRED_SENSOR)
93  /* Here we have the configuration parameters for the accelerometer  */
94  uint16_t sampleRate = 100;  // HZ - Samples per second - 1, 10, 25, 50, 100, 200, 400,
        1600, 5000
95  uint8_t accelRange = 16;    // Accelerometer range = 2, 4, 8, 16g
96  LIS3DH myIMU(0x18);         // Default address is 0x18.
97  int16_t dataHighres = 0;    // Save the raw data before conversion to float
98
99  /* Configuration parameter for the barometer */
100 Adafruit_BMP085 bmp;
101
102 float AccAsseZ = 0;         // Measured acceleration
103 int32_t AccAsseZ_mg = 0;    // Acceleration in milli-g
104 float Temp_Meas = 0;        // Measured temperature (BMP180)
105 int32_t Press_Meas = 0;     // Measured pressure
106 #endif
107
108
109
110 #if defined(SLAVE_MODE)
111 /**
```

```
112   * @brief  SLAVE MODE initialization function
113   * @param  None
114   * @retval None
115   * @note   Initialization of the SLAVE node.
116   */
117 void setup()
118 {
119   Serial.begin(115200);
120   delay(1000); //wait until serial is open...
121 
122   /* Initialization of the accelerometer */
123   if( myIMU.begin(sampleRate, 1, 1, 1, accelRange) != 0 )
124   {
125     Serial.print("Failed to initialize IMU.\n");
126   }
127   else
128   {
129     Serial.print("IMU initialized.\n");
130   }
131 
132   /* Detection threshold can be from 1 to 127 and depends on the Range
133    * chosen above, change it and test accordingly to your application
134    * Duration = timeDur x Seconds / sampleRate */
135   myIMU.intConf(INT_1, DET_MOVE, 13, 2);
136   myIMU.intConf(INT_2, DET_STOP, 13, 10);
137 
138   uint8_t readData = 0;
139 
140   /* Confirm configuration: */
141   myIMU.readRegister(&readData, LIS3DH_INT1_CFG);
142   myIMU.readRegister(&readData, LIS3DH_INT2_CFG);
143 
144   /* Get the ID: */
145   myIMU.readRegister(&readData, LIS3DH_WHO_AM_I);
146   Serial.print("Who am I? 0x");
147   Serial.println(readData, HEX);
148 
149 
150   /* Initialization of the barometer */
151   if (!bmp.begin())
152   {
153       Serial.println("Could not find a valid BMP085 sensor, check wiring!");
154       while (1);
155   }
156 
157 
158   /* Initialization of the nRF24 */
159   /* By default, 'init' configures the radio to use a 2MBPS bitrate on channel 100 (
          channels 0-125 are valid).
160    * Both the RX and TX radios must have the same bitrate and channel to communicate with
          each other.
161    * You can run the 'ChannelScanner' example to help select the best channel for your
          environment.
162    * You can assign a different bitrate and channel as shown below.
163    *   _radio.init(RADIO_ID, PIN_RADIO_CE, PIN_RADIO_CSN, NRFLite::BITRATE250KBPS, 0)
164    *   _radio.init(RADIO_ID, PIN_RADIO_CE, PIN_RADIO_CSN, NRFLite::BITRATE1MBPS, 75)
165    *   _radio.init(RADIO_ID, PIN_RADIO_CE, PIN_RADIO_CSN, NRFLite::BITRATE2MBPS, 100) //
          THE DEFAULT
166    */
167 
168   if (!_radio.init(RADIO_ID, PIN_RADIO_CE, PIN_RADIO_CSN))
169   {
170       SOS_blinking_LED();
171 
172       Serial.println("Cannot communicate with radio");
```

```
173         while (1); // Wait here forever.
174     }
175
176     _radioData.FromRadioId = RADIO_ID;
177 }
178 #endif
179
180
181 #if defined(MASTER_MODE)
182 /**
183   * @brief  MASTER MODE initialization function
184   * @param  None
185   * @retval None
186   * @note   Initialization of the MASTER node.
187   */
188 void setup()
189 {
190     Serial.begin(115200);
191     delay(1000);
192
193     pinMode(PC3, OUTPUT);
194
195     setup_CAN();
196
197   /* By default, 'init' configures the radio to use a 2MBPS bitrate on channel 100 (
          channels 0-125 are valid).
198    * Both the RX and TX radios must have the same bitrate and channel to communicate with
          each other.
199    * You can run the 'ChannelScanner' example to help select the best channel for your
          environment.
200    * You can assign a different bitrate and channel as shown below.
201    *   _radio.init(RADIO_ID, PIN_RADIO_CE, PIN_RADIO_CSN, NRFLite::BITRATE250KBPS, 0)
202    *   _radio.init(RADIO_ID, PIN_RADIO_CE, PIN_RADIO_CSN, NRFLite::BITRATE1MBPS, 75)
203    *   _radio.init(RADIO_ID, PIN_RADIO_CE, PIN_RADIO_CSN, NRFLite::BITRATE2MBPS, 100) //
          THE DEFAULT
204    */
205
206     if (!_radio.init(RADIO_ID, PIN_RADIO_CE, PIN_RADIO_CSN))
207     {
208         SOS_blinking_LED();
209         ErrorSignal_nRF_CAN();
210
211         /* Serial.println("Cannot communicate with radio"); */
212         while (1); /* Wait here forever. */
213     }
214 }
215 #endif
216
217
218 #if defined(WIRED_SENSOR)
219 /**
220   * @brief  WIRED SENSOR initialization function
221   * @param  None
222   * @retval None
223   * @note   Initialization of the WIRED SENSOR.
224   */
225 void setup()
226 {
227   Serial.begin(115200);
228   delay(1000);
229
230   setup_CAN();
231
232   /* Initialization of the accelerometer */
233   if( myIMU.begin(sampleRate, 1, 1, 1, accelRange) != 0 )
```

```
234     {
235       Serial.print("Failed␣to␣initialize␣IMU.\n");
236       ErrorSignal_LIS_CAN();
237     }
238     else
239     {
240       Serial.print("IMU␣initialized.\n");
241     }
242 
243     /* Detection threshold can be from 1 to 127 and depends on the Range
244      * chosen above, change it and test accordingly to your application
245      * Duration = timeDur x Seconds / sampleRate */
246     myIMU.intConf(INT_1, DET_MOVE, 13, 2);
247     myIMU.intConf(INT_2, DET_STOP, 13, 10);
248 
249     uint8_t readData = 0;
250 
251     /* Confirm configuration: */
252     myIMU.readRegister(&readData, LIS3DH_INT1_CFG);
253     myIMU.readRegister(&readData, LIS3DH_INT2_CFG);
254 
255     /* Get the ID: */
256     myIMU.readRegister(&readData, LIS3DH_WHO_AM_I);
257     Serial.print("Who␣am␣I?␣0x");
258     Serial.println(readData, HEX);
259 
260 
261     /* Initialization of the barometer */
262     if (!bmp.begin())
263     {
264         Serial.println("Could␣not␣find␣a␣valid␣BMP180␣sensor,␣check␣wiring!");
265         ErrorSignal_BMP_CAN();
266         while (1);
267     }
268 }
269 #endif
270 
271 
272 #if defined(MASTER_MODE) || defined(WIRED_SENSOR)
273 /**
274   * @brief CAN peripheral initialization function
275   * @param None
276   * @retval None
277   */
278 void setup_CAN()
279 {
280   CAN_DeInit();
281 
282   CAN_Init(CAN_MasterCtrl_NoAutoReTx,
283            CAN_Mode_Normal,
284            CAN_SynJumpWidth_1TimeQuantum,
285            CAN_BitSeg1_8TimeQuantum,
286            CAN_BitSeg2_7TimeQuantum,
287            4);
288 
289   CAN_FilterInit(CAN_FilterNumber_0,
290                  ENABLE,
291                  CAN_FilterMode_IdMask,
292                  CAN_FilterScale_32Bit,
293                  0x00,
294                  0x00,
295                  0x00,
296                  0x00,
297                  0x00,
298                  0x00,
```

```
299                     0x00,
300                     0x00);
301
302     CAN_ITConfig(CAN_IT_FMP,ENABLE);
303   }
304   #endif
305
306
307   #if defined(MASTER_MODE)
308   /**
309     * @brief  NRF24L01 initialization error function
310     * @param  None
311     * @retval None
312     */
313   void ErrorSignal_nRF_CAN()
314   {
315     /* Send an error signal on the CAN bus if radio is not correctly initialized */
316     tx_buffer[0] = 0x00;
317     tx_buffer[1] = 0x00;
318     tx_buffer[2] = 0x00;
319     tx_buffer[3] = 0x00;
320     tx_buffer[4] = 0x00;
321     tx_buffer[5] = 0x00;
322     tx_buffer[6] = 0x00;
323     tx_buffer[7] = 0x01;
324
325     send_data_to_CAN_bus();
326   }
327   #endif
328
329
330   #if defined(WIRED_SENSOR)
331   /**
332     * @brief  BMP initialization error function
333     * @param  None
334     * @retval None
335     */
336   void ErrorSignal_BMP_CAN()
337   {
338     /* Send an error signal on the CAN bus if BMP is not correctly initialized */
339     tx_buffer[0] = 0x00;
340     tx_buffer[1] = 0x00;
341     tx_buffer[2] = 0x00;
342     tx_buffer[3] = 0x00;
343     tx_buffer[4] = 0x00;
344     tx_buffer[5] = 0x00;
345     tx_buffer[6] = 0x00;
346     tx_buffer[7] = 0x02;
347
348     send_data_to_CAN_bus();
349   }
350
351
352   /**
353     * @brief  LIS3DH initialization error function
354     * @param  None
355     * @retval None
356     */
357   void ErrorSignal_LIS_CAN()
358   {
359     /* Send an error signal on the CAN bus if LIS (accelerometer) is not correctly
          initialized */
360     tx_buffer[0] = 0x00;
361     tx_buffer[1] = 0x00;
362     tx_buffer[2] = 0x00;
```

```
363   tx_buffer[3] = 0x00;
364   tx_buffer[4] = 0x00;
365   tx_buffer[5] = 0x00;
366   tx_buffer[6] = 0x00;
367   tx_buffer[7] = 0x03;
368
369   send_data_to_CAN_bus();
370 }
371 #endif
372
373
374 #if defined(SLAVE_MODE)
375 /**
376   * @brief  SLAVE MODE application entry point.
377   * @retval None
378   * @note   Execution of the main program
379   */
380 void loop()
381 {
382   /* Here perform the acquisition from the accelerometer */
383   dataHighres = 0;
384   myIMU.readRegisterInt16( &dataHighres, LIS3DH_OUT_X_L );
385   myIMU.readRegisterInt16( &dataHighres, LIS3DH_OUT_Z_L );
386   AccAsseZ = myIMU.axisAccel( Z );     // store the data in floating point
387   AccAsseZ_mg = (int32_t)(AccAsseZ * 1000);
388   // Print the result, i.e. acceleration in mg
389   Serial.print("Acc. axis Z in mg = ");
390   Serial.println(AccAsseZ_mg); // print with 4 digit after comma
391
392   /* Perform the acquisition from the barometer */
393   Temp_Meas = bmp.readTemperature();
394   Press_Meas = bmp.readPressure();
395   // Print the result: Temperature and Pressure reading
396   Serial.print("Temperature = ");
397   Serial.print(Temp_Meas);
398   Serial.println(" *C");
399   Serial.print("Pressure = ");
400   Serial.print(Press_Meas);
401   Serial.println(" Pa");
402   Serial.print("\n");
403
404   /* Send the data */
405   _radioData.PressureTx = Press_Meas;
406   _radioData.AccelerationTx = AccAsseZ_mg;
407   Serial.print("Sending ");
408   Serial.print(_radioData.PressureTx);
409   Serial.print(" Pa, and ");
410   Serial.print(_radioData.AccelerationTx);
411   Serial.print(" mg");
412
413   /* By default, 'send' transmits data and waits for an acknowledgement.  If no
          acknowledgement is received,
414    * it will try again up to 16 times.  You can also perform a NO_ACK send that does not
          request an acknowledgement.
415    * The data packet will only be transmitted a single time so there is no guarantee it
          will be successful.  Any random
416    * electromagnetic interference can sporatically cause packets to be lost, so NO_ACK
          sends are only suited for certain
417    * types of situations, such as streaming real-time data where performance is more
          important than reliability.
418    *   _radio.send(DESTINATION_RADIO_ID, &_radioData, sizeof(_radioData), NRFLite::NO_ACK)
419    *   _radio.send(DESTINATION_RADIO_ID, &_radioData, sizeof(_radioData), NRFLite::
          REQUIRE_ACK) // THE DEFAULT
420    */
421
```

```
422   if (_radio.send(DESTINATION_RADIO_ID, &_radioData, sizeof(_radioData))) /* Note how '&'
          must be placed in front of the variable name. */
423   {
424       Serial.println("...Success");
425   }
426   else
427   {
428       Serial.println("...Failed");
429       _radioData.FailedTxCount++;
430   }
431
432
433   Serial.print("\n");
434   Serial.print("\n");
435   Serial.print("\n");
436
437   /* At the end, restart the cycle after a certain amount of time (ms) */
438   delay(10);
439 }
440 #endif
441
442
443 #if defined(MASTER_MODE)
444 /**
445   * @brief  MASTER MODE application entry point.
446   * @retval None
447   * @note   Execution of the main program
448   */
449 void loop()
450 {
451
452     digitalWrite(PC3, HIGH);   // just used as a simple debug, to see if the code has
          arrived there
453     while (_radio.hasData())
454     {
455         digitalWrite(PD0, HIGH);
456
457         _radio.readData(&_radioData); // Note how '&' must be placed in front of the
              variable name
458
459         /* This line of code corrects the received data: in fact, the data is sent using a
               FIFO logic related to a single byte [the result is stored as follows:
460          * MSByte(received data)=LSByte(transmitted data), ... , LSByte(received data)=
               MSByte(transmitted data)].
461          * The following line thus applies a bitmask to obtain one byte of the received
               data; a shift to each byte in order to position it correctly inside the 32-
               bit word;
462          * a sum to merge all the 4 repositioned Bytes together. NB: THIS WORKS ONLY FOR
               32-BIT DATA, CHANGE IT FOR OTHER TYPES OF DATA.
463          * Surprisingly, no bit mask is required working with another ST Nucleo 8S 208RB
464          */
465
466         //_radioData.PressureTx = ((_radioData.PressureTx & 0xFF000000)>>24)+((_radioData.
              PressureTx & 0x00FF0000)>>8)+((_radioData.PressureTx & 0x0000FF00)<<8)+((
              _radioData.PressureTx & 0x000000FF)<<24);
467         //_radioData.AccelerationTx = ((_radioData.AccelerationTx & 0xFF000000)>>24)+((
              _radioData.AccelerationTx & 0x00FF0000)>>8)+((_radioData.AccelerationTx & 0
              x0000FF00)<<8)+((_radioData.AccelerationTx & 0x000000FF)<<24);
468
469         String msg = "Radio␣";
470         msg += _radioData.FromRadioId;
471         msg += ",␣pressure:";
472         msg += _radioData.PressureTx;
473         msg += "␣Pa,␣acc:";
474         msg += _radioData.AccelerationTx;
```

```
475         msg += "␣mg.␣";
476         msg += _radioData.FailedTxCount;
477         msg += "␣Failed␣TX";
478 
479         Serial.println(msg);
480 
481         digitalWrite(PD0, LOW);
482 
483 
484         /* Create the bit mask for the PRESSURE data to be transmitted on the CAN bus
485          * int32_t --> 4 x uint8_t  (then, at the receiver size we need to do the same
486          */
487         tx_buffer[0] = (uint8_t)(_radioData.PressureTx);
488         tx_buffer[1] = (uint8_t)(_radioData.PressureTx>>8);
489         tx_buffer[2] = (uint8_t)(_radioData.PressureTx>>16);
490         tx_buffer[3] = (uint8_t)(_radioData.PressureTx>>24);
491         /* Create the bit mask for the ACCELEROMETER data to be transmitted on the CAN bus
492          * int32_t --> 4 x uint8_t  (then, at the receiver size we need to do the same
493          */
494         tx_buffer[4] = (uint8_t)(_radioData.AccelerationTx);
495         tx_buffer[5] = (uint8_t)(_radioData.AccelerationTx>>8);
496         tx_buffer[6] = (uint8_t)(_radioData.AccelerationTx>>16);
497         tx_buffer[7] = (uint8_t)(_radioData.AccelerationTx>>24);
498 
499 
500         send_data_to_CAN_bus();
501     }
502 }
503 #endif
504 
505 
506 #if defined(WIRED_SENSOR)
507 /**
508   * @brief  WIRED SENSOR application entry point.
509   * @retval None
510   * @note   Execution of the main program
511   */
512 void loop()
513 {
514   /* Here perform the acquisition from the accelerometer */
515   dataHighres = 0;
516   myIMU.readRegisterInt16( &dataHighres, LIS3DH_OUT_X_L );
517   myIMU.readRegisterInt16( &dataHighres, LIS3DH_OUT_Z_L );
518   AccAsseZ = myIMU.axisAccel( Z );     // store the data in floating point
519   AccAsseZ_mg = (int32_t)(AccAsseZ * 1000);
520   // Print the result, i.e. acceleration in mg
521   Serial.print("Acc.␣axis␣Z␣in␣mg␣=␣");
522   Serial.println(AccAsseZ_mg); // print with 4 digit after comma
523 
524   /* Perform the acquisition from the barometer */
525   Temp_Meas = bmp.readTemperature();
526   Press_Meas = bmp.readPressure();
527   // Print the result: Temperature and Pressure reading
528   Serial.print("Temperature␣=␣");
529   Serial.print(Temp_Meas);
530   Serial.println("␣*C");
531   Serial.print("Pressure␣=␣");
532   Serial.print(Press_Meas);
533   Serial.println("␣Pa");
534   Serial.print("\n");
535 
536   /* Create the bit mask for the PRESSURE data to be transmitted on the CAN bus
537    * int32_t --> 4 x uint8_t  (then, at the receiver size we need to do the same
538    */
539   tx_buffer[0] = (uint8_t)(Press_Meas);
```

```
540     tx_buffer[1] = (uint8_t)(Press_Meas>>8);
541     tx_buffer[2] = (uint8_t)(Press_Meas>>16);
542     tx_buffer[3] = (uint8_t)(Press_Meas>>24);
543     /* Create the bit mask for the ACCELEROMETER data to be transmitted on the CAN bus
544      * int32_t --> 4 x uint8_t  (then, at the receiver size we need to do the same
545      */
546     tx_buffer[4] = (uint8_t)(AccAsseZ_mg);
547     tx_buffer[5] = (uint8_t)(AccAsseZ_mg>>8);
548     tx_buffer[6] = (uint8_t)(AccAsseZ_mg>>16);
549     tx_buffer[7] = (uint8_t)(AccAsseZ_mg>>24);
550 
551     send_data_to_CAN_bus();
552   }
553   #endif
554 
555 
556   #if defined(MASTER_MODE) || defined(WIRED_SENSOR)
557   /**
558     * @brief  CAN sending data function
559     * @param  None
560     * @retval None
561     * @note   This function sends the specified payload on the CAN bus.
562     */
563   void send_data_to_CAN_bus()
564   {
565     digitalWrite(PA3, HIGH);
566     CAN_Transmit(0x0001,  CAN_Id_Extended, CAN_RTR_Data, 8, tx_buffer);  /* ID: 0b0000 0100
          1001 0110 */
567     delay(1);
568     digitalWrite(PC3, LOW);
569   }
570   #endif
571 
572 
573   /**
574     * @brief  SOS led blinking function
575     * @param  None
576     * @retval None
577     * @note   This function outputs an SOS message using leds, in case of initialization
          errors.
578     */
579   void SOS_blinking_LED()
580   {
581     /* Initialize all the LEDs; will be used to encode error messages */
582           pinMode(PA3, OUTPUT);
583           pinMode(PD3, OUTPUT);
584           pinMode(PD0, OUTPUT);
585           pinMode(PE3, OUTPUT);
586           pinMode(PC3, OUTPUT);
587     /* Encoding that there is a problem: 3 LEDs blink an SOS */
588           digitalWrite(PA3, HIGH);
589           digitalWrite(PD0, HIGH);
590           digitalWrite(PC3, HIGH);
591           delay(100);
592           digitalWrite(PA3, LOW);
593           digitalWrite(PD0, LOW);
594           digitalWrite(PC3, LOW);
595           delay(100);
596           digitalWrite(PA3, HIGH);
597           digitalWrite(PD0, HIGH);
598           digitalWrite(PC3, HIGH);
599           delay(100);
600           digitalWrite(PA3, LOW);
601           digitalWrite(PD0, LOW);
602           digitalWrite(PC3, LOW);
```

```
603             delay(100);
604             digitalWrite(PA3, HIGH);
605             digitalWrite(PD0, HIGH);
606             digitalWrite(PC3, HIGH);
607             delay(100);
608             digitalWrite(PA3, LOW);
609             digitalWrite(PD0, LOW);
610             digitalWrite(PC3, LOW);
611             delay(100);                 // S
612             digitalWrite(PA3, HIGH);
613             digitalWrite(PD0, HIGH);
614             digitalWrite(PC3, HIGH);
615             delay(500);
616             digitalWrite(PA3, LOW);
617             digitalWrite(PD0, LOW);
618             digitalWrite(PC3, LOW);
619             delay(100);
620             digitalWrite(PA3, HIGH);
621             digitalWrite(PD0, HIGH);
622             digitalWrite(PC3, HIGH);
623             delay(500);
624             digitalWrite(PA3, LOW);
625             digitalWrite(PD0, LOW);
626             digitalWrite(PC3, LOW);
627             delay(100);
628             digitalWrite(PA3, HIGH);
629             digitalWrite(PD0, HIGH);
630             digitalWrite(PC3, HIGH);
631             delay(500);
632             digitalWrite(PA3, LOW);
633             digitalWrite(PD0, LOW);
634             digitalWrite(PC3, LOW);
635             delay(100);                 // O
636             digitalWrite(PA3, HIGH);
637             digitalWrite(PD0, HIGH);
638             digitalWrite(PC3, HIGH);
639             delay(100);
640             digitalWrite(PA3, LOW);
641             digitalWrite(PD0, LOW);
642             digitalWrite(PC3, LOW);
643             delay(100);
644             digitalWrite(PA3, HIGH);
645             digitalWrite(PD0, HIGH);
646             digitalWrite(PC3, HIGH);
647             delay(100);
648             digitalWrite(PA3, LOW);
649             digitalWrite(PD0, LOW);
650             digitalWrite(PC3, LOW);
651             delay(100);
652             digitalWrite(PA3, HIGH);
653             digitalWrite(PD0, HIGH);
654             digitalWrite(PC3, HIGH);
655             delay(100);
656             digitalWrite(PA3, LOW);
657             digitalWrite(PD0, LOW);
658             digitalWrite(PC3, LOW);
659             delay(100);                 // S
660 }
```

.II Board Source Code - CL1

```
/**
  ******************************************************************************
  * @file    CL1_unified_multimode_reducedSize.c
  * @author  CL1 Team
  * @version V3.0
  * @date    09-December-2019
  * @brief   This file is aimed at providing an handy tool to the user of the CL1 board.
  *          It allows an easy reconfiguration of the functionality of the board according to the use case,
  *          without wasting resources: only the relevant portion of code will be compiled, exploiting
  *          preprocessor's directives.
  *
  *
  *          Three OPERATING MODES are allowed:
  *          1) Slave mode (Smart Sensor): the CL1 board acquires the pressure and the acceleration, and send
  *          it to the master via radio packet
  *          2) Master mode: the CL1 board has no sensor; acquires radio packet sent by a slave node, and
  *          transmit the packet on the CAN bus
  *          3) Wired Sensor: the CL1 board acquires pressure and acceleration from sensor installed on it, and
  *          transmit the data directly on the CAN bus (no wireless communication in this case)
  *
  *          Pay attention: it is also needed to reconfigure the CE and CSN pin for the nRF24 module, according
  *          to the board where it is used --> change if used on the CL1 board
  ******************************************************************************
  */


/** Uncomment according to what is the functionality required
  * ATTENTION: JUST UNCOMMENT ONE AT A TIME!
  */
/* #define SLAVE_MODE */
#define MASTER_MODE
/* #define WIRED_SENSOR */


#define HIGH_RESOLUTION
#define LIS3DH_DEBUG Serial  /* Enable Serial debbug on Serial UART to see registers wrote */


/* Includes ------------------------------------------------------------------*/
/* Include the libraries, according to what is the functionality required */
#include "Wire.h"
#include <SPI.h>
#if defined(MASTER_MODE) || defined(WIRED_SENSOR)
#include <stm8s_can.h>
#endif
#if defined(SLAVE_MODE) || defined(MASTER_MODE)
#include <NRFLite.h>
#endif
#if defined(SLAVE_MODE) || defined(WIRED_SENSOR)
#include <Adafruit_BMP085.h>
#include "lis3dh-motion-detection.h"
#endif


```

```
55  /* Private variables ---------------------------------------------------------*/
56  #if defined(MASTER_MODE) || defined(WIRED_SENSOR)
57  /* Initialization of the parameters for CAN transmission */
58  unsigned long ID = 0;
59  unsigned char tx_buffer[0x08] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}; //
        payload = 8 bytes
60  #endif
61
62  #if defined(MASTER_MODE)
63  /* Variables name for the Wireless transmission */
64  const static uint8_t RADIO_ID = 0;             // Receiver radio's id.  The transmitter
        will send to this id.
65  #endif
66  #if defined(SLAVE_MODE)
67  /* Variables name for the Wireless transmission */
68  const static uint8_t RADIO_ID = 1;             // Transmitter radio's id.
69  const static uint8_t DESTINATION_RADIO_ID = 0; // Id of the radio we will transmit to.
70  #endif
71  #if defined(SLAVE_MODE) || defined(MASTER_MODE)
72  const static uint8_t PIN_RADIO_CE = 6;         // 6 = PC3 in final board
73  const static uint8_t PIN_RADIO_CSN = 5;        // 5 = PC2
74
75  /* Definition of the radio packet */
76  struct RadioPacket // Any packet up to 32 bytes can be sent.
77  {
78      uint8_t FromRadioId;
79      int32_t PressureTx;
80      int32_t AccelerationTx;
81      uint8_t FailedTxCount;
82  };
83
84  NRFLite _radio;
85  RadioPacket _radioData;
86  #endif
87
88  #if defined(SLAVE_MODE) || defined(WIRED_SENSOR)
89  /* Here we have the configuration parameters for the accelerometer  */
90  uint16_t sampleRate = 100;  // HZ - Samples per second - 1, 10, 25, 50, 100, 200, 400,
        1600, 5000
91  uint8_t accelRange = 16;    // Accelerometer range = 2, 4, 8, 16g
92  LIS3DH myIMU(0x18);         // Default address is 0x18.
93  int16_t dataHighres = 0;    // Save the raw data before conversion to float
94
95  /* Configuration parameter for the barometer */
96  Adafruit_BMP085 bmp;
97
98  float AccAsseZ = 0;         // Measured acceleration
99  int32_t AccAsseZ_mg = 0;    // Acceleration in milli-g
100 float Temp_Meas = 0;        // Measured temperature (BMP180)
101 int32_t Press_Meas = 0;     // Measured pressure
102 #endif
103
104
105
106 #if defined(SLAVE_MODE)
107 /**
108   * @brief  SLAVE MODE initialization function
109   * @param  None
110   * @retval None
111   * @note   Initialization of the SLAVE node.
112   */
113 void setup()
114 {
115   delay(1000); //wait until serial is open...
116
```

```
117     /* Initialization of the accelerometer */
118     if( myIMU.begin(sampleRate, 1, 1, 1, accelRange) != 0 )
119     {
120     }
121     else
122     {
123     }
124
125     /* Detection threshold can be from 1 to 127 and depends on the Range
126      * chosen above, change it and test accordingly to your application
127      * Duration = timeDur x Seconds / sampleRate */
128     myIMU.intConf(INT_1, DET_MOVE, 13, 2);
129     myIMU.intConf(INT_2, DET_STOP, 13, 10);
130
131     uint8_t readData = 0;
132
133     /* Confirm configuration: */
134     myIMU.readRegister(&readData, LIS3DH_INT1_CFG);
135     myIMU.readRegister(&readData, LIS3DH_INT2_CFG);
136
137     /* Get the ID: */
138     myIMU.readRegister(&readData, LIS3DH_WHO_AM_I);
139
140
141     /* Initialization of the barometer */
142     if (!bmp.begin())
143     {
144         while (1);
145     }
146
147
148     /* Initialization of the nRF24 */
149     /* By default, 'init' configures the radio to use a 2MBPS bitrate on channel 100 (
            channels 0-125 are valid).
150      * Both the RX and TX radios must have the same bitrate and channel to communicate with
            each other.
151      * You can run the 'ChannelScanner' example to help select the best channel for your
            environment.
152      * You can assign a different bitrate and channel as shown below.
153      *   _radio.init(RADIO_ID, PIN_RADIO_CE, PIN_RADIO_CSN, NRFLite::BITRATE250KBPS, 0)
154      *   _radio.init(RADIO_ID, PIN_RADIO_CE, PIN_RADIO_CSN, NRFLite::BITRATE1MBPS, 75)
155      *   _radio.init(RADIO_ID, PIN_RADIO_CE, PIN_RADIO_CSN, NRFLite::BITRATE2MBPS, 100) //
            THE DEFAULT
156      */
157
158     if (!_radio.init(RADIO_ID, PIN_RADIO_CE, PIN_RADIO_CSN))
159     {
160         while (1); // Wait here forever.
161     }
162
163     _radioData.FromRadioId = RADIO_ID;
164 }
165 #endif
166
167
168 #if defined(MASTER_MODE)
169 /**
170   * @brief  MASTER MODE initialization function
171   * @param  None
172   * @retval None
173   * @note   Initialization of the MASTER node.
174   */
175 void setup()
176 {
177     delay(1000);
```

```
178
179     setup_CAN();
180
181   /* By default, 'init' configures the radio to use a 2MBPS bitrate on channel 100 (
          channels 0-125 are valid).
182    * Both the RX and TX radios must have the same bitrate and channel to communicate with
          each other.
183    * You can run the 'ChannelScanner' example to help select the best channel for your
          environment.
184    * You can assign a different bitrate and channel as shown below.
185    *   _radio.init(RADIO_ID, PIN_RADIO_CE, PIN_RADIO_CSN, NRFLite::BITRATE250KBPS, 0)
186    *   _radio.init(RADIO_ID, PIN_RADIO_CE, PIN_RADIO_CSN, NRFLite::BITRATE1MBPS, 75)
187    *   _radio.init(RADIO_ID, PIN_RADIO_CE, PIN_RADIO_CSN, NRFLite::BITRATE2MBPS, 100) //
          THE DEFAULT
188    */
189
190     if (!_radio.init(RADIO_ID, PIN_RADIO_CE, PIN_RADIO_CSN))
191     {
192         ErrorSignal_nRF_CAN();
193         while (1); // Wait here forever.
194     }
195 }
196 #endif
197
198
199 #if defined(WIRED_SENSOR)
200 /**
201   * @brief  WIRED SENSOR initialization function
202   * @param  None
203   * @retval None
204   * @note   Initialization of the WIRED SENSOR.
205   */
206 void setup()
207 {
208   delay(1000);
209
210   setup_CAN();
211
212   /* Initialization of the accelerometer */
213   if( myIMU.begin(sampleRate, 1, 1, 1, accelRange) != 0 )
214   {
215     ErrorSignal_LIS_CAN();
216   }
217   else
218   {
219   }
220
221   /* Detection threshold can be from 1 to 127 and depends on the Range
222    * chosen above, change it and test accordingly to your application
223    * Duration = timeDur x Seconds / sampleRate */
224   myIMU.intConf(INT_1, DET_MOVE, 13, 2);
225   myIMU.intConf(INT_2, DET_STOP, 13, 10);
226
227   uint8_t readData = 0;
228
229   /* Confirm configuration: */
230   myIMU.readRegister(&readData, LIS3DH_INT1_CFG);
231   myIMU.readRegister(&readData, LIS3DH_INT2_CFG);
232
233   /* Get the ID: */
234   myIMU.readRegister(&readData, LIS3DH_WHO_AM_I);
235
236
237   /* Initialization of the barometer */
238   if (!bmp.begin())
```

```
239   {
240       ErrorSignal_BMP_CAN();
241       while (1);
242   }
243 }
244 #endif
245
246
247 #if defined(MASTER_MODE) || defined(WIRED_SENSOR)
248 /**
249   * @brief  CAN peripheral initialization function
250   * @param  None
251   * @retval None
252   */
253 void setup_CAN()
254 {
255   CAN_DeInit();
256
257   CAN_Init(CAN_MasterCtrl_NoAutoReTx,
258            CAN_Mode_Normal,
259            CAN_SynJumpWidth_1TimeQuantum,
260            CAN_BitSeg1_8TimeQuantum,
261            CAN_BitSeg2_7TimeQuantum,
262            4);
263
264   CAN_FilterInit(CAN_FilterNumber_0,
265                  ENABLE,
266                  CAN_FilterMode_IdMask,
267                  CAN_FilterScale_32Bit,
268                  0x00,
269                  0x00,
270                  0x00,
271                  0x00,
272                  0x00,
273                  0x00,
274                  0x00,
275                  0x00);
276
277   CAN_ITConfig(CAN_IT_FMP,ENABLE);
278 }
279 #endif
280
281
282 #if defined(MASTER_MODE)
283 /**
284   * @brief  NRF24L01 initialization error function
285   * @param  None
286   * @retval None
287   */
288 void ErrorSignal_nRF_CAN()
289 {
290   /* Send an error signal on the CAN bus if radio is not correctly initialized */
291   tx_buffer[0] = 0x00;
292   tx_buffer[1] = 0x00;
293   tx_buffer[2] = 0x00;
294   tx_buffer[3] = 0x00;
295   tx_buffer[4] = 0x00;
296   tx_buffer[5] = 0x00;
297   tx_buffer[6] = 0x00;
298   tx_buffer[7] = 0x01;
299
300   send_data_to_CAN_bus();
301 }
302 #endif
303
```

```
304
305 #if defined(WIRED_SENSOR)
306 /**
307   * @brief  BMP initialization error function
308   * @param  None
309   * @retval None
310   */
311 void ErrorSignal_BMP_CAN()
312 {
313   /* Send an error signal on the CAN bus if BMP is not correctly initialized */
314   tx_buffer[0] = 0x00;
315   tx_buffer[1] = 0x00;
316   tx_buffer[2] = 0x00;
317   tx_buffer[3] = 0x00;
318   tx_buffer[4] = 0x00;
319   tx_buffer[5] = 0x00;
320   tx_buffer[6] = 0x00;
321   tx_buffer[7] = 0x02;
322
323   send_data_to_CAN_bus();
324 }
325
326
327 /**
328   * @brief  LIS3DH initialization error function
329   * @param  None
330   * @retval None
331   */
332 void ErrorSignal_LIS_CAN()
333 {
334   /* Send an error signal on the CAN bus if LIS (accelerometer) is not correctly
       initialized */
335   tx_buffer[0] = 0x00;
336   tx_buffer[1] = 0x00;
337   tx_buffer[2] = 0x00;
338   tx_buffer[3] = 0x00;
339   tx_buffer[4] = 0x00;
340   tx_buffer[5] = 0x00;
341   tx_buffer[6] = 0x00;
342   tx_buffer[7] = 0x03;
343
344   send_data_to_CAN_bus();
345 }
346 #endif
347
348
349
350 #if defined(SLAVE_MODE)
351 /**
352   * @brief  SLAVE MODE application entry point.
353   * @retval None
354   * @note   Execution of the main program
355   */
356 void loop()
357 {
358   /* Here perform the acquisition from the accelerometer */
359   dataHighres = 0;
360   myIMU.readRegisterInt16( &dataHighres, LIS3DH_OUT_X_L );
361   myIMU.readRegisterInt16( &dataHighres, LIS3DH_OUT_Z_L );
362   AccAsseZ = myIMU.axisAccel( Z );     // store the data in floating point
363   AccAsseZ_mg = (int32_t)(AccAsseZ * 1000);
364
365   /* Perform the acquisition from the barometer */
366   Temp_Meas = bmp.readTemperature();
367   Press_Meas = bmp.readPressure();
```

```
368
369   /* Send the data */
370   _radioData.PressureTx = Press_Meas;
371   _radioData.AccelerationTx = AccAsseZ_mg;
372
373   /* By default, 'send' transmits data and waits for an acknowledgement.  If no
          acknowledgement is received,
374    * it will try again up to 16 times.  You can also perform a NO_ACK send that does not
          request an acknowledgement.
375    * The data packet will only be transmitted a single time so there is no guarantee it
          will be successful.  Any random
376    * electromagnetic interference can sporatically cause packets to be lost, so NO_ACK
          sends are only suited for certain
377    * types of situations, such as streaming real-time data where performance is more
          important than reliability.
378    *   _radio.send(DESTINATION_RADIO_ID, &_radioData, sizeof(_radioData), NRFLite::NO_ACK)
379    *   _radio.send(DESTINATION_RADIO_ID, &_radioData, sizeof(_radioData), NRFLite::
          REQUIRE_ACK) // THE DEFAULT
380    */
381
382   if (_radio.send(DESTINATION_RADIO_ID, &_radioData, sizeof(_radioData))) // Note how '&'
          must be placed in front of the variable name.
383   {
384   }
385   else
386   {
387       _radioData.FailedTxCount++;
388   }
389
390   /* At the end, restart the cycle after a certain amount of time (ms) */
391   delay(10);
392 }
393 #endif
394
395
396 #if defined(MASTER_MODE)
397 /**
398   * @brief  MASTER MODE application entry point.
399   * @retval None
400   * @note   Execution of the main program
401   */
402 void loop()
403 {
404     while (_radio.hasData())
405     {
406         _radio.readData(&_radioData); // Note how '&' must be placed in front of the
              variable name
407
408         /* This line of code corrects the received data: in fact, the data is sent using a
               FIFO logic related to a single byte [the result is stored as follows:
409          * MSByte(received data)=LSByte(transmitted data), ... , LSByte(received data)=
               MSByte(transmitted data)].
410          * The following line thus applies a bitmask to obtain one byte of the received
               data; a shift to each byte in order to position it correctly inside the 32-
               bit word;
411          * a sum to merge all the 4 repositioned Bytes together. NB: THIS WORKS ONLY FOR
               32-BIT DATA, CHANGE IT FOR OTHER TYPES OF DATA.
412          * Surprisingly, no bit mask is required working with another ST Nucleo 8S 208RB
413           */
414
415         //_radioData.PressureTx = ((_radioData.PressureTx & 0xFF000000)>>24)+((_radioData.
              PressureTx & 0x00FF0000)>>8)+((_radioData.PressureTx & 0x0000FF00)<<8)+((
              _radioData.PressureTx & 0x000000FF)<<24);
416         //_radioData.AccelerationTx = ((_radioData.AccelerationTx & 0xFF000000)>>24)+((
              _radioData.AccelerationTx & 0x00FF0000)>>8)+((_radioData.AccelerationTx & 0
```

```
            x0000FF00)<<8)+((_radioData.AccelerationTx & 0x000000FF)<<24);
417
418
419         /* Create the bit mask for the PRESSURE data to be transmitted on the CAN bus
420          * int32_t --> 4 x uint8_t  (then, at the receiver size we need to do the same
421          */
422         tx_buffer[0] = (uint8_t)(_radioData.PressureTx);
423         tx_buffer[1] = (uint8_t)(_radioData.PressureTx>>8);
424         tx_buffer[2] = (uint8_t)(_radioData.PressureTx>>16);
425         tx_buffer[3] = (uint8_t)(_radioData.PressureTx>>24);
426         /* Create the bit mask for the ACCELEROMETER data to be transmitted on the CAN bus
427          * int32_t --> 4 x uint8_t  (then, at the receiver size we need to do the same
428          */
429         tx_buffer[4] = (uint8_t)(_radioData.AccelerationTx);
430         tx_buffer[5] = (uint8_t)(_radioData.AccelerationTx>>8);
431         tx_buffer[6] = (uint8_t)(_radioData.AccelerationTx>>16);
432         tx_buffer[7] = (uint8_t)(_radioData.AccelerationTx>>24);
433
434
435         send_data_to_CAN_bus();
436     }
437 }
438 #endif
439
440
441 #if defined(WIRED_SENSOR)
442 /**
443   * @brief  WIRED SENSOR application entry point.
444   * @retval None
445   * @note   Execution of the main program
446   */
447 void loop()
448 {
449   /* Here perform the acquisition from the accelerometer */
450   dataHighres = 0;
451   myIMU.readRegisterInt16( &dataHighres, LIS3DH_OUT_X_L );
452   myIMU.readRegisterInt16( &dataHighres, LIS3DH_OUT_Z_L );
453   AccAsseZ = myIMU.axisAccel( Z );     // store the data in floating point
454   AccAsseZ_mg = (int32_t)(AccAsseZ * 1000);
455   // Print the result, i.e. acceleration in mg
456   Serial.print("Acc. axis Z in mg =");
457   Serial.println(AccAsseZ_mg); // print with 4 digit after comma
458
459   /* Perform the acquisition from the barometer */
460   Temp_Meas = bmp.readTemperature();
461   Press_Meas = bmp.readPressure();
462
463   /* Create the bit mask for the PRESSURE data to be transmitted on the CAN bus
464    * int32_t --> 4 x uint8_t  (then, at the receiver size we need to do the same
465    */
466   tx_buffer[0] = (uint8_t)(Press_Meas);
467   tx_buffer[1] = (uint8_t)(Press_Meas>>8);
468   tx_buffer[2] = (uint8_t)(Press_Meas>>16);
469   tx_buffer[3] = (uint8_t)(Press_Meas>>24);
470   /* Create the bit mask for the ACCELEROMETER data to be transmitted on the CAN bus
471    * int32_t --> 4 x uint8_t  (then, at the receiver size we need to do the same
472    */
473   tx_buffer[4] = (uint8_t)(AccAsseZ_mg);
474   tx_buffer[5] = (uint8_t)(AccAsseZ_mg>>8);
475   tx_buffer[6] = (uint8_t)(AccAsseZ_mg>>16);
476   tx_buffer[7] = (uint8_t)(AccAsseZ_mg>>24);
477
478   send_data_to_CAN_bus();
479 }
480 #endif
```

```
481
482
483
484 #if defined(MASTER_MODE) || defined(WIRED_SENSOR)
485 /**
486   * @brief  CAN sending data function
487   * @param  None
488   * @retval None
489   * @note   This function sends the specified payload on the CAN bus.
490   */
491 void send_data_to_CAN_bus()
492 {
493   CAN_Transmit(0x0001,  CAN_Id_Extended, CAN_RTR_Data, 8, tx_buffer);  /* ID: 0b0000 0100
        1001 0110 */
494   delay(1);
495 }
496 #endif
```

.III CAN Sniffer Source Code - STM32F407G-DISC1

```
1
2         /**
3   ******************************************************************************
4   * @file    main.c
5   * @author  CL1 Team
6   * @version V3.0
7   * @date    09-December-2019
8   * @brief   This file contains the main program used to read data from the CAN
9   *              bus using STM32F4DISC1. The program also sends the data read to the
10  *              serial monitor using USART interface.
11  ******************************************************************************
12  */
13
14
15  /* Includes ------------------------------------------------------------------*/
16  #include "main.h"
17  #include <stdio.h>
18  #include <string.h>
19  #include <stdlib.h>
20
21
22  /* Private variables ---------------------------------------------------------*/
23  CAN_HandleTypeDef hcan1;
24  UART_HandleTypeDef huart2;
25
26  CAN_FilterTypeDef CAN_FilterConfStructure;
27  CAN_TxHeaderTypeDef CAN_TxMessage;
28  CAN_RxHeaderTypeDef CAN_RxMessage;
29
30  char accString[40];
31  char pressureString[40];
32
33  uint8_t messageReceived = 0;
34  uint8_t receiveBuffer[8] = {0, 0, 0, 0, 0, 0, 0, 0};
35  int32_t accelerometerData = 0;
36  int32_t pressureData = 0;
37  int32_t pressureDataTemp = 0;
38
39
40  /* Private function prototypes -----------------------------------------------*/
41  void SystemClock_Config(void);
42  static void MX_GPIO_Init(void);
43  static void MX_CAN1_Init(void);
44  static void MX_USART2_UART_Init(void);
45
46  void CAN_Config(void);
47  void debugPrint(UART_HandleTypeDef *huart, char out[]);
48
49
50  /**
51    * @brief  The application entry point.
52    * @retval int
53    */
54  int main(void)
55  {
56    /* MCU Configuration--------------------------------------------------------*/
57    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
58    HAL_Init();
59
60    /* Configure the system clock */
61    SystemClock_Config();
62
```

```
63    /* Initialize all configured peripherals */
64    MX_GPIO_Init();
65    MX_CAN1_Init();
66    MX_USART2_UART_Init();
67
68    CAN_Config();
69
70    /* Infinite loop */
71    while (1)
72    {
73
74                  /* In the infinite loop the program just waits for an interrupt to rise,
75                   * receives the message and outputs the data via USART.
76                   */
77
78                  /* Receiving the message from the input FIFO */
79                  HAL_CAN_GetRxMessage(&hcan1, CAN_RX_FIFO0, &CAN_RxMessage, receiveBuffer);
80
81                  /* Decoding the data and copying it into the corresponding variables */
82                  pressureDataTemp = (int32_t) (receiveBuffer[0] + (((int32_t) receiveBuffer
                        [1]) << 8) + (((int32_t) receiveBuffer[2]) << 16) + (((int32_t)
                        receiveBuffer[3]) << 24));
83                  accelerometerData = (int32_t) (receiveBuffer[4] + (((int32_t)
                        receiveBuffer[5]) << 8) + (((int32_t) receiveBuffer[6]) << 16) + ((
                        int32_t) receiveBuffer[7]) << 24));
84
85                  if(pressureDataTemp != pressureData)
86                  {
87                          pressureData = pressureDataTemp;
88                          itoa(accelerometerData, accString, 10);
89                          itoa(pressureData, pressureString, 10);
90                          debugPrint(&huart2, accString);
91                          debugPrint(&huart2, ",␣");
92                          debugPrint(&huart2, pressureString);
93                          debugPrint(&huart2, ";\n");
94                  }
95
96    }
97
98  }
99
100
101 /**
102   * @brief System Clock Configuration
103   * @retval None
104   */
105 void SystemClock_Config(void)
106 {
107   RCC_OscInitTypeDef RCC_OscInitStruct = {0};
108   RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
109   RCC_PeriphCLKInitTypeDef PeriphClkInitStruct = {0};
110
111   /** Configure the main internal regulator output voltage
112   */
113   __HAL_RCC_PWR_CLK_ENABLE();
114   __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
115   /** Initializes the CPU, AHB and APB busses clocks
116   */
117   RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
118   RCC_OscInitStruct.HSEState = RCC_HSE_ON;
119   RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
120   RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
121   RCC_OscInitStruct.PLL.PLLM = 8;
122   RCC_OscInitStruct.PLL.PLLN = 288;
123   RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
```

```
124     RCC_OscInitStruct.PLL.PLLQ = 6;
125     if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
126     {
127       Error_Handler();
128     }
129     /** Initializes the CPU, AHB and APB busses clocks
130     */
131     RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
132                                 |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
133     RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
134     RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
135     RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
136     RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
137
138     if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_4) != HAL_OK)
139     {
140       Error_Handler();
141     }
142     PeriphClkInitStruct.PeriphClockSelection = RCC_PERIPHCLK_I2S;
143     PeriphClkInitStruct.PLLI2S.PLLI2SN = 192;
144     PeriphClkInitStruct.PLLI2S.PLLI2SR = 2;
145     if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInitStruct) != HAL_OK)
146     {
147       Error_Handler();
148     }
149   }
150
151
152   /**
153     * @brief CAN1 Initialization Function
154     * @param None
155     * @retval None
156     */
157   static void MX_CAN1_Init(void)
158   {
159
160     hcan1.Instance = CAN1;
161     hcan1.Init.Prescaler = 9;
162     hcan1.Init.Mode = CAN_MODE_NORMAL;
163     hcan1.Init.SyncJumpWidth = CAN_SJW_1TQ;
164     hcan1.Init.TimeSeg1 = CAN_BS1_8TQ;
165     hcan1.Init.TimeSeg2 = CAN_BS2_7TQ;
166     hcan1.Init.TimeTriggeredMode = DISABLE;
167     hcan1.Init.AutoBusOff = DISABLE;
168     hcan1.Init.AutoWakeUp = DISABLE;
169     hcan1.Init.AutoRetransmission = DISABLE;
170     hcan1.Init.ReceiveFifoLocked = DISABLE;
171     hcan1.Init.TransmitFifoPriority = DISABLE;
172
173     if (HAL_CAN_Init(&hcan1) != HAL_OK)
174     {
175       Error_Handler();
176     }
177
178   }
179
180
181   /**
182     * @brief USART2 Initialization Function
183     * @param None
184     * @retval None
185     */
186   static void MX_USART2_UART_Init(void)
187   {
188
```

```
189   huart2.Instance = USART2;
190   huart2.Init.BaudRate = 115200;
191   huart2.Init.WordLength = UART_WORDLENGTH_8B;
192   huart2.Init.StopBits = UART_STOPBITS_1;
193   huart2.Init.Parity = UART_PARITY_NONE;
194   huart2.Init.Mode = UART_MODE_TX_RX;
195   huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
196   huart2.Init.OverSampling = UART_OVERSAMPLING_16;
197
198   if (HAL_UART_Init(&huart2) != HAL_OK)
199   {
200     Error_Handler();
201   }
202
203 }
204
205
206 /**
207   * @brief GPIO Initialization Function
208   * @param None
209   * @retval None
210   */
211 static void MX_GPIO_Init(void)
212 {
213   GPIO_InitTypeDef GPIO_InitStruct = {0};
214
215   /* GPIO Ports Clock Enable */
216   __HAL_RCC_GPIOE_CLK_ENABLE();
217   __HAL_RCC_GPIOC_CLK_ENABLE();
218   __HAL_RCC_GPIOH_CLK_ENABLE();
219   __HAL_RCC_GPIOA_CLK_ENABLE();
220   __HAL_RCC_GPIOB_CLK_ENABLE();
221   __HAL_RCC_GPIOD_CLK_ENABLE();
222
223   /*Configure GPIO pin Output Level */
224   HAL_GPIO_WritePin(OTG_FS_PowerSwitchOn_GPIO_Port, OTG_FS_PowerSwitchOn_Pin, GPIO_PIN_SET
          );
225
226   /*Configure GPIO pin Output Level */
227   HAL_GPIO_WritePin(GPIOD, LD4_Pin|LD3_Pin|LD5_Pin|LD6_Pin
228                           |Audio_RST_Pin, GPIO_PIN_RESET);
229
230   /*Configure GPIO pin : PDM_OUT_Pin */
231   GPIO_InitStruct.Pin = PDM_OUT_Pin;
232   GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
233   GPIO_InitStruct.Pull = GPIO_NOPULL;
234   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
235   GPIO_InitStruct.Alternate = GPIO_AF5_SPI2;
236   HAL_GPIO_Init(PDM_OUT_GPIO_Port, &GPIO_InitStruct);
237
238   /*Configure GPIO pin : B1_Pin */
239   GPIO_InitStruct.Pin = B1_Pin;
240   GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
241   GPIO_InitStruct.Pull = GPIO_NOPULL;
242   HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);
243
244   /*Configure GPIO pin : BOOT1_Pin */
245   GPIO_InitStruct.Pin = BOOT1_Pin;
246   GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
247   GPIO_InitStruct.Pull = GPIO_NOPULL;
248   HAL_GPIO_Init(BOOT1_GPIO_Port, &GPIO_InitStruct);
249
250   /*Configure GPIO pin : CLK_IN_Pin */
251   GPIO_InitStruct.Pin = CLK_IN_Pin;
252   GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
```

```
253   GPIO_InitStruct.Pull = GPIO_NOPULL;
254   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
255   GPIO_InitStruct.Alternate = GPIO_AF5_SPI2;
256   HAL_GPIO_Init(CLK_IN_GPIO_Port, &GPIO_InitStruct);
257
258   /*Configure GPIO pins : LD4_Pin LD3_Pin LD5_Pin LD6_Pin
259                            Audio_RST_Pin */
260   GPIO_InitStruct.Pin = LD4_Pin|LD3_Pin|LD5_Pin|LD6_Pin
261                         |Audio_RST_Pin;
262   GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
263   GPIO_InitStruct.Pull = GPIO_NOPULL;
264   GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
265   HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);
266
267   /*Configure GPIO pin : MEMS_INT2_Pin */
268   GPIO_InitStruct.Pin = MEMS_INT2_Pin;
269   GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
270   GPIO_InitStruct.Pull = GPIO_NOPULL;
271   HAL_GPIO_Init(MEMS_INT2_GPIO_Port, &GPIO_InitStruct);
272
273 }
274
275
276 /**
277   * @brief CAN peripheral setup
278   * @param None
279   * @retval None
280   */
281 void CAN_Config(void)
282 {
283
284         /* Filter configuration - accept all data*/
285         CAN_FilterConfStructure.FilterFIFOAssignment = 0;
286         CAN_FilterConfStructure.FilterMode = CAN_FILTERMODE_IDMASK;
287         CAN_FilterConfStructure.FilterScale = CAN_FILTERSCALE_32BIT;
288         CAN_FilterConfStructure.FilterIdHigh = 0x0000;
289         CAN_FilterConfStructure.FilterIdLow = 0x0000;
290         CAN_FilterConfStructure.FilterMaskIdHigh = 0x0000;
291         CAN_FilterConfStructure.FilterMaskIdLow = 0x0000;
292         CAN_FilterConfStructure.FilterActivation = ENABLE;
293
294         if(HAL_CAN_ConfigFilter(&hcan1, &CAN_FilterConfStructure) != HAL_OK)
295         {
296           /* Filter configuration Error */
297           Error_Handler();
298         }
299
300         HAL_CAN_Start(&hcan1);
301 }
302
303
304 /**
305   * @brief Serial monitor print redirect from USART function
306   * @param None
307   * @retval None
308   */
309 void debugPrint(UART_HandleTypeDef *huart, char out[]){
310
311         HAL_UART_Transmit(huart, (uint8_t*) out, strlen(out), 10);
312
313 }
314
315
316 /**
317   * @brief  This function is executed in case of error occurrence.
```

```
318   * @retval None
319   */
320 void Error_Handler(void)
321 {
322   /* USER CODE BEGIN Error_Handler_Debug */
323   /* User can add his own implementation to report the HAL error return state */
324
325   /* USER CODE END Error_Handler_Debug */
326 }
327
328
329 #ifdef  USE_FULL_ASSERT
330 /**
331   * @brief  Reports the name of the source file and the source line number
332   *         where the assert_param error has occurred.
333   * @param  file: pointer to the source file name
334   * @param  line: assert_param error line source number
335   * @retval None
336   */
337 void assert_failed(uint8_t *file, uint32_t line)
338 {
339   /* USER CODE BEGIN 6 */
340   /* User can add his own implementation to report the file name and line number,
341      tex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
342   /* USER CODE END 6 */
343 }
344 #endif /* USE_FULL_ASSERT */
```

Appendix D: Datasheets

Evaluation Board

STM8A-DISCOVERY

STM8A-DISCOVERY
Discovery kit for STM8A microcontrollers

Data brief

Features

STM8AF and STM8AL common features:

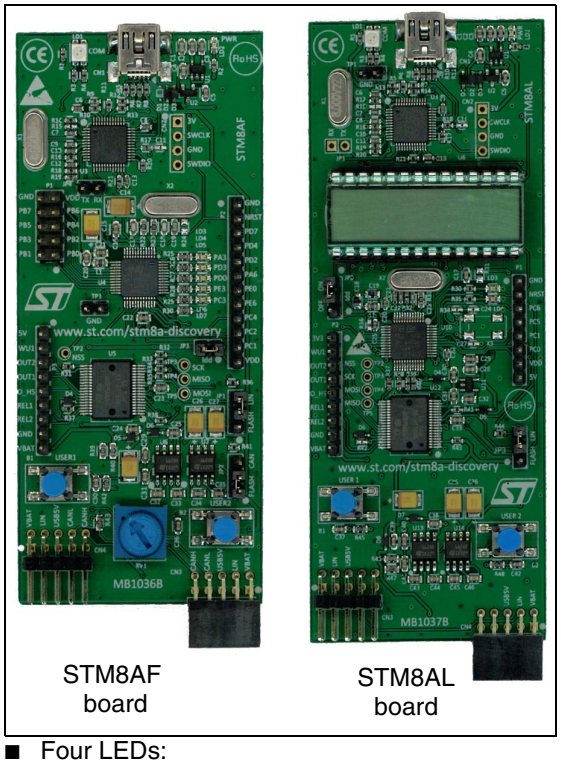
- On-board ST-LINK/V2 included for debugging and programming
- Board power supply: through 5 V USB bus
- Internal dual ST662A step-up converter building the 12 Vdc when powered by USB port
- External application power supply V_{BAT} (up to 14 Vdc)
- 16 MHz HSE XTAL crystal oscillator
- L99PM62GXP power management IC with LIN and high speed CAN with SPI control interface and high-side drivers
- Two push buttons (USER1 and USER2)
- Extension header for L99PM62GXP including relays, high-side outputs and wake-up capabilities

STM8AF dedicated features:

- STM8AF5288T microcontroller featuring 64 Kbytes Flash, 2 Kbytes data EEPROM, LIN, CAN in an 48-pin package
- Seven LEDs:
  - LD1 (red/green) for USB communication
  - LD2 (red) for 5 V power ON
  - Five user LEDs LD3 (red) and LD4 to LD7 (green)
- RV1 potentiometer connected to the ADC peripheral
- Extension headers for MCU connectivity (full Port B, free ports pins, RESET)

STM8AL dedicated features:

- STM8AL3L68T microcontroller featuring 32 Kbytes Flash, 1 Kbytes data EEPROM, LCD in an 48-pin package



STM8AF board

STM8AL board

- Four LEDs:
  - LD1 (red/green) for USB communication
  - LD2 (red) for 3.3 V power ON
  - 2 user LEDs LD3 (red) and LD4 (green)
- 4-digit alphanumeric LCD display including 4 bars display
- Extension header for MCU connectivity (free ports pins, RESET)

Table 1. Device summary

Order code	Reference
STM8A-DISCOVERY	STM8A discovery kit

MCU - STM8AF5288T

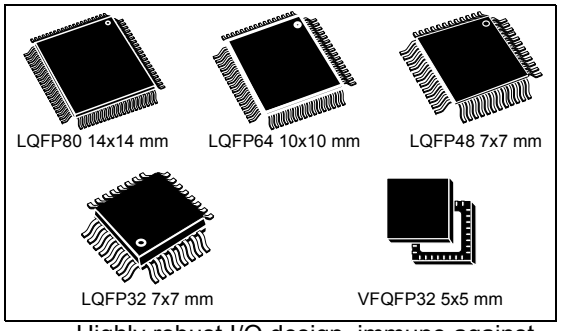
STM8AF526x/8x/Ax
STM8AF6269/8x/Ax

Automotive 8-bit MCU, with up to 128 Kbyte Flash, data EEPROM, 10-bit ADC, timers, LIN, CAN, USART, SPI, I2C, 3 to 5.5 V

Datasheet - production data

Features

- AEC-Q10x qualified
- Core
  - Max f_{CPU}: 24 MHz
  - Advanced STM8A core with Harvard architecture and 3-stage pipeline
  - Average 1.6 cycles/instruction resulting in 10 MIPS at 16 MHz f_{CPU} for industry standard benchmark
- Memories
  - Program memory: 32 to 128 Kbyte Flash program; data retention 20 years at 55 °C
  - Data memory: up to 2 Kbyte true data EEPROM; endurance 300 kcycle
  - RAM: 6 Kbyte
- Clock management
  - Low-power crystal resonator oscillator with external clock input
  - Internal, user-trimmable 16 MHz RC and low-power 128 kHz RC oscillators
  - Clock security system with clock monitor
- Reset and supply management
  - Wait/auto-wakeup/Halt low-power modes with user definable clock gating
  - Low consumption power-on and power-down reset
- Interrupt management
  - Nested interrupt controller with 32 vectors
  - Up to 37 external interrupts on 5 vectors
- Timers
  - 2 general purpose 16-bit timers with up to 3 CAPCOM channels each (IC, OC, PWM)
  - Advanced control timer: 16-bit, 4 CAPCOM channels, 3 complementary outputs, dead-time insertion and flexible synchronization
  - 8-bit AR basic timer with 8-bit prescaler
  - Auto-wakeup timer
  - Window and independent watchdog timers
- I/Os
  - Up to 68 user pins (11 high sink I/Os)



  - Highly robust I/O design, immune against current injection
- Communication interfaces
  - High speed 1 Mbit/s CAN 2.0B interface
  - USART with clock output for synchronous operation - LIN master mode
  - LINUART LIN 2.2 compliant, master/slave modes with automatic resynchronization
  - SPI interface up to 10 Mbit/s or $f_{MASTER}/2$
  - I^2C interface up to 400 Kbit/s
- Analog to digital converter (ADC)
  - 10-bit resolution, 2 LSB TUE, 1 LSB linearity and up to 16 multiplexed channels
- Operating temperature up to 150 °C
- Qualification conforms to AEC-Q100 grade 0

Table 1. Device summary(1)

Reference	Part number
STM8AF526x/8x/Ax (with CAN)	STM8AF5268, STM8AF5269, STM8AF5286, STM8AF5288, STM8AF5289, STM8AF528A, STM8AF52A6, STM8AF52A8, STM8AF52A9, STM8AF52AA
STM8AF6269/8x/Ax	STM8AF6269, STM8AF6286, STM8AF6288, STM8AF6289, STM8AF628A, STM8AF62A6, STM8AF62A8, STM8AF62A9, STM8AF62AA

1. In the order code, 'F' applies to devices with Flash program memory and data EEPROM. 'F' is replaced by 'P' for devices with FASTROM (see *Table 2*, *Table 3* and *Figure 60*).

Barometric Pressure Sensor - BMP388

BMP388

Digital pressure sensor

BMP388 – Datasheet

Document revision	1.1
Document release date	March 2018
Document number	BST-BMP388-DS001-01
Technical reference code(s)	0 273 300 511
Notes	Data and descriptions in this document are subject to change without notice. Product photos and pictures are for illustration purposes only and may differ from the real product appearance.

BMP388

Digital pressure sensor

The BMP388 is a digital sensor with pressure and temperature measurement based on proven sensing principles. The sensor module is housed in an extremely compact 10-pin metal-lid LGA package with a footprint of only 2.0 × 2.0 mm² and max 0.8 mm package height. Its small dimensions and its low power consumption of 3.4 µA @1Hz allow the implementation in battery driven devices such as mobile phones, GPS modules or watches.

Typical applications

- Vertical velocity indication (e.g. rise/sink speed)
- Internet of things
- Enhancement of GPS navigation
  (e.g. time-to-first-fix improvement, dead-reckoning, slope detection)
- Indoor navigation & localization (floor detection, elevator detection)
- Outdoor navigation, leisure and sports applications
- Weather forecast
- Health care applications (e.g. spirometry)
- Fitness applications like enhancement of calorie detection
- AR & VR applications
- Context awareness

Target Devices

- Flying toys
- Drones
- Handsets such as mobile phones, tablet PCs, GPS devices
- Navigation systems
- Portable health care devices
- Home weather stations
- Watches
- White goods

Key features

Table 1: Key Features of BMP388

Package	2.0 mm x 2.0 mm x 0.75 mm metal lid LGA
Digital interface	I²C (up to 3.4 MHz) and SPI (3 and 4 wire, up to 10 MHz)
Supply voltage	V_{DD} main supply voltage range: 1.65 V to 3.6 V V_{DDIO} interface voltage range: 1.2 V to 3.6 V
Relative accuracy	typ. ± 8 Pa, equiv. to ± 0.66 m (900 ... 1100 hPa, 25 . . . 40 °C)
Absolute accuracy	typ. ± 50 Pa (300 ...1100 hPa, -20 ...+65 °C)
Temperature coefficient offset	typ. ± 0.75 Pa/K (0 ... 55°C @700 -1100 hPa)
Current consumption	3.4 µA at 1 Hz pressure and temperature 2.0 µA in sleep mode
Operating range	-40 – +85 °C, 300–1250 hPa
The product is RoHS compliant, halogen-free, MSL1	

BMP388 enables accurate altitude tracking and is specifically suited for drone applications. The best-in-class TCO between -20-65°C for accurate altitude measurement over a wide temperature range of the BMP388 greatly enhance the drone flying experience by making accurate steering easier. It is compatible for use with other Bosch sensors, including the new BMI088 for better performance, robustness and stability. The new BMP388 sensor offers outstanding design flexibility, providing a single package solution that can also be easily integrated into other existing and upcoming devices such as smart homes, industrial products and wearables.

The sensor is more accurate than its predecessor BMP280, covering a wide measurement range from 300 hPa to 1250 hPa.

This new barometric pressure sensor exhibits an attractive price-performance ratio coupled with low power consumption.

It is available in a compact 10-in 2.0 x 2.0 x 0.75 mm³ LGA package with metal lid

Due to the built-in hardware synchronization of the pressure sensor data and its ability to synchronize data from external devices such as acceleration sensors, the BMP388 is ideally suited for fitness and navigation applications which require highly accurate, low power and low latency sensor data fusion.

The new interrupt functionality provides simple access to data and storage. Examples of interrupts than can be used in a power efficient manner without using software algorithms include: Data ready interrupt, watermark interrupt (on byte level) or FIFO full interrupt.

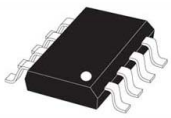
BMP388 also includes a new FIFO functionality. This greatly improves ease of use while helping to reduce power consumption of the overall device system during full operation. The integrated 512 byte FIFO buffer supports low power applications and prevents data loss in non-real-time systems.

One - Axis Accelerometer - AIS1120SX

AIS1120SX
AIS2120SX

MEMS automotive acceleration sensor: single/dual-axis with SPI interface

Datasheet - production data



SOIC8

Features

- AEC-Q100 qualified
- 3.3 V single supply operation
- 14-bit data output
- ±120 *g* full scale
- Slow and fast offset cancellation
- Embedded self-test
- Selectable low-pass filter
- SPI interface
- ECOPACK® compliant
- Extended temperature range -40 °C to +105 °C

Applications

- Airbag systems
- Vibrations, impact monitoring

Description

The AIS1120SX / AIS2120SX is a central acceleration sensor with a single or dual axis sensing element and an IC interface able to provide acceleration information to a master control unit via an SPI protocol.

The sensing element, capable of detecting the acceleration, is manufactured using a dedicated process developed by ST to produce inertial sensors and actuators in silicon.

The IC interface is manufactured using a BCD process that allows a high level of integration. The device is factory trimmed to better tune the characteristics of the sensing element with the acceleration information to be supplied.

The AIS1120SX / AIS2120SX has a full scale of ±120 *g*. The acquisition chain consists of a C/V converter, a full-differential charge amplifier, a 2nd order ΣΔ analog-to-digital converter and a digital core, which includes filtering, compensation and interpolation, control logic and SPI protocol generation.

The differential capacitance of the sensor is proportional to the proof mass displacement; thus, by sensing the differential capacitance, the position of the sensor is determined. Then, since the mass position is known, and the position is related to the input acceleration, the input acceleration can be easily deduced.

The device is available in a plastic SOIC8 package and is guaranteed to operate over a temperature range extending from -40 °C to +105 °C.

Table 1. Device summary

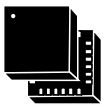
Order code	*g*-range	Sensitivity axes	Operating temperature range [°C]	Package	Packing
AIS1120SXTR	120 *g*	x	-40 to +105	SOIC8N	Tape and reel
AIS2120SXTR	120 *g*	xy	-40 to +105	SOIC8N	Tape and reel

Three -Axis Accelerometer - AIS3624DQ

AIS3624DQ

High-performance motion sensor for automotive applications: ultra-low-power digital output 3-axis accelerometer

Datasheet - production data



QFN 24 (4x4x1.8 mm3)

Features

- Wide supply voltage range: 2.4 V to 3.6 V
- 1.8 V low voltage compatible IOs
- Ultra-low-power mode consumption down to 10 µA
- ±6*g*/±12*g*/±24*g* dynamically selectable full scale
- SPI/I^{2}C digital output interface
- 16-bit data output, 12-bit resolution
- 2 independent programmable interrupt generators
- System sleep-to-wake function
- Embedded self-test
- Extended temperature range -40°C to 105°C
- 10000 *g* high shock survivability
- ECOPACK®, RoHS and “Green” compliant (see *Section 8*)
- AEC-Q100 qualification

Description

The AIS3624DQ is an ultra-low-power high-performance three-axis accelerometer with a digital serial interface SPI standard output, an I^{2}C compatible interface is also available.

The device features ultra-low-power operational modes that allow advanced power saving and smart sleep-to-wake functions.

The AIS3624DQ has dynamically user selectable full scales of ±6*g*/±12*g*/±24*g* and it is capable of measuring accelerations with output data rates from 0.5 Hz to 1 kHz.

The self-test capability allows the user to check the functioning of the sensor in the final application.

The device may be configured to generate an interrupt signal by inertial wakeup/free-fall events as well as by the position of the device itself. Thresholds and timing of interrupt generators are programmable by the end user on the fly.

The AIS3624DQ is available in small, quad flat no-lead package (QFN) with the reduced 4x4 mm footprint required by many applications and it is guaranteed to operate over an extended temperature range from -40 °C to +105 °C.

This product may be used in a variety of automotive non-safety applications such as:

- Motion-activated functions
- Telematic boxes
- Impact recognition and logging systems
- Vibration monitoring and compensation

Table 1. Device summary

Order codes	Temperature range [°C]	Package	Packaging
AIS3624DQ	-40 to +105	QFN 4x4x1.8 24L	Tray
AIS3624DQTR	-40 to +105	QFN 4x4x1.8 24L	Tape and reel

Appendix E: Validation Documents

Test 2_MCU: Source Code

```
#include "stm8s.h"

#define F_CPU 8000000UL

#include "delay.h"

#define DELAY_MS 100

#define VREF 3.3
#define ADC_SLOPE (VREF/1024)
#define BATT_DIV_R1 10
#define BATT_DIV_R2 2.15
#define V_BATT_GAIN ((BATT_DIV_R1+BATT_DIV_R2)/(BATT_DIV_R2))



/*********** Declarations *************/
/*------ Function prototypes ---------*/

//void ADC_INT_Config(void);


/* ----- Global variables ----------- */
volatile uint16_t ADC_Value;

int dataReady = 0;

float volt = 0, v_batt = 0;

int main(void)
{

CLK_HSECmd(ENABLE);
CLK_ClockSwitchCmd(ENABLE);
CLK_SYSCLKConfig(CLK_PRESCALER_CPUDIV1);
CLK_PeripheralClockConfig(CLK_PERIPHERAL_ADC, ENABLE);
CLK_PeripheralClockConfig(CLK_PERIPHERAL_CAN, ENABLE);
CLK_PeripheralClockConfig(CLK_PERIPHERAL_SPI, ENABLE);


/* Configure and start ADC */
ADC2_Init( ADC2_CONVERSIONMODE_CONTINUOUS,
ADC2_CHANNEL_0,
ADC2_PRESSEL_FCPU_D2,
ADC2_EXTTRIG_TIM,
DISABLE,
ADC2_ALIGN_RIGHT,
ADC2_SCHMITTTRIG_CHANNEL0,
DISABLE );



ADC2_StartConversion();

while(1)
{

ADC_Value = ADC2_GetConversionValue();
volt = ADC_Value*ADC_SLOPE;
```

```
v_batt = volt*V_BATT_GAIN;
//_delay_ms(DELAY_MS);
}
}
```

FMEA

	Project name: CL1_custom		Group name: GROUP4			Date: 12/11/2019							
Failure Mode and Effect Analysis CL1 controller													
	Function	**Failure Mode**	**Failure Cause**	**Failure Effect**	**Sev**	**Severity Reasoning**	**Occ**	**Occurrence Reasoning**	**Failure Detection**	**Det**	**Detection Reasoning**	**Risk**	**Failure Handling**
1	Power supply	Power supply failure	Battery disconnection	Board is not supplied	1	Battery is disconnected	1	Connector is not properly connected	None of the components is supplied.	1	None of the components is supplied.	Low	Battery has to be reconnected
2	Power supply	Power supply failure	Cable desoldering	Board is not supplied	2	Battery is not properly connected	1	Board is subjected to high vibrations	None of the components is supplied.	1	None of the components is supplied.	Low	The board has to be demounted from the vehicle in order to allow the cable soldering
3	Power supply	Power supply failure	Battery cell stress	Battery cell overheating	3	Battery cell damage	2	Cell collision due to box protection loss (ie the box throw from the vehicle)	Battery inflates and/or increase in the temperature	2	Obvious problem to the system such as the fact that it is subjected to a large force (fall to the ground)	High	The battery cell is damaged and it cannot be recovered. Shortcircuit with a load the cell in order to allow the complete discharge
4	Power supply	Power supply failure	Battery cell undervoltage	Board is not supplied	3	Undervoltage condition	1	SEPIC converter is shut down too late	Battery cell voltage has to be checked	3	Voltage is too low.	High	The battery cell is damaged and it cannot be recovered. Shortcircuit with a load the cell in order to allow the complete discharge
5	Wireless module	Failure in the wireless connection	TX failure	Sensor data are not trasmitted to the control board	1	No data trasmission	2	Jamming on the selected channel is present or interference from other devices in the same ISM band	No data are received	2	Data is not received in the , so packet with an error message is sent on the CAN line.	Low	Data will be retrasmitted by the TX
6	Wireless module	Failure in the wireless connection	RX failure	Sensor data are not received by the control board	1	No data reception	2	Jamming on the selected channel is present or interference from other devices in the same ISM band	No data are received	2	Data is not received, so packet with an error message is sent on the CAN line.	Low	Data will be retrasmitted by the TX
7	Wireless module	Failure in the wireless connection	TX or RX module are broken or desoldered	Sensor data are not received by the control board and/or transmitted by the sensor board	2	No data reception/transmissio n	1	Board is subjected to high vibrations	No data are received or sent	1	Data is not received, so packet with an error message is sent on the CAN line and visual inspection of the board	Low	System must be reconfigured as a wired system or the wireless module has to be replaced
8	CAN connection	Failure in the wired connection	Wiring failure	No data monitoring is possible	2	No data visualization in the user interface	1	Cable damage due to shear stress	No system CAN ID is sending messages on the line	1	Visual inspection of the wiring	Low	Desolder the damaged cables and solder new ones
10	Barometric pressure sensor	Digital communication between sensor and receiving MCU is corrupted (e.g. bits change due to EMI)	Presence of interfering sources	Data are not correct or even not received	1	Data are not correct	2	Too much sources transmitting in gthe same ISM band. The circuit may be susceptible	Wrong behaviour is unexpected and may be occasional	3		Low	Try the system in a controlled environment, free from interferers.

11	Barometric pressure sensor	Sensor values not plausible	Dust on the sensor	Received data are not as expected	1	Data are not correct	2	Pollution in the air	Visual inspection of the sensor		Visual inspection of the sensor	Low	
12	Accelerometer	Digital communication between sensor and receiving MCU is corrupted (e.g. bits change due to EMI)	Presence of interfering sources	Data are not correct or even not received	1	Data are not correct	2	Too much sources transmitting in gthe same ISM band. The circuit may be susceptible	Wrong behaviour is unexpected and may be occasional	3		Low	Try the system in a controlled environment, free from interferers.
14	All the system	Short circuit between VDD and GND	Environmental factors	Short circuit between VDD and GND	4	Damage of all the components	1	V_BATT and GND terminal in contact, dust or water on the circuit	Possible fire arises	1	Look at the status of the board	High	Impossible to recover easily. Component functioning has to be tested and eventually a new board has to be soldered and tested

Severity	Occurance	Detection
1: Leads to partial or complete loss of functionalities, with easy recovery.	1: Unlikely to occur	1: Very easy detection
2: Leads to partial or complete loss of functionalities, with relative easy recovery.	2: Occures quite frequently	2: Quite easy to be detected
3: Leads to complete loss of functionalities, with hard recovery	3: Occures frequently	3: Difficult to be detected
4: Leads to complete loss of functionalities, impossible to be recovered	4: Sure occurence	4: Impossible to be detected

Appendix F: Weekly Reports

F1 SMART CONTROLLER WEEKLY REPORT – WEEK 41 Date : 11/10/2019

MAIN ACTIVITIES

- Brainstorming with team members for ideas;
- Initial evaluation of the requirements;
- Project planning activities definition;
- Roles and responsibilities definition;
- Questions and doubts for new week progress meeting.

NEXT STEPS

- Complete definition of the architecture;
- Sensor evaluation.

ISSUES

- Confusion on some choices that have to be taken;
- Difficult to clearly assign tasks to the team members.

MAIN DECISIONS TAKEN

- Using a wired connection to the vehicle;
- A single board will be designed, supplied by the vehicle battery;
- PCB to be printed.

KEY LEARNINGS

- Planning the different phases of the project leads to several advantages;
- Each member has to be aware of his role and of what he has to do.

F1 SMART CONTROLLER WEEKLY REPORT – WEEK 42 Date : 18/10/2019

MAIN ACTIVITIES

- Definition of the architecture's main features;
- Definition of the main project management activities;
- First draft of documentation redaction;
- Initial definition of which kind of components are needed.

NEXT STEPS

- Completion of requirement analysis and choice of components;
- BOM redaction (24/10/2019 dealine);
- Circuit simulation where needed;
- Study of software tools.

ISSUES

- Lack of time for component choice and architecture definition could lead to design errors;
- Taking into account all the problems that can be present when the system will be mounted on the vehicle.

MAIN DECISIONS TAKEN

- Wireless technology is used to connect the sensor and the control board;
- Modular solution;
- SEPIC as power supply converter.

KEY LEARNINGS

- A lot of factors have to be considered while choosing the architecture and components;
- Time constraints result in critical constraints during project development.

F1 SMART CONTROLLER WEEKLY REPORT – WEEK 43 Date : 25/10/2019

MAIN ACTIVITIES

- Completion of BOM, all the components have been defined;
- Simulation on SEPIC converter and polarity inversion protection;
- Redaction of documentation related to hardware choices;
- Modification of the architecture of the system.

NEXT STEPS

- FW development: evaluation board/sensor communication;
- FW development: CAN communication;
- HW design: circuit design and PCB layout.

ISSUES

- After the progress meeting on October 17 th 2019 the architecture of the system has been uptated due to the fact that wireless solution has been strongly advised;

MAIN DECISIONS TAKEN

- SEPIC converter;
- NRF24L01+ wireless module;
- LiPo cell 3.7 V 150 mAh as sensor board supply.

KEY LEARNINGS

- During project development some requirements that change the project architeture can completely modify and invalidate what has been done.

F1 SMART CONTROLLER WEEKLY REPORT – WEEK 44

Date : 01/11/2019

MAIN ACTIVITIES

- Check of the components that have been delivered;
- Progress in FW development: first communication with accelerometer LIS3DH;
- Progress in HW design: Eagle schematic completion;
- Redaction of the documentation related to HW design.

NEXT STEPS

- HW design: PCB layout;
- FW development: pressure sensor communication;
- FW development: CAN communication.

ISSUES

- Problems with the BMP388 pressure sensor related to compiler configuration;
- The supplier didn't send the accelerometer for the custom board because it is out of stock.

MAIN DECISIONS TAKEN

- The BMP388 could be substituted with another sensor, if available;
- Substituing AIS1120SX with AIS2120SX.

KEY LEARNINGS

- All components' datasheets have to be checked carefully to verify the compatibility among different devices;
- PCB routing requires patience and attention.

F1 SMART CONTROLLER WEEKLY REPORT – WEEK 45 Date : 08/11/2019

MAIN ACTIVITIES

- FW development: NRF24 communication;
- FW development: CAN communication;
- HW design: PCB layout completion;
- Checked GERBER files and production of the PCB;
- Documentation redaction;

NEXT STEPS

- Questions for the next progress meeting on November 14th 2019;
- PCB soldering and testing;
- FW development: filtering and signal conditioning definition;
- Adapt FW for the custom board.

ISSUES

- Overhead in the final check of the PCB layout and GERBER files;
- Difficulties in interfacing with NRF24;

MAIN DECISIONS TAKEN

- Correctness of the PCB layout;
- Outsourcing of PCB production;

KEY LEARNINGS

- Extensive documentation is required in order to justify and explain the project work;
- CAD and software tools make some difficult tasks easy to perform in short amounts of time.

F1 SMART CONTROLLER WEEKLY REPORT – WEEK 46 Date : 15/11/2019

MAIN ACTIVITIES

- Documentation redaction;
- FW development: wireless communication;
- Completion of the FW for the evaluation board;
- Design of the enclosure for the custom board;
- Air flow simulations on the wing with the box.

NEXT STEPS

- PCB soldering and testing;
- FW development: filtering and signal conditioning definition;
- Improve the design of the enclosure;
- Adapt FW for custom board.

ISSUES

- Pressure sensor LPS22H cannot be found easily;
- Difficulties in the management of the task to be assigned to each member;

MAIN DECISIONS TAKEN

- Enclosure design ready for 3D printing;

KEY LEARNINGS

- Each member needs to have clearly defined task to be execued until a certain deadline;

F1 SMART CONTROLLER WEEKLY REPORT – WEEK 47 Date : 22/11/2019

MAIN ACTIVITIES

- PCB soldering;
- PCB testing;
- Documentation redaction;
- Firmware for CAN line sniffer;
- Final design and 3D printing of the enclosure;
- Report related to air flow simulations.

NEXT STEPS

- Evaluation of validation procedures;
- Software using NI Labview;

ISSUES

- Wrong design of the reduced size custom board: conditioning circuit for NRF24L01+ is missing;
- The 32-pin MCU STM8AF62x has not the SPI interface, so it is not possible to substitute STM8AF5288T with it.

MAIN DECISIONS TAKEN

- Leave the soldering pad of the pressure sensor free.

KEY LEARNINGS

- A diode is needed in the battery monitoring circuit in order to avoid current flowing through the MCU pin;
- An LED could have been useful in order to quickly check the correct operation of the board.

F1 SMART CONTROLLER WEEKLY REPORT – WEEK 48 Date : 29/11/2019

MAIN ACTIVITIES

- Completion of the software for data visualization;
- Thermal simulation of the custom board;
- High Side design for future developments;
- Documentation redaction: testing and future developments;
- FW development: custom board NRF24L01+ communication.

NEXT STEPS

- Force simulation on the enclosure;
- Tests to be performed for code validation;
- LabView interface.

ISSUES

- Problems in AIS2120SX acceletometer configuration;
- User interface has still to be completed;

MAIN DECISIONS TAKEN

- Accelerometer AIS2120SX cannot be used.
- USB is not working, USART is used.

KEY LEARNINGS

F1 SMART CONTROLLER WEEKLY REPORT – WEEK 49 Date : 06/12/2019

MAIN ACTIVITIES

- Documentation related to Firmware and user manual;
- Tests to be performed;
- Mechanical design presentation draft;

NEXT STEPS

- Complete documentation;
- Preparation of the PowerPoint presentation.

ISSUES

- Problem with CL1 control data reception;
- USB connection CAN-Labview is still not working;

MAIN DECISIONS TAKEN

- Don't use USB but USART;

KEY LEARNINGS

F1 SMART CONTROLLER WEEKLY REPORT – WEEK 50 Date : 13/12/2019

MAIN ACTIVITIES

- Sensor board soldering checks;
- Documentation completion;
- Static tests on CL1 custom;
- Dynamic test with car on CL1 controller and evaluation sensors board;
- Presentation draft;

NEXT STEPS

- Presentation set-up;
- Speech set-up;

ISSUES

- CL1 sensor board is still not working;
- Difficulties in speech preparation and test with members that are not present;

MAIN DECISIONS TAKEN

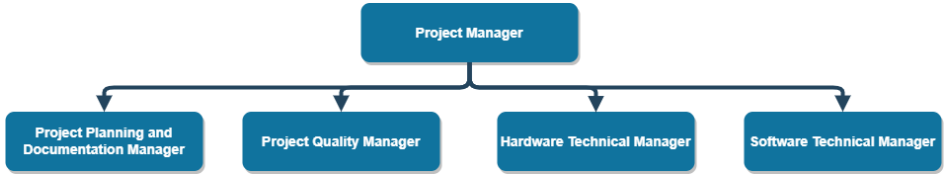
- Presentation structure;
- Speech structure;

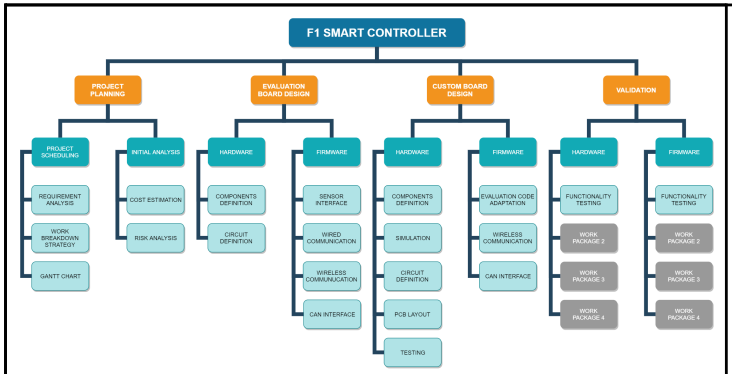
KEY LEARNINGS

Appendix G: Additional Documents

.IV Open Project Macro-activities Report

AE Project - Macro-attività

Tipo	Soggetto	Stato	Responsabile	Assegnatario	Data di fine	Data di inizio
Task	Roles definition	Closed	Giorgio Di Loro	Giorgio Di Loro	10/12/2019	10/10/2019
Definition of individual roles and responsibilities.						
						
Task	Gantt Chart	Closed	Giorgio Di Loro	Giorgio Di Loro	10/14/2019	10/10/2019
Time description of the activities.						
Phase	Project Planning	Closed	Giorgio Di Loro		10/16/2019	10/10/2019
Scheduling of the activities, breakdown of the workload, production of the documentation relative to the planning.						
Task	Requirement Analysis	Closed	Armando Villar		10/16/2019	10/10/2019
Analysis of the requirements of the system: purpose, operating conditions, performance specifications. PERFORMED BY ALL THE MEMBERS OF THE TEAM (except Tong).						
Task	Definition of the Work Breakdown Strategy	Closed	Annachiara Biguzzi	Annachiara Biguzzi	10/16/2019	10/12/2019
Definition of different tasks and deliverables required, division of these in different functional domains.						



Task	Architecture definition	Closed	Giorgio Di Loro		10/20/2019	10/16/2019
Definition of the architecture of the system at different abstraction levels, with top down approach: high level schematic; intermediate level schematic; low level hardware-software architecture. ASSIGNED TO ALL THE MEMBERS OF THE TEAM (according to their roles)						
Phase	Design Definition	Closed	Giorgio Di Loro		10/24/2019	10/16/2019
Final definition of the actual hardware and firmware structure. ASSIGNED TO ALL THE TEAM MEMBERS (according to their roles)						
Phase	Electronic Design	In progress	Giorgio Di Loro		12/04/2019	10/16/2019
Project development general phase: from the evaluation design and realization to the final PCB design and implementation. ASSIGNED TO ALL THE MEMBERS OF THE TEAM (with different roles and objectives).						
Milestone	First meeting w/ Claudio Silenzi	Closed	Giorgio Di Loro		10/17/2019	10/17/2019
Project planning, organization and work partitioning must be concluded and documentation produced. An overall definition of the architecture of the system (high level) must be provided and the different possible solutions for components must be defined (EVALUATION PART SPECIFICALLY). ASSIGNED TO ALL THE MEMBERS OF THE TEAM. Also Tong joined the team on this date.						
Task	Components definition	Closed	Annachiara Biguzzi		10/23/2019	10/17/2019

Selection of the components needed to fulfill the requirements given in the specification. ASSIGNED TO ALL THE MEMBERS OF THE TEAM (according to their roles)						
Task	Design Validation	Closed	Armando Villar		10/23/2019	10/21/2019
Simulation via LTSpice of the components selected and validation of the proposed circuitry.						
Task	Circuit definition	Closed	Annachiara Biguzzi	Giorgio Di Loro	10/28/2019	10/22/2019
Definition of the circuit to be realized in the costum board.						
Phase	Custom Board Design	In progress	Giorgio Di Loro		12/01/2019	10/22/2019
Design and possible realization of the custom board, the final goal of the project.						
Milestone	BOM	Closed	Annachiara Biguzzi		10/24/2019	10/24/2019
Deadline for presenting the list of the components to be purchased by the university (BOM).						
Task	Prototype set up	Closed	Giacomo Ravagli	Luca Mancini	10/29/2019	10/24/2019
Proper connection of sensors and set up of the prototype.						
Phase	Evaluation Prototype Design	Closed	Giorgio Di Loro		11/14/2019	10/24/2019
Full design of the evaluation prototype, composed by Hardware and Firmware design.						
Task	Firmware development	Closed	Giacomo Ravagli	Luca Mancini	11/13/2019	10/27/2019
Development of the firmware (software + interfaces + MCU setup) of the evaluation prototype. Assigned to Giacomo and Luca.						
Task	Simulation	Closed	Armando Villar		11/03/2019	10/29/2019
Simulation of the circuit with LTSpice, in order to evaluate the correct functioning of the designed circuit.						
Task	PCB layout	Closed	Annachiara Biguzzi	Giorgio Di Loro	11/14/2019	11/04/2019
Realization of the layout of the PCB. N.B. Finished 3 days earlier than the expected date.						
Milestone	Second meeting w/ Claudio Silenzi	Closed	Giorgio Di Loro		11/14/2019	11/14/2019

The evaluation part of the project should be concluded and documentation produced. If possible, start to work on validation, verification and custom board HW design and layout. ASSIGNED TO ALL THE MEMBERS OF THE TEAM						
Task	Eval Firmware validation	In progress	Armando Villar		11/21/2019	11/14/2019
Validation of the firmware of the final evaluation board.						
Phase	HW-SW Validation	Scheduled	Armando Villar	Luca Mancini	12/04/2019	11/14/2019
Validation of the last stage of the project.						
Task	Production	Closed	Annachiara Biguzzi	Annachiara Biguzzi	11/21/2019	11/15/2019
Outsourcing of the production of the non pre-soldered PCB. N.B. started 3 days earlier than expected.						
Task	Soldering and Connection Testing	Closed	Annachiara Biguzzi	Giorgio Di Loro	11/24/2019	11/22/2019
Soldering the SMD components to the naked PCB and testing the quality of the connections.						
Task	Firmware development	In progress	Giacomo Ravagli	Luca Mancini	12/01/2019	11/22/2019
Development of the firmware for the final custom board based on the evaluation board's firmware.						
Task	Hardware Validation	In progress	Armando Villar		11/28/2019	11/23/2019
Validation of the hardware of the final custom board.						
Milestone	Third meeting w/ Claudio Silenzi	Closed	Giorgio Di Loro		11/28/2019	11/28/2019
Both the evaluation and the custom board should be working and the main issues should be solved. Minor corrections and refinements left should be discussed with Silenzi.						
Task	Custom Firmware Validation	In progress	Armando Villar		12/04/2019	11/28/2019
Validation of the firmware of the final custom board.						
Task	PowerPoint Presentation redaction	New	Luca Mancini		12/11/2019	12/05/2019
Final PowerPoint to present the project and discuss its features, our choices, our planning and developing activity, etc. ASSIGNED TO ALL THE MEMBERS OF THE TEAM.						
Phase	Project Conclusion	Scheduled	Giorgio Di Loro		12/16/2019	12/05/2019

Last improvements and documents. ASSIGNED TO ALL THE MEMBERS OF THE TEAM						
Task	Final version of the documentation	In progress	Luca Mancini		12/13/2019	12/06/2019
Final redaction of the documentation required (all the deliverables + future projects + others) merged in a single PDF file to be delivered on the 13th of December. ASSIGNED TO ALL THE MEMBERS OF THE TEAM						
Task	Speech Preparation	New	Giorgio Di Loro		12/16/2019	12/12/2019
Preparation of the speech for the conclusion of the project. ASSIGNED TO ALL THE MEMBERS OF THE TEAM						
Milestone	Presentation of the project	Scheduled	Giorgio Di Loro		12/17/2019	12/17/2019
Conclusion of the project.						

.V Software Documentation

CL1 Software Documentation

Final Revision

Generated by Doxygen 1.8.16

1 Class Index **1**

1.1 Class List 1

2 File Index **3**

2.1 File List 3

3 Class Documentation **5**

3.1 RadioPacket Struct Reference 5

4 File Documentation **7**

4.1 C:/Users/Luca/Desktop/DOXYGEN files/CL1_unified_multimode_reducedSize.c File Reference . . . 7

4.1.1 Detailed Description 8

4.1.2 Function Documentation 9

4.1.2.1 ErrorSignal_BMP_CAN() 9

4.1.2.2 ErrorSignal_LIS_CAN() 9

4.1.2.3 ErrorSignal_nRF_CAN() 9

4.1.2.4 loop() 10

4.1.2.5 send_data_to_CAN_bus() 10

4.1.2.6 setup() 11

4.1.2.7 setup_CAN() 12

4.2 C:/Users/Luca/Desktop/DOXYGEN files/evaluation_unified_multimode.c File Reference 12

4.2.1 Detailed Description 14

4.2.2 Function Documentation 14

4.2.2.1 ErrorSignal_BMP_CAN() 14

4.2.2.2 ErrorSignal_LIS_CAN() 15

4.2.2.3 ErrorSignal_nRF_CAN() 15

4.2.2.4 loop() 15

4.2.2.5 send_data_to_CAN_bus() 16

4.2.2.6 setup() 17

4.2.2.7 setup_CAN() 18

4.2.2.8 SOS_blinking_LED() 18

4.3 C:/Users/Luca/Desktop/DOXYGEN files/main.c File Reference 19

4.3.1 Detailed Description 19

4.3.2 Function Documentation 20

4.3.2.1 CAN_Config() 20

4.3.2.2 debugPrint() 20

4.3.2.3 Error_Handler() 20

4.3.2.4 main() 21

4.3.2.5 SystemClock_Config() 21

Index **23**

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

RadioPacket 5

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

C:/Users/Luca/Desktop/DOXYGEN files/ **CL1_unified_multimode_reducedSize.c**
This file is aimed at providing an handy tool to the user of the CL1 board. It allows an easy reconfiguration of the functionality of the board according to the use case, without wasting resources: only the relevant portion of code will be compiled, exploiting preprocessor's directives. Three OPERATING MODES are allowed: 1) Slave mode (Smart Sensor): the CL1 board acquires the pressure and the acceleration, and send it to the master via radio packet 2) Master mode: the CL1 board has no sensor; acquires radio packet sent by a slave node, and transmit the packet on the CAN bus 3) Wired Sensor: the CL1 board acquires pressure and acceleration from sensor installed on it, and transmit the data directly on the CAN bus (no wireless communication in this case) 7

C:/Users/Luca/Desktop/DOXYGEN files/ **evaluation_unified_multimode.c**
This file is aimed at providing the main application for the evaluation setup of the CL1 board project. It allows an easy reconfiguration of the functionality of the setup according to the use case, without wasting resources: only the relevant portion of code will be compiled, exploiting preprocessor's directives. Three OPERATING MODES are allowed: 1) Slave mode (Smart Sensor): the CL1 board acquires the pressure and the acceleration, and send it to the master via radio packet 2) Master mode: the CL1 board has no sensor; acquires radio packet sent by a slave node, and transmit the packet on the CAN bus 3) Wired Sensor: the CL1 board acquires pressure and acceleration from sensor installed on it, and transmit the data directly on the CAN bus (no wireless communication in this case) 12

C:/Users/Luca/Desktop/DOXYGEN files/ **main.c**
This file contains the main program used to read data from the CAN bus using STM32F4DISC1. The program also sends the data read to the serial monitor using USART interface 19

Chapter 3

Class Documentation

3.1 RadioPacket Struct Reference

Public Attributes

- uint8_t **FromRadioId**
- int32_t **PressureTx**
- int32_t **AccelerationTx**
- uint8_t **FailedTxCount**

The documentation for this struct was generated from the following files:

- C:/Users/Luca/Desktop/DOXYGEN files/ **CL1_unified_multimode_reducedSize.c**
- C:/Users/Luca/Desktop/DOXYGEN files/ **evaluation_unified_multimode.c**

Chapter 4

File Documentation

4.1 C:/Users/Luca/Desktop/DOXYGEN files/CL1_unified_multimode_reducedSize.c File Reference

This file is aimed at providing an handy tool to the user of the CL1 board. It allows an easy reconfiguration of the functionality of the board according to the use case, without wasting resources: only the relevant portion of code will be compiled, exploiting preprocessor's directives. Three OPERATING MODES are allowed: 1) Slave mode (Smart Sensor): the CL1 board acquires the pressure and the acceleration, and send it to the master via radio packet 2) Master mode: the CL1 board has no sensor; acquires radio packet sent by a slave node, and transmit the packet on the CAN bus 3) Wired Sensor: the CL1 board acquires pressure and acceleration from sensor installed on it, and transmit the data directly on the CAN bus (no wireless communication in this case).

```
#include "Wire.h"
#include <SPI.h>
#include <stm8s_can.h>
#include <NRFLite.h>
#include <Adafruit_BMP085.h>
#include "lis3dh-motion-detection.h"
```

Classes

- struct **RadioPacket**

Macros

- #define **SLAVE_MODE**
- #define **MASTER_MODE**
- #define **WIRED_SENSOR**
- #define **HIGH_RESOLUTION**
- #define **LIS3DH_DEBUG** Serial

Functions

- LIS3DH **myIMU** (0x18)
- void **setup** ()

  SLAVE MODE initialization function.
- void **setup_CAN** ()

  CAN peripheral initialization function.
- void **ErrorSignal_nRF_CAN** ()

  NRF24L01 initialization error function.
- void **ErrorSignal_BMP_CAN** ()

  BMP initialization error function.
- void **ErrorSignal_LIS_CAN** ()

  LIS3DH initialization error function.
- void **loop** ()

  SLAVE MODE application entry point.
- void **send_data_to_CAN_bus** ()

  CAN sending data function.

Variables

- unsigned long **ID** = 0
- unsigned char **tx_buffer** [0x08] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}
- NRFLite **_radio**
- **RadioPacket _radioData**
- uint16_t **sampleRate** = 100
- uint8_t **accelRange** = 16
- int16_t **dataHighres** = 0
- Adafruit_BMP085 **bmp**
- float **AccAsseZ** = 0
- int32_t **AccAsseZ_mg** = 0
- float **Temp_Meas** = 0
- int32_t **Press_Meas** = 0

4.1.1 Detailed Description

This file is aimed at providing an handy tool to the user of the CL1 board. It allows an easy reconfiguration of the functionality of the board according to the use case, without wasting resources: only the relevant portion of code will be compiled, exploiting preprocessor's directives. Three OPERATING MODES are allowed: 1) Slave mode (Smart Sensor): the CL1 board acquires the pressure and the acceleration, and send it to the master via radio packet 2) Master mode: the CL1 board has no sensor; acquires radio packet sent by a slave node, and transmit the packet on the CAN bus 3) Wired Sensor: the CL1 board acquires pressure and acceleration from sensor installed on it, and transmit the data directly on the CAN bus (no wireless communication in this case).

Author

CL1 Team

Version

V3.0

Date

09-December-2019

4.1.2 Function Documentation

4.1.2.1 ErrorSignal_BMP_CAN()

```
void ErrorSignal_BMP_CAN ( )
```

BMP initialization error function.

Parameters

None	

Return values

None	

Send an error signal on the CAN bus if BMP is not correctly initialized.

4.1.2.2 ErrorSignal_LIS_CAN()

```
void ErrorSignal_LIS_CAN ( )
```

LIS3DH initialization error function.

Parameters

None	

Return values

None	

Send an error signal on the CAN bus if LIS (accelerometer) is not correctly initialized.

4.1.2.3 ErrorSignal_nRF_CAN()

```
void ErrorSignal_nRF_CAN ( )
```

NRF24L01 initialization error function.

Parameters

None	

Return values

None	

Send an error signal on the CAN bus if radio is not correctly initialized.

4.1.2.4 loop()

```
void loop ( )
```

SLAVE MODE application entry point.

WIRED SENSOR application entry point.

MASTER MODE application entry point.

Return values

None	

Note

Execution of the main program

Perform the acquisition from the accelerometer.

Perform the acquisition from the barometer.

Send the data.

Create the bit mask for the PRESSURE data to be transmitted on the CAN bus int32_t --> 4 x uint8_t (then, at the receiver size we need to do the same).

Create the bit mask for the ACCELEROMETER data to be transmitted on the CAN bus int32_t --> 4 x uint8_t (then, at the receiver size we need to do the same).

Perform the acquisition from the accelerometer.

Perform the acquisition from the barometer.

Create the bit mask for the PRESSURE data to be transmitted on the CAN bus int32_t --> 4 x uint8_t (then, at the receiver size we need to do the same).

Create the bit mask for the ACCELEROMETER data to be transmitted on the CAN bus int32_t --> 4 x uint8_t (then, at the receiver size we need to do the same).

4.1.2.5 send_data_to_CAN_bus()

```
void send_data_to_CAN_bus ( )
```

CAN sending data function.

Parameters

None	

Return values

None	

Note

This function sends the specified payload on the CAN bus.

Transmits the data through the CAN bus.

4.1.2.6 setup()

```
void setup ( )
```

SLAVE MODE initialization function.

WIRED SENSOR initialization function.

MASTER MODE initialization function.

Parameters

None	

Return values

None	

Note

Initialization of the SLAVE node.

Parameters

None	

Return values

None	

Note

Initialization of the MASTER node.

Parameters

None	

Return values

None	

Note

Initialization of the WIRED SENSOR.

Initialization of the accelerometer.

Initialization of the barometer.

Initialization of the nRF24.

Initialization of the accelerometer.

Initialization of the barometer.

4.1.2.7 setup_CAN()

```
void setup_CAN ( )
```

CAN peripheral initialization function.

Parameters

None	

Return values

None	

CAN peripheral deinitialization.

CAN peripheral initialization.

CAN input filter configuration.

4.2 C:/Users/Luca/Desktop/DOXYGEN files/evaluation_unified_multimode.c File Reference

This file is aimed at providing the main application for the evaluation setup of the CL1 board project. It allows an easy reconfiguration of the functionality of the setup according to the use case, without wasting resources: only

the relevant portion of code will be compiled, exploiting preprocessor's directives. Three OPERATING MODES are allowed: 1) Slave mode (Smart Sensor): the CL1 board acquires the pressure and the acceleration, and send it to the master via radio packet 2) Master mode: the CL1 board has no sensor; acquires radio packet sent by a slave node, and transmit the packet on the CAN bus 3) Wired Sensor: the CL1 board acquires pressure and acceleration from sensor installed on it, and transmit the data directly on the CAN bus (no wireless communication in this case).

```
#include "Wire.h"
#include <SPI.h>
#include <stm8s_can.h>
#include <NRFLite.h>
#include <Adafruit_BMP085.h>
#include "lis3dh-motion-detection.h"
```

Classes

- struct **RadioPacket**

Macros

- #define **SLAVE_MODE**
- #define **MASTER_MODE**
- #define **WIRED_SENSOR**
- #define **HIGH_RESOLUTION**
- #define **LIS3DH_DEBUG** Serial

Functions

- LIS3DH **myIMU** (0x18)
- void **setup** ()

  SLAVE MODE initialization function.
- void **setup_CAN** ()

  CAN peripheral initialization function.
- void **ErrorSignal_nRF_CAN** ()

  NRF24L01 initialization error function.
- void **ErrorSignal_BMP_CAN** ()

  BMP initialization error function.
- void **ErrorSignal_LIS_CAN** ()

  LIS3DH initialization error function.
- void **loop** ()

  SLAVE MODE application entry point.
- void **send_data_to_CAN_bus** ()

  CAN sending data function.
- void **SOS_blinking_LED** ()

  SOS led blinking function.

Variables

- unsigned long **ID** = 0
- unsigned char **tx_buffer** [0x08] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}
- NRFLite **_radio**
- **RadioPacket _radioData**
- uint16_t **sampleRate** = 100
- uint8_t **accelRange** = 16
- int16_t **dataHighres** = 0
- Adafruit_BMP085 **bmp**
- float **AccAsseZ** = 0
- int32_t **AccAsseZ_mg** = 0
- float **Temp_Meas** = 0
- int32_t **Press_Meas** = 0

4.2.1 Detailed Description

This file is aimed at providing the main application for the evaluation setup of the CL1 board project. It allows an easy reconfiguration of the functionality of the setup according to the use case, without wasting resources: only the relevant portion of code will be compiled, exploiting preprocessor's directives. Three OPERATING MODES are allowed: 1) Slave mode (Smart Sensor): the CL1 board acquires the pressure and the acceleration, and send it to the master via radio packet 2) Master mode: the CL1 board has no sensor; acquires radio packet sent by a slave node, and transmit the packet on the CAN bus 3) Wired Sensor: the CL1 board acquires pressure and acceleration from sensor installed on it, and transmit the data directly on the CAN bus (no wireless communication in this case).

Author

CL1 Team

Version

V3.0

Date

09-December-2019

4.2.2 Function Documentation

4.2.2.1 ErrorSignal_BMP_CAN()

```
void ErrorSignal_BMP_CAN ( )
```

BMP initialization error function.

Parameters

None	

Return values

None	

Send an error signal on the CAN bus if BMP is not correctly initialized.

4.2.2.2 ErrorSignal_LIS_CAN()

```
void ErrorSignal_LIS_CAN ( )
```

LIS3DH initialization error function.

Parameters

None	

Return values

None	

Send an error signal on the CAN bus if LIS (accelerometer) is not correctly initialized.

4.2.2.3 ErrorSignal_nRF_CAN()

```
void ErrorSignal_nRF_CAN ( )
```

NRF24L01 initialization error function.

Parameters

None	

Return values

None	

Send an error signal on the CAN bus if radio is not correctly initialized.

4.2.2.4 loop()

```
void loop ( )
```

SLAVE MODE application entry point.

WIRED SENSOR application entry point.

MASTER MODE application entry point.

Return values

None	

Note

Execution of the main program

Perform the acquisition from the accelerometer.

Perform the acquisition from the barometer.

Send the data.

Create the bit mask for the PRESSURE data to be transmitted on the CAN bus int32_t --> 4 x uint8_t (then, at the receiver size we need to do the same).

Create the bit mask for the ACCELEROMETER data to be transmitted on the CAN bus int32_t --> 4 x uint8_t (then, at the receiver size we need to do the same).

Perform the acquisition from the accelerometer.

Perform the acquisition from the barometer.

Create the bit mask for the PRESSURE data to be transmitted on the CAN bus int32_t --> 4 x uint8_t (then, at the receiver size we need to do the same).

Create the bit mask for the ACCELEROMETER data to be transmitted on the CAN bus int32_t --> 4 x uint8_t (then, at the receiver size we need to do the same).

4.2.2.5 send_data_to_CAN_bus()

```
void send_data_to_CAN_bus ( )
```

CAN sending data function.

Parameters

None	

Return values

None	

Note

This function sends the specified payload on the CAN bus.

Transmits the data through the CAN bus.

4.2.2.6 setup()

```
void setup ( )
```

SLAVE MODE initialization function.

WIRED SENSOR initialization function.

MASTER MODE initialization function.

Parameters

None	

Return values

None	

Note

Initialization of the SLAVE node.

Parameters

None	

Return values

None	

Note

Initialization of the MASTER node.

Parameters

None	

Return values

None	

Note

Initialization of the WIRED SENSOR.

Initialization of the accelerometer.

Initialization of the barometer.

Initialization of the nRF24.

Initialization of the accelerometer.

Initialization of the barometer.

4.2.2.7 setup_CAN()

```
void setup_CAN ( )
```

CAN peripheral initialization function.

Parameters

None	

Return values

None	

CAN peripheral deinitialization.

CAN peripheral initialization.

CAN input filter configuration.

4.2.2.8 SOS_blinking_LED()

```
void SOS_blinking_LED ( )
```

SOS led blinking function.

Parameters

None	

Return values

None	

Note

This function outputs an SOS message using leds, in case of initialization errors.

Initialize all the LEDs; will be used to encode error messages.

Encoding that there is a problem: 3 LEDs blink an SOS.

4.3 C:/Users/Luca/Desktop/DOXYGEN files/main.c File Reference

This file contains the main program used to read data from the CAN bus using STM32F4DISC1. The program also sends the data read to the serial monitor using USART interface.

```
#include "main.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
```

Functions

- void **SystemClock_Config** (void)

  System Clock Configuration.
- void **CAN_Config** (void)

  CAN peripheral setup.
- void **debugPrint** (UART_HandleTypeDef ∗huart, char out[ ])

  Serial monitor print redirect from USART function.
- int **main** (void)

  The application entry point.
- void **Error_Handler** (void)

  This function is executed in case of error occurrence.

Variables

- CAN_HandleTypeDef **hcan1**
- UART_HandleTypeDef **huart2**
- CAN_FilterTypeDef **CAN_FilterConfStructure**
- CAN_TxHeaderTypeDef **CAN_TxMessage**
- CAN_RxHeaderTypeDef **CAN_RxMessage**
- char **accString** [40]
- char **pressureString** [40]
- uint8_t **messageReceived** = 0
- uint8_t **receiveBuffer** [8] = {0, 0, 0, 0, 0, 0, 0, 0}
- int32_t **accelerometerData** = 0
- int32_t **pressureData** = 0
- int32_t **pressureDataTemp** = 0

4.3.1 Detailed Description

This file contains the main program used to read data from the CAN bus using STM32F4DISC1. The program also sends the data read to the serial monitor using USART interface.

Author

CL1 Team

Version

V3.0

Date

09-December-2019

4.3.2 Function Documentation

4.3.2.1 CAN_Config()

```
void CAN_Config (
             void  )
```

CAN peripheral setup.

Parameters

None	

Return values

None	

Filter configuration - accept all data.

4.3.2.2 debugPrint()

```
void debugPrint (
             UART_HandleTypeDef * huart,
             char out[] )
```

Serial monitor print redirect from USART function.

Parameters

None	

Return values

None	

Redirects the USART transmit function to char transmission.

4.3.2.3 Error_Handler()

```
void Error_Handler (
             void  )
```

This function is executed in case of error occurrence.

Return values

None	

4.3.2.4 main()

```
int main (
             void  )
```

The application entry point.

Return values

int	

Reset of all peripherals, Initializes the Flash interface and the Systick.

Configures the system clock.

Initializes all configured peripherals.

In the infinite loop the program just waits for an interrupt to rise, receives the message and outputs the data via USART.

Receiving the message from the input FIFO.

Decoding the data and copying it into the corresponding variables.

4.3.2.5 SystemClock_Config()

```
void SystemClock_Config (
             void  )
```

System Clock Configuration.

Return values

None	

Configure the main internal regulator output voltage

Initializes the CPU, AHB and APB busses clocks

Index

C:/Users/Luca/Desktop/DOXYGEN files/CL1_unified_multimode_reducedSize.c, 7
C:/Users/Luca/Desktop/DOXYGEN files/evaluation_unified_multimode.c, 12
C:/Users/Luca/Desktop/DOXYGEN files/main.c, 19
CAN_Config
    main.c, 20
CL1_unified_multimode_reducedSize.c
    ErrorSignal_BMP_CAN, 9
    ErrorSignal_LIS_CAN, 9
    ErrorSignal_nRF_CAN, 9
    loop, 10
    send_data_to_CAN_bus, 10
    setup, 11
    setup_CAN, 12

debugPrint
    main.c, 20

Error_Handler
    main.c, 20
ErrorSignal_BMP_CAN
    CL1_unified_multimode_reducedSize.c, 9
    evaluation_unified_multimode.c, 14
ErrorSignal_LIS_CAN
    CL1_unified_multimode_reducedSize.c, 9
    evaluation_unified_multimode.c, 15
ErrorSignal_nRF_CAN
    CL1_unified_multimode_reducedSize.c, 9
    evaluation_unified_multimode.c, 15
evaluation_unified_multimode.c
    ErrorSignal_BMP_CAN, 14
    ErrorSignal_LIS_CAN, 15
    ErrorSignal_nRF_CAN, 15
    loop, 15
    send_data_to_CAN_bus, 16
    setup, 16
    setup_CAN, 18
    SOS_blinking_LED, 18

loop
    CL1_unified_multimode_reducedSize.c, 10
    evaluation_unified_multimode.c, 15

main
    main.c, 21
main.c
    CAN_Config, 20
    debugPrint, 20
    Error_Handler, 20
    main, 21
    SystemClock_Config, 21
RadioPacket, 5

send_data_to_CAN_bus
    CL1_unified_multimode_reducedSize.c, 10
    evaluation_unified_multimode.c, 16
setup
    CL1_unified_multimode_reducedSize.c, 11
    evaluation_unified_multimode.c, 16
setup_CAN
    CL1_unified_multimode_reducedSize.c, 12
    evaluation_unified_multimode.c, 18
SOS_blinking_LED
    evaluation_unified_multimode.c, 18
SystemClock_Config
    main.c, 21

.VI MCU - STM8AF5288T STM8 CubeMX Configuration

1. Description

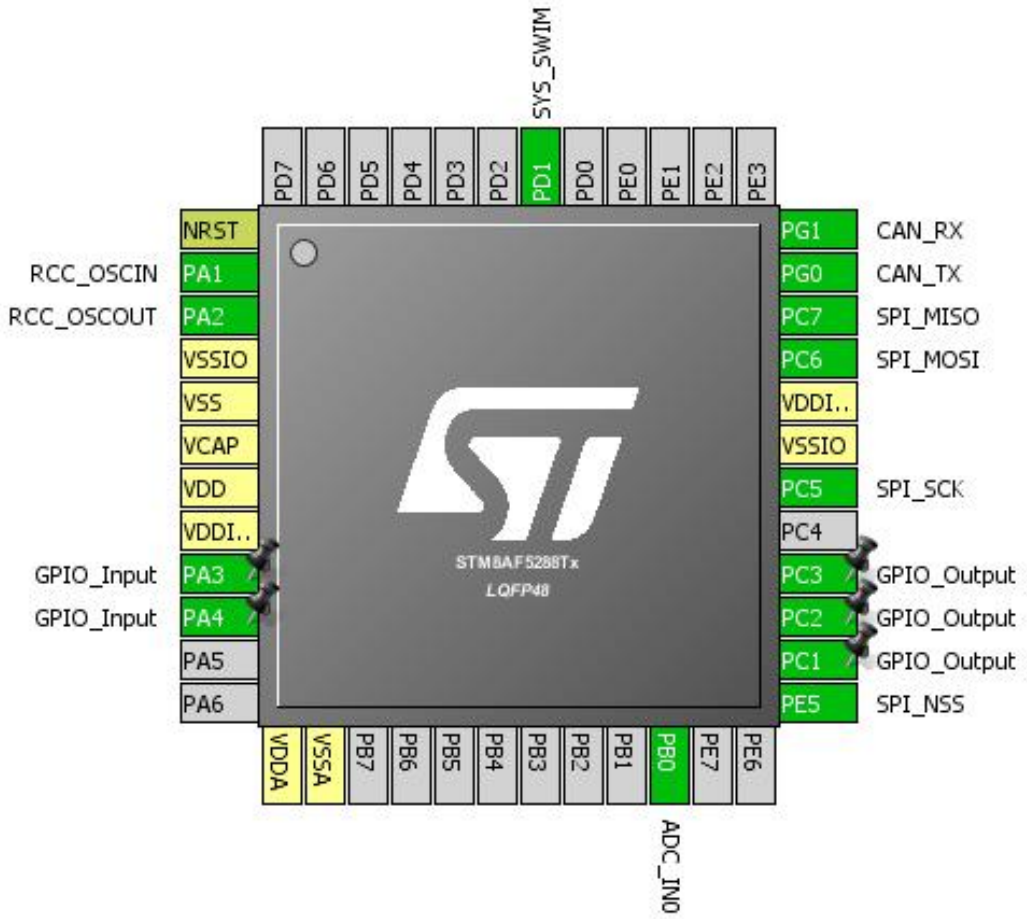
1.1. Project

Project Name	CL1
Board Name	No information
Generated with:	STM8CubeMX 1.4.0
Date	11/20/2019

1.2. MCU

MCU Series	STM8AF
MCU Line	STM8AF52
MCU name	STM8AF5288Tx
MCU Package	LQFP48
MCU Pin number	48

2. Pinout Configuration

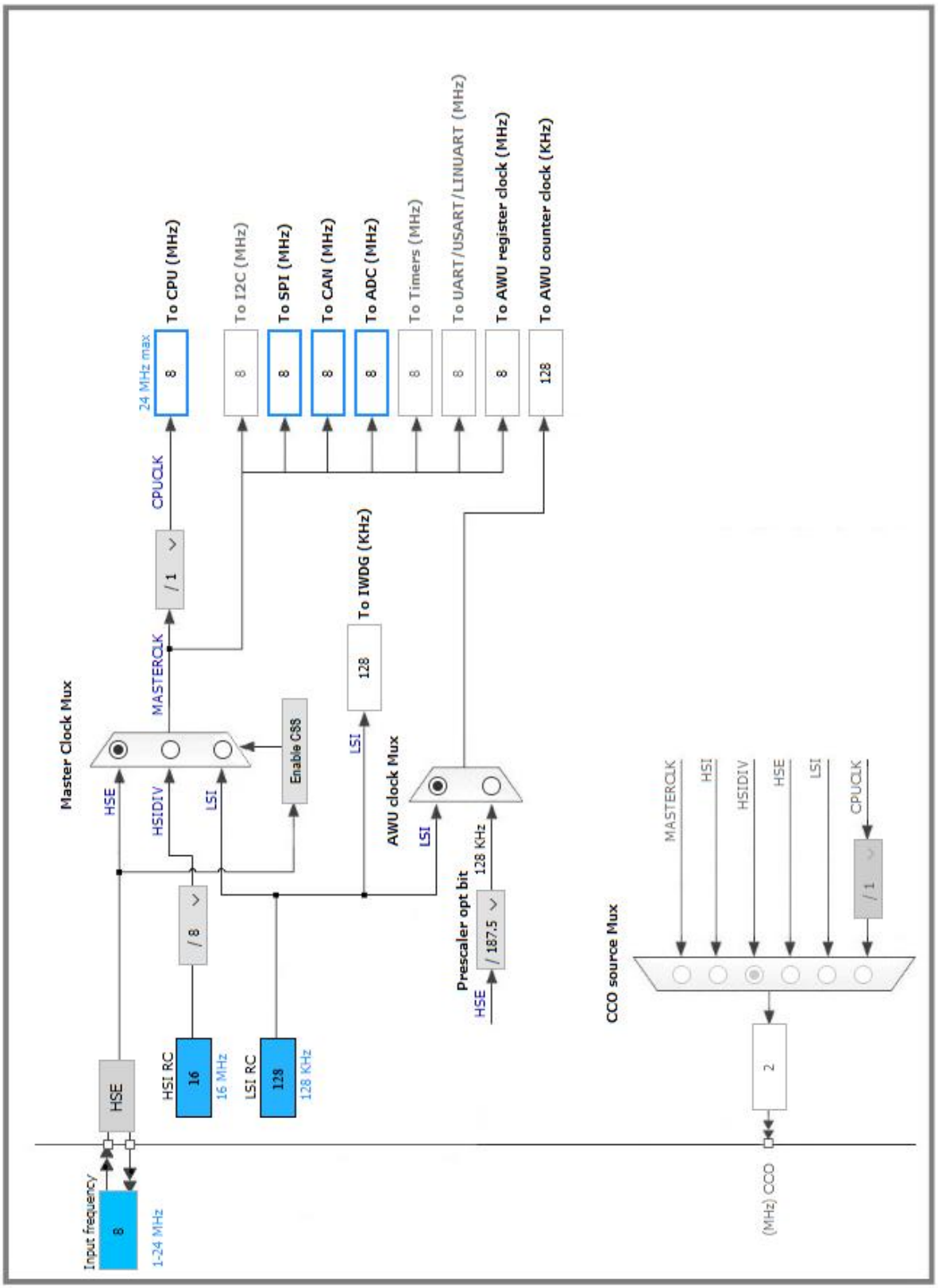


3. Pins Configuration

Pin Number LQFP48	Pin Name (function after reset)	Pin Type	Alternate Function(s)	Label
1	NRST	Reset		
2	PA1	I/O	RCC_OSCIN	
3	PA2	I/O	RCC_OSCOUT	
4	VSSIO	Power		
5	VSS	Power		
6	VCAP	Power		
7	VDD	Power		
8	VDDIO	Power		
9	PA3 *	I/O	GPIO_Input	
10	PA4 *	I/O	GPIO_Input	
13	VDDA	Power		
14	VSSA	Power		
22	PB0	I/O	ADC_IN0	
25	PE5	I/O	SPI_NSS	
26	PC1 *	I/O	GPIO_Output	
27	PC2 *	I/O	GPIO_Output	
28	PC3 *	I/O	GPIO_Output	
30	PC5	I/O	SPI_SCK	
31	VSSIO	Power		
32	VDDIO	Power		
33	PC6	I/O	SPI_MOSI	
34	PC7	I/O	SPI_MISO	
35	PG0	I/O	CAN_TX	
36	PG1	I/O	CAN_RX	
42	PD1	I/O	SYS_SWIM	

* The pin is affected with an I/O function

4. Clock Tree Configuration



5. Power Consumption Calculator report

5.1. Microcontroller Selection

Series	STM8AF
Line	STM8AF52
MCU	STM8AF5288Tx
Datasheet	14395_Rev15

5.2. Parameter Selection

Temperature	25
Vdd	5.0

.VII MCU - STM32F407G_DISC1 STM32 CubeMX Configuration

1. Description

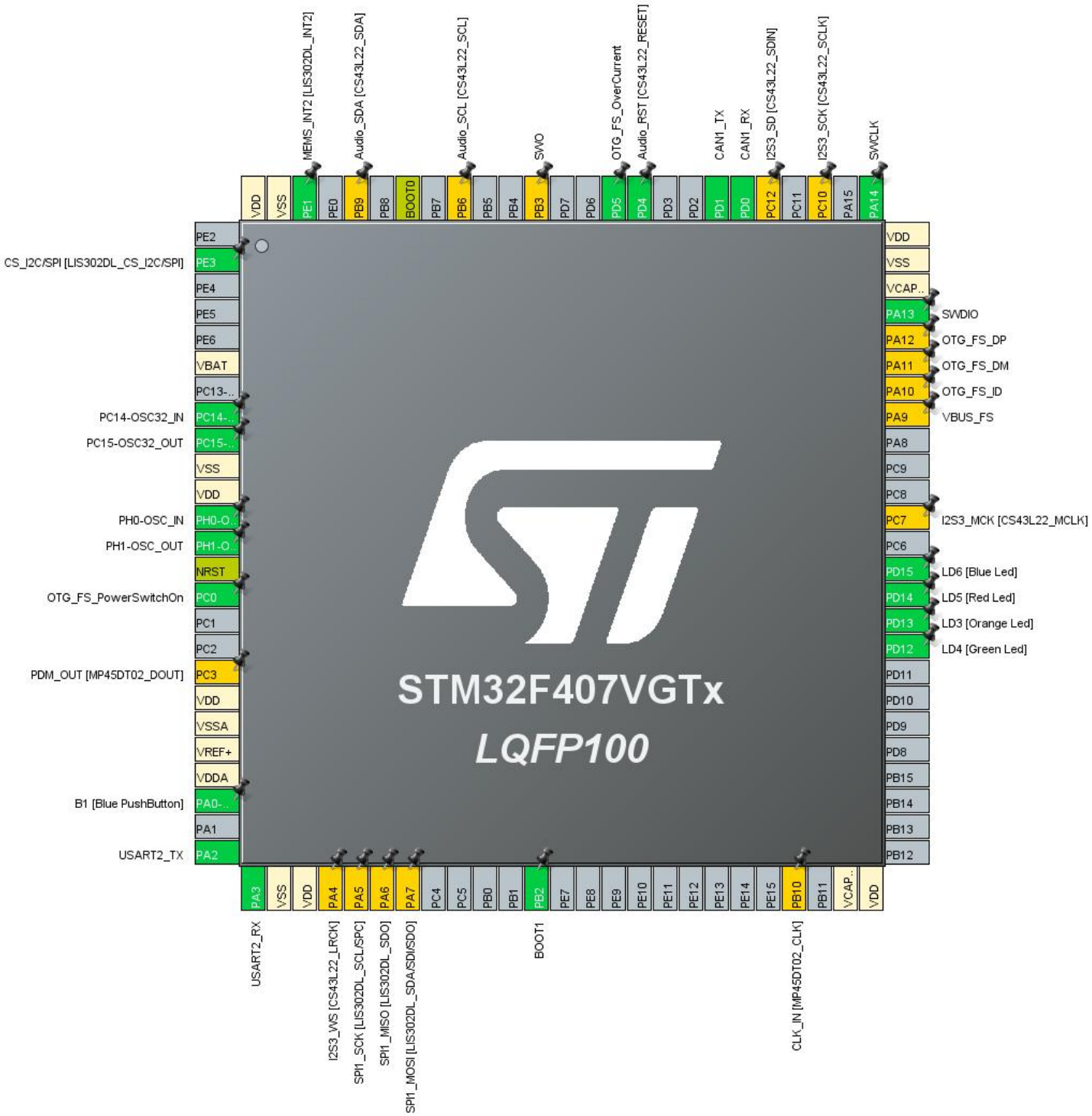
1.1. Project

Project Name	CAN_USART
Board Name	STM32F407G-DISC1
Generated with:	STM32CubeMX 5.4.0
Date	12/09/2019

1.2. MCU

MCU Series	STM32F4
MCU Line	STM32F407/417
MCU name	STM32F407VGTx
MCU Package	LQFP100
MCU Pin number	100

2. Pinout Configuration



3. Pins Configuration

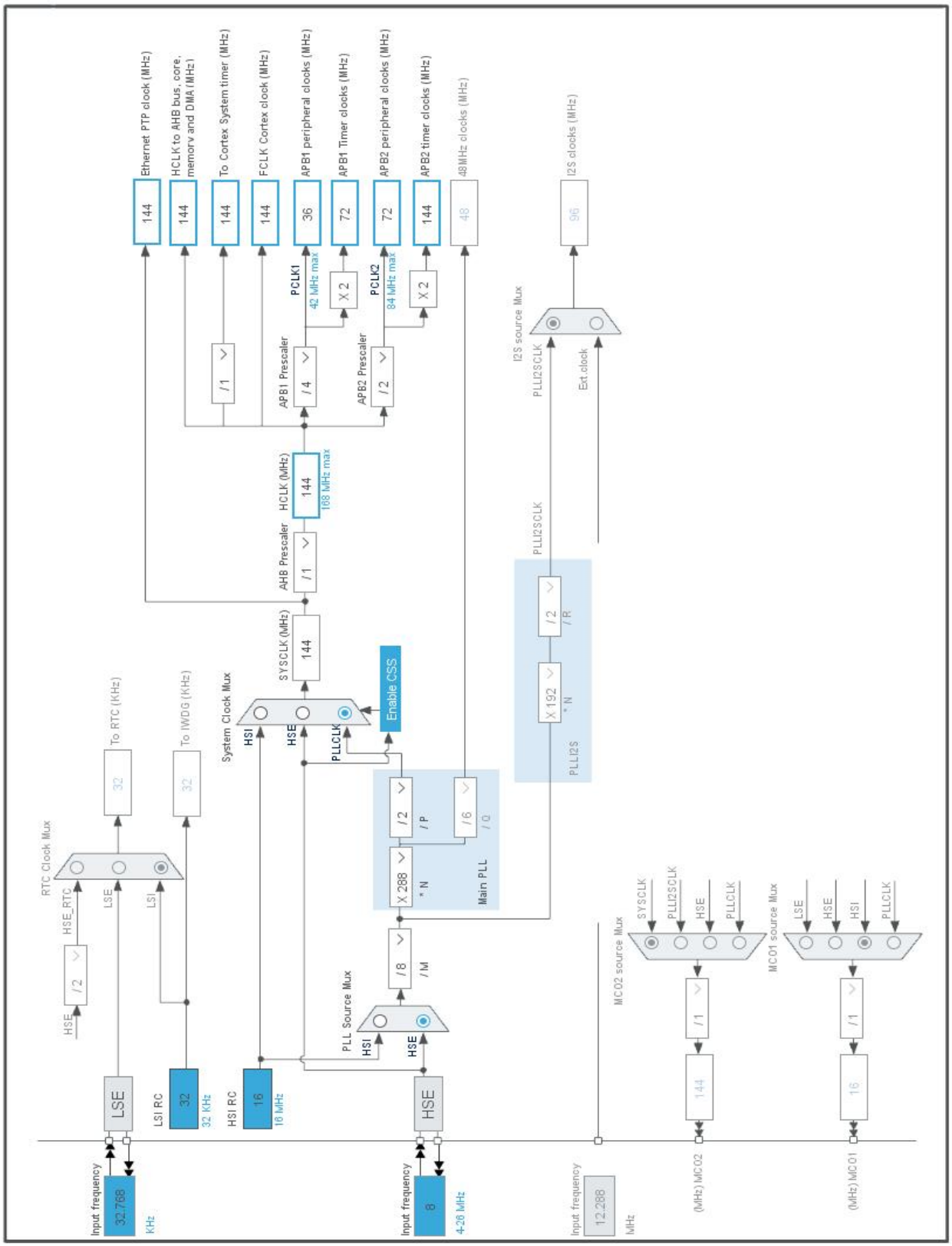
Pin Number LQFP100	Pin Name (function after reset)	Pin Type	Alternate Function(s)	Label
2	PE3 *	I/O	GPIO_Output	CS_I2C/SPI [LIS302DL_CS_I2C/SPI]
6	VBAT	Power		
8	PC14-OSC32_IN	I/O	RCC_OSC32_IN	PC14-OSC32_IN
9	PC15-OSC32_OUT	I/O	RCC_OSC32_OUT	PC15-OSC32_OUT
10	VSS	Power		
11	VDD	Power		
12	PH0-OSC_IN	I/O	RCC_OSC_IN	PH0-OSC_IN
13	PH1-OSC_OUT	I/O	RCC_OSC_OUT	PH1-OSC_OUT
14	NRST	Reset		
15	PC0 *	I/O	GPIO_Output	OTG_FS_PowerSwitchOn
18	PC3 **	I/O	I2S2_SD	PDM_OUT [MP45DT02_DOUT]
19	VDD	Power		
20	VSSA	Power		
21	VREF+	Power		
22	VDDA	Power		
23	PA0-WKUP	I/O	GPIO_EXTI0	B1 [Blue PushButton]
25	PA2	I/O	USART2_TX	
26	PA3	I/O	USART2_RX	
27	VSS	Power		
28	VDD	Power		
29	PA4 **	I/O	I2S3_WS	I2S3_WS [CS43L22_LRCK]
30	PA5 **	I/O	SPI1_SCK	SPI1_SCK [LIS302DL_SCL/SPC]
31	PA6 **	I/O	SPI1_MISO	SPI1_MISO [LIS302DL_SDO]
32	PA7 **	I/O	SPI1_MOSI	SPI1_MOSI [LIS302DL_SDA/SDI/SDO]
37	PB2 *	I/O	GPIO_Input	BOOT1
47	PB10 **	I/O	I2S2_CK	CLK_IN [MP45DT02_CLK]
49	VCAP_1	Power		
50	VDD	Power		
59	PD12 *	I/O	GPIO_Output	LD4 [Green Led]
60	PD13 *	I/O	GPIO_Output	LD3 [Orange Led]
61	PD14 *	I/O	GPIO_Output	LD5 [Red Led]
62	PD15 *	I/O	GPIO_Output	LD6 [Blue Led]

Pin Number LQFP100	Pin Name (function after reset)	Pin Type	Alternate Function(s)	Label
64	PC7 **	I/O	I2S3_MCK	I2S3_MCK [CS43L22_MCLK]
68	PA9 **	I/O	USB_OTG_FS_VBUS	VBUS_FS
69	PA10 **	I/O	USB_OTG_FS_ID	OTG_FS_ID
70	PA11 **	I/O	USB_OTG_FS_DM	OTG_FS_DM
71	PA12 **	I/O	USB_OTG_FS_DP	OTG_FS_DP
72	PA13	I/O	SYS_JTMS-SWDIO	SWDIO
73	VCAP_2	Power		
74	VSS	Power		
75	VDD	Power		
76	PA14	I/O	SYS_JTCK-SWCLK	SWCLK
78	PC10 **	I/O	I2S3_CK	I2S3_SCK [CS43L22_SCLK]
80	PC12 **	I/O	I2S3_SD	I2S3_SD [CS43L22_SDIN]
81	PD0	I/O	CAN1_RX	
82	PD1	I/O	CAN1_TX	
85	PD4 *	I/O	GPIO_Output	Audio_RST [CS43L22_RESET]
86	PD5 *	I/O	GPIO_Input	OTG_FS_OverCurrent
89	PB3 **	I/O	SYS_JTDO-SWO	SWO
92	PB6 **	I/O	I2C1_SCL	Audio_SCL [CS43L22_SCL]
94	BOOT0	Boot		
96	PB9 **	I/O	I2C1_SDA	Audio_SDA [CS43L22_SDA]
98	PE1	I/O	GPIO_EXTI1	MEMS_INT2 [LIS302DL_INT2]
99	VSS	Power		
100	VDD	Power		

* The pin is affected with an I/O function

** The pin is affected with a peripheral function but no peripheral mode is activated

4. Clock Tree Configuration



5. Software Project

5.1. Project Settings

Name	Value
Project Name	CAN_USART
Project Folder	C:\Users\Luca\Desktop\a\CAN_USART
Toolchain / IDE	SW4STM32
Firmware Package Name and Version	STM32Cube FW_F4 V1.24.1

5.2. Code Generation Settings

Name	Value
STM32Cube MCU packages and embedded software	Copy only the necessary library files
Generate peripheral initialization as a pair of '.c/.h' files	No
Backup previously generated files when re-generating	No
Delete previously generated files when not re-generated	Yes
Set all free pins as analog (to optimize the power consumption)	No

6. Power Consumption Calculator report

6.1. Microcontroller Selection

Series	STM32F4
Line	STM32F407/417
MCU	STM32F407VGTx
Datasheet	022152_Rev8

6.2. Parameter Selection

Temperature	25
Vdd	3.3

7. IPs and Middleware Configuration

7.1. CAN1

mode: Mode

7.1.1. Parameter Settings:

Bit Timings Parameters:

Prescaler (for Time Quantum)	**9 ***
Time Quantum	**250.0 ***
Time Quanta in Bit Segment 1	**8 Times ***
Time Quanta in Bit Segment 2	**7 Times ***
ReSynchronization Jump Width	1 Time

Basic Parameters:

Time Triggered Communication Mode	Disable
Automatic Bus-Off Management	Disable
Automatic Wake-Up Mode	Disable
Automatic Retransmission	Disable
Receive Fifo Locked Mode	Disable
Transmit Fifo Priority	Disable

Advanced Parameters:

Operating Mode	Normal

7.2. GPIO

7.3. RCC

High Speed Clock (HSE): Crystal/Ceramic Resonator

Low Speed Clock (LSE) : Crystal/Ceramic Resonator

7.3.1. Parameter Settings:

System Parameters:

VDD voltage (V)	3.3
Instruction Cache	Enabled
Prefetch Buffer	Enabled
Data Cache	Enabled
Flash Latency(WS)	4 WS (5 CPU cycle)

RCC Parameters:

HSI Calibration Value	16

HSE Startup Timout Value (ms)	100
LSE Startup Timout Value (ms)	5000

Power Parameters:

Power Regulator Voltage Scale	Power Regulator Voltage Scale 1

7.4. SYS

Debug: Serial Wire

Timebase Source: SysTick

7.5. USART2

Mode: Asynchronous

7.5.1. Parameter Settings:

Basic Parameters:

Baud Rate	115200
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1

Advanced Parameters:

Data Direction	Receive and Transmit
Over Sampling	16 Samples

*** User modified value**

8. System Configuration

8.1. GPIO configuration

IP	Pin	Signal	GPIO mode	GPIO pull/up pull down	Max Speed	User Label
CAN1	PD0	CAN1_RX	Alternate Function Push Pull	No pull-up and no pull-down	**Very High ***	
	PD1	CAN1_TX	Alternate Function Push Pull	No pull-up and no pull-down	**Very High ***	
RCC	PC14-OSC32_IN	RCC_OSC32_IN	n/a	n/a	n/a	PC14-OSC32_IN
	PC15-OSC32_OUT	RCC_OSC32_OUT	n/a	n/a	n/a	PC15-OSC32_OUT
	PH0-OSC_IN	RCC_OSC_IN	n/a	n/a	n/a	PH0-OSC_IN
	PH1-OSC_OUT	RCC_OSC_OUT	n/a	n/a	n/a	PH1-OSC_OUT
SYS	PA13	SYS_JTMS-SWDIO	n/a	n/a	n/a	SWDIO
	PA14	SYS_JTCK-SWCLK	n/a	n/a	n/a	SWCLK
USART2	PA2	USART2_TX	Alternate Function Push Pull	Pull-up	**Very High ***	
	PA3	USART2_RX	Alternate Function Push Pull	Pull-up	**Very High ***	
Single Mapped Signals	PC3	I2S2_SD	Alternate Function Push Pull	No pull-up and no pull-down	Low	PDM_OUT [MP45DT02_DOUT]
	PA4	I2S3_WS	Alternate Function Push Pull	No pull-up and no pull-down	Low	I2S3_WS [CS43L22_LRCK]
	PA5	SPI1_SCK	Alternate Function Push Pull	No pull-up and no pull-down	Low	SPI1_SCK [LIS302DL_SCL/SPC]
	PA6	SPI1_MISO	Alternate Function Push Pull	No pull-up and no pull-down	Low	SPI1_MISO [LIS302DL_SDO]
	PA7	SPI1_MOSI	Alternate Function Push Pull	No pull-up and no pull-down	Low	SPI1_MOSI [LIS302DL_SDA/SDI/SDO]
	PB10	I2S2_CK	Alternate Function Push Pull	No pull-up and no pull-down	Low	CLK_IN [MP45DT02_CLK]
	PC7	I2S3_MCK	Alternate Function Push Pull	No pull-up and no pull-down	Low	I2S3_MCK [CS43L22_MCLK]
	PA9	USB_OTG_FS_VBUS	Input mode	No pull-up and no pull-down	n/a	VBUS_FS
	PA10	USB_OTG_FS_ID	Alternate Function Push Pull	No pull-up and no pull-down	Low	OTG_FS_ID

IP	Pin	Signal	GPIO mode	GPIO pull/up pull down	Max Speed	User Label
	PA11	USB_OTG_FS_DM	Alternate Function Push Pull	No pull-up and no pull-down	Low	OTG_FS_DM
	PA12	USB_OTG_FS_DP	Alternate Function Push Pull	No pull-up and no pull-down	Low	OTG_FS_DP
	PC10	I2S3_CK	Alternate Function Push Pull	No pull-up and no pull-down	Low	I2S3_SCK [CS43L22_SCLK]
	PC12	I2S3_SD	Alternate Function Push Pull	No pull-up and no pull-down	Low	I2S3_SD [CS43L22_SDIN]
	PB3	SYS_JTDO-SWO	n/a	n/a	n/a	SWO
	PB6	I2C1_SCL	Alternate Function Open Drain	Pull-up	Low	Audio_SCL [CS43L22_SCL]
	PB9	I2C1_SDA	Alternate Function Open Drain	Pull-up	Low	Audio_SDA [CS43L22_SDA]
GPIO	PE3	GPIO_Output	Output Push Pull	No pull-up and no pull-down	Low	CS_I2C/SPI [LIS302DL_CS_I2C/SPI]
	PC0	GPIO_Output	Output Push Pull	No pull-up and no pull-down	Low	OTG_FS_PowerSwitchOn
	PA0-WKUP	GPIO_EXTI0	**External Event Mode with Rising edge trigger detection ***	No pull-up and no pull-down	n/a	B1 [Blue PushButton]
	PB2	GPIO_Input	Input mode	No pull-up and no pull-down	n/a	BOOT1
	PD12	GPIO_Output	Output Push Pull	No pull-up and no pull-down	Low	LD4 [Green Led]
	PD13	GPIO_Output	Output Push Pull	No pull-up and no pull-down	Low	LD3 [Orange Led]
	PD14	GPIO_Output	Output Push Pull	No pull-up and no pull-down	Low	LD5 [Red Led]
	PD15	GPIO_Output	Output Push Pull	No pull-up and no pull-down	Low	LD6 [Blue Led]
	PD4	GPIO_Output	Output Push Pull	No pull-up and no pull-down	Low	Audio_RST [CS43L22_RESET]
	PD5	GPIO_Input	Input mode	No pull-up and no pull-down	n/a	OTG_FS_OverCurrent
	PE1	GPIO_EXTI1	**External Event Mode with Rising edge trigger detection ***	No pull-up and no pull-down	n/a	MEMS_INT2 [LIS302DL_INT2]

8.2. DMA configuration

nothing configured in DMA service

8.3. NVIC configuration

Interrupt Table	Enable	Preenmption Priority	SubPriority
Non maskable interrupt	true	0	0
Hard fault interrupt	true	0	0
Memory management fault	true	0	0
Pre-fetch fault, memory access fault	true	0	0
Undefined instruction or illegal state	true	0	0
System service call via SWI instruction	true	0	0
Debug monitor	true	0	0
Pendable request for system service	true	0	0
System tick timer	true	0	0
PVD interrupt through EXTI line 16	unused		
Flash global interrupt	unused		
RCC global interrupt	unused		
CAN1 TX interrupts	unused		
CAN1 RX0 interrupts	unused		
CAN1 RX1 interrupt	unused		
CAN1 SCE interrupt	unused		
USART2 global interrupt	unused		
FPU global interrupt	unused		

*** User modified value**

I Evaluation Document

I.I Encountered Problems

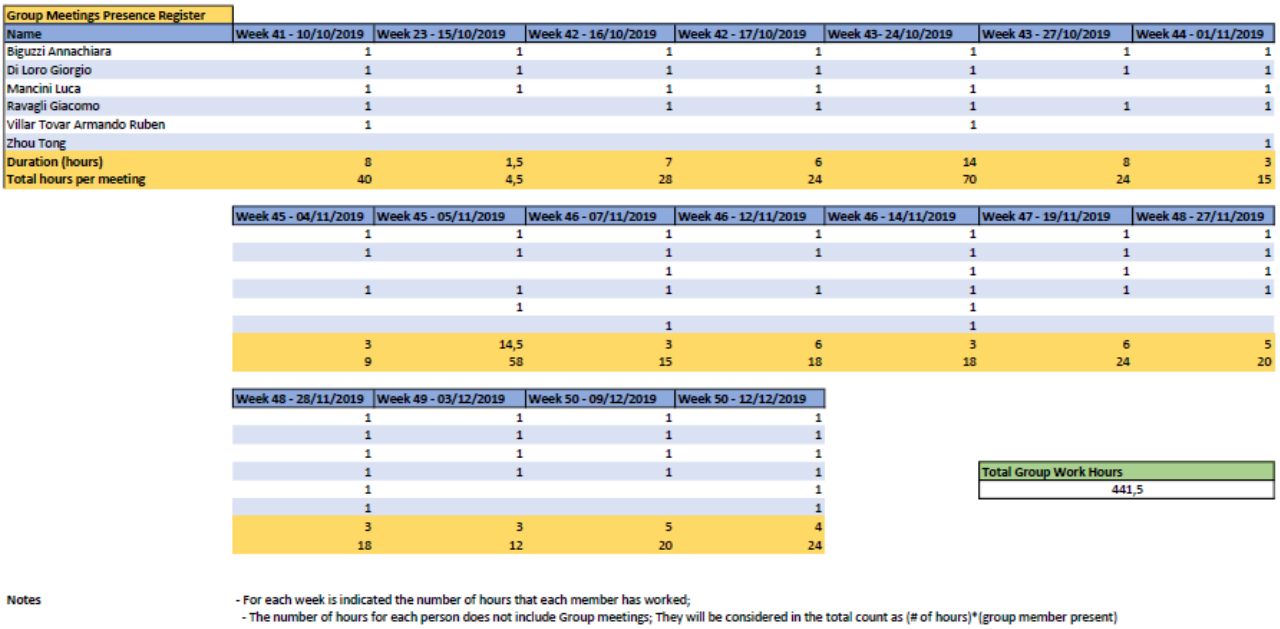
Date	Title	Description
October 31th, 2019	Accelerometer AIS1120SX not in stock on Digikey when the order was sent	AIS1120SX was not delivered with other components. As soon as the BOM was completed, the component was still available on Digikey, however when the whole order was sent it was not in stock. The expected delivery date, after it has been returned to stock, is April 20th, 2020. For this reason it has been chosen to modify the order and use the 2-axis version of the same accelerometer.
November 5th, 2019	Barometric Pressure sensor BMP388 libraries are not compatible with COSMIC compiler	The free compiler COSMIC, supported by STM8 Visual Delevop IDE, does not support int64 data type management. Int64 is used to store intermediate value and it is needed for the calibration phase. To deal with this problem another sensor ha been selected.
November 26th, 2019	CL1 custom soldering	The sensor board does not work correctly, Test 1_MCU not passed. All the soldered components have to be checked.

Table 31: Main problems encountered

I.II Project Working Hours

In this table is not considered the group work aimed to the preparation of the project presentation, because this document has been redacted before.

Total group work for this project has been 441.5 hours. This evaluation considers only group working hours, in addition each team member working hours has to be added.



Group Meetings Presence Register

Name	Week 41 - 10/10/2019	Week 23 - 15/10/2019	Week 42 - 16/10/2019	Week 42 - 17/10/2019	Week 43- 24/10/2019	Week 43 - 27/10/2019	Week 44 - 01/11/2019
Biguzzi Annachiara	1	1	1	1	1	1	1
Di Loro Giorgio	1	1	1	1	1	1	1
Mancini Luca	1	1	1	1	1		1
Ravagli Giacomo	1		1	1	1	1	1
Villar Tovar Armando Ruben	1				1		
Zhou Tong							1
Duration (hours)	8	1,5	7	6	14	8	3
Total hours per meeting	40	4,5	28	24	70	24	15

Week 45 - 04/11/2019	Week 45 - 05/11/2019	Week 46 - 07/11/2019	Week 46 - 12/11/2019	Week 46 - 14/11/2019	Week 47 - 19/11/2019	Week 48 - 27/11/2019
1	1	1	1	1	1	1
1	1	1	1	1	1	1
		1		1	1	1
1	1	1	1	1	1	1
	1			1		
		1		1		
3	14,5	3	6	3	6	5
9	58	15	18	18	24	20

Week 48 - 28/11/2019	Week 49 - 03/12/2019	Week 50 - 09/12/2019	Week 50 - 12/12/2019
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1
1			1
1			1
3	3	5	4
18	12	20	24

Total Group Work Hours
441,5

Notes

- For each week is indicated the number of hours that each member has worked;
- The number of hours for each person does not include Group meetings; They will be considered in the total count as (# of hours)*(group member present)

Figure 125: Group work hours count table

II User Manual

II.I CL1 Custom

In order to proper set-up the system the following steps have to be performed.

- Install the program for data visualization *CL1_USER.EXE*;
- Mount the system on the car wing and connect the two CL1 boards to the supply through connectors.
- The system is provided with the program to be executed that is already present in the MCU memory. In case it is necessary to flash a new program containing an update:

  - Open ST Visual Develop IDE;

  - Connect the board to the ST-LINK for MCU programming;

  PINOUT

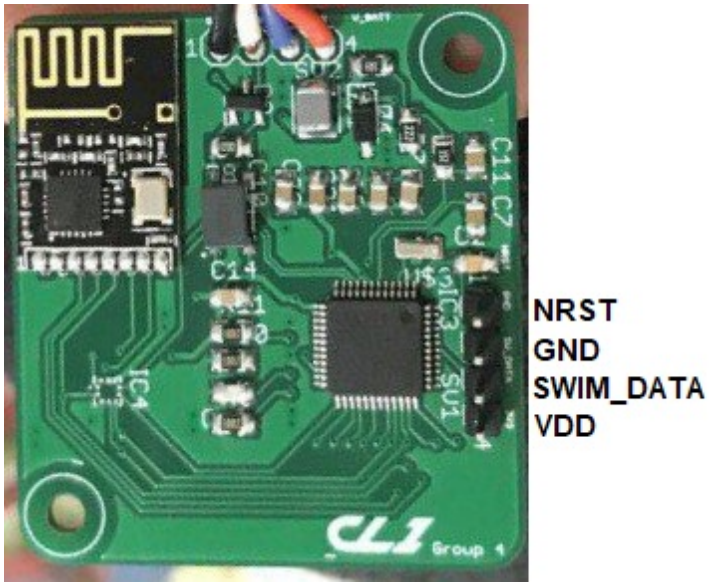


Figure 126: Connection of the ST_LINK for STM8

  - Build and flash the binary file into the MCU memory;

- Open CL1 LabView program;
- Start the execution for data visualization;
- Push STOP when stop the data acquisition;

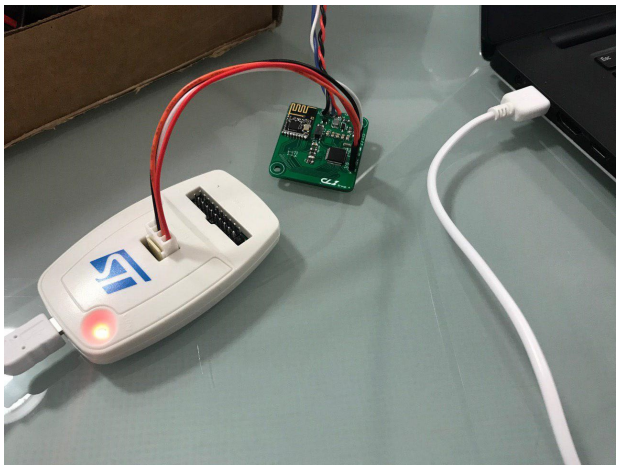


Figure 127: Connection for programming

- A log file with all the data can be found on the Desktop.

Notes

- If there is a configuration error during peripheral initialization a CAN message is sent on the CAN line and automatically recognized by the interface that stops the program. In the following table the messages in different configuration failure.

Error	**Error Message on CAN bus**	**Applies to:**
nRF initialization fail	0x0000000000000001	Wireless Receiver
Barometer initialization fail	0x0000000000000002	Wired Sensor
Accelerometer initialization fail	0x0000000000000003	Wired Sensor

Table 32: Error codes transmitted on the CAN bus

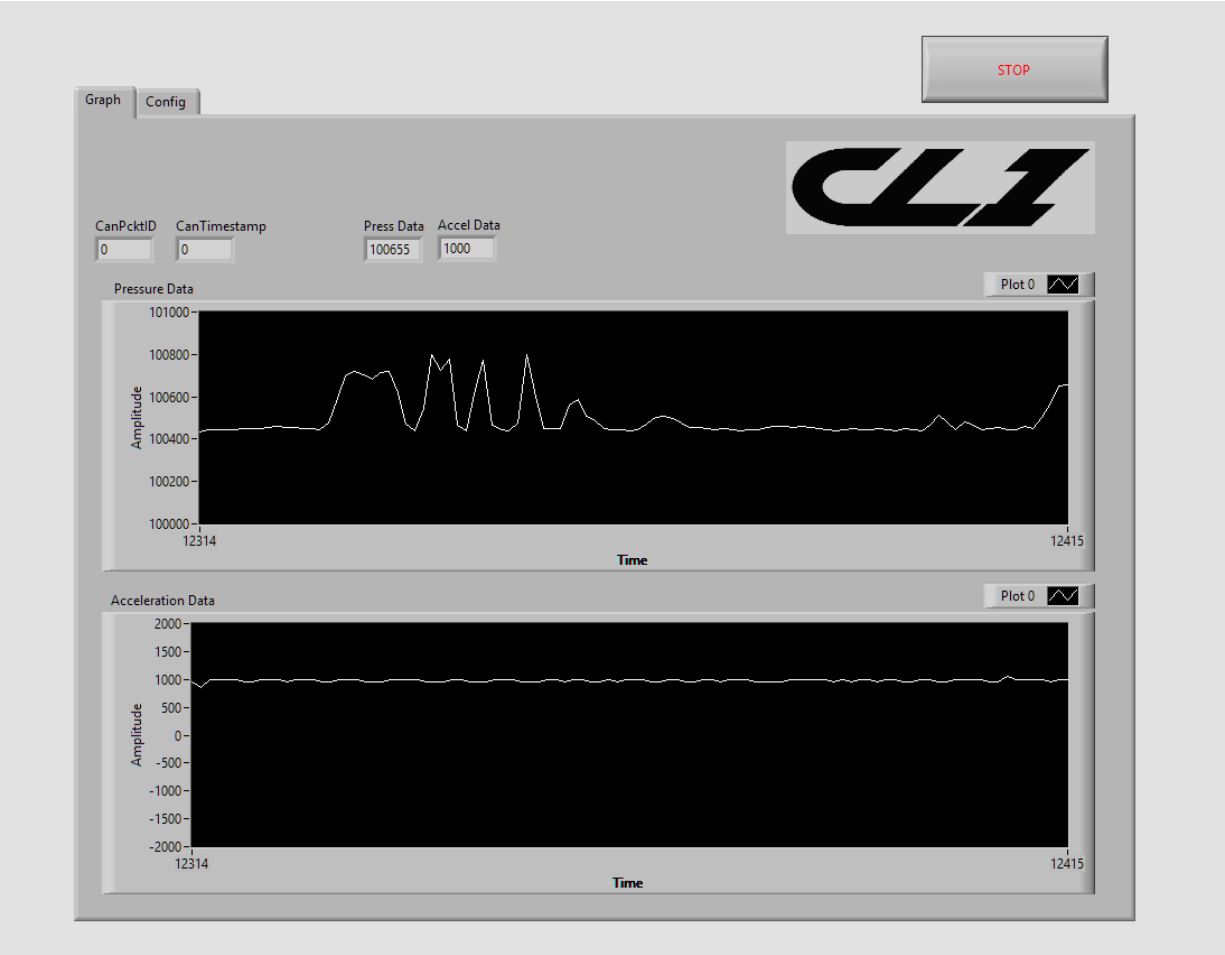


Figure 128: CL1 front panel

References

[1] *STM8AF5288T*, STMicroelectronics https://www.st.com/content/ccc/resource/technical/document/datasheet/e0/31/79/8c/82/d7/40/21/CD00184072.pdf/files/CD00184072.pdf/jcr:content/translations/en.CD00184072.pdf

[2] *AIS3624DQ*, STMicroelectronics https://www.st.com/content/ccc/resource/technical/document/datasheet/ae/89/e4/62/23/de/40/2a/DM00226343.pdf/files/DM00226343.pdf/jcr:content/translations/en.DM00226343.pdf

[3] *BMP388*, Bosch Sensortec https://ae-bst.resource.bosch.com/media/_tech/media/datasheets/BST-BMP388-DS001.pdf

[4] *Comparing CAN- and Ethernet-based Communication* http://canlab.cz/pages/download/artikel_comparison_can_and_ethernet.pdf

[5] *Automotive testing solutions & compliance services* https://www.tuvsud.com/en/industries/mobility-and-automotive/automotive-and-oem/automotive-testing-solutions