



**Universitat Autònoma
de Barcelona**

Human-Machine Interface for Industrial Vehicles

Master's Thesis

Master in Telecommunications Engineering

written by **Eduard Ametller Navas**

and directed by **David Pagès Cisa (IDNEO),**

Jordi Aubert (IDNEO)

and **Joan Oliver Malagelada (UAB)**

Cerdanyola del Vallès, 2 February 2016



The undersigned, Joan Oliver Malagelada
Professor at Escola d'Enginyeria of the UAB,

CERTIFIES:

That the work related with this thesis has been done under his supervision by Eduard Ametller Navas.

And so he signs stating this.

Signed:

Cerdanyola del Vallès, a 29 of January of 2016

Contents

1	Introduction	1
1.1	Objectives	2
1.2	Company description	2
1.3	Product description	2
2	Planning, tools and costs	3
2.1	Planning	3
2.1.1	Project development process	3
2.1.2	Gantt diagram	5
2.2	Tools	7
2.2.1	System	7
2.2.2	Hardware	7
2.2.3	Software	8
2.3	Costs	8
2.3.1	Product	8
2.3.2	Design	9
3	Theoretical concepts	11
3.1	Automotive standards	11
3.1.1	AUTOSAR	11
3.1.2	MISRA	12
3.1.3	SPICE	14

4	Design	17
4.1	System	17
4.1.1	System requirements	17
4.1.2	System architecture	18
4.1.3	Change management system	19
4.1.3.1	Change request information	20
4.1.3.2	Analysis	20
4.1.3.3	Decision	20
4.1.4	Concept art	21
4.2	Hardware	21
4.2.1	Hardware requirements	21
4.2.2	Schematic	22
4.2.2.1	HMI (HW00)	23
4.2.2.2	Reverse polarity and overvoltage protection (HW01)	24
4.2.2.3	Low voltage supply (HW02)	25
4.2.2.4	Microcontroller (HW03)	31
4.2.2.5	CAN transceiver (HW04)	33
4.2.2.6	LEDs driver (HW05)	34
4.2.2.7	Buttons (HW06)	35
4.2.2.8	XBee Module 1 Bluetooth (HW07)	36
4.2.2.9	XBee Module 2 Zigbee (HW08)	36
4.2.2.10	Display (HW09)	36
4.2.2.11	LEDs (HW10)	37
4.2.3	Layout	37
4.3	Software	38
4.3.1	Software requirements	38
4.3.2	Evaluation board	38
4.3.3	Software design	39
4.3.4	Porting the software	42

5 Results	43
5.1 System	43
5.1.1 System requirements	43
5.1.2 System architecture	43
5.1.3 Concept art	43
5.2 Hardware	44
5.2.1 Hardware requirements	44
5.2.2 Hardware schematic	44
5.2.3 Layout	45
5.3 Software	45
5.3.1 Software requirements	45
5.3.2 Software development	45
6 Conclusions	47
6.1 Objectives	47
6.1.1 Innovative	47
6.1.1.1 Hardware	48
6.1.1.2 Software	48
6.2 Enhancements	48
References	49
Annex 1 System requirements	51
Annex 2 System Architecture	61
Annex 3 Hardware and software requirements	63
Annex 4 Hardware schematic	73
Annex 5 Meeting minutes	83

List of Figures

2.1	V design followed by FICOSA.	3
2.2	Gantt diagram of the project.	6
3.1	AUTOSAR schematic.	12
4.1	Excel during the system requirements design.	18
4.2	Enterprise Architect during the system architecture design.	19
4.3	Change Management System workflow.	19
4.4	Screen capture showing Autodesk 3D max 2012 during the design process.	21
4.5	Hardware architecture.	22
4.6	Altium Designer during the hardware design.	23
4.7	Polarity protection circuits.	24
4.8	Circuit for enabling the DC/DC.	26
4.9	Efficiency of the DC/DC converter.	28
4.10	Load map.	29
4.11	Microcontroller supply filtering.	31
4.12	Supply filtering response.	32
4.13	ICD 3 connections to our board.	33
4.14	CAN transceiver recommended configuration.	33
4.15	LED driver configuration for ON/OFF control.	34
4.16	Output current of the LED driver for different values of R_{ext}	34
4.17	Debouncing circuitry.	35

4.18 Component placement sketch.	38
4.19 Software architecture.	39
4.20 MPLAB v8.92 during the design process.	40
5.1 Final release of the concept art.	44
5.2 Representation of the prototype.	45
5.3 Evaluation board running the program.	46

Chapter 1

Introduction

More and more, the vehicles are becoming a complex bunch of electronic devices interconnected forming a network. They need to use the last technologies in order to keep us safe, but also, these devices need to be secure and totally reliable. These devices are not able to fail under some circumstances that could put the driver in risk, that is why automotive is a differentiated subject inside embedded electronics. Any device that will be placed in vehicles has to pass every single test and has to comply with the applicable standards.

That causes an increase of the electronic components and complexity, because every control unit has to communicate any failure, with no matters of the complexity of the function that is doing. As well, if all the units need to communicate its status, it is necessary to form a network of devices. And also it should avoid any disturbances, so it has a lot of electromagnetic compatibility protections.

The aim of this project is to design one of this automotive electronic devices that will sense some information of the vehicle and will be connected to its network. This information will be accessible to the user with some buttons and a display.

This also aims to be a multidisciplinary project. The systems, software and hardware design of this device will be studied. Automotive market is very competitive, with a very aggressive costs, high reliability of designs and developing schedules each year shorter. To reach this quality with short development times, the V development process will be introduced.

But before starting the design itself, a planning should be applied. First, in *Planning, cost and tools*, all the planning and resources will be presented. This section is related with the system architect tasks and the V design process. Its work is to analyze the requirements from the customer in order to distribute the work to the other teams and to keep traceability of the compliance of these requirements. As well, the tools and cost of the final product and of the development will be analyzed.

As well, some standards should be applied in the design process. Then, in *Theoretical concepts*, the applicable standards will be discussed. These standards are the AUTOSAR, the MISRA-C and the SPICE.

This work aims to be multidisciplinary, in order to see all the planning and design process of the product. In section *Design* this process will be presented for the three subjects implied: the system, the hardware and the software.

1.1 Objectives

The following objectives will be achieved.

- Design an automotive electronic device, including the system, hardware and software parts.
- Improve the knowledge in embedded electronics and V design.
- Use the tools and working procedures of the company.
- Create an innovative design making it reusable and modular.

This work will be done in collaboration with the company IDNEO Technologies, S.L that will be now introduced.

1.2 Company description

IDNEO Technologies, S.L. (from now referred as IDNEO) is an company part of the FICOSA group that is specialized in research and development projects. In contrast to FICOSA, IDNEO is not only centered in automotive products, although they are an important percentage of the income of the company.

The business areas of IDNEO are mobility (automotive and railway), medical equipment, consumer electronics and industrial. This project is included in the first area, mobility.

With the purpose of continue growing and being capable to develop more projects, the number of engineers has been increasing since its foundation in 2011, having its peak the past year with 230 employees and a turnover of 17 million euros¹.

1.3 Product description

In order to introduce the device that will be designed, in this section are presented some of its basic properties. More detailed information is shown in the requirements analysis.

The first approach of this project was a device for truck trailers, with a monochrome display, six buttons, one LED and CAN communication. But to have a more flexible device, 3 more LEDs and two XBee modules were added. As well, the project became more general and we focused in create a modular device that can be reused in other future projects and other vehicles.

Chapter 2

Planning, tools and costs

2.1 Planning

2.1.1 Project development process

The organizational structure followed in IDNEO is the V design. This structure is a sequence of processes that goes down from the customer requirements, with a high level of abstraction, to a tangible prototype of the designed device. Later, the processes go up again with the testing part and finishes with the final product. This structure is shown in the following figure and their processes are explained before.

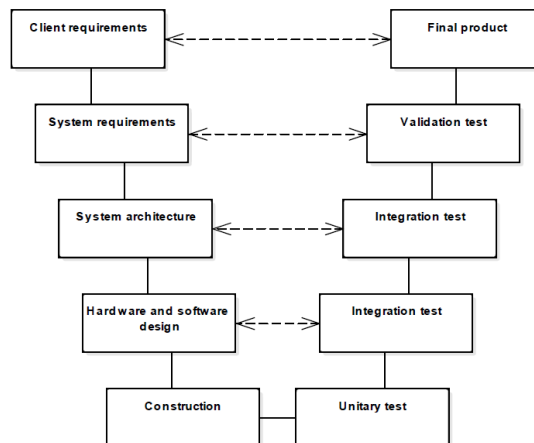


Figure 2.1: V design followed by FICOSA.

Customer requirements

It is the first step of the development process. The customer gives the requirements of what they want. There is not a stipulated way to do so, it can be a list of few points or can be a huge document. Depending on the customer we are more or less freely to interpret what he wants and how it will be done.

System requirements

A list of interdisciplinary requirements with high level of abstraction that define our system. Reading these requirements should be enough to understand perfectly our project at system level. It shall have to be precise and clear and it should not include specific references to the hardware, software or mechanics that will be later in the design. Each system requirement has to satisfy one or many customer requirements.

This document is sent to the customer for its approval. This step is important to assure that the system designed matches perfectly with the customer requirements and expectation.

System architecture

Schematic view of the interdisciplinary blocks that will be implemented and their connections with each other. For example, one of our blocks is the microcontroller block, that will implement a software part and a hardware one. This block will be connected to many other blocks such as the CAN transceiver or the buttons blocks.

Hardware and software requirements

Hardware and software specifications are required, such as the minimum performance of a component or the use of specific programming libraries. These requirements shall satisfy the system requirements. So it has to be a traceability between them, that is to say, it has to be checked which hardware and software requirements satisfies each system requirement.

Hardware and software design

The hardware and the software are designed using the appropriate tools and following the information of the previous steps. It corresponds to the schematic and layout for the hardware and the code and compiled program for the software.

Construction

Once we have the PCB layout, a prototype of the board is manufactured. At the end of this process we have a hardware mounted and a software ready to be downloaded to the microcontroller. The design phase has finished and starts the test and validation processes.

Unitary test

After the construction process start the testing ones. The first test process is the unitary test, that tries to verify if the hardware and software design has been done correctly.

In the hardware department it is tested if there is a correct voltage on power signals, if there are short circuits between test points and they do a visual inspection of correct polarity of each component and soldering.

In the same way, all the software functions are tested. The test checks that each individual function works as expected, for example, writing concrete inputs and checking that the outputs have the expected values.

Integration test

This test procedure verifies if the subsystems are interacting correctly with each other. This step validates the system architecture.

It's divided in two parts. One is the software integration, checking the correct interface between all the software blocks. Later, there is the integration between software and hardware, where it is checked that the software is working as expected with the hardware. This implies for example in checking that the microcontroller pins are correctly assigned.

Validation test

The validation test assures that the requirements have been met. It validates the system requirements.

First of all a validation test plan has to be done. It has to be approved by the customer. For software functionalities of the system, each requirement must be covered for at least one test in a functional validation test plan. The test has a description, the setup to execute and a clear acceptance criteria. On the other side, in the same way there is also an environmental test plan and an EMC and electrical test plan. Unlike the functional validation, this two test plans are not entirely designed by the company, the customer specifies some standards that has to comply and the test plan has to be developed in order to comply the content of this standards.

2.1.2 Gantt diagram

The Gantt diagram was developed using GanttProject program for Windows. The diagram is presented in the next figure and is commented after that.

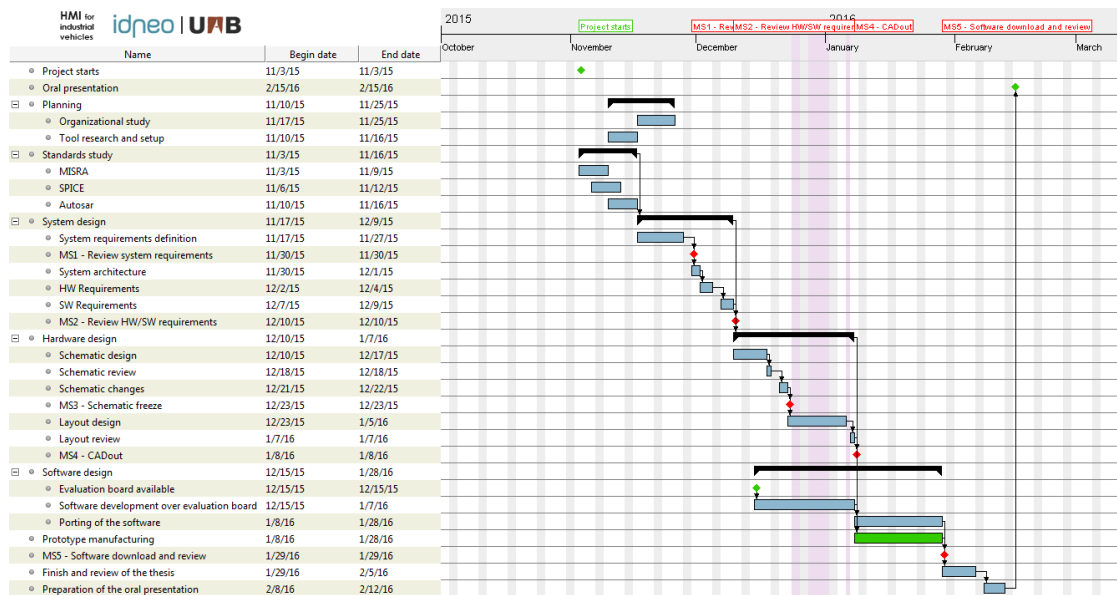


Figure 2.2: Gantt diagram of the project.

The project starts on November 3 with the internship contract for this project. Later, a planning is done with the study of the project development process and the tool research and setup. Next, some standards are studied, including MISRA, SPICE and AUTOSAR.

The system design part will consist in the design of our board at a system level. We have here the first milestone (MS1) after the system requirements definition. Also we group the hardware and software requirements development although it is not part of the system design, but we have to put one single milestone for the whole requirement definition (MS2).

The following step is the hardware design, where the schematic and layout of the PCB will be designed. Each of this parts will finish with a milestone, so we have one for the schematic freeze (MS3) and another for the CADout (MS4).

Meanwhile, the software will be developed in an evaluation board and will be downloaded later in a manufactured prototype done by a lab in FICOSA. This corresponds to the software download and review (MS5).

After that, on January 29 (coinciding with the last day of the contract), the project will be closed and the thesis and presentation will be finished.

2.2 Tools

2.2.1 System

We have several tools in order to manage the requirements and design the system part. First they will be presented and later it will be compared in order to choose the best option.

DOORS

DOORS (Dynamic Object Oriented Requirements System) is a software developed by Telelogic, that was later acquired by IBM. It is used for requirement management and traceability. It works as a client-server application.

Enterprise Architect

Enterprise Architect is a software tool developed by Sparx Systems. It is based in UML(Unified Modeling Language) and has an extensive list of utilities in addition of the requirement management. It also allows to generate different types of diagrams, such as use cases or state machines. It supports SysML diagrams too, a derivation of modeling language UML for systems engineering applications.

Excel

Is the well-known spreadsheet based in cells developed by Microsoft. Although it is not a system requirement oriented software, it can be used perfectly for the requirement analysis.

So finally, Enterprise Architect was chosen to create the system architecture. That was because it gives a fast way to create standardized SysML diagrams and also I have some experience using this program .

On the other hand, for the requirements management and traceability, the Microsoft Excel tool was chosen. As all the requirements are done by the same person we tried to find a way to do it simple but effective. Other tools include more functions but they are more useful in projects with more people and constant changes.

2.2.2 Hardware

For the hardware part was used the Altium Designer tool. Some freeware alternatives were studied for the layout part but was easier to ask for a full license of Altium.

Altium Designer

Altium Designer 14.0 is a software developed by Altium Limited. Among other functions, it allows to draw the schematic and layout in order to design our circuit. Being able to draw everything in one tool is an advantage because it is easier to review the layout.

2.2.3 Software

Next we have the tools used in the software design and some explanation about them.

Evaluation board

An evaluation board will be used in order to start programming the microcontroller before having the final board. Among different possibilities, the Explorer 16 evaluation board was chosen because of its compatibility with the Graphics PICTail Plus Board. These graphics board is an extension with a TFT Samsung S6D0129 display with resistive touch screen.

Compiler

There are two available compilers for the dsPIC family, the C30 and the XC16. There are 60 days evaluation versions of these compilers. Both of this compilers were evaluated and the C30 was finally selected, although the XC16 compiler is newer. This selection is justified later in this thesis.

IDE

The last versions of the two IDEs series provided by Microchip were evaluated. They are the MPLAB IDE v8.92 and the MPLAB X IDE v3.15. As well as happened with the compiler, the IDE selected is the oldest one, the MPLAB IDE v8.92 and the selection will be also justified in the software design part of this thesis.

Microchip Graphics Library

For the development of the interface in a high level of abstraction, the Microchip Graphics Library v2.10 was used.

2.3 Costs

2.3.1 Product

The cost per unit for prototyping and for production was calculated and can be seen in the following table^{ab}.

^aThis is an engineering estimation, the sales department shall do in the future an official budget.

^bThere are other costs like the SMD soldering mask, the assembly, the cost of the end of line test and the manufacturing fees that depend on the number of units for production that have to be added. For prototyping this cost is estimated in 21.8€.

Component	Type	Prototype cost (€)	Cost for production (€)
dsPIC33EP512MU810	Microcontroller	8.87	6.44
IPD50P03P4L-11	Power transistor	0.84	0.30
MAX17542G	DC/DC	3.00	1.98
MAX16910	LDO	1.29	0.63
TJA1043	CAN	1.65	0.81
BCR401U	LED drivers	4 x 0.56	4 x 0.11
LB TTSD/LR T67F	LEDs	1 x 0.72 + 3 x 0.18	1 x 0.32 + 3 x 0.08
TEL0073	BLE XBee	16.47	14.64
Xbee ZB Trace antenna	Zigbee XBee	23.82	21.44
EA DOGXL240W-7	LCD	46.09	34.25
EA LED94X67-W	LCD Backlight	38.39	23.75
SM4T39CAY	TVS	2 x 0.83	2 x 0.27
NX1255GB	Oscillator	1.06	0.74
PCB	-	50.6	1.5
Other	Rs, Cs, Ls, etc.	2.08	1.6
Total		199.32	109.62

Table 2.1: Cost of the product prototype and for mass production.

2.3.2 Design

The design cost in IDNEO is calculated using a standard cost for the employees hours (this standard cost is the half for internship students). In this cost per hour it is also included other costs than the employees work, like the equipment (such as the computers), the prototyping cost, the tools... In the following table, there are presented the hours done in the different parts of the project.

	Hours
Planning	21
Standards study	14
System design	52
Hardware design	52
Software design	17
Total	156

Table 2.2: Hours breakdown.

In the next equations we calculate the total cost of the design obtaining a value of 3478.8€.

$$Cost = Hours \cdot \frac{Internship\ cost}{Hour} \quad (2.1)$$

$$Cost = 156 \cdot 22.3 = 3478.8€ \quad (2.2)$$

Although all the cost of the project is specified in the previous calculations, in the following table there is also presented the real cost of some of the equipment used. The rest of the tools that are not specified can be obtained for free or are discontinued.

	Real cost (€)
MISRA-C:2004	14.11
Explorer-16	120.83
dsPIC33EP512MU810	8.87
MPLAB ICD 3	183.96
Altium Designer	3675.59
Enterprise Architect	642.19
Total	4645.55

Table 2.3: Cost of the project equipment.

Chapter 3

Theoretical concepts

3.1 Automotive standards

3.1.1 AUTOSAR

AUTOSAR (AUTomotive Open System ARchitecture) is a partnership of vehicle manufacturers, suppliers and other companies from the electronics, semiconductor and software industry. There are three types of partners inside the structure, that define their rights and duties: core, premium and associate partners. Some examples of well-known companies that are partners of AUTOSAR are Ford, Toyota, Infineon, NXP or Altium. FICOSA is also an associate partner².

The objective of this organization is to provide a common software architecture for automotive systems and standardized interfaces between its layers. The structure that they define is based in the following groups, ordered from more abstract to more hardware related³:

- **AUTOSAR Application Software** is the application that satisfies the ECU functions. It is not hardware related.
- **AUTOSAR Run Time Environment (RTE)** communicates the application with the basic software.
- **AUTOSAR Basic Software** are standardized software modules that offer services necessary to run the functional part of the upper software layers. Except the MCAL modules, is not hardware dependent.
- **MCAL (Microcontroller Abstraction Layer)** are basic software modules that directly access the microcontroller peripheral modules and external devices that are mapped to memory, and make the upper software layers independent of the microcontroller. This allows us to program in a higher abstraction level and to port the software to different hardware only changing these modules.

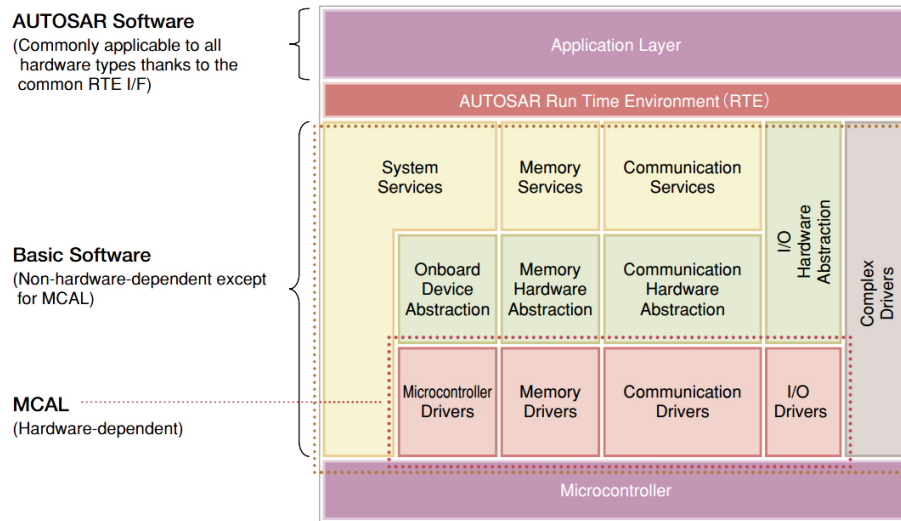


Figure 3.1: AUTOSAR schematic.

In order to manage AUTOSAR and apply its infrastructure to our software design some tools are needed. One of this tools used in IDNEO is the Tresos product line from Elektrobit. It has a lot of different tools for the design and testing of software compliant with the AUTOSAR standard.

3.1.2 MISRA

C is one of the most used languages for embedded applications. The reason is that it gives access to low-level and high-speed operations in addition of being more abstract than assembly language. This makes easier to develop complex applications maintaining a good performance, something that is not achieved by other higher level languages⁴.

In spite of these advantages, there is no guarantee that the final executable code will behave as the programmer intended. Its code is vulnerable to different types of errors, like typing errors, misunderstandings from the coder, compiler errors or run-time errors⁴.

The MISRA (Motor Industry Software Reliability Association), a consortium formed from representatives of different companies, solved these counterparts by defining some rules that have to be followed in order to avoid possible fonts of errors. These guidelines are defined in a document, officially known as MISRA-C:1998 or “Guidelines for the use of the C language in vehicle based software”. Two more editions were released from the first MISRA-C edition in 1998, the MISRA-C:2004 and the MISRA-C:2012⁵.

These guidelines have been widely adopted in safety-critical software applications. Although they are not mandatory, the customer usually demands compliance of these rules. Even if the customer does not demand this, these rules are almost always applied.

Like the ISO and IEC normatives, MISRA-C is not an open standard. The document containing the guidelines has to be bought in order to apply the rules and check for its conformance⁶.

MISRA-C is based in ISO C and supposes a prior knowledge of the language at the moment of applying the rules. They exclude other non-standard versions of C or the C++ language.

For the first two editions, the rules are classified into “required” or “advisory”. The “required” rules are mandatory and the “advisory” ones should be followed as far as they are practical⁷.

In MISRA-C:2012 there are also “mandatory” rules. In this case, the “mandatory” rules can never be broken, while the “required” ones can be broken only with the approval of a manager. The “advisory” are programmer’s responsibility. In this version the rules are also divided in “decidable” (an analysis tool can always determine its compliance) and “undecidable” (have to be reviewed manually)⁷.

The number of rules for the different editions is^{4,8}:

Edition		Mandatory	Required	Advisory
MISRA-C:1998		-	93	34
MISRA-C:2004		-	122	20
MISRA-C:2012	Decidable	4	86	26
	Undecidable	6	27	10

Table 3.1: Number of rules in the MISRA editions.

These rules are presented with the following format⁴:

```

1 Rule <number> (<category>): <requirement text>
2                               [<source ref>]
3 Normative text
```

The <number> label defines the unique number of the rule. <category> is one of the above classifications (mandatory, required, advisory). <requirement text> is the text of the rule. <source ref> is a reference to published source where the rule originates. The requirements of the normative text should be applied in order to conform with the MISRA-C standard⁴.

The rules are organized in several topics. For example, in the “Language extensions” section, we can find a rule that does not allow us to use the “//” comment style that is not supported by all the compilers. Instead, we should use the “/* ... */” style. Other topics that MISRA regulates are for example “types”, “constants”, “functions”, “pointer type conversions”...⁴

The C software will be compliant if all the “mandatory” rules are met, the “required” rules are applied or subject to a formal deviation and the “advisory” rules are met or recorded if they are not applied.

The software team should try to apply this standard when they code in C, and also later review for its conformity. In order to simplify the verification of compliance there are several tools that check for MISRA conformance.

The MISRA edition followed now in IDNEO is MISRA-C:2012, as it is understood that the last version of the standard at the moment of starting a project should be used.

FICOSA is compliant with the 127 rules of MISRA-C.1998 except for five deviations that are properly documented and justified. Polyspace is used for detecting run-time errors before the code is compiled and executed.

MISRA-C also demands the definition of a style guide that should include⁴:

- Code layout and use of indenting.
- Layout of braces “{ }” and block structures.
- Statement complexity.
- Naming conventions.
- Use of comment statements.
- Inclusion of company name, copyright notice and other standard file header information.

This is properly defined in the FICOSA Handbook.

A ISO C compliant compiler should be used whenever possible. Also the use of C++ compilers for compiling C code should be avoid, although there is no other option, it can be used if the differences are fully understood. If the compiler is marketed as C++ but has a conforming ISO C mode, it can be used in this mode.

3.1.3 SPICE

ISO/IEC 15504, also known as SPICE (Software Process Improvement and Capability dEtermination), is a standard that proposes some measures in order to determine the quality and improve the processes of an organization. However, it does not fix a way of developing these processes, only a method to evaluate the model of these processes^{9,10}.

The definition of the standard is general and does not specify the type of processes, but it is normally used in software development processes^{9,10}.

The first draft of the standard was written in June 1995. The first publication as a technical document was in 1998. Between 2003 and 2006, the first five parts were released. Later, five more parts have been developed from the international community⁹.

In August 2005, the Special Interest Group (SIG) created the Automotive SPICE, an adapted version of SPICE for automotive purposes. The most important automotive brands are part of the SIG and demand a certain level of conformity with this standard¹¹.

The SPICE normative has different parts. The two most important parts are the part 2 and part 7. The first one does an evaluation of the capability of the processes. On the other hand, part 7 defines the evaluation methods for the maturity levels. There are six of these levels¹⁰.

Although SPICE offers examples on how to apply it to the software development with the model of processes ISO 12207, they are only examples and its appliance is not mandatory. Based on these examples there are different schemas to determine the processes that have to pass in order to determine the maturity levels, like the Pathfinder or the AENOR models¹⁰.

FICOSA has level 2 for some of the activities and processes of SPICE. It was certified by AENOR.

In SPICE, the work is divided in processes. Every process has the following information: Process ID, Process name, Process purpose, Process outcomes, Base practices and Output work products¹².

Usually the organizations demand to the supplier a certain level of SPICE or CMMI. If the company does not have one of these (or both) certifications, it will be difficult for them to achieve the assignment of a project¹⁰. At this moment all customers require minimum level 2, but level 3 is desirable.

Chapter 4

Design

4.1 System

4.1.1 System requirements

Before starting the design process, we need to have clear what the customer wants. This part of the chain always depends on the type of customer that we have for the project. Some of them are strictly clear on what they want and how it has to be done. Others let you work with full freedom and you only have to respect some few requirements.

But in any case is important to distinguish between what they want and what we can do. The assignation of a project from the customer starts with an RFI (Request For Information). With the RFI the customer can see the capabilities and knowledge of the suppliers, and also a first price for development and unit cost. It is usually a first filter.

This document is usually followed by a RFQ (Request For Quotation). It is the official request. The customer requests a design proposal, with development cost and unit price. This is the official offer and the document used for the future project development.

Initially this project was intended to be a part of a possible outside customer project, but later the project was not assigned to us. The project was then reoriented by the business unit director, Jordi Aubert. So in this case, it turned to be the customer. From different meetings there was set an idea of what shall be done, so the important points extracted are:

- The project shall be as modifiable as possible, in order to be applied fast in future projects for different environments and applications, for example with the use of XBee modules.
- The main idea shall be the same, a graphic interface between the machine and the user, with a little display and some LEDs.

- It shall be usable in automotive applications.

Having these points in mind, the system requirements were defined in an Excel sheet for later traceability by me and reviewed by the system architect David Pagès. After some revisions, the final version of the system requirements (HMI_SYS_REQ_V1.2) was developed and exported into a document that can be seen in Annex 1.

ID	Description	Category
1	The document specifies the system requirements for the Human-Machine Interface for Industrial Vehicles, now on referred as HMI.	
2		
3		
4	The HMI is an Electronic Control Unit (ECU) that will be placed in an industrial vehicle and shall act as an interface between the user and the vehicle information.	General
5	The HMI aims to be used in multiple applications, that is why it shall be modular and reusable.	General
6	In order to work as an interface, the HMI shall include 6 buttons to make possible to the user to select some options. On the other hand, there shall be a display, 4 LEDs and a Bluetooth module to give the user information. This information will be retrieved from CAN communication and Zigbee from other ECUs.	General
7	A connector shall be placed in order to allow CAN communication, power supply and to connect the external LEDs.	Connector
8	The HMI shall be fully operational with a power supply range between 9V and 32V. Outside this values the HMI shall go to off mode without any damage.	Power supply
9	The ECU shall measure the voltage and current of the power supply.	Power supply
10	The HMI shall incorporate a microcontroller that shall be able to handle the display.	Microcontroller
11	The HMI shall be able to communicate with other ECUs using a CAN BUS. It shall be able to retrieve information such as the vehicle speed, alerts/warnings status and car lights state.	CAN
12	The transmission procedure shall be CAN FD High Speed with a transmission rate of 1 Mbps.	CAN
13	The HMI shall be waked-up using CAN BUS.	CAN
14	The HMI shall communicate with other devices using two Xbee modules. The first shall be a Bluetooth module, the second shall be a Zigbee module.	Xbee
15	The Xbee Bluetooth module shall allow the download of information to another compatible Bluetooth device in a range of 5 meters.	Xbee Bluetooth

Figure 4.1: Excel during the system requirements design.

As you can see in the previous figure, each requirement has a unique identifier. This is important for the links and traceability with other documents, like the architecture or the tests.

4.1.2 System architecture

Once the system requirements are done, next step is to define the system architecture based on that requirements. It is high abstraction level block design of the project. It was defined in two levels, the first defining the interactions of our system with the rest of the car and the second of its internal block architecture.

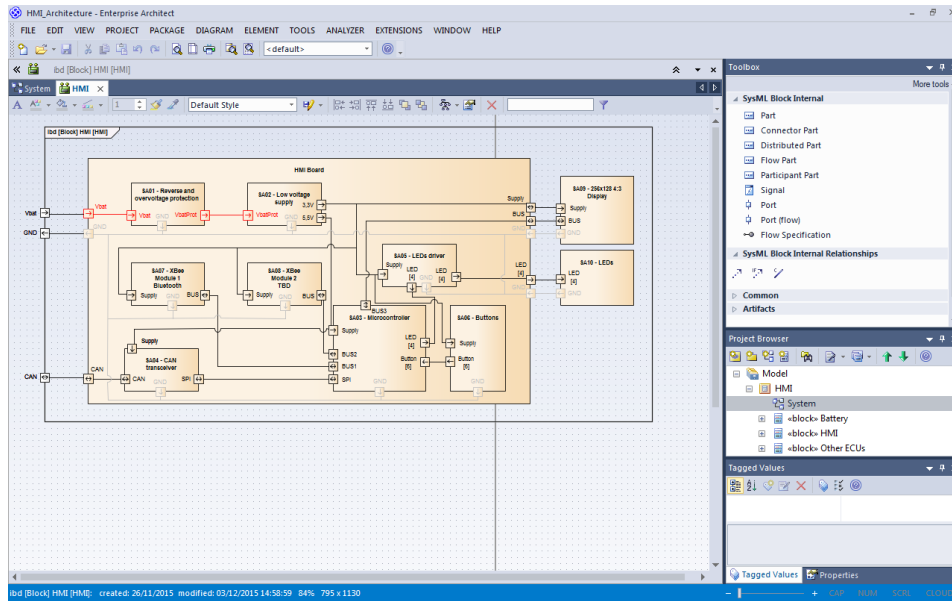


Figure 4.2: Enterprise Architect during the system architecture design.

4.1.3 Change management system

During the design process, there was a change request from FICOSA. One of their teams was involved in a similar project, although they needed one more XBee module. The following steps were followed in order to manage this change.

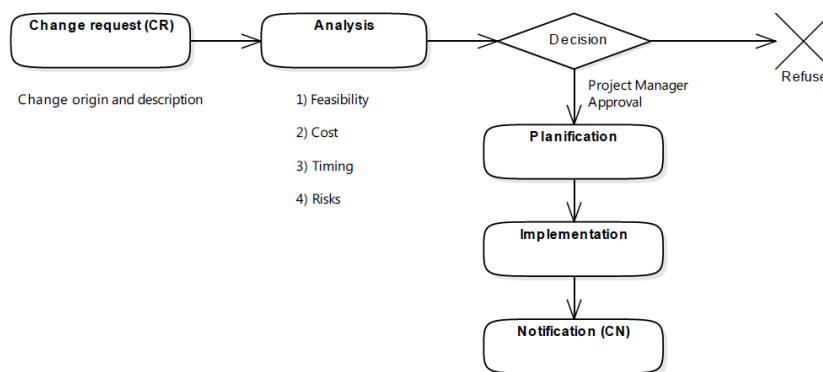


Figure 4.3: Change Management System workflow.

4.1.3.1 Change request information

Origin of the change The change was proposed by FICOSA Solutions, a department of the FICOSA company.

Description This department wants an ECU with a display for an electric motorcycle. This ECU should have Bluetooth, global positioning and another wireless communication technology. All this three systems shall be accomplished using XBee modules, so a third module socket should be placed in order to allow the design to be used in this project.

4.1.3.2 Analysis

Feasibility After connecting the rest of the circuitry to the microcontroller, there are only 18 pins available for connecting the third XBee: 7 can be configured as input or output and 11 are I/O. The module needs 17 microcontroller pins: 1 input, 5 outputs and 11 I/O. So it is possible to add the third XBee module. As well, the low voltage supply seems to have enough power to supply the new module. However, if we need more pins of the microcontroller in the future we will not be able to use them.

Cost There will be a cost increase, but there are no cost restrictions for this project.

Timing The change request was done during the schematic design. This module will have the same footprint of the other XBee, so it will not affect the timing.

Risks Some risks can be possible interferences between modules or that the microcontroller was not powerful enough to control everything.

The interferences problem can be minimized taking care of this paths while doing the layout and adding some circuitry to avoid interferences.

4.1.3.3 Decision

As it seems that is possible to apply the change, it will be implemented as a variation in the future. This supposes the creation of the system block SA11 - XBee Module 3 and the hardware block HW11 - XBee Module 3.

4.1.4 Concept art

In order to visualize the concepts defined by the system requirements, a concept art was designed. The design was done using Autodesk 3D max 2012 and mental ray renderer.

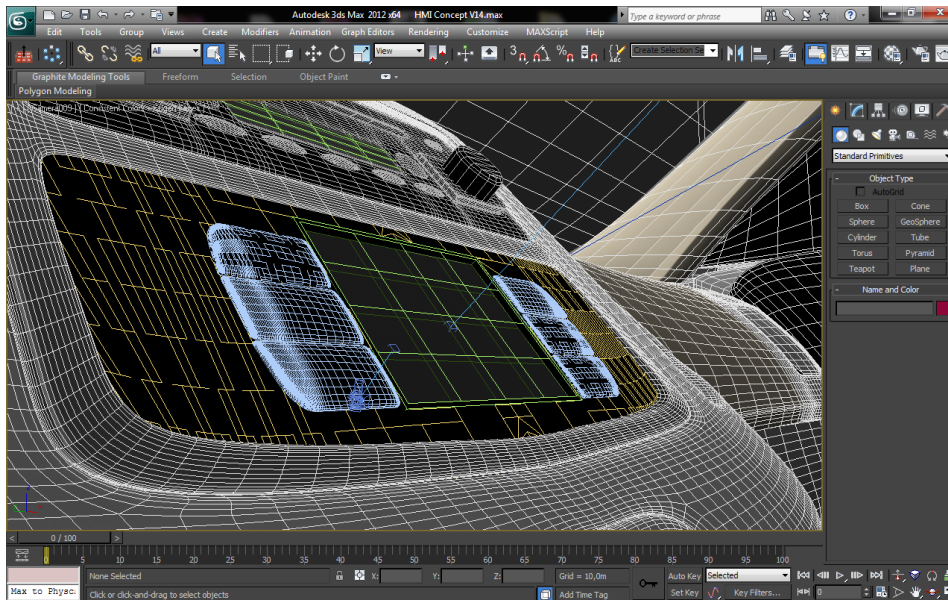


Figure 4.4: Screen capture showing Autodesk 3D max 2012 during the design process.

4.2 Hardware

4.2.1 Hardware requirements

The same process we have seen in the system requirements definition was followed, but now searching hardware related specifications.

In the following lines there are some considerations that justify the design of the requirements:

- As an ECU for industrial vehicles, it has to allow to be supplied with a voltage range from 9V to 32V.
- As an automotive hardware, the operation temperature has to be between -40°C and 80°C .
- A display with LCD and backlight was suggested for simplicity against other options, as for example OLED, that changes a lot its consumption with display changes and needs better filtering.
- The microcontroller was chose having in mind its graphics capabilities and because it was previously used in the company.

- The ECU shall not crash in any circumstance, so this protections shall be added: overvoltage, reverse polarity, short circuits, open circuits, direct connection to ground and to battery, EMC and ESD.

4.2.2 Schematic

First of all, a hardware architecture was developed from the hardware requirements. It defines the hardware blocks and high level of abstraction connections between them, that has been a base for the schematic design.

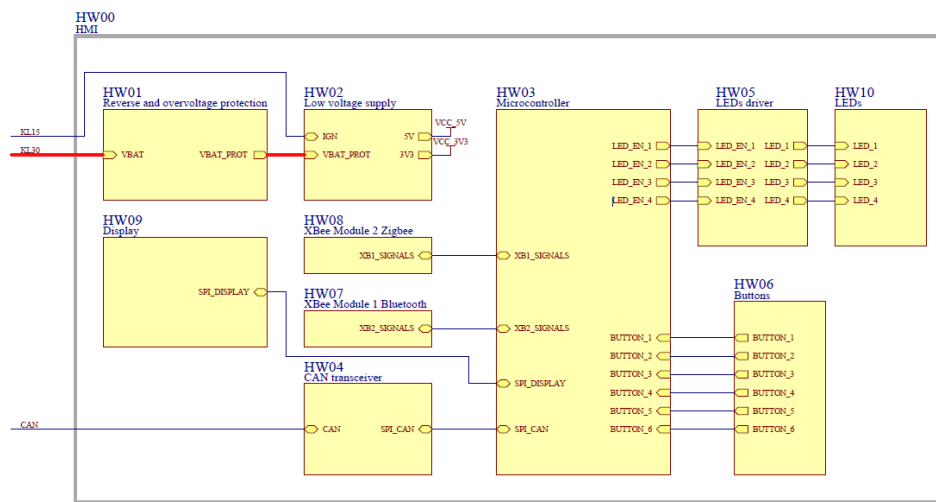


Figure 4.5: Hardware architecture.

The hardware schematic was developed using Altium Designer. In the following subsections, the implementations of the hardware blocks are presented and justified. It is recommended to check Annex 4 during the reading of this subsection.

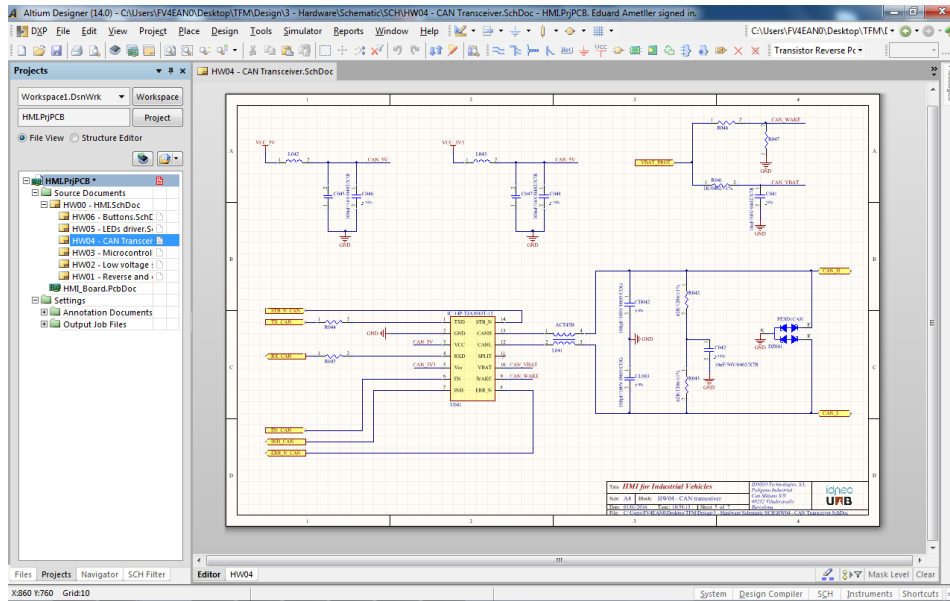


Figure 4.6: Altium Designer during the hardware design.

4.2.2.1 HMI (HW00)

The rest of the subblocks are defined in this block, as well as the connections between them and some circuitry like the connectors and sockets. In this subsection will be done an introduction to the functioning of this blocks and the relations between them.

First of all, the signals and supply voltage from the vehicle are connected using a 14 pin connector from MOLEX. The power supply is connected to the Reverse polarity and overvoltage protection (HW01). This block protects the rest of the board from perturbations in the supply voltage and is connected to the Low voltage supply (HW02).

The Low voltage supply block converts the 9V to 32V of the input into two voltage supplies of 5V and 3V3. With this supplies, the processing part of the circuit, the Microcontroller (HW03) can start working. This block can receive information from the CAN transceiver (HW04), enable the LEDs driver (HW05), obtain the state of the Buttons (HW06) and receive and send information via UART communication with the XBee Modules (HW07 and HW08).

This modules are connected with a socket into the board, as well as the LCD of the Display (HW09), that will be controlled using SPI.

4.2.2.2 Reverse polarity and overvoltage protection (HW01)

Transient-voltage-suppression

First of all, a transient-voltage-suppression (TVS) diode was added in order to protect the circuit from spikes. This component conducts when there is a voltage higher than its breakdown voltage, so it will be our overvoltage protection. In this case, the SM4T39CAY has a minimum breakdown voltage of 36.7V.

This component is also prepared to filter the standard pulses 1, 2a, 3a and 3b defined in the ISO 7637-2. This pulses are perturbations of the supply voltage caused by other devices connected to the battery, that are typical and well known.

Reverse polarity protection

Next, three reverse polarity protection circuits were considered. They are presented in the following figure and explained in the following paragraphs.

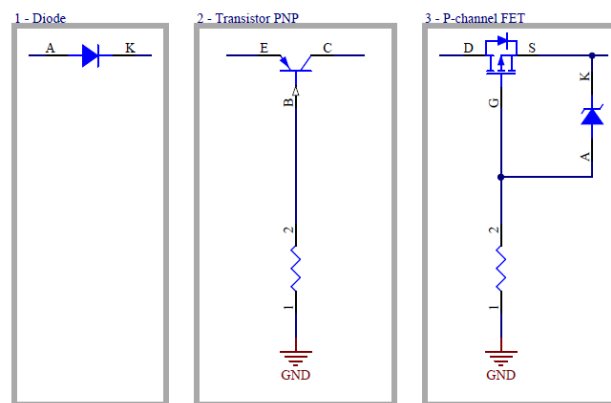


Figure 4.7: Polarity protection circuits.

The first one is the diode configuration. It consists in using a diode, so the advantage is its simplicity and cost. On the other hand, the diode produces a voltage drop that reduces the supply voltage in around 0.7 volts.

The second one is the PNP transistor. The voltage drop and power dissipation are lower than the diode. The disadvantage is that there is a constant power loss from the base current and it does not depend on the power draw by the supplied circuit.

The third one is the P-channel FET. Is the best option in voltage drop and power dissipation but it has also a higher cost. This option is used when there's a big current demand for the unit.

We will use the diode option because we can assume the voltage drop, it is a cheaper solution and we will not have a huge current to handle.

However, we will use a Schottky diode in order to reduce the voltage drop and therefore, the dissipated power.

$$P_{Diode} = V_T \cdot I_{Drain} = 0.49V \cdot 253.44mA = 124.19mW \quad (4.1)$$

We have to clarify if the diode will be able to dissipate this power. For that we will multiply the worst case thermal impedance R_{th} ($^{\circ}C/W$) with the dissipated power in the worst ambient conditions.

$$T_{Max} = T_{Amb} + R_{th} \cdot P_{Diode} \quad (4.2)$$

$$T_{Max} = 80^{\circ}C + 100 \frac{^{\circ}C}{W} \cdot 124.19mW = 92.4^{\circ}C \quad (4.3)$$

This diode can be used because the maximum temperature achievable ($92.4^{\circ}C$) is lower than the maximum temperature allowed by the component ($150^{\circ}C$).

High power filtering

Finally, we have to say that now on, there will be used two series capacitors instead of one for the high power filtering. This is because when the capacitors break, they do a short circuit between its pins. The capacitors are formed by layers of metal and dielectric, and if one of this layers is broken, there may be contact between them. If there is a short circuit between the power source and the ground, the board can burn.

So there are placed two capacitors in a 90° angle in order to avoid that the two components break due to vibrations. There are also safe-fail capacitors that are designed in order to break as an open circuit. If there is a short circuit in the low voltage supplies, the DC/DC and the LDO can detect it and stop the supply, so there are no safety issues for this capacitors.

4.2.2.3 Low voltage supply (HW02)

Power supply and ignition

As it is required to be for industrial vehicles, the board has to work with a supply voltage between 9V and 32V. Obtaining this wide operation range is only possible using a DC/DC converter, instead of a linear regulator. This converter will give the low voltage supply of 5V. In order to obtain the 3.3V supply, a linear regulator is used with the 5V supply of the DC/DC converter.

A way to switch on and off the ECU is needed. For this purpose a KL supply will be used. The name of this lines come from “klemme”, the German word for terminal. This supply voltages are obtained from the vehicle and depend on the key position. The KL supplies correspond with the following situations¹³:

- KLR means ignition switch position 1 (accessory).
- KL15 is ignition switch position 2 (on).
- KL50 is ignition position 3 (start).
- KL30 is battery positive, at high voltage all the time.
- KL31 is battery negative, connected all the time.

The one that we will use to enable or disable the DC/DC converter (and consequently the linear regulator and all the ECU) will be the KL15. The following figure show the circuit that we use to enable the DC/DC, and will be explained afterward.

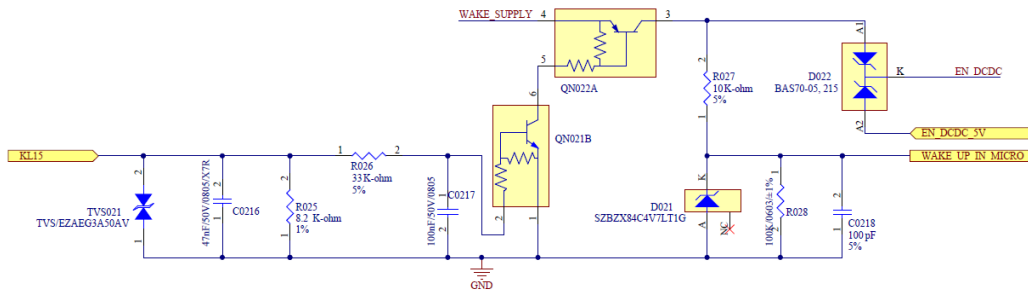


Figure 4.8: Circuit for enabling the DC/DC.

First, the KL15 signal is filtered and protected with the TVS, resistors and capacitors. This signal goes to the base of a NPN transistor. Its collector goes to the base of the PNP transistor. With this configuration, we obtain the WAKE_SUPPLY in the collector of the PNP when KL15 is on. In this case, for the WAKE_SUPPLY we used the protected and filtered battery voltage.

Later, the collector of the PNP is connected to the anode of one of two faced Schottky diodes, that in this case are working as an OR gate. The cathode of this diodes is the enable of the DC/DC.

On the other anode, we have a signal from the microcontroller that can keep the DC/DC alive, usually until it does some closing operations. But it also needs to know when the KL15 is low in order to start turning itself off. For this reason we have a WAKE_UP_IN_MICRO signal that goes to an input of the microcontroller.

Load map

As well, in order to choose and set up a DC/DC we need to make first a load map in order to know the power and current that the converter will need to supply. It is done by calculating the current consumption of the different blocks and doing the summation. This value is known for some blocks and can be seen

in the load map schematic of the next figure. But for the microcontroller (hardware block 3), the buttons (hardware block 6) and the LEDs (hardware block 5), some calculations shall be done.

We can start with the microcontroller. The datasheet of our microcontroller determines that it consumes 1mA for every MHz of the clock. We have also to add the current consumption of the pull-ups for the inputs.

$$I_{HW03} = 8MHz \cdot 1 \frac{mA}{MHz} + 24inputs \cdot \frac{3.3V}{2K\Omega} = 47.6mA \quad (4.4)$$

We have to assume the worst case in order to work under any circumstances. In the case of the buttons block, the worst case is when all the buttons are pressed and we have a resistance between the low voltage supply and ground.

$$I_{HW06} = 6Buttons \cdot \frac{3.3V}{82K\Omega} = 0.24mA \quad (4.5)$$

The drivers in the LEDs block are configured for a maximum consumption of 20mA. We can add the consumption of the four drivers and obtain the total block consumption.

$$I_{HW05} = 4LEDs \cdot 20mA = 80mA \quad (4.6)$$

Once we have all the blocks consumption, we can obtain the current consumption at the input of the LDO and the DC/DC. First, for the LDO, the input current is the same as the output current. However, the product of the voltage and the current will be different in both input and output. This means that the LDO will dissipate some power in the conversion process. These losses are included in the input power, so we do not have to take it into account in future calculations.

Now we have to calculate the current consumption of the DC/DC, therefore, the HMI current consumption. In the following figure we can see the efficiency of our DC/DC converter.

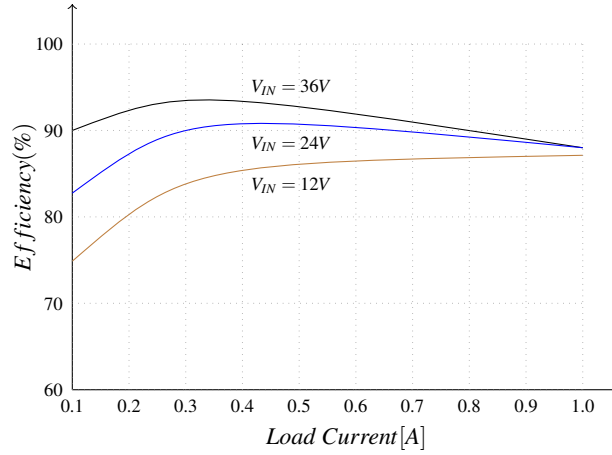


Figure 4.9: Efficiency of the DC/DC converter.

The current consumption of the total HMI board will be determined by the voltage applied and will change the total power consumption as the efficiency of the DC/DC depends on the voltage input. For example, in a car the nominal voltage is 13.5V, so the current consumption can be calculated as we see in the following equations.

$$I_{Consumption} = \frac{I_{DC/DC} \cdot V_{DC/DC}}{\eta_{DC/DC} \cdot V_{Input}} \quad (4.7)$$

$$I_{Consumption} = \frac{253.44 \cdot 5}{0.93 \cdot 13.5} = 100.93mA \quad (4.8)$$

The worst case for the current consumption is set in the minimum 9V voltage input.

$$I_{Consumption_{WC}} = \frac{253.44 \cdot 5}{0.95 \cdot 9} = 148.21mA \quad (4.9)$$

Increasing the voltage will make the current consumption decrease, but for high voltages this reduction will be reduced caused by the reduction of the efficiency.

On the other hand, the worst case for the power consumption is set in the maximum 32V voltage input, as we can see in the following equations.

$$P_{HMI} = \frac{I_{DC/DC} \cdot V_{DC/DC}}{\eta_{DC/DC}} + P_{Diode} \quad (4.10)$$

$$P_{HMI} = \frac{253.44 \cdot 5}{0.85} + 124.19 = 1.61W \quad (4.11)$$

The final load map can be seen in the following figure.

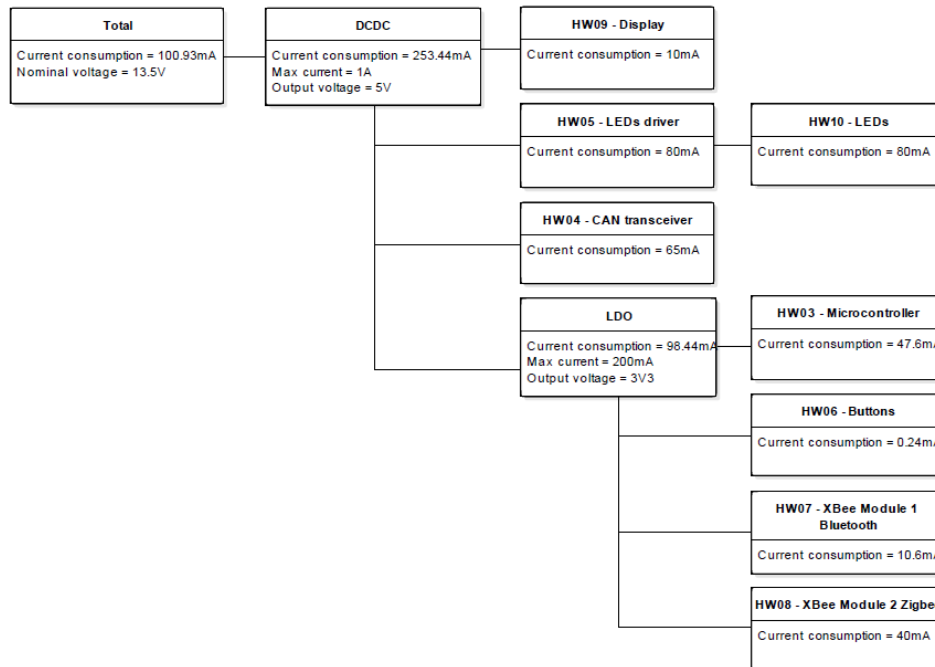


Figure 4.10: Load map.

Once we know the limits of operation what need to have our component, we selected the MAX17542G DC/DC converter. But now we also need to set up this component in order to operate in this conditions. In this type of converters, the supply to the inductor is switched on and off all the time. When it is on, the inductor is charged of energy and also supplies the load. When the switch is off, the inductor continues supplying current to the load. So we obtain a constant voltage with a little ripple due to this commutations and current changes in the load. This ripple also depends on the selected input and output component values of capacitance and inductance.

So what we need to do now is to calculate this values in order to choose the correct components. In this case, the manufacturer determines that the inductance value shall be calculated with the following equation.

$$L = 4 \cdot V_{Out} = 4 \cdot 5 = 20\mu H \quad (4.12)$$

We also have to choose an inductor with a saturation current higher than the peak current limit of the converter and with a low DC resistance. Finally, the inductor selected was the B82477G4223M000, with a saturation current of 4.25A and 0.038 Ω of maximum resistance.

The output capacitor value of the DC/DC has to be calculated in order to obtain a desirable maximum

deviation of the output voltage. The following equation is used for defining the output capacitor value, where ΔV_{Out} is the output voltage deviation, I_{Step} is the load current step and $t_{response}$ is the response time of the controller.

$$C_{Out} = \frac{1}{2} \cdot \frac{I_{Step} \cdot t_{Response}}{\Delta V_{Out}} \quad (4.13)$$

For calculating the $t_{response}$ we use the following equation, where f_C is the target closed-loop crossover frequency and f_{SW} is the switching frequency (600kHz in this case).

$$t_{Response} \cong \frac{0.33}{f_C} + \frac{1}{f_{SW}} \quad (4.14)$$

We assume a load current step of 194.84mA, as this is the maximum change in the load from stand by mode to active mode. The f_C value is set to $\frac{1}{12} \cdot f_{SW}$ in order to avoid instabilities. We will try to get a maximum output voltage deviation of 0.08V. With this values and doing the calculations, we obtain a result of $10\mu F$.

$$t_{Response} = \frac{0.33}{\frac{600 \cdot 10^3}{12}} + \frac{1}{600 \cdot 10^3} = 8.27\mu s \quad (4.15)$$

$$C_{Out} = \frac{194.84 \cdot 10^{-3} \cdot 8.27 \cdot 10^{-6}}{2 \cdot 0.08} = 10\mu F \quad (4.16)$$

Now the LDO have to be set up. The LDO will be supplied with the 5V output of the converter, so it will not need fail-safe o double capacitors in the input filter. The same structure will be used as an output filter for the 3V3. But there is also another thing to have into account because it supplies the microcontroller.

If the microcontroller is enabled when the supply voltage is low, it can produce some errors in the functioning. With the purpose of enabling the microcontroller only when the supply is ready, we use the RESET pin of the LDO. This pin is an output to the MCLR of the microcontroller that enables it when the supply is stable. There is also a TIMEOUT pin that sets a delay after the supply is ready in order to assure that everything is ready for a correct operation.

By default, the delay when TIMEOUT is left unconnected is $60\mu s$. For calculating the equivalent delay if we connect a capacitor, the following equation has to be used, where $C_{Timeout}$ is our capacitor value, I_{TO} is the TIMEOUT Ramp Current ($1\mu A$) and $t_{Timeout}$ is the delay.

$$C_{Timeout} = 0.8 \cdot I_{TO} \cdot t_{Timeout} \quad (4.17)$$

For the recommended value of $1.25ms$ delay, the capacitor value has to be $1nF$.

$$C_{Timeout} = 0.8 \cdot 1 \cdot 10^{-6} \cdot 1.25 \cdot 10^{-3} = 1nF \quad (4.18)$$

4.2.2.4 Microcontroller (HW03)

Microcontroller interfaces

The microcontroller used is the dsPIC33EP512MU810 from Microchip. In the following table, some important information for the hardware design is presented.

Component	Pins	Package	UART	SPI	I2C	I/O pins
dsPIC33EP512MU810	100	TQFP	4	4	2	83

Table 4.1: Technical characteristics of our dsPIC microcontroller from Microchip.

It is important to keep in mind the interfaces that will be needed before doing the pin connections. This is the count: 2 UARTs for the XBee, 1 4-wire SPI for the display, CAN TX/RX and enables, 6 buttons inputs and 4 LEDs enable.

Low voltage supply filtering

The continuous switching operation of the microcontroller changes continuously the power supply consumption. This can affect other devices, so a decoupling filter is needed. This filter is shown in the following figure.

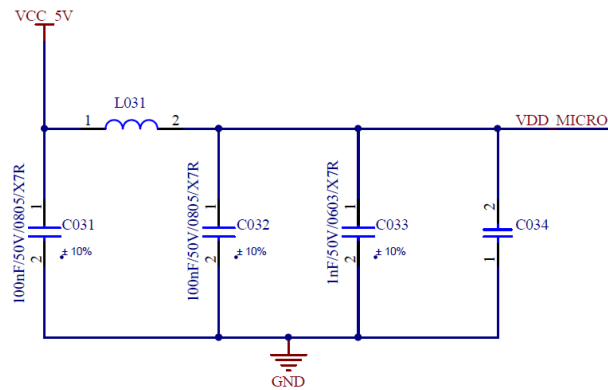


Figure 4.11: Microcontroller supply filtering.

The capacitors, as a non ideal capacitance, have an ESR (Equivalent Series Capacitance) and an ESL (Equivalent Series Inductance). These values depend on the package of the component^a. As we choose different values of capacitance, they will resonate in different frequencies. But we can maintain its

^aUsually the packages have more inductance as the size grows, except if their shape changes. In this case it can be lower. In the case of the 0612 package, that has the pads in the longer side, unlike the rest of the known packages.

impedance at the resonating frequency changing its package, i.e., their parasitic inductance. So we obtain a greater refused band for our filter¹⁴.

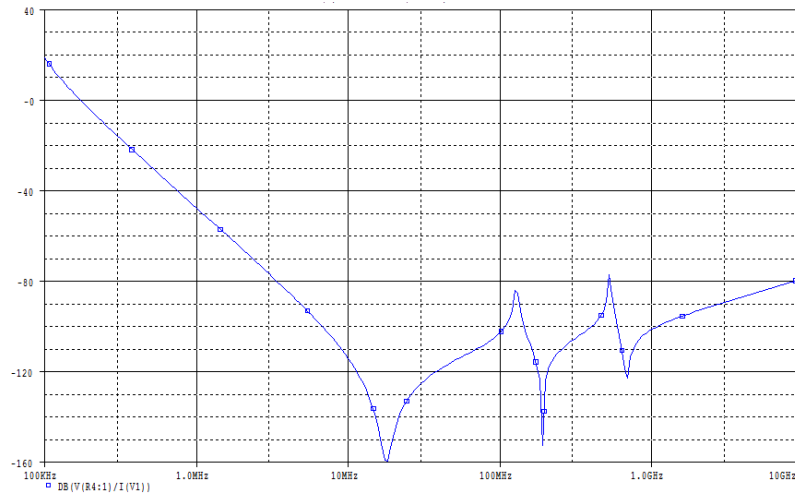


Figure 4.12: Supply filtering response.

Oscillator

An 8MHz external oscillator was used, instead of the internal ones, because we need this precision in order to work without errors while using a CAN transceiver. Two $12pF$ capacitors are connected between the oscillator ports and ground with the purpose of starting the oscillation.

Programming interface

There is also to be set some circuitry with the intention of programming the microcontroller. But before that we need to define in which way the microcontroller will be programmed. For this purpose we will use the ICD 3 in-circuit debugger that will be connected to the PC via USB. This device only has a RJ11 connector for programming the board, so a cable will be crafted in order to connect it to our 5 pin header connector.

In the following figure we can see the configuration and connections needed to program the dsPIC33E with the ICD3.

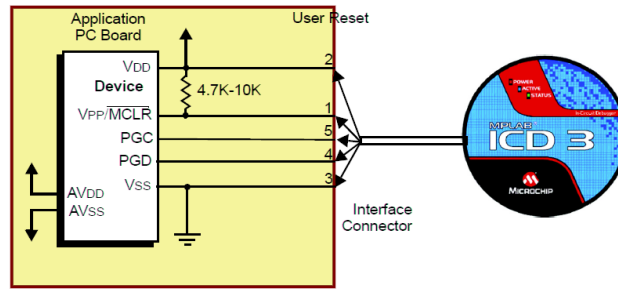


Figure 4.13: ICD 3 connections to our board.

In this case we shall not use filter capacitors, diodes or other signal conditioning except for the pull-up resistor in \overline{MCLR} .

4.2.2.5 CAN transceiver (HW04)

The high speed CAN transceiver TJA1043 from NXP was used for the CAN block design. The component was configured as it is recommended in the application hints for a correct EMC performance¹⁵.

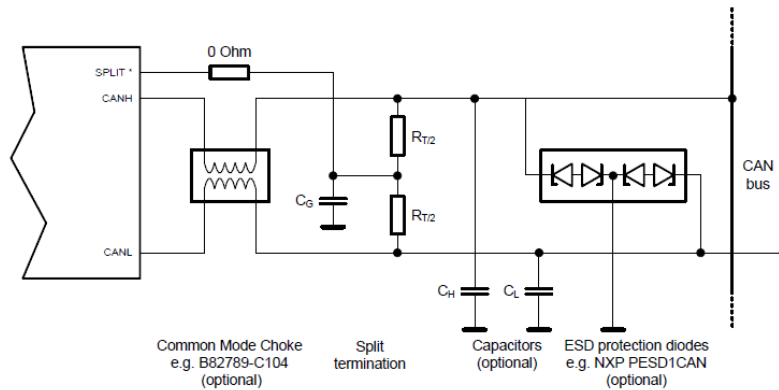


Figure 4.14: CAN transceiver recommended configuration.

The capacitors C_H and C_L act as a low-pass filter, so the associated corner frequency must be above the data transmission frequency. The value recommended for a data rate of 500kbit/s is 100pF.

On the other side of the transceiver, there are set two 100Ω resistors in the Rx and Tx signals that go to the microcontroller. This reduces the ringing effect, and therefore electromagnetic emissions, produced by bouncing in the commutations of the switches that drive this CAN signals.

4.2.2.6 LEDs driver (HW05)

In order to maintain the same level of brightness in spite of supply variations^b, the Infineon BCR401U LED driver was used. The configuration used can be seen in Figure 4.8 and was found on the applications notes of the component. The LED_EN signal activates the driver, and the LED signal supplies the power to the component.

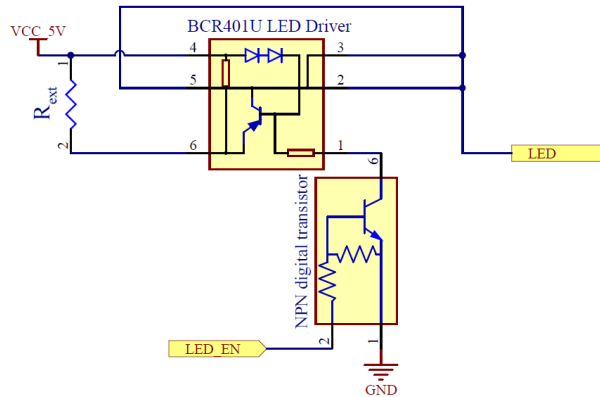


Figure 4.15: LED driver configuration for ON/OFF control.

The value of R_{ext} was chosen using the following graphic found in the datasheet of the component. A value of 100Ω was chosen for a current output of $15 - 20mA$.

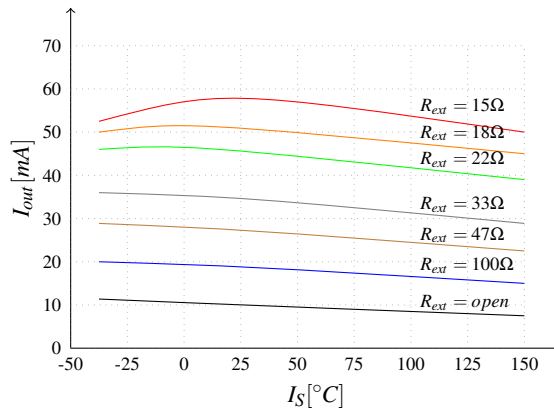


Figure 4.16: Output current of the LED driver for different values of R_{ext} .

^bAlthough the supply will be very stable as we use a DC/DC, other variations as the forward voltage drop between LEDs or the components tolerances will change the current value and, therefore, the brightness. The LED driver provides a constant current and brightness.

With this current value, our LEDs will have a luminous intensity of 450mcd . The chosen components were the LR T67F-U1 (red) and the LB TTSD-T2 (blue).

In applications where it is important for the user to have the exact same brightness for every LED, we use the integrated circuit drivers like the Infineon one and also a selection of LEDs with the same brightness. This LEDs is needed to be selected properly, because the fabrication process makes them different from each other and the manufacturer gives you the option to buy bins with low brightness differences that have been previously selected.

As not all the applications need such good drivers like those, one low cost alternative was also designed. With this other option, variations in the voltage supply or the component tolerances can affect the current output and, therefore, the brightness emitted by the LED. This alternative implementation is formed by a PNP transistor with a resistance calculated for a current flow of 10mA to the LED.

4.2.2.7 Buttons (HW06)

It has been used a simple debouncing circuit¹⁶. This debouncing could also have been done by software, but is not worth it if we can solve the problem adding a little more circuitry. The debouncing circuit used can be seen in the following figure.

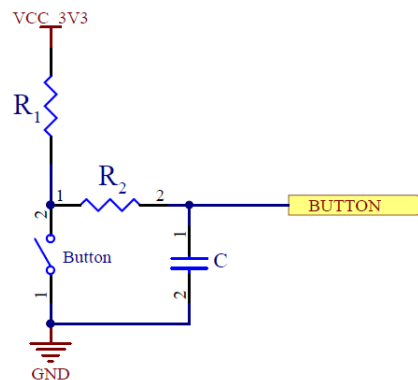


Figure 4.17: Debouncing circuitry.

Now we have to calculate the resistance and capacitor values. The discharge capacitor equation can be rearranged to get the following equation, where t is the debouncing time, C is the capacitance, V_{th} the threshold value and $V_{initial}$ the .

$$R = \frac{-t}{C \cdot \ln\left(\frac{V_{th}}{V_{initial}}\right)} \quad (4.19)$$

$$R = R_1 + R_2 \quad (4.20)$$

All input and I/O ports feature an Schmitt trigger for improved noise immunity. This means that the logic input only changes to 1 or 0 when the voltage input goes over or under a certain threshold. In this case, the bottom threshold is $0.2 \cdot V_{DD}$.

It is suggested to make a 15 to 20 milisecond debouncing in order to avoid the bounce of most of the switches. Only some few inputs are 5V tolerant, so we have to use the 3V3 low power supply for our debouncing circuit.

$$R = \frac{-0.015}{10^{-6} \cdot \ln\left(\frac{0.2 \cdot V_{DD}}{V_{DD}}\right)} \quad (4.21)$$

$$R = \frac{-0.015}{-1,61 \cdot 10^{-6}} = 93K\Omega \quad (4.22)$$

So having this result in mind, two possible values are $R_1 = 82K\Omega$ and $R_2 = 18K\Omega$, using standard resistance values. This means an exact debouncing time of 16.5 miliseconds.

4.2.2.8 XBee Module 1 Bluetooth (HW07)

This hardware block is formed by an XBee module and the header connectors that connects it to the main board of the HMI. An XBee module is a hardware board that can be connected to our board design and act as an interface for wireless communications. The dimensions and operation of this modules are standardized.

A TEL0073 XBee from DFRobot was used as the first module. It uses a CC2540F256 integrated circuit from Texas Instruments. It is a low power system-on-chip for Bluetooth 4.0 applications. It has a programmable flash memory of 8KB that can be accessed using the microUSB incorporated in the XBee board. This SoC communicates with the dsPIC microcontroller via UART, although some other interfaces can be configured.

4.2.2.9 XBee Module 2 Zigbee (HW08)

For the second module we used a XBee ZB with PCB antenna from Digi International Inc.¹⁷. It also communicates with the HMI main board using UART communication and is connected to the board using a two header connectors.

4.2.2.10 Display (HW09)

The display used is a EA DOGXL240W-7 with the EA LED94X67-W backlight from Electronic Assembly. This components are thought to be bought in low quantities and for prototyping, so seemed a good idea to use it in our design. It also allows SPI communication with its UC1611 controller. The display is connected to the main board using two 20 pin vertical header connectors.

4.2.2.11 LEDs (HW10)

The external LEDs have to be mounted in 4 external little PCB boards with a 2 pin connector that will be connected directly to the 14 pin connector of the main board. This two pins will give supply and ground. This boards will be placed under the indicators in order to illuminate them.

4.2.3 Layout

With the hardware schematic we can now design the PCB layout. This design allows the component placement and interconnection when it is printed into the board. Usually, the layout design is done by the CAD department. The components in the schematic also contain information about the footprint of the components, that are loaded in the PCD Editor of Altium when the creation of the layout begins. In this case, due to the lack of time, the layout was planed to be done by myself in nonworking time. But for some problems found, the development of this part was delayed. However, the guidelines of the layout and a representation of it was done. As well, the layout design will be developed later in the company.

The PCB will consist on a 4 layer layout that will be distributed as follows, from the top to the bottom:

1. All the components will be placed in this layer. There will be path between the components.
2. This layer will be used for the ground. The maximum area possible will be covered in order to reduce the resistance between the ground and the components. This ground plane will also help to avoid interferences and act as a shielding for some parts of the circuit.
3. Reserved for the low voltage supplies with the same concept as the ground, big areas in order to reduce resistance.
4. More paths between components.

The different layers will be connected using vias. The border of the PCB will also be covered by vias, what creates a Faraday cage for EMC protection. There is also recommended to connect every component directly to ground, rather than connecting them together after doing the ground connection¹⁸. Another important thing is to prevent interferences in the CAN avoiding to cross signals and not passing lines under the line choke.

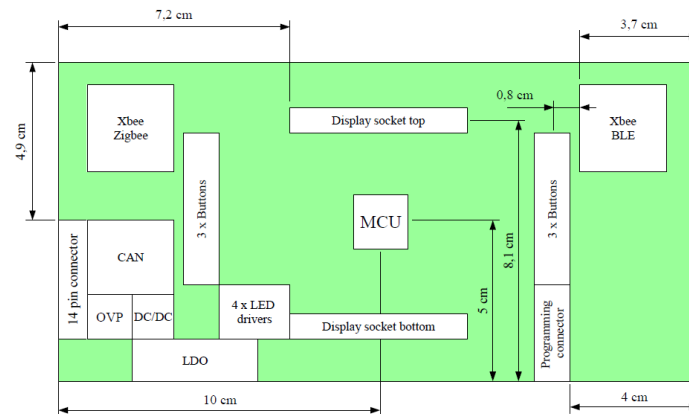


Figure 4.18: Component placement sketch.

4.3 Software

4.3.1 Software requirements

The same process we have seen in the system and hardware requirements definition was followed, but now searching software related specifications.

4.3.2 Evaluation board

In order to reduce the development time and accomplish the project timing, an evaluation board is used for software development. The board used is the Microchip Explorer-16, that has some features like:

- A socket for microcontrollers of the PIC and dsPIC families.
- PICTail Plus connector for expansion boards.
- 4 buttons, 8 LEDs and some connectors for programming.
- Alpha-numeric 16 x 2 LCD display.

The expansion board Graphics PICTail Plus Board Version 1 was used. This board has a Samsung TFT display with resistive touchscreen and a S6D0129 controller.

The evaluation board was programmed using the MPLAB ICD 3 in-circuit debugger. This tool allows to download the compiled hex files into the board.

The software was developed using MPLAB IDE v8.92 and the C30 compiler, that is no longer supported by Microchip. As well, the Graphics Library version is the v2.10, an old release.

This outdated configuration was necessary in order to be able to use the display expansion board, that is also not supported by Microchip anymore. Although this configuration, some changes were done in the libraries in order to support the display.

4.3.3 Software design

Like we did for the hardware part, a software architecture was developed at the beginning as a base for our software design.

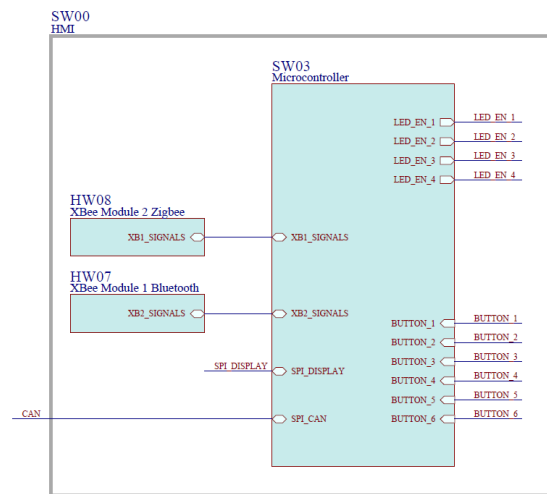


Figure 4.19: Software architecture.

Although there are different modules, only the microcontroller part will be only discussed in the following lines. This is because we are using an evaluation board and we do not have the XBee modules yet.

As we are programming in a different hardware than the final one, the software development can not be hardware dependent. This means that the part that we are developing is the Application Layer. The Basic Software Layer is provided by Microchip, and here we can include the Microchip Graphics Library, that provides services to our application but it is not hardware dependent.

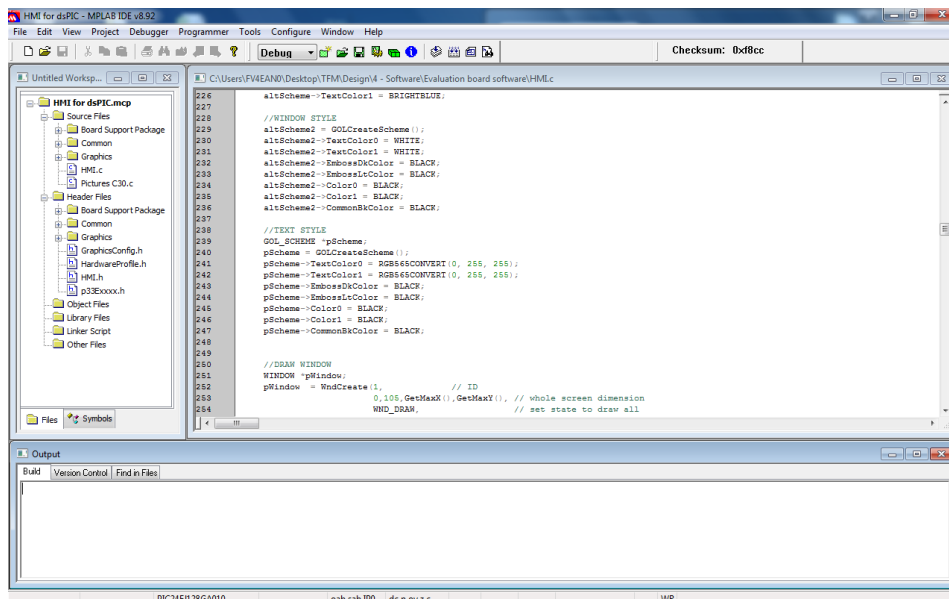


Figure 4.20: MPLAB v8.92 during the design process.

Before start programming, we have to set up our project correctly. This means that we have to edit the GraphicsConfig and HardwareProfile header files, that allow us to select between the different graphic controllers and microcontrollers compatible with the Explorer-16 board. This means that the compiler will use predefined code of the Microchip libraries in order to define the hardware specific layer .

After that we can begin the application software development. First of all, we have to define a style scheme for our future objects. This style is stored in a struct of the type GOL_SCHEME that has a list of variables that let us change every color or font. In the following box there is an example code for creating and editing an style.

```

1 GOL_SCHEME *StyleScheme ;
2 StyleScheme = GOLCreateScheme () ;
3
4 StyleScheme->TextColor0 = WHITE;
5 StyleScheme->TextColor1 = WHITE;
6 StyleScheme->EmbossDkColor = BLACK;
7 StyleScheme->EmbossLtColor = BLACK;
8 StyleScheme->Color0 = BLACK;
9 StyleScheme->Color1 = BLACK;
10 StyleScheme->CommonBkColor = BLACK;

```

Later, we will set a background with the company logo and some other static elements of the interface. With this purpose the following lines will be used.

```

1 WINDOW *pWindow; // Create the pointer to our new object
2
3 // Create the Window
4 // pointer = WndCreate(ID , LEFT, TOP, RIGHT, BOTTOM, Initial state ,
   Bitmap, Text, Scheme);
5 pWindow = WndCreate(WND1_ID, 0, 0, GetMaxX(), GetMaxY(), WND_DRAW, (void
   *) &IDNEO_BG, NULL, StyleScheme);
6
7 WndDraw(pWindow); // Draw the object on the screen

```

As we can see in this lines, with this function we only need to specify the object position and size, the bitmap pointer and the style scheme.

The bitmaps are imported using the Graphics Resource Converter, a Microchip tool that converts the image file directly into memory address values in C code.

If we want to draw now the text that is displayed, we have to code the following lines.

```

1 STATICTEXT *pString; // Create the pointer to our new object
2
3 // Define the text on the string
4 char TextString[] = "Text string validation.";
5
6 // Create the String
7 // pointer = StCreate(ID , LEFT, TOP, RIGHT, BOTTOM, Initial state , Text,
   Scheme);
8 pString = StCreate(STR1_ID, 15, 205, 305, 240, ST_DRAW, TextString ,
   StyleScheme);
9
10 StDraw(pString); // Draw the object on the screen

```

There will be also needed some buttons in order to select the options and to show the state of the communications.

```

1 BUTTON *pButton; // Create the pointer to our new object
2
3 // Create the Button
4 // pointer = BtnCreate(ID , LEFT, TOP, RIGHT, BOTTOM, Initial state ,
   Bitmap, Text, Scheme);
5 pButton = BtnCreate(1, 20, 64, 50, 118, 0, BTN_DRAW, (void *) &
   BUTTON_LOGO, NULL, pButton);
6
7 BtnDraw(pButton); // Draw the object on the screen

```

Like we see, the functions are very similar and simple.

4.3.4 Porting the software

Once the software is developed in the evaluation board, we want it to run in our board. The previous step helped us in developing the software in a high level of abstraction. But the low level, the hardware specific level, has to be changed.

This means that the MCAL layer has to be configured in order, for example, to change the function of some pins of the microcontroller or select an oscillator and change its derived clock modifying the PLL multiplying factor.

As well, a driver for our LCD display controller shall be used. In this case, the display uses an UC1611 that is not supported by Microchip, so we have to develop our own driver. However, Microchip offers a driver for the UC1610 controller that can be probably modified to get our goals.

Chapter 5

Results

5.1 System

5.1.1 System requirements

In every step of the design process, we obtain some information, we use it to create our design and a report is generated as a result. In this case, the document of the system requirements step is a list of 39 requirements presented in Annex 1.

This document was later used for the development of the system architecture and the hardware and software requirements. This list of requirements define our product at system level.

5.1.2 System architecture

The system architecture was developed from the system requirements. It results in two schemes, the first one defines the relationship between our system and its environment. The second one defines the internal blocks of our system.

As Enterprise Architect allows to generate reports of the designs, a template was made in order to output a document for the system architecture. This document is presented in Annex 2.

In this case, this report defines the blocks and their relations at system level. This blocks are needed in order to comply the system requirements.

5.1.3 Concept art

Using the system requirements, a concept art was developed in order to clarify the aims of the project and have a more visual approach. This is useful for assuring that the customer and the designer are having the

same idea on what has to be accomplished. The result of the concept art design is shown in the following figure.

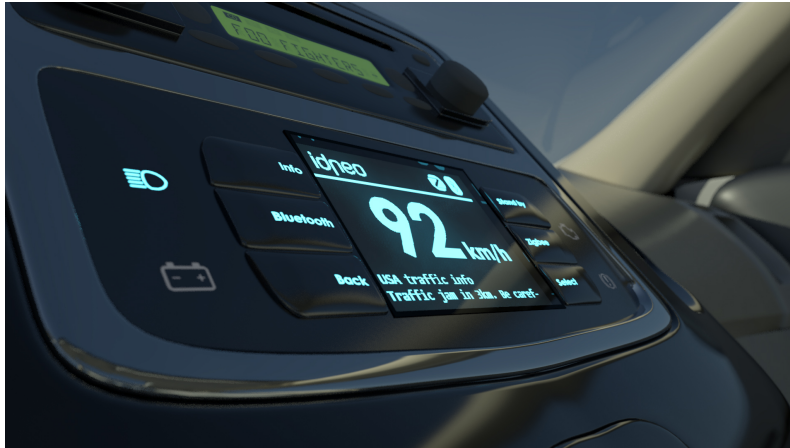


Figure 5.1: Final release of the concept art.

5.2 Hardware

5.2.1 Hardware requirements

This requirements were created from and satisfies every system requirements. The result of the hardware requirements step is a list of 18 requirements presented in Annex 3 section 2. With this information, the hardware designer can be able to design the hardware schematic and the layout. For example, there is specified the type of communication interface of the display, the CAN transceiver that shall be used or the voltage range in which the ECU shall operate.

5.2.2 Hardware schematic

The schematic designed in Altium was exported into a Smart PDF file and is attached in Annex 4.

The different variations that were presented during the design process are now represented in this document. The variation exported is the one that use the diode as a rectifier and the LEDs drivers from Infineon. The components that do not correspond with this variations are marked with a red cross.

As well we have to notice that some indications for the layout designer are indicated indirectly in the schematic. For example, a dedicated ground is defined in the microcontroller oscillator and connected in a short circuit to the main ground. This tells the designer that has to draw a special ground pattern that isolates the oscillator from the rest of the circuit.

5.2.3 Layout

With the layout guidelines, a render of the desired result was developed. In this representation, that we can see in the next figure, we can observe the components placed in the PCB, except for the display that is out of its socket in order to see the microcontroller.

We can see also the paths of a preliminary layout and 4 holes for the screws.

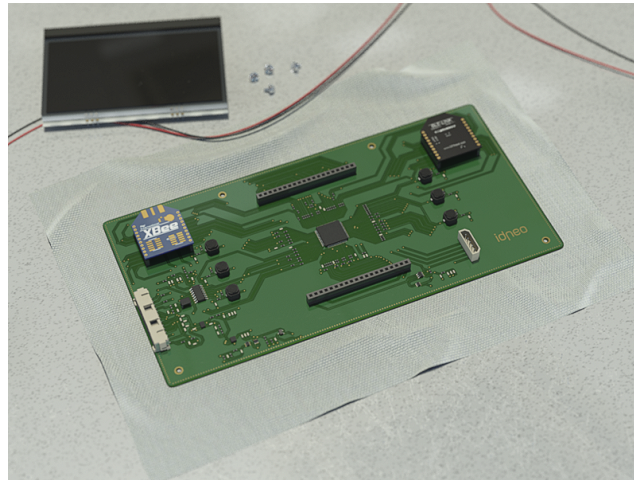


Figure 5.2: Representation of the prototype.

5.3 Software

5.3.1 Software requirements

The result of the software requirements step is a list of 10 requirements presented in Annex 3 section 3. There are included details, such as a state machine diagram, the compiler that should be used or the MISRA-C compliance.

5.3.2 Software development

The output of the software development is an Intel HEX file (a file format for programming integrated circuits) that can be downloaded into the microcontroller.

A photograph of the evaluation board running the program is show in the next figure.

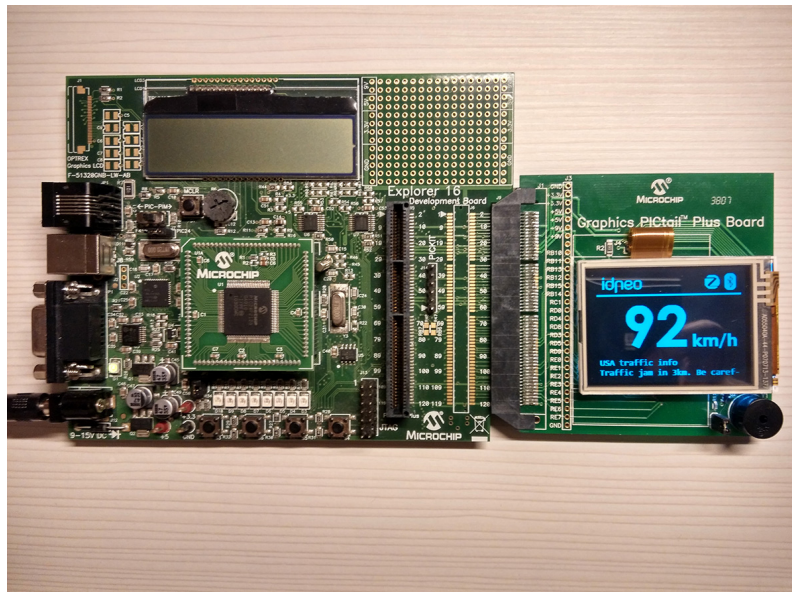


Figure 5.3: Evaluation board running the program.

Chapter 6

Conclusions

6.1 Objectives

This project has accomplished the most important objectives set. First, the steps of the design flow of the V design were followed. As well, my knowledge in embedded hardware design has been expanded a lot.

On the other hand, the project timing was too tight due to the lack of time. For this reason, the last steps defined in the Gantt, the hardware manufacturing and the software porting were not done. In order to compensate this, an action plan was done, doing the prototype representation and delaying the layout development.

Even so, I think that the academical part of the project was discussed, and this steps will be developed later in the company.

6.1.1 Innovative

It is innovative in the perspective and in the technology used. The perspective is different from the rest of the projects that we can see in automotive industry, because of its modularity and reusability. The project could be used to create new child projects with variations or the main board itself can be used changing the dockable parts, the LEDs, display and XBee modules. In any case, this is a solution not explored before.

The technology is innovative too. We are using the last technologies that are being applied now, like the CAN FD, or that are more alternative in this sector, like the XBee modules.

6.1.1.1 Hardware

One of the aims of the project was the modularity and the reusability. The sockets used can allow us to use different preexisting XBee modules or displays. There can be also developed new boards compatible with them and also could be studied if the cost reduction for using the same main board in different projects can overcome the cost of new boards for the display on different projects.

Either way, the interface hardware part can be modified without so much trouble. Buttons can be added and eliminated using the same debouncing circuits and connecting the output to the microcontroller. No much more complications with the LEDs and their drivers.

The changes for a new display depend on the communication interface with the microcontroller and its socket or connector. But basically, the interface will change the number of inputs and outputs to the microcontroller.

Other changes, like the microcontroller or supply blocks are more difficult to implement. Otherwise, the processor is powerful enough to handle a wide range of displays, from low resolution monochromes to QVGA TFT modules.

6.1.1.2 Software

The same happens with the software. If we change the interfaces, the changes are little modifications in the MCAL for the pin assignments and the use or development of another driver for the display.

In the case of changing the microcontroller, some things could be ported from the higher level of abstraction layers, but the lower ones would have to be completely redone, although the change was with a microcontroller of the same family. The change of the XBee modules will imply a change at application level.

Finally, the changes on the supply block components will not mean any changes on the microcontroller, but it can imply some new circuitry for the reset disable when the supply is ready.

6.2 Enhancements

During the design and development process, the following possible enhancements have been found.

- Start the HMI in stand by mode and change to active mode when the KL50 signal changes to high.
- Adapt the brightness of the LEDs by changing the current in function of the state of the car lights (reduce brightness at night).
- Better analysis of Zigbee vs other systems like Digimesh, that could be cheaper or more reliable.

References

1. S.L. IDNEO Technologies. Idneo official webpage. <http://http://www.idneo.com/>. Retrieved: January 10, 2016.
2. AUTOSAR Administration. Autosar official webpage. <http://www.autosar.org/>. Retrieved: January 12, 2016.
3. FUJITSU Semiconductor Magazine. Standard automotive software platform "autosar" and microcontroller driver mcal. *FIND*, 26(3):1 – 3, 2008.
4. MIRA Limited. *MISRA-C:2004 Guidelines for the use of the C language in critical systems*. 2004.
5. Mark Pitchford and Chris Tapp. Misra-c:2012 review. <http://electronicdesign.com/dev-tools/misra-c2012-plenty-good-reasons-change>. Retrieved: November 3, 2015.
6. MIRA Ltd. Misra webstore. http://www.misra.org.uk/shop/buy_now.php. Retrieved: November 3, 2015.
7. LDRA Ltd. Introduction to misra-c:2012. http://www.ldra.com/attachments/category/24/An_Introduction_to_MISRA_C-2012.pdf. Retrieved: November 3, 2015.
8. Red Lizard Software. Misra-c:2012 datasheet. <http://archive.redlizards.com/docs/misrac2012-datasheet.pdf>. Retrieved: November 3, 2015.
9. Antoni Lluís Mesquida Calafat. *Un Modelo para Facilitar la Integración de Estándares de Gestión de TI en Entornos Maduros*. PhD thesis, Universitat de les Illes Balears, Palma, May 2012.
10. Javier Garzas. Understanding iso 15504. <http://www.javiergarzas.com/2010/10/entender-iso-15504-1.html>. Retrieved: November 3, 2015.
11. Quality Management Center in the German Association of Automotive Industry (VDA QMC). Automotive spice. <http://www.automotivespice.com/>. Retrieved: November 3, 2015.
12. Quality Management Center in the German Association of Automotive Industry (VDA QMC). Automotive spice process reference model. http://www.automotivespice.com/fileadmin/software-download/Automotive_SPICE_PAM_30.pdf. Retrieved: November 3, 2015.
13. Auelectronics. K115, k130, k131, k150, k1r and ignition position. <http://www.auelectronics.com/forum/index.php?topic=281.0;wap2>. Retrieved: January 12, 2016.
14. Tamara Schmitz and Mike Wong. Choosing and using bypass capacitors. Application note, INTERSIL, October 2011.
15. NXP Semiconductors. Ah1014 application hints - standalone high speed can transceiver tja1042 / tja1043 / tja1048 / tja1051. http://www.nxp.com/documents/report/AH1014_v1_4_Application_Hints_TJA1042_43_48_51.pdf. Retrieved: November 31, 2015.
16. Jack G. Ganssle. *A Guide to Debouncing*. The Ganssle Group, 2008.
17. SparkFun Electronics Inc. Xbee buying guide. https://www.sparkfun.com/pages/xbee_guide. Retrieved: January 26, 2016.
18. Ashish Kumar. Top 10 emc design considerations. Technical report, Cypress Semiconductor Corp, 198 Champion Court, San José, February 2011.
19. Programming Research Ltd. Misra-c:2012 fact sheet. <http://www.programmingresearch.com/content/misc/PRQA-MISRAC2012-FactSheet.pdf>. Retrieved: November 3, 2015.

Annex 1

System requirements

HMI for industrial vehicles

Created: Eduard Ametller Navas 09/12/2015

Checked: David Pagès Cisa 10/12/2015

Contents

1	Introduction	5
2	General	5
3	Hardware	5
3.1	Connector	5
3.2	Power supply	5
3.3	Microcontroller	5
3.4	Communication	5
3.4.1	CAN	5
3.4.2	XBee	6
3.4.3	XBee Bluetooth	6
3.4.4	XBee Zigbee	6
3.5	Interface	6
3.5.1	General	6
3.5.2	Display	6
3.5.3	LED	6
3.5.4	Buttons	6
3.6	Reliability	7
4	Software	7
4.1	Functional	7
4.1.1	General	7
4.1.2	Default mode	8
4.1.3	Interaction mode	8
4.2	Reliability	8

1 Introduction

This document specifies the system requirements for the Human-Machine Interface for Industrial Vehicles, now on referred as HMI.

2 General

IDN_HMI-1 The HMI is an Electronic Control Unit (ECU) that will be placed in an industrial vehicle and shall act as an interface between the user and the vehicle information.

IDN_HMI-2 The HMI aims to be used in multiple applications, that is why it shall be modular and reusable.

IDN_HMI-3 In order to work as an interface, the HMI shall include 6 buttons to make possible to the user to select some options. On the other hand, there shall be a display, 4 LEDs and a Bluetooth module to give the user information. This information will be retrieved from CAN communication and Zigbee from other ECUs.

3 Hardware

3.1 Connector

IDN_HMI-4 A connector shall be placed in order to allow CAN communication, power supply and to connect the external LEDs.

3.2 Power supply

IDN_HMI-5 The HMI shall be fully operational with a power supply range between 9V and 32V. Outside this values the HMI shall go to off mode without any damage.

IDN_HMI-6 The ECU shall measure the voltage and current of the power supply.

3.3 Microcontroller

IDN_HMI-7 The HMI shall incorporate a microcontroller that shall be able to handle the display.

3.4 Communication

3.4.1 CAN

IDN_HMI-8 The HMI shall be able to communicate with other ECUs using a CAN BUS. It shall be able to retrieve information such as the vehicle speed, alerts/warnings status and car lights state.

IDN_HMI-9 The transmission procedure shall be CAN FD High Speed with a transmission rate of 500kbit/s.

IDN_HMI-10 The HMI shall be waked-up using CAN BUS.

3.4.2 XBee

IDN_HMI-11 The HMI shall communicate with other devices using two XBee modules. The first shall be a Bluetooth module, the second shall be a Zigbee module.

3.4.3 XBee Bluetooth

IDN_HMI-12 The XBee Bluetooth module shall allow the download of information to another compatible Bluetooth device in a range of 5 meters.

IDN_HMI-13 The XBee Bluetooth module shall be secure and only be paired with a PIN number that is shown in the display after a pairing request.

3.4.4 XBee Zigbee

IDN_HMI-14 The XBee Zigbee module shall allow the reception and routing of information in a range of more than 40 meters.

3.5 Interface

3.5.1 General

IDN_HMI-15 The vehicle information shall be retrieved using a monochrome display and 4 LEDs. The LEDs and display will be external to the hardware board.

3.5.2 Display

IDN_HMI-16 Monochrome display resolution shall be 240x128 or higher. The display shall be LCD with backlight.

IDN_HMI-17 On board connector shall be placed in order to connect the display bus with the rest of the hardware.

3.5.3 LED

IDN_HMI-18 The HMI hardware shall allow individual control of every LED.

IDN_HMI-19 One LED shall be blue and the other three shall be red.

3.5.4 Buttons

IDN_HMI-20 The user will select the information using 6 buttons integrated in the HMI board: Info, Stand by, Bluetooth, Zigbee, Back and Select.

3.6 Reliability

IDN_HMI-21 All the hardware shall be automotive, having an operation temperature between -40°C and 85°C.

IDN_HMI-22 The hardware shall be protected against ESD and shall be compliant with the EMC standards.

IDN_HMI-23 The inputs and outputs of the HMI shall be protected against short circuits, open circuits, direct connection to ground and to battery.

IDN_HMI-24 The hardware shall be protected against overvoltage and reverse polarity of the power supply.

4 Software

4.1 Functional

4.1.1 General

IDN_HMI-25 The HMI shall have three operational modes: - Default mode. - Stand by mode (shall be selectable for applications where the display is not always visible). - Interaction mode.

IDN_HMI-26 In Default mode, the HMI shows information in the display that is always turned on.

IDN_HMI-27 In Stand by mode, the display shall be turned off.

IDN_HMI-28 In Interaction mode, HMI display shall show a graphic interface based on 4 selectable items. The items shown by this interface shall be selectable using the buttons.

IDN_HMI-29 The Interaction mode shall be activated just after pushing a button. The displays shall always be turned on in this mode.

IDN_HMI-30 The Default mode shall be activated by default in the start and after 1 minute of no interaction (not pushing a button) or after pushing the Back button in the 4 item display screen.

IDN_HMI-31 If the Stand by function is enabled, the Stand by mode shall be activated after 1 minute in default mode.

IDN_HMI-32 The Bluetooth shall make this data accessible from an Android mobile app: - Log of car malfunctioning alerts. - Log of the speed in the last hour (graphical representation suggested). - Last 3 Zigbee communication messages log.

IDN_HMI-33 The Zigbee communication shall gather information sent by a certified infrastructure manager (such as road conditions and car accidents) and routed by other vehicles Zigbee modules.

IDN_HMI-34 The external LEDs shall show the following information: - On/Off state of the lights (blue LED). - Battery charge warning. - Engine warning. - Brake system warning.

4.1.2 Default mode

IDN_HMI-35 In Default mode, the HMI shall display this information: - Current speed - Alert of other ECUs malfunction. - Bluetooth status (enabled/disabled). - Zigbee status (enabled/disabled). - Road information gathered from Zigbee communication.

4.1.3 Interaction mode

IDN_HMI-36 In Interaction mode, HMI shall display this selectable items: - Information. - Enable/Disable Stand by function. - Enable/Disable Bluetooth. - Enable/Disable Zigbee.

IDN_HMI-37 After selecting the Information function, the HMI shall display: - Maximum registered speed in the last hour. - Log of the speed in the last hour (graphical representation suggested). - Alert status of the car. - Name of the Bluetooth paired device. - Last 3 Zigbee messages log.

4.2 Reliability

IDN_HMI-38 HMI software shall follow the rules and methodologies defined in the MISRA-C 2012 standard.

IDN_HMI-39 Memory occupancy should not exceed 80% of maximum memory available. This memory shall comprehend at least 128KB of FLASH.

Annex 2

System Architecture

HMI for industrial vehicles

Table of Contents

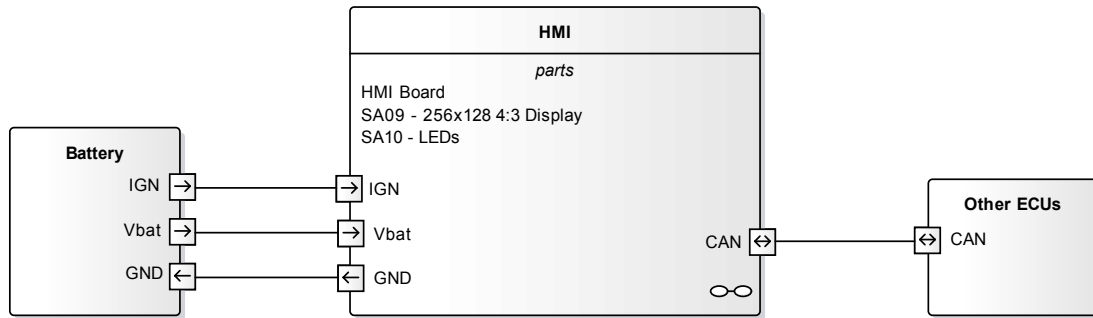
System diagram	2
HMI diagram	2

System diagram

SysML Block Definition diagram in package 'HMI'

System
Version 1.1

FV4EAN0 created on 26/11/2015. Last modified 13/01/2016



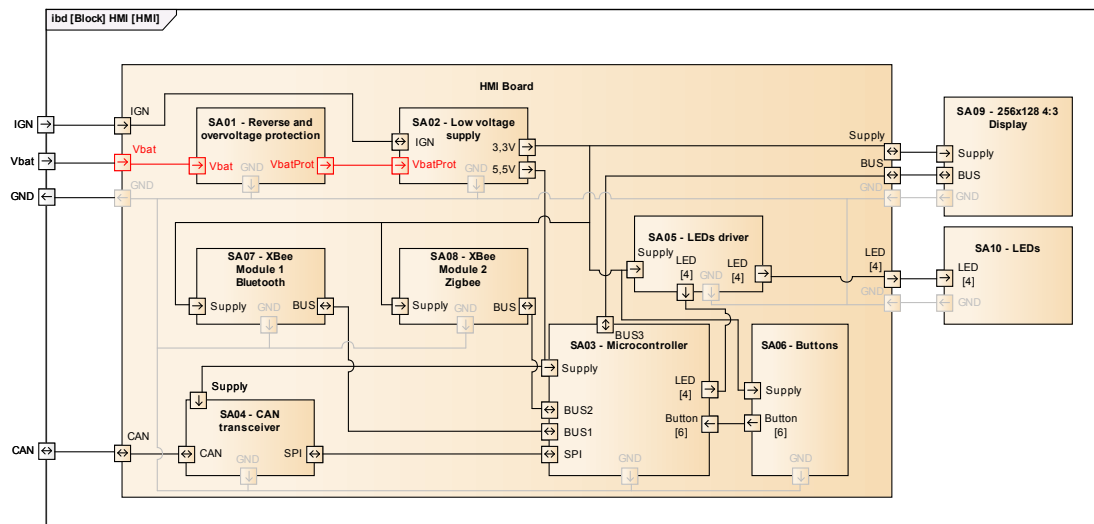
HMI diagram

SysML Internal Block diagram in package 'HMI'

HMI

Version 1.1

FV4EAN0 created on 26/11/2015. Last modified 13/01/2016



Annex 3

Hardware and software requirements and traceability

HMI for industrial vehicles

Created: Eduard Ametller Navas 11/12/2015

Checked: David Pagès Cisa 14/12/2015

Contents

1	Introduction	5
2	Hardware	5
2.1	Connector	5
2.2	Power supply	5
2.3	Microcontroller	5
2.4	Communication	5
2.4.1	CAN	5
2.4.2	XBee	5
2.4.3	XBee Bluetooth	5
2.4.4	XBee Zigbee	6
2.5	Interface	6
2.5.1	Display	6
2.5.2	LED	6
2.5.3	Buttons	6
2.6	Reliability	6
3	Software	7
3.1	Functional	7
3.1.1	General	7
3.1.2	Microcontroller	7
3.2	Reliability	8
4	Traceability	8

1 Introduction

This document specifies the hardware and software requirements for the Human-Machine Interface for Industrial Vehicles, now on referred as HMI.

2 Hardware

2.1 Connector

IDN_HMI_HW-1 A 14 pin connector shall be used for the CAN communications signals, power supply and external LEDs connections.

2.2 Power supply

IDN_HMI_HW-2 The HMI main board shall include the necessary switching regulators to obtain two low voltage supplies of 3,3V and 5V from a power supply range between 9V and 32V.

IDN_HMI_HW-3 A measure circuit shall be implemented for the power supply voltage and current measure.

2.3 Microcontroller

IDN_HMI_HW-4 The microcontroller used shall be the dsPIC33EP512MU810.

2.4 Communication

2.4.1 CAN

IDN_HMI_HW-5 The CAN communication shall be handled by a TJA1043 CAN transceiver. Component info:

- High Speed CAN FD.
- 2 Mbps data rate.
- Wake up source recognition.

2.4.2 XBee

IDN_HMI_HW-6 XBee shall be mounted into sockets in order to increase modularity.

2.4.3 XBee Bluetooth

IDN_HMI_HW-7 The HMI shall have a Xbee module 1 Bluetooth. The model TEL0073 is suggested. Component info:

- Input Voltage: +3,3 DC.
- Transmission distance: 15 to 20m indoor (30m in free space).
- Operating temperature: -40°C to +85°C.
- Throughhole
- Support Android and iOS applications, open source code, suitable for secondary development by the user.

2.4.4 XBee Zigbee

IDN_HMI_HW-8 The HMI shall have a Xbee module for Zigbee communication. Digi XBee® ZB S2 module is suggested. Component info:

- Range 120 m.
- Throughhole
- 250 Kbps

2.5 Interface

2.5.1 Display

IDN_HMI_HW-9 A display with this specifications shall be used:

- Monochrome LCD.
- At least 240x128 resolution.
- Backlight.
- SPI bus.

IDN_HMI_HW-10 A connector shall be placed in the board in order to connect the display bus.

2.5.2 LED

IDN_HMI_HW-11 LED Drivers shall be used to control individually every LED. The BCR401U model is suggested.

IDN_HMI_HW-12 The external LEDs (1 blue, 3 red) will be mounted in an external board.

2.5.3 Buttons

IDN_HMI_HW-13 The 6 buttons shall be mounted in the main board placed in 2 columns of 3 buttons at both sides of the display.

2.6 Reliability

IDN_HMI_HW-14 All the hardware shall be automotive, having an operation temperature between -40°C and 85°C.

IDN_HMI_HW-15 Additional circuitry shall be added in order to be protected against ESD and EMC problems.

IDN_HMI_HW-16 Additional circuitry shall be added in order to be protected against short circuits, open circuits, direct connection to ground and to battery.

IDN_HMI_HW-17 A diode shall be used as reverse polarity protection of the power supply.

IDN_HMI_HW-18 A TVS shall be used as overvoltage protection of the power supply.

3 Software

3.1 Functional

3.1.1 General

IDN_HMI_SW-1 An Android application for the Bluetooth communication with the HMI shall be developed using the open source code offered by the Bluetooth module manufacturer.

3.1.2 Microcontroller

IDN_HMI_SW-2 For the microcontroller code development, C30 or XC16 compilers shall be used.

IDN_HMI_SW-3 The HMI functional modes shall following this state machine diagram:

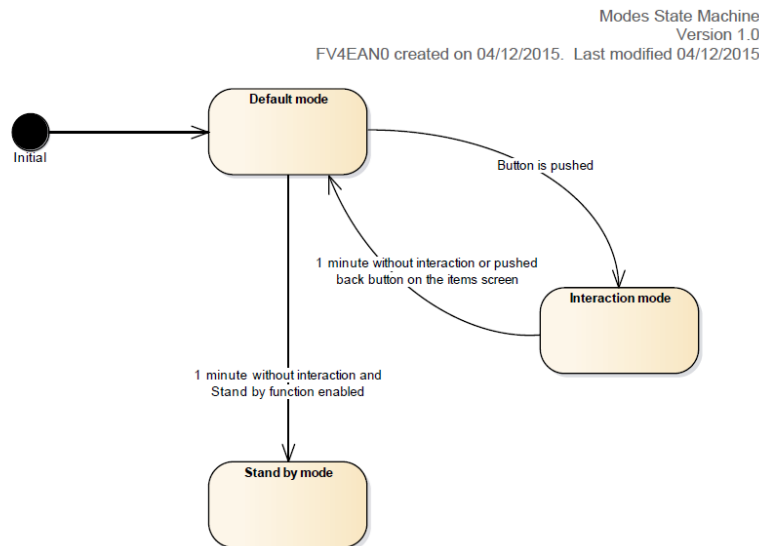


Figure 1: HMI software state machine diagram.

IDN_HMI_SW-4 The use of graphical libraries is suggested. In other case, some graphical functionalities shall be developed, including sprite management and monospaced fonts capabilities.

IDN_HMI_SW-5 The Zigbee communication shall work as a router and as an end device. Functionalities of autofind and connection of near devices shall be implemented.

IDN_HMI_SW-6 The display shall be able to be shutdown and woke up.

IDN_HMI_SW-9 The HMI software shall ask and listen for speed information and warnings messages from other ECUs using the CAN bus. This information shall be later shown using the display and LEDs.

IDN_HMI_SW-10 The HMI shall be able to enable or disable and know the on/off state of the Bluetooth and Zigbee modules. This shall be later shown in the display.

3.2 Reliability

IDN_HMI_SW-11 The code will be checked for MISRA-C compliance with Polyspace tool.

IDN_HMI_SW-12 Memory occupancy will be checked using the compiler options in order to check that it does not exceed 80% of the maximum memory available.

4 Traceability

The following lines show the list of system requirements with the hardware or software requirements that satisfies them.

IDN_HMI-2 IDN_HMI_HW-6

IDN_HMI-4 IDN_HMI_HW-1

IDN_HMI-5 IDN_HMI_HW-2

IDN_HMI-6 IDN_HMI_HW-3

IDN_HMI-7 IDN_HMI_HW-4

IDN_HMI-8 IDN_HMI_HW-5

IDN_HMI-9 IDN_HMI_HW-5

IDN_HMI-10 IDN_HMI_HW-5

IDN_HMI-11 IDN_HMI_HW-7, IDN_HMI_HW-8

IDN_HMI-12 IDN_HMI_HW-7

IDN_HMI-13 IDN_HMI_HW-7

IDN_HMI-14 IDN_HMI_HW-8

IDN_HMI-15 IDN_HMI_HW-9, IDN_HMI_HW-12

IDN_HMI-16 IDN_HMI_HW-9

IDN_HMI-17 IDN_HMI_HW-10

IDN_HMI-18 IDN_HMI_HW-11

IDN_HMI-19 IDN_HMI_HW-12

IDN_HMI-20 IDN_HMI_HW-13

IDN_HMI-21 IDN_HMI_HW-14

IDN_HMI-22 IDN_HMI_HW-15

IDN_HMI-23 IDN_HMI_HW-16

IDN_HMI-24 IDN_HMI_HW-17, IDN_HMI_HW-18

IDN_HMI-25 IDN_HMI_SW-3

IDN_HMI-26 IDN_HMI_SW-6, IDN_HMI_SW-10

IDN_HMI-27 IDN_HMI_SW-6

IDN_HMI-28 IDN_HMI_SW-6, IDN_HMI_SW-10

IDN_HMI-29 IDN_HMI_SW-3

IDN_HMI-30 IDN_HMI_SW-3

IDN_HMI-31 IDN_HMI_SW-3

IDN_HMI-32 IDN_HMI_SW-1

IDN_HMI-33 IDN_HMI_SW-5

IDN_HMI-34 IDN_HMI_SW-9

IDN_HMI-35 IDN_HMI_SW-4, IDN_HMI_SW-9, IDN_HMI_SW-10

IDN_HMI-36 IDN_HMI_SW-4, IDN_HMI_SW-10

IDN_HMI-37 IDN_HMI_SW-4, IDN_HMI_SW-10

IDN_HMI-38 IDN_HMI_SW-11

IDN_HMI-39 IDN_HMI_SW-12

Annex 4

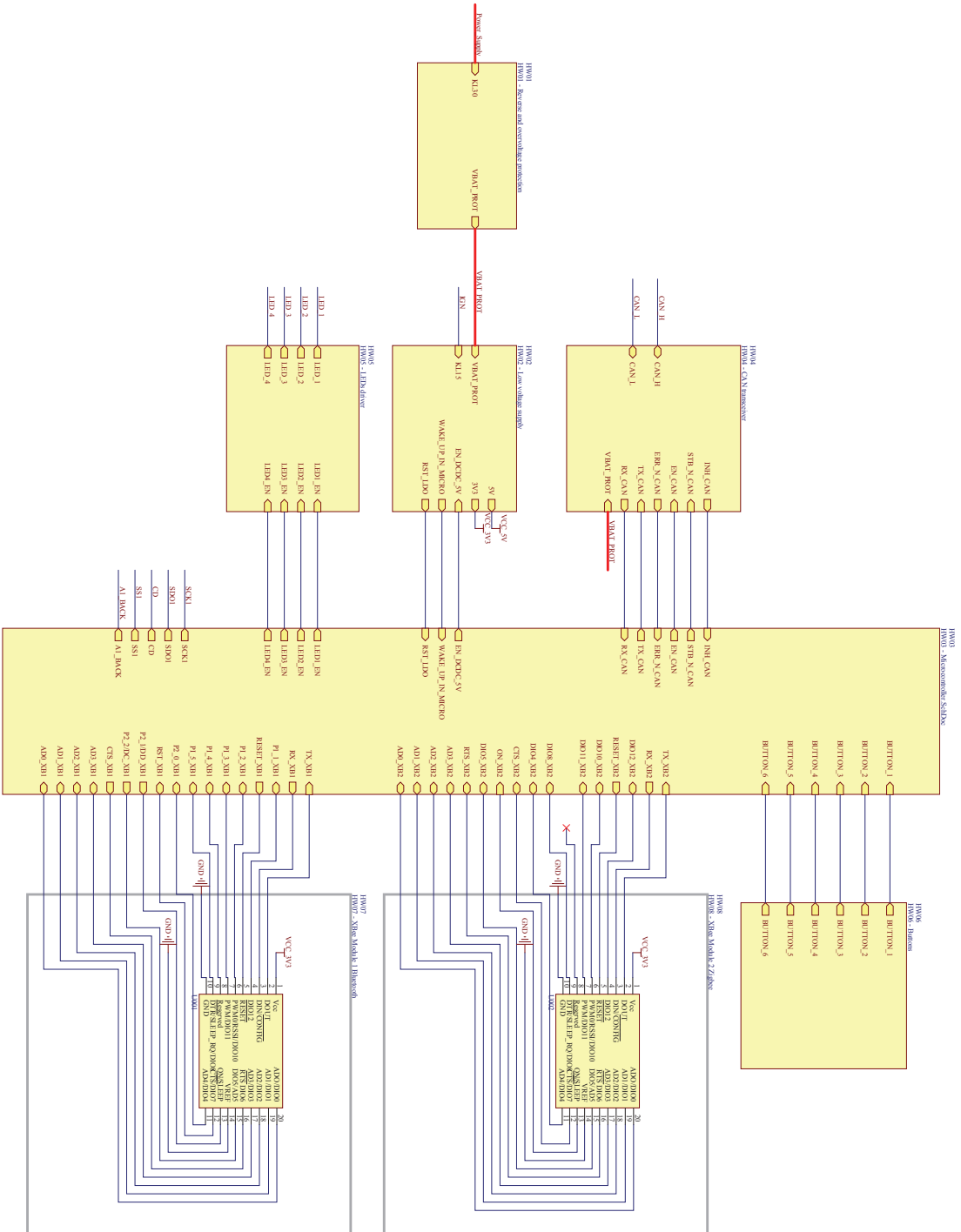
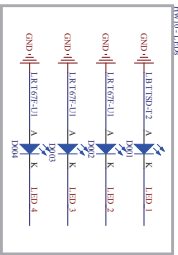
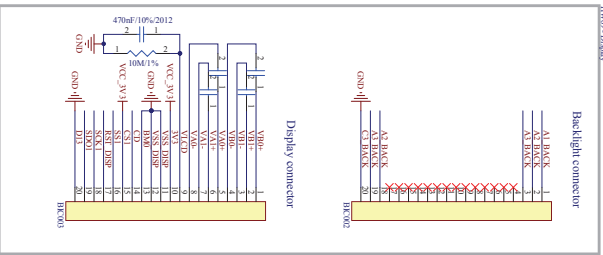
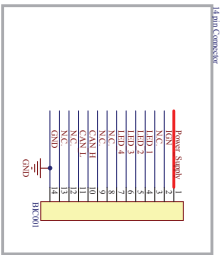
Hardware schematic

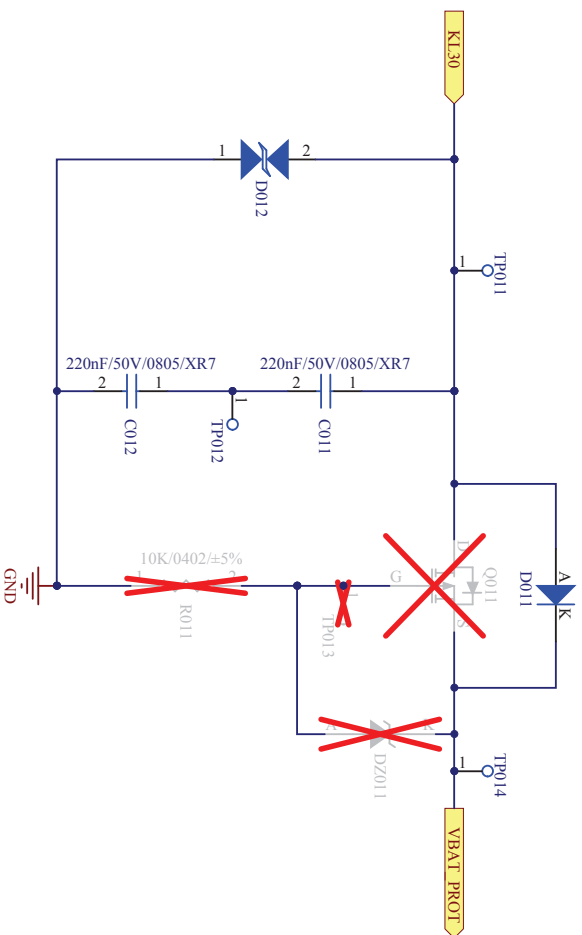
HMI for industrial vehicles

Created: Eduard Ametller Navas 10/12/2015

Modified: Eduard Ametller Navas 20/01/2016

Checked: David Pagès Cisa 22/01/2016



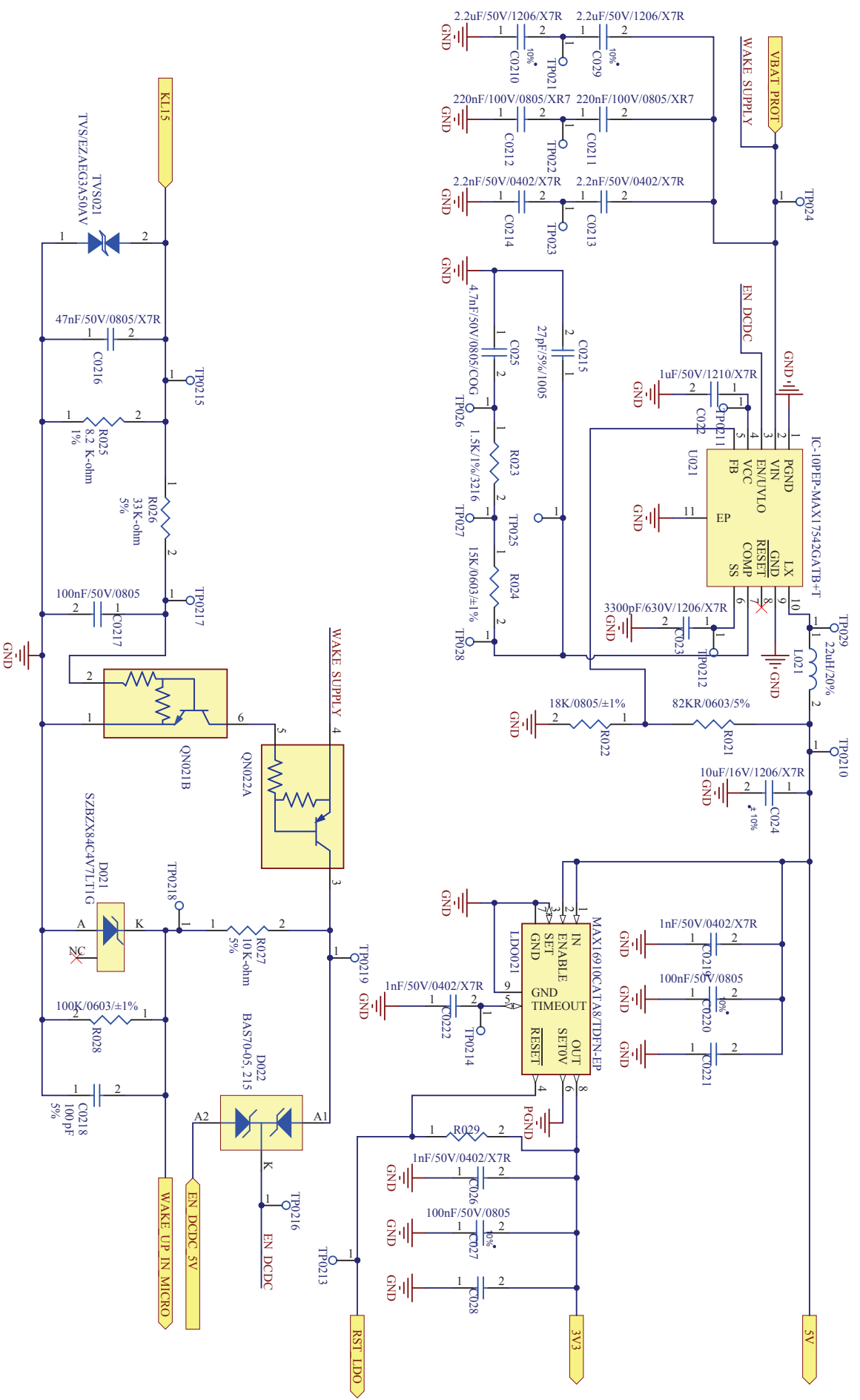


HMI for Industrial Vehicles

Title: **HMI for Industrial Vehicles**
 Size: **A4** | Block: **HW01 - Reverse and overvoltage...**
 Date: 20/01/2016 | Time: 10:03:45 | Sheet 2 of 7
 File: C:\Users\FV4E\AN0\Desktop\TPM\Design3 - Hardware\Schematic\SCH\HW01 - Reverse and overvoltage protection.SchDoc

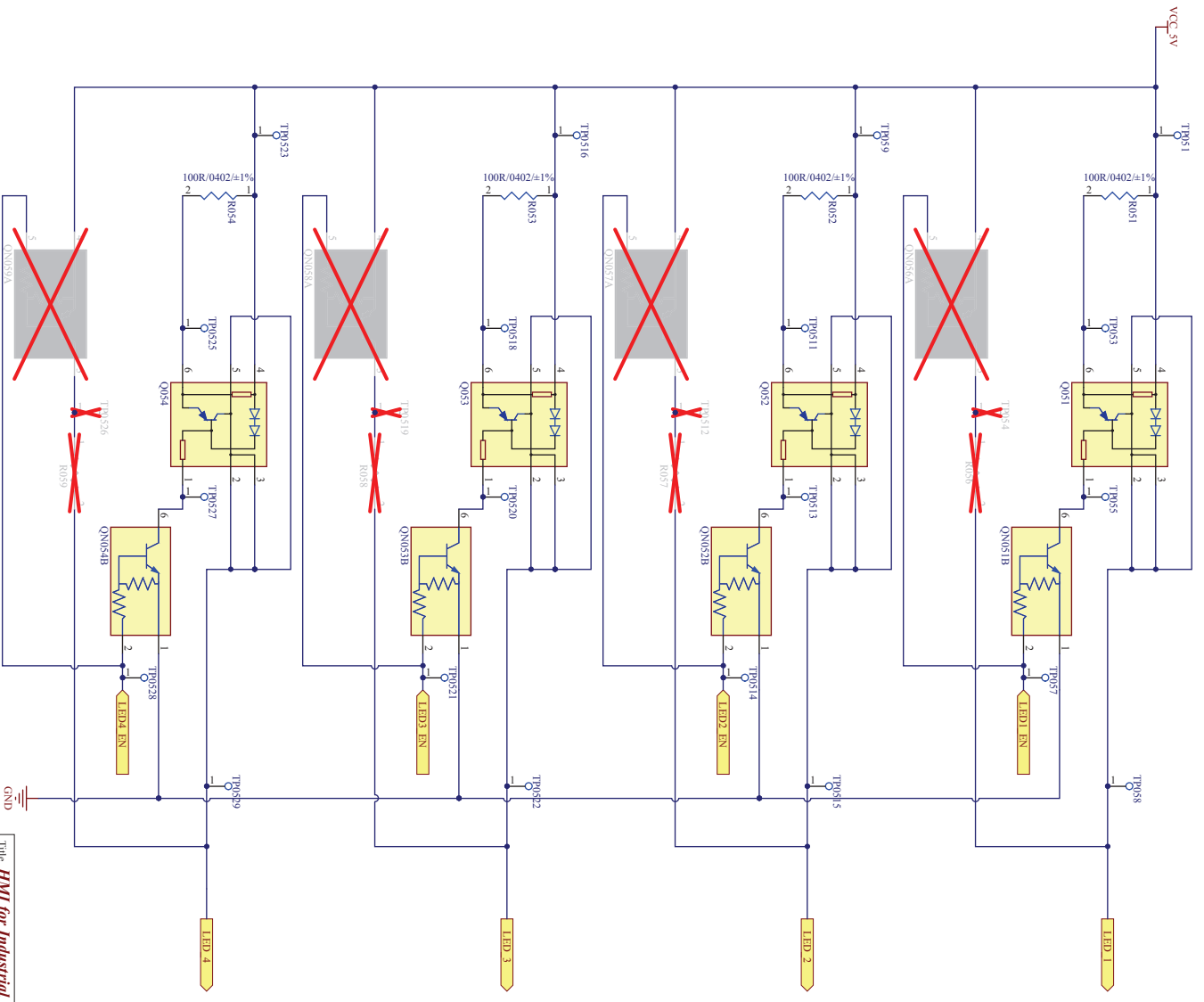
IDNEO Technologies, S.L.
 Poligono Industrial
 Can Mitjans S/N
 08232 Viladecavalls
 Barcelona





HMI for Industrial Vehicles

Title	HMI for Industrial Vehicles		
Size	A4	Block	HW02 - Low voltage supply
Date	20/01/2016	Time	10:03:46
File	C:\Users\FV4E\AN0\Desktop\TfM\Design\3 - Hardware\Schematic\SCH\HW02 - Low voltage supply.SchDoc		
	IDNEO Technologies, S.L.		Polígono Industrial
	Can Mitjans S/N		08232 Viladecavalls
	Barcelona		
	idneo		UAB



Title HMI for Industrial Vehicles		IDNED Technologiae, S.L.	
Spec A3	Block: HW05 - LEDs driver	Car Mirrors S/N	
Date: 20/01/2016	Time: 10:33:47	Rev: 0529 finalcanals	
File: C:\Users\F4E\N01\Desktop\TMI\Design\3 - Hardware\Schematics\SGT\HW05 - LEDs driver.SchDoc		Rev: 0529 finalcanals	



1

2

3

4

A

VCC_3V3

R062 82KR/0603/5%

R061 18K/0805/±1%

K061

C061 10%

GND

TP062

BUTTON 1

A

VCC_3V3

R064 82KR/0603/5%

R063 18K/0805/±1%

K062

C062 10%

GND

TP063

BUTTON 2

A

VCC_3V3

R066 82KR/0603/5%

R065 18K/0805/±1%

K063

C063 10%

GND

TP065

BUTTON 3

B

VCC_3V3

R068 82KR/0603/5%

R067 18K/0805/±1%

K064

C064 10%

GND

TP066

BUTTON 4

B

VCC_3V3

R070 82KR/0603/5%

R069 18K/0805/±1%

K065

C065 10%

GND

TP068

BUTTON 5

B

VCC_3V3

R072 82KR/0603/5%

R071 18K/0805/±1%

K066

C066 10%

GND

TP070

BUTTON 6

D

1

2

3

4

HMI for Industrial Vehicles

Title: HW06 - Buttons

Date: 20/01/2016 Time: 10:03:48 Sheet 7 of 7

File: C:\Users\FV4E\AN0\Desktop\TPM\Design3 - Hardware\Schematic\SCH\HW06 - Buttons_SchDoc

IDNEO Technologies, S.L.
Poligono Industrial
Can Mitjans s/n
08232 Viladecavalls
Barcelona



Annex 5

Meeting minutes

HMI for industrial vehicles

Kick off meeting

Date 3/11/2015 15:00

Place J. Aubert office

Attendees Eduard Ametller Navas, David Pagès Cisa, Jordi Aubert

Meeting purpose Define the project base elements and planning.

Assigned task	Who
Define the system requirements	E. Ametller

Milestone 1 - Review system requirements

Date 30/11/2015 9:00

Place Perafita/Idneo@DIREC

Attendees Eduard Ametller Navas, David Pagès Cisa

Meeting purpose Review of the system requirements.

Assigned task	Who
Define the hardware and software requirements	E. Ametller
Apply minor changes in system requirements	E. Ametller

Milestone 2 - Review HW/SW requirements

Date 10/12/2015 9:00

Place D. Pagès office

Attendees Eduard Ametller Navas, David Pagès Cisa, Jordi Aubert

Meeting purpose Review of the hardware and software requirements.

Assigned task	Who
Apply minor changes in HW/SW requirements	E. Ametller
Preliminary schematic design	E. Ametller
Obtain evaluation board	J. Aubert

Schematic review 1

Date 18/12/2015 17:00

Place D. Pagès office

Attendees Eduard Ametller Navas, David Pagès Cisa

Meeting purpose Review of the schematic.

Assigned task	Who
Select LCD, backlight and LEDs	E. Ametller
Change the low power supply input and output filters	E. Ametller
LDO RESET to microcontroller MCLR configuration	E. Ametller

Schematic review 2

Date 11/01/2016 15:00

Place D. Pagès office

Attendees Eduard Ametller Navas, David Pagès Cisa

Meeting purpose Second review of the schematic.

Assigned task	Who
Add test points	E. Ametller
DC/DC enable circuit	E. Ametller

Milestone 3 - Schematic freeze

Date 18/01/2016 9:00

Place D. Pagès office

Attendees Eduard Ametller Navas, David Pagès Cisa

Meeting purpose The schematic design is closed.

Assigned task	Who
Prototype render	E. Ametller
Send preliminary version of the thesis for review	E. Ametller

Thesis review

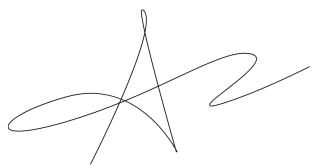
Date 29/01/2016 9:00

Place Moixero/Idneo@DIREC

Attendees Eduard Ametller Navas, David Pagès Cisa

Meeting purpose Review of the thesis.

Assigned task	Who
Apply minor changes	E. Ametller
Schematize conclusions	E. Ametller

A handwritten signature in black ink, consisting of a large, stylized letter 'A' followed by a horizontal line that curves upwards and then downwards, ending in a small loop.

Signed: Eduard Ametller Navas

Cerdanyola del Vallès, 29 of January of 2016

RESUM

El món de l'automoció és un sector clau dins de l'indústria, tant en el nostre país com en el àmbit global. En els últims anys, l'electrònica ha anat agafant un major protagonisme i s'ha realitzat un gran procés d'estandardització per garantir la seguretat dels usuaris. Això es tradueix en una marcada metodologia de treball que garanteix que el disseny dels productes destinats a aquest sector sigui satisfactori i compleixi els criteris mínims de qualitat i seguretat.

Aquest projecte tracta de reproduir tot el procés de disseny seguit en una empresa d'automoció, incloent la part de sistemes, hardware i software. Per a això, s'han estudiat diferents normatives d'útil o obligada aplicació i s'han estudiat i implementat els processos específics del disseny en V.

RESUMEN

El mundo de la automoción es un sector clave dentro de la industria, tanto de nuestro país como en el ámbito global. En los últimos años, la electrónica ha ido tomando mayor protagonismo, y se ha realizado un gran proceso de estandarización para garantizar la seguridad de los usuarios. Esto se traduce en una marcada metodología de trabajo que garantiza que el diseño de los productos destinados a este sector sea satisfactorio y cumpla los criterios mínimos de calidad y seguridad.

Este proyecto trata de reproducir todo el proceso de diseño seguido en una empresa de automoción, incluyendo la parte de sistemas, hardware y software. Para ello, se han estudiado diferentes normativas de útil u obligatoria aplicación y se han estudiado e implementado los procesos específicos del diseño en V.

ABSTRACT

The automotive world is a key sector inside the industry, both in our country and in the global scope. In the last years, the electronics has been taking prominence and a great standardization process has been done to ensure the safety of users. This results in a strong methodology that ensures that the design of the products for this sector is satisfactory and meets the minimum criteria of quality and safety.

This project aims to reproduce the entire design process followed in an automotive company, including the part of systems, hardware and software. To do this, the different standards that are helpful or mandatory were studied. As well, the specific processes of the V design were studied and implemented.