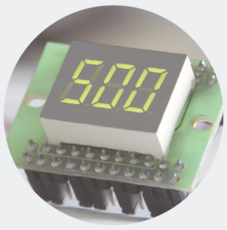
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO

MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES



Internet das Coisas em Veículos Partilhados

JOSÉ RAFAEL PERES SIMÕES SILVA
novembro de 2016

POLITÉCNICO
DO PORTO

IoT on Shared Vehicles

José Rafael Peres Simões Silva

Department of Electrical Engineering

Instituto Superior de Engenharia do Porto

2016

This report fulfills the requirements contained in TEDI course form, of the 2nd year of the Masters in Electrical and Computer Engineering, Automation and Systems

E-mail:

1100446@isep.ipp.pt

Advisor: Dr. Lino Figueiredo, lbf@isep.ipp.pt

Company: CEiiA

Supervisor: Eng. Gonçalo Salazar, goncalo.salazar@ceiia.com

isep Instituto Superior de Engenharia do Porto

Department of Electrical Engineering

Instituto Superior de Engenharia do Porto

November 18, 2016

Ackowledgements

I would like to express gratitude to my advisor Dr. Lino Figueiredo for his full support, expert guidance, understanding and his incredible patience throughout all project development.

I would also like to thank my supervisor Eng. Gonçalo Salazar, Eng. Pedro Silva, Eng. Vasco Rio and the rest of the team for all the help, support, guidance and integration in the company, as well as all the knowledge shared during the internship.

Thanks also go to my friends for the good times, for all the support, and especially friendship.

Finally, I would like to thank to my parents, my brother and all the family for their unconditional love and support during all my studies.

Abstract

Nowadays the need of people to have the power to control everything is increasing. Due to the technological evolution together with the Internet of things, this is already possible. In this context, the shared vehicles are a good example. With just one click people can use a vehicle from a vehicle sharing fleet anywhere, anytime.

During the realization of this project the uMDC was developed. It is a small device capable of managing and controlling different types of vehicles, with the main focus being the electric bicycles.

As a final conclusion of the project, the results obtained with the uMDC have proved very attractive. During its integration in the electric bicycles, the system was capable of controlling the bicycle's different components, as required for the first prototype.

Keywords

Internet of things, Technological evolution, Shared vehicles, uMDC, Electric bicycles.

This page intentionally left blank.

Resumo

Hoje em dia, a necessidade das pessoas terem controlo sobre tudo está a aumentar. Devido á evolução tecnológica juntamente com a Internet das coisas, isso já é possível. Neste contexto, os veículos partilhados são um bom exemplo disso. Com um simples clique, as pessoas podem usufruir de uma viatura de uma frota de veículos partilhados em qualquer lugar, a qualquer hora.

Durante a realização deste projeto, foi desenvolvido o uMDC. Um pequeno dispositivo capaz de gerir e controlar diferentes tipos de veículos, sendo o foco principal as bicicletas elétricas.

No final deste projeto, os resultados obtidos com o uMDC foram bastante satisfatórios. Durante a sua integração nas bicicletas elétricas, o sistema foi capaz de controlar diferentes componentes das mesmas, como requerido para primeiro protótipo.

Palavras-chave

Internet das coisas, Evolução tecnológica, Partilha de veículos, uMDC, Bicicletas elétricas.

This page intentionally left blank.

Contents

This page intentionally left blank.

List of Figures

This page intentionally left blank.

List of Tables

This page intentionally left blank.

Abbreviations

ABS	*Anti-lock Braking System*
AC	*Alternating Current*
API	*Application programming interface*
ASCII	*American Standard Code for Information Interchange*
BLE	*Bluetooth Low Energy*
BOM	*Bill of Materials*
CAN	*Controller Area Network*
CEiiA	*Centro para Excelência e Inovação na Industria Automóvel*
CoAP	*Constrained Application Protocol*
CoRE	*Constrained RESTful Environments*
COTS	*Commercial Of-The-Shelf*
CRM	*Customer Relationship Management*
DC	*Direct Current*
EASY	*Embedded Assist System*
EDA	*Electronic Design Automation*
EEPROM	*Electrically-Erasable Programmable Read-Only Memory*

ESR	*Equivalent Series Resistor*
FIFO	*First In, First Out*
GDB	*GNU Debugger*
GPIO	*General Purpose Input/Output*
GPRS	*General Packet Radio Service*
GPS	*Global Positioning System*
GSM	*Global System for Mobile Communications*
HAL	*Hardware Abstraction Layer*
HTTP	*Hypertext Transfer Protocol*
I^{2}C	*Inter-Integrated Circuit*
IC	*Integrated Circuit*
IBM	*International Business Machines*
ID	*Identifier*
IDE	*Integrated Development Environment*
IETF	*Internet Engineering Task Force*
I/O	*Input/Output*
IoT	*Internet of Things*
IP	*Internet Protocol*
ISEP	*Instituto Superior de Engenharia do Porto*
LCC	*Leadless Chip Carrier*
LDO	*Low Dropout*

LED	*Light-Emitting Diode*
LFCLK	*Low-Frequency Clock Source*
LIN	*Local Interconnect Network*
LLC	*Low Level Controller*
MCU	*Microcontroller*
MDC	*Mobility Device Controller*
MEEC	*Mestrado em Engenharia Eletrotécnica e de Computadores*
MOSFET	*Metal Oxide Semiconductor Field Effect Transistor*
MQTT	*Message Queue Telemetry Transport*
NAT	*Network Address Translation*
NDA	*Non Disclosure Agreement*
NFC	*Near Field Communication*
NMEA	*National Marine Electronics Association*
NRZ	*Non-Return To Zero*
NVM	*Non-Volatile Memory*
OMA	*Open Mobile Alliance*
OTA	*Over The Air*
PCB	*Printed Circuit Board*
PWM	*Pulse Width Modulation*
QoS	*Quality of Service*
RF	*Radio Frequency*

RGBA	*Red/Green/Blue/Amber*
ROM	*Read-only memory*
RTC	*Real Time Counter*
RTC	*Real Time Clock*
SD	*Secure Digital*
SDC	*Smart Device Cloud*
SDK	*Software Development Kit*
SIC	*Smart Integrator Cloud*
SIM	*Subscriber Identify Module*
SMS	*Short Message Service*
SPI	*Serial Peripheral Interface*
SPST	*Single Pole Single Throw*
SSC	*Smart Service Cloud*
SSL	*Secure Sockets Layer*
SSR	*Solid State Relay*
TCP	*Transmission Control Protocol*
TDMA	*Time Division Multiple Access*
TLS	*Transport Layer Security*
UART	*Universal Asynchronous Receiver/Transmitter*
UDP	*User Datagram Protocol*
uMDC	*Micro Mobility Device Controller*

UMTS	*Universal Mobile Telecommunications System*
UPnP	*Universal Plug and Play*
VAC	*Vehicle Addon Controller*
VPN	*Virtual Private Network*
XML	*Extensible Markup Language*
XMPP	*Extensible Messaging and Presence Protocol*

This page intentionally left blank.

Chapter 1

Introduction

Many years ago no one could ever imagine that such a thing as the Internet would be invented. As one of the greatest technological inventions in human history, the Internet has revolutionized the life and the way how people communicate and interact, marking the beginning of a new era of free information for all. With the passage of the time and the fast evolution of the Internet new versions and new ideas were emerge, one of them called the *Internet of Things* (IoT).

IoT is a technological revolution that represents the evolution of computing and aims to establish the interconnection between various smart objects through the Internet, collecting and analyzing data. It is a concept in development and embraces many aspects of life, from connected homes and cities to connected vehicles to devices that track individual's behavior.

The connected vehicles, as with other IoT applications, it's about more than just vehicles communicating with other technology over a network. It is about capturing data to improve efficiency within the sharing economy and enabling aware, autonomous and connected vehicles to transform data into useful information [1].

Linked to the cloud through wireless technologies, smart chips, onboard computers and mobile applications, connected vehicles are leading to new business models and disrupting old ones [2]. Shared vehicles are a good example of that.

In the past to share a vehicle people needed to be personally with the owner to fill the required forms, provide copies of personal documents and collect the vehicle key. At the end people had to return the vehicle at an exact location requiring the presence of someone to collect it. Nowadays this is no longer required.

Nowadays the IoT concept is being implemented almost everywhere. In shared vehicles, sensors and devices are constantly in communication with another nodes and the cloud allowing vehicles to be controlled and tracked remotely. Most of the new shared vehicles services have systems controlled by mobile applications that allows the user to open the vehicle, start the ignition (if available), turn on the lights, use the horn, and most important drive/ride wherever he/she wants. It also allows to collect information about the vehicles: battery status, fuel level, tires pressure and in case of motorcycle if the helmet is stored. At the end, the user just need to park in a valid parking zone, without requiring someone responsible.

IoT devices are becoming smarter and smarter. With a mobile device and cutting edge networks, people are just a click away from connecting with anything at any time.

1.1 Context

This project arose from the need of the *Centro para Excelência e Inovação na Industria Automóvel* (CEiiA) to develop a system capable of managing and controlling different types of vehicles. It is intended for the sharing and tracking of vehicle fleet and, at the same time, having extremely low power consumption, allowing *Global Positioning System* (GPS) location. collect information from the vehicle (velocity, odometer, etc...), interact with the vehicle (start and stop it, lock and unlock it, etc...). Along with all this requirements the solution should have also low size (approximately the size of a credit card).Currently, as far as known, there are no *Commercial Of-The-Shelf* (COTS) solutions that fulfill this requirements. This led to the development of the device being described on this thesis, the *Micro Mobility Device Controller* (uMDC).

The company already have a solution for sharing and tracking vehicles. However their current solution also doesn't fulfill those strict requirements demanded by vehicle with low capacity batteries.

1.2 Goals

The main goal of this internship is to develop a device to enable the sharing and tracking of electric bicycles.

At first it is required to do a market research to evaluate the existing hardware solutions and IoT communication protocols. It is also necessary to perform an analysis of the existing system, which is important during the uMDC architecture proposal. Although, this new system is not intended to be a substitute, it is important to become familiar with the existing hardware and the protocols used by the company.

To validate the chosen components it will be developed a generalist uMDC Development Board, which as no size and layer restrictions. During the production of this board, a prototype for *Secure Sockets Layer* (SSL) communications must be done.

After all the components are validated in the generalist uMDC Development Board and taking into account the current projects, a final solution will be developed. This final solution will have some restrictions and it could not include all the hardware present on development board.

1.3 Timeline

The schedule of the project developed in the internship was planned by the author according to table 1, available in Appendice A. In this table is possible to observe the time devoted to each stage. Since it was developed in an industrial environment, some changes may have occurred from the first project planning, however all the stages were accomplished.

1.4 Structure

This section presents the structure of this thesis, which is divided in the following eight chapters:

After this introductory chapter, the second chapter is intended to the presentation of the CEiiA and its vehicle sharing services.

The third chapter is dedicated to the state of the art, where a market research is presented, with the purpose of doing an analysis of similar devices available in the market. A study of the existing communication protocols for IoT and vehicle communication protocols that may be useful during the development of the project is also shown.

In the fourth chapter, the requirements analysis is performed, which is responsible for explaining the existing system in the company and analyze its main lacks, also presenting the new requirements.

In the fifth chapter is explained the system global architecture, from the hardware to software.

Then, the sixth chapter describes all the steps performed during the development of the system.

In chapter seven some results are presented, including the tests performed and the operation of the first prototype for electric bicycles.

The eighth chapter is intended to show conclusions of all work performed and future work required for the prototype to become a product.

The end of this thesis is reserved for the appendices which contains comparative tables of different components, excerpts of the manufacturer's datasheets and also the uMDC schematic and the *Printed Circuit Board* (PCB) design.

This page intentionally left blank.

Chapter 2

CEiiA

CEiiA is a center of engineering and product development that designs, implements and operates innovative products and systems alongside their partners in aeronautics, mobility, naval/offshore and automotive industries. It was established in December 1999 but it only started its activity in March 2000. The headquarters of the company are shown in figure 2.1 and it is located in Matosinhos, with additional offices in Lisbon and Évora.

Figure 2.1: Ceiia Matosinhos [3].

The organization operates as a bridge between the generation of knowledge at universities and the application to new products, processes and technologies, thereby contributing to a more innovative and a higher value-added industry, geared towards a sustainable economic development [4].

In the mobility area, CEiiA has developed MOBI.ME to connect vehicles and infrastructures, to integrate different information systems and to promote sustainability, offering a complete answer to the user's needs, operators and city authorities [5].

MOBI.ME is CEiiA's intelligent platform, designed for fully integrated, user-centric, management of mobility and energy services in smart cities. It connects charging management with shared mobility, parking and other services, via smart and agnostic integration with operators and any vehicle/smart device, based on open protocols. Integration with transportation and energy systems allows cross-management with public transport and home/fleet intelligent energy management, either through CEiiA's advanced devices or by connecting it to any device from any manufacturer.

Designed for future cities, MOBI.ME established an actual ecosystem where a solid transaction management platform, enables users access to all services from different operators via single authentication mechanism, while choosing the preferred payment and billing method. Operators are provided with full operational support for quick service rollout and cities are enabled with real time monitoring and planning, a powerful tool for policy design and implementation [6].

CEiiA provides integrated mobility solutions, comprising the smart management system, the equipment, and full operational support. Figure 2.2 illustrates MOBI.ME platform architecture which is divided into three layers.

The upper layer presents the Smart User Cloud and it is directed for end-users and operations teams/individuals and allows access to the MOBI.ME universe, including account management, historical information, mobile-based devices authentication and payment.

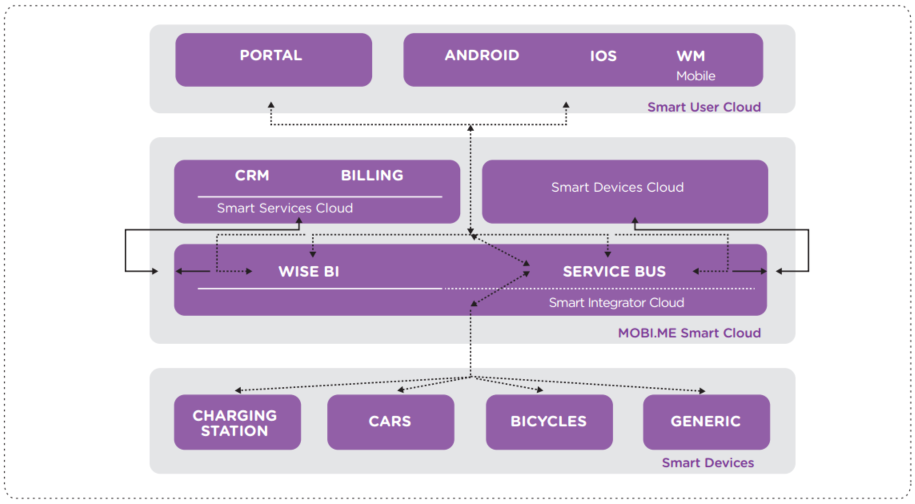


Figure 2.2: MOBI.ME architecture [6].

The middle layer displays the MOBI.ME Smart Cloud including the *Smart Service Cloud* (SSC), *Smart Device Cloud* (SDC) and *Smart Integrator Cloud* (SIC):

- SSC - Aggregation of all information and generation of process related to business and service configuration and management, including *Customer Relationship Management* (CRM), billing, invoicing and payment [6].
- SDC - IoT engine providing remote real-time monitoring and control of smart mobility devices [6].
- SIC - Technical and business reporting. Indicators for operators and cities, building on information generated by the MOBI.ME platform and its smart connected devices [6].

The final layer acts as the Smart Devices layer including charging stations, cars, bicycles, scooters, bus, etc.

At this time CEiiA integrates a wide diversity of projects being one of them the bike-sharing[1]. As most of the CEiiA's projects, this is being developed from scratch, which includes the development of all electronics (hardware and software), the monitoring software and control of the vehicles.

[1]http://www.jornaldenegocios.pt/economia/autarquias/detalhe/quem_mora_em_cascais_pode_estacionar_andar_de_bicicleta_e_autocarro_por_20_euros.html

Chapter 3

State of the Art

This chapter presents a market research to find systems and equipment that may have similar characteristics to the project in question. A study of some communication technologies that may be useful during the development of the project has also been carried out.

Based on this market research, on the communication technologies found and the existing equipment, a description of some key issues for an easy understanding of the choices made during the project development is also shown.

3.1 Vehicles Sharing Services

Due to the fast evolution of technology and the needs of our society, developing new systems and new services has become a requirement for business.

More and more companies of shared vehicles have been improving their systems, simplifying the way, not only of how the rent is made, but also how the vehicles are managed.

Nowadays there are several companies that use IoT for sharing vehicles, many of them are 100% Portuguese. Further ahead it is shown some of the sharing vehicle services and its general mode of operation.

3.1.1 Mobiag

Mobiag is the manager of carsharing system that results from the integration of multiple operators, guaranteeing its continuous integrity, reliability and service levels, as well as counterparty compensation and dispute resolution mechanisms. It promotes intelligent solutions for urban mobility, based on the integration of car-sharing and public transport systems, the benefit of its customers, partners and users, improving the quality of life cities.

Thanks to their own technological platform, called MobiCS, it is possible to manage all car-sharing operations.

The platform supports integration between multiple operators and their costumers through the seamless implementation of the roaming concept between operators in the system.

This service provides a solution to manage the entire business:

- onBoard technology: Control, monitor and manage vehicles with a control box;
- Modular and Flexible Backoffice: Provides a backoffice so that customers can implement operations and define their business, according to their business model and not constrained by technology;
- User Interfaces: Mobiag offers a set of tools (website, mobile Apps and *Application programming interface* (API)) so that customers can define and control the interaction and communication with the clients.

3.1.2 BikeEmotion

BikeEmotion is a Portuguese company founded in 2011 due to several years of joint research and development efforts of three companies and one university. The main goal of this project was to provide an open bike sharing solution with easy locking/unlocking and tracking methods, allowing users to benefit from increased flexibility [7].

This system was developed in the last four years, and it is now being integrated into Bewegen electric bicycle, allowing real-time tracking. Figure 3.1 shows one of the prototypes of this system, representing a few components as buttons, connectors, *Light-Emitting Diode* (LED)s, *Global System for Mobile Communications* (GSM) module and probably a GPS module. Thanks

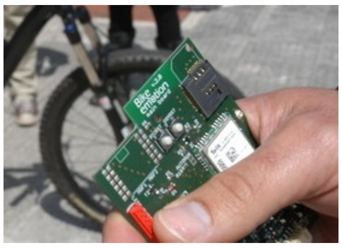


Figure 3.1: BikeEmotion tracking system [8].

to the advanced technology, it is possible unlock the bike with a *Near Field Communication* (NFC) card, a mobile application or with a simple *Short Message Service* (SMS). Due to the GPS system, each bike can be located at any time and, because of its small size, it allows users to adjust the device to any bicycle, and the system to any city. Each bicycle will alert the system and sound alarms in case of damage, theft or inappropriate use. The bicycle can only be returned when locked in an authorized area. Another interesting thing about this system, is the ability to provide useful and targeted information, suggesting interesting nearby places and routes through a context-aware user interface.

3.1.3 Bewegen

Founded by dedicated promoter of bike-sharing and urban mobility Alain Ayotte, Bewegen delivers a flexible bike-sharing as it is shown in figure 3.2 [9].

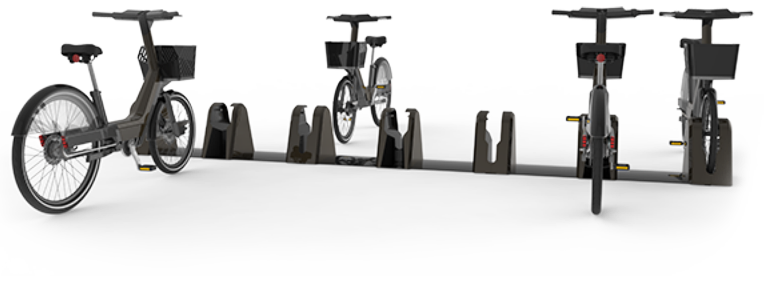
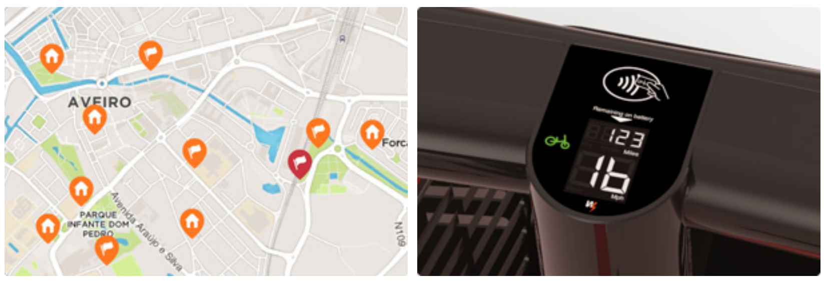


Figure 3.2: Bewegen electric bikes and dock [10].

The next generation electric assisted bicycles, combined with cutting edge technology, offers the ultimate shared transportation solution.

Bewegen's technology is opening up bike sharing to new markets and changing urban transportation for better, throughout the world [11].

Bewegen implements the same system as the BikeEmotion, allowing live GPS location, for individual bike route tracking/reporting, figure 3.3 a).



(a) (b)

Figure 3.3: a) Live GPS [11]. b) Interactive screen [11].

The bike also has an interactive screen where it's possible to view the status, the battery level and the ride information, as shown in figure 3.3 b). The NFC card reader is placed above the screen, allowing to unlock the bike.

To take advantage of the Bewegen system, users must:

- become a member by registering online to receive a pass or request an occasional card at the kiosk [11];
- choose the bike they like and simply swipe the pass/card directly on the bike's screen and remove it from the dock. It is also possible to unlock the bike through the mobile application;
- users can stop a run if needed and use the auxiliary lock, to ensure that nobody else rents the bike. To resume their ride, users just need to swipe their pass/card on the screen [11];
- return the bike at any station to end the rental. If the station is complete, they can request more time at the kiosk or secure the bike with the auxiliary lock at the designed "virtual station" area to end the rental [11].

3.1.4 Connected Cycle

The Connected Cycle Solution is a solution dedicated for connecting bikes and eBikes with smartphones. It provides protection capabilities and live tracking thanks to a GPS and GSM module located inside the bike [12].

As it has an accelerometer which verifies the movement of the bike and every time someone touches it, a notification is sent to the user's smartphone. The system automatically records the speed, route, incline and calories burnt on each bike trip. These statistics are sent to the cloud via GSM and made available to users through the Connected Cycle application available on smartphones.

The Connected Cycle solution is illustrated in figure 3.4 and it was designed for bikes with low energy consumption. It incorporates *Bluetooth Low Energy* (BLE) and also a backup battery that recharges itself thought out the bike electrical circuit. This system is compatible with any kind of bicycles (electric and non electric).

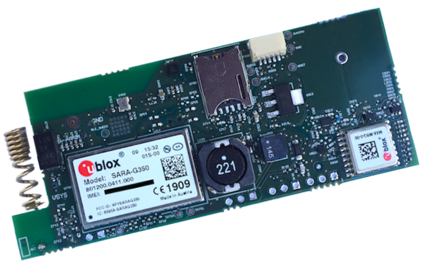


Figure 3.4: Connected Cycle board [12] .

3.1.5 GoBike

In 2008, Eduard Sentis saw a need to develop a shared bicycle which makes it easy to ride in hilly cities, while at the same time creating a beautifully designed piece of urban furniture that will fit elegantly into a city. After the production of the bicycles, 80 were installed and put into operation in the town of Granollers, outside Barcelona. This project was carried out for 3 years. After that, Eduard along with two other partners, teamed up with Software Expert Kasper Larsen to found the company Gobike and enter a bid for the tender. Since then a team of highly skilled experts has been added to develop the concept and the solution further. On June 1st 2012 Gobike was named the winner of the Copenhagen tender and the contract was signed in December 2012.

At the beginning of April 2014, the system in Copenhagen was opened to public use, providing citizens with 200 electric bicycles. Because of their ergonomic shape aluminium frame, the modularity and the internal cable routing, the bicycles are practical and stylish, figure 3.5.

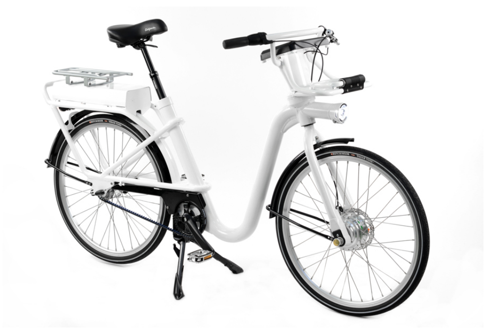


Figure 3.5: Gobike electric bike [13].

As shown in figure 3.6 a) the bicycle integrates a tablet PC with built-in GPS, which is water resistant, and allows the user to create an account without the need of a smartphone, plots routes around town, checks travel booking and reports any problem. This system also detains NFC which simplifies not only the payment of services but also the use of the electric locking system, figure 3.6 b), allowing user to stop the bike safely even if there is no docking station nearby.

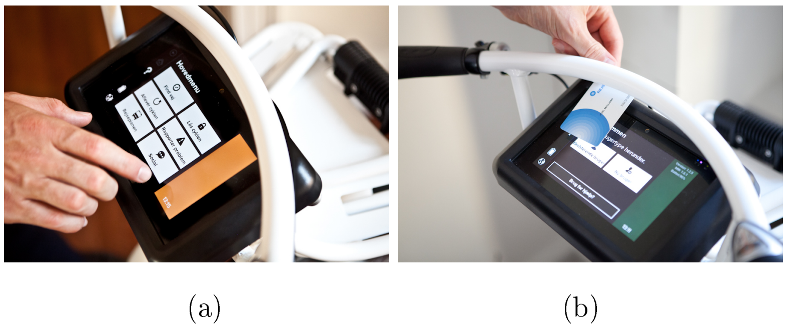


Figure 3.6: a) Integrated Tablet PC [14]. b) IDigital Lock [14].

Being an electric bike, it incorporates an electric motor that provides optional assistance at four different levels with a maximum speed of 22km/hour, and automatically switches off when the brakes are in use. This motor is powered by a battery of 37 V with 10.000 mA, which provides assistance for around 40 kilometers, depending on terrain, temperature and riding style [15]. Figure 3.7 illustrates the size and the location of the battery and the electric motor.

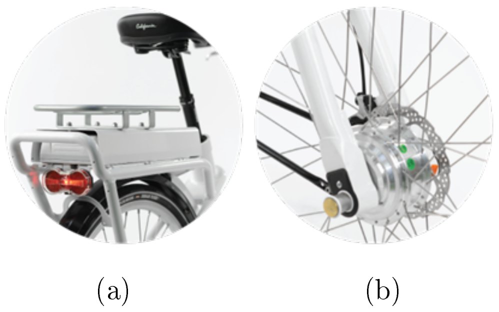
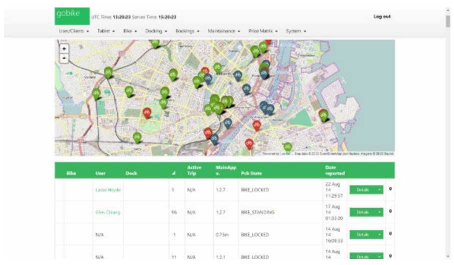


Figure 3.7: a) Battery [14]. b) Electric motor [14].

In figure 3.8 is shown the Administration Website that provides a real-time map overview that shows the location and status of all bikes in operation. This view can be used to quickly see the status of all bikes in the system, both docked bikes, and bikes currently on trips. Notification and reporting tools within the Administration Website give Operators tools to manage a Gobike system effectively [16].



Bike	User	Dock		Active Trip	MainApp v.	Pcb State	Date reported		
	Lasse Heyde		1	N/A	1.2.7	BIKE_LOCKED	22 Aug 14 11:29:57	Details	
	Glen Chiang		56	N/A	1.2.7	BIKE_STANDING	17 Aug 14 01:55:00	Details	
	N/A		-1	N/A	0.73m	BIKE_LOCKED	14 Aug 14 16:08:33	Details	
	N/A		11	N/A	1.3.1	BIKE_LOCKED	14 Aug 14	Details	

Figure 3.8: Gobike real-time map [16].

3.1.6 Inosat

Inosat was founded in January 2000 by a team of professionals in the telematics sector, Inosat is a company specialized in developing, production and marketing of products and services, in the sector of tracking vehicles, people, objects and advanced fleet management via GPS focusing on Mobility Intelligent[1].

Inosat provides two different types of solutions, a business solution, that allows the management of the fleet through GPS, and a particular solution. In figure 3.9 an illustration of the capabilities of the business solutions is shown.

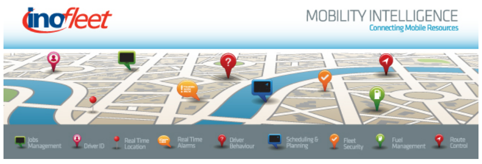


Figure 3.9: Fleet Management Systems [17].

Their technology is 100% portuguese, and the team consists of 30 technicians dedicated exclusively to research, development of hardware and software and costumer support.

The InoFrota BIZ is a vehicle location and fleet management based on GPS and GSM location technology system.

This system enables control of the vehicles, an optimization of the routes, fuel saving, which will increase the productivity and consequently improve the quality of service.

It was developed in order to allow companies to control the entire operation of

[1]Concept based on the GPS location service and fleet management, in real-time, allowing to manage information on mobile resources.

the fleet in real-time, like it is shown in figure 3.10, increasing the profitability and it consists of three main areas:

- Fleet Management in Real Time - The entire fleet in real-time and actions on the map;
- Reports - Management of historical fleet (available for 5 years) through a complete set of reports;
- Alarms - Real time alerts (in the application, by email or sms) of events checked in vehicles.

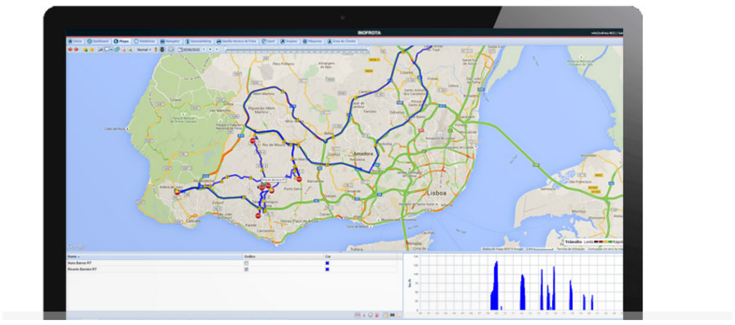


Figure 3.10: Fleet Management in Real Time [18].

Companies are struggling ever since with several problems that jeopardize their profitability and prevent their potential growth. To counter these problems, the InoFrota BIZ states it can ensure some benefits in its implementation such as:

- Reducing Fuel Costs;
- Increasing Productivity;
- Improving Security;
- Punctual Delivery;

- Efficiency in Service Management.

As previously spoken, Inosat is also directed to particular solutions, having already developed two different systems, the Car Locator and the My Locator.

The Car Locator is a GPS private tracking system used to locate and protect vehicles. It uses the GPS/GSM technologies and ensures the safety of the vehicle. This system will be able to receive alerts of abnormal situations, such as car battery cut, reporting situations of leaving the country , among others, and it can remotely immobilize the car whenever it is in danger.

The My Locator has the same operating principle as the Car Locator, also allowing the possibility to define security areas, which triggers an alert when exceeded.

3.2 IoT Protocols

The things around us are becoming smarter, intelligent, perfectly connected and interacting with each other. However, the real question is how do they communicate with each other and what language or protocol will all these devices use to communicate with the Internet.

When the Internet started to evolve, there was a strong desire among all the stake holders that all the computers connected to the Internet should be talking in the same language/protocol (*Transmission Control Protocol* (TCP), *Internet Protocol* (IP), *User Datagram Protocol* (UDP), etc.). Nonetheless the IoT world still lacks a widely accepted and standard protocol, apart from a few protocols that are being currently used.

In IoT, the interconnected devices are usually of short range low power wireless devices with their own predefined set of operations or purpose. Because of this, the protocols that are currently used on the Internet cannot be directly used in these devices. This brought in the need to create a new set of protocols which can be catered to the requirements of the IoT world. Some of the most widely used protocols in IoT are *Message Queue Telemetry Transport* (MQTT), *Constrained Application Protocol* (CoAP) and *Extensible Messaging and Presence Protocol* (XMPP). These protocols specifically target low power devices which have to preserve power so that they can operate for a long time. Compared with the Internet protocols where the pay load is heavy along with big headers and footers, in IoT protocols the payload is very small. So if Internet protocols are being used, the work will be more to reduce the size of the headers and footers for aiding fast communication between the devices and the servers [19].

The protocol selection problem is closely tied to the implementation of the protocol and the components that support the protocol are often essential in the final design. This makes the decision a very complex one. All aspects of deployment, operation, management, and security must be considered as part of the protocol selection including the implementation environment [20].

Taking into account these aspects, it was necessary to carry out a research on some protocols already implemented in IoT, as well as communication protocols with vehicles.

3.2.1 MQTT

MQTT is a publish-subscribe messaging protocol invented by IBM for satellite communications with oil field equipment. It had reliability and low power at its core and thus made good sense to be applied to IoT networks [21]. In this kind of protocols, publishers do not program the messages to be sent directly to specific subscribers. Published messages are characterized into classes instead, without knowing what kind of subscribers there may be. Once MQTT uses a publish-subscribe model it requires a central MQTT broker to manage and route messages among an MQTT network's nodes.

Nodes and broker need to have each other's IP address in order to communicate. However, nodes can publish information and subscribe to other node's published information without any knowledge of each other since everything goes through the central broker.

MQTT has a lightweight packet structure designed to preserve both memory usage and power. A connected device subscribes to a topic hosted on an MQTT broker. Every time another device or service publishes data to a topic, all of the devices subscribed to that topic will automatically get the updated information [23]. This allows the end nodes to operate independently from one another.

Figure 3.11 represents the MQTT architecture where it's possible to notice how the protocols works when a message is published in some topic.

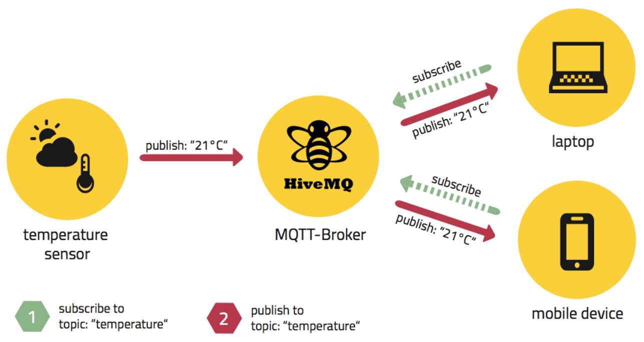


Figure 3.11: MQTT Publish/Subscribe Architecture [22].

A node can publish its information even if other nodes are not active, but when the node state changes, they will receive the information from the broker. This allows nodes to remain in sleepy states, even when other nodes are publishing messages directly to them.

If a node is processing a message, the operation can't be interrupted and the message is queued by the broker until the receiving node finishes the operation. This saves operating current and reduces repeated operations by avoiding interruptions of on-going operations or sleepy states.

MQTT uses unencrypted TCP and it is not secure but it can use *Transport Layer Security* (TLS)/SSL. TLS is a very secure method for encrypting traffic, but it is also a complete resource for lightweight clients. For networks where energy is a very high priority and security much less so, encrypting just the packet payload may suffice [23].

To guarantee a quality of service *Quality of Service* (QoS), the sender and receiver must have an agreement to confirm the delivery of a message. There are three QoS levels in MQTT:

- **At most once (0)** - This is commonly known as *Fire and Forget* and it is a simple transmit burst with no guarantee of message arrival (figure 3.12);

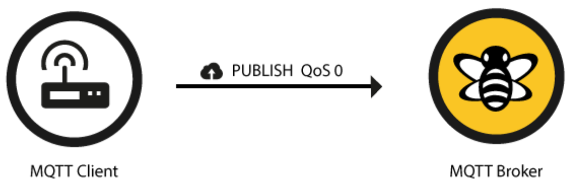


Figure 3.12: QoS 0 - At most once [24].

- **At least once (1)** - When using level 1 of QoS, it is guaranteed that a message will be delivered at least once to the receiver. The sender will store the message until it gets an acknowledgment in form of a PUBACK command message from the receiver (figure 3.13);

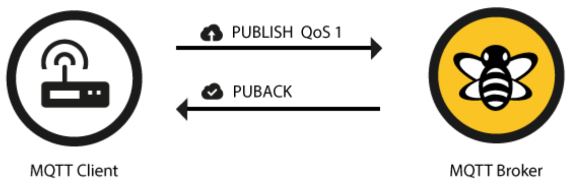


Figure 3.13: QoS 1 - At least once [24].

- **Exactly once (2)** - This level attempts to guarantee the message is received and decoded by the intended recipient. This is the most secure MQTT level of QoS. Its intended recipient gathers the announcement, decodes it and indicates that it is ready to receive the message. Once the recipient understands the message, it completes the transaction with an acknowledgment (figure 3.14).

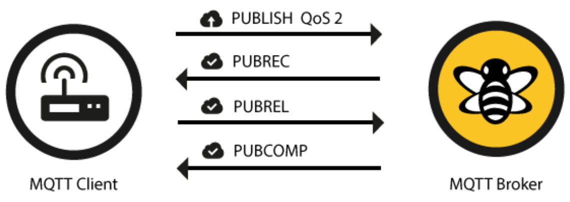


Figure 3.14: QoS 1 - Exactly once [24].

MQTT provides a *last will and testament* message that can be stored in the broker in case a node is unexpectedly disconnected from the network, allowing to retain the node's state and purpose, including the types of commands it published and it's subscriptions. If the node disappears, the broker notifies all subscribers and if the node returns, the broker notifies its prior state.

3.2.2 CoAP

The CoAP was created by an *Internet Engineering Task Force* (IETF) called *Constrained RESTful Environments* (CoRE). This protocol was developed to allow resource-constrained devices to communicate over the Internet using User Datagram Protocol (UDP) instead of Transmission Control Protocol (TCP). As a client/server protocol, provides a simple request/report interaction model with accommodations for multi-cast [23]. CoAP packets use bitfields to maximize memory efficiency, and they make extensive usage of mappings from strings to integers to keep the data packets small enough to transport and interpret on-device. As in MQTT, to guarantee the quality of service, CoAP can define four different types of messages (figure 3.15):

- *Confirmable* - This messages requests that the receiver sends back an acknowledge;

- *Non-Confirmable* - No acknowledge is needed;
- *Acknowledgment* - This confirms that the message was received;
- *Reset* - This confirms that a message was not properly received and it can't be processed.

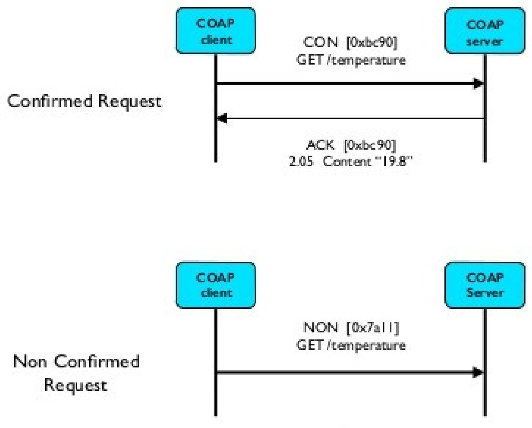


Figure 3.15: CoAP Quality of Service [25].

3.2.3 XMPP

XMPP is an open source based on Extensible Markup Language (*Extensible Markup Language* (XML)) and it's used for instant messaging, presence, multi-party chat, voice and video calls, collaboration, lightweight middleware, etc. XMPP was originally developed in the Jagger open source community to provide an open, decentralized alternative to the closed instant messaging services at that time [26].

In the IoT context, XMPP offers an easy way to address a device. This is convenient if data is going between distant, mostly unrelated points, just like the person-to-person case. It's easy for anyone to run their own XMPP server, allowing device manufacturers and API operators to create and manage their own network of devices. If security is needed, server can be isolated on a company Intranet behind secure authentication protocols using built-in Transport Layer Security (TLS) encryption.

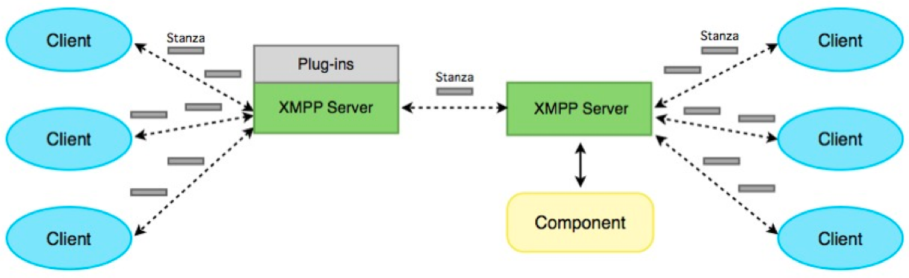


Figure 3.16: XMPP Architecture [27].

Comparison

CoAP, as explained before, runs on UDP and thus can work in extremely resource constrained environments. It is a good mechanism for local network communication, particularly when there is an ecosystem of other CoAP devices. Unfortunately, CoAP is a one-to-one protocol, so broadcast capabilities are not native to the protocol.

XMPP is not designed to be fast, in fact, most implementations use polling, or checking for updates only on demand. It doesn't supports end-to-end encryption which is a requirement in most of IoT devices. Another downside is the lack of Quality of Service (QoS), which can be a truthful deal-breaker depending on the application.

MQTT is a many-to-many communication protocol for passing messages between multiple clients through a central broker. This makes it a good option for remote/cloud communication, since the cloud server acts as the message broker between the IoT device and other applications or services. It offers publish/subscribe semantics (on the same socket) which makes it easier to program on the IoT device side. The downside with MQTT is that it uses TCP connections which requires an always-on connection and it will limit the amount of time the device can be put to sleep. Another disadvantage of MQTT is the lack of encryption in the base protocol. It was designed to be a lightweight protocol and incorporating encryption would add a significant amount of overhead to the connection.

3.3 Vehicle Communication Protocols

With the evolution of technology and the increase of electronics in vehicles, it has become imperative to develop communication protocols between the various subsystems, thereby reducing the amount of wiring and to further interaction and connection between devices from different manufacturers, and at the same time increasing the quality and reliability of communications [28].

Thus, the communication protocols currently used in the automotive field are the *Controller Area Network* (CAN), *Local Interconnect Network* (LIN) and FlexRay.

3.3.1 CAN

The CAN is a digital serial communication protocol that supports the control of distributed systems and real-time systems with high security and reliability [29].

In 1986, Robert Bosh developed the CAN protocol to be used in automotive industry, with the aim of simplifying the complex wire systems in vehicles with a controlling system composed of multiple microcontrollers for engine management, *Anti-lock Braking System* (ABS), suspension control, etc. This would allow the construction of more reliable, safer and more efficient cars. In figure 3.17 is present the notorious difference on the wiring quantity between a network with and without CAN.

Since its inception, the CAN protocol has increasing popularity in the automotive field and industrial automation. At the same time, other markets where the use of communication networks can bring great benefits, such as medical equipment or mobile equipment, are also implementing this protocol.

It is a serial communication in which all nodes on the CAN bus are attached to common connection (figure 3.17) using the same bitrate [30]. It is not an address based protocol, instead of that, the messages transmitted have a certain identifier which means that messages can not be transmitted directly from one node to another. Thus, when a messages is sent to the CAN bus, all nodes will receive it and then, thanks to the local filter provided by the CAN hardware, it is decide whether or not to process the message.

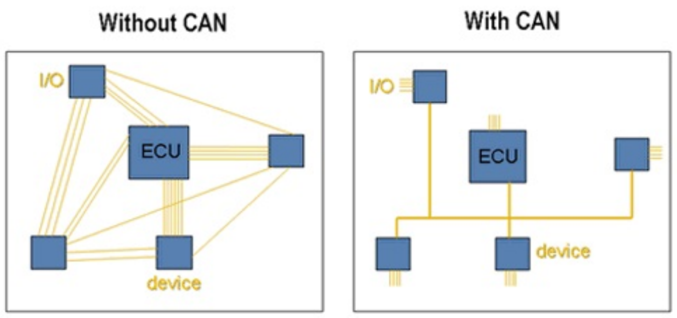


Figure 3.17: CAN networks significantly reduce wiring [31].

Each CAN message only contains a maximum of 8 bytes of payload. However, it is possible send a big amount of data using segmentation.

The bus uses *Non-Return To Zero* (NRZ) with bit-stuffing. The modules are connected to the bus in a wired-AND connection which means, if just one node is driving the bus to a logical "0", then the whole bus is in that state regardless of the number of nodes transmitting a logical "1" [32].

The CAN network may be configured to operate with two different types of messages, the standard format (CAN 2.0A) or the extended format (CAN 2.0B). The difference between these two types is the message *Identifier* (ID). The standard has a 11-bit message ID and the extended has a 29-bit message ID. This distinction is made by the *Integrated Development Environment* (IDE) bit of the CAN message, which is sent as "0" in the case of a standard message, and as "1" in case of a 29-bit message ID. CAN controllers that support CAN 2.0B are also able to send and receive messages on the CAN 2.0A format [28].

3.3.2 LIN

The LIN protocol was designed by a consortium of European auto manufacturers as a low cost, short distance, low speed network. It was specially developed to achieve cost-effective communication for intelligent sensors and actuators in motor vehicles, and it is used wherever the bandwidth and versatility of CAN are not needed [33]. In the middle of 1999 the first LIN protocol (1.0) was released and in 2000 it was updated twice. Three years later the version 2.0 was introduced to expand capabilities and make provisions for additional diagnostics features.

This protocol is based on a master-slave architecture where only the master is able to initiate a communication. The LIN data frame length is unequal (2, 4 and 8 bytes) and it consists of a header and a response part, as shown in figure 3.18.

To initiate a communication with a slave the master sends the header part. If the master wants to send data to the slave it goes on sending the response part. If the master requests data from the slave the slave sends the response part. Direct communication between slaves is not possible. But as all nodes always listen to the bus, a master request can be used to handle slave-to-slave communications. [34].

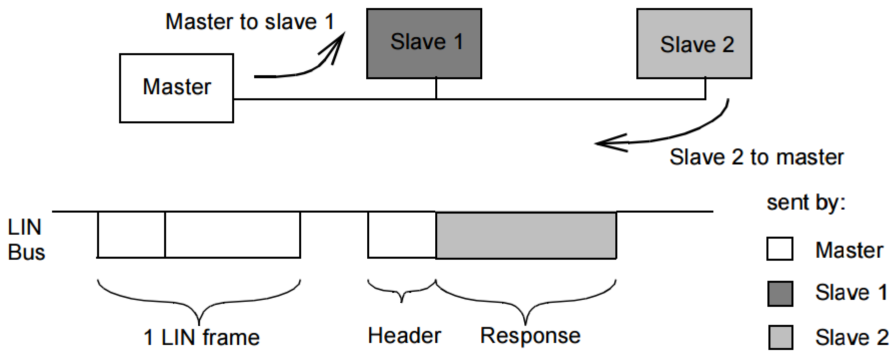


Figure 3.18: Basics of LIN communication [34].

It is a single wire communication with speed rates up to 19.2 kbit/s and a maximum bus length of 40m. The LIN has self synchronization without a quartz or ceramics resonator in the slave nodes and deterministic signal transmission with signal propagation time computable in advance.

LIN is not intended to replace existing protocols as CAN, but to complement them.

3.3.3 FlexRay

The FlexRay protocol was developed by the FlexRay Consortium formed to enable the development of next-generation by-wire systems and to promote an open-source network communications protocol for the automotive industry. FlexRay founders, BMW, DaimlerChrysler, Motorola (currently

Freescale Semiconductor) and Philips (currently NXP) have been working together since 2000 to help speed the adoption of FlexRay protocol.

FlexRay is a serial bus system that speeds up data transfer between vehicle components (up to 10 Mb/s per channel, which is twenty times faster than the conventional CAN bus system). As shown in figure 3.19, it has a dual channel and the nodes can be attached to either both channels (nodes A, B and E) or to a single channel (nodes C and D). If it is just connected to a single channel it does not matter to which one it is connected [35].

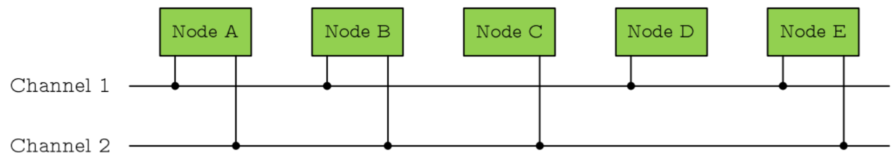


Figure 3.19: FlexRay dual channel configuration [28].

As explained before, the CAN networking uses a priority driven bus arbitration system. This means that a message with a higher priority message ID will be given access to the network if a lower priority message is also looking for accessing to the bus. The resulting message transmission delays can lead to problems for the safety systems and because of this a *Time Division Multiple Access* (TDMA) method was chosen for the FlexRay protocol [36].

With a TDMA scheme every FlexRay node is synchronized to the same clock, and each nodes waits for its turn to write on the bus. Once the timing is harmonious in the TDMA scheme, FlexRay is able to guarantee consistency of data deliver to nodes on the network, this provides many advantages for systems that depend on up-to-date data between nodes [37].

Each node consists of a communication controller, a host processor, two bus drivers and two bus guardians, one for each channel. Figure 3.20 shows the logical connections of each element.

The host controller handles the applications of the system while the FlexRay protocol is handled by the communications controller. Changes in the supply of a node can occur and this can cause defects on the bus, that is why the bus guardian is there, to prevent these defects and to organize the sending of the data on the bus. The bus driver is used to read and write data to the physical medium over which the data is transmitted.

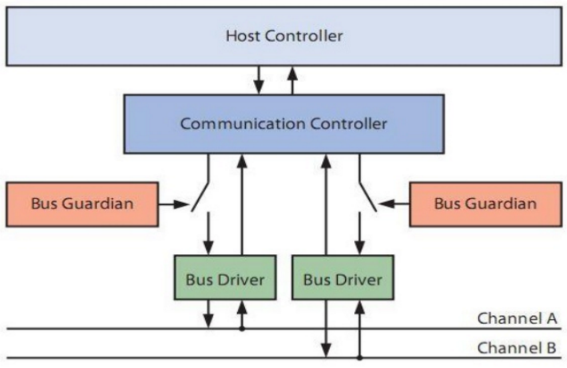


Figure 3.20: Structure of a FlexRay node [28].

In sleep mode it also has the ability to start the wakeup procedure if it detects a wakeup symbol. The communications controller will mainly handle the framing of data and the checking of received data to ensure it was uncorrupted before passing it to the host. The host and communications controller share information such as control information and payload data from the host, while the communication controller relays status information and data received. The host interface to the bus driver allows it to change the operation of the bus driver as well as to read status and error flags.

Due to its time-triggered communication features with deterministic characteristics, redundant transmission channels, fault-tolerance mechanisms and high transmission rates it is ideal for X-by-wire systems. X-by-wire includes steer-by-wire and brake-by-wire systems, which are steering and braking systems, respectively, wherein the mechanical and/or hydraulic components are replaced by another communication bus as FlexRay [28].

Comparison

In short the CAN protocol is scalable and dependable due to its bus topology. However, it lacks deterministic scheduling for real-time events because message arbitration can delay routing indefinitely. On the other hand, LIN and FlexRay both provide deterministic scheduling with the aid of a master network node. In table 3.1 a minor comparison between the three automotive networks is on display.

Table 3.1: Automotive vehicle network buses

Bus	**LIN**	**CAN**	**FlexRay**
Bandwidth	19.2 kbps	1 Mbps	10 Mbps
Physical	1-Wire	2-Wire (interference protection)	2 Channels, 2-Wire (interference protection)
Cost	Low	Medium	High
Typical Applications	Displays, Lightning, Alarm Systems, A/C, Seat and Mirror Adjustments, Windows, etc.	Powertrain (Engine, Transmission, ABS)	High-Performance Powertrain, Safety (Drive-by-Wire, Active Suspension, Cruise Control)

This page intentionally left blank.

Chapter 4

Requirements Analysis

This chapter explains the existing device in the company and analyses the main lacks of it, helping to choose the necessary requirements for the uMDC.

The *Mobility Device Controller* (MDC), illustrated in figure 4.1 was developed in the mid of 2014 aiming to be a versatile device to be installed in vehicles, charging stations, among others.

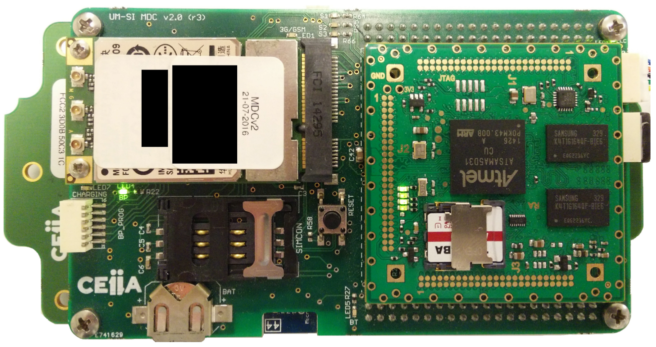

[redacted] [redacted]

Figure 4.1: MDC.

It is divided in four parts, as shown in green in figure 4.2.

The MDC is the core of the system. It runs Linux and it is responsible for the communications to/from the server. This processor turns off when the vehicle is not in use and during that time periodically wakes up to send heartbeats to the server and synchronizes the data available to the server.

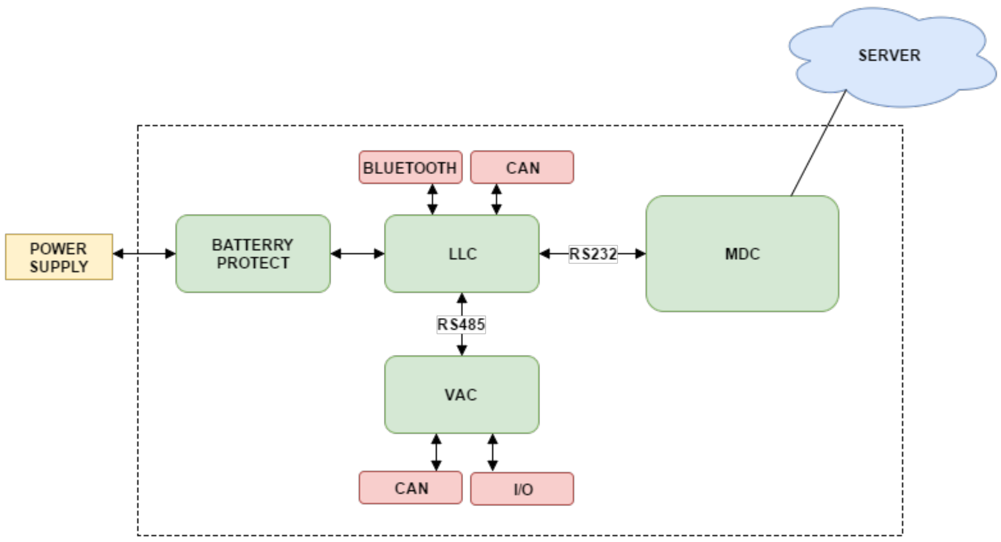


Figure 4.2: MD Vehicle structure.

The *Low Level Controller* (LLC) is a microcontroller responsible for turning on and off the MDC and for that reason it can never be turned off. It is also the interface with the CAN bus and bluetooth allowing to monitor the vehicle CAN bus and wake up the core with messages from each interface.

The Battery Protect controls all the system power and supervises the input voltage from the power supply. It is responsible for protecting the battery and turning off the system when the input voltage is lower than the threshold value. The system turns on again when the input voltage becomes above the defined threshold.

The *Vehicle Addon Controller* (VAC) is an addon for controlling the vehicle sensors and actuators, such as doors sensors and actuators, helmet sensor, horn, central stand, locker, horn, turning lights, start/stop ignition. It adds a second CAN bus to the MD Vehicle and it is connected to the LLC through RS485 protocol.

The main functions of MDC was to collect data and interact with sensors and actuators of the vehicle.

Once understood the general operation of the system and in order to choose the requirements for uMDC, it was necessary to analyze the MDC and recognize the main defects.

As it consists of several modules, it requires different programmers making the MDC a device with a complex production and installation. This feature also requires that the device has a large size making it difficult to access the modules.

The MDC has to much connectors and the programming connectors are fragile which requires a special care. The boot time is too long, around 1 minute and it does not support *Over The Air* (OTA) updates require a presence of someone in the field to update them. It does not have debug and only supports external antennas.

MDC is constantly sending information to the server implying, not only a huge data consumption, but also a high power consumption. In the table 4.1 are represented the different power consumption of the MDC, depending on the operating mode.

Table 4.1: MDC Power Consumption

Power Consumption
$MDC + LLC + BP + VAC = 210mA \times 12.86V = 2.701W$
$LLC + BP + VAC = 50mA \times 12.86V = 0.643W$
$BP = 7.54mA \times 9.87V = 0.074W$

Considering that MDC has some disadvantages, which makes it unsuitable for new challenges and projects as the bike-sharing, it was decided that uMDC must:

- Have unique cable for programming, debug, power;
- Be programmed through needles during production and take less than 30s for it;
- Be capable of debug in order to have some messages that helps during the development;
- Be available in 5 seconds or less;
- Have LEDs interface. This will help developer and/or user to know when the system is ready or when something is wrong;
- Have OTA updates through GSM and/or bluetooth, allowing the update without being in direct contact with the system;
- Be able to communicate with the smartphone through bluetooth. This requires a smartphone application to easily interface with users;
- Be capable of data logging without reporting to the server;
- Cost less than 50€, the ideal would be 30€;
- Have low power consumption in standby mode and consume less than 2W while fully running;
- Be able to resist extreme temperature (-45ºC to 85ºC);
- Have CAN system to monitor and interact with vehicles functionalities like, open the vehicle, start the ignition, open windows, etc;
- Control sharing vehicles and constantly track them through GPS and/or GSM;

- Have electromagnetic compatibility (EMC) and maybe IP[1];
- Measure around 8cm x 5cm x 1.5cm.

[1]Classifies and rates the degree of protection provided against intrusions, dust, accidental contact and water.

This page intentionally left blank.

Chapter 5

System Global Architecture

In this chapter, using a block diagram, the main components are explained as well as how they are connected and the choices made for each one . It also shows how software is structured, the communication protocol chosen and the communication security to be implemented.

Before starting the development of the project it was necessary to idealize all system from the hardware to the software, bearing in mind that the first prototype will be implemented in an electric bike. Given the competitors previously analyzed, it is essential that the uMDC has interface with LEDs, control of the electric motor and communication with the *Smart Device Cloud* (SDC) as it is illustrated in figure 5.1.

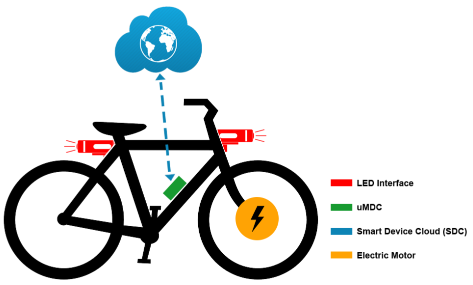


Figure 5.1: uMDC interface with an electric bike.

5.1 Hardware

To have a global concept of the system to be developed a block diagram was designed, as shown in Fig 5.2, where the main parts of the uMDC are illustrated.

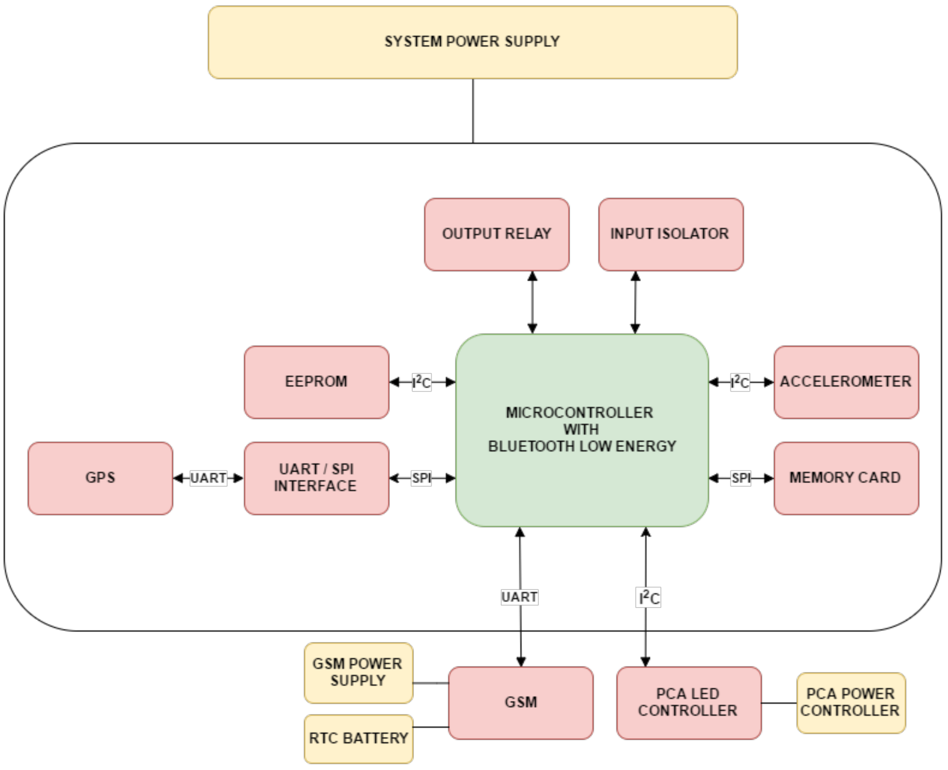


Figure 5.2: Block diagram of the system to be developed.

The largest block corresponds to the micro-controller, which is responsible for controlling all the information that is being exchanged in the system. To be easily updated via OTA and interact with the user smartphone, the micro-controller must incorporate bluetooth.

The GSM block enables the communication with the server through the mobile operators. Because of the consumption values of this kind of modules it is recommended to use a dedicated power supply as shown in figure 5.2.

The GPS block acquires the location of the system through GPS satellites. Both of this blocks will communicate with micro-controller via *Universal Asynchronous Receiver/Transmitter* (UART).

In order to save energy it will use an accelerometer. The values variation of this sensor will allow the system to change between normal mode and energy saving mode.

Since the production of this product will be large it is necessary to assign a unique serial number to each device, for that it was decided to use an *Electrically-Erasable Programmable Read-Only Memory* (EEPROM) that already has that unique serial ID. That EEPROM will also allow to write a few bytes of information if it is necessary. This module interacts with microcontroller through *Inter-Integrated Circuit* (I^2C).

The Memory Card will communicate with the microcontroller via *Serial Peripheral Interface* (SPI) and it will be used to save data when the system is off-line and save logs instead of reporting them constantly to the server.

Since it is a modular system, the use of *Input/Output* (I/O) represented will depend on the system purpose, but they will probably be used to connect a motor, a buzzer, a handbrake or bike locker.

Once idealized the main system, it was necessary to do a research on the wide range of components that are in the electronic market and compare them.

As the world of electronic components is giant, was also asked to AVNET, one of the world's largest global distributors of electronic components, some solutions for this system, bearing in mind one of the main requirements for uMDC which is the maximum value of 50€.

5.1.1 Microcontroller

As most of the people know, the microcontroller is one of the most important (if not the most important) components in a system. It is where all the information will be processed and if it has no capacity for that, the system may crash or even break. In table 2, available in Appendice B , we may consider a comparison between some relevant microcontrollers in the market that were suitable to this solution.

Thanks to the very low price and the integrated bluetooth low energy, the nRF52832 from Nordic Semiconductors was chosen(figure 5.3). It is a powerful, highly flexible and ultra-low power SoC, ideally suited for bluetooth low energy applications. The nRF52832 SoC is built around a 32-bit ARM Cortex-M4F CPU with 512 kB plus 64 kB RAM. The embedded 2.4 GHz transceiver supports BLE, ANT and proprietary 2.4 GHz protocol stack [38]. The BLE will be an asset to the system because it will allow the updates over the air and the interaction with the user.

Figure 5.3: Nordic nRF52382 microcontroller [38].

It is an extremely power efficient device, that can run from a supply between 1.7 V and 3.6 V. All individual peripherals and clocks offer complete flexibility of power down when not required for task operation, thus minimizing power consumption to a minimum, as required for the uMDC. The Integrated Circuit (*Integrated Circuit* (IC)) has a comprehensive system of automated and adaptive power management features [38]. This microcontroller provides three SPI channels and two I^2C channels. However, it only provides one UART and, because of that, it will requires a converter for UART to SPI or UART to I^{2}C, in order to use the GPS or GSM module. The *Real Time Counter* (RTC) module provides a generic, low power timer on the *Low-Frequency Clock Source* (LFCLK). Like the other microcontrollers from Nordic, the nRF52832 holds a special feature in which most of the *General Purpose Input/Output* (GPIO)s can be flexible, i.e., a GPIO can be defined as a UART channel and be immediately defined as a SPI channel or even as a simple I/O.

Finding a solution that fulfills all the requirements is not easy and, unfortunately, this microcontroller is GPIO limited (only 32) and it also hasn't a built-in CAN controller, but it's still a cheap and powerful microcontroller.

5.1.2 GSM

The uMDC will integrate the Mobi.me service and, as explained before, this service manages all the shares of vehicles in real time. Thus, it was decided to adopt the GSM technology as medium communication between the uMDC and the MOBI.ME suite. It is an old telecommunication technology but it is inexpensive and meets the necessary system requirements. In table 4 of Appendice B it's a comparison between five GSM modules, two of them with built-in GPS.

The GSM module has a high energy consumption so it need to be in power saving mode, or even turned off, most of the time. On the other hand, the GPS needs to be frequently updating the location of the uMDC. For that reason the GSM modules with built-in GPS were excluded. After that, and after having seen the high price of the others, the Quectel M95 was chosen. Not only because it is cheaper, but also due to the fact that the same footprint supports *Universal Mobile Telecommunications System* (UMTS) modules. As it is predicted that some GSM antennas will be disconnected in 2017 it is better have a solution that supports both technologies than redesign the whole system.

The Quectel M95 module, illustrated in figure 5.4, is one of the smallest Quad-band GSM/GPRS modules, which makes M95 easily embedded into the low-volume applications and ensures the reliable connectivity with the applications. It has a low power consumption around 1.3 mA and extended temperature range.

Figure 5.4: Quectel M95 GSM module [39].

With surface mounted technology, the low profile and small size of *Leadless Chip Carrier* (LCC) package makes M95 easily embedded into the low-volume applications and ensures the reliable connectivity with the applications. This kind of package is ideally suited for large-scale manufacturing which has the strict requirements for cost and efficiency [39].

Thanks to the built-in QuecFOTA technology, the firmware of Quectel M95 can be remotely updated. It detains additional features such as integrated TCP/IP protocol stack, serial multiplexer and enhanced AT commands that guarantees fast and reliable transmission of data, voice, SMS via GSM/GPRS network and extends the functionality of the application without adding any cost [39].

5.1.3 GSM Power Supply

Since the GSM module is the one that is constantly in touch with the server and given the current peaks and possible restarts of the system, it was decided to use a dedicated power supply. During the analysis of power supplies directed to GSM modules, the TPS54260 from Texas Instruments was chosen.

The TPS54260 is a step down regulator with integrated high-side *Metal Oxide Semiconductor Field Effect Transistor* (MOSFET). It supports an input voltage up to 60 V, which is required since the solution will be used in electric bicycles that typically use batteries up to 42 V. This power supply can also provide a current up to 2.5 A. As stated in the datasheet of the manufacture (Appendice C), this power supply is highly recommended for GSM and *General Packet Radio Service* (GPRS) modules in fleet management.

5.1.4 GPS

For the GPS solution a component from Quectel was chosen, namely the Quectel L70-R, a low cost *Read-only memory* (ROM)-based GPS module, figure 5.5. It has a low power consumption in tracking mode (18 mA), a high precision and sensitivity, a tiny design and it is cheap.

Figure 5.5: Quectel L70-R GPS module [40].

L70-R holds *Embedded Assist System* (EASY) technology that can calculate and predict orbits automatically using the ephemeris data (up to 3 days) stored in internal memory, so L70-R can fix position quickly even at indoor signal levels with low power consumption [40].

5.1.5 UART/SPI Interface

As explained before, the microcontroller is limited to just one UART channel and since both GSM and GPS only support UART protocol, it was necessary to use a converter.

During some researches the SC16IS740 from NXP was found. It is a slave I^2C/SPI interface to a single-channel high performance UART that offers data rates up to 5 Mbit/s and guarantees low operating and sleeping current. This interface operates with an input voltage of 3.3 V and it has a *First In, First Out* (FIFO) of 64 byes. The SC16IS740 also provides additional advantaged features such as auto hardware/software flow control and software reset that allows to reset the UART at any time via software, independent of the hardware reset signal.

5.1.6 Accelerometer

The users safety was one of the things to take into consideration during the meetings about the uMDC. Since this solution will be used in sharing vehicles, which includes not only cars but also bicycles and motorcycles, one of the ideas was to use an accelerometer to determine abrupt variations in acceleration. This will help the system to know if someone falls out of the bike or the motorcycle or if something happened to the car. This sensor will also be used to determine when the system is in use or not, allowing it to enter into power saving mode.

The choice of this components was easy since the MDC has already a built-in accelerometer. For that reason it was decided to use the same, the MMA8453Q from NXP. The MMA8453Q is illustrated in figure 5.6 and it is a smart, low-power, three-axis, capacitive, micromachined accelerometer with 10 bits of resolution. This accelerometer is packed with embedded functions with flexible user programmable options, configurable to two interrupt pins. Embedded interrupt functions allow for overall power savings relieving the host processor from continuously polling data [41].

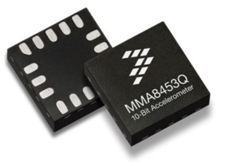


Figure 5.6: NXP MMA8453Q accelerometer [41].

The MMA8453Q has user selectable full scales of $\pm 2g/\pm 4g/\pm 8g$. The device can be configured to generate inertial wakeup interrupt signals from any combination of the configurable embedded functions allowing the MMA8453Q to monitor events and remain in a low-power mode during periods of inactivity [41].

5.1.7 EEPROM

As explained before, in order to have a unique serial number in each uMDC, there was the need to use an EEPROM. So, once the MDC is already using one, we decided to implement the same.

The Atmel AT24MAC602 2 Kb Serial EEPROM device is an application product that contains a unique IEEE-provided 64-bit[1] pre-programmed MAC/EUI address to enable a connected device to connect to the Internet or local net-

[1]https://standards.ieee.org/develop/regauth/tut/eui64.pdf

work. The device also contains a unique read-only 128-bit serial number and 2Kb of user-accessible serial EEPROM *Non-Volatile Memory* (NVM) storage [42].

5.1.8 Output Relay

In order to actuate the electric motor of the bicycle it was necessary to use a *Solid State Relay* (SSR).

The VO14642AT is a high speed *Single Pole Single Throw* (SPST)[2] normally open solid-state relay. The relays are constructed as a multi-chip hybrid device. Actuation control is via an infrared LED. The output switch is a combination of a photodiode array with MOSFET switches. The relays can be configured for *Alternating Current* (AC)/*Direct Current* (DC) or DC only operation [43].

Figure 5.7 illustrates the operating and the pin configuration of the VO14642AT. Once we are going to use DC, pins 4 and 6 will be shunted.

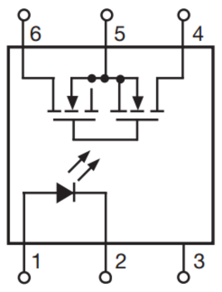


Figure 5.7: VO14642AT internal design [43].

The electric motor of the bicycle works with a tension of 42V and since this relay supports a maximum load voltage of 60V, it will be perfect for the system.

[2]A simple on/off switch where the two terminals are either connected together or disconnected from each other.

5.1.9 Input Isolator

So it can be possible to connect external components such as bicycle brakes, protecting the system from unwanted AC/DC currents, a digital isolator, the ISO7220 from Texas Instruments, was used.

It is a dual-channel digital isolator that has a logic input and output buffer separated by silicon-dioxide isolation barrier, providing galvanic isolation up to 4000 V. Used in conjunction with isolated power supplies, this device blocks high voltage, isolates grounds, and prevents noise currents on a data bus or other circuits from entering the local ground and interfering with or damaging sensitive circuitry [44].

This isolator operates with 3.3 V or 5 V, it has a high electromagnetic immunity and it is qualified for automotive applications.

5.1.10 PCA LED Controller

One of the requirements for the uMDC was the LEDs interface, and since the microcontroller is pin limited, it was necessary to use a LED controller.

The PCA9685 is an I^2C-bus controlled 16-channel LED controller optimized for *Red/Green/Blue/Amber* (RGBA) color backlighting applications. Each LED output has its own 12-bit resolution fixed frequency individual *Pulse Width Modulation* (PWM) controller that operates at a programmable frequency from a typical of 24 Hz to 1526 Hz with a duty cycle that is adjustable from 0% to 100% to allow the LED to be set to a specific brightness value. All outputs are set to the same PWM frequency. Each LED output can be off or on (no PWM control), or set at its individual PWM controller value.

The LED controller operates with a supply voltage range of 2.3 V to 5.5 V and the inputs and outputs are 5.5 V tolerant. LEDs can be directly connected to the LED output (up to 25 mA, 5.5 V) or controlled with external drivers and a minimum amount of discrete components for a larger current or higher voltage LEDs [45].

5.1.11 Power Supply

After having chosen all the main components it was time to decide on the best power supply for the system.

Since the electric bicycle battery has an output voltage up to 42 V, it was imperative that the power supply would tolerate an input voltage higher than that, and once all the components (excluding the GSM module) operates with an input voltage of 3.3 V, the power supply has to be capable of step down 42 V into 3.3 V.

Analyzing these requirements the choice fell on the Allegro A8498. It is a step down regulator that handles a wide input operating voltage range between 8 V to 50 V. The output is adjustable from 0.8 V to 24 V and it has a better efficiency. Figure 5.8 graphically represents the variation the variation of the efficiency versus the output current.

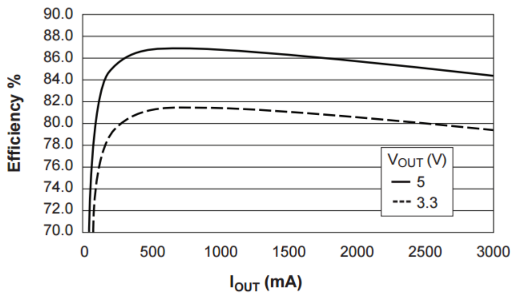


Figure 5.8: Efficiency vs Output Current [46].

As is possible to observe, this solution has an efficiency higher than 80% which is better for the solution to be developed.

5.2 Software

Due to the complexity and number of different modules present in the uMDC system, a simple, clear and modular system architecture is required. Such architecture will ensure the code maintainability and will minimize the number of problems that might arise both in the development phase and during the lifetime of the system.

To ensure this simplicity while dealing with the multiple modules, it was decided to use a state machine based system. This way each module will have a well defined functioning mode with clear transition conditions between states. It will also allow a greater flexibility to introduce new behavior in the system. The interface between the different modules will be simplified by the state machine model since it allows the triggering of events between modules in an unambiguous manner.

To further guarantee the separation of the system's modules and structure the different components, the different levels of the system will be separated in Managers level, Modules level and *Hardware Abstraction Layer* (HAL) level (figure 5.9).

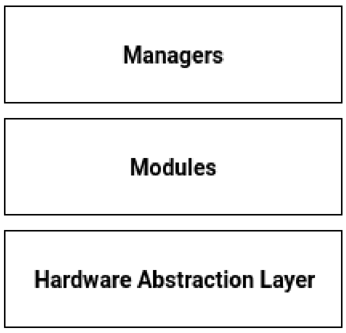


Figure 5.9: uMDC - Software Levels

Managers: The uMDC will have a Manager to handle each different part of the system that requires one. The Managers will implement the system logic while ensuring that the correct actions are taken at each moment and that the events trigger the correct transitions.

For instance, the power state Manager of the uMDC could be responsible for deciding when the uMDC would go to sleep, when to reset, etc.

Modules: The Modules are responsible for handling the different Managers requests and redirecting them to the correct Hardware Abstraction Layer. The different Modules will be responsible for ensuring that the Managers do not access the wrong hardware and that the actions that are requested can be executed at that time.

Different modules in the uMDC can be: GPS, GSM, LEDs, Accelerometer, etc.

Hardware Abstraction Layer: The HAL is responsible for interfacing directly with the hardware. Having a layer between the modules and the hardware allows a safer interaction between the parts. The HAL will ensure that all the condition are verified before taking any action and will provide a clear API that can be called by any module.

Having a HAL also makes it easier to change the hardware definition without breaking the compatibility with what is implemented in the rest of the system.

For instance, if a GPIO needs to be actuated the module will call a GPIO HAL that will be responsible for setting the correct value on the correct register enabling or disabling the defined GPIO.

Every HAL that is responsible for communicating with different components (via UART, SPI, etc) will have to ensure that the communication has the minimal influence in the system loop. To ensure that, the HAL will implement a FIFO bigger than the maximum message it can receive allowing multiple modules to write to the same path.

5.2.1 Debug

During the development phase the system will be debugged using *GNU Debugger* (GDB). However during the regular usage of the system, since it is a microcontroller based system, remote debugging is not possible. To overcome this issue, an optional *Secure Digital* (SD) Card based logging system will be implemented. This system will log errors and problems with the uMDC during the normal usage of the system.

To avoid having a big interference of the logging service in the uMDC usage the log will be code based with a corresponding translation layer.

e100 - Error unsupported message type
e200 - Error no Internet connection

The presence of the SD Card will be optional, to keep the cost down, however the writing loop will always be present to ensure consistent behaviour between the system with and without logging system.

5.2.2 Communication protocol

Among the existing options that were previously analyzed, it was decided to use the communication protocol MQTT.

MQTT provides a scalable and cost efficient way to connect devices over the Internet. It is capable of delivering messages in near real time and guarantees its delivery. Connecting thousands of devices, sending instant updates and broadcasting push notifications this is where MQTT truly excels. It is made for devices and is lightweight on the wire, enabling low-cost device communication that scales without problems.

It is widely adopted by many devices and companies worldwide for data exchange with constrained devices or products out in the field. MQTT keeps bandwidth at an absolute minimum and deals with unreliable networks without complex error handling and a huge implementation effort. It was made to keep a steady line to the devices at a minimal cost supporting real push notifications and near real time communication.

The publish/subscribe paradigm enables easy broadcasting of messages from one publisher to many subscribers. Messages from a huge amount of publishers to a few subscribers is covered as well. Additionally the connection is built-up from the client side, which makes a bidirectional communication possible without *Network Address Translation* (NAT) translation issues (figure 5.10).

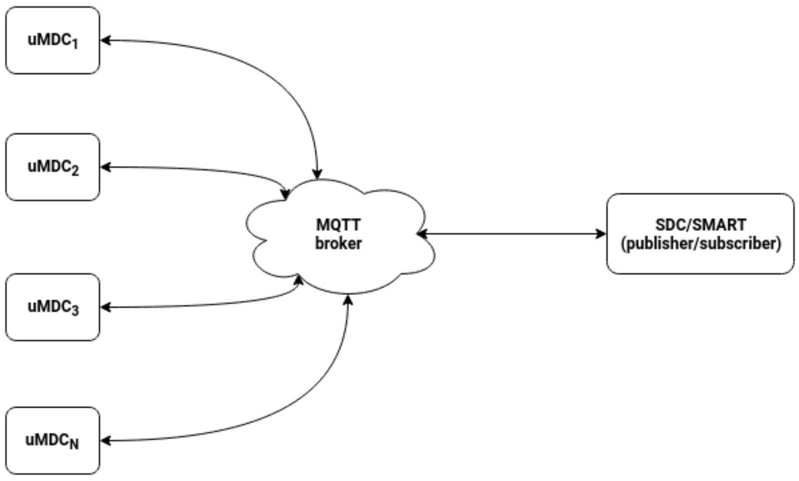


Figure 5.10: uMDC - MQTT architecture

Heartbeat

The uMDC heartbeat service will be based on *MQTT Keep Alive*[3]. The keep alive functionality assures that the connection is still open and both broker and client are connected to each other. Therefore the client specifies a time interval in seconds and communicates it to the broker during the establishment of the connection. The interval is the longest possible period of time, which broker and client can endure without sending a message. That means as long as messages are exchanged frequently and the keep alive interval is not exceeded, there is no need to send an extra message to ensure that the connection is still open.

However, if the client doesn't send any messages during the period of the keep alive it must send a *PINGREQ* packet to the broker to confirm its availability and also to make sure the broker is still available. The broker must disconnect a client, which doesn't send *PINGREQ* or any other message in one and a half times of the keep alive interval. Similarly the client should close the connection if the response from the broker is not received in a reasonable period of time.

Data report

Data will be published to the MQTT topic:

data/<clientID>

Data will be reported always in real-time, which means that the uMDC does not support data synchronization.

Data will be sent as MQTT payload, using only *American Standard Code*

[3] `http://www.hivemq.com/blog/mqtt-essentials-part-10-alive-client-take-over`

for Information Interchange (ASCII) characters, formatted as such:

<dataID>,<size>,<data>

An example for GPS altitude is:

GPS,0A,12.4578126

Event report

Events are similar to data reports and will be published to MQTT the topic:

event/<clientID>

An example, for a report of uMDC going to poweroff is:

PWR,08,poweroff

Function execution

Concerning the function execution, each uMDC will provide all the "clientID" functions using topic wildcards [4] :

function/<clientID>/#

where # is the wildcard for multiple levels.

[4] `http://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices`

Firmware update

The firmware update is a function topic and it will be performed by the MQTT broker publishing to:

function/<clientID>/firmwareupdate

The firmware update will be sent as MQTT payload, using only ASCII characters, formatted as such:

<firmware_version>,<msg#>,<total_msgs#>,<size>,<firmware_bytes>

An example, for firmware update is:

3.1.5,34,60,40,GWMivzvEB8JQF6y2slo1J30M54wmpqIeTWL3v7ZZyDAMs jTmFM6iZPliGyBtEny

Firmware configuration

The firmware configuration is a function topic and will be performed by the MQTT broker publishing to:

function/<clientID>/firmwareconfig

uMDC configuration

The uMDC configuration will be sent by the MQTT broker, after the first boot. This is a function topic and will be performed by the MQTT broker publishing to:

function/<clientID>/umdcconfig

5.2.3 Security

Security in MQTT is divided in multiple layers. Each layer prevents different kinds of attacks. The goal of the protocol is to provide a really lightweight and easy to use communication protocol for the IoT. So that is why in the protocol itself there are only a few security mechanisms clearly specified. But in all common implementations other state-of-the-art security standards are used, like SSL/TLS for transport security.

At the network level, using a physically secure network or *Virtual Private Network* (VPN) as foundation for any communication between clients and the broker is one way to provide a secure and trustworthy connection. This would be suitable for gateway applications, where the gateway is connected to devices on the one hand and with the broker over VPN on the other hand.

At the transport level, when the goal is to provide confidentiality in most cases TLS/SSL is being used for transport encryption. It provides a secure and proven way to make sure nobody can read along and even authenticate both sides, when using client certification authentication.

At the application level, on the transport level it can be ensured that the communication is encrypted and the identity is authenticated. The MQTT protocol provides a client identifier and username/password credentials, which can also be used to authenticate devices on the application level. These properties are provided by the protocol itself. When it comes to authorization or what each device is allowed to do, it lays in the hand of the broker implementation, how to handle it. Another possibility is to use payload encryption on the application level, in order to make the transmitted information secure even without having a full fledged transport encryption.

Additionally to the security provided by MQTT protocol, each uMDC will have a unique device ID programmed into the hardware which will be used on each subscribed topic. Moreover, the MQTT broker will also be configured with permissions for each device added to the network.

This page intentionally left blank.

Chapter 6

System Development

This chapter is dedicated to the hardware and software development, which will be explained in more detail all its realization. During this explanation the tools used for PCB design, some hardware configurations, the tools used in programming and the code developed for the first prototype will be disclosed.

6.1 Hardware

For the PCB design was used the Kicad, an open source software suite for *Electronic Design Automation* (EDA). It was developed by Jean-Pierre Charras and handles schematic capture and PCB layout with gerber output. To better understanding the decisions made during the design, it is individually explained the main assembling of each component.

6.1.1 Microcontroller

Given the importance, and to be easier to design all the other components, the first component designed was the microcontroller. It is powered by 3.3 V, as most of the components, and provides two internal regulators, which need different filters to connect them. Since the DC/DC regulator reduces current consumption compared to the *Low Dropout* (LDO) regulator, it was decided to use it (DC/DC regulator), and for that reason it was required an external LC filter, represented in the figure 6.1 by L5, L6 and C34.

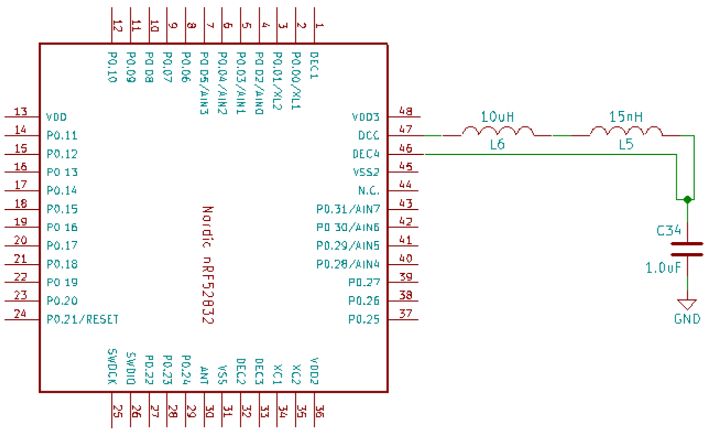


Figure 6.1: External LC filter for the DC/DC internal regulator.

The clock control system of the nRF52832 can source from a range of internal or external high and low frequency oscillators and distribute them to modules based upon a module's individual requirements. It supports two external crystal, Y2 and Y3, as shown in figure 6.2. The crystal Y3 is mandatory because it will provide an accurate clock to the radio. The Y2 is optional, but its use will give a lower average current consumption than using the internal RC oscillator.

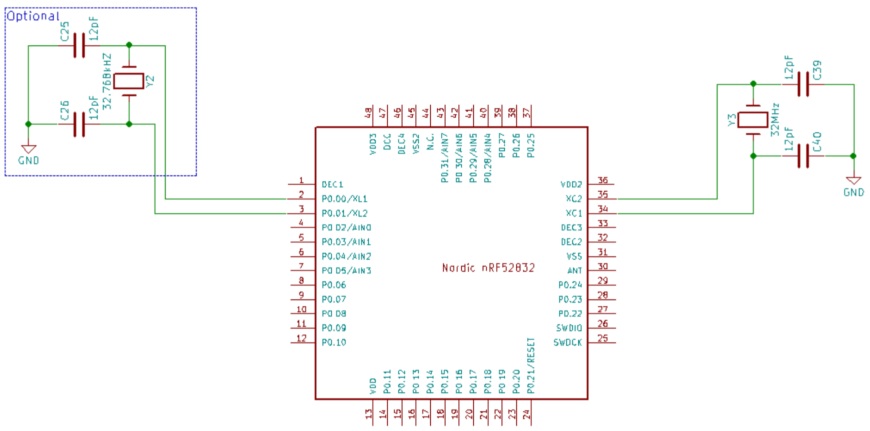


Figure 6.2: External crystals for clock control.

The output impedance of the chip is not 50 ohm. Therefore a matching network[1] was inserted between the chip and the reference point, as shown in figure 6.3. The matching is done with two components L7 and C38. These components also form a low pass filter which will suppress harmonics which are needed for the design to pass various regulatory standards [47].

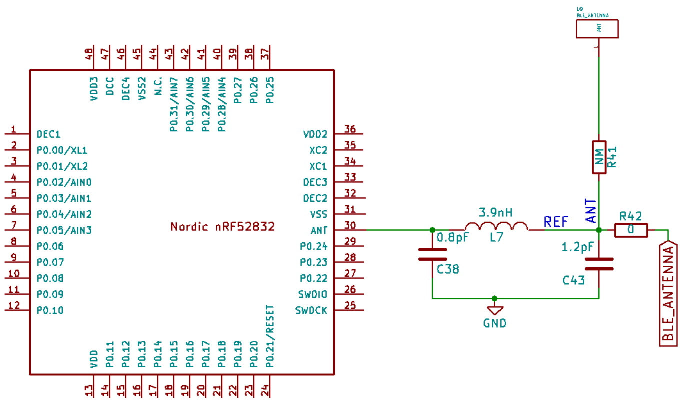


Figure 6.3: Matching between the chip and the antennas.

[1]Impedance matching is to design the electronics so that the impedance in each point of the circuit give maximum power transfer.

It was idealized to use a PCB antenna, and for that reason the matching network circuit for 50 Ω antenna, consist of one shunt component, the capacitor C43 (figure 6.3). However, there is no guarantee that PCB antenna would work properly. For that reason it was also implemented a ceramic antenna, represented by U9.

As required in chapter 4, it was implemented an LED interface to help developer and/or user to know when something is wrong. It is represented by D5 in figure 6.4 and it also uses a resistor of 330 Ω (R35) for limiting the current to 10 mA and does not burn the LED.

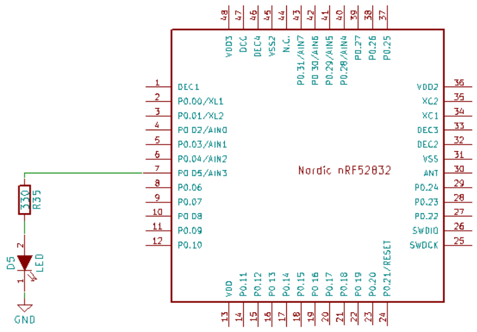


Figure 6.4: System LED interface.

6.1.2 System Power Supply

The output voltage is adjustable from 0.8 V to 24 V, based on the combination of the value of the external resistor divider and the feedback voltage (0.8 V). For that reason, and in order to have an output of 3.3 V, it was used the equation 6.1.

$$V_{OUT} = V_{FB} \times (1 + \frac{R_1}{R_2}) \tag{6.1}$$

To determine which resistor values to use for the output voltage, 2 kΩ was selected for R2 resistor and then it was calculated the R1 which is approximately 6.34 kΩ. This resistor divider is shown in figure 6.5 and it is represented by the resistors R13 and R12.

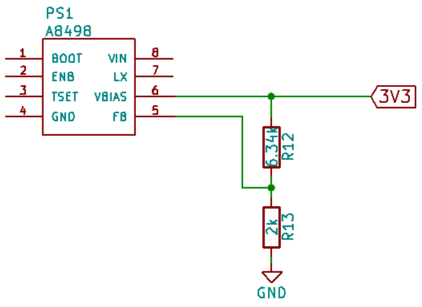


Figure 6.5: Resistor divider for 3.3 V output voltage.

The manufacturer's datasheet provides a table with the recommended components for an input voltage of 42 V. Seen it all the other components were dimensioned according to that, which is available in Appendice D.

6.1.3 GSM

In order to have a better performance of the Quectel M95 it was recommended to place a tantalum capacitor with low *Equivalent Series Resistor* (ESR) in parallel with three ceramic capacitor near the VBAT pin. This implementation is illustrated in figure 6.6 by C2, C3, C4 and C47.

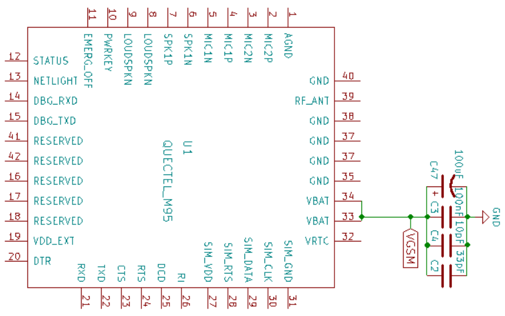


Figure 6.6: VBAT input circuit.

The Quectel M95 module can be turned on/off by driving the pin PWRKEY to a low level voltage (GND), and to control that pin it was used an open collector driver circuit, composed by R1, R2 and Q10, as shown on figure 6.7

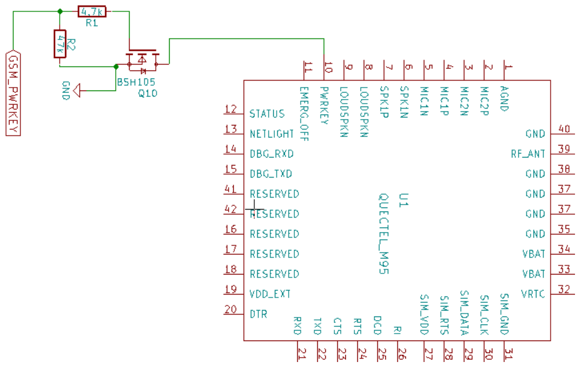


Figure 6.7: Open collector driver circuit to control the pin PWRKEY.

M95 provides a *Radio Frequency* (RF) antenna pad for antenna connection, whose characteristic impedance should be close to 50 Ω , and to be possible to adjust the RF performance it was implemented a $\pi-$match circuit composed by C48, C49 and R8 (figure 6.8). The J1 represented is the connector for an external GSM antenna.

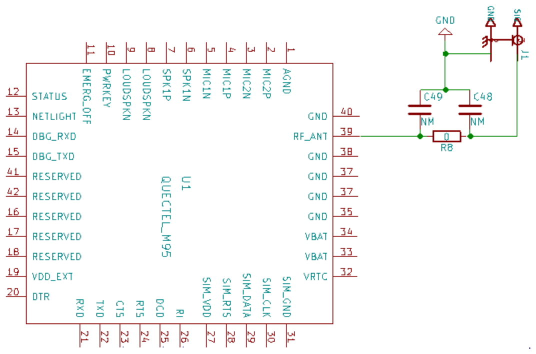


Figure 6.8: GSM RF antenna $\pi-$match circuit.

Once the GSM module as a built-in *Real Time Clock* (RTC), it was used a battery (BT1) to keep time counting even when the GSM power is off. As shown in figure 6.9 it was also used a schottky diode (D3) to protect the circuit against reverse current.

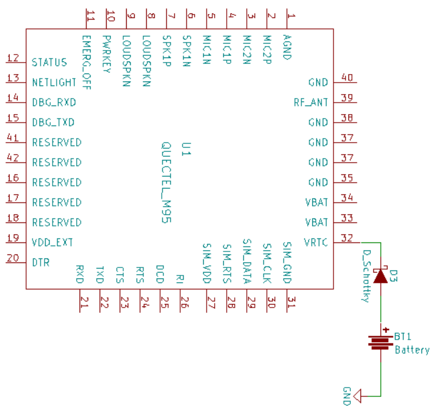


Figure 6.9: RTC Supply from a Non-chargeable Battery.

6.1.4 GPS

The Quectel L70 receives L1 band signals from GPS satellites. The RF signal is obtained from RF_IN pin and the impedance of RF trace line should be controlled by 50 Ω. In the figure 6.10, as used on Quectel M95, it is shown the $\pi-$match circuit (R26, R27 and R29) implemented for antenna impedance modification.

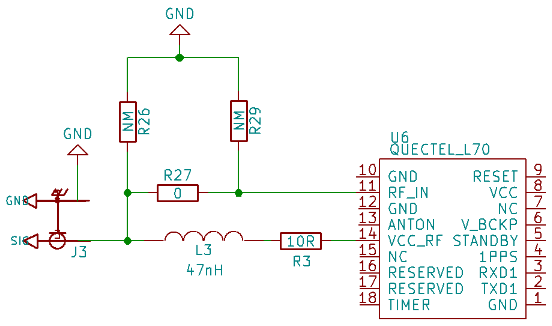


Figure 6.10: GPS RF antenna $\pi-$match circuit.

6.1.5 UART/SPI Interface

As explained in the system architecture, it was required to use a UART to SPI interface module.

This module requires a use of a crystal to form an internal oscillator circuit. The crystal is represented in figure 6.11 by Y1 as well as the capacitors, C10 and C13, which are part of that circuit.

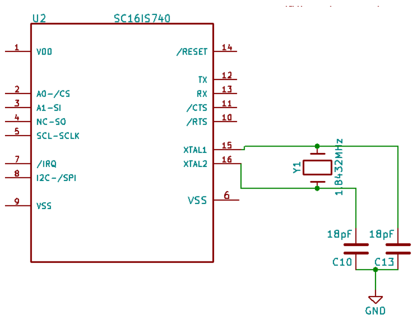


Figure 6.11: Internal Oscillator Circuit.

Usually different components has different I/O voltages. In this project, the motor communication protocol operates at 5 V and once the maximum UART output channel (TX) voltage of the converter is 3.3 V, it was required the use of a level shifter.

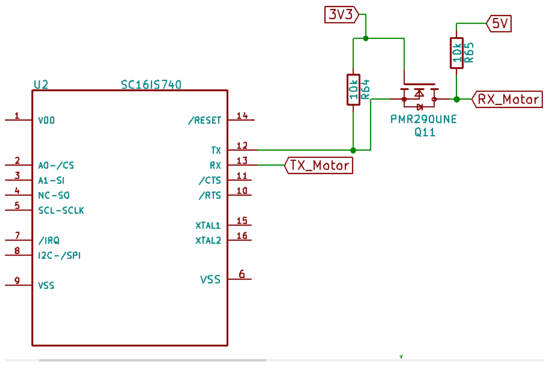


Figure 6.12: UART TX level shifter.

This level shifter is shown in figure 6.12, and it consists in MOSFET (Q11) with two pull-up resistors (R64 and R65). This circuit basically ensures that a 5 V signal reach the high voltage device (motor) without affecting the low voltage device (converter).

For the UART input channel it has not been necessary to implement any circuit because it is 5 V tolerant.

The design of the GSM power supply was based on a application report from Texas Instruments that describes how to design power supply for a GSM module [49]. This document also provides all the recommended components for an output voltage of 3.8 V and an output current of 2.5 A.

All the remaining components were designed according to the manufacturer's datasheet.

Kicad is not a complete software and it has some disadvantages, one of them is the needed to create the *Bill of Materials* (BOM) before starting the PCB design. It is also limited in terms of components footprints, so it was necessary design the footprints of the components that did not exist in the Kicad libraries. One of them was the GSM module footprint, as shown in figure 6.13. All the measures required for design were available on the manufacture datasheet.

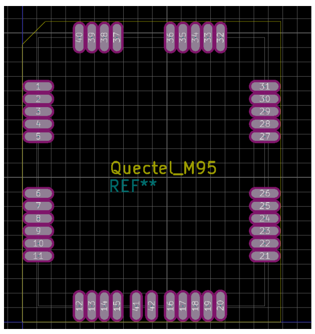


Figure 6.13: Quectel M95 footprint.

When a footprint of a components is being draw, it is necessary to be very careful with the measures, because if they are not correct and the component does not fit the pads, it needs to be redesign which can lead to a monetary loss.

After having conclude the schematic (Appendice E) and all the components having been chosen with the assigned footprints, it was proceeded to design the PCB. It measures 85mm x 55mm x 3mm, which is roughly the size of a credit card and because of that small size, and easy to connect everything four layers were used, two layers for tracks/vias (top and bottom), one layer for power and other layer for ground. The design of the layers is available on Appendice E.

The microcontroller was supposed to be in the center of the PCB to be easier for routing but, as it needs a printed PCB antenna, it was placed closer to the edge. The printed PCB antenna was dimensioned according to Appendice F. It has a width of 1.5 mm and a physical length of 23 mm.

The GSM module and GPS module were also placed close to the edge because of the antennas, being spaced between them so there is no signal frequency conflict. Another consideration to bearing in mind was the *Subscriber Identify Module* (SIM) card. As requested in the GSM manufacturer's datasheet, it needs to stay as close as possible to the GSM module.

In the future, the production of this system will be in large amounts and all the components will be welded in an oven. Since this PCB has components in the both sides, it will be welded on one side at a time. During this process, to prevent the components already welded on one side would fall during the welding process of the opposite side, the larger and heavier components were located on the top layer and the others, mostly resistances, capacitors and inductors on the bottom layer.

6.2 Software

Due to the complexity and the constrained resources of embedded systems, choose which programming language to use is an extremely necessary requirement. It was decided to use Embedded C++ which is a dialect[2] of the C++ programming language for embedded systems.

Developed since 1979 by Bjarne Stroustrup, C++ is an object oriented programming language. It is a newer language, based on C, that adds many modern programming language features that makes it easier to program than C. C++ is considered to be an intermediate level language, as it encapsulates both high and low level language features. Initially, the language was called 'C with classes' as it had all properties of C language with an additional concept of 'classes'. However, it was renamed C++ in 1983 [48].

The purpose of Embedded C++ is to preserve the most useful features of object oriented language minimizing the code size while increasing the efficiency of the system . The development of the uMDC software using this language will allow the development of a modular software and simplify the implementation of new variables and functions protecting the existing ones.

Nordic Semiconductors provides a *Software Development Kit* (SDK) for microcontrollers which includes drivers, libraries, examples and APIs. Since the Nordic Semiconductors libraries are "written" in C, it was necessary to adapt our C++ code to guarantee that these libraries were correctly integrated.

Thereby the method "extern C" was used, as shown below. This method simply notifies the compiler that the noted function is (or needs to be) compiled in C style.

[2]Small variation or extension of the language that does not change its intrinsic nature.

```
extern "C" {
#include "nrf_gpio.h"
}
```

As explained in the software architecture (chapter 5), to ensure a simple, clear and modular system it was used a state machine based system. To guarantee the separation of the system's modules and to struct the different components, the software is divided into three layers: the Managers layer, the Modules layer and *Hardware Abstraction Layer* (HAL).

The first level to be developed was the HAL. It is the API responsible for interfacing directly with the hardware and it is composed by *GPIO*, *I2C*, *SPI*, *SPI_UART*, *UART* and *TIMER*, as represented in figure 6.14.

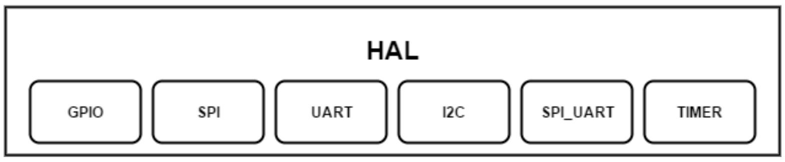


Figure 6.14: HAL layer composition.

With the exception of the *TIMER*, all classes are constituted by functions that simplify the communication between the microcontroller and the remaining hardware.

After the *Hardware Abstraction Layer* (HAL) having been developed, and bearing in mind that this was the first prototype, dedicated classes for testing three different modules, *Accelerometer*, *Motor* and *LEDs controller* were created, as illustrated in figure 6.15.

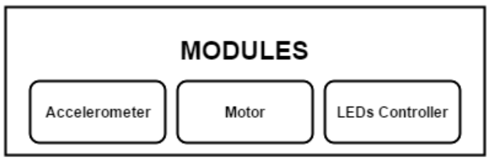


Figure 6.15: Modules layer composition.

The *Accelerometer* class inherits from the *I2C* class (line 1) and it simply implements functions for module configuration (line 10) and reading accelerometer values (line 11).

```
1  class CAccelerometer : private CI2C
2  {
3    public:
4    struct accelerometer_data_t {
5      int16_t x;
6      int16_t y;
7      int16_t z;
8    };
9    CAccelerometer();
10   uint8_t config();
11   void readData(accelerometer_data_t* destination);
12 };
```

The *Motor* and the *LEDs Controller* classes are structured in the same way as *Accelerometer*, however the *Motor* class inherits from *SPI_ UART* instead of *I^2C*.

At some point of this work given over to the system architecture chapter, it came up that LEDs have a dedicated controller and once it interfaces with the microcontroller through I^2C it needs to use the *I^2C* class functions. Thus, the *PCA* class that is responsible for configuring the module, turning on/off the LEDs, setting brightness, among other functions, was set up.

To have a full control over the electric motor the *Motor* class was developed. This class inherits from the *SPI_ UART* class and it is responsible for checking the motor status, turning on/off the motor, setting the assistance level and increasing/decreasing the assistance level when it's necessary.

Working as managers two state machines were implemented, one for accelerometer and another one for the motor controller, as shown in figure 6.16.

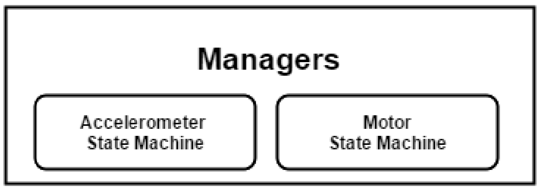


Figure 6.16: Managers layer composition.

The accelerometer state machine will be the responsible for setting the *Moving* and *Not Moving* bicycle states as presented in figure 6.17.

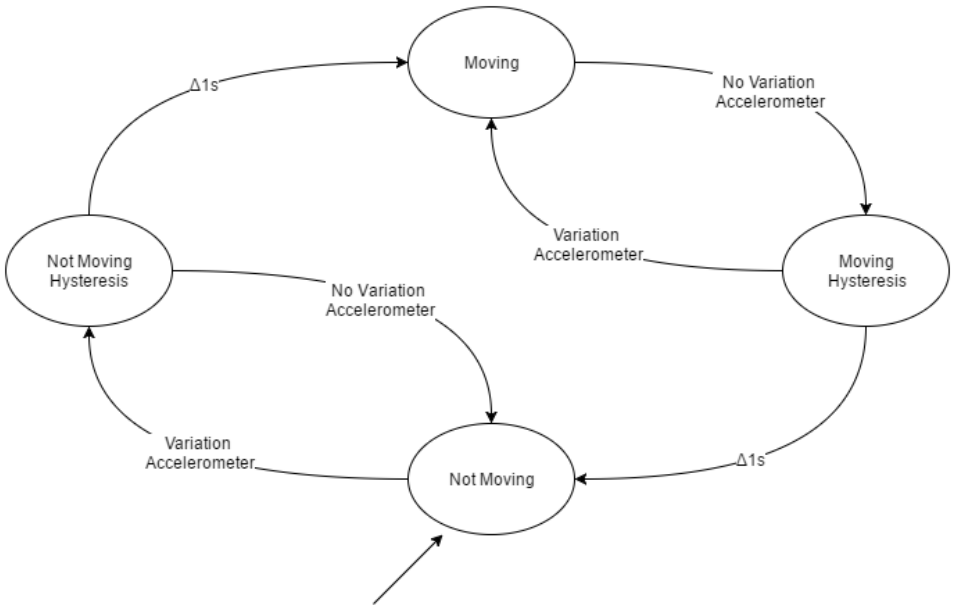


Figure 6.17: Accelerometer state machine.

The second state machine, for the motor controller, which will take over the motor assistance will be responsible for turning it on and turning it off, as shown in figure 6.18.

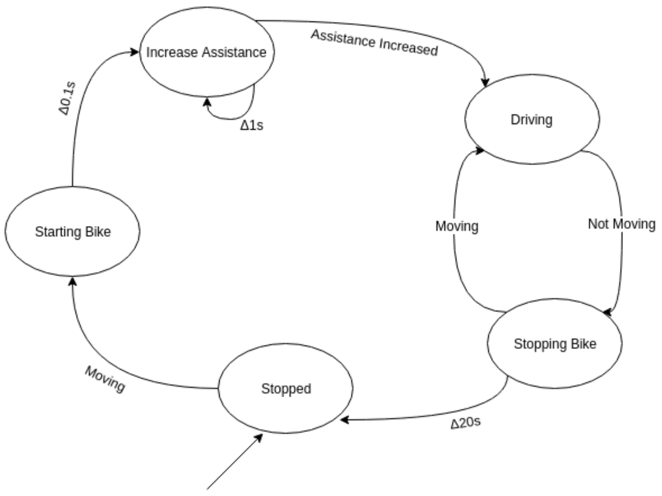


Figure 6.18: Motor state machine.

The motor state machine will be fed by the *Moving* and *Not Moving* states set by the accelerometer state machine. The interaction between both states is illustrated in figure 6.19.

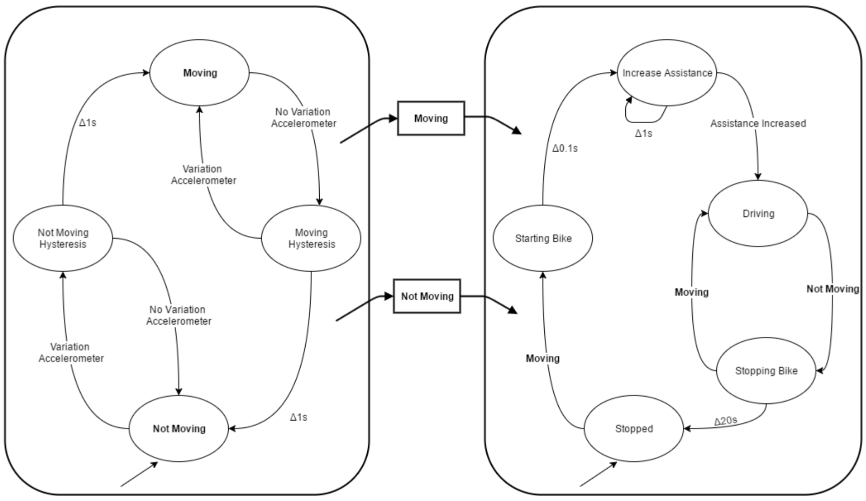


Figure 6.19: Interaction between accelerometer state machine and motor state machine.

Due to a *Non Disclosure Agreement* (NDA), the software developed cannot be more detailed.

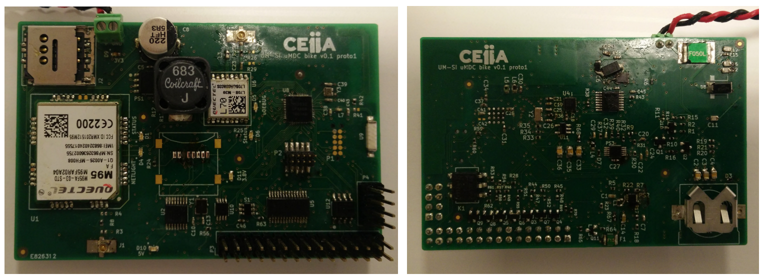
Chapter 7

Results

This chapter is dedicated to presenting some of the results of the tests performed during the project, which facilitate the understanding of the developed system. It is also made a small comparison between the system developed (uMDC) and the existing system (MDC).

7.1 uMDC

The first uMDC prototype was developed with focus in electric bicycle. In figure 7.1 is shown the hardware result.

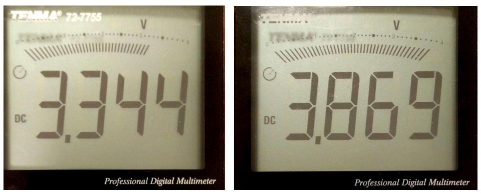


(a) (b)

Figure 7.1: a) uMDC top view. b) uMDC bottom view.

Due to the complexity of the system, and, in order to detect any short circuit that may occur, the validation of the components was made one by one.

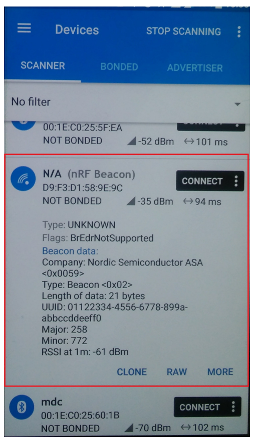
Initially it were validated the power supplies, this because if they were not properly dimensioned they could burn the whole circuit. In figure 7.2 is illustrated the output voltage obtained in the measurement of the different power supplies.



(a) (b)

Figure 7.2: a) System Power Supply. b) GSM Power Supply.

After that it was tested the microcontroller, in particular the bluetooth PCB antenna. Using the programming pins, the microcontroller was flashed with a beacon example code from Nordic SDK, and, using the Nordic smartphone application, it was detected (figure 7.3). This test proved that the micro-controller, in particular the PCB antenna, is working as expected.



00:1E:C0:25:5F:EA
NOT BONDED -52 dBm ↔101 ms

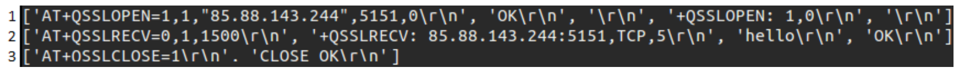
N/A (nRF Beacon)
D9:F3:D1:58:9E:9C
NOT BONDED -35 dBm ↔94 ms

Type: UNKNOWN
Flags: BrEdrNotSupported
Beacon data:
Company: Nordic Semiconductor ASA <0x0059>
Type: Beacon <0x02>
Length of data: 21 bytes
UUID: 01122334-4556-6778-899a-abbccddeeff0
Major: 258
Minor: 772
RSSI at 1m: -61 dBm

mdc
00:1E:C0:25:60:1B
NOT BONDED -70 dBm ↔102 ms

Figure 7.3: uMDC detected by Nordic smartphone application.

To test the GSM module, as represented in figure 7.4, it was made a connection to a remote server (line 1). After the connection established, the server was programmed to send a "hello" message and as shown in line 2, that message was successfully received. In the end the connection with the server was closed (line 3).



```
1 ['AT+QSSLOPEN=1,1,"85.88.143.244",5151,0\r\n', 'OK\r\n', '\r\n', '+QSSLOPEN: 1,0\r\n', '\r\n']
2 ['AT+QSSLRECV=0,1,1500\r\n', '+QSSLRECV: 85.88.143.244:5151,TCP,5\r\n', 'hello\r\n', 'OK\r\n']
3 ['AT+QSSLCLOSE=1\r\n', 'CLOSE OK\r\n']
```

Figure 7.4: Quectel M95 SSL test.

The GPS module didn't need any configurations, because immediately after power up the module, it started sending *National Marine Electronics Association* (NMEA) messages. In the figure 7.5 are the coordinates retrieved from GPS (small red dots near de car).

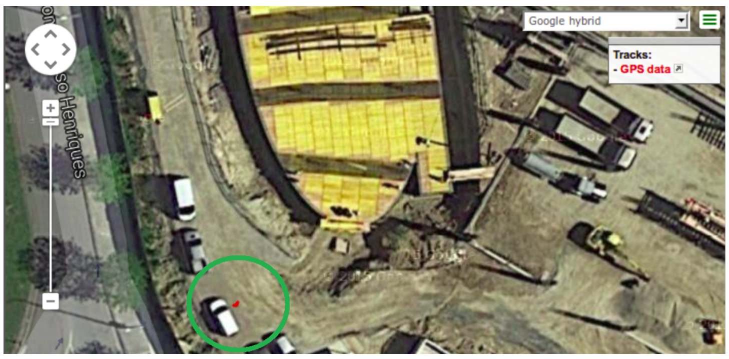


Figure 7.5: Points retrieved from GPS.

All the other components as, accelerometer, LED controller and relay were tested during the development of the first uMDC prototype software.

To demonstrate the interaction of the uMDC with the electric bicycle a video was filmed and it is available at:

https://www.youtube.com/watch?v=xCf1v5ZYodE&feature=youtu.be

7.2 uMDC Development Board

Before the development of the first uMDC prototype, a development board with all the components plus CAN interface (figure 7.6) was found. This board was developed without any size or layers restrictions and its main objective was to test all the components individually, allowing to change components or footprints that were not properly implemented or designed.

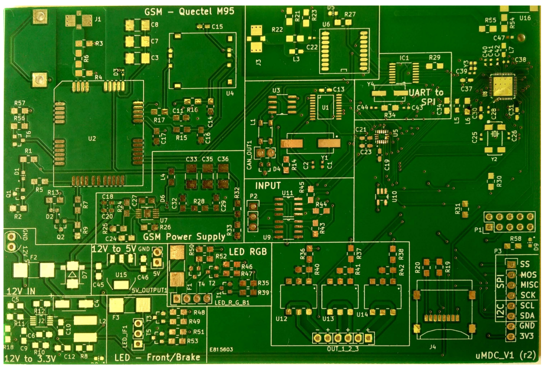


Figure 7.6: uMDC development board.

Once the microcontroller does not have an integrated CAN controller, it was necessary to apply an external solution.

For this purpose, the Microchip MCP2515, a stand-alone CAN controller that implements the CAN specification, version 2.0B, was used. It is capable of transmitting and receiving both standard and extended data and remote frames. The MCP2515 has two acceptance masks and six acceptance filters that are used to filter out unwanted messages, thereby reducing the host MCUs overhead. This controller can run from a supply between 2.7V to 5.5V and interface with microcontrollers via SPI [50].

Due to some recommendations as the 3.3V operating voltage and the common mode higher than 24V (in European trucks, the CAN bus works at 24V), the only solution found for CAN transceiver was the Linear LTC2875. It is a high speed, low power CAN transceiver operating on 3.3V or 5V supplies that features $\pm 60V$ overvoltage fault protection on the data transmission lines during all modes of operation, including power-down. Extended $\pm 36V$ input common mode range and high common mode rejection on the CAN receiver provides tolerance of large ground loop voltages [51].

Since the integration of the uMDC in electric bycicles doesn't use CAN protocol, this solution wasn't implemented in the developed prototype.

7.3 uMDC vs MDC

One of the requirements of the uMDC was to be much smaller and thiner than the MDC.

In figure 7.7 are represented both system. As it is possible to observe, the size difference between both systems is considerable, which indicates that the requirement has been reached.

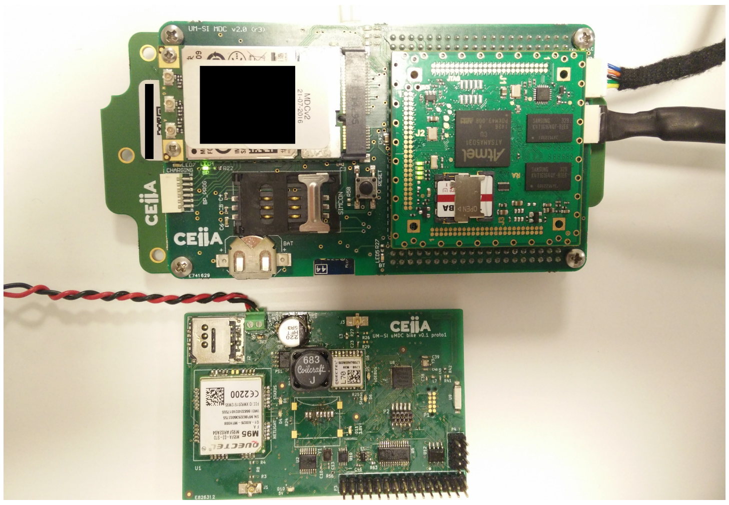
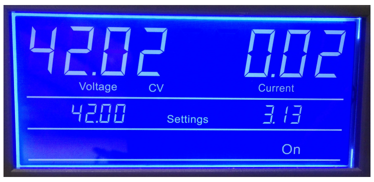
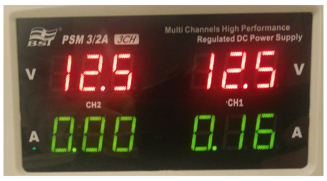


Figure 7.7: uMDC vs MDC size comparison.

Other requirement of the uMDC was to consume less than 2 W while fully running and, with a simple test, where almost every components were turned on, the uMDC didn't exceed 1 W. It was also made a comparison between the current consumption of both system which are available in figure 7.8.



(a)



(b)

Figure 7.8: a) uMDC current consumption. b) MDC current consumption.

The measurements for the MDC and the uMDC power consumptions were not performed under the same conditions. Thus, the MDC power consumption should be considered only as a comparison base to evaluate the measurements taken for the uMDC.

Chapter 8

Conclusions and Future Work

The present thesis aimed to develop a small and robust device, capable of controlling and tracking different types of vehicles, which led to the development of uMDC.

This chapter will summarize the balance between the initial requirements and the final results of the project, as well as the main issues faced during the project. Moreover, suggestions will be made about future work to be performed by CEiiA.

Regarding the uMDC, the results were very good, achieving almost all the requirements. In the near future the first prototype will be running at shared bicycles project and it will be used to lock/unlock, control the motor assistance and the presence/stop lights, communicate with the cloud reporting events and data (GPS coordinates, state, battery level, etc.).

Initially it was required that the system should incorporate the CAN bus communication, which happened on the uMDC development board. However, the components dedicated to that function were not tested, given main focus were the electric bicycles, that does not include CAN bus communication. Thus, it was decided not to incorporate them in the first prototype.

Although all components have been tested individually and are all operational, it will be necessary to make some minor changes. Since the bluetooth PCB antenna works properly, the ceramic antenna can be removed. To avoid external components a GSM PCB antenna and a GPS ceramic antenna could be implemented. It is also important to implement CAN protocol components like in the uMDC development board, if need for future projects.

This internship opportunity provided by CEiiA, contributed to consolidate all knowledge acquired during my academic path. The accomplishment of this project, along with the contact with some suppliers and the exchange of ideas with other company employees, had a direct impact on my professional growth.

In conclusion, I would like to demonstrate gratitude to CEiiA for accepting me and providing a very enriching experience in many ways. It contributed to the acquisition of new skills, which will certainly be useful along my entire career,professional and personal.

Bibliography

[1] VERIZON - 3 Reasons Why the Time for Connected Vehicles is Now. Accessed in February of 2016. Available in: `http://www.verizon.com/about/news/connected-car-vehicle-sharing-economy/`.

[2] TechCrunch - The Cloud-Connected Car Drives IoT Monetization. Accessed in February of 2016. Available in: `https://techcrunch.com/2015/10/20/the-cloud-connected-car-drives-iot-monetization/`.

[3] A Caixa Negra - Sobre Nós. Accessed in March of 2016. Available in: `http://www.acaixanegra.com/works/ceiia/`.

[4] CEiiA; "Code of Ethics and Conduct".

[5] CEiiA - Mobility. Accessed in March of 2016. Available in: `http://www.ceiia.com/mobility`.

[6] CEiiA - Mobi.me. Accessed in March of 2016. Available in: `http://www.mobi-me.eu/MOBIME.pdf`.

[7] BikeEmotion - Bike Sharing Technology. Accessed in June of 2016. Available in: `http://www.bikeemotion.com/#aboutus`.

[8] Amigos D'Avenida - BikeEmotion assume-se além fronteiras. Accessed in June of 2016. Available in: `http://amigosdavenida.blogs.sapo.pt/870442.html`.

[9] Linkedin - Bewegen Technologies Inc.. Accessed in June of 2016. Available in: `https://www.linkedin.com/company/bewegen-technologies-inc-`.

[10] Bewegen - BEWEGEN TECHNOLOGIES. Accessed in June of 2016. Available in: `http://bewegen.com/about-us/`.

[11] Bewegen - OUR SYSTEM. Accessed in June of 2016. Available in: `http://bewegen.com/our-system/`.

[12] Connected Cycle - The connected bike enabling solution. Accessed in June of 2016. Available in: `http://connectedcycle.com/`

[13] Gobike - About Gobike. Accessed in June of 2016. Available in: `http://gobike.com/about/history-about-gobike/`.

[14] Gobike - The Bike. Accessed in June of 2016. Available in: `http://gobike.com/solution/the-bike/`.

[15] Bycyklen - The Bycykel. Accessed in June of 2016. Available in: `https://bycyklen.dk/en/the-bycykel/`.

[16] Gobike - Operator Administration System. Accessed in June of 2016. Available in: `http://gobike.com/solution/operator-administration-system/`.

[17] Linkedin - Inosat Global. Accessed in July of 2016. Available in: `https://www.linkedin.com/company/inosat-global`.

[18] INOSAT - Gestão de Frotas por GPS. Accessed in July of 2016. Available in: `http://www.inosat.pt/inofrota-biz`.

[19] Digital Doughnut - M2M & IoT Protocols. Accessed in July of 2016. Available in: `https://www.digitaldoughnut.com/articles/2016/april/m2m-iot-protocols-an-introduction`

[20] Embedded Computing - IoT requirements and protocols. Accessed in February of 2016. Available in: http://embedded-computing.com/articles/internet-things-requirements-protocols/.

[21] InfoWorld - MQTT. Accessed in February of 2016. Available in: http://www.infoworld.com/article/2972143/internet-of-things/real-time-protocols-for-iot-apps.html.

[22] RisingStack - Getting start with Node.js and MQTT. Accessed in February of 2016. Available in: http://www.hivemq.com/blog/how-to-get-started-with-mqtt.

[23] Linkdin - The IoT communication protocols. Accessed in February of 2016. Available in: https://www.linkedin.com/pulse/iot-communication-protocols-james-stansberry.

[24] HiveMQ - MQTT Essentials part 6. Accessed in February of 2016. Available in: http://www.hivemq.com/blog/mqtt-essentials-part-6-mqtt-quality-of-service-levels.

[25] SlideShare - Internet of Things protocols CoAP. Accessed in February of 2016. Available in: http://www.slideshare.net/vgholkar/io-t-protocolsoscon2014.

[26] An Overview of XMPP. Accessed in February of 2016. Available in: http://http://xmpp.org/about/technology-overview.html.

[27] Introduction to XMPP. Accessed in February of 2016. Available in: http://talks.evilprofessor.co.uk/intro-to-xmpp/#/5/2.

[28] SILVA, Pedro Samuel Nunes; "Caracterização experimental de comunicações Powerline em automóveis" 2009

[29] BOSCH; "CAN Specification, Version 2.0B" 1991.

[30] MICROCHIP; "Introduction to Control Area Network" 2012.

[31] NATIONAL INSTRUMENTS - Controller Area Network (CAN) Overview. Accessed in August of 2016. Available in: http://www.ni.com/white-paper/2732/en/.

[32] KVASER - CAN Protocol Tutorial. Accessed in August of 2016. Available in: https://www.kvaser.com/can-protocol-tutorial/.

[33] VECTOR - The LIN Standard. Accessed in August of 2016. Available in: http://vector.com/vi_local_interconnect_network_en.html.

[34] STMICROELECTRONICS; "LIN (LOCAL INTERCONNECT NETWORK) solutions" 2002.

[35] FORSBERG, Andreas; HEDBERG, Johan; "Comparison of FlexRay and CAN-bus for Real-Time Communication" 2009

[36] SHAWN, Robert; JACKMAN, Brendan Brendan Jackman; "An Introduction to FlexRay as an Industrial Network" 2008

[37] NATIONAL INSTRUMENTS - FlexRay Automotive Communication Bus Overview. Accessed in August of 2016. Available in: http://www.ni.com/white-paper/3352/en/#toc2.

[38] NORDIC - nRF52832. Accessed in August of 2016. Available in: http://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF52832.

[39] QUECTEL - GSM/GPRS M95. Accessed in August of 2016. Available in: http://www.quectel.com/product/prodetail.aspx?id=7.

[40] QUECTEL - GPS L70-R. Accessed in August of 2016. Available in: http://www.quectel.com/product/prodetail.aspx?id=69.

[41] NXP - MMA8453Q. Accessed in August of 2016. Available in: `http://www.nxp.com/products/sensors/accelerometers/3-axis-accelerometers/2g-4g-8g-low-g-10-bit-digital-accelerometer:MMA8453Q`.

[42] ATMEL - AT24MAC602. Accessed in August of 2016. Available in: `http://www.atmel.com/devices/AT24MAC602.aspx`.

[43] VISHAY, "VO14642AT, 1 Form A Solid State Relay" 2011.

[44] TEXAS INSTRUMENTS, "SLLS965C, DUAL DIGITAL ISOLATORS" 2012.

[45] NXP - PCA9685PW. Accessed in August of 2016. Available in: `http://www.nxp.com/products/power-management/lighting-driver-and-controller-ics/i2c-led-display-control/16-channel-12-bit-pwm-fm-plus-ic-bus-led-controller:PCA9685PW`.

[46] ALLEGRO, "A8498, Wide Input Voltage 3.0 A Step Down Regulator" 2016.

[47] NORDIC DEVELOPER ZONE - General PCB design guidelines for nRF52. Accessed in May of 2016. Available in: `https://devzone.nordicsemi.com/blogs/870/general-pcb-design-guidelines-for-nrf52/`.

[48] Techopedia - C++ Programming Language. Accessed in September of 2016. Available in: `https://www.techopedia.com/definition/26184/c-programming-language`.

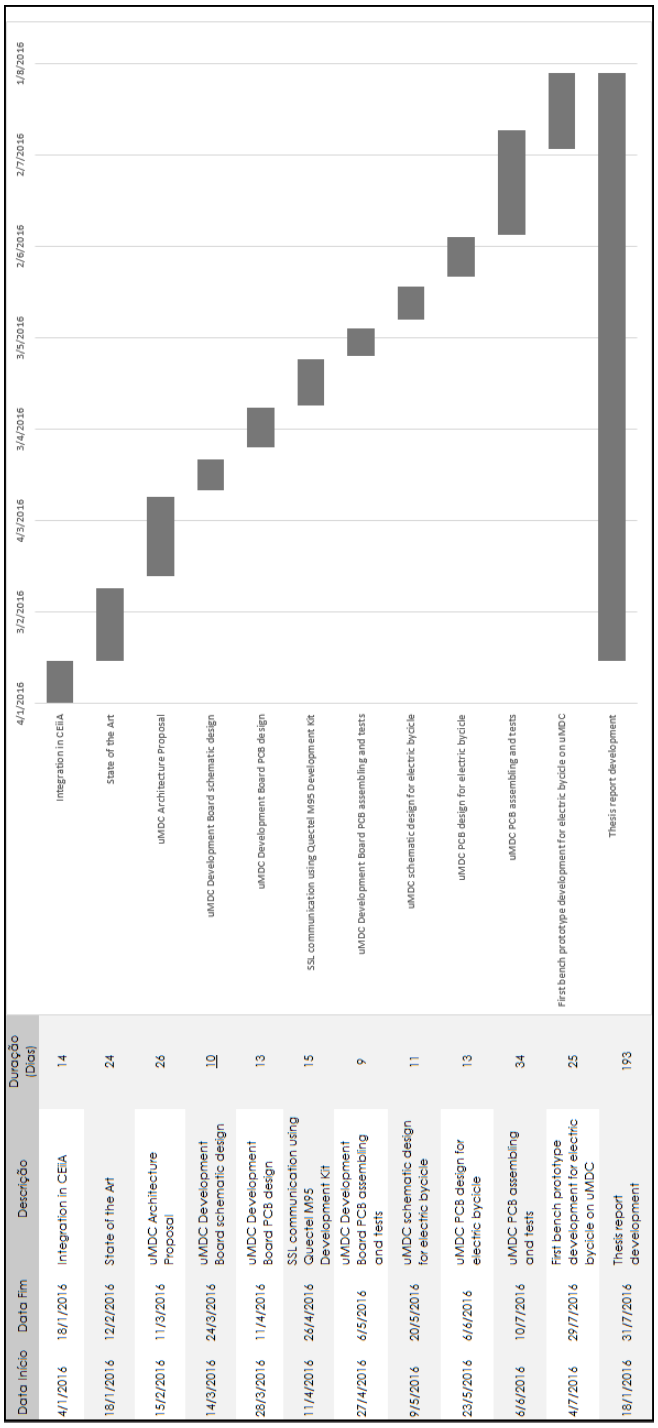
[49] VERMA, Ankur; TUCKER, John; "Creating GSM/GPRS Power Supply from TPS54260" 2010.

[50] MICROCHIP, "MCP2515, Stand-Alone CAN Controller With SPI Interface" 2005.

[51] LINEAR - LTC2875. Accessed in August of 2016. Available in: http://www.linear.com/product/LTC2875.

Appendice A. Timeline

Table 1: Project Schedule.

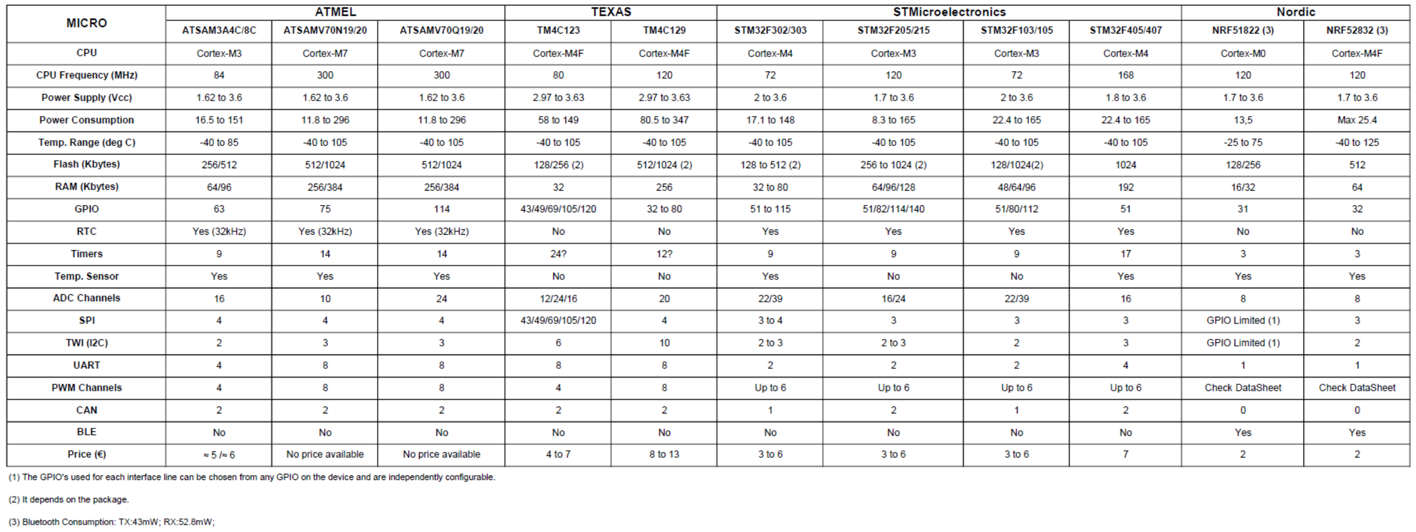

4/1/2016
3/2/2016
4/3/2016
3/4/2016
3/5/2016
2/6/2016
2/7/2016
1/8/2016

Data Início	Data Fim	Descrição	Duração (Dias)
4/1/2016	18/1/2016	Integration in CEiiA	14
18/1/2016	12/2/2016	State of the Art	24
15/2/2016	11/3/2016	uMDC Architecture Proposal	26
14/3/2016	24/3/2016	uMDC Development Board schematic design	10
28/3/2016	11/4/2016	uMDC Development Board PCB design	13
11/4/2016	26/4/2016	SSL communication using Quectel M95 Development Kit	15
27/4/2016	6/5/2016	uMDC Development Board PCB assembling and tests	9
9/5/2016	20/5/2016	uMDC schematic design for electric bycicle	11
23/5/2016	6/6/2016	uMDC PCB design for electric bycicle	13
6/6/2016	10/7/2016	uMDC PCB assembling and tests	34
4/7/2016	29/7/2016	First bench prototype development for electric bycicle on uMDC	25
18/1/2016	31/7/2016	Thesis report development	193

This page intentionally left blank.

Appendice B. Components Comparison

Table 2: Microcontrollers specifications.



MICRO	ATMEL			TEXAS		STMicroelectronics				Nordic	
	ATSAM3A4C/8C	ATSAMV70N19/20	ATSAMV70Q19/20	TM4C123	TM4C129	STM32F302/303	STM32F205/215	STM32F103/105	STM32F405/407	NRF51822 (3)	NRF52832 (3)
CPU	Cortex-M3	Cortex-M7	Cortex-M7	Cortex-M4F	Cortex-M4F	Cortex-M4	Cortex-M3	Cortex-M3	Cortex-M4	Cortex-M0	Cortex-M4F
CPU Frequency (MHz)	84	300	300	80	120	72	120	72	168	120	120
Power Supply (Vcc)	1.62 to 3.6	1.62 to 3.6	1.62 to 3.6	2.97 to 3.63	2.97 to 3.63	2 to 3.6	1.7 to 3.6	2 to 3.6	1.8 to 3.6	1.7 to 3.6	1.7 to 3.6
Power Consumption	16.5 to 151	11.8 to 296	11.8 to 296	58 to 149	80.5 to 347	17.1 to 148	8.3 to 165	22.4 to 165	22.4 to 165	13,5	Max 25.4
Temp. Range (deg C)	-40 to 85	-40 to 105	-40 to 105	-40 to 105	-40 to 105	-40 to 105	-40 to 105	-40 to 105	-40 to 105	-25 to 75	-40 to 125
Flash (Kbytes)	256/512	512/1024	512/1024	128/256 (2)	512/1024 (2)	128 to 512 (2)	256 to 1024 (2)	128/1024(2)	1024	128/256	512
RAM (Kbytes)	64/96	256/384	256/384	32	256	32 to 80	64/96/128	48/64/96	192	16/32	64
GPIO	63	75	114	43/49/69/105/120	32 to 80	51 to 115	51/82/114/140	51/80/112	51	31	32
RTC	Yes (32kHz)	Yes (32kHz)	Yes (32kHz)	No	No	Yes	Yes	Yes	Yes	No	No
Timers	9	14	14	24?	12?	9	9	9	17	3	3
Temp. Sensor	Yes	Yes	Yes	No	No	Yes	No	No	Yes	Yes	Yes
ADC Channels	16	10	24	12/24/16	20	22/39	16/24	22/39	16	8	8
SPI	4	4	4	43/49/69/105/120	4	3 to 4	3	3	3	GPIO Limited (1)	3
TWI (I2C)	2	3	3	6	10	2 to 3	2 to 3	2	3	GPIO Limited (1)	2
UART	4	8	8	8	8	2	2	2	4	1	1
PWM Channels	4	8	8	4	8	Up to 6	Up to 6	Up to 6	Up to 6	Check DataSheet	Check DataSheet
CAN	2	2	2	2	2	1	2	1	2	0	0
BLE	No	No	No	No	No	No	No	No	No	Yes	Yes
Price (€)	≈ 5 /≈ 6	No price available	No price available	4 to 7	8 to 13	3 to 6	3 to 6	3 to 6	7	2	2

(1) The GPIO's used for each interface line can be chosen from any GPIO on the device and are independently configurable.

(2) It depends on the package.

(3) Bluetooth Consumption: TX:43mW; RX:52.8mW;

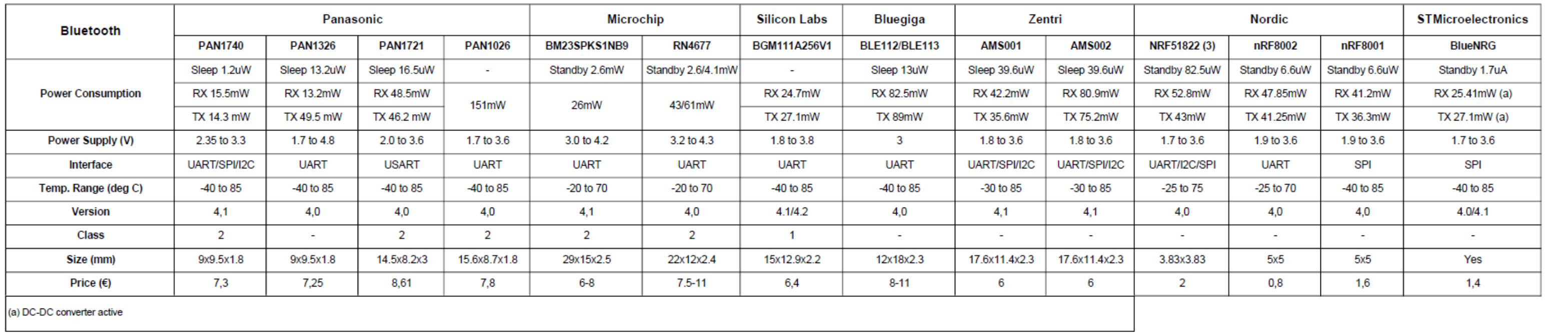
Table 3: GPS modules specifications.

GPS	Origin GPS		GlobalTop GPS		Maestro	Quectel		
	Nano Hornet	Multi Micro Hornet	LadyBird 1 e 3	Titan 3	A2135-H / A2235-H	L70	L80	L86
Power Consumption	Standby 15mW	Standby 15mW	Standby 1.63 mA	Standby 493 uA	27uA (Hibernate)	Standby 200uA	Standby 1mA	Standby 1mA
	Acquisition 40~47 mA	Acquisition 24~32 mA	Acquisition 22~37 mA	Acquisition 29~37 mA	Acquisition 56~69 mA	Acquisition 18 mA	Acquisition 25 mA	Acquisition 26 Ma (86mW)
	Tracking 6~40 mA	Tracking 24mA	Tracking 19~36 mA	Tracking 24~36 mA	Tracking 31 mA	Tracking 12 mA	Tracking 20 mA	Tracking 22 mA (72mW)
Accuracy	< 2.5m	< 2.5m	< 3m	< 3m	< 2.5	< 2.5	< 2.5	< 2.5
TTFF	Cold 35s	Cold 31s	Cold < 35s	Cold < 35s	Cold < 38s	Cold 15s	Cold 35s	Cold 15s
	Hot 1s	Hot 1s	Hot < 1s	Hot < 1s	Hot < 1s	Hot 1s	Hot 1s	Hot 1s
A.GPS	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Satelite	GPS	GPS/GLONASS	GPS	GPS/GLONASS	GPS	GPS	GPS	GPS/GLONASS
Internal Antenna	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes
Temp. Range (deg C)						-40 to 85	-40 to 85	-40 to 85
Power Supply (V)	1.71 to 1.89 (PM01)	3.0 to 3.6	3.0 to 4.3	3.0 to 4.3	3.0 to 3.6	2.8 to 4.3	3.0 to 4.3	3.0 to 4.3
	2 to 5.5(PM04)							
Size (mm)	-	-	-	-	-	10.1x9.7x2.5	16x16x6.45	16x16x6.45
Price (€)	No price available	No price available	No price available	No price available	14 to 15	6	8	8,5

Table 4: GSM modules specifications.

GSM	SIMCOM				Quectel
	SIM800C	**SIM808**	**SIM900**	**SIM908**	M95
Power Consumption	1W transmitting	1W transmitting	1W transmitting	1W transmitting	1W transmitting
	4mW Low Power	4mW Low Power	4mW Low Power	4mW Low Power	4.3mW Low Power
GPS Consumption	-	Acquisition 42mA	-	Acquisition 77mA	-
	-	Tracking 24mA	-	Tracking 76mA	-
Power Supply (V)	3.4 to 4.4	3.4 to 4.4	3.4 to 4.4	3.2 to 4.8	3.3 to 4.6
Interface	Serial	Serial/SPI(optional)	I2C/SPI/Serial	Serial/SPI	Serial
Temp. Range (deg C)	-40 to 85	-40 to 85	-30 to 80	-40 to 85	-40 to 85
GPS	No	Yes	No	Yes	No
Bluetooth	3,0	3,0	No	No	No
Price (€)	8	17	11 to 18	21,8	6 to 9

Table 5: Bluetooth modules specifications.



Bluetooth	Panasonic				Microchip		Silicon Labs	Bluegiga	Zentri		Nordic			STMicroelectronics
	PAN1740	PAN1326	PAN1721	PAN1026	BM23SPKS1NB9	RN4677	BGM111A256V1	BLE112/BLE113	AMS001	AMS002	NRF51822 (3)	nRF8002	nRF8001	BlueNRG
Power Consumption	Sleep 1.2uW	Sleep 13.2uW	Sleep 16.5uW	-	Standby 2.6mW	Standby 2.6/4.1mW	-	Sleep 13uW	Sleep 39.6uW	Sleep 39.6uW	Standby 82.5uW	Standby 6.6uW	Standby 6.6uW	Standby 1.7uA
	RX 15.5mW	RX 13.2mW	RX 48.5mW	151mW	26mW	43/61mW	RX 24.7mW	RX 82.5mW	RX 42.2mW	RX 80.9mW	RX 52.8mW	RX 47.85mW	RX 41.2mW	RX 25.41mW (a)
	TX 14.3 mW	TX 49.5 mW	TX 46.2 mW				TX 27.1mW	TX 89mW	TX 35.6mW	TX 75.2mW	TX 43mW	TX 41.25mW	TX 36.3mW	TX 27.1mW (a)
Power Supply (V)	2.35 to 3.3	1.7 to 4.8	2.0 to 3.6	1.7 to 3.6	3.0 to 4.2	3.2 to 4.3	1.8 to 3.8	3	1.8 to 3.6	1.8 to 3.6	1.7 to 3.6	1.9 to 3.6	1.9 to 3.6	1.7 to 3.6
Interface	UART/SPI/I2C	UART	USART	UART	UART	UART	UART	UART	UART/SPI/I2C	UART/SPI/I2C	UART/I2C/SPI	UART	SPI	SPI
Temp. Range (deg C)	-40 to 85	-40 to 85	-40 to 85	-40 to 85	-20 to 70	-20 to 70	-40 to 85	-40 to 85	-30 to 85	-30 to 85	-25 to 75	-25 to 70	-40 to 85	-40 to 85
Version	4,1	4,0	4,0	4,0	4,1	4,0	4.1/4.2	4,0	4,1	4,1	4,0	4,0	4,0	4.0/4.1
Class	2	-	2	2	2	2	1	-	-	-	-	-	-	-
Size (mm)	9x9.5x1.8	9x9.5x1.8	14.5x8.2x3	15.6x8.7x1.8	29x15x2.5	22x12x2.4	15x12.9x2.2	12x18x2.3	17.6x11.4x2.3	17.6x11.4x2.3	3.83x3.83	5x5	5x5	Yes
Price (€)	7,3	7,25	8,61	7,8	6-8	7.5-11	6,4	8-11	6	6	2	0,8	1,6	1,4

(a) DC-DC converter active

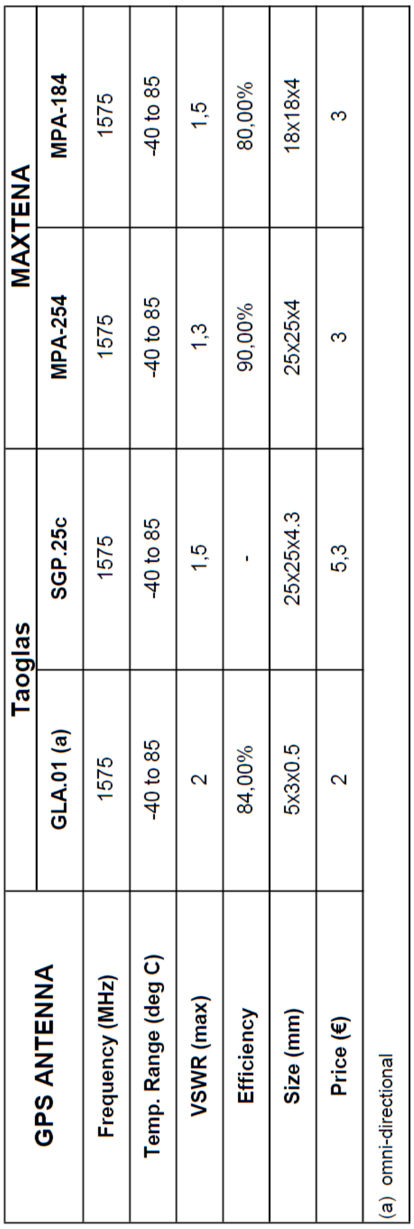
Table 6: CAN transceivers specifications.

CAN Transceiver	Linear	Intersil
	LTC2875	**ISL32452E/57E**
Power Supply (V)	3.3 to 5	3 to 5.5
I/O (V)	-60 to 60	-80 to 80
Common Mode Range	-36 to 36	-20 to 20
Temp. Range (deg C)	-55 to 125	-40 to 85
Size (mm)	3x3	6 x 5
Price (€)	2	2,3

Table 7: CAN controller specifications.

CAN Controller	MICROCHIP
	MCP2515
Power Supply (V)	2.7 to 5.5
Interface	SPI
Temp. Range (deg C)	-40 to 125
Size (mm)	11.5x10.3
Price (€)	2

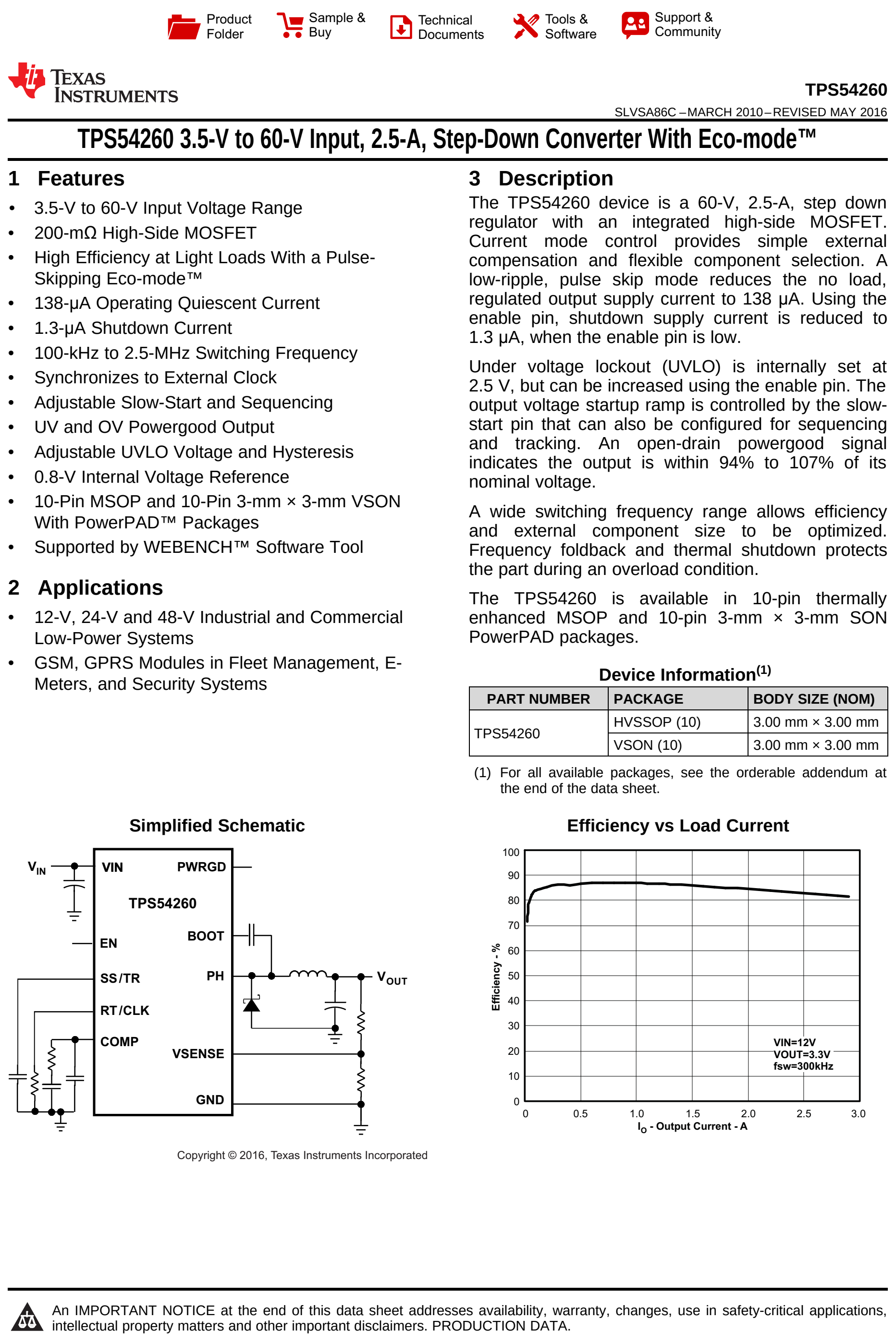
Table 8: GPS antennas.



GPS ANTENNA	Taoglas		MAXTENA	
	GLA.01 (a)	SGP.25c	MPA-254	MPA-184
Frequency (MHz)	1575	1575	1575	1575
Temp. Range (deg C)	-40 to 85	-40 to 85	-40 to 85	-40 to 85
VSWR (max)	2	1,5	1,3	1,5
Efficiency	84,00%	-	90,00%	80,00%
Size (mm)	5x3x0.5	25x25x4.3	25x25x4	18x18x4
Price (€)	2	5,3	3	3

(a) omni-directional

Appendice C. Texas Instruments TPS54260 Efficiency vs Current



Product Folder | Sample & Buy | Technical Documents | Tools & Software | Support & Community

TEXAS INSTRUMENTS

TPS54260

SLVSA86C – MARCH 2010 – REVISED MAY 2016

TPS54260 3.5-V to 60-V Input, 2.5-A, Step-Down Converter With Eco-mode™

1 Features

- 3.5-V to 60-V Input Voltage Range
- 200-mΩ High-Side MOSFET
- High Efficiency at Light Loads With a Pulse-Skipping Eco-mode™
- 138-μA Operating Quiescent Current
- 1.3-μA Shutdown Current
- 100-kHz to 2.5-MHz Switching Frequency
- Synchronizes to External Clock
- Adjustable Slow-Start and Sequencing
- UV and OV Powergood Output
- Adjustable UVLO Voltage and Hysteresis
- 0.8-V Internal Voltage Reference
- 10-Pin MSOP and 10-Pin 3-mm × 3-mm VSON With PowerPAD™ Packages
- Supported by WEBENCH™ Software Tool

2 Applications

- 12-V, 24-V and 48-V Industrial and Commercial Low-Power Systems
- GSM, GPRS Modules in Fleet Management, E-Meters, and Security Systems

3 Description

The TPS54260 device is a 60-V, 2.5-A, step down regulator with an integrated high-side MOSFET. Current mode control provides simple external compensation and flexible component selection. A low-ripple, pulse skip mode reduces the no load, regulated output supply current to 138 μA. Using the enable pin, shutdown supply current is reduced to 1.3 μA, when the enable pin is low.

Under voltage lockout (UVLO) is internally set at 2.5 V, but can be increased using the enable pin. The output voltage startup ramp is controlled by the slow-start pin that can also be configured for sequencing and tracking. An open-drain powergood signal indicates the output is within 94% to 107% of its nominal voltage.

A wide switching frequency range allows efficiency and external component size to be optimized. Frequency foldback and thermal shutdown protects the part during an overload condition.

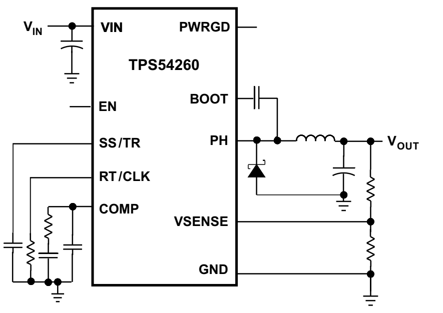
The TPS54260 is available in 10-pin thermally enhanced MSOP and 10-pin 3-mm × 3-mm SON PowerPAD packages.

Device Information(1)

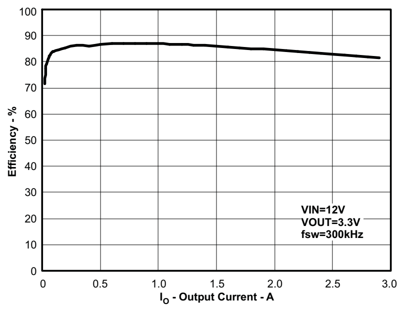
PART NUMBER	PACKAGE	BODY SIZE (NOM)
TPS54260	HVSSOP (10)	3.00 mm × 3.00 mm
	VSON (10)	3.00 mm × 3.00 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

Simplified Schematic



Efficiency vs Load Current



This page intentionally left blank.

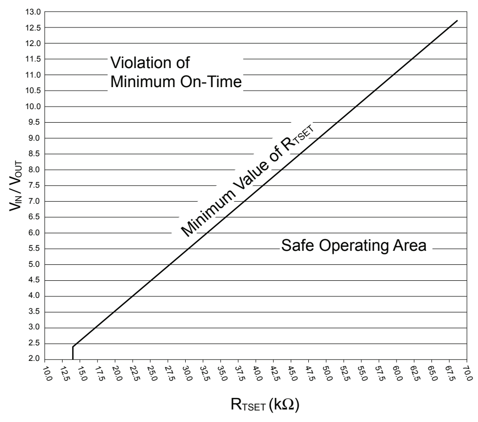
RTSET Selection

Correct selection of RTSET values will ensure that minimum on time of the switcher is not violated and prevent the switcher from cycle skipping. For a given V_{IN} to V_{OUT} ratio, the RTSET value must be greater than or equal to the value defined by the curve in the plot below.

Note. The curve represents the minimum RTSET value. When calculating R_{TSET}, be sure to use V_{IN}(max) / V_{OUT}(min). Resistor tolerance should also be considered, so that under no operating conditions the resistance on the TSET pin is allowed to go below the minimum value.

FB Resistor Selection

The impedance of the FB network should be kept low to improve noise immunity. Large value resistors can pick up noise generated by the inductor, which can affect voltage regulation of the switcher.

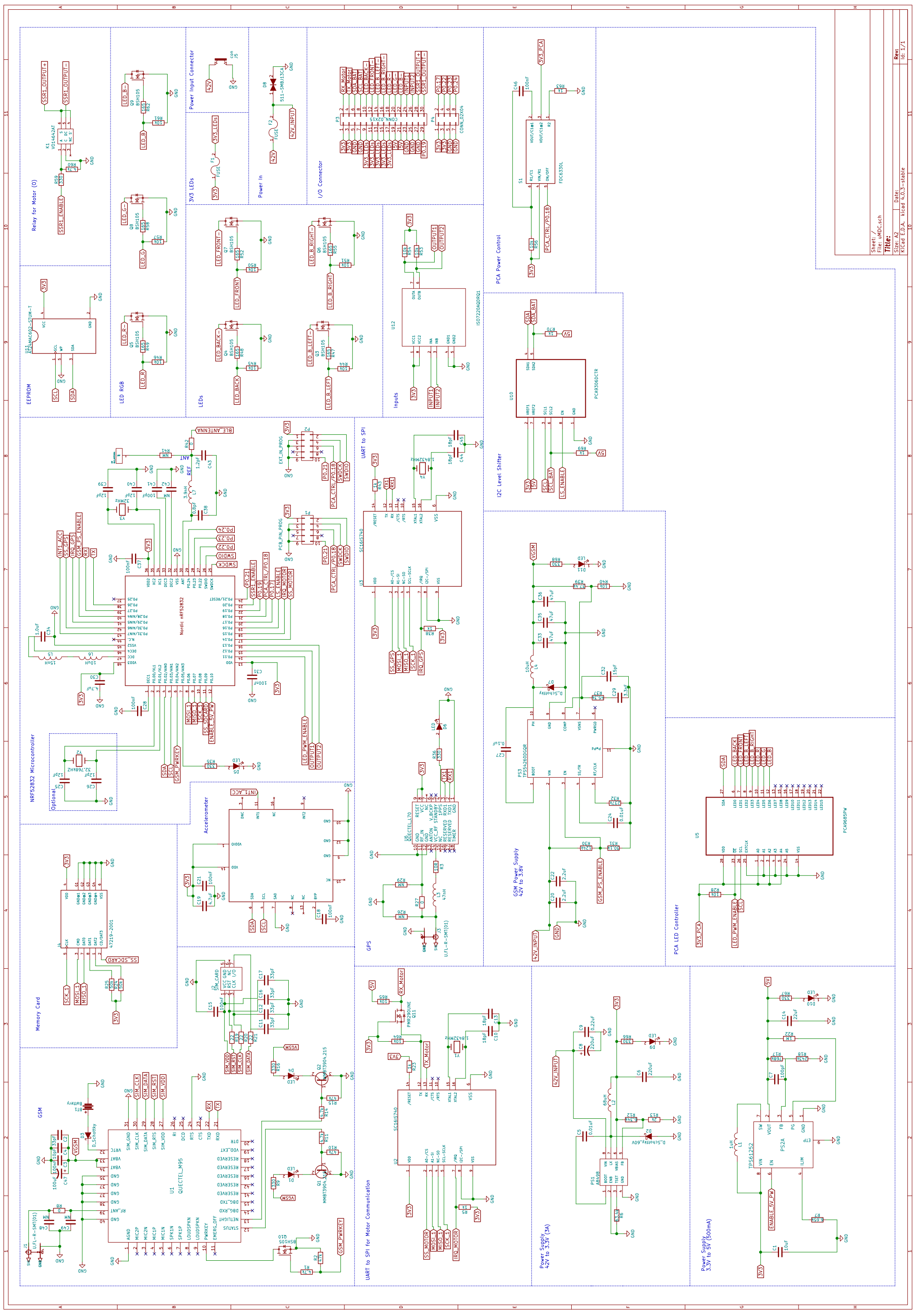


Recommended Components

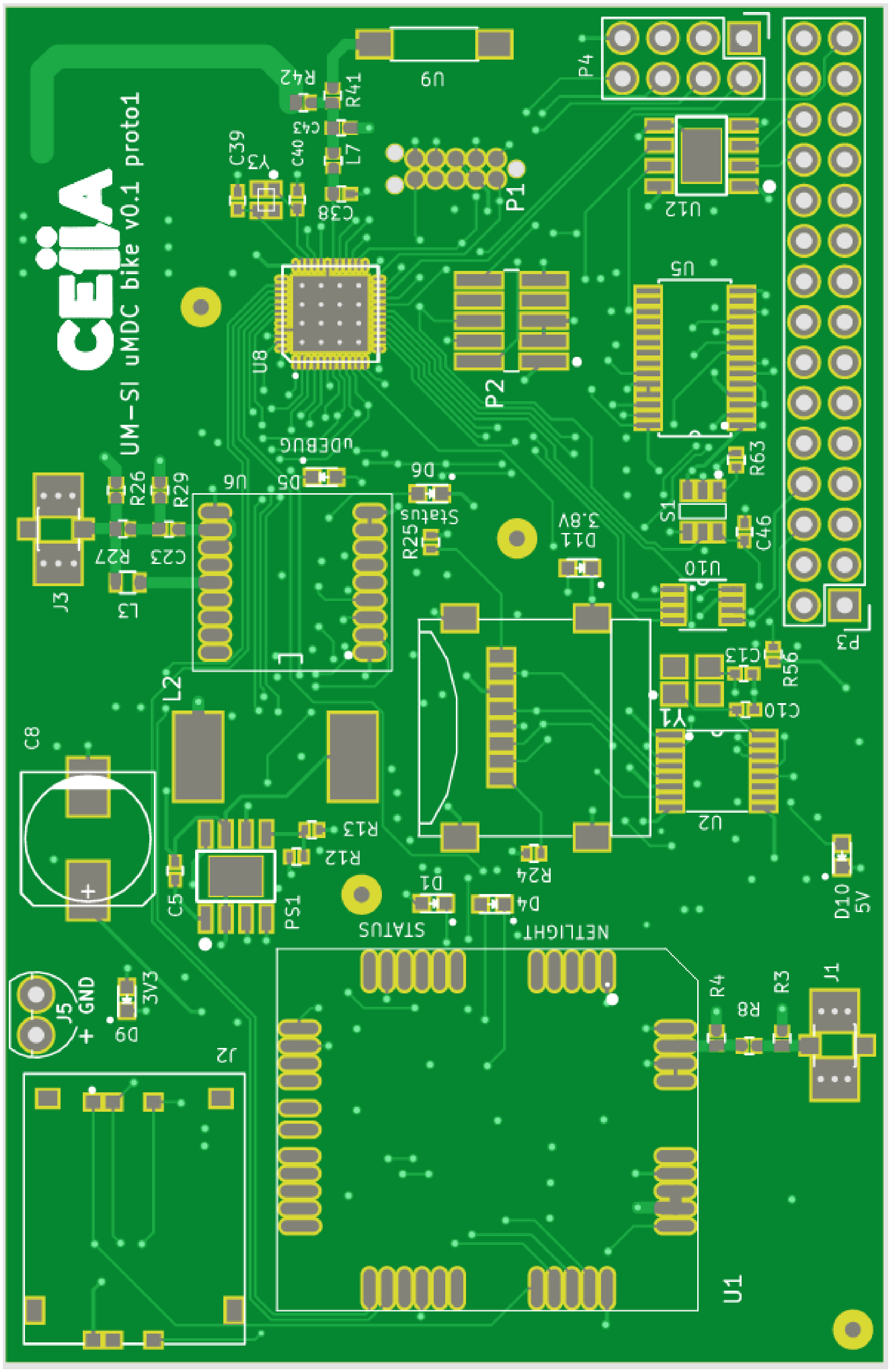
Component	V_{IN} = 42 V (Through Hole)		V_{IN} = 24 V (SMD)		V_{IN} = 12 V (SMD)	
	Description	**Part Number**	**Description**	**Part Number**	**Description**	**Part Number**
Inductor	Sumida, 68 µH	RCH1216BNP-680K	47 µH, 53 mΩ, 3.9 A, ±20%	CDRH127/LDNP-470MC	33 µH, 53 mΩ, 3.9 A, ±20%	CDRH127/LDNP-330MC
Diode	NIEC Schottky Barrier, 60 V, TO-252AA	NSQ03A06	Schottky, 30V, 3A, SMA	B330	Schottky, 20 V, 3 A, SMA	B320
CBOOT	Ceramic X7A, 0.01 µF, 100 V	Generic	Ceramic, X7R, ±10%, 0.01 µF / 50 V	C0603C103K5RACTU (Kemet)	Ceramic, X7R, ±10%, 0.01 µF / 50 V	C0603C103K5RACTU (Kemet)
CIN1	Ceramic X7A, 0.22 µF, 50 V	Generic	Ceramic, X7R, ±10%, 0.1 µF / 50 V	GRM188R71H104KA93D (Murata)	Ceramic, X7R, ±10%, 0.1 µF / 50 V	GRM188R71H104KA93D (Murata)
CIN2	Rubycon ZL, 220 µF, 50 V	50-ZL-220-M-10 X 16	Aluminum electrolytic, 35 V / 82 µF, 930 mA ripple current	35V-ZAV-820-8 X 12 (two)	Aluminum electrolytic, 35 V / 82 µF, 930 mA ripple current	35V-ZAV-820-8 X 12 (two)
COUT	Rubycon ZL, 220 µF, 25 V	25-ZL-220-M-8 X 11.5	Aluminum electrolytic , 6.3 V / 330 µF, 450 mA ripple current	EEVFC0J331P (Panasonic)	Aluminum electrolytic, 6.3 V / 330 µF, 450 mA ripple current	EEVFC0J331P (Panasonic)
R1	2.55 kΩ at VOUT = 1.8 V 6.34 kΩ at VOUT = 3.3 V 10.5 kΩ at VOUT = 5.0 V		2.55 kΩ at VOUT = 1.8 V 6.34 kΩ at VOUT = 3.3 V 10.5 kΩ at VOUT = 5.0 V		2.55 kΩ at VOUT = 1.8 V 6.34 kΩ at VOUT = 3.3 V 10.5 kΩ at VOUT = 5.0 V	
R2	2 kΩ		2 kΩ		2 kΩ	
R_{TSET}	63.4 kΩ		47.5 kΩ		35.2 kΩ	

This page intentionally left blank.

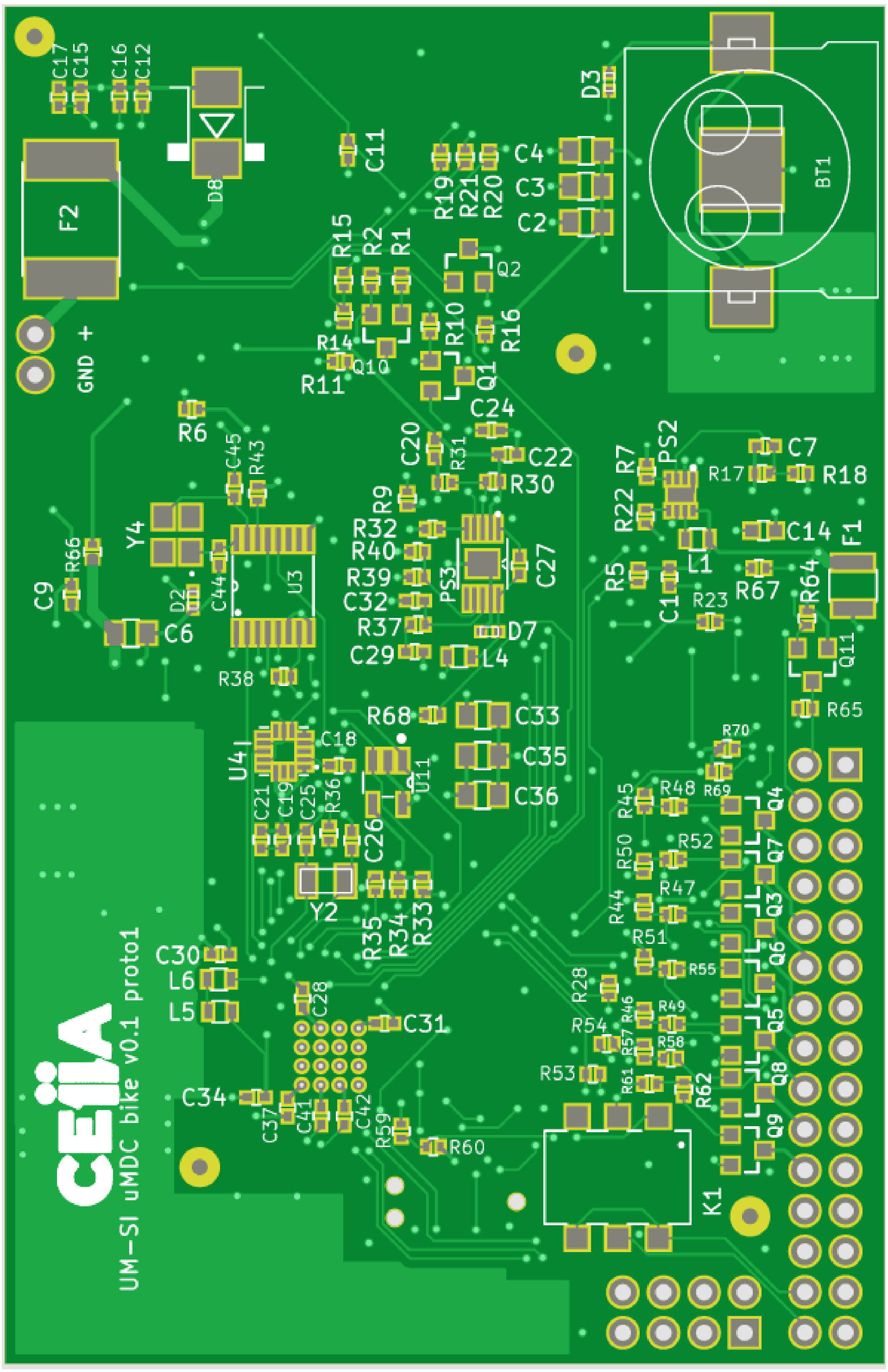
Appendice E. uMDC schematic and PCB layers



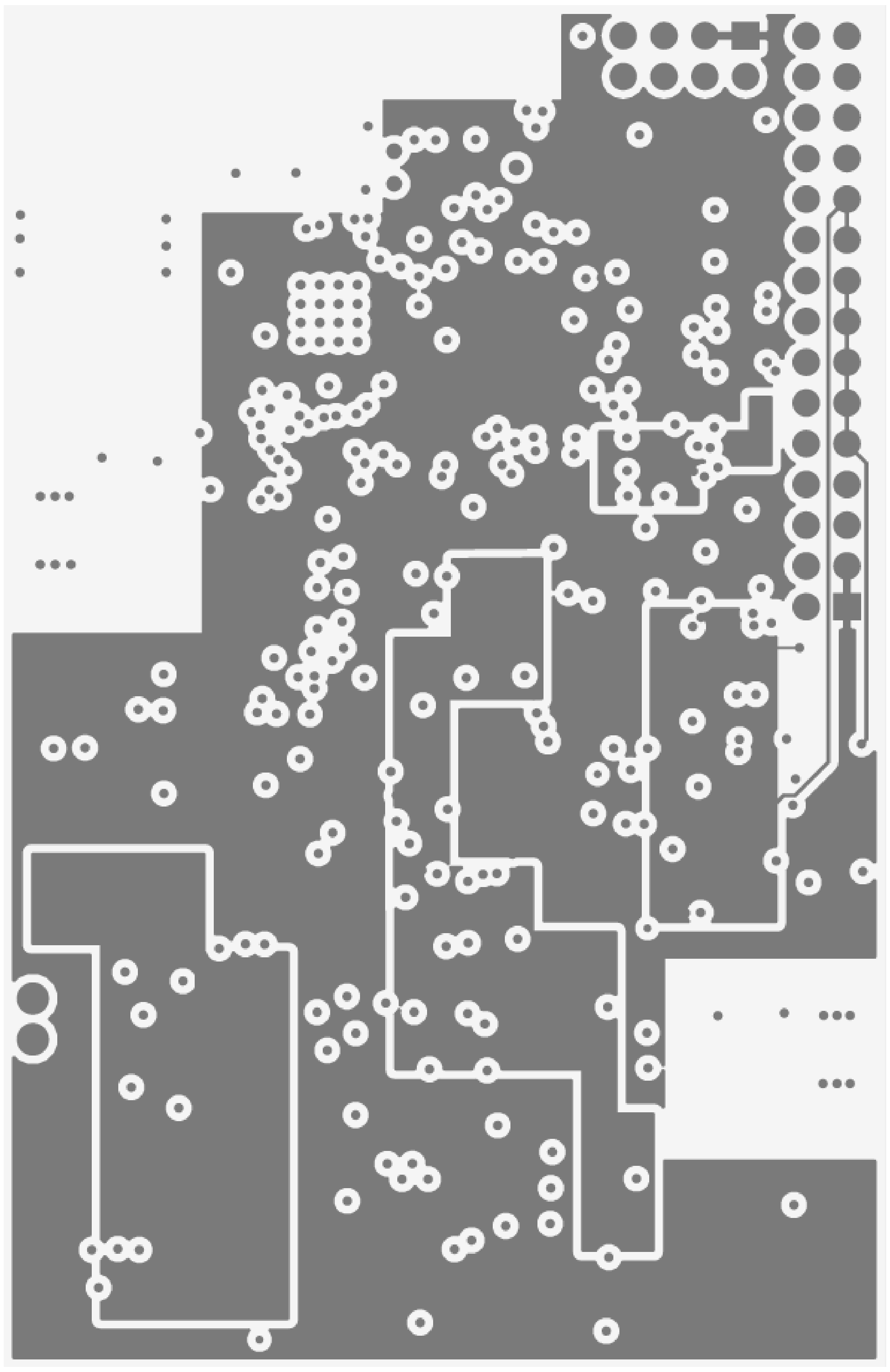
Top layer


UM-SI uMDC bike v0.1 proto1
CEiiA
U9
R42
R41
C43
C40
L7
C39
Y3
C38
P1
P4
U12
U5
U8
P2
R63
S1
C46
uDEBUG
D5
U6
D6
Status
R25
3.8V
D11
U10
R26
R29
R27
C23
J3
L3
C13
R56
C10
Y1
U2
L2
C8
R13
R12
PS1
C5
D1
D4
R24
STATUS
NETLIGHT
D10
5V
J5
GND
3V3
D9
J2
R4
R3
R8
J1
U1
P3

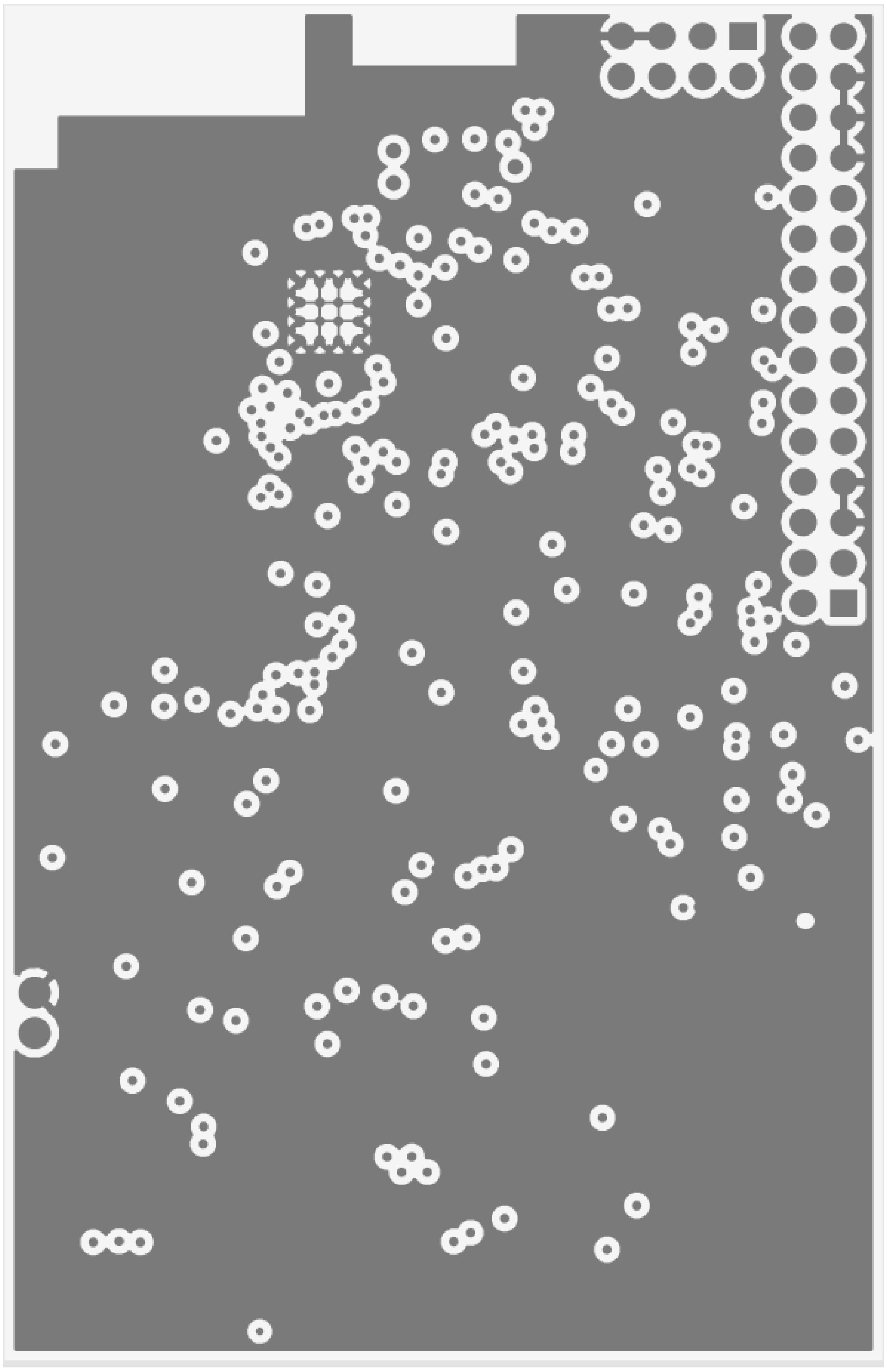
Bottom layer



Power layer



Ground layer



Appendice F. Printed Antenna for 2.45 GHz

WHITE PAPER

$\lambda/4$ printed monopole antenna for 2.45GHz

1. Preface

Taking the demand for small size, easy fabrication and low cost into account in the development of low-power radio devices for short-range 2.4GHz applications, a quarter wavelength monopole antenna implemented on the same printed circuit board as the radio module is a good solution. A printed quarter wavelength monopole antenna is very easy to design and can be tuned simply by slight changes in length.

This article presents basic guidelines on how to design such an antenna for use together with the 2.4GHz transceiver and transmitter devices from Nordic Semiconductor.

The described antenna should be fabricated on standard 1.6mm, low cost FR4 printed circuit board (PCB).

2. Basic properties of a quarterwave monopole antenna

A quarterwave monopole is a ground plane dependent antenna that must be fed single-ended. The antenna must have a ground plane to be efficient, and ideally the ground plane should spread out at least a quarter wavelength, or more, around the feed-point of the antenna. The size of the ground plane influences the gain, resonance frequency and impedance of the antenna.

The length of the monopole PCB trace mainly determines the resonant frequency of the antenna, but because of the very wide gain bandwidth of a quarterwave monopole, the antenna length is not too critical. But like any other antenna types, the gain of a quarterwave monopole will vary if parameters in the surroundings, such as case/box materials, distance to the ground plane, size of the ground plane, width and thickness of the PCB trace are varied. If any of these parameters are changed, a retuning of the monopole PCB trace length may be necessary for optimum performance in each application.

3. Determining the length of the printed monopole antenna

The antenna is fabricated on a standard 1.6mm FR4 substrate material with a typical dielectric constant ε_r of 4.4 at 2.45GHz.

The width of the monopole trace is W = 1.5mm. The wavelength in free air is λ_0 = 122mm. It may be approximated that the guided wavelength λ_g on the FR4 substrate is about

$$\lambda_g \sim 0.75 \cdot \lambda_0 = 0.75 \cdot 122\text{mm} \sim 92\text{mm}$$

The approximate, physical length of a printed quarterwave monopole antenna is then

$$L = 92\text{mm} / 4 = 23\text{mm}$$