Grado en Ingeniería Informática

Proyecto de Final de Carrera

Smart City Parking. Un Sistema de Detección de Vehículos Basado en Sensores del Campo Magnético

Autor:
Vladislav Rykov

Supervisor:
Manolo Gasch
Tutor Académico:
Germán León Navarro

Fecha de Presentación: 11 de Marzo del 2019
Año Académico 2018/2019

Resumen

Este documento contiene un trabajo final de grado en el cual fue desarrollado un sistema de detección de vehículos para aparcamiento inteligente. El sistema forma parte de una infraestructura IoT y cubre las primeras dos capas de su arquitectura, percepción y conectividad respectivamente. El proyecto fue llevado a cabo en la empresa IoTSens, la cual ofrece soluciones basadas en Internet de las Cosas recolectando, almacenando y analizando los datos en los ámbitos de industria inteligente, aguas residuales, medio ambiente y ciudades inteligentes. IoTSens es una división IoT del Grupo Gimeno en Castellón de la Plana.

El sistema desarrollado constituye una de las muchas soluciones existentes para ciudades inteligentes. En la era de optimización y procesamiento de datos la presente solución trae muchos beneficios, reduciendo tanto el tiempo de búsqueda de aparcamiento para un vehículo, como el dióxido de carbono producido por los vehículos en movimiento. El producto final es entregado al cliente en forma de Plataforma como Servicio (*PaaS*) donde los datos son recibidos, recolectados y visualizados.

Para el desarrollo de esta solución se ha llevado a cabo un ciclo completo de actividades del desarrollo de un nuevo producto tales como Investigación y Desarrollo (I+D), Procesamiento de Señales Digitales, diseño de algoritmos, programación de *firmware*, diseño de *hardware* y validación mediante pruebas y test.

Palabras clave

Internet of Things, Smart City, parking, vehicle detection algorithm, AMR sensors, embedded software, firmware, hardware development, RFLink, Texas Instruments

Resumen Extendido

Se puede ver que cada año más y más ciudades se hacen inteligentes siendo la mayoría de ellas ciudades metropolitanas como Madrid o Barcelona, las cuales poseen enormes infraestructuras de industria y turismo. La solución propuesta por este proyecto pretende mejorar en muchos aspectos las infraestructuras metropolitanas.

Primero, se optimizarán los aparcamientos municipales. El mejor lugar disponible para aparcar en un momento dado será indicado de tal modo que se ahorre el esfuerzo y tiempo de los conductores, y los recursos físicos propios de los vehículos. Por otra parte, se consigue que los lugares de aparcamiento se llenen de forma eficiente en términos de espacio que puede ser utilizado por entidades comerciales y municipales.

Segundo, se reducirán las congestiones en el tráfico. Menos vehículos estarán buscando un lugar para aparcar lo cual reduciría el flujo de tráfico.

Tercero, se reducirá la polución. Se ha estimado que las búsquedas de aparcamiento, en la escala mundial, gastan aproximadamente un millón de barriles de aceite por día. Esta solución optimizará la circulación decrementando el tiempo de búsqueda de aparcamiento y, por lo tanto, bajando la cantidad de emisiones y recursos consumidos diarios. Ésto, a su vez, disminuye la huella ambiental global.

Cuarto, se mejorará la experiencia del usuario porque se convertirá en una secuencia de pasos organizada y unificada. Dicha secuencia consta de los siguientes pasos:

1. El conductor paga por un lugar de aparcamiento
2. El lugar de aparcamiento es identificado
3. El conductor es dirigido al lugar de aparcamiento

Todo esto forma parte del proceso de llegada al destino.

Quinto, se introducirán nuevas formas innovadoras de ingresos públicos. El aparcamiento inteligente trae una posibilidad de introducir muchos flujos de ingresos públicos. Por ejemplo, los empresarios encargados de lugares de aparcamiento pueden integrar una capa de servicio que ofrece una opción de pago dependiendo de la posición del aparcamiento. Los programas y bonos de recompensa podrían ser incorporados en modelos existentes para animar a los usuarios a repetir su experiencia de uso.

Sexto, se permitirá la introducción de pagos integrados y los puntos de venta. Los usuarios habituales podrán hacer transacciones de pago diarias desde sus teléfonos móviles de manera automática. Esto habilitará programas de lealtad de usuarios y una recogida de retroalimentación valiosa y verídica.

Séptimo, se incrementará la seguridad. Los empleados y vigilantes de un espacio de aparcamiento tendrán un panorama del estado del aparcamiento en tiempo real, lo cual puede prevenir las

violaciones de parking y actividades sospechosas. Se podría recolectar más datos, p.e. cámaras de seguridad y las de reconocimiento de matrículas de vehículos a fin de mejorar la seguridad. Además, el decremento de tráfico en la ciudad puede reducir accidentes causados por distracción durante búsquedas de aparcamiento.

Octavo, será posible la recolección de datos en tiempo real y su análisis. Día tras día, se generarán los datos que ayudarán a descubrir las correlaciones importantes y modas de usuarios y espacios. Esos espacios pueden traer beneficios increíbles a sus dueños permitiéndoles hacer ajustes y mejoras para los conductores.

Noveno, se decrementarán los costes de gestión. Los procesos automáticos y trabajo manual reducido disminuirá los costes y el desgaste de recursos.

Décimo, se mejorará la imagen de servicio y marca. Una experiencia amena de aparcamiento subirá rápidamente la imagen de marca en los ojos de usuario. No es de especial importancia que el destino sea una tienda, el centro de la ciudad, un aeropuerto o una oficina de negocios. De igual manera los conductores serán impresionados con las últimas tecnologías y factores de conveniencia.

Este proyecto propone una solución basada en microcontroladores. Es un sistema embebido de bajo consumo energético capaz de detectar vehículos que entran en el aparcamiento y vehículos que lo abandonan y enviar los datos de estos eventos a la plataforma en la nube. Instalado en un lugar cercano al centro de la ciudad será una solución vertical para problemas existentes de tráfico excesivo e impacto ambiental. El dispositivo fue diseñado para funcionar en espacios abiertos y generalmente adaptados para poder instalarse en diversos lugares.

El proyecto descrito en el presente documento ha pasado a través de un ciclo completo de desarrollo de un nuevo producto, pasando por la identificación de requisitos, el diseño del producto, el desarrollo del algoritmo de detección de vehículos, el diseño de *hardware* y el desarrollo de *firmware*. Finalmente, las pruebas y validación fueron llevadas a cabo y las posibles mejoras fueron identificadas y consideradas.

Este documento proporciona detalles sobre cada etapa del desarrollo del sensor de aparcamientos y explica los principios de diseño que fueron aplicados durante esas etapas. Se presenta una documentación completa del algoritmo de detección de vehículos y los procesos de validación que tomaron su lugar en cada etapa del desarrollo.

Abstract

This document contains a degree final thesis project in which a smart parking vehicle detection system was developed. The system forms part of a complete IoT environment and covers the first two layers of IoT architecture, perception and network construction layers respectively. The project was developed in the IoTSens company which provides solutions based on Internet of Things for collecting, integrating, storing and analyzing vertical information concerning Smart Industrial, Smart Water, Smart Environment and Smart City. The IoTSens embodies the Grupo Gimeno's IoT division.

The developed system constitutes one of many existent sensor solutions for Smart Cities. In the era of optimization and data processing it brings many benefits reducing the time a vehicle looks for a parking place as well as a carbon dioxide produced by the moving vehicles. The final product is delivered to a customer as Platform as a Service (PaaS) infrastructure where recollected data is received and visualized.

For the development of this solution a complete cycle of product development activities such as Research and Development (R&D), Digital Signal Processing (DSP), algorithm design, firmware programming, hardware development and validation tests was carried out.

Keywords

Internet of Things, Smart City, parking, vehicle detection algorithm, AMR sensors, embedded software, firmware, hardware development, RFLink, Texas Instruments

Extended Abstract

Each year more cities become smart. Most of them are metropolitan scale cities, like Madrid or Barcelona, which have huge industry and tourism infrastructures. The solution which proposes this project pretends to improve many aspects of metropolitan infrastructure.

First, it will optimize city parking. It will help to find the best parking spot available, while saving drivers' time, vehicle as well as driver's mental resources, and his or her efforts. The parking lot fills up efficiently in terms of space that can be utilized properly by commercial and municipal entities.

Second, it will reduce traffic congestions. Since fewer vehicles are required to drive around searching parking spots traffic flow decreases.

Third, it will reduce pollution. It is estimated that searching for a parking burns around one million barrels of oil a day. This solution will optimize driving experience decreasing parking searching time and, thus, lowering the amount of daily vehicle emissions and reducing global environmental footprint.

Fourth, it will enhance user experience. The entire user experience will have an organized and unified action sequence.

1. Driver pays for a parking place.
2. Parking place is identified.
3. Location search.

All these become a part of the destination arrival process.

Fifth, it brings new innovative revenue streams. Smart parking brings a possibility to introduce many new revenue streams. For instance, parking lot owners can integrate tier payment option depending on parking space location. Also, reward programs can be enabled into existing models to encourage repeat users.

Sixth, it permits integrated payments and POS (Point of Sale) . Returning users can make daily cash transactions from their mobile phones automatic. This will enable customer loyalty programs and veridical valuable user feedback.

Seventh, it increases safety. Parking lot employees and security guards possess real-time lot state which will prevent parking violations and suspicious activities. More data can be gathered, i.e. license plate recognition cameras, security video cameras in order to enhance safety. Moreover, decreased spot searching traffic on the roads can reduce accidents caused by distraction during parking place searching.

Eighth, it will allow to recollect real-time data and analyze trend insights. Day by day it will produce data which will uncover important correlations and trends of users and lots.

These lots can bring incredible benefits to lots owners allowing them to make adjustments and improvements to drivers.

Ninth, it will decrease management costs. Automatic processes and less manual labor will save labor costs and resource exhaustion.

Finally, tenth, it will increase service and brand image. A seamless parking experience will skyrocket commercial entities brand image to the user. No matter if the destination is a retail store, city center, an airport or a corporate business office, visitors will surely be impressed with the cutting-edge technology and convenience factors.

This project proposes a microcontroller-based solution, that is parking sensor for smart cities. This is a low-power embedded system which is capable to detect a vehicle coming to a parking place or vehicle leaving and report those events to a cloud platform. Installed in a parking place around a city center it will form a vertical solution to solve the existent problems with excessive traffic and alleviate environmental impact caused by those problems. The device was designed to work in outdoor and generally aggressive environments, therefore it can be placed anywhere.

Project development described in the present document went through a complete cycle of product development. From the requirements identification, it came to the product design which enclosed vehicle detection algorithm development, hardware design, and firmware development. Finally, the tests and validation were carried out and a way for future improvements was considered.

The present document gives the details of each stage of the parking sensor development, exposes the principles which were applied during those stages, provides a complete documentation of the vehicle detection algorithm, and gives an insight into the validation taking place at each step.

Acknowledgements

In the first place I would like to express my gratitude to my Lord, God of Abraham, Isaac and Jacob, Who gave me all my mental and physical abilities, curios mind and perseverance.

In the second place, I am sincerely grateful to Ignacio Llopis, Manolo Gasch and a team of IoTSens, especially hardware department, who were my support and a primary source of learning on each step of project development. Without their friendly tenderness I would not understand fully the importance of a teamwork.

At last, but not least, I want to express thanks to Naara Mitca Valencia Balbuena for all her comprehension and emotional support demonstrated during the time we were together and all her patience and strength that empowered me to finish this project. Without her presence all my efforts would serve for nothing. To you, my love. You will always be the most important part of my life. Sincerely, thank you.

Contents

1 Introduction **21**

1.1 Context and motivation of the project 21

1.2 Project objectives 23

1.2.1 Objectives covered by internship 23

1.2.2 Objectives the company will cover after the internship 24

1.3 Project's structure 24

2 Description of the project **27**

2.1 System components 27

2.1.1 Nodes 28

2.1.2 Gateways 33

2.1.3 Web/Mobile app platforms 34

2.2 Communication technologies 36

2.3 Components' drivers 39

2.4 Development Software 39

2.4.1 Algorithm development 40

2.4.2 PCB Design 40

2.4.3 Firmware development 41

3 Project Planning **43**

3.1 Methodology 43

3.2 Planning 44

3.2.1 Initial Planning 44

3.2.2 Tasks 44

3.3 Resources and project's costs estimation 51

3.4 Project Monitoring 52

3.4.1 Sprints 53

4 Vehicle detection algorithm development **57**

4.1 Introduction 57

4.2 Data Acquisition Process 59

4.2.1 Sliding Window 59

4.2.2 Axes Compensation 60

4.3 Vehicle Detection Algorithm 62

4.3.1 State Machine 62

4.3.2 Achieving Stability 64

5 Implementation Details **69**

5.1 Hardware development 69

5.1.1 Schematic Design 70

5.1.2 Laying out the PCB 74

5.2 Firmware development 81

5.2.1 Identify the Requirements 81

5.2.2 Distinguish architecture from design 81

5.2.3 Manage time 83

5.2.4 Design for tests 83

5.2.5 Future improvements provisions 84

6 Tests and Validation **85**

6.1 Hardware Validation 85

6.2 Firmware Validation 86

6.3 Product Validation 86

6.3.1 Vehicle Detection Algorithm 86

6.3.2 System Requirements 88

7 Conclusions and Future Improvements **91**

7.1 Conclusions 91

7.2 Future Improvements 92

Bibliography **93**

List of Figures

1.1 Grupo Gimeno's Organization Chart 23

2.1 CC1310 Block Diagram 29

2.2 The Hall Effect Principle 31

2.3 The Magnetic Tunnel Junction schematic 32

2.4 One of the gateways installed in Castellón de la Plana 35

2.5 ISM/RSD License-Free frequency bands 38

3.1 Gantt chart. Part 1 47

3.2 Gantt chart. Part 2 48

3.3 Gantt chart. Part 3 49

3.4 Gantt chart. Part 4 50

4.1 The Earth's Magnetic Field Distribution 58

4.2 Disturbance caused by a ferrous object in a uniform field 58

4.3 Vehicle disturbance in the Earth's field 59

4.4 Process of sliding window sampling 60

4.5 Sensor positioning respect to a vehicle 61

4.6 Vehicle detection algorithm state diagram 62

4.7 Dynamic threshold adaption state machine 67

5.1 Schematic sheet 1 71

5.2 Schematic sheet 2 72

5.3 Schematic sheet 3 73

5.4 The layers composition of a double-side PCB 74

5.5 PCB power supply circuitry layout 75

5.6 TI CC1310 microconroller and RF antenna PCB layout 76

5.7 RTC final PCB layout 76

5.8 Flash memory final PCB layout 77

5.9 Temperature sensor final PCB layout 77

5.10 Magnetic field sensor final PCB layout 78

5.11 General PCB top layout view 79

5.12 General PCB bottom layout view 80

5.13 Parking sensor firmware architecture block diagram 82

5.14 Preferred placement of real-time functionality 83

6.1 Huge parking lot located in front of the Grupo Gimeno's office 87

List of Tables

3.1 Cost summary 52

6.1 Experimental results of the vehicle detection algorithm validation 87

6.2 Device power consumption summary 89

Chapter 1

Introduction

Contents

1.1 Context and motivation of the project 21
1.2 Project objectives 23
1.2.1 Objectives covered by internship 23
1.2.2 Objectives the company will cover after the internship 24
1.3 Project's structure 24

1.1 Context and motivation of the project

Any modern city today suffers from traffic jams with drivers circling around and looking for increasingly rare parking places. It is estimated that about 30 percent of cars on the road at any given moment are actively searching for parking places and can spend nearly 20 minutes to find a place in a city center areas or in the busy metropolitan areas [16]. This way a serious strain is put on both traffic congestion and city resources.

In the era of the greatest information technology advances this problem can be easily tackled using Internet of Things (IoT) technologies. Low-powered microcontroller-based embedded systems sense the environment, process data and send valuable information to a cloud system through gateways is a usual scenario for any IoT solution.

This scenario can be applied for smart parking solutions as well and it has five major benefits:

- It will help drivers to easily find parking places reducing traffic congestions. Installed sensors will send data to a cloud platform which guides a driver to a free parking space. This way the carbon dioxide (CO2) escape can be reduced and driver's time saved.
- More benefits can obtain city municipalities getting revenue from the smart parking solution. The busiest city parking areas can be found after some research and parking infrastructure system can be built in those places.

- Traditional parking payment systems can be avoided since drivers can pay over the cloud.
- Parking sensor as any IoT device generates data which can be processed, analyzed, and exploited afterwards.
- It will offer better parking experience and service for neighborhood and increase economic income of small businesses (PYMES)
- Finally, this solution can attract more tourism.

This project presents a smart parking sensor solution integrated to existing IoTSens cloud platform.

The company where development was accomplished is the IotSens Grupo Gimeno's IoT Division. The Grupo Gimeno is a group of companies founded 140 years ago in Castellon de la Plana, Spain. Currently, it comprises companies from different economic areas as water cycle services, restoration, construction and environmental resource management between others. The figure 1.1 resents the organizational chart of the Grupo Gimeno according to economical sector.

ADC Infraestructuras y Sistemas dedicates to development of wireless control systems of the water cycle service. ADC creates its own programs for management of water supply and control of water purification system. IoTSens is a proper company of ADC with specialization in the Internet of Things.

IoTSens is an information technology division of the Grupo Gimeno and provides hardware and software scalable and interoperable solutions to collect and exchange data connecting physical world with computer-based systems. The company is divided into 4 sectors: software, hardware, business management and, public communications. The present project is located in hardware sector.

The internship during which the project development has taken place was completed in the IoTSens hardware department which leader was the internship tutor. However, the project covers hardware design and firmware development and it led to interact with two departments: previously mentioned hardware development department and the embedded development department which is in charge of firmware design and implementation.

Hardware development department comprises two highly qualified engineers who revised the advances I had, shared with me their experience and the way to accomplish design objectives effectively.

Embedded development department is formed by two professionals, as well as the hardware development. They possess ample experience in firmware design and architecture. They transmitted me knowledge about firmware design and architecture and led me through the firmware development.

From the technical perspective there is another department which is in charge of the cloud platform. This is software development department and it is not so closely cohesioned to the hardware and embedded development. For this reason, the communication with co-workers from this department was not that stretch as with the low-level-development departments.

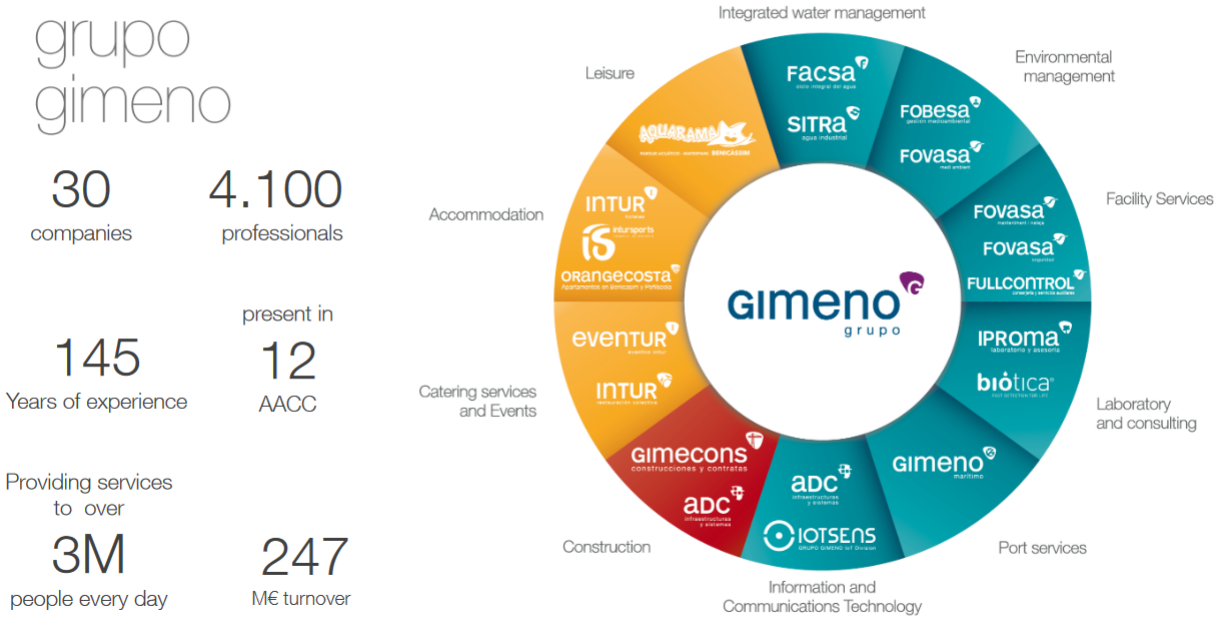

Figure 1.1: Grupo Gimeno's Organization Chart

1.2 Project objectives

In this section the project objectives covered during the internship as well as after it (future development of improvements and maintenance) are presented.

1.2.1 Objectives covered by internship

During the period of internship the compete Smart City Parking product development was expected. The decided order of implementation was as follows:

1. Setup system configuration. Wiring of the magnetic field sensor with the microcontroller used for algorithm development (Atmel 1228b) over the development board.
2. Programming of a library for the sensor management for Atmel 1228b.
3. Achieve correct data acquisition and recollection.
4. Develop an algorithm for vehicle detection using 3-axis MEMS (microelectromechanical system) ultra-low-power magnetic field sensor. Use Atmel 1228b Microcontroller as development board due to its friendly interface and easy way of programming.
5. Design a plastic box for the sensor according to IP68 standard among others.
6. Create a prototype. Develop a PCB (Printed Circuit Board) hardwiring the connections between components. Texas Instruments CC1310 is the microcontroller to be used.

7. Program firmware for the prototype. It includes microcontroller initialization, digital signal processing and communications. Data will be transmitted via RFLink protocol (Radio Frequency).
8. Perform product testing. Make final improvements.
   (a) Corporate parking
   (b) Outside street environment
9. Implement final version of the firmware.
10. Manufacture the product and program it with the final version of firmware.

1.2.2 Objectives the company will cover after the internship

Once the first version of the product is developed, the company will focus on the software level of IoT architecture. Objectives relative to mentioned aim will be:

1. Find an optimal and convenient way of received data representation.
2. Introduce sensor geolocation.
3. Enrich data transmitted from the sensor adding more scenarios which the sensor is capable to detect.

With regard to the hardware architecture the goal is to move the design to modular hardware design. The system might be divided into smaller subsystems called modules. It concerns communication technologies in particular. Depending on a particular environment or customer needs the same parking sensor would transmit data using LoRaWan, WiFi, Bluetooth or GPRS technologies.

1.3 Project's structure

The first chapter was dedicated to project introduction. Context, clear motivations, objectives to achieve, brief explanations and several technologies to use were presented to the reader.

The second chapter describes, first, the system's functionality, next, the components which comprise the system, furthermore wired and wireless communication technologies used and finally component drivers which were indispensable in order to design the final product.

The third chapter brings details into working methodology and initial project planning. Resource costs and project costs estimation are described along with project monitoring and its evolution.

The fourth chapter exposes the system analysis and its multilayered architecture.

The fifth chapter gives the details of the vehicle detection algorithm design, foundations, development strategies, limitations and, possible improvements.

The sixth chapter specifies implementation details. It brings to light a description of each step of development from PCB design coming along by vehicle detection algorithm development, firmware and component drivers implementation, and, finally, verification and validation tests of the system's functionality.

To conclude, the seventh chapter reveals the conclusions and possible future improvements of the project.

Chapter 2

Description of the project

Contents

2.1 System components 27
2.1.1 Nodes 28
2.1.2 Gateways 33
2.1.3 Web/Mobile app platforms 34
2.2 Communication technologies 36
2.3 Components' drivers 39
2.4 Development Software 39
2.4.1 Algorithm development 40
2.4.2 PCB Design 40
2.4.3 Firmware development 41

The idea behind the smart city parking is simple. It was described in the introduction. It aims to save drivers' valuable time spent during search of a parking place, save the air quality and natural environment by reducing continuously increasing traffic jams. Traffic jams provoke considerable carbon dioxide escape. This problem appears with city population growth and infrastructure development. More people move to the cities led by a desire to have a better life and stable working conditions and attracted by opportunities they found there.

This chapter gives a general view of a solution infrastructure and unveils the detailed description of the proposed solution for the expressed problem.

2.1 System components

An IoT ecosystem consists of three dimensions. Each dimension represents a complex system with its proper architecture, whether hardware or software, which supplies necessary functionality.

In this section three IoT layers will be described. The present project corresponds to the first IoT architecture layer thus, it will be considered in detail in the next subsection.

2.1.1 Nodes

In this layer sensors and devices are located. Those actuators are "Things" part of IoT projects. The devices in this layer interact with the physical environment and collect data or object under measurement and turn them into useful data.

The type of node this project is focusing on is the parking sensor. All the principal components of the designed parking sensor system will be described below.

Microcontroller

This is the core component of each sensor device. It performs all the most important operations and orchestrates the data recollection and forwarding to the next layer.

Selecting a right microcontroller is a challenging task. The designer should not only carefully consider a list of technical features, but also business case issues like costs and lead-times that can potentially hider the project [20].

The microcontroller which was selected for the project is Texas Instruments CC1310 SimpleLink Sub-1 GHz Ultra-Low Power Wireless Microcontroller [12]. There are three major steps that were taken before the current microcontroller has been selected. First, a list of required hardware interfaces was composed. Required communication interfaces are I^2C (Inter-Integrated Circuit), SPI (Serial Peripheral Interface), UART (Universal asynchronous receiver/transmitter) and RFLink (Radio Frequency Link) communication module. It should be taken into account how much space each interface driver occupies since microcontrollers are limited in power, memory, and functionality systems. Other kinds of interfaces that should be considered are digital inputs and outputs, analog to digital inputs, PWMs and so on. Necessary number GPIO (General-purpose input/output) pins was considered for the required applications.

The Texas Instruments CC1310 MCU has following features [12]:

- ARM® Cortex®-M3 processor core
  - 48-MHz RC oscillator and 24 MHz XTAL oscillator with an internal doubler
  - 2-kHz XTAL oscillator, 32-kHz RC oscillator or low-power 24-MHz XTAL derivate clock for timing maintenance while in low-power modes
  - ARM® Cortex® SysTick timer
  - NVIC (Nested vectored interrupt controller)
- On-chip memory
  - Flash with 8KB of 4-way set-associative cache RAM for speed and low power

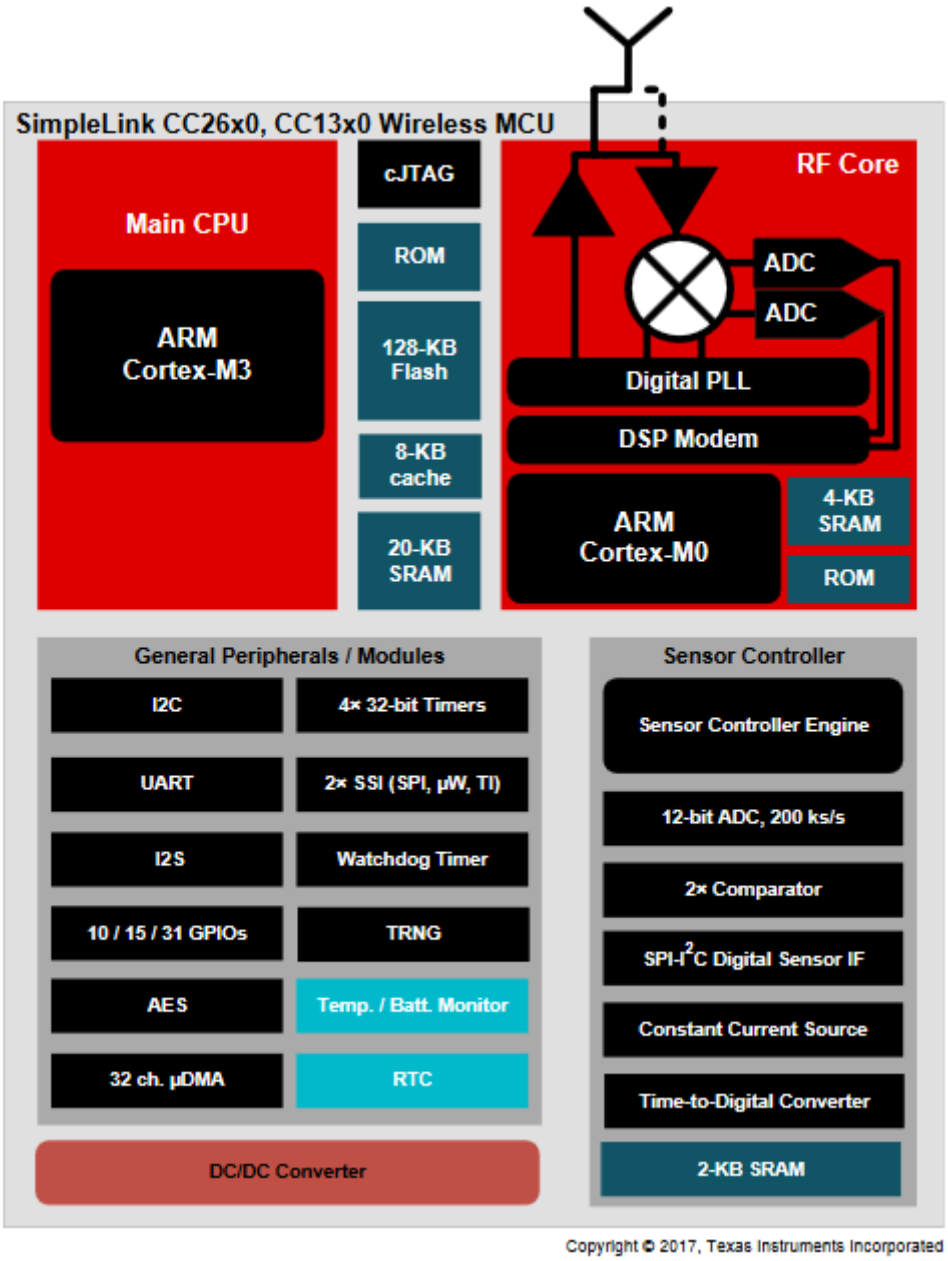

Figure 2.1: CC1310 Block Diagram

  - System RAM with configurable retention in 4KB blocks

- Power management
  - Wide supply voltage range
  - Efficient on-chip DC/DC converter for reduced power consumption
  - High granularity clock gating and power gating of device parts
  - Flexible low-power modes allowing low energy consumption in duty cycled applications

- Sensor interface
  - Autonomous, intelligent sensor interface that can wake up independently of the main CPU system to perform sensor readings, collect data, and determine if the main CPU must be woken
  - 12-bit ADC (analog-to-digital converter) with eight analog input channels
  - Low-power analog comparator
  - SPI or I^2C master bit-banged

- Advanced serial integration
  - UART
  - I2C module
  - SSIs (Synchronous serial interface modules)
  - Audio interface I2S module

- System integration
  - DMA (Direct memory access) controller
  - Four 32-bit timers (up to eight 16-bit) with PWM (pulse width modulation) capability and synchronization
  - 32-kHz RTC (real-time clock)
  - Watchdog timer
  - On-chip temperature and supply voltage sensing
  - GPIO with normal or high-drive capabilities
  - GPIOs with analog capability for ADC and comparator
  - Fully flexible digital pin muxing allows use as GPIO or any peripheral function

- IEEE 1149.7 compliant 2-pin cJTAG with legacy 1149.1 JTAG support

For applications requiring extreme conservation of power the CC1310 device features a power-management system to efficiently power down CC1310 devices to a low-power state during extended periods of inactivity. A power-up and power-down sequencer, a 32-bit sleep timer (an RTC), with interrupt and 20KB of RAM with retention in all power modes positions the CC1310 microcontroller perfectly for battery applications.

Magnetometer

The key-component which will bring expected functionality to the device is the magnetometer. Since the final product aims to cover outdoor and indoor environments, a system based on sonar sensors is not suitable. Thus, the magnetometer was considered to be the best choice because it covers aimed and even more aggressive environments.

Magnetometers are sensor or devices that measure magnetic fields and magnetic flux density normally in units of Tesla. Magnetic materials distort magnetic flux lines which flow around them. Created distortions in the Earth's magnetic flux are detectable by this kind of sensors.

There are many kinds of magnetic sensors, but this documents will only discuss two of them so as to describe a basis for the selection criteria. The most well-known Hall sensor is based on the Hall Effect principle and another one is the most innovative and efficient one based on MTJ (Magnetic Tunnel Junction) or Tunnel magnetoresistance principle.

The Hall Effect principle states that when a current-carrying conductor is placed in a magnetic field, a voltage will be generated perpendicular to the direction of the magnetic field and the flow of current [3]. The uneven distribution of electron density creates a potential difference across the output terminals. This voltage is called the Hall voltage and it indicates the magnitude of crated distortions. This simple principle is illustrated in the Figure 2.2.

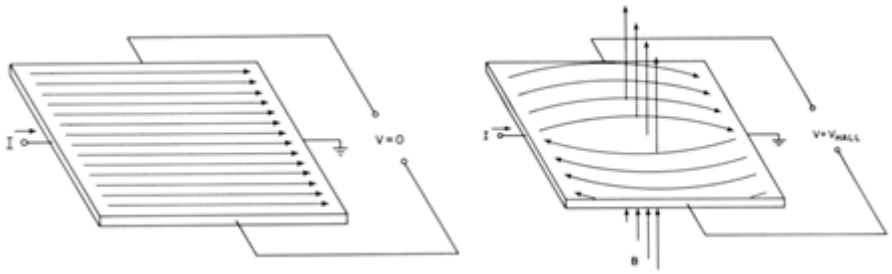

Figure 2.2: The Hall Effect Principle

This kind of sensors are not energy-consumption efficient, neither they possess high sensitivity. Therefore another, the most innovative, principle was considered, magnetic tunnel junction.

A magnetic tunnel junction consists of two layers of magnetic metal, such as cobalt-iron, separated by an ultra-thin layer of insulator, typically aluminum oxide with a thickness of about 1 nm. The insulating layer is so thin that electrons can tunnel through the barrier if a bias voltage is applied between the two metal electrodes [22]. A scheme of this kind of sensors is illustrated in the Figure 2.3.

It is estimated that MTJ sensors are 10 times more sensitive and 100 times more energy-consumption efficient.

The selected MTJ sensor is LIS3MDL, ultra-low-power high-performance three-axis magnetic sensor with user-selectable full scales of $\pm 4/\pm 8/\pm 12/\pm 16$ gauss. It includes an I^2C serial bus interface that supports standard and fast mode (100 kHz and 400 kHz) and SPI serial

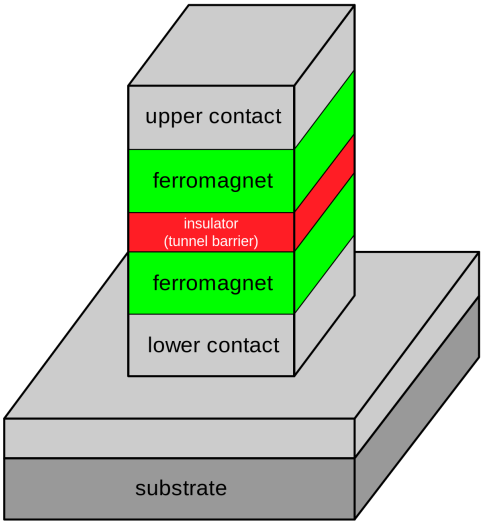

Figure 2.3: The Magnetic Tunnel Junction schematic

standard interface [17].

Real Time Clock (RTC)

In order to reduce power consumption in the uttermost way an external RTC is required. In spite of existing internal RTC inside the microcontroller the use of an external one is advisable. In practice, when the internal RTC is used its circuitry, which can be shared with other peripherals, has to be powered on what increases its power consumption and does not allow to the MCU go to the deepest sleep states. This integrated circuit communicated through I^2C serial bus will generate interruptions to wake up the device for the next measurement while the microcontroller is driven to the power saving standby mode.

For this purpose the MCP79411 RTC IC was elected. It includes a battery switchover circuit for backup power [6] which combined with a supercapasitor can maintain its work for more than 1 week due to its ultra-low-power consumption. In order to run this IC it is enough to connect a low-cost 32.768 kHz crystal and power source.

In addition, a Unique ID in a locked section of EEPROM is included which is factory programmed with an EUI-48 MAC Address. This ID is extremely important and used to identify the device when it transmits status data to the server.

Temperature sensor

Temperature sensor is a small integrated circuit which sense temperature and is capable to deliver it with configurable precision. It carries out two functions:

1. It alerts the system when certain thresholds are surpassed. The established thresholds are

-35°and 75°which correspond to low and high limits the microcontroller, magnetometer, and integrated circuits can properly work. Then these limits reached an alert is generated and adequate information is sent to the cloud platform before the system has gone down.

2. The MTJ is a temperature dependable phenomenon. Thus, in order to make accurate magnetic field measurements temperature compensation should be taken into account during calculations.

The AT30TS75A from Microchip Technology Inc was selected for this goal. It is factory calibrated and and requires no external components to help provide an accurate temperature. To reduce current consumption and save power, the AT30TS75A features a shutdown mode which turns off all internal circuitry except for the internal power-on reset and serial interface circuits [5]. In addition, the device features a power saving one-shot mode that allows the device to make a temperature measurement and update the temperature register and then return to shutdown mode.

Flash memory

The flash memory is supposed to store the most important event records which can be sent to the server at any moment and other log information. It will be the important piece of hardware when the firmware over-the-air update functionality will be added. The downloaded firmware will be stored in this memory and loaded by the bootloader after the updated finished.

The W25Q128JVSIM, manufactured by Winbond Electronics, provides a storage solution for systems with limited, but sufficient space, pins and power. It operates on a single 2.7V to 3.6V power supply with current consumption as low as 1 µA for power-down.

The W25Q128JVSIM array is organized into 65,536 programmable pages of 256-bytes each. Up to 256 bytes can be programmed at a time. Pages can be erased in groups of 16 (4KB sector erase), groups of 128 (32KB block erase), groups of 256 (64KB block erase) or the entire chip (chip erase) [14].

It communicates through the standard SPI interface and it is the unique IC in the project which uses this connectivity.

2.1.2 Gateways

After the data have been aggregated, digitalized, and processed they are prepared for further downstream. The Internet gateways receive the aggregated and processed data and routes it over employed communication technology, i.e. LoRaWan, Sigfox, ZigBee, Wi-Fi and so on, for further processing.

Gateways often sit close to the sensors and devices. However, it depends on the chosen communication technology. It is often a matter of the characteristics and requirements of the project. Basically, there are several factors that are considered when a communication technology is being chosen:

1. **Devices dispersion**: If devices are located in the same area like a building, an industrial plant or even a neighborhood, short-range technologies like Wi-Fi, ZigBee or even Bluetooth would be a good choice. Otherwise, If devices are widespread in a city or country, long-range technologies like LoRa or Sigfox can be the right choice. If an area is not the same, neither wide-ranged, median-range technologies are suitable, like RFLink.

   Parking lots normally are located substantially away one from the other and counted as medium size installations. Thus, the most convenient technology that was preferred was RFLink which, in practice, has a range of connection about 800 m.

2. **Required Data Rates**:Some networking protocols are not suitable depending on the amount of data that the devices send. For instance, Sigfox and LoRa do not provide enough bandwidth when a sensor must send the temperature of a room or the state of a parking lot every minute. RFLink is able to send data with a rate up to 4 Mbps which is enough even if a sensor state will change every 10 minutes.

3. **Network coverage**: It is possible to deploy a private network for the LoRa project including the use of third-party LoRa networks (TTN) or use an available network provided by third parties (2G, Sigfox, NB-IoT, etc.).

   IoTSens has its private RFLink networks which cover several zones throughout the Castellón de la Plana city.

It is worth to note that gateways normally have three or four communication technologies integrated to aggregate data from multiple networks. One of the installed gateways can be seen on the Figure 2.4.

The description of the gateway used in this project is not in the scope of this document.

2.1.3 Web/Mobile app platforms

The software platform of any IoT project will be in charge of managing the devices, receiving and processing the messages. It overlooks devices onboarding process and monitors the current internal configurations and functionality. Using the cloud platform it is possible even to update device firmware when the device is appropriately programmed.

Management Service

Data processing requires unambiguous identification of each datum. Application Program Interfaces (API) for reading and gathering data must be provided. The IoT software platform must be flexible enough to support different communication protocols: MQTT, REST, XMPP, WebSockets, and so on. All these details comprise a management service system upon which reigns concrete applications.

When the parking sensor device reports a change of a parking place state, it transmits the data according to designed protocol. These data are processed according to the provided API and visualized for the corresponding application.

Figure 2.4: One of the gateways installed in Castellón de la Plana

Applications

All IoT projects are carried out for a purpose. IoT applications are just software systems which use the data that are received by the devices and the functionality that they provide. Depending on the level of customization, three categories can be defined:

1. **IoT vertical applications** which provide out-of-the-box functionalities for a specific application domain like smart waste management, smart building monitoring, smart water metering, smart irrigation, or smart parking.
2. **Toolboxes and frameworks** for building a private dashboards, reports, alarms, and graphics. These can be independent products which integrate with external data sources or they can be provided as a part of the IoT software platform.
3. **Custom software applications** which are developed from the ground up using standard software development technologies. These applications will use the IoT software platform APIs as the foundation for building their functionality.

The smart parking solution proposed by this project is delivered as an IoT vertical application and it visualizes on the map parking lots with free or occupied parking spots. However, the scope of the project extends only to the parking sensor and all the subsystems it covers: vehicle detection algorithm development, hardware design, firmware development, and the rest of tasks related to the product development.

The detailed description of the IoTSens cloud platform involved in this project is not in the scope of this document.

2.2 Communication technologies

Communications is one of the most important part of any IoT infrastructure without which the concept of IoT would make no sense. Communications cover each step of system functionality, from the architecture to data transmission.

Microcontroller is the core of commanding and orchestrating the system work. It should communicate with each IC according to the specified protocol and govern them sending commands and receiving responses. After the data have been prepared to send another dimension of communication starts, wireless data transmission which involves completely different technologies and protocols. Thus, there are two dimensions of communications: horizontal or inter-circuit communications and vertical or communication with gateways.

Horizontal communications

This section will provide a brief introduction into inter-IC communication protocols used in the project.

In order to achieve the desired functionality, it is indispensable to coordinate operations performed on the PCB. Each IC carries out its functions, and it must be directed by some means. For this purpose there were engineered inter-IC protocols.

The protocols used in the project are listed bellow.

I^2C The Inter-integrated Circuit Protocol is a protocol intended to allow multiple "slave" digital integrated circuits ("chips") to communicate with one or more "master" chips. Similar to the SPI protocol it is intended only for short-distance communications and like UART it requires only two signal wires to exchange information that can support up to 1008 slave devices.

I^{2}C can support a multi-master system, allowing more than one master to communicate with all devices on the bus (although the master devices can't talk to each other over the bus and must take turns using the bus lines). Most I2C devices can communicate at 100kHz or 400kHz [9].

Each I^{2}C bus consists of two signals: SCL and SDA. SCL is the clock signal, and SDA is the data signal. The clock signal is always generated by the current bus master; some slave devices may force the clock low at times to delay the master sending more data (or to require more time to prepare data before the master attempts to clock it out). This is called "clock stretching" and is described on the protocol page.

The detailed description of I^{2}C protocol is not in the scope of this project.

SPI

Serial Peripheral Interface is an interface bus commonly used to send data between microcontrollers and small peripherals such as common ICs, shift registers, sensors, and SD cards. It uses separate clock and data lines, along with a select line to choose the device you wish to talk to [8].

It's a "synchronous" data bus, which means that it uses separate lines for data and a "clock" that keeps both sides in perfect sync. The clock is an oscillating signal that tells the receiver exactly when to sample the bits on the data line. This could be the rising (low to high) or falling (high to low) edge of the clock signal; the datasheet will specify which one to use. When the receiver detects that edge, it will immediately look at the data line to read the next bit. Because the clock is sent along with the data, specifying the speed isn't important, although devices will have a top speed at which they can operate.

SPI advantages: it's faster than asynchronous serial, the receive hardware can be a simple shift register, and it supports multiple slaves. The disadvantages are: it requires more signal lines (wires) than other communications methods, the communications must be well-defined in advance, the master must control all communications (slaves can't talk directly to each other), and it usually requires separate SS lines to each slave, which can be problematic if numerous slaves are needed.

It is noticeable that the I^2C possesses more advantages and is more attractable during the design process. By the way, three ICs communicate through it (Temperature sensor, RTC, and magnetometer) and the flash memory IC communicates via SPI interface.

Vertical communications

When it comes to vertical communications, it should be thought as a way to communicate or report collected data to a gateway which is normally located at a long-distance range from the device.

The CC1310 is a wireless microcontroller. It is manufactured with built-in radio frequency (RF) module which is used to transmit and/or receive radio signals between the device and a gateway.

The radio frequency used for data transmission must comply with the Frequency Spectrum Allocation standard bands. The unlicensed Industrial, Scientific and Medical (ISC) and Short Range Devices (SRD) bands are used for this aim. The license-free band frequencies according to geolocations can be observed on the Figure 2.5.

In Europe the allowed frequencies listed bellow:

- 433.050 – 434.790 MHz (ETSI EN 300 220)
- 863.0 – 870.0 MHz (ETSI EN 300 220)
- 2400 – 2483.5 MHz (ETSI EN 300 440 or ETSI EN 300 328)

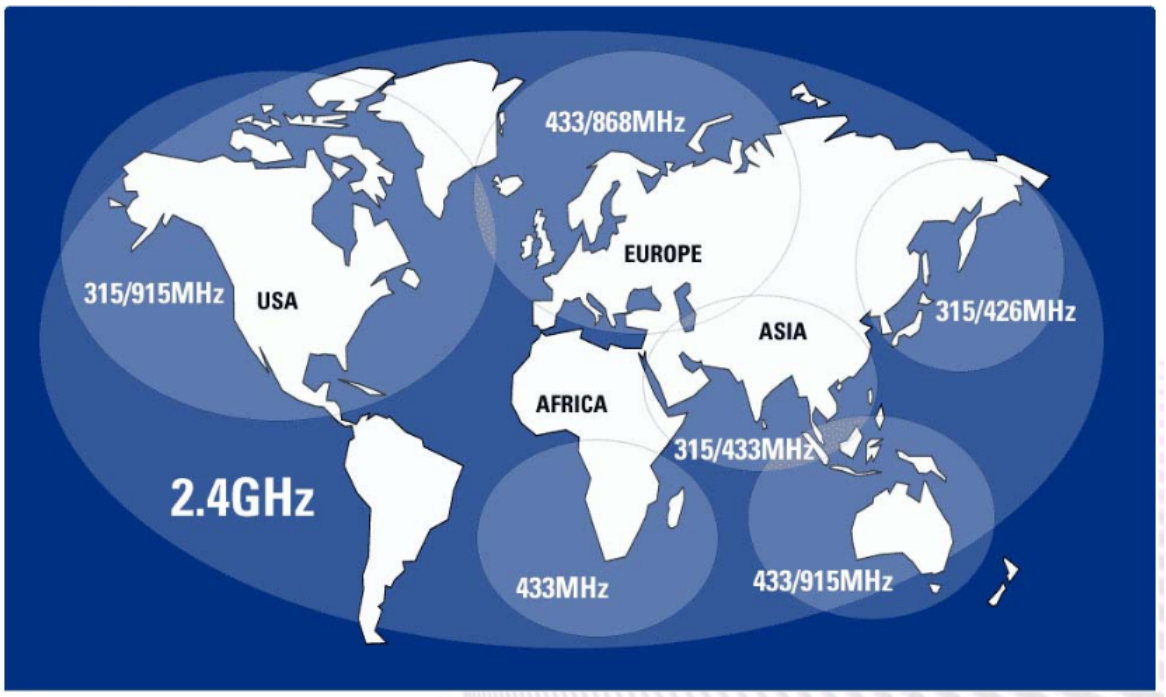

Figure 2.5: ISM/RSD License-Free frequency bands

The radio frequency used in the project is 863.0 MHz.

The radio in the CC1310 device offers a wide variety of different operational modes, covering many different packet formats. The radio firmware executes from the CC1310 radio domain on an ARM® Cortex®-M0 processor, which can provide extensive baseband automation. The application software interfaces and interoperates with the radio firmware using shared memory interface (system RAM or radio RAM) and specific handshake hardware (radio doorbell).

The RF core contains an ARM® Cortex®-M0 processor that interfaces the analog RF and baseband circuitries, handles data to and from the system side, and assembles the information bits in a given packet structure. The RF core offers a high-level, command-based application program interface (API) to the system CPU (ARM® Cortex®-M3). The RF core can autonomously handle the time critical aspects of the radio protocols (802.15.4 RF4CE and ZigBee®, Bluetooth® low energy, and so on), thus offloading the system CPU and leaving more resources for the user's application.

The RF core receives high-level requests from the system CPU and performs all the necessary transactions to fulfill them. These requests are primarily oriented to the transmission and reception of information through the radio channel, but can also include additional maintenance tasks such as calibration, test, or debug features.

As a general framework, the transactions between the system CPU and the RF core operate as follows:

- The RF core can access data and configuration parameters from the system RAM. This access reduces the memory requirements of the RF core, avoids needless traffic between

the different parts of the system, and reduces the total energy consumption.

- In a similar fashion, the RF core can decode and write back the contents of the received radio packet, together with status information, to the system RAM.
- For protocol confidentiality and authentication support purposes, the RF core can also access the security subsystem. It provides an AES-128 circuitry which reduces the use of main ARM® Cortex®-M3 processor.
- In general, the RF core recognizes complex commands from the system CPU (CCA transmissions, RX with automatic acknowledge, and so forth) and divides them into subcommands without further intervention of the system CPU.

2.3 Components' drivers

Any system can be considered in terms of its components which represent functional blocks. The coordination of each functional block is necessary to reach the pretended functionality. When it comes to the PCB design and system composition, the components are the integrated circuits where each IC performs a specific function.

The above written concepts can be easily understood, however it requires a logical support to use the functionality of each IC. All ICs need a driver to manage functions they were created for. Those drivers can be seen from two main perspectives: user and programmer views.

On one hand, a user can see a driver like a program that operates or controls a particular type of device and provides a software interface to the tangible hardware. Thus, the user sees a driver like an instrument that can be used when a regarding function is required.

On the other hand, a programmer sees a driver like a library which implements a series of functions. The functions encompass communications with a device through a bus, parallel electrical wires with hardware connections. Sending signals or transmitting data over the bus is a fashion the device receives commands and produces an output. Thus, the library is compound of functions which transfer bytes of memory, to the device or from it, achieving its purpose.

As it was commented earlier the system has four major components each of each requires a separate library to be managed. There were tree libraries already implemented by the company's employees and manufacturer: RTC and temperature sensor (employees) and external flash memory (Texas Instruments). One library had to be implemented as the project requirement, the library to manage the magnetometer.

2.4 Development Software

All information technology practical projects require a set of computer programs that facilitate their implementation. This section will describe which software was used on each step of development giving a brief insight into a corresponding contribution.

There were tree main phases of development:

1. Algorithm development
2. PCB development
3. Firmware development

2.4.1 Algorithm development

The software which was used during the algorithm development and the functionality it brought is listed below:

Visual Studio Code It is a source code editor developed by Microsoft for Windows, Linux and macOS. It becomes vary developer-friendly when it comes to features it has. Some of them are support for debugging, embedded Git control, syntax highlighting, intelligent code completion, snippets, and code refactoring. It is free and open-source, although the official download is under a proprietary license.

There is one important plugin which converts Visual Code Studio into a great IDE for IoT systems development, mainly PlatformIO. Since all firmware development lies under C/C++ programming languages it introduces Intelligent Code Completion and Smart Code Linter for rapid professional development and multi-projects workflow with Multiple Panes [7].

Special advantage of PlatformIO is Multi-platform Build System without external dependencies to the OS software: more than 400 embedded boards, 20 development platforms, and 10 frameworks. Arduino and ARM embed compatible.

Since during the algorithm development the ATMega Arduino-compatible microcontroller was used, it was a perfect IDE for rapid and efficient programming and loading the source code into the device memory.

HTerm It is a terminal program for the serial interface that runs under Windows and Linux. Its features make it a great debugging tool. Support of connection with any kind of serial ports; all baud rates available on the target hardware; input and output transmissions in ASCII, hexadecimal, binary and decimal formats via those ports; sending and saving files; parity for send and receive; copy any received formats to the clipboard are distinguishable features of the program.

It served as a powerful debugging tool in the project. Continuous system monitoring was possible and used for error correction during the cyclic process of algorithm development.

2.4.2 PCB Design

PCB design is a process which involves a CAD (Computer-Aided Design) and ECAD (Electronic Computer-Aided Design) which is also referred as an EDA (Electronic Design Automation)

software. Essentially, the PCB design is a step in the design for manufacturing a product. It is taken immediately after the diagram of functionality of the circuit has been created. This diagram is called a schematic.

In order to manufacture a PCB, it is necessary to take the design from the functional diagram or schematic and change it into a form of artwork that makes a pattern of components, holes and wires. This pattern is nowadays in the form of digital data and is used by a bare PCB manufacturer to control their photographic imaging techniques and computer-controlled drilling machinery, all resulting in the manufacture the PCB.

There are a number of good EDA software tools that were written for this purpose. Though, in this project, Autodesk's Eagle was used to create the PCB for the final product.

Autodesk Eagle EAGLE contains a schematic editor, for designing circuit diagrams. The connection between a circuit diagram and a PCB layout is continuously maintained. All changes stay automatically in synchronization between the schematic and layout, so it allows to focus on the creative process. It contains many features that accompany a fast and efficient design process. Among other features SPICE simulator that allows to validate circuit performance; modular design blocks which ease the design process; electronic rule checking helps to check all unconnected elements before the design is sent to the manufacturer house; libraries of complete components and manufacturer parts that are immediately used in the design process [4].

The detailed description of the process and main principles of PCB design for this project will be presented in the chapter 5.

2.4.3 Firmware development

As it is known, firmware is a specific software that provides the low-level control for the device's specific hardware. Particularly, since the project can be considered as an embedded system, it is the only program that will run on the system and provide all its functionalities.

Parking Sensor System requires a firmware which should implement a vehicle detection algorithm, diligently care about power consumption, and have data communication with the gateway.

The way of firmware programming completely depends on the microcontroller manufacturer chosen for the design. Each microcontroller has its method of programming or "burning" and each manufacturer supplies the instructions and software for this.

The CC1310's manufacturer, Texas Instruments, provides a specific IDE for the firmware development for the projects that involve their microcontrollers.

Texas Code Composer Studio Code Composer Studio is an IDE that supports TI's Microcontrollers portfolio. It comprises a suite of tools used to develop and debug embedded

applications. It includes an optimizing C/C++ compiler, source code editor, project build environment, debugger, profiler, and many other features [13]. Code Composer Studio is a 'fork' of Eclipse software framework and combines its advantages with advanced embedded debug capabilities from TI resulting in a compelling feature-rich development environment for embedded developers and firmware programmers.

HTerm This software program has been described in Section 2.4.1.

The software described through the section 2.4 is a set of preferred programs used by the company.

Each step that it took through to implement the firmware for the project will be described in the Chapter 5.

Chapter 3

Project Planning

Contents

3.1 Methodology 43
3.2 Planning 44
3.2.1 Initial Planning 44
3.2.2 Tasks 44
3.3 Resources and project's costs estimation 51
3.4 Project Monitoring 52
3.4.1 Sprints 53

3.1 Methodology

IoTSens is relatively young company and maintains the startup spirit, thus almost all projects which are being developed using agile methodology. This project is not an exception, therefore during the algorithm development, PCB design, firmware programming, and testing agile methodologies were applied.

Preferred methodology for the project development was SCRUM. It forms part of Agile Method and was created in 1993 by Jeff Sutherland [15].

Basically explained, the Scrum Method works according to the following guidelines:

1. A product creator makes a prioritized task list called a product backlog.
2. A Scrum Team is created and a "sprint planning" meeting is called. The team decides on the first priority of the product backlog and decides how to implement those pieces.
3. The team has a specific amount of time called a "sprint" to complete the work – usually two to four weeks.

4. The team meets each day to assess the progress. These meetings are called “daily Scrum”.
5. The ScrumMaster supervises and keeps the team focused on its goal to be sure deadlines are met.
6. At the end of the sprint, the work should be ready to show to a customer or project creator to evaluate the progress.
7. The sprint ends with a sprint review and analysis.
8. Then the next sprint begins and the team chooses another part of the product backlog to begin working again.

Every week there was a session with the project's tutor to consider a state of the project and relative progress. Once a current state was considered and previous progress discussed, further decisions have been taken for the next steps or corrections to apply if were necessary.

Apart from the tutor sessions, twice a week the whole team of hardware department had meetings or so called 'dailies' where all current projects were discussed and each member could propose improvements or help with current issues of each project. This methodology facilitated team cooperation and contributed to possible cohesion between different projects.

It should be considered that the whole hardware development team shared common workspace, thus it was easy to communicate each other in case of complex issues, doubts, and faced problems.

3.2 Planning

Though there was certain guidance during sessions with the tutor and 'dailies' it was not enough for integral project development. For that reason a detailed project planning was created. It described the tasks which should have been carried out to finish the project and time that should have been dedicated to each of them.

3.2.1 Initial Planning

The project duration lied in a period of time between the 8th of June 2018 through to 30th of September 2018. During this period a complete development of the project was expected, from algorithm development to the testing and validation. The tasks which were intended to be finish are listed in the following section.

3.2.2 Tasks

The tasks that had to be carried out to complete the development of the project are described in the next listing

- Algorithm development:
  - Utility preparation
    - Development board preparation. (ATMega 1284b)
    - Connection of the magnetometer sensor to the board.
    - Program magnetometer sensor library for the development board.
    - Achieve magnetic field data acquisition.
    - Sensor calibration.
  - Cyclic process of algorithm development
    - Magnetic field dataset recollection.
    - Data visualization.
    - Hypothesis construction.
    - Hypothesis programming.
    - Hypothesis validation.
    - Error corrections and addition of improvements.
  - Machine state creation.
  - Refactoring and code cleaning for implementation of the algorithm's final version.
- PCB design:
  - Final product's microcontroller selection.
  - Necessary integrated circuits selection and evaluation.
  - Functional diagram or schematic preparation.
  - PCB design process. (Wiring the components)
    - Creation of necessary components libraries for the PCB development software (Eagle).
      - Package creation.
      - Symbol creation.
      - Device creation.
    - Microcontroller power supply consideration.
    - Board's geometric form design.
    - Ground plane planning.
    - Peripheral connectivity.
    - General connectivity.
  - Bill Of Materials (BOM) creation.
- Firmware programming:
  - Magnetometer sensor library programming.
  - Magnetometer sensor library testing.
  - RTC library implementation.
  - RTC library tests.
  - Temperature sensor library programming.
  - Temperature sensor tests.

  - NVROM library programming.
  - NVROM tests.
  - Interruptions configuration.
  - RFLink connectivity configuration.
  - Vehicle detection algorithm adaption.
  - Power saving mode for the device idle state programming.
  - Memory pages programming.
  - Code refactoring and cleaning.
- Prototype preparation
  - PCB soldering.
  - System programming (burning).
  - Device assembling.
  - Gateway connectivity establishment.
- Data interpretation API writing.
- Project launching
  - Testing
    * Sensors installation in the corporate underground parking.
    * Corporate underground parking sensor testing.
    * Sensor installation in a city parking lot.
    * Parking lot in a street environment sensor testing.
  - Vehicle detection algorithm adjustments.
  - IoTSens cloud platform integration.

The real time projection of the tasks defined in the previous section will be visualized in the next figures 3.1, 3.2, 3.3, and 3.4.

Figure 3.1: Gantt chart. Part 1

Figure 3.2: Gantt chart. Part 2

Figure 3.3: Gantt chart. Part 3

Figure 3.4: Gantt chart. Part 4

3.3 Resources and project's costs estimation

Initial project duration was 370 hours which extended over 16 weeks including day offs and weekends, 5 hours per working day. If a cost of the working hour of a firmware/hardware developer can be esteemed around 15€ per hour [2], therefore the human costs will be of 4.500€.

All software used during the project development was open source with no additional expenditures. However, there needed multiple microcontrollers for various development purposes.

During the algorithm development an Arduino-like board which was developed by IoTSens employees was used. It has a cost of 30€ plus additional costs of batteries of 7€ to bring autonomy.

After the PCB design was finished, the design board manufacturing was needed. Firstly, it was necessary to develop the parking sensor prototype. Secondly, the testing phase required a number of manufactured sensors to be installed on the streets. Thus, 5 PCBs were manufactured and assembled. The total cost of manufacturing, electronic components, integrated circuits, and delivery services was 225€.

The last and the most expensive part of the project was sensor housing. It demanded a careful research and market monitoring to find a plastic box manufacturer. The way of housing manufacturing was mold injection. It is known that the greater expense consists of the mold preparation and its cost can reach more than 10000€. After the research and contact with companies from several continents a very cheap and convenient price was achieved, 2800€ for internal and external parts, both. As the plastic material reinforced nylon was selected since it is very tough and can overcome accidental car driving over the sensor. Also, it has good properties against acids which can be found in outside environment and other aggressive environments. The cost of one complete housing with reinforced nylon is 2.53€, thus for 5 sensors it summed 12.65€.

One step before manufacturing was to design a proper sensor housing. This task has little to do with the main focus of the project. Hence, the description of the process and the software used during it will not be described in the present work. However, the design cost will be estimated. It took 5 working days, 25 hours, to carry out the design. If a cost of the working hour of a product designer can be esteemed around 16€ per hour, therefore the human costs will be of 400€.

The rest of the tools and required elements of the successful project completion was found available in the company and summed up no additional costs.

The summary of the project expenses can be found in the Table 3.1.

Table 3.1: Cost summary

Item	
Human costs	Cost (€)
Firmware/Hardware development	4500
Product development	400
Algorithm development	
Arduino-like development board	30
Batteries	7
PCB design and manufacturing	
PCB manufacturing	5
Components and ICs	150
Plastic housing manufacturing	
External part mold preparation	1500
Internal part mold preparation	1300
5 housings manufacturing	12,65
Delivery services	
Components delivery	35
PCB delivery	32
Plastic housing delivery	50
Total	8.021,65

This table summarizes all the costs required for the successful project development.

3.4 Project Monitoring

The date which was initially predicted to finish the project was the September 30, 2018. However, it was postponed due to firmware development of a gateway and new protocol implementation that included RFLink connections handling among other features. Thus, the finishing date was delayed until the October 15, 2018.

Normally, the projects of this kind and others do not follow initial planning and suffer many changes during development. Fortunately, the present project was not the common case and the development went according to initially arranged priorities and through the steps sketched during the first working week with tiny deviations.

There was a small deviation during the algorithm development sprint. After, it was clear enough that there were some vehicles that was not perceived by the sensor a decision to buy a new, more sensible, sensor was taken. It took about 3 working days until the new sensor arrived. During this small period of time the microcontroller used in the final product, that is, TI CC1310, was researched and studied. Pin-out, RF-Link communications, interruptions and way of programming MCU were studied and tested.

After the PCB design was finished there was a place for another potential delay. Usually a firmware developer starts to program the firmware when the device or prototype is assembled and on hand. This way the essential and basic functionality of the device can be tested and

debugged. However, the device was not present, moreover, neither manufactured. Luckily, the company had developed several devices which were partially similar, that is, the same ICs were used. This gave an opportunity to develop libraries for those ICs and learn the configuration and data retrieval processes. Given the explanation above, the potential delay was compensated and it was possible to advance the development.

The PCB manufacturer declared a lead time period of 1 week. Thanks to the availability of another similar devices based on the same architecture during this week a firmware programming was greatly advanced despite of the device absence.

The rest of the development continued calmly without any interruptions until it was finished.

3.4.1 Sprints

The development was pivoted by 6 sprints. In this section each of those sprints are described.

1st Sprint

During the first sprint a new workplace was to be assembled and necessary software, listed bellow, configured and installed.

- Visual Studio Code
- PlatfotmIO and plugins for IoTsens boards
- HTerm
- AVRDudes
- ExtraPutty
- Autodesk Eagle
- Gerber Visualizer
- Texas Code Composer
- Texas Smart RF

Once everything was installed and prepared, a communication with the magnetometer was to be established. Using a small breadboard the sensor was placed on and connected with the development MCU board.

2nd Sprint

The first part of the second sprint was dedicated to achieving a magnetic field data reading.

After the data acquisition was reached the first data recollection and visualization was carried out and a path for further algorithm development was laid down.

The rest of the second sprint was spent planing all possible strategies for vehicle detection given an input of magnetic field raw values.

3rd Sprint

Just after the moment when the first magnetic data were acquired the algorithm development started.

During this sprint many cycles of algorithm development ameliorated the methods of a vehicle detection. Each cycle comprised data recollection and tests of developed algorithm. Recollected data were processed based on mathematics and statistics preparing an algorithm improvement for the next development cycle.

Once all general cases for vehicle detection were taken into account, certain special cases have been worked out, e.g. abnormal vehicle leaving.

Finally, when the vehicle presence detection was assured a place for start of the next sprint was prepared.

4th Sprint

Next to the algorithm development it was necessary to start product development phase.

The weeks dedicated to this were spent designing PCB and plastic housing capable to withstand aggressive environments.

Due to the fact that it was a learning process and I was acquiring new skill, it was necessary to redesign several times the whole board layout because of wrong design principles applied.

Professional support from my colleagues, their highly developed skills in these fields helped me to finish this phase applying correct design principles and rules of thumb which are normally used during development step as such one.

5th Sprint

Developing a product starts from prototyping the one. Thus, all previous steps including this one can be summarized as a prototype development.

My internship was going to be concluded soon, and it was important to finish the development according to the established deadlines. However, during the firmware development I faced several problems which slowed down the progress.

Magnetic field sensor IC was not consuming electrical power described in its datasheet. To find out this issue I had to solder out all ICs on the board and at each step measure the power consumption several times. It should have confirmed to the calculations done before design decisions were taken. This was several days were lost.

By the way, the algorithm implementation was not a difficult step. It required small modifications to pass the code from C++ programming language (Arduino framework) to the C (Texas Instruments framework). Finally, the firmware development was finished with 2 days delay due to the problems described above.

6th Sprint

The last weeks of the internship were spent working on a validation model and testing and improvements of the device. The prior goal of the model was to detect weaknesses in the firmware and hardware and their fixes.

Initially, it was started in the corporate parking with one sensor in the test field. This small procedure served as a preface which allowed to correct the most of the bugs and faults in the firmware structure and prepared a way for further testing but on bigger scale.

This small validation step encouraged us to go outside of the corporate parking to a huge parking in front of the company. We installed there 5 sensor prototypes, which were available at the moment, and monitored there activity.

During the time of the validation the device was not only validated as a product, but its functioning and preciseness were improved as well.

Chapter 4

Vehicle detection algorithm development

Contents

4.1 Introduction 57
4.2 Data Acquisition Process 59
4.2.1 Sliding Window 59
4.2.2 Axes Compensation 60
4.3 Vehicle Detection Algorithm 62
4.3.1 State Machine 62
4.3.2 Achieving Stability 64

4.1 Introduction

Vehicle detection algorithms is a well-known topic among scholars and researchers. There are many approaches and different methodologies elaborated to achieve results with minimum error. It is even possible to create a vehicle signature using magnetic data and use it for recognition purposes. However those techniques require complex computation that lead to excessive power consumption not desirable for embedded systems.

Currently used methods are based mostly on the detection of different phenomena associated with vehicles passing by. The most commonly used are induction loops, videodetection, acoustic detection and radiodetection [18].

Among all existing and currently used solutions for vehicle detection a new one emerged recently, that is one based on magnetic field observations. Advances in magnetic sensor manufacturing demonstrated the possibility to sense magnetic field fluctuations up to micro Gauss units. This allows to create a highly protected product that is possible to place into aggressive environment and it will be capable to detect sufficient magnetic field fluctuations to derive vehicle presence.

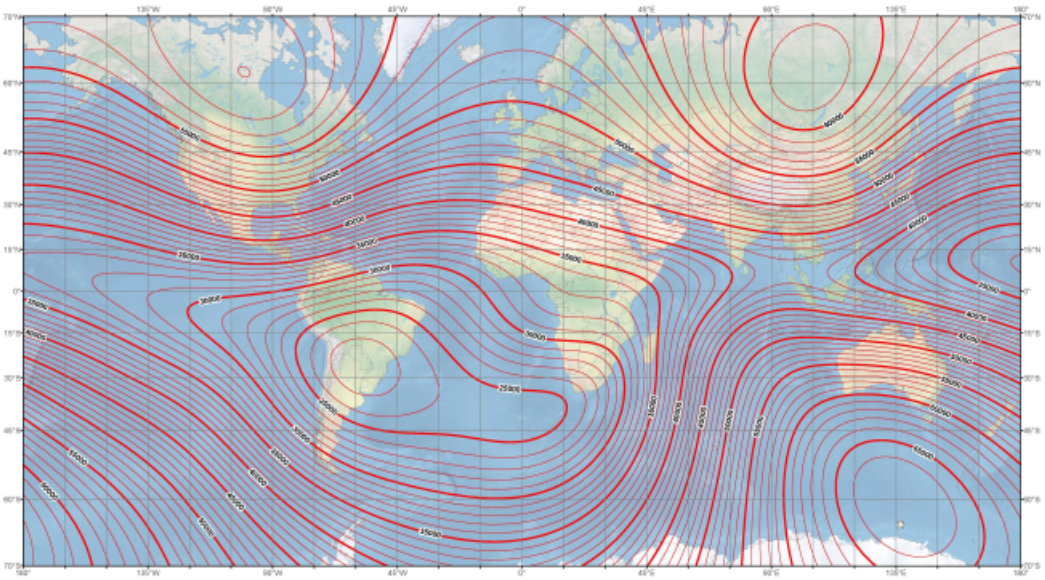

Figure 4.1: The Earth's Magnetic Field Distribution

A uniform magnetic field is provided by the Earth. Its global values maintain within experimentally derived limits and are globally constant. Figure 4.2 depicts the global Earth's magnetic field distribution. In spite of its non-linear nature across the globe it persists constant locally, approximately 2-3 km wide [21].

Maintained constant the Earth's magnetic field is being continuously disturbed by local fluctuations. Those fluctuations are the result of local ferrous objects presence. When a ferrous object is placed into a uniform magnetic field it causes magnetic disturbance which is modeled by a composition of dipole magnets. These dipoles have random but normally unidirectional orientation that create distortions in the Earth's magnetic field [19]. Although the distortions are most evident where the engine and wheel axes located they are also perceivable along the trunk.

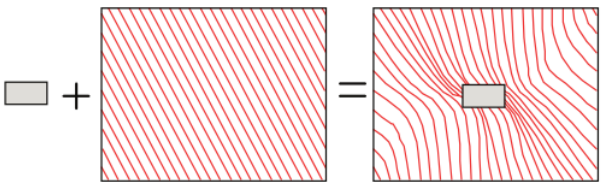

Figure 4.2: Disturbance caused by a ferrous object in a uniform field

A real vehicle is a construction composed by many small and big ferrous metal objects which exercise magnetic field disturbance. This effect can be observed in the Figure 4.3

The algorithm developed in this work is based on the vehicle presence measurement. The main assumption that had been made consists in the reliability of fact that the changes produced by vehicle presence keep on constant in time.

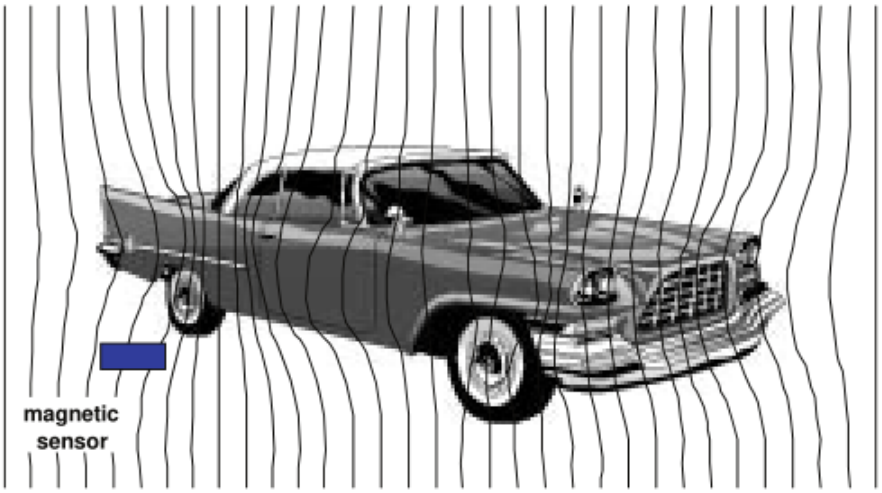

Figure 4.3: Vehicle disturbance in the Earth's field

This chapter is organized as follows. Section 2 presents the state machine which orchestrates the detection process. Section 3 describes the way that the data recollection is made and how the relevance is balanced among different axes. Section 4 describes the proposed vehicle detection algorithm based on dynamically adaptive thresholds and stability seeking.

4.2 Data Acquisition Process

This section describes the way the data were recollected and how the axes relevance is balanced since not all the axes are of the same importance.

Initially one sensor was placed in the corporate underground parking. This node was sampling on 80Hz rate each time when a vehicle approached it.

There are several issues regarding data acquisition that should be solved before going further: noise reduction and magnetic data axes compensation.

4.2.1 Sliding Window

Continuous magnetic field fluctuations result into full of noise data. This noise creates excessive amount of non-sense peaks which have an effect of fake instability.

In order to reduce this effect the use of sliding window is proposed. It consists in averaging of data samples and influencing previous samples on the current one. In other words where is a window comprising N samples and a characteristic value associated to this window, a window average. Each window consists of L samples from previous window and N-L new samples. This sampling is visually represented in the Figure 4.4.

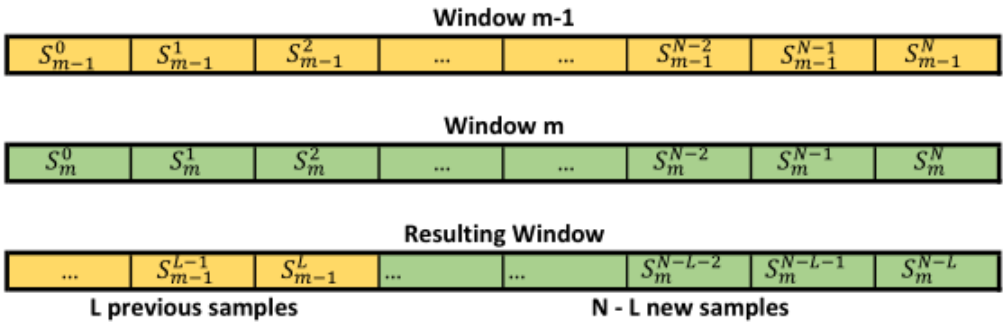

Figure 4.4: Process of sliding window sampling

This process delivers double effect. Firstly, it consistently smooths the curve of sampled data. Secondly, it achieves during the smoothing a sampling memory effect so that current characteristic value smoothed towards the previous window's one.

4.2.2 Axes Compensation

This subsection explains the problem behind the axes compensation. First, it introduces the bases for the given problem and finally, it reveals the problem and provides the solution.

Magnetic field sensors usually sense magnetic field in 3, perpendicular, non-coplanar directions. Those directions are called axes X, Y, and Z. In order to detect vehicle presence these axes have to be unified in some way.

Some example of unification could be one axis selection. It consists in selection of one, the most important axis. Normally, it is the one directed upwards from the earth.

Another example could be a sum of all three axes or magnetic field data vector's magnitude, since there are data from three axes, they can be considered as a vector from 3-dimensional space. However, due to the computational limitation of sensor's hardware in this implementation the sum of squares of the magnetic field axes was preferred. Thus, the chosen unification method is as expressed in the Formula 4.1

$$M = X^2 + Y^2 + Z^2 \tag{4.1}$$

The problem which was found here is unequal relevance of axes data. This problem has emerged due to the sensor positioning respect to a coming vehicle. The initial thought about sensor orientation was as illustrated in the Figure 4.5. From this assumption comes the relation of axes relevance which is given in the Formula 4.2.

$$Z > X > Y \tag{4.2}$$

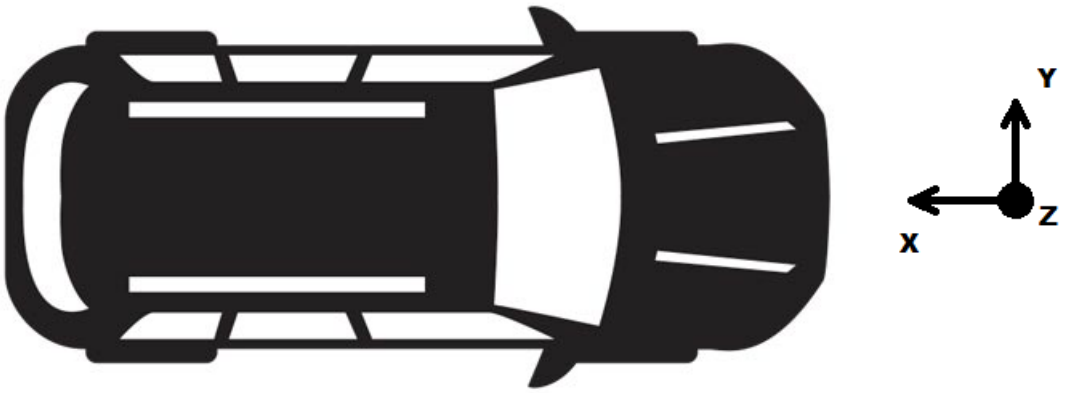

Figure 4.5: Sensor positioning respect to a vehicle

Z axis is the most relevant since it points to the vehicle's trunk. X axis is less relevant, however it is more important then Y axis because it harbours coming vehicle.

The problem comes when in a certain part of the world or as well as in a certain local environment there easily could be given a case when this relation is violated. Therefore, here comes the need of axes compensation.

So as to preserve the desired relation the greatest axis value should be determined and if it violates the relation the compensation value should be derived from it and the relation restored. There are various ways the compensation can be done. The Algorithm 1 describes the way which was used in the project.

Algorithm 1 Axes Compensation Procedure

```
1:  procedure AXES_COMPENSATION
2:      x, y, z ← axes magnetic values
3:      x_comp, y_comp, z_comp ← axes compensation values
4:      min_level ← minimum magnetic magnitude level
5:      if z > x > y then return true
6:      if z! = max(x, y, z) then
7:          z_comp ← max(x, y, z) - z
8:      if x < y then
9:          x_comp ← y − x
10:     if z < min_level then
11:         x_comp ← x_comp + min_level
12:         y_comp ← y_comp + min_level
13:         z_comp ← z_comp + min_level
```

It is worth to note that there is a minimum magnetic magnitude level. This value comes from practical experiments. According to the Formula 4.1 it makes characteristic values more sensitive to considerable magnetic changes.

4.3 Vehicle Detection Algorithm

The proposed algorithm needs to be robust and adaptable to different place and environment. It also needs to be computationally simple due to the limitations put over the sensor node's hardware. The algorithm performs the automatic adaptive threshold (i.e. the threshold is the magnetic level expressed in the unified value) to handle with constant local magnetic fluctuations and to detect vehicle coming and leaving.

The automatic threshold adaption is applied after each measurement whether it is stable or not. It prevents an uncontrollable drift in the magnetic signal due to the local fluctuation and temperature changes. The algorithm is divided in two parts; vehicle coming and vehicle leaving.

As it will be seen from the state machine, the effort to keep things simple has been made.

4.3.1 State Machine

This subsection describes the state machine diagram and will be organized in terms of each state. Each state will be listed; actions and transitions which take place there will be described.

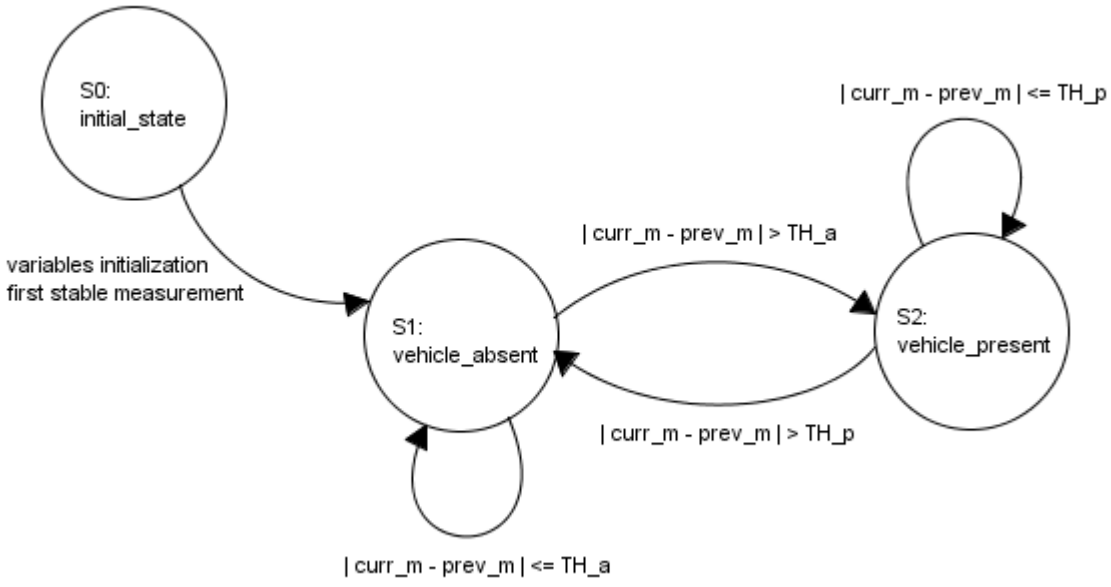

Figure 4.6: Vehicle detection algorithm state diagram

S0: Initial State

The algorithm begins with the S0, initial state, where the system's state must be taken under control and all required variables must be properly initialized.

The most important variables here are sliding window size which determines how many samples of magnetic data should be taken and averaged to be considered a valid measurement, stability window size which use and importance will be explained in the next section and, threshold deviation coefficients.

During the initialization one stable measurement must be taken. The measurement is considered stable once the stability window has been filled. It is necessary to initialize the threshold components.

The algorithm must be initialized when no vehicle is present. It is supremely important because the initial threshold has to be established. The only possible and valid transition from this state leads to the S1 vehicle absent state.

S1: Vehicle absent state

After initializing the variables and establishing first threshold the system jumps to the S1, vehicle absent, state. A new stable measurement is taken and the next characteristic value computed by weighted average of the present measurement and latest characteristic value as shown in Formula 4.3

$$char\_v(k) = (1 - \alpha) * measurement(k) + \alpha * char\_v(k - 1) \tag{4.3}$$

where $char\_v$ is the current characteristic value calculated using the Formula 4.1, given α as a weighting factor. In my experiments α was a value equaling 0.125 to ensure that the noise reduction suppresses undesired peaks. At the S1 state the measurements are being taken continuously as well as threshold is updated in the same manner.

Meanwhile, the characteristic value is updated continuously. When the variation or a difference between previous characteristic value and the current one surpasses the threshold value then it will jump to the S2, vehicle present, state. The variation created by the coming vehicle is stored for the later use and a threshold's portion which corresponds to the vehicle presence is assigned according to the Formula 4.4

$$vehicle\_portion = \beta * coming\_vehicle\_variation \tag{4.4}$$

where β is a weighing factor for the vehicle threshold's portion. During the experiment a value equalling 0.25 was used.

The S1 state represents the state when there is no vehicle detected and an incoming vehicle is awaited.

S2: Vehicle present state

Once the vehicle has been detected, the characteristic values substantially changed in quantity. Also, staying in this state implies that magnetic fluctuations and noise level are steadfast. In this state, the device recollects the data in the same manner as in the S1 state. Meanwhile, it expects the vehicle leaving.

Alike in the S1 state, when the difference between previous characteristic value and the current one surpasses the threshold value then, in contrast to the S1 state, additional computations will take place.

Just before the transition to the S2 state some relevant data have been stored. The variation created by the coming vehicle has been stored for the later use. This variation is the value which has the key role in the vehicle leaving decision.

In the S2 state, when the difference between current and previous characteristic values is proportional to the variation created by the coming vehicle then the vehicle leaving detected. It resets the vehicle portion of the threshold and jumps to the S1 state. The decision condition is expressed in the Formula 4.5.

$$|cur\_ch\_diff - vehicle\_th| < TH\_pl \tag{4.5}$$

where cur_ch_diff is the difference between current and previous characteristic values, $vehicle\_th$ is the variation created by the coming vehicle, and TH_pl is the threshold applied for the leaving vehicle.

The S2 state represents the state when there is a vehicle detected and a vehicle out-coming is awaited.

4.3.2 Achieving Stability

In the previous section the state machine diagram has been exposed and explained in detail. However, there are several details which have been omitted and which deserve to be carefully discussed.

Characteristic value or stable measurement is a term which has been used many times but its meaning and a way it is obtained have not been revealed. Neither various thresholds which have been applied in different parts of the algorithm, nor their dynamic adaption have not been explained.

This section brings to light these details and gives an insight into the related processes.

Stability Window

Continuous magnetic fluctuations, temperature changes, and coming by vehicles are sources of constant magnetic disturbance. Their presence creates doubts about the reliability of a characteristic value. In fact, all the underlying processes does not ensure its reliability, rather improves its quality and coherence. Thus, there is a need of an additional criteria which will provide the desired reliability.

In order to cover this necessity a simple and robust procedure has been made up. Here is where the stability window term comes. The stability window is a requirement for a measurement to persist within the threshold during certain amount of samples. The amount of measurements needed to fill the window is determined by its size. That is, when stability-window-size consecutive measurements have persisted within the threshold, then the average of these values is considered as the valid characteristic value.

The stability window has an important role during the threshold dynamic adaption. The role it has and the process of adaption itself will be discussed further. During the experiments this value has been defined as 8 measurements.

Threshold Dynamic Adaption

So far the reader has found references to different thresholds. The vehicle absence, TH_a, vehicle presence, TH_p, and vehicle leaving, TH_pl, thresholds are three similar in composition though different in purpose thresholds. All the thresholds share the same components or portions.

There are three computed threshold portions shared in one or another way among the thresholds.

1. Characteristic value's percentage. This portion represents a baseline's part that is considered to cover local continuously changing noise level. During the tests and validation process this percentage was equaling 6.125%
2. Derivative average. This portion comes from the characteristic value computation process. Each time when the magnetic field measurement complies to the threshold condition its difference with the previous measurement (derivative) accumulated. When the stability window fills the average of the accumulated derivatives is considered to be a reliable portion of each threshold. It precisely takes in account all peaks and fluctuations and provides more stability to the threshold value.
3. Vehicle presence portion. This part makes sense only when a vehicle is present. The vehicle is a source of magnetic field disturbance. When it is present the level of disturbance is higher contrasted to vehicle absence state. Thus, this portion in the TH_p threshold considerably improves results and provides stability suppressing vehicle magnetic noise.

Once the thresholds' portions have been described the dynamic adaption process can be presented. In order to ease its understanding, it will be depicted using a state machine diagram with explanation provided.

As it has been seen so far, the stability window is a requirement for a measurement to persist within the threshold during certain amount of samples. But what if it does not comply with this requirement? This is the moment when the adaption is the most opportune and needed. Figure 4.7 demonstrates the state machine which leads throughout the way of stability achieving.

S0 is the initial state where the first measurement is to be taken. This measurement is compared to the last stable one and if it does not surpass the threshold value, than it jumps to the S1 state. Otherwise, it jumps to the S2 state.

S1 is the state where a measurement goes under the threshold value and complies to the stability condition. Since the condition has been fulfilled the measurement itself has been saved and the difference respect the previous one has been saved as well.

S2 is the complement state to the S1 state. The important point of this state is that the measurement itself and the difference to the previous one are stored the same way as the S1 state does.

The interplay between these 2 states continues until the stability window is filled whether with stable or unstable measurements. If it is filled with unstable values it jumps to the S3 state, otherwise it jumps to the S4 terminal state.

S3 is the state which corresponds to the situation when the stability window has been filled with unstable measurements. In order to achieve stability the threshold components have to be recomputed. The characteristic's value percentage and derivative average are updated in this step. The percentage is taken from the average of unstable measurements which stability window was filled with and the derivative average is computed as an average of unstable measurements differences. Once the computation of the new threshold has been done, then it jumps to S1 or S2 states according to the next measurement magnitude.

This adaption increases the threshold and brings more stability into the state where high level of noise is produced.

S4 is a terminal state where characteristic value has been computed and the threshold updated. The update of the threshold is performed in a similar fashion as in the S3 state. The percentage is taken from the average of stable measurements and the derivative average is computed as an average of stable measurements differences.

This process is a combination of characteristic value computation and dynamic threshold adaption. It creates sufficient conditions for a developer to rely on the characteristic value, and brings stability environments where it can be hardly reached.

Figure 4.7: Dynamic threshold adaption state machine

Chapter 5

Implementation Details

Contents

5.1 Hardware development 69
5.1.1 Schematic Design 70
5.1.2 Laying out the PCB 74
5.2 Firmware development 81
5.2.1 Identify the Requirements 81
5.2.2 Distinguish architecture from design 81
5.2.3 Manage time 83
5.2.4 Design for tests 83
5.2.5 Future improvements provisions 84

This chapter covers highly important phases of product development: hardware and firmware development. After the vehicle detection algorithm has been developed it is time to create a specific hardware for a device which will deploy created algorithm. Apart from the hardware, it is necessary to create a program which will govern it. This program is called firmware and it is the realm of the algorithm, where its code resides.

The chapter is divided into 2 sections each of which will give an adequate and sufficient insight into the steps which were taken during the implementation and the rules and principles applied.

5.1 Hardware development

Hardware development process consists in a PCB design process. In the introductory chapter it was mentioned that the software used for hardware design is the Autodesk EAGLE. The PCB design is a two-step process. First step is to create a schematic, and second, based on this schematic a PCB has to laid out. It is crucial for the development process to create a well-designed schematic. It allows to easily find errors before the PCB has been fabricated, and it will help during hardware debugging if something does not work as intended.

5.1.1 Schematic Design

Schematic design is a two-step process. First, all ICs and electronic components have to be added on the schematic's sheet. The second part consists in wiring together those components. The steps can be interleaved and partial advances in the development can be accepted also, i.e. adding an IC and wiring it the present components, then adding another IC and repeat the process. During the project the first methodology was used. All components have been added and then wired together.

There is one visual detail which has to be added first, a frame. Within this frame all components are to be placed. Once it has been added and required sections filled one can proceed to add the components.

Among all the components the first one that has to be added is the power input. It is necessary to plan which kind of power supply will be used. In case of the parking sensor it is evident that a batteries suit is the needed power input. Therefore, an appropriate power input connector was chosen and added.

There is a small yet important provision that has been made related to the power supply. The battery suit which was chosen as the main power input is slightly expensive and it was a good idea to provide possible alternative to substitute the main power source. For this purpose the TPS61291 was added. The TPS61291 a boost converter with an integrated bypass mode which can serve as a step down voltage regulator. It provides a possibility to add a variety of power supply sources when a fixed voltage will be constant regardless the power source.

Next step is to add the microcontroller and supporting circuitry. Thus the main component of the PCB was added, that is the Texas Instruments MCU CC1310. According to the datasheet, the CC1310 requires 7 decoupling capacitors of 100 nF which should be placed taking in account the recommended layout. It needs two quartz crystal oscillator ICs: one as a main clock for the primary microcontroller's core and another as a clock for the RF communication core.

As close as possible to the MCU should be placed antenna circuitry. It is an interplay of capacitors and electromagnetic coils of different magnitudes. The specification for the design of antenna system can be found in the same CC1310 datasheet.

After the microcontroller placement the rest of ICs are added: flash memory, temperature sensor, RTC, and magnetometer. Each IC has its own decoupling capacitors and pull up or pull down resistances.

Finally, the external connectors for programming and debugging can be added. The CC1310 uses JTAG as a main connector for programming the device. However it is highly recommended to added the UART connector to trace and debug embedded software.

Since all circuitry were placed, the wiring process starts. It is useful to add labels to the most used traces like VCC, GND, RESET, and others.

The resulting schematic sheets can be seen on the figures 5.1, 5.2, and 5.3.

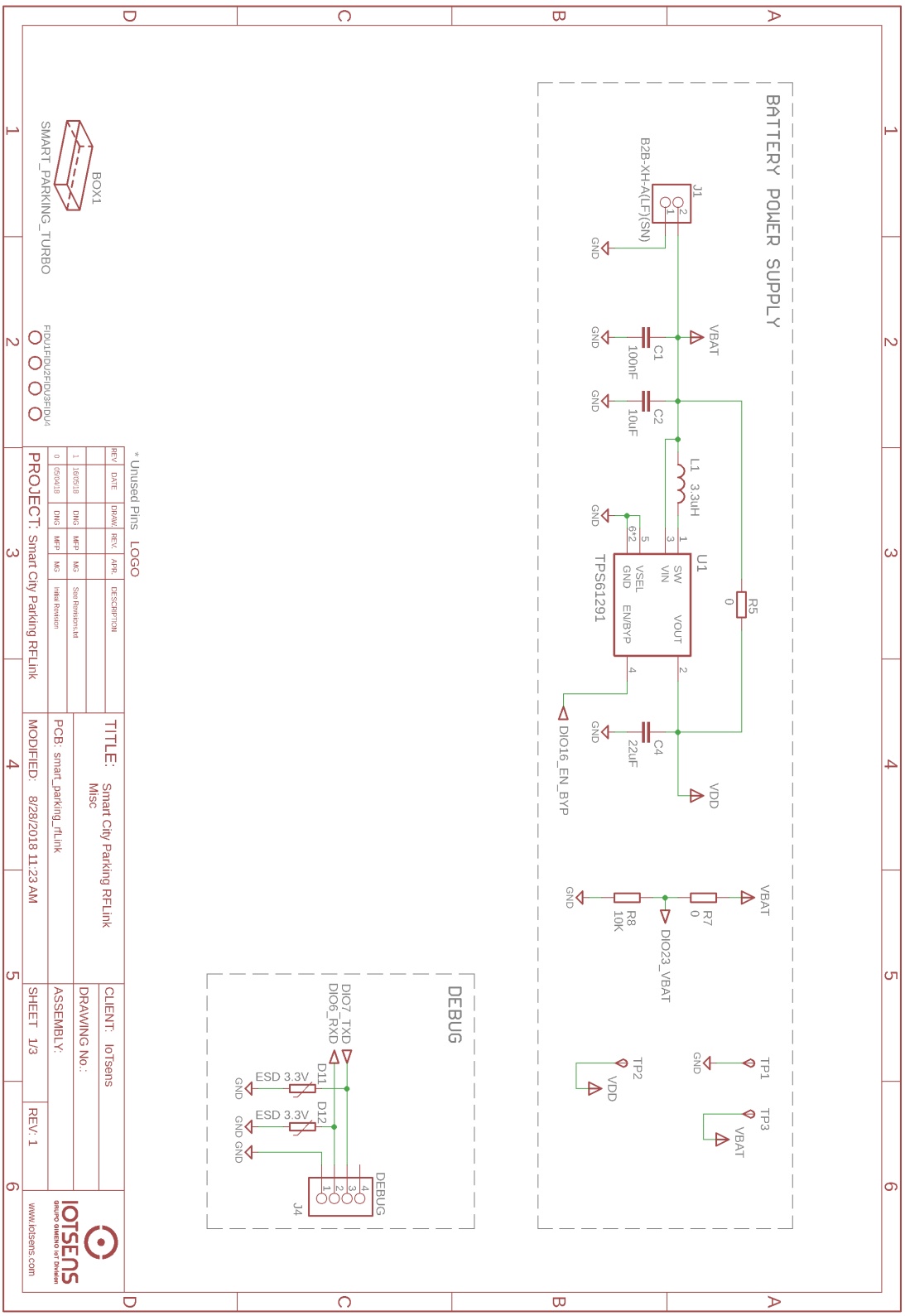

Figure 5.1: Schematic sheet 1

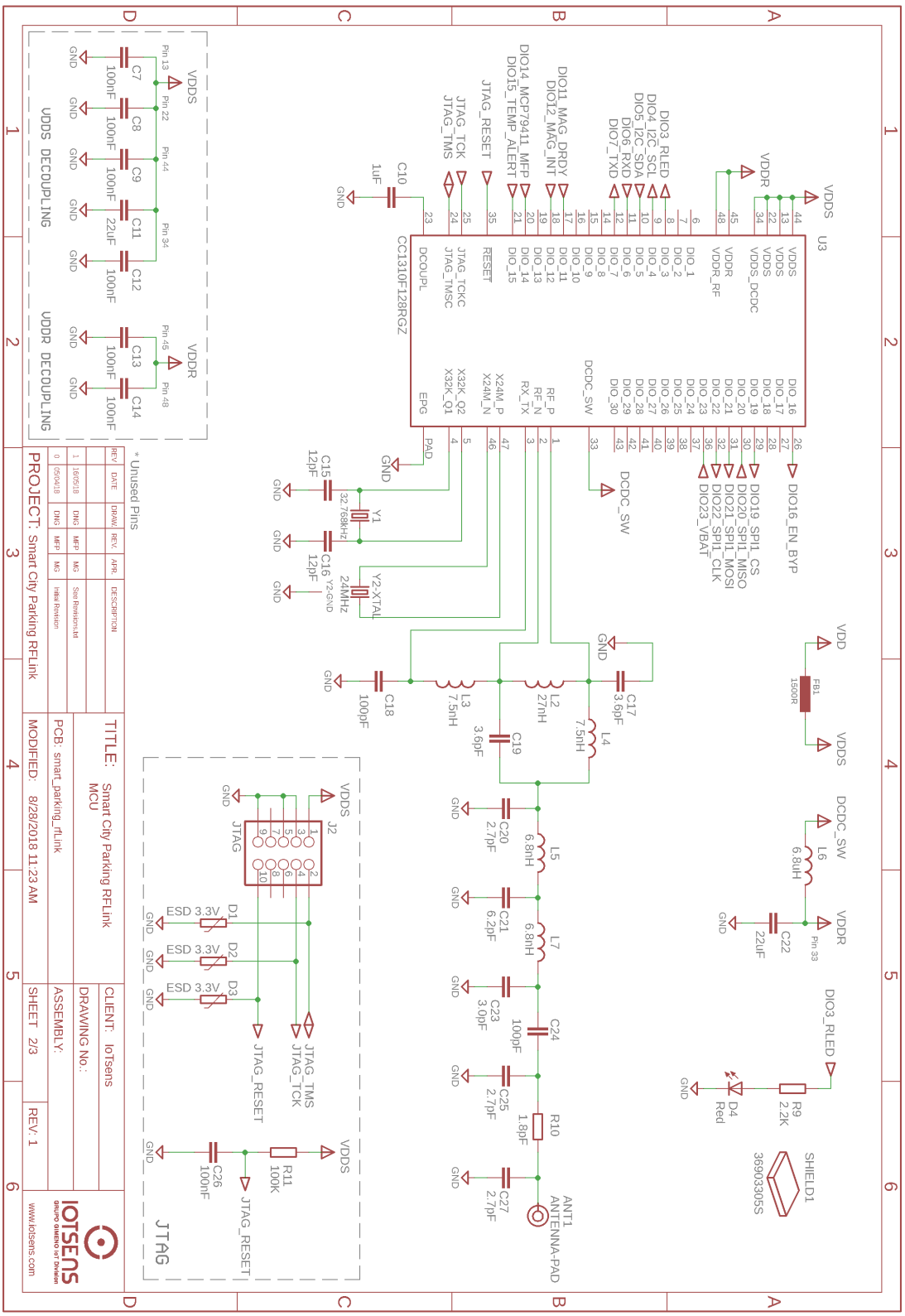

Figure 5.2: Schematic sheet 2

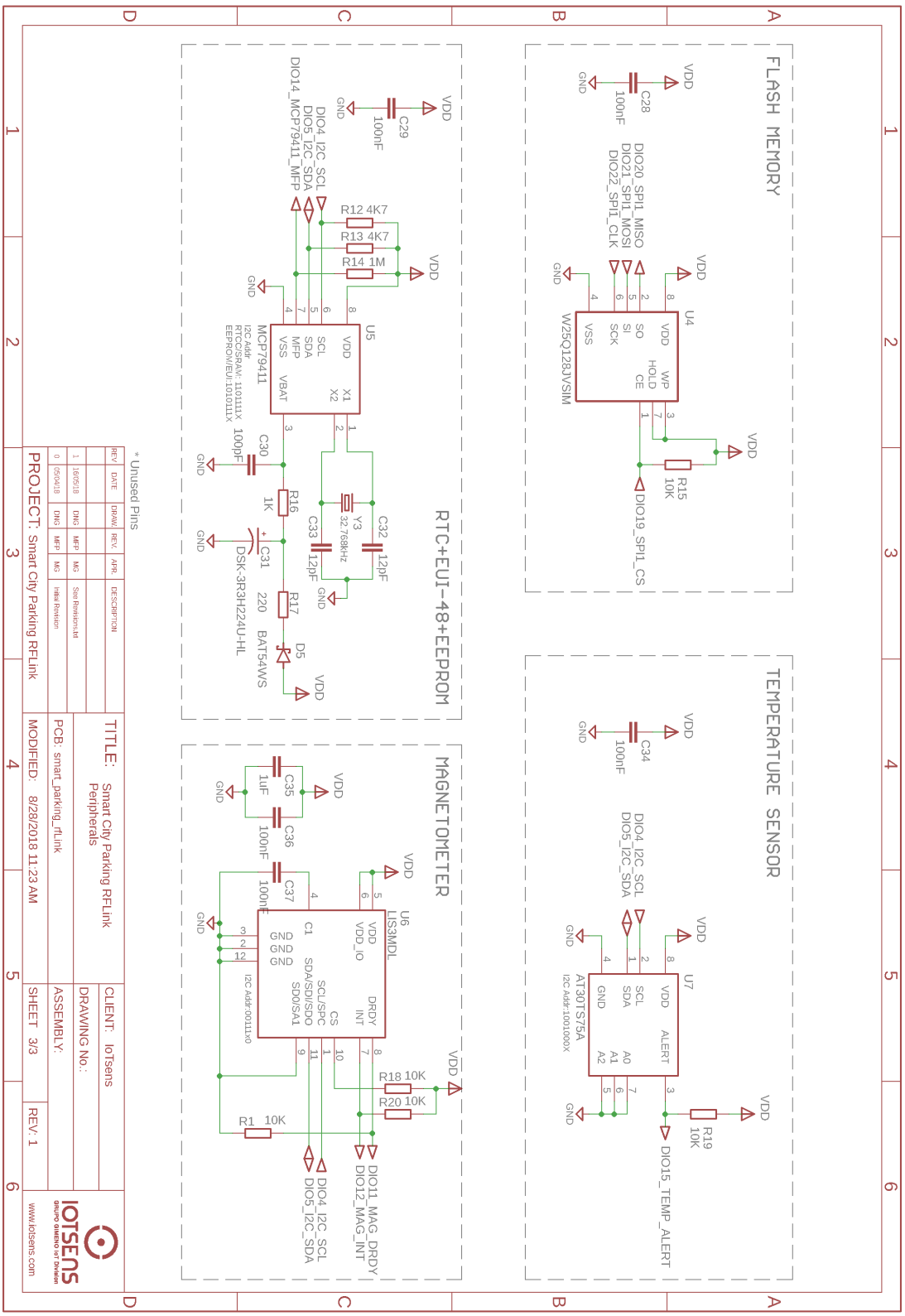

Figure 5.3: Schematic sheet 3

After all the components have been wired, the next step can be taken. From here the PCB laying out starts.

5.1.2 Laying out the PCB

Laying out the PCB is a step where where the form and dimensions of the board come across, components are arranged, and connected by copper traces. In this phase the schematic that had been designed from the previous phase becomes a precisely dimensioned and routed PCB.

Th EAGLE provides by simple Generate/Switch to board command a powerful functionality of automatic board generation.

PCB Layers Overview

Basically, the PCB is composed of one material over another. The thickest, middle part of the board is an insulating substrate. On either side of it is a thin layer of copper, where the electric signals pass through. In order to protect and insulate copper traces, they are covered with a thin layer of a lacquer-like soldermask. This mask gives the PCB color. Finally, on the top of either side a layer of ink-like silkscreen is added. This layer serves to add text, symbols, and logos on the PCB [11]. This composition can be observed on the Figure 5.4.

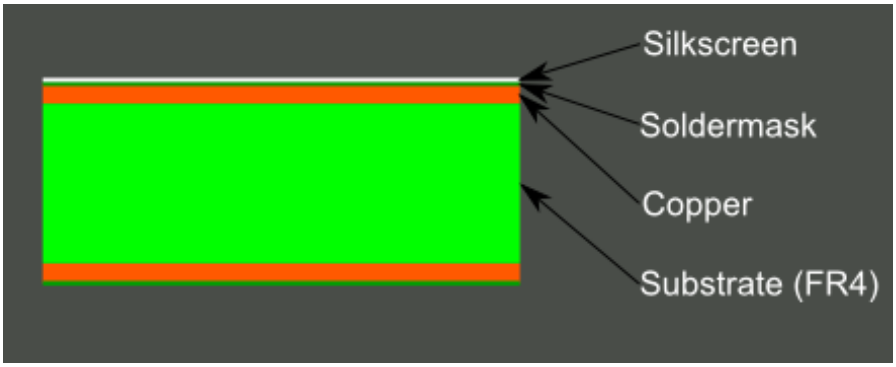

Figure 5.4: The layers composition of a double-side PCB

Routing the board

After the Generate/Switch to board option has been chosen, the EAGLE immediately creates a new board file where all components of your schematic are shown.

Routing the board is similar to solving a puzzle. The job to be done is to turn virtual connections to the up or button copper traces. It is important to not overlap 2 different signals.

First, the power supply circuitry are added. Its final layout can be seen figure 5.5

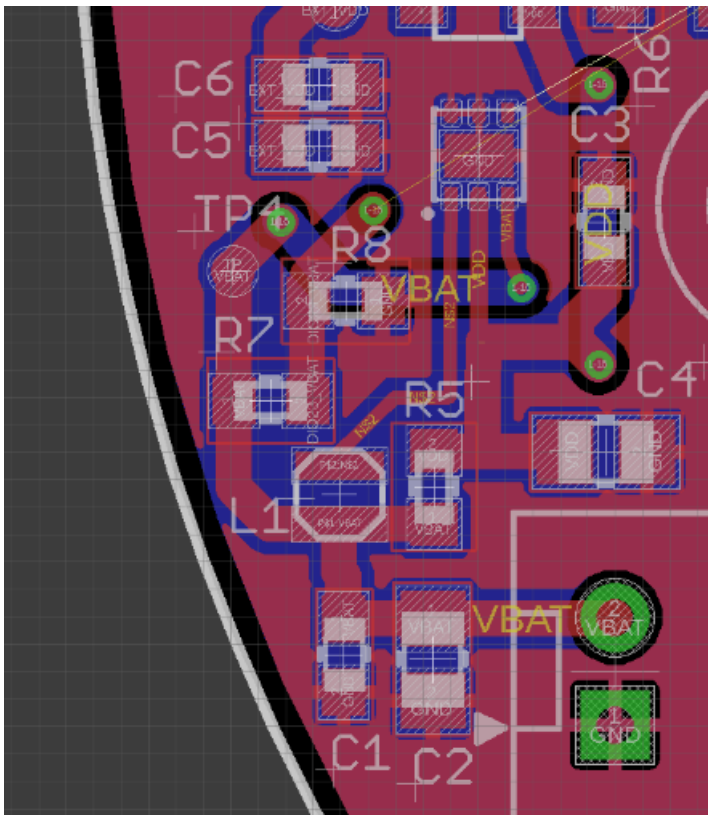

Figure 5.5: PCB power supply circuitry layout

The next component to route is the microcontroller. It is put on the most suitable part of the board. The final layout of the MCU is shown on the figure 5.6

Finally, the rest of the ICs are routed according to the schematic. RTC, flash memory, temperature sensor and magnetic field sensor layouts can be seen no figures 5.7, 5.8, 5.9, and 5.10.

The figures 5.11 and 5.12 provide views of top and bottom sides of general PCB layout.

Coming throughout these steps represents the way the hardware development was carried out. The parking sensors' PCB as it can be seen today is the result of the described in this section steps.

Here finishes the hardware development and starts the phase of the firmware development.

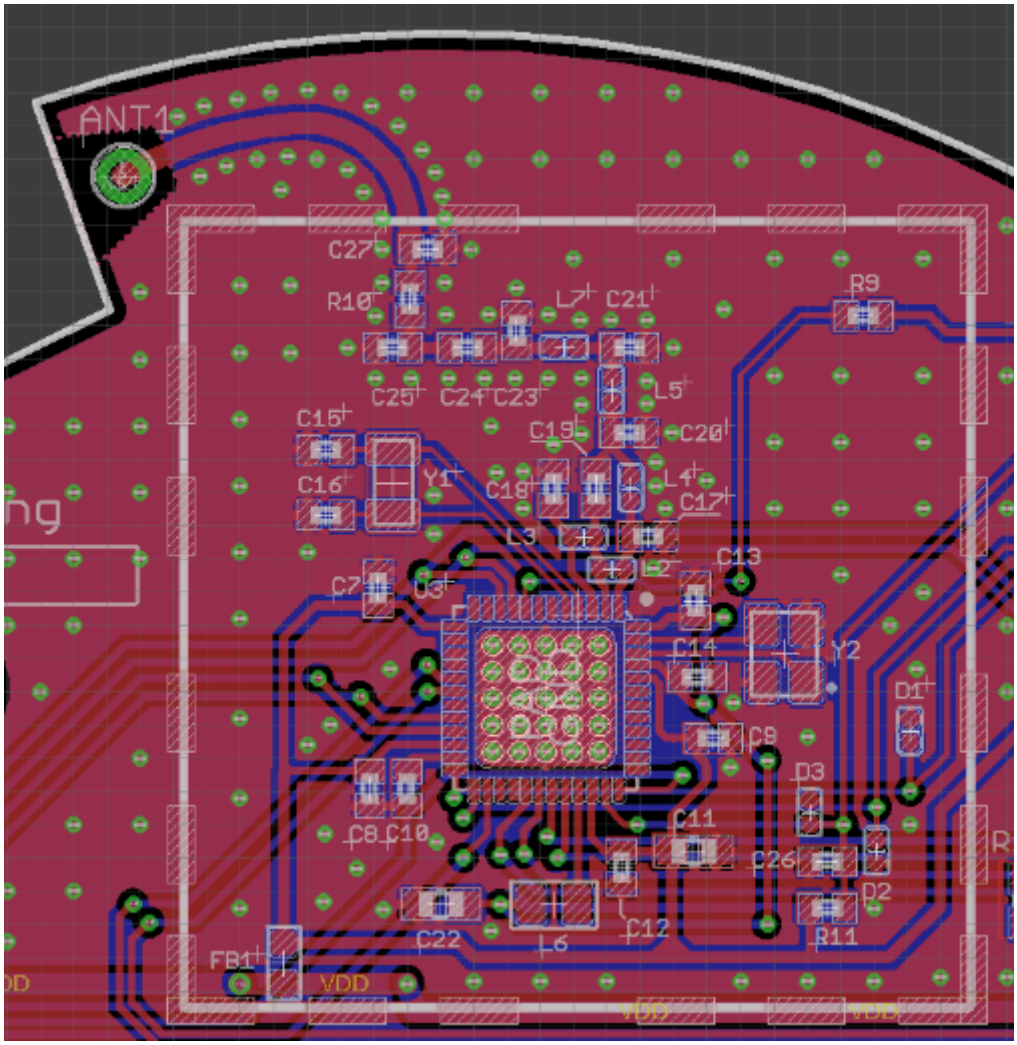

Figure 5.6: TI CC1310 microconroller and RF antenna PCB layout

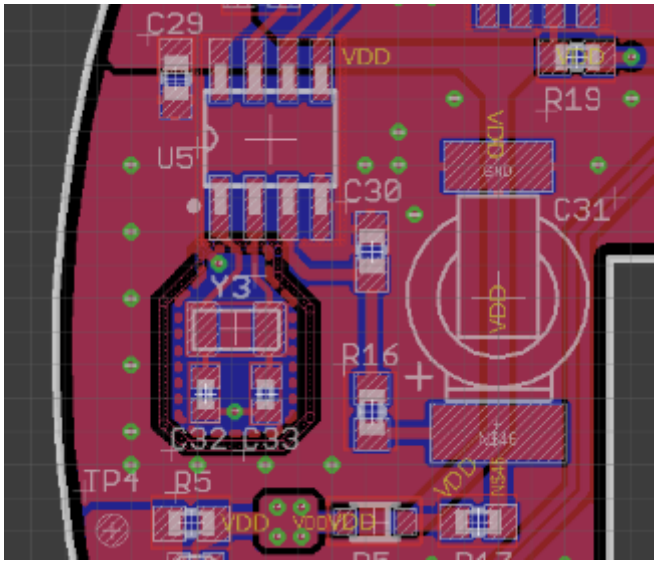

Figure 5.7: RTC final PCB layout

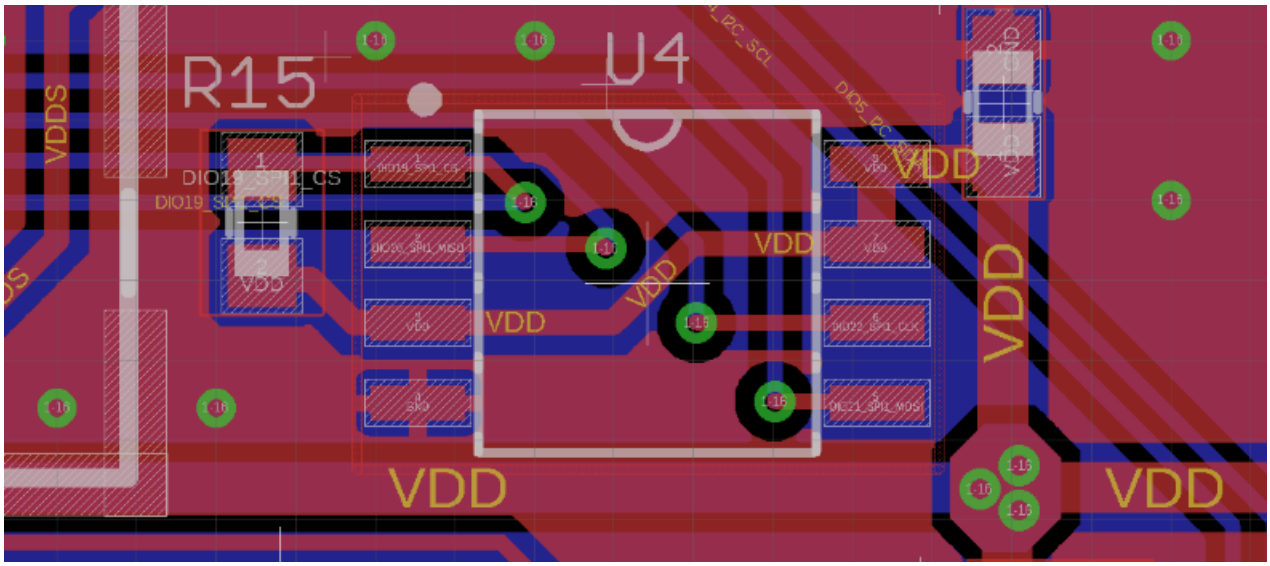

Figure 5.8: Flash memory final PCB layout

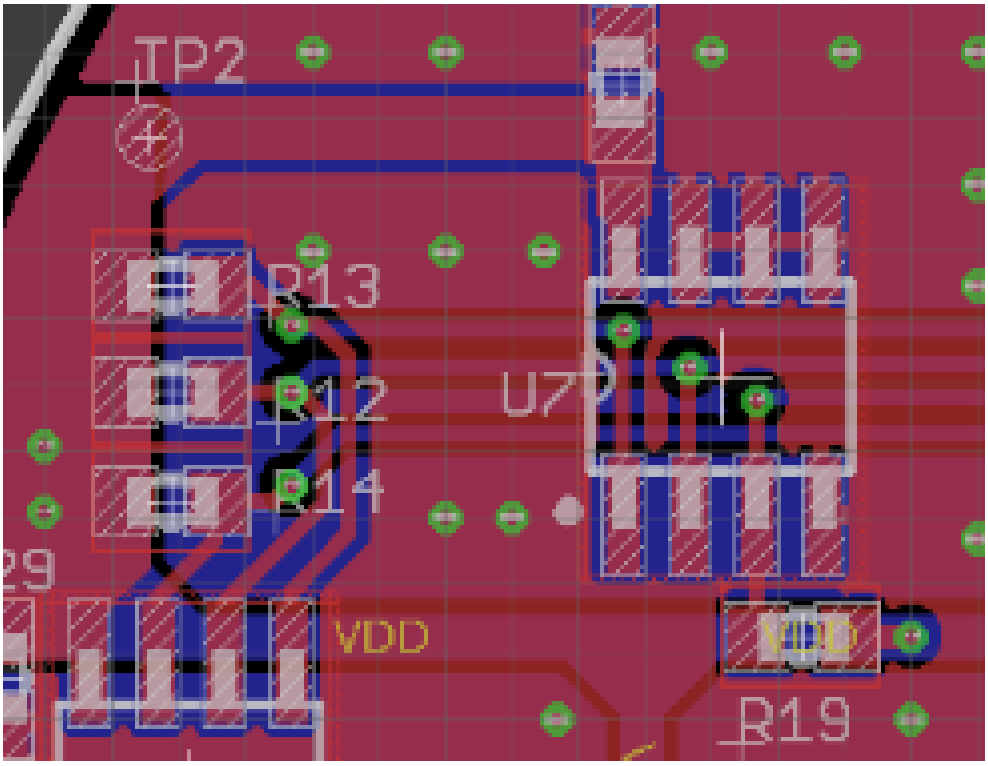

Figure 5.9: Temperature sensor final PCB layout

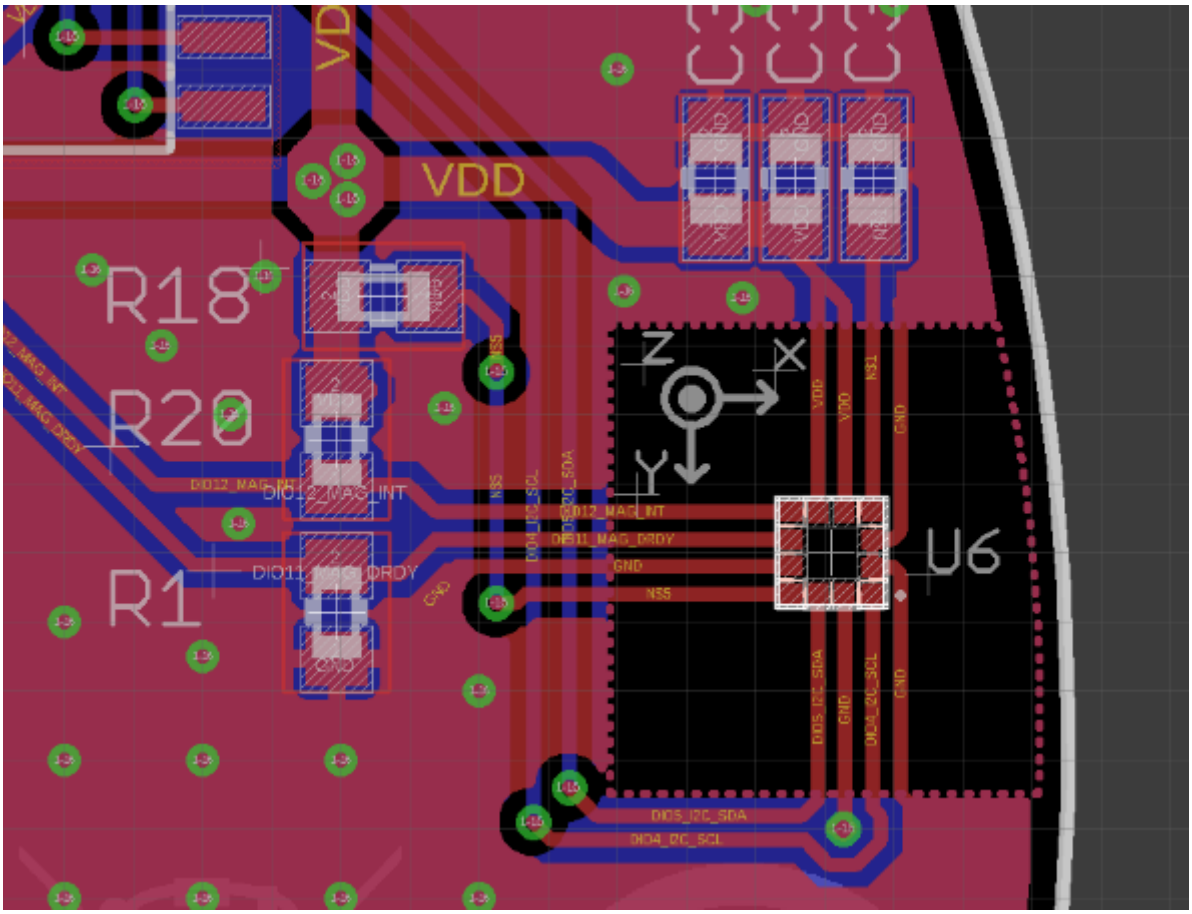

Figure 5.10: Magnetic field sensor final PCB layout

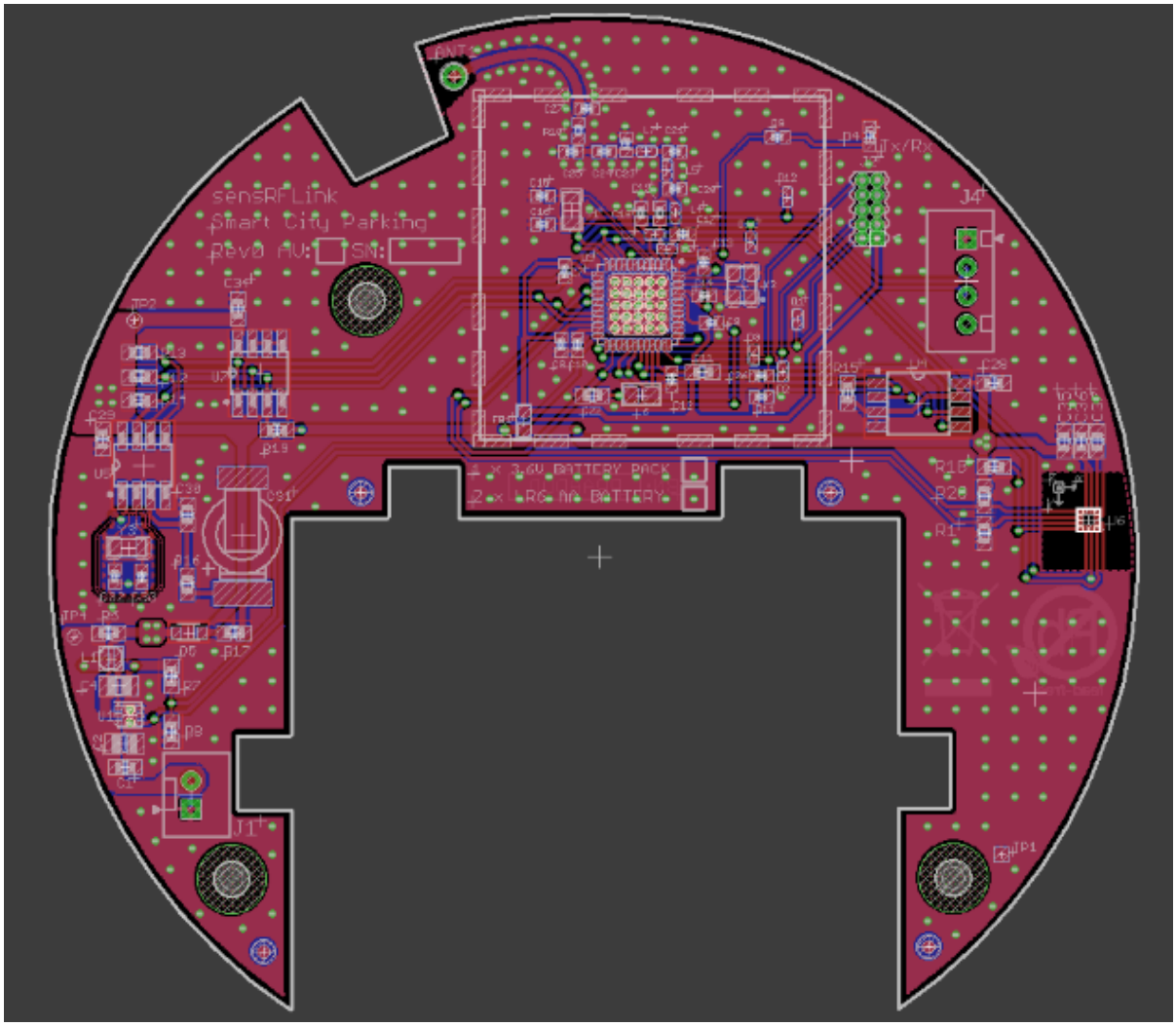

Figure 5.11: General PCB top layout view

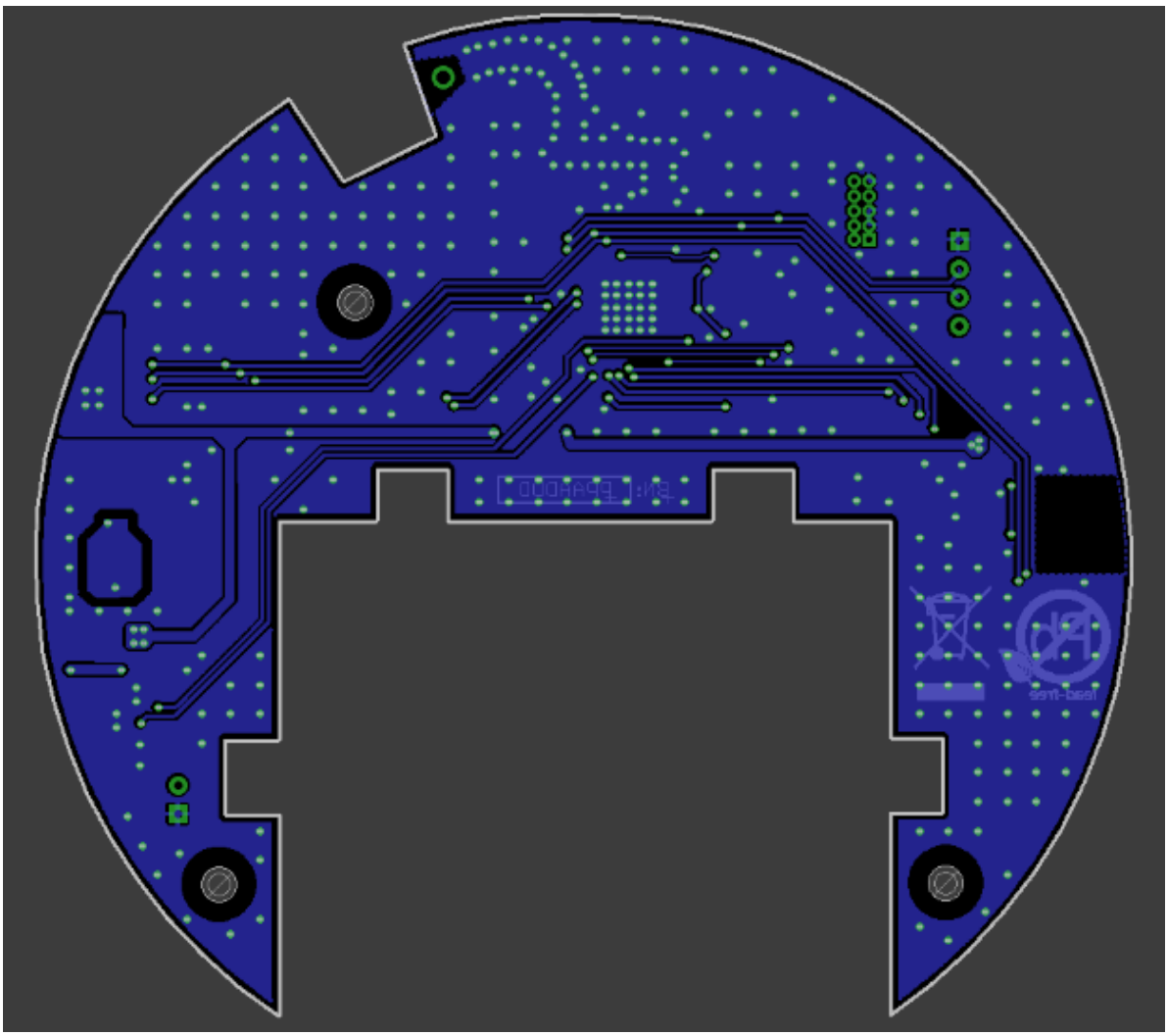

Figure 5.12: General PCB bottom layout view

5.2 Firmware development

Writing code for embedded systems is not the same as programming for a general purpose PC. It is similar to programming a driver for the PC. If there is no operating system, the programmer should take care and control of all hardware system: pins, ICs and sensors. The code should be efficient, minimalistic, stable, readable, and time-predictable.

Normally, embedded software is connected directly to hardware components like motors, relays or external contacts. There are a plenty of ways this software can be implemented. However, there exist good practices that have to be taken into account if the goal is to write high quality firmware code. Those practices are architectural steps following which the team will save many working hours in the future.

5.2.1 Identify the Requirements

Before the embedded software or firmware developing started its requirements must be clear. The requirements respond to the WHAT kind of questions. That is, WHAT the product exactly does for the user? It is notable that those requirements are silent about HOW the specific part of the WHAT can be developed. If there is a need for RTOS (Real Time Operating System), FPGA (Field-Programmable Gate Array) can be used, or there should be a combination of electronics and software.

The requirements must be unambiguous and testable. An unambiguous means that it explains itself and needs no explanation. Testability is the key. If the requirement is properly posed, then a series of tests can be easily created to check if the requirement is met.

The parking sensor had following list of requirements:

- Low power consumption. At least it should have two-years-life using the current battery suit.
- Be able to detect vehicle coming or leaving and report changes of its state within 10 seconds.

Though the requirements list is small, it resumes giant efforts and many tests to achieve each requirement.

5.2.2 Distinguish architecture from design

The architecture describes persistent features. It is hard to modify and must be carefully thought regarding intended and permissible uses of the product. It is best documented via a block diagrams which are connected through directed arrows between its subsystems. Block diagrams identify data flows and shows the boarder between hardware and firmware.

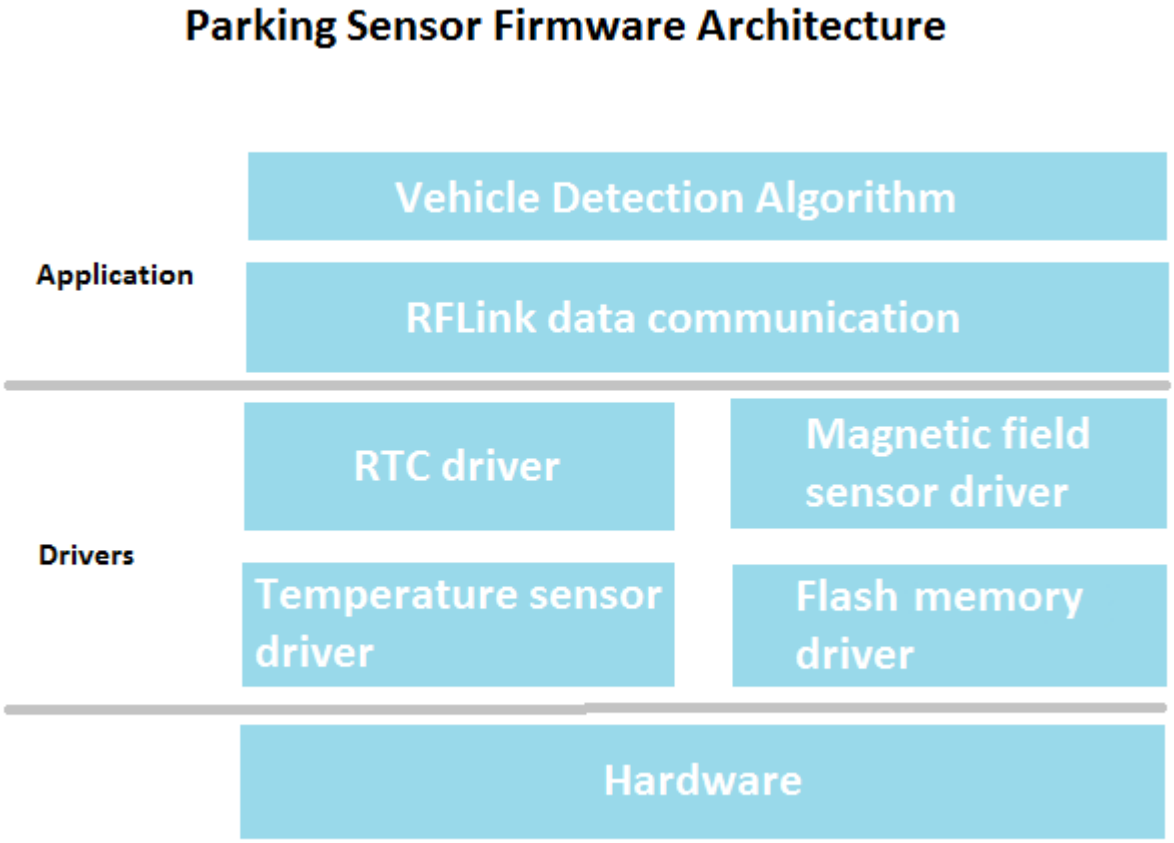

Figure 5.13: Parking sensor firmware architecture block diagram

Diving into the firmware architecture diagram demonstrates subsystem-level blocks such as RTOS, middleware, device drivers, and application components. It should be devoid of confusing details and have a few named components as possible. The architecture is the outermost layer of HOW.

The design is the middle layer of HOW. The architecture does not have specific details like function and variable names. Firmware design document identifies the details such as names and responsibilities of tasks within each subsystem or device driver, details of interfaces between subsystems. The design document establishes task, class, function, variable names agreed by all developers.

The implementation is the lowest level of HOW. The only document which represents the implementation is the firmware source code or hardware schematic. They represent implementation details. If the interfaces were well designed in the above levels, then the embedded engineers could be able to start the development of different subsystems in parallel.

The parking sensor firmware architecture can be observed in the figure 5.13.

5.2.3 Manage time

Since both of the requirements have to do with time, explicitly or implicitly, the architecture should make it easy to meet those deadlines and be certain that they always be met.

Once the deadlines identified, the developer should make effort to push as many time requirements as possible out of the code and onto the hardware.

Figure 5.14 shows the best preferred distribution of real-time functionality. Dedicated CPU or a FPGA is the best place to put a real-time functionality, if it is not possible then it might go down and try a use of interruption routines. If the ISR cannot be used, then the high-priority task should be used [1].

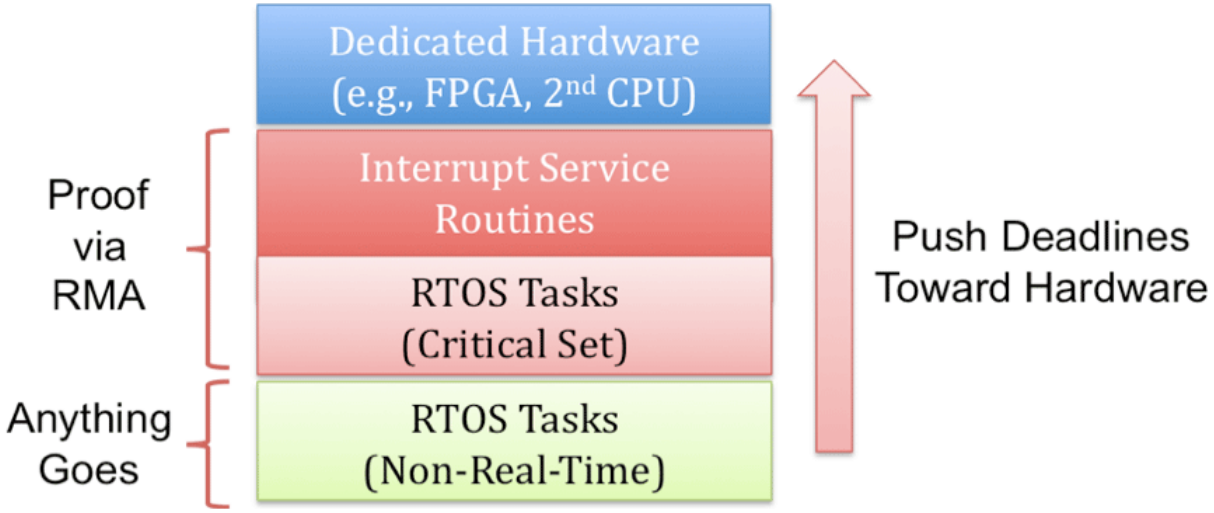

Figure 5.14: Preferred placement of real-time functionality

Fortunately, there is no real-time requirements in the present requirements list. However, there is time requirement related to the vehicle detection process. This requirement was tested and proof to be met during the tests. Many parameters of the algorithm was modified to fit it in the best manner.

5.2.4 Design for tests

Every embedded system and firmware must be tested. Sometimes it is mandatory and in general valuable to perform tests of various levels. Most commonly levels are:

- System tests. Those tests ensure that the product meets or exceeds the posed requirements. Usually, system tests are the part of product validation.
- Integration tests verify that the subsystems blocks of the system interact as expected.
- Unit tests. Those individual tests verify that components of intermediate design level produce reasonable outcome.

By the way, not each step of the development had an associated test. This fact had place due to deadlines imposed by the planning. However, the most important points such as sensor data was carefully tested to meet the requirement of reasonable data acquisition.

Note that for the test and debugging reasons there was added the external UART port. It served as an interface to log events inside the device and show those events to the outside world through corresponding software.

5.2.5 Future improvements provisions

This small point at the first sight seems to be irrelevant, however it is one of the key points during the system architecture design.

To plan future improvements, it is important first to understand the types of changes that are possible to occur. Then architect the firmware in a way that those changes are easiest to make. If the firmware is architectured well, then future changes can be managed just switching through a single software build during compile- or run-time.

The parking sensor is not supposed to be a part of a similar product family. The most demanding change that can be requested is a change of communication technology. This change supposes the hardware modifications. Since the microcontroller TI CC1310 is the wireless one, it has already integrated communication technology, RFLink. It makes little sense to foresee a provision for this change because of the chosen hardware architecture.

However there are possible provisions related to the Firmware Over-The-Air (FOTA) updates. The CC1310 permits the integration of a custom bootloader what can be the best way to add new functionality even to the in-field devices. The improvement does not suppose any firmware changes. It requires a development of a custom bootloader and device programming in a way that instead of the normal start it jumps to the bootloader application first.

These are the steps which were followed to program current version of firmware for the parking sensor device.

Chapter 6

Tests and Validation

Contents

6.1 Hardware Validation 85
6.2 Firmware Validation 86
6.3 Product Validation 86
6.3.1 Vehicle Detection Algorithm 86
6.3.2 System Requirements 88

Verification and validation took place at each step of the product development. This was the preferred way because of it was not possible to start next phase of development without the assurance that the previous phase was developed appropriately. This chapter describes the validation process on each step of development regardless its scale.

In spite of the fact that the project development started with the algorithm development it was not the first to be validated. The hardware which was used during the algorithm development phase was not the same as the one which the final product is based on. It was not appropriate to validate the algorithm with only one device. Those are the reasons because the vehicle detection algorithm validation was postponed for later development phases. Once the working prototypes were assembled it created more favorable and proper condition for the validation.

6.1 Hardware Validation

The first validation step was taken during the hardware development. It consisted of 2 different validation models: continuous validation and final validation.

On the one hand, I had not much experience in hardware development. (Sincerely, I had no experience at all) Therefore the validation was essential during the development. Here, each 3 days my colleagues from the hardware development department, who were my aid and the source of wisdom, validated my advances and decisions. This each-3-days validation followed until the hardware was completely designed.

On the other hand, when the hardware design finished and the PCB was fabricated, the final hardware validation could be done. It consisted primarily in testing of each necessary microcontroller's pin contacts, correct traces connections, and ICs communication tests.

Once the final hardware validation test was deployed it was possible and reasonable to proceed to the firmware or software validation.

6.2 Firmware Validation

Similar to the hardware validation, yet different in essence, there were two kinds of firmware validation: continuous validation and final validation.

Firmware validation was performed according to the section 5.2.4.

Each system component was to produce reasonable outcome, thus unit tests were elaborated to check each component and its proper functionality even changing working conditions.

After all the components were tested, that is small unit tests were written for each one, it was the moment for integration tests. The integration of two principal subsystems was to be carried out: magnetic field sensor was to be integrated into the vehicle detection algorithm as the magnetic data input, and RFLink communication subsystem was to be integrated for the sensor's current state reporting.

The magnetic field sensor integration was easy because the library for the communication with the sensor had been previously written. It consisted in plugging in library functions in the data acquisition process and its apt configuration during initialization.

The RFLink communication subsystem integration was not that easy because it required an interaction with another RFLink node which served as a bridge to communicate the status data to the gateway. This validation test was performed during the last steps of product development.

6.3 Product Validation

Product validation included vehicle detection algorithm and system requirements validation.

6.3.1 Vehicle Detection Algorithm

During the last step of the development there were assembled 5 working parking sensor prototypes. Those devices were installed on 5 different parking spots located in a huge parking lot in front of the Grupo Gimeno's office. The parking lot location and dimentions can be observed on the figure 6.1.

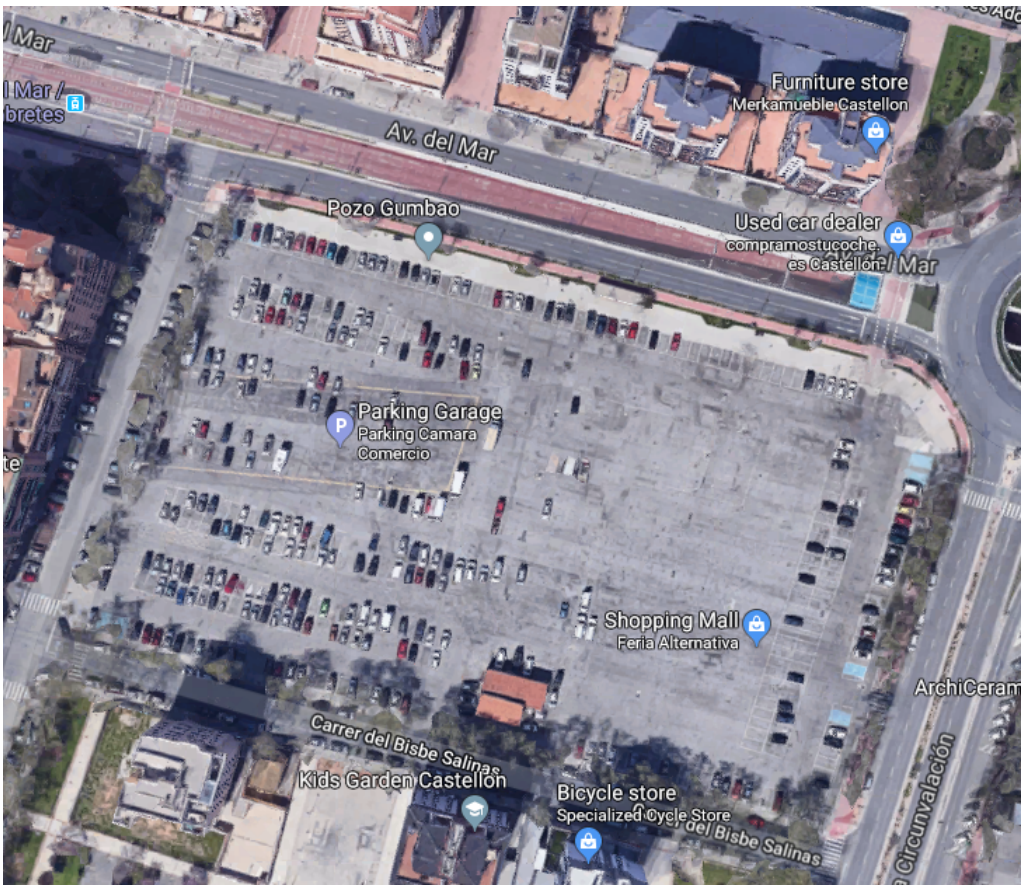

Figure 6.1: Huge parking lot located in front of the Grupo Gimeno's office

For this kind of testing it is important to have reliable Ground truth. For this purpose a small video camera was installed that was recording the place where all 5 sensors were installed. During the tests 247 vehicles were observed: cars(180); pickups(39), and vans(28). The nodes have the correct detection rate of 98.7% for the observed vehicles. (3 missing detections out of 247 vehicles)

Road vehicles	Quantity	Detection rate (%)
True Positive	244	98.7
False Negative	2	0.8
False Positive	1	0.4
Overall	247	100

Table 6.1: Experimental results of the vehicle detection algorithm validation

Performance of the presented vehicle detection algorithm is higher than 98%. However, the missing detections have been found, 2 events during the experiment.

The results of the validation process are quite satisfactory for the time dedicated to the algorithm development. Next step is to validate the system requirements.

6.3.2 System Requirements

The system requirements were clearly defined in the section 5.2.1. This subsection reveals the way that the requirements were verified to be met.

Vehicle Detection Time Requirement

During the development and validation phases the algorithm was optimized to detect a vehicle coming or leaving during 7.3 seconds regardless the time of day, temperature, rain, or any other negative environmental factor. This achievement approves that the vehicle detection time requirement is exceedingly met.

Low Power Consumption

This requirement stated that the system should have a life time of at least two years. How can this requirement can be proved to be met? The answer comes from the power consumption analysis of the system.

The system constantly switching between two states: active state (taking measurements and state changes) and sleep state (no activity). In order to esteem overall power consumption, the time spent in each state and its particular power consumption must be evaluated and summarized.

The active state of the device comprises two stages. The first one is the data recollection and decision making. As it has been placed in the previous section this stage takes 7.3 seconds. The magnetic sensor was optimized to work with 40 Hz output data rate and medium power mode what results in a 145 μA consumption [17]. The second stage is to report the state changes to the gateway. This action requires the activation of RFLink antenna and according to the CC1310 datasheet it takes 25.1 mA during 1 second (actual data transmission) [12].

The active state consumption can be summarized as 145 μA during 7.3 seconds and 25.1 mA during 1 second.

Consulting the same datasheets it can be found that during the sleep state consumption sums up 20 μA.

Table 6.2 recollects the necessary information for the overall power consumption estimation.

If majority of fields of the table 6.2 are clear, some other need explanations. The number of wakeups per hour is estimated according to the number of wakeups per day which is estimated to 6 wakeups. Thus, 6/24 is the factor of wakeups per hour. The derating capacity factor is a good technique to take into account some self discharge and factors like temperature fluctuations.

The formula 6.1 estimates the life of battery.

Capacity rating of battery	1750 mAh
Current of the device when sleeping	0.02 mA
Current of the device when awake	25.2 mA
Number of wakeups per hour	0.25 (estimating 6 vehicle detections per day)
Duration of a single wake	8.3 s
Derating capacity factor	0.85
Life of battery	4.92 years

Table 6.2: Device power consumption summary

$$A_{avg} = ((A_w * T_{wph}) + (A_s * T_{sph}))/3,600,000 \tag{6.1}$$

where A_{avg} is current consumption on average in mA, A_w is the current of the device when awake, T_{wph} is the wake time per hour, A_s is the current of the device when sleeping, T_{sph} is the sleep time per hour, and 3,600,000 is the number of milliseconds are in an hour.

This estimations clearly demonstrate that the low power requirement is exceedingly met.

The validation and verification is not always a simple operation. Depending on the system in can require ubiquity within the development process and additional efforts to check requirements posed by the systems. This chapter exposed all validation steps which took place at each level during the product development.

Chapter 7

Conclusions and Future Improvements

Contents

7.1 Conclusions 91

7.2 Future Improvements 92

7.1 Conclusions

Complete product development cycle was presented in this project. From the first phases of requirements identification throughout all development stages at each level until complete final product verification, an enormous experience has been obtained. The impression given by the supervisor was as desired: the project was finished with all the functional requirements being met. The parking sensor fulfilled all established objectives and presents a ready-to-be-on-the-shelf product.

Personally, I have seen the project as quite opportune and timely. It is true that big cities become bigger each year and more vehicles come or pass through creating a huge road traffic. This implies environmental and many other deteriorations as consequence. The project offers a way to alleviate these environmental and safety threats and even creates a way to gain economic benefit.

On the other hand, the obtained skills, working with different technologies and software are the priceless experience I received. Before starting the project I had never worked with sensors neither with ICs and their programming; I had no idea about hardware design and firmware architecture.

Regarding the internship itself, I felt myself very comfort from the very first moment and incorporated into the team with no delay. My colleagues were the greatest team I had ever partaken in. I appreciated a good working culture and an environment where everyone is ready

to help you, share his experience and tackle with you a problem you are struggling with.

I have understood how a medium size company works, how interact its departments and how important is the communication between them.

7.2 Future Improvements

The project was finished, however there are several parts that can be improved in the future. First, we have seen that the algorithm has high detection rate it still has a margin of improvement. It can be optimized more in detection time and rate. Second, the FOTA update can be implemented in the future. It is a valuable and desired functionality for any embedded system.

At the moment of the product development those are the improvements which can be introduced by the developers in the future.

Bibliography

[1] Michael Barr. Firmware architecture in five easy steps. https://www.embedded.com/design/prototyping-and-development/4008800/4/Firmware-architecture-in-five-easy-steps. [Published on September 21, 2009].

[2] TuFunción Blog de tecnología y programación. Salario medio de un programador. http://www.tufuncion.com/trabajo-programador. [Inquired on November 30, 2018].

[3] ChenYang Technologies GmbH Co. KG Dr J. Liu. Principle of the hall effect sensor. http://www.hallsensors.de/Background2008.pdf. [Published on February 12, 2008].

[4] AutoDesk Inc. Autodesk eagle features. https://www.autodesk.com/products/eagle/features. [Inquired on September 20, 2018].

[5] Microchip Inc. At30ts75a datasheet. http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-8839-DTS-AT30TS75A-Datasheet.pdf. [Inquired on August 28, 2018].

[6] Microchip Inc. Mcp79411 datasheet. https://www.microchip.com/wwwproducts/en/MCP79411. [Inquired on August 25, 2018].

[7] Microsoft Inc. Visual studio code documentation. https://code.visualstudio.com/docs. [Inquired on September 18, 2018].

[8] Motorolla Inc. Spi block guide. https://web.archive.org/web/20150413003534/http://www.ee.nmt.edu/~teare/ee308l/datasheets/S12SPIV3.pdf. [Published on January 21, 2000].

[9] NXP Inc. I2c-bus specification and user manual. https://www.nxp.com/docs/en/user-guide/UM10204.pdf. [Inquired on September 2, 2018].

[10] Plasma Inc. 10 benefits of a smart parking solution. http://www.plasmacomp.com/blogs/benefits-of-smart-parking-solution. [Published on June 27, 2016].

[11] Sparkfun Inc. Using eagle: Board layout. https://learn.sparkfun.com/tutorials/using-eagle-board-layout. [Inquired on February 13, 2019].

[12] Texas Instruments Inc. Cc1310 technical documents. http://www.ti.com/product/CC1310/technicaldocuments. [Inquired on August 16, 2018].

[13] Texas Instruments Inc. Code composer studio (ccs) integrated development environment (ide). http://www.ti.com/tool/ccstudio. [Inquired on September 21, 2018].

[14] Winbond Inc. W25q128jv-dtr datasheet. https://www.winbond.com/resource-files/w25q128jv\%20dtr\%20revb\%2011042016.pdf. [Inquired on August 28, 2018].

[15] Collective Innovation. Scrum methodology: Explained. https://collectiveinnovation.com/scrum-methodology-explained/. [Inquired on January 10, 2019].

[16] International Parking Institute (IPI). Emerging trends in parking study. https://www.parking.org/wp-content/uploads/2015/12/Emerging-Trends-2012.pdf. [Published on December 15, 2015].

[17] ST life.augmented. Lis3mdl datasheet. https://www.st.com/resource/en/datasheet/lis3mdl.pdf. [Inquired on October 2, 2018].

[18] Michal KRECICHWOST Pawel KASPEREK Marcin BUGDOL, Zuzanna SEGIET. Vehicle detection system using magnetic sensors. http://transportproblems.polsl.pl/pl/Archiwum/2014/zeszyt1/2014t9z1_06.pdf. [Published on November 9, 2012].

[19] Lucky S. Withanawasam Michael J. Caruso. Vehicle detection and compass applicationsusing amr magnetic sensors. https://aerocontent.honeywell.com/aero/common/documents/myaerospacecatalog-documents/Defense_Brochures-documents/Magnetic__Literature_Technical_Article-documents/Vehicle_Detection_and_Compass_Applications_using_AMR_Magnetic_Sensors.pdf. [Inquired on February 11, 2019].

[20] Jacob Beningo on ARM community. 10 steps to selecting a microcontroller. https://community.arm.com/iot/embedded/b/embedded-blog/posts/10-steps-to-selecting-a-microcontroller. [Published on January 12, 2014].

[21] British Geological Servey. The earth's magnetic field: An overview. http://www.geomag.bgs.ac.uk/education/earthmag.html. [Inquired on February 5, 2019].

[22] H. J. M. Swagten and P. V. Paluskar. Magnetic tunnel junctions. https://www.tce.co.in/Swagten_Paluskar_Magnetic\%20Tunnel_Junctions.pdf. [Published on April 20, 2010].