Universitat de les Illes Balears

escola politècnica superior

Treball Final de Grau

GRAU EN ENGINYERIA TELEMÀTICA

# LoRaWAN Node For Vehicle Detection Using Pressure

JUAN CARLOS GÓMEZ POMAR

**Tutor**
Bartolomé Alorda

Escola Politècnica Superior
Universitat de les Illes Balears
Palma, August 29, 2017

# CONTENTS

## Resum

En aquest document presentam el treball realitzat a l'empresa IoT Labs S.L. per el desenvolupament d'un node IoT durant el periode Novembre 2016 fins Maig 2017. Aquest node permet orientar als conductors per trobar places lliures al parking. El sensor desenvolupat detecta de vehicles amb variacions de pressió, també transmet i reb dades amb LoRaWAN i finalment, indica les places lliures al conductors amb pantalles LED. El sensor recull dades sobre el fluxe de vehicles en carretera amb possibilitat de que el carrer sigui de doble sentit. Es diferencia entre cotxos i motos, i el sentit de cada un d'ells. El nodes rebran i enviaran trames mitjançant el stack LoRaWAN a la freqüència de lliure llicència a 868MHz. El comput total de les entrades es fà de forma remota en un servidor conectat a internet que serveix aquesta informació tant a usuaris web con als nodes. Una vegada el node rebeix la informació a ser presentada, l'imprimeix a una patalla LED que serà visible per al conductors.

## Abstract

This document presents the works done to develop an IoT node between November 2016 to May 2017 at the company IoT Labs S.L. The developed node helps directing drivers to find free parking spots. The node detects vehicles via pressure, displays the free parking spots with LED screens and transmit the captured information wirelessly with LoRaWAN. To do so, the nodes capture data about the traffic flow, and classify each vehicle that pass by into cars or motorcycles. It also detects the way of each vehicle. The nodes receive and transmit frames using the LoRaWAN stack within the licensed free band at 868MHz. The general operation of the system is realized remotely in a server connected to the Internet which serves both nodes and web users. The nodes receive the number to display within the downlink frames then, they print them on the LED screen that is visible to the drivers.

# ACRONYMS

| | |
|---|---|
| **LPWAN** | Low Power Wide Area Network |
| **LoRa** | Long Range Network |
| **LoRaWAN** | Long Range Wide Area Network |
| **IoT** | Internet of Things |
| **UIB** | Univeristat de les Illes Balears |
| **WSN** | Wireless Sensor Network |
| **DTPM** | Data Transmission Protocol for Machines |
| **ITU** | International Telecommunication Union |
| **IEEE** | Institute of Electrical and Electronics Engineers |
| **PCB** | Printed Circuit Board |
| **ISR** | Interrupt Service Routine |
| **WDT** | Watch Dog Timer |
| **BS** | Base Station or gateway |
| **SF** | Spreading Factor |
| **PS** | Pressure Sensor |
| **3GPP** | Third Generation Partnership Project |
| **NB-IoT** | Narrow Band Internet of Things |

# CHAPTER 1

# INTRODUCTION

The ability to reliably detect vehicles offers significant advantages for asset management, resource allocation and traffic control and monitoring. Moreover, high demand for parking spots create congestion and aggravation among customers. Reducing vehicle congestion, customer aggravation and drive-offs is a must to have a good user experience. One of the key points is directing traffic to available parking. Once drivers are informed, they can easily navigate to empty parking places to access the facility.

Application needs and deployment requirements can be diverse, ranging from indoor, outdoor and partially protected deployments. In this document it is shown the particularities outdoor installations require of isolation and more procedures.

A fully range of sensing technologies to overcome these challenges are currently available and can be used with wired or wireless products to simplify deployments. The use of wireless communications is specially useful in large areas or where wired infrastructure is not practical or cost-effective. A great amount of work and research is already done for understanding the requirements and needs of Wireless Sensor Network (WSN). LoRaWAN is novel communication standard for Low Power Wide Area Network (LPWAN) which facilitates the development of WSN. The final node uses also other communication standards: RS-485, Data Transmission Protocol for Machines (DTPM) and serial among others.

The detection mechanism is based in the pressure variation when the air is compressed within a slim tube by the wheels of the vehicles. In order to understand the behaviour of the pressure of a slim compressible elastic tube when it is exposed to vehicles passing over, an experiment was conducted. After succeeding in the experiment, the prototyping phase was boosted with collaboration of Laboratori d'Electrònica de la Univeristat de les Illes Balears (UIB). Our vehicle detection solution node is ideal for ports, traffic and parking lots that lack of wired infrastructure.

**Goal for this project** The main goal is to explore different technologies such as the LoRaWAN stack for developing WSN and similarly, use pressure variations to describe the response of vehicles. This project is embedded in the IoT Labs S.L. process of finding the most suitable wireless technology for wireless networks depending on features as the network capacity, reliability, coverage, cost and power consumption per node.

In this document, we describe the following phases: the design, implementation, deployment and finally the results obtained about the sensors of this parking system, that indicate us the reliability of this project.

## 1.1 Work in Progress

This document will only present the first product in deployment. A further development is being on to minimize the production costs and simplify the installation and maintenance processes. It would also necessary for upgrading the product to add new requirements as power consumption optimization and extend this solution to other transmitting protocols both wired and wireless. Some of this proposals are contemplated in the last chapter.

## 1.2 Document Structure

A general introduction to the Internet of Things (IoT) and vehicle detection state of the art is presented in chapter 2. Then, we expose the main characteristics of the design in chapter 3. Chapter 4 explains in detail the implementation of the previously described design. A deployment case study is done in chapter 5. Following with a general overview of the project and results presented in chapter 6, the system is evaluated in in chapter 7.

CHAPTER 2

# State Of The Art

In this chapter is presented the state of the art in the three main aspects covered in the document. In section 2.1 the mechanism used for detect the vehicles are exposed. Then, in section 2.2 it is shown the IoT definition and communication standards. Finally, in section 2.3 we cover the latest available general purpose microcontroller to use within IoT.

#### Input Type

Usually, the typical input for a parking is exclusively cars, nonetheless, there are multitude of parking lots that are reserved for motorcycles, bicycles and buses.

In this project, we will only consider the following vehicles: buses, buses with 3 axis also known as accordion buses, car, motorcycles and bicycles, figure 2.1.

For every one of this vehicles there is a typical way of knowing the quantity of them that are in a location. However, parking solutions are mostly presented for cars. It also exist for city public bicycles as shown in figure 2.2.

There is no standard nor de-facto standard for getting information about the kind of vehicle that the inputs are.

## 2.1 Vehicle Detection Mechanisms

The underlying physics used for detection mechanism vary substantially from system to system. Most of the systems take profit from the disturbance produced by ferrous materials on magnetic fields. Some systems use ultrasounds because it is a good mechanism for proximity detection. Others, use its big opaque volume to interfere with infra-red light. We base our detector on the pressure variations induced by the wheels in tubes laid on the path.

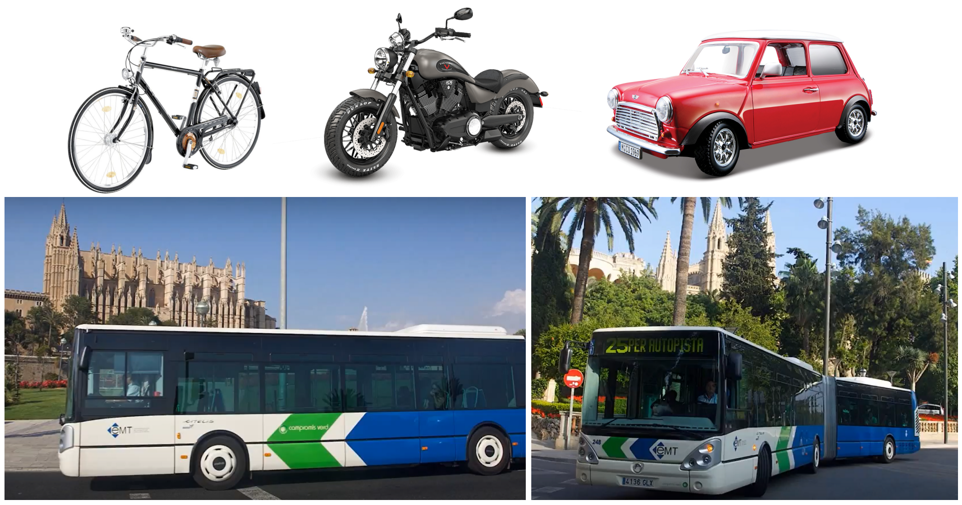


Figure 2.1: Kind of vehicles considered in this project. In order, left to right and upside to down: bicycle, motorcycle, car, bus and 3-axis bus. Images from [2].

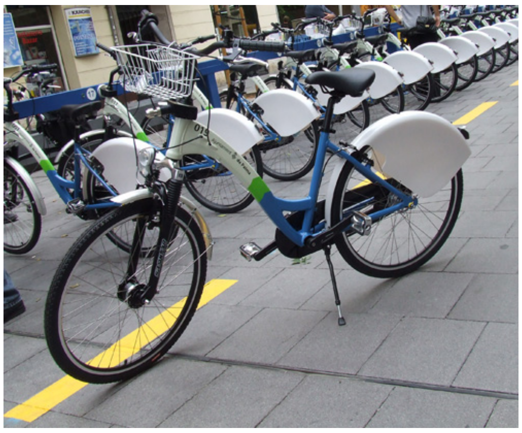


Figure 2.2: Bicycle detector and holder for public bicycles using strong electro-magnets. Image from [3].

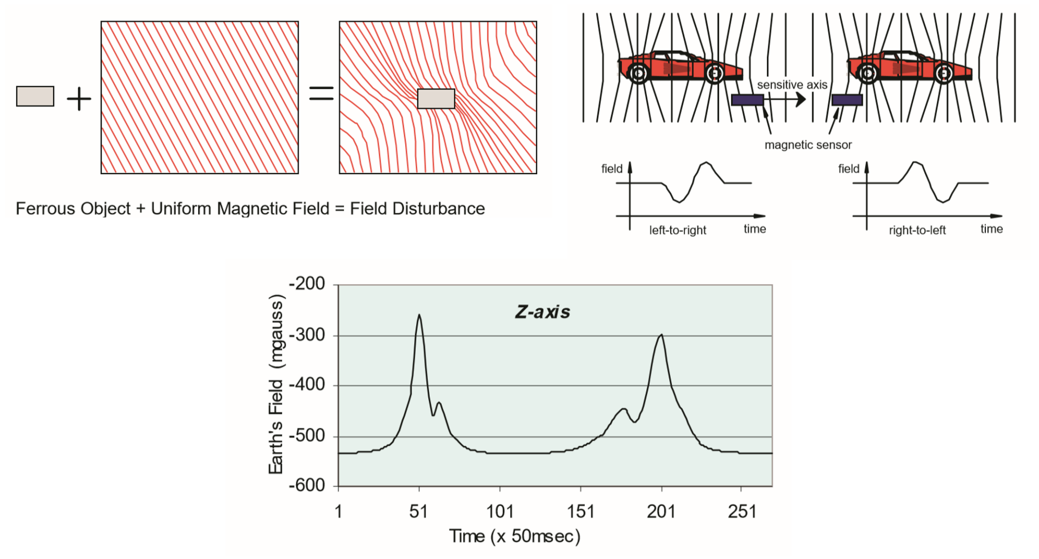


Figure 2.3: Detecting vehicles with magnetic field disturbances. Earth's magnetic field variance due ferrous object passing by. Figures from [4].

### 2.1.1 Magnetic Field Disturbance

Nowadays, this is the preferred method to detect vehicles. This technique is used for detect vehicles passing by in the motorway and also used for detecting parked vehicles. This method does not disturbe the drivers. For instance, the sensor used could be an anisotropic magneto-resistive (AMR) sensor [4] that is used to detect disturbances in the earth's magnetic field created by ferrous objects like vehicles, figure 2.3. And then, it takes advantage of these anomalies to detect traffic and classify vehicles.

This technology is already exploded and multitude enterprises offer this product. It is known as Vehicle Loop Detector, figure 2.4. There is an example of this implementation in the Carretera de Valldemossa, Palma near UIB images 2.5.

We can see in figure 2.5 that there are two sensors for lane, being eight the total. All of them are connected to the controller in the big display. With this system they can have information about flow and congestion in the path.

### 2.1.2 Pressure

We introduce in this work a novel way to detect vehicles and direction. This mechanism is based on the compression of air transmitted through a tube. It is known that when the air is compressed by reducing the volume of its container the pressure increases. Then, by locating a pressure sensor in the edge of a tube this variations on the pressure can be easily be ranged in a voltage scale.

The changes in pressure are made by the wheels of each axis of the vehicle as they pass over the tubes. An experiment was made in 4.1 to test the viability to detect the direction and vehicle type.

Once in the voltage domain, it is possible to classify the responses by the intensity and duration. Then, we can differentiate the kind of vehicle. We will show in our

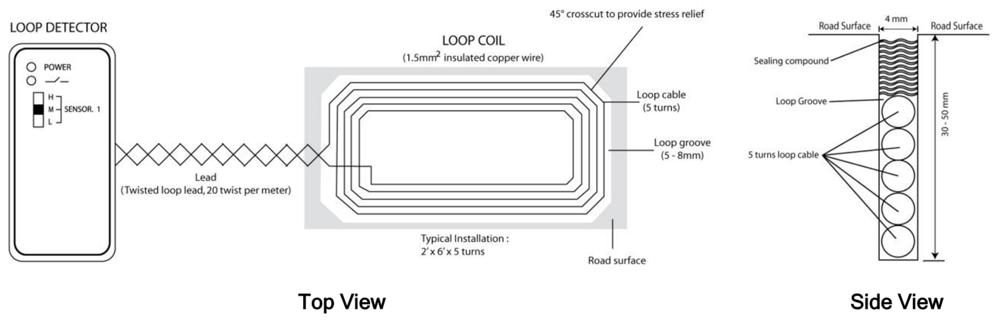


Figure 2.4: The Earth magnetic field is considered uniform arround the coil. When a ferromagnetic object (e.g. the car) passes by, it concentrates and disperses the strength of the magnetic field causing a voltage response in the terminals, in concordance with the Faraday-Lenz law. In order not to break the coil, the loop coil is buried under a tire of sealing compound. Image from [5]

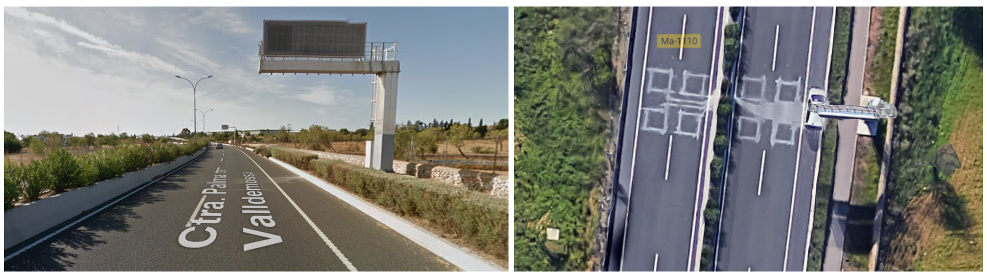


Figure 2.5: Vehicle Loop Detector on the road. Image from Google Maps

study that in most cases this classification is indeed useful to map the responses to the different vehicles.

### 2.1.3 Others

#### Ultrasound

This technique is largely used for detecting free car spots in indoor parking. They need to put one ultrasound detector for each spot. Typically the detector is put in the ceiling pointing to the ground. When the spot is idle, the ultrasound wave goes from the ceiling to the floor and returns to the detector using $2 \cdot v_{sound} d_{\text{ceiling}}$ seconds. When a car is using the parking lot, the time that the sound takes to go and return is severely reduced.

#### Barrier and Ticket Machine

Although it is widely deployed for controlling the entrance and free places of parking around the globe, it is not really a detection mechanism as it is usually triggered by a button pressed by the driver of the car.

Nowadays, it is upgraded by taking a picture of the car and process the image with computer vision to get the license plate number [6], so it can know the use of the parking by end-user car.

#### Laser Reflector

Even though it is not strictly attached to vehicle detection, it is used along with barriers to avoid collision when the barrier closes. The detection is based on the opaqueness of the mobile.

If it were to exist a detection system based solely on this effect, no distinction could easily be made to differentiate cars from buses, motorcycles or even pedestrians.

## 2.2 Technologies for Wide Wireless Sensor Networks & IoT

In this section we will cover the main transmission technologies that are competing in the WSN field. The technologies presented are:

- All generations of mobile technologies 2.2.2.
- NB-IoT 2.2.3.
- Sigfox 2.2.4.
- LoRaWAN 2.2.5.

We will see why the telephony infrastructure is inconvenient for this WSN project. More specifically, we will describe and analyse this new technologies, Sigfox and LoRaWAN, that are included in the so-called framework of LPWAN.

The LPWAN protocols and techniques have emerged to fulfill the following communication requirements [7] [8]:

- Low power consumption for end-devices.
- Long range radio coverage: usually extensions up to 10-15 Km.
- Easy-to-implement and cheap transceivers.
- Rapid scalability to help growing the application.
- Security in communications is a very relevant issue.
- Few information to send, even less to receive.

### 2.2.1 IoT & WSN

The concept of IoT and WSN aren't new but often misused. We present a IoT definition by the International Telecommunication Union (ITU) and a clarification of the Institute of Electrical and Electronics Engineers (IEEE) about this issue.

#### How does the ITU understands IoT?

The ITU [7] defines the IoT paradigm as a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies. This includes:

- Machine to machine (M2M) communications.
- Autonomic networking.
- Data mining.
- Decision-making.

#### How does the IEEE include WSN within IoT paradigm?

The IEEE in the "Towards a definition of the Internet of Things" [8] differences IoT and WSN.

They declare WSN as a spatially distributed network of autonomous sensors that monitor physical or environmental conditions and cooperatively pass their data to a central location.

Nevertheless, they see IoT as a system which scope adds smartness to the objects so they can do the work of actuation to achieve a certain goal without human intervention and the connection to the Internet is a necessary feature.

Finally, the IEEE exposes that the WSN can be one part of the IoT in that sensors used in a IoT system can be networked to achieve a coordinated result.

### 2.2.2 Mobile Technologies: GPRS, UMTS, LTE, LTE-A Pro & LTE-M

The mobile communications 2G, 3G and 4G are deployed and maintained worldwide for end-user devices such as telephones and the so called smart-phones. They base their infrastructure on the use of the private bands of the electromagnetic spectrum to transmit.

The technologies that are used for mobile networks are dimensioned for enabling multiple telephone conversations and for downloading web pages, images and videos. Their motivation was spectrum efficiency and throughput.

The implementation of the latest LTE technologies up to the release 12 are highly challenging and expensive [9]. They need from high digital signal processing techniques and synchronization. For this reason they are not suitable for IoT nor WSN. Long Term Evolution for Machine communications (LTE-M) is the solution presented by the Third Generation Partnership Project (3GPP) on the release 13 and is candidate to support IoT in current LTE cellular networks.

LTE-M has an incredible advantage in front of the technologies that we will see ahead, their infrastructure is already deployed and highly reliable. Nonetheless, this technology is deployed on licensed frequencies which ultimately represent a periodic cost per device to the service.

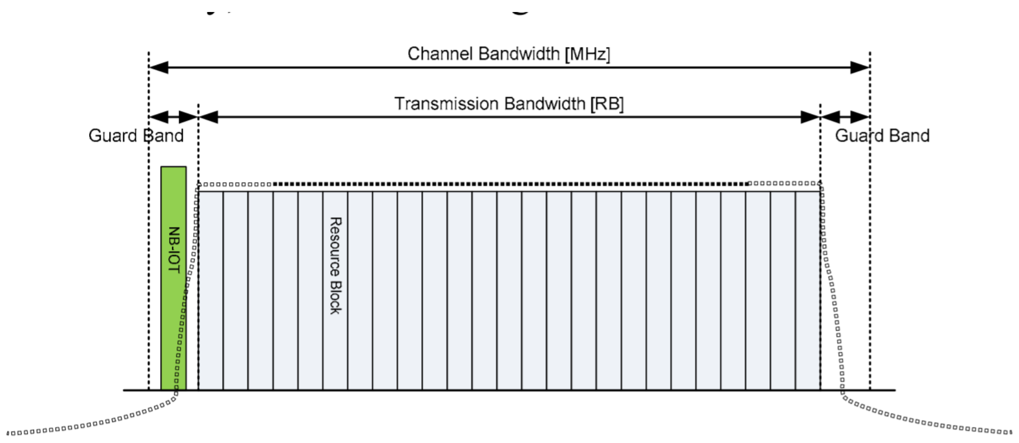


Figure 2.6: Guard-band operation for NB-IoT. Image from [11].

### 2.2.3 NarrowBand IoT

The Narrow Band Internet of Things (NB-IoT) is standard of the 3GPP. The NB-IoT specification was frozen at Release 13 of the 3GPP specification (LTE-Advanced Pro), in June 2016 [10]. NB-IoT is the approach of 3GPP to IoT allowing the production of low-power, low-cost communications with narrowband techniques. The design of NB-IoT networks targets the following operations:

- Improve indoor coverage. 20dB compared to legacy GPRS.
- Massive number of low-throughput devices. 525447 devices per cell.
- Improve power efficiency. Maximum transmit power per device 23dBm.
- Latencies less than 10 seconds for 99% of the devices.

Although it can be deployed on stand alone bands or within an LTE bandwidth, it is expected to be deployed on 180 kHz bandwidth in the guard band of LTE deployments, figure 2.6.

At the day of starting this project November 2016 there are not commercial devices of NB-IoT.

### 2.2.4 SIGFOX

This technology present a solution for LPWAN. The company Sigfox with headquarters in France, is proprietary of the Sigfox's standard, protocol and radio. Also, the standard is not completely open but they give the highlights in their website [12].

The solution presented by Sigfox is similar to the cellular companies. There is an entity called operator which deploys the infrastructure of base stations in a region. The user of the service is billed periodically for each activated device.

Sigfox offers a cloud solution:

- The devices are activated on-line.

- The messages are automatically and only sent from the antennas to Sigfox's servers.
- They assure a capacity of 1 million devices per Base Station.

The available information about the radio and medium access is the following:

- Asynchronous transmission.
- It transmits over the publicly available and unlicensed ISM bands of 868-869 MHz in Europe.
- Does not allow transmission between nodes.
- Up to 140 unconfirmed (without acknowledgement) uplink messages and 4 downlink messages per day.
- The uplink message maximum payload is of 12B, it is 8B for downlink messages.
- The power consumption of the modem is 25mW.

### 2.2.5 LoRa & LoRaWAN

The Long Range Network (LoRa) protocol stack provides another solution for LPWAN. LoRa stands for "Long Range" and is a radio modulation for wide wireless communications. It is a novel transmission protocol, the 1.0 version of the standard is from 2015. Long Range Wide Area Network (LoRaWAN) [1] is a MAC protocol used for sending network commands and carry application data.

The LoRa physical layer presents Chirp Spread Spectrum (CSS) as its default radio modulation technique, but it also accepts FSK symbols.

- Asynchronous data transmission.
- CSS modulation with 8 orthogonal spreading factors.
- It has been standardized transmissions over the ISM bands of 868MHz and 433MHz in Europe.

LoRaWAN has a semi-propietary character. Although LoRaWAN is completely open, it's physical layer is property of Semtech.

- Only BS to device or device BS communication. Being the device-BS the most common.

#### LoRaWAN Classes

There are three LoRaWAN classes for end-devices: A, B or C [1]. All classes can send and receive messages from BSs and those messages can be confirmed or unconfirmed. Those classes vary in its availability to receive messages from the LoRa network.

**Class A** Bi-directional end-devices. The class A node opens two short downlink receive windows after each uplink. Then, only can receive downlinks after an uplink. For nodes with very low-power restrictions and low-rate transmissions.

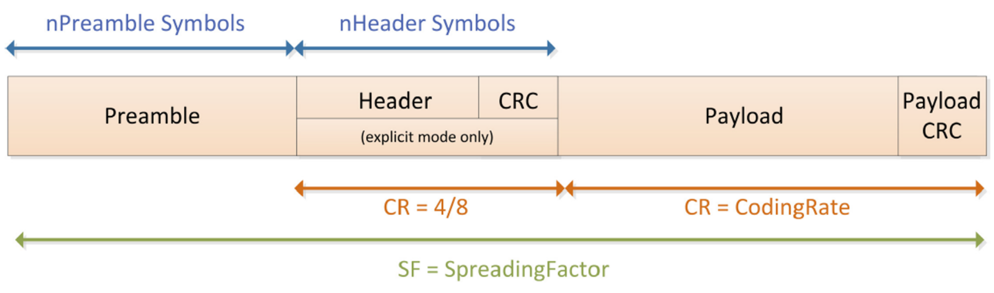


Figure 2.7: LoRa packet format. Preamble, Header, Header CRC, Payload and Payload CRC. Image from [13].

**Class B** Bi-directional end-devices with scheduled receive slots. An extra receive windows are added to the Class A. Those extra receive windows are scheduled in time. It is synchronized with beacon frames from the gateway.

**Class C** Bi-directional end-devices with with maximal receive slots. This class have complete open windows for downlinks, except when device is transmitting.

### LoRaWAN MAC Commands

LoRaWAN presents a similitude with cellular networks, it allows the network to send commands to the end-device regarding to layer 1 and 2 parameters such as transmit power or increase the time between messages. Currently at May of 2017, there are 7 kinds of mandatory commands, each one needs both a request and a answer. We make an especial mention to the *LinkADR* command, it allows the network to ask end-device module for changing the:

- Data rate.
- Transmit power.
- Repetition rate or channel.

It can be very useful to let the network manage the power and the data rate in order to maximize the network use remotely.

## 2.2.6 Overview LoRaWAN Vs Sigfox

In order to sum up all the information of the subsections above, we present in table 2.1 the comparison of similar features between Sigfox and LoRaWAN. Within the metrics we have included the OPerational EXpenditure (OPEX) in this case specifies the monthly costs due to use of the network per device.

Even though, it is true that the Sigfox network is widely deployed in Europe, the main problem is that the communications are not free. After the initial expenditure, it needs from a periodic payment to maintain them. If this is prolonged for all the devices and for years, the price rises enough as to build a proprietary wireless network.

Table 2.1: Comparison between the LoRaWAN stack and the Sigfox stack. Data from [1] and [12].

| | LoRaWAN | SIGFOX |
|---|---|---|
| OPEX | Free data transmission | Subscription to Operator |
| TX Restriction | 99% guardband at 868MHz. Band restriction per 125KHz band | 140 messages per day per device |
| Number of bands | 52 of 125KHz | 45 of 200KHz |
| Payload maximum length | 292 | 12 Bytes |
| Device to device communication | No | No |
| ADR (Adapatative Data Rate) | Yes | No need |
| Encryption | Yes | Yes |
| Time in the air | 12 bytes of payload, CR = 4/5 Min: 62ms. Max: 528.4ms | 12 Bytes: 6240ms |
| Power compsumtion | 0.6 $\mu$Ah for 12 B | 110.1 $\mu$Ah per for 12 B |
| BS Ack | Yes | No |
| Downlink Availability | One for each transmission | 3 for day |
| Mobility | Yes | No if v > 20km/h |

## 2.3 Microcontroller Platform

We present two alternatives to perform the logic part of the project. There are plenty of platforms and boards currently in the market that are able to solve our needs. It is not an objective of this document to mention all the possibilities and compare them. However, we will present two platforms to develop IoT devices.

### 2.3.1 Arduino

Arduino is an open-source electronic platform. The Arduino project offers a range of software tools, hardware and documentation enabling fast prototyping and developing for most of projects based on general purpose microcontrollers with an architecture of 8-bit or 16-bit.

Arduino has multiple boards to complete the typical microcontroller tasks. We will focus on the Arduino Uno Rev. 3.

The third revision of the Arduino Uno is a shield with an Atmel ATMEGA 328P microcontroller connected to pin strips, pin headers, USB-A and DC connector for power supply, figure 2.8.

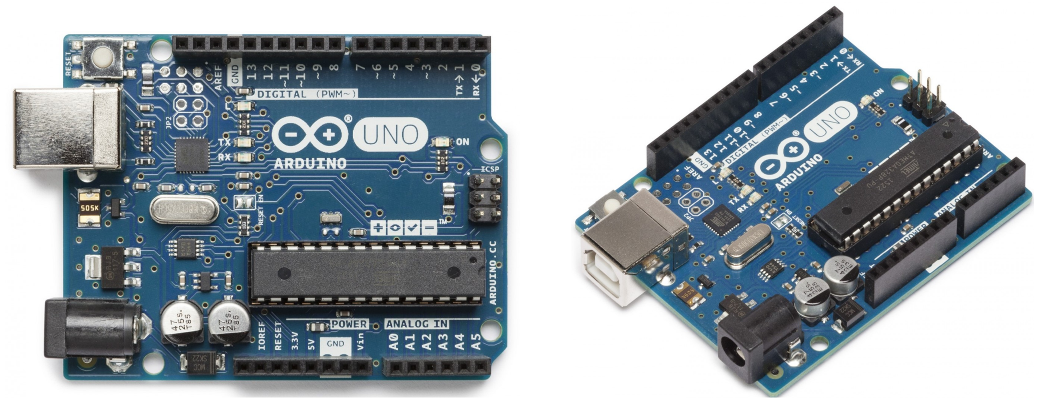


Figure 2.8: Arduino UNO Rev. 3 from Arduino.cc [14].

#### Power Supply

The Arduino Uno board can be powered via the USB connection or with an external power supply. The board can operate on an external supply from 6 to 20 volts. The recommended range is 7 to 12 volts.

#### Pin IO

There are 14 digital pins:

- Serial: 0 (RX) and 1 (TX). Used to receive (RX) and transmit (TX) TTL serial data.
- External interrupt pins: 2 and 3. These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value.
- There is a built-in LED driven by digital pin 13.

There are also 6 PWM pins, 10-13 pins can be used for SPI communication and 6 analog pins labelled A0-A5.

### 2.3.2 Waspmote

Waspmote is an electronic platform to build up sensor networks with a low power consumption. The waspmote board is design to adapt sensors and the transmit modules simply, as shown in figure 2.9.

As well as Arduino, it has open-sourced their API and software but not their board. The sensors vary from pressure and humidity to accelerometers and GPS. It has available all the transmit technologies commented above in this chapter: GSM, UMTS, LoRaWAN and Sigfox.

### 2.3.3 Overview Waspmote Vs Arduino

We sum up all the information of the subsections above in the table 2.2. We introduce here the concept of CAPital EXpenditure (CAPEX) that refers to the amount of capital

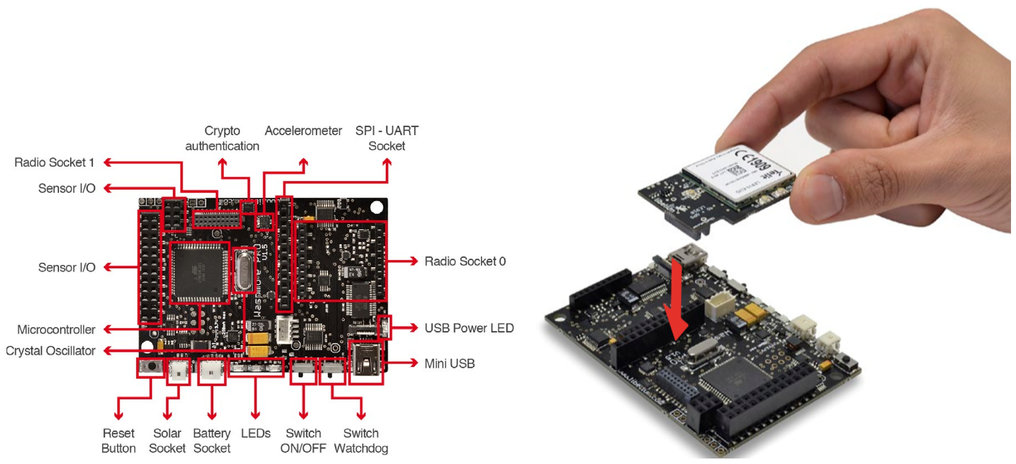


Figure 2.9: Waspmote device with a GPRS transmit module. Image from [15].

Table 2.2: Comparison of the capabilities of Arduino and the Waspmote. Data extracted from [14] and [16].

| Magnitudes | Arduino | Waspmote |
|---|---|---|
| CAPEX | 22.5e | 173e |
| Microcontroller chip | ATmega 328P | ATmega 1281 |
| SRAM memory | 2KB | 8KB |
| FLASH memory | 32KB | 128KB |
| EEPROM memory | 1KB | 4KB |
| Architecture | 8-bit | 8-bit |
| CPU clock frequency | 16MHz | 14MHz |
| Current consumption: active | 25mA | 17mA |
| Current consumption: sleep | 10mA | 30uA |
| Interrupt ports | 2 | No |
| Serial / I2C | Yes | Yes |
| XBee sockets | No | 1 |
| Other sockets | No | Yes, solar, battery radio, acelometer |
| Inner encryption chip | No | Yes |

that has to be invested to produce or buy the product. We obtained the CAPEX from the manufactures at [14] and [16]

It is clear that the performance of the waspmote board is superior in comparison to the Arduino UNO. It is also simpler for developing sensors and transmit modules. But it has a payback, its the price. Nonetheless, the key factor to develop on the Arduino board is that it is the open-sourced hardware. This means that the step from prototyping to to final industrialized product is severally reduced because each component in the shield is documented and it is possible to modify the schematics with free charge. Taking into account that the aim of this project is also simplify future developments, the Arduino is our target developing board.

Nevertheless, it is necessary to add a shield to interface the transmit module with the Arduino. The Multiprotocol Radio Shield from libelium enables the use of the XBEE interface of the transmit module for the Arduino Uno Rev. 3.

# CHAPTER 3

# DESIGN

In this chapter we will cover the main aspects of the design proposed for the overall system and more specifically for the vehicle detection sensors.

Firstly, we will take a look into the requirements that the clients specify for this project. Then, we will overview the global project. Finally, we will introduce the main aspects of the design of proposed for the sensor devices.

As mentioned in the title of the project we will use pressure sensors to detect vehicles and LoRaWAN to transmit the results.

## 3.1 Stakeholders & Requirements

We will firstly describe the main stakeholders and concepts of this project and afterwards, we will introduce the main characteristics that this project need to solve.

### 3.1.1 Stakeholders

On the first hand, the stakeholders that will play a role in this project are the following:

**Fundació Bit** Is a non-profit foundation attached to the *Vicepresidencia y Conselleria de Innovació, Reserca i Turisme*. Promoting research and technology within the Balearic Islands is their main objective.

**Multimedia** Public company in charge of the telecommunication towers and public telecommunication infrastructure.

**Parc Bit** *Parc Balear d'Innovació Tecnologica*. It is located in the Carretera de Valldemossa. It is managed by the Fundació Bit. A map representing the Parc Bit is in figure 3.1.

**IoT Labs** Design company, it will take care of the prosperous outgoing of the project.

**Driver** The user is any person with a vehicle that wants to park in the target parking.

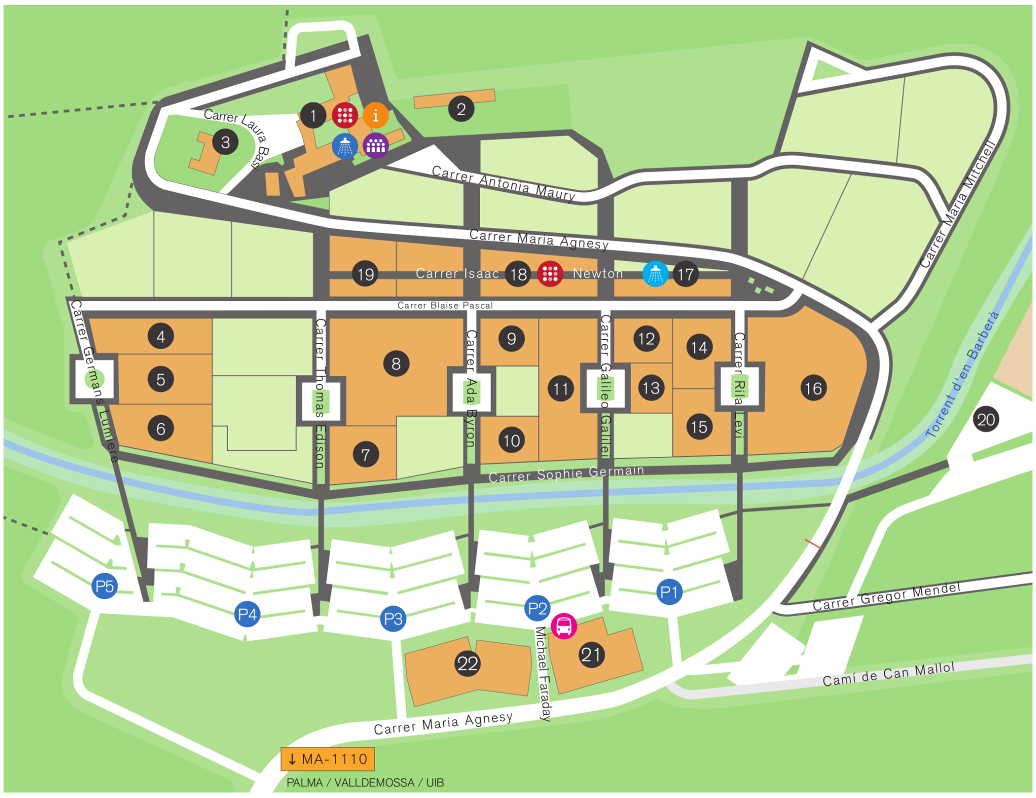


Figure 3.1: Parc Bit Map. From Parc Bit's web [17].

**Web User** The user that connects to the IoTLabs web application to know the state of each parking zone.

**Parking** Is the Parc Bit's parking, it is own by the Fundació Bit.

### 3.1.2 Requirements

We will describe the requirements that a general parking will have.

#### Functional Requirements

We will define the requirements in order to satisfy the necessities of the stakeholders shown before:

- The nodes collect data from the flow of vehicles and transmit it to the gateway.
- The system delivers information about the information about the free spots of each parking zone (defined in 3.3).
- The detection system does not disturb the drivers. It is transparent to them.
- Take into account the user experience. Do not show wrong data to the drivers.

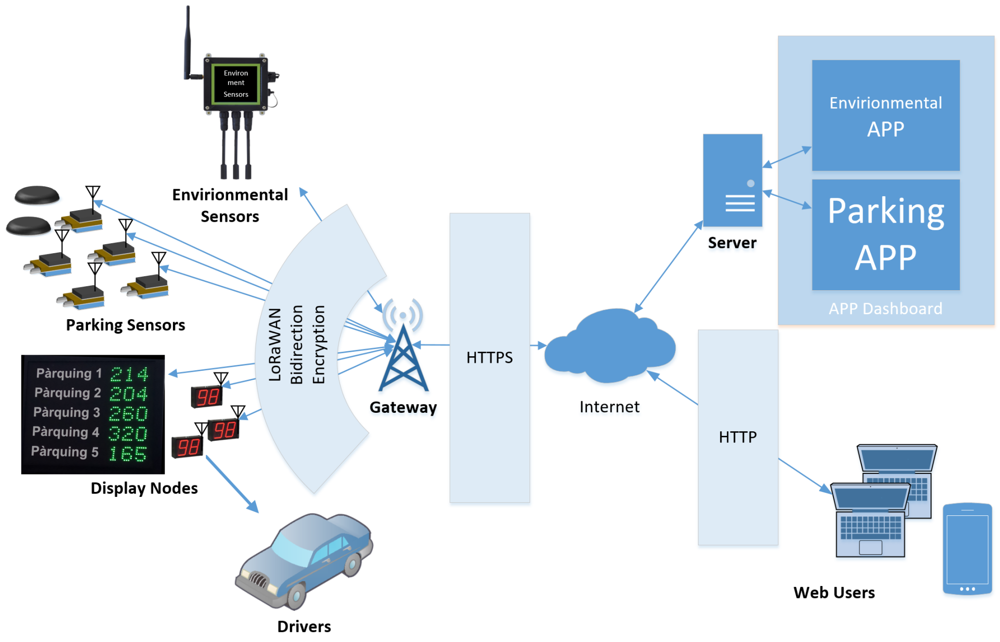


Figure 3.2: Complete proposal diagram. Six actors will play a role in the service.

**Non-Functional Requirements**

- It has to be available the 99.5% of the time. The rest is used for maintenance.
- Autonomous operation after setup.
- During maintenance the system must not lead the drivers to misunderstanding.
- The delay between the captures to the display of the free spots on the screens must be the lowest possible to maintain the validity of the information. This lead to determine the maximum delay to 1 minute.
- The parking will be outdoors, it has to be isolated from the environment.

## 3.2 Project Overview: Solution presented

In figure 3.2 there is a diagram representing all the actors involved to deliver the parking service. The actors are described in the subsection ahead and, the workflow of the system as a whole is described in 3.2.2.

### 3.2.1 Actors involved in the service

In figure 3.2 there are six actors:

**Motes** or nodes. In this case, the motes produce/capture information:

- Parking sensors. There are two types of vehicle detectors: flow counters at the entrances and spot-to-spot counters at each handicapped spot.
- Display nodes. There are **four display of one line** of three digits **and one big display of 5 lines** with three digits each.
- And the environmental station.

**Gateway** Is network infrastructure that bridges between the Internet and LoRaWAN network.

**Server** Computes the logic for the services. Possibly, it is virtualized in the cloud with DB. It receives frames from motes and sends other frames back to the motes.

**Application** Is hosted in the Server. It does the logic referred to handling the data from the nodes and presents them to the web users.

**Drivers** People going to the ParcBit parking by car.

**Web user** The information regarding to the use of the parking will be available on-line via web for the web users.

The environment sensor will transmit the temperature, atmospheric pressure, humidity and the particles per million part of $CO_2$, CO and NO gases.

There is a LoRaWAN antenna previously installed by IoT Labs S.L. in the Multimedia location for antennas in the ParcBit. The location is shown in figure 3.1 upper center, zone number 2.

### 3.2.2 Parking System: General Procedure

As it is shown in figure 3.2, there are 4 types of nodes/motes on the parking. All of them are on the field and will transmit wirelessly with LoRaWAN to the gateway.

- Handicapped spot sensors will transmit whether the spot is being used or not.
- The vehicle detection sensors will report periodically the traffic of motorcycles and cars.
- The vehicle detection sensors located at the entrances of the parking apart from the counting will be connected to a screen to display the number of free-spots.
- A big display node (DM) that will print the value of each parking zone. This value is received via downlink and formatted by the server.

The motes capture information about the flow of vehicles at the entrances and the use of handicapped spots, this information is sent to the server via uplink. The server implements the logic of the service and performs downlinks that will inform the display nodes the numbers to print on the screens.

#### Uplink flow

All the information regarding to the traffic flow detection is encrypted and transmitted wirelessly to the gateway located in the ParcBit.

Once the gateway receives the frames, it checks the validity and only if succeed, the data of the received frames is transmitted via HTTPS to the server over the Internet.

The server is located in a cloud service and hosts the different applications in this case: Environment and Parking.

#### Downlink flow

The downlink frames will be used to inform the nodes which number display on the screens. The server compounds the information of the received frames from the nodes. The server encapsulates the frames in a HTTPS response to the gateway. As it is specified in the LoRaWAN standard, the gateway performs downlinks after the transmission of uplinks. When the nodes receive the downlinks they will decrypt the frame and transmit the value to print to the display. To update the value presented to the drivers, the display nodes will need from downlink messages.

## 3.3 Work embedded in this document

In this work, we will focus on the sensors of the parking that measure the traffic flow and the nodes that communicate to the screens. The environment station and handicapped spot sensors will be acquired, they are specified in other IoT Labs' document.

#### Nomenclature used the parking

Through this document we will use a common vocabulary with the following few exceptions:

**Parking Zone** or subparking is a zone of the parking to be gauged. Each parking zone has one or more entrances and exit. To successfully count the vehicles inside the parking zone, there must be a vehicle detection mote in every entrance and every exit of that zone.

**Entrance** will refer to a unique two way street that is used for going in and out of the parking.

**Connexion** is the path that connects two parking zones within the enclosure.

In figure 3.4, we have the spatial distribution of the set of parking zones. There are 5 different parking zones P1, P2, P3, P4 and P5.

A brief description of the distribution of the parking:

- There are 5 parking zones in the parking, figure 3.3:

  **P1** 224 car places. One entrance.

  **P2** 216 car places. One entrance. 20 motorcycle places.

  **P3** 267 car places. One entrance. 20 motorcycle places.

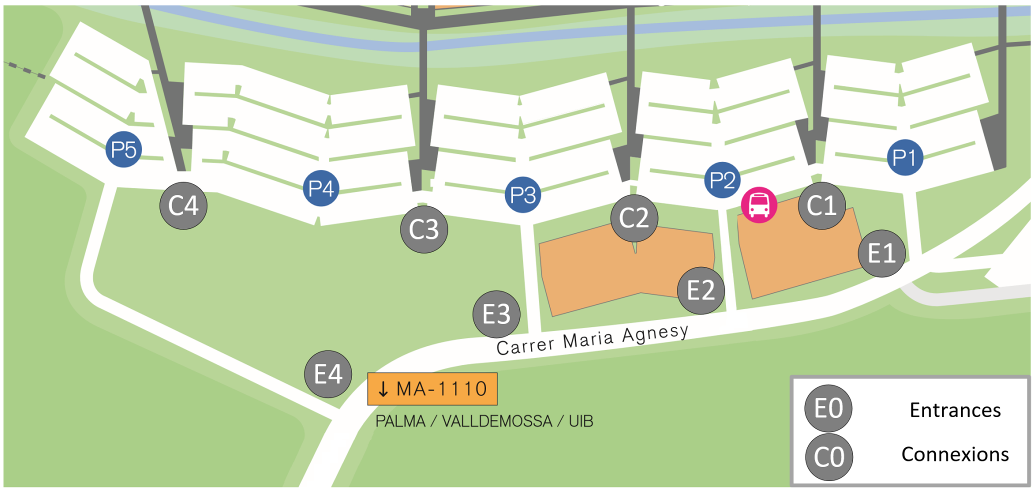


Figure 3.3: ParcBit parking zones are separated from the exterior by E1-E4 and internally by C1-C4.

**P4** 326 car places. No entrance.

**P5** 168 car places. One entrance. 10 motorcycle places.

- There are 4 entrances in the parking, figure 3.3:

**E1** Entrance to the P1.

**E2** Entrance to the P2.

**E3** Entrance to the P3.

**E4** Entrance to the P5.

- There are 4 connexions in the parking, figure 3.3:

**C1** From P1 to P2.

**C2** From P2 to P3.

**C3** From P3 to P4.

**C4** From P4 to P5.

### 3.3.1 Overall parking solution presented

A descriptive map of the solution presented is in figure 3.4, it consists of 8 sensing devices, from S0 to S7, 4 one-line displays, from D1 to D4, and a general display, DM.

The sensors, S0 to S7, are to be located in the 4 entries as well in the 4 connexions. Every sensor S0-7 is connected to two tubes to detect the vehicles, figure 3.5.

**S0** in E1.

**S1** in C1.

**S2** in E2.

**S3** in C2.

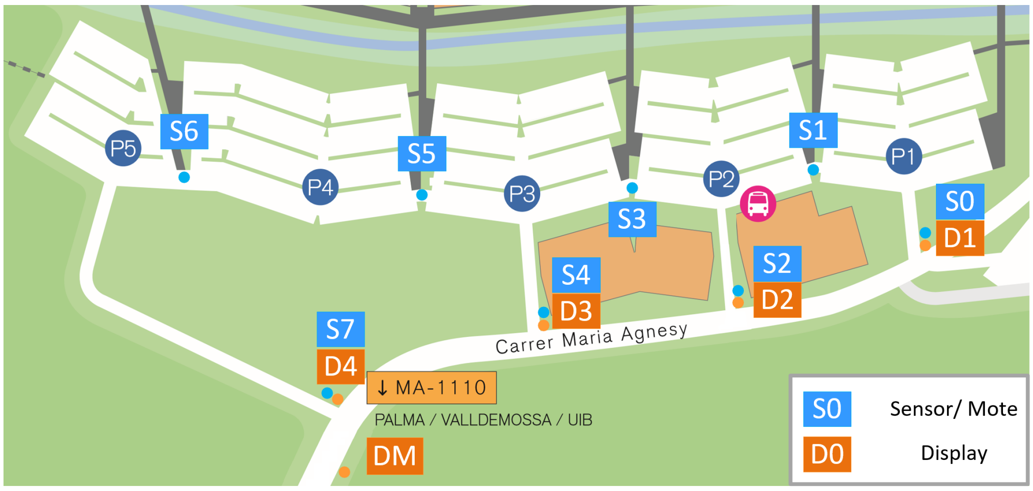


Figure 3.4: Designed location of the parking nodes and displays in the Parc Bit.

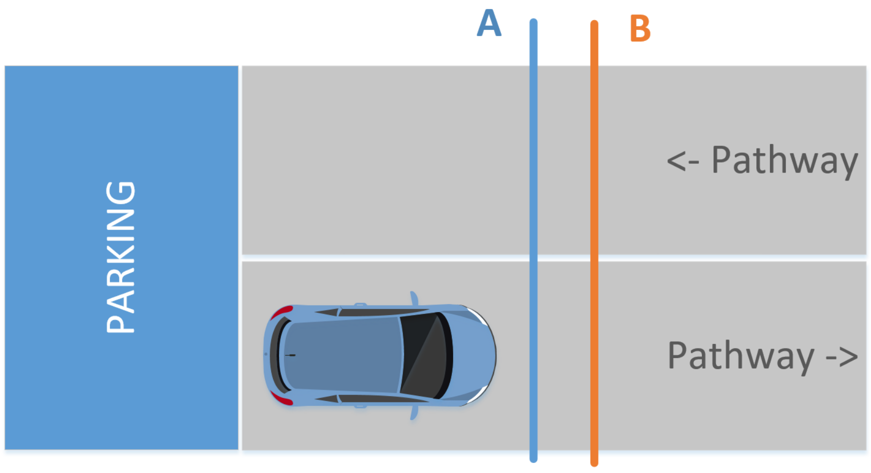


Figure 3.5: The node will classify the vehicles on their way entering or leaving and motorcycle or car. This distinction will be done with pressure tubes, A and B, laid on the road.

**S4** in E3.

**S5** in C3.

**S6** in C4.

**S7** in E4.

**DM** before E4 in MA-1110

**Device Function Description**

The sensors S0-7 will consist of a microcontroller with a pressure shield and transmitting pressure tubes that go on the path. Each sensor will count with the pressure shield 4 vehicle counters:

| | |
|---|---|
| inCar | Number of cars that entered in the parking. |
| outCar | Number of cars that leaved the parking. |
| inMoto | Number of motorcycles that entered in. |
| outMoto | Number of motorcycles that leaved in the parking. |

The sensors located in entrances: S0, S2, S4 and S7 will also have a connection to the 4 three-digit displays. Then, those devices will also be in charge of displaying the free parking spots.

### Power Supply

Two possibilities are propoussed to supply power to the devices:

- With solar stand-alone stations near the sensing devices or DM. It would consist of 9 solar all-in-one (panel, inverter and batteries) stations for:
  - 1 for S0 and D1.
  - 1 for S1.
  - 1 for S2 and D2.
  - 1 for S3.
  - 1 for S4 and D3.
  - 1 for S5.
  - 1 for S6.
  - 1 for S7 and D4.
  - 1 for DM.
- With 230V AC electrical current. To do so, it has to be made a dig and carry the current cables to 9 places:
  - 1 set of 25W for S0 and D1.
  - 1 set of 25W for S1.
  - 1 set of 25W for S2 and D2.
  - 1 set of 25W for S3.
  - 1 set of 25W for S4 and D3.
  - 1 set of 25W for S5.
  - 1 set of 25W for S6.
  - 1 set of 25W for S7 and D4.
  - 1 set of 100W for DM.

### Update values: Send frames to LoRa gateway

The sensors will send information about the 4 vehicle counters. The motes will trigger the transmission when any of the two conditions mentioned above are satisfied:

**Time Threshold** Once it passed this time from latest transmitted frame, the device will send other uplink frame

**Flow Threshold** An integer number that represents the total number of vehicles that has passed by the node from the latest transmission, another transmission will be performed.

### 3.3.2 Hardware

We have seen that for the IoT solution we have presented we need a very specific motes. The nodes we have the following parts:

**Pressure Shield** Differentiates between motorcycles and cars with the the output of a pressure sensor. Once it detects an axis it will trigger an interrupt to the micro-controller. The microcontroller interfaces the shield to obtain the information. We will design the shield that will detect the vehicles passing by.

**LoRaWAN Module** Transmits/receives data within a LoRaWAN scheme. We will buy the LoRa transmitters, because we lack of the technology to produce them.

**Display** Displays the information to the drivers.

**microcontroller** Interfaces the 3 devices/shields mentioned above, and controls the logic (described in 3.3.3).

A power supply is required to source the microcontroller, the pressure shield and LoRaWAN Module. The communication with the display will require a driver as is likely to be one or two meters away. The microcontroller will also supply that shield and multiplex the information to the LoRaWAN and to the display. The display have their own power supply system.

#### Pressure Shield

In figure3.6 there is a general diagram of the pressure shield. This diagram shows all the functions that the pressure shield has to accomplish:

- Firstly, from the physical world it has to measure the relative pressure between the tube and the atmosphere.
- Then, it has to be translated to electrical signal to be later amplified to a standard voltage range.
- After, it processes the signal to extract the information about what happened in the tubes. The possible cases are tagged as follows::

  Nothing, no interrupt.

  **#a** An interrupt caused on the tube A by a motorcycle.

  **#b** An interrupt caused on the tube B by a motorcycle.

  **#A** An interrupt caused on the tube A by a car.

  **#B** An interrupt caused on the tube B by a car.
- Once in the digital world, the information is stored in registers to be accessed later. The output of the registers are evaluated to perform a global logical OR and trigger the interrupt service in the microcontroller.
- Finally, the interface is used by the microcontroller to access the data and restore the pressure shield to the idle state.

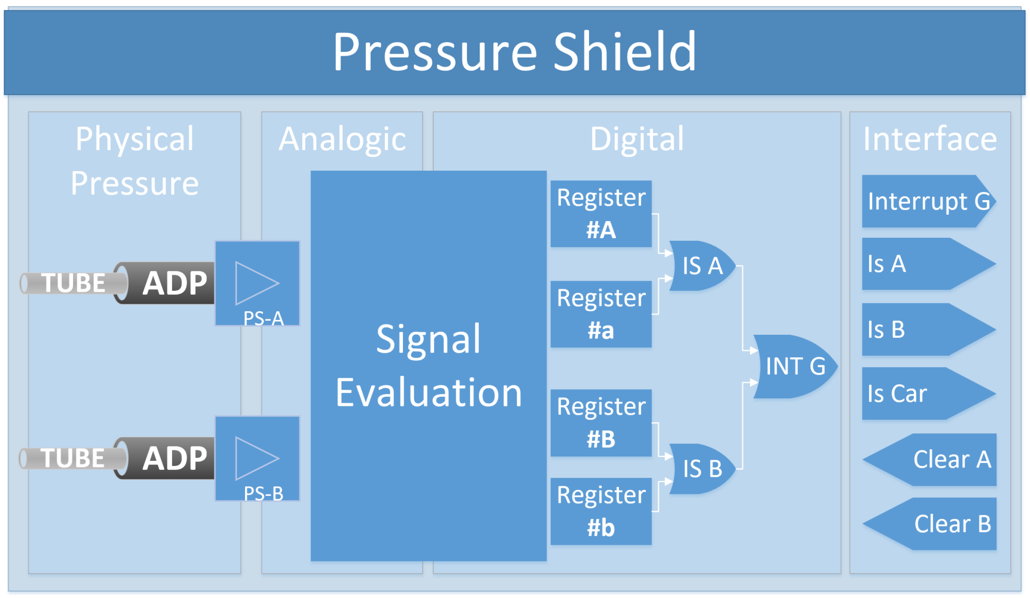


Figure 3.6: Pressure Shield function diagram. It is composed of four sections: physical, analogic, digital and the interface. With a two pressure sensors PS-A and PS-B the applied pressure is measured at the tubes. This information is evaluated and four possible states are decoded in the digital section (E.g.: **#A** a car in the tube A). The interface helps the microcontroller to access the captured information.

We hypothesize that the response is influenced by the the weight in the axis, thickness of the tube inner and outer diameter, the width of the wheels and the velocity of the vehicle. In the experiment conducted we will see how the pressure tubes response.

### 3.3.3 Software

We will need to program a microcontroller to be able to perform as designed. We include a diagram of the indispensable modules of the software in 3.7.

The different types of nodes we describe in this project 3.3.1 will need of the modules described in 3.7.

**DM** Major Display (5-line screen):
**Eeprom Handler** + **Error Handler** + **Transmit Module Controller** + **Display Controller**.

**S1,S3,S5 and S6** Traffic detector:
**Eeprom Handler** + **Error Handler** + **Transmit Module Controller** + **Interrupt Handler** + **Pressure Shield Driver**+ **Detection Algorithm**

**S0,S2,S4 and S7** Traffic detector and Display (1-line screen):
**Eeprom Handler** + **Error Handler** + **Transmit Module Controller** + **Interrupt Handler** + **Pressure Shield Driver**+ **Detection Algorithm** + **Display Controller**.

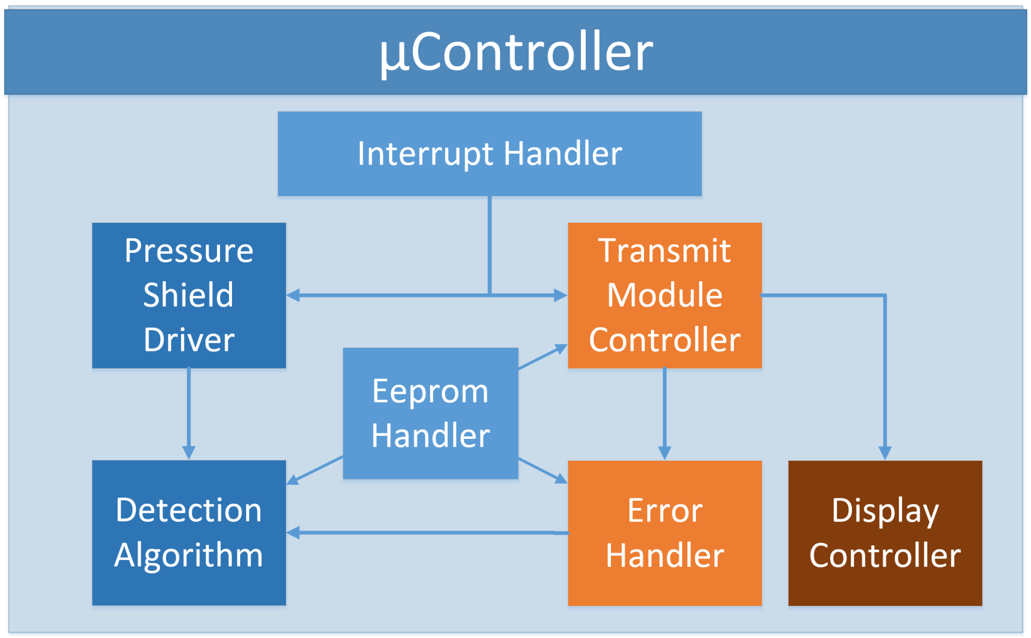


Figure 3.7: Diagram of the algorithm to implement in the microcontroller. It consist of 4 general modules: in blue INTERNAL (Interrupt Handler, Eeprom Handler), DETECTION in dark blue (Shield Driver and Detection Algorithm), TRANSMIT in orange (Transmit Module Controller and Error Handler) and DISPLAY in brown (Display Controller). A designed algorithm for detecting vehicle's way is necessary once read from Pressure Shield. A non-volatile memory Eeprom Handler is added to save device's state. Transmit Error module logs the transmit module responses.

**Pressure Shield Driver**

The pressure shield driver module will be in charge of reading the events captured in the shield and serve the data to the Detection Algorithm.

This module will be executed in the Interrupt Service Routine (ISR). Then, when the wheels of the vehicles compress the tube, the shield will set different registers on the pressure shield depending on the captured response. If one of those registers are set, the interrupt pin of the microcontroller will be triggered. The shield driver has to read up to 4 types of events:

**#a** An interrupt caused on the channel A by a motorcycle.

**#b** An interrupt caused on the channel B by a motorcycle.

**#A** An interrupt caused on the channel A by a car.

**#B** An interrupt caused on the channel B by a car.

Once the information is obtained, the pressure shield driver will be in charge of returning the shield to the reading state.

#### Detection Algorithm

The Detection Algorithm module will be awaken when the pressure shield driver is triggered by an interrupt. It will implement the logic to extract information about the events based on the four types of register that can set mentioned in the paragraph above.

#### Eeprom Handler

The microcontrollers in general have Electrically Erasable Programmable Read-Only Memory, the so called EEPROM. It is extremely useful to save information relevant in time that should be kept in case of power supply failure.

It has a counterpart, it has finite write/reset cycles. Depending on the manufacturer the write/reset cycles oscillate between 100.000 and 1.000.000 writes.

We will use the EEPROM to save the parameters relevant to transmit triggers, transmit failures on a row and the number of times that the device has awaken.

#### Transmit Module Controller

The transmit module will be in charge of the communications with the Base Station or gateway (BS).

On the first hand, the **uplink** transmission will be used for indicating the current device state and for describing the quantity of cars and motorcycles that had passed in and out.

On the other hand, the **downlink** transmissions will be used for updating the device state and the value displayed on the screen, if the device is connected to a display.

#### Display Controller

The Display Controller performs the tasks related to transmit the value to display to the screen: compose the frame and configure the device to transmit.

#### Error Handler

The LoRaWAN transmissions use ALOHA as medium access technique with restrictions due to the public ISM band use. Moreover, it is possible and legal for a person to interrupt the communications on this band by introducing noise. Then, we can't assure that the channel will be idle during transmissions. Therefore, the Error Handler module will define polices that will be followed in case of a erroneous transmission.

The polices have to take into account the avoidance of congestion of the network.

### 3.3.4 Reminder: Functionalities not included in this document

As we have said in the overview of this chapter, there must be a gateway and a server to perform the general count for each parking.

The configuration and programming of the gateway and the applications in the servers will not be described in this document as it is other IoT Labs' project. The nodes are completely independent of the underlying LoRa infrastructure of gateways and servers. Which gateway response to the frame is not decision of the transmit nodes.

CHAPTER 4

# IMPLEMENTATION

In this chapter we present firstly the experiment conducted to understand the behaviour of the pressure sensor in 4.1. Then, we will present the design of a board and the components needed to detect the events mentioned in 4.2. Section 4.3 shows how the algorithms are implemented on the Arduino environment. Next, we will see the trade off of our detection mechanism and how differentiate motorcycles from cars 4.4. After, we will introduce the connection between our microcontroller and the display 4.5. We will finish this chapter with a description of the transmit module and how to use it for our purpose 4.6.

## 4.1 Pressure Experiment

**Objective**

The main goal of this study is to know the characteristics of the pressure response produced by vehicles passing over tubes. In order to differentiate between the responses produced by different vehicles, the tubes are stimulated to cars and motorcycles. The pressure inside the tube is measured with a pressure sensor at the end of the tube.

### 4.1.1 Introduction

The elastic tube constrains when the wheels of the vehicle are on it. When the wheels compress the tube, the pressure rises as the air inside is in outgoing movement and the volume is reduced. With a Pressure Sensor (PS) at the end of the tube, it is possible to measure the increment of the pressure. When the wheel is no longer on the tube, the pressure decreases as the volume expand to previous state.

No air is lost in this process as the tube is shut with a tight knot at one end and the PS is in the other end.

The velocity in the change of pressure is expected to be related to:

- Mass of the vehicle over the tube.

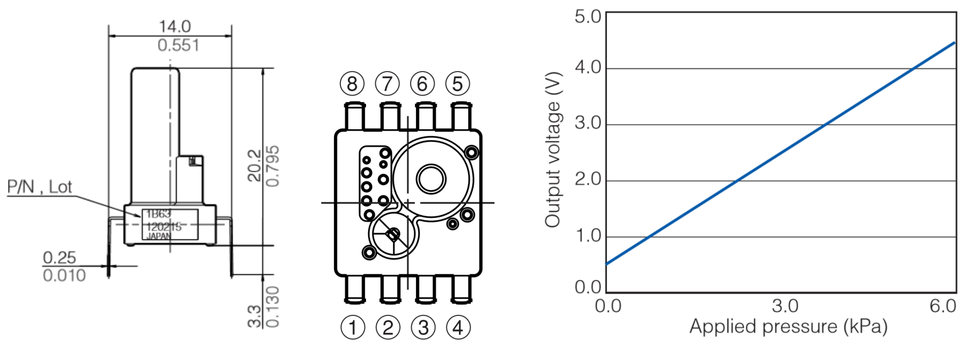


Figure 4.1: From left to right. An diagram of the dimensions of the sensor from lateral. A numeration of the pins, 2 (VCC), 3 (GND) and 4 (Vout), rest not usable. The transference function of the PS voltage in front of applied pressure.

- Width of the wheels.
- The velocity of the vehicle passing over the tube.

It is easy to observe that each tube will be pressed by the two axis of the vehicle (we will describe what happens when has more than 2 axis in 4.7.3).

### 4.1.2 Pressure Sensor Description

It will be used the Panasonic ADP51B63 as pressure sensor. It is a low pressure sensor mainly for air. In figure 4.1 the PS is represented. The sensor measures the relative pressure between the atmosphere and the tube known as the applied pressure which is measured in KPa on the x-axis.

The range of rated pressure is 0-6KPa with 5V of supply. It has a built-in amplifier. On idle has an output of 0.5V and the top voltage is 4.5V. The PS is chosen to be DIP terminal sized to be easy to handle and able to connect to the tubes and get fixed.

### 4.1.3 Components and Materials

We describe in this paragraph the materials and components used for the experiment:

**Oscilloscope** Tektronix TDS 210.

**Scopes** To connect to the correspondent pins of the PS.

**PS** 6KPa Panasonic ADP51B63.

**Silicone Tubes** 20 meters of 3x5 mm white transparent tube. It has to be flexible and resilience enough for supporting the load of a car and then restore to the previous state.

**Tube Adapters** 20 cm of 5x7 mm.

**Power Supply** 5v DC output.

**Vehicles** We used 3 different vehicles described in 4.1.3.

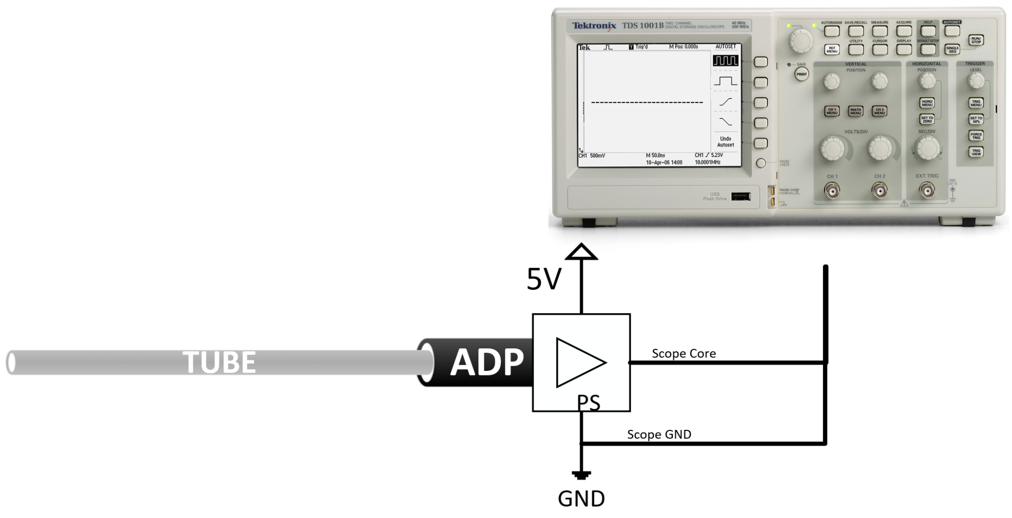


Figure 4.2: Capturing the response of PS.

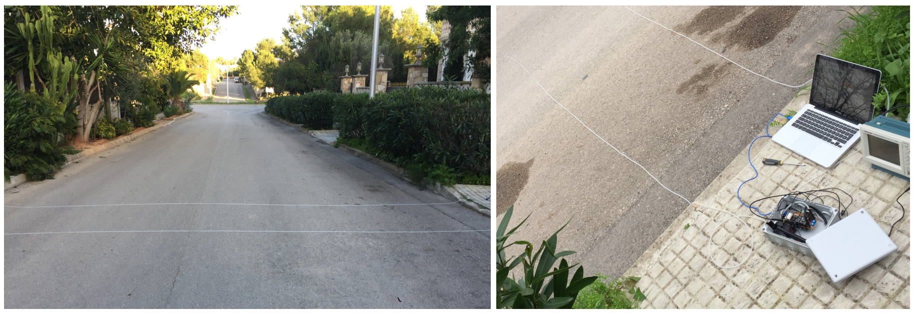


Figure 4.3: Setup for the experiment in a street of Santa Ponsa. For the voltage supply we used the power of PC USB as shown in figure4.3. The ground of the scope was connected to the ground of the USB.

### Measuring Diagram

We built up a circuit, figure 4.2, with an oscilloscope to know the response of vehicles passing by. The tube (TUBE) will be connected to the adapter (ADP). The adapter is also connected to the PS. The scope is put in channel 1 and the core of the scope is connected to the 4th pin of the PS with the reference of the scope to ground. We can observe in images in 4.3 how the tubes are transversal to the path so when the vehicles passes all the wheels in the same axis pass at the same time.

### Oscilloscope Configuration

The default $V_o$ of the PS is 0.47V when powered with 5 volts. We configured the oscilloscope in trigger mode:

Table 4.1: List of vehicles sampled. Data from manufacturers technical sheet.

| **Vehicle Description** | | | | |
|---|---|---|---|---|
| Magnitude | $d_1$ [cm] | $d_2$ [cm] | $d_3/d_4$ [cm] | $m$ [Kg] |
| Saab 9-3 | 267 | 178 | 20/21.5 | 1660 |
| Opel Zafira | 270 | 180 | 19.5/20.5 | 1405 |
| Caviga Raptor | 137 | - | 11/15 | 125 |

- Normal trigger.
- DC coupling.
- On edge.
- Rising edge.
- Voltage level to trigger: 0.6V.
- Voltage scale: 1.0V.
- Time scale raging between: 10ms and 400ms.
- Time arrow to the left of the screen.
- Voltage cursor to the bottom of the screen. There are only positive values.

**Vehicles used in the experiment**

We used 3 different vehicles at different velocities to study the response they made on the PS.

As we are studing the response on PS, we foculs only on some characteristics:

1. $d_1$ Distance between the center of the wheels of the axis.
2. $d_2$ Vehicle width external side to external side.
3. $d_3$ Front wheel width.
4. $d_3$ Rear wheel width.
5. $m$ Vehicle mass, no fuel, no driver.

The three different cars used for the measures are the following:

**Vehicle Number 1** Car: Saab 9-3 cabrio.

**Vehicle Number 2** Car: Opel Zafira.

**Vehicle Number 3** Motorcycle: Cagiva Raptor 125

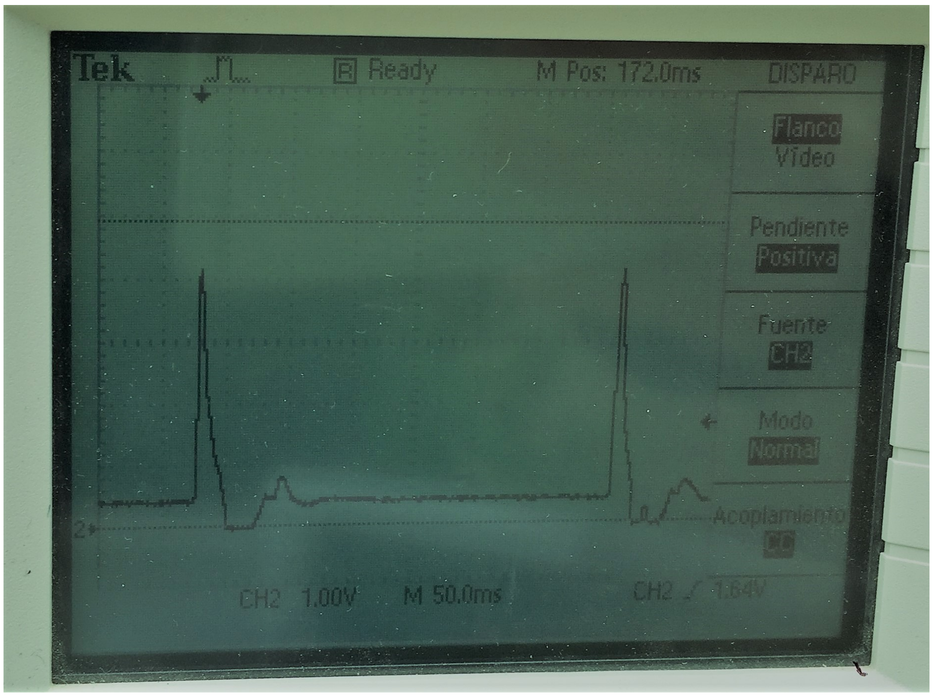


Figure 4.4: Voltage time response caused by a car (vehicle 1) while passing at 29 Km/h

### 4.1.4 Pressure Curves in Time

With the described diagram of the paragraph above, we made a series of experiments to get the data. With the oscilloscope well configured, we we took captures of the oscilloscope screen for the different vehicles of table 4.1. We can observe in figures 4.4 and 4.5 two different responses produced by cars and motorcycles that the PS takes while passing.

At a certain velocity $v$, a vehicle with length between axis $d_1$ we know that the time to pass $t$ is $t = d_1/v$. Every axis produces a disturbance in the pressure field in the tube that lengths $t_{\text{disturbance}}$ seconds.

The signals of the PS when the tubes are pressed by vehicles rises and then decreases to the idle value. E.g. in figure 4.4 it was captured the response of the PS with a car passing at 29 Km/h over one tube. There are two main disturbances in the image that are very similar. The second cursor of the oscilloscope is in voltage mode and is centred in zero volts. We can observe that for about 75ms the graphs stays in idle at 0.5V. The first axis of the vehicle causes the first disturbance, the voltage peaks sharply to 4 volts in approximately 4ms, when arrives to the maximum it decreases rapidly to 0V where it remains for 22ms. Then, it returns to 0.5V gradually where the disturbance repeats the shape minimally. The whole event lengths 100ms.

The two events are separated by a stable idle state that lengths 225ms. The second perturbation is caused by the second axis of the same vehicle. The rear axis causes a similar response in amplitude and time to the first one.

There are some differences between the figures 4.4 and 4.5. The main difference is the amplitude of the responses, the motorcycle response is much more attenuated

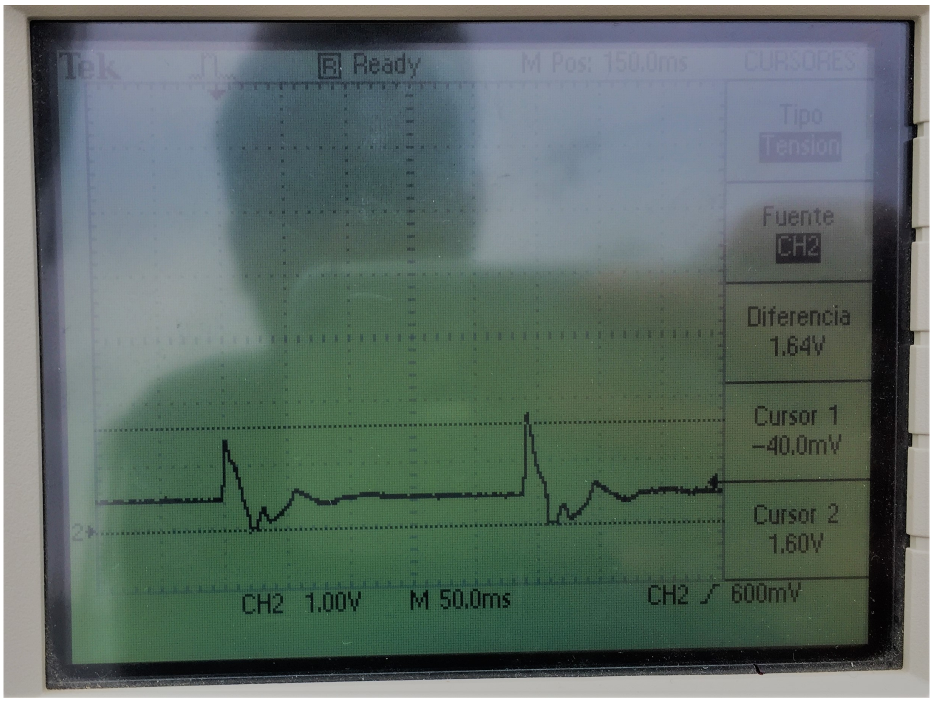


Figure 4.5: Voltage time response caused by a moto (vehicle 3) while passing at 20 Km/h

than the car's response. The duration of the perturbation is very similar in both cases.

There are more examples of the responses in figure 4.6.

### 4.1.5 Pressure Curves $J$ Operator

In order to study the responses caused for the vehicles we use an operation to study the curves $s(t)$ to be continuous. We define $\Delta t = t_2 - t_1$ for a certain, finite amplitude $A$ where $t_1$ is the outer left value with $s(t_1) = A$. And $t_2$ is the outer right value with $s(t_2) = A$, figure 4.7.

$$t_1 = \min_t \{s(t) = A\}$$

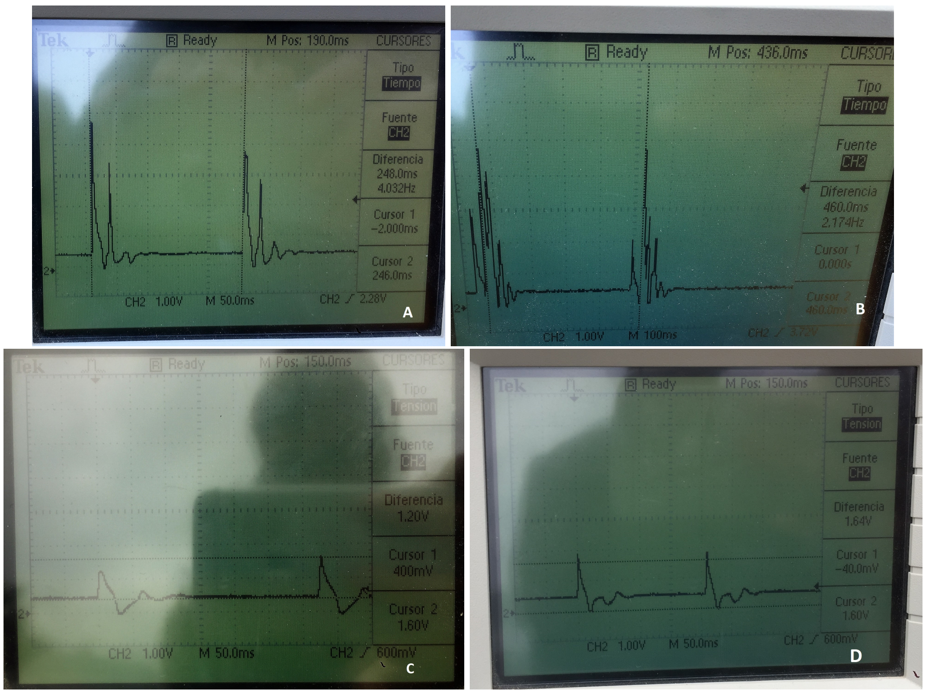
$$t_2 = \max_t \{s(t) = A\}$$

$$J\{s(t)\}(A) = \Delta t(A)$$

$J(A)$ indicates the max quantity of time that the function can value an amplitude A, in this document measured in volts. In figure 4.7 it is represented an example of the operation. The function is time dependent $s(t)$ that is a parable between t = [0,2] and constant otherwise.

$$s(t) = \begin{cases} -x(x-2)+1 & t \in [0,2] \\ 1 & \text{Otherwise} \end{cases} \tag{4.1}$$

$$J\{s(t)\} = t_i(A) = \begin{cases} 2\sqrt{2-A} & A \in (1,2] \\ +\infty & A = 1 \\ 0 & \text{Otherwise} \end{cases} \tag{4.2}$$



Tek R Ready M Pos: 190.0ms CURSORES
Tipo Tiempo
Fuente CH2
Diferencia 248.0ms 4.032Hz
Cursor 1 −2.000ms
Cursor 2 246.0ms
CH2 1.00V M 50.0ms CH2 ⁄ 2.28V

R Ready M Pos: 436.0ms CURSOR
Tipo Tiempo
Fuente CH2
Diferencia 460.0ms 2.174Hz
Cursor 1 0.000s
Cursor 2 460.0ms
CH2 1.00V M 100ms

Tek R Ready M Pos: 150.0ms CURSORES
Tipo Tension
Fuente CH2
Diferencia 1.20V
Cursor 1 400mV
Cursor 2 1.60V
CH2 1.00V M 50.0ms CH2 ⁄ 600mV

Tek R Ready M Pos: 150.0ms CURSORES
Tipo Tension
Fuente CH2
Diferencia 1.64V
Cursor 1 −40.0mV
Cursor 2 1.60V
CH2 1.00V M 50.0ms CH2 ⁄ 600mV

Figure 4.6: Voltage time response caused by vehicle 2 (A, B) and vehicle 3 (C, D)

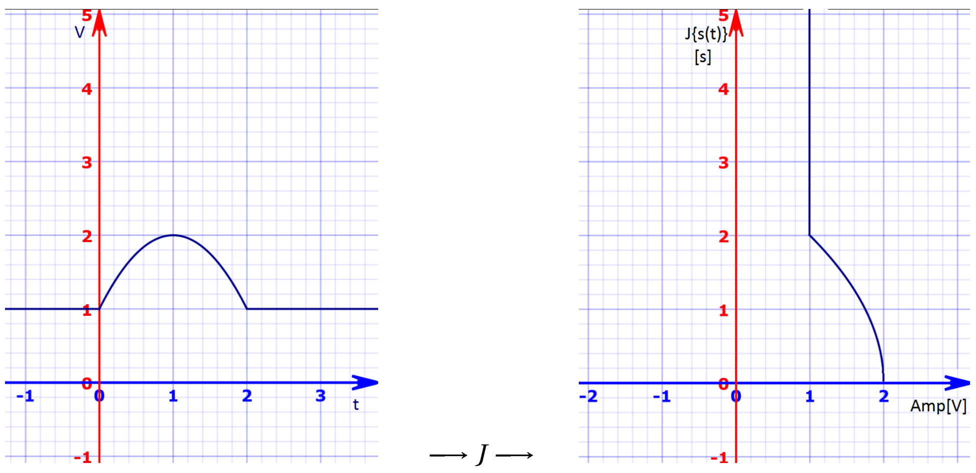
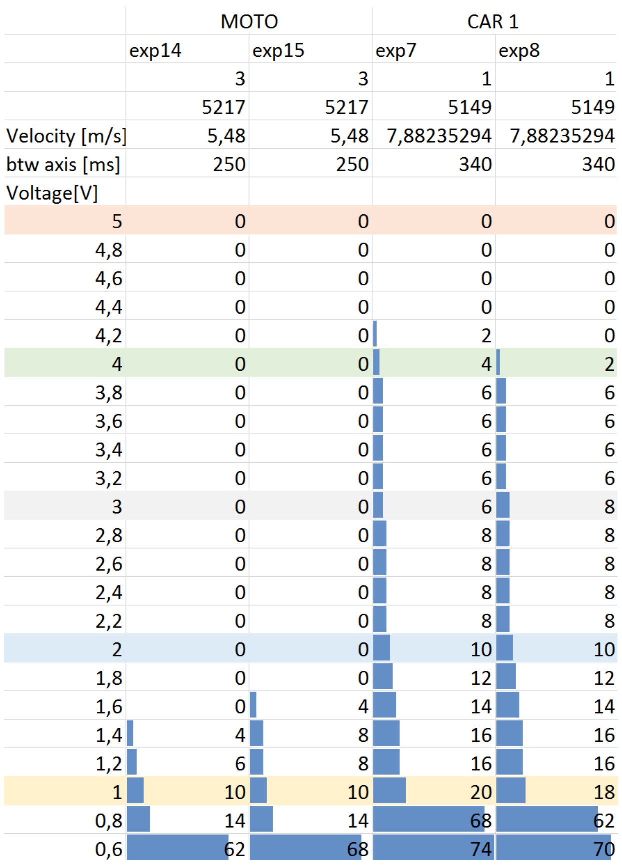


Figure 4.7: Example of the operation to do to the results of the experiment. Function to operate (left) and the result (right). Both are defined in 4.1 and 4.2.

Table 4.2: Values of images 4.5 and 4.4 respectively after the $J$ operation process.



| | MOTO | | CAR 1 | |
|---|---|---|---|---|
| | exp14 | exp15 | exp7 | exp8 |
| | 3 | 3 | 1 | 1 |
| | 5217 | 5217 | 5149 | 5149 |
| Velocity [m/s] | 5,48 | 5,48 | 7,88235294 | 7,88235294 |
| btw axis [ms] | 250 | 250 | 340 | 340 |
| Voltage[V] | | | | |
| 5 | 0 | 0 | 0 | 0 |
| 4,8 | 0 | 0 | 0 | 0 |
| 4,6 | 0 | 0 | 0 | 0 |
| 4,4 | 0 | 0 | 0 | 0 |
| 4,2 | 0 | 0 | 2 | 0 |
| 4 | 0 | 0 | 4 | 2 |
| 3,8 | 0 | 0 | 6 | 6 |
| 3,6 | 0 | 0 | 6 | 6 |
| 3,4 | 0 | 0 | 6 | 6 |
| 3,2 | 0 | 0 | 6 | 6 |
| 3 | 0 | 0 | 6 | 8 |
| 2,8 | 0 | 0 | 8 | 8 |
| 2,6 | 0 | 0 | 8 | 8 |
| 2,4 | 0 | 0 | 8 | 8 |
| 2,2 | 0 | 0 | 8 | 8 |
| 2 | 0 | 0 | 10 | 10 |
| 1,8 | 0 | 0 | 12 | 12 |
| 1,6 | 0 | 4 | 14 | 14 |
| 1,4 | 4 | 8 | 16 | 16 |
| 1,2 | 6 | 8 | 16 | 16 |
| 1 | 10 | 10 | 20 | 18 |
| 0,8 | 14 | 14 | 68 | 62 |
| 0,6 | 62 | 68 | 74 | 70 |

We have used this operator of time functions to gain a better understanding of the responses. We made a calculus data sheet to store the remaining after the $J$ operation.

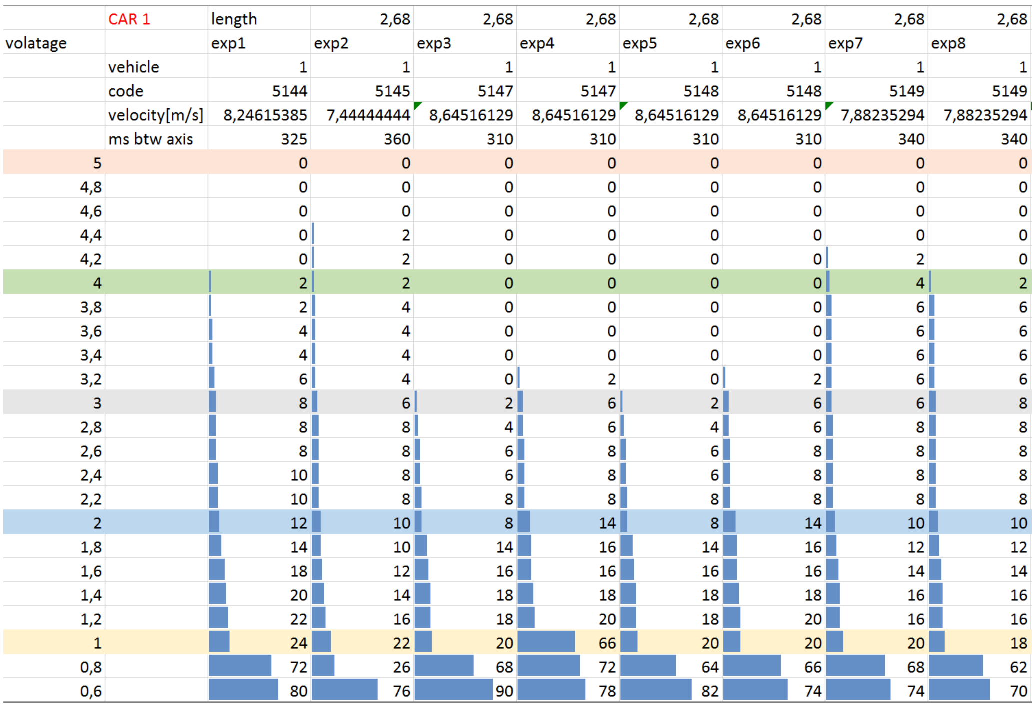
In table 4.3 we can see some of the responses captured . The voltage is ranged from 5 Volts to 0,6 V in top left column. We save the information regarding to each experiment the vehicle, in this case is the vehicle number one and the code. There can be up to two experiments with the same number, that refers to an image, which stands for two signals in the same image. We also wanted to record the velocity (expressed in meters per second). We used the time between the beginning of the both disturbances and the distance between axis to obtain the velocity. In this table 4.3, we can see that the experiments are at the same velocity raging from 7,88 m/s to 8,65 m/s.

### 4.1.6 Results

Finally we present the result of the experiment. We present the data collected with the $J$ operation mentioned in the subsection above. All the data is in a calculus excel file `exp/measurements.xlsx`, that is attached with this document. As we said, the result of the operation shows the amount of time that the signal can value an amplitude.

In the table 4.4 we can see the minimum, maximum and average values obtained ranged for voltage and type of vehicle. For the first vehicle CAR 1, with $d_1 = 2,70$, we

Table 4.3: Responses of the vehicle 1 after the $J$ operation processing. The blue horizontal bars represent the width in time.



|  | CAR 1 | length | 2,68 | 2,68 | 2,68 | 2,68 | 2,68 | 2,68 | 2,68 |
|---|---|---|---|---|---|---|---|---|---|
| volatage |  | exp1 | exp2 | exp3 | exp4 | exp5 | exp6 | exp7 | exp8 |
|  | vehicle | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
|  | code | 5144 | 5145 | 5147 | 5147 | 5148 | 5148 | 5149 | 5149 |
|  | velocity[m/s] | 8,24615385 | 7,44444444 | 8,64516129 | 8,64516129 | 8,64516129 | 8,64516129 | 7,88235294 | 7,88235294 |
|  | ms btw axis | 325 | 360 | 310 | 310 | 310 | 310 | 340 | 340 |
| 5 |  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4,8 |  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4,6 |  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4,4 |  | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4,2 |  | 0 | 2 | 0 | 0 | 0 | 0 | 2 | 0 |
| 4 |  | 2 | 2 | 0 | 0 | 0 | 0 | 4 | 2 |
| 3,8 |  | 2 | 4 | 0 | 0 | 0 | 0 | 6 | 6 |
| 3,6 |  | 4 | 4 | 0 | 0 | 0 | 0 | 6 | 6 |
| 3,4 |  | 4 | 4 | 0 | 0 | 0 | 0 | 6 | 6 |
| 3,2 |  | 6 | 4 | 0 | 2 | 0 | 2 | 6 | 6 |
| 3 |  | 8 | 6 | 2 | 6 | 2 | 6 | 6 | 8 |
| 2,8 |  | 8 | 8 | 4 | 6 | 4 | 6 | 8 | 8 |
| 2,6 |  | 8 | 8 | 6 | 8 | 6 | 8 | 8 | 8 |
| 2,4 |  | 10 | 8 | 6 | 8 | 6 | 8 | 8 | 8 |
| 2,2 |  | 10 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| 2 |  | 12 | 10 | 8 | 14 | 8 | 14 | 10 | 10 |
| 1,8 |  | 14 | 10 | 14 | 16 | 14 | 16 | 12 | 12 |
| 1,6 |  | 18 | 12 | 16 | 16 | 16 | 16 | 14 | 14 |
| 1,4 |  | 20 | 14 | 18 | 18 | 18 | 18 | 16 | 16 |
| 1,2 |  | 22 | 16 | 18 | 20 | 18 | 20 | 16 | 16 |
| 1 |  | 24 | 22 | 20 | 66 | 20 | 20 | 20 | 18 |
| 0,8 |  | 72 | 26 | 68 | 72 | 64 | 66 | 68 | 62 |
| 0,6 |  | 80 | 76 | 90 | 78 | 82 | 74 | 74 | 70 |

can observe that the ranged velocities: 5,58 m/s to 13,4 m/s. Maximum value observed is 4,6 V and the minimum amplitude signal produced is 2,8 V. The average time for the series measured for 2,8 V is 7,4 ms. In this table there is also a graphical aid to read it. The blue horizontal bars indicate the value of the cell. The top voltage recorded is 4.8 V and it is from CAR 2. The minimum amplitude recorded 1.2 V it is from the motorcycle. The minimum and maximum values of the time between axis on top of the table does not ralate with the values of column, it is just the minimum/maximum/average value sampled of the time between axis.

In figure 4.8 it is shown the columns average of the vehicles sampled of the table 4.4. The graph in this figure shows three functions. The x-axis represents amplitude in volts and the y-axis is the amount of time that the signal can value that amplitude in milliseconds. We can see that in average, none of the signals becomes equal or grater than 5. The first vehicle can maintain in time longer for voltages less than 1 V. The second vehicle function shows that in average takes more time at high voltages than the rest. The motorcycle's response in average does not take values greater than 2V.

A more precise differentiation is visible in figure 4.9. In this figure, we present three functions of the amplitude: the maximum response collected from the motorcycle and the minimum responses collected from both cars. We can see that the maximum response obtained in the experiment does not longer more than 25ms. Moreover, we found that the responses collected of the cars are always greater or equal 2.8V. Alternatively, the responses of the motorcycle never are greater than 2.2V.

Out of the data obtained, we can not generalize about every vehicle, but it does in-

Table 4.4: Maximum, minimum and average response for three different vehicles. Results partly represented in 4.9 and 4.8.

| | CAR 1 | | | CAR 2 | | | MOTO | | |
|---|---|---|---|---|---|---|---|---|---|
| | min | max | average | min | max | average | min | max | average |
| btw axis [ms] | 200 | 480 | 340,833 | 178 | 300 | 240,167 | 100 | 375 | 254,938 |
| Voltage[V] | | | | | | | | | |
| 5 | 0 | 0 | - | 0 | 0 | - | 0 | 0 | - |
| 4,8 | 0 | 0 | - | 0 | 4 | 1,667 | 0 | 0 | - |
| 4,6 | 0 | 4 | 0,222 | 0 | 4 | 2,667 | 0 | 0 | - |
| 4,4 | 0 | 4 | 0,333 | 0 | 4 | 2,833 | 0 | 0 | - |
| 4,2 | 0 | 4 | 0,444 | 0 | 4 | 2,833 | 0 | 0 | - |
| 4 | 0 | 4 | 1,000 | 0 | 24 | 5,167 | 0 | 0 | - |
| 3,8 | 0 | 8 | 2,222 | 0 | 24 | 5,667 | 0 | 0 | - |
| 3,6 | 0 | 8 | 2,778 | 0 | 24 | 5,833 | 0 | 0 | - |
| 3,4 | 0 | 8 | 3,333 | 0 | 30 | 9,500 | 0 | 0 | - |
| 3,2 | 0 | 12 | 4,111 | 0 | 30 | 9,833 | 0 | 0 | - |
| 3 | 0 | 12 | 6,000 | 2 | 30 | 11,833 | 0 | 0 | - |
| 2,8 | 2 | 12 | 7,444 | 2 | 30 | 12,167 | 0 | 0 | - |
| 2,6 | 6 | 12 | 8,556 | 2 | 30 | 12,167 | 0 | 0 | - |
| 2,4 | 6 | 32 | 10,444 | 2 | 32 | 13,333 | 0 | 2 | 0,125 |
| 2,2 | 8 | 36 | 12,778 | 4 | 32 | 19,833 | 0 | 6 | 0,313 |
| 2 | 8 | 40 | 15,111 | 8 | 32 | 26,833 | 0 | 6 | 0,625 |
| 1,8 | 10 | 44 | 17,333 | 8 | 32 | 27,167 | 0 | 6 | 1,250 |
| 1,6 | 12 | 44 | 19,556 | 24 | 34 | 29,333 | 0 | 10 | 3,063 |
| 1,4 | 14 | 48 | 21,667 | 24 | 34 | 29,833 | 0 | 10 | 5,063 |
| 1,2 | 16 | 56 | 23,333 | 24 | 34 | 30,333 | 6 | 12 | 8,000 |
| 1 | 18 | 66 | 30,111 | 26 | 54 | 37,500 | 6 | 16 | 10,750 |
| 0,8 | 20 | 78 | 56,333 | 34 | 62 | 49,667 | 6 | 24 | 14,750 |
| 0,6 | 60 | 116 | 80,000 | 34 | 68 | 56,500 | 8 | 76 | 46,250 |

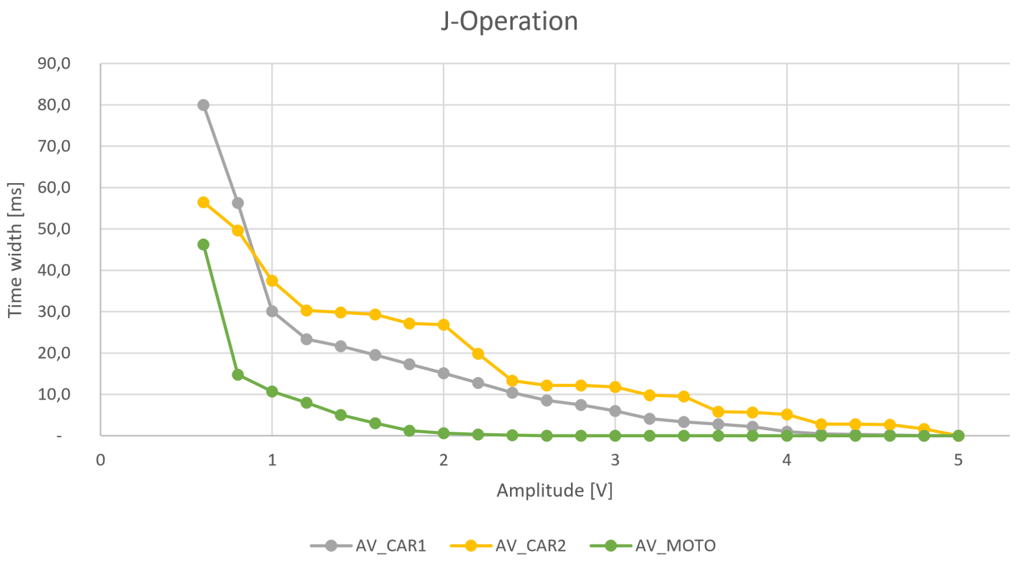


Figure 4.8: The average response per vehicle, this data is not normalized. In general, the responses obtained from the motorcycle are much less larger in amplitude than the responses obtained from both cars.

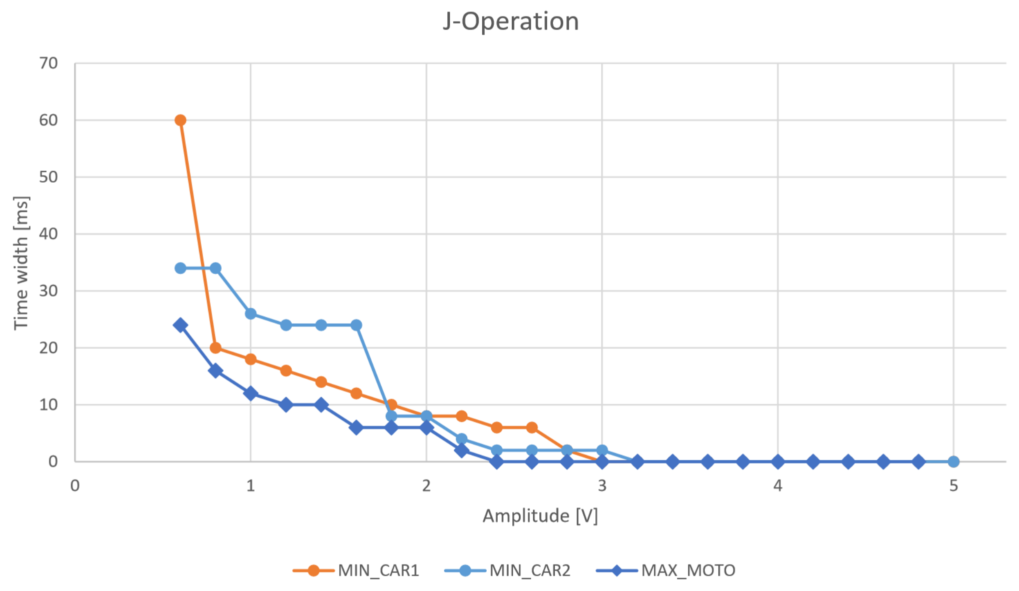


Figure 4.9: The minimum responses obtained from cars in the experiment are greater in amplitude than the maximum responses obtained with the motorcycle.

dicate that there is a correlation between the type of vehicle and the response captured in the PS. We also conclude that heavy motorcycles with wider wheels will probably fall in between the categories.

## 4.2 Pressure Shield

The sensor shield for this project consist of a Printed Circuit Board (PCB) with a series of electronic components that allow us to detect two range of pressure on each tube and implement an efficient interface to obtain data from the microcontroller.

As it is mentioned in the design 3.3.2 the sensor shield will be connected to micro-controller, in this case the Arduino Rev. 3 Board.

Whit the results of the experiment 4.1, it is possible to develop an electronic system capable of differentiate between different types of vehicles in most cases. The main results obtained suggest that at the same velocity, motors produce a different response than cars. The responses of the cars can be a factor 2 (up to 3) bigger in amplitude than the produced by motorcycles.

The shield developed is based on the amplitude of the response captured with the PS in order to differentiate vehicle type. Comparators are used for carrying out the analogical processing. We define 2 thresholds:

`V-CAR` Is the voltage threshold at the output of the PS from which on the event is considered produced by a car.

`V-MOTO` Is the voltage threshold at the output of the PS from which on the event is can be considered as produced by a motorcycle.

**We have to remark that when a car axis passes by both comparators will switch to on, because `V-MOTO` < `V-CAR`. But, there are not triggered at the same time due to the time the pressure takes to rise.**

The schematic design of the shield is in figure 4.11, and the board file is in figure 4.14.

### 4.2.1 Components

In table 4.7 the components required for producing one sensor node are listed.

In the paragraphs ahead we describe the components used for the shield as it is divided on the design: physical, analogical, digital and interface.

#### Physical Pressure

The tubes and the adapter are made of latex. We use an adapter to adjust and fix the tube to the PS. The PS will be the same as in the study done Panasonic ADP51B63.

#### Analogical

The PS will carry out the tasks for transferring the pressure magnitude to an analogical signal and then, amplifying the signal to the voltage range of the microcontroller.

The `V-CAR` and `V-MOTO` values are two analogical voltage constants that came from two voltage divisors shown in figure 4.10. Each voltage divisor is made with AMD resistors and a potentiometer of 5KΩ to adjust the values. We use a potentiometer to adjust the possible differences in the resistance value of the AMD resistors due to a tolerance of 5%. Using the results of the experiment we set:

`V-CAR` must be 2.5 volts. R3 = 26,31$K\Omega$, R4 =23,81$K\Omega$. There are not resitences in the market with that values, we use R3 = 26$K\Omega$ and R4= 23$K\Omega$. AMD values R3 = "263" and R4 = "233".

$$\begin{cases} 5\dfrac{R3\cdot 1.05}{R4\cdot 0.95 + R3\cdot 1.05 + 5K\Omega} = 2.5V \\ 5\dfrac{R3\cdot 0.95}{R4\cdot 1.05 + R3\cdot 0.95 + 0K\Omega} = 2.5V \end{cases} \tag{4.3}$$

`V-MOTO` must be 1 volt. R1 = 105 $K\Omega$ , R2 = 23,75$K\Omega$. AMD values 13 = 104 and R2 = 233.

$$\begin{cases} 5\dfrac{R2\cdot 1.05 + 0K}{R1\cdot 0.95 + R2\cdot 1.05 + 0K\Omega} = 1V \\ 5\dfrac{R2\cdot 0.95 + 5K}{R1\cdot 1.05 + R2\cdot 0.95 + 5K\Omega} = 1V \end{cases} \tag{4.4}$$

#### Digital

We will use the following structure to name the two possible digital states:

**HIGH** Digital values that are higher than 2,9V up to 5V.

**LOW** Digital values that are lower thatn 2,1V down to 0V.

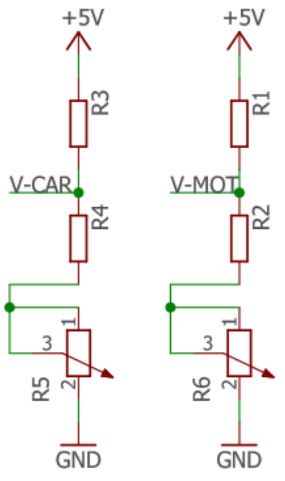


Figure 4.10: The values `V-MOTO` and `V-CAR` are obtained with a voltage divisor. The terminals 1 and 3 of the potentiometers are short circuited to avoid electro-magnetic noise.

The digital part of the shield has two main sections, figure 4.11:

- It has to transfer the analogic signal to a discrete cases. We use the LM358P with two independent OPAMP integrated. Component code in schematic: IC3 and IC4.
  - The PSA output signal (PSA-O) is connected to the 3rd and 5th pins of IC3, that are the positive terminal of the A channel's OPAMP. The signals `V-CAR` and `V-MOTO` are connected to the 2nd and 6th pins of IC3, that correspond to the negative terminals.
    - The output of the OPAMP, pin 1, IC3A is high when the PSA-O value is greater than `V-CAR`. Performing the intermediate signal PSA-C signal.
    - The output of the OPAMP IC3B, pin 7, is high when the PS value is greater than `V-MOTO`. Performing the intermediate signal PSA-M signal.
  - The PSB output signal (PSB-O) pin is connected to the 3rd and 5th of IC4, that are the positive terminal of the B channel's OPAMP. The signals `V-CAR` and `V-MOTO` are connected to the 2nd and 6th pins of IC4, that correspond to the negative terminals.
    - The output of the OPAMP, pin 1, IC4A is high when the PSB-O value is greater than `V-CAR`. Performing the intermediate signal PSB-C signal.
    - The output of the OPAMP IC4B, pin 7, is high when the PSB-O value is greater than `V-MOTO`. Performing the intermediate signal PSB-M signal.
- The other function of the Digital part is to store the information to be accessed later. We use flip-flop registers, the component is the 74HC74N which has 2

Table 4.5: Truth table of the possible cases. Some of them are not possible because `V-MOTO<V-CAR`

| CARA | MOTOA | CARB | MOTOB | INTG | INTCAR | INTB |
|---|---|---|---|---|---|---|
| LOW | LOW | LOW | LOW | LOW | LOW | LOW |
| LOW | LOW | LOW | HIGH | HIGH | LOW | HIGH |
| LOW | LOW | HIGH | LOW | HIGH | HIGH | HIGH |
| LOW | LOW | HIGH | HIGH | HIGH | HIGH | HIGH |
| LOW | HIGH | LOW | LOW | HIGH | LOW | LOW |
| HIGH | LOW | LOW | LOW | HIGH | HIGH | LOW |
| HIGH | HIGH | LOW | LOW | HIGH | HIGH | LOW |
| LOW | HIGH | LOW | HIGH | HIGH | LOW | HIGH |
| LOW | HIGH | HIGH | LOW | HIGH | HIGH | HIGH |
| LOW | HIGH | HIGH | HIGH | HIGH | HIGH | HIGH |
| HIGH | LOW | LOW | HIGH | HIGH | HIGH | HIGH |
| HIGH | LOW | HIGH | LOW | HIGH | HIGH | HIGH |
| HIGH | LOW | HIGH | HIGH | HIGH | HIGH | HIGH |
| HIGH | HIGH | LOW | HIGH | HIGH | HIGH | HIGH |
| HIGH | HIGH | HIGH | LOW | HIGH | HIGH | HIGH |
| HIGH | HIGH | HIGH | HIGH | HIGH | HIGH | HIGH |

independent flip-flops type D inside. Component code in schematic = IC2 and IC7.

- Each signal of the intermediate set of signals: PSA-C, PSA-M, PSB-C and PSB-M. Each signal is connected to the clock pin of independent flip-flops.
- Each flip-flop has its D pin connected to HIGH. When the clock pin rises the output Q pin values HIGH.
- The set pin of the flip-flops are by LOW-value. All of them are connected to HIGH.
- The reset pin of the registers of the same channel are hardwired. IC2A with IC2B and IC7A with IC7B.

• The four output of the registers (CARA, MOTOA, CARB and MOTOB) are connected with an CD4071BE. The CD4071BE is a quadruple OR gate, each one with to inputs. The component is the IC1 to perform the following truth table, which compresses the information from four digital signals to three:

### Interface

From the microcontroller's point of view the shield uses 5 pins, figure 4.11:

**RSA** INPUT Reset of the registers linked to the tube A.

**RSB** INPUT Reset of the registers linked to the tube B.

Table 4.6: Output of the shield to the microcontroller.

| INTG | INTCAR | INTMOTO | EVENT RECORDED |
|---|---|---|---|
| LOW | X | X | Nothing, no event. |
| HIGH | LOW | LOW | A motorcycle on the tube A. |
| HIGH | LOW | HIGH | A motorcycle on the tube B. |
| HIGH | HIGH | LOW | A car on the tube A. |
| HIGH | HIGH | HIGH | A car on the tube B. |

**INTG** OUTPUT An edge has happen in either tube A or B.

**INTCAR** OUTPUT The edge amplitude was greater than the `V-CAR` threshold.

**INTB** OUTPUT Edge localized on the tube B.

In table 4.6 it is shown how with the signals INTG, INTCAR and INTB we understand the events captured.

**Reset Procedure**

The flip-flops connected have a LOW-level reset pin. The reset of the flip-flops of the same channel will be connected together to perform a channel reset:

**RSA** Reset of the A-channel. If HIGH, normal procedure. If LOW, both registers connected to the PS-A are reset.

**RSB** Reset of the B-channel. If HIGH, normal procedure. If LOW, both registers connected to the PS-B are reset.

### 4.2.2 Schematic Design

From the paragraph above we made a schematic file in Eagle which is represented in figure 4.11. The shield will use the following pins of the microcontroller:

5V It is hardwired to the top layer and to VCC of the shield.

GND It is hardwired to the bottom layer and the ground reference of the shield.

PIN 2 The digital signal INTG is hardwired to the 2nd digital pin of the Arduino Uno, which happens to be also a external interrupt pin.

PIN 8 The digital signal INTCAR is connected to the 8th digital pin of the Arduino Uno.

PIN 9 The digital signal INTB is connected to the 9th digital pin of the Arduino Uno.

PIN 10 The digital signal RSA is connected to the 10th digital pin of the Arduino Uno.

PIN 11 The digital signal RSB is connected to the 11th digital pin of the Arduino Uno.

Table 4.7: List of components

| **Microcontroller** | | | |
|---|---|---|---|
| Name | Design | Fabricant Number | Quantity |
| Arduino | microcontroller | Arduino Uno Rev 3 | 1 |
| Charger | Power Source | – | 1 |
| **Pressure Shield** | | | |
| Name | Name in `.sch` | Fabricant Number | Quantity |
| PBS From UIB | PBC | – | 1 |
| FlipFlop D-Type | IC2A, IC2B, IC7A, IC7B | 74HC74N | 2 |
| Pressure Sensor | $PSA, PSB$ | Panasonic ADP51B63 | 2 |
| Comparator | IC3A, IC3B, IC4A, IC4B | LM358P | 2 |
| OR gate | IC1A, IC1B, IC1C, IC1D | CD4071BE | 1 |
| 5k Potentiometer | R5, R6 | - | 2 |
| AMD Resistors | R1, R2, R3, R4 | - | 4 |
| Pin Stripes | 10-PINH, INTGPINH, VCCGND-PINH, UPSIDE DOWN | 2.54mm, 32 | 1 |
| **LoRaWAN Module** | | | |
| Name | Design | Fabricant Number | Quantity |
| Shield adapter to XBEE | Shield adapter | Multiprotocol Radio Shield | 1 |
| LoRaWAN module | LoRaModule | RN2483 | 1 |
| Antenna 868MHz | antenna | - | 1 |
| **Pressure Tubes** | | | |
| Name | Design | Fabricant Number | Quantity |
| Tubes | tube$_A$ , tube$_B$ | – | 2x10m |
| Adapter | tubeAdp$_A$, tubeAdp$_B$ | – | 5cm |
| Holder | holder $_{A1,A2,B1,B2}$ | – | 4 |
| **Display** | | | |
| Name | Design | Fabricant Number | Quantity |
| Display | microcontroller | Inteled | 1 |
| RS485 cable | rs485 | – | Depends, 10-50m |

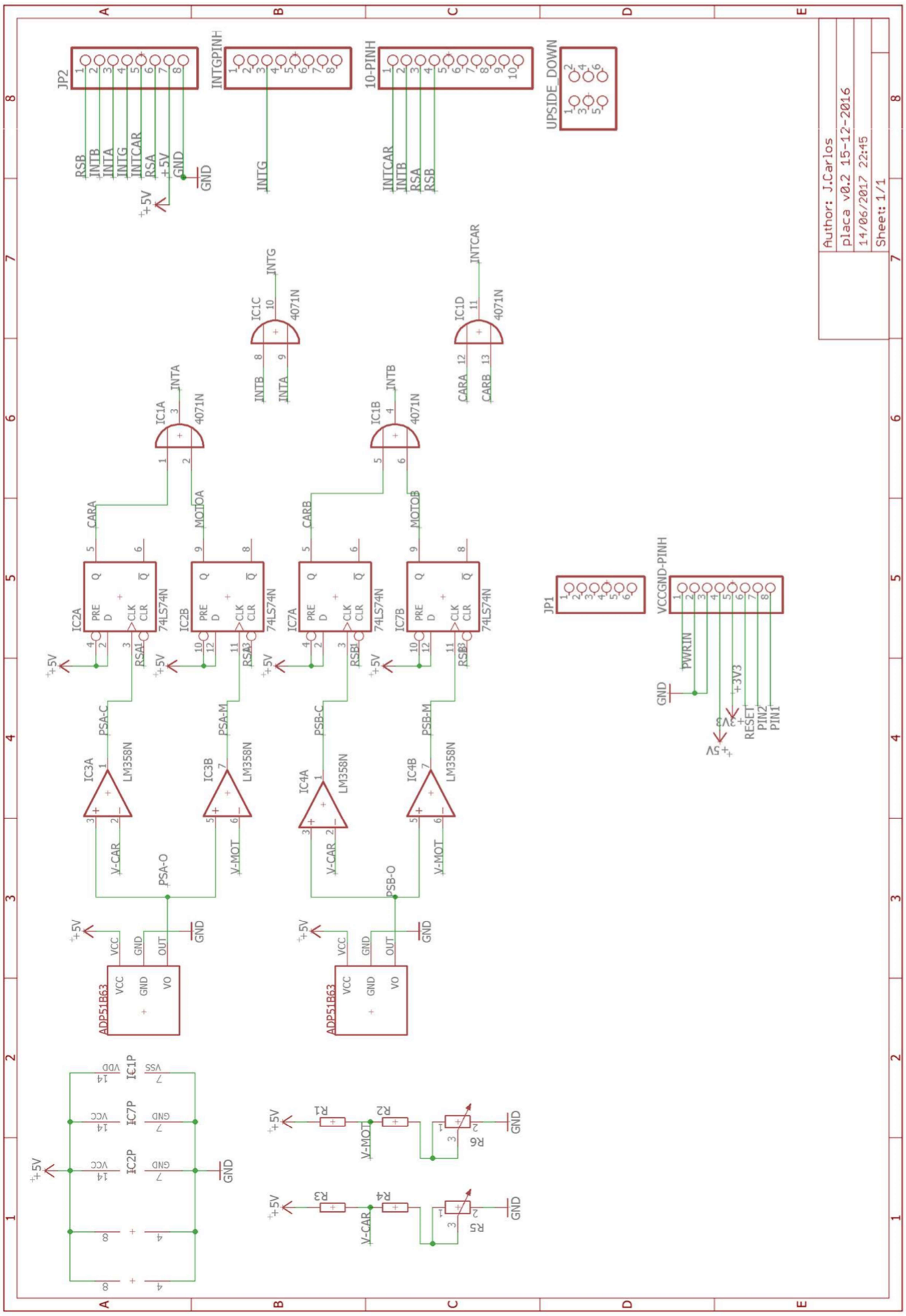


Figure 4.11: Schematic Eagle Design

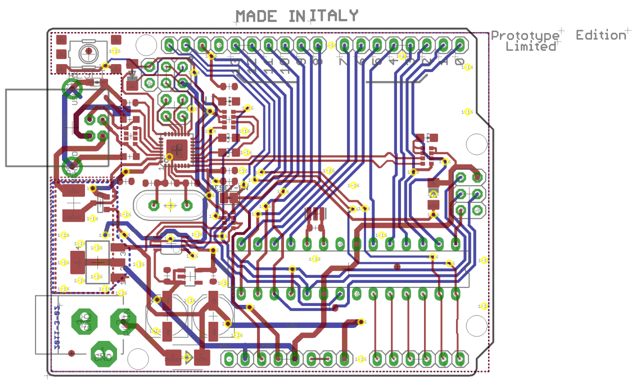


Figure 4.12: Arduino shematics, in red the bottom paths and in blue the top paths. The pin pads are in green. Image from [14].

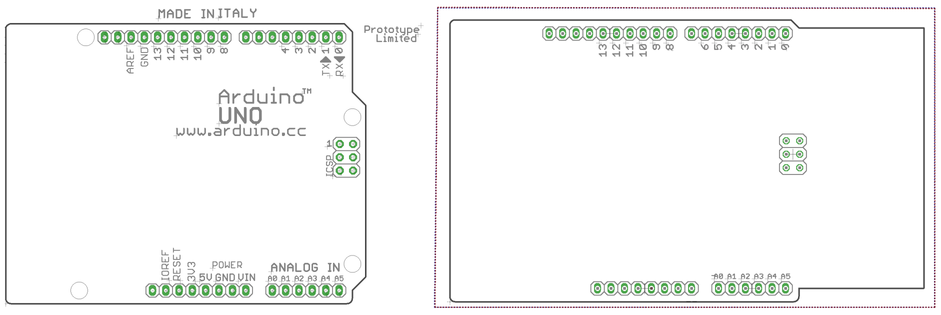


Figure 4.13: Process for creating our shield.

### 4.2.3 Shield for the Arduino Rev 3

The eagle schematics of the latest Arduino Uno revision, Rev. 3, are available online on their web page [14]. The board `.brd` file is represented in figure 4.12.

We removed everything but the pads of the central ISCP and the two string of pads as shown in figure 4.13. We leaved the ISCP pins because the transmit shield, Multiprotocol Radio Shield, need from those pins to connect to VCC and ground. We also expanded the area of the shield allowing the pressure sensor to fit with the transmit module on top of it otherwise, it would be impossible to put the tubes on the pressure sensor.

The top layer of our board is hardwired to the 5V pin of the arduino. Alternatively, the bottom layer of the shield is hardwired to ground with the two ground pins of the arduino. All the components on the shield are connected to VCC = 5V and GND = VSS =0V. Both faces, top in red and bottom in blue, of the PBC in figure 4.14 are designed are used for signal transport.

The final product is in the images of the figure 4.15. With this disposition of components on shield it is also possible to access the potentiometers easily without the need to demount the shield from the microcontroller.

All the documentation and eagle files regarding the shield are in the `PressureShield/pssrShield.zip` file attached with this document.

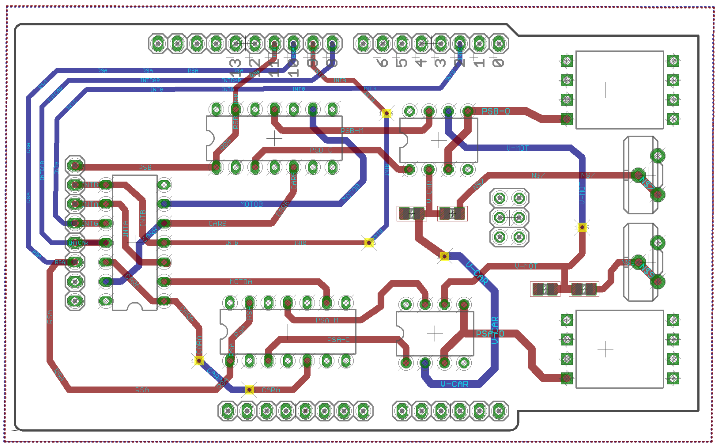


Figure 4.14: Pressure shield `.brd`.

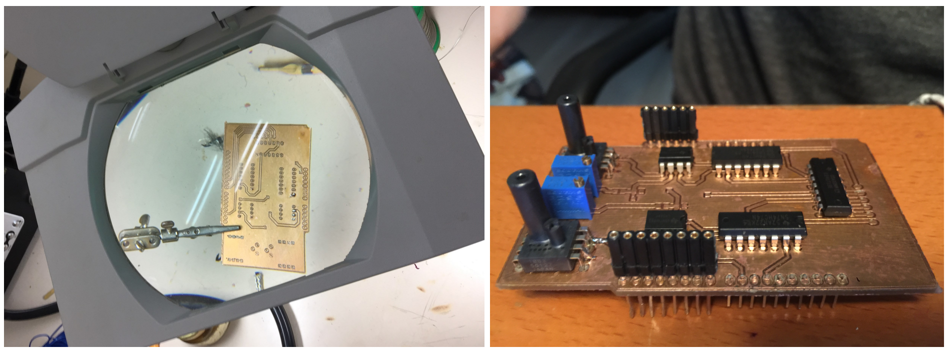


Figure 4.15: The PS under soldering process on the left and the 3-tier device on the right.

## 4.3 Software

In this and following sections, we provide the main highlights of the different logics implemented for the algorithm. We will start commenting on the programming language and compiler options. Then, the algorithm for classifying vehicles and its direction are described in depth in section 4.4. In the section 4.5 we cover how to communicate the value to display to the external screen. Finally, we will enumerate the steps to use the transmit shield and how to log the result in section 4.6. Some peculiarities of the algorithm are in the section 4.7.

### Compiler and IDE

The principal programming language for developing on Arduino is C++. The developers at Atmel, Arduino's microcontroller manufacturer, provide C++ libraries to access the registers of the different microcontrollers. Moreover, the libraries of Arduino use those libraries to perform more global functions.

There are two main compilers for Arduino boards. One from arduino.cc and other from arduino.org. We used the version 1.8.3 of the IDE from arduino.org [18] because it maintains the avr C libraries that include `sprintf_P` functions that are necessary for the LoRaWAN programming interface library. Despite the previous versions of the arduino.cc compiler 1.0.X did have included those functions, from 1.1 and on, these are not included.

The Arduino software has two principal functions to organize the code: `setup()` and `loop` [14]. The setup() function is called when the microcontroller awakens. The setup function will only run once, after each powerup or reset of the Arduino board. The loop() function runs consecutively. It is used to actively control the Arduino board.

#### Arduino Libraries

The code made for this project is divided into two types of files:

- The `main/main.ino` arduino file that holds the logic of the code.
- The c++ libraries. We have made 3 libraries that are extensible to other boards and compilers, because they are written in c++.

  **DisplaySoftSerialUtils** `libraries/DisplaySoftSerialUtils.h` It is used for formatting the frame to send to the display and handle the TIA-484 driver.

  **PressureShieldDriver** `libraries/PressureShieldDriver.h` This library performs the logic for running the pressure shield. It is written to be easy to parametrize with the `#define` c++ commands.

  **EepromUtils** `libraries/EepromUtils.h` It is written to maximize the use of the Eeprom and extend the use from bytes to integers and to be used in different compilers of Arduino.

### 4.3.1 Types used

In order to minimize the use of the memory, we used the minimum type that satisfied the needs for the implementation. In the Arduino compile environment the length of the types are as follows:

| | |
|---|---|
| char | Singed integer of 1 Bytes. |
| byte | Unsinged integer of 1 Bytes. |
| int | Signed integer of 2 Bytes. |
| long | Signed integer of 4 Bytes. |

We also use the type descriptors `volatile` and `unsigned`. The `unsigned` descriptor tells the compiler that the integer variable will be used to save only positive values.

The `volatile` descriptor is used to tell the compiler to fetch this variable every time is used. It is used on all the variables that are read/write in both: during the ISR and outside the ISR. This avoids optimizations of the compiler that are undesired that could lead to dysfunctional behaviour.

### 4.3.2 Interrupt Handler

The Interrupt Handler module is used to focus on the interrupt logic when the pressure shield detects a vehicle.

In the `setup` of our code, it is attached the INTG pin of the pressure shield to execute the function `interruptService()` when it detects a rising edge. This is done with the following line of code:

```
attachInterrupt( digitalPinToInterrupt(intG), interruptService, RISING);
```

Where `intG` is a constant of value 2, that refers to the 2nd pin. `interrputService` is the function to execute when the ISR is triggered and `RISING` tells which kind of mode and direction for triggering. The possible modes are: RISING, FALLING, LOW or CHANGE.

In the code below, it is shown the function `interruptService()`, a flowchart of this function is presented in 4.17. In order to catch every interrupt during the ISR, the function presented is recursive. The condition to call itself is in the 19th line. The 2nd pin (`intG`) is HIGH when the shield detects that one of the two tubes are pressed.

During the first iteration of the Interrupt Service Routine (ISR) it prepares the `Serial` and reduces the time to trigger the Watch Dog Timer (WDT). The 17th line `shield.interruptService()` executes the function to detect and interpret the events produced which will be explained below in the Detection Algorithm section. After, in case that other edge has passed, the function `shield.interruptService()` will executed again until there are no more edges or the number of the next recursion is grater or equal seven. Finally, if the shield is not triggered again the shield, it ends the `Serial` and sets the WDT to the previous value.

```
1  /*
2   * This method runs the shield isr routine.
3   * We run this code into a save interrupt space.
4   * This is a recusrive function that hears the shield to recurse.
5   * The maximum times it can recurse is 7.
6   * (Interrupt Service Routine).
7   */
8  void interruptService() {
9    wdt_reset();
10   deepInInterruptRecursion ++;
11   if(deepInInterruptRecursion ==1){ // only the first Recursion-
       iteration
12     wdt_enable(WDTO_120MS); // Change WD Value to lesser.
13     disableInterrupt();
14     Serial.begin(57600); // Start serial.
15   }
16   //// InterruptServiceRoutine /////////
17   shield.interruptService();
18 
19   if(digitalRead(intG)==HIGH){// Has it happen any other event?
20     if (deepInInterruptRecursion < 7){
21       interruptService();
22     }
23   }
24   deepInInterruptRecursion--;
25   if(deepInInterruptRecursion==0){// Restoring Previous state.
26     wdt_enable(WDTO_8S);// WatchDog to 8 seconds
27     Serial.end(); // End serial.
```

```
    enableInterrupt();
  }
}
```

**Enabling `Serial` during ISR**

In order to use the microcontroller in the ISR as it would be used outside the ISR, we have to add some instructions at the beginning and ending of the ISR.

The problem resides in the fact that during the execution of the ISR all the functions that use the interrupt system are frozen. This happens because during the ISR the microcontroller has disabled the interrupts. We want to use the functions `Serial` and the `millis`, but also avoid the system to lost the pc counter. This is done detaching the interrupt pin from the function and then enabling the interrupts.

```
/*
 * This method does both:
 *  - Disable interrupts.
 *  - Serial & delay keeps working.
 */
void disableInterrupt(){
        noInterrupts();
        // Detach external interrupt.
        detachInterrupt(digitalPinToInterrupt(intG));
        interrupts();
}
/*
 *      This method enable interrupts on the intG pin.
 */
void enableInterrupt(){
        noInterrupts();
        // Returns to the previous configuration.
        attachInterrupt(digitalPinToInterrupt(intG),
            interruptService, RISING);
        interrupts();
}
```

## 4.4 Detection Algorithm

With the information collected from the experiment we can now define some parameters that will be indeed useful for vehicle detection. The functions for detecting vehicles are classified in a stack, figure 4.16. Figure 4.17 shows the workflow of the algorithm of the ISR.

### 4.4.1 Pressure Shield Driver, detecting rising edges

The pressure shield driver implements the microcontroller side of the interface that we have described in 4.2.1. The 2nd and from the 8th to 11th pins of the microcontroller-have to be configured correctly as input or output.

2nd Pin Input with pull-up resister. `pinMode(intG, INPUT_PULLUP);`

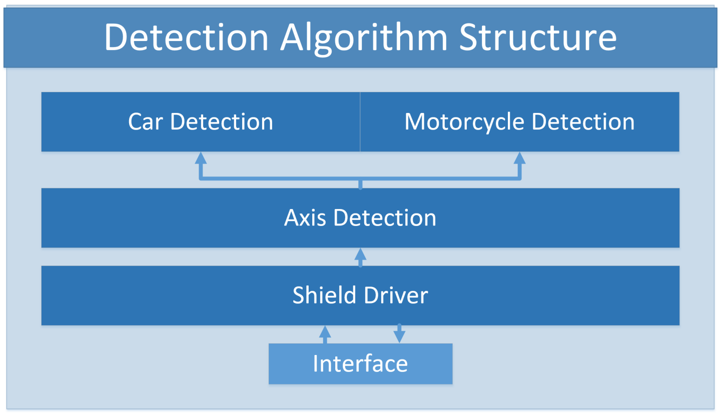


Figure 4.16: The ISR of the microcontroller is triggered when the shield detects a rising edge. Then, the microcontroller reads the shield to know the kind of event recorded. The information is passed to the Axis Detector. The Axis Detector determines whether the event read is from a new axis or a previous one. The information is demultiplexed to the Detectors which performs the same detection algorithm.

8th Pin Input. `pinMode(PIN_IS_CAR,INPUT);`

9th Pin Input. `pinMode(PIN_IS_CHAIN_B,INPUT);`

10th Pin Output. `pinMode(PIN_RST_CHAIN_A,OUTPUT);`

11th Pin Output. `pinMode(PIN_RST_CHAIN_B,OUTPUT);`

### $t_i$ Interruption time

The $t_i$ parameter will describe the fixed time that a interruption will hold the system.

$t_i$ ensures that every interrupt will freeze the node the same amount of time. It does not interfere with the detection algorithm.

**Reading INTB Procedure** After the initialization of the ISR the internal timer value is stored. Next, the pin 9 (INTB) is read. With the initialization of the ISR and reading time we give enough time to read properly the INTB pin.

- If it is HIGH, the edge captured is from the tube B.
- If it is LOW, the edge captured is from the tube A.

**$t_c$, Checking INTCAR time** Considering that `V-MOTO` will be the voltage threshold for detection of any event. As we have seen, `V-MOTO` < `V-CAR` then, the pressure will always trigger `IS-MOTO` before `IS-CAR`.

The properly amount of time to check the INTCAR measured in the experiment is the time that it takes to arrive from 0.5 V to 2.5 V (`V-CAR`) when a car does pass. That is 4ms, we will use 10ms.

**Reading INTCAR Procedure** After reading from the 9th pin (INTB), the following step is to know whether the amplitude was greater than the `V-CAR` threshold. As it is mentioned, the microcontroller has to wait $t_c$ to access the pin 8 (INTCAR).

**Reset Procedure**

**Reset Channel A** Set pin 10 to LOW for 500 $\mu s$, then return it to HIGH.

**Reset Channel B** Set pin 11 to LOW for 500 $\mu s$, then return it to HIGH.

### 4.4.2 Axis Detection

Every time the shield detects an edge on the tubes the axis detector is called within the ISR runtime. The information about type of edge and tube is passed. The goal of this submodule is to determine whether the current edge is from a new axis or it does own to the last axis. It will need from a time stamp, $t_{\text{axis}}$ to decide it.

- A car axis is considered in a tube if the first edge of a series of edges within the same $t_{\text{axis}}$ is greater than the `V-CAR` threshold.
- Alternatively, a motorcycle axis is considered if all the edges of a series of edges within the same $t_{\text{axis}}$ are greater than `V-MOTO` and lesser than `V-CAR`.

#### $t_{\text{axis}}$, Maximum time a axis event can longer

The amount of time that the disturbance can longer it is stochastic but bounded. With the measures made in experiment we can determine an upper limit of time for this event, $t_{\text{axis}}$.

We define $t_{\text{axis}}$ as the maximum amount of time that detections triggered from the same chain will be considered part of the same axis. The maximum value we have measured at its lowest lever `V-MOTO` is 37.5 ms, we extend that value to 120ms because we have not measured all the vehicles at all velocities to know the absolute value.

The implementation of this algorithm is presented above. The parameter $t_{\text{axis}}$ is implemented as a constant `MAX_MICROS_TO_PASS_AXIS`.

```
/*
 * Method to interpret the events in a greater time scale.
 * i represents the tube:
 *              -1 if it is from tube B.
 *               1 if it is from tube A.
 */
void PressureShieldDriver::basicCarDecoder(char i) {
        unsigned long delta = micros() - lastMicro;
        // Tube has changed
        // or more than MAX_MICROS_TO_PASS_AXIS has happen
        if (i == state && delta < MAX_MICROS_TO_PASS_AXIS)
                return; // Edge from latest axis ... ignoring
        // New axis detected ... procesing
        state = i; // Updating state
        lastMicro = micros(); // Updating the latest time
        if (lastMicro==0){      lastMicro = 1;  }
        unsigned long d_milis = millis() - lastEdgeMilis;
```

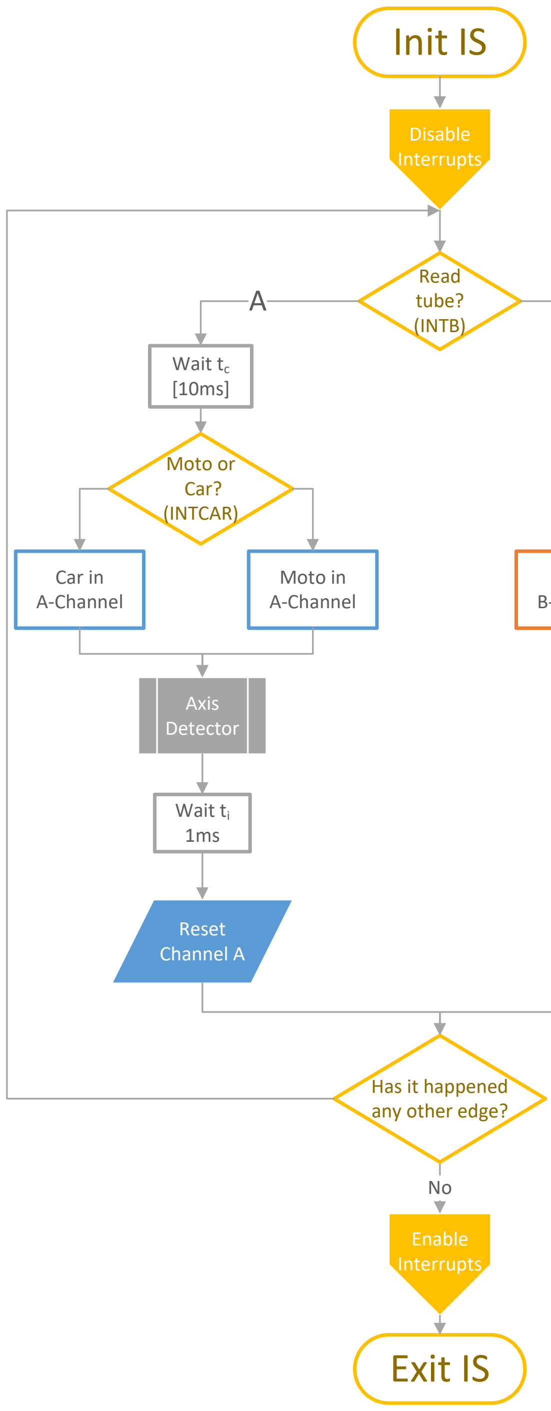


Figure 4.17: Interrupt Service Runtime Flowchart. The parts included are Shield Driver and Interface. The interface blocks are: reading from the digital inputs to know which tube is pressed, discriminate from edges of cars or motorcycles and resetting channels. The driver reads properly the inputs. Once it is differentiated, the information is passed to the axis detector. Before leaving the ISR, we process other event if has happened.

```
        decideCar(i);
}
```

### 4.4.3 Vehicle Detection

The axis information obtained with the algorithm described in the section above is passed to the vehicle detection module. Now, it is simpler to design an algorithm to capture the behaviour of the vehicles.

The figures 4.18 and 4.19 show the diagrams of the events produced by a car leaving and entering the parking respectively. At the top of the figures we have a diagram of the scene where we can see the tubes A and B separated by a distance $d_{\text{tube}}$, the car and its direction. On the bottom, there are two representations of the axis events $A(t)$ and $B(t)$ and a variable `iCar` that will be explained below.

#### Vehicle leaving the parking

As it is shown in figure 4.18, when a car leaves the parking, it firstly press with the frontal axis the tube A and then, the tube B. The sequence of this information is used to perform the detection algorithm.

The algorithm for detecting cars is triggered when the system detects an axis caused by a car.

- The events are labelled in order of arrival $t_0 - t_3$. The first one is caused by the frontal axis pressing the tube A. The second is also caused by the frontal axis. Both axis events are separated by a time defined by the velocity $v_{\text{car}}$ of the mobile and the separation between the tubes $d_{\text{tube}}$.
- The sequence recorded is: #A, #B, #A and #B.
- We define a variable `iCar` that will be our variable to perform the algorithm:
  - If the event is caused by #A the value `iCar` is decreased by one.
  - Otherwise, if it is a #B is increased by one.
- In idle state the variable `iCar` is 0.
- When a car leaves `iCar` varies twice from `-1` to 0.
- **When `iCar` varies two times from -1 to 0 the system add one to the leaving car counter.**
- Otherwise, **if `iCar` varies two times from 1 to 0 the system counts an entering car.**

On the other hand, we have in figure 4.19 the diagram of events that are produced when a car is entering the to the parking. We remark that in this figure the value taken by `iCar` is opposite to the one obtained in figure 4.18.

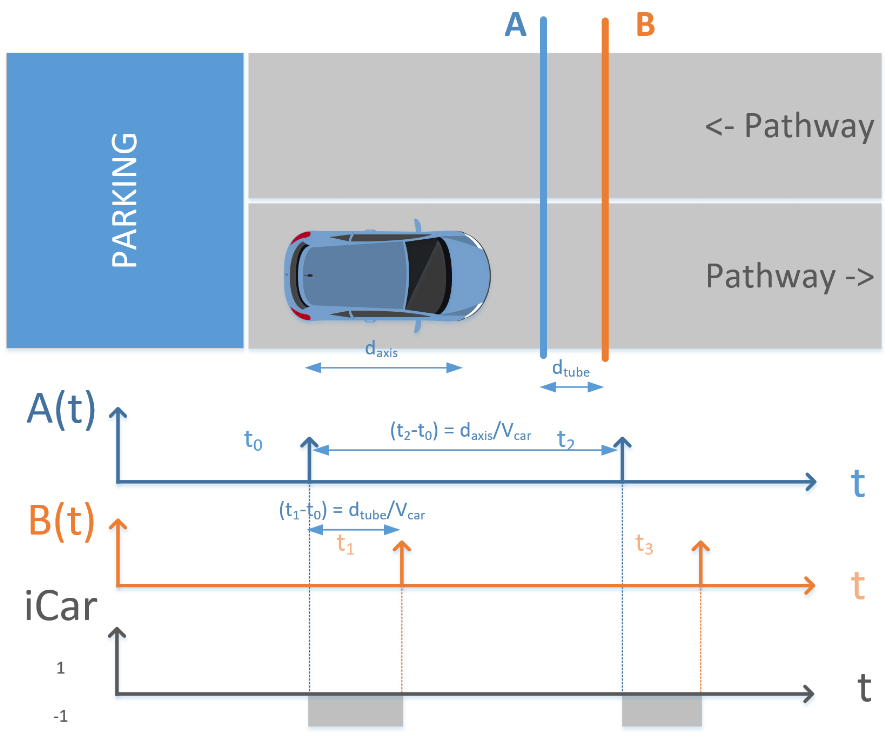


Figure 4.18: Car leaving the parking going to pass over the tubes.

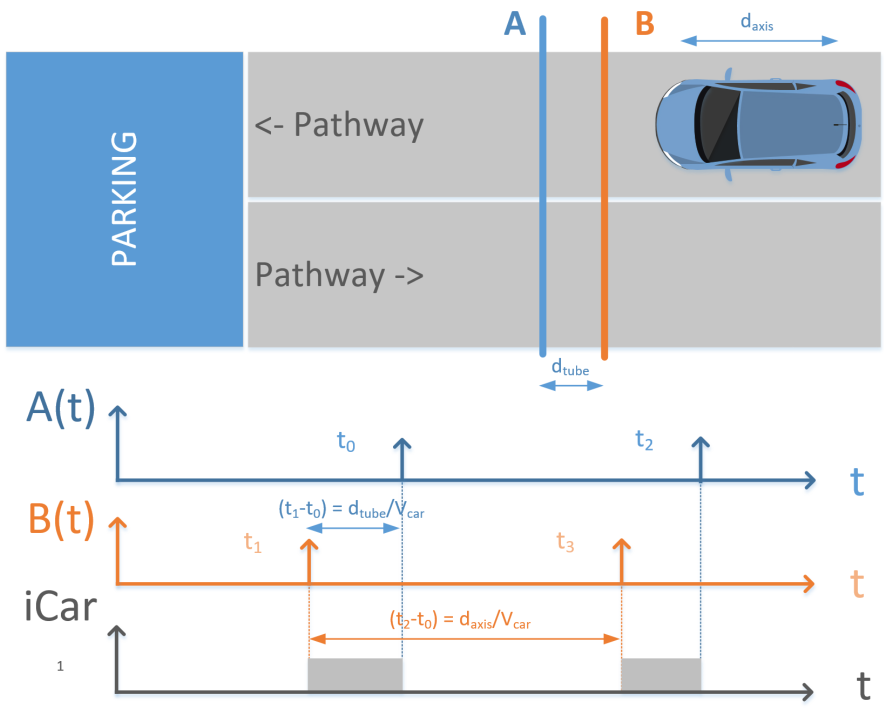


Figure 4.19: Car entering the parking going to pass over the tubes.

### $t_{\text{Btw axis}}$, Vehicle maximum time between axis to pass

In case of any error of counting edges properly, may it be for the coincidence of two cars simultaneously stepping the same tube and the system not being able to differentiate them. We save our algorithm of being chaotic just adding a simple time record $t_{\text{Btw axis}}$.

We will define $t_{\text{Btw axis}} = 3$ seconds as the maximum amount of time that an event should be considered from the latest vehicle detected. Once this time has passed, the new event is interpreted as new. This is indeed useful when considering 3-axis vehicles as we will see in section 4.7.3. The parameter $t_{\text{Btw axis}}$ is implemented above as `MAX_MILLIS_BETWEEN_HALF_VEHICLE`.

```
/**
 * This method passes from car axis detected to
 * inputs or outputs of cars.
 * b can obtain to values:
 *              -1: car axis in tube B.
 *               1: car axis in tube A.
 *
 * It uses the variable iCar to perform it.
 */
void PressureShieldDriver::decideCar(char b){
        iCar =iCar+ b;
        if ((32+iCar)%2 !=0) return;

        if(iCar==0){
                if((iCar-b)>0)
                        addOneToInHalfAxis();
                else
                        addOneToOutHalfAxis();
        }else{
                if(iCar>0)
                        addOneToInHalfAxis();
                else
                        addOneToOutHalfAxis();
        }
}

/*
 * If two half in-car-axis are detected.
 * -> We add one to the inCar.
 * Always taking into account
 * MAX_MILLIS_BETWEEN_HALF_VEHICLE.
 */
void PressureShieldDriver::addOneToInHalfAxis(){
        if (inHalfCar==0){
                lastestHalfInCar = millis();
                inHalfCar++;
                return;
        }
        unsigned long elapsed = millis()-lastestHalfInCar;
        #if DEBUGGING_SERIAL > 2 //if debug ON print auxiliar output
                Serial.println(elapsed,HEX);
        #endif
        if(elapsed<MAX_MILLIS_BETWEEN_HALF_VEHICLE){
                inCarCount ++;
                #if DEBUGGING_SERIAL > 0 //if debug enabled print...
46                              Serial.println(F("#I"));
47                      #endif
48                      inHalfCar=0;
49              }else{
50                      // inHalfCar is already set to one
51                      lastestHalfInCar = millis();
52              }
53      }
```

We used the `#if - #endif` (lines 42-44 and 65-67) C++ compile flow commands to indicate if we want to compile the output through serial.

We made a `PressureShieldDriver` object `"shield"` in `main.ino` to perform the high-level functions of the shield. The public functions are:

**shield.interruptService()** It implements all the logic presented for the ISR except the interrupt functions: `Interrupt Disable` and `Interrupt Enable`.

**shield.setupPressureShield()** To set the pin modes to be use correctly the shield.

**shield.inCarCount** To access the number of cars that have entered.

**shield.outCarCount** To access the number of cars that have leaved.

**shield.inMotoCount** To access the number of motorcycles that have entered.

**shield.outMotoCount** To access the number of motorcycles that have leaved.

### 4.4.4 Testing the Shield with vehicles

In figure 4.20 we can see how the parameters commented where implemented in an Arduino Uno with the shield and tested on the road with real vehicles. We found this test successful as we could know the vehicle direction and differentiate from cars and motorcycles.

## 4.5 Display Controller

As it was commented in 3.3.3, while driving to the parking, the users have to be able to know how many free places are in each parking. If the drivers have this information their time invested searching spots will be reduced.

Then, we will need a display configurable with low rate, low cost communication standards. We acquired the display from the Spanish provider Inteled.

The display has the following characteristics:

- Three color configurable: Green, amber and red. The amber is produced by enabling green and red leds simultaneously.
- 3-digits. Each digit is a 7x8 led matrix display.
- Configurable with RS232/RS484, TCP/IP or Wi-Fi. RS232/RS485 is the preferred option.
- Auto-bright with a fotoresistor.

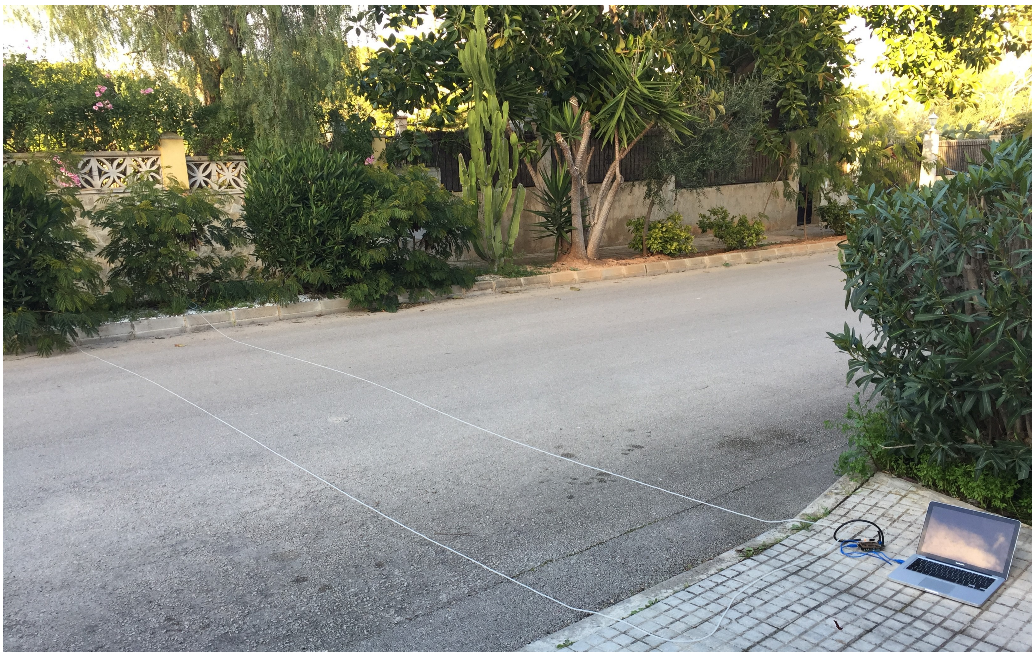


Figure 4.20: Testing the configuration with: $t_c = 10$ms, $t_{\text{axis}} = 120$ms, `V-CAR` = 2.5V and `V-MOTO` = 1.2V.

- IP67 water-dust protection.

We have chosen RS485, instead of RS232 as the main communication protocol with displays. This communication bus, now called TIA-485 (because since 2003 is managed by Telecommunication Industry Association) is more suitable in this scenario because of:

- It needs less cables.
- It does not need a complex connector, only stripped cables. The less connections, the more simple.
- Low voltage drivers, less hardware.
- More resistant to interference as the signal is differential. It does not need ground reference.
- 485 covers lot more distance, up to 1.200 m in front of 50 feet of the 232.

### 4.5.1 Serial Communication

In order to communicate the display and the micro-controller, we have to include a driver in the shield. The driver used is the *MAX485* that allows using a TIA-485 bus with only one chip. A general diagram of the connection is in 4.21. We use a half duplex bus at 9600 baud.

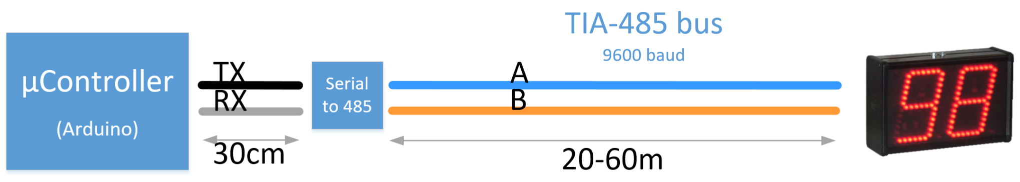


Figure 4.21: Although node and display are separated up to 60 meters, the microcontroller transmits/receives data.

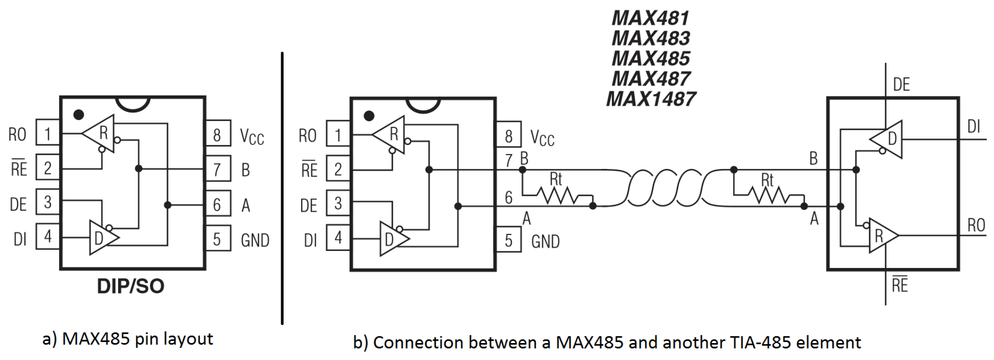


Figure 4.22: TIA 485, two 485 drivers communicate over a 485 bus, figure from [19].

### MAX485 pin layout

MAX485 [19], consists of a unique chip to bridge between serial and TIA-485. Apart from the VCC-GND connection it has 6 pins. Two pins are used for 485 connection (A & B), other two for serial (RX & TX). Our 485 bus is half-duplex, two wires, to enable reading or writing from serial it has two pins DE (Drive Enable) and $\overline{\text{RE}}$ (Read Enable). A detailed diagram of the connection of the device is in figure 4.22. Before transmitting Drive must be enabled and Read disabled $\longrightarrow$ DE = $\overline{\text{RE}}$ = HIGH. Otherwise, if we want to read from the bus the pin state must be the contrary $\longrightarrow$ DE = $\overline{\text{RE}}$ = LOW.

## 4.5.2 DTPM Communication Protocol

Data Transport Protocol Machine (DTPM) is a standard designed for exchanging data between simple nodes. It has numerous capabilities and use scenarios related to data and commands for nodes.

We will use DTPM over TIA-485 to tell the display the coloured digits it has to print on, and for turning it off.

In 4.23 the frame structure of the message is shown. The header has the id of the recipient. The data includes the commands and data relevant to the target. The checksum is a two-byte field valued with the sum of the header and data fields.

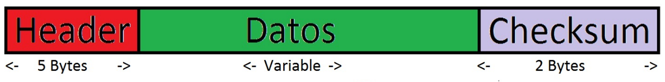


Figure 4.23: DTPM frame composition.

Table 4.8: List of data commands used for this project

| **Data commands (0x03)** | | | |
|---|---|---|---|
| Name | Description | pre-token | value |
| Colour<**n**> | From here the colour will be the **n**th | 0x03 \| 0xA1 | '1'(0x31) red or '2'(0x32) green and '3' (0x33) amber |
| Line<**n**> | Text will be displayed on the **n**th line | 0x03 \| 0xC7 | From '1' (0x31) to '5' (0x35) |
| Bright<**n**> | Display bright as a percentage | 0x03 \| 0xD0 | '1' to '100' %, '0' is automatic |

**Commands**

The command is a byte field in the header section. The command 0x27 known as FASTEXEC tells the display to execute the bytecode/program in the data field.

The manufacturer recommends following this data field order: Data, mode, time and effect. For this project we will only use data instructions. The data instructions include: blinking, changing colour, line and brightness among others. In table 4.7 we present the commands included in messages. A null terminator 0x00 is needed at the end of the program.

**Brightness** We set by default automatic brightness. The display uses a photoresistor to adjust bright on the fly.
**Example: Set automatic bright** The data field must be 0x03 0xD0 0x30.

**Colour** Even though the display is able to show digits in three colours (red, green and amber), the manager of the parking told us to use only the red and green ones.

To clarify the users of the parking we set 2 color categories:

- If the free places are less or equal than 9, we print the value in red.
- Otherwise the value is printed in green.

**Example: Print '5' in red** The data field should be: 0x03 0xA1 0x31 0x35.

**Line** The display with 5 lines needs the microcontroller to specify the line where the information goes.
**Example: Print '10' in the second line** The data field must be 0x03 0xC7 0x32 0x31 0x30.

### Frame example for printing 15 in green: HEADER + DATA + CHECKSUM

If we want to send the value 15 to the display we may use the three digits like `0 1 5`, but we will print a blank space on the first digit to result be `_ 1 5`. Because 15 is greater than 9 we set the color to green. And as we said, it is necessary to put a null value (0x00) at the end of the program. **Final frame:** [0x16, 0x0d, 0x00, 0x01, 0x27, 0x03, 0xc7, 0x31, 0x20, 0x31, 0x35,0x00, 0xcc, 0x01]

### Frame example for turning off the screen: HEADER + DATA + CHECKSUM

The way to turn off all leds of the screen is to send a fast execute frame command with the mode field **now 0x04,0xf0**, and a null terminator 0x00 at the end.

**Final frame:** [22, 16, 0, 1, 39,3, 199, 48, 4, 240, 32, 32, 32, 0, 156, 2]

### SoftwareSerial

The MAX485 converts the TTL serial of their RO (Receive Output) and DI (Drive Input) to 485 symbols that will propagate through the 485-bus. The microcontroller has built-in support for serial communication on pins 0 and 1, but they are already connected to the transmit module. The `SoftwareSerial` library allows serial communications on other digital pins of the Arduino.

The connection of the pins to the microcontroller are described below:

GND To the 7th digital pin of the microcontroller.

RO To the 6th digital pin of the microcontroller. Which is also RX of the `SoftwareSerial`

DE and RE Both pins are hardwired to the 5th digital pin of the microcontroller.

DI To the 4th digital pin of the microcontroller. which is also the TX of the `SoftwareSerial`.

VCC To the 3th digital pin of the microcontroller.

In order to enable communication, we have to set the pin modes. This is done in the function `setupSoftSerial()` presented above. During this function the `SoftwareSerial` sets up the pin modes to interface correctly the MAX485 and prepares the serial communication.

```
/* regarding to the MAX485:
 * if both RE_pin and DE_pin are high, transmission is enabled.
 * if both RE_pin and DE_pin are low, reception is enabled.
 */
byte GND_pin = 7;
byte TX_pin = 6;
byte MODE_pin = 5; // hardwired to RE and DE
byte RX_pin = 4;
byte VCC_pin = 3;
/*
                Method for declaring the pins correctly.
 */
void DisplaySoftSerialUtils::setupSoftSerial() {
        // Open serial communications and wait for port to open:
        pinMode(TX_pin, OUTPUT);
        pinMode(RX_pin, INPUT);
        pinMode(MODE_pin, OUTPUT);
        // VCC
        pinMode(VCC_pin, OUTPUT);
        digitalWrite(VCC_pin, HIGH);
        // GND
        pinMode(GND_pin, OUTPUT);
        digitalWrite(GND_pin, LOW);
        // Inteled's display works with 9600 baudrate.
        ser.begin(9600);
        ser.flush();
}
```

### Sending Frames to the Display: Turn Off the Leds

With the same procedure of this section, we introduce the instructions to be followed to transmit data to the display. The implementation to shut down the screen of the display is found below. At the beginning we found the definition of the array `msgTurnOffLeds`. This DTPM message is written to turn off the 3 digits of the screen. In the function `turnOffLeds(byte line)` defined above, copies this array onto another array. This array is modified to specify the **line** passed as an argument `line` and also correct the **checksum** at the ending of the frame. Once the frame is formatted, it is passed to the function `sendFrame(byte frame [],byte frame_length)` that use the SoftwareSerial object created to send the bytes to the MAX485 which will transform them to TIA-485 symbols.

```
// predefined message for turning leds off on the 0-line
const byte msgTurnOffLeds [16] = {22, 16, 0, 1, 39, 3, 199, 48, 4,
    240, 32, 32, 32, 0, 156, 2};

/**
 * This methods performs a shutdown of the sceen.
 * It prints 3 blank spaces.
 * Used specially for maintenance.
 */
void DisplaySoftSerialUtils::turnOffLeds(byte line) {
    setupSoftSerial();
    delay(100); // wait to set every pin correctly.
    byte frame_length = 16;
    byte msgToSend [frame_length];
    memcpy(msgToSend, msgTurnOffLeds, frame_length);
    msgToSend[7] += line ;
    msgToSend[frame_length - 2] += line;
    sendFrame(msgToSend, frame_length);
    unsetupSoftSerial();
}
```

To shut down the screen, the function should be called as `display.turnOffLeds(1);`.

### Sending Frames to the Display: Displaying Coloured Numbers

Analogously, we used the predefined array `msgOneZeroRed` to modify it and sendit to the screen. `msgOneZeroRed` is a DTPM frame to show: "_ _ 0" in red. where "_" is a blank space. By modifying the byte the bytes that specifies the line, color and the numbers we can show any number of three digits in one of three colors: red, green and yellow.

The function `sendFrame(int value, byte line)` carries out the tasks for detecting which color and numbers to display. This information is passed to the function `sendFrame(byte a, byte b, byte c, byte line, byte color)` that makes the array to be sent to the function `sendFrame(byte frame[],byte frame_length)` previously defined.

To print a 100 green in the first line the function has to be called as `display.sendFrame(100,1);`.

## 4.6 Transmit Module: Microchip RN2483

We used the RN2483 module from the manufacturer Microchip. This module enables LoRa and LoRaWAN Class A communications with a high-level UART interface. This modules implements the full LoRaWAN stack for receiving (decrypting) and transmitting (encrypting) LoRaWAN messages.

The LoRaWAN Class A devices can join the network in two ways: OTA and ABP. The OTA stands for: Over The Air activation, additionally ABP stands for Activation By Personalization. The OTA uses a pre-shared key to perform an exchange of encryption keys over the air to log in the LoRa network. The ABP uses pre-shared keys to transmit directly without the use of previous transmissions.

The information needed to set up a LoRaWAN communication is:

- **Device Address** 32 bits.
- **New Session Key** of 128 bits.
- **Application Session Key** of 128 bits.

The payload is encrypted with a 128-AES cipher in the RN2483.

### 4.6.1 Functions Performed

#### Configuring the Module: Set up

Most parameters of the module can be saved on the non-volatile memory of itself. These parameters are loaded in the set up of the microcontroller.

| | |
|---|---|
| Device Address | The unique identifier used by the node to transmit. |
| New Session Key | The key used for encrypting LoRaWAN network commands. Port = 0. |
| App Session Key | Key used for encrypting application data. Port $\neq$ 0. |
| Auto Data Rate | Sets the ADR on to automatically hear the instructions of the gateway related to transmit power and data rate. It is useful because the network managers can change parameters to optimize the network. |

**Configuring the Module: Before Transmission**

There are some parameters that are not saved into RN2483's EEPROM and therefore, they must be passed every time the module awakens. The volatile parameters are:

Auto Reply It is used to declare to the node: To stay hearing the radio interface if the gateway declares more frames to send (Pending Bit Flag). And to answer with an Ack to the gateway after receiving a confirmed downlink.

ReTX Sets number of ReTXransmission in case of failure. The default parameter is 7 which is very inconvenient because can hold up to one and a half minutes the node only waiting. ReTX is set to 0. In case of failure, do not retransmit. Which also makes collisions less probable.

Join ABP They keys are stored but it has to perform this function before transmitting to read the module's EEPROM memory and load the keys.

### 4.6.2 API: Protocol with Server

The protocol to exchange information with the server is a oriented-to-byte protocol with fixed fields in the message. The information exchanged varies with the device type.

#### Uplink

The node transmits the latest information about the cars that passed during the last successful transmission. It also transmit the values `Flow Threshold` and `Time Threshold` that are used to define when the device transmits.

The fields are disposed as it is shown in figure 4.24.

**Time Threshold** 1 unsigned byte. Time from latest transmission to trigger another transmission.

**Flow Threshold** 1 unsigned byte. An integer number that represents the total number of vehicles that has passed by the node from the latest transmission, another transmission will be performed.

**inCar** 1 unsigned byte. Number of cars that went in from the latest transmission.

**outCar** 1 unsigned byte. Number of cars that left from the latest transmission.

**inMoto** 1 unsigned byte. Number of motorcycles that went in from the latest transmission.

**outMoto** 1 unsigned byte. Number of cars that left from the latest transmission.

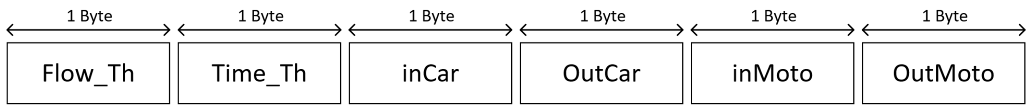


Figure 4.24: Uplink frame structure.

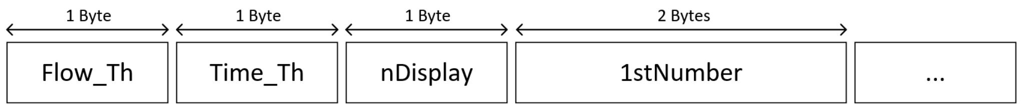


Figure 4.25: Downlink frame structure.

### Downlink

The information received form the gateway is used for parametrize the communications and for specifying the number to display, if necessary.

The fields are disposed as it is shown in figure 4.25.

**Time Threshold** 1 unsigned byte. Value to update in the parametrization of the node.

**Flow Threshold** 1 unsigned byte. Value to update in the parametrization of the node.

**nDisplay** 1 unsigned byte. Number of displaying lines attached to the node. The nodes **S0**, **S2**, **S4** and **S7** are connected uniquely to a display, then the field values **1**. The nodes **S1**, **S3**, **S5** and **S6** are not connected to any display, this field values **0**. The node **DM** is connected to a display that shows the 5 values of the parkings.

**1stNumber** (S0,S2,S4,S7 and DM) If nDisplay > 0. 2 unsigned byes. This field tells the node which number to display in the first line of the screen. If both bytes are ones, **0xFFFF** it means to turn off the leds of that line.

**2ndNumber** (Only DM) If nDisplay = 5. 2 unsigned byes. This field tells the node which number to display in the second line of the screen. If both bytes are ones, **0xFFFF** it means to turn off the leds of that line.

**...**

**5thNumber** (Only DM) If nDisplay = 5. 2 unsigned byes. This field tells the node which number to display in the fifth line of the screen. If both bytes are ones, **0xFFFF** it means to turn off the leds of that line.

### 4.6.3 Send Message Algorithm

#### Transmission Procedure

In the main loop, the microcontroller is rady to get interrupted by events produced on the pressure tubes. The events are sent to the gateway which will send them back to the

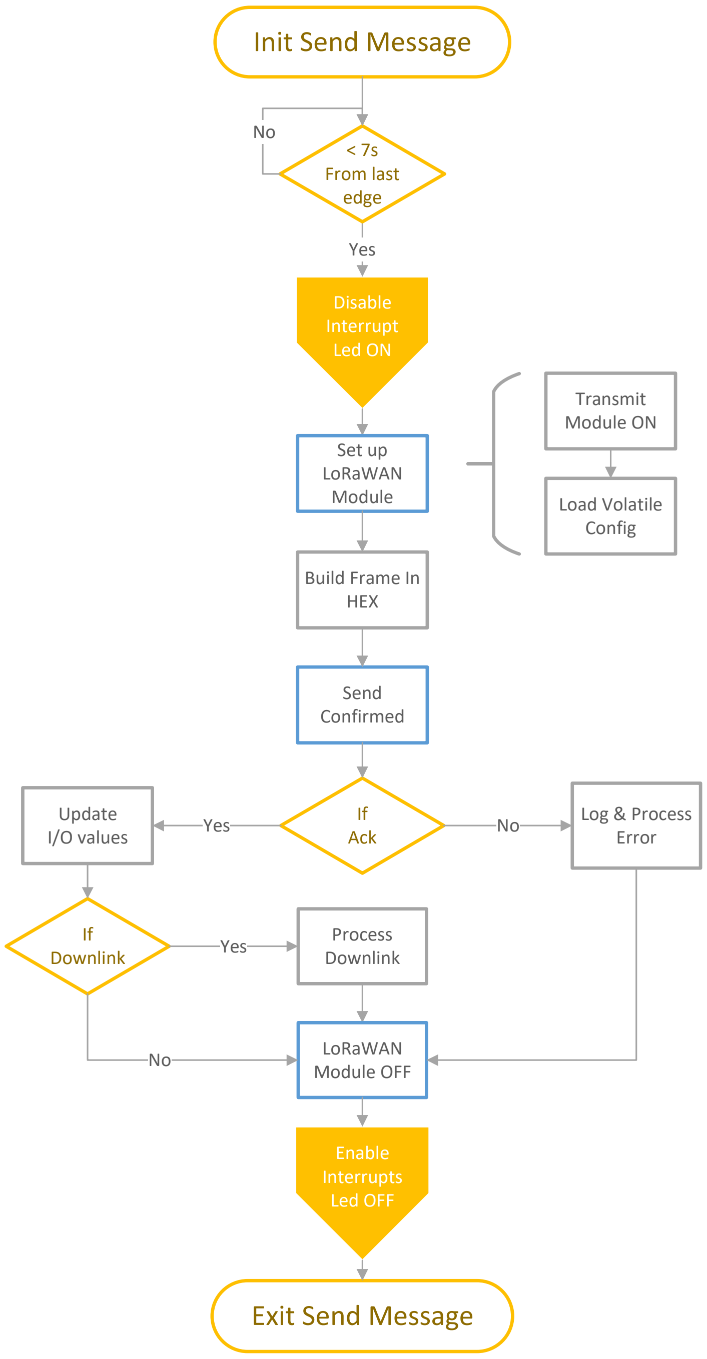


Figure 4.26: Send Message procedure flowchart. In order not to break flow of captured events, the system waits until more than 7 seconds from the latest captured axis. Then, it disables the interrupts as defined in the Interrupt Handler module to later set up the volatile configuration of the transmit module. After formatting the uplink frame, the microcontrollerpasses it to the transmit module, with the application-port 3 (for parking). If the transmission was successful and a downlink was received, the microcontroller processes the payload. If there was a transmission error it is handled by the Error Handler module in the function `error_plus_plus()`. Finally, the microcontroller enables the interrupts to be able to detect vehicles.

server which is the one that performs the counting of the parking zones. A flowchart of this is visualize in figure 4.26.

The implementation of the transmission procedure described in figure 4.26 is found below.

```
/*
 * This method is for sending data to the server side.
 * The types of messages are predefined.
 * Frame: All fields are little endian.
 * +-------+-------+-------+-------+-------+-------+
 * | F_TH  | T_TH  | inCar | outCar| inMoto|outMoto|
 * +-------+-------+-------+-------+-------+-------+
 *    1B      1B      1B      1B      1B      1B
 */
void sendMessage(){
  // Seven seconds from latest edge.
  while ((unsigned long)(micros()-shield.lastMicro) < 7000000 ){
    delay(500);
    wdt_reset(); // reseting WDT
  }
  wdt_reset(); // reseting WDT
  digitalWrite(13,HIGH);/// LED ON
  disableInterrupt();
  byte error;
  error = LoRaWAN.ON(socket);
  wdt_reset(); // reseting WDT
  error = LoRaWAN.setAR("on"); // set automatic reply to on.
  error +=LoRaWAN.setRetries(RETRANSMISSIONS_LORA_WAN); // max
      number of retransmissions.
  error += LoRaWAN.joinABP();
  byte data[6];
  // formatting data
  data[0] = FLOW_THRESHOLD;
  data[1] = TIME_THRESHOLD;
  data[2] = shield.inCarCount;
  data[3] = shield.outCarCount;
  data[4] = shield.inMotoCount;
  data[5] = shield.outMotoCount;
  // From byte to HEX String
  char dataS [12];
  for(char j = 0; j < 6; j++){
    sprintf(&dataS[2*j], "%02X", data[j]);
  }
  wdt_reset(); // reseting WDT
  error = LoRaWAN.sendConfirmed( PORT, dataS);
  wdt_reset(); // reseting WDT
  if(error == 0){
    // No errors found, ACK received. Update values.
    shield.inCarCount -= data[2];
    shield.outCarCount -= data[3];
    shield.inMotoCount -= data[4];
    shield.outMotoCount -= data[5];
    if (LoRaWAN._dataReceived && strlen(LoRaWAN._data) > 0){
      // Downlink received, process output.
      processDataReceived();
    }
    // Save data properly.
52      eeprom.writeByte(LAST_CAR_IN_ADDR,shield.inCarCount);
53      eeprom.writeByte(LAST_CAR_OUT_ADDR,shield.outCarCount);
54      eeprom.writeByte(LAST_MOTO_IN_ADDR,shield.inMotoCount);
55      eeprom.writeByte(LAST_MOTO_OUT_ADDR,shield.outMotoCount);
56      eeprom.writeInt(ERROR_COUNTER_EEPROM_FIRST_ADDR,0);
57      error = LoRaWAN.OFF(socket);
58    }else{
59      error_plus_plus();// Process error
60    }
61    wdt_reset();
62    enableInterrupt();
63    digitalWrite(13,LOW);// LED OFF
64  }
```

### Process Data Received

If the ack transmission is found to have a payload (line 49th) then, a downlink was delivered. The information of the payload from the RN2483 is an hexadecimal string. We can access information about the downlink the following parameters:

`._dataReceived` It is a flag that tells whether there was payload with the Ack of the confirmed message.

`._data` It is a string (char array) of hexadecimal characters representing the payload of the received frame.

The microcontroller has to convert the hex string to byte string. If the length is 2 bytes then only the `Flow Threshold` and the `Time Threshold` are updated. The microcontroller gets the number to display from the 4th and 5th byte of the frame. It access that information checking first that the field `nDisplay` is set to one.

If the bytes are set to 0xFFFF then, it sends a message to the screen to turn off the leds with the `display.turnOffLeds(1)`. Otherwise, the information is sent to the screen with the `display.sendFrame(value)` being value an unsigned integer with length two bytes.

### Running out of memory

The Arduino has very limited resources. Low processing power and low memory. The programming interface library to use the transmit module is a very large C++ class that has a lot of functions to access and set all the parameters of the RN2483. It also has multiple attributes to save the configuration obtained from the module and be accessed programtically accessible. This is in fact a good thing because it enables an easy access to the variables of the RN2483. But once we have defined all our configuration and all the parameters we are going to use, there are many of the attributes of the class that never are accessed and yet, had reserved memory on the Arduino's RAM.

The problem was found to be that with the code from the libraries created and the manufacturers', the code may work until an undetermined time e.g. 1h. Then, it would start losing the flow control of the logic and finally halt.

This problem was solved by firstly reducing the size of our libraries and then, reading the library of the manufacturers and realizing where the great volume of memory was being wasted. The library was purged to adjust to the final use. **After reducing the manufacturer library, the problems related to random malfunction of the microcontroller were resolved.**

#### TX duration trade-off

Even though the time over the air varies between 65ms to 550ms (different spreading factors), the total time the module takes to: turn on, configure, transmit one frame, receive the Ack and save the configuration is about 4,5 to 6 seconds.

This means that with the disposition commented, the node holds 6 seconds for each transmission. We advance that this caused problems to the node's reliability because we want the information to be accurate and in time. But, the more transmissions causes worst performance, and for the opposite, the less transmissions makes the data shown be more delayed.

This problem is farther commented on the two last chapters.

## 4.7 Special Cases

In this section we provide some cases that are peculiar as they test our system in its limits. What happens when a bus passes, and a three-axis bus or if two cars pass at the same time.

### 4.7.1 Bus

The pressure system is sensible to the mass of the vehicles and the width of the wheels. The buses are large mass objects with wheel width grater than the cars. This means that the system detects a big pressure during an amount of time bigger than the car. Obviously, both comparators of `V-CAR` and `V-MOTO` are triggered on the pressed tube. To sum up, when the bus passes the node detects it like it was a car.

### 4.7.2 Car/Bus crossing transversally

When one or both axis of the car passes through the tubes non-perpendicularly it causes an *echo* in the signal that receives the sensor. This means that the edges produced can be seen duplicated.

When is this can happen? When the sensor is deployed in a curve or after one.

There are two solutions planned for this problem:

- Put the tubes perpendicularly to the actual direction of the vehicles.
- Increase the parameter `MAX_MICROS_TO_PASS_AXIS` by a factor 2 or 4 depending on the angle of attack of the vehicles.

### 4.7.3 Three-axis Bus

Up to this point we have consider a vehicles with two axis like almost every car or motorcycle. But, it happens that there are a bus station where three-axis buses stop.

Figure 4.27: Instance of a bus with 3 axis: Excelsior model XB60. If a vehicle has more than two axis it won't fall within our detection mechanism. It will detect one car and a half.

Furthermore, there can be cases where trucks or other vehicles with more than two axis pass by.

As we have said in the paragraph above, the system will detect one car for the first two axis and half axis by the rear axis. This means that there is a pending half axis. It will only count for one extra vehicle if other vehicles with three axis passes just in a row with the first one. Then, in that only case the system counts for one extra vehicle. But if the vehicles are sepparated more than `MAX_MILLIS_BETWEEN_HALF_VEHICLE` [3 seconds] the system will gauge the two 3-axis vehicles as two.

# Chapter 5

# Deployment

## 5.1 Product isolation and power supply

The nodes are outdoors and consequently, they will be through rough environmental conditions such as dust, rain, wind, sun and temperatures above 50 degrees Celsius. Keeping the devices isolated from such difficult environmental conditions is extremely important for the well-function of the nodes.

The nodes are in IP57 outdoors boxes to isolate them from the dust and rain and are fifty centimetres above the floor to protect them of ground-level water. Figures 5.1 and 5.3.

In figure 5.2, the power source is 230 V AC current, we include a transformer in the box to supply 7V to the nodes. The pressure shield of the node is connected to the pressure tubes laid in the asphalt as it is shown in figure 5.4.

## 5.2 Tube disposition

In figure 5.5, a car is passing over the tubes. The vehicles, in parking locations, usually do not follow strictly the ways draw drawn on the path.

We found that the disposition of the devices on the edge of the road (as it was designed 3.4) was not helpful with vehicle detection. The vehicles were not passing perpendicular and that caused errors on the detection. Although they could be solved by increasing the parameter `MAX_MICROS_TO_PASS_AXIS`, we relocated the nodes to the center of the entrances where the vehicles passed perpendicular. The final location of the nodes is presented in the map of figure 5.6.

In figure 5.7 there are two images taken during the deployment. We fix the pressure tubes on the road with screws. The screws have a dull header where the tube is knotted. The screw is always outside the road where the vehicles pass. Nonetheless, the header is bended to be plain to the wheels.

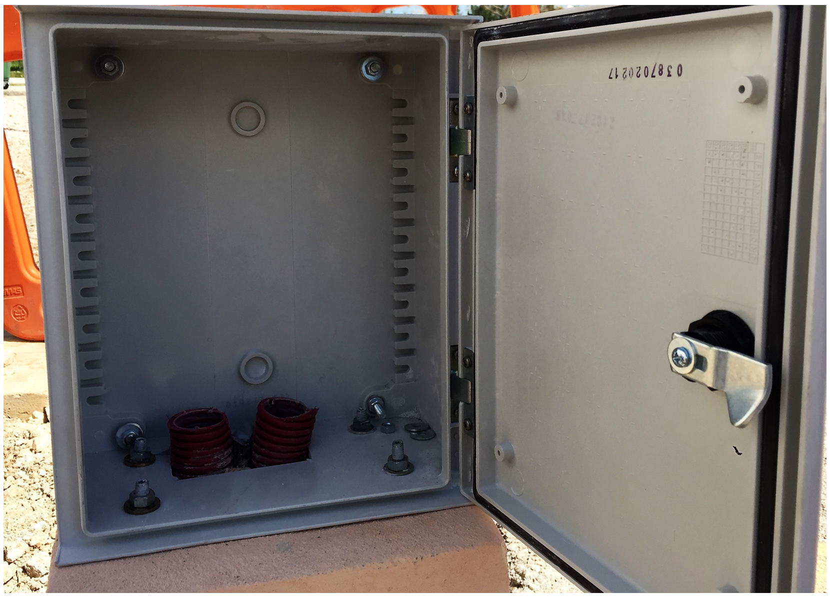


Figure 5.1: The power supply wires (left red tube) and the pressure tubes (left red tube) inside the IP57 box (E2).

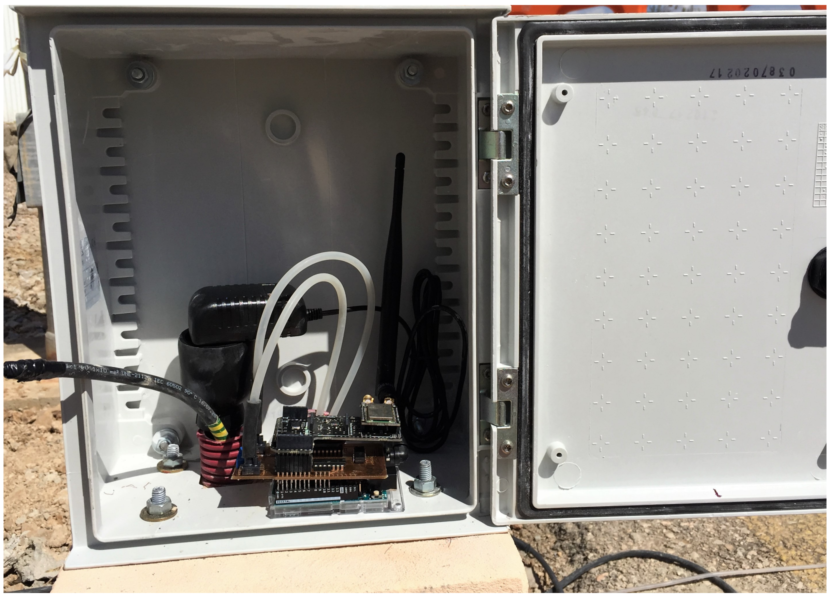


Figure 5.2: The node (S2) is connected to the power source and the pressure tubes.

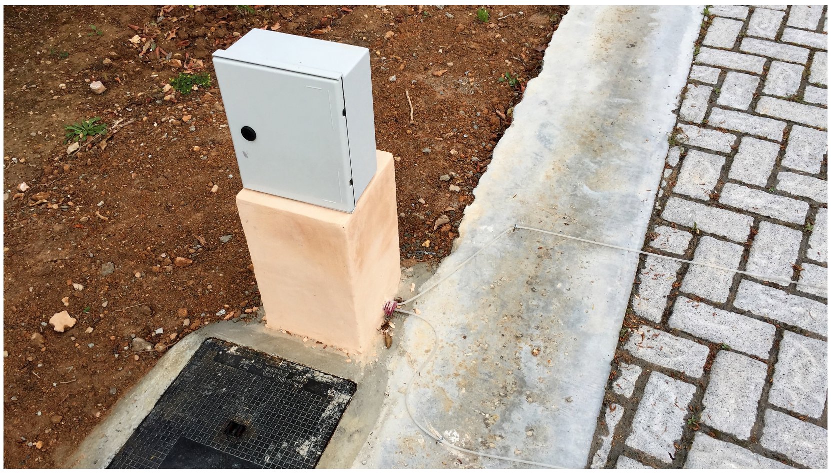


Figure 5.3: The node is isolated from the environment.

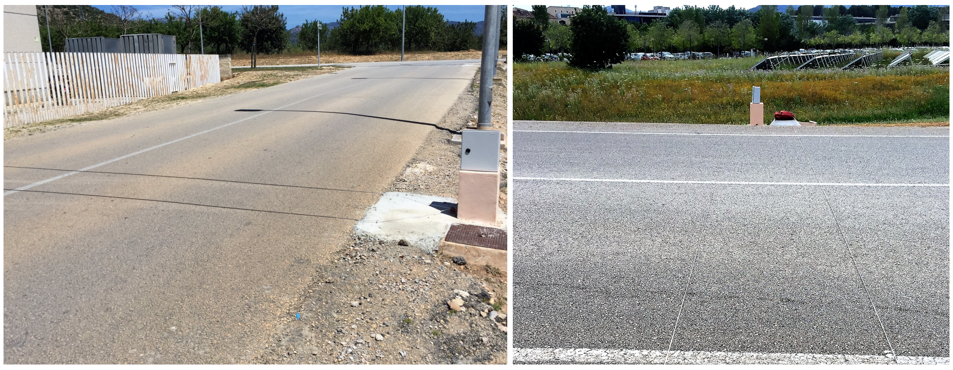


Figure 5.4: Node S4 in E2 (right) and node S7 in E4 (ledft). The tubes are laid perpendicular to the road's direction.

We also found that the distance between the tubes to led a proper detection can not be arbitrary. The distance has to be enough small that the most little vehicle can not put the two axis on the two tubes at the same time. The smallest vehicles are the motorcycles, the minimum distance between axis among motorcycles is about 1.20 m. **For this reason the distance used varies from 0.6 to 0.9 meters**. By pressing the two tubes at the same time they do not create errors into the detection algorithm but, it is better that the interruptions are separated in time.

## 5.3 Display communication

The displays are in the traffic signals that were already. The traffic signals are in entrances E1, E2, E3 and E4. These traffic signals are located in zones where no power nor electric tube arrived. In order to supply energy, a via to each traffic signal was made. The device is connected to the display with a pair of wires that are passed underground to the display, figure 5.9.

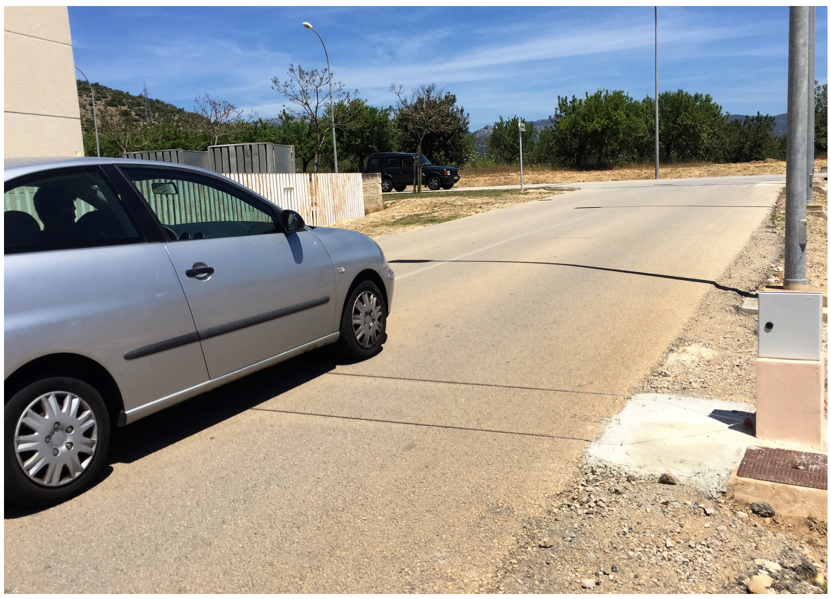


Figure 5.5: The vehicles pass over the tubes disposed on the asphalt.

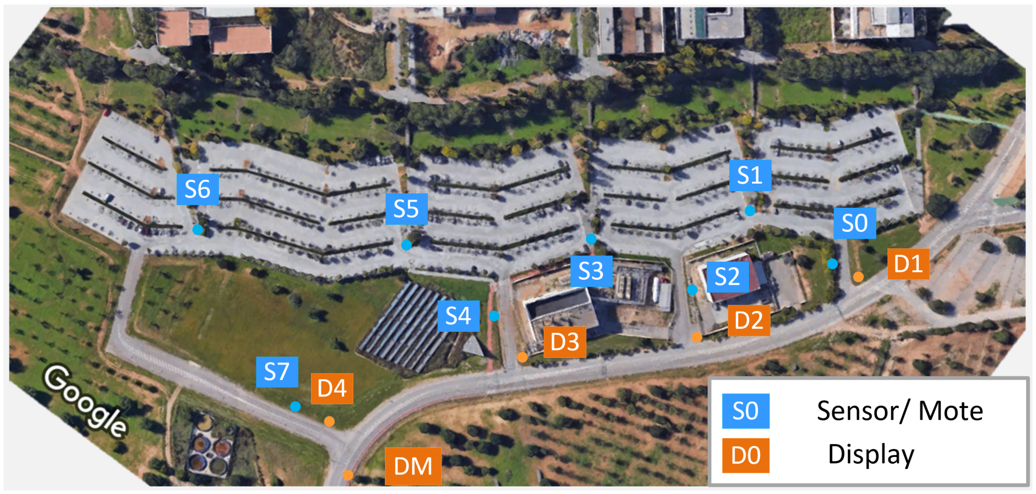


Figure 5.6: Final location of the devices. The connection from the motes to the displays are all underground. Used image from Google maps.

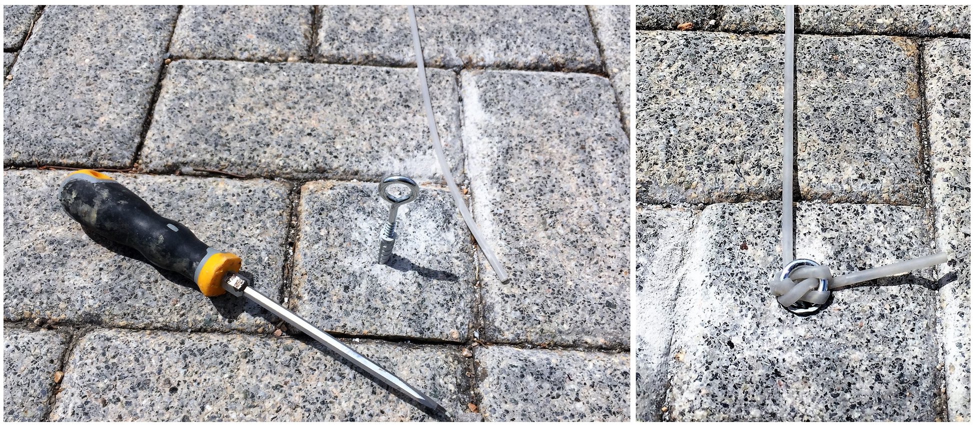


Figure 5.7: After doing a hole in the pavement, expansion anchor and the screw are inserted. Screw inserted and tube fixed.

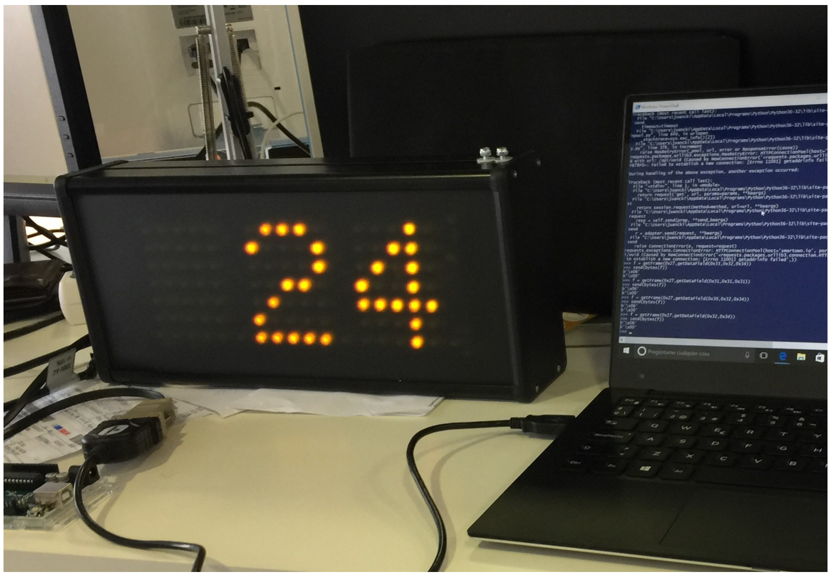


Figure 5.8: Display connected to the laptop's USB via a transceiver USB to RS232. The python script made controls serially the screen.

In figure 5.12, the display with 5 lines (DM) is located in the S/ Maria Agnesy before the entrance of P5.

We had to update the firmware of all the screens because, by default when all the fields were blank the result was red zero, which made the feature of turning off the leds impossible. After updating, the displays could turn off. In E1, we can see the result of turning a screen off 5.11.

The display of the node S7, figure 5.10, has the traffic signals of both zones P4 and P5. The number displayed on this screen is the addition of both zones.

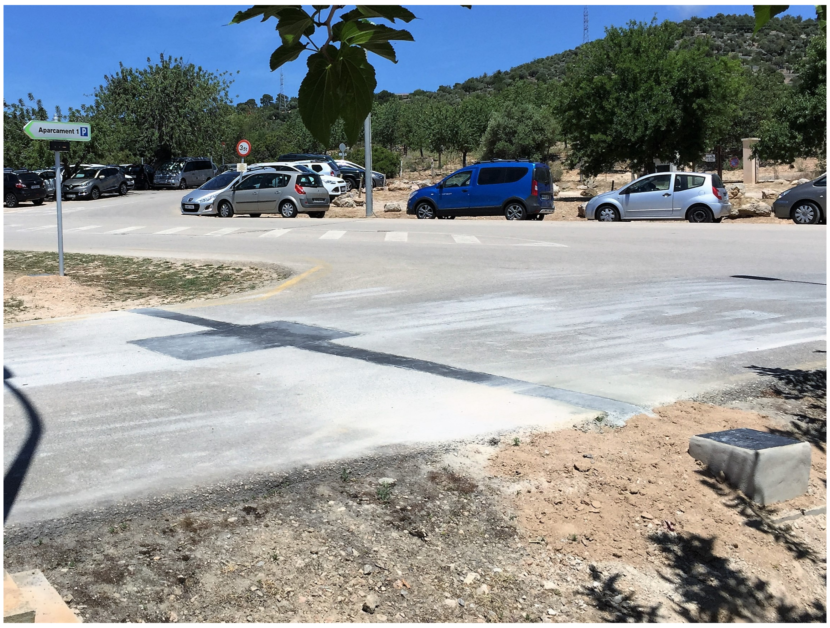


Figure 5.9: The communication and power wires are passed underground to the display.

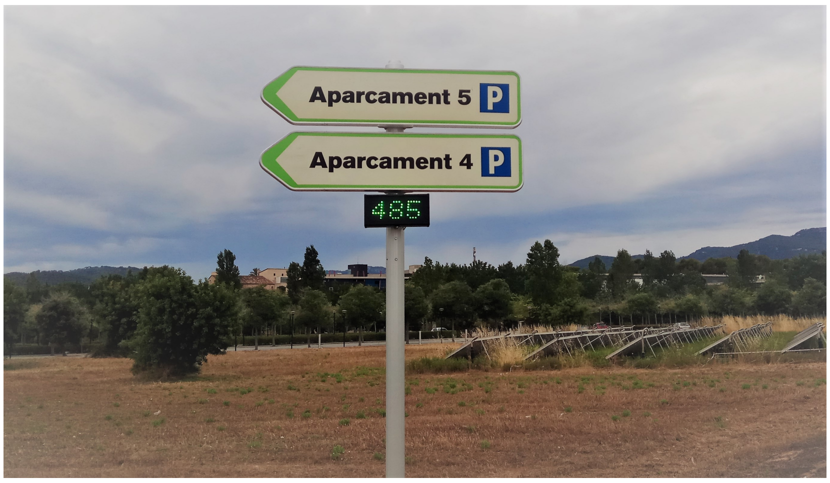


Figure 5.10: Display at E4 present the added value of the parking zones P4 and P5.

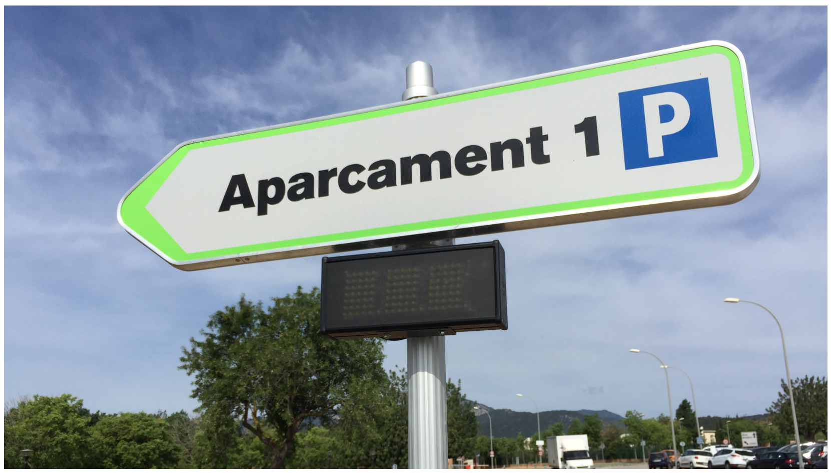


Figure 5.11: Display at E1 is turned off.

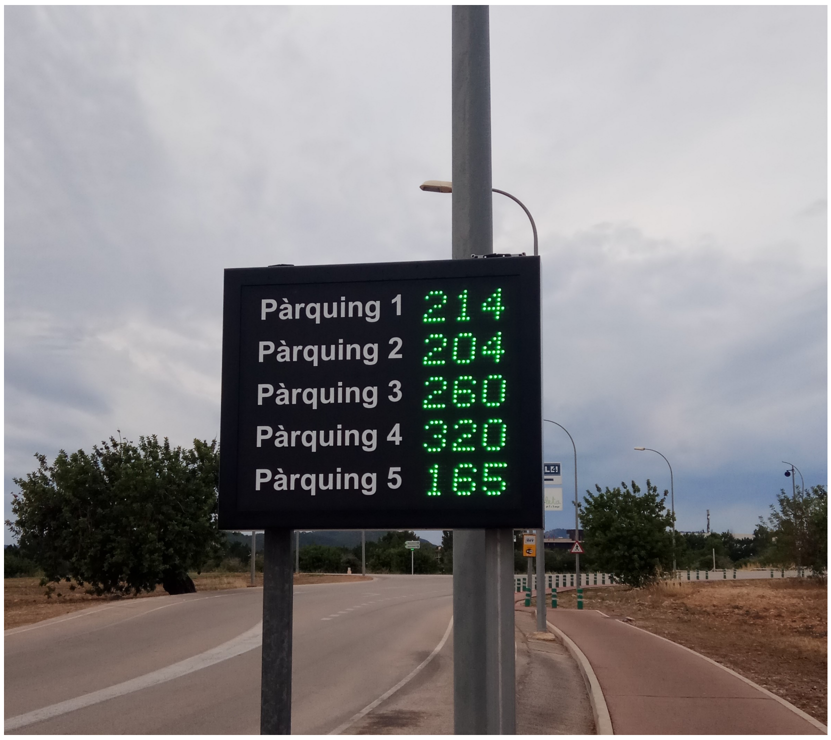


Figure 5.12: 5-line display showing the quantity of free spots in every parking zone.

## 5.4 Use of WDT

As it is said at the ending of the previous chapter, the project run onto random halts due to the lack of memory on the microcontroller. A preventive solution for halting was the use of WDT to reboot the system. After solving the problem as it is commented, we found useful to maintain the WDT configuration as the devices will be harvesting data on the field. At time of writing, we found no evidence on the devices of halting nor rebooting because of WDT.

We also modified the libraries for the transmit module RN2483 to include resets of the WDT to ensure every path of the code is save when enabling the WDT.

CHAPTER 6

# Results

## 6.1 Monitor Devices

### 6.1.1 Pressure Shield

We used the rubric presented in figure 6.1 to measure the performance of the shield. The negative timestamps marked on the left column are to use them when the performance is checked on the server side, to know that every frame of process has been sent. The same is for the minutes over 30 to confirm that every car counted before the minute 30 is sent to the server side.

The measure of the performance routine is the following:

1. Turn on the node and check that the tubes and every connection is correct.

2. The device starts automatically to detect vehicles.

Optional Turn on/connect the device to catch the debug serial output.

3. With the rubric disposed, start annotating the type of vehicle that are passing by.

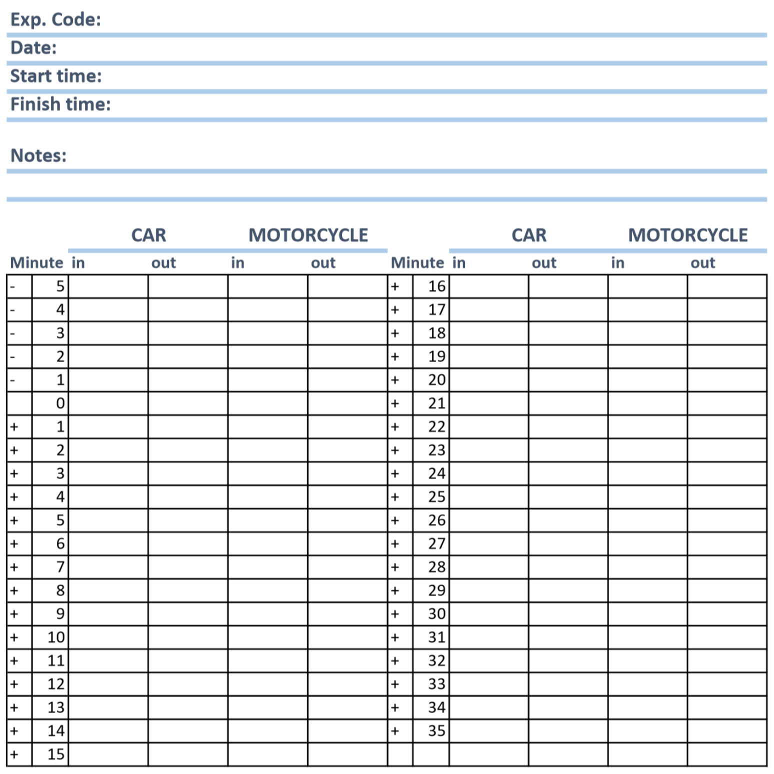
4. During the following 40 min annotate anomalies observed in the sheet.

5. Once the time has passed, proceed to calculate the performance of the node during the half hour of between.

6. The reliability is counted as:

$$R = 1 - \text{Probability}_{\text{failure}} = 1 - \frac{N_{\text{failed}}}{N_{\text{total}}}$$

We annotate in the rubric the values



Exp. Code:

Date:

Start time:

Finish time:

Notes:

| | | CAR | | MOTORCYCLE | | | | CAR | | MOTORCYCLE | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Minute | | in | out | in | out | Minute | | in | out | in | out |
| - | 5 | | | | | + | 16 | | | | |
| - | 4 | | | | | + | 17 | | | | |
| - | 3 | | | | | + | 18 | | | | |
| - | 2 | | | | | + | 19 | | | | |
| - | 1 | | | | | + | 20 | | | | |
| | 0 | | | | | + | 21 | | | | |
| + | 1 | | | | | + | 22 | | | | |
| + | 2 | | | | | + | 23 | | | | |
| + | 3 | | | | | + | 24 | | | | |
| + | 4 | | | | | + | 25 | | | | |
| + | 5 | | | | | + | 26 | | | | |
| + | 6 | | | | | + | 27 | | | | |
| + | 7 | | | | | + | 28 | | | | |
| + | 8 | | | | | + | 29 | | | | |
| + | 9 | | | | | + | 30 | | | | |
| + | 10 | | | | | + | 31 | | | | |
| + | 11 | | | | | + | 32 | | | | |
| + | 12 | | | | | + | 33 | | | | |
| + | 13 | | | | | + | 34 | | | | |
| + | 14 | | | | | + | 35 | | | | |
| + | 15 | | | | | | | | | | |

Figure 6.1: Rubric to measure the performance of pressure shield + detection algorithm. Then this sheet is compared to the actually detected cars. The total time measuring is 40 minutes.

### Only detection: Microcontroller + Pressure Shield

The number of detected vehicles by the shield is observed using the serial output of the Arduino in a laptop. The actual numbers of vehicles is measured with the rubric.

We did this process 5 times at different hours on the field, the results show an **average reliability of the 98.4%** with the parameters are configured as it is said in the document. It is noticeable that the failures came from detecting one or more car axis as motorcycle's.

### Full Node: Microcontroller + Pressure Shield + TX Module + Display

Similarly to the measures of the latest paragraph, we count the actual number of each vehicle with the rubric, but in this case the detected vehicles is seen in the gateway's log.

This was done 5 times. With this configuration **the performance decreased substantially to the 94%**. We configured the `Time Threshold` to two minutes and the `Flow Threshold` to six vehicles.

**The lack of accuracy measuring is not suitable for describing the use of the parking, for that reason in last chapter we present the current evolution of the device to address this issue.**

### 6.1.2 LoRaWAN

We will use the gateway's logging file to represent the data received. This file has a series of parameters to guide the reader.

How to interpret Kerlink's log of figures 6.2, 6.4 and 6.5.

**Id** Kerlink internal frame counter. One for each frame received.

**Time** Express data and time when the frame is received by the antenna.

**Seq** Sequence number: Represents the uplink frame counter of the LoRaWAN transmitter. If it is repeated means that there was a retransmission.

**Port** Application port, the port zero is reserved for LoRaWAN Commands and we use the third port for the parking service.

**Radio** Antenna which received the frame. This gateway is omnidirectional

**Channel** Band of 125KHz that was used.

**SNR** Signal noise ratio, LoRa sensibility reach up to -20 dB.

**RSSI** Signal strength received in dBm.

**Modulation** LoRa or GFSK.

**Datarate** closely related to Spreading Factor (SF) that ranges from 7 (best datarate, worst SNR sensibility) to 12 (best SNR sensibility, worst datarate) and BW refers to the bandwidth used in KHz.

**CR** Coding rate of the payload of the frame, it can vary from 4/5 to 8/5.

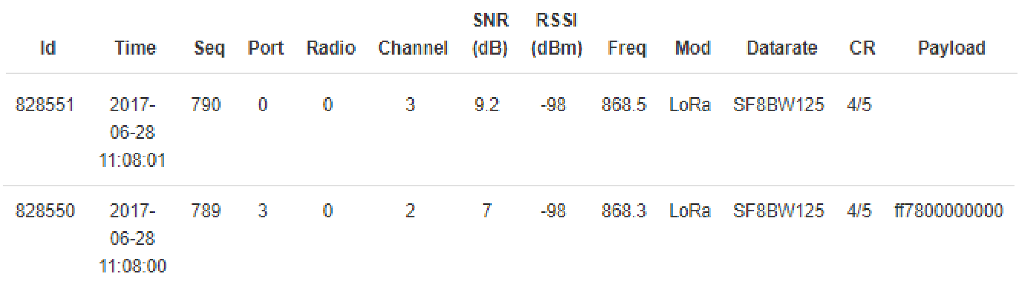
**Payload** Information sent by mote represented in a hexadecimal char string.

#### LoRaWAN Confirmed Message: Acknowledgements

The message performed to the gateway are confirmed to ensure that the information is equal on both sides. In figure 6.2 there is a real example of S7 frame log data on the gateway. The image shows the gateway's log for received messages. The gateway sends periodically the messages received and status to the server.

#### LoRa SNR

The nodes S0-S7 and DM are situated inside a IP57 box. From the box to the gateway there is line of sight therefore, the SNR is very high. As we can see from the figures and the information in the gateway, the SNR varies from 4dB to 10dB which is a highly good signal relation as LoRa modulation can work with up to -20dB with SF12.



| Id | Time | Seq | Port | Radio | Channel | SNR (dB) | RSSI (dBm) | Freq | Mod | Datarate | CR | Payload |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 828551 | 2017-06-28 11:08:01 | 790 | 0 | 0 | 3 | 9.2 | -98 | 868.5 | LoRa | SF8BW125 | 4/5 | |
| 828550 | 2017-06-28 11:08:00 | 789 | 3 | 0 | 2 | 7 | -98 | 868.3 | LoRa | SF8BW125 | 4/5 | ff7800000000 |

Figure 6.2: The Ack is shown as a blank payload with port 0. The node sent a message at 11:08:00 and then an ACK in the next second. A confirmed downlink was performed successfully.

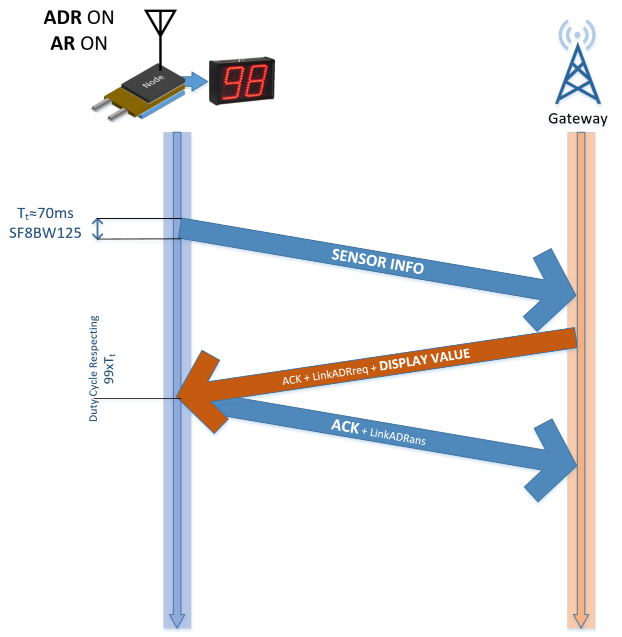
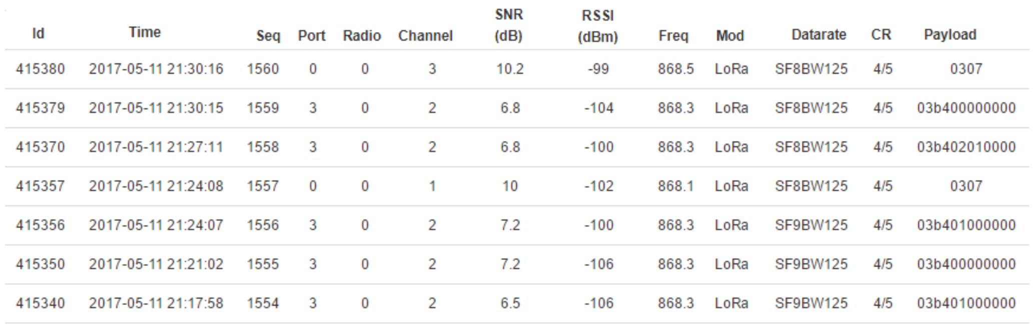


Figure 6.3: The commands ADRreq and ADRans are piggybacked with user payload.



| Id | Time | Seq | Port | Radio | Channel | SNR (dB) | RSSI (dBm) | Freq | Mod | Datarate | CR | Payload |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 415380 | 2017-05-11 21:30:16 | 1560 | 0 | 0 | 3 | 10.2 | -99 | 868.5 | LoRa | SF8BW125 | 4/5 | 0307 |
| 415379 | 2017-05-11 21:30:15 | 1559 | 3 | 0 | 2 | 6.8 | -104 | 868.3 | LoRa | SF8BW125 | 4/5 | 03b400000000 |
| 415370 | 2017-05-11 21:27:11 | 1558 | 3 | 0 | 2 | 6.8 | -100 | 868.3 | LoRa | SF8BW125 | 4/5 | 03b402010000 |
| 415357 | 2017-05-11 21:24:08 | 1557 | 0 | 0 | 1 | 10 | -102 | 868.1 | LoRa | SF8BW125 | 4/5 | 0307 |
| 415356 | 2017-05-11 21:24:07 | 1556 | 3 | 0 | 2 | 7.2 | -100 | 868.3 | LoRa | SF9BW125 | 4/5 | 03b401000000 |
| 415350 | 2017-05-11 21:21:02 | 1555 | 3 | 0 | 2 | 7.2 | -106 | 868.3 | LoRa | SF9BW125 | 4/5 | 03b400000000 |
| 415340 | 2017-05-11 21:17:58 | 1554 | 3 | 0 | 2 | 6.5 | -106 | 868.3 | LoRa | SF9BW125 | 4/5 | 03b401000000 |

Figure 6.4: The ADR messages are piggy backed with user data, with AR on the ADR message are sent to the gateway to confirm ADR request without user payload. The ADR goes through the port 0. If the request was successfully performed the payload is 0307.

### MAC Command: ADR

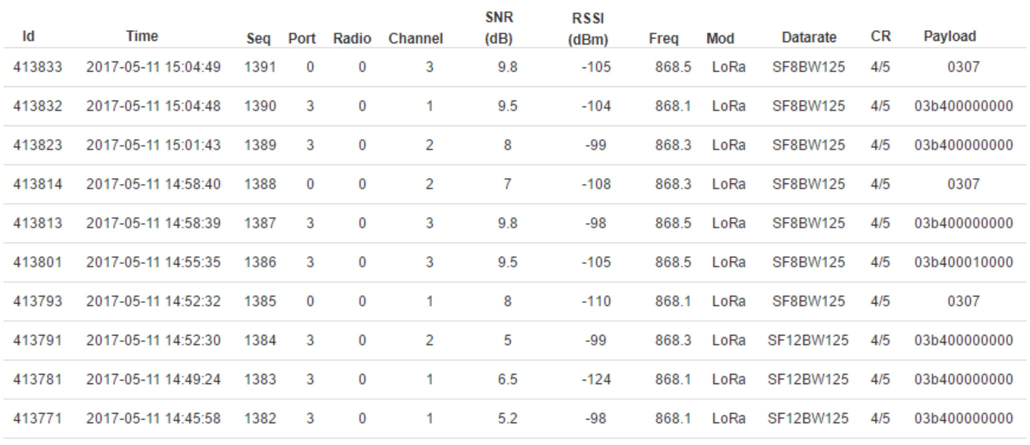
Automatic Data Rate is a LoRaWAN feature to control the Datarate, Transmit power and bands usable. We can see in figure 6.4 the Kerlink's log with the responses to ADR requests sent by the gateway. In figure 6.3, there is a representation of the flow of the messages regarding to the nodes with display.

- The first one (below) with port 3 and payload 0x03b400000
- The second one with port number 0 and payload 0307 that is a LinkADRans.
  - The first byte 0x03 is the CID (*command id*) 3 that refers to ADR request/answer LoRaWAN MAC commands.
  - The second byte 0x07 refers to the changed parameters, there are only three:
    - 000001XX The channel mask was successfully interpreted. All currently defined channel states were set according to the mask.
    - 00000X1X The data rate was successfully set.
    - 00000XX1 Transmit power was successfully set.

### Downlinks

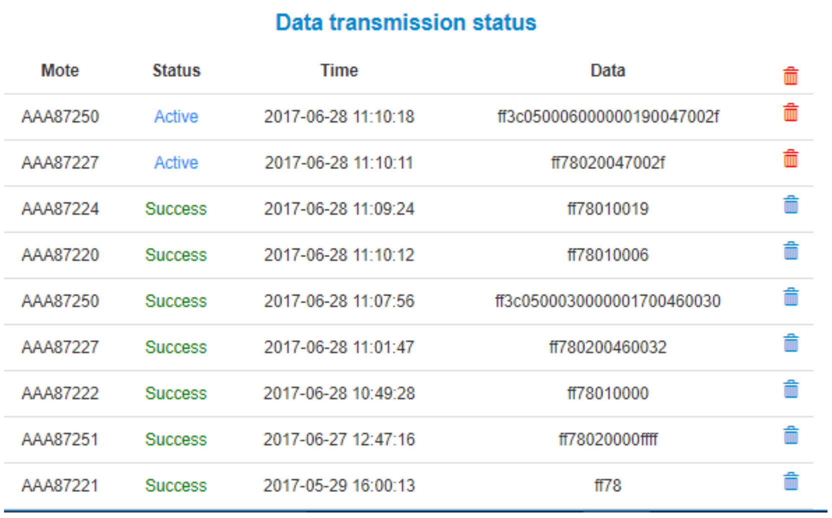
The messages from the sever are sent to the correspondent gateway. The gateway is in charge of transmitting the downlinks to the nodes. The gateway can only send downlink to the Class A LoRaWAN devices when receives a message from the node. The message from the node can be either confirmed or unconfirmed. In both cases, after the transmission, the node hears during two seconds to the radio to detect incoming messages.

The configuration of the gateway's downlink scheme is not optimal for the performance of our system because it can only transmit downlinks to the node if the server has sent it before. This means that when the node receives a downlink it is not generated by that very moment but from the past messages from the server. The downlinks because



| Id | Time | Seq | Port | Radio | Channel | SNR (dB) | RSSI (dBm) | Freq | Mod | Datarate | CR | Payload |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 413833 | 2017-05-11 15:04:49 | 1391 | 0 | 0 | 3 | 9.8 | -105 | 868.5 | LoRa | SF8BW125 | 4/5 | 0307 |
| 413832 | 2017-05-11 15:04:48 | 1390 | 3 | 0 | 1 | 9.5 | -104 | 868.1 | LoRa | SF8BW125 | 4/5 | 03b400000000 |
| 413823 | 2017-05-11 15:01:43 | 1389 | 3 | 0 | 2 | 8 | -99 | 868.3 | LoRa | SF8BW125 | 4/5 | 03b400000000 |
| 413814 | 2017-05-11 14:58:40 | 1388 | 0 | 0 | 2 | 7 | -108 | 868.3 | LoRa | SF8BW125 | 4/5 | 0307 |
| 413813 | 2017-05-11 14:58:39 | 1387 | 3 | 0 | 3 | 9.8 | -98 | 868.5 | LoRa | SF8BW125 | 4/5 | 03b400000000 |
| 413801 | 2017-05-11 14:55:35 | 1386 | 3 | 0 | 3 | 9.5 | -105 | 868.5 | LoRa | SF8BW125 | 4/5 | 03b400010000 |
| 413793 | 2017-05-11 14:52:32 | 1385 | 0 | 0 | 1 | 8 | -110 | 868.1 | LoRa | SF8BW125 | 4/5 | 0307 |
| 413791 | 2017-05-11 14:52:30 | 1384 | 3 | 0 | 2 | 5 | -99 | 868.3 | LoRa | SF12BW125 | 4/5 | 03b400000000 |
| 413781 | 2017-05-11 14:49:24 | 1383 | 3 | 0 | 1 | 6.5 | -124 | 868.1 | LoRa | SF12BW125 | 4/5 | 03b400000000 |
| 413771 | 2017-05-11 14:45:58 | 1382 | 3 | 0 | 1 | 5.2 | -98 | 868.1 | LoRa | SF12BW125 | 4/5 | 03b400000000 |

Figure 6.5: The transmit module expands the SF (Datarate column) to send the message in the least time possible. SF12 is slower than SF8. The messages are usually sent with the minimum coding on the payload 4/5.



Data transmission status

| Mote | Status | Time | Data |
|---|---|---|---|
| AAA87250 | Active | 2017-06-28 11:10:18 | ff3c050006000000190047002f |
| AAA87227 | Active | 2017-06-28 11:10:11 | ff78020047002f |
| AAA87224 | Success | 2017-06-28 11:09:24 | ff78010019 |
| AAA87220 | Success | 2017-06-28 11:10:12 | ff78010006 |
| AAA87250 | Success | 2017-06-28 11:07:56 | ff3c0500030000001700460030 |
| AAA87227 | Success | 2017-06-28 11:01:47 | ff780200460032 |
| AAA87222 | Success | 2017-06-28 10:49:28 | ff78010000 |
| AAA87251 | Success | 2017-06-27 12:47:16 | ff78020000ffff |
| AAA87221 | Success | 2017-05-29 16:00:13 | ff78 |

Figure 6.6: Kerlink's Downlink Queue

of that are at best 1 min before in time. Then, the freshness of the information showed to the users can vary from 1 to 4 minutes into the past depending on the casuistic of the transmissions.

### 6.1.3 Use of Spectrum: Metric for LPWAN

The ALOHA-like LoRaWAN approach for medium access simplifies the operations and consumption for nodes, but with network efficiency as its price [20]. We introduce a metric of use of network of this system $U_{sys}$ to its maximum load acceptance point. And particularly we calculate the use in the same terms for the **n** node $U_n$ or the average for this system $\overline{U}_{n|sys}$.

Table 6.1: LoRa physical data rate relationship with the SF with BW=125KHz. Data from [20].

| SF | Preamble duration [ms] | Preamble duration [bits] | PHY bit rate [bps] |
|---|---|---|---|
| 12 | 401.41 | 100.1 | 250 |
| 11 | 200.70 | 88.3 | 440 |
| 10 | 100.35 | 98.3 | 980 |
| 9 | 50.18 | 88.3 | 1760 |
| 8 | 25.05 | 78.3 | 3125 |
| 7 | 12.54 | 68.6 | 5470 |
| $\sum$ | - | - | 12025 |

As we have said the LoRa modulation has 6 different SF in each band. The SF are multiple access orthogonal channels. We have set the 3 EU-mandatory bands of 125KHz: 868.1, 868.3 and 868.5 MHz.

As we can see in table 6.1, the total capacity of a 125KHz band is 12.025Kbps, because the 6 different SF are orthogonal.

With the devices deployed, we have measured the use by the information received on the gateway. This is data is presented in the `csv` file in `exp/rxdata.csv`. In this file all the received transmissions between the '2017-06-27 13:19:37' and the '2017-06-28 11:11:17'.

### Parking devices S0-S7

On the first hand, we have the sensors S0-S7 that capture information about the flow in the whole parking. In figure 6.7 it is shown the area covered with the sensors (S0-S7). The area is 65218m$^2$ and its perimeter is 1251m.

The results for the 8 traffic detectors, show:

**Successful TX** 7623.

- S0 1019 transmissions.
- S1 881 transmissions.
- S2 1047 transmissions.
- S3 885 transmissions.
- S4 **1066 transmissions.**
- S5 881 transmissions.
- S6 892 transmissions.
- S7 952 transmissions.

**Time** 21 hours 50 minutes and 40 seconds between the last and first transmission. Which in average implies 43.59 frames per hour per device, or one transmission for each minute and 23 seconds per device.

**SF used** 7511 transmissions with SF8 and 112 with SF7.

**Coding rate** 4/5.

**Bandwidth** 125KHz.

**Symbols on preamble** 8 symbols.

**MAC Payload length** 6 bytes.

**CRC length** 2 bytes.

**Total frame** (without preamble,no opts) 30 bytes.

**Transmit time per frame** (with preamble):

- with SF8: $30 \cdot 8 + 78.3 = 318.3$ bits = 101.86ms.
- with SF7: $30 \cdot 8 + 68.6 = 308.6$ bits = 56.42ms

The total useful physical bits (not MAC payload bits) introduced during 22 hours is $Bits = N_{SF8} \cdot 318.3 + N_{SF7} \cdot 308.6$

$$= 7511 \cdot 318,3 + 112 \cdot 308.6 = 2390751.1 + 34563.2 = 2.453 \cdot 10^6 \text{bits}$$

Supposing a 2-fold factor for the whole transmission procedure with acknowledged messages, the usage of physical bit rate is $\frac{2 \cdot 2.453 \cdot 10^6 \text{bits}}{78700 \text{seconds}} = 61.63$bps.

- The total use of the medium (3 bands) per sensing system is $\overline{U}_{sys} = \frac{61.63}{36075} = 1.701 \cdot 10^{-3} = 0.17\%$.
- The use per sensor of medium (3 bands) is $\overline{U}_{S_n|sys} = \frac{\overline{U}_{sys}}{N_{\text{motes}}} = \frac{1.701 \cdot 10^{-3}}{8} = 0.0213\%$.
- The average of sensors per 10000m$^2$ is = 1.23 motes per $10^4 m^2$.
- The average of use per 10000m$^2$ is = $2.6266 \cdot 10^{-4} = 0.026\%$.

**Big display DM**

On the other hand, we have also the DM mote which displays the information of the parking that also uses the network. Then the system globally S0-S7 and DM covers the area is 71854 m$^2$ and its perimeter is 1374m.

The results for the 5-line display :

**Successful TX** 1699.

**Time** 21 hours 50 minutes and 45 seconds between the last and first transmission.

**SF used** 1663 transmissions with SF8 and 36 with SF7.

**Coding rate** 4/5.

**Bandwidth** 125KHz.

**Symbols on preamble** 8 symbols.

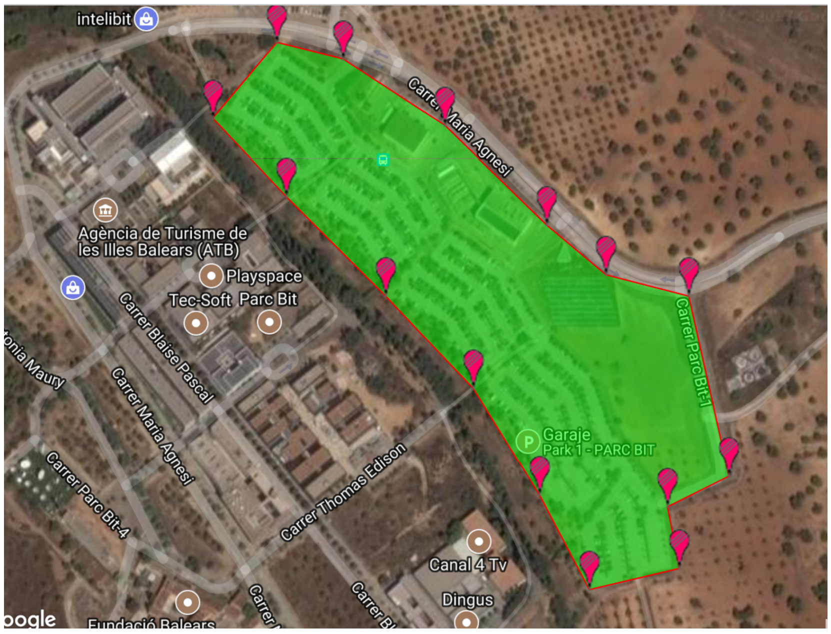


Figure 6.7: Measuring area by the sensors S0-S7.

**MAC Payload length** 6 bytes.

**CRC length** 2 bytes.

**Total frame** (without preamble,no opts) 26 bytes.

**Transmit time per frame** (with preamble):

- with SF8: $26 \cdot 8 + 78.3 = 286.3$ bits = 101.86ms.
- with SF7: $26 \cdot 8 + 68.6 = 276.6$ bits = 56.42ms

With the same assumption as in the paragraph before, the usage of the physical bit rate is $\frac{2 \cdot (1663 \cdot 286.3 + 36 \cdot 276.6)\text{bits}}{78700\text{seconds}} = 6.176\text{bps}$.

- The use per big display of the medium (3 bands) is $\overline{U}_{DM} = \frac{6.176\text{bps}}{36075} = 1.712 \cdot 10^{-4} = 0.017\%$.

## 6.2 Documentation Included

With the present document we attach the following files distributed as follows:

- code
  - node
    - node.ino
  - bigDisplay
    - bigDisplay.ino

- libraries
  - PressureShieldDriver.zip
  - EepromUtils.zip
  - DisplaySoftSerialUtils.zip
  - arduinoApi.zip
  - arduinoLoRaWAN.zip
- librariesNoZIP
  - PressureShieldDriver
    - PressureShieldDriver.h
    - PressureShieldDriver.cpp
  - EepromUtils
    - EepromUtils.h
    - EepromUtils.cpp
  - DisplaySoftSerialUtils
    - DisplaySoftSerialUtils.h
    - DisplaySoftSerialUtils.cpp
  - arduinoApi
    - arduinoClasses.h
    - arduinoMultiprotocol.h
    - arduinoMultiprotocol.cpp
    - arduinoUART.h
    - arduinoUART.cpp
    - arduinoUtils.h
    - arduinoUtils.cpp
  - arduinoLoRaWAN
    - arduinoLoRaWAN.h
    - arduinoLoRaWAN.h
    - examples[...]
- PressureShield
  - pssrShield.zip
- exp
  - exp_results.xlsx
  - car1 [...]
  - car2 [...]
  - moto [...]
- memory.pdf

## 6.3 How to use: Compile the C++ Code for Arduino

To use the code implemented, the user has to follow the list of instructions for installation:

1. Install the Arduino IDE version 1.8.2 from arduino.org [18]. At date of writing there is the 1.8.3 version which also works.

2. Include the `.zip` of libraries located in the `libraries` folder. To include go to the tab `Program`, click on `Include Library`, `Add Library.ZIP` and then select one by one the libraries. Apart from our libraries `PressureShieldDriver`, `DisplaySoftSerialUtils` and `EepromUtils` we strongly recommend including the libraries we modified for the transmit module: `arduinoApi` and `arduinoLoRaWAN`.

3. Then, open the `node.ino` file located in the `code/node` folder with the Arduino IDE.

4. Once the code is on the screen, verify the LoRaWAN keys and Device address of the device.

5. Select the USB port in the `Tools`' tab and `Port` section.

6. Finally click on the upload button.

CHAPTER 7

# Conclusions

## 7.1 Project Evaluation

The developing process needs from investing time analysing the advantages and inconveniences of each part and the system as a whole to optimize the outcome for the client and the user. In this chapter, the highlights of the developed nodes and the challenges to overcome are presented.

### 7.1.1 Pros

The advantages:

- Easy installation.
- No wired communications required.
- Low-cost, light weight and discrete.
- Remote control for granular accuracy.
- Low-consume, possibility of design an autonomous system with a low power solar panel.

### 7.1.2 Cons

The counterparts:

- The tubes used wear away and break with the use.
- It does not have double check mechanism.
- It needs more development to become industrialized.
- The detection mechanism implemented perse lacks of high accuracy, fails 1.6%.

### 7.1.3 Future Work

Throughout this document some weaknesses in the initial design have been found. Further development has been carried out to address this issues. The following topics are currently under development:

- Find other solution for pressure tubes. They have to maintain the flexibility pressure detection and improve their resilience to recover successfully after deformation.
- Improve the reliability of the node. When the nodes presented transmit data, they can not attend the events detected in the shield. This leads to a decrease of reliability to the 94%. In order to solve this problem, a microcontroller is added.

  Now, with two microcontrollers, one for the Multiprotocol Shield and Pressure Shield on the other, allow return to the accuracy to 98.4%. By adding other microcontroller it can transmit more often to arrive to 1 minute delay of our requirements.

It is also possible to study other ways of interconnecting the devices to explode the use of the medium:

- The nodes are disposed in a star-like network, the transmission are according to the LoRaWAN stack that only allows node-to-gateway and gateway-to-node transmissions. This means that the server needs to be triggered with every sent frame. It is currently under consideration moving this scheme to mesh-network with a central node that carries out the logic of the parking by communicating to the rest of nodes and then, update periodically the status of the whole parking to the server.

  This scheme has multiple pros. For instance: increasing the reuse of frequencies per area using much less transmit power, being able to function when the gateway or the server are down and reduce the access to the database and reduce CPU time in the server side. Nonetheless, the development of autonomous nodes on the field performing a mesh network is known to be difficult to implement as in general the reliability and sensibility of the nodes are fairly worse than in BSs. New transmit shields could be the trigger to evolve to a LoRa mesh network.

#### Next Shield

If there was a soar in the demand of this product, the production method will have to be industrialized. Furthermore, we visualize the industrialized version of this product to be a unique shield composed of all the parts mentioned: pressure sensor, two microcontrollers and a LoRaWAN module. Also, it would have to have a socket for a TIA-485 bus. It would be necessary to decide whether to use a battery or not just in case of power failure.

### 7.1.4 Other Applications

The node developed could be interesting for other kind of situations. The key points of this project makes it suitable for:

- Measuring traffic flow during special events in towns. This would allow to know which affluence had the town.
- Present data remotely with a led screen similar to those presented in this document with low update rate.

## 7.2 Author's Personal View

This project has been highly challenging because all of its aspects required knowledge of very diverse disciplines. For instance, while debugging the node it had been necessary to test both parts, software and hardware.

During this project it has been possible to have further understanding of the Arduino. I have found that the OpenHardware and OpenSoftware platforms provide a softer learning curve than other similar systems with proprietary character.

For this project, almost every feature of the Arduino Uno Board is used. Features used of both software and hardware range from interrupts, I/O, UART, SoftwareSerial, Watchdog, different compilers to power supply and shields.

It has been an incredible opportunity to understand and develop with a novel LPWAN protocol such as LoRaWAN. It has been possible to learn how the topology of this network is useful for low-rate, low-cost nodes. Nevertheless, the use of LoRaWAN as a communication protocol for real-time services is discouraged.

In addition, it also showed me how difficult it is to develop any kind of product, no matter its complexity. Moreover, it has been demonstrated that the ability to envision, design and organise properly are essential skills.

# BIBLIOGRAPHY

[1] "Lora specification v1.0," January 2015. [Online]. Available: https://www.lora-alliance.org/portals/0/specs/LoRaWAN%20Specification%201R0.pdf 2.2.5, 2.2.5, 2.1

[2] "Videographic documentation of the emt ("empresa municipal de transport")." [Online]. Available: http://www.emtpalma.es/es/-/videos-informatius 2.1

[3] "Palma en bici web page." [Online]. Available: http://www.palmaenbici.com/ 2.2

[4] L. S. W. Michael J. Caruso, "Vehicle detection and compass applications using amr magnetic sensors." 2.3, 2.1.1

[5] "Datasheet of anisotropic magneto-resistive sensor." [Online]. Available: http://www.magnet.com.my/wp-content/uploads/2016/02/BRD01_Sprcification_RV1.pdf 2.4

[6] "Opencv, widely used open source software for computer vision." [Online]. Available: http://opencv.org/ 2.1.3

[7] "Overview of the internet of things," *ITU*, 2012. 2.2, 2.2.1

[8] "Towards a definition of the Internet of Things. Differences between IoT and WSN, page 72," *IEEE Internet of Things*, May 2015. 2.2, 2.2.1

[9] "In depth performance evaluation of lte-m for m2m communications," *IEEE, Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2016. 2.2.2

[10] "Standarization of nb-iot completed." [Online]. Available: http://www.3gpp.org/news-events/3gpp-news/1785-nb_iot_complete 2.2.3

[11] "Nb-iot system for m2m communication," *IEEE, Wireless Communications and Networking Conference (WCNC)*, 2016. 2.6

[12] "Sigfox web page," https://www.sigfox.com/, accessed: 2017-06-21. 2.2.4, 2.1

[13] S. Corporation, "Lora modem designer guide," 2013. [Online]. Available: http://www.semtech.com/images/datasheet/LoraDesignGuide_STD.pdf 2.7

[14] "Arduino uno rev.3." [Online]. Available: https://www.arduino.cc/en/Main/ArduinoBoardUno 2.8, 2.2, 2.3.3, 4.12, 4.2.3, 4.3

[15] "Waspmote, board that allow different hacks, lorawan radio module and shield for arduino." [Online]. Available: http://www.libelium.com/downloads/documentation/waspmote_technical_guide.pdf 2.9

[16] "Cooking hacks, lorawan radio module and shield for arduino." [Online]. Available: https://www.cooking-hacks.com/lorawan-radio-shield-for-arduino-868-mhz 2.2, 2.3.3

[17] "Parc bit's park map. accessed june 2017." [Online]. Available: http://www.parcbit.es/wparcbitfront/images/pdf/planoParque.pdf 3.1

[18] "Arduino ide 1.8.2, compiler." [Online]. Available: http://www.arduino.org/downloads 4.3, 1

[19] "Max 485, tia-485 bus driver and reader." [Online]. Available: https://datasheets.maximintegrated.com/en/ds/MAX1487-MAX491.pdf 4.22, 4.5.1

[20] "A Study of LoRa: Long Range & Low Power Networks for the Internet of Things," *MDPI http://www.mdpi.com/1424-8220/16/9/1466/htm*, May 2016. 6.1.3, 6.1