Faculdade de Engenharia da Universidade do Porto

Monitoring and Communication System Targeting a Pavement Energy Harvesting Application

Wilson Ismael Leite Cândido

MSc Dissertation

Supervisor: Prof. Doutor João Paulo Filipe de Sousa

Co-Supervisor: Doutor Francisco João Anastácio Duarte

July, 2019

wilson.ismael.candido@gmail.com

Monitoring and Communication System Targeting a Pavement Energy Harvesting Application

Wilson Ismael Leite Cândido

MSc Dissertation

July, 2019

Abstract

Pavnext is a startup company that develops technological pavements, and is currently developing a device that allows the extraction of kinetic energy from vehicles reducing their speed without driver action and without any side effect to the vehicle, promoting road safety more effectively in places where low speeds are required. The harvested energy is converted to electric energy with high efficiency and without any emission.

This dissertation documents the design, assembly and test of a monitoring and communication system capable of integrating the prototype of the above mentioned product, in order to perform data logging for supporting the development process and, once the final prototype is completed, will generate data regarding traffic and speed in real time, as well as from the generated and consumed energy. This data, once sent to the cloud, can be used to generate statistical studies and reports.

Several concept solutions are proposed, compared and discussed, and the most appropriate for this application is fully described.

The solution focus on the development of a modular system, distributed by the units placed on the pavement, in which characteristics such as the presence of changeable characteristics have a strong impact on the proposed design. Thus, a system based on general-purpose, versatile and low-cost components, which communicate through a long-distance I^2C bus, is proposed.

Without compromising the system's already defined structure from a mechanical and structural point of view, the implementation of the system allows the low-level layer of monitoring and communication system to open up opportunities to study the collected data, boosting sustained development.

Agradecimentos

Agradeço à Pavnext e na sua pessoa, ao Francisco, pela oportunidade de acompanhar de perto um projeto promissor e entusiasmante. Obrigado também ao Ricardo, ao Eduardo, ao João, ao David, ao Henrique, à Inês e ao Nuno pela fácil integração proporcionada.

Agradeço aos meus alunos pela paciência e compreensão durante este ano letivo. Pelo apoio e formação, um obrigado à Dra. Sandra, ao Professor Pedro Arantes e à Engenheira Ana Cadete.

Ao Professor João Sousa agradecer o esforço, a abordagem, o trato e a lucidez no aconselhamento para a tomada de decisões. Foi um prazer partilhar o espaço de aprendizagem durante os últimos anos.

Pelo apoio incondicional, um obrigado aos meus pais. Pela confiança cega, um obrigado ao meu irmão. Pelo ânimo e alegria contagiante, um obrigado à minha irmã.

Wilson Cândido

"The question is not what you look at, but what you see."

Henry David Thoreau

Contents

List of Figures

List of Tables

Abbreviations

ADC	Analog-to-Digital Converter
CDT	C/C++ Development Tooling
CPU	Central Processing Unit
GPIO	General Purpose Input/Output
I^2C	Inter-Integrated Circuit
ICs	Integrated circuits
IDE	Integrated Development Environment
Km/h	Kilometers per hour
OS	Operating System
PCB	Printed Circuit Board
RPEH	Road Pavement Energy Harvesting
RTOS	Real-Time Operating System
SA1	System Architecture 1
SA2	System Architecture 2
SA3	System Architecture 3
SCL	Serial Clock
SDA	Serial Data
SRE	Speed Reducer Equipment
TRL	Technology Readiness Level

Chapter 1

Introduction

1.1 Context and Motivation

Industry, non-governmental organizations and governmental agencies nowadays, focus their resources to a set of solutions, that support business and citizens behavioural patterns: in an smarter and environmentally friendly fashion. Driven by highly populated cities, a set of common problems arise. Analysis on that matter reveal that more and more citizens are moving into cities, resulting in high density population records on those areas. Approximately 54% of the world's population is located in cities [1] and this number may increase up to almost 70% until 2050 [2] [3].

History shows that industrial revolutions, tend to be implemented faster and to last shorter. At the time being, the fourth industrial revolution - known has "Industry 4.0", is being adopted. As any proper revolution, inherent to it, one can find some principles so far not fully explored or valued. More than ever, monitoring and data acquisition represents high value, both economically wise and as a tool to provide better services. A remarkable market share, has achieved impressive results based on data acquisition, selling it as meaningful information.

Being a dynamic new trend, an ongoing development of smartening every little human usable object is a reality. Some of them are just a market happening, fed by human fantasies: as the ability of talking to their watch. Although, many environmentally friendly, logistically feasible and economically responsible solutions emerge, providing industries, stakeholders and citizens' lifestyle with ground-breaking options.

The migration patterns, high energy consumption, and industrial revolutions, when related, fit the overall definition of a smart city.

Pavnext's road pavement energy harvesting (RPEH) system emerges as solution to smarten road pavements. As a whole, the Pavnext's RPEH solution promotes road safety, sets an environmentally friendly electrical energy production unit and performs traffic logging.

The solution to build a road pavement energy harvesting infrastructure, is growing towards its final design and prior to master thesis start, prototypes manufacturing has started. The following months being dedicated to test the prototype under a real environment, place the Pavnext solution between TRL6 and TRL71[1].

A test stage takes place at a private and controlled environment, comprising a conceptual implementation of the system and its last prototype version. Used as a test bench, during the first semester of 2019 several stress tests are scheduled. The company plans to implement a pilot version of the system by mid 2019, on a environment that fits its design: an urban location, targeting light-vehicle traffic.

Road safety enhancement and environmentally friendly electrical energy production wise, the Pavnext's system follows the Energy Harvesting concept (also called energy scavenning): by which energy is captured, converted and stored, not using fossil fuels and enabling decentralized generation units [5].

Monitoring the RPEH system allows the traffic logging potential to be explored. Also, as the mechanical system is being refined, monitoring it would provide useful information to consider for the ongoing prototype development stage.

Figure 1.1 illustrates the Pavnext solution overview. The monitoring and communication systems play a core role from end-to-end operational fields, in order to provide the ground basis for control processes to be later developed.

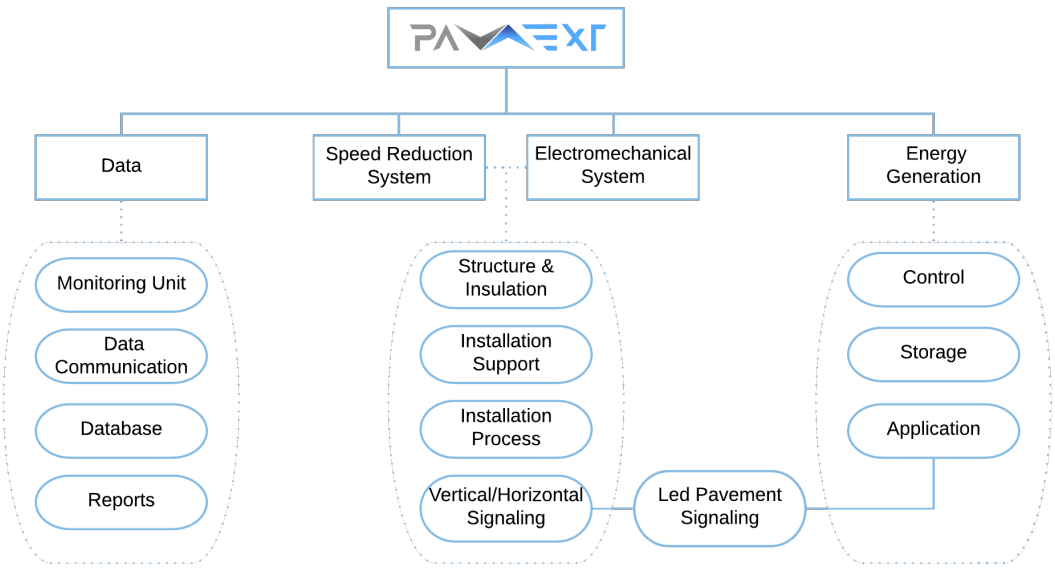


Figure 1.1: Pavnext's RPEH Overview

[1]Technology Readiness Levels (TRL) are a type of measurement system used to assess the maturity level of a particular technology [4].

1.2 Objectives

The main objective may be summarized as the task to design and test a system capable of monitoring the pavement solution.

Embedded on the road pavement energy harvesting system yet under development, the system must guarantee feasible integration and should also provide the possibility to stack more features and resources on top of the low-level architecture, given the premature state of the project subject to non-obvious changes. Figure 1.2 illustrates ongoing and pending projects, the master thesis scope, and how the master thesis fits into the development process.

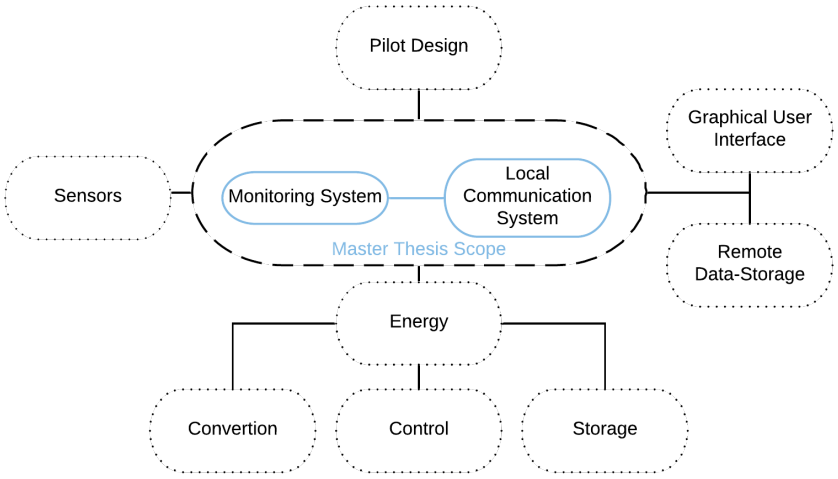


Figure 1.2: Scope of the work

1.3 Reading Guide

This document is structured in eight chapters and one annex. In some images there are blurred areas due to confidentiality issues.

Chapter one, the present chapter, features the context and motivation of this work, its objectives, and a brief outline of the dissertation.

Chapter two discusses the concept and presents the system overview.

Chapter three presents three different approaches for the solution. Market analysis, cost-benefit study and key performance indicators are some of the topics addressed.

Chapter four is a literature review of the used communication technology.

Chapter five describes the workplan, methodology and process execution. Decision making, deviations from the solution implementation defined at chapters two and three and difficulties or achievements, are included in this chapter.

Chapter six documents the development of the project and chapter seven presents and discusses the results.

Finally, chapter eight gives an insight on possible future work.

Chapter 2

System Characterization

2.1 Introduction

This chapter provides a brief description of the mechanical system to be monitored and helps to establish the context of the developed work.

2.2 Concept

The Pavenext RPEH system is a pavement energy harvesting system capable of slowing vehicles without the direct intervention of the drivers. It provides:

- enhanced road-safety;
- environmentally friendly electric energy;
- relevant data acquisition and creation of meaningful information.

Road vehicles release energy by way of different components, and part of the energy is transferred to the road pavement. The maximum energy transfer to the pavement occurs when there is a break action or when the wheels face an elevated obstacle - 15 to 21% of the energy transmitted from the vehicles wheels onto the the pavement is released as vibrations and heat [6].

The Pavnext RPEH system extracts the energy delivered to the pavement by a moving vehicle, collects and transforms it into other forms of energy, namely electric or hydraulic. The system is actuated by the passage of a vehicles wheels over it. The vehicle-to-system interaction is described as: when a vehicles wheel reaches the units surface, a downward

vertical displacement is induced; as soon as the wheel no longer contacts the device, the units movable surface starts an upward movement until it reaches the initial position. These two vertical movements are the source of mechanical energy that will be transformed in hydraulic or electric energy.

The third mentioned feature - data acquisition and information creation focusing traffic logging, emerge as result of the system privileged positioning. Being placed at urban environments (for example in the vicinity of cross-walks) the system can trigger pavement lights whenever a vehicle crosses the system. This action can be linked with the monitoring of ambient light resulting in the powering of the pavement lights only when needed rather than managed by a fixed schedule. This way the energy waste can be minimized. Still related to the data acquisition and information creation, is the opportunity to perform traffic-logging. Aside from counting the number of vehicles, the system has potential to determine the profile of each passage: speed, energy produced, number of axles, weight and weight distribution.

2.3 System Overview

Focusing on urban environments and light vehicles, the system is currently designed as a set of modules, each divided in single units: inner and outer side units. Figure 2.1 illustrates a scenario with 6 modules (3 per lane) and a route to accommodate a soft passage for two-wheelers[1].

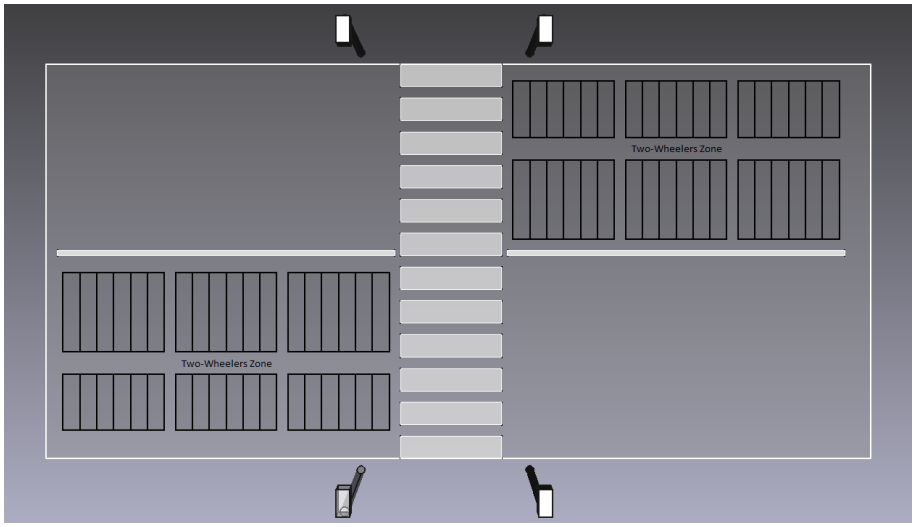


Figure 2.1: Implementation scenario with 3 modules per lane (top view)

[1]The number of modules does not represent the RPEH maximum extension, but just a possible implementation scenario.

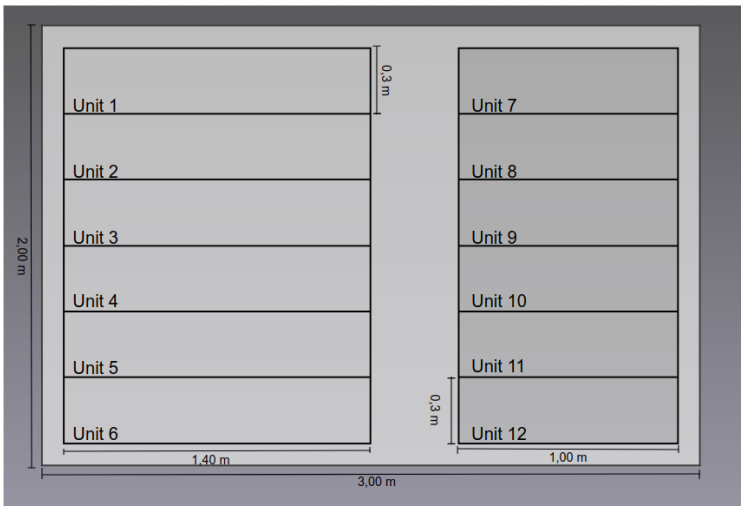


Figure 2.2: Representation of a single module (top view)

Each module is 3 meters wide and 2 meters long, comprising 12 units as shown in figure 2.2. Units placed on the inner side of the lane are larger than the ones on the outer side: 1,4 meters wide for the inner modules and 1 meter wide for the others. All the units are 0,3 meters long. Currently there are four different unit models: (a) two spring based units: M100 and M140; and (b) two generator based units: G100 and G140.

The M140 and G140 units are placed at the outer side of the lane whereas M100 and G100 are placed at the inner side. Whilst M140 and M100 units cannot produce electric energy G140 and G100 units carry two electrical generators, one at each end of its base as shown in figure 2.3.

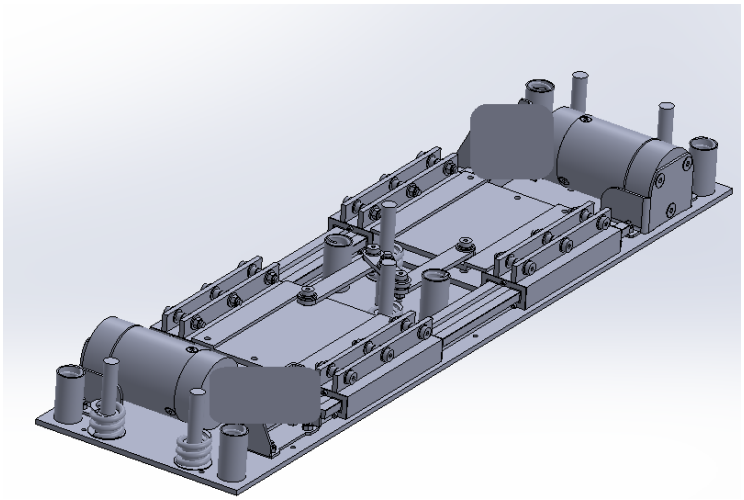


Figure 2.3: Unit G100 Base - Isometric view

For both types of units, the vertical displacement is internally converted into an horizontal displacement. For the M140 and M100 units, the horizontal displacement induced by a vehicle passage is supported by a spring system, where the goal is to reduce vehicles speed. Besides speed reduction, the G140 and G100 units use the horizontal displacement to actuate a rack-and-pinion mechanism coupled to an electrical generator, allowing electrical energy to be produced - figure 2.4.

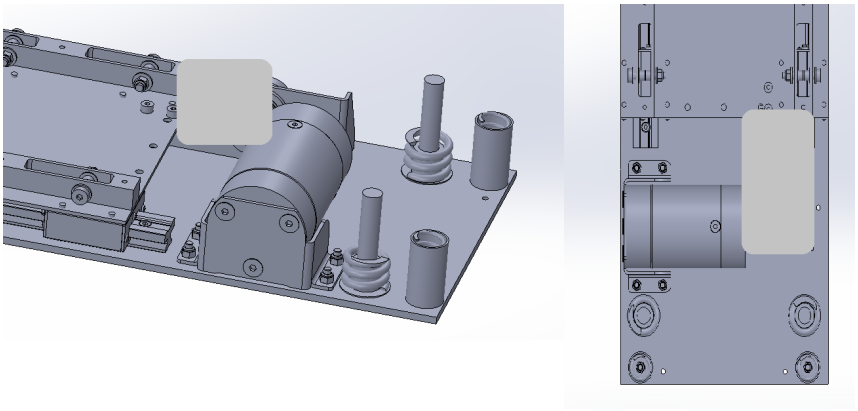


Figure 2.4: G140/G100 units - electrical generator configuration

The RPEH system is designed for light vehicles which weight distribution per axle is bellow one ton and travelling speed less than 50 Km/h which are the main vehicle-to-system interactions for urban environments.

The slope profile and the units cover length are designed for better yields in such scenarios; however, the system is able to withstand vehicles whose speed exceeds 50 Km/h and even heavy vehicles that may circulate carrying more than the allowed load. The mechanical design supports a maximum vertical displacement of 30 millimeters at a maximum velocity of 4 m/s.

Regarding the final product, the system is highly modular and different implementations may have a different number of modules. Towards establishing a limit for the number of modules, predicting implementation scenarios, no more than 10 modules per lane are considered. Such limit implies 20 meters of RPEH length to be monitored.

2.3.1 Units Composition

Each unit is composed of three main components: cover, base and insulation case as shown in figure 2.5.

The cover is made of a plastic layer between 2 layers of aluminum and as an interface to vehicles wheels, a layer of rubber covers the units entire length.

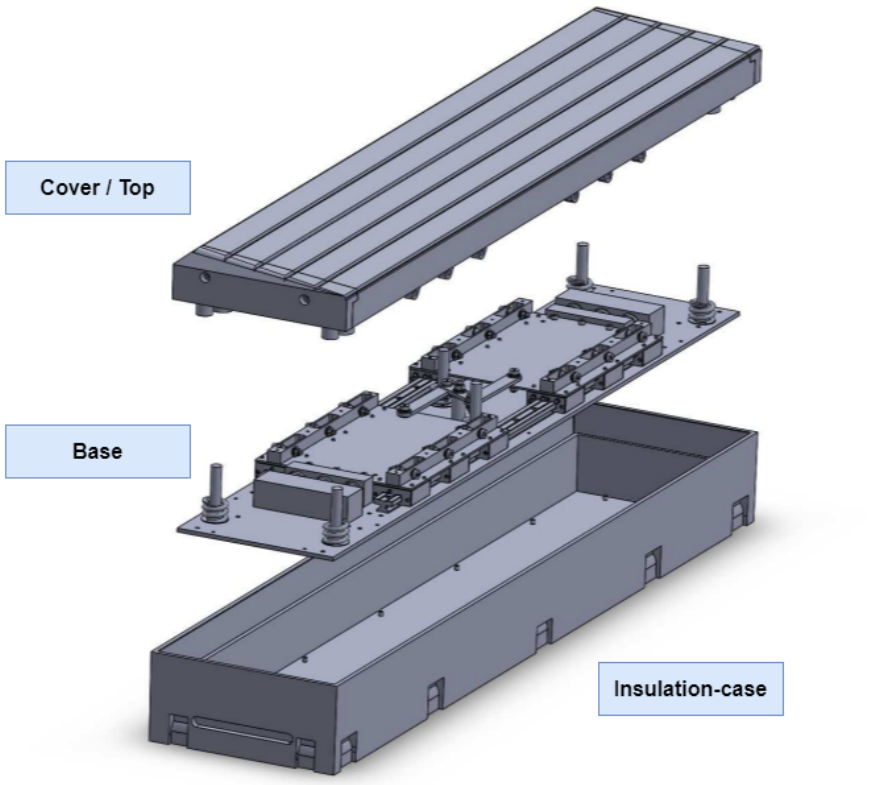


Figure 2.5: Units cover, base and insulation-case

The base is made of aluminum and allows the various elements to be fixed. The interconnection of the cover and the base is made by 12 connecting rods, distributed in groups of 3.

In the upward/downward movement are the connecting rods that distribute the applied force evenly; this results in the movement of 2 plates that run in 2 skates, in opposite directions. A central element ensures that the movement of the two plates is uniform during downward/upward movements with a maximum horizontal displacement of 30 millimeters, as well as the vertical displacement.

As soon as the two elements (lid and base) are connected, the unit is ready to be closed by a housing case. It seeks to insulate the units against external elements such as humidity and dust that can damage the system. Moreover, the insulation case has an external profile which allows the units to be fixed to the ground as soon as they are placed on the target location. To enhance the insulation, a layer of cork is applied inside the unit (placed on the cover, on the inner side), which promotes the balance of the units internal temperature.

When thoroughly assembled, each unit has the airflow restricted to a minimum: only

through holes that allow drained water from the cover to the sides of the case to flow out from units case. Moreover, the units external surface is made of dark coloured rubber to minimize the visual impact, so when exposed to sunlight, the temperature inside the unit rises, being aggravated during intensive cycles of operation.

The power generation system is composed of 2 generators per unit (G100 and G140) and due to the back and forth plates movement, each is actuated twice by a single axle. Each power generator carries an inertia-wheel, coupled to its shaft, so it is expected that the number of revolutions increase by the passage of consecutive axles.

The conversion, storage and energy management part of this system runs at its own pace, and is independent of the monitoring and communication project. Thus, it is not yet predicted whether the control will be done internally of each production unit or if it requires a communication system so that it acts in a distributed fashion.

Furthermore, another independent project is studying the viability of storing harvested energy in a hydraulic way, where the energy generated by each passage is stored in the first moment, then allowing to control the moment and the way in which the electric energy is generated. The applicability and performance of the system will be studied in comparison with the present concept, in which electricity generation is triggered by vehicle-passage events.

Regarding the interface with the parameters to be monitored and the interface with them, it is directly observable that the current prototypes do not have a stipulated location for the mounting and fixing of sensors, and lack a secure path for wiring avoiding direct contact with moving parts.

Once enclosed by the insulation cases, the purpose is to keep the units water- and dust-proof. For this reason, the insulation case does not have a proper way out for cables - something that needs to be adapted as the system evolves.

The G100 modules are the ones that most restrict the monitoring system placement within the units body, due to the constrained area to be shared with power circuitry.

2.3.2 Vehicle-to-System Interaction

The system consists of a set of units that make up a module. For the moment, a module is composed of a series of 6 units, each 1,40 meters wide, side by side with another 6 units of 1 meter wide. Due to the design for light vehicles, between the 1,40 meters and the 1 meter units, there is a 0,40 meters wide zone, free of successive road slopes, appropriate for two-wheelers.

At the maximum urban circulation speed of 50 Km/h (in Portugal) each vehicles axle crosses the 0,30 meter units length in 21,6 milliseconds, although, considering the tires surface that interfaces with the pavement, a slightly larger contact zone between the vehicles wheel and the units surface must be considered.

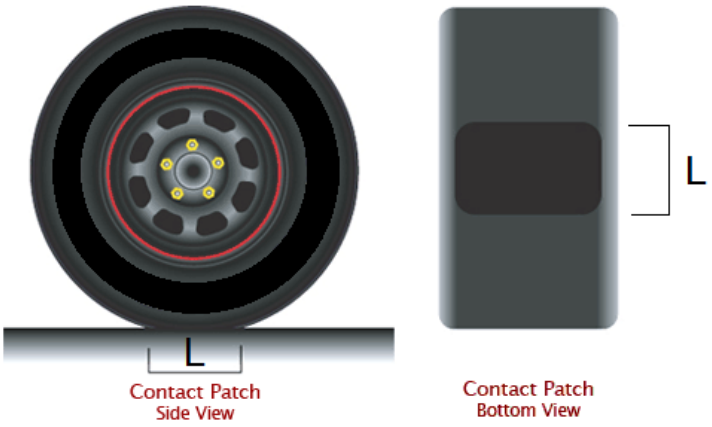


Figure 2.6: Tyre-to-Unit Contact Patch

The contact patch area that is in contact with the surface varies from vehicle to vehicle, so to provide an example value, 0,40 meters are considered. As a result, for 0,40 meters contact patch, at 50 km/h a vehicle crosses the RPEH and actuates each unit for about 29 milliseconds (per axle). However, considering that some drivers may not strictly follow the speed limits, a reasonable margin has to be considered. Table 2.1 resumes some reference speeds for the described situation, considering a contact zone length of 0,40 meters.

Speed [Km/h]	Actuation Period per Single Axle [ms]
10	144,0
20	72,0
50	28,8
80	18,0

Table 2.1: Actuation period per single Asxle

The movable surface will be displaced vertically as a function of both speed and weight per axle, and the expected pattern for a generic front-engine 2 axle vehicle is illustrated in figure 2.7.

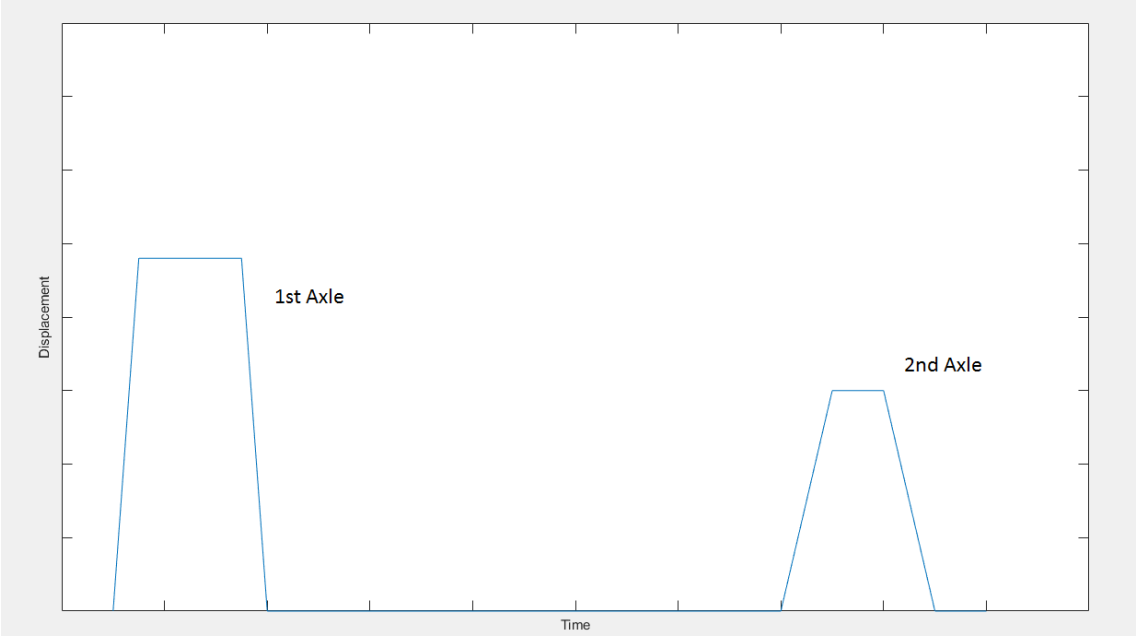


Figure 2.7: Virtual displacement - 2 axle vehicle passage

The mechanical prototypes still have many open issues and uncertainties, namely related to its dynamic behaviour and system-to-vehicle interaction. To have a better view of all these issues and of their interaction, a monitoring and communication subsystem was designed and tested.

The monitoring and communication system is intended to extract information from every unit, by sampling data according to the requirements imposed by each parameter and allowing the data to be post-processed and analyzed.

The expected result is to grant a solution that allows the Pavnext team to integrate sensors in every unit and fetch the data to a single machine placed at the outer-side of the road-lane.

This way besides logging traffic information, the monitoring capabilities are extended to internal data that will allow to assess the operational correctness of the system.

In synthesis, it is expected that, in the future, the monitoring system could be integrated with the other projects, in order to respond to the needs shown in figures 2.8 and 2.9.

The 4 mentioned prototypes will be used in the test-stage to provide the answers to the many still open questions. From noise and permeability tests, to the system mechanical validation, the learning process will dictate whether or not the envisaged path is viable.

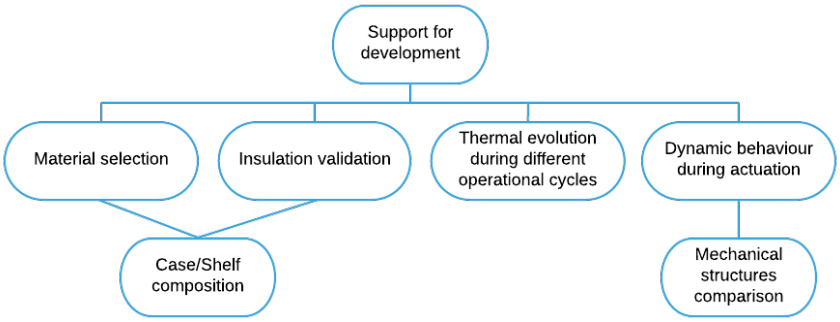


Figure 2.8: Support for the development process

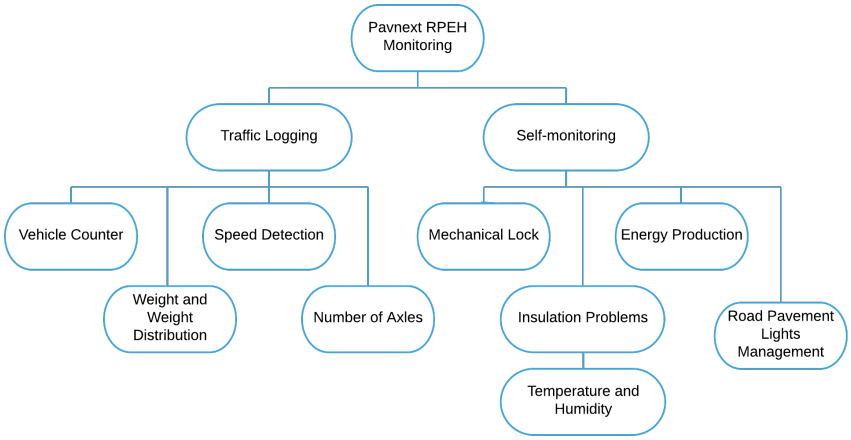


Figure 2.9: Monitoring requirements of the final prototype

Once the test-stage has been completed, hopefully giving leads for changes or improvements to be made, the monitoring and communication system will integrate the final prototype, and will allow the system to respond to the needs shown in figure 2.9.

The monitoring and communication subsystem is the main focus of this work and is described and discussed in the following chapters.

Chapter 3

Proposed Solutions

3.1 Introduction

The monitoring and communication system can not be completely built without defining the restrictions and considerations addressed in chapter 2. Nevertheless, the possibility to design the solution side-by-side with the development of the RPEH system allows to make the first steps in order to promote a modular architecture as much as possible independent of the system mutations, and simultaneously promoting a seamless integration of all units.

A possible solution is first endorsed in abstract terms, then the expected operation of the system is summarized, followed by an exercise of design space exploration for specific solutions addressing the needs of the final product.

3.2 Overview of an abstract solution

The monitoring and communication system may be addressed by two different approaches:

1. An application specific design, where the tools and solutions are restrictively designed and used for the specified purpose.
2. An application specific design, where the tools and solutions used to solve the problem are general enough to target a wide spectrum of similar problems.

The first approach tends to be expensive unless a large scale production is anticipated. Moreover, such approach is only feasible when the context is completely defined and will not change in the future. On the other hand, the second approach carries the inherent

advantage of its components: flexibility achieved by general purpose modules grouped to form a specific solution. However, the second approach also carries problems such as component compatibility and large number of components requiring maintenance and support. At the current stage, the Pavnexts concept is still prone to suffer mutations. Prototyping is a learning process, and precautions should be taken when designing a monitoring system at the same time.

Lessons learned dictate that a long-term vision regarding scalability, variations and growth potential, is difficult to attain. Nonetheless, the choices made should not, by default, preclude it. Lack of modularity will bottleneck any growth and imply costs to redesign, when/if needed. Contrarily, an overly optimistic view on growth and application scenarios can also lead to higher development and production costs, by including features not needed or unused. Any solution should carefully analyze this trade-off.

The solution may be of two different types:

1. Monitoring system for assistance to the development of the prototype;
2. Monitoring system for the final prototype.

A solution of type 1 is a system composed of 4 different units, in which the purpose of monitoring is to provide the ability to perceive the dynamic behavior over time, with a relative extension of information produced per unit. A set of problems that once solved, provide information that help the decision making process and bound the development path. It is characterized as an heterogeneous group of systems that requires different approaches.

A solution of type 2 is a system composed by a number of units dispersed by several meters over the pavement. The purpose of the monitoring is to provide the capacity to create the necessary information to trace the vehicles crossing profile and, at the same time, to verify the correct operation of each unit.

While type 1 is demanding on the amount of data per unit, type 2 is demanding on the total number of nodes to address, length of the communication medium and the the total amount of information to be transmitted.

In order to create a system of type 1, i.e. a system meeting the objectives illustrated in figure 2.8, there must be units that acquire data, by properly sampling the target signals, store and/or process the data so that it can be sent to a central unit. To meet the needs of a type 2 system, illustrated in figure 2.9, mechanisms for data acquisition, storage and/or processing are required, so that data may later be sent to a single unit. The volume of information to be produced in each of the scenarios defines the criteria to which the solution

to be designed will have to respond, otherwise the design may, by default, constrain the application.

It is also needed to define the means for storing, processing and transporting the information acquired, considering the desired profile, in a versatile and low cost implementation. The sensors themselves constitute an independent project running in parallel with this one.

3.2.1 Considerations on Data Acquisition

As a baseline, and regarding the expected results (figures 2.8 and 2.9), the following physical parameters are considered:

- temperature;
- humidity;
- vertical displacement;
- horizontal displacement;
- applied force over the unit surface;
- electric energy produced (only for the G140/G100 units).

Sensor wise, many options are available to perform each of the tasks. Market offers such a wide variety of solutions that many configurations may be implemented. For instance, for temperature and relative humidity there are independent sensors or sensors combined in a single package; there are models with serial output interface and models with analog outputs; there are many different precisions, etc. the diversity is huge, but as already stated, this is being addressed by a different project.

The solution must guarantee that water is drained. To perform such requirement, a draining path is connected to the local drainage system. The units are coated with P50 (a product developed by Amorim Cork Composites) to properly insulate them from noise and heat. The prototypes shelf is also designed to drain the water from the pavement surface without compromising the possibility of water falling into the units body. Nevertheless, measuring the relative humidity and temperature inside the unit body, can reveal insulation problems. Such measurement is useful not only during the development stages of the case, but also to provide an efficient tool to call for maintenance once the system is in production. The insulation constraints air flow and the design lacks an active mechanism to control the

temperature. Consequently, the inner temperature will rise during the day, aggravated in case of heavy operational cycles.

When a vehicle crosses the pavement structure, every unit performs a vertical displacement. This vertical movement is supported by the mechanical structure, sized to support over-loaded heavy-vehicles. Springs are responsible to absorb the impact and to push the structure back to pavement level once compression is released. The vertical displacement is mechanically converted into an horizontal movement inside the unit and distributed by 12 connecting rods that drive two metal plates. Connected to a synchronization mechanism, hence moving the same distance in opposite directions, the mechanical system design enables to reduce the number of displacements to measure: one of the displacements allows to calculate the other. The displacement measurement allows to evaluate how the load and speed of a vehicle interacts with the system. Moreover, since the physical implementation does not provide any mechanism to detect a vehicles passage other than the system actuation itself, the displacement measurement may also work as a vehicle-passage detection trigger. Displacement logging has an important role in fault detection, since all units should perform an homogeneous vertical displacement.

Once coupled, the top and base are connected (figure 3.1) by 12 connecting rods. Combined, they are intended to distribute the load evenly, assuring that even if a vehicle crosses the units surface on its far end (either left or right extremes), a stable and balanced vertical displacement is performed. All the twelve connecting rods are similar, hence the total load may be calculated if acquired the deformation that one of the rods was exposed to. Fault detection wise, by direct comparison between units readings, it is possible to figure when a unit has a structural problem. All units are designed to have the same dynamic behaviour when exposed to the same load and a vehicle's load does not change during its passage.

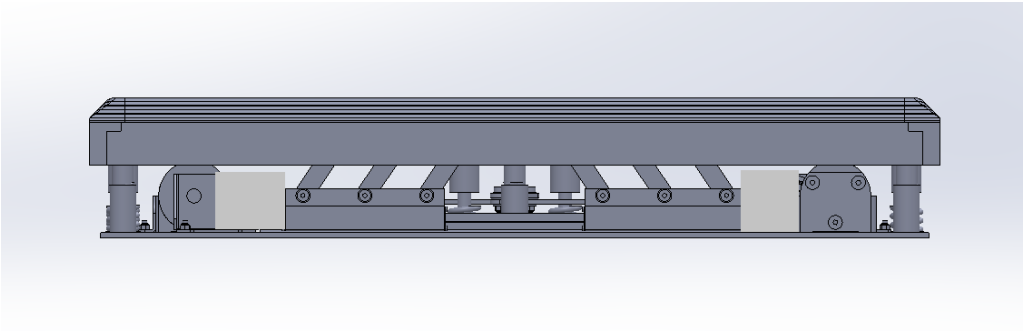


Figure 3.1: G100 top and base assembled

When acting as traffic logger this system makes the possibility to know the profile of the drivers and how the road is used, by computing secondary parameters such as:

- Traffic volume (per period of time);
- Light/heavy vehicles counting;
- Speed, direction and over-weight infractions report.

To determine vehicles speed many system resources can be used as long as the acquired data is tagged with a time-stamp. Vertical displacements for example, may be used to trigger the vehicle passage at a certain unit and associated with such trigger, a time stamp. Thus, given that all the units have the same length, it is trivial to determine the velocity a given vehicle entered and exited the system. For this purpose all data acquisition units must be time synchronized.

Units capable to convert the harvested mechanical energy into electric energy are equipped with two electric generators and a gear-based solution that is coupled to the rotor shaft. As mentioned, a vertical displacement of the unit makes two metal plates to perform a linear horizontal movement. A rack-and-pinion configuration actuates the generator and a regular 2 axles vehicle is able to actuate the electrical system 4 times, due to the back-and-forth horizontal movement executed by the sliding plates (downwards followed by upward displacement). Units that produce electric energy are to be connected to a bank of super capacitors currently under evaluation, placed under the two-wheelers passage zone. This connection means that the unit must have an exit on its insulation body, which may be used for the monitoring and communication system.

3.2.2 Considerations on Digititalization

For the purpose of making the acquired data to be analyzed, potentially filtered, stored or transmitted, two main option are considered:

- a micro-controller with a built-in ADC;
- a dedicated external ADC.

External ADCs, when compared with the built-in ADCs that most commercial micro-controllers have, have strong points on their favor:

- greater bit depth;
- greater sampling rate;
- several sampling methods available;

- differential inputs.

The micro-controller option has the main advantage, when compared to the dedicated ADC option, of a single package incorporating both computation unit and an ADC. Due to the desired characteristic of a modular system that can accommodate system changes, having a general purpose equipment capable of doing both computation and acquisition, is a plus. Although, if data sources are spatially distributed, the path between the data source and the micro-controller unit sets this option more prone to noise and data corruption. Several external ADCs placed near the data source promote the possibility of data being exchanged as a digital signal, minimizing noise interference and contribute to an overall performance increase with respect to data acquisition.

3.2.3 Considerations on Computation

Driven by space limitations and the general purpose flexibility of the solution to be designed, one of the major candidates to compute the tasks needed is a micro-controller approach. A micro-controller not only stands out by its small package size but also by the wide range of options this class has to offer.

In order to make a solid choice to one of the core components that will integrate the system, some key features are listed:

- processing power;
- pins and peripherals available;
- flash and RAM size;
- power consumption;
- package size;
- tool-chain and documentation.

The time needed to get used to tool-chain and the embedded architecture is also a critical feature, otherwise it might slow the development start, eventually compromising deadlines. Regarding the described considerations, any *obscure*[1] micro-controller option is to be discarded. The decision-making must also consider:

- volume (how many micro-controllers);

[1] Poor documentation, lack of examples and/or libraries, restricted tool-chain options/access, etc.

- micro-controller cost;
- testing cost;
- tool-chain cost;
- compilers and code analyzers;
- available libraries;
- firmware update and support.

Moreover, how easily the access to technical support and application notes is granted, the existence of development kits or how active the community is, are some other factors to consider.

3.2.4 Considerations on Communication Interfaces

Some embedded systems are self-contained units and do not require any interaction and data transfer with other sub-systems or external world. On the other hand, certain embedded systems may be a part of a large distributed system and require interaction and data exchange between various devices and modules. The external communication interface can be either a wired media or a wireless media and it can be a serial or parallel interface. Essential for communicating with various sub-systems of the embedded system and with the external world, the communication interface can be divided in two different perspectives: device/board level (on-board communication interface) and product level (external communication interface). Serial interfaces like I^2C, SPI, 1-Wire and parallel bus interface are examples of on-board communication interfaces available on low-cost micro-controller boards. Serial, USB, CAN, Ethernet, are examples of system-level communication interfaces.

The system requirements entails the need for data acquisition and a way to send information to a single machine, placed at the outer-side of the road-lane. In the solution space definition process, requirements as throughput, implementation cost, robustness, error detection are important, as well as the possibility for the solution to be implemented a few centimeters under the pavement level, covering a few dozen meters, possibly inside an insulation case.

After scrutinizing the physical and spatial structure of the system, with regard to the distribution of acquisition/processing units, the most appropriate implementation can be defined.

3.2.5 Proposed System Architectures

As described, the possibility to design a system that is capable of fitting both the development and final product is desired. Core topics that connect both problems are the data acquisition and the communication system. Whereas the pilot version is intended to provide the necessary means for traffic-logging side-by-side with the self-check capabilities, the system to monitor the 4 prototypes configuration requires that the information figures the whole signal profile of each parameter under evaluation. Once established how acquisition and communication systems should perform for the pilot version, it is possible to determine if the data-logging requirements regarding support for development are attainable (using the same configuration).

The following three system architectures address the data-acquisition and processor distribution in different ways. They are discussed and compared and a detailed analysis on the impact of each architecture is presented.

3.2.5.1 System Architecture 1 - SA1

The first proposed system architecture is based on a simple single-unit monitoring system developed at Pavnext, as illustrated in figure 3.2. The single-unit monitoring system uses an 8-bit micro-controller with 6 multiplexed ADC channels - an AVR ATMega328p - to acquire, process and provide a communication interface.

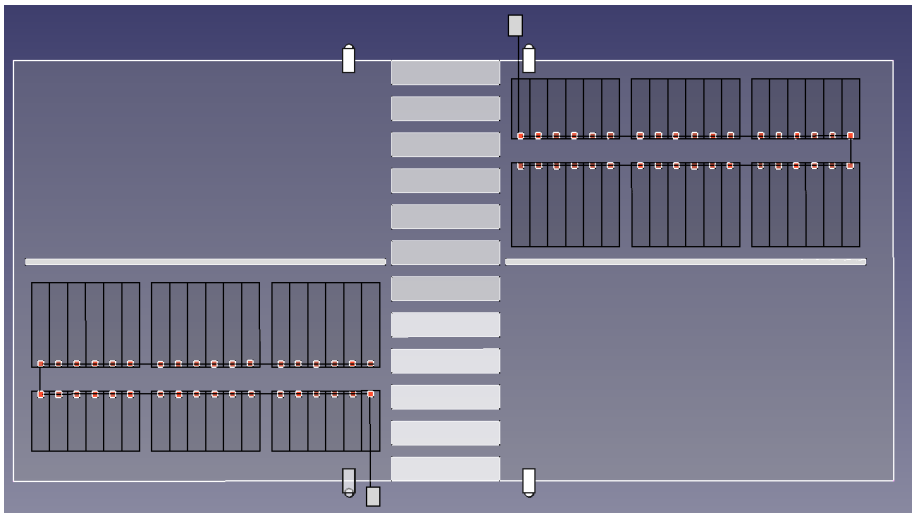


Figure 3.2: Representation of system architecture 1

Such system architecture has one micro-controller per unit, thus responsible for the units monitoring. Avoiding extra hardware, the number of signals to digitize define the required number of analog channels that a micro-controller must have.

Looking towards communication between micro-controllers, since all of them are inside the insulated body of each unit, under the pavement, wireless options are discarded. Units that carry an electrical generators also have wires that go through the insulation case, connecting them to the super-capacitors placed outside of the lane. Thus, the same wired path may be used to implement a communication medium, therefore minimal impact is introduced by this design option.

As the units are positioned side by side, a possible way to implement the wiring is to introduce two breakage points into the insulation case, freeing the need to use the central aisle - exposed to moisture, dust and other adverse agents.

3.2.5.2 System Architecture 2 - SA2

The second proposed system architecture derives from the high number of micro-controllers installed per module in the previous discussed architecture. Since (currently) the main purpose of the micro-controllers is to acquire a few parameters per unit, choosing an appropriate micro-controller - for instance, with enough ADC inputs for monitoring 4 units, results on the system architecture represented at figure 3.3.

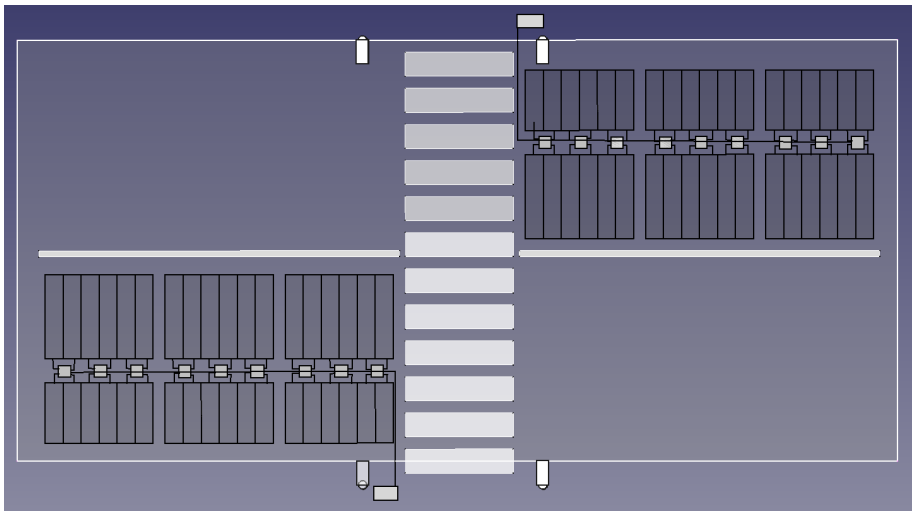


Figure 3.3: Representation of system architecture 2

The path mapped to the wiring in this scenario requires the use of the central aisle, located underneath the two-wheelers drive structure. Thus, the number of wiring input/output points in the insulation box is reduced by half, but requires means for protecting the micro-controllers from adverse environment agents.

3.2.5.3 System Architecture 3 - SA3

The third system architecture is a hybrid solution. This solution still has a micro-controller per unit, but also includes a second hierarchy layer. Figure 3.4 illustrates a total of 6 modules, where each module carries a micro-controller per single unit, plus another micro-controller to close the module architecture.

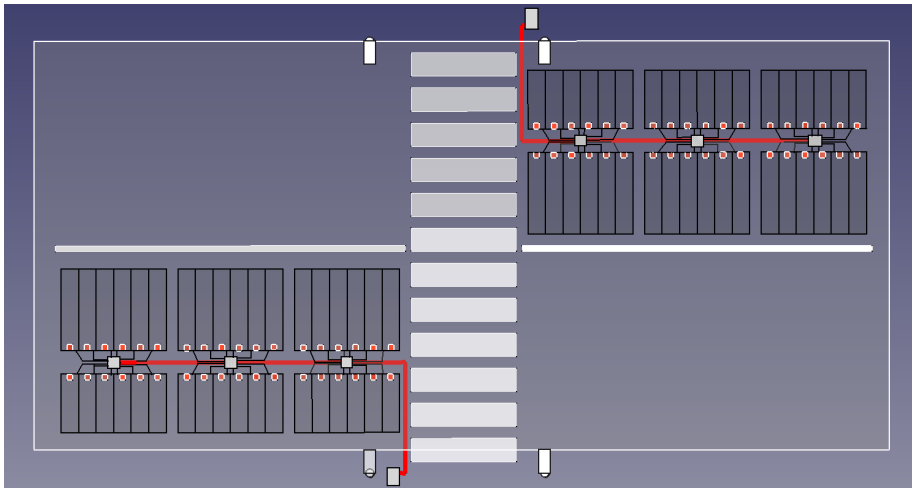


Figure 3.4: Representation of system architecture 3

Each lane has its own wired routing to the outer side of the road and the wired path between modules is placed under the 0,40 meter wide two-wheelers zone.

The thicker line going through all the modules illustrates how they may be interconnected, whereas the thinner lines at each module represents how the first layer (each unit's micro-controller) is connected to the second layer.

3.3 Comparison of system architectures

To efficiently compare the three proposed solutions some metrics and restrictions were defined and are illustrated in figure 3.5. The monitoring and communication system should fit the RPEH system, ideally without invasive or costly deviations to the units design. Once figured out the physical distribution of the data-acquisition and processing units that conveniently fits the production and installation processes, the less restrained features may be addressed, eventually leading to the discussion of the final solution as a whole.

This comparison reflects the discussions held in the first meetings of the team, during the initial phase of the work. At the time it was noticeable the necessity to close the

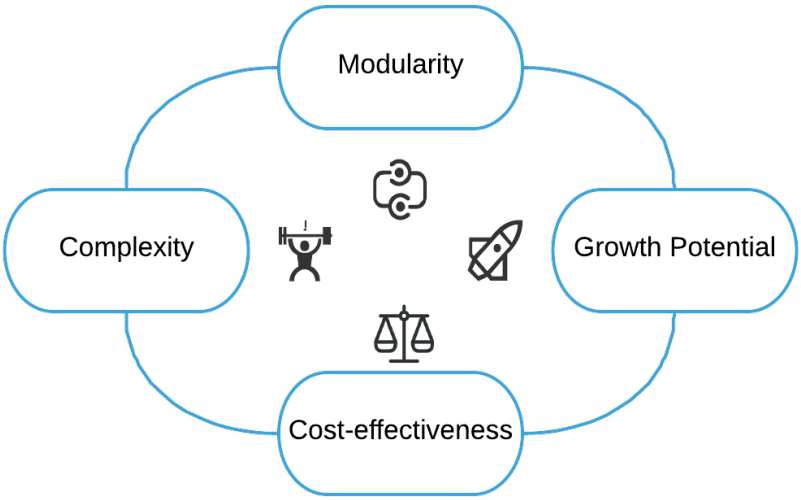


Figure 3.5: System architecture comparison metrics

abstract discussion to a set of metrics and goals, in order to narrow the solutions space and define the approach to take.

Regarding the implementation scenario, the monitoring and communication system implementation must consider assembling and disassembling procedures, so that to be as simple and fast as possible. If the configuration allows to easily unplug a faulty module and plug a new one, the time a road or lane is blocked due to maintenance is minimized.

SA1 requires a large number of micro-controllers just for the purpose of acquiring data inside each unit case - at the time no other features were defined. Such abstract model is inspired by the possibility to produce single units in a standardized fashion, comprising only power and communication wires from within the insulation case to go through the it, then possibly connected to other units using the path under the pavement.

In SA2 (SA2) the number of micro-controllers is decreased by using one micro-controller to monitor more than one unit. Although, another crucial technical requirement is present: the shortest the distance between the analog signal and the digitization unit, the better, as noise interference is by design decreased. Long wires between the analog sensors output and the micro-controllers input affect the measurement and thus the monitoring system performance. The implementation possibilities are described in chapter 6, but for comparison purposes its important to mention that the micro-controllers placement inside the units case must be done at one of its ends and G140 and M140 units are 1,40 meters wide. Thus, if a sensor is placed at the opposite end, aproximately 1,40 meters of wire carrying an analog signal might be required (plus the routing workarounds).

SA2 significantly reduces the number of micro-controllers needed (a quarter of the micro-controllers imposed by SA1), but placing the micro-controllers outside of the units case increases length between sensors output and ADCs input. Also related to cabling, all system architectures require at least power and communication cabling to penetrate the units insulation shelf, whereas SA2 also requires sensors wiring to be accessed from the outside of the units case.

Regarding hardware protection against humidity, temperature and dust, SA1 allows the insulation provided by the units case to be used to protect the micro-controller devices. Contrarily, SA2 requires extra hardware for the same purpose.

In the absence of a detailed procedure, substituting a faulty unit seems to be simpler and faster with SA1: remove the faulty unit and insert the new one; plug the device to the central wiring bus and check the system correctness. Although, (currently) the process of removing an installed units requires the removal of the two-wheelers' path structure in order to access a single unit, meaning that besides the cabling configuration needed to perform an installation/substitution, both SA1 and SA2 require the same process/time.

System architecture 3 (SA3) increases complexity, but is the most featured solution. Its purpose emerges from the communication system considerations addressed in a premature stage, where SA1 and SA2 allow a relatively short maximum number of nodes for certain addressing schemes (for example I^2C 7-bit addressing scheme), which might be extended by introducing a second communication layer. Intended to perform the lower layer of the monitoring system, all solutions need to sample the sensors and deliver the data. Data loads for traffic logging are relatively shorter than the necessary amount of data to evaluate the dynamic behaviour correctness. System architectures 1 and 2 require to choose devices capable of storing that amount of data before delivering it. System architecture 3 allows the devices inside each unit to be less featured by introducing a second layer of data storage plus a local communication network for each module.

The complexity increase does not introduce a proportional level of benefits, as for instance, SA1 and SA2 might just have dedicated memory ICs to overcome the memory requirements during intensive traffic loads. Also, as the mechanical design matures, it is expected that the necessary amount of self-monitoring requirements decrease. For instance, some of the scheduled tests are to evaluate different mechanical structures approach and performance. This will require a dedicated setup to produce the desired answers, which will lose its value/purpose once the results are evaluated and corrections are performed.

No single system architecture discussed is a clear winner, each bringing singular characteristics to the system.

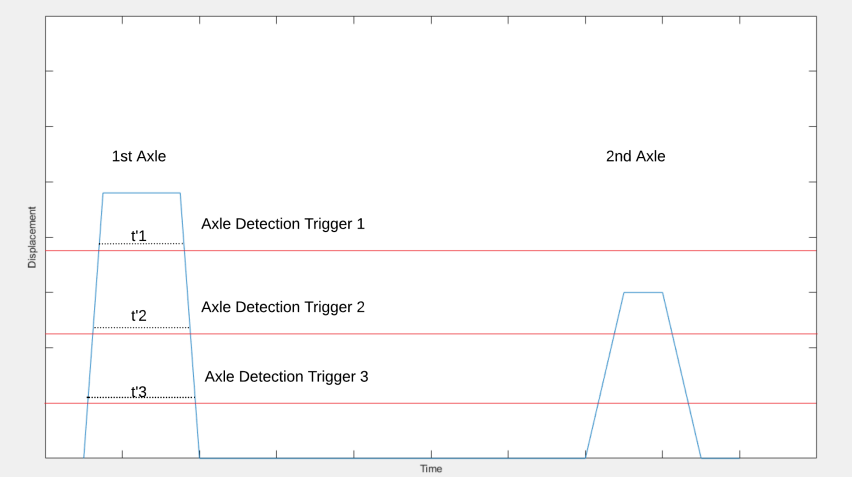


Figure 3.6: Data logger requirements

3.3.1 Computation Unit

The choice of the computing unit requires knowing in advance the requirements that will be imposed on the system. So far the objective of monitoring the infrastructure, requires data that allows to compute:

- the number of vehicles;
- the time interval associated with the passage of a vehicles axle;
- the maximum displacement induced per vehicles axle;
- current and voltage, time-stamped so that their values can be linked to the axle passage event;
- number of revolutions performed by the electrical generator.

The parameters above, allow to define the vehicle-system interaction. However, the requirements derived from the structural and dynamic point of view of the system require additional parameters, such as temperature and/or humidity inside the unit, or the complete upward/downward displacement report.

Such scenario is still relatively reduced in number of bytes per vehicle/axle, when compared to the need of a support system for development, where the requirements dictate to extract information periodically on each of the parameters, allowing the central unit to deliver sufficient data to represent the virtual vehicles passage represented in figure 3.6.

To trace the vehicle passing profile with detail, the associated number of bytes increases significantly. While it is not clear what kind of sensors will be used, and the expected end, the task of linking a choice to the computing unit becomes unbounded. As an example - this example results from an analysis of possible tests that may be performed in the future, if an accelerometer is used to determine the maximum displacement, then processing is required to determine the maximum distance traveled. Moreover, the angle performed by the units top, relative to its base, when vehicle passage occurs, may be determined by the same approach - an accelerometer, or by using multiple linear displacement sensors spread by the units structure. This example figures one of the possible test stages to find out if there is a purely vertical movement when a vehicle crosses the units surface at one of its ends rather than crossing it centrally.

While the use of an accelerometer requires some processing to determine the maximum displacement reached, the use of multiple linear displacement sensors requires, in addition to multiple ADC channels, essentially memory and transmission speed for the central unit, if desired that the the total path trace profile is sent, sensor by sensor, to the central unit.

Having multiple possibilities benefit the attack strategy, being a plus to have multiple solutions for the same problem. However, uncertainty at this stage, regarding the test stages and the setup that is required, make it difficult to choose a set of tools that are appropriate: do not restrict objective and do not entail costs that do not bring benefits.

Conscious of the uncertainty, resulting from the discussion at a stage prior to the choice of tools, the selection criterion is filtered by options that reasonably allow to work in a single architecture, minimizing the number of resources for which development and support is necessary, even though aware that at a later stage the port for an architecture dedicated to the final product is required. In this way, the development process is accelerated so that the answers can be given more quickly and a functional system can be obtained. However, the need for tuning the solution to a tailored system is necessary.

Based on the requirements described above (versatility, modularity and wide margin of growth), micro-controllers based on the ARM Cortex-M0 architecture are initially identified as potential targets for the units monitoring purpose, given the number of internal peripherals, communication options and acquisition of analog signals in a low-power configuration.

Micro-controllers based on the ARM Cortex-M3 architecture are identified as a potential solution for the unit placed outside the pavement, given the processing power and memory space for integration of the latter resources to be implemented towards the intended full-featured system.

Given the options provided by different manufacturers whose development boards

include these architectures, a preliminary analysis makes it possible to identify the product line offered by ST Microelectronics as a desirable solution due to the support provided both in terms of development tools and the vast documentation available. Moreover, the availability of these devices in the local market is vast and there are also several (unofficial) replicas as feasible options.

Considering system architectures 2 and 3, where the micro-controllers are placed outside of the units insulation case, wireless communication interfaces bring the possibility to perform some sort of interaction/configuration with modules without disassembling the infrastructure. Playing as the middle layer micro-controller described at SA3, ESP32 [7] has built-in Bluetooth and Wi-Fi interfaces that place it as a potential candidate. Although, SA1 would not explore its featured connectivity potential.

Micro-Controller	32-bit Core CPU	Max Clock	Flash	SRAM	ADC	Communications
STM32F103CB	ARM Cortex-M3	72 MHz	128k	20k	2 ADCs 8 x 12-bit	Up to 2 IIC and SPI 1 USB and CAN 6 USART
STM32F103C8			64k	20k		
STM32F103C6			32k	10k		
STM32F030C8	ARM Cortex-M0	48 MHz	64k	8k	1 ADC 16x12-bit	2 IIC, 2 SPI, 6 USART

Micro-Controller	32-bit Core CPU	Timers	Power Supply	Current	Temperature	Cost
STM32F103CB	ARM Cortex-M3	11 *2 x WDT *1 x SysTick	2.0V - 3.3V	373 uA/Mhz	-40 / 105	1
STM32F103C8				373		0.875
STM32F103C6				337		0.783
STM32F030C8	ARM Cortex-M0		2.4 - 3.3	458	-40 / 85	0.28

Table 3.1: STM32 Micro-Controller Comparison

The mainstream ARM Cortex-M3 based boards provided by ST Microelectronics, stand as an overall good solution. The STM32F103xx medium-density performance line family has many configurations available, that enables usage for both roles (monitoring unit and central unit). Less featured boards, such as the STM32F030x4/6/8/C micro-controllers series, ARM Cortex-M0 based, also fit a good solution for the monitoring role.

Reducing the the diversity of development architectures to support is a desired goal that allows to re-use the tool-chain, ease and fasten the development process and inherently, the development cost decreases. Although, the cost of carrying overpowered (expensive) devices for a simple task must be analyzed side-by-side with the control requirements that will be integrated as soon as the energy harvesting power management is introduced as well as the hydraulic system running inside each single unit. Furthermore, choosing devices that require the same power supply level not only enables a single circuitry design but also eases the communication interface integration which is required by certain communication protocols such as I^2C.

3.3.2 Data Acquisition

The data acquisition procedures are defined by the real-time dynamics and the solution to interface with it. At the time the solution design kicks-off, not much information about the sensors to be used is available.

Whereas some sensors are sophisticated and perform multiple samples before delivering the acquired value, others simply provide an analog output to be read. Procedures regarding data acquisition are delayed until the sensors definition is available.

3.3.3 Communication Interface

All proposed system architectures impose a communication interface and use the contemplated wired-path for power cabling. Regarding the serial interfaces provided by the micro-controllers previously referred, the two-wired Inter-Integrated Circuit protocol is a good solution to explore, given the the possibility to use it without any extra hardware for bus extension. This will require further tests regarding the extension of the bus for distances in the order of 20m.

3.3.4 Clock Synchronization

The existence of a global notion of time allows the micro-controller distributed system to agree on the time events occurred [8]. Clock synchronization on low-cost distributed embedded systems, where computation and communication resources are relatively constrained, makes unfeasible the approach to directly use the available solutions for the large-scale distributed systems paradigm.

Unless SA2 is used, where the same micro-processor monitors two groups of units physically separated by a fixed distance, is becomes necessary to synchronize the micro-controllers to a certain level of precision. Otherwise, the requirement of speed detection fails to succeed.

Given the low-speed characteristics of the chosen communication protocol, an asymmetric synchronization mechanism is the chosen approach. Unlike synchronization mechanisms that make use of time-servers so that each node of the network may request synchronization service, or synchronization services based on the message exchange between the networked devices so that the time correction of each node is performed based on the time of all the nodes of the system, asymmetric correction mechanisms only require that one of the nodes imposes its own notion of time and the remaining nodes correct their time-awareness based on the received value. The asymmetric correction mechanism

lacks precision when compared to symmetric synchronization mechanisms, but it favors architectures with limited resources due to the reduced overhead required.

3.3.5 Software Architecture

As the system carries a distributed configuration, combined with the number of different requirements and goals for each of the test stages, the usage of a real-time kernel will benefit the accommodation of different tasks that resemble the necessary approach from device to device, specially considering the control requirements that will be introduced on latter development stage. For instance, every device will perform data-acquisition procedures on a timely fashion as well as communication routines. The amount of data exchanged may differ, as it also may differ the sampling rate at each device, driven by the evolution of the real-time entity to monitor.

Regarding the monitoring and communication system, there is no immediate benefit of using an RTOS. In order for the task scheduling to take advantage of the RTOS mechanisms, the RTOS itself must run at a higher frequency; Moreover, scheduling, management and communication between tasks, entails an overhead. However, in view of future enhancements to the system it was decided to follow such approach since the beginning of the work.

FreeRTOS is the real time operating system selected because it supports dozens of different architectures (including the ones selected for this work) and is supported by a large and very active community.

3.4 Conclusion

Each system architecture carries benefits and inherent problems. The long-term vision is to develop a solution that easily fits the growth potential, placing it as a cost-effective approach.

SA1 represents the optimal solution by covering hardware protection necessity with the micro-controllers placement inside the units insulation case. Placing the micro-controllers hardware outside of the units case requires enclosures with Ingress Protection rating of at least IPx1 (depending on geographic location it might be necessary to use higher ratings against water ingress). Cabling requirements are evenly accommodated by all the systems, but SA1 is also the solution that requires the smallest quantity of connectors needed, which must be robust against water, dust and humidity. Feature wise, system architecture 3 is the one that carries the most potential, but also the one that faces more complexity, mostly

because the second communication layer. Reducing the number of micro-controllers but increasing the distance between the sensors and the digitization units, places SA2 as the worst solution for data acquisition. Also, considering that the only mechanism to detect a vehicle passage is the event of vertical/horizontal displacement of a unit, SA2 is the easiest solution to acquire the event and trigger the system from a power saving mode. Although, a hybrid solution might be ideal, promoting the master re-allocation of system 1 and 3 to one of the first units a vehicle faces when crossing the pavement.

Driven by the possibility of using the same architecture to run both master and slave roles, the ARM cortex-M3 based STM32F103 is the preferred development board. This choice is not the cheapest one but the one that reduces the number of different devices to support and figures as a suitable solution to assure integration and deviations required by system mutations. As the choice enables its usage for all the roles required on the system configuration, the time required to develop and maintain different architectures is reduced.

Also, it is important to mention that the number of micro-controllers to buy is equal to the number of micro-controllers needed for the test-stage: 5 units total. The bundle options that promote the cheapest price per unit is not an option at the time and market analysis over the international market channels allows to find non-official STM32F103 boards costing 0,25 euros more than the cheapest non-official STM32F030 on the same market channel. Regarding the time international channels take to deliver the product, the market analysis over national channels results in the same pattern for 5 units, although national supplier channels are 9 times more expensive than the international offer.

Using a distributed micro-controller approach, where the communication relies on a single-master I^2C protocol covering dozens of meters, brings a set of challenges that require careful analysis and proper test scenarios to be firstly addressed.

Chapter 4

Underlying Communication Technology

4.1 Literature Review on I^2C

Low-cost micro-controllers usually carry built-in internal peripherals to handle the I^2C or IIC serial synchronous communication protocol. In spite of being used for intra-board communications it was extended for system-level communications in this work.

The I^2C Bus is a time-proven, industry standard, communication protocol used in a wide variety of electronic products. The I^2C bus is a two-wire, low to medium speed, communication bus developed by Philips Semiconductors in the early 1980s created to reduce the manufacturing costs of electronic products [9] and it provides a low-cost, chip-to-chip communication link within these products. I^2C has expanded its role to include a wide range of applications and today, I^2C can be found in a wide variety of computer, industrial, entertainment, medical, and military systems.

The I^2C Bus Specification (version 6, updated in 2014), provides a communication protocol definition of the signal activity on the I^2C bus. This specification helps to instruct semiconductor device manufacturers, and electronic product developers, in the correct use of the technology.

Prior to I^2C, chip-to-chip communications used many wires in a parallel interface, often requiring ICs to have many dedicated pins to communicate. Many of these pins were used for inter-chip addressing, selection, control, and data transfers. In a parallel interface, 8 data bits are typically transferred from a sender IC to a receiver IC in a single operation.

I^2C performs chip-to-chip communications using only two wires in a serial interface,

allowing ICs to communicate with fewer pins. The two wires in the I^2C bus are called Serial Clock (SCL) and Serial Data (SDA). These two wires carry addressing, selection, control, and data, one bit at a time. The SDA wire carries the data, while the SCL wire synchronizes the sender and receiver during the transfer. ICs that use the I^2C bus can perform the same function as their larger parallel interface counterparts, but with far fewer pins. This greatly reduces the size and cost of ICs based on the I^2C bus but it also reduces the maximum achievable data rate.

A second saving from the two-wire I^2C bus design is in printed circuit board (PCB) size and costs. With ICs based on the I^2C bus needing far fewer wires (copper traces) for inter-chip communications, circuit boards using I^2C ICs are greatly reduced in size, complexity, and cost.

The protocol operates a master-slave hierarchy and I^2C devices are classified as master and/or slave. A device that initiates a message is called a master, while a device that responds to a message is called a slave. A device can be master-only, slave-only, or switch between master and slave, as the application requires.

I^2C can connect many ICs on just two-wires and each I^2C slave device has its own unique slave address. When a master sends a message, it includes the slave address at the beginning of the message. All devices on the bus hear the message, but only the slave that recognizes its own address participates in the transfer. The protocol also supports multiple master devices on the bus at the same time, a powerful feature that optimizes bus use by keeping bus message traffic to a minimum. To support multiple masters, devices must resolve signal conflicts, should two or more master devices try to talk on the bus at the same time. This feature, called bus arbitration loss detection, allows a master to detect when its bus signals are conflicting with those of another master. A master that detects arbitration loss terminates its use of the bus, allowing the message generated by another master to cross the bus unharmed.

I^2C uses an open-drain/open-collector stucture as shown in figure 4.1 with an input buffer on the same line, which allows a single data line to be used for bidirectional data flow. Open-drain refers to a type of output which can either pull the bus down to a voltage (ground, in most cases), or "release" the bus and let it be pulled up by a pull-up resistor. In the event of the bus being released by the master or a slave, the pull-up resistor R_{pu} on the line is responsible for pulling the bus voltage up to the power rail. Since no device may force a high on a line, this means that the bus will never run into a communication issue where one device may try to transmit a high, and another transmits a low, causing a short (power rail to ground). I^2C requires that if a master in a multi-master environment transmits a high, but see's that the line is low (another device is pulling it down), to halt communications because another device is using the bus. Push-pull interfaces do not allow

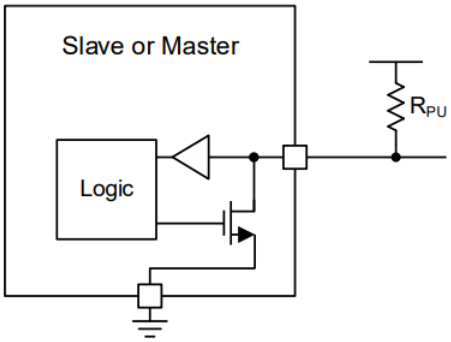


Figure 4.1: Open Drain Configuration

for this type of freedom, which is a benefit of I^2C.

Figure 4.1 shows a simplified view of the internal structure of the slave or master device on the SDA/SCL lines, consisting of a buffer to read input data, and a pull-down FET to transmit data. A device is only able to pull the bus line low (provide short to ground) or release the bus line (high impedance to ground) and allow the pull-up resistor to raise the voltage. This is an important concept to realize when dealing with I^2C devices, since no device may hold the bus high. This property is what allows bidirectional communication to take place.

Open-Drain setup may only pull a bus low, or "release" it and let a resistor pull it high. Figure 4.2 shows the flow of current to pull the bus low. The logic wanting to transmit a low will activate the pull-down FET, which will provide a short to ground, pulling the line low.

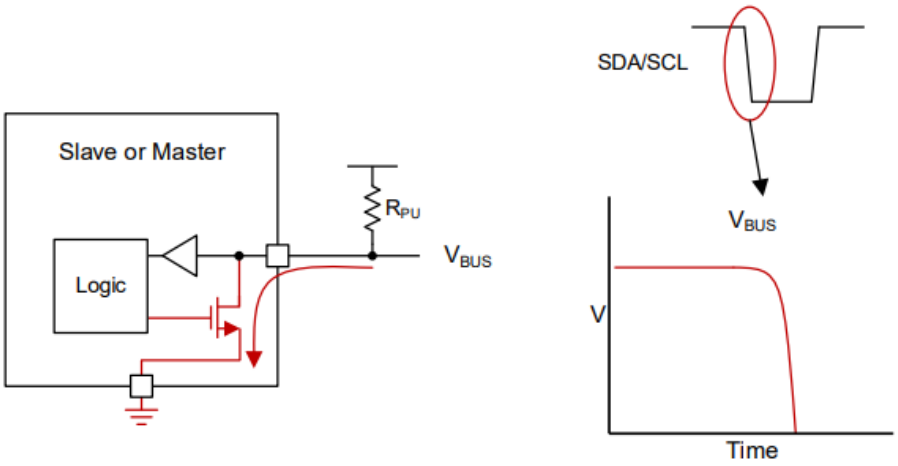


Figure 4.2: Open Drain Pulling Low

When the slave or master wishes to transmit a logic high, it may only release the bus

by turning off the pull-down FET. This leaves the bus floating, and the pull-up resistor will pull the voltage up to the voltage rail, which will be interpreted as a logic high. Figure 4.3 shows the flow of current through the pull-up resistor, which pulls the bus to a logic high state.

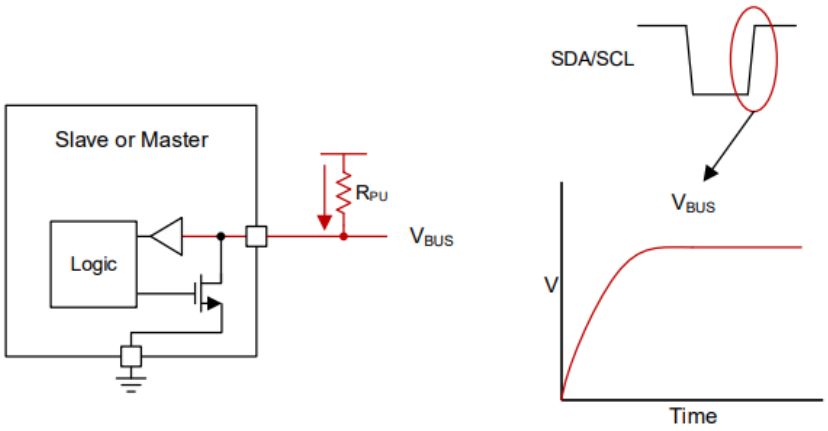


Figure 4.3: Open Drain Releasing Bus

The I^2C bus is a standard bidirectional interface that uses a controller, known as the master, to communicate with slave devices. A slave may not transmit data unless it has been addressed by the master. Each device on the I^2C bus has a specific device address to differentiate between other devices that are on the same I^2C bus. Many slave devices will require configuration upon startup to set the behavior of the device. This is typically done when the master accesses the slave's internal register maps, which have unique register addresses. A device can have one or multiple registers where data is stored, written, or read. The physical I^2C interface consists of the serial clock (SCL) and serial data (SDA) lines. Both SDA and SCL lines must be connected to VCC through a pull-up resistor. The size of the pull-up resistor is determined by the amount of capacitance on the I^2C lines. Data transfer may be initiated only when the bus is idle and a bus is considered idle if both SDA and SCL lines are at logic high state, after a STOP condition. The general procedure for a master to access a slave device is the following:

1. If master wants to send data to a slave:
    - Master-transmitter sends a START condition and addresses the slave-receiver;
    - Master-transmitter sends data to slave-receiver;
    - Master-transmitter terminates the transfer with a STOP condition;
2. If a master wants to receive/read data from a slave:
    - Master-receiver sends a START condition and addresses the slave-transmitter;

- Master-receiver sends the requested register to read to slave-transmitter;
- Master-receiver receives data from the slave-transmitter;
- Master-receiver terminates the transfer with a STOP condition;

I²C communication is initiated by the master sending a START condition and terminated by the master sending a STOP condition. A high-to-low transition on the SDA line while the SCL is high defines a START condition. A low-to-high transition on the SDA line while the SCL is high defines a STOP condition.

A repeated START condition is similar to a START condition and is used in place of a back-to-back STOP then START condition. It looks identical to a START condition, but differs from a START condition because it happens before a STOP condition (when the bus is not idle). This is useful for when the master wishes to start a new communication, but does not wish to let the bus go idle with the STOP condition, which has the chance of the master losing control of the bus to another master (in multi-master environments).

One data bit is transferred during each clock pulse of the SCL. One byte is comprised of eight bits on the SDA line. A byte may either be a device address, register address, or data written to or read from a slave. Data is transferred Most Significant Bit (MSB) first. Any number of data bytes can be transferred from the master to slave between the START and STOP conditions. Data on the SDA line must remain stable during the high phase of the clock period, as changes in the data line when the SCL is high are interpreted as control commands (START or STOP).

Each byte of data (including the address byte) is followed by one ACK bit from the receiver. The ACK bit allows the receiver to communicate to the transmitter that the byte was successfully received and another byte may be sent. Before the receiver can send an ACK, the transmitter must release the SDA line. To send an ACK bit, the receiver shall pull down the SDA line during the low phase of the ACK/NACK-related clock period, so that the SDA line is stable low during the high phase of the ACK/NACK-related clock period. Setup and hold times must be taken into account. When the SDA line remains high during the ACK/NACK-related clock period, this is interpreted as a NACK. There are several conditions that lead to the generation of a NACK:

1. The receiver is unable to receive or transmit because it is performing some real-time function and is not ready to start communication with the master.

2. During the transfer, the receiver gets data or commands that it does not understand.

3. During the transfer, the receiver cannot receive any more data bytes.

4. A master-receiver is done reading data and indicates this to the slave through a NACK.

Data must be sent and received to or from the slave devices, but the way that this is accomplished is by reading or writing to or from registers in the slave device. Registers are locations in the slave's memory which contain information, whether it be the configuration information, or some sampled data to send back to the master. The master must write information into these registers in order to instruct the slave device to perform a task.

While it is common to have registers in I^2C slaves, not all slave devices will have registers; some devices are simple and contain only 1 register, which may be written directly to, by sending the register data immediately after the slave address, instead of addressing a register. An example of a single-register device is an 8-bit I2C switch, which is controlled via I^2C commands. Since it has 1 bit to enable or disable a channel, there is only 1 register needed, and the master merely writes the register data after the slave address.

To write on the I^2C bus, the master will send a start condition on the bus with the slave's address, as well as the last bit (the R/W bit) set to 0, which signifies a write. After the slave sends the acknowledge bit, the master will then send the register address of the register it wishes to write to. The slave will acknowledge again, letting the master know it is ready. After this, the master will start sending the register data to the slave, until the master has sent all the data it needs to (sometimes this is only a single byte), and the master will terminate the transmission with a STOP condition.

Reading from a slave is very similar to writing, but with some extra steps. In order to read from a slave, the master must first instruct the slave which register it wishes to read from. This is done by the master starting off the transmission in a similar fashion as the write, by sending the address with the R/W bit equal to 0 (signifying a write), followed by the register address it wishes to read from. Once the slave acknowledges this register address, the master will send a START condition again, followed by the slave address with the R/W bit set to 1 (signifying a read). This time, the slave will acknowledge the read request, and the master releases the SDA bus, but will continue supplying the clock to the slave. During this part of the transaction, the master will become the master-receiver, and the slave will become the slave-transmitter. The master will continue sending out the clock pulses, but will release the SDA line, so that the slave can transmit data. At the end of every byte of data, the master will send an ACK to the slave, letting the slave know that it is ready for more data. Once the master has received the number of bytes it is expecting, it will send a NACK, signaling to the slave to halt communications and release the bus. The master will follow this up with a STOP condition.

I²C buses are not limited to a defined voltage. When different nodes use different operating voltages a level-shifter may be necessary.

When I²C was first introduced the typical electrical high-level voltage of electronics was 5V and the initially specified speed was a maximum of 100 kHz. With these parameters it was possible to operate over a wide range of electrical parameters in terms of bus termination and capacitance.

With the definition of fast mode, fast mode plus and high speed the timing requirements became more strict and this implied the need for higher terminations, lower capacitance and minimized serial resistance. Apart from the bus speed more and more circuitry is moving to 3.3V or even 1.8V and below logic levels. An example is NXP's IO expander PCA9554C [10]. Since I²C is an open drain concept the high-level is not critical for the operation, as long as all components on the bus can accept the voltage on the Input/Ooutput pins and are able to detect the logic levels.

Chapter 5

Methodology

5.1 Work Distribution

Designing and testing an embedded system to integrate an ongoing development process is a risky endeavour, prone to suffer delays or deviations. The development process established relies on successive development-validation steps, from the crucial data-acquisition and data-communication tasks up to less nuclear features.

Looking towards bounding work-package execution periods, the Gantt chart in figure 5.1, is the result of defining a feasible schedule, comprising the period and order of execution of each of the tasks.

The prototypes' production stage is scheduled to finish by week 4 and the 5 months period through which this work is performed - from February to June, are set to design, develop and use the prototypes as a test-bench. The opportunity to follow the prototype's production stage allows to understand the system composition and the role played by each component.

The sensors project is scheduled to start by week 5 (March), meaning that the sensors integration is delayed until its definition. A successful test stage regarding I^2C long-range, enables to use this slack period to address the protocol definition and the procedures by which master and slaves interpret each other commands. An unsuccessful test-stage regarding the long-range communication without resorting to bus extenders hardware, defines a crucial milestone, which might require a solution reformulation.

Once defined the sensors to use, it is possible to design the embedded system for each of the test-stages. Following the design defined, development, implementation and validation occur in a succession.

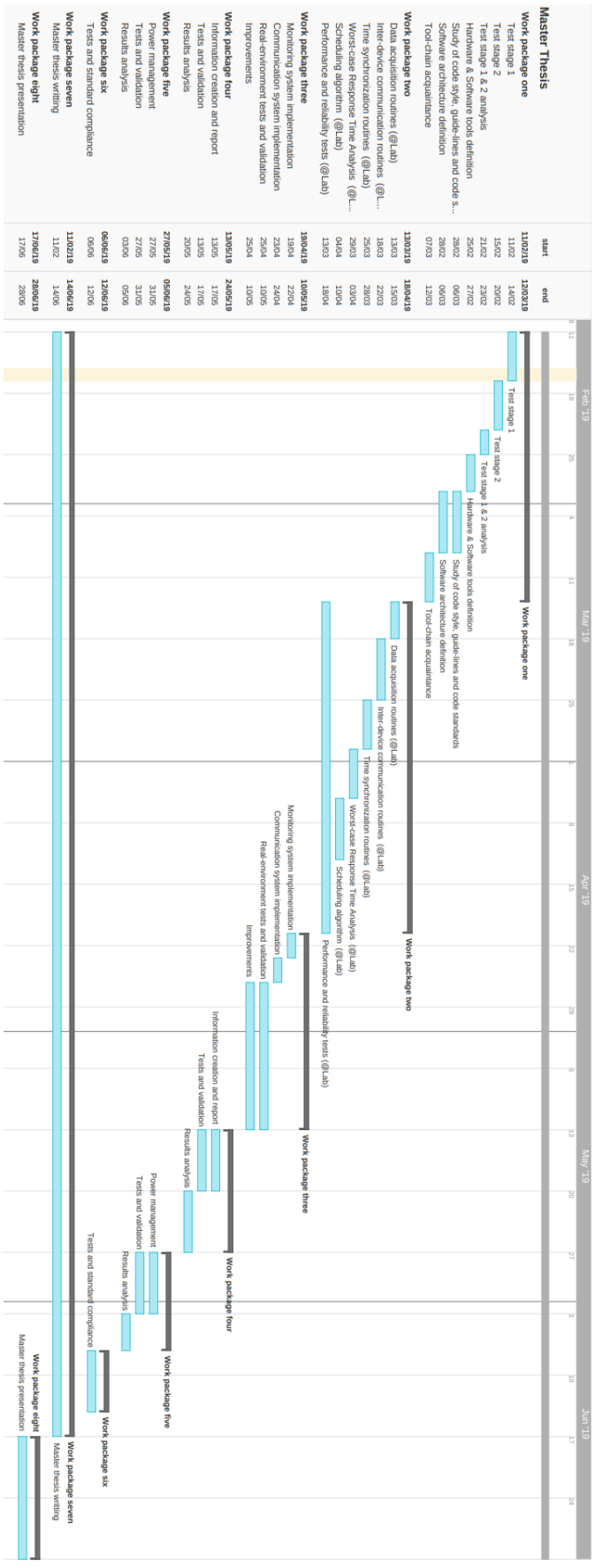


Figure 5.1: Work Planning - Gantt Chart

Once the production of the prototypes is finished, a mechanical validation test stage and its correction period begins. In order for real environment tests to take place, it is necessary to build and install a concrete base that allows to fix the modules onto the pavement. The installation process also requires the connection of the entire structure to the local water drainage system. Once a functional structure has been reached, tests related to the impact of the installation in the real environment will be carried out, such as the necessary regulation regarding sound limits in urban surroundings.

Taking into account that the period of execution of this work coincides with months associated with rainy conditions, the tests may be run continuously in real environment only after validating the correct isolation of each unit with respect to permeability indices. Otherwise, not only the hardware safety (related to the monitoring system) becomes compromised, but also the entire metal structure is exposed to the presence of corrosive agents. If not possible to conduct the test stage under the ideal conditions, it is necessary to install and remove the units each time a test period is scheduled.

Order	Work-Package
1	Hardware and software tools definition.
2	Tool-chain setup and acquaintance.
3	Implement an I^2C long-range test-bench.
4	Evaluate the I^2C long-range performance.
5	Define the test stages and sensors project integration.
6	Software architecture definition.
7	Monitoring and communication system development (test-stage oriented).
8	Lab tests.
9	Real-environment tests and validation.
10	Monitoring and comunication system development (Pavnext RPEH oriented).
11	Tests and validation.
12	Performance analysis and improvements.
0	Dissertation writing.

Table 5.1: Work-packages definition

5.2 Project Management

Selecting the tool-chain to work with, is one of the first tasks to perform once the problem definition is terminated.

Hardware selection and the software approach must be defined concurrently and both were briefly addressed in the previous chapters. Free tools play an important role for the immediate cost of a solution; although, long-term wise, the support and stability provided by commercial solutions might save the time to port to another tool-chain.

The tool-chain setup description is left to the implementation chapter (chapter 6), although the procedure to select them is illustrated in figure 5.2.

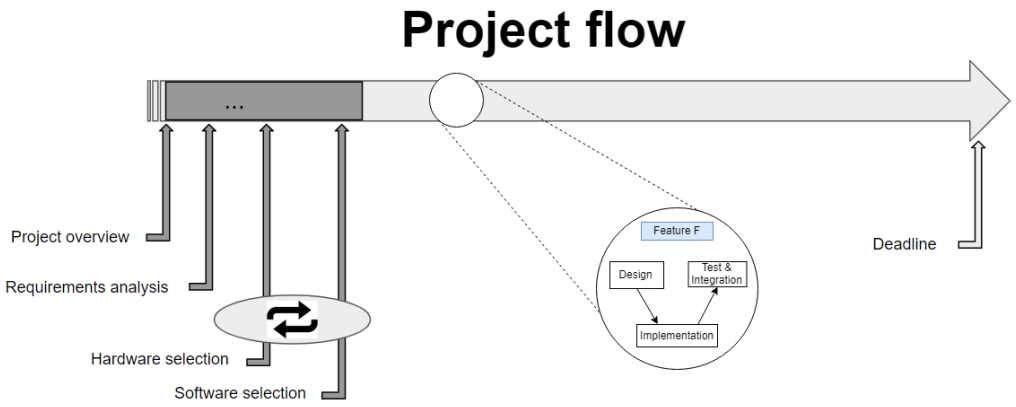


Figure 5.2: Tool selection within the project flow

As the solution's core is based on the data-acquisition and data-transmission tasks, these are the two tasks to be validated in the first place. Once achieved a stable solution for data acquisition and transmission between micro-controllers, the concept is validated. However, the process of integrating the developed system into any system that needs to be monitored must be tailored for the target application: its constraints and requirements. With regard to the Pavnext RPEH system, the sensors placement and installation is not planned, therefore it will be necessary to design, build and install the physical support for each sensor and also to connect them to the micro-controller.

5.3 Risk Management

Promoting a discussion of different system architectures, each carrying their own characteristics to solve the monitoring and communication system problem, raises the possibility to differentiate several layers of complexity. Such approach enables to chose the solution that fits the problem, and assures that each solution fits the whole system design.

Upon definition of the software approach, code style guide-lines to promote a solid and self-defensive development will be used. Also related to the software development process, the means to achieve a proficient debug techniques could determine the success of a more or less robust solution. Embedded systems debugging using the traditional academical methods, such as serial prints or output manipulation using an LED, are way less powerful than professional debugging tools, although, it requires a small setup procedure. Moreover, maintenance of a bug database and the act of not implementing new features on top of known bugs, promote robustness and design faults to be found sooner, when complexity

levels are smaller and the debug process may address a bug at a time. A software version control will also carry several benefits that allow features to be developed independently, so that complexity increase is set on top of functional solid grounds. Code repository brings a higher level of security, so the developed work keeps safe from any destructive occurrence that could happen to the development devices. Long-term projects, specially for novice level, are prone to suffer deviations from the objectives if the process and path designed is not well defined. Work-packages definition combined with milestones distributed along the project timeline, will prevent the loss of awareness and direction.

Chapter 6

Project Development

6.1 Tool-chain setup

STMicroelectronics' STM32F103 family of 32-bit ARM Cortex-M3 core-based microcontrollers is supported by a wide range of software integrated development environments (IDEs) with C and C++ support, plus debuggers from major third-parties that are complemented by tools from ST, allowing to configure and initialize the MCU or monitor its behavior in run time. After experimenting the free licence eclipse based Atollic TrueSTUDIO and AC6 System Workbench for STM32 (SW4STM32) IDEs, both offer the same interface organization and all-in-all the same features. Whereas SW4STM32 IDE allows the integration of Eclipse plug-ins, Atollic TrueSTUDIO has a set of tools pre-defined on the installation package. Both use GCC C/C++ compiler, GDB debugger and support for the ST-Link. As the Atollic product is now being replaced by the STMCubeIDE, AC6s SW4STM32 IDE is the selected IDE.

STMCube is an ST product to ease the development process, providing a graphical user interface to configure the peripherals. It also provides code generation utility, where the source code to initialize and use the selected peripherals is exported as an SW4STM32 project. Two main drivers are available: HAL (Hardware Abstraction Layer) and LL (Low Layer) [11].

After a few tests and navigation through the several source files created by the code-generation tool, it is noticeable the effort behind an hardware abstraction layer to support the whole STM32 portfolio. The HAL APIs are feature-oriented and hide much information. The Low-Layer drivers offer hardware services based on STM32 peripherals, reflecting the hardware capabilities. Selecting the Low-Layer driver results on a lower level of abstraction, but eases the process of following the STM32F103 data-sheet [12] and its reference

manual [12].

A Git-hub cloud repository was created, featuring two main projects - Slave and Master. As the project evolved, the master and slave projects carried a series of forks from the main API reference. Resorting to the Mantis bug-tracker, a bug database to keep the development process organized and aware of the bugs found, it is possible to defined the reproducible scenarios for a bug (if a reproducible pattern is found), to include snapshots of either the oscilloscope or debugger console and classify the occurrence: enabling to decide if it is possible to proceed or intervention is required before any further development stage takes place.

The oscilloscope available at Pavnext is a digital Pico 3000 series, which only has a stable software version for Windows OS. After using the logic analyzer feature to decode I^2C frames, unexpected glitches and shutdown occurs (running on Kubuntu 18.04 LTS). The PicoScope official website states that Linux builds are not formally tested and should be considered of beta quality.

As the whole tool-chain also has support for Windows OS, it was decided to change the development environment to the Windows OS.

6.2 I^2C Long-Range Test

The test setup is designed to validate I^2C as the communication protocol for a long-range configuration. Targeting the long-range test scope to the maximum length of a Pavnext RPEH installation, the bus length used for the lab test is twice as high, as a precaution against factors that only a field test may reveal.

As previously defined, the micro-controllers will be placed inside the units' case. Following system architecture 1 configuration, the insulation case is compromised twice. Although, system architecture 2 and 3 require only 1 entry-point and shorter bus. Revisiting the system architecture 1 configuration, it might be improved by adopting the cabling disposition issued by SA2/SA3. In order to evaluate the I^2C long-range viability, the laboratory test-stage follows the setup represented in figure 6.1.

The STM32 board offers the possibility to select internal pull-up and pull-down resistors. With the intent of making the connection independent of the board configuration/selection but also due to the fact that the internal pull-up resistor set on the STM32F103CBT6 is too weak (typical value is 40k Ohm - [13], table 35), the application designed uses an external pull-up resistor.

ST Microelectronics' application note AN4899 [14] addresses the pull-up calculation

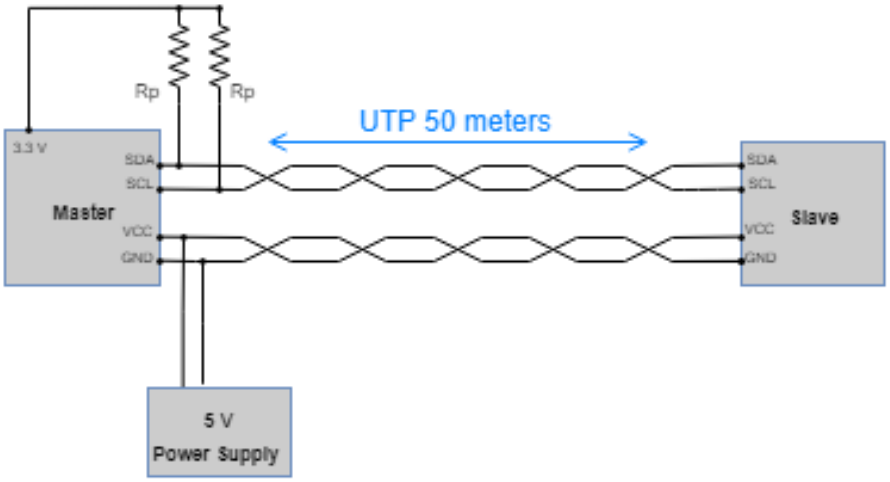


Figure 6.1: I^2C Long-range Test Setup

for general GPIOs use. However, output and input levels compatibility and the way to calculate the appropriate pull-up resistor for the STM32F103 GPIO open-drain output when connected to an external device, is different in case of I^2C bus. The main reason is because the I^2C mode must be considered, and the calculation depends on both the devices V_{DD} and the bus capacitance.

A low pull-up resistor value prevents I^2C pins to generate a low level on the bus and the minimum value is defined by equation 6.1.

$$R_{PUmin} = \frac{(V_{DD} - V_{OLmax})}{I_{OL}} \tag{6.1}$$

The bus capacitance (C_b) defines the maximum pull-up resistance, limited by the I^2C rise-time (t_r) specifications - 1000 nanoseconds for Standard Mode [9]. Once released by a, master/slave device, the I^2C line may not rise to a logical high before it is pulled low, if the the pull-up resistor value is too high. The maximum value for the pull-up resistor is calculated by equation 6.2.

$$R_{PUmax} = \frac{t_{r_{max}}}{(0.8473 \times C_b)} \tag{6.2}$$

In case of communication exchange, master/slave output signals must be compatible with the V_{IL} and V_{IH} levels of the receiver device, whereas inputs must be compliant with the V_{OL} and V_{OH} levels of the transmitter device - figure 6.2.

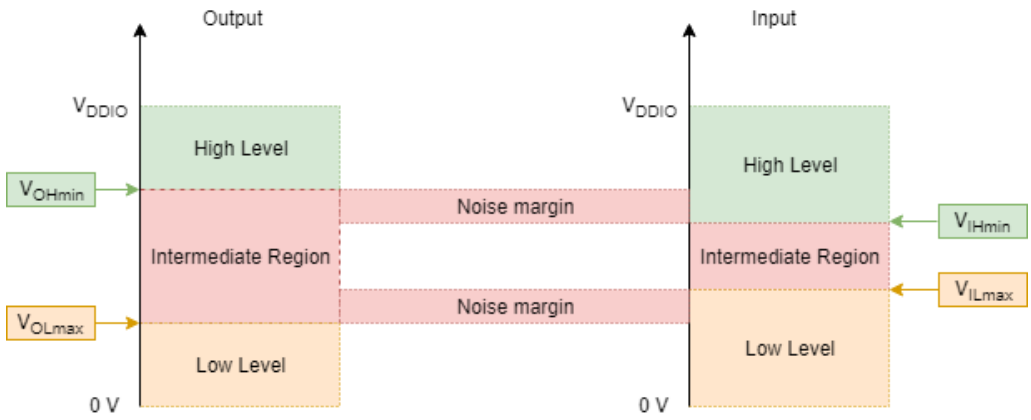


Figure 6.2: Logical level compatibility

For this test, the micro-controllers are powered using a 5V power supply and the micro-controllers internal regulator sets V_{DD} to 3.3V. The STM32F103CBT6 I^2C pins are 5V tolerant and can be directly connected directly to devices running at 5V. However, under these circumstances it is necessary to consider that a current is injected into each of these pins, due to the external input voltage V_{IN} being higher than V_{DD}. According to the micro-controller datasheet [13], the injected currents can not exceed 5mA.

Table 6.1 addresses $V_{IH_{min}}$, $V_{IL_{max}}$, $V_{OL_{max}}$ and $V_{OH_{min}}$ values, according to STMs application note 4899 [14] and STM32F103CBT6 datasheet.

Symbol	Voltage Level
$V_{IH_{min}}$	(2/3)*V_{DD}
$V_{IL_{max}}$	(1/3)*V_{DD}
$V_{OH_{min}}$	V_{DD} - 0.4
$V_{OL_{max}}$	0.4

Table 6.1: Micro-controller logical level compatibility

The UTP cable used for the test is slightly over 50 meters and it has a rated capacitance of 3pF/m. To safeguard the device correct operation and integrity, the pull-up resistors must be greater than 1k ohm, otherwise the injected current can lead to micro-controller malfunction or permanent damage. Since the implementation is error-prone, if one of the 5V power cables is connected to one of the I^2C pins, the 1k ohm resistors must be considered as an effective minimum value.

STMicroeletronics' application note AN2824 [12] addresses the I^2C firmware implementation, by use of polling, interrupts or direct-memory-access. In this case both the master and the slaves manage the I^2C peripheral by means of interrupts. Figure 6.3 shows

the SDA and SCL signals on the 50 meter unshielded twisted-pair (UTP) bus during a master operation performed after power-up. The master is configured to search for devices on the I²C bus, by performing a master-transmitter operation to all the 127 possible unique addresses.

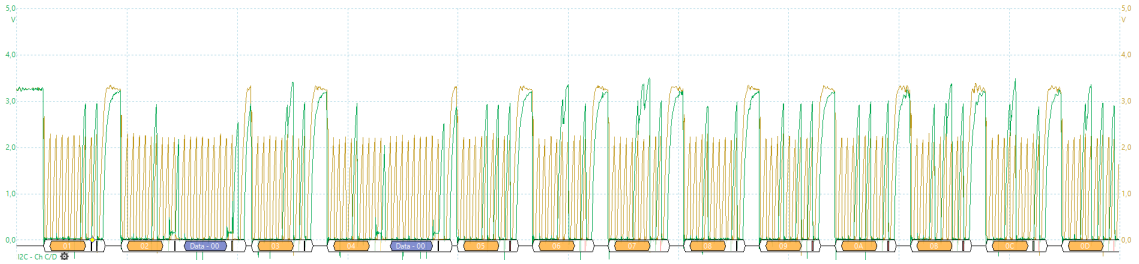


Figure 6.3: I²C SDA and SCL signals 50 meter UTP bus

The test results reveal that the master is able to successfully find the 2 slaves on the bus, which in turn acknowledge both the address and master-write operation.

To further evaluate the performance, the master is configured to send 256 bytes to each slave, from 0 to 255. Every time a slave non-acknowledges a byte reception, an interrupt is fired. In turn, the master is configured to:

- clear the *NACK* register flag;
- re-transmit the non-acknowledged byte.

Each slave receives the data bytes until a stop condition is detected and check if the received values match the expected pattern - 256 bytes, from 0 up to 255.

Concluded the first-stage, the master proceeds to the second test-stage, performing a master-receiver operation. In this stage, each slave act as a transmitter and send the same pattern to the master until a stop condition is issued. Whenever the slave-transmitter identifies a non-acknowledge issued by the receiver, it performs exactly like the master:

- clear the *NACK* register flag;
- re-transmit the non-acknowledged byte.

Using the USART to write to the serial terminal, both master and slaves write the number of error occurrences after 1000 iterations. The error occurrence is defined as the master/slave not receiving the expected 256 bytes (from 0 to 255) and a byte reception is only considered if acknowledged.

The results indicate that 5,06 percent of the transfers are not what the receiver (slave or master) expects. Repeating the test for 10000 iterations, under the same conditions, results on 4,91 percent failed transmissions. Table 6.2 summarizes the results, by performing the same test-pattern over different bus lengths, all using the minimum 1k ohm pull-up resistor.

Bus Length [m]	**Pull-up Resistors [Ω]**	**Failed Occurrences (%)**
20	1000	0,39
50	1000	4,91
70	1000	100

Table 6.2: Failed receptions over 20, 50 and 70 meters UTP I^2C bus

Performing the same test using higher pull-up resistors increases the rise-time and decreases the data-communication system capability to reach the high logical level, since it is pulled-low before reaching V_{IHmin}. Distributing the 1k ohm pull-up by placing the corresponding pull-up resistors closer to each of the slaves does not decrease the failed occurrences count.

Figure 6.4 represents the SDA and SCL signals for a general address call, on a 70 meter bus, where no slaves acknowledged.

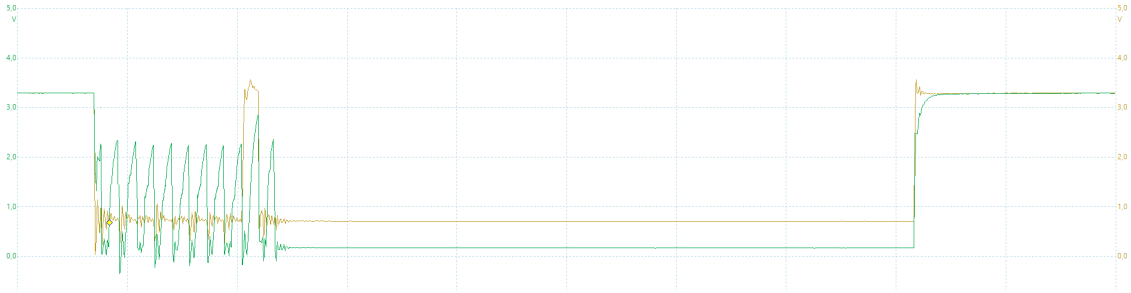


Figure 6.4: I^2C SDA and SCL signals - 70 meter UTP bus

As preliminary conclusions, the proposed solution for the communication protocol may proceed, given that the micro-controllers are capable to perform data exchange over a bus twice as long as the RPEH limit implementation scenario. The results also reveal that an error-detection mechanism should be implemented, as the registered error occurrence for an I^2C bus test that emulates non-ideal conditions (emulated by the bus extension) fails to perform 100% successful transfers.

6.3 Data Acquisition Test

In order to simultaneously validate the slave data acquisition operation, and that the master correctly receives the acquired data, a known signal is given to the slaves. Using a waveform-generator, a rectified sinusoidal wave with an equal period to the vehicles crossing the units surface at 20, 50 and 80 Km/h is the defined test.

With regard to the slave, the data-acquisition process executes every millisecond, and together with the acquired value, a time-stamp is saved. A hardware timer is responsible for generating the periodic 1 millisecond interrupt and before starting the ADC conversion, a queue receives an unsigned 16-bit integer, representing the time in milliseconds. After the queue-write operation, the register that starts the ADC conversion is set. Configured to use one of the DMA channels available, the resulting 12-bit value is written to a predefined memory zone. Concluded the DMA write operation, an interrupt is generated indicating the end-of-conversion. In this interrupt, the acquired value is in turn sent to a data-queue, along with the time-stamp value written right before the ADC conversion started.

Besides the interrupt driven operation, the slave also has 2 FreeRTOS tasks, being the predefined IDLE task and the highest priority that is set to be executed every 5 milliseconds. The 5 milliseconds tasks removes the values from the queue and filters them. The filtering process is done in such a way that consecutive samples that differ from each other less than 136 (each sample is 12-bits, stored in a half-word) are discarded. Non-discarded samples are sent to a queue, waiting for the master to perform a read operation.

On the master side a FreeRTOS task is configured to perform every second, asking the slave the number of samples that it has stored on the queue. The process requires the master to send a command that is interpreted by the slave to load its transmission buffer with the number of bytes stored in the queue. After the master-write, a master-read is performed, so that it receives the number of samples that the slave has stored. Knowing the number of samples stored by the slave, the master proceeds with the master-write and master-receive operations, by firstly executing a master-write operation followed by a 2 byte command:

1. byte interpreted by the slave to load the stored values into the transmission buffer;
2. byte indicating how many samples the master wants the slave to transmit.

Once the data-samples transmission is completed, the FreeRTOS task yields, so that the a lower-priority task retrieves the received values from the master reception buffer, to be sent to a serial terminal through the USART. It is possible to verify that the slave sampled the data, filtered consecutive samples following the filter criterion and along with

the analog sampled values the respective time-stamp. As the graphical interface is not yet defined, the data is parsed to the *.csv* format, which results are shown in the following figures.

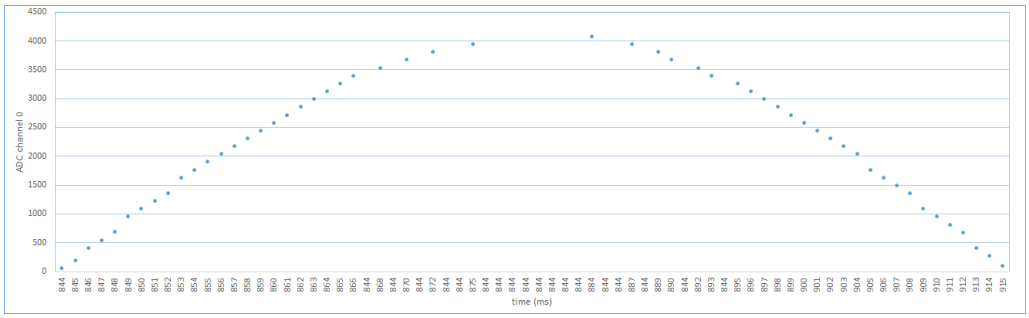


Figure 6.5: 7 Hz signal on ADC channel 0

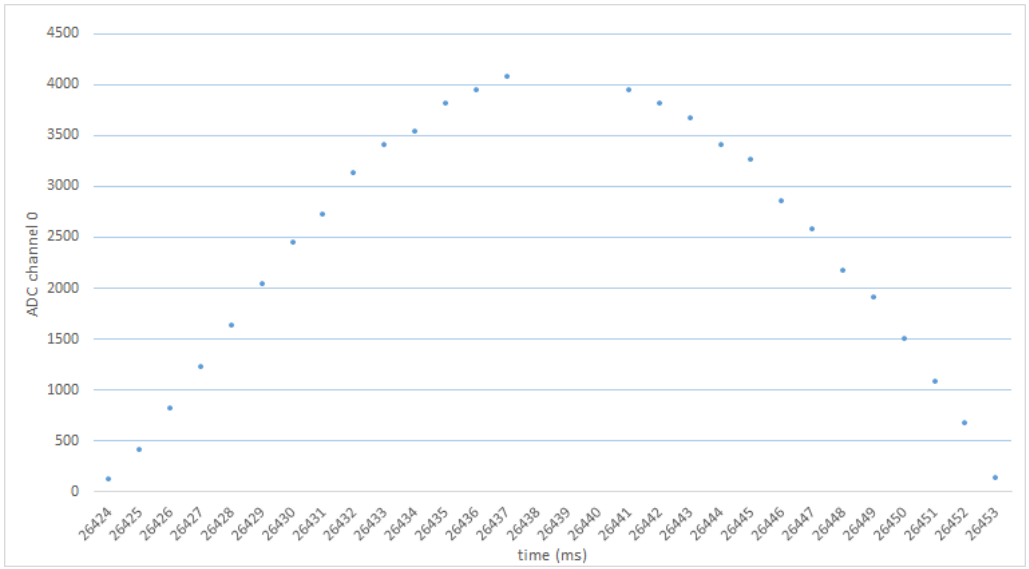


Figure 6.6: 17 Hz signal on ADC channel 0

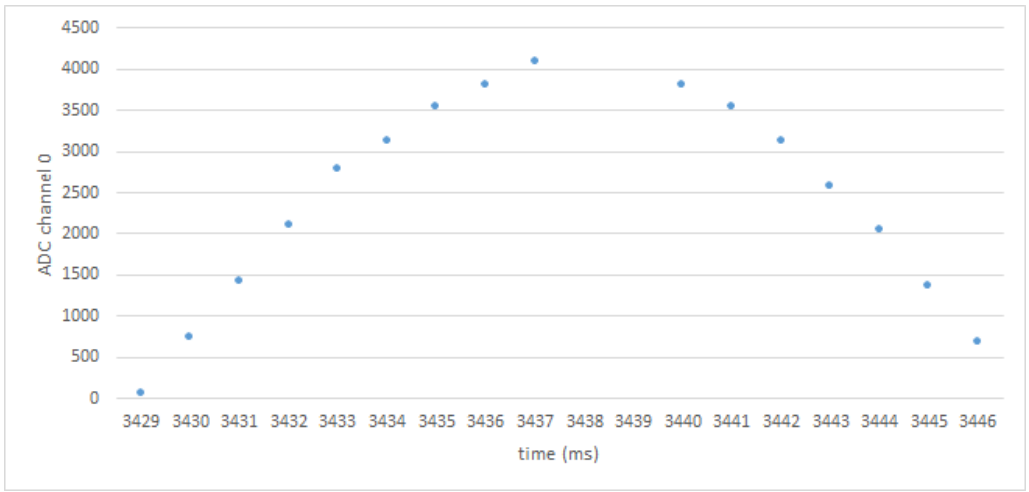


Figure 6.7: 28 Hz signal on ADC channel 0

6.4 Time Synchronization

In order to implement a micro-controller synchronization mechanism, an asymmetric mechanism is used. Each of the slaves receives 2 bytes from the master. The sending of the 2 bytes is done using the address 0x00, to which all slaves should give acknowledge. Since this procedure does not allow the master to know if one of the slaves actually received the bytes sent, each slave must be able to continue the time-stamp process. Furthermore, to minimize the impact on the I^2C bus, each of the slaves is only aware of the time in milliseconds, using 2 bytes to represent it. Following the time-stamp process described in the previous section, each slave is now configured to always update its own time value once the general-call and respective 2 bytes are received. In case the slaves are ahead of the time received, they do not delay their clock. Currently, the system is only informative and therefore there is no control processes based on the values acquired, although time does not go backwards. Thus, in case of being advanced, each of the slaves will average the samples acquired between the time interval corresponding to the time lag between the current time of the slave and the time received.

To test this mechanism, the master switches 2 digital outputs between 0 and 3.3 V every 50 milliseconds, with mutually negated values, each connected to a single slave. Every second the master starts the synchronization process, followed by data collection of the slaves, where they return the time instant relative to each low-high transition during the previous second.

The result of this test reveals that there is no jitter in the communication (with enough impact to be noticed in the defined milisecond time base), although there is jitter in the reception and update processing of the received time value. A maximum mismatch of 3 milliseconds is recorded between the slaves.

6.5 Sensor Integration and Assembling

To monitor the vertical and horizontal displacements, three different linear-potentiometers were available - figure 6.8. Sensor a) from figure 6.8 is the most fragile (10 thousand full cycles), but also the cheapest; sensor c) from the same figure is the most robust (10 million full-cycles), but requires the top/base to be drilled and internally threaded. Sensor b) is spring based and features as an intermediate option both in price and robustness (1 million full cycles, 5 million dither cycles).

Only the sensors a) of figure 6.8 are available in sufficient quantity to carry out a complete installation. For their placement, the following support was designed (figure 6.9),

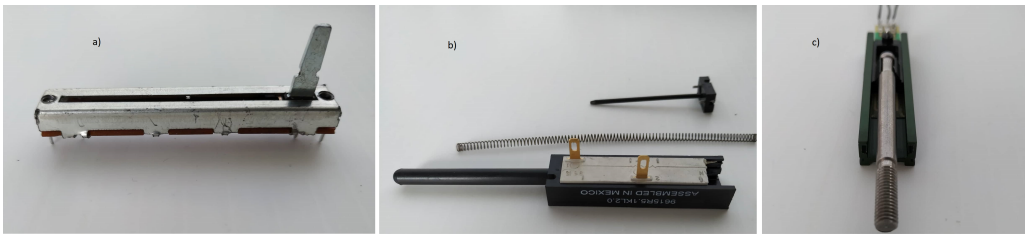


Figure 6.8: Linear Potentiometers Available

built and installed, as shown in figure 6.10. To acquire the vertical/horizontal displacements, units carrying an electric generator are set to have 4 linear-potentiomenters - 2 at each end (placed vertically) and other 2 placed horizontally on the bottom surface, connected to the moving plates and 2 hall sensors. Coupled to the generators inertia-wheel, a magnet is used to trigger the hall sensor, so that revolutions per minute may be computed.

Units M100 and M140 have 6 linear position sensors, all vertically mounted: 1 in each corner and the other 2 at the center - figure 6.10. As the prototype units design does not contemplate sensors installation and their placement, the routing options are limited. In figures 6.10 and 6.11 it is possible to verify the sensors installation performed and also realize that the wiring arrangement does not avoid moving parts.

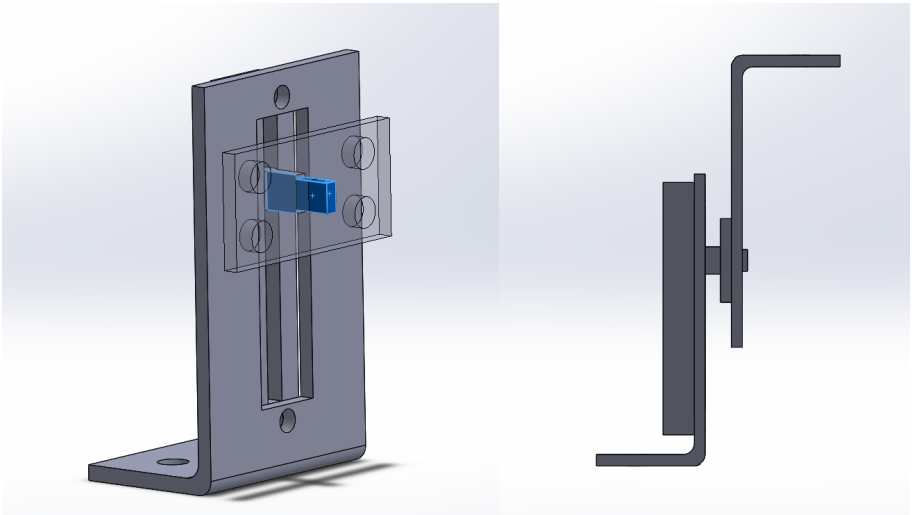


Figure 6.9: Linear Vertical Displacement Sensors Support

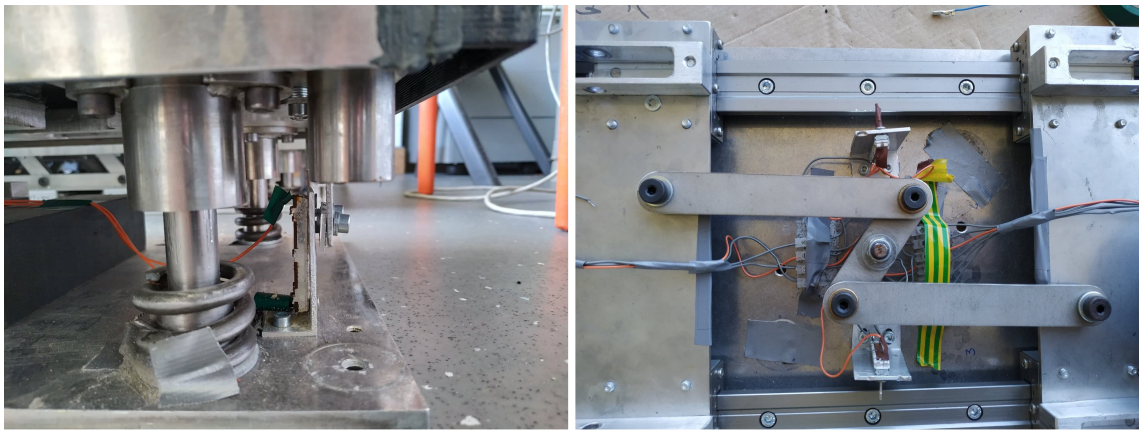


Figure 6.10: M100 unit: linear position sensors vertically mounted

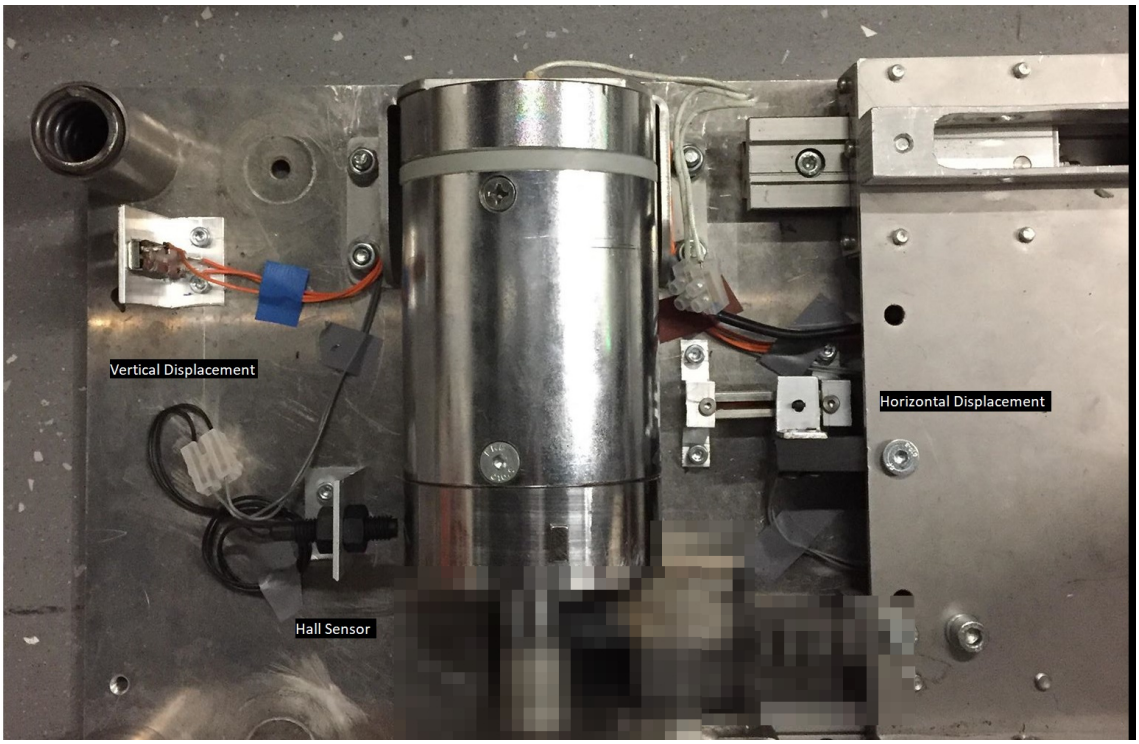


Figure 6.11: G140 unit - Sensors disposition

6.6 Data Logging

For all units it is necessary to measure the vertical/horizontal displacement. Besides the hall sensor, the power electronics project also under development, addresses two alternatives for a first stage of energy storage within the unit. The circuitry is designed to include

two voltage outputs, which range from 0 to 3.3V, representing current and voltage sensor outputs.

The objective is to monitor the evolution over time of these parameters, allowing to extract the data from each of the units and visualize them in a graphical interface. As a result, the system is composed by 4 data-loggers and a central unit that collects and sends the received data over the USART to the graphical interface.

The mechanical system is sized for a maximum (non-continuous) displacement speed of 4 meters per second, which corresponds to a mechanical limit by design that is not reproducible in the lab tests. The maximum displacement is 30 millimeters and assuming a continuous 4 meters per second, each millimeter is traversed every 250 microseconds. In the required decision making for the sampling frequency of the ADC, a sampling period 4 times higher is initially considered for the lab-tests.

In order for the master to be able to collect data on the I^2C bus, the amount of data produced in a time interval must not exceed the amount of data that can be extracted. For the 12-bit resolution ADC of the micro-controller, if each slave samples at a rate of 1KHz, 1500 bytes will be created per linear displacement sensor. For instance, only for the spring units, M140 and M100, this corresponds to 18000 bytes. At a frequency of 100KHz, even excluding the address frame, the amount of data-bytes that can be withdrawn per second is exceeded. This approach would have the benefit that it would not be necessary to include a time-stamp to the data, since the sampling rate to which they are sampled is known.

In order to make it possible for the produced data to be read in full, each slave may sample at 1KHz rate, but only store values that present relevant information. That is, there is no relevant meaning in acquiring values from slaves when the system is not being solicited. A simple approach is to define a value for which the system awakens, and when below that value no samples are stored. However, since the system lacks parameterization, there is an interest in monitoring the entire scale. For instance, it is possible that right after a vehicles passage, the units upward movement momentarily exceeds the initial position.

For the volume of data produced by each slave to be sent to the communication system, without losing the entire scale of values, each sample periodically acquired is filtered, comparing it with the immediately previous one. If no more than a given quantity differs between the samples, the newer sample is discarded. The amount required for two consecutive samples to be stored is set to 1 millimeter, where the full range is 30 millimeters. For the 12-bit resolution provided by the micro-controllers ADC, 1 millimeter corresponds to a value of 136.

However, this means that the collected data may not be temporally spaced for 1 millisecond and the approach requires to assign the time at which the value was acquired. On the other hand, acquiring the number of revolutions per minute of the generator does not need to have a time associated with each revolution, but rather the number of revolutions that have been executed in a certain period of time.

All slaves inside a unit have their own awareness of time, making it necessary to synchronize them. Even if every slave shares hardware architecture, running at the same clock frequency, exposed to the same temperature, both drift and skew phenomenons are present. The adopted procedure aims at an asymmetric synchronization mechanism, where the master using the general call address, is the only one which spreads its time view so the rest have to synchronize to it.

Concerning cost-effectiveness, asymmetric schemes favor low-speed communication systems when compared to symmetric schemes, which require a high number of messages exchange in a short time interval, and this overhead can overwhelm the capacity of the low-speed I^2C bus for the long range data logging purpose.

6.6.1 Slaves M140 and M100

M140 and M100 slaves share sensors configuration, all of which are 10k ohm linear displacement sensors, positioned in the same arrangement. For data logging purposes, and taking into account the micro-controllers ADC acquisition modes, multiple channel scan in single conversion mode is the one that fits the arrangement. This mode allows to configure channel sequences to be read successively, with different sampling times, thus allowing to save CPU load and the necessity to stop the ADC during conversion process to reconfigure the next channel [15].

The maximum external impedance allowed for an error below 1/4 of LSB is defined by the external impedance and referring to the STM32F103 datasheet [12], for 10k ohm the ADC should be configured to perform at least 12 cycles (equation 6.3).

$$T_s > 14 \times 10^6 \times 8 \times 10^{-12} \times ln(2^{(12+2)}) \times (10+1) \times 10^3 \tag{6.3}$$

As with interrupts, Direct Memory Access (DMA) performs data transfers without having to continuously poll. Combined with multiple channel in single conversion, an End-of-Conversion (EOC) interrupt is set once the samples are written to the memory addresses defined. After the EOC is detected, the samples are time-stamped and ready to be filtered. Each ADC channel has its own queue for data to be stored and if two consecutive values

differ by more than 136, then both data and corresponding time-stamp are stored into the corresponding queue.

This task defines the creation of information at the local level of each of the slaves, which also have a mechanism to make it available to the master. Each of the slaves has a unique 7-bit address among all of the network and can still acknowledge the general address call. A specific protocol defines a series of commands that can be interpreted by each of the slaves, depending on the characteristics that it has. These characteristics are shared with the master after the power up, in order to allow the master to dynamically adapt to each slave.

6.6.2 Slaves G140 and G100

Besides the 10k ohm linear displacement sensors which for data logging purposes share the same configuration previously addressed for slaves M140 and M100, G140 and G100 also have the same procedure after power up, which allows the master to know the slave configuration. Slaves within electrical energy producing units, interface with an hall sensor mounted at each generator. Each time the magnet passes by the hall sensor, an interrupt is fired and updates a counter. Every time the master reads the slaves data, the counter value is cleared.

6.6.3 Communication Protocol

In order for the master to know the data he can request from each slave, he needs to know the data they produce. It can be done in hardcoded form, or dynamically adjusted in the power-up process. The potential benefit is that after installing a unit and it needs to be replaced, it is not necessary to reconfigure the master. For this procedure it is necessary for the master to know the different combinations possible, but once defined, the configuration process becomes automatic.

All slaves have the following parameters defined:

- micro-controller ID;
- units model ID (G140, G100, M140, M100);
- units position;
- how many sensors are insalled.

Based on the number of sensors, the slave data structure has:

- sensor ID;
- sensors position within the unit;
- number of resolution bits per sample.

On the master side, it is required to know how many slaves it can have at most. After the power up, the master blocks for 5 seconds, and elapsed the period it starts scanning the I^2C bus. As soon as a slave is discovered, the master saves the address. When the number of expected slaves is found the master starts the configuration process. If the 127 addresses have been traversed and the master has not found the expected number of slaves, then repeats the discovery operation without addressing the slaves already found. If still have not found the number of slaves that are expected, the master generates an alert - this operation is intended for integration with the graphical user interface. Once the alert is generated, the master continues the configuration operation, in which it queries each of the slaves found, to send the above-mentioned parameters by means of master-writer followed by master-reader operations.

As a shortcut to the integration of the system to be developed in the future, in which pavement-lights are to be installed, the master reorders the address vector found in the position order. In this way, it is simple to traverse the slaves address vector in order to first request data from the slaves that are the first to be actuated, decreasing the time between a system actuation and the pavement-lights control.

Then, the continuous process is based on the periodic master calls to each of the slaves, beginning with a master-write message, where the frame consists on the slaves 7-bit address and a write bit, followed by the command bytes.

6.6.4 Master

For the data logger configuration associated with development support, in which the master has the role of reading the data produced by each of the slaves and making them available for interpretation, two tasks are defined: fetch data from the slaves and store it to further transmission.

After the process of discovering and configuring slaves, the master knows how many sensors each slave has and the number of bits per sample. The process of reading data from slaves is thus started and is executed every 1 second. For each slave, the master-transmitter queries how many bytes the queue of a particular sensor has, followed by a master-receiver process to receive the data. Based on the response, the master may request to read all the stored data, or just a portion of it, wich is interperted by the slave using a third command byte.

On the master side, a maximum queue number that each slave can contain is statically defined, and if a slave contains fewer sensors than the maximum value, this memory space is not being used. Dynamic allocation would benefit memory management efficiency, but would make it more difficult to perform static analysis of resource usage at compile time. Thus, at the end of the reading of each sensor in each of the slaves, the data contained in the dedicated receiver buffer I^2C are placed in the queue that the master has allocated for each sensor of each slave.

For this whole process the master defines a timeout period, initializing a hardware counter before the transmission. A transmission ends with the interrupt trigger associated with the end of a master-read or master-write message, with the non-acknowledge interrupt, or the expiration of the defined timeout period.

When master tries to read the data stored on the slaves, each has two attempts. When the two attempts are used and the master was not able to read the data or even communicate with the slave, it proceeds to the remaining slaves. A message report is created and stored for transmission to the graphical interface, emulating and alarm for maintenance, whenever a slave misses to communicate for 3 slaves read cycles (3 seconds). At the time there is no definition regarding communication with the master micro-controller by interpreting commands sent by the graphical interface, although two possibilities are defined:

- the master keeps trying to communicate with the slave, after reporting it does not answer the address call;
- the master stops trying to communicate with the slave, after reporting it does not answer the address call.

Both possibilities are implemented to allow future implementation of the system to behave as the higher hierarchical layer defines.

Chapter 7

Results and Conclusions

The I^2C long-distance tests reveals that for 1k Ohm pull-up resistors, the communication is guaranteed. The I^2C bus capacitance will always depend on the cable used and its length, plus the number of micro-controllers (40pF per micro-controller). Since the test was performed with a bus extension 2 times longer than a maximum length installation, the solution is feasible.

Placing the pull-up resistors only on the master side, has the benefit of allowing any intervention on the bus (bus length increase or replacement) to only require access to one micro-controller, avoiding to disassemble/open each of the units. If during the I^2C test stage the communication to one of the slaves was defective, it could be attempted to distribute the pull-up resistors by the bus, placing it closer to slaves, but besides unnecessary, it would imply to configure every installation according to the number of slaves. Such a situation would not only be error-prone, as it would increase management/planning overhead. Thus, by configuring all installations for a pull-up value of 1k Ohm, homogeneity and safety are ensured in case of a wrongly connected cable from the 5V power supply to one of the I^2C pins.

The fact that each of the micro-controllers is powered at 5V is only due to the fact that it is convenient to use a commercial power bank. If each of the micro-controllers placed inside the units G can be locally fed by the energy produced, it may be necessary to electrically isolate them, not only from the power circuit, but also from the communication bus due to different reference levels.

During the development process, the prototypes were used for several mechanical integrity tests. Some of the tests produced destructive results and consequently maintenance periods restrained the integration of the monitoring system.

The G140 and G100 units presented a mechanical lock issue when exposed to vehicles

load, and it was not possible to test them during the period in which this work was carried out. The M units have been the target of laboratory tests, however in this environment there is no rigorous means to emulate the real environment.

Target of a structural change, the G units have been reconfigured to M-type units. The process of installing the monitoring and communication system in the prototypes is error-prone and cables are either cut-off or disconnected. In laboratory tests the modules are tested without the isolation case, allowing that the connections are rectified whenever necessary. Before conducting field tests, the units are closed and the connections are tested before being transported to the private tests facilities. After transportation, the units are installed and fixed to the ground. Performing the wiring and proceeding with the same process done at the lab, the master was not able to successfully discover the slaves. After successive failed attempts and refinements to the process, either the communication or power conductors are disconnected. Since the unit are closed it is not possible to intervene, being necessary to remove the units from the pavement and transport them back to the laboratory.

The results obtained do not go beyond the laboratory emulation scenario. In this context it was possible to exploit the management of the pavement lighting system. Promoted by the services of the FreeRTOS kernel, a simple task integration results in a periodic task that defines the light management in two alternatives:

- Basic mode: to interleave an on-off period every 500 milliseconds;
- Interactive mode: The *on* period is only started when the master checks that one of the slaves has been actuated.

For the interactive mode, the *on* period lasts for 5 seconds, in which one of the masters output is set to generate a PWM with duty cycle proportional to an LDR read.

As the energy management project started, it was requested to port the master firmware to another micro-controller (the XMC4500, an ARM Cortex-M4 MCU), due to the DSP features that were found necessary.

Monitoring the currently built prototypes is a task that faces several hardware installation challenges, given the nature and intent to exploit them primarily from the mechanical point of view. The lack of space resulting from not contemplating installation of a monitoring system, the current prototype design reveals that future models should exploit volumetric expansion within the units. In fact, the design of the units has been redesigned and future models will be 5 centimeters longer.

Mechanical performance has yet to be improved so that testing can proceed smoothly. The presence of mechanical interlocks prevents successive tests from taking place without the maintenance of the unit structure.

Due to the scheduling of real-time tasks and the consequent need to activate the clock stretching mechanism (so that higher priority tasks are firstly attended), the worst case recorded for the clock signal frequency is 89 KHz.

Although in the absence of control loops, the work done in the context of RTOS and the maximum recorded jitter of 3 milliseconds between distributed system microcontrollers can confidently provide a solid basis for implementing control layers.

Chapter 8

Future Work

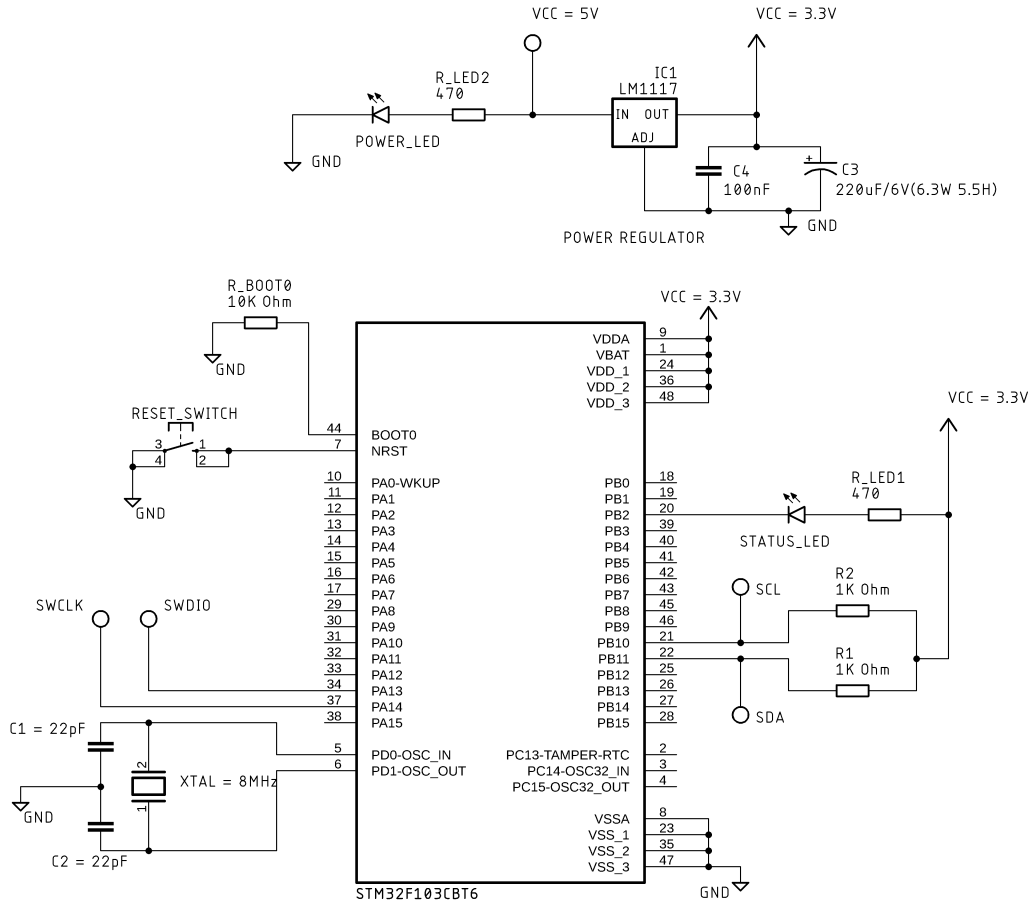
The result of this work promotes the possibility of integrating the remaining sensors that will allow to monitor the behavior of the system described in chapter 2. The developed low-level communication system, together with the availability of the data acquired by each of the units to be displayed on the graphical interface, will facilitate the course of the tests and the analysis of the data. Once the local monitoring operation is validated, the project can evolve to remote monitoring. The fact that the port for the infineon micro-controller (XMC4500) was made, along with the peripherals already available on the development board - Ethernet adapter, will speed up the process.

Once the power generation, regulation and storage project are completed, the system can be integrated into the monitoring and communication system developed, facilitated by FreeRTOS kernel resources. Having completed the entire system and reaching a stable version, then the behavior of the system in time regime can be evaluated and ensure that it meets the deadlines.

The new unit's model designed for the pilot version will eventually be the reason for reformulations and will certainly begin a new learning stage, however it is expected that this will evolve faster and make use of the resources hitherto developed for the same purpose.

Appendix A

Electric Schematic



Note: The slave schematic has the same configuration, with only one exception: it does not have pull-up resistors.

Responsible dept.	Technical Reference	Document type: A4 Schematic drawing	Document status: Released
U.PORTO Faculdade de Engenharia	Created by: Wilson C. Approved by: Wilson C.	Title, Supplementary title: Monitoring & Comm System – Master (schematic)	Rev.: A; Date of issue: 04/07/2019 10:09; Sheet: 1/1

References

[1] Halvard Buhaug and Henrik Urdal. An urbanization bomb? population growth and social disorder in cities. *Global Environmental Change*, 23(1):1 – 10, 2013. URL: http://www.sciencedirect.com/science/article/pii/S095937801200129X, doi:https://doi.org/10.1016/j.gloenvcha.2012.10.016.

[2] UN. The world's cities in 2016. united nations - department of economic and social affairs,. 2016. URL: http://www.un.org/en/development/desa/population/publications/pdf/urbanization/the_worlds_cities_in_2016_data_booklet.pdf.

[3] WHO. Global report on urban health: equitable, healthier cities for sustainable development. world health organization, geneva, switzerland. 2016. URL: http://www.who.int/kobe_centre/publications/urban-global-report/en/.

[4] NASA. Technology Readiness Level Definitions. *Nasa*, page 1, 2012. URL: https://www.nasa.gov/directorates/heo/scan/engineering/technology/txt_accordion1.html.

[5] O. C. Khaligh, A. and Onar. *Energy harvesting: solar, wind, and ocean energy conversion systems.* CRC Press Inc., Boca Raton, FL, USA., 2010.

[6] FJA Duarte. Pavement Energy Harvesting System to Convert Vehicles Kinetic Energy into Electricity. 2018. URL: https://estudogeral.sib.uc.pt/handle/10316/79526.

[7] *ESP32 Series Datasheet*, 2019. URL: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf.

[8] Paulo Veríssimo. Ordering and timeliness requirements of dependable real-time programs. *Real-Time Systems*, 7(2):105–128, Sep 1994. URL: https://doi.org/10.1007/BF01088801, doi:10.1007/BF01088801.

[9] Philips Semiconductors. UM10204 I2C - bus specification and user manual Rev. 6-4 April 2014 User manual Document information Info Content. (April), 2014. URL: https://www.nxp.com/docs/en/user-guide/UM10204.pdf.

[10] NXP. PCA9554B; PCA9554C Low-voltage 8-bit I2C-bus and SMBus low power I/O port with interrupt, weak pull-up. (August):1–36, 2015. URL: https://www.nxp.com/docs/en/data-sheet/PCA9554B_PCA9554C.pdf.

[11] STMicroelectronics. STMF1 HAL Library UM1850 User manual. (April), 2017. URL: https://www.st.com/resource/en/user_manual/dm00154093.pdf.

[12] STMicroelectronics. AN2824 Application note STM32F10xxx I^2C optimized examples. (June):1–16, 2010. URL: https://www.st.com/resource/en/application_note/cd00209826.pdf.

[13] STMicroelectronics. STM32F103x8 STM32F103xB Usermanual. 2015. URL: https://www.st.com/resource/en/datasheet/stm32f103c8.pdf.

[14] STMicroelectronics. Application note STM32 GPIO configuration for hardware settings and low-power consumption. (September):1–31, 2017. URL: https://www.st.com/resource/en/application_note/dm00315319.pdf.

[15] STMicroelectronics. STM32^{TM}'s ADC modes and their applications. (March):1–18, 2010. URL: https://www.st.com/resource/en/application_note/cd00258017.pdf.