António João Marques de Andrade Pereira

Licenciado em Ciências de Engenharia Electrotécnica e de Computadores

Smart Sensor Data Acquisition in trains

Dissertação para obtenção do Grau de Mestre em Engenharia Electrotécnica e de Computadores

Orientador: João Paulo Pimentão, Professor Auxiliar, FCT-UNL

Co-orientadores: Pedro Alexandre Sousa, Professor Auxiliar, FCT-UNL

Júri:

Presidente: Professor Doutor Rodolfo Alexandre Duarte Oliveira, FCT/UNL

Arguentes: Professor Doutor Ricardo Luís Rosa Jardim Gonçalves, FCT/UNL

Vogais: Professor Doutor João Paulo Branquinho Pimentão, FCT/UNL

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2017

Smart Sensor Data Acquisition in trains

«Perseverança não é uma corrida longa, são muitas corridas curtas, uma após a outra.» - Walter Elliot

Acknowledgements

I would like to start by thanking professors João Paulo Pimentão and Pedro Sousa for having accepted to be my supervisors and for giving me the opportunity to work on an interesting and ambitious project. Without their know-how, insight and patience this dissertation would not be possible.

A word of appreciation for the staff of Holos for all the help they have given me in the development of this dissertation and in solving some problems in its implementation. In particular, a special thanks to Sérgio Onofre for his precious help and enormous patience.

To all my college mates who have contributed in some way to my academic journey, I owe a special thanks, since without them none of this had been possible. I would like to highlight two colleagues who have followed me in this course more closely, André Cardoso and Filipe Perestrelo, whom I thank for the friendship and companionship in these five years of course.

A big thank you to my mother, my siblings, and my grandmother for all the support they gave me in the good and bad times of my academic journey. A word still to Eng. Rosário Orvalho, who encouraged me to follow this course and to whom I owe a very special thanks.

Abstract

Whether for work or leisure, we see a large number of people traveling by train every day. In order to ensure the comfort and safety of passengers, it must be checked whether the composition is working normally. For this purpose, a constant monitoring of a train must be done, followed by a diagnosis of the composition, prediction of failures and production of alarms in the event of any anomaly.

To perform monitoring on a train, it is necessary to collect data from sensors distributed along its carriages and send them to a software system that performs the diagnosis of the composition in a fast and efficient way.

The description of the activities necessary for monitoring of a train immediately refers to topics such as distributed systems, since the intended system will have to integrate several sensors distributed along the train, or Smart Systems, since each sensor must have the capacity to not only acquire data, but also transmit it, preferably, wirelessly.

However, there are some obstacles to the implementation of such a system. Firstly, the existence of sources of distortions and noise in the medium interferes both in the acquisition and transmission of data and secondly the fact that the sensors distributed along the train are not prepared to be connected directly to a software system.

This dissertation seeks to find a solution for the problems described by implementing a data acquisition system that is distributed and takes advantage of

the current technologies of low-cost sensor nodes as well as web technologies for sensor networks.

Keywords: distortion, noise, distributed systems, Smart Systems, Smart Sensors, data acquisition, digital processing

---

Resumo

Quer seja por trabalho ou por lazer, vemos todos os dias um grande número de pessoas a viajarem de comboio. De modo a garantir o conforto e segurança dos passageiros, deve de se verificar se a composição está a funcionar normalmente e se não existe o risco de suceder algum acidente.

Para isso, deve de se fazer uma monitoração constante do comboio seguida de um diagnóstico da composição, predição de avarias e produção de alarmes no caso de se verificar alguma anomalia.

Para efectuar monitoração num comboio, é necessário recolher dados de sensores distribuídos pelo comboio e enviá-los para um sistema de *software* que realize o diagnóstico da composição de uma forma célere e eficiente.

A descrição das actividades necessárias para a monitoração de um comboio remete imediatamente para temas como sistemas distribuídos, uma vez que o sistema a implementar terá de integrar vários sensores, ou Smart Systems, dado que cada sensor deve ter a capacidade de não só adquirir dados como, de preferência, os transmitir sem fios.

Contudo, existem alguns obstáculos à implementação de um sistema desta natureza. Em primeiro lugar, a existência de fontes de ruído e distorções no meio que interferem quer na aquisição quer na transmissão dos dados e em segundo o facto dos sensores analógicos distribuídos ao longo do comboio não poderem ser integrados directamente num sistema de *software*.

Esta dissertação propõe uma solução para os problemas descritos através da implementação de um sistema de aquisição de dados distribuído e que tira

partido das tecnologias actuais de nós sensores de baixo custo e de tecnologias web para redes de sensores sem fios.

Palavras-chave: distorção, ruído, sistemas distribuídos, Smart Systems, aquisição de dados, processamento digital

---

Contents

List of figures

List of Tables

Glossary

µTESLA – micro-version of Timed-Efficient Stream Loss-tolerant Authentication

ADMR – Adaptive Demand-Drive Multicast Routing

ASIC - Application-Specific Integrated Circuit

CBQ – Code Blue Query

CMOS - Complementary Metal–Oxide–Semiconductor

DSP - Digital Signal Processor

EKG – Electrocardiograph

EMG – Electromyograph

GND - Ground

GPIO - General Purpose Input Output

GPRS - General Packet Radio Service

GPS - Global Positioning System

GSM - Global System for Mobile communications

HTTP - Hypertext Transfer Protocol

IEEE - Institute of Electrical and Electronics Engineers

MAC – Message Authentication Code

MCU - Microcontroller Unit

MD5 - Message-Digest algorithm 5

MEMS - Micro-Electro-Mechanical Systems

NAWMS – Nonintrusive Autonomous Water Monitoring System

P2P – Peer-to-Peer

PDA - Personal Digital Assistant

PHP - Hypertext Preprocessor

RC5 - Rivest Cipher 5

REST - Representational State Transfer

RFID - Radio-Frequency Identification

SHM – Structural Health Monitoring

SNEP - Sensor Network Encryption Protocol

SOAP - Simple Object Access Protocol

SPIN - Security Protocols for Sensor Networks

TESLA - Timed-Efficient Stream Loss-tolerant Authentication

WSDL – Web Service Description Language

WSN - Wireless Sensor Network

WWW - World Wide Web

XML - eXtensible Markup Language

1

Introduction

1.1 Motivation

Collecting information about faults, state of surrounding environment and equipment is crucial in systems that could jeopardize human lives, such as trains. Normally, data on trains is obtained through sensors that are not ready to be integrated into a software systems, making it difficult to integrate existing command and control systems with systems to monitor train's data.

The fact that human lives are involved requires near-real-time monitoring[1] of the data of various devices that make up a train, so that, in the event of an anomaly, the problem is identified quickly and the best strategy for solving that problem is adopted.

The evolution in the miniaturization of electronics allowed the integration of a large number of sensors in a small physical area, something that provides a superior monitoring capacity of an environment, introduction of redundancy, which allows a better response to failures, and the development of sensors with tasks that go beyond simple sensing such as processing or communication [1], [2].

[1]Although concepts such as "real-time systems" or "real-time computing" exist, that is, concepts that use the term "real-time", the designation adopted in this thesis was "near-real-time" because it is considered a more rigorous and more realistic designation.

This miniaturization of sensors favored the emergence of other types of systems called S*mart Systems*. These systems support features such as sensing, actuation and control, through sophisticated components or subsystems such as digital signal processing devices, analog RF and wireless devices, sensors and actuators for specific application, power sources and energy storage devices [3].

A fairly recent paradigm that has applications in Smart Systems is the paradigm of service-oriented computing. This paradigm uses services as support for a fast and low-cost development of distributed applications. Service-oriented computing represents a fusion of various technologies including distributed systems, software engineering, information systems, computer languages, Web-based computing, and XML [4].

1.2 Problem Identification

As mentioned, the monitoring of a train in near-real-time allows, in case of an anomaly, a quick action on the composition and preserves human lives.

Most trains have sensors spread over their carriages. These sensors provide information about the operation of the devices and mechanisms that constitute the composition, however, they are not prepared to be integrated into a software system that allows remote and near-real-time monitoring of train's operation.

Therefore, in order to implement a system that promotes monitoring of this nature, it is necessary, first and foremost, to design a connection element between the sensors and the software used to analyze sensor data.

The ability to self-diagnose, self-heal, and self-organization of so-called *Smart Systems* makes them particularly interesting for contexts that require a rapid response to anomalies that may arise in an environment, since these systems are therefore less dependent on the human agent.

An example of Smart Systems are *wireless sensor networks* (WSNs) which are networks of small, lightweight and low-power devices that are normally deployed in large numbers and which may have several applications [1]. These devices that make up a WSN are called *Smart Sensors*, which are sensors that, be-

sides having the sensing provided by sensors, have processing and communication capabilities. The characteristics of these devices allows them to be networked with other transducers.

These sensor networks are applicable not only in industrial systems, where the need for these technologies is obvious, as well as in systems integrated into everyday life of the common citizen [5].

An issue that arises in a system with Smart Sensors is the mitigation of distortions and noise at the level of sensors. A train has several sources of noise and distortions that affect the data collection of sensors, so Smart Sensors must be designed with this factor in mind. For this purpose, Smart Sensors must include a circuit that performs noise and distortion attenuation. The noise and distortions from the medium can also affect the communications that are made by Smart Sensors. This makes it important to choose the communication protocol to use on these devices. The protocol to be used should be highly mobile, easy to install, non-invasive and reliable.

Another issue to consider in a wireless sensor network is scalability. A high density of sensors allows redundancy and fault tolerance, however, the network protocols developed for this type of network must be capable of handling a large number of sensor nodes [6].

Regarding the interoperability between sensor network and sensor web layers, is not currently possible to dynamically install sensors with minimal human configuration effort. The sensor web layer relies on World Wide Web (WWW) and related protocols, while the network sensor layer is based on low-level communication protocols such as ZigBee [7], Bluetooth [8] or the IEEE 1451 [9] family of standards.

Most standard communication protocols are not specifically designed for sensor applications. Each wireless sensor network specifically defines its data formats. The sensor data cannot be exchanged or shared between the different wireless sensor networks [10].

Although the interoperability in sensor networks is currently a matter of discussion, the project developed in this dissertation comprises a sensor network

with sensors of the same type. This implies that there's no specific requirement for interoperability in the network, since there is a single platform for the sensors.

Considering what has been said before, the solution developed in this work seeks to be a fully distributed approach to data acquisition systems that allows a more robust and, at the same time, flexible data acquisition architecture for the context of a train. To this end, a scalable, fault tolerant, distortion and noise resistant sensor network must be implemented. For this purpose, the sensor nodes must be distortion and noise resistant in terms of data acquisition and data transmission. The sensor network data is gathered through Web Services.

1.3 Dissertation Structure

This section presents an introduction to the main theme of this dissertation. Its main concepts and related definitions are explained. Section 2 presents the state of the art on the subject, reviewing the technologies and the existing background work that is relevant to this dissertation. In section 3 an explanation of the architecture proposed to solve the presented problem is made. Section 4 presents the steps for the implementation of the proposed architecture. Section 5 presents and discusses the results obtained in the tests performed, taking into account the implementation done. Finally, section 6 presents a conclusion about the work and research developed, as well as a reflection on possible future projects.

2

State of the art

This chapter presents the state of the art in four areas that are involved in the theme of this dissertation: Smart Systems, Distributed Systems, Wireless Sensor Networks and Web services. In Section 2.1, an introduction to Smart Systems is made. Section 2.2 gives an overview of the concept of Distributed Systems. In Section 2.3 a definition of Wireless Sensor Networks is presented, taking into account the concept of Smart Sensor and the modules that constitute this type of sensors and some security issues. In Section 2.4 some applications of Wireless Sensor Networks are presented in several areas. Finally, Section 2.5 introduces the concept of Web service.

2.1 Smart Systems

Smart Systems are systems with the ability to sense, diagnose, describe, qualify and manage a situation and its operation improved through the interaction established with other systems. In order to support those functions, these systems must have components such as digital signal processors (DSPs), analog devices for wireless communication, sensors and actuators [11].

A Smart System should be able to collect usage and performance data to improve its operation, to collect data and process it to help the human agent make decisions and use the collected data to make decisions autonomously.

Traditionally, Smart Systems are typically classified into three generations, according to their autonomy, in terms of human agent supervision and energy self-sufficiency[11].

The first generation of Smart Systems integrates sensing and/or acting with signal processing for various types of actions. This type of Smart Systems is already well implemented and disseminated in several areas. An example of such systems are systems capable of monitoring people's health status and initiating necessary actions or safety systems in automotive applications [11].

The second generation of Smart Systems are predictive and adaptive systems with the ability to perform a self-test and thus better respond in critical environments. In addition, they are equipped with network facilities and have the ability to manage and harvest energy. Systems of this nature are more responsive to higher uncertainty given that they have the ability to learn and adapt, change environmental conditions and respond appropriately. An example of such systems are intelligent RFID labels with the ability to measure various parameters such as temperature, acceleration or slope for real-time transport monitoring [11].

The third generation of Smart Systems are systems that seek to replicate human behavior and actions and generate power. Systems of this generation act autonomously without human control or decision. They also have the ability to perform a self-test and thus perform self-calibration and self-healing. An example of such systems are highly or fully automated cars that have the ability to autonomously set direction, acceleration, de-equalization by monitoring the environment by itself requiring the human agent only as a backup for a potential emergency situation [11].

It is important to understand that the various generations of Smart Systems have different development rhythms. While the first generation systems still require deep research and development efforts, the second and the third generation systems still have scientific, material and manufacturing challenges that have to be assessed.

Current market trends show that Smart Systems are used in an ever wider range of contexts and environments, from everyday tasks to critical and demanding missions [12]. The architecture of each Smart System should be designed to

meet the constraints and requirements of the application in terms of autonomy, performance and safety while keeping the design and manufacturing costs as low as possible.

2.2 Distributed Systems

Distributed Systems can be defined as systems where hardware or software components located on networked computers communicate and coordinate their actions just by passing messages [13]. Networked computers can be spatially separated by any distance. The capabilities of a distributed system are defined from the following criteria [13], [14]:

- *Concurrency*. This kind of systems must consist of more than one sequential process, which may be this system or user processes, but each process must have its own thread of control; implicitly or explicitly. The ability of the system to manage shared resources can be increased by introducing more resources on the network;
- *Interprocess communication*. When programs need to cooperate with each other, they coordinate their actions by sending messages to each other. Messages need a finite time to pass from one process to another, however, there are limits to the accuracy with which computers synchronize their clocks. This notion implies the absence of a global clock signal, i.e., there is no global notion of the right time. This is a direct consequence of the fact that the only communication between processes is based on the sending of messages over the network;
- *Fault Response*. All computer systems can fail and therefore the consequences of each failure in the system must be planned. Failures on a network may involve isolation of computers, but this does not imply that they cease working. Programs on computers alone are not able to detect if the network has failed or if it has become abnormally slow, just as when abrupt termination of a program on a computer (a *crash*) occurs the other computers connected to it do not immediately have the notion of this failure. Each component of the system can fail independently, leaving the rest still running.

There are several examples of distributed systems that are used in everyday life in a variety of applications. Most systems are structured in a client-server logic where the server is the device that owns data or resources and provides services to several geographically distributed clients. There are applications that do not depend on a central server (peer-to-peer systems) and there are several database-oriented applications. In addition to sensor networks, there are some applications of distributed systems [13]:

- *Peer-to-peer (P2P) networking* is a paradigm where a set of Internet users machines exchange resources with other computers without requiring a central authority. With this paradigm, geographical boundaries became irrelevant in the absence of central authorities. Peers include collaborators and competitors and the orderly sharing of resources must be done from decentralized protocols. Scalability is an important feature of this concept;
- *Process control systems* apply to a large section of industry and influence all factors behind the manufacturing of a product. This involves manipulating various processes and variables in order to achieve automatic control. The number of variables depends on the product being designed. Typically, more than one variable may change during the production process. This type of system has the function of manipulating all these variables during the production process, each having an influence on the quality of the final product;
- *Grid Computing* is a form of distributed computing that supports parallel programming on a network with a variable number of computers. At the low level, a computational grid can use a fraction of the computational resources of one or two organizations, while at the high level it can combine millions of computers worldwide to work on large computational projects. The aim is to solve computational problems more quickly and in a less expensive way than in conventional methods.

2.3 Wireless Sensor Networks

Wireless Sensor Networks (WSN), also known as Wireless Sensor and Actuator Networks, are networks of objects capable of simple sensing, actuation, communication and computation. The performance of this kind of networks is maximized through the cooperation between all the objects. These networks, in turn, are able to communicate with other intelligent objects, other networks, other controllers and even with other users through interfaces [1].

For reasons related to the management of energy resources, it was necessary to change the communication strategies in sensor networks. In many cases, the desired optimization objectives for the sensor networks are difficult to achieve because of the restricted interactions between the various layers of the communication protocols. [1]

In order to overcome these limitations, solutions involving layer crossing were presented so that the information circulated through several layers and, in this way, allowed a joint optimization of the problems. However, solutions of this nature would lead to a reduction of modularity and thus an increase in the complexity of systems [1].

Due to the progression of artificial intelligence, (particularly of computation involving fuzzy logic) sensor networks have become more proactive rather than reactive in order to learn from environmental changes to make the most appropriate decisions and predict future behavior based in the application needs.

Objects that make up a WSN are denoted as *nodes* and the data they acquire is forwarded to a *sink* (also known as *controller* or *monitor*) that can use it locally or can be connected to other networks through a gateway[13],[15],[16].

The WSNs can be classified by the number of sinks and the number of hops located between the various nodes and sinks.

A single sink and single hop WSN allows direct transmission of data from nodes to sinks, which allows a reduced latency in data transmission, however, this type of network is not scalable, that is, with the increase of the number of nodes, the amount of data received by the sink increases. In case the sink capacity is reached, the network size cannot be increased [17].

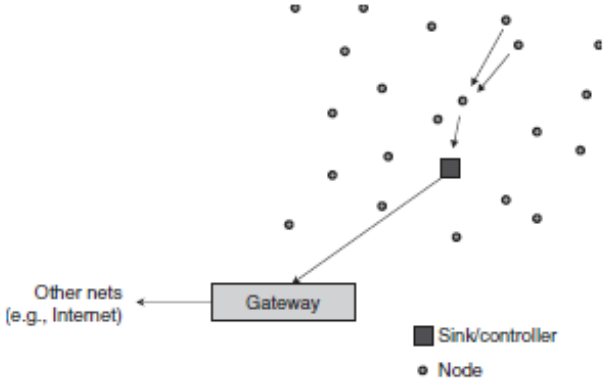

Figure 1 - Single-sink WSN [17]

In the case of the single sink and multi-hop network, data goes through several hops, which means that the number of transmissions required will be equal to the number of hops in the network. This causes the network to have less node capacity than in the case of the single sink and single hop WSN [17].

In the multi-sink and multi-hop scenario, the network can be scalable and reduces the possibility that groups of nodes cannot transmit data due to poor signal propagation conditions. In this type of WSN, the sinks can be connected by a separate network (wired or wireless) or disconnected [17].

In the first case, the nodes have to transmit data to an element of a set of sinks. The choice of the sink that receives data will be made according to a suitable criterion (e.g. minimum number of hops, minor delay, etc...). In this situation, a higher performance of the network will be guaranteed, however, there will be a greater complexity in the communication protocols. In the case where the sinks are disconnected, the presence of several sinks causes the monitored area to be fractioned into smaller areas. The WSN model with connected sinks is the most common and most interesting solution because it guarantees better performance.

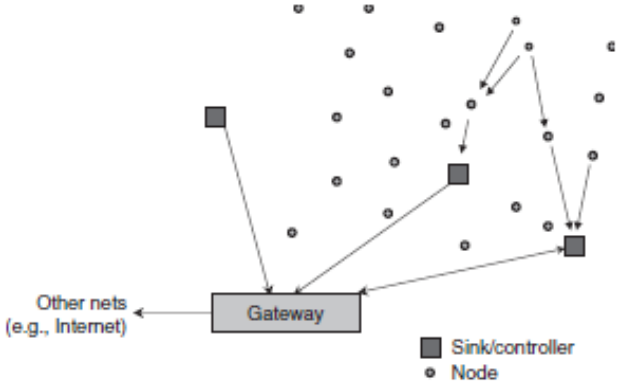

Figure 2 - Multi-sink WSN [17]

2.3.1 Smart Sensors

According to the IEEE 1451.2 specification, a Smart Sensor is defined as "A sensor version of a smart transducer" [18].

A transducer can be defined as a device that converts one form of energy into another, and can be also a sensor or an actuator. A smart transducer corresponds to a transducer that performs tasks that go beyond the correct representation of sensed or controlled quantities, and can be integrated in a network with other transducers [19].

The principle behind a Smart Sensor is that the sensor complexities must be concealed internally and must be transparent to the host system. Smart Sensors are designed to present a simple face to the host structure via a digital interface, so that the complexity is supported by the sensor and not by the central signal processing system [20].

The basic requirement for a Smart Sensor is that it be a system with dedicated "on-chip" signal processing. This means that the processing hardware of an electronic signal is dedicated to each sensor and miniaturized to the point where it becomes part of a sensor package [20].

Taking into account the concept of Smart Sensor [20],[18] it is possible to identify some of its main modules:

- Sensor, which will communicate with the surrounding environment;
- Interface circuit, which will perform operations such as noise attenuation;

- Circuit supply;
- Signal processing through a digital signal processor (DSP) with an associated Microcontroller Unit (MCU).

In general, a Smart Sensor will have the previously presented modules. The power will depend on the platforms used for DSP and interface circuit.

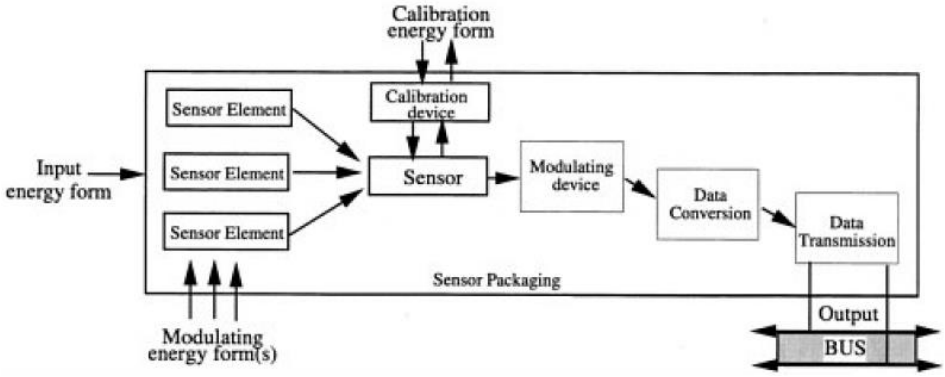

Figure 3 - Schematic of a Smart Sensor [20]

Signal processing module and interface circuit will be the modules of Smart Sensor that require a more in-depth study in its design. Therefore, following sections are intended to specify each of these modules.

2.3.1.1 Signal processing

A programmable DSP is a microcomputer designed for extensive arithmetic computation and digital signal processing functions through downloadable or resident software / firmware [21].

A standalone DSP, that is, without an embedded MCU, is not best suited for use on a sensor interface for four reasons [22]:

- DSPs do not have flexible interrupt structures;
- DSPs are not efficient at bit manipulation, e.g., individual I / O pins;
- DSPs rely heavily on off-chip memories and peripherals;
- DSPs are rarely available in very low pin counts, which is an important requirement for sensor processing since these applications tend to be space-constrained and cost-sensitive.

The DSP, in the context of a sensor node, will have two main functions: digital information processing and digital communication processing [20]. A memory is also included to store temporary data or during processing [17].

In the context of this dissertation, there are several solutions for the intended platform, Table 1 compares 3 solutions and their characteristics:

Table 1 - Comparison of various platforms[23]

Name	Analog Inputs	Digital I/O Pins	USB ports	Supply Voltage	RAM	Programming languages
Arduino	6	14	1	7-12 V/ USB	16-32 Kb	Arduino
Raspberry PI	0	14	1-2	5 V/ USB	256-512 Mb	C, C++, Python, Java
UDOO (Quad)	14	62+14	5	6-15 V	1 Gb	Arduino, C, C++, Java

An analysis of this table and the specifications provided allows to draw some conclusions about the various platforms. The Raspberry Pi has a smaller supply voltage, however, it does not have any analog pin, only General Purpose Input Output (GPIO) pins that provide a current of 3 mA, unlike the Arduino that, besides having analog pins, its digital I / O pins provide a current of 40 mA [24].

The fact that the pins provide more current makes it possible to feed more circuits without having to resort to additional circuitry, hence the importance of this parameter.

On the other hand, sometimes it is necessary to use many analog signal inputs, a need that neither Arduino nor Raspberry Pi can satisfy completely, only the UDOO board satisfies. The UDOO board has the advantage of supporting Arduino programming language, which makes it possible to implement Arduino projects on this board.

2.3.1.2 Interface circuit

The interface circuit will be responsible for signal conditioning through the following actions: attenuation of noise and distortions coming from sensors and

reduction input signal voltage to the operating voltage of digital signal processors.

For this purposes, a possible solution will be the Schmitt-Trigger circuit model. Considering the Schmitt Trigger inversion model with op-amps, this circuit functions as a comparator that switches its output negative when the input signal exceeds a given positive reference voltage and changes its output positive when the input signal is below a certain negative reference voltage [25].

This circuit utilizes a positive feedback of a negative signal to keep the output connected to the negative supply voltage until the input signal is below the lower voltage threshold, further allowing the output to stabilize and withstand any impulsive variations that may arise [25].

In addition to the model with op-amps, there is another variant of the Schmitt Trigger circuit with bipolar junction transistors.

Although the op-amp model can occupy a larger area and consume more power, it has a higher slew rate and, due to the high voltage gain of the op-amp, the values of the threshold levels are predominantly dependent on the feedback resistor values [26].

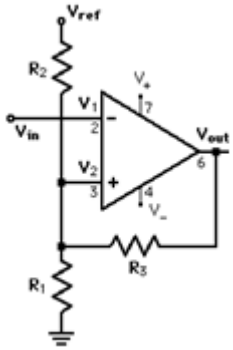

Figure 4 - Schmitt Trigger inversion model[25]

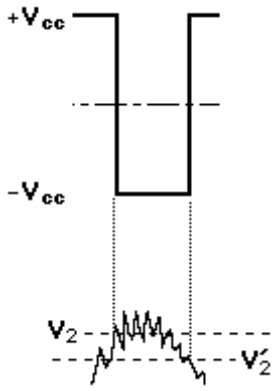

Figure 5 - Threshold voltages for Schmitt Trigger[25]

The threshold voltages have the following expressions [25]:

$$\begin{cases} v'_2 = \frac{R_{123}}{R_2} \cdot V_{ref} - \frac{R_{123}}{R_3} \cdot V_{cc} \\ v_2 = \frac{R_{123}}{R_2} \cdot V_{ref} + \frac{R_{123}}{R_3} \cdot V_{cc} \end{cases} \quad (1)$$

The resistor R_{123} represents the parallel of resistors R_1, R_2 and R_3.

2.3.1.3 Communication protocols

Once the structure of Smart Sensors is defined, it is also necessary to define the protocol to be used for sensor data transmission.

For this work, five protocols were analyzed: GSM, Wi-Fi, 3G, 4G and Ethernet. The following table presents some characteristics of various protocols:

Table 2 - Communication protocols[27],[28]

Name	Frequency Band	Signal Rate	Nominal Range	Nominal TX power
Ethernet	-	100 Mbps	100 m	-3/-9 dBm
GSM (GPRS)	850/900; 1800/1900 Mhz	168 Kbps	2-35 Km	0-39 dBm
Wifi	2.4; 5 Ghz	54 Mbps	10-100 m	15-20 dBm
3G	1400-1800 MHz	384 Kbps to 2 Mbps	2-35 Km	21-33 dBm
4G	2-8 GHz	20-100 Mbps	2-35 Km	43-48 dBm

Table 3 - Normalized Energy Consumptions [27]

Name	RX	TX
GSM(GPRS)	4207.3 mJ/Mb	6402.4 mJ/Mb
Wi-Fi	13.1 mJ/Mb	13.3 mJ/Mb

An analysis of the previous tables allows to draw some conclusions about the various protocols. Ethernet has the best signal transmission rate and between the five protocols would be the best in terms of data reliability, however, it has low mobility, which makes its assembly less flexible. Of the others, the GSM has a lower transmission rate but has a higher signal strength, which makes the detection of the sent signals easier, and a higher range.

As can be seen in Tables 2 and 3, GSM consumes more power due to its high range, which makes it energetically more costly. 3G and 4G have a range similar to that of GSM with the difference that they have higher transmission rates. 4G is energetically more expensive than GSM and 3G.

Although the Wi-Fi protocol is more susceptible to noise and has a low range, it has the best compromise between cost, mobility and efficiency [27].

2.3.2 Security

In this section some issues are discussed about safety in sensor networks, from the identification of adversaries to some security measures.

A large part of applications involving sensor networks are quite vulnerable to attacks since they are not designed with security as a top priority. Considering the typical set-up of a sensor network, where a reliable base station receives data from multiple sensor nodes, adversaries can be classed into three different classes: *passive*, *active* and *malicious* [13].

A *passive* adversary quietly robs unprotected data, or tampers data in transit or launches a replay attack. *Active* adversaries do a lot more damage since they can physically capture a node, extract the codes and keys, steal protected information using a stolen key or launch an attack by spying malicious code on the captured nodes. PC-class adversaries can affect multiple nodes remotely and launch *sinkhole* or *wormhole* attacks by forging route information [13].

Malicious adversaries attempt to damage the network by draining power from one or more nodes causing partitions in the network, tampering with data that could potentially harm the intended application or blocking communications in critical applications such as anti-theft control or military surveillance in order to facilitate a possible future enemy attacks [13].

2.3.2.1 SPIN

In order to protect the sensor networks against passive adversaries, a cryptographic protocol called SPIN [29] was created.

The implementation of a typical cryptographic protocol would be difficult to apply in a sensor network as it requires long public keys, high computational capabilities to verify these keys and high overhead of communication, something that sensor networks can not satisfy due to their limited resources.

Only fast secret-key cryptography can be used in this context, which is where SPIN fits.

This protocol follows the traditional configuration where a base station communicates with multiple sensor nodes from a source routing, but some nodes may not be reliable. Messages may be corrupted, however, all messages will be delivered to a sensor node eventually. SPIN consists of two main components: a Sensor Network Encryption Protocol (SNEP) and a μTESLA (micro-version of Timed Efficient Stream Loss-tolerant Authentication) [29].

SNEP allows *confidentiality*, *two-party data authentication*, *integrity* and *freshness*. Each node shares a unique master key with the base station being this key used to derive all other keys as the data encryption key, the message authentication code (MAC) key and the key for generation of random numbers.

The SNEP derives a unique encryption key through a message counter and the master key. This key is XOR-ed with the message bits and sent. The message counter is not transmitted directly, but communication processes try to track it and the eavesdropper is not aware of it. The receiver generates an identical key and XOR's it with the cipher text to receive clear text, an action that allows to maintain the confidentiality.

SNEP offers the following properties [29]:

- *Semantic security*, since the value of the counter is incremented after each message causing each message to be encrypted differently. The counter will be long enough to avoid repetition over the lifetime of a node;
- *Data authentication*. If the MAC verifies correctly, the receiver knows that the message originated from the intended sender;
- *Replay protection*. The counter value on the MAC prevents the repetition of old messages. If the counter was not present on the MAC, an adversary could easily repeat messages;
- *Weak freshness*. If the message checks correctly, a recipient knows that the message was sent after the previous message was received correctly. This forces that there is an order among the messages, which yields weak freshness;
- *Low communication overhead*. The counter state is maintained at each end point and does not need to be sent in each message.

RC5 [30] was chosen as the block cipher given the reduced size of its code and high efficiency. SNEP has a low communication overhead (8 bits per message).

µTESLA is a lightweight version of the TESLA protocol [31] for authenticated transmission that was designed for more heavy-duty platforms. Traditional authentication typically uses asymmetric key cryptography that is not easy to apply in sensor networks given the limitation of resources on the sensor nodes. µTESLA uses symmetric keys and authenticates messages by introducing asymmetry through a method involving delayed disclosure of the symmetric keys.

Each MAC key is an element of a key chain that is generated through a public one-way function F (e.g. a cryptographic hash function such as MD5 [32]). Keys are generated at regular time intervals and the sender associates each key of the key chain with a time interval. µTESLA generates a MAC key K_i for interval i ($i > 0$) using the formula $K_i = F(K_{i-1})$. Since F is a one-way function, anyone can calculate K_{i-1} through K_i, but only the base station can derive K_i through K_{i-1}.

When the base station sends a packet in the interval 0, with MAC key K_0, the receiver has no way to authenticate it since it does not have the verification key, however, no one else knows this key so it is impossible for anyone to have generated data. The receiving node is assured by the MAC that the verification key is only known by the base station, thus, the receiving node is assured that the packet it receives has not been affected by an adversary. Each verification key is revealed at the end of a set of time slots.

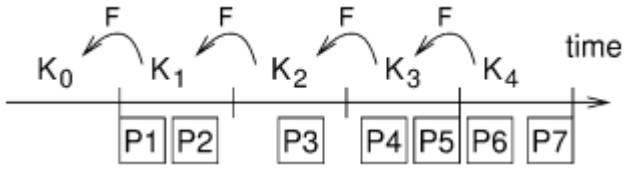

Figure 6 - µTESLA one-way key chain [29]

2.3.2.2 Attacks on routing

An active adversary can alter or falsify routing information by creating routing loops, divert traffic through one or more target nodes to drain their energy and partition the network or attract network traffic to affected nodes.

There are several attacks that can happen [13]:

- *Selective forwarding attack*, where an affected node releases important packets in order to cause damage to an application;
- *Sinkhole attacks* that attract traffic to affected nodes so they can manipulate data as they wish. A sinkhole attack is thrown when an affected node falsely advertises a high-quality path to the base station;
- *Wormhole attack* creates the illusion of a high-quality path by grouping data from one part of the network into another remote part of the network through a low latency path. This connection uses an out-of-bounds channel that is only accessible to the attacker. The low latency path attracts traffic and creates a sinkhole. Wormhole attacks can also be combined with eavesdropping and selective forwarding attacks.

Some applications require nodes to periodically send heartbeat messages. A PC-class adversary conveying a message of this nature to several nodes can

convince them that he is their neighbor. When an adversary advertises a high-quality path to a base station, other nodes adopt this path and send their data to an opponent, thus creating a sinkhole (or black hole) using a slightly different method.

The described attacks happen in the network layer, however, other attacks may happen in other layers, namely, in the physical layer.

2.3.3 WSN applications

The WSN paradigm has gained importance in recent years, which has led to extensive research on various aspects of it. In addition to this research, the applications of sensor networks with special focus on applications for WSNs have also been discussed.

Initially, the WSN were used for military purposes, since the sensor nodes were a way of detecting enemy troops in a reliable and discreet way [33].

2.3.3.1 Smart Dust

Smart Dust [34] is an example of the early WSN applications in this area. WSNs get the information they need to operate in critical situations allowing the establishment of a self-configured and self-organized WSN for a war scenario. In military applications, the information collection is made in order to detect movements of the enemy, presence of harmful chemical products and to evaluate the stability of infrastructures.

The main objective of the Smart Dust project was to develop small sensors (cubic millimeter packages) and focus on the development of transceivers and microcontroller units (MCUs), instead of focusing on just one sensor. The sensor node obtained was a prototype of 100 cubic millimeters with two chips: the MEMS corner cube optical transmitter array and the CMOS ASIC with an optical receiver, charge pump and simple digital controller.

Smart Dust nodes are also used to monitor the quality of food products. In this case, the temperature or humidity of meat or dairy products is monitored.

2.3.3.2 ZebraNet

The autonomous capacity of WSNs also allowed the development of several environmental applications. Applications for tracking the movement of animals, monitoring of environmental conditions affecting biotic communities, Earth monitoring and planetary exploration, meteorological and geophysical research and studies about pollution[35]–[39].

An example of such applications is the ZebraNet[35], [36] system which corresponds to a system for detecting long-term patterns in the movement of zebra species, their interactions with individuals of the same or another species, as well as estimation of the impacts of anthropic action. This system was implemented in Kenya to track two species of zebras.

ZebraNet consists of a collar that is attached to the neck of the zebras to collect location data that contains a GPS unit, an associated microcontroller, two types of radios (short and long range), lithium-ion polymer high-energy-density battery and a solar panel to recharge the battery.

The collar records information about the location through the GPS unit every 3 minutes, this information being collected through a base station and used intermittently by the investigators during their trips to the field. From the description of the system it is possible to conclude that, in fact, it is a highly mobile sensor network without a stationary sink.

Figure 7 - ZebraNet sensor collar [34]

The location information is transmitted in a delay-tolerant way, that is, the information will only be transmitted when there is an opportunity to communicate with the base station.

Since some sensor nodes may not be accessible to mobile sinks, to circumvent this limitation an information sharing strategy, between the various sensor nodes is adopted, wherein each sensor node shares the information collected with its neighbors, thus allowing, when a sensor node communicates with the sink, to transmit its information and that of its neighbors.

Two delay-tolerant routing protocols are used for data sharing. The first is a flooding protocol in which a sensor node sends its data to a neighbor whenever a new neighbor is detected. This protocol has the advantage of increasing the possibilities of data transmission to the sink, however, it overloads the network with data and limits the storage space in each node.

The second protocol relies on the successful communication history between the sink and the nodes assigning a hierarchical level [36] to each node. When a node is within the base station range, its hierarchical level increases. On the other hand, when a node is out of reach of the base station, its hierarchical level decreases over time, to the rhythm of one level for each consecutive D scans, that is, if D is 5, a decrease occurs of 1 hierarchical level for every 5 consecutive scans in which a node is not detected. Thus, each node shares its location with the neighbor with a higher hierarchical level with higher probability.

2.3.3.3 Code Blue

Another area where there are various applications for WSN is health. Due to the development of implanted biomedical devices and integrated Smart Sensors it was possible to create sensor networks for biomedical applications. Some health applications are, for example, support interfaces for people with disabilities, integrated patient monitoring, hospital drug administration, telemonitoring of human physiological data, tracking and monitoring of patients and doctors inside a hospital[40]–[43].

The Code Blue project at the Harvard University [41] is an example of a patient monitoring system that monitors the vital signs of a patient daily through wearable sensors. Sensor plates containing an electrocardiograph (EKG), a pulse oximeter and an electromyograph (EMG) circuit were designed for MicaZ and Telos "motes" [44], [45]. Parameters such as oxygen saturation in the blood,

heartbeat, electrical activities of the heart, muscle activity and patient movements can be thus monitored continuously.

The Code Blue software platform allows the various nodes to operate in network setting, allowing medical personnel to monitor patients through a PDA. Several sensor motes are attached to the patients for monitoring. Medical personnel can access the motes through a PDA or through a computer. The network is based on a publish/subscribe mechanism, being the sensors motes the ones who publish the information they obtain from patients and the medical staff to subscribe to this data under certain conditions.

The publish/subscribe engine is based on the Adaptive Demand-Drive Multicast Routing (ADMR) protocol, which maintains routing information based on published information. When a mote publishes data, each node in the network is informed about the best path to that node, so that each node periodically has information about the position of other nodes and thus withstand changes in the wireless channel. When a sensor connects to the network, its characteristics are communicated to the network through a discovery protocol.

The subscription of the published data is done through the Code Blue query (CBQ) layer, which specifies the query through a tuple (S, τ, chan, ρ, C, p), where S is the set of id's of subscribed nodes, τ is the type of subscribed sensors, chan is the channel type used for routing, ρ is the sampling rate and C is the maximum optional number of samples to be reported. There is also a filter predicate, p, which is used in the query to filter the sensor information according to user preferences.

In addition to the publish/subscribe service, Code Blue also provides a location service through a set of nodes that periodically transmit beacon messages while allowing a mobile node to estimate its location through the RF signal signatures received from the beacon nodes.

2.3.3.4 Water Monitoring System

Technologies such as Smart Sensor nodes and actuators have also led to the development of home devices with the ability to communicate with each other and with networks outside the home via the Internet or satellite, allowing control of the home devices either locally or remotely.

An example of such applications is the Nonintrusive Autonomous Water Monitoring System (NAWMS) [46], whose main objective is to locate wastes in the use of water and inform tenants about more efficient ways of using water resources.

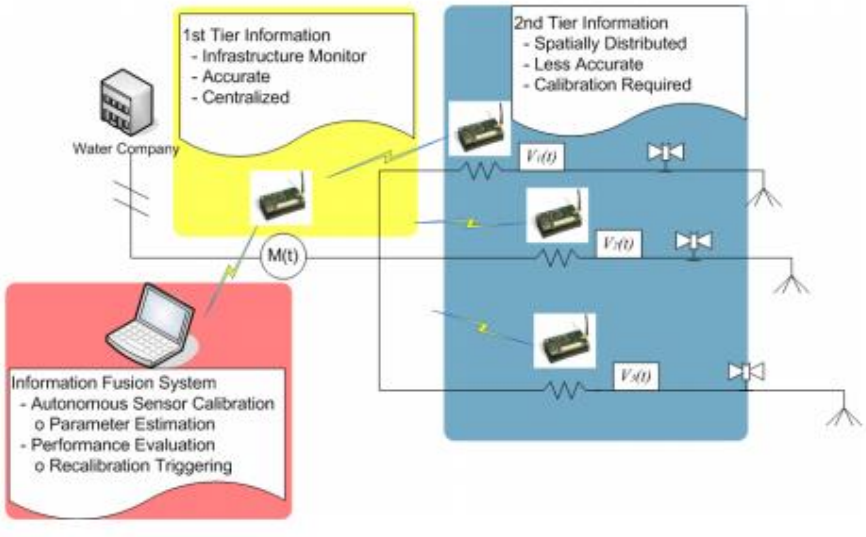

Figure 8 - NAWMS system [46]

Water companies only give a sense of the total expenditure of water in a house, making it difficult to identify specifically the points where water wastes the most. With a WSN it becomes possible to monitor all water sources in a home and determine the points where more water is wasted.

The principle behind the NAWMS is based on the fact that the flow of water in a pipe can be determined by the vibrations that occur because there is a proportionality relationship between the vibrations and the water flow. For this purpose, wireless sensor nodes are attached to the water pipes to measure vibrations from accelerometers.

Because there is a nonlinear relationship between vibrations and water flow, each sensor node needs to be calibrated to determine the optimal setting of parameters relating the acceleration to water flow. This calibration is performed automatically by the system from the main water meter.

The NAWMS architecture consists of three component types:

- A wireless sensor node attached to the main water meter, used to collect information about the water flow and transmit that information to the rest of the network;
- A vibration sensor installed in each water conduit to monitor the vibrations being generated. Information is transmitted to a central computing node that will automatically calibrate each sensor and determine the water usage in each pipe.
- Calibration is performed on the computation node by an online optimization algorithm, which is based on the fact that the sum of the water flows in each uncalibrated sensor must equal the flow in the water meter, which is known to be accurate.

2.3.3.5 Structural Health Monitoring

Wired sensor networks have long been used in the industrial field as industrial sensing and control applications, building automation and access control. However, the cost of implementing wired sensors limits the applications of such systems and an upgrade of this kind of systems may cost as much as a new system

For all of the reasons described above, WSNs are an interesting alternative because of its easy implementation, high granularity and high accuracy provided through battery-powered wireless communication units.

Some applications serve to monitor material degradation in infrastructure, monitor product quality, command and control robots in automated manufacturing environments, interactive museums, interactive toys, vehicle detection and tracing, or detection and monitoring of car thefts[47]–[53].

Structural Health Monitoring (SHM) is one of the possible applications of WSNs in industrial environments [47].

In these systems, the WSNs are used to detect spatio-temporal patterns of induced vibrations in structures since the potential degradation of an infrastructure can be detected and determined its extent almost in real time.

The actuators, which apply forced excitations in the structures, are connected in a network with the sensors, which have the function of detecting the

effects of the excitation of actuators. The challenges posed by data loss intolerance, stringent synchronization requirements and large volumes of raw data are resolved through a distributed WSN. The SHM was later extended to netSHM [48] which provides a two-level architecture in which the details of the WSN operation are hidden from the application.

2.3.3.6 Summary of applications

The previous examples show that wireless sensor networks have applications in various areas. From military and industrial applications to environmental applications, these networks enable the monitoring of an environment quickly, efficiently, and cost-effectively.

The SHM example shows that in wireless sensor networks, actuators can also be present and allow actions in addition to the sensing that the sensor nodes provide. In this case, data collection by the sensors depends on an action of the actuator, i.e. the actuator complements the action of the sensor.

In the examples of CodeBlue and ZebraNet the notion of mobile sensor nodes is present. This notion requires a greater interaction between the sensor nodes in order to know the location of the various nodes in the network.

CodeBlue combines a publish/subscribe (ADMR) mechanism, that allows sensors to publish data and medical personnel have access to that data, with a sensor node location service.

As for ZebraNet, sensor nodes are present in zebras that travel long distances during their migrations. This implies a great mobility of the network, requiring communication protocols that allow communication over long distances, a great interaction of the sensor nodes in order to have a sense of their location and also implies the existence of mobile sinks to make them more accessible to the sensor nodes.

The NAWMS has the peculiarity of automatically calibrating, through a computing node, the accelerometers attached to the sensor nodes, something that preserves the integrity of the network.

2.4 Web services

A Web service is a platform-independent, loosely coupled, self-contained, and programmable Web-enabled application service that can be described, published, discovered, coordinated, and configured using XML (eXtensible Markup Language) artifacts to develop distributed interoperable applications. Web services have the ability to incorporate other services into a common computing platform in order to: complete a specific task, conduct a business transaction or solve a complex problem [4].

In addition to this, Web services reveal their characteristics programmatically over the Internet using standard Internet languages (XML-based) and standard protocols and can be implemented through a self-descriptive interface based on open Internet standards.

A Web service is not the same as a Web page that provides access to an application over the Internet and across organizational boundaries. Web pages are targeted to human users while Web services are mainly designed to be accessed by automated applications.

Web services are different from previous distributed computing architectures because their protocols, interfaces, and service registry allow applications to work cooperatively with each other using the principle of loose coupling. To achieve this requirement, the service interface is defined in a neutral way that is independent of the underlying platform, operating system, and programming language in which the system is implemented. This feature allows services, built on a variety of such systems, to interact with each other in a uniform and universal way [4].

Web service is a self-contained software module that performs a single task. The module describes its own interface characteristics, i.e. the available operations, parameters, data entry and access protocol so that other software modules can determine what it does, how to invoke its features and what the expected outcome. In this regard, Web services are contracted software modules since they publicly provide available descriptions of the characteristics of the interface used to access the service. The client uses the description of the Web service interface to bind to a service provider and invoke its services [4].

Web services are described in terms of a standard description language. The Web Services Description Language (WSDL) [54] describes the characteristics of both functional and non-functional services. Functional features include operational characteristics that define the overall service behavior while non-functional characteristics mostly describe features of the hosting environment.

There are several differences between Web services and Web-based applications, of which the following ones may be mentioned [55]:

- Web services act as resources for other applications that can request and initialize those Web services, with or without human intervention. This means that Web services can call other Web services to outsource parts of a complex transaction to these other Web services. This provides a high degree of flexibility and adaptability that are not present in today's Web-based applications;
- A Web service knows what functions it can perform and what inputs it needs to produce its outputs, being able to describe all this to potential users and other Web services. A Web service can describe its non-functional properties, that is, the cost of invoking a service, the geographic areas that the Web service covers, security measures involved in the use of the Web service, performance characteristics, contact information, among others. In sum, Web services are modular, self-aware and self-descriptive applications;
- Web services are more visible and manageable than Web-based applications since the status of a Web service can be monitored and managed at any time through the use of external control applications and workflow systems. Although Web services may not run on local systems or may be written in unfamiliar programming languages, they can be used in the same way by local applications that can detect their status and manage the state of their outcomes.

Web services can be traded or auctioned. If multiple Web services use the same task, then multiple applications may submit proposals for the opportunity to use the requested service. The negotiator can base its choice on attributes of competing Web services (cost, speed, or degree of security).

2.5 State of art summary

The development of Smart Systems in recent years reflects a need for an autonomization of these systems relative to the human agent. This implies not only an autonomy in its normal operation but also a capacity for self-diagnosis and self-healing that prolongs the period of operation of these systems.

In the particular case of WSNs, the so-called Smart Sensors were fundamental in their development, as they allowed the integration of a large number of low-cost and multifaceted sensors in small physical areas.

The characteristics of WSNs seem to fit in the particular context of a train as they are prepared to deal with problems such as signal noise and distortions, which may arise at the level of sensors in data collection or transmission, are fault tolerant due to the redundancy that a large number of sensors introduces into the network, are scalable, since these networks have networking protocols prepared to handle a large number of sensor nodes and have low installation costs.

The use of the service-oriented computing paradigm allows to use web services in order to develop low-cost distributed applications through a platform-independent service interface. This feature increases the modularity of the system as well as its scalability.

The middleware area for sensor networks has been explored in recent years with particular interest. So far, few researches have been done on a middleware that integrates sensor networks with sensor webs. Most solutions typically focus on low-level functionality such as economic message routing rather than investing in strategies to facilitate the integration of sensor networks and the sensor web [56].

3

Architecture

In this chapter, the system architecture developed within this dissertation is presented. As noted in the state of art chapter, there are three basic elements of a wireless sensor network: nodes, sinks and gateways. In the case of the system presented here, sensor nodes are located in a train's carriage and have to communicate with a sink located in the same carriage. Subsequently, the sink communicates with a gateway located at the locomotive of the train.

For the system developed in this dissertation, the type of WSN used is multi sink single hop since in each carriage, a sensor node communicates with a single sink located in the same carriage. The conditions of connection to the sink, for each sensor node, are similar, that is, there's not a significant difference in the binding conditions of each sensor node, so there is no possibility that sensor nodes do not send data due to poor connection conditions.

In this system there are two gateways; one slave and another master. This solution is intended to introduce redundancy to the system and allow more robust data transmission from the sinks to the gateways, since in case one gateway suffers a problem, the other remains in operation and the network is not compromised.

Distortions and noise are problems to be taken into account in data transmission, which requires a careful choice of communication protocols to use. The priority is to ensure acceptable rates of data transmission. In data transmission

between the gateway and the data warehouse where all train's data is collected and processed, the priority will clearly be the transmission range.

3.1 Sensor Node

The sensor node has the function of receiving signals from the train's sensors and transmitting them to a sink located in the same carriage as the node. These signals, before being transmitted to the sensor node, pass through an interface circuit that has the functions of reducing operating voltage, attenuate distortions and noise. This circuit must be designed in such a way that as to be inexpensive, since there will be an interface circuit for each sensor located in the carriage.

Energy efficiency issues, which arise in most WSNs, are not considered in the network developed in this dissertation since each sensor node has a static location that allows it to be powered, that is, it does not need its own energy source.

In a train there are already multiple digital sensors in their carriages, something that allows to introduce redundancy in the network of sensors. Redundancy is important to introduce fault tolerance in the operation of a wireless sensor. In each carriage there must be more than one sensor node because if there were only one sensor node per carriage, in the event of a failure in that node, data collection from that carriage would be compromised.

After processing the signals, data is sent to a sink. The communication method to be used must be chosen taking into account that a train has several sources of noise and distortions that can influence the transmission of data. Since in the system developed in this dissertation the sensor nodes are located near the sink, the range of the method to be used will not be a priority, so the priorities will be to make the method with low installation costs and good transmission reliability.

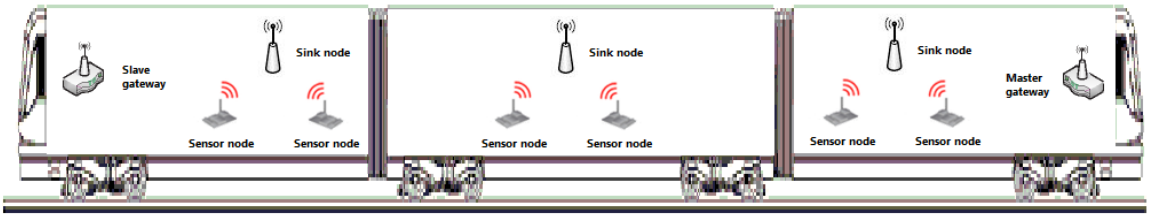

Figure 9 - Wireless Sensor Network in a train

3.2 Sink node

The sink node must receive data from sensor nodes that communicate with it and send it to a gateway. Considering the classifications of WSNs introduced in the chapter of the state of art, the WSN of this dissertation is of the type multi sink single hop with disconnected sinks, something that implies a good capacity of storage of the several sink nodes. This can lead to network scalability issues, a bit similar to what happens in networks of the type single sink single hop.

However, the fact that there is a sink for each carriage reduces the number of sensor nodes that communicate with the sink, favoring the scalability of the network. On the other hand, the fact that the network has disconnected sinks allows the use of less complex communication protocols and fraction of the monitored area in smaller areas (train's carriages).

The sink node's location on the network also has to be carefully thought to choose the best communication method to use. If it is further away from the sensor nodes, then a communication protocol with a higher signal strength is required, otherwise the signal power will not be a priority and protocols that require lower energy consumption can be chosen.

The communication protocol to be used between sink and gateway should be chosen taking into account the noise and distortions that may occur and the distance they are from the gateway since not all sink nodes are at the same distance from the gateway.

The specifications referred to in the last two paragraphs will be presented in the next chapter.

3.3 Web Service and Database

The communication between sink and gateway is made through a web service. The characteristics of web services contribute to the scalability of the network and the reduction of network costs, since web services can be implemented using reliable transport mechanisms.

In this dissertation the gateway, in addition to receiving data from the sinks and sending them out of the train, has a database where the data received from the various sink nodes are stored. The existence of this local database is a strategy to respond to situations where there are failures in communication outside of the train. Thus, if there is a communication problem, data is present in the database and ready to be accessed, or resent once the link is re-established.

The gateway, when receiving data from multiple sink nodes, has to deal with the competition problem since data has to be written in a critical resource: The database. In this way, semaphores are used to prevent race conditions.

4

Implementation

4.1 Interface Circuit

As already mentioned, the interface circuit basically has two functions: amplitude reduction and attenuation of noise and distortions of the input signal. The first objective can be reached through a voltage divider circuit. The voltage divider is a circuit that produces an output voltage corresponding to a given fraction of the input signal amplitude and is represented in the following figure:

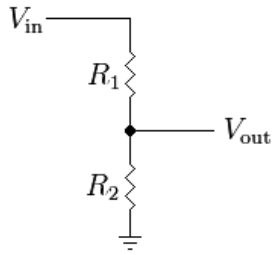

Figure 10 - Voltage Divider Circuit

From Figure 10, the relationship between the input and output signal is the following:

$$V_{out}=V_{in}\cdot\frac{R_2}{R_1+R_2} \qquad (2)$$

This circuit allows the reduction of the amplitude of the input signal, however, the noise and distortions present in the input signal have not yet been attenuated, something that forces to adopt other options.

The solution chosen was already discussed in the state of art chapter: the Schmitt Trigger circuit. This circuit allows the choice of two voltage levels for which the signal saturates only from the dimensioning of resistors and a suitable choice of the op-amp to use.

In this way, the noise and distortions present in the input signal are attenuated and the output signal is ready to be sent to the sensor node.

Through the models introduced previously, it is possible to define the interface circuit to be used. The following figure illustrates that circuit:

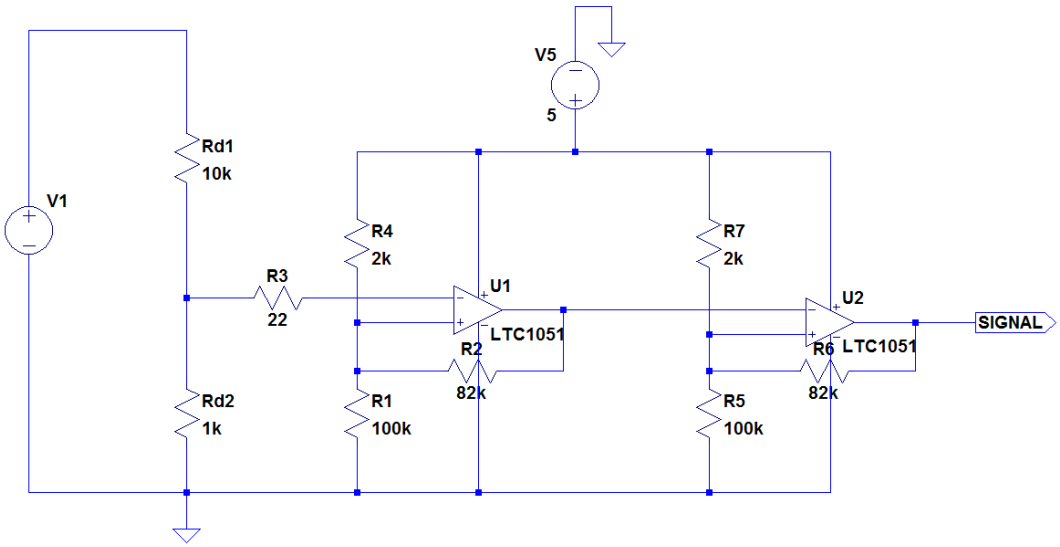

Figure 11 - Interface circuit

The op-amps used in the circuit of Figure 11 are of the LTC 1051CN8 [57] model of Linear Technology. This model was chosen, mainly, due to its low noise and consumption, two objectives defined at the outset for the system proposed in this dissertation. Resistor R_3 serves to control the current entering the circuit.

In order to verify the performance of the circuit in the presence of noise, a noise simulation was performed using the ".NOISE" command of the SPICE simulator of electronic circuits LTSpice. The following figure represents the output referred noise of the circuit:

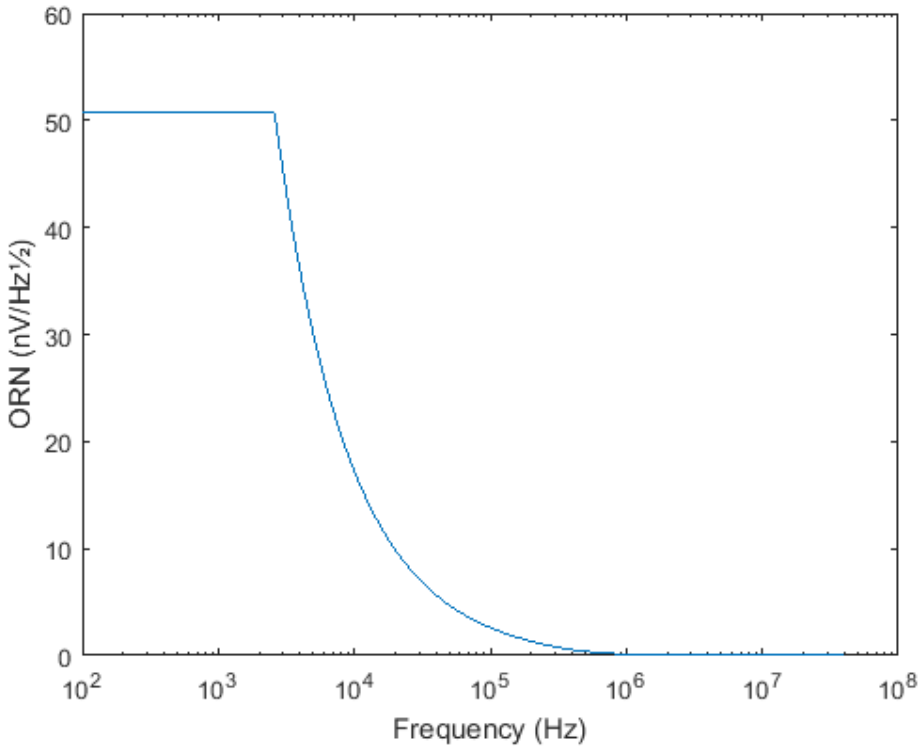

Figure 12 - Output referred noise of the circuit

As can be seen in Figure 12, the noise remains constant at low frequencies, with a power spectral density value around 50nV / Hz ½. For frequencies above about 3 KHz, noise decays.

4.1.1 Circuit dimensioning

Considering as an input signal a square wave with amplitude between 0 and 72 V, a period of 6 ms (166.67 Hz) and a duty cycle of around 33%. It is intended to reduce the amplitude from 0 to 72 V for values between 0 and 5 V.

In this way, the output signal of the voltage divider circuit must be 0 V when there is 0 V at the input and 5 V when there is 72 V at the input.

For an input signal at 0 V, by (2) with the resistors in Figure 11, it is possible to verify that V_{out} will be equal to 0 V independently of the values of resistors Rd_1 and Rd_2.

For when V_{in} equals 72 V, it will be necessary to dimension the resistors so that V_{out} is equal to 5 V. Thus, assigning the value of 1 KΩ to the resistor Rd_2 and using equation (2):

$$5=72\cdot\frac{1000}{Rd_1+1000} \Leftrightarrow Rd_1=\frac{72000-5000}{5} \Leftrightarrow Rd_1=13.4\ \text{K}\Omega \approx 13\ \text{K}\Omega \quad (3)$$

With the voltage divider circuit properly dimensioned, the Schmitt Trigger circuit remains to be dimensioned. Considering the circuit of Figure 4, the values considered for v_2 and v'_2 in this example were 4.9 and 4.6 V, respectively. Taking into account the equations in (1) with $V_{ref} = V_{cc} = 5$ V, $R_2 = 2$ KΩ, $R_3 = 82$ KΩ and $R_{123}=\frac{1952\cdot R_1}{1952+ R_1}$:

$$\begin{cases} v_2= 5\cdot\left(\frac{\frac{R_1\cdot R_{123}}{R_1+R_{123}}}{2000}+\frac{\frac{R_1\cdot R_{123}}{R_1+R_{123}}}{82000}\right) \\ v'_2= 5\cdot\left(\frac{\frac{R_1 R_{123}}{R_1+R_{123}}}{2000}-\frac{\frac{R_1\cdot R_{123}}{R_1+R_{123}}}{82000}\right) \end{cases} \Leftrightarrow R_1=100\ \text{K}\Omega \quad (4)$$

The dimensioning done is shown in Figure 11. Note that the Schmitt Trigger circuit at its output has the voltage levels inverted. To invert again, it is enough only to add another Schmitt Trigger comparator with the first Schmitt Trigger output signal to enter into its negative input, as shown in figure 11.

Theoretically, it would be sufficient to add only one comparator with the output of the first Schmitt Trigger connected to its negative input and the positive input connected to V_{ref}, however, this solution only results if the output of the first Schmitt Trigger gives values very close to 0 or 5 V. Since this in practice does not occur, a second Schmitt Trigger circuit was used.

As can be seen in the circuit of Figure 11, one of the resistors of the voltage divider is not in accordance with the theoretical design. Resistor Rd_1 is 10 KΩ instead of 13 KΩ.

For the circuit developed under this dissertation, it is necessary to give a greater margin for the voltages v_2 and v'_2 in order to embrace more values. To do this, two possible actions can be done: resizing of the Schmitt Trigger circuit or resizing of the voltage divider circuit.

The second action is clearly easier to implement than the first, since scaling the v_2 and v'_2 levels from the Schmitt Trigger circuit implies a change in the resistor set of this circuit which is a more complex task than just changing a resistor in the voltage divider circuit.

In order to test the circuit in terms of noise attenuation, a sinusoidal wave of 10 V amplitude and frequency of 1 KHz was added to the original signal.

The results obtained were as follows:

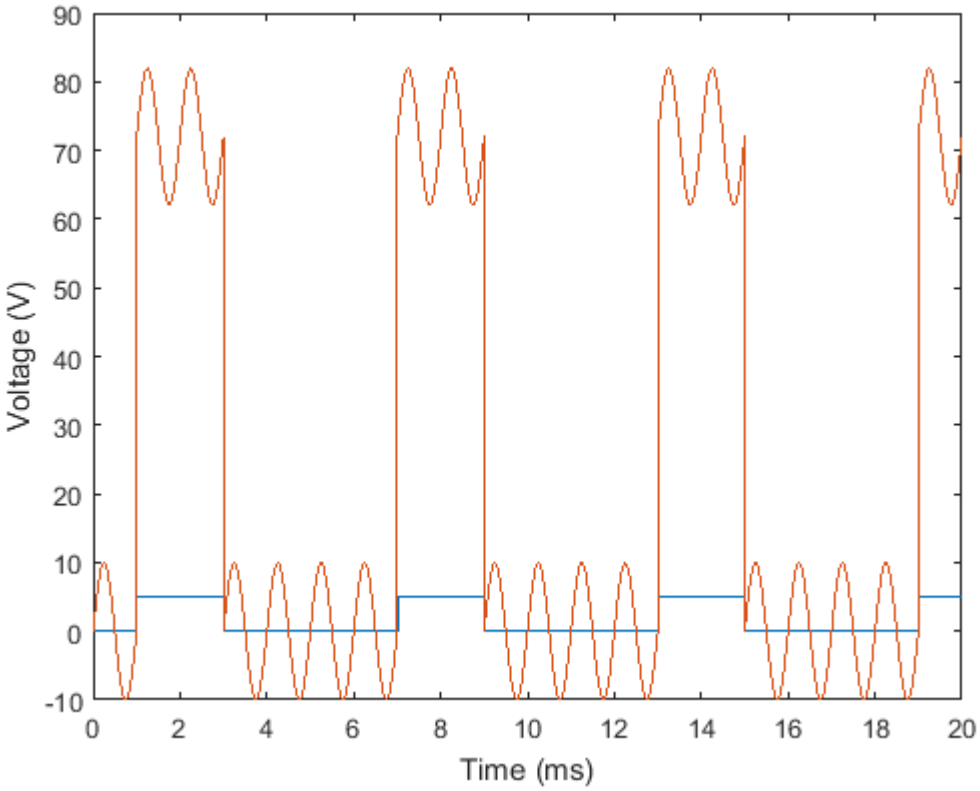

Figure 13 - Output signal (blue wave) and input signal (red wave)

4.2 Sensor and Sink nodes

The sensor and sink nodes are implemented in an Arduino UNO WiFi board. In chapter 3, it was mentioned that one of the tasks to be performed before implementing the network was to define the best communication method to use between the sink and sensor nodes. Any communication protocol that could be chosen would entail additional costs for the facilities and communications to be used, in addition to the previously mentioned problem of noise and distortions that greatly influence communication.

Another factor that was decisive in defining the dynamics between sink and sensor nodes was the position of the digital sensors in the train. The sensors are

located in static and known positions, something that allows a wired connection of sensors to the Arduino board.

According to the WSN definition and examples presented in chapter 2, sink and sensor nodes constitute separate entities or the sink simply may not exist. In the case of this sensor network, the Arduino board simultaneously represents a sink node and several sensor nodes since the data of each digital sensor is sent to the board and processed, as in the situation where we have several sensor nodes with its own power supply, storage and processing capabilities and the data of all sensors is available to be sent wirelessly to a gateway, as in a sink node. Therefore, we will have multiple sensor nodes "virtualized" by the Arduino board.

The board corresponds to an Arduino UNO with an integrated Wi-Fi module. This solution allows to take advantage of the documentation and projects already existing and tested in Arduino UNO with the advantage of not having to include a Wi-Fi shield, in this case.

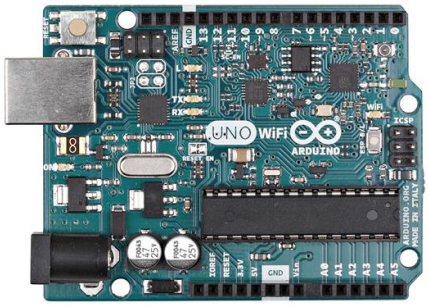

Figure 14 - Arduino UNO WiFi

There is a library called WiFi Link that allows to take advantage of the syntax and functions of another existing Arduino library called WiFi.h. The original WiFi.h library cannot be used directly with Arduino UNO WiFi board since the functions depend on the presence of a Wi-Fi shield.

For this library to work on the UNO WiFi board, it is necessary to install the WiFi Link firmware on this card. To upgrade the board's firmware, it's needed to upload the ESP recovery sketch, which is in the examples section of Arduino IDE in "WiFi Link library", and put the board in DFU mode. To put the board in DFU mode simply press the ESP B / L button whose location is shown in the following figure.

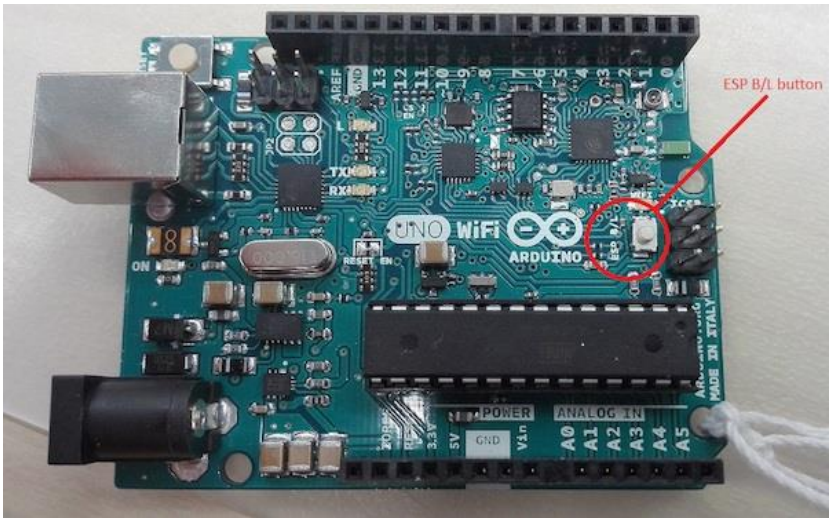

Figure 15 – ESP B/L button location

As in the existing Arduino WiFi.h library, it is needed to connect to a Wi-Fi network, so the SSID and password for that network must be provided. To send HTTP requests to a server, must be created an object of type "WiFiClient" and use the "print" or "println" functions.

Knowing how the data will be sent, it was needed to develop the code in the Arduino IDE. The following figures represent parts of the generated code:

```
//Check if communication with wifi module has been established
if (WiFi.status() == WL_NO_WIFI_MODULE_COMM) {
  Serial.println("Communication with WiFi module not established.");
  while (true); // don't continue:
}

// attempt to connect to Wifi network:
while (status != WL_CONNECTED) {
  Serial.print("Attempting to connect to SSID: ");
  Serial.println(ssid);
  // Connect to WPA/WPA2 network. Change this line if using open or WEP network:
  status = WiFi.begin(ssid, pass);

  // wait 10 seconds for connection:
  delay(10000);
}
Serial.println("Connected to wifi");
printWifiStatus();
```

Figure 16 - Code to connect to a Wi-Fi network

```
void senData(String data, String data2, String location){

  Serial.println("\nStarting connection to server...");
  // if you get a connection, report back via serial:
  if (client.connect(ip, 80)) {
    Serial.println("connected to server");
    // Make a HTTP request:
    client.println("GET /tese/serv.php/" + data + ";" + data2 + "/" + location + " HTTP/1.1");
    client.println("Host: " + SERVER_ADDR);
    client.println("Connection: close");
    client.println();
  }

  client.flush();
  client.stop();
}
```

Figure 17 – Function "senData"

```
String getLocation(){
  char c;
  String location = "";

  Serial.println("\nStarting connection to server...");
  // if you get a connection, report back via serial:
  if (client.connect(ip, 3333)) {
    Serial.println("connected to server");
    // Make a HTTP request:
    client.println("GET /LocationWS/rest/getLocation HTTP/1.1");
    client.println("Host: " + SERVER_ADDR);
    client.println("Connection: close");
    client.println();
  }

  delay(500);

  while (client.available() && status == WL_CONNECTED) {
    if(c == '\n'){
      location = "";
    }
    c = client.read();
    location = String(location + c);
  }

  client.flush();
  client.stop();

  return location;
}
```

Figure 18 - Function "getLocation"

Figures 16, 17 and 18 represent the parts that were considered most important in the code.

The input voltages of the Arduino's digital pins are transferred to the ATmega328P microcontroller which, depending on the voltage on a digital pin, indicates whether the pin is HIGH (1) or LOW (0).

Knowing that the input signal of each digital pin is a rectangular wave with peaks around 0 and around 5 V, it is assumed that when the microcontroller registers the HIGH value the pin has the value of voltage 5 V, whereas if it has the value LOW the pin has a value of 0 V. This processing is done at the level of the digital pins.

At the level of analog pins the approach is different. In these pins, the value obtained corresponds to an integer value between 0 and 1023 which corresponds to the output of an analog-to-digital converter with 10 bits of resolution.

By this value, it is possible to determine the voltage present on the analog pin. In order to carry out processing similar to the work done on the digital pins, it is necessary to know the voltage values for which the HIGH and LOW values are considered.

Using the datasheet of ATmega328P microcontroller [58], it is possible to verify that the HIGH value is considered for a voltage above 4.1 or 4.2 V and LOW for a value below 0.9 or 1.0 V.

For this dissertation, it is assumed that for voltages above 4.2 V has a value of 5 V and below 4.2 V has 0 V.

Since both server and adapter are on the same Wi-Fi network, this allows us to set a fixed IP for each device. The board address is configured by first connecting a computer to the Arduino-generated network with an SSID of type "Arduino-Uno-WiFi-xxxxxx" and secondly by connecting the computer, through a browser, to the link http://192.168.240.1/.

The IP and Wi-Fi network connection to which the board and the server are connected are already configured and it's possible to verify the board's address and network SSID.

The "senData" function, shown in Figure 17, serves to send sensor data and location of the board to the server. This function has as input parameters three Strings, two that have data sensor data ("data" and "data2") and one that has information about board's location ("location").

The "ip" variable, present in the "senData" function, corresponds to a String that has the IP of the server where the data will be sent and the variable "client"

corresponds to the object of type "WiFiClient" through which the HTTP request will be made to server.

The sensor data correspond to the measurement made by a sensor and the id of that same sensor. The location data of the board is latitude, longitude and a value that corresponds to the route of the train where the board is located called "mission".

For sending the measurement and location data in the "senData" function, all data is appended to the default command in the HTTP request, "/tese/serv.php". The measurement data is separated from the location data by a "/" character, as can be seen in Figure 17. The request is made from port 80 of the address defined in the variable "ip".

To receive the location data in the "getLocation" function, the default command in the HTTP request, "/ LocationWS / rest / getLocation", is sent to the server located at port 3333 of the address defined by the variable "ip", as can be seen in Figure 18.

4.3 Web service and Gateway

The Gateway on this network must have a REST server to receive data from the various Arduino UNO WiFi boards, store the received data in a SQL database and send that data off the train from a SOAP client.

The first task was to create the REST server. For this purpose, a PHP script called "serv.php", an Apache HTTP server and a MySQL database were created in the PHP XAMPP development environment. This option was taken due to the fact that XAMPP is a stack package of free and open source multiplatform web server solutions that is easy to use and install.

As mentioned earlier, Arduinos send the measurement of a sensor and its id so, after converting the data string into a string vector, what remains is to save each pair of values in the database.

Before continuing to analyze the PHP script, it is necessary to talk in detail about the database and how it was designed. For the network developed in this dissertation, five entities were created:

- Board, which represents the Arduino boards. It has an id of the board, which is an auto-incremental integer, the board number and the carriage identifier, both are of the type varchar(45);
- Type, which represents the type of sensor. It has a type id, which is an auto-incremental integer, and the type name that corresponds to a varchar(45) object;
- Unit, which represents the unit of measurement of each sensor. It has a unit id, which corresponds to an auto-incremental integer, and the unit name of the type varchar(45);
- Sensor, which corresponds to the sensors in the network. It has a sensor id, which corresponds to an auto-incremental integer, the id of the board to which the sensor is associated, the type id associated with the sensor and the id of the unit associated with the sensor. The last three referred variables correspond to foreign keys and are all integers;
- Measure, which corresponds to a measurement made by a sensor. It has a measurement id, which corresponds to an auto-incremental integer, the time stamp when the measurement was made, of the Datetime type, the registered value, of the float type, and the id of the sensor where the measurement was made. This last variable corresponds to a foreign key and is an integer.

In order to better illustrate the relationships between the various entities of the database, the following figure represents the relational model of this database.

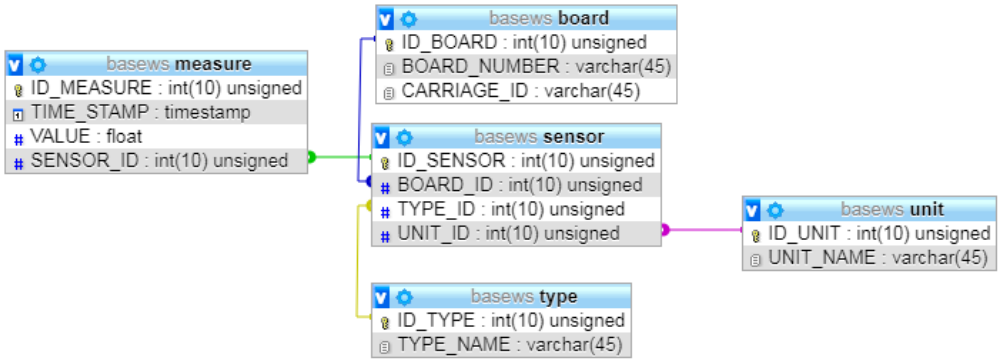

Figure 19 - Relational database model

Knowing the database, it is possible to understand the rest of the PHP script code. When the sensor ids and the measurement values are obtained from the data string, they are inserted in the database's measure table. As mentioned, the measure entity table has four rows, so when the sensor ids and measurement values are passed, the time stamp and the measurement id are automatically generated in the database.

Once the data is stored in the database, it will be sent off the train through a SOAP client.

One problem that can arise in this script are the race conditions, since the various boards when sending their data require simultaneous access to a critical resource: the database. To guard against this problem, semaphores were introduced in order to control access to the critical resource.

Semaphores in PHP are not available for the Windows OS, so a Semaphore class has been used in order to implement semaphores regardless of the platform where the PHP script runs. If the operating system is different from Windows, then it's used the PHP semaphores, if the operating system is Windows, then a file with a .lock extension is used. The PHP flock function executes the LOCK_EX operation under the .lock file when it's wanted to acquire the semaphore and LOCK_UN when it's wanted to release the semaphore.

5

Tests and results

After the system described in chapter 4 has been implemented this chapter presents, the practical tests, each one of the components that constitute this system. The tests of the system involves three main components: a circuit that represents the interface circuit between the sensors and the Arduino, the Arduino board itself and a personal computer that behaves, in this phase of tests, as a gateway.

The first tests were done on the interface circuit. To perform this tests, a dual trace oscilloscope KIOTTO G-5040 and a function generator Topward Electronics Instruments 8105 were used. To verify the operation of the circuit, a rectangular wave was generated in the function generator, which corresponds to the input signal of the circuit, varying the amplitude and frequency of this same signal and verified the behavior of the output of the circuit with the variation of these two parameters.

The second set of tests was done between the Arduino board and the PC that represents the gateway. The test laptop was an ASUS P50IJ with an Intel Core 2 Duo 2 GHz, 4GB RAM, and running Windows 7 Ultimate 64-bit operating system The Arduino IDE version 1.8.4 was used in the board. The values of the analog and digital inputs of the board were read and sent to the PC. From the side of the PC, the correct reception of the data was verified.

In the PC test three web services were published on an internal network on two different servers. The server running the PHP script that receives the Arduino data is an Apache 2.0 server launched through the XAMPP 3.2.2 platform. In this server a PHP script that simulates a location data delivery service is run.

The second server publishes a SOAP web service that simulates the web service that will receive the data of the train and is an Apache Tomcat 7.0.77. The PHP script receives Arduino data, sends information about the measurements to the SOAP web service. The data sent is the time stamp of the measurement, information about the location of the board (latitude, longitude and mission), sensor identifier and measurement value.

5.1 Interface circuit test

The interface circuit comprises two Schmitt Triggers and a voltage divider. For this test, only the two Schmitt Triggers were mounted with the input signal coming from the function generator because it was not possible in the laboratory to generate a wave of sufficiently high amplitude to apply the voltage divider circuit. The assembly is shown in the following image.

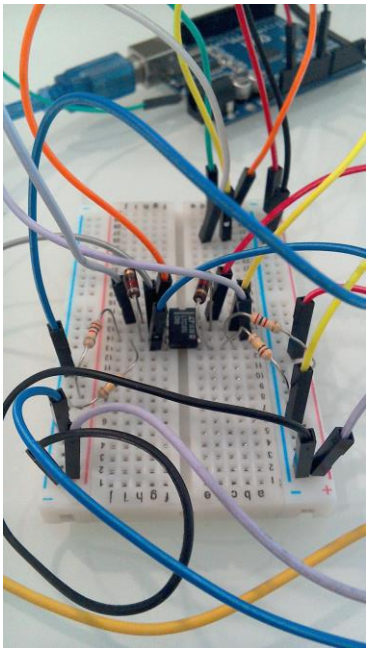

Figure 20 - Interface circuit without voltage divider

In the previous image it is possible to check in the middle of the breadboard only one element, which corresponds to the op-amps of the two Schmitt Triggers,

flanked by a set of resistors. This element has two LTC1051 [55] op-amps, the output of one being the input of the other. The power of the circuit in the test was produced by the power pins of the Arduino board using the 5V and GND pins.

Regarding the resistors used in this dissertation, all have a tolerance of 5% and the power reference of each one was chosen in order to support the power required for each one in the circuit. Taking into account the circuit of Figure 11, all resistors have a power of 0.25W except resistor Rd_1, which has a reference of 2W. This is because this resistor is located in the voltage divider circuit and is responsible for reducing the input voltage from 0/72V to 0/5V.

In order to verify the value of the power reference, the same simulation described in chapter 4.1 was performed and a graph of the power in resistor Rd_1 over time was obtained. The result is shown in the following figure:

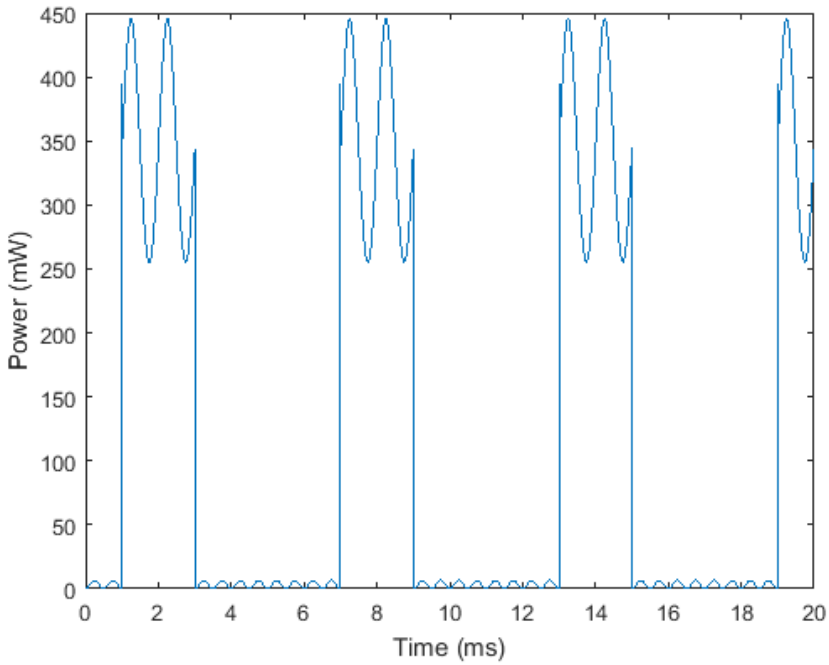

Figure 21 - Power in resistor Rd_1 over time

As can be seen, the power has a maximum value of about 440 mW, so a power reference of 0.25W would be insufficient. Although, for example, a 1W reference is sufficient, this value should be chosen with a higher margin in order to avoid any unforeseen events that may occur in the operation of the circuit, therefore, the value chosen for the power reference was 2W.

With all components of the circuit properly dimensioned, the preparation of the test follows. The input signal was a square wave to which the amplitude and frequency are varied. The output signal was checked on the oscilloscope.

For frequencies lower than 100 Hz, it was not possible to verify the behavior of the circuit since the oscilloscope did not present the output wave clearly. Therefore, the tests performed were done with frequencies above 100 Hz.

To perform the tests, two frequencies were chosen: 200 Hz and 1 KHz. These frequencies have been selected so that we have a low enough value (200 Hz) to check the performance of the circuit at low frequencies and, at the same time, to check the output signal on the oscilloscope, and a value high enough (1 KHz) to verify the behavior of the circuit to higher input frequencies.

Figure 22 - Function generator with 200 Hz square wave

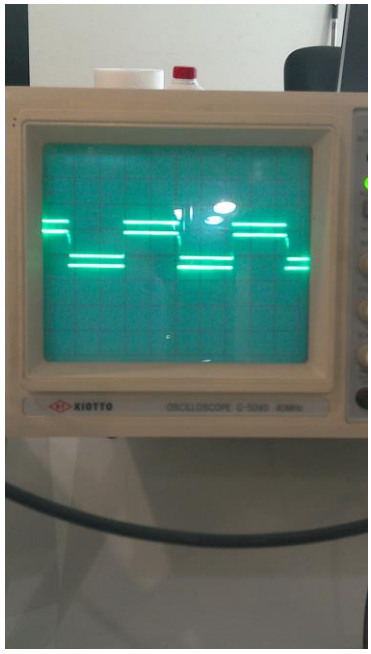

Figure 23 - Input and output signals on the oscilloscope, with an input wave of 200 Hz and the output wave saturated at 5 V.

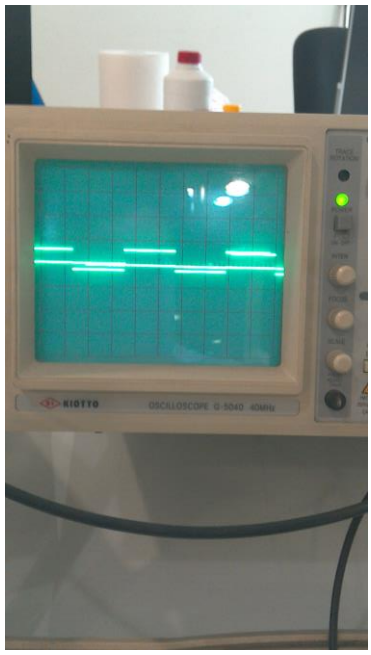

Figure 24 - Input and output signals on the oscilloscope, with an input wave of 200 Hz and the output wave saturated at 0 V.

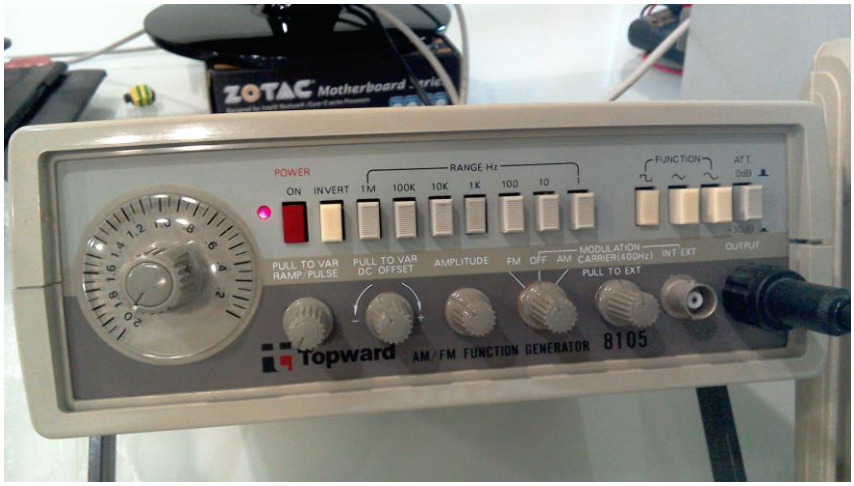

Figure 25 - Function generator with 1 KHz square wave

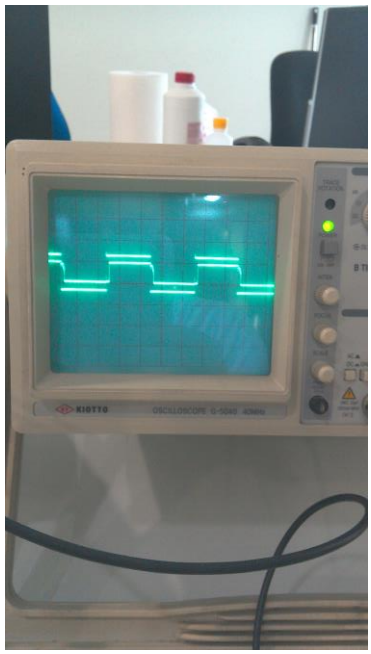

Figure 26 - Input and output signals on the oscilloscope, with an input wave of 1 KHz and the output wave saturated at 5 V

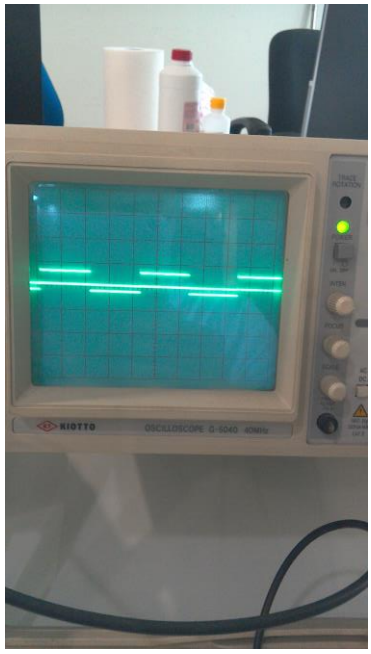

Figure 27 - Input and output signals on the oscilloscope, with an input wave of 1 KHz and the output wave saturated at 0 V

The tests showed that up to frequencies around 1 KHz the circuit could follow the input signal and make the amplitude correction. For frequencies above 1 KHz, the output signal has some delay with respect to the input signal, as can be seen in Figure 26.

The output signal saturates to 5V for any voltage above 4V, i.e. maintains the saturated value at 5V for any value above 4V. This means that if the input voltage were reduced from 72V to 5V, there would be a margin of 14.4V below 72V for which the output signal from the circuit would remain saturated at 5V. This margin, for this dissertation, is insufficient, hence the strategy of reducing the voltage division at the level of the voltage divider circuit.

Considering the circuit of Figure 11, with Rd_1 = 10 KΩ and Rd_2 = 1 KΩ the range of values for which the output signal saturates to 5V increases to about 28V.

5.2 Communication between Arduino and PC

With the interface circuit tested, it is possible to pass the connection test of the Arduino board with the PC that represents the gateway. Each output of an

interface circuit is connected to an Arduino analog or digital pin and these values are sent to the gateway.

The first action that is performed is obtaining the location through a web service REST. Once the location of the board is obtained, the location data is attached to the measurement data of the sensors. All data is then sent to a PHP script named “serv.php” that receives and stores it in a MySQL database.

The following figure shows the Arduino IDE serial monitor which registers the reception of the location data by the board and the sending of the location and measurement data.

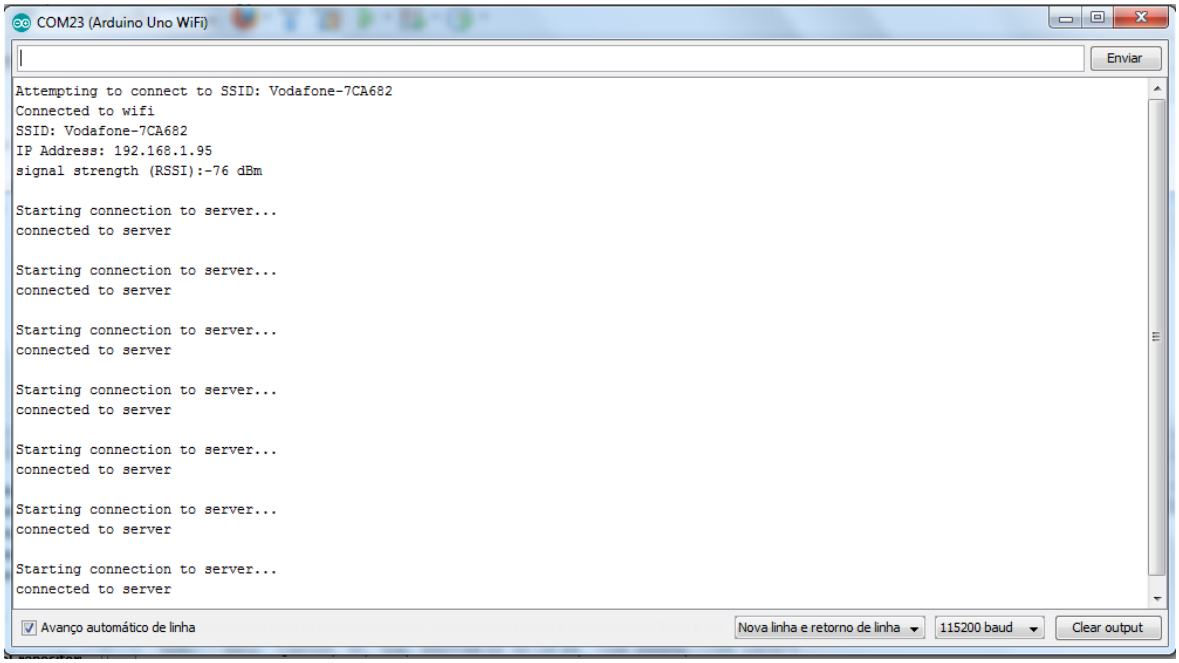

Figure 28 - Arduino IDE serial monitor

In Figure 28 it is possible to verify the connection of Arduino to the local network, moment registered in the first string written in the Arduino serial monitor, followed by several strings that indicate the connection of the Arduino to the gateway for sending data.

The data is sent to the PHP script "serv.php" which stores it in a database and sends it to a SOAP web service. To verify that the data is sent to a SOAP web service, the web service was created and the data that was being sent by the PHP script was printed. The web service was implemented in the NetBeans IDE development environment and published on a Glassfish 4.1.1 server. The web

method responsible for receiving the PHP script data is represented in the following figure.

```java
/**
 * Web service operation
 */
@WebMethod(operationName = "send")
public String send(@WebParam(name = "number") ArrayList<String> number) throws Exception {
    System.out.println("Data: " + number.toString());
    return "Hello Arduino! How are you?";
}
```

Figure 29 - SOAP web service method

The results obtained were as follows.

```
Java DB Database Process | GlassFish Server 4.1.1 | Apache Tomcat 7 | Apache Tomcat 7 Log
Info:   Data: [jst123, 01, 0cd, 2017-09-22 00:29:43, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 0cd, 2017-09-22 00:29:43, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 0cd, 2017-09-22 00:29:43, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 0cd, 2017-09-22 00:29:43, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 0cd, 2017-09-22 00:29:43, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 0cd, 2017-09-22 00:29:43, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 0cd, 2017-09-22 00:29:43, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 0cd, 2017-09-22 00:29:43, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 0cd, 2017-09-22 00:29:43, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 0cd, 2017-09-22 00:29:43, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 0cd, 2017-09-22 00:29:43, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 5g, 2017-09-22 00:29:46, -144.333304, -189.132027]
Info:   Data: [kjh123, 01, 5kg/(m^3), 2017-09-22 00:29:46, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 5cd, 2017-09-22 00:29:46, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 5cd, 2017-09-22 00:29:46, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 5cd, 2017-09-22 00:29:46, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 0cd, 2017-09-22 00:29:46, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 0cd, 2017-09-22 00:29:46, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 0cd, 2017-09-22 00:29:46, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 0cd, 2017-09-22 00:29:47, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 5cd, 2017-09-22 00:29:47, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 5cd, 2017-09-22 00:29:47, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 5cd, 2017-09-22 00:29:47, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 0cd, 2017-09-22 00:29:47, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 0cd, 2017-09-22 00:29:47, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 0cd, 2017-09-22 00:29:47, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 0cd, 2017-09-22 00:29:47, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 0cd, 2017-09-22 00:29:47, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 0cd, 2017-09-22 00:29:47, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 0cd, 2017-09-22 00:29:47, -144.333304, -189.132027]
Info:   Data: [jst123, 01, 0cd, 2017-09-22 00:29:47, -144.333304, -189.132027]
```

Figure 30 - Data received in SOAP web service

In the previous figure, the printout of the received data array of the gateway is verified. Data, shown in figure 30, are arranged as follows: carriage id followed by mission, measurement value (with the respective unit), timestamp, latitude and longitude. The values represented in the figure are not actual values, they are only test data sent to test the connection between the gateway and the SOAP web service.

The sampling time recorded between the various measurements was 2 s.

6

Conclusions and Future Work

This chapter presents a summary of conclusions drawn from the work performed of this dissertation. It also proposes future work that can be done to complement and improve the underlying project.

6.1 Conclusions

A train requires constant monitoring of its operation in order to ensure the safety of the human lives it transports. A monitoring effort of this nature requires a great autonomy of monitoring the system so that anomalies are rapidly detected and there is a swift intervention to solve them, or at least minimize their impact.

A total autonomy of the system with respect to the human agent is something difficult to perform since there are always actions in which a human has to intervene. However, there are actions that given their monotonous and standardized nature can be replicated by other elements (namely actuators controlled by a Smart System that, in this way, contribute to the reduction of costs and failures.

In the case of a wireless sensor network, each sensor node is capable of sensing the physical environment, collecting and processing sensed data, and communicating with each other or with a base station in order to accomplish certain common tasks. In addition to performing these tasks, these sensor nodes are de-

signed to have a low cost and low power consumption, making energy and installation costs low. The attenuation of noise and distortions in the measurement of the values of digital sensors also contributes to the decrease of the probability of measurement errors, making the sensor network more resistant to failures.

To this end, an electronic system, to deal with attenuation of the effects of electromagnetic noise within the train, while, at the same time allowing the collection of digital (0-5V) data from sensors that are based on the train's electrical infrastructure and therefore use 72V DC had to be developed, implemented, tested and analyzed.

In the wireless sensor network developed under the scope of this dissertation, the presence of web services is fundamental for increasing the mobility of the system, to promote flexibility, for reduction of cost and installation time. For each Arduino board to send its data to the gateway, it only has to connect to the same network where the gateway is present and make an HTTP request to the server where the web service is published. The same happens when the Arduino board obtains its location data, in which case it sends a HTTP request to the server where the location service is published and receives the data of its location.

The interoperability of web services was a very important feature for the development of this sensor network because it allowed to explore several platforms and choose the best combination.

As it was possible to verify in the chapter of tests and results, the period of sampling of the values of the sensors was very long, something that prevents the conception of a near-real-time monitoring system. This is due to the fact that the Arduino board used in this dissertation (Arduino UNO WiFi) connects ATmega328 microcontroller with ESP8266 Wi-Fi module using IO expander SC16IS750. The Wi-Fi module is connected to IO expander with Serial, but the ATmega328 is connected to IO expander using analog ports A4 and A5 with TWI protocol. The IO expander converts the protocol. This means that speed is limited to 19200 baud, which corresponds to a very low transmission rate.

In conclusion, although there is a limitation related to the data transmission rate of the Arduino board, the developed system allows the integration of sensor

nodes in a flexible, fast and low cost way allowing important tasks to be performed without the intervention of the human agent. Noise and distortion attenuation promotes greater resistance to system failures at the level of digital sensor data acquisition.

There is also a problem related to the communication protocol used to send the data out of the train in the system devised in this dissertation.

For advanced 3G / 4G networks with high mobility, the maximum speed that can be reached is 350 Km / h, something that makes this type of networks little suitable to apply in a train [59]. This forces to look for alternatives to apply in the developed solution.

6.2 Future Work

The Arduino board used in this dissertation corresponds to a developer edition that does not yet have its libraries and functionalities properly explored and developed, so other options may be explored in the future in order to correct this situation.

The test of the network in real train conditions, an environment that will may introduce noise and distortions components other than the ones that were done in laboratory, is important to verify the action of the interface circuit in a real context, i.e. to verify how data acquisition is being processed at the level of digital sensors and also serves to verify how is processing the communication between the Arduino board and the gateway, because in case of communication failures it is necessary to develop additional strategies to overcome this obstacle. Unfortunately, during the period of development of the system, we were not able to secure the conditions (authorisation, namely regarding safety clearance) to perform the tests in a real running train.

After the system is tested in real conditions, the autonomy of the network can be assessed. The cost of sensor nodes tends to reduce with the passage of time, as well as the physical space they occupy. These two factors can contribute, in the future to the integration of a greater number of sensors and, thus, to promote a monitoring capacity not only of the environment but also of the network itself, in order to promote activities such as self-healing and self-diagnosis.

In order to solve the problem of out-of-train communication, the proposed alternative to the 3G / 4G protocol is the standard for wireless telecommunications GSM-R (Global System for Mobile Communication) [60].

Approved and sponsored by the International Union of Railways, it is based on GSM and the EIRENE – MORANE [61] specifications that guarantee performances at speeds up to 500 km / h without loss of communication [60].

GSM-R corresponds to GSM but with specific features directed to trains. It operates in the 900 MHz band and provides a wide range of functions such as logistics, passenger information, on-board signaling, etc [61].

6.3 Scientific Contributions

As part of the development of this dissertation, the following scientific article was produced and accepted for publication in the proceedings of IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society (IES)(iecon2017.csp.escience.cn/):

Smart Sensor Data Acquisition in trains, Pereira, A., Pimentão, J., Sousa, P. Onofre, S.; *Published in the Proceedings of IECON*, 2017, ISBN: 978-1-5386-1126-5/17, Pages: 5598-5603, Beijing, November, 2017.

7

References

[1] Ibnkahla, Mohamed. *Wireless sensor networks: a cognitive perspective*. Crc Press, 2012.

[2] Chason, Marc, et al. "Micro-electro mechanical system." U.S. Patent No. 6,649,852. 18 Nov. 2003.

[3] Sassone, Alessandro, et al. "Smart Electronic Systems: An Overview." Smart Systems Integration and Simulation. Springer International Publishing, 2016. 5-21.

[4] Papazoglou, Michael. *Web services: principles and technology*. Pearson Education, 2008.

[5] Ghayvat, Hemant, et al. "WSN-and IOT-based smart homes and their extension to smart buildings." *Sensors* 15.5 (2015): 10350-10379..

[6] Akyildiz, Ian F., and Mehmet Can Vuran. *Wireless sensor networks*. Vol. 4. John Wiley & Sons, 2010.

[7] OECD (2009), Conference proceedings: ICTs, the environment and climate change, In *Proceedings of the High-Level OECD Conference Helsingør*, Denmark.

[8] OECD. Publishing. *OECD environmental outlook to 2030*. Organisation for Economic Co-operation and Development, 2008.

[9] Yick, Jennifer, Biswanath Mukherjee, and Dipak Ghosal. "Wireless sensor network survey." *Computer networks* 52.12 (2008): 2292-2330.

[10] Mukhopadhyay, Subhas Chandra. *Advances in wireless sensors and sensor networks*. Ed. Henry Leung. Berlin, Germany: Springer, 2010.

[11] Smart Systems in the Multi-Annual Strategic Research and Innovation

Agenda of the JTI ECSEL, Part D, EPoSS Industry Association, March 2014.

[12] Bombieri, Nicola, Massimo Poncino, and Graziano Pravadelli, eds. *Smart systems integration and simulation*. Springer, 2016.

[13] Ghosh, Sukumar. *Distributed systems: an algorithmic approach*. CRC press, 2014.

[14] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair.Distributed Systems: Concepts and Design. Addison-Wesley Publishing Company, USA, 5th edition, 2011.

[15] D. Culler , D. Estrin and M. Srivastava , "Overview of Sensor Networks," http://doi.ieeecomputersociety.org/10.1109/MC.2004.93, *Computer*, vol. 37, no. 8, 2004,pp. 41-49.

[16] C. Chong and S.P. Kumar, "Sensor Networks: Evolution, Opportunities, and Challenges," *Proc. IEEE*, vol. 91, no. 8, 2003, pp. 1247-1256.

[17] R. Verdone, D. Dardari, G. Mazzini, and A. Conti, *Wireless Sensor and Actuator Networks: Technologies, Analysis and Design*. Elsevier, 2008.

[18] IEEE 1451.2 Standard, "A Smart Transducer Interface for Sensors and Actuators - Transducer to Microprocessor Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats," Piscataway, NJ: IEEE Standards Department, 1998.

[19] Institute of Electrical and Electronics Engineers, IEEE Std 1451.5TM 2007, IEEE Standard for a Smart Transducer Interface for Sensors and Actuators Common Functions, Communication Protocols, and Transducer Electronic Data Sheet (TEDS) Formats, no. October. 2007.

[20] National Research Council, Expanding the Vision of Sensor Materials Washington D.C.: National Academy Press, 1995, pp. 15-17.

[21] Baier, Paul Walter, Peter Jung, and Anja Klein. "Taking the challenge of multiple access for third-generation cellular mobile radio systems-a European view." *IEEE Communications magazine* 34.2 (1996): 82-89.

[22] P. Sinha, "Smart Sensing Power on a Chip: Digital Signal Controllers," Sensors Magazine, September 2003.

[23] M. Maksimović, V. Vujović, N. Davidović, V. Milošević, B. Perišić, "Raspberry Pi as Internet of Things hardware: Performances and Constraints," 1st International Conference on Electrical, Electronic and Computing Engineering - IcETRAN 2014, Vrnjačka Banja, Serbia 2014.

[24] Monk, Simon. *Raspberry Pi cookbook: Software and hardware problems and solutions.* " O'Reilly Media, Inc.", 2016.

[25] Georgia State University. HyperPhysics. Retrieved from http:// hyperphysics.phy-astr.gsu.edu/.

[26] Kargarrazi, Saleh, Luigia Lanni, and Carl Mikael Zetterling. "Design and characterization of 500° C Schmitt trigger in 4H-SiC." *Materials Science Forum*. Vol. 821. Trans Tech Publications, 2015.

[27] Chakkor Saad et al, "Comparative Performance Analysis of Wireless Communication Protocols for Intelligent Sensors and Their Applications", International Journal of Advanced Computer Science and Applications, SAI Publisher, Volume 5 No 4, May 2014.

[28] Fagbohun, O. (2014). Comparative studies on 3G, 4G and 5G wireless technology. *IOSR Journal of Electronics and Communication Engineering*, 9(3), 88-94.

[29] Perrig, Adrian, et al. "SPINS: Security protocols for sensor networks." *Wireless networks* 8.5 (2002): 521-534.

[30] Rivest, Ronald L. "The RC5 encryption algorithm." *International Workshop on Fast Software Encryption*. Springer Berlin Heidelberg, 1994.

[31] Perrig, Adrian, et al. "Efficient authentication and signing of multicast streams over lossy channels." *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*. IEEE, 2000.

[32] R. Rivest, The MD5 message-digest algorithm. RFC 1321, Internet Engineering Task Force (1992).

[33] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, and V. Naik, "A line in the sand : a wireless sensor network for target detection , classification , and tracking q," vol. 46, pp. 605–634, 2004.

[34] J.M. Kahn, R.H. Katz, K.S.J. Pister, Next century challenges: mobile networking for smart dust, Proceedings of the ACM MobiCom'99, Washington, USA, 1999, pp. 271–278.

[35] P. Zhang, C. Sadler, S. Lyon, and M. Martonosi. Hardware design experiences in ZebraNet. In Proceedings of ACM SenSys'04, Baltimore, MD, USA, November 2004.

[36] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet. ACM SIGOPS Operating Systems Review, 36(5):96–107, 2002.

[37] ENSCO Inc. Retrieved from http://www.ensco.com.

[38] E. A. Basha, S. Ravela, and D. Rus. Model-based monitoring for early warning flood detection. In Proceedings of ACM SenSys'08, pp. 295–308, Raleigh, NC, USA, November 2008.

[39] G. Werner-Allen, K. Lorincz, M. Ruiz, O. Marcillo, J. Johnson, J. Lees, and M. Welsh. Deploying a wireless sensor network on an active volcano. IEEE Internet Computing, 10(2):18–25, March/April 2006.

[40] Artificial Retina project. Retrieved from http://artificialretina.energy.gov.

[41] D. Malan, T. Fulford-Jones, M. Welsh, and S. Moulton. CodeBlue: an Ad Hoc sensor network infrastructure for emergency medical care. In Proceedings of Workshop on Applications of Mobile Embedded Systems (WAMES 2004), Boston, MA, USA, June 2004.

[42] T. Gao, C. Pesto, L. Selavo, Y. Chen, J. G. Ko, J.H. Lim, A. Terzis, A. Watt, J. Jeng, B. r. Chen, K. Lorincz, and M. Welsh. Wireless medical sensor networks in emergency response: implementation and pilot results. In Proceedings of IEEE International Conference on Technologies for Homeland Security, pp. 187–192, Waltham, MA, USA, May 2008.

[43] A. Wood, G. Virone, T. Doan, Q. Cao, L. Selavo, Y. Wu, L. Fang, Z. He, S. Lin, and J. Stankovic. ALARMNET: wireless sensor networks for assisted-living and residential monitoring. Technical report CS-2006-11, Department of Computer Science, University of Virginia, 2006.

[44] Memsic. MicaZ – wireless measurment system. Retrieved from http://www.memsic.com/.

[45] Crossbow. TELOSB – TeloB mote platform. Retrieved from http://www.willow.co.uk/.

[46] Y. Kim, T. Schmid, Z. M. Charbiwala, J. Friedman, and M. B. Srivastava. NAWMS: Nonintrusive Autonomous Water Monitoring System. In Proceedings of ACM SenSys'08, pp. 309–322, Raleigh, NC, USA, November 2008.

[47] J. Paek, K. Chintalapudi, J. Cafferey, R. Govindan, and S. Masri. A wireless sensor network for structural health monitoring: performance and experience. In Proceedings of the 2nd IEEE Workshop on Embedded Networked Sensors (EmNetS-II), Sydney, Australia, May 2005.

[48] A. Chianese and F. Picciali, "Designing a smart museum: When cultural heritage joins IoT," in *Proceedings - 2014 8th International Conference on Next Generation Mobile Applications, Services and Technologies, NGMAST 2014*, 2014, pp. 300–306.

[49] K. Chintalapudi, J. Paek, O. Gnawali, T. Fu, K. Dantu, J. Caffrey, R. Govindan, and E. Johnson. Structural damage detection and localization using NetSHM. In Proceedings of IPSN/SPOTS'06, pp. 475–482, Nashville,TN, USA, April 2006.

[50] Chan, Albert WT. "Interactive toys." U.S. Patent No. 6,089,942. 18 Jul. 2000.

[51] Hameed, Shihab A., et al. "Car monitoring, alerting and tracking model: Enhancement with mobility and database facilities." *Computer and Communication Engineering (ICCCE), 2010 International Conference on*. IEEE, 2010.

[52] Cheng, Hong. "Vehicle Detection and Tracking." *Autonomous Intelligent Vehicles*. Springer London, 2011. 61-80.

[53] Pedersen, Mikkel Rath, et al. "Robot skills for manufacturing: From concept to industrial deployment." *Robotics and Computer-Integrated Manufacturing* 37 (2016): 282-291.

[54] Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web Services Description Language (WSDL) 1.1. W3C note, W3C (2001).

[55] S. E. Aldrich, "Anatomy of Web Services", Patricia Seybold Group, Inc. 2002.

[56] Emary, Ibrahiem MM El, and S. Ramakrishnan. *Wireless sensor networks: from theory to applications*. CRC Press, 2013.

[57] *Linear Technology Data Sheet LTC1051/LTC1053*; Linear Technology Corporation: Milpitas, CA, USA, 1990.

[58] Atmel Coorporation, "Atmel ATmega328P Datasheet," http://www.atmel.com/Images/Atmel-42735-8-bit-AVR-microcontroller-ATmega328-328P_Datasheet.pdf, August 2017.

[59] ITU, *ITU-R M.2134, Requirements related to technical performance for IMT-Advanced radio interface.* ITU, 2008.

[60] Safertal, F.; *GSM R - A New Standard for Rail Telecommunications*, ITS 13th European Regional Conference, 2002.

[61] A. Ogunsola and A. Mariscotti, *Electromagnetic Compatibility in Railways*. New York: Springer, 2012.