UNIVERSITY OF BEIRA INTERIOR
Faculty of Engineering

A Platform to Support the Development of Applications Based on the Segmentation and Labelling Algorithm

Sabrina Santos Guia

Dissertation to obtain the Master's Degree in
Electrical and Computer Engineering
(2nd study cycle)

Advisor: Professor Doctor António Eduardo Vitória do Espírito Santo
Co-advisor: Professor Doctor Vincenzo Paciello

Covilhã, October of 2015

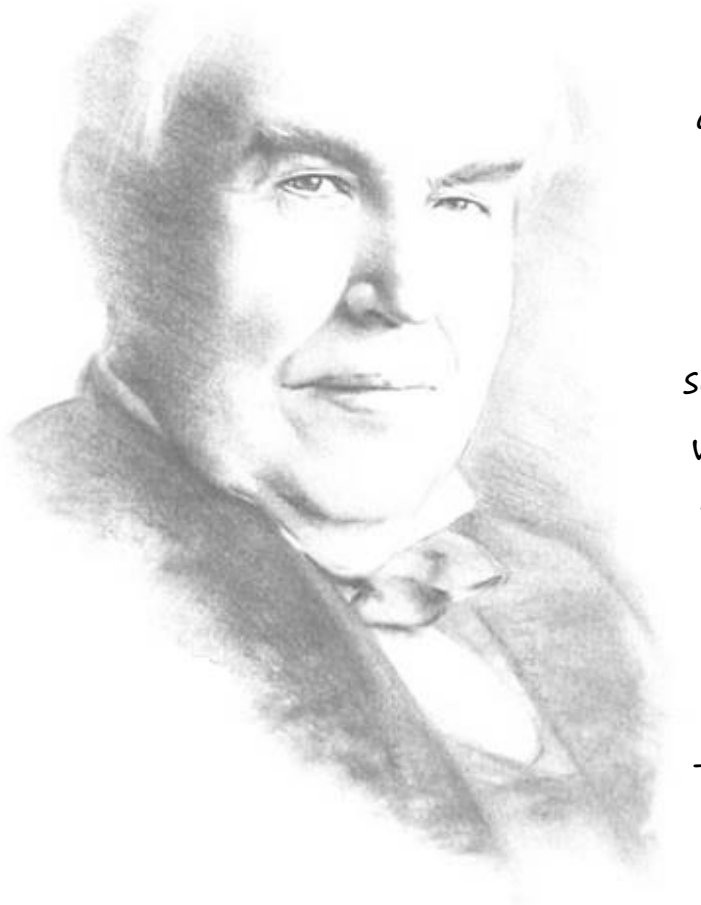
"I haven't failed...
I've just found 10,000 ways that won't work!"

and I learned that

"When you have exhausted all possibilities,
remember this: you haven't."

because

"Our greatest weakness lies in giving up!
The most certain way to succeed is always to try
just one more time."



and reminding that

"Just because
something doesn't do
what you planned it
to do doesn't mean
it's useless."

Thomas Alva Edison

Acknowledgements

To my advisor Professor Doctor António Eduardo Vitória do Espírito Santo and to my co-advisor Professor Doctor Vincenzo Paciello, for their availability to discuss and dispel doubts and uncertainties, for the sincerity and exemplary professionalism that they have shown. For the interest in showing me devices and electronic equipment and teach me new concepts, even though they were not related to the project in development. For encouraging the growth both personally and academically. I am also grateful for the scientific orientation, and for any suggestions made during the writing and development of this project and its final revision.

To Tânia Campos and Pedro Serra, for the sympathy and availability and patience in making all the linguistic revision.

To Tuna Feminina da Universidade da Beira Interior – "As Moçoilas", that beyond the good and bad times past have always supported and helped me. For having entrusted the post of association leader and have inculcated a way of life.

To Patrícia Melo and Fabiana Guilherme, by understanding and friendship that although the distance, always supported me throughout my academic career, always leaving me nostalgic for the moments with the Banda Musical de Gouviães and from Escola EB 2,3/S de Tarouca, respectively.

To Kelly Amaral, Hugo Oliveira, Tiago Mota, Joana Serralheiro, Tiago Ferreira, Marina Barbosa, Filipa Moreira, Elisabete Alves and Valéria Nascimento the endless hours of willingness spent in labour talks, but more importantly by the sincere friendship, support and patience in being with me in the hardest times. For the teachings transmitted not only over the degree in Bioengineering but also throughout the master's degree in Electrical Engineering and Computer in the Bionic Systems Branch.

To my parents for all the sacrifices in giving me always all it took, sometimes more than was possible to them, throughout my academic journey. I am also grateful for the affection and unconditional support, though at distance, were always by my side encouraging to never give up!

Finally, to all those who, indirectly, contributed to this work, in particular, all colleagues and professors from the *Universidade da Beira Interior* and all members from the *MISTRAL Laboratory* of the *Università Degli Studi di Salerno*, my most sincere acknowledgments.

Abstract

The need for networks of sensor that operate in an increasingly efficient way is contributing to boost the research methods with the aiming of find the best methods about how to manage the information. At the same time, the technological explosion that is taking care of our daily life, have influence not only on what is likely to be standardized, but also in the possibility of execution. An example of this is the need to exchange the knowledge about how the sensor's signal behaves in a standardized manner, as well as, some of its features and isolated parameters. The compaction of data to be managed and transmitted clearly becomes useful, especially when there is interest in the study of signal characteristics, such as; amplitude; the presence of noise; steady state values tendency, etc. Small parameter capable of induce the general behaviour of the measured signal.

In the world of sensors, intelligence is focused on the same point of measurement made by the transducer. From this point, three algorithms are presented. The first is based on the rapid measurement of procedural techniques of the Fourier transform (FFT), while the second is based on the compressive sampling theory (CS). Finally it is presented a proposed algorithm, in the time domain, based on the processes of segmentation and labelling (S&L) of a sampled signal that is also proposed for defining the new IEEE 1451 standard. The main goal of this work is focused on the management of a smaller amount of data, but the acquisition of the same amount and quality of information. Thus, the ideal is that the signal is sampled in order to create redundancy between neighbouring samples. It is therefore not necessary segmentation of each pair of samples, storing and transferring only the samples that are held to bring important information. As a result of the segmentation process, two vectors are obtained for storing amplitudes and time indices of samples and labelling process, the segments are classified according to their behaviour in eight different classes and stored in a third vector. This set of MCT vectors offers a structurally standardized platform that supports sensor's data exchange. The reconstruction of the acquired signal is also allowed by this structure. The method was implemented and each function and their respective operating modes are described in detail. In addition, input and output parameters of each function are also described. Afterwards, the project was prepared to implement in a microcontroller, whose architecture from ARM. In order to demonstrate the performance of the proposed algorithm, experimental results in the area of instrumentation, analysis and signal processing, are reported and exposed. After the execution of the above-mentioned procedure, the sequence segments analysis reveals that the algorithm has the ability to extract global information form the acquired signals, such as a human observer. In addition, its low

computational cost allows the inclusion of the proposed method in smart sensors in a variety of application, enabling execution in real time.

The document is divided into four chapters. In the first chapter is made a brief research concerning the state of the art related with the main subject of this work. Theoretical considerations concerning knowledge extraction from sample, analysis and data processing are presented. The best known algorithms are described in the second chapter special focus is given observing the application area. In the third chapter software and hardware tools used in the implementation process are described. The fourth chapter carries out the implementation of the proposed algorithm and described the respective implementation. The members of API functions are individually tested and the results presented and analysed. In the fifth, and final, chapter final observations are drawn with conclusions. Suggestion about future work is also presented.

Keywords

Smart sensors in networks, Internet of things (IoT), Software engineering, Standard IEEE 1451, ARM microcontroller implementation, Fast Fourier Transform (FFT) Algorithm, Compressive Sensing (CS) technique, Segmentation and Labelling (S&L) method, Signal sampling, MCT normalized platform, Acquisition, analysis and signal processing.

Resumo

A necessidade do uso de redes de sensores que operem de forma cada vez mais eficiente impulsiona os métodos de investigação destinados à melhoria da gestão de informação. Ao mesmo tempo, a explosão tecnológica que se apodera da nossa vida quotidiana, tem uma influência não só sobre o que é plausível a ser padronizado, mas também na sua possibilidade de execução. Um exemplo disso é a necessidade de intercâmbio de conhecimento do comportamento do sinal do sensor de forma padronizada, bem como de algumas das suas características e parâmetros isolados. A diminuição de dados a serem geridos e transmitidos torna-se claramente útil, em especial, quando há interesse no estudo de características dos sinais, tais como: amplitude, presença de ruido, tendência para valores em estado estacionário, etc., pequenos parâmetros capazes de induzir o comportamento geral do sinal medido.

No mundo dos sensores, a inteligência dos mesmos é concentrada no ponto de medição efetuada pelo transdutor. A partir deste ponto, são apresentados três algoritmos. O primeiro é baseado na técnica de procedimento de medição rápida da transformada de Fourier (FFT), enquanto o segundo é baseado na teoria de amostragem compressiva (CS). Por último apresenta-se uma proposta de algoritmo, no domínio do tempo, baseado nos processos de segmentação e rotulagem (S&L) da sobre amostragem de um sinal, também proposto para a definição do novo padrão do IEEE 1451. Sendo que o seu objetivo principal foca-se na gestão de uma quantidade menor de dados, mas na aquisição da mesma quantidade e qualidade de informação. Deste modo, o ideal será que o sinal seja sobre amostrado, por forma a criar redundância entre as amostras vizinhas. Não sendo por isso necessária a segmentação de cada par de amostras, armazenando e transferindo apenas as amostras que forem consideradas portadoras de informação importante. Como resultado do processo de segmentação, são obtidos dois vetores para o armazenamento das amplitudes e índices de tempo das amostras essenciais, e do processo de rotulagem, os segmentos são classificados, segundo o seu comportamento, em oito classes diferentes e armazenados num terceiro vetor. O conjunto com vetores, MCT, forma uma plataforma estruturalmente normalizada de apoio à cooperação de dados sensoriais. A mesma permite, também, assegurar a reconstrução do sinal adquirido. Este último foi implementado e cada função e seus respetivos módulos de operação foram descritos ao pormenor. Além disso, foram também indicados os respetivos parâmetros de entrada e de saída de cada função. Posteriormente, o projeto foi preparado para implantação num microcontrolador, cuja arquitetura é ARM. Resultados experimentais são relatados e expostos, com a finalidade de demonstrar o desempenho do algoritmo proposto, na área de aquisição, análise e processamento de sinais. Após execução do

procedimento anteriormente mencionado, a análise de sequência de segmentos revela que o algoritmo possui a capacidade de extrair informações globais, de sinais adquiridos, à semelhança do comportamento de um observador Humano. Além disso, o seu reduzido custo computacional permite a sua incorporação em sensores inteligentes destinados aos mais variados contextos de aplicação, viabilizando a sua execução em tempo real.

O documento está dividido em quatro capítulos. No primeiro capítulo é feita uma breve investigação do estado da arte, são apresentadas as considerações teóricas na área de extração de conhecimento a partir da aquisição, análise e processamento de dados. No segundo capítulo são descritos os algoritmos mais conhecidos no que diz respeito à aplicação na área de estudo. No terceiro, são mostradas as ferramentas de *software* e os dispositivos de *hardware* aplicados à respetiva implementação e implantação do projeto de programação. No quarto capítulo é realizada a implementação do algoritmo proposto e feita a respetiva implantação do mesmo. As funções integrantes da API são individualmente testadas e os resultados expostos e analisados. No quinto, e último, capítulo são feitas as considerações finais, tiradas as respetivas conclusões e apresentadas as sugestões de trabalho futuro.

Palavras-chave

Sensores inteligentes em redes, Internet das coisas (IoT), Engenharia de *software*, a norma IEEE 1451, Implementação num microcontrolador ARM, Algoritmo de Transformada Rápida de Fourier (FFT), Técnica de Amostragem Compressiva (CS), Método de Segmentação e Rotulagem (S&L), Amostragem de sinais, Plataforma MCT normalizada, Aquisição, análise e processamento de sinais.

Index

List of Figures

List of Tables

List of nomenclatures

f_s	Sampling Frequency
μs	Microseconds
A	Amplitude of The Signal
B	Frequency Band Limitation
c	Propagation Velocity
C	Classes[ ] vector
D	Vertical Offset
f	Signal Frequency
Hz	Hertz
k	Wave Number
kHz	Kilohertz
M	Marks[ ] vector
ms	Milliseconds
N	Complex Multiples
n	Even Number
N-1	Complex Adds
ns	Nanoseconds
r	Indices
s	Sparse representation
s	seconds
T	Times[ ] vector
Trend	Relation between Maxima and Minima Tendency
Trendmax	Tendency of Maxima
Trendmin	Tendency of Minima
W_n	Twiddle Factor
x	Input Signal
x(k)	Input Signal
X(k)	Input Signal
x(t)	Analog Input Signal
y	Vector of Acquired Samples
λ	Wave-Length
π	Pi Value
T	Signal Period
φ	Orthonormal Basis [nxn]

ψ	Angular Frequency
Ω	Highest Frequency
ω	Phase Shift
Φ	Sampling [nxn] Matrix

List of Acronyms

ACoMS	Autonomic Context Management
ADC	Analog Digital Converter
API	Application Programing Interface
ARM	Acorn RISC Machine
ASCII	American Standard Code For Information Interchange
CPU	Central Processing Unit
CS	Compressive Sensing
CSV	Comma Separated Value
DC	Digital Converter
DFT	Discrete Fourier Transformation
ECG	Electrocardiogram
FFT	Fast Fourier Transform
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
IS	International System of Units
IT	Information Technology
MCT	Marks, Classes and Times vectors
NCAP	Network Capable Application Processor
NIST	National Institute of Standards and Technology
OS	Operative System
POM	Point of Measurement
PQ	Power Quality
S&L	Segmentation and Labelling
SI	System of Information
SPIS	Strategic Planning Information Systems
TEDS	Transducer Electronic Data Sheet
TIM	Transducer Interface Module
URL	Uniform Resource Locator
VCS	Version Control Systems

Chapter

1

With the wider use of intelligent sensors, the need to apply them in network proved to be an asset, making them more efficient on data transfer by improving their methods of investigation. Considering the computational costs associated with data transmission, it will be quite useful to reduce the amount of data to send. So, the idea is to transmit less data and processing but acquire the same amount of information requiring fewer CPU resources. For this purpose, it is necessary to reduce the size of the sensor signal being processed and analyzed revealing how, in the world of sensors, intelligence is concentrated in measuring point within the transducer. In a final stage, it is also shown the organization of this document.

1. Introduction

The world, as it is known, is surrounded by new smart technologies that are rising every day, each one being related to a specific subject. All this electrical and electronic devices are used to solve different issues, ranging from the simplest one, such as domestic appliances, to the most complex that can be found in the industry. Sensors play an important role today. The way how they are adjusted depends from the context where they operate. In real-world applications, sensors are used for a wide range of purposes. One of those purposes is gathering data from the environment and, in this field, the communication among different types of devices becomes a necessity. To accomplish this task, embedded systems are being developed as Autonomic Context Management Systems (ACoMS) [1]. When in a context-aware application, a specific behaviour of an embedded system can be triggered if the system is described using a framework that models their sensing and data processing capabilities. Additionally, the appropriately setting of their dynamic composition enables, not only, their application in different context environments, but also, to perform the data transmission more effectively and efficiently. This translates the importance and necessity of communication between elements in both wired and wireless networks [2].

The application of sensors for complex signal processing, in the Point of Measurement (POM), requires improvements managing the information acquired by the sensors [1], [3]–[5]. A solution is to decrease the amount of data to process [1], [3]. This will aid, limited to the Central Processing Unit (CPU) characteristics, in monitoring the needed inputs whilst reducing waste, consumption, size, loss and costs. It will also allow us to know what to replaced or repair, and when, in the POM. In summary: to convert the information to a better format, with reduced size, in order to improve the performance of communication and storage mechanisms [1], [3]. However, the design of ACoMS poses several challenges [1]:

- **Integration:** ability to fit in multiple computing environments enabling the sensor to respond to a wider range of applications;
- **Scalability:** responsiveness of the data management collected, regardless of its size, amount or even its contextualization environment from which are acquired by the sensor;
- **Interoperability:** configuration capability of the communication protocols in order to develop a more effective response, regardless of the context environment in which it is applied to a measuring for a posterior representation of the information;
- **Flexibility:** data collection capacity with similar characteristics (such as type, data size, quantity, etc.) from different types of context sources in which the sensor is applied;

- **Reliability and Availability:** attempting to implement the above capabilities may give rise to a number of potential losses of information or communication failures of the sensor.

Bearing in mind the objectives and operation policies, during the development of a system with an autonomic computing vision, the goal is to create a self-managing system that seamlessly supports both low-level sensing infrastructures and high-level adaptive applications. At the same time, the idea is to provide to the system, the development of an effective response for better performance. To provide that, it is necessary to optimize the operation of running in time domain and, to create the capacity of configuration and/or reconfiguration, of the application settings, on different levels of programming complexity [1], [3]. Parallel to the development of sensory instrumentation of measurement, the development of communication interactions contribute to the interoperability between devices of a common sensors network, whether wired or wireless, platforms and/or applications [6].

1.1. Work Background and Motivation

Several standards for sensor data processing already have been presented. The main goal was to extend the interoperability among sensors. Considering this, the National Institute of Standards and Technology (NIST) has proposed a family of standards as a universal solution [1], [3], [7]. This normalization effort establishes a standard of response to the challenge to preform connections and observed communication protocols among smart sensors, transducers and actuators, in different kinds of networks, wired or wireless, applied to different systems in numerous contexts. The IEEE 1451 family is a set of interfaces for smart transducers that define, not only a solution to the previously described challenges but also as a way to establish independent connections between those transducers, such as architecture description, message formats and software objects [1], [8], [9]. Given that, besides the instrumentation needed, it is also necessary to define the networks' control. An example of that is the signal analyses and processing field, where algorithms are required to process the acquired data in order to reduce its amount to subsequent transmission. As a complement to this, an electronic datasheet was defined to contribute for the sensor interoperability achieved with the plug-n-play architecture [6]. This datasheet intends to replace the paper version and includes, among many others, transducer identification, measurement units, data rage and scale, manufacturer-related information for the specific sensor, bandwidth and calibration details [3], [10]. When combined with the IEEE 1451 standards family, this electronic version of the datasheet is named Transducer Electronic Data Sheet (TEDS) and can present different levels of complexity [1], [9]. This combination allows not only the signal synthesis and analysis, but also the connection and communication among smart transducers

to validate the measurement procedure and exchange of information. Furthermore, it provides a platform that promotes the fusion of sensors knowledge. All the information acquired through data mining is by this way optimized and, as a consequence, so is the interaction between sensors and actuators with the real world [4], [11]. Embracing plug-n-play technology and its application to sensors, users can experience increased benefits, such as [6]:

- More atomized and faster systems;
- Faster and simpler system setup and configuration;
- Improved efficiency and effectiveness of the measurements accuracy;
- Increased diagnostics and troubleshooting;
- Decreased time for repairing and/or replacing sensors;
- Optimized sensor data management, bookkeeping and inventory management;
- Automated use of calibration data for signals analysis and processing.

In this line of reasoning, the Internet of Things (IoT) is a new way to enable the exchange of signal information among sensors and transducers, from different manufacturers, in order to establish a dialogue among them. The information obtained from the sensors raw data and from the network states of the instrumentation helps to achieve higher reliability, since the key to achieve efficient and effective solutions for computer science problems is data representation [3], [4], [11]. As the main objective, the sensor needs to be smart enough to measure signal parameters. These led to the recognition of signal characteristics, allowing to study their general behaviour and to conclude about the type of the measured signal. Algorithms of signal analysis and processing can easily conceive this idea. For example, it is essential to know where signal segments are suitable for additional or time consuming processes like: high order statistics, wavelet, fuzzy logic, neural networks, spectrum analysis and any other signal processing technique. In other situations, such tasks are not adequate. For example, when the signal is constant [11]. Therefore, it is of great interest to know a few noteworthy information about the signal's behaviour like: shape form; periodicity; amplitude; time constant; steady state value; tendency and future values; sampling frequency; mean and variance values; occurrence of impulsive noise and detection of patterns [3], [5], [11].

With all this, it is clear the importance that information has in our everyday lives. At the same time, the impact of Information Technology (IT) is a key element in achieving strategic and competitive advantages in the world of industry. Because of that, their respective implementation should be strategic and carefully planned and structured. Therefore, for us to take advantage of the potential of new technologies, it is necessary to make a large investment in software and hardware.

A System of Information (SI) is defined as an integrated set of resources (human and technological) whose objective is the proper satisfaction of all the information needs of software that wish, in short, process different inputs and produce various outputs in order to meet the objectives outlined. There is therefore a fundamental of the following characteristics:

- **Flexibility:** ability to evolve face the prerequisites;
- **Reliability:** reducing the occurrence of problems during operation;
- **Organization:** regards implementation needs;
- **Interaction:** ease of use, allowing for a more friendly and intuitive interface.

The proper functioning of the systems is crucial and because of that, before starting any process of development of information systems architecture components, it is crucial that it can be conceived from a global point of view. Thus, it is possible to ensure complete integration between the components and the prioritization of their implementation, which is the working part of a Strategic Planning Information Systems (SPIS). The SPIS defines the components of a SI, working as a guide for all future interventions. As a process, after the definition of the global strategy and the identification of components needed to develop their achievement goes to the field of software engineering. This area receives information support from the following areas: systems analysis, project management, scheduling and quality control. Thus, the activities associated with software engineering can be grouped into three broad phases, according to its goal of development and operation: design planning, implementation and maintenance.

Developing a software is a very complex process. For this reason, it is divided into a set of activities less complex. Its uniform execution, partially ordered, leads to the production of software. As for the study of its features, this is ensured by software engineering studies. Therefore, to obtain a quality software, it is necessary to respond and accomplish the technical requirements and the following mechanisms from Figure 1.1:

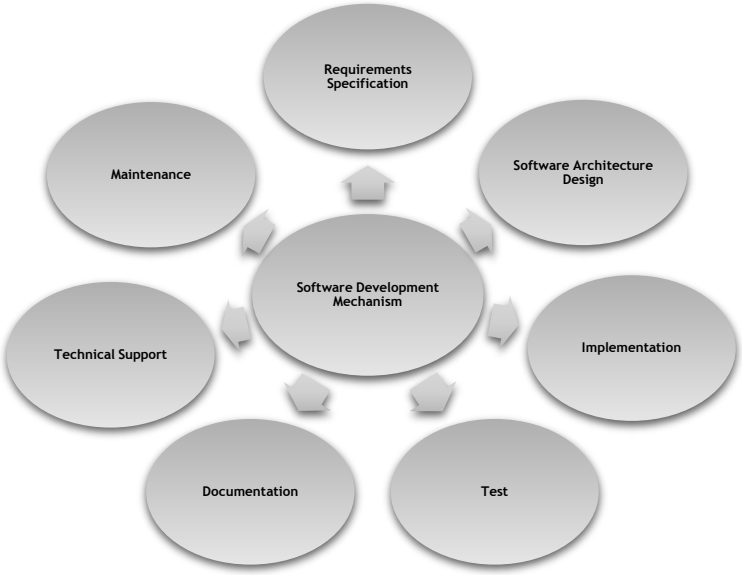


Figure 1.1. Stages of software development mechanism.

The requirements of each software development mechanism are described in Table 1.1:

Table 1.1. Software development mechanisms requirements description.

Mechanisms of Software Development	Description
Requirements Specification	Describes the software that will be written accurately, preferably in a mathematically rigours way. Therefore, the activities involved include identifying: ▪ The general purpose of the system; ▪ The project objectives; ▪ The technical restrictions; ▪ The problems and associated risks; ▪ The definition of the project control process; ▪ The most relevant functions and implementation of alternatives, carrying out their assessment and selection; ▪ The presentation of findings and recommendations with technical justification, functional and financial; ▪ Preparing and obtaining the approval of the project plan (identification of tasks, making diagrams, etc.).

Software Architecture Design	Abstract representation of the system meets the product requirements, but also ensures that future requirements can be satisfied. Directs the interfaces between software systems and other software products, as well as with the basic hardware or operating system.
Implementation	Transformation of the project in source code. Although, it is the most obvious part of the software engineering work, but does not mean necessarily that is the major portion of it.
Test	Test, step by step, each part of the system. Several activities ate performed in order to validate the software product in different situations, testing each feature of each module, taking into account the specification made in the design phase. As result, the test report contains all relevant information about the founded errors in the system, and their behaviour in various aspects.
Documentation	Serves as technical support to internal software project for future purposes, such as maintenance and enhancements.
Technical Support	Development of technical material to support the implementation of the software by users. The datasheets and user guides are great examples of this material. The great advantage of this phase is making of the users to submit questions and identify problems in the software performance, which leads to the next stage.
Maintenance	Reviewing and improving the software, dealing with the problems and requirements of the previous stage. It may take more time that the one spent on the initial development phase. Apart from the possibility of being required to add code to match the original design, it may also be necessary to determine how the software will work after maintenance.

That said, it can be concluded that these are necessary to support certain types of technical development, such as:

- Model before performing;
- Estimate several factors before proceeding;
- Test and measurement before, during and after production;
- Carry out the study and analysis of risk factors.

The applications of these techniques must be rigorous, systematic, efficient and controllable. Naturally, the effectiveness of this process also instils the use of tools to support the process. Thus, the benefits associated with innovative functional use are:

- Reduction of operating costs by automating and reformulation processes;
- Satisfaction of information requirements of users;
- Contribution to the creation of new products and services;
- Improving the effectiveness of decision support methods;
- Improve and increase the level of service provided to customers;
- Improve and automate the performance of electronic devices.

This document is structurally divided into five chapters. The first chapter conducted a background study and work motivation to the present project proposal. The second chapter will be made respective theoretical considerations in the area of acquisition, analysis and signal processing. There will also be presented some examples of techniques and theorems applied to the study area. In a final stage, will be also described a proposal of new algorithm for the same purpose. In the third chapter, it will be briefly exposed both hardware and software tools applied to support the development and production of the project, flaunting their environments and interfaces, as well as its features and specifications. In the fourth chapter, it will be performed a depth study of the algorithm performance and of the integral parts of the project, that is, not only will be described the specifications of each Application Programming Interface (API) function, but also an experimental setups and their respective analysis. In the fifth and final chapter, the final remarks will be made, as well as the general conclusions of the project and suggestion of some future prospects.

Chapter

2

Parallel with the implementation of various patterns, these are also being developed to extend a greater interoperability between sensor systems. Here, will be explored a subset of multiple sensors standards. The IEEE 1451 is a set of standards developed to unify the various standards and protocols, providing a basic protocol. It will be studied the characteristics of the IEEE 1451.0 standard, where the communication of all networked transducers is ensured by TEDS, with the same format for all sensors and actuators for both networks, wired or wireless. They will also be discussed some of the more popular methods of acquiring and presented a more effective proposal to effect answering the need to reduce computational cost while maintaining the same amount of processed information.

2. Theoretical Considerations on Sampling and Measurement

Traditionally, the sampling technics follow the Nyquist-Shannon theorem. This allows a reliable reconstruction and/or reproduction of the signal, image or video. If a signal is to be reconstructed from acquired samples, the signal should have a finite bandwidth and must be sampled with a sampling frequency (f_s) at is least equal to twice highest frequency (Ω) presented in the signal. It is possible to express this requirement using equation (2.1) [12]:

$$f_s = 2 \times \Omega \tag{2.1}$$

In most cases, the signals doesn't owns the requisite aforementioned, but nonetheless, despite the absence of information affecting the tail side of the spectrum, generally, the error resulting in the reconstruction of the signal is acceptable. To solve this problem a filter is used, before the digital to analogue converter, with an acceptable f_s as mentioned above [12]. The notion of taking less data but acquiring the same information does not apply to the acquisition of the whole waveform, instead of that, the algorithm only keeps some of the samples. Each sample carries a part of the whole information. This notion proves to be enough for the execution of the proposed algorithms for analysis and signal processing for sensors developed for both wired or wireless networks, with intelligence distribution of intelligence and computational capabilities [3]. Standards based approaches offer, despite some disadvantages, advantages over proprietary solutions as presented on Table 2.1 [8]:

Table 2.1. Standards advantages and disadvantages.

Advantages	**Disadvantages**
Sensor architecture defined;	Firmware programming in object-oriented language style;
Software flexibility;	Increase of the algorithm complexity in case of the embedded controller doesn't support this type of language;
Firmware functions;	
Universal applications.	Become burdened by the amount of memory required to instantiate objects in the firmware.

The literature presents several algorithms to overcome the bridging gaps created by the disadvantages presented. Among them, as was previously mentioned, there are different levels of complexity that characterize the algorithms. Within the most complex, as the best known it stands out the Fast Fourier Transform (FFT) algorithm, but there are many others, like for example the Compressive and Sensing (CS) algorithm. Observing low computational algorithms requirement, there is an algorithm that by Segmentation and Labelling (S&L) of acquired samples, during the sampling process, facilitates the raw data mining process by having the same signal information distributed by fewer samples (the most significant). Additionally, based on the sensor information, allows the adaptation of algorithm to the application context in which it is inserted [3], [10].

The next three subsections will present a description of how these algorithms work using a smart sampling process. The first one, is a well-known measurement procedure [2]. The second one, is a technique that takes, randomly, a fixed number of samples from a signal [3]. The third one, a new algorithm is proposed for information extraction from samples of a signal. Suited for data mining process and adapted to sensor signals where a control algorithm is applied based on a technique that process acquired signal samples in order to generate a set of three vectors, constituted by raw information [3], [10]. Divided into two different stages, segmentation and labelling, it will be possible to understand how it is possible to extract parameters from an oversampled signal, allowing the induction of its global behaviour by applying six distinct functions presented in the fourth chapter of this document [4], [11].

2.1. Fast Fourier Transform

The present algorithm, one of the most important amid the best known, presents as the main goal to calculate efficiently the frequency contents of signals. Being this an algorithm developed for analysis and processing signals through microcontrollers and microprocessors, is a fast computational algorithm to perform numerically and an efficient way for the calculation of the Discrete Fourier Transform (DFT), where its big difference is the time complexity. The basic concept is taking an array in time domain, T_d seconds, of samples and processes it using the Fast Fourier Transform (FFT) algorithm, in order to produce a new array in frequency domain, B_b Hz (bidirectional bandwidth). This last one is a spectrum of samples. Both arrays present exactly the same number of samples. Real value samples are received by the input of the algorithm that return in the output side complex value samples [13]–[15]. Actually, there are several ways to implement the FFT algorithm, although some require many resources, other are more simply and not optimized. Thus, through the literature, the idea is to describe, briefly, how to perform an analysis and processing of a signal using an implicit Fourier analysis on mathematical methods. Therefore, the chosen technique, presented by Cooley and Tukey in 1965, the DFT can be determined using equation (2.2) [14]:

$$\boldsymbol{X(k)} = \sum_{\boldsymbol{n=0}}^{\boldsymbol{N-1}} \boldsymbol{x(n)} \times \boldsymbol{e^{-i\frac{2\pi}{N}kn}} \tag{2.2}$$

Assuming that $\mathrm{x(n)}$ and $\mathrm{X(k)}$ are periodic input signal, and exploring the symmetries of the complex $\mathrm{e^{-i\frac{2\pi}{N}kn}}$, conventionally defined as a twiddle factor $\mathrm{W_N = e^{-i\frac{2\pi}{N}}}$ and this is like saying that in an a unit circle from Figure 2.1 one has that [14]:

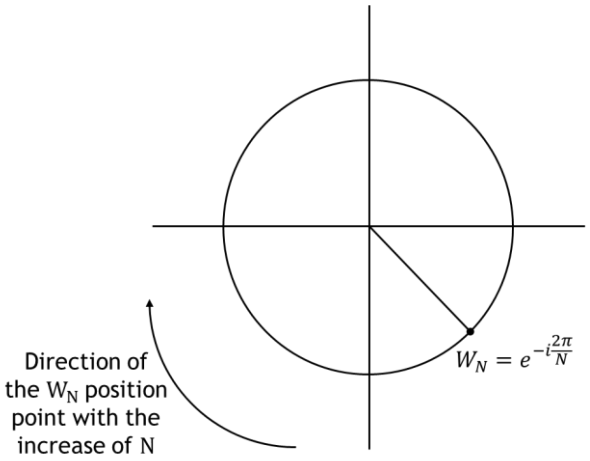


Figure 2.1. W_N representation on a unit circle.

Then, in equation (2.2), the twiddle factor can be conjugated to get equation (2.3) [14]:

$$\boldsymbol{X(k)} = \sum_{\boldsymbol{n=0}}^{\boldsymbol{N-1}} \boldsymbol{x(n)} \times \boldsymbol{W_N}^{\boldsymbol{kn}} \tag{2.3}$$

Where $\mathrm{k} \in \mathbb{R}^{\mathrm{n}}$ can assume values on the interval 0 up to $\mathrm{N-1}$ considering how much computational is involved, it takes $\mathrm{N^2}$ complex multiplies and $\mathrm{N(N-1)}$ complex additions, to get each $\mathrm{X(k)}$ value. In this way, considering $\mathrm{O(N^2)}$ order of computations for direct DFT and $\mathrm{O(Nlog_2N)}$ order for FFT, it can be seen on Table 2.2, the FFT will be reduce considerably [13], [14]:

Table 2.2. Time to, computationally, calculate the orders of DFT vs. FFT.

Order of Computations	**Necessary Number of Operations to Compute FFT**		
N	10^3	10^6	10^9
N^2	10^6	10^{12}	10^{18}
N $\log_2$ N	10^4	$20*10^6$	$30*10^9$

Seeing each operation in nanoseconds (ns) that is, each operation took a nanosecond, 10^{18}ns are approximately 31.2 years whereas 30×10^9ns are 30s. Given this, it can be concluded that

how much greater is w_N^{kn}, bigger is the difference between both orders. Consequently, in order to reduce the number of operations to the order $O(N\log_2 N)$ the FFT algorithm is applied. To make it possible, the idea is to do decompositions into smaller DFTs; execute the simplifications like w_N^{kn} which is equals to 1 and $w_N^{\frac{N}{2}(k)}$ which is equals to -1. This will reduce computation complexity; reminding the periodicity on the input, assimilate that $w_N^{n(k+N)} = w_N^{k(n+N)} = w_N^{kn}$ and $w_N^{k(N-n)} = w_N^{-kn} = (w_N^{kn})^*$. After these steps, the first phase can be called as decimation in time, where n is an even number. In this way, considering equation (2.3), lets split the sum into the even entries of x, to get equation (2.4) [14]:

$$\boldsymbol{X(k)} = \sum_{\boldsymbol{n\ even}} \boldsymbol{x(n)} \times \boldsymbol{W_N}^{\boldsymbol{kn}} + \sum_{\boldsymbol{n\ additional}} \boldsymbol{x(n)} \times \boldsymbol{W_N}^{\boldsymbol{kn}} \tag{2.4}$$

If n is an even number, it can be consider as $2r$, resulting in equation (2.5) [14]:

$$\boldsymbol{X(k)} = \sum_{\boldsymbol{r=0}}^{\boldsymbol{\frac{N}{2}-1}} \boldsymbol{x(2r)} \times \boldsymbol{W_N}^{\boldsymbol{2rk}} + \sum_{\boldsymbol{r=0}}^{\boldsymbol{\frac{N}{2}-1}} \boldsymbol{x(2r+1)} \times \boldsymbol{W_N}^{\boldsymbol{(2r+1)k}} \tag{2.5}$$

At this point, what was done is rename the index calling it Radix-2. Mathematically simplifying equation (2.5) it is obtained equation (2.6) [14]:

$$\boldsymbol{X(k)} = \sum_{\boldsymbol{r=0}}^{\boldsymbol{\frac{N}{2}-1}} \boldsymbol{x(2r)} \times \boldsymbol{(W_N^2)^{rk}} + \boldsymbol{w_N^k} \sum_{\boldsymbol{r=0}}^{\boldsymbol{\frac{N}{2}-1}} \boldsymbol{x(2r+1)} \times \boldsymbol{(W_N^2)^{rk}} \tag{2.6}$$

Looking again to the unit circle, from Figure 2.1, but this time replacing the value N by 8, results in Figure 2.2 [14]:

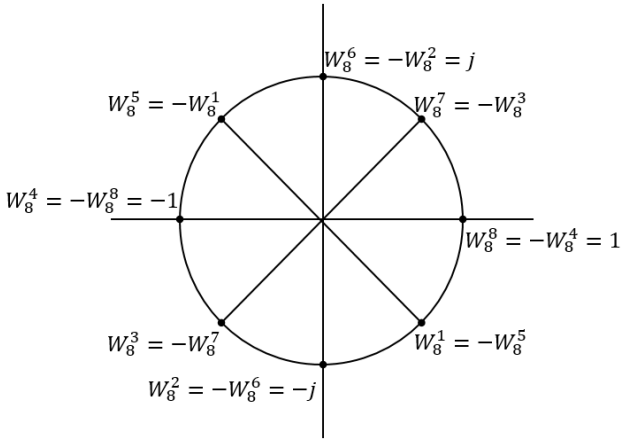


Figure 2.2. W_N representation on a unit circle with N equal to 8.

Observing Figure 2.2 it can be modified equation (2.6) so they look like two shorter DFTs represented by equation (2.7) [14]:

$$\boldsymbol{X(k)} = \sum_{\boldsymbol{r=0}}^{\frac{\boldsymbol{N}}{\boldsymbol{2}}-\boldsymbol{1}} \boldsymbol{x(2r)} \times (\boldsymbol{W}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}})^{\boldsymbol{rk}} + \boldsymbol{w}_{\boldsymbol{N}}^{\boldsymbol{k}} \sum_{\boldsymbol{r=0}}^{\frac{\boldsymbol{N}}{\boldsymbol{2}}-\boldsymbol{1}} \boldsymbol{x(2r+1)} \times (\boldsymbol{W}_{\frac{\boldsymbol{N}}{\boldsymbol{2}}})^{\boldsymbol{rk}} \tag{2.7}$$

Where $\sum_{\mathrm{r=0}}^{\frac{\mathrm{N}}{2}-1} \mathrm{x(2r)} \times (\mathrm{w}_{\frac{\mathrm{N}}{2}})^{\mathrm{rk}}$ is the length $\frac{\mathrm{N}}{2}$ DFT of even entries and $\sum_{\mathrm{r=0}}^{\frac{\mathrm{N}}{2}-1} \mathrm{x(2r+1)} \times (\mathrm{W}_{\frac{\mathrm{N}}{2}})^{\mathrm{rk}}$ is the length $\frac{\mathrm{N}}{2}$ DFT of additional entries. Each one can be written on a compact way $\mathrm{G(k)}$ and $\mathrm{H(k)}$, respectively, getting equation (2.8) [14]:

$$\boldsymbol{X(k)} = \boldsymbol{G(k)} + \boldsymbol{w}_{\boldsymbol{N}}^{\boldsymbol{k}}\, \boldsymbol{H(k)} \tag{2.8}$$

At this point, considering that one wants to compute N equals to 6, $\mathrm{k} = [0, 2, 4, 6]$ are the even entries and $\mathrm{k} = [1, 3, 5, 7]$ are the additional entries, to get the final $\mathrm{X(k)}$ where $\mathrm{k} = [0, 1, 2, 3, 4, 5, 6, 7]$. With this, the DFT value calculation is shown on Figure 2.3 [13], [14]:

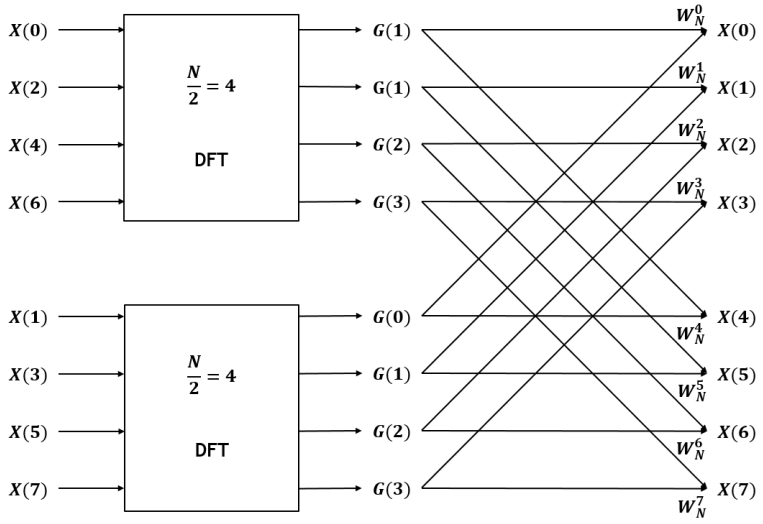


Figure 2.3. Radix-2 Fast Fourier Transforms, Butterfly algorithm.

There are no need to stop here. Keeping the same even and additional entries, it is still possible to simplify Figure 2.3 as shown on Figure 2.4 [13]–[15]:

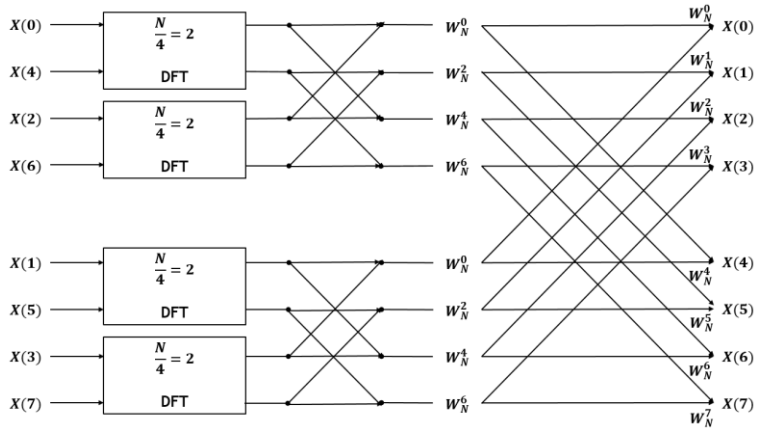


Figure 2.4. Radix-4 Fast Fourier Transforms, Butterfly algorithm.

Now, confronting Figure 2.4 with Figure 2.2 and replacing the required values, it is obtained Figure 2.5 [13], [14]:

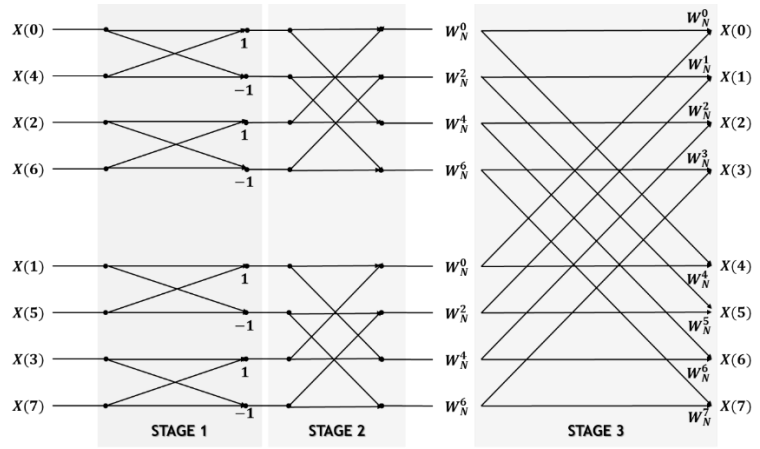


Figure 2.5. Substitution by the unit circle value and divide in three stages of decomposition.

With this, it is possible to conclude how much decomposition it is possible to do. If N is a power of 2, it is decomposed the DFT into $\log_2 N$ stages. Which means that is achieved $N\log_2 N$ computational multiplies. This explains the previously mentioned speeds difference between the DFT and the FFT, making the FFT algorithm the fastest in a computational way. Bearing in mind the computational storage, to compute the length of DFT, p^{th} and q^{th} values in the $(m-1)^{st}$ (first) stage are used to get p^{th} and q^{th} values in the m^{th} stage. In this way, it can be done by what is called “in place” which means no extra storage. Then, to arrange the input value may be placed in an algebraic perspective, getting the Figure 2.6 [13], [14]:

$$\begin{bmatrix} X(0) \\ X(1) \\ X(2) \\ X(3) \\ X(4) \\ X(5) \\ X(6) \\ X(7) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & W_8^1 & W_8^2 & W_8^3 & -1 & -W_8^1 & -W_8^2 & -W_8^3 \\ 1 & -j & -1 & j & 1 & -j & -1 & j \\ 1 & W_8^3 & j & W_8^1 & -1 & -W_8^3 & -j & -W_8^1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -W_8^1 & W_8^2 & -W_8^3 & -1 & W_8^1 & -W_8^2 & W_8^3 \\ 1 & -W_8^2 & -1 & W_8^2 & 1 & -W_8^2 & -1 & W_8^2 \\ 1 & -W_8^3 & -W_8^2 & W_8^1 & -1 & W_8^3 & W_8^2 & W_8^1 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \\ x(5) \\ x(6) \\ x(7) \end{bmatrix}$$

Figure 2.6. Algebraic representation of the computational storage.

Appreciating the grey background, both are equal but negative to each other, except for the first column only. It is possible to rewrite it in a simple way as given on Figure 2.7 [12]:

$$F_8 = \begin{bmatrix} I & I \\ I & -I \end{bmatrix} \begin{bmatrix} I & & & \\ & \begin{bmatrix} 1 & & & \\ & W_8^1 & & \\ & & W_8^2 & \\ & & & W_8^3 \end{bmatrix} \end{bmatrix} \begin{bmatrix} F_4 & \\ & F_4 \end{bmatrix} \begin{bmatrix} X_{even} \\ X_{additional} \end{bmatrix}$$

8 by 8 Fourier Matrix	Sums and Differences in "Butterfly"	"Twiddle Factors"	Length 4 DFTs	The even entries and the additional entries

Figure 2.7. Calculating the Fourier matrix.

Now, taking a time domain signal, it can be separated it in smaller and smaller pieces by applying a strategy called Decimation in Frequency FFT. Considering again equation (2.1), the even samples of $X(k)$ are given by equation (2.9) [13]:

$$\boldsymbol{X(2r) = \sum_{n=0}^{N-1} x(n) \times W_N{}^{2nr}} \tag{2.9}$$

Where $r \in \mathbb{R}^n$ can assume values in the interval from 0 up to $\frac{N-1}{2}$. Splitting equation (2.8) into two half parts of elements, getting equation (2.10) [13]:

$$\boldsymbol{X(2r) = \sum_{n=0}^{N/2-1} x(n) \times W_N{}^{2nr} + \sum_{n=N/2}^{N-1} x(n) \times W_N{}^{2nr}} \tag{2.10}$$

Preforming the calculations, obtaining equation (2.11) [13]:

$$X(2r) = \sum_{n=0}^{N/2-1} x(n) \times W_{\frac{N}{2}}^{nr} + \sum_{n=N/2}^{N/2-1} x(n+\frac{N}{2}) \times W_N^{2\left(n+\frac{N}{2}\right)r} \quad (2.11)$$

Again, can be made a few simplification to get equation (2.12) [13]:

$$X(2r) = \sum_{n=0}^{N/2-1} x(n) \times W_{\frac{N}{2}}^{nr} + \sum_{n=N/2}^{N/2-1} x(n+\frac{N}{2}) \times W_{\frac{N}{2}}^{nr} \quad (2.12)$$

That can be simplified as expressed by equation (2.13) [13]:

$$X(2r) = \sum_{n=0}^{N/2-1} [x(n) + x\left(n+\frac{N}{2}\right)] \times W_{\frac{N}{2}}^{nr} \quad (2.13)$$

This is like a $\frac{N}{2}$ DFT of summed input. The additional samples of $X(k)$ can be showed by equation (2.14) [13]:

$$X(2r+1) = \sum_{n=0}^{N/2-1} [x(n) + x\left(n+\frac{N}{2}\right)] \times W_N^n \times W_{\frac{N}{2}}^{nr} \quad (2.14)$$

Finally, it is obtained the decimation in time that both of them are most likely equivalent. As a conclusion, it can be done a series of DFT just by performing some stages and for each stage it can be made as showed on Figure 2.5. The number of necessary stages is directly dependent of the DFT length [13], [14].

2.2. Compressive Sensing Technique

The mains goals of the present algorithm aims to investigate the applicability of the CS technique in Power Quality (PQ) measurement. In particular, it will be study the ability of the given measurement accuracies comparable with standard requirements and typical PQ measurement methods; the capacity to reduce the required memory with respect to traditional measurement methods; and the transformation of raw data information in other domains [3], [12]. In this way, it is possible to represent the signal using a small number of samples that are acquired with a step of aleatory sampling. In order to apply this technique, to enable their representation on is destiny domain, it is necessary to choose the right expansion basis for m samples acquired. These m samples are just a few samples, and from them, only a few are meaningful. The Figure 2.8 represents the acquisition of the same signal (a) using two different techniques: (b) acquisition of n samples equally spaced and observing the Nyquist-Shannon theorem and (c) acquisition of m random samples using the CS technique [3].

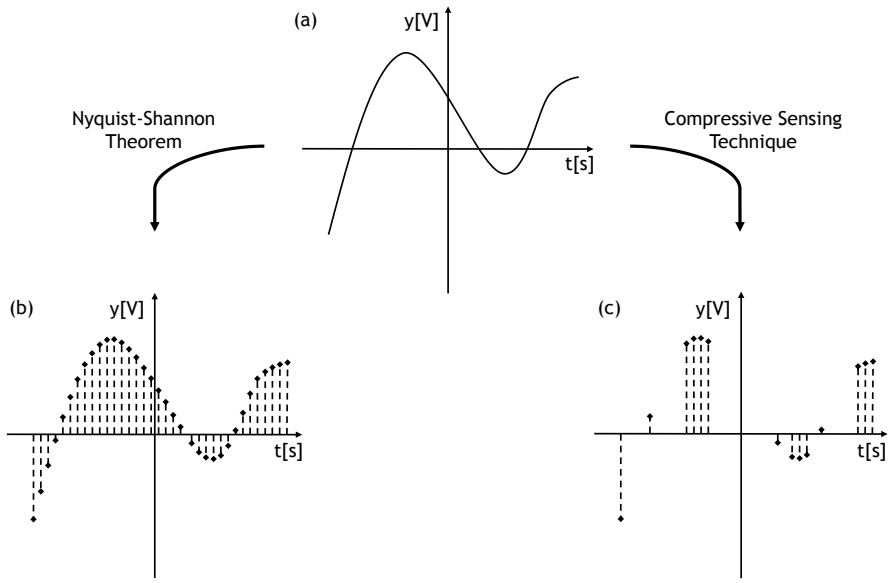


Figure 2.8. Representation, in time domain, the acquisition of samples from the signal (a), observing the Nyquist-Shannon theorem (b) and using the Compressive Sensing technique (c).

Considering the time domain, it is possible to express the mechanism of sampling using equation (2.15) [3], [12]:

$$\boldsymbol{y} = \boldsymbol{\phi} \times \boldsymbol{x} \tag{2.15}$$

Where $y \in \mathbb{R}^m$ represents the vector for measures, $x \in \mathbb{R}^n$ correspond to the input signal and ϕ is a sampling $[n \times n]$ matrix [11]. Comparing the number n and m of samples acquired, m is much smaller than n ($m \ll n$) but samples are enough to reconstruct the signal, as explained before. Given by this, the CS algorithm can reduce the amount of data being transmitted but, simultaneously, having exactly the same information as it was previously mentioned above as being an ideal idea. The compression of data is available due to the fact of having a wide set of signals that have a sparse representation. With this in mind, it is possible to express $x \in \mathbb{R}^n$ using equation (2.16) [3], [12]:

$$\boldsymbol{x} = \boldsymbol{\varphi} \times \boldsymbol{s} \tag{2.16}$$

Where $s \in \mathbb{R}^n$ correspond to a sparse representation of the signal x on the orthonormal basis $\varphi\,[n \times n]$. Comparing the CS algorithm with the traditional techniques for data compression, Nyquist-Shannon sampling theorem, it is noted that the transformed signal s is then computed from x, which means a proper orthonormal bases, and the sparse significant coefficients are adaptively singled and retained [12]. With this, on the signal reconstruction stage, the original signal is obtained by inverse transformation of those coefficients [11]. Combining the two previous equations (2.15) and (2.16), it is now possible to express the vector of the

acquired samples $y \in \mathbb{R}^m$ according to the representation of the sparse representation of the signal x, using equation (2.17) [2]:

$$\boldsymbol{y} = \boldsymbol{\phi} \times \boldsymbol{\varphi} \times \boldsymbol{s} = \boldsymbol{Z} \times \boldsymbol{s} \tag{2.17}$$

The system of equations presented above has an infinity number of solutions. To perform a CS with efficient success, the matrices ϕ and the orthonormal basis matrix φ need to be incoherent. Eventually, if the s is sufficiently sparse and the system of equations can be solved, providing a single solution [2]. By this is possible to affirm that the CS algorithm allows both acquisition and compression operations on a unique step and, accurately reconstruct a signal from a few samples [11].

2.3. Segmentation and Labelling Method

The increased demand for intelligent systems is able to provide more and better capabilities and features encouraged by the development of smart sensors and embedded devices that are capable of communicating directly with each other's. Thus, the exchange of data measured by the system itself, introduced more flexibility to the intelligent transducer, thereby improving overall system performance [9]. Intelligent devices such as sensors and actuators with digital outputs have been improved, not only in terms of performance but also in terms of reducing computational cost. By this way, the smart sensors components, including signal conditioners, communication microcontrollers and electronics have also been improved [7]. As grounded already, in Chapter 1 of this dissertation, the family of IEEE 1451 standard defines a set of communication interfaces to interconnect electronic devices such as sensors, actuators and transducers with systems with embedded microprocessors. In addition, they also provide protocols used by applications in both networks, wired or wireless [16]. The IEEE 1451.0 standard provides a set of common features, simplifying the creation of standards for different interfaces. At the same time the interoperability among members of this family is maintained [9], [16]. This interoperability is ensured by the TEDS, because it contains the necessary calibration data and operating methods to create a calibrated results observing the standard International System (IS) of units [7]. The main objective of this set of standards is to structure the interoperability of elements that integrate a system allowing the existence of transducers with plug-and-play features. With this, it is possible to reduce or even eliminate errors of Human origin, which sometimes occur. The IEEE 1451.0 defines the actions that must be performed by a Transducer Interface Module (TIM) and the common features to all devices implementing it [17]. Furthermore, this family of standards also specify formats for the TEDS and defines a set of commands to facilitate control of the TIM, as well as, also provide supplementary information to the application every time the systems needs to read or write data. The information is passed between the Network Capable Application Processor (NCAP) and the TIM through the software interface on a common hardware. The software

interface is defined by the IEEE 1451 standard. The task of adding new transport interfaces is simplified. In order to enable communications, it was created the API [7], [9]. In this section, a new method for analysis and signal processing will be presented, as well as, the evaluation of its performance. The S&L algorithm names the method and formulates a proposal for a new version of the IEEE 1451 standard [2].

The idea of performing analysis and processing a digital signal involves the extraction of knowledge from raw data. To produce this effect, as have been described in the previous sections, there are several methods. Some of them are more efficient than others. Despite its advantages and disadvantages, generally, they are all very sensitive to the presence of noise and other artefacts in the signal. The majority of signal analysis algorithms focuses on acquiring data and apply directly processing methods, in order to extract the data information. The method presented here attempts to go beyond, proposing an algorithm applicable to smart sensor, whose performance must be the image and likeness of a human being. In other words, the main objective is that the algorithm recognizes a signal and gets the same conclusions estimated by the Human observer [4], [11]. To achieve this task, the algorithm focuses on getting an intermediate stage after signal acquisition. This consists of introducing a processed data structure to subsequently allow for their examination [17].

Generally, the conversion of data domain acquired for obtaining information from the data by detecting specific patterns involve a high computational cost. Thus, the importance of a standardized structure proves to be an asset when, at the outset, it presented several goals. With this, we abdicate the necessity of return to the original acquired samples, and go through the processing stage, each time we intend to apply a function of the algorithm. In Figure 2.9 is represented the distinction between the most conventional methods and the technique proposed here enhancing, notably, a considerable reduction in the computational cost [17].

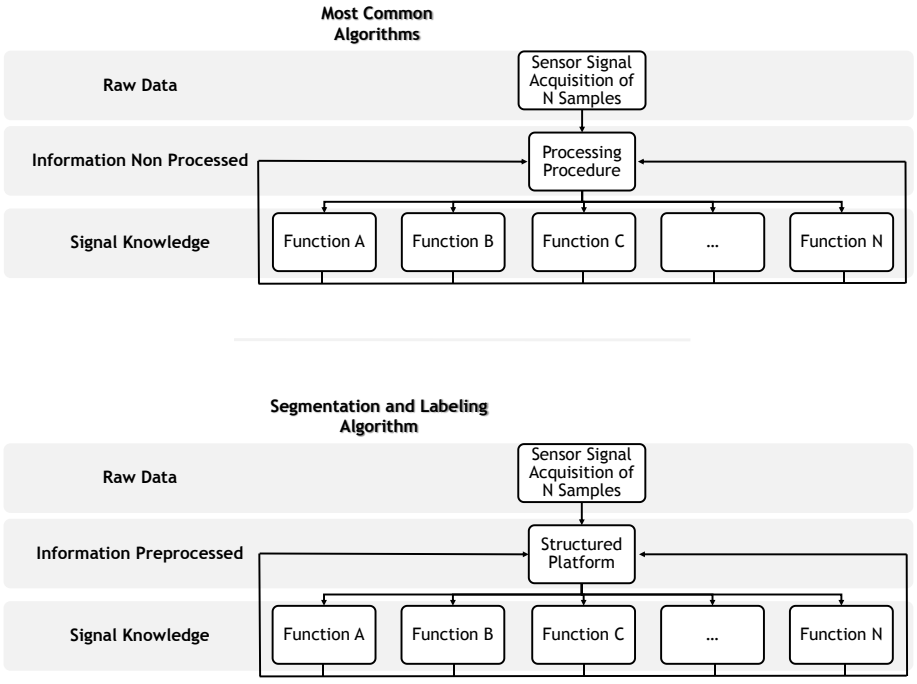


Figure 2.9. Above, structural representation of the classical approach of an algorithm analysis and processing a signal from the raw data to obtain signal knowledge. Below, representation of the proposed algorithm for the same purpose.

As can be observed in Figure 2.9, the algorithm here proposed has a layered structure where the output of one stage is the respective input of the next [11]. Due to the variety of functions provided by the signal processing technique presented here, it can be applied to any type of signal.

The method begins by performing a signal sample acquiring a defined set samples, according to the aforementioned sampling theorem of Nyquist-Shannon [3], [17]. Then it performs a pre-processing of the signal, in the time domain, obtaining a uniform platform with embedded information, from which a homogeneous analysis according to the intention of the same application is possible [17]. The platform will also allow the detection of waveforms from small isolated features that lead to signal characteristics extraction and, consequently, signal recognition.

Now, the interaction of the segmentation process, followed by the raw information labelling process, are involved in getting this intermediate stage and how this sampling proposed technique focuses up to emulate an human observer, to infer the overall performance of the sampled signal [4], [17].

A. Segmentation Process

Considering a specific signal such as sinusoidal, exponential or even noise, it can define some characteristics of it, such as exponential behaviour, maximum and minimum values, and the presence of pulses. These parameters will enable the identification of, for example, waveform types and trends. Accordingly, clearly do not observe isolated samples but, the relationship between them. Thus, although the signal precision is not measurable, it can be taken awareness of their behaviour. Given the need for a more complete sampling process that takes into account the relationships between data segments by the sequence of samples, the idea to take into account is the shape of the signal path between the acquired samples. Naturally, it is not necessary to observe, consecutively, each pair of samples. The key, is to compare the signal path with a linear computational trajectory obtained [4], [17]. This comparison can be obtained by linear interpolation between samples [10]. The first step is signal sampling to acquire samples and, from the obtained result, constitute the linear interpolation process among signal segments.

Let $x(t)$ be an analogue signal acquired from a sensor which is band limited to B Hz. The discrete signal is obtained sampling the analogue signal uniformly with a frequency of f_s, where $f_s \gg 2B$ [4], [17]. In a normal sampling process, because the signal is over-sampled, not all the samples show the same degree of relevance. In some situations, redundancy between adjacent samples is observed. However, if a sample cannot be obtained by linear interpolation of adjacent samples it is identified also as relevant [3], [11], [17]. Thus, the input waveform of the sensor can be described by the amplitude and position of the more relevant samples [18]. The algorithm starts after the acquisition of a defined set of samples. The main idea of this algorithm is to identify the most significant samples [3], [17]. This identification process is described in Figure 2.10. The algorithm starts computing the interpolated value of a sample located between two samples. For instance, a linear interpolation value of sample X_2 is obtained from samples X_1 and X_3. After, two types of errors are computed. First, a substitution error defined as the difference between the actual sample X_2 and the interpolated value is calculated. Second, an error defined as an accumulation error results as the accumulated error introduced by the interpolation process is computed. If the error is less than a predefined threshold, called the interpolation error, the sample is marked as being not relevant. While the interpolation error is below the threshold defined by the interpolation error the algorithm continues. When this happens, the sample X_5 is the next sample to be used in the interpolated process, replacing sample X_3. Now, the interpolation error is computed using the sample X_3. If the difference exceeds the threshold, the sample X_5 is marked as important and all samples interpolation process is restarted from this sample, i.e., the interpolated sample when marked as essential becomes the X_1 of the next iteration process). Visually, the sample that was left over during the linear

interpolation process is now the first sample that constitutes the segmentation process. The segment is defined by the samples used in the interpolation process. The segmentation process is schematically represented in Figure 2.10 [4], [10], [11], [17].

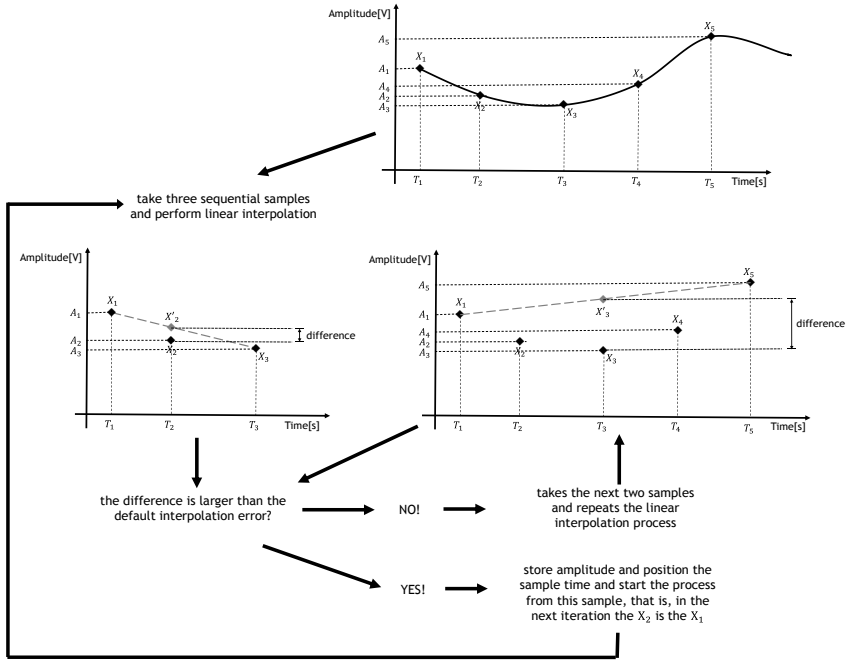


Figure 2.10. Representation of the segmentation process.

That said, in the all set of the acquired samples, if none sample is considered important, the last sample is marked as important and the segment is given as completed [10], [11], [19]. This algorithm returns two vectors of values, one stores the amplitude of each sample marked as relevant, designated as the $\mathrm{marks[n]}$ vector, and the other holds the respective positions of the samples in time as is called the $\mathrm{times[n]}$ vector. The algorithm is extremely simple but nonetheless allows the segmentation of the signal in real time. At the same time it offers us a standardized platform to store and analysis signal bases on segments [4], [10], [11], [17].

B. Labelling Process

The algorithm operates as a decimation process. The segmentation process inspecting all set of samples acquired keeping only those deemed as the most significant, removes the founded redundancy from the defined set of the acquired samples. The labelling process classifies each pair of two samples, that define a segment, with one new parameter [3] - [4], [17]. Therefore, if the signal is over-sampled, the actual path instilled in the segment is bounded by the interpolation error [17], [18]. Whereas the values of samples define the segment based on the interpolation error, the trajectory of the segment can be understood as one of the eight classes shown in Table 2.3 [17].

Table 2.3. Graphic representation and brief description of the eight classes.

Classes	Representation	Description
A	Constant behaviour (strictly linear) Corresponding Hexadecimal ASCII code: 65	
B	Constant behaviour (strictly linear increase)  Corresponding Hexadecimal ASCII code: 66	The segment ends at the prefixed N value and, interloping the left and right samples, the interpolation error it is always less that the predefined threshold.
C	Constant behaviour (strictly linear decrease)  Corresponding Hexadecimal ASCII code: 67	
D	Increasing (+) exponential (+) behaviour  Corresponding Hexadecimal ASCII code: 68	The signal grows rapidly and, after N segments, the length of instilled segments decreases.
E	Decreasing (+) exponential (-) behaviour  Corresponding Hexadecimal ASCII code: 69	The signal quickly decays and, after N segments, length of instilled segments decreases.
F	Decreasing (-) exponential (-) behaviour  Corresponding Hexadecimal ASCII code: 70	The signal grows rapidly and, after N segments, the length of instilled segments increases. Where, the predicted steady state value exceeds a prefixed limit.
G	Increasing (-) exponential (+) behaviour  Corresponding Hexadecimal ASCII code: 71	The signal quickly decays and, after N segments, the length of instilled segments increases. Where, the predicted steady state value represses a prefixed limit.
H	Purely noisy behaviour  Corresponding Hexadecimal ASCII code: 72	The signal is noisy and, after N segments, probably is a sensor failure.

For a better knowledge of the definition of this signal behaviour signal representation, refer to the paper from reference [10]. This classification is amplitude and scale invariant in time, segment by segment, and is sequentially stored in a vector of characters designed as classes[n] vector. To each one of the segments one of the following classes is associated {a, b, c, d, e, f, g, or h}. The structure of vectors: marks[n], classes[n] and times[n], represented by the acronym MCT, allow the description of the signal. Accordingly, these three vectors are the support base of the fundamental structure used by the functions developed and presented later in this work. The MCT vector can be viewed as returned from an intelligent sampling process, where the relationship between samples labelled as essential as enough to analyse the signal. The need for an oversampling frequency is related with the intention of obtaining a uniform behaviour within the segment. Inside each segment, most of the interpolation errors and have a positive or negative sign feature. Thus, given the conditions of over-sampling, if the values accumulation of the interpolation errors deviating from zero, the probability of occurrence the segments a, b, c, or h approaches zero [4], [17]. Therefore, together with the errors obtained by the linear interpolation as the difference between the leftmost and rightmost sample of each segment, is possible to proceed to the classification of signal behaviour. That is, for each pair of consecutive samples is being assigned a label [3], [10], [11], [17]. Therefore, the more likely it is that the segment is classified as one of the remaining four classes: d, e, f or g [4], [17].

Its graphical representation of the sequential classes[n] vector will identify the shape of the signal, i.e. representing the behaviour of the acquired signals [19]. Therefore, the signal can be reconstructed from their MCT vectors. The intelligent transducer which identifies the measured signal type as well as some of its characteristics [10], [17]. In sum, it is an efficient process, which can be calculated via an algorithm based on the operator signal. This can easily be summarized by the following equation (2.18) [10], [11]:

$$diference(i) = \sum_{k=k_1}^{k_2} sign[x'(n) - x(n)] \tag{2.18}$$

Where x(n) represents the signal acquired by the sensor, x′(n) the resulting signal interpolation, k_1 and k_2 are the contents of the sample leftmost and rightmost of segment i, respectively, whose size is given by $k_1 - k_2$ [10], [11]. Therefore, at computational level, the proposed algorithm is considered cost effective. In this mode, it can be applied in smart sensors and executed in real time, providing a standard platform, obtained from a set of three vectors which allow the analysis and signal processing [4], [17]. The Figure 2.11 represents the behaviour of this algorithm as applied to a random signal.

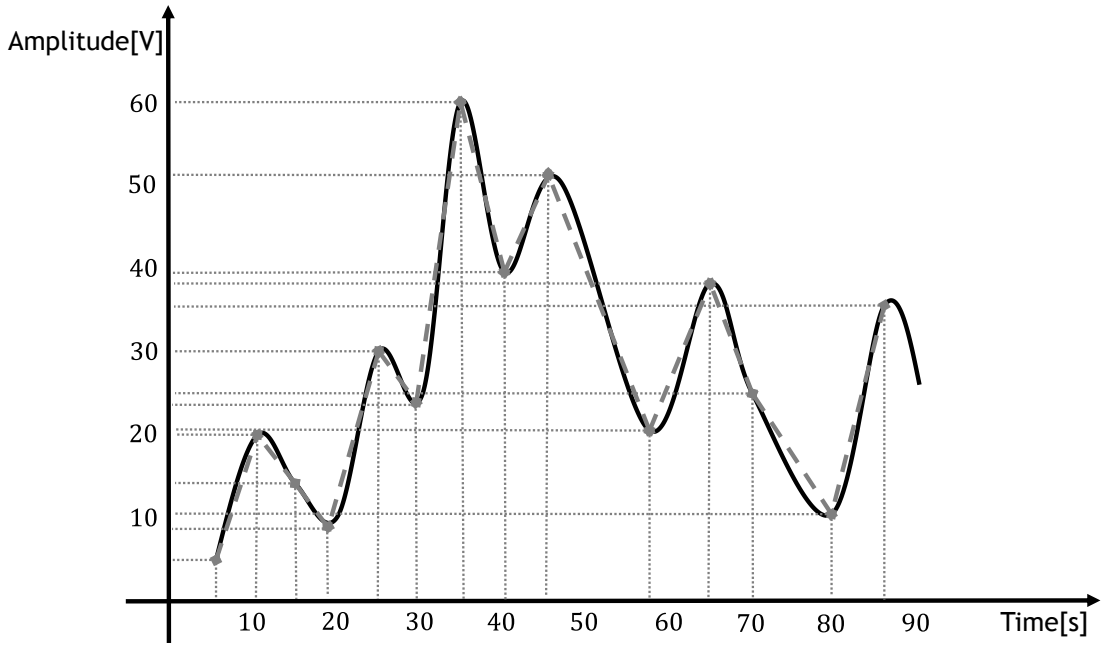


Figure 2.11. Representation of the S&L algorithm behaviour when applied to an aleatory signal.

Respectively, in the Table 2.4, it are presented the output return variables.

Table 2.4. Output return variables of the S&L algorithm application on an aleatory signal.

Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
marks[ ]	5	20	14	8	30	24	60	39	51	21	38	25	11	35
class[ ]	Z	G	E	F	G	E	G	E	G	E	G	E	F	D
times[ ]	6	10	15	19	24	29	35	40	45	57	64	70	79	86

At the same time, as a result, the smart sensors can define the type of acquired signal, as well as, some of their characteristics, depending on the applied function; prove it to be more interesting. Some of these features may be, for example, pattern detection, meaning, variance, standard deviation, trend, signal amplitude, presence of noise, etc. Finally, one has to obtain a layered structure in which an output of a stage is the input of the next. The implementation of this algorithms makes possible the implementation of a large number of different functions [4], [18], [19]. The flowchart in Figure 2.12 represents schematically the implementation of segmentation and labelling process by the, hereinafter called, MCT function.

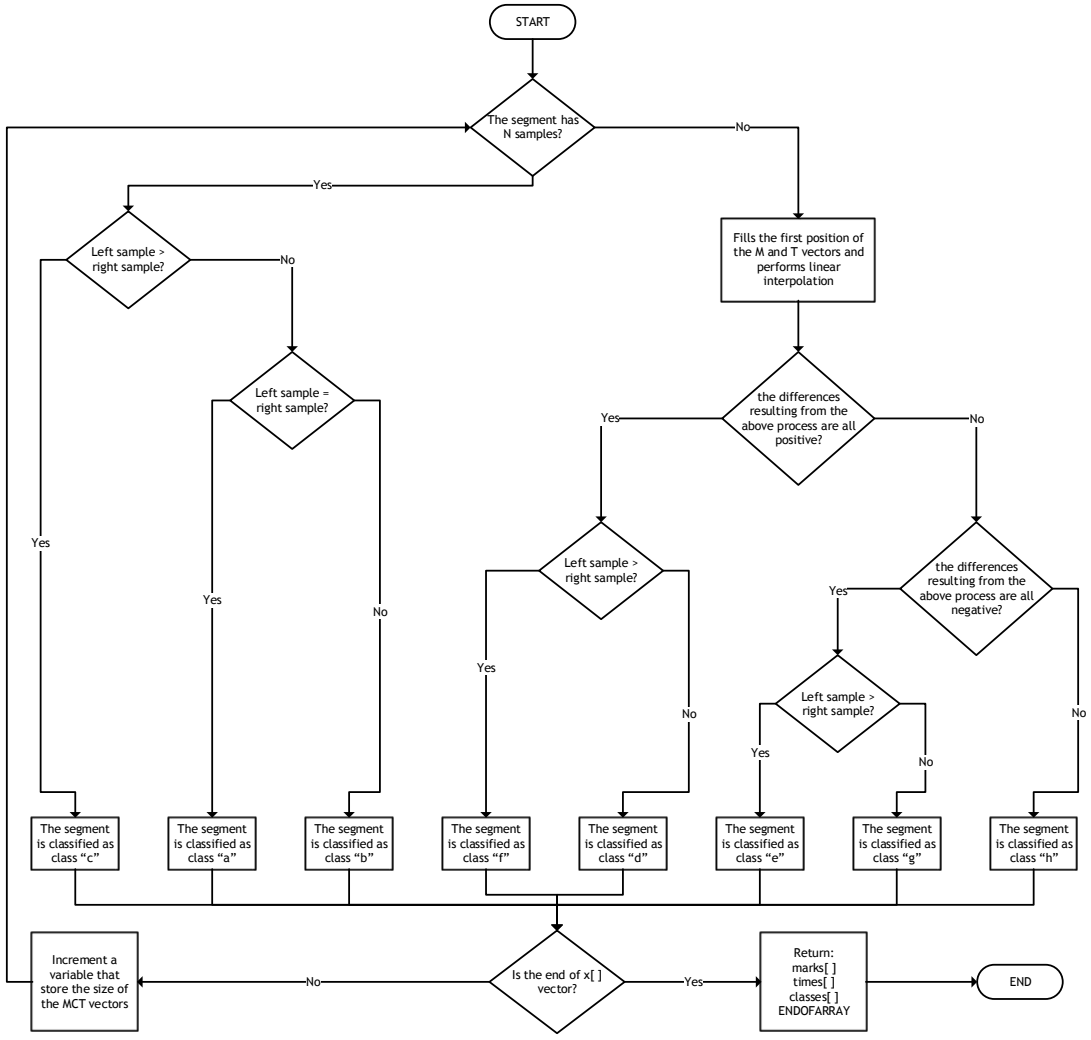


Figure 2.12. Flowchart of segmentation and labelling processes.

In short, the above inputs and outputs are reported in Table 2.5.

Table 2.5. Listing inputs and outputs of the MCT function.

INPUTS	
$\boldsymbol{int\ x[\,]}$	Vector that storage the original sampled signal.
$\boldsymbol{int\ X\_length}$	Defined value to dimension the $x[\,]$ vector.
$\boldsymbol{int\ erro}$	Interpolation error value considered for comparison during linear interpolation process.
$\boldsymbol{int\ delta}$	Increment of the error at each iteration.
$\boldsymbol{int\ cycles}$	Indicates the number of samples that will be read each time.
$\boldsymbol{int\ segmentLength}$	Define the number of samples that compose the size of the segment.

OUTPUTS	
$int\ marks[\,]$	Vector that stores the amplitudes of the labelled samples. Expressed in ADC levels.
$char\ classes[\,]$	Vector that stores the classes of the labelled segments. Expresses by the ASCII code.
$int\ times[\,]$	Vector that stores the time position of the labelled samples. Expressed in time resolution.
$int\ ENDOFARRAY$	Keep the size of the MCT vectors and is given by their number of indexes. Expressed samples.

In the next subsection will be demonstrated a comparison between the simulation of this algorithm using MATLAB and real performance on an ARM microcontroller from the board ET-ARM STAMP STM32, power supplied by the ET-STM32F103 module. The photography in Figure 2.13 reveals the mentioned assembly scheme.

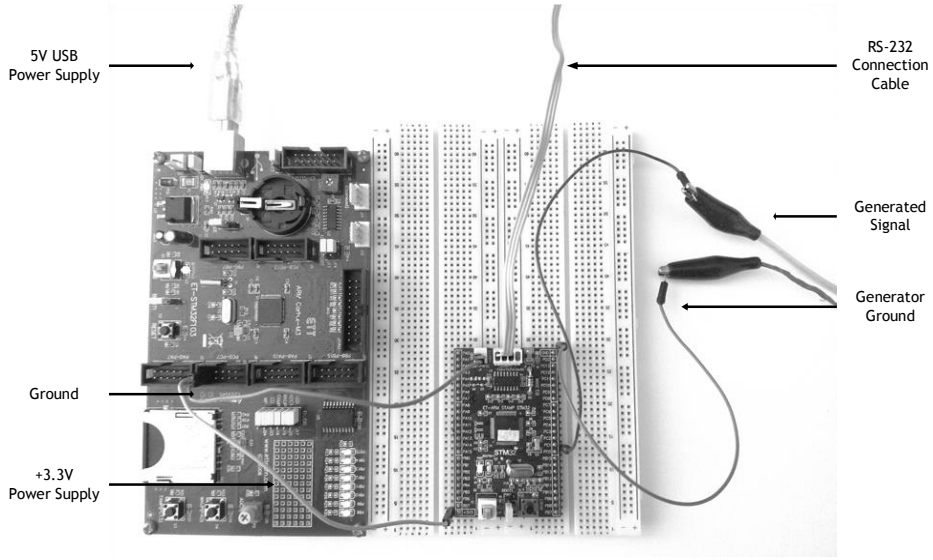


Figure 2.13. Representation of the assembly scheme.

2.3.1. Simulation Environment versus ARM Microcontroller Implementation

The Electrocardiogram (ECG) is a record of changes in electrical potential generated by the heart's electrical activity. Therefore, inspired in this knowledge, the S&L algorithm was tested in two different environments and were recorded their respective results and comments. Initially, the physiological signal was simulated by MATLAB. Note that MATLAB works with floating representation, which makes the generated values to describe the ECG

signal of float type. In return, the microcontroller represents the data using integer types to reduce computational load. Thus, before being applied the S&L algorithm (by MATLAB or by the microcontroller) a conversion of the values that represent the ECG signal from floating to integer values was performed. The algorithm was applied and the results of the MATLAB simulation are plotted in Figure 2.14.

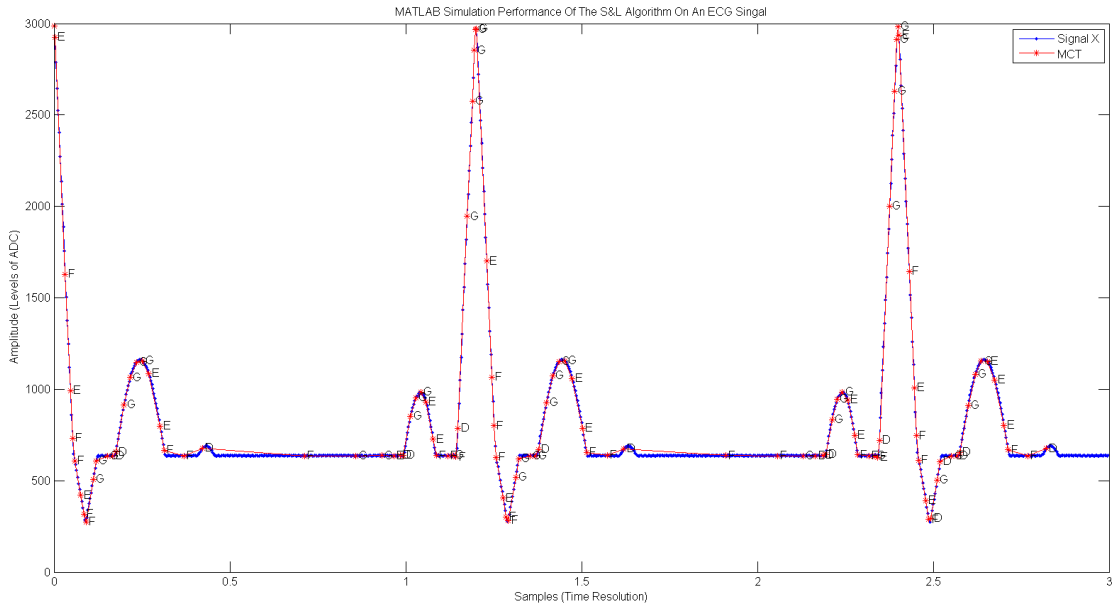


Figure 2.14. Graphical representation of the MATLAB simulation performance of the S&L algorithm on an ECG Signal.

The vector of data representing the ECG $x[n]$ vector obtained in the simulation was transferred into the microcontroller's memory in other to ensure that the both tests were performed with the same set of samples allowing results evaluation. The results of this segmentation process it is also represented graphically in Figure 2.15.

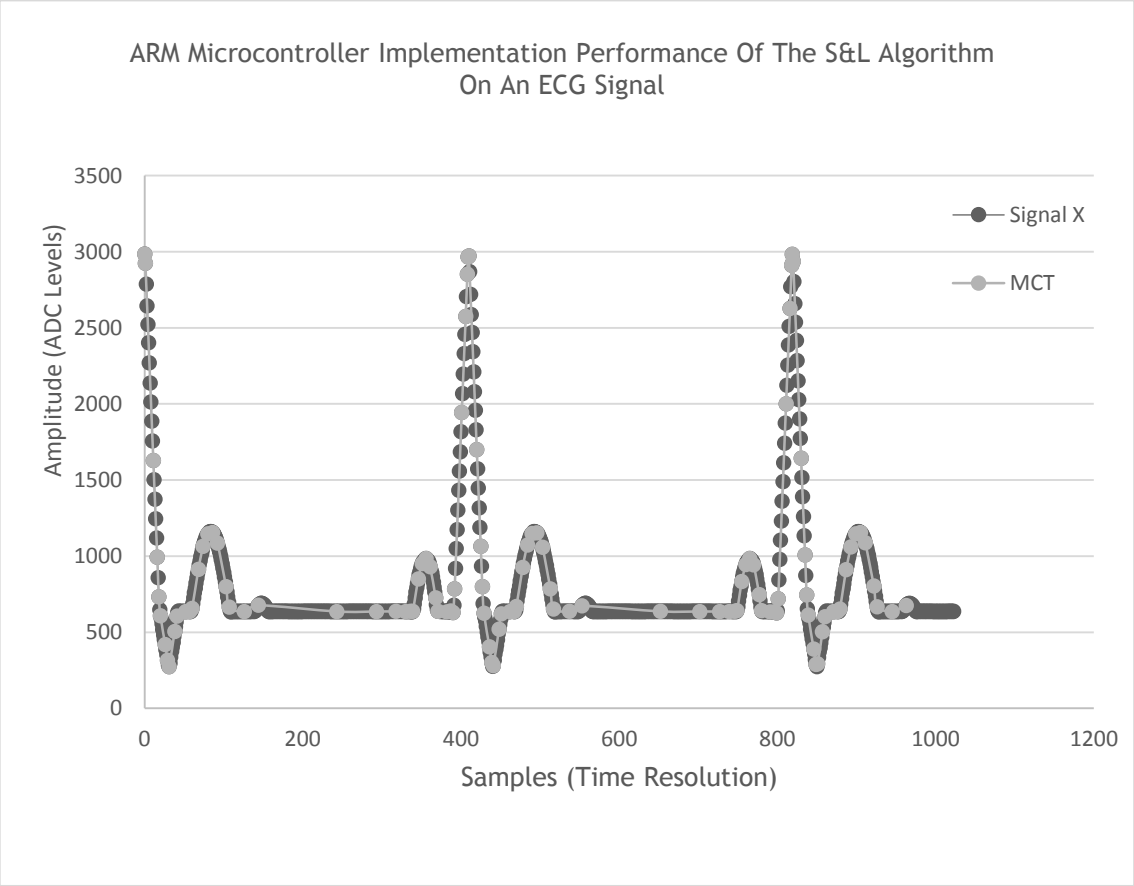


Figure 2.15. Graphical representation of the ARM microcontroller implementation performance of the S&L algorithm on an ECG signal.

Finally, the input and output results from both steps were recorded and compered in Table 2.6.

Table 2.6. Input and Output of the MCT function.

	MATLAB SIMULATION	**ARM MICROCONTROLER IMPLEMENTATION**
$int\ Number\_of\_Acquired\_Samples$	1024	1024
$int\ Number\_of\_Samples\_Labeled$	107	107
$int\ Number\_of\_Maxima$	12	12
$int\ Number\_of\_Minima$	13	13
$int\ X\_length$	1024	1024
$int\ cycles$	1	1
$int\ delta$	0	0
$int\ segment\_Length$	1024	1024
$int\ Interpolation\_Error$	33	33

From the evaluation of data represented in Table 2.6 it is concluded that the segmentation process made by the microcontroller is identical to the one returned by Matlab. For the same set of samples, and given the same function input parameters, both implementations return the same output. This means that the microcontroller response is satisfactory, insofar as it is able to mimic the ideal response obtained in simulation. Looking into Figure 2.14 and Figure 2.15, comparing it is visible that the labelling defined to each segment is exactly equal in both. With this, it can be concluded that besides the response being the same, the MCT vectors allow the graphic reconstruction.

Chapter

3

Collectively with the necessity of establishing a standardized implantation of interoperability between intelligent sensors, also came the need to develop a simple and functional software for easy user interaction and hardware accessible and effective in application execution. Thus, the software presented here are interactive development tools and, at the same time, the hardware devices are robust and operational for implementation of the computer projects.

3. Hardware and Software Development Tools

The development tools, also known as programming tools, are designed to allow users to create, debug and maintain code projects. Generally, the beginners start with small and simple programs, the *HelloWorld* for example is a well known case, and then they will progressively improve, not only their interaction capabilities with the software development tool, but also their expertise in hardware and firmware. As advanced users, they search for emulated systems with low-cost in-circuit debugger/programmer. These tools support embedded systems designers, giving them advanced features and allowing the achievement of more benefits from the systems developed by him. In the next two sections, the features and characteristics of the applied software tools and the hardware devices are described.

3.1. Interface, Environments, Development and Production Tools

The most basic tools, for development and production, are a source code editor and a compiler, which are used ubiquitously and unceasingly in software engineering. However, there are other tools that are more or less, dependent from the programing language, the development methodology, the individual engineer. Tools may be an isolated application software that runs separately. Otherwise, they may be parts of a platform that aggregate different types of applications, called an Integrated Development Environment (IDE). Conventionally, there are two levels of programming. At the lowest level, the executable code consists generally in an assembly code, correspondent to specific instructions to an individual processor. On the other hand, at the highest level, is easier and more efficient to code in C/C++, when compared with the assembly code, this one seems to be more close to our natural language. However, this last one is translated, using compilation or interpretation, or a combination of the two, into assembly that is later translated to machine code. Considering this, in the next five subsections, the environmental tools applied for the development of the present project will be presented.

3.1.1. Keil for ARM Devices

An offline tools allows the interface between the user and the board. An example of this, especially for ARM Devices, is the Keil compiler. In addition to being made for high-performance development boards, it also comes with support features making it ideal as a stand-alone controller or development. With no drivers to install, it just requires a software installation to allow the user enjoy of a C/C++ programming environment. It is mainly suggested for user to do the download of the setup file, freely available on the web page from reference [20], and execute him to perform the respective installation of this software for his Operative System (OS) available. By this way, a personal developer work area will be

created in his computer [20]. To learn more about this offline tool for programmers, check how to getting started with the guide in Annex A.

3.1.2. Documentation with Doxygen

The professional development of systems is fully associated with the importance of the source code of the document. Of course, if the documentation is not done efficiently the involved code files, will be going to create the possibility of difficulty in who will make future maintenance, revisions, improvements and/or changes to the source code. As this is a good programming practice, the best way to execute it effectively and efficiently is to apply code documentation techniques. First, it is needed to have a sense of what needs to be documented. All source code files, regardless of their language, should be corrected and documented respectively. Second, obtaining knowledge of how this can be done to get the respective documentation. The way that the programmer should document, depends largely on the type of file that aims to build. Therefore, to facilitate this task it has been developed Doxygen [21]. This tool supports not only this task but also all documentation generation of various programming languages. The documentation generated by this software may take various forms:

- Online browser, in HTML;
- Offline reference manual, in LaTeX;
- Several other types of output, such as, in hyperlinked PDF.

To learn more about how to process the source code documentation, and how it is possible to generate it, check the getting started guide in Annex B.

3.1.3. Version Control with Subversion

The implementation of large projects often requires the safeguarding of data that allows multiple users to monitor the development and evolution of it. To match this interest, the Apache Software Foundation has created an open-source version control tool universally recognized as Subversion. This is a file repository like a normal folder except that it allows the visual history and file restoration after its amendment. At any time is possible to restore the older versions of files. Thus, it allows multiple users to follow step by step upgrade each source file, allowing each one to check and introduce corrections at any time, once they all are connected to the same project via a server. For sure only the list of versions is not enough, and the users need to track each one of the changes introduced by him or other developer into the project. To solve and manage possible conflicts that may occur among users, it is possible for each version to be imported into the repository to associate a message which shall include a list of all the changes and revisions implemented in the code. Later, it

can be observed the differences in source files, and even review and apply unified diff files. For the current project, TortoiseSVN was selected to perform version control in parallel with the VisualSVN Server to provide a server for all users. To learn more about this version controller tools, and how it is possible to share a project between several users, check the getting started guide in Annex C.

3.1.4. ArbExpress Application

The most common generators that, beside all their features, provide a set of standard signal functions: sine, square, ramp, pulse and a set of arbitrary signals: DC, noise, exponential rise, exponential decay, Gaussian, Lorentz, Haversine and $\sin(x)/x$. One can easily think in some other signals that are not included in the previous list, such as: the ECG, signals with impulses, etc. To avoid these cases, the Tektronix developed a software that, despite all that has been mentioned, allows the simulation of real-world complex signals. Designers aimed beyond the simulation signals and conceived the ArbExpress in order to allow the transfer thereof to generators as an arbitrary waveform function. Among its benefits are highlighted: ease replication signs replication from the *.csv* file import data; save time by adapting several standard and advanced waveforms to get what is desired to simulate; and many others as can be consulted in reference [22]. To learn more about this arbitrary function generator, and how it is possible to simulate and transfer data from a *.csv* file to the generator, check the getting started guide in Annex D.

3.2. Hardware Features and Specifications

The hardware used to implement this project was carefully selected. Although the algorithm that the project intends to develop has low computational requisites, it is important that the applied hardware reply satisfactorily project's needs, such as, data range and scale, bandwidth, memory size and computational speed. Considering this, it will be possible to increase the potential advantages of the algorithm developed to execute the project. In the next three subsections, the hardware applied for the execution of the present project in study will be presented.

3.2.1. STM Microcontrollers With Keil

The ET-STM32F103RBT6 it integrates a 32-bit ARM Cortex-M3 that is quite perfect and suitable for learning, development and to apply in many projects. All the detailed information required by the user to help him to understand how to use the resources available in the hardware system is available in references [23]–[25]. This board was designed for medium-density performance and operates at CPU frequencies of up to 72 MHz, running up to 90 MIPS (1.25DMIP/MHz) [26].

The system is packaged in a small DIP with 46 general purpose I/O pins (GPIO), 0.1-inch pitch form-factor making it convenient for prototyping with trough hole PCBs, strip board and breadboard, and includes also a built-in drag-and-drop USB FLASH programmer. Included is also a JTAG connector, LCD connector, 8 test LEDs plus on power and boot LED. With 5V USB power supply and a real time clock with battery backup, it also presents a frequency up to 128 KB of flash memory and an up to 20 KB of data memory (SRAM), running with a crystal at 8000 MHz at a rate of 72 MHz and a SD Card Socket. The peripheral complement includes: CAN channel, 3 SPI interfaces, 2 I2C-bus interfaces, 2 UART (for RS232 connection and program download), 2 16-bit Timers, 16 12-bit ADC [27].

Considering all this, it is easy to understand that the design purpose of the design of this board is to support users that need to learn, modify and develop. It also allows agile development with fine-tuning of both hardware and software on-the-fly at each prototyping stage. Investment in application shields and protected as most shields can be used across various projects leveraging the scalability and diversity [28].

The Figure 3.1 shows an exemplar of this board and the RS-232 connection cable, include on the package, for communication between the board and the computer [27].

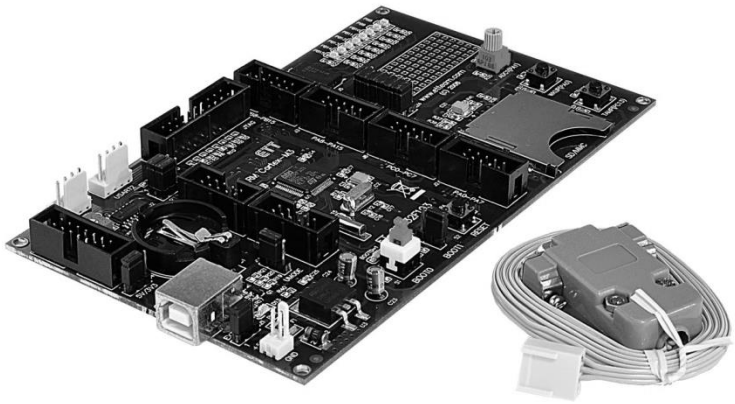


Figure 3.1. Module ETT-STM32F103 and RS-232 connection cable.

To have an extra load on memory, as another example of a board for algorithmic implementation with Keil, is the ET-ARM STAMP STM32. Compared with the previous module, this board present large memory size in a small module, suiting a wide range of embedded applications and controllers [29]. An exemplar of this module is shown on Figure 3.2.

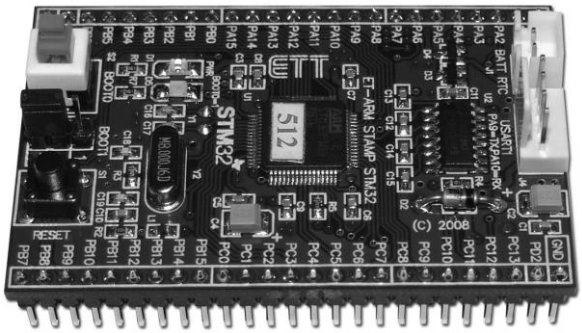


Figure 3.2. Module ET-ARM STAMP STM32.

Presenting almost the same characteristics as the module ETT-STM32F103, despite some other things, the ET-ARM STAMP STM32 t is particularly innovative as regards the storage capacity [29]. Making a comparison between features provided by both Table 3.1 [30].

Table 3.1. Comparison of the features specifications between two different STM modules.

Specification Peripheral	ETT-STM32F103	ET-ARM STAMP STM32F103
MCU	STM32F103RBT6	STM32F103RET6
FLASH	128K	512K
RAM	20K	64K
SPI	2	3 (I^2S x 2)
I^2C	2	2
USART	3	5
USB	1	1
CAN	1	1
SDIO	-	1
ADC 12 Bit	16	16
DAC 12 Bit	-	2

In sum, the powerful ARM Cortex M3 processor is ideal for real-time high-speed applications. With this, develops a high-performance optimized for support of writing source code in high-level programming languages. Furthermore, both these modules will be applied to the implementation of an experimental setup and its performance will be respectively evaluated.

Chapter 4

To approximate the proposed algorithm, presented above, the emulation of a human observer, was developed a set of functions that, from a preprocessed data platform, are able to retrieve information about the characteristics and behavior of any type of signal. Taking into account the interaction and application usability, it is presented all API specifications of the respective functions and. Moreover, it will be performed an experimental setup were, comprised by operating description of the function, the input and output parameters are defined. Further, for each function, there will be presented some of the results obtained, as well as, some analysis and commentary to them.

4. Application Programming Interface

An API is a set of routines and standards established by the software to be used by applications. In general, the API consists of a series of programmable functions that allow the user to access the traditional software features.

Nowadays, the use of API has been widespread with plugins, i.e., software accessories that complement the functionality of a program. The authors of the main program provide a specific API interface that plugins' authors should follow to create, according to their goals of interest, extend and improve the functionality of a program.

Naturally, when thinking about implementing an API that has to take into account the application usability term. This concept is intended to set the ease with which people can use a tool in order to accomplish a specific task. The importance of this concept refers not only to the reliability of the information obtained by the API but also to the methods of measurement and study of the principles behind its efficiency.

In Human-computer interaction and in computer science, usability usually refers to the simplicity and ease with which a user is able to interact with a computer program or a website. The term is also used in the context of products like electronics, in areas of communication and transfer of knowledge products, such as manuals, documents and online help. It can also refer to the efficiency thereof.

4.1. User API Specifications

In accordance with the definition of API, the project is currently implemented provides six different functions. Thus, any user can potentially create their one plugins in accordance with the objectives of application that he intends to produce. Modifying Figure 2.9 now may have a new perspective, in Figure 4.1, of the structural representation of the proposed algorithm for analysis and processing a signal from the raw data to obtain signal knowledge.

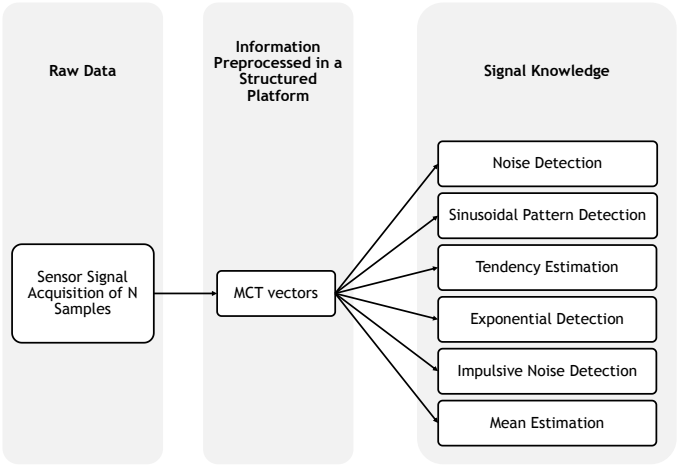


Figure 4.1. Structural representation of the proposed algorithm for analysis and processing a signal from the raw data to obtain signal knowledge.

To achieve this effect, the ideal is that the user should be able to enable/disable this same set of functions by simply using small command lines programmed in a practical and flexible language. Thus, to allow the users to build their own applications, it is only necessary to give the input parameters for, at the function call, to access the output parameters in run time.

In the next section are described the workings of the six functions provided by the algorithm proposed for signal analysis and processing, as well as all input and output parameters. It will be noted also, the range of signals that this algorithm is applicable.

4.2. Functions API Specifications

As aforementioned, the proposed algorithm it based on a proposal for a new standard that will be included in the IEEE 1451 family. As showed before, the idea is that the application of this algorithm should be capable of recognizing any signal preforming not only pattern detection, but also, returning parameters that characterize the signal. Observing this aspiration, the draft presents a layer with six functions.

In the next six subsections, the implementation of the functions will be briefly described and. A set of several testes will be presented with respective analysis review and comments. Once again, it should be reminded that all this functions are applied to the MCT vectors and not to the 1024 samples acquired from the original signal. Note that the ARM microcontroller selected to run all this testes was installed in the evaluation board ET-ARM STAMP STM32, the power was supplied by the board ET-STM32F103. At the same time, all signals were generated by the dual channel arbitrary/function generator AFG3022C, from Tektronix, and, at the same time, monitored by the scope DSO-X 2012A, from Agilent Technologies, as shown on Figure 4.2.

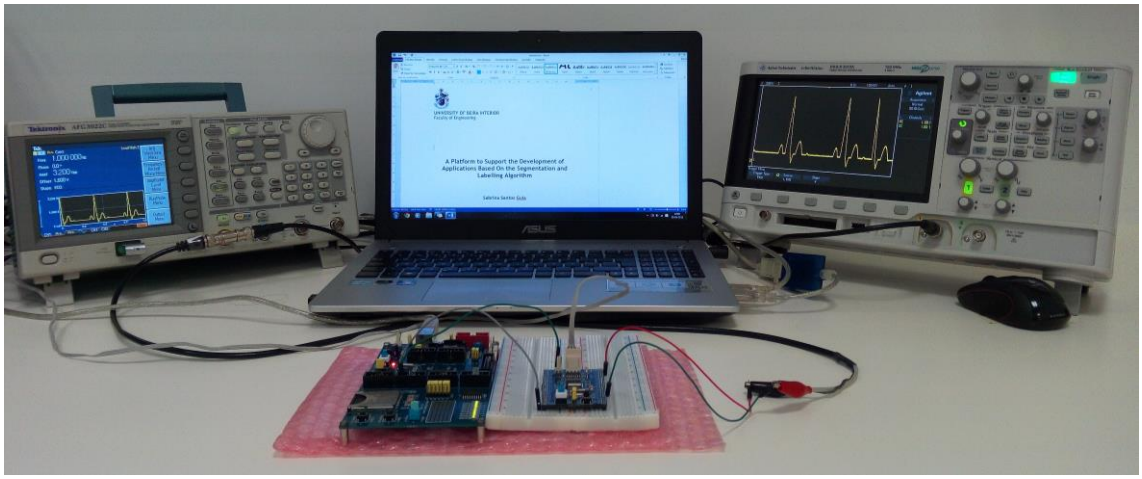


Figure 4.2. Electronic equipment applied to perform the experimental setup.

4.2.1. Noise Detection

Noise is defined as a degradation of the original signal, mainly, by external factors. This component of a real world signal does not carry useful information. Stepping through the set of N pre-processed signal samples, the presence of noise is revealed by computing the distance between the maximum amplitude, sequence of grouped classes "ge" and "gf") and the minimum amplitude (sequence of grouped classes "fd" and "fg"). In a simplified manner, the occurrence of a maximum is always followed by the occurrence of a minimum, and vice versa. Calculating the time difference between these two singularities, in sequence, if it is below a predefined value, it is activated a flag that identifies the acquired signal as being noisy. The flowchart in Figure 4.3 represents schematically the implementation of the Noise Detection function.

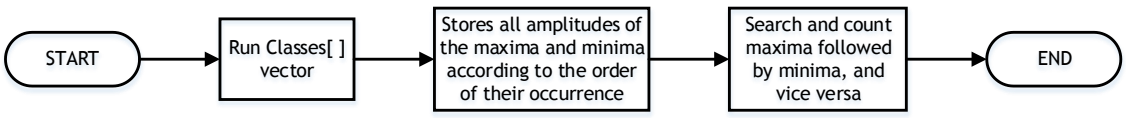


Figure 4.3. Flowchart of Noise Detection function.

In short, the above inputs and outputs are reported in the Table 4.1.

Table 4.1. Listing inputs and outputs of the Noise Detection function.

INPUTS	
$int\ SamplingPeriod$	Integer that defines the frequency of acquisition. Time resolution is expressed in microseconds.
$int\ AmountNoisy$	Integer that defines the necessary amount of consecutive detection of

	noisy pattern.
OUTPUTS	
int Noisy	Boolean variable that is defined as $+1$ if the signal is Noisy, -1 if the signal does not have noise, and 0 if the function cannot defined the signal shape.

Next, the tests conducted with this function are described. The most relevant results are presented. Afterwards there will be done some Analysis and comments are provided at the end of this section.

A. Experimental Setup

To test this function two well-known signs were used: purely noisy; and DC. Obtained from the signal generator, these signals were tested applying the following sampling frequencies: 100Hz, 500Hz, 1kHz, 5kHz, 10kHz, 50kHz and 100kHz. For each sampling frequency, are also applied the following interpolation errors: 409, 90, 45 and 10 ADC levels. A screenshot image of each generated signal is shown in Figure 4.4, respectively.

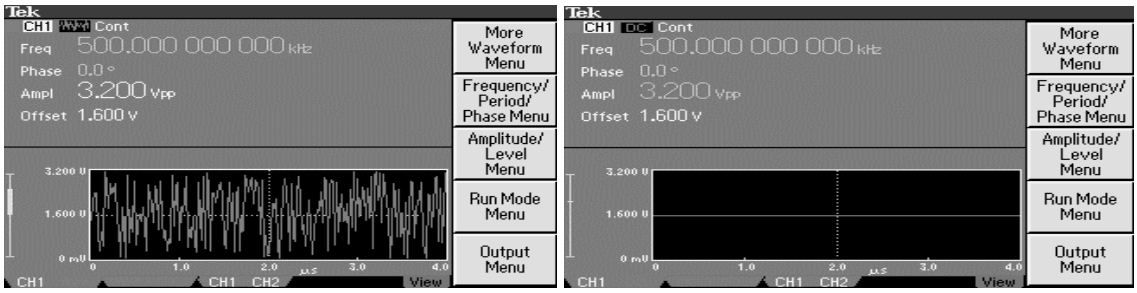


Figure 4.4. On the right, the generated noise signal; On the left, the generated DC signal.

For all the cases, the SamplingPeriod it was set up as 10ms and the AmountNoisy was set up as 8 levels of the ADC full scale. Next, some of the results are listed in Table 4.2 and Table 4.3.

Table 4.2. Results of the Noise Detection function applied to a pure noisy signal.

Test Number	Number of Acquired Samples	Number of Samples Labelled	Interpolation Error	Sampling Frequency	Boolean Noisy
1		81	409	100Hz	
2		729	90		
3		854	45		
4		996	10		
(…)		(…)			

13	1024	85	409	5kHz	1
14		694	90		
15		889	45		
16		978	10		
(…)		(…)			
25		112	409	100kHz	
26		718	90		
27		865	45		
28		977	10		

Table 4.3. Results of the Noise Detection function applied to a DC signal.

Test Number	Number of Acquired Samples	Number of Samples Labelled	Interpolation Error	Sampling Frequency	Boolean Noisy
1	1024	1	409	100Hz	-1
2			90		
3			45		
4		37	10		1
(…)			(…)		
13		1	409	5kHz	-1
14			90		
15			45		
16		41	10		1
(…)			(…)		
25		1	409	100kHz	-1
26			90		
27			45		
28		34	10		1

B. Results Analysis

From a detailed analysis of the results presented in the tables above, it can easily be concluded that, overall, the performance of this function is quite satisfactory. However, there are some cases where the results returned from the function reveals some shortcomings. Analysing the results in Table 4.2. Apparently, the response function response is perfect in all cases, but, whenever, the interpolation error is 409 ADC levels, the signal resulting from the reconstruction based on the MCT is considerably different from the original signal. Both signals are plotted in graph from Figure 4.5.

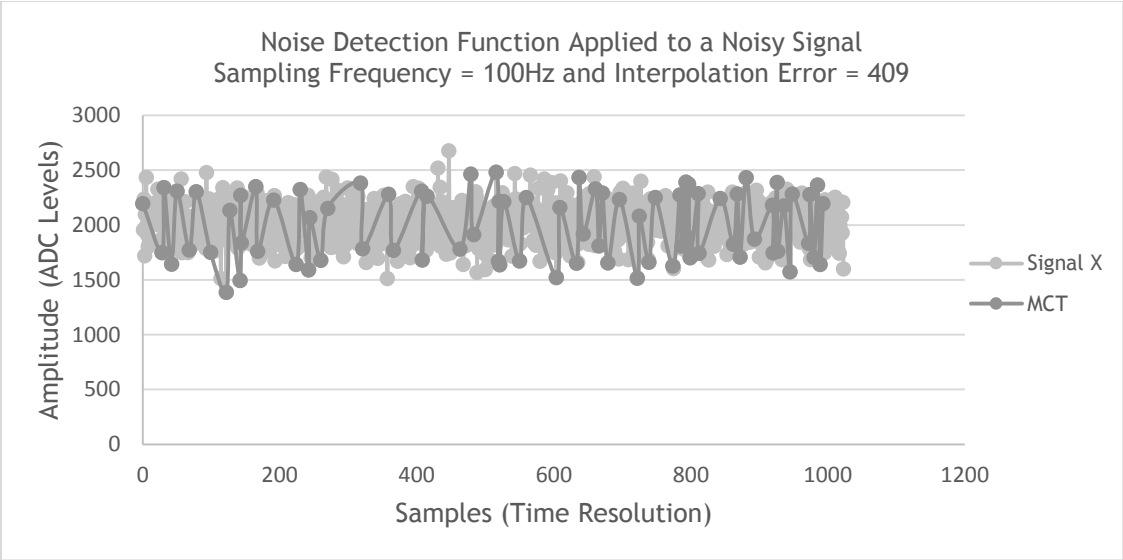


Figure 4.5. Representative graph of the response from the Noise Detection Function when applied to a pure noisy signal with a sampling frequency equals to 100Hz and an Interpolation error equals to 409 ADC levels.

The information stored in the MCT vectors does not accurately describe the original signal. However returns a satisfactory answer. From analysis of the results in Table 4.3. Unlike the previous case, attention is now redirected to the cases in which the interpolation error assumes the value of 10 ADC levels. In this situation, and because the interpolation error is so small, the representation based on the values from the MCT vectors describes very closely the original signal, highlighting the sequences of maximum and minimum present in the signal and, by consequence, considering it as noisy. This case is represented in Figure 4.6.

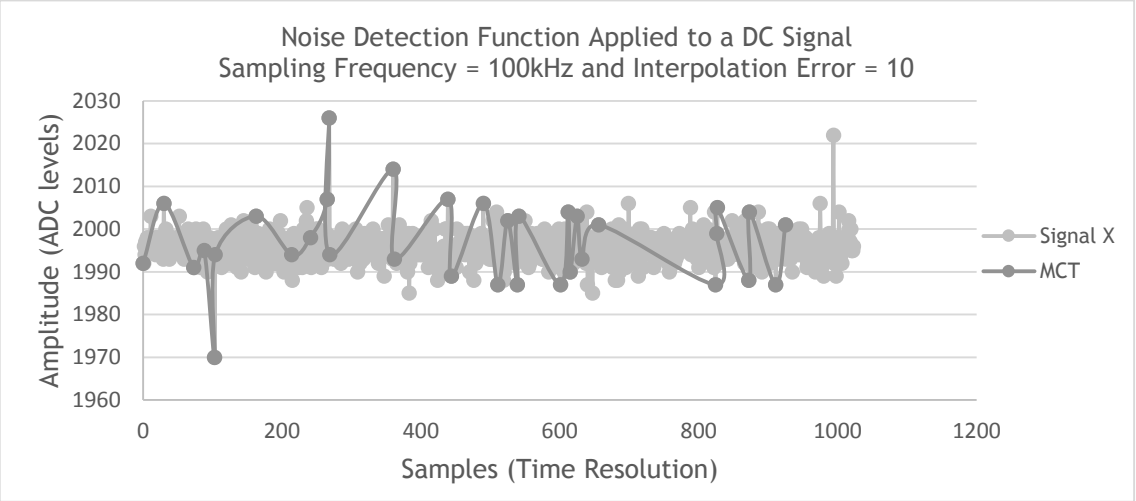


Figure 4.6. Representative graph of the response from the Noise Detection Function when applied to a DC signal with a sampling frequency equals to 100kHz and an Interpolation error equals to 10 ADC levels.

In this case, just like the previous case, the representation of the signal based from data stored in the MCT vectors does not accurately describes the original signal, but still returns a satisfactory answer.

4.2.2. Sinusoidal Pattern Detection

A sinusoidal signal, also known as sine, is any signal whose waveform has an equation which a variable is directly proportional to the sine of another, performing simple harmonic motion. Thus, this movement can be described by the equation (4.1):

$$y = A\,sin(kx - \omega t - \psi) + D \tag{4.1}$$

Where A represents the amplitude of the signal, k defines the wave number, ω the phase shift, ψ the angular frequency and D the vertical offset. By correlation, the wave number k is related to the angular frequency φ, as shown at by the equation (4.2):

$$k = \frac{\omega}{c} = \frac{2\pi f}{c} = \frac{2\pi}{\lambda} \tag{4.2}$$

Where c represents the wave propagation velocity, λ defines the wave-length and f the signal frequency that also can be obtained by the equation (4.3):

$$f = \frac{1}{\mathrm{T}} \tag{4.3}$$

Where T represents the period of the signal.

As already mentioned, a maximum occurs in the group of classes "ge" or "gf" and a minimum occurs when the collation classes "fd" or "fd". Visually, the sinusoidal pattern is composed by the sequence of the four segment classes "gefd", including repetitions of each one of them. Therefore, the period of the signal can be easily obtained by calculating the difference between singularities of the same type, that is, the difference between maxima, or minima. At the same time, the number of periods is given by the number of maxima presents on the signal less one, or minima, depending which appears first in the signal. Additionally, the difference between the peak-to-peak amplitude determine whether the oscillations are stable (describing the signal as being stable). If they increase then the signal is classified as undamped. If the difference value decreases the signal is defined as damped. The flowchart in Figure 4.7 represents how the *Sinusoidal Pattern Detection* function is implemented.

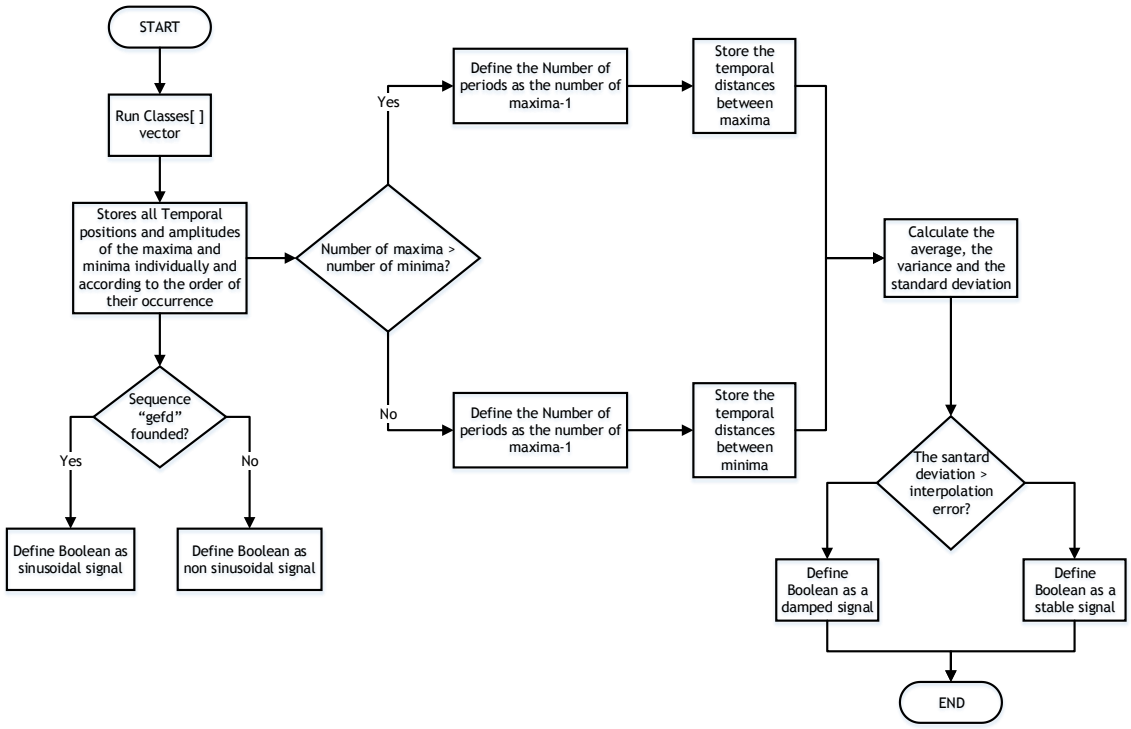


Figure 4.7. Flowchart of Sinusoidal Pattern Detection function.

In short, the above inputs and outputs are reported in the Table 4.4.

Table 4.4. Listing inputs and outputs of the Sinusoidal Pattern Detection function.

INPUTS	
int InterpolationError	Despite his function as a value to establish comparisons during the linear interpolation process, this function also establishes comparisons with the standard deviation value for signal stability study.
OUTPUTS	
int periods_number	Computed value for the number of periods.
float periods_time	Computed value for the time occurrence of periods. Expressed in seconds.
float average	Calculated mean ($\bar{x}$) value of the time occurrence of periods.
float variance	Calculated variance (δ^2) value of the time occurrence of periods.
float standardDeviation	Calculated standard deviation (δ) value of the time occurrence of periods.
float value_amplax	Keep the value of the maximum amplitude value of the periods. Expressed in ADC levels.
float mean_amplax	Storage the mean value of the maximum amplitude values of the periods. Expressed in ADC levels.
float value_amplin	Keep the value of the minimum amplitude value of the periods. Expressed in ADC levels.
float mean_amplin	Storage the mean value of the minimum amplitude values of the periods. Expressed in ADC levels.
float value_ampic	Keep the value of the peak-to-peak amplitude value of the periods. Expressed ADC levels.
flaot mean_ampic	Storage the mean value of the peak-to-peak amplitude values of the periods.
int SinusoidalDetection	Boolean variable that is defined as $+1$ if it is a sinusoidal signal, -1 if is not a sinusoidal signal and 0 if the function cannot defined the signal shape.
int DampedOscillations	Boolean variable that is defined as $+1$ if the is damped, -1 if the signal is stable and 0 if the function cannot defined the signal shape.

A. Experimental Setup

To test the *Sinusoidal Pattern Detection* function two standard signals were used: sinusoidal wave; and ramp shape. The experimental setup for this function was:

- Signals generated at 10Hz of signal frequency (f), the function was tested at 100Hz and 1kHz of sampling frequency (f_s);
- Signals generated at 50Hz of signal frequency (f), the function was tested at 500Hz and 5kHz of sampling frequency (f_s);

- Signals generated at 100Hz of signal frequency (f), the function was tested at 1kHz and 10kHz of sampling frequency (f_s);
- Signals generated at 500Hz of signal frequency (f), the function was tested at 5kHz and 50kHz of sampling frequency (f_s);
- Signals generated at 1kHz of signal frequency (f), the function was tested at 10kHz and 50kHz of sampling frequency (f_s);
- Signals generated at 5kHz of signal frequency (f), the function was tested at 50kHz of sampling frequency (f_s).

Additional tests were also performed with the addition 1% and 5% of white noise to the signal amplitude. For each sampling frequency, it was applied the following interpolation errors (IE): 409, 90, 45 and 10 ADC levels. Some of the results are presented in Table 4.5 and in Table 4.6.

Table 4.5. Results of the Sinusoidal Pattern Detection function applied to a pure sine signal.

Test	f (Hz)	f_s (Hz)	Noise	IE	Periods Number	Periods Time (s)	$\bar{x}$ (s)	δ (s)	Sine Boolean	Stable Boolean
						(…)				
49	100	1000	0%	409	101	0.01	1.00E-02	1.40E-04	1	-1
50				90	101	0.01	1.00E-02	1.71E-04	1	-1
51				45	101	0.01	1.00E-02	1.40E-04	1	-1
52				10	101	0.01	1.00E-02	1.40E-04	1	-1
53			1%	409	100	0.01	1.00E-02	3.17E-04	1	-1
54				90	101	0.01	1.00E-02	1.40E-04	1	-1
55				45	101	0.01	1.00E-02	2.99E-04	1	-1
56				10	101	0.01	1.00E-02	2.22E-04	1	-1
57			5%	409	100	0.01	1.00E-02	6.19E-04	1	-1
58				90	101	0.01	1.00E-02	4.35E-04	1	-1
59				45	101	0.01	1.00E-02	3.46E-04	1	-1
60				10	101	0.01	1.00E-02	3.86E-04	1	-1
61		10000	0%	409	9	0.01	1.00E-02	9.76E-19	1	-1
62				90	9	0.01	1.00E-02	2.46E-04	1	-1
63				45	9	0.01	1.00E-02	1.32E-04	1	-1
64				10	9	0.01	1.00E-02	1.92E-04	1	-1
65			1%	409	9	0.0101	9.92E-03	2.372681	1	-1
66				90	9	0.01	9.99E-03	3.82E-04	1	-1
67				45	10	0.0102	1.00E-02	1.75E-04	1	-1
68				10	28	0	1.24E-03	2.81E-03	-1	-1

69				409	9	0.0101	9.92E-03	2.22E-04	1	-1
70				90	19	0	7.44E-03	2.08E-02	-1	-1
71			5%	45	47	0	6.60E-04	1.85E-03	-1	-1
72				10	138	0	2.61E-05	9.91E-05	-1	-1

(…)

Table 4.6. Results of the Sinusoidal Pattern Detection function applied to a ramp signal.

Test	f (Hz)	f_s (Hz)	Noise	IE	Periods Number	Periods Time (s)	$\bar{x}$ (s)	δ (s)	Sine Boolean	Stable Boolean
					(…)					
49				409	100	0.01	1.00E-02	1.71E-04	-1	-1
50				90	100	0.01	1.00E-02	1.71E-04	1	-1
51			0%	45	101	0.01	1.00E-02	1.40E-04	1	-1
52				10	101	0.01	1.00E-02	1.40E-04	1	-1
53				409	100	0.01	1.00E-02	5.02E-04	-1	-1
54		1000		90	100	0.01	1.00E-02	1.71E-04	1	-1
55			1%	45	100	0.01	1.00E-02	1.97E-04	1	-1
56				10	100	0.01	1.00E-02	1.71E-04	1	-1
57				409	100	0.01	1.00E-02	4.92E-04	-1	-1
58				90	101	0.01	1.00E-02	1.99E-04	1	-1
59			5%	45	100	0.01	1.00E-02	2.00E-04	1	-1
60	100			10	101	0.01	1.00E-02	1.99E-04	-1	-1
61				409	9	0.01	9.98E-03	6.67E-05	1	-1
62				90	9	0.01	1.00E-02	6.01E-05	-1	-1
63			0%	45	9	0.01	9.99E-03	3.33E-05	-1	-1
64				10	9	0.01	1.00E-02	9.88E-10	-1	-1
65				409	9	0.01	9.98E-03	6.67E-05	1	-1
66		10000		90	9	0.01	1.00E-02	9.88E-10	-1	-1
67			1%	45	9	0.01	1.00E-02	7.07E-05	-1	-1
68				10	9	0.01	1.00E-02	9.88E-10	-1	-1
69				409	9	0.01	9.98E-03	6.67E-05	1	-1
70				90	9	0.0101	9.98E-03	1.39E-04	-1	-1
71			5%	45	35	0	9.06E-04	1.95E-03	-1	-1
72				10	97	0	4.64E-05	1.48E-04	-1	-1

(…)

B. Results Analysis

Signal period calculation can be easily made considering the relationship expressed by equation (4.3) can easily support the result of the periods time calculation. Another aspect that helps to understand this function is that a big amount of noise can contributes to the failure of calculation of the period number and, as a consequence, makes difficult the calculate of the time period. This situation is due to the fact that, because the large amount of noise, several local maxima and minima occurs in the maxima and minima absolute signal, as can be the situation represented in Figure 4.8.

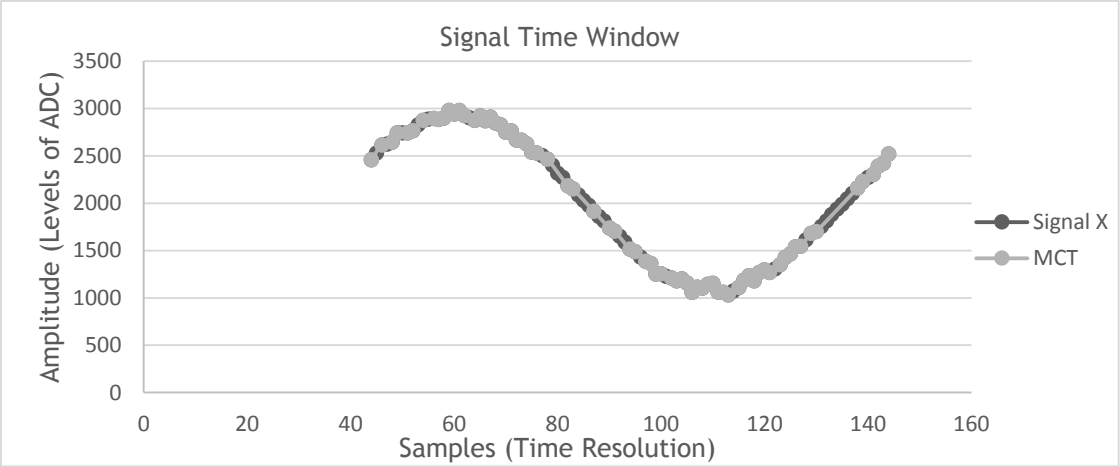


Figure 4.8. Graphical representation of a 100 samples time window of the sinusoidal signal test number 72.

Another parameter that proves the good performance of this function is that, as it was expected, the standard deviation is always less than half the value of the time period and the actual mean of the signal. Finally, in Table 4.5 it can seen that the waveform detection works properly using the Boolean variable declared for the same effect. This does not apply in Table 4.6 since that, in some cases, the ramp signal was identified as a sinusoidal signal. The causes of this artefact are due to the fact that the smaller the interpolation error, the higher the probability of more than one sample labelling order to define the absolute maxima and minima of the function and therefore consequently increasing the likelihood the sequence of classes "gefd" that define the signal as being sinusoidal.

4.2.3. Tendency Estimation

The estimation of signal's tendency is related with how to predict the steady state value to which the signal tends. In the presence of oscillations the signal is changing between a maxima and minima value. The calculation of the peak-to-peak difference value allows to know if the signal is increasing or a decreasing. A Boolean variable is used to store the trend direction. A trend defined as $+1$ means that the amplitude of a maximum is always greater than it's previous, unlike what happens when the trend is set as -1, which means that a maximum amplitude is less than the previous. The same methodology and reasoning applies to minimum values of the signal. The value of both can be obtained by equation (4.4) and (4.5):

$$Trend_{max} = \sum_{max} \frac{(T_{x+1} - T_x)\,(M_{x+1} - M_x)}{(T_{x+1} - T_x)\,|M_{x+1} - M_x|} \tag{4.4}$$

$$Trend_{min} = \sum_{min} \frac{(T_{x+1} - T_x)\,(M_{x+1} - M_x)}{(T_{x+1} - T_x)\,|M_{x+1} - M_x|} \tag{4.5}$$

It is easy to conclude that the extension of these formulas is from -1 to $+1$. In other words, the tendency is the average between this two, as shown in equation (4.6):

$$Trend = \frac{Trend_{min} + Trend_{max}}{2} \tag{4.6}$$

With the knowledge of the maxima and minima trends, separately, it is also possible to obtain information regarding the oscillatory behaviour presented by the acquired signal. If the trend of maxima and minima converges, the signal has stable oscillations. On the other hand, if a divergent tendency is observed, the signal is classified as possessing unstable oscillations. There is no need to analyse the entire signal, the definition of a small window is sufficient to predict the trend of the signal. Thus, the number of samples analysed in the same window is given, in seconds, by equation (4.7):

$$samples\_to\_analyse = \frac{TendencyWindow}{SamplingPeriod} \times 1000000 \tag{4.7}$$

The flowchart in Figure 4.9 represents schematically the implementation of the *Tendency Estimation* function.

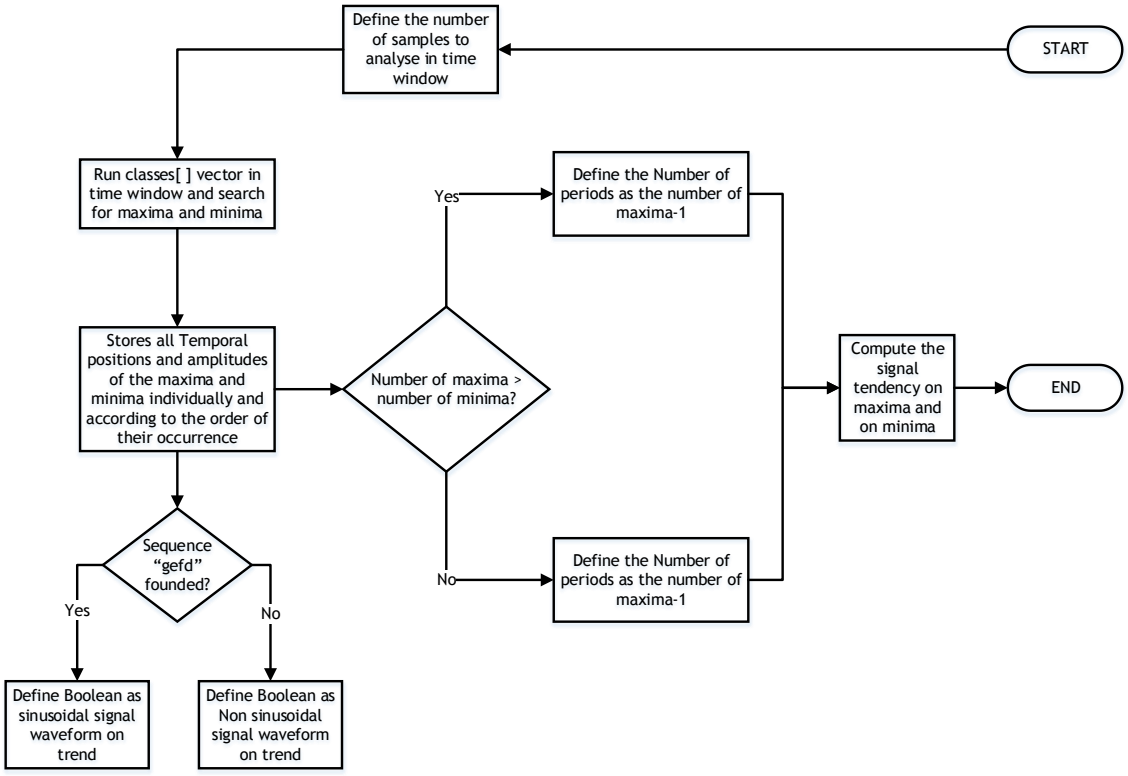


Figure 4.9. Flowchart of Tendency Estimation function.

In short, the above inputs and outputs are reported in the Table 4.7.

Table 4.7. Listing inputs and outputs of the Tendency Estimation function.

INPUTS	
$\boldsymbol{int\ SamplingPeriod}$	Integer that defines the frequency of acquisition. Expresses the time resolution is in microseconds.
$\boldsymbol{int\ TendencyWindow}$	Integer that defines the size of the windows in time. Expresses the time resolution is in seconds.
OUTPUTS	
$\boldsymbol{int\ SinusoidalDetectionOnTrend}$	Boolean variable that is defined as $+1$ if the signal has a sinusoidal shape, -1 if the signal does not have sinusoidal shape and 0 if the function cannot defined the signal shape.
$\boldsymbol{int\ number\_periods}$	Integer given by the number of maxima-1, or minima-1, depending on which occurs first.
$\boldsymbol{int\ MaxTendency}$	Calculated as aforementioned.
$\boldsymbol{int\ MinTendency}$	
$\boldsymbol{int\ Trend\_maxmin}$	

A. Experimental Setup

To test this function damped sinusoidal signals were used. Both rising and decaying were tested. Although they may be obtained by the generator signal modulation function, both signals were computed in an Excel document and then generated by the arbitrary function from the signals generator. It was hard to get any signal without breaks, even in burst mode, these signals were tested at sampling frequencies: 100Hz and 500Hz. For each sampling frequency, the following interpolation errors were also considered: 409, 90, 45 and 10 ADC levels. For all cases, SamplingPeriod it was set up as 10ms and the TendencyWindow it was set up as 1 second. The results obtained are listed in Table 4.8 and in Table 4.9.

Table 4.8. Results of the Tendency Estimation function applied to a signal presenting a rising tendency.

Test Number	Sampling Frequency	Interpolation Error	Sinusoidal Pattern on Trend	Periods Number	Maxima Trend	Minima Trend	MaxMin Trend
1	100 Hz	409	1	7	-1	1	0
2		90					
3		45					
4		10					
5	500Hz	409	1	7	-1	1	0
6		90					
7		45	-1	70	0	0	
8		10		89			

Table 4.9. Results of the Tendency Estimation function applied to a signal with a presence of a decaying tendency.

Test Number	Sampling Frequency	Interpolation Error	Sinusoidal Pattern on Trend	Periods Number	Maxima Trend	Minima Trend	MaxMin Trend
1	100 Hz	409	1	7	-1	1	0
2		90					
3		45					
4		10					
5	500Hz	409	1	7	-1	1	0
6		90					
7		45	-1	23	0	0	
8		10		83			

B. Results Analysis

Analysing the results presented in the tables above, it can easily be concluded that, overall, the performance of this function is quite satisfactory for sampling frequencies equals or lower than 100Hz, regardless of interpolation error. However, it is also possible to conclude that when the number of periods inside the time window is low, the signal will be evaluated by the more accurately function. For those signs, the signal wasn't modulated with noise because, given the reasoning similarity of this function with the previous function, the algorithm's response is predictable. This analysis applies to both signal cases represented by the scope images in Figure 4.10.

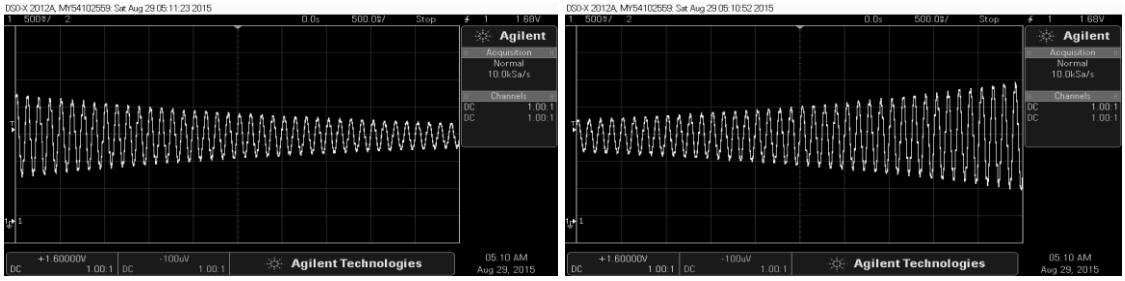


Figure 4.10. On the right, the scope image of a generated signal with a decaying tendency of 3.2V_{pp} of amplitude and 1.6V of offset; On the left, the scope image of a generated signal with a rising tendency of 3.2V_{pp} of amplitude and 1.6V of offset.

4.2.4. Exponential Detection

The exponential function can be characterized as an extension of the enhancement process. As it is shown in Table 10, there are four classes which describe four different cases of exponentials when repeated throughout the set of N samples. In particular, unlike the classes e and d, the classes f and g are related with signals owning a long-term value once both tend towards a steady state. This same value can be predicted using the MCT vectors. Thus, if the amount of time that the same class repeats is higher than or equal to a predetermined value, then the signal is classified as an exponential type. In addition, by accessing the position of the first segment defined by the class and the last segment described by the same class, it is possible to calculate the exponential time. The flowchart in Figure 4.11 represents schematically the implementation of the *Exponential Detection* function.

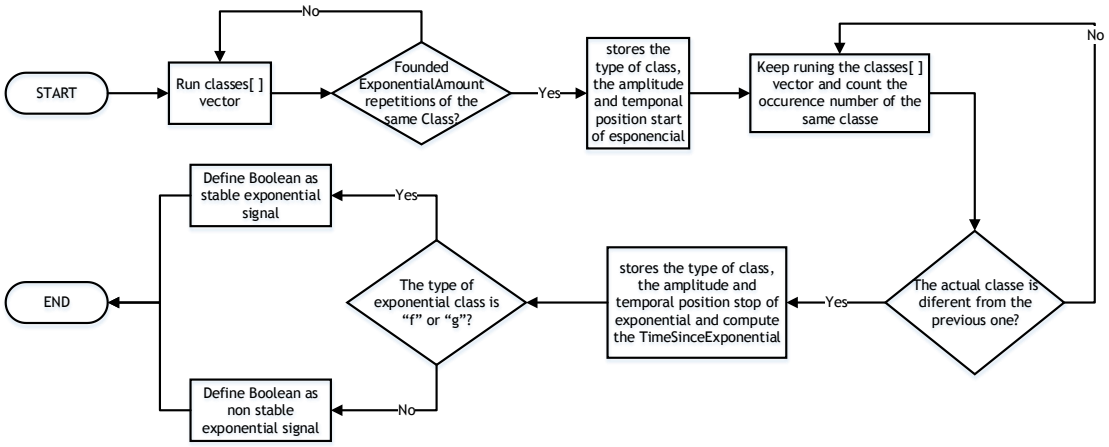


Figure 4.11. Flowchart of Exponential Detection function.

In short, the above inputs and outputs are reported in the Table 4.10.

Table 4.10. Listing inputs and outputs of the Exponential Detection function.

INPUTS	
$\boldsymbol{int\ SamplingPeriod}$	Integer that defines the frequency of acquisition. Expresses the time resolution is in microseconds.
$\boldsymbol{int\ ExponentialAmount}$	Integer that defines the number of consecutive segments required to consider an exponential start.
OUTPUTS	
$\boldsymbol{int\ ExponentialType}$	The class that represents the exponential signal shape. Expressed in ASCII code.
$\boldsymbol{int\ ExponentialStable}$	Assigned depending on the **ExponentialType, it is a** Boolean variable that is defined as $+1$ if the signal type is "g" or "f", -1 if the signal type is "e" or "d" and 0 if the function cannot define the signal shape.
$\boldsymbol{int\ TimeSinceExponential}$	Integer calculated as already mentioned above. It is expressed in microseconds.

A. Experimental Setup

To test this function a rising signal type "g" and decaying type "f" were both used. Signals were generated using arbitrary features provided by signals generator. It is hard to get a signal without breaks, even in burst mode, these signals were tested at sampling frequencies: 100Hz and 500Hz. For each sampling frequency, the following interpolation errors were also considered: 409, 90, 45 and 10 ADC levels. For all cases, SamplingPeriod it was set up as 10ms and the ExponentialAmount it was set up as 3. The results are listed in Table 4.11 and in Table 4.12.

Table 4.11. Results of the Exponential Detection function applied to a signal with a presence of a rising exponential.

Test Number	Sampling Frequency	Interpolation Error	Exponential Type	Exponential Stable	Time Since Exponential
1	100 Hz	409	71	1	7890000µs
2		90			5180000µs
3		45			6400000µs
4		10			400000µs
5	500Hz	409	0	0	0µs
6		90			
7		45	71	1	910000µs
8		10	0	0	0µs

Table 4.12. Results of the Exponential Detection function applied to a signal with a presence of a decaying exponential.

Test Number	Sampling Frequency	Interpolation Error	Exponential Type	Exponential Stable	Time Since Exponential
1	100 Hz	409	70	1	8220000µs
2		90			5860000µs
3		45			5370000µs
4		10			940000µs
5	500Hz	409	0	0	0µs
6		90			
7		45	70	1	998000µs
8		10	0	0	0µs

B. Results Analysis

From an analysis of the results presented in the tables above, it can easily be concluded that, overall, the performance of this function is quite satisfactory for sampling frequencies equals or lower than 100Hz, regardless of an interpolation error. However, it is also possible to conclude that for a higher number of samples labelled, the result of the signal evaluated by the function will be more accurate. The test case when the $f_s = 500\text{Hz}$ and $\text{error} = 10$, is an exception example, because the algorithm labels several samples but, at the same time, the interpolation error is low, this increases the chance of catching more than one class due to the presence of natural noise in the signal. This analysis applies to both signal cases represented by the generator images in Figure 4.11.

Figure 4.12. On the right, the generated exponential stable signal-rising (class type "g") in Burst Mode; On the left the generated exponential stable signal-decaying (class type "f") in Burst Mode.

4.2.5. Impulsive Noise Detection

Pulse events are related with power surges, and may have small or large amplitude but generally are short in time. The unwanted presence of such a signal comes from the interference of a short period of noise in the signal, being called noisy boost. Therefore, to be considered impulsive noise, i.e. as the presence of noise in the signal, it is made a maximum and minimum search. When the amplitude of a maximum peak is above a defined threshold added to the amplitude of the impulse base, is extracted knowledge by computing some variables, such as:

- The symmetry of the impulse, obtained by the impulse duration given by the time difference between of the positions before and after the impulse which are below the established threshold;
- The signal area within the impulse;
- The shape of the impulse wave;
- The impulse position in time of the peak;
- The impulse amplitude of the peak;
- The respective classes that define the impulse.

The flowchart in Figure 4.13 represents schematically the implementation of the *Impulsive Noise Detection* function.

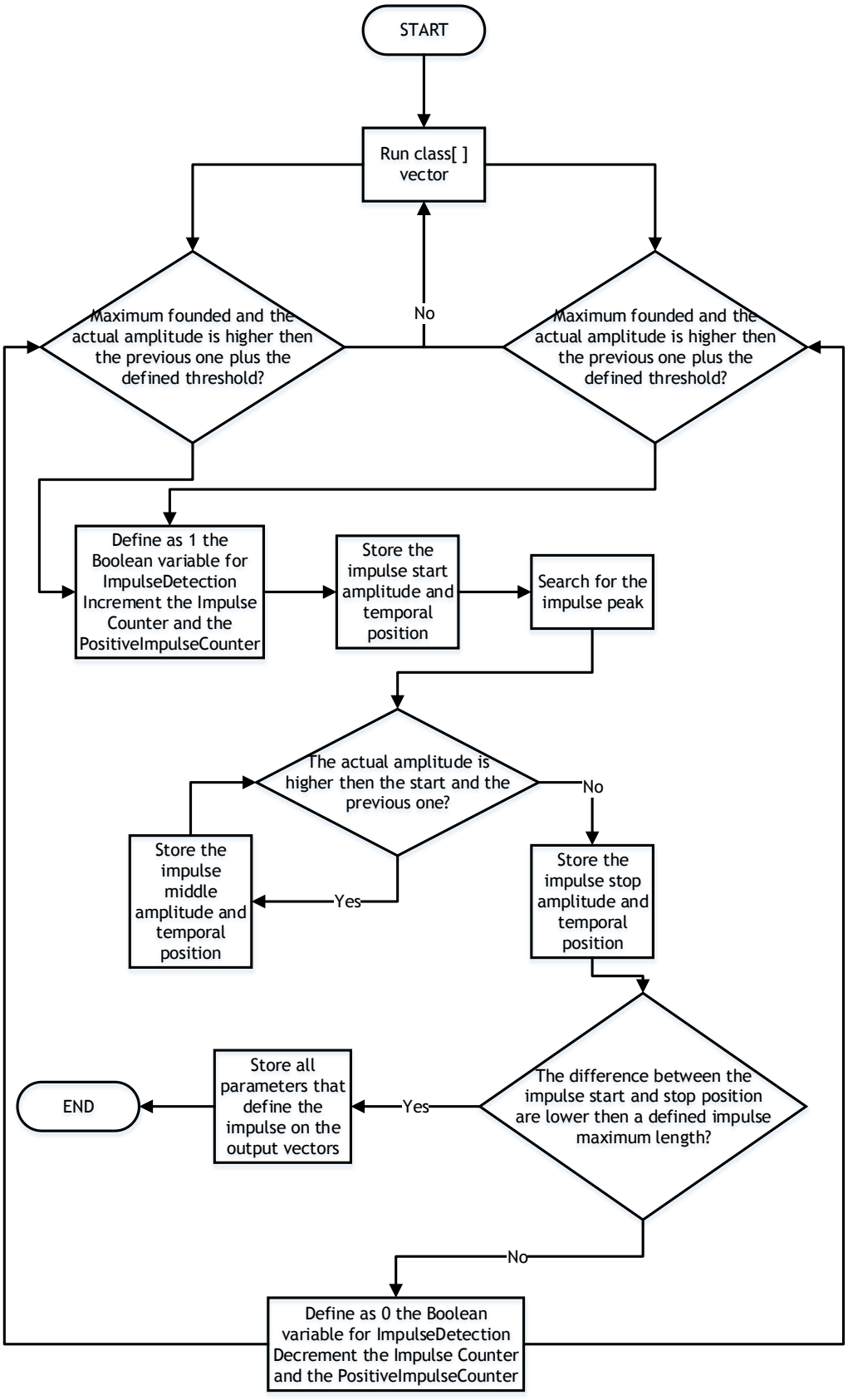


Figure 4.13. Flowchart of Impulse Noise Detection function.

In short, the above inputs and outputs are reported in the Table 4.13.

Table 4.13. Listing inputs and outputs of the Impulsive Noise Detection function.

INPUTS	
$\boldsymbol{int\ FullScale}$	Integer that represent the full scale range of the sensor. It is expressed in Volts.
$\boldsymbol{int\ ImpulsiveThreshold}$	Integer that defines the minimum amplitude change to be considered as an impulsive noise. The defined value is a percentage of the full scale.
OUTPUTS	
$\boldsymbol{int\ ImpulseDetection}$	Boolean variable that is defined as $+\mathbf{1}$ if a presence of impulses is detected in the signal, $-\mathbf{1}$ if not, $\mathbf{0}$ if is not possible to define.
$\boldsymbol{int\ ImpulseCounter}$	Integer that keep the number of impulses present on the signal.
$\boldsymbol{int\ PositiveImpulseCounter}$	Integer that keep the number of the positive impulses present on the signal.
$\boldsymbol{int\ NegativeImpulseCounter}$	Integer that keep the number of the negative impulses present on the signal.
$\boldsymbol{int\ ImpulseArea}[\,]$	Integer that storage the area of the impulses present on the signal. It is expressed in Volts*Seconds.
$\boldsymbol{int\ ImpulseShape}[\,]$	Integer that storage the shape of the impulses present on the signal. It is a dimensionless variable.
$\boldsymbol{int\ ImpulseLength}[\,]$	Integer that storage the length of the impulses present on the signal, given by the temporal difference between when it starts and when it ends. The number of samples in time domain expresses it.
$\boldsymbol{int\ ImpulsePosition}[\,]$	Integer that storage the time position of the impulses present on the signal. It is expressed in time domain.
$\boldsymbol{int\ ImpulseAmplitude}[\,]$	Integer that storage the amplitude of the impulses present on the signal. It is expressed in ADC levels.
$\boldsymbol{int\ ImpulseAsymmetry}[\,]$	Integer that storage the asymmetry of the impulses present on the signal. It is expressed in Volt/Seconds.
$\boldsymbol{int\ ImpulseAsymmetryType}[\,]$	Boolean variable that compere the asymmetry position with zero defining as $+1$ if exists presence of a positive impulses at maximum, -1 if exists presence of a negative impulses at maximum, $+2$ if exists presence of a positive impulses at minimum, -2 if exists presence of a negative impulses at minimum and 0 if is not possible to define.
$\boldsymbol{int\ ImpulseType}[\,][\,]$	Integer that storage the classes that constitute the impulses present on the signal. Expressed in ASCII code.

A. Experimental Setup

To test this function a DC signal was used with some random impulses added and a sinusoidal signal with some impulses. Both signals were computed in an Excel document, built by the ArbExpress Application software and, after, generated by the arbitrary function from the signals generator. Generated at 1Hz of signal frequency, these signals were tested at sampling frequencies: 100Hz, 500Hz, 1kHz and 5kHz. For each sampling frequency, following interpolation errors were also used: 409, 90, 45 and 10 ADC levels. For all cases, FullScale it was set up as 33 and the ImpulsiveThreshold it was set up as 10. Some of the results are listed in Table 4.14 and in Table 4.15.

Table 4.14. Results of the Impulse Noise Detection function applied to a DC signal with some random impulses added.

Test Number	Signal Frequency	Sampling Frequency	Interpolation Error	Impulse Detection	Impulses Number	Positive Impulses	Negative Impulses
1	1Hz	100Hz	409	1	26	23	3
2			90		36	16	20
3			45		36	13	23
4			10		42	20	22
(…)		(…)					
13		5kHz	409	1	2	1	1
14			90		3	3	0
15			45		2	2	0
16			10		3	3	0

Table 4.15. Results of the Impulse Noise Detection function applied to a sinusoidal signal with some impulses.

Test Number	Signal Frequency	Sampling Frequency	Interpolation Error	Impulse Detection	Impulses Number	Positive Impulses	Negative Impulses
1	1Hz	100Hz	409	1	24	5	19
2			90		26	16	10
3			45		26	9	17
4			10		20	4	16
(…)		(…)					
13		5kHz	409	1	0	0	0
14			90		1	0	1
15			45		0	0	0
16			10		1	1	0

B. Results Analysis

The results from tests reveal the existence of a certain restriction as to obtain acceptable results. This is revealed in cases where, for the same frequency signal, the sampling frequency is high and the interpolation error is low. It happens the same for the opposite case, i.e. for cases where the sampling frequency is reduced and the interpolation error is increased. Observing the tested results of each signal individually.

For the first case, the DC signal with addition of random impulses in the first three tests, the interpolation error revealed a strong influence on the performance of the function. Under these conditions, the algorithm labels fewer samples in the DC signal causing, thereby, that some of the maximum peaks are classified as minimum due to the classes that compose them, and vice versa. Conversely, in the other cases, as the interpolation error is low, although there is visual confirmation of the results to function detects impulses in the DC signal. To understand the sign in question, visualize Figure 4.14.

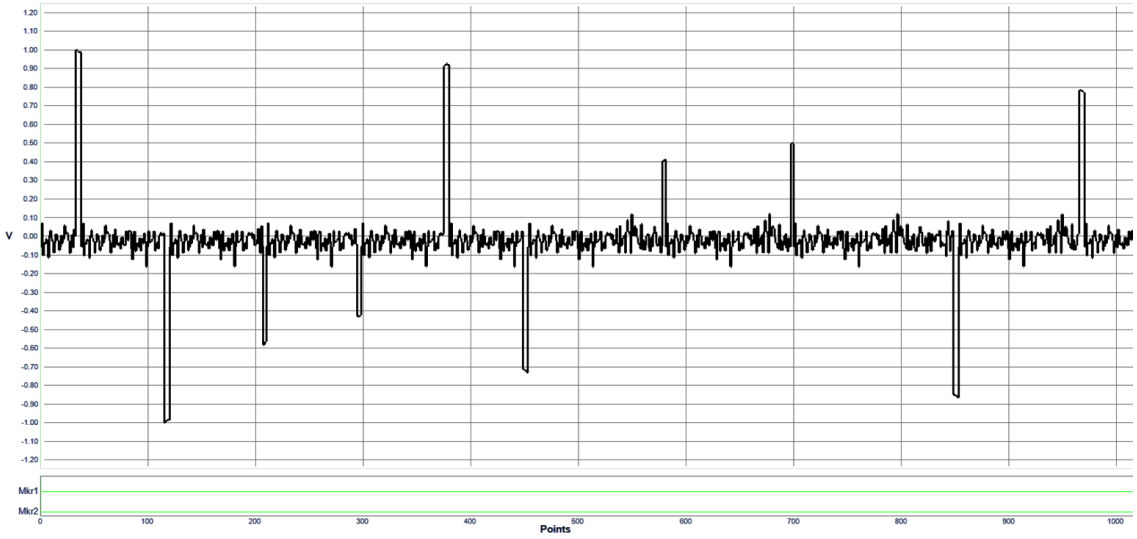


Figure 4.14. Cropping out the generation of the DC signal, with some random impulses added, by ArbExpress software.

For the second case, to the sinusoidal signal random impulses are added, the vulnerability is highlighted also by the maxima and minima absolute sinusoidal. For all the other cases, beyond visual confirmation of the results, the function detects impulses according the intentions previously described. To understand the sign in question, visualize Figure 4.15.

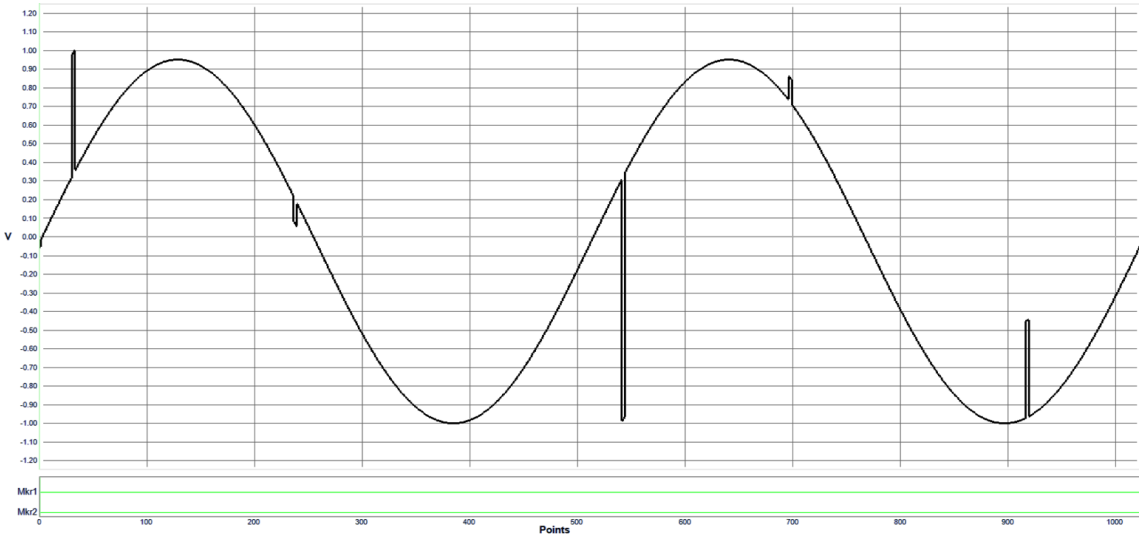


Figure 4.15. Cropping out the generation of the sinusoidal signal, with some impulses, by ArbExpress software.

In short, for signal frequencies exceeding the ones tested, it is expected a reduction on the function responsiveness. The absence of tests with a higher sampling frequency is because, by this way, it would be obtained a very small fraction of the signal and it will show no interest in what comes to evaluation of this function. Finally, to obtain better results, it is suggested to decrease the signal frequency.

4.2.6. Mean Estimation

Signal mean estimation is defined as the value that is more reliable. Given the knowledge obtained by previous function can be excluded, in the estimation of signal mean, some artefacts. Thus, the segments that belong to impulsive noise are discarded from the mean computation. The flowchart in Figure 4.16 represents schematically the implementation of the *Mean Estimation* function.

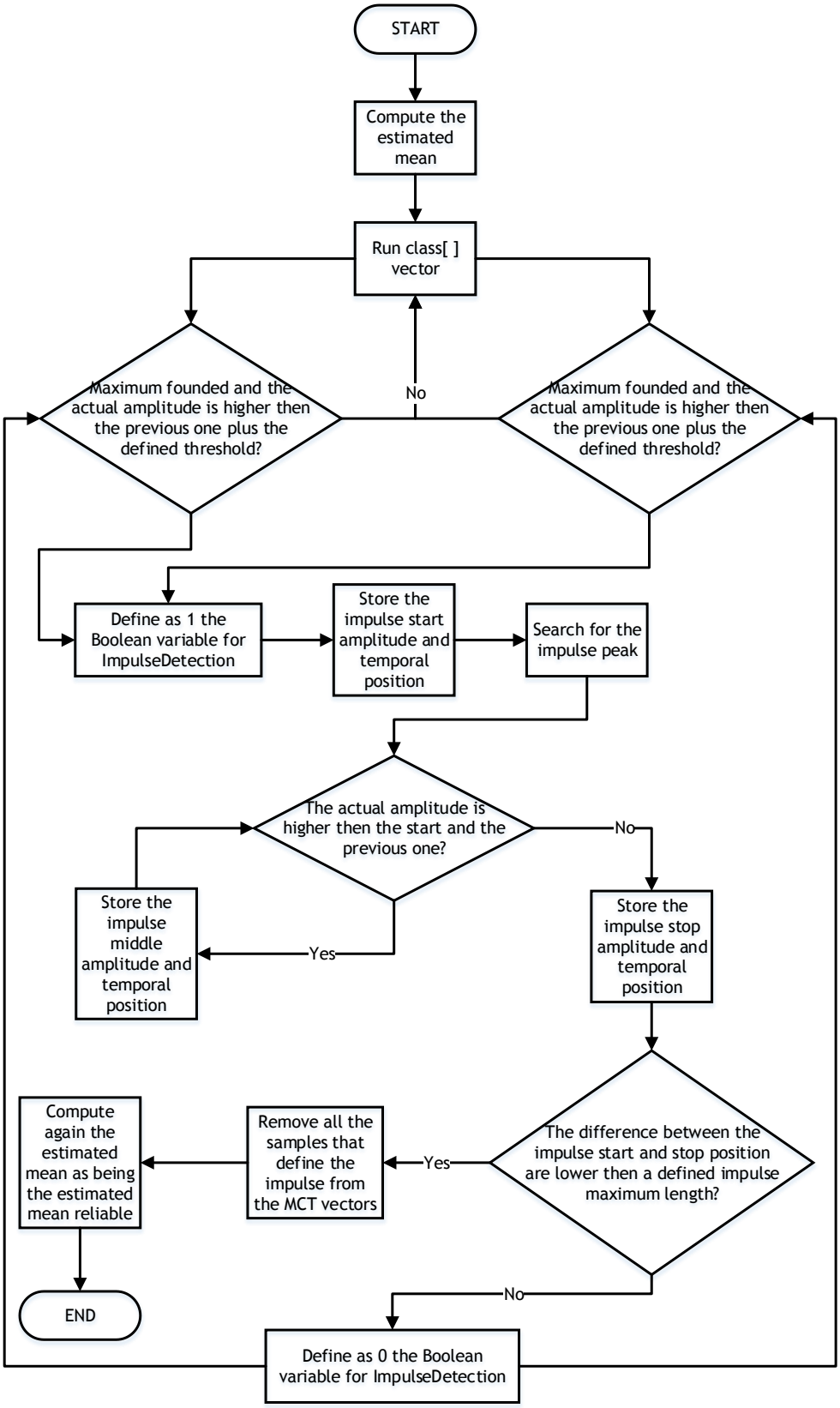


Figure 4.16. Flowchart of Mean Estimation function.

In short, the above inputs and outputs are reported in the Table 4.16.

Table 4.16. Listing inputs and outputs of the Mean Estimation function.

INPUTS	
$int\ FullScale$	Integer that represent the full scale range of the sensor. It is expressed in Volts.
$int\ ImpulsiveThreshold$	Integer that defines the minimum amplitude change to be considered as an impulsive noise. The defined value is a percentage of the full scale.
OUTPUTS	
$int\ SIZEOFARRAY$	Length of the MCT vectors after removing of the impulses located by the Impulse Noise Detection function.
$float\ EstimatedMean$	Keep the calculation of the amplitudes average stored in the marks[n] vector. It is a dimensionless variable.
$float\ EstimatedMeanReliable$	Keep the new calculation of the amplitudes average stored in the marks[n] vector, after removing of the impulses located by the Impulse Noise Detection function. It is a dimensionless variable.

A. Experimental Setup

The same signals from the previous function were used to evaluate this function: a DC signal with random impulses added; and a sinusoidal signal with impulses added. Both signals were computed in an Excel document, built by the ArbExpress Application software and, after, generated by the arbitrary function from the signals generator. Generated at 1Hz of signal frequency, these signals were tested at sampling frequencies: 100Hz, 500Hz, 1kHz and 5kHz. Each sampling frequency was evaluated using the following interpolation errors: 409, 90, 45 and 10 ADC levels. For all cases, FullScale it was set up as 33 and the ImpulsiveThreshold it was set up as 10. Some of the results are listed Table 4.17 and in Table 4.18.

Table 4.17. Results of the Mean Estimation function applied to a DC signal with some random impulses added.

Test Number	Sampling Frequency	Interpolation Error	End Of Array	Size Of Array	Estimated Mean	Estimated Mean Reliable
1	100Hz	409	139	93	1983.281	1910.591
2		90	496	434	1967.873	1968.931
3		45	748	636	1951.608	1962.761
4		10	971	929	1954.863	1953.628

			(…)			
13	5kHz	409	26	16	2056.269	1969.75
14		90	179	173	1969.698	1943.04
15		45	344	377	1953.831	1939.134
16		10	564	553	1967.342	1955.206

Table 4.18. Results of the Impulse Noise Detection function applied to a sinusoidal signal with some impulses.

Test Number	Sampling Frequency	Interpolation Error	End Of Array	Size Of Array	Estimated Mean	Estimated Mean Reliable
1	100Hz	409	140	91	1892.421	2211.692
2		90	305	183	1992.108	1946.437
3		45	397	247	1986.587	2264.603
4		10	730	494	1931.914	2076.453
			(…)			
13	5kHz	409	1	1	3670	3670
14		90	17	17	1839.941	1839.941
15		45	21	21	2919.095	2919.095
16		10	329	321	2262.204	2219.595

B. Results Analysis

From both tables it is visible that the pulses detected by the previous function have been removed from the MCT vectors. The same is evident not only from the results presented in tables, in particular the difference between the feedback variable SIZEOFARRAY and ENDOFARRAY, but also by the flowchart of this function. Given that the computational reasoning of Mean Estimation function and Impulsive Noise Detection function is very similar, it is easily understandable that this function is receptive to the vulnerabilities of the previous function. This can also be proved through the tabulated results. Remembering that the tests carried out were the basis for the application of both functions, preforming a comparison the values of EstimatedMeanReliable and EstimatedMean, it can checked that in some cases the difference between these variables is zero or below while the expected would that the difference was greater than zero.

Chapter 5

In this final chapter, the final remarks of this document will be made, and also be presented the conclusions withdrawn from the obtained results after experimental evaluation. Some suggestions for perspectives of future work will also be presented.

5. Conclusions

Living in an interconnected world, the interaction between the sensors and actuators is utilized, if he took it from the results obtained in this study field. The rules that manage how the data is identified and how it is shared are defined in the IoT concept. It also comes from the common sense that information is only useful when properly processed and shared with all the interested. Thus, smart sensors are now becoming more important than ever. Its possible today to process all the collected information collected from sensors at the point of measurement. There are several techniques that could be included in smart sensors, as an example of it, two analysis and processing algorithms were here presented: the well-known FFT algorithm; and the CS technique. Being both based on a standard signal analysis and processing techniques, to supplement the information provided by a smart sensor control and monitoring system, both of these algorithms can be seen as complete sampling procedures.

Technological evolution requires the adaptation of standards and the inclusion of higher processing capabilities in the transducer. As a consequence, this operation is very complex. This work proposes a new operating principle based in an algorithm that works in the time domain. A pre-processing stage is used, where the signal obtained from the sensor is first sampled and after segmented and labelled. This process converts the periodically sampled signal in a vector containing only the samples that have more representativeness. Although all the algorithms have as common goal the extraction of information from the acquired data, the one here proposed proved to be more effective. This is justified by the fact that the same amount of information is extracted, after the data reduction. Whenever a signal is oversampling, a high degree of correlation between samples is observed. Under these conditions, the complexity of the algorithmic is simplified, and this situation is supported by the fact that the process of segmenting and labelling the original vector of samples is obtained from a classification of the signal in eight classes. Moreover, the method here described offers an excellent platform to perform data mining. This platform, whose information is uniformly standardized, it is composed of three vectors of three vectors MCT. The first is for the amplitude storage of the essential samples, the second for the classification label of the segments behaviour and the third for their respective position in time.

In communication networks, wired or wireless, the need to transfer less data means consuming less energy and using less data bandwidth. Therefore, this revealed a considerable reduction in computational cost because the amount of data to be transferred is also significantly reduced. Moreover, comparatively with the above algorithms, the proposal here presented is characterized by a reduction in the implementation complexity, thus the processing time is also reduced. Being aware of the signal domain, in addition to the

information extraction and knowledge about the signal and its behaviour, emulating the Human observer and its principles, the S&L algorithm also allows the reconstruction of the input signal. The knowledge of interest that can be extracted and inferred more accurately lies between: frequency, amplitude, presence of noise, pattern detection, etc.

To support the software engineering involved on the proposed project, there were presented some hardware and software: the source code compiler, interface environment for documentation, version control, waveform creator and editor, as well as the electronic devices for the programming project implementation.

This document described each function that make part of the API, as well as, the respective experimental setups and consequent analysis and feedbacks withdraw at the end of the tests. These results comproved that the IEEE 1451 standard, once approved it will have a big impact on the way how sensors are developed and used in the near future. Naturally, conducting standardized approaches offers a wide range of advantages. The proposed functions, not only can be incorporated into a smart sensor, but can also be applied in different fields of research multiplexed, being capable of measuring any kind of sign given by them. Exemplifying:

- Noise detection function: is often associated with systems that originate signals of electrical nature, digital, optical or even acoustic;
- Sinusoidal Pattern Detection function: associated with systems that have the ability to produce sequences, for example: the emission of unmodulated radio waves;
- Tendency Estimation function: systems related with physical systems, such as damped systems, for example, a mass-spring damped system;
- Exponential function: ideal for systems where the occurrence of a leak is a possibility, for example current leakage in a capacitor;
- Impulsive Noise Detection function and Mean Estimation function: can be both used in natural stable systems, like a system that measures the temperature from a system. These kinds of systems suffer external perturbation with short time duration.

Thus, smart sensors allow to extend the knowledge about of a specific domain. In addition, it is possible to perform the implementation the programming project proposal in a simple microcontroller, whose requirements and computational resources are reduced. The proposal aimed to improve the use of smart sensors in different kind of applications, such as:

- Smart filtering: based in the behaviour of the signal the filtering task will be more efficiently;

- Pattern recognition: by detecting a specific sequence of segments, it is able to identify the type of waveform signal, as well as, some of their main characteristics;
- Signal prediction: once the intelligent transducer extract knowledge about the acquired signal shape, it can also can predict future values, including values in steady state;
- Sensor fusion: sensor collaboration will possible to use the knowledge obtained from all the transducers to make more reliable results, and at the same time validate the measurements and infer system's different states of operation.

At last note, the current implementation of the API doesn't support yet the real time operation. As a perspective of future work, this mode of operation can be achieved in smart sensors since the computational cost is very low.

6. References

[1] P. Hu, R. Robinson, and J. Indulska, “Sensor Standards : Overview and Experiences,” no. ii, pp. 485–490, 2007.

[2] S. S. Guia, “A Comparison Between FFT and MCT for Period Measurement with an ARM Microcontroller.”

[3] F. Abate, S. Member, V. Huang, S. Member, and G. Monte, “A Comparison Between Sensor Signal Preprocessing Techniques,” no. 1, pp. 1–9.

[4] G. Monte, N. Scarone, A. Hossian, P. Liscosvky, U. T. Nacional, F. Regional, and D. Neuquén, “Proposal of a Standardized Treatment for Transducers Signal Behavior.”

[5] G. Monte, Z. Liu, F. Abate, V. Paciello, a. Pietrosanto, and V. Huang, “Normalizing transducer signals: An overview of a proposed standard,” *2014 IEEE Int. Instrum. Meas. Technol. Conf. Proc.*, pp. 614–619, May 2014.

[6] A. N. E. W. Ieee, S. Standard, W. Make, I. T. Easier, T. O. Set, M. Data, A. Systems, T. Contain, and A. L. Number, “A NEW IEEE SENSOR STANDARD WILL MAKE IT EASIER TO SET UP AND,” 2004.

[7] D. Wobschall, “IEEE 1451—a universal transducer protocol standard,” *2007 IEEE Autotestcon*, pp. 359–363, Sep. 2007.

[8] R. L. Oostdyk, C. T. Mata, and J. M. Perotti, “A Kennedy Space Center Implementation of IEEE 1451 Networked Smart Sensors and Lessons Learned,” *2006 IEEE Aerosp. Conf.*, pp. 1–20, 2006.

[9] S. Manda and D. Gurkan, “IEEE 1451 . 0 Compatible TEDS Creation Using . NET Framework,” 2009.

[10] G. Monte, “Sensor signal preprocessing techniques for analysis and prediction,” *2008 34th Annu. Conf. IEEE Ind. Electron.*, pp. 1788–1793, Nov. 2008.

[11] G. Monte, “A proposal of a new standard for sensor signal analysis,” *2010 IEEE Int. Conf. Ind. Technol.*, pp. 1637–1642, 2010.

[12] R. S. Lo Moriello, “On the Suitability of Compressive Sampling for the Measurement of Electrical Power Quality,” no. i.

[13] F. Franchetti and M. Puschel, “The fast Fourier transform,” *Spectrum, IEEE*, no. 2, pp. 1–14, 2000.

[14] J. Suto and S. Oniga, “A Simple Fast Fourier Transformation Algorithm to Microcontrollers and Mini Computers,” vol. 1, no. 8, pp. 61–65, 2014.

[15] J. Takala and K. Punkka, “Butterfly unit supporting radix-4 and radix-2 FFT,” *Proc. 2005 Int. TICSP …*, no. 2, 2005.

[16] I. Standard, E. Song, and K. Lee, “Smart Transducer Web Services Based on the,” pp. 1–6, 2007.

[17] P. Huincul, “A novel time-domain signal processing algorithm for real time ventricular fibrillation detection,” 2011.

[18] F. Abate, V. Paciello, A. Pietrosanto, G. P. Ii, and F. Sa, “Period measurement with an ARM microcontroller,” pp. 1–4, 2015.

[19] A. Pietrosanto, “A Comparison Between FFT and MCT for Period Measurement with an ARM Microcontroller,” pp. 1–5, 2015.

[20] “Keil Embedded Development Tools for ARM, Cortex-M, Cortex-R4, 8051, C166, and 251 processor families.” [Online]. Available: http://www.keil.com/. [Accessed: 01-Dec-2014].

[21] “Doxygen: Main Page.” [Online]. Available: http://www.stack.nl/~dimitri/doxygen/index.html. [Accessed: 27-Sep-2015].

[22] “ArbExpress® Signal Generator Software | Tektronix.” [Online]. Available: http://www.tek.com/product-software-series/arbexpress-signal-generator-software. [Accessed: 20-Sep-2015].

[23] S. Cortex-m, “Programming manual,” no. April, 2010.

[24] S. A. Mcus, “Reference manual,” no. January, 2011.

[25] K. B. Flash, “STM32F103x8,” no. August, pp. 1–105, 2013.

[26] B. Microcontroller and A. R. M. Cortex-m, “ET-STM32F103.”

[27] “STM32F103 ARM Cortex M3 Development Board.” [Online]. Available: http://www.gravitech.us/starmcom3deb.html. [Accessed: 09-Dec-2014].

[28] “STM32F103RB Mainstream Performance line, ARM Cortex-M3 MCU with 128 Kbytes Flash, 72 MHz CPU, motor control, USB and CAN - STMicroelectronics.” [Online]. Available: http://www.st.com/web/catalog/mmc/FM141/SC1169/SS1031/LN1565/PF164487?sc=internet/mcu/product/164487.jsp. [Accessed: 09-Dec-2014].

[29] “ET-STM32 Stamp.” [Online]. Available: http://www.futurlec.com/ET-STM32_Stamp.shtml. [Accessed: 12-Sep-2015].

[30] B. S. Stm, “User ’ s Manual of Board Microcontroller “ ET-ARM STAMP STM32 (Cortex-M3).”

[31] “mbed | welcome.” [Online]. Available: https://mbed.org/. [Accessed: 01-Dec-2014].

[32] “mbed LPC1768 | mbed.” [Online]. Available: http://developer.mbed.org/platforms/mbed-LPC1768/. [Accessed: 01-Dec-2014].

[33] “STSW-MCU005 STM32 and STM8 Flash loader demonstrator (UM0462) - STMicroelectronics.” [Online]. Available: http://www.st.com/web/en/catalog/tools/PF257525. [Accessed: 09-Dec-2014].

[34] “Termite: a simple RS232 terminal.” [Online]. Available: http://www.compuphase.com/software_termite.htm. [Accessed: 09-Dec-2014].

[35] “Keil MDK-ARM Version 5 - Legacy Support.” [Online]. Available: http://www2.keil.com/mdk5/legacy/. [Accessed: 09-Dec-2014].

[36] “Doxygen Manual: Documenting the code.” [Online]. Available: http://www.stack.nl/~dimitri/doxygen/manual/docblocks.html. [Accessed: 27-Sep-2015].

[37] “Structural commands: structcmd.h File Reference.” [Online]. Available: http://www.stack.nl/~dimitri/doxygen/manual/examples/structcmd/html/structcmd_8h.html. [Accessed: 27-Sep-2015].

[38] “Doxygen Manual: Doxywizard usage.” [Online]. Available: http://www.stack.nl/~dimitri/doxygen/manual/doxywizard_usage.html. [Accessed: 28-Sep-2015].

[39] “TortoiseSVN - Home.” [Online]. Available: http://tortoisesvn.net/. [Accessed: 30-Mar-2015].

[40] “VisualSVN - Subversion-based version control for Windows.” [Online]. Available: https://www.visualsvn.com/. [Accessed: 30-Mar-2015].

Annex A

The applied tools for programming projects implementation are the compilers. These software allow building objects code, editing and running source code debug. Generally, these tools are offline and desktop since, generally speaking, are software for private and particular use. However, these can also be online in order to allow multiple users to work in the development of the same project. Here, it will be presented an example for each type of compiler, Mbed as an online software and Kiel as an offline desktop tool, how to use them to interact with the user and with microprocessors, and yet, how is possible to interoperate between these two types of compilers.

The compilers are programs applied to convert a high level programming language into an object code. They can be offline, for a particular and personal development, or online, also for personal development but allowing the sharing with other users. Desktop compilers produce an output object code for the underlying microprocessor. Then, a complier is defined as a program that analyses and execute each line of source code in succession, translating source code into an object code. In the present attachment it will be presented one example for each type of compiler. The Mbed as a web platform based on an online compiler for developers and the Keil as an offline desktop compiler for programmers. Furthermore, there will also be presented the possibility to transfer a project to the offline compiler from the online platform.

I. Getting Started with an Online Platform Based on a Compiler

To get started with a soft interaction, it is necessary to understand a little more about the online compiler. Considering this, to do a demonstration more intuitive for understanding the organization of the online Mbed web home page [31], observe the Figure A.1 and his correspondent legend.

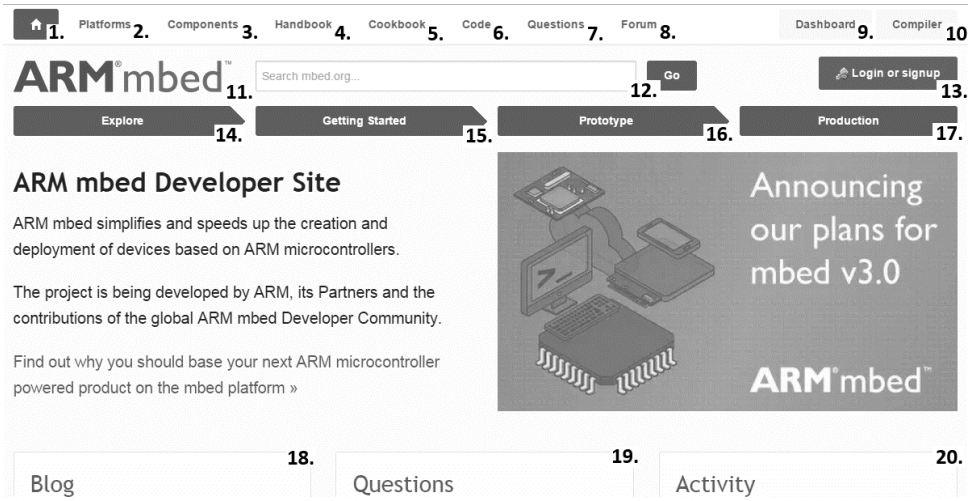


Figure A.1. Online developer platform.

Legend:

1. **Home -** give access to the home page;
2. **Platforms -** get a general view and individual information details of all platforms available;
3. **Components -** get knowledge and individual information details of all components available;
4. **Handbook** - online help to getting started working with boards and the online platform;

5. **Cookbook** – online accesses to information and tutorials;
6. **Code** – database with code projects and his correspondent information;
7. **Questions** – area to make a question or see the more frequent questions;
8. **Forum** – place to open or private discussions;
9. **Dashboard** – repositories to team works;
10. **Compiler** – online compiler platform;
11. **ARM mbed** – Mbed home page;
12. **Search** – browser to respond at the user necessities;
13. **Login or Signup** – access to the developer private area;
14. **Explore** – to know more about the Mbed environment;
15. **Getting Started** – quickly access to a productive and easy started;
16. **Prototype** – teaches how to do to publish personal project so that it is available for any user to use, and also allows permits contribute back fixes, libraries and support for other developers to help everyone get things done even faster;
17. **Production** – when the prototype proves to be success, the Mbed platform can also help support to develop it into a product;
18. **Blog** – a normal blog where users can turn public they last developing news;
19. **Questions** – shortcut to access the questions area;
20. **Activity** – it is also connected to the personal dashboard.

Having now a clearer idea of paging, accessing to the board page through the *platform* tab from reference [32], it can be seen a classic *Hello World!* program implementation that it can be imported to compiler area by pressing the button *import program* on the upper right corner of the code window. Doing this, it will automatically open the online compiler and a small importation window given the URL source and asking if the user pretends to import the project as a program or a library, the import name and if the user pretends to make an update of all libraries to latest revision. Pressing the *import* button a paste with the chosen name for the project will appear and the respective code that allows the user change the source code. With the menus represented on Figure A.2 it is possible for the user to make several operations.

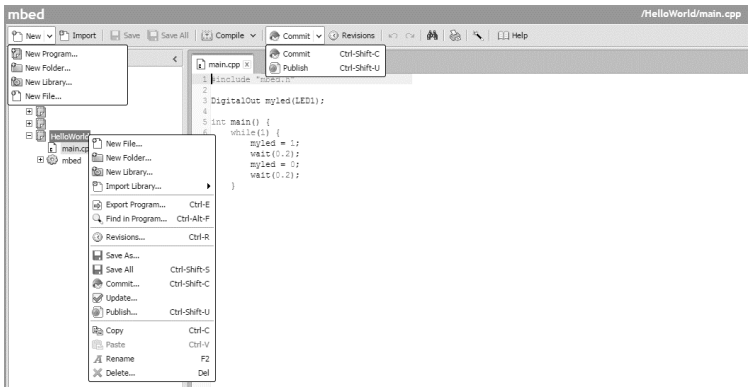


Figure A.2. Available menus on the Mbed online compiler platform.

If the user pretends to build an object code and see the output of the code, first, he has to select a platform. To do that, access the platform list through the through the button positioned on the top right of the Figure A.3.

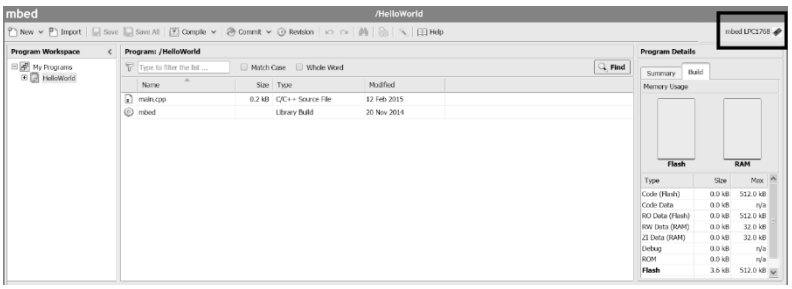


Figure A.3. Accessing the registered target board list.

A window like the one on Figure A.4 will show up with the registered target board list for selection.

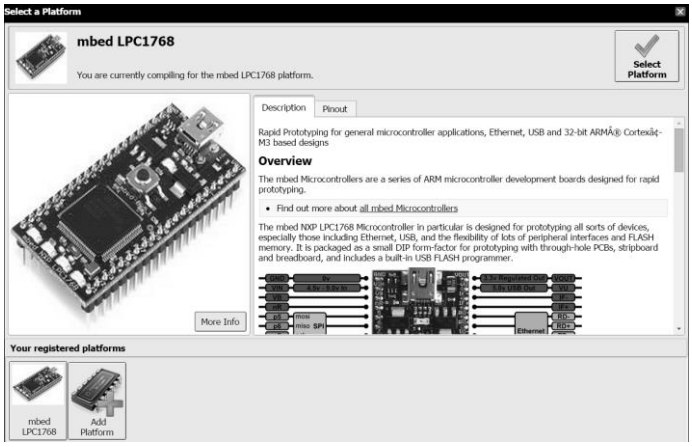


Figure A.4. Selecting a platform.

Select the desired platform from the list that appears on the window bottom. If the desired platform does not appear in the list of registered platforms, click on *Add Platform* and follow the steps to choose the desired target board. Once this step is finished, select the *Compile* tab from the compiler tools bar and save the project on the board folder. After this, just press the reset button to see the output of the code that, in this case, observed the LED1 flashing. On the other hand, to create a project from scratch, use the available option on *New* menu.

II. User Guide For an Offline Desktop Compiler

An offline desktop compiler is quite similar to an online web platform for programming. Considering this, to do a demonstration more intuitive for understand the organization of an offline tool, it will be used the version 5.12.0.0 of the µVision Keil software installed on the Ultimate version of the Windows 7 (64-bits). To install this software on the available OS, access the executable setup file through the *Download* tab from reference [20]. The installation is quite easy but, in case of errors occurrence, refer to online help. Here it will be seen how to create a project in a simple way but first, it will be demonstrated how to select the target board. Access at menu *Project* from the tools bar and select the option *Select Device for Target* to open the window with the list of targets available, as shown on Figure A.5.

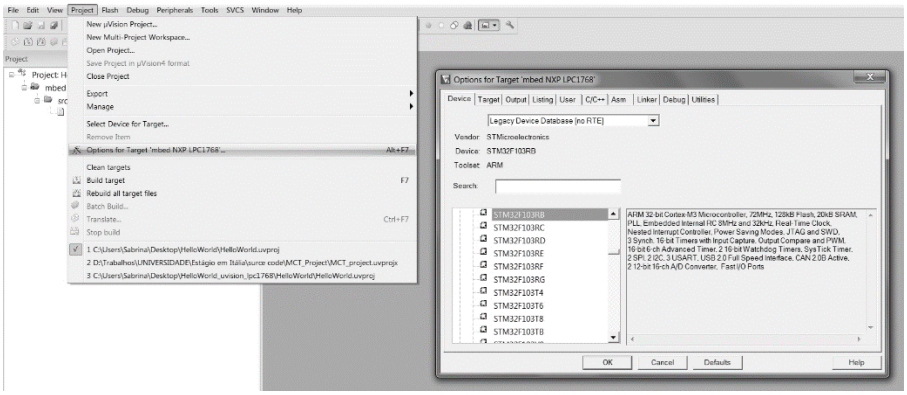


Figure A.5. Selection of the destination Target.

Select the desired target device from the list that appears on the window. Now, to create a project from scratch, start by selecting *New µVision Project…*, from the *Project* tab, and give to him a desired name and directory, as shown on Figure A.6. After this, a window will appear to allow the user to choose the target device from an available list. The software will create a file for the project. Now, to open a file to implement source code access *New*, from the menu *File* of the tools bar, and save it on the same directory of the project and select *Build Target* to see the output as explained further more.

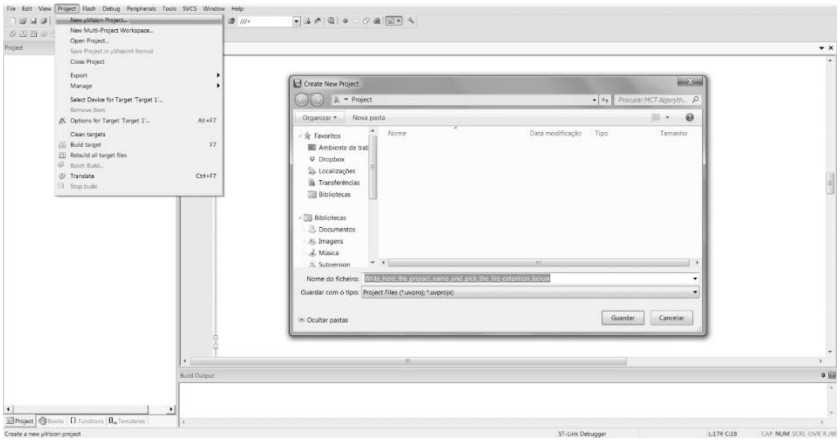


Figure A.6.Creating a new project on Keil compiler.

The program implementation finished and without problems, errors or warnings in *Build*, to see the output of the code, just save the project on the initial directory selected. If the user only want to see the output on the board, it will just be needed the last version of the program *Flash Loader Demonstrator* from reference [33]. The execution of its installation is quite simple, easy and intuitive. If the user pretends to see an output on the computer screen has, also, to make the download of the last version of the program *Termite* from the reference [34]. In this way, connect the board to the computer using a USB cable and a RS232 communication cable and search in wish *PORT COM* it is connected. After this, execute the program *Flash Loader Demonstrator*. In the first widow that show up, select the *PORT COM* correspondent with the board and click next. In the next window, make sure that the traffic lights symbol shows a green light and click next again. Now, select the corresponding target and in the next window to preform de *Download to Device* from the project directory, click next, and wait until the download is finished and close the program. Now, open the program *Termite* and connect to the correspondent *PORT COM*. With this, pressing the *RESET* button from the board, the microcontroller will start running the code and sending the output to the screen. Note that is important that the settings are correctly selected. For more information, please refer to the online help on [20] and seek everything necessary to reach an advanced use of this compiler.

III. Transferring a Project from Mbed to Keil

Assuming that the user already have done a project on the Mbed online compiler. To export a project from Mbed, only need to execute the steps as represented on Figure A.7.

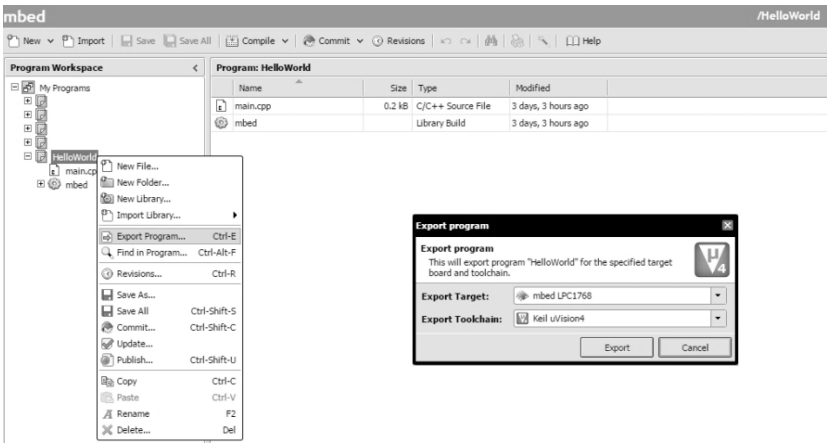


Figure A.7. Exportation of a project from the Mbed platform to Keil.

Here, the *Export Target* is already defined by default. It is possible to define Keil as the *Export Toolchain* before do the exportation to the computer user. Posteriorly, open the available version of Keil and use the menu *Project* of the tools bar to open the project on the work environment, from the directory were it was previously saved, as it is showed in the procedure on Figure A.8.

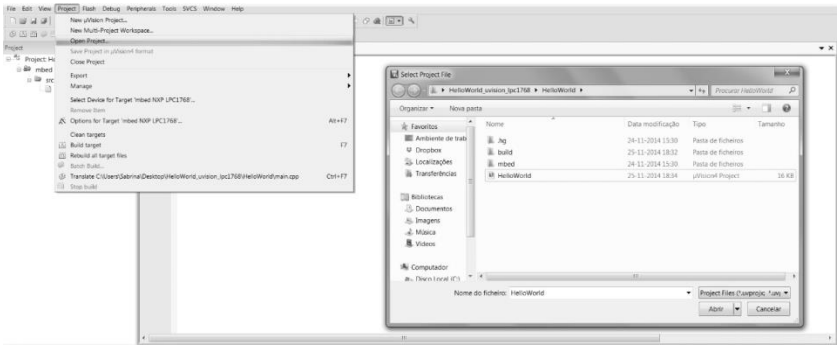


Figure A.8. Opening on KEIL an exported project from the Mbed platform.

The user must check in this version of this software if he has all the packages necessary to support this project in the target board that he pretends to use. If not, he just need to proceed to the respective download from reference [35] and install them in his computer. After this, open the software work space. As it is observable, on the right side of the window in the *Project* tab, it can be seen the target board that is selected. By default, from the import it will be the LCP1768 so, to change, execute the steps on Figure A.5. When desired, to see the source code output, *Build* the object code, send it to the board and run as mentioned in the previous section, to see the performance of the project.

Annex B

Considering the necessity and the advantages associated to the use of programming project documentation, the executable Doxygen is the main program that analyses commented lines of code and generates the documentation. The documentation techniques are adaptable to different types of programming language that can be applied. Optionally, the executable Doxywizard can be used as a graphical front-end for configuring and running Doxygen, to edit the configuration file. Given this, it is granted the possibility of review the code and preform maintenance to the programming project.

The documentation generation can easily be done by reading the especial commented lines in the source code files. In general, the generic syntax of comments used by this tool, to perform documentation, is to label comments with two asterisks. However, it offers an alternative approach for the documentation of some parameters. In the following sections, it will be listed the main ones that need to be documented, either for automatic generation of external documentation, as for to facilitate teamwork.

To obtain the installation file of this software, access to the *Download* tab on the left of the web page referred to in [21]. Do not forget to consider the OS available. The next step is to know how to comment on the source code lines, i.e., what are the commentary types used for generate documentation. Here, what is important for which type of programming language is being applied. To do that, access to the *Documenting the code* and *Special comment blocks* tab from the *Documentation Manual* tab on the left of the web page on reference [21]. To document the project, this user guide has everything to do it in a simple and effective way. An example of how to document source code in C language is showed on Figure D.1.

```
/*! \file structcmd.h
    \brief A Documented file.

    Details.
*/

/*! \def MAX(a,b)
    \brief A macro that returns the maximum of \a a and \a b.

    Details.
*/

/*! \var typedef unsigned int UINT32
    \brief A type definition for a .

    Details.
*/

/*! \var int errno
    \brief Contains the last error code.

    \warning Not thread safe!
*/

/*! \fn int open(const char *pathname,int flags)
    \brief Opens a file descriptor.

    \param pathname The name of the descriptor.
    \param flags Opening flags.
*/

/*! \fn int close(int fd)
    \brief Closes the file descriptor \a fd.
    \param fd The descriptor to close.
*/

/*! \fn size_t write(int fd,const char *buf, size_t count)
    \brief Writes \a count bytes from \a buf to the filedescriptor \a fd.
    \param fd The descriptor to write to.
    \param buf The data buffer to write.
    \param count The number of bytes to write.
*/

/*! \fn int read(int fd,char *buf,size_t count)
    \brief Read bytes from a file descriptor.
    \param fd The descriptor to read from.
    \param buf The buffer to read into.
    \param count The number of bytes to read.
*/

#define MAX(a,b) (((a)>(b))?(a):(b))
typedef unsigned int UINT32;
int errno;
int open(const char *,int);
int close(int);
size_t write(int,const char *, size_t);
int read(int,char *,size_t);
```

Figure B.1. Example of how to document source code in C language. Withdrawn from [36].

That said the development projects comprise variables that need to be thoroughly monitored and analysed within the program. After commenting the project according to the programming language used, the next and final step is the generation of updated documentation files. To the corresponding HTML documentation that is generated by Doxygen, accesses reference [37]. Then, to archive an output like in [37], learn how to do it in the OS using the *Doxygen usage* and *Doxywizard usage* from the same manual. The Doxywizard is a graphical front-end for configuring and running Doxygen. When started, it will display the main window and the look depends on the OS used so, if the intention is to use this tool in Windows, it will probably look different from the one on [38]. However, to get the output in [37], just follow the steps as shown on [38].

This type of documentation contains all relevant information that is important for the development of the project and, that can be read and interpreted by any user. Once finalized the project, the generated documentation will serve as a checklist to identify possible errors or even critical failures within the project. Still serve as a basis for implementing the manual, using it is way much simpler to create the manual and thus concludes it. There are other things that can be done to personalize the documentation. For example, apply Custom commands and/or Special commands on the project source code. About it, and some other things, the advanced knowledge can be obtained by studying and applying the information contained in the other user's guide manual tabs.

Annex C

When referring to the configuration and handling software, the control version becomes an important concept. This term defines by itself the management of all types of files, whether documentation or the source code. In short, it is a great advantage is the possibility of editing the same file by multiple users, as well as all the advantages associated with the recovery of previous versions. However, as usual, this concept also presents some disadvantages, like the necessity of using a server to allow access to files by multiple people. Here a tool will be presented to perform version control, Tortoise SVN, and a server, Visual SVN, to allow sharing information among multiple users.

The Version Control Systems (VCS) are most commonly run as stand-alone applications, however, the control version is also embedded in several types of software such as word processors and spreadsheets for content management systems. Each version is a revision of the previous version. The word processors are an electronic device or computer software application that perform composition, editing, formatting and printing of documents. Whereas the spreadsheets are an interactive computer application program for organization, developed as computerized simulations of paper accounting worksheets to perform both analysis and data storage. They allow not only the track of edition reviews, but also the ability to recuperate previous versions of the document revision. Given this, the control version tools are the essential software for the organization of multi-developer projects.

The present attachment pretends to serve as a tutorial for new users of the respective software stated below: TortoiseSVN and VisualSVN Server. Please note that for them, there were made the respective downloads compatible with the Ultimate version of Windows 7 whose Operating System (OS) is 64-bits. Thus, alert that in the case of this trial replication, take into account that the Windows or Linux edition has its availability and its respective operating system. For more information for advanced users, search it on the user guide available for each of them:

- To TortoiseSVN: through the *Support/Docs* tab from the reference [39] where are all available supporting documentation in several languages;
- To VisualSVN server: through the *SUPPORT* tab from the reference [40] where are available several articles dealing with different issues.

I. Getting Started

First of all, proceed to the installation of both software bearing in mind the information initially provided, they can be carried as follows:

- To TortoiseSVN: through the *Downloads* tab from the reference [39] where are all available versions and their respective language packages. Note that for this document, it was installed the 1.8.10.26129 (64-bits) version;
- To VisualSVN server: through the *download* tab from the reference [40] where are available the latest version of the software as well as previous versions. Note that for this document, it was installed the 3.3.1 (64-bits) version.

In case of problems occurrence during the installation, move forward to the next step. For any question or problem, contact the technical support available in each of the references suggesting that, in case of answer absence, search a solution to the problem in forums or video tutorials available in the YouTube channels. To help the users of the present

attachment, it was made two distinct sections: one for submitting a basic example of the TortoiseSVN application; and another one, for a brief explanation of how to apply a server to allow the access of multiple users to the project, using the functions provided by the VisualSVN Server.

II. TortoiseSVN

Attending that when there are multiple users accessing a file at the same time to checking or changing, when submitting the new version. In order to prevent their occurrence there are two different concepts: the repository and the working copy. The repository is a folder located on a board controlled by TortoiseSVN, which contains all final versions of the project as well as their respective documented history. The term "final version" is all the versions one is working on. Moreover, the working copy is a temporary copy of the project on which a user can work only. Once all checks and changes are completed, the next step is the submission of the new version of the design to the repository. If there is more than one user working on the same version of the code, the intersection of all the versions submitted will be made, automatically, creating a single version with all reviews, changes and corrections.

Following up on the previous question, to create a repository one must select a board and access to the *Create repository here* option and it will appear a window to confirm the repository creation as Figure C.1.

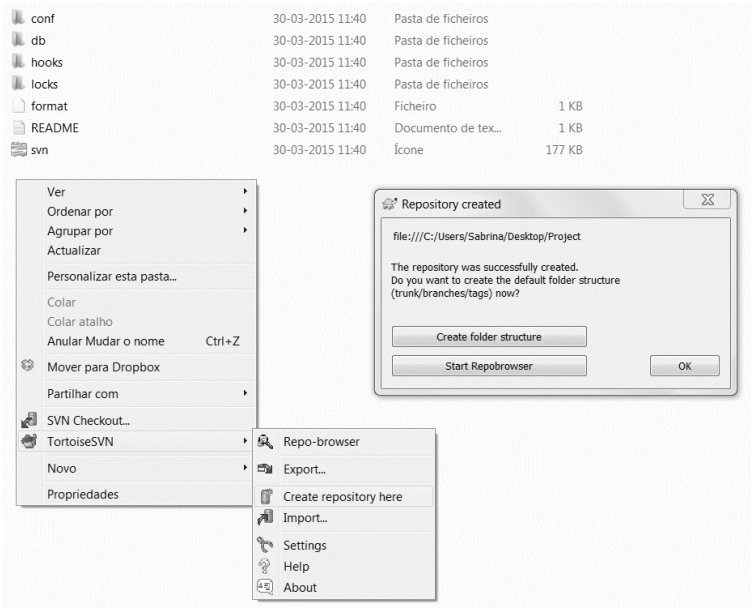


Figure C.1. Creating a new repository, in a directory, demonstration.

Once the creation of the repository has ended, the next step is to import a project. Assuming the project has already a source code available for download, go to the *Import*... option from

the same menu. In the window that appears consecutively, confirm the URL of the repository and write a message which shall include all the information deemed relevant for future scans or revisions. Once everything is in accordance, by selecting the *OK* button it should appear a window. Where, it is possible to view the import of the files in detail, such as shown on Figure C.2.

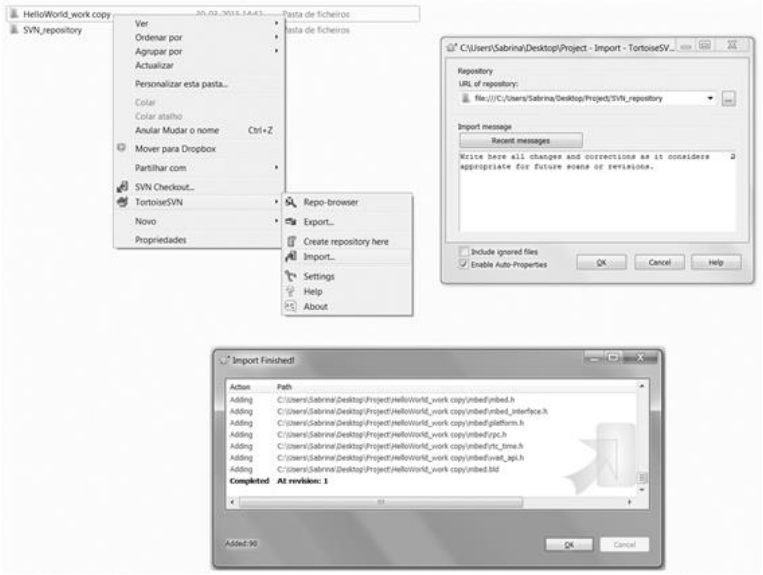


Figure C.2. Importation of files, from a project, demonstration

The next step is to confirm that the import was done correctly. For this, choose the menu *Repo-browser* option, confirming the URL, and then will be generated a window where it is possible to check whether all the contents were or not correctly imported. That said, whenever is wished to make changes to the files of the contents, just call the function *SVN Checkout*... to create a copy of the files with source code as well as all respective information versions (*.svn* files). Case, for some reason, it is necessary to have a copy of the files with source code without the *.svn* files opt for *Export*... option from the menu. Whereas the user opted *SVN Checkout*... function and to make some changes to the source code of a working copy, for example, as shown on Figure C.3.

```
#include "mbed.h"

DigitalOut myled(LED3);

int main() {
    while(1) {
        myled = 1;
        wait(0.2);
        myled = 0;
        wait(0.2);
    }
}
```

```
#include "mbed.h"

DigitalOut myled(LED3);

int main() {
    while(1) {
        myled = 1;
        wait(0.5);
        myled = 0;
        wait(0.5);
    }
}
```

Figure C.3. Changing data log.

To submit the preformed changes, use the *SVN Commit…* function and a window will be generated which contains the list of changed files and the possibility of adding a message with any information one considers relevant, similar to what is done when importing files to the repository, obtaining the following confirmation as showed on Figure C.4.

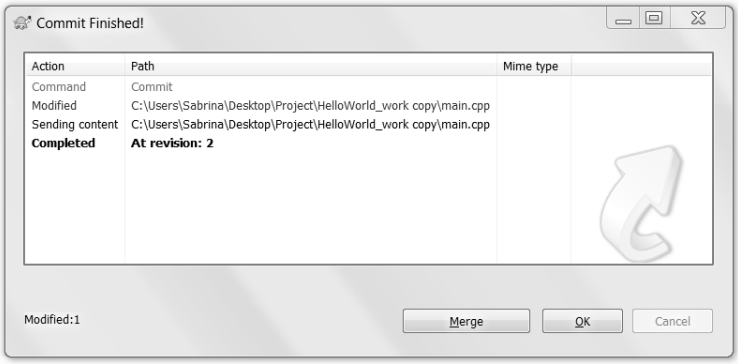


Figure C.4. Commit confirmation window.

Again, to confirm that the changes were properly submitted, refer to the *Repo-browser* option from the submenu. If it is necessary to add any files to the repository, since this created in the working copy, use the *Add*... option from the submenu and follow the same line of reasoning. The great advantage of using this software is the ability to gain access to the project change history. To this end it is applied the Show log submenu option to generate a window with that actions list. Duly dated and commented by his author, it will be made a copy of the working files and then, imported or submitted to the repository, as can be observed on Figure C.5.

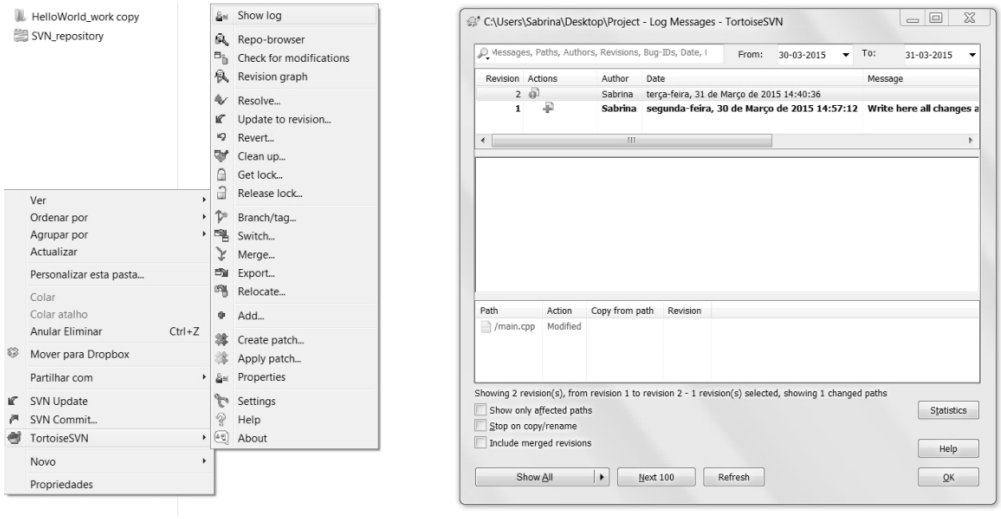


Figure C.5. Verification of the revisions and changes list.

III. VisualSVN Server

Now that an effective method to make the file version control and the revisions list view and made changes has been introduced and presented, the next step is to understand how can be allowed multiple users to perform the same operations using a server installed on a computer. Let's get started by downloading the setup file, compatible with OS available, from reference [40]. The performance of the contract is very simple but, it is suggest to choose the VisualSVN Server installation and Management Console. Do not forget to add the Subversion command-line tools to the path environment variable. Once more, for any question or problem contact the technical support available in each references, suggesting that if do not get the answer with the desired solution to the problem, search in forums or tutorials videos available in YouTube channels. The completed installation of the standard version is necessary to start an immediate interaction and–the first step is to create a user, as shown on Figure C.6.

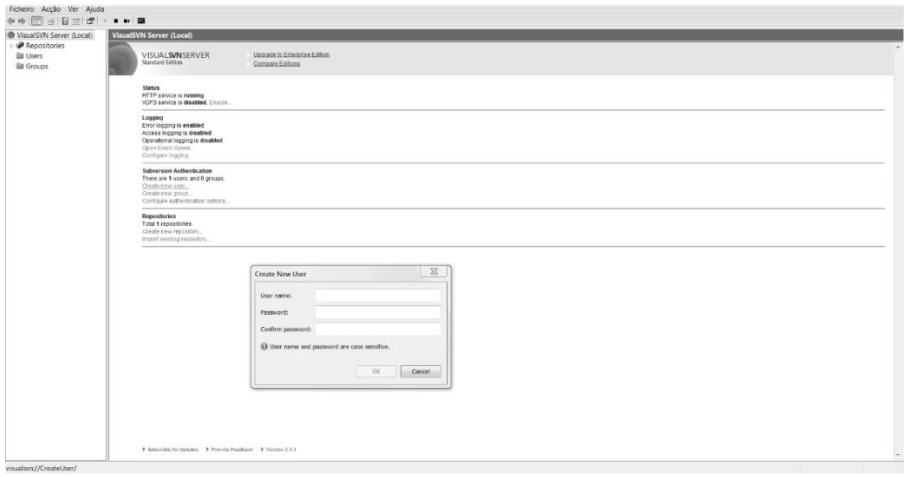


Figure C.6. One way to create a new user demonstration.

Being also possible through the Users tab on the right side of the window, as can be seen from Figure C.7.

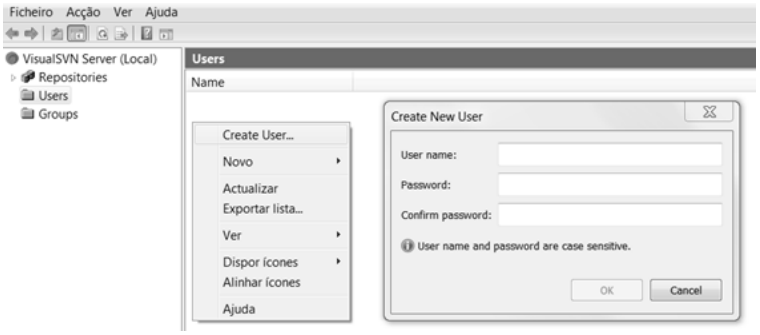


Figure C.7. Another way to create a new user demonstration.

Then create what will be a repository in the *Repositories* tab similar to what is done in the previous step. Additionally, select the type of repository, name it and set privacy and structure. Ending this task, as can be seen in the upper part of the store window it is necessary to use the same URL. When is desired to import some information to the repository, perform it by the same way as the importation of a project with TortoiseSVN but the target board must be the *trunk* server folder. To do this, connect to the server via the URL even when the project is being imported with TortoiseSVN, i.e. the repository URL should be the URL of the *trunk* server folder. From here, all the other features are processed similarly. Although in theory it is possible to put a repository on a shared network and have multiple users accessing it, this is most definitely not recommended. In fact, it is strongly discouraged and such use is not supported by given two reasons:

a) It can be defined so that is consented to the interaction of multiple users. Allowing them to check and change the contents of the repository, which can lead to accidental errors, such as unintentional deletion of files;
b) Since this is shared on a network, there may be access attempts by external users to design leading to possible data corruption.

For these reasons, it is highly recommend that the user(s) study and reflect, beforehand, about the necessity of resort to using a server.

Annex

D

Responding to the user needs, it was developed an efficient and effortless software that allows creating and editing waveforms. Modifying and transferring sign shapes, the execution of DUT level tests it is made feasible. This also allows the importation of sign shapes from the ArbExpress software directly to the both instrumentation of Tektronix AWG and AFG. The signals transferred from the software to the hardware are automatically converted to be within the allowed range. Then, the contents of screen interface and basic operating producers, will be presented.

The ArbExprees is a Windows-based software tool for creating and editing waveforms for both Tektronix AWG and AFG instruments. With this software, users are able to quickly and conveniently create desired sign shapes and send them to the generator. With low system requirements, it can easily be applied the Tektronix ArbExpress – AXW100, on the OS considering that, this software is supported by all windows versions from Windows XP up to Windows 7 (32-bits only). For the present quick getting started guide, the software version applied was the 3.1.2014.01.20 and the OS was the Ultimate version of the Windows 7 (64-bits). Please note that it can be preformed the download of this software directly from the reference [22], and it is and it is advisable to install all features and tools. Starting by the creation of a waveform, there are four different ways available in the *File* menu:

- Starting from scratch, using sing the *Blank sheet*;
- Starting from a *Standard Waveform*, using vertical and horizontal manipulation;
- Starting from scratch and applying standard waveform knowledge, by using *Equation editor* or *Waveform Math* tool;
- Starting from *open* a *.csv* file.

Here, it will only be mentioned the last one but, for more information about the other options or for advanced users, refer to the available *LIBRARY* tab on the reference [22] or to any other ArbExpress online help.

Considering that the desired signal has been created on the Microsoft Excel, make sure that the data is disposed in the first column, as shown on Figure D.1 for the 2013 version of the program.

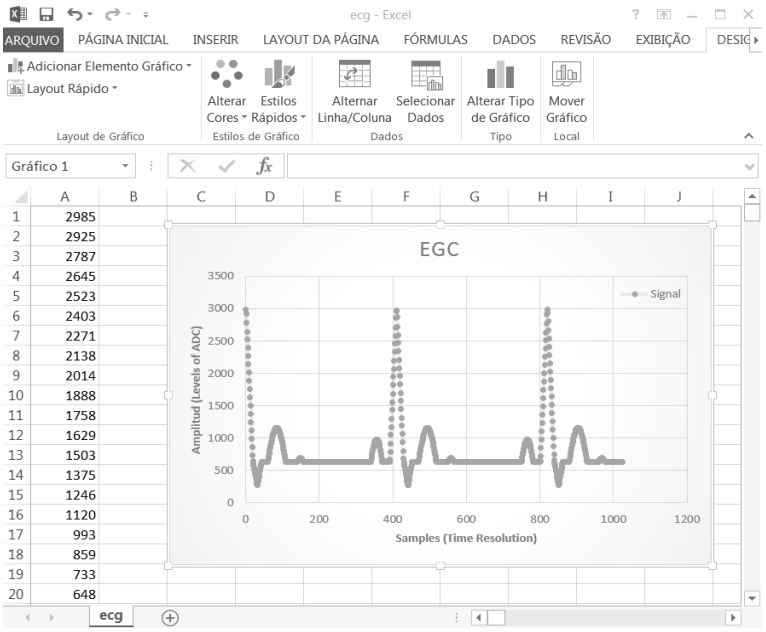


Figure D.1. Signal creation on a Microsoft Office Excel file.

When saving, define a name and a directory, and select the *.csv* type of file. Next, on the ArbExpress software, open the created *.csv* file through the *open* option from the *File* menu. It will appear a window to select a file format, like Figure D.2.

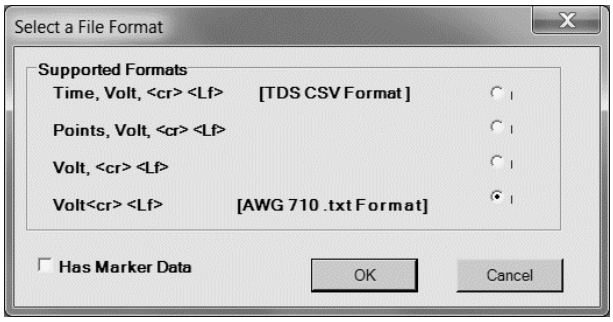


Figure D.2. Selecting a file format.

Select the last option and continue. A window will pop up with a preview of the created signal, as in Figure D.3.

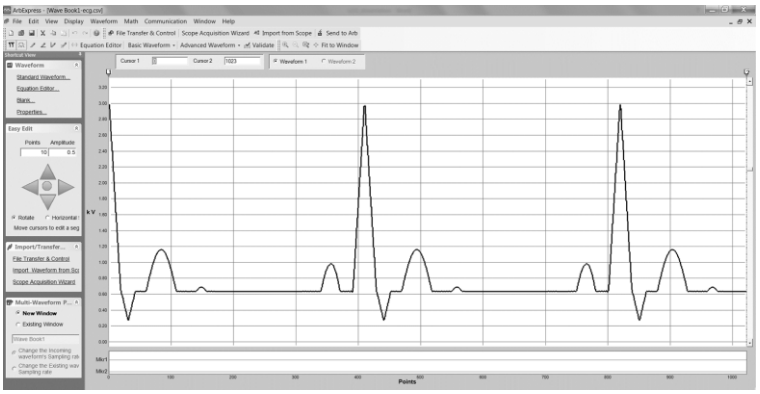


Figure D.3. ArbExpress signal created layout.

The next step is to transfer the created signal to the generator. This can be done by two different ways:

- Remotely connect the software to the instrument, through the *Communication* menu, by selecting *AWG/AFG File Transfer Control…* option to display a dialog box were the connection to the desire instrument is granted by select him form the Arb List;
- Save on a USB flash drive memory as a *.tfw* type of file and load the waveform in the arbitrary function generator.

Once that the created waveform is on the generator, just set the remaining desired parameters and the signal is automatically converted in order to fit the set limits.