UNIVERSITÀ DEGLI
STUDI DI SALERNO

Facoltà di Ingegneria
Dipartimento di Ingegneria dell'Informazione, Ingegneria Elettrica e Matematica Applicata

Dottorato di Ricerca in Ingegneria dell'Informazione
XIV Ciclo – Nuova Serie

TESI DI DOTTORATO

Innovative Algorithms and Data Structures for Signal Treatment applied to ISO/IEC/IEEE 21451 Smart Transducers

CANDIDATO: **FRANCESCO ABATE**

COORDINATORE: **PROF. MAURIZIO LONGO**

TUTOR: **PROF. VINCENZO PACIELLO**

Anno Accademico 2014 – 2015

Summary

Introduction

Technologies and, in particular sensors, permeate more and more application sectors. From energy management, to the factories one, to houses, environments, infrastructure, and building monitoring, to healthcare and traceability systems, sensors are more and more widespread in our daily life. In the growing context of the Internet of Things capabilities to acquire magnitudes of interest, to elaborate and to communicate data is required to these technologies. These capabilities of acquisition, elaboration, and communication can be integrated on a unique device, a *smart sensor*, which integrates the sensible element with a simple programmable logic device, capable of managing elaboration and communication.

An efficient implementation of communication is required to these technologies, in order to better exploit the available bandwidth, minimizing energy consumption. Moreover, these devices have to be easily interchangeable (*plug and play*) in such a way that they could be easily usable.

Nowadays, *smart sensors* available on the market reveal several problems such as programming complexity, for which depth knowledge is required, and limited software porting capability.

The family of standards IEEE 1451 is written with the aim to define a set of common communication interfaces. These documents come from the *Institute of Electric and Electronic Engineers* (**IEEE**) with the aim to create a standard interface which allows devices interoperability produced from different manufacturers, but it is not concerned with problems related to bandwidth, management, elaboration and programming. For this family of standards, now under review, it is expected a further development, with the aim to renew applicable standards, and to add new layers of standardization.

The draft of the **ISO/IEC/IEEE 21451.001** proposes to solve problems related to the bandwidth and the elaboration management, relocating a part of processing in the point of acquisition, taking advantage of elaboration capabilities of smart sensors. This proposal is based on a ***Real Time Segmentation and Labeling*** algorithm, a new sampling technique, which allows to reduce the high number of samples to be transferred, with the same information content. This algorithm returns a data structure, according to which the draft expects two elaboration layers: a first layer, in order to elaborate basic information of the signal processing, and a second layer, for more complex elaboration.

In this thesis is presented this integration of the IEEE 1451 family of standard, and it is organized as follows:

- in the first chapter the state of art and outlook development is shown, in the context of the *Internet of Things*;
- in the second chapter the actual smart sensor family of standards is shown, and their own problems are underlined;
- in the third chapter the proposed algorithm for the standardization is shown, with the related signal processing algorithms of the first and the second layer;
- in the fourth chapter **RTSAL** algorithm characteristics are analyzed, and a feasible implementation of the first layer algorithms on a microcontroller is shown;
- in the fifth chapter **RTSAL** algorithm results are shown, used for period and mean computation applications, and an application of this technique for the analysis of vibrations signals from faulty bearing.

Chapter 1

Networking Measurement for an Interconnected World

Everyday information demand increases, many data of all kinds are available in many different fields. The Internet, indeed, allowed to create a powerful informative network, by means of which more and more services are spread: from information to communication, from banking services to goods purchasing.

Nowadays, the number of devices connected to the Internet overtakes the world's human population, and the forecast says this tendency is growing [1]. As for sensor networks, more and more devices and machines are interconnected, in order to acquire and exchange information and to take decisions. In this sense, the so-called concept of the Internet of Things is growing.

1.1 The Internet of Things

1.1.1 An overview

The term "Internet of Things" was coined by Kevin Ashton in 1999 [2] and it meant a network which allows communication among machine (M2M: machine to machine communication). Now the concept of the Internet of Things is different from just the M2M communication: the machine to machine communication is focused on connecting machines, and it could be well implemented mainly with proprietary closed systems, whereas, the Internet of Things is about the way humans and machines connect, using common public services [3]. More in depth, with this kind of network, a way to digitize the physical world will be implemented, in order to interact with near or far objects, creating services and application in any branch. Moreover, it has the potential to radically shift the way people interact with their surroundings. The abilities to monitor and to manage with objects in the physical world, and to interact with them by means of an electronic interface, makes possible to bring data-driven decision making to new field of applications of human activity, optimizing the performance of systems and processes, saving time for people and business, and improving the quality of life.

The term “Things” could refer to a wide variety of physical smart objects; in other words, an object would be able to acquire data from the environment, process and communicate them to other objects of the same environment: those things have to be embedded with electronics, software, sensors and communication capabilities. As the Internet of Things evolves, existing networks, and many others, will be connected with added security, analytics, and management capabilities (Fig. 1.1). This will allow the Internet of Things to become even more powerful in what people can achieve through it.

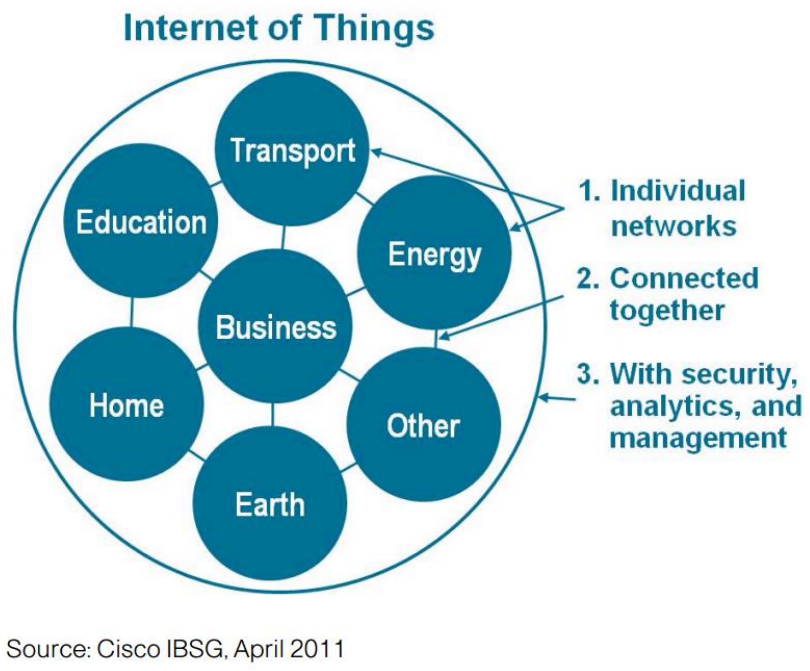


Fig. 1.1 IoT can be viewed as a network of networks.

Some examples of existing devices are the RFID transponder on farm animals or for stocks management to implement traceability strategies and to acquire useful data, or vehicles with built-in sensors. In the future, the Internet of Things systems could also be responsible for performing actions: several important research fields now are about:

- **Energy management**, like the *Advanced Metering Infrastructure* (AMI), thought to integrate sensing and actuation systems for a smart grid system, able to communicate with the supply or distributor company in order to balance power generation and energy usage [4].
- **Factories and Worksites management**, in order to improve productivity and safety, optimizing the equipment use and inventory management, operating on the efficiencies, predicting maintenance, etc.…
- **Domotics** or, more broadly, in building automation, to directly control and regulate lighting, heating, ventilating and air conditioning (*HVAC*) or home security devices, depending on sensing of the home environment, or a remote commanding [5 - 7].
- **Road safety** and auxiliary services for drivers and traffic management, sensing the road with different kind of sensors, integrated in the asphalt road or in the guardrail [8], and giving important information about an accident or traffic presence. Also by means of micro systems inside moving vehicles (including cars, trucks, ships, aircraft, and trains) to monitor conditions which could require maintenance, or pre-sale analysis.
- **Environmental sensing and monitoring** to assist in environmental protection, by keeping under observation air [9] or water quality, atmospheric or soil condition [10].
- **Infrastructure and building applications** to monitor and control operations of urban and rural infrastructures like bridges, railway tracks, on and off-shore wind farms, and buildings [11]. Those applications would be useful to monitor critical structural conditions, or to scheduling repair and maintenance activities, communicating the acquired data and taking decisions.

- **Healthcare** systems to remote monitoring customers health, by means of big machineries, smart sensors, wearable devices, and even ingestible devices, connected to the Internet of Things [12].
- **Traceability** systems to keep track of products during all the production chain, or to manage and organize stocks or archives [13, 14].

A recent study made by the McKinsey Global Institute predicts the huge development of the market with all the activity related to create and to implement this “Internet of Things”: to realize the full potential from possible IoT applications, technology will need to continue to evolve, providing lower costs and more robust data models. In almost all organizations, taking advantage of the IoT opportunity will require to truly embrace data-driven decision making [15]. In this way, this study, analyzing more than 150 IoT use cases across the global economy, and using detailed bottom-up economic model, estimates the economic impact of these applications by the potential benefits they can generate (including productivity improvements, time savings, and improved asset utilization), as well as an approximate economic value for reduced disease, accidents and deaths. Applying those concepts to possible future applications in different environments, such as homes, offices and factories, and considering as the key insight of the IoT applications benefits the critical contribution made by interoperability among different IoT systems (“*On average, interoperability is necessary to create 40 percent of the potential value that can be generated by the Internet of Things in various settings*”), the McKinsey global Institute estimates the total potential economic impact included from $3.9 trillion to $11.1 trillion per year in 2025.

1.1.2 Obstacles to be overcome

In this future scenery, expected to be really complex, there is much to implement to achieve this interconnected system. Achieving a huge level of impact on daily life and economics will require overcoming different technical and methodological boundaries.

For a successful implementation of a unique and global platform like the “Internet of Things” there are several essential and fundamental technical requirements:

- *Communication Protocols*: they allows data communication among electronic devices, following a procedure that, depending on the case, specifies more or fewer details of the ISO/OSI model (Fig. 1.2). There are hundreds of different protocols to transmit data, which differ in physical medium, frequencies, format and standard, at a different level of the ISO/OSI model.

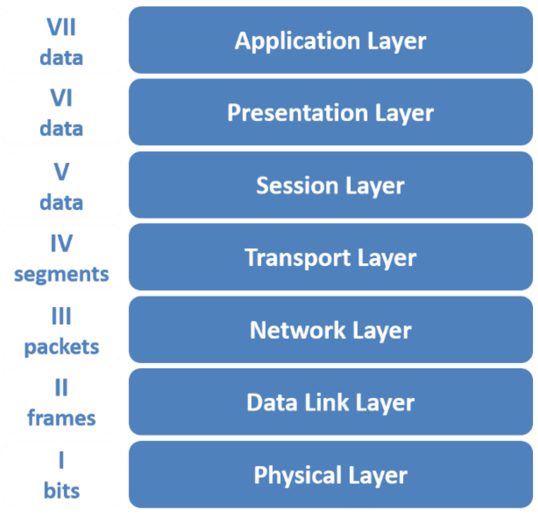


Fig. 1.2 ISO/OSI model.

The interoperability is one of the key feature for the future of the Internet of Things: the ability of devices and systems to work together is critical for realizing the full value of the IoT applications; without this property, several benefits figured out for those applications could not be implemented. Adopting open standards, or implementing system or platform that enable different IoT system to communicate with one another could be used in order to develop this kind of common network.

- *Bandwidth*: for an efficient and widespread diffusion of the Internet of Things, it will be necessary to guarantee management of an enormous amount of data. So, appropriate networks and infrastructures, in terms of bandwidth, or efficient management of information and bandwidth would be necessary.

The *addressability* is necessary to univocally identify objects, devices or machineries, into the network. Several solutions are feasible like, for an example, EPC (*Electronic Product Code*), which is a universal identifier that provides a unique identity for every physical object anywhere in the world, already implemented for the RFID technology (it could be written in the memory of the transponder) [16, 17]. The next generation of devices connected to the internet using protocol version 6 (IPv6) would be able to communicate with devices attached to virtual or physical objects, because of the extremely large address space of the IPv6. The Internet Protocol version 4, IPv4, would not be able to address the big amount of devices expected for the Internet of Things. A combination of these ideas can be found in the current EPC Information Services (EPCIS) specifications [18].

The *physical* or *geographic location* of things will be critical. Now, the internet processes mainly information, but in the Internet of Things, the position of a sensor would be important, to localize other sensors, gateway or concentrator, in order to handle neighbor operations.

Security and *privacy* will be handled, because of the big amount of data collected and managed by the Internet of Things. "User consent" and "anonymity" are some examples of how it complex could be to acquire and handle different kind of data [19]. The types, amount, and specificity of data gathered by billions of devices will create concerns among individuals about their privacy and among organizations about the confidentiality and integrity of their data. Providers of IoT devices or services will have to ensure transparency of data usage and protection of personal and private data. A complex system like the Internet of Things have to guarantee security: it is a network where all sensors and the actuators are interconnected, so, the case in which unauthorized users could exploit those things remotely must be prevent. The Internet of Things could introduce new categories of risks, because it extend the information technology systems to many devices, which could be considered as potential breaches; it will manage and control physical assets: consequences associated with a breach in security overtake the simple needs of sensitive data, because they could potentially cause physical harm.

Moreover, for a widespread adoption of the Internet of Things, many other barriers have to be overcome, such as *technology*, *intellectual property*, and *organization*.

The cost of basic *technology* must continue to decrease. Low cost and low power sensors are essential. The price of MEMS (micro-electromechanical systems), which are already widely used for smartphones, has dropped by 30 to 70 percent in the last five years; a similar trajectory is needed for radio-frequency identification (RFID) tags, and other possible technology solutions to make tracking and identification practical for low-value items. Moreover, progress for low-cost battery power is also needed to keep distributed sensors and active tags operating. In almost all applications, low cost data communication links are essential, and the cost of computing and storage must also continue to decrease, in addition to a general development of analysis and visualization software.

The *intellectual property* of this huge amount of data, and the related ownership rights, is an important topic, in order to unlock the full potential of IoT. Data will be acquired from sensors, produced by a manufacturing company, will be a part in a solution, projected by another company, in a setting owned by a third party: who has the rights to the data has to be clarified.

The *organization* is critical: IoT combines the physical and digital worlds, changing conventional notions that we know about organization. Just now, information technologies application operate on the product, or managing the information about the product, in the retail shops or in stocks. In an IoT world, information technologies will be embedded in physical assets and inventory, and directly affects the business metrics against which the operations are measured, so these functions will have to be much more closely aligned. Furthermore, companies need the capability and mindset to use the Internet of Things to guide data-driven decision-making, as well as the ability to adapt their organizations to new processes and business models.

1.2 Smart Sensors

One of the technological development fields of our time, in the field of sensor networks or, in a wider and futuristic view, in the field of the Internet of Things, is the field of *smart sensors*.

"S*mart Sensor*" means a complex device, equipped with several function blocks: there is not only the sensing element, but also electronics and interfaces that handle signal conditioning, analog to digital conversion, and communication, integrating everything on a single device, and returning, to an application level, the digital version of the acquired signal. The term "*smart*" is added for those kinds of sensors equipped with an "intelligent" device, like microcontroller units (MCU), digital signal processors (DSP), and application-specific integrated circuits (ASIC) [20]. Modern technology advancement is allowing production of ever smaller, low power consuming and accurate devices.

Therefore, smart sensors must have a sensible element, from which would be possible to obtain an electric signal, proportional to the physical magnitude to be measured (temperature, pressure, flow, etc...). Feasible technologies are many, from traditional sensors, to the modern MEMS (micro electromechanical system [21]), created using semiconductor manufacturing processes. Smart sensors, including also the conditioning electronic, for the attenuation or amplification of the sensor output signal, and the electronic devices needed to the analog to digital conversion, modify a classical acquisition system structure, in which even the electronic dedicated to these kinds of operations is relocated in the point of acquisition (Fig. 1.3). Consequently, all the acquisition points, where a smart sensor could be used, has to be connected to a power source: typical supply voltages are 5V or 3.3V and even lower voltages. For several systems, demand of different supply voltages poses challenges that are not typically associated with sensors, and adds complexity to the system and sensor. In addiction, energy efficiency is an important task to manage for every smart sensors, and, in particular, for wireless ones. In fact, transmission is the most power consuming task among all of a wireless sensor. Several approaches propose protocols that are developed to provide energy saving [22-23].

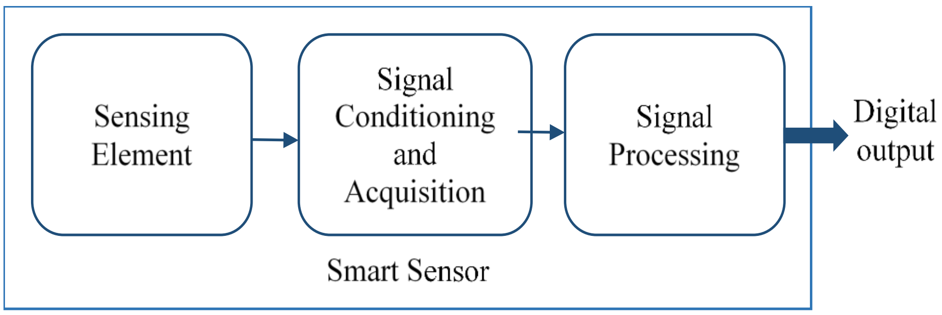


Fig. 1.3 Smart Sensor block diagram.

To use a unique device, which acquires a physical magnitude and communicates acquired data in digital format to the application level is an innovative idea, which will allow several advantages in the future:

- means to remote control of sensor nodes, equipped with a programmable logic;
- simplification of sensors replacement in complex systems: wanting to replace a sensor with another which acquires the same physical magnitude, problems could be manifold, such as the operating voltage range, the power absorption, the transfer function, etc.… With a smart sensor, all of these operations would be simplified, without requiring modifications of the acquisition system [24].

For a further development of smart sensors, in order to have a better interoperability and the employment of these kind of sensors in a complex system as the Internet of Things, reference standards are essential, to make sure that smart sensors would allow communication on different level, following common reference rules.

1.3 An example: CAN bus protocol

As an example, a communication protocol is reported, already popular in the automotive environment: the CAN-Bus (Controller Area Network). This communication protocol, which at first was proprietary, has become an international official standard afterwards (ISO standard [25]) allows to implement a micro network within the car or, generally, within an industrial process or a production line. For this type of network, each node is equipped with a programmable logic device (usually microcontrollers) which acquires and manages information of interest, and it sends it, thus implementing the concerned protocol.

This protocol was created by *Robert Bosch GmbH* during the first part of the 80s, for automotive applications, and, as time went by, for reliability features and characteristics, it has increasingly established itself as protocol for industrial automotive field buses and across industries (aerospace, naval, and railway).

CAN-Bus [25] has allowed to radically modify the project philosophy of the control nodes in all of the systems where it is applied. It differs from most of the existing control systems until then: those systems were composed by a central elaboration system, with high computational capabilities, and were interconnected with other sensors and actuators. Usually, the central system had the task to monitor every sensors, and the task to elaborate all the control strategies. Therefore, the grid required a complex wiring system distributed on the process, which required significant modification to the system for each change (substitution or addition of other nodes).

In particular, in automotive field, tendency was to divide systems into closed smaller subsystems, which did not share any kind of data, so as to manage each subsystem with its own control system.

The basic idea behind the CAN-Bus is to distribute computational capabilities across the grid, among different nodes, and to keep in communication, in a unique network, all controllers, in order not to use the same element duplicated in different systems, because they are no longer isolated among each other. This idea has become technologically possible, thanks to the spread and the cost reduction of microcontrollers.

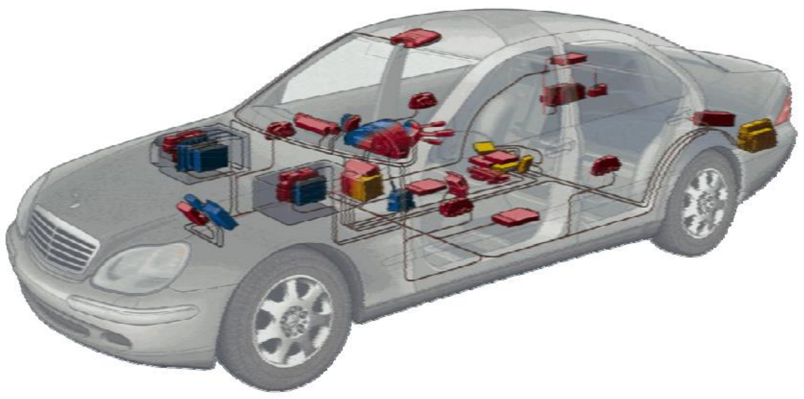


Fig. 1.4 Example of CAN network inside a car.

In fact, CAN Bus allows the communication among smart devices, sensors and actuators able to produce data independently, and then give them to the transmission channel. They perform different tasks: amplification of the signal given by the sensing element, translation of the signal in an appropriate range for the analog to digital conversion, elaboration of data, and emission of data on the channel. This is a serial bus protocol, and it implements a broadcast digital communication. It allows a distributed real-time control, with a very high level of reliability.

General acceptance of CAN protocol is due to noteworthy advantages that it offers:

- *Strict response time*: fundamental specification in process control. CAN technology expects many hardware and software instruments and development systems for high level protocols, which allow to connect a large number of devices, keeping strictly time constraints.
- *Simplicity and flexibility of wiring*: CAN is a serial bus typically implemented on a twisted pair (shielded or not depending on the requirements, Fig. 1.5). Nodes have no address which identify themselves, so they could be added or removed without reorganizing the whole system, or part of it.

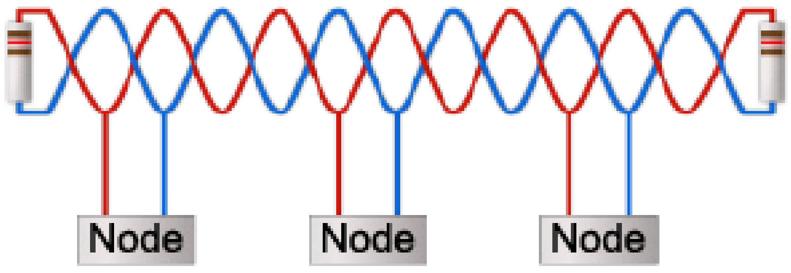


Fig. 1.5 A twisted-pair copper cable usually realizes the physical transmission.

- *Good noise rejection*: the ISO11898 standards recommend that interface devices could continue to communicate even under extreme conditions, like the interruption of one of the cables or a short circuit of one of them with ground or power source.
- *High reliability*: error detections and retransmission requests are managed directly by the hardware with 5 different methods (two of them on bit layer, while the other three on message layer)
- *Error isolation*: each node is able to detect its own fault and to exclude itself from the communication bus if it persists. This is one of the working principles that allows this technology to maintain the timing constraints, preventing that a single node could undermine the whole system.
- *Ripeness of the standard*: widespread of CAN protocol in the last twenty years has entailed the wide availability of rice-transmitter devices, a microcontroller that integrates CAN gates, development tools, beyond the substantial cost decrease of these systems. This is important to ensure that a standard asserts itself in the industrial field.

Essentially, on the vehicle systems it is common to choose a distributed system of sensors, instead of a centralized one, substituting complex wiring and redundant sensors systems, with a digital network inside the vehicle. Moreover, this network has become a reference standard, and more and more car industries use this communication protocol, instead of a proprietary protocol, as in the past.

ISO 11898 directives establish only the first two layers of the ISO/OSI model for communication protocols: physical layer and data link layer. The data link layer is divided by the standard into two sub layers: the Logical Link Control (LLC), which manages data exchanges, and the Media Access Control (MAC), which checks errors and manages the enclosure of data in the format required.

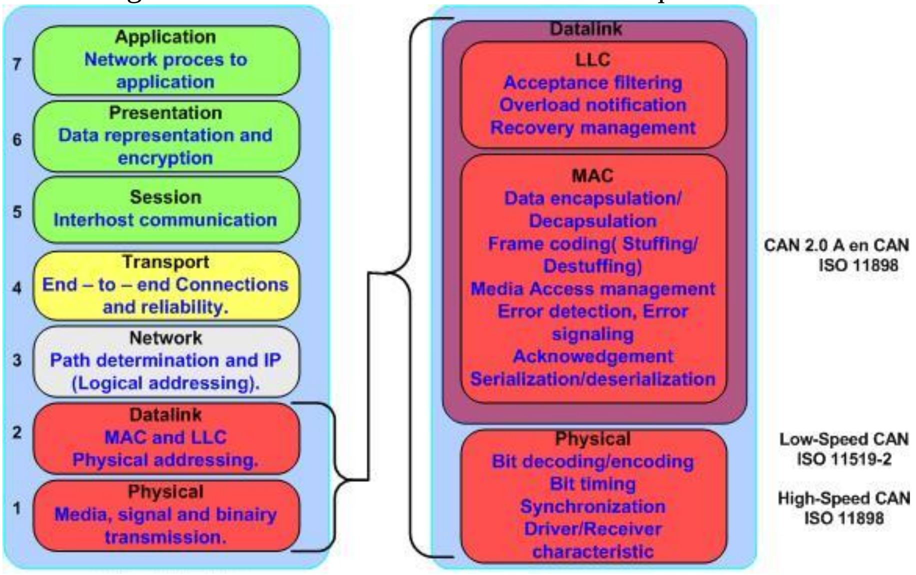

7 Application
Network proces to application
6 Presentation
Data representation and encryption
5 Session
Interhost communication
4 Transport
End – to – end Connections and reliability.
3 Network
Path determination and IP (Logical addressing).
2 Datalink
MAC and LLC
Physical addressing.
1 Physical
Media, signal and binairy transmission.
Datalink
LLC
Acceptance filtering
Overload notification
Recovery management
MAC
Data encapsulation/ Decapsulation
Frame coding(Stuffing/ Destuffing)
Media Access management
Error detection, Error signaling
Acknowedgement
Serialization/deserialization
Physical
Bit decoding/encoding
Bit timing
Synchronization
Driver/Receiver characteristic
CAN 2.0 A en CAN ISO 11898
Low-Speed CAN ISO 11519-2
High-Speed CAN ISO 11898

Fig. 1.6 ISO/OSI model for CAN-Bus.

Therefore, this protocol, already requiring a smart sensors network, does not manage data above the data link layer, then the format of information is not established, and the communication at an application layer is not generalized. Thus, it is needed to arrange an ad-hoc format for data, to manage the upper layer of the ISO/OSI model, and to conveniently program all the network nodes.

An additional level of standardization could allow to use this type of communication in an easier way, not requiring details, setting up a network, related to higher layers of the ISO/OSI model.

1.3.1 CANopen and CiA

CANopen is a CAN-based communication system, which comprises higher-layer protocols and profile specifications, developed by the CAN in Automation (CiA) [26], a non-profit international users' and manufacturers' group. The CANopen standard consists of an addressing scheme, several small communication protocols and an application layer defined by a device profile. The communication protocols have support for network management, device monitoring and communication between nodes, including a simple transport layer. The lower lever protocol implementing the data link and physical layers is usually Controller Area Network (CAN), according to ISO 11898-1 and ISO 11898-2.

Standardized CANopen devices and application profiles simplify the task of integrating a CANopen system, to achieve "plug-and-play" capability for CANopen devices in a CANopen network. This standard unburdens the developer from dealing with CAN hardware-specific details such as bit timing and acceptance filtering.

From a broader point of view, this is the purpose of the family of standard IEEE 1451, which proposes a unique platform for smart sensors, standardizing also the application layer. This would make smart sensors usable as plug and play devices, allowing an easier interfacing among them through the network.

Standards, protocols, procedures or platforms like these are building the technical and organizational foundation for the development of the Internet of Things.

Chapter 2

A Standard Approach for Smart Transducers

A network could connect many different sensors or actuators (in general transducers, which include both), by means of a digital wired or wireless implementation of these types of communication, network and transducers digital incompatible protocols are in use by sensor manufacturers and industries products. Some of those are standards, others are proprietary, and for this reason, an integration on a unique network, as expected for the Internet of Things technologies, is really complex.

Nowadays, on a distributed smart sensor network, smart nodes are network and transducer specific with manufacturer specific data and control models. Any sensor could speak a different language: an integration would require many conversion layers, different for each specific device.

Therefore, in order to manage communications among transducers in a common, easier, faster, cheaper and simpler way, an agreement is strongly needed. The adoption of well-defined and widely accepted standards for sensor communication and description is a preferable approach.

To solve these problems, the *Institute of Electrical and Electronic Engineers (***IEEE***) Instrumentation and Measurement Society's (***IMS***) Technical Committee on Sensor Technology* supported a series of projects, indicated with the abbreviation IEEE 1451, with the aim of developing standard software and hardware for smart transducers, and for their integration on a network. These projects would simplify communication capabilities, without restrictions on the electronic device to be used. A common standard could be referred to develop network independent and manufacturer independent transducer interfaces, reducing configuration steps for each different device [27].

2.1 A family of standards

The family of standards IEEE 1451 is edited with the aim of defining a set of common communication interfaces. So, this family of reference standards, establishes the sensor network implementation, in which those devices could communicate, regardless of the communication physical channel, of the specific characteristics of each device, of the communication protocol implemented by manufacturing companies. Purposes are different:

- Utilization of sensors in an open manner on the hardware and software specifications, without the need to install drivers, remove and update devices (*Plug-and-Play)*. A major problem with sensor networking applications is network configuration management. Any kind of network uses its own connections and protocols, and, in addition, configurations, nodes and connections could change while they operate. The concept of plug-and-play sensors addresses two main problems: a standardized electrical interface to the network, allowing a wide variety of sensor types to be used, and a self-identification protocol, allowing the network to configure dynamically and describe itself [28].
- Communication with different physical channels (both wired and wireless), due to the definition of various communication standards. "*Recognizing that no single sensor bus or network is likely to dominate in the foreseeable future, the IEEE 1451 set of standards was developed to unify the diverse standards and protocols by providing a base protocol that allows interoperability between sensor/actuator networks and buses. A key feature of the IEEE 1451.0 standard is that the data […] of all transducers are communicated on the Internet with the same format, independent of the sensor physical layer (wired or wireless)*" [29].
- Definition of the Transducer Interface Data Sheets (TEDS), which includes all the information related to codification, communication and control of transducer information and

acquired data. Therefore, TEDS contains the calibration and operating data necessary to create a calibrated result in standard SI units, additional information necessary to uniquely identify the Transducer Interface Module (**TIM**), and it provides supplemental information to the application.

- Integration capabilities for the sensible element with conditioning, acquiring, elaboration and communication electronic in a unique device, hence making easy to implement a sensor network with distributed intelligence, helping operations of sensor network management and, thus, the needed actions for replacement.

These documents are meant to be applied on new technology sensors, the so-called "*smart sensors*" (see paragraph 1.2). The term smart sensor was employed supposing a device that has not simply its own elaboration unit integrated, but it would be also easy to integrate in a network. IEEE 1451 transducers would have capabilities of self-identification, self-description, self-diagnosis, self-calibration, location awareness, time awareness, data processing, data fusion, alert notification, communication protocols, standard based data format. So, a smart transducer needs the integration of an analog or digital sensor or actuator element, a processing unit, and a communication interface. Based on this premise, a general smart transducer model has been already shown in Fig. 2.1.

IEEE 1451 smart transducer architecture differs from this general architecture because of the partition of the system into two major components: a Network Capable Application Processor (**NCAP**), a Transducer Interface Module (**TIM**) and a Transducer Independent Interface (**TII**) between the **NCAP** and the **TIM**.

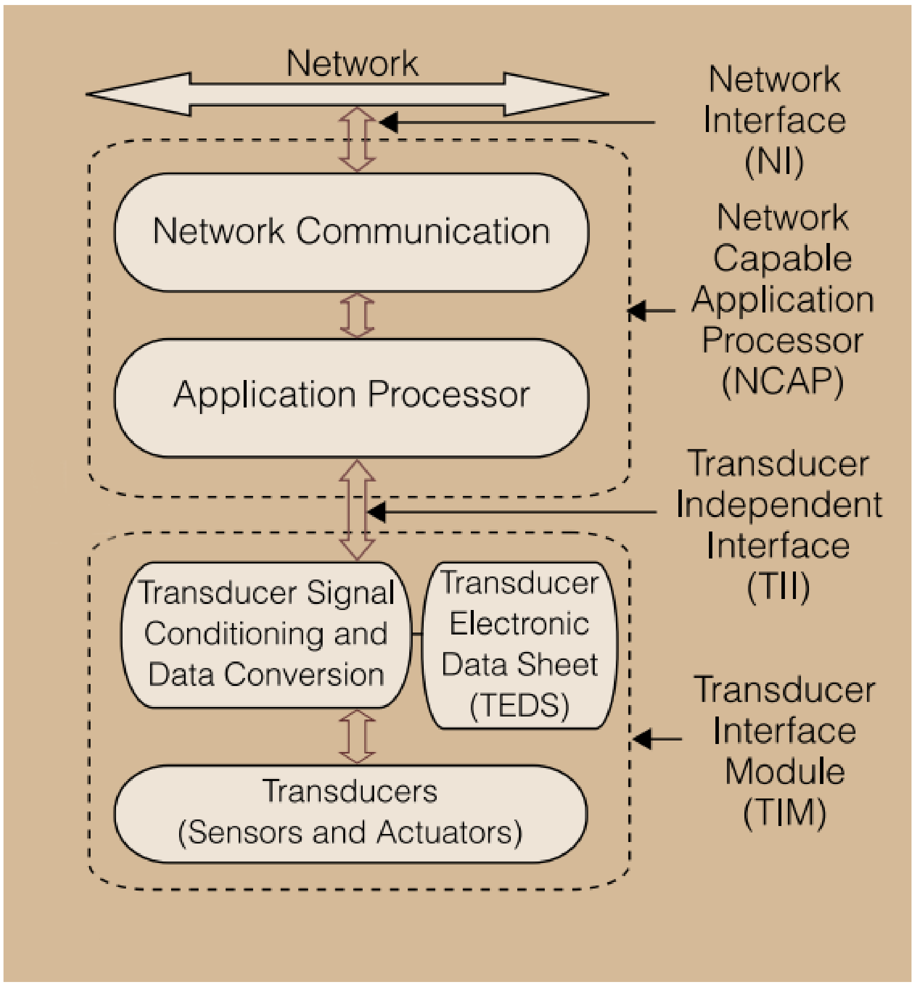

Network
Network Interface (NI)
Network Communication
Network Capable Application Processor (NCAP)
Application Processor
Transducer Independent Interface (TII)
Transducer Signal Conditioning and Data Conversion
Transducer Electronic Data Sheet (TEDS)
Transducer Interface Module (TIM)
Transducers (Sensors and Actuators)

Fig. 2.1 IEEE 1451 block diagram.

- **TIM** (Transducer Interface Module) is the module to which sensors and actuators management is entrusted, data conditioning, and their transmission to the elaboration system. This module is composed of several blocks, including sensors and actuators, microcontroller and the electronic dedicated to the conditioning system, acquisition and pre-elaboration of data, memory blocks used to store the transducer electronic data sheets, and the communication interface to manage the data transfer towards the NCAP.

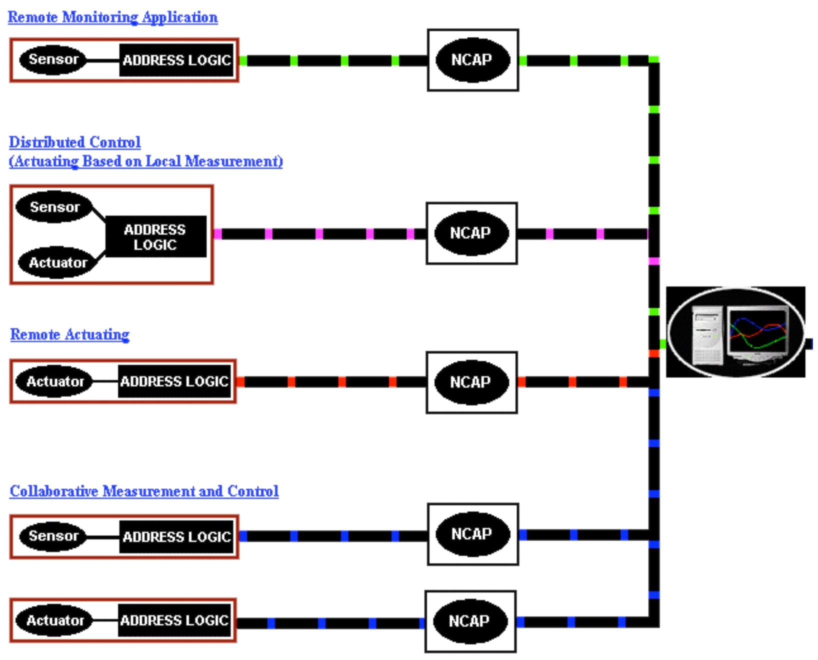


Fig. 2.2 IEEE 1451 architecture, NIST architecture for 1451 [30].

- **NCAP** (Network Capable Application Processor) is the device located, in this architecture, between the TIM and the network. It has the purpose to manage data to and from the TIM and the external network, and to make available data acquired by sensors for a network, on an application layer. Carrying out the task of a gateway in a sensor network, once received the electronic data sheet from the TIM, this device sets the maximum bit rate supported by communication module in transmission, how many channels it contains, and data format of each transducer. Moreover, depending on the operation mode of the smart sensor, the NCAP could start the acquisition, sending a trigger event to the TIM, and it could manage communication with an appropriate protocol, in order to manage errors as well (hardware errors, calibration errors, anomaly, etc...).

- **TTI** (Transducer Independent Interface) defines a communication medium and a protocol for transferring sensors information. This interface provides a set of operations, such as read, write, read and write message, read and write response, etc.

The other main difference from the general architecture, a key feature of an IEEE 1451 smart transducer is the specification of the standardized, Transducer Electronic Data Sheet (TEDS), and their formats. The TEDS attached to the transducer is like an identification card: it stores manufacturer related information for the transducer, such as manufacturer identification, measurement range, accuracy, calibration data and timing constraint, similar to a real datasheet content normally provided by the manufacturer.

2.1.1 Advantages

The main advantages that an IEEE 1451 compliant smart transducer offers, due to this architecture choice, and to the introduction of TEDS, are:

- Auto-identification capability: a sensor or actuator, equipped with the IEEE 1451 TEDS can identify and describe itself to the host or network, by sending the TEDS information.
- Self-documentation: the TEDS in the sensor can be updated and store information, such as the location of the sensor, recalibration date, or any type of information in a custom format.
- Automatic configuration of sensor parameter, like calibration curve (data stored in TEDS), reducing error due to entering sensor parameter by hands.
- "*Plug-and-Play*" sensors: this means ease installation, upgrade and maintenance of sensors. A TIM and a NCAP, based on the same standard version, are able to be connected with a standardized physical communication media and are able to operate without any change to the system software. In this way, different TIM from different manufacturers could "plug-and-

play” among them, and with different NCAP, with no need to reconfigure anything.

Moreover, this family of standards is comprehensive enough to cover nearly all sensors and actuators in use today; it has many operating modes (buffered, streaming, timestamp, grouped sensors, etc...) managing efficiently binary protocols. In addition, it is compatible with many wired and wireless sensor buses and networks.

2.2 Existing Standards

In this paragraph, a brief analysis of the document of the IEEE 1451 family of standard is shown.

2.2.1 IEEE 1451.0

The goal of IEEE 1451.0 is to achieve data level interoperability for the IEEE 1451 family. In this document [31], the network general structure expected by this architecture, services that TIMs and NCAPs have to provide, data types, functional specifications and commands are defined. Moreover, this document establishes common features and defines:

- transduction channel, with the classification of sensors, event sensors and actuators;
- data structure, used to store and to transmit acquired data from a sensor node, or to send to an actuator;
- sampling modes with which it acquires data from a sensor;
- transmission modes of data sets;
- trigger mechanisms and the needed strategies for synchronization;
- TEDS content and structure.

These functionalities are all physical communication media independent (1451.X).

2.2.1.1 TEDS – Transducer Electronic Data Sheet

Transducer electronic data sheets are blocks of information, preferably stored in non-volatile memory of the TIM, whose format is specified in the family of standard. In several applications, where it is not possible to store TEDS on the device, the standard allows storing TEDS in other memory location of the user's system. When TEDS are stored in another memory location, other than internal memory, they are called "*virtual*" TEDS: transducer manufacturer has to provide these virtual TEDS on an appropriate different media. Therefore, the user of the system in which the smart sensor will work is responsible for specifing the connection among the TIMs, identified by means of a universally unique identifier (UUID), and the location where virtual TEDS are stored. The NCAP has the task to manage this service, where it is necessary. The TIM will provide, answering to the query from the NCAP for electronic data sheets, a flag that indicates if these data sheets are supported, or if they are virtual. There are four mandatory fields for TEDS IEEE 1451.0 compliant, whereas all the others are optional.

Required **TEDS** are:

- **Meta-TEDS**: it gives critical timing parameters of the **TIM**, which are read by the **NCAP** to define time-out values, in order to understand when the **TIM** is not answering.
- **TransducerChannel TEDS**: it gives detailed sensor information in the TIM, about the physical measured or controlled unit, sensors or actuators working range, digital communication detail, operating mode, and timing, acquisition and communication information.
- **User's Transducer Name TEDS**: it stores the name by which the **TIM** could be identified in a network. The structure of this **TEDS** is simply suggested in the standard, but it could be defined from the user.
- **PHY TEDS**: this **TEDS** depends on the communication physical channel used to interconnect **TIM** and **NCAP**. It make available at the interface all of the information needed to gain access to any channel, plus the information common to all

channels. It is not described any further in this document of the family of standards.

Whereas, optional **TEDS** are:

- **Calibration TEDS**: it include calibration constants needed to convert sensor output into the appropriated measurement unit. Two methods are supported: one is linear, which uses the generic formulation $y = mx + b$, while the other one considers a completely generic formulation, which can support nearly all necessary characteristics.
- **Frequency Response TEDS**: it includes a table to find out the frequency response of the sensor.
- **Transfer Function TEDS**: it provides a series of constants that can be used to describe the transfer function of the transducer.
- **Text-based TEDS**: it includes information on **TIM** in a text format, and it can be in different languages. It is about a directory that simplifies access to specific sub block of **TEDS**, followed by XML defined blocks.
- **Commands TEDS**: it allows manufacturers to specify additional commands, which are not considered from the standard. These commands are in a textual format. These ones have to be inserted, mainly, in order to implement commands needed to tuning sensors or conditioning blocks.
- **Identification TEDS**: this **TEDS** was used to give textual information about the smart sensor, the transduction channel, and the calibration. Now this information is included in different specific **TEDS**, as established by the IEEE 1451.2, and these kinds of information are included in the **Text-based TEDS**.
- **Geographic location TEDS**: it stores statistic information about geographic position, in textual format. The user would write this information in order to indicate position in which the smart sensor is placed. The geographic position should be stored in the *Geography Markup Language* (GML), developed by the *Open Geospatial Consortium* (OGC) and specified in the ISO/DIS 19136.
- **Units extension TEDS**: this **TEDS** allows to give information about measurement units, in textual format. There could be

some cases in which it is not possible to state the physic quantity measured, even in a complex way, with an SI unit. As an example, for chemical sensors, which has the task to sense to a percentage of a concentration, result is a unit ratio (moles).

- **End User Application Specific TEDS**: memory block reserved to the end user, in which any information can be stored, with no restriction on the format.
- **Manufactured-defined TEDS**: these are constructor defined **TEDS**, for the purpose of giving specific information in additional **TEDS**, which are not defined by the standard. They are not needed by the standard specification; the manufacturer defines use and structure completely.

According to the standard IEEE 1451.0 specifications, **TEDS** must be structured in the TLV format, which is the data structure Type/Length/Value, as reported in Table 2.1.

Table 2.1 – Generic format for any TEDS

Field	**Description**	**Type**	**# octet**
---	TEDS length	UInt32	4
1 to N	Data block	Variable	Variable
---	Checksum	UInt16	2

2.2.2 IEEE 1451.1

The purpose of this document [32] is to provide a network-neutral application model that will reduce the effort in interfacing smart sensors and actuators to a network.

This document defines an interface to connect network capable processors to control network through the development of a common control network information model for smart sensors and actuators. The object model specifies the software component type used to implement application systems, and it includes definitions of transducer functions and NCAP blocks. The data model specifications include types and form of the information in both local and remote communication by the object interfaces. Eventually, the network communication models are specified as two way to interface the communication network and the application objects.

2.2.3 IEEE 1451.2

This document [33] has the purpose to specify an independent digital communication interface for point-to-point communication. This interface provides a minimum implementation subset that allows self-identification and configuration of sensors and actuators, and also it allows extensibility by vendors to provide growth and product differentiation. It provides all the specifications needed to implement the Transducer Independent Interface (TII): line definition, protocols, timing, electrical and physical specifications. It does not specify signal conditioning, signal conversion, or how the **TEDS** data is used in applications. This standard is being revised to interface with IEEE 1451.0 and to support two popular serial interfaces: UART and Universal Serial Interface.

2.2.4 IEEE 1451.3

The purpose of this document [34] is to define an independent standard for interfacing multiple physically separated transducers, which allows time synchronization of data. This standard provides a minimum implementation that allows multidrop, hot swapping, self-identification and configuration of transducers that may not be located in the same enclosure, but are confident to a relatively small space. So, it allows transducers to be arrayed as nodes, on a multidrop transducer network, sharing a common pair of wires. It acts from the third to the fifth layer of the ISO/OSI model (Network, Transport and Session layers), as shown in Fig. 2.3, in a physical network wired as shown in Fig 2.4 (TBIM is the acronyms for Transducer Bus Interface Module).

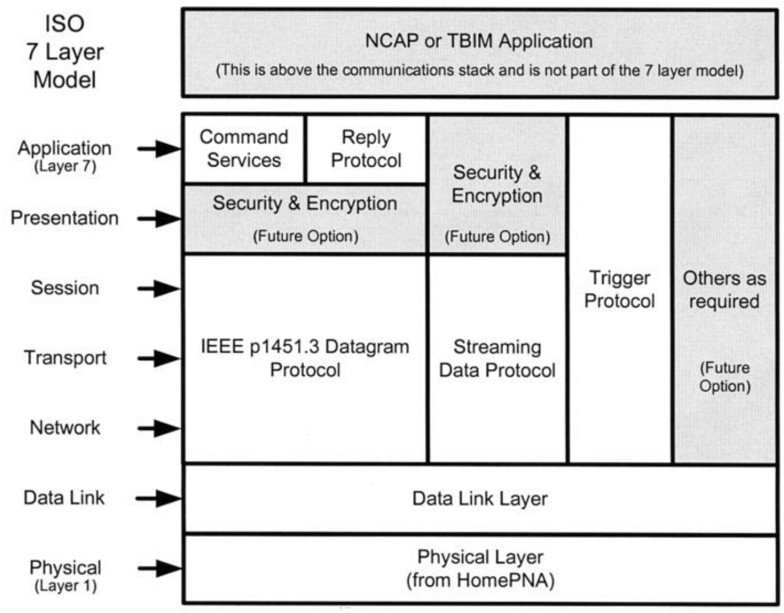

ISO
7 Layer
Model
NCAP or TBIM Application
(This is above the communications stack and is not part of the 7 layer model)
Application (Layer 7)
Presentation
Session
Transport
Network
Data Link
Physical (Layer 1)
Command Services
Reply Protocol
Security & Encryption (Future Option)
Security & Encryption (Future Option)
Trigger Protocol
Others as required (Future Option)
IEEE p1451.3 Datagram Protocol
Streaming Data Protocol
Data Link Layer
Physical Layer (from HomePNA)

Fig. 2.3 Model of the protocol stack.

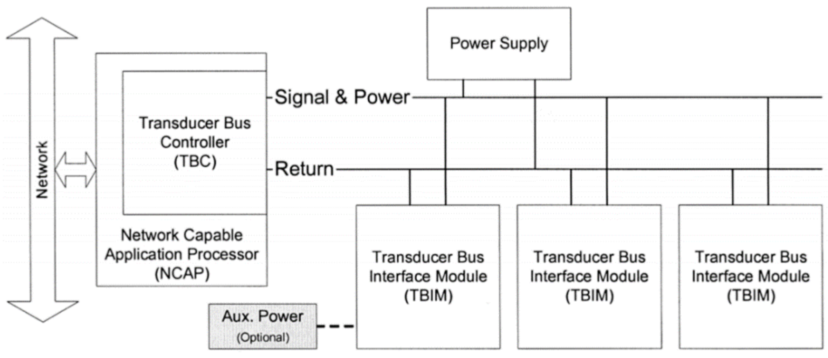

Network
Power Supply
Signal & Power
Return
Transducer Bus Controller (TBC)
Network Capable Application Processor (NCAP)
Transducer Bus Interface Module (TBIM)
Transducer Bus Interface Module (TBIM)
Transducer Bus Interface Module (TBIM)
Aux. Power (Optional)

Fig. 2.4 Physical context for the smart transducer interface.

2.2.5 IEEE 1451.4

This document [35] defines a mixed-mode transducer interface (MMI) for analog transducers with analog and digital operating modes. The standard provides two classes of communications for this Mixed-Mode:

- class 1: it sequentially transfers either a transducer signal or digital data;
- class 2: it uses separate connections to transfer transducer signals and digital data.

The **TEDS** model is also refined to allow above minimum of pertinent data to be stored in a physically small memory device, as required by tiny sensors.

2.2.6 IEEE 1451.5

The main purpose of this standard [36] is to define an open wireless transducer communication standard, which would accommodate various existing wireless technologies. Nowadays, wireless connections among electronic devices are increasingly used [37], so this document would enhance and support the acceptance of the wireless technology for transducers connectivity. It specifies radio specific protocols, thus, it requires a specific architecture, in which the NCAP and the WTIM (Wireless Transducer Interface Module) are equipped with one or more radio modules; the basic architecture is shown in Fig 2.5.

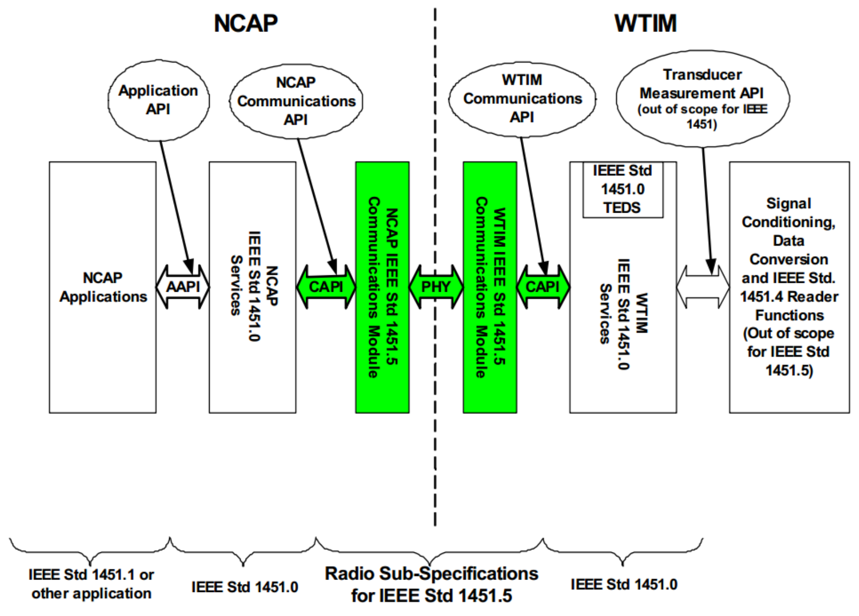

NCAP
WTIM
Application API
NCAP Communications API
WTIM Communications API
Transducer Measurement API (out of scope for IEEE 1451)
NCAP Applications
AAPI
NCAP IEEE Std 1451.0 Services
CAPI
NCAP IEEE Std 1451.5 Communications Module
PHY
WTIM IEEE Std 1451.5 Communications Module
CAPI
IEEE Std 1451.0 TEDS
WTIM IEEE Std 1451.0 Services
Signal Conditioning, Data Conversion and IEEE Std. 1451.4 Reader Functions (Out of scope for IEEE Std 1451.5)
IEEE Std 1451.1 or other application
IEEE Std 1451.0
Radio Sub-Specifications for IEEE Std 1451.5
IEEE Std 1451.0

Fig. 2.5 Functional context for the radio sub-specifications for IEEE 1451.5 services.

This standard specifies protocols for achieving this wireless interface: wireless standards such as 802.11 (WiFi) [38-41], 802.15.1 (Bluetooth) [42], 802.15.4 (ZigBee) [43], and 6LoWPAN [44-45]. It adopts necessary wireless interfaces and protocols to facilitate the use of technically differentiated, existing wireless technology solutions.

2.2.7 IEEE p1451.6

This is the project for edit the draft of the standard [46], which proposes a high-speed CAN open based Transducer Network Interface for Intrinsically Safe and non-Intrinsically Safe (IS) applications (Fig. 2.6). The proposed IEEE p1451.6 is a developing standard that defines a transducer-to-NCAP interface and TEDS. It adopts the CANopen device profile for measuring devices and closed-loop controllers. The application layer may be implemented free of licenses and royalties.

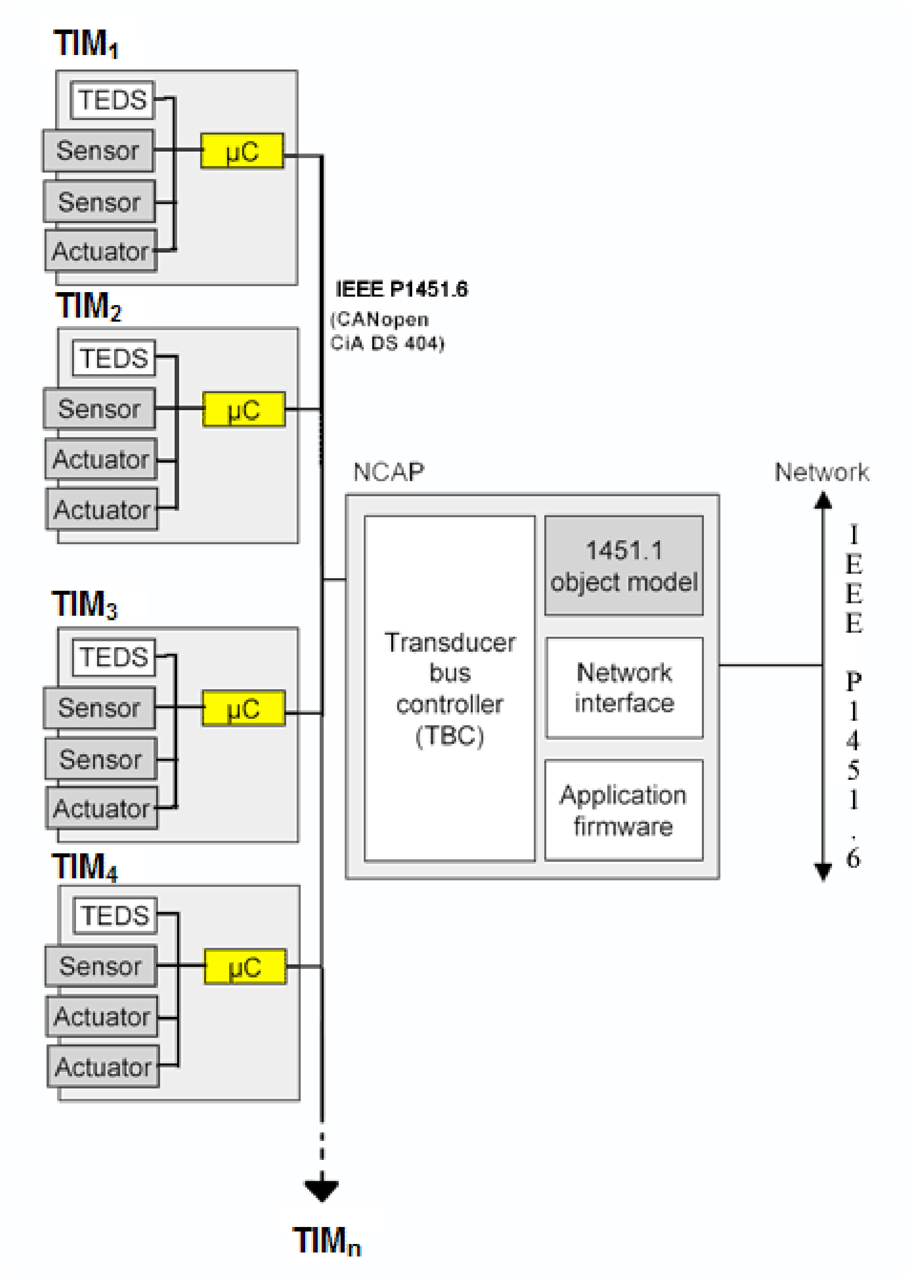


Fig. 2.6 Proposed architecture of the IEEE p1451.6.

2.2.8 IEEE 1451.7

This standard [47] opens new opportunities for sensor and RFID system manufacturers by providing sensors information in supply-chain reporting. Integration of Radio Frequency Identification in a sensor network is a necessary result for information technology and applications development [48]. This document provides methods for interfacing transducers and RFID tags, and for reporting transducer data within the RFID infrastructure. This standard will reduce the cost and time required to integrate transducer and RFID systems, as well as it will provide a means for device and equipment interoperability.

2.3 Issues regarding consistency and interoperability

There are some limitations in the standards and in the process of harmonizing the standards. "*The main concern with regard to the limitations is the consistency of standards, in terms of maintenance and management. It is a non-functional consideration, but crucial to determining whether a standard like IEEE 1451 can be more widely accepted and adopted by manufacturers*" [49]. Other issues are related to memory management, regarding, for example, storage of TEDS on tiny and low resources TIM.

In the last few years, these regulations of family of standards are being reviewed. Moreover, new standard versions will be also ISO (someone is already an active standard [50-51]), new projects are developing documents for other layers and, in the whole architecture, there is free space to allow new layers for other standardization proposals.

Chapter 3

The ISO/IEC/IEEE 21451.001 proposal, based on a segmentation and labeling algorithm

The IEEE 1451 family of standards is now evolving in the ISO/IEC/IEEE 21451 family of standards. In order to meet industry's needs, this family of standards, responsible for the developing technology for smart transducers, is growing: some of the old IEEE 1451 standard versions are under revision, whereas new members are being developed. Currently working groups are convened to revise the 21451-1, 21451-2 and 21451-4, whereas other working groups are developing new layers not envisaged in the old version of the family of standards. These renewal and integration operations of new layers are made necessary by the will to enhance and facilitate device and data interoperability of distributed sensor networks.

As refers to a distributed sensor network, different considerations should be done to understand the complexity of this kind of system: main problems being generally reliability, communication bandwidth, and the big amount of data, with the related problem of flooding the central elaboration system with more information than it can process.

In a distributed sensing system, each sensor provides information about its surroundings to the central system. In different applications, the number of sensing elements in a network changes dynamically: the computational capabilities demand for the central elaboration system would increase with the number of sensors connected to the network. It would be better to manage a sensor network, with a variable number of connected nodes, without changing the computational requirements for the central system. So, in many applications, a part of the data processing is moved from the central elaboration unit, to the sensor units, in such a way that allows sensors to pre-process data in the point of acquisition, giving as an output less data.

The challenge is to develop distributed and collaborative methods, in order to reduce the communication bandwidth required, and to increase reliability. However, the fusion central system in decentralized networks only receives condensed information from the sensors: the performance of such systems is suboptimal, in comparison with a completely centralized processing scheme, due to the loss of information in the local process [52]. Depending on the specific application and the hardware platform, methods could be optimized, quantifying data losses and comparing the performance versus the communication trade off, and, in literature, there are a large number of different examples of algorithms for clustering and classification [53-55].

On the other hand, to optimize an algorithm for the particular application or hardware is totally in contrast with the main goal of the family of standard for smart transducers: to achieve device and data interoperability. Moreover, "*the distributed measurement and control (DMC) industry is migrating away from proprietary hardware and software platforms, in favor of open systems and standardized approaches*" [56]. In order to propose a common preprocessing technique that will reduce problems, related to channel bandwidth and reliability, without requiring big computational capabilities, the *Institute of Electrical and Electronic Engineers (***IEEE***) Instrumentation and Measurement Society's (***IMS***) Technical Committee on Sensor Technology* has developed a new document to standardize this procedure. This document, identified with the code 21451.001 is now in the final step of developing before balloting and becoming a member of the ISO/IEC/IEEE 21451 family of standards. With this standard, smart transducers will ensure interoperability, using a common preprocessing technique and data structures.

3.1 A novel approach to signal preprocessing

ISO/IEC/IEEE 21451.001 is such a standard to regulate algorithms running on a transducer, a sensor or an actuator. The purpose is to facilitate the flow from sensor raw data processing to sensor information extraction and fusion, promoting and enhancing preprocessing on smart transducers.

The role of this preprocessing technique is to share part of the computational load needed in a sensor network among the nodes of the network, facing a smart transducer resource constrains, such as:

- low computational capabilities;
- limited storage;
- short battery life;
- inadequate communication ability.

Moreover, needed algorithms must be flexible, depending on the sensors and the specific application.

So, the standard is structured in order to make different configurations possible:

1) the classic configuration where there is no data processing on sensors, and all the data are sent to the central system for processing. In this configuration there are low computational load constrains on sensors, whereas there is big traffic load of data in the network (Fig. 3.1 sensor 1).
2) A configuration where data are processed by sensors, and only results are used, sending them directly to a cloud or a fusion center. In this configuration there is a big computational load for sensors (which generally entails cost and complexity) but a lightened load for the network (Fig. 3.1 sensor 2).
3) Another configuration, where data are pre-processed by sensors and post-processed by central systems, which mitigate both the computational load required for sensors and the network traffic load (Fig. 3.1 sensor 3).

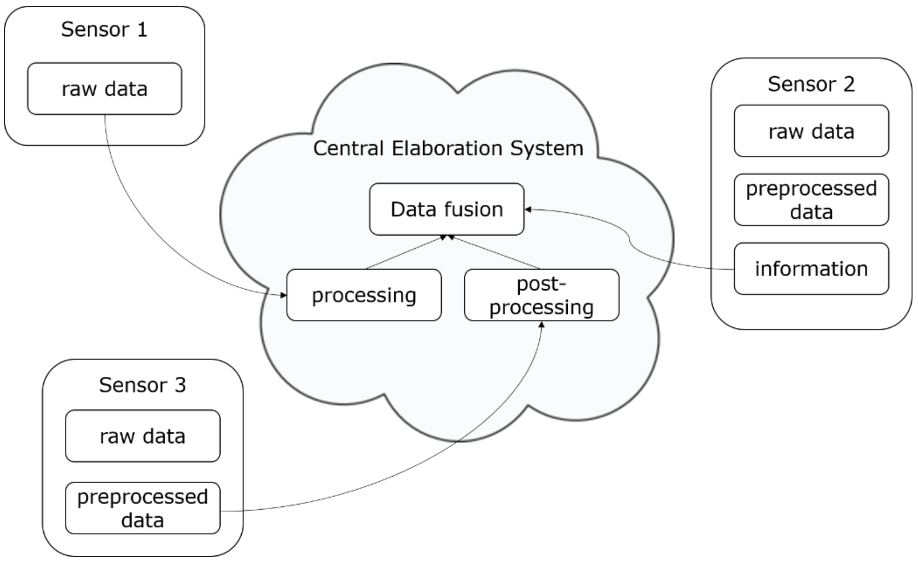


Fig. 3.1 Different configuration for a smart transducers network.

Using the hardware needed by the IEEE 21451 Transducer Interface Module, which requires a system based on a micro-controller, it is possible to implement, outside or directly on the TIM, data treatment services, providing extracted features rather than raw data for further analysis and processing.

The purpose of this preprocessing technique is to extract basic information from the raw signal, the same information that a human being could provide looking at the signal [57, 58]. This technique emulates a human observer, in the sense that it relates samples to infer a global behavior, detecting, for example, a minimum or a maximum, or inferring signal shape parameters.

The basis of this technique is the real-time segmentation and labeling (RTSAL) algorithm, which derives a data structure from the raw data [59].

3.2 A real time segmentation and labeling algorithm

The basic idea of this algorithm is that it would describe the input signal as a human being could do: he would not enumerate all the acquired samples, but he would describe the shape of the signal, with only some noteworthy samples. This technique bases its analysis on the hypothesis that the needed information is not in any single sample, but it is contained in the relation among them. The aim of this technique is to store several meaningful samples, with which it is possible to keep several information about the signal, and from which is possible to extract several simple features. In this way, it is possible to have the same informative content, removing the redundancy of an oversampled input signal. Moreover, it implements a classification of the acquired raw samples, describing the signal trend.

The algorithm is described in the following paragraphs, divided into two parts: the segmentation and labeling phase.

3.2.1 Segmentation Step

In this first step, the algorithm chooses the samples, to be saved from the raw samples, acquired with a constant sampling time. It works according to an input parameter, which is a kind of threshold for the algorithm: the *interpolation error*. Once it starts, it acquires the first three samples, and it linearly interpolates the first and the third sample, in order to calculate the amplitude the sample in the middle would have, if the signal were strictly linear. Then the algorithm calculates the difference between the amplitude of the interpolated sample and the amplitude of the sample acquired in that instant of time (Fig. 3.2).

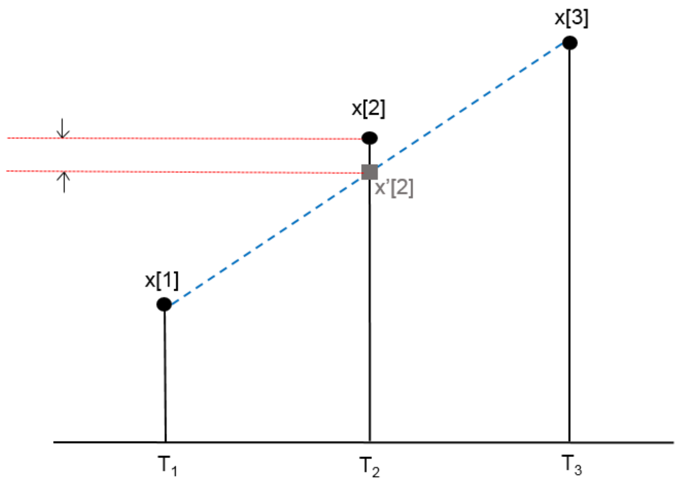


Fig. 3.2 The RTSAL algorithm makes linear interpolations and comparisons.

At this point, the difference (**d**) is compared with the interpolation error, defined as a percentage value of the maximum value of the signal: if this difference does not exceed the interpolation error, the algorithm does not store the sample, and, once acquired two more samples, it interpolates the first and the fifth sample, computing the value of the third middle sample, to be compared with the real acquired sample. This algorithm goes on like this (Fig. 3.3, 3.4), increasing the interval in which it searches an important sample, until it finds a new one, or a maximum limit of comparisons is reached.

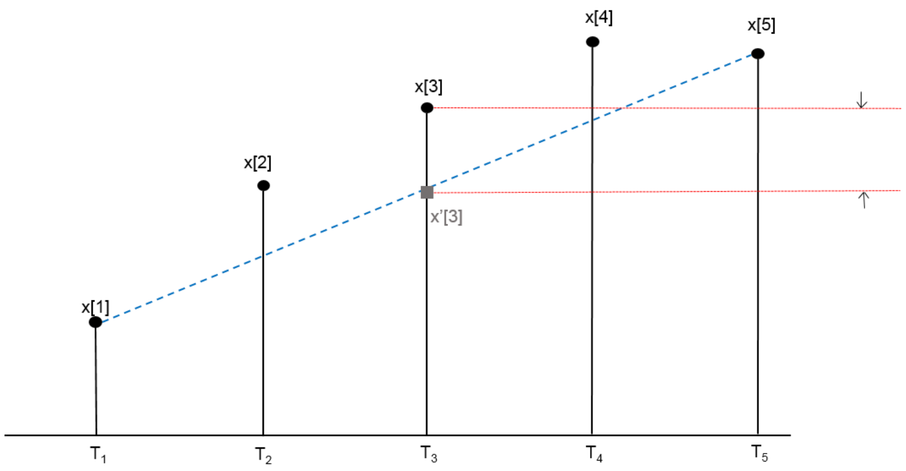


Fig. 3.3 Example of RTSAL algorithm comparison, on an interval of five samples.

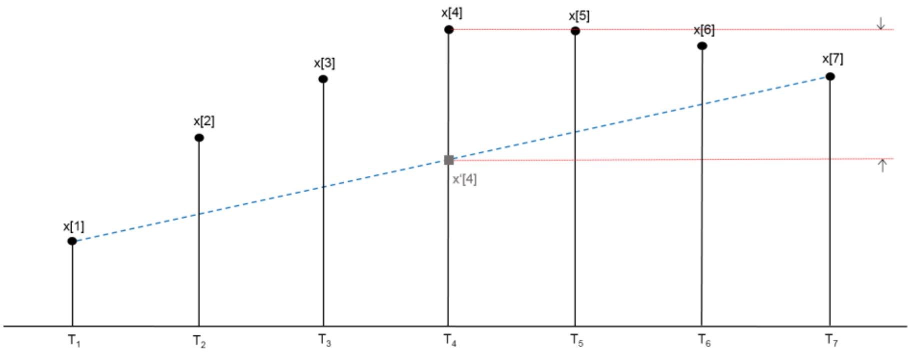


Fig. 3.4 Example of RTSAL algorithm comparison, on an interval of five samples.

Whereas if **d** is bigger than the interpolation error, the algorithm stores the amplitude and the time position of the sample, considering this sample as important. These two values are stored in two different vectors (called “**M**”, for amplitude, and “**T**”, for time). When it stores a new sample, it begins again, starting from the last stored sample. Required operation for this algorithm are only sums, subtractions, and just one division by two for each iteration.

In Fig 3.5 there is the flowchart with an example version of the algorithm [59], where the interpolation error is compared with the parameter **d** computed as in (1), where *i* and *k* are two different counters, the first increased by one every time a new point is stored, whereas *k* is increased by two when a new point is not stored, to increase the interval.

$$\mathbf{d} = \left|\frac{\mathbf{S}(i)+\mathbf{S}(i+k)}{2} - \mathbf{S}\left(i+\frac{k}{2}\right)\right| \tag{1}$$

The *interpolation error* could be interpreted in different ways: it is a threshold for amplitude values, so it could be specified whether in the unit of the acquired signal, or coded to be compared directly with the output of the analog to digital converter (an entire number), or as a relative value. In the following paragraphs, the *interpolation error* value will be stated as a relative value of the full scale value of the sensor, except where expressly indicated (for the same reasoning, in simulations, signals have unitary maximum value).

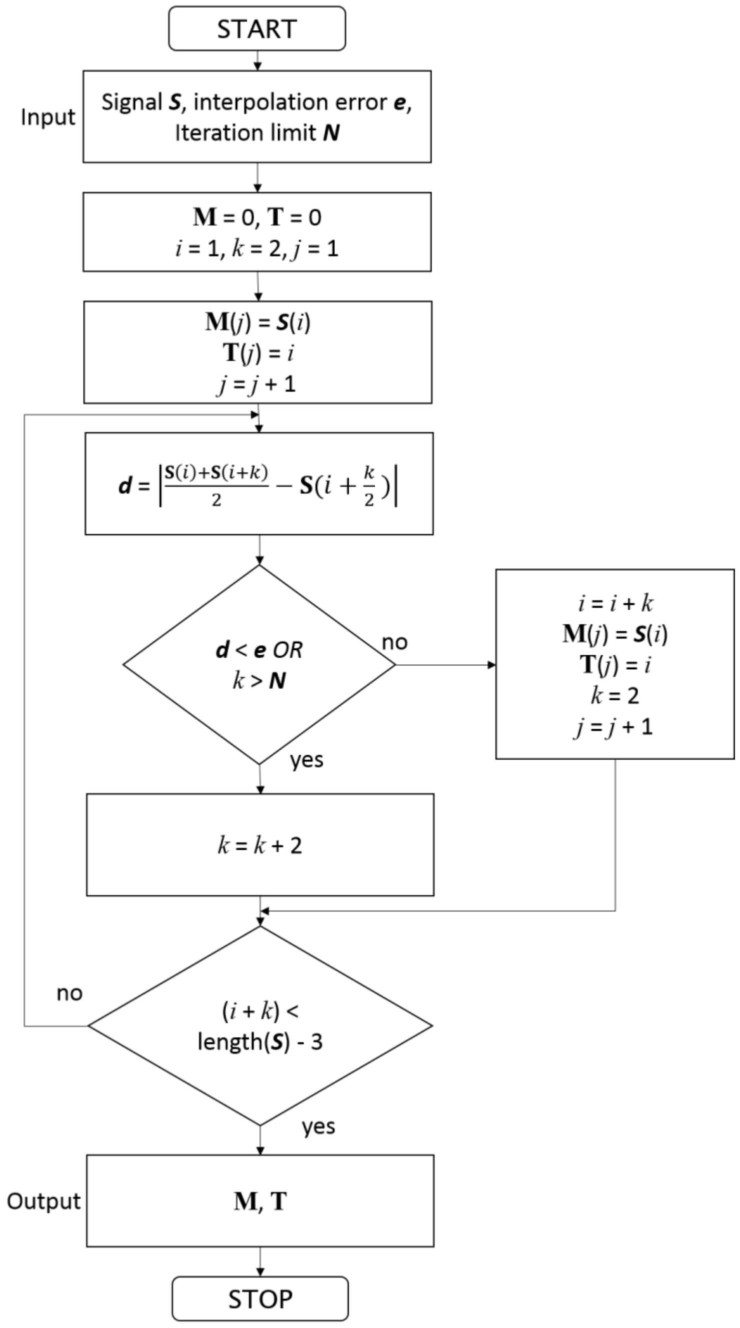

START
Input
Signal $\boldsymbol{S}$, interpolation error $\boldsymbol{e}$, Iteration limit $\boldsymbol{N}$
$\mathbf{M} = 0$, $\mathbf{T} = 0$
$i = 1$, $k = 2$, $j = 1$
$\mathbf{M}(j) = \boldsymbol{S}(i)$
$\mathbf{T}(j) = i$
$j = j + 1$
$\boldsymbol{d} = \left|\frac{\mathbf{S}(i)+\mathbf{S}(i+k)}{2} - \mathbf{S}(i + \frac{k}{2})\right|$
$\boldsymbol{d} < \boldsymbol{e}$ OR $k > \boldsymbol{N}$
no
$i = i + k$
$\mathbf{M}(j) = \boldsymbol{S}(i)$
$\mathbf{T}(j) = i$
$k = 2$
$j = j + 1$
yes
$k = k + 2$
no
$(i + k) <$ length($\boldsymbol{S}$) - 3
yes
Output
$\mathbf{M}$, $\mathbf{T}$
STOP

Fig. 3.5 Segmentation algorithm flowchart.

3.2.2 Labeling Step

This step starts only when a new segment is saved. In this phase, the algorithm chooses a class in order to label the shape of the signal between two consecutive stored samples.

During the first phase, the algorithm saves also the sign of each computed difference **d**. When a new sample is stored, the tendency of the signal between this sample and the previous one is evaluated depending on the difference between these values, and on the majority of error sign accumulation. It uses ‘***d***’, ‘***e***’, ‘***f***’ and ‘***g***’ or ‘***h***’. If the algorithm saves a new value because the maximum length is reached, it uses linear classes ‘***a***’, ‘***b***’ and ‘***c***’, depending on the slope. A graph of labeling process is shown in Fig. 3.6.

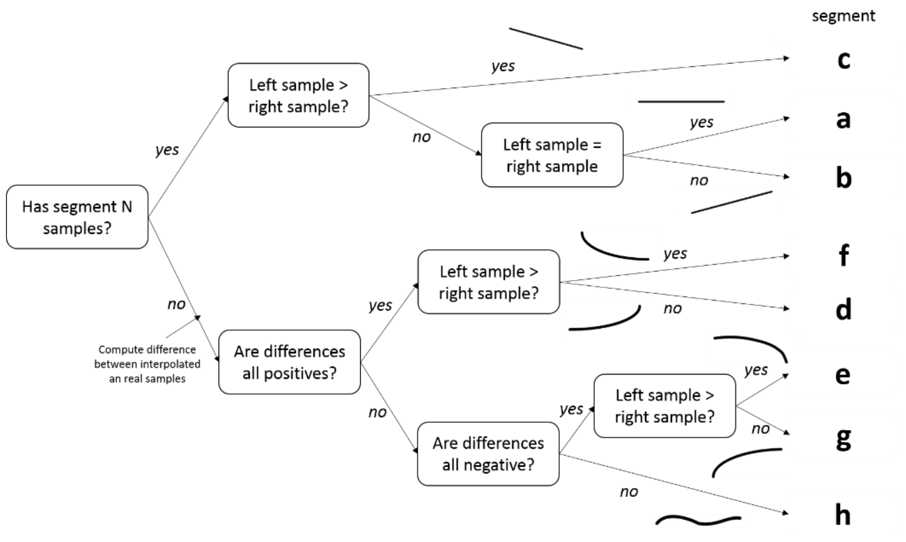

segment
Has segment N samples?
yes
no
Compute difference between interpolated an real samples
Left sample > right sample?
yes
no
Left sample = right sample
yes
no
Are differences all positives?
yes
no
Left sample > right sample?
yes
no
Are differences all negative?
yes
no
Left sample > right sample?
yes
no
c
a
b
f
d
e
g
h

Fig. 3.6 Labeling scheme of comparisons.

For each sample, the class is stored in a third vector, called “**C**”: in this vector all information about the signal shape is saved. Several signal parameters could be easily extracted by processing the elements of this vector [59]; it has not any relation with amplitude or time scales, it stores only a qualitative information. The most common classes are ‘***d***’, ‘***e***’, ‘***f***’ and ‘***g***’, and the sequence of these classes gives additional information on the signal trend; the occurrence of classes ‘***a***’, ‘***b***’, ‘***c***’, and ‘***h***’ is reduced, increasing the sampling frequency, compared to the frequency of the input signal (a condition called *oversampling*), and/or decreasing the interpolation error value. As an example, maxima occurrences can be located with sequences of classes ‘***df***’, ‘***de***’, ‘***gf***’ and ‘***ge***’, whereas minima with sequences of classes ‘***eg***’, ‘***ed***’, ‘***fg***’ and ‘***fd***’. An exponential shape can be found checking the repetition of the same class ‘***d***’, ‘***e***’, ‘***f***’ or ‘***g***’, and even other simple features could be identified by a specific pattern of consecutive classes.

The set of three values stored in **M**, **C** and **T** arrays are also called segments: for each set, one input sample is stored, with its temporal position and class.

The **RTSAL** algorithm requires only few operations: one sum, one subtraction, and one division by two (which could be easily implemented as a shift in a binary register) each iteration. So, it is easy to implement on micro-controllers and can run in real time.

3.3 Draft proposal for a recommended practice

The set of vectors *mark*, *class* and *time*, in short **MCT,** is the basic structure on which are grounded algorithms proposed by the draft of this recommended practice. The dimension of a single variable is fixed, whereas the size of the vectors depends on the RAM dedicated for this practice.

The amplitude value, stored in the *mark* vector, could be an unsigned integer, if it contains the output value of the analog to digital converter (2 bytes, UInt16). The class value, stored in the *class* vector, as already explained in the previous paragraph, can be only one of eight different values (1 byte, UInt8). The time stamp, stored in the *time* vector, works as a time reference (millisecond from midnight, 4 bytes variable, UInt32). These vectors could be continuously generated and stored in circular buffer structures.

Algorithms included in this recommended practice are divided in two layers: the first layer algorithms take the **MCT** structure as an input, whereas the second layer algorithms also take results of the first layer algorithms as an input. The algorithms in the first layer are for the extraction of simple and basic properties or parameters: these algorithms are the *exponential detection*, the *noise detection*, the *impulsive noise detection*, the *sinusoidal pattern detection*, the *tendency estimation* and the *mean estimation*. The algorithms in the second layer are conceived for the computation of complex parameters: these algorithms are the *steady state value estimation*, *smart filtering* and *compression*. There is also a *user defined application code* block, which allows manufactures or users to include their own algorithms based on **MCT** vectors and/or first layer algorithms. This structure (Fig. 3.7) is flexible, meaning that it could be implemented on different levels, choosing layers to be implemented, depending on the hardware and other constraints.

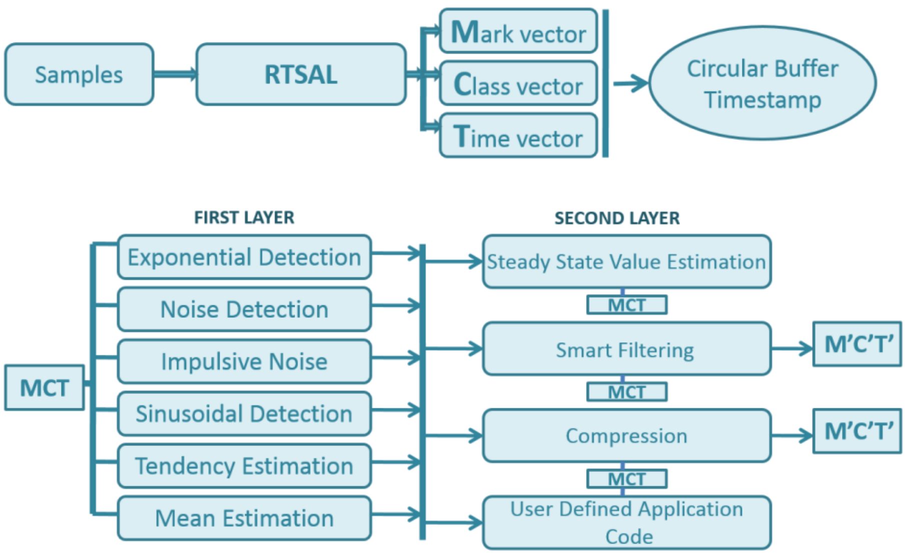


Fig. 3.7 Algorithms based on MCT structure.

3.4 First layer algorithms

3.4.1 Exponential Pattern Detection

The aim of this function is to detect an exponential shape, with a fixed number of consecutive segments, using **MCT** vectors. The exponential detection function searches through the class vector, looking for consecutive equal classes. The number of times that a class has to be consecutively found is the only input parameter for this algorithm. When it finds a fixed number of consecutive segments of the same class, it stores the starting point and continues, until the class of the following segment is the same as the previous one. It returns the class detected, as information on the shape of the exponential signal, and the time interval in which the pattern is detected. Below, an example flowchart for this algorithm (Fig. 3.8) and examples of Matlab® simulations are shown (Fig. 3.9 and 3.10).

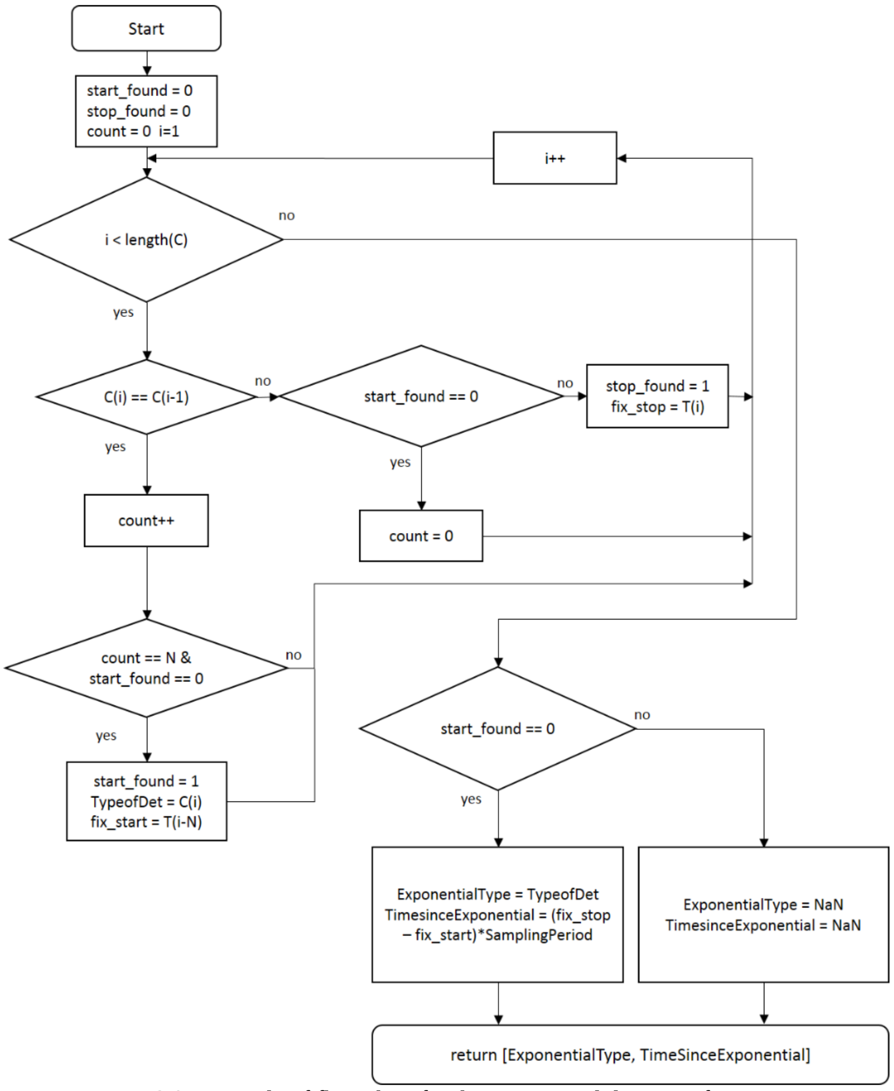


Fig. 3.8 Example of flow chart for the exponential detection function.

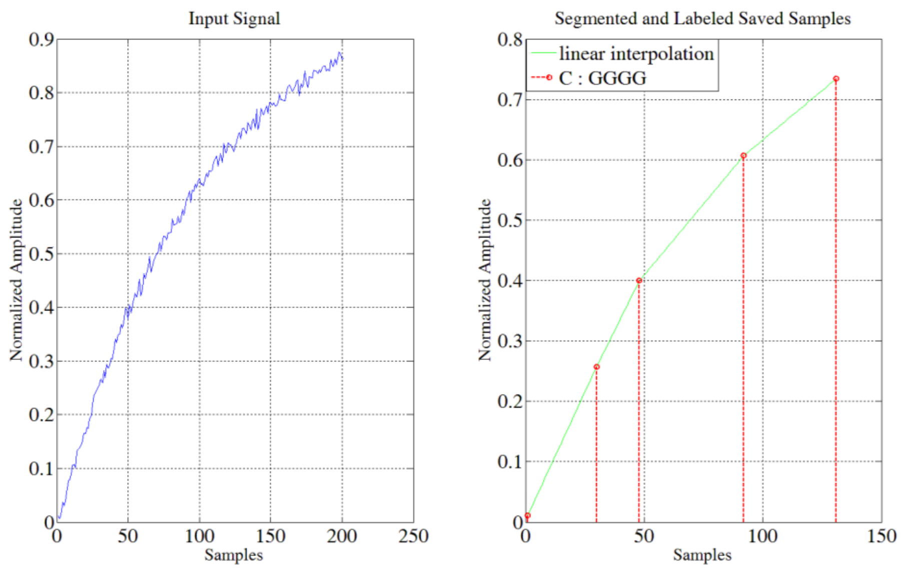


Fig. 3.9 Example of exponential patter detection, class ‘***g***’ (interpolation error 0.03). On the left the acquired signal, on the right, in red dots, the segments stored in the MCT structure.

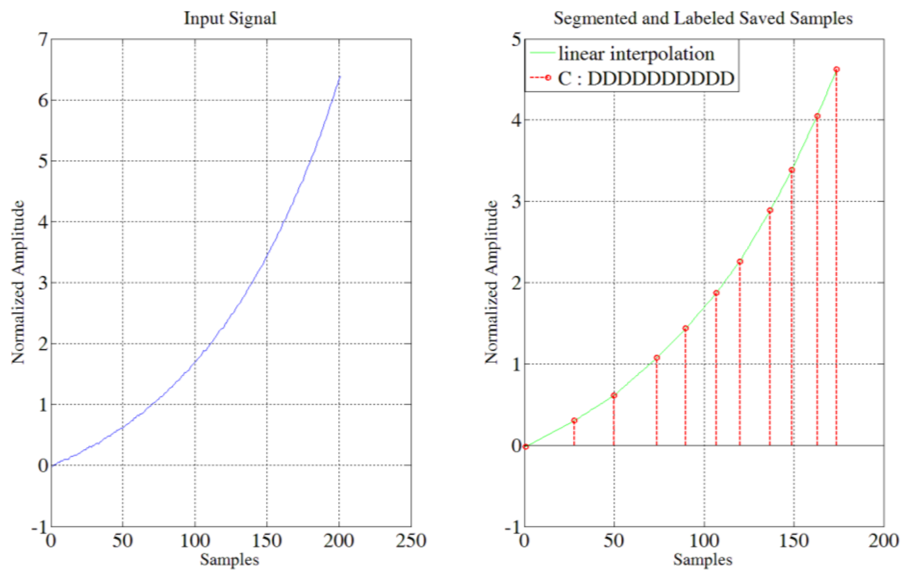


Fig. 3.10 Example of exponential patter detection, class ‘***d***’ (interpolation error 0.03). On the left the acquired signal, on the right, in red dots, the segments stored in the MCT structure.

3.4.2 Noise Detection

This function detects noise computing the distance between maxima and minima. For this type of labeling, a maximum is always followed by a minimum, and vice versa. The algorithm has three input parameters: the *NoisyPeriod*, *AmountNoisy*, and *ThresholdPeriodicDetection*. This algorithm compares the time interval between a maximum and its following minimum, or between a minimum and its following maximum, with the *NoisyPeriod*, an input parameter for the maximum time interval, as well as counting the number of times that the MCT version of the signal is smaller than this minimum interval consecutively. If this count is bigger than the *AmountNoisy*, the algorithm return a flag that indicates detection of noise. Moreover, it could be possible to implement a periodic noise detection: standard deviation of the noise is an indicator of random noise. Computing the mean and the standard deviation of periods observed in the temporal window, the algorithm calculates the ratio, and compares this value with the input parameter *ThresholdPeriodicDetection*. If this ratio is lower than the fixed threshold level, the algorithm calculates the period of the signal, and it returns this value as an output. Below some examples of different Matlab® simulation are shown (Fig. 3.11 and 3.12).

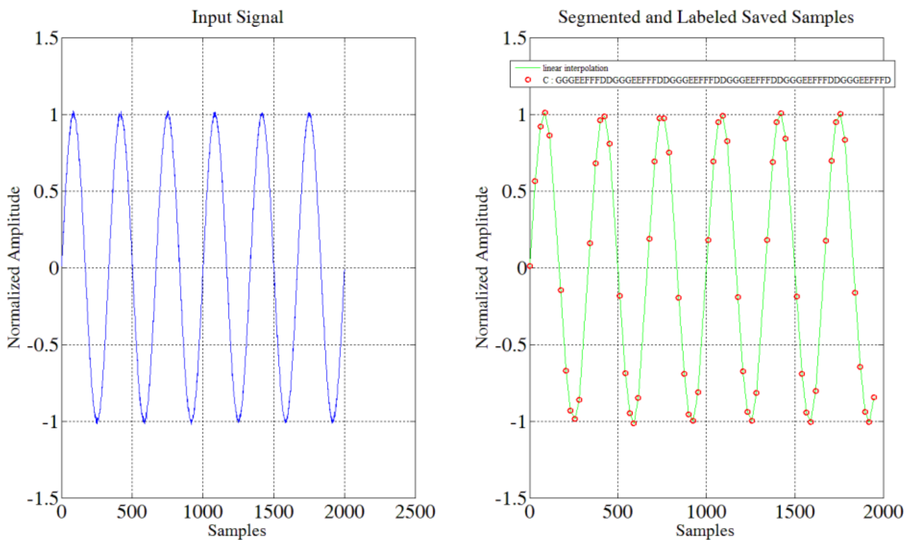


Fig. 3.11 Example of sinusoidal waveform, not flagged as “Noisy” (interpolation error = 0.1, signal frequency = 30 Hz, *NoisyPeriod* = 1ms, *AmountNoisy* = 5, SNR = 40 dB).

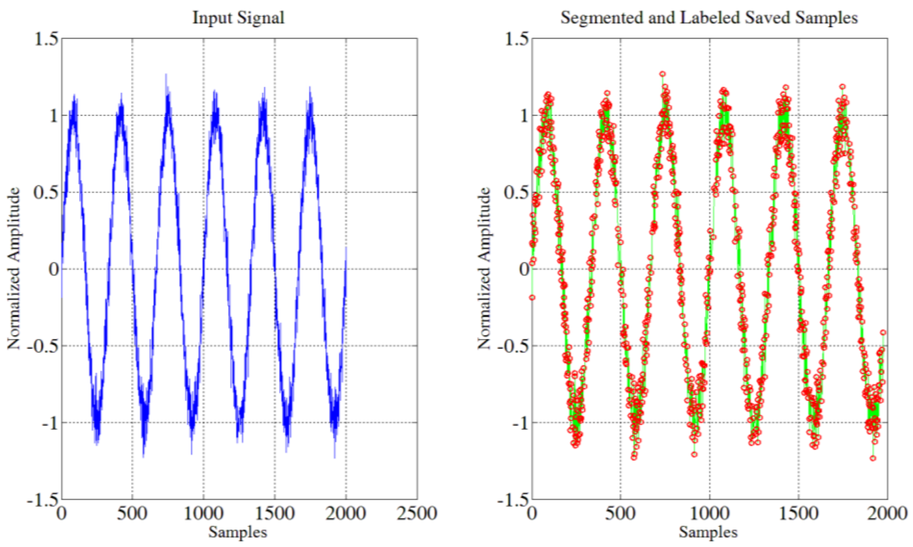


Fig. 3.12 Example of sinusoidal waveform, flagged as "Noisy" (interpolation error = 0.1, signal frequency = 30 Hz, *NoisyPeriod* = 1ms, *AmountNoisy* = 5, SNR = 20 dB).

3.4.3 Impulsive Noise Detection

This function considers as an impulsive noise a significant change of the amplitude in two consecutive segments: when it finds a minimum or a maximum, it computes the amplitude change, taking into account the amplitude of the segment before the minimum or the maximum. Input parameter for this function is the *ImpulseThreshold*, which defines a threshold for the amplitude differences. The Impulsive Noise Detection function browses the entire class vector, looking for maxima and minima and, when it finds one, it computes the amplitude difference between this value and the previous one, and then compares this last value with the threshold. This algorithm could return several different information about the impulse, like position in the time vector, amplitude, duration or area. As an example, different simulation results are reported (Fig. 3.13 and 3.14)

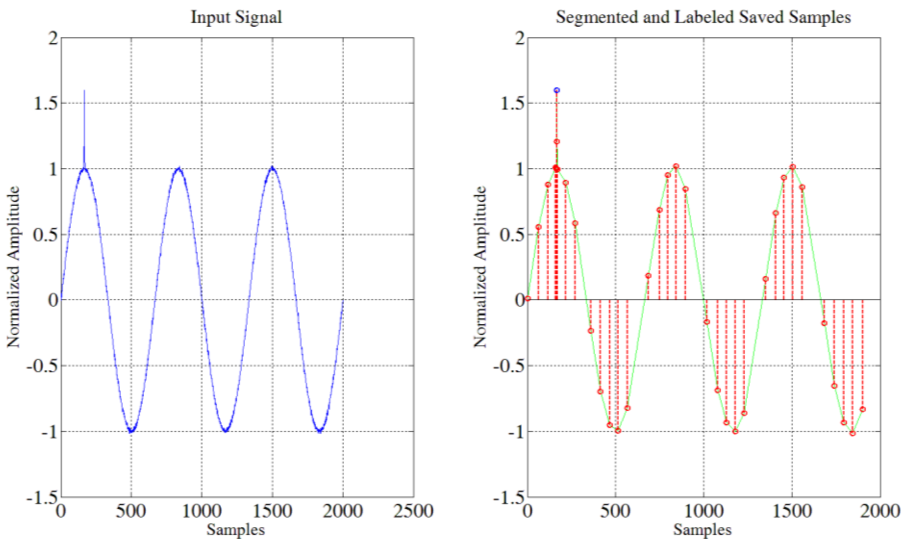


Fig. 3.13 Example of sinusoidal waveform with an impulse added (interpolation error = 0.1, signal frequency = 15 Hz, *ImpulsiveThreshold* = 0.3).

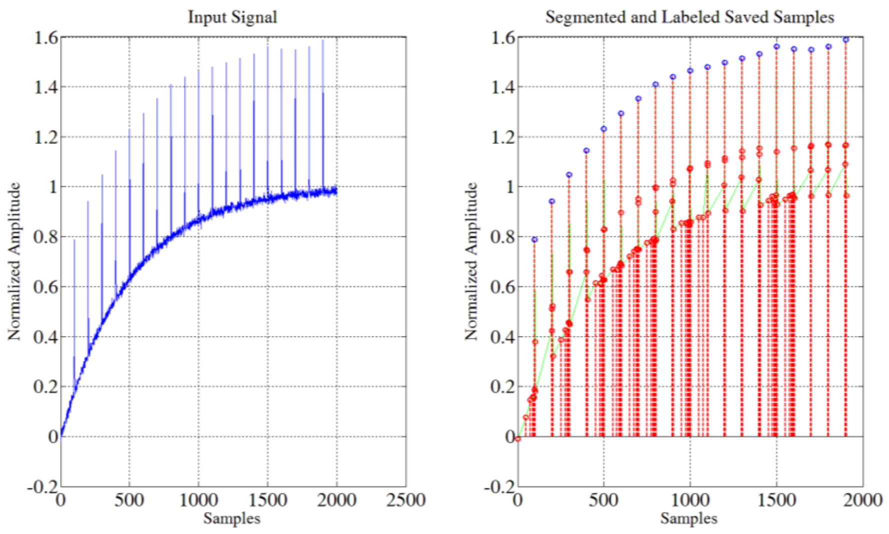


Fig. 3.14 Example of exponential waveform with periodic impulses added. (interpolation error = 0.1, signal frequency = 15 Hz, *ImpulsiveThreshold* = 0.3).

3.4.4 Sinusoidal Pattern Detection

The sinusoidal pattern detection algorithm is based on the pattern of classes shown by a sinusoidal waveform, saved in the **MCT** structure with the segmentation and labeling algorithm. The sinusoidal pattern of classes is composed of segments ‘***g-e-f-d***’: the classes has to be in this order to assume that the input signal is a sinusoidal waveform. The algorithm search through the class vector and, if the pattern of classes is correct, it returns the period of the sine, computed using the value of the time vector of two consecutive maximum segments, and it checks if the input signal has damped or undamped oscillations. An example of Matlab® simulation is shown (Fig. 3.15).

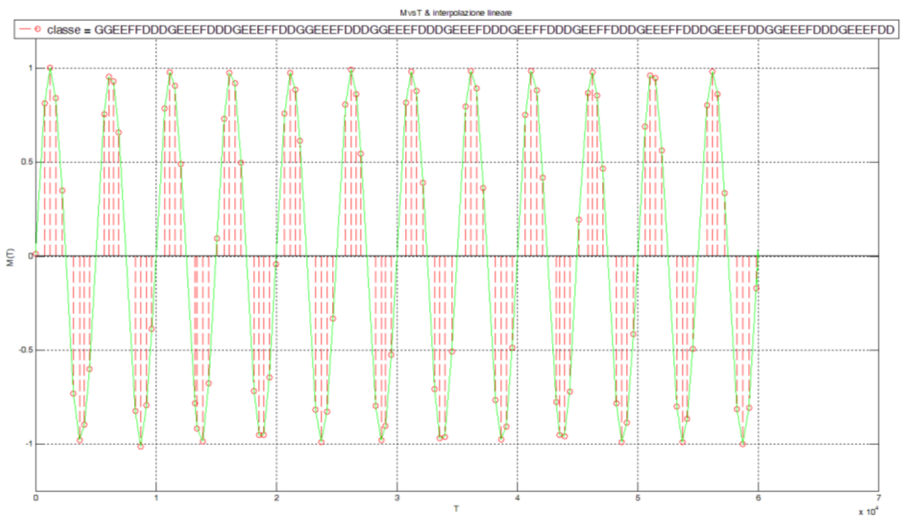


Fig. 3.15 Segments saved by the segmentation and labeling algorithm of an ideal sinusoidal waveform, with interpolation error equal to 0.06 (signal frequency = 60).

In order to clarify the role of the interpolation error on the conversion in **MCT** by the segmentation and labeling algorithm, results of the sinusoidal pattern detection function are shown in the top of the Fig. 3.16. In this chart, red dots are for a correct pattern labeled by the algorithm, whereas blue dots are for an incorrect pattern for a sinusoidal waveform: this analysis is carried fixing the input signal, varying the interpolation error used to segment the input signal, and the signal to noise ratio with which the signal is generated. As an example, in the center and in the bottom of Fig. 3.16, two different cases of segmentation of the same sinusoidal signal are shown, with interpolation error of 0.2 and 0.05, respectively.

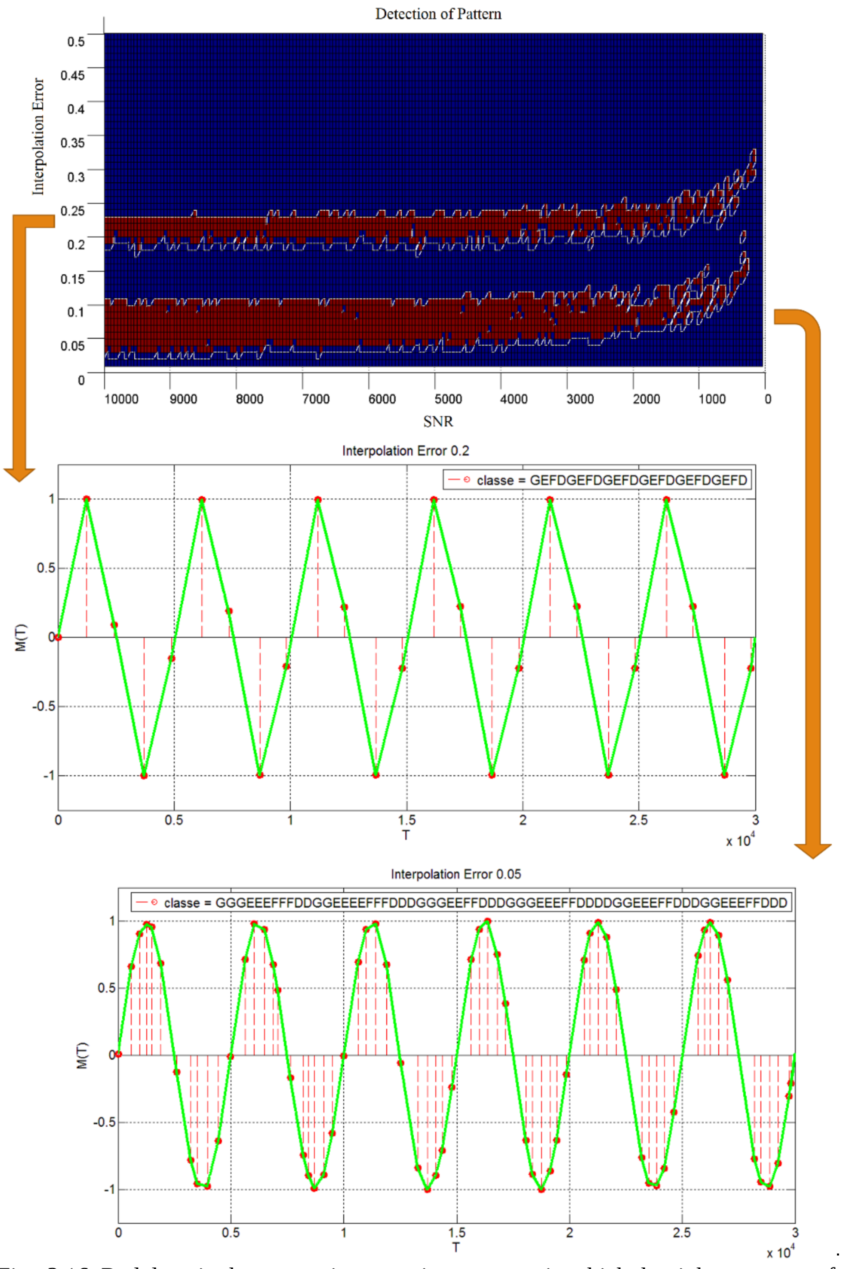


Fig. 3.16 Red dots, in the upper picture, point out cases in which the right sequence of classes is stored, as a function of signal to noise ratio and interpolation error.

3.4.5 Tendency Estimation

Tendency of input signal is computed processing only maximum and minimum segments: the algorithm applies formula (2) on a vector made by maximum segments, and formula (3) on a vector made by minimum segments. The number of segments taken into account depends on application: it is possible to run this algorithm when the **MCT** buffer is full, or it is possible to run this algorithm on the segments acquired in a fixed time window. These formulas return a normalized factor which gives information about trends, and their span is from -1 to +1. If the tendency of maxima is positive and the tendency of minima is negative, the signal grows its peak to peak amplitude (example: unstable oscillations), whereas, if the tendency of maxima is negative and the tendency of minima is positive, the signal diminishes (example: damped oscillations). In addiction, the average of these two factors returns information about the global tendency of the signal.

$$\text{Trend}_{\text{max}} = \frac{\sum_{max}(T_{x+1}-T_x)(M_{x+1}-M_x)}{\sum_{max}(T_{x+1}-T_x)|M_{x+1}-M_x|} \tag{2}$$

$$\text{Trend}_{\text{min}} = \frac{\sum_{min}(T_{x+1}-T_x)(M_{x+1}-M_x)}{\sum_{min}(T_{x+1}-T_x)|M_{x+1}-M_x|} \tag{3}$$

3.4.6 Mean Estimation

One of the most important algorithms, in different applications, is the mean computation. This function could be implemented in different ways: the mean of the vector **M** of the **MCT** structure, take into account even values of the vector **T**, because samples stored in the **MCT** structure are not equally spaced in time (see par. 4.2.5). Even in this case, the number of segments taken into account depends on the application (running this algorithm when the **MCT** buffer is full, or when segments acquired in a fixed time window are stored).

3.5 Second layer algorithms

3.5.1 Steady state value estimation

The aim of this second layer function is to estimate the steady state value of an exponential signal. When an exponential signal is detected, using the first layer function shown in paragraph 3.4.1, the steady state value of this signal can be estimated, using values stored in the **MCT** vectors. For example, if a class '***g***' exponential signal is detected (Fig. 3.17), the steady state value can be estimated. The formula (4) gives the general function for this kind of signals, where X_M is the steady state value, and X_0 is the starting value.

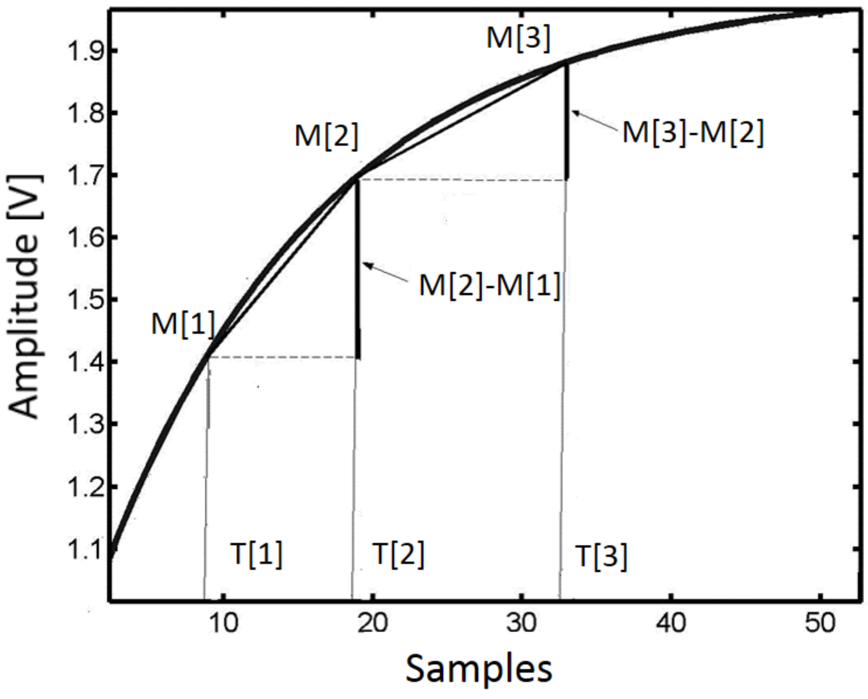


Fig. 3.17 The steady state value is estimated, using an approximation of the signal slope.

$$x(t) = X_M - [X_M - X_0]e^{-t/T} \quad (4)$$

The idea is to approximate the shape of the exponential between two consecutive segments in the **MCT** structure with a linear shape: the samples stored in the **MCT** structure are on the exponential curve, whereas the value of X_M, the steady state value, and T, the time constant, are unknown. The derivate of $x(t)$ at $t=0$ is:

$$\frac{dx}{dt}\bigg|_{t=0} = \frac{X_M - X_0}{T} \tag{5}$$

The derivate can be approximated with the slope of segments at the starting points. Considering:

$$\frac{X_M - M[1]}{T} = \frac{M[2]-M[1]}{T[2]-T[1]} = M_a \tag{6}$$

$$\frac{X_M - M[2]}{T} = \frac{M[3]-M[2]}{T[3]-T[2]} = M_b \tag{7}$$

where M_a and M_b are slopes in the starting point. Solving (6) and (7) for X_M:

$$X_M = \frac{M_a M[2] - M_b M[1]}{M_a - M_b} \tag{8}$$

The steady state value X_M can be estimated, without knowing time constant T.

3.5.2 Smart Filtering

Basing on the **MCT** data structure, it is possible to reconstruct an approximation of the sampled signal. The starting acquired signal is included in an interval smaller than the *interpolation error* value used in the segmentation step. The reconstructed signal is a combination of a scaled and shifted version of two generating functions:

$$\mathrm{x(t)} = \sum a_k \varphi(t/a_k - i) \tag{9}$$

where $\varphi(t)$ is the normalized generating function for each subspace, as shown in Fig 3.18.

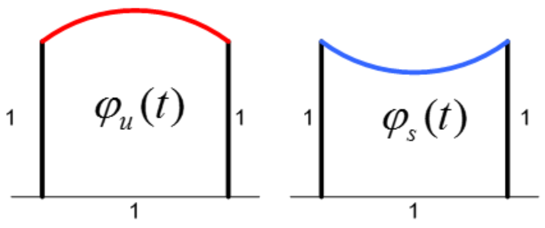


Fig. 3.18 Normalized generating functions for segments.

Segments are input signal samples stored, the reconstruction of the signal has to assume values of these essential samples. Spline polynomials can be used to reconstruct the signal within the segment. The two generating functions converge to a straight line between essential samples, in case of linear interpolation reconstruction.

A low pass filtering effect can be achieved, applying repeatedly the **RTSAL** algorithm on an acquired signal: once obtained the **MCT** structure, from the input samples, it is possible to reconstruct the signal using a linear interpolation, and then, applying the **RTSAL** on this reconstructed signal, it is possible to obtain a new **MCT** structure. This new set of vectors, **M**'**C**'**T**', has lost part of the high frequency content, and this principle can be iterated, in order to remove some unwanted high frequency components or noise.

Applying **RTSAL** algorithm, the number of local minima and maxima on a fixed time window can be used as a control variable for the number of iteration. In the following picture (Fig. 3.19), a flow chart of the proposed method is shown. The variable **N** is the total number of minima and maxima of the time window analyzed by the **RTSAL** algorithm in the current iteration, whereas **N'** is the total number of minima and maxima in the previous one. This procedure generates an **MCT** data structure for the input signal, after that it reconstructs the signal with a linear interpolation, with the same time step with which the input signal is acquired, and then it computes the value of **R** (**R** = (**N** – **N**')/**N**'). If **R** is smaller than 0.1, the procedure is repeated. An example of application is reported in Fig. 3.20, with a Matlab® simulation.

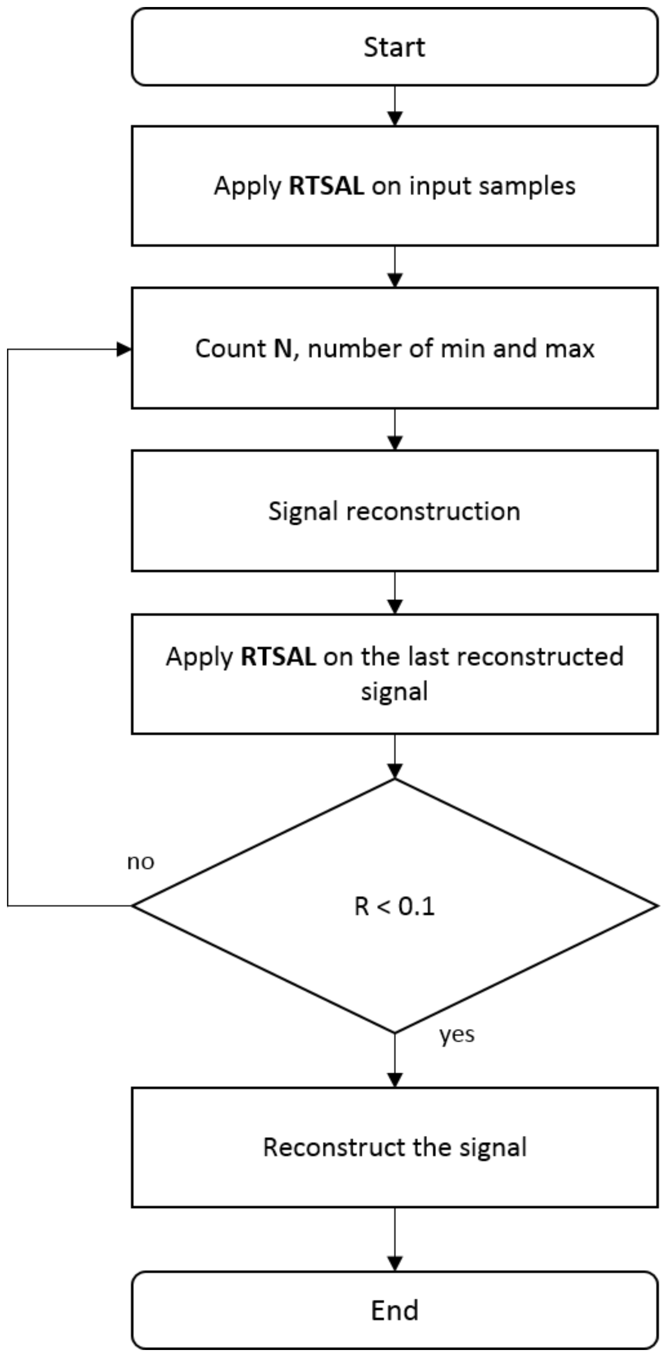

Start
Apply RTSAL on input samples
Count N, number of min and max
Signal reconstruction
Apply RTSAL on the last reconstructed signal
R < 0.1
no
yes
Reconstruct the signal
End

Fig. 3.19 Flowchart for low pass filtering.

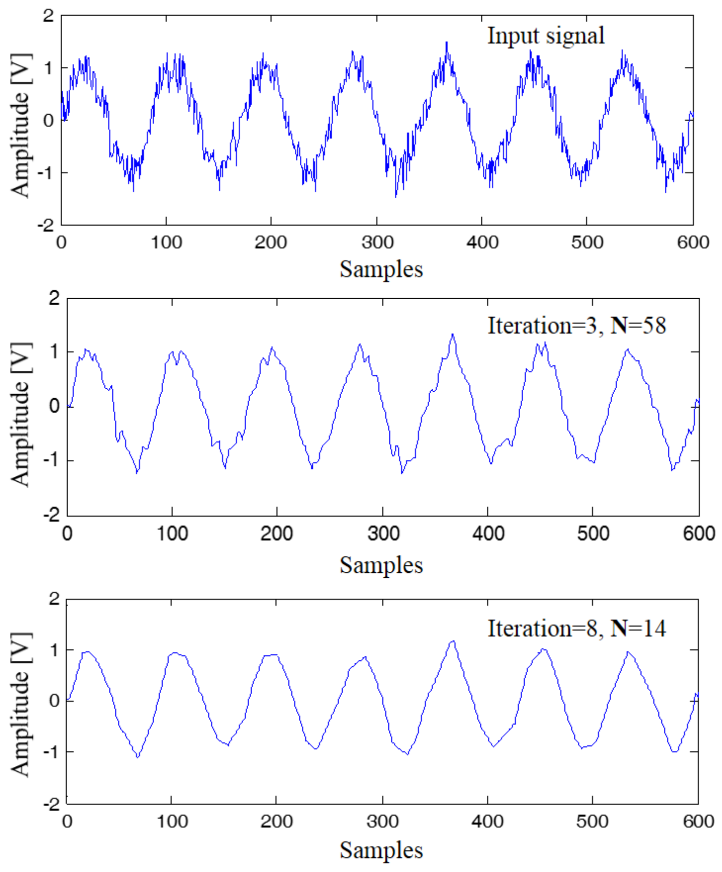


Fig. 3.20 Filtering: input signal (top), reconstructed signal after 3 iterations (middle), and reconstructed signal after 8 iteration (bottom).

3.5.3 Compression

The **RTSAL** algorithm achieves a conversion of the input vector of samples in a data structure, but it does not mean that there are benefits in all the cases. A correct choice of the *interpolation error* could allow both enough information and data size reduction, but it requires to know the typical input signals. In order to obtain a compression, even when the domain of the input signal is unknown, an automatic procedure to make a reasonable choice of the *interpolation error* is needed. A large value of the *interpolation error* leads to high compression rate and compression error, whereas smaller value of the *interpolation error* results in lower compression rate and error. Input parameter for this procedure are *l*, the changing step for the *interpolation error* value, and the R_{min}, lower bound for the correlation coefficient. This procedure starts applying the **RTSAL** algorithm with a relative high *interpolation error* value, it process the **MCT** output in order to reconstruct the signal (for example with a linear interpolation), and then it computes the correlation between the input signal and the reconstructed one. If this correlation is smaller than the value of R_{min}, it reiterates this procedure, once decreased the *interpolation error* of a value equal to *l*. When the correlation is appropriate, it returns the new data structure **M**'**C**'**T**'. A flow chart of this procedure is reported (Fig. 3.21).

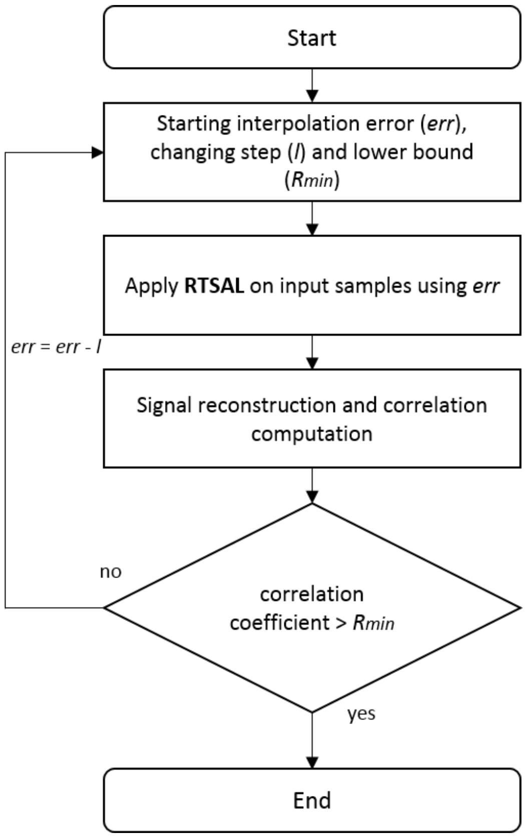

Start
Starting interpolation error (*err*), changing step (*l*) and lower bound (R_{min})
Apply **RTSAL** on input samples using *err*
err = err - l
Signal reconstruction and correlation computation
no
correlation coefficient > R_{min}
yes
End

Fig. 3.21 Flowchart for interpolation error selection.

3.5.4 User Defined Application Code

Producers or end users could want to apply their own algorithms or procedures, based on the **MCT** data structure. In the second layer some space is left, to allow to charge the custom code.

Chapter 4

RTSAL algorithm: characterization and microcontroller implementation

The segmentation and labeling algorithm can be implemented in different ways. Several simulations, as those shown in the previous chapter, are carried out implementing the algorithm on a fixed time window and saving the central sample of the segmentation interval, the sample for which the interpolation error threshold is exceeded. In this chapter, different possible implementation strategies are analyzed for this algorithm, characterizing it in terms of data size reduction, execution time, reconstruction capabilities, harmonic distortion. Moreover, first layer algorithms are shown, developed to run in real time on a microcontroller.

4.1 Comparison among different strategies

This algorithm, as already seen in the previous chapter, segments input signal samples, depending on a threshold level and on the trend of the specific signal. Different choices and strategies could be made and planned on the practical implementation of this algorithm, in particular on the selection of the sample to be stored:

1) "*saving the final sample of the segmentation interval*" is the implementation strategy which favors the execution time. This implementation is preferable when the sampling frequency is really higher than the acquired signal frequency (condition called "*oversampling*"), since the algorithm saves the final sample of the interval in the **MCT** data structure. The implementation is simple, and it requires less operation

compared to the other two, but it is also the less accurate. In the following tests, this implementation is referred as **testMCT2** (Fig. 4.1).

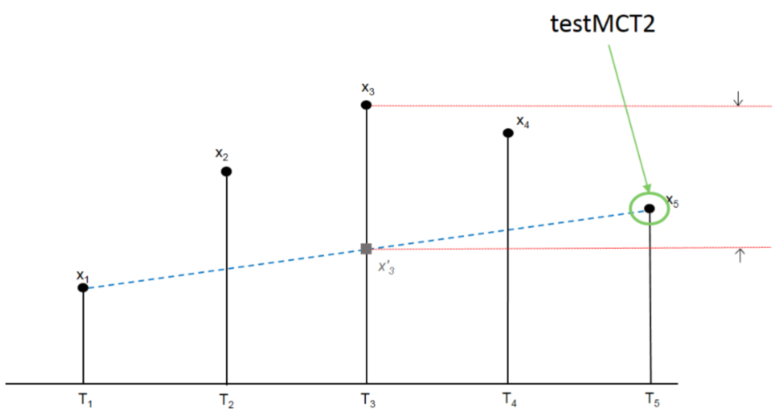


Fig. 4.1 The RTSAL algorithm saves the final sample of the analyzed interval.

2) "*saving the central sample of the segmentation interval*" is the implementation strategy which stores the middle sample, the one that exceeds the threshold limit. The choice of storing this sample requires reanalyzing some samples, restarting the algorithm from this sample. In the following tests, this implementation is referred as **testMCT3** (Fig. 4.2).

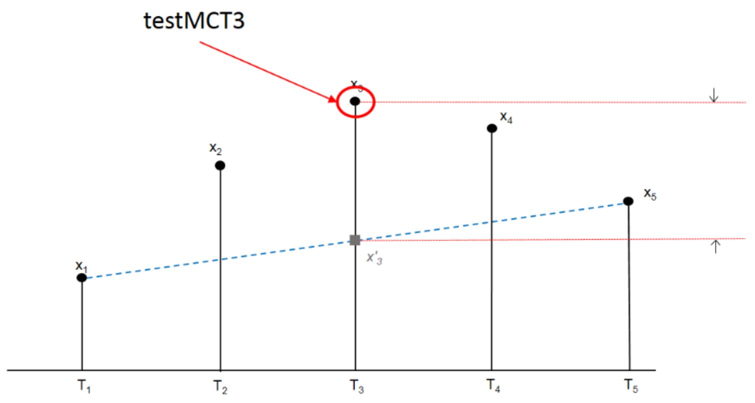


Fig. 4.2 The RTSAL algorithm saves the central sample of the analyzed interval.

3) "*double comparison*" is the implementation strategy which makes two comparisons for each iteration. In either previous cases, several signal samples are never compared with the linearized signal (for example, in Fig. 3.3, if sample x_3 is stored, the fourth sample is never taken into account). If the sampling frequency is not so higher than the signal frequency, many important samples could be missed: this technique is better for acquired signals for which all the samples could have a significant informative content. This technique implements a second comparison in the second half of the comparison range. In the following tests, this implementation is referred as **testMCT4** (Fig. 4.3).

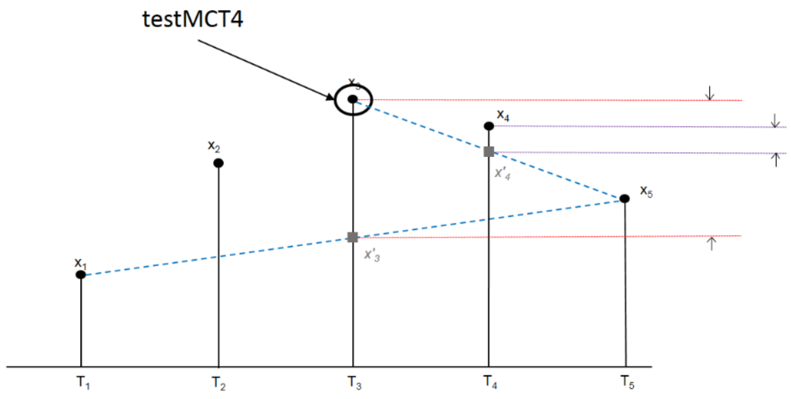


Fig. 4.3 The RTSAL algorithm makes two comparisons for each interval.

In this part of the thesis work, a preliminary analysis of each strategy is tackled, highlighting properties, advantages and disadvantages. The comparison is done in Matlab® simulations fixing the input signal, implying the simulated acquired samples are the same (the goal is to compare the algorithm) and fixing the interpolation error for all the different strategies. In Fig. 4.4 the input signal is shown in blue, a sinewave of 1 kHz, declared with 1000 samples for each period (to emulate a sampling frequency of 1 MHz), and a signal to noise ratio of 30 dB (additive white Gaussian noise). Moreover, three different signals are shown, which are reconstructed from the output **MCT** data structures; these signals are reconstructed using a linear interpolation between consecutive segments of the **MCT** data structures, using the same time step of the input signal. The signal, reconstructed linearly interpolating output **MCT** data structure segments of the strategy called **testMCT2**, is shown in green; the signal, reconstructed linearly interpolating output **MCT** data structure segments of the strategy called **testMCT3**, is shown in red; the signal, reconstructed linearly interpolating output **MCT** data structure segments of the strategy called **testMCT4**, is shown in black.

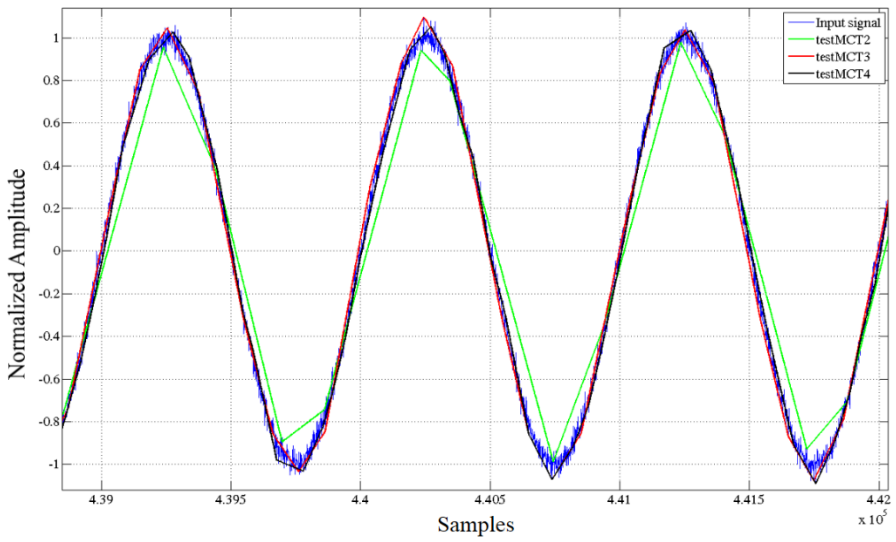


Fig. 4.4 Comparison among different strategies of **RTSAL**.

In the following paragraphs, several results are shown in order to compare these three processing strategies.

4.1.1 Reconstruction

In this paragraph, a comparison among reconstructing capabilities of these three strategies is shown, varying the interpolation error value. The main goal of the **RTSAL** algorithm, as already said, is to select only meaningful samples, from which it is possible to extract basic information; it is not the best technique if the final goal is to reconstruct the signal. Therefore, this analysis is done to compare results among different implementation strategies.

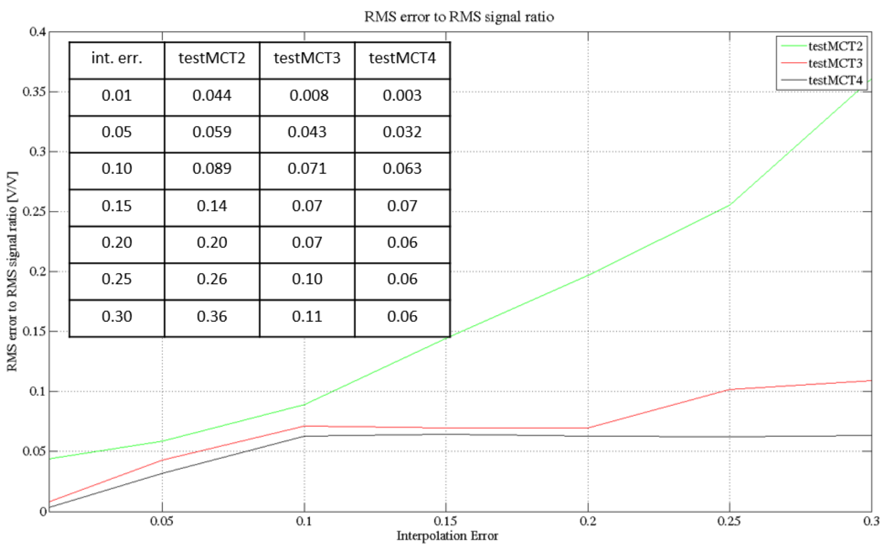
Defining, as *reconstruction error*, the difference between the input signal and the signal reconstructed from the **MCT** data structure:

$$e = x_{in} - x_{ri} \tag{10}$$

the ratio between the root mean square value of this error, and the root mean square value of the input signal is computed, as an estimator of the reconstruction capabilities of these techniques.

$$R = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_{in}-x_{ri})^2}}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}{x_{in}}^2}} \tag{11}$$

In Fig. 4.5, values computed for this ratio are graphed, as a function of the interpolation error value.

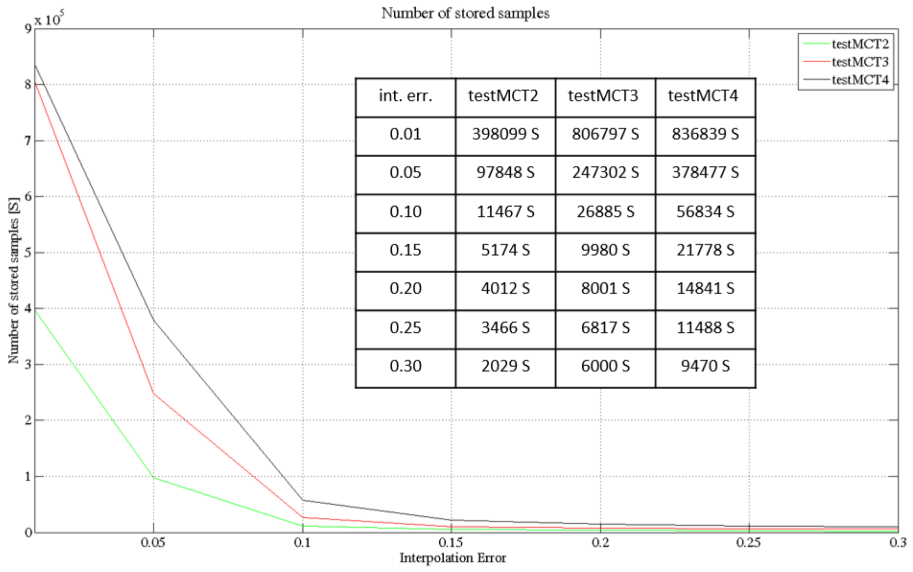


int. err.	testMCT2	testMCT3	testMCT4
0.01	0.044	0.008	0.003
0.05	0.059	0.043	0.032
0.10	0.089	0.071	0.063
0.15	0.14	0.07	0.07
0.20	0.20	0.07	0.06
0.25	0.26	0.10	0.06
0.30	0.36	0.11	0.06

Fig. 4.5 RMS reconstruction error to RMS signal ratio, as a function of the interpolation error value.

4.1.2 Memory requirements

The **RTSAL** algorithm aims to reduce necessary samples to extract specific information from the input signal. These three different implementation strategies store a different amount of input samples, with the same interpolation error. In Fig. 4.6 the stored samples number is graphed for the three strategies, as a function of the interpolation error.



int. err.	testMCT2	testMCT3	testMCT4
0.01	398099 S	806797 S	836839 S
0.05	97848 S	247302 S	378477 S
0.10	11467 S	26885 S	56834 S
0.15	5174 S	9980 S	21778 S
0.20	4012 S	8001 S	14841 S
0.25	3466 S	6817 S	11488 S
0.30	2029 S	6000 S	9470 S

Fig. 4.6 Comparison among different strategies on the number of stored samples (the same input signal as in Fig. 4.4, input vector of 10^6 samples).

In order to better understand what this number of samples means, in the perspective of the **MCT** data structure memory requirements, it is necessary to underline that one sample stored in this structure requires more memory than a normal sample, because it is necessary to store its *class* and *time position* too. As mentioned in paragraph 3.3, if two bytes are dedicated to the output value of the analog to digital converter of the smart sensor, one element of the *mark* vector also needs two bytes. In addition, one element of the *class* vector needs one byte, and one element of the *time* vector needs four bytes (saving the time stamp as millisecond from midnight). Therefore, storing a sample in this data structure means to allocate seven bytes instead of two. In the following tables (Tab. 4.1 and 4.2) memory requirements are shown for the same simulation of Fig. 4.6. In Tab. 4.1 memory allocation is shown as a function of the interpolation error, whereas, in Tab. 4.2, the percentage of memory allocated, compared to the memory required to store the entire input signal of one thousand samples, is shown.

Tab. 4.1 Memory requirements comparison

int. err.	**testMCT2**	**testMCT3**	**testMCT4**
0.01	2.78 MB	5.64 MB	5.85 MB
0.05	684.9 kB	1.73 MB	2.65 MB
0.10	80.2 kB	188.2 kB	397.8 kB
0.15	36.2 kB	69.8 kB	152.4 kB
0.20	28.0 kB	56.0 kB	103.9 kB
0.25	24.2 kB	47.7 kB	80.4 kB
0.30	14.2 kB	42.0 kB	66.3 kB

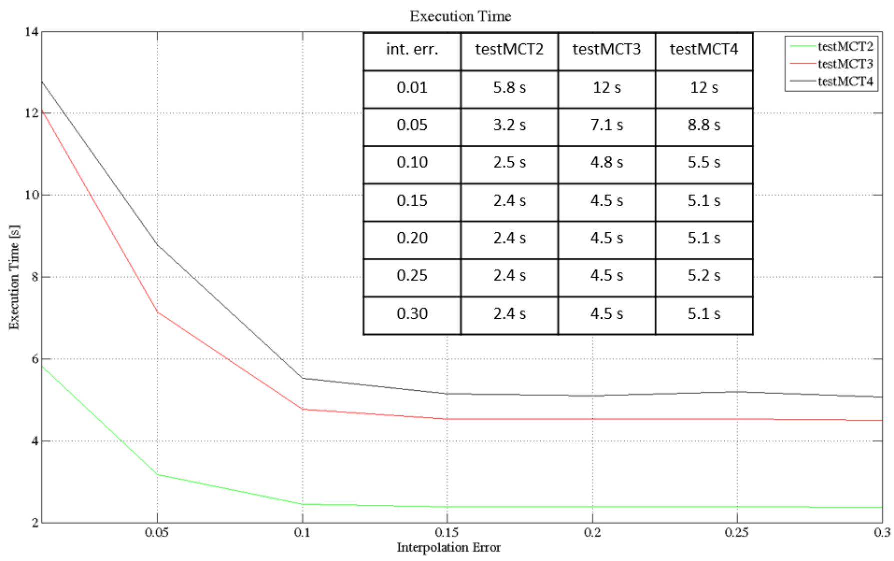
Tab. 4.2 Percentage memory requirements comparison

testMCT2	**testMCT3**	**testMCT4**
139.33%	282.38 %	292.89 %
34.24 %	86.56 %	132.47 %
4.01 %	9.41 %	19.89 %
1.81 %	3.49 %	7.62 %
1.40 %	2.80 %	5.19 %
1.21%	2.39 %	4.02 %
0.71%	2.10 %	3.31 %

Applying this segmentation and labeling algorithm is not always convenient: in fact, using a small interpolation error value will lead to store many samples, allocating much memory than storing the acquired vector of samples, which would make this technique useless.

4.1.3 Execution times

These three different implementation strategies of the **RTSAL** algorithm have different complexity. Strategies **testMCT2** and **testMCT3** perform the same operations for each iteration: they acquire two samples, calculate linear interpolation, which means a sum and a division by two, and they compare the values, which implies a subtraction and a comparison. They differ in the management of the intervals: **testMCT3** starts from the last stored sample, so it analyzes, at least, twice the second half of the interval, whereas **testMCT2** always goes afterwards, starting from the last point. The strategy **testMCT4** does two interpolations and two comparisons for the same interval instead. A comparison of execution times of Matlab® simulations is reported in Fig. 4.7: this comparison is made on the same signal as the previous simulation, as a function of the interpolation error, and the simulations are executed on the same machine.



int. err.	testMCT2	testMCT3	testMCT4
0.01	5.8 s	12 s	12 s
0.05	3.2 s	7.1 s	8.8 s
0.10	2.5 s	4.8 s	5.5 s
0.15	2.4 s	4.5 s	5.1 s
0.20	2.4 s	4.5 s	5.1 s
0.25	2.4 s	4.5 s	5.2 s
0.30	2.4 s	4.5 s	5.1 s

Fig. 4.7 Comparison among different strategies on the execution time (the same input signal as in Fig. 4.4, input vector of 10^6 samples).

4.1.4 Frequency analysis

In order to understand the harmonic distortion, which this reconstructing technique introduces linearly interpolating stored segments, this analysis is carried out comparing the input signal spectrum, computed with the fast Fourier transforms, with spectra of reconstructed signals with these three different strategies. Signals are reconstructed, padding the interval between consecutive segments with values that lay on the linear interpolation, with the same interval as the input signal. Therefore, the reconstructed signals have the same number of points as the input signal. In the following figure (Fig. 4.8) fast Fourier transforms are shown.

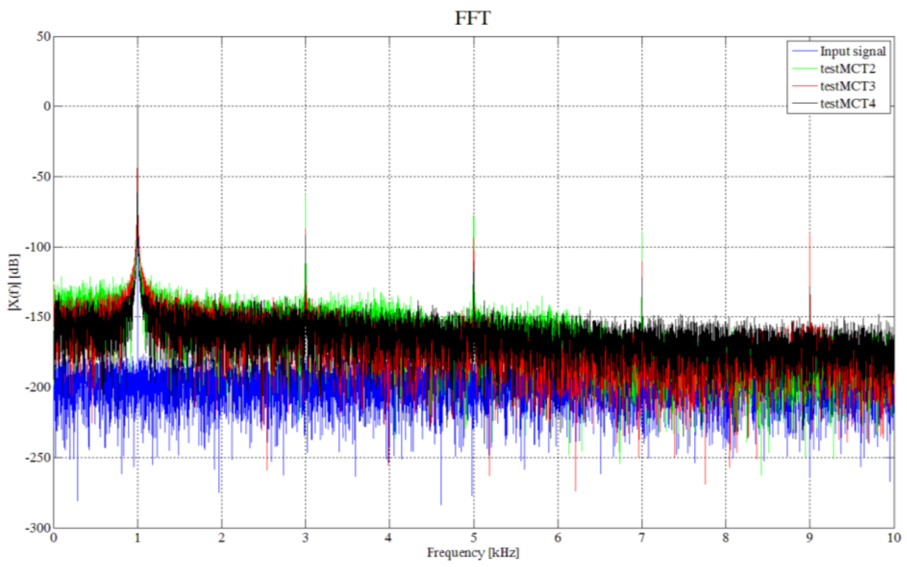


Fig. 4.8 FFT of the input signal (blue) and FFT of signals reconstructed linearly interpolating output segments of the **testMCT2** (green), **testMCT3** (red), **testMCT4** (black).

Simulations result in an attenuation of more than 3 dB of the fundamental harmonic for **testMCT2**, 0.4 dB for **testMCT3**, and a gain of almost 0.05 dB for **testMCT4**. On the other hand, there is a distortion effect of this technique, due to the linear interpolation, which introduces odd harmonics. The third harmonic of the signal reconstructed from the **testMCT2** segments is at -60 dB, the one of the signal reconstructed from the **testMCT3** segments is at -87 dB, and the one of the signal reconstructed from the **testMCT4** segments is at -92 dB. Moreover, the floor level of the reconstructed signals differ significantly from the input signal one: the floor level is lower than -170 dB for the input signal, whereas, for the three different approaches, the floor level is around -120 dB near the first harmonic, and it decreases for bigger frequencies. In addition, even the main lobe width is bigger than the main lobe of the input signal transform.

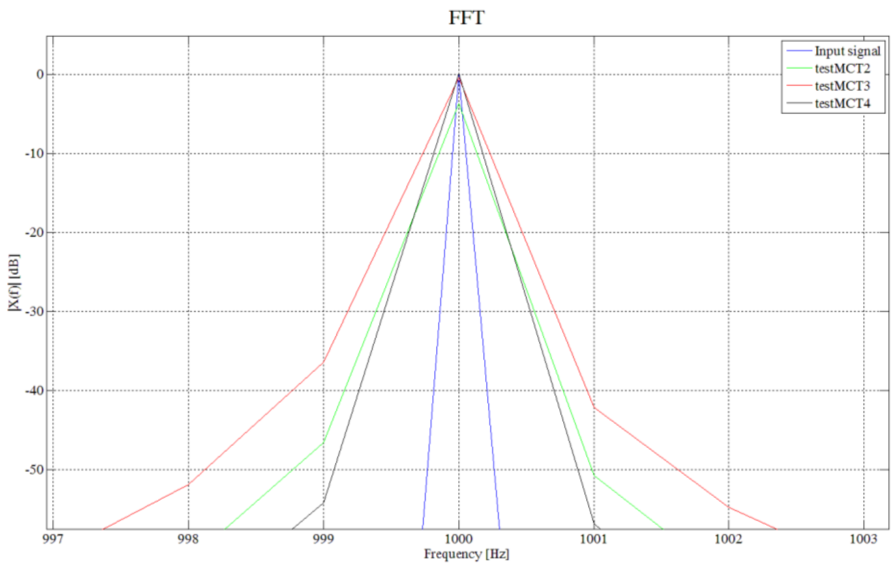


Fig. 4.9 FFT of the input signal (blue) and FFT of signals reconstructed linearly interpolating output segments of the **testMCT2** (green), **testMCT3** (red), **testMCT4** (black) around 1 kHz.

Summarizing, the strategy called **testMCT2** stores fewer samples, so it requires less memory allocation or fewer data to send; it is also faster, in terms of execution time, but it is also the worst in terms of accuracy. On the other hand, **testMCT4** is the more accurate strategy, but it pays this accuracy with a more complex strategy than the other ones, storing more samples and requiring more execution time. Eventually, **testMCT3** has been considered the best solution among them, because it gives a better accuracy than the **testMCT2** strategy, without increasing the computational and the memory requirements as the **testMCT4** does. Who wants to use the **RTSAL** algorithm can choose the best technique and the proper interpolation error value for the specific input signal. In the following, the **testMCT3** strategy is used in all cases.

4.2 Implementing RTSAL algorithm on a microcontroller

In this paragraph a feasible implementation strategy of the **RTSAL** algorithm and of the first layer algorithms, based on the **MCT** data structure, is proposed, in order to implement this technique on a microcontroller. The goal, implementing these algorithms on a microcontroller, is to test real time processing capabilities: in fact, the main reason for these strategies is to implement basic standardized processing capabilities on smart transducer, which is based on a microcontroller. This implementation is developed for an STM32F103, an ARM Cortex-M3 based microcontroller, which integrates the analog to digital conversion, the **MCT** data structure development, and the first layer algorithms processing strategies. In the following, several algorithms are shown, developed in C code with Keil [60]; they are tested using a development board from ETT.

The **RTSAL** algorithm in this implementation differs, in samples and memory management, from those implemented in Matlab® simulations (referring to the **testMCT3** implementation, shown in paragraph 4.1). In simulations, it takes a vector of samples of a fixed size, and then it processes samples; whereas, in this implementation on microcontroller, the function that has the task to segment and store samples is called once each two acquired samples. It manages samples in a manner of keeping them into a temporary memory, until the storage of a new segment in the **MCT** data structure. When a new segment is stored, the segmentation step, for the next segment search, begins with an interval equal to half the previous one, plus the following one or two new acquired samples, depending on the size of the last interval. In fact, to compute the linear interpolation value between samples on the edge of the interval, and compare this value with the acquired value for that sampling time, an interval of an odd number of samples is needed. This microcontroller has a 12-bit ADC converter: the output value of the converter is not converted in any format, and the interpolation error is given as a number of levels.

Several algorithms are called every time a new segment is stored in the **MCT** data structure, whereas others, which must be applied on more segments, are called when the dedicated buffer is complete. In order to process data in the buffer, without stopping acquisition, the storage of the **RTSAL** algorithm has been implemented with a double buffer structure (Fig. 4.10).

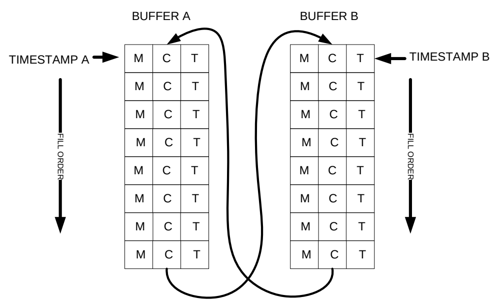


Fig. 4.10 Buffer structure for MCT data structure.

4.2.1 Exponential Pattern Detection

The exponential pattern detection function is developed with a state machine structure, as shown in Fig. 4.11. The goal of this function is to count the number of consecutive segments of the same class. This function is called each time a new segment is stored. The starting case is state 0: the algorithm is in this state if the class of the last segment differs from the class of the previous one, or if there is an ‘***a***’, ‘***b***’, ‘***c***’ or ‘***h***’ class. The state changes to one when it finds a segment of ‘***g***’, ‘***f***’, ‘***e***’ or ‘***d***’ class, and if the class of the new segment is equal to the class of the previous one. In this manner is possible to count consecutive segments with the same class, and manage simple operations.

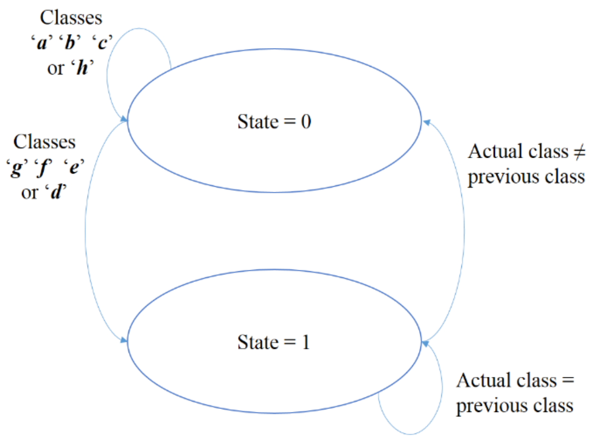

Classes ‘a’ ‘b’ ‘c’ or ‘h’
State = 0
Classes ‘g’ ‘f’ ‘e’ or ‘d’
Actual class ≠ previous class
State = 1
Actual class = previous class

Fig. 4.11 State machine of the Exponential Pattern Detection algorithm.

4.2.2 Noise Detection

The noise detection function is developed with a state machine structure, as shown in Fig. 4.12. This function detects noise computing the distance between maxima and minima. This function is called each time a new segment is stored, and the code related to the actual state is executed. Starting from state 0, after the first segment the algorithm goes into state 1 or 2, depending on the class, where it stays until the first minimum or maximum is found. Once found the first, it continues to look for minima and maxima and to compute time interval between two consecutives of themselves, in state 10 and 20. If this time interval is smaller than the *NoisyPeriod* for more than *AmountNoise* time, the signal is considered noisy. If a class ‘***a***’, ‘***b***’, ‘***c***’ or ‘***h***’ is found, the algorithm returns in state 0.

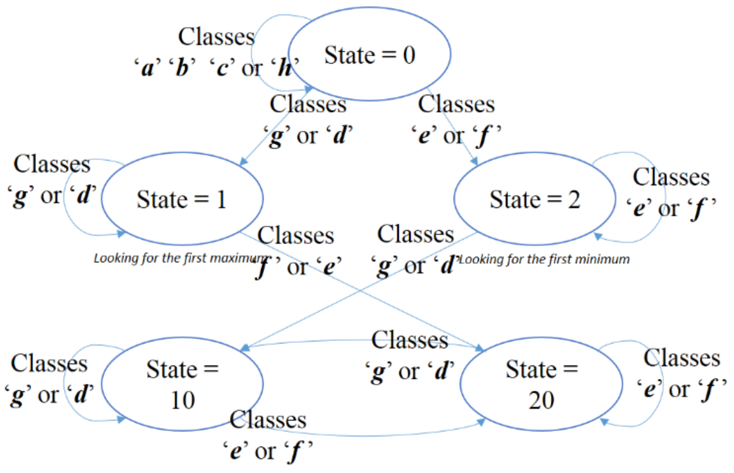


Fig. 4.12 State machine of the Noise Detection algorithm.

4.2.3 Sinusoidal Pattern Detection

The sinusoidal pattern detection function is developed with a state machine structure, as shown in Fig. 4.13, which allows the detection of the pattern of classes ‘***g***’-‘***e***’-‘***f***’-‘***d***’. This function is called each time a new segment is stored, and the code related to the actual state is executed. Starting from state 0, if one of these classes is found, the algorithm changes the state variable, in one state related to this class (state 1 stands for class ‘***g***’, state 2 for class ‘***e***’, state 3 for class ‘***f***’ and state 4 for class ‘***d***’). When it is in one of these four states, and if the following segment is of the same class as the previous one, the state is still the same. If the following segment is the one that is expected for this specified pattern, the state changes to the following one, whereas if the next segment class differs from the previous class or the expected one, the state returns to 0. For example, if a class ‘***g***’ segment is followed by another ‘***g***’ segment, the state remains the same; if it is followed by an ‘***e***’ class segment, the state changes from state 1 to state 2, whereas if it is followed by one of the other possible classes, the state returns to 0. In this algorithm are implemented all the functionalities already described in the paragraph 3.4.4.

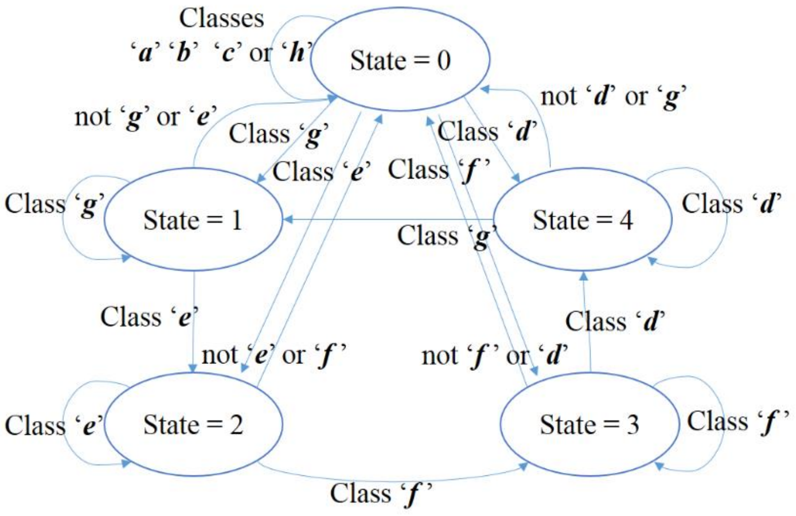


Fig. 4.13 State machine of the Sinusoidal Pattern Detection algorithm.

4.2.4 Tendency Estimation

The Tendency Estimation function operates only on maxima and minima of the signal, in a specific time window. So, this algorithm has to extract only these values, and to apply the formula for the Tendency Estimation (see par. 3.4.5), taking into account only maxima and minima included in the specified time window. This function is developed with a state machine structure, as shown in Fig. 4.14, which allows the detection of maxima and minima. This function is called each time a new segment is stored, it saves maxima and minima in a separate buffer, and it computes tendency parameters. It starts with state 0, and it returns to this state only in case of classes ‘***a***’, ‘***b***’, ‘***c***’ or ‘***h***’.

Other two states, state 1 and 2, are for maxima and minima search respectively.

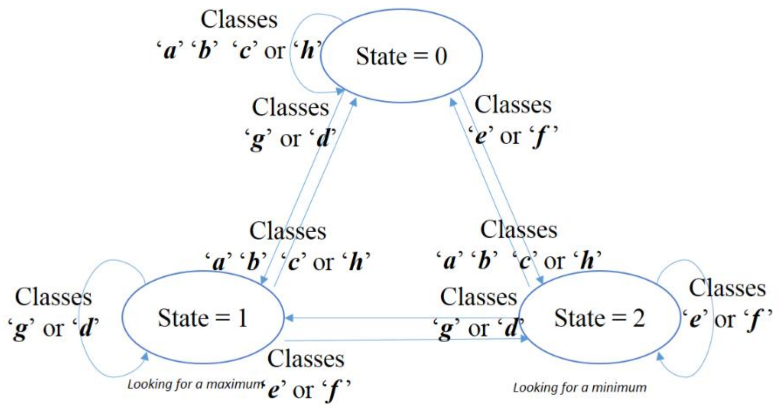


Fig. 4.14 State machine of the Tendency Estimation algorithm.

4.2.5 Mean Estimation

The mean estimation function is called each time one of the two buffers of the data structure for the **MCT** vectors is complete. It takes this entire structure and computes an estimation of the mean, using a set of samples which are not separated by the same time interval (Fig. 4.15). The **MCT** data structure gives amplitudes and positions, therefore, it is possible to compute an estimation of the mean, basing on the trapezoidal area computation formula (12).

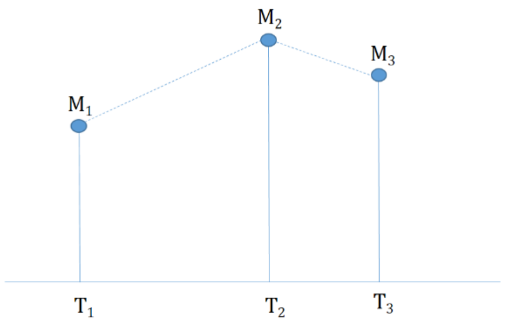


Fig. 4.15 Three samples stored by the **RTSAL** algorithm.

$$\bar{x} = \frac{1}{2(T_N - T_1)} \sum_{i=1}^{N-1} (M_{i+1} + M_i)(T_{i+1} - T_i) \quad (12)$$

Chapter 5

Applications of a RTSAL technique

The **RTSAL** algorithm can be used in different applications: in this chapter several uses of **MCT** data structure, generated with this algorithm, are shown. This technique is used in order to extract several parameters from the analyzed signals, like the period, the amplitude, the mean, and so on. The thesis shows results of these algorithms compared, in simulations and on real signals, with other techniques. Comparisons in simulations are done in Matlab®, and those algorithms are charged on the oscilloscope, in order to test same algorithms with acquired samples. This chapter consist of three main parts:

- an in-depth properties examination of the period measurement capability; the working principle of the algorithm and problems related to the maximum search, based on the **MCT** data structure are analyzed. Moreover, results of simulations are shown for different signals, varying the interpolation error value, sampling frequency and signal to noise ratio. Therefore, these strategies are implemented utilizing an on board Matlab® on an oscilloscope. In addition, this technique is also compared with the *compressive sensing*. Furthermore, this algorithm is tested on microcontroller, in order to test functionality and execution times.
- an analysis of mean estimation algorithm results.
- the **RTSAL** algorithm is exploited to analyze vibrations signals from faulty bearing.

5.1 Period Measurement

A period is the time duration of one cycle in a repeating event. Searching through the class vector, it is possible to identify maxima and minima as a determined sequence of classes: a maximum can be located when there is a ‘**g**’ or ‘**d**’ class, followed by an ‘**e**’ or ‘**f**’ class, whereas a minimum can be located when there is an ‘**e**’ or ‘**f**’ class, followed by a ‘**g**’ or ‘**d**’ class. The period can be determined as the time interval between the position of two following maxima (Fig. 5.1), whose amplitude values fall in the same range of twice the interpolation error (to avoid coarse errors due to higher order harmonics or noise).

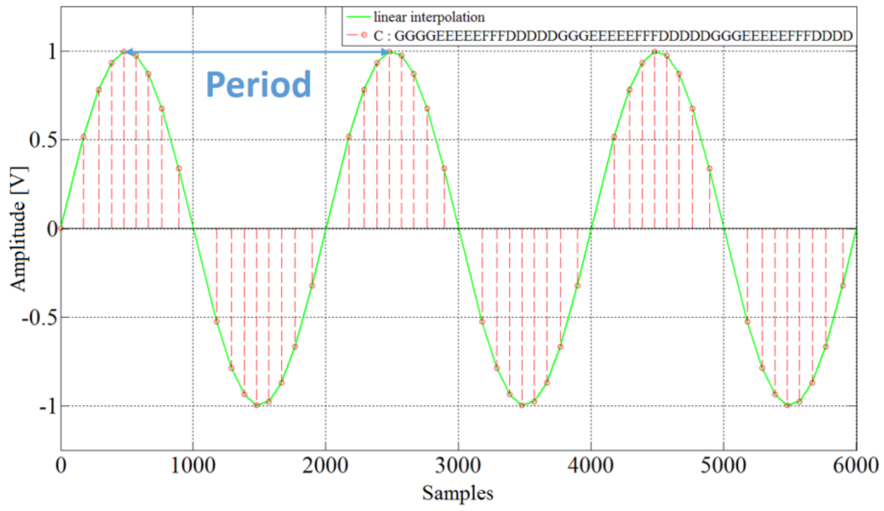


Fig. 5.1 Evaluation of period on a signal processed with the real time segmentation and labeling algorithm.

This means that, by the interpolation error value, the sensitivity to the amplitude variations of the signal can be regulated and, as a consequence, the number of samples saved in the structure of the three arrays. For small values of interpolation error, comparable to the noise amplitude, the position of signal maxima on the time axis can be characterized by jitter. Moreover, many input samples would be stored, increasing the computational load and requiring more memory. On the contrary, increasing the interpolation error, the noise susceptibility decreases, the number of stored samples decreases, together with the number of stored samples. A simple flow chart of the algorithm for period computation is shown in Fig. 5.2: inputs are the vector of the **MCT** data structure, and the interpolation error value (E), whereas outputs are period and maximum amplitude values. The resolution is related to the sampling frequency, whereas the precision of this type of algorithm is also related to the chosen interpolation error, which depends on the signal amplitude.

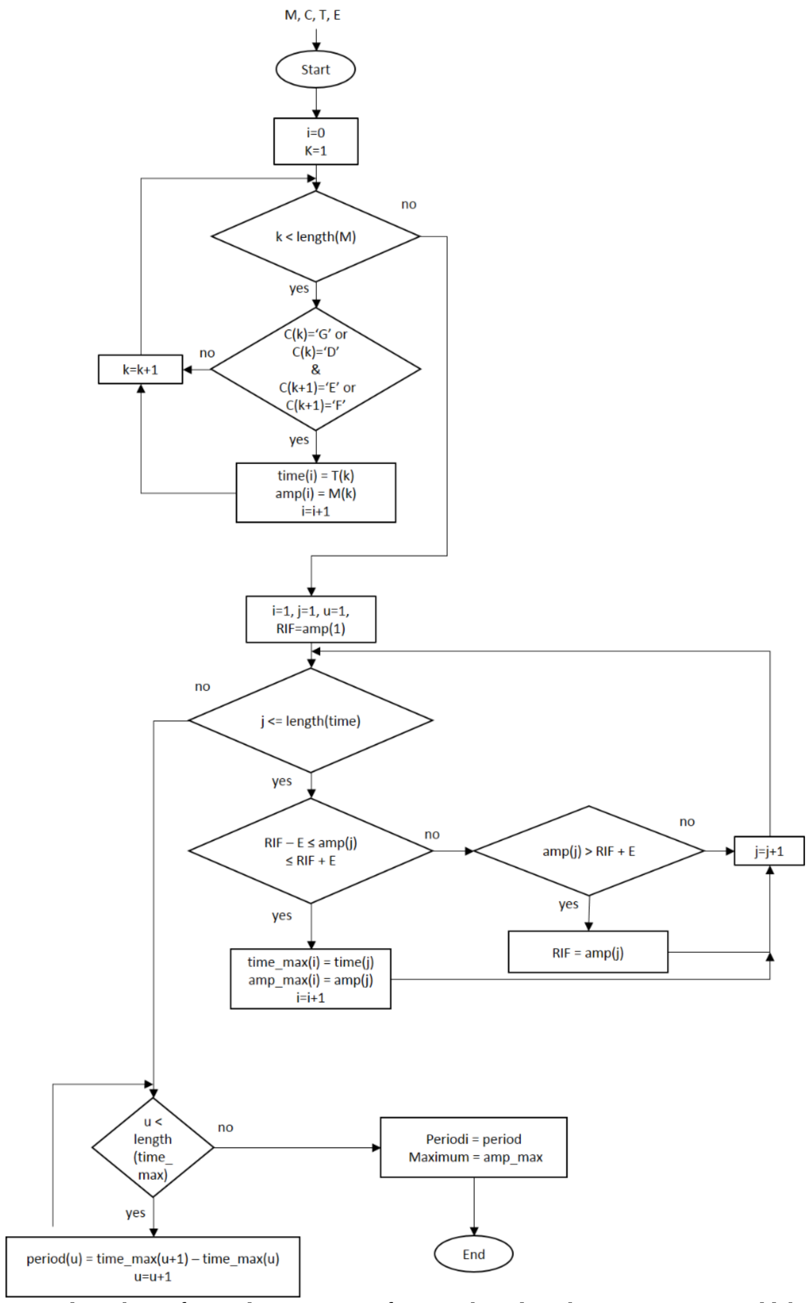


Fig. 5.2 Flow chart of period computation function based on the segmentation and labeling algorithm.

5.1.1 Maximum search

Several simulation tests were carried out in order to evaluate the residual error and the uncertainty in the period evaluation. To this aim at first the uncertainty in the maximum position identification is quantified. In the following, several simulations will be described, implemented on a personal computer with Matlab® software. For example, different cases of maximum search are shown in the following graph (Fig. 5.3-5.4). Simulations employ the same maximum research and a signal is declared with the same sampling frequency. In Fig. 5.3 are reported, for the same ideal input signal, in the same observation window, the ideal maximum position, the maximum position identified by the algorithm, and the related error, in number of samples, with different interpolation error values. In Fig. 5.4 are reported, for the same input waveform, in the same observation window, the ideal maximum position, the maximum position identified by the algorithm, and the related error, with different interpolation error. Additive noise of 30 dB has been considered by means of the Matlab® command *awgn*.

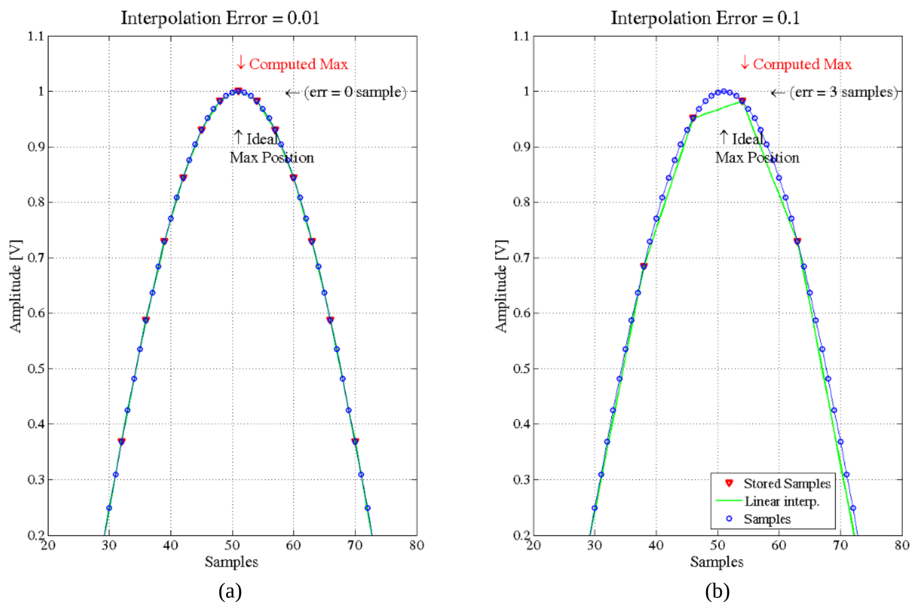
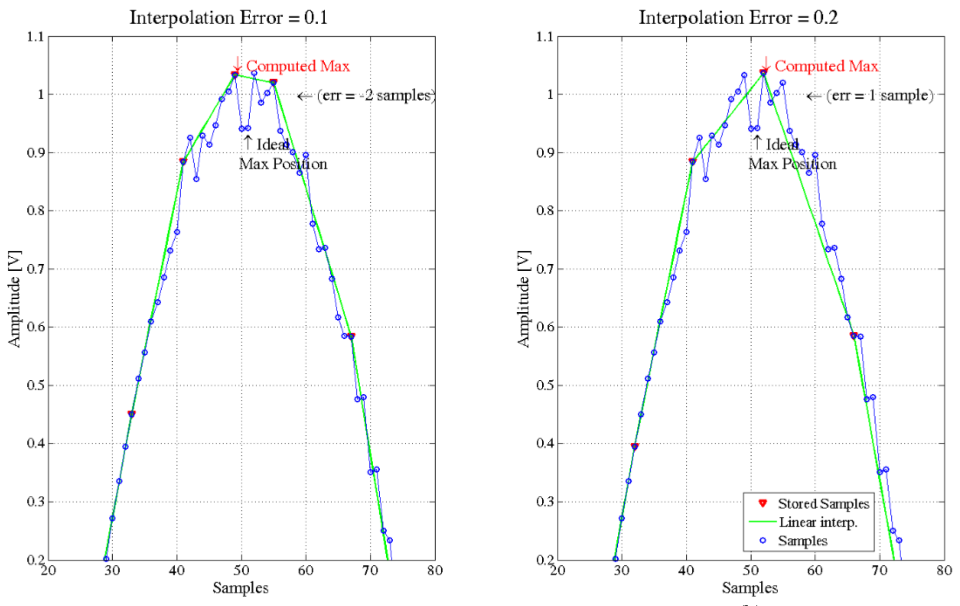


Fig. 5.3 Maximum search, with interpolation error of 0.01 (a), and 0.1 (b). The input samples of an ideal sinusoidal waveform are in blue, whereas stored segment are in red (classes are not reported).



(a) (b)

Fig. 5.4 Maximum search, with interpolation error of 0.1 (a), and 0.2 (b). The input samples of a sinusoidal waveform with a signal to noise ratio of 30 dB are in blue, whereas stored segment are in red (classes are not reported).

In the maximum search using real time segmentation and labeling algorithm, many parameters could influence the result accuracy: the interpolation error value, the starting point of segmentation, the signal to noise ratio, the sampling frequency, and the signal amplitude.

In the following graph, results are reported, obtained plotting mean and standard deviation of the relative error done identifying the maximum position, varying the interpolation error value. The input signal is a sinusoidal waveform of unitary amplitude, and the interpolation error value is considered as percentage value of the input sine amplitude. The mean and standard deviation of the relative error are computed, varying the interpolation error with a step of 0.05, on one hundred repeated simulations (N=100), for which is randomly varied, the starting point for the segmentation algorithm, in a neighborhood of the minimum equal to the fifteen percent of the input sine period. The input sine has a frequency of 50 Hz, with a signal to noise ratio of 30 dB, whereas the sampling frequency is equal to 100 kHz. In Fig. 5.5 results are shown.

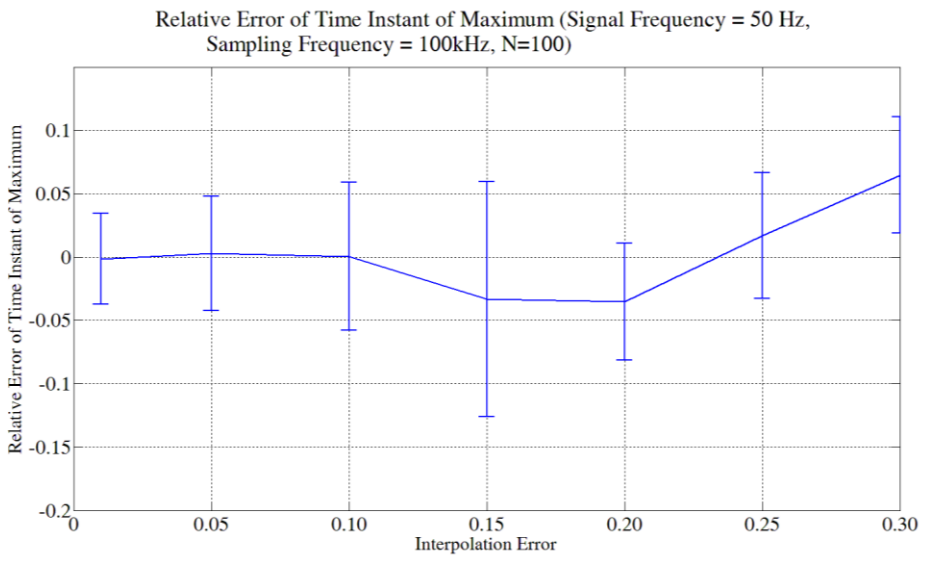


Fig. 5.5 Mean and standard deviation of relative error computing the position of the maximum of an input sinusoidal waveform of 50 Hz (signal to noise ratio 30 dB, sampling frequency 100 kHz). Results are shown as a function of the interpolation error, evaluated on 100 repeated simulations for each step.

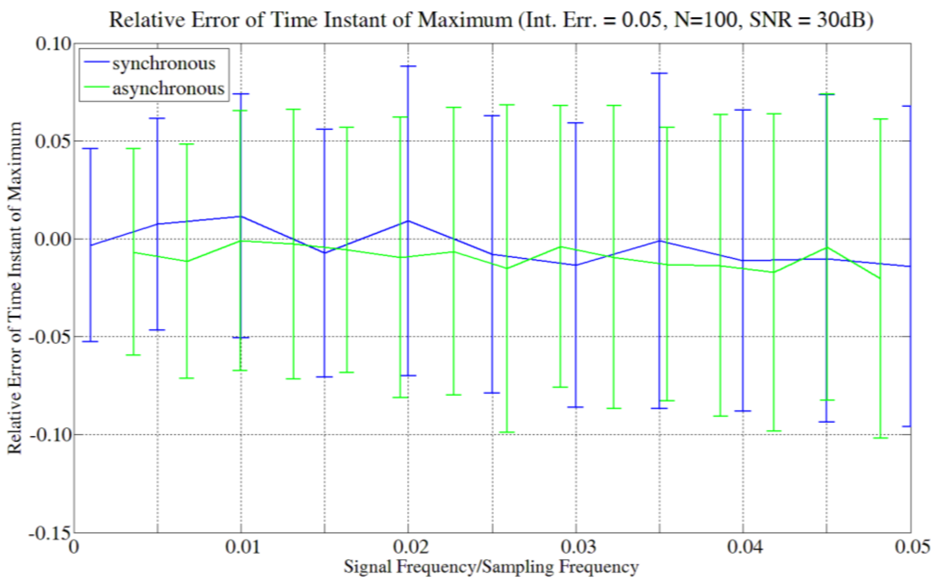


Fig. 5.6 Mean and standard deviation of relative error computing the position of the maximum of an input sinusoidal waveform (signal to noise ratio 30 dB, interpolation error 0.05). Results are shown as a function of the signal frequency to sampling frequency ratio, evaluated on 100 repeated simulations for each step. Blue for synchronous sampling and green for asynchronous sampling.

Moreover, in order to show the effect of the ratio between sampling frequency and the input signal frequency for the same input signal, in Fig. 5.6 the mean and standard deviation of the relative error are shown, as a function of the ratio between sampling frequency and the input signal frequency. The interpolation error value for these simulations is 0.05; blue forks are for synchronous sampling frequencies, whereas green forks are for asynchronous sampling frequencies: results are not affected by an asynchrony between signal and sampling frequencies.

5.1.2 Simulation of period computation

The Matlab® software has been used for simulations that were made in order to evaluate residual error and uncertainty in period calculation, with reference to canonic waveforms as sine, square wave or multi-tone.

Based on the selection of the stored samples, and based on maxima found by searching through the classes of the class vector, it is possible to implement the period calculation as the difference of temporal position between the segments that identifies two consecutive maxima. Respect to other algorithms, like the "*zero crossing*", that need two trigger level crossing to estimate the signal period, the real time segmentation and labeling algorithm more easily returns the time positions of local maxima and minima. Of course, due to the null slope of maxima and minima, the proposed algorithm should be more susceptible to noise, compared with algorithms that can chose suitable trigger levels. Then the simulations have made with the aim of evaluating the influence of the interpolation error on the noise susceptibility of the period measurements based on **MCT** vectors. It has to be noted that, in this case, the quantity to be measured is the time distance between two consecutive maxima. As a further explanation, in Fig. 5.7 same cases of the segmentation process on a periodic waveform, with different interpolation error values, are shown. As a result, for a determined signal period, the bigger the interpolation error, the fewer the MCT vector elements. However these results confirm that, even if the amplitude error increases, the signal period error decreases because the time interval between two consecutive maxima should be less affected by noise.

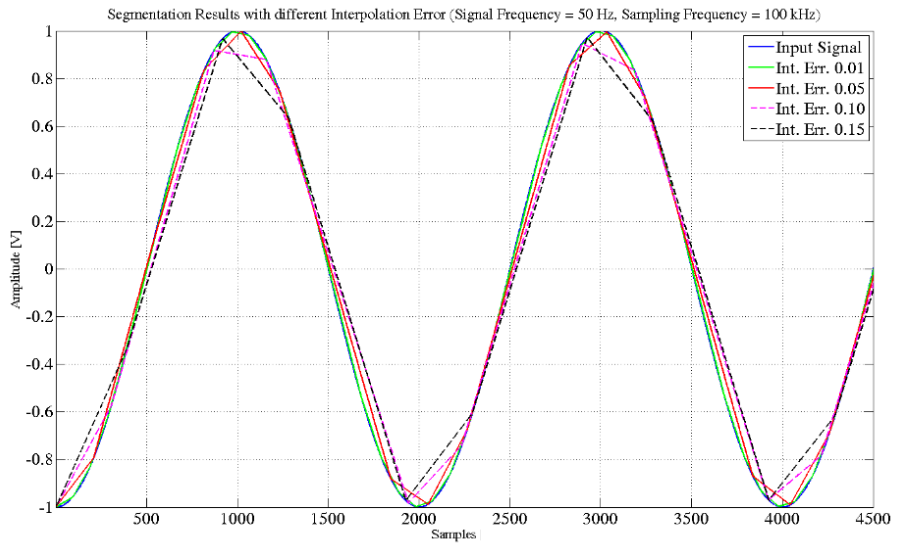


Fig. 5.7 Examples of segmented sinusoidal waveform with several different interpolation error values.

As it can be observed, even if a small interpolation error grants a more detailed waveform reconstruction in the time domain, it leads to higher errors in period calculation. As a matter of fact, a small interpolation error leads to higher sensitivity to noise in a way that depends on the signal to noise ratio. For example, different cases of period computation are shown in the following figures (Fig. 5.8-5.9). Simulations employ a signal of 50 Hz, which is declared with sampling frequency of 100 kHz and 30 dB of signal to noise ratio. In Fig. 5.8 an example of stored samples from this signal by the segmentation and labeling algorithm is reported, for an interpolation error of 8%, whereas, in Fig. 5.9, for the same input signal, an example of stored samples for an interpolation error of 20% is reported.

So, searching for maxima in the class vector, all the class sequences classified as maxima are taken into account, and those ones whose amplitude values fall in the same range of twice the interpolation error are used to compute the period. This segmentation and labeling technique acts to extract a global information from the signal: a right interpolation error choice is fundamental to obtain good results, both in terms of memory saving and global behavior understanding. The interpolation error value has to be used like the LSB of an ADC: with a small interpolation error (lower than the noise amplitude) the algorithm stores many samples, which do not contain significant information content, whereas, with a bigger one (value near the noise amplitude) it stores only meaningful samples in the **MCT** data structure, which are free from this redundancy.

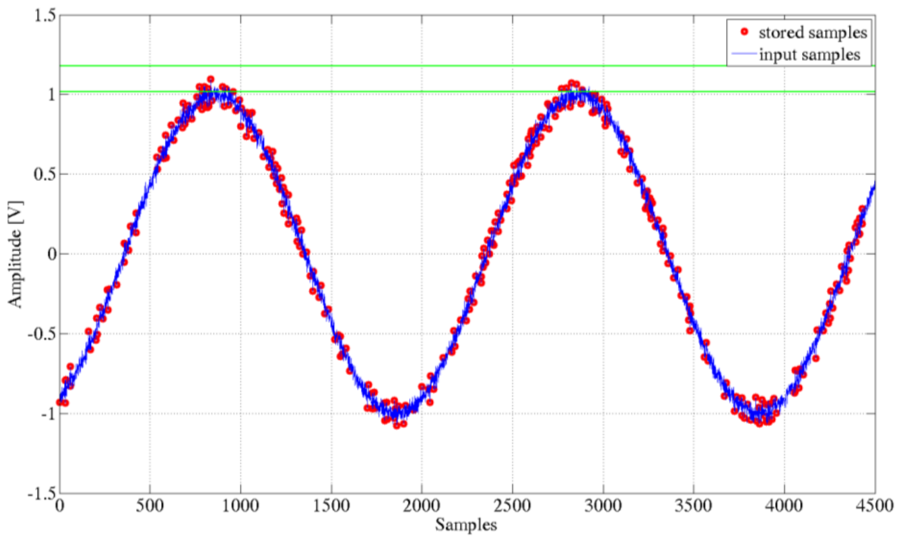


Fig. 5.8 Example of stored samples by the **RTSAL** algorithm. Input sinusoidal signal of 50 Hz, sampling frequency 100 kHz, SNR 30 dB, interpolation error 8%.

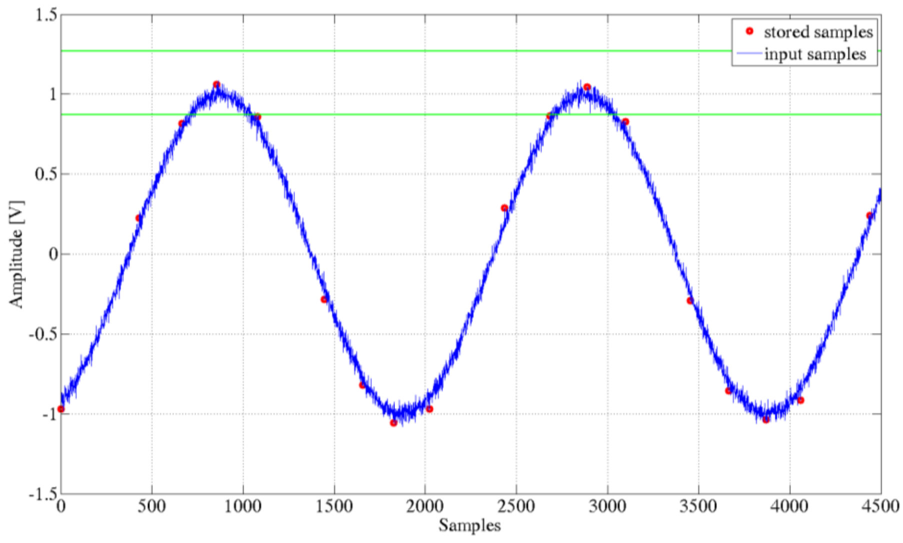


Fig. 5.9 Example of stored samples by the **RTSAL** algorithm. Input sinusoidal signal of 50 Hz, sampling frequency 100 kHz, SNR 30 dB, interpolation error 20%.

In the following figures, some results of the simulations are shown: as for in the computation of the period, the systematic error and its standard deviation versus the interpolation error are reported. Mean and standard deviation are computed, varying the interpolation error with a step of 0.01, in one thousand repeated simulations (N=1000), for which the starting point for the segmentation algorithm is randomly varied, in the neighborhood of the minimum equal to the 15% of the input signal. The following three figures are for different input signals: a sinusoidal signal (Fig. 5.10), a 2-tone signal (Fig. 5.11), and a square wave signal (Fig. 5.12). All the signals have a fundamental harmonic at 50 Hz, they are acquired with the same sampling frequency of 100 kHz, and have a signal to noise ratio of 30 dB.

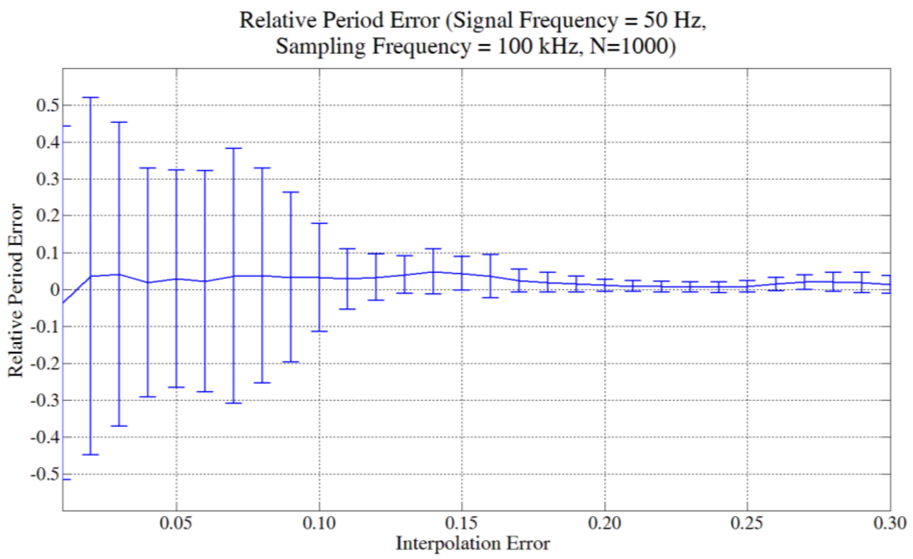


Fig. 5.10 Mean and standard deviation of relative error computing the period of an input sinusoidal waveform of 50 Hz (signal to noise ratio 30 dB, sampling frequency 100 kHz). Results are shown as a function of the interpolation error, evaluated on 1000 repeated simulation for each step.

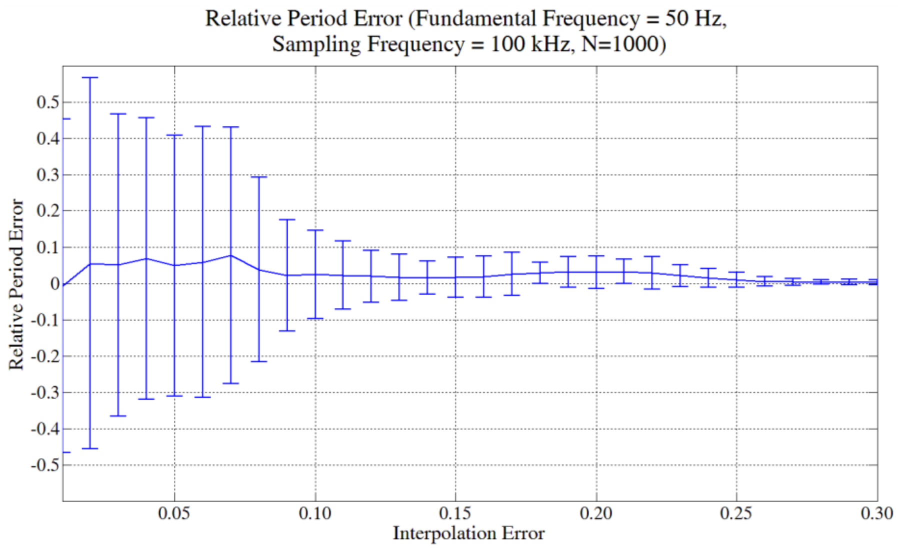


Fig. 5.11 Mean and standard deviation of relative error computing the period of an input two tone waveform, with fundamental of 50 Hz (signal to noise ratio 30 dB, sampling frequency 100 kHz). Results are shown as a function of the interpolation error, evaluated on 1000 repeated simulations for each step.

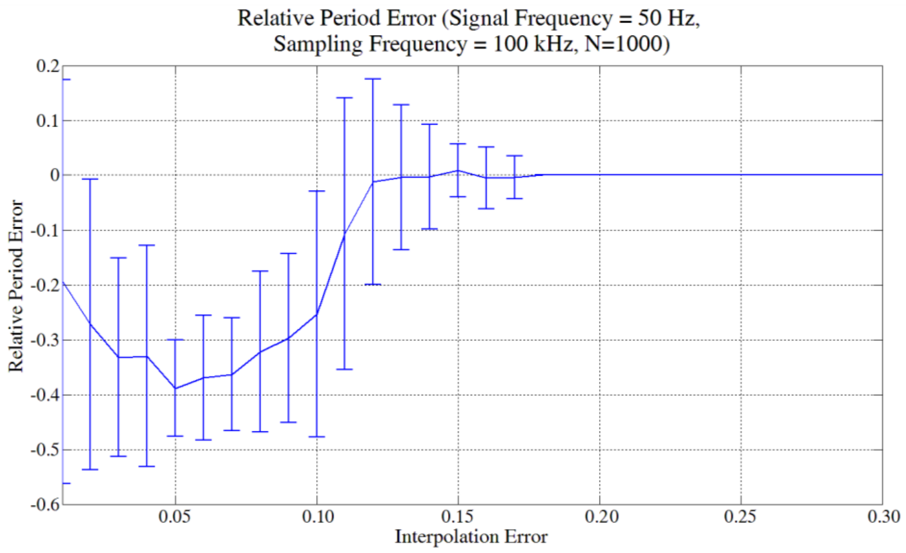


Fig. 5.12 Mean and standard deviation of relative error computing the period of an input square wave signal, with fundamental frequency of 50 Hz (signal to noise ratio of 30 dB, sampling frequency 100 kHz). Results are shown as a function of the interpolation error, evaluated on 1000 repeated simulations for each step.

Furthermore, an error analysis is made fixing the interpolation error value, and varying the sampling frequency. The interpolation error selected is equal to 20% of the maximum value: this value is selected based on the results of the previous simulations. Around this value, the error made calculating period is smaller than the error computed for other values for the analyzed input signal: this value has to be chosen for the specific application, and for the specific signal. The range of sampling frequencies analyzed is from 10 kHz to 1 MHz. The input signal is still a sinusoidal waveform: with a signal to noise ratio of 30 dB in the first figure (Fig. 5.13), and with a signal to noise ratio of 40 dB in the second figure (Fig. 5.14). In addition, in those simulations the starting point for the segmentation algorithm is varied randomly, in a neighborhood of the minimum equal to the fifteen percent of the input sine period. Means and standard deviations are computed on one hundred repeated simulations (N=100). In these figures, blue forks stand for synchronous sampling frequencies, whereas green forks stand for asynchronous sampling frequencies, in order to underline that results are not affected by an asynchrony between signal and sampling frequencies. Error results are shown in relative values, computed relating to the period of the input waveform, in order to simplify the reading. The error is smaller at higher sampling frequencies.

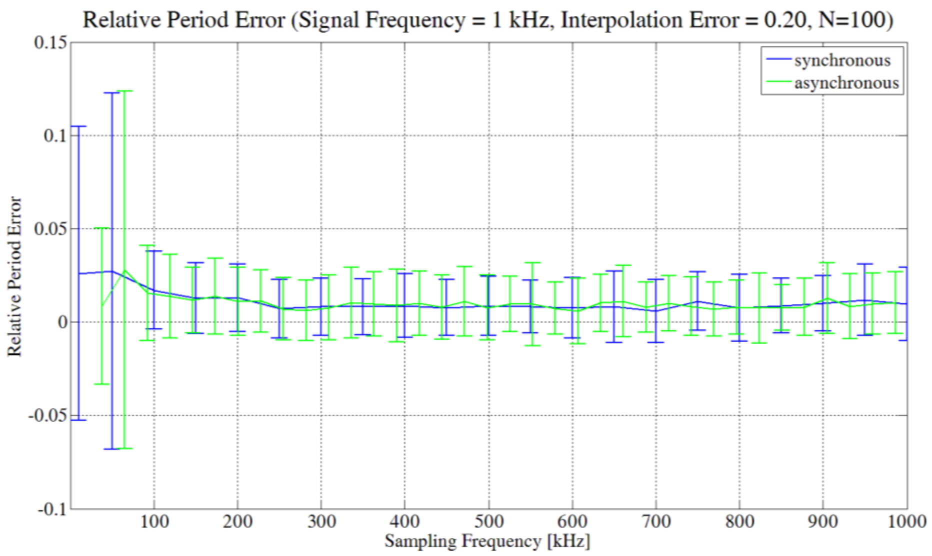


Fig. 5.13 Mean and standard deviation of relative error, computing the period of a sinusoidal waveform, with frequency of 1 kHz (signal to noise ratio 30 dB, interpolation error 20 % of the input maximum value). Results are shown as a function of the sampling frequency, evaluated on 100 repeated simulations for each step. Blue for synchronous sampling and green for asynchronous sampling.

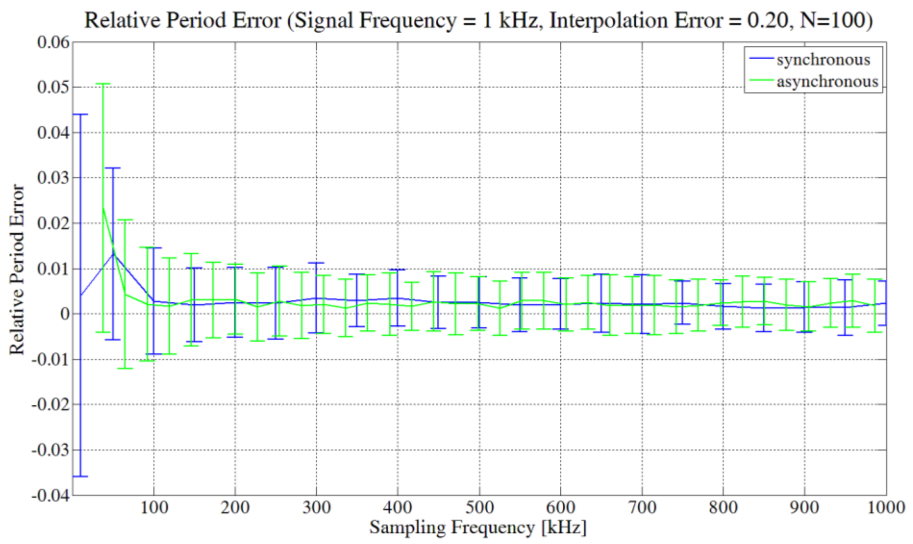


Fig. 5.14 Mean and standard deviation of relative error, computing the period of a sinusoidal waveform, with frequency of 1 kHz (signal to noise ratio of 40 dB, interpolation error 20 % of the input maximum value). Results are shown as a function of the sampling frequency, evaluated on 100 repeated simulations for each step. Blue for synchronous sampling and green for asynchronous sampling.

Every real signal is affected by noise, depending on the type of sensors and applications, varying the type and the amount of noise. Several simulations implemented in Matlab® have been done in order to evaluate the performances of this period computation based on the **RTSAL** algorithm as a function of the signal to noise ration. A sinusoidal waveform is stated on Matlab®, with a period of 0.02 s and a sampling period of 10 µs (2000 samples per period are taken) on a temporal window of 0.2 s. Fig. 5.15 shows the computed period error tendency related to the nominal period, as a function of the interpolation error value, and as a function of the signal to noise ratio for the input signal. The minimum interpolation error parameter to be chosen (for which good results for the period error are achieved), increases with signal to noise ratio decreasing. Furthermore, Fig. 5.16 and 5.17 show the period and the amplitude calculated as a function of the sampling frequency (from 1 kHz to 100 kHz), with the interpolation error parameter fixed for the **RTSAL** algorithm. The input signal for these two simulations is still a sinewave with a period of 0.02 s, on an observation temporal window of 0.04 s (in order to have only two maxima in this window).

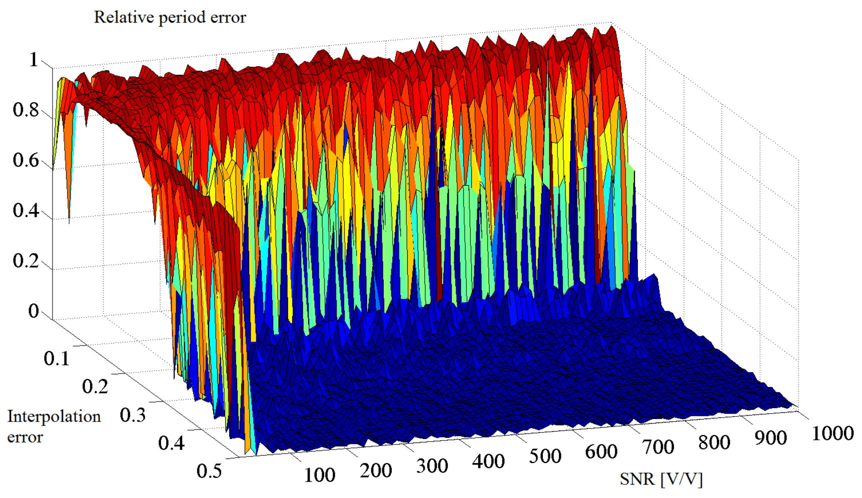


Fig. 5.15 Relative period error (referred to the nominal period) as a function of interpolation error and signal to noise ratio.

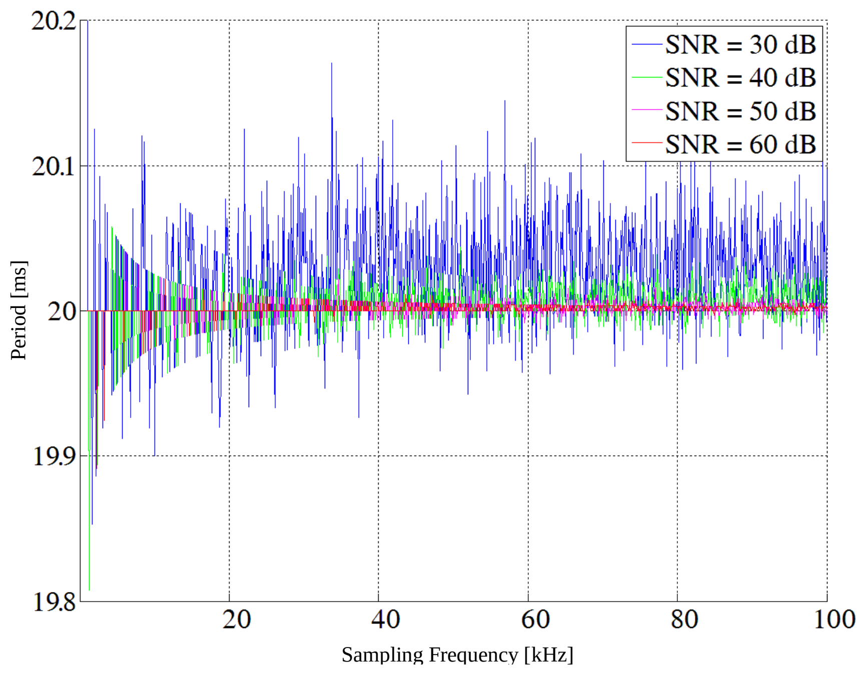


Fig. 5.16 Period as a function of sampling frequency. Four different colours are for different value of the signal to noise ratio simulated, as reported in the legend (for these graphs the interpolation error parameter is 0.35)

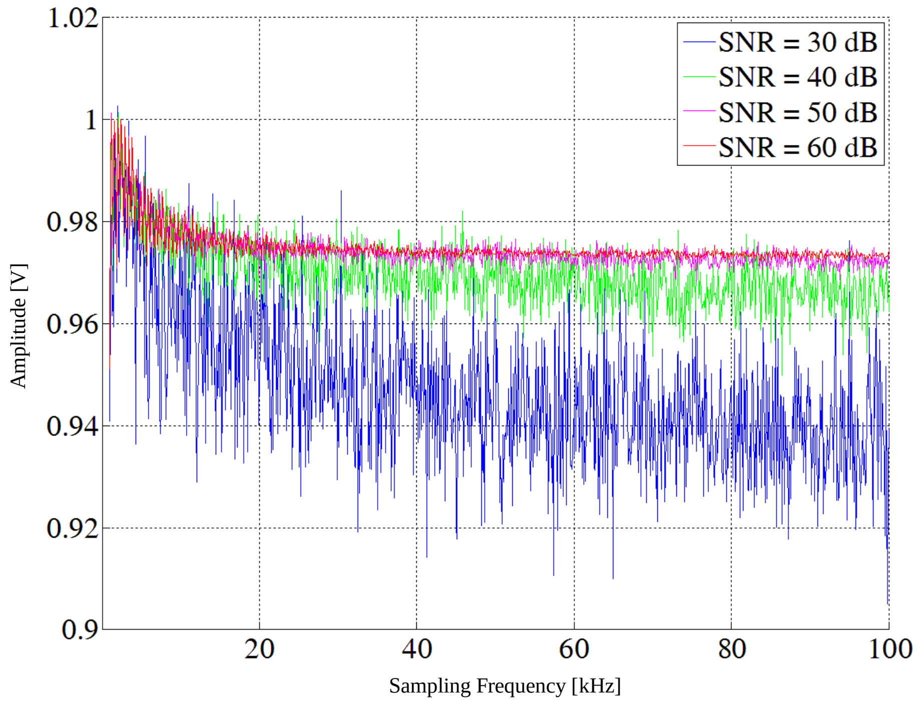


Fig. 5.17 Amplitude as a function of the sampling frequency. Four different colours for different value of the signal to noise ratio simulated, as reported in the legend (for these graphs the interpolation error value is 0.35).

Every real signal is affected by noise, depending on the type of sensors and applications, varying the type and the amount of noise. Once a sinusoidal waveform has been chosen as input, the error analysis has been made by varying the signal to noise ratio. In the Fig. 5.18, the mean and the standard deviation of the relative period error is graphed, computed on 1000 repeated simulations, for a sinusoidal signal of 50 Hz, acquired with a sampling frequency of 100 kHz. Results are shown for interpolation error value of 15% of the maximum value (in red), and for interpolation error value of 20% (in black). The slight difference between the two curves once again evidences the different behavior of the period calculation algorithm if the interpolation error changes. In particular, the higher the noise level, the higher the interpolation error, to reduce noise susceptibility. However, the interpolation error must be lower in absence of noise, to reduce errors and dispersion of period measurements.

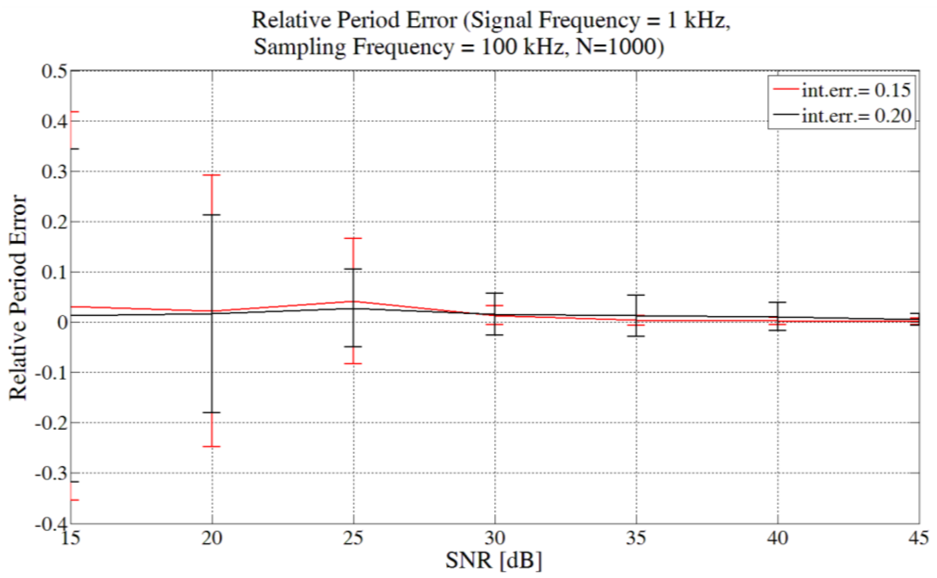


Fig. 5.18 Relative error of computed period on 100 repeated simulations, graphed as a function of the signal to noise ratio. Input sinusoidal waveform of 50 Hz, sampling frequency 50 kHz, results are for different interpolation error value.

5.1.3 Experimental results for period computation

The proposed algorithm for period measurement is tested on real signals: the oscilloscope LeCroy WaveMaster 8620A is used, running the Matlab® algorithm directly on the oscilloscope, with a Tektronix AWG2020 signal generator, in order to compare results of the period measurement of the oscilloscope, with results of the algorithm for period measurement based on the RTSAL algorithm. The algorithm implemented on the oscilloscope is *zero crossing*. Repeated measures are performed, varying input signal frequency, sampling frequency, dedicated record length of the oscilloscope, and interpolation error. As an input of the two algorithms on the oscilloscope, a sinusoidal waveform is stated on the AWG2020 signal generator, with amplitude of 1 V, and without offset. In order to let the algorithm work correctly a time window which contain at least two period must be selected, because the input vector for the Matlab® algorithm is the vector of waveform points acquired by the oscilloscope (*WformIn1*). Results of the period measurement algorithm of the oscilloscope are taken directly from the graphic interface of the oscilloscope, whereas the period measures based on the **RTSAL** algorithm are obtained utilizing the on-board Matlab®. In Tab. 5.1 and 5.2 the comparison of the two technique is shown. Data shown are referred to 25 repeated measures for each type, fixing all the parameters. A new acquisition of the input waveform is performed each time, and the two algorithm are applied on the new waveform points. Results are consistent: the technique based on the **RTSAL** algorithm, in the best case, has a standard deviation bigger than the one obtained with the other technique of one order of magnitude. Moreover, results are not affected by systematic effects, and they are always compatible with results of the *zero crossing*; results of period measures depend on the interpolation error value.

TABLE 5.1. PERIOD MEASUREMENTS (ON ONE THOUSAND SAMPLES)

Input Freq	**Record length: 1kS**				
	Sampling Freq	***Interpolation Error***		***T-Scope***[a]	***T-S&L***[a]
50 Hz	20 kHz	0.06	mean	19.991	20.08
			std	0.023	0.84
		0.07	mean	---[b]	19.98
			std	---	0.79
		0.08	mean	---	20.3
			std	---	1.2
1 kHz	500 kHz	0.06	mean	0.999	0.999
			std	0.001	0.045
		0.07	mean	---	1.017
			std	---	0.041
		0.08	mean	---	1.029
			std	---	0.050

a. Results are in milliseconds

b. Results for the oscilloscope measurement are the same, changing only the interpolation error

TABLE 5.2. PERIOD MEASUREMENTS (ON FIFTY THOUSAND SAMPLES)

Input Freq	**Record length: 50kS**				
	Sampling Freq	***Interpolation Error***		***T-Scope***[a]	***T-S&L***[a]
50 Hz	1 MHz	0.105	mean	20.008	20.12
			std	0.053	1.44
		0.110	mean	---[b]	19.84
			std	---	0.62
		0.115	mean	---	19.66
			std	---	0.87
1 kHz	25 MHz	0.105	mean	1.0001	1.021
			std	0.0027	0.037
		0.110	mean	---	1.015
			std	---	0.042
		0.115	mean	---	1.033
			std	---	0.047

a. Results are in milliseconds

b. Results for the oscilloscope measurement are the same, changing only the interpolation error

5.1.4 A comparison with another sensor signal preprocessing techniques: compressive sampling

Reducing the amount of data to be transmitted on a sensor network, whether wired or wireless, is a useful choice. In communication, the need for transferring less data means consuming less energy and using a smaller bandwidth for the data links.

The **RTSAL** algorithm is not the only possible strategy to achieve this goal: the objective of various signal processing algorithms is the reduction of the amount of data to be transmitted. In fact, there are a lot of possible algorithms, some of them computationally simpler, others more complex, and, generally, the complex ones require a high computational loads for obtaining the compression of the data, and simple ones require lower computational loads [61]. In this paragraph, a comparison with the compressive sensing [62-64] is shown, in order to compare period and amplitude calculation, as well as execution times and amount of data required for these algorithms.

The compressive sensing technique is based on an algorithm that takes a fixed number of samples from a signal. Therefore, from this point of view, it is a technique which allows to reduce the data amount to be transmitted, sending only several acquired samples, and allowing reconstruction of the input signal.

This algorithm is based on the signal transformation in other domains where this signal is represented by a lower quantity of samples, acquired with a casual step of sampling (Fig. 5.19).

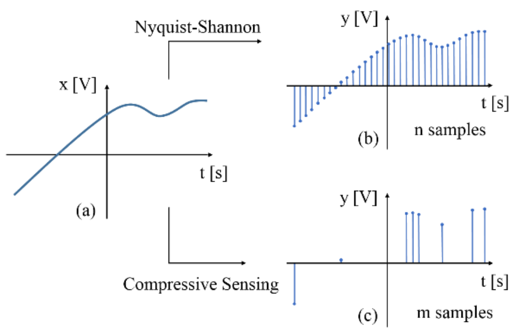


Fig. 5.19 Samples acquired following the Nyquist theorem (b) and the Compressive Sensing directives (c) on the same signal (a).

For example, a sinusoidal signal, or even a signal including various frequency components, has a Fourier transform such that only a few of those are substantial. When a signal has such a property it is said to be S-sparse. In order to apply the compressive sensing technique it would be necessary to choose the right expansion basis for the class of signals in which it would be used. Those signals would be represented, in their destination domain, by almost all zero samples and only few meaningful samples, which carry all the information contents.

For this algorithm, it is necessary to take random samples. Sampling mechanics in time domain can be expressed as:

$$y = \Phi \cdot x \tag{13}$$

where $y \in \mathbb{R}^m$is the vector of measures, $x \in \mathbb{R}^n$ is the input signal and Φ is the sampling matrix. The number of acquired samples, m, is reduced by this algorithm but is sufficient for reconstruction ($m \ll n$). The compressive sensing is an algorithm which can reduce the size of the transferred data while still having the same information content. The considered data compression is possible because there is a wide set of signals that have a sparse representation. Let us express x as:

$$x = \Psi \cdot s \tag{14}$$

where $s \in \mathbb{R}^n$ is a sparse representation of the signal x in the orthonormal basis $\Psi\,[n \times n]$. Combining the previous equations, the vector of the acquired samples can be expressed as a function of the sparse representation of the signal x:

$$y = A \cdot s = \Phi \cdot \Psi \cdot s \tag{15}$$

This equation system is characterized by infinite solutions. To make compressive sensing work with success, a further condition has to be met. The sensing matrices Φ and the orthonormal basis matrix Ψ have to be incoherent. Eventually, if the s vector is sufficiently sparse, the system is solved by an L1-norm minimization algorithm, which gives, under suitable conditions, the input signal as a unique solution [65-66].

Therefore, the sinusoidal signal has a sparse representation in the frequency domain, in which the fundamental frequency, and its amplitude value, can be located, implementing a search for the maximum value. In the following, Matlab® simulation results are shown: to calculate the fundamental period and amplitude of a sinusoidal signal, in order to obtain a random sampling of the waveform, a signal with equidistant samples is declared and a fixed number of samples are taken randomly (m).

The samples are located with a related definition of the sensing basis matrix $\Phi\ [m \times n]$, and also the coefficients of the Fourier transform are considered in the signal basis matrix $\Psi\ [n \times n]$. The L1-norm minimization is implemented with the Matlab® toolbox cvx. A simple search for the maximum values is done on the s vector and some processing are done on the value of the maximum. In order to obtain the fundamental period, it calculates the ratio between the number of the signal samples and the position of the maximum in the array of complex coefficients of the Fourier transform, and it multiplies by the inverse ratio of the sampling frequency. On the other hand, the amplitude of the fundamental is calculated from the absolute value of the maximum. A simple chart is shown in Fig. 5.20.

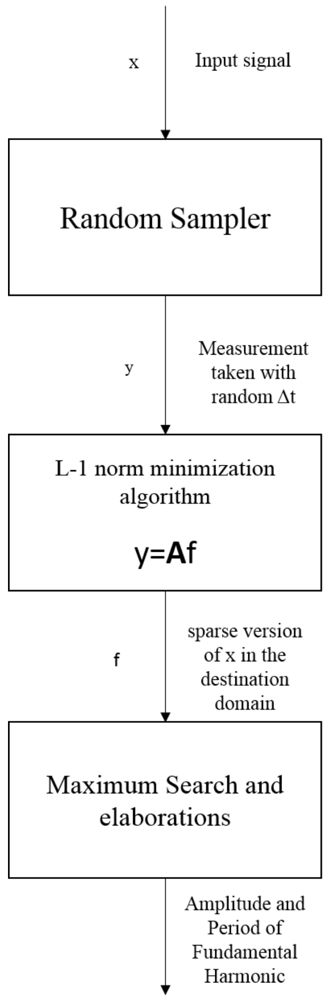

x
Input signal
Random Sampler
y
Measurement taken with random Δt
L-1 norm minimization algorithm
y=Af
f
sparse version of x in the destination domain
Maximum Search and elaborations
Amplitude and Period of Fundamental Harmonic

Fig. 5.20 Flow chart of period computation function based on the compressive sensing algorithm.

Several simulations have been executed in order to evaluate the performance of the compressive sensing algorithm. The same type of signal, as mentioned in the previous paragraph, is stated in the simulation program, in order to give it as an input to the algorithm, which returns as output the signal period and amplitude. Fig. 5.21 shows the period calculated based on compressive sensing, as a function of the sampling frequency derived from the equidistant samples of the sine waveform. The number of samples taken casually for the compressive sensing is 20, called m. Raising the sampling frequency, but taking the same number of samples with a fixed observation temporal window for the signal, the error decreases (it is as if the samples are taken more casually).

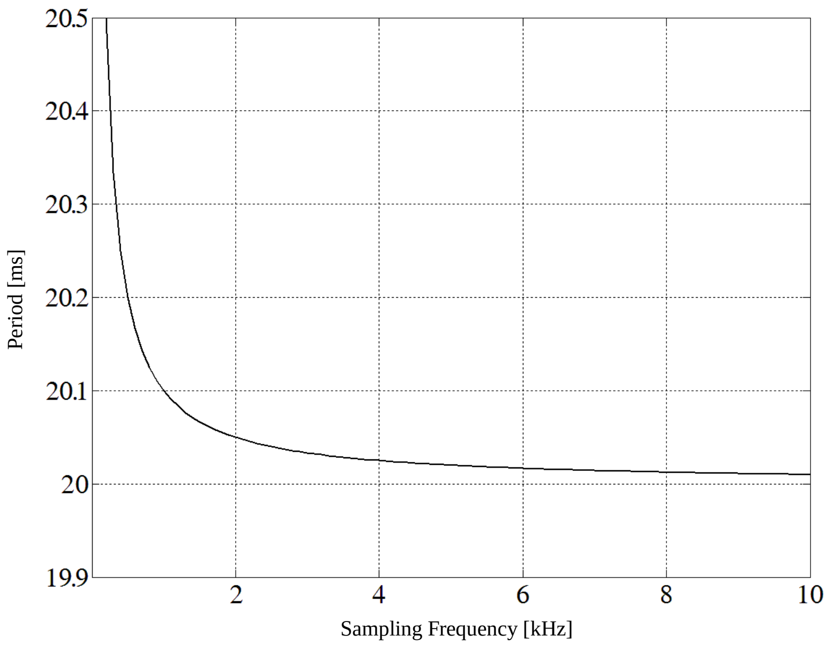


Fig. 5.21 Calculated period with compressive sensing as function of the sampling frequency.

These kinds of algorithms, this **RTSAL** algorithm and the compressive sensing, could be useful to reduce the data transferred or stored, starting from the raw samples, having the same information content. Therefore, these two algorithms are compared from this point of view. The first comparison shows the main differences between the results: Tables 5.3 and 5.4 show the period and the amplitude calculated by the algorithms, for the same input signal simulated in Matlab® (a sinusoidal waveform whose amplitude is 1 V, frequency is 50 Hz, in a fixed temporal window of 0.2 s, and with a signal-to-noise ratio of 40 dB of white Gaussian noise added). For the **RTSAL** algorithm, are reported the interpolation error value used, the mean and the standard deviation of the period, and the mean and the standard deviation of the amplitude, because it returns a period value for each period found in the analyzed temporal window and an amplitude value for each maximum found. For the compressive sensing the number of samples taken casually, m, is fixed at 30, and the calculated period and amplitude are reported in the corresponding table.

TABLE 5.3. PERIOD MEASUREMENT

Sampling Frequency [kHz]	*Interpolation Error*	*Period Mean (**RTSAL**) [ms]*	*Period Standard Deviation (**RTSAL**) [ms]*	*Period (**CS**) [ms]*
1	0.01	30	15	22.33
1	0.15	20.00	0.01	22.33
1	0.20	20.00	0.01	22.33
1	0.30	22.9	7.6	22.33
1	0.40	20.0	1.1	22.33
1	0.50	20.00	0.01	22.33
5	0.01	25.7	9.5	22.24
5	0.15	20.00	0.28	22.24
5	0.20	20.15	0.30	22.24
5	0.30	20.00	0.21	22.24
5	0.40	19.95	0.14	22.24
5	0.50	19.95	0.14	22.24
10	0.01	26	19	22.23
10	0.15	19.98	0.16	22.23
10	0.20	20.05	0.35	22.23
10	0.30	20.00	0.11	22.23
10	0.40	19.98	0.17	22.23
10	0.50	19.93	0.21	22.23
50	0.01	7.6	9.4	22.22
50	0.15	19.99	0.12	22.22
50	0.20	19.98	0.07	22.22
50	0.30	20.02	0.09	22.22
50	0.40	20.00	0.09	22.22
50	0.50	19.96	0.13	22.22

TABLE 5.4. AMPLITUDE MEASUREMENT

Sampling Frequency [kHz]	*Interpolation Error*	*Amplitude Mean (**RTSAL**) [V]*	*Amplitude Standard Deviation (**RTSAL**) [V]*	*Amplitude (**CS**) [V]*
1	0.01	0.9641	0.0045	0.9774
1	0.15	0.9479	0.0057	0.9660
1	0.20	0.9494	0.0051	0.9634
1	0.30	0.9485	0.0074	0.9566
1	0.40	0.724	0.081	0.9923
1	0.50	0.5883	0.0068	0.9489
5	0.01	0.9954	0.0048	0.9837
5	0.15	0.981	0.011	0.9574
5	0.20	0.972	0.016	0.9836
5	0.30	0.940	0.016	0.9888
5	0.40	0.899	0.014	0.9741
5	0.50	0.841	0.017	0.9711
10	0.01	0.76	0.50	0.9848
10	0.15	0.9688	0.0079	0.9893
10	0.20	0.983	0.016	0.9905
10	0.30	0.9499	0.0068	0.9848
10	0.40	0.9048	0.0095	0.9943
10	0.50	0.844	0.018	0.9905
50	0.01	0.92	0.29	0.9833
50	0.15	0.9678	0.0041	0.9857
50	0.20	0.9835	0.0078	0.9814
50	0.30	0.9473	0.0068	0.9870
50	0.40	0.9074	0.0087	0.9845
50	0.50	0.858	0.011	0.9807

Looking at this table, it is clear that periods and amplitudes, computed by the **RTSAL** algorithm, change depending on the value of the interpolation error parameter.

Table 5.5 compare, for the same signal mentioned before, the number of bytes necessary to be transferred and the execution times, for a fixed interpolation error value. Regarding the memory allocation for the output of the algorithms, for the **RTSAL** algorithm it is a total of 7 bytes for each sample: one byte for the amplitude value **M**, one for the class **C**, and four bytes for the time **T** (in seconds to midnight). Whereas, for the compressive sensing, fixed 30 samples casually taken, for each sample are considered 5 bytes, one for the amplitude value, and another four bytes for the time instant.

TABLE 5.5. NUMBER OF BYTES AND EXECUTION TIME

Sampling Frequency [kHz]	*Interpolation Error*	*Number of bytes to send (**RTSAL**)*	*Number of bytes to send (**CS**)*	*Execution Time (**RTSAL**) [ms]*	*Execution Time (**CS**) [s]*
1	0.01	595	150	22.81	1.08521
1	0.15	147	150	23.38	1.06306
1	0.20	147	150	22.35	1.04783
1	0.30	147	150	22.07	1.04555
1	0.40	140	150	22.07	1.05290
1	0.50	140	150	22.43	1.05705
5	0.01	1771	150	24.66	1.35985
5	0.15	287	150	23.31	1.41131
5	0.20	175	150	24.51	1.40717
5	0.30	147	150	23.85	1.35811
5	0.40	147	150	23.83	1.36800
5	0.50	147	150	23.75	1.34012
10	0.01	2912	150	29.43	1.93774
10	0.15	287	150	26.36	1.84686
10	0.20	224	150	26.76	1.84195
10	0.30	147	150	26.32	1.88189
10	0.40	147	150	25.92	1.79951
10	0.50	147	150	26.45	1.78880
50	0.01	14560	150	60.08	8.72041
50	0.15	287	150	44.78	9.32604
50	0.20	287	150	45.16	8.77634
50	0.30	147	150	47.79	8.92872
50	0.40	147	150	49.79	8.19088
50	0.50	147	150	45.96	11.58102

There are enormous differences between the times needed by these algorithms: the compressive sensing is complex because it involves a minimization method; on the contrary, the **RTSAL** algorithm involves only sum operations, comparisons, and a division by two (which could be implemented with a simple register shift).

This means that the **RTSAL** algorithm can be implemented easily on devices with limited hardware resources, like smart sensors, and it could work in real-time while acquiring the samples. Furthermore, compressive sensing requires keeping in memory the matrix *A*, which could be very large.

In the event when it is useful not to process the samples directly, and it is possible to process these data on a remote, more powerful device, the compressive sensing could be implemented, transferring only the casually acquired samples and their position.

Eventually, experimental results of the implemented algorithms are reported: these algorithms are applied on the acquired signal with the LeCroy WaveMaster 8620A oscilloscope. Some results are reported, in Tables 5.6, 5.7, 5.8 and 5.9, with sinusoidal, triangular, and square waveforms as inputs. Period, amplitude and reconstruction percentage error of the two algorithms are reported and compared. These results are similar to those obtained with the simulated signals. It must be considered that, for the compressive sensing, the same vector Ψ is utilized. Then, the reconstruction percentage error, for other types of signals, has increased considerably. Depending on the type of signals it is necessary to select the proper vector, in order to obtain a sparse representation of the acquired signal.

TABLE 5.6. EXPERIMENTAL TESTS – SINUSOIDAL WAVEFORM INPUT SIGNAL

Signal Frequency [Hz]	*Sampling Frequency [kHz]*	*Interpolation Error*	*Period Mean (**RTSAL**) [s]*	*Period (**CS**) [s]*	*Amplitude Mean (**RTSAL**) [V]*	*Amplitude (**CS**) [V]*	*Mean of Reconstruction Percentage Error (**RTSAL**) [%]*	*Mean of Reconstruction Percentage Error (**CS**) [%]*
50	5	0.01	0.016	0.02500	0.8132	0.95394	2.5	3.7
50	5	0.10	0.0206	0.02500	0.9281	0.93286	6.8	3.9
50	5	0.20	0.0200	0.02500	0.9447	0.93917	15	3.6
50	100	0.01	0.0022	0.02500	0.513	0.91970	2.5	4.3
50	100	0.10	0.0203	0.02500	0.9170	0.71230	6.4	8.4
50	100	0.20	0.0200	0.02500	0.9060	0.88270	11	6.8
1000	100	0.01	0.002	0.00111	0.8088	0.91891	2.5	4.4
1000	100	0.10	0.0010	0.00111	0.9207	0.93279	7.2	4.7
1000	100	0.20	0.0010	0.00111	0.9256	0.94573	18	3.6
1000	500	0.01	0.0005	0.00111	0.825	0.93059	2.5	4.5
1000	500	0.10	0.0009	0.00111	0.8698	0.89438	5.3	5.4
1000	500	0.20	0.0010	0.00111	0.9305	0.94352	12	4.4

TABLE 5.7. EXPERIMENTAL TESTS – TRIANGULAR WAVEFORM INPUT SIGNAL

Signal Frequency [Hz]	*Sampling Frequency [kHz]*	*Interpolation Error*	*Period Mean (**RTSAL**) [s]*	*Period (**CS**) [s]*	*Amplitude Mean (**RTSAL**) [V]*	*Amplitude (**CS**) [V]*	*Mean of Reconstruction Percentage Error (**RTSAL**) [%]*	*Mean of Reconstruction Percentage Error (**CS**) [%]*
50	1	0.01	0.260	0.02083	0.8397	0.74863	3.8	5.2
50	1	0.20	0.0242	0.02083	0.6669	0.73011	29	4.7
50	1	0.40	0.0200	0.02083	0.2978	0.75222	37	4.2
50	100	0.01	0.001	0.02500	0.5545	0.70646	2.2	9.2
50	100	0.20	0.0201	0.02500	0.7678	0.60910	15	11
50	100	0.40	0.0202	0.02500	0.5082	0.69162	28	8.7
1000	100	0.01	0.001	0.00111	0.8231	0.70800	2.5	6.5
1000	100	0.20	0.0010	0.00111	0.7193	0.74052	18	4.4
1000	100	0.40	0.0010	0.00111	0.4833	0.73615	32	5.6
1000	500	0.01	0.0007	0.00111	0.2375	0.66437	2.7	8.2
1000	500	0.20	0.0010	0.00111	0.7218	0.75929	17	5.1
1000	500	0.40	0.0010	0.00111	0.5131	0.69472	29	6.3

TABLE 5.8. EXPERIMENTAL TESTS – SQUAREWAVE (DUTY 50%) INPUT SIGNAL

Signal Frequency [Hz]	*Sampling Frequency [kHz]*	*Interpolation Error*	*Period Mean (**RTSAL**) [s]*	*Period (**CS**) [s]*	*Amplitude Mean (**RTSAL**) [V]*	*Amplitude (**CS**) [V]*	*Mean of Reconstruction Percentage Error (**RTSAL**) [%]*	*Mean of Reconstruction Percentage Error (**CS**) [%]*
50	1	0.01	0.012	0.02083	0.9909	1.17890	17	37
50	1	0.10	0.0230	0.02083	1.0010	0.93221	78	39
50	1	0.20	0.0200	0.02083	0.9838	1.07043	89	38
50	100	0.01	0.003	0.02500	0.4683	0.68253	3.3	55
50	100	0.10	0.0025	0.02500	0.9837	0.78738	10	53
50	100	0.20	0.0200	0.02500	1.0165	0.84215	115	46
1000	100	0.01	0.0017	0.00111	0.0552	0.60818	5.9	61
1000	100	0.10	0.0009	0.00111	1.0054	0.58141	57	56
1000	100	0.20	0.0010	0.00111	0.9889	0.67602	112	49
1000	500	0.01	0.0001	0.00111	0.9092	0.84250	3.7	46
1000	500	0.10	0.0017	0.00111	0.5561	0.75593	27	56
1000	500	0.20	0.0010	0.00111	1.0027	0.70866	116	49

TABLE 5.9. EXPERIMENTAL TESTS – SQUAREWAVE (DUTY 15%) INPUT SIGNAL

Signal Frequency [Hz]	*Sampling Frequency [kHz]*	*Interpolation Error*	*Period Mean (**RTSAL**) [s]*	*Period (**CS**) [s]*	*Amplitude Mean (**RTSAL**) [V]*	*Amplitude (**CS**) [V]*	*Mean of Reconstruction Percentage Error (**RTSAL**) [%]*	*Mean of Reconstruction Percentage Error (**CS**) [%]*
50	1	0.01	0.053	0.02083	1.0054	0.67681	10	32
50	1	0.10	0.0200	0.02083	0.9760	0.34214	117	40
50	1	0.20	0.0200	0.02083	0.9769	0.53154	155	19
1000	100	0.01	0.0015	0.00111	0.8839	0.44719	8.3	64
1000	100	0.10	0.0010	0.00111	0.8888	0.30100	101	56
1000	100	0.20	0.0010	0.00111	0.9695	0.44268	172	47
1000	500	0.01	0.0007	0.00111	0.6526	0.34938	5.5	50
1000	500	0.10	0.0007	0.00111	0.6526	0.54401	5.5	53
1000	500	0.20	0.0010	0.00111	0.9999	0.33524	173	58

A comparison between two different methods for extracting fewer data, but acquiring the same information is shown. Compressive sensing is characterized by a greater complexity, but it is an algorithm with which, knowing the domain of the analyzed signal, it is possible to reconstruct the input signal. The segmentation and labeling algorithm is less complex, it allows a worse signal reconstruction and it is less immune to noise, because the selection criteria for the samples is related to the amplitude.

Then, in some cases, the algorithm which gives minimal use of the bus would be chosen: in fact, compressive sensing needs a fixed number of samples. The **RTSAL** algorithm needs even more samples, if the signal has to be reconstructed. In other cases, it should select the algorithm which needs fewer operations, or less processing time. The **RTSAL** algorithm, unlike the compressive sensing, could work in real time, because it is not so complex, it would need a little memory, and it could even be possible to directly communicate the extracted parameters of interest (period, amplitude, etc…), rather than the **MCT** data structure.

5.1.5 Period measurement with an ARM microcontroller

In this paragraph, a basic implementation on an ARM microcontroller is shown, based on the shown period computation technique. This analysis highlights the performance of the technique in terms of execution time.

Experimental results obtained for the period measurement with a microcontroller are shown in this section. A development board from ETT for a STM32F103 ARM Cortex-M3 is used for testing the algorithm, with an AWG2020 waveform generator. The microcontroller acquires a fixed number of points (256) with a fixed sampling frequency, it applies the **RTSAL** algorithm, and it elaborates the **MCT** data structure, calculating the period of the sinusoidal waveform (Fig. 5.22).

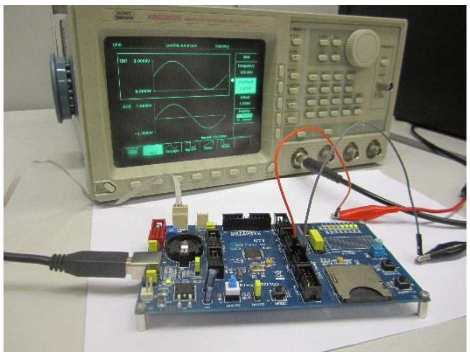


Fig. 5.22 Period measurement with an ARM microcontroller.

The results, in Table 5.10, are reported in milliseconds, as a function of nominal frequency of the input waveform, and as a function of sampling frequency for the ADC. In order to acquire the input signal correctly, some constraints have to be observed:

- This technique selects the most significant samples of a signal, sampled with a high sampling frequency. It is not useful to apply a technique that selects samples, on a signal that has 2 points for period, as the strictly minimum of Nyquist condition says. In those measures, taking 10 samples for each period, could be an acceptable minimum;
- Temporal window must include, at least, two periods of the input waveform. For the correct operating condition of the technique based on the segmentation and labeling algorithm, it is needed to have at least two periods, in order to calculate the time interval between two consecutive maxima.

TABLE 5.10. PERIOD MEASUREMENT (ON MICROCONTROLLER)

Sampling Freq \ Input Freq	50 Hz	100 Hz	200 Hz	500 Hz	1 kHz
500 Hz	20.00	---	---	---	---
1 kHz	20.00	9.20	---	---	---
2 kHz	19.00	10.00	5.00	---	---
5 kHz	18.00	10.00	4.80	2.00	---
10 kHz	---	10.60	5.00	2.00	1.00
20 kHz	---	---	4.80	2.00	1.00
50 kHz	---	---	---	1.72	0.92

Results are in milliseconds (max percentage standard deviation of 3%)

This table reports means value of 20 measures for segmentation and labeling algorithms. The maximum of the percentage standard deviation for these measures is about 3%.

Eventually, the execution time of this technique is evaluated: it takes between 0.23 ms and 0.69 ms to run. It takes into account the acquisition time needed to capture the waveform and to execute the measuring task. This algorithm could be good with different acquisition systems, especially in distributed system, like sensor networks, where computational load is generally a big issue. In fact, in a sensor network, a typical node does not have many computational capabilities.

5.2 Mean estimation with RTSAL algorithm

With only several samples, not equally spaced in time, other techniques to compute the mean value have to be applied, in order to take into account the temporal position of the stored samples. Mean value estimation could be implemented, assuming the shape of the signal as linear between two stored samples, and computing the trapezoid area under the assumed linear signal. So, to compute the area under the line that connects two consecutive stored samples, an acceptable approximation is to compute the area of a trapezoid for each time interval (Fig. 5.23).

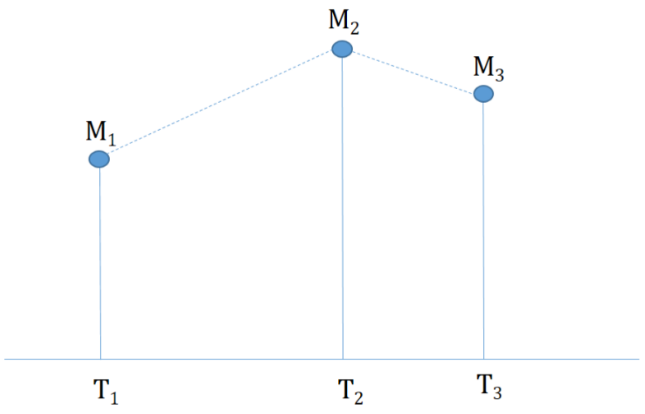


Fig. 5.23 Trapezoidal rule. Stored samples are not equally spaced, so the region under the graph are approximated as a trapezoid.

Therefore, the formula (1) (**MTA**, mean trapezoid area) is applied, instead of applying the (2) (**SM**, standard mean) on the stored samples.

$$\bar{x} = \frac{1}{2(T_N - T_1)} \sum_{i=1}^{N-1} (M_{i+1} + M_i)(T_{i+1} - T_i) \quad (16)$$

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} M_i \quad (17)$$

An application of the algorithm, within simulations and real signals, is analyzed. At first, the algorithm is tested on simulated signal, implementing the algorithm in Matlab® on a personal computer, and formulas for mean value computations are applied. Then, this algorithm is tested on real acquired signal with real instrumentation. The oscilloscope LeCroy WaveMaster 8620A is used, running the Matlab® algorithm directly on the oscilloscope, with a Tektronix AWG2020 signal generator, in order to compare the results of the mean measurement of the oscilloscope, with the results of the algorithm for the same parameter, based on the **RTSAL** technique. Simulations done in Matlab® are reported in Table 5.11.

The proposed techniques, used for the mean value in (16), are compared with results of the (17) applied on the segments. The input signal is a sinusoidal waveform of unitary amplitude, and no offset, and these tests are done varying the input frequency, the sampling frequency, the interpolation error value and the SNR. The **RTSAL** algorithm is applied to this signal, and then these formulas are applied to the **MCT** data structure. In this table are also reported the execution time needed by the two different computation, and the number of stored samples, which directly depend on the interpolation error value.

TABLE 5.11. MEAN VALUE ESTIMATION (SIMULATIONS)

500Hz SNR=30dB	**Sampling Frequency - (Number of samples)**					
	10 kHz (500 S)		**100 kHz (5000 S)**		**1 MHz (50000 S)**	
	MTA (16)	**SM(17)**	**MTA(16)**	**SM(17)**	**MTA(16)**	**SM(17)**
Interpolation Error = 0.01	0.13 V	0.15 V	0.13 V	0.14 V	0.13 V	0.13 V
Execution time	13.66 µs	5.99 µs	120.60 µs	50.30 µs	1194.68 µs	500.72 µs
Stored Segment	390 S		4013 S		40274 S	
Interpolation Error = 0.05	0.12 V	0.09 V	0.13 V	0.12 V	0.13 V	0.13 V
Execution time	6.66 µs	3.33 µs	38.31 µs	16.32 µs	377.79 µs	167.57 µs
Stored Segment	128 S		1219 S		12428 S	
Interpolation Error = 0.10	0.059 V	0.13 V	0.13 V	0.26 V	0.13 V	0.14 V
Execution time	2.99 µs	1.66 µs	5.66 µs	2.66 µs	40.98 µs	17.66 µs
Stored Segment	30 S		104 S		1284 S	
Interpolation Error = 0.15	0.068 V	0.12 V	0.09 V	0.18 V	0.11 V	0.12 V
Execution time	2.66 µs	1.66 µs	3.33 µs	1.99 µs	4.99 µs	2.33 µs
Stored Segment	21 S		26 S		58 S	
Interpolation Error = 0.20	0.027 V	0.027 V	0.07 V	0.10 V	0.11 V	0.12 V
Execution time	2.66 µs	1.66 µs	3.66 µs	1.99 µs	5.33 µs	2.33 µs
Stored Segment	18 S		21 S		50 S	
Interpolation Error = 0.25	0.043 V	0.086 V	0.028 V	0.029 V	0.096 V	0.103 V
Execution time	2.66 µs	1.33 µs	3.66 µs	1.99 µs	4.33 µs	1.99 µs
Stored Segment	16 S		18 S		49 S	

With formulas (1), which take into account the time position of the stored samples, the execution time is always bigger than the time required for the formulas (2), because it needs multiply operations, for the mean value computation. The error of the formulas varies, depending on how many samples for each period are acquired, and how many samples are stored by the segmentation and labeling algorithm, to which these formulas are applied. Using this technique, the aim must be to reduce the number of stored or sent samples, so the choice of the sampling frequency and the interpolation error for the specific application is a complex issue.

Results of the mean estimation algorithm of the oscilloscope are taken directly from the oscilloscope, whereas the measures based on the **RTSAL** algorithm are obtained utilizing the on-board Matlab®. The input vector for the Matlab® algorithm is the vector of waveform point acquired by the oscilloscope (*WformIn1*).

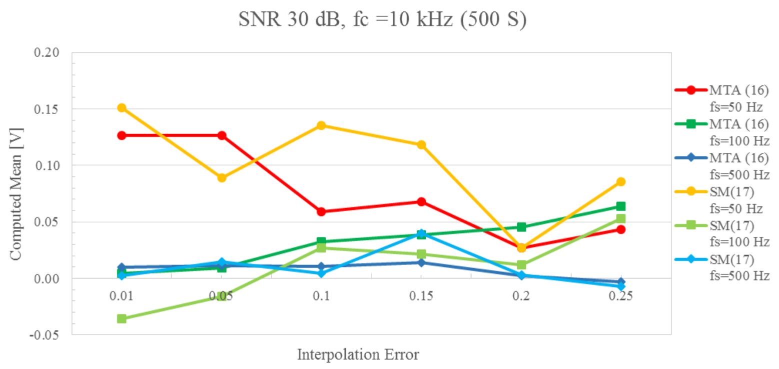


Fig. 5.24 Mean computed with formulas (16) and (17) on the **MCT** data structure, with input frequency signal of 50 Hz, 100 Hz and 500 Hz.

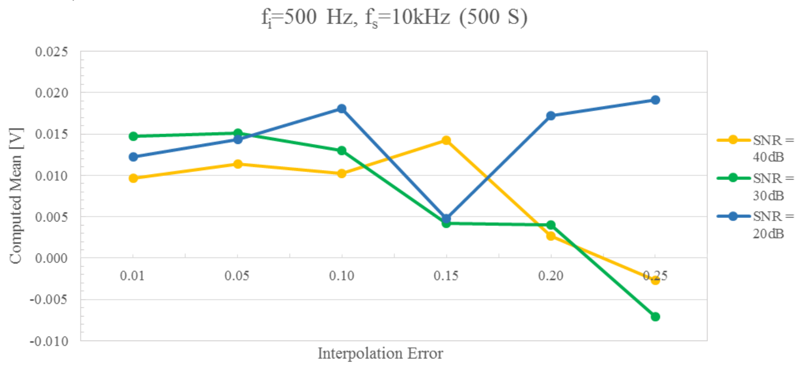


Fig. 5.25 Mean computed with formula (16) on **MCT** data structure, with input frequency signal of 500 Hz, and sampling frequency of 10 kHz, with SNR of 40, 30 and 20 dB.

In Table 5.12, the comparison of the two techniques is shown. Data shown are referred to 20 repeated measures for each interpolation error, fixing all the other parameters. A new acquisition of the input waveform is performed each time, and the algorithms are applied on the acquired waveform points. Results are consistent, the technique based on the **RTSAL** algorithm, in the best case, always has a standard deviation bigger than the one obtained with the oscilloscope. Thus, results of measures are strongly affected by the interpolation error value, and are less accurate.

TABLE 5.12. MEAN VALUE ESTIMATION (OSCILLOSCOPE)

f_i = 500 Hz		**f_s =10 kHz (500 S)**		
Interpolation Error		**Mean-Scope [mV]**	**MTA [mV]**	**Saved Samples [S]**
0.05	Mean	-5.4	-16.9	124.6
	Standard Deviation	1.7	5.2	3.1
0.10	Mean	-5.4	-8	97.5
	Standard Deviation	1.7	10	3.4
0.15	Mean	-5.4	-13	70
	Standard Deviation	1.7	13	17

5.3 Analysis of vibrations signals from faulty bearing

The **RTSAL** algorithm has been tested also for processing vibration signals to detect and isolate bearing faults. The database of the Case Western Reserve University Bearing data center was used in these tests [67]. Test signals are all sampled at a frequency of 48 kHz. Fig. 5.26 shows the accelerometer signal in unfaulty conditions. On the right, the signal is zoomed to evidence the waveform.

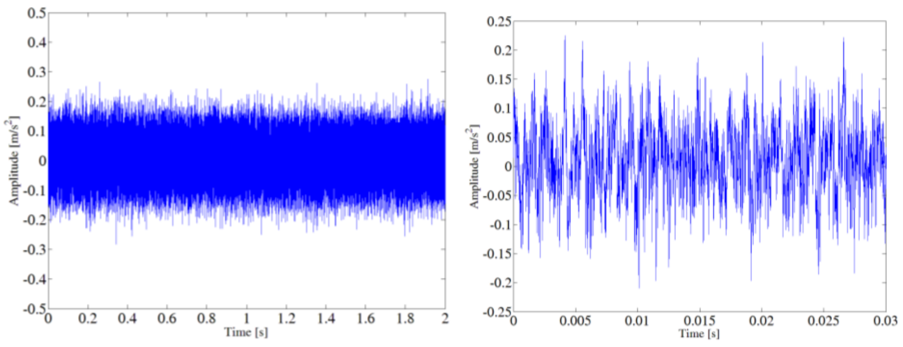


Fig. 5.26 Left: Normal accelerometer signal (no defect). Right: Zoom of 30 milliseconds.

Signals related to single point bearing faults made by using electro-discharge machining are available in the same database as well. For example, in this paper signal concerning fault diameters of 7 mils, 14 mils and 21 mils (1 mil=0.001 inches) are used to test algorithms.

In the upper frame of Fig. 5.27 a signal obtained with an inner race defect of seven mils is represented, whereas, in the lower frame a zoom of a signal segment is shown. Note that the signal looks like being a modulated impulsive carrier, and the modulation would be evident by enveloping the amplitudes of the local maxima. As a consequence, the local maxima obtainable after a single iteration of the **RTSAL** algorithm, can be used to detect the signal modulation.

Fig. 5.28 shows the local maximum signal corresponding to the signal of Fig. 5.27.

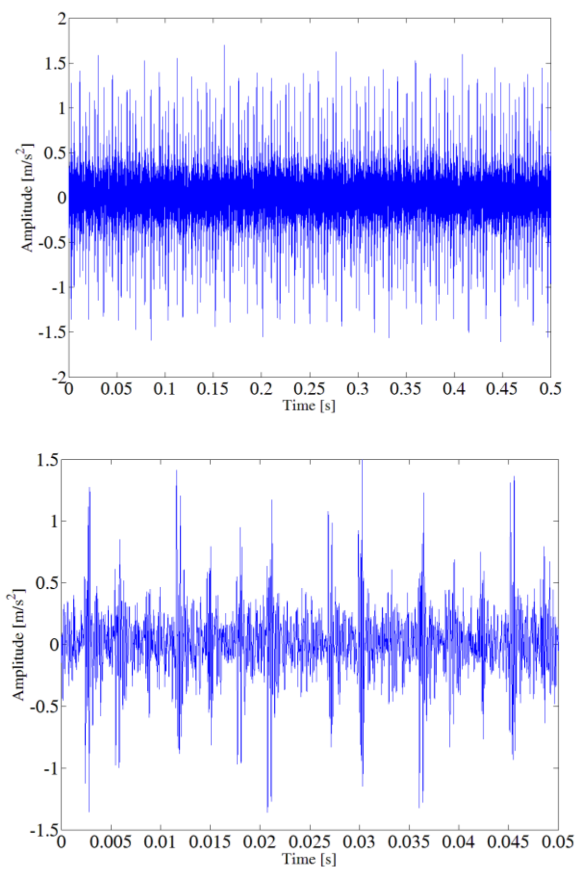


Fig. 5.27 Upper: vibration signal for an inner race defect of 0.007''. Lower: Zoom of a 50 ms time window. Sampling frequency: 48 kHz.

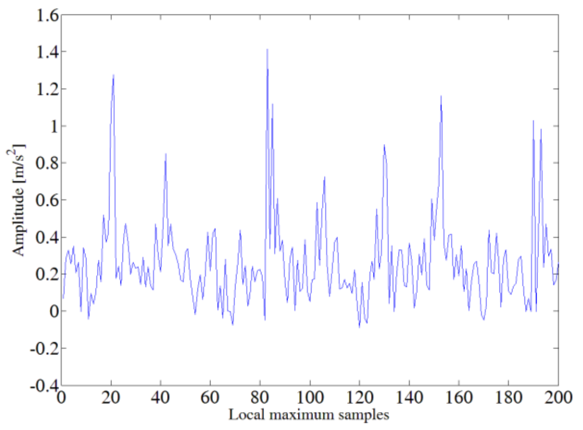


Fig. 5.28 Envelope using maximum points for the inferior signal in Fig. 5.27.

The steps of a procedure based on the **RTSAL** algorithm, which could be able to detect and isolate faults through a signal classification, are the following:

1. Apply the **RTSAL** algorithm on the buffer of samples;
2. Compute mean value of the signal built with only the samples that belong to a local maximum, called “*NewSignal*”. This random variable is named ***MeanMaxMCT***;
3. Apply the **RTSAL** algorithm to “*NewSignal*”, and repeat step 2 calling the outputs “*NewMaxSignal*” and ***MeanNewMaxMCT*** respectively;
4. Compute variance of *NewMaxSignal*. This random variable is named ***VarMaxMCT***.
5. Graph a point in a Cartesian plane where ***MeanMaxMCT*** values are on x axis and ***VarMaxMCT*** values are on y axis.

The position of this point on the Cartesian plane can help to recognize the bearing status.

To this aim, in Fig. 5.29 the points related to inner race defective signals for different intensities, and the ones related to unfaulty signals, are shown. Whereas, in Fig. 5.30, the points related to inner race defective signal for different types of defect are shown with the points related to unfaulty signals.

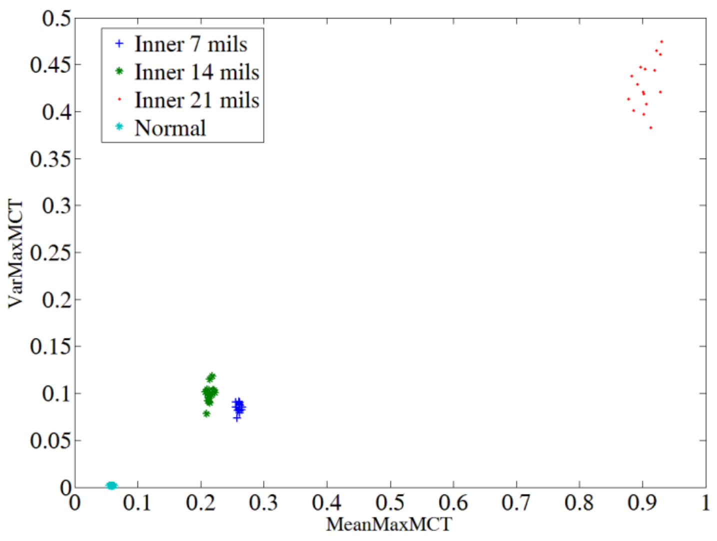


Fig. 5.29 Classification graph for an inner race defective signal of 7, 14 and 21 mils.

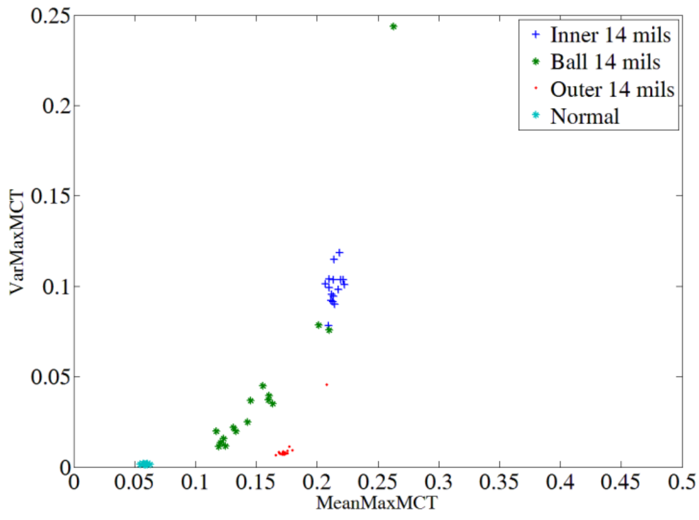


Fig. 5.30 Classification graph for defective signals of balls, inner race and outer race of 14 mils.

The relationship between ***MeanMaxMCT*** and ***VarMaxMCT*** provides information about the type of defect. The distance from the origin, i.e. module, provides information about the damage intensity. In addition, the ***MeanMaxMCT*** is information about the amount of organized patterns that reflects the defect, whereas from the signal "*NewMaxSignal*" it is possible to extract information about the envelope shape. For example, for an inner race defective signal as shown in Fig. 5.27, there are big differences among results of different inner race sizes. Signals of a normal bearing are the nearest to the origin, with a mean of almost 0.06 and no variance; whereas, for inner race of 7 mils, the smallest mean is 0.25, and for inner race of 14 mils, the biggest mean is 0.21.

Conclusions

This thesis shows an in depth analysis of the **RTSAL** algorithm performance, in different application contexts. Firstly, this algorithm was analyzed, in order to evaluate differences of the various implementation techniques, underlining advantages and disadvantages of each one. Moreover, this algorithm, together with the other ones of the first layer expected from the draft, was implemented on a microcontroller, in order to verify the real time implementation effective simplicity and to demonstrate the functional capability of a sensor network node, based on a microcontroller, which implements this technique. Period and mean results based on the data structure returned by this technique, were analyzed, both in simulations and on real hardware. Furthermore, the proposed technique was compared with compressive sampling, highlighting differences. From these analyses, it is possible to understand how results strongly depend on the interpolation error choice, both in terms of accuracy and in terms of number of data to be managed. Eventually, an application example of the algorithm for signal processing was shown for the analysis of vibrations signals from faulty bearing.

Utilizing this technique is fundamental: the application of this algorithm, even in other contexts, is important for a widespread use of this practice. This technique could have significant results improving the bandwidth and the energy management in distributed sensor networks, and therefore even lightening the large amounts of data to be analyzed by the central elaboration systems. These benefits may be achieved only if this technique will be widely accepted and used by smart sensor manufacturers.

In the context of the Internet of Things, where data to be exchanged and communicated are one of the main problems, the reduction of data at the root, that is in the point of acquisition, without information losses, is the best way to facilitate the development of this network, which will interconnect all the objects in the future.

Acknowledgements

This thesis is written within the agreement "*Avviso Pubblico destinato a giovani campani, tipologia progettuale: dottorati in Azienda*", by the Campania region, with P.O.R. Campania F.S.E. 2007/2013 funds. This agreement was taken between the "*Università degli Studi di Salerno*" and the "*Italdata S.p.A.*", and the research activities are done in the research and development laboratories of Italdata in Avellino, and the measurement laboratories of the DIIn at the University of Salerno.

References

[1] D. Evans, "T*he Internet of Things: How the Next Evolution of the Internet Is Changin Everything*". White Paper, Cisco IBSG, April 2011 (Retrieved 4 September 2015).

[2] A. Wood, "*The internet of things is revolutionizing our lives, but standards are a must*", The Guardian. Retrieved 31 March 2015.

[3] Texas Instruments "*The Internet of Things: Opportunities & Challenges*", 2014. Available at: www.ti.com/ww/en/internet_of_things/pdf/14-09-17-IoTforCap.pdf.

[4] S. Jain, N. V. Kumar, A. Paventhan, V.K. Chinnaiyan, V. Arnachalam, M. Pradish, "*Survey on Smart Grid Technologies-Smart Metering, IoT and EMS*" IEEE Electrical, Electronics and Computer Science (SCEECS), 2014.

[5] D. Kyriazis, T. Varvarigou, A. Rossi, D. White, J. Cooper, "*Sustainable smart city IoT applications: Heat and electricity management & Eco-conscious cruise control for public transportation*", IEEE 14th International Symposium and Workshops on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013.

[6] M. Feldmeier, J.A. Paradiso, "*Personalized HVAC Control System*", IEEE Internet of Things Conference, 2010.

[7] D. Brunelli, I. Minakov, R.Passerone, M. Rossi, "*Smart monitoring for sustainable and energy-efficient buildings: a case study*", IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems (EESMS), Trento, 2015.

[8] G. Caliano, L. De Vito, F.P. Di Candia, G. Mazzilli, F. Picariello, "*Architecture of the Monitoring System designed for an Active Guard Rail*", 20th IMEKO TC4 International Symposium and 18th International Workshop on ADC Modelling and Testing, Research on Electric and Electronic Measurement for the Economic Upturn, Benevento, Italy, September 15-17, 2014.

[9] D. Spirijakin, A. Baranov, A. Karelin, A. Somov, "*Wireless Multi-Sensor Gas Platform for Environmental Monitoring*" IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems (EESMS), Trento, 2015.

[10] S. Li, H. Wang, T. Xu, G. Zhou, "*Application Study on Internet of Things in Environment Protection Field*" Lecture notes in Electrical Engineering Volume 113, 2011.

[11] E. Bassoli, L. Vincenzi, M. Bovo, C. Mazzotti, "*Dynamic identification of an ancient masonry bell tower using a MEMS-based acquisition system*", IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems (EESMS), Trento, 2015.

[12] S. Amendola, R. Lodato, S. Manzari, C. Occhiuzzi, G. Marrocco, "*RFID Technology for IoT-Based Personal Healthcare in Smart Spaces*", IEEE Internet of Things Journal, vol.1, issue.2, pp. 144-152, March 2014, doi: 10.1109/JIOT.2014.2313981.

[13] F. Abate, G. Di Leo, A. Paolillo, V. Paciello, "*An Advanced traceability system based on semi-active RFID devices*", 20th IMEKO TC4 International Symposium and 18th International Workshop on ADC Modelling and Testing, Research on Electric and Electronic Measurement for the Economic Upturn, Benevento, Italy, September 15-17, 2014.

[14] L.B. Campos, C.E. Cugnasca, "*Towards an IoT-Based Architecture for Wine Traceability*", International Conference on Distributed Computing in Sensor Systems (DCOSS), Fortaleza, June 2015.

[15] J. Manyika, M. Chui, P. Bisson, J. Woetzel, R. Dobbs, J. Bughin, D. Aharon, "*The Internet of Things, mapping the value beyond the hype*", McKinsey Global Institute, June 2015.

[16] F. Thiesse, C. Floerkemeier, M. Harrison, F. Michahelles, "*Technology, Standards, and Real-World Deployments of the EPC Network*", IEEE Internet Computing, vol. 13, issue 2, pp. 36-43, April 2009, doi: 10.1109/MIC.2009.46.

[17] Z. Liu, D. Liu, L. Li, H. Lin, Z. Yong, "*Implementation of a New RFID Authentication Protocol for EPC Gen2 Standard*", vol. 15, issue 2, pp. 1003-1011, September 2014, doi: 10.1109/JSEN.2014.2359796.

[18] "EPCIS – *EPC Information Services Standard*", GS1. Available at: www.gs1.org/epcis/epcis/latest.

[19] C. Perera, R. Ranjan, L. Wang, S.U. Khan, A.Y. Zomaya, "*Big Data Privacy in the Internet of Things Era*", IEEE IT Professional Magazine: Special Issue Internet of Anything 2015 (to be published).

[20] R. Frank, "*Understanding Smart Sensors*", 2nd Ed., Artech House, Norwood, MA 02062, 2000.

[21] S. Gervais-Ducouret, "*Next smart sensors generation*", IEEE Sensors Applications Symposium (SAS), San Antonio, TX, 2011, doi: 10.1109/SAS.2011.5739775.

[22] W.B. Heinzelman, A.P. Chandrakasan, H. Balakrishnan, "*An Application-Specific Protocol Architecture for Wireless Microsensor Network*", IEEE Transactions on Wireless Communications, vol. 1, issue. 4, pp. 660-670, Oct. 2002, doi: 10.1109/TWC.2002.804190.

[23] N. Amini, M. Fazeli, S.G. Miremadi, M.T. Manzuri, "*Distance-Based Segmentation: An Energy-Efficient Clustering Hierarchy for Wireless Microsensor Networks*", Fifth Annual Conference on Communication Networks and Services Research (CNSR), Frederlcton, NB, pp. 18-25, 14-17 May 2007, doi: 10.1109/CNSR.2007.27.

[24] Y. Zhang, Y. Gu, V. Vlatkovic, X. Wang, "*Progress of Smart Sensor and Smart Sensor Network*", IEEE Fifth World Congress on Intelligent Control and Automation, (WCICA), Hangzhou, P.R. China, 2004, doi: 10.1109/WCICA.2004.1343265.

[25] ISO 11898-1:2003, "*Road vehicles – Controller area network (CAN) – Part 1: Data link layer and physical signaling*".

[26] CiA 301 "*CANopen application layer and communication profile*" specification, available at: www.can-cia.org/standardization/specifications.

[27] K. Lee, R. Schneeman, “*Distributed measurement and control based on the IEEE 1451 smart transducer interface standards*”, IEEE Transaction on Instrumentation and Measurement, vol.49, (no. 3), pp. 621-627, Jun 2000.

[28] M. Dunbar, “*Plug-and-play sensors in wireless networks*”, IEEE Instrumentation & Measurement Magazine, vol. 4, no. 1, pp. 19-23, Mar 2001, doi: 10.1109/5289.911169.

[29] D. Wobschall, “*Networked sensor monitoring using the universal IEEE 1451 Standard*”, IEEE Instrumentation & Measurement Magazine, vol. 11, no. 2, pp.18-22, April 2008, doi: 10.1109/MIM.2008.4483729.

[30] National Institute of Standards and Technology (NIST), “IEEE 1451 Smart Transducer Interface Standard”. Available at: http://www.nist.gov/el/isd/ieee/ieee1451.cfm.

[31] *Standard for a Smart Transducer Interface for Sensors and Actuators – Common Functions, Communication Protocols, and Transducer Electronic Data Sheet (TEDS) Formats*, IEEE STD 1451.0-2077, IEEE Instrumentation and Measurement Society, TC-9, The Institute of Electrical and Electronics Engineers, Inc., New York, NY, October 5, 2007.

[32] *Standard for a Smart Transducer Interface for Sensors and Actuators – Network Capable Application Processor (NCAP) Information Model*, IEEE STD 1451.1-1999, IEEE Instrumentation and Measurement Society, TC-9, The Institute of Electrical and Electronics Engineers, Inc., New York, NY, June 26, 1999.

[33] *Standard for a Smart Transducer Interface for Sensors and Actuators – Transducer to Microprocessor Communication Protocols and Transduer Electronic Data Sheet (TEDS) Formats,* IEEE STD 1451.2-1997, IEEE Instrumentation and Measurement Society, TC-9, The Institute of Electrical and Electronics Engineers, Inc., New York, NY, September 25, 1998.

[34] *Standard for a Smart Transducer Interface for Sensors and Actuators – Digital Communication and Transducer Electronic Data Sheet (TEDS) Formats for Distributed Multidrop Systems*, IEEE STD 1451.3-2003, IEEE Instrumentation and Measurement Society, TC-9, The Institute of Electrical and Electronics Engineers, Inc., New York, NY, March 31, 2004.

[35] *Standard for a Smart Transducer Interface for Sensors and Actuators – Mixed-Mode Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats*, IEEE STD 1451.4-1994, IEEE Instrumentation and Measurement Society, TC-9, The Institute of Electrical and Electronics Engineers, Inc., New York, NY, December 15, 2004.

[36] *Standard for a Smart Transducer Interface for Sensors and Actuators – Wireless Communication and Transducer Electronic Data Sheet (TEDS) Formats*, IEEE STD 1451.5-2007, IEEE Instrumentation and Measurement Society, TC-9, The Institute of Electrical and Electronics Engineers, Inc., New York, NY, October 5, 2007.

[37] L. Ferrigno, V. Paciello, A. Pietrosanto, “*A Bluetooth-based proposal of instrument wireless interface*”, IEEE Trans. Instrum. Meas., vol. 54, pp. 163-170, 2005.

[38] K. Lee, E. Song, "*Wireless Sensor Network Based on IEEE 1451.0 and IEEE 1451.5-802.11*", 8th International Conference on Electronic Measurement and Instruments (ICEMI) 2007, Xi'an, pp. 4-7,4-11, 16-18 Aug., 2007, doi: 10.1109/ICEMI.2007.4351239.

[39] E. song, K. Lee, "*An interoperability test system for IEEE 1451.5-802.11 Standard,*" IEEE Sensors Applications Symposium (SAS) 2010, Limerick, pp. 183-188, 23-25 Feb., 2010, doi: 10.1109/SAS.2010.5439406.

[40] E. Song, K. Lee, "*An IEEE 1451.5-802.11 standard-based wireless sensor network with embedded WTIM,*" IEEE Instrum. Meas. Tech. Conference (I2MTC) 2011, Binjiang, pp. 1-6, 10-12 May, 2011, doi: 10.1109/IMTC.2011.5944344.

[41] E. Song, K. Lee, "*Testing system for IEEE 1451.5-802.11 standard-based wireless sensors,*" IEEE Instrum. Meas. Tech. Conference (I2MTC) 2014, Montevideo, pp. 862-867, 12-15 May, 2014, doi: 10.1109/I2MTC.2014.6860865.

[42] J.M. Larrauri, B.A. Larrinaga, M.L. Lopez, J.S. Cubillo, "*A Bluetooth sensor network based on the IEEE 1451 standard: A sensor network solution to evaluate the wellbeing of the passenger and improve safety in cars*," Int. Wireless Information Networks and Systems (WINSYS) 2010, Athens, pp. 1-6, 26-28 July, 2010.

[43] V.S. Harikrishnan, S. Irene, R. Pitchiah, "*Implementation of transducer electronic data sheet for ZigBee wireless sensors in smart building*," 7th International Conference on Sensing Technology (ICST) 2013, Wellington, pp. 906-911, 3-5 Dec, 2013, doi: 10.1109/ICSensT.2013.6727781.

[44] R. Seng, K. Lee, E. Song, "*An implementation of a wireless sensor network based on IEEE 1451.0 and 1451.5-6LoWPAN standards*," IEEE Instrum. Meas. Tech. Conference (I2MTC) 2011, Binjiang, pp. 1-6, 10-12 May, 2011, doi: 10.1109/IMTC.2011.5944334.

[45] J.E. Higuera, J. Polo, "*IEEE 1451 Standard in 6LoWPAN Sensor Networks Using a Compact Physical-Layer Transducer Electronic Datasheet*," IEEE Trans. on Instrum. Meas., vol. 60, no. 8, pp. 2751-2758, 2011, doi: 10.1109/TIM.20112129990.

[46] IEEE P1451.6-Proposed Standard for a High-Speed CANopen-Based Transducer Network Interface for Intrinsically Safe and Non-Intrinsically Safe Applications, (2007, May 25) [Online] Available http://grouper.ieee.org/groups/1451/6.

[47] *Standard for A Smart Transducer Interface for Sensors and Actuators – Transducers to Radio Frequency Identification (RFID) systems Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats*, IEEE STD 1451.7-2010, IEEE Instrumentation and Measurement Society, TC-9, The Institute of Electrical and Electronics Engineers, Inc., New York, NY, June 26, 2010.

[48] W. Luo, P. Han, R. Zhao, "*Study on Design and Application of Wireless sensor Network Based on Communication of Radio Frequency Identification System*," 5th International Conference on Wireless Communications,

Networking and Mobile Computing (WiCom) 2009, Beijing, pp. 1-6, 24-26 Sept., 2009, doi: 10.1109/WICOM.2009.5303533.

[49] H. Peizhao, R. Robinson, J. Indulska, "*Sensor Standards: Overview and Experiences*," International Conference on Intelligent Sensors, Sensor Networks and Information (ISSNIP) 2007, Melbourne, pp. 485-490, 3-6 Dec., 2007, doi: 10.1109/ISSNIP.2007.4496891.

[50] ISO/IEC/IEEE 21451-1:2010 *Information technology – Smart transducer interface for sensors and actuators – Part 1: Network Capable Application Processor (NCAP) information model*, IEEE Instrumentation and Measurement Society, TC-9, The Institute of Electrical and Electronics Engineers, Inc., New York, NY, May 15, 2010.

[51] ISO/IEC/IEEE 21451-4:2010 *Information technology – Smart transducer interface for sensors and actuators – Part 4: Mixed-mode communication protocols and Transducer Electronic Data Sheet (TEDS) formats*, IEEE Instrumentation and Measurement Society, TC-9, The Institute of Electrical and Electronics Engineers, Inc., New York, NY, May 15, 2010.

[52] R.R. Tenney, N.R. Sandell, "*Detection with Distributed Sensors*", IEEE Transaction on Aerospace and Electronic Systems, vol. 17, no. 4, pp. 501-510, July 1981.

[53] N.R. Sandell, P. Varaiya, M. Athans, M.G. Safonov, "*Survey of decentralized control methods for large scale systems*", IEEE Transaction on Automatic Control, vol 23, no. 2, pp. 108-128, April 1978, doi: 10.1109/TAC.1978.1101704.

[54] E.J. Keogh, M.J. Pazzani, "*An Enhanced Representation of Time Series which Allows Fast and Accurate Classification, Clustering and Relevance Feedback*", Fourth International Conference on Knowledge Discovery and Data Mining, 1998, New York, 27-31 August, 1998.

[55] J.F. Chamberland, V.V. Veeravalli, "*Decentralized detection in sensor networks*", IEEE Transactions on Signal Processing, vol. 51, no. 2, pp. 407-416, February 2003, doi: 10.1109/TSP.2002.806982.

[56] B.L. Kang, R.D. Schneeman, "*Distributed Measurement and Control Based on the IEEE 1451 Smart Transducer Interface Standards*", IEEE Transactions on Instrumentation and Measurement, vol. 49, no. 3, pp. 621-627, June 2000, doi: 10.1109/19.850405.

[57] G. Monte "*Sensor signal preprocessing techniques for analysis and prediction*" IEEE Industrial Electronics 34th Annual Conference, 2008. IECON 2008, pp. 1788-1793, 10-13 November 2008, Orlando (FL), doi: 10.1109/IECON.2008.4758225.

[58] G. Monte, V.K. Huang, P. Liscovsky, D. Marasco, "*Standard of things, first step: Understanding and normalizing sensor signals*", IEEE Industrial Electronics Society 39th Annual Conference, 2013. IECON 2013, pp. 118-123, 10-13 November 2013, Vienna, doi: 10.1109/IECON.2013.6699121.

[59] G. Monte, Z. Liu, F. Abate, V. Paciello, A. Pietrosanto, V. Huang, "*Normalizing Transducer Signals: An Overview of a Proposed Standard*", IEEE International Instrumentation and Measurement Technology

Conference (I2MTC) 2014, pp. 614-619, 12-15 May 2014, Montevideo, doi: 10.1109/I2MTC.2014.6860817.

[60] Keil Microcontroller Tools. Available at: www.keil.com.

[61] L. Klein, "*Sensor and Data Fusion Concepts and Applications*", SPIE Society of Photo-Optical Instrumentation Engineers, WA, 1999.

[62] D.L. Donoho, "*Compressed Sensing*", IEEE Transaction on Information Theory, vol. 52, no. 4, pp. 1289-1306, 2006.

[63] E.J. Caldès, M.B. Wakin, "An Introduction to Compressive Sampling," IEEE Signal Processing Magazine, March 2008, pp. 21-30.

[64] F. Bonavolontà, M. D'Arco, G. Ianniello, A. Licciardo, R. Schiano, L. Moriello, L. Ferrigno, M. Laracca, G. Miele, "*On the Suitability of Compressive Sampling for the Measurement of Electrical Power Quality*", Instrumentation and Measurement Technology conference (I2MTC), 2013, pp.126-131, Minneapolis 6-9 May 2013.

[65] E.J. Caldès, T. Tao, "*Decoding by linear programming*", IEEE Transaction on Information Theory, vol. 51, no.12, pp. 4203-4215, December 2005.

[66] E.J. Caldès, T. Tao, "*Near-optimal signal recovery from random projections: Universal encoding strategies?*" IEEE Transaction on Information Theory, vol 52, no. 12, pp. 5406-5425, Dec. 2006.

[67] http://csegroups.case.edu/bearingdatacenter/pages/welcome-case-western-reserve-university-bearing-data-center-website.