

IEEE Recommended Practice for Signal Treatment Applied to Smart Transducers

IEEE Industrial Electronics Society

Sponsored by the
Industrial Electronics Society Standards Committee

IEEE Instrumentation and Measurement Society

Sponsored by the
Sensor Technology Committee
and the
Signals and Systems in Measurement Committee

IEEE
3 Park Avenue
New York, NY 10016-5997
USA

IEEE Std 21451-001™-2017

IEEE Recommended Practice for Signal Treatment Applied to Smart Transducers

Sponsors

Industrial Electronics Society Standards Committee
of the
IEEE Industrial Electronics Society

and

Sensor Technology Committee
and
Signals and Systems in Measurement Committee
of the
IEEE Instrumentation and Measurement Society

Approved 23 March 2017

IEEE-SA Standards Board

Abstract: Signal processing algorithms and data structure are defined in this recommended practice in order to share and to infer signal and state information of an instrumentation or control system. These algorithms are based on the signal and transducers attached to the system. This recommended practice also defines a set of transducer signal treatment services based on signal treatment algorithms as application programming interface (API), which is used for applications to use or call these transducer signal treatment services.

Keywords: IEEE 1451™, IEEE 21451™, oversampling, segmentation, serial, signal shape analysis, smart actuator, smart filtering, smart sensor, smart transducer, time-domain signal analysis

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Copyright © 2017 by The Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 20 December 2017. Printed in the United States of America.

IEEE is a registered trademark in the U.S. Patent & Trademark Office, owned by The Institute of Electrical and Electronics Engineers, Incorporated.

CORBA is a registered trademark of Object Management Group, Inc. in the United States and/or other countries.

Java is a registered trademark of Oracle and/or its affiliates.

Microsoft, Visual C++, Visual C#, Visual Studio, and Windows are registered trademarks of Microsoft Corporation in the United States and/or other countries.

PDF: ISBN 978-1-5044-4054-7 STD22608
Print: ISBN 978-1-5044-4055-4 STDPD22608

IEEE prohibits discrimination, harassment, and bullying.

For more information, visit <http://www.ieee.org/web/aboutus/whatis/policies/p9-26.html>.

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page, appear in all standards and may be found under the heading “Important Notices and Disclaimers Concerning IEEE Standards Documents.” They can also be obtained on request from IEEE or viewed at <http://standards.ieee.org/IPR/disclaimers.html>.

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents (standards, recommended practices, and guides), both full-use and trial-use, are developed within IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (“IEEE-SA”) Standards Board. IEEE (“the Institute”) develops its standards through a consensus development process, approved by the American National Standards Institute (“ANSI”), which brings together volunteers representing varied viewpoints and interests to achieve the final product. IEEE Standards are documents developed through scientific, academic, and industry-based technical working groups. Volunteers in IEEE working groups are not necessarily members of the Institute and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE Standards do not guarantee or ensure safety, security, health, or environmental protection, or ensure against interference with or from other devices or networks. Implementers and users of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.

IEEE does not warrant or represent the accuracy or content of the material contained in its standards, and expressly disclaims all warranties (express, implied and statutory) not included in this or any other document relating to the standard, including, but not limited to, the warranties of: merchantability; fitness for a particular purpose; non-infringement; and quality, accuracy, effectiveness, currency, or completeness of material. In addition, IEEE disclaims any and all conditions relating to: results; and workmanlike effort. IEEE standards documents are supplied “AS IS” and “WITH ALL FAULTS.”

Use of an IEEE standard is wholly voluntary. The existence of an IEEE standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE should be considered the approved IEEE standard.

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, or be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position of IEEE.

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE. However, IEEE does not provide consulting information or advice pertaining to IEEE Standards documents. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to comments or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in revisions to an IEEE standard is welcome to join the relevant IEEE working group.

Comments on standards should be submitted to the following address:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854 USA

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under U.S. and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, IEEE does not waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate fee, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Updating of IEEE Standards documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

Every IEEE standard is subjected to review at least every ten years. When a document is more than ten years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE Xplore at <http://ieeexplore.ieee.org/> or contact IEEE at the address listed previously. For more information about the IEEE-SA or IEEE's standards development process, visit the IEEE-SA Website at <http://standards.ieee.org>.

Errata

Errata, if any, for all IEEE standards can be accessed on the IEEE-SA Website at the following URL: <http://standards.ieee.org/findstds/errata/index.html>. Users are encouraged to check this URL for errata periodically.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE-SA Website at <http://standards.ieee.org/about/sasb/patcom/patents.html>. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patent Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

Participants

At the time this recommended practice was completed, the Signal Treatment for Smart Transducer Working Group had the following membership:

Gustavo Monte, *Chair*
Ruqiang Yan, *Vice Chair*

Francesco Abate
Allen Chen
Antonio Espirito Santo
Sabrina Guia

Rodica Holonec
Victor Huang
Kang Lee
Zheng Liu
Vincenzo Paciello

Antonio Pietrosanto
Helena Ramos
Eugene Song
Hubert Zangl

The following members of the individual balloting committee voted on this recommended practice. Balloters may have voted for approval, disapproval, or abstention.

Mihaela Albu
Frank Basciano
William Bloethe
Riccardo Brama
Dietmar Bruckner
Paul Cardinal
Cheng-Jen Chen
Keith Chow
Helena Maria Geirinhas Ramos
Randall Groves
Werner Hoelzl
Rodica Holonec
Victor Huang

Atsushi Ito
Robert Johnson
Kenneth Karg
John Kay
Kang Lee
Michael Maytum
Tremont Miao
Gustavo Monte
Arthur Neubauer
Michael Newman
Vincenzo Paciello
Sergio Rapuano

Robert Robinson
Bartien Sayogo
Christopher Searles
Robert Seitz
Suresh Shrimavle
Jeremy Smith
Yuyin Song
Joseph Stanco
Thomas Strasser
John Vergis
Stephen Webb
Ruqiang Yan
Oren Yuen

When the IEEE-SA Standards Board approved this recommended practice on 23 March 2017, it had the following membership:

Jean-Philippe Faure, *Chair*
Gary Hoffman, *Vice Chair*
John D. Kulick, *Past Chair*
Konstantinos Karachalios, *Secretary*

Chuck Adams
Masayuki Ariyoshi
Ted Burse
Stephen Dukes
Doug Edwards
J. Travis Griffith
Michael Janezic

Thomas Koshy
Joseph L. Koepfinger*
Kevin Lu
Daleep Mohla
Damir Novosel
Ronald C. Petersen
Annette D. Reilly

Robby Robson
Dorothy Stanley
Adrian Stephens
Mehmet Ulema
Phil Wennblom
Howard Wolfman
Yu Yuan

*Member Emeritus

Introduction

This introduction is not part of IEEE Std 21451-001-2017, IEEE Recommended Practice for Signal Treatment Applied to Smart Transducers.

IEEE Std 21451-001, a recommended practice, defines signal processing algorithms and data structure in order to share and infer signal and state information of instrumentation or control systems. These algorithms are based on acquired signals and the transducers attached to the systems. This recommended practice also defines the commands and responses for requesting information and algorithms for shape analysis, such as exponential, sinusoidal, impulsive noise, noise, and tendency.

Contents

1. Overview	10
1.1 Scope	10
1.2 Purpose	10
1.3 Conformance	10
2. Normative references.....	11
3. Definitions, abbreviations, and units	11
3.1 Definitions	11
3.2 Abbreviations	11
3.3 Units	12
4. Need for the recommended practice.....	12
4.1 Introduction	12
4.2 A Simple example of application.....	14
4.3 Outline of the recommended practice.....	14
5. Smart transducer functional specification	15
5.1 ISO/IEC/IEEE 21451 reference model.....	15
5.2 User's network.....	15
5.3 NCAP network interface.....	16
5.4 ISO/IEC/IEEE 21450 NCAP transducer services.....	16
5.5 ISO/IEC/IEEE 21450/IEEE 1451.X NCAP module communication interface	16
5.6 Physical interface between NCAP and TIM.....	16
5.7 ISO/IEC/IEEE 21450/IEEE 1451.X TIM communication interface	16
5.8 ISO/IEC/IEEE 21450 TIM transducer services.....	16
5.9 IEEE 21451-001 TIM transducer signal treatment services	16
5.10 Signal conditioner and data conversion	17
5.11 Transducers.....	17
6. Algorithms for signal description	17
6.1 Introduction	17
6.2 Segmentation algorithm.....	17
6.3 Labeling description	19
6.4 Algorithm for real-time segmentation and labeling (RTSAL).....	22
6.5 Description of RTSAL algorithm	24
7. Transducer signal features extraction	24
7.1 Introduction	24
7.2 Initial parameters	25
7.3 Variable definitions	25
8. First layer algorithms.....	26
8.1 Introduction	26
8.2 First layer functions	26
8.3 Exponential pattern detection	26
8.4 Noise detection	28
8.5 Impulsive noise detection	32
8.6 Sinusoidal pattern detection.....	35
8.7 Tendency estimation.....	40
8.8 Mean estimation	42

9. Services required for the first layer algorithms	43
10. Prediction.....	44
10.1 Introduction	44
10.2 Steady-state signal value estimation	44
10.3 Prediction for damped oscillations	46
11. Filtering	46
11.1 General	46
11.2 Filtering based on iteration selection	49
12. Compression	53
12.1 General	53
12.2 Example	53
12.3 Algorithm	55
13. Complex pattern analysis: impulsive-like signals	58
13.1 Detection of QRS pattern in ECG signals.....	58
13.2 Unexpected segments	59
14. User-defined application code	61
15. Transducer algorithm data sheet (TADS).....	61
16. IEEE 21451-001 transducer signal treatment services API.....	62
17. Implementation guide	71
Annex A (informative) Bibliography	73
Annex B (informative) Strategies of samples selection.....	74
Annex C (informative) An example of the first layer-sinusoidal pattern detection implementation.....	78

IEEE Recommended Practice for Signal Treatment Applied to Smart Transducers

1. Overview

1.1 Scope

This recommended practice defines signal processing algorithms and data structure in order to share and to infer signal and state information of an instrument or control system. These algorithms are based on their own signal and also on the transducers attached to the system. The recommended practice also defines the commands and replies for requesting information and algorithms for shape analysis such as exponential, sinusoidal, impulsive noise, noise, and tendency.

1.2 Purpose

The purpose is to define a standardized and universal framework that allows smart transducers to extract features of the signal being generated and measured. With the definition of these practices, the raw data can be converted into information and then into knowledge. In this context, knowledge means understanding of the nature of the transducer signal. This understanding can be shared with the system and other transducers in order to form a platform for sensory knowledge fusion.

1.3 Conformance

Conformance to IEEE Std 21451-001™ requires that all non-optional clauses be implemented in the vendor device.

Several keywords are used to differentiate among various levels of requirements and optionally, as follows:

shall: The key word “shall” indicates a mandatory requirement. Designers are required to implement all such mandatory requirements to promote interoperability with other products that conform to the specifications for IEEE Std 21451-001.

recommended: Recommended is a key word indicating flexibility of choice with a strong preference alternative. The word “should” has the same meaning.

should: Should is a key word indicating flexibility of choice with a strong preference alternative. The phrase “it is recommended” has the same meaning.

may: May is a key word that indicates flexibility of choice with no implied preference.

2. Normative references

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used; therefore, each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

IEEE P1451.2/D32 (April 2016), Draft Standard for a Smart Transducer Interface for Sensors and Actuators—Serial Point-to-Point Interface.¹

IEEE Std 1451.5™-2007, IEEE Standard for a Smart Transducer Interface for Sensors and Actuators—Wireless Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats.^{2,3}

ISO/IEC/IEEE 21450:2010, Information technology — Smart transducer interface for sensors and actuators — Common Functions, Communication Protocols, and Transducer Electronic Data Sheet (TEDS) formats.⁴

ISO/IEC/IEEE 21451-7:2011, Information technology — Smart transducer interface for sensors and actuators — Part 7: Transducer to radio frequency identification (RFID) systems communication protocols and Transducer Electronic Data Sheet (TEDS) formats.

3. Definitions, abbreviations, and units

3.1 Definitions

For the purposes of this document, the following terms and definitions apply. The *IEEE Standards Dictionary Online* should be consulted for terms not defined in this clause.⁵

impulsive noise: Interference on a signal caused by a short period of noise.

oversampling: A sampling frequency that is much greater than the Nyquist frequency required to avoid aliasing in uniform sampling.

tendency: Long-term behavior of a transducer signal.

uniform sampling: In a continuous-time to discrete-time signal conversion, when the time interval between samples is constant.

3.2 Abbreviations

CORBA®	Common Object Request Broker Architecture ^{6, 7}
CPU	central processing unit
ECG	electrocardiogram

¹ Numbers preceded by P are IEEE authorized standards projects that were not approved by the IEEE-SA Standards Board at the time this publication went to press. For information about obtaining drafts, contact the IEEE.

² The IEEE standards or products referred to in this clause are trademarks owned by The Institute of Electrical and Electronics Engineers, Incorporated.

³ IEEE publications are available from The Institute of Electrical and Electronics Engineers (<http://standards.ieee.org/>).

⁴ ISO/IEC publications are available from the International Organization for Standardization (<http://www.iso.org/>), the International Electrotechnical Commission (<http://www.iec.ch/>), and the American National Standards Institute ([http://www.ansi.org/](http://www ansi.org/)).

⁵ *IEEE Standard Dictionary Online* is available at <http://dictionary.ieee.org>.

⁶ CORBA is a registered trademark of Object Management Group, Inc. in the United States and/or other countries.

⁷ This information is given for the convenience of users of this standard and does not constitute an endorsement by the IEEE of these products. Equivalent products may be used if they can be shown to lead to the same results.

IDL	Interface Definition Language
LSB	least significant bit
MCT	mark, class, and time (i.e., the three vectors that represent the sampled transducer signal)
NCAP	network-capable application processor
POM	point of measurement (in a sensor)
QRS	a wave pattern related to electrocardiograms (i.e., wave Q, wave R, and wave S)
RTSAL	real-time segmentation and labeling
TADS	transducer algorithm data sheet
TEDS	transducer electronic data sheet (see IEEE Std 1451.0™-2007 [B3], 3.1.37) ⁸
TIM	transducer interface module (see IEEE Std 1451.0-2007, 3.1.33)

3.3 Units

UInt32	unsigned 32-bit integer
UInt16	unsigned 16-bit integer
UInt8	unsigned 8-bit integer
Float32	a single-precision real number, i.e., a 32-bit binary sequence
Float64	a double-precision real number, i.e., a 64-bit binary sequence
_String	string data type
_Boolean	Boolean data type
UInt32Array	an arbitrary number of 32-bit signed integers
UInt16Array	an arbitrary number of 16-bit signed integers
UInt8Array	an arbitrary number of 16-bit signed integers
Float32Array	an arbitrary number of single-precision real numbers
Float64Array	an arbitrary number of double-precision real numbers
StringArray	an arbitrary number of _String data types
BooleanArray	an arbitrary number of _Boolean data types

4. Need for the recommended practice

4.1 Introduction

The real world is inferred through sensor signals. The sensors and actuators can do much more than what humans usually use them for, i.e., performing their basic functions. The expanded application of embedded systems has direct impact on transducer design. The all-inclusive microcontrollers and the application-specific integrated circuits offer everything needed to turn a regular transducer into a smart transducer, with the integration of a sensor element into a single unit. General sensor signal processing today is basically the same as 10 years ago. While the processing power and memory size of the central processing unit (CPU) have grown exponentially, the costs of sensor signal processing remain relatively constant. Taking into account the size, power consumption, cost, etc., a paradigm shift in the sensor world is on the horizon: the signal will be processed entirely at the point of measurement (POM). Although today this is only relevant to sensors with complex signal processing related to specific issues, it will occur for most transducers in the future, even for small and low-cost sensors. However, this issue is out of the scope of this recommended practice.

⁸ The numbers in brackets correspond to the numbers of the bibliography in Annex A.

No universal treatment for transducer signals allows signal synthesis and analysis and a dialogue among transducers to validate the measurement processes and to provide a platform for data mining. This recommended practice allows smart transducers from different manufacturers to interchange signal information in order to infer states of the instrumentation system, and such capability helps to achieve higher reliability and better real-world inference.

Once the sensor signal is digitized, a great amount of data will be generated. If there are dozens or hundreds of sensors, it may be desirable not to interchange these data because, most of the time, there is no need to do it. Besides, the data flow is restricted by the transmission channel capacity. Since the trend is to process the signal in the POM instead of raw data, there will be a need to interchange knowledge of the sensor signal. Therefore, taking sensor fusion into account, there is a requirement for processing these data from the transducer to obtain data and then convert them into knowledge. For some specific applications, this knowledge can be obtained directly from samples. For multiple objectives, an intermediary platform is needed to simplify and speed up the signal processing as illustrated in Figure 1. This signal treatment should be carried out in a normalized scheme in order to allow interchange of knowledge among transducers. In this context, knowledge means understanding of the nature of the transducer signal. ***This understanding must be shared with the system and transducers in order to build a platform for sensory knowledge fusion.*** This point is the main objective of this recommended practice.

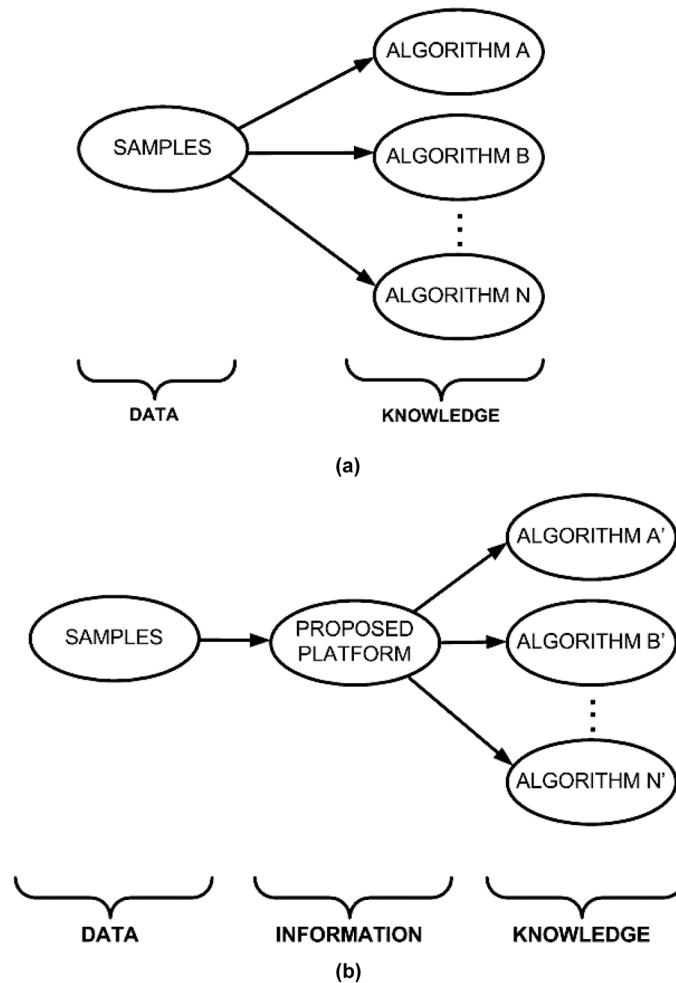


Figure 1—Intermediary platforms. (a) Classical approach – from data to knowledge.
(b) IEEE 21451-001 scheme – from data to information to knowledge.

4.2 A Simple example of application

A simple example may help to understand the aforementioned concepts. For example, a tank contains an actuator, e.g., a heater, and a temperature sensor. As part of the process, the temperature of the liquid inside the tank must be raised. Therefore, the actuator receives a command to turn on the heater. If everything is correct, the heater resistance is in range, there is liquid in the tank, and the temperature sensor works properly. The liquid temperature will start to increase, and it will be measured by the sensor. A simple question with a yes/no answer may validate the whole process. The actuator asks the sensor if the temperature signal exhibits an exponential growth. If the answer is a yes, the normal function of the entire system, i.e., actuator-liquid-sensor, is validated. If not, a problem is detected immediately after it occurs. The traditional way to detect an abnormal function is to trigger an alarm due to a timeout. If the actuator is a brand A and the temperature sensor is a brand B, a standardized sensor signal treatment is necessary to make this example application a reality. In addition, since the temperature sensor knows its signal shape, it can predict the steady-state value or the time to reach a desired value.

4.3 Outline of the recommended practice

Standard-based approaches offer many advantages over proprietary solutions, especially when the application is universal. What kind of knowledge should be included in this recommended practice? As a first approximation, it should be basic information of sensor signal shape, impulsive noise, oscillations, noise, mean, and tendency. The aim is to include similar information that may be provided for an observer to monitor the signal. The following clauses present the approach:

- Clause 5 describes the functional specification of smart transducers.
- Clause 6 presents the algorithm description with emphasis on the universal treatment that can be applied to all sensor signals. These algorithms implement a smart sampling process. The segmentation and labeling process is to obtain the raw information to infer a global behavior and to extract important features of the signal.
- Clause 7 presents the recommended practice structure that includes representative signal shape parameters.
- Clause 8 presents the first layer of the advanced feature extraction that may be embedded into a smart transducer.
- Clause 9 describes the services required from the first layer algorithms.
- Clause 10 presents the prediction algorithm.
- Clause 11 describes the algorithm for filtering.
- Clause 12 presents the compression algorithm.
- Clause 13 presents the algorithm to be implemented for a complex pattern analysis.
- Clause 14 describes the capability to load a user-defined application code.
- Clause 15 presents the structure for a transducer algorithm data sheet (TADS).
- Clause 16 describes the application programming interface (API).
- Clause 17 provides guidance for implementing the set of standardized transducer signal treatment services defined in this recommended practice.

For the convenience of the reader, variable names have been italicized in this document.

5. Smart transducer functional specification

5.1 ISO/IEC/IEEE 21451 reference model

This clause provides a description of the functions to be included in ISO/IEC/IEEE 21451 smart transducers. The reference model presented in Figure 2 shows the relationships among the various documents of the ISO/IEC/IEEE 21451 family of standards. The model is used to show the relationship between each standard and to define what is in each of the standards. It is not intended to imply a required implementation of devices that comply with the standards. Subclauses 5.2 through 5.10 give a further description of the elements shown in Figure 2.

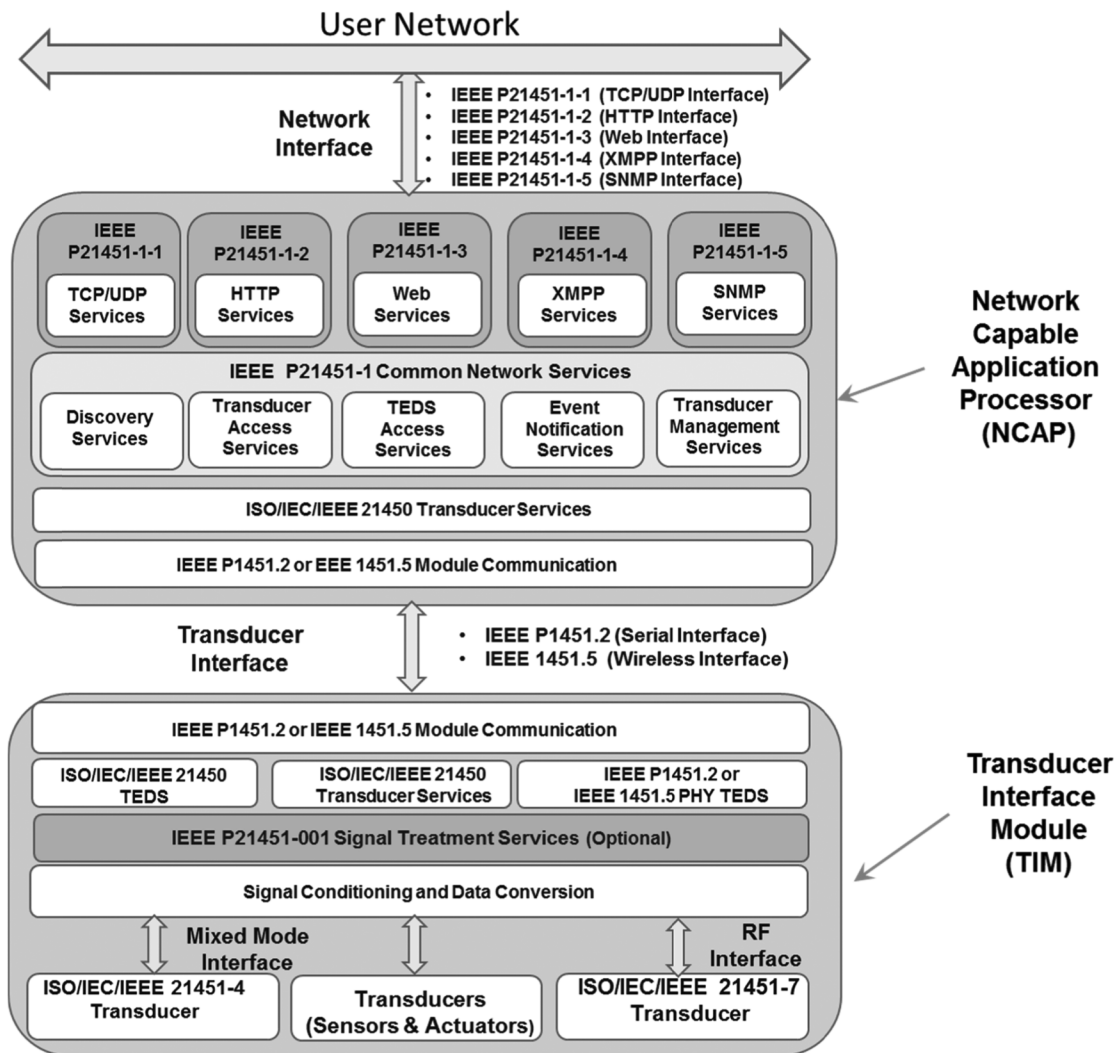


Figure 2—ISO/IEC/IEEE 21451 family reference model

5.2 User's network

The user's network is beyond the scope of the ISO/IEC/IEEE 21451 family of standards. The network-capable application processor (NCAP), shown in Figure 2, has the network interface. Different NCAPs are required for networks using different physical communications media or protocols. The network types are not defined in ISO/IEC/IEEE 21451.

5.3 NCAP network interface

IEEE P21451-1 [B1], shown in Figure 2, defines a set of common network services for ISO/IEC/IEEE 21451 smart transducers. The services consist of discovery services, transducer data access services, transducer electronic data sheet (TEDS) access services, event notification services, and transducer manager services in order to achieve sensor data interoperability at the network level. The NCAP network interface shown in Figure 2 includes IEEE P21451-1-1, IEEE P21451-1-2, IEEE P21451-1-3, IEEE P21451-1-4 [B2], and IEEE P21451-1-5 services, which implement IEEE P21451-1 common network services in IP, HTTP, Web Services, XMPP, and SNMP services, respectively.

5.4 ISO/IEC/IEEE 21450 NCAP transducer services

The ISO/IEC/IEEE 21450 transducer services are common services for transducers. These functions include the command/response sets and TEDS. Features are included in the standard to support capabilities, such as triggering and synchronous sampling of sensors, but detailed specification is defined in each relevant document of the family of standards.

5.5 ISO/IEC/IEEE 21450/IEEE 1451.X NCAP module communication interface

The module communication interface is used to pass information across the interface between the ISO/IEC/IEEE 21450 NCAP services and the IEEE 1451.X NCAP module communication. IEEE P1451.2 and IEEE Std 1451.5-2007 are examples of the IEEE 1451.X standards.⁹

5.6 Physical interface between NCAP and TIM

The physical interface between the NCAP and the transducer interface module (TIM) is defined in the other documents of the IEEE 1451.X family of standards. Functions needed across this interface, such as synchronization, if applicable, are described, but not their implementation. IEEE P1451.2 and IEEE Std 1451.5-2007 are examples of the IEEE 1451.X standards.

5.7 ISO/IEC/IEEE 21450/IEEE 1451.X TIM communication interface

The ISO/IEC/IEEE 21450/IEEE 1451.X TIM communication interface is logically the complement of the ISO/IEC/IEEE 21450/IEEE 1451.X NCAP module communication interface. The ISO/IEC/IEEE 21450/IEEE 1451.X TIM communication interface is defined in the corresponding documents of the IEEE 1451.X family of standards. IEEE P1451.2 and IEEE Std 1451.5-2007 are examples of the IEEE 1451.X standards.

5.8 ISO/IEC/IEEE 21450 TIM transducer services

The ISO/IEC/IEEE 21450 TIM transducer services are logically the complement of the ISO/IEC/IEEE 21450 NCAP transducer services.

5.9 IEEE 21451-001 TIM transducer signal treatment services

This recommended practice defines a set of transducer signal treatment services, which include signal processing algorithms (signal segmentation and labeling algorithm, signal features extraction and/or detection algorithm, and advanced feature extraction or filtering algorithm) and data structure in order to share and to infer signal and state information of an instrument or control system. These algorithms are based on the acquired signal and also on the transducers attached to the system. This recommended practice also defines an application programming interface (API), which is a set of request/response pairs discussed in detail in Clause 13.

⁹ Information on normative references can be found in Clause 2.

5.10 Signal conditioner and data conversion

The details of the signal conditioning and data conversion functions are beyond the scope of this recommended practice.

5.11 Transducers

Transducers may be classified into three categories:

- ISO/IEC/IEEE 21451-4 transducers
- ISO/IEC/IEEE 21451-7 transducers
- General transducers

For ISO/IEC/IEEE 21451-4 transducers, the interface between transducers and signal conditioning is the ISO/IEC/IEEE 21451-4 mixed-mode interface (MMI). For the ISO/IEC/IEEE 21451-7 transducers, the interface between transducers and signal conditioning is the ISO/IEC/IEEE 21451-7 radio frequency (RF) interface. For general transducers, the interface between transducers and signal conditioning is beyond the scope of this recommended practice.

6. Algorithms for signal description

6.1 Introduction

This clause defines and explains the main algorithm that conforms to the basic structure, which is the input for all other processes defined in this recommended practice.

A time-domain algorithm is a building block for signal analysis. It is based on signal segmentation following a labeling process. As a result, a smart transducer realizes what kind of signal it is measuring or generating. The computational cost of the algorithm is extremely low so it can be built into smart transducers and executed in real time to provide a standardized platform for sensor signal identification and sensory information fusion.

6.2 Segmentation algorithm

By looking at the signal in Figure 3, a person can learn the characteristics of the signal. The signal may be interpreted as “noisy” in the beginning. Then it reaches a maximum and a minimum and grows and falls rapidly with a spike-like shape. After that, it starts to increase and then tends toward a stable value. The observers are aware of the signal behavior since they relate shape at different instants. Therefore, it is desirable to have a more complete sampling process in order to take into account sample sequences. The key process is to identify what the signal trajectory between samples is. It is not necessary to do this for every pair of samples. Therefore, the first process is to segment the signal and then relate these segments.

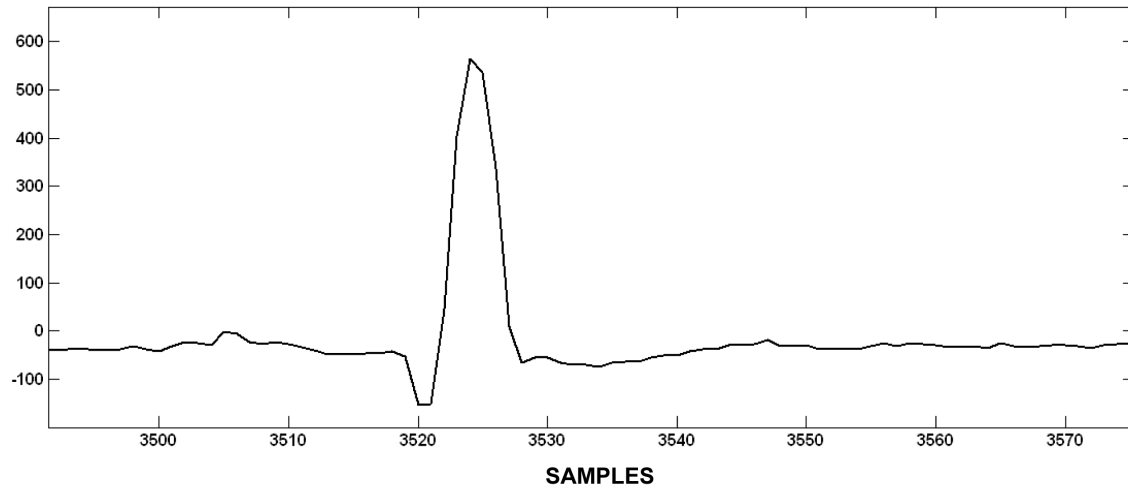


Figure 3—A typical sensor signal

Let $x(t)$ be the analog sensor signal with band limited to B Hz and uniformly sampled at a frequency F_s , which is much greater than $2B$. As the signal is oversampled, there will be redundancy between adjacent samples. The algorithm starts interpolating between two samples, i.e., interpolates sample 2 from 1 and 3. Then, the error, i.e., the difference between the real sample and the interpolated sample, is computed. If the error is less than a predefined value, the sample is relatively unimportant because its value can be obtained from its neighbors. Now, sample 3 is interpolated from 1 and 5, and so on. If the error exceeds a threshold, the sample is tagged as important, and the process starts again from this sample. The interpolation is implemented with linear interpolation. If no important sample is found during N samples, the last sample is tagged, and the process starts again from this sample. The tagged samples will be the left and right samples of the segment. Figure 4 shows an example of a segment.

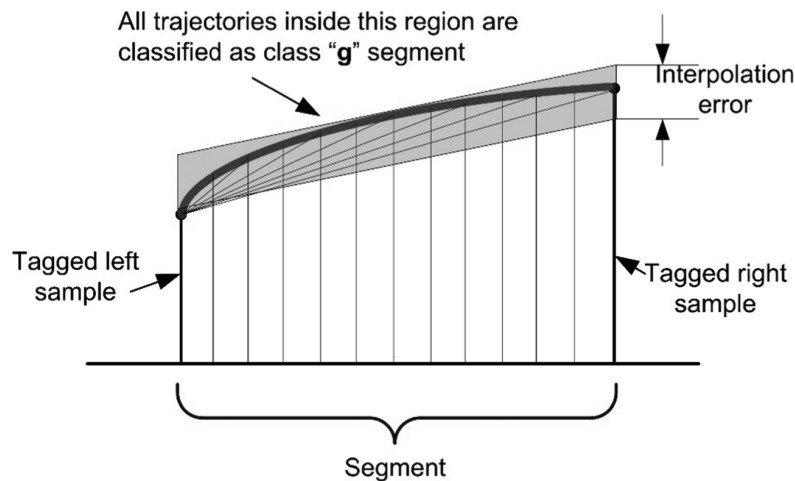


Figure 4—Example of segmentation and labeling

As a result of the algorithm, two vectors are obtained: the tagged or essential samples vector $mark(n)$, and the temporal index vector $time(n)$. The algorithm is very simple and easy to implement; nevertheless, it allows real-time signal segmentation and a platform for the segment shape analysis. Only a division operation by two is involved in the average computation.

6.3 Labeling description

Each time the difference is computed in the interpolation process, the sign of the error is recorded. When the segment ends, based on the majority of accumulated error signs, the behavior inside the segment can be classified. The error sign is the clue to classify the segment. If the signal is oversampled, the trajectory of the real signal inside the segment is restricted by the interpolation error.

Considering the value of the left and right samples and the sign of the interpolation error, it is possible to see the simplified shapes shown in Table 1 and Table 2. In Table 1, the interpolation error does not reach a threshold; therefore, the segment ends as it has reached the maximum size of N samples. In Table 2, the segments end due to interpolation error.

NOTE—*Why it is necessary to have an oversampling frequency?* The reason is that it is desirable to interpolate faster than the signal rate of change. In this way, it is possible to obtain a uniform behavior within the segment, i.e., most of the interpolation errors are all negative or they are all positive.¹⁰

Table 1—Segment classes a, b, and c

Class	Segment
a	
b	
c	

Table 2—Segment classes d, e, f, g and h

Class	Segment
d	
e	
f	
g	
h	

¹⁰ Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement the standard.

NOTE—The segments “class h” are not important, and they should be avoided. If there is oversampling AND the interpolation error is small, the probability of segments h is close to zero. But if these conditions are not satisfied, it is possible to have the h segments. The best way to deal with this is to use the majority of signal error signs to determine the segment class. The only chance to have h segments is to have the same amount of positive and negative error signs. In this case, the algorithm waits for the next sample until an unbalance occurs.

A new vector, *class(n)*, is added to the sensor signal description, whose elements are the segment classes **a**, **b**, **c**, **d**, **e**, **f**, **g**, and/or **h**. The set of vectors *mark(n)*, *class(n)*, and *time(n)* (in short, MCT) conforms to the fundamental structure from which arises all the IEEE 21451-001 algorithms, and it is the IEEE 21451-001 platform that captures information from data. The MCT vectors can be seen as a smart sampling process where the relationship among essential samples is clearly stated by the class vector. Why is there the need for oversampling? It is to obtain a uniform behavior within the segment, i.e., most of the interpolation errors are all negative or all positive. Under conditions of oversampling and interpolation error approaching zero, the probability of occurrence of segments **a**, **b**, **c**, and **h** tends to be zero. Therefore, only four classes of segments are important (**d**, **e**, **f**, and **g**), and the *sequence* of them identifies the sensor signal. The vector *class(n)* is raw information in order to imitate a human observer. The vector *class(n)* is amplitude and time scale invariant. It captures only the shape information. For example, the pattern “**gefd**” or “**gggeeeeffffddd**” captures a sinusoidal waveform, and the sequence “**ddddffff**” codifies a positive spike-like signal. The frequency and amplitude can be obtained from the MCT vectors. In addition, the *class(n)* vector provides precise localization of the signal local maxima and minima. Maxima occur at the union of “**df**,” “**de**,” “**gf**,” and “**ge**” segments. Minima occur at the union of “**eg**,” “**ed**,” “**fg**,” and “**fd**” segments. It is important to note that the local minima and maxima are controlled by the interpolation error.

The *class(n)* vector is the “missing link” from the analog to the digital “world” in the traditional digitalization process or uniform sampling because it naturally captures the shape information of the signal. Important signal features can be inferred from the sequence of segments. For example, a sequence of class “**g**” segments indicates that the signal has an exponential shape that tends toward a steady-state value if the lengths of the segments are increasing. A sequence of class “**d**” segments indicates that the signal is growing rapidly, and a sequence of class “**f**” segments indicates that it is decaying quickly. Table 3 gives some examples of waveform identification looking for sequences of segments.

Table 3—Basic examples of waveform identification

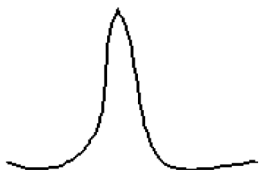
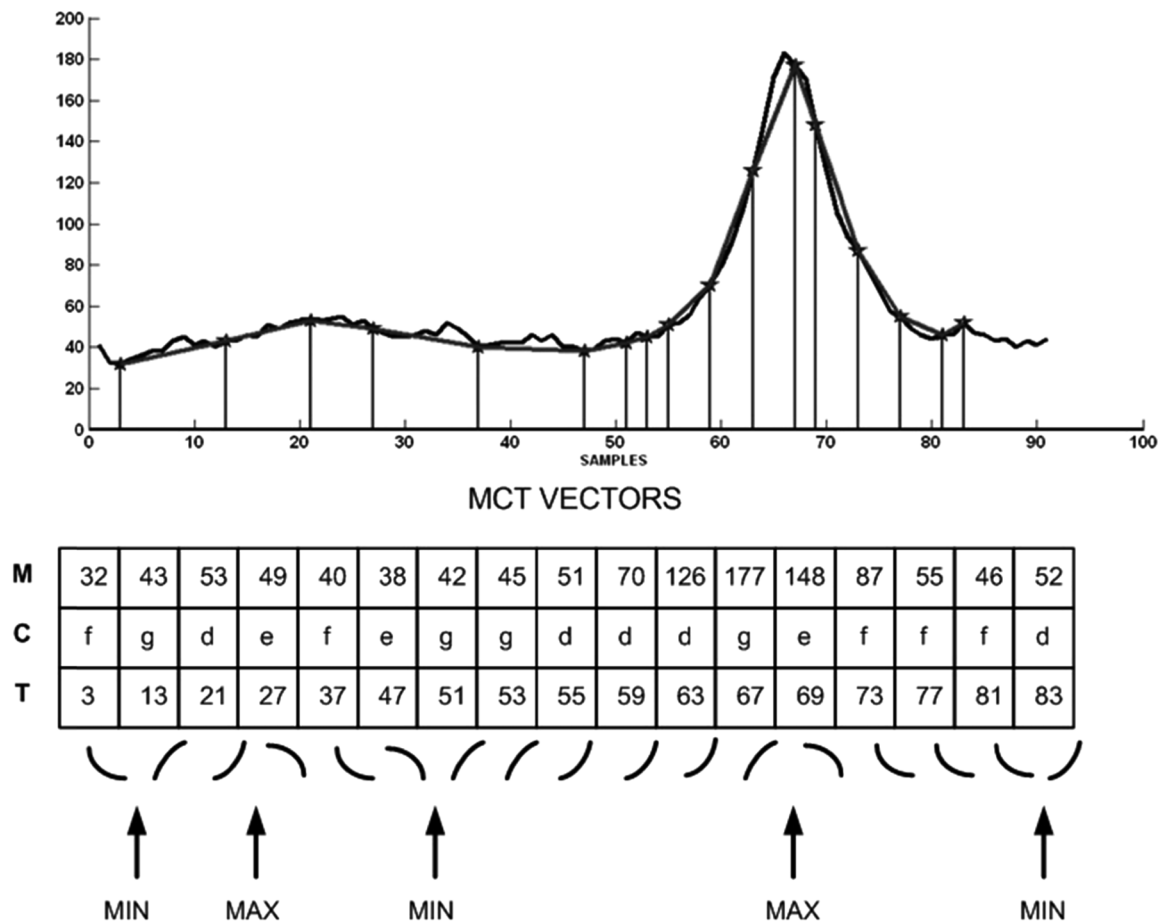
Waveform	Pattern
	Sequence of segments class g
	Segments class g followed by class f , maximum at union of gf
	Sequence class dgef , maximum at union of ge
	Sequence class eg , minimum at union eg segments
	Segments class df , maximum at union of df segments
	Sequence class gefd , maximum at union of ge , minimum at union of fd segments

Figure 5 shows an example of MCT vectors. The local maxima and minima are controlled by the interpolation error, and they are obtained in real time. Maxima and minima are good estimators of frequency content, and they are used for the smart filtering algorithm.



NOTE: It is important to remark that the interpolation maximum and minimum are controlled by the interpolation error.

Figure 5—MCT vectors of a test signal, interpolation error 3 LSB

6.4 Algorithm for real-time segmentation and labeling (RTSAL)

Figure 6 shows an algorithm for segmentation and labeling. This algorithm can be executed in real time. It needs only three samples to do the linear interpolation based on the designated interpolation error magnitude and sign. The input parameters are the maximum segment length (N) and the interpolation error (**Err**). The algorithm outputs are the MCT vectors, and the size of these vectors should be adapted to the memory available on the transducer. The algorithm starts by initializing parameters and output vectors. The first sample is tagged as the left essential sample. Then, it waits for two more samples to do the linear interpolation. The difference between the middle sample and the interpolated sample is checked against the interpolation error (**Err**). The error sign defines if the real signal is above or below the interpolated sample, and it provides information to determine the class segment. The segment ends for two reasons: 1) the interpolation error is greater than **Err**, or 2) the length of the segment reaches N samples. In the first case, based on the **Up** and **Dwn** variables and the values of the first and last sample, the segment class is defined. In the second case, the segment class is **a**, **b**, or **c** based on the values of the first and last sample.

The MCT vectors are updated, in circular structure; the last sample is the first sample for the new segment; and the process starts again. Note that the MCT vectors are obtained in real time.

Input parameters:
N = Max segment length
Interpolation error= Err

ALGORITHM FOR REAL TIME SEGMENTATION AND LABELING

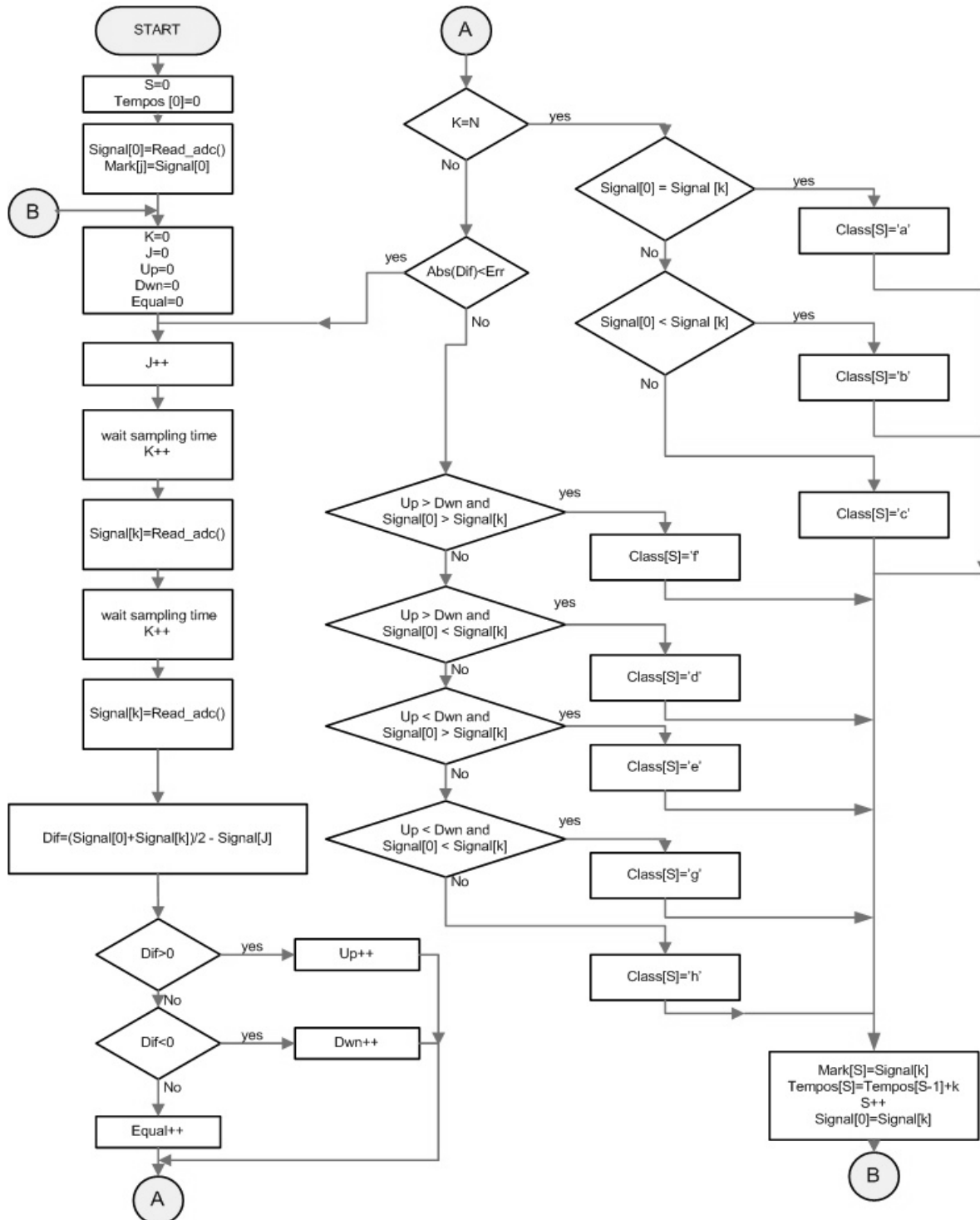


Figure 6—Flowchart of the segmentation and labeling algorithm

6.5 Description of RTSAL algorithm

NOTE—Refer to Figure 5 and Figure 6 for better understanding of the algorithm.

Step 1. Initialization. All buffers must be initialized to zero. A timestamp is obtained by reading the *millisecondsFromMid* counter and saving it in the *TimeStamp* variable. Therefore, *Time* vector index zero means the *TimeStamp* value. When the *Time* vector index value reaches its maximum, for example, 0xFFFF for 16 bits, the *TimeStamp* value is updated. Variable **K** is the index for samples while **J** is the index for interpolation. Variables *Up*, *Dwn*, and *Equal* are counters for the interpolation error sign. (Note that *Equal* is not used in this version.) **S** is the index for the MCT vectors. *N*, the maximum segment length in samples, and **Err**, the interpolation error, are inputs to the algorithm. *Signal* is a vector that holds at most *N* samples from the analog-to-digital converter. Signal vector is initialized with the first sample.

Step 2. Acquisition of two more samples. The linear interpolation needs three samples. Due to the limited CPU resources, the routine that implements the algorithm should be called each time a new sample is available.

Step 3. Interpolation. Interpolate the sample between the extreme samples, and compare it to the real sample. The magnitude of the interpolation error is checked against the maximum error **Err**, and also the error sign defines which *Up*, *Dwn*, or *Equal* counter increases. If the error is lower than **Err** and the segment has less than *N* samples, the sample index *I* is increased, and the algorithm goes back to step 2.

Step 4. Termination. If the error exceeds the **Err** value or the segment has reached *N* samples, the segment ends. If the segment has *N* samples, it implies that the segment never exceeded the **Err** limit, and thus it is classified as segment **a**, **b**, or **c**, based on the values of the extremes samples. If the error has reached the **Err** limit, based on the counters *Up*, *Dwn*, and the value of the extreme samples, the segment is classified as **d**, **e**, **f**, **g**, or **h**. In either case, the MCT vectors are updated. The last sample is the first sample for the next segment, and the algorithm goes back to step 2.

7. Transducer signal features extraction

7.1 Introduction

Figure 7 summarizes the algorithms based on MCT vectors. These vectors should be generated all the time with a circular structure and a time reference. There is a need to synchronize these vectors with those from other transducers. Millisecond from midnight (32 bit variable, *millisecondsFromMid*) is recommended as a candidate for the timestamp. The time reference is important to locate signal features and to relate actuator/sensor signal interference. The size of the buffer depends on the RAM available for this recommended practice. The basic algorithms that can be included in this recommended practice, as a first layer, are exponential, noise, impulsive noise, sinusoidal patterns, tendency, and mean estimation. They, in turn, are the basis for more complex algorithms, i.e., second layer, which can be optionally included in this recommended practice. The *user-defined application code* block is important in this recommended practice. End users or manufactures can include algorithms, based on MCT vectors and/or first layer algorithms, to be embedded into smart transducers. These can be shared with the community to provide an additional wealth to this recommended practice. In this manner, this recommended practice has a collaborative component adaptable to different types of signals. To be able to send the MCT vectors to the system would be a desirable feature. With this feature, algorithms can be tested outside of the transducer and then embedded in the transducer. It is important to note that the MCT vectors allow reconstructing the sensor signal within the interpolation error.

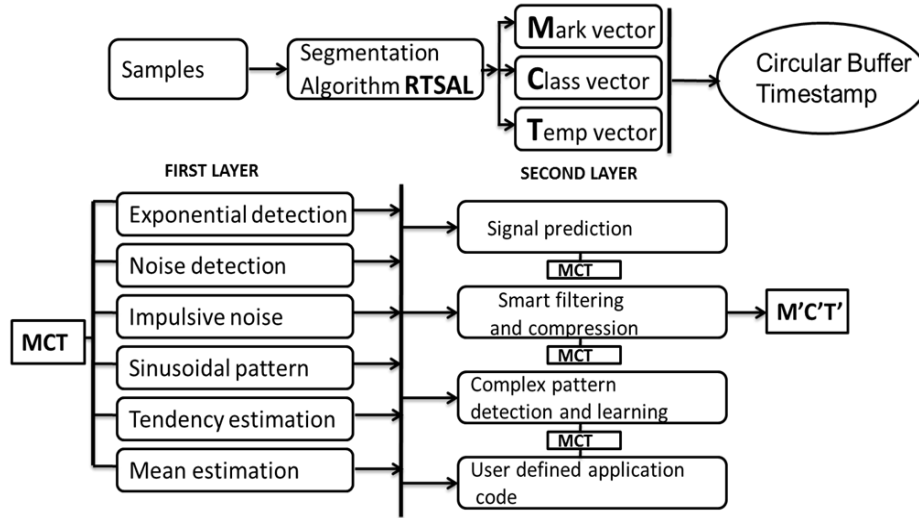


Figure 7—Algorithms based on MCT vectors

7.2 Initial parameters

The transducers need the following initial parameters, and by using the “initialization” function, it is possible to assign initial values:

Input parameters:

- Sampling period : [UInt16 x 0.1 ms], where 1 LSB means 100 μ s (*SamplingPeriod*)
- Interpolation error : [UInt8] (LSB) (*Err*)
- channelId : [UInt16] (*Channel*)
- Buffer size : [UInt32] (element) *BufferSize* Circular buffer size for MCT vectors.
- Maximum segment length : [UInt16] (samples) (*N*)

7.3 Variable definitions

7.3.1 Signal samples

[UInt16] (*SignalSamples*) Access to the raw data from the analog-to-digital converter. If the resolution is lower than 16 bits, the analog-to-digital results must be right justified. Note that if the analog-to-digital conversion is signed, it must be converted to unsigned.

Mark vector [UInt16]	(<i>M</i>)
Class vector [UInt8]	(<i>C</i>)
Time vector [UInt32]	(<i>T</i>)

7.3.2 Resources

Real-time interruption of one millisecond where the variable *millisecondsFromMid* is increased.

RAM memory for MCT, signal vectors, and variables.
Non-volatile memory to store IEEE 21451-001 algorithms.

7.3.3 MCT buffer structure

A double buffer structure with timestamp is shown in Figure 8. The buffer starts in buffer A with a timestamp. When this buffer is full, a new timestamp is created and related to the first position of buffer B. When this buffer is full, the process starts in buffer A, and so on.

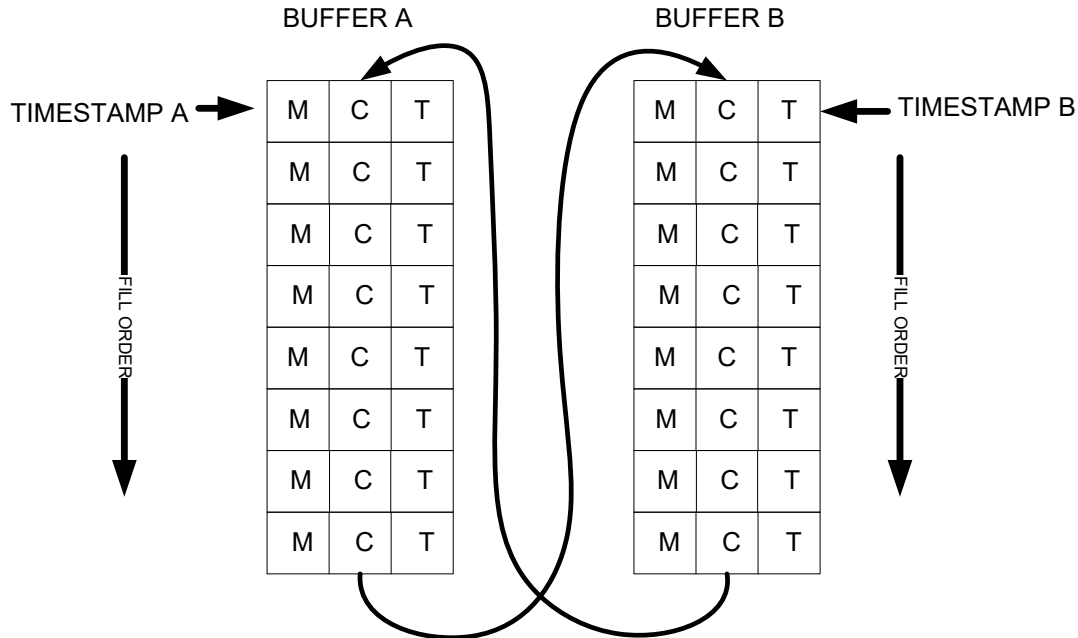


Figure 8—Buffer structure for MCT vectors

8. First layer algorithms

8.1 Introduction

Each time a new set of MCT vectors is obtained, the first layer algorithms are ready to execute. The algorithms that are enabled will run, and the results will be stored in a buffer with a timestamp. Each algorithm must have its own buffer.

8.2 First layer functions

These functions are considered essentials in this recommended practice because they are raw information for building knowledge about the signal behavior. The functions are exponential pattern detection, noise detection, impulsive noise detection, sinusoidal pattern detection, tendency estimation, and mean estimation.

8.3 Exponential pattern detection

The objective is to detect an exponential shape, with a fixed number of consecutive segments, using MCT vectors.

Input parameters:

- **UInt16Array** *mark* : Array that stores the amplitude
- **UInt8Array** *class* : Array that stores the class

- **UInt32Array** *time* : Array that stores time information
- **UInt8** *ExponentialAmount* : Number of consecutive segments required to start the algorithm
- **UInt16** *samplingPeriod* : [UInt16 x 0.1 ms], where 1 LSB means 100 μ s

Output parameters:

- **UInt8** *ExponentialType* : Type of detection (four types of exponential shape: d, e, f, g)
- **UInt16** (x 0.1 ms) *TimeSinceExponential* : Time since exponential detection
- **_Boolean** *ExponentialStable* : Predictable signal (yes/no)

The exponential detection function searches for consecutive equal values in the class vector. When it finds a fixed number (*ExponentialAmount*) of consecutive segment with the same class, it saves the starting point and continues until the vector has the same class in the following segments. It gives as output the class of the shape (*ExponentialType*), the time interval during which the classes are repeating (*TimeSinceExponential*), and an indication of whether the exponential waveform tends toward a constant value (*ExponentialStable*). Figure 9 and Figure 10 show examples of these functions. A flow chart of the algorithm is shown in Figure 11.

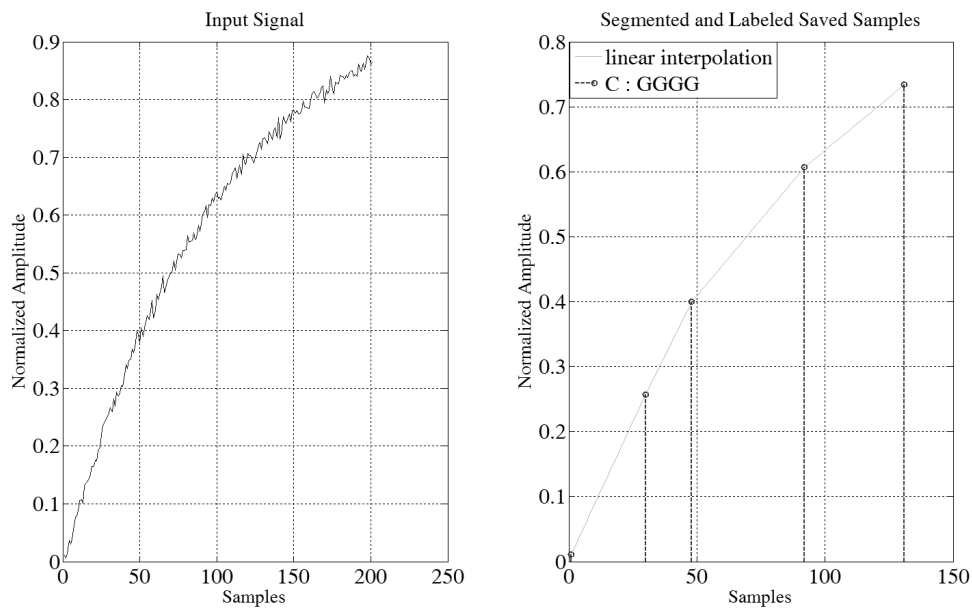


Figure 9—Exponential shape detected (Err = 0.03, ExponentialAmount = 3)

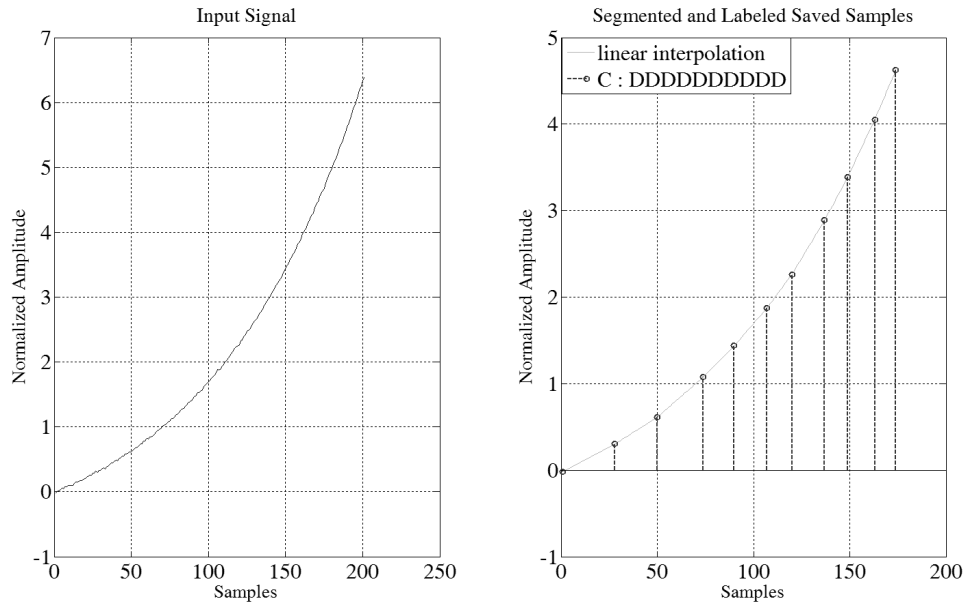


Figure 10—Exponential shape detected (Err = 0.03, ExponentialAmount = 3)

8.4 Noise detection

Noise is detected by computing the distance in samples among maximum and minimum. If a maximum occurs, the next one is a minimum, and vice versa. If the temporal difference between these two singularities is less than a predefined value for N times, a noisy flag is turned on.

Input parameters:

- **UInt16Array** *mark* : Array that stores the amplitude
- **UInt8Array** *class* : Array that stores the class
- **UInt32Array** *time* : Array that stores time information
- **UInt8** (x 0.1 ms) *NoisyPeriod* : Minimum amount of time between maximum-minimum or minimum-maximum to be considered as a noisy signal
- **UInt8** *AmountNoisy* : Number of consecutive detection of noisy signal
- **UInt8** *ThresholdPeriodicDetection* : Threshold for discriminating between casual or periodic noise
- **UInt16** *samplingPeriod* : [UInt16 x 0.1 ms], where 1 LSB means 100 μ s

Output parameters:

- **_Boolean** *Noisy* : Boolean to discriminate between a noisy signal and not (yes/no)
- **_Boolean** *Periodic* : Periodic noise (yes/no)
- **UInt8** (x 0.1 ms) *PeriodofNoise* : Period of noise

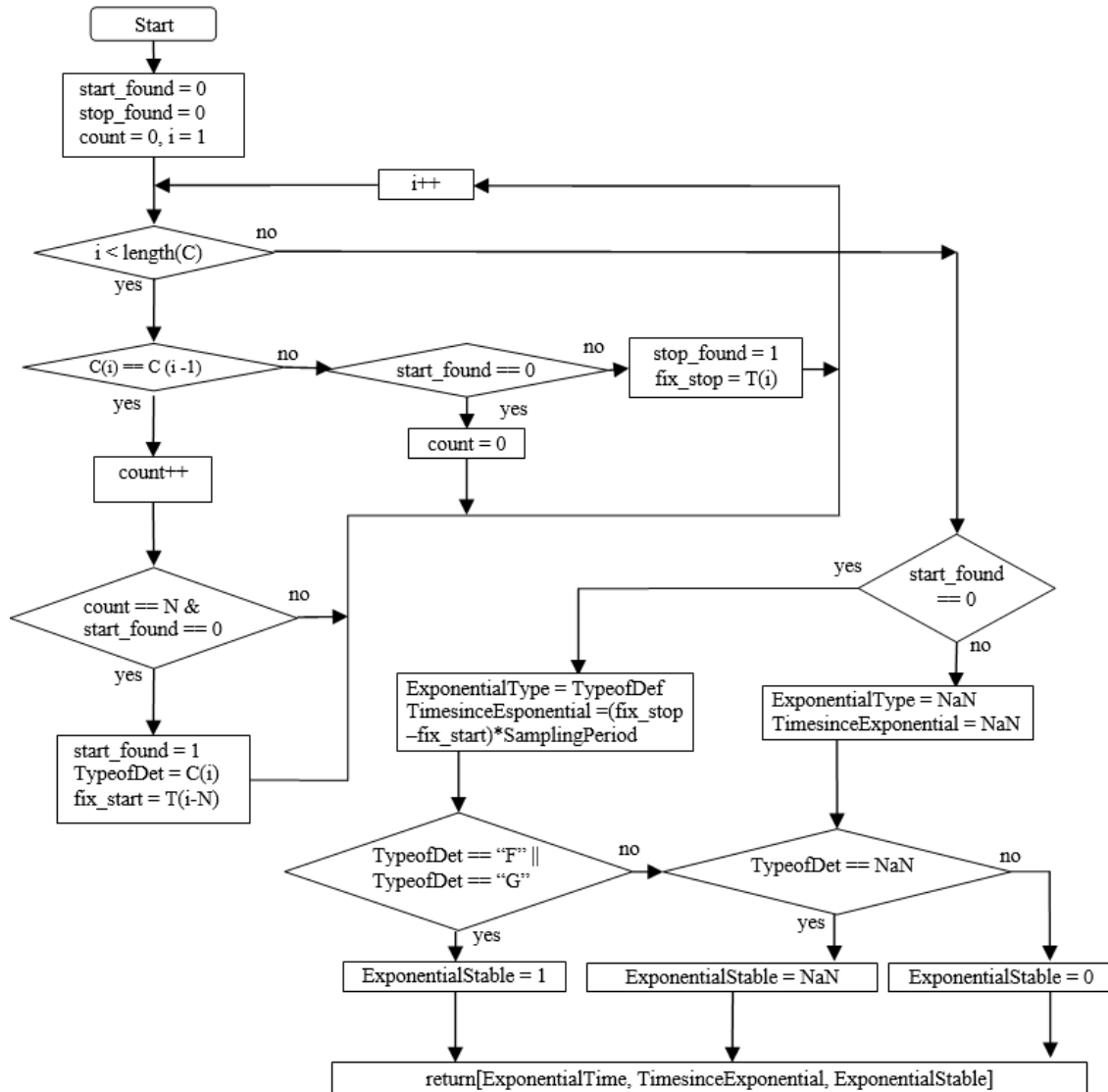


Figure 11—Flow chart of the exponential detection

The noise detection function searches for all the minimum and maximum segments in the whole class vector. It saves separately the type of peak and its position in the T vector. After that, it computes the time intervals between a series of a minimum and a maximum, or a maximum and a minimum instead, and it compares those values with the input parameter *NoisyPeriod*. If the number of times for which this calculated interval is smaller than the *NoisyPeriod* is greater than the *AmountNoisy*, the algorithm evaluates this signal as noisy; otherwise, it is not noisy.

The standard deviation s of the noise periods may be an indicator of random noise. When N consecutive periods are found lower than *NoisyPeriod* and if the ratio between standard deviation of the N values and their mean (σ/Np_{mean}) is higher than a threshold, the detected noise is random; otherwise, the detected noise is periodic.

If the signal is classified as “noisy,” the algorithm computes the mean and the standard deviation of periods observed in the temporal window; after that, it calculates the ratio σ/Np_{mean} 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 in output. Figure 12 and Figure 13 show examples of clean and noisy detection. A flow chart of the algorithm is shown in Figure 14.

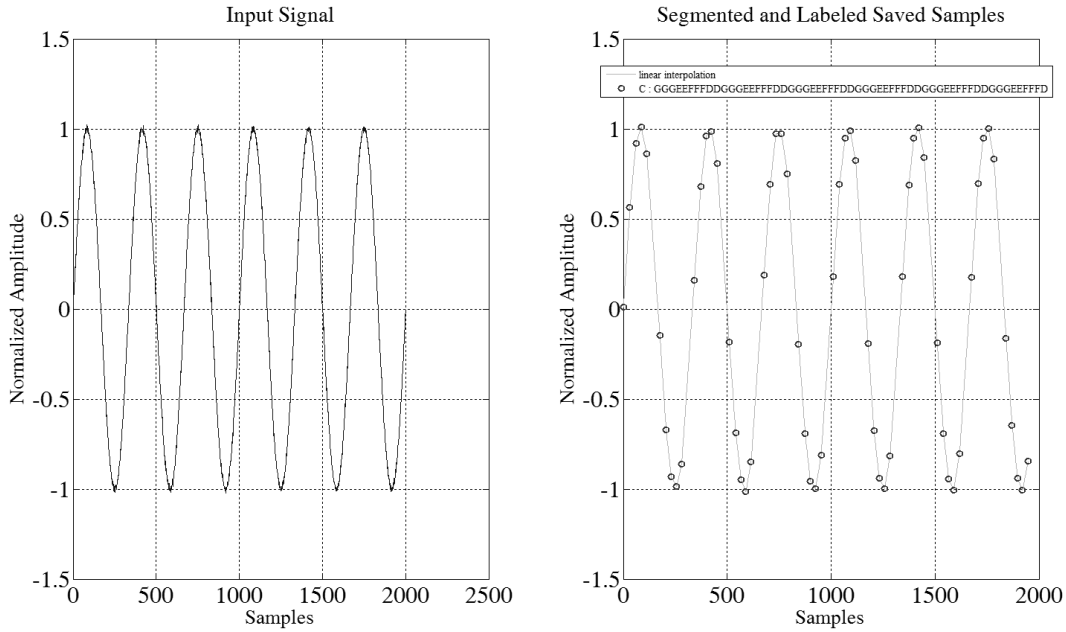


**Figure 12—Sinusoidal waveform not detected as “Noisy”
(Err = 0.1, signal frequency = 30 Hz, NoisyPeriod = 1ms,
AmountNoisy = 5, SNR = 40 dB)**

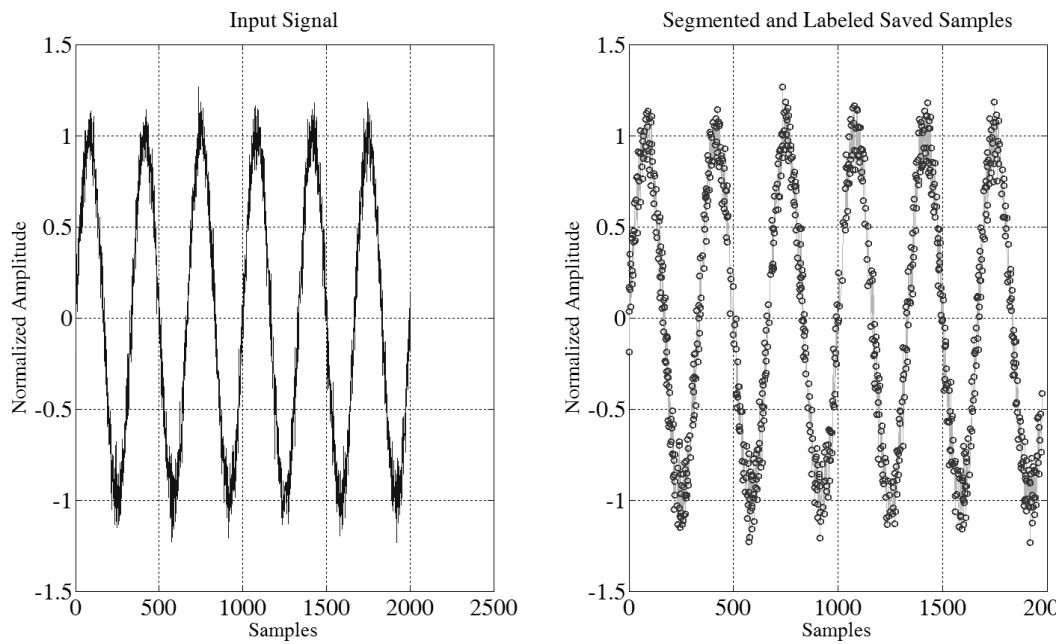


**Figure 13—Sinusoidal waveform detected as “Noisy”
(Err = 0.1, signal frequency = 30 Hz, NoisyPeriod = 1ms,
AmountNoisy = 5, SNR = 20 dB)**

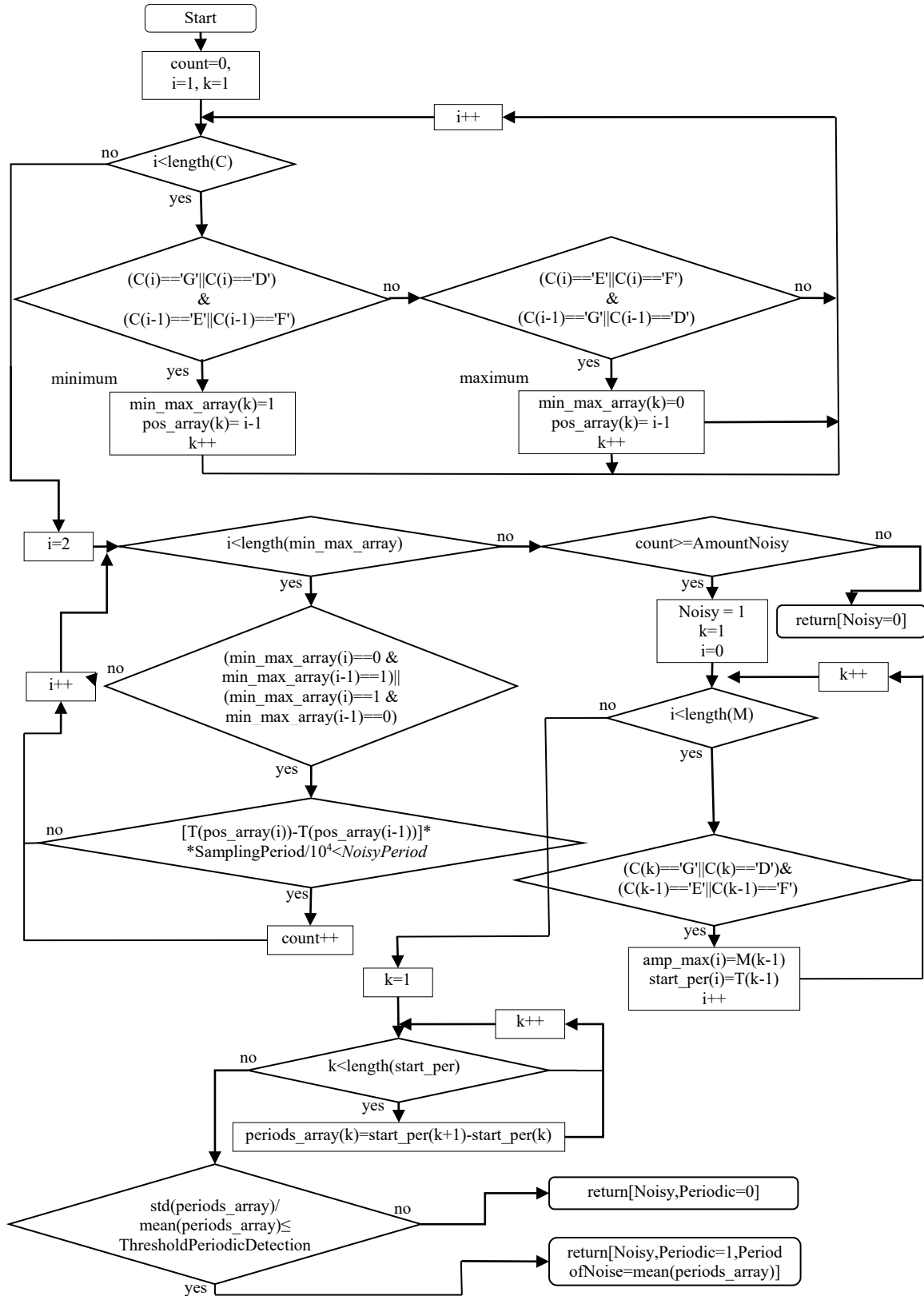


Figure 14—Flowchart of noise detection

8.5 Impulsive noise detection

Impulsive noise is defined as interference on a signal caused by a short period of noise. To be considered as impulsive noise, the change in amplitude must be significant. Once the algorithm has started, it looks for the first maximum or minimum. Then, it computes the amplitude change from the value before the length reduction plus a percentage value of the full-scale value of the sensor. The length of the impulsive noise is computed from the temporal difference between the first segment and the last one. For example, the sequence “**dddfff**” indicates the occurrence of a maximum, and the impulse length is the difference between the temporal position of the last “**f**” and the first “**d**”.

Input parameters:

- **UInt16Array** *mark* : Array that stores the amplitude
- **UInt8Array** *class* : Array that stores the class
- **UInt32Array** *time* : Array that stores time information
- **UInt8** (in percentage [1 to 100]) *ImpulseThreshold* : Minimum amplitude change to be considered as an impulsive noise
- **UInt16** *FullScale* : Full scale of sensor

Output parameters:

- **_Boolean** *ImpulseDetection* : Detection of impulsive noise (yes/no)
- **UInt8** *ImpulseType* : Type of impulse (“**df**”, “**eg**”, “**fd**”...)
- **UInt16** *ImpulseAmplitude* : Amplitude of impulse
- **UInt16** *ImpulseLength* : Length of impulse in samples
- **UInt16** *ImpulsePosition* : Position of impulse
- **UInt16** *ImpulseArea* : Area of impulse
- **UInt16** *ImpulseShape* : Shape of impulse
- **UInt16** *ImpulseAsimmetry* : Asimmetry of impulse
- **UInt16** *ImpulseAsimmetryType* : Type of asimmetry

Figure 15 and Figure 16 show examples of impulsive noise detected in sinusoidal and exponential waves.

The impulsive noise detection function browses the entire class vector looking for the maximum or minimum. When it finds one of them, it compares the value of the maximum/minimum with a threshold value calculated starting from the segment before and adding or subtracting the given percentage value of the full-scale sensor from the previous value (adding or subtracting depends on whether it is analyzing a maximum or a minimum). It returns a set of outputs:

- *ImpulseDetection* is a boolean value that communicates the detection.
- *ImpulseType* communicates the classes that form the impulse.
- *ImpulseAmplitude* is the value of the impulsive peak.
- *ImpulsePosition* is the sample number where the peak is found (time indication).
- *ImpulsiveLength* is the number of samples between the segment before the peak and the segment after the peak.
- *ImpulsiveArea* is an indication of the impulse energy.
- *ImpulsiveShape* is the slope.
- *ImpulsiveAsimmetry* is an indicator for the trend of the base of the impulse.
- *ImpulsiveAsimmetryType* indicates if this trend is increasing, decreasing, or constant.