Using Algorithms on Smart Transducer: An IEEE Standard Perspective

Zheng Liu, *Senior Member, IEEE,* Fahd Banakhr *Member, IEEE,* G. Monte *Member, IEEE,* Victor Huang *Senior Member, IEEE*

***Abstract*—This paper describes the role of algorithm development in the IEEE standard 21451.001. The purpose is to promote and enhance the computational capabilities of smart transducers, and facilitate the flow from sensor raw data processing to sensor knowledge fusion. A case study on real-time segmentation and labelling is conducted to demonstrate the feasibility of the first layer algorithm. The key role of the IEEE 21451.001 standard is to standardize the data structure for user defined application code and pattern learning algorithms and regulate the protocols for data sharing and exchange. The considerations for the standard development are illustrated and discussed. The foreseeable impact of the standard for algorithms is highlighted as well.**

***Index Terms*—Smart transducer, algorithm, feature extraction, sensor knowledge fusion.**

I. Introduction

SMART transducers have been widely adopted and used in various applications, including health care, civil infrastructure, environment monitoring, manufacturing etc. A smart transducer is an integration of sensor or actuator element, processing unit, and a communication interface [1]. For the sensors, the raw data or signal will be digitalized and transmitted to a central unit for further processing. Usually, applications may require large number of sensors to perform a broad range measuring and monitoring tasks. All the transducers/sensors at diverse locations need to be networked to transmit data and information. The communication interfaces of smart transducers to networks are defined by IEEE P1451 standards, which help to achieve the interchangeability and interoperability of transducers' connection to networks [2]–[5]. Moreover, the transducer electronic data sheet (TEDS), which contains the necessary information to identify and characterize a transducer and its outputs, is standardized as well [6]. The TEDS can be deployed in two ways, i.e. through a EEPROM resided in a transducer/sensor or a separate file called virtual TEDS. These standards help the manufacturers to rapidly distribute network-independent solutions for different industrial applications.

The use of sensor data starts from raw data pre-processing and feature extraction. More information can be further derived from the extracted features and relevant knowledge is generated based on the obtained information. Thus, the data processing and analytic capabilities become essential for a smart transducer. The networked sensors in distributed measurement and control systems generate a huge amount of data to transmit and process. This may introduce intensive load to both communication and computational operations. How to balance and manage such loads depends on application requirements and available resources and also remains as a topic for research. One of the strategies for big sensor data analytics is to move data processing closer to the sensor.

Varied applications employ different transducers/sensors and thus may require very different algorithms for sensor data processing [7]. Moreover, the inputs and outputs from different sensors may have different data structures. Even for the same type of sensor, the data structure may vary with vendors. To run certain data processing algorithm on a transducer/sensor, it is necessary to assess the inputs and outputs, and control the parameter settings. A standardized data interface can facilitate the data flow and information share. Thus, there is the need to define an universal framework for the practice of sensor data processing. With this standard, the end user should be able to tell and identify:

- Algorithms being used by the transducer/sensor;
- Algorithm control information (version, implementation, etc.);
- Parameter settings (inputs);
- Sensor outputs and settings (data structure, measurement unit, etc.);

Those settings are also known as user defined application code. IEEE 21451.001 is such a standard to regulate the algorithms running on a transducer/sensor. At the first layer, a real-time segmentation and labeling (RTSAL) algorithm is implemented, which generates three output vectors, i.e. ***m****ark* vector, ***c****lass* vector and ***t****empos* vector (known as MCT vectors) [8].

This paper discusses the role of IEEE 21451.001 standard for running algorithms on smart sensor and the potential use of the MCT vectors for further data processing and analysis. In a case study, the MCT vectors are compared with original signal in terms of a number of parameters for signal measurement. The feasibility of the RTSAL algorithm is understood and it is possible to implement feature extraction and representation based on the MCT vectors.

II. The Role of IEEE 21451.001 Standards

A. Algorithms for Smart Transducer

The algorithms in transducer/sensor facilitate flow of raw sensory data and features to the information and knowledge. The analytic capability makes sensor more competent to

Dr. Z. Liu is with Toyota Technological Institute, Nagoya, 468-8511, Japan e-mail: zheng.liu@ieee.org
Dr. F. is with Loughborough University, UK
Dr. G. Monte is with Universidad Tecnologica Nacional, Argentina
Dr. V. Huang is with the Better Wrold-IES Standards, USA
Manuscript received XXXX 00, 2014; revised XXXXX 00, 2014.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSEN.2014.2360987, IEEE Sensors Journal

generate information from the deluge of data. However, a smart transducer faces the resource constrains, such as:

- Insufficient computational capability;
- Limited storage;
- Short of battery life;
- Inadequate communication ability.

The implementation of data analytic algorithms in smart transducer needs to take these challenges into account. Meanwhile, the algorithms may vary with the sensors and its applications. The connection of sensors to the network is revolutionizing the practice of using sensors in varied applications. The internet of things (IoT) enables the cloud-based data collection, analysis and communication between myriad sensors [9]–[13]. The sensory data can travel through the wired or wireless network to be stored in the cloud. The cloud computing infrastructure offers high-performance analytic tools to process sensor data. Therefore, there are three possible scenarios for implementing algorithms in a smart sensor as illustrated in Fig. 1. The first scenario is that the sensor just transfers the raw data to the cloud without any local processing, e.g. sensor A in Fig. 1. In the second scenario, all the data are processed by the sensor and results are sent to the cloud for direct use, such as sensor D in Fig. 1. The data can also be partly pre-processed at the sensor side and further processed in the cloud, such as senor B and C. This is the third scenario. As shown in Table I, there should be a balance between the local computational load for the sensor and the traffic load for the network. Again, the balance point may vary from sensor to sensor.

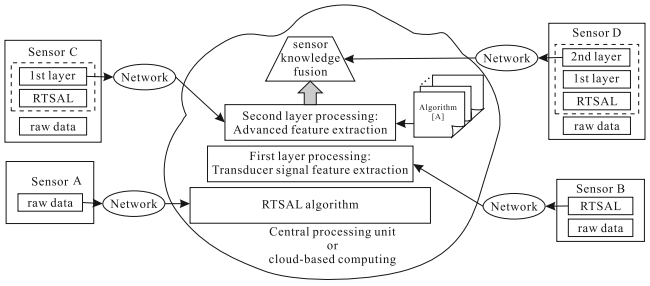


Fig. 1. Illustration of analytic algorithms for smart transducer/sensor.

Thus, it is possible to build a smart transducer/sensor consisting of two parts: one is the physical sensor and the other is the sensor data processing element located at a central processing unit or in the sensor computing cloud. In this case, the algorithm will not be limited by the inherent constraints of the sensor's hardware. A four bit micro-controller never be able to process the video streams acquired in a surveillance application. However, an intelligent camera with high-performance processor is competent to take this assignment. With high-performance computers in the cloud, sophisticated algorithms can be configured and implemented to make sense of the deluge of sensor data. The algorithms may also reside on both sides, i.e. sensor and cloud. The collaborative local and central computing scheme may take advantages of both computational modes as well as the communication facility provided by the network.

The algorithms evolve with the emerging of new sensing and data processing technologies. There will never be a complete list for the algorithms used by sensors. To get more information, multiple algorithms may be applied sequentially or in parallel to obtain multiple features from the raw data. With these in mind, the IEEE 24151.001 standard should be able to manage the situation and all the algorithms employed in sensor applications.

B. The Need of Standard for Algorithms

The algorithms can actually be implemented either locally on a transducer or remotely in a computing could. The algorithm to process the data from temperature sensor will be different from the one used for accelerometers [14]. Thus, the algorithms largely depend on the specific sensor and application. Different features may be required from the same sensor in different applications. As illustrated in Fig. 2, both the sensor and the algorithms are determined by the application's requirements, i.e. the needed information. Thus, a smart transducer should be reconfigurable. An algorithm dictionary or database is needed. When the end user looks up for a piece of information, the dictionary or database will recommend the algorithms. Or with a particular sensor at hand, corresponding algorithms associated with this sensor will be presented. However, there is no need to implement all the available algorithms in one transducer/sensor. Depending on the platform, algorithms may be implemented in different forms or versions. Again, this depends on the specific application.

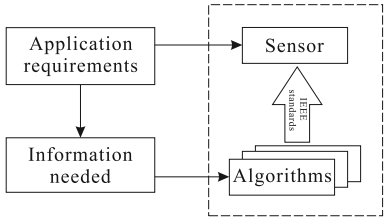


Fig. 2. The choices of sensors and algorithms are application dependent.

Motivated by TEDS, a transducer algorithm data sheet (TADS) should be created to identify a specific algorithm. The basic information in TADS is listed in Table II. First, the algorithm is identified by its ID. However, the implementation of the algorithm may vary with the platforms and application needs. Thus, a version control number should be assigned. Meanwhile, a sequential number is needed to identify how the algorithms are organized and arranged to work together. When there are multiple algorithms, their sequence can be learnt from this number, i.e. which is the first processing and which is the next. Other important parameters include the inputs and outputs of the algorithm. The parameter ID helps to control the input parameters for algorithm tuning and precision setting while the feature ID will tell which feature is extracted for output. The detailed data structure corresponding to a specific algorithm should be defined and available to the end user by searching with the TADS. TADS may serve as a complementary piece of information to TEDS, but it is committed to sensor data and signal processing. Similarly,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSEN.2014.2360987, IEEE Sensors Journal

TABLE I
THE ALGORITHM IMPLEMENTATION IN SMART SENSOR.

Algorithm implementation	Description	Sensor computational load	Network traffic load
Scenario 1	There is no data processing at sensor side. All the data will be sent to cloud for processing.	Low	High
Scenario 2	Data will be processed by the sensor and results will be used directly or sent to cloud.	High	Low
Scenario 3	Data pre-processing is done by sensor and post-processing is carried out in cloud.	Moderate	Moderate

TABLE II
THE INFORMATION NEEDED BY THE TRANSDUCER ALGORITHM DATA SHEET.

Field	Description
Algorithm ID number	identify a specific algorithm
Algorithm version number	control different implementations and versions
Algorithm sequential number	identify the sequence of the algorithm
Parameter ID number	identify the input parameters to the algorithm
Feature ID number	identify the output features of the algorithm

we need to consider the storage capacity to accommodate the amount of information for a specific algorithm.

Table III lists some typical sensor data and signal processing algorithms, which are available in commercial software tools like Matlab™ and LabVIEW™. One task of the standard is to figure out how to migrate these algorithms or implementations to smart transducers through properly defining the interface and corresponding protocols.

TABLE III
TYPICAL SENSOR DATA/SIGNAL PROCESSING ALGORITHMS.

Number	Algorithm Description
1	Moving average filtering
2	Filtering with a designated filter
3	Detrending data by subtracting the mean or a best-fit line
4	Descriptive statistics: maximum value, mean value, median value, minimum value, standard deviation, variance, most frequent value
5	Signal normalization
6	Sensor data modeling: linear correlation with time, regression against time, residuals and goodness of fit (R^2)
7	Sensor data resampling (downsampling, upsampling)
8	Power spectral density (PSD) estimation
9	Cepstrum analysis
10	Discrete cosine transform (DCT)
11	Walsh-Hadamard transfor (WHT)
12	Fast Fourier transform
13	Hilbert transform
14	Wavelet transform
15	Auto-correlation
16	Sensor signal measurements: root mean square value of periodic waveforms, slew rate of triangular waveform, duty cycle of rectangular pulse waveform.
17	Empirical model decomposition

III. SIGNAL TREATMENT FOR SMART TRANSDUCERS

A. RTSAL Algorithm and MCT Vectors

The IEEE 21451.001 working group is making effort to standardize the methodologies for the transducer's signal treatment, conditioning, and data conversion [8]. Figure 3 illustrates the basic structure of the algorithms in IEEE 21451.001, which consists of three parts. The basis is the real-time segmentation and labeling (RTSAL) algorithm, which derives a set of feature vectors from the raw data. The set of vectors include *mark*, *class*, and *tempos* and are presented with an abbreviation MCT [8].

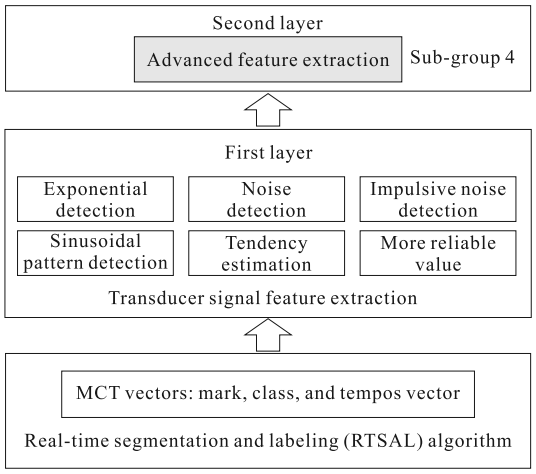


Fig. 3. The basic structure of IEEE 21451.001 and the role of algorithms.

The first two vectors, i.e. *mark* and *class*, are generated from a segmentation process. The interpolation is applied to the first and third of three consecutive samples. If the difference between the interpolated result and the second sample is less than a preset threshold value, the second sample is discarded as it can be duplicated through such interpolation. The retained samples constitute the vector mark while the corresponding timestamp information is put in the temporal index vector *tempos*. To further describe the sensor signal, a third vector, *class*, is added in the labeling process. Totally, eight classes of segments are defined with letter "a" to "h" as illustrated in Table IV. The trend of the signal can be characterized with the combination of the eight basic segments. The RTSAL algorithm generates the MCT vector sets.

As illustrated in Fig. 4, the RTSAL algorithm starts from selecting the samples from sensor signal S for the *mark* vector. There are three major parameters for the RTSAL algorithm, i.e. interpolation error E, incremental error δ, and iteration number $iter$. The first value of the signal $S(i)$ $(i = 1)$ is assigned to the first element of $mark(j)(j = 1)$. Then, given

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSEN.2014.2360987, IEEE Sensors Journal

TABLE IV
CLASSES OF SEGMENTS: A-H [15].

Class	a	b	c	d
Segment				
Class	e	f	g	h
Segment				

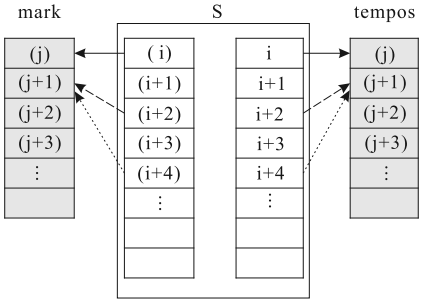


Fig. 4. Illustration of sample selection for vector $mark$ and $time$ from sensor signal S.

$k = 2$, the difference is calculated as:

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

If d is larger than predefined interpolation error E (variable $erro$ in the program), then do the following assignments:

$$\begin{cases} mark(j+1) = S(i+2) \\ tempos(j+1) = i+2 \end{cases} \quad (2)$$

If d is smaller than $erro$, then the step is enlarged by 2 through operation $k \leftarrow k + 2$. Referring to equation (1), d will be re-calculated with updated k. Again, if the difference is smaller, the sample $S(i+k)$ and its time stamp $i+k$ is retained in the $mark$ and $tempos$ vectors respectively as illustrated in Fig. 4. This process will go through all the samples in S. Once it is done, we obtain the initial $mark$ and $tempos$ vectors. The sensor signal S will be updated through linear interpolation using vector $mark$ and $tempos$. For example, given $i = j = 1$, let $mark(1) = S(1)$ and $mark(2) = S(5)$. So, the first two elements of the $tempos$ vectors are $t_1 = 1$ and $t_2 = 5$, respectively. Then, signal $S(r)$ will be updated by:

$$S(t_1 + r) = S(t_1) + m \times r, \ r = 1, \cdots, t_2 - t_1 - 1 \quad (3)$$

where there is $m = {}^{mark(2)-mark(1)}/_{t_2-t_1}$. With the updated S, the process enters the next iteration to select new $mark$ and $tempos$ vectors, which are the final outputs of the RTSAL when the iteration ends.

Figure 5 gives the detailed flowchart of the implementation. The operation READ is to load the signal. Operation CLEAR is to remove old variable *mark* and *tempos* as they will be assigned with new values in the next loop. Finally, the OUTPUT operation returns the resulted vectors. It should be mentioned that Fig. 5 does not present the labelling processing, where the obtained labels can be used to learn the pattern of the signal.

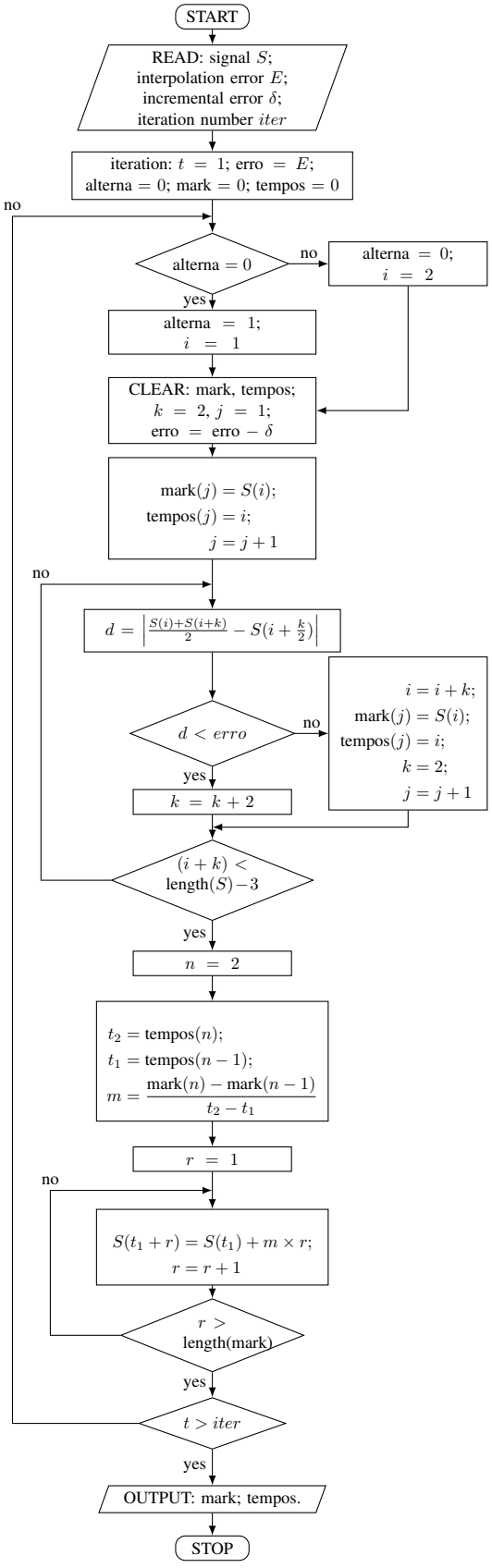


Fig. 5. The flowchart of the RTSAL algorithm (Labeling is not presented in this flowchart).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSEN.2014.2360987, IEEE Sensors Journal

Incremental error δ is the reduce the value of interpolation error E in each loop. In the following experiment, δ is set as zero.

Further sensor signal feature extraction can be carried out on the MCT vectors. At the first layer, six function modules, which include exponential detection, noise detection, impulsive noise detection, sinusoidal pattern detection. tendency estimation, and more reliable value, are implemented. These function modules consist of the fundamental operations of the sensor data.

Referring to the three scenarios in Fig. 1, the basic RTSAL, first, and second layer algorithms can be implemented on a transducer, in a central computing unit or computing cloud, or both of the two. For example, the RTSAL algorithm can be embedded in a transducer/sensor while the first and second layer processing is done in the central computing unit or cloud. Subject to the available resources on the sensor, the first layer can be implemented on the sensor while higher level computing at the second layer is carried out remotely in cloud computing.

B. Further Algorithms Development with MCT Vectors

The RTSAL algorithm is easy to implemented on micro-controllers, which actually also achieves the sensor data compression. The testing on micro-controllers is under way and the performance will be reported. Further feature extraction or pattern learning can be built with the derived MCT vectors for the applications of signal detection, classification, or quantification. The basic operations may include those items listed in Table III. For example, the learnt regression model against time can be used to predict the sensor reading at a designated time. The deviation to the predicted value can provide a piece of evidence for anomaly occurrence. The balance point here is the efficiency and accuracy. The use of MCT vectors is more efficient, but the accuracy needs to be taken into account. In other words, the features extracted from MCT vectors should work for the application as the features extracted from the original signal directly. This is again an application dependent issue. In next section of experiments, a case study was carried out to see the differences between the results obtained from original signal and MCT vectors in a general sense.

IV. Experimental Results

In the case study, a segment of ECG (electrocardiogram) signal from MIT-BIH arrhythmia database was used in the experiments [16], [17]. An exercise on processing the ECG signals is presented in [18]. However, this study did not consider any specific applications originating from the data, such as heart failure detection, diagnosis of neurologic disease etc. We only considered the general features of the signal. The RTSAL algorithm targets the embedded implementation on the micro-controller or processor integrated with a sensor or transducer. Thus, the signal measurement functions from Matlab™ were employed to provide an observation of the changes before and after applying the RTSAL algorithm. More specifically, the following six measurements were employed (N is the total number of data sample X_n, where $n = 1 \cdots N$):

- Band power: the average power of the input signal $X_{BP} = \frac{1}{N}\sum_{n=1}^{N}|X_n|^2$;
- Maximum-to-minimum difference: the difference between the maximum and minimum values of the input signal $X_{PP} = X_{max} - X_{min}$;
- Root-mean-square (RMS) level: the root mean square value of the input signal $X_{RMS} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}|X_n|^2}$;
- Root-sum-of-squares level: the root sum of squares $X_{RSS} = \sqrt{\sum_{n=1}^{N}|X_n|^2}$;
- Peak-magnitude-to-RMS ratio: ration of the largest absolute value of the signal to its RMS value $X_{PRMSR} = \frac{\max(|X_n|)}{X_{RMS}}$.

In addition, we also considered the signal variations against time, i.e. pulse period, pulse width, pulse fall and rise time, slew rate, and duty cycle.

The ECG signal segment is given in Fig. 6, which is a periodical signal with four pulses and three cycles. The original signal is added with random noises and given at the bottom of Fig. 6. This case study investigates the impact of added noises on RTSAL algorithm. Both of the two signals were processed with RTSAL algorithm and compared with the MCT vectors in terms of the criteria listed above.

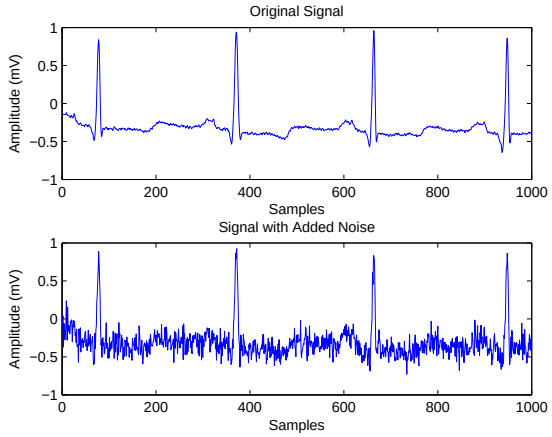


Fig. 6. The ECG signals used in this case study: (top) original signal and (bottom) signal with added noise.

The original signal has 1000 samples. Table V gives the number of samples in the MCT mark vector obtained by different settings of the algorithm, e.g. the interpolation error and iteration number.

When the error tolerance increases, the total sample number reduces. For the original signal, a higher compression rate was achieved while the signal with added noises did not change a lot due to the change of the error settings. This is because the RTSAL algorithm tries to keep more information introduced by the noises. By keeping the same interpolation error (0.01) and varying the iteration number from two to four, the samples in the mark vector of original signal reduced significantly (around 50%) from iteration one to iteration two. Again, the *mark* vector of signal with noise shrunk, but not as significant as the original signal. While continuing the increase of the iteration number, the number of samples reduced gradually. Figure 7 and 8 plot the signals and corresponding outputs of

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSEN.2014.2360987, IEEE Sensors Journal

the RTSAL algorithm. With the increase of the error tolerance or the iteration number, the signal will lost its details and just kept the approximation of it. Such approximation can still show the change and trend of the signal against the time.

TABLE V
THE NUMBER OF SAMPLES OF ORIGINAL SIGNAL AND MCT MARK VECTOR OBTAINED BY RTSAL ALGORITHM WITH DIFFERENT SETTINGS (UNIT: SAMPLE).

Signal (with 1000 samples)	Error (mV) (iteration=1)			Iteration (error=0.01)		
	0.01	0.02	0.03	2	3	4
Without noises	217	86	62	98	72	57
With added noises	468	434	404	419	300	212

The changes of the signal are presented with the numbers in Table VI, which characterizes the signal with the measurements. The increase of error tolerance enlarges original signal's band power, but does not affect the noised signal a lot. This is because more details (noises) are kept in the signal. The root-mean square (RMS) level has a similar trend as band power while the peak-magnitude-to-RMS ratio decreases. The root-sum-of-squares (RSS) level reduces the value due to the root square operation applied to the item $\sum_{n=1}^{N} |X_n|^2$.

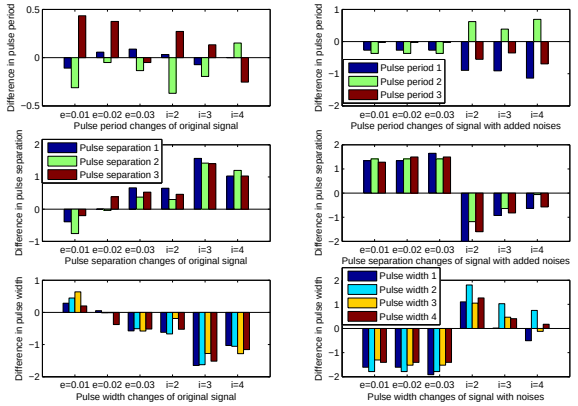
The changes to the four pulses and three cycles can be observed from Fig. 9. The left seven groups are from the original signal while the rest seven from the signal with added noises. The interpolation error and iteration number have impact on these signal characteristics. The general trend is that the error increases with the increase of the interpolation error tolerance. However, the increase of the iteration number does not always generate the same trend for all the features. For example, the errors in pulse separation and width get smaller when more iterations are applied, but on the contrary the errors in fall and rise time increase.

Thus, when the RTSAL algorithm is applied to sensor data, it is necessary to figure out how the algorithm settings affect the specific features of the signal, which are critical for further data analysis and signal characterization.

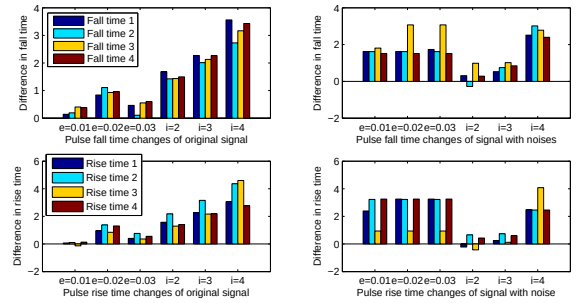
V. THE FUTURE PERSPECTIVE

The RTSAL algorithm is simple and easy to implement and thus suitable for smart transducer. Other downsampling techniques and recently developed compressive sensing are not considered due to the complexity and load for microcontroller based computation [19], [20]. The local processing is constrained by the computational capability of the microcontroller residing on the sensor. IEEE 21451.001 standard will regulate the protocols of the algorithm, which decide the data structure and flow, algorithm setting and running.

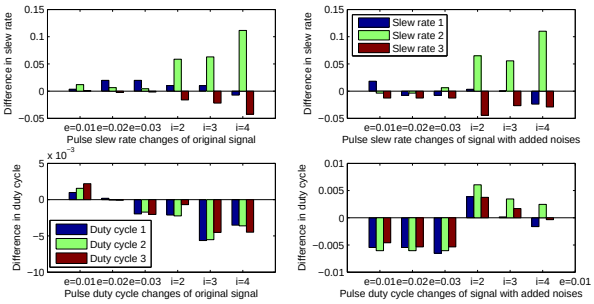
Figure 10 illustrates how to use an algorithm on smart transducer. The application selects the sensor and corresponding algorithms. Depending on the application, the first layer RTSAL algorithm can be applied. And the sensor specified algorithms can be implemented at the second layer based on MCT vectors. The algorithms do not have to be in the IEEE 21451.001 standard, but the algorithms need to be regulated by IEEE 21451.001 standards so that the data can flow smoothly from lower to higher level. This is the key role of IEEE 21451.001 standard.



(a) Changes of pulse period, separation, and width (top to bottom).



(b) Changes of pulse fall time and rise time (top to bottom)



(c) Changes of slew rate and duty cycle (top t to bottom)

Fig. 9. The changes with interpolation error and iteration number: (left) original signal and (right) signal with added noises.

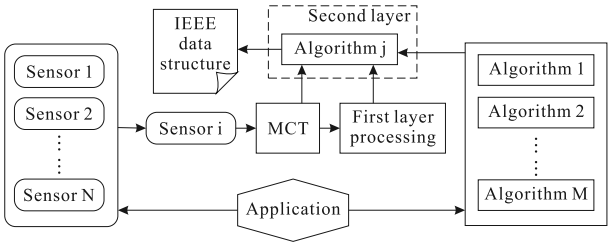


Fig. 10. The use of algorithm on smart transducer.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JSEN.2014.2360987, IEEE Sensors Journal

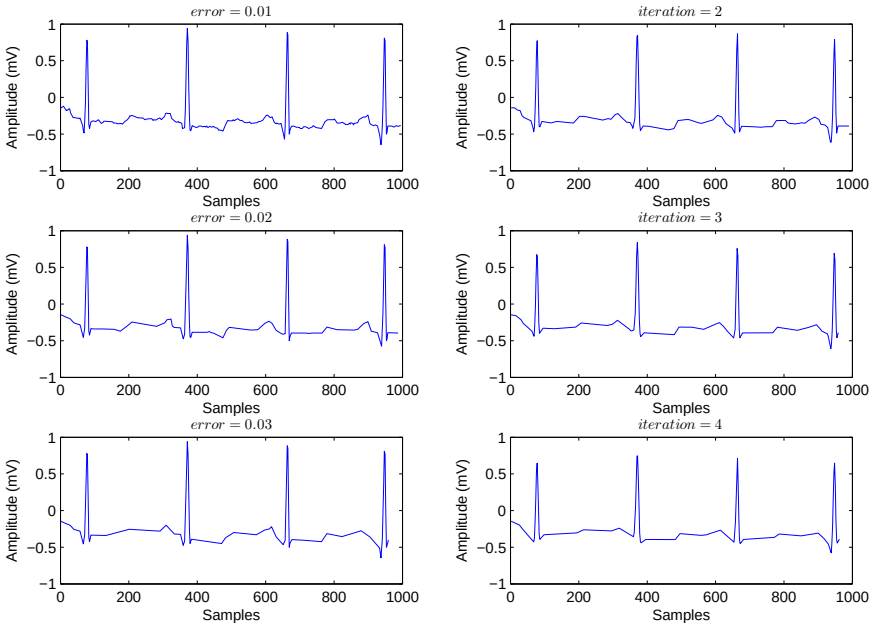


Fig. 7. The original ECG signal and the corresponding MCT samples obtained by different settings of error and iteration number.

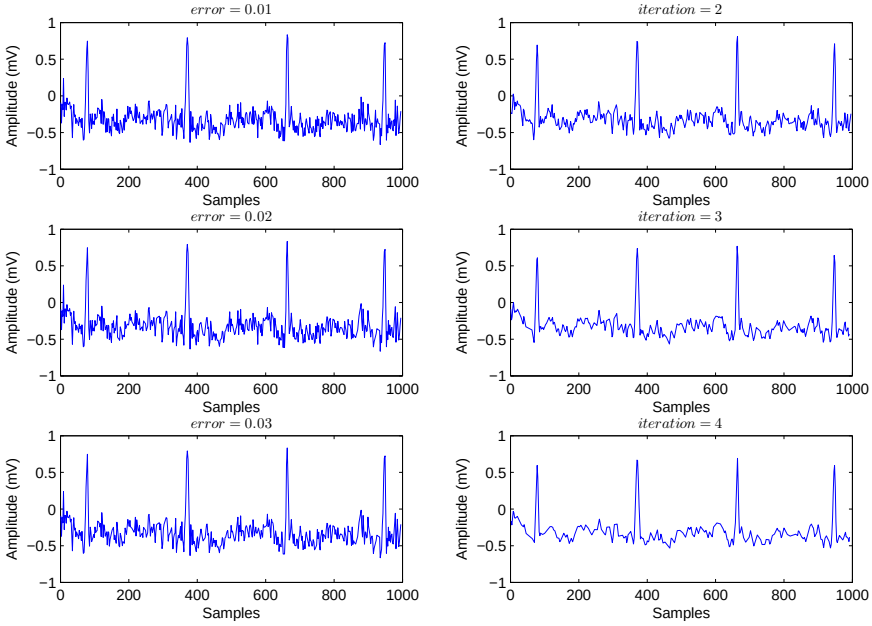


Fig. 8. The ECG signal with added noises and corresponding MCT samples obtained by different settings of error and iteration number.

TABLE VI
THE MEASUREMENTS OF THE ORIGINAL SIGNAL AND SIGNAL WITH ADDED NOISES.

Measurement of original signal							
Signal characterization	Original signal	Error (mV)			Iteration		
		0.01	0.02	0.03	2	3	4
Band power	0.1309	0.1442	0.1714	0.1993	0.1596	0.1567	0.1556
Maximum-to-minimum difference	1.6050	1.5850	1.5150	1.5850	1.4788	1.4494	1.3241
Root-mean-square level	0.3617	0.3797	0.4140	0.4464	0.3996	0.3959	0.3944
Root-sum-of-squares level	11.4393	5.5938	3.8395	3.5151	3.9554	3.3592	2.9778
Peak-magnitude-to-RMS ratio	2.6538	2.4754	2.2704	2.1056	2.1712	2.1187	1.8920
Measurement of signal with added noises							
Signal characterization	Signal with noises	Error (mV)			Iteration		
		0.01	0.02	0.03	2	3	4
Band power	0.1416	0.1400	0.1390	0.1404	0.1349	0.1328	0.1327
Maximum-to-minimum difference	1.6524	1.4977	1.4977	1.4977	1.4079	1.3333	1.2244
Root-mean-square level	0.3763	0.3742	0.3728	0.3747	0.3673	0.3644	0.3642
Root-sum-of-squares level	11.9002	8.0951	7.7666	7.5310	7.5188	6.3118	5.3032
Peak-magnitude-to-RMS ratio	2.4559	2.2288	2.2371	2.2259	2.2057	2.1039	1.9004

VI. CONCLUDING REMARKS

With the deployment of large amount of sensors to all kinds of applications, a huge amount of data are collected and accumulated for processing. One strategy for big sensor data analytics is to move data processing closer to the sensor. IEEE 21451.001 standard is to offer such an option to the smart transducer. The data treatment services on smart transducer can provide extracted features rather than raw data for further analysis and processing. The current draft proposes a real-time segmentation and labelling (RTSAL) algorithm as a basis or first layer of the standards. The RSTAL has advantages for easy implementation and capability to run with limited resources. Algorithm testing on micro-controller is under way and detailed performance report on computational efficiency is not available yet. This paper presents a case study on ECG signal processing. Currently, the local computational capability is limited by the hardware resources. However, the network of sensors makes it possible to implement more sophisticated algorithms on cloud based computing. This paper aims to highlight the key role of IEEE 21451.001 standard in the project of smart transducers and sensors. Its scope is to regulate the protocols of data structure and flow as well as the functionalities for algorithm setting and running. The standard will facilitate the data sharing and data abstraction from raw data to knowledge.

REFERENCES

[1] W. Elmenreich and S. Pitzek, "Smart transducers – principles, communications, and configuration," in *Proceedings of the 7th IEEE International Conference on Intelligent Engineering Systems*, vol. 2, Assuit – Luxor, Egypt, Mar. 2003, pp. 510–515.

[2] "IEEE 1451 smart transducer interface standard," http://ieee1451.nist.gov.

[3] K. B. Lee and R. D. Schneeman, "Distributed measurement and control based on the IEEE 14151 smart transducer interface standards," *IEEE Transactions on Instrumentation and Measurement*, vol. 49, no. 3, pp. 621–627, June 2000.

[4] K. Lee. (1999) A synopsis of the IEEE P1451- standards for smart transducer communication. http://ieee1451.nist.gov/1451synosis-599F.pdf. National Insitute of Standards and Technology.

[5] J. Mark and P. Hufnagel, "The IEEE 1451.4 standard for smart transducers," https://standards.ieee.org/develop/regauth/tut/1451d4.pdf, July 2014, retrieved.

[6] National Instruments. An overview of IEEE 1451.4 transducer electronic data sheets (TEDS). http://www.ni.com/white-paper/3468/en/.

[7] M. K. Banavar, B. Chakraborty, H. Kwon, Y. Li, J. J. Zhang, C. Chakrabarti, A. Papandreou-Suppappola, A. Spanias, and C. Tepedelenlioglu, "A review on sensor, signal, and information processing algorithms," Sensor, Signal and Information Processing Center, Arizona State University, Tempa, AZ, USA, Technical Report 0704-0188, 2010.

[8] G. Monte, F. Abate, V. Paciello, A. Pietrosanto, Z. Liu, and V. Huang, "Normalizing transducer signals: An overview of a proposed standard," in *Proceedings of 2014 IEEE International Instrumentation and Measurement Technology Conference*, Montevideo, Uruguay, May 12-15 2014.

[9] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787 – 2805, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128610001568

[10] S. Karlin, "Crossing disciplines to advance the internet of things," *IEEE The Institute*, p. 14, March 2014.

[11] A. Alamri, W. S. Ansari, M. M. Hassan, M. S. Hossain, A. Alelaiwi, and M. A. Hossain, "A survey on sensor-cloud: Architecture, applications, and approaches," *International Journal of Distributed Sensor Networks*, vol. 2013, pp. 1–18, 2013.

[12] W.-Y. Chung, P.-S. Yu, and C.-J. Huang, "Cloud computing system based on wireless sensor network," in *Computer Science and Information Systems (FedCSIS), 2013 Federated Conference on*, Sept 2013, pp. 877–880.

[13] I. Klimchynski, "Extensible embedded web server architecture for internet-based data acquisition and control," *Sensors Journal, IEEE*, vol. 6, no. 3, pp. 804–811, June 2006.

[14] M. A. P. Pertijs, A. Aita, K. A. A. Makinwa, and J. Huijsing, "Low-cost calibration techniques for smart temperature sensors," *Sensors Journal, IEEE*, vol. 10, no. 6, pp. 1098–1105, June 2010.

[15] G. Monte, "IEEE P1451.001 draft recommended practice for signal treatment applied to smart transducers," Unapproved IEEE Standards Draft, November 2012, subject to change.

[16] G. B. Moody and R. G. Mark, "The impact of the MIT-BIH arrhythmia database," *IEEE Engineering in Medicine and Biology Magazine*, pp. 45–50, May/June 2001.

[17] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, pp. 215–220, June 2000.

[18] G. Sibushri, "Analysis of ECG signals for arrhythmia using matlab," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 2, pp. 1093–1099, March 2014, special Issue.

[19] E. J. Candes and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, pp. 21–30, March 2008.

[20] M. Davenport, P. Boufounos, M. Wakin, and R. Baraniuk, "Signal processing with compressive measurements," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 4, no. 2, pp. 445–460, April 2010.