Layered architecture approach of the sensor software component stack for the Internet of Things applications

BRAGARENCO ANDREI
Department of Microelectronics and Biomedical Engineering
Technical University of Moldova
168, Stefan cel Mare Blvd., MD-2004, Chisinau
REPUBLIC OF MOLDOVA
andrei.bragarenco@mib.utm.md

MARUSIC GALINA,
Department of Computer Science and Systems Engineering
Technical University of Moldova
168, Stefan cel Mare Blvd., MD-2004, Chisinau
REPUBLIC OF MOLDOVA
galina.marusic@adm.utm.md

CIUFUDEAN CALIN
Department of Computers, Automatics and Electronics
University Stefan cel Mare of Suceava
ROMANIA
ciufudean.calin@gmail.com

Abstract: - The paper stands for a layered architecture approach of the sensor software component stack for the Internet of Things applications. A well-defined architecture is one of the key success factors for a project, as it improves the maintainability, reusability, and other things related to the efficiency in software development. The concept is inspired by the AUTomotive Open System ARchitecture (AUTOSAR) approach of the Software component Development with the proposal to extend the architecture with the Sensor and Actuator component stacks as parts of so named, here, Extended Software (ESW). The first part presents a generic architecture for an IoT device, following with a generic software component stack proposal, applied for any component from the proposed architecture, going through the description of all layers from the Service down to Hardware abstraction, with an implementation proposal, also covering the HW/SW association. In the end, an architecture example for environmental data acquisition is presented.

Key-Words: - AUTOSAR, architecture, device, electronic, environment, IoT, layer, network, component stack.

1 Introduction

1.1 IoT Challenge

We are living in the modern era, where time and money are getting more critical in all fields of activities. This also includes the software development domain, in particular, automotive field, because the OEMs want to add more functionality in existing platforms or new platforms that includes these functionalities, but realized with the same financial and temporal constraints. More functionality implies a growing complexity of the developed software, a large number of the variants for the same car platform through the same or separated Electronic Control Units (ECUs) [1].

A recent forecast made by IDC projects the Internet of Things (IoT) and the associated ecosystem to be a $1.7 trillion market by 2020, which will include 212 billion connected things. The IoT will fuel a paradigm shift of a “truly connected” world in which everyday objects become inter-connected and smart with the ability to communicate many different types of information with one another as well as with human users Fig.1, presents a graphical forecast of IoT explosion over the coming years as estimated by Cisco [2].

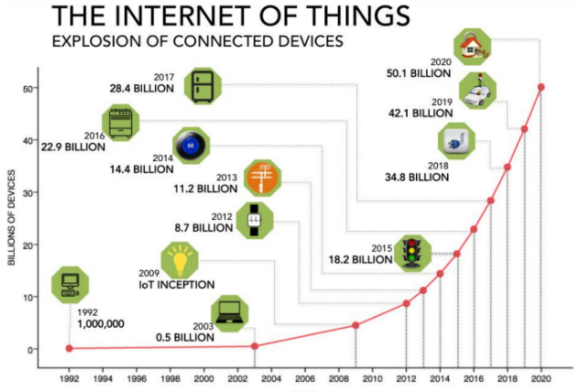

Fig.1. The explosion of the Internet of Things [2]

Exponentially growing Internet of things domain also requires a well defined architectural approach to control de system complexity it involves.

1.2 AUTOSAR Layered architecture overview

A very well example of a well-defined architecture is the AUTomotive Open System ARchitecture (AUTOSAR) concept that is the basis for the development of the new modules in automotive systems and stands for the newest worldwide automotive trend. AUTOSAR standard defines the reference architecture and method for the development of automotive software systems and supplies the language (metamodel) for their architectural models. It also specifies the architectural modules and functionality of the middleware layer known as the Basic Software (BSW) [3].

The concept is abstracting the electrical equipment through a particular layer called Runtime Environment (RTE) so that the application (ASW) does not consider the type of equipment or the location of the required resources. The responsibility for the generic functionalities and connectivity is transferred to the BSW, Fig.3.

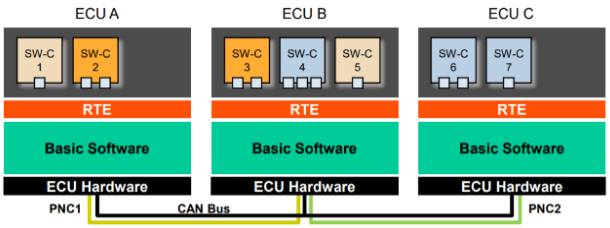

Fig.2. AUTOSAR component interaction concept [3]

Moreover, this concept allows the application to access resources located on other equipment as if it were part of the equipment it is running on, which is due to an even higher abstraction where all RTE layers form a concept of Virtual Function Bus (VFB). This concept allows the realization of distributed applications where the components can be distributed on different electrical equipment units (ECU) Fig.3.

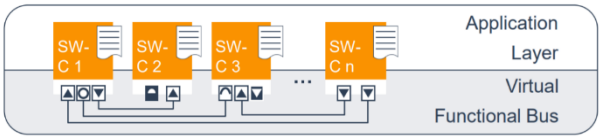

Fig.3. AUTOSAR Virtual Functional Bus concept [3]

Generic functionalities such as Memory, Communication, or Security are performed in BSW inside of the Layered component stacks, distributed between three generic layers such as Service Layer, ECU Abstraction Layer (ECAL), and MCU Abstraction Layer (MCAL). The functionalities that cannot be generalized are defined as Complex Device Driver Fig.4.

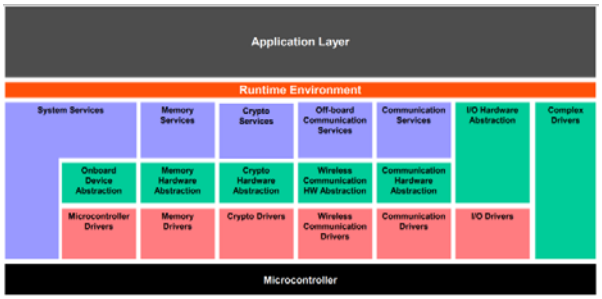

Fig.4. AUTOSAR Layered architecture [3]

The Sensor/Actuator is a specific type of AUTOSAR Software Component for sensor evaluation and actuator control. Though not belonging to the AUTOSAR Basic Software, it is described here due to its strong relationship with local signals. It has been decided to locate the Sensor/Actuator SW Components above the RTE for integration reasons. Fig 5 [3].

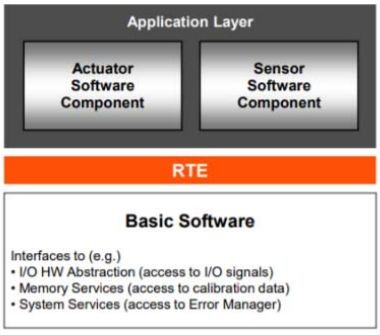

Fig.5. Sensor/Actuator component in the AUTOSAR architecture [3]

Inspired by the AUTOSAR concept, it is proposed that the components of the Sensors and Actuators that in the AUTOSAR architecture are part of the Application Layer to be defined with layered component stacks in a group so-called Extended Software (ESW) as an extension of the BSW. This supplies an abstraction for most of the generic architecture components proposed in this work.

2 Problem Formulation

Before going into the concept of the Extended Software (ESW) component stack proposal, we will analyze the Internet of Things (IoT) system as a spread-out device and the architectural approach for the generic IoT Device on which the IoT consists.

2.1 Spread Out System Concept

On the highest abstraction level, the system could be viewed as a device that takes information from the environment, processes it, and reacts based on internal and environmental factors. Within this paper, two aspects will present the system, the structural and the functional.

Structurally, the system stands for a collection of interconnected components, Fig 6. For an IoT device such components could list a sensor, an actuator, user interaction, communication, database, and power management or device-specific application components, as presented in Fig.9.

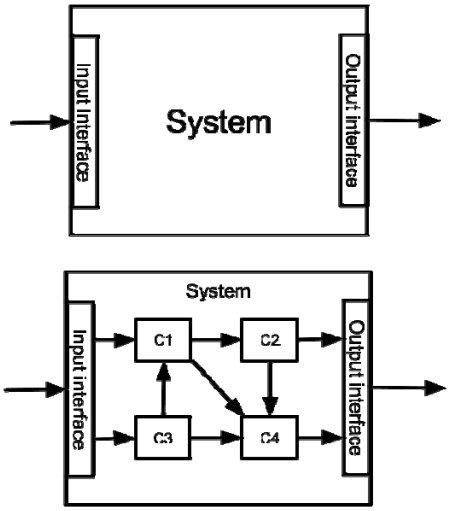

Fig.6. A system as a collection of components that compose it

Functionally, the system represents the set of transfer functions through which a signal from the environment interface is propagating through various transfer functions till reaching the output of the system, providing signals back to the environment, as shown in Fig.7.

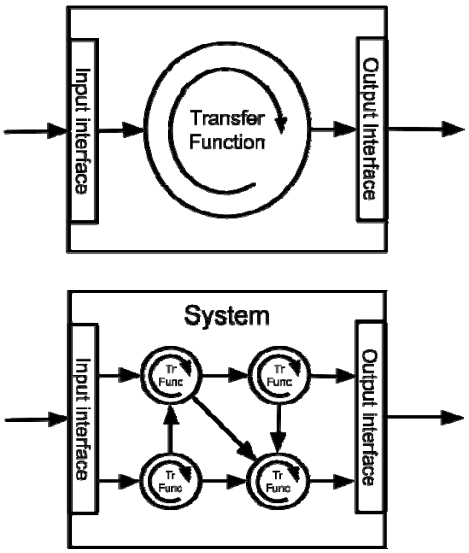

Fig.7. A system as a collection of transfer functions that compose it

A device network could be considered as an electronic spread out system, where the network replaces or abstracts the wired connections for signal interconnection, Fig.8.

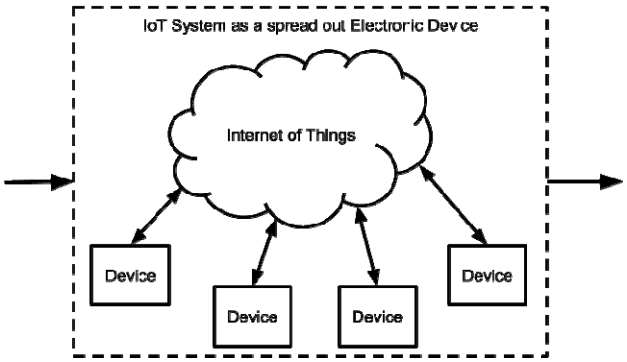

Fig.8. Spread-out electronic devices with IoT interconnections

The abstraction is applied through all domains, including either mechanical engineering (ME), electrical engineering (EE), software engineering (SWE), or of any other kind, considering the system as a set of components and with their transfer function.

2.2 The generic architecture of an IoT device

Functionalities of the system could be clustered in some specific generic components, as usual parts of a standard device. These components contain functionalities according to the problem it solves as follows: Acquisition, Actuation, User Interaction,

Communication, Database, Power Management, Basic Software or Operating System components, Fig.9.

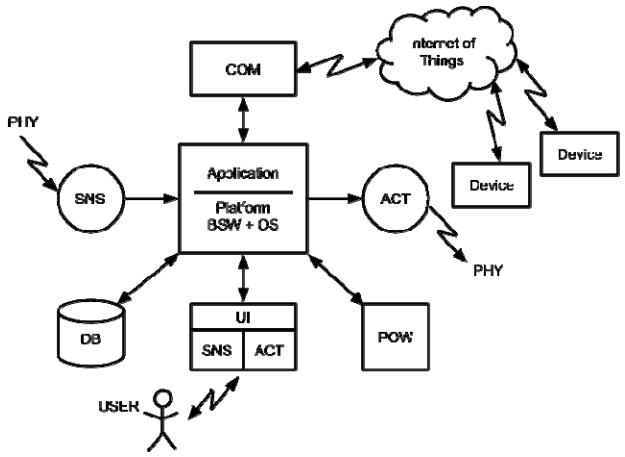

Fig.9. The generic architecture of an IoT Device

Two aspects of the device should be mentioned – the interaction parts, which are the environment and the communication network between the user and the devices, and those that are part of the device structure as Sensors, Actuators, User Interaction, Database, Power Management, Communication.

Interactions

- ***PHY*** - the environment represented by its physical parameters to be acquired or influenced by the device.
- ***USER*** - the user, either human or inhuman, that can intervene with the introduction or tracking of messages or actions from the device.
- ***IoT*** - Internet of Things stands for all media and communication technologies between Devices.

Device Components

- ***SNS*** - Sensor stands for the set of components implemented by Mechanical Engineering (ME), Electrical Engineering (EE), or SW Engineering (SWE) involved in the acquisition and transformation of a physical parameter of the environment into internal information of the system. PHY to INFO.
- ***ACT*** - Actuator is the set of components implemented by Mechanical Engineering (ME), Electrical Engineering (EE), or SW Engineering (SWE) involved in transforming the control information into action to the physical parameter of the environment, INFO to PHY.
- ***UI*** - User Interaction represents the set of all SNS or ACT components specialized for human or inhuman user interactions.
- ***COM*** - Communication stands for the set of components made by Mechanical Engineering (ME), Electrical Engineering (EE), or SW Engineering (SWE) that intervene to ensure the transfer of information between interconnected devices.
- ***DB*** - Database represents the set of components for storage and access to data stored for the short or long term within the Device, or access to remote storage media. It could also be identified as memory (MEM).
- ***POW*** - Power Management represents all the components that ensure the normal functioning of the device from the electrical point of view, as well as the management of energy consumption.
- ***BSW*** – is the platform that consists of the Operating System (OS) and services available to the Application running on the platform, ensuring a high level of abstraction for the Device.

3 Problem solution

Following the system architecture of a generic device for an IoT, the next step is to define the architecture for every component it consists. In this paper, we will assume that we will reuse the concept proposed by AUTOSAR, so we will consider that some of the components of IoT Device could be covered by AUTOSAR BSW, such as OS, COM, MEM/DB, IO, CDD.

In this paper, we propose a generic concept for the so-called component class Extended Software (ESW) for an AUTOSAR-like architecture with HW/SW association., focusing on a solution for the Sensor (SNS) component stack.

3.1 Sensor software component stack

Every component of the system consists of parts implemented by Mechanical Engineering (ME), Electrical Engineering (EE), or SW Engineering (SWE). These are organized according to a layered architecture to ensure interactions between all domains, to facilitate the safe and qualitative transfer of the signal/information to the application, and vice versa, from the application to the environment, user, or other systems, Fig.10.

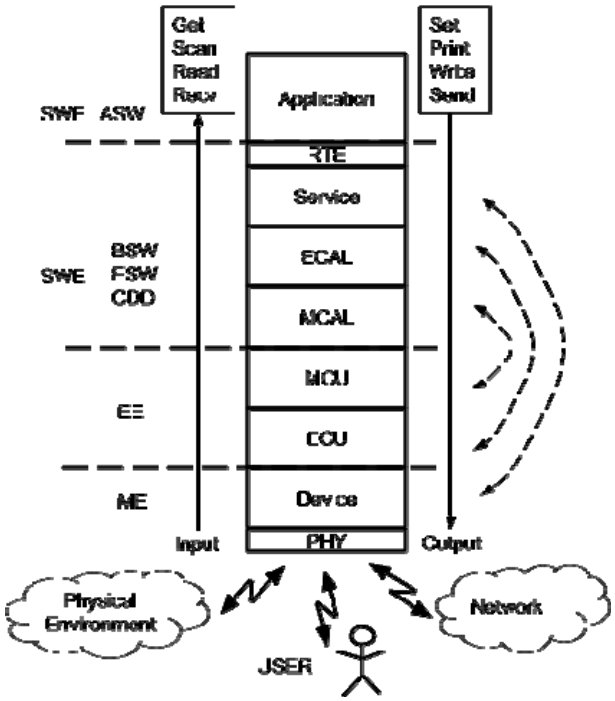

Fig.10. Layered architecture for a generic component

According to the presented concept, in the Software domain, there are defined many layers that abstracts and supply with services related to the associated layers from the Electrical and Mechanical domain. Those services also are known as Device Drivers – libraries to support peripheral equipment. In this way, according to the diagram from Fig.10, we will obtain the following associations:

- ***MCU vs. MCAL***: where the MCU stands for all the internal hardware peripherals of the Microcontroller such as GPIO, TIMER, ADC, USART, SPI, EEPROM, WDT, etc. The MCAL (MCU Abstraction Layer) represents the software services to support the microcontroller's peripherals. Among the resources of this layer can provide functionalities such as *DigitalRead, DigitalWrite, PortRead, PortWrite, AnalogRead, AnalogWrite, SerialRead, SerialWrite*, which are specific for the information transfer, but also other specific to peripheries features.
- ***ECU vs. ECAL***: where the ECU level represents the electronic components located on the microcontroller board or attached to it. The ECAL being the layer of drivers for the electronic solutions on the ECU. Those electronic components being the electrical parts of the components from the architectural concept of a device presented in this paper in Fig.9, such as Sensors, Actuators, external memory chips, user interaction devices, communication modules, Power Management mechanisms.
- ***Device vs. Service***: where the Device stands for the physical device itself, to which all the characteristics according to the specification are associated. The Service represents the upper-level SW of the component. It is the producer of services provided to the Application and ensures the functionalities of managing the component, as well as the interaction with the other services from other components under RTE.
- ***PHY - RTE***: probably not the best association, but as PHY represents the physical environment through which the device interacts with the environment, RTE is the one by which the Application interacts with the environment providing functions prefixed by *Get, Scan, Read, Recv* to bring information from the environment and respectively *Set, Print, Write, Send* to transfer information to the environment. Respectively we can assume that RTE is the abstraction to the PHY, the environment.
- ***Application***: does not have an associated level and represents the system itself. At the application level, the behavior of the system is implemented. The application can be implemented locally on a single device or distributed on several devices. Similarly to that, multiple applications could run on the same device the device to be part of multiple systems. Fig.3.

3.2 Sensor Overview

In this paper, we are focusing more on the sensor component stack definition, Actuators being the proposal for further works.

A sensor represents the totality of components realized through Mechanical Engineering (ME), Electrical Engineering (EE), or SW Engineering (SWE) that participates in the acquisition and transformation of a physical parameter of the environment into internal information of the system, PHY to INFO.

There are a vast number of different sensors, applying different measurement techniques, and using different interfaces to a controller. This, unfortunately, makes sensors a difficult subject to cover [4].

To find the proper sensor according to the

application and define the services it provides, it makes sense to classify them according to certain criteria, such as:

- ***By the physical nature of the sensor and the acquired signal.*** It is essential to clarify first what kind of environment in which the measurements are acquired, which is the physical parameter measured, and which is the sensor construction and what acquisition methods it uses. With this information, it is easier to identify which is the most suitable sensor according to its technical specifications.
- ***By the signal type at the sensor output.*** This classification allows the identification of a suitable electrical interface and may impose restrictions on the resources of Microcontroller or other electronic components of the Device.
- ***By localization of the sensor.*** From the Device's point of view, it is necessary to distinguish the localization of the sensor. Is it a local or a global one.
- ***By the signal source.*** It is essential to distinguish between proprioceptive/internal sensors and exteroceptive/ external sensors.
- ***By the interaction with the environment***, passive and active sensors could be classified. The active is the sensor that performs a stimulus to the environment to extract the value of interest.

3.2.1 Signal Acquisition

The original information about the environment has a physical state or physical effect, depending on its nature. For example, the luminosity is defined by a photon flux, the humidity - concentration of water molecules in the air, the temperature – by infrared irradiation, sound – by mechanical oscillations, the distance is expressed in metric units, etc.. Considering the non-electric nature of the effect and the fact that the contemporary equipment operates with electrical signals, a conversion of the non-electric quantities into electric ones is necessary. So, this conversion is the primary function of the sensor.

At the component level - the sensor senses the physical changes occurring in the surrounding and converting it into a readable quantity. This value often could not be used directly, so additional conditioning is required, a function that is in the responsibility of another component - the translator. The transducer converts the physical quantity or nonelectrical into another signal or electrical signal. As a conclusion, to obtain a physical size expressed in the electrical signal, two components are needed - a sensor connected to a transducer. Usually, in the context, both names are used to refer to the whole pair, Fig.11.

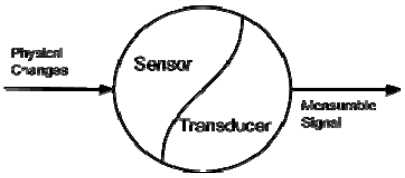

Fig.11. Generic sensor concept

3.2.2 Signal Conditioning

The value directly collected from the sensual element is usually a raw one that requires conditioning before being used as a real physical value within the system. Thus, the signal from the moment of acquisition by the sensor goes a long way through numerous transfer functions. Those functions could be performed on both sides, in the electronic domain, same as in the Software domain. There are using a sensor with a digital interface that already may have the conditioning mechanism that may be implemented in the transducer. However, in the classical case, or in the design of the transducer itself, the conditioning must be designed considering all aspects of the acquisition method and the environment through which the signal propagates.

A typical path that a signal would follow for an analog signal source is to go through the electronic domain with preliminary conditioning following conditioning in the Software domain, Fig.12.

Signal acquisition - stands for the transfer function that converts the physical parameter into an electrical signal. This transfer function can be found in the datasheet of the sensor in case it is an industrially produced one, or the experimental curve can be extracted by setting environmental conditions and measuring the electrical parameter at the sensor output.

Electrical amplification/attenuation and offset - the obtained electrical signal needs amplification in order to be able to see the change of the physical value applied to the sensor on the electrical output of the sensor, as well as brought to the acceptable value range by the interface of the microcontroller by establishing an offset, which usually is in the range of 0-5 Volts. The amplification, as well as the offset setting on the electrical side, can be performed with the methodology of functions with operational amplifiers. Attenuation can be obtained with passive circuits such as a voltage divider, or by the same method, with operational amplifiers with a 1/x amplification coefficient.

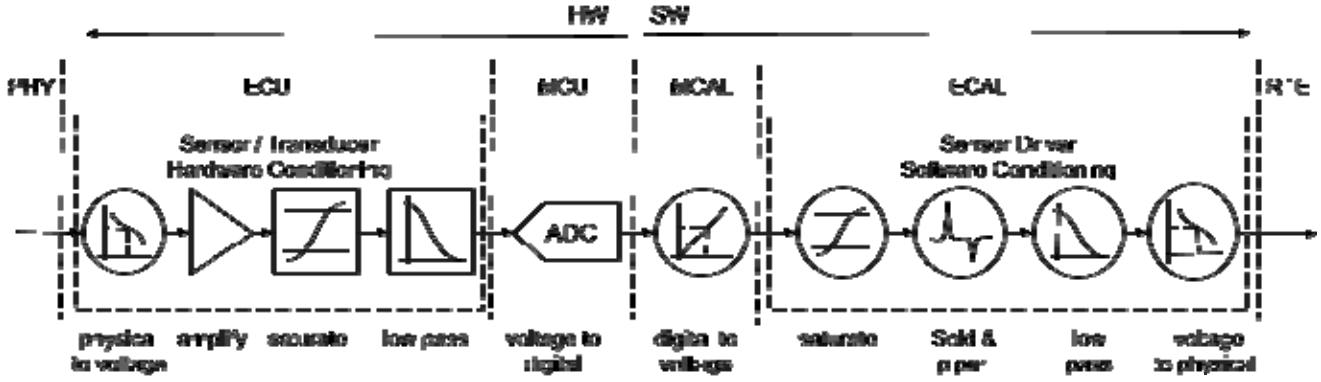

Fig.12. Typical signal conditioning flow

Electrical saturation - it is essential not to admit that the voltage levels outside the acceptable range of values to be applied to the microcontroller, which could damage this circuit. Assuming that in the amplification and offset phase are brought to the acceptable range for microcontroller acquisition, the values outside this domain are out of interest. The realization of this mechanism is possible both with simple circuits type stabilizing diodes, as well as by the methodology of performing the functions with operational amplifiers.

Electrical filtering – it is recommended to have a preliminary filtration in the electrical conditioning in order to exclude electromagnetic influences and signal acquisition disturbances. Usually, a LowPass filter could be involved in filtering the high frequencies, specific to the white noise, but a HighPass or bandpass filter also could be applied.

The HighPass filter could exclude the variation of the reference point, the offset of the signal. A particular case is the application of the Notch filter, which excludes the frequencies induced by the household voltage sources 50-60 Hz. The electric filters can be made either with passive elements, with active elements such as operational amplifiers, or specialized circuits.

Analog-Digital Conversion - assumes that the analog signal applied to a pin of the microcontroller is converted into a digital value that can be retrieved with software engineering techniques, meaning, reading the data from a register in the address space of the peripherals. Analog-Digital conversion limits the resolution of the definition of the signal value as well as the maximum sampling frequency to that specific to the converter. The characteristics of an A/D converter include Accuracy expressed in the number of digits it produces by value (for example 10bit A/D converter), Speed expressed in maximum conversions per second (for example 500 conversions per second), Measurement range expressed in volts (for example 0-5V) [4]. The converter can be one from inside of the microcontroller, allowing the direct collection of the analog signal, external one with the serial or parallel interface, switched to the respective interfaces of the microcontroller, or one included in the component of the sensor that provides the digital converted signal. Many A/ converter modules include a multiplexer as well, which allows the connection of several sensors, whose data can be read and converted subsequently, but this property of the converter also implies a decrease in the sampling frequency per channel divided by the number of sensors connected [4].

Conversion from digital RAW value to voltage value - implies the inverse operation of Analog-Digital conversion, thus obtaining the physical value of the voltage level at the pin of the microcontroller to which the signal is applied. In many cases, this conversion has a linear dependency where the equation of the straight line can using the minimum and maximum values from the ADC associated with respective voltages. For example, 0.5V - 0.1023 for the converter with a 10-bit resolution. From this point of the evolution of the signal, it can be assumed that the measurement process is finished, and the voltage value is transferred to an internal data structure, so the signal is ready to be processed with Software methods.

Software signal saturation - is a function similar to the one in the electronic domain and presents the limitation of the signal value between a Minimum and a Maximum value. When in the electronic domain saturation was a mechanism of electrical protection of the electronic interfaces, in the Software, this mechanism limits the field of

definition of value to minimize the computing effort with large numbers, which, however, has no interest because they are outside the definition domain.

Median Filtering - represents the filter that excludes local minimum and maximum pulses from the signal, caused, for example, by electromagnetic spikes because of commutation in the environment. The median filter also called statistical filter or salt and piper, can be implemented by sorting on a sample window from the input signal and extracting the value from the middle. In this way, the local minima and maxima, after sorting, reach the edges of the sorted vector, and in the center is the most probable value for the given sample [5]. The recommendation is for the sample window to have an odd number of elements to facilitate the selection of the median, Fig.13.

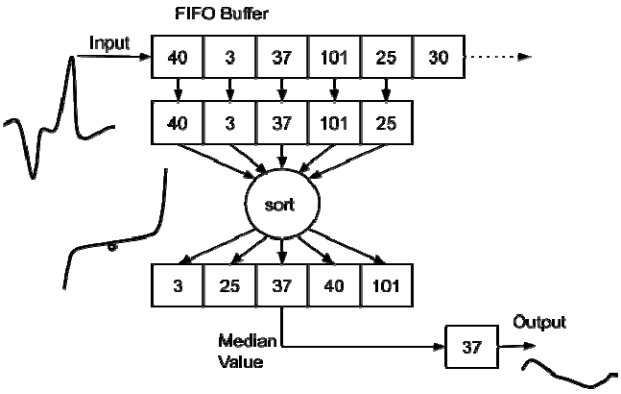

Fig.13. Dataflow diagram for sold & piper filter implementation

Weighted Average Filter - represents a low pass filter (LPF) for the exclusion of uniform noise, also called white noise. The weighted average is applied to a sample window from the input buffer signal. For efficiency in terms of performance, it is recommended to have a minimal sample window, which for low-performance microcontrollers can be reduced even to the last acquired two values. If the weights are equal, the simple arithmetic average formula could be applied.

$$M(X)=\frac{x_1n_1+x_2n_2+\ldots+n_ka_k}{n_1+n_2+\ldots+n_k}=\frac{\sum_{i=1}^{k}x_in_i}{\sum_{i=1}^{k}n_i} \quad (1)$$

Where x(i) represents the signal values in the sampling window, and n(i) are the weights of the values in the obtained result, Fig.14.

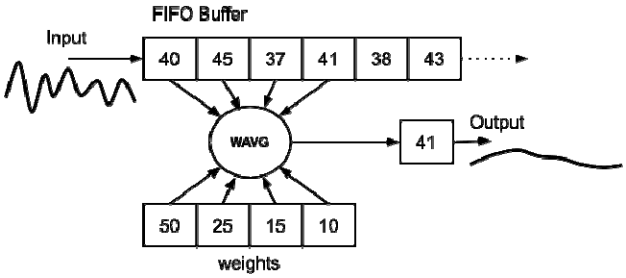

Fig.14. Dataflow diagram for weighted average filter

Voltage conversion to the physical value - represents the inverse process of converting the voltage to the physical value associated with the sensor, which converts a physical value into voltage, but in value as internal information used in software processing. As the basis of calculation for this conversion is the input/output dependency that can be found in the datasheet of the sensor or obtained experimentally. In the case of linear dependency, the canonical equation of the straight line is used with two sensor-specific reference points, Fig.15.

$$\frac{x-x_1}{x_2-x_1}=\frac{y-y_1}{y_2-y_1} \quad (2)$$

Where (x1, y1) and (x2, y2) are the associated reference points as x - voltage, y - physical size.

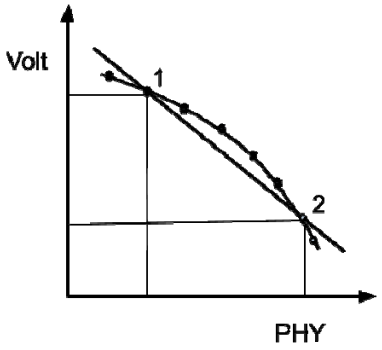

Fig.15. Example of a nonlinear transfer function for a sensor

In the case of nonlinear dependency, Fig.15., the experimental curve is divided into small segments according to the tables collected experimentally, with a linear approximation, on which also could be applied the conical equation of the straight line for conversion.

3.2.3 Strengths and Weakness

There are cases when the same conditioning operation could be implemented in software either in hardware. On choosing the best method, it is

necessary to analyze the strengths and weaknesses of those methods.

The strength of the electrical conditioning is the operability, on the other hand, the weak part is the fact that the electronic components involve costs, and in the case of the elements of high stability and precision, these costs have a high impact on the production. The reduced flexibility is another weakness. The adjusting elements entail additional costs and are often uncomfortable with handling, as well as their reliability. An even bigger problem is on changing the conditioning method, which involves the complete reorganization or replacement of the entire electronic part.

The weakness of the Software conditioning is its operability, assuming it involves processing time, which is shared between several functionalities of the system. On the other hand, programmable resources can be reprogrammed and do not involve material costs. Modifications and adjustments are made even during the operation of the system. This property has the consequence that more and more functionalities are being transferred from the electrical domain to the Software domain, involving the reduction of the production costs of the devices.

3.2. Sensor Services

The Services for the sensor are provided by the upper-level SW of the sensor component. It is the producer of services provided to the Application and ensures the functionalities of managing the component, as well as the interaction with the other services from other components under RTE. As the provider of the environmental information to upper layers, to the application via RTE, here in the service layers are implemented the functions prefixed by *Get, Scan, Read, Recv* to bring information from the environment. Also, here is the right place for the sensor manager allows to build special sensor features based on multiple sources of information. A good example could be the implementation of virtual sensors, which are providing information with no gathering information from a real sensor, but by a special model instead.

Also, here in the service layer should be implemented the Sensor specific symptom detection and diagnostic qualification, providing this information to the rest of the system. A couple of generic diagnosis symptoms for sensors could be named, same as the qualification methods and type of reactions associated, that are presented in the subsections below.

3.2.1 Diagnosis Symptom

Diagnosis is the identification of the nature and cause of a certain phenomenon. Diagnosis is used in many different disciplines, with variations use of logic, analytics, and experience, defining the proper "cause/effect model. In system engineering and computer science, it is typically used to determine the causes of symptoms, mitigations, and solutions [6].

The symptom of the diagnosis comes as an indicator of detection of the situation in logical size "0", "1" or LOW, HIGH. As primary diagnoses on the signals received from the sensors, we will identify the following:

Threshold symptom - It involves comparing a signal with a preset value. If the signal value is higher than a preset, we have diagnoses of Over Value (e.g. Over Voltage, Over Temperature) and complimentary for diagnosis of Under Value (e.g. Under Voltage, Under Temperature), Fig.16.

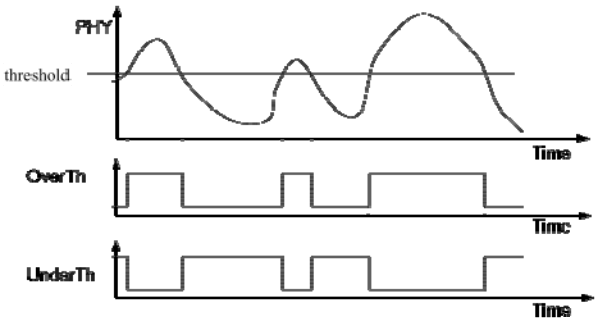

Fig.16. Threshold symptom detection

Range Symptom - Assumes the position of value between two preset values. In this case, we could identify four diagnostic signals - In range, Out of Range High, Out of Range Low, Fig.17.

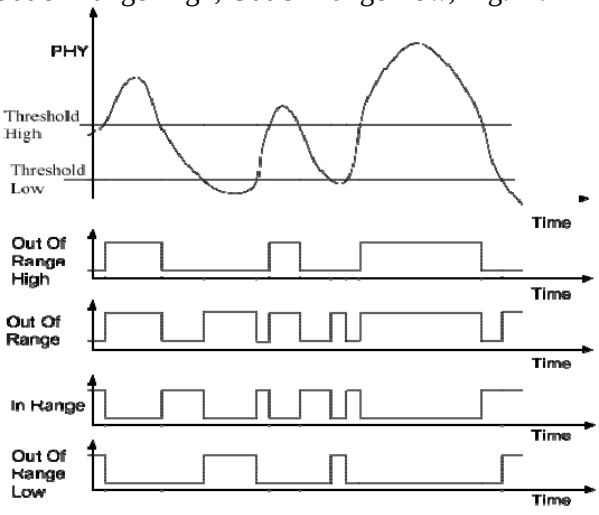

Fig.17. Range symptom detection

Plausibility Symptom - This diagnosis verifies the plausibility of measurements based on the acquisition of the same parameter with two or more sensors. If the signal difference is higher than the permissible limit, a diagnostic symptom will be generated, Fig.18.

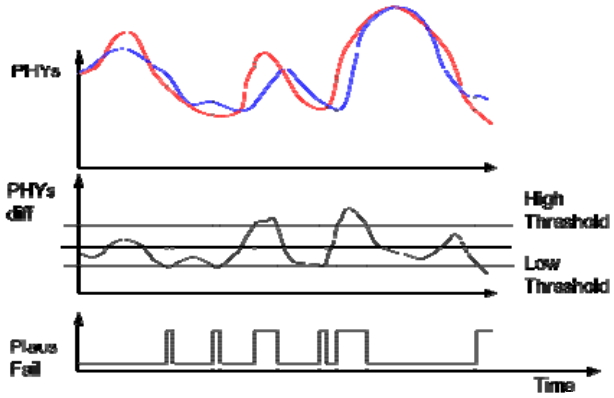

Fig.18. plausibility symptom detection

Stall in the range Symptom. Regardless of the nature of the signal, there are always some variations on the acquired data. this diagnosis detects the freezing of the signal evolution, which could be because of a signal source dysfunction, such as, for example, a short circuit or connection damage, Fig.19.

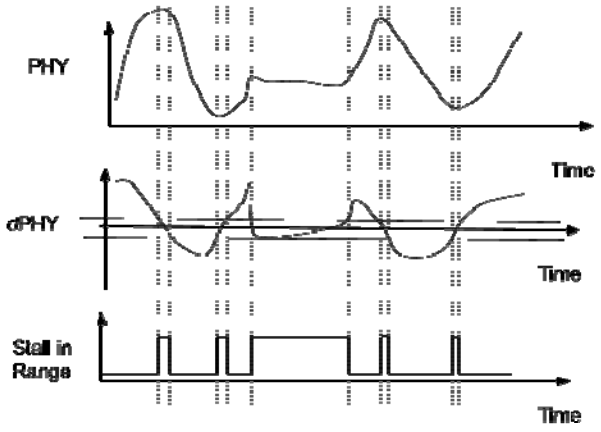

Fig.19. Stall in range symptom detection

3.2.2 Diagnosis Qualification

For any diagnosis symptom to be considered as a valid one and be processed by the sensitive systems on these diagnoses, preliminary processing is necessary in order to qualify them first. The qualification of diagnoses could be implemented by a binary filter, which assumes that a diagnosis can be qualified or disqualified if the symptoms persist longer than a specific period. The binary filter in this context filters a binary signal and could be implemented with an Anti-Bouncing Counter, Fig.20.

Qualification: When the symptom appears, the counter will increase with a default value ABC_INC. Upon reaching an ABC_MAX value, the diagnosis will be considered as qualified and will keep the active state until its disqualification. The counter in case of valid symptom is saturated to a maximum value.

Disqualification: When the symptom disappears, the counter will decrease with a preset value ABC_DEC. Upon reaching an ABC_MIN value, the diagnosis will be considered as disqualified and will remain in this state until the next qualification. The counter in case of no diagnosis symptom is saturated to the minimum value.

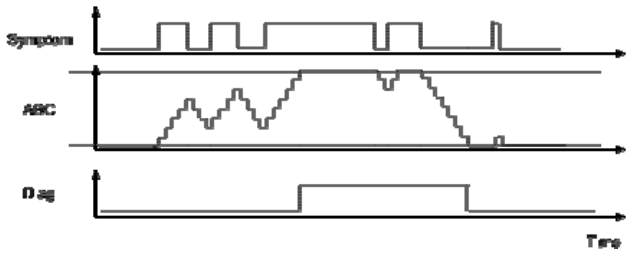

Fig.20. Qualification with antibounce counter

3.2.3 Reaction for Qualified diagnostics

Once a diagnosis is qualified, it should be associated with a specific reaction. The ordinary reactions could be listed as follows:

- ***Inhibition*** - involves ignoring the detected symptom for a given period. For example, a situation occurred where the diagnosis is not relevant, or another higher priority situation persists.
- ***Informing*** - involves the transmission of a message to a user interface for information purposes, for example, a recommendation to charge the device battery.
- ***Blocking*** - involves disabling certain functionalities during the of a valid diagnosis, for example, blocking a mechanical action during the presence of a person in the danger zone.
- ***Derating*** - involves limiting the functionalities when detecting specific diagnoses — for example, limiting the power consumption when a low battery detected or limiting the screen brightness to the laptop. Derting can be binary or follow a derating function following a Min function between the applied power and the derating function depending on the input signal.

3.2.4 Internal vs. external diagnosis

Depending on the signal source for diagnoses, certain specific situations can be identified, and the proper reactions performed, Fig.21.

The internal diagnosis or self-diagnostics checks the internal symptoms of the system operation in order to ensure that the system is behaving in admissible limits, and no system failure occurs during the device operation. Detecting internal diagnostics in early time could prevent system failures involving the proper reaction before any serf-damage to the device.

For example, if the RANGE diagnostics will be applied on a RAW signal, before performing any conditioning, the detection will detect that the sensor provides voltage levels that are outside of the passport data, respectively, as the conclusion - that the sensor may be defective or is a connection problem, electrical, either short on the ground or power supply, or broken wire. In this case, the reaction could be to disconnect the functionalities dependent on the respective signal source.

The external diagnostics are performed to detect some symptoms of the environment and are generated for the functional behavior of the system. In the case when the RANGE diagnosis is on the physical value, meaning that the diagnosis point is after the conditioning, and the diagnosis of detection sensor defects is ok - the conclusion may be that the ambient temperature range is above the allowable norms, and the reaction may be a recommendation message, or derating to avoid the evolution of the situation. A more concrete example would be overheating the engine at high speeds.

Like the conditioning, the diagnoses and reactions can be implemented both sides, in the Electronic domain and in the Software domain, the implementation methods are domain-specific, but following the same principles.

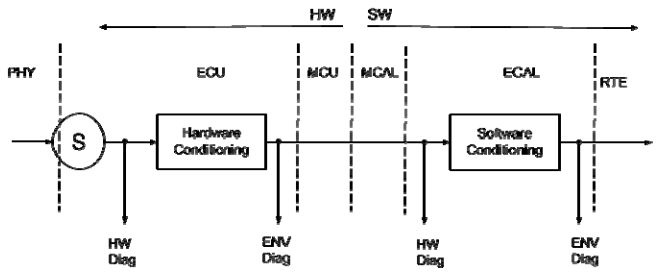

Fig.21. Diagnosis source example

4 Conclusion

Considering the analyses presented in this paper, a methodology and design process was established that resulted in a modular system. It was proposed reuse of the components designed and applied to the processing of a specific signal also for the other signals. This principle allows even the existing system to be extended to other parameters of interest, by adding a sensor and configuring the system to treat the signal source, similar to the existing ones.

A study was performed in order to identify a generic method for a system definition based on layered principle for a Spread out the electronic device as an IoT system.

A generic architectural concept was proposed for a device definition, as well as a layered architecture for a component was proposed.

A typical signal conditioning methodology was defined, and the diagnostics approach for the signal dataflow.

Following the proposed concept architecture for an Internet of Things System, a prototype for Environment Map Acquisition was developed. It consists of a Coordinator and multiple IoT Devices. The acquisition devices collect data about the environment and transmit it to the IoT network. The server coordinator collects the information via IoT network from the IoT Devices and uses it for its various application proposal. The functionality of the IoT Device was implemented following the layered architecture concept as defined in this paper same as the system architecture of the coordinating server. In the server coordinator, the sensor information is provided via the communication stack, transferred on the service layer to the sensor component stack and provided to the Application.

References:

[1] Catalin-Virgil Briciu Ioan Filip Franz Heininger "A new trend in automotive software: AUTOSAR concept" Applied Computational Intelligence and Informatics (SACI), 2013 IEEE 8th International Symposium. DOI: 10.1109/SACI.2013.6608977

[2] Arkady Zaslavsky, Prem Prakash Jayaraman, "The internet of things: Discovery in the internet of things" Magazine Ubiquity Volume 2015 Issue October, ACM New York, NY, USA DOI: 10.1145/2822529

[3] Layered Software Architecture - AUTOSAR Release 4.2.2. https://www.autosar.org/fileadmin/user_uplo

ad/standards/classic/4-3/AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf [accessed Oct 22, 2019].

[4] Bräunl T. Sensors. In: Embedded Robotics. Springer, Berlin, Heidelberg , (2003) ISBN 978-3-662-05101-6

[5] Steven B. Leeb, Alfredo Ortiz, Robert F. Lepard, Steven R. Shaw, and James L. Kirtley, Jr., “Applications of Real-Time Median Filtering with Fast Digital and Analog Sorters” , IEEE/ASME TRANSACTIONS ON MECHATRONICS, VOL. 2, NO. 2, JUNE 1997

[6] A Guide to Fault Detection and Diagnosis - https://gregstanleyandassociates.com/whitepapers/FaultDiagnosis/faultdiagnosis.htm [accessed Oct 22, 2019].