A Middleware Based Architecture for the Industrial Internet of Things

Ioan Ungurean[1,2], Nicoleta Cristina Gaitan[1,2] and Vasile Gheorghita Gaitan[1,2]
[1] Stefan cel Mare University of Suceava, Str. Universitatii 13, 720229 Suceava, Romania
[2]Integrated Center for Research, Development and Innovation in Advanced Materials, Nanotechnologies, and Distributed Systems for Fabrication and Control (MANSiD), Stefan cel Mare University, Suceava, Romania
[e-mail: ioanu@eed.usv.ro, cristinag@eed.usv.ro, gaitan@eed.usv.ro]
*Corresponding author: Ioan Ungurean

Received February 13, 2016; revised April 2, 2016; revised May 3, 2016; accepted May 20, 2016; published July 31, 2016

Abstract

Due to its potential, the Internet of Things (IoT) begins to be used in the industrial field. Nevertheless, although the concept called the Industrial Internet of Things does not know a widely accepted definition, it is based on the existing technology, and introduces the IoT in the industrial environment through the compliance with specific additional requirements and by integrating various fieldbuses with distinctive features, from the real-time point of view. The aim of this paper is to propose a practical architecture for the Industrial Internet of Things. The proposed architecture is designed around the Data Distribution Service for Real Time System (DDS) middleware protocol, and it is organized on four levels. Between these levels, it is defined standard interface, and they can be developed independently. For this reason, the architecture is scalable because at the middleware level, the DDS can be replaced by other middleware systems. At the final, it is performed a comparison between several middleware systems used in the proposed architecture.

Keywords: Industrial Internet of Things, Data Distribution Service, Fieldbus, OpenDDS

This paper was supported by the project "Sustainable performance in doctoral and post-doctoral research PERFORM - Contract no. POSDRU/159/1.5/S/138963", project co-funded from European Social Fund through Sectorial Operational Program Human Resources 2007-2013, , using the infrastructure from the project “Integrated Center for research, development and innovation in Advanced Materials, Nanotechnologies, and Distributed Systems for fabrication and control”, Contract No. 671/09.04.2015, Sectoral Operational Program for Increase of the Economic Competitiveness co-funded from the European Regional Development Fund.

http://dx.doi.org/10.3837/tiis.2016.07.001 **ISSN : 1976-7277**

1. Introduction

Proposed by Kevin Ashton MIT Auto-ID Center, the term "*Internet of Things*" (IoT) refers to the connection between the information provided by RFID and the Internet [1]. Initially, this paradigm included only RFID and wireless sensor networks [2], but currently it has been expanded to any technology that can connect physical objects to the Internet, in order to activate the communication between them. Basically, this paradigm refers to the device to device or machine to machine communication, using the Internet infrastructure, and is aimed at interconnecting things from the virtual world with things from the physical world.

Currently, although the literature in the field proposes several definitions and a number of initiatives to standardize the concept, there is no universal recognized definition for IoT. For example, one possible definition is provided by ITU (International Telecommunication Union) through the ITU-T Study Group 13 that works on the standardization of networks of the future. This group defines IoT as [3] "*a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies".*

Another standardization initiative is that of the working group ISO/IEC JTC 1/SWG 5 Internet of Things (IoT) of the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) [4]. This group, with the international secretariat in Republic of Korea (Korean Agency for Technology and Standards), includes the IEEE Standards Association that manages the IEEE P2413 Working Group Standard for an Architectural Framework for the Internet of Things (IoT), including representative companies from the ICT field [5].

Part of the Internet of the Future, the IoT is under a continuous and dynamic development. For this reason, many industrial companies are interested in the concept, bringing forth on the market the concept of Industrial Internet of Things (IIoT) [6]. This concept refers mainly to the interconnection of objects from industrial environments, where there are many solutions for sensor networks. Based TCP/IP stack protocols and using various communication protocols, IIoT must include the existing infrastructure and enable the interconnection of objects. All solutions should enable critical activities that are specific to the industrial field and must be decoupled from the Internet. Moreover, the data published on the Internet must meet different requirements in terms of integrity, authenticity, confidentiality, and security.

General Electric asserts that IIoT has a great potential in the commercial area, estimating that it can generate a turnover of 514 billion by 2020 [7]. Furthermore, the amount of raw data that must be processed and analysed will exponentially grow [7]. The Consulting company McKinsey & Company estimates that the development of the IIoT will increase productivity, and by 2025, it will reach an added value of 2.7 to 6.2 trillion dollars per year [8]. Cisco also analysed the market potential of the IoT and estimated that in the next decade, IoT will interconnect 50 billion devices and will generate an additional profit of 14 trillion dollars, due to increased productivity [9]. A successful example increased productivity due to the implementation of IoT is provided by the Airbus European aircraft manufacturer which uses the "System on Module" solution provided by National Instrument [10]. The increase in productivity was generated by the acceleration of the manufacturing process and by the avoidance of errors caused by the human factor.

The most of the IoT solutions are based on middleware systems to transport information on the Internet. In order to measure the performance under different operating conditions of these solutions, it can be used the following metrics: CPU usage, memory usage, disk usage, network load, failed transactions (%) over time, input/output – bytes received/transmitted per second (bandwidth), throughput of requests, number of middleware-related incidents (errors), etc. Because these solutions can run on systems with limited resources, the focus is on metrics that measure the resources utilization in certain operating conditions.

As a solution for the IIoT, we propose and present in this article an architecture including communication systems (fieldbuses [11]), used in the industrial environments, and that can provide real-time facilities and predictability using a middleware system allowing the setting of different Quality of Service parameters. Furthermore, at the level of things, intelligent embedded systems can be developed, that can implement hard real time requirements specific to certain industrial processes.

Furthermore, this paper is organized as follows: section 2 presents the most important IoT architectures already existing in the industrial field, as well as the middleware system used to develop them; section 3 describes the architecture for the industrial IoT proposed by the authors; section 4 focuses on several considerations regarding the implementation, whereas section 5 presents various tests performed for the proposed architecture; section 6 contains the final conclusions.

2. Related Work

There are many papers in the specialized literature regarding the field of the IoT [12]. Most of these papers focus on various ways to add Internet connectivity to the things from the physical world, through RFID or wireless sensor network technologies. Such architectures are presented in [13][14]. Others research papers are oriented towards specific applications, such as those presented in [15][16]. There are also projects focusing on security issues 0 or even on the development of relationship between objects similar to human interaction [17][18].

Nevertheless, in the relatively new field of IIoT, there are much fewer papers. For example, [20] is a study regarding the use of the IoT in industry; here, [20] the authors analyse the existing bibliography to determine the trend of the IoT concept in industry and future research directions. However, despite the fact that authors reviewed the use of IoT in the food supply chain, healthcare, fire-fighting and transportation, they mainly focused on a single industrial area, that is mining.

Industrie 4.0, defined by the German government as the future industrial development strategy using industrial IoT [21], is considered the 4th industrial revolution that will lead to the development of smart factories (factories of the future) through the Cyber-Physical Systems (CPS) and IoT. The main purpose of Industrie 4.0 is to secure the future of German manufacturing industry as a European leader. For the efficient development of this strategy, a working group has been formed, carrying out research and development activities in the following key areas: standardization and reference architecture, managing complex systems, comprehensive broadband infrastructure for industry, safety and security, work organization and design, regulatory framework, and resource efficiency. At the present moment, all German companies from the industrial area responded by providing practical solutions for Industrie 4.0. The working group published a document that defined a reference architecture organized on four levels, each level defining the requirements for the architecture development [21]: the Internet of Thighs (with requirements regarding the connectivity

provided by IPv6), Internet-based system & service platforms (with requirements regarding the model-based development platforms), the Internet of Services (with requirements regarding the knowledge of business areas and applications), and top-level Applications (with requirements regarding the access to markets and customers).

An interesting research project related to the use of IoT in industry is IoT@Work [22], led by Siemens AG and integrated into a FIAT factory. In this project, it was defined the specifications of a reference architecture organized on five layers: Field/Control Infrastructure and Network, Device and Network Embedded Services, Device Resource Creation & Management Services, Application Level Middleware Services, and Automation Applications. A middleware system based on the standard Advanced Message Queuing Protocol (AMQP) is used to distribute information. This architecture enables automatic configuration with low human intervention and contains effective mechanisms to ensure reliability, security and real time.

Herman Storey (co -chair ISA 100) et. al., presenting a discussion regarding the use of IoT in industry [6], propose the term of Industrial Internet of Things. The authors stipulate that the introduction of this concept in industry can lead to a convergence of the very diversified automation communication systems, to a greater flexibility, and to a significant reduction of costs. For this purpose, the architecture proposed by ISA100.15 Wireless Backhaul Backbone Network Working Group, that may be used for IIoT [23], contains a Common Network Interface which can be connected to the existing communication systems by using a protocol translator.

Most architectures in this field use middleware systems in order to distribute information via the Internet. Currently, there is no middleware system dedicated to the IoT concept, but there are several candidates in this sense. The most important middleware systems (each having at least 10 independent implementation [24]), which can be used in the development of IoT architecture, are the following: Advanced Message Queuing Protocol (AMQP), Distribute Device Data for Real Time Systems (DDS), Extensible Messaging and Presence Protocol (XMPP) [26], Message Queuing Telemetry Transport (MQTT). AMQP is used for server-to-server communications, MQTT and XMPP are used for Device-to-Server communication; DDS is used for Device-to-Device [27] communications.

Among the protocols listed above, XMPP does not support QoS parameters and MQTT supports three QoS parameters, namely: at most once, at least once, exactly once. AMPQ has QoS parameters in RabbitMQ implementation. From the four protocols listed above, the most extensive set of QoS parameters are provided by the DDS: liveliness, reliability, history, durability, resource limits, deadlines, lifespan. In addition, they can be considered thigh that has no QoS parameters but several attempts to implement, and OPC UA RESTfull who have not specified QoS parameters. In addition, it can be considered Coap [24] that has no QoS parameters but several implementation attempts, and OPC UA [24] and RESTfull that do not have specified QoS parameters.

There are no many solutions for IIoT and existing systems are based on middleware systems. Industry 4.0 includes a middleware level where it is used in OPC UA, IoT@Work uses the AMQP middleware system, and ISA100.15 architecture includes a middleware level without specifying the middleware system used. For this reason, we will make a comparison between the main middleware systems that can be used to develop the IoT and the IIoT solutions [25]. AMQP is messages oriented and allows communication point to point and publisher-subscriber. The main advantages are reliability and security. As disadvantages can be specified that for systems with more than a thousand of nodes the performance

significantly decreases and complexity increases [25]. DDS is a data centric middleware protocol based on the publisher-subscriber paradigm with peer to peer communication. Among the advantages can be included scalability, real time and it has several commercial and open source implementations. It has not a very high complexity and it can be used on embedded systems with limited resources. The main drawback is security; the security specifications being in development stage [25]. XMPP is a message-oriented protocol based on XML. Among the advantages of this protocol may include scalability to thousands of nodes, it can include security, and it can be executed on systems with limited resources because its complexity is small. The main drawback is that it is based on text messages. For this reason, the volume of data transmitted by nodes is much higher than binary encoding [25]. MQTT is an open protocol based on messages via a central broker. The main advantage is that it was designed to be simple and can be used on devices with limited resources. Due to its simplicity can be hacked very easy, and it has bad performance if it is used for more than 1000 nodes [25]. OPC UA is a machine-to-machine communication protocol designed for industrial environments. The main advantage is security, and between disadvantages can be noted: it is based on web services; it does not have real-time capabilities, and it is based only on the client-server paradigm.

After analysing these middleware systems, we chose to use DDS protocol in order to develop the proposed IIoT architecture. The reason is that the device-to-device communication DDS protocol offers standard interoperability and Quality of Service facilities and, therefore, can be used in systems with real time requirements. Furthermore, the protocol operates on the publisher-subscriber paradigm that is more suitable for the IoT than the client-server paradigm, as it allows the development of peer-to-peer communication architectures. For the development of the IIoT architecture proposed in this paper, we have also chosen to use the OpenDDS implementation.

3. The Proposed Industrial IoT Architecture

Before describing the proposed architecture for IIoT, it can be defined a theoretical example based on the definition of the IoT. It is desired to design a system that may have at the application level virtual things and virtual images of physical things (represented by temperature, pressure, or other data acquired from physical things). These objects can exchange data with each other and can display data in a graphical form for the user. These interconnections can be achieved regardless of the nature of the things, the only restriction being the data format. At the application level, it is transparent from where data come and what middleware system is used. Furthermore, the field busses used by the physical devices and their characteristics are transparent. Because it is desired to design an IIoT, it will be used a middleware system with real-time capabilities, and it will be used a lower level to the middleware to transmit data in the same format regardless of the field buses used. Starting from this theoretical example, it can be elaborated several practical examples. An example might be a biodiesel plant which is controlled by means of several PLCs interconnected via a CANOpen field bus. These PLCs have analog/digital inputs that can acquire data from process and analog/digital outputs that can send command to the actuators form the process. The CANOpen field bus is connected to a computer system via USB-CAN interfaces. All analog/digital inputs/outputs are seen in the application levels as things that can be interconnected or connected to the virtual things. Through virtual things, the user can view the data from the process in a graphical form and can send command to the process. Through these things, it can monitor temperature and pressure in the production chain, consumption of

raw materials and the amount of biodiesel produced in a certain time. This monitoring can be achieved within a local network or remotely via the Internet. All real-time commands are performed by the PLCs. A similar scenario can be the monitoring of flow meters that can measure consumption of raw materials in an industrial process. These flow meters are interconnected via MODBUS filed bus and by using a USB-RS485 interface which are connected to a PC. At the application level, these things are connected with virtual things in order to display information in a graphical format. Another scenario may be the monitoring of the electrical energy consumption in a smart build. It can be used a set of smart plugs that can monitor electrical energy consumption via a relay, they can stop/ start the power. These plugs are interconnected through a wireless ZigBee filed bus and through a ZigBee-Modbus gateways are connected to a PC. At the application level, it can be monitored energy consumption of each plug and may turn off/on the power depending on certain conditions defined by the user.

This section describes the proposed IIoT architecture based on the DDS middleware standard with OpenDDS [28] open source implementation. The general block diagram of the proposed architecture is presented in **Fig. 1.** It is based on a solution using the OPC specifications that has been extended and developed in order to implement the Industrial IoT concept based on the DDS middleware protocol. Unlike the OPC specifications based on the client-server paradigm [29], the proposed architecture is based on the publisher-subscriber paradigm [24] that [29] allows the development of peer-to-peer networks and facilitates the communication between IIoT things. The proposed architecture is organized on the following 4 levels (see **Fig. 1**): Things level; Data provider level; Middleware level; Application level.

All sensors and devices that can acquire information from the environment in which they operate can be found at the *Things level*. These devices and sensors can be connected to wireless or wired networks (fieldbuses), used either in the industrial field (CANOpen, Modbus, Profibus, EtherCAT, etc.) or in building automation (BACnet, Lonwork, ZigBee, etc). In order to integrate these networks and fieldbuses in the proposed architecture, simple interfaces (e.g. USB-RS485, USB-CAN) can be used. Besides these interfaces, and in case the applications require real time facilities (frequent in industrial environments), intelligent devices developed around a micro-controller can be used.

The next level is the *data provider*. The purpose of this level is to acquire data from the fieldbuses and to provide this data in a consistent form to the middleware level, regardless of fieldbus type. Furthermore, this level must transmit the data received from the middleware level to the fieldbuses that will reach the object to which it is addressed. In order to acquire or to send data to the devices connected to the fieldbuses, at this level, a wrapper (software module) has been developed for each type of fieldbus included in the proposed architecture. The purpose of these wrappers is to connect the fieldbuses to the PC and to hide the communication details of the fieldbuses to the upper levels. These software modules provide the data acquired from fieldbuses and provide the same interface for transmitting data to the fieldbuses.

The purpose of the *middleware level* is to share the information on the Internet. In this case, we chose to use the DDS middleware protocol, working on the publisher-subscriber paradigm. More than one subscriber may subscribe to a data publisher, for this reason the developers are able to design peer-to-peer architectures. For the development of the IIoT architecture, we have chosen OpenDDS open source implementation of the DDS standard.

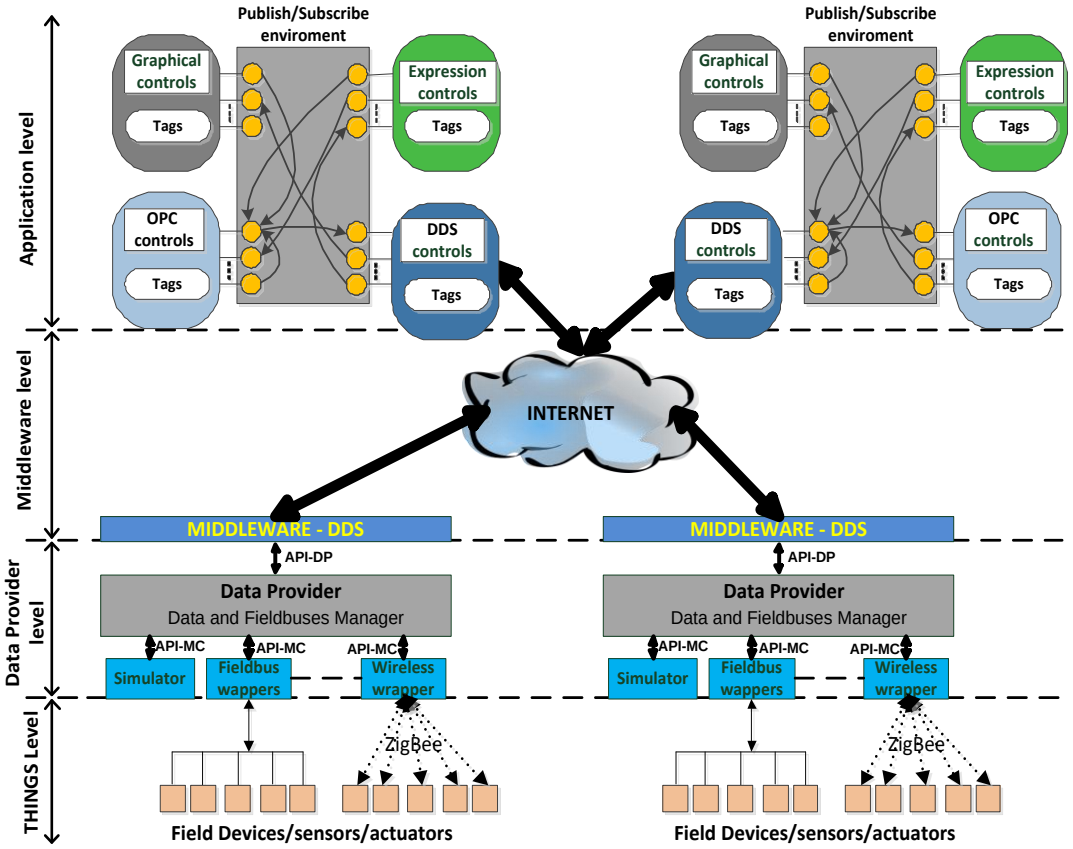

Fig. 1. The proposed Industrial IoT architecture

At the *application level*, the HMI-Application presented in [30] was further developed by adding a DDS object. An instance of this object will allow the communication between HMI-Application and the application that acquires data from the fieldbuses. Each DDS object is associated with a DDS domain having its own QoS parameters, and, if there are requests to publish data with other QoS parameters, the new DDD object should be instantiated.

4.1. The Things Level

At the *Things level*, there are industrial things (sensors, actuators, and devices for monitoring and control) that need to be integrated into the IIoT architecture. In order to accomplish the functionalities for which they have been developed, these things, interconnected via wired or wireless fieldbuses, can operate independently, or can communicate with other things connected to the same fieldbus. At this level, the infrastructure must meet the requirements specific to the industrial environment: hard real-time, reliability, and predictability. The infrastructure must not be changed significantly by the proposed IIoT solution.

An important issue in the development of IIoT architectures is the large number of existing fieldbuses, each with its own characteristics in terms of hard real time. A list of fieldbuses [31] includes the following: AS-i, BSAP – Bristol Standard Asynchronous Protocol, CC-Link Industrial Networks, CIP (Common Industrial Protocol), CANOpen, DeviceNet, ControlNet, DF-1, DirectNet, EtherCAT, Ethernet Global Data (EGD), EtherNet/IP, Ethernet Powerlink, FOUNDATION fieldbus – H1 & HSE, HART, HostLink, Interbus, MACRO Fieldbus,

MECHATROLINK, MelsecNet, Modbus, Optomux, PieP, Profibus, PROFINET IO, RAPIEnet, Honeywell SDS, SERCOS III, SERCOS interface, GE SRTP, Sinec H1, SynqNet, TTEthernet.

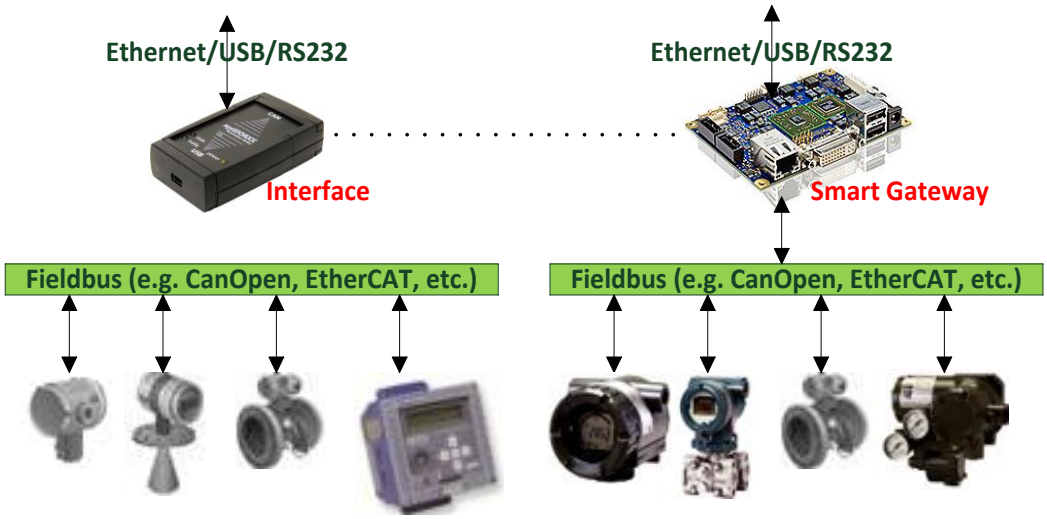

Fig. 2. Things level for the proposed IIoT architecture

In order to integrate these fieldbuses in the IIoT architecture, simple interfaces can be used to connect them to a computer system containing the top-levels of the IIoT architecture (e.g. USB-RS485, USB-CAN, Ethernet-CAN, etc.). Furthermore, for managing one or more fieldbuses, various smart devices can be developed. In order to meet the requirements for industrial environments (real time, reliability and predictability), these devices can be developed around a microcontroller and a real-time operating system [32] and transmit the data to the upper level, using a known communication protocol, such as MODBUS TCP/IP. It should be noted that the hardware support for the development of complex smart devices can take over some higher-level tasks (data provider). At present, there are microcontrollers based on ARM Cortex M4 architecture, capable of operating at a frequency of 200 MHz (there are multicore solutions, such as NXP LPC4300 from [33]) with different communication peripherals that can run intensive applications with hard real time facilities (using a real-time operating system). The microcontrollers based on ARM Cortex M7 architecture, to be launched in 2015 [34], might represent a solution.

In this case, for the thing level, the architecture of the proposed solution is presented in **Fig. 2**. One or more fieldbuses can be connected to computers which may be geographically distributed and communicate through the Internet infrastructure.

4.2. The Data Provider Level

This level, aimed to acquire data from fieldbuses and to transmit it to the middleware level, contains a software communication module (as a software library) for each fieldbus. The purpose of this module is the implementation of both a specific fieldbus communication protocol and of the acquisition cycle [11], or even of the master specific to the fieldbus. Each software module exposes the same interface (API-MC from **Fig. 1** and **Fig. 3**) for receiving and transmitting data. This interface is used by the data provider (a separate software module) for acquiring data from the industrial networks and for storing them in a cache buffer memory. This memory is used to respond more quickly to requests from the middleware level.

As noted in the previous section, currently, there is a large number of fieldbuses and devices that connect to these fieldbuses and arguably, this great diversity represents one of the main problems in the development of the IIoT applications. Developing a unified approach to handle the fieldbuses, can lead to a significant increase in productivity. In this sense, three major solutions can be identified by analysing the problems of describing the devices: EDDL (Electronic Device Description Language) [35], FDT (Fieldbus Device Tools) [36], and EDS (Electronic Data Sheet) [37]. Another initiative, named FDI (Field Device Integration) [38], aims to integrate EDDL and FDT technologies, using OPC UA. The existing solutions for device description do not cover all fieldbuses; moreover, their complexity (such as EDDL) makes them difficult to understand by an automation engineer.

Fig. 3 describes from a logical point of view the architecture of this level. On the upper level, the API-DP interface, used by the middleware level to access data from the fieldbuses is defined. This level is associated with a list of fieldbuses with which it can communicate (it can also communicate with one or more fieldbuses of the same type, each with its own communication parameters and different communication ports). Based on this list, wrapper software modules are instantiated. Each wrapper has a list of devices, each associated with an EDS description file. These description files are used by the wrapper modules for the application of a buffer (cache) memory. Based on this information, a tree-structured address space is created, with the first level containing the list of fieldbuses (each with a unique description), and with the second level containing the device list for each fieldbus (each with a unique description and an associated EDS file). The objects for each device are on the third level, whereas the data members of each object are on the forth level. A data type (integer, real, string etc.) restrictions in terms of access (read only, write only or read-write) are assigned to each member. This address space will be exposed to the middleware level through API-DP interface.

Furthermore, this level contains the Data and Fieldbus Manager, a software module that allows the configuration of fieldbuses from the system (inserting/deleting a fieldbus, scanning a fieldbus, introducing the description for fieldbuses, devices, objects and other object members); this configuration is performed through a graphical user interface.

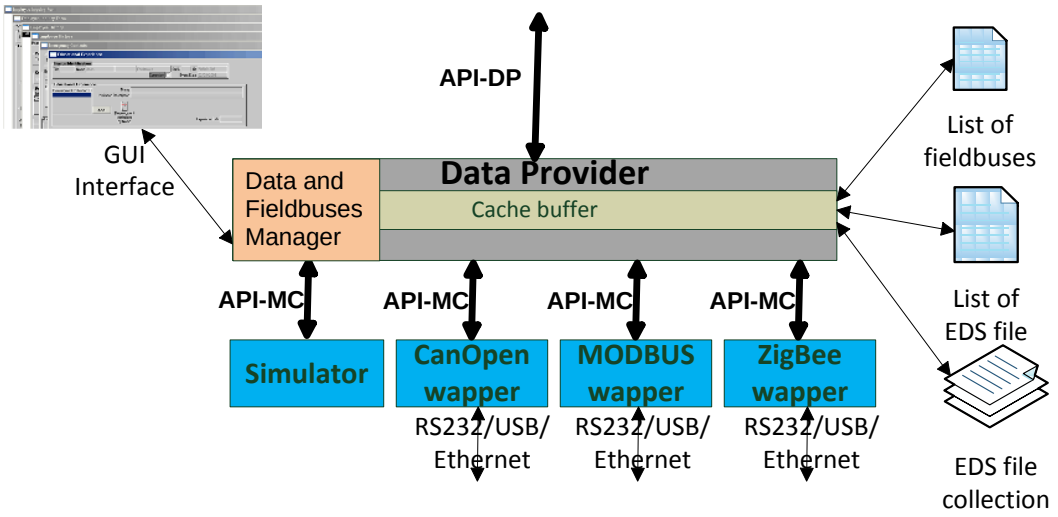

Fig. 3. Data provider level for the proposed IIoT architecture

4.3. The Middleware Level

This level is intended to distribute information in a local network, or even the Internet, in order to activate the IoT concept. The lower level provides a standard interface (API-DP, described in **Fig. 1** and **Fig. 3**) that allows the use of several types of middleware systems. This architecture was first developed for middleware systems based on OPC specifications (OPC DA, OPC .NET, OPC UA), and we employ it in the present paper from the point of view of the DDS middleware standard. This standard allows the development of the peer-to-peer network architectures. Another important feature of this standard is the ability to configure the QoS parameters that allow the achieving of the real-time performance required in industrial processes. This is a big advantage, compared to OPC specifications, widespread in SCADA systems.

The Middleware level is represented by an application that retrieves data from the Data Provider and publishes it via the DDS middleware protocol. In this case, the application publishes data in a domain that has its own set of QoS parameters. Data members are chosen from the address space provided by the lower level, and published in the domain as topics. The application can subscribe to one or more topics in order to transmit data to devices connected to fieldbuses. The first step in activating the concept of the Internet of Things is achieved by allowing at this level the interconnection of things.

4.4. The Application Level

At this level, there is an application that enables/allows both the graphical visualization of information acquired from the fieldbuses, and the transmission of data that will reach the devices connected to the fieldbuses. This application can instantiate three types of controls: graphical, middleware and expression. Each control has a set of data members that can be inter-connected via a mathematical expression. Because of this structure, and depending on the user's requirements and preferences, graphical interfaces can be easily created. **Fig. 4** presents graphically these interconnections. The application from this level contains a publisher-subscriber environment, used by the controls to both publish its data members and to connect them to those of other controls, published in the same environment.

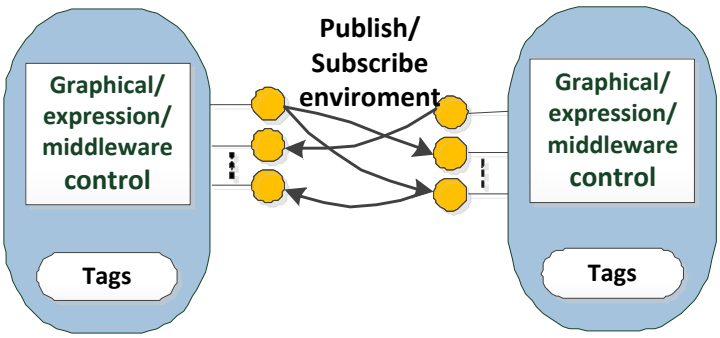

Fig. 4. The interconnection of controls for the HMI-Application

DDS middleware control is one of the objects within this application. It allows the publishing of data from the application environment through the DDS standard, or the subscribing of the published data on the DDS domain, to which the control is connected. If it is necessary that the control connects to a different domain, another DDS middleware control must be instantiated.

4. Implementation Considerations

From the software point of view, the proposed architecture consists of two applications that can be interconnected using different middleware systems. The two applications are referred to as Acquisition-Application (includes Things, Data Provider and Middleware levels) and HMI-Application (includes Middleware and Application levels). Each application implements a security mechanism, in order to restrict unauthorized accesses.

The Acquisition Application includes Things, a Data Provider and Middleware levels as various software modules. This application is developed in C++, and it distributes data from fieldbuses via the OpenDDS middleware (an open-source implementation of the DDS middleware standard). The first step in the development of this application was to define an IDL interface, specific to the DDS protocol. In this case, the following interface has been defined:

```
#include "orbsvcs/TimeBase.idl"
module MCPI_DDS
{

        #pragma DCPS_DATA_TYPE "MCPI_DDS::ItemTopic"
        #pragma DCPS_DATA_KEY  "MCPI_DDS::ItemTopic ticker"
        enum ItemTip { NUMERIC, LOGIC, TEXT};

          struct ItemTopic
          {
                                string ticker;
                                double valueN;
                                boolean valueL;
                                string valueS;
                                TimeBase::TimeT timestamp;
                                short Quality;
                                ItemTip Type;
};
```

It should be noted that the interface is simpler than in the case of the OPC DA, OPC.NET, OPC UA specifications, or in the case of DAIS (Data Acquisition from Industrial System) standard for CORBA [39]. This simplicity results from the fact that the functions specific to connection, connection management, discovery of items from DDS domains, publishing and reading data are implemented in OpenDDS, and offered as standard services. This interface allows the interconnection of several Acquisition-Applications that are distributed geographically, or the interconnection with HMI-Applications. Acquisition-Applications may publish or subscribe data in a single domain with certain QoS parameters, described in **Fig. 5**.

The goal of the proposed architecture is not the improving of the QoS parameters since this would imply the changing of the middleware system and DDS protocol that defined the use of QoS parameters. The code will not have a number as big of tests performed, and it cannot be said that it is used a mature middleware implementation. Efficient use of QoS parameters will be set by the user in the system configuration stage in accordance with real time requirements of the application.

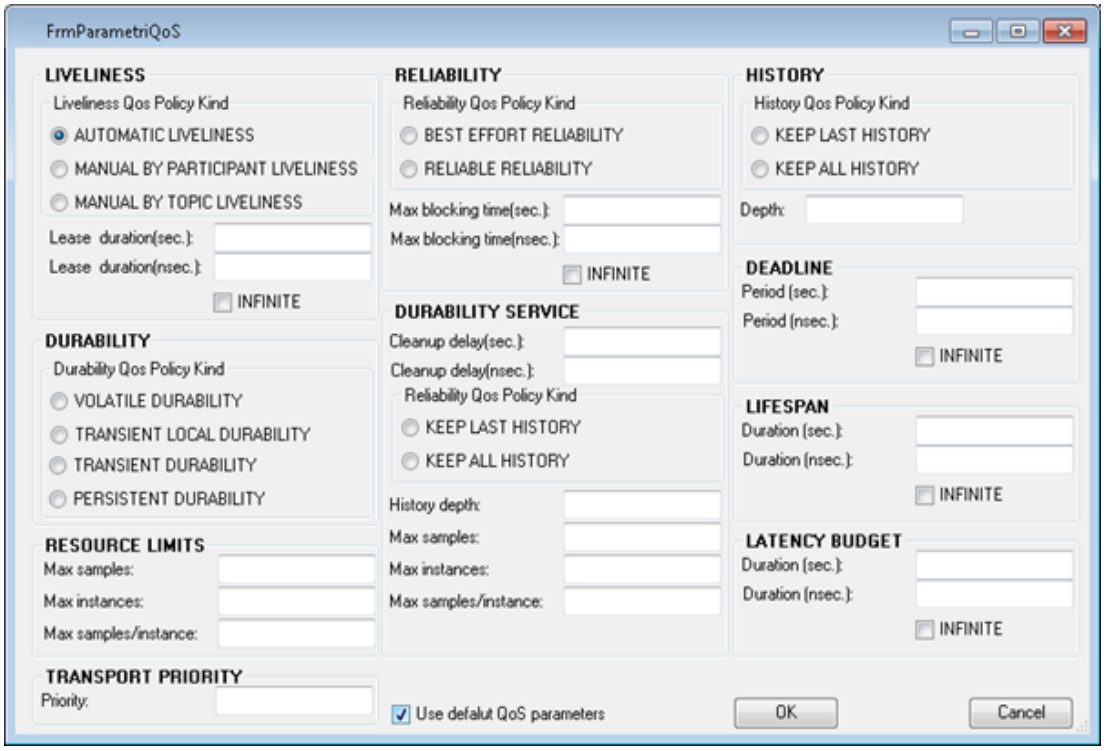

Fig. 5. QoS parameters for a DDS domain

The application uses a dll library that implements the Data Provider level. This library exports the API-DP interface (see **Fig. 3**), represented by the following functions: *LoadNetworkConfiguration()* – inits the list of fieldbuses from the systems, *ShowNetManagerDlg()* – displays the graphic interface of the Data and Fieldbus Manager, *sds()* – sets the security level used by the Data Provider, *GetItemProperties()* – determines the list of item properties, *Browse()* – performs the browse operation in the address space provided by the Data Provider, *GetItemID()* – gets the ID associated to an item, *SyncRead()* – reads the values for a list of items, *SyncWrite()* - writes the values for a list of items, *GetItemAttributes()* – gets the values of the attributes (proprieties) for an item. The *ShowNetManagerDlg()* function enables the configuration of the fieldbuses from the system; afterwards, the settings are saved in an encrypted XML file (including the devices connected to the fieldbuses and the EDS files associated to these devices). This XML file is used by *LoadNetworkConfiguration()* to allocate cache memory and to load the module wrapper specific to each fieldbus.

The Data Provider can be instantiated by other applications that allow the distribution of the information through other middleware systems. It should be noted that a fieldbus can be managed by a single provider. The Data Provider will instantiate a library (wrapper module) for each type of fieldbus from which data is purchased. Furthermore, EDS - Electronic Data Sheet is used for the description of the devices from the fieldbuses. EDS enables the description of the device capabilities connected to fieldbuses; the EDS file is updated by adding a new section specific for the fieldbus communication protocols. This allows the integration of new devices in the system, without the recompilation of the software modules. If it is necessary to introduce a new fieldbus type, or to use other communication interfaces, a

software module and a communication protocol must e developed, both specific to the PC and to the fieldbus interface. The fieldbuses wrapper software modules export the API-MD interface (see **Fig. 3**), represented by the following functions: *AddCommunication()* – adds a fieldbus, *DefCommunication()* – defines the communication parameters, *ModifCommunication()* – modifies/defines the communication parameters, *StartCommunication()* – starts the acquisition, *StopCommunication()* – start/stops the acquisition, *SetPointerEvFct()* – sets the function pointer used to transmit data to the Data Provider, *ScanNetwork()* – scans the fieldbus in order to identify the devices connected to the fieldbus, *GetAchisitionPARAM()* – gets the acquisition parameters for the acquisition cycle, specific to the fieldbus, *SetAchisitionPARAM()* sets the acquisition parameters for the acquisition cycle specific to the fieldbus, *StopScan()* – stops the scan operation, *DelCommunication()* – deletes a fieldbus, *AddDevice()* – adds a device to the fieldbus, *GetCommunicationDescription()* – gets the description of the fieldbus. At the present moment, wrapper modules for Modbus TCP/IP, Modbus RTU, CanOpen, BACnet/IP are being implemented.

The HMI Application is developed in C# and contains several types of controls: middleware, graphics and expression. Each control has a set of data members. The application enables the creation of projects that instantiate controls; these controls can be interconnected using the publisher-subscriber paradigm. Graphical controls are used to display information according to the user's needs, and middleware objects are used to bring or send data using different middleware systems. At the present moment, controls are developed for the following middleware systems or technologies: OPC DA, OPC UA, OPC.NET, TAO (The ACE Orb) and OpenDDS. These middleware controls enable either the exposure of the remote server's address space, or the transmission of data to remote fieldbuses. A detailed description of the application from the application level can be found in [30]. In the case of OpenDDS control, the IDL interface presented for the Acquisition-Application has been implemented. This interface was developed as a C++ dll that exports a set of functions which can be called from C#.

5. Experimental Results

Based on the OpenDDS middleware system in relation to other middleware systems, this section describes the tests performed for the IIoT architecture proposed in this paper. The tests were conducted within a local network comprised of eight identical computers (AMD Athlon (tm) 64 X2 Dual Core Processor 4200+ 2.21GHz, 1GB of RAM and Windows 7 as operating system) interconnected by a switch with 100Mbs Ethernet ports. The Acquisition-Application is executed on one computer, with different middleware systems for data distribution, and a simulator module for fieldbuses (a wrapper module that generates data). The HMI-Application, reading a set of 16 items associated with different analogue values from the first application, is executed on the other seven computers. The software architecture of the test network is shown in **Fig. 6**.

The 16 items are periodically read at a refresh rate of 100ms, 200ms, 300ms, 400ms, 500ms and 1000ms. All these data were read using the following technologies: OPC UA with TCP as transport protocol, OPC.NET with TCP as transport protocol, OPC DA, the ACE ORB (TAO version 6.2.5) and OpenDDS with TCP as transport protocol. For the OpenDDS, the Acquisition-Application was set to generate data at the desired acquisition refresh rate, whereas, for the other technologies, the HMI-Application connects to the Acquisition-Application and creates a group/ list/subscription (depending on the technology

used) with the desired refresh rate. For all cases considered, the bandwidth was monitored and measured with the Colasoft Capsa software package. **Fig. 7** shows a synthesis of these tests. In addition, there are monitored the follow metrics: CPU load, memory usage and number of bytes for each message using Wireshark tool.

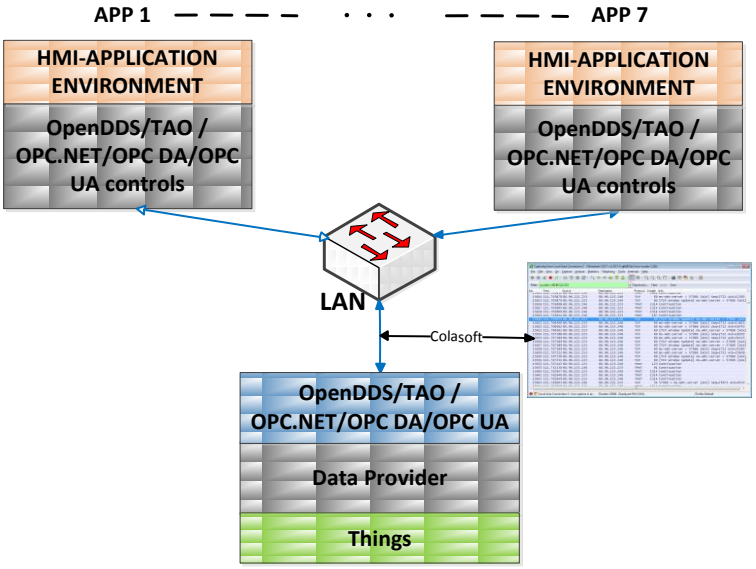

Fig. 6. The software architecture of the performed tests

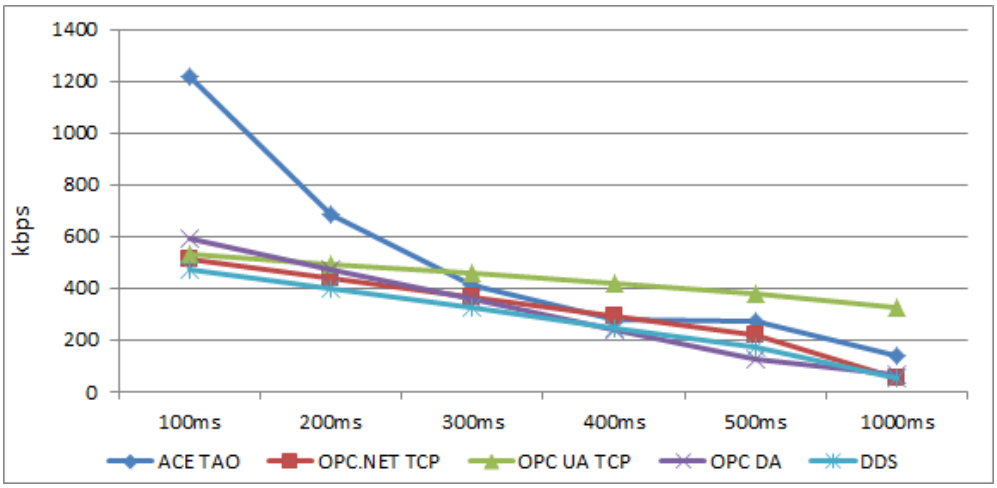

Fig. 7. The bandwidth for the experimental tests

For the OpenDDS, the tests were conducted with all QoS parameters set to default values. It can be easily seen in **Fig. 7** that the bandwidth, limited to a local network, is very close to the OPC DA (very common in SCADA systems). Moreover, the DDS control can set various QoS parameters. In the case that these parameters are not met, the control can send events which can be used to take countermeasures for meeting the requirements of the monitored industrial process. Tests were conducted within a local network, because the control of the industrial process with real-time requirements cannot be achieved through the Internet, where the network load is very important.

From **Fig. 7**, it can be seen that the bandwidth occupied by DDS is in the same trend with the others middleware used. This is because it is transmitted the same set of data. Another factor is represented by the transport protocol used for data transmission. At the refresh rates less than 500ms, DDS has a lower bandwidth than other protocols, which means that it can achieve better performance when it is needed to obtain real-time performance within a local network. Data publication of in the Internet (which is a best-effort network type) is likely performed at the refresh rates greater than or equal to 1 second. In these cases, according to **Fig. 7**, DDS uses about the same bandwidth as the others middleware systems used in the tests presented in **Fig. 7**.

Since not all middleware systems used in comparison provide QoS parameters, there was not performed a comparison related the QoS parameters. The comparison of the parameters QoS cannot be performed in terms of bandwidth. QoS parameters are used to set the operation of the system under certain constraints. For example, it can be configured the durability QoS parameters when a DataReader subscribes to a topic must receive previous data transmitted in this topic. Deadline parameter specifies the maximum time between two samples, and if it is not met this time, the system can perform different actions specified by the user. Resource limits parameter enables the user to limit the resources used by the system. If these resources are exceeded, the system calls a function in which the user can specify operations that will be performed to fit limited resources. Liveliness parameter allows detection of disconnections or death connections.

From the point of view of the memory footprint for a refresh rate of 100ms for 16m items, the working set for the Acquisition-Application with ACE-TAO is about 11MB, for OPC.NET TCP is 22MB, for OPC DA is 10MB and for DDS is 14MB, while the processor load is 22% for ACE-TAO, 36% OPC.NET TCP, 8% for OPC DA and 15% for DDS.

Table 1 presents the number of bytes of Ethernet frames sent by the Acquisition-Application in order to update 1 items. This information was obtained with Whireshark tool. The smallest frames are obtained by DDS, but the other middleware systems have a number of bytes very close.

Table 1. The number of bytes at the Ethernet level

	ACE-TAO	**OPC.NET TCP**	**OPC DA**	**DDS**
1 items	456 Bytes	314 Bytes	351 Bytes	306 Bytes

6. Conclusions

In this paper, the authors propose a four-level architecture for the Industrial Internet of Things. This architecture was practically implemented, and it is consolidated on an open-source middleware system, based on the DDS standard. Furthermore, the proposed architecture aims to integrate into the system as many fieldbuses used in industrial environments or smart building as possible. For this purpose, a description method for the field devices is used, in order to either introduce new fieldbuses, or to easily integrate new types of devices for the fieldbuses already included in the system. Moreover, the architecture is very customizable, allowing the development of various network architectures that are geographically distributed. The architecture is customizable because at the middleware level, it can be used any middleware system with the condition to implement de interface with the Data Provider Level, and at the application level, to implemented a middleware control. Furthermore, at the things level it can be used a wide variety of fieldbusses and devices connected to them. At the

application level, the user can use the control provided by the application to accomplish custom graphical forms in order to display information acquired from the middleware level.

References

[1] Ashton Kevin, "That 'internet of things' thing," *RFiD Journal*, 22 June, 2009.

[2] Govind P. Gupta, Manoj Misra and Kumkum Garg, "Distributed Information Extraction in Wireless Sensor Networks using Multiple Software Agents with Dynamic Itineraries," *KSII Transactions on Internet and Information Systems*, vol. 8, no. 1, pp. 123-144, 2014. Article (CrossRef Link).

[3] International Telecommunication Union - ITU-T Y.2060 - (06/2012) - Global information infrastructure, "Internet protocol aspects and next-generation networks - Overview of the Internet of things," Accessed Feb. 2016. Article (CrossRef Link).

[4] Committees ISO/IEC JTC 001/SWG 05, "Internet of Things (IoT)," Accessed Feb. 2016. Article (CrossRef Link).

[5] IEEE SA, "P2413 - Standard for an Architectural Framework for the Internet of Things (IoT)," Accessed Feb. 2016. Article (CrossRef Link).

[6] Herman Storey (co - chair ISA 100), Rick Bullota and Daniel Drolet, "The Industrial Internet of Things," *Control Engineering*, 15 May, 2013. Article (CrossRef Link).

[7] Jeffrey Moraes and John Darsie, "General Electric (GE) Breaks Out on New, Industrial Internet Project," 18 June, 2013. Article (CrossRef Link).

[8] James Manyika, Michael Chui, Jacques Bughin, Richard Dobbs, Peter Bisson, Alex Marrs, "Disruptive technologies: Advances that will transform life, business, and the global economy," *McKinsey Global Institute*, May, 2013.

[9] John Chambers, "Internet of Everything," *Cisco*, 21 Feb. 2013. Article (CrossRef Link).

[10] Bill Lydon (Ed.), "Airbus Implements Industrial Internet of Things (Smart Tools)," *Automation.com*, 26 September 2014. Article (CrossRef Link).

[11] Vasile Gheorghita Gaitan, Nicoleta Cristina Gaitan, Ioan Ungurean, "A flexible acquisition cycle for incompletely defined fieldbus protocols," *ISA transactions*, vol. 53, no. 3, pp. 776-786, May 2014. Article (CrossRef Link).

[12] Min-Woo Ryu, Jaeho Kim, Sang-Shin Lee, Min-Hwan Song, "Survey on Internet of Things: Toward Case Study," *Smart Computing Review*, vol. 2, no. 3, pp. 195-202, June 2012. Article (CrossRef Link).

[13] Biplob R. Ray, Jemal Abawajy, Morshed Chowdhury, "Scalable RFID security framework and protocol supporting Internet of Things," *Computer Networks*, vol. 67, no. 4, Pages 89-103, July 2014. Article (CrossRef Link).

[14] Li, Shancang, Li Da Xu, Shanshan Zhao. "The internet of things: a survey," *Information Systems Frontiers*, vol. 17, no. 2, Pages pp 243-259, Apr. 2015. Article (CrossRef Link).

[15] Sean Dieter Tebje Kelly, Nagender Kumar Suryadevara, Subhas Chandra Mukhopadhyay, "Towards the Implementation of IoT for Environmental Condition Monitoring in Homes," *IEEE Sensors Journal*, vol.13, no.10, pp.3846-3853, Oct. 2013. Article (CrossRef Link).

[16] Paul J. Reaidy, Angappa Gunasekaran, Alain Spalanzani, "Bottom-up approach based on Internet of Things for order fulfillment in a collaborative warehousing environment," *International Journal of Production Economics*, vol. 159, pp 29-40, Jan. 2015. Article (CrossRef Link).

[17] Zhikui Chen, Ruochuan Ling, Chung-Ming Huang,Xu Zhu, "A scheme of access service recommendation for the Social Internet of Things," *International Journal of Communication Systems*, vol. 29, no. 4, pp. 694–706, Mar 2016. Article (CrossRef Link).

[18] Luigi Atzori, Antonio Iera, Giacomo Morabito, "From "smart objects" to "social objects": The next evolutionary step of the internet of things," *IEEE Communications Magazine*, vol. 52, no. 1, pp. 97-105, Jan. 2014. Article (CrossRef Link).

[19] Kai Zhao, Lina Ge, "A Survey on the Internet of Things Security," in *Proc. of 9th International Conference on Computational Intelligence and Security (CIS)*, pp .663-667, 14-15 Dec. 2013. Article (CrossRef Link).
[20] Li Da Xu, Wu He, Shancang Li, "Internet of Things in Industries: A Survey," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233-2243, Nov. 2014. Article (CrossRef Link).
[21] ACATECH, "Recommandations for implementing the strategic initiative INDUSTRIE 4.0,"Apr. 2013. Article (CrossRef Link).
[22] IoT@Work, Accessed Feb. 2016. Article (CrossRef Link).
[23] Hiroshi Miyata, "Introduction to backhaul architecture model in ISA100.15," in *Proc. of SICE Annual Conference (SICE)*, pp.697-700, 20-23 Aug. 2012.
[24] Héctor Pérez, J. Javier Gutiérrez, "A survey on Standards for real-time distribution middleware," *Journal ACM Computing Surveys*, vol. 46, no. 4, March 2014. Article (CrossRef Link).
[25] Ovidiu Vermesan and Peter Friess, (Eds.), "Internet of Things --- From Research and Innovation to Market Deployment," River Publishers, 2014.
[26] Sung-Chan Choi, Jaeho Kim, Jaeseok Yun, and Il-Yeop Ahn, "A Tutorial for Energy-efficient Communication for XMPP-based Internet of Things," *Smart Computing Review*, vol. 3, vol. 6, pp. 471-479, Dec. 2013. Article (CrossRef Link).
[27] Seung-Yeon Kim, Chi-Hun Lim and Choong-Ho Cho, "Performance Analysis of a Dense Device to Device Network", *KSII Transactions on Internet and Information Systems*, vol. 8, no. 9, pp. 2967-2981, 2014.. Article (CrossRef Link).
[28] OpenDDS, Accessed Feb. 2016. Article (CrossRef Link).
[29] ZhaoXin Guo, XingQuan Xie, ZhiGao Ni, "The application of OPC DA in factory data acquisition," in *Proc. of 2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, pp. 209-212, 25-27 May 2012. Article (CrossRef Link).
[30] Ioan Ungurean, Nicoleta Cristina Gaitan, Vasile Gheorghita Gaitan, "Transparent interaction of SCADA systems developed over different technologies," in *Proc. of 18th International Conference System Theory, Control and Computing (ICSTCC)*, pp. 476-481, 17-19 Oct. 2014. Article (CrossRef Link).
[31] Richard Zurawski, (Ed.), "Industrial communication technology handbook," *Second Edition, CRC Press*, 2014.
[32] Vasile Gheorghita Gaitan, Nicoleta Cristina Gaitan, Ioan Ungurean, "CPU Architecture Based on a Hardware Scheduler and Independent Pipeline Registers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 9, pp. 1661-1674, Sept. 2015. Article (CrossRef Link).
[33] LPC4300 - NXP Semiconductors, Accessed Feb. 2016. Article (CrossRef Link).
[34] ARM Cortex M7 chip promises to give IoT a shot in the arm, Accessed Feb. 2016. Article (CrossRef Link).
[35] Xing Zhang, Feng Liu, "Research on EDDL," in *Proc. of 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, pp. 1224-1227, 21-23 Apr. 2012. Article (CrossRef Link).
[36] Satoshi Noguchi, Kenji Suzuki, Shinichiro Chino, Hiroshi Sakurada, Isao Tarui, Nobuyuki Ban, Pontes Charles, "FDT technology for CC-link network," in *Proc. of 2011 Proceedings of SICE Annual Conference (SICE)*, pp. 1560-1565, 13-18 Sept. 2011.
[37] Wolfhard LAWRENZ, "CAN system engineering." Springer, 2013.
[38] Dirk Schulz, Roland Braun, Ulrich Topp, Martin Stöckl, "Beyond instrumentation: FDI for modular subsystems with proprietary protocols," in *Proc. of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, pp. 1-7, 16-19 Sept. 2014. Article (CrossRef Link).
[39] Data Acquisition from Industrial Systems Specification, Accessed Feb. 2016. Article (CrossRef Link).

Ioan Ungurean received the M.S. and Ph.D. degrees in computer science from the Stefan cel Mare University of Suceava, Suceava, Romania, in 2008 and 2011, respectively. He is currently an Associate Professor with the Department of Computers, Stefan cel Mare University of Suceava. His current research interests include fieldbuses, real-time systems, middleware systems and distributed data acquisition systems.

Nicoleta Cristina Gaitan received the M.S. and Ph.D. degrees in computer science from the Stefan cel Mare University of Suceava, Suceava, Romania, in 2008 and 2010, respectively. She is currently a Lecturer with the Department of Computers, Stefan cel Mare University of Suceava. Her current research interests include digital systems design with FPGAs, microprocessors and microcontrollers systems, fieldbuses, real-time systems, and distributed data acquisition systems.

Vasile Gheorghita Gaitan received the M.S. and Ph.D. degrees from the Gheorghe Asachi Technical University of Iasi, Iasi, Romania, in 1984 and 1997, respectively. He is currently a Professor with the Department of Computers, Stefan cel Mare University of Suceava, Suceava, Romania. His current research interests include real-time scheduling, middleware systems, digital systems design with FPGAs, fieldbuses, and embedded system application.