Integrated Manufacturing Management using Internet of Things

Roberto A. Dias
Embedded and Distributed System Group - IFSC- Brazil
Mauro Ramos Av, 950.
88020-300. Florianópolis – SC

Igor T. M. Mendonça
Embedded and Distributed System Group - IFSC- Brazil
Mauro Ramos Av, 950.
88020-300. Florianópolis – SC

Adriano Regis
Embedded and Distributed System Group -IFSC- Brazil
Mauro Ramos Av, 950.
88020-300. Florianópolis – SC

ABSTRACT
The integration between programmable logic controllers (PLCs) and supervision and control software's has been a critical factor for industrial production management. The goal of this work is to show a new approach for development of an integration framework for industrial processes using the best practices of Service Oriented Architecture (SOA) and the recent integration technology called Devices Profile for Web Services (DPWS). The main result is a functional prototype used in a small-scale bottle filling industrial process, using DPWS technology. Through this scenario was possible to validate the model's feasibility on the industrial environment, making the model's performance evaluation and observing the characteristics of notification, description, discovery, reporting and control messages.

Keywords
Internet of Things, Manufacturing Automation, DPWS, Web Services.

1. INTRODUCTION
The growing need for effective integration and secure data collection from devices installed on the workshop in order to improve the decision making in corporations, demands innovative concepts of production management to increase flexibility and modularity of systems. This is only possible using information technology and communication.

The future of manufacturing industry can be characterized mainly by frequent change of market demands, the time to market pressure, the continuous flow of new emerging technologies and especially the global competition. Thus, it is of great importance to adopt flexible and adaptable technologies [1].

The operating cost of a plant is strongly linked to the facilities installation and maintenance. Whenever an industrial process is altered due to the replacement or modification of a product, reconfigurations should be performed with risks of production interruption, increasing the operating costs. Existing technologies have contributed so far to the development of industrial processes, but its features do not include the new requirements demanded by industries. According to Souza [2] the main problem is that these technologies depend on each device having a controller driver to communicate with a system database. This driver provides information for the communication device with a given system. These solutions are restricted to the system and device supported by these driver or resource, and strongly depends on the connectivity between the device and the database.

There is a need for technologies to improve workshop integration with information management systems, with a direct, independent and transparent platform. A recent alternative is the use of Service-Oriented Architecture (SOA) that, according Papazoglou [3]: "It is the characterization of distributed systems, where the system's functionality is exposed through an interface that allows description, location and invocation through a standardized format." The SOA has as main objective, the ability to connect a wide variety of systems without the use of proprietary software in order to achieve a truly open interoperability.

SOA by itself is a concept that allows two programs that were written in different languages and are running in different operating systems, to communicate in an easier integration by adopting an open and independent platform.

Within this context, this paper aims the development of a model for the integration between existing devices in industrial environments with supervision and control software's using an actual paradigm, "The Internet of Things" [4]. In order to achieve this, we used the concepts of Service Oriented Architecture using the Devices Profile for Web Services DPWS [1].

2. INTERNET OF THINGS
In the present study, we adopted a new approach to integrate data collected in the workshop with enterprise information management systems, using as intermediary software layer (middleware) by use of Web technologies and software architectures that adhere to a new paradigm of development called "Internet of Things" - IoT.

2.1 Definition
According Atsori [4], IoT is a new paradigm that is gaining importance especially in the modern system of wireless communication and identification systems for radio frequency tags (RFID). This new approach favors the development of applications that integrate physical systems such as sensors, actuators, mobile devices and smart phones with an environment of cooperation and interoperability.

In this sense, IoT appears as a "glue" that allows integrating diverse environments. Devices used in everyday life, like sensors and actuators are integrated via the Internet communication systems wired or wireless. Among the main elements used in the implementation of IoT there are the adoption of intermediary software layer (middleware) based on SOA and should be performed in the physical device.

In this work, the Devices Profile for Web Services (DPWS) specification was adopted as a middleware to develop an application based on IoT. The implementation of DPWS adopted in this work was based in the DOT NET Micro Framework (NETMF) ported to a development platform

based on a microcontroller with compatible ARM 7 core, called NETduino Plus.

3. DPWS

The DPWS defines a minimum set of functionalities that allows devices with limited computational resources to execute Web Services (WS). This minimal set is basically formed by an exchange of secure messages provided by discovery services, notification services, events services, description services, allowing direct implementation on embedded devices, in general, without compromising compliance with the standardization of WS.

3.1 Architecture

The DPWS protocol stack is shown in Figure 1 and is composed of standard protocols and some WS extensions. An important detail, shown in Figure 1, is that the specification for the DPWS [5] defines that it needs SOAP and WSDL in version 1.2.

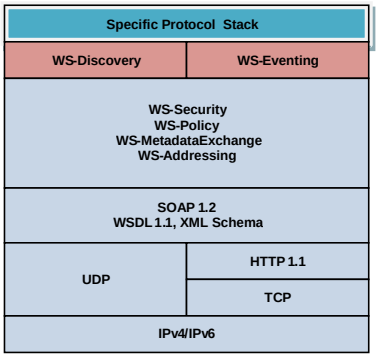

Figure 1. DPWS Protocol stack.

3.2 Standardization

The DPWS specifies the SOAP-over-UDP and Web Services Dynamic Discovery [5]. It was initially developed by a consortium composed by Microsoft, Ricoh, Intel and Lexmark, currently standardized by OASIS, through a technical committee under the alias WS-DD. Currently the three specifications are in version 1.1

3.3 DPWS Model

In the DPWS computational model the devices can take on different roles: consumers of services (clients), services or even both. In the case of services, two types are distinguished: hosting services and hosted services. Figure 2 illustrates the two types of services.

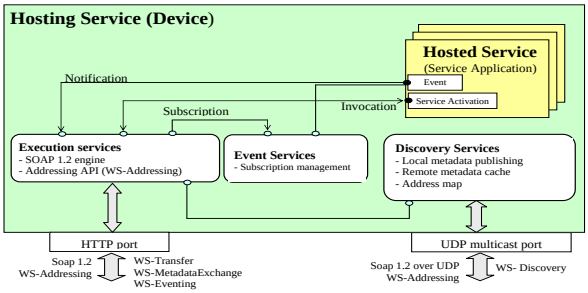

Figure 2. DPWS computational model [6].

The so-called hosting service is an important part of the DPWS model. Several of the non-functional aspects that work in the evolution of application services are concentrated on hosting service components, on the form of built in services. Services that enable dynamic discovery and exchange of metadata (WSDL interfaces and their attachments as WS-Policy, XML Schemas) are examples of these services based on the component device. The engine itself of the Protocol SOAP 1.2 is also part of the component hosting services.

The hosted services are specific WS applications and provide the behavior of the application. A device may have several hosted services, each having their own endpoint and that's where the name "hosted service" comes from. These hosted services are device's dependent. To simplify the use of its hosted services, the discovery service also can be included.

The hosted services involve four areas: discovery services, event services, description and messaging services.

Dynamic Discovery Services are used by devices to advertise their services on a network and be discovered by customers. The WS-Discovery protocol uses the stack of SOAP on the UDP / IP multicast to transmit and listen to the messages of discovery. On "W3C Recommendation" [7] are defined the publishing services (event sources) and the data record on an event source. The combination of application services included with these services allow customers to sign up and receive asynchronous messages (events) produced by hosted services. Metadata for hosted services are available to customers through the use of WS-Metadata Exchange specifications. The message exchanges occur following the specifications of SOAP 1.2, as previously reported. The header information follows the WS-Addressing specification, enabling their availability for any transport protocol (HTTP, SMTP, TCP, UDP, etc.).

4. INTEGRATED MANAGEMENT OF MANUFACTURING USING DPWS

One common problems faced by industry is the integration between programmable logic controllers (PLCs) with SCADA (Supervisory Control and Data Acquisition) and other PLCs. This integration is done through drivers that need to be programmed by the developers of each SCADA system.

One of the existing solutions on the integration of industrial networks is the use of Object Linking and Embedding for Process Control (OLE for process control or simply OPC). The OPC provides a common way to connect data sources like PLCs, automation devices and database with an application server. The OPC connects the sources of data (automation devices) with the applications, like connecting SCADA and the PLC, for example. OPC technology is based on the Distributed Common Object Model (DCOM). Developed by Microsoft for distributed computation, it allows the integration of industrial applications and data sharing using local objects via Local Area Network - LAN [8].

However, many features inherent to the DCOM, affect applications performance in some situations. Some of its disadvantages are:

• It is platform dependent;

• The high complexity of the messages generated;

• The messages sending over the Internet is difficult due to the presence of firewalls [9][10];

Another technology like DCOM is the Common Object Request Broker Architecture (CORBA) that is platform independent, but contains some deficiencies among other DCOM applications.

With this in mind, the integration of PLCs with other devices in the industrial environment, using cross-platform technology that includes features as: self-description of their function, self-discovery, and events orientation, is the main challenge of this work.

4.1 Problem Definition

The first advantage of our approach is the low coupling between the client and server applications. By using SOA, the Web Service is developed to be consumed by the customer with little or no knowledge of the existing code in the Web Service, enabling the interaction between client and server, using only existing operations in Web Service. The lower the coupling, the easier will be to reuse of the code [11].

Another advantage is the use of consolidated Web Services protocols and standards. The XML is used as a way of formatting information and uses the HTTP protocol as a transport. Since HTTP is supported on any platforms that allow viewing of web pages, and XML is a standard transport and data storage that uses plain text, Web services become independent of platforms and systems.

As previously mentioned, the use of the precepts of SOA through Web services as a tool for integration of distributed systems, is already widely used for general IT purposes. But the idea of using the service paradigm for communication of embedded systems is just beginning. Using the SOA, this paper aims to expand the integration of SCADA software and DSS (Decision Support Systems) from the workshop level to the enterprise systems, by the use of services embedded in the PLC.

By treating the CLP as a Web service, it exposes its features, eliminating the need for device drivers or integration technologies such as OPC. Procedures for configuration or reconfiguration can be automated through self-discovery and self-description features inherent to services.

As an example, a beverage plant uses one of their treadmills for bottles of 600 milliliters, but a new requested configuration is needed for bottles of 355 milliliters. This configuration can be done directly by the supervision and control software, through an existing operation in the Web service embedded in the PLC.

Another example is the replacement of a PLC due to any problem. Through the discovery service, the characteristics of the new PLC are obtained. Thus, the Web Service can be automatically selected and configured accordingly to the new PLC without the need of drivers or OPC servers. When the new PLC is installed and connected to the network, a message will be sent to the SCADA software announcing the new PLC and describing its features, which can be aggregated by the supervision and control software.

4.2 Proposed Solution

The model presented in Figure 3 is based on the concepts of SOA in the context of advertising, dynamic discovery and auto description of the services in each element of the industrial environment. To accomplish this, it is proposed the DPWS as an integration technology (middleware) of the various elements.

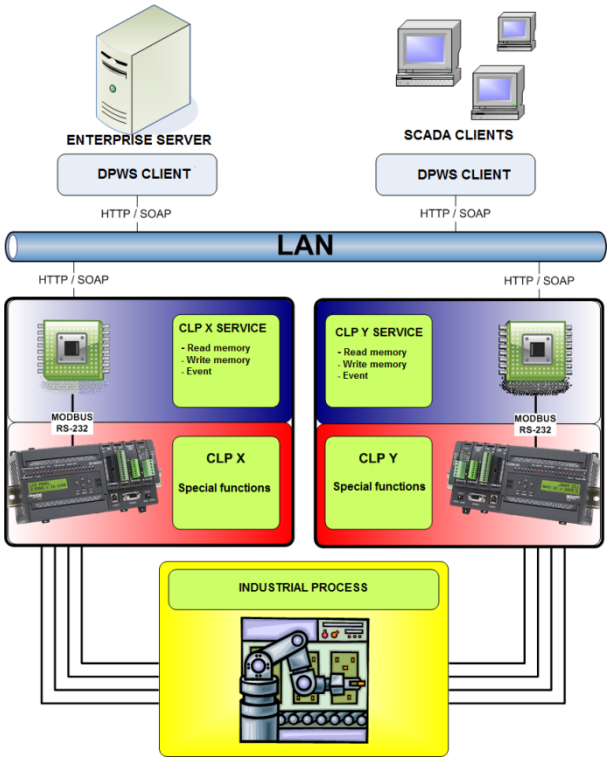

Figure 3. Model Solution Based on SOA.

By embedding a Web service in the device, such as a PLC, it creates the concept of a "universal" driver, which supports any operating system or software platform. Once the PLC is connected to the TCP / IP network, it will be announced, and the applications will have its description in order to use it.

4.3 Development Platform

As a development platform for embedded Web Service, is proposed NETDUINO PLUS [12]. A hardware based on the ARM7 processor manufactured by Atmel. This platform is programmable in C# on the environment Dot NET Micro Framework (NETMF), version 4.1.

The NETMF was developed by Microsoft as a platform for developing embedded systems, using at least 250 KB of RAM. The main objective with NETMF was the development of embedded systems with high-level languages, making faster development of embedded applications and effortless code maintenance.

The development of applications using NETMF can be accomplished through the Visual Studio program, a tool widely used for developing applications on Windows and Internet. Visual Studio, for example, has tools provided for recording, debugging and emulation.

By launching the NETMF 4 in November 2009, Microsoft released its code through Apache 2.0 license, which allows free use and distribution of source codes and the final applications, royalties free [13].

As a validation scenario, was adopted a simple bottle filling industrial process.

The manufacturing process is simple. It contains two sensors, a timer in the PLC and an electric motor. In spite of its simplicity, the process is sufficiently complex to study the characteristics of the model.

4.4 Embedded Web Service

The web service was developed through Dot *NET Micro Framework* (NETMF) API. It was modeled so that we could get and send information to the PLC. To prevent network overload with signaling messages, was created an event service to read the device memory only when a particular memory location has its value changed. It can be seen in Figure 4 the methods used.

Figure 4. PLC Web service Class.

The service was modeled in the form of class "CLP Service", which inherits the features of a hosted service "Dpws Hosted Service".

4.5 DPWS Client

The DPWS client is a simple HTTP client to send messages to a HTTP server and processing received XML messages. Everything is organized so that it is possible to interpret the SOAP messages and control events.

4.6 Security Service

Mendonça [14] proposed a security DPWS extension with the specific use of the WS-Security specification, which was used in this work. The WS-Security is the main security specification for Web services and relies on standards XML-Signature and XML-Encryption to provide secure message exchanges. The specification is intended to be flexible and allows the use a wide variety of security mechanisms. This makes possible to provide support for different types of security credentials (security tokens), multiple signature formats and various data encryption technologies. The simplicity e variety of options is important to achieve interoperability between different technologies.

In the literature, some studies advocate in the most emphatic manner the use of WS-Security in the model of the DPWS. With that, it is possible rely properties of reliability, authenticity and integrity at application level protocols with end to end security guarantees. This is the path taken in the experiments with the DPWS in this work.

In Figure 5 is illustrated as a result, a Security Service used in the DPWS model. This service provides the interception of incoming and outgoing device messages doing the needed WS-Security message processing.

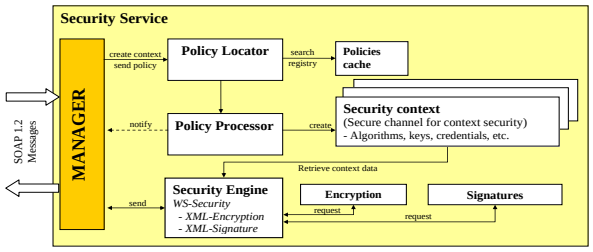

Figure 5. Features Security Service

Table 1 describes the Security Service modules presented in Figure 5 and the features of the elements shown in the same figure. Encryption and signature modules are implemented as a need for the application developed in the device and other modules are fixed elements of the architecture. The reason for the variation of signatures and encryption APIs is the diversity of existing encryption algorithms. Each device may have its implementation according to their computing power and security requirements.

Table 1. Security Service Modules

Modules	Function
Encryption and Signature APIs	These modules are available in the form of libraries and are responsible for data encryption and decryption and rules for signing and verification, respectively. It implementation will vary depending on the needs and capabilities of the device.
Security Contexts	For each secure channel established with the device, is created a security context stored in the form of security policy (WS-Policy).
Policies Cache	This module keeps the policies metadata repository of the service interfaces that compose the device's recent knowledge.
Policies Locator	This component is used for storage and retrieving policies. It identifies and stores in the local cache the policies shared with known peers devices.
Policies processor	Performance comparison of policies. In the process of establishing a secure channel, this module will create a valid security context.
Security Engine	Encrypts, decrypts, sign and verify signatures in SOAP messages according to the WS-Security.
Manager	Link between the components responsible for making decisions and answers to their peers. This module intercepts the messages and defines the need to implement mechanisms and policies that act on the messages.

5. RESULTS

To evaluate the proposed manufacturing management system based on DPWS, it was assembled a prototype bottle filling system according to the diagram in Figure 6.

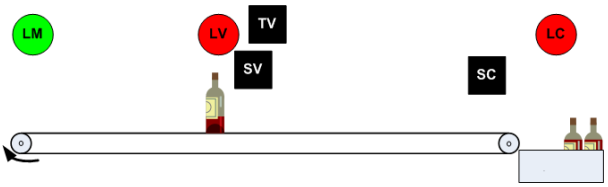

Figure 6. Test Apparatus.

The process starts with the engine running, which is stopped when the bottle reaches the sensor valve (SV). In this point other engine opens the valve (LV) in a preprogrammed time valve interval (TV). This time (TV), sets the amount of liquid to be loaded into the bottle. When the timer reaches the value expected, the valve will close and the motor continues until the end line sensor (SC), indicating that the bottle is in the box. Finally when the last bottle is placed in the box, it will be activated the full box LED (LC), which indicates that the bottle tray is complete.

The DPWS Service is responsible for supervision and control and supports the following features:

• Adjustment of open valve time and number of bottles per tray;

• Start of production;

• Production Stop;

• Sending notices to each full tray completed.

To generate the notification “full box complete", we use an event service, installed in the device.

The DPWS client is responsible for the service discovery, sending the control messages and receiving the full box completed notifications.

Whenever a device containing a DPWS service is connected on the LAN, it sends a *hello* message in the form of IP multicast to any DPWS customer present on the network. The DPWS client program responsible for receiving this notice was called "Explorer". In the Explorer are listed the services of DPWS network and detailed information about each one.

With the "Explorer" each client can conduct a network survey (probe) to verify the existence of all devices on the local network. In the Explorer, a reply message verifies the existence of a particular device and gets a description message of the hosting and hosted services.

The DPWS client captures the announcement of the “CLP service” and can consume the service called "bottler", where they will control and supervise the manufacturing process. Here are the main operations used by the available DPWS service. Through it, we can get and set the valve opening time and amount of bottles to be stored per tray. It is also possible to obtain the number of bottles through a triggered event by every completed tray.

The performance evaluation of the data exchange in an Ethernet 100 Mbps network was performed. In order to evaluate the performance of the DPWS messages exchange it was used the software Wireshark [15]. This software is a network analyzer that captures packets sent and received from the network card, documenting the arrival or departure of a package.

With this tool, the time between request and response messages of DPWS was measured. The results are shown in Table 2.

Table 2 – Messages time response

Operation	Connection Time (seconds)	Request Response (seconds)	Total Time (seconds)
Probe	**No connection**	**1.180**	**1.180**
Restart (oneWay)	**0.355**	**0.547**	**0.902**
readHolding-Registers (twoWay)	**0.357**	**0.872**	**1.220**
presetSingle-Register (twoWay)	**0.349**	**0.798**	**1.147**
subscribe	**0.361**	**1.109**	**1.471**
Register-Changed (Eventing)	**0.023**	**0.052**	**0.211**

Observing Table 2, we can apprehend that trying to control DPWS messages with the inclusion of the security service has resulted in a considerable lag, consuming over a second to simple control messages. But the event messages, that avoid the network scans, are very fast, taking only 212 milliseconds to be completed.

Other measurements were performed with the same mapping, but using OPC in place of Web services. It was found that the reading of the PLC tag (variable) consumes about 480 milliseconds. Similar values were found in the literature [16]. This time is shorter then the regular read Holding-Registers time (two way). However, the OPC has no service notification of events such as the DPWS. The event notification to the process supervisory uses UDP protocol, which explains the low latency found in table 2. So with this feature, the use of the DPWS becomes quite feasible. Furthermore, OPC don’t have security implementation, contrary to the system developed on this work.

6. CONCLUSION

The use of SOA and IoT paradigm for integration between factory plant devices and supervision and control systems is a promising alternative. The use of middleware like DPWS to

this integration showed a poor performance for applications requiring lower latency. However, the objective of this work was to test and to validate the functionality of the model.

As a future work, is proposed the study and implementation of a new approach to supervision and control of industrial networks employing the new OPC-UA specification [17], with SOA support for industrial appliances. The new OPC-UA uses a novel protocol stack that provide Web Services on devices on the similar way of our DPWS approach, but OPC-UA have a more compact and efficient TCP-IP specific TCP-IP stack.

7. REFERENCES

[1] Jammes, F. and Mensch, A. and Smit, H. 2007. "Service-Oriented Device Communications Using the Devices Profile for Web Services", no 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07), Niagara Falls, Ontario, Canada.

[2] Souza, L. M. S. et al. Socrades 2008. A web service based shop floor integration infrastructure. Proc. of the Internet of Things (IOT 2008). Springer; http://people.inf.ethz.ch/mkoehler/papers/IoT08.pdf.

[3] Papazoglou, M. P. (2003). Service-Oriented Computing: Concepts, Characteristics and Directions. Fourth International Conference on Web Information Systems Engineering (WISE'03). Roma;

[4] Atsori, L. and Lera A. and Morabito, G. 2010. The Internet of Things: A survey. Computer Networks. Number 54; pp 2787-2805.

[5] Oasis. Web services discovery and web services devices profile: (WS-DD). 2009. http://http://www.oasis-Open.org/committees/tc_home.php?wg_abbrev=ws-dd>.Last Access 2012 April.

[6] Mensh, A. Rouges, S. 2009. DPWS Core Version 2.1: User Guide. Version 1.0, April 14, 2009. On-line: https://forge.soa4d.org/docman/view.php/8/45/DPWSCore+User+Guide.pdf

[7] W3C Recommendation, 2006. "Web Services Addressing 1.0 – Core", May 2006, on-line: http://www.w3.org/TR/ws-addr-core/, Last access 19/10/2009.

[8] W3C Working Draft, 2009. "Web Services Eventing (WS-Eventing)", online: http://www.w3.org/TR/ws-eventing/, Last Access 20/10/2009.

[9] Pietrzak P. Kyusakov R. Eliasson, Jens. 2011. Roadmap for SOA event processing and service execution in real-time using Timber. 2011 IEEE International Symposium on Industrial Electronics (ISIE 2011): Gdansk, Poland, 27 - 30 June 2011.

[10] Kapsalis, V. et al 2003. Architecture for Web-based services integration. The 29th Annual Conference of the IEEE Industrial Electronics Society (IEEE-IECON'03). Virginia; pp. 866- 871.

[11] Henning, M. 2006. The Rise and fall of Corba. ACM QUEUE, New York, v. 4, n. 5; pp. 28- 34.

[12] Walker, Cris. Getting Started With Netduino. O´Reilly Media. 1st Edition. February. 2012.

[13] Gandolpho, C. 2008. Cresce uso de SOA. Info Corporate. <http://info.abril.com.br/corporate/infraestrutura/cresce-uso-de-soa.shtml

[14] Mendonça, I. Fraga, J. da S. Dias, R.A., 2010. Extensão de Segurança para o Perfil DPWS. X Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais. Fortaleza.CE. Outubro de 2010.

[15] Sanders, Chris. The practical packet analysis. Starch Press San Francisco. 2007.

[16] Galli, P. Microsoft to Open Source the.NET Micro Framework. Port 25: The Open Source Community at Microsoft, 16 November 2009. Disponivel em: <http://port25.technet.com/archive/2009/11/16/microsoft-to-open-source-the-net-micro-framework.aspx>. Last Access: 29 May 2010.

[17] Marcin Fojcik and Kamil Folkert. Introduction to OPC UA Performance. Communications in Computer and Information Science, 2012, Volume 291, 261-270, DOI: 10.1007/978-3-642-31217-5_28