# AN MTCONNECT-BASED MONITORING SYSTEM FOR CNC MACHINES

Paulo Estevão Teixeira Martins*, Marcos Vinícius Teixeira Martins†, Adriel Magalhães Souza‡, Eraldo Jannone da Silva§

*University of São Paulo (USP)
Department of Electrical and Computer Engineering
São Carlos, SP, Brazil

†Federal University of Triangulo Mineiro (UFTM)
Department of Electrical Engineering
Uberaba, MG, Brazil

‡University of São Paulo (USP)
Department of Mechanical Engineering
São Carlos, SP, Brazil

§University of São Paulo (USP)
Department of Production Engineering
São Carlos, SP, Brazil

Emails: pauloetm@usp.br, mvtmartins@gmail.com, adrielmagalhaes@usp.br, eraldojs@sc.usp.br

**Abstract**— With the arrival of the internet as the main means of communication, the interest in monitoring industrial processes has risen. Supervisory systems based on desktop applications had their spot taken over by Web applications that can be accessed from anywhere in the world. Although such supervisory systems are already very present in the industrial environment, there has been great difficulty in applying them to manufacturing systems, due to the non-uniformity of data collected from machine-tools. The MTConnect™ protocol came to standardize the data collected from the shop floor and offer them to a Web application in a standard format. This study describes an application of the MTConnect™ system in a real manufacturing system. The whole architecture suggested for the case study has been laid out, as well as the Web system proposed for the supervision of the processes. The implementation of this system has promoted the centralization of the data of interest and has enabled their use in researching applications, confirming that applications in the Internet of Things (IoT) benefit the 4.0 Industry.

**Keywords**— Internet of Things, Manufacturing Systems, MTConnect™, 4.0 Industry

## 1 Introduction

Production means are based upon the Information Technology (IT) paradigms, considering that its expansion implies in the evolution of manufacturing processes, which tend to follow the rhythm of changes (Sakurai et al., 2016). This scenario is intimately linked to the Internet of Things (IoT), which determines the modus operandi of physical devices used to collect and transmit data through the web (Santucci, 2010).

In order for the IoT applications to be efficient, we need: data exchange through ubiquitous wireless technologies, with due safety (Santos et al., 2013); interoperability; monitoring and control; diversification of devices; among others (Sobral et al., 2015).

The IoT enables the control and detection of devices in a remote way, through the existing network infrastructure. It allows for a direct integration of physical and computing systems, resulting in greater efficiency and accuracy. Monitoring and control are thus benefitted, since that by establishing a direct communication between the components, human intervention is reduced (Mattern and Floerkemeier, 2010).

The aforementioned scenario marks the fourth industrial revolution, also called 4.0 Industry, which introduces a possibility for machines to become more and more autonomous (smart) regarding the possibility for interpreting and sending data about the manufacturing conditions, even reporting possible flaws (Pisching et al., 2016).

4.0 Industry focus on the infusion of several technologies that enable the projection of products, procedures and smart processes. Such technologies bring more freedom and flexibility in manufacturing, promote innovation and increase quality and customization of products in order to cater for the individual demands (Dutra and Silva, 2016). However, even though the model looks attractive, its application is not simple (da Silva et al., 2015).

The automated production of several types of parts depends on the data supplied by a diversified set of equipments used in the industrial environment. Thus, a uniform and strong communication is essential in modern production systems. Remote Machine Monitoring Systems (RMMS) are softwares that have been developed by their companies for this reason (Evans, 2011). Several companies within this field provide RMMS solutions for their clients. Such solutions enable the monitoring of machines by using a client-server structure. However, modern factories operate by using a great diversity of proprietary protocols, which makes it harder to implement standard systems (Mori et al., 2008).

The development of the Asynchronous JavaScript (AJAX) and eXtensible Markup Language (XML) communication technologies has revolutionized the world of World Wide Web (Web) applications, after serving as a basis for applying protocols such as CyberOPC (Torrisi, 2011). Initially created with the intention of monitoring tools with Computer Numerical Control (CNC), CyberOPC created opportunities for the efficient use of monitoring protocols. MTConnect™ steps into this scene aiming to establish a more direct communication between the equipments by using XML-structured data as a pattern in applications for the industry (Sobel, 2009a). In the MTConnect™ data sets are included resources related to the use of production equipments, sensors and several pieces of hardware (Vijayaraghavan et al., 2008).

It is in this current field that this study fits and aims to develop a system for monitoring the CNC machines available in the Laboratory for Advanced Process and Sustainability (LAPRAS), a part of Nucleus of Advanced Manufacturing (NUMA) of the São Carlos School of Engineering (Escola de Engenharia de São Carlos - EESC) of University of São Paulo (USP), based upon the MTConnect™ standard.

The paper is structured into 5 subsequent sections, including this one. In Section 2 the MTConnect™ is described in its basic version for the shop floor monitoring in general. In Section 3 we describe the data standard proposed by MTConnect™ Institute with greater detailing. In Section 4 we introduce the case study with the application of the aforementioned standard. Finally, in Section 5, we express the main conclusions of this work.

## 2 The MTConnect™ System

The ITs integration into production systems faces the barriers imposed by the lack of uniformity in the data generated by the various independent systems. The purpose of the MTConnect™ standard is to provide a common means of communication between these devices through existing technologies (Vijayaraghavan et al., 2008).

The MTConnect™ system architecture is based on terminologies used in its definition (Figure 1).

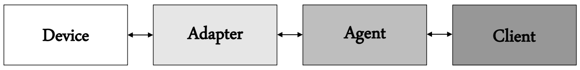


Figure 1: The MTConnect™ System Basic Architecture.

### *2.1 Device*

*Device* is a set of components that provides data about some specific application. Thus, a machine tool can be characterized as a *Device* (Sobel, 2009a).

### *2.2 Adapter*

*Adapter* is a component that connects the *Device* to *Agent*, being optional in the MTConnect™ standard, depending on the *Device* existing in system architecture (Sobel, 2009a). With the MTConnect™ standard emergence in the industrial environment, some manufacturers already provide machines capable of making data available directly to an MTConnect™ *Agent* component.

This *Adapter* component is a TCP/IP server, which in absence of connected *Agents* works in stand-by. When an *Agent* component is connected, the *Adapter* starts to collecting data, verifying the modified data at each iteration and sending the new data to the *Agent*.

### *2.3 Agent*

*Agent* is responsible for implementing the MTConnect™ interface, receiving the data from the *Adapter* component and structuring them in the standard, the XML output (Sobel, 2009a).

The data acquired by the *Agent* is stored in an adjustable size buffer. When the buffer capacity is exceeded, the new data begins to replace the oldest ones, which are thus lost. One way to avoid this is to use a *client* application to read this data and store it in a database. To perform this communication, the *Agent* MTConnect™ plays the role of an HTTP server, processing the *client* component HTML requests and providing standardized data.

### *2.4 Client*

The *Client*, in general terms, is responsible for requesting the XML output data for some specific function. In other words, it is the component that will use the data provided by *Agent* (Sobel, 2009a).

## 3 The XML-based Model

The MTConnect™ model is basically composed of two main types of XML elements: structural elements and data elements. Structural elements describe the physical and logical parts and subparts of a device. Data elements describe the data that can be collected from a device (Lei et al., 2016). Figure 2 illustrates an example machine represented in the MTConnect™ standard model. Its basic components can be classified according to functional and operational aspects.

### *3.1 Controller*

The *Controller* interprets information from the monitored system, returning feedbacks that directly influence the *device* operating conditions. The data items analyzed by the *controller* consist mainly of tool numbers, *controller* modes (manual, automatic) and execution states (ready, active, feed-hold).

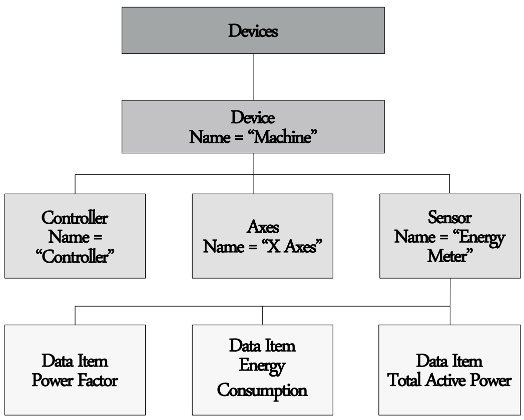


Figure 2: An MTConnect™ standard schematic model.

### *3.2 Axes*

The *Axes* provide information that coordinates the execution of linear and rotational motions. The data items for linear movements consist of position, feed, and acceleration; and the data items for rotational motions determine the speed and mode of rotation.

### *3.3 Asset*

For a complete model description, it is necessary to define an *asset*. It can be characterized as an element that despite being associated with the manufacturing process is not a component of the *device*, which allows its removal without impairing the operation. In the example of Figure 2 the Energy Meter sensor is an *asset* component of the presented *Device*.

## 4 Case Study: MTConnect™ Standard Application

A supervisory system was developed, based on the MTConnect™ standard, for monitoring the following machines existing in LAPRAS: a Zema Numerika G-800HS external cylindrical grinding machine, a Romi D-800 machining center and a Romi GL-240M turning center, which have GE-FANUC CNCs. The MTConnect™ system architecture (*Adapters*, *Agent* e *Client*) implemented in LAPRAS is shown in Figure 3.

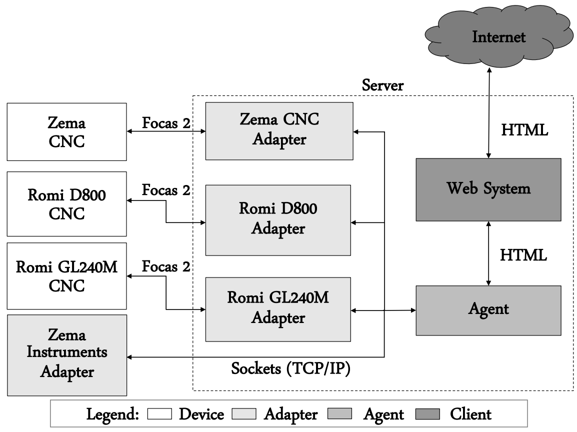


Figure 3: MTConnect™ monitoring system architecture implemented in LAPRAS.

### *4.1 Adapters*

Four *Adapter* components were implemented, as shown in Figure 3. As these programs start, they wait for an *Agent* component connection, and when it is established, they begin to read the device data and send it to the *Agent* component through (TCP/IP) *sockets*.

#### 4.1.1 CNC Adapters

CNC *adapters* were coded using C++ language. These are executed by the server, where each one is responsible for making the connection with one of the three machines mentioned above, through the FANUC proprietary protocol, Focas 2 (Fanuc OpenFactory CNC API Specification 2). Adapters communicate directly with machine CNCs and send collected data to *Agent* MTConnect™ (Figure 3).

#### 4.1.2 Zema Instruments Adapter

The Zema Instruments Adapter was specially developed to standardize the data acquired from instruments installed externally to Zema grinding. These instruments correspond to an electric multivariate power meter (solid state transducer) UPD 600, from Ciber Brasil; a turbine type compressed air flow meter, VTG-019A020211RAA20 model, Incontrol trade mark; a Contech Turbine Cutting Fluid Flowmeter model SVTL 1", with a Contech microprocessed indicator model CTH 2265; and a sound noise meter (decibelimeter), model DEC-460, trade mark Instrutherm. This *Adapter*

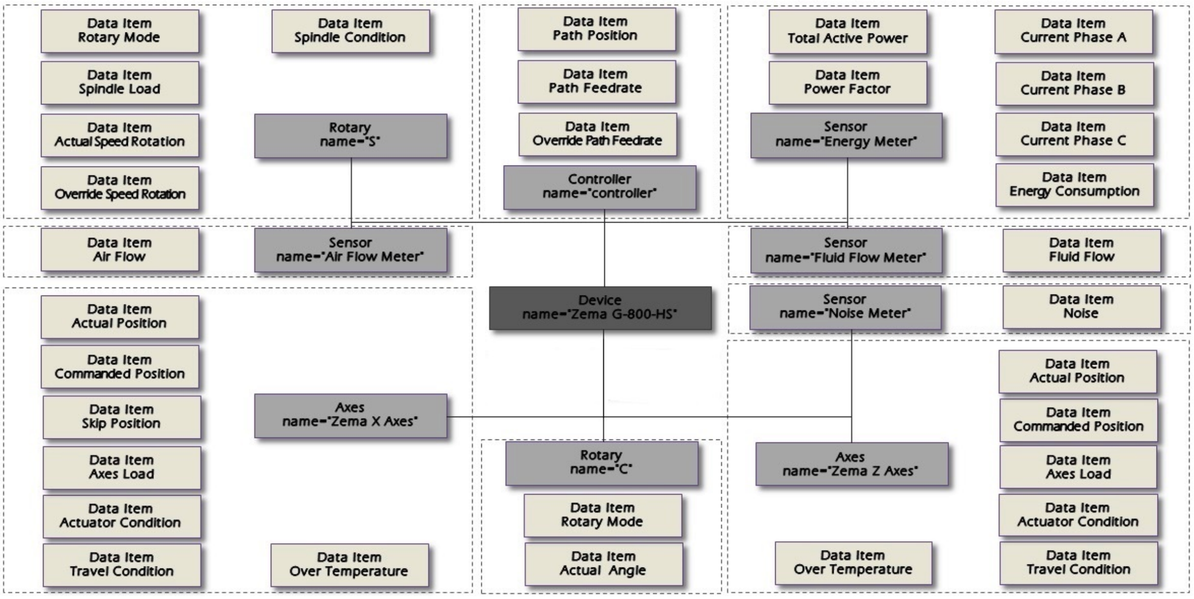


Figure 4: Zema Numerika G-800HS grinding machine MTConnect model.

was encoded using the *Labview* 8.5 version. It captures the information at 5 samples per second (5 Hz), running outside the server computer, but connected to the same internal LAPRAS network.

### *4.2 Agent MTConnect*

For the *Agent* component implementation the 1.4.0.2 cppagent version was used, which is an open source program coded in C++ (Sobel, 2009b). In the system architecture implemented in LAPRAS a single *Agent* component was used (instead of one *Agent* for each *Adapter*), in order to centralize the data flow and facilitate future machines incorporation into the monitoring system. To do this, a simple change in the *XML Input* scope has to be done. For example, to add the Zema machine it is necessary simply create a new device named "Zema G-800-HS" inside the **Devices** structure. It is not necessary to implement a new *Agent*, nor even change the current component code. Figure 4 presents the Zema G-800-HS device XML model, representing all the components of the existing machine tool in LAPRAS.

In order to keep up the system with the cppagent updates it was decided to use the MTConnect Institute cppagent without modification. This choice facilitates the incorporation of new features and benefits into the system, once there is no need to adapt the cppagent every time a new version is released. As the data are already formated in the MTConnect standard by cppagent it is unnecessary to do this in the adapters, which it would be an option since all adapters where coded by the authors. If we choose to do this we will lose the facility of update the *Agent* component, since we will have to modify the cppagent at every new release and adapt it to our system architecture. Besides this modification would have to be done in all adapters, which will become a redundant work and if new devices were incorporated into the system it would be necessary to add this code for their adapters too.

### *4.3 Client*

The stage of collected data visualization could have been made in the *Agent* itself, by joining the *Client* and *Agent* components. This is possible because the *Agent* already implements an HTTP server, where the data is already available in the MTConnect$^{TM}$ standard. Therefore, it would be sufficient to stylize the data, and the user could view them through a browser. In fact, the 1.4.0.2 cppagent version already comes with stylesheets that format the data in tables, which is much better than viewing them in the *XML Output* itself. However, using this approach would deprive the project of other tools used in Web projects, which would imply dynamic pages, with more elaborate graphical components to analyze and handle the data, and would allow the connection to some database.

Therefore, we chose to deploy a Web server as a single block in the system architecture (Figure 3), keeping the *Agent* component as just a data server. This Web system was encoded in PHP (Hypertext Preprocessor) and JavaScript, using MySQL (MY SQL - Structured Query Language) as database.

Symfony was used in this Web server development, which is a widely used PHP framework in the Web developer community. Symfony is distributed under Open Source MIT license and is maintained by SensioLabs and a huge fan community and framework investors (*What is Sym-*


Figure 5: Overview of the Web System developed for LAPRAS.

*fony*, 2017). Furthermore, Symfony is designed to be fast, secure, flexible and easy to update. Thus, using this tool ensures a Web application with long term support, facilitating eventual changes over new features embedded to the platform.

For the design, the Bootstrap framework was used, which allows the creation of responsive pages, that adapt to any screen size.

The overview of the developed Web system is shown in Figure 5. This system enables the machine tools monitoring in real time, showing the complete information of each machine. Production order, current part, and total number of parts data are available in this system home screen. Production details, process data and machine tool specific data, can be accessed at the bottom of each machine or at the side navigation bar. Besides the real-time data monitoring possibility, the system allows them to be saved in standard formats, such as: .csv, .pdf, .txt, among others.

The supervisory system enables the data visualization separately, according to the machine structure itself, based on each one XML model. This data arrangement facilitates the supervisor's analysis and decision-making process. Moreover, the Web system provides the option of viewing the data in tables or graphs, and the latter can provide a more detailed analysis of specific data. For example, one Zema grinding interest parameter is the electrical power consumed by the machine during the manufacturing process, because of its close link to the grinding forces and consequently to the specific energy in grinding.

According to Figure 6, at the beginning occur one short increment on active power, being necessary to activate the primary machine's subunit. Followed by the turn on the machine's subunits hydraulic, cooling and pumping of cutting fluid system, causing increase in the required active power. Later there are two peaks caused by inrush currents of three-phase induction motors, existing in exhaustion subunit and grinding wheel, respectively. Finishing with subunit shutdown and restart operation.

Besides the active power graphic, we may generate graphics for each electric current phase, path position, path feedrate, air flow, fluid flow and sound spectrum. Those graphics are generated on real time, being an important indicator for monitoring the actual stage of grinding operation.

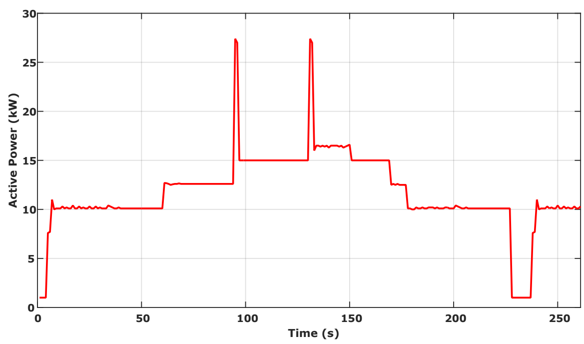


Figure 6: Total active power consumed by Zema G-800HS grinding during a manufacturing process.

## 5 Conclusions

This paper has presented a supervisory system, based on the MTConnect™ standard, applied in the industrial environment of LAPRAS, according to worldwide tendencies in IoT for the development of 4.0 Industry. A Web application was developed to allow monitoring in real time and export data in a remote way, allowing it to be accessed from anywhere with an internet connection, through a browser.

The use of Symfony for implementing the Web system has brought several advantages for the developed application, such as: extensive support; fast browsing; and flexibility, making it easier for occasional future changes. Besides, methods for processing HTML requisitions in a safe and reliable way are offered, which is one of the main concerns about development of Web applications, especially in manufacturing environments. The implementation of the supervisory system based on this platform consists of an edge regarding to the easiness of keeping the Web system up-to-date with safety requirements that shall be implemented by the Symfony community. The possibility of creating graphs or tables is also an advantage, because it provides a graphical analysis of the project in real time, helping the analysis of the manufacturing processes. It is important to highlight that Symfony platform was developed with the aim of being a freeware framework which provides interoperability between several applications, thus harmonizing with the MTConnect™ Institute ideology itself.

Finally, it can be evidenced that IoT applications benefit the 4.0 Industry, especially in the high-performance measures, due to the high aggregate value.

## References

da Silva, R. M., Filho, D. J. S. and Miyagi, P. E. (2015). Modelagem de sistema de controle da indústria 4.0 baseada em holon, agente, rede de petri e arquitetura orientada a serviços, *XII Simpósio Brasileiro de Automação Inteligente - SBAI 2015*, pp. 1218–1223.

Dutra, D. S. and Silva, J. R. (2016). From smart products to product-service systems: a service engineering perspective from the industry 4.0, *XXI Congresso Brasileiro de Automática - CBA2016*, pp. 1922–1927.

Evans, D. (2011). The internet of things - how the next evolution of the internet is changing everything, *CISCO white paper* **1**(2011): 1–11.

Lei, P., Zheng, L., Li, C. and Li, X. (2016). Mtconnect enabled interoperable monitoring system for finish machining assembly interfaces of large-scale components, *Proceedings of the 9th International Conference on Digital Enterprise Technology - Intelligent Manufacturing in the Knowledge Economy Era*, Vol. 56, Elsevier, pp. 378–383.

Mattern, F. and Floerkemeier, C. (2010). From the internet of computers to the internet of things, *From active data management to event-based systems and more*, Springer, pp. 242–259.

Mori, M., Fujishima, M., Komatsu, M., Zhao, B. and Liu, Y. (2008). Development of remote monitoring and maintenance system for machine tools, *CIRP Annals 2007 - Manufacturing Technology*, Vol. 57, Elsevier, pp. 433–436.

Pisching, M. A., Junqueira, F., Santos Filho, D. and Miyagi, P. E. (2016). Architecture for industry 4.0: related issues about modelling and simulation, *XXI Congresso Brasileiro de Automática - CBA2016*, pp. 545–550.

Sakurai, L. H., Pisching, M. A., Pessoa, M. A. O., Junqueira, F. and Miyagi, P. E. (2016). Aplicação da internet das coisas em sistemas produtivos com foco na indústria 4.0, *XXI Congresso Brasileiro de Automática - CBA2016*, pp. 2683–2688.

Santos, A., Macedo, D., Silva, I., Neto, A. and Guedes, L. A. (2013). Ferramenta para gerenciamento de redes industriais wirelesshart, *XI Simpósio Brasileiro de Automação Inteligente - SBAI 2013*.

Santucci, G. (2010). The internet of things: Between the revolution of the internet and the metamorphosis of objects, *Vision and Challenges for Realising the Internet of Things* pp. 11–24.

Sobel, W. (2009a). Mtconnect standard part 2 - components and data items, [Online]. Available: *www.mtconnect.org*. Accessed: 16 Feb. 2017.

Sobel, W. (2009b). Releases - mtconnect/cppagent, [Online]. Available: *https://github.com/mtconnect/cppagent/releases*. Accessed: 16 Feb. 2017.

Sobral, J. V., Rabelo, R., Filho, J. L., Souza, R., Sousa, N., Araújo, H. and Filho, R. H. (2015). Uma estrutura baseada em inteligência computacional para melhorar o desempenho de aplicações iot, *XII Simpósio Brasileiro de Automação Inteligente - SBAI 2015*, pp. 975–980.

Torrisi, N. M. (2011). Monitoring services for industrial applications, *IEEE Industrial Electronics Magazine* **5**(1): 49–60.

Vijayaraghavan, A., Sobel, W., Fox, A., Dornfeld, D. and Warndorf, P. (2008). Improving machine tool interoperability using standardized interface protocols: Mtconnect, *Proceedings of 2008 International Symposium on Flexible Automation - 2008 ISFA*, pp. 1–6.

*What is Symfony* (2017). [Online]. Available: *https://symfony.com/*. Accessed: 16 Feb. 2017.