Smart Lab: an application of Industry 4.0 design principles to calibration laboratories

Gean Marcos Geronymo
Instituto Nacional de Metrologia, Qualidade e Tecnologia, Inmetro, Rio de Janeiro, Brazil
E-mail: gmgeronymo@inmetro.gov.br

Abstract. The integration of the Internet of Things and Cyber Physical Systems, in the context of Industry 4.0, leads to the Smart Factory concept. In this paper we propose the extension of this concept to calibration laboratories: the Smart Lab. We present the four design principles of Industry 4.0 and show how they can be applied to the context of calibration laboratories. Finally, the development of an environmental conditions monitoring system based on the Industry 4.0 principles is presented as a case study.

Keywords: Industry 4.0, Digitalization, Metrology 4.0, Internet of Things, Cyber Physical Systems.

1. Introduction
The convergence of industrial production, information and communication technologies has built the foundation for the fourth industrial revolution, which is about to take place right now [1]. The term “Industry 4.0” or “Industrie 4.0” (the later notation is commonly used in publications from the German-speaking area) originates from a project in the high tech strategy of the German government, promoting the digitization of industry [1]. The “Industrie 4.0 Working Group” name three key components of Industry 4.0: the Internet of Things and Services (IoT), Cyber-Physical Systems (CPS) and Smart Factories [2]. The Internet of Things and Services (IoT) allows a variety of “things” and “objects”, such as RFID tags, sensors, actuators and mobile phones to interact with each other and cooperate with their neighbours to reach common goals [3]. Cyber-physical systems (CPS) are “smart” systems that synthesizes the fusion between the physical and digital (cyber) worlds [2][4]. A CPS generally involves sensing, computation and actuation [5] and can be network compatible [1]. Within this concept, with CPS physical objects and processes are replicated in the digital world [1][4]. The integration of IoT and network-compatible CPS leads to the Smart Factory, a key feature of Industry 4.0 [1][2]. In a smart factory, CPS monitor and control physical processes and make decentralized decisions. Using the IoT, CPS communicate and cooperate with each other and with humans, and can be assisted by technologies such as cloud computing, big data analysis [6] and machine learning [7]. In this work, we propose the extension of the Smart Factory concept to calibration laboratories: the Smart Lab.

2. Design Principles
Based on a quantitative text analysis and a qualitative literature review, Hermann, Pentek and Otto have identified that there are four design principles in Industry 4.0: interconnection, information transparency, decentralized decisions and technical assistance [1].
2.1. Interconnection
There are three types of collaboration within the IoT concept: human-human collaboration, human-machine collaboration and machine-machine collaboration. For connecting machines, devices, sensors and people with each other, the use of common communication standards is essential [1]. In the context of the Smart Lab, we propose the use of an open application programming interface (API) following the Representational State Transfer (REST) principles [8]. Using a REST API, digitally enhanced objects (smart things) are accessible using Uniform Resource Identifiers (URIs) that can be exchanged, referenced and bookmarked [9]. Any laboratory equipment that has some communication interface, such as IEEE 488 or RS 232 can turn into a “Smart Thing” using low cost development boards such as Raspberry Pi [10] or Arduino [11]. In this scheme, equipment and software of different vendors (or even in-house developed) can be interconnected.

2.2. Information transparency
Within the concept of Cyber Physical Systems (CPS), a digital copy of the physical world is created through linking sensor data with digitalized plant models. This enables the development of context-aware systems that accomplish their tasks based on information coming from the digital and the physical world [1]. In Smart Lab’s context, information transparency can be achieved with the interconnection of all laboratory equipment, measurement systems, calibrations schedule software and environmental monitoring systems, for example, by using a common API.

2.3. Decentralized decisions
The combination of interconnection and information transparency leads to the possibility of implement decentralized decisions, the key feature of Industry 4.0. In the context of the Smart Lab, automated measurement software can receive real time information about temperature and humidity and decide if a measured point should be discarded, for example. The measurement software can access the calibration scheduling and decide when to turn on or turn off the equipment, optimizing the power consumption during weekends and assuring that the system will be ready to use when needed.

2.4. Technical assistance
In the Industry 4.0 paradigm, the main role of humans ceases to be just that of a machine operator and becomes that of a strategic decision maker and problem solver [1]. The scenario where CPS are interconnected in complex networks and make decentralized decisions demands the development of assistance systems to support humans. In Smart Lab’s context, assistance systems can guide the technician during the whole calibration process. When the DUT arrives at the laboratory, it receives a RFID tag and is registered in the integrated system, using an online app accessed via a mobile device (like a smartphone or a tablet). The integrated system provides quick and easy access to the laboratory documentation involved with each step of the DUT inside the laboratory. When the DUT is on the test bench, all of its information already stored can be recovered by the measurement software, which is interconnected to the network. The measurement software can show connection diagrams, test procedures, calibration history data and all the relevant documentation to assist the technician. All relevant data obtained during the calibration, including laboratory’s environmental conditions, are stored and unique identified in a database, for easy access in the future.

3. Case study: environmental conditions monitoring
A low cost system for monitoring environmental conditions is being developed by the Laboratory of Metrology in Electrical Standardization (Lampe), at Inmetro, following the Industry 4.0 design principles. The system is being designed to be the first step towards the Smart Lab philosophy, with an easy to use REST API that can be integrated in other software projects.
3.1. System architecture

A block diagram of the system architecture is presented in Figure 1. The core of the system is a REST API [8]. We built our system using the existing corporate network available in the laboratory. The server module is based on a Linux Web Server, with Apache (HTTP server), PHP (scripting language) and PostgreSQL (database management system). The server can be a local physical or virtual machine, on the same LAN, or a remote server, accessible through the Internet. When a user accesses the Graphical User Interface (Web GUI) using a web browser, a simple user interface is presented, with charts of time series data, for both temperature and humidity, for each sensor. New sensors can be easily registered using the Web GUI.

The Cyber Physical Systems (CPS) can be built by integrating temperature and humidity sensors with any hardware platform capable of connecting to the corporate network, via Ethernet or Wi-Fi, and send HTTP POST requests to the server. Some low cost options are Arduino with an Ethernet or Wi-Fi module, ESP8266, ESP32 and Raspberry Pi. Our first prototypes were built using a Raspberry Pi connected to a low cost temperature and humidity sensor. We plan to build another prototype connecting another Raspberry Pi to a commercial thermo-hygrometer with RS-232 interface, to illustrate the concept of converting conventional equipment into a CPS using the IoT.

3.2. REST API

The two most common architectures for IoT servers today are MQTT and REST. MQTT [12] is a machine-to-machine IoT connectivity protocol, designed as an extremely lightweight publish/subscribe messaging transport. REST [8], an acronym for Representational State Transfer, is an architectural style for providing standards between computer systems on a network, making it easier for systems to communicate with each other. MQTT is specifically designed for IoT, and has many advantages for applications that need instant response and lower power consumption, for example. On the other hand, REST is easier to be implemented using the existing network infrastructure (corporate WiFi network that only allows HTTP traffic through TCP ports 80 and 443, for example). For this project, instant response and low battery consumption are not the main requisites, so we decided to go with the REST architecture, using the already available corporate WiFi network, for simplicity. REST APIs use Uniform Resources Identifiers (URIs) to address resources. The four basic data operations (create, read, update...
Table 1. Data operations.

| Operation | HTTP Method | SQL (database) |
|-----------|-------------|----------------|
| Create    | POST        | INSERT         |
| Read      | GET         | SELECT         |
| Update    | PUT         | UPDATE         |
| Delete    | DELETE      | DELETE         |

http://server/termhigr/datalogger/id/sensor1/quantity/temperature/format/html

Server address Controller Resource Parameters Format

Figure 2. REST API URI pattern.

and delete) are mapped to HTTP methods (POST, GET, PUT and DELETE). The REST API interacts with the database translating the data operations to SQL statements. The relations between the four basic data operations, HTTP methods and SQL statements are presented in Table 1.

The proposed REST API is based on the CodeIgniter Rest Server project [13]. The URI pattern of the API is presented in Figure 2. For this particular example of an environmental conditions monitor, the resource is named “datalogger”. For example, if we want to retrieve temperature data from a sensor named “sensor1”, in html format, we should do a HTTP GET request to the URI presented in Figure 2. The parameters part of the URI, “id/sensor1/quantity/temperature” is translated to the variables “id=sensor1” and “quantity=temperature”. The last part, “format/html” tells the API that we want data presented in html format, for easy viewing in a web browser (the API support other formats such as JSON and XML, which can be useful for some applications). If we want to get humidity data, we must set “quantity/humidity” on the URI, for example.

The developed API is open for HTTP GET requests, so any application can read data without authentication. To send data to the API (via HTTP POST requests), a basic authorization scheme was implemented: each sensor must be registered to get a unique API key, a forty character random alphanumeric string. Then, this key must be inserted in the sensor’s configuration file. During normal operation, each sensor sends periodically a HTTP POST request to the URI http://server/termhigr/datalogger, with the API key in the HTTP headers and the variables date, temperature and humidity. The API checks if the received data comes from a registered sensor, and stores the data in the database.

3.3. Cyber Physical Systems

The first client prototype was built using a Raspberry Pi 3B [10] as hardware platform. The Raspberry Pi 3B is a full computer on a board of the size of a credit card, with a CPU, RAM memory, video interface with HDMI output, Ethernet interface, SD card interface for running the operational system and data storage, USB interface, WiFi network and Bluetooth. As a full computer, a Raspberry Pi must run an operational system. We developed our software to run on the lightweight version of the official operating system of the Raspberry Pi Foundation, Raspian Lite. The Raspberry Pi has a General Purpose Input Output (GPIO) interface with hardware pins that can be easily accessed in software to control external sensors and peripherals.
3.3.1. **Hardware**  For our first prototypes we used a stock Raspberry Pi 3B board with the addition of a real time clock (RTC) module and a 16x2 LCD display. A temperature and humidity sensor (DHT22) was connected to the GPIO pins. For future prototypes we plan to add a RS232 interface to connect to commercial thermos-hygrometers.

3.3.2. **Software** The software was written using Python language, with the help of the pigpio library [14]. The software runs as a system service, reading the data from the sensor, storing it locally in a text-file log and sending to the REST API via a HTTP POST request periodically. The adjustable parameters are configurable through a text-file, datalogger.ini, accessible in the boot partition of the Raspberry Pi’s SD card. In this text-file we can configure the interval between readings, the base URI of the REST API, the API key and enable or disable the 16x2 LCD display.

3.4. **Web Graphical User Interface**
The Web GUI is presented in Figure 3. A simple interface was developed, showing a chart of temperature data and a chart for humidity data. The web application retrieves the data from the REST API in JSON format, and a JavaScript application draws the chart.

3.5. **External Applications**
The REST API can provide data to virtually any application or programming language that can perform a HTTP GET request. Measurement software built using LabVIEW, for example, can easily send data to the API via HTTP POST requests. Using this common API, every instrument and software of the laboratory can be interconnected.

![Web GUI](image-url)
4. Conclusions
We presented the Smart Lab concept, aligned with the Industry 4.0 principles. A simple REST API used for an environmental conditions monitoring system was developed following the Smart Lab concept. This solution can be easily and readily implemented with existing technology, using low cost hardware and open source software. The main purpose of the API is to be extended and integrated in other projects, in order to interconnect all software and equipment in the laboratory and provide quick and easy access to all the data, paving the way for the implementation of solutions such as decentralized decision-making.

References
[1] M. Hermann, T. Pentek and B. Otto, “Design Principles for Industrie 4.0 Scenarios”, 2016 49th Hawaii International Conference on System Sciences (HICSS), Koloa, HI, 2016, pp. 3928-3937.
[2] H. Kagermann, W. Wahlster, and J. Helbig, “Recommendations for implementing the strategic initiative Industrie 4.0: Final report of the Industrie 4.0 Working Group”, Frankfurt, 2013.
[3] L. Atzori, A. Iera, G. Morabito, “The Internet of Things: A survey”, Computer Networks, Volume 54, Issue 15, 2010, Pages 2787-2805, ISSN 1389-1286.
[4] “Agenda brasileira para a Indústria 4.0” [Online]. Available: http://industria40.gov.br/
[5] E. R. Gifford, C. Greer, D. A. Wollman, and M. J. Burns, “Framework for cyber-physical systems: volume 1, overview,” National Institute of Standards and Technology, Jun. 2017.
[6] Snijders, C.; Matzat, U.; Reips, U.-D. (2012). “‘Big Data’: Big gaps of knowledge in the field of Internet”. International Journal of Internet Science. 7: 1–5.
[7] Bishop, C. (2006). Pattern recognition and machine learning. New York: Springer.
[8] R. T. Fielding and R. N. Taylor, “Principled design of the modern web architecture,” ACM Trans. Internet Technol., pp. 115–150, May 2002.
[9] M. Collina, G. E. Corazza and A. Vanelli-Coralli, “Introducing the QEST broker: Scaling the IoT by bridging MQTT and REST,” 2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications - (PIMRC), Sydney, NSW, 2012, pp. 36-41.
[10] “Raspberry Pi Foundation”. [Online]. Available: https://www.raspberrypi.org
[11] “Arduino”. [Online]. Available: https://www.arduino.cc
[12] “MQ Telemetry Transport,” April 2012. [Online]. Available: http://mqtt.org
[13] CodeIgniter Rest Server. Available: https://github.com/chriskacerguis/codeigniter-restserver
[14] Piggio. Available: https://abyz.me.uk/rpi/piggio/