IIoT vs. SCADA

by Dr. Burkhard C. Heisen

Prerequisites

As you are reading this sentence, you seem to have a general interest in the topic of industry digitization. That's all you need, lean back and simply enjoy this article.

Introduction

So, we want to compare two very abstract things, a "classical" **Supervisory Control and Data Acquisition (SCADA)** system against a "modern" **Industrial Internet of Things (IIoT)** system. Let's structure the comparison by looking at the problems that each system is going to solve or in other words, the benefits that each system is intended to provide to the user(s).
To further structure the comparison we define the locations where the two systems are acting in:

1. Things
2. Gateway
3. Cloud
4. Mobile

The Things layer describes the location of the physical equipment the systems are going to interact with. Typically, Things on that level are not directly connected to the Internet, but typically to one or more local networks. The Gateway describes the location where information from the Things layer is aggregated. It furthermore is the location that defines the transition between the local network(s) and the public Internet.
The Cloud layer is located in the Internet and describes a set of servers that are dealing with the data as provided by one or more Gateways. The Mobile layer finally describes the location of a human end-user. Irrespective of the geographical location, the user will interact with the data as provided by the Cloud services always having direct access to the Internet.
You already notice that the topic is large and complex. That is why the comparison will be sliced into digestible pieces. Today's article will start with a focus on Things and how they are represented.

Representing Things

At the lowest level we want to interact with **Things**, which can be – as the name suggests – quite anything. Fortunately, we are talking about IIoT and SCADA so let's immediately reduce the scope and look at Things from an industrial perspective.

In an industrial context (especially in SCADA lingo) things are often called **Devices**, a general name for motors, pumps, grippers, RFID readers, cameras, sensors and so on. In other words, a Device can reflect any piece of (digitized) hardware of any complexity.
For structuring the further discussion let's divide Devices into two kinds: **actors** and **sensors**, with actors being Devices that are „doing something", and sensors being Devices that are „reading something".
Having this defined, one could believe that the entire **shop-floor** (a company's total set of Devices) can be categorized this way. However, this quickly turns out to be tricky as – depending on the perspective – Devices are complex entities most often including several sensors and actors at the same time.
In its extreme, already a valve is a complex Device composed of typically one actor (valve motion) and two sensors (sensing open and closed position).

View on things

The trickiest thing for any software system is to provide interactions with Devices on any abstraction level and perspective. Depending on the user, the definition of what a Device is turns out to be completely different: the PLC programmer may see a single analog output as a Device whereas the control room operator may perceive an entire plant-subsystem as a Device (to formulate an extreme case).

Problem to solve

Provide a system that can interact with actors and sensors at the most atomic level but at the same time provide arbitrary logical layers on top of the underlying physical realities. Those logical abstractions must fit the different end-user's requirements regarding read-out and control and typically vary and overlap in abstraction-level (vertically) as well as in composition-level (horizontally).

Technical solution

We will call the logical representation of a Device a **Digital Twin**. Digital Twins then form vertices in a tree, with the root vertex expressing the most abstract view of a Device. Lower levels of abstractions are reached by traversing the tree downwards in direction of the leaves each vertex representing a Sub-Device its parent is composed of. The leaves finally represent the lowest level of abstraction and may e.g. reflect a single physical analog input. In SCADA systems such most atomic entities are called **Process Variables (PVs)**. The number of children per parent indicates the horizontal complexity of the respective parent. Finally the shop-floor is represented as a forest of the above described trees indicating a disjoint union of all entities.

Side Effect

In such a solution writing to or reading from a Digital Twin may result in an interaction with a physical or logical property, as Digital Twins may act on each other (as described above) or directly on physical entities. This poses an (graph-)algorithmic challenge of correctly identifying all Digital Twins affected by a failure or complete loss of connection to a physical property and for the entire access control layer that must give different users different permissions on the available properties.

SCADA vs IIoT

SCADA systems have a very strong focus on creating logical representations of Devices at a very early stage. A full set of a logical Device hierarchy is established and visualized to the user already on-premise. Most of the data hence never hit the cloud (i.e. the Internet) but is used to immediately feedback into the system or to the operator. Although distributed, SCADA systems aggregate data within (private) local networks and hence have less focus on Internet security or web-standards for communication.
IIoT systems, in contrast, try to make no assumptions on where the representation from raw information to logical Devices is happening. Furthermore, they do not assume that data is used solely for observing and controlling a plant but for completely – yet unknown – use cases, such as integration into other administrative layers of an enterprise. Consequently, communication is prepared to follow most recent standards of security and transport protocols from the beginning on and much effort is undertaken to have a very flexible, albeit semantically clear description of the raw Device data to be consumed by any other higher abstraction layer.

Final words

Hopefully, you could already grasp some fundamental differences between the two systems. In the following articles we will sharpen these differences and have a look at further locations of activity (Gateway, Cloud and Mobile).
In a final article we will demonstrate how the Cybus Connectware implements all the discussed requirements for a modern IIoT system and how you can use it for your special use-case.