SPOTLIGHT *series*

Critical OS Platform Strategies for Industrial IoT Systems

DARON UNDERWOOD

When the Internet of Things (IoT) started to take hold, it was a nebulous term that everyone wanted to use for everything. Soon, however, companies in the embedded space started to put parameters around the idea specific to their industries, giving rise to initiatives like Industry 4.0, the Industrial Internet of Things (IIoT), China 2025, and more. At their core, these paradigms focus on the same goal: how to use the power of connectivity and communications to empower embedded/industrial systems with information and analytics to increase efficiency and control at all levels of production and manufacturing.

KINGSTAR

All of which was going well – until the Meltdown and Spectre vulnerabilities reared their ugly heads. Like a shark fin rising out of the water in the movies, these threats destroyed the IIoT industry's false sense of security about connected devices and forced organizations to reevaluate their approach. The Meltdown/Spectre issue is particularly devastating because it is a hardware design flaw, not a software problem that can be relatively easily corrected. As such, it affects essentially every major platform in use today, from ARM to Intel architecture, going back a decade or more, and of course, the problem is exacerbated by the inherent connectivity of IIoT systems and their more transient states compared to the static systems of the past.

In this environment of uncertainty, it's even more critical to create embedded systems that are not only as secure as possible, but also flexible, scalable, maintainable, and cost-effective. The following strategies for OS selection, system deployment, configuration, updates, and security can help protect IIoT systems while making them more powerful and productive. It is imperative to select an OS that offers all of these capabilities to reduce the internal resources required to support the platform at development and throughout the lifecycle of the product.

Operating System (OS) Selection

The right operating system plays a crucial role in the success of an IIoT system. Based on the criteria above and its commercial availability, Windows 10 is usually the best bet for IIoT systems. While developers are free to choose any flavor of Windows 10, from retail versions such as home or pro to enterprise and IoT versions, IoT versions come with significant advantages. Perhaps most importantly, the IoT Enterprise version is on the Long-Term Servicing Channel (LTSC) like the standard LTSC version, and as such is less volatile and guaranteed to be supported for much longer.

Why is LTSC so important for IIoT in general and embedded systems specifically? The Long-Term Servicing Channel is the service channel that houses both the Enterprise products as well as the IoT products, including both IoT Core and IoT Enterprise. IoT Core is targeted at a specific-purposed device and is built to the specific device hardware platform, such as the Qualcomm Dragonboard 410c which is an ARM-based platform. IoT Enterprise is like Windows 10 Enterprise (only running on x86/x64 platforms) plus or minus a few features designed to enable embedded computing and remove anything irrelevant to IoT use cases. For instance, IOTE does not include the Microsoft Store applications.

The Windows 10 IoT Core offerings are also starting to make inroads into IIoT/embedded products markets. In fact, Microsoft is "doubling down" on Windows 10 IoT with support for new hardware platforms. In addition, the company recently announced that the upcoming release of both Windows 10 IoT Core and Enterprise will be supported for 10 years. This is extremely important for IIoT and especially embedded devices, which are generally only updated in rare cases once deployed. Keep in mind that Microsoft only releases a new LTSC version of products every 2-3 years, unlike the retail and business channel products which have a biannual update cadence.

With Microsoft making these long-term commitments to the IoT/IIoT community, it is virtually a "no-brainer" to seriously consider Windows 10 for new products.

System Deployment and Configuration

Most IIoT systems can benefit enormously from a strategic combination of cloud and local edge computing. This approach to networking is literally making the functional application local system agnostic.

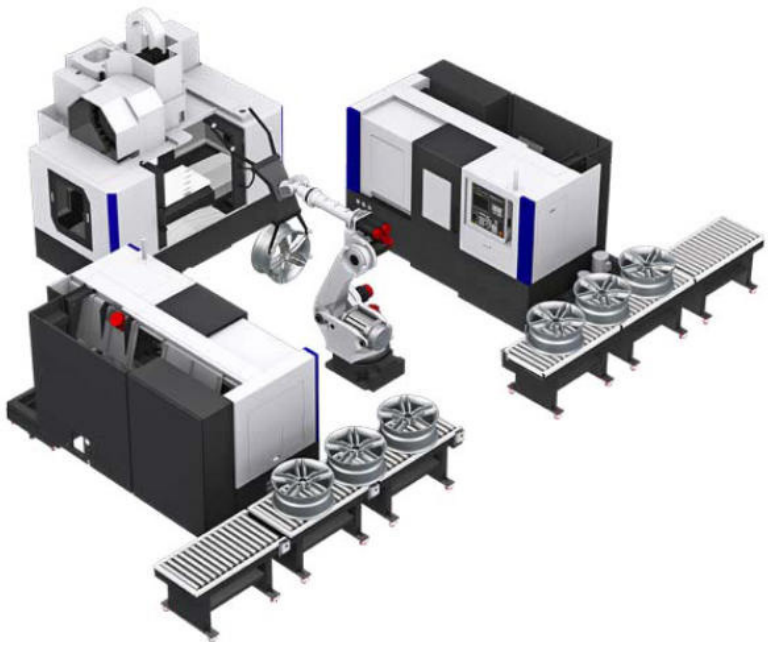
To make the point, let's consider a Java application. Java is essentially a Virtual Machine-based runtime, allowing a developer to write an application once and deploy that application on any system that can provide the required resources and has an executable Java runtime. This has been a very powerful way for developers to capitalize on their codebases and extend the number of supported systems with no-to-minimal additional efforts.

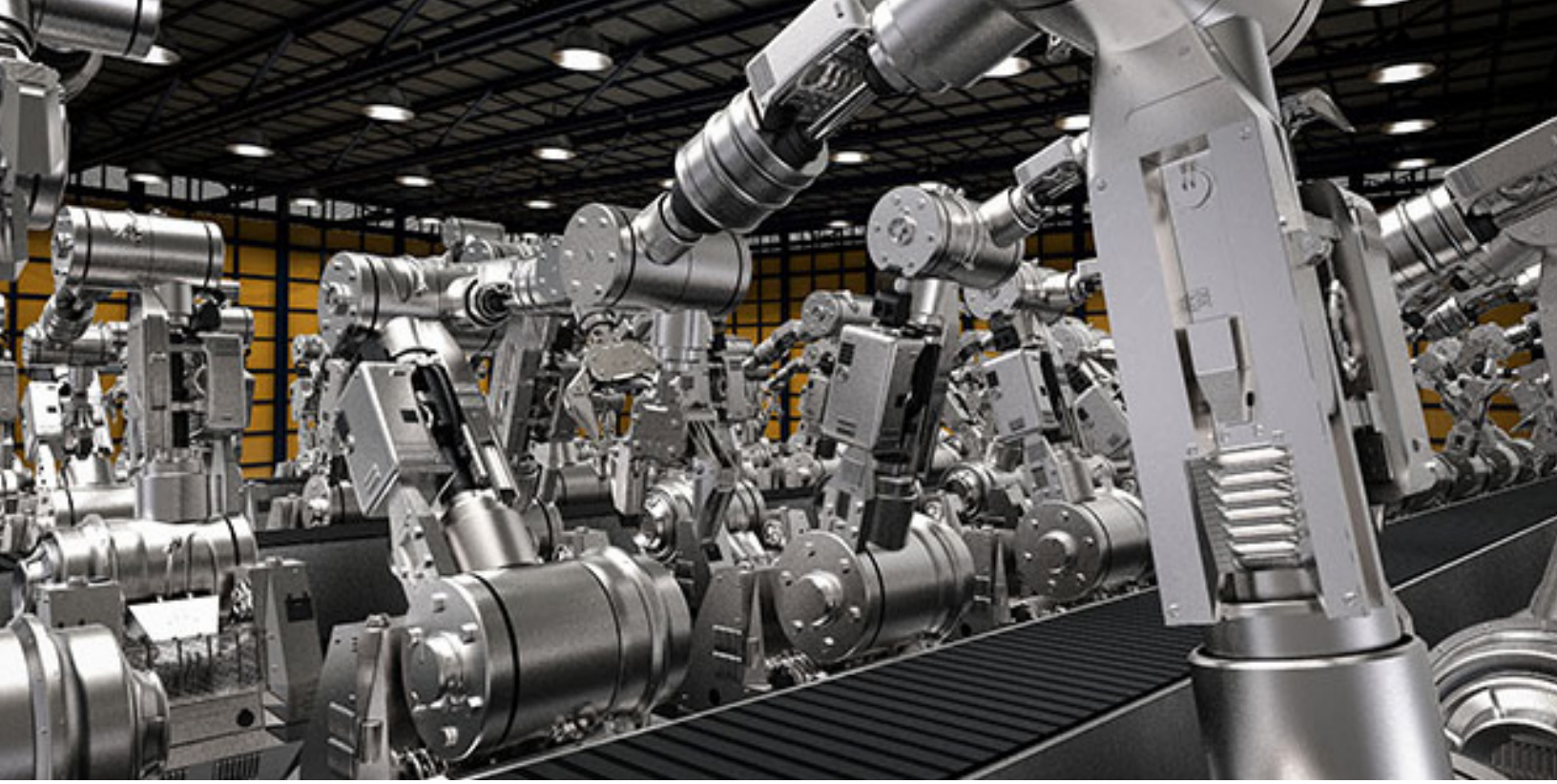
The power of this method is not only the ability to run on many different systems, but also use the network connectivity to deploy the application to the system. Also, to take it a step further, the Java runtime itself can be downloaded and installed on the system on which the user wants to run the application.

Now let's think about this same type of scenario in terms of IIoT/Industry 4.0. If we think of the cloud as a hub for our solution as a whole, part of that solution is the ability to deploy and configure systems at the local manufacturing site to support the manufacturing and processing needs of our solution. In this situation, we can have a system at the local site that is connected via some IO/fieldbus to the manufacturing equipment, or perhaps even multiple systems connected to the same equipment. If we consider the system-agnostic view explained above, we should be able to deploy to either of these systems to provision it to execute the required tasks to use the connected equipment to produce the expected outcome.

Let's look at an example to visualize this point.

Consider a manufacturing cell where there are 3 CNC milling machines with a robot arm positioned between them.





In the primary operation, the cell uses the central machine to do a relatively fast rough milling of a part. When complete, the part is picked up by the robot and transferred to one of the other machines for a final finishing milling that takes considerably longer. The cell is optimized for using the two finishing mills to improve throughput. In the event that the roughing mill goes down, the cloud management system can detect the issue, order maintenance, and most importantly, reprogram the entire cell to continue in a two mill operation. It does this by provisioning one of the finishing mills to now be a roughing mill and reprogramming the robot to support this new configuration. Although the throughput is decreased, it continues until the original roughing mill is brought back online and the cell is reverted to its original configuration.

This type of operational model has interesting and important ramifications. When we reduce the specifications for the hardware down to just a few requirements, like having the appropriate performance and resources to support an RTOS for example, the software solution and network/cloud become the real stars. We can borrow an analogy from another industry to illustrate this point: fleet management for deliveries. If I'm in charge of fleet management, I don't care what make and model the vehicles are, I simply need to know the capabilities of the vehicles that I am managing and be able to deploy them when and where they are needed.

This is exactly the type of flexibility and scalability that new technologies are bringing to IIoT/Industry 4.0 solutions today. The combination of Windows 10 IoT, Azure IoT Hub, and Azure Edge creates a powerful set of tools for developing new innovative products and processes.

Field Updates

Now that we have selected a good long-term OS product for our solution, we need to consider the ramifications of that selection on service and maintenance. In the past, field updates of a deployed device were rare, and in many cases were not done unless absolutely necessary. This was especially true when the entire platform was required to go through a certification process. The prevailing wisdom fit the old mantra: if it's not broke, don't fix it.

However, even if the issue that an update is addressing is not causing direct or foreseeable problems with your application's functionality, the update may still be critically important. With the Meltdown/Spectre vulnerabilities, for instance, the real issue was with device information security, not the specific device application function. In this case, the problem was a manifestation of the actual hardware design, but everyone still wanted to update the firmware and OS to address the issues. This response was in stark contrast to developers' typical attempts to keep the systems as stable as and static possible; in many cases, locking down the systems from getting automatic updates.

A cloud deployment solution like that mentioned above solves these challenges. You could not only control your application updates, but also broker when your deployed systems could update their firmware and OS as well as any other subsystem software, such as a real-time extension with IntervalZero's RTX64. Even better, this could all be managed and initiated with the push of a button from your cloud-based consoles - on your schedule and at your convenience. This is not a pipe dream. It's all very possible with today's technology, and it's very clear how much time and money could be saved by not having to run updates in the field.

Security

Security is obviously of utmost importance to systems that are connected to a cloud-based component. While Meltdown/Spectre is mostly an IT-focused issue, a threat such as the Stuxnet computer worm, which actually affected the processing in the facilities, was very much an operational technologies (OT) attack. Monitoring and securing both of these types of attack vectors is clearly important individually, but in today's climate, it's even more crucial to consider them together. An IT attack can be very costly to a company, but an OT attack is just as hazardous – and could even be fatal to individuals.

Every component of a product has its own unique security risks. With Windows 10 IoT, the management of that risk is largely in the hands of Microsoft, so your company's engineering/IT teams are relieved from spending too much effort. Once a threat is identified and there is a fix for it, the update can easily be scheduled into the cloud-based update process as described above.

Conclusion

As embedded systems move to the realm of IIoT, organizations and developers are realizing that the cloud is only one piece of the puzzle; the symbiosis of the cloud and the edge devices is the partnership that is already changing the world. There is no doubt that the device is still key, but the 1-to-1 relationship of the device-to-hardware is changing. The device is no longer tied to a specific hardware platform, but is now defined by the combination of the software running on a more agnostic hardware system. This approach is already powerful, and will only become more so in the near future, until the hardware almost fades away and the software fully becomes the device.

With the strategies above, embedded product companies and developers can embrace this paradigm while keeping a firm grip on reality: the device is a key and powerful partner in the new cloud-centric world. With this new paradigm, we need a way to mitigate the risks and costs of the operating system for the IIoT devices. To achieve this, you need to select an OS that is constantly monitored and updated in near real-time. This is exactly what Windows 10 brings to the table. The Windows-as-a-Service (WaaS) model meets this challenge, while giving developers options on how and when the system is updated.

The OS should now be treated as a commodity that you can purchase for use and forget about. No longer should IIoT product companies be burdened to use internal resources and costs associated with supporting the OS. Instead, the combination of Windows 10 and edge devices and cloud services like Azure redirects resources back to the things that matter and gives developers an unparalleled set of tools to achieve real innovation in their services, products, and processes.