MatrikonOPC

Keys To Developing an Embedded UA Server

Liam Power, Embedded OPC UA Subject Matter Expert
Darek Kominek, P.Eng, MatrikonOPC Marketing Manager
Edmonton, Alberta, Canada - 2013

Executive Summary

Strong demand for improved access to shop-floor data is driving control automation vendors to make their devices as easy to integrate as possible. Embedded OPC UA presents tremendous opportunities for device vendors to make their products stand out with native open data connectivity that is more secure, easier to integrate in multi-vendor environments, and opens door to new markets due to the widespread use of OPC. This paper discusses the key challenges device vendors must overcome to successfully harness this breakthrough but complex technology.

SHORT OPC BACKGROUND

OPC, the world's most popular standard for open automation data connectivity, is really a series of specifications that define how automation data can be shared between controllers, sensors, applications, and virtually all networked devices. Of these specifications, the most common one is OPC Data Access (DA) which deals with real-time data transfer followed by OPC Historical Data Access (OPC HDA) and OPC Alarms & Events (OPC A&E) dealing with the transfer of historical and alarm and event data respectively.

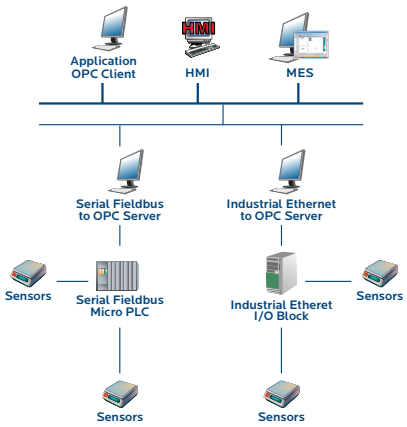


Figure 1: PC-only based connectivity.

First released in 1996, the original OPC standard relied on proprietary Microsoft Windows technology (COM/DCOM) to provide application level security and data transport between OPC clients and OPC servers. Relying on the Microsoft technology simplified OPC client/server development by having the Windows operating system take care of all the low level details but came at a price: OPC technology only ran on Windows based systems and provided very little visibility and control over communication parameters (ex. timeout durations). The first limitation effectively shut out the chance to allow most devices, controllers, and non-Windows based systems in general from supporting OPC natively, on board. The second limitation created a lot of headaches between IT and Operations departments due to DCOM related issues.

Limitations of traditional PC-only based connectivity

While the first generation of OPC Servers brought open connectivity to control automation, the OPC Servers had to run on Windows based PCs and often relied on device drivers (also running on PCs) to communicate with each device on the shop floor. This typically meant that multiple physical layers and protocols had to be set up and maintained for data connectivity to function. Physical layers such as RS485, CAN, and Industrial Ethernet are some common examples with protocols like Modbus, PROFIBUS, and DeviceNet sitting atop them. The OPC Servers exposed the process data as OPC tags or data points. This was incredibly successful but is not without its limitations.

One such limitation is that typically an OPC tag must be manually created by a commissioning engineer. This is necessary because the protocol used to talk to the device is not OPC and we have to figure out how to map the process variable from the device protocol to an OPC tag. This commissioning effort is labour intensive and like any manual operation is prone to errors.

Another limitation is that many of the device protocols in use are insecure. MODBUS TCP for example is in widespread use on Industrial Ethernet networks and is completely insecure. As well as eavesdropping on process activity, for devices incorporating outputs, this potentially exposes the process to unauthorized actuation where the network boundary has been compromised by an attacker.

In most cases aggregating device communications through a dedicated OPC Server makes sense. It provides an efficient network architecture while facilitating logging of historical data at a central point. There are some cases however where it does not. What if you just want to get a single tag into a visualization client from a field device in a standalone installation? In many cases Windows based OPC Servers are introduced into a plant for this purpose where they are complete overkill for the simple need that exists.

"The OPC servers exposed the process data as OPC tags or data points. This was incredibly successful but is not without its limitations. One such limitation is that typically an OPC tag must be manually created by a commissioning engineer. This commissioning effort is labor intensive and prone to errors. Another limitation is that many of the device protocols in use are insecure."

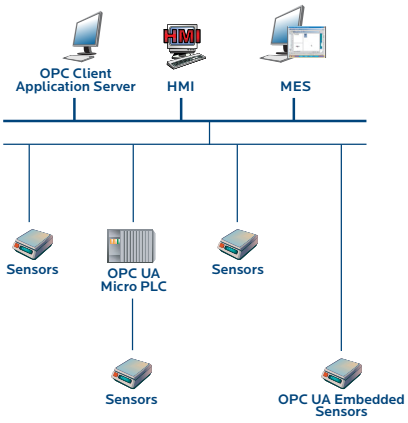


Figure 2: Embedded OPC UA connectivity

INTRODUCING OPC UA

OPC UA (Unified Architecture) is the OPC Foundation's next generation of OPC standards. Learning from past successes and challenges as well as the evolved needs of modern control automation environments the OPC Foundation stayed true to the original vendor-neutral data connectivity philosophy but redefined and upgraded the flexibility, power, and security of the entire OPC model. Rapidly gaining traction in many industrial spaces outside of process control (ex. building automation), OPC UA was designed from the ground up to be platform and OS independent – enabling seamless communication between all components of an automation system and the enterprise.

Thanks to OPC UA's flexibility, OPC UA applications can be developed for non Windows platforms such as Linux and for embedded systems running an RTOS (Real Time Operating System) or even "bare metal" environments where there is no operating system. OPC UA can even be embedded in microcontrollers costing less than $5.

THE NEXT STEP: EMBEDDED OPC UA

Why embed an OPC UA server in a device? In a word: simplicity. For end users, the holy grail of data connectivity is finding the easiest, most efficient and cost effective way to access their data when and where they need it – all without adding additional PCs and performing additional configurations and maintenance. Having OPC UA run natively (embedded) right on the devices themselves makes good sense.

Advantages of going Embedded

If you embedded an OPC Server directly in the device you can create an optimal solution for almost every application. Because the OPC tags are natively present in the device, the commissioning engineer simply has to point and click to choose the OPC tags he wants to visualize or log. Because the network is using OPC from end to end there is no need to manually create OPC tags and map process variables from other protocols. This greatly reduces commissioning time and reduces the potential for error.

"Why embed an OPC UA server in a device? In a word: simplicity."

Because OPC UA communications can be authenticated and encrypted the installation has the potential to be more difficult for an attacker to compromise. Simply getting inside the network boundary is not sufficient to carry out an attack on a process.

While the data from embedded UA Servers would typically be routed via a central server or redundant servers there is always the option to connect to the device directly if required. This provides many options for device configuration & management as well as cost reductions for very small installations.

Finally, OPC UA is more than an industrial automation protocol. It also contains an extensible information model that makes it very attractive to many vertical markets. Adding OPC UA support to your device offers the potential to open up new markets for your product.

OVERCOMING KEY EMBEDDED UA DEVELOPMENT CHALLENGES

There are many benefits to embedding an OPC UA server in your product. Whether you realize these benefits or not however, depends on how well you address five key development challenges which we will look at in this section. They are:

Figure 3: 5 Key Embedded UA Development Challenges

1. Development cost and time to market

The Issue: OPC UA is a large set of specifications spanning more than 13 documents and 1000 pages. The standard specifies many aspects including transport protocols, security, services, information models, profiles and others. The first question a design team needs to ask when considering embedding OPC UA connectivity in their product is how long development will take and what is the risk?

Attempting to implement an OPC UA embedded server in a microcontroller based product from first principles is a major undertaking – one that is not recommended when you are also racing to get your product out to market. Why? Because it takes man years of effort and even then, the product quality, delivery date, and overall project success cannot be reliably estimated.

The Solution: In the majority of cases the only sensible course of action is the use of a commercial Software Development Kit (SDK).

MatrikonOPC offers the only OPC UA Embedded Server SDK specifically targeting resource constrained platforms such as microcontrollers. The SDK and its associated protocol stack were completely implemented from first principles to be a best in class solution for this application area. The SDK hides almost all of the complexity of the standard from the application developer enabling rapid application development. It is not unusual for prototype UA Servers to be integrated into an existing product by a customer's software engineer in less than one day. It is important to be aware that not all 'OPC UA Embedded Server SDKs' are created equal.

2. Footprint Size

"MatrikonOPC Offers the only OPC UA Embedded Server SDK specifically targeting resource constrained platforms such as microcontrollers. It has the smallest footprint in the industry. "

The Issue: If you are designing a new product you want to keep cost, complexity, form factor and power consumption to a minimum in order to maximize your return on investment while delivering value to your customers. If you have an existing product you want to enhance with OPC UA connectivity - you are constrained by the existing onboard hardware. There is only so much Flash and RAM available for the additional OPC UA server functionality. If your implementation does not fit - you cannot complete the development. For these reasons it's important to use an OPC UA implementation that minimizes memory utilization to ensure a successful development.

The Solution: Using an OPC UA Embedded Server SDK takes the guess work out of trying to determine how many of your limited memory resources the resulting OPC UA embedded server will require because that is largely predetermined.

For example, the OPC UA Embedded Server SDK from MatrikonOPC has the smallest footprint in the industry. With a Flash footprint starting at 240kB and RAM footprint starting at 35kB the server easily fits into the internal memory of a low cost microcontroller.

Table 1 - Minimum Server Footprint

Configuration	Flash (kB)*	RAM (kB)
Nano Embedded Device Server Profile	240	35
Micro Embedded Device Server Profile	265	65
Micro Embedded Device Server Profile (with full type information model)	370	65

*Metrics obtained for ARM Thumb2 instruction set (Cortex-M4F), GCC -Os

3. CPU Utilization

The Issue: Similar to memory footprint it is critical that an OPC UA Server does not swamp the product's CPU which could cause the device to malfunction or become unresponsive. After all, a typical industrial electronic device's CPU does a lot more than just perform communications. A new design will want to minimize BOM (Bill of Materials) cost while an existing design will have a fixed amount of CPU bandwidth available for communications. As with footprint, if the solution is too resource intensive it cannot be used.

The Solution: MatrikonOPC's SDK has an extremely efficient internal architecture designed to minimize CPU utilization. Typical applications on an ARM microcontroller can be implemented using less than 10% of the CPU bandwidth. SDK performance scales predictably according to the number of tags to be monitored and the tag sample rate, etc.

Table 2 - Typical Server Performance

Test	Conditions	Hardware	CPU Utilization (%)*
100 continuously changing tags	Sampling & reporting every 100ms	ARM Cortex-M4F (STM32F407) @ 168MHz	12.50
1,000 continuously changing tags	Sampling & reporting every 100ms	ARM Cortex-A8 (AM3359) @ 1GHz	31.00

*Metrics obtained for using GCC -O3

4. Ease of Integration

The Issue: As stated previously, OPC UA is a large and complex standard. When integrating advanced technology into your product there are a number of questions you need to ask yourself. What architectural demands does the technology place on my product? Do I need to change my software architecture in major ways in order to achieve integration? Is it compatible with my existing infrastructure such as my TCP/IP stack and variable database?

"The OPC UA Embedded Server SDK from MatrikonOPC is a simple single threaded implementation that runs in a single RTOS task or in a bare metal environment. "

The Solution: An ideal solution will hide the complexity from the application developer and easily slot in to your existing software framework regardless of how bespoke it may be. The OPC UA Embedded Server SDK from MatrikonOPC is a simple single threaded implementation that runs in a single RTOS task or in a bare metal environment. Integration is as easy as hooking up a small number of API function calls to your application. Minimal changes are required so you don't have to waste time modifying software unnecessarily.

5. Reliability

The Issue: With any embedded system whether writing software from first principles or integrating third party IP, quality is absolutely critical. An unreliable and/or underperforming product can damage your brand's reputation and cost you market share. While subsequent fixes are possible via firmware upgrades – they are inconvenient and costly for your customers. Reliability is not just a result of doing things well; you also have to do the right things. Take memory management for example. In a PC based environment it is quite normal for software developers to allocate software data structures on the "heap" or "free store". This is known as dynamic allocation. While an ideal approach on the desktop, this approach does not scale well when applied to resource constrained environments. When device memory is limited and software relies on heap based allocation bad things can happen. The most obvious problem is heap exhaustion whereby the program simply runs out of memory during normal operation. A lesser known problem is heap fragmentation whereby there is sufficient memory available but it becomes so fragmented over time that there is no sufficiently large block of memory available to service a particular memory allocation. In both such cases, the application cannot continue to perform its intended task.

"A properly written OPC UA embedded server SDK uses a special memory model that avoids heap exhaustion and fragmentation from occurring regardless of how many years the device runs. "

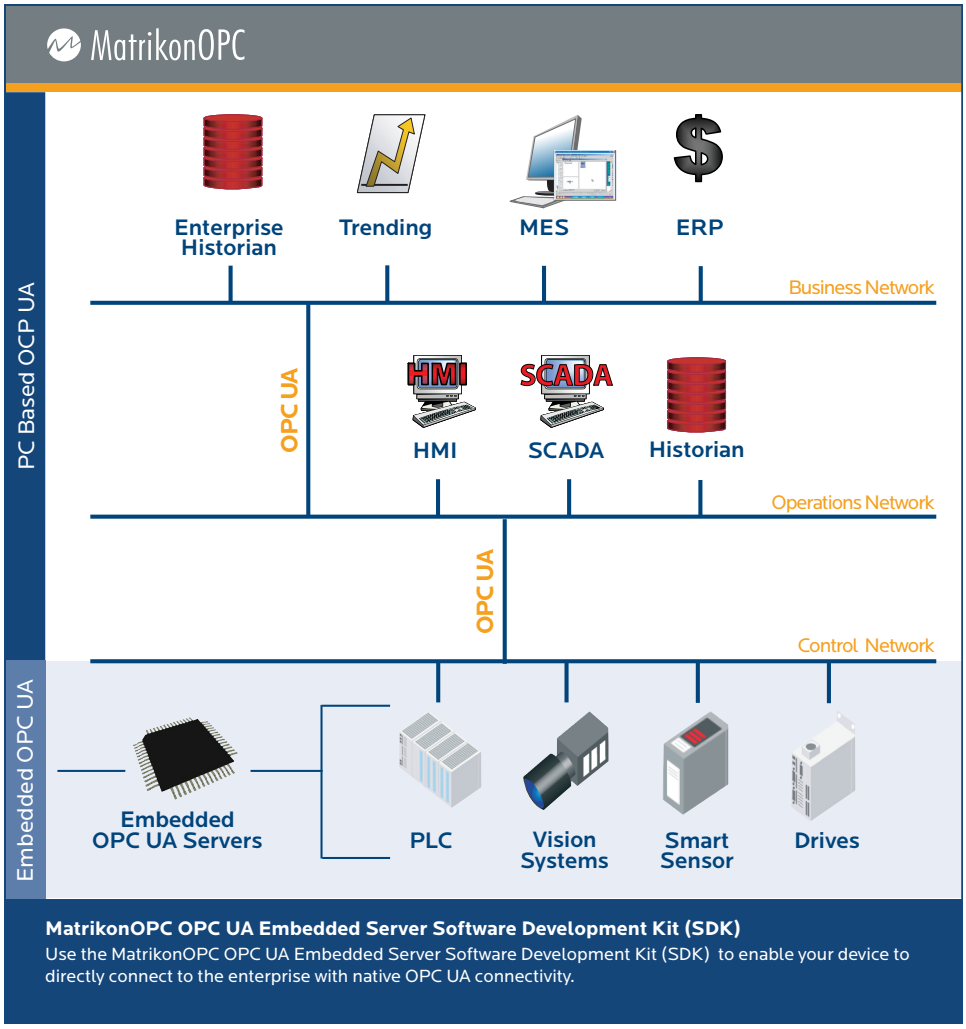
The Solution: A properly written OPC UA embedded server SDK uses a special memory model that avoids heap exhaustion and fragmentation from occurring regardless of how many years the device runs. Given the nature of industrial processes and their need for dependable up-time - this needs to be a key characteristic of the embedded OPC UA library you choose to put into your product.

Using an OPC UA Embedded Server SDK that has been independently certified by the OPC Foundation is a key way to ensure that you are using a high quality SDK. For this reason you should insist on only using components that have been independently certified. The OPC UA Embedded Server SDK from MatirkonOPC was the first OPC UA SDK to be certified as being compliant by the OPC Foundation (August 2011).

CONCLUSION

The demand for control automation products that are easy to integrate into a company's data sharing infrastructure is on the rise. OPC UA, the latest generation of the world's most popular open data connectivity standard allows for OPC UA servers to run on virtually any platform or OS – including embedded applications. By using a high quality OPC UA Embedded Server software development kit from MatrikonOPC – you can quickly add value to your industrial electronic devices while avoiding five common development challenges.

"MatrikonOPC OPC UA Embedded Server SDK enables your devices to directly connect to the enterprise with native OPC UA connectivity."



MatrikonOPC OPC UA Embedded Server Software Development Kit (SDK)
Use the MatrikonOPC OPC UA Embedded Server Software Development Kit (SDK) to enable your device to directly connect to the enterprise with native OPC UA connectivity.

"Add real value to your product by enabling direct point and click configuration, management and monitoring from any OPC UA Client."

OPC COMPOMENTS DESCRIBED IN THIS PAPER

MATRIKONOPC OPC UA EMBEDDED SERVER SOFTWARE DEVELOPMENT KIT (SDK)

The OPC-UA Embedded Server SDK from MatrikonOPC is a software development kit that allows you to quickly and easily add an OPC UA Server to your embedded product. Our scalable, standards based SDK can be integrated into every class of device, from discrete sensors and actuators to programmable controllers and beyond. Add real value to your product by enabling direct point and click configuration, management and monitoring from any OPC UA Client.

Toll Free 1-877-MATRIKON (1-877-628-7456)
Ph: +1 (780) 945-4099
Email: info@MatrikonOPC.com
Web: www.MatrikonOPC.com