

THE EVOLUTION OF INDUSTRIAL MACHINE ARCHITECTURES

A series of bright green light trails and streaks, appearing as if from a high-speed camera or a digital simulation, set against a dark background. The lines are dense and radiate from the left side, creating a sense of motion and energy.

THE TRANSITION TO
HUMAN-CONNECTED
MACHINES

By Terry West

SeriousTM

A series of bright blue and purple light trails and streaks, similar to the green ones above, set against a dark background. The lines are dense and radiate from the right side, creating a sense of motion and energy.



TABLE OF CONTENTS

Introduction	5
Industrial Machine Architectures: Separating Control from Power and I/O	9
Towards Fully Distributed Power and I/O Subsystems	17
The Evolution of the Control Subsystem	23
The Software Ownership Trap	27
System on Modules to the Rescue?	31
SOMs to the Next Level – Intelligent Subsystems	35
The Evolution of Industrial Machine Architectures: A Case Study	37
Human-Connected Industrial Machines	49



TODAY'S CONSUMER DEVICES are highly interactive, highly connected digital platforms designed with flexibility and the best possible user experience in mind. They are easy to use and remotely accessible. They are, in a phrase, "human-connected machines."



INTRODUCTION

Over the past 30 years, industrial, medical, and commercial devices have also experienced a dramatic transformation from pure mechanical or electromechanical systems to “mechatronic” systems. This transformation applies to devices of all types and sizes, from benchtop laboratory equipment and production line equipment to large factory floor and retail food service machines. Their mechanical and electrical architectures have given way to more reliable electronic sensors and actuators. Their control systems have been revolutionized by highly flexible and programmable microcontrollers (MCUs).

However, compared with their consumer counterparts, these devices are still several evolutionary steps behind. Largely isolated from IT, Internet of Things (IoT)/cloud, and industrial networks, they typically also have woefully outdated operator controls of pushbuttons; LEDs; and, at best, small text displays for basic menus, configuration, and status information.

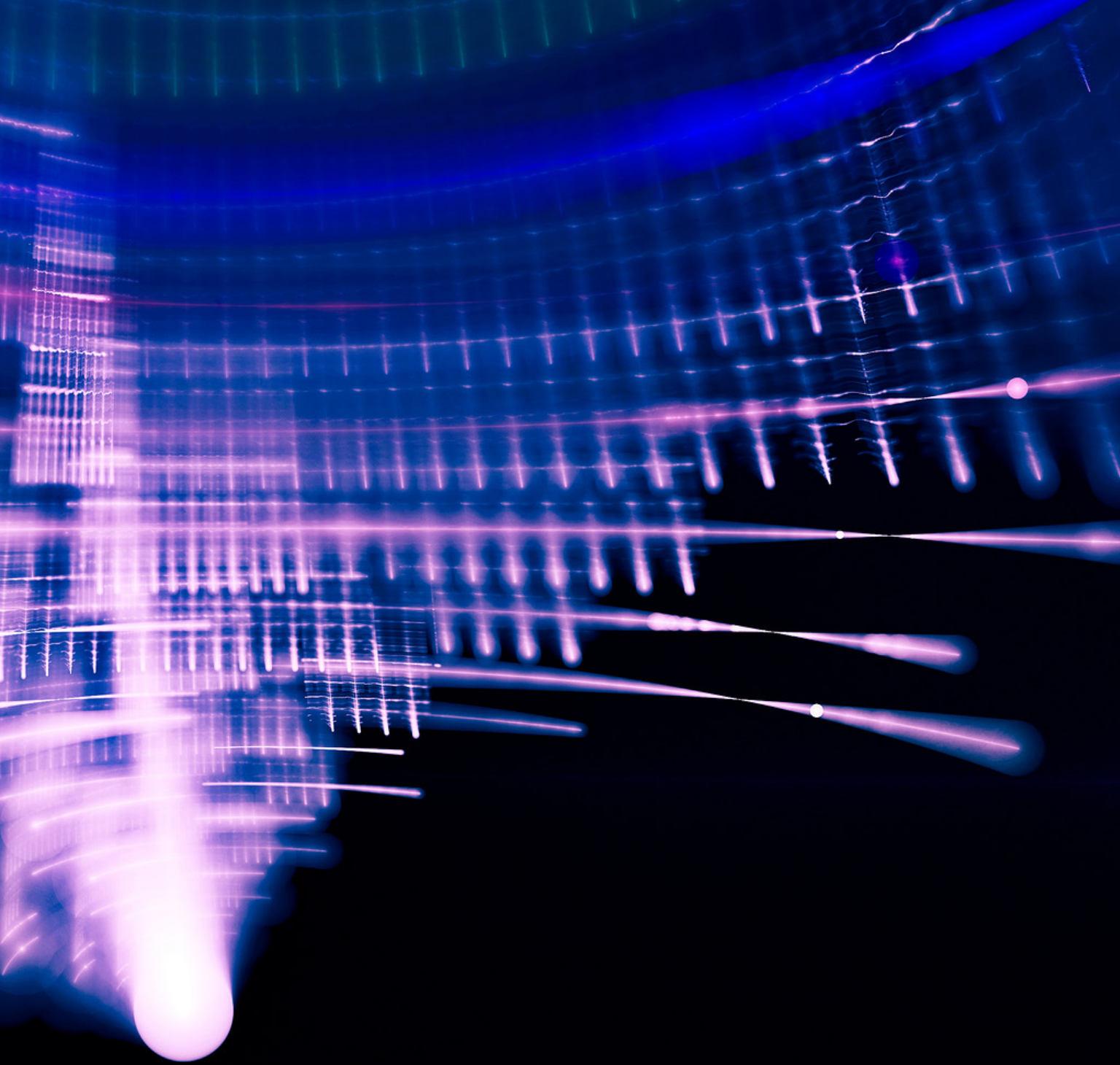
Architecturally, these systems are the product of 30 years of electronic design incrementalism supported by ever-increasing MCU memory and processing power. An evolutionary plateau of industrial devices has become the status quo due to a variety of market forces, including security, long lifecycles, and technology reuse. The cost of further evolving them into highly interactive, highly connected devices keeps going up, especially in terms of software investment.

Still, the benefits of transforming these industrial systems into highly human-connected machines are getting harder to ignore – with improved application flexibility, remote upgradeability, and ease-of-use come significant gains in end-user productivity and efficiency. These features also present the opportunity for original equipment manufacturers (OEMs) to increase market share.

But to really understand why millions of industrial products are lagging so far behind the consumer technology evolution, we need to start at the beginning.

LARGELY ISOLATED from IT, Internet of Things (IoT)/cloud, and industrial networks, they typically also have woefully outdated operator controls of pushbuttons; LEDs; and, at best, small text displays for basic menus, configuration, and status information.



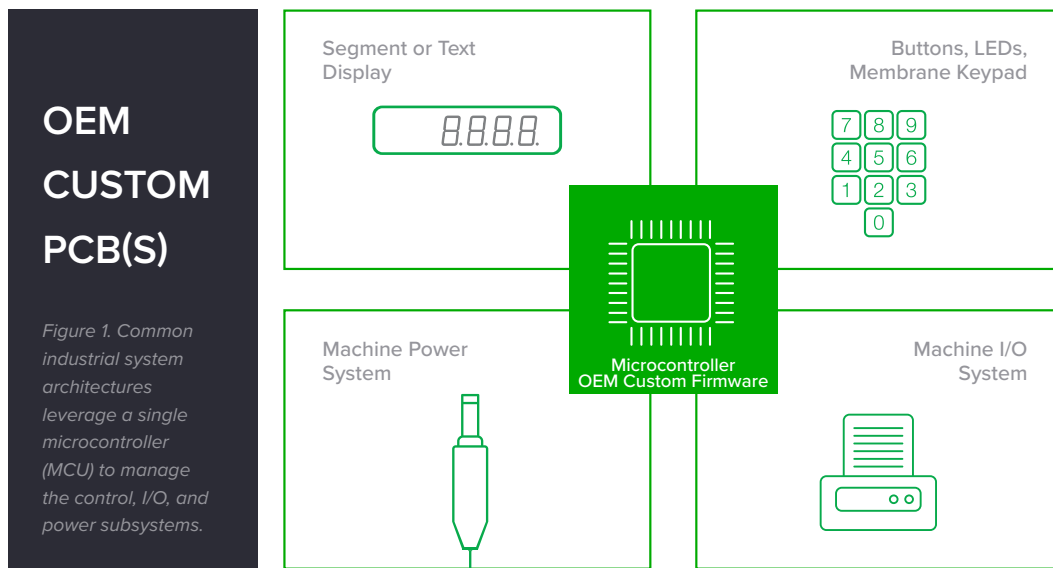


INDUSTRIAL MACHINE ARCHITECTURES: SEPARATING CONTROL FROM POWER AND I/O

At the heart of traditional industrial machine architectures lies a central, unified MCU that drives the control, power, and I/O subsystems (Figure 1):

- ▶ **Control subsystem:** The control subsystem of an industrial machine is responsible for I/O monitoring, control, and sequencing; operator interface management; external system communications; as well as machine accessory control.
- ▶ **I/O subsystem:** The I/O subsystem is typically comprised of relays, sensor interfaces, and analog front ends that transfer input signals (such as commands entered through an operator interface) and output signals (such as the information returned to the operator interface) between subsystems and users.
- ▶ **Power subsystem:** The power subsystem handles the transmission, conversion, distribution, and utilization of AC and DC electrical currents throughout a machine.

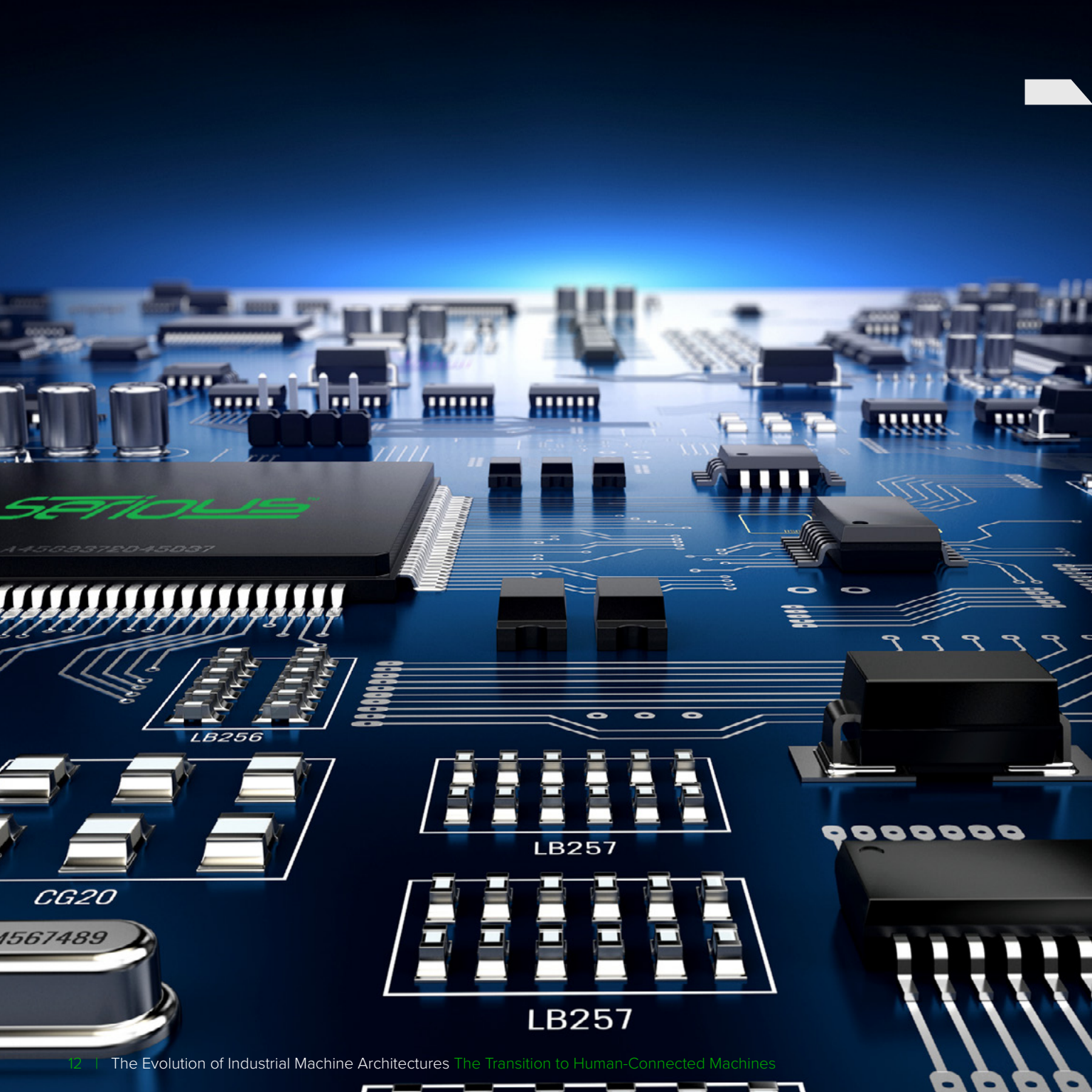
HISTORICALLY, OEM software teams producing industrial systems have focused on constantly refining, debugging, and updating the custom firmware required to manage these subsystems on a single MCU.



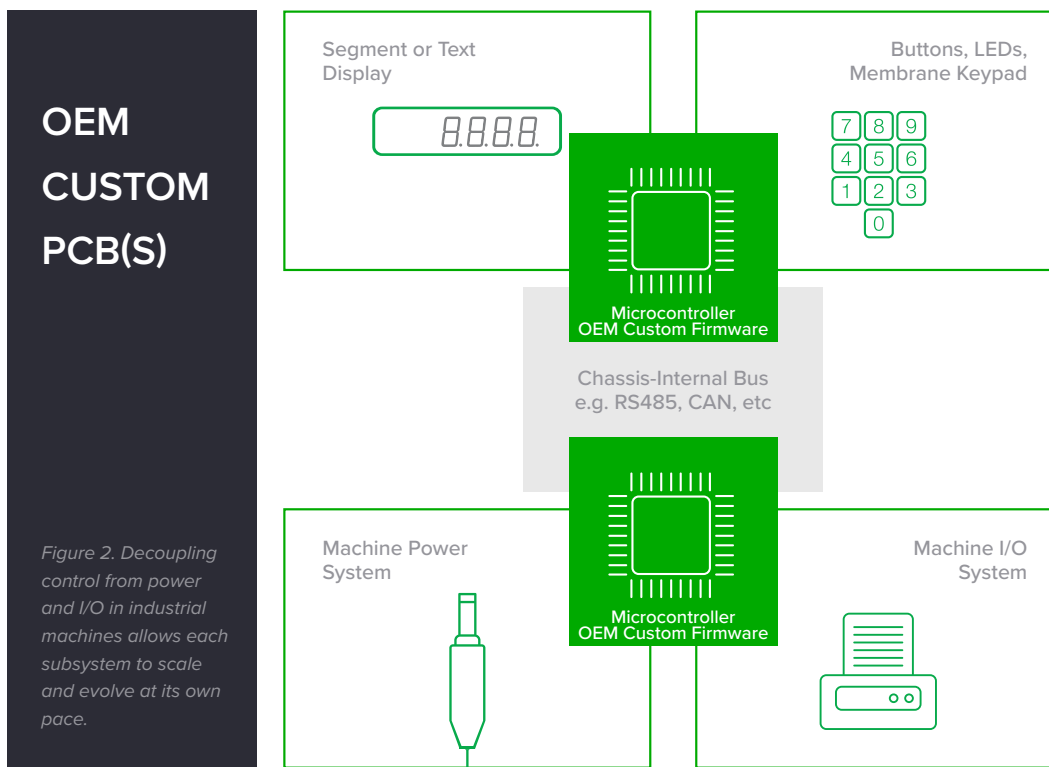
Given the tight integration of software components required by this architecture, it's not surprising that the proliferation of unwieldy legacy codebases is common in OEM engineering organizations. The pattern is widespread. Years of software maintenance, feature additions, and growing complexity are followed by hardware redesigns that provide access to more processing and memory – all so that codebases can continue to grow.

This engineering incrementalism is the obvious (and seemingly lowest cost) approach at each step. But, like with all incremental engineering paths, inevitably the moment comes when the cost of maintenance and sustaining engineering overwhelms the resources available. At this point, the architecture must be redone to simplify the codebase and enable a new set of capabilities.

So, as OEMs looked to increase functionality over time they came to a distinct realization: control subsystem technology advances at a faster pace than the power and I/O subsystems.



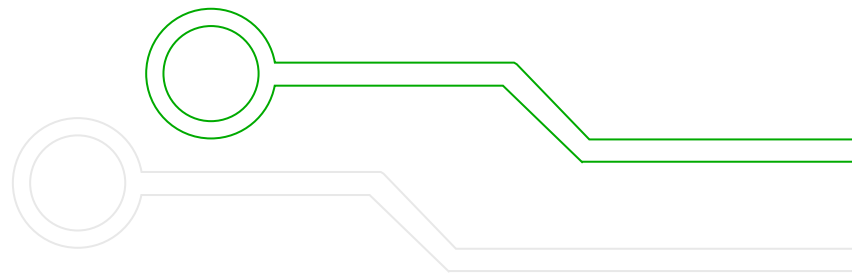
This fact has dual significance. First, it has been a key driver of investment in bigger, faster MCUs to support expanding and more complex control system software. Second, it has led to recognition of the natural, physical separation between the control subsystem and power and I/O (Figure 2).



This decoupling is significant in that it provides several positive outcomes for system designers:

- ✔ **Distributed processing and software reduction:** With 32-bit MCUs available for less than \$1 in volume, distributed processing architectures that provide a dedicated MCU for the power and I/O subsystem are now economically viable. As a result, the software on these slower-evolving subsystems can be minimized, well-reviewed, and seldom changed. The dedicated level of intelligence also enables functions such as real-time sequencing of events and safety interlock capabilities.
- ✔ **Software partitioning:** Partitioning the software of each subsystem allows independent software maintenance, evolution and scalability. This extends to physical system components such as wiring and the printed circuit board (PCB), which do not have to change in the event of a control subsystem upgrade. Machine retrofits and product line segmentation are therefore simplified.
- ✔ **PCB and component technology separation:** Decoupling allows the control subsystem to operate on PCBs with finer pitches that utilize high-speed surface-mount technology (SMT) and lower voltages, while the power and I/O subsystems can remain on lower technology. These lower technologies include cheaper PCBs with coarser pitches that support larger components and higher voltages.
- ✔ **System design:** The separation of subsystems also frees up physical restrictions

within an enclosure. For instance, a power and I/O board can be positioned at the back or base of a machine near the power source and sensor/actuator wiring harness, while the control system can be co-located with the human-machine interface (HMI) at the front of the machine. Large power noise issues typical in I/O and power subsystems can be completely isolated from the more sensitive high-speed control system technology. Furthermore, the control board may be able to forego expensive treatments such as conformal coating when the subsystems are separated.





TOWARDS FULLY DISTRIBUTED POWER AND I/O SUBSYSTEMS

The previously described benefits are similar to what has transpired in the automotive industry. There, I/O systems for intelligent sensors and actuators became completely distributed over CAN networks managed by one or more control systems.

In industrial device architectures, OEMs often leverage off-the-shelf CAN or RS-485 differential wiring and transceivers to connect these distributed control and power and I/O subsystems within a single enclosure. These communications solutions are supported by small, often-homegrown protocols that abstract I/O much like a programmable logic controller (PLC). For example, turning on a relay would involve the control subsystem MCU sending the command “set output 43 to 1” over CAN/RS-485 to the power and I/O subsystem MCU. Inputs can also be polled in a master/slave topology, such as “get input 34, a floating-point ADC input value.”

While even this simple step in topology evolution can yield great results for system designers and entire product lines, it is only the beginning. The evolution can continue to its logical conclusion of separating the power subsystem from the I/O subsystem, as is done in modern automotive system architectures where sensors and actuators are fully distributed, often hierarchical, and separate from the power system. This enables subsystem-level scalability across different products, easier subsystem unit testing during design and manufacture, and more effective machine assembly and maintenance.

For instance, in a fully decoupled subsystem topology, the hub-and-spoke wiring of an I/O board gives way to independent, somewhat intelligent sensors and actuators that reside on the internal control bus (Figure 3). Power is received over a separate power bus that is routed throughout the machine.

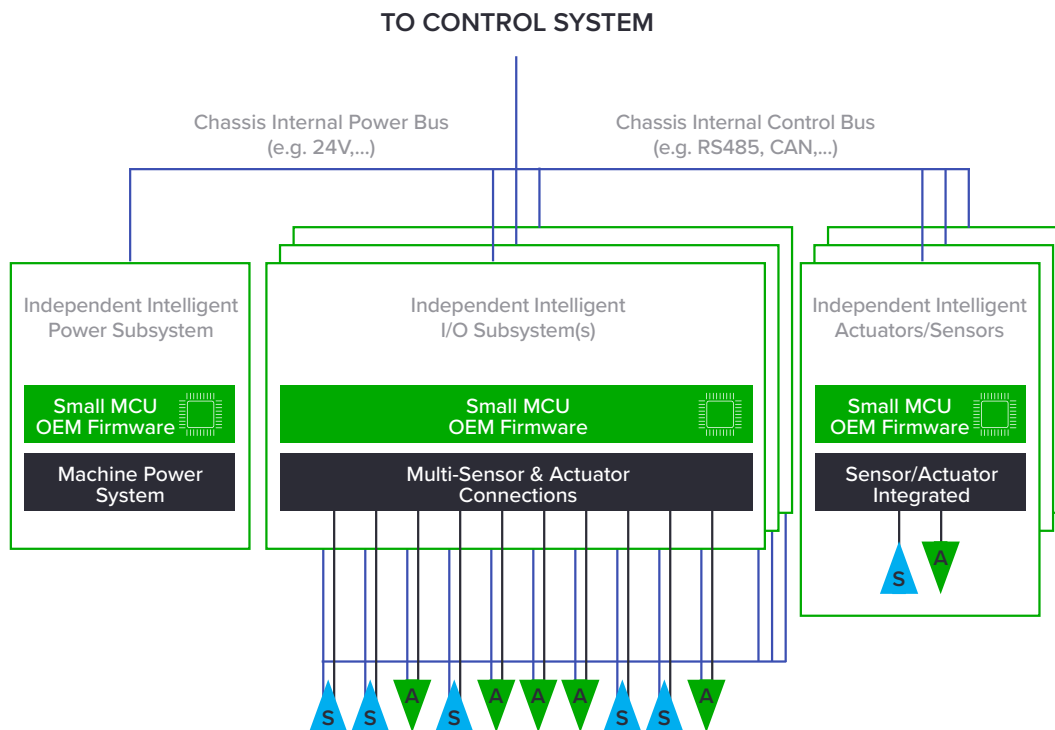


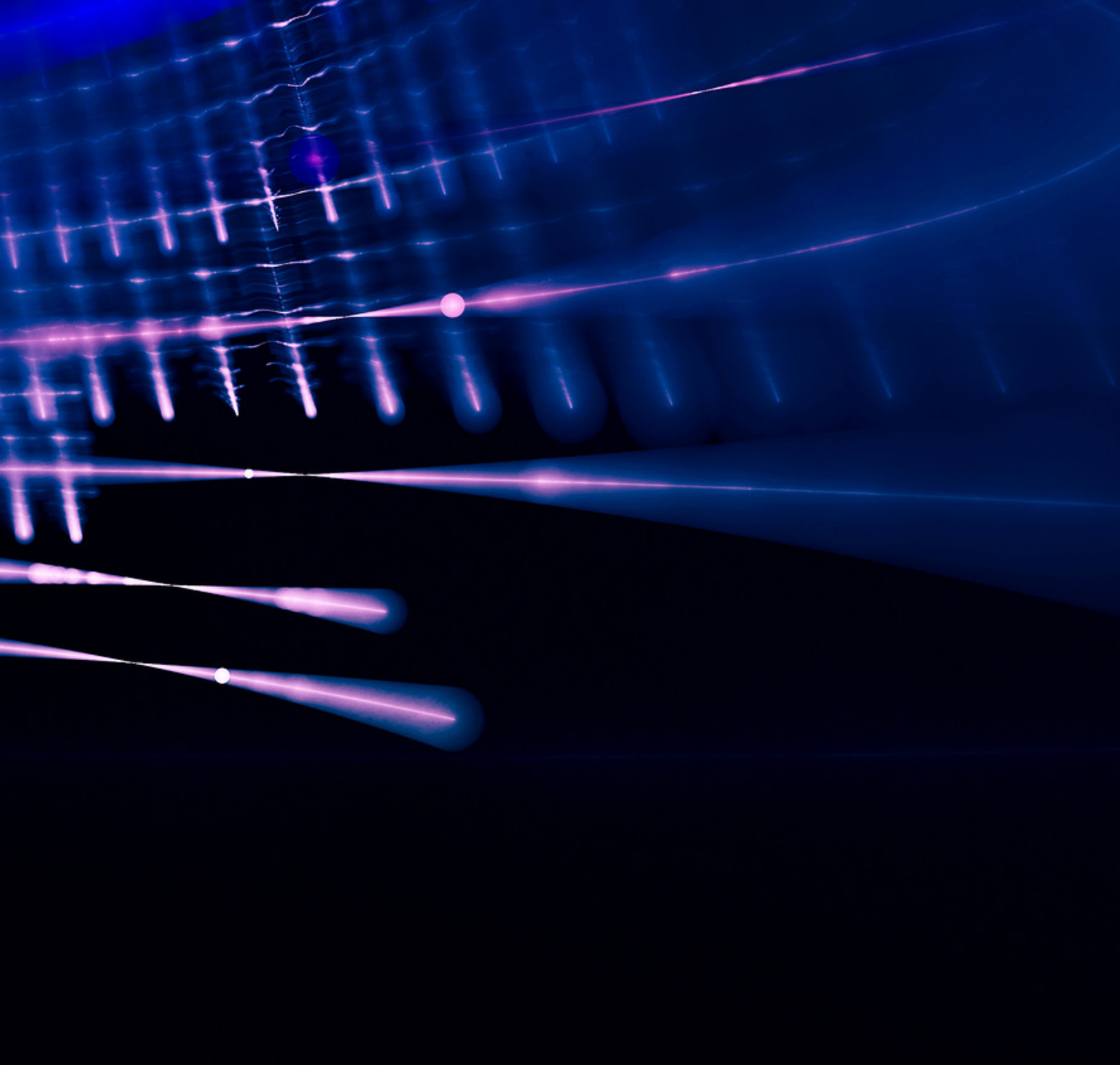
Figure 3. The continued separation of industrial machine subsystems enables independent scalability and levels of intelligence not previously available.

The elimination of traditional “large I/O boards” permits:

- ✔ Scalable machine capabilities through the simple addition or removal of mechanical components and their associated sensors and actuators on the bus
- ✔ Sensor and actuator abstraction that enables multi-vendor sourcing (for example, an inexpensive thermistor can be integrated into a precision temperature measurement system)
- ✔ Faster and cheaper diagnostics and maintenance when a sensor or actuator fails
- ✔ Vastly simplified machine wiring and debugging on the control and power buses
- ✔ Significantly improved system assembly and manufacturing times
- ✔ Reduced churn in large and/or complex I/O PCBs

A close-up photograph of a black, perforated steering wheel. The background is dark and out of focus, showing various lights and mechanical components, suggesting a car's interior or a workshop.

THIS IS NEITHER A NEW NOR RADICAL CONCEPT. The automotive industry made this transition decades ago for exactly these reasons. Modern industrial machine designers are now standing on those shoulders as they evolve their own products.



THE EVOLUTION OF THE CONTROL SUBSYSTEM

While the basic ingredients of control subsystems have not changed, the requirements on each element have increased dramatically as industrial machine architectures have evolved:

- ▶ **I/O monitoring, control, and sequencing** – The control system is normally responsible for sequencing the machine through its various operational states and sub-states. The control system in traditional machine architectures reads sensors and fires actuators directly. In highly distributed architectures, however, sensors and actuators become separate and independently intelligent. Data can therefore be preprocessed at the sensor, with the control loop maintaining a central data structure of abstracted I/O variables that it reads for control sequencing. A separate mechanism is used to synchronize these variables with various remote I/O elements.
- ▶ **Operator interface** – The operator interface in a traditional machine includes a collection of indicators (such as LEDs and digit displays) and switches or knobs (such as membrane keypads). These are controlled by a super-loop software architecture that scans and sequences indicators based on machine state.

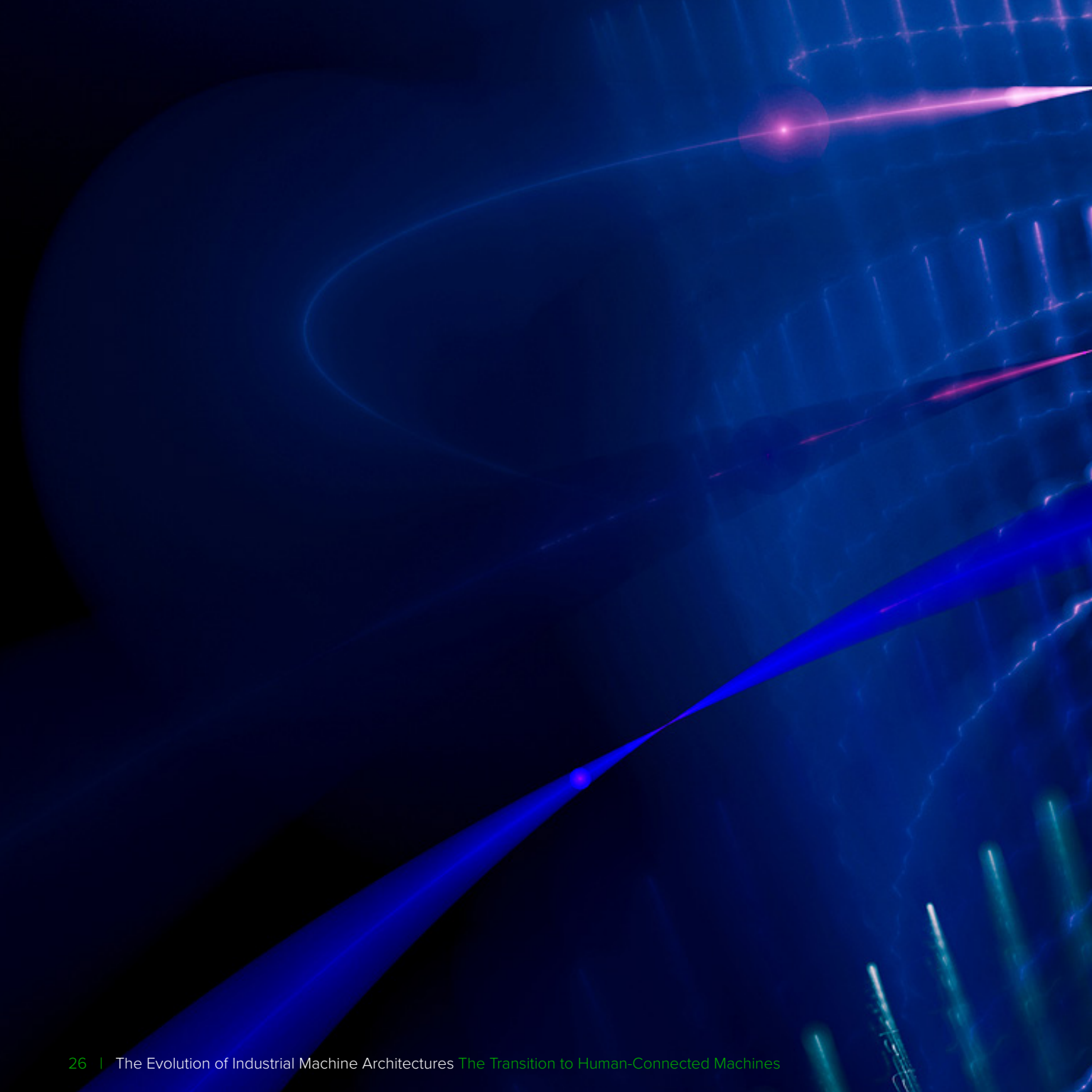
As the complexity of the operator interface increases with monochrome segment LCDs or touch-screen graphical user interfaces (GUIs), more complex software and higher performance MCUs with more memory are required. A super-loop would collapse under its own weight given this complexity. As a result, human-connected industrial machines are transitioning to the use of RTOS kernels, repartitioned code, and software libraries or high-level GUI builders.

- ✔ **External communications** – More options have emerged for industrial machine communications and connectivity over the past ten years than in all prior decades combined. New connectivity options, including USB, TCP/IP (Ethernet, Wi-Fi), Bluetooth, and many other technologies offer new ways to interact with machines. Over the wire/air updates, remote monitoring and control, PC connectivity, and inexpensive peripherals are powerful capabilities, albeit at the expense of software and system architecture headaches. They also drive MCU performance and power requirements.
- ✔ **Machine accessories and inter-machine coordination** – As software architectures are reinvented in a more modular fashion, they enable new machine configurations capable of post-sale upgrades and feature additions. Along with machine-to-machine (M2M) coordination, these can create new revenue models and business opportunities.

Just as industrial system designers began evolving machines to these new architectures, the world shifted. Then came advanced industrial networks, IT networks, and now the cloud and the Industrial Internet of Things (IIoT).

While IT and Internet connectivity have been technically feasible for industrial machines since the 1990s, the last decade changed expectations through a whole new level of human-machine connectedness. Remote cloud dashboards, analytics, predictive maintenance, monetized and franchised data, and many more capabilities are now an essential part of the feature roadmap for every industrial machine.

All of these trends point to the single biggest challenge of the industrial machine evolution: managing exploding software complexity.



THE SOFTWARE OWNERSHIP TRAP

OEMs continue to do the majority of their software and hardware development in-house. Software teams are rarely equipped with the resources to maintain, grow, and evolve ever-expanding codebases rapidly enough to keep pace with business and market demands.

A significant portion of the software challenges lie in maintaining the infrastructure required to support an application's growing needs, not in the application itself. This infrastructure includes, but is not limited to, support for connectivity to the cloud and other machines, security, and advanced HMIs.

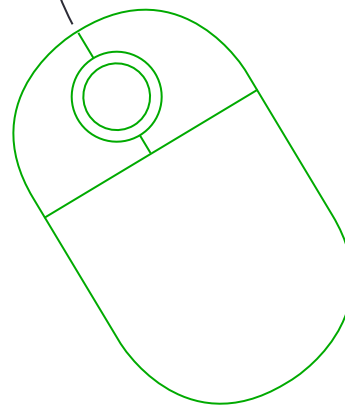
To augment their software development efforts, OEMs frequently leverage foundational elements of their software from MCU vendors offering free libraries and pre-packaged software stacks. Unfortunately, these only tend to exacerbate the problem.

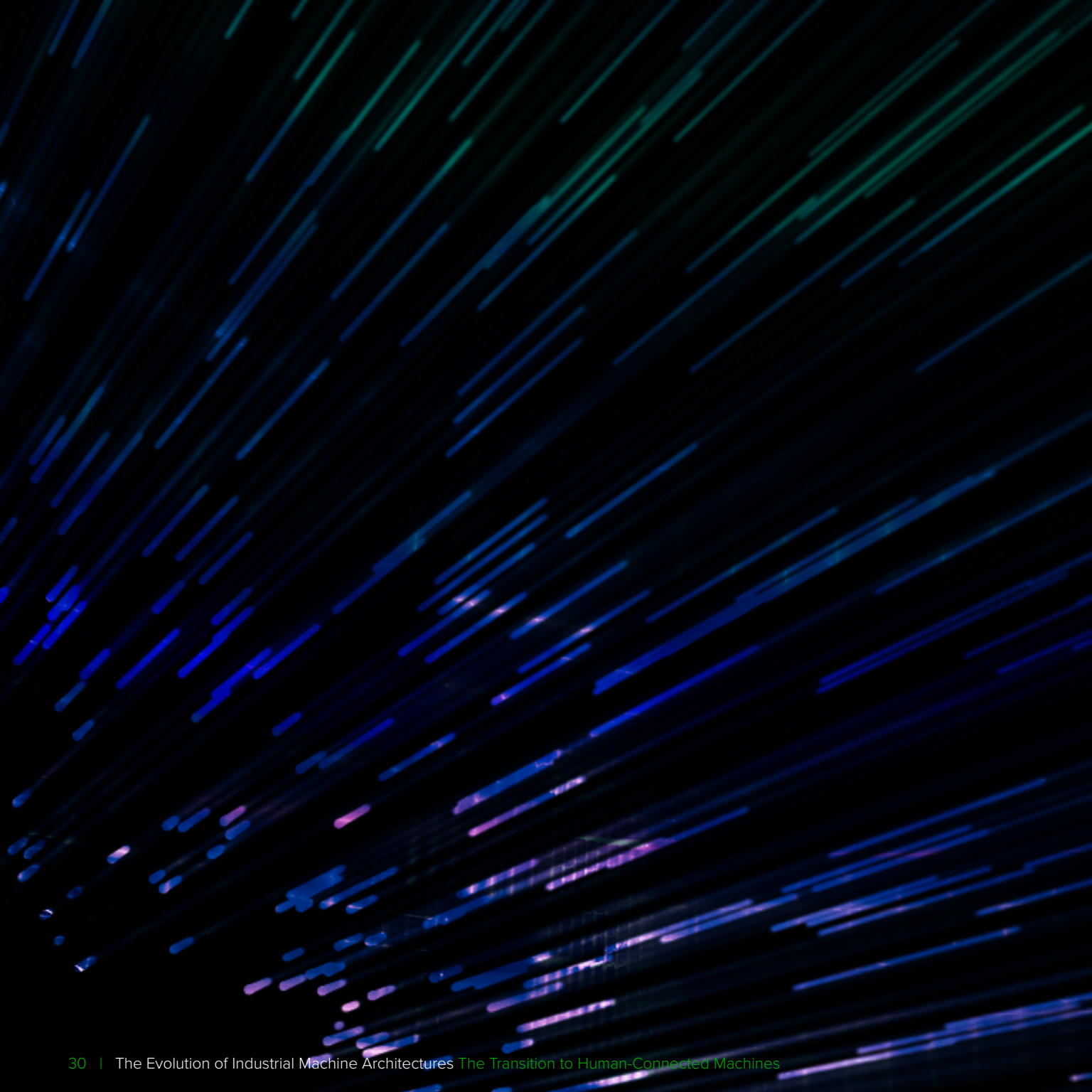
CERTAINLY, the “free” aspect of these MCU vendor libraries is attractive. But as an MCU vendor’s software is changed, expanded, and rewired in an OEM application over time, responsibility for maintenance of the codebase slowly shifts back to the OEM.

Furthermore, the provided stacks are typically little more than “as-is” demo code that doesn’t correspond with any published roadmap. Ambiguous change management and inadequate support often ensue.

Given these costs it’s surprising how few OEMs incorporate robust, commercially supported third-party software, such as an embedded operating system (or “RTOS”) and associated stacks such as file systems, TCP/IP, USB, and cryptography. The one-time fee and ongoing maintenance subscription for these full-featured solutions can cost less than a single software engineer.

This approach frees the OEM from a specific silicon vendor so that the MCU can be commoditized into raw horsepower, peripherals, and memory. But, just as important, the software architectures are inherently more modular, maintainable, and portable. Swaths of custom infrastructure software and hardware-specific firmware are replaced with commercially supported, off-the-shelf software ingredients. This simple step allows OEM software teams to focus on their core intellectual property (IP) and value proposition rather than maintaining low-level software infrastructure.






SYSTEM ON MODULES TO THE RESCUE?

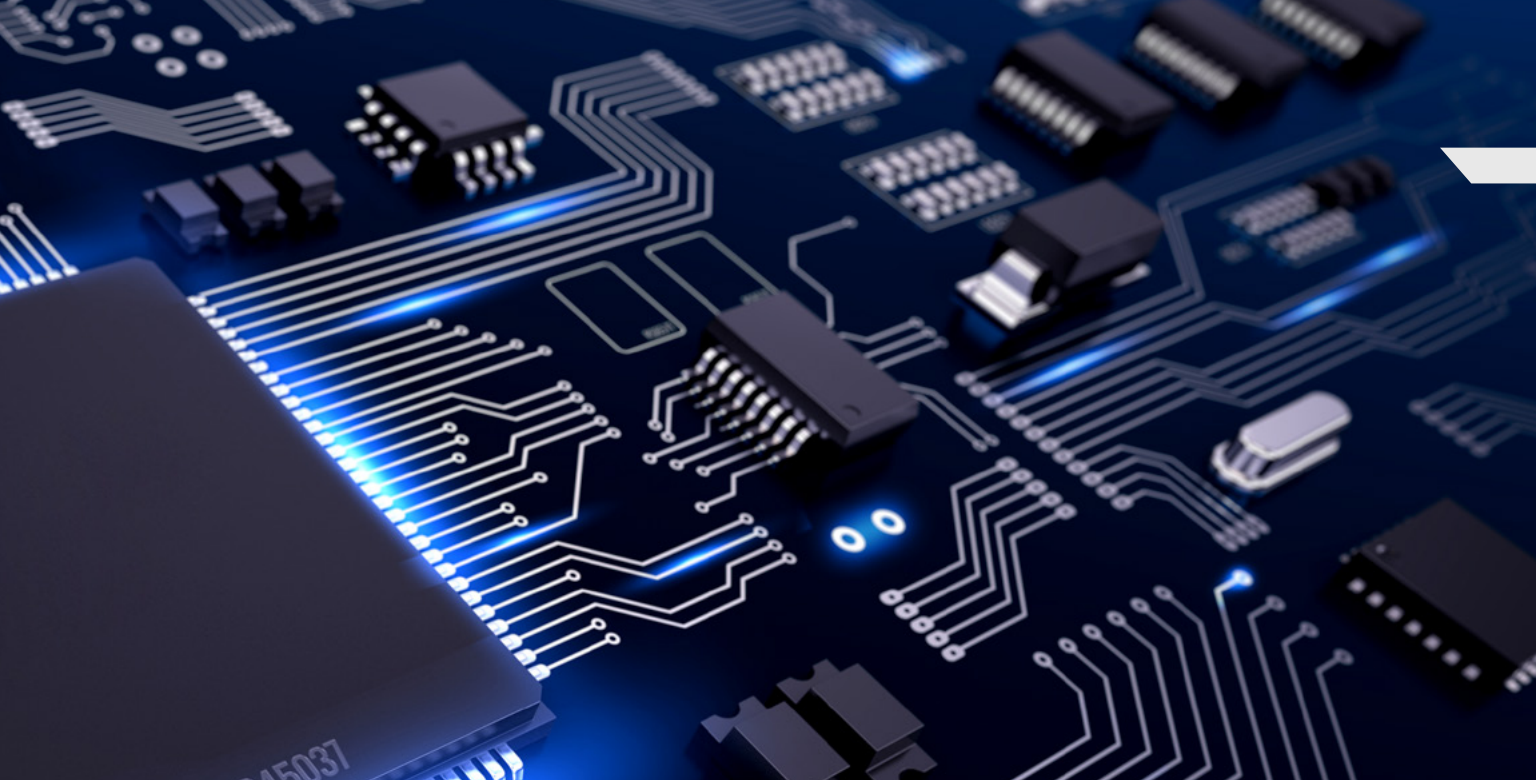
The hardware, layout, and certification challenges of using, for example, Wi-Fi chipsets in a design are non-trivial. However, many system designers will assert that these challenges pale in comparison to the challenges of merging, debugging and maintaining the chipset vendor's free software stack within their own software architecture.

The system on module (SOM) deployment model has been so successful for this reason. A Wi-Fi SOM, for instance, integrates an MCU, chipset, layout, and certification in a single, off-the-shelf package. It solves not only hardware design challenges, but also avoids software integration issues by burying pre-engineered TCP/IP and radio software stacks inside the module. The SOM approach essentially turns highly complex subsystem design challenges into much simpler off-the-shelf subsystem integration challenges.

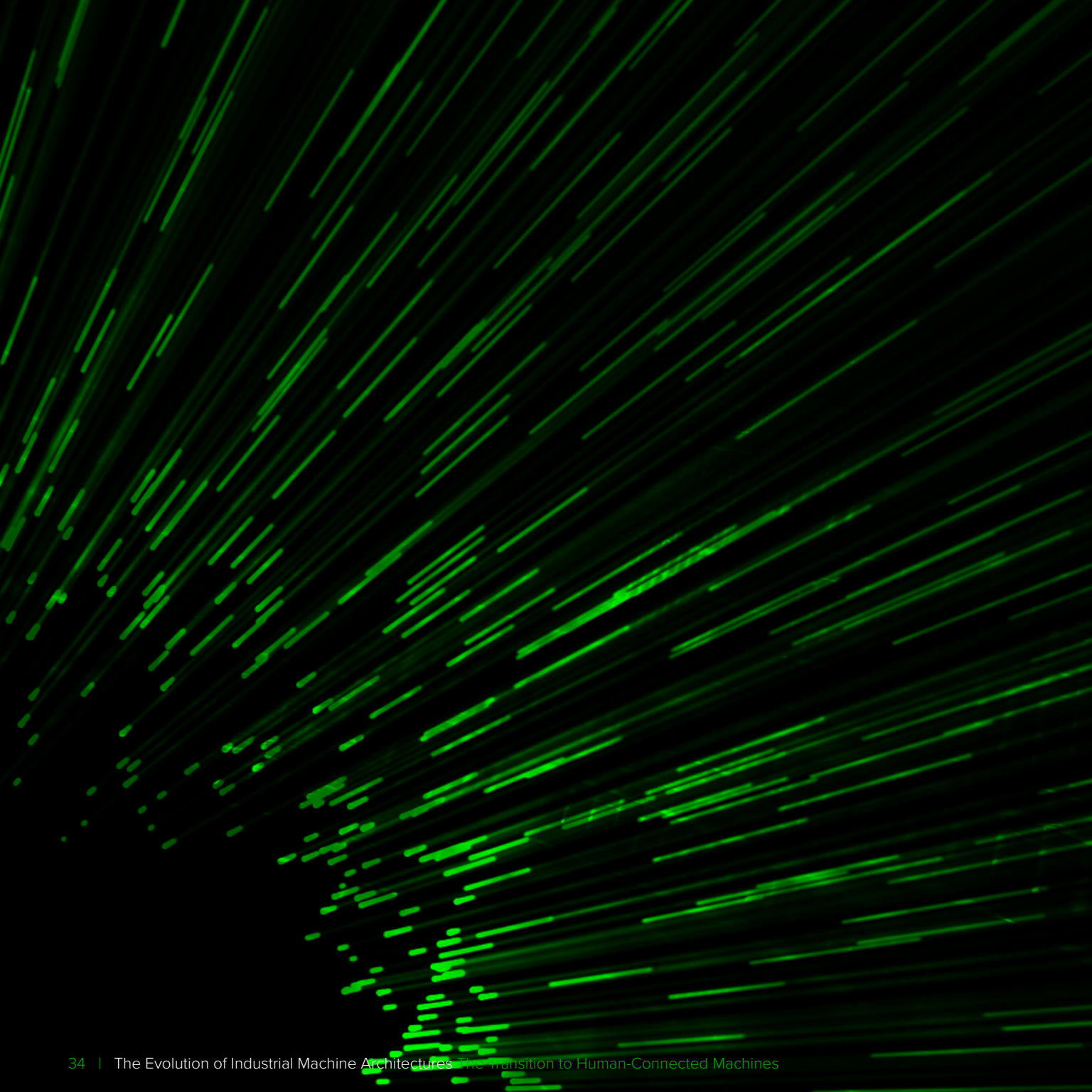


SOMs effectively create distributed processing architectures within OEM designs. Smart subsystems can be managed by the main control MCU and interconnected using UART, SPI, or I2C ports. Where necessary, OEMs can leverage the expertise of SOM original design manufacturers (ODMs) to create semi-custom derivative SOMs that are highly consistent with off-the-shelf versions from a software perspective. Compared to developing these hardware subsystems and software stacks from the ground up, the time-to-market and engineering savings more than offset the small premium for these services.

But while SOMs simplify the initial work of adding technologies such as Wi-Fi, Bluetooth, 4G LTE, and others to industrial products, end-to-end system requirements again contribute to burdensome software stacks. For example, many Wi-Fi SOMs have built-in TCP/IP stacks that make it easy to connect to simple sensors or MCUs with little overhead. Unfortunately, most industrial, medical, and commercial products are not simple sensors, but complex systems with dozens of sensors and actuators coordinated by a sophisticated control system. These products have complex connectivity needs, including two-factor authentication, improved Transport Layer Security (TLS), and the ability to perform seamless handoffs from one wireless technology to another.



UNLIKE SIMPLE MOTES AND SENSORS, these products basically demand the equivalent of an IoT gateway built into the product. This added sophistication outstrips the capabilities of stacks inside of SOMs. Low-level drivers and wireless certifications retain their value, but more complex system-level software falls back on the OEM to conquer in-house.



SOMS TO THE NEXT LEVEL – INTELLIGENT SUBSYSTEMS

The SOM concept is a good one in that it distributes the software engineering burden to the SOM manufacturer rather than the OEM. It's just at the wrong level of granularity.

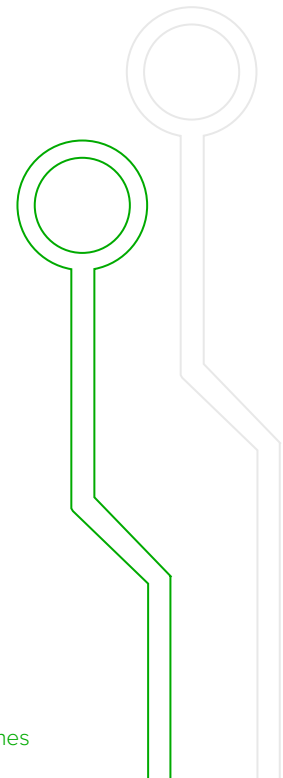
Take, for example, the Wi-Fi module case. If the SOM could be abstracted up a level in the system architecture to a communications hub, the handoff and TLS software integration issues could be resolved. The TCP/IP stack could be partitioned to deliver the required resources to peripherals like Ethernet, Wi-Fi, and 4G LTE, with a unified security and authentication stack residing there as well.

Similarly, graphical/touch HMIs could be partitioned from the control system in this way to offload GUI management, file and event systems,

and free the OEM from churn in the LCD supply chain. It also provides front-panel scalability independent of the control system and machine I/O.

Of course, not all machines and devices – not even all IoT-connected products – need all of these features. But this concept does provide the ability to isolate scalable subsets of a system without increasing the software burden on OEMs.

And these architectures are not theoretical.





THE EVOLUTION OF INDUSTRIAL MACHINE ARCHITECTURES:

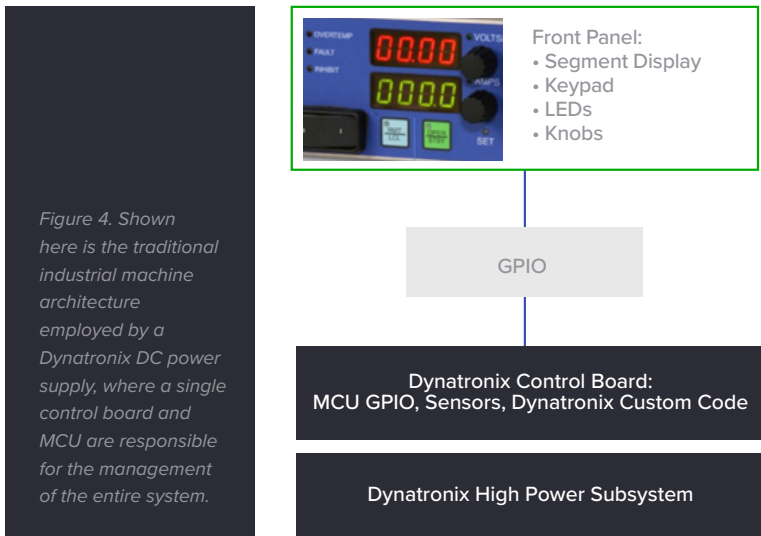
A CASE STUDY

CASE STUDY

HEADQUARTERED in Amery, WI, Dynatronix has been manufacturing DC, pulse, and reverse power supply systems since 1971.

For most of the company's existence, its power supply platforms have employed a traditional industrial machine architecture based on integrated control, power, and I/O subsystems. These system designs were built around a control board equipped with a single MCU that was responsible for managing the power supply subsystem, as well as knobs, keypads, buttons, and segment LED displays over GPIO (Figure 4).

From engineering to sales and marketing, the [Dynatronix](#) team had a vision for the future of their products that included advanced graphical HMIs and industrial and IoT connectivity. Unfortunately, their internal engineering capacity limited the company's ability to evolve their existing software and hardware infrastructure in such a dramatic fashion.



In late 2016, Dynatronix decided to implement their vision in the upcoming line of DTX 2400 Series DC rectifiers (Figure 5). Key features of the product would be an advanced graphical HMI; Ethernet connectivity with a path to IoT connectivity; and over-the-wire monitoring, diagnostics, and upgradability. Despite this aggressive list of transformational features, they also needed to be in-market with the devices in less than 8 months.



Figure 5. The Dynatronix DTX 2400 represents several evolutionary jumps in industrial machine architecture that enable interactive HMIs, industrial network communications, future IoT connectivity, remote monitoring, and over-the-wire upgrades.

DISTRIBUTED ARCHITECTURE OF THE DYNATRONIX DTX 2400

Where the heart of previous platforms had been an integrated power, I/O, and control board, Dynatronix realized a very different architectural approach was needed if they were to evolve their product now and in the future. The DTX 2400 came with an extensive list of communications and HMI design requirements (Table 1).

To meet the demands of their new highly connected system, Dynatronix selected the *Serious Communications/Control Module 208 (SCM208)* and *Serious Integrated HMI Module 231*

DTX 2400 Requirements

COMMUNICATIONS

- ✔ RS485, and 10/100 Ethernet networking
- ✔ Processing and memory resources for current communications and control algorithms, as well as headroom for future protocols (such as Ethernet/IP), web servers, etc.
- ✔ Out-of-the-box commercial software supporting application-level development, including:
 - Operating system (OS), TCP/IP, file system, Ethernet, and USB stacks and drivers
 - Over-the-wire upgrade frameworks
 - Data movement and protocols for HMI and in-system communications
 - Easy addition of high-level protocols (such as Ethernet/IP)
 - Easy addition of high-level TCP/IP systems (such as web servers)
- ✔ GPIO for keypad scanning and indicator LEDs
- ✔ 12 VDC input power supply

HMI

- ✔ 4.3" color graphic LCD (non-touch-screen as the typical operator wears heavy gloves)
- ✔ Excellent graphics capabilities supporting a modern, brand-aligned GUI
- ✔ High-level GUI development tools to simplify design and deployment
- ✔ Built-in data movement capabilities to coordinate GUI control/data with the rest of the system
- ✔ Upgradability over-the-wire for feature additions and GUI refinements
- ✔ Compact form factor
- ✔ Ability to scale to larger displays without changing architecture

Table 1. To support modern features as well as future requirements, the DTX 2400 Series had an extensive list of requirements.

(*SIM231*) from Serious Integrated, Inc. The *SIM231* can dock directly on the *SCM208* to form an effective HMI plus communications plus control solution with plenty of headroom for future needs.

The *SCM208* communications and control module is equipped with a Renesas *S7G2* MCU and includes a complete commercial OS and TCP/IP and USB stacks from Segger. It also incorporates a set of frameworks and tools from Serious Integrated for out-of-the-box application-level development in C.

In the *DTX 2400* system architecture, the *SCM208* communicates with the Dynatronix power and I/O board (now only responsible for managing the power subsystem) over RS-485 using a custom protocol. The *SCM208* also uses GPIO to interface with a separate, simple PCB that manages a membrane keypad and other front-panel I/O. Given the electrical fields generated by the interior power systems of the *DTX 2400*, Dynatronix elected to bring an Ethernet PHY (rather than Wi-Fi) from the *SCM208* out to an RJ-45 MagJack connector on the simple PCB.

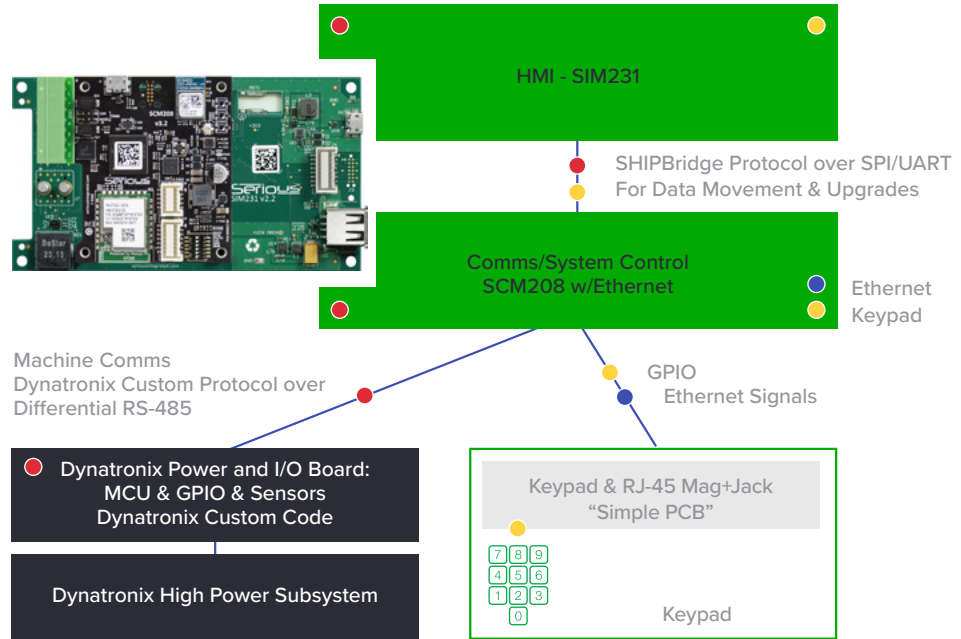
The *DTX 2400*'s HMI subsystem is handled by the *SIM231-A02*, a non-touch-screen 4.3" LCD variant of the *SIM231* HMI SOM. It includes high-level tools for rapid GUI development as well as support for the SHIPBridge data movement and over-the-wire update protocol.

The SHIPBridge protocol enables data transfer between the *SIM231* and *SCM208* over UART and/or SPI. For example, the *SCM208* scans and processes events via GPIO from the membrane keypad attached to the simple PCB, then transfers that key-press/release data

CASE STUDY

over SHIPBridge to the SIM231. Similarly, system status information from the power and I/O board (such as the voltage and amperage of Dynatronix' high-power subsystem) are transmitted to the SCM208 over a custom protocol, and then relayed on to the SIM231 using SHIPBridge. The SIM231 then drives GUI actions once data has been received.

Figure 6. The distributed subsystem architecture of the DTX 2400 Series DC rectifiers consists of intelligent bi-directional communications through a central communications/control hub, the Serious Integrated SCM208.



The SIM231 also uses SHIPBridge to communicate HMI-generated commands (such as requests to turn the power supply on or off) over UART or SPI to the SCM208. In turn, the SCM 208 passes these commands on to the power and I/O board using a custom protocol over RS-485.

As shown in Figure 6, the SCM208 is the hub of all of this activity, with data to and from DTX 2400 subsystems and external networks flowing through it. The architecture also makes it easy to expand system connectivity in the future. For instance, a web server stack could be dropped into the SCM208 and seamlessly share system status and control data with a remote web browser over Ethernet.

MOST NOTABLY, HOWEVER, the DTX 2400 Series architecture allows Dynatronix to focus development efforts on their secret sauce – the custom power supply control code running in their power and I/O board. It further enables the power and I/O subsystem to scale independently, with much of the other high-leverage engineering work being outsourced to other technology providers.

CASE STUDY

FROM ARCHITECTURAL TO OPERATIONAL

While the architecture provided a foundation, the next step was to get the DTX 2400 Series operational.

To kickstart development, Dynatronix engaged *Serious Integrated, Inc.* in a 10-week technology onboarding project. During this process, the *Serious Services* team provided scaffolding around the SCM208 and SIM231 platforms, which helped accelerate Dynatronix' development efforts while also giving them the control to fine tune the final product.

This scaffolding included:

- ✔ Creating the initial core data framework for moving data between the power and I/O subsystem and the HMI
- ✔ Building a custom task for communicating data and commands between the SCM208 and power and I/O board using Dynatronix' custom protocol over RS-485
- ✔ Connecting the SCM208 to the SIM231 through the SHIPBridge protocol
- ✔ Developing the GUI's initial pages
- ✔ Building the keypad task that translated the key presses (GPIO signals) into GUI commands

The onboarding project gave Dynatronix the ability to minimally control and monitor the power and I/O subsystem of the DTX 2400 Series from the HMI. The communications/control and HMI projects

for the SCM208 and SIM231 were also delivered to Dynatronix software engineers as part of the onboarding process, with training and support provided. In essence, the Serious Services team delivered complete end-to-end connectivity components for the various distributed elements of the DTX 2400 Series architecture.

FROM THERE, From there, Dynatronix proceeded to rapidly flesh out the capabilities of the system. For instance, Dynatronix engineers were able to quickly and easily finalize the complete HMI experience on the DTX 2400 Series front panel using the SHIPTide GUI development tool.

They also built on the initial C-code framework provided during onboarding to add keypad and communications tasks so that data could flow between the HMI and power and I/O subsystems. The latter enabled new ways for the SCM208 to not only direct the flow of internal messages, but also control the system's operation.

The DTX 2400 Series was fully functional within 6 months.

FEATURE CREEP WELCOME HERE

Initially, Dynatronix leveraged the hardware and software capabilities of the SCM208 so that its engineering team could focus on developing unique aspects of their system. But the company also realized that the platform and architecture enabled rapid future innovation in IoT and other types of connectivity. In this regard, The SCM208's pre-licensed Segger TCP/IP stacks instilled a high degree of confidence in their connectivity development path (Table 2).

During the DTX 2400 development cycle, as fate would have it, Dynatronix pre-sales customer engagements drove the addition of a major new feature in the product's initial release: Ethernet/IP.

Ethernet/IP is a widely used protocol that enables devices to connect to factory networks for coordination and control of multi-stage production lines. Indicators from the sales and marketing

Segger RTOS and stacks pre-integrated in the SCM208:

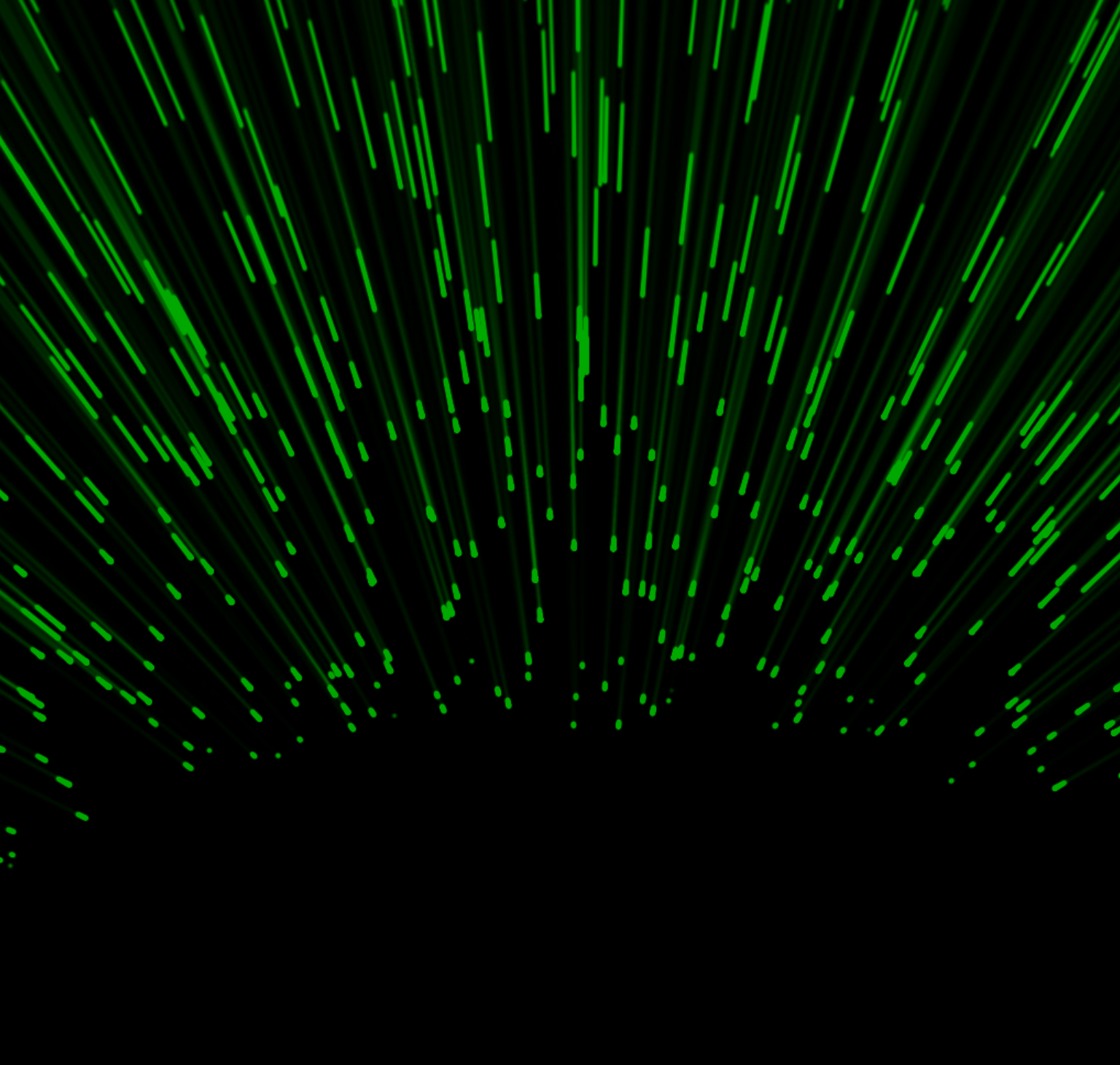
- ✔ embOS RTOS, which is pre-licensed and pre-ported to the SCM208
- ✔ embOS IP, a commercial TCP/IP stack that includes SSH, SSL, and TLS libraries
- ✔ emCrypt cryptography software
- ✔ emFile filesystem
- ✔ emUSB-device that supports file storage, data logging, and upgrades
- ✔ Fully unlocked commercial license to Segger Embedded Studio, a high-performance integrated development environment (IDE) for embedded C development and debug (SCM208 development kit)

Table 2. The SCM208 includes a complete suite of commercially supported, non-open source, pre-licensed, and pre-ported software and tools from Segger.

department suggested that the addition of Ethernet/IP in the DTX 2400 would significantly increase demand and likely displace numerous competitive products. The decision to move forward with the protocol was made.

The integration of a communications stack can be very time consuming and costly. Under normal circumstances, such an addition would either cause significant project delays or a complete architectural reevaluation if the existing MCU and/or memory resources were insufficient. However, with the distributed SOM-based architecture of the DTX 2400 Series, Ethernet/IP was exceedingly simple to integrate.

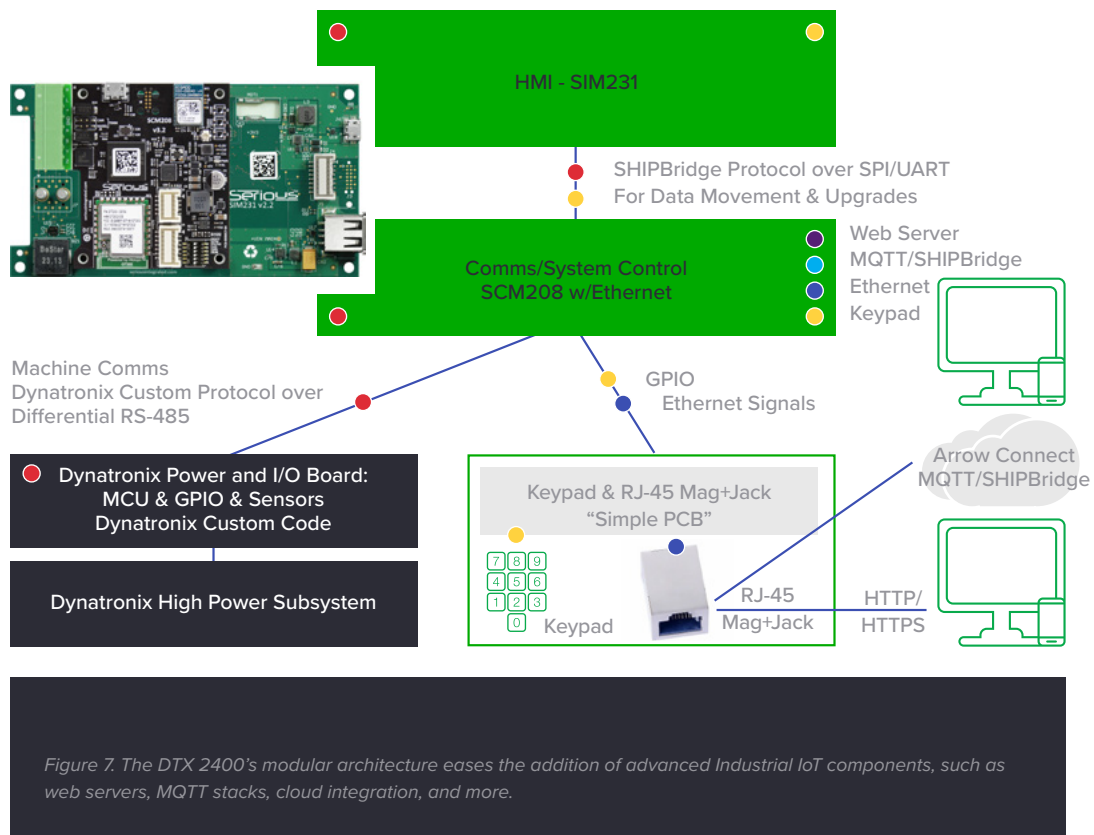
Consulting their SOM partner, Dynatronix was introduced to [Real-Time Automation \(RTA\)](#), a leading provider of industrial networking protocols and associated certification services. *Within two weeks* a functional Ethernet/IP stack was up and running on the SCM208. According to RTA it was the easiest port they had ever done because of the completeness and robustness of the SCM208's built-in Segger emTCP/IP stacks, Ethernet drivers and Serious Integrated software frameworks.



HUMAN-CONNECTED INDUSTRIAL MACHINES

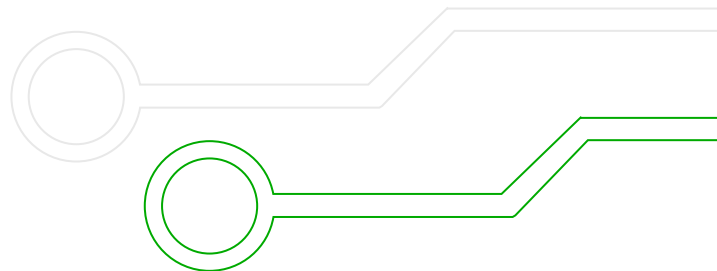
As demonstrated by the easy addition of an Ethernet/IP stack into the DTX 2400 Series, a significant amount of scale is possible in modern, modular system architectures. Web servers, MQTT stacks, and cloud integration with platforms such as [Arrow Connect](#) can be easily inserted or removed for advanced applications such as remote diagnostics, web-based monitoring, firmware upgrades, and predictive maintenance (Figure 7). For example, a PC on the same network could be connected directly to the DTX 2400 box for local monitoring and diagnostics; industrial networking protocols such as [PROFINET](#) and other PLC-type network connectivity can be added based on customer demand.

The DTX 2400 is an example of a platform that is IIoT-ready. It's fully functional today with the ability to migrate and scale with IIoT requirements of the future.



But the key result of this evolutionary leap in industrial machine architectures is not scalability, nor portability, nor reuse. It is the transformational effect on Dynatronix' engineering processes and time-to-market velocity. The independent subsystems can easily proliferate across various Dynatronix product lines, while the company invests the majority of its efforts in the creation of new features that run on the HMI and communications/control building blocks.

IN THE AGE OF IIoT, OEMs can now focus on innovation rather than the hardware and software infrastructure needed to support it.



The logo for 'SERIOUS' is rendered in a bold, green, sans-serif font. The letters are closely spaced, and the 'S' and 'I' have a distinctive horizontal bar extending from their top. A small 'TM' trademark symbol is positioned to the upper right of the 'S'.

SERIOUS™

Human Connected Machines

The background features a dark, textured surface with a pattern of fine, light-colored lines. Two large, bright green fiber optic bundles are visible, one in the upper right and one in the lower left, both appearing to converge towards the center. A white diagonal line bisects the image from the bottom left to the top right.

SERIOUSINTEGRATED.COM