FOUNDRIES.IO OFFERS INDUSTRY'S FIRST COMPLETE EDGE PLATFORM AS A SERVICE FOR IOT APPLICATIONS

EXECUTIVE SUMMARY

The first edge platform as a service (EPaaS) for IoT

In 2016, Softbank's Masayoshi Son compared the growth of IoT devices to the Cambrian explosion, referring to the biological Big Bang over 500 million years ago that resulted in our planet's incredible diversity of life. He predicted a trillion connected devices in 20 years. When he made that bold prediction, analysts reckoned that we'd have 30 to 75 billion connected devices by 2022. The actual number turned out to be 8 to 15 billion, depending on how analysts define "connected devices." This paper explains why IoT device deployments are not yet on a "trillion devices" trajectory and identifies the technology trend that makes Masayoshi Son's bold prediction more realistic. We then show how FoundriesFactory, Foundries.io's new and disruptive edge platform as a service (EPaaS), accelerates that trend.

Platform convergence is the disruptive trend that accelerates connected device deployment growth across the whole spectrum of embedded solutions. Converged platforms support multiple chip architectures, hardware designs, and OS configurations while providing a unified software environment that separates platform development from application development. Unified software environments transform IoT development from balkanized, DIY, system-focused mashups to scalable, modular, application-focused projects delivering predictable results using mainstream DevSecOps practices. This transformation follows a familiar pattern, similar to Windows and Linux on PCs and iOS and Android on mobile devices.

This paper begins by explaining edge platform convergence in a historical context, highlighting the characteristics of Foundries.io's EPaaS that power the trend. Then, I describe my hands-on experiences with FoundriesFactory on two commonly available IoT platforms – the Raspberry Pi and the new Arduino Pro Portenta X8. I confirmed that programmers with limited Linux kernel expertise can build product-tailored OSs and

begin writing containerized applications in a few hours. Finally, we summarize the benefits and cost-effectiveness of Foundries.io in a typical product lifecycle.

THE PLATFORM CONVERGENCE TREND

Plug-and-play software for IoT

Development complexity is the main reason most IoT projects fail to deliver expected results, and it also explains why connected device growth is so much slower than expected. IoT solutions are inherently complicated because engineers have to embed connected computing systems into real-world “things” that constrain every facet of product design – physical size, power consumption, connectivity, human interface, and overall cost. These constraints preclude using off-the-shelf computing platforms, forcing developers to customize embedded hardware and system software to meet unique solution requirements.

The IoT customization imperative results in a balkanized development model where every product requires unique platform software. Creating and maintaining these product-specific embedded software stacks is slow, expensive, and requires rare and costly engineering skills spanning hardware, OS, system software, embedded programming, security, and networking. System-level software customization lengthens time to market (TTM), increases project cost and introduces risk without adding product value. Applications generate value, not system code.

Why do IoT devices require system software customization while PCs, servers, smartphones, and mainframes do not? Mainstream computing platforms provide unified software development environments that standardize application development, enabling applications to run with little or no modification on many different hardware configurations. But this was not always the case. Each platform began as a diverse mixture of hardware architectures and software environments, then converged into a small number of broadly adopted ecosystems.

If this scenario sounds familiar, it’s because industry-wide platform convergence has happened three times in the past. In the 1960s, IBM created OS/360, enabling the same software to run on many different mainframe models. Twenty years later, Intel PCs and Microsoft OSs disrupted the diverse microcomputer industry with unified PC hardware and software specifications, then Linux did the same for servers. Twenty years after that, in 2007, iOS and Android created unified platforms for app development on Arm-based smartphones. In each of these three cases, independent software vendors (ISVs)

rapidly emerged and thrived, resulting in high-volume, high-quality, low-cost software solutions for mass markets while simultaneously accelerating hardware innovation.

TABLE 1: HISTORICAL PLATFORM CONVERGENCE DISRUPTIONS

Platform	Disruption Date	Hardware Ecosystem	Software Ecosystem
Mainframe	1960s	System/360	OS/360, MVS, VM
PCs, servers	1980s	Intel PCs	Windows, Linux
Smartphones	2000s	Arm Cortex-A	iOS, Android
IoT (Linux devices)	2020s	Arm Cortex-A, Intel, RISC-V, …	FoundriesFactory

Source: Moor Insights & Strategy

The first three platform convergence disruptions (mainframes, microcomputers, and smartphones) followed this five-phase sequence:

1. **Fragmentation** – A plethora of hardware and OSs evolve on new, immature computing hardware.
2. **Standardization** – Open and de-facto standards reduce fragmentation.
3. **Plug-and-play** – Unified, multi-vendor hardware and software environments emerge, reducing variability and accelerating application development.
4. **Defragmentation** – Darwinian forces unify the industry around a few plug-and-play platforms, usually two.
5. **Massive scale** – ISVs rapidly evolve on plug-and-play platforms.

IoT platform convergence is next, and it's already underway. It's following the same familiar pattern, currently entering the plug-and-play phase:

1. **Fragmentation** – Driven by IoT solution constraints, device platforms fragmented into thousands of unique hardware and software combinations.
2. **Standardization** – Standards that reduce platform variability are already available – OpenEmbedded Linux, LTS kernel, Yocto, OSTree, U-Boot, UEFI, OP-TEE, and others.
3. **Plug-and-play** – Mature software tools and unified OS distributions built on these standards are now available. Foundries.io is the first complete DevSecOps framework that assembles these standards and tools into a complete IoT software platform that disruptively accelerates product development.

Now that IoT platform convergence has reached the plug-and-play stage, developers can build applications that run on many different chip architectures and system

configurations with little or no customization. Foundries.io is leading the plug-and-play phase for the reasons discussed below. Welcome to the new world of plug-and-play IoT application development – efficient, predictable, scalable, secure, and maintainable. Perhaps we're now on the road to a trillion devices.

FOUNDRIES.IO EDGE PLATFORM AS A SERVICE

Transitioning the IoT from DIY to plug-and-play

Reducing device development complexity requires an off-the-shelf software platform that lets developers focus on creating revenue-generating applications. This is preferable to adding technical debt by creating and maintaining unique, undifferentiated system software stacks. IoT plug-and-play techniques unify IoT application development the same way iOS and Android unified mobile apps and Windows and Linux unified PC applications.

IoT plug-and-play platforms have been slow to emerge because of product design constraints imposed by the very nature of squeezing little networked computers into real-world "things." Constraints on size, power, computational capability, security, networks, hardware interfaces, human interfaces, ruggedness, reliability, over the air (OTA) updates, long-term support, and cost prevented developers from using general-purpose OSs as-is. The only alternative was to build custom versions of entire platform software stacks, including the OS, drivers, security, and OTA update infrastructure, before writing a single line of application code. Recognizing that IoT products always impose real-world platform limitations that require bespoke tailoring, Foundries.io developed a complete DevSecOps development environment that delivers the application development benefits of off-the-shelf OS distributions to the diverse world of IoT, including:

- **Customization** – Flexible platform software configuration with minimal impact and cost.
- **Security** – Enterprise-grade security built into the platform, not added on during development.
- **Incremental OTA updates** – Infrastructure for long-term support with over-the-air incremental updates.
- **Containerized, platform-independent applications** – Developers focus on applications, not OSs or platform code.

- **DevSecOps** – Continuous integration and deployment leveraging industry-standard open-source development tools such as Git, OSTree, LS, Yocto, U-Boot, UEFI, OP-TEE.
- **Hardware agnostic** – Build products with Arm, Intel, AMD, and other architectures with native support for many system-on-modules (SoMs) and (single board computers (SBCs).
- **Faster TTM, lower cost** – Focus on applications; reduce the need for deep in-house system expertise

The following sections explain how FoundriesFactory delivers these benefits as the industry's first functionally complete EPaaS , delivering a customizable Linux IoT platform as an affordable service.

CUSTOMIZATION

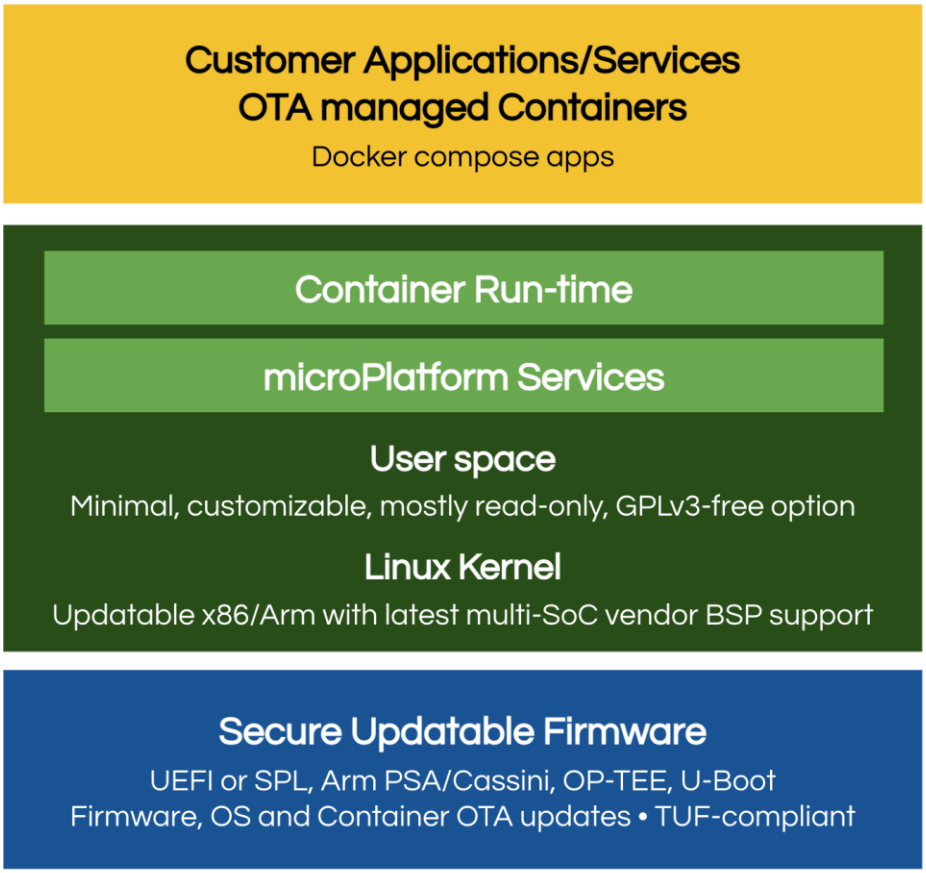
The practice of IoT customization has been an inconvenient necessity since the first embedded systems in the 1960s. It's certainly not a new problem, and it's not going away in the foreseeable future. Beginning about 20 years ago, the Open-Source community created automatic build systems such as Buildroot and Yocto that simplify creating Linux distributions for use on embedded devices. Foundries.io uses Yocto, which is ideal for IoT platforms because it supports continuous integration and uses a unique layer model for modular customization. But Yocto requires considerable Linux expertise, has a steep learning curve, and needs an environment with a complete set of software layers, including an OS core, security support (i.e., TrustZone), optional OS layers, board-support packages for all target devices, and IoT-friendly update services. Foundries.io flattens the Yocto learning curve and automates most of its functions within a Git-based development environment.

The Foundries.io Linux microPlatform (LmP) is a Yocto-based Linux OS developed for IoT and edge devices. Yocto configures LmP to meet the needs of specific hardware platforms and IoT products using "recipes" that customize OS builds by organizing custom code into "layers" such as:

- OpenEmbedded Core build system.
- OpenEmbedded kernel and images.
- OpenEmbedded utilities (with network support).
- OSTree and TUF / Uptane (for OTA updates).
- Linaro's OP-TEE trusted execution environment.
- Meta layers for vendor-specific modules and boards.

Yocto configures LmP using only the layers and packages that the target application needs. You'll find a complete description of LmP here. As shown in Figure 1, a solid foundation of secure, standards-based, Arm PSA-certified, updateable firmware supports the Linux kernel, customizable (GPLv3) user space, platform services, and the container run-time environment. Containerized customer applications plug-and-play on top of this stack.

FIGURE 1: FOUNDRIES.IO'S LINUX MICROPLATFORM (LMP)



Source: Foundries.io

FoundriesFactory has all the packages required for chip support, board support, and security already installed for commonly used hardware platforms. So, LmP builds correctly for these chips, modules, and boards with no need for customization by product developers. FoundriesFactory integrates with DevSecOps workflows to automatically adapt to the constantly changing hardware and system software environment, allowing product developers to focus on application code instead of system code.

SECURITY

Getting security right earns manufacturers a checkmark. But getting it wrong is a huge miss that impacts customer relationships and brand reputation. Hence, every solution provider is motivated to deliver high levels of security. However, security seldom generates direct revenue because customers rightly expect it to be “standard equipment” in every product. This dilemma requires a systemic approach. Every company in the IoT supply chain, from silicon to finished goods, must build well-defined levels of security into all hardware and software components from the start. Product developers cannot add security as an afterthought. Here are three ways that FoundriesFactory simplifies building trustworthy IoT devices.

First, every product component must be certifiably secure – silicon, bootstrap, OS, trusted execution, identity, and external connections. The Foundries.io Linux microPlatform conforms with Arm Platform Security Architecture (PSA), Arm SystemReady, and Intel’s Platform Trust Technology (PTT) programs. Several chips and boards are already certified for use with LmP, and more are on the way. Security starts with a root of trust during system boot, attested using signed certificates with keys installed during manufacturing or in the field. For Arm chips, Linaro’s OP-TEE provides a trusted execution environment. Farther up the stack, LmP securely authenticates to public clouds (Azure, Google Cloud, AWS) and uses WireGuard to create secure VPN tunnels.

Second, even the best software has vulnerabilities that require patching. The most secure software is the latest version, so suppliers must provide over-the-air updates throughout a product’s lifetime. Few companies do this correctly because the infrastructure for updating fleets of embedded devices over the air is complicated and costly. FoundriesFactory provides robust OTA update procedures that keep devices secure through long product lifecycles. Refer to the Incremental Updates section below for more details.

Third, security talent is specialized, expensive, hard to recruit, and often harder to retain. Although every IoT development project needs software engineers with security knowledge and experience, FoundriesFactory enables application developers who are not security experts to build trustworthy platforms. FoundriesFactory also eliminates the high cost and complexity of building OTA update procedures and infrastructure. In short, delivering security as part of an EPaaS solution frees developers to focus on value-added development.

INCREMENTAL OTA UPDATES

Ideally, IoT devices should receive updates over the air just like smartphones, with separate update processes for the OS and apps. Applications frequently deliver new features and bug fixes, so application development tools should continuously integrate and deploy updates (CI/CD) as part of a unified DevSecOps practice. OS updates happen less frequently and with stricter requirements for testing and deployment. Secure, robust firmware updates for apps and OSs are integral parts of FoundriesFactory. All updates are incremental, containing only the changed components determined by code check-ins as part of a shared DevSecOps environment. The Factory also orchestrates updates to fleets of devices by using "device groups" to schedule sets of devices as a single block. "Waves" control the sequence of updates to device groups, enabling staged rollouts so that unforeseen problems detected on early waves don't propagate to the entire fleet.

CONTAINERIZED, PLATFORM-INDEPENDENT APPLICATIONS

FoundriesFactory separates developing applications from building OSs and writing platform code. App developers can use Docker Compose to build Compose Apps that the Factory manages from a repository (containers.git). The OTA update mechanisms described above automatically deliver Compose Apps from the repository to IoT devices.

This process solves one of the most significant barriers to IoT growth – creating an application environment that works more like an app store instead of managing applications as part of the system image. FoundriesFactory lets developers focus on applications, use mainstream languages and tools, and update devices frequently without destabilizing embedded OSs or system code.

DEVSECOPS

At the highest level, FoundriesFactory is a cloud-based DevSecOps framework for IoT. The framework uses familiar, open, industry-standard development tools such as Git and Docker, so software engineers are immediately comfortable and productive. The role-based framework supports Agile teams by continuously integrating each git commit, managing layered customization (no need to fork the core platform), and tagging releases to enable parallel work across multiple branches.

HARDWARE AGNOSTIC AND OPEN SOURCE

FoundriesFactory can build distributions for just about any Linux computing platform, including Arm, Intel, AMD, and many modules and boards. You'll find the complete list here.

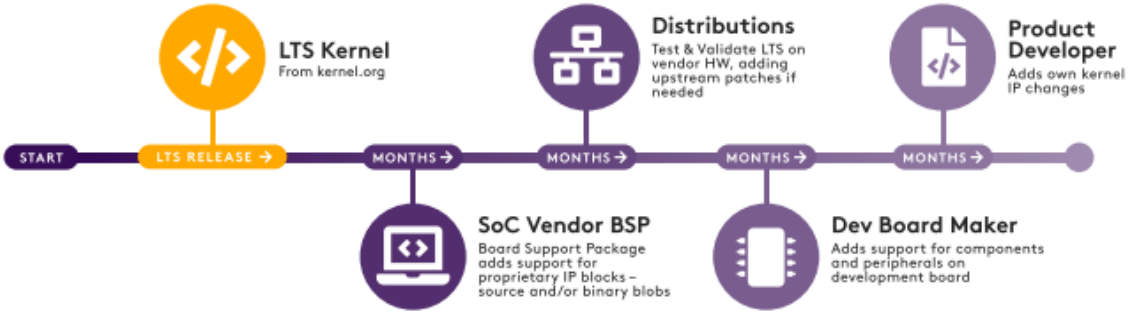
FoundriesFactory is not a "black box" development system. The code is open-source, built on industry-standard components, and does not lock projects into specific chips, modules, or boards. It's about as future-proof as you can get. Here are some of the standard components used by FoundriesFactory.

- OS configuration – Yocto, BitBake, OSTree.
- Agile DevSecOps – Git, Docker.
- Containers – Docker, Docker Compose, Containerd.
- Device fleet management – APIs and CLI.
- Managed incremental updates – TUF-compliant.
- Secure boot – U-Boot, UEFI.

FASTER TTM, LOWER COST

For product companies, the most impactful benefits of FoundriesFactory are speed and efficiency – faster TTM and lower cost. The Factory's application-focused approach eliminates unnecessary platform development, reduces the need for deep in-house OS and security expertise, supports Agile development best practices, and enables container-based applications. But FoundriesFactory can also fix a big problem with how Linux modifications propagate through the embedded software supply chain, removing another obstacle to the "trillion devices" future.

FIGURE 2: TYPICAL EMBEDDED LINUX SUPPLY CHAIN

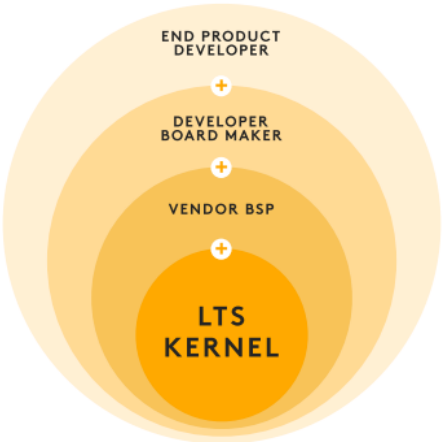


Source: Foundries.io

The long-term support (LTS) kernel is the best foundation for embedded Linux platform stacks. However, as shown in Figure 2, kernel changes from silicon (SoC) vendors, Linux distributions (Ubuntu, SuSe, Red Hat, others), and board makers are required before product developers start adding product-specific code.

Each step along the chain further diverges from the original LTS kernel, adding layers of changes that developers should push upstream for eventual integration into the LTS kernel. Unfortunately, many vendors are slow to push changes up to LTS, and some never do. Figure 3 shows that the product developer is ultimately responsible for building, testing, and maintaining a unique stack of kernel modifications from multiple vendors. These changes often involve large amounts of code (graphics and AI, for instance) and take many months to propagate down the chain to the product developer. Using the latest LTS kernel as-is does not work for most embedded platforms because it's unlikely to have the required vendor-specific software.

FIGURE 3: KERNEL MODIFICATION HIERARCHY



Source: Foundries.io

This situation is known as the "snowflake problem" – every product is a unique system platform. It's a big reason that the IoT is not scaling predictably and that embedded security and software quality often disappoint.

PCs and servers don't have this problem, mainly because the variability from one platform to another is easier to manage. Linux evolved organically with built-in logic for adapting builds to broadly adopted PC architectures, hardware features, and peripheral configurations. So, enterprise Linux distributions already have all the support needed for

most system configurations – Intel, AMD, and Arm architectures, plus thousands of vendor hardware options and peripherals.

The snowflake problem is unique to embedded (IoT) products because the LTS kernel cannot rapidly adapt to the unlimited variety of IoT SoCs, modules, boards, and custom product configurations. Hence, the IoT platform software supply chain must manage cumulative vendor changes spanning silicon, OS distros, modules, and boards. FoundriesFactory insulates product developers from IoT supply chain complexities by maintaining the whole core platform as part of the EPaaS subscription, including vendor packages for supported chips and boards. Foundries.io collects and distributes vendor-specific kernel modifications, ensuring that product developers automatically get the correct vendor software in every build. For product developers, independent development methods for applications and platforms enable optimized DevSecOps methods with unique CI/CD workflows for each type of software.

From a strategic point of view, insulating IoT application development from system dependencies lets app developers focus on apps, not OSs. Specialized vertical application development is a game-changer because it creates an independent software vendor (ISV) business model for IoT, similar to PCs and smartphones, leading to significant reductions in total solution cost and TTM.

FOUNDRIESFACTORY HANDS-ON

Can an application programmer build a customized, secure OS in a few hours?

Intrigued by the Foundries.io technology and value proposition, I decided to see for myself if a rusty old application programmer like myself could use FoundriesFactory to build and deploy a customizable OS distribution, complete with OTA updates, and start writing containerized applications in a few hours. I figured if I could do it, any programmer could. Here's a log of my experience with FoundriesFactory on two different hardware platforms – a Raspberry Pi 3 and the new Arduino Pro Portenta X8.

FOUNDRIESFACTORY ON THE RASPBERRY PI 3

Foundries.io offers a free trial that lets customers use FoundriesFactory on a single hardware platform for one month at no cost. That's plenty of time to learn the basics of the platform, build an OS for off-the-shelf SoC evaluation kits (EVKs), single-board computers (SBCs), or system-on-modules (SOMs), and prototype real-world applications. FoundriesFactory has built-in support for over thirty popular EVKs, SBCs

and SoMs. I decided to start with a Raspberry Pi 3 because it's simple and ubiquitous. I had no prior knowledge of FoundriesFactory other than high-level product descriptions and marketing materials. My objective was to see how much I could learn by working through the tutorials and writing some trivial Docker applications.

I used a clean Windows 11 laptop as a development system. Before starting, I installed all the software and tools needed to cruise through the tutorial without stopping, like binging a Netflix series. I installed these six packages:

- WSL – Windows Subsystem for Linux.
- Git – Version control.
- Fioctl – FoundriesFactory management tool.
- Rufus – Create bootable flash drives.
- Docker Desktop – Application containers.
- Source code editor – (Your choice. I used VS Code).

Here's a log of my experiences with the Foundries.io tutorials.

1. **Sign up** as a new user on Foundries.io. No credit card is required.
2. **Create a new Factory** and pick a supported platform. I selected the Raspberry Pi 3.
3. **Configure Git**. Generate and install FoundriesFactory source access tokens.
4. **Configure fioctl**. Generate API token (OAuth2 credentials) used by fioctl to authenticate with the back end.
5. **Flash a bootable SD card**. The Factory automatically builds the first system images. Watch the builds on the "Targets" tab.
6. **Boot the SBC** (RPi in my case) using the flash drive. Connect with Ethernet (or Wi-Fi). Access the new platform with ssh and a keyboard/monitor.
7. **Register the new device** to FoundriesFactory. The device appears in the Factory's device list under the Devices tab.

Milestone: I built a customizable OS and booted it on the RPi. The next step is to build a simple Docker application.

8. **Clone containers.git**, the Docker compose repository. The files appear on the PC file system in a "containers" folder under the home folder.
9. **Edit an application** to personalize it. I just used a simple shell script.
10. **Create a Dockerfile** that copies the app into a Docker image and specifies the entry point.

11. **Build the Docker container** and run it on the development PC (Docker Desktop). The tutorial walks through these steps.
12. **Create a docker-compose.yml**, a YAML file that configures application services so Docker can start them with a single command.
13. **Commit and push the application** to the Factory repository using git. The Factory starts building the new target automatically, then updates the device with the new application.
14. **Test the application** running in a Docker container on the target device over the network using curl.

Milestone: I built a simple application, pushed it to the Foundries repository, and tested it. The Factory built the new target (update package) with the application and deployed it to the device via OTA update.

15. **Learn more about aktualizr-lite and fioconfig.** The tutorial walks through an introduction to these device-side FoundriesFactory management daemons. Aktualizr-lite checks for new updates and applies them using The Update Framework (TUF), while fioconfig manages device configuration data. Both applications communicate with the Factory to control the OTA update process.

Results: I configured a customizable OS with FoundriesFactory, booted it on an RPi 3, built a simple application, packaged it with Docker-compose, and pushed it to the Factory. The push triggered an automatic target build, followed by an OTA device update. I then ran the Docker application on the RPi. I also learned about the two key device management daemons.

This exercise was a complete success. The tutorial walked me through the key features of FoundriesFactory, providing a solid base for further exploration. My total time investment was one afternoon, which could have been shorter if I hadn't strayed from the tutorial a few times to learn more about interesting aspects of the platform. I walked away with enough practical knowledge to begin writing and deploying simple applications on the RPi – precisely what I set out to do. Although I've written a lot of code in my career, I haven't been a professional programmer for a long time. If I can get this far in a few hours, I'm confident that any software engineer can do the same. FoundriesFactory is a big system, so I don't want to imply that anyone can be an expert in a day. But the platform is solid, it's built on familiar tools, the documentation is good, the source is open, and it all works.

FOUNDRIESFACTORY ON THE ARDUINO PORTENTA X8

FoundriesFactory's enterprise-grade features leverage trustworthy devices to achieve very high levels of security. Arduino recently announced a trustworthy Linux platform, the Arduino Pro Portenta X8, based on the NXP® i.MX 8M Mini. The X8 has industrial-grade security built-in with Common Criteria EAL 6+ compliance and PSA certification up to the OS level. You'll find details here.

The Portenta X8 is a surprisingly powerful platform with a quad Cortex-A53 running Linux, 2 GB of DDR4 DRAM, and 16 GB of onboard flash. The NXP SE050 secure element chip provides a hardware root of trust and comprehensive crypto features. The X8 also has a Cortex-M4 microcontroller as a co-processor for real-time applications. The M4 side is compatible with other Arduino boards and programmed with the familiar Arduino IDE. But here's the best part. The Linux side of the X8 uses FoundriesFactory to provide a full range of security features – bootstrapping the OS, running secure applications, and communicating with cloud services. Enhanced security, combined with the other FoundriesFactory operational benefits described in this paper, makes the X8 an excellent choice for enterprise edge applications.

FIGURE 4: ARDUINO PRO PORTENTA X8 SOM – 66 MM X 25.4 MM

Source: Arduino

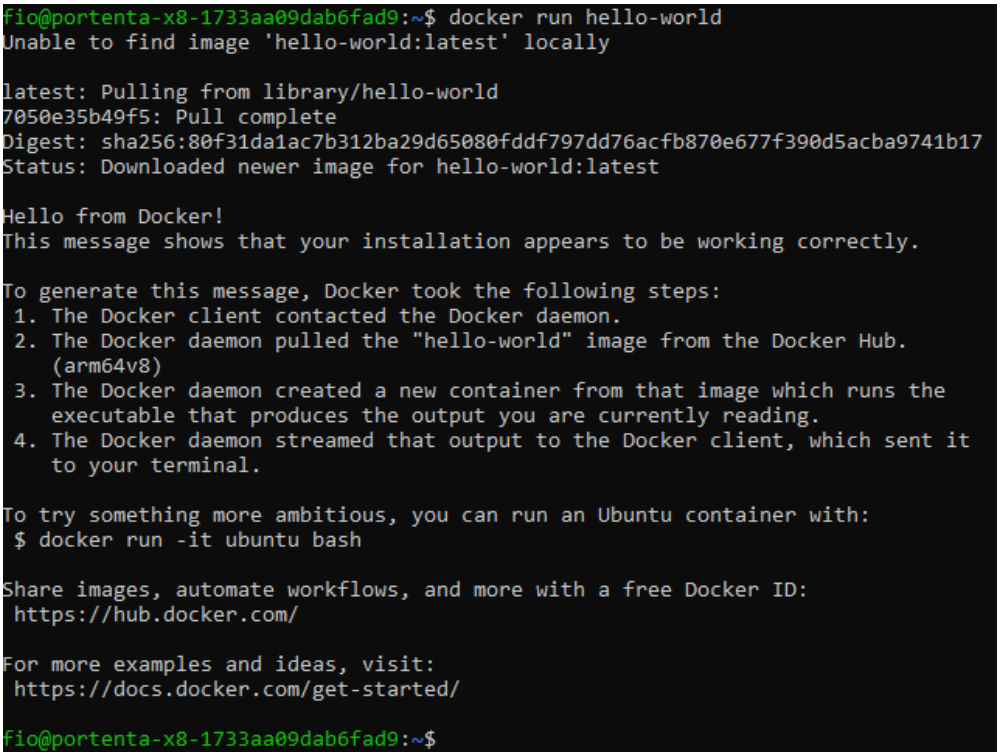
Here's a summary of my experiences with the X8. Please refer to the "Portenta X8 Getting Started" tutorial.

I connected the X8 to my development computer using USB, and it booted right up. The next step is to configure Wi-Fi, and a slick UI makes that easy. At this point, the microcontroller side of the X8 is programmable using the familiar Arduino IDE. The X8 works like other microcontroller-powered Arduino boards, so I wrote a simple sketch and ran it on the X8. So far, so good.

Milestone: I can use the Arduino IDE to write sketches for the microcontroller side of the X8.

The next step is to fire up the X8 Linux environment. That's easy because the X8 ships with Linux already installed. I logged in with SSL and found a familiar Linux environment with Docker already running. I started pulling Docker containers right away, starting with "hello-world," of course. The command "docker run hello-world" pulls that container from Docker Hub in the cloud and runs it. As you can see from the console log in Figure 5, this could not be any easier.

FIGURE 5: DOCKER ON PORTENTA X8 LINUX, RIGHT OUT OF THE BOX



Source: Moor Insights & Strategy

Milestone: I used Linux with Docker right out of the box on the X8.

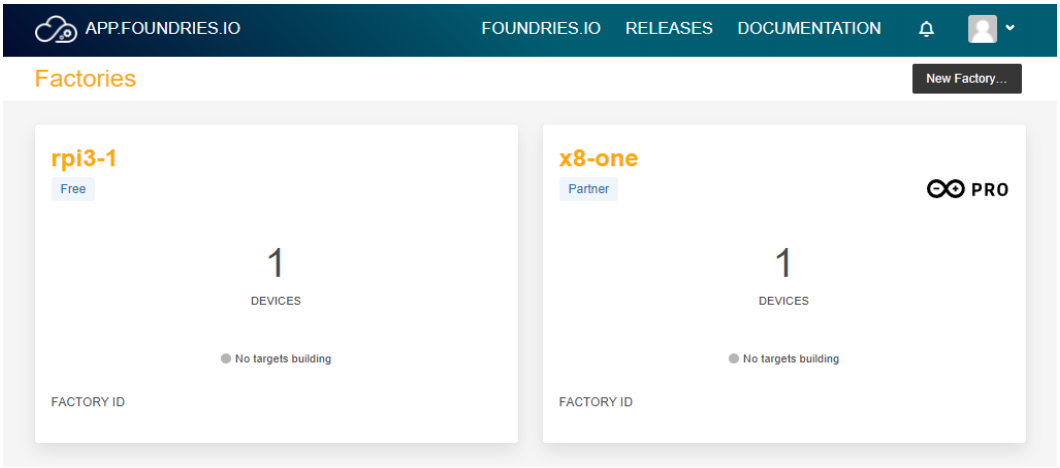
Developers and experimenters using the X8 for non-commercial applications can stop here. But if you're a professional developer building secure, maintainable, commercial edge products, you'll want to take the next step – FoundriesFactory integration.

There are two ways to do this. If you already have a FoundriesFactory trial account (free for a month) or are already a subscriber, you can create a new Factory with the X8 as the target platform. Your new X8 Factory environment works the same as other supported hardware platforms like the Raspberry Pi. You're all set for commercial product development, including Docker apps, system customization, enterprise-grade security, OTA updates, and long-term support.

Another option is establishing an Arduino Cloud account and managing the X8 using a "partner" Foundries.io account. You'll need an Arduino Cloud subscription. At the time of this writing, it's $350 per month, a fraction of the Foundries.io subscription cost. However, "partner" Foundries.io subscriptions do not include platform customization. Although Arduino Cloud users can't change the Linux platform itself, FoundriesFactory provides application development with OTA application updates, security support, OS patches, and OTA platform updates. This option is great for building scalable, maintainable applications for the X8.

In summary, there are two options for using FoundriesFactory on the X8. (1) Use a standard FoundriesFactory subscription (free trial available) for application and platform development, or (2) use FoundriesFactory with an Arduino Cloud subscription, limited to application development. I tested option (2). Linking FoundriesFactory to Arduino Cloud was easy, and both of my Factories, the Raspberry Pi 3, and the X8, appeared on my FoundriesFactory dashboard, as shown in Figure 6.

FIGURE 6: FOUNDRIESFACTORY DASHBOARD WITH RASPBERRY PI AND ARDUINO PRO PORTENTA X8



Source: Moor Insights & Strategy

Milestone: I connected FoundriesFactory to the Arduino Cloud, the Factory updated the X8's OS, then I built and ran simple Docker applications.

Secure, full-featured, general-purpose EVKs and SoMs such as the Arduino Portenta Pro X8 have the computing power and flexibility to address a broad spectrum of solutions. The FoundriesFactory EPaaS takes these platforms to the next level by delivering a highly productive software development environment with commercial product support –automatic OS and vendor platform updates, customization, security, and OTA updates. FoundriesFactory is a speedy and safe road to a complete, enterprise-grade edge solution.

SUMMARY

Product lifecycle workflows and benefits

FIGURE 7: FOUNDRIESFACTORY IOT PRODUCT LIFECYCLE MANAGEMENT

Source: Foundries.io

PRODUCT LIFECYCLE WALK-THROUGH

FoundriesFactory is a managed environment for designing, developing, testing, deploying, and maintaining IoT device software. Figure 7 shows the parallel, iterative product lifecycle management workflow.

Development projects begin by selecting reference hardware. Off-the-shelf EVK, SBC, and SoM boards enable developers to build software on proven foundations without waiting on hardware development. My evaluations confirm that it's easy for application developers to configure commodity hardware, get it working with FoundriesFactory, and start writing applications for the selected reference board. Meanwhile, engineers build custom hardware prototypes in parallel, with solution-specific sensors, actuators, user interfaces, power sources, networks, and physical configurations. FoundriesFactory makes it easy to switch or modify hardware platforms late in the development cycle as prototypes evolve into production hardware – there's no vendor or device lock-in. Hardware and software are then continuously integrated and deployed (CI/CD), iteratively converging on solid commercial solutions. CI/CD cycles continue after deployment to add features, improve performance, and fix bugs.

Deployment begins by creating product manufacturing workflows. This process is easy if the manufacturer already supports the security features of the hardware and the software stack – key and identity management, secure boot, and OS installation. Industry certifications like PSA and SystemReady, combined with native firmware support from chip and board manufacturers, significantly decrease (or eliminate) the need for custom, product-specific manufacturing procedures. After manufacturing, deployed products connect with onboarding services that establish secure application connections. Onboarding is easy if the platform already supports the required procedures, but developing and testing these capabilities from scratch can be very costly. After deployment, IoT products enter a long-term maintenance phase, where incremental application and platform updates happen over network connections (OTA). Managing updates to fleets of devices is a complex problem requiring interfaces with the development CD system, secure firmware download, attestation, and fleet orchestration. FoundriesFactory already has these deployment features built-in.

SUMMARY OF BENEFITS

FoundriesFactory delivers many benefits to companies across the IoT supply chain – silicon, modules, boards, ODMs, ISVs, and OEMs. From an IoT OEM perspective, here are the most impactful benefits of using FoundriesFactory:

- Reduce product development costs and shorten TTM.
- Reduce IoT platform customization costs.
- Build value-added applications, not undifferentiated OSs.
- Begin application development and hardware prototyping immediately after selecting a reference platform.
- Reduce reliance on rare and costly OS and security experts.
- Use best-practices DevSecOps methods and tools.
  Avoid unique embedded techniques.
- Use customizable off-the-shelf platform software.
  Avoid full-stack embedded development.
- Use platforms that are already security-certified.
  Avoid DIY security.
- Use an existing, proven, incremental OTA device update service.
  Avoid DIY update infrastructure.
- Use existing manufacturing workflows.
  Avoid unique, proprietary manufacturing steps.
- Control costs – EPaaS subscription with no royalties or per-unit fees.
- Reduce undifferentiated friction across IoT supply chains.

In addition to these benefits, FoundriesFactory delivers advantages across the whole IoT supply chain:

- **Silicon, module, and board makers** – Enable customers to begin development immediately, customize rapidly, and deploy efficiently. Accelerate downstream board support package availability and adoption.
- **ODMs** – Simplify OEMs product prototyping, customization, and deployment. Reduce solution TTM and cost.
- **ISVs** – As the industry moves to the EPaaS model, IoT ISVs emerge to capitalize on new plug-and-play software opportunities, dramatically increasing IoT software availability and quality. The same thing happened with PCs and smartphones.
- **Enterprises and end-users** – Customers reap all of these benefits, resulting in IoT solutions that are cheaper to buy, cheaper to own, higher quality, and more secure while also offering a much wider variety of solution software.

The bottom line is significantly reduced friction across the IoT device supply chain, making the “trillion devices” future more realistic.

CONCLUSION

Plug-and-play software arrives at the edge

Real-world IoT product designs constrain embedded device computing platform size, power, and cost. These constraints usually rule out mainstream, off-the-shelf OSs, forcing developers to customize, secure, build, maintain, and update each product's OS and system code. Platform customization is costly, and it diverts development resources away from application development without adding any customer-visible value. The resulting undifferentiated development friction is the most common reason IoT projects fail to deliver expected results, and it's why industry-wide IoT growth lags far behind projections.

Although IoT devices always require some customization, there's now a better way to do it. Foundries.io offers a much more efficient alternative to creating bespoke full-stack software platforms. FoundriesFactory is the industry's first complete EPaaS (embedded platform as a service). This fixed-cost, cloud-based subscription service delivers a secure, customizable Linux platform with over-the-air updates, device fleet management, containerized applications, and a DevSecOps environment built on mainstream software tools, all for a fraction of the cost of a single embedded software engineer. The EPaaS approach transforms IoT development from an esoteric process requiring deep OS and security expertise to a higher-level software engineering task comparable to PC and mobile application development. This transformation is evidence of IoT's maturity – shifting product development focus from building unique system stacks for each product to writing applications that plug-and-play on commercially supported platforms, fundamentally changing the economics of IoT product development.

The time is right for IoT plug-and-play. New hardware platforms such as the Arduino Pro Portenta X8 have the compute power, security components, and firmware support to run a secure, customized Linux OS hosting componentized, cloud-native applications. Security certifications (i.e., PSA and SystemReady) plus incremental OTA updates ensure trustworthy, long-lived products. And using FoundriesFactory enables application developers without deep system and security expertise to build secure device stacks. I verified this claim for myself on two SBCs.

Insatiable global demand for edge intelligence across every vertical market is driving product companies to accelerate embedded development, exacerbating the device development bottleneck. It's time to stop thinking about IoT devices as specialized,

embedded gadgets with unique software stacks, realize that the IoT is on an evolutionary path similar to PCs and smartphones, and start developing applications accordingly. We will look back on 2022 as the start of the IoT plug-and-play era.

IMPORTANT INFORMATION ABOUT THIS PAPER

CONTRIBUTOR
Bill Curtis, Analyst In-Residence, Industrial IoT and IoT Technology

PUBLISHER
Patrick Moorhead, CEO, Founder and Chief Analyst at Moor Insights & Strategy

INQUIRIES
Contact us if you would like to discuss this report, and Moor Insights & Strategy will respond promptly.

CITATIONS
This paper can be cited by accredited press and analysts but must be cited in-context, displaying author's name, author's title, and "Moor Insights & Strategy". Non-press and non-analysts must receive prior written permission by Moor Insights & Strategy for any citations.

LICENSING

DISCLOSURES
Foundries.io commissioned this paper . Moor Insights & Strategy provides research, analysis, advising, and consulting to many high-tech companies mentioned in this paper. No employees at the firm hold any equity positions with any companies cited in this document.

DISCLAIMER
The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. Moor Insights & Strategy disclaims all warranties as to the accuracy, completeness, or adequacy of such information and shall have no liability for errors, omissions, or inadequacies in such information. This document consists of the opinions of Moor Insights & Strategy and should not be construed as statements of fact. The opinions expressed herein are subject to change without notice.

Moor Insights & Strategy provides forecasts and forward-looking statements as directional indicators and not as precise predictions of future events. While our forecasts and forward-looking statements represent our current judgment on what the future holds, they are subject to risks and uncertainties that could cause actual results to differ materially. You are cautioned not to place undue reliance on these forecasts and forward-looking statements, which reflect our opinions only as of the date of publication for this document. Please keep in mind that we are not obligating ourselves to revise or publicly release the results of any revision to these forecasts and forward-looking statements in light of new information or future events.