O'REILLY®

Compliments of thingworx®

# Foundational Elements of an IoT Solution

## The Edge, The Cloud, and Application Development

Joe Biron &
Jonathan Follett


ThingWorx is purpose-built for the Internet of Things, with tools, APIs, and marketplace extensions that lower costs, increase developer productivity, and speed time-to-market.

With the ThingWorx IoT Platform, you have access to a powerful development engine and a broad set of innovative technologies that extend the power of the IoT:

- CONNECT to any Thing
- CREATE Apps for all Users
- ANALYZE Machine Data
- EXPERIENCE all Things through Augmented Reality

Learn more about how the ThingWorx IoT Platform is the right choice to power your organization's digital transformation.

http://www.thingworx.com/go/IoTFoundations

ptc

# Foundational Elements of an IoT Solution

## *The Edge, The Cloud, and Application Development*

*Joe Biron and Jonathan Follett*

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®

**Foundational Elements of an IoT Solution**
by Joe Biron and Jonathan Follett



Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.



**Editors:** Susan Conant and Jeff Bleiel
**Production Editor:** Kristen Brown
**Copyeditor:** Colleen Toporek
**Interior Designer:** David Futato
**Cover Designer:** Karen Montgomery
**Illustrator:** Rebecca Demarest

March 2016: First Edition

**Revision History for the First Edition**
2016-03-30: First Release



978-1-491-95097-5

[LSI]

# Table of Contents

# CHAPTER 1
# Introduction

The Internet of Things (IoT) has a rich technological legacy and a bright future: ubiquitous connectivity has created a new paradigm, and the closed, static, and bounded systems of the past will soon be obsolete. With the connection of low-cost sensors to cloud platforms, it's now possible to track, analyze, and respond to operational data at scale. The promise of the IoT is indeed wonderful: intelligent systems made up of smart machines that talk with each other and with people in real time, and data analytics driving optimization and transformation in industries as varied and far-reaching as aeronautics and agriculture, transportation and municipal services, manufacturing and healthcare, and even within our homes.

## Building the Internet of Things

The Internet of Things presents exciting opportunities to transform business, but the specific approaches and patterns remain somewhat ill-defined. So, maybe it's not entirely surprising that the recent tidal wave of marketing hype has engendered some well-deserved skepticism about the IoT's true business and social value. Questions about security and fears that such wide-ranging connectedness will make privacy all but extinct are commonplace. These are legitimate issues that are being addressed, and will require continuing maturity of both the business and technology factors if the IoT is to achieve long-term, broad-based success.

Regardless, it's clear that, in order to take on the challenges of design for this new connected world, engineers, designers, technologists,

and business people need to fundamentally shift their thinking. IoT design will be quite different from design for other complex systems; data will be the critical material, shared across open and flexible networks. Making the most of IoT for your business requires strategic thinking and careful planning.

If you don't quite know where to start with the IoT, you've come to the right place. This guide is for those who have heard both the grand promise and the skeptical inquiries and nevertheless want to get their boots on the ground. The guide introduces you to the high-level concepts, components, and patterns for any type of IoT solution. It will help you to understand the technology and architecture, so that you, the technologist, can dispel misconceptions within your organization and assess the opportunities for the IoT to advance your business. The potential of the IoT may well be limitless—but in order to get to that promise, we need to get started.

## What This Guide Is Not

You'll find a bevy of other IoT primers on the websites of technology vendors, standards groups, and industry consortiums, many of them extremely insightful, but all slightly biased towards either a technology or philosophical premise about how the IoT should work. There isn't anything wrong with these sources, and you are encouraged to check out what they have to say, but the goal of this guide is to provide you with the real-world tools and patterns that are in use, or on the near-term horizon, based on practical hands-on experience in hundreds of IoT solutions over the last decade. This guide is about what works for the IoT today and what the considerations are for implementing something right now.

## A Technologist's Definition of the IoT

In 1999, Kevin Ashton of the Massachusetts Institute of Technology (MIT) coined the term *Internet of Things*. At the time, industrial automation technologies were starting to move from the factory into new environments like hospitals, banks, and offices. This early form of intercommunication often involved machines of the same type—such as a one ATM machine talking to another in the same general location—hence the term, Machine-to-Machine, or M2M. As early M2M implementations grew increasingly more sophisticated, machines were connected to other kinds of devices like servers.

Those servers ultimately moved from on-premise locations into data centers and eventually "the cloud."

We can appreciate the prescience of Kevin Ashton's term. Yet while the "IoT" is a catchy phrase, it doesn't help us understand the full implications of this new paradigm. While the Internet is, of course a critical, enabling element, it is only a part of the essential concept—the idea that we can connect our reality, part and parcel, to the virtual world of information systems—that is so truly transformational for smart connected products and operations alike.

Today, the Internet of Things can include industrial and commercial products, everyday products like dishwashers and thermostats, and local networks of sensors to monitor farms and cities. In an IoT solution, objects can be sensed and controlled through the Internet, whether these objects are remote devices, smart products, or sensors that represent the status of a physical location. And information can be made available to applications, data warehouses, and business systems.

## Guide Outline

For some developers, the IoT may seem like a mishmash of technologies arranged in a bewildering set of combinations. It's true that this is an area where embedded computing, MEMs, broadband and mobile networking, distributed cloud computing, advanced distributed database architectures, cutting-edge web and mobile user interfaces, and deep enterprise integrations all converge. But thankfully there are some clean layers that we can use to inform our mental model of IoT solutions.

Our guide is divided into four chapters:

*Chapter 2, Solution Patterns for the Internet of Things*
: As we tackle other topics in the Internet of Things, it is helpful to think about recurring architectural patterns—in smart, connected products versus smart, connected operations, new and innovative experiences, and so on. The first section of the guide gives you a mental framework to think about your solution.

*Chapter 3, The Edge of the IoT*
: The edge of the IoT is where all the "Things" reside: from sensors to vehicles, everyday products to entirely new kinds of

gadgets. Our focus in this section is on how we will connect, secure, and interact with things from the cloud.

*Chapter 4, The Cloud*
: The cloud, of course, is a critical component of any IoT solution. This section of the guide outlines the key cloud technologies, design goals, and implementation details associated with IoT.

*Chapter 5, IoT Applications*
: All our hard work in connecting the edge to the cloud would be for naught if we didn't surface information about these "Things" through software applications. This part of the guide covers ways to get your applications to market or into the hands of your business quickly and effectively.

For technologists, the IoT has the potential to be most rewarding; it's where hardware, software, and networks bring new solutions to life, bridging the physical and digital worlds.

## Acknowledgments

This book would not have been possible without the contributions of Linda Frembes, and the O'Reilly editorial team, especially Susan Conant and Jeff Bleiel. Thank you for all your work.

CHAPTER 2

# Solution Patterns for the Internet of Things

How do we move from our disconnected world to a new, connected one where the boundaries between complex hardware and software systems are blurring? The Internet of Things presents us with design challenges at all system levels—from overall architecture to device connectivity, from data security to user interaction—and in the search for solutions, it's all too easy to get lost in the forest of standards, technology options, and product capabilities.

## Design Patterns and the IoT

While popular industry verticals like *connected health* and the *connected home* do not map cleanly to implementation approaches, there is another way of subdividing the space. We can map architectural patterns (spanning industry verticals) by examining existing, real-world IoT implementations irrespective of the hardware and software tools used. Let's identify those—in the spirit of the Gang-

of-Four[1] and Christopher Alexander's Design Patterns[2]—and use that understanding to help us place technical capabilities in the proper solution context. Throughout this book, as we tackle other topics related to the Internet of Things, we can use this initial solution pattern language to build a mental framework that supports other important details.

## Pattern Elements

For our general IoT solution patterns, we'll want some consistent characteristics with which to evaluate advantages and disadvantages, and compare and contrast between them. The five elements listed below help us, as technologists, extract the initial patterns and then analyze real scenarios:

*Solution creator*
: Who designs, engineers, and builds this IoT solution?

*Audience*
: Who buys the solution, and who will use it?

*Position in the product/service lifecycle*
: Is the solution positioned as a product or service that is an end-to-itself or does it enhance or augment an existing, mature product or service?

*Connection*
: How does the solution connect to the Internet?

*Integration*
: Does the solution require integration with other business or enterprise systems?

Armed with these characteristics for evaluation, let's examine three common, high-level recurring patterns that we see in the real world.

---

1 The software engineering classic *Design Patterns: Elements of Reusable Object-Oriented Software* (Addison-Wesley, 1994) describes a variety of solutions to common software design problems. The book's authors, Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, are often referred to as the "Gang of Four."

2 Noted Austrian architect Christopher Alexander's book, *A Pattern Language: Towns, Buildings, Construction* (Oxford University Press, 1977), on urban design and community livability, created a pattern language to enable anyone to design and build at any scale.

Of course, with the IoT, there are numerous technical patterns and subpatterns we can explore, but we'll start with these broad strokes.

# Smart, Connected Products

If you're in your home or office right now, you're likely surrounded by machines that you use on a daily basis: from televisions to LCD projectors, dishwashers to washing machines, ceiling fans to air conditioning units. For every one of these products, it's likely technologists are in the process of connecting them to the IoT, if they haven't done so already.

## The New Product-Consumer Relationship

As the products that we've been using for years, perhaps even decades, become enhanced through connectivity, the nature of the product-consumer relationship will change in a significant way. Manufacturers will be able to continually optimize both user and machine interactions through regular analysis of sensor data. Products will evolve on an ongoing basis, through their software, and manufacturers will continue to innovate well after the physical product has shipped. Perhaps most importantly, products will have features and functions resident in the cloud, outside of their physical footprint.

This shift has major implications for the product development and manufacturing lifecycle. In the past, when a product line matured—characterized by wide adoption but minimal sales growth—manufacturers attempted to rejuvenate them by adding more features and finding new uses and audiences.

With smart, connected products, manufacturers have an opportunity to continually rejuvenate their lines—not only through regular updates, but via analysis of usage data returning from these machines, making dynamic customization on a user level possible. This data-driven interplay between company and consumer alters the product lifecycle to more of an ongoing flow, a kind of living relationship.[3]

3 Follett, Jonathan, *The Future of Product Design*. O'Reilly Media, 2015.

As technologists, we should consider how a company could be hyper-responsive to users of its products. Smart, connected products offer great potential for creating ongoing dynamic interaction. For example, consider the numerous home appliances that can respond to energy cycles, from washing machines to dryers to dishwashers. Variables, such as the speed of agitation and the amount and temperature of water or air, can be customized based on personal usage.

## Elements of Smart, Connected Products

Let's examine the five key elements of smart, connected products.

### Solution creator

Product creators of every stripe—from big consumer electronics firms like Samsung to manufacturers like Deere & Co. to startups like Rest Devices, who produce the connected Mimo baby monitor—are looking to differentiate their offerings by giving users more compelling experiences. Often, this takes the form of features that are only possible by integrating the product functions with an Internet connection. Samsung's connected televisions, for instance, offer applications and programming that are Internet-based, as well as software updates to improve performance. Deere & Co., a leader in agricultural machinery, provides farmers with connected tractors that can be monitored in the field via their JDLink telematics system (as in Figure 2-1), and the Mimo baby monitor delivers video, audio, waking/sleep state, and even respiration information to the parent's smartphone anywhere in the world.

Figure 2-2 shows the Mimo IoT ecosystem: the "turtle" sensor talks to the "lilypad" gateway, which in turn transmits data about the infant to the cloud and eventually, the iOS or Android application. In Figure 2-3, you can see the Mimo mobile monitoring application, which displays infant position and respiration data, among other factors that parents can access anywhere on their smartphones.


*Figure 2-1. Monitoring John Deere's connected tractor in the field (Illustration courtesy Deere & Co.)*

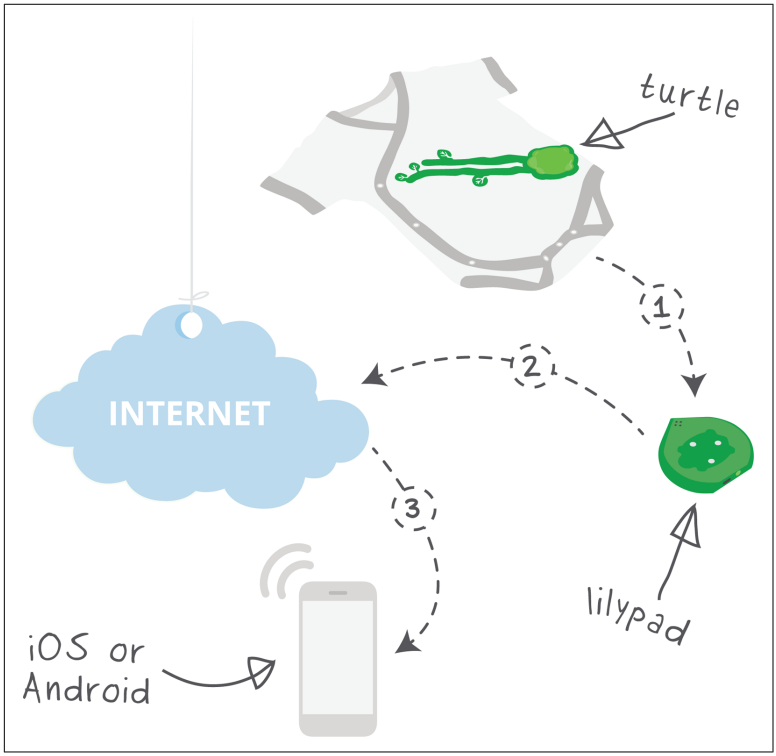


*Figure 2-2. The Mimo IoT ecosystem (Illustration courtesy Rest Devices, Inc.)*

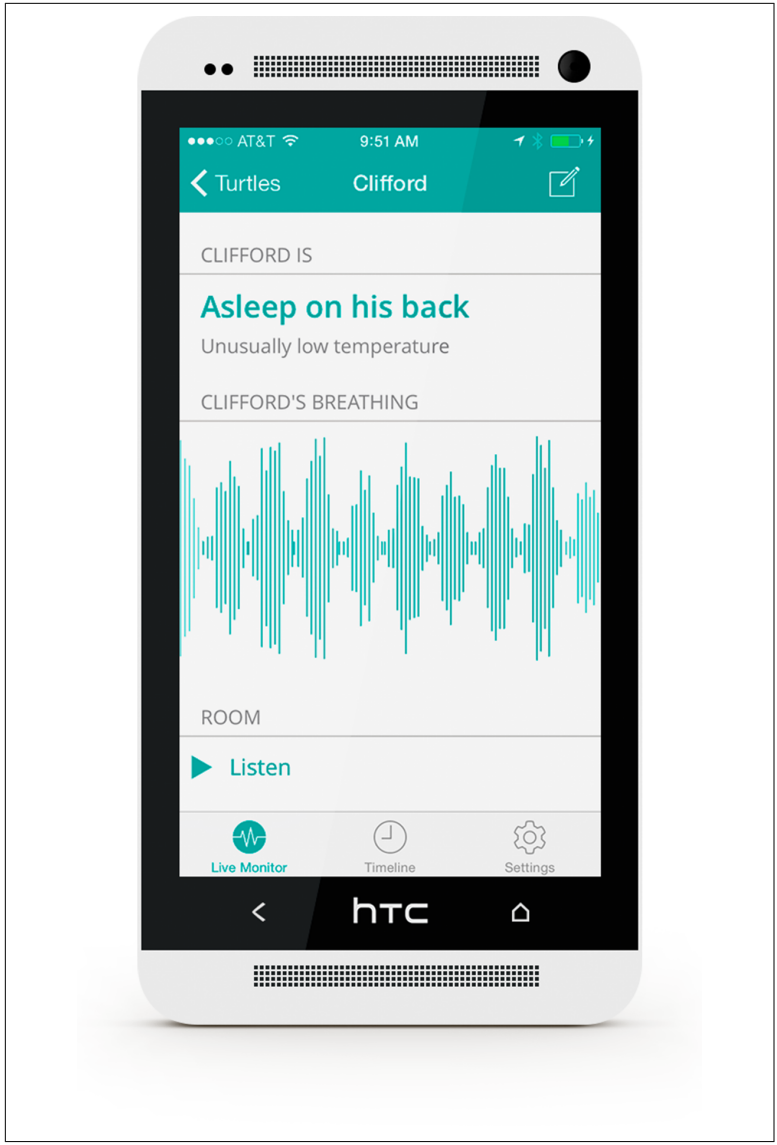


*Figure 2-3. The Mimo mobile monitoring application (Photo courtesy Rest Devices, Inc.)*

With connected products, manufacturers can collect and analyze usage data in order to refine future generations of the product. This refinement may come in the form of better understanding of failure modes so the product engineers may create a more reliable product,

or proactively schedule maintenance. Or it could mean understanding which features of the product get the most use, so the product managers and designers can hone in on what features are working well and what features are being ignored.

### Audience

It's important to understand who buys and uses the smart, connected product. In the long run, the audience will likely adhere to the same demographics as those who were buying the previous static, disconnected version. From the manufacturer's perspective, however, it's critical that the effort expended to design and build that smart, connected product result in meaningful differentiation and economic rent in the competitive marketplace.

### Position in the product/service lifecycle

Typically, these products serve as augmented versions of their disconnected counterparts, extending the features of the existing product types and categories that we understand today. However, as the IoT matures, we'll see products come to market that could not have been fully realized without an initial set of connected capabilities.

### Connection

Since 2012, the trend has been toward manufacturers designing connectivity directly into their products. Previously, when manufacturers were interested in connecting their high-value products—so that, for instance, services teams could remotely troubleshoot and react to product issues without the need for an engineer on site—they were forced to retrofit them for the IoT.

### Integration

Service monitoring aside, in most instances enterprise and business system integration for smart connected products is likely to be lightweight, if it exists at all. However, from the consumer software side, mobile applications, web portals, and analytics will be high value drivers, along with the function of the connected product itself. Business system integration, however, could become a common addition to such products, particularly if initial product pilots prove solution efficacy.

## Smart, Connected Operations

Sometimes the connected "thing" isn't a single product or device, but rather an entire operation that can be instrumented and optimized, with access to real-time system data and control capabilities from the cloud.

Smart, connected operations differ from the aforementioned products in that they often require retrofitting existing infrastructure with the sensors and communication modules that make an IoT solution possible. Additionally, system analytics, artificial intelligence for discovery and autonomous decision-making, and deep business system integration add a layer of complexity to connected operations not necessarily seen with individual products.

Smart, connected operations make a new level of system visibility and flexibility possible for industries as varied as agriculture, energy, transportation, and manufacturing. Let's look at a few examples of these to further examine the kinds of scenarios and use cases that make up the smart connected operations pattern.

### Agriculture

It should come as no surprise that agricultural operations face environmental conditions that can be highly unpredictable, requiring ongoing management of potential outcomes. Smart agriculture solutions track and react to these conditions by monitoring sensors in the field, managing information from weather and mapping services, and capturing actionable data, helping to spot potential issues with crops before they happen.

Smart agriculture solutions can also leverage AI technology that automatically learns from data, discovers patterns, and builds validated predictive models. Such predictive analytics can, for example, solve irrigation strategy challenges by maintaining crops within ideal soil moisture range, reducing water costs, and even predicting when water will be needed for irrigation. In this way, smart agriculture solutions cuts operating costs and increases a farm's production yield.

## Manufacturing

In smart manufacturing, businesses use the IoT to connect assets within operations and business systems, and provide real-time visibility for monitoring, control, and optimization. IoT applications connect and manage a complex set of disparate sensors, devices, and software solutions into a "system of systems," monitoring equipment condition and operating parameters to automatically trigger alerts and proactively initiate response from maintenance teams as soon as problems occur.

## Cities

Across the United States, from New York to Los Angeles to Boston, there are a variety of new initiatives to develop smart city services, using sensor technology and connected public resources—from street lights to trash bins to roads—to improve the quality of urban living. Examples of these initiatives range from well-coordinated transportation services using big data to reduce traffic congestion and save commuters time and fuel, to public safety and security services controlling police dispatch, municipal repairs, and even snow removal.

## Energy

Energy companies today face a whole raft of challenges: aging, patchwork infrastructure, increased regulatory controls, complex interconnected, interdependent systems—that make efficient, reliable delivery of energy increasingly difficult. IoT solutions help enable a smart grid to manage and automate the flow of both energy and information between utilities and consumers, leveraging a combination of sensors, smart meters and software controls, and analytics.

## Buildings

Commercial office buildings are increasingly becoming connected environments that connect HVAC, lighting, security, and safety systems with an array of embedded sensors that enable them to respond to real-time building occupancy and usage scenarios. These IoT solutions provide connected intelligence and automation to

reduce energy costs and increase visibility across building operations.

## Elements of Smart, Connected Operations

Here are the five key elements of smart, connected operations.

### Solution creator

Who is building the smart, connected operation? The answer varies widely based on the operation in question. A manufacturer may build a smart factory operation to streamline the production of the product; a systems integrator may instrument a building or a plant with sensors; or a city council may contract with multiple parties to transform their city.

### Audience

Typically an enterprise organization—whether it's a corporation or public sector agency—will contract with vendors or a systems integration firm to build out a smart, connected operation. And while smart, connected products may be designed and built prospectively, hoping that the market reacts favorably, smart, connected operations typically begin with a specific ROI target and objective in mind.

### Position in the product/service lifecycle

While the operation itself may be something that has been going on for years or decades, it's likely that it has not been instrumented for data collection or remote control. The IoT augments and brings efficiency to the existing processes in the smart, connected operation.

### Connection

To be sure, there are a wide variety of disconnected machines involved in most operations: factories, for instance, are filled with presses, riveters, and industrial robots. And while it's possible that the manufacturer of this equipment has already made them smart connected products, it is, more often than not, a required exercise to retrofit sensors to existing machines or to the environment itself. As such, it will be typical to find a gateway device communicating with sensors and/or existing data bus technologies that were already part of the operation.

#### Integration

With smart, connected operations, there are almost certainly many other business/enterprise systems with which to integrate. In many respects, the entire raison d'etre for the smart connected operation is to inform other critical systems such as enterprise resource planning (ERP), logistics, or manufacturing execution systems.

# New and Innovative Experiences

Thanks to the confluence of cheap microprocessors, ubiquitous WiFi, fast cellular connections, and shrinking devices, the IoT has the potential to create entirely new categories of product and services that will challenge our expectations. In some cases, these innovative experiences may even disrupt the marketplace, displacing older technologies entirely. Regardless, innovative experiences, as an IoT design pattern, represent a mash-up of hardware and services that generate new value beyond that of the speed, convenience, and optimization that connected products and operations can provide.

## Wearables

Wearables, such as smart watches and fitness bands, are excellent examples of products that typify this kind of innovation. The traditional experience of wearing a watch is being transformed entirely by a combination of sensors, connectivity, data aggregation, and analysis. The wearable wrapped around your wrist can track your steps, monitor your stress level, and even let your husband, wife, or significant other know if you've gotten lost in the wilderness during a long trail run.

In the near future, it's conceivable that data obtained from a wearable could be streamed to your health care provider, your personal trainer, and maybe even your boss, operating without any intervention from you, the user. And, as wearables further shrink in size and increase in availability, they'll be used to track employees at work, children at play, and even the elderly in assisted living.

For instance, the FitBit Surge (Figure 2-4) provides distance, time, and heart rate data to the user in addition to a host of other factors.


*Figure 2-4. The FitBit Surge tracks everything from exercise type to sleep stage (Photo courtesy FitBit)*

## Connected Environments for Work, Play, and Health

The smart, connected home will open up new markets for entertainment, collaboration, security, and even health monitoring, as audio-visual equipment, lighting, and climate control systems combine with sensors, medical devices, and communication tools.

Technologists have already demonstrated that they can make cool sensors you'll wear on your body. Soon, they'll design new products to capture your data beautifully and invisibly. Consider your future bathroom, connected to the IoT and awash in invisible sensors that snag your physiological data—weight, heart rate, blood flow, even urinalysis, all recorded automatically. If you think that's crazy, consider that Withings, for example, is already connecting a scale (Figure 2-5), heart monitor, and other diagnostics to the IoT.


*Figure 2-5. The Withings Smart Body Analyzer and mobile app (Photo courtesy Withings)*

This is where machine learning, big data, and design merge with the IoT. And your bathroom will be transformed into a healthroom,[4] as pictured in Figure 2-6, outfitted with a panoply of noninvasive diagnostics for the early detection of chronic diseases. Talk about disruption!

4 Follett, Jonathan, *Designing for Emerging Technologies*. O'Reilly Media, 2014.

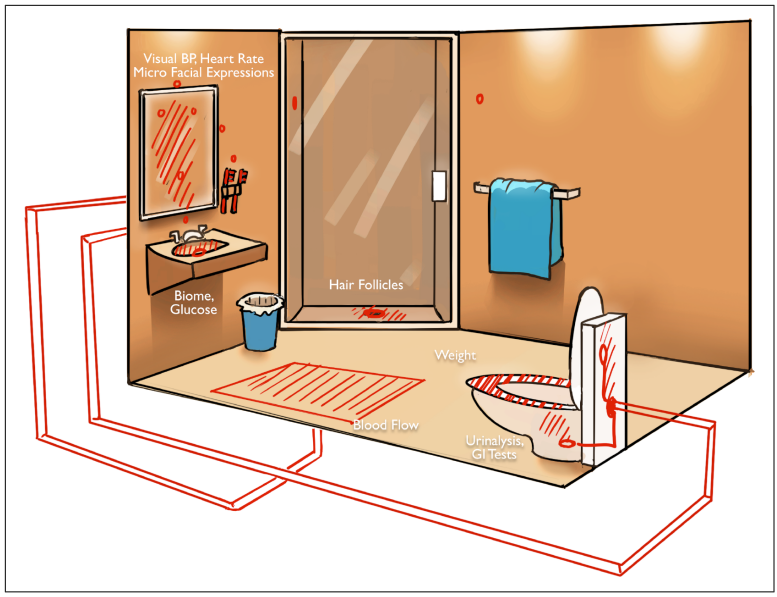


*Figure 2-6. The connected bathroom as healthroom (Design by Juhan Sonin, illustration by Quentin Stipp, courtesy of Involution Studios)*

## Elements of Innovative Experiences

Let's break down the elements of new and innovative experiences.

### Solution creator

The innovators creating these experiences will run the gamut from big technology players such as Apple with the Apple Watch, to start-ups such as AdhereTech (the creator of the smart pill bottle for medication adherence, shown in Figure 2-7), who have the potential to become market leaders of the future. Large companies, however, with their set infrastructure and focus on existing products, may find it more difficult to innovate than smaller, more nimble competitors who are not beholden to the past.

*Figure 2-7. The smart pill bottle from AdhereTech (Photo courtesy AdhereTech)*

### Audience

While innovative experiences can often be the result of unexpected opportunities that are difficult to predict, inefficient, capital-intensive markets and industries are prime targets. The health care industry, for example is primed for disruption, and the retiring Boomer generation is a major audience with increasing health needs —a huge opportunity for the AdhereTech smart pill bottle or connected healthroom.

### Position in the product/service lifecycle

These IoT innovations are entirely new and stand alone, challenging our expectations of what connected products and services can do for us. In the product/service lifecycle, they are at the earliest, introduction phase. For this reason, we can expect that recyclability and designing for reuse will be important factors. As technologists, we must ensure that IoT innovations that do not succeed in the marketplace avoid becoming landfill.

### Connection

Disruptive experiences could connect to Internet directly or through a gateway. But, given the time to market for developing a brand new product from scratch, it will be common to see new ways of doing old things in existing environments by bringing in retrofit gear.

### Integration

Are there many other business or enterprise systems to integrate with? Perhaps, but an innovative experience could also be something so paradigm-shifting that it stands on its own.

CHAPTER 3

# The Edge of the IoT

The edge of the IoT is where the action is. It includes a wide array of sensors, actuators, and devices—those system end-points that interact with and communicate real-time data from smart products and services.

By 2020, it's projected there will be anywhere from 25 to 50 billion[1] Things[2] connected to the IoT—that's up to seven connected Things for every person on planet Earth. On our way to this milestone, we can anticipate that these billions of connected objects will generate data volume far in excess of what can easily be processed and analyzed in the cloud, due to issues like limited bandwidth and network latency (among others).

## Living on the Edge

*Edge computing* or *fog computing*—a paradigm championed by some of the biggest IoT technology players, including Cisco, IBM, and Dell—represents a shift in architecture in which intelligence is pushed from the cloud to the edge, localizing certain kinds of analysis and decision-making. Edge computing enables quicker response

1 "Gartner Says 4.9 Billion Connected 'Things' Will Be in Use in 2015; In 2020, 25 Billion Connected 'Things' Will Be in Use." *http://www.gartner.com/newsroom/id/2905717*, accessed Mar. 24, 2016.

2 "Cisco Global Cloud Index: Forecast and Methodology, 2014–2019." *http://bit.ly/25wqkDN*, accessed Mar. 24, 2016.

times, unencumbered by network latency, as well as reduced traffic, selectively relaying the appropriate data to the cloud.

Regardless of whether system intelligence is ultimately located in the cloud or the fog or some hybrid of the two, development for the Internet of Things requires technologists to have a clear understanding of edge architecture and how information is both gathered from devices and communicated.

## An Abstract Edge Architecture Model

While specific solutions—from smart homes to smart grids to smart factories—may have their own unique variations, for the purpose of discussion, let's abstract a basic edge architecture that describes the key elements.

The foundation of our stack, pictured in Figure 3-1, is the "Thing," that critical product or environment that is the core reason for our IoT system build.

In the next layer, *sensors* and *actuators* provide the Thing's "read and write" capabilities. These sensing and actuating components may either be built into the smart product or environment, or, in the case of a retrofit, they could be added after the fact.

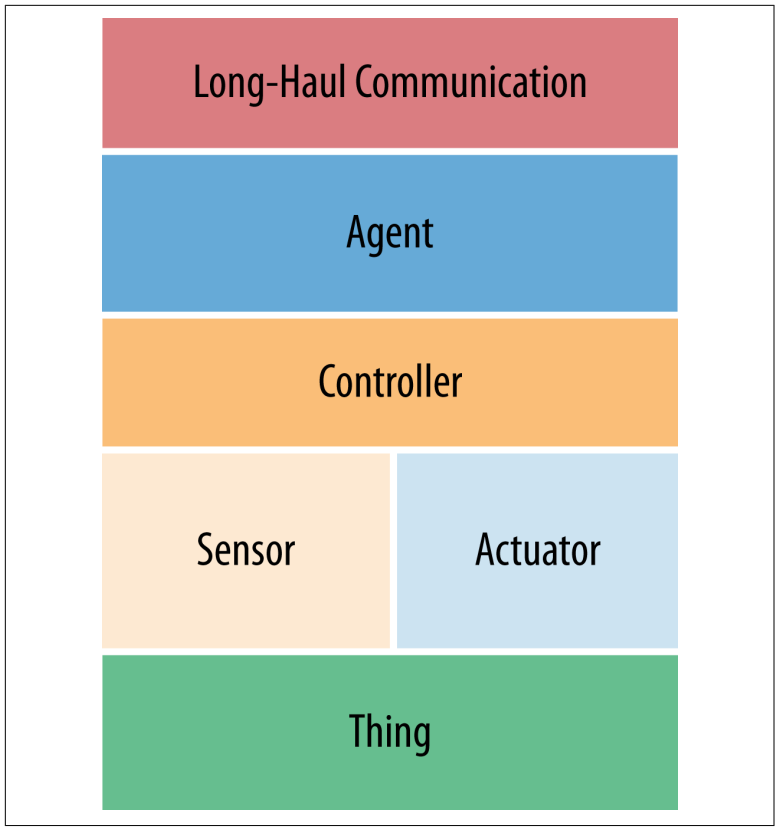


*Figure 3-1. An IoT "Thing" anatomy describing the key elements resident in an edge device*

### Sensors

Sensors read and report on the real-world status of connected products, machines, and local environments. They are the eyes and ears of the system, monitoring environmental elements like temperature, light, and moisture. Ongoing sensor innovation, an often-overlooked area of IoT technology, will be critical for evolving and improving solutions.

While we might think of sensors only as physical objects, *anything that can be read*—from files to product-specific data—can and should be considered sensor input. For example, a piece of industrial equipment may have hundreds of data points unique to that product, and every one of them could be considered a sensor.

Sensors may be physically hardwired, built into the product, or communicate via a short-haul communication protocol like Bluetooth Low Energy (LE) or ZigBee.

Examples of sensors include:

- Temperature sensors
- Light sensors
- Moisture sensors
- GPS receivers
- Vehicle on-board diagnostics
- Files
- Product-specific data

For instance, the Grove water flow sensor, shown in Figure 3-2, is part of Seeed Studio's hardware innovation platform for technologists to take new ideas from prototype to production. It reads liquid flow rate using a water rotor, whose speed changes depending on how fast the water is moving. The signal output comes from a Hall effect sensor, which pulses as the rotor turns.

*Figure 3-2. The Grove water flow sensor (Photo courtesy Seeed Development Limited)*

### Actuators

Actuators affect the electromechanical or logical state of a product or environment. They are the system's hands and feet. Actuators might include a light that can be turned on and off, or a valve that can be opened and closed.

System commands sent to embedded applications—such as remote reboot, configuration updates, and firmware distribution—should also be considered actuation because, by changing its software, the system is in fact changing the physical reality of a product.

Examples of actuators include:

- Lights
- Valves
- Motors
- Commands ("soft" actions, file distribution, firmware updates)

For instance, the 6000 series indexing valve from K-Rain, pictured in Figure 3-3, is a robust distribution valve that can be used for high-flow city water, irrigation, and even wastewater applications. The valve acts as a manifold, directing the flow of water from zone to zone, and can be coupled in an IoT solution with an intelligent valve monitor to ensure even water distribution and alert operators to potential malfunctions.

*Figure 3-3. Indexing valve (Photo courtesy K-Rain)*

### Controller

The next layer in our stack is the *controller*, a hardware or software component that interacts electrically or logically with sensors and actuators. It is in the controller that we'll find our low-level, short-haul communication.

While in many instances the controller may be fused within other elements of the stack, it is always present logically. For example, a controller may be a simple circuit that reads an analog signal from a temperature sensor and digitizes the signal into discrete transmissions that the upper layers of the stack can understand.

Over short distances, local communication from sensors can come via a simple serial connection between devices, or short-haul wireless technologies like ZigBee. Industries may define standard protocols for interfacing with equipment—for example, OBD-II for automobiles, or DEX and MDB for vending machines. All of these represent short-haul protocols, because they are meant for local communication between sensors, control systems, and an agent.

### Agent

The next layer in the stack is the *agent*, an embedded program that runs on or near the IoT device and reports the status of an asset or environment. The agent acts as a bridge between the controller and the cloud, deciding what data to send and when to send it. This process operates in reverse as well, as the agent processes and responds to cloud-based commands and updates.

As an example of the controller and agent working in concert, imagine that we're engineering a proof-of-concept device for an IoT system using a Raspberry Pi and an Arduino with a breadboard. The Arduino is the *controller* running LEDs and servos, and acquiring data from a sensor. The Raspberry Pi is interfacing with the Arduino, and running a software *agent* that decides when to send the sensor data to the cloud, via a long-haul connection to WiFi / Ethernet.

### Long-haul communication

On the top layer of our architecture, we find our long-haul communication to the Internet. IoT solutions invariably require that environment or device status be made available to a cloud-based application for consumption by a variety of stakeholders. Once an agent has received information via short-haul, it must retransmit that information to the cloud. The desired characteristics of these long-haul protocols are much different than short-haul, particularly in the categories of security, footprint, and reliability. There are a wide variety of long-haul options for IoT solutions, dependent on the use case; they include cellular and satellite, WiFi and wired Ethernet, as well as subgigahertz options like LoRa and SigFox.

Networking protocols for long-haul communication are similarly diverse; they include TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) for the transport layer, and HTTP (Hypertext Transfer Protocol) and CoAP (Constrained Application Protocol) for the application layer, among many others.

## Device Types

To further our understanding of how the edge of the IoT works, let's look at some typical examples of the key hardware components that make up our devices. While it's true there can be an overwhelmingly wide variety from which to choose, the basic component types—from modules to microcontrollers—provide us with concrete examples of the elements required for edge connectivity and logic. The examples that follow are largely focused on cellular technology, although devices using WiFi, wired Ethernet, etc., for connectivity are also considered.

### Modules

*Communication modules* are components for connecting to WiFi, cellular, or long-range wireless networks. While modules typically have little to no programmability, vendors do provide a variety of configurable options and even lightweight scripting.

Original equipment manufacturers (OEMs) and custom solution providers building IoT capabilities directly into their products often incorporate communication modules into a custom board design. For instance, the AirPrime MC Series communication module from Sierra Wireless, pictured in Figure 3-4, is incorporated into both connected consumer electronics, as well as industrial grade solutions like Itron's OpenWay Smart Grid.

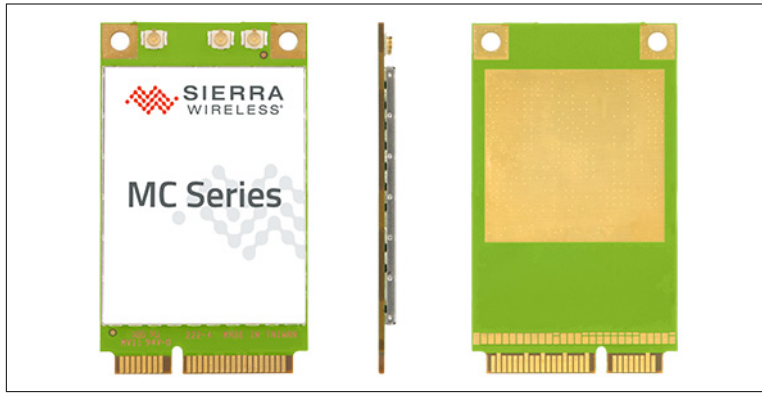


*Figure 3-4. AirPrime MC Series communication module (Photo courtesy Sierra Wireless)*

Vendors that build modems and routers also use modules in their products. However, because manufacturers don't pre-certify communication modules, any custom products incorporating the module must be certified by the carrier prior to usage.

### SoCs and microcontrollers

For building connectivity and logic into your IoT product, you'll need a *microcontroller* like the CC3200 from Texas Instruments, pictured in Figure 3-5, or a *system on a chip* (SoC) like Qualcomm's Snapdragon processor (designed originally for mobile computing but now able to be embedded just about everywhere).


*Figure 3-5. The CC3200 microcontroller (Photo courtesy Texas Instruments)*

While both SoCs and microcontrollers are integrated circuits that include many of the components of a complete computer, SoCs typically have greater amounts of RAM and more powerful processors. As a result, SoCs are capable of running robust operating systems such as Linux or Windows and more complex software. Microcontrollers and SoCs are used in a wide variety of IoT applications, from wearables to connected cars, among others.

### Modems and routers

At its core, a *modem* is essentially a communication module on a board, enclosed in a physical housing, with a serial connection to plug into a computer. Since modems are pre-certified by the manufacturer, they can be used right away after purchase, as long they're on a carrier's certified list. In scenarios where PCs are already controlling high-value units—industrial machines in factories or large medical devices in hospitals—modems can be the right fit for adding connectivity.

*Routers* are similar to modems. However, instead of a single serial connection to one PC, routers allow for multiple computers to share

a cellular data connection. A router can also initiate the connection to the cellular network itself, as opposed to a modem, which needs to be explicitly controlled by the serial-connected PC.

For instance, Cisco's Connected Grid Router, shown in Figure 3-6, built specifically for IoT applications like smart cities and smart grids, can be installed outdoors to support sensor networks.


*Figure 3-6. The Cisco Connected Grid Router (Photo courtesy Cisco)*

### Gateways

*Gateways* from companies like Dell, Intel, Multi-Tech, and others are essentially small form factor computers. They have built-in connectivity (WiFi or cellular), capable CPU architectures, and plenty of memory, running full-featured operating systems such as Windows or Linux. Figure 3-7 shows Dell's Edge Gateway 5000.

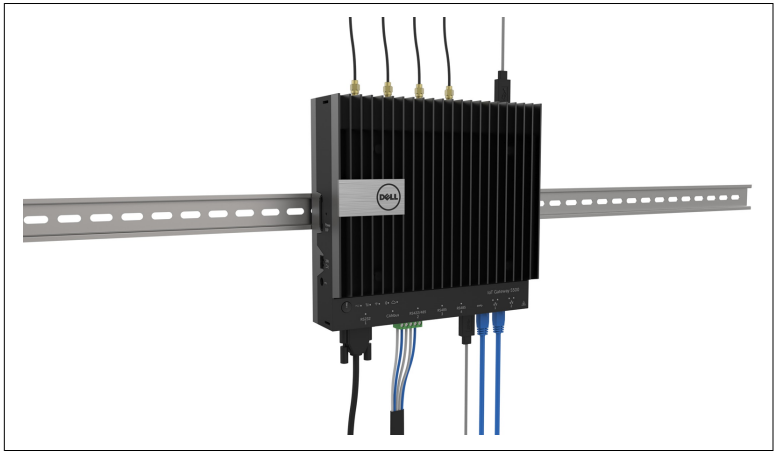

*Figure 3-7. Dell Edge Gateway 5000 Series (Photo courtesy Dell)*

Gateways are great for retrofitting unconnected Things, enabling the connectivity of industrial devices and other systems to the IoT. Gateways provide a place to put your IoT agent code and often include an SDK to make the coding and deploying of an agent straightforward. They connect legacy and new systems, and enable data flow between edge devices and the cloud.

### Trackers

A *tracking unit* is an excellent example of a fixed-purpose, machine-to-machine device engineered to perform a specific set of functions for a vertical market. Trackers, like the CalAmp LMU 3030 pictured in Figure 3-8, can track vehicle speed and location as well as access vehicle diagnostic data. They're useful for fleet tracking, car rental driver monitoring, and other automotive applications.

*Figure 3-8. CalAmp LMU 3030 (Photo courtesy CalAmp)*

Trackers are essentially modems coupled with a CPU. As such, they have built-in connectivity, GPS, and advanced configurability to dictate behavior. Trackers have built-in agents, as well as their own long-haul protocols.

### Prototyping boards

This list of edge components would not be complete without a discussion of *prototyping boards* and their impact on ideation and design for the IoT.

Prototyping boards such as the Arduino Uno (shown in Figure 3-9), Intel Galileo (shown in Figure 3-10), and Raspberry Pi are ideal for proof-of-concept work. They have just about everything a designer or engineer needs to start creating for the Internet of Things:

*Connectivity*
: A prototyping board comes equipped with a wireless module or an Ethernet chip and RJ-45 port to connect to the wired Internet.

*Application development stack*
: Prototyping boards have a way to load code, whether it be Arduino Sketch, Java, or a C programming language.

*Expandability*
: You can easily add breadboards, expansion boards, and sensors to prototyping boards.


*Figure 3-9. The Arduino Uno is used to quickly prototype IoT solutions (Photo courtesy Arduino)*

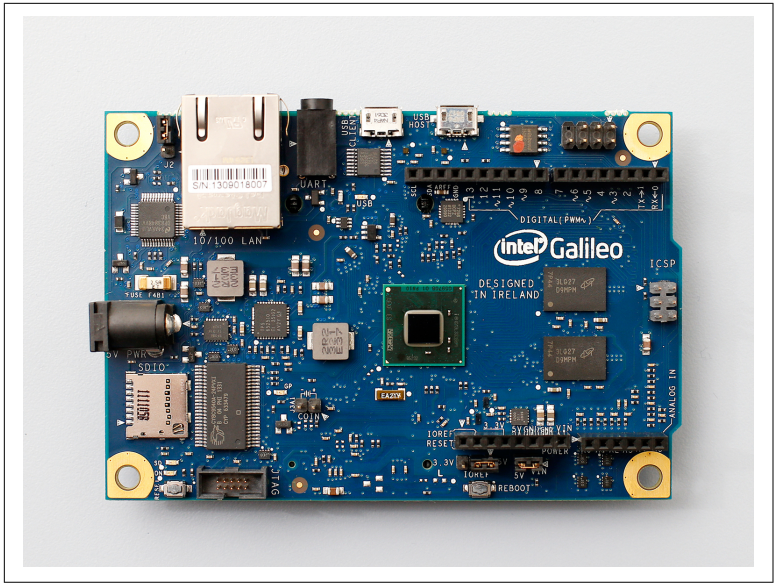


*Figure 3-10. Intel's Galileo is based on the x86 architecture and supports Arduino hardware expansion cards and software libraries (Photo courtesy Intel)*

While it's commonplace for IoT product companies to use prototyping boards to test out new ideas, these boards usually are not appropriate for deployment scale usage from a cost standpoint. When manufacturing a high-volume production run of a product, product engineers and designers will necessarily be concerned with the keeping the bill of materials (BoM) in check. To conserve costs, it's often necessary to alter the prototype design for a *production build*, tossing out the parts of the board that are not actually being used.

For example, if the prototyping board has a powerful CPU with a 1.1 Ghz processor, but the product really only requires a 400 Mhz processor, appropriately specifying a less powerful processor will positively affect the bottom line. In these cases, finding the right prototyping board vendor to help you transition from the prototype stage to production can be critical.

In the end, it's unlikely that your dishwasher will be connected to the IoT by a Raspberry Pi inside of it. However, it's quite likely that the dishwasher manufacturer may embark on its IoT journey by creating a proof-of-concept with a Raspberry Pi.

# Edge Architecture Examples

To make our abstract edge architecture a little more real, let's compare and contrast two different solutions, one using an embedded agent with WiFi for long-haul communication, and another using a gateway with cellular connectivity.

For this exercise, we'll use a common, but surprisingly transformative IoT solution: the smart, connected vending machine. This example typifies the potential of the IoT to provide real business value to industries that might seem unadventurous from a technological perspective.

From inventory management to better provide customers with popular food options, to remote diagnosis and repair to maintain the machines, to power consumption and temperature monitoring for increased energy savings, the connected vending machine provides a great example of how the IoT can bring business transformation through data transparency.

The connected vending machine solutions pictured in Figure 3-11 and Figure 3-12 use the same foundational edge architecture.

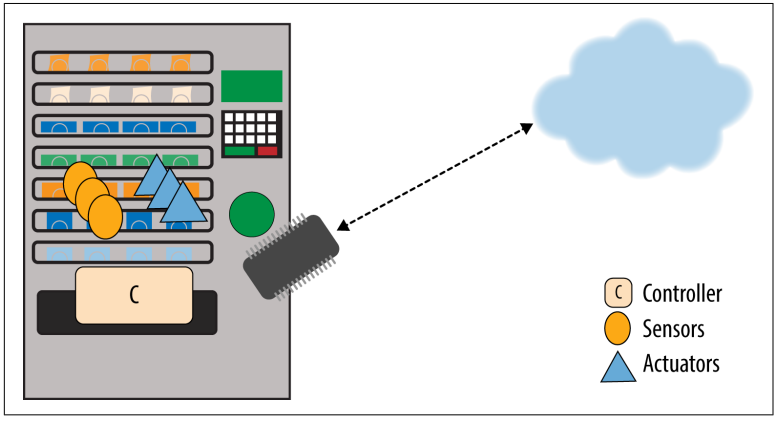


*Figure 3-11. Embedded agent with WiFi long-haul*

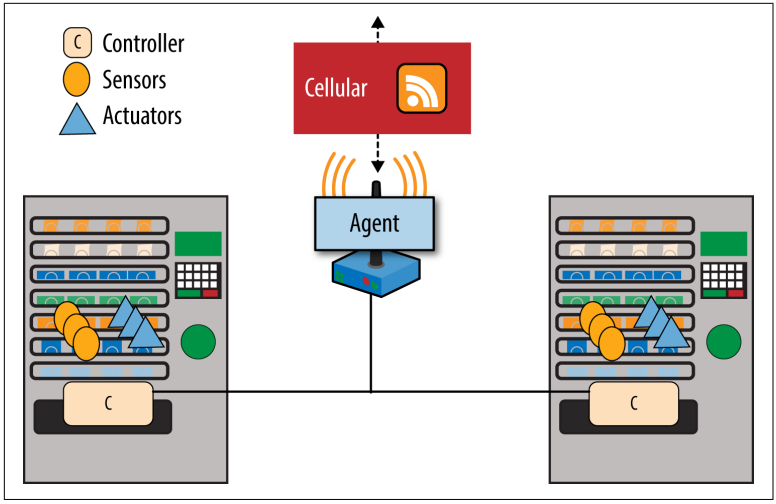


*Figure 3-12. Gateway example with cellular*

The *sensors and actuators* for the vending machine include the internal electromechanical units, such as a bill validator for receiving payment, and motors for manipulating and moving product inventory.

The *controller layer* uses the vending industry standard control busses:

*MDB (multi-drop bus)*
: Allows transactional devices like the aforementioned bill validator to communicate with the vending machine's brain.

*DEX (digital exchange)*
: The means by which the product inventory is tracked.

The *agent* interfaces with the DEX and MDB control layer to determine the amount of products the vending machine has in inventory, as well as how much money it has collected over the course of the day. It then relays that data to the cloud.

## Real-World Deployment

From this point on, our two example architectures differ. Where the agent resides and how it connects to the cloud is determined in part by the topography and nature of the machines' real-world deployment.

If the vending machine is located by itself, perhaps in the middle of a theme park, we'll need a solution where connectivity can be embedded directly inside the machine itself, as in Figure 3-11. In this instance, our agent can be loaded on to a microcontroller and our long-haul communication handled through the local Wi-Fi network.

However, in contrast, if there are 10 vending machines co-located in a single environment, perhaps in a skating rink, it's unnecessary for each one to have its own agent and long-haul transport. Instead, all 10 machines can be wired together with one IoT gateway, as depicted in Figure 3-12. In this case, each vending machine can talk to the local gateway, where the aggregation logic, other business logic, and agent logic reside, as well as the long-haul connectivity. In this example, our agent is loaded onto the gateway and long-haul communication is handled through cellular.

From these examples, we can see how embedded and gateway architectures can be equally valid and have their place, depending on the real-world circumstances that bound the IoT solution.

# CHAPTER 4
# The Cloud

The cloud is the central station of any IoT solution, and a critical component. While it may be tempting to think about the IoT device cloud as the equivalent of a web or mobile application, the Internet of Things introduces its own unique characteristics and subsystems. In this chapter, we'll examine some of those characteristics, including cloud-device connectivity and security, device ingress and egress, and data normalization and protocol translation.

In IoT technical diagrams such as Figure 4-1, it's common to structure visualizations of the architecture from the bottom up, reflecting the mental model of edge devices being "on the ground" relative to the cloud. Using this diagram, let's follow the journey of a message as it flows from the edge to the cloud.

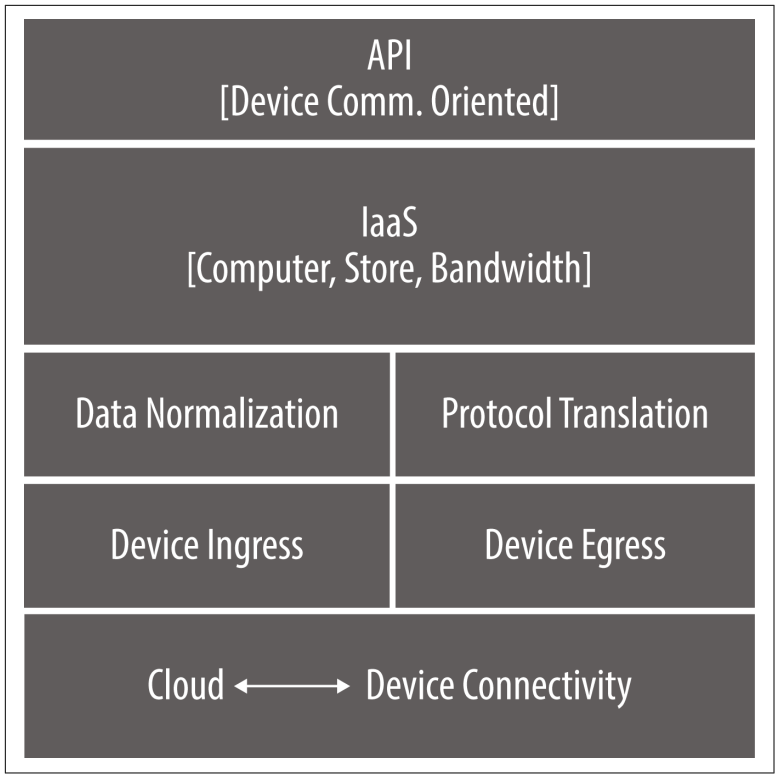


*Figure 4-1. The device cloud is the central station of an IoT solution, processing and analyzing data received from the edge*

## Cloud-to-Device Connectivity

Making sure we get the right message from the edge device to the cloud is paramount. When it comes to cloud-to-device connectivity, security is increasingly a major area of concern, and rightly so. Even a modest-sized IoT installation can provide numerous potential intrusion points via edge devices that, unlike modern PCs or smartphones, have limited computing resources. This combination of traffic quantity, device constraints, and the wide variety of IoT solution configurations makes for a bevy of challenges.

*Device authentication* prior to any data transmission is a critical security measure (Figure 4-2). In a web or enterprise application there are authenticated users, systems the users are authorized to access, and roles that designate the users' privileges. In the IoT, however, we see a new kind of actor in the system: the device itself,

which must supply its own credentials and identity to the cloud. Device authentication prevents data spoofing and junk data, and guards against cloud-to-device information flow that could get into the wrong hands. Since this access point represents a new attack surface, a device identity and authorization layer is a requirement for any system. IoT-connected devices also need the benefit of encryption standards like Transport Layer Security (TLS) v1.2 and banking-grade certificate exchange and validation.

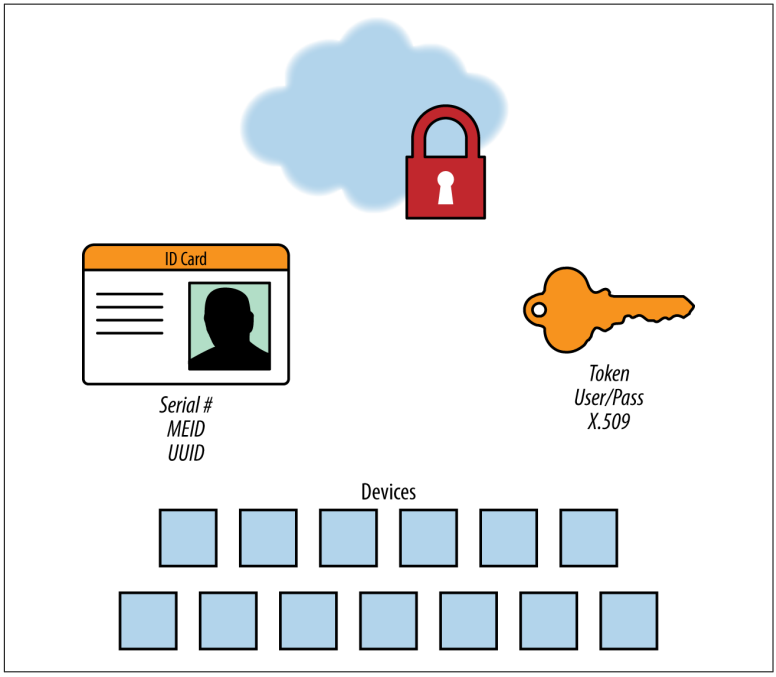


*Figure 4-2. Device authentication is a critical security measure*

In an IoT solution, every product, sensor, gateway, and communication device at the edge requires a unique identifier. As you might expect, there are a variety of techniques for achieving this, which is why it's important for the cloud software to be flexible in the way it identifies a device. Often, you can use the serial number of a product (stamped-in by the manufacturer) as a unique identifier. However, it's not unusual to encounter products that lack a serial number altogether. If the device has a mobile radio, you can use the mobile equipment identifier (MEID) or another form of cellular identifier.

Once you've chosen an identifier, you'll need to authenticate to the cloud. But common techniques in consumer and enterprise systems to solve this problem fall short when it comes to the IoT. For instance, in the case of username/password combinations, how do you update the password of an IoT device at the edge? And what if there are millions of them? X.509 certificates that identify a client securely to a communicating server are the best approach. However, a public key infrastructure (PKI) is needed to produce these certificates and sign, distribute, and revoke them when they've been challenged.

## Messaging and the IoT

How do you represent a message in an IoT system? To begin with, we need our credentials—our device ID card—to be part of the *message envelope*. The envelope protocol tells the system, "Here's who I am. Here's where I want this message to go. And here are some characteristics of how I want the message to be delivered."

When it comes to envelope protocols, there are many emerging standards, some of them more pertinent for certain industries or certain types of IoT solutions. MQTT (Message Queuing Telemetry Transport) and CoAP (Constrained Application Protocol) are strictly envelope protocols. They do not care about the contents—to them, the message inside is just a bucket of bytes. When describing the message contents, engineers may use JSON, HTML, CSV, or even a proprietary binary format. Other protocols such as AlwaysOn are designed to incorporate both envelope and message information (Figure 4-3).

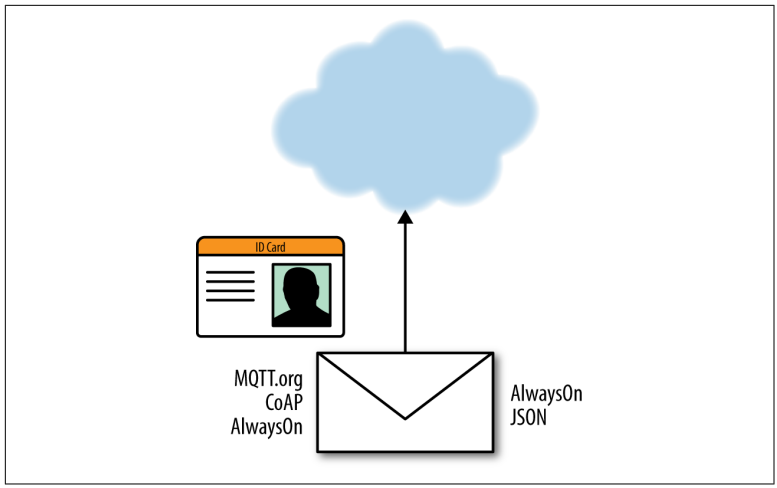


*Figure 4-3. While messaging protocols contain identity information for authentication purposes, most leave the content to another format*

## Back to the Future: How Reliable Is Your Network?

At the beginning of the Internet revolution in the late '90s, network reliability was not a given, and outages were not unusual. Software engineers could not assume they would have continuous connectivity. Over time, improved connectivity and fault-tolerance made these scenarios less of a concern. However, when it comes to the Internet of Things, the relatively rudimentary nature of device connectivity requires adjusting the engineer's outlook yet again—perhaps back to the cautious mindset of the early Internet days.

IoT networks, as you might expect, are generally not as reliable as today's mobile or business Internet connections. The IoT does not, as yet, have the same fault-tolerance built into the client and server layers that you might expect with consumer-oriented modern web browsers and smartphone applications. In comparison, IoT edge devices run on relatively rudimentary operating systems and are subject to a variety of real-world factors that can limit connectivity.

For example, in a smart, connected operation for commercial vehicle fleet-deployment and monitoring, a truck might be outfitted with telematics device, communicating over a cellular network to provide regular updates regarding location, status, and route. However, at any moment, data flow from the vehicle could be compromised by something as simple as driving into a parking garage or

tunnel. Suddenly, the truck disappears from the network and the system receives a fragment of a message. The software waits on those bytes, eating up resources in our cloud networking layer. The truck may be gone for minutes, hours, or days. We have no way of knowing. And then, as suddenly as it left, it returns, and the system receives the rest of the message. This all-too-common scenario needs to inform the design of IoT systems. Importantly, our software layer needs to understand and compensate for network unreliability, and not assume that we'll receive complete messages or have a well-behaved network client from connected devices.

## Device Ingress/Egress

Next, as we move up our IoT cloud stack, is the *device ingress and egress layer*. IoT solutions, the majority of the time, are concerned with device ingress—receiving incoming messages from the edge to the cloud. The sheer volume of data can challenge the scale of cloud systems. A system might have hundreds of thousands of connected devices, each sending multiple messages every minute. For example, if every device in a 250,000-device system sent just 3 messages per minute, it would result in over a billion messages per day. By way of comparison, Twitter, as of this writing, has about 320 million monthly active users who send about 900 million messages per day. It's feasible that the data ingress of *just one IoT installation* can easily match or even surpass the total traffic of Twitter on a daily basis.

In contrast, once in a while, an IoT system needs to communicate with one of the many connected devices. Cloud-to-edge messaging may include sending notifications about new configurations, firmware, or commands to trigger actuation. While it's far more infrequent than device ingress, when you need device egress, it's important to have the right communication protocol and communication infrastructure in place. Particularly when triggering actuation, it's important that users see timely responses to their cloud-based commands. Perhaps a smartphone user is asking their garage door to close because they realize they left it open when they exited the house. From a user experience perspective, if the command is successful, the end result can seem almost magical—the ability to instantly and remotely command one of many connected devices.

Of course, some classes of edge devices are not always powered on or do not maintain a persistent network connection. Duty-cycled

devices simply connect on their own schedule and receive updates when reconnection occurs, much like retrieving email from a POP3 email client. The downside with this setup is that you don't get your messages exactly when they're sent. Some IoT frameworks allow for an active connection from the cloud to the edge, pushing messages as soon as they're available. This gives a much crisper interaction for end users who want to control and update IoT devices. Even though device egress happens rarely, when it does happen, it needs to work well.

# Data Normalization and Protocol Translation

While industry coalitions are hard at work defining open standards for interoperability with the IoT, there are many legacy M2M systems with thousands of connected devices that currently speak different proprietary protocols. For this reason, a well-designed IoT cloud system includes a dedicated protocol translation layer. This layer translates the "over the wire/air" protocol to the native, or canonical, protocol that's understood by the upper layers of the system.

Once a message is received, the data needs to be normalized, which involves extracting information from the device message and putting it into a data storage schema. As the message comes through the device ingress layer, you'll need to parse the JSON, CSV file, or proprietary format. IoT data is often *time series* data—discreet values with identifiers, like sensor type or status reading, and a timestamp of when that value was ascertained. However, just having time series-data storage doesn't completely solve the problem, as there are also many forms of unstructured data such as log files, diagnostics, images, and video.

## Data Consistency

Message delivery *consistency* has a significant impact on your IoT applications. The order in which messages are received and processed makes all the difference.

When scaling the messaging infrastructure for an IoT application, be wary of dumping messages into a queue and processing them later. When you start to distribute those queues and distribute your architecture, it's very easy to forget to connect the temporal locality of a single device that sent us a series of messages being processed in order. For instance, if a device has a status of "door open" and then a

second later "door closed," what happens if the messages are received out of order? Now there's a temporal ordering issue. The system needs to assert that the first message the device sent is received and processed by the application before the second message.

# Infrastructure

The next layer in our cloud stack is the Infrastructure layer. For many IoT solution creators, the elastic compute resources of an *Infrastrucure-as-a-Service* (IaaS) provider are a sensible alternative to the substantial initial investment and ongoing maintenance cost of constructing their own data centers. Infrastructure-as-a-Service, whether public or a managed private cloud from an IaaS vendor, can give solution creators the ability to automatically provision new compute, storage, and network resources on demand. This is particularly useful for IoT solutions that may have significantly large but temporary workloads. For example, within the annual lifecycle of an IoT system, there are certain events that demand more compute and bandwidth resources than others—for instance, a firmware update deployed to a million devices, or the seasonal launch of a new product. IaaS resources are convenient and scalable: the third-party vendor owns all compute resources, storage, and networking capabilities, and handles all system maintenance.

## How Much Data Is Too Much Data?

Collecting trillions of data points demands a well-thought-out strategy in order to set automated policies for data retention and "data governance." IoT data can be big, fast, unstructured, and write-heavy. When the data is pouring in, it might seem like you're collecting and storing information that's just meaningless noise. And it's true that some classes of IoT applications rarely read data values a second time once they've been written. In cases like this, some would argue the data isn't going to do any good, because no one will ever look at it. In those situations, it's a question whether to store the data at all.

But we do, indeed, want to store it. The useless data "noise" of today might well be the most impactful information of tomorrow. With the help of a *machine learning algorithm*, you may find that some of your data is in fact a signal that something is about to go wrong, giving you non-intuitive insight into your IoT system.

Even if you don't have the machine learning implementation and algorithms to process it today, don't throw these data points away, because training those future machine learning models with historical data can be absolutely critical.

### The Value of Small Data

With all this discussion of big data, it's worth keeping in mind the great value that you can get from *small data* generated by an IoT installation. There are many use cases where simply knowing the location and status of a piece of equipment on a daily basis can provide tremendous value. For instance, if you've leased a piece of construction equipment to a customer who's required to keep it in Massachusetts, perhaps once a day you'll want to know that your equipment remains in the right area. You don't need to hear from it every second, or need big data from it. You just need to know where it is. In scenarios such as this, small data can be very useful.

## APIs

The final layer in our IoT cloud device stack is the API, the *lingua franca* of modern systems. The majority of API usage occurs at the *application layer*, which is discussed in more detail in Chapter 5; however, there are a few notable elements we should discuss here. In particular, the device cloud stack should provide lower-level access to insulate the application layer from the devices. Common API functions may include:

- Provisioning a new device
- Querying data elements supplied by a device
- Sending commands to devices

## The Topology of the Cloud

The tremendous power and potential of the Internet of Things comes from the fact that it binds digital resources to the physical world and vice versa. However, while the cloud is an excellent metaphor that helps us to discuss abstract and complex concepts about servers, networks, and data centers, we must always be cognizant of

the *physical nature* of the Internet of Things and the resulting challenges and limitations that brings.

For example, the physical location of cloud assets such as data centers can be critical in an IoT system. It's an all-too-common scenario to have data generated by devices in one region that cannot leave a certain geographic area for reasons of network latency, governmental controls, or even industry regulations. As a case in point, it's not acceptable for medical device data generated in Central Europe to be transmitted just anywhere in the cloud. The European authorities will tell you the data cannot transgress European boundaries. In situations like these, your abstraction of the cloud needs to be both regionally and locally aware.

Importantly, as we've discussed in previous chapters, digital messages are not immune from the inconvenient realities of the physical world. Cloud processing is far from instantaneous, subject to unforeseen delays and network unreliability.

The concept of fog or edge computing attempts to transcend some of these physical limitations. According to Cisco's "Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are":

> The Internet of Things (IoT) is generating an unprecedented volume and variety of data. But by the time the data makes its way to the cloud for analysis, the opportunity to act on it might be gone.

For example, when you have an edge device that needs to communicate with another entity—an application, business process, or even the device right next to it—it must first send that message to the cloud, which processes the information and transmits a response back down to the edge. Communication with the cloud could very well mean sending a data transmission thousands of miles away over a network with questionable reliability. In contrast, with fog computing, that processing happens on nodes *physically closer* to where the data is originally collected.

According to Cisco, fog computing provides intelligent infrastructure at or closer to the edge that:

- Analyzes the most time-sensitive data at the network edge, close to where it is generated instead of sending vast amounts of IoT data to the cloud.
- Acts on IoT data in milliseconds, based on policy.
- Sends selected data to the cloud for historical analysis and longer-term storage.

Flexible cloud topology like this can be a requirement in environments like hospitals, where there's a mission-critical need to quickly analyze real-time data from connected devices and initiate immediate action. In this kind of scenario, you'll want a multi-tiered approach to data collection and analysis, starting with a smart gateway or an IoT server to apply localized intelligence, aggregation, and business rules before communicating up to the next level. At the second tier, you might have a regional instance to aggregate big data, applying business rules and logic that spans local regions. Lastly, you'll have a fully distributed master system capable of seeing the complete picture, while collecting some trimmed down data for the purposes of analytics. The variety of IoT systems and the need for flexible solutions that respond to real-time events quickly make fog computing a compelling option.

# CHAPTER 5
# IoT Applications

Over the next few years, as the Internet of Things brings billions of new connected devices into the world, there is tremendous potential to unlock previously hidden insight into physical processes for both businesses and consumers. However, to access the value of all these new connected things, we require a host of new software applications that can make sense of the constant data flow.

Through embedded sensors and intelligence, we can give nerves to products, services, and operations. Now we need to think about how we're going to process the signals from our newfound senses. We certainly can't look at every bit of data these systems generate: information overload is already a substantial problem. Well-designed software and predictive analytics help us make some of those determinations, but we still must decide what machine and digital elements will make up the autonomic nervous system of our IoT solutions, and conversely, what will make up the somatic nervous system, requiring human intervention for critical decision-making.

Creating applications that derive value from the IoT involves much more than generating a user interface on a web or mobile device. While it's true that some IoT software may take the form of an app that gives a user a new experience on a smart, connected product, it could also consist of a prediction from an analytics model derived through machine learning and assimilated after an examination of multitudes of data, or it could mean the integration of a data feed from a connected operation into another business system.

IoT application design begins by establishing a *model* of the connected device and the worlds—both physical and digital—into which the device fits. This model enables the APIs, user interfaces, enterprise integrations, and analytics to access a common solution framework, even in the face of changing underlying technology. To effectively design and develop IoT applications, we need to model the specific problem domain, apply business logic, and surface that information to users.

In this chapter, we explore some of the considerations for IoT application architecture, as well as principles of good application design.

### A Collaborative Process

IoT application design and development is a collaborative process that could involve a wide variety of people—from domain experts and technologists to system designers and developers—all of whom have unique skills and responsibilities. For example, the domain expert possesses a deep understanding of the application subject matter, the system entities, and the relationships between them. She is responsible for the language of the application domain. The technologist/system designer is responsible for understanding and defining the application structure, and the developer is responsible for coding the application.

While each person has a specific assignment, it's important for everyone on the application team to collaborate across disciplines and understand the fundamentals of how the total system works. This knowledge is critical for identifying the constraints that govern an IoT solution, and ultimately providing a better experience for the user. For example, understanding the reasons why the network is unreliable gives the proper context for potentially faulty information flow that can up-end a user experience.[1]

## The Semantic Model

The IoT connects the digital and physical worlds so that smart devices and operations can provide feedback to users via a digital repre-

1 For a complete discussion of these considerations, it's worth having a look at *Designing Connected Products: UX for the Consumer Internet of Things*, by Claire Rowland, Elizabeth Goodwin, Martin Charlier, Ann Light, and Alfred Lui. O'Reilly Media, 2015.

sentation. When embarking on your software design, you should first ask, "What item am I connecting to, and how do I represent it digitally?" The answer to these questions varies wildly, depending on whether you're modeling a smart product or a more complex system such as a smart building, or even a smart city.

A semantic model is a good way to build a bridge between the physical thing and its "digital twin" so a user can understand the relationship. It gives a person a way to make sense of the physical device or operation that may be located at a great distance. Think of the semantic model as similar to an API, an application program interface for the object or system. It's a useful exercise to consider your connected product or environment as having routines and protocols. Ask yourself, "How do I want users to think about and work with this digital representation in the cloud?" The API of your connected device or operation defines the properties and services you want to expose to your application developers. Questions to consider include:

*Properties*
: What are the properties I want to expose? In the semantic model for a product such as a connected tractor, for example, these properties could consist of critical operational metrics like fuel capacity and usage, engine temperature, ground temperature, location (latitude and longitude), engine runtime, last oil change, oil level, fuel level, and engine RPM.

*Services*
: What are the services that my IoT device supplies? In our connected tractor example, users might need to get diagnostics and maintenance entries, and possibly even shut down the machine remotely.

*Events*
: What events does my device emit? The connected tractor might indicate that it's time for maintenance or send a fault code if there's a malfunction.

Ultimately, the people informed by your IoT solution are not going to be interacting directly with connected devices and services, but with the information through this shared model. The semantic model bridges the human brain and the connected device or system so we can deliver applications and user experiences that create value, spark insight, and enable useful action.

## Software UX Design Considerations

How do we consume data from the IoT effectively in a world already saturated with information? At their best, applications for the Internet of Things convey insights to decision makers and even automate responses, saving time and money by creating process efficiencies and improving system performance. At their worst, IoT apps can become glorified dashboards stacked with widgets and UI cruft, delivering nothing more than information overload. UX design in this area is fraught with bad metaphors (computer software for smart operations, for instance, should not at all resemble an aircraft cockpit) and misguided best practices (such as designing your software for the newbie rather than for the repeat user).

The purpose of your IoT application is to answer fundamental questions about your smart product or operation and allow users (or machines) to make immediate, informed decisions in response to the data. The time-to-decision matters. A well-designed application alerts users to problems when something goes wrong, and provides them with the right information to make decisions that have impact.

With many possible user types, from consumers to operations managers, system administrators to domain specialists, no two IoT solutions are ever the same, and one size never fits all. The key to designing an effective application lies in understanding both the needs of the user consuming the information, and the fundamental needs of the business, including the overall goals, decision requirements, and workflow.

What follows are some design tenets for balancing complexity, aesthetics, and information design in IoT applications—the design considerations, trade-offs, and other issues that you should work through as you create your software.

### The Right Design

What's the data story of your intelligent operation or smart product? The system is a living organism, and we're measuring its health: its unique strengths, capabilities, and metrics. For your IoT application, you may require an appropriate visual representation of the management layer, as well as easy ways to understand and drill down to the data level required to make effective decisions.

### Gather the right data

Oftentimes, easily accessible data is displayed in user interfaces rather than the data that's truly important to operational staff. Using a business and user-centric development approach coupled with advanced information visualization and aesthetic engineering best practices significantly upgrades the decision support capability of your UI. What data and metrics do the users really need to see? What context does each metric require to make it meaningful? Do users need to see the trend, the breakdown by region, or target?

### Provide the right data display

Detailed information should be easily accessible within the same interface. The user should be able to understand cause and effect, trends, and correlation to key drivers, thereby getting a clear picture of system status and how it got here–all in a single, integrated view.

### Enable mission-critical decision-making

Good information design matters. What is the visual representation that best communicates the metric? Not enough attention is spent on information visualization techniques and pushing business and operational critical data to decision makers.

What are the decisions the user needs to make? IoT applications should empower good decisions that have a positive impact on the individual or business.

To increase the pace of and instill higher confidence in decision-making, make sure your aesthetic presentation is contextually appropriate for specific roles to maximize clarity. The data display should be relevant, clear, and memorable.

## Machine Learning and Predictive Analytics

If we want the IoT data we collect and analyze to have real-world impact, machine learning, predictive modeling, and process adaptation are essential tools. Through this *learn-predict-adapt cycle*, software for smart products and systems can turn intelligence into action, moving beyond simple monitoring to anticipate problems and take proactive steps to improve our systems (Figure 5-1).

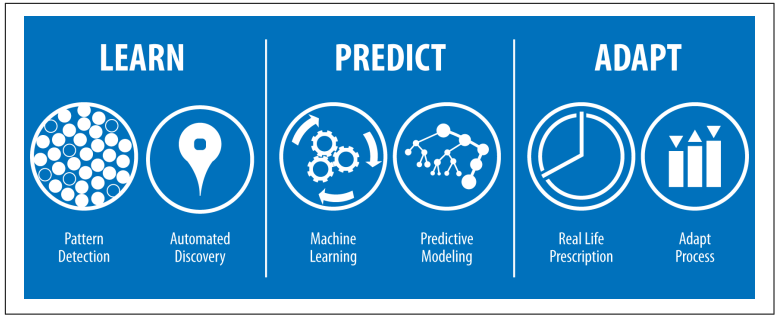


*Figure 5-1. IoT data analysis requires machine learning, predictive modeling, and system adaptation*

## Learning

Machine learning is an important tool for *data description and discovery,* particularly if you have an incomplete understanding of the classifications, ranges, or kinds of device data that make up a particular domain. Deciding which data aggregations make sense can often require a depth of understanding about a problem domain, which may or may not be available to you.

It can be difficult to understand exactly what a user needs to know, which is why *automated discovery* can be so important. For example, users may only care about a metric, like temperature, as an average over the course of a day. A machine-learning algorithm can help you understand when you need to consistently monitor a particular signal, or, in contrast, when you really should only care about discovering the outliers. Additionally, by integrating and analyzing information from multiple data sets, including those from third parties like weather and geographic information, automated discovery algorithms can help find valuable patterns that would otherwise remain hidden.

## Predicting

You may be familiar with the popular IFTTT (If This, Then That) service. The IFTTT pattern expects that a human has a priori knowledge of what to do. However, within any set of potential IoT events, the subset that we know what to do about is vanishingly small. Here in the data, then, are mysteries to uncover. Which are signals, and which are noise? Of the petabytes of data generated by

my devices, where's the insight I'm looking for? Where do we find it? In short, If This, Then What?

IFTTW seeks answers to questions like:

- Is it time for preventative maintenance?
- Is it time to adjust a configuration?
- Is it time to order consumables?
- Is it time to procure a new part?
- Is it time to send a repair technician?
- Should we change the product? How?

When you leverage an analytics engine with your IoT solution, you get not only a rearview-mirror, historical log of events that have occurred, you also get a predictive view of the future, based on an analytical model trained by that history.

Machine-learning algorithms can convert overwhelming amounts of data, billions of points of information, into clear patterns. By examining historical system data to understand what has happened in the past, machine-learning algorithms can discover predictive models that a human would likely never find.

These predictions help generate problem solving policies and methods—heuristics derived from experience with similar issues or scenarios—which can lead to important system adaptations. We can use this intelligence to affect the business process if we know that a problem or undesirable outcome is imminent.

As an example of this, let's look at an IoT solution for automated parking kiosks (Figure 5-2). Predictive modeling can help inform the parking lot's operations manager if a kiosk is at risk of failing, and alert him to take preventative action. In this scenario, a machine-learning algorithm examining historical data could detect, for instance, that a high volume of kiosk transactions per day combined with significant precipitation and an average outdoor temperature below a certain threshold signals that the kiosk's credit card acceptor is likely to fail within a short timeframe. This heuristic, derived from a group of unique situational factors and long-term data, would have taken a field service technician a decade of experience with the product to understand. That discovery, made by a

machine-learning algorithm, becomes a new input into the data model for the kiosk, a prediction that becomes part of its API.

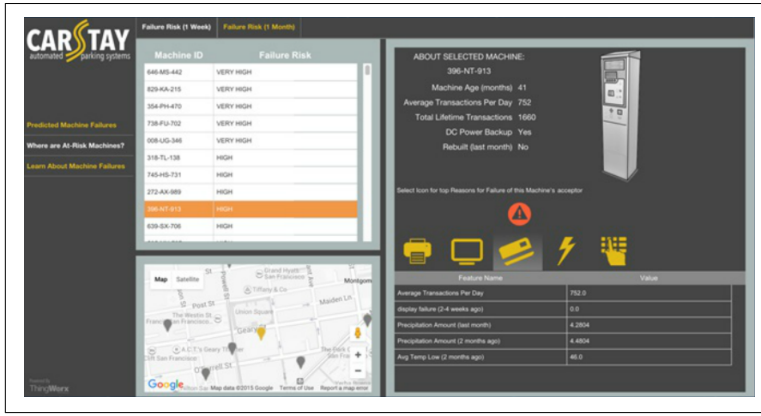


*Figure 5-2. A dashboard depicting data analysis for automated parking systems*

## Adapting

Not long ago, systems were designed with the pattern that looked like this:

1. Dump data into a database.
2. Every night, run queries against the new data and apply some business logic.
3. Dump the output of the batch operation into a database.
4. Run a report explaining what happened.

Unfortunately, this pattern doesn't get us very far in the faced-paced world of the IoT. As we've seen, the store-then-query approach of the past is not tractable, and moreover, not timely. We need to react to events in real time.

Predictive analytics can enable preventative action and facilitate real-time system adaptation, but only if the insights from machine learning are discoverable by users. For this to happen, it's important to create intelligent systems that are integrated tightly with the operational processes and technologies already in place.

In smart manufacturing, machine-learning algorithms automatically detect patterns and anomalies in complex production processes and

flag machines for maintenance before downtime or manufacturing errors occur. For instance, an algorithm might discover that when a facility experiences a drastic swing in humidity levels, machines operating above a certain temperature are most likely to fail. It's not enough for our IoT software to provide this heuristic in isolation; it must deliver the information at the right time to the right user. This might be achieved by integrating predictive information as a continuous data feed in the factory's existing production planning and supply chain management (SCM) systems. On days when humidity becomes an issue, the software surfaces a series of alerts. When the machines run at a higher temperature, workers can adapt operations accordingly by either increasing the dehumidifying processes or running the machines during cooler, evening temperatures. In this way, manufacturers can adapt their processes to better manage potential inhibitors to successful production runs.

## Rapid Application Development

You may (or may not) remember the revolution in development that came about when Microsoft first launched Visual Basic in 1991. Prior to Visual Basic, if you wanted to develop a graphical user interface (GUI) for software, you submitted yourself to strange and arcane forms of coding, with complicated APIs to draw basic objects like lines and rectangles. Visual Basic and other similar tools were responsible for the explosion of productive developers who were, for the first time, able to deliver consistent—if not particularly engaging—software interfaces. In many ways, Visual Basic helped drive the desktop PC revolution.

So, aren't software developers still using such tools today? Well, yes and no. In many ways, we have taken a few steps back in productivity, not because we no longer value *rapid application development* (RAD) tools, but rather because the types of user experiences and accompanying rendering environments have increased dramatically: from desktop to tablet, mobile, devices, and everything in between. Now engineers might require multiple sets of tools to create the same experience across platforms and devices.

With an *application enablement platform* (AEP), you can leverage the foundational elements of the edge, the cloud, and the model layers using a set of visual UI and functional design patterns designed

specifically for the IoT, and pre-wired for connected devices produced by a wide range of manufacturers.

Leveraging an AEP shields designers and developers from a degree of complexity by providing abstractions that allow for rapid development. A growing number of companies offer application enablement platforms, including ThingWorx (a PTC company) and Xively (from LogMeIn), among others, with Microsoft and SAP adding IoT capabilities to their product lines, as well. To be business-agile and ready to respond to changes, it's worth considering an AEP that's tuned for the application-building requirements of the Internet of Things.

# APPENDIX A
# Companies, Products, and Links

Throughout this book, we've discussed a variety of companies and products to illustrate important concepts in and approaches to foundational solutions for the Internet of Things. The following list of these companies and products, ordered alphabetically, contains relevant links to further information.

| Product | Company | Link |
|---|---|---|
| Arduino Uno | Arduino | *https://www.arduino.cc/en/Main/ArduinoBoardUno* |
| AirPrime MC Series Communication Module | Sierra Wireless | *http://www.sierrawireless.com/products-and-solutions/embedded-solutions/em-and-mc-series/* |
| Bathroom/ healthroom | Involution Studios | *http://www.goinvo.com* |
| CalAmp LMU 330 | CalAmp Corp. | *http://www.calamp.com/products/tracking-and-telemetry-devices/fleet-tracking-units/lmu-3030* |
| Cisco Connected Grid Router | Cisco Systems, Inc. | *http://www.cisco.com/c/en/us/products/routers/1000-series-connected-grid-routers/index.html* |
| Dell Edge Gateway 5000 Series | Dell | *https://www.dell.com/learn/us/en/uscorp1/secure/2015-10-20-dell-edge-gateway-5000-internet-of-things* |
| FitBit Surge | FitBit | *https://www.fitbit.com/surge* |
| Intel Galileo | Intel | *http://www.intel.com/content/www/us/en/embedded/products/galileo/galileo-overview.html* |
| JDLink | Deere & Co. | *https://www.deere.com/en_INT/products/equipment/agricultural_management_solutions/jdlink_telematics/jdlink_telematics.page* |
| K-Rain Indexing valve | K-Rain | *http://www.krain.com/indexing-valves/6000-series-indexing-valves-6-outlet.html?uasCatalog=1* |

| Product | Company | Link |
|---|---|---|
| Mimo Baby Monitor | Rest Devices, Inc. | *http://www.mimobaby.com* |
| Samsung Smart TV | Samsung | *http://www.samsung.com/us/experience/smart-tv/* |
| Grove water flow sensor | Seed Development Limited | *http://www.seeedstudio.com/depot/G12-Water-Flow-Sensor-p-635.html* |
| Smart Body Analyzer | Withings, Inc. | *http://www.withings.com/us/en/products/smart-body-analyzer* |
| Smart Pill Bottle | AdhereTech | *http://adheretech.com/* |
| Texas Instruments CC3200 Microcontroller | Texas Instruments | *http://www.ti.com/product/cc3200* |
| ThingWorx IoT Platform | PTC, Inc. | *http://www.thingworx.com/IoT-Platform* |
| Xively IoT Platform | LogMeIn, Inc. | *https://xively.com/whats_xively/* |

## About the Authors

As VP of IoT Technology at ThingWorx, a PTC business, **Joe Biron** leads a team that guides the technical architecture between the ThingWorx IoT platform and ThingWorx Ready partners. Joe has broad knowledge of IoT solutions, has led engineering teams focused on edge technology as well as cloud services, and has been deeply involved in the solution architecture for many ThingWorx customers.

**Jonathan Follett** is a Principal at Involution Studios where he is a leader of the firm's emerging technologies practice, working with clients like Partners HealthCare, the Personal Genome Project, and Walgreens. His work has been featured in the *Atlantic*, *Forbes*, the *Huffington Post*, and *WIRED*. Jon is an author of four books, including *Designing for Emerging Technologies* (O'Reilly Media, 2014), which offers a glimpse into what future interactions and user experiences may look like for rapidly developing technologies—from genomics and nano printers to workforce robotics and the IoT.