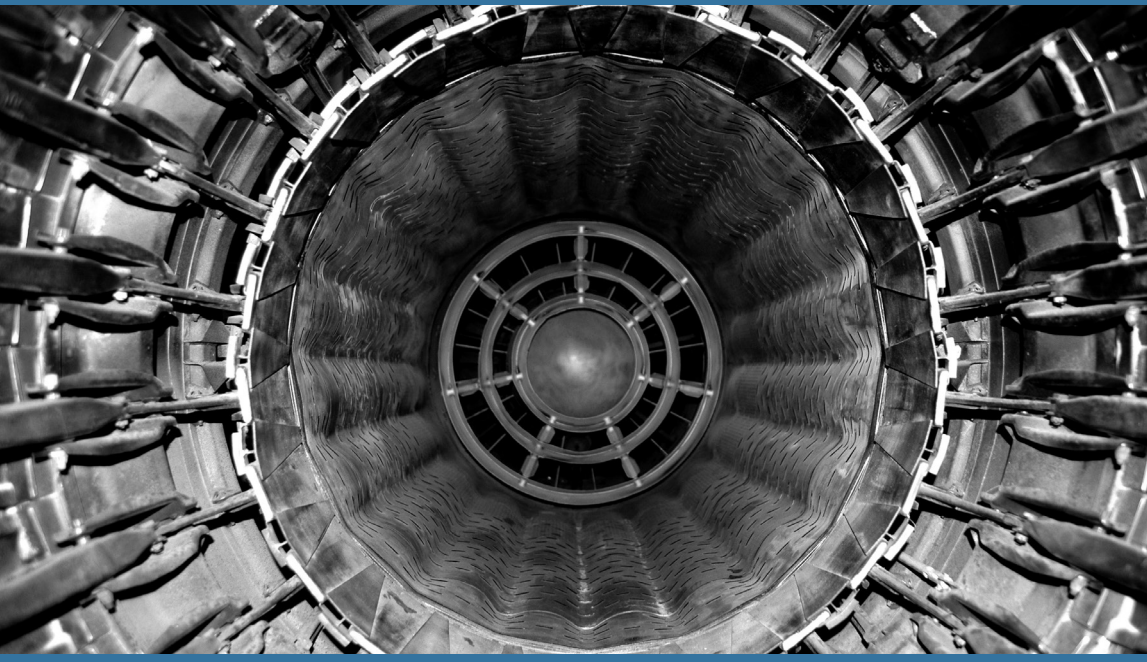
O'REILLY®

# What Is the Internet of Things?



Mike Loukides & Jon Bruner

# What Is the Internet of Things?

*Mike Loukides and Jon Bruner*

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo O'REILLY®

**What Is the Internet of Things?**
by Mike Loukides and Jon Bruner



Printed in the United States of America.

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

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://safaribooksonline.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or corporate@oreilly.com.

January 2015: First Edition

**Revision History for the First Edition**

2015-01-13: First Release



978-1-491-92180-7

[LSI]

# Table of Contents

# What Is the Internet of Things?

At our first Solid conference in 2014, we started a wide-ranging discussion about the "intersection between software and the physical world." A year later, this discussion is continuing, with even more energy and passion than before. You can hardly read a newspaper (if you still read newspapers) without seeing something about the "Internet of Things" and the rise of the hardware startup. But what does the intersection between software and hardware mean?

Early in 2013, we sat around a table in Sebastopol to survey some interesting trends in technology. There were many: robotics, sensor networks, the Industrial Internet, the professionalization of the Maker movement, hardware-oriented startups. It was a confusing picture, until we realized that these weren't separate trends. They're all more alike than different—they are all the visible result of the same underlying forces. Startups like FitBit and Withings were taking familiar old devices, like pedometers and bathroom scales, and making them intelligent by adding computer power and network connections. At the other end of the industrial scale, GE was doing the same thing to jet engines and locomotives. Our homes are increasingly the domain of smart robots, including Roombas and 3D printers, and we've started looking forward to self-driving cars and personal autonomous drones. Every interesting new product has a network connection—be it WiFi, Bluetooth, Zigbee, or even a basic form of piggybacking through a USB connection to a PC. Everything has a sensor, and devices as dissimilar as an iPhone and a thermostat are stuffed with them. We spent 30 or more years moving from atoms to bits; now it feels like we're pushing the bits back into the atoms. And we realized that the intersection of these trends —the conjunction of hardware, software, networking, data, and

intelligence—was the real "news," far more important than any individual trend. The Internet of Things: that's what we've been working on all these years. It's finally happening.

We've seen software transformed over the last decade by a handful of truly revolutionary developments: pervasive networking that can make the Internet a central part of any piece of software; APIs that make systems available to each other as abstracted modules; clouds like Amazon Web Services that dramatically reduce the capital needed to start a new software venture; open source projects that make expertise available to anyone; selling services rather than products. We now see the same developments coming to the physical world. As Tomasz Tunguz has said, the potential of the Internet of Things isn't just "linking millions of devices," any more than the potential of Google or Amazon was building massive IT infrastructure. The IoT is about "transforming business models" and "enabling companies to sell products in entirely new and better ways."

| | Software | New Hardware |
|---|---|---|
| Infrastructural services reduce capital requirements | Amazon Web Services | PCH |
| Open source improves access to expertise | GitHub | Thingiverse; Arduino; Robot Operating System |
| APIs let developers build on platforms | Twitter API | Smart Things; IFTTT |
| Ubiquitous connectivity | WiFi | ZigBee, Bluetooth |
| Data enables optimization | Netflix recommendations | Taleris (airline management) |
| Direct-to-consumer retail channels | Apple App Store | ShopLocket; Quirky; Etsy |
| Products sold as services | Salesforce.com | Uber; Zipcar |
| Hobbyists become entrepreneurs | Yahoo! | MakerBot emerges from NYC Resistor |

| | Software | New Hardware |
|---|---|---|
| Software intelligence decreases need for interaction | Google Now | Nest thermostat; Google driverless car |

## So What's New?

Hardware that has software in it isn't the least bit new. TVs and cars have had software-driven components since the 1980s, if not earlier. Chrysler introduced a computerized anti-lock braking system back in 1971. Microwave ovens, dishwashers, probably even those fancy beds that let you adjust the position of the mattress and many other settings: these all have hefty doses of microprocessors and software. So what's new? Didn't software and hardware converge a long time ago?

The hardware renaissance of the last few years entails more than just embedding CPUs into appliances. It's built on ubiquitous networking, which changes the game radically. Devices with embedded computers become much more powerful when they're connected to a network. Now we have networked televisions, networked loudspeakers that receive MP3s from a server, and networked devices that let us find our lost keys—and we call that the "Internet of Things" or the "Internet of Everything" or the "Industrial Internet," or "Fog Computing," depending on which vendor's language you like.

The functionality of new hardware depends on both modest, efficient local computing as well as powerful cloud computing. A fitness tracker, for instance, can carry out some functionality, like basic data cleaning and tabulation, on local hardware, but it depends on sophisticated machine-learning algorithms on an ever-improving cloud service to offer prescription and insight.

The Internet of Things crossed our radar back in 2001, at O'Reilly's first Foo Camp. It was an interesting idea: we wondered what would be possible if we could assign an IP address to every physical object. Back then, we were skeptical: who cares? Why would I want my shoes to have an IP address? That question is still useful, but the answers we're getting now are much different. In 2001, networked shoes sounded like gratuitous geekery, but in 2015, in the context of the Quantified Self, network-enabled shoes that log your every step

make complete sense. At the same time, the conversation has moved beyond easy-to-lampoon examples like connected refrigerators and has come to include important industrial and commercial applications for connected devices: jet engines, power plants, cars. In this context, the Internet of Things promises to make the world dramatically more efficient, safer, and more accessible. And while network-enabled refrigerators remain examples of techno-geek overreach, home automation is a hotbed of useful innovation. Keen Home is a New York-based startup whose first product is a network-enabled home heating vent. The company's founder, Will McLeod, believes that the most important devices to automate are the most boring ones, the devices that do something important (like control your home's heating and air conditioning), but that you never bother to touch. We've also seen smart ceiling fans. Will the Internet of Things give us more comfortable homes because our vents and fans talk to our thermostats? Yes.

Networking is hardly a new technology. Some industrial controls and building systems have had various kinds of network connectivity since the mainframe era, and local networks inside passenger cars have been commonplace since the 1980s. Remote supervisory control of utility assets has been a basic safety feature for decades. What makes networking in 2015 different is 20 or 30 years of Internet history: we understand how to build standard protocols, from the lowest layer of the hardware up through the applications themselves. We understand how to make devices from different manufacturers interoperate.

This ubiquitous connectivity is meeting the era of data. Since working with large quantities of data became dramatically cheaper and easier a few years ago, everything that touches software has become instrumented and optimized. Finance, advertising, retail, logistics, academia, and practically every other discipline has sought to measure, model, and tweak its way to efficiency. Software can ingest data from lots of inputs, interpret it, and then issue commands in real time.

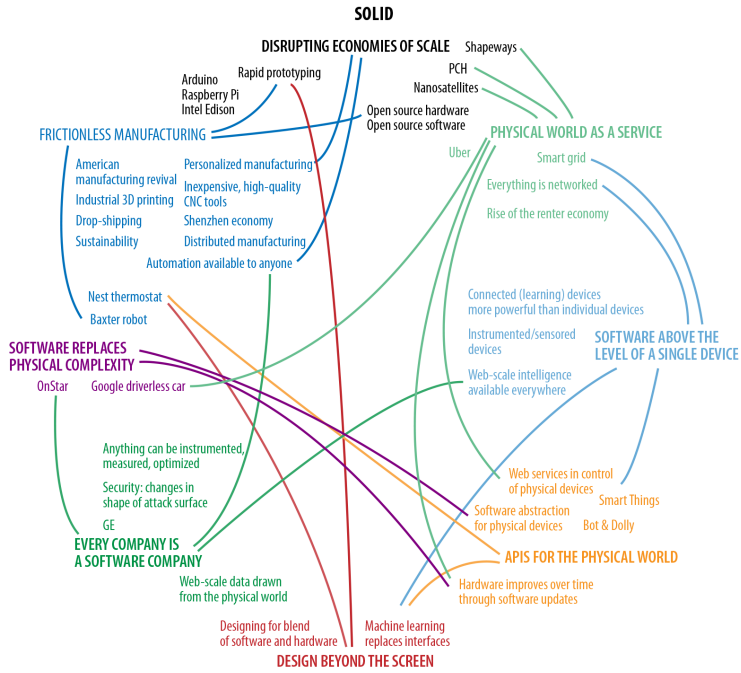
That intelligence is coming to the physical world now. Software in the cloud, operating above the level of a single machine, can correspond with millions of physical devices—retrieving data from them, interpreting that data in a global context, and controlling them in real time. The result is a fluid system of hardware and software.

Now let's add something else to the mix: the "new manufacturing." In the past few years, we've seen major changes in how manufacturing works: from the earliest prototyping stage through the largest mass-market production runs, 3D printers, CNC machine tools, design software, and new supply chain models are transforming the way that companies create and build hardware.

Many companies provide manufacturing services that can be used (with care) by the smallest of startups; and hardware startup incubators like Highway1 help make connections between local entrepreneurs and these manufacturing services. Small-scale CNC technologies make low-cost prototyping possible, and in the future, may bring manufacturing into the home.

Like software, hardware must be carefully designed, developed, and deployed—but we call deployment manufacturing. Software and hardware are merging into a single fluid discipline, with a single development process that encompasses both software and hardware. Few people need deep, low-level understanding of every module (just as few people need deep, low-level understanding of every software technology), but many people will soon need some integrated understanding of both hardware and software.

As we set out to define a program that encompasses the fusion of hardware and software, we recognized eight key concepts that are present in just about every interesting hardware project, from mobile accessories to airliners.




# Disrupting Economies of Scale

The new manufacturing lets the Internet of Things go viral. It's all about reducing the friction of getting a product to market, and it's analogous to using Amazon Web Services to launch new software at very low up-front cost. It enables entrepreneurs to prototype a product and bring it to market without investing tens of millions of dollars; all you really need is a successful Kickstarter or IndieGoGo campaign, and there have been many. We fundamentally don't believe that the world of smart, interconnected devices will be dominated by the entrenched industrial giants. Instead, it will be driven by network effects from widely distributed players. Taking the friction out of manufacturing disrupts the economies of scale. Just as the Web allows musicians to create and market their own music, to the dismay of the entertainment industry, the new manufacturing allows innovators to create and market their own physical products. You don't have to be Sony or Samsung to bring a product to market.

Small companies taking advantage of low-cost design, prototyping, and manufacturing can afford to be innovative. They aren't limited

to products that make sense to the large industrials. What can we do if we're able take the cost of a hardware startup down to $500,000, and the cost of a prototype down to $5,000? What if we can cut product development time down to four months instead of four years? There's a qualitative change in what you can do that mirrors the kinds of changes we've seen as software has become a universally-available tool. You can take risks; you can be experimental. This is a new kind of creativity, and it's worth looking at some of our favorite IoT poster children to think about how it works.

We've been fascinated by the Tile "lost and found" device. If you frequently lose your keys, you can buy a Tile tag for your keychain. It's roughly an inch square and contains a Bluetooth Low Energy (BLE) radio that allows an app on your phone to locate your keys when you lose them. Its battery is built in; when it runs out (over a year), you recycle the old one.

Tracing back the origins of Tile, we see a remarkably fast spin up. According to TechCrunch, Tile initially received $200,000 of angel funding. The product itself could easily have been prototyped with a couple of prebuilt modules with an enclosure made on a 3D printer. When the prototype was ready, Tile launched a SelfStarter campaign to raise $20,000 for the initial production run. It ended up raising over $2 million, which confirmed that the company was on the trail of something hot: crowdfunding doubles as market research, with the added benefit that it pays you. Jabil, a specialty manufacturer based in the US, is building the device; Tile doesn't have its own manufacturing facilities. Companies like Jabil also manage supply chain and fulfillment, so startups don't need to spend scarce attention on managing warehouses and shipping contracts.

The story of the Pebble watch is similar. Beyond the way it was made, or the margin by which its founders exceeded their $100,000 Kickstarter fundraising goal, the really interesting thing is its competition. A team of upstarts with no manufacturing capabilities managed to raise funds, design and release a product, and beat Apple, one of the world's most capable companies, to market by more than a year. Pebble spent vastly less than Apple in the process; and it's reasonable to guess that Pebble's presence significantly raised Apple's stakes, forcing them to redesign and rethink the iWatch. As in Web software—where low costs and constant clean-sheet hacking mean that one-person startups can release better products than

incumbents— the economies of scale in hardware are on their way to being abolished.

Additive manufacturing and other digital fabrication techniques are further disrupting the advantages that established companies hold. These technologies reduce fixed costs, and although they generally cost more per unit than conventional manufacturing techniques, they support highly customized products and make it dramatically easier for product creators to bring design refinements to market. Entrepreneurs are exploring markets for digitally-fabricated products from custom earbuds to children's orthotics.

Open qPCR is another project doing an end-run around established industries. Biohackers have had a number of low-cost PCR options for a few years now. The next step beyond PCR is qPCR, which incorporates both the PCR process and real-time analysis. qPCR machines are available from traditional lab suppliers for tens of thousands of dollars. But Chai Biotechnologies' Open qPCR has successfully designed and prototyped an open source machine, and raised funds for initial production of a $1,500 unit on Kickstarter. Do you want to detect infections diseases, identify food contaminants, and more on your desktop? Now you can.

The Internet of Things isn't just about tiny Silicon Valley–style garage startups, though. We don't expect the established industrial companies to stand still, and we don't want them to.

The industrial giants are already in the process of reinventing themselves. Automakers are racing to create app platforms for cars that will decouple the development cycles for entertainment, navigation, and connectivity from the much longer development cycles for cars. Ford's OpenXC data bus promises to open the company's cars to spontaneous innovation by developers who can think up creative applications for drive-train data. Nearly 30 automakers have joined Google in an effort to bring the Android operating system to cars. Initiatives like these open the doors to third-party innovation, but more than that, they recognize how our markets are shifting. Where is the value in a car? Is it in the chassis, the wheels, the body? Increasingly, the value is diffuse: it's in the software components and accessories that are layered onto the car. Building an open API onto the car's data bus recognizes that reality.

That shift of value into a blend of machine and software changes the business of producing cars. It simplifies inventory, since cars ship

with GPS and other accessories installed, which are then enabled or disabled by the dealer. No need to stock seven different accessory packages; the car becomes whatever it's supposed to be when it's bought. If the sport model has an oil pressure gauge and the family sedan has an idiot light, that's just a software setting that changes what's sent to a screen.

Such accessories can also be updated in the field. Functionality that's cast in stone years before the car reaches the market doesn't cut it, and automakers have a new opportunity to respond to market demand with much more flexibility. As software reaches deeper into integration with the drive train, even bigger upgrades become possible: Tesla adjusted the suspensions on every one of its sedans through an over-the-air software update in late 2013, and a month later adjusted the cars' chargers the same way.

Over-the-air updates aren't just for big items like cars. Nest's Protect fire alarm could initially be silenced with a wave of the hand. This feature made it possible to disable the device inadvertently. Rather than recalling the units, they were able to issue an automatic update that disabled "wave" control. Customers didn't need to do anything, which is as it should be. And all the devices were updated, not just the ones whose owners were willing to take them down and mail them in.

## Being a Software Company

Ford, GE, and other industrials are realizing that they are software companies, even if their products are shipped as tons of steel and aluminum. The big machines that make up the basic infrastructure of our lives—cars, power turbines, furnaces—have been refined over many decades. They're exquisitely optimized. The cost of further improvements in materials and physical design is high. Software intelligence of the sort that's commonplace on the Web is just starting to touch these machines, and it offers an entirely new set of improvements as well as a less expensive approach to existing features.

What does this mean for innovation? Considered purely as plastic or metal, it's hard to imagine significant improvements to garbage cans and street lamps. But a garbage can that's accidentally forgotten could notify the crew that is hasn't been picked up. A can buried by a snowplow could make its presence known. An autonomous

garbage truck could locate garbage cans without human assistance, and dump them without human intervention. Likewise, imagine a networked streetlamp: it could act as a public WiFi access point for anyone in the area, it could display notifications about bus service, weather updates, or whatever you want. It could even call the police if it "hears" an auto accident or a human cry for help.

These ideas require looking at a garbage can or a streetlamp as a node in a software system. In turn, the companies that make them need to think of themselves not as organizations that stamp steel, mold plastic, or forge iron, but companies that develop software. That organizational change is happening, and corporations that don't get it will be left behind. Those that do get it will find that they can be more creative, more novel, and more impactful than many of the companies that we conventionally call "tech companies."

Insurance, which Tim O'Reilly has said might become "the native business model for the Internet of Things," illustrates this transformation even in non-tech companies that don't have anything to do with manufactured goods. In 2008, Progressive Insurance made behavior-based car insurance pricing available nationwide. Customers who agree to install a monitoring device on their car's diagnostic port get a break on their insurance rates and pricing that varies according to usage and driving habits.

Connected hardware, by reporting conditions as they actually exist rather than as they're predicted to exist, could transform insurance markets. Insurers who understand hardware can use it to mitigate risk, perhaps underwriting monitoring and safety infrastructure in the process, and compete effectively. These companies that once existed strictly in the abstracted world of finance must now become tech companies expert in connected devices and their possiblities.

## Frictionless Manufacturing

One of the biggest trends enabling the Internet of Things (and all its variants) is what we've called the "new manufacturing." The new manufacturing is really about removing the friction that makes hardware difficult.

We've already mentioned offshoring in China, and seen that it is now an option for small companies and even individuals—a dramatic shift from just a few years ago. While working with overseas

vendors can be hard for a small startup, companies like PCH can now handle all the complexity, including supply chain management, production, and quality control. PCH's startup incubator, Highway1, is dedicated to helping hardware startups get off the ground. Highway1 works with companies as they go from the prototype stage to manufacturing, dealing with offshore vendors, financing inventory, and learning to manufacture at scale. Large production runs of high-quality hardware used to be territory reserved for established industrials; no more.

New tools have also taken the friction out of the design and prototyping process. The current generation of software tools from established vendors like Autodesk and new projects like Makercam make it easy to generate 3D models for your product, and they're increasingly bridging the gap between computer and prototyping machine, working seamlessly with drivers for 3D printers and CNC tools. We're now seeing more advanced home printers that can deal with metals and carbon fiber. We're also seeing home computer–controlled milling machines. And while home 3D printers are great, milling machines are magical. When they're combined, the capabilities of even a small machine shop start to approach those of a large industrial plant, at least for small runs.

Following additive and subtractive manufacturing to the desktop, automated electronics assembly is rapidly becoming accessible, too. Tempo Automation is developing an inexpensive desktop pick-and-place machine that makes small-run assembly of surface-mounted electronics possible. Tools like this development loops that might last for days when working with an outside contract manufacturer and condense them to hours. The result will be software-like development cycles and a rapid expansion of the problem space that electronics can address.

It's tempting to think of the reduction in friction as a technological revolution, but there's nothing here that's really new on its own. Computer-controlled milling machines have been around for decades, as have 3D printers and CAD tools. What's different is that the tools are much more accessible. Few people could create a design on 1980s CAD tools; they were highly specialized, and had difficult user interfaces that required lengthy training. Now that hobbyist 3D printers are under $1,000, CAD vendors have realized that to flourish, they have to make tools that are very capable, easy to use, and inexpensive. Just about anyone could create a design using Autode-

sk's iPad apps, many of which are free. Tool vendors are learning that consumer grade is the new professional grade; professionals now want the same ease of use that consumers required.

For applications where a professional-scale printer is still necessary, there's always a supplier ready to make the stuff for you. Mark DeRoche, founder of Aerofex, was able to design large carbon fiber parts for an airframe on his laptop, ship the design files to a fabrication facility, and receive samples within days. That's a reduction in friction. Just as Amazon Web Services revolutionized data centers by giving startups access to computing services that they could never afford to build, custom fabrication shops give hardware startups access to tooling and equipment that they could never buy.

We may never eliminate friction entirely. The real world has constraints that you just don't find in software development. Still, the kind of software that can understand and deal with real-world constraints is becoming available to individual prototypers. Programs that can map out toolpaths, identify conflicts, and find ways to work around them are now a matter of running an open source script on your laptop.

Amateurs won't start fabricating their own processors or jet engines any time soon. True expertise is as important as ever, but expertise is becoming available in modules, rather than in rigid take-it-or-leave-it packages. If you know something about design and marketing but not much about manufacturing, you can have PCH handle that part—obsessing over suppliers, multiple-injection processes, and logistics—while you keep your mind on product vision.

## Software Above the Level of a Single Device

The Internet of Things is bringing software above the level of a single device, one of the key advances in Web 2.0, to the physical world. It's not remarkable that a thermostat, an airplane, or a cellular tower is "smart" (in the sense that it has a microprocessor in it that performs some sort of realtime control). Real intelligence comes from interconnections between devices and to lots of different kinds of software—as when an electric car starts charging at night because software at the level of the grid signals that electricity prices have gone down (a process that is itself kicked off by lots of networked measurements across the grid).

To take advantage of that kind of intelligence, hardware companies are now building their products with explicit APIs. Devices from Philips Hue light bulbs to John Deere tractors have published APIs that let developers find novel uses for their products and turn them into platforms. The result is a kind of modular comparative advantage: the best heavy-equipment maker can build tractors, and software engineers can find ways to integrate them with the best software.

Whether we're talking about clothes and a dryer, GPS and maps, home lighting, or garbage cans, we're talking about software systems that run across many devices. Smart light bulbs are somewhat interesting; they become much more powerful when they're connected in a system with other devices. It's relatively simple to build lighting that follows you from room to room. The individual light bulbs are controlled by a program that also reads motion detectors scattered throughout the house. You might find lights switching on and off as you move a bit creepy, but it's not hard to do.

Google Maps is an excellent example of software running across multiple devices. Although Maps looks like it's just displaying a map on your phone and plotting your location on the screen, the service's amazingly accurate realtime traffic data hints at much more. Those traffic conditions don't come from helicopter reports—they come from millions of Android phones, each of which is reporting its status to the servers. If the phones are moving, traffic is flowing. If the phones on Route 101 are moving at 5 miles per hour, there's a traffic jam. Your phone isn't just displaying traffic conditions, it's also reporting traffic conditions, as part of a much larger system.

We will see many more systems like this. Electric cars, air conditioners, and other appliances that consume a lot of power can communicate with a smart electrical grid and optimize their use of power. You'll program them with your preferences, which could be "don't run when electricity is expensive." Once you have this ability, it makes sense for the electrical company to give you discounted rates for moving your power consumption to off-peak hours. Why dry your clothes during the day when electrical use is at a premium? If you're starting and stopping the dryer by hand, shifting your usage to off-peak is at best a pain, and probably impractical. But a smart dryer can take care of that for you by participating in a network of devices that works above the level of any single device.

If you've driven around much in California, you've probably seen large wind farms for generating electricity. Optimizing the location of the windmills and adjusting the pitch of the blades for each windmill in the farm is a difficult problem. In a wide-ranging post on GE's Edison's Desk blog, Colin McCulloch explains that every turbine is in its own unique environment, determined in part by the turbines around it. His solution is "both automatic and remotely executed"; the optimal configuration for a turbine is computed remotely and downloaded to the turbines automatically as they run. The turbines sense conditions around themselves and transmit data to GE's cloud, which optimizes a solution across the entire farm, and transmits it back to the individual turbines. "Every turbine can be… tested simultaneously, and each will receive optimized parameter settings unique to its own micro-environment." Software above the level of the single device indeed!

## Standards for Connected Devices

A network isn't just a bunch of devices that magically communicate with each other. We knew how to do this back in the 1970s. The brilliance of the ARPANET was the idea of protocols that were both standard and public, so that anyone could build equipment that could interoperate with everything else.

We take this for granted now, but think what the Web would be like if you needed one browser to watch the news on CNN, another to use Facebook, and another to watch movies on Netflix. Imagine if these browsers didn't even run on the same hardware. That was more or less the situation we lived in until the 1990s. You used a TV to watch the news, a radio to listen to music, and you connected a VCR or DVD player to your TV to watch movies. There were standards, but the standards were so specific that they didn't standardize much. One box could never do what another box did.

Standard protocols turn devices into platforms. As the networking protocols that became the Web achieved dominance, we started to understand that you didn't need specialized equipment, that a laptop with a high-definition monitor could be a better TV than a TV, and that both producer and consumer would be better off if any browser could interact with any web service.

We need to do the same thing for the Internet of Things. Imagine a house full of network-enabled light bulbs. As Matthew Gast points

out, many layers need to be standardized. What do we want to do besides turn them off and on? Is there other information we want to carry? Philips' Hue lets you control the intensity of each color component. There are many, many ways to use that capability: do you want lights that adjust themselves to the music you're playing? Do you want lights that turn off when they detect that you're in bed? Should lights dim when they detect trace compounds from a bottle of wine? A light bulb (or any other device) could sense contextual information and transmit it back to the application.

But fascinating as the possibilities are, none of this can happen meaningfully without standards. You probably have many light bulbs in your home. You don't want to buy them all from Philips; you'd rather take advantage of local sales and deals as they come up. And you probably don't want different apps for controlling the Philips bulbs in the kitchen, the GE bulbs in the living room, and the Sylvanias in the bedroom. Just remembering which bulbs are where is a huge headache.

But more importantly, the most brilliant applications for smart light bulbs are likely to come from some unexpected source other than light bulb manufacturers. A sleep expert, for instance, might develop lighting cycles that help us get better sleep, or a maker of entertainment systems could find a new way to make light bulbs work in concert with movies and music.

APIs need to enable entire systems to work together: if you want your security system to turn on your lights when it thinks something is wrong, you need a software layer between the two that has access to both. In the same way, an airline that uses instrumented jet engines won't realize their full potential until it connects them with its human resources, ticketing, and weather databases to completely optimize its operations.

In a world where every vendor rolls its own communications and hopes that the winner will take all, you end up with a mess: incompatible light bulbs from different vendors with different capabilities and different control software. There is no "win" here; everyone loses.

We've seen a few steps in the right direction as the market demand for standardization and openness has become clear. As soon as Philips made its Hue bulbs available, users began reverse-engineering them. Philips, sensing demand and seeing opportunity in giving

developers a platform, released its own official API for the bulbs in early 2013. But this is only a start. Independent efforts, like Thing System from Alasdair Allan and Marshall Rose, promise a single centralized API for controlling many unstandardized devices. Zigbee offers a home automation standard for controlling many different types of devices, including lighting, locks, alarms, shades, heating, cooling, and electrical power. It's unclear how widely this standard has been adopted, though their alliance has roughly 150 members, including Philips.

## Design Beyond the Screen

Software has been steadily wandering away from the PC since the early 2000s—first into mobile phones, then into tablets. Design had to change as consumers' modes of interaction with software changed, but mobile software carried over a great deal of PC vernacular: you click on icons to launch applications, tap on text fields to start typing, and bring up toolbars to change settings. The coming explosion of intelligent devices will require a more radical change for designers as they're asked to create systems that blend software and hardware into entirely new kinds of interfaces.

The Misfit Shine activity tracker, for instance, has no screen and no buttons. Users interact by tapping it and reading patterns from its twelve LEDs: double-tap and the device first shows activity progress, then displays the time in a sequence of blinks. (Full disclosure: Misfit is a portfolio company of O'Reilly's sister venture capital firm, OATV.)

Many devices go beyond omitting the conventional screen and keyboard; they have interfaces that attempt to forego interaction completely. Connected devices can apply machine learning and use data gathered above the level of an individual device to anticipate needs and preempt them. The Nest has an intuitive interface that's easy to use, but its real brilliance is that, after a few weeks of occasional adjustment, the typical user won't have to interact with it at all. Beautiful as the Nest is, it's not an iPhone. Its goal isn't to make you spend time with it. Its goal it to disappear into the infrastructure, to be ignored.

Similarly, an impact-sensing football or hockey helmet is ultimately just a helmet. It can't have any more of a "user interface" than a traditional helmet. It just collects data and reports it back to a cell

phone. And the cell phone itself should remain quiet, unless there's a dangerous impact or the coach wants a summary report. The technical sophistication of the helmet can't call attention to itself. If it doesn't disappear, it's a failure.

The Internet of Things demands that we think about what we want from our devices: what are they for, what do we want to accomplish, and how can they assist us and get out of the way. We don't necessarily want artificial intelligence, as Kevin Kelley argues. We want artificial smartness, smart assistants that can help with specific tasks, like Palate Home, which promises to turn you into a restaurant-quality chef. Its goal isn't to be amazing, it's to make you amazing. Doing that requires rethinking user interfaces. A cooking appliance with hot surfaces is a hostile environment for a touch screen. And, unlike the Nest or a helmet, this appliance requires some interaction with the user: it needs to know what you want to cook, it needs to tell you when it's done. There's one control on the device itself; otherwise, it communicates with you through your phone. Just as the browser became a generalized user interface for Web 2.0, the smart phone is becoming the standard user interface for the Internet of Things. But what is that user interface? Just text? A smart phone supports many UI options that didn't exist for Web 2.0. Voice, video, SMS: it's all fair game, as designers try to craft user experiences that do what's necessary and then get out of the way.

When devices combine hardware and software, it is crucial for designers to understand the constraints of both, and to understand manufacturing well enough to turn it into a design input.

That's a staggering order: web design alone is complicated enough. The good news is that modular approaches and new ways of engineering, making, and delivering products mean that no one person needs to do low-level engineering in both hardware and software on a reasonably simple product. Software can ease the expertise challenge by handling some of the complexity of design. Even browser-based design software and mobile apps can produce refined designs and send them to fabrication services and CNC routers. Iris, a robotic cinematography platform from Bot & Dolly, includes a Maya plug-in so that production designers can control industrial robots through tools they're already familiar with—no expertise in industrial controls needed.

Design has always been important, but in the past year, we've really recognized its centrality. Apple's success has shown that design is a key business asset. Nest would have failed if their thermostats weren't practically art objects. Designers boil experiences that could be maddening and complicated down into experience that are simple and beautiful. A well-designed object anticipates the user's needs and improves the experience, rather than getting in the way. Designers don't just make things pretty after the engineers have done their job. They change the world.

## Everything as a Service

As software becomes more deeply integrated in the machines around us, it brings the software-as-a-service model with it, supporting business models that sell physical devices incrementally. Software as a service recognizes comparative advantage: Amazon is better at running a server center than you are, so let Amazon run your servers while you focus on building your product. Machine-as-a-service builds on the same idea: let a company that knows how to manage a car's life cycle own and maintain cars, and rent them by the mile as needed.

It also realigns incentives, letting companies that build machines capture the value of their longevity. GE, as a manufacturer of jet engines, is good at maintaining them, and it's in a position to realize value from its maintenance contracts by building jet engines that are reliable and easy to maintain. Consumers aren't always very good at investing in long-term value, but institutions with longer horizons and better information can afford to make those investments, and they'll be able to take advantage of physical-world-as-a-service models.

Here's an example. We've already mentioned that the portion of a car's value corresponding to software-defined features (navigation, entertainment, comfort) has become higher in the last few years, and a software improvement to those features can improve the value of a car. About a third of new cars in America are leased rather than bought outright. When a lease ends, its underwriter has a used car on its hands that it needs to sell. By making more of a car software-defined—and therefore easily upgradable long after it's sold—automakers have been able to improve the resale value of their cars and lower lease prices as a result.

The idea that everything is a service is changing the way we do business in other ways, pushing web-style business models into industries that haven't really touched the web before. Just as the Cover app handles restaurant bills automatically—you walk in, order, eat, and walk out, and the app pays the bill without involving you at all—a smart car could also pay for its own gas. GPS tells the car where it's located; maps identify the particular gas station; and a billing API, like Stripe, processes the charge. Automated payments could prove essential to new business models. Cover handles simple problems, like splitting the bill with your dining partners. As a "sharing economy" becomes more real, and as cars become shared goods rather than personal property, cars will need to "split the tab" between users according to the miles they've driven. (Full disclosure: Cover is a portfolio company of O'Reilly's sister venture capital firm, OATV.)

The next step forward in implementing the physical world as a service is allowing the users themselves, rather than professional developers, to build the connections. We see a hint of that in If This Then That (also known as "IFTTT"), a service that allows users to write very simple "recipes" that connect their services (called "channels"). Most of the channels are typical software services: you can use IFTTT to post your tweets to Facebook, for example, or to get a text message when a package you're tracking changes status. They've been adding hardware services: you can use IFTTT to script your Nest, your Android Wear devices, your own LittleBits creations, and other devices (WiThings, Jawbone, lighting, electrical outlets, and more). While scripting your environment may sound like leading edge geekery, IFTTT's programming language is extremely simple: it's really just "if this, then that" (if something happens, do something). It's the logical plumbing for the Internet of Things. If the physical world is a service, then you need to be able to use those services on your own terms.

## Software Replaces Physical Complexity

When software and hardware are tightly joined, you can choose whether to handle a problem with bits or with atoms. Take a physical problem and make it a control problem, then handle with software—or avoid the computation step and handle a problem with a mechanical contrivance or analog circuit.

Baxter, the $25,000 manufacturing robot from Rethink Robotics, uses powdered-metal gears rather than machined gears in order to keep costs down. But these less-expensive gears have more backlash—movement due to gaps between the gear teeth—than the more expensive gears used in traditional (and more expensive) industrial robots. Rethink's answer: to model the backlash in Baxter's software and preempt it. A single upgrade to Baxter's software last fall made the robot 40% faster and twice as precise.

Consider another physical problem that can be made into a control-system problem easily and effectively: a furnace. Making a furnace more efficient might involve improving its materials or its physical design. Those approaches entail substantial engineering efforts and a reworking of manufacturing processes—and do nothing for furnaces that are already installed. But add software to the thermostat that controls the furnace and you can improve its efficiency dramatically by simply optimizing the way it's switched on and off.

Makani Power uses autonomous kites to generate electricity more reliably than conventional turbines. The ability to control a kite through on-board processors lets it replace a huge, static support weighing many tons. As a result, a kite requires much less material than a wind turbine, and can take advantage of stronger and more consistent winds at higher altitudes. Makani's founder, Saul Griffith, calls it "replacing mass with math".

Josh Perfetto of Open qPCR states this principle succinctly: "So much of the performance and usability of our machine comes from software." To him, that's their core advantage: they can upgrade their machines through frequent updates, and deliver updates through the web. "Simple stuff, I know, but our competitors just don't do it."

# Roadblocks on the Information Highway

Any new technology is subject to unintended consequences. The convergence of hardware, software, and networking is no exception. Here are a few of the problems we see surfacing as we enter the era of networked devices.

## Security

The convergence of hardware and software changes the problem of computer security in terms of both increased exposure (through

many more connected nodes) and increased acuteness (through dangerous equipment that wasn't connected to the Internet a few years ago). Could a terrorist open a gas valve through a remotely-supervised industrial control system? Injure hospital patients by hacking into connected life-support machines?

Yes, in theory. But those scenarios, as scary as they are, must be balanced against the benefits that are at the heart of a decision to connect something: connecting a machine is often about improving operational safety even as it increases the chance of a network attack. Remotely-monitored gas pipeline valves are much safer than unmonitored valves. The small risk of an intentional attack is outweighed by the certainty of a faster response to an equipment failure. Networked hospital systems can improve patient outcomes by anticipating changes and alerting doctors. And, in many cases, Internet-enabled intrusions are more difficult and more fanciful than the traditional, physical, intrusions. Someone could open your door remotely, but it's already relatively easy to pick a common lock.

The real problem with smart hardware is that the scope of a potential attack changes. Unlike a physical attack, a network attack can be programmed. You could potentially unlock doors by the thousands by discovering a single vulnerability—a significantly different problem from the world of thieves and pranksters working one trick at a time.

We've seen an inkling of that already. Stuxnet spread through Iran's nuclear agency by exploiting a print spooler vulnerability in Windows, propagating itself until it eventually found an opening into the Siemens industrial-control software that ran Iran's centrifuges. A recent attack targeted network-enabled refrigerators. The goal of the attack wasn't to melt ice cubes or spoil milk. Who would care? This attack was all about building a botnet for sending spam email. And while spam may feel like a 1990s problem, in this case it demonstrates that the security game isn't what it used to be. Similarly, there's a malware kit that uses devices from thermostats to home routers to launch massive distributed denial-of-service (DDoS) attacks. Unlock 10,000 doors, yes—but the real goal probably isn't what's behind those doors.

Likewise, security expert and O'Reilly author Nitesh Dhanjani has demonstrated an attack against the Philips Hue light bulbs. The significance of this attack isn't its prank potential. It's that an attacker

could potentially turn off the lighting in a hospital, or even cause a widespread blackout by disabling the individual bulbs. Attacking a power plant is risky, difficult, and most damage is relatively easily repaired. Disabling millions of devices across a large city is a different kind of distributed damage that may well be more devastating. The greatest danger of the Internet of Things is that it changes the nature of an attack; we're now playing a distributed game where any vulnerability is multiplied by thousands.

## Privacy

When all our devices are interconnected, we become interconnected. If your refrigerator knows what you buy, will it tell your insurance company about your diet? If your washer knows what kind of clothes you wear, will it sell that information to marketers? Will all this data end up in the hands of the NSA, and if so, what will they do with it? Everything we do casts off information in a kind of data smog. In the past, we haven't had the tools to collect and make use of that information. Now we do. Every device on the Internet of Things is both a data generator and a data-collection point.

We don't yet understand all the implications of being connected. In some respects, our concern with privacy is a creation of the 1950s and the anonymity of the suburbs. We were shocked when Target figured out a teenager was pregnant, based on her buying patterns, and sent her advertisements for baby products before her parents knew. One of your authors has argued that a small-town pharmacist would have made the same observation in the 1930s or 40s. But the stakes are higher now. What does it mean for Google to know how you've set your thermostat? In commercial and industrial applications, could connected devices amount to industrial espionage?

Think back to any of the fuel crises of the past 50 years, when people were asked to reduce their thermostats. Think of the current drought in California, and Internet-enabled devices to control lawn sprinklers. Do we want enforcement agencies to subpoena water sprinkler data? Data is already a useful tool for enforcing regulations about how we use scarce resources. But the larger point is that we don't know how our data will be used, and we don't know the consequences of using it. And we don't know what our laws mean, or how they will be enforced; after all, collecting the search histories of virtually every citizen (to say nothing of motion-sensor data in our homes) wasn't even remotely feasible a few years ago.

The "we don't know" is scary. Social norms surrounding privacy are certainly changing, and it would be a mistake to impose the 50s-based culture that we grew up with on the future. But it's equally mistaken to pretend that we aren't facing critical issues. When all our things are watching us, who watches the watchers?

## Network Effects

Will our current network technology survive the convergence of hardware and software? It can support very high data rates, up to 100Gbps for short-haul wired Ethernet, and well over 100Mbps for wireless. But that's not the problem. Our current technology makes an implicit assumption that devices on the network transmit relatively large chunks of information. So far, we've rarely, if ever, felt the effects of that assumption. A device that wants to send a few bytes at a time isn't a big issue, because there aren't many of them. What do we have on a typical home network? So far, a few laptops, some cell phones and tablets, some printers, IP-enabled cameras, and a few other devices—a few dozen at most. An office might be 100 times that.

But what if every conceivable device was network-enabled? What if every square foot of wall had a temperature sensor, every item of clothing had some sort of network tag, every light bulb, and so on? There could easily be thousands of devices in a home, and millions in an office. What if these devices are relatively chatty, and send lots of small pieces of data? Could our current network survive?

Probably not. The MQTT protocol is a relatively unknown part of the TCP/IP family that is designed to be implemented on relatively simple devices that transmit small chunks of information, in applications where network bandwidth is at a premium. IBM has released an open source implementation of MQTT. Cisco's fog computing is about providing the infrastructure necessary to push intelligence out from a centralized "cloud" to the edges of a network. (It's perhaps ironic that this was the original Internet vision, before we became addicted to huge centralized data centers.) When there's more intelligence at the edges of the network, there may be less need to send data into the center. There's certainly a need for more sophisticated routing and switching in the customers' premises. And at the October 2014 Strata + Hadoop World conference in New York, Anna Gilbert spoke about how smart devices at the edges of a network can preprocess and precompress data before transmitting it.

At the lower levels of the network, Google, Samsung, and several other companies have formed the Thread Group, which is a consortium to develop a low-power mesh wireless network that can interconnect large numbers of devices. Since Thread uses the same frequencies and chipsets as Zigbee, existing products can be converted with no more than a software update.

Whether MQTT, Fog Computing, and similar efforts from other vendors solve the problem remains to be seen—regardless of advances in protocols, we'll still need much bigger tubes and better ways to connect to them through both formal and ad-hoc means.

## Patents

It's unfortunate that we have to bring up intellectual property, but we do. If standard APIs for the physical world enable the Internet of Things to be productive, and not just a mess of curious gadgetry, then the idea that APIs are patentable, as Oracle argued in its lawsuit against Google, is pure poison. Fortunately, the courts have ruled against Oracle once, but the case has only started the appeals process. Could Philips sue another company that implements its light bulb API? If so, that will be the death of smart lighting.

It's not just about APIs, though; whether for hardware or software, most patents are written so vaguely that they any significant new technology is covered by some out-of-date patent. Nest has been sued by Honeywell and several other companies for patents such as using a circular control to select a temperature. As Nest argues, the point of this litigation isn't about patent royalties—it's clearly to stifle innovation in an industry that has scarcely changed in decades.

Startups are particularly vulnerable in patent fights, since they have little money to spend and would rather not spend it on lawyers. More than a few startups have ended under the threat of patent litigation. (Now that Nest has been bought by Google, they have little to worry about.) Whether or not there's any merit to the case, if a young company can't afford to fight, it loses. It would be tragic if the convergence of hardware and software, driven by small companies taking advantage of frictionless manufacturing and pervasive computing, were to come to an end through the patent courts. We may be seeing a disruption in economies of scale, but legal fees are driven by scale, and have no respect for disruptors.

## Inventing the Future

We haven't invented the future (yet)—we're still inventing the tools for inventing the future. We haven't taken any of the innovations we've discussed as far as they can go. We're a long way from smart dust; home manufacturing is almost a reality, but there's still friction in the manufacturing process. We haven't yet solved supply chains, order entry, or fulfillment, though there are contractors and services ready to handle all these problems, for a fee. And most of our ideas about what to build are still relatively uninspired.

Will frictionless manufacturing lead to a new industrial revolution? Probably so, but we don't yet know. Will affordable hardware innovation reinvigorate economies both in urban centers like Detroit and in rural areas that have long been abandoned by the factories of the first and second industrial revolution? We hope so, but it's a long way from hope to reality. Will our creations enrich our lives, or will we degenerate to couch potatoes searching the Web, until we're eventually jettisoned on a spaceship bound for nowhere? We hope it's not the latter. As Kelsey Breseman says in an interview with O'Reilly Radar, when our devices are smarter, they will enable us to "stop interacting with our devices, stop staring at screens, and start looking at each other, start talking to each other again."

We're still at the beginning. We can see some of the possibilities, but we're more aware of limitless possibility than of any particular thing we might create. We're excited by the idea of a Tile chip and a Pebble watch, even though both of us rarely lose our keys and don't wear watches much. These particular products aren't important in themselves; what's important is that they're signposts pointing toward a creative future, filled with products we can't yet imagine.

We don't know where we're headed, but we never have. We're in it for the journey.

## About the Authors

**Mike Loukides** is Vice President of Content Strategy for O'Reilly Media, Inc. He's edited many highly regarded books on technical subjects that don't involve Windows programming. He's particularly interested in programming languages, Unix and what passes for Unix these days, and system and network administration. Mike is the author of *System Performance Tuning* and a coauthor of *Unix Power Tools*. Most recently, he's been fooling around with data and data analysis, languages like R, Mathematica, and Octave, and thinking about how to make books social.

**Jon Bruner** is a data journalist who approaches questions that interest him by writing and coding. Before coming to O'Reilly, where he is editor-at-large and co-chair of the Solid program, he was data editor at *Forbes Magazine*. He lives in San Francisco, where he can occasionally be found at the console of a pipe organ.