DZone

THE DZONE GUIDE TO THE

INTERNET OF THINGS

2015 EDITION

BROUGHT TO YOU IN PARTNERSHIP WITH

aeris CloudOne GE Software ThingWorx A PTC Business

Dear Reader,

"When wireless is perfectly applied, the whole earth will be converted into a huge brain, which in fact it is, all things being particles of a real and rhythmic whole."
-Nikola Tesla, 1926

The general idea of the Internet of Things is old. The new parts are the capabilities of the nodes and the edges, the dendrites and the axons, the terminals and the network. "IoT" captures a threshold, not an invention. Whether IoT is fluff or reality is simply a matter of deciding when to care.

People have started caring. The technical reasons for this are simple: cellular networks are ubiquitous; lithium-ion batteries are about as good a compromise (energy density, cost per watt) as compact electrochemistry is going to achieve for a while; and developers of all sorts are optimizing for power consumption at kernel level and above. Networks are smarter, partitions less catastrophic. Processors and storage keep getting cheaper, which both lowers fabrication expense and also permits the redundancy necessary for truly persistent interconnected systems. Lightweight application protocols are maturing—even HTTP/2 is cutting human-readable flab—connectivity protocols are handling fragmentation better, and patterns for robust client-server state sharing have softened the complexities of concurrency in distributed systems of relatively weak machines.

But the technical challenges of IoT aren't going away. Wireless power is a serious issue, but so is wired power, as smart grid security and distribution problems haven't been solved. Increasingly crowded wireless bandwidth and inevitably iffy device performance make latency even less negligible. Mesh, fog, and edge topologies complicate transport time prediction. Cheap IoT devices die fast. Who knows what the computational expense will turn out to be relative to the valuable intelligence gathered via storage, transport, and processing of dumb sensor data. And let's not forget security, the biggest IoT-related concern of all.

The *DZone Guide to the Internet of Things — 2015 Edition* addresses all of these issues and more. We're particularly excited to introduce a new research component, the first in our Guides: dozens of in-depth interviews with industry leaders, summarized as executive insights and offered exclusively here.

Explore the world of IoT and let us know what you think.

JOHN ESPOSITO
EDITOR-IN-CHIEF, DZONE RESEARCH
RESEARCH@DZONE.COM

Table of Contents

Credits

EDITORIAL

John Esposito
research@dzone.com
EDITOR-IN-CHIEF

G. Ryan Spain
DIRECTOR OF PUBLICATIONS

Jayashree Gopal
PROJECT MANAGER

John Walter
CONTENT CURATOR

Lauren Clapper
CONTENT CURATOR

Mitch Pronschinske
SR. RESEARCH ANALYST

Benjamin Ball
RESEARCH ANALYST

Matt Werner
MARKET RESEARCHER

Moe Long
MARKET RESEARCHER

BUSINESS

Rick Ross
CEO

Matt Schmidt
PRESIDENT & CTO

Kellet Atkinson
VICE PRESIDENT & PUBLISHER

Hernâni Cerqueira
LEAD SOFTWARE ENGINEER

Matt O'Brian
DIRECTOR OF BUSINESS DEVELOPMENT

Jane Foreman
DIRECTOR OF MARKETING

Alex Crafts
sales@dzone.com
DIRECTOR OF MAJOR ACCOUNTS

Jillian Poore
SALES ASSOCIATE

Jim Howard
SALES ASSOCIATE

Chris Brumfield
CUSTOMER SUCCESS ADVOCATE

Chelsea Bosworth
MARKETING ASSOCIATE

Chris Smith
PRODUCTION ADVISOR

Andre Powell
PRODUCTION ADVISOR

ART

Ashley Slate
DESIGN DIRECTOR

Yassee Mohebbi
GRAPHIC DESIGNER

Special thanks to our topic experts Matt Butcher, Andrew Trice, Henryk Konsek, Andreas Dharmawan, John Mueller, Travis Van, Rahul Awasthy, and our trusted DZone Most Valuable Bloggers for all their help and feedback in making this report a great success.

WANT YOUR SOLUTION TO BE FEATURED IN COMING GUIDES?
Please contact research@dzone.com for submission information.

LIKE TO CONTRIBUTE CONTENT TO COMING GUIDES?
Please contact research@dzone.com for consideration.

INTERESTED IN BECOMING A DZONE RESEARCH PARTNER?
Please contact sales@dzone.com for information.

Executive Summary

DZone surveyed more than 500 IT professionals for our Guide to the Internet of Things – 2015 Edition *to discover how organizations build IoT applications and develop strategies for implementing ubiquitous, secured systems. In this summary, you will learn how the majority of organizations are anticipating the changing landscape of IoT, and where their energies are focused.*

RESEARCH TAKEAWAYS

01. SECURITY IS THE BIGGEST CONCERN FOR DEVELOPERS AND COMPANIES ALIKE

Data: 79% of respondents stated security is a major concern for developing for the Internet of Things.

Implications: IoT security has continued to be on the minds of both companies and developers. A Gartner report predicted that in 2015, an estimated 4.9 billion connected devices would be in use [1]. By the end of 2017, over 20% of organizations will have security services devoted to their IoT implementations. Security will be most effective when implemented during both hardware and software development. When developing system architectures, security must be at the forefront of all decision making.

Recommendations: Minimizing data storage and transfer, and maximizing authentication and hardware efficiency are proactive steps in creating secure IoT environments. There is no one correct way to implement a foolproof IoT security infrastructure, but there are several layers of defense that will minimize risk. Refer to the IoT Security Checklist in this guide to see best practices for creating a secure IoT from the device, gateway, and server sides, as well as Henryk Konsek's "IoT Gateways and Architecture" to understand the most effective way of securely building systems.

02. HOME AUTOMATION IS WHERE DEVELOPER INTEREST LIES

Data: Developers are most interested in developing for consumer-focused, home automation platforms. 23% of respondents claimed they had worked on smart home devices, more than any other domain within IoT. 68% of respondents have expressed the desire to work with smart home devices.

Implications: Developers are focused on early adopters in IoT, so they're heavily engaged in developing for home automation devices, which is where a lot of interest lies for consumers. Most projections show, however, that the Industrial Internet of Things (IIoT) is where the future of the industry lies. According to Cisco, the IIoT has the potential to add $19 trillion to the global GDP by 2020 [2]. While home automation platforms will give developers experience with working in small-scale IoT environments, the future of the Internet of Things lies within the enormous infrastructure of the Industrial Internet of Things.

Recommendations: The vast majority of revenue for the entire Internet of Things industry will be within IIoT. While developer interest currently resides in the consumer side of IoT, developers within corporations need to recognize the importance of and prepare for the shift towards industrial applications of these technologies. Refer to Tom Smith's "Executive Insights of the IoT" in this guide to get the perspective of major decision makers and trendsetters within the IoT space.

03. THERE IS NOT ENOUGH AWARENESS OF IOT TRENDS

Data: While 53% of respondents said they were aware of Bluetooth LE (Low Energy), other widely used protocols, like ZigBee (34%), were largely unknown. Almost one quarter (23%) of respondents *said they had not heard of any of 9 protocols we referenced.*

Implications: The vast number of protocols and technologies within the Internet of Things is overwhelming for developers. It's difficult to stay abreast of current technologies when new protocols are being developed and implemented so rapidly in the IoT landscape. Understanding the pros and cons of each protocol is important for creating the most efficient IoT systems. Devices of all types will need to be able to communicate with each other quickly and efficiently, and utilizing the correct protocol is paramount to the success of these systems.

Recommendations: Decision makers will need to stay up-to-date on the most current technologies, and understand which protocols will best maximize their IoT infrastructures. Refer to Matt Butcher's article, "JSON, HTTP, and the Future of IoT Protocols" in this guide for insight on the future of protocols and the best practices for developing small- and large-scale systems that are able to efficiently communicate with each other.

04. IOT WILL BE RELEVANT FOR ALL ORGANIZATIONS

Data: 58% of respondents stated the Internet of Things is already relevant to their organizations, and 87% responded that IoT will be relevant in the future. With the amount of potential revenue available for organizations, it's clear that the Internet of Things will be an important factor in business decisions for most companies going forward.

Implications: Developers will have to adapt current development practices for this new paradigm: more so than ever before, hardware and software will need to be developed in tandem to create secure, efficient systems. Because advanced IoT systems are typically divided into three main categories—sensors, gateways, and the cloud—development for any system will involve massive undertakings. Teams will have to be able to work separately on these three domains, then integrate them within connected systems.

Recommendations: Understand your organization's vision for the Internet of Things, and document best practices to allow your development team to be at its most effective. Refer to Andreas Dharmawan's "Strategies for IoT Software Development and Delivery" in this guide for an in-depth look at utilizing existing development practices to create an efficient, secure, and powerful IoT system.

[1] gartner.com/newsroom/id/2905717
[2] forumblog.org/2014/01/are-you-ready-for-the-internet-of-everything/

Key Research Findings

More than 500 IT professionals responded to DZone's survey for the *Guide to the Internet of Things – 2015 Edition*. From these responses we've been able to draw several conclusions about how developers perceive IoT, which IoT applications pique their interests, how IoT affects and will affect their careers, what worries them most about IoT, and how much IoT development has matured.

Respondents identify mostly as developers (41%) or development team leads (24%), and are employed at organizations of all sizes (no dominant size range). The majority of respondents are located in the US (43%) or Europe (32%).

HOBBYISTS INTERESTED; COMPANIES EMBRACING IOT

Buzzwords both promote and obscure by magnifying and cloaking reality with hype. IoT hype is particularly difficult to disentangle from reality, both because the conceptual space is huge and also because IoT devices and platforms have become relatively cheap and easy to play with. These two factors often leave developers with an impression that, at the moment, IoT seems better suited either to hobbyists or to a relatively small number of large industrial or government enterprises.

With the abundance of consumer IoT products and prototyping boards on the market, hobbyists have certainly embraced development for the Internet of Things. 66% of respondents stated they are interested in developing for IoT as a hobbyist. Many of these hobbyists are developers within organizations where there is potential to work on IoT-related projects—44% of respondents said they are interested in creating IoT products for their company or team. 40% of respondents are also interested in building a startup for either consumer or industrial IoT.

DEVELOPERS BEGINNING TO TRANSITION FROM CURIOSITY TO ACTION

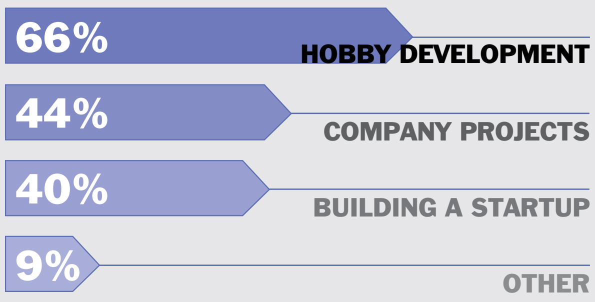
As developer awareness of IoT increasingly influences professional interest, so also has direct involvement in IoT applications grown over the past year. When we asked which use cases developers have worked on or would be interested in working on, responses of "would like to but have not worked on X use case" decreased 4% on average from last year, while responses of "have actually worked on X technology" increased 2%. Actual work on wearables increased most out of all use cases (4.1%). Presumably this reflects the increasing size and mainstreaming of the wearables market, thanks partly to the Apple Watch.

Note also the absence of 1:1 proportion between the decrease in "would like to but have not" and the increase "actually have worked on". Insofar as a proportional increase in the third response ("not interested") accounts for the difference between the first two, it seems that developers are growing slightly more confident in their ability to judge the value (positively or negatively) of entering a particular IoT problem space.

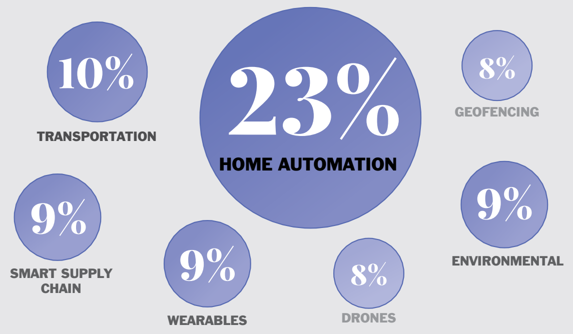
INTEREST IN TYPES OF IOT APPLICATIONS: WEARABLES DOWN, ENVIRONMENT UP

IoT in the abstract is simple: sensors and actuators communicating via IP. But each type of Thing is very different, and the ways each works are more varied still.

01. **WHERE ARE YOU INTERESTED IN USING IOT?**



02. **WHAT TYPES OF IOT PRODUCTS HAVE YOU WORKED ON?**



Just as developers drive technology adoption elsewhere, we suspect that developer interest in particular types of IoT applications will drive the Internet of Things as well. To learn which Things are most attracting developers, we asked survey respondents to rank which IoT applications interest them most. Two changes from last year were most striking. First, interest in wearables is down: last year wearable applications were ranked third most interesting, while this year wearables were ranked fourth. Second, interest in environmental IoT applications is significantly up, from sixth place in 2014 to third in 2015. This change reflects the growing maturity—and revenue potential—of the IoT technical space, as consumer IoT devices account for more of the inflationary buzz while industrial and governmental IoT (particularly infrastructure and environmental management) quietly continue to gain connectivity over IP.

NEVERTHELESS, IOT REMAINS IN THE FUTURE

Although developers are increasingly interested in mature IoT applications and actually developing IoT applications more frequently than last year, for most developers IoT remains in the future. This future-orientation is strong: 87% of respondents believe IoT will be relevant to their organization in the future, versus 58% who believe IoT is relevant to their organization right now.

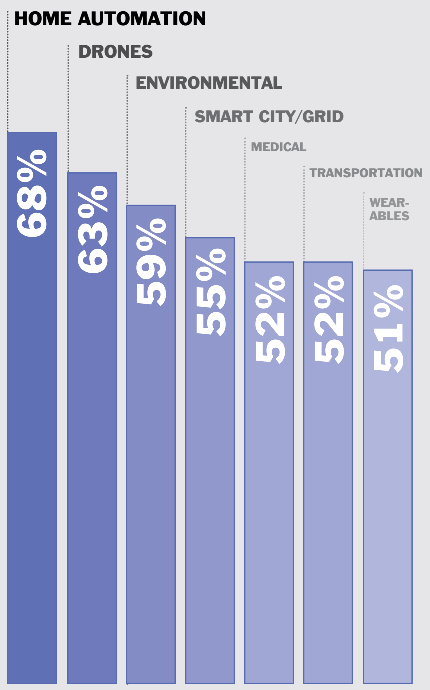
This difference between perceived present and projected future relevance has not changed from last year, despite the year-over-year shift in maturity of interest in and actual development of IoT applications. Perhaps this indicates a discrepancy between developers' thoughts about IoT applications on the one hand, and IoT as an industry-level concept on the other: while IoT applications are gaining traction, the general notion of IoT, conceived in relation to an organization rather than a particular use-case, remains hazy.

WORRIES ABOUT IOT ARE INCREASING, PARTICULARLY IN HIGHLY PRAGMATIC AREAS

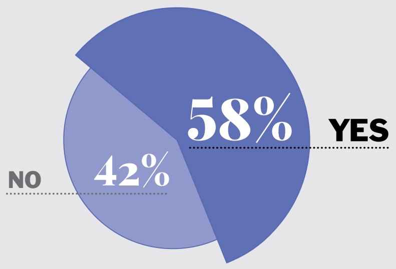
Developers agree with security professionals and industry leaders that security is the most concerning aspect of IoT. 79% of our survey respondents are worried about IoT security, which is up slightly from 77% last year and 11% more than the next greatest concern, privacy. The concept of IoT and the nature of IoT devices are sufficient to account for this level of concern, namely: the magnitude of the stakes, the probability of attack, the comparative peripherality of security to the core activities of application development, and the technical difficulties of maintaining security on low-power, low-reliability, in-the-field, ad-hoc connected, ubiquitous, wireless devices.

Two year-over-year increases in developers' areas of concern corroborate the claims regarding increasing seriousness and maturity of IoT development suggested above. The first is a 6% increase in developers' concerns about hardware and software maintenance. The second is a 5% increase in developers' concerns about connectivity and device management. Both are highly pragmatic concerns and are similar to issues that developers—especially but not exclusively mobile developers—already encounter professionally. As developers worry more in the context of IoT applications about the sort of problems that make other, more currently mainstream software development difficult, the gap between IoT and non-IoT application developer mindspaces shrinks.

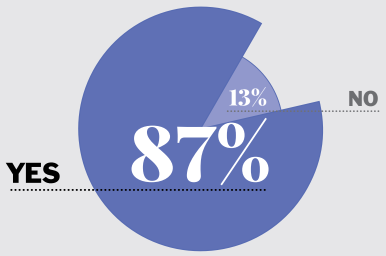
03. **WHAT TYPES OF IOT PRODUCTS ARE YOU INTERESTED IN WORKING ON?**



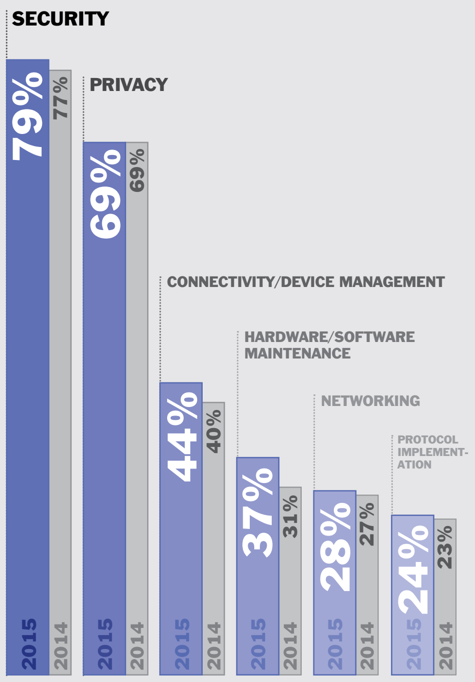
04. **IS IOT CURRENTLY RELEVANT TO YOUR ORGANIZATION?**



05. **WILL IOT BE RELEVANT TO YOUR ORGANIZATION IN THE FUTURE?**



06. **WHAT ARE YOUR CONCERNS REGARDING IOT?**



IoT Gateways and Architecture

BY HENRYK KONSEK

QUICK VIEW

01
Gateways can act as a single access point for monitoring operational fields.

02
IoT gateways allow updates over-the-air, which is particularly important for the delivery of critical security fixes.

03
The right gateway software will highly impact the total maintenance cost of your system.

The typical architecture of IoT solutions is usually far more complex than the architecture of most enterprise systems. One of the main factors that increases the complexity of IoT systems is that backend services residing in the data center, which is the heart of most enterprise systems, are actually just a piece of the bigger IoT picture. With IoT solutions, we have to deal with a myriad of devices working in the field. Because the nature of these devices is very different from web, desktop, or even mobile clients, we need an intermediate architectural element that will act as a proxy between the world of field devices and the enterprise data center. What we need is an IoT gateway.

WHY YOU NEED AN IOT GATEWAY

You may be wondering now: what exactly are the key reasons behind introducing a gateway into your IoT architecture? Let me shed some light on this issue by discussing some of the most important aspects of how gateway architecture functions.

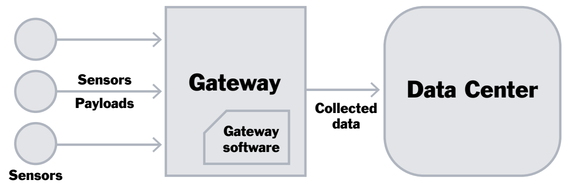
First, sensors usually have very limited capabilities in terms of networking connectivity. Your sensors can likely utilize Bluetooth Low Energy (BLE), just as the majority of beacons available on the market can; there is also the possibility that some of your sensors offer connectivity using the ZigBee protocol. There are also a bunch of other protocols that can be found in the Local Area Network (LAN), Home Area Network (HAN), or Personal Area Network (PAN). All of these protocols have one thing in common—they can't directly connect to larger networks like Wide Area Network (WAN) or the Internet. You need a gateway that can provide your sensors with a single point of contact with external networks by using WiFi, GSM, or some other type of connectivity.

Keep in mind that a gateway is not just a dump proxy that forwards data from sensors to backend services. Sending all the information collected by sensors to a data center would be highly ineffective in terms of performance and network utilization. An IoT gateway is needed to perform the pre-processing of information in the field, before they're sent to the data center. Such pre-processing includes message filtering and aggregation.

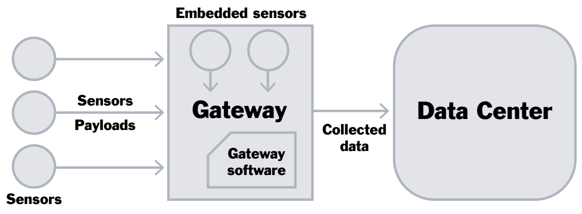
The gateway should also act as a single point of access for monitoring the selected area of the operational field. You don't want to connect to every sensor with your monitoring software; it is easier to monitor only the gateway, which in turn is responsible for gathering all the necessary metrics from the sensors.

ARCHITECTURAL OVERVIEW

The following gateway architecture diagram is the most common architectural design where the gateway itself is not equipped with sensors. The gateway software installed on the device is responsible for collecting data from the sensor, pre-processing that data, and sending the results to the data center.



Keep in mind that it is possible to have variations on this sensor architecture where some of the sensors are located at the gateway device, as illustrated in the following diagram.



Embedded sensors that might be present at the gateway could include options like a GPS unit or a temperature sensor connected to the gateway using the GPIO interface.

THE GATEWAY SOFTWARE

The software application is the heart of the gateway. The gateway software is responsible for collecting messages from the sensors and storing them appropriately until they can be pre-processed and sent to the data center. The gateway software decides if the data at a given stage of processing should be temporary, persistent, or kept in-memory.

The gateway software should be designed with failure and disaster recovery in mind. Since gateway devices are often operated in the field, you should prepare for working conditions that are far from ideal. For example, the gateway software should be prepared for a power outage or other actions that may result in an interruption of gateway processing. The gateway software should be bootstrapped and started automatically as soon as power returns to the device, and it should continue to work from the point where it was interrupted.

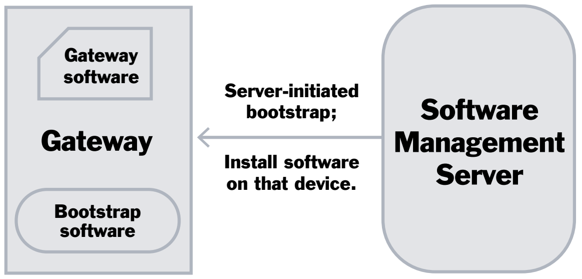
Gateway software should also be smart enough to properly handle system logging. It has to find the right balance between the number of log entries stored on the device and those sent to the data center.

SOFTWARE INSTALLATION AND UPDATES

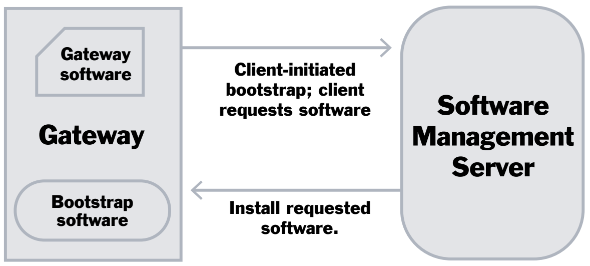
How does the gateway software get into the device? There are three main approaches for this issue.

The first approach is pre-installing the software on the gateway disk (or memory card). This approach is called *factory bootstrap*. As you may guess, this technique doesn't scale well if your solution includes a larger number of the gateways.

The second approach is the *server-initiated bootstrap*. In this mode, the central software management server communicates with the gateway device and deploys the proper version of the software to it. This approach scales much better than the factory bootstrap, but still requires the initiation of deployment action on the server side.


Gateway software
Gateway
Bootstrap software
Server-initiated bootstrap;
Install software on that device.
Software Management Server

The third approach is the *client-initiated bootstrap*. This mode assumes that it is the gateway's responsibility to connect to the central repository server and download the proper version of the software. In this scenario, the gateway is required to have lightweight bootstrap software installed so it can communicate with the software management server. This approach is the most scalable one, as it doesn't require any centralized coordination of the deployment action. Every gateway device downloads the software as soon as it is powered on.


Gateway software
Gateway
Bootstrap software
Client-initiated bootstrap; client requests software
Install requested software.
Software Management Server

One extremely important feature of IoT gateways is the ability to download updates over-the-air. Keep in mind that after you install the gateway software onto a device and deliver it into the field, you have very limited capabilities in terms of the gateway software maintenance. The ability to download software updates over-the-air is particularly important from a security perspective, as it can affect the delivery time of critical security fixes.

SENSOR CONSUMERS

If the software application is the heart of the gateway, then the sensors are the eyes and ears of the gateway. Sensors are small hardware devices that can measure some aspects of the real world. Common types of data collected by the sensors are temperature, GPS coordinates, humidity, air pressure, and so on.

The messages collected by the gateway from the sensors are usually small in size. For example, the current value of the temperature measured by the sensor is just a decimal number. GPS coordinates are two decimal numbers, which represent longitude and latitude. This is an important thing to remember: the gateway operates on a large number of small messages.

While the sensors themselves can generate messages frequently, it is important to anticipate how many messages we really need to gather from the sensors. For example, we can read the temperature from a sensor every millisecond, but do we really need this kind of precision when measuring temperature changes? In the majority of cases, reading the sensor value a few times per second is more than enough, as we are more interested in the metric changing over a longer period of time. Gateway software usually polls the sensor data periodically. Good gateway software allows you to easily configure the polling interval for every sensor. You definitely don't want to put unnecessary sensor data into the gateway, as obsolete messages consume the precious processing power of your constrained gateway device.

GATEWAY DATA TRANSFER

Usually gateways are connected to the Internet using GPS, WiFi, or ethernet. Some gateways can also work in both GPS and WiFi modes (for example, gateways mounted in moving vehicles). In general, non-GPS connectivity is preferred to send data, as it doesn't require a subscription to a paid mobile plan. Some gateways will be constantly connected to inexpensive local networks, but those using GPS connectivity should be very conservative in terms of what data they send to the data center. The gateway should apply business logic against the data it collects to understand which messages should be sent over expensive GPS networks, and which data can be cached on the device for deferred offline processing.

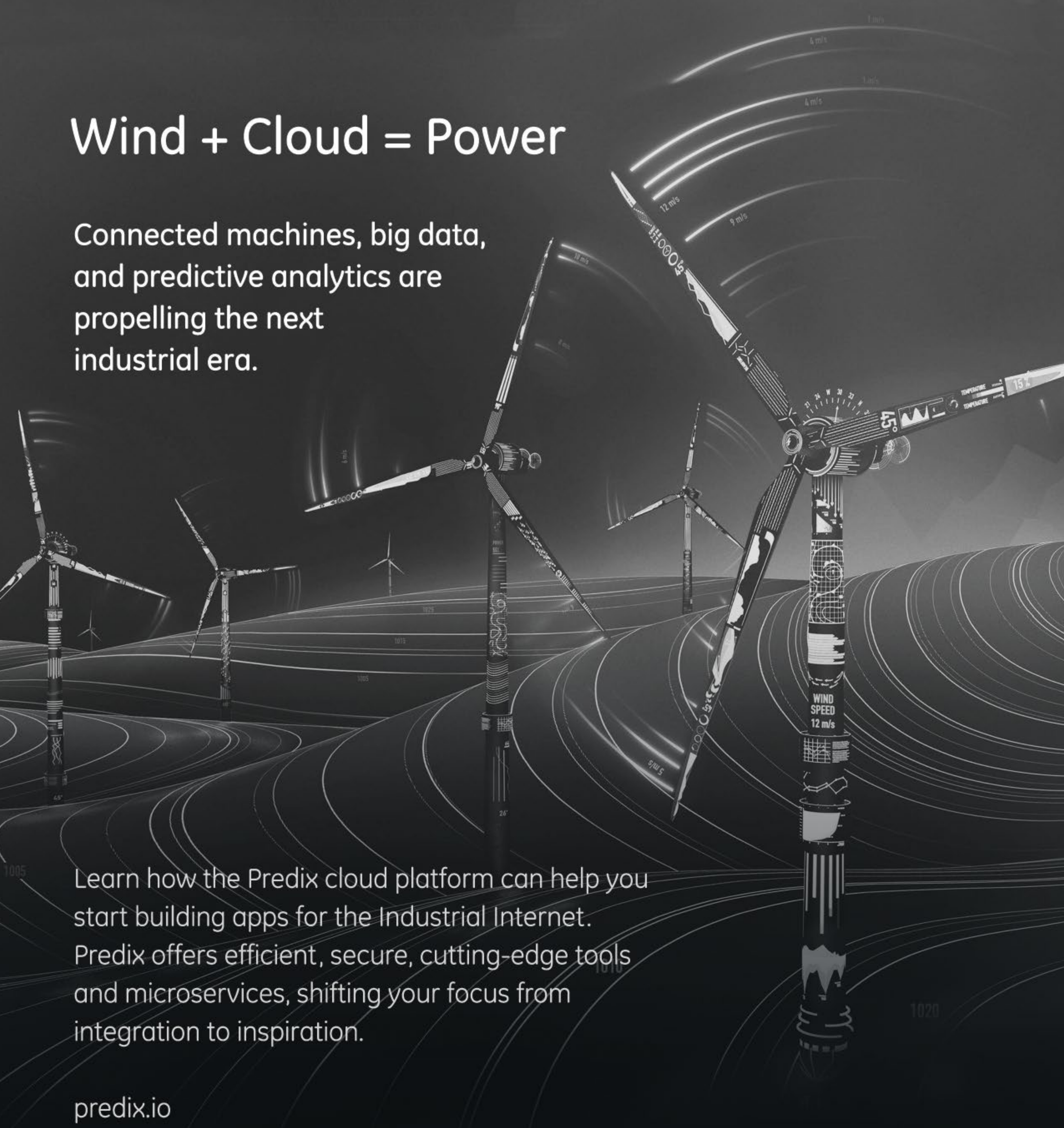
SUMMARY

The gateway is a key component of every IoT solution. Before you decide what kind of hardware you would like to purchase as your gateway platform, spend some time analyzing your message flow and the data formats of the payloads, and try to filter out or aggregate as much data as you can before sending it from the gateway to the data center. Also, while the choice of proper hardware for your IoT solution is very important, you have to keep in mind that picking up the right gateway software and management infrastructure (like the LWM2M server for managing your devices) is a factor that will highly impact the total maintenance cost of your system.

HENRYK KONSEK is an Engineer at Red Hat, where he's worked on Apache Camel, JBoss Fabric8, and JBoss Hawt.io. He is keen on the marriage of the Internet of Things, messaging, and the cloud.

Wind + Cloud = Power

Connected machines, big data, and predictive analytics are propelling the next industrial era.



Learn how the Predix cloud platform can help you start building apps for the Industrial Internet. Predix offers efficient, secure, cutting-edge tools and microservices, shifting your focus from integration to inspiration.

predix.io

Developers Wanted for the Industrial Internet

No doubt about it, the Internet of Things is pretty cool—especially for software developers. Write a few lines of code and your refrigerator can remind you to buy milk.

Interesting, but not life changing.

At GE Software, when we talk about connected machines, we're thinking bigger. A lot bigger.

You may have heard of the Industrial Internet—or what I like to call the Internet of Really Important Things. GE was early to recognize the convergence of heavy industry or "big iron" with big data and predictive analytics. The result? Industrial companies can now achieve unprecedented levels of efficiency, safety, sustainability, productivity, and profitability.

We created our GE Predix platform to help developers simplify the delivery and consumption of Industrial Internet applications by using the automation and elasticity of cloud computing for faster time to market, improved agility, and reduced operating and capital expenses. You can optimize the performance of any business across plants and fleets, down to the individual asset.

With Predix, a hospital's CT scanner can communicate with infusion pumps for better real-time patient care. A wind turbine can check itself into maintenance and chat with other machines on the farm (and the grid itself) to optimize output. Connected locomotives and trains can self-optimize their route plans across train networks to improve passenger service.

But we can't advance the Industrial Internet alone. We need the global developer community to help us spark a new industrial era that could add $10 trillion to the global GDP.

We need the global developer community to help us spark a new industrial era that could add $10 trillion to the global GDP.

The Industrial Internet is dramatically different than the consumer-based Internet of Things. It's bigger. It's tougher. It requires a common operating system, such as Predix. And it demands that the brightest minds come together and solve the world's biggest challenges.

What will you build with Predix?

WRITTEN BY HAREL KODESH
VICE PRESIDENT AND CHIEF TECHNOLOGY OFFICER, **GE SOFTWARE**

Predix

Predix powers industrial-strength apps by bringing together device connectivity, data integration, analytics, cloud, and mobility.

TARGET MARKET	IDE PROVIDED	OPEN SOURCE	REAL TIME MESSAGING	PROPRIETARY HARDWARE REQUIRED
B2B	✗	✗	✓	✗

CASE STUDY

From space, Norfolk Southern's 20,000-mile rail system resembles a neural network. And it increasingly works like one, thanks to GE's Movement Planner, an Industrial Internet app that helps guide hundreds of trains traveling the railroad's network across 22 states daily. Powered by GE Predix, Movement Planner uses big data to improve machine and infrastructure efficiency and help the environment. The app takes logistical information and combines it with schedules, track grades, train movement, and other data. As a result, trains run faster and more efficiently on existing routes without laying new tracks. Since the company turned on Movement Planner a few years ago, fuel use is down 6.3%, and velocity is up 10% to 20%.

PRODUCT HANDLES

- M2M Gateway
- Data Services
- Networking Hardware
- Web Services
- Analytics

NOTABLE CUSTOMERS

- Pitney Bowes
- BP
- Dubai Aluminum
- City of San Diego
- Norfolk Southern
- E.ON

BLOG gesoftware.com/blog **TWITTER** @GEsoftware **WEBSITE** predix.io

Powering Wearables for the Enterprise

QUICK VIEW

01
An app's business logic and server interaction can be shared code assets so that both the services and workflow code can be reused across both projects

02
Use the same mobile platform infrastructure to deliver secure data to wearable apps.

03
Building apps for wearable platforms is largely the same as building for mobile.

BY ANDREW TRICE

Wearables and smartwatches have been around for a while now. Simple fitness trackers really started the movement, but things have evolved quite a bit since their introduction. The Pebble smartwatch kicked things off back in 2013, quickly followed by Samsung and Motorola who produced similar entries in the market. Not long after that, Android Wear emerged and created its own product ecosystem. Most recently Apple has set the wearables space on fire with the Apple Watch. In fact, Apple recently announced it has sold over one billion dollars worth of devices, capturing 75% of the smartwatch market in a single quarter. The market for apps for this new class of devices is rapidly evolving as well.

While apps like fitness trackers and "remotes" (meaning a remote controller for either an app on your phone or a remote control for a physical object) have been popular so far, that is not where things will end. In fact, wearables and smartwatches are working their way into the enterprise, much like smart phones have already done. The phone allows for a more personal experience than the desktop computer, because it's a device that you almost always have with you. It has enabled access to data that is important to you, with respect to your personal context. You can expect this to continue and mature in the context of wearables. Apps developed for wearables and smartwatches will be even more personal than ever. Not only will they deliver the information that you need, when you need it, but they will go even further with an emphasis on simple, easily consumable, or easily actionable interfaces. Context is the critical difference. A smartwatch is just a glance away, and can notify you when there is news you should see, without having to pull something out of your pocket. Information can be delivered based upon your location, history, or any other kind of preference—all right to your wrist. It's not just about the consumption of information; smartwatches also enable simple, personal collection of information. Whether it's your location, a simple tappable form on your wrist, or a remote for a more complex app or peripheral device, for both consumption or creation of data, it is a more personal context than ever.

So, is your enterprise ready for the incoming wave of smartwatches and wearable devices? Do you have a strategy to integrate these kinds of devices into your systems or workflows?

Let's focus on the smartwatch category. Luckily for most of us, everything that we need to integrate smartwatches into our ecosystems already exists. If you have a successful mobile application, then chances are very high that your smartwatch app can reuse this infrastructure and existing code. You don't necessarily want the exact same user experience. It is highly likely that your user interface and user interactions will be extremely different due to the lack of a keyboard and significantly less screen real estate, but backend services and workflow logic can probably be reused.

A smartwatch app is, in essence, functionally the same as a mobile app on a phone. Many Android-based smartwatches are more or less small phones strapped to your wrist. They contain the processor, memory, and networking capabilities of a phone, and they act as independent devices. When you write apps for them, you will have to use the smartwatch platform SDK, but you can reuse much of the logic and services that power your mobile app. The Apple Watch, in its current release, is a peripheral device. The Apple Watch contains limited processing ability; rather, it delegates all processing logic and networking requests to an extension (separate application binary) that runs on your phone. This will change slightly with watchOS 2, which enables on-device program execution, but the way that you write your apps remains largely the same as for an iPhone. In both of these models, the UI is built using Xcode's Interface Builder, and application logic is written in either Swift or Objective C.

This gives you the ability to access backend systems in exactly the same way that your mobile apps do. The app's business logic and server interaction can be shared code assets so that both the server-side code/services and the client-side workflow code can be reused across both projects.

Apps developed for wearables and smartwatches will be even more personal than ever

At a high level, the most common architecture for enterprise apps will be that the app (on the phone or on the watch) makes HTTP requests to the server. These requests either push data to the backend or retrieve data from the backend system. The server then processes the request, saves or retrieves data, and returns whatever data is needed to the mobile client. The services could be RESTful, SOAP-based (though I don't recommend SOAP because of its verbosity), or just plain old HTTP requests. It really doesn't matter. Likewise, the application server tier doesn't really matter much either, because we are operating on standard web protocols. Your existing infrastructure based on an enterprise Java platform, built on top of Node.js, built on top of .NET, PHP, or really anything else, is still sufficient to deliver information to and from your mobile and wearable apps; I'm also assuming that your backend is built in a service-oriented fashion that enables this architecture.

Let me rephrase this in terms that are related to my role as a Developer Advocate with IBM: you have lots of options for building mobile applications with IBM products. There is the MobileFirst Platform, which is a comprehensive suite of enterprise-class tools for efficiently building and maintaining mobile apps; there are MobileFirst MBaaS services on IBM Bluemix (cloud services); and there are also individual Platform-as-a-Service (PaaS) offerings on Bluemix, such as the Node.js infrastructure, Java/WebSphere Infrastructure, the Cloudant NoSQL database, and more. There are also the Watson cognitive computing services that can be used with any of your mobile applications.

If you have a successful mobile application, then chances are very high that your smartwatch app can reuse this infrastructure and existing code

If you already have an application that, for example, uses the MobileFirst platform to deliver secure data and active monitoring on a mobile app, then that existing infrastructure and investment can be reused easily within a wearable or watch app with almost no code changes (besides the user interface). This is already being done today: major institutions are leveraging the services that they created for their mobile experiences to power their most personal interactions at a glance on smartwatches.

If you have already built an app on top of Node.js that uses IBM Watson to perform complex cognitive analysis and return it to your iPhone and Android apps, then the exact same code and processes can be leveraged within your wearable application. Imagine that you have an app that leverages IBM Watson language services to translate from English to another language in real time as you interact with data. Now imagine that you want to move this experience to the wrist to enable more efficient and contextual interactions in everyday life. You can reuse the exact same logic and operations as the mobile client, and only need to create a user interface for the new form factor.

Delivering apps on the Apple Watch, Android Wear, or other wearable platforms really is not anything new in a technical sense. You have a new user interface, a new form factor, and a new notification and interaction mechanism, but the logic flow and order of operations are largely the same. If you have built your existing mobile experiences in a modular fashion that leverages standard protocols, then there is a good chance that most of your work to build the wearable experience is already complete.

ANDREW TRICE is a Developer Advocate with IBM, bringing to the table more than 15 years of experience designing, implementing, and delivering rich applications for the web, desktop, and mobile devices.

One Platform to Design, Build & Analyze your Internet of Things

CloudOne

Where IoT is Made™

OnCloudOne.com

Building *Your* Internet of Things Platform

The era of hyper-connectivity called the "Internet of Things"—where smart, connected products, product systems, and other things connect to new business applications and create new sources of competitive advantage through software, the cloud, and service—is a fundamental transformation of how companies make products and exchange value with their customers. Savvy companies need to get to the future of IoT first—ahead of their competitors. Now is your company's time to become the victor rather than the victim of IoT. Forward thinking makers of Internet of Things companies are proactively using this disruption to leap to the front, so that they're not only ready for what's next, but they are way out in front of their competition.

To get to the front, companies first need a vision for the future. What really goes into making things for the Internet of Things? We at CloudOne believe there are 5 key areas that interact with each other to make an opportunity for companies to make things in the Internet of Things.

- How you DESIGN and TEST things.
- Places where you BUILD things.
- Systems that FEED and READ things.
- MESSAGES from the things.
- DATA and ANALYTICS on everything.

Accelerating Your Internet of Things
Unlike other IoT platforms that require you to abandon everything you have already built and forgo the technology and software you already own to start from scratch using only their technology stack, CloudOne takes the software that you are already using and pre-integrates your software, your data and your configurations into your own platform for IoT. Your company's custom platform is engineered to connect you to the next generation of services and software so that you can develop, test, manage, share, and analyze like never before to make smarter products with competitive edge.

Now is your company's time to become the victor rather than the victim of IoT.

Designed entirely to accelerate development and deployment with software-as-a-service, collaboration, and Virtual Private Clouds at its core, CloudOne is the one platform that connects Continuous Engineering, DevOps, Enterprise Asset Management, Big Data, and Analytics and Global Device Messaging into one platform, solving the puzzle that is IoT.

To learn how today's leading enterprise organizations are using CloudOne's IoT platform and solution services to leap ahead of the competition, visit www.OnCloudOne.com

WRITTEN BY
JOHN MCDONALD, CEO, CLOUDONE
NICOLE DELMASTRO, DIRECTOR OF MARKETING, CLOUDONE

CloudOne IoT Platform

CloudOne

The CloudOne IoT Platform brings together a wide range of technologies, hardware, sensors, devices, apps, telematics, data, and connectivity to the cloud delivered back to you as a managed service.

TARGET MARKET	IDE PROVIDED	OPEN SOURCE	REAL TIME MESSAGING	PROPRIETARY HARDWARE REQUIRED
B2B	✓	✗	✓	✗

CASE STUDY
CloudOne operates the world's largest software-based telemetry analytics solution for Cummins Engine, into which tens of thousands of data packets from engines all over the world are analyzed live for trends and issues, with results fed back to development and to production factories for action. We also operate the global development environment for Panasonic Automotive, connecting over 1,600 developers in 4 continents to one single set of code used in all of their infotainment systems supplied to most of the world's car manufacturers—something we also do for the #2 producer of infotainment systems, Harmon Electronics, meaning we provide the development environment for 9 out of 10 infotainment systems produced globally.

PRODUCT HANDLES
- Device Management
- M2M Gateway
- Data Services
- Sensors/actuators/ controllers
- Web Services
- Analytics
- ESB

NOTABLE CUSTOMERS
- Cummins
- Panasonic Automotive
- Whirlpool
- Chamberlain
- McDonald's
- Shell
- Hilton
- Lenovo

BLOG oncloudone.com/resources/blog **TWITTER** @OnCloudOne **WEBSITE** oncloudone.com

Strategies for IoT Software Development and Delivery

BY ANDREAS DHARMAWAN

QUICK VIEW

01
An ideal IoT product would have three software teams working on embedded, Big Data, and mobile application spaces.

02
Software updates and the orchestration of tooling for IoT challenges the bandwidth of traditional Agile development.

03
An IoT platform is needed to handle all the different components of these three teams while also continuously delivering software.

Imagine the Following Scenario:
Your car recognizes you as you enter the car. It also knows that it's 6 p.m. on a Friday. After consulting with your smartphone calendar, the car knows that on Fridays you always go swimming at Hotel Nikko in downtown San Francisco. It then checks the real-time traffic information and automatically recommends the best route to the Hotel. When you are about to head home from swimming, your smart refrigerator notifies your phone to stop and pick up some milk because you've run out, so your car routes you to your favorite local store.

The next day, you're driving to Los Angeles when the car notifies you that the fuel pump is about to go bad. It recommends you to a dealership along your route that it specifies is open on Saturdays. Prior to making the recommendation, the car has already checked that the dealership is open, has the part, and that they can schedule your appointment. After a quick service, you're back on the road and heading for a relaxing weekend in LA.

This scenario is no longer science fiction, and the technology is already here.

ARCHITECTURAL OVERVIEW

To support these advancements, the car manufacturer has three software teams:

1. The first team focuses on developing the software that's embedded in the car: this software is responsible for the interacting with the driver and providing health data, phone connectivity, etc. This team works predominantly with Real-Time Operating Systems (RTOS) and works collaboratively with the mechatronics (mechanical and electronics) team. The most common programming language for this team is C/C++ and software upgrades are usually deployed Over-the-Air (OTA).

2. The second team focuses on Big Data applications: the software that aggregates and analyzes data in real time from millions of cars on the road and all third-party connected services. This software component is the one that receives the SOS signal from the car about the impending fuel pump failure, finds the dealer, and directs the car to the shop. This team is tasked with processing massive amounts of real-time data and is mostly concerned with horizontal scalability, which enables them to support data throughput as more devices are sold.

3. The third team focuses on building the mobile app: this application is used for seamless integration with the car's infotainment system. Mobile developers often prefer to use Software-as-a-Service (SaaS) tools for the creation of the app. The mobile apps are updated at a high frequency.

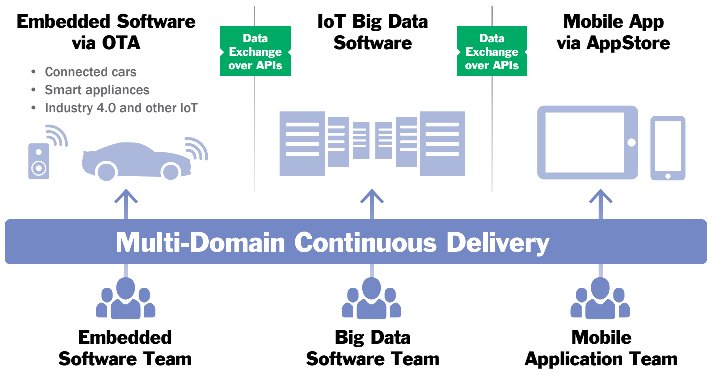
A multi-target IoT solution must enable zero downtime upgrades and automatic rollbacks for full-stack or partial IoT service updates.

WHAT DO YOU NEED IN ORDER TO MANAGE IOT SOFTWARE COMPLEXITY?

Coordinating the three software teams will be a challenge without the proper DevOps platform, because any software upgrade must be coordinated in such a way that it doesn't break the functionality between the different software components installed on different devices. A complex software design with such high stakes requires shared visibility, shared reporting, and an integrated dashboard for central management of the software delivery process. This allows project team leaders to see the progress of any change requests or updates on three different software tracks; it also helps ensure that each software release is going smoothly with no quality assurance issues or possible integration failures that could disrupt service.

The three software teams will need a single integrated DevOps platform that can handle three different deployment targets, each with its own specific deployment process, stack, and more. It needs to handle the following:

1. The embedded software in the car itself, where software upgrades are usually deployed OTA.
2. The data center for Big Data storage and computation, where software updates are done via the Internet.
3. The mobile app, which is upgraded via the AppStore.



The three software teams that are part of the development of an IoT product and services have distinct characteristics. They each use different technologies, stacks, deployment patterns, and delivery practices in their work. Their day-to-day tasks and workflows are different, and so friction exists by nature when the three teams must coordinate their integration and system tests. If the possible failure points are not minimized among the three different processes, then delays will inevitably happen; if delayed, the product's quality and market share can suffer.

In addition to common Agile development practices and Continuous Delivery/DevOps platform requirements, there are unique requirements from a tooling perspective to enable efficient and streamlined IoT application delivery. A single platform is needed that can address the three different domains as well as integrate and orchestrate the work transitions and handoffs between teams throughout the product's lifecycle. In addition, the platform must be able to track artifacts, the processing outcomes, and who works on different stages. Here are some additional requirements for a multi-target IoT solution that accelerates software delivery securely and reliably while also improving the quality of service:

- Handle different deployment paths (e.g. an embedded device via OTA updates, a data center via the internet, and a mobile app via the app store) from a single integrated solution.
- Enable teams to own the pieces of orchestration pertaining to their applications while also enforcing a separation of duties.
- Orchestrate the delivery pipelines for each team and manage the dependencies between these pipelines.
- Provide an artifact repository to store and trace the life of each artifact.
- Provide centralized dashboards and processes to facilitate the monitoring and management of delivery pipelines and releases.
- Enable zero downtime upgrades and automatic rollbacks for full-stack or partial IoT service updates.
- Provide complete traceability with automated compliance reports that are available on demand.

Through a single, integrated DevOps platform, the project team leaders can have a single dashboard to track team progress and variability management of artifacts from three project teams.

GETTING IOT RIGHT

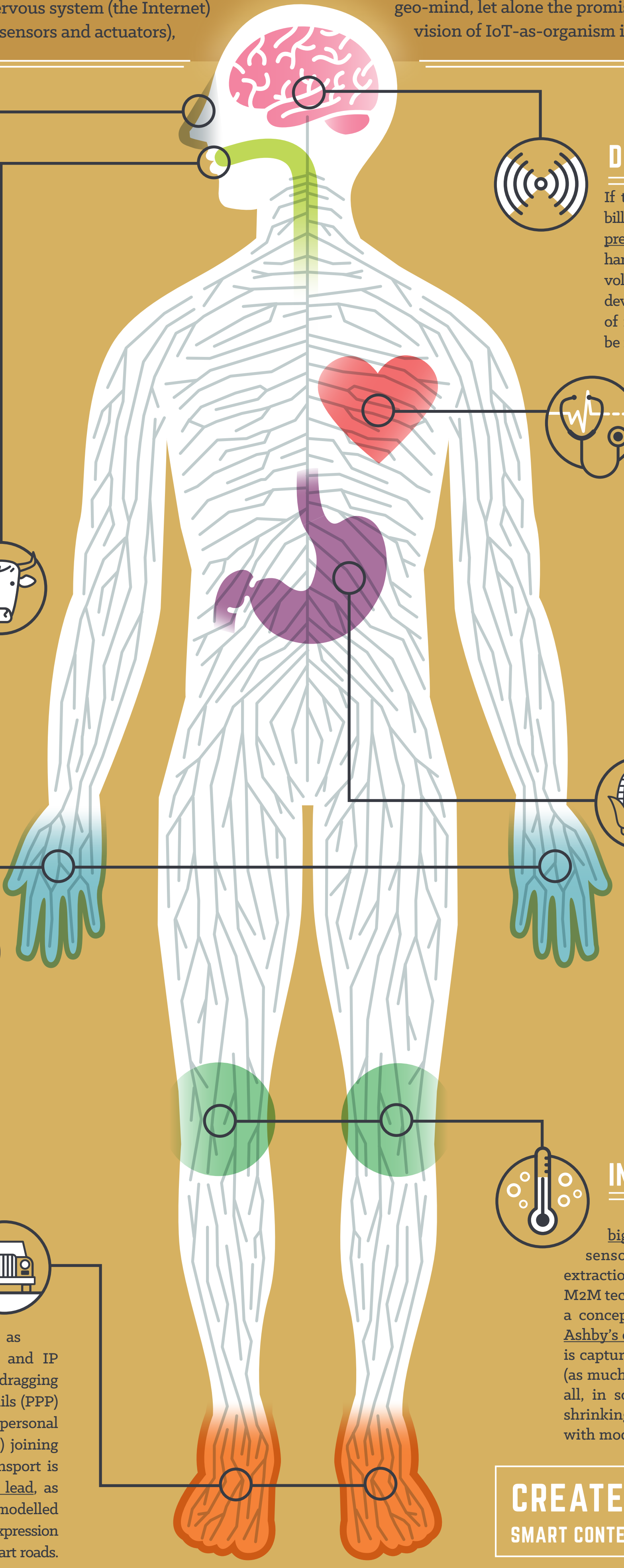
The Internet of Things brings rise to a plethora of new and useful services that enrich our lives, simplify it, or save us time and money. To provide these connected and complex services, software companies must have at least three different software teams in order to deliver different, integrated service components across different platforms and devices. In addition, software upgrades must be coordinated across all environments to ensure service continuity. Only an integrated DevOps platform can provide the traceability, visibility, shared control, and ability to react quickly for these complex software development, testing, and deployment processes.

ANDREAS DHARMAWAN is the Vice President of Technical Field Operation at Electric Cloud. Dharmawan works with reseller partners in Japan, Korea, China, and Germany to support overseas customers and expand sales in APAC and EMEA. Before working at Electric Cloud, Dharmawan worked for Audi Club of North America, Spirent Communications, and FanFare Group.

DZONE.COM PRESENTS

THE ANATOMY OF IoT

Nikola Tesla predicted that wireless electrical transmission would turn the earth into one giant brain. But the modern vision of IoT goes beyond this, transforming the world into a complete organism: the nervous system (the Internet) connects organs and limbs (sensors and actuators), each of which has its own system-internal or system-external function. But "the internetworking of physical inputs and outputs" doesn't fully capture Tesla's grand vision of the geo-mind, let alone the promise of IoT. Here's how the vision of IoT-as-organism is panning out today.



MACHINE OLFACTION

When IoT imitates living organisms directly, biological adaptations sometimes introduce strict constraints. For example, in order to view the world like a human nose, an olfactory sensor must respond to chemicals concentrated at 10-12 g/mL; respond very little to temperature and humidity; react, recover, and recalibrate quickly; and process a fair amount of data before output.

DATA CENTERS

If the Internet of Things will connect 26 billion devices within five years (as Gartner predicts), then data centers will be forced to handle absurdly "big" data (with regards to volume, velocity, and variety) from IoT devices. Security, bandwidth, and analysis of minimally processed information will be major concerns.

HEALTHCARE

Wearables are already supporting individual fitness, but IoT's potential for public health is practically incalculable, especially because the effects of two leading first-world killers (diabetes and heart disease) can be mitigated massively by real-time monitoring. Other applications are more focused: sensors can detect falls and alert paramedics, warn about excess UV radiation exposure, notice dangerous overcrowding, or alert public officials about low air quality.

ANIMAL FARMING

Most of what makes animals tick—and become optimally digestible for humans—happens under the skin, invisible to all but researchers. If only livestock farmers could gather vital health data in real time! Well, cheap livestock 'wearables' and analytics platforms are finally making this possible, adding to the growing network of sensors for toxic gas, herd movement, and separation between parents and offspring.

AGRICULTURE

"AgTech" + Internet = Agricultural IoT, which is immature but burgeoning. Networked sensors help farmers monitor soil moisture, control climates in greenhouses and silos, selectively irrigate, kill pests, or plan for weather. Dropping sensor prices are making adoption easier, but connectivity is still poor in rural areas, software for agricultural IoT is limited (though not nonexistent), and security around farming data is a new and unsolved problem.

ENVIRONMENT

"Smart Environment" comprises IoT's deepest conceptual cut, shattering the distinction between controlled system and inert surroundings. Wireless sensor networks monitor ambient temperature, moisture, barometric pressure, air composition, snow levels, logging activities, and earthquakes—they even detect forest fires.

INDUSTRIAL CONTROL

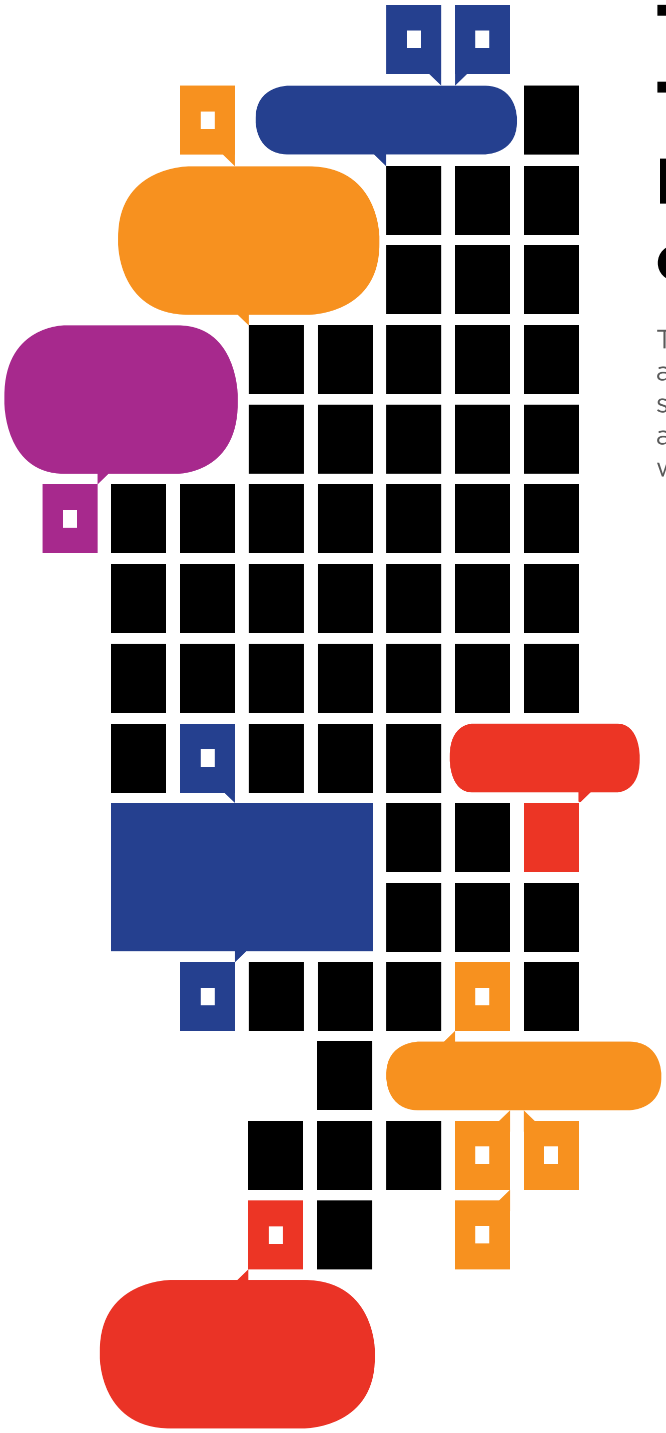
Industrial IoT—the area of IoT's biggest economic impact by far—uses sensor data to streamline resource extraction, manufacturing, and transport. M2M technology is extremely mature, having a conceptual lineage that dates as early as Ashby's cybernetics. Far more industrial data is captured than is used to maximum benefit (as much as 99% of sensor data is not used at all, in some cases), but this inefficiency is shrinking as industrial analytics catch up with modern data gathering capabilities.

TRANSPORTATION

Smart transportation is the real-world analogue of IoT conceived as the convergence of mobile (vehicles) and IP (roads). As steam engines (mainframes) dragging unpowered cars (dumb terminals) on rails (PPP) gave way to user-commanded cars (personal microcomputers) on local roads (LANs) joining freeways via on-ramps (gateways), transport is finally following infotech's distributed lead, as the self-servicing adaptivity of OSI-modelled plumbing is finally seeing its locomotive expression in sensor-riddled cars coordinating on smart roads.

CREATED BY DZONE.COM
SMART CONTENT FOR TECH PROFESSIONALS

ThingWorx
A PTC Business



ThingWorx. The only complete IoT platform for the connected world.

The ThingWorx IoT platform provides a complete application design, runtime, and intelligence environment so you can rapidly design and continuously iterate IoT applications. Reduce the time, cost, and risk associated with building innovative IoT applications.

Develop and Deploy 10x Faster
The ThingWorx application modeling environment makes it easy to model Things, business logic, visualization, data storage, collaboration, and security.

Create IoT Apps Rapidly
Leverage a complete set of UI widgets, extensive collaboration components, data visualization charts, grids, and forms without the need for coding.

Innovate with Search-Based Intelligence
ThingWorx SQUEAL™ brings search to connected devices and distributed data. Correlate collaboration data, line-of-business system records, and equipment data.

Choose Your Connectivity
Connect your devices via 3rd party device clouds, direct network connections, Open APIs, or ThingWorx AlwaysOn™ connectivity using the scalable, secure ThingWorx Edge Microserver.

Start Building Today
QuickStarts from ThingWorx allow you to immediately take advantage of the ThingWorx IoT platform. Getting started is as easy as drag and drop.

Explore ThingWorx IoT QuickStarts:
www.thingworx.com/go/developer

Hello everything.

Solutions Patterns for the Internet of Things

If the Internet of Things (IoT) really is a technology paradigm rather than an industry, then it is important to map *how* that paradigm provides solutions across different industries. Real-world IoT implementations tend to fall into three major solution patterns:

- Smart, Connected Products
- Smart, Connected Operations
- New, Disruptive Experiences

Smart Connected Products: Products live at the edge of the IoT and are the "Things" in the IoT paradigm. Products we have used for years—like dishwashers and photocopiers—are enhanced through connectivity. Since 2012, we have seen a major trend towards manufacturers designing connectivity into their products.

Transforming Existing Operations into Smart, Connected Operations: Sometimes the "Thing" is not a single product or device but rather an operation—like a factory or the management of a city infrastructure—that is instrumented with access to real-time data and control capabilities from the cloud.

New, Disruptive Experiences: Products and services are emerging that have jumped the development curve thanks to the confluence of cheap microprocessors, ubiquitous WiFi, fast cellular connections, and shrinking devices. For example, I wear a watch and carry a smartphone while running. Imagine a smartwatch that taps into my location and notifies my wife in real time if I have gotten lost during a run. Wearing a watch is now a completely new experience and has revolutionized a once iconic accessory.

These patterns give us a common mental framework to discuss the infinite design possibilities and permutations of the IoT. They drive us to understand the importance of using a technology platform like ThingWorx to handle the security, scale, application logic and UX, and business system integration challenges that are uniform across each of these three patterns. Only by leveraging a platform like ThingWorx can you rapidly and successfully bring your solution to life.

WRITTEN BY JOE BIRON
VP OF BUSINESS DEVELOPMENT FOR IOT TECHNOLOGY AT THINGWORX, **A PTC BUSINESS**

ThingWorx, a PTC Business

ThingWorx
A PTC Business

ThingWorx is the most complete IoT offering and technology stack, featuring flexible connectivity options, end-to-end app modeling, and analytics.

TARGET MARKET	IDE PROVIDED	OPEN SOURCE	REAL TIME MESSAGING	PROPRIETARY HARDWARE REQUIRED
B2B	✓	✗	✓	✗

CASE STUDY

OnFarm, a highly specialized integrator of agriculture field asset and information systems for the farming industry, utilized ThingWorx to overcome major barriers to increased agricultural use of sensor technology due to interoperability. Actionable, real-time information remained elusive since data from one sensor platform is often incompatible with data from other field resources. It was clear to OnFarm that ThingWorx was well-engineered and provided exactly the IoT development framework they needed. Utilizing ThingWorx would eliminate the core engineering OnFarm would have otherwise needed to develop in-house. With ThingWorx, OnFarm can deliver real-time, field-data-driven reports providing business intelligence to their customers.

PRODUCT HANDLES

- Device Management
- M2M Gateway
- Data Services
- Web Services
- Analytics

NOTABLE CUSTOMERS

- Joy Global
- Sysmex
- GE Power & Water
- OnFarm
- Vantron
- LumenData
- Devicify
- All Traffic Solutions

BLOG thingworx.com/blog

TWITTER @ThingWorx

WEBSITE thingworx.com

JSON, HTTP, and the Future of IoT Protocols

BY MATT BUTCHER

QUICK VIEW

01
JSON is a problematic protocol for IoT products: it has a narrow range of types, a lack of schema, fields are essentially unordered, and it's not a space-efficient encoding.

02
HTTP has its own IoT problems: it tends to be bloated, and it's too focused on short-lived connections.

03
While REST is a strong candidate for future IoT products, there are a host of growing alternatives including HTTP/2, CoAP, QUIC, CBOR, and Protobuf.

With the rise of web-based APIs, we have come to think of REST (Representational State Transfer) as being synonymous with JSON over HTTP. Unsurprisingly, JSON has supplanted XML as the data format of choice for the web. While early IoT technologies have embraced the JSON/HTTP mix, that could soon be changing. The concept of REST will survive, but JSON and HTTP may not be the common language of IoT data interchange for much longer.

At its core, REST is an architectural pattern for uniformly accessing and modifying a resource. One entity (the server) is the authority over the current state of an object. Other entities may request a "representation" of the current object, and may also send requests to create, modify, or delete the object. The current popular REST model uses URIs to identify objects ("/lamp/1234"), HTTP verbs to specify an action, and JSON to represent the object. To fetch an object, a client may send an HTTP request to "GET /lamp/1234". The server may respond with an HTTP 200 and a body containing JSON data.

The HTTP/JSON model is deeply entrenched in web APIs, and its popularity has naturally seeped into IoT technologies. Samsung, Nest, and Apple have all published APIs that rely on JSON over HTTP, but this early trend will wane. While the REST model works well for the distributed network that makes up the new IoT world, HTTP 1.1 and JSON are not the right fit.

WHAT'S WRONG WITH JSON?

When JavaScript legend Douglas Crockford introduced the JSON format, he was interested in specifying a format that eased data interactions between web applications and JavaScript-based clients. Because it's a lightweight alternative to XML, JSON quickly gained traction among web developers, and later reached a more general audience.

Several features of JSON make it a great candidate for general purpose data interchange. First, it is schemaless; as long as the JSON is well-formed, it is valid. Second, JSON supports a minimal and straightforward set of data types: strings, numbers, booleans, objects, arrays, and a null value. Third, the data is represented in JavaScript syntax, which makes it both human-readable and easily parseable. One would be hard pressed to find a popular programming language that does not have at least one JSON parser.

These features make JSON a useful general purpose format, but the typical use cases for IoT may cause us to doubt JSON's appropriateness for the embedded systems that together make up the smart device landscape. IoT devices typically need to optimize along the following lines:

- Keep network traffic small and fast.
- Minimize the amount of raw computation for network encoding and decoding.
- Use only small amounts of memory and storage.

Devices may run with less than a megabyte of memory or storage, and often run on small batteries. For power consumption reasons, they may only be on the WiFi network for a few seconds at a time, and sometimes only a few times a day. Even high-end hub devices are unlikely to have more than 25MB of storage at their disposal. For these devices, efficiency is key, especially when it comes to networking.

JSON is not the best candidate for meeting these requirements. First, in spite of its claims to leanness, JSON is not a space-efficient encoding. All data is expressed as ASCII strings, often with copious white space added. Every label field must be repeated in its entirety for each occurrence. Binary data must be escaped, though there is no standard method for doing so in JSON.

This leads to a second issue with JSON. The simplicity of the data format introduces complexity in implementation. JSON's simple types rarely match the types typically used in IoT programming. While languages like C support a broad array of numeric types, JSON's only numeric type is number. The official JSON specification, ECMA-404, does not even define the maximum size of a number field. This means that a JSON consumer must do a fair amount of examination to determine which underlying type best matches a given number. This is complicated by the fact that two or more fields with the same apparent structure and field names may contain different "types" of numbers. The field "age" may in one occurrence be an unsigned positive integer and in another be a floating point.

The problem is compounded by JSON's lack of schema. Arrays can contain any number of types, and there are no constraints on how the fields of an object are used or whether they are used consistently. Developers rely only on convention to determine what data a JSON structure will contain.

Finally, there is the problem of interpreting a JSON data structure. Fields are essentially unordered (except for arrays). Valid JSON may, as noted above, contain arbitrary data that violates expectations, and it is up to the parser to solve any given data structure. Strategies used for efficient field-level processing generally don't work well with JSON. Practically speaking, that means parsing an entire object and storing the results in memory.

"JSON is clearly not the best technology for data encoding"

JSON is clearly not the best technology for data encoding. HTTP 1.1, the other half of the ubiquitous REST implementation, does not come out looking much better.

WHAT'S WRONG WITH HTTP?

HTTP 1.1 has served web developers well. It is flexible, straightforward, enjoys broad implementation, and has a huge developer base. But the HTTP faults that have irked web developers for years may have an even bigger impact on IoT developers.

Like JSON, HTTP tends toward the bloated side. HTTP headers are a good example. As plain text strings with no compression of any sort, they bloat the network protocol.

Network usage is another one of HTTP's shortfalls. The original HTTP specification was designed around the idea of short-lived network connections. The client opens a connection and then requests a page, the server delivers it, and the connection is closed. But the average web page now may fetch over a dozen resources at once. HTTP 1.1 introduced some capabilities for keeping connections opened and reusable for short periods of time, but HTTP essentially remained focused on short-lived connections.

Consider the networking aspect of an IoT device. Establishing a connection is expensive in terms of power and time, especially with SSL/TLS negotiation included; every added connection brings along a substantial computational hit. Repeatedly opening heavyweight network connections is an unnecessary drain on resources.

In the IoT world, every byte sent and received from an embedded device takes its toll on performance. A good IoT protocol not only makes it easy for the developer to send the right information, but it also reduces the burden on the device and its network. The HTTP payload model is great for IoT, but a better protocol would streamline security, optimize transmission sizes, and focus on multiplexing requests and responses over long-lived network connections.

THE FUTURE IS BINARY

REST is a good model for IoT. Each device can easily make its state information available, and can standardize on a way to create, read, update, and delete that data. Developers can quickly build a mental REST model for many IoT devices. Get the state of the lightbulb: it is off. Send a request to turn it on. Get the current temperature from the space heater: it is too hot. Send a lower target temperature. The model seems to intuitively match the problem space.

But what is to be done about JSON and HTTP? IoT developers need REST without needless bloat.

For JSON, the future of IoT is bleak: a host of better-suited encodings are flooding the space. Apache Thrift and Google's Protocol Buffers (Protobuf) each provide binary encodings better suited for constrained devices, and both have the advantage of automatically enforced schemas. CoAP, an emerging standard for IoT communication, defines an encoding called CBOR. CBOR is self-describing, and the encoding is focused on producing small message sizes. Even the venerable ASN.1 family of encodings may be getting a new IoT spin. All of these provide encoding characteristics better suited to embedded devices than JSON.

IoT developers need REST without needless bloat

For HTTP, the story may play out differently. True, it will face some competition; for example, CoAP defines a concise REST-like transport protocol that is a compelling alternative to HTTP 1.1. But growing out of Google's SPDY efforts, the HTTP/2 standard indicates that HTTP may have solved its own problems.

HTTP/2 shows a renewed interest in network performance. Headers in HTTP/2 are efficiently encoded. The protocol

Both QUIC and HTTP/2 are well-designed, and are likely to gain acceptance in the emerging IoT space.

supports multiplexing many streams of data over one connection, as well as server-initiated pushes, and the reconstruction of the protocol maintains SSL/TLS as a central piece. One SSL/TLS negotiation can then protect multiple streams of data, thus reducing the setup overhead, but maintaining a high degree of security.

In addition to HTTP/2 and CoAP, the emerging QUIC protocol may also gain traction among resource-constrained devices. QUIC, also a Google protocol drawing from SPDY, trades TCP for UDP. By removing some of TCP's connection management overhead, QUIC aims to reduce latency, particularly during initial establishment of a network connection.

Because QUIC and HTTP/2 are based on a similar protocol stack, the competition between the two is not a zero-sum game. Both are well-designed, and are likely to gain acceptance in the emerging IoT space.

THE TURNING TIDE

The REST model is a strong fit for IoT. However, the traditional REST implementation of JSON over HTTP is ill-fitting at best. JSON's string-oriented payloads are no match for binary encodings when it comes to data transmission in terms of speed and ease of parsing. Encodings like CBOR and Protobuf are compelling alternatives to JSON.

In contrast, the HTTP/2 specification indicates that HTTP may remain the application protocol of choice. And its emerging sister protocol, QUIC, will compliment and strengthen the position of web protocols in the IoT space.

MATT BUTCHER is a core contributor to the Deis platform at EngineYard. He has worked on numerous cloud and IoT technologies at places like Google, HP, and About.Com. He holds a Ph.D. in Philosophy, and teaches in the Computer Science department of Loyola University Chicago. Matt is the author of dozens of articles and eight technical books, the latest of which is *Go in Practice* (Manning). Matt loves a good cup of coffee and enjoys mountain biking. You can find Matt on twitter at @technosophos, and read his blog at technosophos.com.

diving deeper

INTO THE INTERNET OF THINGS

Top 10 #IoT Twitter Feeds

@POSTSCAPES @MJAYS @IANSKERRETT @TREVOR_HARWOOD @AHAWKINSON

@ADAMJUSTICE @IOTWATCH @SZBALAZS87 @THEIOT @ROBVANK

IoT Zones

IoT Zone

dzone.com/iot

The Internet of Things (IoT) Zone features all aspects of this multifaceted technology movement. Here you'll find information related to IoT, including Machine to Machine (M2M), real-time data, fog computing, haptics, open distributed computing, and other hot topics. The IoT Zone goes beyond home automation to include wearables, business-oriented technology, and more.

Mobile Zone

dzone.com/mobile

The Mobile Zone features the most current content for mobile developers. Here you'll find expert opinions on the latest mobile platforms, including Android, iOS, and Windows Phone. You can find in-depth code tutorials, editorials spotlighting the latest development trends, and insight on upcoming OS releases. The Mobile Zone delivers unparalleled information to developers using any framework or platform.

Cloud Zone

dzone.com/cloud

The Cloud Zone covers the host of providers and utilities that make cloud computing possible and push the limits (and savings) with which we can deploy, store, and host applications in a flexible, elastic manner. This Zone focuses on PaaS, infrastructures, security, scalability, and hosting servers.

Top IoT Podcasts

Farstuff
farstuff.com

IoT Podcast
iotpodcast.com

Peggy Smedley Show
peggysmedleyshow.com

Top IoT Websites

All the Internet of Things
alltheinternetofthings.com

Postscapes
postscapes.com

Internet of Things Council
theinternetofthings.eu

Top IoT Newsletters

IoT Weekly
iotweeklynews.com

O'Reilly IoT Newsletter
oreilly.com/solid/solid-newsletter.csp

IEEE IoT Newsletter
iot.ieee.org/newsletter.html

Executive Insights on The Internet of Things

QUICK VIEW

01 Many executives are concerned there aren't enough developers for IoT.

02 IoT applications should be agile and flexible, and not rely on a specific platform or operating system.

03 There must be a concerted effort to build solutions that serve to solve real-life problems.

BY TOM SMITH

In order to more thoroughly understand the Internet of Things space, we interviewed 22 executives with diverse backgrounds and experience with IoT technologies representing both the industrial and consumer product spaces.

Specifically, we spoke to:

***Chuck Sathrum**, VP - Energy, Embedded Logix · **Keith McKechnie**, Solutions Engineer, USAT · **Chuck Speicher**, Founder, Security Fabric Alliance · **Sean Lorenz**, Director of IoT Market Strategy, LogMeIn.com · **Darren Guccione**, CEO and Co-Founder, Keeper Security · **Andrew Trice**, MobileFirst Developer Advocate, IBM · **Kevin Pope**, COO and Co-Founder, MatterHackers · **Mikko Jarva**, CTO Intelligent Data, Comptel · **Michael Oblak**, CTO, RentingLock · **Bill Balderaz**, President, Fathom Healthcare · **Brad Bush**, COO, Dialexa · **Ameer Sami**, Founder and Chief Engineer, Ottomate · **Yannis Tsampalis**, CEO, DogStar Life · **Fred Bargetzi**, CTO, Crestron Electronics · **Mark Wright**, Director of Product Management, Ayla Networks · **Tony Paine**, CEO, Kepware Technologies · **Andreea Borcea**, Founder & Consultant, Efficient Entrepreneur · **Charles Wilson**, Principal, Fancy Company · **Beatrice Witzgall**, Founder, Lumifi · **Rich Carpenter**, Chief Strategist, GE Intelligent Platform Software · **Kevin Coppins**, General Manager - Americas, EasyVista · **Imad Mouline**, CTO, Everbridge*

Throughout our conversations, we found that executives were often lining up with consistent answers to our questions, and had similar opinions about the present and future of IoT and M2M technologies. This consistency helped us draw some major conclusions about how businesses are addressing opportunities and problems in the space from an executive level. The following are the key insights from our interviews:

01. There's not a single definition of IoT.

This isn't much of a surprise. Your definition depends on your perspective, background, and experience. By summarizing the diverse responses to the question of how they define IoT, we ended up with the following definition:

"The communication, connectivity, and computing ability of devices sharing data via the Internet to help improve products, services, responsiveness, and quality of life."

That being said, it's not all that important to have a universal definition of IoT. What is important is to have agreed-upon standards of connectivity and security to ensure a future of IoT technologies that can communicate and collaborate instead of existing in their own siloed ecosystems.

02. Twitter, TechCrunch, and face-to-face communications...

...are how executives are staying abreast of industry trends. Executives tend to follow specific individuals and companies that have credibility in the IoT space.

Respondents also receive tremendous value from face-to-face interactions with colleagues at conferences, events, and meetups. Just as important is meeting with clients and prospects to hear what's important to them and their customers.

Developers should avail themselves of every opportunity to meet with manufacturers, prospects, clients, end users, and other developers in the IoT space.

03. The biggest problem solved by IoT is real-time monitoring.

This is being realized in industrial and healthcare verticals right now. Unlike anything preceding M2M and IoT, users are now able to see data variances in real time and respond quickly to changing situations.

IoT makes monitoring possible where it wasn't possible before; it makes things simpler, less expensive, and more accurate where possible. The key is to know what the problems are you're looking to solve. For example, Goldman Sachs sees the potential opportunity for $305 billion in savings from digital healthcare in the near future. As much as $200 billion could be largely due to the elimination of redundant and wasteful expenditures. The other $100 billion is forecast to come from telehealth, which expands access to healthcare regardless of a patient's geographic location.

04. Diverse experience is key to developing successful IoT products.

Respondents are looking for people with experience wearing multiple hats across the development process, ranging from applications to cloud to security. Developers and other team members need to be able to see how things interconnect beyond just the level at which they are working.

Also important is the ability to interact with people. Manufacturers, who may not have experience with IoT, will need a lot of help integrating IoT devices into their products. Being able to interact with end-user customers gives everyone working on the product insight into what the customer needs and wants, as well as what's working for them and what isn't. It's invaluable to see the customer using your product in the real world.

05. The future of IoT is your imagination.

Ubiquity, possibilities, elegance, and simplicity of interacting with multiple devices is the ultimate vision for the future of IoT. The changing business model can already be seen in services like Netflix, Airbnb, Uber, and the millions of sensors monitoring shipments and production lines in industrial settings. Industries will be completely disrupted.

IoT will drive intelligent actions informed by data. It will help companies and people collaborate across silos and form communities across geographies. And though it'll bring many quality of life enhancements, M2M and IoT technologies will certainly displace many line workers with automated solutions, hence the need for a future workforce that understands computers, technology, and programming.

06. Be operating system agnostic.

Ideally, all devices should work together to build open systems. It's possible that a single platform and standard could evolve over time, but the focus right now needs to be on organizations being agile and flexible to meet market needs. This means creating products with a focus on connectivity and integration. The majority of respondents replied that their products are platform and OS agnostic—which means that these executives understand the push for open and connected technologies.

Two respondents were using their own proprietary operating systems for freestanding products. Everyone else was typically using a combination of operating systems (or an intentional lack of one) with the understanding that they need to be able to integrate with whatever their customers, end users, or manufacturers use.

07. Security and privacy are the biggest concerns.

When asked about concerns, positive responses far outweighed the negatives; however, virtually everyone said security needed to be addressed, especially for consumer IoT products. Articles like "Hackers Remotely Kill a Jeep on the Highway—With Me in It" in *Wired* are very disconcerting, but reinforce the consumer concern for data privacy.

Many executives were also concerned that the need for developers far outnumbers available candidates. As such, anything developers can do to diversify their skills in the areas of hardware development, cloud operations, remote protocol, data and system integration, algorithms, and cryptography will make them more valuable to emerging IoT industries.

08. There's a significant gap between M2M/Industrial IoT and hobby/consumer IoT with regard to security and standards.

Companies and standards bodies (e.g. OPC and IIC) need to identify a set of protocols and standards to ensure the security of information being shared between machines and connected services.

The scale of IoT is significantly greater than security solutions for systems focused on servers, desktops, or smartphones. It means building highly scalable solutions to deal with tens of billions of devices and endpoints.

Also, Industrial IoT projects typically have defined needs and products aren't being built "on spec." With consumer products, too many people are building solutions for which no problem exists. There needs to be a greater focus on solving problems over making cool gadgets.

09. Stay creative and don't be constrained by the existing technology.

If the idea is good enough, the technology will be developed to support and execute the idea, whether it's for industrial or consumer use.

Empower developers by having them eat their own dog food. Take the product you're working on to a facility and see how it interacts with other products and machines. Take the product home and see how your family likes and interacts with it. Build products on established, secure, and scalable platforms that can be changed in the field as you get more and more real-time data. If something's not working or not delivering the desired end-user experience, be able to fix it in the field without having to touch every device.

While many of the executives we interviewed are engineers still actively working on their own products, often the thoughts about what makes a technology valuable can vary within even the organization. We're interested in hearing from developers and other IT professionals if this kind of data offers real value to them—do you think it's helpful to see these kinds of executive insights? We welcome your feedback at research@dzone.com.

TOM SMITH is a Research Analyst at DZone who excels at gathering insights from analytics—both quantitative and qualitative—to drive business results. His passion is sharing information of value to help people succeed. In his spare time, you can find him either eating at Chipotle or working out at the gym.

IoT Security Checklist

In the DZone 2015 Internet of Things Survey, respondents said security was their biggest concern regarding the Internet of Things. Ideally, security should be a part of the initial development of both hardware and software, as retrofitting security into your system can lead to vulnerabilities. In order to effectively secure the Internet of Things, you need to approach security from three levels: on the device (sensors), on the network (gateways), and on the system (the cloud). This comprehensive checklist will maximize your IoT solution security and minimize your security risk.

DEVICE-LEVEL (SENSORS)

- ☐ Decide when sensor data should be encrypted
- ☐ Minimize the amount of data being stored on-device
- ☐ Write efficient, lightweight applications
- ☐ Develop hardware and software in tandem when possible
- ☐ Ensure your device is encrypted at sign-on via digital signature
- ☐ Limit the access a single device has to data being managed or to other devices on the network
- ☐ Authenticate a device with the network every time it sends or receives data
- ☐ Implement a firewall on-device
- ☐ Send small, minimal bandwidth patches to devices
- ☐ Don't allow devices to access open ports
- ☐ Maintain that data is confidential
- ☐ Define lifecycle controls for devices

NETWORK-LEVEL (GATEWAY)

- ☐ Encrypt the storage of gateway credentials
- ☐ Periodically rotate gateway passwords
- ☐ Establish a failsafe if the gateway detects a malicious sensor
- ☐ Process security from the sensors to ensure proper authentication
- ☐ Create mechanisms to alert an administrator of any gateway tampering
- ☐ Implement an authentication/authorization framework
- ☐ Operate in secure boot mode

SYSTEM-LEVEL (DATA CENTER)

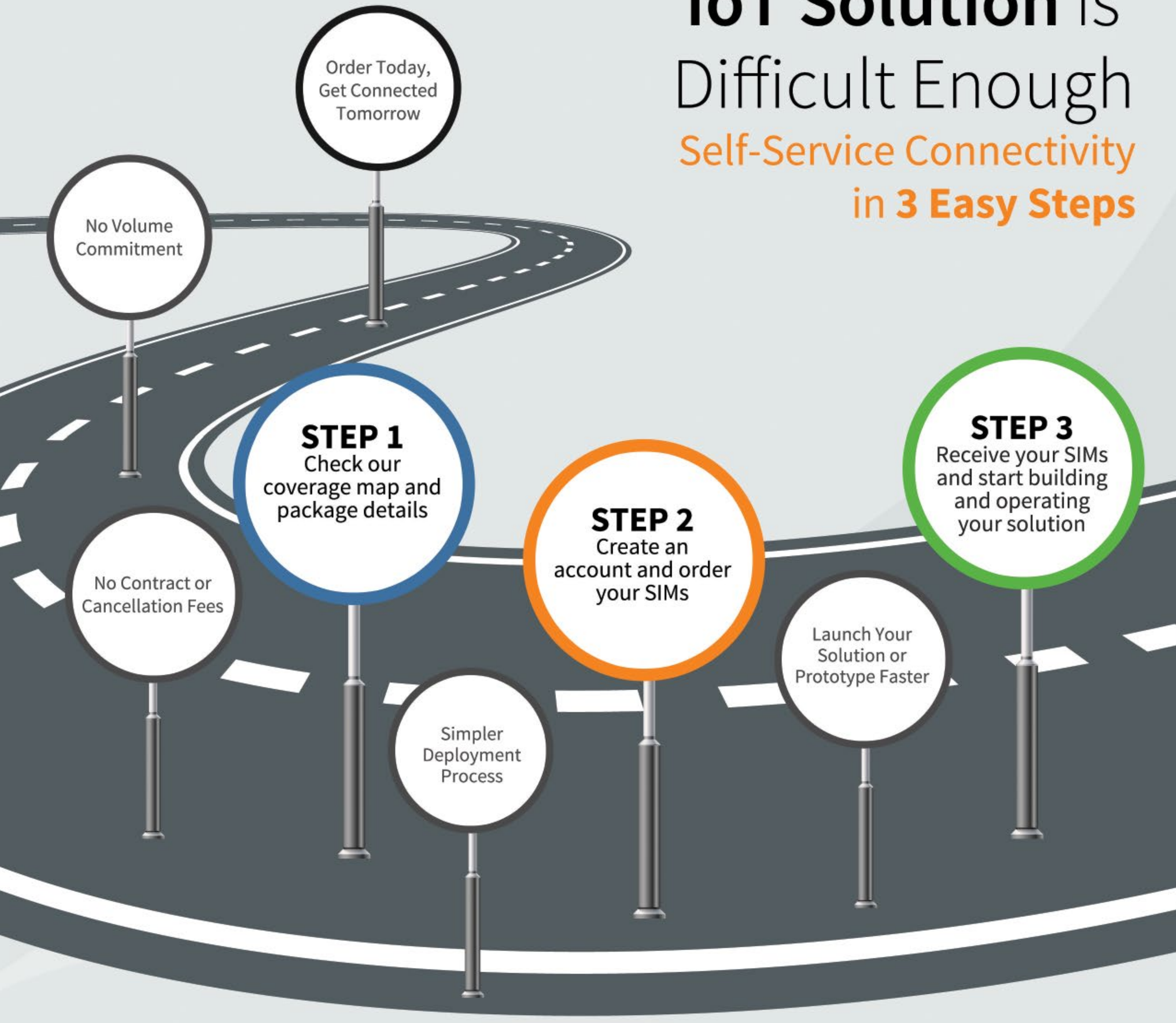
- ☐ Follow standard cloud security practices
- ☐ Authenticate all data transfers
- ☐ Collect only necessary data
- ☐ Monitor file integrity
- ☐ Create a layered defense with strong, regularly-updated firewalls
- ☐ Scan for open ports and close them
- ☐ Document MAC addresses for an added layer of authentication
- ☐ Implement a message broker
- ☐ Encrypt all data

SOURCES: bit.ly/windriver-iot and bit.ly/securityforearlyadoption

Neo™ POWERED BY AERIS The Fast Lane to IoT

Because Creating an **IoT Solution** is Difficult Enough

Self-Service Connectivity in **3 Easy Steps**


Order Today, Get Connected Tomorrow
No Volume Commitment
STEP 1 Check our coverage map and package details
No Contract or Cancellation Fees
Simpler Deployment Process
STEP 2 Create an account and order your SIMs
Launch Your Solution or Prototype Faster
STEP 3 Receive your SIMs and start building and operating your solution

Get Connected TODAY at neo.aeris.com/dzone

aeris®

Solutions Directory

This directory of platforms, middleware, software development kits, and hardware solutions provides comprehensive, factual comparisons of data gathered from third-party sources and the tool creators' organizations. Solutions are selected for inclusion in the directory based on several impartial criteria, including solution maturity, technical innovativeness, relevance, and data availability.

IoT PLATFORM

PRODUCT	LANGUAGES	MESSAGING	HOSTING	WEBSITE
Aeris	Language-independent	SMPP, SOAP	Hosted	aeris.com
AirVantage M2M Cloud	Java, Python, JavaScript, Ruby	MQTT, AMQP, OMA Lightweight M2M, HTTP	Hosted	sierrawireless.com
AWS Elastic Beanstalk	PHP, Java, Python, Ruby, Node.js, .NET, Go	N/A	Hosted	aws.amazon.com
Ayla IoT Platform	C, C++, C#, Java, Python	SMS, SMTP, HTTPS	Hosted	aylanetworks.com
Bluemix by IBM	Java, Node.js, Go, PHP, Python, Ruby on Rails	MQTT	Hosted	ibm.com/bluemix
Carriots	Java, Python, Groovy, C, C#	MQTT, HTTP	Hosted or On-Premise	carriots.com
CloudOne IoT Platform	Language-independent	MQTT	Hosted	oncloudone.com
DeviceHive by DataArt	Java, Python, C, C++, C#	Protocol Plug-in Architecture	Hosted or On-Premise	devicehive.com
Dweet.io by Bug Labs	Java, Python, C, C++, C#, Node.js, JavaScript	HTTPS	Hosted or On-Premise	dweet.io
Electric Imp	Squirrel	HTTP	Hosted	electricimp.com
Etherios Device Cloud	Java, Python, C, C++	HTTP, TCP/IP	Hosted	etherios.com
Everyware Device Cloud by Eurotech	Java	MQTT	Hosted or On-Premise	eurotech.com
EVRYTHNG Engine	All major languages	MQTT, WebSockets	Hosted	evrythng.com
Golgi	Node.js, JavaScript, Java	HTTP, MQTT, Zigbee	Hosted	golgi.io
Jasper Control Center	HTML, XML	SMS, TCP/IP, Proprietary Protocols	Hosted	jasper.com
Mender.io	Language-independent	TLS	Hosted	mender.io
Microsoft Azure	JavaScript, Python, .NET, PHP, Java, Node.js	AMQP	Hosted	azure.microsoft.com
Mojio	C#, JavaScript, PHP	WebSockets	Hosted	moj.io
Niagra by Tridium	HTML5, JavaScript	Protocol-independent	Hosted	tridium.com
Oracle Internet of Things Platform	Java	HTTP, MQTT	Hosted	oracle.com
Pivotal Cloud Foundry	Java, Ruby, Python, Go, PHP, Node.js	AMQP, MQTT, STOMP	Hosted	pivotal.io
Predix	All major languages	ModBus, OPC-UA, TCP Sockets, HTTP, HTTPS	Hosted	predix.io
Salesforce1 Platform	Java, JavaScript, Apex, Objective-C, Ruby	REST, SOAP	Hosted	salesforce.com
ThingFabric IoT Platform by 2lemetry	Java, Python, C, C++, C#	MQTT, AMQP, XMPP, CoAP, DDS, OMA Lightweight M2M	Hosted or On-Premise	2lemetry.com
ThingWorx, a PTC Business	Drag-and-drop interface	MQTT, AMQP, XMPP, CoAP, DDS, WebSockets	Hosted or On-Premise	thingworx.com
Xively by LogMeIn	Java, Python, C, C++, Objective-C	MQTT, AMQP, XMPP, CoAP	Hosted	logmein.com
Zatar by Zebra Technologies	Java, JSON, Python	CoAP, OMA Lightweight M2M	Hosted	zatar.com

DEV TOOLS

PRODUCT	CATEGORY	MARKET	DEVICES SUPPORTED	WEBSITE
Android Wear	API	Wearables	Galaxy Gear	android.com/wear
Apple HomeKit	SDK	Home Automation	iOS devices, Philips Hue, Texas Instruments, iHome, and Honeywell products	developer.apple.com
Belkin WeMo SDK	SDK	Home Automation	WeMo line of products	developers.belkin.com/wemo/sdk
Control4 DriverWorks SDK	SDK	Home Automation	Roku, Nest, Sonos, Kwikset, and others	control4.com
Fitbit	API	Wearables	Fitbit products	dev.fitbit.com
Jawbone UP	API	Wearables	Jawbone products	jawbone.com/up/developer
Nest Developer Program	API	Climate Control	Nest Products, "Works With Nest" Network	developer.nest.com
Pebble	SDK	Wearables	Pebble products	developer.getpebble.com
Philips Hue	SDK	Home Automation	Hue LED Lights	developers.meethue.com
Razer Nabu	SDK	Wearables	Razer Nabu Smart Watch	developer.razerzone.com

MIDDLEWARE

PRODUCT	LANGUAGES	MESSAGING	HOSTING	WEBSITE
MuleSoft Anypoint Platform	Java, Python, JavaScript	MQTT, AMQP, WebSockets, HTTP, TCP	Hosted or On-Premise	mulesoft.com
ProSyst mBS	Java	MQTT, CoAP, OMA Lightweight M2M	On-Premise	prosyst.com
Red Hat JBoss A-MQ	Java, C, C++, JavaScript, .NET	MQTT, AMQP, STOMP, Openwire, WebSockets	On-Premise	redhat.com
Vortex by PrismTech	Node.js, HTML, Java, JavaScript	DDS, CoAP, MQTT, AMQP, JMS, REST	Hosted	prismtech.com
WSO2 Platform	Java, JavaScript	MQTT, AMQP, XMPP	Hosted or On-Premise	wso2.com

IoT HARDWARE

PRODUCT	LANGUAGES	PRICE	ADD-ON SUPPORT	WEBSITE
Arduino Yún	Python, Ruby, Node.js, PHP	$55	Yes	arduino.cc
Artik 5 by Samsung	C, C++, Java, Groovy	TBD	Yes	artik.io
BeagleBone Black	Java, C, C++, Python, Perl, Ruby, Node.js	$89	Yes	beagleboard.org
Edison by Intel	JavaScript, C++, Python	$50	Yes	intel.com
Gizmo 2	Any	$200	No	gizmosphere.org
Kinoma Create	JavaScript	$150	No	kinoma.com
Mojo V3	Lucid	$75	Yes	embeddedmicro.com
Netduino 3 WiFi	Visual Basic	$79	Yes	netduino.com
Papilio Duo	Drag-and-drop interface	$88	Yes	papilio.cc
Raspberry Pi 2 Model B	Python, Scratch, C, C++, Ruby	$35	Yes	raspberrypi.org
Tessel 2	Node.js, JavaScript, Python, Rust	$35	Yes	tessel.io
TI Connected Launchpad CC3200 SimpleLink WiFi	C	$30	Yes	ti.com
Wiring S	C++	$27	No	wiring.org.co

diving deeper

INTO FEATURED INTERNET OF THINGS PRODUCTS

Looking for more information on individual IoT solutions providers? Nine of our partners have shared additional details about their offerings, and we've summarized this data below.

If you'd like to share data about these or other related solutions, please email us at research@dzone.com.

IBM.COM/BLUEMIX — **PLATFORM**

Bluemix
BY IBM

SECURITY CERTIFICATIONS
n/a

USE CASES
1. Smart Supply Chain
2. Medical
3. Smart Service Sector
4. Environmental

MESSAGING
MQTT

LANGUAGES
- Java
- Node.js
- Go
- PHP
- Python
- Ruby on Rails

ONCLOUDONE.COM — **PLATFORM**

CloudOne IoT Platform

SECURITY CERTIFICATIONS
- SSAE 16
- SOC 2
- FISMA
- ITAR

USE CASES
1. Transportation
2. Smart Supply Chain
3. Smart City
4. Medical

MESSAGING
MQTT

LANGUAGES
Language Independent

EUROTECH.COM — **PLATFORM**

Everyware Device Cloud
BY EUROTECH

SECURITY CERTIFICATIONS
n/a

USE CASES
1. Transportation
2. Smart Service Sector
3. Smart City
4. Smart Supply Chain

MESSAGING
MQTT

LANGUAGES
Java

AERIS.COM — **PLATFORM**

Neo
BY AERIS

SECURITY CERTIFICATIONS
Device-level authentication

USE CASES
1. Energy
2. Transportation
3. Smart Supply Chain
4. Environmental

MESSAGING
- SMPP
- SOAP

LANGUAGES
Language Independent

PREDIX.IO — **PLATFORM**

Predix

SECURITY CERTIFICATIONS
- ISO 27001
- AICPA SOC 1&2
- HIPAA
- FIPS 140-2

USE CASES
1. Medical
2. Transportation
3. Smart City
4. Environmental

MESSAGING
- Modbus
- OPC-UA
- TCP Sockets
- HTTP
- HTTPS

LANGUAGES
Language Independent

REDHAT.COM — **MIDDLEWARE**

Red Hat JBoss A-MQ

SECURITY CERTIFICATIONS
n/a

USE CASES
1. Smart Supply Chain
2. Medical
3. Smart Service Sector
4. Environmental

MESSAGING
- MQTT
- AMQP
- STOMP
- WebSockets

LANGUAGES
- C
- C++
- Java
- JavaScript
- .NET

SALESFORCE.COM — **PLATFORM**

Salesforce1

SECURITY CERTIFICATIONS
- ISO 27001
- SSAE 16
- SOC 2
- FedRAMP

USE CASES
1. Wearables
2. Smart Supply Chain
3. Medical
4. Transportation

MESSAGING
- REST
- SOAP

LANGUAGES
- Java
- JavaScript
- Apex
- Objective-C
- Ruby

THINGWORX.COM — **PLATFORM**

ThingWorx, a PTC Business

SECURITY CERTIFICATIONS
- SSAE 16
- SOC 2

USE CASES
1. Smart Supply Chain
2. Medical
3. Smart Service Sector
4. Environmental

MESSAGING
- MQTT
- AMQP
- XMPP
- CoAP
- DDS
- WebSockets

LANGUAGES
Drag-and-drop interface

WSO2.COM — **MIDDLEWARE**

WSO2 Platform

SECURITY CERTIFICATIONS
n/a

USE CASES
1. Environmental
2. Transportation
3. Medical
4. Smart Service Sector

MESSAGING
- MQTT
- AMQP
- XMPP

LANGUAGES
- Java
- JavaScript

glossary

ACTUATOR A mechanism that performs a physical task based on input from a connected system.

ADVANCED MESSAGE QUEUING PROTOCOL (AMQP) An open application layer protocol for message-oriented middleware with a focus on queuing, routing (P2P, PubSub), security, and reliability.

BLUETOOTH LOW ENERGY (BLE) A wireless personal area network (PAN) aimed at devices with reduced power consumption and cost while maintaining a similar communication range to regular Bluetooth.

CONSTRAINED APPLICATION PROTOCOL (COAP) An application layer protocol used in resource-constrained devices that allows internet connectivity and remote control.

EMBEDDED DEVICE/SYSTEMS A computer with a dedicated function within a larger mechanical or electrical system; it is embedded as part of a complete device.

ENDPOINT DEVICE An internet-capable device on a TCP/IP network.

FLOW-BASED PROGRAMMING (FBP) A programming paradigm that views applications as sets of data-streams affected by discrete processes, as opposed to monolithic systems with many interacting parts.

GATEWAY A data communication device that connects a host network to a remote network.

GEOFENCING A technology that creates virtual boundaries around a physical area in order to trigger an action on a connected device, usually through a combination of GPS and RFID tags.

HOME AUTOMATION A combination of hardware and software solutions that allow for the control and management of electronics, appliances, and devices within a home.

iBEACON A small network transmitter used to identify, track, and interact with connected systems using Bluetooth Low Energy. It's an Apple trademark, but it is also available on Android devices.
v

INTERNET OF THINGS (IOT) A network of objects (such as sensors and actuators) that can capture data autonomously and self-configure intelligently based on physical-world events, allowing these systems to become active participants in various public, commercial, scientific, and personal processes.

INTERNET PROTOCOL SUITE (TCP/IP) The language a computer uses to access the Internet. It consists of a suite of protocols designed to establish a network of networks to provide a host with access to the Internet.

INTERNET PROTOCOL VERSION 4 (IPV4) An internet layer protocol that provides end-to-end transmission across multiple IP networks, and can utilize 32-bit IP addresses.

INTERNET PROTOCOL VERSION 6 (IPV6) An Internet layer protocol that provides end-to-end transmission across multiple IP networks, and can utilize 128-bit IP addresses.

IPV6 OVER LOW-POWER WIRELESS PERSONAL AREA NETWORKS (6LOWPAN) This refers to technology that uses IPv6 for a diverse range of hardware applications, including resource-restricted devices for the Internet of Things.

LIGHTWEIGHT PROTOCOL Refers to any protocol that has a lesser and leaner payload when being used and transmitted over a network connection.

LONG RANGE COMMUNICATION PROTOCOLS Used to refer to universal long range radio frequencies for multi-generation wireless standards such as 2G, 3G, 4G, and 4G LTE.

MACHINE-TO-MACHINE (M2M) Refers to a network setup that allows connected devices to communicate freely, usually between a large number of devices; M2M often refers to the use of distributed systems in industrial and manufacturing applications.

MESH NETWORK A type of network topology in which a device transmits its own data and also serves as a relay for other nodes by providing the most efficient data path through routers.

MESSAGE QUEUING TELEMETRY TRANSPORT (MQTT) A lightweight messaging protocol that runs on TCP/IP protocol. It is designed for communicating with small devices in remote locations with low network bandwidth.

MICROCONTROLLER (MCU) A small computer on a single integrated circuit designed for embedded applications and used in automatically controlled embedded systems.

NEAR-FIELD COMMUNICATION (NFC) A feature, based on technical standards, that allows devices to establish radio communication with other nearby systems or mobile devices.

PERSONAL AREA NETWORK (PAN) A network created through the interconnection of information technology devices within the context of a single user.

RADIO FREQUENCY IDENTIFICATION (RFID) A technology that incorporates the use of electromagnetic coupling and radio frequency to identify objects and persons. It consists of three components: an antenna, transceiver, and transponder.

SENSOR A device or component that perceives and responds to physical input from the environment.

SENSOR NETWORK A group of sensors with a communications infrastructure intended to monitor and collect data from multiple locations.

SINGLE-BOARD COMPUTER (SBC) A complete computer built on a single circuit board with all the components required of a functional computer.

SYSTEM ON A CHIP (SOC) An integrated chip that is comprised of electronic circuits of multiple computer components to create a complete device.

TRANSMISSION CONTROL PROTOCOL/ INTERNET PROTOCOL (TCP/IP) A basic client/server model of communication protocol for the Internet and private networks.

WEARABLES Connected devices that can be equipped with different types of sensors and are worn on a person's body. They are meant to monitor, collect, and quantify data about a person's life and environment, and allow them to interface with that data.

WIFI (802.11) A wireless local area network (WLAN) that uses radio waves to provide wireless high-speed Internet and network connections.

Z-WAVE A wireless protocol for home automation that communicates using a low-power radio frequency technology specifically designed for remote control applications.

ZIGBEE An open standard for wireless communication designed to use low-power digital radio signals for personal area networks (PAN); it is used to create networks that require a low data transfer rate, energy efficiency, and secure networking.

Tell Your Boss You Need To Go.

Minds + Machines 2015
Sept 29 – Oct 1, San Francisco

Help your company invest in innovation by attending Minds + Machines, the premier event for the Industrial Internet. Find out how to use GE Predix to build industrial-strength apps that will boost productivity and profitability. **Code the Industrial Internet.**

Register at gemindsandmachines.com.