DZone

THE DZONE GUIDE TO

Internet of Things

Applications, Protocols, and Best Practices

VOLUME IV

BROUGHT TO YOU IN PARTNERSHIP WITH

Digi-Key Electronics

InterSystems Health | Business | Government

NEURA

DEAR READER,

We are living in a time where the Internet of Things is truly ubiquitous in every ecosystem and industry. There are use cases embedded in all industries that utilize technology from home automation to manufacturing, healthcare and smart grid development. It's only set to get bigger as industry analysts Gartner forecasted in February this year that 8.4 billion connected things will be in use worldwide in 2017, up 31 percent from 2016, and will reach 20.4 billion by 2020. Total spending on endpoints and services will reach almost $2 trillion in 2017.

To date, IoT's most compelling use case has been in the industrial workplace, where whole sectors have been transformed by its ability to solve problems that have plagued traditional industries for centuries, from machine repairs and supply chain processing to worker shortages and waste management. Similarly, today's farmer is most likely to rely on a smartphone than a pitchfork as they utilize IoT data to get the most accurate information about the weather, growing conditions, and their farm's soil quality – metrics previously unavailable or difficult to attain in real-time.

At the same time, we see a whole new field of intersectional technologies rubbing shoulders with IoT: such as AI, Big Data, and machine learning. Through this we get robots that not only respond to commands but can diagnose and treat their own technical problems and teach tasks to other robots, removing the need for human direction. In health science, wearable tech provides real-time physiological data that can be shared remotely with health professionals, changing not only the doctor-patient relationship but how health care is delivered. And before long, autonomous transport will become a reality, with the birth of new driverless vehicles that change our relationship with the car forever.

IoT will not only introduce new tech, but also new jobs, thanks to the growing list of hardware platforms, networks, operating systems, cloud services, endpoint management tools, and ecosystems. The demand for software engineers is substantial with career opportunities ranging from API design, data integration, and microcontroller programming, to name a few. In simple terms, data collection, storage, and analysis will feature highly in more and more jobs today and in the future.

However, like any sector, IoT is not without its growing pains. Each use case requires its own approach and protocol depending on what is being built and how it is to be powered with range, data, power and battery life all needing to be considered. As further use cases are constantly being refined and expanded and more and more products become connected, new protocols are needed. Yet poor physical environments, a lack of Interoperability, and power constraints challenge the effectiveness of the technology.

Then of course, there's the elephant in the room: device and data security. At its best, a security breach is a minor inconvenience like your anonymized wearable data being posted online. At its worst, it's botnet attacks and cyber hacking that could disable life-saving hospital equipment, shut down an entire city's traffic lights or disable local utilities. Thus, we can expect to see just as many careers emerge in cyber security, insurance, and asset management as we will in data analytics and IoT architecture.

In this guide, we'll provide you with a detailed understanding of the rapidly expanding sector that is IoT. We'll cover AI, predictive maintenance, IoT software stacks, consensual software, and of course privacy to name a few areas. Research and use cases will give you a feel for where IoT is right now and where it will head in the future. Get ready to get excited and learn about this fascinating part of technology and its role in the future.

BY CATE LAWRENCE, ZONE LEADER, DZONE

TABLE OF CONTENTS

PRODUCTION

CHRIS SMITH
DIRECTOR OF PRODUCTION

ANDRE POWELL
SR. PRODUCTION COORDINATOR

G. RYAN SPAIN
PRODUCTION PUB. EDITOR

ASHLEY SLATE
DESIGN DIRECTOR

MARKETING

KELLET ATKINSON
DIRECTOR OF MARKETING

LAUREN CURATOLA
MARKETING SPECIALIST

KRISTEN PAGÀN
MARKETING SPECIALIST

NATALIE IANNELLO
MARKETING SPECIALIST

MIRANDA CASEY
MARKETING SPECIALIST

JULIAN MORRIS
MARKETING SPECIALIST

EDITORIAL

CAITLIN CANDELMO
DIRECTOR OF CONTENT + COMMUNITY

MATT WERNER
PUBLICATIONS COORDINATOR

MICHAEL THARRINGTON
CONTENT + COMMUNITY MANAGER

MIKE GATES
SR. CONTENT COORDINATOR

SARAH DAVIS
CONTENT COORDINATOR

TOM SMITH
RESEARCH ANALYST

JORDAN BAKER
CONTENT COORDINATOR

ANNE MARIE GLEN
CONTENT COORDINATOR

BUSINESS

RICK ROSS
CEO

MATT SCHMIDT
PRESIDENT + CTO

JESSE DAVIS
EVP & COO

MATT O'BRIAN
DIRECTOR OF BUSINESS DEV.

CHRIS BRUMFIELD
SALES MANAGER

ALEX CRAFTS
DIRECTOR OF MAJOR ACCOUNTS

JIM HOWARD
SR ACCOUNT EXECUTIVE

JIM DYER
ACCOUNT EXECUTIVE

ANDREW BARKER
ACCOUNT EXECUTIVE

BRIAN ANDERSON
ACCOUNT EXECUTIVE

ANA JONES
ACCOUNT MANAGER

TOM MARTIN
ACCOUNT MANAGER

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.

SPECIAL THANKS to our topic experts, Zone Leaders, trusted DZone Most Valuable Bloggers, and dedicated users for all their help and feedback in making this guide a great success.

Executive Summary

BY MATT WERNER
PUBLICATIONS COORDINATOR, DZONE

The Internet of Things industry has been predicted to exceed $470 billion by the year 2020 between device manufacturers, software developers, and network providers. However, only 31% of the DZone audience has developed an IoT application or device in their personal or professional lives—but the majority of respondents are interested in doing so in the future. For those who haven't delved into the Internet of Things yet, there are probably several questions they still have. Where can I start? What's the difference between messaging protocols, and why can't we just have one? Is it actually practical? To help answer these questions, DZone has collected five expert articles and surveyed 535 tech professionals who either had developed IoT applications or devices, or were interested in doing so, to get a sense of how this still-nascent field is evolving and help developers dive in head-first.

THE REALITY OF IoT IS SINKING IN

DATA 65% of respondents believe that IoT will be important to their organizations in the future, up from 48% who believed so in last year's survey. Only 11% believe it will not be important, down from 24% last year. While they may not have worked on an IoT project before, 62% are interested in working on IoT projects in the future.

IMPLICATIONS Connected devices and the Internet of Things are becoming more and more engrained in everyday lives, and developers are starting to realize its importance. For all developers interested in IoT development, 63% were interested in either getting their feet wet with, or continuing to work with, personal projects, such as devices built from development boards like the Raspberry Pi.

RECOMMENDATIONS Experimenting with hobbies or personal IoT projects will be useful for building knowledge of the space and for career advancement as the IoT becomes more of a strategic focus for one's organization. For more details on why learning to develop for the Internet of Things is important, see Pradeep Peiris' article on Ubiquitous Computing on page 14, which goes into detail on how IoT and mobile technologies will permeate most aspects of everyday life.

PAIN POINTS ARE MORE SPECIFIC

DATA While 45% of readers are most concerned about unpredictable physical environments, many respondents recognized that other pain points of IoT development were more of a concern than they originally thought. 44% of users are worried about power constraints, up from 40% last year. Latency challenges are also more of a concern (39%, up from 35% last year). The biggest jump between this year's and last year's results is the concern around the volume of unstructured data, which saw a 9% increase (34% this year, up from 25%).

IMPLICATIONS Concerns about general "device reliability" decreased by 5%, from 46% last year to 41% this year, while several other areas saw a significant increase in concern, suggesting that readers have become more knowledgeable about what exactly concerns them in regards to IoT development. Big Data storage, analytics, and visualization will be a major part of IoT development and maintenance, as sensors will be constantly reading and sending new data every second, so organizations need to be prepared to handle it.

RECOMMENDATIONS Developers will need to be conscientious of developing applications that don't use too much power, and either work closely with IT networking professionals, or learn about it themselves. Due to unpredictable environments, a monitoring system should be adopted to check on the health of all devices in a network, and make plans to replace a device if it fails or needs a change of batteries. A solution to store and process the enormous amounts of data is also critical to make time-sensitive business decisions. Be sure to read *DZone's Guide to Big Data*, and investigate various Big Data and IoT data analytics solutions in our solutions directories.

TRENDS IN PROTOCOL USAGE

DATA The most popular protocols within the DZone audience are Wi-Fi Direct (38%), Bluetooth LE (Low-Energy) (31%), AMQP (24%), and MQTT (24%). Wi-Fi Direct was most popular among those developing for drones (35%) and environmental projects (34%). MQTT was favored among those working on geofencing (33%) and smart grid/city projects (32%), AMQP is also popular among developers working on geofencing applications (33%).

IMPLICATIONS Since few users have actually developed IoT applications or devices, the number of those who have used protocols is a fairly small percentage of the total respondents. Though it was only used by one in ten developers in production, a third of those who have used MQTT have used the protocol for large-scale applications. Wi-Fi Direct is used both by hobbyist developers and for certain large-scale projects.

RECOMMENDATIONS When it comes to protocols, one size does not fit all. Each protocol has different strengths and weaknesses, often dealing with different layers of the IoT stack, and developers should learn about them now, rather than waiting for an IoT consortium or nonprofit group try and create a catch-all protocol to unite everyone together. The sooner you can get developing for either a hobbyist project or even a small-scale environmental or home automation system, the better. For a detailed example on three popular protocols, refer to Paolo Patierno's article comparing use cases for MQTT, AMQP, and HTTP, and how each protocol operates, on page 24.

Key Research Findings

BY G. RYAN SPAIN
PRODUCTION COORDINATOR, DZONE

535 people completed *DZone's 2017 Internet of Things* survey. Demographics of these respondents include the following:

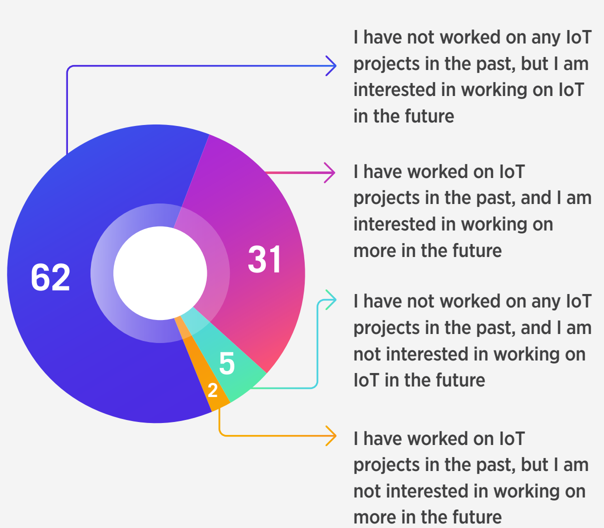
- 22% of respondents work at organizations with more than 10,000 employees; 21% at organizations between 1,000 and 10,000 employees; and 19% at organizations between 100 and 1,000 employees.
- 38% of respondents work at organizations based in Europe, and 29% at organizations based in the US.
- The average respondent has 14.3 years of experience as an IT professional. 32% had 20 years of experience or more; 68% has 10 years or more.
- 33% identify as developers or engineers; 19% as developer team leads; and 17% as software architects.
- 80% work at companies that use Java; 75% work at companies that use JavaScript (either client-side, server-side, or both); 44% work at companies that use Python.

AS TIME GOES BY

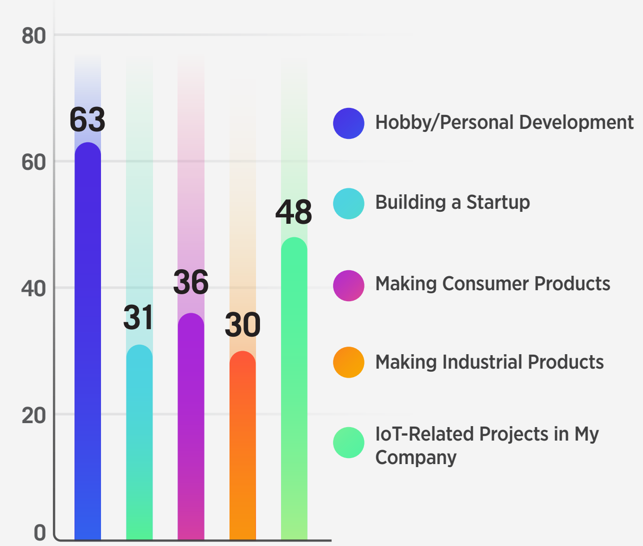
Interest in working directly with IoT projects still revolves largely around hobby or personal development. 31% of respondents to our 2017 IoT survey said they have worked on one or more IoT projects before, showing little-to-no change from last year's 30%, but another 62% of respondents, while they have not worked on IoT in the past, are interested in trying it out in the future. Of this 93% of survey respondents, 63% are interested in IoT for hobby or personal development, well ahead of the 48% interested in working on IoT projects in their organization, or the 36% interested in creating consumer products. However, the recognition of the significance of IoT within respondents' organizations has increased since last year's survey. This year, 65% of respondents believe that IoT is currently relevent in their organization, as opposed to 43% last year, and while last year almost one in four respondents did not think IoT would be relevent to their business in the future (24%), this year that ratio dropped to about one in ten (11%).

Challenges experienced while developing for the Internet of Things are shifting along with the technologies used

How would you describe your relationship with IoT?



In which contexts are you interested in IoT?



in the IoT technology stack. Unpredictable physical environments remain the number one challenge IoT developers face, with 45% of respondents experiencing that difficulty (compared to 48% last year). But concerns about device reliability have dropped 5% from last year, from 46% to 41%, now overtaken by power constraints (40% in 2016, 44% in 2017) as an IoT challenge. Device addressing and discovery dropped from 39% in 2016 to 31% in 2017, while there were increases in challenges with latency (35% in 2016, 39% in 2017) and the volume of unprocessed data (25% in 2016, 34% in 2017).

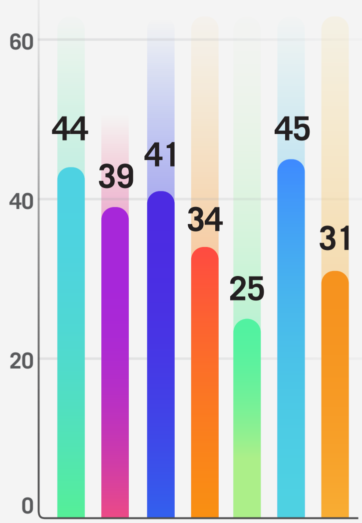
A BOARD BY ANY OTHER NAME

There are hundreds of single-board computers to choose from that can be used for a vast array of IoT projects, but by far the most popular boards among our IoT survey respondents were the Raspberry Pi (53%) and Arduino (28%) boards (note: our survey did not specify particular models of these boards, e.g. the Raspberry Pi 3 or Arduino Uno), but roughly half of respondents who said they own one of these boards (56% for Raspberry Pi, 51% for Arduino) also said they have not worked on IoT projects in the past. Arduino owners, however, are 5% more likely to be interested in hobby/personal IoT development than Raspberry Pi owners (71% vs. 66%). Arduino was also the most frequently mentioned technology when we asked respondents with IoT experience what they had developed on or for in the past, and outside of "smart home automation" and "drone" technology, Arduino and Rasberry Pi were the most mentioned IoT technologies respondents were interested in adopting within the next 6 months.

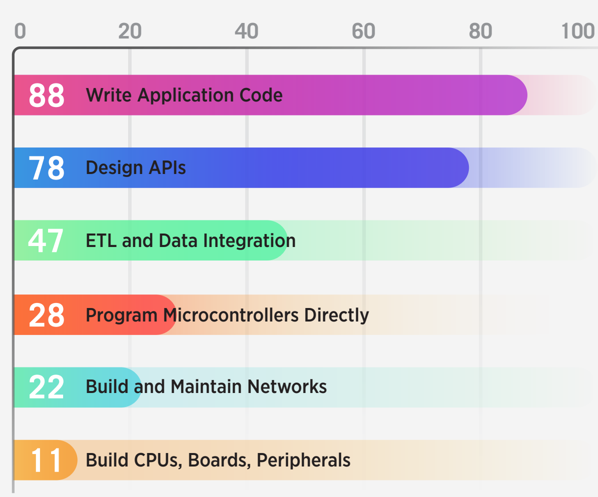
PROTOCOL POPULARITY

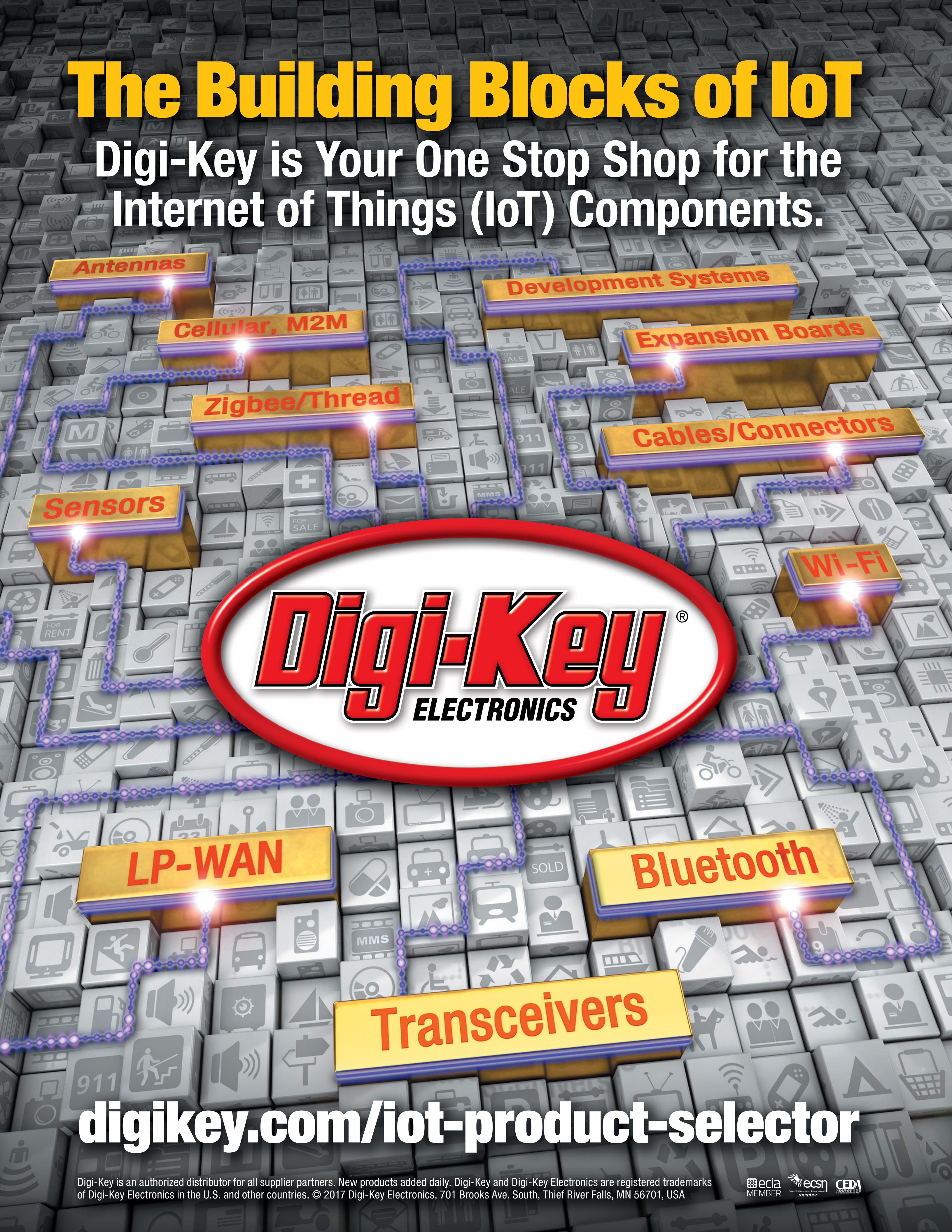
The relatively low number of respondents who have actually developed for the Internet of Things directly impacted usage numbers seen for IoT-related protocols. Even the most popularly used protocols — Bluetooth LE and Wi-Fi Direct — were used by only a minority of respondents. 14% of respondents said they have used Wi-Fi Direct in production, and 8% said they had used Bluetooth LE; for non-production environments, 24% had used Wi-Fi Direct and 23% had used Bluetooth LE. Protocols 6LoWPAN and CoAP were completely unknown to 71% and 70% of respondents, respectively. Looking at protocol usage for respondents who have developed for IoT before showed some protocol favorites: Wi-Fi direct was a particularly popular production protocol for respondents who had worked on drone projects and those who worked on environmental IoT projects, with 35% and 34%, respectively, using this protocol in production. MQTT was favored in production among respondents who had worked on geofencing projects and smart grid/city projects, garnering 33% and 32% for usage in production, respectively — much higher than MQTT's 10% overall average for the survey. Respondents working on geofencing projects also used AMQP heavily, with 33% of projects using the protocol in production, compared to AMQP's 14% overall survey average.

Which of the following challenges have you encountered while developing IoT applications?



What task(s) are you likely to perform while developing an IoT product?




The Building Blocks of IoT
Digi-Key is Your One Stop Shop for the Internet of Things (IoT) Components.
Antennas
Development Systems
Cellular, M2M
Expansion Boards
Zigbee/Thread
Cables/Connectors
Sensors
Wi-Fi
Digi-Key®
ELECTRONICS
LP-WAN
Bluetooth
Transceivers
digikey.com/iot-product-selector
Digi-Key is an authorized distributor for all supplier partners. New products added daily. Digi-Key and Digi-Key Electronics are registered trademarks of Digi-Key Electronics in the U.S. and other countries. © 2017 Digi-Key Electronics, 701 Brooks Ave. South, Thief River Falls, MN 56701, USA
ecia MEMBER
ecsn member
CEDA

LPWAN and LR-LAN's for the IoT

There are a seemingly unlimited number of wireless connectivity options for you to connect anything and everything to your IoT platform. Trying to wade through an ever-changing technology landscape can be very difficult but there are two significant technologies that have been through a lot of work to be fine-tuned for the IoT; Low-Power Wide-Area Networks (LPWANs) in Cellular and unlicensed bands and Long-Range Local Area Networks (LR-LANs) with advancements in Bluetooth and Wi-Fi. Typically, you will be forced to make a compromise in your technology decisions, but with all the options available do you really need to?

IOT BY 3GPP

The 3rd Generation Partnership Project, 3GPP, has put work into the last few user equipment specification releases to better enable IoT connectivity. Release 8 included LTE CAT-1, which enables a solid 3G replacement technology and is great for applications that need to be always on or require low latency. LTE CAT-M1 offers a reduced complexity transceiver that enables a lower cost design and power saving features to enable battery operation. M1 only requires a single antenna, which further reduces system costs. Most Mobile carriers have or will soon begin implementing M1 across their global networks.

UNLICENSED BANDS

Technologies like LoRa, Sigfox, and others are utilizing unlicensed bands in each region to enable Low-Power Wide Area Networks, or LPWAN. These technologies enable long range connections from a central base station, similar to Cellular networks. LoRa operates in the 10s of kbits/second, while Sigfox is designed for very small packets of data (12 bits).

THE LAST FEW METERS

With the ratification of Bluetooth v5.0 late in 2016, the Bluetooth SIG took a big step toward supporting more of the IoT. This latest version supports up to double the speed and up to four times the range of previous versions. These enhancements broaden the scope for Bluetooth significantly, and enable many phones and tablets to become a ubiquitous gateway. With mesh capabilities in the works for a future version release, Bluetooth becomes a promising and capable IoT connectivity option. The Wi-Fi alliance isn't keeping quiet either, with the introduction of Wi-Fi HaLow in 2016 the 802.11 crowd now has a technology for long-range, low-power connectivity as well.

As the industry ramps up its focus on IoT connectivity, we now have technologies designed for the IoT, rather than form-fitting existing hardware for our needs. Prices are lower, integration is higher, and the future is bright for the IoT.

WRITTEN BY JOSH MICKOLIO

PRODUCT MANAGER, **DIGI-KEY ELECTRONICS**

Privacy and Consensual IoT

QUICK VIEW

01 Explicit consent for data and interactions is important in all software

02 Strong IoT security is a key enabler of consensual IoT

03 Key capabilities to face IoT challenges and enable consensual IoT include configurability, testability, operability, and releasability

BY **MANUEL PAIS**
DEVOPS CONSULTANT AND EDITOR, **SKELTON THATCHER CONSULTING LTD**

Consensual IoT is the application of the idea of consensual software to the Internet of Things.

Consensual IoT means that all IoT providers need to respect and take all measures in their power to protect users' privacy and safety. They need to explicitly ask for consent when in doubt. They need to care not only about the safety of devices and the physical environments around them but also the safety of personal and location data.

CONSENSUAL SOFTWARE

According to Danielle Leong from GitHub's Community & Safety Team, the term "consensual software" means getting explicit consent from users to interact with them and to disclose their personal data.

Assuming users' consent to expose data or have them unwillingly participate in certain interactions must be actively discouraged not only because of the security risks, but also to ensure users are in control of their interactions with the software. On top of that, data exposure leaves the door wide open to abuse (for example, it used to be possible to add GitHub users to a repository without their explicit permission, so users could end up listed as contributors of repositories with racist names without there having been any action on their side).

Leong stresses the need to have meaningful discussions about consent and security in design phases. Thinking about user wants, requesting explicit permissions (opt-in by default, not opt-out) to interact with them, and keeping audit trails are all crucial for building trust and safety in the software.

Trust will sustain the exponential growth of IoT. But fixing today's frequent hacking and manipulation of devices won't be enough. **We need to aim for consensual IoT to (re)gain users' trust.**

Specifically, trust in IoT requires safe device data storage, secure data transmission, secure data center data storage, and controlled access to devices and third-party integrations.

WHY IoT SECURITY IS NOT THERE YET

IoT security is difficult to implement due to the amount of factors to address. IoT backends require the same security as any web system, but we also need to care about the physical security of devices and the security of the networks they are connected to. However, the technical challenges are generally not hard to solve.

Although we might be tempted to point out negligence, the real reason why those challenges are not addressed is the lack of a coherent business model, as highlighted by Alasdair Allan during his QCon London 2017 keynote on "Security War Stories: The Battle for the Internet of Things." **When the financial value for IoT providers resides in selling the device, the incentive to deploy secure software is missing.** Functional and "plug-and-play" are the differentiators today, not configurability and security.

Usage-based IoT business models, on the contrary, provide financial incentives to deliver reliable devices *and* software, effectively leading to a service provider model focused on operability and support.

IoT SPECIFIC CHALLENGES

Just like web or mobile applications, software running in physical devices needs to be consensual, as well. However, there are additional challenges for IoT.

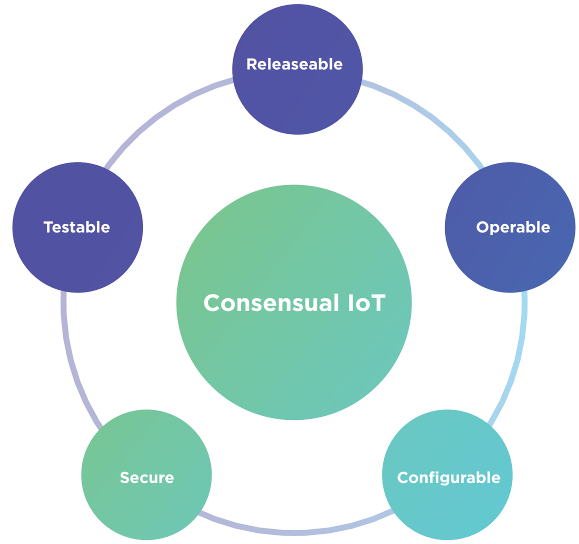
First, **current architectures often treat devices as dumb terminals with little to no configurability built in**. Accompanying mobile apps might help here, but only if they allow configuration of data sharing, along with modes of operation.

The second concern is the lead time to roll out a software update into the device. If we take the GitHub example, requesting explicit consent for adding repository contributors (the "repository invitations" feature) required multiple adjustments to the software: UI changes, email notifications, database changes to keep state, etc. What seems like a simple update often hides complexity and requires changes to multiple layers. The level of complexity will depend on how modular and testable the software architecture is. Nevertheless, final rollout of the feature probably wasn't that hard or time consuming because we've built the mechanisms for rapid releases. That is not the case for IoT, where **frequently deploying software updates to physical devices is not the norm yet**.

Further complicating matters, **a single vulnerability in the device's software (including OS) can be exploited across a large number of devices carrying that same software**. This has already led to massive exploits with botnets, such as the now famous record-breaking DDoS exploiting Internet-connected security cameras. The same applies to privacy abuse, such as the recent Burger King ad purposely triggering Google Home devices to read out a marketing blurb. Google took three hours to stop the abuse (apparently by updating the device's database of sound clips to ignore), yet there was no inherent issue with the device. Thus, there's a need to think ahead of possible interactions and misuse of the device!

CAPABILITIES REQUIRED FOR CONSENSUAL IoT

Obviously, we need technical solutions for specific security issues. But we might only be fighting fires if we don't grow the capabilities required to develop secure and reliable IoT systems. What are the core capabilities required for consensual IoT?



CONFIGURABILITY

This could fall under Operability below, but **configurability is so crucial for consensual IoT that we need to single it out**. Often, IoT devices need to send data over the network to fulfill a goal, but that does not equal a wild card for all data collected by the device to be sent over at any time.

For example, an energy meter needs to send usage data to the energy provider, but shouldn't send location data by default (if compromised, it could allow an attacker to establish usage patterns and find out when the house is empty). With IoT, we're not only talking about privacy breaches only but physical danger, as well.

Configurable devices need (more) conservative defaults to respect their users. This goes beyond secure defaults. For example, allowing a device to connect to a public network with a generic default password is neither secure nor conservative. Transmitting encrypted location data (if not critical for the service) by default might be secure but still disrespectful.

TESTABILITY

Testing the software in the device in isolation (data is being correctly measured and transmitted) is important, but testability for IoT systems means we can also reproduce the ecosystem around the device and how users and other systems interact with it.

In modern web systems, we aim at production-like environments for testing. We can reproduce an actual live system using the exact same (version-controlled) scripts and tools for provisioning and deploying applications.

We need the same production-like environments for

IoT testing to ensure data is secure and not exposed to attacks and to exercise device configurability, ensuring users have correct and sufficient control over how their data is being used.

The difficulty for IoT is mimicking the connectivity with external actors. In our test labs, we need to go for the lowest security denominator. This means, for example, assuming our device will connect to insecure wireless networks, or that an attacker can connect to the device via Bluetooth or even gain physical access to USB or other ports!

A good test pipeline needs to cover what's "inside" the system border (for example, functional tests of the software running in the device or performance tests on the backend) as well as interactions with what's "outside" our control (e.g. other devices in the same network or third parties that the device is sending data to). We need things like test rigs and digital twins in order to do this. But **above all, we need proactive and continuous IoT testing skills**.

We also need to test for operability, making sure our logging and monitoring works. Beware that this is a never-ending work in progress. As we face new issues in operation, we realize what pieces were missing in our logs and dashboards and amend and re-test them.

OPERABILITY

We live surrounded by complex sociotechnical systems. Forward-thinking organizations acknowledge that they can't possibly pinpoint, much less test every possible failure scenario. That's why they increasingly focus on operability (like centralized logging, event tracing using correlation IDs, and monitoring) of modern systems and actively invest in cultural acceptance of failures as sources of learning (post-mortems, blameless culture, and shift-right testing).

IoT adds even more complexity; thus, we need to apply the same ideas to allow quick diagnosis and recovery from incidents.

We need to actively explore potential failure modes of our software, both standalone and in interaction with other systems. Attackers will explore them if we don't do it first.

In the worst-case scenario, where we fail to prevent data from being leaked or abused, we need to be able to quickly identify which interactions took place before and after the incident, look for deviations in usage patterns, and learn how could we have identified the problem earlier.

In short, minimize exposure as much as possible, but **be prepared for abuse to happen and have the mechanisms in place to identify and diagnose them quickly**.

RELEASABILITY

It's not enough to identify and diagnose incidents. We also need to be able to take swift action. Often, fixing the problem requires a change in the software running on the device.

It might be a one-line change as in the energy meter example (stop sending location data) or it could be a more complex change requiring backend modifications (imagine that the energy provider was tracking customers' consumption based on meter location, not on a customer ID).

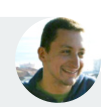
Regardless of the change size, being able to deploy frequently, quickly, and reliably is fundamental for IoT, either to fix a problem or to provide more control to users over their data and device interactions.

Modern mobile and web systems rely on Continuous Delivery techniques (such as deployment pipelines or canary releases) to test changes at multiple stages, from source control to a subset of instances running in production to a full rollout to the entire live environment. If a change fails to produce the expected results in production, it should be straightforward to redeploy the previous good version from an artifact repository.

CONCLUSION

Consensual software should be the norm in today's world, including in IoT. We need strong security in place as a key enabler of consensual IoT. We also need strong capabilities to make our IoT systems configurable, testable, operable, and releasable.

There's a wealth of techniques stemming from the Continuous Delivery and DevOps movements that we can apply when developing and operating IoT systems. Unfortunately, incentives for all the above won't be there until IoT moves to a service-driven business model.



Manuel Pais is a team-first technologist at Skelton Thatcher Consulting. DevOps advocate with a diverse background as developer, build manager, and QA lead. Manuel enjoys helping organizations adopt test automation, Continuous Delivery, and cloud, from both technical and human perspectives. Co-author of the books "Team Topologies" and "Team Guide to Software Releasability." Also InfoQ DevOps lead editor and co-organizer of DevOps Lisbon meetup. Tweets @manupaisable, blogs at skeltonthatcher.com.

An Overview of IoT Monitoring Systems

BY **FRANCESCO AZZOLA**
IT ARCHITECT

QUICK VIEW

01 The Internet of Things will bring great benefits to the manufacturing sector, and the use of sensors can greatly improve the quality of the production process.

02 This project demonstrates how by using an IoT development board and sensors, it is possible to monitor a production process with a few development steps.

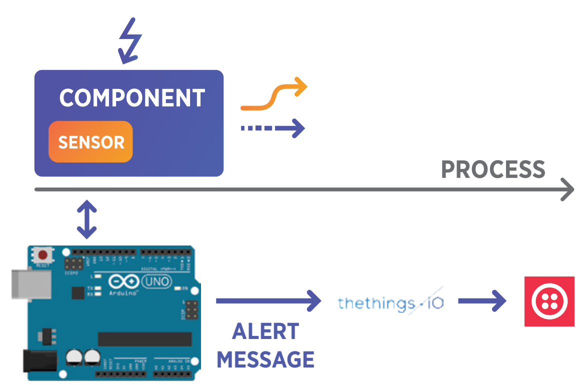
03 We can think about smart production systems that can decide to take corrective actions to avoid damaging the system itself or the component they are building.

The Internet of Things will bring great benefits to the manufacturing industry, and the use of sensors will greatly improve the quality and speed of the production process. Just think about the possibility of smart production systems with the power to make data-informed decisions and take corrective actions to avoid damaging components they are building or even the systems themselves. There are several scenarios where there is the need to monitor a process and send an alert every time a problem occurs during this process. Monitoring systems and sensors will help with solving these problems, and that's exactly what I plan to talk about here. This article assumes some familiarity with Arduino hardware and modules.

For instance, we may want to monitor the alignment of some component during a manufacturing process. Moreover, let's say the IoT system we are building must be able to detect the acceleration or deceleration of this component. These aspects have a great importance in the overall process quality. A scenario where we can apply this monitoring system might include a component that moves from one machine to another during the production process. In this context, we are interested in monitoring its position—or even better, its alignment—and the forces acting on it.

Moreover, what if we want to send an alert as soon as something happens to the production machinery or the component that the machine is working on accelerates or decelerates on its way through a manufacturing process? In order to achieve this, the IoT monitoring system will need to be integrated with an IoT cloud platform so that we can, for example, send a short message to a user's mobile phone.

For the sake of simplicity, let's suppose that a sensor is attached to the component we want to monitor. This sensor detects all the movement and angle or alignment of the component during the manufacturing process. It exchanges data with an Arduino that monitors the data returned by the sensor. If this date hits a certain a threshold, the IoT board sends an alert message using Twilio and TheThings.io. The picture below shows the project overview:



BUILDING THE PROJECT

In order to build this project we need:

- Arduino Uno (or compatible board)
- MPU 6050
- TheThings.io
- Twilio account

MPU 6050 is a 6-axis (Gyro + Accelerometer) sensor with a built-in motion tracking system. In more details, this sensor has a three-axis gyroscope and three-axis accelerometer with a Digital Motion Processor that helps us to precisely control our component. If you want to have more information about this sensor you can refer to this link. To enable the notification system based on a short message, this project uses TheThings.io. This platform provides integration features that allow us to create dashboards.

The IoT monitoring project can be divided into two parts:

1. The first part retrieves data from sensors.
2. The second part integrates the project with Twilio and TheThings.io.

INTEGRATING THE MPU6050 WITH ARDUINO

The MPU6050 sensor is an I2C sensor that manages another pin used as interrupt. The connection schema is trivial and not covered here.

We do not want to only acquire raw data from the 3-axis gyrometer and 3-axis accelerometer. Instead, we want to use the built-in DMP module, which combines data from the two built-in sensors in order to reduce errors when performing complex calculations, so that the movement tracking information retrieved is more precise. In order to use the sensor with Arduino, we have to import the Wire and MPU 6050 libraries. You can download these libraries at this link.

Once we have imported these in our project, we can start developing the code, where the *ARDUINO_INTERRUPT_PIN* is 2:

```
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#include "Wire.h"
void setup() {
  Serial.begin(115200);
  Wire.begin();
  Wire.setClock(400000);
  device.initialize();
  pinMode(ARDUINO_INTERRUPT_PIN, INPUT);
  deviceStatus = device.dmpInitialize();
  // Set here the offset
  if (deviceStatus == 0) {
    Serial.println("Device OK...");
    device.setDMPEnabled(true);
    attachInterrupt(digitalPinToInterrupt(ARDUINO_INTERRUPT_
PIN), deviceDataReady,
                              RISING);
    deviceStatusCode = device.getIntStatus();

    packetSize = device.dmpGetFIFOPacketSize();
    Serial.println("Packet Size: " + String(packetSize));
  }
}
```

Moreover, we attach the interrupt to the PIN 2 calling the function *deviceDataReady*. We use this approach because the MPU6050 triggers an interrupt whenever new data is available. The *deviceDataReady* functions set a variable to true value:

```
void deviceDataReady() {
  interruptReady = true;
}
```

Finally, we can implement the loop function:

```
void loop() {
  // We wait for interrupt
  while (!interruptReady && fifoCount < packetSize) {
  }

  interruptReady = false;
  deviceStatusCode = device.getIntStatus();


  fifoCount = device.getFIFOCount();
  if (fifoCount == 1024 || (deviceStatusCode & 0x10) ) {
    device.resetFIFO();
  }
  else if (deviceStatusCode & 0x02) {
   Serial.println("Status 2");
    while (fifoCount < packetSize)
       fifoCount = device.getFIFOCount();

    device.getFIFOBytes(fifoBuffer, packetSize);
    fifoCount -= packetSize;
    Serial.println("Dump data...");
    // Get quaternion data
    device.dmpGetQuaternion(&q, fifoBuffer);
    device.dmpGetGravity(&gravity, &q);
    device.dmpGetEuler(euler, &q);

    // Check the delta

   device.dmpGetQuaternion(&q, fifoBuffer);
   device.dmpGetAccel(&aa, fifoBuffer);
   device.dmpGetGravity(&gravity, &q);
   device.dmpGetLinearAccel(&aaReal, &aa, &gravity);

  // check the delta

 }
```

Here's a step-by-step explanation of the source code:

1. The sketch waits until the interrupt is HIGH and the data in the FIFO queue is the right length. Remember that MPU 6050 uses a FIFO queue that holds the processed sensor data.

2. Once the queue is ready, we can read the data.

3. The data is represented using quaternions. As stated before, we want to measure the acceleration and the rotation angle. To this purpose, the library provides a set of functions to easily calculate these values. An interesting aspect is that we want to filter the gravity component from the result. For example, to calculate the rotation angle we use:

```
device.dmpGetQuaternion(&q, fifoBuffer); // Read data
from the buffer
device.dmpGetGravity(&gravity, &q); // Get the gravity
device.dmpGetEuler(euler, &q); // Calculate the rotation
angle (euler form)
```

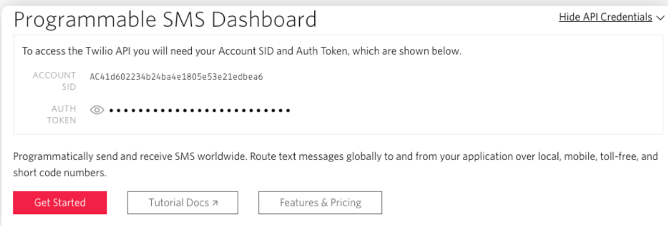
4. To know if we have to trigger the notification event, we have to compare the current result with the previous

values (stored in another variable). If the difference is bigger than the delta, we trigger the notification event.

5. Using the same process, we can calculate the acceleration/deceleration of a component through an assembly line.

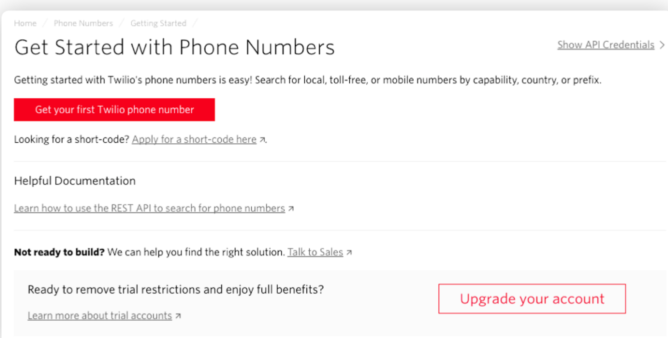
BUILDING THE NOTIFICATION SYSTEM

Once, we have built the system that detects the motion and alignment we can focus on the notification part. As you may already know, Twilio is a platform that sends short messages. The first step is creating an account. Once the account is ready, you can access your dashboard and get information about your API credentials:

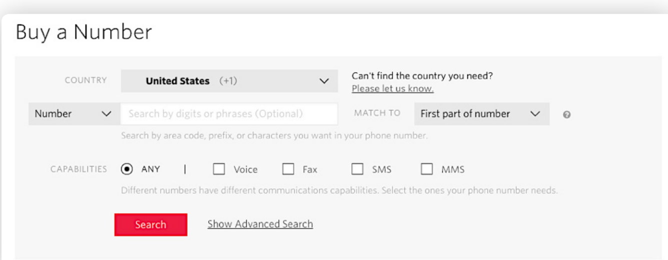


Your account **SID** and your **AUTH Token** is what you should be looking for because we will use them with TheThings.io. Before sending a short message, it is important that you create a virtual phone number which supports SMS features.

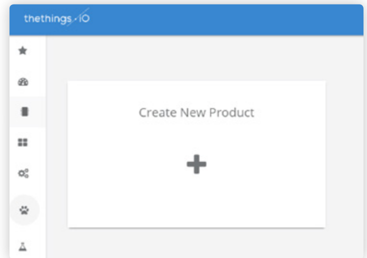
Here is how you can get a sender number:



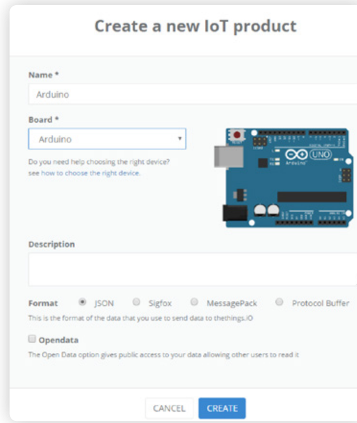
You can configure your phone number here:



NOTE: for this example, we only used the free account to avoid paying any subscription fees. Now we have all the information needed to configure TheThings.io. The first step is creating a new product:

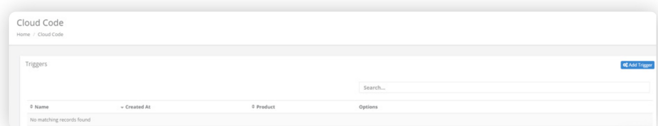


Next:



Once you have configured your sensor you have to activate it in order to send data. TheThings.io provides a simple sketch in order to activate your board. Remember to install the TheThings.io library before running the sketch.

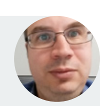
The next step is creating a trigger:



A trigger monitors a variable that our IoT board sends to the TheThings.io platform. When the value of this variable goes over a certain threshold, the platform runs a script. In this context, when the deltas are over the threshold we have defined, we send a variable with a fixed value to the TheThings.io so that the trigger is fired. To implement this trigger, you can refer to this tutorial that describes step-by-step how to invoke Twilio.

Upon defining your threshold values for the delta acceleration and the delta angles, you can test the sketch. The delta is the difference between the two consecutive values. Run the sketch and then rotate the sensor. You will notice that you will receive a short message as soon as the delta retrieved from the sensor is over the threshold you have defined.

Francesco Azzola is an electronic engineer with over 15 years of experience in computer programming. He is SCEA, SCWCD, and SCJP certified. He loves creating IoT projects using Arduino, Raspberry Pi, Android, and other platforms. He is interested on the convergence between IoT and the mobile applications. You can read more from Francesco on Android and IoT at survivingwithandroid.com or on DZone.



IoT Strengthening Ubiquitous Computing

BY **PRADEEP PEIRIS**
SR. SOFTWARE ENGINEER, **DIGITAL RIVER WORLD PAYMENTS**

QUICK VIEW

01 Ubiquitous Computing considers taking computer capabilities on actual physical things and encouraging natural human behavior towards the physical world.

02 IoT applications today follow the idea of virtualizing the devices on the Cloud, which is not scalable for the expected growth of IoT in upcoming years.

03 The requirements for the next generation of IoT and Ubiquitous Computing are the need for standard middleware and application protocols.

04 An IoT platform with peer-to- peer middleware protocols and application protocols as multi-agents would help achieve the concept of Ubiquitous Computing.

In the last two decades, the advancement of computation technology has produced more powerful processors, storage, memory, and networking devices for low prices. Eventually, many physical things will be embedded with more computation capabilities and interconnected via the Internet in a cost-effective manner. The widespread adoption of IoT technologies enriches the idea of ubiquitous computing that Mark Weiser came up with in 1988 — that is, bringing computation into the physical world and making effective use of computer technologies.

UBIQUITOUS COMPUTING

Mark Weiser defined ubiquitous computing as "the method of enhancing computer use by making many computers available throughout the physical environment but making them effectively invisible to the user." This definition highlights the following main properties:

- **Pervasive computers:** The technology that makes computation capabilities available throughout our physical environment.
- **Invisible computers:** Ubiquitous computing enables many computation capabilities throughout the physical environment, but makes them invisible to the user.
- **Enhanced computers:** The objective of ubiquitous computing is to enhance computer usage with the following characteristics:
  - **Expand human consciousness:** Many physical things can be embedded with computation features that expand users' consciousness. For example, smart homes automatically control electricity. Refrigerators inform you if an item is out of stock.
  - **Make context-aware and responsive environments:** Ubiquitous computing enhances physical things with a new dimension of features without affecting the context of the actual usage. It enables other interconnected things to know the user's current context so that the devices respond in a way that improves the overall activities of the user.

Altogether, ubiquitous computing improves the performance of human activities by moving technology into the background, which allows users to focus only on their natural activities towards the world. The technology drives users' tasks in an optimized manner in the background.

ANALYSIS OF EXISTING IoT TECHNOLOGY

There are many research and commercial IoT products available today; for example, smart wearables, homes, cities, factories, etc. A few common factors are apparent in existing IoT technology. A lack of standardization is a commonly seen concern in most of the ecosystem that limits collaboration and interoperability between IoT applications. For example, smart electrical appliances together can build better smart homes. In the same way, smart homes could be made in a standardized way, helping build up to better smart cities.

Another common factor is the engagement of similar architecture patterns. IoT frameworks today follow the idea of virtualizing devices on the cloud and provide a set of services over mobile and web applications. As seen in the following diagram, these applications are not autonomous and interoperable with each other, nor is any collaboration possible. Also, more IoT devices are being released every day. The cloud will not be the best place to virtualize all these IoT devices.

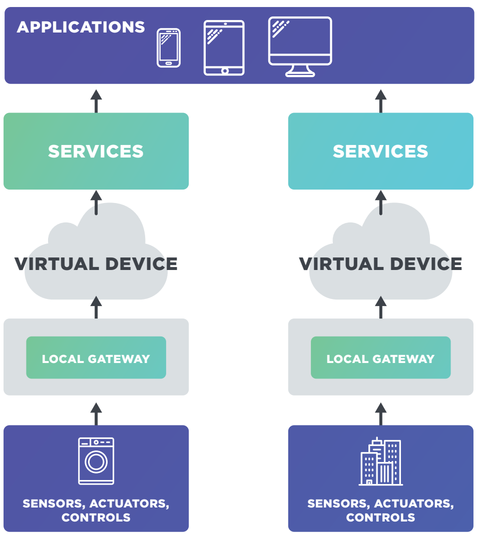


Figure 1. Common IoT architecture pattern

REQUIREMENTS FOR NEXT-GENERATION IoT

The requirements for the next generation of IoT that enrich the characteristics of ubiquitous computing can be classified as the need for standard middleware and application protocols. The **middleware protocol** should abstract the underlying network complexity of heterogeneous devices and provide a simplified API for the **application protocol** to develop autonomous applications that lead users toward a natural way of interacting with the physical environment.

STANDARD MIDDLEWARE PROTOCOL

The number of IoT devices is expected to grow exponentially in upcoming years. A standard middleware protocol should support management of large-scale IoT and the overwhelming influx of data. Also, there will be new, emerging security risks. The middleware protocol should guarantee basic security and privacy principles for IoT. In addition, the middleware protocol should encourage the general community to contribute more IoT devices to the network.

STANDARD APPLICATION PROTOCOL

The application protocol should direct the development of autonomous applications, which encourage a natural way of interacting with physical things in a context-aware and responsive environment. Also, the applications should be capable of reading large-scale datasets, which analyze and confirm users' goals and preferences.

The applications will be contributed to and developed by many individuals with different goals. Therefore, the applications should be able to communicate and cooperate with each other without causing any conflict. As all these applications run with the same middleware and protocols, the applications themselves should be able to negotiate and reach an agreement with each other when accessing the shared set of IoT devices.

PLATFORM FOR NEXT GENERATION IoT

The requirements for a standard middleware and application layer can be summarized as follows:

MIDDLEWARE PROTOCOL	APPLICATION PROTOCOL
• Management of heterogeneous devices • Scalability over IoT • Privacy and security • Network churn • Mobility • Encourage utility	• Autonomous • Reasoning • Communication • Collaboration • Cooperation

Fortunately, the requirements for a standard middleware protocol are already part of the research field of peer-to-peer computing and have been producing various successful protocols during that time. In the same way, the requirements for a standard application protocol have been already improved and standardized in the field of multi-agent systems.

PEER-TO-PEER COMPUTING

A peer-to-peer (P2P) system is about the sharing of computer resources in a decentralized manner. In P2P systems, each host directly connects to each other over the network, forming virtuality for resource sharing.

P2P systems first became popular in 2000 with Napster, a community-driven file sharing system. BitTorrent and Blockchain are other successful P2P systems today. All these systems address the issues of network complexity, scalability over a broad set of communities, intensive data load, and network churn while encouraging peer resources as the utility.

MULTI-AGENT SYSTEMS

Michael Wooldridge defines an agent as "a computer system that is situated in some environment, and is capable of autonomous action in this environment in order to meet its delegated objectives." An agent delegates its objectives toward the environment, and becomes intelligent with the following properties:

- **Reactivity:** Intelligent agents are able to perceive their environment and respond in a timely fashion to the changes that occur in it order to satisfy their design objectives.
- **Proactiveness:** Intelligent agents are able to exhibit goal-directed behavior by taking the initiative in order to satisfy their design objectives.
- **Social ability:** Intelligent agents are capable of interacting with other agents in order to satisfy their design objectives.

A multi-agent system arises when multiple intelligent agents cooperate in a single environment. Like humans, intelligent agents are expected to communicate with each other and reach agreements in achieving their design objectives.

Agents can be designed to perceive information from sensors and to decide actions internally to take toward effectors or actuators. With a P2P middleware protocol, the agents are no longer limited to particular sensors or actuators. Figure 2 depicts how an agent's view can be adapted to modern IoT applications.

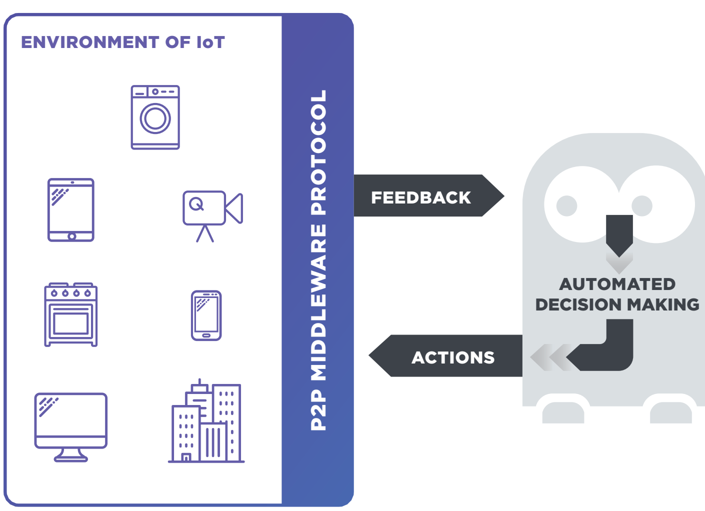


Figure 2. Agents are open for many sensors or actuators over the P2P middleware protocol

SUMMARY

The following figure summarizes all the key aspects discussed previously for having a standard IoT platform.

As it presents, the environment will consist of many IoT devices that are supposed to grow rapidly over the next few years. A standard P2P middleware protocol will provide a reliable distributed system model, which is intrinsically scalable over the expected growth of IoT.

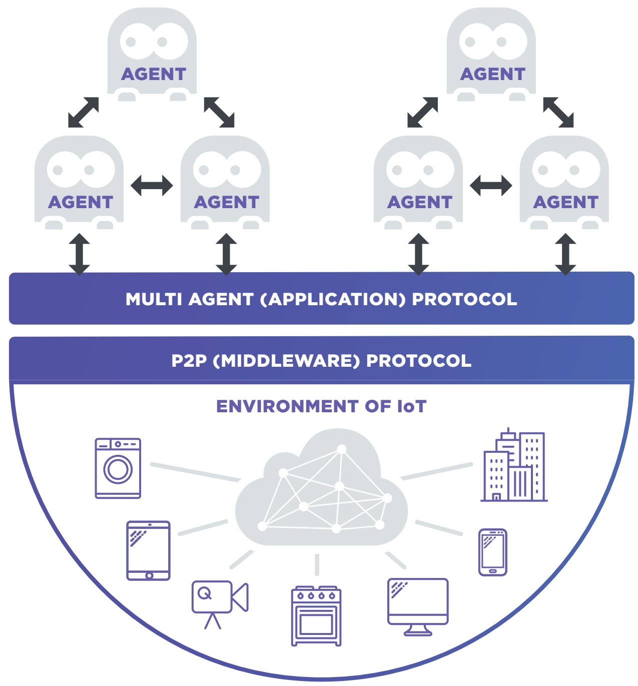
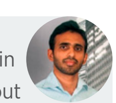


Figure 3. Overview of a next-level IoT platform

An application protocol based on a multi-agent system will allow for the implementation of autonomous applications capable of communication, cooperation, and negotiation with each other. Unlike general applications, agents are designed with goals to be fulfilled on behalf of their users — that is, agents will take necessary actions efficiently towards its environment over a P2P protocol. With a P2P middleware protocol, there are a lot of possibilities to understand the user's context accurately. The agents will use all this information to make context-aware actions for users.

In this way, we can develop applications (agents) that perceive sensor data from one vendor device and make actions toward actuators from another vendor. The new platform opens many possibilities to produce novel (multi-agent) applications that enrich the idea of ubiquitous computing.



Pradeep Peiris is an experienced Software Engineer/Architect in diverse technologies and business domains. He is passionate about learning new concepts and technologies. Currently, Pradeep resides in Stockholm and works for Digital River World Payments as Senior Software Engineer.

Diving Deeper

INTO INTERNET OF THINGS

TOP #IoT TWITTER FEEDS

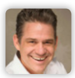
 @AmyxIoT

 @DavidOro

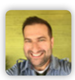
 @adigaskell

 @lornagoulden

 @Doug_Laney

 @JoeSpeeds

 @ToriMcClellan

 @theRab

 @damian_pike

@IoTchannel

TOP IoT REFCARDZ

Getting Started with the Industrial Internet
dzone.com/refcardz/getting-started-with-industrial-internet
Introduces basic concepts and technologies of the Industrial Internet, including sensors and actuators, industrial control systems, human-machine interfaces, real-time streaming data, device security, and more.

IoT Applications with Java and Raspberry Pi
dzone.com/refcardz/iot-applications-with-java-and-raspberry-pi
Covers hardware setup, Raspbian installation, and general-purpose I/O (GPIO) via two Java libraries. So where will your next project take you – a self-driving car, a portable gaming console, or...?

Getting Started With MQTT: A Protocol for the IoT
dzone.com/refcardz/getting-started-with-mqtt
HTTP is too heavyweight for IoT devices, and request-response isn't responsive enough. Learn MQTT instead. Refcard includes message types, QoS levels, client libraries, and security.

IoT ZONES

IoT 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.

Big Data dzone.com/bigdata
The Big Data/Analytics Zone is a prime resource and community for Big Data professionals of all types. We're on top of all the best tips and news for Hadoop, R, and data visualization technologies. Not only that, but we also give you advice from data science experts on how to understand and present that data.

Cloud 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. The Cloud Zone focuses on PaaS, infrastructures, security, scalability, and hosting servers.

IoT WEBSITES

IoT Council theinternetofthings.eu

Postscapes postscapes.com

DataTau theinternetofallthings.com

IoT RESEARCH

IoT for Smart Cities ieeexplore.ieee.org/document/6740844

IoT-Based Big Data Storage Systems in Cloud Computing ieeexplore.ieee.org/document/7600359

An IoT-Aware Architecture for Smart Health Systems ieeexplore.ieee.org/document/7070665

An IoT Tinkerer's *Tips of the Trade*

To create any application or system, you need a wide breadth of knowledge across several disciplines. Building an effective and useful IoT system is no different. Learning how to develop applications or prototype devices is easy, compared to the challenge of learning how to manage the logistics of collecting and analyzing data, as well as keeping your devices connected, secured, and powered on. We had a chat with two of our Zone Leaders to pass along some wisdom and best practices for data collection and analysis, reducing power consumption, device and data security, and networking and reliability. Housed below are some helpful tips to remember as you tinker with your IoT system.

POWER CONSUMPTION

1. Keep library modules custom code lean because of constrained memory disks and CPU
2. Uninstall anything you don't use in the OS
3. Use low-level languages like C/C++ for embedded devices
4. Adopt low-power communication protocols such as WiFi Direct, Bluetooth LE, and NFC
5. Apply a scheduler to limit the number of calls you make to third-party services

SECURITY

1. Encrypt and authenticate all communications
2. Don't use your own protocols
3. Use the most recent versions of libraries and supporting software you can (including Operating Systems, if necessary)
4. Isolate privileged operations via judicious process use, if using embedded Linux
5. Minimize the functionality on the device to as little as needed, hardening everything that's left

NETWORK AVAILABILITY + RELIABILITY

1. Ensure you have the ability to update the software on your devices
2. Keep a small testing suite on the device
3. Take advantage of process management to manage and monitor services, if using embedded Linux
4. Assume failure, not success, when using network communication
5. Don't block network communications

DATA COLLECTION/ ANALYSIS

1. Use MiNiFi to collect data as soon as it's generated
2. Keep your data as compact as possible; use short names if you're sending JSON
3. Adopt stream processing to quickly filter and process incoming data from devices ASAP
4. Understand why you need the data you're collecting and the most valuable way to use it
5. Build basic analytic capabilities into your device before implementing predictive analytics

Context is Key to Next-Gen IoT

BY **YOUVAL VAKNIN**

HEAD OF DEVELOPERS' PROGRAM AT **NEURA**

QUICK VIEW

01 IoT is everywhere, and you can use whatever is connected to you even if it's not the device's main purpose.

02 Use data flows wisely. Don't get caught in the so called "smart" corner where the flip of a switch has been replaced with a button on your device.

03 Understanding your users' routine can improve any interaction you have with them.

04 You don't need huge infrastructures from the get go, baby steps can be taken even for connected devices.

When mentioning connected devices these days, we immediately think about the cutting edge. A refrigerator that can order more eggs when you've just run out, or a speaker that gets pizza to your door by just asking it to. The more common ones are the smart watches, activity trackers, smart light bulbs, and so on.

The western world is literally swarmed by connected devices, but the most important one is the smartphone. Packed with state of the art sensors, each one of them benefits from the always-on connectivity of their host device. But that's not all, your car's Bluetooth entertainment system is a connected device, and every single Wi-Fi router you are connected to is a device that can indicate something about your daily routine.

IoT can be described as a data flow. For example, the data flow is from a controller (mostly a smartphone, but can be a sensor like a volume sensor) to a device (light bulb) is "smart" only because it can respond to commands sent through the network instead of an analog switch. Another example is the movement from an activity tracker sensor to your phone, where the "smart" part is the way this data is displayed on the dedicated app for that tracker.

Providing these bare minimum data flows is not what should be considered "smart", they are just new interfaces which provide just a little less friction for the underactive among us. Combining data flows and making sure they are interconnected will create a true smart environment around the center of it all: humans.

The base of this human-centered data flow is a network of sensors, but not all sensors were born the same. Let's compare single- and multi-channel data sensors. Single-channel sensors provide one data input. For example, a thermometer provides the temperature at a point in time; steps counter – cumulative steps in a time range; Glucometer – a person's sugar level at the time of measurement. The smartphone is a multi-channel device – it has dedicated motion sensors like a gyroscope, and inferred motion sensors like the GPS receiver and the cellular tower's reception level. It has location sensors — a dedicated one (GPS again) and an inferred one, the Wi-Fi antenna. The phone is also equipped with sensors that can detect its surroundings like light sensors, the microphone, camera, and sometimes even a barometer.

This differentiation is key when analyzing data and using it for various IoT-dependent products, because when you're fighting for your users' attention, context is king. Approaching them at the right time with the right messaging can change the way they relate to your product.

The smartphone, an always-on, always-connected, and almost always-carried device is the perfect choice for a daily routine descriptor. It goes wherever its owner goes during the day. Accumulating and aggregating the data from it would reveal amazing insights about the owner's daily routine.

Looking deeper into the data and finding simple patterns can help realize more complex moments, such as when the user sleeps, when a device is idling for a long period of time, connected to a power supply, and is kept static, and when the user wakes up, the first time the device moves after being idle during the night (relative to the device's time zone).

The next step would be to try and predict the user's behavior. If I can identify when the person is asleep every night, I can try and see if something else is happening around that time, such as lights being turned off or an alarm system turned on. Now I know when the user is planning to go to bed. These tiny moments during the day are summed up in an actual image of someone's daily routine. That image that influences the way we can analyze every new piece of data coming our way. All of this is directly connected to where the user is, which can be discovered by GPS and WiFi routers. An awesome example of a routine of seeing the same SSIDs all the time. For instance, my SSID at home is "It's a trap!". I'm always connected to it when I'm there and it can be quickly identified as my home network. This is easy to see, in part, when looking at how many times I've connected to that network, for what length of time, as well as how long it took for the device to connect.

While the smartphone helps us focus on the daily routine, it's time to put single data channels to good use. For example, a connected thermometer indicates an elevated temperature of 38 degrees C/100F, which might translate to a health condition, but the daily routine of the person would show that they were outside all day, walking and running, and it was a hot day according to the weather conditions in that location. Instead of prompting the user to make a doctor's appointment, the wellness app would suggest moving away from the sun and drinking some water. Messaging is provided with context to the real situation of the user.

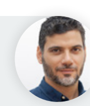
And it can get more interesting. Connected devices can improve otherwise limited lifestyles and help make them as active as they can be. People with type 1 diabetes are constantly struggling with their sugar levels, from the discomfort of the hyperglycemia to the life-threatening hypoglycemia. When a person with diabetes finishes an exercise, he or she is probably in some level of hypo, and driving in that state might be dangerous. By understanding the current situation, with their fitness device connected to an Android Automobile, the diabetes app provides immediate advice to eat some fruit and make sure their glucose levels are suitable for the needed concentration levels for that activity.

Implementations are not only medical. What is a "Smart Home"? It should be a home where connected appliances are reacting to you. The smartphone identified that you arrived home? Turn the lights on, open the blinds, let the connected speaker let you know that the water heater made sure you can take a shower now. Your surroundings respond to you with minimal activity. All of this can be achieved by making sure your current "smart" devices are switched on by an event, in this case arriving home, rather than a finger tap.

In the world of content, demographics are the center of everything. Understanding the person's location (Geography), age, and gender are considered keys to delivering the best conversion rate for ads and making sure content will get the most exposure possible. Google (based on searches) and Facebook (based on interactions and likes) are known by using these attributes for better segmentation. But at the end of March, Netflix's VP of product said "Geography, age, and gender? We put that in the garbage heap". Instead, they decided to concentrate on taste alone, using the thumbs up and down ratings for each show. It worked very well as a suggestion algorithm but it lacks context, which in this case is "What will I watch today?". Some viewers will prefer some lightheaded short episodes after a busy day, while others might rather watch a food documentary if their day finished early. All based on their general preferences, but with relation to their daily routine.

Context is the invisible ingredient that turns clumsy interactions to frictionless behavior. And with even a superficial interpretation of data channels around the users, their experience will improve.

I avoided the term "Machine Learning" throughout this article to show that even a huge subject such as that can be approached with baby steps and lean thinking. By utilizing Wi-Fi, which is everywhere, GPS data, and motion sensors that are available on every smartphone out there, data can be input to a simple condition clause that can later grow into more complex decision trees and pattern matching. Eventually, to make the most out of the data provided, complex algorithms would be the next logical step. Huge leaps can be achieved by using the basic input from the data around us to actual actions right here, right now.



After more than 15 years of software development experience ranging from image processing using MATLAB and C++, security and authentication, digital health, and iOS development, **Youval Vaknin** is focused on making Neura's mobile SDK for AI-as-a-service a streamlined product for developers. As Head of Developers' Program, Youval engages developers, creates tools and samples for them to use, and learns what their limitations are when using this product.

IoT Applications Present New and Challenging Requirements in Data Management

Every day, machines, containers, measurement devices, vehicles, and even people are being equipped with internet-connected sensors that transmit information, receive instructions, and even act based on the information they receive. Today, there are more than nine billion connected devices around the world, and that figure is rapidly growing. McKinsey predicts the total economic impact of Internet of Things (IoT) will reach between \$3.9 trillion and \$11.1 trillion per year by the year 2025. The potential for organizations that can effectively capitalize on IoT opportunities is massive.

Most traditional data management technologies were not built to handle the unique requirements of IoT applications. IoT applications must reliably ingest, process, and persist large volumes of data that devices and sensors continuously generate at very high rates, analyze vast sets of dissimilar data, and perform intelligent programmatic actions in real-time.

IoT applications require a different kind of data platform, one that can accommodate the challenging requirements, high throughput, and scale associated with IoT applications.

InterSystems Data Platform is different. It is a complete platform for developing, executing, and maintaining IoT applications in a single, consistent environment leveraging a distributed architecture to support massive data ingest rates and data volumes. It provides a proven, enterprise grade transactional-analytic multi-model database to ingest, process, and persist data from a wide range of devices in different formats at high ingest rates, the ability to process high transactional and analytic workloads concurrently in the same database, a complete set of integration, event processing and integrated analytics capabilities, and business process orchestration capabilities. A technology like InterSystems Data Platform can provide a capable and proven platform for companies that are implementing IoT applications in a range of industries.

WRITTEN BY JULIE LOCKNER

DIRECTOR, PRODUCT MARKETING AND PARTNER PROGRAMS, **INTERSYSTEMS**

PARTNER SPOTLIGHT

InterSystems Data Platform By Intersystems

For more than 35 years, InterSystems has been the engine behind the world's most important applications.

NEW RELEASES
Major release annually, 2 minor releases annually

OPEN SOURCE
No

CASE STUDY
The European Space Agency (ESA) launched an ambitious mission to chart a three-dimensional map of the Milky Way. Because an enormous amount of data will need to be quickly stored and analyzed, ESA has selected Caché as the advanced database technology to support the scientific processing of the Gaia mission.

Gaia will spend five years monitoring a billion stars in our galaxy. In the course of its operation, AGIS must be able to insert up to 50 billion Java objects into a database within seven days. Caché is the only database ESA found that could provide the necessary performance and scalability with only moderate hardware requirements. The information Gaia collects will allow scientists to learn a great deal about the origin, structure, and evolutionary history of our galaxy.

PRODUCT
InterSystem Data Platform is the industry's only multi-workload, multi-model NoSQL and relational database with embedded data and application interoperability with built-in structured and unstructured data analytics and a comprehensive rapid application development environment, without sacrificing high performance, scalability, reliability, and security.

STRENGTH
Complete data platform for reliable, complex IoT applications

NOTABLE CUSTOMERS
- Ontario Systems
- MFS
- TD Ameritrade
- ESA
- Netsmart
- WS Trends

WEBSITE www.intersystems.com

TWITTER @InterSystems

BLOG intersystems.com/intersystems-blog/data-matters

Strengths and Weaknesses of IoT Communication Patterns

BY **PAOLO PATIERNO**

SENIOR SOFTWARE ENGINEER, **RED HAT**

QUICK VIEW

01 IoT has its pillars in messaging because it's all about communication between devices and external/cloud systems exchanging messages.

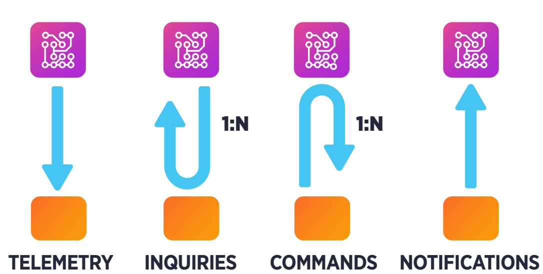
02 IoT communication patterns are available thanks to the corresponding main messaging patterns.

03 The mainly used protocols implement IoT communication patterns in different ways, each one with its strengths and weaknesses

04 Different protocols can be used in the same IoT solution for implementing different patterns they fit well.

The IoT world is all about communication between devices, gateways, and the Cloud; messages are exchanged between all of these parties in order to provide a comprehensive end-to-end solution. It's common to consider the following as the main communication patterns, which are different in terms of how messages flow between parties and the objectives they have:

- **Telemetry**: Data flows in one direction from the device to other systems for conveying status changes in the device itself (i.e. sensors reading, ...)
- **Inquiries**: Requests from the device looking to gather required information or asking to initiate activities
- **Commands**: Commands from other systems sent to a device (or a group of devices) to perform specific activities expecting a result from the command execution, or at least a status for that
- **Notifications**: Information flows in one direction from other systems to a device (or a group of devices) for conveying status changes



Every pattern could have persistence needs, so they'll be using a **store and forward mechanism** (i.e. using queues, topic/subscriptions in a broker) or relying on **direct messaging**, where the receiver needs to be online in order to allow the devices to send data. The former is mainly used by the Command pattern, because the device might not be online at the time the command is sent; sometimes a TTL (Time To Live) on the command message is useful in order to avoid the possibility that an offline device will execute an "old" message that is not useful at the time the device comes back online. The direct messaging mechanism is often used for Telemetry even if, in some use cases, storing telemetry data could be useful as well.

Sometimes the Telemetry pattern can evolve into the **Event** pattern. The main differences are:

- **Better QoS** (Quality of Service) in terms of delivery. For Telemetry, the "most at once" delivery is enough because even if one data value is lost, a new, updated value will be sent in a very short time. For Event, an "at least once" delivery is needed considering, for example, alarms. In such cases, of course, the destination system should be able to handle the "idempotent" nature of the action related to the received message.
- **Message Persistence** in order to avoid losing messages if the destination system isn't working properly or isn't online when the message arrives.

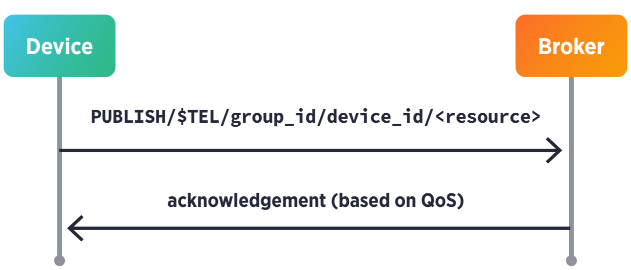
The main protocols used in IoT are really different in the way they implement the patterns above. IoT is more about messaging, which already defines its own patterns, like **competing consumers, request/reply, pub/sub**, and so on. Protocols provide such patterns in different ways (natively or not).

TELEMETRY

The HTTP protocol can implement this pattern in two ways: acting as a "client" and sending a PUT/POST request on the resource to update (i.e. representative of a sensor value), or acting as a "server", which receives GET requests from other systems for getting data. In any case, it's all about request/reply. The drawbacks are that HTTP is really verbose because it is "text-based", and it doesn't provide QoS (only based on TCP).

If working in "server" mode, HTTP has networking problems for connecting with the device when there is a NAT or roaming (mobile networks).

The MQTT protocol was born for telemetry, and it natively implements the publish/subscribe pattern. The device acts as a "publisher" for publishing data to a "topic". On the other side, the system acts as a "subscriber", so it's subscribed to that topic for getting messages. MQTT provides all the well-known QoS levels, but no flow control; a broker can be flooded by messages without the possibility of stopping devices from doing that.



The AMQP protocol is well-known because it provides both request/reply and publish/subscribe natively, so it fits great for Telemetry as well having the device send messages to a destination. The big advantage is related to the built-in flow control at two different levels: in terms of bytes exchanged and in terms of messages. In the latter case, there is a "credit-based" mechanism; a sender needs "credits" from the receiver in order to be able to send messages. AMQP provides all QoS levels as well.

INQUIRY

Implementing this pattern with HTTP is pretty simple thanks to the request/reply nature of this protocol. It's all about a GET request from a device to a server resource for getting information. The drawback is that all the mechanisms work in a synchronous way.

Inquiry is "more" difficult to implement with MQTT because it needs new semantics at the application level. The device subscribes to a "dedicated" topic (i.e. with a request-id information on it) in order to receive the reply. To send such a request, it has to be a publisher on the inquiry topic putting the request-id inside the published message payload in some way. The request-id is needed by the server to connect to the topic for which a message will be publishing the response. There is the need to create a correlation between request and reply at the application level.

IoT is more about messaging, which already defines its own patterns, like competing consumers, request/reply, pub/sub, and so on. Protocols provide such patterns in different ways (natively or not).

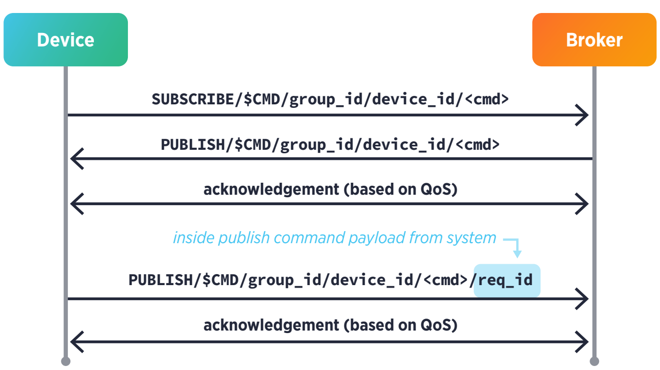
The AMQP protocol fits very well in this case, because an AMQP message provides some useful system properties (as metadata). First of all, every message has an identifier (a message-id) and a reply-to property as well; it means that the devices send an inquiry telling the server where (the address) it wants to receive the response. The server will send the reply setting a correlation-id property (using the message-id of the request). The correlation mechanism is provided for free at the protocol level. Another strength is its asynchronous nature; replies can be provided by the server in any order (if needed) without losing the correlation with the related requests.

COMMAND

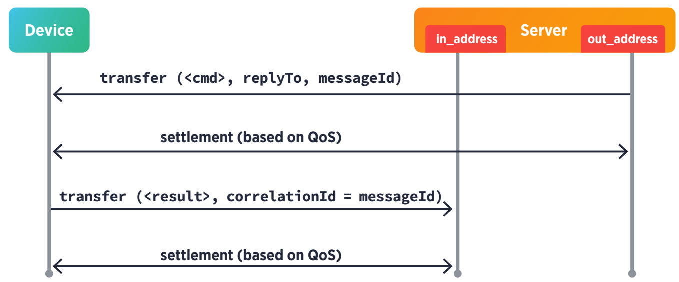
This time, it's not the device that starts the communication, as in the previous patterns, but the external system; it means that the device should be reachable in some other way.

With HTTP, it can be implemented by having the device act as a "server" or "client". If it acts as a "server", the external system sends a POST request, waiting synchronously (long polling) for the result through the related HTTP response. The other possibility is polling the device with subsequent GET requests for getting the command result. In any case, there is always the problem of reaching a device behind NAT or roaming on mobile networks. If it acts as "client", the device sends a GET request to the system, asking if there is a command to execute; the reply can be received through long polling, so that the system doesn't send an HTTP response until a command is available, or with continuous polling. When the command is executed, the device sends another POST request to provide the command result to the server.

The implementation with MQTT is quite similar to the Inquiry. There is no support for request/reply, so it has to be built at the application level using an additional semantic on topics. The device subscribes to a topic for receiving the command, and the server sends such commands as messages, where the payload should contain a sort of request-id. After executing the command, the device sends the result publishing a message to a topic with the above request-id in the name. Even in this case, the command/result correlation is built at application level. What happens if the device is offline? Using the "retain" flag, the last command can be delivered when the device comes back online, but commands are queued only if the session feature is used.



AMQP is better than the other protocols from this point of view. The server can send a message command specifying the message-id and the reply-to system properties. After executing the command, the device sends the result as a message with the correlation-id set to the above message-id. This provides a correlation between the command and result; such a message is sent to the reply-to specified address, where the server is waiting for the result. Of course, the exchange happens in a completely asynchronous way; if more commands are not strictly related, the server can send them all together and receive the results asynchronously (eventually out of order). In order to avoid an offline device executing an "old" command when it comes back online, the TTL (Time To Live) property provided in an AMQP message can be used.



NOTIFICATION

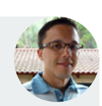
This pattern is quite the opposite of the Telemetry, and the flow is similar to the Command pattern, but without the need for a reply.

With HTTP, the device receives a notification from the external system acting as "server" (sending with a POST request) or acting as a "client" (sending a GET request) on a specific resource. In the latter case, the server can reply only when the notification is ready (long polling) or immediately (with or without a result): It means that the device has to poll the server additional times.

If HTTP is working in "server" mode, addressing problems related to NAT and roaming are still there.

With MQTT, the device subscribes to a specific topic where the server publishes the notifications. This pattern fits very well with this protocol due to its publish/subscribe nature — with the drawback of not having built-in flow control: It means that the device could be overwhelmed by a lot of notifications.

Finally, with AMQP, the device is able to receive the notification pushed by the server as with MQTT, but with the big strength of built-in flow control: The device can stop the server from sending messages if it isn't able to process them.



Paolo Patierno is a Senior Software Engineer working for Red Hat on the messaging and IoT team. At Red Hat, he is focusing on integration between AMQP-based projects and products like Apache Kafka and Apache Spark, aside from bridging other different protocols like MQTT. In the IoT landscape, he is lead/committer for the Eclipse Hono project, which provides an IoT connectivity layer at scale. He is also a committer for the Eclipse Paho project, which provides MQTT client implementations in different languages and for the Eclipse Vert.x project, as a maintainer for some IoT-related components.

The Three Software Stacks Required for IoT Architectures

IAN SKERRETT
VP MARKETING, ECLIPSE FOUNDATION

A typical IoT solution is characterized by many devices (i.e. things) that may use some form of gateway to communicate through a network to an enterprise back-end server that is running an IoT platform, which helps integrate the IoT information into the existing enterprise. The roles of the devices, gateways, and cloud platform are well defined, and each of them provides specific features and functionality required by any robust IoT solution.

Stack for Constrained Devices: Sensors and Actuators

The "Thing" in the IoT is the starting point for an IoT solution. It is typically the originator of the data, and it interacts with the physical world. Things are often very constrained in terms of size or power supply; therefore, they are often programmed using microcontrollers (MCU) that have very limited capabilities.

The key features of the software stack running on a device may include:

1. IoT OPERATING SYSTEM – Many devices will run with 'bare metal,' but some will have embedded or real-time operating systems that are particularly suited for small constrained devices, and that can provide IoT-specific capabilities.

2. HARDWARE ABSTRACTION – A software layer that enables access to the hardware features of the MCU, such as flash memory, GPIOs, serial interfaces, etc.

3. COMMUNICATION SUPPORT – Drivers and protocols allowing the device to connect to a wired or wireless protocol like Bluetooth, Z-Wave, Thread, CAN bus, MQTT, CoAP, etc., and enabling device communication.

4. REMOTE MANAGEMENT – The ability to remotely control the device to upgrade its firmware or monitor its battery level.

Stack for Gateways: Connected and Smart Things

The IoT gateway acts as the aggregation point for a group of sensors and actuators to coordinate the connectivity of these devices to each other and to an external network. An IoT gateway can be a physical piece of hardware or functionality that is incorporated into a larger "Thing" that is connected to the network **The key features of a gateway software stack include:**

1. OPERATING SYSTEM – Typically a general purpose operating system such as Linux.

2. APPLICATION CONTAINER OR RUN-TIME ENVIRONMENT – IoT gateways will often have the ability to run application code and to allow the applications to be dynamically updated. For example, a gateway may have support for Java, Python, or Node.js.

3. COMMUNICATION AND CONNECTIVITY – IoT gateways need to support different connectivity protocols to connect with different devices (e.g. Bluetooth, Wi-Fi, Z-Wave, ZigBee). IoT Gateways also need to connect to different types of networks (e.g. Ethernet, cellular, Wi-Fi, satellite, etc.) and ensure the reliability, security, and confidentiality of the communications.

4. DATA MANAGEMENT AND MESSAGING – Local persistence to support network latency, offline mode, and real-time analytics at the edge, as well as the ability to forward device data in a consistent manner to an IoT Platform.

5. REMOTE MANAGEMENT – The ability to remotely provision, configure, and startup or shutdown gateways, as well as the applications running on the gateways.

Stack for IoT Cloud Platforms

The IoT Cloud Platform represents the software infrastructure and services required to enable an IoT solution. An IoT Cloud Platform typically operates on a cloud infrastructure (e.g. OpenShift, AWS, Microsoft Azure, Cloud Foundry) or inside an enterprise data center and is expected to scale both horizontally, to support the large number of devices connected, as well as vertically, to address the variety of IoT solutions. The IoT Cloud Platform will facilitate the interoperability of the IoT solution with existing enterprise applications and other IoT solutions. The core features of an IoT Cloud Platform include:

1. CONNECTIVITY AND MESSAGE ROUTING – IoT platforms need to be able to interact with very large numbers of devices and gateways using different protocols and data formats, but then normalize it to allow for easy integration into the rest of the enterprise.

2. DEVICE MANAGEMENT AND DEVICE REGISTRY – A central registry to identify the devices or gateways running in an IoT solution, with the ability to provision new software updates and manage the devices.

3. DATA MANAGEMENT AND STORAGE – A scalable data store that supports the volume and variety of IoT data.

4. EVENT MANAGEMENT, ANALYTICS, AND UI – Scalable event processing capabilities, the ability to consolidate and analyze data, and to create reports, graphs, and dashboards.

5. APPLICATION ENABLEMENT – The ability to create reports, graphs, and dashboards, and to use an API for application integration.

Cross-Stack Functionality

Across the different stacks of an IoT solution are a number of features that need to be considered for any IoT architecture, including:

1. SECURITY – Security needs to be implemented from the devices to the cloud. Features such as authentication, encryption, and authorization need be part of each stack.

2. ONTOLOGIES – The format and description of device data are important features to enable data analytics and data interoperability. The ability to define ontologies and metadata across heterogeneous domains is a key area for IoT.

3. DEVELOPMENT TOOLS AND SDKs – IoT Developers will require development tools that support the different hardware and software platforms involved.

Key Characteristics for IoT Stacks

Across the different stacks of an IoT solution are a number of features that need to be considered for any IoT architecture, including:

1. LOOSELY COUPLED – Three IoT stacks have been defined but it is important that each stack can be used independently of the other stacks.

2. MODULAR – Each stack should allow for the features to be sourced from different suppliers.

3. PLATFORM-INDEPENDENT – Each stack should be independent of the host hardware and cloud infrastructure. For instance, the device stack should be available on multiple MCUs and the IoT Cloud Platform should run on different Cloud PaaS.

4. BASED ON OPEN STANDARDS – Communication between the stacks should be based on open standards to ensure interoperability.

5. DEFINED APIS – Each stack should have defined APIs that allow for easy integration with existing applications and integration with other IoT solutions.

Find out how IoT companies are avoiding saying the “C” word by using AI to increase app engagement by over 50%

NEURA

AI for IoT: Are We There Yet?

Like IoT, AI has been co-opted to sell everything from speakers to toothbrushes. But really, what is AI for IoT? Or, a better starting place is to ask "What is it not?"

Despite a lot of hype, connectivity and voice-activated commands are not AI. While both may be super-cool experiences, neither requires any extra intelligence to perform tasks. For example, if you ask your in-home personal assistance device "what's the weather like?", an internet search is initiated. That if-then action has been programmed.

True AI for IoT requires the machine learning that will inform the IoT device. The result is a device that over time learns both your usual practices and can adapt when you don't follow your normal routine.

THE TRANSITION FROM SMART HOME TO "INTELLIGENT HOME"

As we see Smart Homes begin to take hold, it's an excellent time to implement technology to leapfrog to AI-enhanced IoT devices, which for lack of a better name we'll give the moniker "intelligent home." The difference is the ability to learn how the individual or family interacts with products. For example, switching from a programmed or app-activated thermostat to an AI-enhanced thermostat will enable it to "know" the family members are on their way home and thus to turn on, or when to adjust for the family heading to bed.

A more complex example of an intelligent home is a typical bedtime routine. As the user gets in bed, the AI-enhanced IoT products now confirm the front door is locked, adjusts the thermostat, turns on the music/white noise that is preferred for bedtime, and the IoT pillbox reminds the user to take his nighttime meds. Moving from programming based on time to API calls to determine the right moment – in this case when the user goes to bed – will have a significant impact in transitioning smart home IoT devices as cool novelty items to essential lifestyle products.

WRITTEN BY DROR BREN

PRODUCT MARKETING MANAGER, **NEURA**

PARTNER SPOTLIGHT

Neura AI Engine By Neura

NEURA

Neura delivers customer awareness that anticipates users' needs and preferences. Its AI engine increases engagement and retention for some of the world's most cutting edge products.

CATEGORY
Artificial Intelligence for IoT and apps

NEW RELEASES
Two week sprints

OPEN SOURCE
No

CASE STUDY

Through artificial intelligence (AI), Neura equips My Days with user awareness.

Since integrating Neura, My Days has seen reduced churn and increased engagement. Month over month, end-user engagement has risen by 51 percent. In the first 30 days alone with the Neura integration, My Days saw an 18 percent boost in engagement. To assess Neura's impact, My Days divided a sample of users into two groups: 1) users receiving time-based reminders (old method); and 2) users receiving moment-based reminders powered by Neura's personal AI engine. The results showed that users who received moment-based reminders (powered by Neura) engaged with My Days at an 18 percent higher rate than those who received time-based reminders.

STRENGTHS

- Artificial intelligence engine enhances IoT devices and apps to provide personalized experiences that anticipate a user's needs and preferences
- Neura enhanced products are proven to increase engagement and retention
- Machine learning provides deep understand of a user's typical life throughout each day
- The Neura AI Engine incorporates data from more than 80 IoT data sources

WEBSITE www.theneura.com | **TWITTER** @theneura | **BLOG** medium.com/@TheNeura

Executive Insights on IoT Today

QUICK VIEW

01 Success in IoT begins with having a strategy and a use case in mind.

02 To be successful, an IoT initiative needs to be focused on the problem it's trying to solve and how to improve the customer experience.

03 Today, companies in all industries are realizing the value IoT data brings to them, and their end-user customers, to solve real business problems.

BY **TOM SMITH**
RESEARCH ANALYST, **DZONE**

To gather insights on the evolution of IoT at this point in 2017, we spoke to 19 executives who are familiar with the current state of the Internet of Things. **Here's who we spoke with:**

SCOTT HANSON, Founder and CTO, Ambiq Micro
ADAM WRAY, CEO, Basho
PETER COPPOLA, SVP, Product Marketing, Basho
FARNAZ ERFAN, Senior Director, Product Marketing, Birst
SHAHIN PIROOZ, CTO, Data Endure
ANDERS WALLGREN, CTO, Electric Cloud
ERIC FREE, S.V.P. Strategic Growth, Flexera
BRAD BUSH, Partner, Fortium Partners
MARISA SIRES WANG, Vice President of Product, Gigya
TONY PAINE, Kepware Platform President at PTC, Kepware
ERIC MIZELL, Vice President Global Engineering, Kinetica
CRYSTAL VALENTINE, PhD, V.P. Technology Strategy, MapR
JACK NORRIS, S.V.P., Database Strategy and Applications, MapR
PRATIBHA SALWAN, S.V.P. Digital Services Americas, NIIT Technologies
GUY YEHAIV, CEO, Profitect
CEES LINKS, General Manager Wireless Connectivity, Qorvo
PAUL TURNER, CMO, Scality
HARSH UPRETI, Product Marketing Manager, API, SmartBear
RAJEEV KOZHIKKUTTUTHODI, Vice President of Product Management, TIBCO

KEY FINDINGS

01 Several keys to success were recommended for an effective and successful IoT strategy. The most frequently mentioned tips were focused on **having a strategy and use case in mind before starting a project**. Understand what you want to accomplish, what problem you are trying to solve, and what customer needs you are going to fulfill to make their lives simpler and easier. Drive business value by articulating the business challenge you are trying to solve – regardless of the vertical in which you are working.

Architecture and data were the second most frequently mentioned keys to a successful IoT strategy. You must think about the architecture for a Big Data system to be able to collect and ingest data in real-time. Consider the complexity of the IoT ecosystem, which includes back-ends, devices, and mobile apps for your configuration and hardware design. Start with pre-built, pre-defined services and grow your IoT business to a point where you can confidently identify whether building an internal infrastructure is a better long-term investment.

02 Companies can leverage IoT by **focusing on the problem they are trying to solve, including how to improve the customer experience**. Answer the question, "What will IoT help us do differently to generate action, revenue, and profitability?" Successful IoT companies are solving real business problems, getting better results, and finding more problems to solve with IoT.

Companies should also **start small and scale over time** as they find success. One successful project begets another. Put together a journey map and incrementally apply IoT technologies and processes. Remember that the ability to scale wins.

Data collection is important, but you need to **know what you're going to do with the data**. A lot of people collect data and never get back to it, so it becomes expensive to store and goes to waste. You must apply machine learning and analytics to massage and manipulate the data in order to make better informed business decisions more quickly. Sensors will collect more data, and more sophisticated software will perform better data analysis to understand trends, anomalies, and benchmarks, generate a variety of alerts, and identify previously unnoticed patterns.

03 IoT has made significant advancements in the adoption curve over the past year. **Companies are realizing the value IoT data brings for them, and their end-user customers, to solve real business problems**. IoT has moved from being a separate initiative to an integral part of business decision-making to improve efficiency and yield.

There's also **more data, more sources of data, more applications, and more connected devices**. This generates more opportunities for businesses to make and save money, as well as provide an improved customer experience. The smart home is evolving into a consolidated service, as opposed to a collection of siloed connected devices with separate controls and apps.

04 There is not a single set of technical solutions being used to execute an IoT strategy since IoT is being used in a variety of vertical markets with different problems to solve. Each of these verticals and solutions are using different architectures, platforms, and languages based on their needs. However, everyone is in the cloud, be it public or private, and needs a data storage solution.

05 The real-world problems being solved with IoT are expanding exponentially into multiple verticals. The most frequently shared by respondents include: **transportation and logistics, self-driving cars, and energy and utilities**. Following are three examples:

A shipping company is getting visibility into delays in shipping, customs, unloading, and delivery by leveraging open source technologies for smarter contacts (sensors) on both the ship and the 3,500 containers on the ship.

Renault self-driving cars are sending all data back to a corporate scalable data repository so Renault can see everything the car did in every situation to build a smarter and safer driverless car that will result in greater adoption and acceptance.

A semiconductor chip manufacturer is using yield analytics to identify quality issues and root causes of failure, adding tens of millions of dollars to their bottom line every month.

06 The most common issues preventing companies from realizing the benefits of IoT are the lack of a **strategy**, an unwillingness to **"start small,"** and concerns with **security**.

Companies pursue IoT because it's a novelty versus a strategic decision. Everyone should be required to answer four questions: 1) What do we need to know? 2) From whom? 3) How often? 4) Is it being pushed to me? Companies need to identify the data that's needed to drive their business.

Expectations are not realistic and there's a huge capital expenditure. Companies cannot buy large-scale M2M solutions off the shelf. As such, they need to break opportunities into winnable parts. Put a strategy in place. Identify a problem to solve and get started. Crawl, walk, then run.

There's concern around security frameworks in both industrial and consumer settings. Companies need to think through security strategies and practices. Everyone needs to be concerned with security and the value of personally identifiable information (PII).

Deciding which devices or frameworks to use (Apple, Intel, Google, Samsung, etc.) is a daunting task, even for sophisticated engineers. Companies cannot be expected to figure it out. All the major players are using different communication protocols trying to do their own thing rather than collaborating to ensure an interoperable IoT infrastructure.

07 The continued evolution and growth of IoT, to 8.4 billion connected devices by the end of 2017, will be driven by **edge computing, which will handle more data to provide more real-time actionable insights**. Ultimately, everything will be connected as intelligent computing evolves. This is the information revolution, and it will reduce defects and improve the quality of products while improving the customer experience and learning what the customer wants so you will know what to be working on next. Smarter edge event-driven microservices will be tied to blockchain and machine learning platforms; however, blockchain cannot scale to meet the needs of IoT right now.

For IoT to achieve its projected growth, everyone in the space will need to balance security with the user experience and the sanctity of PII. By putting the end-user customer at the center of the use case, companies will have greater success and ROI with their IoT initiatives.

08 All but a couple of respondents mentioned **security as the biggest concern** regarding the state of IoT today. We need to understand the security component of IoT with more devices collecting more data. As more systems communicate with each other and expose data outside, security becomes more important. The DDoS attack against Dyn lat year shows that security is an issue bigger than IoT – it encompasses all aspects of IT, including development, hardware engineering, networking, and data science.

Every level of the organization is responsible for security. There's a due diligence responsibility on the providers. Everywhere data is exposed is the responsibility of engineers and systems integrators. Data privacy is an issue for the owner of the data. They need to use data to know what is being used and what can be deprecated. They need a complete feedback loop to make improvements.

If we don't address the security of IoT devices, we can look for the government to come in and regulate them like they did to make cars include seatbelts and airbags.

09 The key skills developers need to know to be successful working on IoT projects is **understanding the impact of data**, how databases work, and how data applies to the real world to help solve business problems or improve the customer experience. Developers need to understand how to collect data and obtain insights from the data, and be mindful of the challenges of managing and visualizing data.

In addition, stay flexible and keep your mind open since platforms, architectures, and languages are evolving quickly. Collaborate within your organization, with resource providers, and with clients. Be a full-stack developer that knows how to connect APIs. Stay abreast of changes in the industry.

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.

Solutions Directory

This directory of IoT development and device management platforms, prototyping boards, and networking tools provides comprehensive, factual comparisons of data gathered from third-party sources and the tool creators' organizations. Solutions in the directory are selected based on several impartial criteria, including solution maturity, technical innovativeness, relevance, and data availability.

COMPANY NAME	PRODUCT	PRODUCT TYPE	VERTICAL	WEBSITE
Aeries	Aeris	IoT Platform, Connectivity Middleware	Healthcare, Logistics, Transportation, Utilities	aeris.com
Afero	Afero	IoT Platform, Security, Cloud Infrastructure, Connectivity Middleware	Industrial IoT, Wearables, Medical, Transportation, Home Automation	afero.io
Altizon	Datonis	Connectivity Middleware, Device Management, Big Data Analytics	Smart City, Industrial IoT, Utilities	altizon.com/datonis-industrial-internet-of-things-platform
Amazon	Amazon Echo	Consumer Product	Home Automation	amazon.com/echo-superbowl-commercial/b?ie=UTF8&node=9818047011
Amazon	AWS IoT Platform	IoT Platform, Connectivity Middleware, Device Management	Smart City, Transportation, Healthcare	aws.amazon.com/iot-platform
Apache Foundation	MyNewt	IoT Operating System	Device Management, Connectivity	mynewt.apache.org
Apcera	NATS	Messaging Middleware	Connectivity	nats.io
Apple	Apple HomeKit	Developer Program, SDK	Home Automation	developer.apple.com/homekit
Applied Informatics	macchina.io	IoT Platform, Messaging Middleware	App Development, Device Management	macchina.io
Arduino	Arduino Uno	Development board	Prototyping, Hobbyists, DIY	arduino.cc
ARM	mbed IoT Device Platform	IoT Platform, Operating System, Device Management	Prototyping, App Development	mbed.com
Arrayent	Arrayent IoT Cloud Services	IoT Platform, Data Analytics, Device Management	App Development	arrayent.com/iot-cloud-services
Arrow	Dragonboard 410c	Development board	Prototyping, Hobbyists, DIY	arrow.com/en/campaigns/the-dragonboard-is-here
Atmel	Atmel Microcontrollers	Microcontrollers	Hardware	atmel.com/products/microcontrollers
Ayla Networks	Agile IoT Platform	IoT Platform, Messaging Middleware	Home Automation, Wearables, Logistics	aylanetworks.com/products
Ayyeka	Ayyeka Wavelets	Sensors	Utilities, Smart City, Industrial IoT	ayyeka.com
Beagleboard.org	BeagleBone Black	Development board	Prototyping, Hobbyists, DIY	beagleboard.org
BestMile	BestMile	Smart Car Connectivity Platform	Transportation	bestmile.com

COMPANY NAME	PRODUCT	PRODUCT TYPE	VERTICAL	WEBSITE
Blue Pillar	Aurora	IoT Platform	Industrial IoT, Energy	bluepillar.com/aurora-energy-network-of-things-platform
Blynk	Blynk	IoT Mobile App	Prototyping, Hobbyists, DIY, Mobile	blynk.cc
Bosch	ProSyst	Connectivity Middleware	Transportation, Healthcare, Home Automation, Utilities, Industrial IoT	prosyst.com
Bosch Software Innovations	Bosch IoT Suite	IoT Platform, Device Management	Industrial IoT, Agriculture, Home Automation, Transportation, Logistics	bosch-si.com/iot-platform/bosch-iot-suite/homepage-bosch-iot-suite.html
Buddy	Ohm	Smart Building Sensors	Smart Buildings, Smart Cities	buddy.com/ohm
Bug Labs	Dweet.io	IoT Messaging Platform	Device Management, Monitoring	dweet.io
Bug Labs	Freeboard	IoT Device Visualization	Monitoring, Analytics	freeboard.io
C3 IoT	C3 IoT Platform	IoT Rapid Application Development Platform	Logistics, Smart Cities, Manufacturing	c3iot.com/products/c3-iot-platform
Calliope	Calliope Buoy	Water System Sensors	Home Automation, Utilities	calliopewater.com
Canonical	Ubuntu Core	IoT Operating System	Industrial IoT, Robotics	ubuntu.com/core
Carmine	Carmine Telematics	Fleet Management and Monitoring	Logistics, Transportation	carmine.io/telematics
Carriots	Carriots	IoT Platform, Device Management, Messaging Middleware	App Development, Data Analytics	carriots.com
Carvi	Carvi	Smart Car Sensors	Transportation, Logistics, Insurance	getcarvi.com
Casa Jasmina	Casa Jasmina	Development Community	Home Automation	casajasmina.arduino.cc
Cisco	Cisco Internet of Things Dev Center	Networking, Messaging Middleware	Industrial IoT, Smart City, Data Analytics	developer.cisco.com/site/iot
Cisco	Jasper Control Center	IoT Platform, Networking	Device Management, Analytics	jasper.com/control-center-for-iot
Compology	WasteOS	Sensor Network	Logistics, Smart City	compology.com
Concirrus	Concirrus Platform	IoT Platform, Device Management	Insurance, Data Analytics	concirrus.com
Connected Technologies	Connect One	Networking	Device Management, Healthcare, Industrial IoT, Smart City, Home Automation	simplifywithconnectone.com
ConnectM	Yantra Cloud	IoT Platform, Load Balancing, Data Analysis	Home Automation, Logistics, Industrial IoT	connectm.com/yantra-cloud.html
Control4	Control4 DriverWorks SDK	Developer Program, SDK	Home Automation	control4.com
CoreRFID	CoreRFID	Sensors	Logistics, Manufacturing, Monitoring	corerfid.com
Cumulocity	Cumulocity	IoT Platform, Messaging Middleware	Device Management, Analytics	cumulocity.com
Daintree Networks	Daintree Enterprise IoT	Sensors, Device Management	Utilities, Home Automation, Logistics	daintree.net
DataArt	DeviceHive	IoT Platform, Data Analytics	Connectivity, Analytics	devicehive.com
DGLogik	DGluk5	IoT Platform	Agriculture, Analytics, Healthcare, Industrial IoT, Smart City, Logistics	dglogik.com/products/dglux5-ioe-application-platform

COMPANY NAME	PRODUCT	PRODUCT TYPE	VERTICAL	WEBSITE
Digi	Digi XBee/RF Solutions	Networking, Sensors	Smart City, Industrial IoT	digi.com
Digi-Key	Digi-Key	IoT Electronics, Development Board	Wearables, Hardware, DIY	digikey.com
DotMatrix Technologies	DotMatrix Smart Device Language	IoT Platform	App Development and Deployment	dotmatix.net
Eclipse Foundation	Kura	Connectivity Middleware	Device Management	eclipse.org/kura
Eclipse Foundation	Vorto	IoT Platform	Device Management	eclipse.org/vorto
Edyn	Edyn Garden Sensor	Sensors	Agriculture	edyn.com
ElasticM2M	Elastic IoT Platform	IoT Platform, IoT Modules, Data Analytics	Transportation, Marine, Energy, Smarth Buildings, Climate	elasticm2m.com/products
Electric Imp	Electric Imp Platform	IoT Platform, Connectivity Middleware, Security	Utilities, Industrial IoT	electricimp.com/platform
Embedded Micro	Mojo V3	Development board	Prototyping, Hobbyists, DIY	embeddedmicro.com/products/mojo-v3.html
enModus	enModus	Smart Lighting Modules	Utilities, Smart City, Home Automation	enmodus.com/products
Eurotech	Everywhere Cloud	IoT Platform	Device Management	eurotech.com/en/products/iot
Eurotech	COM Design Boards	Development Board	Prototyping, Manufacturing	eurotech.com/en/products/iot
Evothings	Evothings Studio	Mobile App Development Platform, Device Management	Healthcare, Home Automation, Industrial IoT	evothings.com
Exosite	Murano	IoT Platform	Industrial IoT, Home Automation, Healthcare	exosite.com/platform
F5 Networks	Big-IP	Networking, Load Balancing	Industrial IoT, Smart City	f5.com/products/big-ip
Filament	Filament	Networking	Industrial IoT	filament.com
FitBit	Fitbit	Developer Program, API	Wearables	dev.fitbit.com
Gadget Factory	Papilio Duo	Development Board	Prototyping, Hobbyists, DIY	papilio.cc
GE	GE Predix	IoT Platform, Operating System	Industrial IoT	ge.com/digital/predix
Gizmosphere	Gizmo 2	Development Board	Prototyping, Hobbyists, DIY	gizmosphere.org/products/gizmo-2
Golgi	Golgi Programmable Device Cloud	Connectivity Middleware, IoT App Development Platform, Developer Program	Manufacturing, Home Automation, Device Management	golgi.io/what-is-golgi
Google	Android Wear	Developer Program, Consumer Product, API	Wearables, Mobile	android.com/wear
Google	Android Studio	IoT Development Platform	Wearables, Mobile	developer.android.com/studio
Google	Nest Developer Program	Developer Program, Smart Thermostat, API	Home Automation	developers.nest.com
Google	Project Jacquard	Connected Clothing	Device Management	atap.google.com/jacquard
Greenwave Systems	AXON Platform	Messaging Middleware, IoT Gateway, Analytics	Networking, Sensors, Mobile	greenwavesystems.com/product/axon-iot

COMPANY NAME	PRODUCT	PRODUCT TYPE	VERTICAL	WEBSITE
Helium	Helium Smart Sensors	Sensors	Home Automation, Environmental, Analytics	helium.com
Huawei	Huawei	Networking	Industrial IoT, Smart City, Agriculture, Environmental	huawei.com/minisite/iot
IBM	Bluemix	IoT Platform	App Development, Big Data Analytics	ibm.com/bluemix
Imprint	RIOT OS	Operating System	IoT Hardware and App Development	riot-os.org
Infiswift	SwiftLab	IoT Platform, Stream Processing, Data Analytics	Energy, Agriculture, Climate, Smart Cities, Home Automation, Transportation	infiswift.com/swiftlab
Insteon	Insteon Hub	Sensors, Developer Program, API	Home Automation	insteon.com/which-hub-are-you
Intel	Edison	Development board	Prototyping, Hobbyists, DIY	software.intel.com/en-us/iot/hardware/edison
Intersystems	InterSystems Data Platform	Data Management Platform for IoT Applications	Data Platform	intersystems.com/library/library-item/intersystems-data-platform-iot-applications
Jawbone	Jawbone UP	Developer Program, API	Wearables	jawbone.com/up/developer
Kentix	Kentix360	Sensors	Home Automation, Home Security	kentix.com/de/product-category/startersets
Kontakt.io	Location Engine	Messaging Middleware	Networking, Smart City, Industrial IoT, Home Automation, Agriculture	kontakt.io/location-engine
Kontakt.io	Kontakt Beacons	Beacons, Sensors	Networking, Smart City, Industrial IoT, Home Automation, Agriculture	store.kontakt.io
leakSMART	leakSMART Platform	Sensors	Home Automation, Industrial IoT	getleaksmart.com
Link Labs	Symphony Link	Development Board Modules, Routers, API	Prototyping, Industrial IoT	link-labs.com
Litmus Automation	Loop	IoT Platform, Messaging Middleware, Database	Device Management	litmusautomation.com/loop-modules
Logmein	Xively	IoT Platform, Connectivity Middleware	Industrial IoT	xively.com/xively-iot-platform
Marvell	Kinoma Create	Development Board	Prototyping, Hobbyists, DIY	kinoma.com/create
Marvell	Kinoma Element	Development Board	Prototyping, Hobbyists, DIY, Home Automation	kinoma.com/element
Mender	Mender	IoT Platform	DevOps, Automated Deployment	mender.io/product
Meshdynamics	Meshdynamics	Surveillance, Hardware, Networking	Security, Monitoring, Industrial IoT, Smart Grid	meshdynamics.com
Meshify	Meshify Enterprise	IoT Platform, Messaging Middleware	Device Management, Industrial IoT	meshify.com/applications
Microduino	mCookie	Development Board Modules	Prototyping, Hobbyists, DIY	microduino.cc/index-mcookie
Microsoft	Microsoft Azure IoT Suite	IoT Platform, Data Analytics	Device Management, Analytics, Industrial IoT	microsoft.com/en-us/internet-of-things/azure-iot-suite

COMPANY NAME	PRODUCT	PRODUCT TYPE	VERTICAL	WEBSITE
Microsoft	Windows 10 IoT Core	IoT Operating System	Robotics, Industrial IoT, Home Automation	developer.microsoft.com/en-us/windows/iot/explore/iotcore
mnubo	mnubo	Predictive Analytics for Devices	Manufacturing, Industrial IoT, Agriculture, Home Automation	mnubo.com
Mojio	Mojio	IoT Platform	Transportation, Smart Cars	moj.io/connected-car-platform
MuleSoft	MuleSoft Anypoint Platform	Connectivity Middleware	Device Management, Connectivity	mulesoft.com/platform/enterprise-integration
Muzzley	Muzzley	IoT Platform	Device Management, Home Automation	muzzley.com
MyDevices	Cayenne	IoT App Development Platform, Messaging Middleware	Data Analytics	mydevices.com/cayenne/features
Myriad Sensors	PocketLab	Sensors	Environmental	thepocketlab.com
NetBeast	Yeti	IoT Platform	Home Automation	getyeti.co
Netduino	Netduino 3 WiFi	Development Board Modules	Prototyping, Hobbyists, DIY	netduino.com/netduino3
Netvibes	Dashboard of Things	Analytics Platform	Home Automation	netvibes.com/en/explore/dashboard-of-things
Neura	Neura AI Engine	AI Engine for Personalization of Products	Healthcare, Consumer Products	theneura.com
Node-RED	Node-RED	IoT Platform	Connectivity, Device Management	nodered.org
NPM	HomeStar	Messaging Middleware	Home Automation	github.com/dpjanes/node-iotdb/blob/master/docs/homestar.md
NXP	Kinetis Cortex-M Microcontrollers	ARM Processors and Microcontrollers	Hardware	nxp.com/products/microcontrollers-and-processors/arm-processors/kinetis-cortex-m-mcus:KINETIS
Onion	Omega2	Development Board Modules	Prototyping, Hobbyists, DIY	onion.io
Open Hybrid	Open Hybrid	IoT Platform	App Development, Prototyping	openhybrid.org
Oracle	Oracle Internet of Things Cloud Service	IoT Platform	Device Management, Analytics	oracle.com/solutions/internet-of-things
Particle	Particle Cloud	Networking, Messaging Middleware	Device Management	particle.io/products/platform/particle-cloud
Particle	Particle Photon	WiFi Development Board	Prototyping, Hobbyists, DIY	particle.io/products/hardware/photon-wifi-dev-kit
Pebble	Pebble	Development Program, Smart Watch, SDK	Wearables	developer.getpebble.com
Philips	Philips Hue	Development Program, SDK	Home Automation	developers.meethue.com
PlatformIO	PlatformIO	IDE, IoT Platform	App Development	platformio.org
PrismTech	Vortex	IoT Data Analysis	Healthcare, Transportation, Smart Cities, Energy	prismtech.com/vortex/overview

COMPANY NAME	PRODUCT	PRODUCT TYPE	VERTICAL	WEBSITE
PTC	Axeda Machine Cloud	IoT Platform, Device Management	Retail, Industrial IoT, Logistics, Utilities, Healthcare	ptc.com/axeda
PTC	Thingworx Foundation by PTC	IoT Platform	App Development, Device Management, Big Data Analytics	thingworx.com/platforms
PubNub	PubNub Data Stream Network	Networking, Messaging Middleware	Connectivity, Device Management	pubnub.com
Raspberry Pi	Raspberry Pi 2 Model B	Development board	Prototyping, Hobbyists, DIY	raspberrypi.org
Razer	Razer Nabu	Developer Program, SDK	Wearables, Video Games	razerzone.com/nabu
Red Hat	Red Hat JBoss A-MQ	Connectivity Middleware	Messaging Device Management	redhat.com/en/technologies/jboss-middleware/amq
Reekoh	Reekoh	IoT Platform, Connectivity Middleware	Device Management	reekoh.com
Relayr.io	Relayr Cloud	Device Management, Machine Learning Analytics, Connectivity Middleware	Smart Cities	relayr.io/cloud
Remforce	Remforce Boiler and Leak Monitoring	Sensors	Home Automation	remforce.com
resin.io	resin.io	IoT Platform	DevOps, Automated Deployment	resin.io/features
RTI	Connext DDS	Connectivity Middleware	Industrial IoT	rti.com/products/dds
Runtime.io	Runtime.io	Device Management, Apache MyNewt Management	Device Management, Connectivity	runtime.io
Salesforce	App Cloud Lightning	IoT Platform, Device Management, Data Analytics	Connected Devices, App Development	salesforce.com/iot-cloud
Samsara	Samsara	Sensors	Logistics, Utilities, Industrial IoT, Environmental, Transportation	samsara.com
Samsung	Artik Modules	Development board	Prototyping, Hobbyists, DIY	artik.io/modules
Samsung	Artik Cloud	Data Management Platform, Device Management	Data Analytics	artik.cloud
Scanalytics	Scanalytics Floor Sensors	Sensors	Analytics, Industrial IoT, Home Automation, Retail	scanalyticsinc.com
Seeed Studio	Wio	Development Boards and Modules	Prototyping, Hobbyists, DIY	seeedstudio.com/series/Wio-11.html
Sense	Sense Home Energy Monitor	Sensors	Home Automation, Utilities	sense.com/product.html
Sierra Wireless	Sierra Wireless Embedded Solutions	Embedded Modules, Routers, IoT Gateways	Hardware	sierrawireless.com
Sigfox	Sigfox	Networking	Industrial IoT	sigfox.com
Silver Spring Networks	Starfish	Networking	Industrial IoT, Smart City	silverspringnet.com
Slock.it	Etherium Computer	IoT Device Rental	Device Management, Home Automation	slock.it/ethereum_computer.html

COMPANY NAME	PRODUCT	PRODUCT TYPE	VERTICAL	WEBSITE
Stream Technologies	IoT-X Platform	IoT Platform, Connectivity Middleware, Networking	Smart Cities	stream-technologies.com/iotx
Structural Health Systems	Concrete Sensors	Sensors	Industrial IoT, Construction	concretesensors.com
Telit	Telit IoT Portal	IoT Connectivity, IoT Platform	Smart Transportation, Agriculture, Retail, Healthcare, Automotive, Smart Manufacturing, Smart Energy, Smart Cities	telit.com/products-and-services/iot-platforms/iot-portal/
Telit	Telit IoT Modules	Wireless IoT Models	Smart Cars, Cellular Communication, Networking	telit.com/sr-rf/
Tessel.io	Tessel 2	Development Board	Prototyping, Hobbyists, DIY	tessel.io
Texas Instruments	TI Connected Launchpads	Development Boards and Modules	Prototyping, Hobbyists, DIY	ti.com/lsds/ti/tools-software/launchpads/launchpads.page
thethings.iO	thethings.iO	IoT Platform	Agriculture, Logistics, Industry, Smart Home, Smart Cities	thethings.io/iot-dashboards-features/
ThingPlus	ThingPlus Platform	IoT Platform, Connectivity Middleware	App Development, Device Management, Home Automation, Agriculture	thingplus.net
Tridium	Niagra 4	IoT App Development Platform, Analytics	Industrial IoT, Smart Cities	tridium.com/en/products-services/niagara4
U-Blox	U-Blox	Development Board WiFi Modules	Prototyping, Connectivity, Networking	u-blox.com/en/u-blox-products
Ubidots	Ubidots	IoT Platform, Data Analytics	Hobbyists, DIY, Logistics, Home Automation, Industrial IoT	ubidots.com/features
Verdigris	Verdigris	Sensor Network and Platform	Smart Buildings, Smart Cities	verdigris.co
Verizon	Hum+	Sensors, Mobile App	Smart Cars	hum.com
VersaSense	VersaSense	Sensors	Home Automation, Industrial IoT, Hobbyists, DIY	versasense.com
VSCP	VSCP	IoT Platform	Device Management	vscp.org
WebNMS	Symphony IoT Platform	Connectivity Middleware, Analytics	Energy, Smart Cities, Logistics	webnms.com/iot/unified-webnms-iot-platform.html
Wiring	Wiring S	Microcontroller IoT Platform	Prototyping, Hobbyists, DIY	wiring.org.co
WSO2	WSO2 Device Cloud	Connectivity Middleware	Device Management, Connectivity	wso2.com/iot
Wyliodrin	Wyliodrin Studio	IDE, IoT Platform	Prototyping, Hobbyists, DIY	wyliodrin.com
Zebra Technologies	Zatar by Zebra Technologies	IoT Platform, Connectivity Middleware	Prototyping, Device Management, Healthcare, Industrial IoT	zatar.com
Zolertia	RE-mote	Development Board	Prototyping, Hobbyists, DIY	zolertia.io/product/hardware/re-mote

GLOSSARY

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

AGENT
A piece of software that autonomously makes decisions for a user or another agent proactively and reactively.

ARDUINO SHIELD
An expansion board for Arduino devices that is mounted on top of the original board. Shields add new features in terms of connectivity, computational power, and so on.

COMPETING CONSUMERS
A messaging patterns in which more consumers get messages from a common source (i.e. queue) but each message is delivered to only one consumer.

CONNECTED DEVICES
Components that make up the Internet of Things. Many have built-in sensors and/or actuators and collect data to help users or other devices make informed decisions and monitor or affect outside events.

CONSENSUAL IoT
Consensual IoT is the application of the idea of Consensual Software to the Internet of Things.

CONSENSUAL SOFTWARE
Software designed to get explicit agreement from users to interact and share personal data with that software.

DIRECT MESSAGING
A messaging mechanism in which the sender and receiver are directly connected or can exchange messages through one or more intermediate hops, which do not take ownership of each message but just forward it (routing).

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.

INDUSTRIAL INTERNET
The integration of machine learning, big data technology, sensor data, and machine-to-machine communication automation. This is done with the knowledge that the Internet of Things will be scaled and driven by enterprises. The idea is that smart machines can more accurately capture and communicate data to help corporations find problems sooner and increase overall efficiency.

INTERNET OF THINGS
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.

IoT CLOUD PLATFORM
A cloud platform that provides a set of services that simplify the integration process between the services provided by cloud platforms and IoT devices. Some platforms include development tools and data analytics capabilities.

IoT DEVELOPMENT BOARD
A board that can be used to prototype and create IoT hardware. There are several boards available on the market with different features.

MESSAGING PROTOCOLS
The way information is transferred and communicated amongst devices, the cloud, and data storage. Different protocols are used for different results.

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

MULTI-AGENT SYSTEM
A network of multiple agents which act in an environment and interact or communicate with each other to achieve their design objective.

OPERABILITY
Operability is the measure of how well a software system works when operating in production, whether that is the public cloud, a co-located datacenter, an embedded system, or a remote sensor forming part of an Internet of Things (IoT) network.

PEER-TO-PEER COMPUTING
A network or architecture that splits computing tasks between several different nodes, called peers.

RELEASABILITY
Releasability is the ability to quickly deploy changes to a software system, but also to quickly recover from disaster and adapt to changing technical and business challenges.

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.

STORE AND FORWARD
A messaging mechanism in which a broker is involved between sender and receiver, so that the broker gets ownership of the message from the sender, stores it for reliability and then delivers the message itself to the receiver.

TESTABILITY
The ease of testing a piece of code or software in a given test context.

UBIQUITOUS COMPUTING
The method of enhancing computer use by making several computers available throughout a physical environment, but making them effectively invisible to the user.