THE INTERNET OF THINGS

PATTERNS FOR BUILDING REAL WORLD SYSTEMS

SUMMARY

The rapid growth of connected devices is poised to revolutionize the Internet as we know it, covering everything from our bodies to the planet. The wide range of Internet of Things ("IoT") solutions will fuel the growing API economy, providing developers endless opportunities to innovate by providing insight into the world around us. To power this future, providers of infrastructure, networks, and software will need to keep up with the growing resource demands.

This Iron.io whitepaper covers the impact on systems, and provides those building out IoT solutions a deep look into the technical considerations around operating a production-grade system. As there is no single "IoT Platform," it will take a combination of tightly integrated components that come together to form a complete platform stack that covers the flow of data, event-driven computing, API management, security, and monitoring.

CONTENTS

INTRODUCTION

Wristbands that measure every step we take to track our daily fitness activity; homes that set the inside temperature based on when you're there; cars that track mileage and driving behavior for insurance reporting; street lights that adjust levels based on weather patterns; buildings that monitor structural integrity to ensure safety standards are met. These are just a handful of the many real world IoT use cases in existence today, and why it's generating so much interest from consumers to enterprises to governments alike.

Various predictions from Gartner, IDC, Cisco, and Intel estimate the number of connected devices to reach between 25 and 200 billion by 2020. While there's a wide gap in future projections, it's clear the IoT will soon be fully engrained our daily lives.

"We are entering a world where physical objects are seamlessly integrated into the information network, and where the physical objects can become active participants of business processes." -SoftWeb

What this level of connectivity represents is the ability to capture anything, anywhere, any time... and then act on it. While the sci-fi movie fan may fear a dystopian future, the potential impact of the growing IoT market could actually "change the world" by providing intelligent context around everything, from our bodies to the planet. Manufacturing, transportation, healthcare, utilities, retail, and more are poised to be revolutionized at a micro and macro level through the latest boom in technological advancement.

"IoT is expected to be the next great technological innovation and business opportunity. It will exceed in size and importance both the personal computer and mobile communications markets, and even the development of the Internet itself." -ABI Research

IMPACT ON SYSTEMS

Despite this futuristic image of seamless connectivity all around us, there's a lot happening behind the scenes at massive scale. Sensors and actuators embedded into the billions of devices are continually capturing exabytes of data that then need to be processed, stored, and analyzed for building out intelligent applications. Given that IoT systems will still rely on the Internet as the backbone, there will be a significant impact on core infrastructure as the market expands.

A recent study by IDC predicts that IoT workloads will increase 750% by 2019, putting intense pressure on the data centers providing the infrastructure resources, the telecom companies providing the network, and the software companies providing complementary services. High availability, scalability, security, reliability, and consistency become that much more important when dealing with systems that impact the real world.

At the recent Cloud Foundry Summit in Santa Clara, Harel Kodesh, CTO of GE Software, gave a compelling keynote on the differences between the traditional Internet and the Industrial Internet. He pointed out that while Twitter may generate 80GB of data per day, a single airplane flight can generate up to a terabyte of data. While a lost connection may interrupt a phone call, another lost connection could start an oil spill. While a mobile phone may have a lifespan of a year, a turbine may need to function for up to 25 years. These comparisons really put the level of scale into perspective.

The Cloud Ecosystem is Up to the Challenge

Advancements in cloud technologies in recent years have paved the way for the rise of the IoT market, without which these new solutions wouldn't be possible. As an ecosystem we've continued to improve and optimize across the board, always ready for the ever increasing scale that is continually upon us. After initial hesitation, the cloud has become commonplace in the Enterprise and is poised to handle the coming demand of IoT workloads across a wide range of solutions.

OPPORTUNITIES FOR DEVELOPERS

Just as the capabilities of the cloud will be tested, software developers and data scientists will be tested for their ability to create compelling applications. The challenges will be in making sense of all the raw data these devices generate and how to present results in a meaningful way.

What's interesting is that for many of these IoT use cases, it's the software that provides the most value, making some hardware companies think twice about where their core business opportunities lie. Fitbit is a great example of a consumer-oriented device that becomes much more powerful through its accompanying applications. Wall Street agrees, doubling its IPO price in the first day of trading.

"IoT software platforms will become the rage, displacing the hardware."
- Forrester

Event-Driven Patterns

How IoT applications are built will be different than traditional web and mobile applications given the nature of the data sources. Using an example from a typical manufacturing use case, we can apply a general software model that can be applied to any number of IoT solutions. The business implications lie both in providing insight into the captured data, and through optimization based on the results.

1. **Things become events:** Sensors embedded into a car assembly line continually capture machine, human, and environmental data points.
2. **Events become data:** The data points are collected as a sequence of time series data points that are delivered to a data store.
3. **Data becomes insight:** Data scientists build queries on the data as it relates to part quality, process performance, etc; then turn those into consumable APIs.
4. **Insight becomes business:** Developers use APIs to build real-time dashboards that management can use to optimize the entire assembly line.

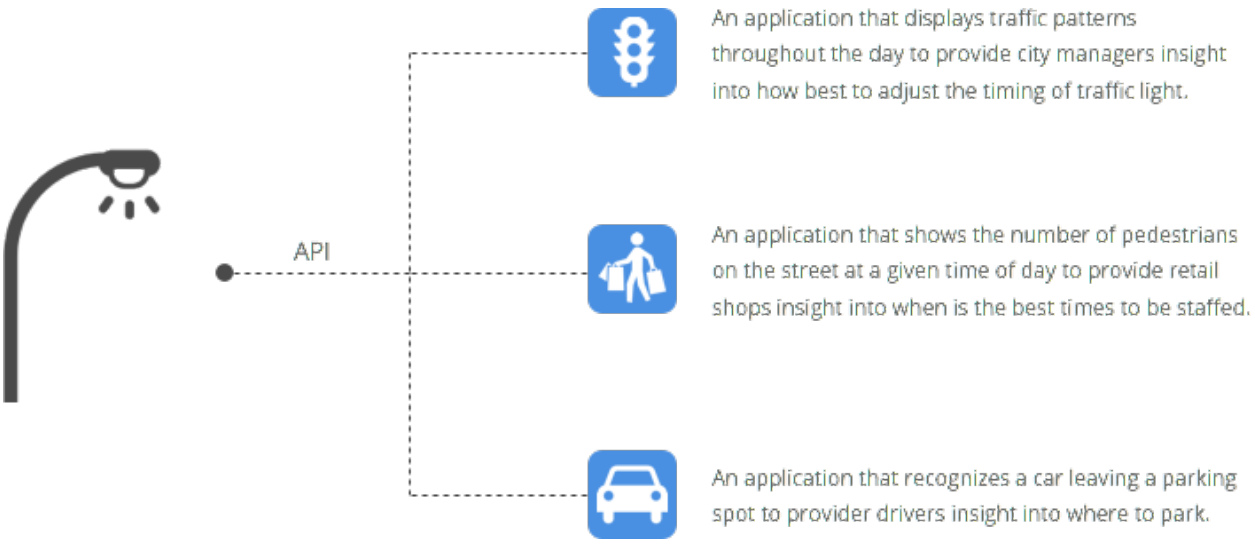
The API Economy

The key to developer empowerment in this new era will be in the consumable APIs, as transforming data into actionable intelligence is the major driver of the growing IoT market. Exposing connected device APIs also gives organizations a new opportunity to build their own developer communities, which in turn creates new business opportunities.

Where IoT applications take the API economy one step further is through the ability to compound use cases across a wide range of markets. Businesses and consumers may respond differently to the same set of data, opening up an endless amount of possibilities for developers.

Using another example, we can see how a single API gives developers the power to build disruptive solutions that can cross multiple verticals. Sensors mounted to a number of street lights continually capture passing data. Through an API, developers can build:

Figure 1: Compound Applications



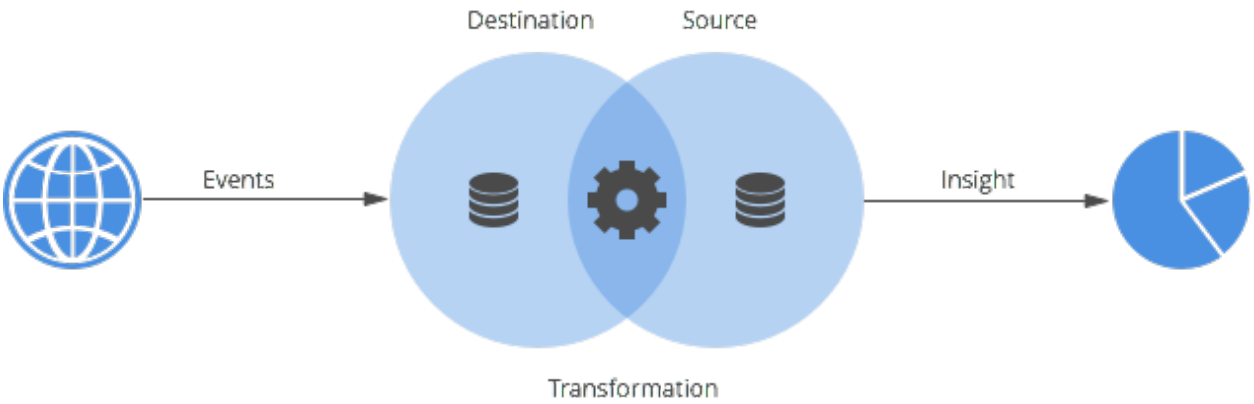
CHANGES IN DEVELOPMENT PATTERNS

With this change in how data is accessed to build solutions comes a shift in how applications and systems are structured. Modern API-driven applications have evolved away from the traditional request/response model to a more event-driven model, where services and processes are choreographed for on-demand computing. While end user applications may follow a similar model as traditional web and mobile applications, the systems behind the scenes doing the heavy lifting behave very differently.

Data Flow

In the traditional Internet with which we're most familiar, the applications we consume generally have a two-way connection with a data source for reading and writing data. What's different about IoT applications is that the data *destination* of the "things" becomes the data *source* for the applications. This means sensors and actuators need to deliver data to where it can be translated for applications to build upon. It's a modern approach to the unidirectional Extract, Transform, Load ("ETL") pattern that has been commonplace for many years for getting data from one place to another. This new data flow pattern is in line with the IoT software model of turning events into insight through transformation.

Figure 2: Data Flow From Devices to Applications

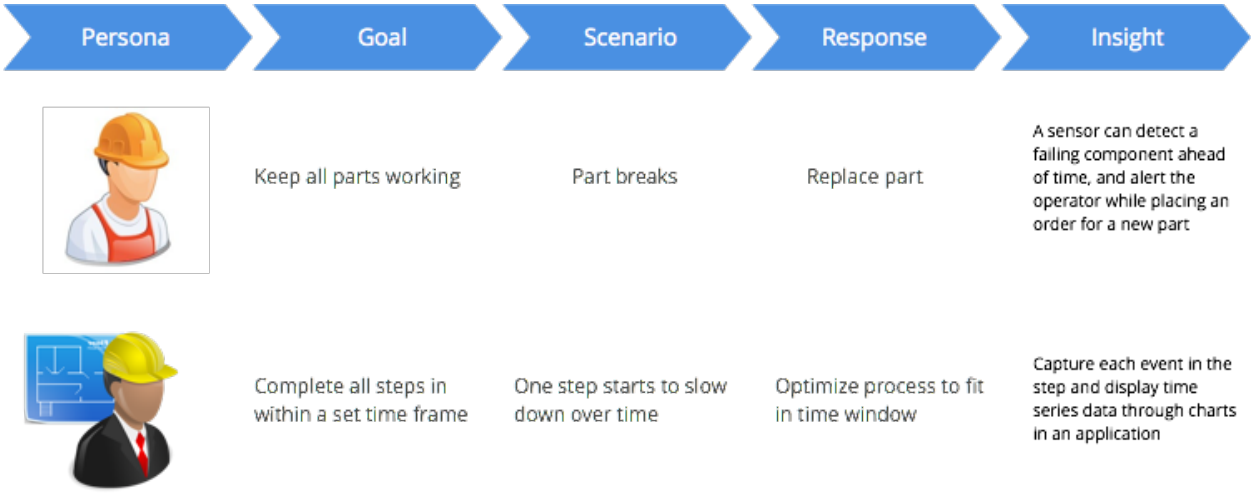


A Fresh Look at UX

Good software is designed and developed with the user experience at the forefront. The exercise of creating unique personas to describe interactions is key to any well thought out application and agile development process. With IoT applications, however, this process has evolved beyond how people interact with applications to how people interact with the world, and more importantly in many cases, how the world interacts with people. For example, changing tides can shift the schedules of commercial fisherman, while also giving a surfer a reason to get up early.

A common approach to perform a UX exercise is to define a set of personas with different goals, and then model desired functionality. With IoT applications, however, various scenarios are introduced that can impact both the persona and the goal. Building functionality means examining the response to the different scenarios, then determining the best ways to present insight. Using an example from an assembly line, we can perform a UX exercise with two personas.

Figure 3: UX Personas



NEW CONSIDERATIONS IN ARCHITECTURE DESIGN

As IoT applications have real world implications, a new set of considerations must be accounted for when building out production-grade systems. Given the high stakes, these challenges must be addressed for a successful roll out.

Varying Protocols

With such a wide array of connected devices, there will be no single standard for how they communicate. Network protocols and transport mechanisms will vary from device to device and system to system. Distinguishing factors may include whether the communication is synchronous or asynchronous, how much bandwidth is required, what the latency threshold is, and whether or not delivery guarantees are required. What's important is that regardless of the methods, the data can be translated into a structured format understood by the accompanying applications.

Table1: Machine-oriented protocols

MQTT	Lightweight TCP-based protocol meant for machine-to-machine communication. Supports pub/sub with varying QoS levels. TSL/SSL encryption.
CoAP	RESTful UDP-based request/response protocol meant for API-driven communication without the overhead of HTTP. Supports confirmable and non-confirmable transactions. Requires additional DTLS security layer.
XMPP	Legacy TCP-based protocol that supports both pub/sub and request/response. Passes XML data, which can create unnecessary overhead. TSL/SSL encryption.

Table 2: System-oriented protocols

HTTP	Cloud-native RESTful protocol meant for API-driven communication. More overhead and less performant than machine-based protocols, however standardized with the modern cloud and most application APIs. SSL encryption.
AMQP	Legacy queuing protocol used for both TCP and HTTP communication. Supports pub/sub with varying QoS levels. TSL/SSL encryption.
Websockets	Session- based communication over TCP that maintains an open connection to poll for data. Meant for realtime synchronous communication, however no QoS guarantees. TSL/SSL encryption.

Data in Transit

Massive volumes of data will be crossing networks, which will not only put a strain on bandwidth, but will also force developers to place extra care in how data is collected and delivered. With the importance of machine generated data within the end user applications, it's crucial that all captured data makes it to the backend systems doing the processing for building consumable APIs. Any point of failure in the network can result in data loss, and with more moving parts, the chances are far from slim.

Architects will select a method depending on how the data is accessed and how it needs to be delivered. Considerations will be in distance traveled, throughput requirements, protocols, and API endpoints. For persistent and ordered data transport, a queueing mechanism is highly recommended as there is less risk for data loss, and target systems won't be overloaded with too much incoming data. Message queues can be used as a buffer when large volumes of data are being passed around.

Table 3: Data Transfer Methods

Point-to-point	Direct machine-to-machine communication. Meant for real-time communication when a direct line can be made, but high risk of failure if link is broken.
Client/Server	Request/response communication method for distributing messages to backend systems. Meant for query-based communication.
Pub/Sub	Queue-based communication method where messages are published to topics, which registered subscribers pull from. Meant for asynchronous communication.
Message Queue	Messages placed on a queue can be pushed to endpoints or systems can pull on demand. Meant for event-driven communication and work dispatch.
Polling	Maintain an open socket for retrieving messages on the fly. Meant for real-time activities that need to limit the number of requests.

Data at Rest

While the lifecycle of a streamlined IoT system favors continuous motion, there's a need for robust storage mechanisms that can handle large volumes of data, and be readily accessible for analysis and query. The frequency of data collection is a key driver towards the storage type more so than its size; however, scalability is still a major factor given the potential volume.

Modern applications are built with varying data storage services to optimize accessibility. A key to any federated data architecture is to ensure eventual consistency when sharing data across multiple targets. Strong consistency is only a requirement for the end user Application API, given the nature of real-time queries.

Table 4: Data Storage Types

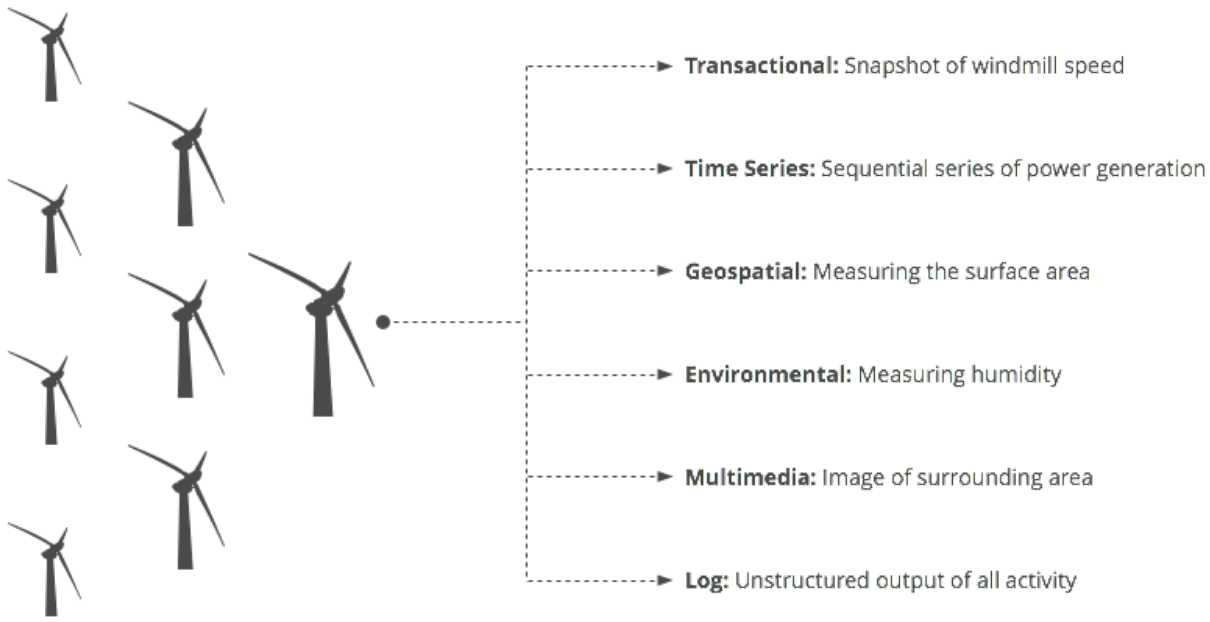
Cache	Possibly considered data in motion, a cache provides temporary storage for data that needs to be accessible in real-time on the edge of the network. Any data that needs to be persisted should still opt for a more temporary solution, but a cache can be used for reference-based rapid access.	Redis Memcached
SQL/NoSQL	Meant for structured data that has already been filtered/ tagged for applications to query. Both formats have pros and cons depending on the nature of the data (relational/document), however both scale through replication and/or sharding, and should be as highly available and consistent as possible.	PostgreSQL MongoDB
File/Object Store	Meant for storing data and content in file format such as images and video. Use a file store when a structured file system is needed and an object store when only a reference to a file is needed as objects are quicker to access.	S3 Riak
Log Store	Meant for unstructured data that needs to be searched, filtered or tagged. Varying tools can turn a collection of logs into a searchable data store. Scalability is important, but frequency of access and response time less important.	Splunk Elasticsearch
Warehouse	Production-grade systems will need to have backups of data for auditing purposes. Meant for large scale data warehousing systems that rarely need to be accessed.	Redshift Azure

Data Processing

In order to provide intelligent context around the raw data generated from devices, there will need to be varying levels of processing done to translate, filter, and analyze into useful metrics. Given the sheer volume of data and near real-time expectations around the applications themselves, effective processing methods and efficient compute resources will be needed. Classic Big Data analytics patterns may apply in some cases, but the range of data types makes for a more customized approach.

Using an example of a windmill farm, we can demonstrate a few different data types that can be captured as events for further processed. Transaction-oriented data will have large volumes of small sized data points, while multimedia-oriented data will have small volumes of large data points. Handling either will involve breaking down the processing into more manageable pieces, whether it be by volume or size.

Figure 4: Types of Data to be Processed



Security and Privacy

Restricting access to data will be a front and center theme as more IoT applications emerge. Already a hot topic as more and more of our social activities occur online, it will be compounded as connectivity enters more aspects of our lives. Unbreakable security measures must exist at every layer from the devices to the networks to the applications, especially as the cloud becomes a focus for handling the workloads.

The concerns around security aren't far from those of the early cloud. It'll take time and Enterprise-grade proof points to ease organizations with more sensitive data; however, continued assurances from the cloud ecosystem with a drive towards standardization will eventually meet requirements. In the meantime, hybrid cloud implementations will be the norm, with sensitive data and processing happening behind the firewall.

"Embedded security, trusted computing, security protocols - these are all fledgling areas of product development for the IoT and manufacturers are still trying to find their feet and just investment in secure design, development, and product lifecycle." - ABI Research

Monitoring

A complete IoT system will cover devices, networks, and data centers, with software at each layer. Monitoring each of the components will be crucial to understanding data flow, system status, and performance. There is no one tool that can span all of the different pieces that need to be monitored, so it will be a crucial exercise to pick the right tool for the respective component. Additional responsibilities will be to alert and respond accordingly, with failure scenarios in place at each step in the lifecycle to ensure that a single component does not take down an entire system.

COMPONENTS OF AN IoT SYSTEM

To bring all of the pieces together in a streamlined manner, an interoperable and federated architecture is needed to handle all aspects of a system's compute, storage, and networking functions to power large-scale IoT applications.

"The IoT presents key architectural and integration challenges for companies that are trying to take advantage of new capabilities. Devices, services, and analytics systems all need to be carefully connected together to create a multi-part value chain." - Saugatuck Technology

Gateway API

With such a heterogeneous landscape, being able to manage all the interactions will be key. It will be a common practice to introduce a glue layer between the devices and systems to handle the machine-level communication protocols, authenticate and authorize users, then route to the backend system components according to events that occur across the lifecycle of the entire solution.

The Gateway API is such a crucial piece to any IoT system, and as such, the market for integration platforms and API management tools has been heating up with companies like Mulesoft and 3Scale offering robust solutions.

Application API

The foundation for the end user applications will require an API layer that translates the device data into meaningful endpoints upon which developers can build. This component is no different than the API layer for traditional web and mobile application, it's just how the source data is compiled that makes the distinction as seen in the data flow process.

Again, an API management layer is recommended, with extra importance in proper event handling given the lifecycle of continuous data in motion.

Message Queue

If the Gateway API is the glue that connects devices and systems together, then the message queue is the glue that connects multiple systems together. Any large-scale distributed IoT application will have a number of services working together, and a message queue is the best way to transfer data and dispatch workloads across the entire system.

IronMQ by Iron.io is an industrial-strength message queue solution built for the modern cloud era, giving developers and systems architects a reliable way to incorporate varying communication layers within an IoT system. IronMQ is persistent by design, with FIFO one-time delivery and a REST API that delivers JSON data over secure HTTP, allowing for interoperability and simple integrations.

Event-Driven Computing

In the modern cloud era, containers have become the new unit of computational scale, minimizing the footprint to only the dependencies needed for the process code itself to run. Even with these optimizations, the sheer amount of compute required is still daunting. To achieve this level of scale, a system is needed that can choreograph workloads as independent tasks running within lightweight, ephemeral containers.

IronWorker by Iron.io is an event-driven computing platform that leverages containers for its runtime, giving developers and systems architects an effective environment for powering IoT workloads at massive scale. IronWorker tasks can be triggered via API call, schedule, webhook, or push queue, and scale out concurrently for maximum efficiency.

Bringing it all Together

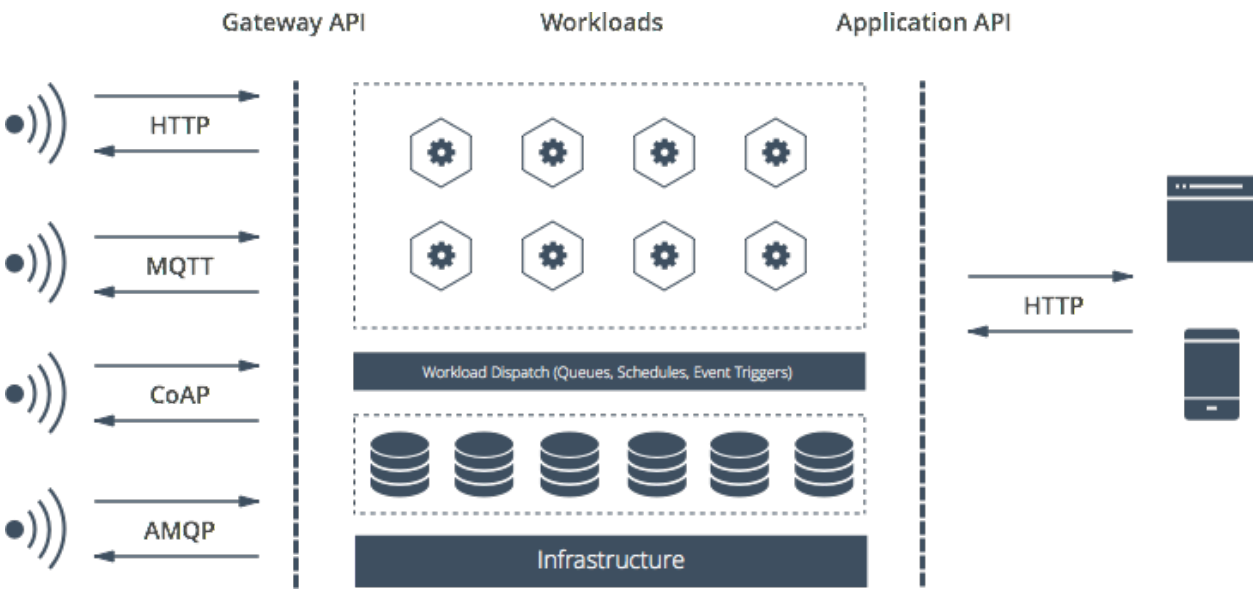
The varying components of an IoT system combined with infrastructure resources for compute, storage, and networking come together to form a complete platform to power the end user applications.

The API layers will be of the upmost importance to maintain the integrity of data; both at rest and in motion, as well as providing the structure for developers to build

upon. Without a solid Gateway API, trying to handle the various protocols and data formats will overly complicate systems. On the other end, without a solid Application API, developers will not have the flexibility to create meaningful solutions.

Too power the APIs, any large-scale system will require a robust backend to keep up with the scale of data and processing. It will be crucial to optimize the underlying infrastructure to keep costs manageable and maintain an efficient backend. Properly handling the system environment will involve components that respond to events, and dispatch workloads accordingly. These middleware components include a message queue and a task scheduler that provide effective workload management through persistent dispatch methods.

Figure 5: IoT Solution Architecture Diagram



INTEGRATED ECOSYSTEM

Despite industry hype, there is no single “IoT Platform” nor will there be one any time soon. There are just too many components to any solution, and organizations generally don’t want to be locked in with one vendor for such critical operations. While the shelf life of a mobile phone may be a year at best, a wind turbine may need to last 25 years, with the surrounding hardware and software powerful enough to handle the workloads, yet interchangeable in case of a change in the environment.

Today we solve these challenges with a number of integrated tools that interact with each other to form a cohesive, modular IoT framework. Packaged services and pluggable APIs connect these components together to become a custom platform specific to each use case. With each component comes a management, security, and monitoring layer on top to ensure proper functionality.

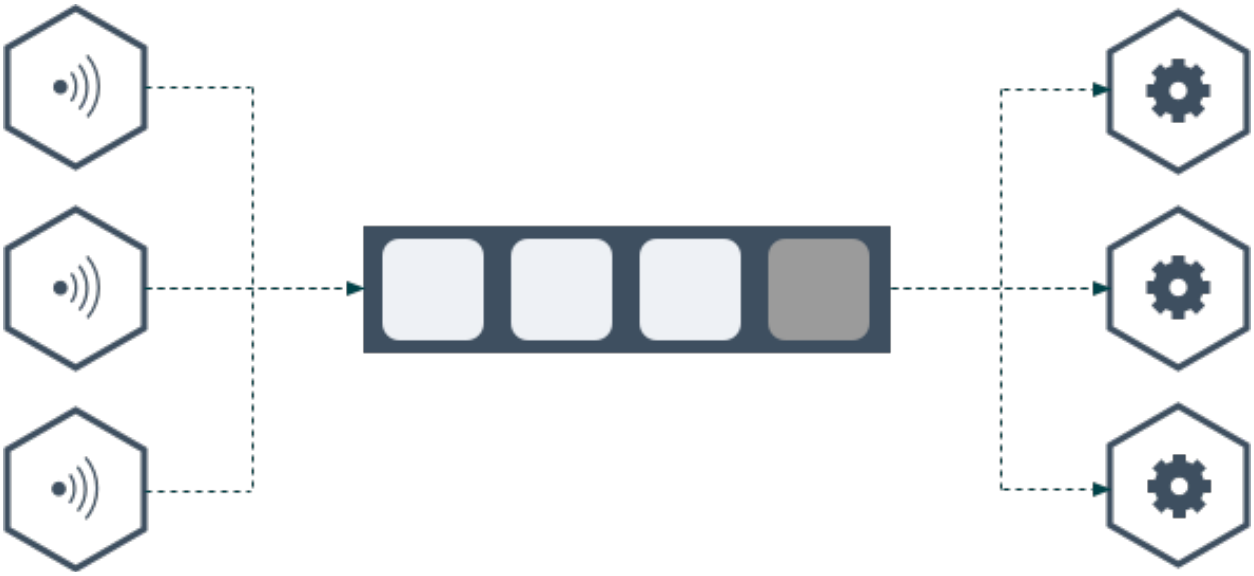
It will be up to the organization building the solutions to determine the best path to take with the vendors in each respective space depending on the data being generated, the end user applications, and the devices themselves. To that end, it’s also up to the software companies in the ecosystem to build the tools to satisfy advanced feature requirements, with simple integration and deployment capabilities to work well with the complete architecture stack.

“We’ll see increased focus on the software and especially the cloud services to make all these sensors connect, upload data, and drive analytics that generate insights and enable business improvements.” - Forrester

WHERE IRON.IO FITS

Given the asynchronous nature of IoT workloads, Iron.io is a natural fit within any large-scale system. Iron.io industrial strength products provide reliable data in transit with **IronMQ**, and effective task processing with **IronWorker**. Architected to be modular, both enable flexible deployment capabilities that can easily be integrated within systems in the cloud or on-premises for a consistent and streamlined environment.

Figure 5: Event-Driven Workload Management With Iron.io



Collect and Transport Data

Use IronMQ to pass data to and from devices and systems over HTTP without the risk of data loss.

Connect Systems Together

Use IronMQ to keep all integrated components in sync through persistent communications.

Choreograph Workloads

Use IronWorker to schedule and trigger processes without having to manage and operate infrastructure.

Process Asynchronous Tasks

Use IronWorker as an efficient compute environment containerized with only the dependencies for the task.

CONCLUSION

Whether you are a device manufacturer, developer, or solutions provider, the IoT space is one to closely watch. This "third wave" of the Internet will only further connect ourselves with the digital world, and is projected to be a multi-trillion dollar industry in just a few short years. As such, every industry will be affected significantly, and it will be up to the providers of infrastructure, network, and services to keep up with the rapid growth.

What we'll likely see as the ecosystem matures is a drive towards standardization for things like communication protocols and interfaces; however, it will remain integration-driven for the foreseeable future. It will take strategic partnerships and industry alliances to satisfy the growing demands and system requirements.

As a contributing member of the IoT ecosystem, Iron.io is dedicated to making its deployments flexible across any public or private environment, as well as tightly integrating with leading platforms and services. This allows Enterprise organizations of all kinds to incorporate Iron.io into their systems as a first class citizen.

ABOUT THE AUTHOR

Ivan Dwyer is Head of Business Development at Iron.io, working with partners across the entire cloud technology and developer services ecosystem to form strategic alliances around real world business solutions.


Iron.io