

Extracted from <http://picosec.org:8080/landscape/>

**Technology Landscape**

* [Connection types](http://picosec.org:8080/landscape/connection_types);
* [Transport security](http://picosec.org:8080/landscape/transport_security).
* [Existing protocols](http://picosec.org:8080/landscape/protocols)
* [OSs and execution environments](http://picosec.org:8080/landscape/iot-os)
* [Security technologies](http://picosec.org:8080/landscape/security-tech)

**Commercial Landscape**

* [Commercial players](http://picosec.org:8080/landscape/commercial-players)

**Implementation (hardware) Technology Research**

* [Survey of hardware tech](http://picosec.org:8080/landscape/hardware)
* [Sensor research](http://picosec.org:8080/landscape/sensors)

# Connection types

## Overview

Need to cater for a wide range of connection types. Connection capabilities of any given attached device are determined by a combination of:

* available bearers: capacity, availability, multihoming, addressability, discoverability;
* device software: processing power, available code space and available runtime memory.

Whilst trying to cater to the very widest possible range of devices, we still wish to create generic solutions; so solutions that are designed specifically for a single network type, or a very specific device constraint, or only a single use-case, are out of scope.

## Broad classification of device and connection capabilities

### Full TLS-capable devices

These devices are capable of operating as a TLS server or client, and running the TLS-based Webinos protocol with device authentication by locally-stored certificate. This entails that the devices also have the capability to store local certificates securely. These devices might not be full PZPs because of some other constraint (eg minimally interactive) but at a protocol level these would use the standard Webinos protocol.

These devices are not really in scope.

### TCP-capable devices

These are lower-end devices that are capable of communication as a TCP/IP client. These do not fall into the category of “Full TLS-capable devices”, probably as a result of one or more of the following restrictions:

* insufficient processing power, code space or runtime memory space to run PKIX aspects of TLS;
* unable to store certificates, or any long-lived credentials, securely.

For devices that are TCP/IP-capable, an alternative protocol is required to ensure the necessary transport security attributes (authentication, confidentiality, integrity, as appropriate to the use case) in lieu of the Webinos TLS/PKIX-based protocol.

TCP/IP-connected devices, and their associated connection technology, will still exhibit important differences in the connection characteristics. These differences include:

* incidence of dropped connections;
* keepalive frequency and time-to-detection of dropped connections;
* single-homing or multi-homing to networks and/or gateways;
* stability of addressing;
* discoverability.

### Bus-attached devices

These are devices that are attached to a non-IP bus or sensor network (eg Zigbee). They are connected without running TCP/IP. They might be capable of UDP connection but, more likely, they will connect and use only a transport that is specific to the networking technology they use and offers an unreliable transport akin to a serial port connection.

The connection technology may use mesh networking or otherwise be subject to frequent reconfiguration. Associations between end-devices and the infrastructure should remain in place and continue to work in the presence of these sub-network reconfigurations.

There must exist one or more points of interconnection between the sensor network and the general TCP/IP network. A mediating entity at that point of interconnection - either a transparent gateway (essentially simply “port-forwarding” for a single device) or an aggregating gateway (mediating connections for many devices).

An objective of the architecture is to make such gateways as simple as possible.

Multihoming from end devices to gateways is a requirement. Some webinos services may be proxied, or handled fully, at the gateway.

Compare the gateway architecture with [mqtt-sn](http://mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN_spec_v1.2.pdf).

# Transport security

## Overview

MXD aims to create a security architecture and protocol that is better suited to the use-cases and technology contraints of embedded IoT devices than he existing TLS/PKIX-based webinos protocols.

Embedded devices have a number of differing requirements that constraints that will drive this solution. Key issues are:

* insufficient processing, memory or connection capability to support full TLS. See the discussion in [connection types](http://picosec.org:8080/landscape/connection_types);
* inability to hold credentials securely in the long term.

The above factors affect the protocols used to ensure required security attributes of s transport (authentication, confidentiality etc).

Other aspect is that the existing Webinos trust and access control model is identity-based: a client attempting to use a service presents and identity, and a serving device uses a locally-defined policy to determine whether or not that identity has permission to perform the attempted action. For many of the IoT use-cases this model may be inappropriate because it does not naturally fit the actual trust relationships between entities. An IoT serving device is not going to be capable of understanding the real identity of a requesting client (such clients would be too numerous) and would not, in any case, make access-control decisions that depend on any specific identity. The **Policy Decision Point**, PDP (in XACML terminology) is not at the serving device but somewhere in the infrastructure.

A model that may we be more natural for the IoT use-cases is a capability-based model. Here, a service device might advertise a number of specific resources or services that are available, and might expect to see something from a requesting client that asserts that the client has the right to access that resource. A **capability** is a token that is presented by the requesting client that asserts its right to that specific access. Elsewhere in the system a PDP might have determined that right based on the requesters identity and issued the capability; but so far as the serving device is concerned it only needs to see, and be capable of verifying, the capability.

## Bus-based vs end-to-end device authentication

In the mPZP model there is a reliance on bus-based device authentication methods for tethered devices. The standard Webinos authentication protocol is terminated at a proxy device, and the end-device is authenticated only by mechanisms that are specific to the connection technology: eg a bonding operation for Bluetooth.

For MXD we seek to widen the classes of device for which authentication can be end-to-end. Ideally this will be both for TCP/IP-capable devices and for bus-attached devices.

## Trust use cases

We need to identify the trust use-cases that we plan to address.

eg:

### Capability-based access control

* client access service from IoT device;
* device grants access based on a capability presented by the clinet;
* device does not need to verify identity of client, only that it proves possession of the capability;

### Device-specific identity and authentication

* device has an identity (what? issued/managed how?)
* in creating an association between a client and a device, the device identity is authenticated (proved to the client directly or verified by some mediating device/proxy?)

### Non-specific device identity and authentication

* device has does not have an individual identity
* device is a member of a group or category which forms the basis of a trust decision made by a relying client;
* device proves membership of the group/category (proved to the client directly or verified by some mediating device/proxy?)
* an application may use data from the device based without knowing, or needing to rely on, the identity (eg a temp sensor sends gps position and sensed temperature; identity is not relevant to the app provided that the app knows that the sender is trusted to the necessary degree).

### Sensor devices as publisher only

* device publishes sensor data routinely, not in response to request;
* device may be authenticated to infrastructure/broker
* device may perform a one-time capability check, or authentication, of broker at subscription time (if there is an explicit subscription)

### Devices mutually authenticated by “bonding” step

* a one-time procedure creates a pre-shared key between device and infrastructure
* may be key agreement protocol seeded by commonly entered PIN
* this acts as implicit device authentication and implicit capability for access to device services. (However, this use case is no different from the way mainstream deployments currently work, so maybe we don’t need to specify how it works - it already just works.)

# Protocols in scope

* MQTT
* CoAP
* XMPP
* DDS
* AMQP

# MQTT

IBM sponsored protocol for IOT data transmission. Notable also for being the messaging protocol that underpins facebook messenger

MQTT[1] is a publish-subscribe based “light weight” messaging protocol for use on top of the TCP/IP protocol. It is designed for connections with remote locations where a “small code footprint” is required and/or network bandwidth is limited. The Publish-Subscribe messaging pattern requires a message broker. The broker is responsible for distributing messages to interested clients based on the topic of a message. Andy Stanford-Clark and Arlen Nipper Cirrus Link Solutions authored the first version of the protocol.

The specification does not specify the meaning of “small code foot print” or the meaning of “limited network bandwidth”. Thus the protocol’s availability for use will depend on the specific context. In 2013 IBM submitted MQTT v 3.1 to the OASIS specification body with a charter that ensured only minor changes to the specification could be accepted.[2] MQTT-S [3] is a variation of the main protocol aimed at embedded devices on non-TCP/IP networks, such as ZigBee.

Historically, the ‘MQ’ in ‘MQTT’ came from IBM’s MQ message queuing product line.[4] However, queuing per se is not required to be supported as a standard feature in all situations.[5]

Other protocols that may be used include the IETF Constrained Application Protocol[6] and XMPP.[7][8]

[MTQQ-SN spec](http://mqtt.org/new/wp-content/uploads/2009/06/MQTT-SN_spec_v1.2.pdf)

# CoAP

IETF specification on internet of things

<http://en.wikipedia.org/wiki/Constrained_Application_Protocol>

**Constrained Application Protocol (CoAP)** is a software protocol intended to be used in very simple electronics devices that allows them to communicate interactively over the Internet. It is particularly targeted for small low power sensors, switches, valves and similar components that need to be controlled or supervised remotely, through standard Internet networks. CoAP is an application layer protocol that is intended for use in resource-constrained internet devices, such as WSN nodes. CoAP is designed to easily translate to HTTP for simplified integration with the web, while also meeting specialized requirements such as multicast support, very low overhead, and simplicity.[1][2] Multicast, low overhead, and simplicity are extremely important for Internet of Things (IoT) and Machine-to-Machine (M2M) devices, which tend to be deeply embedded and have much less memory and power supply than traditional internet devices have. Therefore, efficiency is very important. CoAP can run on most devices that support UDP or a UDP analogue.

The Internet Engineering Task Force (IETF) Constrained RESTful environments (CoRE) Working Group has done the major standardization work for this protocol. In order to make the protocol suitable to IoT and M2M applications, various new functionalities have been added.[3] The core of the protocol is specified in RFC 7252, important extensions are in various stages of the standardization process.

# MQTT CoAP comparison

The IoT needs standard protocols. Two of the most promising for small devices are MQTT and CoAP.

Both MQTT and CoAP:

* + Are open standards
* Are better suited to constrained environments than HTTP
* Provide mechanisms for asynchronous communication
* Run on IP
* Have a range of implementations
* MQTT gives flexibility in communication patterns and acts purely as a pipe for binary data.
* CoAP is designed for interoperability with the web.

## MQTT

MQTT is a publish/subscribe messaging protocol designed for lightweight M2M communications. It was originally developed by IBM and is now an open standard.

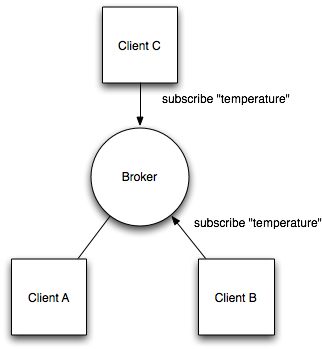
### Architecture

MQTT has a client/server model, where every sensor is a client and connects to a server, known as a broker, over TCP.

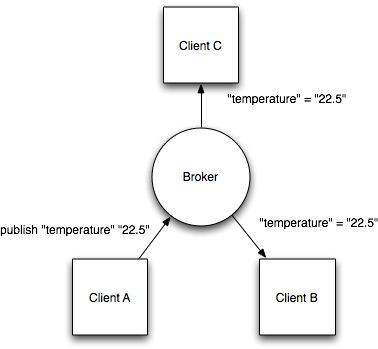
MQTT is message oriented. Every message is a discrete chunk of data, opaque to the broker.

Every message is published to an address, known as a topic. Clients may subscribe to multiple topics. Every client subscribed to a topic receives every message published to the topic.

For example, imagine a simple network with three clients and a central broker.

All three clients open TCP connections with the broker. Clients B and C subscribe to the topic temperature. 

At a later time, Client A publishes a value of 22.5 for topic temperature. The broker forwards the message to all subscribed clients.



The publisher subscriber model allows MQTT clients to communicate one-to-one, one-to-many and many-to-one.

### Topic matching

In MQTT, topics are hierarchical, like a filing system (eg. kitchen/oven/temperature). Wildcards are allowed when registering a subscription (but not when publishing) allowing whole hierarchies to be observed by clients.

The wildcard + matches any single directory name, # matches any number of directories of any name.

For example, the topic kitchen/+/temperature matches kitchen/foo/temperature but not kitchen/foo/bar/temperature

kitchen/# matches kitchen/fridge/compressor/valve1/temperature

### Application Level QoS

MQTT supports three quality of service levels, “Fire and forget”, “delivered at least once” and “delivered exactly once”.

### Last Will And Testament

MQTT clients can register a custom “last will and testament” message to be sent by the broker if they disconnect. These messages can be used to signal to subscribers when a device disconnects.

### Persistence

MQTT has support for persistent messages stored on the broker. When publishing messages, clients may request that the broker persists the message. Only the most recent persistent message is stored. When a client subscribes to a topic, any persisted message will be sent to the client.

Unlike a message queue, MQTT brokers do not allow persisted messages to back up inside the server.

### Security

MQTT brokers may require username and password authentication from clients to connect. To ensure privacy, the TCP connection may be encrypted with SSL/TLS.

### MQTT-SN

Even though MQTT is designed to be lightweight, it has two drawbacks for very constrained devices.

Every MQTT client must support TCP and will typically hold a connection open to the broker at all times. For some environments where packet loss is high or computing resources are scarce, this is a problem.

MQTT topic names are often long strings which make them impractical for 802.15.4.

Both of these shortcomings are addressed by the MQTT-SN protocol, which defines a UDP mapping of MQTT and adds broker support for indexing topic names.

## CoAP

CoAP is the Constrained Application Protocol from the CoRE (Constrained Resource Environments) IETF group.

### Architecture

Like HTTP, CoAP is a document transfer protocol. Unlike HTTP, CoAP is designed for the needs of constrained devices.

CoAP packets are much smaller than HTTP TCP flows. Bitfields and mappings from strings to integers are used extensively to save space. Packets are simple to generate and can be parsed in place without consuming extra RAM in constrained devices.

CoAP runs over UDP, not TCP. Clients and servers communicate through connectionless datagrams. Retries and reordering are implemented in the application stack. Removing the need for TCP may allow full IP networking in small microcontrollers. CoAP allows UDP broadcast and multicast to be used for addressing.

CoAP follows a client/server model. Clients make requests to servers, servers send back responses. Clients may GET, PUT, POST and DELETE resources.

CoAP is designed to interoperate with HTTP and the RESTful web at large through simple proxies.

Because CoAP is datagram based, it may be used on top of SMS and other packet based communications protocols.

### Application Level QoS

Requests and response messages may be marked as “confirmable” or “nonconfirmable”. Confirmable messages must be acknowledged by the receiver with an ack packet.

Nonconfirmable messages are “fire and forget”.

### Content Negotiation

Like HTTP, CoAP supports content negotiation. Clients use Accept options to express a preferred representation of a resource and servers reply with a Content-Type option to tell clients what they’re getting. As with HTTP, this allows client and server to evolve independently, adding new representations without affecting each other.

CoAP requests may use query strings in the form ?a=b&c=d. These can be used to provide search, paging and other features to clients.

### Security

Because CoAP is built on top of UDP not TCP, SSL/TLS are not available to provide security. DTLS, Datagram Transport Layer Security provides the same assurances as TLS but for transfers of data over UDP. Typically, DTLS capable CoAP devices will support RSA and AES or ECC and AES.

### Observe

CoAP extends the HTTP request model with the ability to observe a resource. When the observe flag is set on a CoAP GET request, the server may continue to reply after the initial document has been transferred. This allows servers to stream state changes to clients as they occur. Either end may cancel the observation.

### Resource Discovery

CoAP defines a standard mechanism for resource discovery. Servers provide a list of their resources (along with metadata about them) at /.well-known/core. These links are in the application/link-format media type and allow a client to discover what resources are provided and what media types they are.

### NAT Issues

In CoAP, a sensor node is typically a server, not a client (though it may be both). The sensor (or actuator) provides resources which can be accessed by clients to read or alter the state of the sensor.

As CoAP sensors are servers, they must be able to receive inbound packets. To function properly behind NAT, a device may first send a request out to the server, as is done in LWM2M, allowing the router to associate the two. Although CoAP does not require IPv6, it is easiest used in IP environments where devices are directly routable.

### Comparison

MQTT and CoAP are both useful as IoT protocols, but have fundamental differences.

MQTT is a many-to-many communication protocol for passing messages between multiple clients through a central broker. It decouples producer and consumer by letting clients publish and having the broker decide where to route and copy messages. While MQTT has some support for persistence, it does best as a communications bus for live data.

CoAP is, primarily, a one-to-one protocol for transferring state information between client and server. While it has support for observing resources, CoAP is best suited to a state transfer model, not purely event based.

MQTT clients make a long-lived outgoing TCP connection to a broker. This usually presents no problem for devices behind NAT. CoAP clients and servers both send and receive UDP packets. In NAT environments, tunnelling or port forwarding can be used to allow CoAP, or devices may first initiate a connection to the head-end as in LWM2M.

MQTT provides no support for labelling messages with types or other metadata to help clients understand it. MQTT messages can be used for any purpose, but all clients must know the message formats up-front to allow communication. CoAP, conversely, provides inbuilt support for content negotiation and discovery allowing devices to probe each other to find ways of exchanging data.

Both protocols have pros and cons, choosing the right one depends on your application.

Do experiments, build prototypes and deploy test devices on networks.

# XMPP

XMPP was originally called “Jabber.” It was developed for instant messaging (IM) to connect people to other people via text messages (Fig. 4). XMPP stands for Extensible Messaging and Presence Protocol. Again, the name belies the targeted use: presence, meaning people are intimately involved.

XMPP uses the XML text format as its native type, making person-to-person communications natural. Like MQTT, it runs over TCP, or perhaps over HTTP on top of TCP. Its key strength is a name@domain.com addressing scheme that helps connect the needles in the huge Internet haystack.

In the IoT context, XMPP offers an easy way to address a device. This is especially handy if that data is going between distant, mostly unrelated points, just like the person-to-person case. It’s not designed to be fast. In fact, most implementations use polling, or checking for updates only on demand. A protocol called BOSH (Bidirectional streams over Synchronous HTTP) lets severs push messages. But “real time” to XMPP is on human scales, measured in seconds.

XMPP provides a great way, for instance, to connect your home thermostat to a Web server so you can access it from your phone. Its strengths in addressing, security, and scalability make it ideal for consumer-oriented IoT applications.

# DDS

In contrast to MQTT and XMPP, the Data Distribution Service (DDS) targets devices that directly use device data. It distributes data to other devices (Fig. 5). While interfacing with the IT infrastructure is supported, DDS’s main purpose is to connect devices to other devices. It is a data-centric middleware standard with roots in high-performance defense, industrial, and embedded applications. DDS can efficiently deliver millions of messages per second to many simultaneous receivers.

Devices demand data very differently than the IT infrastructure demands data. First, devices are fast. “Real time” is often measured in microseconds. Devices need to communicate with many other devices in complex ways, so TCP’s simple and reliable point-to-point streams are far too restrictive. Instead, DDS offers detailed quality-of-service (QoS) control, multicast, configurable reliability, and pervasive redundancy. In addition, fan-out is a key strength. DDS offers powerful ways to filter and select exactly which data goes where, and “where” can be thousands of simultaneous destinations. Some devices are small, so there are lightweight versions of DDS that run in constrained environments.

Hub-and-spoke is completely inappropriate for device data use. Rather, DDS implements direct device-to-device “bus” communication with a relational data model. RTI calls this a “DataBus” because it is the networking analog to a database. Similar to the way a database controls access to stored data, a data bus controls data access and updates by many simultaneous users. This is exactly what many high-performance devices need to work together as a single system.

High-performance integrated device systems use DDS. It is the only technology that delivers the flexibility, reliability, and speed necessary to build complex, real-time applications. Applications include military systems, wind farms, hospital integration, medical imaging, asset-tracking systems, and automotive test and safety. DDS connects devices together into working, distributed applications at physics speeds.

# AMQP

Finally, the Advanced Message Queuing Protocol (AMQP) is sometimes considered an IoT protocol. AMQP is all about queues (Fig. 6). It sends transactional messages between servers. As a message-centric middleware that arose from the banking industry, it can process thousands of reliable queued transactions.

AMQP is focused on not losing messages. Communications from the publishers to exchanges and from queues to subscribers use TCP, which provides strictly reliable point-to-point connection. Further, endpoints must acknowledge acceptance of each message. The standard also describes an optional transaction mode with a formal multiphase commit sequence. True to its origins in the banking industry, AMQP middleware focuses on tracking all messages and ensuring each is delivered as intended, regardless of failures or reboots.

AMQP is mostly used in business messaging. It usually defines “devices” as mobile handsets communicating with back-office data centers. In the IoT context, AMQP is most appropriate for the control plane or server-based analysis functions.

# General discussions

Interesting discussion on power consumption in idle MQTT connections:

[MQTT Power consumption issues](http://stephendnicholas.com/archives/219)

[Useful survey of issues implementing end-to-end security on constrained devices](http://tools.ietf.org/html/draft-aks-crypto-sensors-01)

# MQTT debate

[MQTT Part 1](http://picosec.org:8080/landscape/mtqq-1-vasters)

[MQTT Part 2](http://picosec.org:8080/landscape/mtqq-2-vasters)

[MQTT Part 3](http://picosec.org:8080/landscape/mtqq-3-vasters)

# Misc Proprietary protocols

(taken from wikipedia)

## Process automation protocols[edit]

AS-i – Actuator-sensor interface, a low level 2-wire bus establishing power and communications to basic digital and analog devices BSAP – Bristol Standard Asynchronous Protocol, developed by Bristol Babcock Inc. CC-Link Industrial Networks – Supported by the CLPA CIP (Common Industrial Protocol) – can be treated as application layer common to DeviceNet, CompoNet, ControlNet and EtherNet/IP Controller Area Network utilised in many network implementations, including CANopen and DeviceNet ControlNet – an implementation of CIP, originally by Allen-Bradley DeviceNet – an implementation of CIP, originally by Allen-Bradley DF-1 - used by Allen-Bradley PLC-5, SLC-500, and MicroLogix class devices DirectNet – Koyo / Automation Direct[1] proprietary, yet documented PLC interface EtherCAT Ethernet Global Data (EGD) – GE Fanuc PLCs (see also SRTP) EtherNet/IP – IP stands for “Industrial Protocol”. An implementation of CIP, originally created by Rockwell Automation Ethernet Powerlink – an open protocol managed by the Ethernet POWERLINK Standardization Group (EPSG). FINS, Omron’s protocol for communication over several networks, including ethernet. FOUNDATION fieldbus – H1 & HSE HART Protocol HostLink Protocol, Omron’s protocol for communication over serial links. Interbus, Phoenix Contact’s protocol for communication over serial links, now part of PROFINET IO MACRO Fieldbus - “Motion and Control Ring Optical” developed by Delta Tau Data Systems. MECHATROLINK – open protocol originally developed by Yaskawa. MelsecNet, supported by Mitsubishi Electric. Modbus PEMEX Modbus Plus Modbus RTU or ASCII or TCP OSGP – The Open Smart Grid Protocol, a widely use protocol for smart grid devices built on ISO/IEC 14908.1 Optomux – Serial (RS-422/485) network protocol originally developed by Opto 22 in 1982. The protocol was openly documented[2] and over time used for industrial automation applications. PieP – An Open Fieldbus Protocol Profibus – by PROFIBUS International. PROFINET IO RAPIEnet – Real-time Automation Protocols for Industrial Ethernet Honeywell SDS – Smart Distributed System – Originally developed by Honeywell. Currently supported by Holjeron. SERCOS III, Ethernet-based version of SERCOS real-time interface standard SERCOS interface, Open Protocol for hard real-time control of motion and I/O GE SRTP – GE Fanuc PLCs Sinec H1 – Siemens SynqNet – Danaher TTEthernet – TTTech

## Industrial control system protocols[edit]

MTConnect OPC OPC UA

## Building automation protocols[edit]

1-Wire – from Dallas/Maxim BACnet – for building automation, designed by committee ASHRAE. C-Bus CC-Link Industrial Networks, supported by Mitsubishi Electric DALI DSI Dynet EnOcean – Low Power Wireless protocol for energy harvesting and very lower power devices. KNX – previously AHB/EIB LonTalk – protocol for LonWorks technology by Echelon Corporation Modbus RTU or ASCII or TCP oBIX HDL-Bus- main protocol for HDL home automation system. VSCP xAP – Open protocol X10 – Open industry standard ZigBee – Open protocol INSTEON

## Power system automation protocols[edit]

DNP3 – Distributed Network Protocol IEC 60870-5 IEC 61850 IEC 62351 – Security for IEC 60870, 61850, DNP3 & ICCP protocols Automatic meter reading protocols[edit] ANSI C12.18 DLMS/IEC 62056 IEC 61107 M-Bus ZigBee Smart Energy 2.0 Modbus

## Automobile / Vehicle protocol buses[edit]

Controller Area Network (CAN) – an inexpensive low-speed serial bus for interconnecting automotive components DC-BUS[3] – automotive power-line communication multiplexed network FlexRay – a general purpose high-speed protocol with safety-critical features IDB-1394 IEBus J1708 – RS-485 based SAE specification used in commercial vehicles, agriculture, and heavy equipment. J1939 and ISO11783 – an adaptation of CAN for agricultural and commercial vehicles Keyword Protocol 2000 (KWP2000) – a protocol for automotive diagnostic devices (runs either on a serial line or over CAN) Local Interconnect Network (LIN) – a very low cost in-vehicle sub-network Media Oriented Systems Transport (MOST) – a high-speed multimedia interface SMARTwireX Vehicle Area Network (VAN)

# Conclusions

* There are many protocols open and proprietary in teh IOT space
* For Open future looking IOT protocols MQTT and CoAP are pre-eminant
* Both protocols deal with security very badly
* Many real world IOT deployments do not even work on IP networks (see connection landscape)
* There is a real and immediate need to address security in an robust interoperable way for the future success of IOT technologies

## Contiki

Contiki is a open source operating system for IOT nodes

Contiki is an open source operating system for networked, memory-constrained systems with a particular focus on low-power wireless Internet of Things devices. Contiki provides three network mechanisms: the uIP TCP/IP stack, which provides IPv4 networking, the uIPv6 stack, which provides IPv6 networking, and the Rime stack, which is a set of custom lightweight networking protocols designed specifically for low-power wireless networks. The IPv6 stack was contributed by Cisco and was, at the time of release, the smallest IPv6 stack to receive the IPv6 Ready certification. To run efficiently on memory-constrained systems, the Contiki programming model is based on protothreads. A protothread is a memory-efficient programming abstraction that shares features of both multi-threading and event-driven programming to attain a low memory overhead of each protothread.

[Contiki](http://www.contiki-os.org/)

[Contiki supported hardware](http://www.contiki-os.org/hardware.html)

<http://www.wired.com/2014/06/contiki/>

## RIOT

<http://riot-os.org/>

Riot is a real-time multi-threading operating system that explicitly considers devices with minimal resources but eases development across the wide range of devices that are typically found in the Internet of Things. RIOT is based on design objectives including energy-efficiency, reliability, real-time capabilities, small memory footprint, modularity, and uniform API access, independent of the underlying hardware (this API offers partial POSIX compliance). Several libraries (e.g. Wiselib) are already available on RIOT, as well as a full IPv6 network protocol stack including the latest standards of the IETF for connecting constrained systems to the Internet (6LoWPAN, IPv6, RPL, TCP and UDP).

## vxworks

<http://www.machinetomachinemagazine.com/2014/03/03/wind-river-reinvents-operating-system-for-internet-of-things/>

<http://www.windriver.com/products/vxworks/index.html>

For embedded system manufacturers looking to introduce Internet of Things (IoT) capabilities to new or existing devices, VxWorks 7 delivers the right combination of scalability, safety and security, and virtualization capabilities to meet next-generation requirements.

## FreeRTOS

<https://github.com/iot-lab/iot-lab/wiki/FreeRTOS-support>

FreeRTOS is designed to be small and simple. The kernel itself consists of only three or four C files. It provides methods for multiple threads or tasks, mutexes, semaphores and software timers. Key features are very small memory footprint, low overhead, and very fast execution. IoT-LAB uses FreeRTOS by default for basic development for WSN430 and M3 nodes.

## TinyOS

<https://github.com/iot-lab/iot-lab/wiki/TinyOS-support>

TinyOS is a component-based operating system and platform targeting wireless sensor networks. TinyOS is an embedded operating system written in the nesC programming language as a set of cooperating tasks and processes. TinyOS programs are built out of software components, some of which present hardware abstractions. Components are connected to each other using interfaces. TinyOS provides interfaces and components for common abstractions such as packet communication, routing, sensing, actuation and storage.

## OpenWSN

<https://github.com/iot-lab/iot-lab/wiki/OpenWSN-support>

The OpenWSN project is an open-source implementation of a fully standards-based protocol stack for capillary networks, rooted in the new IEEE802.15.4e Timeslotter Channel Hopping standard. IEEE802.15.4e, coupled with Internet-of-Things standards, such as 6LoWPAN, RPL and CoAP, enables ultra-low power and highly reliable mesh networks which are fully integrated into the Internet. The resulting protocol stack will be cornerstone to the Internet of (Important) Things.

## Embedded Linux

<https://github.com/iot-lab/iot-lab/wiki/Linux-support>

Embedded Linux is created using OpenEmbedded, the build framework for embedded Linux. OpenEmbedded offers a best-in-class cross-compile environment.

Only A8 nodes are powerful enough to support an embedded Linux.

# Security technologies

A list of security technologies, but specifically only those deemed relevant to the IOT profile.

## Authentication and Authorization for Constrained Environments

An emmerging IETF specification, presumably related to CoAP for authenticaion and authorisation

[Main link for index of docs](https://datatracker.ietf.org/wg/ace/)

## User-Managed Access

Base on OAuth User-Managed Access (UMA) is a profile of OAuth 2.0. UMA defines how resource owners can control protected-resource access by clients operated by arbitrary requesting parties, where the resources reside on any number of resource servers, and where a centralized authorization server governs access based on resource owner policy.Met at advisory in Feb 2014.

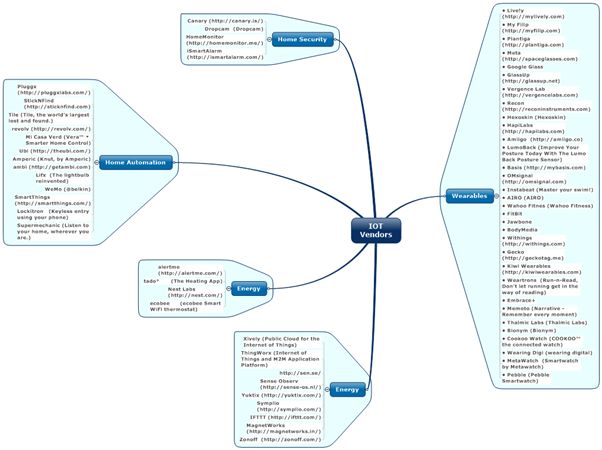
[UMA 1.0 Core Protocol](http://kantarainitiative.org/confluence/display/uma/UMA+1.0+Core+Protocol)

# IOT Management Market

|  | **By 2020 the IoT Will Include 26 Billion Units, Creating New Challenges for All Aspects of the Data Centre\_\_The Internet of Things (IoT) has a potential transformational effect on the data center market, its customers, technology providers, technologies, and sales and marketing models, according to Gartner, Inc. Gartner estimates that the IoT will include 26 billion units installed by 2020, and by that time, IoT product and service suppliers will generate incremental revenue exceeding $300 billion, mostly in services.)** |
| --- | --- |
|  |  |

Gartner have identified some principle challenges to the IOT market development

* Security — The increasing digitization and automation of the multitudes of devices deployed across different areas of modern urban environments are set to create new security challenges to many industries.
* Enterprise — Significant security challenges will remain as the big data created as a result of the deployment of myriad devices will drastically increase security complexity. This, in turn, will have an impact on availability requirements, which are also expected to increase, putting real-time business processes and, potentially, personal safety at risk.
* Consumer Privacy — As is already the case with smart metering equipment and increasingly digitized automobiles, there will be a vast amount of data providing information on users’ personal use of devices that, if not secured, can give rise to breaches of privacy. This is particularly challenging as the information generated by IoT is a key to bringing better services and the management of such devices.
* Data — The impact of the IoT on storage is two-pronged in types of data to be stored: personal data (consumer-driven) and big data (enterprise-driven). As consumers utilize apps and devices continue to learn about the user, significant data will be generated.
* Storage Management — The impact of the IoT on storage infrastructure is another factor contributing to the increasing demand for more storage capacity, and one that will have to be addressed as this data becomes more prevalent. The focus today must be on storage capacity, as well as whether or not the business can harvest and use IoT data in a cost-effective manner.
* Server Technologies — The impact of IoT on the server market will be largely focused on increased investment in key vertical industries and organizations related to those industries where IoT can be profitable or add significant value.
* Data Center Network — Existing data center WAN links are sized for the moderate-bandwidth requirements generated by human interactions with applications. IoT promises to dramatically change these patterns by transferring massive amounts of small message sensor data to the data center for processing, dramatically increasing inbound data center bandwidth requirements.



## Background

<http://techcrunch.com/2013/05/25/making-sense-of-the-internet-of-things/>

The emerging Internet of Things — essentially, the world of physical devices connected to the network/Internet, from your Fitbit or Nest to industrial machines — is experiencing a burst of activity and creativity that is getting entrepreneurs, VCs and the press equally excited.

The space looks like a boisterous hodgepodge of smart hobbyists, new startups and large corporations that are eager to be a part of what could be a huge market, and all sorts of enabling products and technologies, some of which, including crowdfunding and 3D printing, are themselves far from established.

### Building Blocks

The concept of the Internet of Things is not new (the term itself was coined in 1999), but it is now in the process of becoming a reality thanks to the confluence of several key factors.

First, while still challenging, it is easier and cheaper than ever to produce hardware – some components are open sourced (e.g. Arduino microcontrollers); 3D printing helps with rapid prototyping; specialized providers like Dragon Innovation and PCH can handle key parts of the production process, and emerging marketplaces such as Grand St. help with distribution. Crowdfunding sites like Kickstarter or Indiegogo considerably de-risk the early phase of creating hardware by establishing market demand and providing financing.

Second, the world of wireless connectivity has dramatically evolved over the last few years. The mobile phone (or tablet), now a supercomputer in everyone’s hand, is becoming the universal remote control of the Internet of Things. Ubiquitous connectivity is becoming a reality (Wi-Fi, Bluetooth, 4G) and standards are starting to emerge (MQTT). The slight irony of the “Internet of Things” moniker is that things are often connected via M2M (machine to machine) protocols rather than the Internet itself.

Third, the Internet of Things is able to leverage an entire infrastructure that has emerged in related areas. Cloud computing enables the creation of “dumb” (simpler, cheaper) devices, with all the intelligence processed in the cloud. Big data tools, often open sourced (Hadoop), enable the processing of massive amounts of data captured by the devices and will play a crucial role in the space.

### Verticals

Unlike the Big Data space, where the action is gradually moving from core infrastructure to vertical applications, the Internet of Things space is seeing a lot of early action directly at the vertical application level. Some notable players like Nest Labs seem to have adopted a deeply integrated vertical strategy where they control key pieces of the product, including both hardware and software, in order to have complete control over the end-user experience (a lot like Apple, which is not surprising considering the founders’ background).

Beyond the Nest, home automation in general has become the central battlefield of the Internet of Things, with some of the most exciting startups in the space jockeying for position. Another hot consumer-facing area is obviously quantified self, which is playing a huge role in developing consumers’ awareness of the potential of the Internet of Things.

Beyond consumer, B2B/enterprise vertical applications of the Internet of Things, fueled in part by robotics, hold considerable promise in a number of areas such as manufacturing, transportation, healthcare, retail and energy. Some of clearest revenue opportunities for IoT startups are in the enterprise area.

### Horizontals

While a lot of the action is happening at the vertical application level, the ultimate prize for many ambitious players in the space is to become the software platform upon which all vertical applications in the Internet of Things will be built. For example, several of the home automation providers (SmartThings, Ninja Blocks, etc.) also provide a software platform, and seem to be leveraging their vertical focus as a way to kickstart activity on the platform.

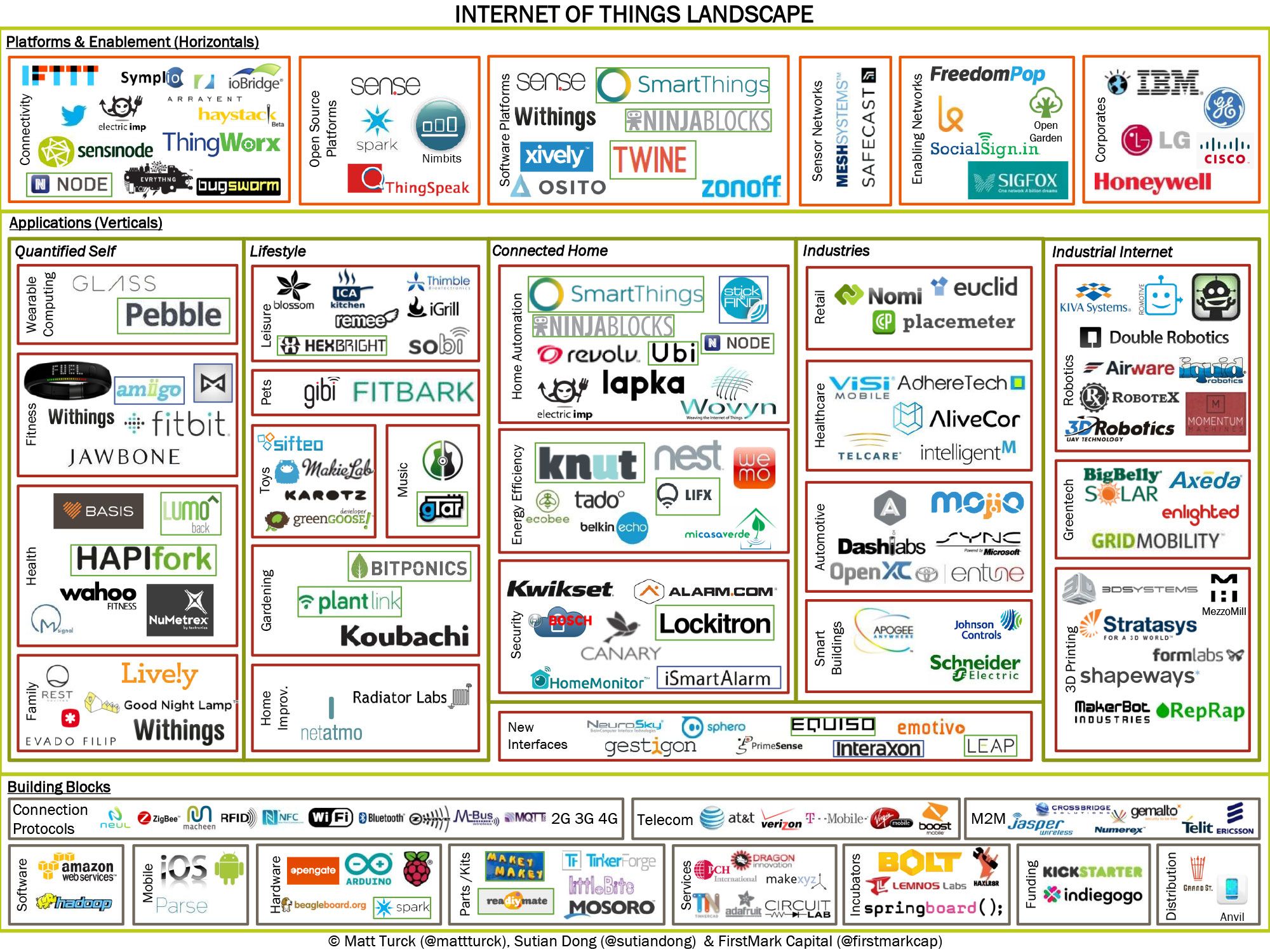
Large corporations (GE, IBM, etc.) are very active in the space and are developing their own platforms. Carriers (AT&T, Verizon) have a large opportunity in the area, as well.

One open question is whether a platform developed for a vertical will easily translate to another vertical. In addition, whether the winning platforms are open or closed will play a huge role in the future of the space. My bet would be on openness.

The related area of connectivity (connecting objects to the network/Internet and to one another through all sorts of rules) is also a very significant opportunity.

The space is extraordinarily exciting, but still very much in its infancy – expect this chart to change dramatically over the next few months and years.

The chart to the right is an attempt at making sense of this frenetic activity. From bottom to top, I see three broad areas – building blocks, verticals and horizontals:



## IOT

**Wearables**

* Live!y ([http://mylively.com](http://mylively.com/))
* My Filip ([http://myfilip.com](http://myfilip.com/))
* Plantiga ([http://plantiga.com](http://plantiga.com/))
* Meta ([http://spaceglasses.com](http://spaceglasses.com/))
* Google Glass
* GlassUp ([http://glassup.net](http://glassup.net/))
* Vergence Lab ([http://vergencelabs.com](http://vergencelabs.com/))
* Recon ([http://reconinstruments.com](http://reconinstruments.com/))
* Hexoskin (Hexoskin)
* HapiLabs ([http://hapilabs.com](http://hapilabs.com/))
* Amiigo ([http://amiigo.co](http://amiigo.co/))
* LumoBack (Improve Your Posture Today With The Lumo Back Posture Sensor)
* Basis ([http://mybasis.com](http://mybasis.com/))
* OMsignal ([http://omsignal.com](http://omsignal.com/))
* Instabeat (Master your swim!)
* AIRO (AIRO)
* Wahoo Fitnes (Wahoo Fitness)
* FitBit
* Jawbone
* BodyMedia
* Withings ([http://withings.com](http://withings.com/))
* Gecko ([http://geckotag.me](http://geckotag.me/))
* Kiwi Wearables ([http://kiwiwearables.com](http://kiwiwearables.com/))
* Weartrons (Run-n-Read, Don’t let running get in the way of reading)
* Embrace+
* Memoto (Narrative - Remember every moment)
* Thalmic Labs (Thalmic Labs)
* Bionym (Bionym)
* Cookoo Watch (COOKOO™ the connected watch)
* Wearing Digi (wearing digital)
* MetaWatch (Smartwatch by Metawatch)
* Pebble (Pebble Smartwatch)

**Home automation**

* Pluggx (<http://pluggxlabs.com/>)
* StickNFind ([http://sticknfind.com](http://sticknfind.com/))
* Tile (Tile, the world’s largest lost and found.)
* revolv (<http://revolv.com/>)
* Mi Casa Verd (Vera™ ▾ Smarter Home Control)
* Ubi (<http://theubi.com/>)
* Amperic (Knut, by Amperic)
* ambi ([http://getambi.com](http://getambi.com/))
* Lifx (The lightbulb reinvented)
* WeMo (@belkin)
* SmartThings (<http://smartthings.com/>)
* Lockitron (Keyless entry using your phone)
* Supermechanic (Listen to your home, wherever you are.)

**Home Security**

* Canary (<http://canary.is/>)
* Dropcam (Dropcam)
* HomeMonitor (<http://homemonitor.me/>)
* iSmartAlarm (<http://ismartalarm.com/>)

**Energy efficiency**

* alertme (<http://alertme.com/>)
* tado° (The Heating App)
* Nest Labs (<http://nest.com/>)
* ecobee (ecobee Smart WiFi thermostat)

**Software platforms**

* Xively (Public Cloud for the Internet of Things)
* ThingWorx (Internet of Things and M2M Application Platform)
* <http://sen.se/>
* Sense Observ (<http://sense-os.nl/>)
* Yuktix (<http://yuktix.com/>)
* Symplio (<http://symplio.com/>)
* IFTTT (<http://ifttt.com/>)
* MagnetWorks (<http://magnetworks.in/>)
* Zonoff (<http://zonoff.com/>)

![] (commercial2.jpg)

## IOT Vendor - Thingworx

ThingWorx™ provides the first platform designed to efficiently build and run the applications of today’s connected world. ThingWorx’s model-based design and search-based intelligence reduces application development efforts by 10X, minimizing cost, risk, and time to market. - See more at:[http://www.thingworx.com/#sthash.tCA9r2rA.dpuf](http://www.thingworx.com/" \l "sthash.tCA9r2rA.dpuf)

ThingWorx enables rapid creation of end-to-end “smart” applications for a wide range of markets including: Smart Agriculture, Smart Cities, Smart Grid, Smart Water, Smart Buildings, and Telematics. - See more at: <http://www.thingworx.com/#sthash.tCA9r2rA.dpuf>

ThingWorx is used with connected products in markets such as Medical/Healthcare, Construction Equipment, Transportation, Financials, Public Safety, and Consumer Products. - See more at: <http://www.thingworx.com/#sthash.tCA9r2rA.dpuf>

ThingWorx enables a new level of connectedness and intelligence for traditional industries, benefitting markets such as Automotive, Life Sciences, Manufacturing, Mining, Oil & Gas, Food & Beverage, and Utilities. - See more at: <http://www.thingworx.com/#sthash.tCA9r2rA.dpuf>

#### Acquisition

<http://www.thingworx.com/2013/12/ptc-acquires-leading-internet-of-things-platform-provider-thingworx/>

**NEEDHAM, MASS., December 30, 2013 –\*\*** [**PTC**](http://www.ptc.com/) \*\*(Nasdaq: PTC) today announced it has acquired\_\_ [ThingWorx](http://www.thingworx.com/), creators of an award-winning platform for building and running applications for the Internet of Things (IoT), for approximately $112 million, plus a possible earn-out of up to $18 million. The acquisition of ThingWorx positions PTC as a major player in the emerging Internet of Things era.

The ThingWorx acquisition extends PTC’s strategy by accelerating its ability to support manufacturers seeking competitive advantage as they create and service smart, connected products. As part of PTC, ThingWorx will continue to help customers in a wide range of industries seeking to leverage the IoT, including telecommunications, utilities, medical devices, agriculture, and transportation, as well as an emerging partner network of IoT-enabled service providers.

According to a recent research report \_\_ [Disruptive technologies: Advances that will transform life, business, and the global economy](http://www.mckinsey.com/insights/business_technology/disruptive_technologies)**(May, 2013) \_** from the McKinsey Global Institute **,** \_the Internet of Things has the potential to create economic impact of $2.7 trillion to $6.2 trillion annually by 2025. The firm believes perhaps 80 to 100 percent of all manufacturing could be using Internet of Things applications by then, leading to potential economic impact of $900 billion to $2.3 trillion, largely from productivity gains. For example, with increasingly sophisticated Internet of Things technologies becoming available, companies can not only track the flow of products or keep track of physical assets, but they can also manage the performance of individual machines and systems.

In the IoT era, PTC’s customers are bringing to market increasingly smart and connected products which can generate value in new ways as streams of real-time operational data are captured, analyzed, and shared to deepen a company’s understanding of its products’ performance, use, and reliability. PTC will use the ThingWorx platform to speed the creation of high value IoT applications that support manufacturers’ service strategies, such as predictive maintenance and system monitoring, in complement to PTC’s existing \_\_ [service lifecycle management (SLM)](http://www.ptc.com/solutions/service-lifecycle-management/)**and extended**[product lifecycle management (PLM)](http://www.ptc.com/solutions/product-lifecycle-management/)\_\_ solution portfolio. With ThingWorx, PTC will also now offer its customers a means to establish a secure, reliable connection to their products as well as a platform to rapidly develop applications for maintaining and operating them – and ultimately for finding ways to create new value from them.

“All aspects of our strategy to date have centered on helping manufacturing companies transform how they create and service smart, connected products,” said PTC president and CEO Jim Heppelmann. “For manufacturers today, it is clear to us that improved service strategies and service delivery is the near-term ‘killer app’ for the Internet of Things and this opportunity has guided our strategy for some time. With this acquisition, PTC now possesses an innovation platform that will allow us to accelerate how we help our customers capitalize on the market opportunity that the IoT presents.”

The opportunity, however, goes well beyond this immediate pragmatic application. Industries of all types are poised to see disruption from the Internet of Things and the expanding networks of connected sensors and devices, and a growing ecosystem of ThingWorx partners is forming to capitalize on this growth. As part of PTC, ThingWorx intends to continue serving this diverse market with senior management continuing to focus on its current path.

“At ThingWorx, we share PTC’s vision for helping organizations fundamentally leverage the connected world,” said Russell Fadel, CEO and co-founder, ThingWorx. “We believe all industries, but especially manufacturing, will be transformed in the Internet of Things era. We are excited to pursue this broad set of opportunities with the resources and proven solution portfolio that PTC provides.”

The acquisition is expected to add more than $10 million of revenue over the next 12 months, with $5 million to $7 million of revenue in FY’14. As a result of cost synergies and investment plans for ThingWorx, PTC still expects FY’14 non-GAAP EPS of $2.00 to $2.10. PTC drew $110 million from its credit facility to finance the transaction.

- See more at:[*http://www.thingworx.com/2013/12/ptc-acquires-leading-internet-of-things-platform-provider-thingworx/#sthash.IeCJvR09.dpuf*](http://www.thingworx.com/2013/12/ptc-acquires-leading-internet-of-things-platform-provider-thingworx/#sthash.IeCJvR09.dpuf)

## IOT Vendor - Evrything

EVRYTHNG is a Web of Things™ software company, making products smart by connecting them to the Web. EVRYTHNG helps manufacturers connect directly with customers and partners through their own products, using smartphones and intelligent identities for physical products on the Web.

The technology at the heart of making products smart is the EVRYTHNG Engine which makes it easy to turn any physical thing into a channel for personalized digital services, one-to-one communications and ongoing relationships, tied to transactions.

By making products smart, EVRYTHNG’s technology helps world-leading brands get closer to their customers and access real-time data analytics about how their products are being made, sold and used. EVRYTHNG is enabling the Web of Things™ by powering the next revolution in customer interaction and product experiences.

The EVRYTHNG Engine provides high scale, industrial technology to create and serve millions of Active Digital Identities™ for a company’s products and other objects. These unique online profiles create a persistent, unique digital presence for any physical object on the Web. Think of a Facebook for Things™ where individual objects, just like people on social networks, have their own unique digital profiles that enable communications, apps and services.

The Evrythg architecture is a centralised architecture where all infomraion is passed through a central repository.

## IOT Vendor - Alertme

AlertMe is a UK company that provides energy and home monitoring hardware and services. AlertMe produces hardware and software to enable users to monitor and control their home energy use.

## IOT Vendor - Xively

Xively by LogMeIn is dedicated to helping make the IoT real for your business as a trusted partner. Our years of experience and robust solution portfolio simplifies the complexities of the IoT, resulting in the accelerated discovery, design and implementation of real solutions that drive real results for your business.

XIvely was oringally pachube (later COSM) and was acquired in 2011 by logmein

<http://techcrunch.com/2011/07/20/logmein-acquires-internet-of-things-startup-pachube-for-15m-in-cash/>

[LogMeIn](http://logmein.com/), which lets you remotely control computers and mobiles, has acquired\_\_ [Pachube](http://pachube.com/), a UK startup which is building software for sensor-enabled devices or the legendary “Internet of Things”. The $15 million purchase is all cash. We understand the team is staying on.

Pachube networks appliances, environmental sensors, cars and personal health monitors – you name it, this is a market set to explode over the next few years. Michael Simon, CEO of LogMeIn said the purchase will extend its Gravity platform into smart embedded devices.

Pachube’s realtime monitoring platform means users send more than seven million datapoints to the service each day. Founder, Usman Haque said “we are in a strong position to bring our shared vision for the Internet of Things to fruition.”

It’s also another “win” for a company based in the ‘Silicon Rounadabout’ area of London which the UK government is trying to re-brand internationally as “TechCity”.

# IOT Hardware

In considering hardware there are a number of dimensions to consider

* Microcontroller (CPU)
* Radio technology
* Integrated solutions - combined radio and microcontroller
* Developer environment

The developer environment will be more fully fleshed out in the tool chain section

## Micro controllers

The following list of devices are typical of the type of microcontroller to be found in an IOT device

Power is the principle limiting constraint on such systems

* TI 8051
* Arm Cortex family
* ATMega family

### TI 8051

<http://www.ti.com/mcu/docs/mcugeneralcontent.tsp?sectionId=98&tabId=1515>

Microcontrollers Parametric MSC USB MHz 33 Up to 60 MIPS 8 Up to 30 Architecture 8-bit 8-bit Code Space Up to 32 kB Flash Up to 32 kB RAM RAM Up to 1.2 kB Up to 40 kB ADC Channels Up to 8 - ADC Resolution 24-bits - DAC Channels Up to 4 - DAC Resolution 16-bits - Vref Yes - Serial Interfaces SPI, I2C, USART USB, I2C, RS232, IrDA Pricing (1KU) $4.60 - $20.95 $1.15 - $5.10

### Arm Cortex ME

<http://en.wikipedia.org/wiki/ARM_Cortex-M3#Cortex-M3>

Key features of the Cortex-M3 core are:[<http://infocenter.arm.com/help/topic/com.arm.doc.ddi0337i/DDI0337I_cortexm3_r2p1_trm.pdf> Cortex-M3 r2p1 Technical Reference Manual; ARM Holdings.]{{cite web | last=Sadasivan | first=Shyam | title=An Introduction to the ARM Cortex-M3 Processor | url=<http://www.arm.com/files/pdf/IntroToCortex-M3.pdf> | publisher=[[ARM Holdings]]}}

* ARMv7-M architecture
* Instruction sets **Thumb (entire)**Thumb-2 (entire) \*\* 1-cycle 32-bit hardware multiply, 2-12 cycle 32-bit hardware divide, saturated math support
* 3-stage [[pipeline (computing)|pipeline]] with [[branch predictor|branch speculation]]
* 1 to 240 physical [[interrupt]]s, plus [[non-maskable interrupt|NMI]]
* 12 cycle interrupt latency
* Integrated sleep modes
* 8 region [[memory protection]] unit (MPU) (silicon option)
* 1.25 DMIPS/MHz
* [[90 nanometer|90 nm]] implementation{{cite web | title=ARM Cortex-M3 Specifications | url=<http://arm.com/products/processors/cortex-m/cortex-m3.php> | publisher=[[ARM Holdings]]}} **32 µW/MHz**0.12 mm2

#### Chips

The following microcontrollers are based on the Cortex-M3 core:

* Actel Actel SmartFusion|SmartFusion, SmartFusion 2
* Analog Devices ADuCM360
* Atmel AT91SAM#AT91SAM3|SAM3A, SAM3N, SAM3S, SAM3U, SAM3X
* Cypress Semiconductor PSoC 5
* Energy Micro EFM32|EFM32 Tiny, Gecko, Leopard, Giant
* Fujitsu FM3
* Holtek HT32F125x
* Luminary Micro LM3S1968
* NXP Semiconductors|NXP LPC1300, LPC1700, LPC1800
* ON Semiconductor Q32M210
* Silicon Laboratories|Silicon Labs Precision32
* STMicroelectronics STM32|STM32 F1, F2, L1, W
* Toshiba TX03

The following chips have a Cortex-M3 as a secondary core:

* CSR plc|CSR Quatro 5300 series (Cortex-M3 as co-processor)
* Texas Instruments F28, LM3, TMS470, OMAP#OMAP 4|OMAP 4470 (one ARM Cortex-A9 MPCore|Cortex-A9 + two Cortex-M3)

The ARM Cortex M0+ is lower down the hierachy

Nordic supplies a chipset

<http://www.nordicsemi.com/eng/Products/Bluetooth-R-low-energy/nRF51822>

which is built into the more developer friendly

<http://www.rfduino.com/>

### ATMega 328P

<http://www.atmel.com/devices/atmega328.aspx>

The high-performance Atmel 8-bit AVR RISC-based microcontroller combines 32KB ISP flash memory with read-while-write capabilities, 1KB EEPROM, 2KB SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible timer/counters with compare modes, internal and external interrupts,serial programmable USART, a byte-oriented 2-wire serial interface, SPI serial port, 6-channel 10-bit A/D converter (8-channels in TQFP and QFN/MLF packages), programmable watchdog timer with internal oscillator, and five software selectable power saving modes. The device operates between 1.8-5.5 volts. By executing powerful instructions in a single clock cycle, the device achieves throughputs approaching 1 MIPS per MHz, balancing power consumption and processing speed.

ParameterValue Flash (Kbytes): 32 Kbytes Pin Count: 32 Max. Operating Freq. (MHz): 20 MHz CPU: 8-bit AVR

### Atmel ATSAM3S4A

<http://www.atmel.com/devices/sam3s4a.aspx>

A member of the Atmel® SAM3S series microcontrollers based on the high-performance 32-bit ARM® Cortex™-M3 RISC processor. It operates at a maximum speed of 64MHz and features 256KB of flash and 48KB of SRAM. The extensive peripheral set includes a FS USB device port with embedded transceiver, USART, two UARTs, two TWIs, two SPIs, I2S, PWM timer, three 16-bit timers, RTC, eight 12-bit ADCs and an analog comparator. The QTouch® library offers an easy way to implement buttons, wheels and sliders. The parallel data capture mode on the PIOs collects data from external devices not compliant with standard memory read protocols, such as low-cost image sensors. The data is then transferred to memory by DMA offloading the CPU. The device operates from 1.62V to 3.6V, is available in 48-pin QFP and QFN packages, and is pin-to-pin compatible with the SAM3N4A.

Price $6 at volume

## Radio solutions

## Integrated solutions

These solutions include both radio and MCU

### Silicon labs SI 106x

<http://www.silabs.com/products/wireless/wirelessmcu/pages/si106x-8x.aspx>

$4 single unit price

### Freescale

Freescale Kinetis range include Arm Cotex MCUs. and subghz radios

<http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=KW0x&nodeId=01624698C9F3F4>

vs

<http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=KW2x>

<http://uk.pcmag.com/cpus-components-products/news/8052/freescale-intros-tiny-arm-based-mcu-for-internet-o>

price 2.7 for KWO0 to 5.8 for WW2

The higher spec unit comes with

* + Cryptography Acceleration Unit (CAU)
* AES encryption (FIPS 140)
* External tamper detect
* 32-bit CRC

### Higher end freescale

MC13224V <http://www.freescale.com/webapp/search.partparamdetail.framework?PART_NUMBER=MC13224VR2&buyNow=true&fromSearch=true> end of life but contikis supported

$4.7

### SQI50

<http://www.sequoia.co.uk/product.php?id=1852#Specification>

Price $5 at 100 units

For more details of the specification and MCU hardware details please download the datasheet.

Ultra Low Power: 0.9 to 3.6 V Operation

-Typical sleep mode current < 0.1 μA; retains state and RAM contents over full supply range; fast wakeup of < 2 μs -Less than 600 nA with RTC running -Less than 1 μA with RTC running and radio state retained -On-chip dc-dc converter allows operation down to 0.9 V. -Two built-in brown-out detectors cover sleep and active modes 10-Bit Analog to Digital Converter

-Up to 300 ksps -Up to 18 external inputs -External pin or internal VREF (no external capacitor required) -Built-in temperature sensor -External conversion start input option -Autonomous burst mode with 16-bit automatic averaging accumulator Dual Comparators

-Programmable hysteresis and response time -Configurable as interrupt or reset source -Low current (< 0.5 μA) On-Chip Debug -On-chip debug circuitry facilitates full-speed, non-intrusive in-system debug (No emulator required) -Provides breakpoints, single stepping -Inspect/modify memory and registers -Complete development kit High-Speed 8051 μC Core

-Pipelined instruction architecture; executes 70% of instructions in 1 or 2 system clocks -Up to 25 MIPS throughput with 25 MHz clock -Expanded interrupt handler Memory

-4352 bytes internal data RAM (256 + 4096) -64 kB Flash; In-system programmable in 1024-byte sectors—1024 bytes are reserved in the 64 kB devices

**Transceiver**

-Frequency range = 433,470,868,915 MHz ISM Band -Sensitivity = –121 dBm -FSK, GFSK, and OOK modulation -Max output power = +20 dBm -RF power consumption -18.5mA receive -18 mA @+1 dBm transmit -40mA @+13 dBm transmit -100mA @+20 dBm transmit -Data rate = 0.123 to 256 kbps -Auto-frequency calibration (AFC) -Transmit/receive switch control -Programmable packet handler -TX and RX 64 byte FIFOs -Frequency hopping capability -On-chip crystal tuning Digital Peripherals

-19 or 16 port I/O plus 3 GPIO pins; Hardware enhanced UART, SPI, and I2C serial ports available concurrently -Low power 32-bit SmaRTClock -Four general purpose 16-bit counter/timers; six channel programmable counter array (PCA) Clock Sources

-Precision internal oscillators: 24.5 MHz with ±2% accuracy supports UART operation; spread-spectrum mode for reduced EMI; Low power 20 MHz internal oscillator -External oscillator: Crystal, RC, C, CMOS clock -SmaRTClock oscillator: 32.768 kHz crystal or self-oscillate -Can switch between clock sources on-the-fly; useful in power saving modes and in implementing various power saving modes I/O Port

-19 or 20 port I/O (5 V tolerant except for GPIO\_2) Package

-30 pin SMD (11x25X2.0 mm)

### Microchip MCU +RF

<http://www.microchip.com/wwwproducts/Devices.aspx?product=PIC12F529T48A>

MHzPIC12LF1840T48A7.1K418

PIC12LF1840T48A7.1K418–868 MHz

with 256 bytes of ram

looks too low range

### Texas Instruments

<http://www.ti.com/general/docs/lit/getliterature.tsp?genericPartNumber=cc1110f32&fileType=pdf>

Sub Ghz

32 flash/ 4 ram/ 8051 MCU

$2.5

<http://www.ti.com/product/CC2533/samplebuy>

96 flash/ 4 ram/ 8051 $3.1

<http://www.ti.com/product/cc2538>

CC2538 - model 802.14.5 + ZIgbee + 6lowpan

Microcontroller Powerful ARM Cortex M3 With Code Prefetch Up to 32 MHz Clock Speed 512-kB, 256-kB or 128-kB In-System-Programmable Flash Supports On-Chip Over-the-Air Upgrade (OTA) Supports Dual ZigBee Application Profiles Up to 32-kB RAM (16-kB With Retention in All Power Modes) cJTAG and JTAG Debugging

512-128 Flash/ 32 ram/ Arm3

$4 at volume

### ATmega256RFR2

An IEEE 802.15.4 compliant single chip combines an industry-leading AVR microcontroller and best-in-class 2.4GHz RF transceiver. It offers the industry’s highest RF performance for single chip devices, with a link budget of 103.5dBm while consuming 50% less current than the existing offerings. The device features hardware assisted multiple PAN address filtering (MAF), improved channel masks on CH25 and CH26 allowing the device to run full power (1W) on these channels using external power amplifiers, wake-on radio, improved link efficiency and reliability using RX override, 32-bit MAC symbol counter, temperature sensor, automatic transmission modes, 128-bit AES crypto engine, true random number generator, high data rate modes, and antenna diversity support.

Parameter Value Flash (Kbytes): 256 Kbytes Max. Operating Freq. (MHz): 16 MHz Pin Count: 48 Max I/O Pins: 38 Pin Count: 64 SPI: 1 TWI (I2C): 1 CPU: 8-bit AVR UART: 1 ADC channels: 8

Price: $5 at 1000 units

## Develop grade boards

We distinguish these from the commercial systems largely based on price, you can typically pay 10x price for developer friendly boards over and above the base components

These include the Arduino based solutions

### JeeNode

<http://moderndevice.com/products/jeenode-kit>

JeeNodes are a collaboration between Modern Device and a talented and energetic engineer, Jean-Claude Wippler. See his impressive blog and webshop. We do much of the manufacturing for JeeLabs and are the US representative for JeeLabs products.

The flagship Jee Product is called the JeeNode. Jean-Claude was inspired partly by the RBBB, but he added two big ideas of his own. The first idea was to couple a low-cost radio to the RBBB, enabling wireless communication. For many people who wish to just send a few bytes from a sensor to a receiver, the available wireless options, such as XBee, are expensive overkill. So we think this little board fills an useful niche at an affordable price.

Radios are available in 433 MHz & 915 MHz in the US and 433 & 868 MHz in Europe, all unlicensed ISM (Industrial, Scientific, Medical) bands. We only ship 915 MHz radios to countries that use the 915 MHz band. This includes countries in North and South America, Greenland, Israel, and Australia for a brief list. Please fill us in if your country uses the 915 Mhz ISM band.

Price $20 per node

### PanStamp

Price 13.50 euro per node

### General list from Postscapes

<http://postscapes.com/arduino-wifi#rf>

### Anarduino - build in 868

<http://www.anarduino.com/miniwireless/>

Better from price point that above system. Also considerably smaller.

### GBoard built in arduino and GSM

<http://imall.iteadstudio.com/im120411004.html>

# Developer board transceiver systems

## enocean 868 module

<http://uk.farnell.com/enocean/enocean-pi-868/rf-module-transceiver-ask-868mhz/dp/2322460>

## 868 extension for PI

<http://busware.de/tiki-index.php?page=COC>

Price $50+

# Development Systems

## Silicon Labs Development board

<http://www.digikey.com/product-search/en?v=336&mpart=1060-915-DK>

based on Si106x/8x Wireless MCUs

## Freescale dev boards

<http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=KWIKSTIK-K40>

<http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=KINETIS_SDK>

<http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=KDS_IDE>

<http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=FRDM-KL25Z>

# TI Development boards

# TI - Contiki supported

<http://www.ti.com/tool/cc2538dk>

The CC2538 development kit provides a complete hardware performance test platform and generic software development environment for the ARM Cortex M3-based, IEEE 802.15.4 compliant CC2538 system-on-chip from Texas Instruments. The kit includes two CC2538 RF evaluation modules (CC2538EM), two general purpose development boards (SmartRF06EB) for software and hardware prototyping, one CC2531 USB dongle for packet sniffing, cables, and documentation to get you up and running with the CC2538 quickly and easily.

The CC2538 evaluation modules can be used as reference modules for prototyping and for verifying the performance of the CC2538 RF IC. In combination with SmartRF06EB and SmartRF Studio, you have a complete set of tools for measuring RF parameters and more generally evaluate the RF performance of the chip. SmartRF Studio can generate register values and set up the radio to send and receive packets, set up a continuous wave signal and read the received signal strength (RSSI).

The SmartRF06EB has integrated the XDS100v3 debug probe, so no additional hardware tools are required for debugging of software running on the devices. The debug probe is compatible with Code Composer Studio and IAR Embedded Workbench. Other JTAG probes that support ARM Cortex M3 devices, like JLink from Segger and I-jet from IAR, can also be used to download and debug software on the CC2538.

The CC2538EM boards come pre-programmed with a Packet Error Rate test, which can be used for practical range testing of the radio. Texas Instruments’ ZigBee stack (Z-Stack) is of course also available for the kit.

<http://uk.farnell.com/texas-instruments/cc2538dk/cc2538-zigbee-802-15-4-dev-kit/dp/2356505>

Developer kit £220

### ATMel ATMega

volume prince 10us

can work out if it has integrated RF- contiki says yes but cannot find

### Freescale MC13224V: 2.4 GHz 802.15.4 RF and 32-bit ARM7™ MCU with 128KB Flash, 96KB RAM

too high spec - must be pricey and power hungry farnell stated discontinued

# Conclusions

We will need to refine and experiment with profiles over time

The sweet spot on price (sub dollar) is a 32 Flash 4k RAM 8051 MCU with Sub Ghz RF. However this will be a too challenging for initial development

The initial target platform will be based on Texas Instrument kit

* <http://www.ti.com/product/cc2538>
  + 256 flash
  + 32 k ram
  + ARM cortex M3
  + 2.4-GHz IEEE 802.15.4 Compliant RF Transceiver
  + JTag debugging
  + 2 × SPI
  + 2 × UART
  + I2C
  + $4 volume pricing
* SDK <http://www.ti.com/tool/cc2538dk>
* Supports contiki

# List of gas sensor

<http://playground.arduino.cc/Main/MQGasSensors>

# Gas sensor to buy

<http://www.futurlec.com/Gas_Sensors.shtml>

# CO2 induced

<http://davidegironi.blogspot.it/2014/01/cheap-co2-meter-using-mq135-sensor-with.html#.U76II_kh> jpU

<http://www.veetech.org.uk/Prototype_CO2_Monitor.htm>

## Misc Tools and Apps

An IOT commercial dashboard application

<https://datatracker.ietf.org/wg/jose/>)