Components in IoT System Design : Software Components

Device Software

Allows you to implement communication to the Cloud or to other local devices.

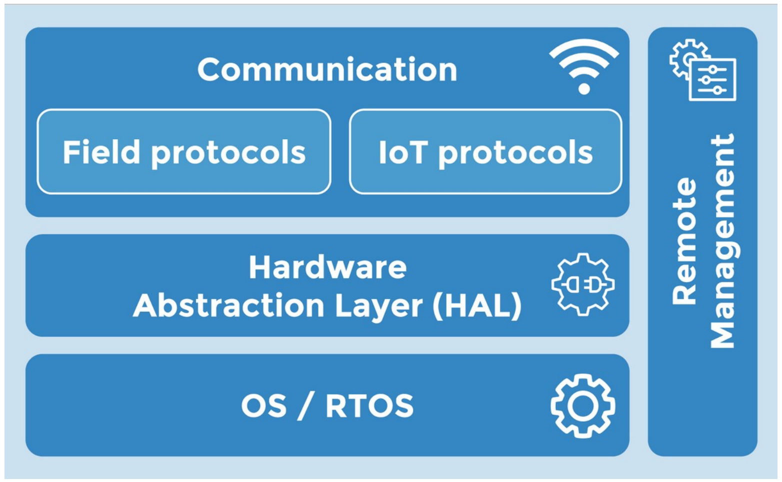
Serves as the glue between the real world (hardware) and Cloud Applications.

Implement real-time analytics, data acquisition from device's sensors and control

The software stack for an IoT System can be divided as :

1. Software Stack of Devices
2. Software Stack for Gateways
3. Software Stack for Cloud

Software Stack for Devices



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

IoT Operating System – embedded or real-time operating systems that are particularly suited for small constrained devices providing IoT-specific capabilities.

Hardware Abstraction – a software layer that enables access to the hardware features of the MCU, such as flash memory, GPIOs, serial interfaces, etc.

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

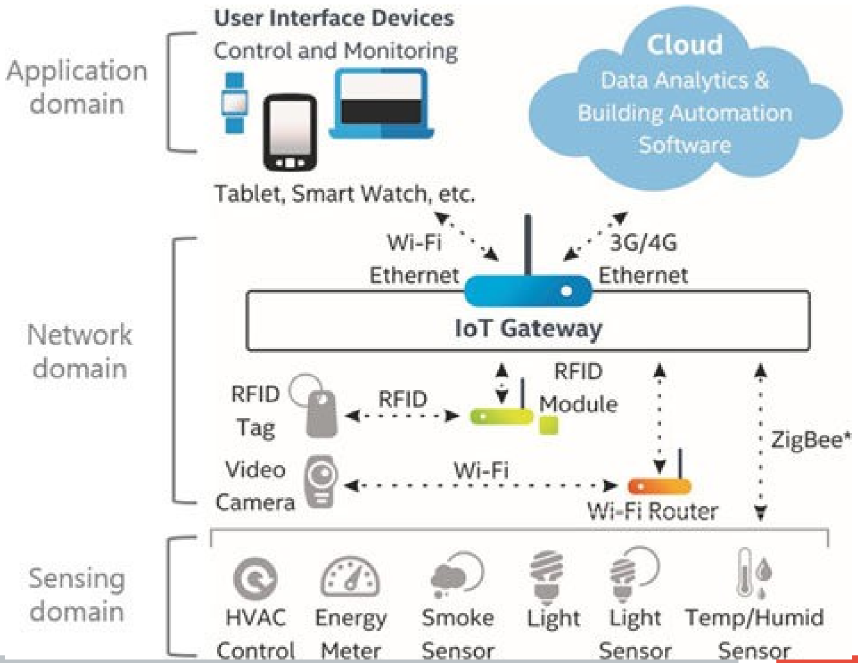
Remote Management – the ability to remotely control the device to upgrade its firmware or to monitor its battery level.

Software Stack for Gateway

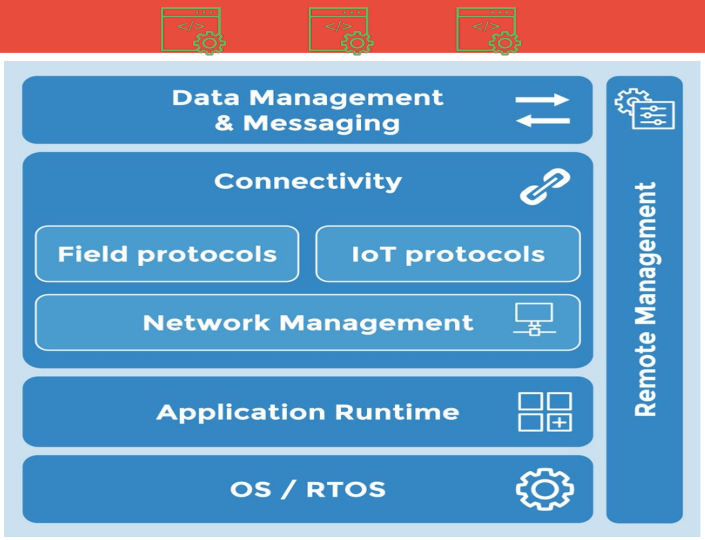
Gateway : Aggregation point for group of sensors/actuators

It can be a physical piece of hardware or functionality that is incorporated into a larger “Thing” that is connected to the network.

An IoT gateway will often offer processing of the data “at the edge” and storage capabilities to deal with network latency and reliability. For device to device connectivity, an IoT gateway deals with the interoperability issues between incompatible devices. A typical IoT architecture would have many IoT gateways supporting masses of devices.



Software Stack for Gateways



Operating System – typically a general purpose operating system such as Linux.

Application Container or Runtime Environment – IoT gateways will often have the ability to run application code, allow the applications to be dynamically updated. For example, a gateway may have support for Java, Python, or Node.js.

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

Data Management & Messaging – local persistence to support network latency, offline mode, and real-time analytics at the edge, as well as the ability to forward device data in a consistent manner to an IoT Platform.

Remote Management – the ability to remotely provision, configure, startup/shutdown gateways as well as the applications running on the gateways.

Sofftware Stack for Cloud

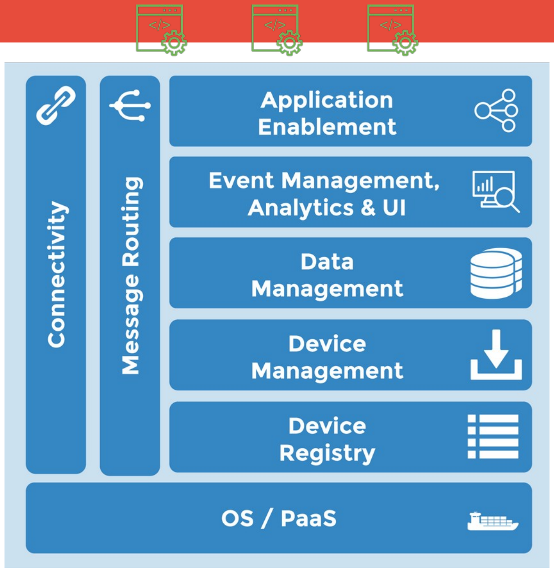
The IoT Cloud Platform represents the software infrastructure and services required to enable an IoT solution.

An IoT Cloud Platform typically operates on a cloud infrastructure scale both horizontally, to support the large number of devices connected, as well as vertically to address the variety of IoT solutions.

Example : OpenShift, AWS, Microsoft Azure, Cloud Foundryor inside an enterprise data center and is expected to The IoT

Cloud Platform will facilitate the interoperability of the IoT solution with existing enterprise applications and other IoT solutions.

Software Stack for Cloud



Features of an IoT Cloud Platform include

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

Device Management and Device Registry – a central registry to identify the devices/gateways running in an IoT solution and the ability to provision new software updates and manage the devices.

Data Management and Storage – a scalable data store that supports the volume and variety of IoT data.

Event Management, Analytics & UI – scalable event processing capabilities, ability to consolidate and analyze data, and to create reports, graphs, and dashboards.

Application Enablement – ability to create reports, graphs, dashboards and to use API for application integration.

Cross-Stack Functionality

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

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

Ontologies – The format and description of device data is an important feature to enable data analytics and data interoperability. The ability to define ontologies and metadata across heterogeneous domains is a key area for IoT.

Development Tools and SDKs – IoT Developers will require development tools that support the different hardware and software platforms involved.

IoT Stack Characteristics

Loosely coupled - Three IoT stacks have been defined but it is important that each stack can be used independently of the other stacks. It should be possible to use an IoT Cloud Platform from one supplier with an IoT gateway from another supplier and a third supplier for the device stack.

Modular - Each stack should allow for the features to be sourced from different suppliers.

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

Based on open standards - Communication between the stacks should be based on open standards to ensure interoperability.

Defined APIs - Each stack should have defined APIs that allow for easy integration with existing applications and integration with other IoT solutions.