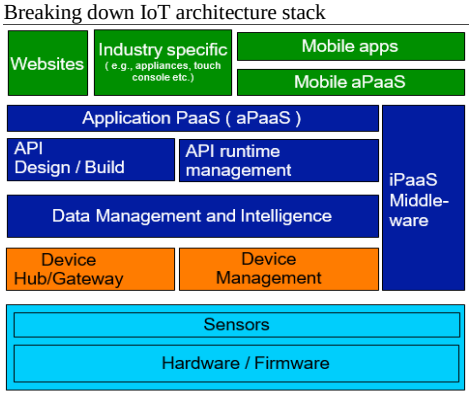
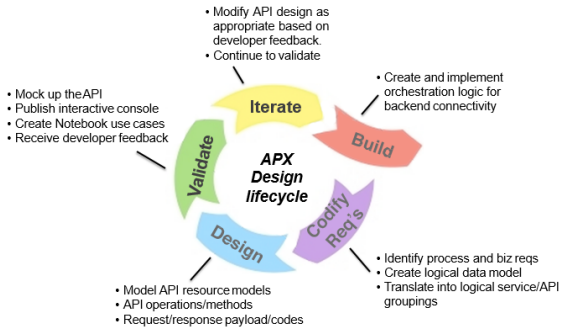
1. Connecting the physical world to the Web (Environmental, connected car, social&local, building management, home automation, personalized, logistic&shipping, identity&tracking, farming, energy grid, healthcare)
2. IoT stack arsitektur

App
Data processing dan platform
Edge
Thing/device

3. Breaking down IoT architecture stack

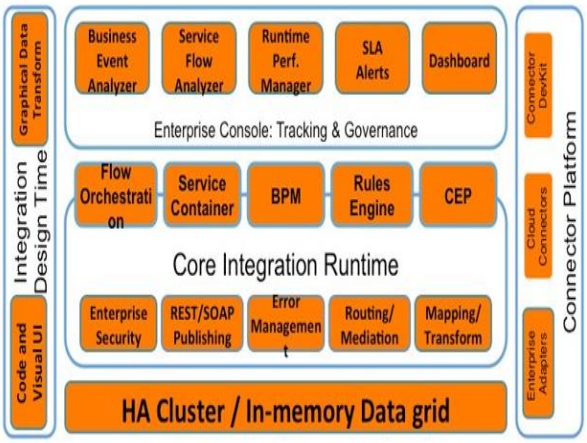


4. Device/thing, yaitu sensor dan hardware/firmware
   Device: banyak chipsets/pplatform yang menjadi pilihan.
   Contoh: ARM, Raspberry Pi, Qualcomm, Intel Galileo, AMD, MARVEL, MICROCHIP, electric imp, NVIDIA, Arduino
   Sensor dapat mendeteksi Suhu, radiasi, ozon, NOx, Noise, Kelembaban, humidity, GPS, CO, UV, CO2, PM, Glukosa, Oximetry, detak, temperature, kecepatan, dan lain-lain
   Kerjaan sensor pintar: onboarding, menerima notifikasi, menerima konfig, mengirim data/events
5. Device edge: device hub/gateway dan device management
   Kunci hak khususnya adalah untuk membangun dan mempertahankan koneksi aman, kuat, toleran kesalahan antara cloud dan perangkat yang berada di ujung (device dege) untuk:
- Mengumpulkan dan kumpulan data perangkat
- Mengelola perangkat
  Referensi kemampuan untuk sebuah gateway: memungkinkan scalable, real-time, dapat diandalkan, performa tinggi dan data dapat dioperasikan dan pertukaran relasi pengaturan device antara publisher dan subscriber.
  Tantangan: terlalu banyak perbedaan ekosistem. Terlalu banyak gateway, hub, protocol, app
  Solusi: butuh interoperabilitas antara device/mesin jadi mereka bias saling berkomunikasi
6. Data management and intelligence
   Kebutuhan kemampuan untuk data management dan inteligen
- Data collection, storage, and analysis of sensor data
- Run rules on data streams
- Trigger alerts
- Advanced analytics/machine learning
- Expose HTTP (REST) APIs
  Contoh: data, http, connectivity |pengkayaan data | pattern discovery/model re-training | proses peristiwa waktu nyata | routing dan orchestration | identifikasi penyebab akibat | proses batch | connectivity solusi bigdata | analisis prediktif
7. API lifecycle tooling dan platform: API runtime management
   dapat dibagi menjadi API desain/build:
- API desain lifecycle **(ADA GAMBAR)**



- API spec creation
- Reusable API pattern
- API mocking/modelling
- Deployment automation
  dan API runtime management
- Pembatasan tingkat/throttling
- Multi-tenant org/RBAC support
- API SLA management
- Deployment automation
- Custom policy engine
- API and data security

8. Application PaaS (aPaaS)
   Kemampuan: Host pada cloud, menyediakan platform untuk membangun aplikasi
   Contoh: OS/DB, storage, server, network | desain dan alat development | managemen dan alat analisis | routing, mengubah, layanan orchestration | web, database, server aplikasi, portal administrasi
9. End application: website industry specification mobile app dan mobile aPaaS adalah untuk driver tampilan mobile / table
10. Integrase iPaaS: middleware



IoT konsep yang setiap domain aplikasi tertentu berinteraksi dengan layanan domain independen, sedangkan di masing-masing domain sensor dan aktuatr berkomunikasi langsung dengan satu sama lain.
Arsitektur IoT adalah ilmu desain dan membangun struktur, layout, formasi, pengaturan.
Tipe arsitektur:

1. Tiga lapisan

Lapisan aplikasi	lapisan aplikasi menyediakan manajemen global data dari lapisan jaringan.
Lapisan jaringan	Lapisan Network juga bisa disebut 'Transmission Lapisan'. Lapisan ini aman mentransfer informasi dari perangkat sensor ke sistem pengolahan informasi. Dengan demikian, lapisan Jaringan mentransfer informasi dari lapisan Persepsi ke lapisan Middleware.
Lapisan persepsi	Lapisan Persepsi ini juga dikenal sebagai 'Device Lapisan'. Ini terdiri dari benda-benda fisik dan perangkat sensor. Lapisan ini pada dasarnya berkaitan dengan identifikasi dan koleksi benda-benda informasi spesifik oleh perangkat sensor. Informasi yang dikumpulkan kemudian diteruskan ke Jaringan lapisan untuk transmisi aman untuk sistem pengolahan informasi.

2. Berdasarkan middleware

Lapisan aplikasi		Menyediakan pengaturan global untuk aplikasi
Lapisan muddleware		Untuk mengatur layanan dan menghubungkan database
Lapisan koordinasi		Fasilitasi komunikasi antara aplikasi yang berbeda
Lapisan jaringan backbone		Bagian dari jaringan computer infrastruktur yang menghubungkan berbagai jaringan
Keberadaan system aplikasi sendiri	Lapisan akses	
	Teknologi ujung	

3. Arsitektur orientasi layanan

Aplikasi	Menyediakan layanan yang diminta kustomer
Komposisi layanan	Bertindak sebagai perasa sebenarnyasebagai repository dari semua service
Managemen layanan	Memasangkan layanan dengan permintaan berdasarkan alamat dan nama
Object abstraction	Transfer data yang dihasilkan oleh lapisan objek ke managemen layanan

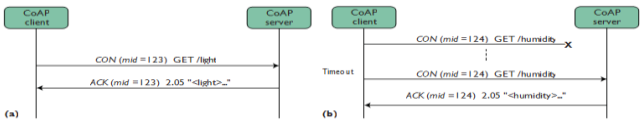
object	Menyajikan ulang sensor fisik yang bertujuan untuk mengumpulkan dan proses informasi

4. Lima tingkat

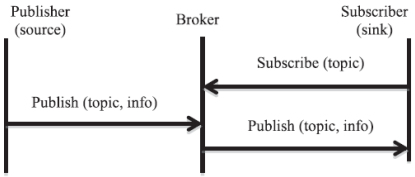
Lapisan bisnis	Mengatur keseluruhhan aktivitas IoT dan layanan. Kewajibannya adalah membangun model bisnis, graf, flowchart, dan lain-lain
Lapisan aplikasi	
Pengaturan layanan	
Object abstraction	
object	

Aplikasi layer protocol untuk IoT

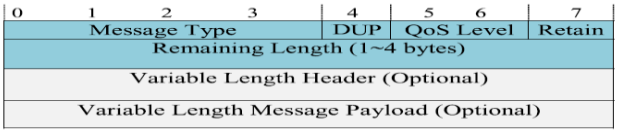
- Fungsionalitas: menawarkan aplcation programming interface (api) dengan fungsi builth-in untuk pengguna akhir juga dan memberikan pilihan untuk memantau dan mengontrol perangkat akhir dari jarak jauh untuk cliet; bertindak sebagai asynchronous node intermediate antara akhir-perangkat dan aplikasi akhir yang berjalan pada perangkat seperti smartphone, tablet dan desktop
- Protocol:
- DDS,
- CoAP, adalah protocol transfer untuk konsrain nodedan jaringan. Berbasis UDP dan terbebani oleh sejarah. CoAP bertujuan untuk mencapai tujuan sederhana dengan kerumitan lebih sedikit. Motivasi: 1) simplicity dengan tenaga lebih sedikit dengan membutuhkan akses internet menjadi simple; 2) kuat; 3) efisien energi; 4) mampu beroperasi dengan teknologi saat ini. Menggunakan UDP. Menggunakan metode GET, PUT, POST, DELETE. Code respon, seperti 4.04 artinya 4*32+04



- MQTT, adalah protocol open source untuk perangkat konstrain dan low-bandwidth, high-latency network. Memiliki transfer pesan publish/subscribe. Memiliki 3 komponen: publisher, broker, sunscriber. Menggunakan meknanisme routing one-to-one,one-to-many,many-to-may dan optimalkan M2M. Dibangun di atas TCP. QoS level: Fire and Forget(pesan dikirim seklai dan tanpa acknowledgemnt dibutuhkan), Deliverred at least once(pesan dikirim sekali dan acknowledgment dibutuhkan), Delivered exactly once(mekanisme four way handshake digunakan untuk meyakinkan pesan terkirim setidaknya sekali)



Format Pesan



- MQTT-SN, - XMPP,
- HTTP REST: arsitektur berupa web Sumber daya merupakan kunci untuk arsitektur Web: server dikendalikan abstraksi proses aplikasi membuat tersedia, diidentifikasi melalui URI. Klien mengakses sumber daya server yang dikendalikan secara sinkron permintaan-respon menggunakan metode seperti GET, PUT, POST, dan DELETE. Server memiliki keadaan asli dari sumber daya, dan akses ke representasi yang memungkinkan untuk caching, proksi-ing, dan mengarahkan permintaan dan tanggapan, memungkinkan interoperation mulus melalui proxy.Memiliki teknologi: HTML, HTTP/REST, URL. Code respon, seperti 404 not found

Komparasai antara protocol aplikasi IoT

Application Protocol	RESTful	Transport	Publish/ Subscribe	Request/ Response	Security	QoS	Header Size (Byte)
COAP	✓	UDP	✓	✓	DTLS	✓	4
MQTT	×	TCP	✓	×	SSL	✓	2
MQTT-SN	×	TCP	✓	×	SSL	✓	2
XMPP	×	TCP	✓	✓	SSL	×	-
AMQP	×	TCP	✓	×	SSL	✓	8
DDS	×	TCP UDP	✓	×	SSL DTLS	✓	-
HTTP	✓	TCP	×	✓	SSL	×	-

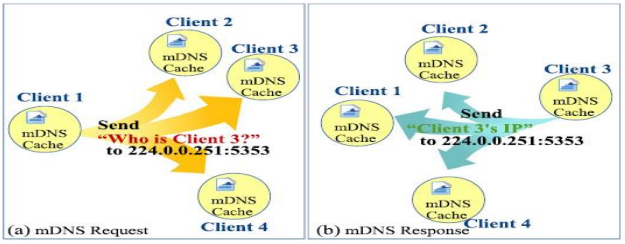
Standarisasi upaya dalam dukungan dari IoT

Application Protocol		DDS	CoAP	AMQP	MQTT	MQTT-SN	XMPP	HTTP REST
Service Discovery		mDNS				DNS-SD		
Infrastructure Protocols	Routing Protocol	RPL						
	Network Layer	6LoWPAN					IPv4/IPv6	
	Link Layer	IEEE 802.15.4						
	Physical/ Device Layer	LTE-A		EPCglobal		IEEE 802.15.4		Z-Wave
Influential Protocols		IEEE 1888.3, IPSec					IEEE 1905.1	

Layanan deteksi(Service Discovery Protocols(SDP)) adalah protocol jaringan yang dapat otomatis mendeteksi perangkat dan service (sensor) yang ditawarkan oleh perangkat pada jaringan komuter. Layanan penemuan membutuhkan bahasa yang sama untuk memungkinkan agen perangkat lunak untuk memanfaatkan salah satu layanan lain tanpa membutuhkan intervensi pengguna terus-menerus.

Protocol:

- Multicast DNS (Mdns): basis service berupa Name Resolution. Mdns dapat dilakukan pada tugas dari unicast DNS server. Kelemahan utama mDNS adalah kebutuhan untuk caching entri DNS terutama ketika datang perangkat untuk sumber daya yang terbatas. Namun, waktu cache untuk interval tertentu dapat memecahkan masalah ini. Bonjour dan Avahi adalah dua implementasi terkenal yang meliputi mDNS. mDNS bertanya nama dengan mengirim pesan IP multicast ke semua node dalam domain lokal seperti yang ditunjukkan pada Gambar tersebut. Dengan query ini, klien meminta perangkat yang memiliki nama yang diberikan untuk membalas kembali. Ketika mesin target menerima namanya, itu multicast pesan respon yang berisi alamat IP-nya. Semua perangkat dalam jaringan yang mendapatkan pesan respon memperbarui cache lokal mereka menggunakan nama yang diberikan dan alamat IP.



- DNS Service Discovery (DNS-SD): terdapat dua langkah pada proses Service Discovery. menemukan nama host dari layanan yang dibutuhkan seperti printer (fungsi pasangan dari layanan) dan pasangan alamat IP dengan nama host mereka menggunakan mDNS. Fungsi pasangan dari jasa yang dibutuhkan oleh klien disebut sebagai penemuan layanan berbasis DNS-(DNS-SD). DNS-SD, seperti mDNS, merupakan bagian dari bantuan konfigurasi nol untuk menghubungkan komputer tanpa administrasi eksternal dan konfigurasi. Kelemahan utama dari ini DNS-SD adalah kebutuhan untuk caching entri DNS terutama ketika datang perangkat untuk sumber daya yang terbatas. Namun, waktu cache untuk interval tertentu dapat memecahkan masalah ini. Bonjour dan Avahi adalah dua implementasi terkenal yang meliputi DNS-SD.