# Control System for everything: New and right way of building internet of things

## Intro

### The most important aspect: interoperability

There are many aspect that IoT must addresses such as: !!!.

The biggest challenge IoT is facing is interoperability.

Many effort seems been put into it. But little have been achieved. Different hardware, software vendors, and solution providers tend to use proprietary protocols for business or other reasons. Even if there is a protocol that everyone is willing to use there still are problems. Things do not have knowledge of each other. The data may be presented in different ways like temperature may be in Celsius or Fahrenheit. Things may talk to each other but may not understand each other. Semantic issues are much more complex then syntax issues.

### The current IoT approach is wrong (not a title)

### There is no good to give everything an IP address

### The current protocol are just **lies**

### New approach that focus on make everything interoperable in a feasible way

## Modeling of things

### A Model that applies to everything

This does not model every aspect of a thing, just focusing on interoperability in IoT application cases that is to say control system. Explain control here. In control system, the activities can be abstracted as: 1. Observe the status of various object; 2. Based on observation and predefined strategies generate commands; 3. Actuators get the command and take actions.

为什么要这么设计**？**

### Status

A thing may have a set of status. Status is a property that others can observe but cannot change. It can only be changed by a thing itself. It represent the current states of a thing that the thing wants others to know. All the information others needs to know should be made into status. And by looking at the statuses solely, one can determine all the output and behavior of the thing. Of course a thing may hide internal states that other don’t need. This is not status.

### Configuration

A thing may have a set of configurations. Configuration is a property that the thing it self cannot change but can only be changed by others. It represents how the others want the thing to behave, and the supported behaviors of this thing.

The effect of configuration changes will take place based on the things designated behavior. The above is to say configuration does not necessarily tells how a thing should behave, and others cannot determine the output of the thing by looking at the configurations (should by looking at status). Once a configuration changes, it may result in a transitional state of thing. It takes time for a thing to transit from one set of configurations to another. But the transition will definitely be presented in statutes. Moreover in some situations, one may want to change a number of configuration than let the change take place instead of change the behavior in the middle of these changes. So the thing may not apply these configuration change until a method is called.

### Method

A thing may have a set of methods. A method is a command like property, like a method of member function in OOP. A method is a command to a thing that it must react to it immediately. A method can accept parameter or not. The result of a method may depends on statuses and configuration. A method may or may not acknowledge the caller.

If you took a glance of the titles, you will find there is no event in this model. Yes event in this model is considered as an implicit call to a method of a thing. Why? In widely accepted design, event is often a publish/subscribe design. When the event publisher rises an event, the subscriber will get noticed and response by calling the event handler. You see? If there is no subscriber the event resulting nothing just like no event happened. If there is a subscriber an event always results in calling the –event handler- method. So in this model if a thing wants to rise an event it calls one or many methods on the subscriber things. One can make a publisher/subscriber thing that handle managing the method call chain and the event generators and handlers.

### Resource

Resource is a property of a thing. There are three types of resource: status, configuration and methods just described above.

A resource can be located and accessed. That is to say with a way one can uniquely refer to resource on a thing and using predefined way to access it, like read and write or call the method.

The three types of resource modeling is the very first step of interoperability. With this all the thing looks alike. M2M communication now falls down to locate and access resources.

You see the resource has 2 essential characteristics: uniquely locatable and has different ways of access. This make it looks very like some of the RESTful feature. Located by URI, and indicating way of access by using HTTP verb. This will be discussed in later sections.

这个设计有啥好处**？**

### World

Interoperability concept should not be unlimitedly expand. There are sets of things that will never need to interoperate. They belong to their own group of things that interoperate with each other and the groups may differ greatly. They are designed for non-related types of business. Although they all comply the three types of resource model, ways of implementing communication protocols to locate and access these resource may be quite different. Different protocol may have different feature that may be important to a certain business, like performance or interoperability among vast variety of thing. Most importantly there are absolutely no needs to enforce interoperability between those groups, it’s a total waste of effort to do that. So a concept of world is brought in. World is an enclosure, that everything inside it must be able to locate and access the resources on thing in this world. That means every thing in a world can interoperate. Moreover, things in different worlds do not enforce interoperability. So when talking about IoT, the first thing we need to do is to define our world. This does not mean things in a world use the same communication protocol. Brokers and gateways can help in this situation. But thing in a world do share some similarities. The similarities are in business aspect. They all has similar missions, goal, they need information of each other to finish the job. Brokers or gateways between worlds can be implemented but that’s mostly unnecessary, or implemented for limited function if one in a world would like to take a peek at another world.

For examples, the in traditional IoT concept, a thing may want anyone that is on the internet to talk to it. Than it belongs to a world, you can name it the internet world. But in another case M2M communication only happens to a relevantly small group of things. Like a treatment control system of a proton therapy machine.

World hierarchy…..??

World这个定义有啥好处？

### **What is a thing?**

A thing can be anything, as long as the world treat it as one self-contained object. A thing owns a set of status, configurations and methods. That means the above definitions apply to that object. Different things can be grouped and aggregate to a bigger thing.

A thing does not need to be physical, it can be a service, software as long as it complies the three types of resources described above.

## Dumb and Smart Things

One major reason IoT does not fly is... cost

Give everything an IP address is cost.

A protocol that is interoperable among vast variety of things are often heavy.

In one world, we can divide it to 2 level hierarchy and things into two categories. One talks with protocol with direct interoperability, one does not.

Smart things has enough computing power to run heavy protocol that talks to other smart device.

Dumb things just use simple protocol often not IP. Like….. to be accessed by or accessing others dumb things have to connect to smart things, and the smart thing will map them to the more complex protocol.

So smart things is a hub/gateway? Not only. As stated in 2.7 things can aggregate to bigger thing. Smart things does this job. When dumb things connected to smart things smart thing will not only adapt the dumb things resources to smart world protocol. It may also contain additional business logic that make the thing behave smart. It could aggregate some dumb things into a bigger thing and orchestra them to do a complex job they are not able to do as individuals. The smart thing may hide some of the dumb things’ resources and synthesis new ones. So when smart things connect to a smart thing they became parts of it. The aggregated thing in smart thing for dumb thing is called an avatar of the dumb thing.

Why doing so? To maintain interoperability among great numbers and types of things, it need a protocol that is easy to work with across all the platforms. This protocol is often heavy. Not everything can do that. So why not let the ones with more power to do more job. With Great Power Comes Great Responsibility. So smart things not only implement the heavier smart world protocol but also implement some dumb world protocols. Not only being a gateway, it also use its power to make the dumb thing act smarter.

## Dumb thing interoperability

Yes smart things are interoperable, and they can control dumb things. But does that means every M2M communication have to go through the smart things? Imaging if flick a switch to turn on a lamp has to go through a hub, that delay and overhead and reliability is way worse than pre-computer era. So in fig. ? we let the dumb thing to talk to its dumb friends. They use simple old protocols. This communication is hidden. No smart things need to know this. But this communication will either invoke a method on another dumb things or change its configurations which results in change of status. This is known by the smart things. So a set of dumb things that has hidden communications are aggregated on smart things as one thing. The hidden communication does not mean the smart things cannot understand it, it means it does not contain direct reflect a status configuration of this thing.

### Dumb protocol pattern

MQTT and CoAP is not dumb and not interoperable.

Let from the nature world. We need a very simple easy to implement protocol.

This is not a protocol it’s a pattern.

**Language:**

If it speaks a language you doesn’t understand, than it is considered as noise.

Here language means if you can receive and parse this message. This is a wide definition. Like media, transport protocol and encryption they all falls into language zone.

**Identity:**

If you can hear and understand the language, then you can process this information. You immediately checks if it is of your interests. If you are involved in a conversation the first thing you check is if it is talking to you. If not you can ignore it. Then who send the message is also important since multiple object may involve.

Here identity means you extract the sender and designated receiver of the message. If it is send to you than you can further process the message. And one good thing to notice is you now can extract the sender identity, this is critical as well.

**Knowledge:**

The last thing in knowledge. If someone talks about quantum physics to me, I would have now chance of understanding it, even I know every single word.

Here knowledge means, you can understand from that message about the status, configuration and method, and update the resource in the smart things avatar. This not just you can parse the resource name and value from it, it means you understand it. Like if the smart thing is hosting an avatar of a thermometer, it understand a “temperature is 22 degree” but will not understand “voltage is 55V”. it does not have the knowledge of status named voltage.

You see the above does not define a protocol, but designed a design pattern for design and using the protocol. Many protocols can be adapted to this pattern just using semantics ways.

### Bindings

To bind a dumb thing to its avatar, the smart thing must have the correct language. This is simply done by load the right network stack.

For identity and resource binding, there are 3 ways.

**Static:**

The avatar it preconfigure and knows the identity of the dumb things, and have the knowledge of the resource on the dumb things.

**Interactive:**

The avatar is not preconfigured. But when the dumb thing request a bind the avatar is created and the smart device can be recognized the dumb device and then establish the knowledge from pre-configurations or looking up a knowledge base but not from the dumb thing.

**Dynamic:** **(not required in recent versions, but consider supporting it should not introduce breaking changes)**

The avatar is not preconfigured. When the dumb thing request a bind the avatar the avatar is created and the smart device cannot be recognized. The smart thing cannot establish the knowledge base. It then negotiate with the dumb thing to establish the knowledge.

## Web of things

When talking about IoT, internet is the key. But there are already lots of things connected to the internet but interoperability is not achieved. But the web is an exception. Grab any device it is very likely it has access to resource in web. Browsers are compatible with all kinds of applications and more applications has transformed into web.

The fundamental of web is HTTP and HTML. We will leverage on these.

Scalability!

### RESTful

Resource located by URI.

Status will just support get verb.

Configuration support post put and delete, post and delete are for configurations that is a collection. Whereas the put support all kinds of configurations. Put will be the common case. You can use get to get the current value of the configuration.

Method will support post. This will call the method once. Method may support get this will retrieve the metadata of this resource. In fact all resource should support get.

Metadata is presented using hypermedia. Using JSON-LD is the trend. Using the same protocol is just first few steps of interoperability, understanding is the next step. This requires knowledge base. Semantic web is the key. JSON-LD is a preferred way of implementing this.

### HTML and JS

This is not for M2M commutation. Any control system application will not miss the HMI. Using web HMI is not a separate part. With HTML and JS, we can build SPA that has the same UX as client side running UI application. Aside from normal web goodness (more here), one good thing is with above metadata HMI can be generated automatically and dynamically.

## Implementations

### Smart things

Core is a hub that let others access resource one the same machine.

Resources sample are value object.

Core support plug-ins to modify the samples during a resource access.

Thing is called things.

Web, RESTful is implemented as things, the web thing support view expensing.

Not just web other protocol like dds can be implemented as a thing.

Avatar is things and created by things that create and manage other things.

### Dumb things

**Extreme simple:**

nRF24L01+ Arduino (attiny85 45 24) or nRF24LE1

Key fob remote and a RGP lamp.

**All-in-one but simple:**

ESP8266 single chip sensor and actuator node

**Cheap yet powerful:**

Arduino + Communication node