



**OPEN** CONNECTIVITY  
FOUNDATION™

# BEYOND CONNECTIVITY: IOT PROGRAMMING & DATA MODELING

Shao-Wen Yang, Ph.D.

Staff Research Scientist

Intel Labs, Intel Corporation



# Executive Summary

- What are beyond connectivity?
  - Making *SENSE* of the data
  - Making *USE* of the data

# DATA MODELING

Making *SENSE* of the data

# These Differences Serve No One



- Manufacturers have to include ALL the right protocols
- Service providers have to choose a single ecosystem or build their own proprietary solution
- Customers have to choose a single ecosystem and can't choose products "outside of the plan"





# How should the IoT work?

Creation of new devices should scale at Internet speed

- New interfaces should take minutes to develop, not months

All ecosystems and devices should work together

- The device maker shouldn't worry about being isolated by a technology choice

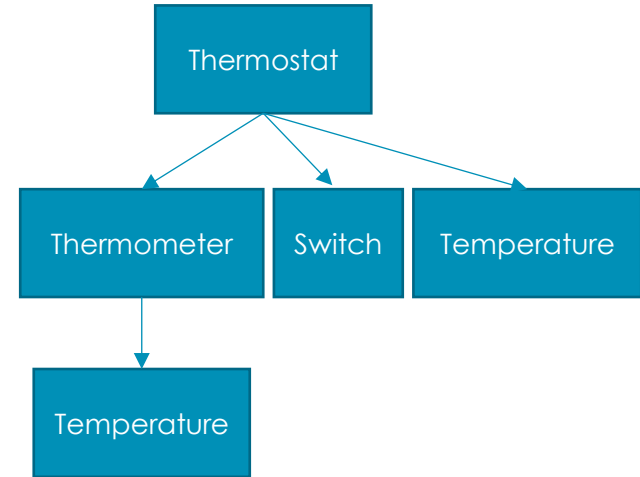
Verification should be simple

- A great idea shouldn't be hampered by its origin or an unnecessarily lengthy process

# THE CONSTRUCTIVE DEVICE DATA MODEL (SCALES AT INTERNET SPEED)



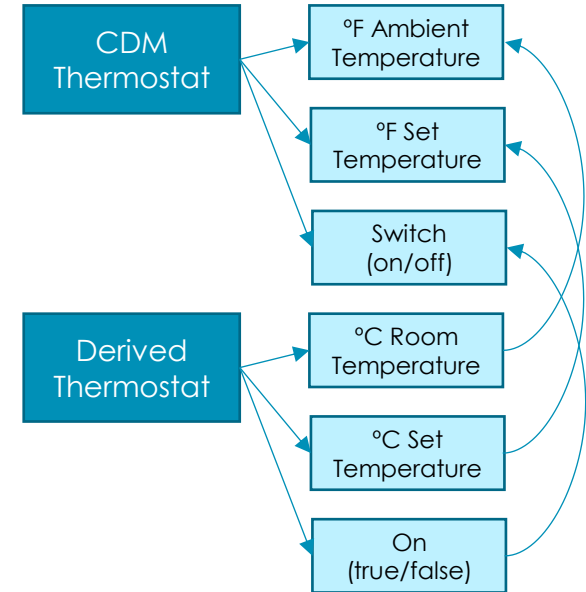
- Choose a generic description strategy (e.g. RAML, JSON schemas)
- Start with physical properties (e.g. temperature, mass)
- All new devices are defined as collections of physical properties and previously defined devices (e.g. a thermostat is a collection of temperature, thermometer and switch)
- Abstract devices can also be defined (e.g. Clarke's house, upstairs bedrooms)



# THE DERIVED DEVICE DATA MODEL (ALL ECOSYSTEMS WORK TOGETHER)



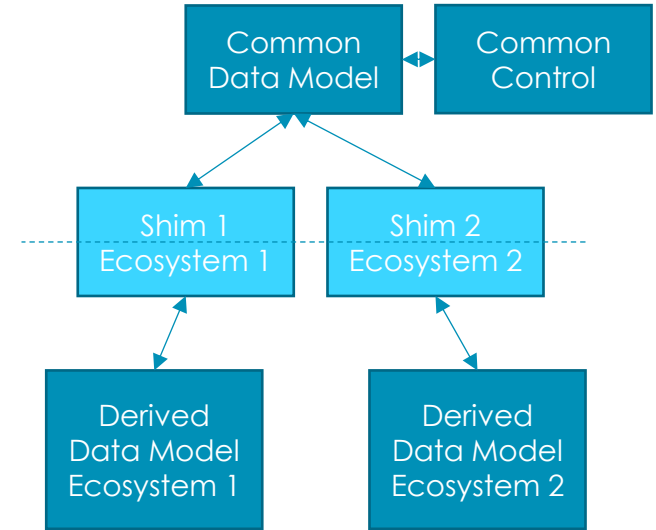
- ALL interoperable devices are defined exactly once in the common data model (CDM)
- Devices defined in other ecosystems (AllSeen, UPnP, etc.) are derived from devices in the common data model
- The definition of derived devices allows for differences in ecosystems (property names, variable types, range differences and conversions)



# THE DERIVED DEVICE DATA MODEL (CONT.) (ALL ECOSYSTEMS WORK TOGETHER)



- In operation, a shim layer (code stubs automatically generated from the device data model) provides for conversion between ecosystems
- Since all ecosystems derive from the common data model, there are at most two conversions
- The conversion can happen in a gateway, in the cloud or in end devices





# THE ONEIOTA TOOL (VERIFICATION IS SIMPLE)



- A crowd-sourced Integrated Development Environment (IDE) for the Internet of Things device models (oneloTa.org)
- RAML & JSON validated and syntax aware editors with shared editing
- Automatic support for derived models and multiple organizations
- Submission and approval process per organization

Indicates the proposal is in pending status awaiting approval by the organizations "Manager."

RAML files must reference at least one JSON\_schema. Clicking this link will bring the corresponding file to the edit window

"Reviews" are able to Approve, Reject, or make edit to a file to make it compliant for their specific technology.

The screenshot shows the OneIoT interface with the following details:

- Top navigation: All Models (63), Proposals (1), Releases (0), Organizations (0), Users (17)
- Model Name: activityCount.raml
- Versions: 20 Jan 2016 (PENDING)
- References: [oic.r.sensor.activity.count.json](#)
- Proposal: PENDING
- ORGANIZATION: UPnP
- Actions: Approve, Reject
- FILES list: activityCount.raml, atmosphericPressure.raml, audio.raml, autofocus.raml, automaticDocumentFeeder.raml, button.raml, carbonDioxide.raml, chestXray.konopedia.raml
- RAML Editor (Right):

```
1 #RAML 0.8
2 title: OICActivityCount
3 version: v1.0-20150727
4 schemas:
5   - Count: !include oic.r.sensor.activity.count.json
6 traits:
7   - interface:
8     queryParameters:
9       if:
10         enum: ["oic.if.s","oic.if.a"]
11
12 /ActivityCountResURI:
13   description: |
14     This resource specifies an activity count.
15     The resource can be readonly (oic.if.s interface) in a
16     The resource can be readwrite (oic.if.a interface) in a
17     The count property is an integer representing either a
18
19 displayName: Activity Count
20 is: [ interface ]
21
22 get:
23   description: |
24     Retrieves the current activity count.
25   responses:
26     200:
27       body:
28         application/json:
29           schema: Count
30           example: |
31             {
32               "rt": "oic.r.sensor.activity.count",
33               "id": "unique_example_id",
34               "count": 2500
35             }
36
37 post:
38   description: |
```



# Intel Labs – Research

**SIMPLE**

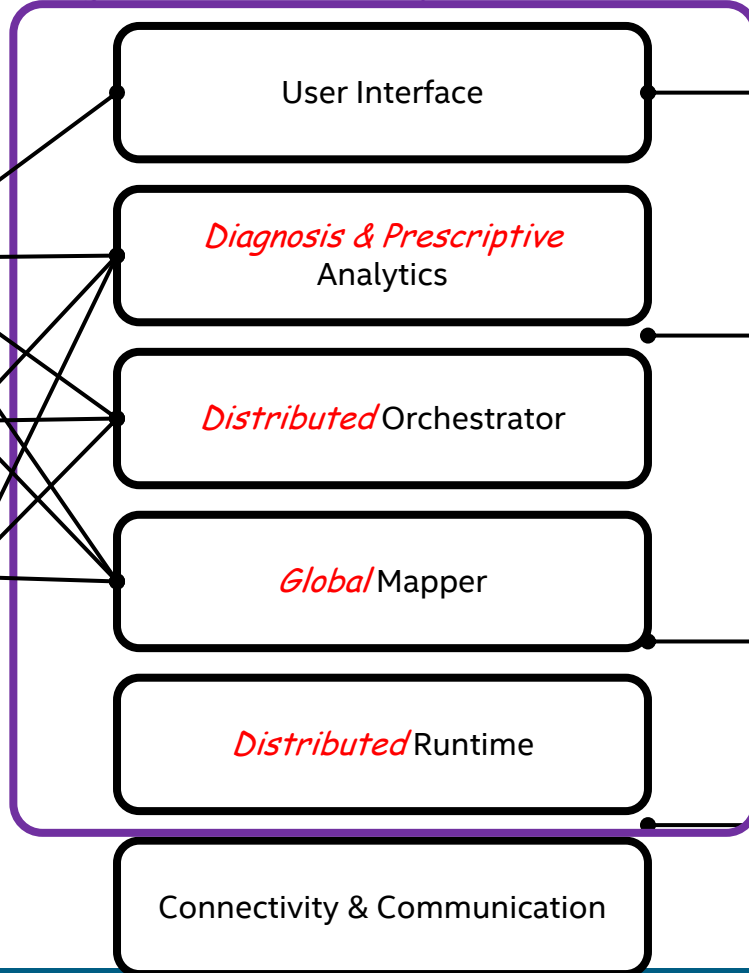
"Write-once-deploy-everywhere"

**RESILIENT**

"Deploy-once-run-forever"

**EFFICIENT**

*Design Consideration: Usability for end-users/installers*



User Interface

*Diagnosis & Prescriptive Analytics*

*Distributed Orchestrator*

*Global Mapper*

*Distributed Runtime*

Connectivity & Communication

**Programming Model**

- **Syndrome:** IOT systems are hardly programmable
- **Root-cause:** Most IOT users are end-users
- **Gap:** Current programming models are not at the proper level of abstraction for IOT
- **New insights:** Abstraction layers for "write-once-deploy-everywhere" and "deploy-once-run-forever"

**Anomaly Detection for Auto-Reconfiguration**

- **Syndrome:** IOT systems are unreliable
- **Root-cause:** IOT devices are usually resource constrained
- **Gap:** The prior art is mostly centralized
- **New insights:** Exchanging models rather than continuous large volume data

**Maximal Bipartite Matching**

**Missing Data Imputation**

- **Syndrome:** IOT data can be intermittent
- **Root-cause:** IOT devices are diverse and may operate in harsh environments
- **Gap:** The prior art assumed a share variance
- **New insights:** Variance-aware collaborative filtering

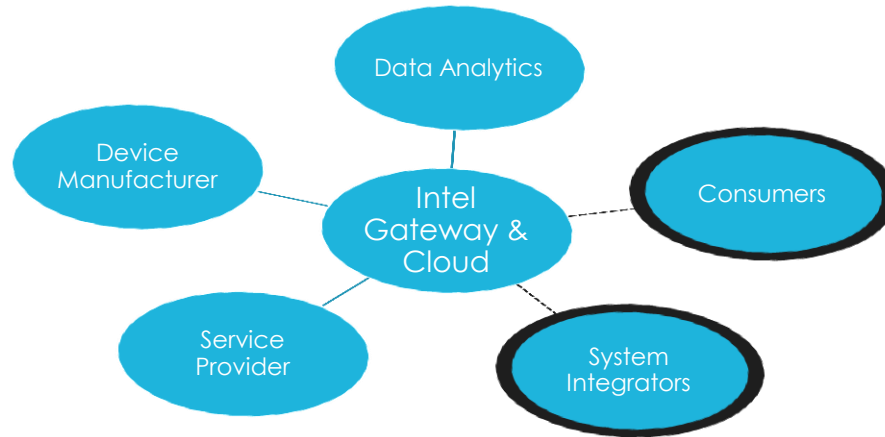
# PROGRAMMING MODELING

Making *USE* of the data

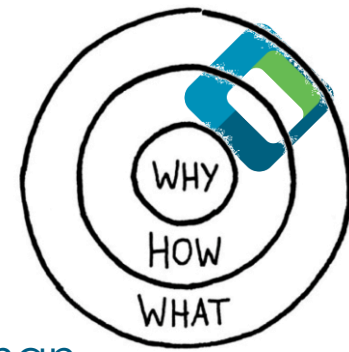


# Programming Paradigm Shift from PC/Internet to IOT

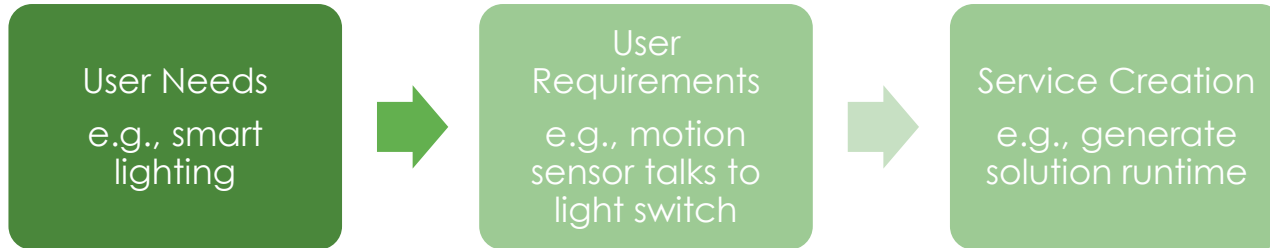
- Help close the gaps between User Convenience and System Efficiency in IOT
  - How to program IOT at scale? Write-once-run-everywhere?
  - How to deploy IOT at scale? Deploy-once-run-forever?



# Scope & Key Impact



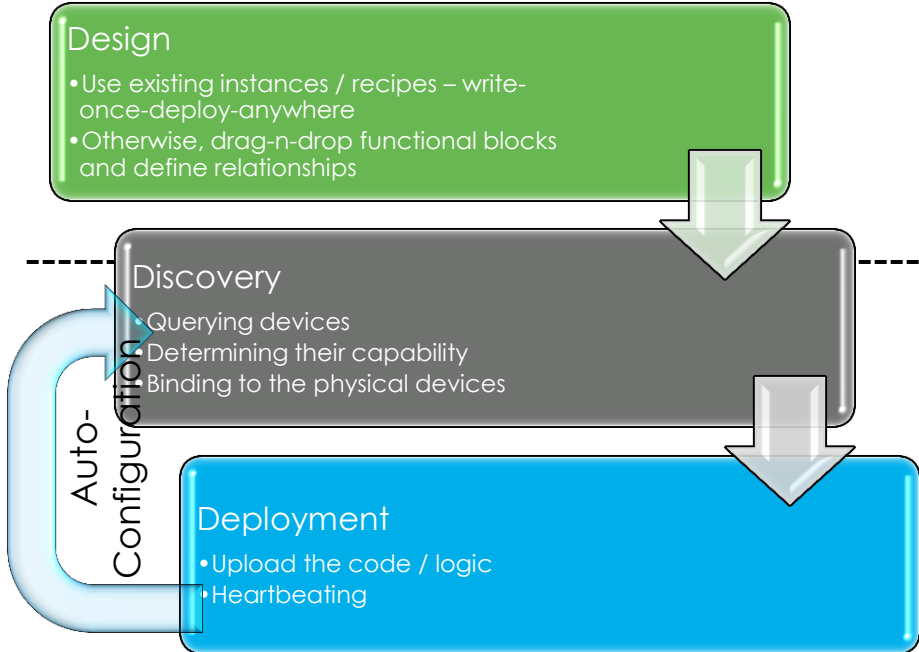
- Scope:
  - Why? 20B devices are hardly possibly manageable by human operators.
  - How? Bridge the gaps between needs and reqs, and reqs and deployment.



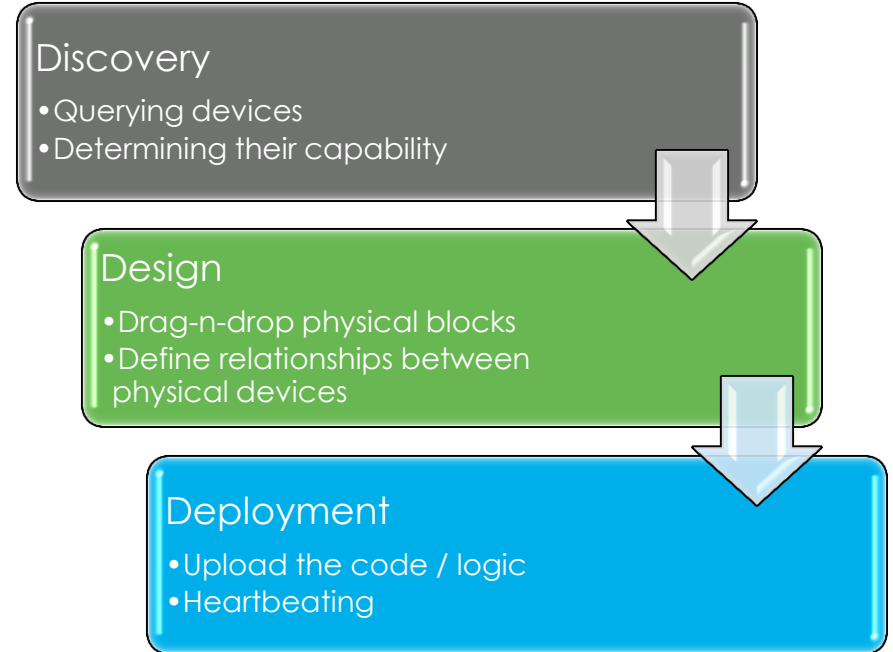
- What? Abstraction makes services portable, reusable and sharable.



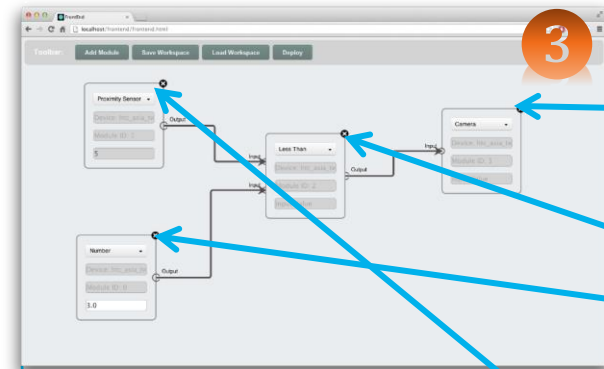
## Abstracted Programming Modeling



## State of the Art



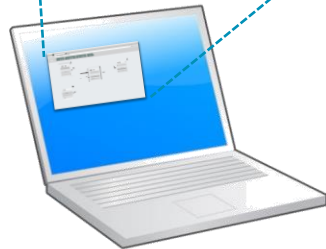




3 Deployment



1 Design



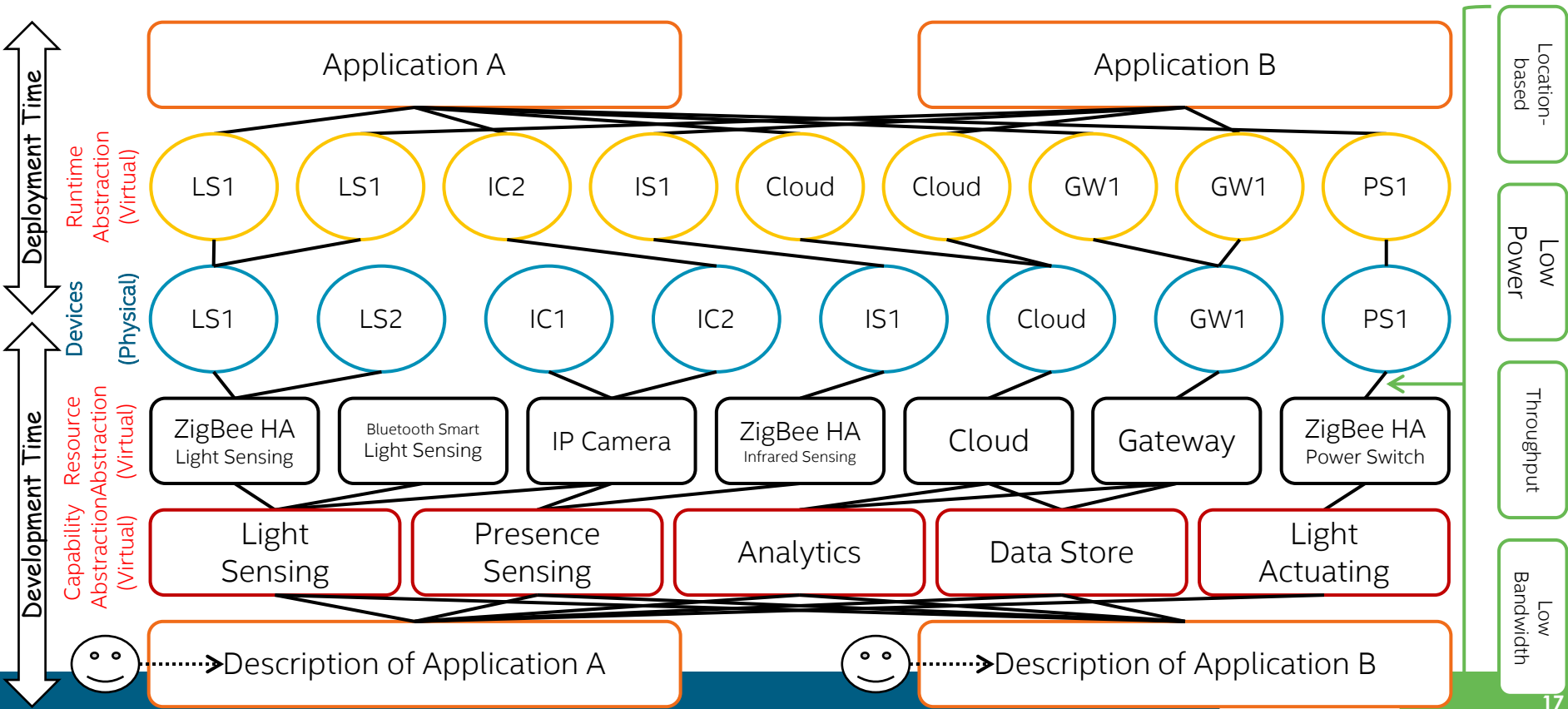
2 Discovery





# Abstraction Layers

## Software Defined Everything with Abstractions



# Visual Interface as an Example



■ Undeploy

▶ Deploy

Q Search

Input

123

Number

Light Sensor

Occupancy Sensor

Temperature Sensor

Output

Power Switch

Warning Device

Function

Less Than

And

Equal

Or

Sheet 1 deployed

Light Sensor

Number

Less Than

Temperature Sensor

Number

Equal

And

Power Switch



# Concluding Remarks

- What are beyond connectivity?
  - Making *SENSE* of the data for **interoperability**
    - Defragmenting IOT data/device models
    - Abstracting IOT data into semantically meaningful forms
  - Making *USE* of the data for **usability & reusability**
    - Improving IOT programmability
    - Enabling end-user programming
- Contact me at [shao-wen.yang@intel.com](mailto:shao-wen.yang@intel.com)



**OPEN** CONNECTIVITY  
FOUNDATION™