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

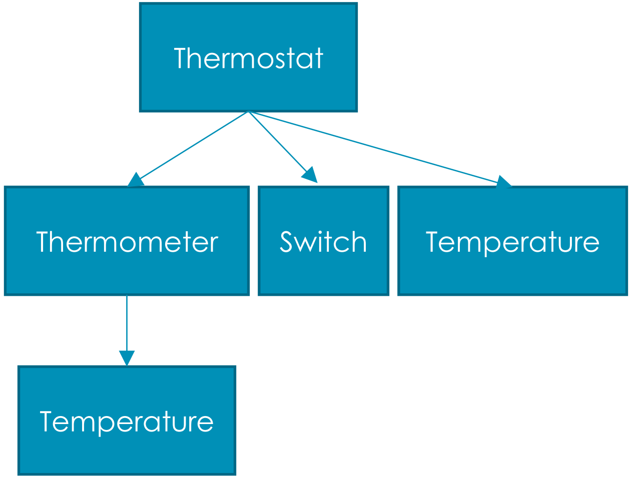
- The device maker shouldn’t worry about being isolated by a technology choice

Verification should be simple

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

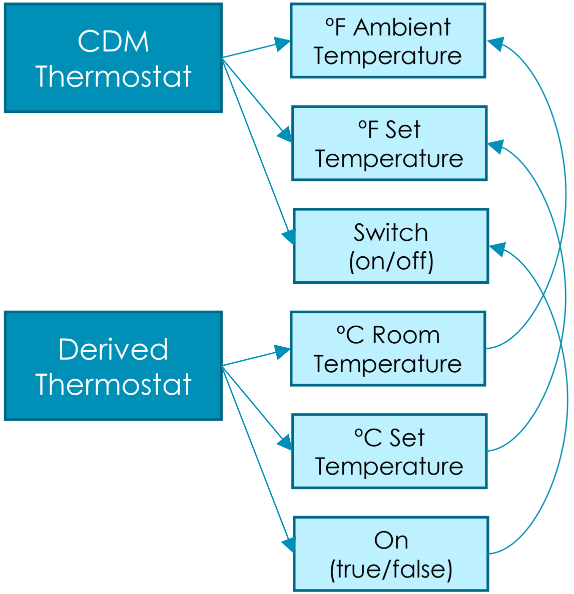
THE CONSTRUCTIVE DEVICE DATA MODEL (SCALES AT INTERNET SPEED)

- 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)



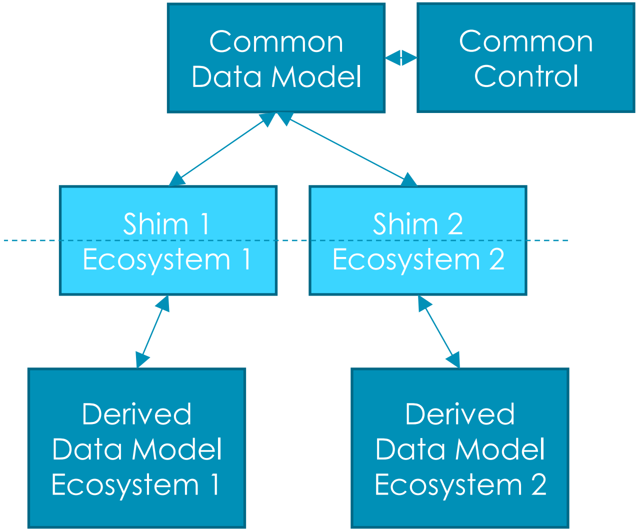
THE DERIVED DEVICE DATA MODEL (ALL ECOSYSTEMS WORK TOGETHER)

- 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)



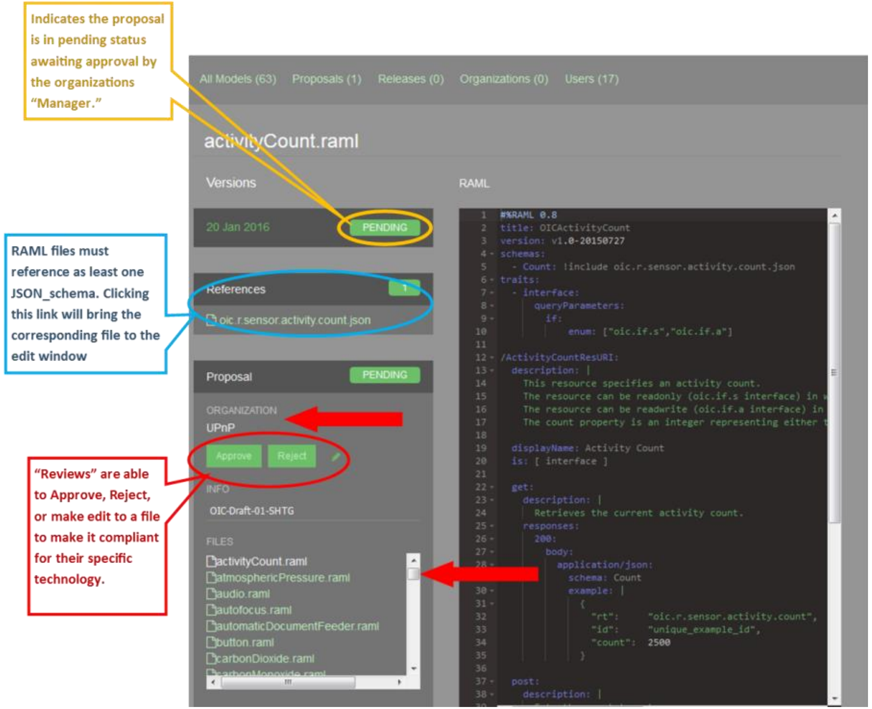
THE DERIVED DEVICE DATA MODEL (CONT.) (ALL ECOSYSTEMS WORK TOGETHER)

- 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

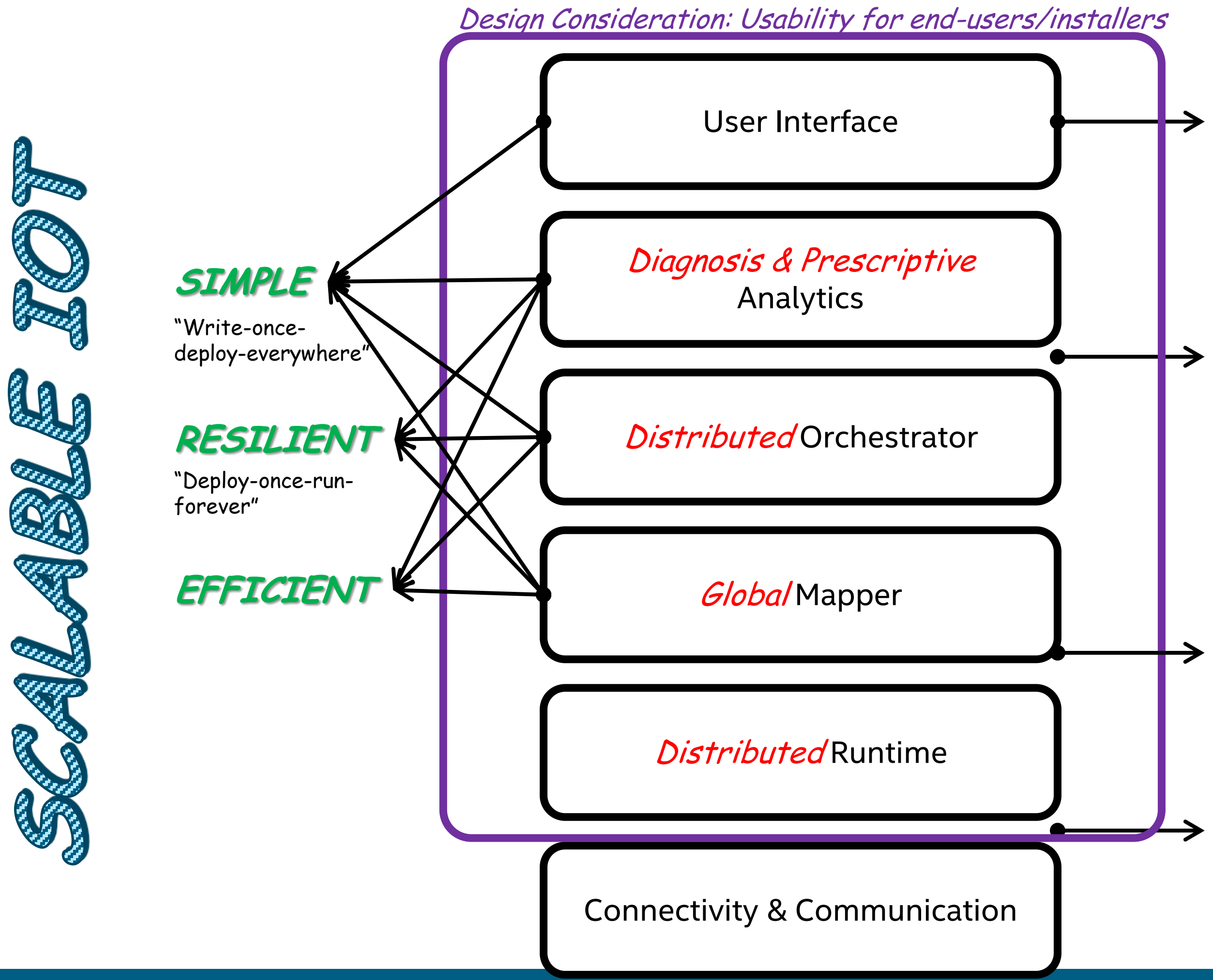


THE ONEIOTA TOOL
(VERIFICATION IS SIMPLE)

- A crowd-sourced Integrated Development Environment (IDE) for the Internet of Things device models (oneIoTa.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



Intel Labs – Research



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

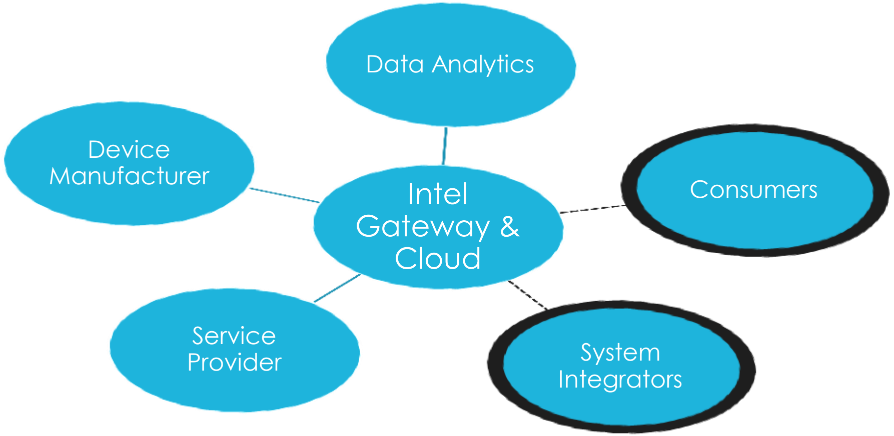
- **Syndrome:** IOT data can be intermittent
- **Root-cause:** IOT devices are diverse and may operate in harsh environments
- **Gap:** The prior art assumed a share variance
- **New insights:** Variance-aware collaborative filtering

PROGRAMMING MODELING

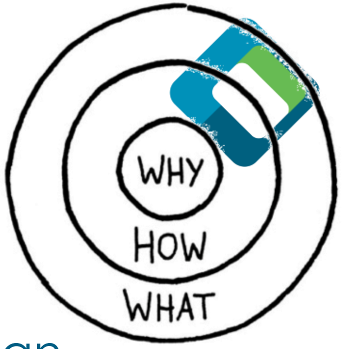
Making *USE* of the data

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.

User Needs

e.g., smart lighting

→

User Requirements

e.g., motion sensor talks to light switch

→

Service Creation

e.g., generate solution runtime

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

Methodology

Abstracted Programming Modeling

Design
- Use existing instances / recipes – write-once-deploy-anywhere
- Otherwise, drag-n-drop functional blocks and define relationships

Discovery
- Querying devices
- Determining their capability
- Binding to the physical devices

Deployment
- Upload the code / logic
- Heartbeating

Auto-Configuration

State of the Art

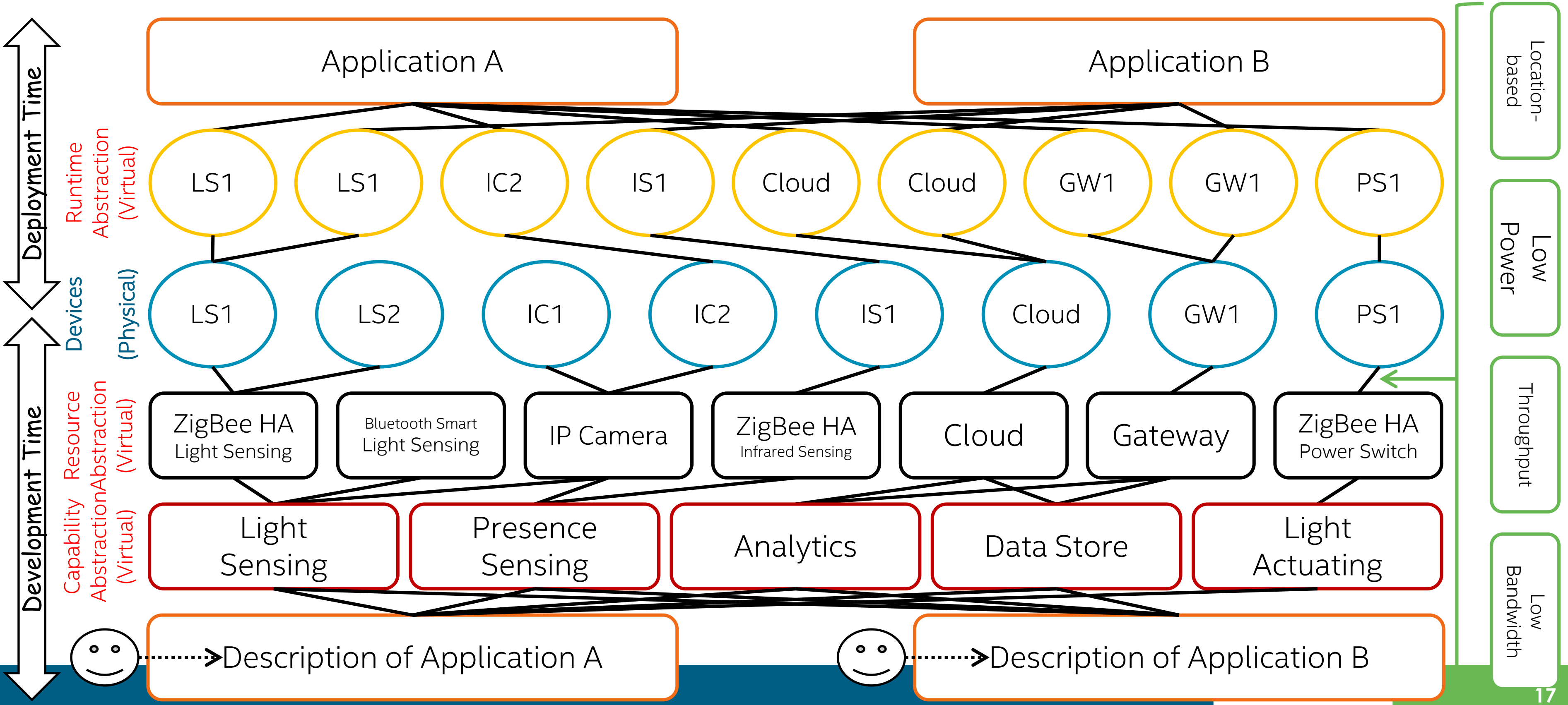
Discovery
- Querying devices
- Determining their capability

Design
- Drag-n-drop physical blocks
- Define relationships between physical devices

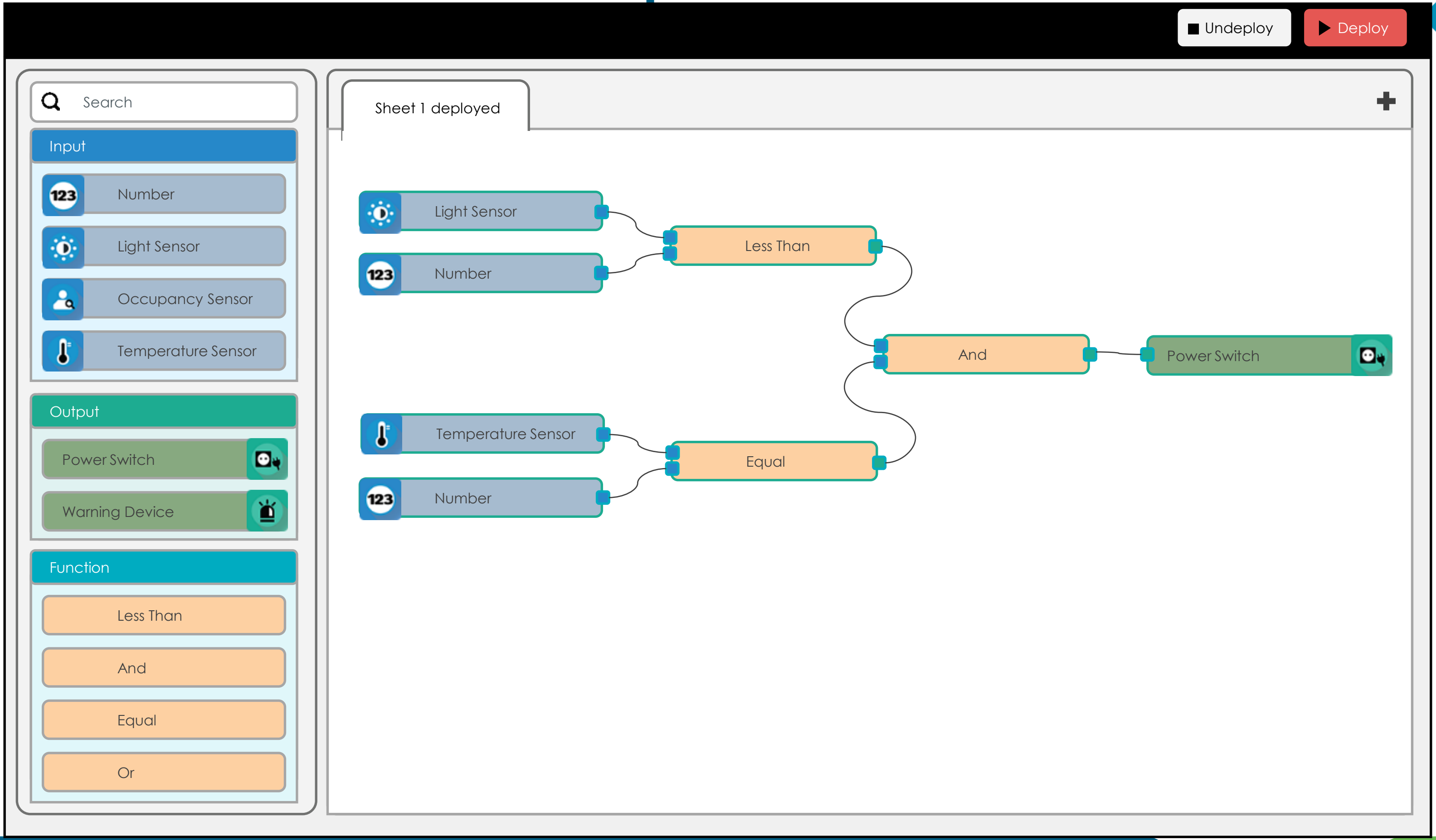
Deployment
- Upload the code / logic
- Heartbeating

3 Deployment
1 Design
2 Discovery
Toolbar:
Add Module
Save Workspace
Load Workspace
Deploy
Proximity Sensor
Less Than
Camera
Number
Output
Input
TX
LOW BAT.
HOME SENTINEL
SAMSUNG

Abstraction Layers
Software Defined Everything with Abstractions



Visual Interface as an Example



Concluding Remarks

- What are beyond connectivity?
  - Making *SENSE* of the data for **interoperability**
    - Defragmenting IOT data/device models
    - Abstracting IOT data into semantically meaningfully forms
  - Making *USE* of the data for **usability** & **reusability**
    - Improving IOT programmability
    - Enabling end-user programming

  - Contact me at shao-wen.yang@intel.com

OPEN CONNECTIVITY FOUNDATION™