Using the Raspberry Pi to Prototype the Industrial Internet of Things

Rich Blomseth, Glen Riley, and Bob Dolin  |  Oct 31, 2013
Presented at ARM TechCon, 2013
Agenda

- IoT Communication Models
- Requirements for an autonomous federation of peers
- Datapoints for a publisher/subscriber, data driven control model
- Organizing the network
- Python to the rescue!
- Demonstration hardware, code & Web pages
- Demo!
- How to get a free SDK
IoT Communication Models

• IoT has roots in asset management
  • Sensors sending data up to a database/server machine

• Generalized asset management model
  • Monitoring air/water quality, livestock, fleets of trucks, temperature of food, etc.

• Client/Server model is well suited to using IP

• The Industrial Internet is a distributed peer-to-peer model – not just client/server
  • Needed for distributed local control
  • IP protocols need extensions to handle multicasting, confirmed multicasting, secure multicasting, and more.
The Industrial Internet of Things Architecture

IP Control / Communities (IP-C)
Enabled common architecture
- Choice of SOCs
- Choice of media
- Routers
- Service platform
IzoT™ Platform – for the Industrial Internet

- Based on an open standard protocol with services added above the IP transport layer
- Every device is an equal peer
  - Any device can communicate with any other device, any group of devices, or all devices in the network – transparently to the application
- Devices communicate using datapoints
  - A simple data model keeps device applications independent of network topology, network addresses, etc.

IzoT is a trademark of Echelon Corporation
Datapoints

- IzoT devices communicate using *datapoints*
- A datapoint is a 1 to 228 byte value
- A device can have up to 4096 datapoints
- Sensors publish information based on events that occur and actuators subscribe to the information that applies to them
Standard Datapoint Types

- Facilitate communication among devices from diverse manufacturers
  
  - For example, the output from a light switch from one manufacturer is completely understood by the input controlling the lamp built by another manufacturer.

- User-defined datapoints can be developed by any developer, if required

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNVT_iot_alarm</td>
<td>Alarm</td>
</tr>
<tr>
<td>UNVT_iot_analog</td>
<td>Analog Value</td>
</tr>
<tr>
<td>UNVT_iot_load_control</td>
<td>Load Control Setting</td>
</tr>
</tbody>
</table>
Connecting Datapoints

- Datapoints are connected either by a network installation tool or by the devices themselves.
- Tools or the devices themselves can confirm that datapoint types are compatible.
Providing Reliable Message Delivery

- **Acknowledged Service**
  - Sending device requires acknowledgments (ACK) from all receiving devices
  - Automatic retries if ACK is not received during configurable period
  - Logs error (timeout) in the sending device upon failure of the message packet delivery
  - Reliable delivery

- **Request/Response**
  - Two-way service that behaves like acknowledged service
Other Message Delivery Services

- **Unacknowledged Service**
  - Only one update sent

- **Repeated Service**
  - One update sent, plus a configurable number of repeats
  - No acknowledgement sent
  - Conserves bandwidth in fan out connections
  - Probability of delivery with three retries is very high
ISI Protocol Key Features

• Fire-and-forget
• Network-wide bandwidth control
• Support for network growth
• Flexible connection model
Fire-and-forget

• Managed systems use a central server to allocate unique resources
• ISI uses the following patented algorithm to assign a resource called “Fire and Forget”
  1. Choose resource locally based on combination of random selection, predetermined algorithm, and local knowledge
  2. Consider chosen resource unique and use immediately
  3. Periodically inform other devices of resource usage
  4. A device receiving these resource usage notifications detects any conflicts and resolves conflicts locally using the same algorithm
• Fire-and-forget governs allocation of network addresses and datapoint connection information
Connection Model

- An ISI enrollment assembly is an application-specific, arbitrary, set of 1 – 254 datapoints
- Each device can support 0 – 254 assemblies
Three Methods to Make a Connection

• **Manual**
  • User presses a Connect button on each device
  • Suitable for devices with common device types in a network—like lights and switches

• **Automatic**
  • Devices automatically form connections
  • Suitable for unique devices like appliances

• **Controlled**
  • Similar to manual, but controlled from a user interface
  • Suitable for the same devices as the manual method
Creating a Manual Connection

Open Enrollment Message

ClosedLoopSensor

nvoValue

Bind

ClosedLoopActuator

nviValueFb

Connected

ClosedLoopActuator

nviValueFb

Connected

ClosedLoopActuator

Connected
Python to the Rescue!

- IzoT Python classes deal with much of the complexity of implementing datapoints, binding datapoints and creating web pages to display the data on the network

- Here is the code you would write to implement a sensor:

```python
from izot.resourcesprofiles.riotAnalogInput import iotAnalogInput

temperatureSensor = app.block(profile=iotAnalogInput(), ext_name='Temperature')
...

temperatureSensor.nvoAnalog.data.present_value = float(temperature)
```
IzoT Stacks

**Device Stack**
- User C/C++ Application
- User Python Application
- C API
- Python Package
- ISI Engine
- Control Services
- Linux UDP/IP Stack
- Ethernet/Wi-Fi

**Server Stack**
- REST
- REST Server
- Database
- Controller
- Collector
- Network Server
- Control Services
- Linux UDP/IP Stack
- Ethernet/Wi-Fi

- Free beta license
- Source code in C, C++, Python
- Sample code for applications and Web pages using the Raspberry Pi
Demo Hardware and Web Pages

- Adafruit Model 439 TSL2561 Digital Light Sensor
- Adafruit Model 189 Passive Infrared Sensor
- Adafruit Model 385 DHT22 Temperature-Humidity Sensor
- Adafruit Model 335 2.8” 320x240 18-bit Color TFT Touchscreen Display
- Adafruit Model 1083 ADS 1015 12-Bit 4-Channel ADC
- Solarbotics Model 60160 3 Watt RGB Star LED Controller

Environment Sensor

- Touch Keypad

IzoT Server

LED Controller

- Adafruit Model 815 16-Channel 12-Bit PWM Driver
Demo!

- Deploy the programs onto the Pis
- Use ISI to organize the network
- Test the network
- Interact with the network using the RESTful API
Now Please DO TRY THIS AT HOME!

• Download your free IzoT SDK at: http://iiot.echelon.com/get-started
• Buy your PI’s and the I/O that you want for your project
• Create some cool web pages
• ...and impress your friends!