IoT Mashups with the WoTKit

Mike Blackstock, Rodger Lea

Media and Graphics Interdisciplinary Centre
University of British Columbia
Vancouver, Canada
Motivation

- IoT mashups are simple, personal, situational, short lived applications
- Existing platforms often steep learning curve and complex tool chains
- Leveraging web protocols important step forward - *Web of Things*
- Web-centric IoT toolkits can increase pool of developers and applications.
Overview

• Other Toolkits
• Application Experience
• Requirements
• WoTKit
• Lessons Learned
Many Other Toolkits

- High end M2M systems
- Custom solutions
- Not focused on web, sharing
- Can be complex
- Similar web-centric toolkits
- Key differences
  - what is included
  - how services are delivered
Application Experience

• Health
  • Remote monitor of pulse oximeter
  • Connect heart rate monitor alerts when heart rate is high

• Environment
  • air quality data from web to display on mobile application
Application Experience

- Transport
  - vehicle location monitoring
  - location and transport mode
  - mobile traffic visualizations
  - ship locations
- IT, Social Networks, Home
  - Phidget, Arduino integration
  - Zigbee gateways
  - CPU monitors
  - Home and appliance power monitoring
  - Social network integration
Toolkit Requirements

• Integration
• Meta-data and Storage
• Visualizations
• Control
• Sharing
• Processing and Alerts
• APIs
WoTKit

- Platform and Mashup Services
- Sensor manager and aggregator
- Visualizations
- Finding & sharing sensors
- Processing and alerts

all are core system facilities, not plug ins or add ons.
Architecture

Management and visualization UI

• Processing engine

• RESTful API

• Shared data model

• Database and message broker
Integration and Sharing

• Easy integration using HTTP
• 2 way – sense and control
• Sharing meta data and sensor data
• Saves integration task for others
• Allows easy search and creation of mashups
• Provide services for connecting ‘islands of things’.
Visualizations

• First task after integration: *see your data*

• Widgets on dashboards
  • Google Charts
  • Maps
  • jQuery plugins
Processing and Alerts

• Inspired by Yahoo Pipes
• Users generate pipe descriptions by dragging and dropping modules and wires
• Input, output, processing, user scripting, debugging, testing, integration
Processing Engine

- Pipe descriptions are ‘compiled’
- Instantiates pipe modules in the server (checking for scripting and configuration errors)
- Routing table of module connections
- Messages from sensors, external systems are received by modules and added to queue.
- Modules may send output to downstream modules for processing, the WotKit, or to external systems
Lessons Learned

• Data Schema and Representations
• Sharing
• Component Model
• Sensor Updates and Processing Model
• Batteries included
Schema and Representations

Important to be simple and flexible

• e.g. location can be meta data or sensor data

• Numeric sensor values are not enough

• physical and virtual sensor feeds (tweets)

• Start with a very simple schema (timestamped value) that can be extended

• True for sensor data and meta data
Sharing

- Important to create network effect; increasing value of platform
- Public things, private things, group things
- Subscribe to favourite sensors
- Tagging for easier search
- Share via social networks
Component Model

- Dashboard widgets
- visuals + sensors
- Pipes and modules
- modules - input, output and processing
- Sufficiently different to warrant separate integration points
Sensor Updates and Processing Model

- **Push (Event):** firewall friendly, but can store and process data no one is interested in

- **Pull (Query):** infrastructure polling for unnecessary updates, real time alerting not as timely

- Both current state *and* stream of historical values are needed by applications

- Both updates and processing models are required.
Batteries Included

- If users are able to easily capture and visualize data, they will invest time in exploring other toolkit capabilities.
- From there, explore data processing, more visualizations, actuators, etc.
- Provide RESTful API and integration points for more complex applications and contributions to platform.
Conclusions

- IoT applications can be complex
  - Many IoT toolkits but can have a steep learning curve
  - Require sophisticated middleware and development toolchains.
  - For mashups, rapid development environments are needed:
    - Use familiar web technologies and tools
    - Quick ability to find and visualize things
    - Mashup a variety of data from things and data sources
    - Provide path to more complex applications
Thank you

• Give it a try!
• http://wotkit.sensetecnic.com
• Academic version available
• Commercialized by Sense Tecnic Systems
• mblackst@magic.ubc.ca