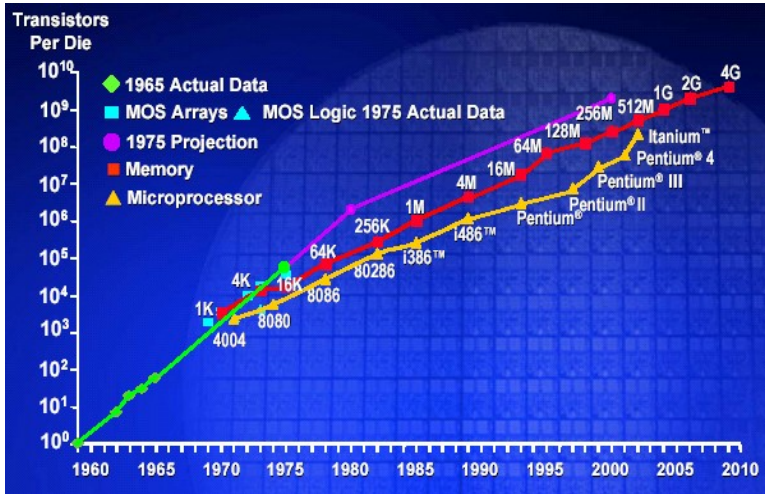


Building Sensing Applications with the Owl Platform

- » Presented by Richard Martin
- » And more, including
 - » Robert Moore, Bernhard Firner,
 - » Yanyong Zhang, Richard Howard,
 - » Eitan Fenson, and many others

The Opportunity

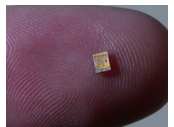
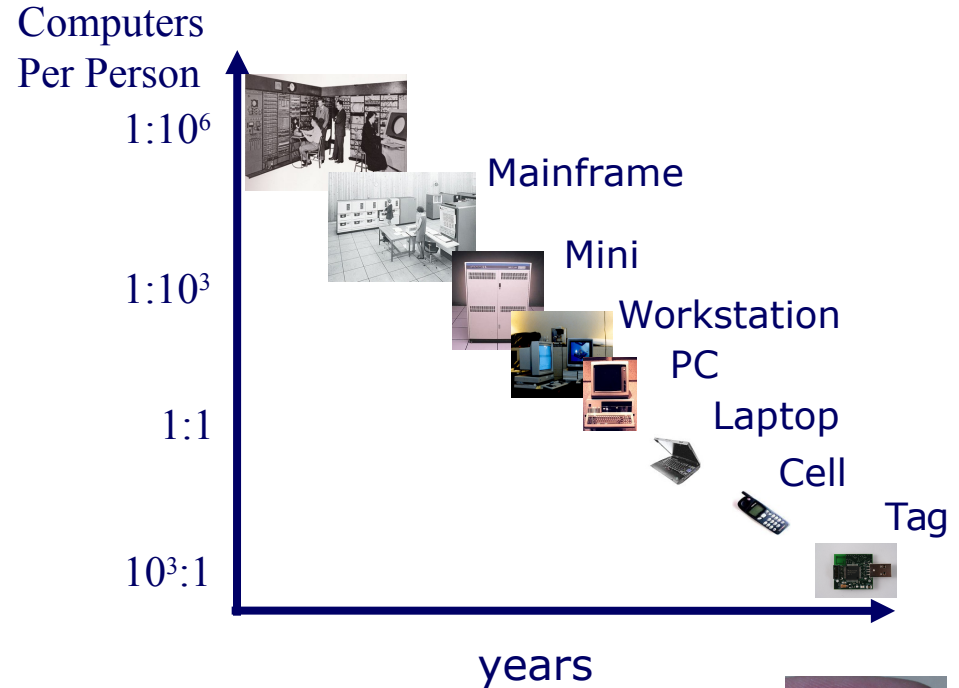
Moore's Law: # transistors on cost-effective chip doubles every 18 months



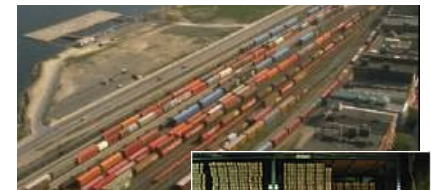
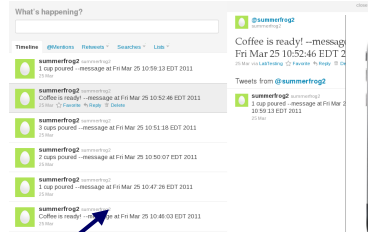
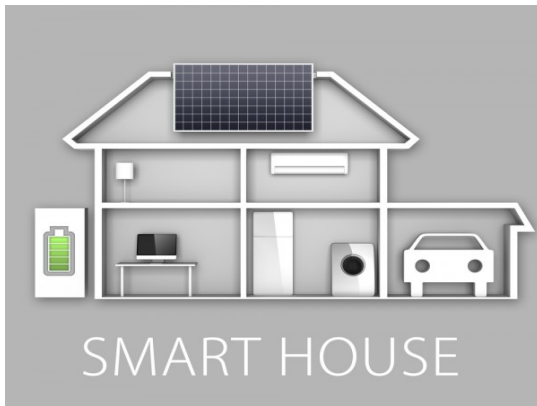
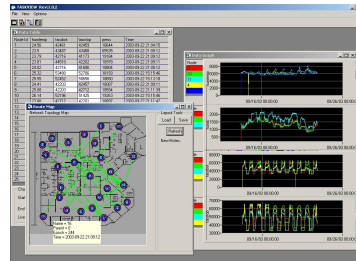
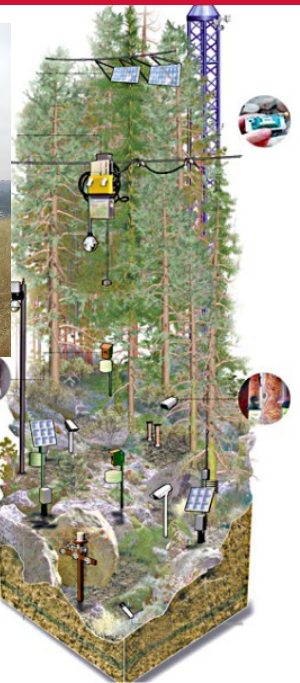
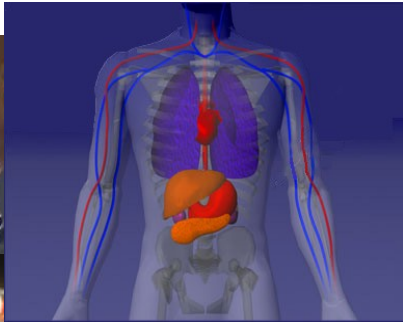
Today: 1 million transistors per \$

Same fabrication technology provides CMOS radios for communication and micro-sensors

Bell's Law: a new computer class emerges every 10 years



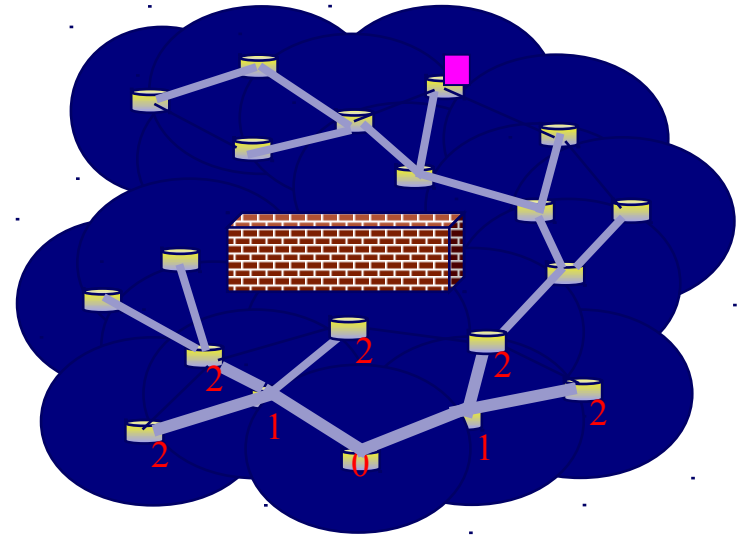
The Vision



Down the garden path of sensor networks

Programming a sensor network:

- Multi-hop
- Ad-hoc
- Aggregation and compression
- Energy conservation of whole application is paramount
- Novel operating systems, programming languages and environments



A rose by any other name

- 1999 Smart Dust
- 2000 Sensor Networks
- 2004 Internet of Things
- 2005 Ambient Intelligence
- 2009 Swarms

- ~15 years on, we still have not realized the vision.

What happened?

Problems

- Problems people talked about:
 - Energy conservation
 - Scaling number of sensors
 - Efficiency of code data size in small sensors
 - Routing
- More meaningful problems:
 - Too expensive for application domains
 - Difficult to develop applications
 - Can't re-use infrastructure
 - Not general purpose

Owl Platform

Novel constraint:

Enable application development by undergraduate level programmers

Standard languages, programming environments

Separation of concerns:

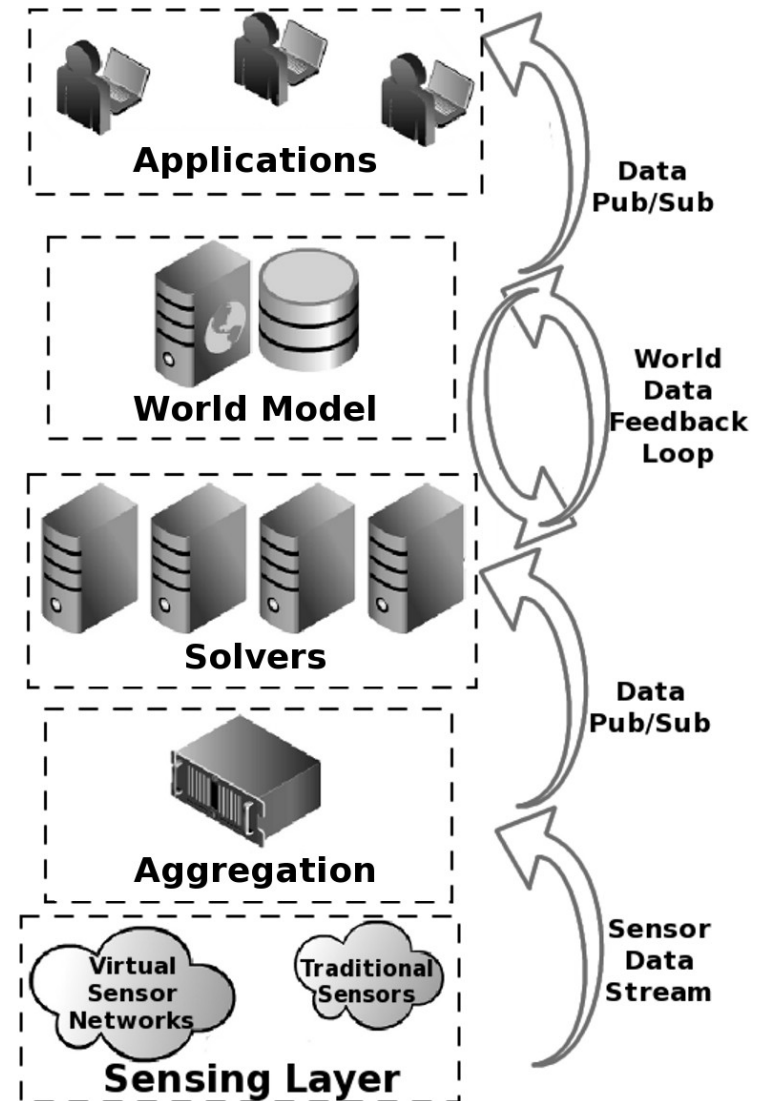
- Application developer: what is the data and app logic?
- Sensor designer: hardware and interface to aggregation layer
- System administrator: keeping the system running

Solves energy and scaling issues differently

- Move to the sensor designer level, leave the app out of it.

A Different Model: Layers

- Sensors connect to an intermediate layer that hides details
- Solvers build higher-level representations from low-level ones
- A uniform model of the world allows sharing
- Applications run in standard environments in the cloud



Layered Model – cont.

Layering allows for separation of concerns

- Sensor designers
- Deployment/ IT staff
- Solver/Algorithm developers
- Application developers

Each layer exports interfaces and methods

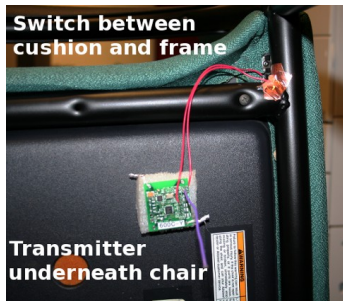
- libraries for different languages (Java, C++, Ruby)

Components communicate by state in the aggregator and world models.

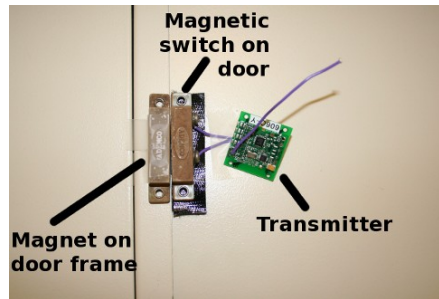
- components (solvers, application) use network
- allows for proprietary solvers, open source system

Sensor examples

Chair occupancy



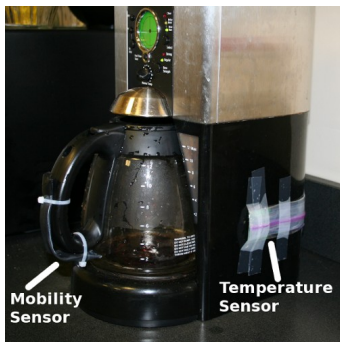
Door open/close



Kinect Skeleton



Coffeepot Temperature



Power Consumption



Phone Tracking



Owl sensor model

I want to add sensed data. What do I do?

An adaptation layer puts the sensor data into this format:

```
Physical layer, Source Sensor ID, Target ID, Time, Signal  
Level, Sensed Data
```

Sends the above over the Internet to the aggregator

World model

World server holds a model of the world

- Shared state between applications
- Partitioning of the name space between applications

The world is a hierarchical name space of variables

- Similar to LDAP, Windows Registry, SNMP MIB
- Balance of structure, open-ness

Variables have types, times, and an origin

World Model Data Format

Enter URI filter pattern:

Enter a new URI:

Object URI	Attribute Name	Origin	Data	Created	Expires
▶ Halloween2011					
▶ region					
▼ test temperature					
test temperature	creation	Ben	0x	1319731453082	0
test temperature	sensor.temperature	Ben	1.26	1319731470449	0
test temperature	location.uri	grail/discriminator solver	Halloween2011.locations.Ben's Desk	1319764031065	0
test temperature	temperature	grail/temperature solver	19	1319763017202	0
test temperature	mobility	grail/mobility_solver	0x00	1319763659777	0
test temperature	Add New Attribute				
▶ test uri					

World Model Data Format – cont.

Object URI name:

Example: edu.rutgers.owl.makefaire.keys

Attributes/Data: - similar to fields in a data structure

Attribute is a string, data is binary

Examples:

- Mobility 0/1
- Person1 : XY points of a kinect skeleton

Origin: - who or what created the data

Could match to a public key – not done yet

Created/Expires:

When was the data created, and how long is it valid?

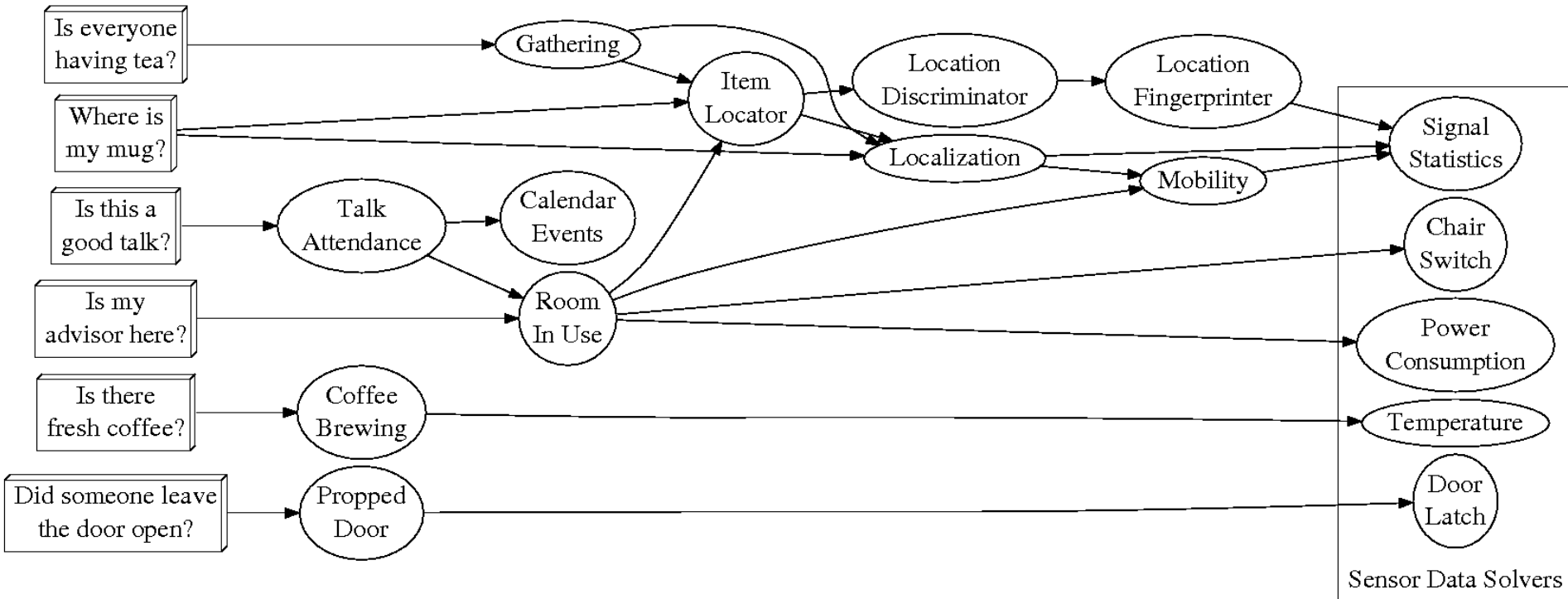
Owl solver chains

Semantic
Meaning

High level
Solvers

Low level
Solvers

Sensor
Data



Owl application patterns

SMS/Email Alerts

Status Maps



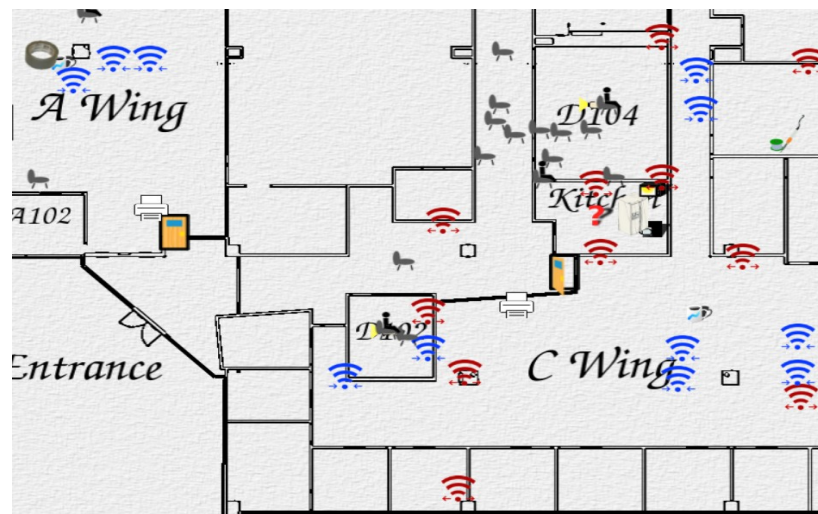
Owl 平台演示

当前状态

最近状态 - 全部传感器

<p>Office 提示 最近状态</p>	<p>2013-6-1 上午11:34:12 A-Wing门关闭.</p>
<p>C-Wing 提示 最近状态</p>	<p>2013-6-1 上午1:44:54 Office门关闭.</p>
<p>Leak Detector 提示 最近状态</p>	<p>2013-5-31 下午11:33:54 Main Door门关闭.</p>
<p>Filing Cabinet 提示 最近状态</p>	<p>2013-5-31 上午11:57:46 C-Wing门关闭.</p>
<p>A-Wing 提示 最近状态</p>	<p>2013-5-30 下午11:49:52 Filing Cabinet门关闭.</p>
<p>Main Door 提示 最近状态</p>	<p>2013-5-28 下午2:31:49 Leak Detector检测干燥.</p>

最近一次更新: 2013-6-12 下午12:56:23



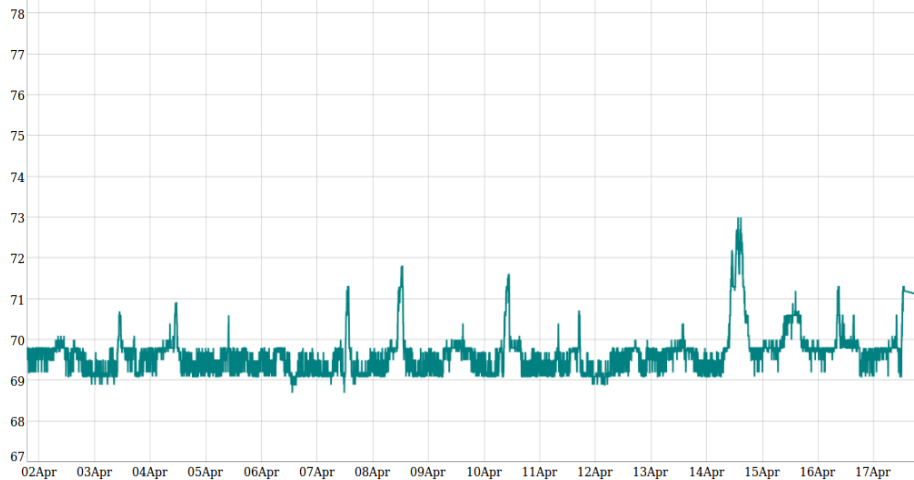
Example Owl application patterns- cont.

Reports

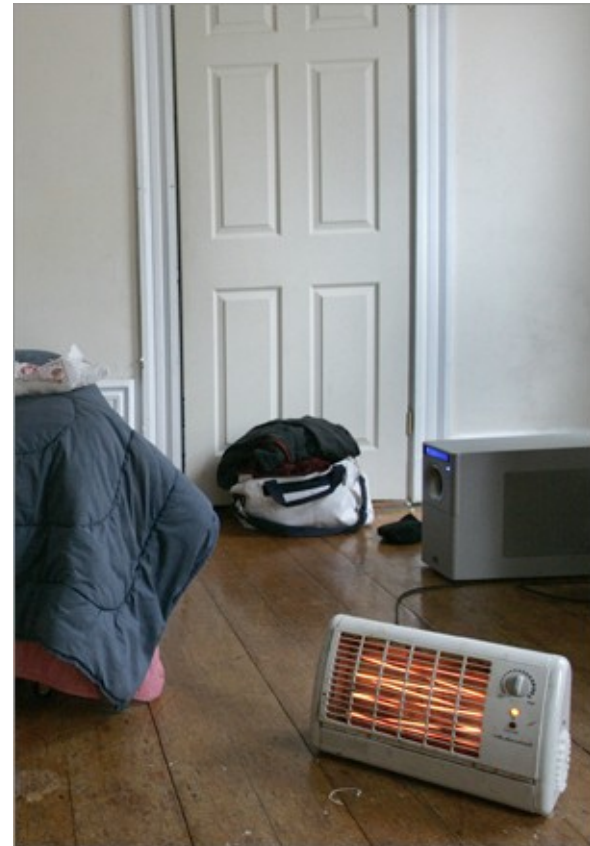
Temperature for Room 141 (New) - 4 Weeks

[1 Hour](#) [12 Hours](#) [24 Hours](#) [7 Days](#) [4 Weeks](#)

Click and drag to zoom. Shift-click to scroll left and right. Double-click to reset zoom.



Physical Actuation



Deployed App: Home Monitoring

Owl Platform Online | Current Status | Sensor History | Settings | Account | Logout

Sensor Status

Toy Owl | History | Settings

32.0°F | Changed 3 days ago

Front Door | History | Settings

68.0°F | Changed 2 days ago

Chair | History | Settings

32.0°F | Changed 21 hours ago

Water Heater | History | Settings

Sensor signal lost.

71.6°F | Changed 11 days ago

Owl Platform Online | Current Status | **Sensor History** | Settings | Account | Logout

Sensor History

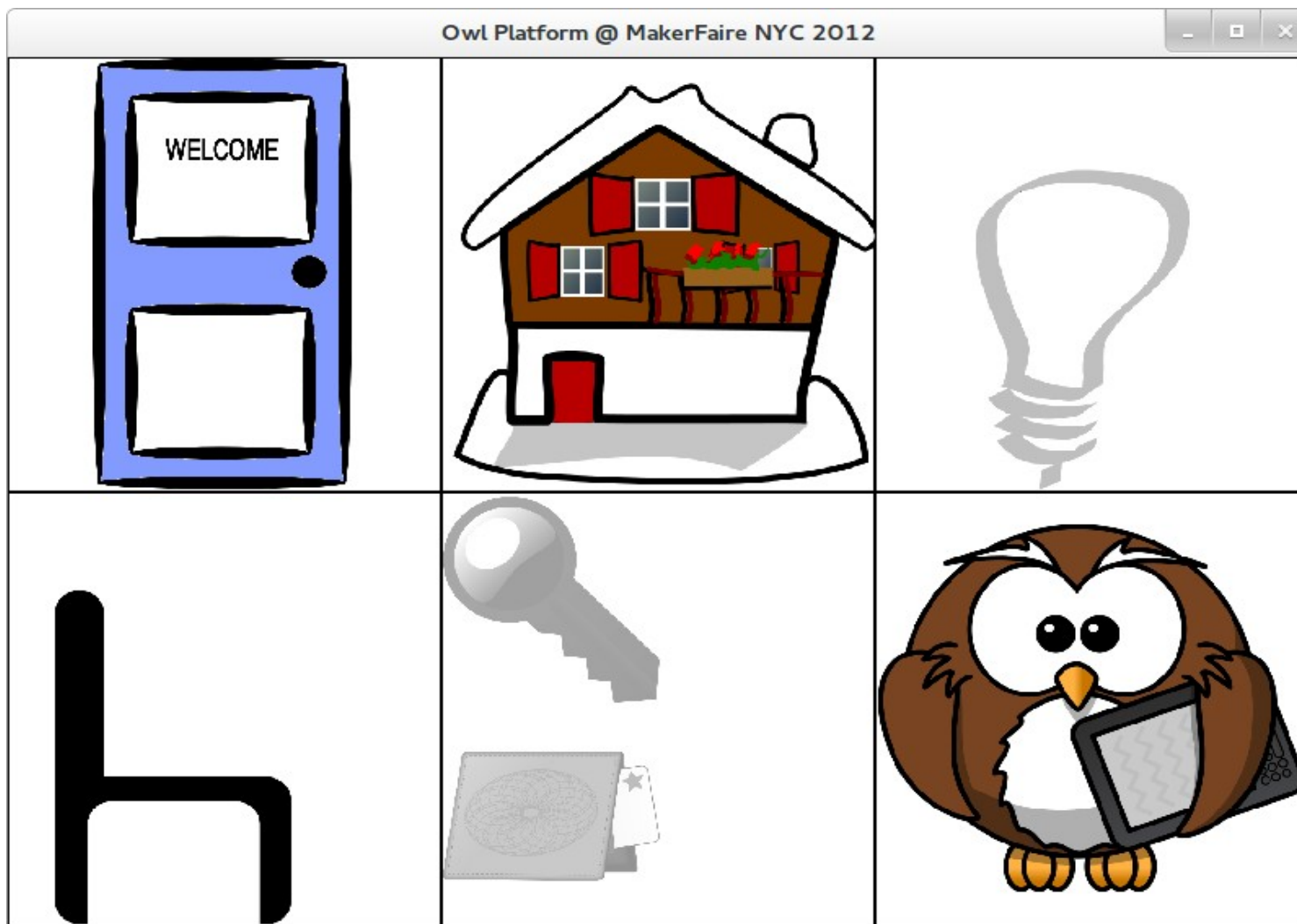
Sensor	Data Type
Toy Owl	Open Closed
Front Door	Open Closed
Chair	Open Closed
Water Heater	Wet Dry

Front Door Past 24 Hours

1

Timestamp	Event	Duration
2013-10-08 03:22:22 pm	open	2 seconds
2013-10-08 03:21:34 pm	open	4 seconds
2013-10-08 03:20:56 pm	open	3 seconds
2013-10-08 03:20:12 pm	open	1 second
2013-10-08 03:19:17 pm	open	26 seconds
2013-10-08 03:17:57 pm	open	a minute

Putting it all together: Demo Panels App



Putting it all together: Demo Panels App



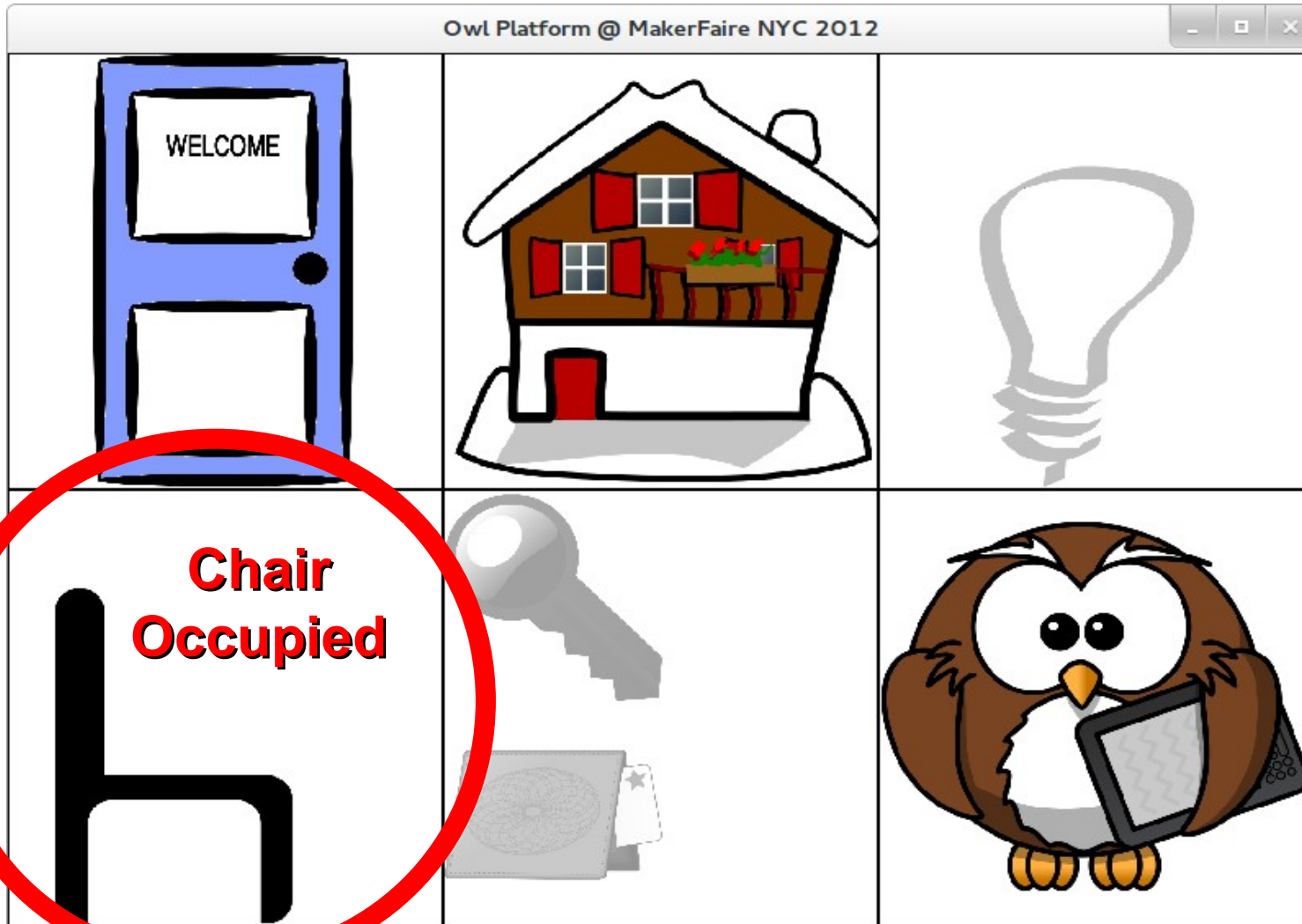
Putting it all together: Demo Panels App



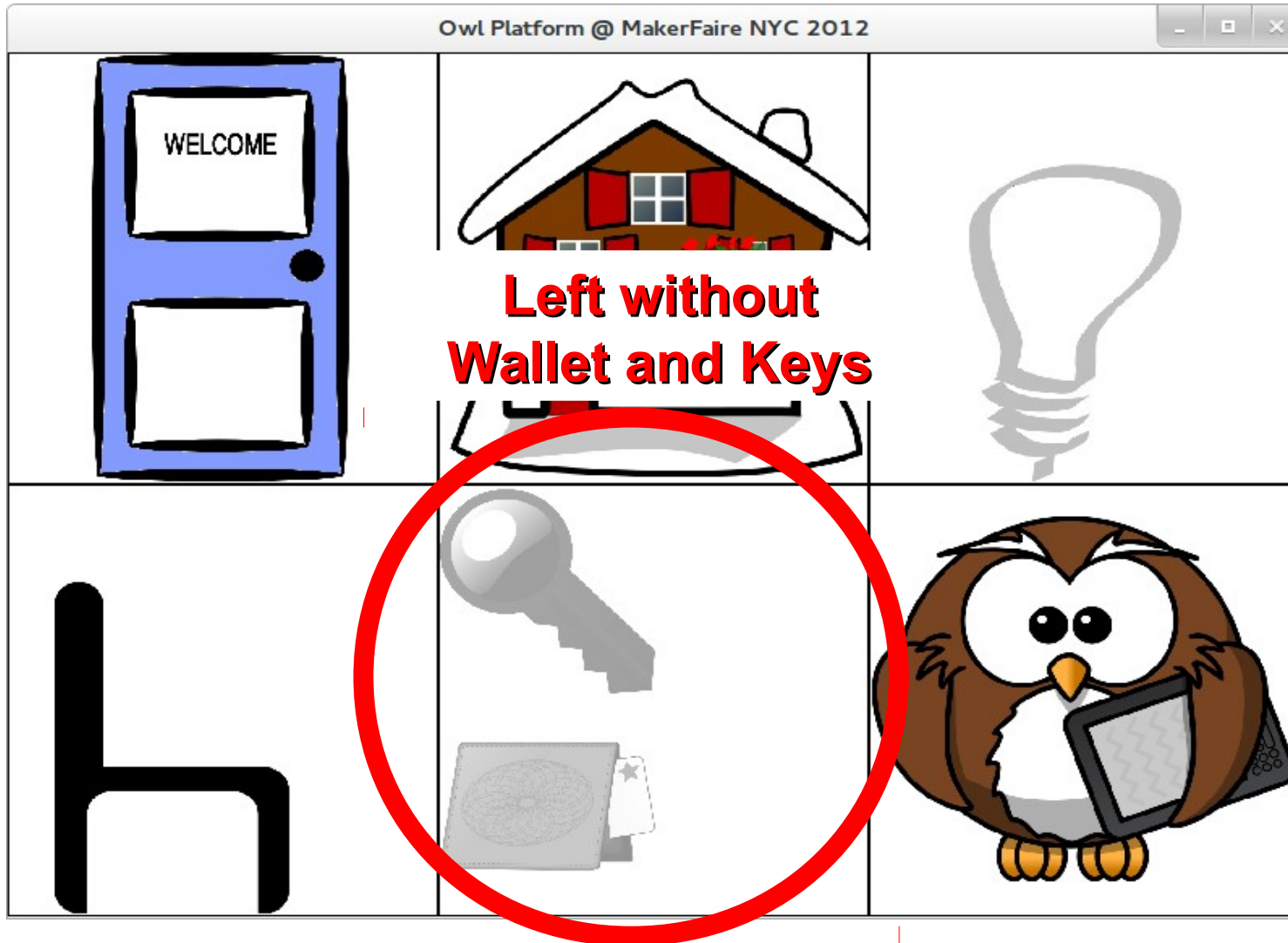
Putting it all together: Demo Panels App



Putting it all together: Demo Panels App



Putting it all together: Demo Panels App



Wallet and Keys application

When (the door changes state to open)

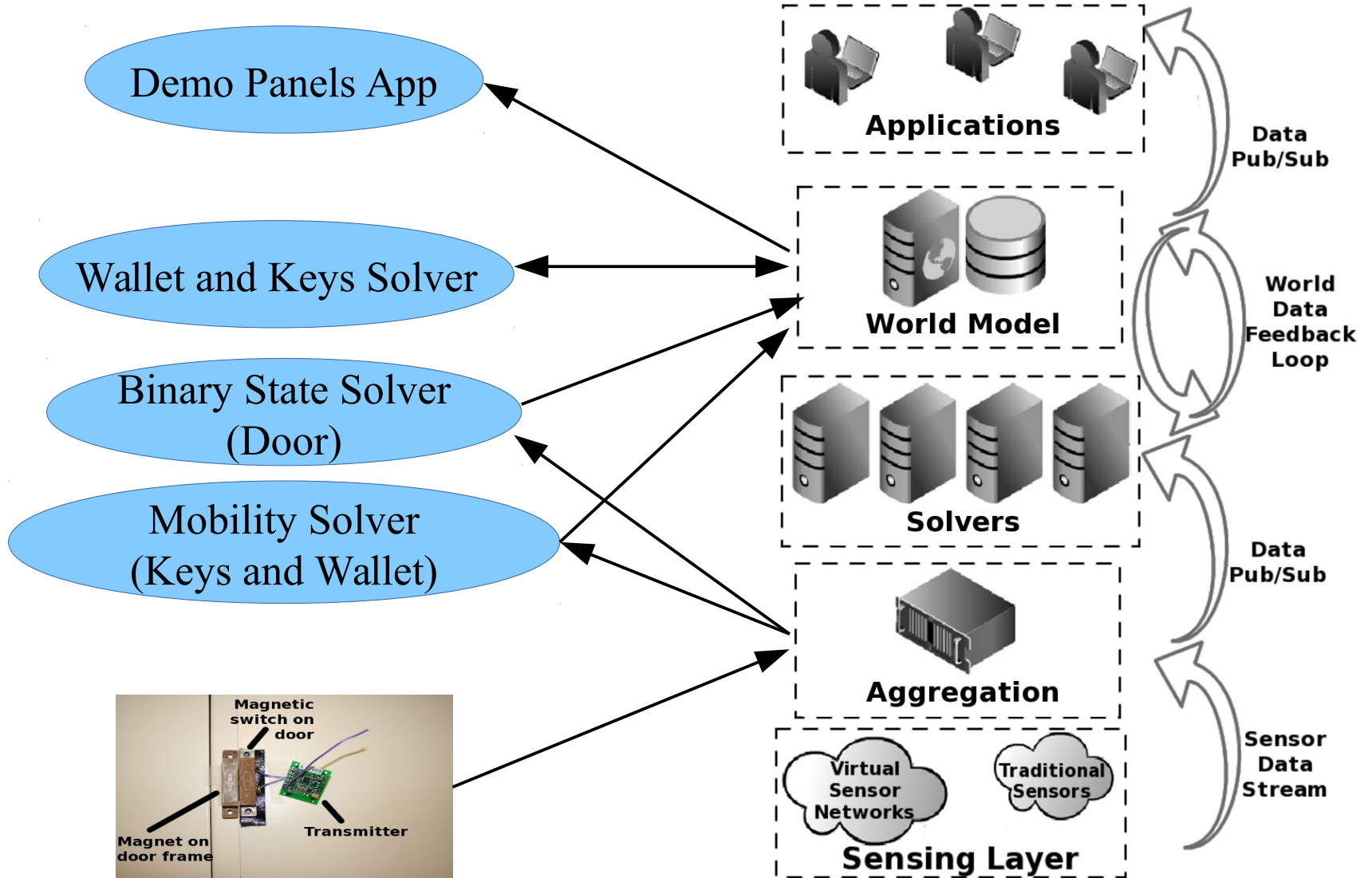
 Foreach item in [wallet, keys]

 If (the item has not moved in the last 10 seconds) then
 add it to the list of missing items

 If (the list is non-empty)

 send an alert with the list of missing items.

Solver chain for the Wallet and Keys panel



Binary State and Mobility Solvers

Binary state:

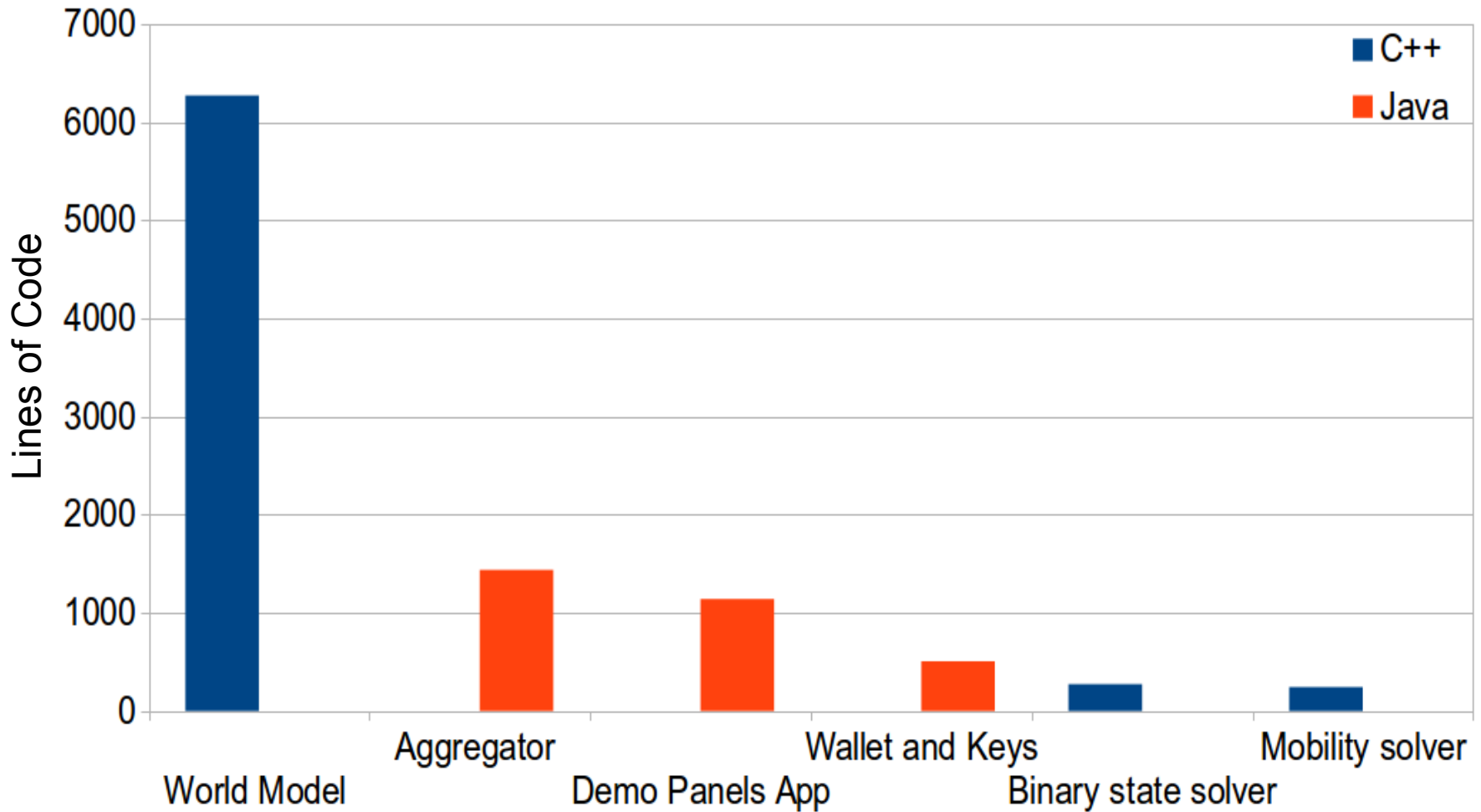
Read sensor value, put open/closed state in the world model

Mobility detection:

Read wireless received signal power over period of N seconds

If signal variance is over a threshold, change object's state to moving in the world model

Owl Lines of Code



Other Applications

- Leak detection
 - Sense standing water, email/SMS if water detected
- Office space assignment
 - Sense door open/closes, assign new students to lightly used offices
- Fresh Coffee
 - Sense temperature of coffee pot, email/SMS if a temp spike
- Chair Stolen
 - Email/SMS if a chair is moved away from the owner's cubicle
- Loaner Bicycle Inventory
 - Count # of bicycles in a room to see if one is available.

Evaluation of ease of development

Perform a user-study in next 3 months.

Have undergraduates develop simple application in Owl and another system (Smartthings)

Send an alert when 3 door sensors are triggered in order within a time window.

Metrics: learning time, development time, lines of code, functionality, code quality.

Experiment has passed human subject review approval process

Conclusions and Future Work:

Conclusions:

- Application development simplified
- Codebase to accomplish sensor applications surprising small

Future work:

- Leverage origin ID for security and privacy
- Continue to add applications in student seminars and projects
- Need to add actuation layers for next version

Owl Resources:

Main Developer's page:

<http://www.owlplatform.com/developers.php>

World Model:

<https://github.com/OwlPlatform/world-model>

Aggregator:

<https://github.com/OwlPlatform/aggregator>

Makerfaire Demo Application:

<https://github.com/romoore/maker-demo>

Wallet and Keys solver:

<https://github.com/romoore/wallet-and-keys>

Binary state solver (switch):

https://github.com/OwlPlatform/binary_state_solver

Signal strength solvers (mobility):

<https://github.com/OwlPlatform/signal-strength-solvers>

Backup slides

Backup slides

Lines of Code

World Model (C++): 6274

Aggregator (Java) : 1439

Makerfaire Demo Application (Java): 1142

Wallet and Keys solver (Java): 507

Binary state solver (switch) (C++): 273

Signal strength solver (mobility) (C++): 244

Finding Variables in the World Model

```
StepResponse mobilityResponse =
    this.asClient.getStreamRequest(itemIds,
        System.currentTimeMillis(), 0, mobilityAttributes);

StepResponse doorResponse =
    this.asClient.getStreamRequest(doorIds,
        System.currentTimeMillis(), 0, doorAttributes);
try {
    // Keep going until an error or a mobility update
    while ((!mobilityResponse.isComplete() &&
        !mobilityResponse.isError() &&
        (!doorResponse.isComplete() && !doorResponse.isError()))) {
        if (mobilityResponse.hasNext()) {
```

Connecting to the World Model

```
StepResponse mobilityResponse =
    this.asClient.getStreamRequest(itemIds,
        System.currentTimeMillis(), 0, mobilityAttributes);

StepResponse doorResponse =
    this.asClient.getStreamRequest(doorIds,
        System.currentTimeMillis(), 0, doorAttributes);
try {
    // Keep going until an error or a mobility update
    while ((!mobilityResponse.isComplete() &&
        !mobilityResponse.isError() &&
        (!doorResponse.isComplete() && !doorResponse.isError()))) {
        if (mobilityResponse.hasNext()) {
```

Example aggregator code

```
/** Connection to the aggregator. using "poll" */  
  
SolverAggregatorConnection agg = new  
    SolverAggregatorConnection();  
  
if (!this.agg.connect(10000)) {  
    System.err.println("Unable to connect");  
    return false;  
}
```

Example aggregator code

```
/** Connection to the aggregator. using "poll" */  
  
SolverAggregatorConnection agg = new  
    SolverAggregatorConnection();  
  
if (!this.agg.connect(10000)) {  
    System.err.println("Unable to connect");  
    return false;  
}
```

Connecting to the Aggregator (cont)

```
SubscriptionRequestRule everythingRule = new
    SubscriptionRequestRule();
    everythingRule.setUpdateInterval(01);

everythingRule.setPhysicalLayer(SampleMessage.PHYSICAL_LAYER_AL
L);
this.agg.addRule(everythingRule);

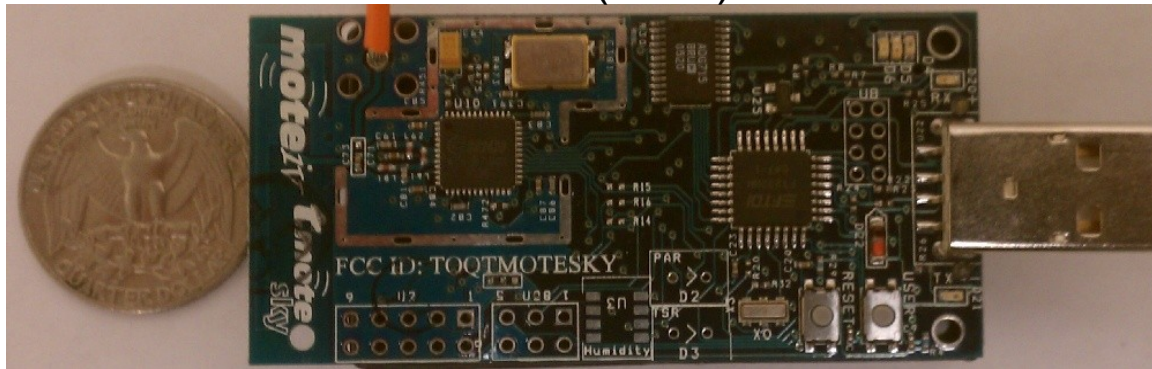
SampleMessage sample = null;
while ((sample = this.agg.getNextSample()) != null) {
    Attribute attr = new Attribute();
    // If the RSSI value is above threshold, say it's "nearby".
    if (sample.getRssi() > RSSI_THRESHOLD) {
        System.println("Chair is nearby");
    }
}
```


Sensor simplicity

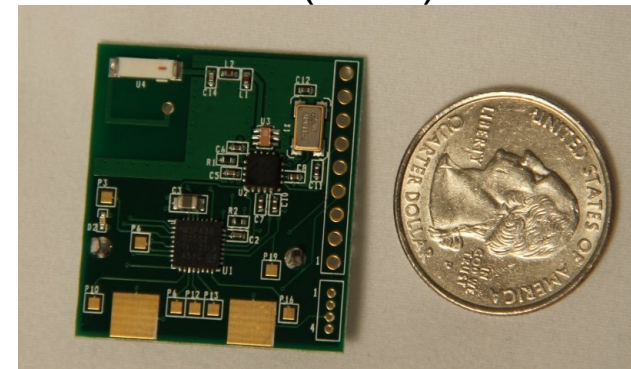
- Sensor node cost is a limitation for many applications
 - Applications enabled at sensor cost of \$100, \$10, \$1, 10¢, 1¢ ?
- Cost assumptions based on scaling Moore's law real omit real constraints
 - 15 years show these constraints are fundamental
- Cost is driven by the number and type of components, not Moore's law!
- TO reduces costs by several factors
 - enough to expand the application space (\$80->\$10)
- Marginal costs will only go down if there is a true single-chip sensor
 - But high fixed costs remain a barrier for a true single chip solution!

Two wireless sensor boards

Classic
TelosB (2004)



Transmit-Only
TO-PIP(2013)

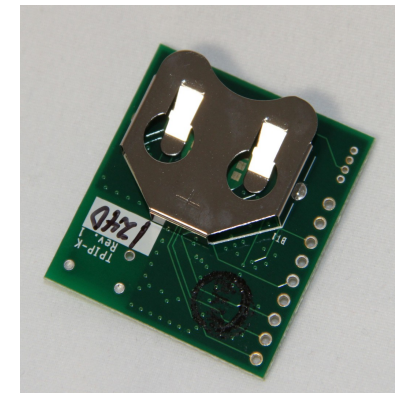


Antenna

Radio

Micro controller

Battery



Component counts

