Accessors
A Software Architecture for IoT

Edward A. Lee and Christopher Brooks

EECS, UC Berkeley

Tutorial on Tools for Real-Time Systems (TuToR)

In conjunction with the Real-Time Systems Symposium (RTSS)

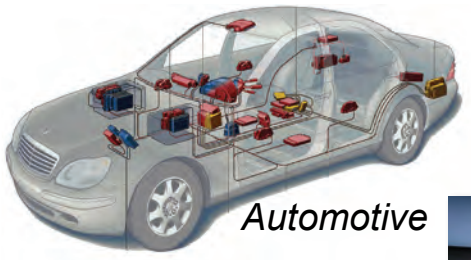
December 5, 2017 – Paris, France

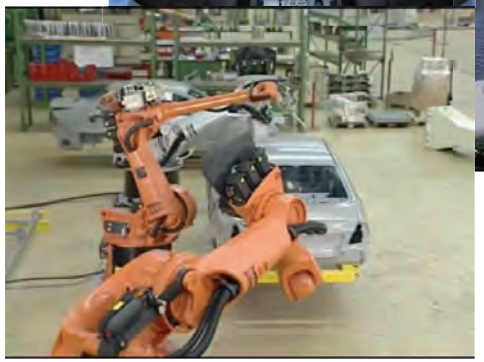
University of California at Berkeley

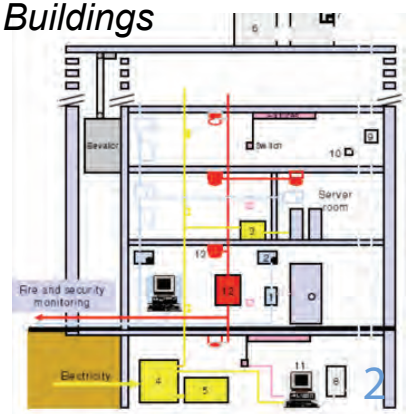
Cyber-Physical Systems
Focus on the Internet of *Important* Things

Not just information technology:

- Cyber + Physical
- Computation + Dynamics
- Security + Safety


Automotive


Manufacturing



Properties:

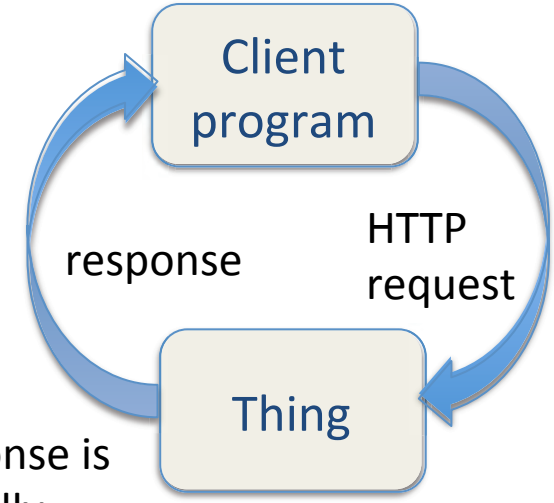
- Highly dynamic
- Safety critical
- Uncertain environment
- Physically distributed
- Sporadic connectivity
- Resource constrained

We need engineering models and methodologies for dependable cyber-physical systems.

A Common IoT Design Pattern: REST with AACs

A RESTful service [Fielding & Taylor 2002] is accessed using a design pattern common on the web that we call *Asynchronous Atomic Callbacks* (AAC) (also called the *Reactor Pattern*).

In the Web, AAC is widely used. It is central to many popular internet programming frameworks such as Node.js & Vert.x, and to CPS frameworks such as TinyOS.



Response is typically asynchronous to avoid blocking the client program.

Response handler executes atomically.

URL encodes all state info (credentials, commands, etc.)

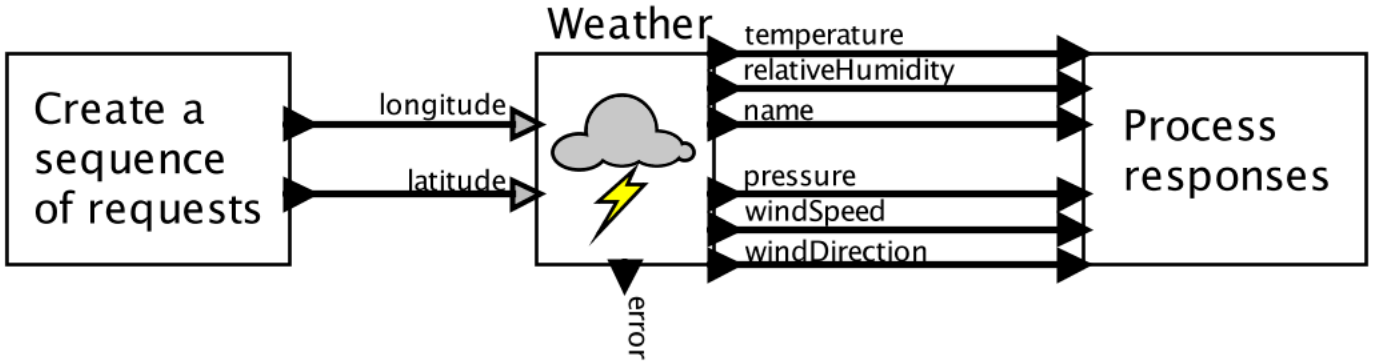
Example in JavaScript

```
// Import a module providing network services
var http = require("http");
// Construct a URL encoding a request
var url = "http://foo.com/deviceID/...";
// Issue the request and provide a callback
http.get(url, function(response) {
    // ... handle the response ...
});
```

The callback function will be called atomically some time later when the server response arrives.

Another Common Design Pattern: *Actors*

Streaming requests:

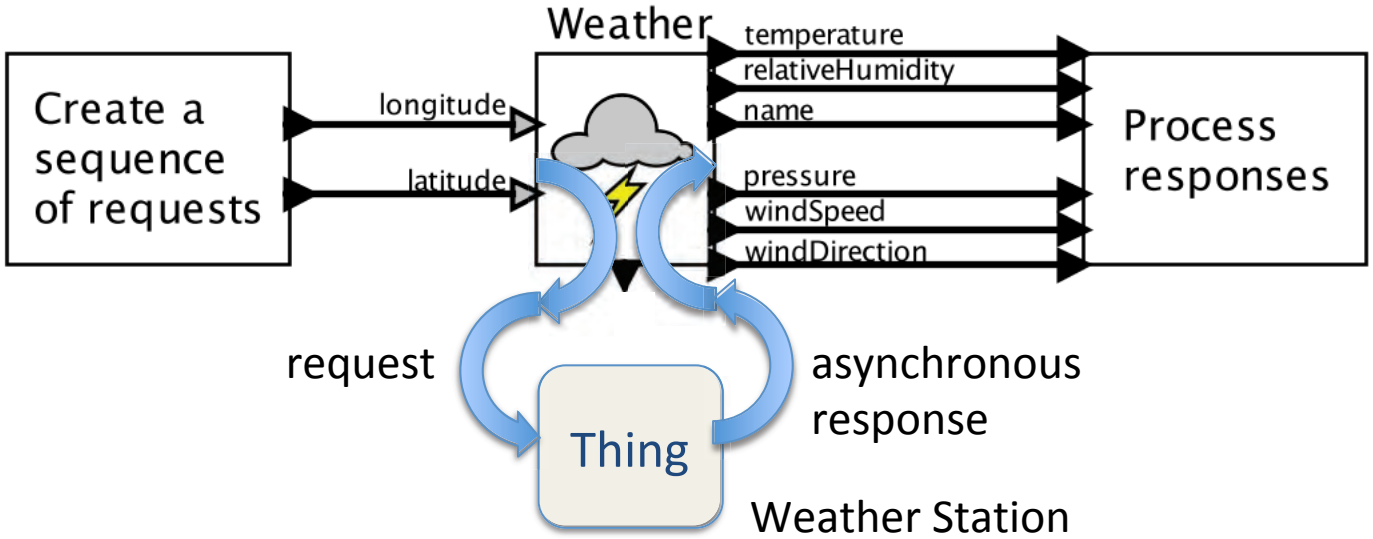


Sequence of requests for a service (a stream) triggers a sequence of responses.

Actors embrace concurrency and scale well.

Actors and AAC

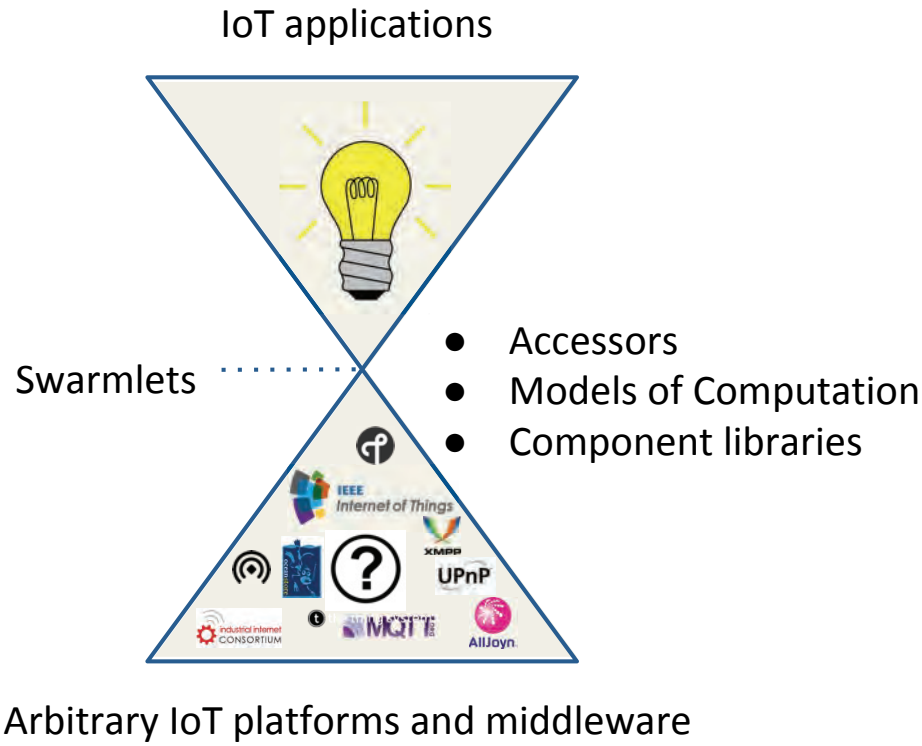
Streaming requests:



This is a key property of accessors, a design pattern for IoT that embraces concurrency, asynchrony, and atomicity.

Principled Composition of Devices and Services

- A **Swarmlet host** is a platform for component-based design (analogous to a **browser**, but for things, not people).
- A **Swarmlet host** can instantiate **accessors**, which serve as local proxies for devices and services.
- **Accessors** can be provided by device vendors or third parties.
- **Accessors** enable interoperability of **independently designed** components.

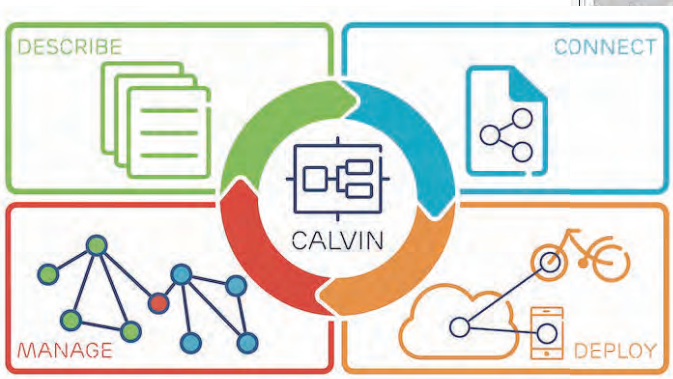


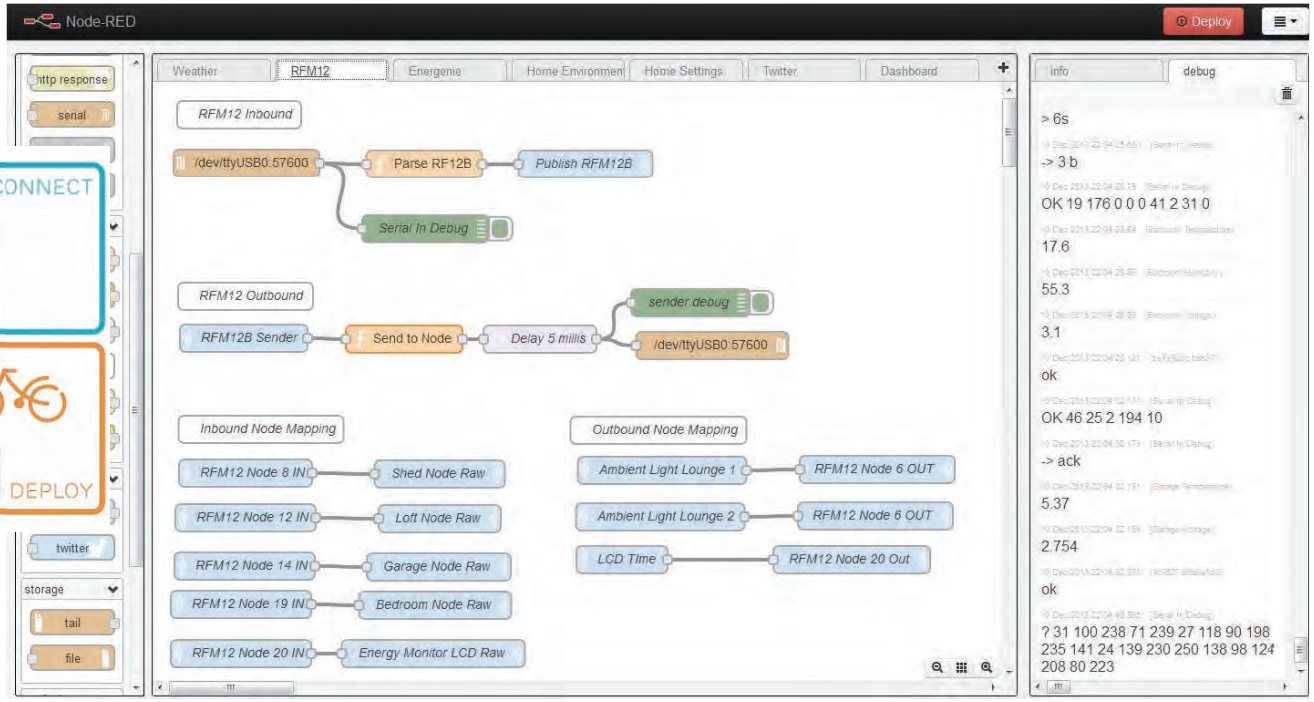
We are not alone pursuing this approach

Notable efforts:

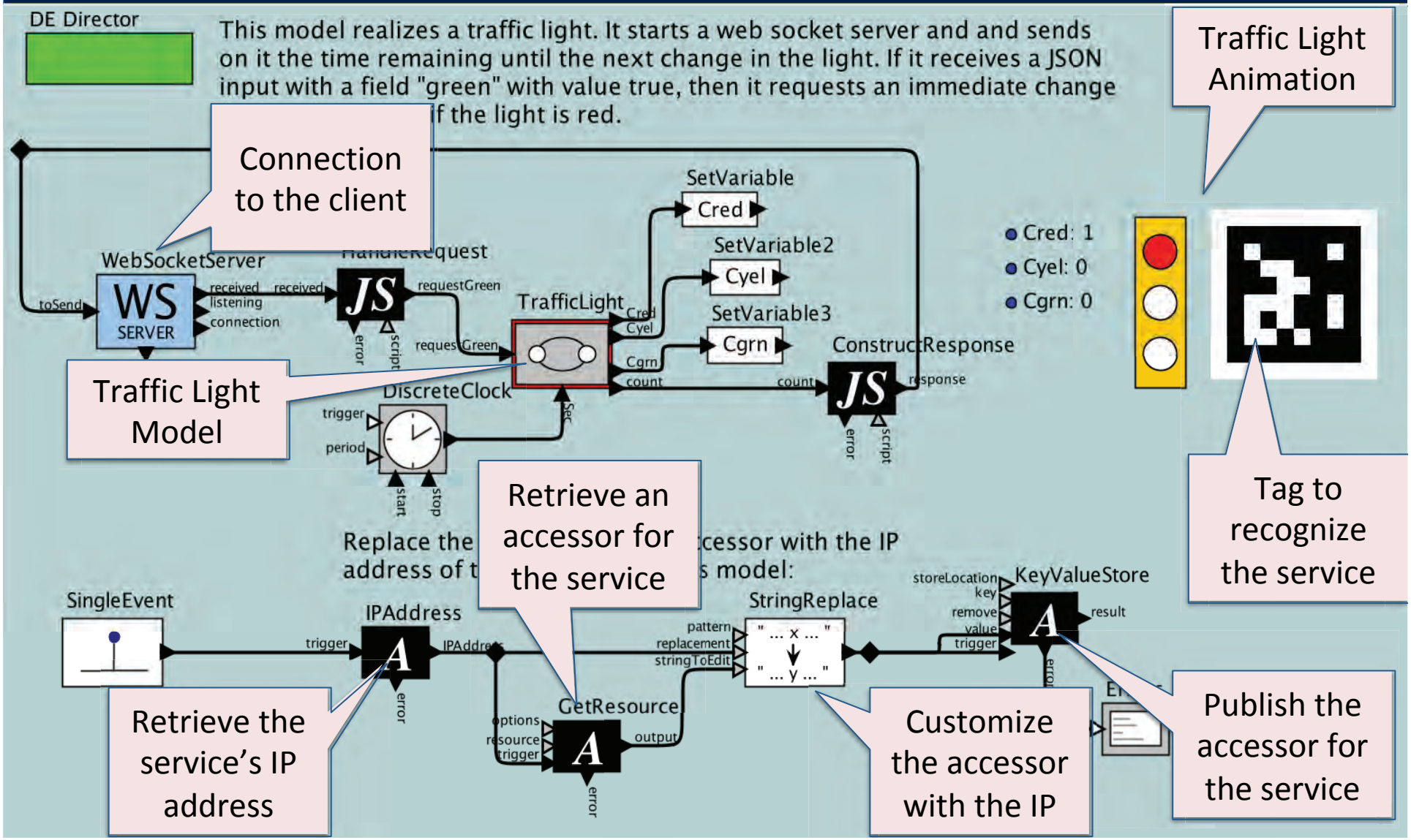
- Node Red (IBM)
- Calvin (Ericsson)

From: “Home Automation with Node Red, JeeNodes and Open Energy Monitor,” Dom Bramley's Blog of Maximo and the 'Internet of Things’, IBM Developer Works, Dec., 2013.

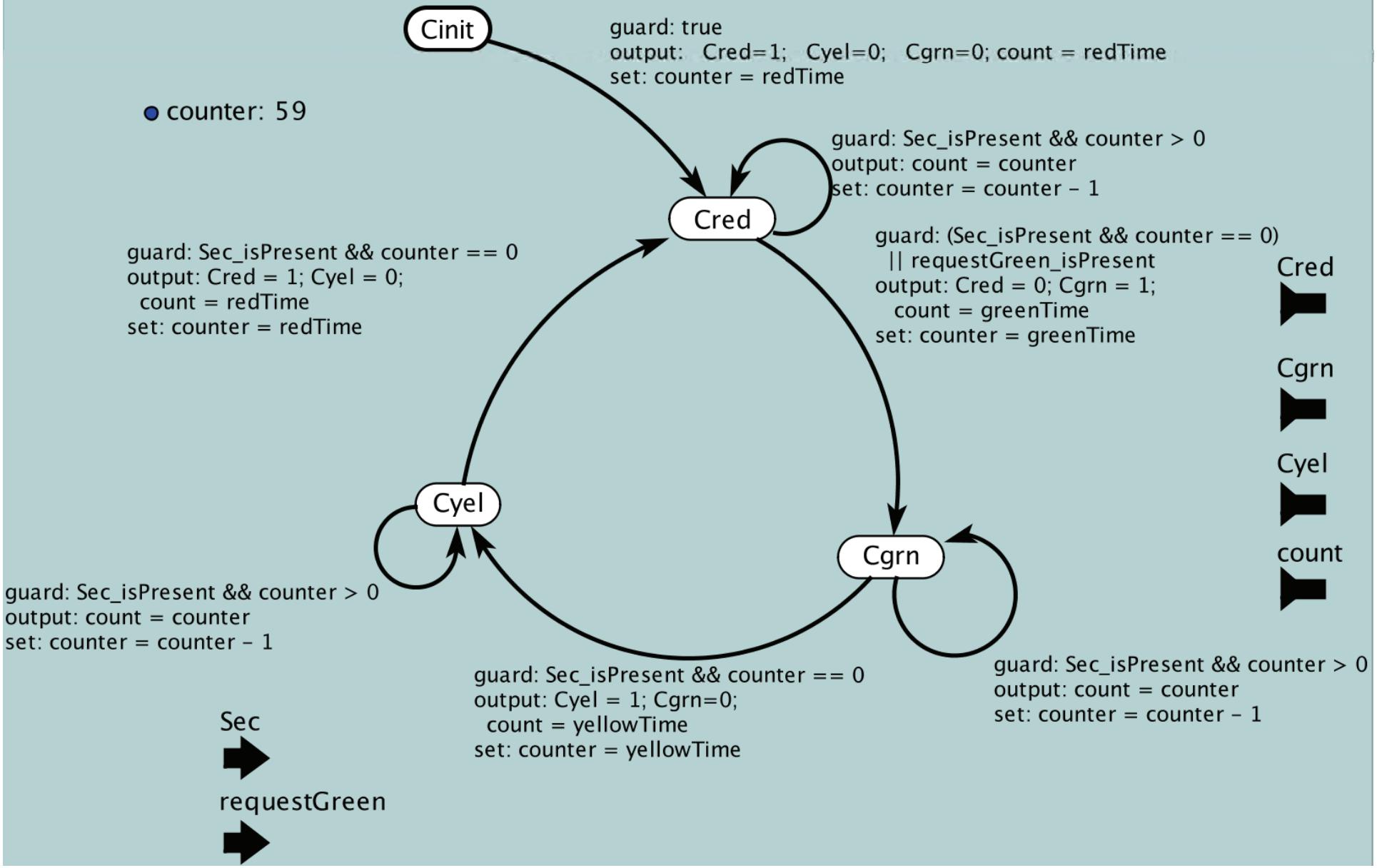




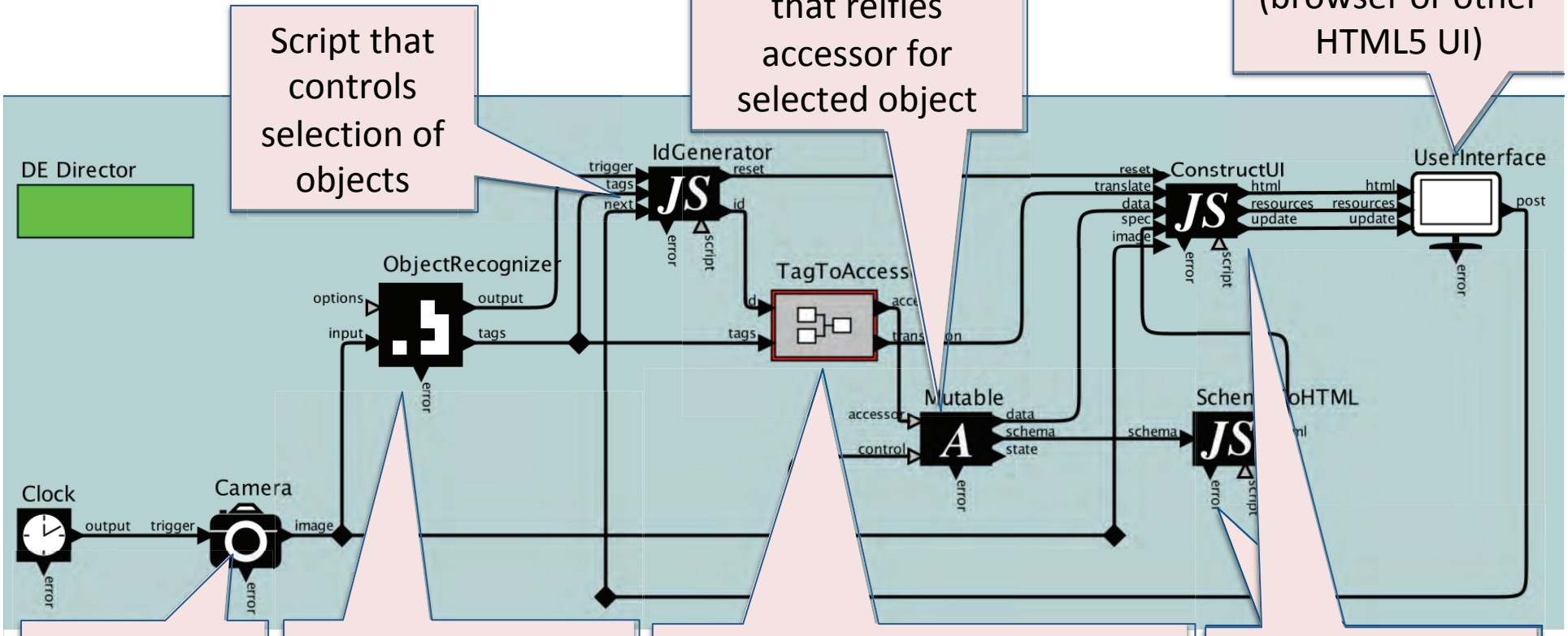
Demo: A Service



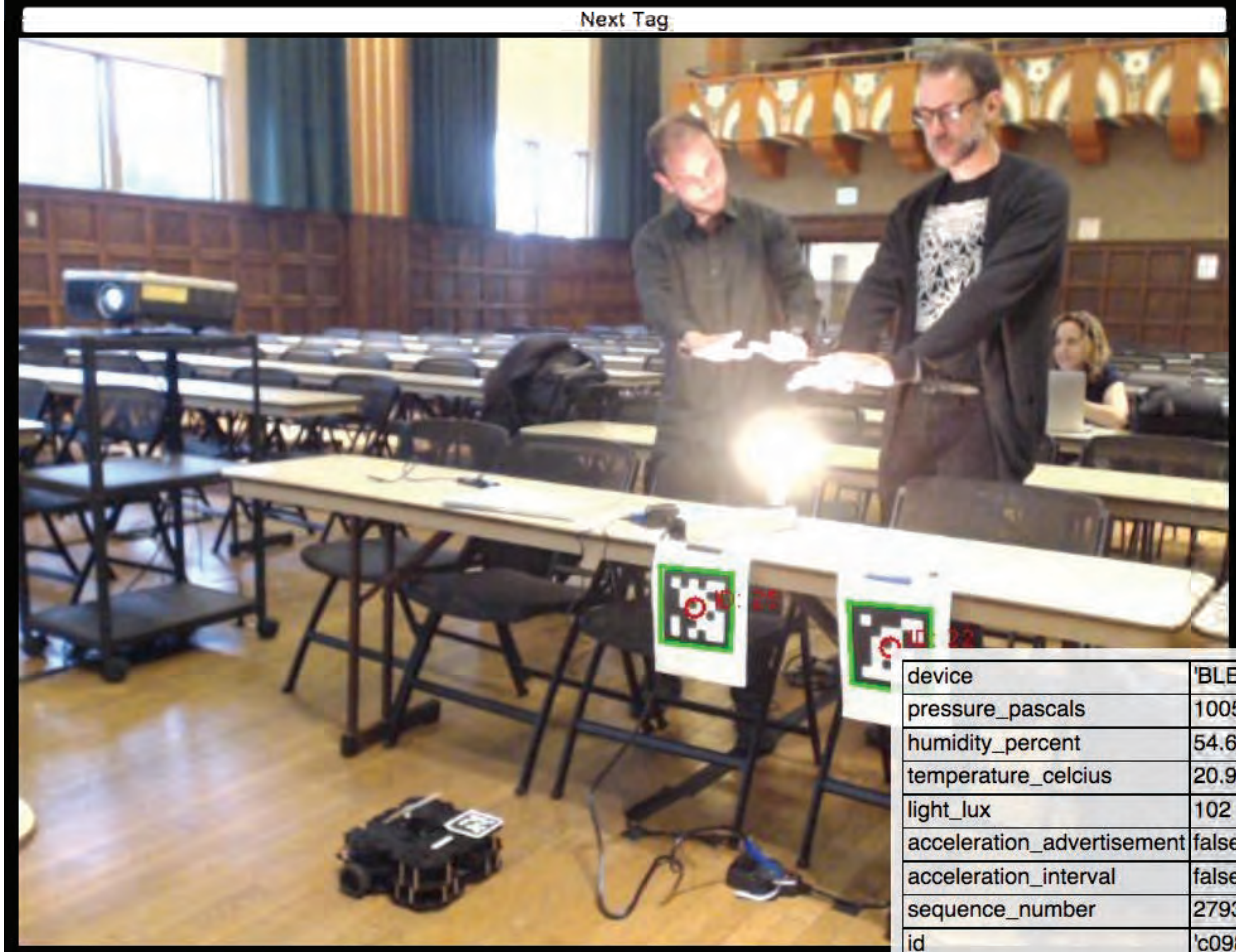
Traffic Light Model



Demo – Augmented Reality



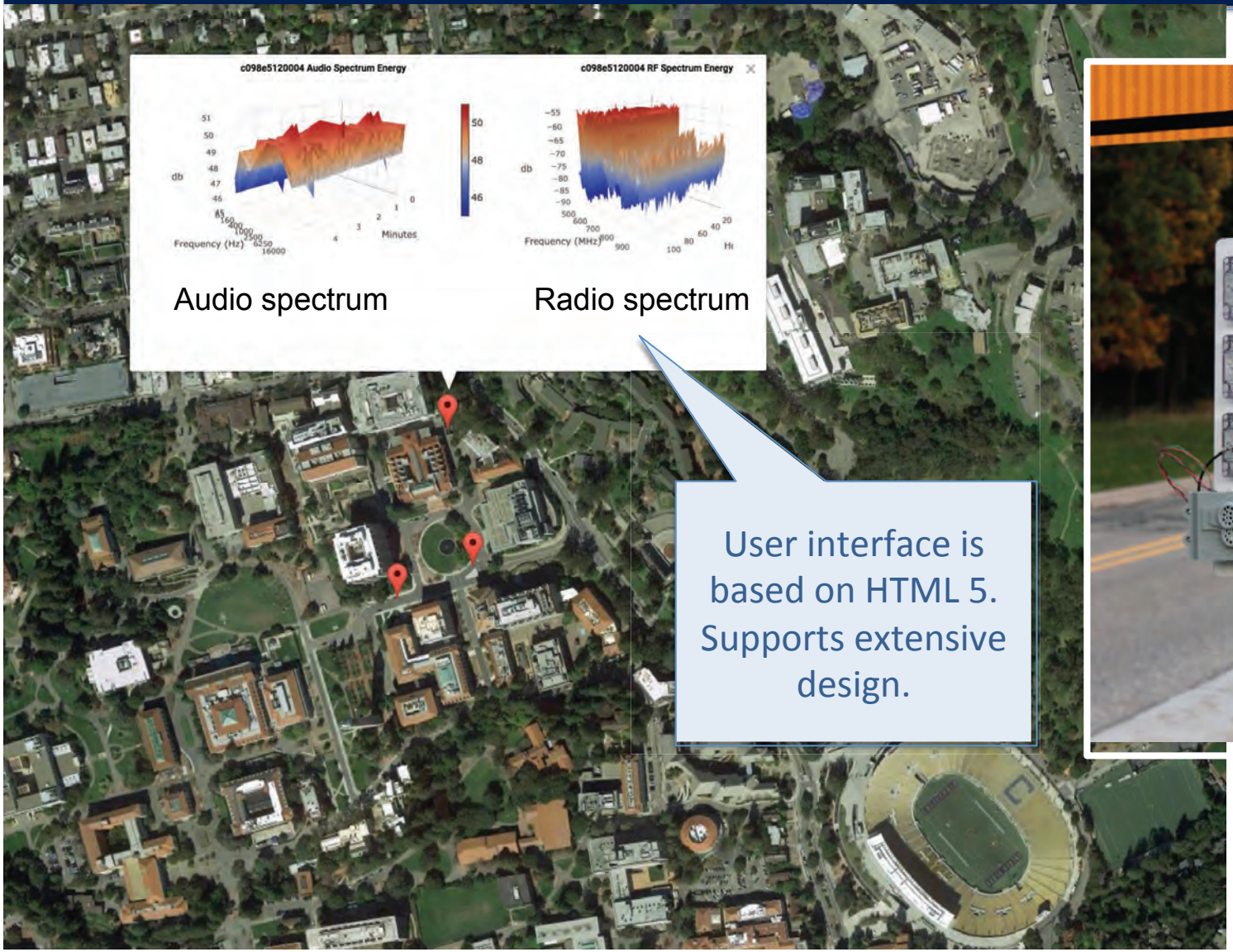
Example UI for Augmented Reality



Exactly the same model being used at the TerraSwarm annual meeting to access a variety of sensors and actuators.

device	'BLEES'
pressure_pascals	100543.3
humidity_percent	54.69
temperature_celcius	20.96
light_lux	102
acceleration_advertisement	false
acceleration_interval	false
sequence_number	27938
id	'c098e530005a'
_meta	{ received_time: '2017-10-11T16:26:02.781Z', device_id: 'c098e530005a', receiver: 'ble-gateway', gateway_id: 'c0:98:e5:c0:00:1c' }

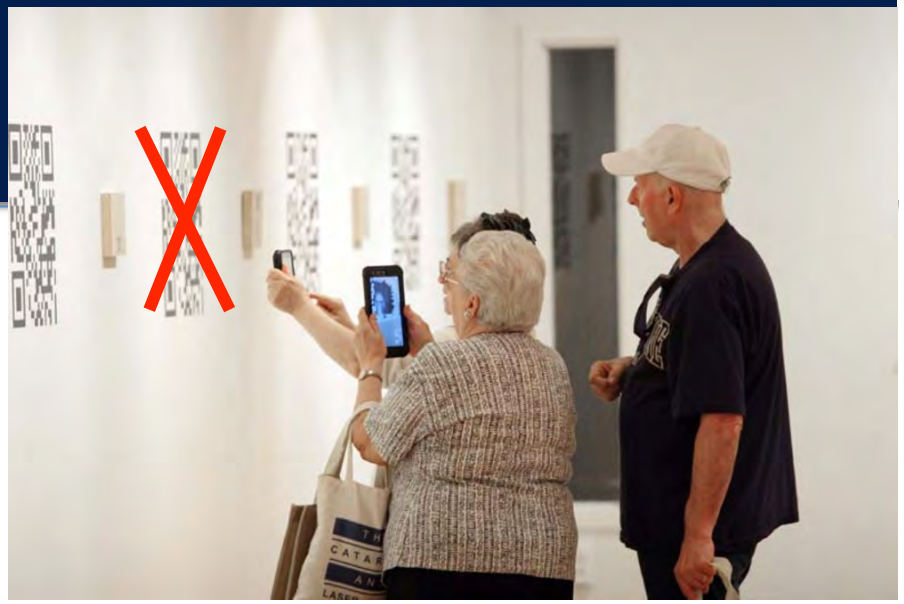
Better User Interfaces

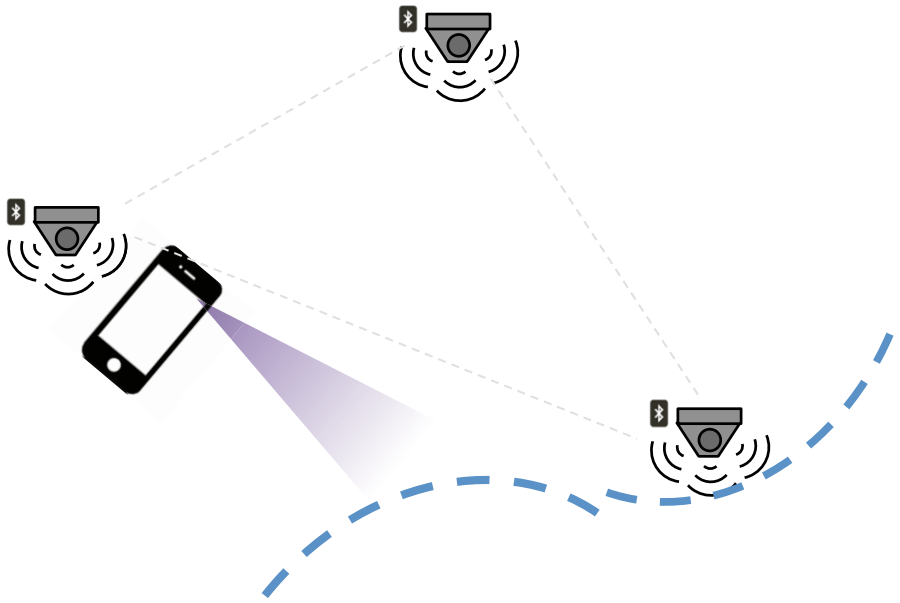


Visualization interface by Beth Osyk

Tagless Odometry

Using object recognition, GPS, inertial sensors, WiFi fingerprinting, etc., we can hope to get rid of the tags.

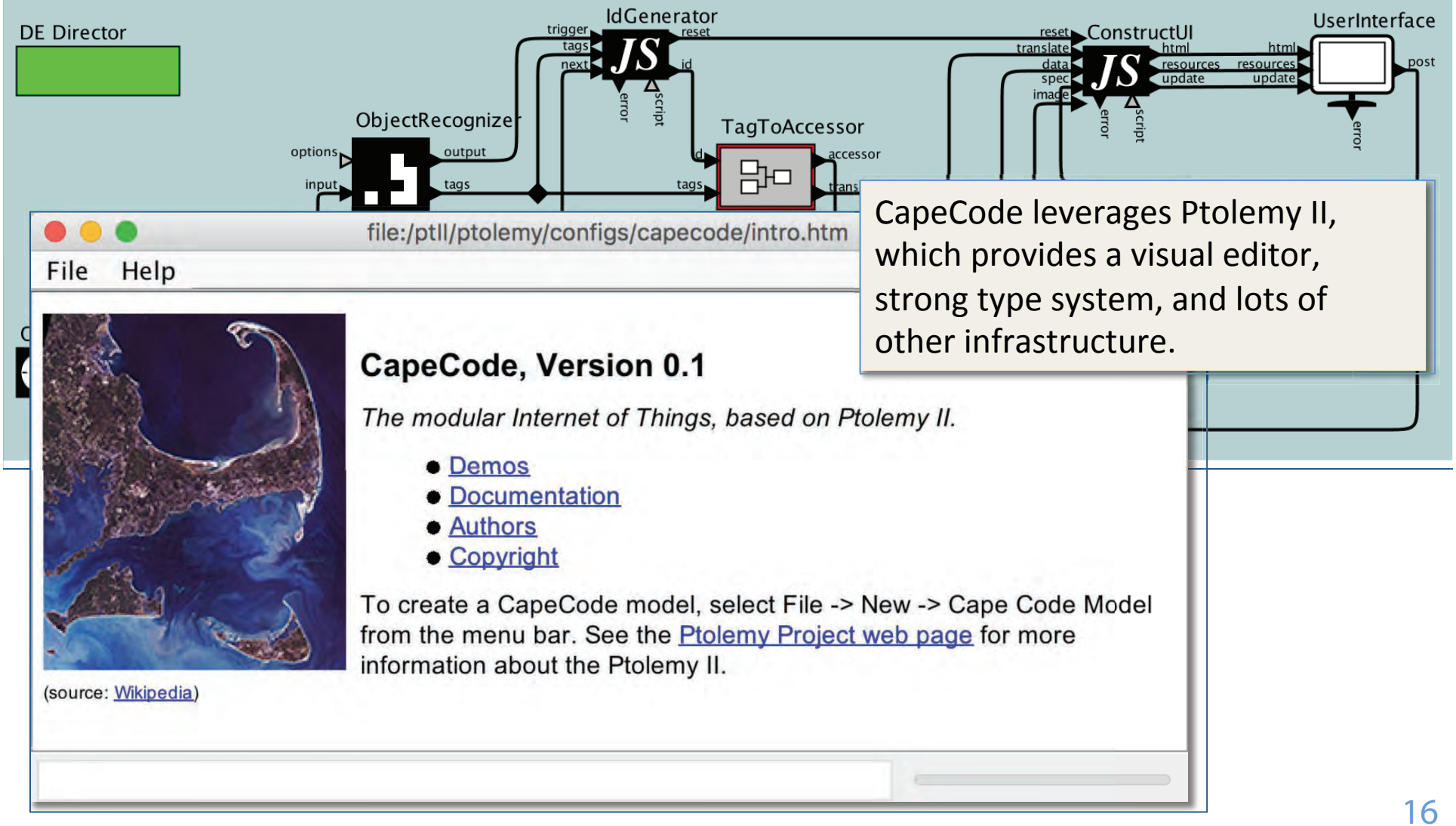




Local Devices/Services
Demonstrated at TerraSwarm Meeting

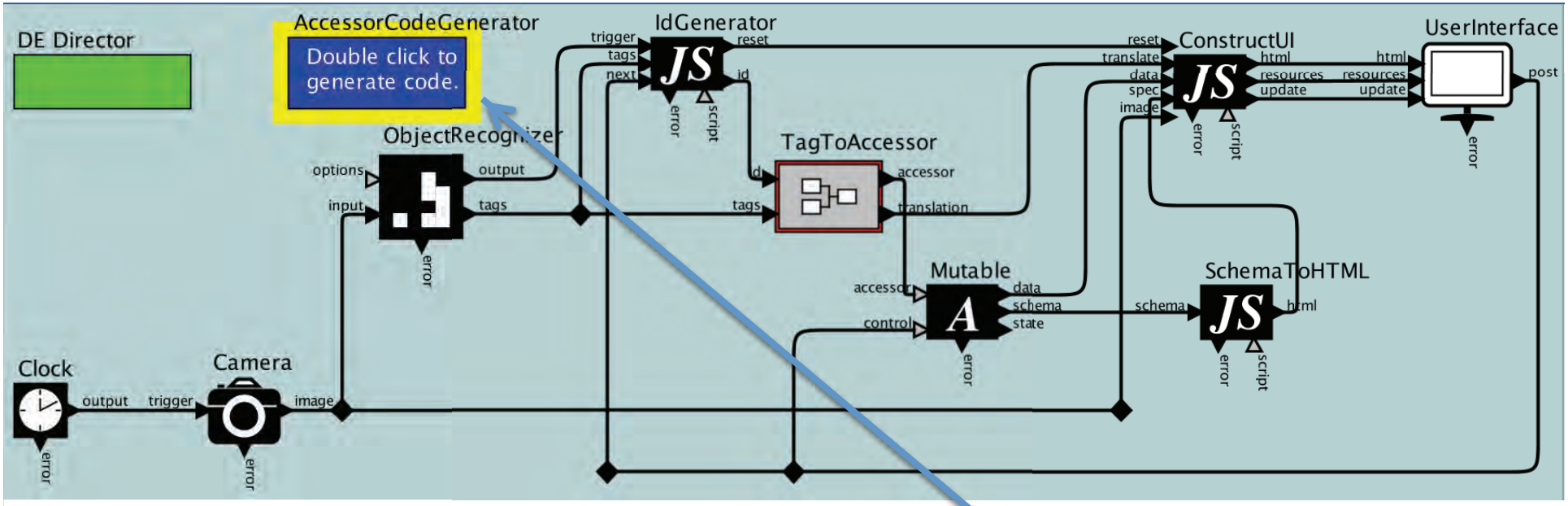
- BLEE sensors via MQTT (local comm.)
- BLEE sensors via GDP (global comm.)
- Powerblade sensors via MQTT (local comm.)
- Powerblade sensors via GDP (global comm.)
- Hue lightbulb via HTTP (local comm.)
- Robot via ROS Bridge (local comm.)
- Audio service via WebSocket (local comm.)
- MessageBoard via HTTP (local comm.)

CapeCode: Swarmlet Development
Our Programming Framework for the IoT



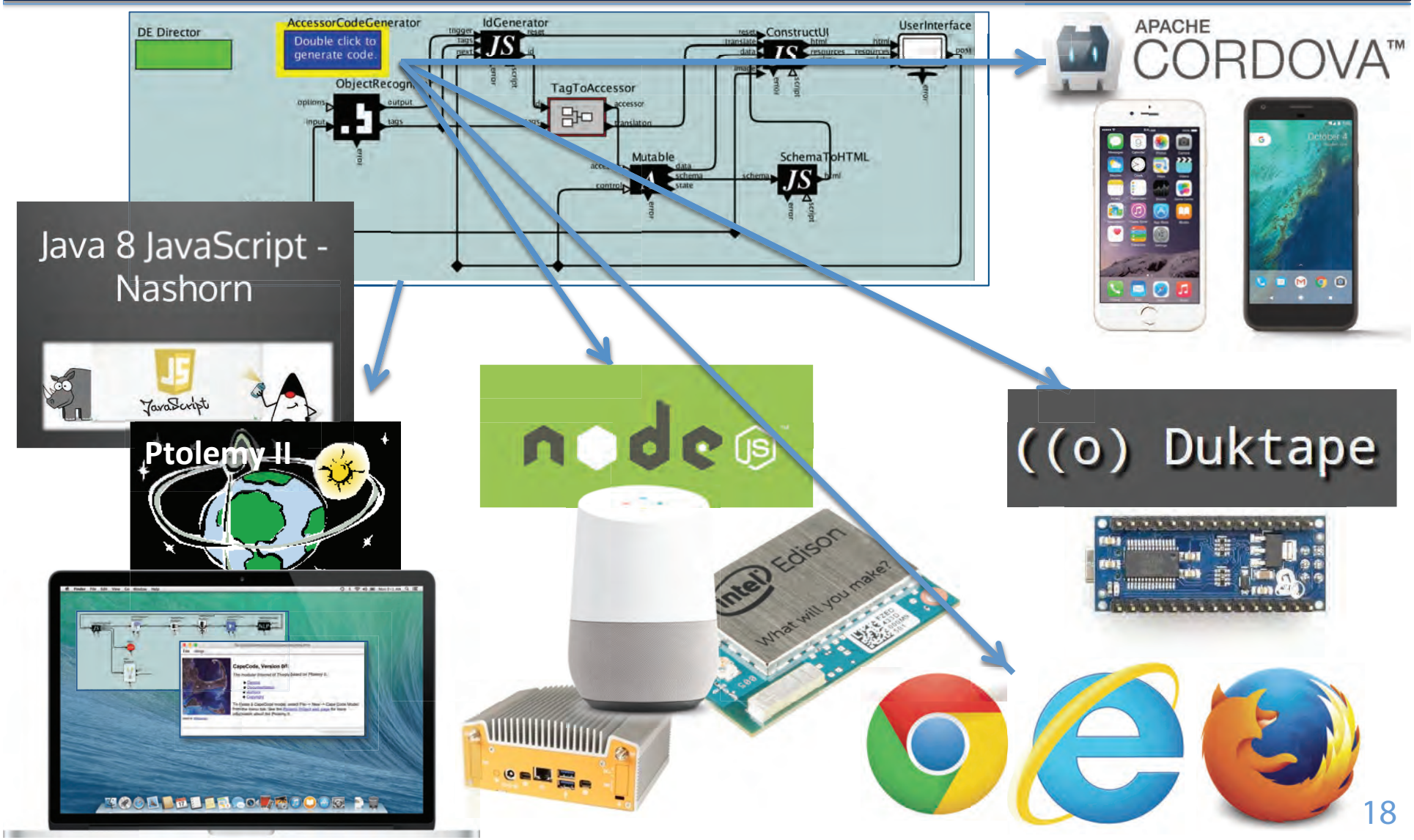
CapeCode leverages Ptolemy II, which provides a visual editor, strong type system, and lots of other infrastructure.

Deploying Swarmlets

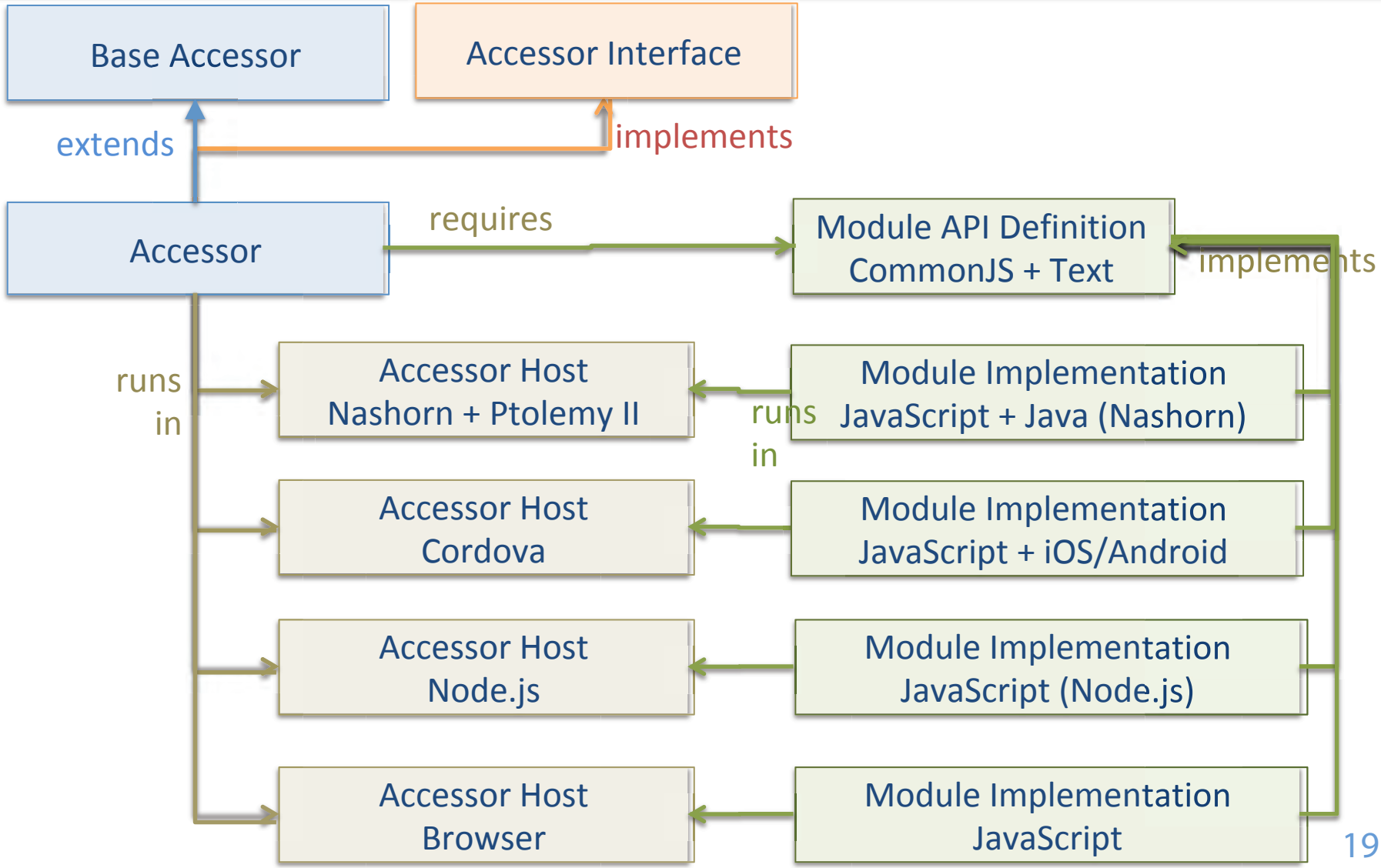


Code generator produces JavaScript files that can run on a lightweight JavaScript host.

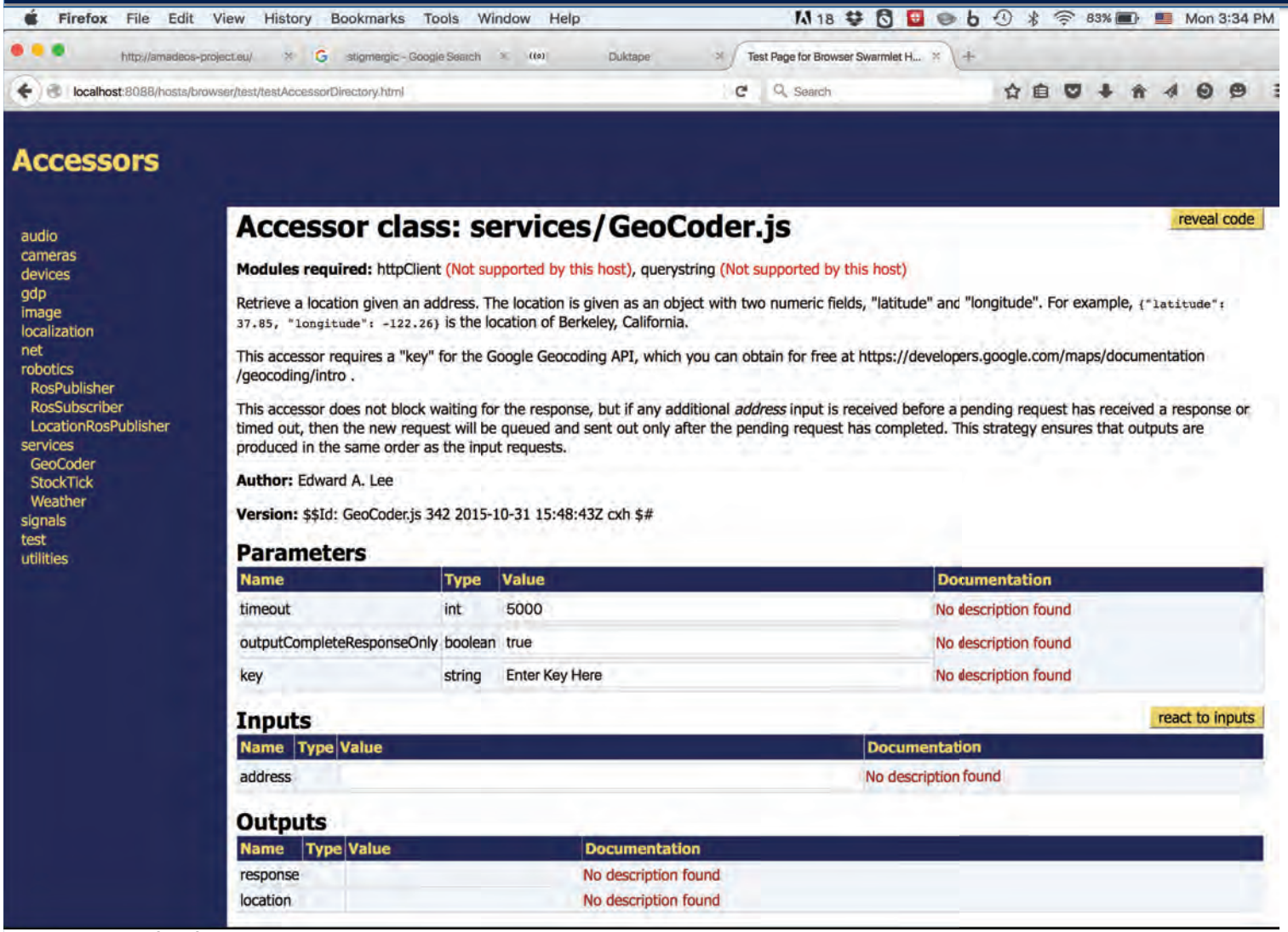
Deploying Swarmlets
Write Once, Run Everywhere?



Accessor Architecture Version 1.0
http://accessors.org



Browser Host



Key challenge: Many accessors require modules that cannot be supported in a browser due to security constraints.

Node.js Host

https://accessors.org/wiki/Main/NodeHost

Install and invoke:

```
svn co https://repo.eecs.berkeley.edu/svn-anon/projects/terraswarm/accessors/trunk/accessors
cd accessors/web/hosts/node
node nodeHostShell.js
```

Run the Node.js host:

```
Welcome to the Node swarmlet host (nsh). Type exit to exit, help for help.
nsh> var a = instantiate('a','test/TestGain')
Reading accessor at: /ptII/org/terraswarm/accessor/accessors/web/test/TestGain.js
Instantiated accessor a with class test/TestGain
undefined
nsh> a.initialize()
undefined
nsh> a.setParameter('gain', 4)
undefined
nsh> a.provideInput('input', 5)
undefined
nsh> a.react()
TestGain: inputHandler: input: 5 gain: 4
undefined
nsh> a.latestOutput('scaled')
20
nsh> exit
exit
```

Challenges

- Multi-platform strategy
  - Maintaining compatible modules across hosts
- Regression tests
  - Need dummy devices and services
- Brittle designs
  - APIs for remote services change
  - Services and devices disappear
- Safety and security
  - Authorization (even without network connectivity)
- Privacy
  - Keep data local whenever possible
- Timing
  - Best-effort timing is not good enough
  - Cloud variability is too much for many applications.

iCyPhy: The Home for this Research
Industrial Cyber-Physical Systems Center

ICyPhy is a university-industry partnership to pursue pre-competitive research on design, modeling, and analysis techniques for cyber-physical systems, with emphasis on industrial applications. Topics:

- Hardware and software architectures
- Model-based design for CPS
- Verification, validation, and certification
- Highly dynamic networked systems
- The Internet of things (IoT)
- Safety, privacy, and security
- Synthesis and learning
- Localization and location-aware services
- Learning and optimization
- Safety-critical systems
- Human-in-the-loop systems.
- Systems-of-systems design
- Semantics of timed systems

http://icyphy.org




Prabal Dutta, Edward Lee, Alberto Sangionvanni-Vincetelli, Sanjit Seshia