Dominique Guinard, CTO – co-founder
@domguinard
@EVRYTHNG

Workshop: Building the Web of Things

One layer at a time!

@domguinard @vladounet

Presented at:

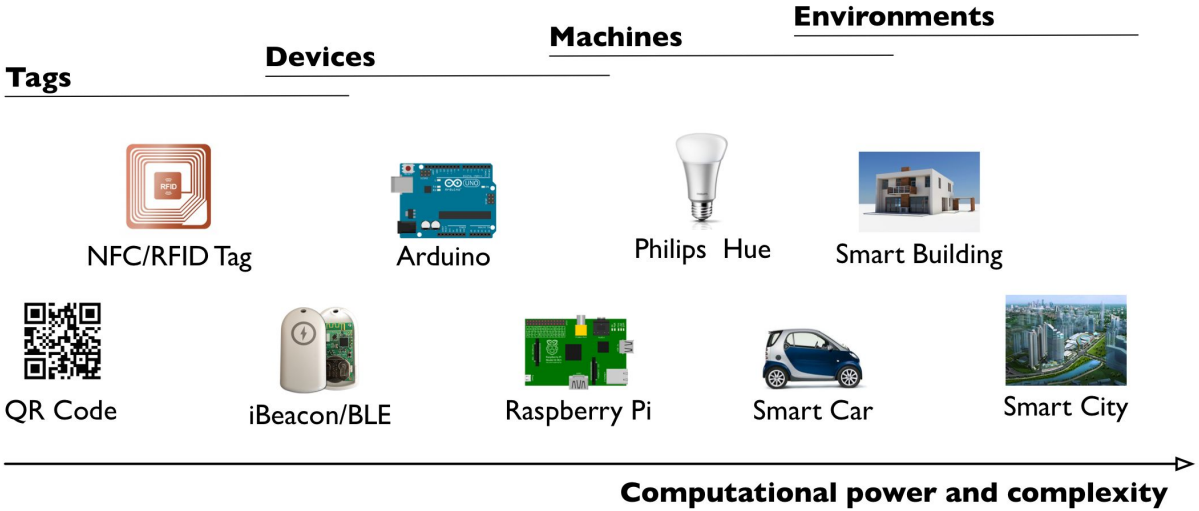
Smarter products
come with EVRYTHNG

https://bit.ly/wot-lecture

Where the Web of Things was born...

http://webofthings.org

Define IoT!



Source: Building the Web of Things: book.webofthings.io
Creative Commons Attribution 4.0

- **DEFINITION:**
  The Internet of Things is a system of physical objects that can be discovered, monitored, controlled, or interacted with by electronic devices that communicate over various networking interfaces and eventually can be connected to the wider **Internet**.

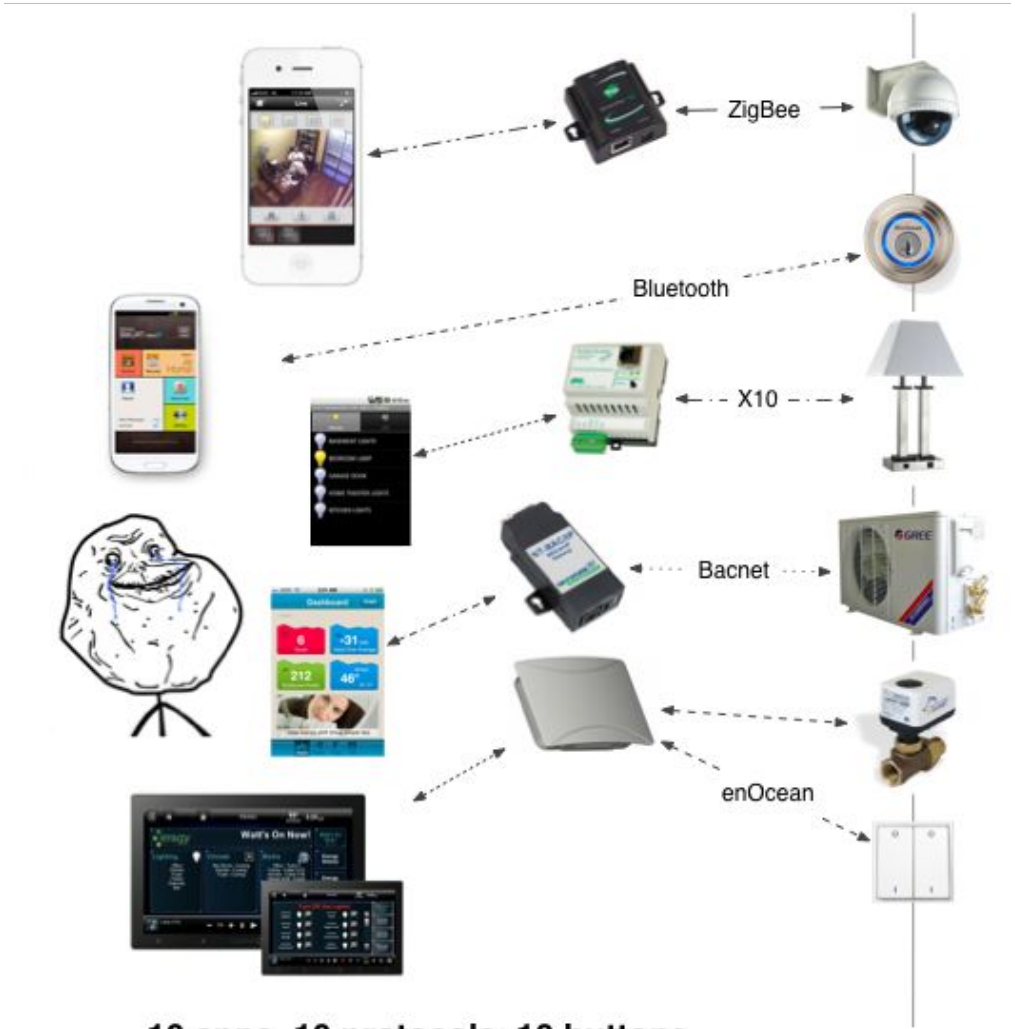
“The IoT is a science primarily focusing on creating the most complex ways of turning lights on.„

[@domguinard]

Pre IoT

+

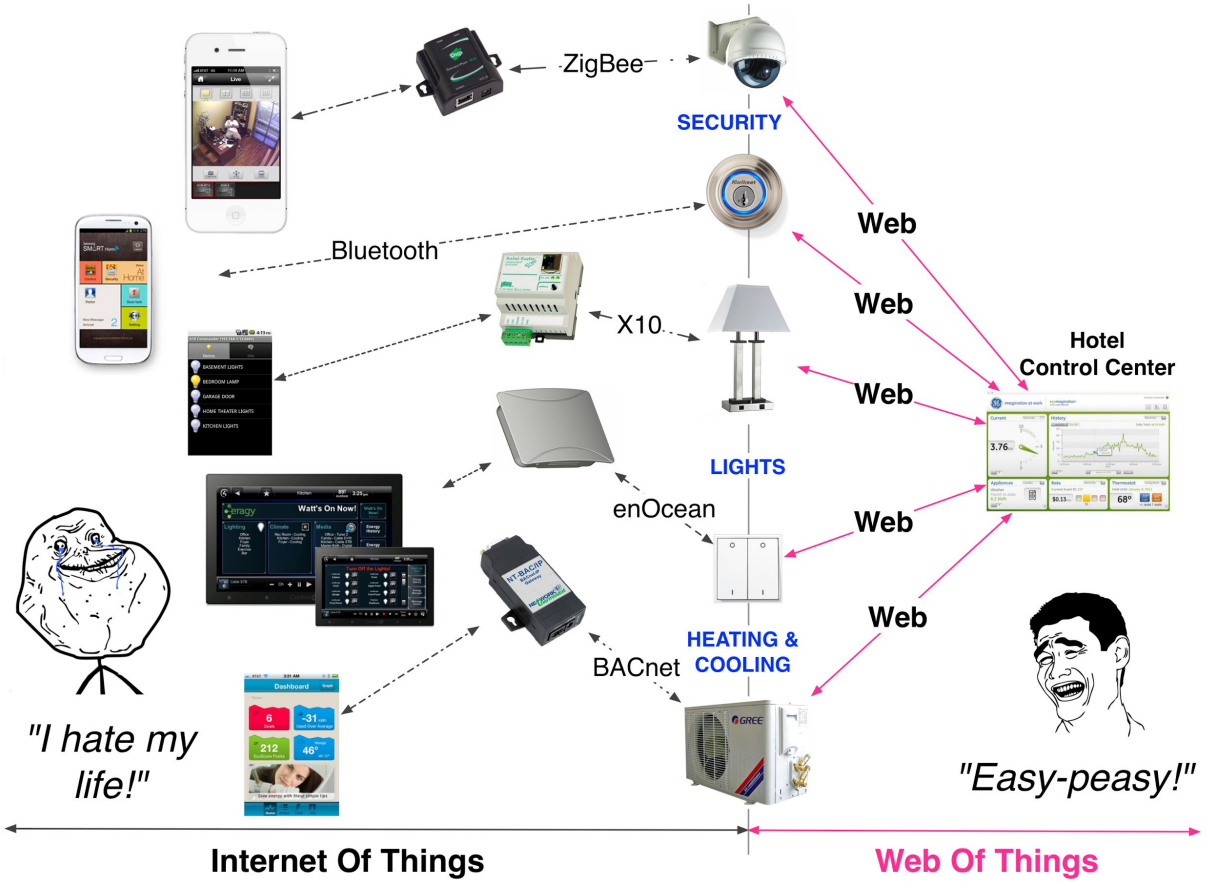
Post IoT


ZigBee
Bluetooth
X10
Bacnet
enOcean
10 apps, 10 protocols, 10 buttons

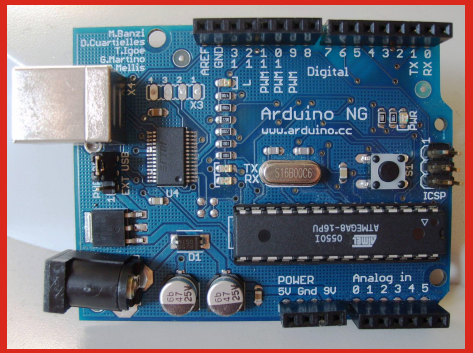
“The Web of Things is a refinement of the Internet of Things by integrating smart things not only into the Internet (network), but into the Web Architecture (application)”

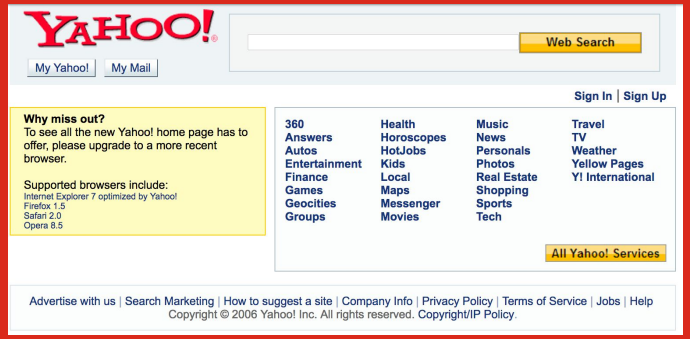
[@domguinard]

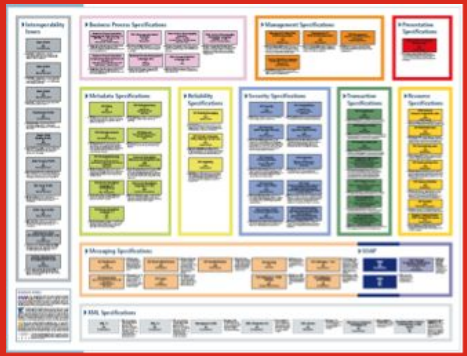
Enters the Web of Things!




WEB ENABLE
ALL THE THINGS
imgflip.com

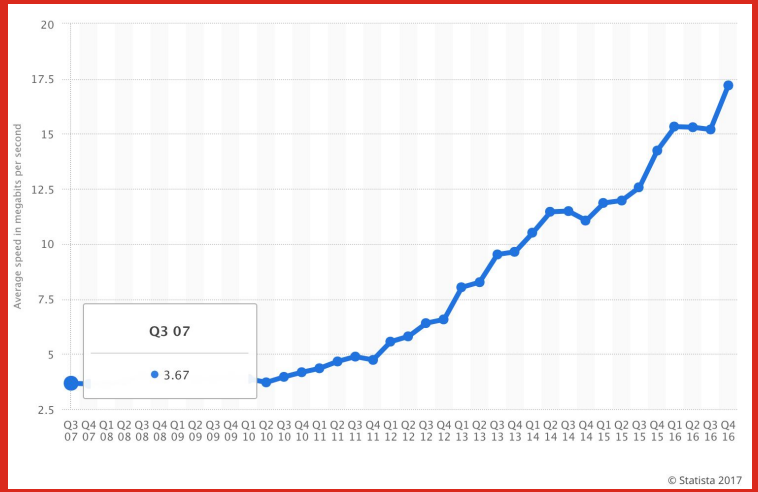






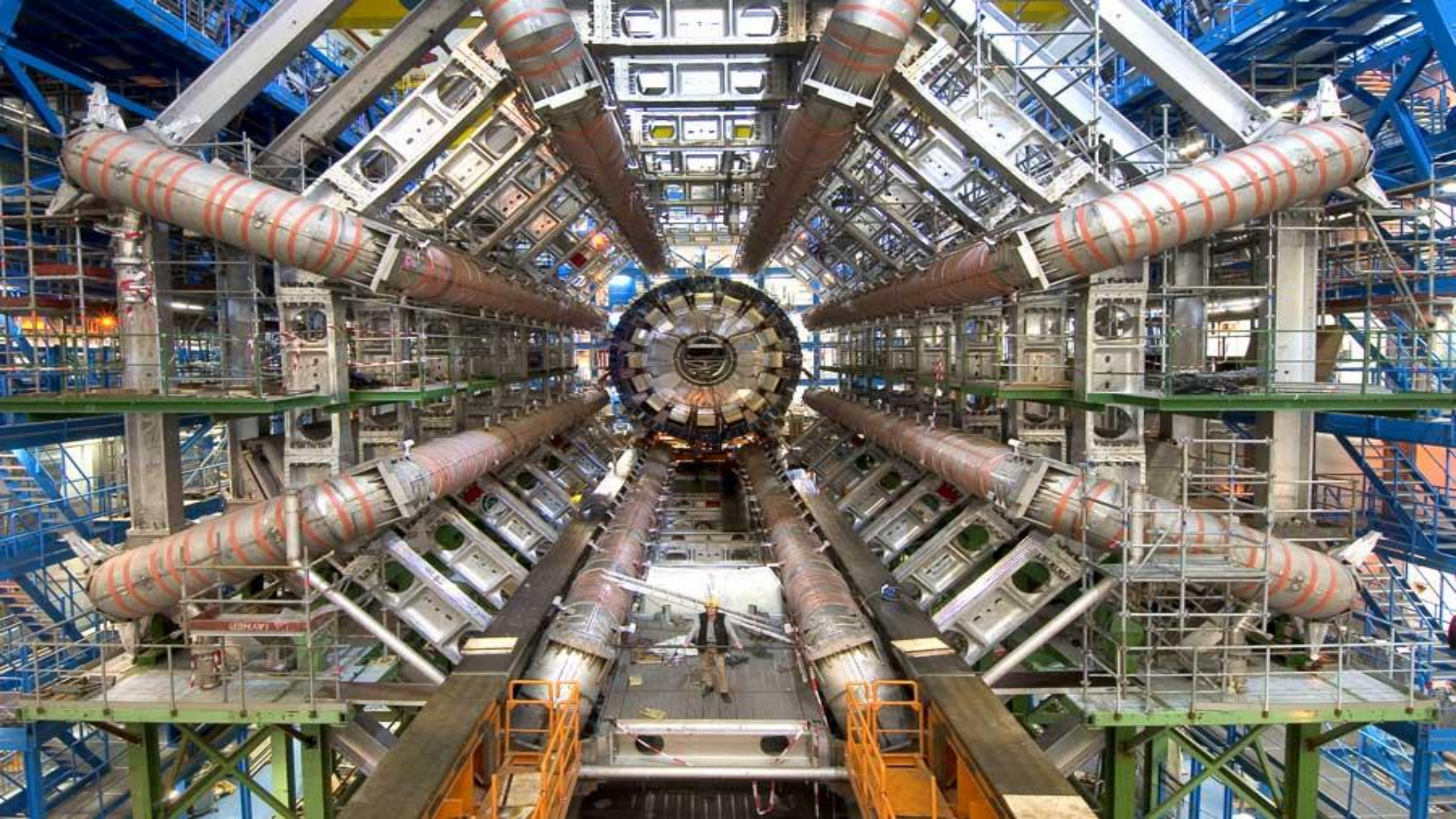
Great, but this was 2007!

Figure 6.7.: The computer hosting the RFIDLocator

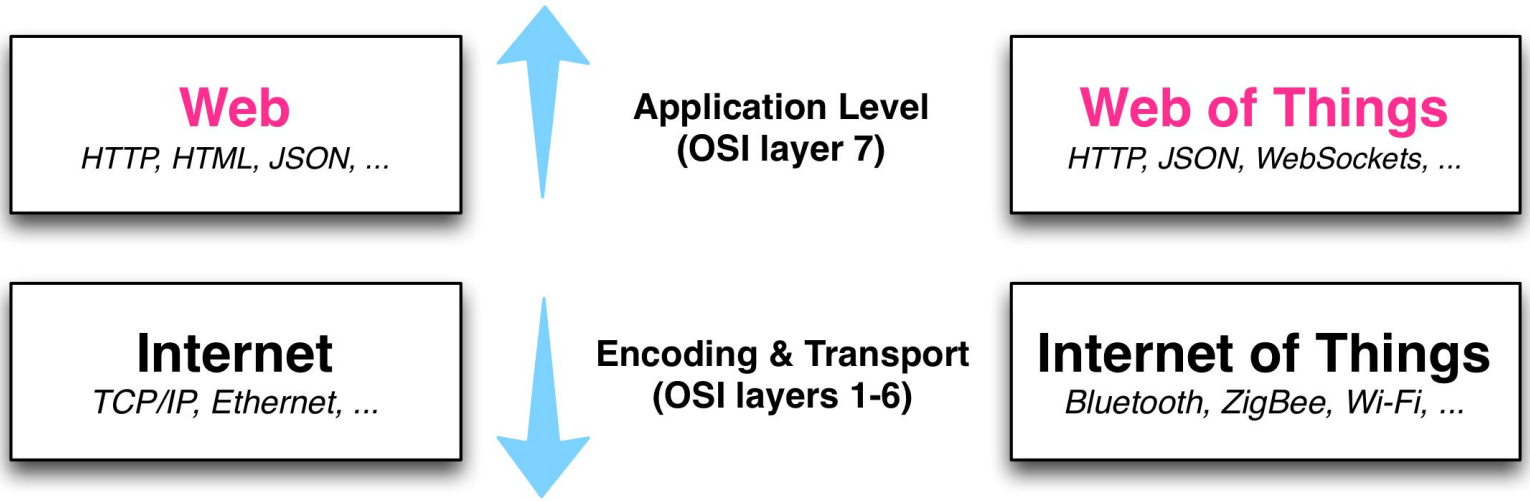


“ Yeah, sure.
Not! ”

[The embedded/IoT community]



Easier to program, faster to integrate data and services, simpler to prototype, deploy, and maintain large systems



More lightweight and optimized for embedded devices (reduced battery, processing, memory and bandwidth usage), more bespoke and hard-wired solutions

Source: Building the Web of Things: book.webofthings.io

Researching the Web of Things...

Diss. ETH No. 19890

Building Blocks for a Participatory Web of Things: Devices, Infrastructures, and Programming Frameworks

A dissertation submitted to the
ETH Zurich

for the degree of
Doctor of Science

Presented by
Mihai Vlad Trifa
Dipl. Ing. EPFL
Born on 12 March 1982, in Oradea, Romania
citizen of Bex (VD), Switzerland

accepted on the recommendation of
Prof. Dr. Friedemann Mattern, examiner
Prof. Dr. Cesare Pautasso, co-examiner
Prof. Dr. Gero Mühl, co-examiner

2011

Diss. ETH No. 19891

A Web of Things Application Architecture - Integrating the Real-World into the Web

A dissertation submitted to
ETH Zurich

for the degree of
Doctor of Science

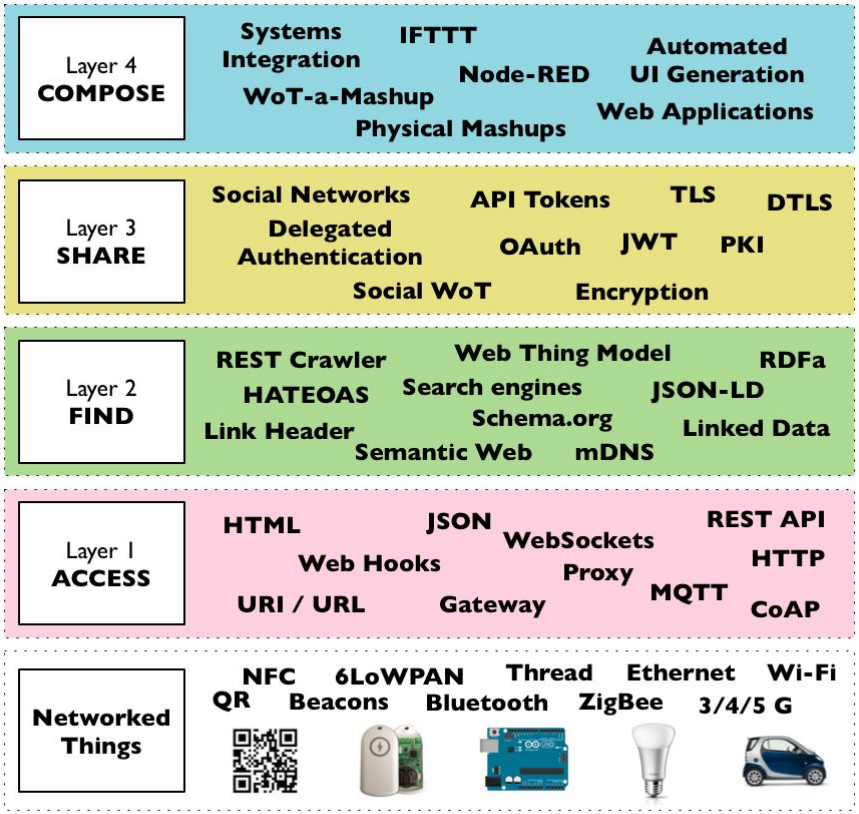
Presented by
Dominique Guinard
M.Sc. in Computer Science, University of Fribourg
born February 27, 1981
citizen of Switzerland

accepted on the recommendation of
Prof. Dr. Friedemann Mattern, examiner, ETH Zurich
Prof. Dr. Gustavo Alonso, co-examiner, ETH Zurich
Prof. Dr. Sanjay Sarma, co-examiner, MIT Boston

2011

Get the theses from: http://webofthings.org/publications

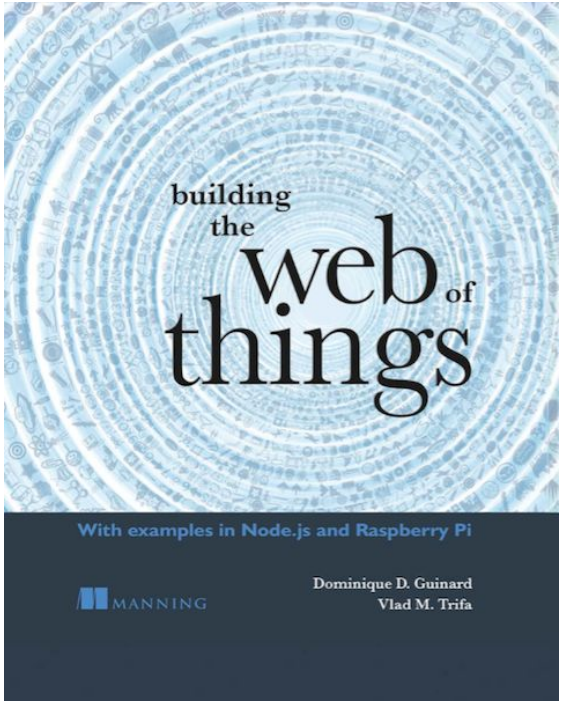
The Web of Things Architecture



Source: Building the Web of Things: book.webofthings.io

- Converge all the Things towards Web protocols!
  - Web Gateway
- WoT principles:
- Reuse the Web!
- Unless:
  - Battery powered
  - Very low-power
  - Need for a mesh
- => Choose Web protocols
  - HTTPS, WSS, etc.

WoT in use @EVRYTHNG on billions of products!



> *IoT needs an application layer, and leveraging the web is the right thing to do! This terrific book will show you how to get there in a few weeks.*
>
> Sanjay Sarma, AutoID Labs, MIT

> *Dom and Vlad are thought leaders in IoT, focused on how to achieve results in practice.*
>
> Andy Chew, Cisco UK

> *A complex subject covered in detail from beginning to end ... very readable too!*
>
> Steve Grey-Wilson, Thingworx, A PTC Business

sparkfun™ ELECTRONICS

Pi Supply

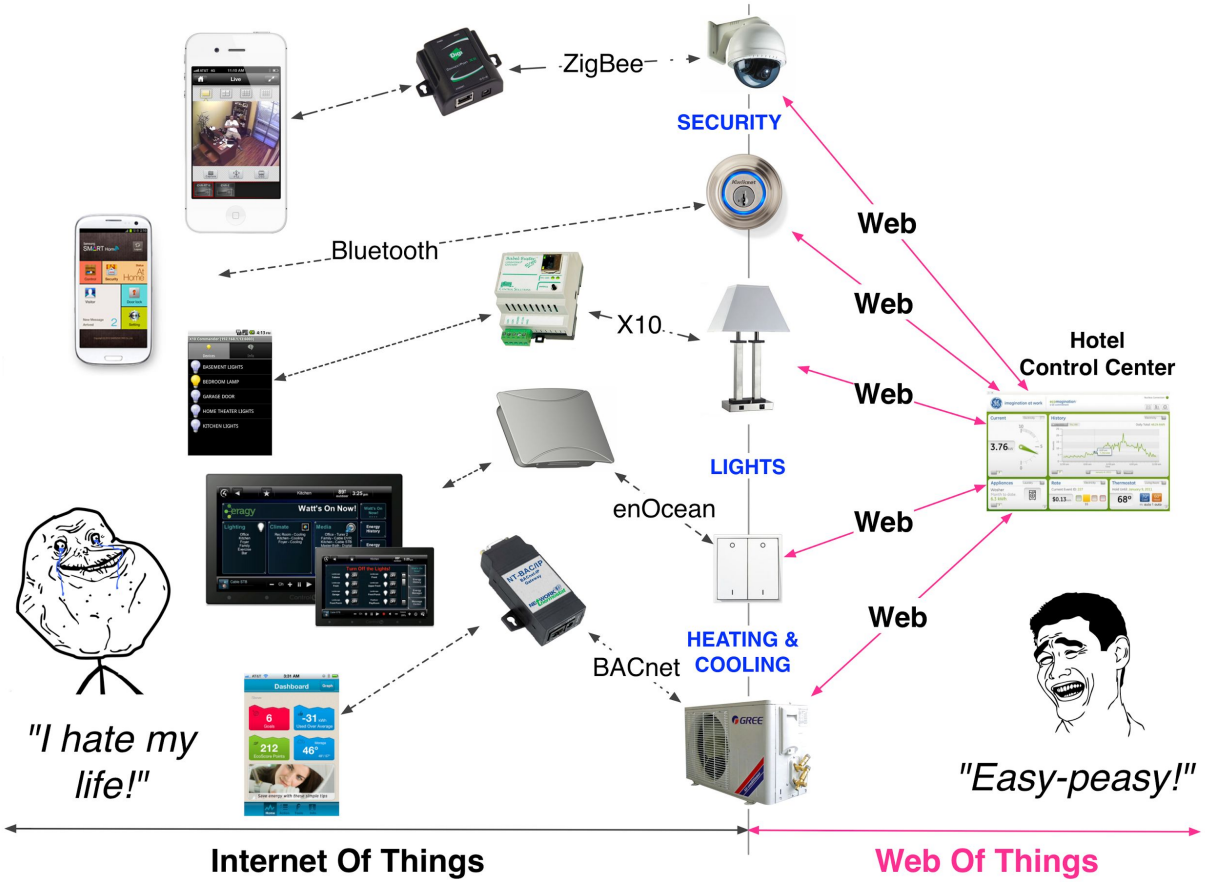
39% off “Building the Web of Things” with code “39guinard” on:

http://book.webofthings.io

Discover the WoT

Chapters 1 - 2

Chapter 1: IoT vs WoT



Source: Building the Web of Things: book.webofthings.io
Creative Commons Attribution 4.0

Chapter 2: Browsing Things

http://devices.webofthings.io

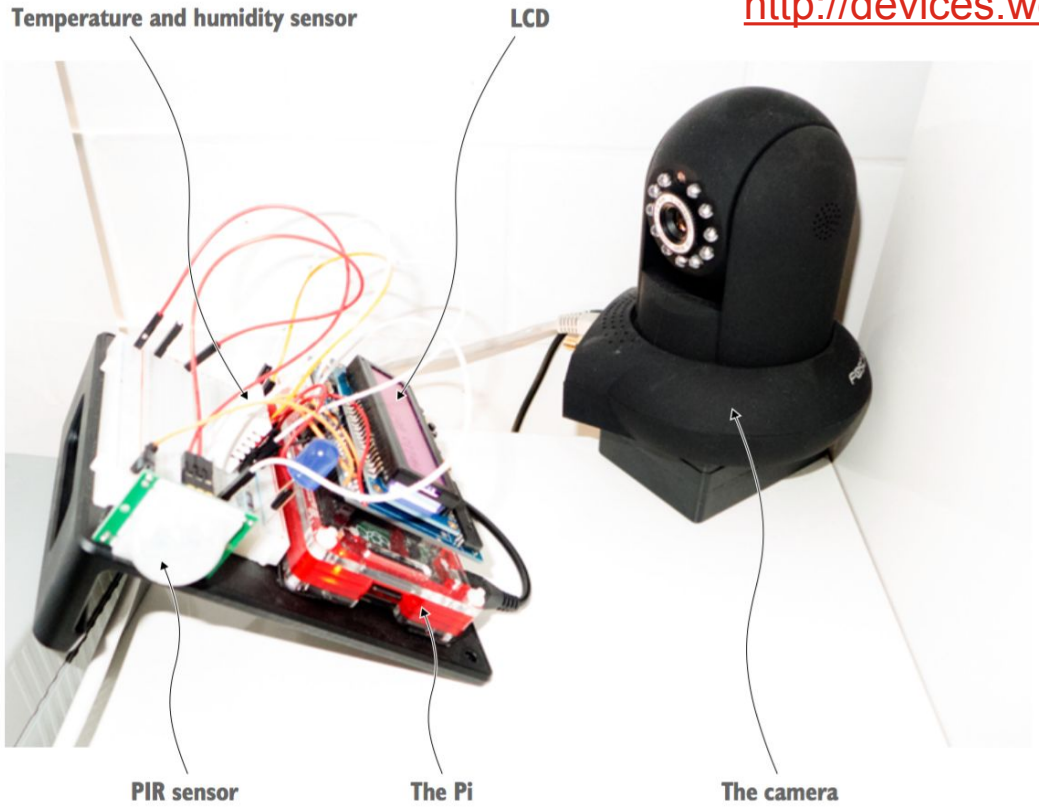


Figure 2.1 The Raspberry Pi and webcam you are accessing as they are set up in our London office


Great experienc
e. thanks!


KonaBoulder, CO
< 24.3


Happy Birthday,
Ellen!


Variety is the
spice of life!


BrucenTaipei, T
W < 0.00


PaulaBelfast, N
I < 24.3


thanks for the
great book


Love your book!


from Rosario,
ARGENTINA

Interact with Things in a few lines of JS code!


```
$(document).ready( //#A
 function doPoll() {
   $.getJSON('http://devices.webofthings.io/pi/sensors/temperature', //#B
     function (data) { //#C
       console.log(data);
       $('#temp').html(data.value + ' ' + data.unit); //#D
       setTimeout(doPoll, 5000); //#E
   });
 });
//#A Wait until the page is loaded and then call doPoll()
//#B Use the AJAX helper to get the JSON payload from the temperature sensor
//#C When the response arrives, this function is called
//#D Select the "temp" HTML element and update its content using the data.value (the value) and data.unit
    (the unit) returned in the JSON payload
//#E The doPoll() function sets a timer to call itself again in 5 seconds (5000 milliseconds)
```

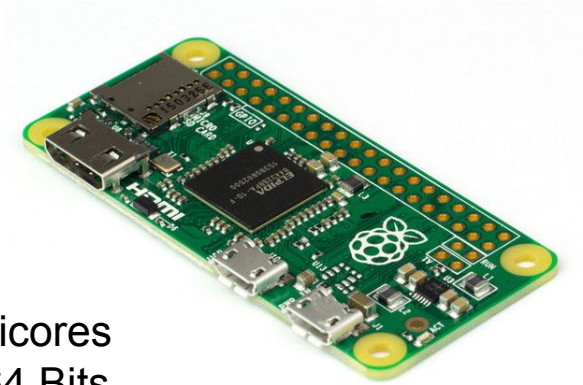
Lab 1: Hello WoT: The Web as an Application Layer

See also: Chapter 2 from page 36

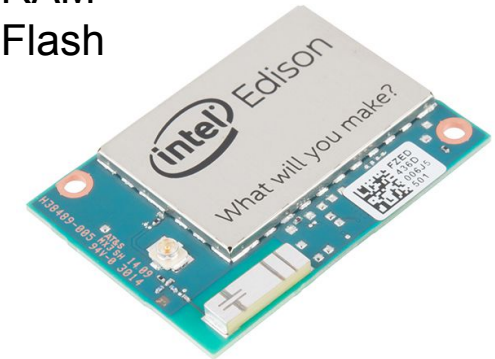
1. Fork & Clone the book code:
   - `git clone https://github.com/webofthings/wot-book --recursive`
   - Try pushing from your machine, pulling from the Pi
2. Browse the device as a *Human* on: http://devices.webofthings.io/
3. Install Postman and browse the device as an *App*
   - URL: http://devices.webofthings.io/
   - `Accept: application/json`
4. Modify 2.2 code to get the humidity value every 5 seconds
5. *Bonus: change the type of graph in 2.2*

Embedded Systems & Node

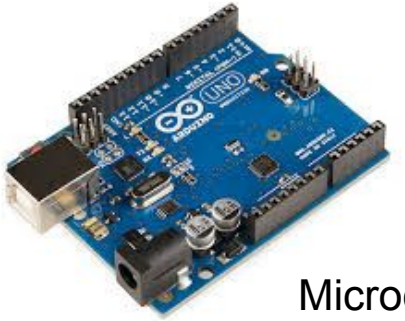
Chapter 3 - 4



Multicores
32-64 Bits
X GB of RAM
X GB of Flash



VS

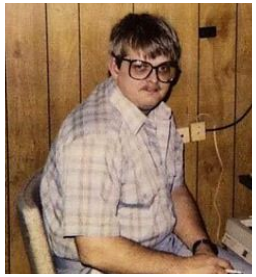


Microcontroller
8 Bits
X KB of RAM
X KB of ROM

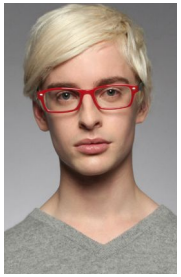
Table 4.1 An overview of some IoT embedded platforms. Platforms targeting hobbyists usually cost more but also have more resources (RAM, CPU, and so on). Industrial platforms tend to offer lower specifications but the costs are usually lower.

Brand	Models	CPU	RAM	+	Price	Type	Connectivity
Arduino	20+ and many clones (Spark, Intel, and so on)	ATmega, 8–64 MHz, Intel Curie, Linino	16 KB–64 MB	Largest community	~30 USD	RTOS, Linux, hobbyists	Pluggable extension boards (Wi-Fi, GPRS, BLE, ZigBee, and so on)
Raspberry Pi	A, A+, B, B+, 2, 3, Zero	ARMv6 or v7, 700 MHz -1.2 GHz	256–1 GB	Full Linux, GPU, large community	~5-35 USD	Linux, hobbyists	Ethernet, extension through USB, BLE (Pi3)
Intel	Edison	Intel Atom 500 MHz	1 GB	X86, full Linux	~50 USD	Linux, hobbyist to industrial	Wi-Fi, BLE
BeagleBoard	BeagleBone Black, X15, and so on	AM335x 1 GHz ARMv7	512 MB–2 GB	Stability, full Linux, SDK	~50 USD	Linux, hobbyist to industrial	Ethernet, extension through USB and shields
Texas Instruments	CC3200, SoC IoT, and so on	ARM 80 MHz, etc.	from 256 KB	Cost, Wi-Fi	<10 USD	RTOS, industrial	Wi-Fi, BLE, ZigBee
Marvell	88MC200, SoC IoT, and so on	ARM 200 MHz, etc.	from 256 KB	Cost, Wi-Fi, SDK	<10 USD	RTOS, industrial	Wi-Fi, BLE, ZigBee
Broadcom	WICED, and so on (also at the heart of the Raspberry PIs)	ARM 120 MHz, and so on	from 256 KB	Cost, Wi-Fi, SDK	<10 USD	RTOS, industrial	Wi-Fi, BLE, ZigBee, Thread

Chapter 3: Node.js for Embedded Devices? Really?



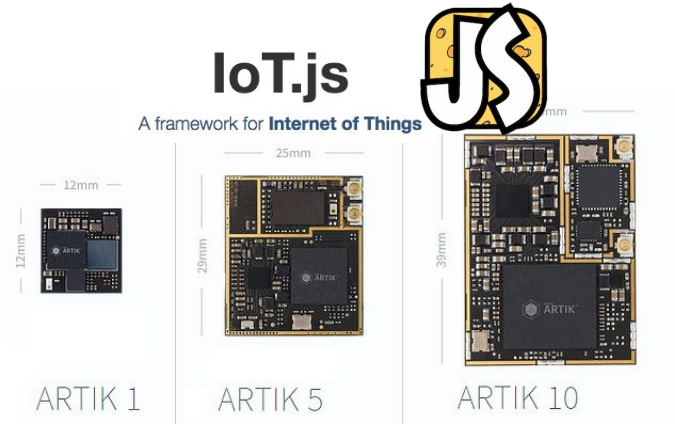
- Before:
  - C rules
  - Windows based IDEs, 1 per platform
  - Small community, highly specialized
  - Very resource constrained devices
  - Integration via specialized SDKs



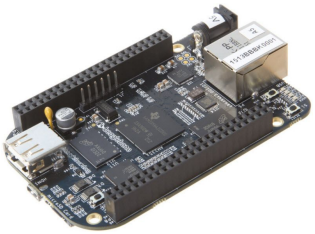
- After
  - Node.js is taking over!
  - Larger community, more reach, more innovation
  - Huge ecosystem of libraries
  - Integration via the Internet and the Web

Node on embedded devices: Hardware support

Espruino

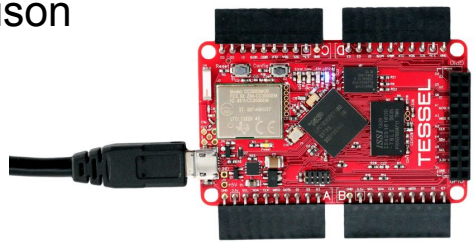


Artik



Beaglebone

Edison

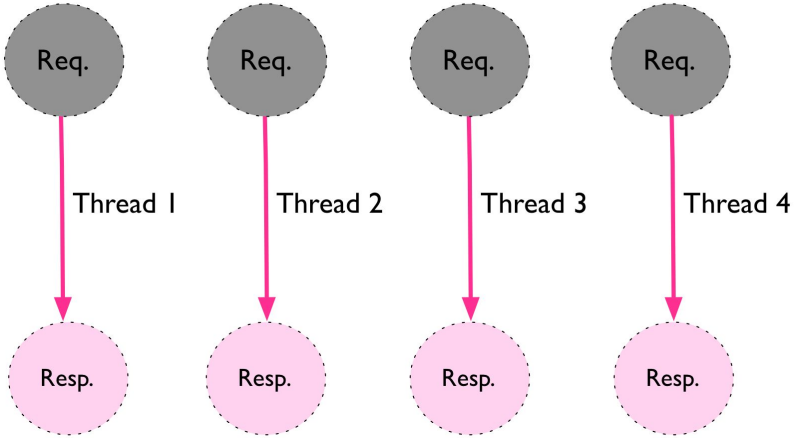


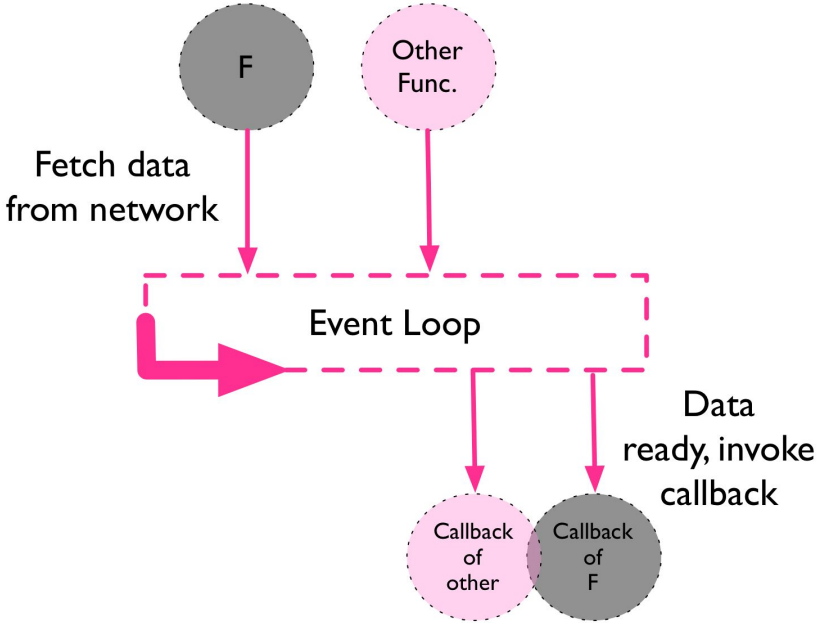
Tessle

Kinoma

Raspberry Pi
(Pi Zero incl.)

Why Node?


Req.
Req.
Req.
Req.
Thread 1
Thread 2
Thread 3
Thread 4
Resp.
Resp.
Resp.
Resp.


F
Other Func.
Fetch data from network
Event Loop
Data ready, invoke callback
Callback of other
Callback of F

Lab 2: Hello Node.js

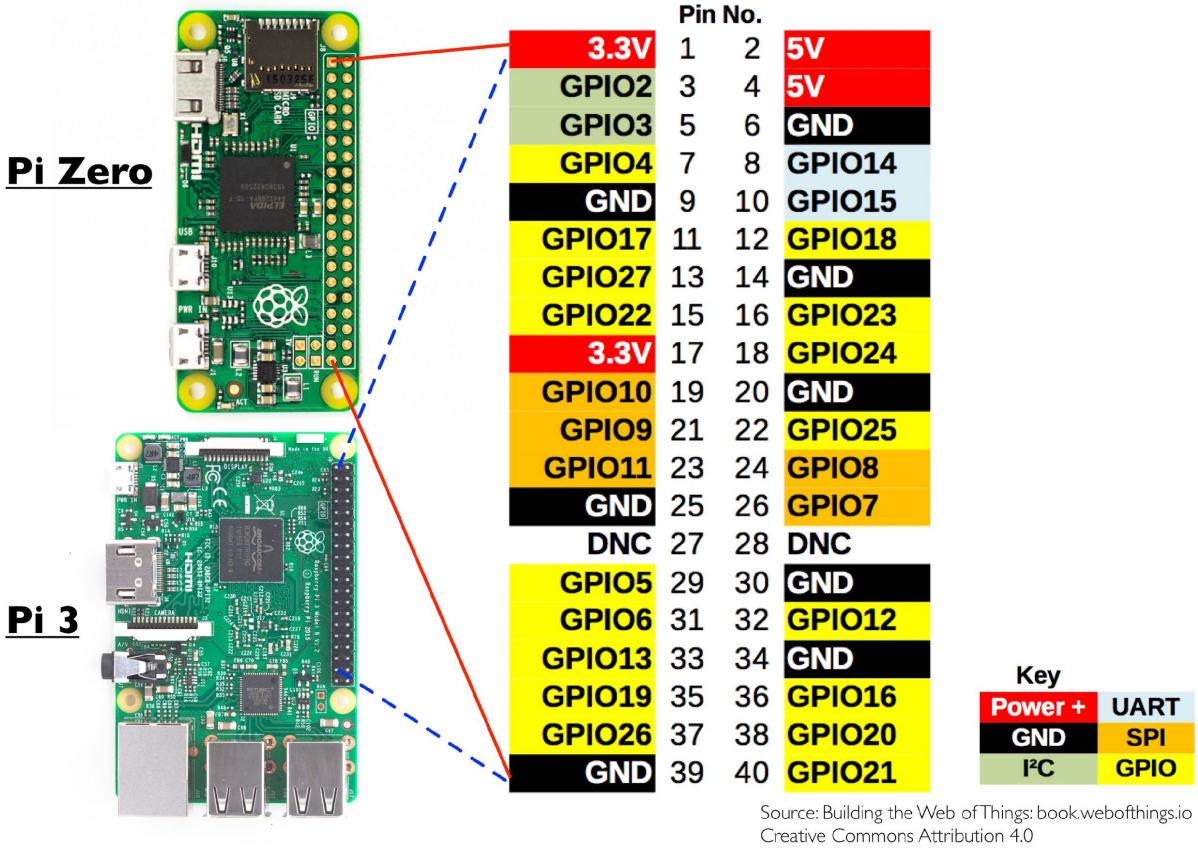
See also: Chapter 4, page 98

1. Install NVM & Node.js on your Pi and computer
   - `curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.2/install.sh | bash`
   - `nvm install v4.8.3`
2. Build your (first?) Node HTTP server
3. *Bonus: build a more advanced server, see Listing 3.2 page 66*

GPIOs: sensing & actuating

Chapter 4

Chapter 4: Sensors, Actuator & GPIOs



GPIO support via Node on Embedded Systems

http://johnny-five.io

https://github.com/intel-iot-devkit/mraa

https://cylon.js

https://github.com/fivdi/onoff

heimcontrol.js

http://ni-c.github.io/heimcontrol.js/

https://github.com/webofthings/webofthings.js

An example with On/Off: connecting a PIR sensor

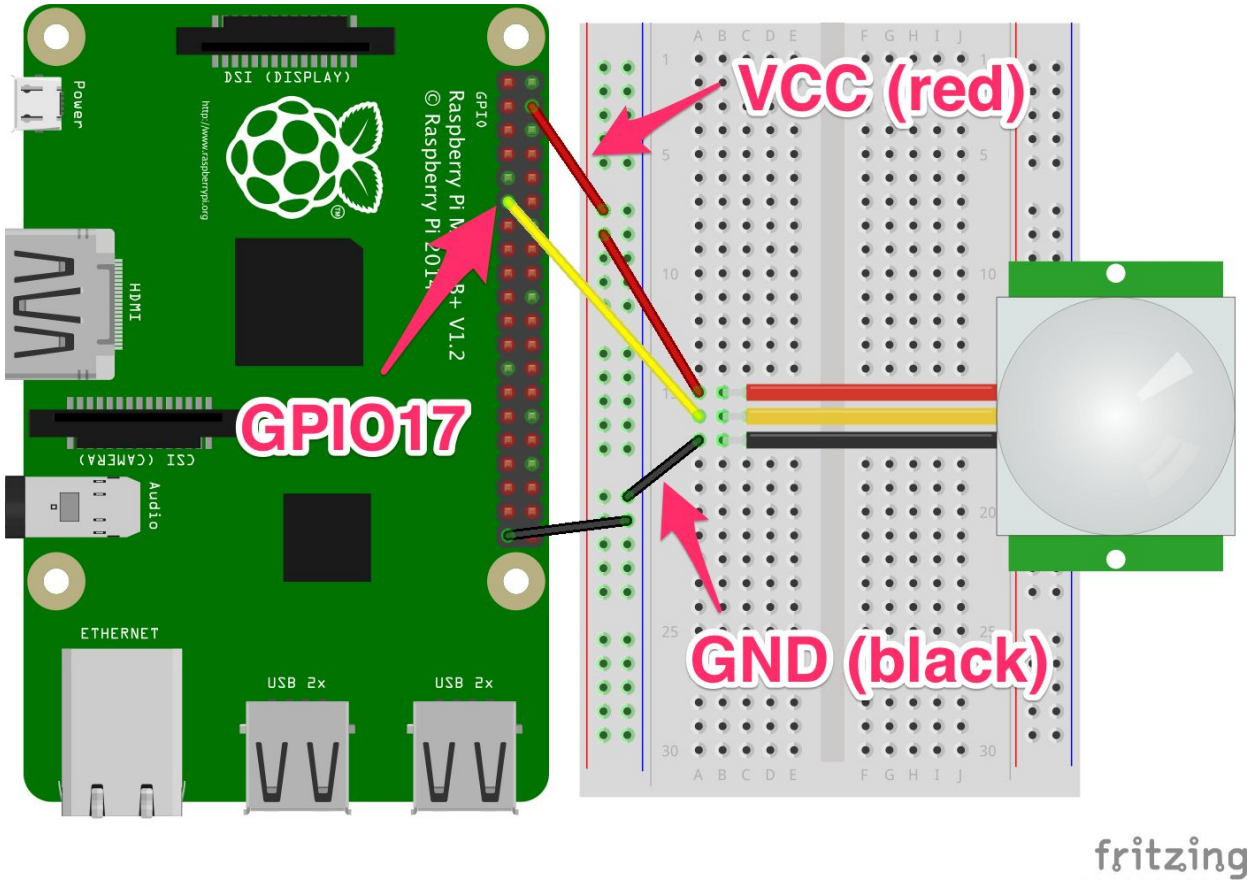
```
var Gpio = require('onoff').Gpio,
 sensor = new Gpio(17, 'in', 'both');    //#A
sensor.watch(function (err, value) { //#B
 if (err) exit(err);
 console.log(value ? 'there is someone!' : 'not anymore!');
});
function exit(err) {
 if (err) console.log('An error occurred: ' + err);
 sensor.unexport();
 console.log('Bye, bye!')
 process.exit();
}
process.on('SIGINT', exit);

// #A Initialize pin 17 in input mode, 'both' means we want to handle both rising and falling interrupt edges
// #B Listen for state changes on pin 17, if a change is detected the anonymous callback function will be called
     with the new value
```

Lab 3: GPIO & Node

See also: Chapter 4, from page 102

1. Setup the PIR sensor (see page 104)
2. Connect it to onoff.js code (see `chapter4-gpios/pir.js`)
3. *Bonus: setup the DHT sensor (see Chapter 4 from page 105)*

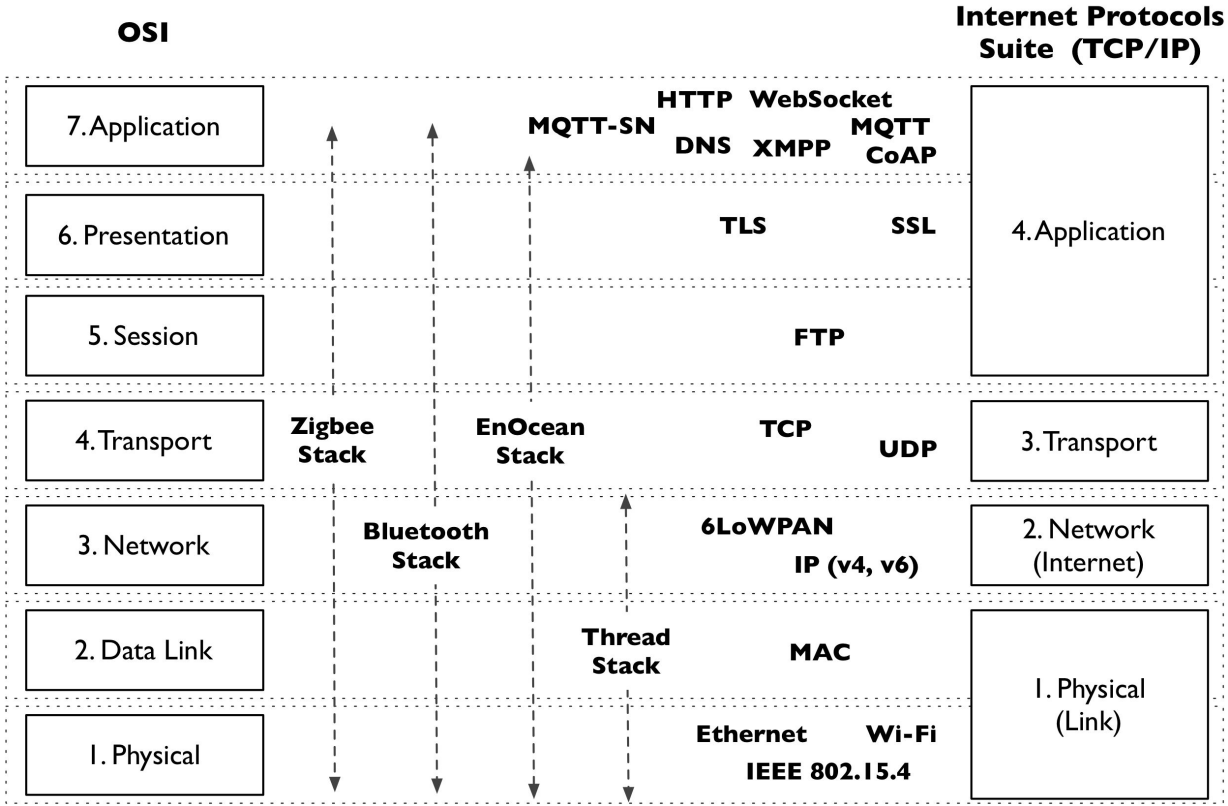
Wiring the PIR sensor



IoT Networks

Chapter 5

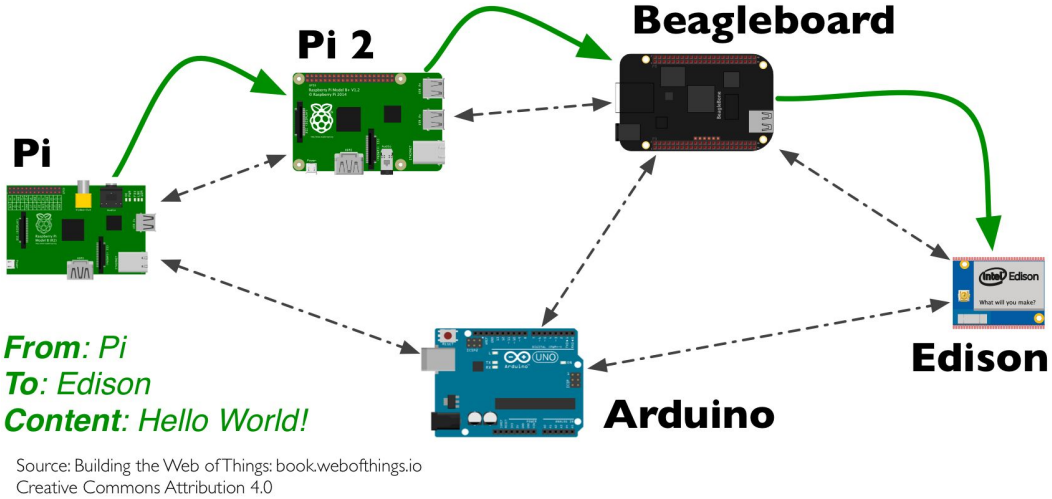
Different Protocol Stacks



Source: Building the Web of Things: book.webofthings.io
Creative Commons Attribution 4.0

The 2 valid reasons for not choosing IP+Web end-to-end

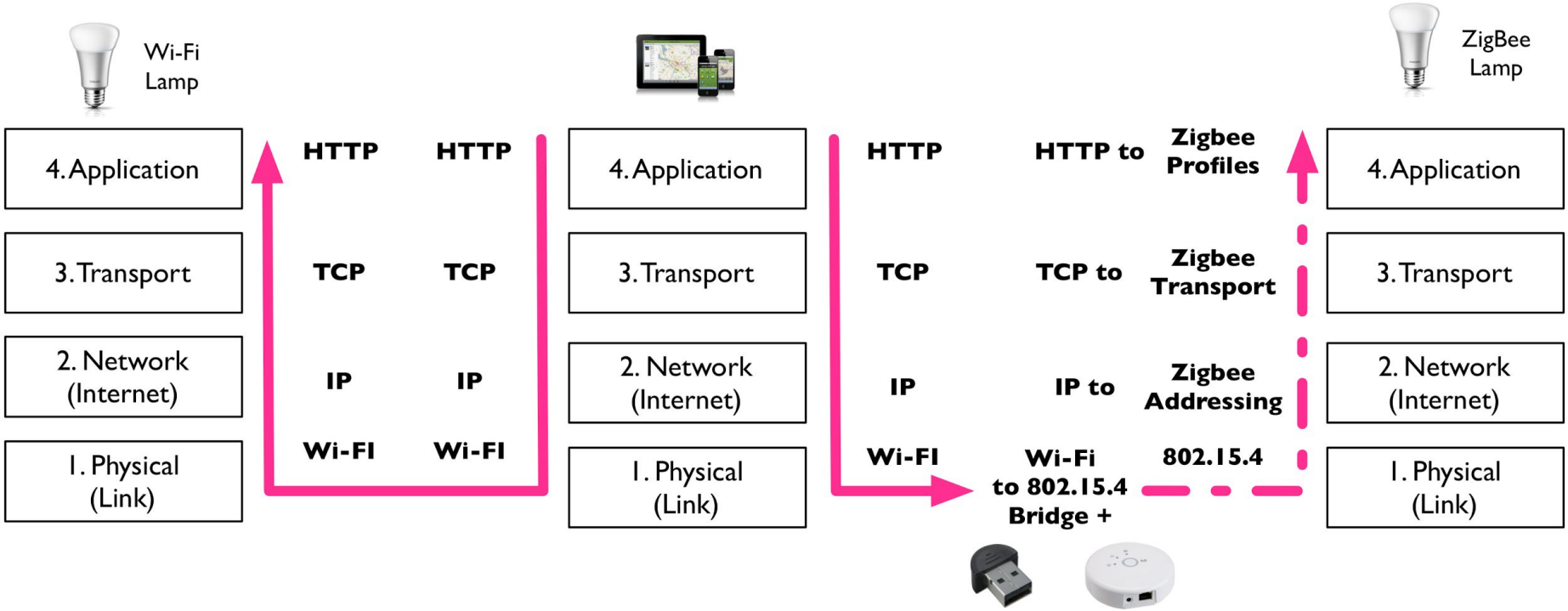
Battery Powered Devices



Source: Building the Web of Things: book.webofthings.io
Creative Commons Attribution 4.0

Deployment requires a mesh

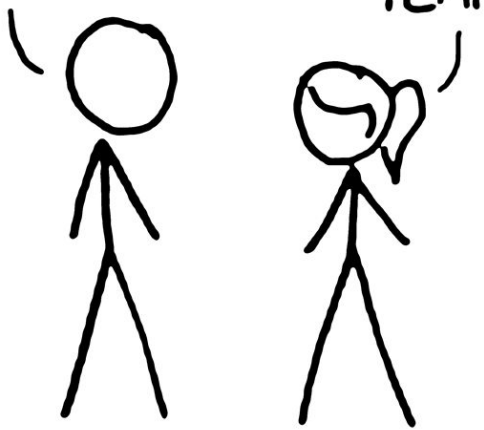
Web vs Not Web: Why should I care?



HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)

SITUATION:
THERE ARE
14 COMPETING
STANDARDS.

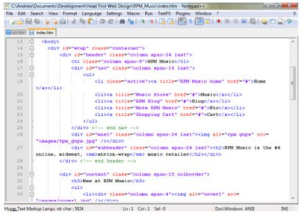
14?! RIDICULOUS!
WE NEED TO DEVELOP
ONE UNIVERSAL STANDARD
THAT COVERS EVERYONE'S
USE CASES.

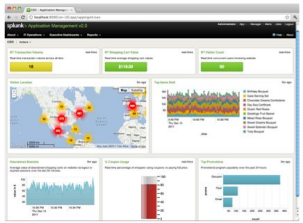


SOON:

SITUATION:
THERE ARE
15 COMPETING
STANDARDS.

Source: http://xkcd.com





Web

WWW/LAN

GET

http://geneva.ch/weather/

PUT

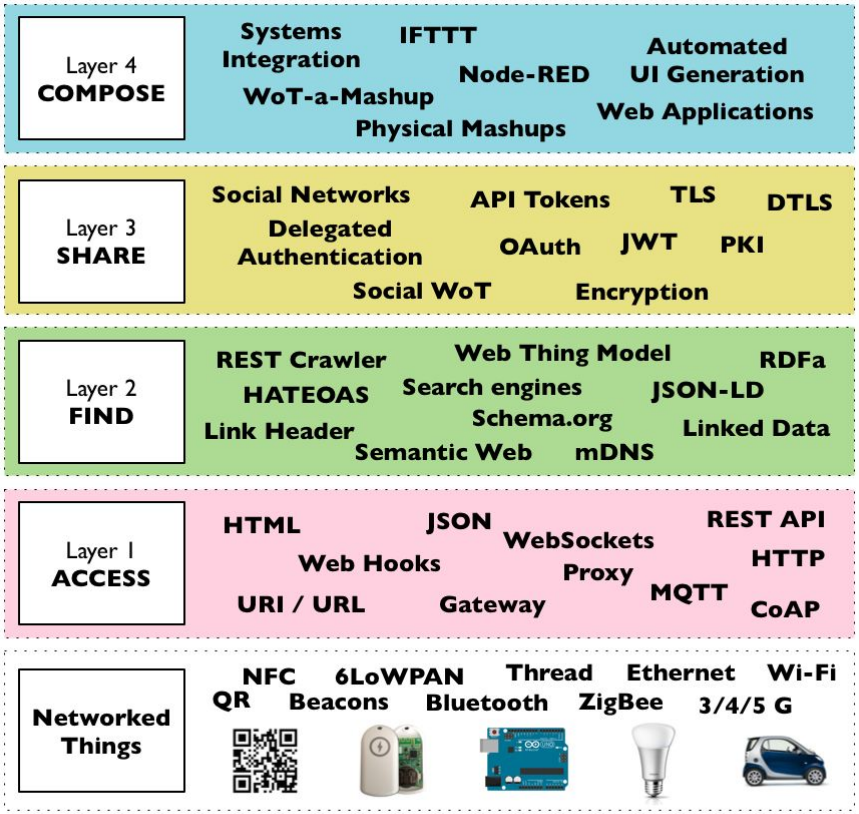
http://hotel.ar/room105/lock

GET

http://myhome.london/fire/alerts.rss

- Web Developers
- Native/Desktop Apps
- Web services & APIs
- Analytics, storage

The Web of Things Architecture

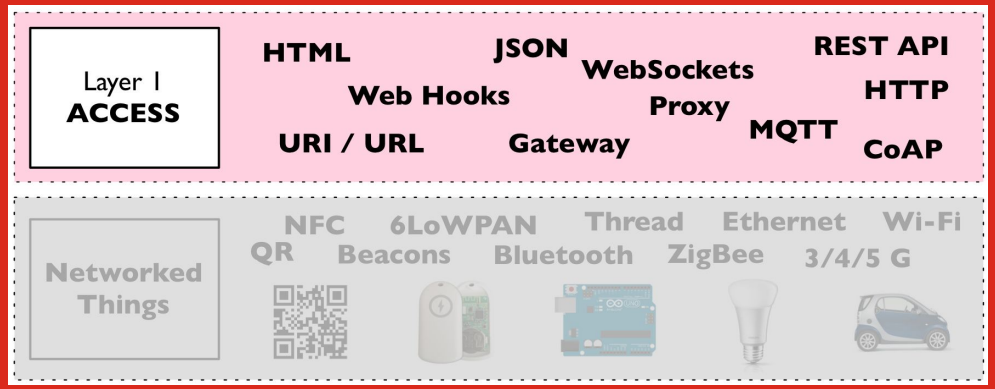


Source: Building the Web of Things: book.webofthings.io

- Converge all the Things towards Web protocols!
  - Web Gateway
- WoT principles:
- Reuse the Web!
- Unless:
  - Battery powered
  - Very low-power
  - Need for a mesh
- => Choose Web protocols
  - HTTPS, WSS, etc.

WoT Architecture: Access

Chapters 6 - 7



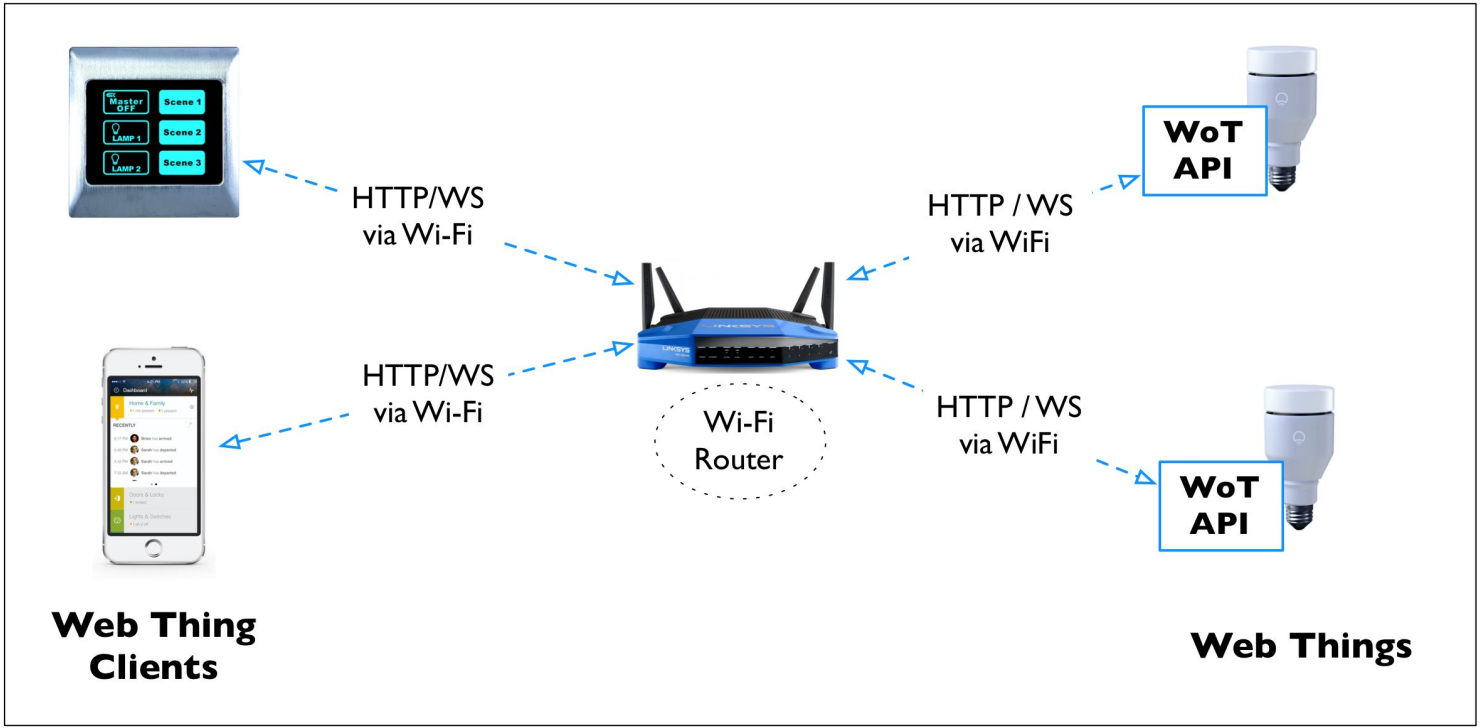
Web API for Things: 5 Steps Design Process

1. ***Integration strategy***—Choose a pattern to integrate Things to the internet and the web.
2. ***Resource design***—Identify the functionality or services of a Thing, and organize the hierarchy of these services.
3. ***Representation design***—Decide which representations will be served for each resource.
4. ***Interface design***—Decide which commands are possible for each service, along with which error codes.
5. ***Resource linking design***—Decide how the different resources are linked to each other.

[Gui2010] Guinard, D., Trifa, V., Wilde, E. *A Resource Oriented Architecture for the Web of Things*. IoT 2010

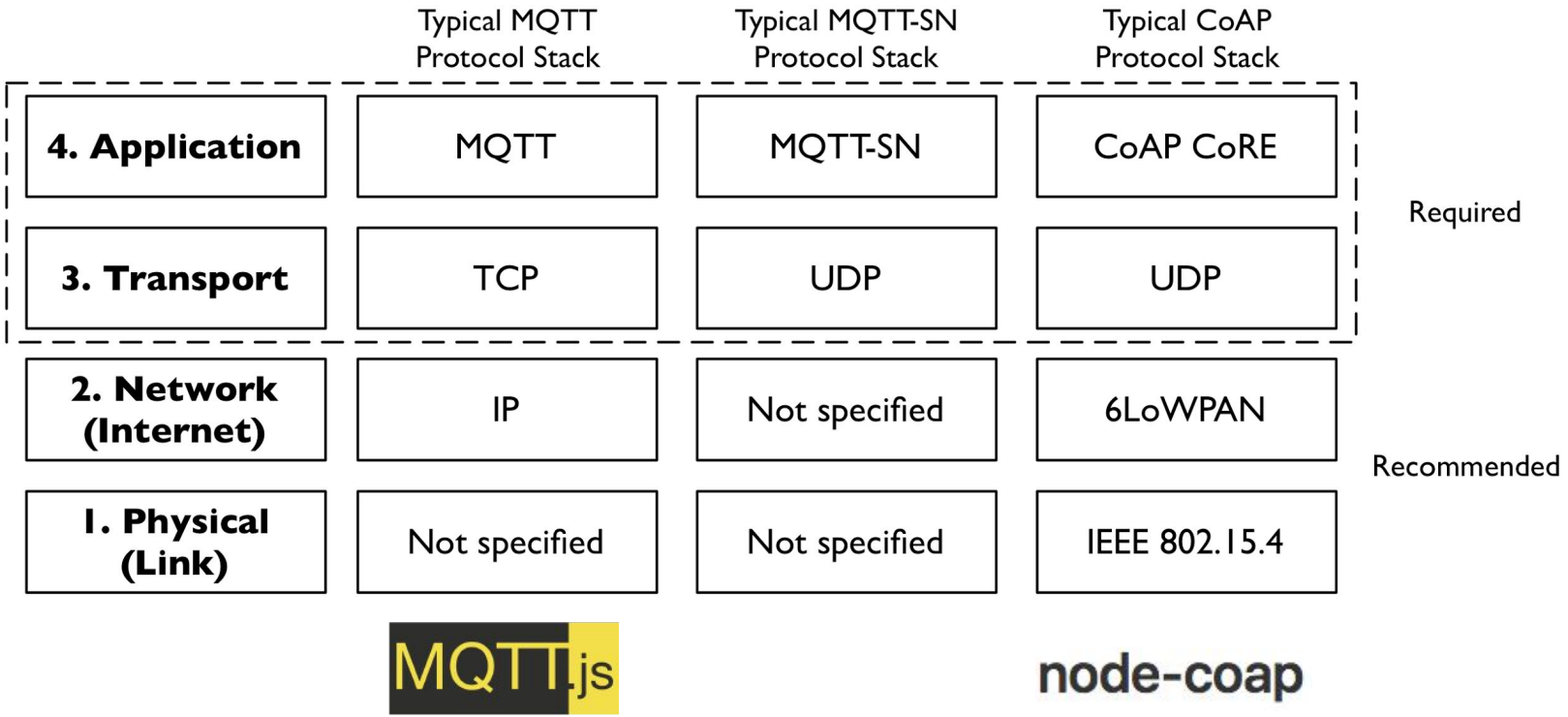
1. Integration Strategy

The Web on Devices!

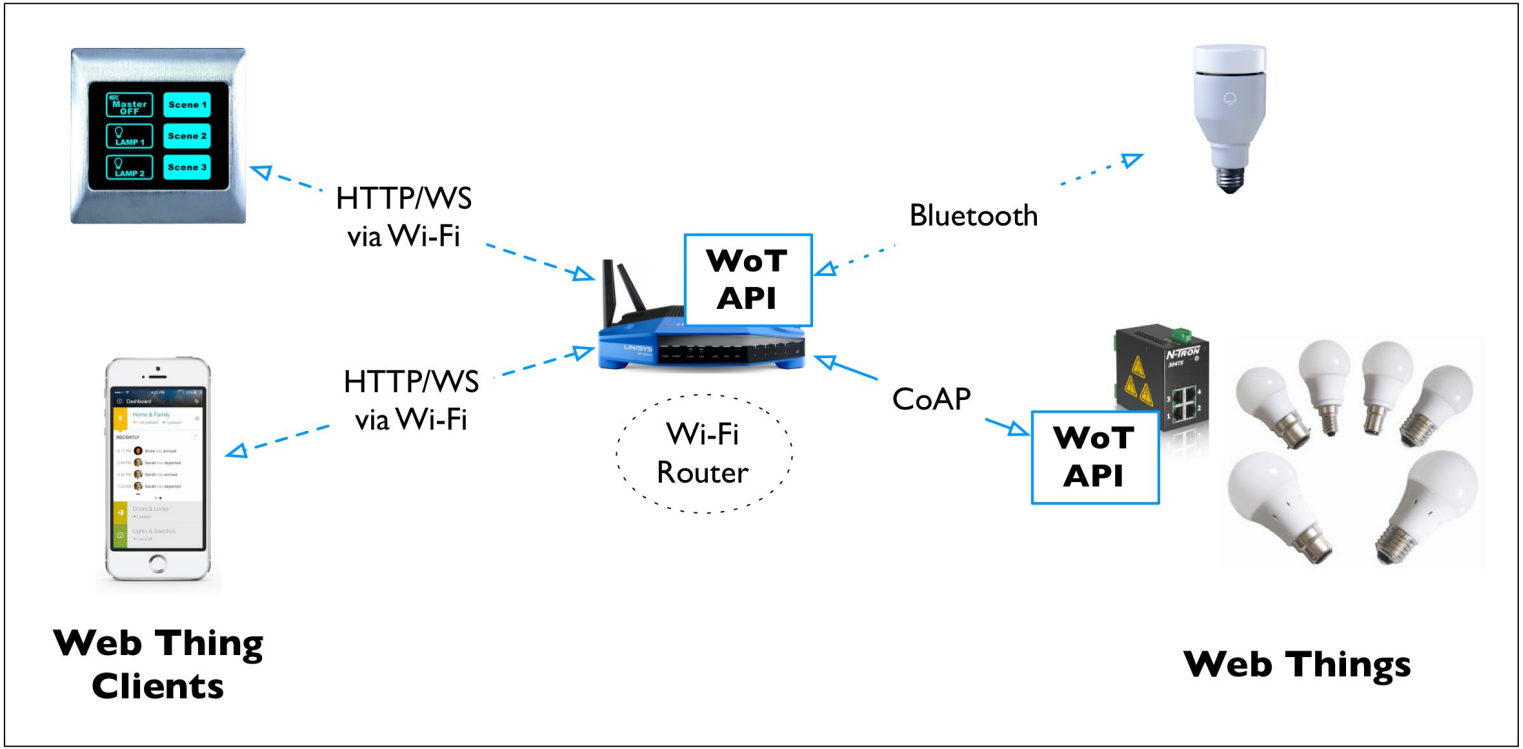


Source: Building the Web of Things: book.webofthings.io

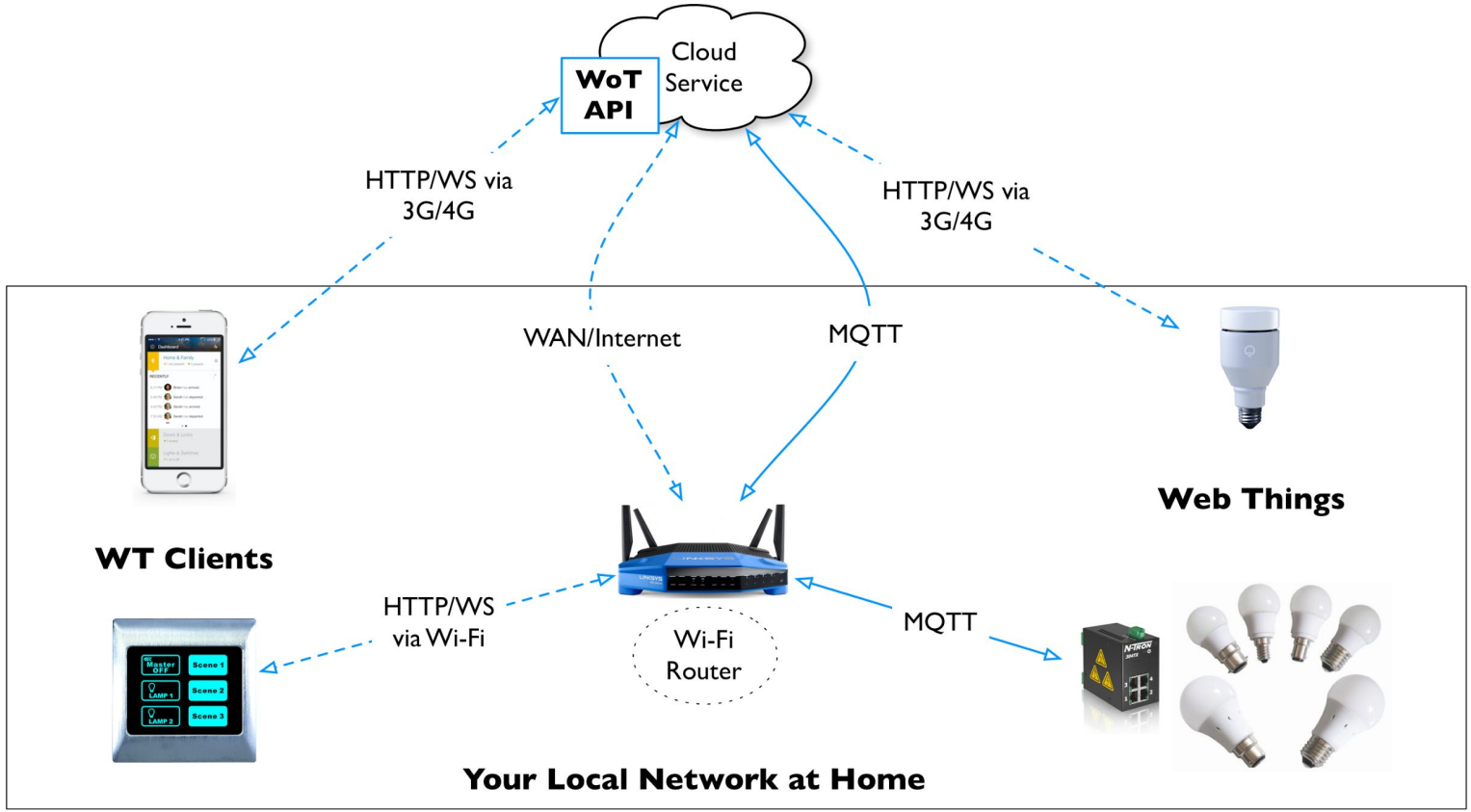
But: Not all devices can speak Web!



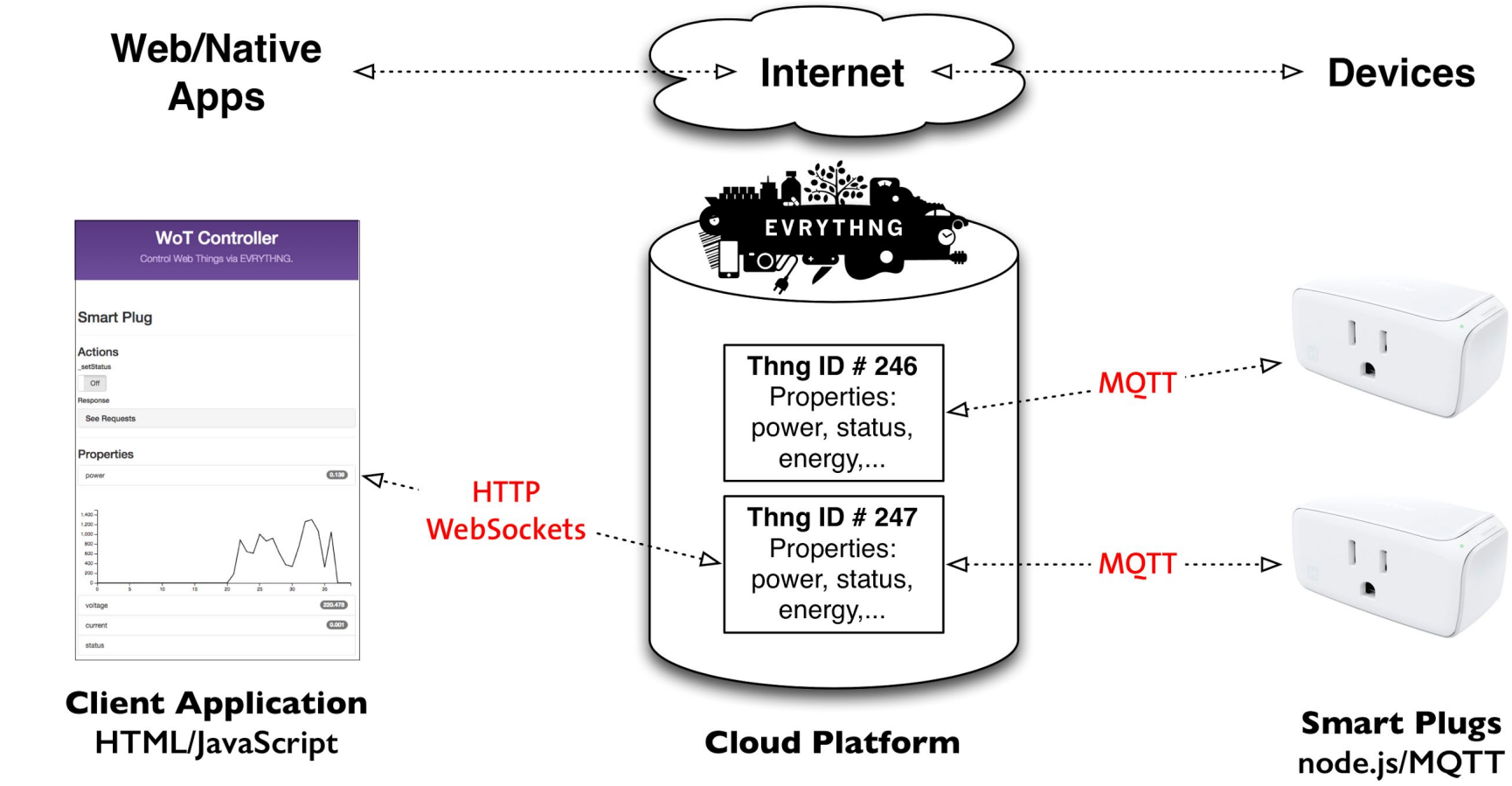
Integration via Gateway



The Cloud as a Gateway (e.g., EVRYTHNG)



Example: EVRYTHNG Smart Products Platform

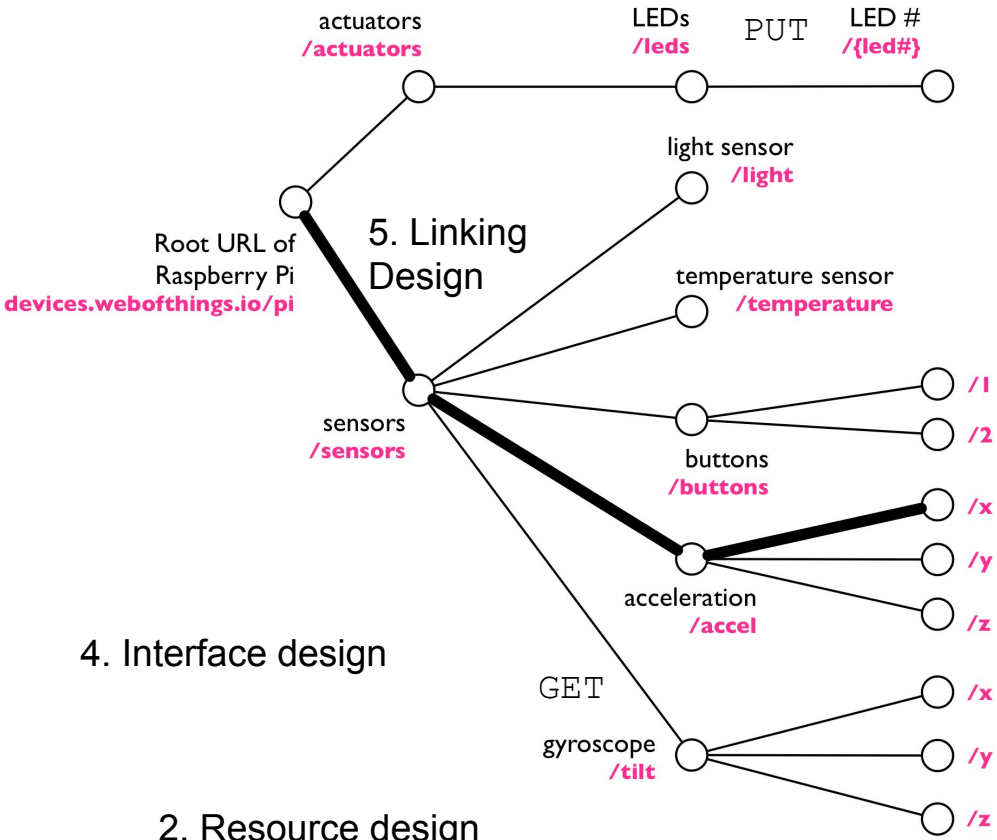


2. Resources, Representations & Links

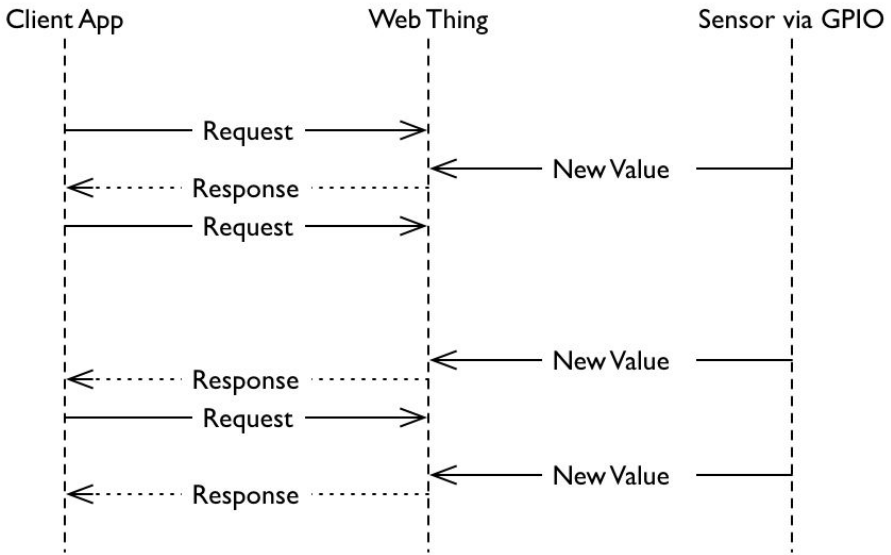
Design Process Applied

3. Representation design

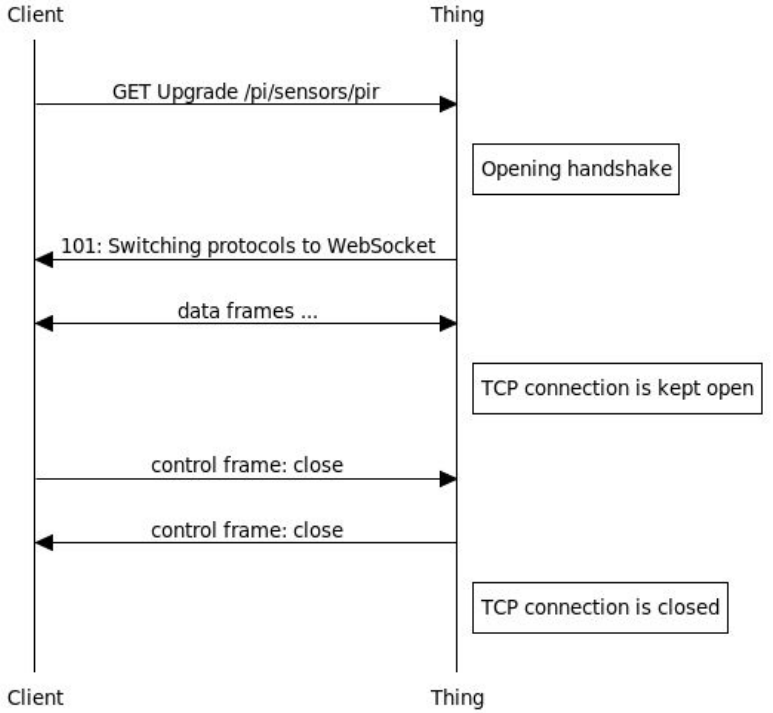
```
Request:
GET /pi
Host:
  devices.webofthings.io
Accept: application/json
Response:
200 OK
Content-Type:
  application/json
{
   "name" : "Pi"
   ...
}
```



Beyond HTTP: Websockets for Event Driven Communication



HTTP 1.1



WebSockets

WebSocket Client

```
function subscribeToWs(url, msg) {
  var socket = new WebSocket(url);

  socket.onmessage = function (event) {
      console.log(event.data);
  };
  socket.onerror = function (error) {
      console.log('An error occurred while trying to connect to a Websocket!');
      console.log(error);
  };
  socket.onopen = function (event) {
      if (msg) {
          socket.send(msg);
      }
  };
}
//subscribeToWs('ws://localhost:8484/pi/sensors/temperature');
```

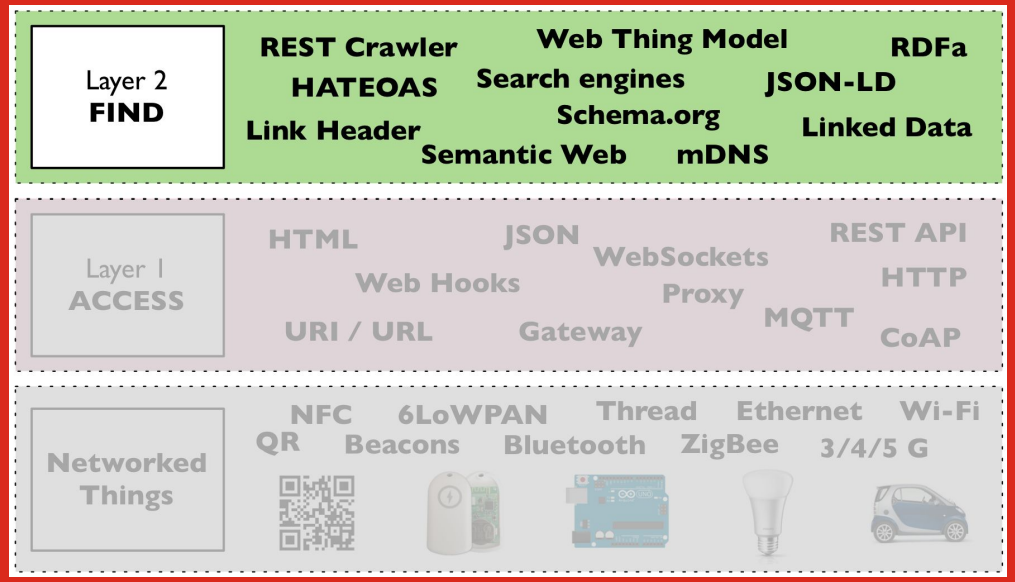
Lab 4: Designing the Pi API

See also: Chapter 7

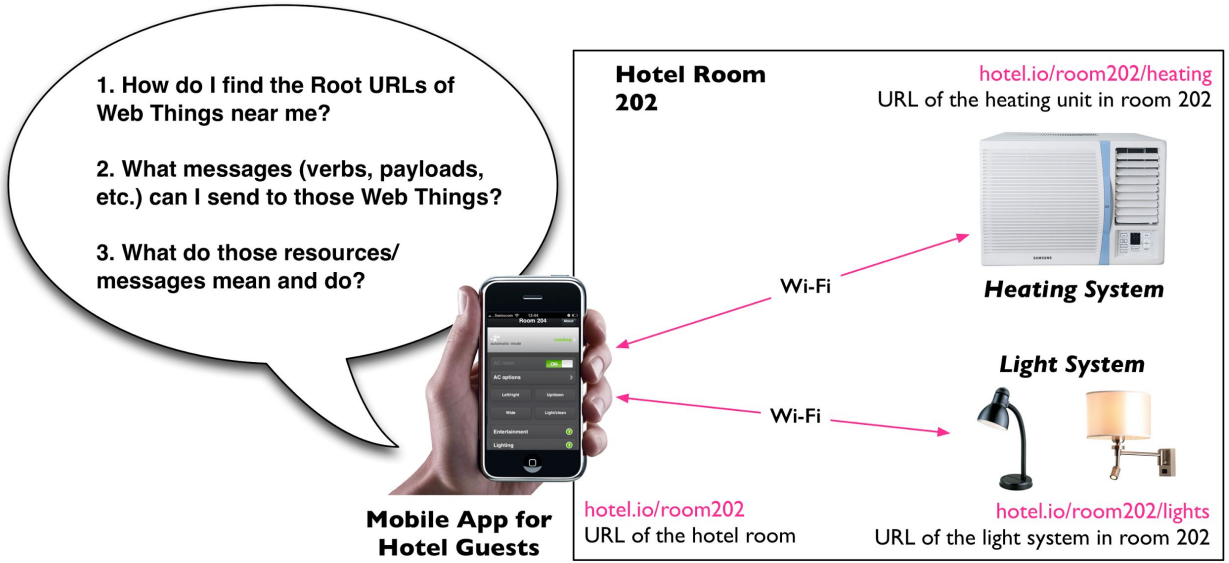
1. Code deep-dive `chapter7-implementation/part1-2-direct-gateway`
   - Resources (add a noise sensor)
   - Representation (see messagepack)
2. Adding a representation
   - Add CBOR support
   - `npm install --save cbor`
   - In `converter.js`
3. Communication via WebSocket
   - `chapter2-hello-wot/client/ex-2.3-websockets-temp-graph.html`
4. *Bonus: bind the PIR sensor of your Pi (see page 183)*

WoT Architecture: Find

Chapter 8



3 Challenges in IoT Findability



1. Bootstrap URL?
2. What's the format? (syntax)
3. What does that mean? (semantics)

The Web can help with all 3!

How to find the URL of a Thing? mDNS!

```
service up:  {
  interfaceIndex: 4,
  type:
   { name: 'http',
     protocol: 'tcp',
     subtypes: [],
     fullyQualified: true },
  replyDomain: 'local.',
  flags: 3,
  name: 'Brother MFC-8520DN',
  networkInterface: 'en0',
  fullname:
'Brother\\032MFC-8520DN._http._tcp.local.',
host: 'EVT-BW-BROTHER.local', The service
port: 80, local IP address addresses: [
'192.168.0.6' ]
}
```

- mDNS clients listen for mDNS messages (on UDP)
- DNS tables are populated from what they catch
- Your Pi broadcasts mDNS messages as we speak!

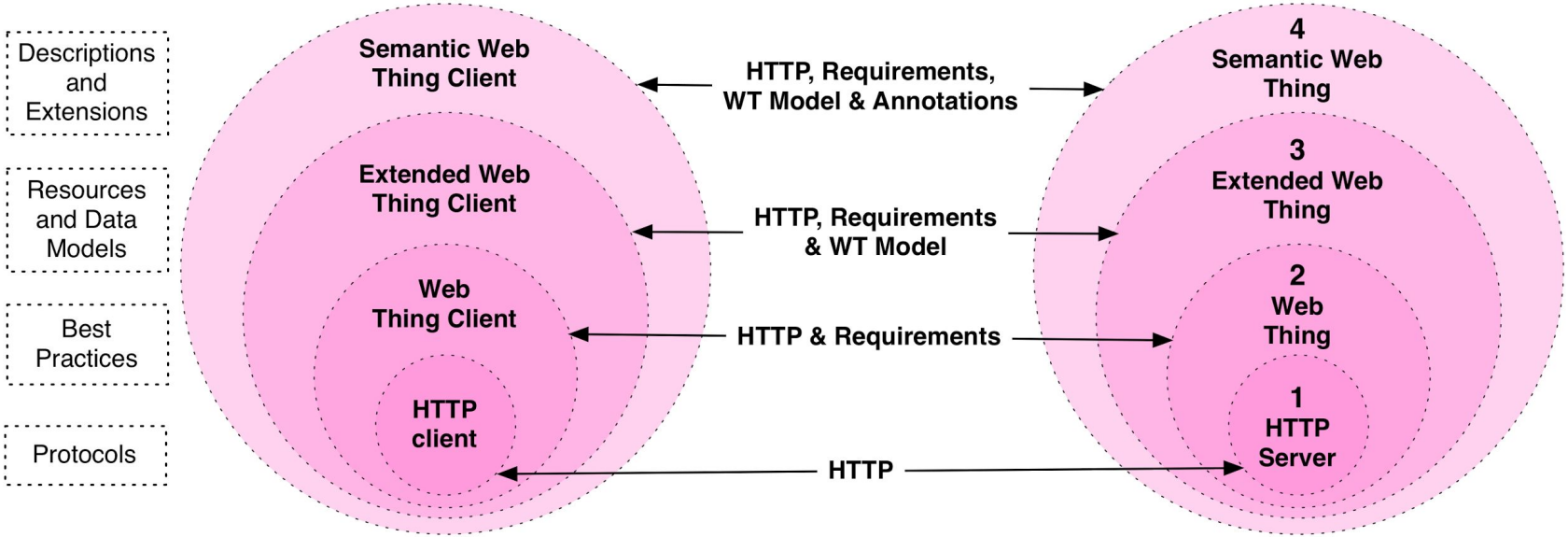
Web Thing Model & Semantic Web

W3C Member Submission

Web Thing Model

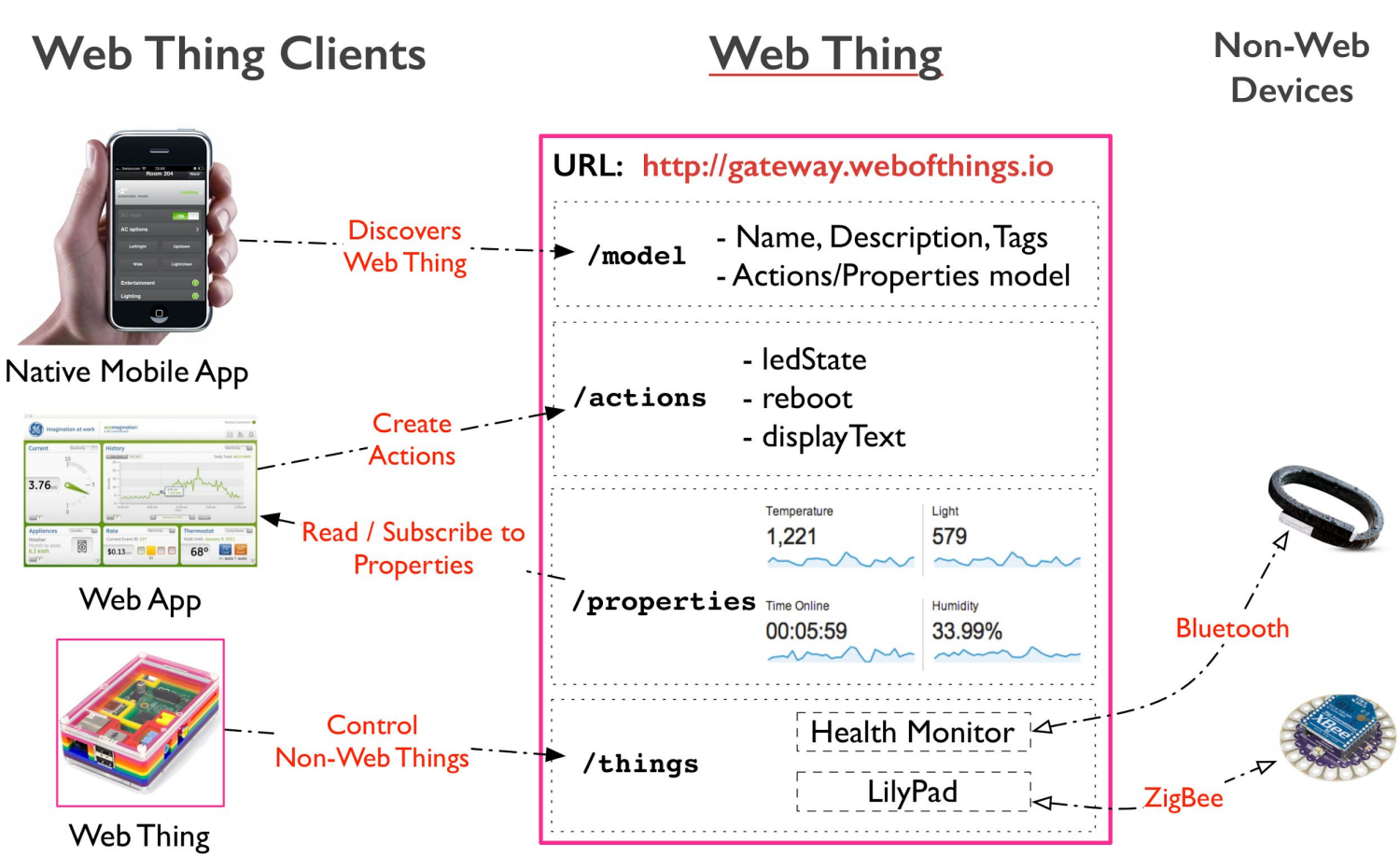
W3C Member Submission 24 August 2015

http://model.webofthings.io
http://gateway.webofthings.io



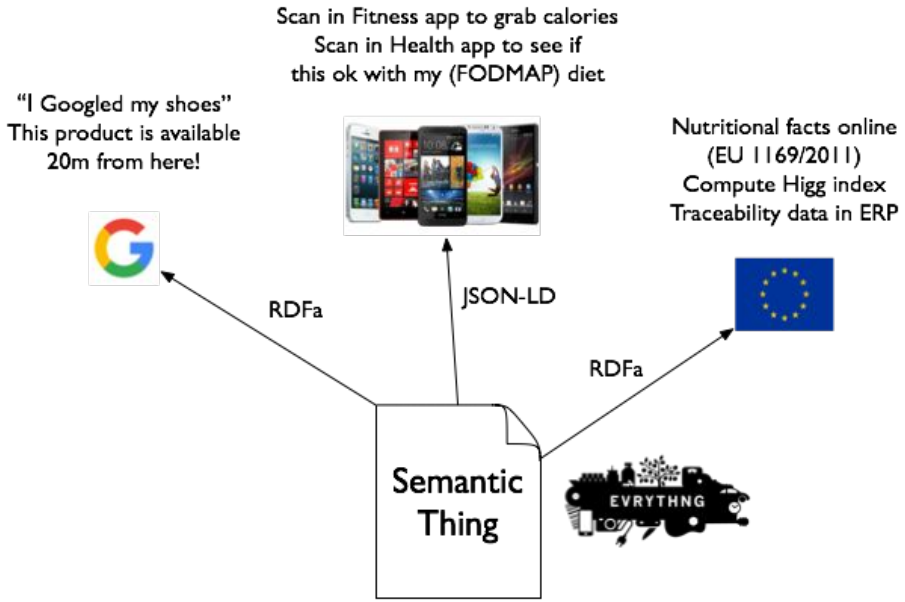
Source: Building the Web of Things: book.webofthings.io
Creative Commons Attribution 4.0

Web Thing Model Resources



Say Hi to the Semantic Web (of Things!)

- Semantic extensions [via JSON-LD]
  - Enhance semantics: What is that Thing really?
  - Schema.org
- Fosters:
  - Findability
  - Interoperability
  - Compliance
- More details:
  - http://model.webofthings.io



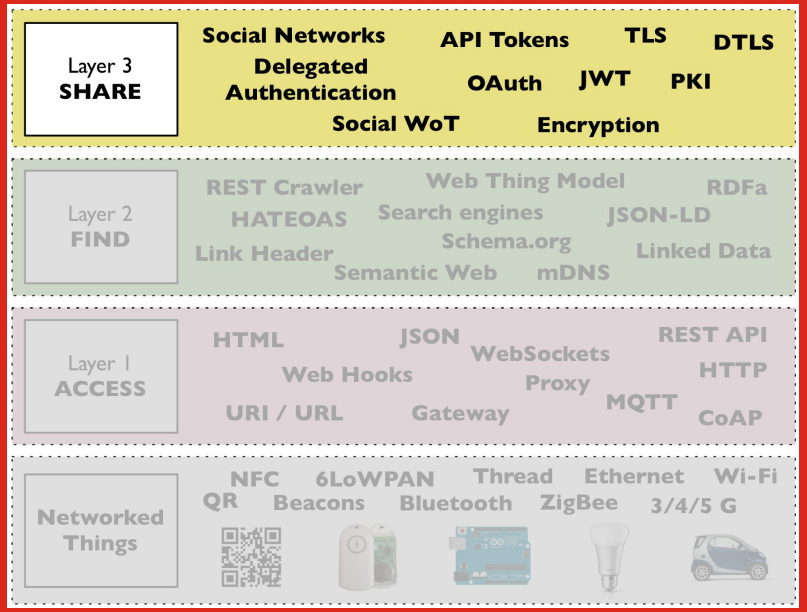
Lab 5: Hello Semantic Web of Things

See also: Chapter 8

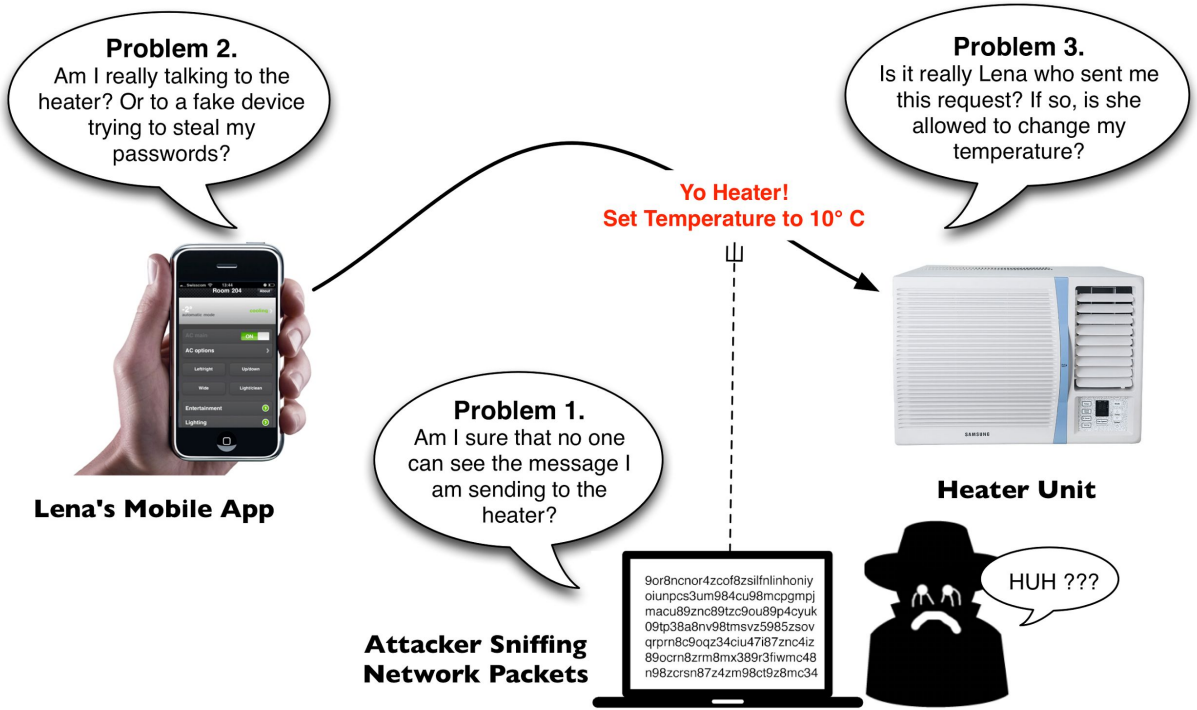
1. Experiment with the Web Thing Model in Action: Automatic UI generation
   - http://localhost:8484/model
   - `chapter10-mashups/UI/`
2. Change the mDNS address of your Pi
   - SSH to your Pi
   - `sudo nano /etc/hosts` - domguinards-pi
   - `sudo nano /etc/hostname`
   - `sudo reboot`
3. *Bonus:*
   - *download and run webofthings.js (see page 176)*
   - *implement your own mDNS server (see page 219 and `mdns` folder)*

WoT Architecture: Share & Secure

Chapter 9



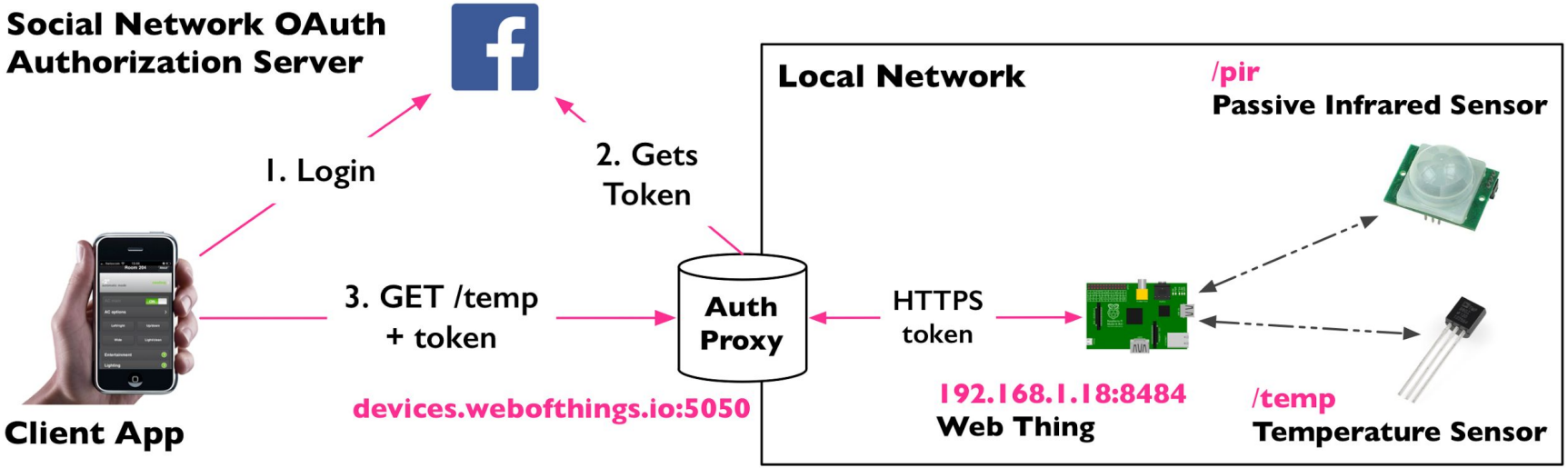
A. Securing Things (over simplified)



Source: Building the Web of Things: book.webofthings.io

- The most dangerous thing about Web Things is to bring them to the Web! (but also sort of the point :))
- Problem 1:
  - Web Encryption
- Problem 2:
  - TLS (SSL) certificates
- Problem 3:
  - API keys (oAuth)
  - Authorization header
  - Token query param

B. Sharing Things: Social Web of Things



http://webofthings.org/2010/02/02/sharing-in-a-web-of-things/

Lab 6: Sharing Things

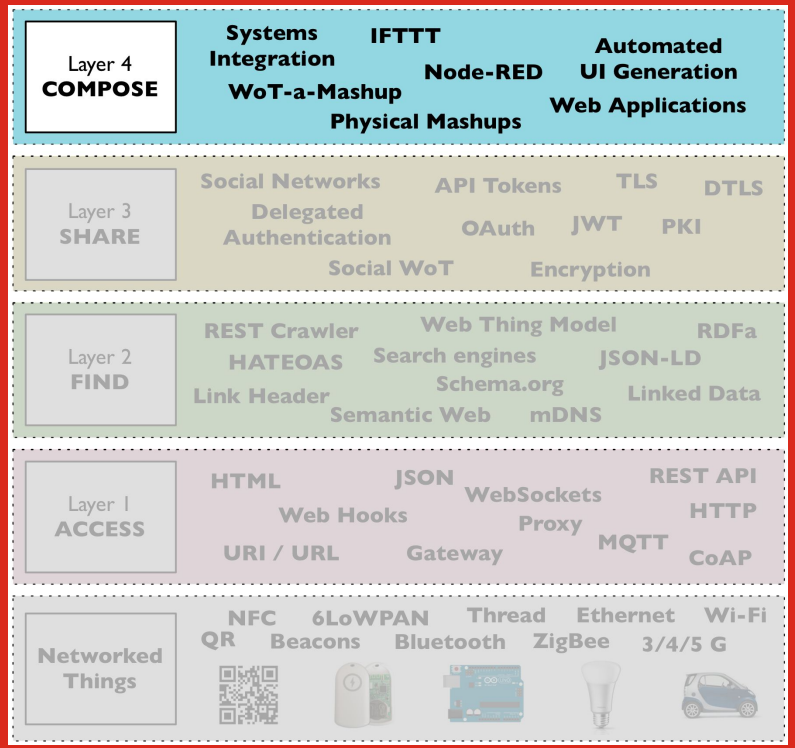
See also: Chapter 9

1. Demo of the Social WoT proxy
   - https://localhost:5050/login
2. Securing our server
   - Encryption:

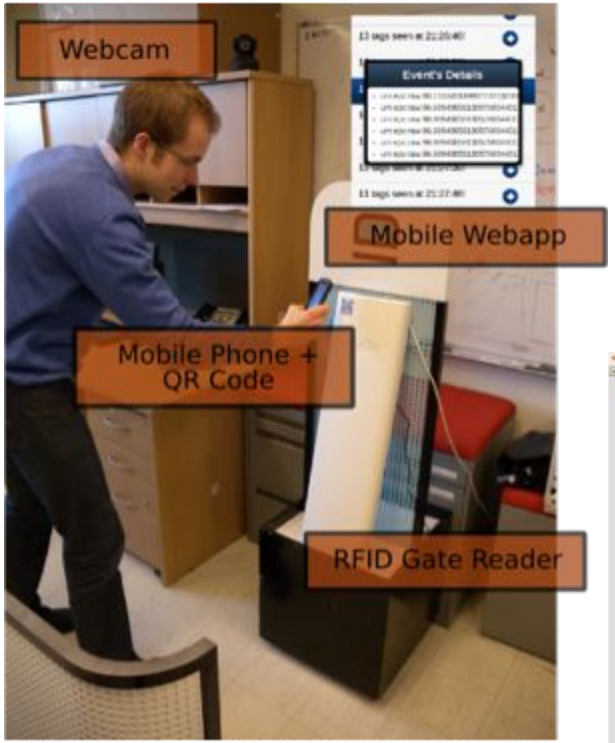
     ```
     openssl req -sha256 -newkey rsa:4096 -keyout privateKey.pem
     -out caCert.pem
     -days 1095 -x509
     ```
   - Try: https://localhost:8484/pi/
3. *Bonus: see how the API key support was added (page 260)*

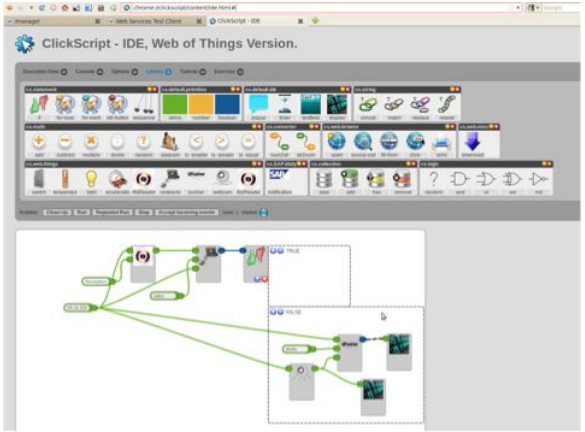
WoT Architecture: Compose

Chapter 10



Physical Mashups Born @MIT

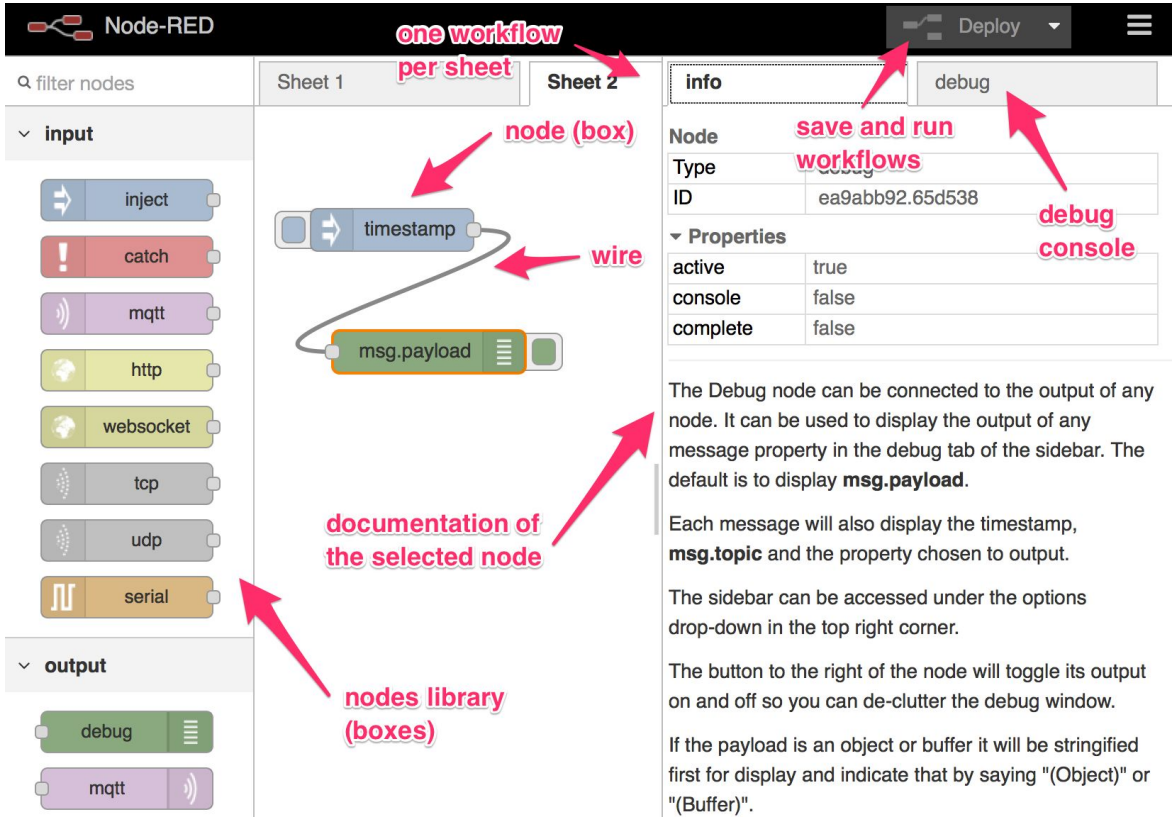




- Idea: prove to Walmart we could implement a security workflow in minutes, thanks to the Web
- Physical Mashups

Dominique Guinard, Christian Floerkemeier, Sanjay Sarma
Cloud Computing, REST and Mashups to Simplify RFID Application Development and Deployment.

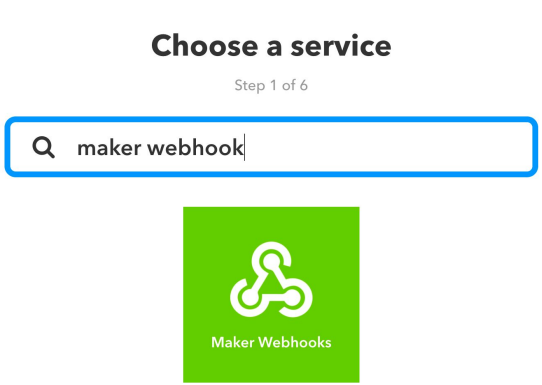
Node-RED

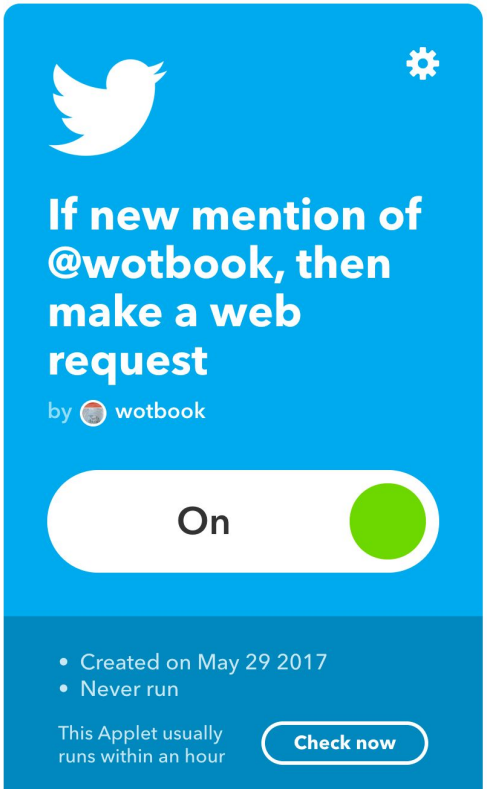


http://node-red.org

- Mashup tool for makers
- Box and wires
- Wire your prototypes
- Large community support
  - Nodes
  - E.g., https://flows.nodered.org/node/node-red-contrib-evrythng

IFTTT: Solid Mashups for the Masses





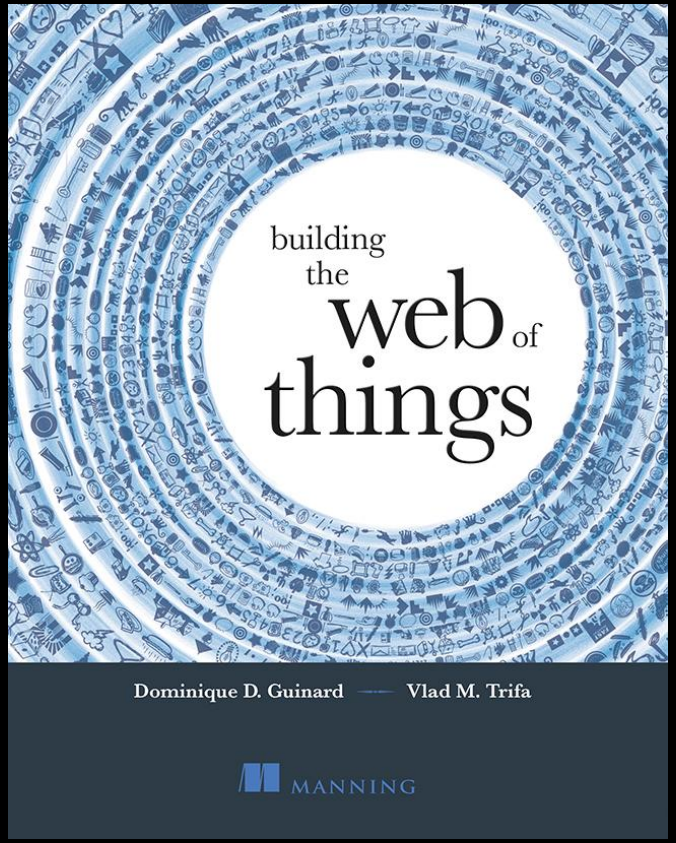
- If This Then That
- Wizard based mashups
- Mashups made accesible to anyone
- “Secret” Maker Hook chanel
  - Supports REST (HTTP Webhook)

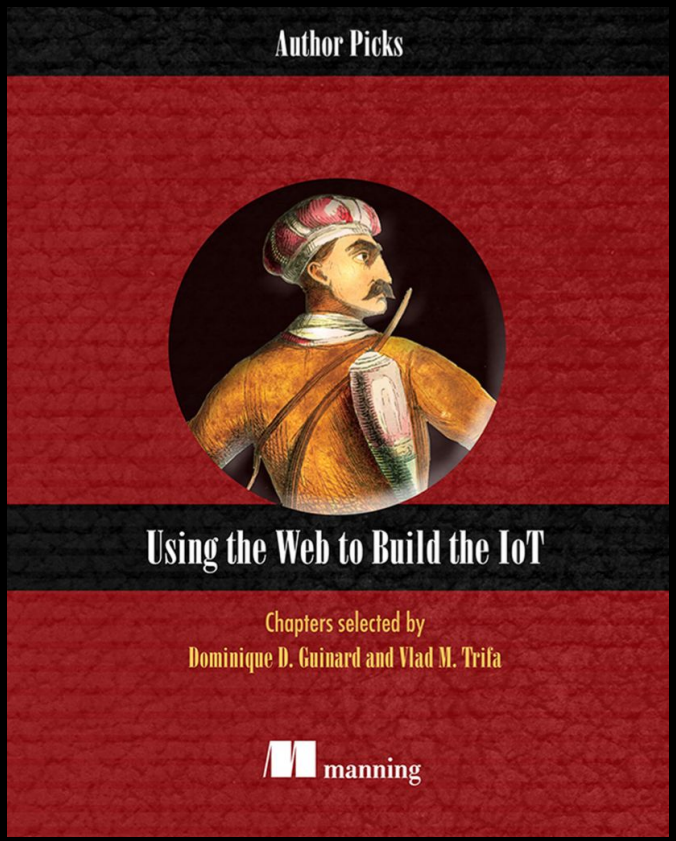
http://ifttt.com

Lab 7: Composing Things: Node-RED & IFTTT

See also: Chapter 10

1. Connect our PIR sensor to Node-RED (see page 293)
   - Launch the unsecure version (chapter 8)
   - `node-red`
   - `Connect to: ws://localhost:8484/properties/pir`
2. Create an IFTTT mashup that
   - Posts Tweets from your to the screen of the EVRYTHNG WoT Pi:
     - `http://devices.webofthings.io/pi/actuators/display/content`
     - `{"value":"message"}`
3. *Bonus:*
   - *Implement the full Node-RED mashup (see page 292)*
   - *Try the full JS mashup of chapter 2:* `ex-5-mashup.html`





39% off “Building the Web of Things”
with code “39guinard” on http://manning.com
See: http://book.webofthings.io

Contact:
@domguinard
@vladounet

Backups