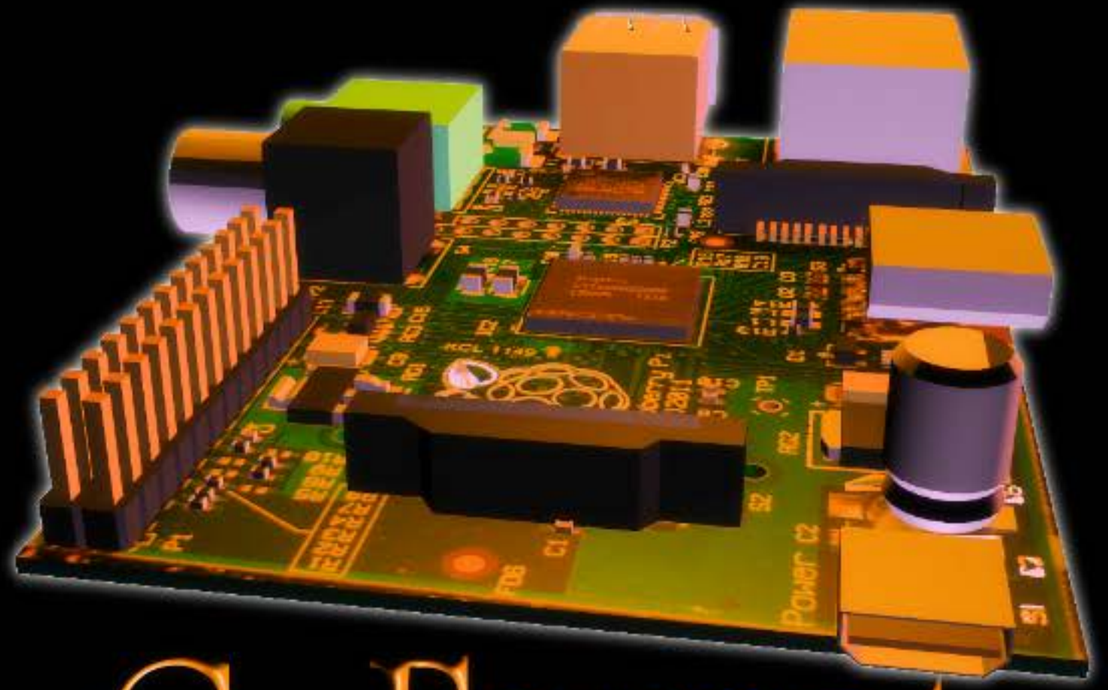


Mark Heckler
José Pereda

Jfokus 2014
IoT

G-Force!

Using Java to Build Self-Powering Global Sensor Nets for the Internet of Things

Introductions

José Pereda

- Dr. Industrial Engineer in structures
- University of Valladolid, Spain
- JavaFX, Embedded, Virtual UI
- Commercial, government & open source
- http://jperedadnr.blogspot.com.es
- @JPeredaDnr

Introductions

Mark Heckler

- Java Software Architect/Engineer
- Checkmark Solutions, Oracle USA
- Java EE, JavaFX, Embedded
- Government, commercial & open source
- http://www.thehecklers.org
- https://blogs.oracle.com/javajungle
- @MkHeck

Topics

- The Need for Remote Monitoring
- Requirements
- One Approach
  - Software
  - Hardware
  - Power
- **Building It!**
- Questions?

Topics

- The Need for Remote Monitoring
- Requirements
- One Approach
  - Software
  - Hardware
  - Power
- Building It!
- Questions?

The Need for Remote Monitoring

Systems

- Retail locations
- Industrial
- Energy production

The Need for Remote Monitoring

Security mechanisms

- Barriers
- Gates
- Panels (hardware or software)
- Aural/visual

The Need for Remote Monitoring

Other

- Seismic
- Atmospheric
- Anything that can be monitored or measured with a sensor!

Topics

- The Need for Remote Monitoring
- Requirements
- One Approach
  - Software
  - Hardware
  - Power
- Building It!
- Questions?

Requirements: Entire System

Ease of setup

- Should work with very few components
- Fast and easy to install/configure
- Minimal physical and logical footprint

Requirements: Entire System

Communication

- Mechanisms to support any required/desired connectivity
- Reliable

Requirements: Entire System

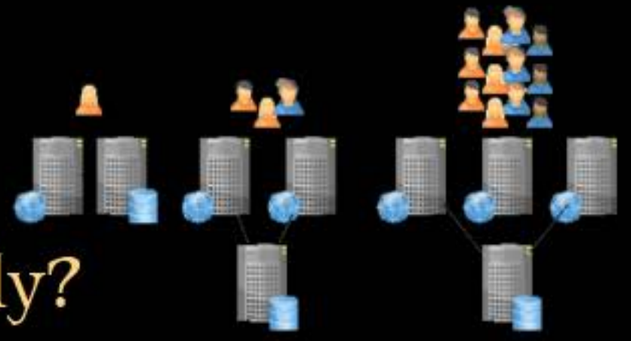
Performance

- System performance should be sufficient
- Various criteria (devices, comm, etc.)
- Tunable

Requirements: Entire System

Expandability/Scalability/Migratability

- In most situations, small wins
- Full functionality from basic configuration
- Add components (system nodes, client nodes) without disruption
- What if system must grow exponentially?



Requirements: Entire System

Standards-based

- The Internet of Things is a young area
- Smart approach is to build upon standards
- Extend when necessary

Requirements: Entire System

Deployability

- Remote accessibility/configurability

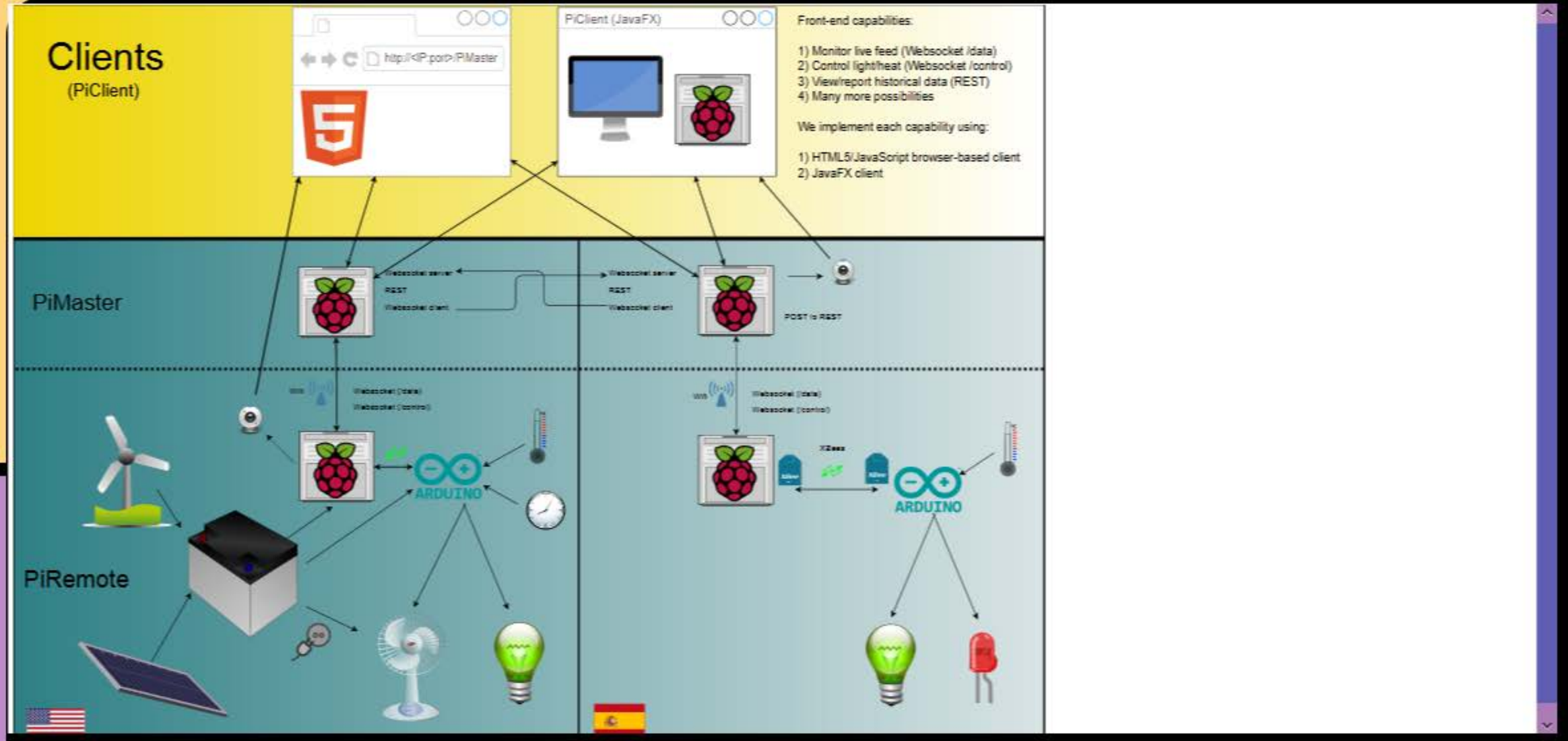
  (ssh, vnc, etc.)
- Compactness

Low power requirements

Topics

- The Need for Remote Monitoring
- Requirements
- One Approach
  - Software
  - Hardware
  - Power
- Building It!
- Questions?

One Approach: Overview



One Approach: Software

Java forms the heart of the system

- Servers
- Remote node “concentrator”
- Rich clients

One Approach: Software

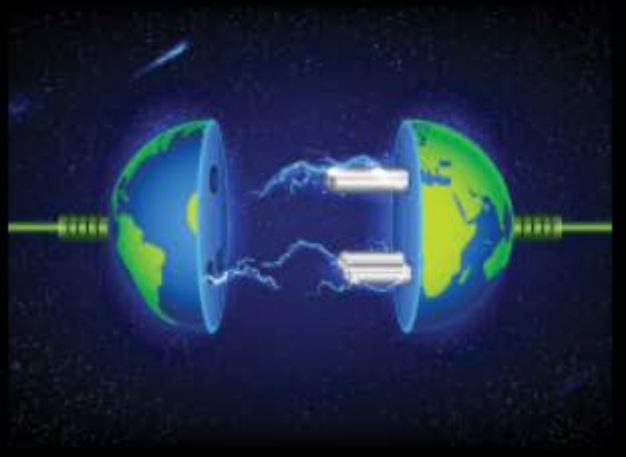
Arduino handles sensor duties

- Standards-based
- Open source
- Rich ecosystem

One Approach: Software

Web standards fill out the rest

- REST! (Web, Java)
- WebSocket (Web, Java)
- HTML5/JavaScript for everywhere else



One Approach: Hardware

Raspberry Pi

- Raspbian ships with Java SE Embedded
- Readily supports Java EE 7 (GlassFish 4)
- Compact
- Inexpensive
- Widely available

One Approach: Hardware

Arduino

- Shields, shields, shields!
- Inexpensive & widely available
- Processing language similar to Java
- Open source **hardware**
- Barely sips power, robust/durable
- 7-12V rated input voltage; can take 6-20V!
- Provides both 3.3V & 5V outputs

One Approach: Hardware

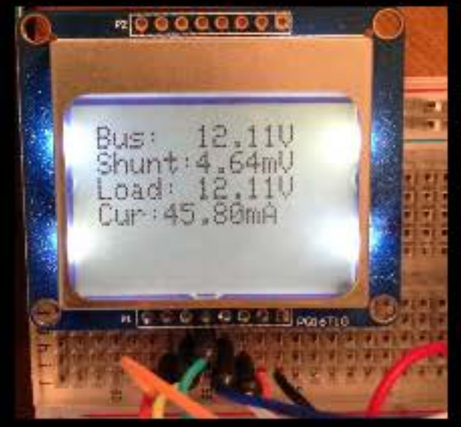
Communication hardware

- WiFi adapters
- Serial connections
- Ethernet (for Pi servers)
- XBee

One Approach: Power

Incorporating into Renewable Energy system

- Monitors RE production/consumption
- Powered by system it monitors



Low power consumption

- Good for entire system
- Critical for remote node cluster

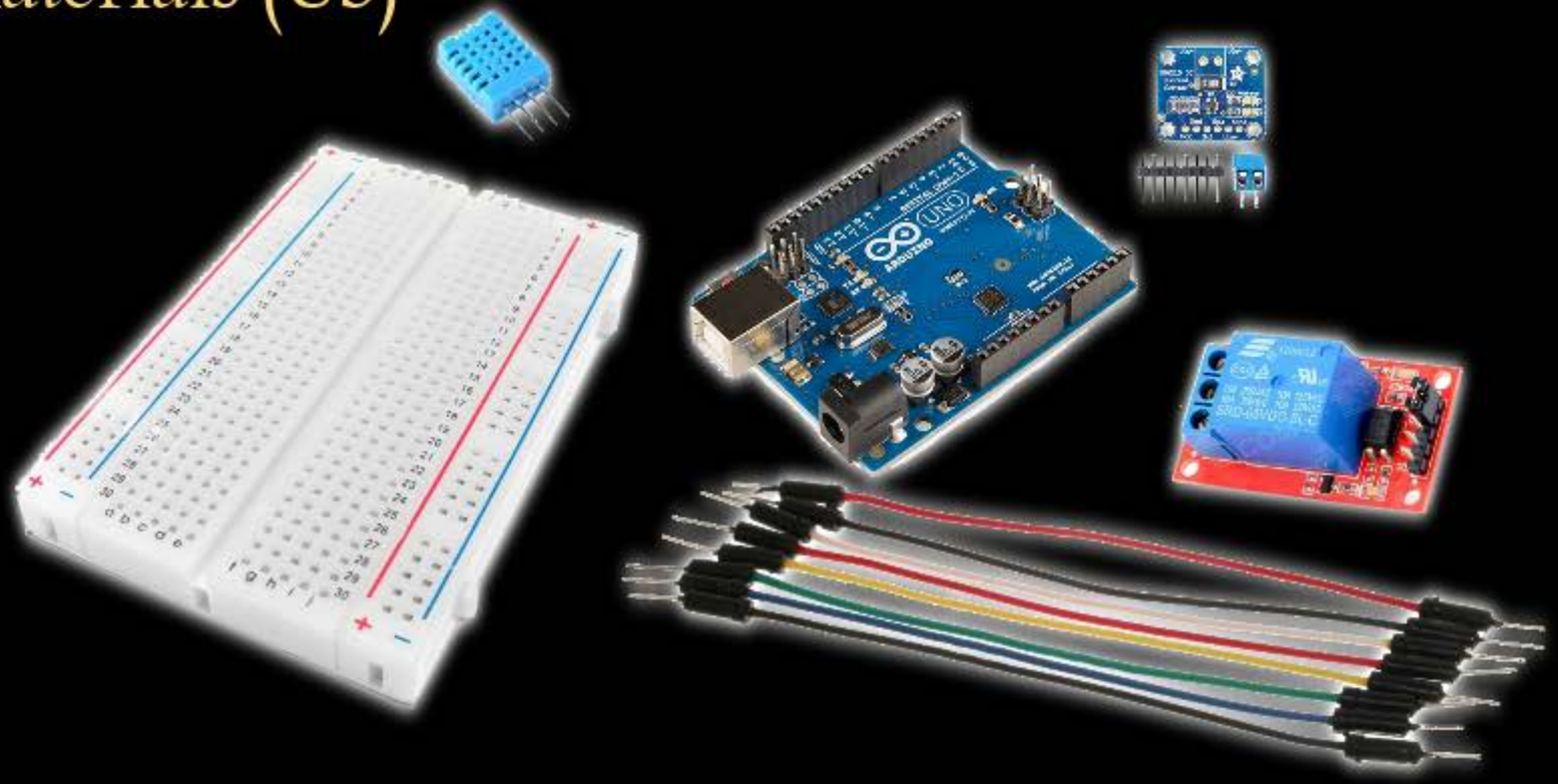
Topics

- The Need for Remote Monitoring
- Requirements
- One Approach
  - Software
  - Hardware
  - Power
- **Building It!**
- Questions?

Building It! Backend

Start with the sensors

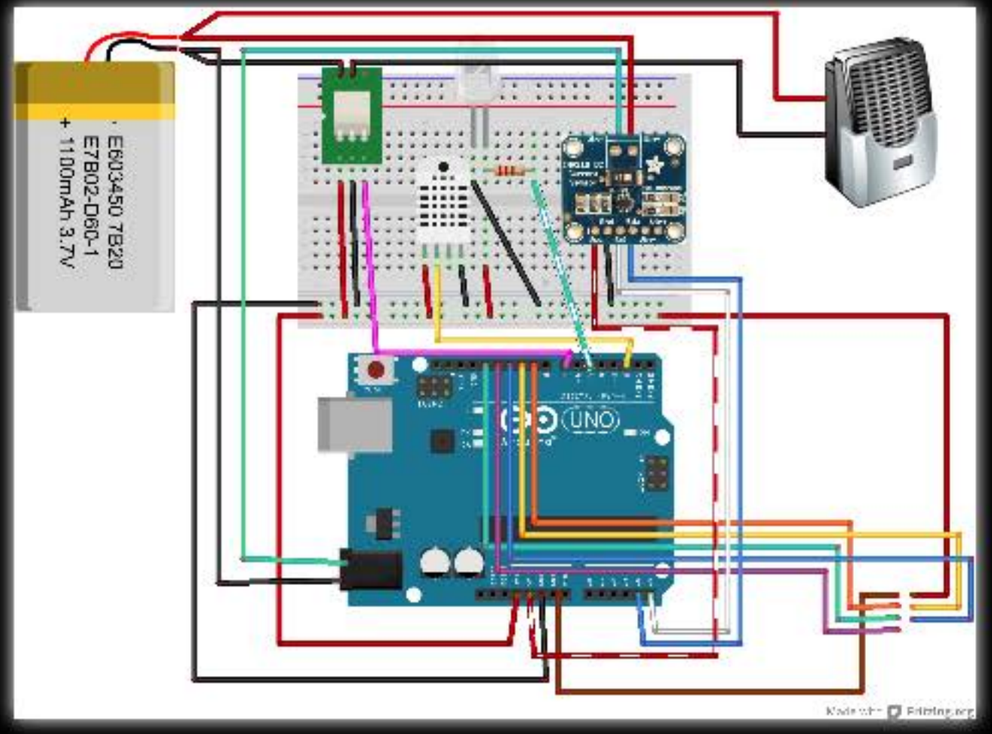
- Bill of Materials (US)



Building It! Backend

Start with the sensors

- Schematics – Fritzing (US)



Building It! Backend

Arduino IDE

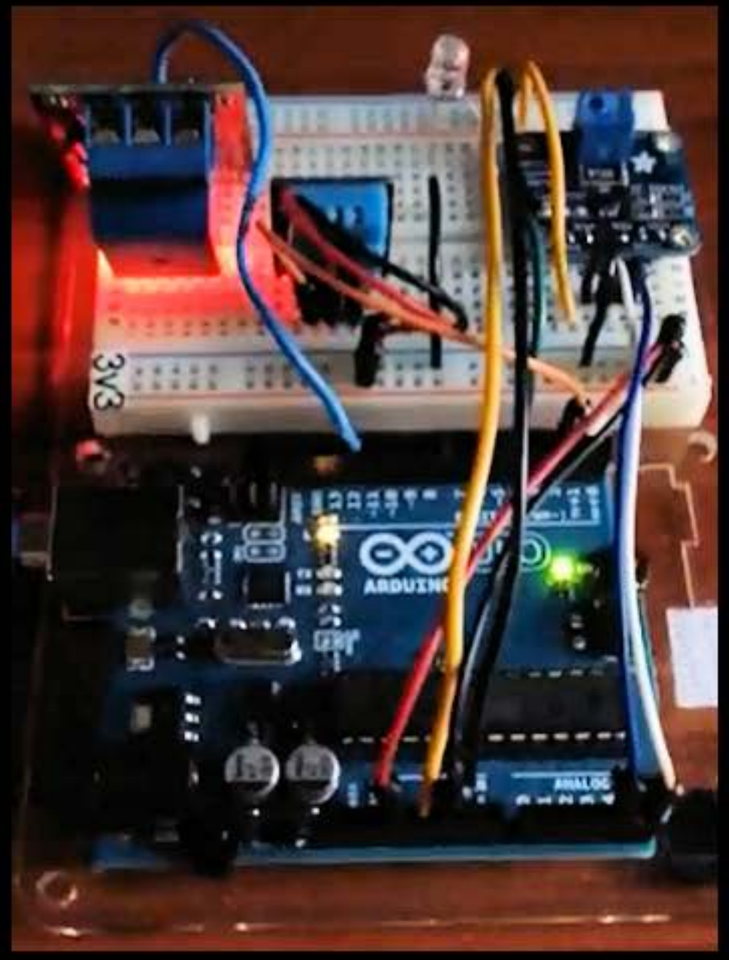
RemoteControl

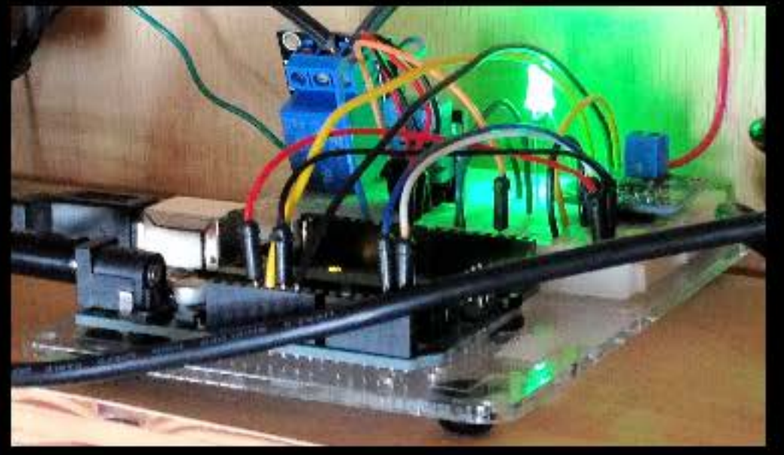
```
12
13  #include <dht11.h>
14  #include <Wire.h>
15  #include <Adafruit_INA219.h>
16
17  // Hardware configuration
18
19  // Set up sensors
20  dht11 DHT11;
21  Adafruit_INA219 ina219;
22
23  // And now the variables for capturing sensor values
24  int chk;
25  float busVoltage;
26  float shuntVoltage;
27  float current_mA;
28  float loadVoltage;
29  String msg;
30
31  // State & other variables
32  int inByte;
33  int lightPin = 5;          // LED pin
34  int powerPin = 12;         // Assign "for real" when relay is connected
35  boolean isAutonomous = true;
36  boolean isLightOn = false;
37  boolean isPowerOn = false;
```

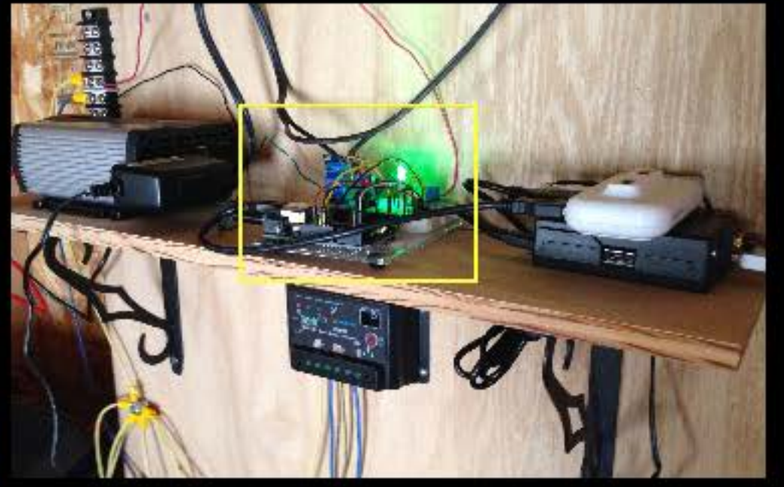
Building It! Backend

Deploy the sensors

- Shed (US)



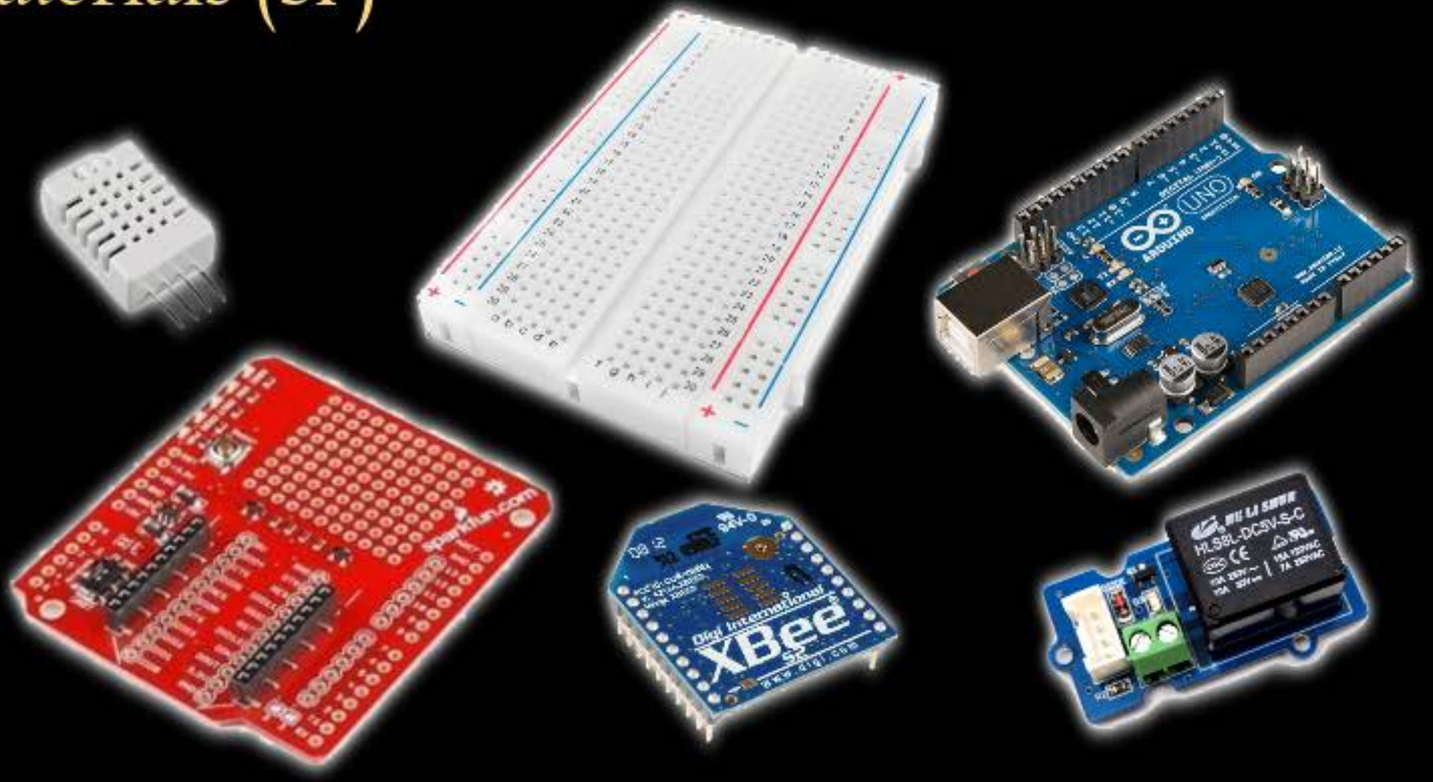




Building It! Backend

Start with the sensors

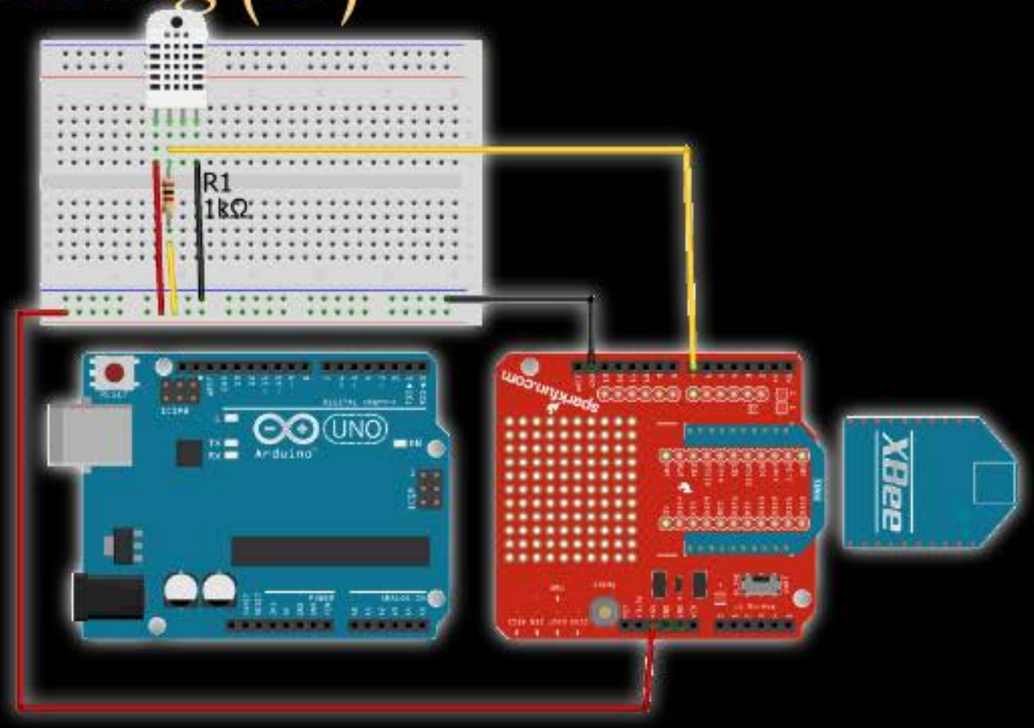
- Bill of Materials (SP)



Building It! Backend

Start with the sensors

- Schematics – Fritzing (SP)



Building It! Backend

Arduino IDE

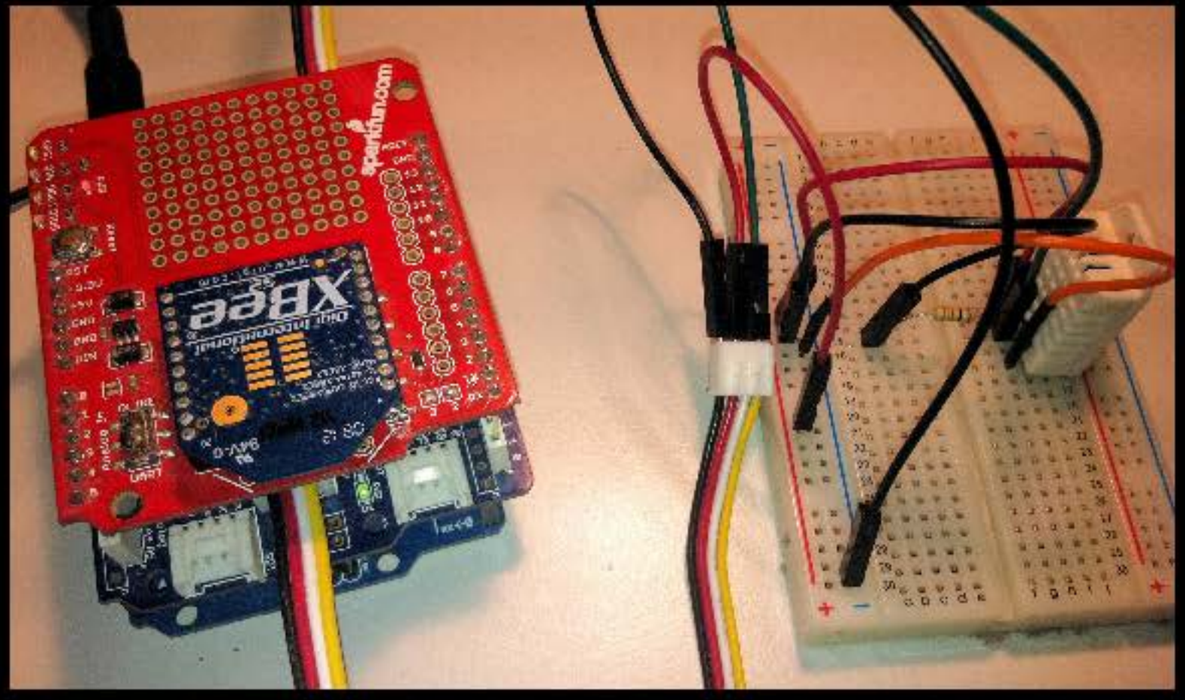
SerialDHTControl

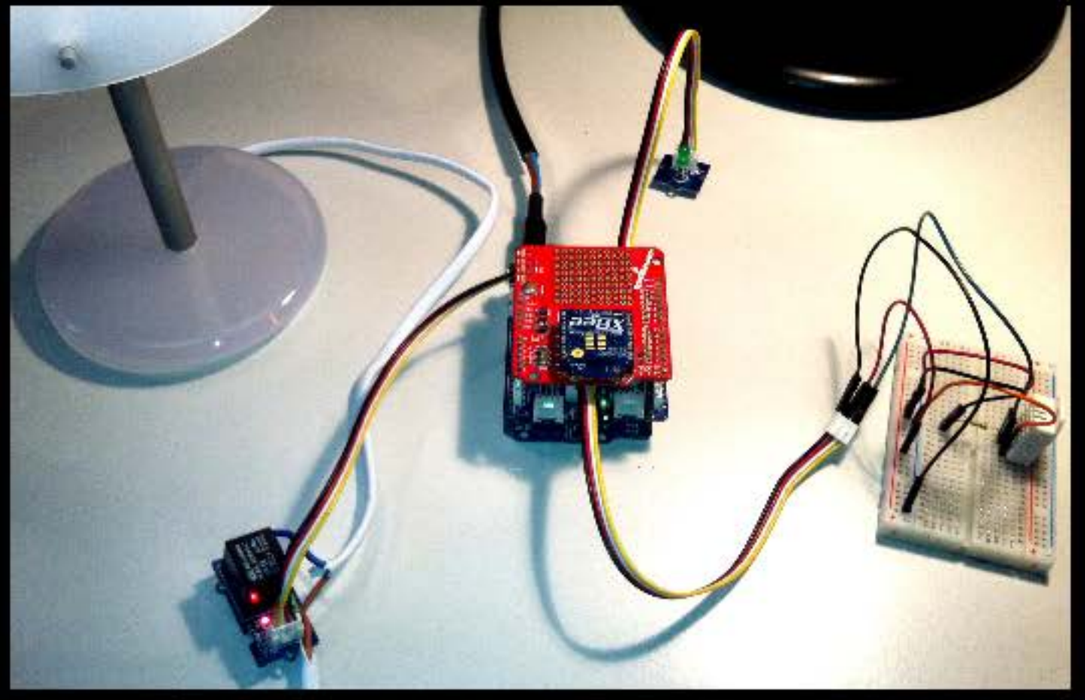
```
12
13  #include <DHT22.h>
14  #include <stdio.h>
15
16  // Data wire is plugged into port 7 on the Arduino
17  // Connect a 4.7K resistor between VCC and the data pin (strong pullup)
18  #define DHT22_PIN 7
19
20  // Setup a DHT22 instance
21  DHT22 myDHT22(DHT22_PIN);
22
23  int cont=0;
24  int lightPin = 5; // LED pin
25  int powerPin = 13;
26  boolean isAutonomous = true;
27  boolean isLightOn = false;
28  boolean isPowerOn = false;
29  int powerOnSeconds = 0;
30
31  void setup(void)
32  {
33    // start serial port
34    Serial.begin(9600);
35    Serial.println("Initializing...");
36    pinMode(lightPin, OUTPUT);
37    pinMode(powerPin, OUTPUT);
```

Building It! Backend

Deploy the sensors

- Home (SP)

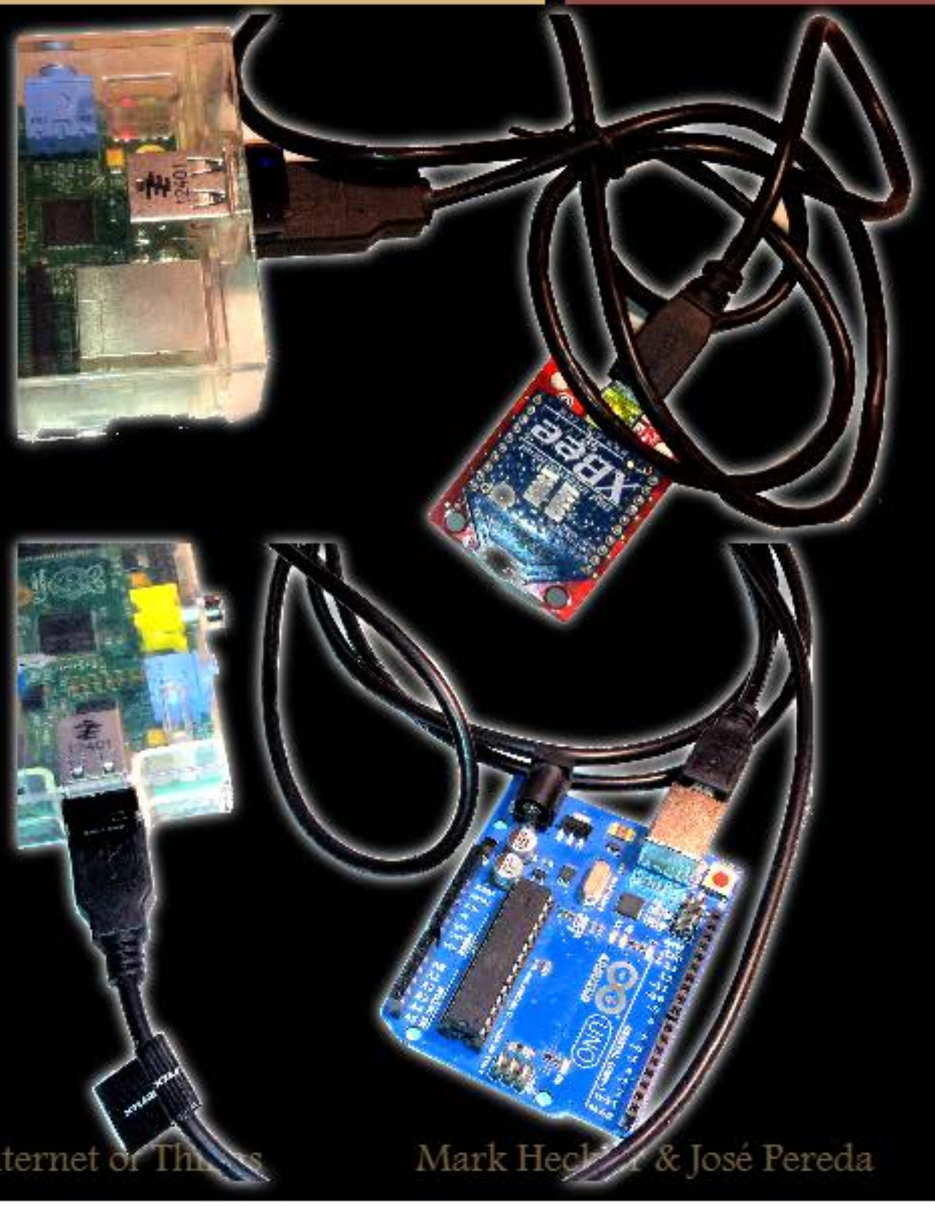




Building It! Gateway

Gateway: Remote "concentrator"

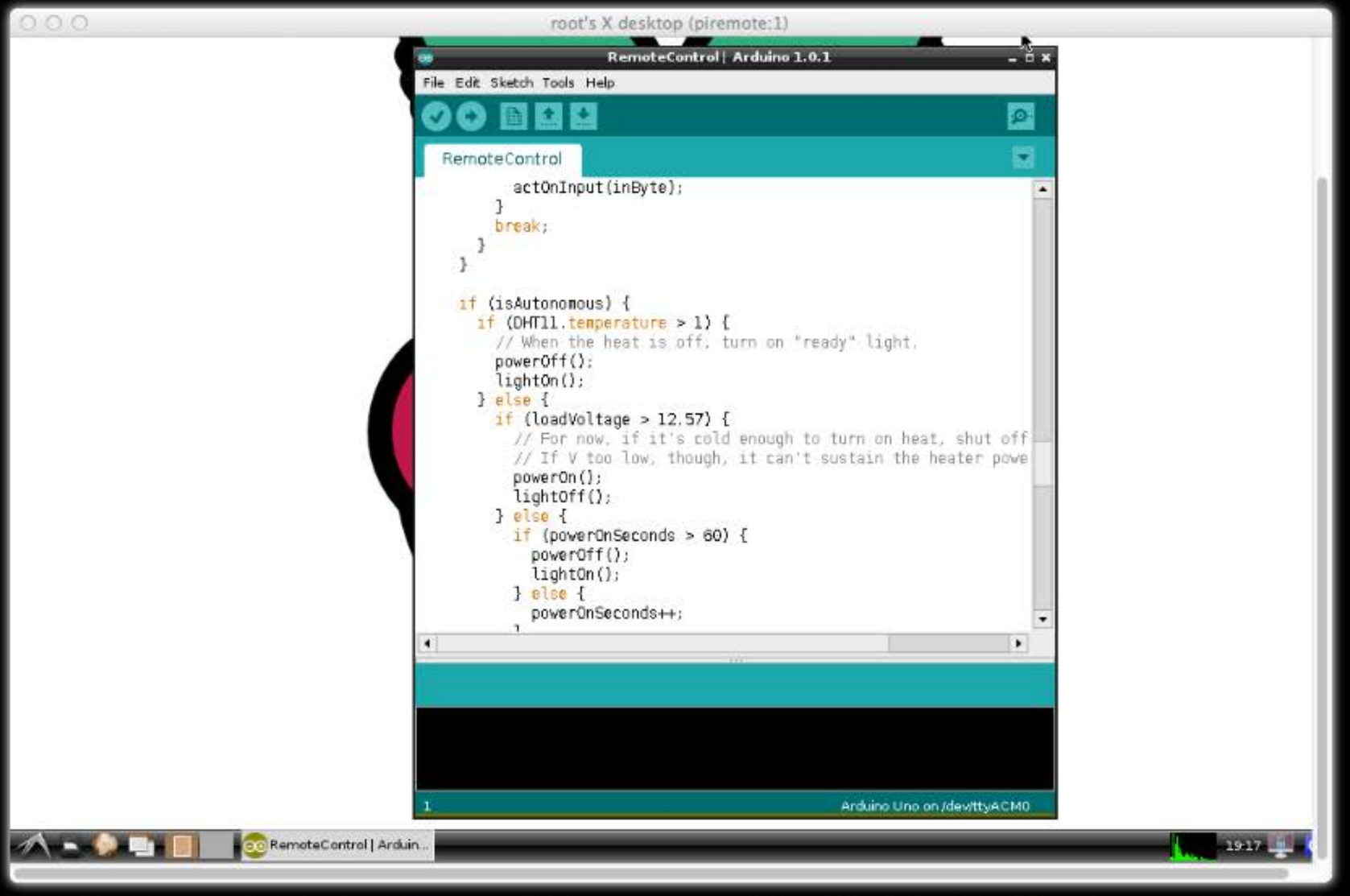
- Raspberry Pi
- Java SE
- Serial: RXTX library
- Arduino: USB/ZigBee
- Local Log
- Server: WebSockets



Building It! Gateway

Remote Control

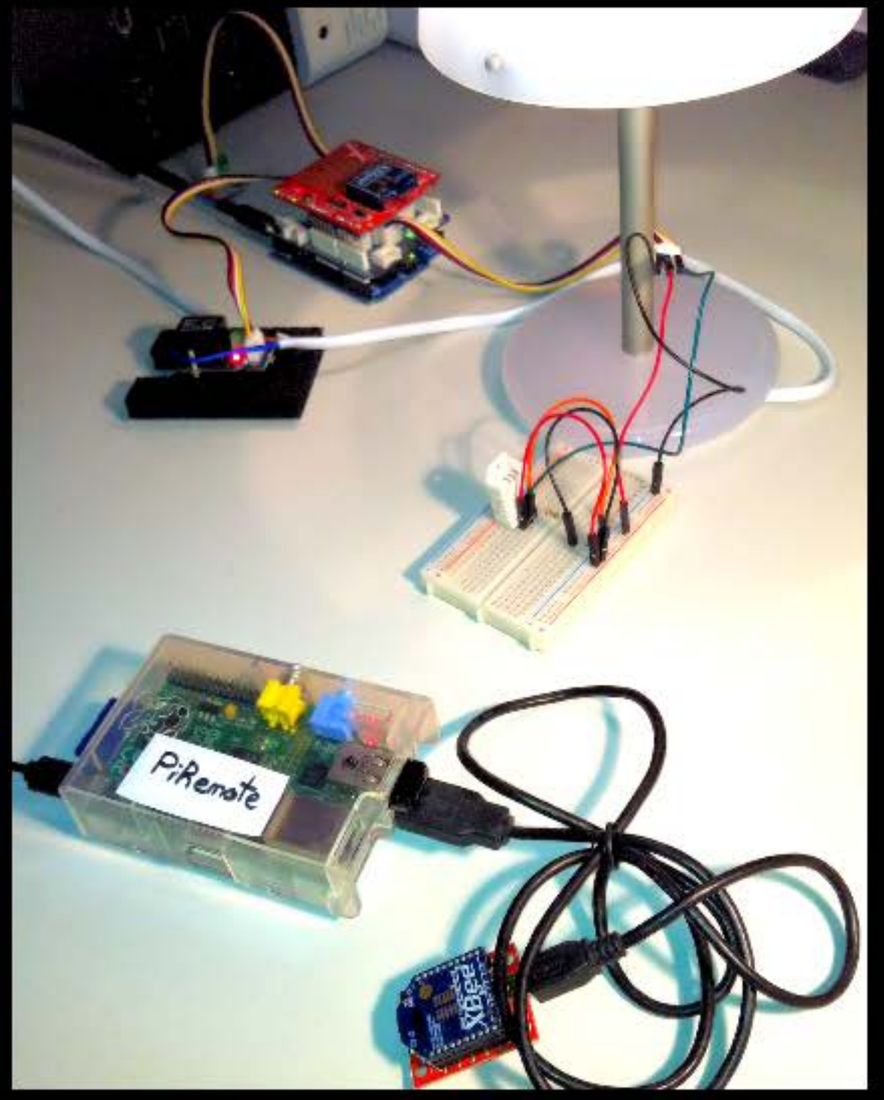
- Arduino



Building It! Gateway

Deploy the Gateway

- Shed (US) & Home (SP)



Building It! Server

Application Server

- Raspberry Pi
- GlassFish 4.0
  - Web Profile
  - WebSockets
  - RESTful web services
- JavaDB
- Optional logging

Building It! Server

Crossed Link between servers

- US server acts as client of SP server

{"id":0,"sensor":1,"time":"Tue Feb 04 01:39:30 CET 2014","temp":17.8,"hum":41.4,"volts":1210.0,"amps":-1.0}
{"id":0,"sensor":1,"time":"Tue Feb 04 01:39:36 CET 2014","temp":17.8,"hum":41.4,"volts":1210.0,"amps":-1.0}
{"id":0,"sensor":1,"time":"Tue Feb 04 01:39:42 CET 2014","temp":17.8,"hum":41.5,"volts":1210.0,"amps":-1.0}

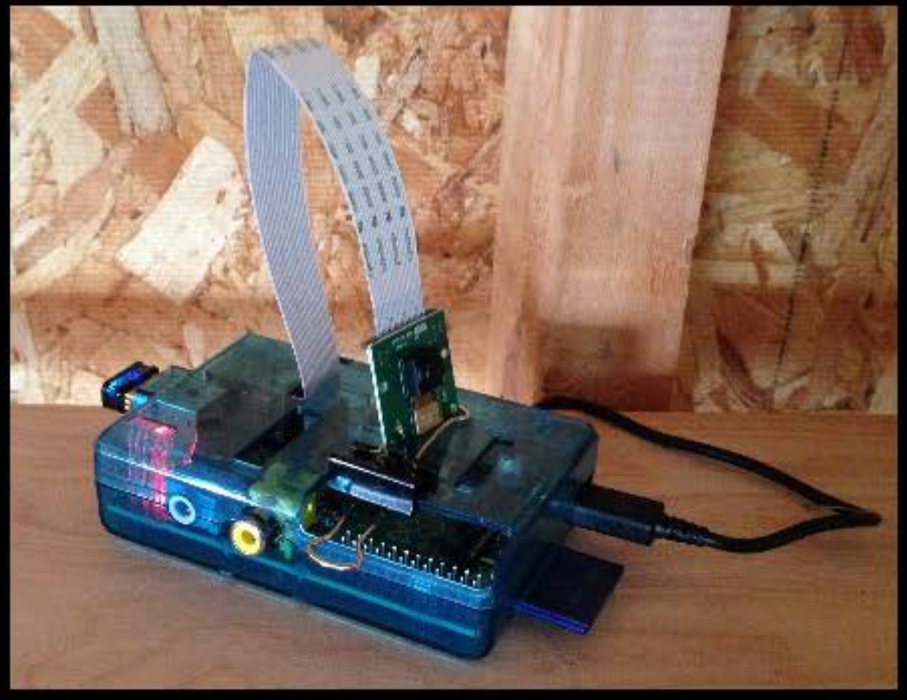
- SP server acts as client of US server

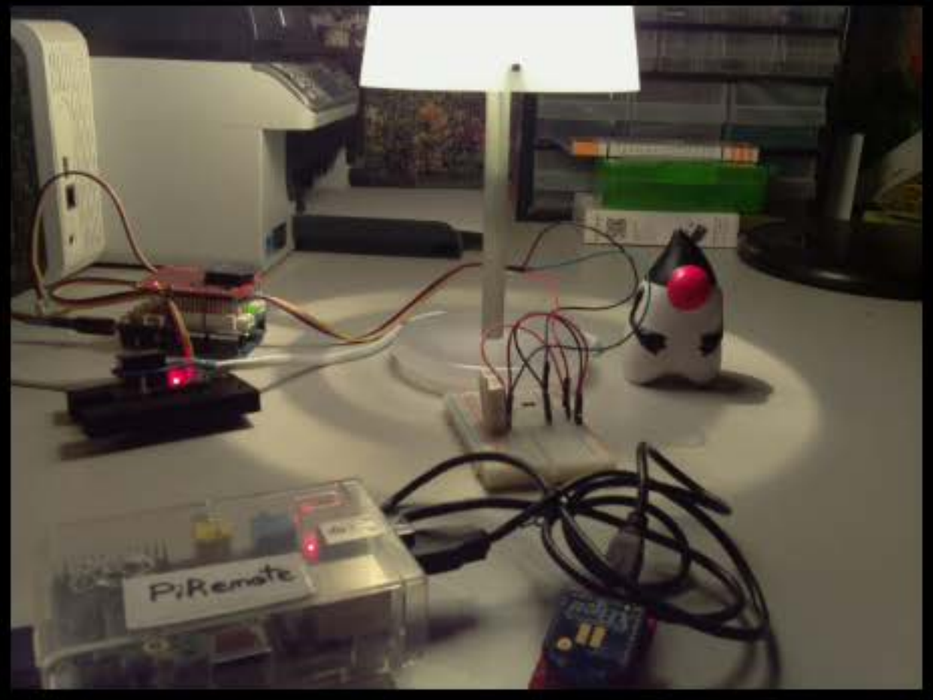
{"id":0,"sensor":0,"time":"Mon Feb 03 18:39:31 CST 2014","temp":3.0,"hum":45.0,"volts":13.13,"amps":0.066}
{"id":0,"sensor":0,"time":"Mon Feb 03 18:39:46 CST 2014","temp":4.0,"hum":45.0,"volts":13.126,"amps":0.065}

Building It! Server

Using a camera to keep an eye on things

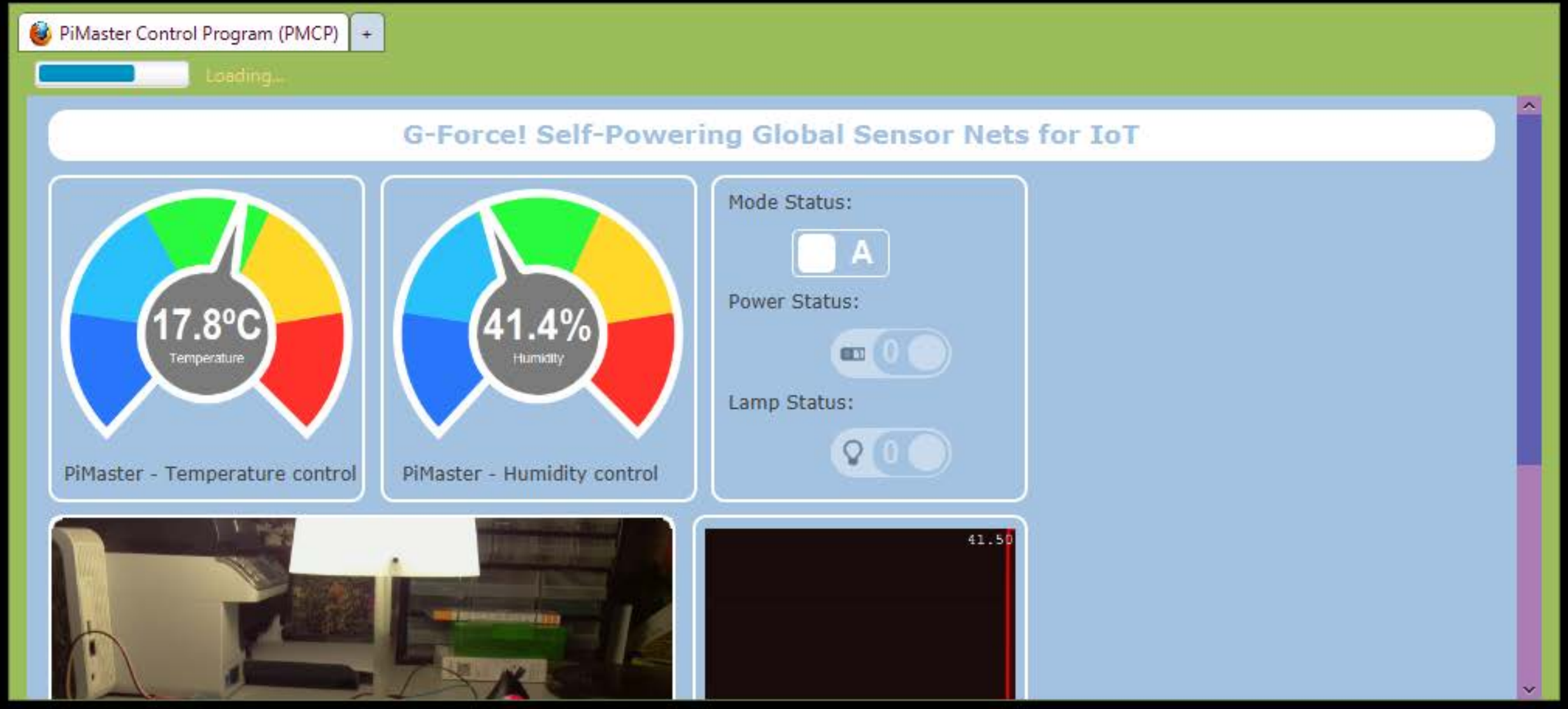
- MJPG streamer (raspistill) 1 fps





Building It! Client

HTML5/Javascript



Building It! Client

JavaFX Client (Desktop-Raspberry Pi)

Keep it Going!

José

- http://jperedadnr.blogspot.com.es
- @JPeredaDnr

Mark

- http://www.thehecklers.org
- https://blogs.oracle.com/javajungle
- @MkHeck

Questions?