End-to-end IoT solutions
with Java and the
Eclipse IoT stack
IoT is Big
THE NUMBER OF IOT DEVELOPERS 2014-2020

Source: VisionMobile estimates, 2014

Report: IoT: Breaking Free From Internet And Things | vmob.me/IoT
©VisionMobile | June 2014 | Licensed under CC BY ND
End-to-end IoT?

Actuators/Sensors

+ Gateway

[ Cloud ]

+ User front-end
1. Sensors/Actuators

→ **Sense** the physical environment

→ **Act** on it
1. Sensors/Actuators

➔ Manipulate \texttt{sysfs} directly
➔ Use \texttt{Pi4J} to have full support of GPIO/I2C/SPI
➔ Device I/O API with Java or Java ME
Sensors/Actuators

→ Pi4J – http://pi4j.com

✓ Complete access to GPIOs/I2C/SPI
✓ Very mature codebase, based on WiringPi
✓ Support for popular shields (PiFace, Gertboard, …)
✓ Lots of code samples
Pi4J in action

GpioController gpio = GpioFactory.getInstance();
GpioPinDigitalOutput pin = gpio.provisionDigitalOutputPin(
    RaspiPin.GPIO_01, "MyLED", PinState.HIGH);

Thread.sleep(5000);

pin.low();

Thread.sleep(5000);

pin.toggle();

gpio.shutdown();
Gateway
Gateway
Gateway

Connect
sensors to the world

Manage the hardware
and software running at the edge
Connect?

➔ CoAP

✓ « HTTP over UDP »
✓ Expose your device as a resource to the Internet of Things

➔ MQTT

✓ Publish/Subscribe model
✓ More room for local processing
CoAP: The web-of-things

/walk
/hand/left/raise
/eye/picture

/mtbf

/bat-level

/CO2
/noise
/lights/on

/engine/status
/position
/fuel
Eclipse Californium

➔ Focus on scalability and usability
➔ To be used in IoT cloud servers or M2M/IoT devices running Java
➔ Includes **DTLS** implementation (Scandium), HTTP/CoAP bridge, Plugtests, ...
1. Implement custom resources (extend `CoapResource`)
2. Add resources to the CoAP server
3. Start the server
import static org.eclipse.californium.core.coap.CoAP.ResponseCode.*; // shortcuts

public class MyResource extends CoapResource {

    @Override
    public void handleGET(CoapExchange exchange) {
        exchange.respond("hello world"); // reply with 2.05 payload (text/plain)
    }

    @Override
    public void handlePOST(CoapExchange exchange) {
        exchange.accept(); // make it a separate response
        if (exchange.getRequestOptions() ...) {
            // do something specific to the request options
        }
        exchange.respond(CREATED); // reply with response code only (shortcut)
    }
}
MQTT: Publish & Subscribe

- **Pub** KETTLE232/temp
  - Payload: 21°C

- **Sub** KETTLE232/#

  - **Payload:** 21°C
Eclipse Paho

➔ Open-source MQTT clients
➔ Pick your language!

✓ Java
✓ JavaScript
✓ C/C++, Objective C
✓ Go, Lua, Python, .Net, WinRT, ...
MqttClient c = new MqttClient("tcp://m2m.eclipse.org:1883", 
    MqttClient.generateClientId());

mqttClient.setCallback(new MqttCallback() {
    @Override
    public void messageArrived(String topic, MqttMessage message) 
    throws Exception {
        // process received message
        // ...
    }
});

mqttClient.connect();
mqttClient.subscribe("mygateway/#");
MQTT brokers

➔ Eclipse Mosquitto
  ✓ C implementation
  ✓ Scalable (1000 clients == 3MB RAM)

➔ Eclipse Moquette
  ✓ Java implementation
  ✓ Based on Netty and LMAX disruptor
Manage?

➔ **Gateway itself**
  ✓ wireless modem, firewall, ...

➔ **Applications**
  ✓ Install/Uninstall software packages
  ✓ Start/Stop applications

➔ **Sensors**
  ✓ H/W abstraction layer
Installing Kura

```bash
cd ~
sudo apt-get update
wget https://s3.amazonaws.com/kura_downloads/raspbian/release/1.0.0/ 
  kura_1.0.0_raspberry-pi_armv6.deb
sudo dpkg -i kura_1.0.0_raspberry-pi_armv6.deb
sudo apt-get install -f
sudo reboot
```
First steps with Kura

➔ Network management
  ✓ Cellular Modem, WiFi
  ✓ Firewall
  ✓ NAT

➔ OSGi and system administration

➔ IoT server communication settings
Kura API

➔ OSGi services that you can re-use in your own components

✓ ClockService
✓ DataService, CloudService
✓ CryptoService (AES, base64, SHA-1)
✓ PositionService (geolocation)
✓ ... and many others
Demo time!
Practical example: Greenhouse

➔ A SensorService implementation leveraging Pi4J for GPIO/I2C

➔ A configurable GreenhousePublisher component that:
  ✓ consumes a SensorService (GPIO manipulation)
  ✓ consumes a DataService (MQTT communication)
  ✓ exposes a SensorChangedListener (be notified of external GPIO updates)
  ✓ exposes configuration metadata (Web UI config)
Talk is cheap...
Show me the code!
And how about CoAP?

➔ **Californium** can easily be embedded in a Kura bundle

➔ Another OSGi component can:

   ✓ consume the SensorService
   ✓ expose CoAP resources

⚠ You need to open port 5683 in Kura!
End-user interaction

➔ JavaFX Charts

➔ Eclipse BIRT

➔ Smartphone app (e.g. Android)
  ✓  https://www.eclipse.org/paho/clients/android

➔ MQTT + WebSockets = ♡
  ✓  https://www.eclipse.org/paho/clients/js
If you had to remember only 3 things...

#1

Kura is awesome!
Go download it now!

http://eclipse.org/kura
If you had to remember only 3 things...

#2 Build your own greenhouse & follow the tutorial

http://iot.eclipse.org/java/tutorial
If you had to remember only 3 things...

#3 Eclipse Open IoT Stack for Java is much more than Kura

http://iot.eclipse.org/java
Get Involved!

WE WANT YOU!!
Thank you! Questions?

benjamin@eclipse.org
@kartben

http://iot.eclipse.org