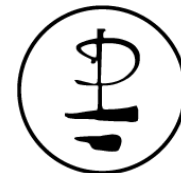



Internet Of Things:
Eclipse SmartHome with OpenHAB

Giuseppe Salinaro

Bio

- Student @ Department of Computer Science - University of Bari "Aldo Moro"
- Linux user since 2006
- Member of the openSUSE 11.0 (2008) translation team
- **Il Software Libero nella Pubblica Amministrazione (2010)**





Rise of the machines: what the Internet of Things means for your business

A report recently highlighting 3 key facts about our headlong rush into the Internet of Things.

- 1)The market for IoT will grow 347% in the next four years.
- 2)that its growth will eclipse the human-use internet before the end of the decade.
- 3)that the true winners of IoT may not be your typical tech companies.



Rise of the machines

Restrooms, kitchens, televisions, mobile devices, cars, gasoline pumps, packages, refrigerators, vending machines, SCADA systems: all will soon be vital communication tools in helping to make our lives easier/healthier/more productive.

The Internet of Things is not just about gathering data; it's also about the analysis and use of that data.

Eclipse IoT



Eclipse IoT is an ecosystem of companies and individuals that are working together to establish an Internet of Things based on open technologies.

Eclipse IoT

CANONICAL

GENERATIVE
SOFTWARE



openHAB
FOUNDATION



bitreactive





Eclipse IoT

The community is composed of over 200 active contributors working on 24 different projects. These projects are made up of over 2 million lines of code and have been downloaded over 500,000 times.

The Eclipse IoT Working Group includes 30 member companies that collaborate to provide software building blocks in the form of open source implementations of the standards, services, and frameworks that enable an Open Internet of Things.



Eclipse IoT

A vast majority of today's IoT solutions are designed in an ad hoc manner.

Depending on the business domain and on the targeted platform (OS, H/W capabilities, ...) very different and often incompatible architectures are implemented.

Eclipse IoT provides building blocks that sit on top of open standards and protocols and provide additional services and frameworks for device management, wired/wireless communication, vertical solutions like home automation, ...



Eclipse IoT

Building interoperable IoT solutions is a real challenge. From sensors and actuators on the field to backend systems, there are many aspects of an end-to-end solutions where it is important to rely on standards:

- Protocols used to implement the device-to-device or device-to-server communications,
- Device Management protocols to allow remote control of IoT devices and gateways,
- Gateways and Servers interfaces.

While Open Standards are key, we believe that it is also important to make available open-source implementations of such standards, to encourage adoption of such standards both by IoT developers and the IoT industry at large.

Eclipse IoT

The group provides open source implementations of IoT protocols -- CoAP, ETSI SmartM2M, MQTT and LwM2M -- and backs extensible services and frameworks to foster the development of IoT applications with open APIs. The four projects include:

- Eclipse Kura , a framework for building IoT gateways.
- Eclipse Paho, for client-based implementations of the MQTT and MQTT-SN messaging protocols.
- Eclipse SmartHome, a framework for creating smart home solutions focusing on heterogeneous environments, designed for embedded devices like Raspberry Pi, BeagleBone Black and Intel Edison.
- Eclipse OM2M, which implements the oneM2M standard, providing horizontal IoT services to foster the development of IoT solutions that are independent from underlying networks.

Eclipse SmartHome



The SmartHome project is a framework that allows building smart home solutions that have a strong focus on heterogeneous environments, i.e. solutions that deal with the integration of different protocols or standards. Its purpose is to provide a uniform access to devices and information and to facilitate different kinds of interactions with them. This framework consists of a set of OSGi bundles that can be deployed on an OSGi runtime and which defines OSGi services as extension points.



Eclipse SmartHome

I could go on, and with the Internet of Things we'll see many more products rising. But how can it all be put together? With such fragmentation, how do I know that the system I choose is the one that will gain mass market adoption? This is exactly what the Eclipse Smart Home project aims to resolve.

OpenHAB



Kai Kreuzer, who started openHAB in 2010, is leading up the project, and has contributed the core of openHAB to create the Eclipse SmartHome framework.

openHAB itself already has a number of devices and technologies that can be integrated, giving you just a sample of what is possible.

openHAB 2.0 will be built on top of the SmartHome framework.



OpenHAB

- Insteon and older X-10 devices.
- Nest thermostat, you can get information from the Nest and use it to configure roller blinds, attic fans, or other HVAC devices.
- You can even add a contact to a window (which doubles as a security contact) and when the window is open, flip the Nest into "Away" mode.
- Philips HUE, Wemo, Plex, Z-Wave devices
- it's all modular. You set up the bindings and application integrations you want, and then you can have events on one thing trigger others.

OpenHAB 2 vs OpenHAB1

- the openHAB 1.x branch - the focus being on the bindings and other add-ons. For the core runtime and the designer, only critical bugs will be fixed.
- openHAB 2 has a different focus: User comfort. While in openHAB 1.x you need to configure everything in text files and figure out the right syntax and possibilities of a certain binding in the wiki, openHAB 2 allows bindings and other add-ons to self-describe their configuration, so that it is possible to offer user interfaces for system setup and configuration. Additionally, auto-discovery (e.g. through UPnP, AllJoyn, etc.) is offered, so that new devices can be added by a simple click of a button.



OpenHAB 2 vs OpenHAB 1

- A second major design goal of openHAB 2.x is the optimization for embedded platforms.
- The core runtime and its APIs will change fundamentally.
- In consequence, the source repository of openHAB 2 will start with no bindings at all and we encourage the developers in our community to migrate their existing 1.x bindings to the 2.x concepts, once these are fully in place.
- For the time being, there is a "1.x compatibility bundle" that allows to use openHAB 1.x add-ons with the openHAB 2 runtime - this will not run for all of them out of the box, but for the majority of them.



OpenHAB 2

- The central configuration file "openhab.cfg" is gone. Instead, you can now have a separate file for every add-on. This clearly improves the overview over your configuration parameters as your configuration file won't be cluttered with information about add-ons that you are not using.
- The whole directory structure has been overhauled - there are now three main folders.

This new structure makes system upgrades much easier and also facilitates the installation on embedded systems where one does not want to use flash memory for continuously writing logfiles etc.



OpenHAB 2

Three main folders:

- "runtime" containing the binaries and other content that is needed to run the system.
- "conf", which holds all your personal configurations and customizations (such as e.g. custom icons).
- "userdata", which is the only directory that the system actively writes to (log files, databases, etc.).

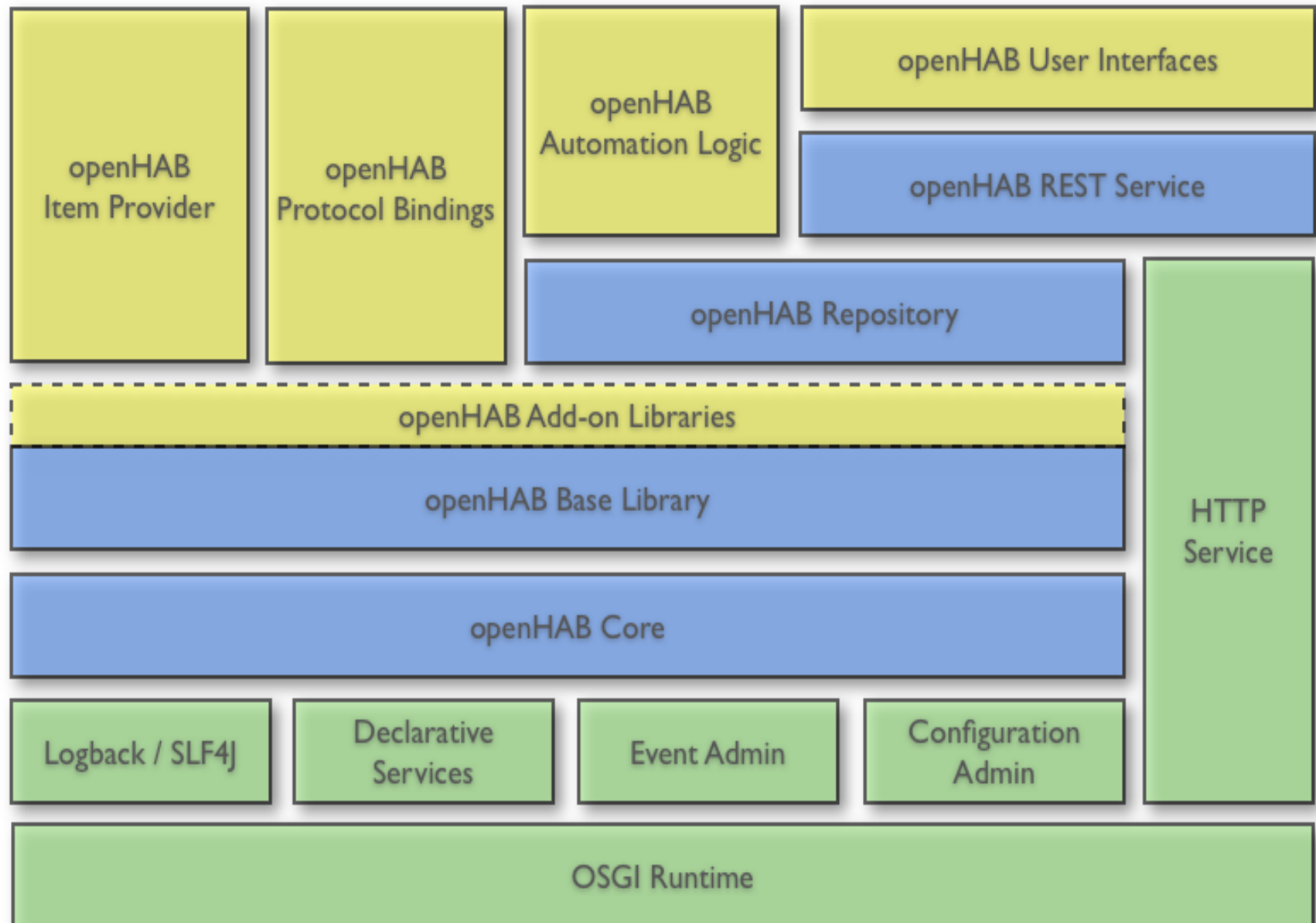


OpenHAB

- A core concept for openHAB is the notion of an “item”. An item is a data-centric functional atomic building block - you can think of it as an “capability”.
- All features offered by openHAB are using this “item” abstraction, which means that you will not find any reference to device specific things (like IP addresses, IDs etc.) in automation rules, UI definitions and so on.
- A very important aspect of openHAB’s architecture is its modular design. It is very easy to add new features (like the integration with yet another system through a “binding”) and you can add and remove such features at runtime.

openHAB Architecture Overview

- openHAB Add-ons
- openHAB Core Components
- OSGi Framework



Sitemap

openHAB comes with a generic textual configuration for its user interfaces: The so-called Sitemap. The sitemap is a tree structure of widgets, which define the different pages of a UI and their content. Widgets can be associated to items, for which they should show the status and/or control elements.

DEMO SETUP

OpenHAB

openHAB comes as a platform independent zip file, which you only need to extract to some folder.

You will find the following folders:

- conf: This contains all your user specific configuration files.
- runtime: This contains the openHAB binaries, there should normally be no need to touch anything in here - the whole folder can be considered to be read-only.
- userdata: Here you will find all the data that is generated during runtime: log files, database files, etc. In theory this should be the only folder where openHAB needs write permission on.
- addons: Here you can drop add-ons (or any other OSGi bundles) that you want to be deployed in your instance. These can be add-ons for openHAB 1.x and 2.x likewise.

OpenHAB

The demo package consists of example configuration files and samples of add-ons and UIs. In order to install the demo, you must edit the file 'conf/services/addons.cfg'. Uncomment the line `package=` and set it to demo as shown below:

```
# The base installation package of this openHAB instance (default is "standard")
# Valid options:
#   - minimal : Installation only with dashboard, but no UIs or other addons
#   - standard : Typical installation with all standard UIs
#   - demo    : A demo setup which includes UIs, a few bindings, config files etc.
package = demo

...
```




OpenHAB

Starting the Runtime

Once you have configured your runtime as above, start the openHAB runtime from the terminal by calling `./start.sh` (`start.bat` on Windows)

Point your browser to `http://<hostname>:8080` (allow the runtime some time to start before the HTTP server is available, especially on the very first start) and you will be welcomed by the openHAB Dashboard, which is the entry point to the different web UIs

OpenHAB

Using the Shell

openHAB uses Apache Karaf and thus comes with a very powerful shell for managing the installation. Useful commands e.g. include:

log:tail: Show the live logging output, end it by pressing ctrl+c.

log:set DEBUG org.openhab.binding.sonos: Enables debug logging for a certain binding.

feature:list: Lists all features available and shows there status.

feature:install openhab-binding-knx: Installs a certain add-on (here KNX).

bundle:list -s: Lists all installed bundles with their symbolic name.

logout: Shuts down openHAB.



HABmin

HABmin is a modern, professional and portable user interface for openHAB, providing both user and administrative functions (eg sitemaps for users, and configuration utilities to aid setup).



HABmin

Features:

Responsive. Should work well on all devices. Of course some functions may be removed or be difficult to use on small devices (eg the graphical rule editor).

Theme-able. Multiple themes are available - take your pick (currently 3 themes). If you want a different look, we're using bootswatch themes - vote for your favourite by raising an issue.

Charting. Modern, fast charting of historical data.

Graphical rule editor. No need to learn rule syntax.

International support. Currently translated in English, Deutsch, Français. Add support for your language...

Available as native app for **Android**.



openHABian

- Hassle-free openHAB 2 Raspbian image as a minimal unattended netinstaller for the Raspberry Pi.
- The provided image of only 64MB contains a minimal boot system. This system will then install Raspbian followed by openHAB and a set of useful tools. All packages will be downloaded in their newest version.

openHABian

- openHAB 2 latest snapshot (package repository)
- Oracle Java 8 (build 1.8.0_101, needed for my.openhab)
- Samba (preconfigured)
- custom .bashrc and .vimrc files
- openHAB syntax highlighting in vim and nano
- uses whole SD card by default (8GB or 16GB SD card sufficient)
- 16MB GPU memory split
- git based versioning of etc by the help of etckeeper
- useful packages like screen, mc, htop ...



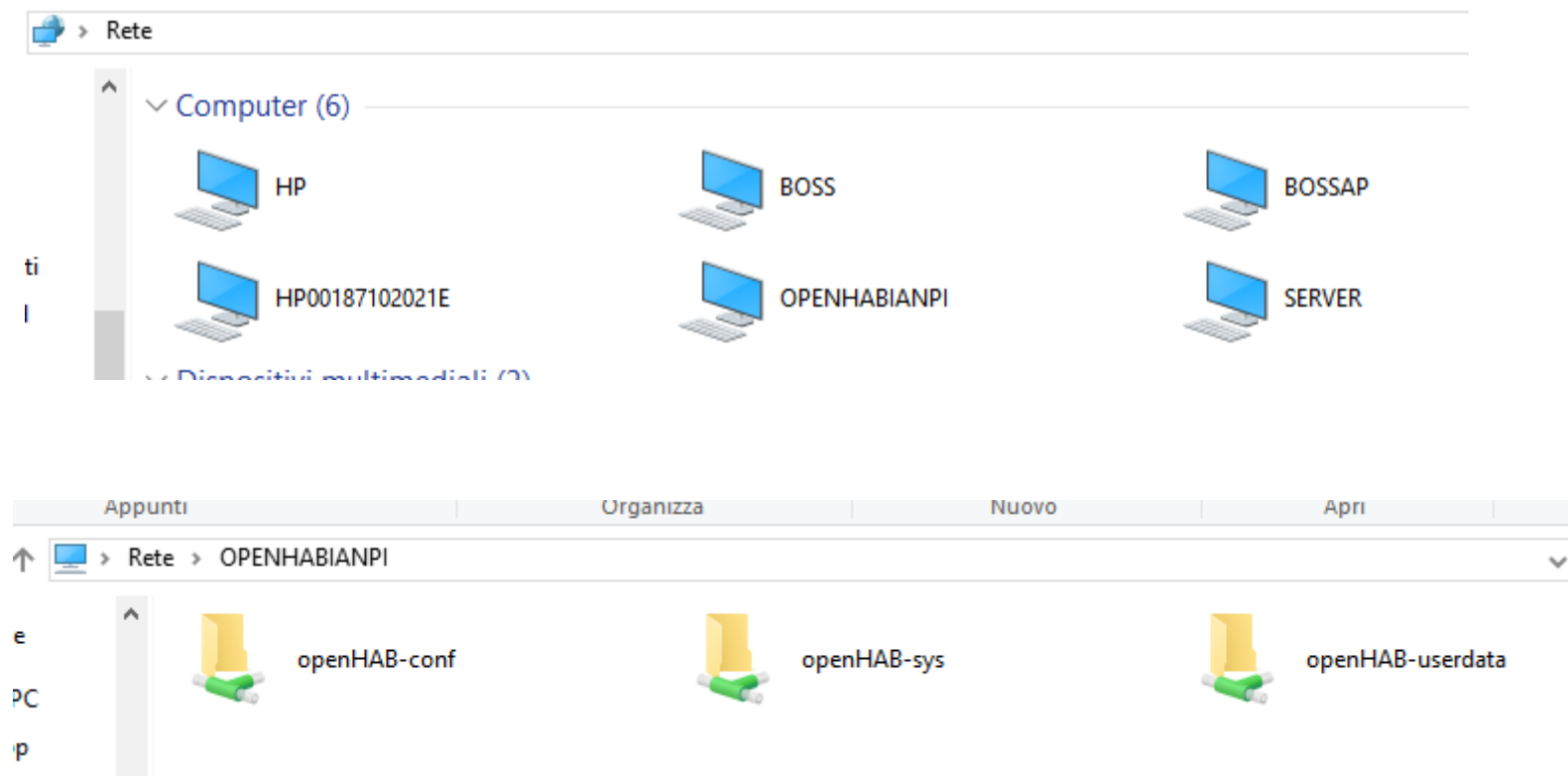
openHABian

Setup

- Connect Ethernet, SD card and power to your Raspberry Pi
- Wait up to 45 minutes (setup depends on your downlink as almost everything is downloaded live)
- Green LED will indicate when setup is finished
- Irregular blinking: setup in progress...
- Steady "heartbeat": setup successful
- Fast blinking: error while setup, check `/var/log/raspbian-ua-netinst.log`, create GitHub Issue
- Connect to the openHAB 2 portal (available after another 15 minutes): `http://openhabianpi:8080`
- Connect via ssh with `pi:raspberry`
- Connect to the Samba network share with `openhbab:habopen`

openHABian

SMB (samba)



“È chiaro che altri problemi come il fondamentalismo religioso, la sovrappopolazione, i danni all'ambiente e il dominio del business su governi, scienza, pensiero e società sono molto più grandi del software non libero. Ma molta altra gente ci sta già lavorando, e io non ho nessuna grande capacità o idee per indirizzarle. Così sembra che sia meglio che io continui a lavorare sul problema del software libero. Ad ogni modo, il software libero contrasta uno degli aspetti del dominio dell'economia sulla società.”

Richard Matthew Stallman(FSF)



salinaro@ld16bari :-> umount /dev/talk

#iovotoNOwindows