O-MI reference implementation

Table of Contents

1. Introduction

This is a project of Home Automation. Home Automation is the science of automating things within a house, room or other facility. It uses a wide range of different sensors for data input, and can implicitly react based on that data with automated adjustments of the environment and ecosystem.

2. Goal of this project

This project will be developed as a reference implementation for the Aalto University. The goal is to take one specific room in the University and 'automate' it as described in the introduction. This room can be used as an example of how the whole University could be controlled in the future. So you could call it a proof of concept, based on one can drive forward the technical infrastructure of the University.

3. The Room

The room that will be used as reference is within the Department of Computer Science of the Aalto University in Helsinki. The exact address is:

Konemiehentie 2, room B126, 02150 Espoo

4. The system architecture

There are multiple components, all together build the system. The core component is a O-MI node. O-MI is an Internet of Things data server. It implements the Open Messaging Interface as well as the Open data format for exchanging data and communicating with other components of the system. The O-MI node holds all the current data in a database. Agents can push data to the server and update the current state of the room, which is represented by the economic data delivered by the sensors. Agents can as well request those data and then decide to control hardware that is able to adjust the state of the room.

The input sensors are connected to multiple Raspberry Pi computers. On those computers there are running agents that keep track of the sensor inputs and push O-MI messages to the O-MI Node if there are data change events. So the O-MI node keeps up to date of the current room state.

As controlling hardware, web-enabled outlets are used. Their state can be switched with a simple HTTP request. Since the outlets are reachable via their IP-addresses, but the IP-addresses will most likely be assigned dynamically by a DHCP-server, it is not possible to store the IP-addresses of the outlets directly in the O-MI Node. To solve this, the unique MAC-address of each outlet is stored instead. Additionally there is an IP-management server that stores tuples of the MAC-addresses and current IP-addresses of the outlets. All of the outlets inform the IP-management server if their IP-addresses change.

To actually toggle a plug, one can send an write request to the O-MI node with the MAC address of the outlet and the requestet state. The O-MI Node writes the new state in its database and triggers a callback to a callback-server. This callback-server asks the IP-management server for the current IP-address of the outlet based on the given MAC address. Then the callback-server sends the HTTP request to the outlet, which causes the actual operation.
This architecture leads to a problem here - the new state of the outlet is stored in the database before the actual command is sent to the outlet. So if the outlet's state can't be modified for some reason, this leads to wrong entries in the database. Unfortunately this issue can't be resolved with the current version of O-MI node. Maybe there will be adjustments to the O-MI Node software in the future so that this issue can be resolved.

5. Input: used sensors

To keep track of the room, there are a few different sensors listed below.

- Temperature and light sensors
- Occupancy sensor
- Humidity sensor
- CO2 sensor
- Power Consumption sensor
- Contact sensor for door monitoring

For temperature, light and humidity monitoring, there are used multiSensors that combine two sensors each.
They are connected to the Raspberry Pi via 1-Wire bus, and are connected among themselves in a chain. On the Raspberry Pi there is an so-called 'External Agent' running (refer to the O-MI Node documentation for further information) that keeps track of the data and sends them to the O-MI Node.

6. Output: Used controlling hardware

For being able to control the room, Allnet All3073 outlets are used. They can be turned on and off from basically every machine that has network access, because they communicate via HTTP. You can plug in electronic devices and so switch it on and off. Electronic devices can be literally anything with a power plug, from a coffee maker over a TV until an air condition machine.
The used electronic devices in combination with the electronic outlets are:

- A fan to control the air flow in the room
- A standard lamp

7. Mounting the 1Wire-Hardware on a Raspberry Pi

- To install, follow this guide: http://wiki.m.nu/index.php/OWFS_with_i2c_support_on_Raspberry_Pi_(English_version)#Installation_of_OWFS

- To actually mount the hardware, use this command:

  sudo /opt/owfs/bin/owfs -u /dev/ttyUSB0 --allow_other /mnt/1wire/

  (you can put the command in /etc/rc.local for auto mounting on boot)

8. Controlling Output

As output controlling devices, Allnet ALL3073WLAN outlets are used. To use it successfully, there are a few configuration steps needed. This can be done in the outlet's own web-interface. To access it, follow the instruction manual.

- First, open the web interface of the (factory resetted) outlet, choose language, and apply the default network settings without changing anything. Save inputs.

- Then go to Configuration/Device settings --> Remote control, enable both - read only and switching - and enter *aalto* as username and password. Save inputs.

- Next, go to Configuration/Web server and users --> SSH Server setting. Set an SSH password (default would be *aalto*). Note that the SSH server setting is set to enabled. Save inputs.

- Copy the scripts from the folder ip_management/client_scripts/ to the outlet. The simplest way is to use SCP for that, but you can use SSH as well. Copy the script 'say_hello.sh' to /root and the script 'S90_sayhello' to /etc/init.d. Next, make sure both scripts are executable (chmod 775). These scripts will automatically broadcast the devices' IP address to the ip management server if it changes. To identify each of the outlets, the MAC address of the wired network adapter is used.

- After this step is done, go to Configuration/LAN settings and set the network settings to DHCP to automatically optain an IP address. The device will reboot, and from now on the device will notify the IP management server every time the IP address changes.

- **If used as wireless device:**
  - Configure WLAN settings (SSID, password) too.
  - **!! CATUTION: wifi only works when lan settings is set to DHCP !!**

- Optional: set things like device name, description etc.

==========

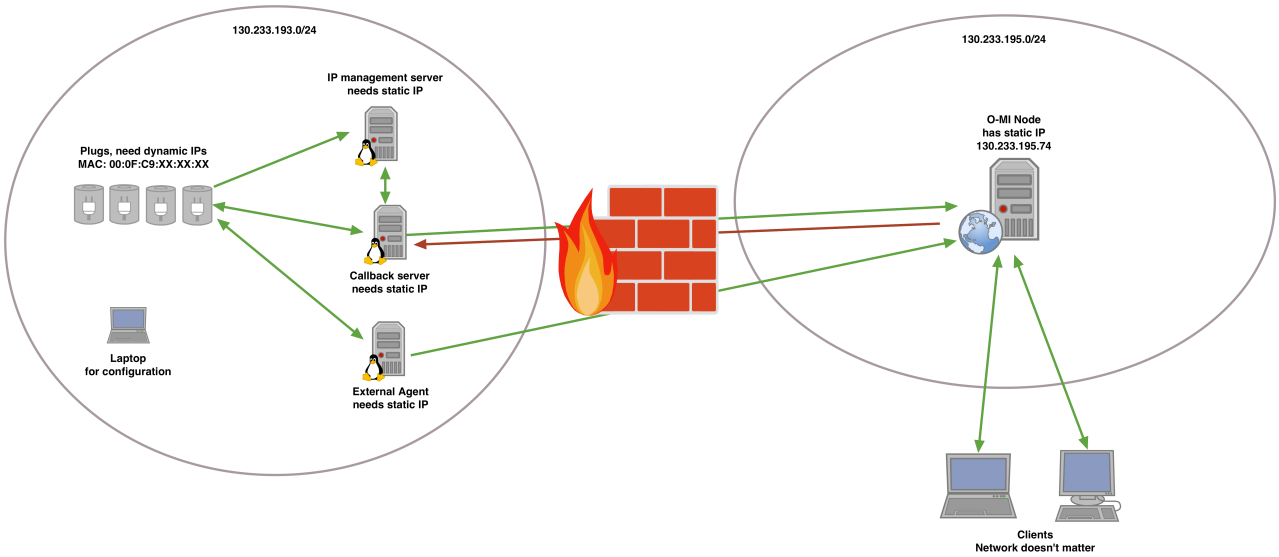
- **IMPORTANT:** The IP management server needs to be in the same subnet as the outlets, otherwise the communication will be blocked by the Aalto firewall. If you face any problems, consider asking the IT guys.

==========

- On every new plug, you should print its MAC address to identify them. By now, the MAC addresses are just printed and fixed with scotch tape. To hold a consistent look, the font type is Bitstream Vera Sans Mono, size 13, bold.

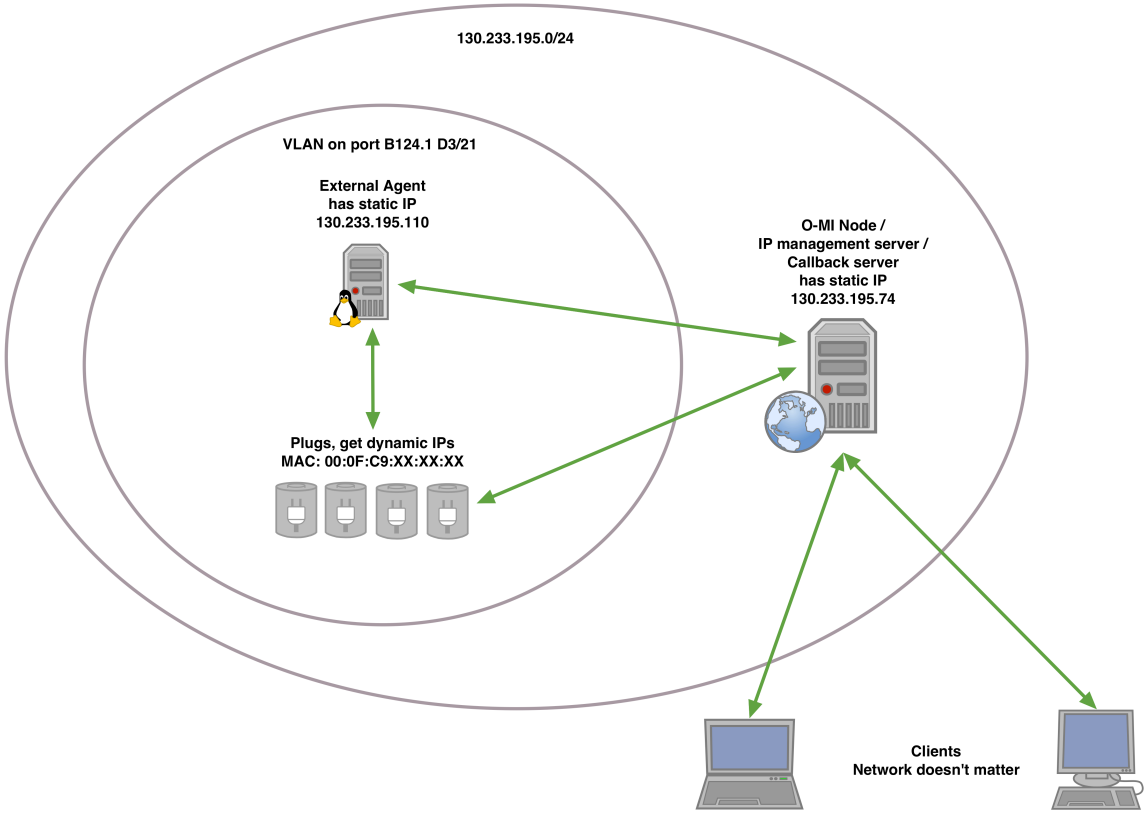
9. University network situation

Since the Aalto internal network has a lot of restrictions, it is not that simple to port the application from development status, where everything runs on development machines, to the real world scenario. Basically the main problem here is that the official O-MI server running at the University lies in a completely different subnet than all the other machines that get dynamic IP addresses assigned through a DHCP service. Unfortunately the firewall restricts cross-subnet-connections. So machines from different subnets can of course access the O-MI Node server, but the server cannot access machines that lie outside of its own subnet. The following diagram shows the connections between all involved network devices. The red arrow is the connection that gets blocked by the firewall.



For this reference implementation, I worked on a solution together with the IT guys that manage the network. As a result, we built an overlay VLAN on a specific physical network port in my Office (port B124.1 D3/21 in room B126). On this port we connected a switch, and on the switch we connected all the devices that needed to be at the same subnet than the O-MI Node server. The VLAN lies in the same subnet as the O-MI server, so all the devices connected get IP addresses inside that subnet too.
To make things easier, the callback server and the ip management server run on the same physical machine than the O-MI Node, so there are two less devices to take care of. So only the plugs and the external agent device have to be plugged in the switch. Inside the VLAN there is a MAC filter that filters the vendor part of each MAC address. If the vendor part points to an ALLNET device (the manifacturer of the plugs) then the device get a dynamic ip address. So there is no need to explicitly add a new dhcp entry for each new plug.
The following diagram shows the new network structure concerning this project. It is much simpler, however the disadvantage is that all plugs and the external agent device has to be plugged into one specific physical network port. If this system should be used

across multiple rooms there has to be figured out another solution with the IT guys. The current network security policy restricts any more inplicit solutions.



10. Integration in Otaniemi3D

Otaniemi3D is an interactive user-friendly web-frontend for the O-MI Node, specially made for the Aalto University buildings. So the goal is to integrate the room with all its sensors and plugs to that system so that everyone (respectively the authorized personal) can interact with the room.
To achieve this, several steps are neccessary. First is that the actual building in which the room is is not yet implemented in Otaniemi3D, however the University holds the neccessary data to do that. Because of this, an empty room in the current model is used instead of the original room. So here is the mapping:

Original: ***Room B126*** ←→ Mapped: ***Room-147a***

This results in the following: all external agents push its data not to the O-MI id */CS Building/Room 126* but to */K1/Room-147a*. By doing this, the integrity and generic features of the O-MI Node system won't get touched.

However, there were several code changes neccessary in order to get the plugs working with Otaniemi3D. The system was originally meant to be read-only, so you could retrieve sensor data from the O-MI Node but you could not do write requests to it. So what I basically did was to add a new sensor type 'plugs' which will get recognized by the Otaniemi3D. The condition lays in the name of the sensor - the plugs' names are their IP-addresses, so the system checks for exactly 5 ':' in the name and claims it as plug if so.
When a sensor is claimed as a plug, an additional button will be shown behind the sensor value, which you can use to send a write request to the O-MI node and initialize the toggle process.

For the panorama view of the room, the image files and the appropriate xml file were added to the Otaniemi3D instance running at the university. The 3D-view uses krPano as viewer, which has its own plugin system. So basically the code changes made in the Otaniemi3D heatmap were done again inside the relying krPano plugin. Here was another challenge - the shown tooltip containing the sensor data was only shown while the mouse hovered the sensor hotspot. But to insert a button in this tooltip that is actually clickable, the tooltip has to stay somehow persistent so that the mouse cursor could reach the button and click it. That was not possible with the stock krPano plugin, so there was to need the use of another plugin called 'textfield.swf'.

All the code changes were pushed on a new branch in the Otaniemi3D github repository. A pull request was made so the main developers can populate the code in the official branch.

11. Conclusion

The O-MI Node is an open source data server specially designed for Internet of Things. It was meant to be only for storing sensor values in a central place, which has not unconditionally to be a cloud but rather your own server in your own home. With all the data in one place, you can access them everytime from everywhere and keep track of your house for example.
With the changes developed in conext of this work, the system is now able to not just passively keep track of the sensor data, but rather actively react to those data and physically change something in the house. The functionality was implemented using the example of outlets which are switchable over the internet. Those outlets can be controlled via the O-MI Node and so the user can remotely turn electrical devices on and off. That is a huge advancement for the system - with the example of outlets, you can extend the service by for example electrical roller shutters and so step by step transform your home in a completely monitored and controllable state.
In my opinion, the next step would be to write software that monitors all those sensor values and manages to make those physical adjustments in the home automated. Because that is the actual advantage of IoT: it is not just about storing and accessing lots of sensor data, in fact it is more about automated adjustments without the need to take action themselves.