Power Management of Laptops Batteries in Dynamic Heterogeneous Environments Using iPOPO

Shadi Abras[1,3], Thomas Calmant[3], Benoit Delinchant[1], Stéphane Ploix[2], Frédéric Wurtz[1], Mahendra Pratap Singh[1]

Univ. Grenoble Alpes, G2Elab[1], G-SCOP[2], LIG[3], France

***{Benoit.Delinchant, Frederic.Wurtz}@g2elab.grenoble-inp.fr**
***{Stephane.Ploix}@gscop.grenoble-inp.fr**
***{Shadi.Abras, Thomas.Calmant}@imag.fr**

ABSTRACT. This paper contributes to the design of a smart monitoring system. The objective of this paper is to show that by using iPOPO framework, as an embedded application for devices, it is possible to improve the configuration/reconfiguration of monitoring system.
The proposed experimentation platform, called PREDIS/MHI, based on iPOPO framework, is characterised by its openness, its scalability and its capability to manage diversity.
In this paper, we show how application based on iPOPO framework inspired from OSGi, well adapted to represent problems spatially distributed and opened, can dynamically be adapted to various contexts of environments. This paper presents an energy management application for the laptops batteries : optimal storage life management of laptops batteries according to power supplied by photovoltaic panels and by the grid.

KEYWORDS. Ambient intelligence, OSGi/iPOPO, Energy management, Smart-Home based architecture component.

RESUME. Cet article contribue à la conception d'un système de surveillance intelligent. L'objectif de cet article est de montrer qu'en s'appuyant sur la plate-forme iPOPO inspirée d'OSGi, comme une application embarquée pour des equipements, il est possible d'améliorer la configuration / reconfiguration du système de surveillance.
La plate-forme d'expérimentation proposée, appelé PREDIS/MHI, basée sur la plate-forme iPOPO, se caractérise par son ouverture, son extensibilité et sa capacité à appréhender la diversité.
Dans cet article, nous montrons comment l'application basée sur la plate-forme iPOPO, bien adapté pour représenter des problèmes ouverts et distribués, peut être adaptée dynamiquement à différents contextes d'environnements. Cet article présente une application de gestion de l'énergie pour les batteries des ordinateurs portables : la gestion de stockage optimale des batteries des ordinateurs portables en fonction de l'énergie fournie par les panneaux photovoltaïques et par le réseau d'électricité.

MOTS-CLÉFS. Intelligence Ambiante, OSGi/iPOPO, Gestion d'énergie, Maison intelligente à base de l'architecture des composants.

1 Introduction

A monitoring system basically consists of devices, sensors and actuators linked via a communication network allowing interactions for control purposes. Thanks to this network, a load control mechanism can be implemented : this is called distributed control. A monitoring system is made of sensors, which measure physical quantity, and actuators, which control some devices such as computer batteries. However, faced to the increasing complexity in buildings, it is necessary to develop monitoring systems able to plan ahead changes of configuration of buildings. The development of a dynamic environment composed of heterogeneous devices and technologies is very complex. Smart dynamic environments can be found in many domains such as : domestic, social, biology, medical, etc.

(Lesser et al., 1999) proposed a GatorTech Smart House system. Sensors and actuators are fitted on a number of devices : mailbox, entrance door, floor etc which are connected to an operational platform designed to optimise the comfort of the inhabitant. It also uses a high precision ultrasonic tracking system to locate occupants and evaluate their mobility habits to better control the environment (Helal et al., 2005).

(Michael, 1998) proposed an adaptive control of home environment. It monitors the environment and observes the actions taken by the inhabitants (using lights, adjusting thermostat). This data is further used to infer patterns in the house, using reinforcement learning, a stochastic form of dynamic programming.

In the House of the Future project (Tapia et al., 2004) aimed to design strategies for more flexible environments that meets inhabitants physical and cognitive needs than current environments. The system consists of three components, a set of state-change sensors used to collect data about the use of objects, a context-aware Experience Sampling Tool (ESM) used by the inhabitant to label his activities, and pattern recognition and classification algorithms for recognising activities. Naive Bayesian classifier is used to train the model and predict future activities of inhabitants.

However, the previous works based their control system on a centralised approach ; they do not opt for solutions fostering modularity.

Many recent works provide solutions with many facets to cater cohesive integration between building subsystems (Davidsson et Boman, 2005). There are also several solutions specifically targeting intelligent integration which have come under many forms including the use of artificial intelligence to make buildings more intelligent (Duangsuwan et Kecheng, 2008). In general, many of these current subsystems use proprietary information patterns and communication protocols that prevent the easy modification of the intelligent building operation, and at the same time limit the plug and play applications and the addition of new components in the system. Such implication leads to interoperability

Problem statement

Owing to the number and diversity of devices in the home, leads us to opt for solutions fostering modularity. The need for structural self-adaptation, or more technically the need for plug-and-play appliance, leads to a Service-Oriented Architecture (SOA) platform (Perumal et al., 2010).

Indeed, a home monitoring system must be open-ended and extendible : it must be possible to add or remove appliances (or new types of sensors/actuators) at any time without calling into question the overall operation of the system. It must allow to implement or to modify the energy management policy without calling into question the overall operation of the system.

The framework iPOPO (in Python inspired from OSGi) has been used in order to develop a smart dynamic environment for PREDIS/MHI : Monitoring & Habitat Intelligent. It is an experimentation platform consisting of many kinds of sensors/actuators of different technologies such as X10, Oregon Scientific, Zigbee, OPC, Philips Hue, HTTP GET and USB (Fig. 1).

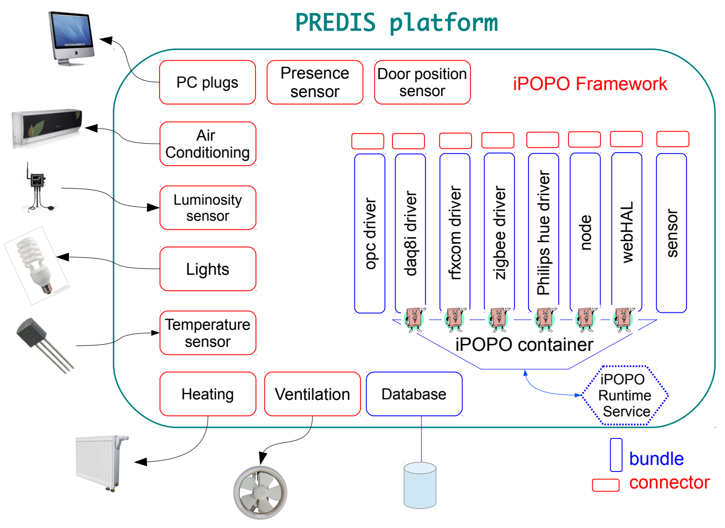


FIGURE 1. Components of PREDIS platform

The objective of this study is to design a building electric energy management system able to determine the best energy assignment plan for PREDIS/MHI laptops batteries, according to given criteria. A building energy management system can be seen from two points of view : a) load management and b) local energy production management (ex. Photovoltaic panels) in smart dynamic environment.

In what follows, firstly, the paper describes briefly the PREDIS/MHI platform. Secondly, it describes the architecture of the iPOPO-based implementation of PREDIS platform. Thirdly, it presents the two main mechanisms dedicated to energy management of computer batteries which exist in the PREDIS/MHI platform. Finally, a conclusion of the contribution of the iPOPO framework, based on SOA techniques, to the monitor system is presented.

2 PREDIS/MHI

PREDIS is a smart building experimentation, which aims to ensure better energy management and human comfort in buildings : the building's maximal energy consumption did not exceed 50kWhPE. $(PrimaryEnergy)/year.m^2$

It combines physical models and experimental measurements in order to have more adapted models for virtual simulation and optimal control.

This platform is developed around :

- Multi-sensor monitoring,
- User activities and their energy impact analysis,
- Multi-physical modelling, measurement handling and sensitivity analysing.

The first floor of this building is composed of two rooms : a teaching room and the researchers' open space. The teaching room is used for training courses of Grenoble University and equipped with 15 laptops, which are connected to the **electrical grid** and to **photovoltaic** generators. The researcher open space, where the control, supervision and measurement are installed, is daily occupied by about six researchers.

The first floor of PREDIS is equipped of many systems :

- Heating, Ventilation and Air Conditioning system,
- Lighting control,
- Experimental instrumentation : temperature sensors, airflow sensors, CO_2 sensors, energy meters, and switching devices.

In this platform, a system called HAL (Home Abstraction Layer) has been added as a general interface to the control systems and sensors/actuators. HAL integrates drivers of different communication protocols relating to physical devices. It shows data through a RESTful Web interface, accessible via a web browser or a client application. Its objective is to provide a uniform access to devices and information. The HAL system has been developed in Python because most of sensors drivers have been provided in this language. Figure 2 illustrates the architecture of components of the presented system (HAL).

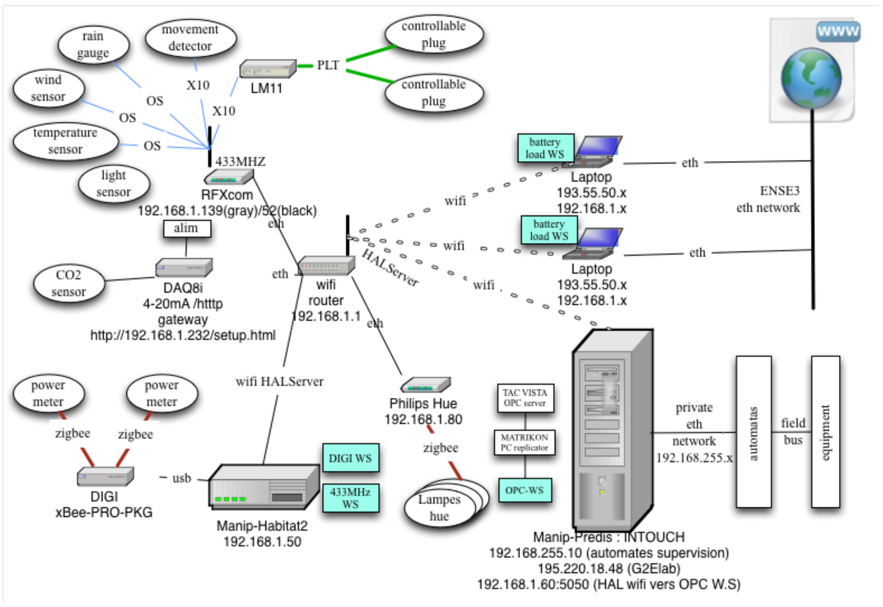


FIGURE 2. Architecture of HAL system components

In figure 2, many types of sensors are used : temperatures, humidity, CO_2, airflows, electric energy, electric power, light, presence, door position, etc. Theses sensors rely on many technologies : X10, Oregon Scientific, Zigbee, Philips Hue, HTTP GET and USB. There is also a legacy energy management system : an Intouch SCADA system, which can be connected via the OPC protocol which works with another layer : Modbus, Lonworks and Dali protocols. This SCADA system is also available via the HAL system.

Once the HAL system has been deployed in the PREDIS system, all functions and facilities of this system can be used. However, if a driver linked to sensors/actuators fails, the entire integrated system does not well work : it leads to the failure part or the whole of the system. Indeed, when an additional sensor or actuator is put in operation, it call into question the overall operation of the system : the system must be restarted in order to take into account the new appliance. If a driver linked to sensors/actuators is put out of operation, either accidentally or through administration action, the system and its peripheral components are severely affected. Indeed, if the energy management policy is changed, it calls into question the overall operation of the system. In the HAL system, the life cycle of each sensor, actuator, driver or control algorithm is managed by the developer. For example, for each time, when a new sensor is added to the system, the developer has to do new configuration for the whole system, which takes some time as it implies to restart the system. The HAL system depends on the life cycle of sensors and the configurations defined by the developer. The HAL system has to manage the access to the functionalities provided by each each of its elements. But, it has also to manage the dynamism of the models representing the environment, which should be outside its scope. These two aspects of the system being particularly different, the implementation of the HAL system has become complex, leading to a number of malfunctions.

Progress in the fields of Service-Oriented Architectures (SOA), Component-Based Software

Engineering (CBSE) and Service-Oriented Component Models (SOCM), which are particularly adapted to distributed and open problems, is likely to lead to a "Home Monitoring System" made up of bundles providing support for the different sensors and actuators. Those bundles could, with embedded algorithms of intelligence, make joint decisions with respect to the operating requirements of the appliances to which they are connected. This is why the iPOPO framework (Calmant et al., 2012), an SOCM framework, was selected.

This HAL system requires a specification-based substitutability, which has been implemented using iPOPO (Abras et al., 2014) for several reasons, including :

- The drivers of sensors were already developed and provided in Python.
- iPOPO is inspired on the OSGi specifications, ensuring a robust SOA API.
- It supports Remote Services natively, allowing to turn the application in a distributed architecture easily.
- iPOPO combines many advantages :
  - Simplicity : iPOPO provides a very simple development mode. It provides an excellent foundation for building dynamically extensible Python-based applications.
  - Power : Supporting the principles of OSGi services in Python, iPOPO provides many other features that aim to simplify developing sophisticated applications. For example, iPOPO supports configuring components, notifies of component events, provides a publish-subscribe service, etc. It also provides an embedded HTTP server, specified as an HTTP service, that will heavily simplify the implementation of PREDIS/MHI.
  - Performance : iPOPO is small and is designed to stay small. The core size of iPOPO is less than 10KB. It has been optimized to instantiate a large number of components in a short time.

A **Java** version of **Pelix Remote Services** has been integrated. It allows Pelix/iPOPO components to interact with OSGi/iPOJO (Escoffier et Hall, 2007) ones.

In the next section, a system's control architecture is presented to determine the best energy assignment plan for laptops batteries in PREDIS/MHI, according to given criteria. This system can be seen from two points of view : a) load management and b) local energy production management (ex. Photovoltaic panels) in smart dynamic environment.

3 Energy Management System

3.1 Architecture of Energy Management System for Laptops Batteries

The objective of this work is to design a Home Monitoring System on a dynamic, modular, configurable and controllable basis.

Figure 3 shows the structure of energy management system for laptops batteries in PREDIS/MHI, based on the notion of services.

The different components of this architecture are implemented in service-oriented component model. The OSGi/iPOJO (Java) and OSGi/iPOPO (Python) frameworks are both used in order to develop all components of the systems. Each component in figure 3 represents a part of information allowing to perform the optimal control for laptop batteries presented in the next section :

- *ENSE^3 Calendar* : this component gives time information on reserved courses in the classroom of PREDIS/MHI. This allows anticipating the charge of laptops batteries, to be available for students when using.
- *Photovoltaic panels* : this component gives information, in real time, on the produced energy of photovoltaic panels installed near of the PREDIS/MHI site. This shows whether the laptops batteries can be charged with the energy coming from photovoltaic panels or form the grid.
- *PC Battery Info* : this component gives information about the state of charge of the

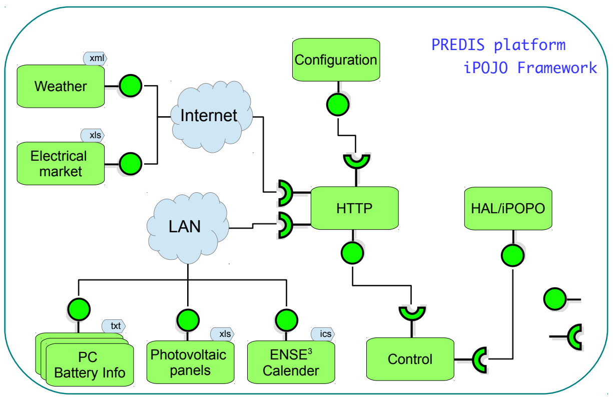


FIGURE 3. Structure of energy management system for laptops batteries in PREDIS/MHI

battery of each computer to determine if an emergency situation will be trigged or not. This component can send, when needed, estimation about the remaining battery time for the current activity, the remaining time for charging the battery, the full battery time and the total time for charging.

- *Electrical market* : this component gives information about the energy prices, for each period of one hour, coming form the energy grid.
- *Weather* : this component gives information about the sunshine in order to anticipate the energy production of Photovoltaic panels.
- *HTTP* : this component collects all information coming from the previous components and lets it be available for the control mechanism.
- *Configuration* : this component handles the configuration of all components such as the IP address, port, web address, etc.
- *HAL/iPOPO* : This component is presented in details in (Abras et al., 2014). It applies the plan coming form the control component to switch off/on the laptops.
- *Control* : This component contains the two main mechanisms (reactive & anticipative) relying on the information collected from the previous components.

The HAL/iPOPO service in figure 3 is implemented in Python using OSGi/iPOPO. The other components are developed in Java using OSGi/iPOJO. The Pelix Remote Services let Pelix/iPOPO components interact with OSGi/iPOJO ones directly.

Once all information is available from various components, the control mechanism could apply a strategy for charging the laptops batteries. This strategy is described in the next section.

3.2 Control Mechanism

The biggest difficulty when addressing the problem of assigning energy resources for laptops in PREDIS/MHI is taking into account very different dynamics. Some phenomena require very short response times, such as violation of the maximum resource requirement calling for swift management of conflicting energy demands. There are also some physical phenomena that are relatively slow such as the price of energy bought or local energy production capacity such as solar energy. Hence, the system's control architecture must be able to manage the periodic cyclical phenomena occurring in the PREDIS/MHI. The proposed system has two main control levels. These levels can be identified according to the different sampling periods and time horizons : the anticipative mechanism and the reactive mechanism.

The reactive mechanism is a short time adjustment mechanism which is triggered when the level of laptops batteries falls below weak values (10% for example). This mechanism reacts quickly to avoid violations of energy constraints. Therefore, the reactive mechanism adjusts, at a short sample, the set points coming from the predicted plan generated by the anticipative mechanism.

The anticipative mechanism aims to avoid frequent emergency situations. The objective of this mechanism is to compute a plan for production/consumption of laptops batteries in PREDIS/MHI according to local energy production (Photovoltaic panels) and the laptops usage in PREDIS/MHI. This plan can be directly transmitted to laptops or adjusted by the reactive mechanism in case of constraint violation.

This paper focuses only on reactive mechanism. This reactive mechanism can be summarised using algorithm Alg. 1.

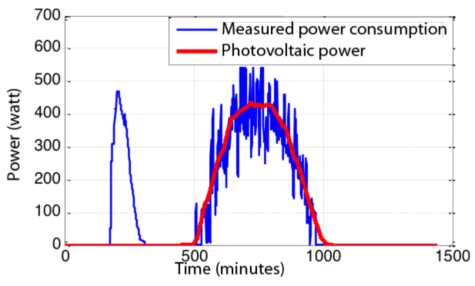
```
Require : Emergency ← Emergency level for laptops;
For (period ∈ reactive periods) do
    sum = 0;
    Require : PV.production ← Real production for this period from Photovoltaic Panels;
    For ((PC ∈ List of laptops)) do
        Require : PC.SOC ← PC Info : State of charge;
        Require : PC.Set_Point ← Set point of PC from anticipative plan;
        If (PC.Set_Point = ON) then
            Switch PC ON;
            sum = sum + PC.consumption; [if PC.production ¡ sum : use PV; else use grid]
        else
            If (PC.SOC < Emergency) then
                Switch PC ON;
                sum = sum + PC.consumption; [if PC.production ¡ sum : use PV; else use grid]
            else
                Switch PC OFF;
            end If
        end If
    end For
end For
```

Algorithm 1: *Reactive mechanism.*

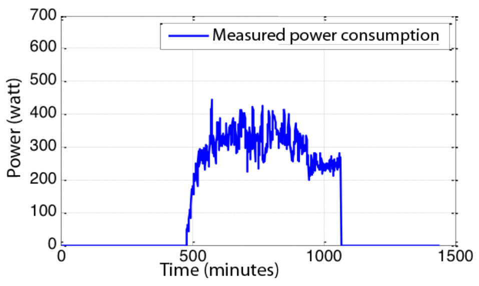
3.3 Discussion

Figure 4 shows the results of laptops consumption during one day.

Figure 4 (left side) expresses the total power consumption of laptops using the control algorithm and the power produced by Photovoltaic panels. The power consumption of laptops follows the production curve of Photovoltaic panels during the sunshine period (between 8am and 6pm). In other words, the produced power has been used by the laptops. Indeed, a power consumption is done (between 2am and 5 am), where the price of electricity coming from the grid is the cheapest one. If the control algorithm is not used, figure 4 (right side) shows the total power consumption where the laptops consumes power when need. Concerning the consumption cost, it is much cheaper : it passes from 5.8 euros, without control system, to 0.5 euros when the control algorithm is applied, for the month of May 2013.



”*With the management system.*



”*Without the management system.*

Figure 4. ”*Total laptops consumed power during one day.*

4 Conclusion and Perspectives

Service-Oriented Component Models are an excellent way to ensure that developed software comes with built-in best practices such as reusability, high cohesion, loose coupling, high flexibility and dynamism. The proposed system based on iPOPO does offer advantages over the centralised approach : its openness, its scalability and its capability to manage diversity.

The different components of this architecture are implemented in service-oriented components using OSGi/iPOJO (Java version) and OSGi/iPOPO (Python version). The Pelix Remote Services let Pelix/iPOPO components to interact with OSGi/iPOJO ones directly. The decoupling implied by remote services also allows to use sensors developed in different languages.

This paper has presented the structure of a monitoring system decomposed into two main mechanisms working on a different time scale : the reactive mechanism and anticipation mechanism.

This paper has presented an energy management application for the laptops batteries : optimal storage life management of laptops batteries according to power supplied by photovoltaic panels and by the grid.

Références

Abras, S., Calmant, T., Ploix, S., Donsez, D., F. Wurtz, O. G., et Delinchant, B. (2014). Developing dynamic heterogeneous environments in smart building using ipopo. In *the 3rd International Conference on Smart Grids and green IT Systems*, Barcelona, Spain. Springer.

Calmant, T., Americo, J., Gattaz, O., Donsez, D., et Gama, K. (2012). A dynamic and service-oriented component model for python long-lived applications. In *Proceedings of the 15th ACM SIGSOFT symposium on Component Based Software Engineering*, CBSE ’12, pages 35–40. ACM.

Davidsson, P. et Boman, M. (2005). Distributed monitoring and control of office buildings by embedded agents. In *Intelligent Embedded Agents*, pages 293 – 307.

Duangsuwan, J. et Kecheng, L. (2008). Multi-agent control of shared zones in intelligent buildings. In *Computer Science and Software Engineering, 2008 International Conference on*, pages 1238–1241.

Escoffier, C. et Hall, R. (2007). Dynamically adaptable applications with ipojo service components. In *Proceedings of the 6th international conference on Software composition*, SC’07, pages 113–128. Springer-Verlag.

Helal, S., Mann, W., El-Zabadani, H., King, J., Kaddoura, Y., et Jansen, E. (2005). The gator tech smart house : A programmable pervasive space. In *IEEE Computer*, pages 50–60.

Lesser, V., Atighetchi, M., Benyo, B., Horling, B., Raja, A., Vincent, R., Wagner, T., Xuan, P., et Zhang, S. (january 1999). The intelligent home testbed. In *Proceedings of Autonomy Control Software Workshop*, page 8.

Michael, C. (1998). The neural network house : An environment that adapts to its inhabitants. In *Proceedings of the AAAI Spring Symposium on Intelligent Environments*, pages 110–114. AAAI Press'98.

Perumal, T., Ramli, A., Leong, C., Samsudin, K., et Mansor, S. (2010). Middleware for heterogeneous subsystems interoperability in intelligent buildings. In *Automation in Construction*, pages 160 – 168.

Tapia, E., Intille, S., et Larson, K. (2004). Activity recognition in the home using simple and ubiquitous sensors. In *Proceedings of Pervasive 2004, vol. LNCS 3001*, pages 158–175. Springer-Verlag.