TTM3 Smart Heating System Report

Xiao Chen, xiaoch@stud.ntnu.no
Pavel Arteev, pavela@stud.ntnu.no

October 2013

1 Introduction

By the magnificent growth of using technology in the daily life, the concept of the smart housing system is raised these years. Since smart cell phone and other daily using device have more and more functions than its original function, the self-adaptive system is necessary service to control and manage other devices in the house.

Self-adaptive system in this report will be also dynamic component system. It is the system emphasis on context constraints that govern how components may be composed (without prescribing a particular structure), and compositional adaptation is bounded by the components.

The project of this report will be implemented in OSGi[7] framework. Since this project would demonstrate a dynamic component system scoped in smart heating system in the house, OSGi framework is a good choice here. The OSGi framework is a module system and service platform for the Java programming language that implements a complete and dynamic component model, something that does not exist in standalone Java/VM environments. Applications or components (coming in the form of bundles for deployment) can be remotely installed, started, stopped, updated, and uninstalled without requiring a reboot; management of Java packages/classes is specified in great detail. Application life cycle management (start, stop, install, etc.) is done via APIs that allow for remote downloading of management policies. The service registry allows bundles to detect the addition of new services, or the removal of services, and adapt accordingly.

There are several benefits point for OSGi framework using for Java project, since it improves some not advanced part of the Java programming language in self-adaptive system.

For Development time benefits[3]:

- Strict development time (and runtime) enforcement of module boundaries
- A service-oriented architecture that works for managing service dependencies between modules.
- Better ability to structure development teams the way you want to

- Faster team-based development
- Faster testing cycles
- Support for versioning as part of dependency management
- Less road-blocks

For Runtime benefits:

- Full information about the installed modules and their wiring is available at runtime - a level of insight operations teams have never had before.
- Isolate changes
- Share dependencies
- Use just the server facilities you need

However when we first time used OSGi framework, it is not so easy to use in this project which we will discuss in the conclusion section of this report for some feedback points.

This project will focus on implementation a self-adaptive system, smart heating system with four heater manager simulator and four heater device simulator and two temperature sensor simulator. The communication channel between heater device and temperature sensor is through heater manager. And all the communication will be based on Hydna[1], which is a hosted backend into which you can send data and have in instantly appear on other devices. More detail about the system design will be covered in the section 2.

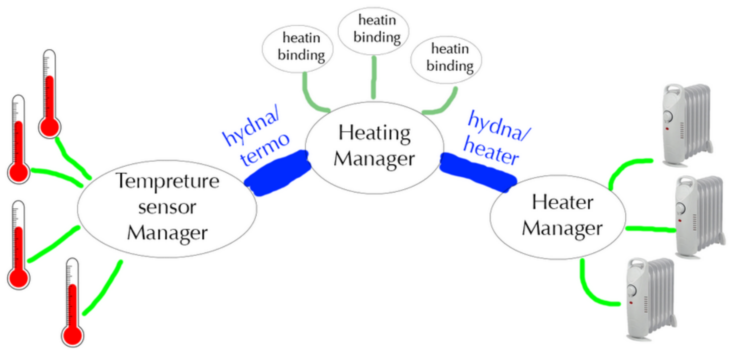


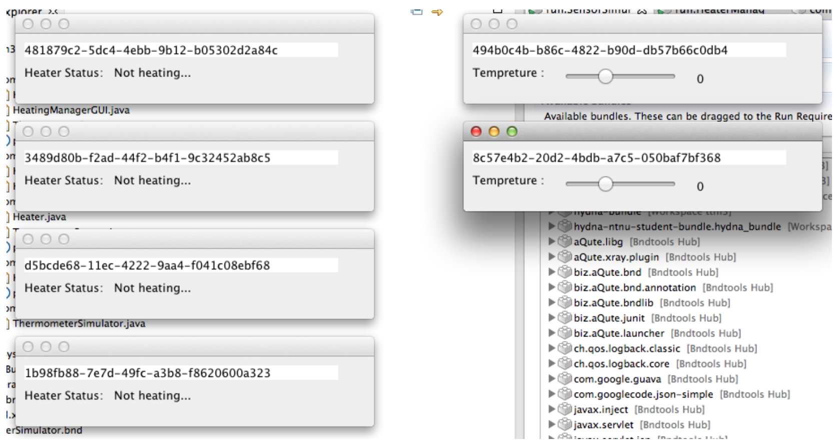
Figure 1: Smart Heating System Architecture

2 System Design

At the design and preparation stage for this project, there is an important feature should be taken account when development, which is this self-adaptive system need to be easy to modified for other services in the smart house scope. After that, we decide to make this project working for big range of different services in the smart house scope by making the complicated central management and easy client devices architecture. The main architecture of the system is shown in Figure 1.

2.1 Temperature Sensor Manager

Temperature Sensor Manager is a service to provide control command to temperature sensor and observe temperature sensor client. In this prototype project it is included in the Heater Simulator OSGi bundle. In this prototype it has an user friendly control panel to demonstrate the temperature sensor client changes. In the normal use case it should communicate with other 'stupid' temperature sensor in some local communication channel, like Zigbee, Bluetooth or Radio Frequency. The main functions of temperature sensor manager are to get the raw data from the temperature sensor (in prototype project which is only string type data) and to send these raw data to the Heating Manager through remote communication channel (Hydna backend service in this project, channel domain: ithouse.hydna.net/thermometer) and listen to this communication channel to wait for heating manager control command if necessary. That means the local communication channel and remote communication channel both have two way function to work with in the system. In the prototype project, there is a user friendly user interface to simulate the temperature changes through slide the slides in the control panel of temperature simulator. The working application is shown in the Figure 2 with the user interface slide label 'Tempreture'. Once the sliding the slide to demonstrate the temperature changes in the real work then temperature sensor manager will send message with format in Code



481879c2-5dc4-4ebb-9b12-b05302d2a84c
Heater Status: Not heating...

3489d80b-f2ad-44f2-b4f1-9c32452ab8c5
Heater Status: Not heating...

d5bcde68-11ec-4222-9aa4-f041c08ebf68
Heater Status: Not heating...

1b98fb88-7e7d-49fc-a3b8-f8620600a323
Heater Status: Not heating...

494b0c4b-b86c-4822-b90d-db57b66c0db4
Tempreture : 0

8c57e4b2-20d2-4bdb-a7c5-050baf7bf368
Tempreture : 0

Available bundles. These can be dragged to the Run Require

- hydna-ntnu-student-bundle.hydna_bundle [Workspa
- aQute.libg [Bndtools Hub]
- aQute.xray.plugin [Bndtools Hub]
- biz.aQute.bnd [Bndtools Hub]
- biz.aQute.bnd.annotation [Bndtools Hub]
- biz.aQute.bndlib [Bndtools Hub]
- biz.aQute.junit [Bndtools Hub]
- biz.aQute.launcher [Bndtools Hub]
- ch.qos.logback.classic [Bndtools Hub]
- ch.qos.logback.core [Bndtools Hub]
- com.google.guava [Bndtools Hub]
- com.googlecode.json-simple [Bndtools Hub]
- javax.inject [Bndtools Hub]
- javax.servlet [Bndtools Hub]

Figure 2: Smart Heating System Sensor Simulators

Snippet1. Then the central heating manager will decide if it need to send message to notify binding heater to turn on or turn off. In this whole process all the device clients have only one basic functionality based on what kind of device they are. This architecture of the system make other service could be possible to reuse it by just changing different OSGi service bundle in the system, which is the key point of our project.

Code Snippet 1 Hydna Message Format

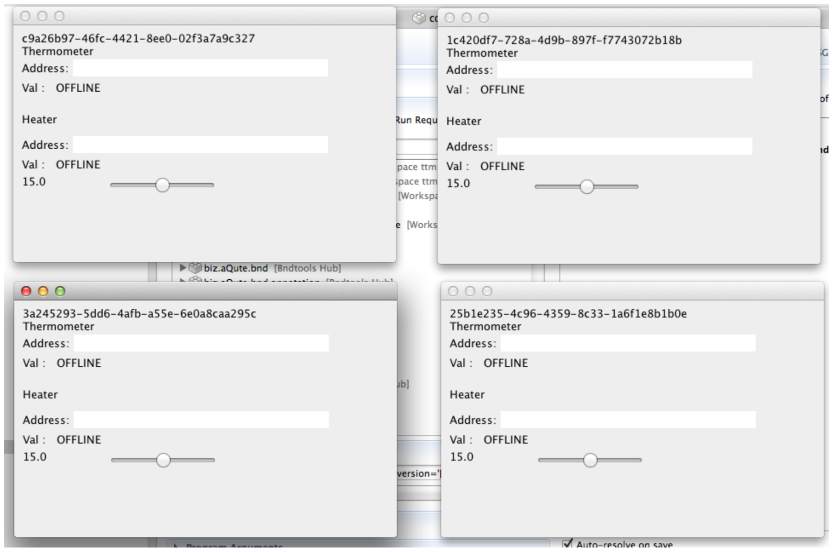
```
<hydna message type> : <message source> : <message destination> : <message command>
```

2.2 Heater Manager

Heater Manager is a similar service as Temperature Sensor Manager to provide bridge between Heating Manager and heater devices. In our prototype project, it is included in Heater Simulator OSGi bundle as well. The running application of it is shown in the Figure 2 with the user interface label 'Heater Status'. The different between Heater Manager service and Temperature Sensor Manager service is that Heater Manager need to listen to another Hydna remote communication channel(channel domain: ithouse.hydna.net/heater) to get the correct control command to responsible heater device. Once it get the control command from the Heating Manager then it will understand what kind of command it is then make the corresponding heater device working correctly according to the command. For the goal of simplifying the project message in the Hydna, in this project we are using the same message format in Code Snippet 1, only with the different hyena message type and message command content. The code to provide the function of processing message got from Heating Manager is shown in the Code Snippet 2. The function works based on not only the control command but also the current status of the corresponding heater device. Then in this system, again the manager service takes care of the 'dump' device clients and device clients only care about basic function of them. This mechanism make the boundaries between the different service component in the system, that's the key part of the self-component system to separate the different services in different component to reduce the dependence of each other. By OSGi, we can easily install and upgrade some service bundle in the system later to make some improvement of the system or replace with other service.

2.3 Heating Manager

Heating Manager is the core of the whole system. It provides the logic to bind the different temperature sensor and heater device as an useful working service for the user and also provide the control logic by the message from temperature sensors and heater device clients. In our prototype project, there is an user interface as well which is shown in Figure 3. There are four initial services user interface allocated for the purpose of demonstration. By typing the UUID in the text box in the user interface, then the corresponding heater and temperature sensor will be bind as a user heating service. There are a threshold slides bar in the service which can be controlled by the user to set the ideally temperature of the room or house. Then the Heating Manager will do the working process according to the system algorithm which is shown in Figure 4, it will trigger some control message if necessary then send it to the Hydna remote communication channel(if control message is for heater, most case, the channel domain will be: ithouse.hydna.net/heater) with the same message format in Code Snippet 1. Then the Heater Manager will get the message to do the command to corresponding heater device. Since the Heating Manager need to observe the temperature data in 'thermometer'(channel domain: ithouse.hydna.net/thermometer) channel and heater manager connection status in 'heater' (channel domain: it-



```
c9a26b97-46fc-4421-8ee0-02f3a7a9c327
Thermometer
Address:
Val :   OFFLINE

Heater
Address:
Val :   OFFLINE
15.0
```

```
1c420df7-728a-4d9b-897f-f7743072b18b
Thermometer
Address:
Val :   OFFLINE

Heater
Address:
Val :   OFFLINE
15.0
```

```
3a245293-5dd6-4afb-a55e-6e0a8caa295c
Thermometer
Address:
Val :   OFFLINE

Heater
Address:
Val :   OFFLINE
15.0
```

```
25b1e235-4c96-4359-8c33-1a6f1e8b1b0e
Thermometer
Address:
Val :   OFFLINE

Heater
Address:
Val :   OFFLINE
15.0
```

Figure 3: Smart Heating System Heating Manager GUI

house.hydna.net/heater)channel, then we implemented these listener in different thread to avoid the conflict between different message event in different channel. But at first there is no getinstance function in provided hydna-ntnu-student-bundle[6], then we make our own generateNewApiInstance method in the hydna-ntnu-student-bundle. It provides resources to run the api service in different Java thread, it is a fast way to get the multiple thread solution but maybe not the best way.

Code Snippet 2 processMessage function in Heater Manager

```
  // Message  type:src:dst:cmd
private void processMessage(String message){
String[] strArray = {};
strArray = message.split(":");
if(strArray.length<4){
return;
}else{
String msgType = strArray[0];
String msgSrc = strArray[1];
String msgDst = strArray[2];
String msgCmd = strArray[3];
// trying to process message
if(msgType.equals(Heater.SERVER_MESSAGE)){
//find address in array
HeaterSimulatorItem hsi = heaterMap.get(msgDst);
if(hsi!=null){
if(msgCmd.equals(Heater.CMD_STATUS)){
hydnaSvc.sendMessage(Heater.CLIENT_MESSAGE
+ ":" + msgDst + ":"
+ msgSrc + ":"
+ Heater.CMD_STATUS
+ "@" + hsi.getHeater().getStatus());
}else if(msgCmd.equals(Heater.CMD_HEATER_ON)){
hsi.getGUI().startHeating();
hsi.getHeater().setHeat(true);
hydnaSvc.sendMessage(Heater.CLIENT_MESSAGE
+ ":" + msgDst + ":"
+ msgSrc + ":"
+ Heater.CMD_STATUS
+ "@" + hsi.getHeater().getStatus());
}else if(msgCmd.equals(Heater.CMD_HEATER_OFF)){
hsi.getGUI().stopHeating();
hsi.getHeater().setHeat(false);
hydnaSvc.sendMessage(Heater.CLIENT_MESSAGE + ":"
+ msgDst + ":"
+ msgSrc + ":"
+ Heater.CMD_STATUS
+ "@" + hsi.getHeater().getStatus());
}
}
}else if(msgType.equals(Heater.CLIENT_MESSAGE)){
//System.out.println(Heater.CLIENT_MESSAGE
+"(Heater Simulator)" +"-> "
+msgType+":"
+msgSrc+":"
+msgDst+":"
+msgCmd);
}
}

}
```

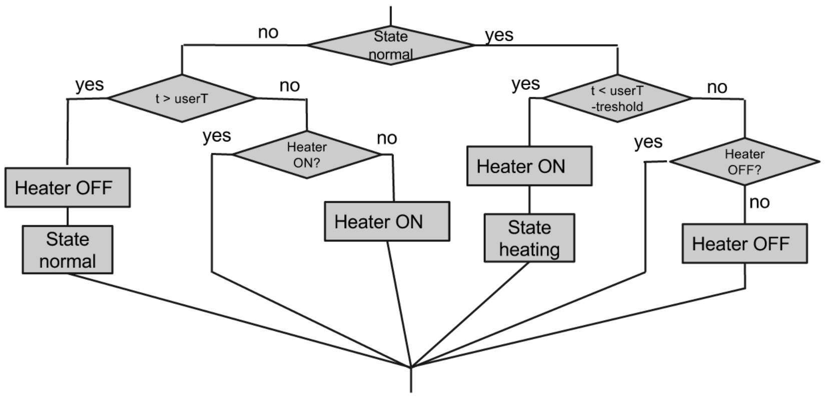


Figure 4: Smart Heating System Heating Manager Algorithm

In the system algorithm shown in Figure 4, it showed that there are two states for each heater, one is state normal, the other is state heating. The state machine implement method helps this system to be simple and flexible with other services later as well. In the Heating Manager service bundle, it just needs to ask for the current state of the heater then according to the message of the temperature sensor and user configured threshold to send control command to the Heater Manager. Heating Manager just need to make sure if the communication between itself and remote communication channel working well. When the Heater Manager is offline for some reason, then the binding service will be trashed by Heating Manager, the only notification is from remote Hydna 'heater' channel.

Figure 5: Spheramid Gateway

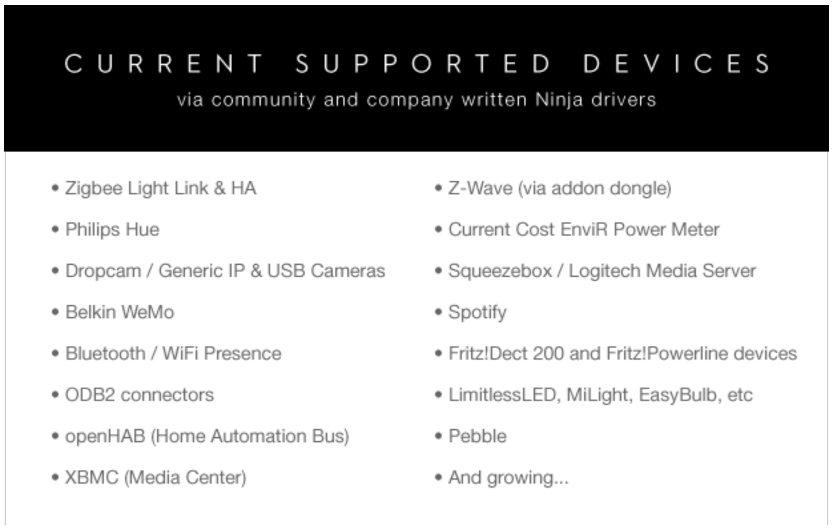
3 Related Work

In the section 2, it explain the prototype system of this report. The concept is to use three bundles to communicate through local communication channel or remote communication channel to build the self-adaptive system to control the house heating system. There are several similar concept consuming product on the market, they will be discussed and compared with our self-adaptive heating system in this section.

3.1 Ninja Sphere

Ninja Sphere[2] is a kickstarter[8] project just currently founded on the kickstarter website. It is a next generation control of the environment with accurate in-home location data and a gesture control interface. Ninja Sphere learns about the user and user environment. It uses data from sensors and actuators to build a model that can inform user if something is out of place. It can monitor temperature,lighting,energy usage, user and even user pets' presence, and anything else user connect to their sphere. By using data from user's devices,environment and location, ninja sphere is able to advise you intelligently and give user control only when user need it. Like our prototype project, the devices connected with Ninja Sphere is 'dump' device, there are no directly communication between these 'dump' device, every device only care about their own responsible function. The developer of the Ninja Sphere has released the supported devices for the system to connect with in Figure6.

And according to Ninja Sphere system architecture in Figure 7, the main control component in the system is Spheramid Gateway5, it is the main service component to be used for other connected devices communicate with. In our



CURRENT SUPPORTED DEVICES

via community and company written Ninja drivers

- Zigbee Light Link & HA
- Philips Hue
- Dropcam / Generic IP & USB Cameras
- Belkin WeMo
- Bluetooth / WiFi Presence
- ODB2 connectors
- openHAB (Home Automation Bus)
- XBMC (Media Center)
- Z-Wave (via addon dongle)
- Current Cost EnviR Power Meter
- Squeezebox / Logitech Media Server
- Spotify
- Fritz!Dect 200 and Fritz!Powerline devices
- LimitlessLED, MiLight, EasyBulb, etc
- Pebble
- And growing...

Figure 6: Ninja Sphere Current Supported Devices

project system, it will the same as heater binding to bind one heater device and one temperature sensor as a service block in the system. Although in the prototype of this report is only based on internet communication not like the Spheramid Gateway has Wifi, Bluetooth Classic & Low Energy, ZigBee and USB communication method, the basic concept is the same to be as a main communication bridge to cooperate with other component in the system.

For Ninja Sphere system, the user can user their smart phone or wearable smart device (like Google Glass or smart watch) to control and get notification from other daily using devices in the house. For example, user Adam left home and forget to turn off the heater, then the Ninja Sphere system will make the decision send the notification to Adam's smart device to ask user if they forgot to turn off the heater. Once Adam think he forgot to turn off the heater, then he can use the smart device to send turn off command to the system, after that the system will send the command to turn off the running heater in the house. And in current project of this report, the user case is almost the same except the notification and cooperate with smart device function is not implemented in current phase.

In the system of this report, the system has some configuration value from the user which can be changed by the user as well, then the other 'dump' component in the system will do the self-adaptive process when they reach the certain threshold value. The basic concept is the same as Ninja Sphere system.

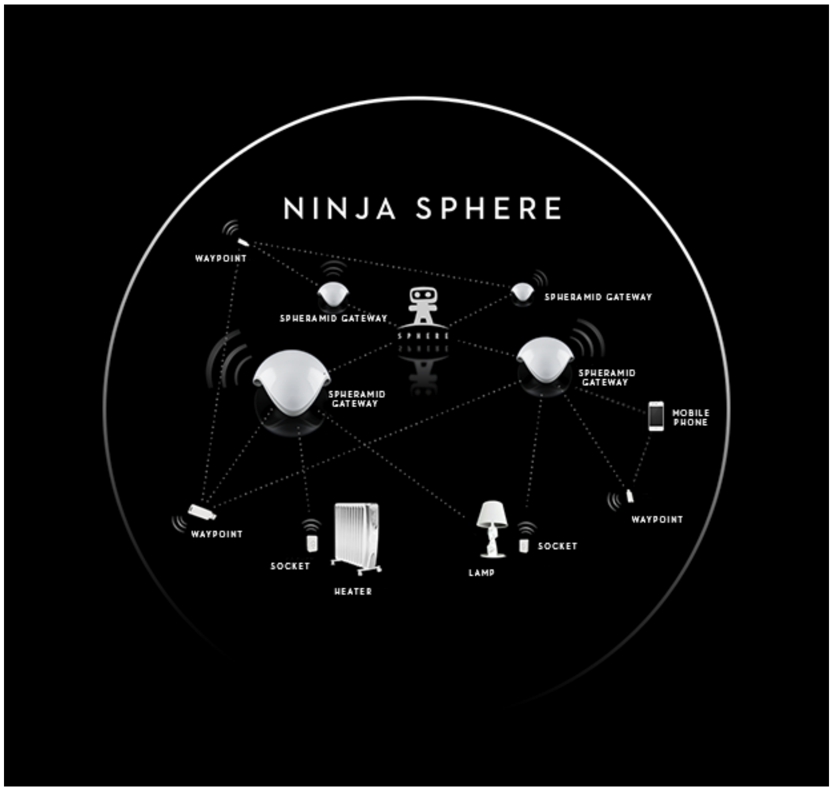


Figure 7: Ninja Sphere System Architecture

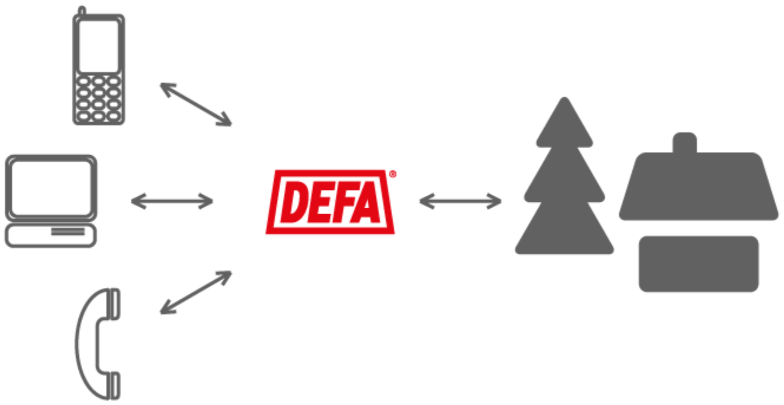


Figure 8: HyttaMi System Architecture

3.2 HyttaMi

HyttaMi[5] service is a consuming product developed by DEFA AS[4]. Hyttami is a subscription based service including GSM subscription for product costs associated with the use of the service , and it provides the easiest and most effective way for remote control and monitoring of electricity, heating and alarm in the cabin.

HyttaMi gives the user a safe and secure solution with full control of heating and alarms in the house . It is very easy to control and monitor both heat , temperature and alarm via the internet , telephone and sms.

The main function HyttaMi has are:

- Turn on or off the heat using a text message
- Check the status of the cabin using a text message
- Turn on or off the heat using the internet
- Check the status of the cabin using the internet
- Turn on or off the heat using a normal phone

According to the system architecture in Figure 8, the working process is also a self-adaptive and user controlling system. Although HyttaMi provides more services than heating control system, other services compare to heating control system has the same basic working logic which is also the same as the prototype system in this report.

The normal user case of HyttaMi could be like, user Eva wants to go to her cabin this weekend, but she still needs to work in the weekday, so originally there is no way for her to make the heating system working in her cabin unless she goes there. But now by using Hyttami, she can just use her phone to send some sms message to the correct number, then the system will turn on the heating system and wait her coming to enjoy the cabin weekend. This working process is also implement in the system of this report. Once user binds the heater device and temperature sensor together, it initially start the heating logic, then the temperature sensor would send current temperature value to the control component, according to certain threshold user set before, the heating would be working to reach the temperature threshold. Moreover since the heater binder, heater device and temperature sensor are not communicate directly, they all communicate with the Hydna service, then each bundle can be initiated and allocated resources in different places. So the remote control process is also possible in the current prototype system.

3.3 Feedback from Related Work

After researching about the related work in the consumer market, it is quite interesting to notice that there are quite a few product have the same working process and concept as the system of this report. And since the development of the smart devices, more and more the system provide the service which mainly communicated with smart device component in the reality.

Because of the time limitation of the system in this report, the potential function of this kind system which are communicated with smart devices has not been implemented yet. But the work flow to implement this important function will not be so hard in the future work of this system since currently all the communication are based on the internet and the main advantage of the smart device is to keep the user on the internet as long as possible.

It is really promising feature if the cooperation of smart devices are done for this project.

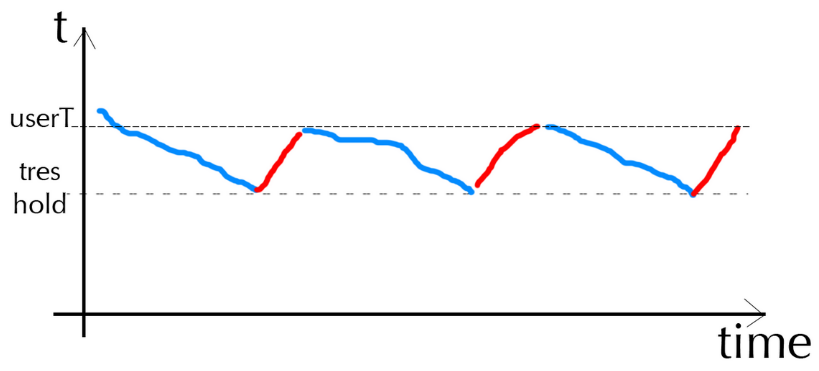


Figure 9: Smart Heating System working Temperature and Time diagram

4 Conclusion

The main purpose to make a self-adaptive system for house heating system is to reduce the energy cost in the residential area. With smart heating system, it can use a big amount of energy because it will not keep heater always running and also every heater run and stop is based on the big scale space temperature not like current most heater is based on the temperature itself. By using our system, ideally the temperature and time corresponding diagram should be like the Figure 9. The red part of the cruel in the diagram is the heater heating process, it is quite less compared to the whole cruel in the diagram, that means that the use of the heater is quite efficient in the smart heating system.

By using OSGi framework, it is quite hard to use at the beginning for us. Although the framework is based on Java, but the building process is not like normal Java application, we need to understand how the bundles work with each other and which part of the system should be service, and which part of the system should be separate bundle. These cases normally will not be considered that much in normal Java application. And also the OSGi framework has not really good IDE (Integrated Development Environment) Editors to work with, it makes our working process quite slow at the beginning when we need to set up a lot of configuration and import different bundles from external project. Although Eclipe is the IDE Editor we are using in this project, it has quite a few annoying problems. And it seems like there is no other optional IDE can be chosen in this project OSGi framework working environment. Then it makes it is quite hard to collaborate with teammates even we are using Git as version control tool because of the build path configuration and Bndtool configuration.

Last but no least, we believe the concept of the OSGi framework would help Java programming more efficient and more clear. And the concept of self-adaptive system is really good to work in telecommunication real-time system which is highly required for self-adaptive feature.

References

[1] Hydna AB. *Hydna*. 2013. URL: https://www.hydna.com/ (visited on 12/01/2013).

[2] Ninja Blocks. *NINJA SPHERE: Next Generation Control of Your Environment*. 2013. URL: http://www.kickstarter.com/projects/ninja/ninja-sphere-next-generation-control-of-your-envir?ref=users (visited on 12/01/2013).

[3] ADRIAN COLYER. *Why should I care about OSGi anyway?* 2008. URL: http://spring.io/blog/2008/05/15/why-should-i-care-about-osgi-anyway/ (visited on 12/01/2013).

[4] DEFA. *DEFA offical website*. 2013. URL: http://www.defa.com/no/fp/ (visited on 12/01/2013).

[5] DEFA. *HyttaMi offical website*. 2013. URL: http://www.hyttami.no/om_hyttami (visited on 12/01/2013).

[6] Snorre Lothar von Gohren Edwin. *hydna-ntnu-student-bundle*. 2013. URL: https://github.com/Snorlock/hydna_osgi/tree/master/hydna-ntnu-student-bundle (visited on 12/01/2013).

[7] Wikipedia. *OSGI*. 2013. URL: http://en.wikipedia.org/wiki/OSGi (visited on 12/01/2013).

[8] Wikipedia. *Wikipedia–Kickstarter*. 2013. URL: http://en.wikipedia.org/wiki/Kickstarter (visited on 12/01/2013).