Lappeenranta University of Technology

Stay warm save cash

Bartosz Rzepkowski
Omar Mater
Shaghayegh Royaee
Hanting Zhang

Sustainability CodeCamp Spring -2016

Table of contents

1. Context

The main objective of our project is to automate the heating system of a standard household in order to reduce the energy consumption and waste. The code camp project on sustainability is a perfect opportunity to test and implement a domotic system.
Based on a FHEM server hosted on a Raspberry Pi Model B, we chose for our project some FS20 devices to implement a scenario that will be described below.

Our main focus is the heating of a typical household (regular or electric heating), regulated by a central thermostat. The temperature should be adjustable from either the thermostat or a web/mobile app. The base of our system is an adaptive time schedule, firstly predefined by the user that will adjust itself based on the user behavior (duration of stay for example). Disruptive events will act as overriding exceptions in the schedule, thus regulating the heating and preventing energy wasting. For example, if a door or window is left opened, there is no need for keeping the regular heating temperature.

2. Scenario

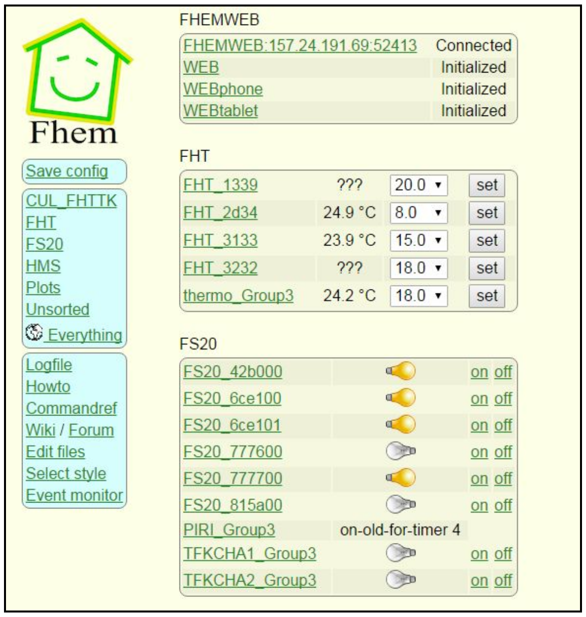
Our scenario goes as follows:
Firstly, a schedule should be defined. The first schedule is defined by the user. The prior can define when he needs the temperature to be warm or chillier (at home, or sleeping) and when heating is unnecessary (at work or away from town).
An example:

> At night, turn temperature down. During the day, if away from home, turn heater off. Else, turn heater up.

This schedule will be time adjustable by the user using either the thermostat or the web/mobile app. Also, this schedule can change depending on the day of the week.
Then, in our scenario, 2 events are susceptible to modify or override the schedule. A sensor for detecting opening doors and windows sends a signal to indicate that the heating should be turned off if doors and windows are opened for too long. A movement sensor is also used to detect if there is anyone in the house. If there is no movement detected in at least 48h hours, the heater should be turned off.
Moreover, our application will display Energy usage statistics and comparative Energy saving with a manual handling of the heater. We could also imagine a comparative graph for each week or month, showing the evolution of the Energy consumption.
Lastly, though we will not implement it during the code camp intensive week, our system could detect the proximity of the user and turn the heat up if the user is approaching in a certain radius.
Experience

3. Hands on FHEM

At first, it might be a good idea to set the auto-create function to enabled. This function lets the FHEM server to define a device every time it encounters a new one on its logs. That way, it is possible to detect and manipulate any FS20 device in the reach of the transceiver.



FHEMWEB

FHEMWEB:157.24.191.69:52413	Connected
WEB	Initialized
WEBphone	Initialized
WEBtablet	Initialized

FHT

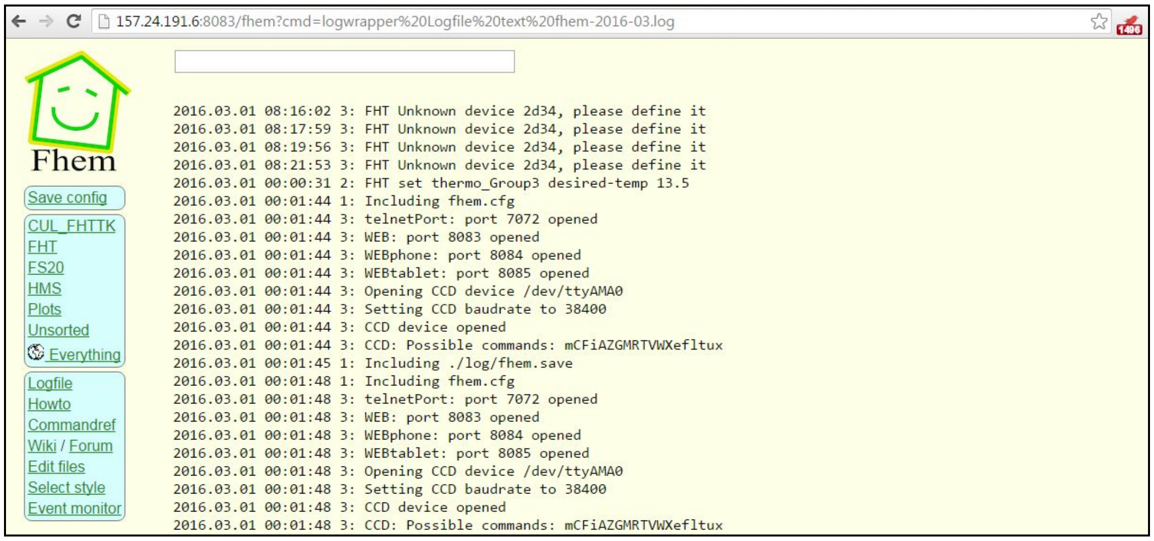
FHT_1339	???	20.0	set
FHT_2d34	24.9 °C	8.0	set
FHT_3133	23.9 °C	15.0	set
FHT_3232	???	18.0	set
thermo_Group3	24.2 °C	18.0	set

FS20

FS20_42b000	●	on off
FS20_6ce100	●	on off
FS20_6ce101	●	on off
FS20_777600	○	on off
FS20_777700	●	on off
FS20_815a00	○	on off
PIRI_Group3	on-old-for-timer 4	
TFKCHA1_Group3	○	on off
TFKCHA2_Group3	○	on off

Fig. 1: Web Page showing all the devices defined by FHEM

To define manually a device, we need its code. By default, the fhem server logs every signal it gets.



```
2016.03.01 08:16:02 3: FHT Unknown device 2d34, please define it
2016.03.01 08:17:59 3: FHT Unknown device 2d34, please define it
2016.03.01 08:19:56 3: FHT Unknown device 2d34, please define it
2016.03.01 08:21:53 3: FHT Unknown device 2d34, please define it
2016.03.01 00:00:31 2: FHT set thermo_Group3 desired-temp 13.5
2016.03.01 00:01:44 1: Including fhem.cfg
2016.03.01 00:01:44 3: telnetPort: port 7072 opened
2016.03.01 00:01:44 3: WEB: port 8083 opened
2016.03.01 00:01:44 3: WEBphone: port 8084 opened
2016.03.01 00:01:44 3: WEBtablet: port 8085 opened
2016.03.01 00:01:44 3: Opening CCD device /dev/ttyAMA0
2016.03.01 00:01:44 3: Setting CCD baudrate to 38400
2016.03.01 00:01:44 3: CCD device opened
2016.03.01 00:01:44 3: CCD: Possible commands: mCFiAZGMRTVWXefltux
2016.03.01 00:01:45 1: Including ./log/fhem.save
2016.03.01 00:01:48 1: Including fhem.cfg
2016.03.01 00:01:48 3: telnetPort: port 7072 opened
2016.03.01 00:01:48 3: WEB: port 8083 opened
2016.03.01 00:01:48 3: WEBphone: port 8084 opened
2016.03.01 00:01:48 3: WEBtablet: port 8085 opened
2016.03.01 00:01:48 3: Opening CCD device /dev/ttyAMA0
2016.03.01 00:01:48 3: Setting CCD baudrate to 38400
2016.03.01 00:01:48 3: CCD device opened
2016.03.01 00:01:48 3: CCD: Possible commands: mCFiAZGMRTVWXefltux
```

Fig.2: Textual logs of the FHEM server, logging every event on the FHEM server

Then the house code of the device can be find using the timestamps of the logs.
It is thus possible to define the device by command line or in the fhem.cfg file as follows:

```
define thermo_Group3 FHT 5012
# Defines a device for the FHEM server
# The attributes are the type of device and its house code defined in the factory settings
# Or by the user
attr thermo_Group3 model fht80b
# This line defines the specific type of device model, here a thermostat FHT80B
attr thermo_Group3 room FHT
# This line defines a logical “room” for the device
# Rooms can make it easier to regroup devices in a system
define FileLog_thermo_Group3 FileLog ./log/thermo_Group3-%Y.log thermo_Group3
# Defines a log file and its path on the server
attr FileLog_thermo_Group3 logtype fht:Temp/Act,text
# Defines the type of the file log: text or just values of Temperature and Actuator
define weblink_thermo_Group3 weblink fileplot FileLog_thermo_Group3:fht:CURRENT
# Defines a file for the plot
attr weblink_thermo_Group3 label "thermo_Group3 Min $data{min1}, Max $data{max1},
Last $data{currval1}"
# Determines the values needed for the plot tracing
```

Fig.3: Configuration for the FHT 80b Thermostat on fhem.cfg

Hence, defining devices on our FHEM server leads us to 3 devices registered and paired with our transceiver:

thermo_Group3: our FHT80b thermostat
PIRI_Group3: our PIRI2 motion detector
TFKCHA1_Group3: our opening window sensor

Using FHEM, it is possible to link devices via events. For example, every time a window is opened, the temperature of thermostat should go down. Using the “notify” command, it is possible to set conditions and causalities.
Let’s look back at our example:
Every time the server sees the state of the TFKCHA1_Group3 switching to ON, this means that the window is opened. Then, the desired temperature (variable desired-temp on FHEM) should go down to 18°C.
For this, it is possible via the fhem command line to set a notify function as follows:

define mvtNotify notify PIRI_Group3:on. set thermo_Group3 desired-temp 18*

4. Objective

Regulate automatically the heat, with an adaptive schedule, and auto regulating heating system parameters:

isolation => windows or doors opened
time schedule => learning curve of your habits (home, work etc.)
presence => inside or in the proximity of your home

5. Tools used

FHEM Server on Raspberry PI B:

Although not necessary, the Raspberry PI is a good candidate to host the FHEM server. Because of its size and low energy consumption (less than 100Wh), the Raspberry PI can also be utilized for other uses in the same home. Also, the FHEM server can handle FS20 devices as well as other devices using other technologies.

Window and door movement sensor: FS20 TFK:

Working by measuring the magnetic field of a magnet as a switching condition (opened/closed), the FHEM server as a notify condition stating that if the windows are opened, the temperature should be reduced or the heater closed.

Thermostat and Heater regulator: FS20 fht80b:

In our prototype, the thermostat is one of the centerpieces that allows regulation of the temperature. It already has a scheduling system that can be configured via the FHEM server

Movement Sensor, FS20 PIRI2:

Sending a signal to the server every 8 seconds if a movement is detected, this sensor allows us to detect if someone is at home during a time range where there should be no one, so that the heating can regulate itself to reach a more comfortable temperature.

6. Limitations during the Code Camp week

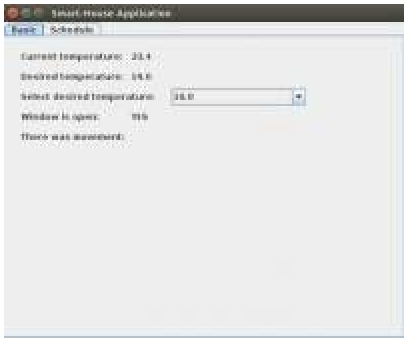
As we are 8 groups in a same room, using roughly the same frequency on our RF transmissions, sending and receiving signals can be long, as signal error and superposition of signals are not uncommon. Thus, for example, setting the desired-temp on our thermostat can take up to 5 minutes. However, as our main focus is heating, change of temperature doesn't happen instantly, a slight delay in the communications is not critical.

7. Java Application

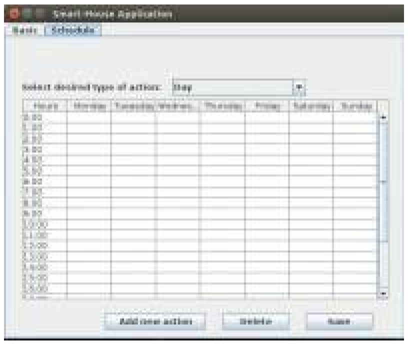
As an interface between the FHEM system and the user, we developed a Java based application retrieving essential values (temperature, status of sensors) from our system and also able to set those values through FHEM. The Dataflow of our domotic system can be interpreted as shown below:

We developed the Java Application to act as a GUI to retrieve information on the system and to act and set parameters. In our prototype, the GUI is composed of 2 tabs.

The “Basic” tab, where informations about the system is shown and where the user can set the temperature of the thermostat:


Smart-House Application
Basic
Schedule
Current temperature: 23.4
Desired temperature: 19.0
Select desired temperature:
Window is open:
There was movement:

The “Scheduled” tab, where the user can define night/day cycles for the temperature of the apartment:



Our JAVA code retrieves information directly from the html web page of the FHEM server. We made the choice of retrieving from the html page rather than from a telnet session and the perl api of the fhem server simply because we were not really familiar with the entire API behind the FHEM server and the WEBFHEM was the tool that we understood most.

Here is the example method (which sets the desired temperature on thermostat) that shows the idea in which our application sets and retrieves information:

```
public void setTemperature(double temperature) throws Exception {
            @SuppressWarnings("resource")
                  WebClient webClient = new WebClient(BrowserVersion.FIREFOX_38);
            // Get the page
            HtmlPage page = webClient.getPage("http://157.24.191.6:8083/fhem?detail=thermo_Group3");
            HtmlSelect select = page.getElementByName("arg.setthermo_Group3");
            List<HtmlOption> options = select.getOptions();
            select.setSelectedAttribute(options.get(2), true);
```

```
        HtmlSelect temperatureSelect = page.getElementByName("val.setthermo_Group3");
        temperatureSelect.setSelectedAttribute(Double.toString(temperature), true;

        HtmlSubmitInput button = page.getElementByName("cmd.setthermo_Group3");
        button.click();
    }
```

Among many classes used by our application we use three, that reflect real devices. Those classes implement the following methods:

```
Thermostat
        |_getDesiredTemperature()
        |_setTemperature(double Temperature)
        |_setScheduleTime(String dayTime, String time)
        |_getMeasuredTemperatures()
        |_getCurrentTemperature()
MovementDetector
        |_checkIfThereWasMovement()
WindowDetector
        |_checkIfWindowIsOpened()
```

Class implementing our GUI creates new thread for movement detector and also for window detector. It checks the value returned by methods from suitable classes every given period of time.

8. Change in Human Behavior: Energy Consumption and Cost

Changing human behavior regarding to the environment can be tricky. Old habits and lack of motivation for change are notably some of many obstacles to changing our day to day actions regarding environment, and in our case, energy consumption. The economy in the past century created the habit of buying and throwing away when the product is broken, or obsolete. Our behavior regarding products and services is fixed because of decades of formatted consumerism, thus making it hard to change.

During this code camp week, we focused our project around the cost of energy. One of the biggest motor to change in a household is usually linked to finances. A hefty bill can and usually is a signal that something has to change, and it often forces the end-user to change its consumption. As we chose the topic of heating a household, our main focus here is to show how much a user can save up by using our system.

For our calculations, we need a reference appartement for comparison: 30 meters square, 2.5 meter in height, normally isolated. Also, the outside temperature is around 10 Degrees Celsius. As a reference for comparison, heating constantly this apartment at 20 degrees takes 1100 Wh. For 16 degrees, it takes 700 Wh.

Here is our sample scheduled for the user:

- From 6-8: temp at 20
- From 8-16: heater off
- From 16-23: temp at 20
- From 23-6: temp at 16

So the total arrives at: 1100*(2+7)+700*7=15kWh for our reference scenario where heat is constantly kept at 20 Degrees Celsius: If the user remembers to turn off the heat during work hours (8-16), when the user is outside, at 20 degrees the total amounts to: 1100*16=18kWh

At a reference cost of 30 cents/ kWh, the savings with an automated system like ours is 90 cents/day, which represents roughly 30 euros / month.

Our system uses different tools that add up to a total of 200 Euros:

- Raspberry pi model B, hosting the FHEM server = 30 Euros
  - http://www.amazon.de/Raspberry-Pi-Mainboard-MicroSD-Speicherkartenslot/dp/B00LPESRUK/ref=sr_1_5?ie=UTF8&qid=1457429114&sr=8-5&keywords=raspberry+pi
- FS20 (868Mhz) transceiver for Raspberry PI = 70 Euros
  - https://www.comprise.de/heimautomatisierung/raspberry/busware-cc1101-transceiver-fuer-raspberry-pi-inkl-antenne/a-45950/
- FS20 TFK window and door movement detector = 30 Euros
  - http://www.elv.de/2-kanal-tuer-fenster-kontakt-fs20-tfk-komplettbausatz.html
- FS20 FHT80b Thermostat and heating regulator = 30 Euros
  - http://www.elv.de/raumregler-fht-80b-arr-bausatz-inkl-batterien.html
- FS20 PIRI2 movement detector = 40 Euros
  - http://www.idealo.de/preisvergleich/OffersOfProduct/3911907_-fs20-piri-2-elv.html

Thus, according to our calculations for energy consumptions, the cost of our prototype can be absorbed in roughly eight months.

Though our prototype might not be cheap, it does represent however a return on investment in less than a year. Automation of systems might not look like a motor to change in behavior at first. However, by saving on the bills, the end-user may slowly realize that making systems more efficient is good for its finance, but also for the environment.

However, though the cost saving is an important aspect, it is essential for such a system to not be forgotten. A well-designed and intuitive interface with the system makes the user want to use the system. Also, the automation aspect of the system requires little effort from the user. Installing the system and then configuring it should be a little a hassle as possible. All these aspects, combined with a system that actually works as intended of course, makes the system pleasant for the user. Moreover, a good experience on one automation system (that saves you money, reduces your energy consumption and your “footprint” on the environment) might entice you to do the same on other systems, i.e: low consumption lighting, greener source of energy.