Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Smart Homes and Weather Events

Bachelor's Thesis

Marc Dominique Hüppin

hueppinm@student.ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Tobias Langner
Prof. Dr. Roger Wattenhofer

January 20, 2013

Acknowledgements

I would like to thank Tobias Langner for his supervision and support, Miguel Rodriguez for taking his time every week to discuss my work with me, the whole team of digitalSTROM for always being availabe for me if I ran into problems or had questions.

Abstract

This thesis discusses the results and research related to the task of building an application for the home automation system of digitalSTROM. The idea of the application is to use weather forecast data by MeteoSuisse in a smart way.

The program maximizes the energy gained through solar radiation on the windows and provides the user with information about the amount. To do that a model was developed to calculate the amount of direct solar radiation through the windows from the given meteorogical data of global radiation on a horizontal plane.

The meteorogical data from MeteoSuisse is read into a MySQL database on a server. On that server a Web service was implemented that handles JSON requests and calculates the amount of direct solar radiation through the windows using the aforementioned model. The response contains information for the current day with an hourly resolution.

An app for digitalSTROM was also developed. It makes daily requests to this Web service and reacts to the information by adjusting the blinds and presenting the results to the user.

To get an idea for results of the calculations there was also a simulation over two months, the results are also displayed and discussed in this thesis.

Keywords: Smart Homes, Weather, digitalSTROM, Solar Radiation

Contents

Chapter 1

Introduction

Energy saving today is more and more important not just for the personal gain of saving money but also on a more general level by saving fuel and helping the planet. There are incredibly many ways to save energy. This thesis focuses on saving energy with the use of meteorogical forecast data. Besides giving us electric power by shining on solar panels, the sun also provides heating energy. A lot of modern houses have large windowfronts often facing south, in winter when the sun is always quite low and shines at a very direct angle onto this windows this can result in heat gain.

My goal was to estimate the amount of this heat gain in advance by using weather forecast data. This makes it possible to adjust the heating in advance and results in saving energy as well as a more comfortable and stable temperature inside a home. In addition to the meteorogical data there's also a necessity for information about the home we want to make this calculation for, namely the geographical location and the sizes of the windows. Besides calculating the results it is also necessary to make sure the blinds are adjusted accordingly, so that the sun is not locked out. For the estimation a model was built to calculate the solar heat gain.

The basic design decision was to create a Web service that will use our model and weather forecast data provided by MeteoSuisse to calculate an accurate estimation of the solar heat gain gained through the windows. Then an app for the digitalSTROM system was implemented that uses the Web service, interprets the results and makes the necessary adjustments to the blinds.

So far I just talked about the situation during the winter time, this is due to the fact that this is much more interesting than the situation during summer. In summer we are not interested in gaining heating energy through the sun but to limit it as much as possible. The idea here is to keep the blinds shut whenever there is sunshine.

In this thesis I will describe my product. Chapter 2 is about related work and the differences to the product of this thesis. Chapter 3 discusses the calculations that were necessary to give a best possible estimate for the energy gain.

Chapters 4 and 5 are about the implementation and functionality of the Web service and the app for the digitalSTROM system as well as a brief introduction to digitalSTROM. At the end I will evaluate my radiation model for a fictional house over two months to estimate the possible energy savings in chapter 6 and give a look into future work in chapter 7.

Chapter 2

Related Work

There are a lot of home automating systems and some of them also work with weather data. Most of these systems choose to get the data from a locally installed weather station. The advantage in that is of course that the data is more accurate than the data obtained through a weather data provider like MeteoSuisse. As an example there is the company Weather Hawk which develops weather stations and suggests their use in home automating systems. They briefly explain the possibilities for home automating systems they thought about on their website[1]. One of their ideas is also to control blinds in response to solar radiation. With a local weatherstation you get live data which again is more accurate than predictions but has the main disadvantage of not being able to react to future events. This makes it impossible to adjust the heating in anticipation of the solar radiation of the day. This can lead to overheating your house in the morning due to not foreseeing the impact of the sun. In general most home automating systems like to react to current weather events, like taking cautionary action when the windspeed is to high. As web research has shown most systems limit themselves to just display weather forecast to the user and let him interpret them.

In addition a solution with a weatherstation costs more, here is a list of products from Weather Hawk[2], the cheapest solutions still cost above $2000.

[1]http://www.weatherhawk.com/home-automation-more
[2]http://www.weatherhawk.com/weather-stations

Chapter 3

Calculations

To calculate the energy gained by solar radiation on windows we built a model. This model can be split up into four sections. First we calculate the position of the sun relatively to our location in section 3.1. Then we have to divide the global radiation into diffuse and direct radiation. This is described in section 3.2. After this we need to calculate the projection of the direct radiation on a horizontal to a vertical plane in section 3.3 and finally we will compute the solar energy gained through specific windows in section 3.4. To illustrate my model I will apply it to a specific situation at the end of each section.

3.1 Position of the Sun

For this part I used a model found in [1]. First of all we want to determine the exact suntime of our location. The term suntime describes the exact local time, meaning at midday the sun is exactly in the south. It is only dependent on the longitude of our place. The equation for the suntime is as follows:

$$suntime = standardtime + 4(\varphi_{local} - \varphi_{st}) + E \tag{3.1}$$

where both time parameters are in minutes and the standardtime is in our case CET (Central European Time). The other parameters are explained here:

$$\begin{aligned} \varphi_{local} &= \text{the longitude of our place} \\ \varphi_{st} &= \text{the longitude of the standard meridian for CET, so 15} \\ 4(\varphi_{local} - \varphi_{st}) &= \text{stands for the constant deviation which is 4 minutes} \\ &\quad\ \text{per degree} \end{aligned}$$

$E =$ time equation, this is necessary because the earth is not moving in a perfect circle
$E = 9.87 \cdot \sin(2 \cdot B) - 7.53 \cdot \cos(B) - 1.5 \cdot \sin(B)$
$B = (n - 81) \cdot \frac{360}{364}$
$n =$ day of the year

With the exact suntime we can now calculate the position of the sun relatively to our location. We describe this position with two angles, β describing the elevation of the sun and ψs representing the azimuth angle, $-180° \leq \psi_s \leq 180°$

$$\sin\beta = \cos\phi_{local} \cdot \cos\delta \cdot \cos\omega + \sin\phi_{local} \cdot \sin\delta \tag{3.2}$$

$$\sin\psi_s = \cos\delta \cdot \frac{\sin\omega}{\cos\beta} \tag{3.3}$$

The three values ϕ_{local}, δ and ω, which are needed to calculate these two angles, are defined as:

$$\begin{aligned}
\phi_{local} &= \text{latitude of our place} \\
\delta &= \text{declination of the sun, at which latitude is the sun in the zenith} \\
\delta &= 23.45° \cdot \sin[(284 + n) \cdot \frac{360}{365}],\ -23.45° \leq \delta \leq 23.45° \\
& \quad n = \text{day of the year} \\
\omega &= \text{hourly angle depending on the suntime, which changes } \frac{0.25°}{min} \\
\omega &= 0.25 \cdot (suntime - 720)
\end{aligned}$$

3.1.1 Example

As an example I will apply these calculations for the hour between 14:00 and 15:00 on November the 19th for a house located in Wädenswil.

At first we calculate the suntime for the standard time of 14:30. To do that we need the three inputs:

$$\begin{aligned}
standardtime &= 870 \\
\varphi_{local} &= 8.667° \\
n &= 324
\end{aligned}$$

Applying eq. (3.1), we get $suntime = 854.5$.

$$\begin{aligned}
\phi_{local} &= 47.2206° \\
\delta &= 23.45° \cdot \sin[(284 + n) \cdot \frac{360}{365}] = -20.24° \\
\omega &= 0.25 \cdot (859.15 - 720) = 33.63°
\end{aligned}$$

Using these values as input for eqs. (3.2) and (3.3) we get the following results:

$$\beta = 16.06^\circ \text{ and } \psi_s = 32.73^\circ$$

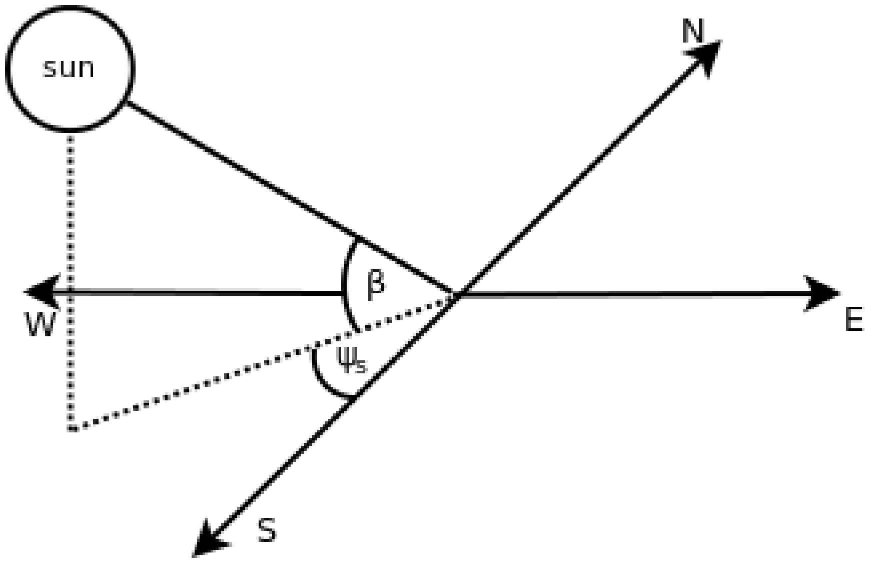


Figure 3.1: The position of the sun relative to a location

3.2 Diffuse and Global Radiation

3.2.1 Different Models

As meteorogical data we get the global shortwave radiation, which is the total amount of solar radiation on a horizontal plane in W/m^2. The global shortwave radiation can be split into two parts: the direct and the diffuse radiation. For the amount of energy we gain, we are only interested in the direct radiation. As mentioned before, we unfortunately only have measurements of the global radiation so we need to extract the direct radiation from that. There are quite a few models which address this issue. I tried to find one that is possible to use with the data available. I found Fabienne Lanini's masters thesis on the subject [2]. It describes several approaches to solve the mentioned problem. Most of these approaches are two complex and often need more input than we can provide. The Reindl* method, which is purely statistical can be used with the data we have at hand. I continued my search and came across the dissertation of Andreas Gassel [3]. He suggested another function to get the ratio of diffuse radiation to global radiation. His model is also applicable to the data available to us.

3.2.2 Reindl*

$$\beta = \text{the solar elevation, like in the previous section} \tag{3.4}$$
$$I = \text{the measured global radiation} \tag{3.5}$$
$$I_d = \text{the diffuse radiation on the horizontal surface} \tag{3.6}$$
$$I_b = \text{the direct radiation on the horizontal surface} \tag{3.7}$$
$$I_0 = \text{extraterrestrial radiation, the solar energy in W/m}^2 \text{ on the atmosphere} \tag{3.8}$$
$$I_0 = 1356.5 + 48.5 \cdot \cos[0,01721 * (n - 15)] \tag{3.9}$$
$$k_t = \text{the clearness factor } 0 \leq k_t \leq 1 \tag{3.10}$$
$$k_t = \frac{I}{I_0 \cos \Theta_z} \tag{3.11}$$
$$\Theta_z = 90° - \beta \text{ it is the zenith angle, angle between the zenith and the sun} \tag{3.12}$$

The clearness factor describes the clearness of the sky. An overcast sky implies a low clearness factor and the clearer the sky the closer we get to one. The Reindl* model is a piecewise regression depending on the clearness factor. An additional condition for the model to show good results is that the elevation of the sun can be no lower than 5° because of the cosine response in eq. (3.11).

$$I_d/I = 1.020 - 0.248k_t \qquad \text{for } k_t \leq 0.3 \tag{3.13}$$
$$I_d/I = 1.400 - 1.749k_t + 0.177\sin(\beta) \qquad \text{for } 0.3 < k_t < 0.78 \tag{3.14}$$
$$I_d/I = 0.147 \qquad \text{for } k_t \geq 0.78 \tag{3.15}$$

The disadvantage is that the function is only continuous within the three parts but not on their boundaries. The first part, eq. (3.13), is for a very low clearness factor this hints to an overcast sky. The ratio here decreases linearly with an increasing clearness factor, because less clouds lead to less diffuse light. For the middle part, eq. (3.14), which represents partly cloudy skies we have again a linear dependancy to k_t but also we have a dependancy on the elevation of the sun. Higher solar elevation contributes to a higher diffuse fraction of the global radiation. For very clear situations with no clouds, eq. (3.15), we have a constant value for the ratio which is 0.147.

3.2.3 A. Gassel

The second model I want to present is found in [3]. His model also is just an optimal approximation of statistical values.

$$\frac{I_d}{I} = 0.53 - 0.34 \cdot \arctan(5.5 \cdot \frac{I}{I_{ex}} - 3.16) \tag{3.16}$$

Here I_{ex} can be calculated using the solar constant I_0 and the adjusted height β_{adj}, which is regarding breaking of the sunlight in the atmosphere.

$$\beta_{adj} = \beta + \frac{1.4705}{3.0427 + \beta} - 0.0158$$
$$I_{ex} = I_0 \cdot \sin \beta_{adj}$$
$$I_0 = 1356.5 + 48.5 \cdot \cos[0, 01721 * (n - 15)]$$

This model is just depending on the solar constant and the suns elevation.

3.2.4 Testing

I got a small testset from MeteoSuisse for a weatherstation in Geneva where we had measured values for both the global and the diffuse radiation. I inserted both data into the models and compared the results with the measured values for diffuse radiation. The Reindl* model showed an overall better performance. The evaluation can be seen in fig. 3.2.

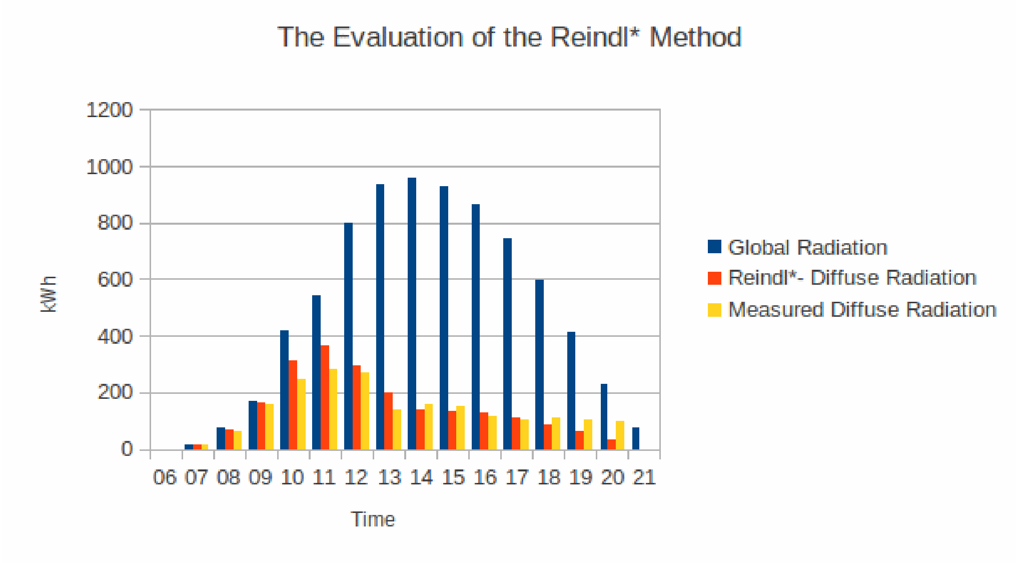


Figure 3.2: The evaluation of the Reindl* model for data from Geneva

3.2.5 Example

We shall apply this next step to our example. From the previous section we have the values $\beta = 16.06°$ and $\psi_s = 32.73°$. To make the next step we need the measured value for the global radiation: $I = 167.6$ W/m^2. Then we also need the extraterrestrial solar radiation, using eq. (3.9) with input $n = 324$ we get 1394.6 W/m^2. Now we can use eq. (3.11) to calculate our clearness factor resulting in 0.43. This means a partially overcast sky and we need eq. (3.14) to calculate $fracII_d$. We get the ratio 0.69 for diffuse radiation which means we have direct radiation of $I_b = 0.31 \cdot I = 52.09$ W/m^2.

3.3 Solar Radiation on Vertical Surface

The next step in our model is to transform the direct radiation on a horizontal plane to a vertical one. This is a purely geometrical transformation. The source for these next computations is [4].

$$I_{b,vert} = \frac{I_b}{\cos \Theta_z} \cdot \cos \Theta_i \tag{3.17}$$

We use I_b for direct radiation on the horizontal plane and $I_{b,vert}$ for the direct radiation on our vertical plane. Θ_i is the solar zenith angle and Θ_i is the incidence angle of the radiation on the vertical surface. It can be calculated with the following equation.

$$\cos \Theta_i = -\sin \delta \cos \phi_{local} \cos \alpha + \cos \delta \sin \phi_{local} \cos \alpha \cos \psi_s + \cos \delta \sin \alpha \sin \psi s \tag{3.18}$$

Where δ is the inclination of the sun, ϕ_{local} the latitude of the location, α the horizontal orientation of our vertical plane, where 0° is south, and ψ_s stands for the horizontal angle of the sun.

3.3.1 Example

We shall now calculate the direct radiation on our windows facing directly south. From the previous sections, we have $I_b = 52.09$ W/m^2, $\beta = 16.06°$ and $\psi_s = 32.73°$.

To apply eq. (3.17) we need to first use eq. (3.18) to find $\cos \Theta_i$. The angles here are $\delta = -20.24°$, $\phi_{local} = 47.2206$, $\alpha = 0$ and ψ_s as shown before. This results in $\cos \Theta_i = 0.81$. Furthermore $\cos \Theta_z = \cos 90° - 16.06° = 0.27$. Now we can apply eq. (3.17) and get $I_{b,vert} = 153.5$ W/m^2.

3.4 Solar Heat Gain Through the Windows

Now that we have the direct solar radiation on our windows, we have to calculate the transmittance of this energy through them. This depends on the window itself and also on the angle between the direct radiation and the window. For this

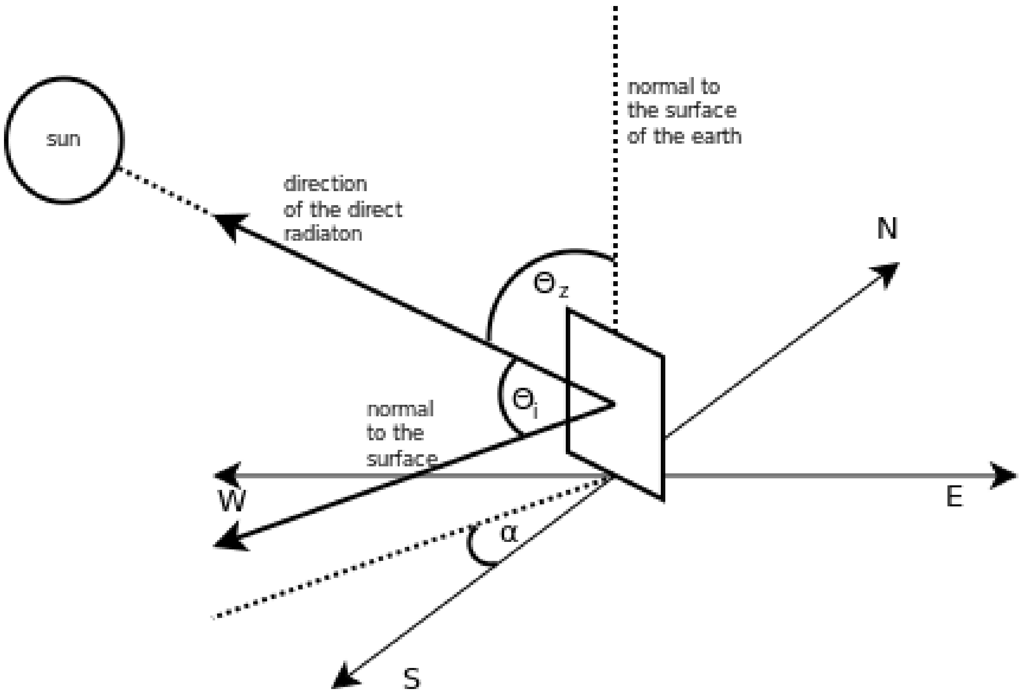


Figure 3.3: The angles for the part of the model in section 3.3

calculation I looked at a model from [5]. They have a polynomial approach for the g-value which represents the transmission rate of solar energy. But because we would need additional parameters that we don't have access to, I chose to go with the simpler model from Montecchi and Polato that is also briefly explained in [5].

$$g = g(0)(1 - \tan^x(\Theta_i/2))$$

Here Θ_i is again the angle of incidence between the window and the direct radiation. And x is a constant depending on the window type. In the paper it is described that this value can assumed to be 4 for most types. $g(0)$ is the g-value (the solar transmittance) of the window if the incident angle is 0. With the online information from[1] I decided to go with the average value of 0.76. This is the average g-value for double glazing windows which are the most common. Therefore the adjusted version is:

$$g = 0.76(1 - \tan^4(\Theta_i/2)) \tag{3.19}$$

[1]http://www.energieberaterkurs.de/export/sites/default/de/Dateien_Kennwerte/kennwerte_fenster.pdf

So in the end if we have the following three parameters for windowsizes *south*, *east* and *west*. We can calculate the energy passing through the south window like this:

$$\varepsilon_{south} = south \cdot 0.76 \cdot (1 - \tan^4(\Theta_i/2)) \cdot \frac{I_b}{\cos \Theta_z} \cdot \cos \Theta_i \tag{3.20}$$

We can do this equally for *east* and *west* resulting in ε_{west} and ε_{east} all three impacts up results in the total amount of energy gained.

$$\varepsilon_{total} = \varepsilon_{south} + \varepsilon_{west} + \varepsilon_{east} \tag{3.21}$$

3.4.1 Example

Now to complete the example we assume our windows facing south have a combined size of 12 m^2. We apply eq. (3.19) to find Θ_i we can apply the arccos function to the value found in section 3.3.

This gives us a total solar heat gain through the south window of 1367.06 Wh or 1.37 kWh in the hour between 14:00 and 15:00 on November the 19th 2012.

Chapter 4

Web Service

4.1 The Meteorological Data from MeteoSuisse

MeteoSuisse provides forecast data in form of an XML file uploaded to an FTP Server. The XML files contain information for several stations. During the development of the project these were Wädenswil, Zurich Fluntern and Zurich Affoltern. They contain the forecast for each station for the next 33 hours. A small java programm is running on the server which checks for new data. If a new file has arrived it is downloaded and the provided data written into a MySQL database. Since the files are not updated regularly every three hours we just run this program every hour. The scheduling is done by crontab.

4.1.1 MySQL Database

There are three simple tables in our database: stations, files and weatherdata.

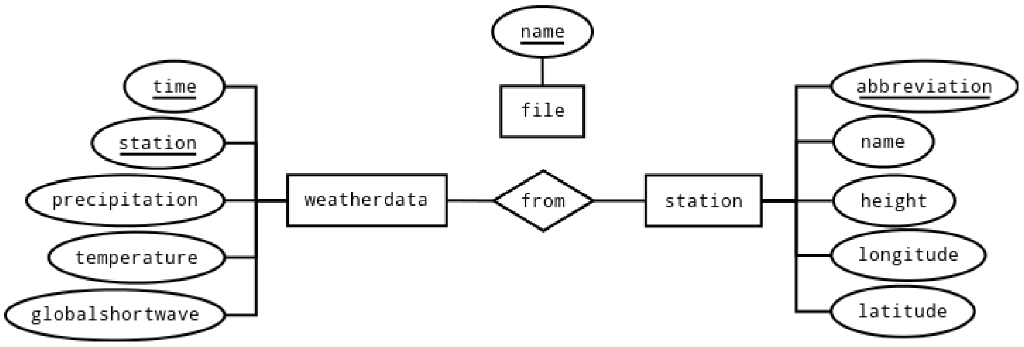


Figure 4.1: Entity Relationship Model

The stations table is mainly needed to determine the closest station to the place where the request is made from. The weatherdata contains information about the weather the time is in UTC. Currently we store precipitation, average temperature and the global radiation for every hour. Precipitation and average temperature are not needed at the moment but are stored for future applications or Web services. The file table contains the name of all the files we have read so far it is kept to check whether we'll have to update the database or if information from all XML files have already been read into the MySQL database. Like previously mentioned we receive a new file about every three hours and each file contains the forecast for the next 33 hours. This leads to multiple data for every hour. Upon receiving new data for an hour we overwrite the previous information. The reasoning behind that is, that the new data should be more accurate.

4.2 The JSON Web Service

The purpose of our Web service is to provide the client with information about the solar heat gain through windows on the current day. Upon receiving a request with the necessary parameters it queries the database for the adequate data and uses the model described in chapter 3 to calculate the solar heat gain. It then returns the results to the client in form of a JSON object.

JSON

JSON (JavaScript Object Notation), is a lightweight data interchange format[1]. A JSON object consists of an unordered set of name5/value pairs. Where the name is a string describing the value. There are also JSON arrays which consist of an ordered set of values. The value can in both cases be a string, a number, a boolean, another JSON object or array or it can be null. An example of a JSON object can be seen in listing 4.1.

4.2.1 Classes

The Web service is based on the Restlet Framework[2] for Java and contains the following four classes.

WeatherApplication

This is the main application, every call to our url first gets here. Inside of the class the routes were defined. To each route belongs a string for example "/weather" and a class that handles the requests for this route. This means, that calls to our url with the substring "/weather" get redirected to the defined class, which in our case is WeatherResource.

[1]http://www.json.org
[2]http://www.restlet.org

WeatherResource

This class contains the method which is started when the URL receives a call. It stores the parameters and initalizes the calculations. After the computations are finished it returns the resulting JSON object to the client.

Calculations

For every call a new instance of Calculations is constructed. Inside of the class we apply the model described in chapter 3 to get the desired values. This class also generates the JSON object that is later returned.

MySQLAccess

MySQLAccess handles the accesses to the database. It receives method calls from the Calculations class, generates the desired query, parses the response and returns the results.

4.2.2 The Execution In Detail

The Web service handles GET requests and returns a JSON object. The request must contain the following parameters:

- longitude
- latitude
- angle
- south
- east
- west

longitude and *latitude* are the coordinates of the home for which the calculations are desired. The other four parameters are all data from the house, while *south*, *east* and *west* refer to the corresponsive windowsizes. The *angle* describes the exact orientation of the house.

We extract the parameters from the query and create an instance of the class Calculations. An instance of Calculations represents the data for a specific home on a specific day. At construction we store the received parameters and then compute values which are valid for the whole day. These are namely the day of the year (n), the declination of the sun (δ), the time equation (E) and the adjusted extraterrestrial radiation (I_0).

As a next step we have to apply hour model to hourly data. To do this we make exactly two queries to the database. The first one returns the weather station from our database that is closest to the location. With the result of the first query we can now get the desired data from the closest station.

We then return an array containing the global radiation for every hour of the day. We iterate over the array, if we meet an hour with a global radiation value that is bigger than 5 we apply our model to calculate the solar heat gain through the windows for the hour. This is done in three steps. We first calculate the sun's position in the middle of the hour, then find out the ratio of diffuse radiation and finally compute the energy gained through the windows.

In addition to that we need to find out whether the sun is actually shining, or not. Sunshine is defined as whenever the difference between global and diffuse radiation is higher than 120 W/m^2, this is defined by the WMO[3]. But as we only get hourly averages we have to set this threshhold much lower. If we assume the sun is shining for 5 minutes and for the other 55 minutes the ratio of global to diffuse radiation is close to one. We get a value of $120 \cdot 5/60 = 10$ W/m^2 direct radiation for the hour. For that reason we chose 10 as a threshhold so the boolean sun is true whenever the direct radiation for the hour is bigger than 10 Wh/m^2. Finally we return a JSON-Object containing the information to the WeatherResource class which in turn returns it to the client.

```
{
  hours : [
      {
  hour : 0,
  energy : 0,
  sun : false
      },
      ...
      {
  hour : 12,
  energy : 4069,
  sun : true
      },
      ...
      {
  hour : 23,
  energy : 0,
  sun : false
      }
  ]
}
```

Listing 4.1: Example of a returned JSON-Object

[3]World Meteorological Organization http://www.wmo.int

Chapter 5

dSS App

5.1 digitalSTROM

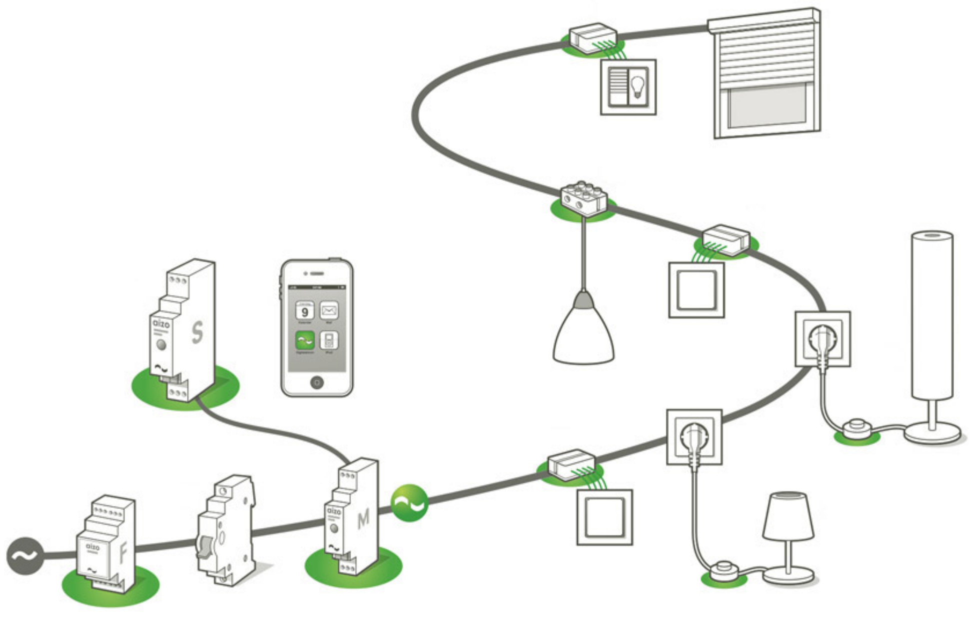
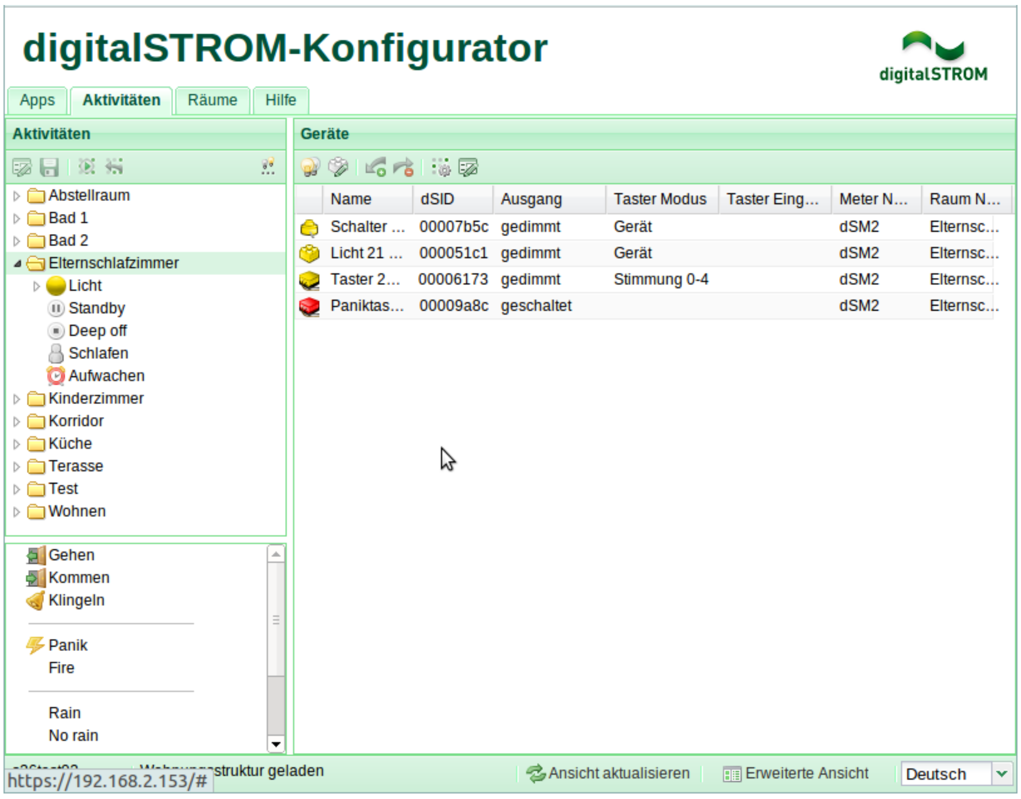


Figure 5.1: Graph of a digitalSTROM system

digitalSTROM is a home automating system developed by aizo. The system connects all devices of a home. The communication between the devices is done through the powerline.

To make it possible for the devices to communicate, you simply attach a clamp to each device. These clamps are miniature computers able to control a variety of devices. There are different clamps for different types of devices. For instance yellow for lights and red for security. We will mostly be concerned with the gray clamps which represent the group for shade and control the blinds and the shutters.

Additionally to the clamps a digitalSTROM system also comes with a digitalSTROM Meter or dSM and a digitalSTROM Server or dSS. The dSM is responsible for the communication between the devices and monitors the consumption. There exists one for every circuit. The dSS connects the dSMs and can communicate by a special protocol. To communicate with other applications it uses TCP/IP.



Name	dSID	Ausgang	Taster Modus	Taster Eing…	Meter N…	Raum N…
Schalter …	00007b5c	gedimmt	Gerät		dSM2	Elternsc…
Licht 21 …	000051c1	gedimmt	Gerät		dSM2	Elternsc…
Taster 2…	00006173	gedimmt	Stimmung 0-4		dSM2	Elternsc…
Paniktas…	00009a8c	geschaltet			dSM2	Elternsc…

Figure 5.2: Screenshot of the dSS WebUI

The dSS gives the user a Web user interface (WebUI) to gain information and change settings in the system. As you can see in fig. 5.2 the devices can be grouped into different rooms. It is possible then to for instance switch off every device in one of these rooms.

In the WebUI you can also define scenes. These scenes are basically presets, a combination of settings for different devices. These scenes can then be called for instance via the WebUI but also by other applications. You could define a scene for when you are watching TV. For instance you want all the lights in your living room to dim to about 10 percent and all blinds to be shut. After defining the scene you can call it whenever you are watching TV. The lights will dim and the blinds will shut.

A way to make this even more comfortable can be achieved by using the so called events. You could raise an event in the digitalSTROM system whenever you turn on the TV and as a reaction to this event you can define the according scene. I will explain this more thoroughly in section 5.2.3.

There is also the tab Apps in the WebUI. Here you can install applications for your digitalSTROM system. For achieving the goal of this thesis, to control the blinds in response of sunshine, such an application was developed.

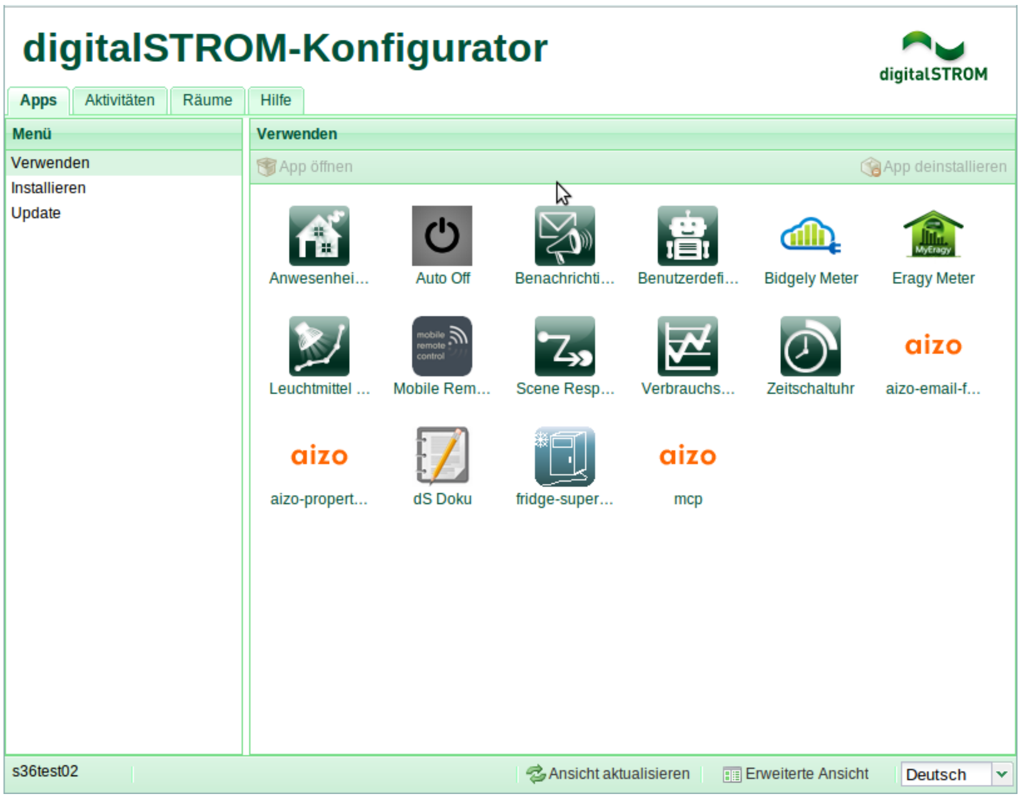


Figure 5.3: Screenshot of the dSS WebUI

5.2 The digitalSTROM App

The dSS app provides two main functionalities. The first is presenting information about the gained energy through solar radiation on the windows during the day in the WebUI of the app and the second one is adjusting the blinds on an hourly basis regarding whether the sun is shining or not. Which blinds are to be shut and which not has to be defined by the user.

5.2.1 Implementation

The app consists of three parts. There are the user interface, the scripts we need to run and the configuration part located in the three folders 'ui', 'scripts' and 'config'. We shall now have a look at these parts in detail.

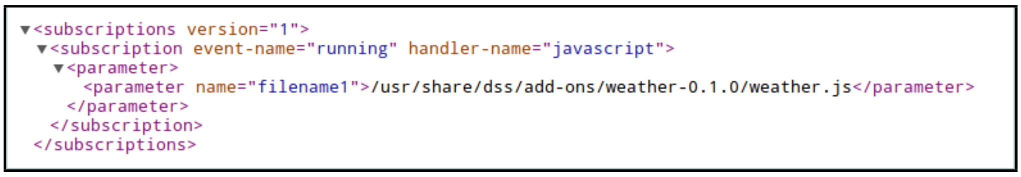
5.2.2 The User Interface

The user interface consists of an HTML page where the user has to enter the sizes of windows, facing east, south and west, and an angle. The angle must be given in degrees, it is the correction angle and describes the exact direction the southern side is facing. If it is facing south exactly the angle is $0°$ (east would be $-90°$). On submitting the form, javascript functions are called. They store the provided information in the property tree of the dSS which is done by JSON calls with jQuery. Below the form on the HTML page is another section where the results are displayed. In detail we display the expected amount of solar energy gained throughout the day. Aside from the total amount, there is also an hourly resolution present.

5.2.3 Events

In order to explain the two other parts of the application I will further introduce the concepts of events. There are a few basic events in the system, like 'running', which is raised when the system is up or 'panic' which is raised when the panic button is pushed. Scripts inside the dSS can create and raise events as well. Three types of events are available. There is the basic event on which occurs instantaneously after being raised. With the TimedEvent you can give a time parameter, that represents the delay in seconds, it occurs with the defined delay after it is raised. Finally there is the TimedICalEvent which occurs at a defined time and date just once or at a rate defined with a reccurence rule[1].

In order to react to these events we need subscriptions.



```
▼<subscriptions version="1">
  ▼<subscription event-name="running" handler-name="javascript">
    ▼<parameter>
       <parameter name="filename1">/usr/share/dss/add-ons/weather-0.1.0/weather.js</parameter>
     </parameter>
   </subscription>
 </subscriptions>
```

Figure 5.4: A static subscription

In fig. 5.4 we see an example of a subscription that makes sure that when the event with the name 'running' is raised we run the weather.js script. In

[1]for a description of the iCalendar format visit http://tools.ietf.org/html/rfc2446

addition to these static subscriptions we can also create dynamic subscriptions from within a script.

5.2.4 Scripts and Configuration

In our scripts folder we have two javascript files. The first one has two main functions, 'init' and 'main'. The 'init' function is called when the system is started. It creates an Event that reoccurs daily at 05:00 in the morning. The values for the home of the user are all initialized to zero if they have not been set before. The 'main' function is the one that is now called every day at five in the morning. Here the JSON call to the Web service is made, we get the energy data for every hour of the current day. Once we have the data, we create new TimedICalEvents for the current day. All of them are raised at the full hour and do either have the name 'sun' or 'nosun'. In addition we store the solar heat gain for the day and for every hour on the dSS.

For both of these events we have the same handler. The script checks whether there will be sunshine or not and then calls the associated scene.

The configuration part is simply an XML file. It specifies which events the two scripts have to handle.

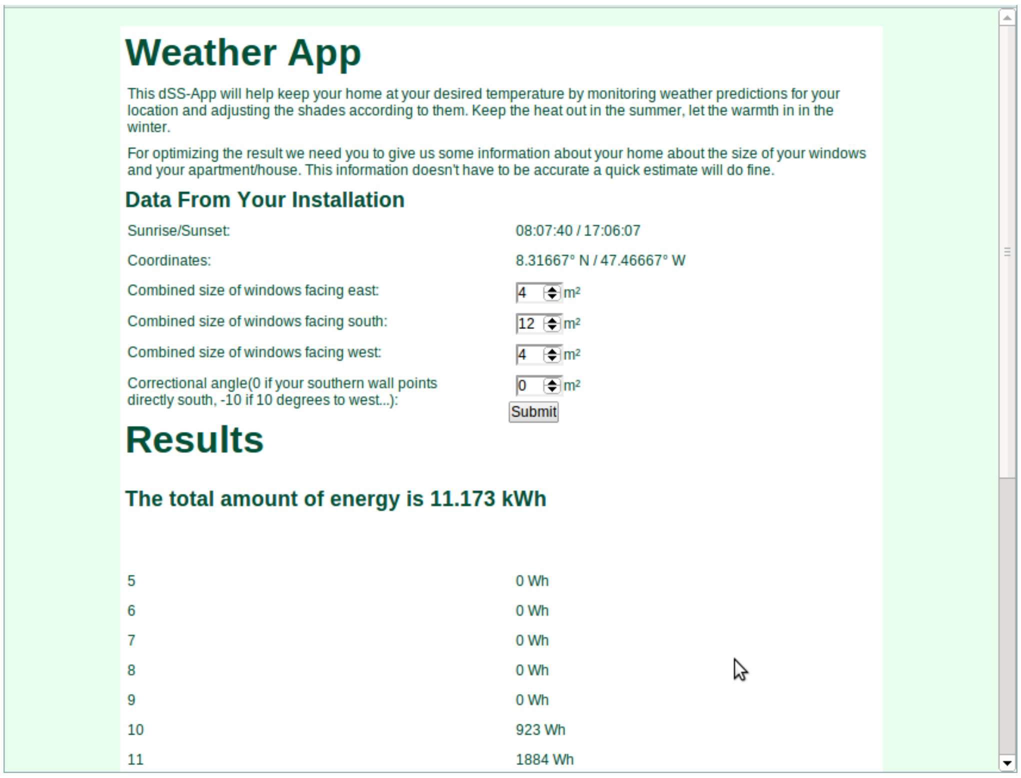
5.2.5 The Web App

To make testing easier, and address people without digitalSTROM, I also decided to create a small website that allows everybody to learn about the amount of solar energy their home will receive during the day[2]. The design of the web interface is very similiar to the one of the dSS app.

The user has to input information about his home just like in the dSS app. After submitting the data we display the total amount of energy that is gained and a list overviewing the hours between sunrise and sunset.

The big drawback here of course is that we currently only receive weather forecasts for three different stations that are all located around Zurich. This of course limits the usefulness of the website, since only people close to those stations actually get accurate data. But even for others it should be able to give a general idea of how much solar energy they can gain on a sunny day.

[2]http://weather.aizo.com/WeatherTesting



Weather App

This dSS-App will help keep your home at your desired temperature by monitoring weather predictions for your location and adjusting the shades according to them. Keep the heat out in the summer, let the warmth in in the winter.

For optimizing the result we need you to give us some information about your home about the size of your windows and your apartment/house. This information doesn't have to be accurate a quick estimate will do fine.

Data From Your Installation

Sunrise/Sunset:	08:07:40 / 17:06:07
Coordinates:	8.31667° N / 47.46667° W
Combined size of windows facing east:	4 m²
Combined size of windows facing south:	12 m²
Combined size of windows facing west:	4 m²
Correctional angle(0 if your southern wall points directly south, -10 if 10 degrees to west...):	0 m²

Submit

Results

The total amount of energy is 11.173 kWh

5	0 Wh
6	0 Wh
7	0 Wh
8	0 Wh
9	0 Wh
10	923 Wh
11	1884 Wh

Figure 5.5: The interface of the dssApp

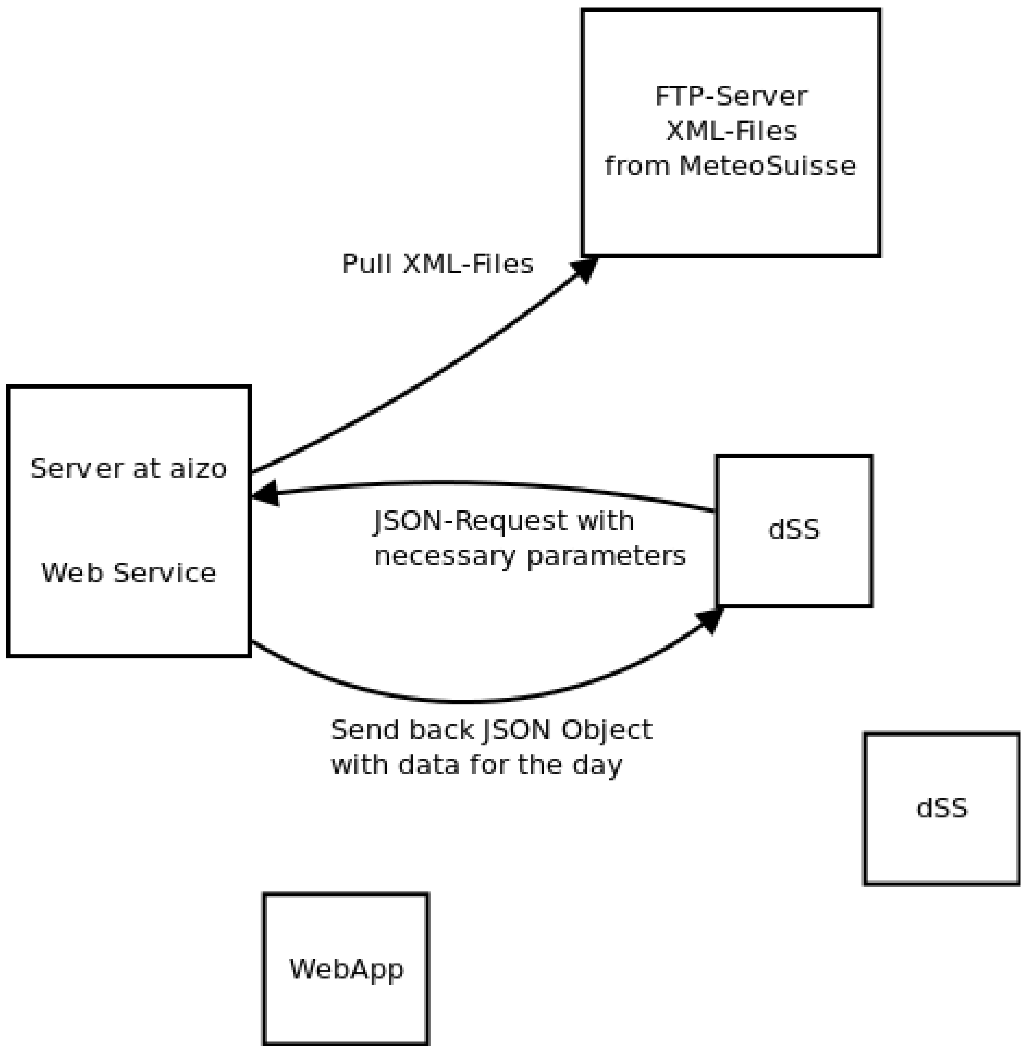


Figure 5.6: The interaction between the different parts of the system, the Web service can be used by multiple dSS apps and the Web app

Chapter 6

Example/Simulation

To have a better idea of the benefits achievable with our program, a simulation over the months of November and December 2012 was done. Our fictitious house is located in Wädenswil. Like a lot of modern houses it has a big windowfront facing south that is 2 meters high and about 6 meters wide. On the eastern and western side there are two windows each with a size of 1 by 2 meters.

$$\begin{aligned} south &= 12 \\ east &= 4 \\ west &= 4 \\ longitude &= 8.6777 \\ latitude &= 47.2206 \end{aligned}$$

We applied our model from chapter 3 for every hour of every day in that timespan and stored the amount of energy.

6.1 Results

The results of the simulation can be seen in figs. 6.1 and 6.2

We gained a total of 502.5 kWh of solar heating energy during these two months. The cost for producing the same amount of heating energy artificially depends on your heating system and can change with fluctuating prices in the commodity market. Because oil-fired heating is still widely used in Switzerland we assume an oil-fired heating running on standard fuel. We calculate the equivalent price for heating our house with 502.5 kWh. Standard fuel has a minimal fuel value of 45.4 MJ/kg or 10.74 kWh/l which corresponds to a heating value of 42.6 MJ/kg or 10.08 kWh/l[1]. So to gain 502,5 kWh we need about 20.9 l of

[1] numbers from http://www.iwo.de/fachwissen/brennstoff/

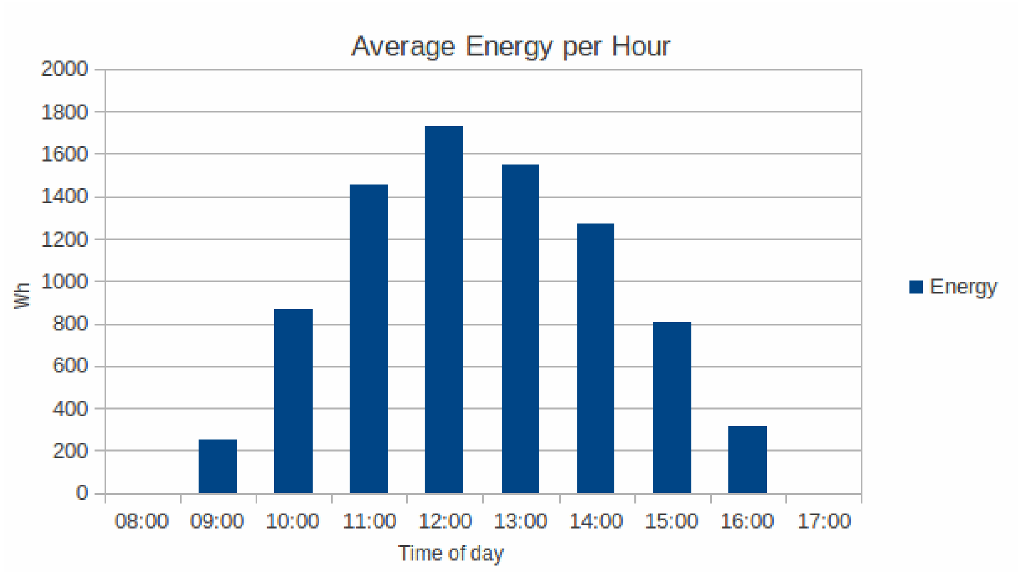


Figure 6.1: The values stand for the hour before the mentioned time, so 12:00 means the span of 11:01-12:00

fuel. Prices for fuel depend on the amount you buy, if we assume a purchase of fuel between 3000 and 6000 l. In November of 2012 we would have paid 102.50 per 100 l. This means our expenses to heat our house with 502.5 kWh would be at about 21.50 CHF[2].

6.1.1 How to Interpret These Results

First of all by not using digitalSTROM and the Weather App you would still gain a big part, or all of the energy, if shades were kept open most of the time during the day. But it still maximizes the use of solar energy even when you're not at home. In addition to this you gain the information of an estimate of solar energy you let in to your home in advance. So you can adjust your heating manually in the morning and take it down a notch before leaving your home in the morning if you're looking at a sunny day. This results in a house at a more comfortable temperature and lowers the fuel usage and therefore your expenses.

We will now have a more specific look at two days. The first day is the 1st of November. It was a day with multiple hours of sunshine, where we gained a maximum of about 20 kWh of solar energy. The second day we will look at is the 12th of November. There was a lot of fog on that day and pretty much no direct radiation reached the surface. Therefore we didn't get any solar energy.

[2]numbers from http://www.bfs.admin.ch/

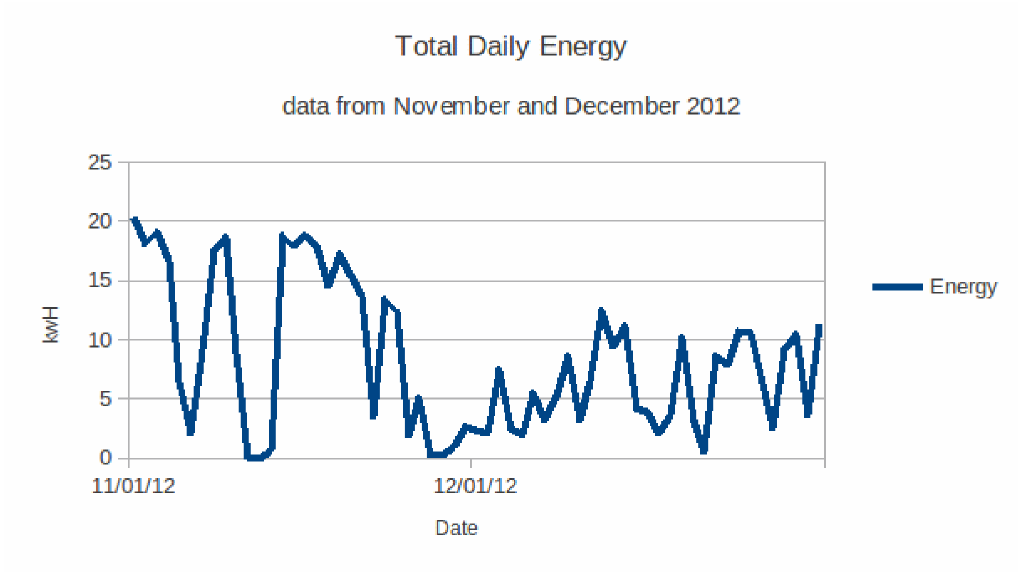


Figure 6.2: Total: 502.5 kWh

In Germany the average need of fuel is about 15.4 liters per m^2 per year[3]. If we assume our house is about 100 m^2, which is about the size of an average four room home, we need about 1540 liters per year. Because the big part of heating fuel is used during the cold months we can assume, that we need about two thirds of the heating energy from November to March. This would mean about 1000 liters over a span of 150 days. Which results in about 6.6 liters per day using the numbers from before this leads to about 66.5 kWh of heating energy per day. So on November the 1st we could have saved almost a third of the heating energy.

This is under the assumption that the useage of oil is equally distributed over these five months, which is of course not true. Still it should give us an idea of how we can use the sun to save energy.

[3]data from http://www.heizungstipp.de/oelheizung/verbrauch

Chapter 7

Future Work

The digitalSTROM project is still very much in development and during thw work on my thesis I was rather limited in using my calculations as all I could control were the shades. Most of the calculations are merely done to provide information for the user.

The goal of digitalSTROM of course is to be able to control more and more devices so at some point it will also be able to control the heating devices of homes. I've tried to make my program adaptable for future projects and it is my believe that you can easily use the Web service to do more than adjust the blinds, once the necessary hardware is present. The data can then be used to adjust the heating in anticipation of the solar heat gain during the day to save energy and keep the home at the desired temperature.

As more systems will be added it will also be necessary to get forecast data from more stations.

Finally more Web services can be implemented for other uses of the weather forecast. An idea here would be to use the forecast for precipitation to control the watering of the garden. But again the watering system is something that is not yet controllable with digitalSTROM.

Bibliography

[1] Drueck, D.I.H.: Manuskript zur vorlesung, solarthermie i. In: Universitaet Stuttgart Institut fuer Thermodynamik und Waermetechnik. (2012)

[2] Lanini, F.: Division of global radiation into direct radiation and diffuse radiation. (2010)

[3] Gassel, A.: Beitraege zur berechnung solarthermischer und exergieeffizienter energiesysteme. (1996)

[4] Coleinne Demain, M.J., Bertrand, C.: Evaluation of different models to estimate the global solar radiation on inclined surfaces. In: Renewable Energy 50 (2013). (2012)

[5] Karlsson, J., Roos, A.: Modelling the angular behavious of the total solar energy transmittance of windows. In: Solar Energy Vol. 69. (1999)