Towards a wireless sensor network for building performance assessment

Author(s):
Frei, Mario

Publication Date:
2014

Permanent Link:
https://doi.org/10.3929/ethz-a-010352509

Rights / License:
In Copyright - Non-Commercial Use Permitted
Towards a Wireless Sensor Network for Building Performance Assessment

Mario Frei

Dr. Zoltan Nagy
Adviser

Prof. Dr. Arno Schlüter
Institute of Technology in Architecture (ITA)
Architecture & Building Systems (A/S)
Swiss Federal Institute of Technology Zurich (ETH)

Oktober 2014
Preface

I would like to thank all the people, who were involved in the conduction of this thesis. Special thanks goes to Dr. Zoltan Nagy, who contributed a lot to this work, as he was my supervisor. He always had advice when needed and made sure, that this thesis is going into the right direction. Furthermore, I want to express my gratitude to Prof. Dr. Arno Schlüter, who is head of the Architecture & Building System (A/S) group at the Institute of Technology in Architecture (ITA), ETH Zurich. He provided valuable advice. Further thanks goes to the whole A/S-team, who was never shy to help and provided a very good atmosphere to work in. Also, I want to thank Frank Theßeling for the good collaboration. Last but not least, I want to thank my family who supported me during this thesis and my entire studies.
Abstract

Although, buildings play an important role in reducing energy consumption and greenhouse gas emissions, the actual state of the building’s insulation and other important building parameters are rarely known. Most times, these parameters are estimated or calculated and not measured. Therefore, undocumented changes during construction, the behaviour of the occupants and other factors, such as prebound and rebound effect, can make a significant difference between the calculations and the true values.

This thesis builds upon the work of [1]. During this thesis a second iteration of a wireless sensor network (WSN) was developed. It is capable of assessing some the current states of the building’s performance. This WSN is capable of measuring heat flux, temperature, solar irradiation, electrical current, signal pulses, air humidity and luminosity. The combination of these measurement leads to u-values, g-values, heat output of radiators and oil consumption, comfort and electrical power usage measurements.

In addition to the hardware the methodology of a previous paper [1] was extended, in order to incorporate the g-value and u-value of windows.
 Contents

 Abstract ii

 List of Tables v

 List of Figures vi

 Notation viii

 Symbols ix

1 Introduction 1

 1.1 Task Definition and Goals 2

 1.2 U-Value 3

 1.3 G-Value 5

2 Theory and Extension of Methodology 7

 2.1 Extension of Methodology 7

 2.2 Influence of Measurement Uncertainty 12

 2.3 Sensitivity 14

 2.4 Numerical Example 15

3 Experimental Implementation 18

 3.1 Measuring Systems Overview 18

 3.2 Predecessor System 19

 3.3 Hardware Architecture 21

 3.4 Software Architecture 33

 3.5 Case Study 37

4 Summary and Conclusions 49

 4.1 Summary of Achievements 49

 4.2 Limitations, Lessons Learned 49

 4.3 Outlook, Impact, Future Work 51
## References

55

### A Source Code

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.1 Node Arduino Script</td>
<td>58</td>
</tr>
<tr>
<td>A.2 Gateway Python Script</td>
<td>67</td>
</tr>
<tr>
<td>A.3 Server PHP Script</td>
<td>73</td>
</tr>
</tbody>
</table>

### B Matlab Code

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.1 U-Value Calculations with Average Method</td>
<td>79</td>
</tr>
<tr>
<td>B.2 U-Value Calculations with Dynamic Method</td>
<td>79</td>
</tr>
</tbody>
</table>

### C Electronic Schematics

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.1 G-Value Schematics</td>
<td>82</td>
</tr>
<tr>
<td>C.2 Sensor Node Schematics</td>
<td>83</td>
</tr>
</tbody>
</table>

### D Second Case Study

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>D.1 Comments</td>
<td>84</td>
</tr>
</tbody>
</table>
List of Tables

1 Notation .................................................. viii
2 Symbols ................................................... ix
3 Set-Points for the Numerical Example .................. 16
4 Arduino Pro Mini Specifications ......................... 23
5 Adhesive Test Overview .................................. 31
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Notation for the zone model</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>General Sensitivity Plot</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>Notation for the zone model</td>
<td>17</td>
</tr>
<tr>
<td>4</td>
<td>Predecessor System</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>Network Architecture</td>
<td>21</td>
</tr>
<tr>
<td>6</td>
<td>Gateway</td>
<td>22</td>
</tr>
<tr>
<td>7</td>
<td>Sensor Node</td>
<td>23</td>
</tr>
<tr>
<td>8</td>
<td>Amplifier Circuit</td>
<td>24</td>
</tr>
<tr>
<td>9</td>
<td>Adhesive Endurance Tests</td>
<td>30</td>
</tr>
<tr>
<td>10</td>
<td>Software Overview</td>
<td>33</td>
</tr>
<tr>
<td>11</td>
<td>Overview MySQL Tables</td>
<td>36</td>
</tr>
<tr>
<td>12</td>
<td>Simulation of the Oil Energy Consumption</td>
<td>38</td>
</tr>
<tr>
<td>13</td>
<td>Floor Plan with Sensor Positions</td>
<td>39</td>
</tr>
<tr>
<td>14</td>
<td>Temperature and Heat Flux Sensor mounted on a Wall</td>
<td>40</td>
</tr>
<tr>
<td>15</td>
<td>Temperature Sensors mounted to the Pipes</td>
<td>41</td>
</tr>
<tr>
<td>16</td>
<td>U-Value Measurement: Interior to Exterior 1 (U1)</td>
<td>41</td>
</tr>
<tr>
<td>17</td>
<td>U-Value Measurement: Interior to Exterior 2 (U2)</td>
<td>42</td>
</tr>
<tr>
<td>18</td>
<td>U-Value Measurement: Interior to Balcony (U3)</td>
<td>42</td>
</tr>
<tr>
<td>19</td>
<td>U-Value Measurement: Interior to Roof (U4)</td>
<td>43</td>
</tr>
<tr>
<td>20</td>
<td>U-Value Measurement: Roller Shutter Housing (U5)</td>
<td>43</td>
</tr>
<tr>
<td>21</td>
<td>Supply and Return Temperature Measurement 1</td>
<td>44</td>
</tr>
<tr>
<td>22</td>
<td>Supply and Return Temperature Measurement 2</td>
<td>44</td>
</tr>
<tr>
<td>23</td>
<td>Comparison U-Values: Assumptions vs. Measurements</td>
<td>45</td>
</tr>
<tr>
<td>24</td>
<td>G-Value Setup mounted to a Window</td>
<td>46</td>
</tr>
<tr>
<td>25</td>
<td>Bang Bang Control of G-Value Apparatus</td>
<td>46</td>
</tr>
<tr>
<td>26</td>
<td>G-Value Measurement with Bang Bang Control</td>
<td>47</td>
</tr>
<tr>
<td>27</td>
<td>PID Control of G-Value Apparatus</td>
<td>48</td>
</tr>
<tr>
<td>28</td>
<td>G-Value Measurement with PID Control</td>
<td>48</td>
</tr>
<tr>
<td>29</td>
<td>Next Generation</td>
<td>54</td>
</tr>
<tr>
<td>30</td>
<td>G-Value Schematics</td>
<td>82</td>
</tr>
<tr>
<td>31</td>
<td>Sensor Node Schematics</td>
<td>83</td>
</tr>
<tr>
<td>32</td>
<td>Floor Plan with Sensor Positions</td>
<td>84</td>
</tr>
<tr>
<td>33</td>
<td>Outside Temperatures of the Nearest Weather Station</td>
<td>85</td>
</tr>
</tbody>
</table>
List of Figures

34 U-Value - Utility Room - Wall ........................................ 86
35 U-Value - Utility Room - Window .................................. 86
36 U-Value - Utility Room - Ceiling ................................. 87
37 U-Value - Utility Room - Ceiling with Different Outdoor Temperature ........................................ 87
38 U-Value - Swimming Pool Room - Wall ........................... 88
39 U-Value - Swimming Pool Room - Window ....................... 88
40 Supply and Return Temperatures - Boiler Room .................. 89
## Notation

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>6LoWPAN</td>
<td>IPv6 over Low power Wireless Personal Area Networks</td>
</tr>
<tr>
<td>ADC</td>
<td>Analog-to-Digital Converter</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>DIP</td>
<td>Dual In-Line Package</td>
</tr>
<tr>
<td>EEPROM</td>
<td>Electrically Erasable Programmable Read-Only Memory</td>
</tr>
<tr>
<td>FDM</td>
<td>Fused Deposition Modelling</td>
</tr>
<tr>
<td>GSM</td>
<td>Global System for Mobile Communications</td>
</tr>
<tr>
<td>HVAC</td>
<td>Heating, Ventilation and Air Conditioning</td>
</tr>
<tr>
<td>I/O Pins</td>
<td>Input and Output Pins</td>
</tr>
<tr>
<td>I^2C</td>
<td>Inter-Integrated Circuit</td>
</tr>
<tr>
<td>Op-Amp</td>
<td>Operational Amplifier</td>
</tr>
<tr>
<td>PCB</td>
<td>Printed Circuit Board</td>
</tr>
<tr>
<td>PLA</td>
<td>Polylactic Acid</td>
</tr>
<tr>
<td>PWM</td>
<td>Pulse Width Modulation</td>
</tr>
<tr>
<td>SHGC</td>
<td>Solar Heat Gain Coefficient</td>
</tr>
<tr>
<td>SRAM</td>
<td>Static Random Access Memory</td>
</tr>
<tr>
<td>URL</td>
<td>Uniform Resource Locator</td>
</tr>
<tr>
<td>WSN</td>
<td>Wireless Sensor Network</td>
</tr>
</tbody>
</table>

Table 1: Notation
## Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Explanation</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_E$</td>
<td>External ambient temperature</td>
<td>°C</td>
</tr>
<tr>
<td>$T_I$</td>
<td>Internal ambient temperature</td>
<td>°C</td>
</tr>
<tr>
<td>$T_S$</td>
<td>Supply temperature</td>
<td>°C</td>
</tr>
<tr>
<td>$T_R$</td>
<td>Return temperature</td>
<td>°C</td>
</tr>
<tr>
<td>$T_Z$</td>
<td>Zone temperature</td>
<td>°C</td>
</tr>
<tr>
<td>$U_E$</td>
<td>U-value of opaque envelope</td>
<td>$W/(m^2K)$</td>
</tr>
<tr>
<td>$A_E$</td>
<td>Area of opaque envelope</td>
<td>$m^2$</td>
</tr>
<tr>
<td>$U_F$</td>
<td>U-value of fenestration</td>
<td>$W/(m^2K)$</td>
</tr>
<tr>
<td>$A_F$</td>
<td>Area of transparent envelope</td>
<td>$m^2$</td>
</tr>
<tr>
<td>$U_H$</td>
<td>Heat transfer coefficient of the heating system</td>
<td>$W/(m^2K)$</td>
</tr>
<tr>
<td>$A_H$</td>
<td>Area of the heating system</td>
<td>$m^2$</td>
</tr>
<tr>
<td>$Q_{L1}$</td>
<td>Conductive heat loss</td>
<td>$W$</td>
</tr>
<tr>
<td>$Q_{L2}$</td>
<td>Conductive heat loss</td>
<td>$W$</td>
</tr>
<tr>
<td>$Q_L$</td>
<td>Conductive heat loss</td>
<td>$W$</td>
</tr>
<tr>
<td>$Q_{H1}$</td>
<td>Heat supplied to the zone</td>
<td>$W$</td>
</tr>
<tr>
<td>$Q_{H2}$</td>
<td>Heat supplied to the zone</td>
<td>$W$</td>
</tr>
<tr>
<td>$Q_S$</td>
<td>Solar heat gain</td>
<td>$W$</td>
</tr>
<tr>
<td>$Q_{NL1}$</td>
<td>Total power supplied to the zone</td>
<td>$W$</td>
</tr>
<tr>
<td>$Q_{NL2}$</td>
<td>Total power supplied to the zone</td>
<td>$W$</td>
</tr>
<tr>
<td>$G$</td>
<td>G-value</td>
<td>-</td>
</tr>
<tr>
<td>$I$</td>
<td>Solar irradiation</td>
<td>$W/m^2$</td>
</tr>
<tr>
<td>$c$</td>
<td>Specific heat capacity of water</td>
<td>$J/(kgK)$</td>
</tr>
<tr>
<td>$m$</td>
<td>Mass flow of water</td>
<td>$kg/s$</td>
</tr>
<tr>
<td>$d$</td>
<td>Thickness</td>
<td>$m$</td>
</tr>
<tr>
<td>$\kappa$</td>
<td>Current state of the heating system</td>
<td>-</td>
</tr>
<tr>
<td>$\kappa_1$</td>
<td>Current state of the heating system</td>
<td>-</td>
</tr>
<tr>
<td>$\kappa_2$</td>
<td>Current state of the heating system</td>
<td>-</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>Current state of the losses</td>
<td>-</td>
</tr>
<tr>
<td>$\gamma_1$</td>
<td>Current state of the losses</td>
<td>-</td>
</tr>
<tr>
<td>$\gamma_2$</td>
<td>Current state of the losses</td>
<td>-</td>
</tr>
<tr>
<td>$\gamma_3$</td>
<td>Current state of the losses</td>
<td>-</td>
</tr>
<tr>
<td>$\gamma_E$</td>
<td>Current state of the losses</td>
<td>-</td>
</tr>
<tr>
<td>$\gamma_F$</td>
<td>Current state of the losses</td>
<td>-</td>
</tr>
<tr>
<td>$\lambda$</td>
<td>Thermal conductivity</td>
<td>$W/(mK)$</td>
</tr>
<tr>
<td>$\varphi$</td>
<td>Increase of insulation</td>
<td>$W/K$</td>
</tr>
<tr>
<td>$\eta$</td>
<td>Change of the current heating area</td>
<td>-</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>Change of the fenestration u-value</td>
<td>-</td>
</tr>
<tr>
<td>$\theta$</td>
<td>Change of the g-value</td>
<td>-</td>
</tr>
<tr>
<td>$(T_{SR})_1$</td>
<td>Supply temperature after retrofit</td>
<td>°C</td>
</tr>
<tr>
<td>$(T_{SR})_2$</td>
<td>Supply temperature after retrofit</td>
<td>°C</td>
</tr>
<tr>
<td>$(\gamma)^r$</td>
<td>After retrofit</td>
<td>-</td>
</tr>
<tr>
<td>$(\gamma)^c$</td>
<td>Current / before retrofit</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 2: Symbols
1 Introduction

In 2012 the HVAC, domestic hot water and lighting, accounted for 45 percent of the total energy consumption in Switzerland [2]. This large share on the energy consumptions hints a large leverage on energy consumption and greenhouse gas emissions [3], [4], [5].

Currently, the energy consumption of buildings is calculated based on assumptions of the building properties. Standards, like the SIA 380/1 in Switzerland, are available for these calculations. Values for the building properties may not always be available or correct. The building materials may behave unexpectedly due to ageing, moisture or wrong application. Undocumented changes may have taken place during or after construction. This is why, the assumed properties can differ significantly from the true value.

Further, the behaviour of occupants has a significant influence on the performance of the building. Two of these effects are known as rebound and prebound effect [6]. That is, that occupants consume less energy then suggested by the energy performance rating in buildings with a bad performance (prebound) and they consume more energy than suggested by the energy performance rating, in low-energy dwellings (rebound).

In order to access better data of the building properties and the occupants behaviour, we suggest to measure some of these actual properties. For this task we developed a sensor kit. We believe, that on-site measurements will lead to a more accurate building performance assessment. In addition, the occupants behaviour and its influence on the buildings energy demand can be assessed. Overall the number of assumptions can be lowered. These measurements build a solid base for further action, such as suggestions for retrofit measures or alternations on the current building operation. After a retrofit, the sensor kit could be deployed a second time on the same site, in order of a quality inspection.

Within this thesis an already existing sensor network [1] was further developed in order to make it smaller, to incorporate more sensors and make the usage of it more versatile. Further the methodology, which was carried out with the previous sensor network was extended by the u-value and g-values of the fenestration. The newly developed sensor kit was used in an actual on-site measurement. This case study with the process of setting up the system, supervision during the measurement, dismantling and data analysis will be presented as well.
This report is structured as follows: In the following section, the extended methodology is carried out. In the third sections the development of the wireless sensor network is presented. In the fourth section the measurements and error analysis are carried out, including error and sensitivity analysis. And in the last section the conclusions are discussed.

1.1 Task Definition and Goals

In this subsection the requisites to the sensor kit and the theory will be discussed. The sensor kit to be developed has to be able to measure the following entities:

- U-value
- G-value
- Air temperature and humidity
- Oil consumption
- Electric power consumption of appliances in a room
- Electric power consumption of an entire building
- Heat output of a radiator
- Luminosity

Additionally the sensor kit should become easier to install on site and it should have an cleaner look. Further the sensor node type (sensor configuration) should be adjustable in the hardware and not anymore in the software as before. Overall, the sensor node should become more compact in size and easier to use. In the beginning, there was the idea, that one could use one sensor kit to access the thermal performance of one room. And hence the requirement to the hardware and software to be scalable. Meaning, that more than one sensor kit could be used within a building. Finally, the new system should be able to work independently of on-site LAN access. The often mentioned wireless sensor kit refers only to the communication. We decided to forgo the energy self-sufficiency of the device in this iteration step. Thus, the all the devices are still wired in the sense of power supply.
1.2 U-Value

The u-value is the thermal transmittance of building element. In the ISO 7345 it is defined as the "Heat flow rate in the steady state divided by area and by the temperature difference between the surroundings on each side of a system". It is a measure of conductive heat loss through the building element. The unit of the u-value is W/(m²K). There are several methods to obtain the U-value of a building element: thermal imaging, multiple temperature measurements and the heat flux method.

1.2.1 Thermal Imaging

[7] gives a good overview and insight of thermal assessments using thermal imaging. Thermal imaging generally does not yield quantitative results. However, [8] proposes a method to quickly measure the u-value using thermal imaging. Nevertheless, it is useful to show temperature distribution and locating thermal inhomogeneities. But one has to be careful with the thermal bridges found by thermal imaging. It is hard to identify the origin of heat which lead to higher temperature on the surface. Attention has to be paid also on the color-scale used to represent the temperatures. The choice of the color-scale can alter the appearance element under inspection. Further, the thermal image is influenced amongst others by the temperature profile before the actual measurement and the solar irradiation. This makes it difficult to compare thermal images. Nevertheless, thermal imaging can help to find a proper (homogeneous) spot on the building element for the measurement of the u-value with one of the following methods. Thermal imaging was not used in the scope of this thesis.

1.2.2 Multiple Temperature Measurements

It is possible to measure the u-value only using temperature-probes. For this, the interior and exterior ambient temperatures have to measured, as well as the inner surface temperature of the building element. Then the u-value can be calculated as follows:

\[ q_1 = \alpha (T_{air,i} - T_{wall,i}) \]  
\[ q_2 = U (T_{air,i} - T_{air,e}) \]
\[ q_1 = q_2 \]  
\[ U = \alpha \frac{(T_{\text{air},i} - T_{\text{wall},i})}{(T_{\text{air},i} - T_{\text{air,e}})} \] 

Where \( q_1 \) [W/m²] is the convective heat transfer from the inside air to the inside wall surface per area, \( q_2 \) [W/m²] is the heat transfer from the inside air through the building element to the outside air per area, \( T_{\text{air},i} \) [°C] is the inside air temperature, \( T_{\text{air,e}} \) [°C] is the outside air temperature, \( T_{\text{wall},i} \) [°C] is the wall temperature inside, \( U \) [W/(m²K)] is the u-Value and \( \alpha \) [W/(m²K)] is the convective heat transfer coefficient, which can be assumed to be constant for all walls. The minimal temperature difference between inside and outside is 10-15 K. In the scope of this thesis, this method was not applied.

### 1.2.3 Heat Flux Method

The heat flux method is the method of choice in the scope of this thesis. The u-value can be calculated from the measurements of the ambient temperatures on the inside and the outside of the building element under investigation, as well as the measurement of the heat flux density through the element on the inside of the element. The, to this method corresponding, standard is ISO 9869: "Thermal Insulation, Building Elements, In-situ Measurement of Thermal Resistance and Thermal Transmittance" [9]. For the calculation of the u-value from the measurement two methods are described in the standard: the average method and the dynamic analysis method. The former is straightforward, the latter is more sophisticated but should yield better results with less data.

#### 1.2.3.1 Average Method

The average method is simple and straightforward. As described in [11] and [9], the u-value can be obtained by dividing the sum of the measured heat fluxes by the sum of the measured temperature differences from one side to the other.

\[ U = \frac{\sum_{i=1}^{N} q_i}{\sum_{i=1}^{N} (T_i^I - T_i^E)} \] 

Where \( U \) [W/(m²K)] is the u-value, \( q \) [W/m²] is the heat flux density through the building element and \( T_i^I \) [°C] and \( T_i^E \) [°C] are the indoor and outdoor ambient temperatures. \( i \) is the index of the individual measurement and \( N \) is the total
1.3 G-Value

number of measurements. The standard suggests that the minimum test duration is at least 72h and an integer multiple of 24 hours. The corresponding Matlab file can be found on page 79.

1.2.3.2 Dynamic Analysis Method  The dynamic analysis method is more sophisticated than the average method. This method takes into account the variations of temperature, which usually occur on-site, due to the day and night cycle, as well as the weather. The whole algorithm is presented in the Appendix B of [9]. The algorithm was implemented in Matlab. This method takes into account the dependence of the current heat flux density on the previous temperatures and may potentially lead to better results, especially when the temperatures fluctuate considerably. However, this method is much more complex as the average method and therefore more cumbersome to apply. The corresponding Matlab file can be found on page 79. In the case study, that can be found in the appendix, the output of the dynamic method script is plotted along with the output of the average method script.

1.3 G-Value

The g-value characterizes the energy transmittance by solar irradiation through glazing. Other names for the g-value are, total solar energy transmittance, solar heat gain coefficient (SHGC), solar gain or solar shading coefficient. The definition might vary slightly, depending on the designation.

When solar radiation hits glazing, some part of the radiation is reflected, some part is transmitted through the glazing and another part of the radiation is absorbed by the glazing. The absorbed part of the radiation is converted into thermal energy and heating up the glazing. This thermal energy is distributed through the glazing and may flow to the surface, given that the temperatures surrounding the glazing are lower than the glazing itself. This means, that the thermal energy leaves the glazing to the outside, as well as to the inside. The thermal energy entering the building and the solar radiation transmitted directly through the glazing are the total amount of solar energy, that entered the building. Solar radiation passing through multiple glazing fenestration is subject to
multiple reflections and re-radiations.

\[ g = \frac{E_r + E_c}{I_0} \]  \hspace{1cm} (1.6)

Where \( g \) \([-\) is the g-value, \( I_0 \) \([W/m^2]\) is the total solar radiation, \( E_r \) \([W/m^2]\) is the energy transmitted by radiation and \( E_c \) \([W/m^2]\) is the energy transmitted by conduction. The g-value is the ratio of solar energy transmitted through the window over total solar radiation. The g-value is measured normal to the glass surface in the spectrum from 300 nm to 2500 nm. The according standard is the "ISO 9050: Glass in building - Determination of light transmittance, solar direct transmittance, total solar energy transmittance, ultraviolet transmittance and related glazing factors" \[11\]. Using spectrophotometer the g-value can be measured in the laboratory \[12\].
2 Theory and Extension of Methodology

In this section the methodology of [1] is introduced and extended. Then the analysis of the measurement uncertainty and sensitivity analysis are performed, followed by a numerical example.

2.1 Extension of Methodology

The goal of the following is to relate the supply temperature $T_S$ to the retrofit parameters. The zone consists of three main elements: a heating system, wall and a window, denoted with the subscripts $H$, $E$ and $F$ respectively. Further denote the temperature of the zone by $T_Z$, the exterior temperature by $T_E$ and the return temperature in $T_R$. All temperatures are given in °C. The conductive heat loss through the envelope $Q_L$, in [W], consists of the heat loss through the wall $Q_{L1}$, in [W], and through the fenestration $Q_{L2}$, in [W],

$$Q_L = Q_{L1} + Q_{L2}. \quad (2.1)$$
The heat loss through the wall $Q_{L1}$ can be expressed by

$$Q_{L1} = U_E \cdot A_E \cdot (T_Z - T_E),$$  
(2.2)

where $U_E$, in $[W/(m^2K)]$, and $A_E$, in $[m^2]$, are the u-value of the wall and the area of the opaque envelope. The heat loss through the window $Q_{L2}$ can be described by

$$Q_{L2} = U_F \cdot A_F \cdot (T_Z - T_E),$$  
(2.3)

where $U_F$, in $[W/(m^2K)]$, and $A_F$, in $[m^2]$, are the u-value of the window and the area of the transparent envelope. Inserting (2.2) and (2.3) in (2.1) yields

$$Q_L = (U_E \cdot A_E + U_F \cdot A_F) \cdot (T_Z - T_E).$$  
(2.4)

Energy is supplied to the zone by the heating system and by solar irradiation. The heat $Q_R$, in $[W]$, supplied by the heating system can be expressed by

$$Q_{R1} = \dot{m} \cdot c \cdot (T_S - T_R),$$  
(2.5)

where $\dot{m}$, in $[kg/s]$, expresses the mass flow rate of water in the heating system, $c$, in $[J/(kgK)]$, is the specific heat capacity of water. Alternatively, the heat supplied by the zone can be expressed without the mass flow rate of water. This yields to

$$Q_{R2} = U_H \cdot A_H \cdot \frac{T_S - T_R}{\log \left(\frac{T_S-T_Z}{T_R-T_Z}\right)},$$  
(2.6)

where $U_H$, in $[W/(m^2K)]$, and $A_H$, in $[m^2]$, are the heat transfer coefficient and the area heating system. The solar heat gain $Q_S$, in $[W]$ can be expressed as

$$Q_S = G \cdot I \cdot A_F.$$  
(2.7)

with $I$, in $[W/m^2]$, as the solar irradiance and $G$ as the g-value, which is unit-less. The incoming power from the heating system and the sun can be combined to

$$Q_{IN1} = G \cdot I \cdot A_F + \dot{m} \cdot c \cdot (T_S - T_R)$$  
(2.8)
or without the mass flow rate of the heating system’s water to

\[ Q_{\text{IN2}} = G \cdot I \cdot A_F + U_H \cdot A_H \cdot \frac{T_S - T_R}{\log\left(\frac{T_S - T_Z}{T_R - T_Z}\right)}. \]  

(2.9)

In order to eliminate the return temperature \( T_R \), (2.5) and (2.6) can be equated.

\[ Q_{R1} = Q_{R2} \]  

(2.10)

\[ \dot{m} \cdot c \cdot (T_S - T_R) = U_H \cdot A_H \cdot \frac{T_S - T_R}{\log\left(\frac{T_S - T_Z}{T_R - T_Z}\right)} \]  

(2.11)

Dividing both sides of the equation (2.11) by \((T_S - T_R)\) yields

\[ \log\left(\frac{T_S - T_Z}{T_R - T_Z}\right) = \frac{U_H \cdot A_H}{\dot{m} \cdot c}, \]  

(2.12)

which can be solved for \( T_R \):

\[ T_R = T_Z + (T_s - T_Z) \cdot \exp\left(-\frac{U_H \cdot A_H}{\dot{m} \cdot c}\right) \]  

(2.13)

Assuming steady state conditions the energy entering the zone \( Q_{\text{IN}} \) and the energy leaving the zone \( Q_L \) are equal.

\[ Q_L = Q_{\text{IN1}} = Q_{\text{IN2}} \]  

(2.14)

This is the same as equalizing (2.1) and (2.8), replacing \( T_R \) by (2.13), which leads to

\[ (U_E \cdot A_E + U_F \cdot A_F) \cdot (T_Z - T_E) = \dot{m} \cdot c \cdot (T_S - T_Z) \cdot \left(1 - \exp\left(-\frac{U_H \cdot A_H}{\dot{m} \cdot c}\right)\right) + G \cdot I \cdot A_F. \]  

(2.15)

Solving (2.15) for the supply temperature \( T_S \) yields

\[ T_s = T_Z + \frac{(U_E \cdot A_E + U_F \cdot A_F) \cdot (T_Z - T_E) - G \cdot I \cdot A_F}{\dot{m} \cdot c \cdot \left(1 - \exp\left(-\frac{U_H \cdot A_H}{\dot{m} \cdot c}\right)\right)}. \]  

(2.16)

The current state of heating system can be defined from the following equa-
\[
Q_{R1} = Q_{R2} \tag{2.17}
\]

Dividing by \((T_S - T_R)\) yields
\[
\kappa_1 = \log \left( \frac{T_S - T_Z}{T_R - T_Z} \right) \tag{2.18}
\]
for one side of the equation and
\[
\kappa_2 = \frac{A_H c \cdot U_H}{\dot{m} \cdot c} \tag{2.19}
\]
for the other side of the equation, i.e. \(\kappa_1\) and \(\kappa_2\) are equal. \(\kappa_1\) is a measure for the current state of the heating system. It is unit-less. The advantage of \(\kappa_1\) is, that it is easier to determine since it only need temperature measurements. Where as \(\dot{m}, A_H\) and \(U_H\) are generally more cumbersome to measure in-situ or determine otherwise.

For the current state of losses can be derived from the following equation:
\[
Q_{IN1} = Q_L \tag{2.20}
\]

Inserting the corresponding terms for \(Q_{IN1}\) and \(Q_L\) yields:
\[
\dot{m} \cdot c \cdot (T_S - T_R) + G \cdot I \cdot A_F = (U_E \cdot A_E + U_F \cdot A_F) \cdot (T_Z - T_E) \tag{2.21}
\]

This equation can be rearranged, such that there appears the current state of losses, as in [1], on one side denoted as \(\gamma_1\) and on the other side the rest of the terms, summarized in \(\gamma_2\).
\[
\gamma_1 = \frac{(T_S - T_R)}{(T_Z - T_E)} \tag{2.22}
\]
\[
\gamma_2 = \frac{U_E \cdot A_E + U_F \cdot A_F}{\dot{m} \cdot c} - \frac{G \cdot I \cdot A_F}{\dot{m} \cdot c \cdot (T_Z - T_E)} \tag{2.23}
\]

\(\gamma_2\) can be further split up, in three parts corresponding to energy transfers through the wall and window.
\[
\gamma_2 = \frac{\gamma_E}{\dot{m} \cdot c \cdot (T_E - T_Z)} + \frac{\gamma_E}{\dot{m} \cdot c} + \frac{\gamma_F}{\dot{m} \cdot c} \tag{2.24}
\]
\( \gamma_E = \frac{A_E \cdot U_E}{\dot{m} \cdot c} \) \hspace{1cm} (2.25)

\( \gamma_E \) corresponds to the conductive losses through the opaque envelope.

\( \gamma_F = \frac{A_F \cdot U_F}{\dot{m} \cdot c} \) \hspace{1cm} (2.26)

\( \gamma_F \) corresponds to the conductive losses through the window.

\( \gamma_G = \frac{-A_F \cdot G \cdot i}{\dot{m} \cdot c \cdot (T_E - T_Z)} \) \hspace{1cm} (2.27)

\( \gamma_G \) corresponds to the radiative losses through the window. Since solar irradiation is the only radiation considered in the calculations, it is not a loss but an energy gain, hence the negative sign. Note that all \( \gamma \)'s are unit-less.

Four simplified retrofit measures are introduced:

- Increase of the u-value of the wall
- Increase of the heating system’s area
- Decrease of the u-value of the window
- Increase of the g-value of the window

For an increase of the u-value of the wall, it is assumed that an additional layer of insulating is applied to the outer opaque envelope. This layer has a thickness of \( d \), in [m], and a a thermal conductivity of \( \lambda \), in [W/(mK)]. This can be stated as

\[
\frac{1}{U_{E^r}} = \frac{1}{U_{Ec}} + \frac{d}{\lambda}
\]

where \( U_{Ec} \) and \( U_{E^r} \) are the wall’s u-value before and after the retrofit, respectively.

\[
U_{E^r} = \frac{U_{Ec}}{1 + U_{Ec} \cdot \frac{d}{\lambda}} = \frac{U_{Ec}}{1 + U_{Ec} \cdot \varphi}
\]

\( d \) and \( \lambda \) can be concentrated in \( \varphi \), which will be a retrofit parameter, with \( \varphi = \frac{d}{\lambda} \geq 0 \). The other retrofit parameters are more straight forward. The increase of the heating system’s area is simply expressed by

\[
\eta = \frac{A_{H^r}}{A_{H^c}} \geq 1,
\]
where $\eta$ is unit-less. The decrease of the window’s $u$-value can be expressed by

$$\sigma = \frac{U_F^c}{U_F^r} \geq 1,$$

(2.31)

where $\sigma$ is unit-less. Since an increase of the window’s $u$-value usually means, that the window as to be replaced, this expression is simpler than $\varphi$. The increase of the $g$-value can be expressed by

$$\theta = \frac{G^r}{G^c} \geq 1,$$

(2.32)

where $\theta$ is unit-less. Note, that the retrofit value is in the numerator. This is that an increase of any of the retrofit parameter yields a better thermal performance of the zone.

The current supply temperature can be rewritten as

$$T_{sc} = T_Z + \gamma \cdot \frac{(T_Z - T_E)}{1 - \exp(-\kappa)}.$$

(2.33)

The retrofit supply temperature can be expressed as

$$(T_{sr})_1 = T_Z + \left(\frac{U_F^c}{1 + \varphi \cdot U_E^c} \cdot A_E + \frac{\sigma^{-1} \cdot U_F^c \cdot A_F}{\eta \cdot \sigma} \cdot (T_Z - T_E) - \theta \cdot G \cdot I \cdot A_F \right) \cdot \frac{m \cdot c \cdot (1 - \exp(-\eta \cdot \kappa))}{\eta \cdot \sigma}$$

(2.34)

or alternatively as

$$(T_{sr})_2 = T_Z + \left(\frac{\gamma_E}{1 + \varphi \cdot U_E^c} + \frac{\gamma_F}{\sigma} - \theta \cdot \gamma_G \right) \cdot \frac{(T_Z - T_E)}{1 - \exp(-\eta \cdot \kappa)}.$$

(2.35)

### 2.2 Influence of Measurement Uncertainty

In this subsection the influence of measurement uncertainty on the $u$-value measurement will be presented. In [13] demonstrates an error analysis method for the averaging method based on a Taylor expansion method. However, in this section we approach the matter differently. We start at

$$U = \frac{\sum_{i=1}^{N} q_i}{\sum_{i=1}^{N} (T_i^d - T_i^E)},$$

(2.36)
which is, how the u-value is measured. This equation was already introduced in section 1.2.3.1 as equation 1.5. The fraction can be expanded with \(1/N\), which yields

\[
U = \frac{\frac{1}{N} \sum_{i=1}^{N} q_i}{\frac{1}{N} \sum_{i=1}^{N} (T_i^I) - \frac{1}{N} \sum_{i=1}^{N} (T_i^E)}. \tag{2.37}
\]

Now there is an average value of heat fluxes in the numerator and a difference of average values of temperatures in the denominator. By averaging the measurement values the error propagates as following:

\[
\bar{f} = \frac{1}{N} \sum_{i=1}^{N} f_i \tag{2.38}
\]

\[
\Delta \bar{f} = \frac{1}{\sqrt{N}} \Delta f \tag{2.39}
\]

Where \(f\) is substituting for \(T_i^E, T_i^I\) and \(q\). \(\bar{f}\) is the average of all single measurements \(f_i\), \(\Delta f\) is the error of \(f_i\) and \(\Delta \bar{f}\) is the error of the average \(\bar{f}\). The error of the u-value can then be retrieved as follows,

\[
\Delta U = \sqrt{\left(\frac{\partial U}{\partial \bar{q}} \Delta \bar{q}\right)^2 + \left(\frac{\partial U}{\partial T_i^E} \Delta T_i^E\right)^2 + \left(\frac{\partial U}{\partial T_i^I} \Delta T_i^I\right)^2}. \tag{2.40}
\]

With the partial derivatives being

\[
\frac{\partial U}{\partial \bar{q}} = \frac{1}{T_i^I - T_i^E}, \tag{2.41}
\]

\[
\frac{\partial U}{\partial T_i^I} = -\frac{\bar{q}}{(T_i^I - T_i^E)^2}, \tag{2.42}
\]

\[
\frac{\partial U}{\partial T_i^E} = \frac{\bar{q}}{(T_i^I - T_i^E)^2}. \tag{2.43}
\]

### 2.2.1 Example

Assuming wall with on a cold day with an outside temperature \(T_i^E\) of 0 °C, an inside temperature \(T_i^I\) of 20 °C and a heat flux \(\bar{q}\) of 10 W/m² through it. Further the measurement errors \(\Delta \bar{q}\) of 1.5 W/(m²K), \(\Delta T_i^I\) 0.5 K and \(\Delta T_i^E\) 0.5
2.3 Sensitivity

*K* are assumed. Equation \([2.37]\) yields an \(u\)-value of 0.5 \(W/(m^2 K)\). With equation \([2.40]\) the estimated error \(\Delta U\) is 0.0771 \(W/m^2 K\) or 15.4 %. Performing the same measurement on a warmer day, i.e. with an outside temperature \(T_E\) of 10 \(^\circ\)C, leads to an estimated error \(\Delta U\) of 0.166 \(W/m^2 K\) or 33.2 %. This example demonstrates the need for high temperatures, in order to improve measurement uncertainty.

2.3 Sensitivity

In this section the sensitivity of the supply temperature after the retrofit to the retrofit parameters \(\varphi, \eta, \sigma\) and \(\theta\). For this task the partial derivatives of the supply temperature after the retrofit are expressed:

\[
\frac{\partial T_{Sr}}{\partial \varphi} = -\frac{A_E (-T_E + T_Z) (U_E c)^2}{c (1 - e^{-\eta \kappa}) \dot{m} (1 + U_E c^2 \varphi)^2} \tag{2.44}
\]

\[
\frac{\partial T_{Sr}}{\partial \eta} = \frac{e^{-\eta \kappa} \kappa \left( -A_F Gi \theta + (-T_E + T_Z) \left( \frac{A_F U_F c}{\sigma} + \frac{A_F U_F c^2}{1 + U_E c^2 \varphi} \right) \right)}{c (1 - e^{-\eta \kappa})^2 \dot{m}} \tag{2.45}
\]

\[
\frac{\partial T_{Sr}}{\partial \sigma} = -\frac{A_F (-T_E + T_Z) U_F c}{c (1 - e^{-\eta \kappa}) \dot{m} \sigma^2} \tag{2.46}
\]

\[
\frac{\partial T_{Sr}}{\partial \theta} = -\frac{A_F Gi c}{c (1 - e^{-\eta \kappa}) \dot{m}} \tag{2.47}
\]

These equations can be simplified to:

\[
\frac{\partial T_{Sr}}{\partial \varphi} = \frac{c_1}{(c_2 + \varphi)^2} \tag{2.48}
\]

\[
\frac{\partial T_{Sr}}{\partial \eta} = -\frac{e^{-\eta \kappa} c_4}{(1 - e^{-\eta \kappa})^2} \tag{2.49}
\]

\[
\frac{\partial T_{Sr}}{\partial \sigma} = \frac{c_4}{\sigma^2} \tag{2.50}
\]

\[
\frac{\partial T_{Sr}}{\partial \theta} = c_5 \tag{2.51}
\]

From equation \([2.51]\), one can see, that the change of the supply temperature
is independent of $\theta$. This means, that after an improvement of $\theta$, the supply temperature can still be influenced with the same impact, by further improving $\theta$. This is not the case for equations (2.48) to (2.50). Figure 2 shows a very general exemplary contour for these equations. After decreasing the supply temperature by a certain amount, by increasing the retrofit parameter by a certain amount, further improvement of the retrofit parameter by the same amount will yield a smaller decrease of the supply temperature.

Figure 2: General Sensitivity Plot

2.4 Numerical Example

In this section the formula earlier developed for the supply temperature will be demonstrated in a numerical example. At the moment, this example is purely
virtual no actual counterpart in reality. The chosen setpoints for the numerical example are listed in table [3]. The solar irradiation was set to zero. This allevi-

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_E$</td>
<td>-7.8</td>
<td>°C</td>
</tr>
<tr>
<td>$T_S$</td>
<td>44.8</td>
<td>°C</td>
</tr>
<tr>
<td>$T_R$</td>
<td>41</td>
<td>°C</td>
</tr>
<tr>
<td>$T_Z$</td>
<td>21.6</td>
<td>°C</td>
</tr>
<tr>
<td>$A_E$</td>
<td>14.4</td>
<td>m²</td>
</tr>
<tr>
<td>$A_F$</td>
<td>14.4</td>
<td>m²</td>
</tr>
<tr>
<td>$A_H$</td>
<td>10</td>
<td>m²</td>
</tr>
<tr>
<td>$U_E$</td>
<td>2.25</td>
<td>W/(m²K)</td>
</tr>
<tr>
<td>$U_F$</td>
<td>1</td>
<td>W/(m²K)</td>
</tr>
<tr>
<td>$I$</td>
<td>0</td>
<td>W/m²</td>
</tr>
<tr>
<td>$c$</td>
<td>4180</td>
<td>J/(kgK)</td>
</tr>
<tr>
<td>$m$</td>
<td>0.12</td>
<td>kg/s</td>
</tr>
<tr>
<td>$\kappa$</td>
<td>0.22</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 3: Set-Points for the Numerical Example

ates the graphical representation of the supply temperature after the retrofit as function of the retrofit parameters. Because of the lack of solar irradiation, the retrofit parameter $\theta$, corresponding to a change of the g-value of the window, can be neglected, since it has no influence without solar irradiation. The result of inserting the set-points from table [3] in to equation (2.34) is depicted in figure [3]. Each plane represents the possible combination of retrofit parameters $\varphi$, $\eta$ and $\sigma$, which have the same supply temperature as a result.
Figure 3: Notation for the zone model
3 Experimental Implementation

In this section an overview of other existing measuring systems will be given. Furthermore, the predecessor system will be presented. Moreover, the electronics, housings and software of the further developed sensor kit will be described.

3.1 Measuring Systems Overview

Wireless communication has become very popular and is apparent in our daily live. Cellphones, smartphones, notebooks, head-sets, we use these gadget virtually every day. All of them can communicate with each other using different technologies, such as wireless LAN, Bluetooth, GSM amongst them. Wireless sensor networks (WSN) are getting more popular as well. [14] gives a general overview over wireless sensor networks. General concerns usually are reliability, power supply, scalability and low maintenance. Radio frequency can be difficult to work with. With more wireless gadgets around, there are almost everywhere devices communicating wireless already. Disturbances from other wireless devices and building elements may occur. Changes in the environment may also influence the communication. This is why wireless mesh networks are often used. Because data packets can be routed along different devices, mesh networks are very robust to changes in the environment and disturbances. It is also very simple to add an additional sensor node or to remove one. Besides the physical placement or displacement of the sensor node, no additional configuration of the network has to performed. That is why mesh networks allow for scalability and reliability. [15] gives guidelines about best practice with wireless mesh networks. Power supply is another issue with distributed sensors. Changing one battery doesn’t sound like much, but with lots of distributed sensors it will get cumbersome. Often a regenerative power source like solar radiation or vibration is used to charge a battery. Indoor, this task is more difficult than outdoor, because of the lack of energy sources and because the restrictions in dimension of the sensors. [16] discusses these issues.

As for wireless communication Zigbee and 6LoWPAN are usually considered. In building environments one might think that W-LAN would be a good solution as well. But because of the electrical power requirement of W-LAN modules, this approach is usually abandoned. ZigBee is a specification of a communication
3.2 Predecessor System

The following subsection will give an overview over the predecessor sensor network. In [1] Nagy et al. developed a sensor network, in order to gather data for the simulation of the building. This network consisted of sensor nodes, hop nodes and coordinator nodes. The sensor nodes were built from an Arduino Uno, XBee-module and a XBee-shield for the Arduino. The XBee-module was configured as a router and sensors could be attached to the prototyping area of the XBee-Shield. Everything was packed into a off-the-shelf plastic enclosure. The sensor node was powered by wall adapter power supply. The XBee-shield was configured differently, for each sensor combination used with the sensor-node.

The hop nodes were basically XBee-modules configured as routers and powered by a wall adapter power supply. For protection, they were inside a smaller off-the-shelf plastic enclosure. The third node type was a coordinator node, which consisted of an Arduino Mega with an ethernet-shield, a XBee-shield, an XBee-module and a hardware real time clock. The XBee-module was configured as the coordinator of the ZigBee-Mesh-Network. Everything was again packed in to an off-the-shelf plastic enclosure. Figure 4 shows the hardware used in the previous system. Triggered by a request from the coordinator node, the measured values were sent from the sensor nodes to the coordinator nodes over ZigBee-network. In the case that, there was no direct connection between coordinator and sensor node, hop nodes could be placed in between. The hop nodes functioned as relay stations, expanding the ZigBee network. The coordinator nodes were plugged into the local LAN and had internet access through this LAN. The sensor values were
then sent via a http-request to a server hosting a php-script. The php-script then inserted the values into an MySQL-table. For each floor a separate coordinator node was used. On the coordinator nodes and the sensor nodes two different C++ scripts were running. The coordinator-script had to be adapted for each physical coordinator-node. The MAC-address, the local IP-address and all XBee-addresses of the corresponding sensor-nodes had to be changed in the code. For the sensor-node the sensor type was defined in the software and had therefore to be changed for different sensor-node-types as well. The XBees had to be configured differently for each floor, since only one coordinator-XBee is allowed per XBee-network. This previous system used air temperature and humidity sensors (SHT15), lumionisity sensors (TSL2561), temperature sensors (DS18B20) and an oil flow meter (BZ-5-RR). There were four different sensor node types: external (SHT15, TSL2561), internal (SHT15), radiator (two DS18B20) and oil flow (two DS18B20, HZ-5RR). The previous sensor network was a first attempt to access the thermal performance of a building. It was assembled and deployed in a very short time. That system was able to log all the needed data and executed its task successfully over several month. Despite the success of the first operation, there was still room for improvement and from the experience of the first use, new requests have arisen. This extended set of requirement is part of the task definition for the hardware part of this thesis, which will be discussed in the next subsection.
3.3 Hardware Architecture

In this subsection the hardware architecture of the new kit will be described, including network, electronics, housings and sensor mountings. The network architecture is pretty similar to the previous one. The sensor data is gathered on peripheral sensor nodes, which then send the gathered information over the ZigBee mesh network to the gateway, which relays the data from the ZigBee network to the internet. There is a server, hosting a php script, which inserts all the data into a MySQL table. In [22] a very similar setup was built. They built a sensor network for agricultural monitoring using, Arduinos, Xbees and a Raspberry Pi. They had access to a LAN-network so the GSM part could be omitted. In return their sensor node is energy-wise self-sufficient and therefore truly wireless. Further they use Java as programming language where we use python. Figure 5 gives an overview of the networking devices.

![Network Architecture](image)

Figure 5: Network Architecture

3.3.1 Electronics

This subsection will show all the electronic devices developed for the sensor kit, namely the sensor nodes with sensors, hop nodes, the gateway and the g-value measurement contraption.

3.3.1.1 Gateway The device, called gateway in this report, mainly consists of an Raspberry Pi Model B, XBee module and an USB-GSM-modem. The Rasp-
3.3  Hardware Architecture

Raspberry Pi is an credit-card-sized single-board computer featuring a 700 MHz processor, 512 MB RAM, general purpose digital input and output pins, USB host ports and an ethernet port amongst other specifications. It is developed by the Raspberry Pi foundation as a platform for education. With a price tag of approximately 40 CHF it is very inexpensive. It is even cheaper than the Arduino Mega with the ethernet-shield used in the previous system, which cost combined around 90 CHF. The Raspberry Pi is not only cheaper, it also offers a lot more interfaces, memory and a lot more flexibility on the the software side.

An XBee module is connected via an ”Slice of Pi”-adapter-board\footnote{http://shop.ciseco.co.uk/slice-of-pi-add-on-for-raspberry-pi/}. The GSM connectivity is provided by an ”Huawei E303” USB-GSM-modem. The Raspberry Pi could still be connected to the internet via the ethernet connection, instead of the GSM-modem. Adding a WiFi-dongle to the setup would enable the gateway to even use on-site wireless LAN. Because the Raspberry Pi has power limit of approximately one amp\footnote{http://www.raspberrypi.org/documentation/hardware/raspberrypi/power.md}, the USB-GSM-modem is connected to a powered USB-Hub. The Raspberry Pi obtains its power from this USB-Hub as well, and has a second USB-connection for data transfer to the USB-Hub. This way only one wall adapter power supply is needed. For protection, the whole assembly was put inside an off-the-shelf plastic casing as it can be seen in figure\footnote{http://www.arduino.cc/} 6.

![Figure 6: Gateway with Sensor Node](image)

3.3.1.2 Sensor Node  The main components of the sensor node are an Arduino Pro Mini and an Xbee-module. Arduino\footnote{http://www.arduino.cc/} is an open source microcontroller platform consisting of hardware and software. A vast amount of software libraries
are available ready to use. There is also a huge online community build around this platform. This makes it a very good tool for rapid prototyping. The Arduino Pro Mini runs on 3.3 volts with an 8 MHz clock. Additional technical specifications can be found in table 4. The 3.3 volt version for the Arduino Pro Mini was chosen to have the same voltage level on the Arduino and the XBee-module and therefore no voltage-level conversion was needed. 3.3 volt operation also promises lower power consumption, which will become important for self sufficient power supply. Further there is a 2.1 mm center-positive power jack for power supply and a 4-bit rotary DIP-switch, which let’s the user configure sensor node type. Additionally, there is one LED to enable the sensor node, to give some feedback. To connect the actual sensor to the sensor node eight polarized connectors can be found on the circuit board as depicted in figure 7. The heat flux sensors and the pyranometer output a voltage in the micro-volt range which is proportional to the measured entity. The smallest voltage detectable by the 10 bit analog-to-digital

<table>
<thead>
<tr>
<th>Abbreviation / Symbol</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microcontroller</td>
<td>ATmega328</td>
</tr>
<tr>
<td>Operating Voltage</td>
<td>3.3 V</td>
</tr>
<tr>
<td>Digital I/O Pins</td>
<td>14 (6 PWM)</td>
</tr>
<tr>
<td>Analog Input Pins</td>
<td>8 (10 Bit ADC)</td>
</tr>
<tr>
<td>Flash Memory</td>
<td>16 KB</td>
</tr>
<tr>
<td>SRAM</td>
<td>1 KB</td>
</tr>
<tr>
<td>EEPROM</td>
<td>512 Bytes</td>
</tr>
<tr>
<td>Clock Speed</td>
<td>8 MHz</td>
</tr>
</tbody>
</table>

Table 4: Arduino Pro Mini Specifications
3.3 Hardware Architecture

The converter (ADC) of the Arduino is 3.23 mV (3.3V / 1023). This circumstance makes an amplification of the signal of the heat flux sensor and the pyranometer necessary. For this task an operational amplifier (op-amp) was chosen, specifically the ADA4528-2 by Analog Devices. A data logger, which came with the heat flux sensors, was using the same amplifier. By consulting its data sheet, we made sure that it is compatible with the microcontroller used in this project. In figure 8 the amplifier circuit is depicted. It is kept very simple, only consisting of the op-amp and two resistor for each input, which determine the gain of circuit. All the electronic schematics and circuit board layout were done in EAGLE 5.

The schematics can be found in the Appendix on page 82. The EAGLE-files were sent to Beta-Layout 6 who produced and assembled the circuit boards. The price was relatively high. But with higher volume and longer delivery times the price drops significantly.

3.3.1.3 Sensors

Besides the sensors used in the previous project, new ones, like pyranometer, heat flux sensor and current clamp, were introduced. All sensors used are shortly described in the following paragraphs.

**DS18B20** The DS18B20 programmable 1-Wire digital thermometer by Dallas Semiconductor was already used in the previous setup. It communicates via one-wire interface. Multiple Arduino libraries are available online. Only one 4.7k pull-up resistor is necessary as an additional hardware component. This sensor is used for the inside and outside temperatures of the u-value measurement, for the pipe temperatures and for the g-value measurement. It is available as a raw

\[ V_{\text{in}} \]  
\[ V_{\text{out}} \]  
\[ R_1 \]  
\[ R_2 \]

Figure 8: Amplifier Circuit

6 [http://www.pcb-pool.com](http://www.pcb-pool.com)
3.3 Hardware Architecture

electronic device and a waterproofed version is available as well. With 10 CHF for a waterproofed sensor it is still fairly inexpensive.

**TSL2561**  The TSL2561 light-to-digital converter by Texas Advanced Opto-electronic Solutions is, with 7 CHF, an inexpensive luminosity sensor. For this project a breakout version by Adafruit[^7] was used. This allowed to use the sensor without additional hardware components. To communication between microcontroller and sensor is established via I²C. Again, Arduino libraries are available from multiple sources.

**SHT15**  Sensirion’s[^8] SHT15 temperature and humidity sensor communicates over a 2-wire serial interface, which is not to be confused with I²C. It is similar, but not exactly the same. Using the breakout version from Sparkfun[^9], allowed, again, the use of the sensor without any additional components. This sensor is a high precision device, hence the price of about 50 CHF. The sensor is used to measure general air temperature and humidity.

**Heat Flux Sensor**  We chose to use heat flux sensors from greenTEG[^10]. The sensors output a voltage in the range of microvolts, which is proportional to the heat flux density through the sensor. This small voltage needs an amplification step to be detectable by de microcontroller. greenTEG offers amplifiers (150 CHF), but for the sake of compactness, we chose to use an op-amp on the sensor-node-pcb. The silicone surface of the sensor allows it, to compensate for the roughness of surfaces to be measured. It is important to note, that each heat flux sensor has a slightly different sensitivity. This issue has to be taken care of in the software for the calculation of the heat flux. With a price tag of about 400 CHF these sensors are one of the more expensive components. For the g-value-apparatus a XO-64-9C-sensor was used. For the u-value measurement a XO-66-9C-sensor with higher sensitivities were used. According to greenTEG the XO-66-9C-sensor has the same specifications as the XO-66-7C-sensor which is now listed on their web-shop.

[^7]: http://www.adafruit.com/
[^8]: http://www.sensirion.com/
[^9]: https://www.sparkfun.com/
[^10]: http://www.greenteg.com/
3.3 Hardware Architecture

**Pyranometer** The pyranometer MS-602 by EKO INSTRUMENTS[^11] outputs a voltage that is proportional to the incoming solar irradiation \([W/m^2]\). This voltage is, as with the heat flux sensors, very small and needs therefore an amplification. EKO offers an voltage to current converter. But for the sake of compactness we chose to use an op-amp on the sensor-node-board. The pyranometer is very sturdy and with a price of about 700 CHF a more expensive component of this project.

**Current Clamp** The current clamp is a very handy device to measure alternating current. It can be clamped around the live or neutral wire of a power cord without touching actual high voltages. It outputs a smaller but proportional current. In conjunction with a burden resistor the microcontroller is able to read a voltage, which is proportional the current in the power cord. With 20 CHF it is also an inexpensive component.

3.3.2 Housing and Mounting

The housing and mounting of various components is not that much of importance for the theory and calculations, but it is very important for the practical handling of the sensor kit, particularly for the installation and dismounting of the sensor kit on-site. The 3D-modelling was done in NX 8.5 by Siemens PLM Software. The physical objects were produced with Fused Deposition Modeling (FDM) 3D-printer. The printer was an Ultimaker[^12]. The models generated in NX were exported as STL-files. These STL-files were then fed into Cura. Cura is a software that came with the 3D-printer. It converts the STL-files into g-code. G-code-files contain commands for the 3d-printer actuators. Green PLA was used as printing material.

3.3.2.1 Sensor-Node The previous setup used an off-the-shelf-casing for the sensor-node. For the sake of compactness a custom housing was created. For the housing of the printed circuit board (PCB), the PCB itself was first modelled and later the casing around it. The casing consists of on upper and an lower part, which snap together with latches. There are openings for the plugs, LED,

[^12]: https://www.ultimaker.com/
antenna and access to the rotary dip-switch. The PCB sits on four tiny distance holder placed in all four corners of the lower half of the case. To secure the PCB from moving another distance holder in the top half of the case holds down on the housing of the power jack of the PCB, making it impossible for the PCB to move vertically. Horizontal movement is inhibited by a tight fit of the case-walls to the PCB outline. In case of the need of watertightness, the whole node, including the housing, was put into an off-the-shelf watertight casing.

3.3.2.2 TSL2561 The casing of the luminosity sensor provides protection to the electronics, a way of mounting and a visual face-lift in order to make it less bothersome for occupants. The housing consists of a upper and lower half, which snap together with latches. In the lower half are four distance holder which can receive the break out board. There are openings for the light to enter and for the cable. The sensor can again be mounted with the application of adhesive.

3.3.2.3 SHT15 The casing of the temperature and humidity sensor was also needed to give the sensor more visual appeal, protection and an mean of mounting. There are ventilation slits to allow air circulate around the sensor and an additional opening for the cable. The two halves of the housing snap together with latches and the lower half is outfitted with distance holder, which receives the sensors break-out-board. For mounting purposes adhesives are suggested.

3.3.2.4 DS18B20 For the DS18B20 temperature sensors need to be exposed to the air for the u-value measurement. The version of the sensor, that is used in this project is already waterproof and pretty sturdy. But to make the sensor stick out from the wall, a bracket was constructed. With cable ties the sensor can be attached to the bracket. The backside of the bracket is kept flat so that it can be mounted to walls with adhesives.

3.3.2.5 Pyranometer Two mounting options are available for the pyranometer. We successfully mounted a pyranometer with Tesa® Powerstrips®[13] The Powerstrips held on well under exposure to the weather. More on adhesive testing can be found in subsection 3.3.2.9 on page 29. The second option to mount the pyranometer is using a suction cup. For this we used a suction cup with a

pump manufactured by Manfrotto\textsuperscript{14}. This suction cup is intended as accessory for photo and video equipment. In order to connect the pyranometer and the suction cup, an adapter-plate was 3d-printed. This plate bolts to both, the pyranometer and the suction cup. The suction cup can be placed on the window and by pressing the pump air is removed from underneath the rubber cup. To avoid injuries by a falling suction, it was secured with a piece of cord to the nearby hinge of the window. The suction cup bear the weather well. It was mounted to a south facing window (HPZ G24) from the June 19, 2014 to August 11, 2014. The windows surface was not cleaned before applying the suction cup. The pump has only been used once to build up the initial underpressure.

\subsection*{3.3.2.6 Hop-Nodes} The previous setup had already housings for the hop-node. It was a compact off-the-shelf case. For more visual appeal and consistency a custom housing was made. It is again a two-piece that snaps together with latches. There are openings for the indicator LED, antenna and the power cable. The hop node casing from the predecessor system was smaller than the current one.

\subsection*{3.3.2.7 Gateway} The gateway assembly consist off several loose parts (USB-hub, Raspberry Pi, USB-GSM-modem), of which only the Raspberry Pi has dedicated mounting holes. This fact and the bigger size needed for the gateway housing lead to use an off-the-shelf casing for gateway assembly. Distance holder on the inside of the casing were removed, for more space, and a hole for the power cords was drilled into the side of the case. Then everything could carefully be put inside. The case snapped together with latches.

On one test site the GSM connection broke during the measurements due to bad reception, most probably caused by the concrete structure of the building. This forced us to place the modem outside of the building. For this reason the modem was packed into an off-the-shelf watertight casing. The cable feedthrough through the wall of the casing, was secured and sealed against water with glue. An USB-extension cord then connected the modem with the Raspberry Pi, which was placed inside the building.

\footnote{http://www.manfrotto.ch/}
3.3.2.8 **Cables** To connect the peripheral sensors to the sensor node, cables are needed. To reduce the visual impact of the whole equipment very thin white cables would work the best. Since the sensors and the node are often placed on or close to walls, which often happen to be grey or white in buildings. Such cables are available, but they do not come cheap. greenTEG offer the cables, that they use for the heat flux sensors, for about 12 CHF per meter of cable. The cables are shielded and available with two and four conductors. With an outer diameter of less than 1.5 mm, these cables are ideal from the visual aspect. Another cable supplier of very thin cables could be located. But the concerned company failed to answer our request. However, because of the price, we decided to use common shielded cable from the university’s supply store. The cable come with two or four conductors and have an outer diameter of 3 or 5 mm. With a meter price around 1 CHF, these cables are much more inexpensive. Shielded cables were chosen to avoid issues with sensor-microcontroller-communication over longer distances.

To get the cables out of the way, adhesive bases were used. To this bases the cables could be attached with cable ties on one side, and the other side could be stuck to a flat surface. To ensure the integrity of the surface, on which the base was adhered, a special double sided adhesive strip was inserted between base an surface. Alternatively a adhesive plasticine could be used. More details on adhesives can be found in section 3.3.2.9.

3.3.2.9 **Adhesives** It was right away obvious, that there will be some cabling and cases that has to be taken care of by carefully securing or hiding. Our approach was to use adhesives. However, applying adhesive successfully is one part, removing it without leaving traces another. Since it is intended to use the sensor kit in inhabited buildings, application and removal of adhesive are of importance. For this issue several adhesives were mustered, namely Tesa® Powerstrips®, UHU® patafix and Pattex® Strips. These products were either applied directly to the respective case and subsequently the case was adhered to a surface or the products were used in combination with the adhesive bases, described earlier, to secure cables. In order to get an impression on the performance of the adhesives they were tested an indoor concrete wall, indoor plaster wall and an outdoor concrete wall. The cables used for the adhesive test were the same used to connect the sensors with the sensor node. One cable segment measured approximately
135 cm and weighed approximately 55 grams. Outdoor and indoor on the plaster wall the cables were aligned horizontally and vertically on the indoor concrete wall as depicted in figure 9. The tests were conducted in May and June 2014. If the cables fall off, they were remounted using new adhesives. If the cables came quickly, the test was aborted. Details on how long the adhesives worked and how often they had to be reinstalled can be found in table 5.
<table>
<thead>
<tr>
<th>Nr.</th>
<th>Adhesive</th>
<th>Base</th>
<th>Surface</th>
<th>Duration</th>
<th>Reinstallation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tesa Powerstrip</td>
<td>Yes</td>
<td>Plaster</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Pattex Strip</td>
<td>Yes</td>
<td>Plaster</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>UHU patafix</td>
<td>Yes</td>
<td>Plaster</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>UHU patafix</td>
<td>No</td>
<td>Concrete Indoor</td>
<td>47</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>UHU patafix</td>
<td>Yes</td>
<td>Concrete Indoor</td>
<td>47</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>Pattex Strip</td>
<td>Yes</td>
<td>Concrete Indoor</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Tesa Powerstrip</td>
<td>Yes</td>
<td>Concrete Indoor</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Tesa Powerstrip</td>
<td>Yes</td>
<td>Concrete Outdoor</td>
<td>14</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>UHU patafix</td>
<td>Yes</td>
<td>Concrete Outdoor</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>UHU patafix</td>
<td>No</td>
<td>Concrete Outdoor</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Pattex Strip</td>
<td>Yes</td>
<td>Concrete Outdoor</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>UHU patafix</td>
<td>No</td>
<td>Concrete Outdoor</td>
<td>14</td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Adhesive Test Overview
3.3.3 G-Value Measurement Device

For the measurement of heat flux through the window for the g-value a special device was built. It is heavily inspired by the work of [23] and [24]. The idea was to build a device working by the same principle and make it more portable and integrate it into the sensor kit. In [23] the device is called fenestration radiometer. The device allows sunlight to enter through a window into an insulated box, that was painted black inside. On the backside of the box, heat flux meters were installed along with a cooling system, which allowed heat to be pulled out of the box through the sensors. In order to get the solar heat gain factor, the global solar radiation was measured with a pyranometer placed in the same plane as the window of the box. Dividing the cumulative heat flow through the sensor by the cumulative solar irradiation leads to the g-value. The power added by the peltier element can be omitted, since it is inserted after the sensor. Two cooling systems were tested: convection cooling with a fan and water cooling with an thermostatic bath. In [24] the system was similar to the previous one. The cooling system was replaced by a peltier element. And a fan was placed inside the box to force a more uniform temperature distribution. They also varied the size of the box.

To make it more portable, we shrunk the device to a 5 cm by 5 cm inlet. The inner box is made out of copper and painted black. The absorptivity of the paint is unknown. Around four sides of this copper box 5 mm of styrofoam was put in place. A DS18B20 temperature sensor was embedded in the styrofoam, such that it is in contact with the copper box. On the backside of the box, a greenTEG heat flux sensor was sandwiched between the box and a 4 cm by 4 cm peltier element. On the free side of the peltier element a heat sink and a fan were mounted. The remaining free surface on the back side of the box was covered with layers of cork. A casing enclosing the box with insulation, was 3D-modelled in NX and 3D-printed. The whole contraption could then be adhered to a window with adhesives on the plastic casing. To control the peltier element an extra electronic circuit was built on a perfboard. The schematic can be found on page 82 in the appendix. The circuit basically consist of an Arduino Mini for control, a power transistor to control the peltier element and two DS18B20 temperature sensors to keep the box temperature close to the room temperature. The circuit is powered by a 12 V / 65 W power supply. The fan for the power transistor and the fan for the measuring contraption are directly powered by the power supply. There
is no dedicated control of the fans. To keep the two temperatures as close as possible a bang-bang-control was tried first and later it was upgraded to a PID control in order to reduce the peaks of the heat flow through the heat flux sensor. To measure the solar irradiation a pyranometer was mounted on the windows as described on page 27.

3.4 Software Architecture

This subsection describes all the software needed to measure the physical entities and transfer this measurement data to the MySQL-table.

The software architecture of the whole kit, was kept as simple as possible, in order to keep it maintainable by one person and to allow a trouble-free project transfer. A C++ script running on the sensor node was in charge of interfacing with the sensors and to send the measured data to the gateway. The python script, running on the gateway, received the data, rearranged it and sent it to the server in the internet, which was hosting a PHP-script. This script then inserted the received data into the MySQL-table. The data could then be used by any software capable of interfacing the MySQL database.

![Software Overview Diagram]

Figure 10: Software Overview
3.4.1 Arduino

On each sensor node a C++ script runs on the Arduino. This script can be found in the appendix. In the first line of code the address of the Xbee of the gateway has to be defined. This is the only configuration that has to take place before the compilation of the code. After compiling the Arduino-code the node can only be used with the corresponding gateway.

In the setup-function all initializations take place. Firstly the serial connection is initialized followed by the Xbee library and the GPIO pins of the rotary switch. Then, according to the node type, which is set by the rotary switch, necessary initializations for the attached sensors are performed. At the very end of the setup-function the LED is initialized and flashes as many times, as the position of the rotary switch indicates.

In the loop-function the necessary sensors are read out, according to the pre-set node-type, and the obtained values are bitwise arranged into the payload and subsequently sent away to the gateway. At the end of the loop function a pause in form of a delay function, is set in place, in order to wait a predefined time before the loop function is called again.

3.4.2 Raspberry Pi

As operating system on the Raspberry Pi, Occidentalis by Adafruit was used. It is derived from Raspbian Wheezy, which, in turn, is based on Debian Wheezy. The operating system was modified so that the file system is mounted with read-only privileges. This was necessary, because of problems that occurred while running the system over longer periods. It then would happen that the SD-card got corrupted and the whole system stopped working. The exact mechanism behind this faulty behaviour could not be determined. Because of this issue an additional USB-memory stick was used to save the backup file, instead of saving the backup data on the SD-card.

An entry in the /etc/rc.local file allowed the autstart.sh script in the Gateway-directory to be executed. Which in turn initiated the internet connection over the GSM modem. Then, an additional entry in the /home/pi/.bashrc file allowed the python script, which interfaced with the Xbee-network and the internet, to start and be displayed on the console. Because we experienced the losses of the internet connection without successful reconnection, a cronjob was set into place.
The cronjob restarts the Raspberry Pi every day at 12.00. This allows the USB-GSM-modem to power off and on again. The exact cause of the connectivity problem is not known and was not further investigated. Since the Gateway only transfers the data received on the Xbee-Network to the internet, the python script could be kept short and simple. Notably no configuration concerning the sensor nodes are necessary. The script is written in Python. After the initialisation of the the libraries and the serial connection a while-true-loop contains the rest of the code. After receiving a Xbee-data-package the length of the package is checked. Then the payload gets unpacked into meaningful values. These values then get assembled into an URL, which can be called in order to transmit the data to the server. As a data-safety measure, the values transmitted with the call of the URL, also get written into a local text file on a separate USB memory stick. This way in a case of a internet connectivity problem, the data can be retrieved later, and the measurement is not lost. Everything inside the while-true-loop is embedded into a try statement. If anything goes wrong inside this statement, the while-loop just restarts and nothing is blocked. The python script can be found in the appendix on page 67. For interfacing the usb-gsm-modem we made use of umtskeeper\footnote{http://mintakaconsciencia.net/squares/umtskeeper/index.html} and Sakis3g\footnote{https://github.com/Trixarian/sakis3g-source}.

### 3.4.3 Server

The script takes the values transmitted in the URL, and inserts them into a MySQL-table. The script is written in PHP. First data values get written into a text file as another backup. In case of a failure of the MySQL-server. The data can be retrieved from this server side backup file or from the backup file on the Raspberry Pi. Then a connection to the MySQL-database is established and the values are inserted into the according table. The script can be found in the appendix.

### 3.4.4 MySQL

phpMyAdmin was used to set up and manage the MySQL tables. The database is designed to be relational. The central table to store the measured sensor values is the table mf\_measurement. In this table the mf\_measurement\_id, time,
node_id, measurement_type and the measurement_value are written to. The measurement_id is the unique key for each measurement. The data stored in this table is later accessed with Matlab for further processing. The tables mf_node and mf_node_type are used in the php script to relate the Xbee-address to the node_id, which is handier to use, and the node_type to the type of measured data. The last table mf_measurement_type is not yet used in the processing of data. The table describes the measurement_type_ids in human readable words. Following are the MySQL tables with the according columns listed: Figure 11 shows the relationship of the MySQL tables.

```
<table>
<thead>
<tr>
<th>Table</th>
<th>Columns</th>
</tr>
</thead>
<tbody>
<tr>
<td>mf_measurement</td>
<td>- measurement_id</td>
</tr>
<tr>
<td></td>
<td>- time</td>
</tr>
<tr>
<td></td>
<td>- node_id</td>
</tr>
<tr>
<td></td>
<td>- measurement_type</td>
</tr>
<tr>
<td></td>
<td>- measurement_value</td>
</tr>
<tr>
<td>mf_node</td>
<td>- node_id</td>
</tr>
<tr>
<td></td>
<td>- xbee_address</td>
</tr>
<tr>
<td></td>
<td>- node_description</td>
</tr>
<tr>
<td></td>
<td>- room_id</td>
</tr>
<tr>
<td>mf_node_type</td>
<td>- node_type_id</td>
</tr>
<tr>
<td></td>
<td>- value1_type_id</td>
</tr>
<tr>
<td></td>
<td>- value2_type_id</td>
</tr>
<tr>
<td></td>
<td>- value3_type_id</td>
</tr>
<tr>
<td></td>
<td>- value4_type_id</td>
</tr>
<tr>
<td></td>
<td>- node_type_description</td>
</tr>
<tr>
<td>mf_measurement_type</td>
<td>- measurement_type_id</td>
</tr>
<tr>
<td></td>
<td>- measurement_type_description</td>
</tr>
<tr>
<td></td>
<td>- measurement_type_unit</td>
</tr>
</tbody>
</table>
```

Figure 11: Overview MySQL Tables

More tables were created, in order to be able to manage multiple clients and
objects. But the use of these tables is not necessary for the pure measurement of building properties and is out of the scope of this thesis. Therefore, these tables will not be presented in this report.

3.5 Case Study

3.5.1 Procedure

The general procedure of the installation of the sensor kit consists of an initial inspection of the site, setting up the sensor kit, inspection of the installed sensor kit and the dismounting of the whole kit.

During the first inspection of the place, one had to find appropriate location to place the sensors. The availability of a nearby electrical outlet had to be assured. For this, we used to have a floor plan, where we could note the electrical outlets and other remarks. The biggest concern was to find a place to put all the cabling, without bothering the residents and minimizing the possibility, that the cables were torn down. We also tested the GSM reception on-site to avoid connectivity troubles later. After the inspection, the cables could be prepared for the installation. This means, to assure that the power cables and sensor cables are long enough for the task. Sometimes the cables had to be passed through a window, e.g. for the measurement of an outdoor temperature measurement. In this case, the cable was cut, and a piece of ribbon cable was soldered in place, in order to allow the window to be closed. Usually, after the kit was prepared for deployment, it was first tested in the office over several days, to ensure all everything works as intended. Arriving with the prepared sensor kit, the Raspberry Pi gateway was installed first. Then the sensors, which were fixed rather fast to the proper location. Tucking away all the sensor cables and power cables took much longer. The hop nodes were installed along with the sensors. The hop nodes have a dedicated LED to indicate, if they are connected to the coordinator Xbee. This makes it easy to see, if the hop node is in a proper location, or if it has to be moved somewhere else. With the gateway installed first and already running, the database can be checked for incoming measurement data, indicating the proper operation of the system. The most time was spend to tuck away all the cables. In average it took about 70 minutes for a sensor node to be installed. This includes the gateway, but not the hop nodes, since most hop nodes were ad-
pered to their transformer and didn’t add much additional work. On the first try, there were problems with the GSM connectivity and corruption of the SD-card. This is why we had to return to the site one week later. After that experience, we budgeted a visit to check the proper operation of the system on-site for upcoming measurements.

After the measurement was completed the sensor kit had to be collected. This could be done much faster than setting up the system. Because of the proper choice of adhesives, all the cables and their fastener could be removed easily. Back in the office the fasteners, adhesives and cables were separated, cleaned and prepared for the next engagement. Overall, the dismantling took less than 20 minutes per node, including the gateway, not including the hop nodes.

3.5.2 Measurements

In this section the measurement of u-values and supply and return temperatures are presented.

3.5.2.1 Background Information  The object where the measurement was performed is an apartment building in the vicinity of Zurich. It was built in the 1970’s. In 2006 the insulation of the envelope has been improved. The owner of the building plans on further improving the building performance. In the planning process of this alternation a simulation on the energy consumption was done. The result of this simulation and the recorded oil consumption data are in figure 12 depicted. Between 2006 and 2007 a drop in the oil consumption, due to the mentioned retrofit, can be observed. Further, notice, that the simulation predicts an
energy consumption, that is only half of the actual energy consumption. Because of this mismatch between reality and simulation, we got the opportunity to investigate by measuring the u-value. Supply and return temperatures were measured as well. In figure 13 the positions of the u-value measurements are marked with red dots. The u-values were measured three times on walls to the exterior, once at the ceiling to the roof and once at a roller shutter casing. The case study took place in September and October 2014.

Figure 13: Floor Plan with Sensor Positions
3.5.2.2 U-Value  For the measurement of the u-value two temperature probes and a heat flux sensor are necessary. The temperature sensors were mounted on the inside and outside of the building element, usually a wall. This was done with the use of the 3d-printed brackets and adhesives. The heat flux sensor was mounted next to the temperature sensor on the inside of the building element. The heat flux sensor was adhered with special thermoconducting double-sided tape. For more security the heat flux sensor was additionally adhered with electrical tape. The placement of the sensors should be reasonably far away from the borders of the wall, in order to minimize influence of the boundaries. An exemplary set-up on an inside of a wall can be seen in figure 14.

![Temperature and Heat Flux Sensor mounted on a Wall](image)

Figure 14: Temperature and Heat Flux Sensor mounted on a Wall

3.5.2.3 Supply and Return Temperature  When we got an opportunity to test the sensor kit on-site, we where asked, if we could also measure the supply and return temperature of the pool heating system. For this task, a new node type was added to the MySQL table, which expects temperature readings from two DS18B20 temperature sensors. This allowed the sensor kit to only measure the two temperatures without any further modification. The temperature sensors were taped to the pipes and covered with 2cm of insulation. The mounting of the temperature sensors to the pipes is depicted in figure 15. The measured data was handed over to the commissioning consultants and was not further processed by us. However, in [25] a method, to estimate the water flow by cross-correlation of temperature variations, is described.
3.5 Case Study

3.5.3 Results

Figure 15: Temperature Sensors mounted to the Pipes

Figure 16: U-Value Measurement: Interior to Exterior 1 (U1)
3.5 Case Study

Figure 17: U-Value Measurement: Interior to Exterior 2 (U2)

Figure 18: U-Value Measurement: Interior to Balcony (U3)
3.5 Case Study

Figure 19: U-Value Measurement: Interior to Roof (U4)

Figure 20: U-Value Measurement: Roller Shutter Housing (U5)
3.5.4 Evaluation

Figure 23: the u-values assumed for the simulations are represented by the blue bars. For the external walls and for the roof, all four measured u-values exceed the assumed value. Hinting that differences between the assumed u-value and the true
may be one cause of the difference between the simulated energy consumption and the measured one. Counting, the area on the wall covered by the roller shutter housing, as window area, the assumption matches the measurement quite good. But if the area covered by the roller shutter casing, was assumed to perform like the rest of the wall, the mismatch would be the greatest in this case. The errors were calculated using the procedure in section 2.2.

Figure 23: Comparison U-Values: Assumptions vs. Measurements

3.5.5 G-Value

The g-value apparatus was tested on an window in the office, not on a actual site. The pyranometer was mounted on a suction cup, which in turn was attached to the outside of the window. On the inside of the window, the thermal box was adhered to the window with double-sided tape. The set-up is depicted in figure 24. The initial approach was, to control the temperature inside the box with a bang bang control approach. This way the peltier element was only turned on or off, if the temperature difference between the temperature inside the box and outside of the box got big. As can it can be seen in figure 25, the setup was able to keep the the temperatures close. This means that peltier is powerful enough to perform as intended. But as we discovered in figure 25, the generated heat flux through the heat flux sensor exceeds the measurement range. This means heat fluxes beyond 650 W/m² can not be measured with the current arrangement.
of the heat flux measurement. This means that the g-value calculated from the measurement is meaningless.

In order to improve the situation with the heat flux sensor limit, the bang bang controller was replaced by an PID controller. The PID parameters were chosen experimentally. In consequence the temperatures are closer together than before and the controller action got smaller, as it can be observed in figure 27. But the heat flux sensor still reaches its limits, although less often, and the calculated g-value is still not usable.
A potential solution would be to change the op-amp arrangement with the resistor. But this would mean that a node only could be used to either measure the g-value or the u-value, but not both. Because of the time limitation of this thesis, it was decided, to not further improve the g-value apparatus in the scope of this thesis.
3.5 Case Study

Figure 27: PID Control of G-Value Apparatus

Figure 28: G-Value Measurement with PID Control
4 Summary and Conclusions

This section will summarize the work, that was done in the frame of this thesis. Achievements, limitations and next steps will be discussed.

4.1 Summary of Achievements

In this thesis the methodology of [1] was extended by including the u-value and g-value of windows. With the additional variables the final equation for the retrofit supply temperature got more complex and less convenient to handle. The numerical example has shown that the influence of the g-value is minor.

Further, an iteration step was performed on the previous measurement hardware kit of [1]. The sensor nodes became smaller and more versatile. Namely, different combinations of sensors can now be attached to the same sensor node unit. Also the number of different sensors has been increased. In particular, it is now possible measure heat fluxes and solar irradiation. With the GSM connectivity the sensor kit is independent of on-site internet access, like ethernet or Wifi. Besides of the achievements in theory, hardware and software a lot of experience could be gained in the field of energy and buildings, electronics design and manufacturing, wireless communication, software development, as well as in practical measuring.

4.2 Limitations, Lessons Learned

The main limitations of the current system is the gateway and the cables. The SD-card corruption issue was a big problem and the loss of internet connectivity was another big and issue.

The problems with SD-card corruption were rather unexpected. With a corrupted SD-card the whole Raspberry Pi is rendered useless. Even the data backup on the SD-card or an USB-stick doesn’t work anymore. Rebooting the system doesn’t work ether, once the SD-card is corrupted. With the gateway at the heart of the system, it is very important, that it performs reliably. The combination of changing the SD-card brand, mounting the file system read-only and logging on a USB-memory stick solved the problem. We expected to lose the GSM connectivity now and then, but the long down time and the need of a reboot was a surprise. However, this issue can be and must be covered with alternations to
the software. Since one can not rely on the internet connection at all, the data has to be timestamped on the gateway and only sent, when the internet connection is up.

During the hardware design, we had mainly dry rooms like offices, living rooms and alike in mind. But the first request for a measurement, was to measure the u-values in a room with a swimming pool, which was rather warm and humid. In order to prohibit complications, the electronics were enclosed in a water tight housing. Although, this can be taken as an odd request, placing sensors out side prone to the weather is not so unusual anymore. This and for a general increase in robustness, it is desirable to make the sensors and the sensor-node to some degree water resistant.

For the setup of the measurement, a lot of time was spent on taking care of the cables, including: measuring how long all the cables had to be, where a exactly a ribbon cable had to be inserted, such that the cable could be routed through a window, finding electrical outlets near by and tucking all the cables away. In addition the cable tangle doesn’t look nice. This is why, it seems worth to invest some effort on improving the situation with the cables.

The sensitivity of the heat flux sensor advertised on the homepage of the manufacturer was $1.5\mu V/(W/m^2)$. The calibration certificate of the first sensor ordered, revealed that the sensitivity of this particular sensor was $2.4\mu V/(W/m^2)$. Later five more sensors were ordered. The latter five sensors exhibited a sensitivity of around $10\mu V/(W/m^2)$. We then learned, that the manufacturer had improved, among other things, the sensitivity and the old sensors were not available anymore. Higher sensitivity is a good thing and the big change in sensitivity could be adjust by changing the gain of the op-amp by changing the resistor. But the small differences among the sensitivity of each sensor, were not considered during the design phase. Because of this, it always had to be noted which heat flux sensor was connected to which sensor node, in order to be able to correct for the different sensitivities in the software. This small variation of the sensitivity are present on the pyranometer as well.

The Matlab script for the average method for u-value measurements, works as intended and the results are similar to the ones found in the papers. The u-value calculated with the average method, only using values during the night, usually yield lower results, which is expected. The output of the Matlab script for the
dynamic method for u-value measurements is not yet satisfying. No convergence is visible. Instead it fluctuates a lot.

4.3 Outlook, Impact, Future Work

The gateway most of all has to become more reliable in operation. Because of its central position of the data flow from the sensor to the database, it is very important, that it performs reliable. For this, the Python script should be extended to handle internet connection outages and only send data, when the connection is alive. The SD-card corruption issue seems to be resolved with the action taken to counter it. To further secure reliable operation, a hardware switch could be added, with which a proper shutdown can be triggered.

We added GSM connectivity for monitoring and remote maintenance. Over the course of this thesis, it caused a lot of problems and remote maintenance could not be done easily, because of the private IP of the USB-GSM-modem. However, if it wasn’t for the GSM-connection, we would have returned to the site, learning that the SD-card got corrupted and would not have any data from measurements. In the development phase it is still advantageous, even with the drawbacks. With a reverse SSH-tunnel or by asking the internet service provider for a public IP and with a better handling of GSM connection, it is possible to overcome this obstacles. When the system advances more in the development and works reliable, software update should not be needed anymore, in this case the internet connectivity still offers some advantages. It can be checked for damages of sensor nodes, for example a node could fall off or could be torn down. In the case of no internet connection or a simple data logger, one finds out about this only, when arriving to collect the system. Alternatively, one has to go by the side to inspect all the elements. With internet connectivity, the processing of data can be done from the beginning of the measurements and the measurement can be stopped as soon as the demanded criteria are met. Also the recollection of the sensor-kit and the data processing can be done independently. It is possible to deliver the processed data, without collecting the sensor-nodes, which shortens the time from the deployment of the sensor-kit to the delivery of the processed data.

It is worth improving the situation on the cable tangle. This matter can be divided into two parts: sensor cables and power supply. For the sensor cable
it seems enough to replace the current cables by a very thin one, which can sit between the window and its frame, without getting damaged, damaging the window or the need to insert a ribbon cable. Such cables are available, but are much more expensive than the regular cable, used in this setup. However, with the experience gained, the saved money is not worth the trouble caused by the thicker cables. In order to get rid of the power supply and its cable, each sensor node has to become self-sufficient energy-wise, making the system truly wireless. Having switched from a pull to a push communication scheme, already allows the sensor node to be put in sleep-mode, between sending measurements. Considering two AA 3.6 volts / 2500 mAh lithium batteries, holding 32400J of energy each, the current sensor node would only last for 15 days. This is because the Xbee is always on and consumes about 70 percent of the total energy, assuming an on-time of 15 seconds and off-time of 285 seconds. Enabling the microcontroller to switch of the Xbee and putting the microcontroller into a sleep mode, this modified setup is expected to run for 66 days. This calculations were performed for the u-value setup with the heat flux sensor and two DS18B20.

It has to be noted, that it is unlikely to make the gateway and the hop nodes self-sufficient. Since they have to be up all the time. Further a GSM-modem consumes much more energy than a Xbee-module. But since the placement of the gateway and the hop nodes is not that critical, they can be placed just beside the electrical outlet, without the need of an extension cable. Changing to a more energy efficient components, namely microcontroller, wireless communication module and power regulator, may improve the battery lifetime further.

The electronics used in this thesis consisted mostly of off-the-shelf through-hole parts. That is, why there is still a great potential for shrinking the electronics footprint even further by consistently using surface mount technology.

When preparing multiple sensor kits with different gateways, the Xbees had to be loaded with a different profile for each gateway and its according sensor nodes and hop nodes, since only one coordinator per Xbee network is allowed. Otherwise the sensor kits could not be tested simultaneously before deployment. This is very cumbersome. That is, why a more automated node-gateway-association is desirable.

Currently, there is no detection of system failure in place. To see if the system was alive, the time of the latest MySQL entry was checked. If this time was older
then 5 minutes, a failure was assumed. A system health check could spare manual checking of the MySQL table, and further check, if each node is still active. And subsequently, the health check system, would notify with a health report of the system, only in the case of a failure.

All the communication performed in the scope of this thesis, was done without any encryption or other security measures. The data measured are not personal and manipulation of third parties was not suspected. Further, dealing with security of communication would have been beyond the scope of this thesis. However, it is recommend to evaluate the situation on the aspects of security of communication.

The data analysis performed in this report was done in Matlab, with some manual inputs. Which is fine for the development phase of the project. But as the project progresses, it is desirable to have a more user-friendly interface, which is able to associate the nodes to rooms and objects and performs the data analysis automatically for each room without further manual input. Also the implementation of the the dynamic analysis method for the u-value measurement is not yet satisfying. It is advised to have a second look on the current work, in order to improve the results.

For the g-value setup, it is suggested to take a step back and build a prototype that is working in the lab without being tied to the sensor network nor considering size or noise constraints. Once the prototype is working in the lab and validated, it can be reintegrated in the sensor network for on-site use. As the theory suggests the influence of the g-value is minor, so this topic is of minor importance.

With the request for measurements, it was also asked to measure the supply and return temperature and the mass flow of water. As seen in the theory section, some effort was put into substituting the mass flow in the equation. Thus, it may simplify things, if a non-intrusive water flow meter could be added to the sensor kit.

As it was learned during the development of this sensor kit, sensors may exhibit individual specification or need different amplification for different application and additional sorts of sensor may be needed to complete the sensor kit. This is why we suggest to divide the sensor node into two parts as depicted in figure 29. One part takes care of the wireless communication and power management, while the second part is permanently attached to the sensors and takes care
of reading the sensor values and setting the node type. The first part supplies the second part with energy and receives the measured values from the second part. This way, one would only have to deal with the individual characteristics of the sensors while programming the second part. This proposal bring some advantages over the current system. It allows to develop the first part independent of the second part. One could even incorporate the communication and power management from other projects. Further, in order to change or add sensing hardware, it is required to change the hardware of the whole sensor node, possibly making different versions incompatible with each other. With the newly proposed concept, changes only apply to the second parts used with the concerning sensors. And for new sensors only the second parts must be freshly developed. This implies that the communication protocol between the first and the second part is designed to handle the changes of the second parts.

Figure 29: Next Generation
References


References

of the 4th World Conference on Structural Control and Monitoring (4WC-SCM), pages 11–13, 2006.


A Source Code

A.1 Node Arduino Script

```c
#ifndef gatewayAddress 
#define gatewayAddress 0x408C0EFD // Lower part of 64-bit Xbee-address of the gateway

int nodeType = -1; // Node type, can be changed with rotary DIP-Switch

int measurementInterval = 5*60; // Time between two consecutive measurements [seconds]

typedef union _data { float f; byte b[4]; } myFloat; // Data structure to help splitting float values into single bytes for transmitting purposes

myFloat sensorValue1; // Variable to temporarily store sensor value before inserting it into payload[]
myFloat sensorValue2; // Variable to temporarily store sensor value before inserting it into payload[]
myFloat sensorValue3; // Variable to temporarily store sensor value before inserting it into payload[]
```
myFloat sensorValue4;
    // Variable to temporarily store sensor value
    before inserting it into payload[]

byte payload[18];
    // Payload for transmitting via xbee

// Pin Assignments
    // Digital Pins
    // XbeeRxTx1 0
    // XbeeRxTx2 1
#define SHTdataPin 2
    // SHT15 Data Bus
#define SHTclockPin 3
    // SHT15 Clock Bus
#define DS18B20Pin 4
    // One Wire Bus
#define pulsePin 5
    // Pulse counter input, attached to interrupt
#define switchPin1 6
    // Rotary DIP switch input
#define switchPin2 7
    // Rotary DIP switch input
#define switchPin4 8
    // Rotary DIP switch input
#define switchPin8 9
    // Rotary DIP switch input
#define ledPin 10
    // Signal LED output pin
#define CurrentClampPin 0
    // Current clamp analog input pin
#define heatfluxPin 1
    // Heatflux sensor analog input pin
#define pyranometerPin 2
    // Pyranometer analog input pin
#define AnalogPin 3
    // Available
// TSL2561_SDA 4 // I2C Bus
// TSL2561_SCL 5 // I2C Bus
// 6 // Not yet assigned / still available
// 7 // Not yet assigned / still available

void setup()
{
  Serial.begin(9600); // Does not work when wrapped in initCommunication
  initCommunication;
  initSwitch();
  nodeType = readSwitch();

  switch (nodeType)
  {
    case 0: // Hop–Node
      break; // nothing to initialize
    case 1: // Comfort
      break; // nothing to initialize
    case 2: // Luminosity
      initLuminosity(); // initialize TSL2561 luminosity sensor
      break;
    case 3: // Radiator
      initTemperature(); // initialize DS18B20 temperature sensors
      initHeatflux(); // initialize heat flux sensor
      break;
    case 4: // U–Value
      break;
  }
}
initTemperature();  // initialize DS18B20 temperature sensors
initHeatflux();  // initialize heat flux sensor
break;
case 5:  // U-Value 1 (indoor)
    initTemperature();  // initialize DS18B20 temperature sensors
    initHeatflux();  // initialize heat flux sensor
    break;
case 6:  // U-Value 2 (outdoor)
    initTemperature();  // initialize DS18B20 temperature sensors
    break;
case 7:  // G-Value
    initTemperature();  // initialize DS18B20 temperature sensors
    initPyranometer();  // initialize pyranometer
    initHeatflux();  // initialize heat flux sensor
    break;
case 8:  // Current
    initCurrent();  // initialize current clamp
    break;
case 9:  // Oil
    initPulse();  // initialize pulse counter
    break;
case 10:  // Not defined
    break;
case 11:  // Not defined
    break;
case 12:  // Not defined
    break;
case 13:  // Not defined
    break;
case 14:  // Not defined
    break;
case 15:  // Not defined
    break;
default:
;
}
initLed();          // Initialize LED
flashLED(2);       // Flash LED twice to signal completed setup routine*/
Serial.println("\n\nSetup Complete");

void loop ()
{

  // Reset payload and sensorValue-variables
  for (int i = 0; i < sizeof(payload); i++)
  {
    payload[i] = 0;
  }
sensorValue1.f = -10000;
sensorValue2.f = -10000;
sensorValue3.f = -10000;
sensorValue4.f = -10000;

  // Listen for request from Gateway XXX

  // Get sensor data
  switch (nodeType)
  {

case 0:       // Hop−Node
   break;

case 1:       // Comfort
   sensorValue1.f = readComfort('t');    // float
   sensorValue2.f = readComfort('h');    // float
   break;

case 2:       // Luminosity
   sensorValue1.f = readLuminosity('f'); // integer
   sensorValue2.f = readLuminosity('v'); // integer
   sensorValue3.f = readLuminosity('i'); // integer
   break;

case 3:       // Radiator
   sensorValue1.f = readTemperature(0);   // float
   sensorValue2.f = readTemperature(1);   // float
   sensorValue3.f = readHeatflux();        // float
   break;

case 4:       // U−Value
   sensorValue1.f = readTemperature(0);   // float
   sensorValue2.f = readTemperature(1);   // float
   sensorValue3.f = readHeatflux();        // float
   break;

case 5:       // U−Value 1 (indoor)
   sensorValue1.f = readTemperature(0);   // float
   sensorValue2.f = readHeatflux();        // float
   break;

case 6:       // U−Value 2 (outdoor)
   sensorValue1.f = readTemperature(0);   // float
break;

case 7:  
  // G-Value
  sensorValue1.f = readHeatflux();  
  // float
  sensorValue2.f = readPyranometer();  
  // float
  sensorValue3.f = readTemperature(0);  
  // float
  sensorValue4.f = readTemperature(1);  
  // float
  break;

case 8:  
  // Current
  sensorValue1.f = readCurrent();  
  // float
  break;

case 9:  
  // Oil
  sensorValue1.f = readPulse();  
  // integer
  XXX
  break;

  case 10:  
  // Not defined
  case 11:  
  // Not defined
  case 12:  
  // Not defined
  case 13:  
  // Not defined
  case 14:  
  // Not defined
  case 15:  
  // Not defined

  default:
  
  ;

  }

  // Assemble payload[]
  payload[0] = (byte)nodeType;  
  // Insert nodeType into payload[]
  payload[1] = readRssi();  
  // Insert Received Signal Strength Indicator (RSSI) into
  payload[]
if (sensorValue1.f > -10000) // If sensorValueX contains something more meaningful than
  // then insert sensorValueX, which is a float, into payload
  {
    // byte by byte
    payload[2] = sensorValue1.b[0];
    payload[3] = sensorValue1.b[1];
    payload[4] = sensorValue1.b[2];
    payload[5] = sensorValue1.b[3];
  }

if (sensorValue2.f > -10000) // If sensorValueX contains something more meaningful than
  // then insert sensorValueX, which is a float, into payload
  {
    // byte by byte
    payload[6] = sensorValue2.b[0];
    payload[7] = sensorValue2.b[1];
    payload[8] = sensorValue2.b[2];
    payload[9] = sensorValue2.b[3];
  }

if (sensorValue3.f > -10000) // If sensorValueX contains something more meaningful than
  // then insert sensorValueX, which is a float, into payload
  {
    // byte by byte
    payload[10] = sensorValue3.b[0];
    payload[12] = sensorValue3.b[2];
    payload[13] = sensorValue3.b[3];
  }
if (sensorValue4.f > -10000) // If sensorValueX contains something more meaningful than
 // then insert sensorValueX, which is a float, into payload
{
    // insert sensorValueX, which is a float, into payload byte by byte
    payload[14] = sensorValue4.b[0];
    payload[15] = sensorValue4.b[1];
    payload[16] = sensorValue4.b[2];
    payload[17] = sensorValue4.b[3];
}

// Transmit payload[]
//debugSensors(true); // Just for debugging
//debugValues(true); // Just for debugging
//debugPayload(true); // Just for debugging
transmit(); // Transmit payload via Xbee

for (int i = 0; i < measurementInterval; i++) // Wait before next measurement (This is an ugly
    // workaround, because delays over 35000 milliseconds do not work)
{
    delay(1000);
}

sourceCode/Sensor_Kit_Node_v0/Sensor_Kit_Node_v0.ino
A.2 Gateway Python Script

```python
#!/usr/bin/python
# coding=utf-8

"""
Xbee−Internet Gateway for Sensor Kit v0
"""

import sys
sys.path.append('XBee-2.1.0')
from xbee import ZigBee
import serial
import struct
import urllib
import datetime
import time
import os

PORT = '/dev/ttyAMA0'
BAUD_RATE = 9600

def hex(bindata):
    return ''.join('%02x' % ord(byte) for byte in bindata)

ser = serial.Serial(PORT, BAUD_RATE)  # Open serial port

xbee = ZigBee(ser, escaped=True)  # Create Xbee−
```
API object

```python
usb_folder = os.listdir('/media')[0] # Find folder where the USB-memory-stick got mounted
backup_path = '/media/' + usb_folder + '/backup.csv'

while True:
    try:
        response = xbee.wait_read_frame() # Read information from serial / Xbee
        #print response # Just for debugging
        nodeAddress = hex(response['source_addr_long'][4:]) # Access lower part of Xbee address
        payload = hex(response['rf_data']) # Access payload of Xbee-data-packet
        datalength = len(response['rf_data']) # Access length of payload
        if datalength==18: # If payload has the right size proceed with decoding of the payload
           .nodeType = struct.unpack('B',response['rf_data'][0])[0] # First byte is the nodeType
            rssi = struct.unpack('B',response['rf_data'][1])[0] # Second byte is the Received Singal Strength Indicator (rssi)
            sensorValue1 = struct.unpack('f',response['rf_data'][2:6])[0] # First measured sensor value
            sensorValue2 = struct.unpack('f',response['rf_data'][6:10])[0] # Second measured sensor value
```
sensorValue3 = struct.unpack('f', response['rf_data'][10:14])[0] # Third measured sensor value

sensorValue4 = struct.unpack('f', response['rf_data'][14:])[0] # Fourth measured sensor value

timestamp = datetime.datetime.now() # Access time

print timestamp.strftime('%d-%m-%Y %H:%M:%S') # Just for debugging

print 'nodeAddress=', nodeAddress, ' payload=', payload, ' nodeType=', nodeType, ' rssi=', rssi # Just for debugging

print 'sensorValue1=', sensorValue1, ' sensorValue2=', sensorValue2, ' sensorValue3=', sensorValue3, ' sensorValue4=', sensorValue4 # Just for debugging

# Send data to server

query_args = { # Assembling
    'nodeAddress': nodeAddress,
    'nodeType': nodeType,
    'rssi': rssi,
    'sensorValue1': sensorValue1,
    'sensorValue2': sensorValue2,
    'sensorValue3': sensorValue3,
    'sensorValue4': sensorValue4
}

encoded_args = urllib.urlencode(query_args) # Convert parameters in order to suit URL

url = "https://sustain.arch.ethz.ch/asl_scripts/sensorkit/gateway2mysql.php/?" + encoded_args # Assemble URL

try:
    # Try to open
response = urllib.urlopen(url)  # Open URL
except:  # Could not open URL
    print 'Internet connection failed'  # Just for debugging
    pass  # Continue with script
    #print response.geturl()  # Just for debugging
    #print response.read()  # Just for debugging

# Log data into backup file
f = open(backup_path, 'a')  # Open file named backup.csv with the cursor at the end of the file
f.write(timestamp.strftime('%d-%m-%Y %H:%M:%S'))  # Write timestamp into file
f.write('t')  # Write tabs as separator
f.write(str(nodeAddress))  # Write nodeAddress
f.write('t')
f.write(str(nodeType))  # Write nodeType
f.write('t')
f.write(str(rssi))  # Write rssi
f.write("
")
# Write first measured sensor value with 2 decimal precision
f.write("
")
# Write second measured sensor value with 2 decimal precision
f.write("
")
# Write third measured sensor value with 2 decimal precision
f.write("
")
# Write fourth measured sensor value with 2 decimal precision
f.write("
")
# Write new line
f.close()  # Close File
# Just for debugging
def time.sleep(3)
# if payload is not of the expected length, show me what I received
else:
    print 'datalength does not match: ', source.address, ', ', payload # Just for debugging
except KeyboardInterrupt:
    # Exit script with Ctrl-C
    break
except:
exceptions than KeyboardInterrupt,
    print 'Error, something has gone wrong (very last except)'
    "KeyError: 'rf_data'" happens on power up of a node

    pass
    "while True" loop

ser.close()  # Close serial
    connection

sourceCode/gateway_v0.py
A.3 Server PHP Script

```php
<?php
// Beginning of File<br>
// Just for debugging

echo "Beginning of File<br>"

// Debugging
error_reporting(E_ALL);
ini_set("display_errors", 1);
ini_set('auto_detect_line_endings',TRUE);

// Database information
$db_server  = "localhost";
$db_username = "florastrasse";
$db_password = "Therv8JTJtUuEfPZ";
$db_name    = "florastrasse";

// Received data from gateway
$nodeAddress = strtoupper($_GET['nodeAddress']); // Lower half of Xbee-address (e.g. 5061AAE9) (in capital letters: strtoupper())
.nodeType    = $_GET['nodeType']; // Node type (integer)
$sensorValue[0] = $_GET['rssi']; // Received Signal Strength Indicator (integer)
$sensorValue[1] = $_GET['sensorValue1']; // First measured sensor value (float)
$sensorValue[2] = $_GET['sensorValue2']; // Second measured sensor value (float)
```
```php
// Third measured sensor value (float)
$sensorValue[3] = GET['sensorValue3'];

// Fourth measured sensor value (float)
$sensorValue[4] = GET['sensorValue4'];

$ip = SERVER['REMOTE_ADDR']; // Additional information
$time = time(); // Server time

// Write data to file (as backup)
$fh = fopen("backup.csv", "a+"); // Open file with cursor at the end of the file
fwrite($fh, date("d.m.Y H:i:s", $time)); // Write to file, tabs as separator
fwrite($fh, "$ip" . "\t" . "$time" . "\t" . $nodeAddress . "\t" . $nodeType . "\t" . $sensorValue[0] . "\t" . $sensorValue[1] . "\t" . $sensorValue[2] . "\n");
fclose($fh); // Close file
```

fwrite($fh, $sensorValue[3]);
fwrite($fh, "\t");
fwrite($fh, $sensorValue[4]);
fwrite($fh, "\t");
fwrite($fh, $ip);
fwrite($fh, "\n"); // Use new line for every set of data
fclose($fh); // Close file

// Insert data into database
$conn = mysqli_connect($db_server, $db_username, $db_password, $db_name); // Connect to database

if ($conn->connect_error) // Check database connection
{
    trigger_error('Database connection failed: ' . $conn->connect_error, E_USER_ERROR);
}

$sql = "SELECT node_id FROM mf_node WHERE xbee_address='$nodeAddress'"; // Get node_id from xbee_address
if (!$result = $conn->query($sql))
{
    die('There was an error running the query [' . $conn->error . ']');
}
$row = $result->fetch_assoc();
if (isset($row['node_id']))       // Set $nodeID to the value of $row['node_id'] from the database
{
    $nodeID = $row['node_id'];       // In the case of that $row['node_id'] is empty set $nodeID to -1
}
else
{
    $nodeID = -1;
}
//echo "nodeID: "; echo $nodeID; echo "<br><br>";       // Just for debugging

$measurementTypeID[0] = 1;       // measurementTypeID[0] is set to zero because it always belong to the rssi value
$sql = "SELECT * FROM mf_node_type WHERE node_type_id='$nodeType'";       // Get measurement_type_id from the node_type

if (!$result = $conn->query($sql))
{
    die('There was an error running the query [' . $conn->error . ']');
}
while ($row = $result->fetch_assoc())
{
    $measurementTypeID[1] = $row['value1_type_id'];
    $measurementTypeID[2] = $row['value2_type_id'];
$measurementTypeID[3] = $row['value3_type_id'];
$measurementTypeID[4] = $row['value4_type_id'];
}

for ($i=0; $i<5; $i++) // Insert received values into database
{
    if ($measurementTypeID[$i] != 0)
    {
        $sql = "INSERT INTO mf_measurement (node_id, measurement_type_id, measurement_value) VALUES ($nodeID, $measurementTypeID[$i], $sensorValue[$i])";
        //echo $sql; echo "<br>\n"; // Just for debugging
        if (!$conn->query($sql) === false)
        {
            trigger_error('Wrong SQL: ' . $sql . ' Error: ' . $conn->error, E_USER_ERROR);
        }
    }
}

$sql = "INSERT INTO mf_ip_address (node_id, gateway_ip) VALUES ($nodeID, '$ip')"; // Insert IP address into database
if (!$conn->query($sql) === false)
{
    trigger_error('Wrong SQL: ' . $sql . ' Error: ' . $conn->error, E_USER_ERROR);
}
$conn->close();  // Close db connection

echo "End of File";  // Just for debugging

sourceCode/gateway2mysql.php
B Matlab Code

B.1 U-Value Calculations with Average Method

```matlab
% Average Method U-Value
function uvalue = uvalue_avg(heatflux, temp1, temp2)
    for i=1:length(heatflux)
        if isnan(heatflux(i))
            heatflux(i) = 0;
            temp1(i) = 0;
            temp2(i) = 0;
        end
    end
    uvalue = abs(cumsum(heatflux)./cumsum(temp1-temp2));
end
```

sourceCode/uvalue_avg.m

B.2 U-Value Calculations with Dynamic Method

```matlab
function [U, q, S] = dynamicUvalue(Tau1, Te, Ti, q, t)

% Further Parameters
dt = 300;  %diff(t);
N = length(Ti)-1;
M = 400;
p = N-M;  %dt/10 < Tau1 < p*dt/2  (Test: 30 < Tau1 < 10500)
r = 5;
```
% Tau1 = 100; %p*5/3;
 Tau2 = Tau1/r;
 Tau3 = Tau1/(r^2);

 beta (1,:) = exp(-dt./Tau1);
 beta (2,:) = exp(-dt./Tau2);
 beta (3,:) = exp(-dt./Tau3);

% Configuring X-Matrix
 for i=N-M+1:N
    X1(i-(N-M)) = Ti(i)-Te(i);
    X2(i-(N-M)) = (Ti(i)-Ti(i-1))./dt;
    X3(i-(N-M)) = (Te(i)-Te(i-1))./dt;
    X4(i-(N-M)) = 0;
    X5(i-(N-M)) = 0;
    X6(i-(N-M)) = 0;
    X7(i-(N-M)) = 0;
    X8(i-(N-M)) = 0;
    X9(i-(N-M)) = 0;
    for j=(i-p+1):(i)
        X4(i-(N-M)) = X4(i-(N-M)) + (Ti(j)-Ti(j-1))/dt *(1-beta(1)) *beta(1) *(i-j);
        X5(i-(N-M)) = X5(i-(N-M)) + (Te(j)-Te(j-1))/dt *(1-beta(1)) *beta(1) *(i-j);
        X6(i-(N-M)) = X6(i-(N-M)) + (Ti(j)-Ti(j-1))/dt *(1-beta(2)) *beta(2) *(i-j);
        X7(i-(N-M)) = X7(i-(N-M)) + (Te(j)-Te(j-1))/dt *(1-beta(2)) *beta(2) *(i-j);
        X8(i-(N-M)) = X8(i-(N-M)) + (Ti(j)-Ti(j-1))/dt *(1-beta(3)) *beta(3) *(i-j);
        X9(i-(N-M)) = X9(i-(N-M)) + (Te(j)-Te(j-1))/dt *(1-beta(3)) *beta(3) *(i-j);
    end
B.2 U-Value Calculations with Dynamic Method

end

X = [X1', X2', X3', X4', X5', X6', X7', %, X8', X9'];

% Compute Estimations
Z_star = inv((X')*X)*(X')*(q(N-M+1:N));
q_star = X*Z_star;

% Compute Square Difference
S_square = sumsqr(q(N-M+1:N)-q_star);
S = S_square;
U = Z_star(1);
q = q_star;

end

sourceCode/uvalue_dyn.m
C Electronic Schematics

C.1 G-Value Schematics

Figure 30: G-Value Schematics
Figure 31: Sensor Node Schematics
D Second Case Study

The entities to measure, were distributed in three rooms; an enclosed swimming pool, an utility room and a boiler room. The gateway was placed in the utility room, with the USB-GSM-modem on the outside window ledge. To bridge the Xbee-Network from the utility room to the swimming pool and to the boiler room Hop-nodes were placed in between. The measurements took place from the 8th July 2014 to the 13th August 2014. Because of SD-card corruption, only four days worth of data could be recovered from this measurement.

U-values were measured in two rooms, on five building elements in total. The rooms were a utility room and a enclosed swimming pool. In both rooms the u-value of a wall and a window to the exterior were measured. In the utility room the u-value from the ceiling to the exterior was measured as well. The supply and return temperatures were measured in the boiling room. The position of the sensors are marked in figure 32.

Figure 32: Floor Plan with Sensor Positions

The building was built in the 1960s in fair-faced concrete, and has undergone a retrofit in 2013. It is in the vicinity of Zurich. The high amount of concrete walls made the use of hop-nodes necessary. With the hop-nodes the Xbee communication worked without any difficulty.

D.1 Comments

Optimal conditions for u-value-measurements are steady temperatures indoor and outdoor, big temperature difference between the inside and the outside and minimal solar radiation. Usually this kind of conditions are best met during winter. During our measurement the outside temperature varied as much as 13K over a day and the temperature differences even changed the sign. The time frame
available to measure the u-value on this particular site may have been amongst the worst in the entire year. As one can see from figure 33, the week of the 15th July 2014 was amongst the warmest in 2014. Because of the GSM-connectivity and SD-card corruption issue, only four days of data were available for analysis.

Figure 33: Outside Temperatures of the Nearest Weather Station
Figure 34: U-Value - Utility Room - Wall

Figure 35: U-Value - Utility Room - Window
Figure 36: U-Value - Utility Room - Ceiling

Figure 37: U-Value - Utility Room - Ceiling with Different Outdoor Temperature
Figure 38: U-Value - Swimming Pool Room - Wall

Figure 39: U-Value - Swimming Pool Room - Window
Figure 40: Supply and Return Temperatures - Boiler Room