UNI FR
UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

HUMAN-IST
HUMAN CENTERED INTERACTION SCIENCE & TECHNOLOGY

Master Thesis

Sensing Human Comfort: An Inclusive Implementation of Indoor Environmental Data Collection

Student: Léonard Stalder*
Supervisor: Hamed S. Alavi†
Professor: Denis Lalanne‡

University of Fribourg
Bd. de Pérolles 90
CH-1700 Fribourg

October 2015

*mailto:leonard.stalder@unifr.ch
†mailto:hamed.alavi@unifr.ch
‡mailto:denis.lalanne@unifr.ch

Acknowledgements

Thanks to :

Prof. Dr. Denis Lalanne for the opportunity to work on this topic and for his help.

Dr. Hamed Alavi for his support, his orientation, his help and the reviewing of this report.

Emmanuel Gendre for his help in electronics.

My close family, my girlfriend and my friends for supporting me through this project and the people who helped me reviewing this paper.

Table of Contents

Table of Contents i

List of Figures iii

List of Tables iv

1 Introduction 1
- 1.1 Background and Motivations . . . 1
- 1.2 Overall Objectives . . . 1
- 1.3 How this Report is Structured . . . 2

2 Literature Review on the Concept of Comfort 3
- 2.1 Understanding Indoor Human Comfort . . . 3
  - 2.1.1 Thermal comfort . . . 3
  - 2.1.2 Visual Comfort . . . 7
  - 2.1.3 Acoustic Comfort . . . 9
  - 2.1.4 Respiratory Comfort . . . 10
- 2.2 Human-Building Interactions . . . 11

3 Literature Review on Sensing Comfort 12

4 An Inclusive Sensor-System: Concept Overview 14
- 4.1 Used Standards and Metrics . . . 14
- 4.2 Parameters to Capture . . . 14
- 4.3 The Concept of Comfort and Event Boxes . . . 15

5 An Inclusive Sensor-System: System Implementation Overview 16
- 5.1 Sensor Network . . . 17
- 5.2 Message Broker . . . 17
- 5.3 Visualization and Storage . . . 18

6 Implementation of the Sensor Network 19
- 6.1 Choice of Solution . . . 19
  - 6.1.1 Commercial Products . . . 19
  - 6.1.2 Solutions to Build . . . 19
  - 6.1.3 Our Choice: Open Hardware, Open Software . . . 20
- 6.2 Sensor Network Overview . . . 21
- 6.3 The Sensor Nodes . . . 21
  - 6.3.1 Hardware Choices . . . 21
  - 6.3.2 Prototyping . . . 23
  - 6.3.3 Code Implementation . . . 23
  - 6.3.4 Printed Circuit Board Fabrication . . . 25
  - 6.3.5 Case Printing . . . 26
- 6.4 The Radio-Frequency Network . . . 26
- 6.5 The Internet Gateway . . . 28

7 Implementation of the Message Broker 29
- 7.1 State of the Art of the Device to Server Protocols . . . 29

7.2 MQTT Protocol Mechanism . . . 30
7.2.1 MQTT Broker Deployment on Server . . . 30

8 Implementation of the Database and Visualisation Tool 32
8.1 Technology Choices . . . 32
8.1.1 Database Choice . . . 32
8.1.2 Client Webpage Choice . . . 32
8.2 Database . . . 33
8.2.1 Database Structure . . . 33
8.2.2 Database Access Object . . . 33
8.2.3 Database Deployment . . . 34
8.3 Client Website . . . 34
8.3.1 Concept Overview . . . 34
8.3.2 Client Website Architecture . . . 36
8.3.3 Client Website Deployment . . . 36

9 Performance Tests 37
9.1 Performance Tests Environment . . . 37
9.2 Performance Measurement Procedures . . . 37
9.3 Performance Results . . . 38
9.3.1 Time from Sensor to Visualization . . . 38
9.3.2 System Interruptions or Crashes . . . 38

10 Applications 39
10.1 Research Tool . . . 39
10.2 Commercial Solution . . . 42

11 Conclusions 43
11.1 Contributions . . . 43
11.2 Future Work . . . 43
11.2.1 Hardware Improvements . . . 43
11.2.2 System Improvements . . . 44
11.2.3 Concept Improvements . . . 44

References 45

Appendices 47
Met value table . . . 47
Clothing insulation table . . . 48
Illuminance levels for different tasks . . . 49
Communication module comparison . . . 50
Wiring illustration of the primary sensor node . . . 51
Node code example . . . 52
Install mosquitto 1.4.2 with websockets on ubuntu . . . 55
Python MQTT to MySQL script . . . 56
Install MySQL . . . 57
JavaScript MQTT client code example . . . 58
Python script stores arduino time-stamp . . . 59

List of Figures

1 The four dimensions of comfort . . . 3
2 Measuring light levels . . . 8
3 The Noise Rate curves . . . 9
4 The Comfort Box on the desks and the Event Box on the walls . . . 15
5 An overview of the real-time web-interface . . . 15
6 system architecture . . . 16
7 The Comfort Boxes on the desks and the Event Boxes on the walls . . . 16
8 Comfort Box sends a radio-message to the central gateway . . . 17
9 Central gateway runs as interface between radio and TCP/IP . . . 17
10 the gateway publish messages to the broker and clients consumes . . . 18
11 the broker forwards messages to the subscribed clients . . . 18
12 sensor network solutions overview . . . 19
13 Sensor network with Internet hub . . . 21
14 sensor networks solutions overview . . . 21
15 Selected sensors . . . 22
16 nRF24L01+ communication chip . . . 22
17 Picture of our first prototype after wiring . . . 23
18 Picture of our first prototype after wiring . . . 23
19 Example of a code execution . . . 24
20 Example of a code execution . . . 25
21 First produced PCB . . . 25
22 Production steps for a Comfort Box . . . 26
23 Different network configurations . . . 26
24 nRF24L01+ packet structure . . . 27
25 Raspberry Pi 2 B . . . 28
26 layers of IoT protocols . . . 29
27 How does mqtt works . . . 30
28 SensorValues table example . . . 33
29 Ways to display the data . . . 34
30 User-interface : display the users . . . 35
31 User-interface : display the rooms . . . 35
32 Sensor Values table example . . . 36
33 Some values of a day . . . 39
34 Some values of a day . . . 40
35 Left : correlation matrix when the user is present, right: correlation matrix when nobody is in the room . . . 41
36 Concept of an improved Comfort Box . . . 42
37 Wiring of the Comfort Box . . . 51

List of Tables

1 Fanger's Thermal Comfort Scale 4
2 Frequency and decibel sensed test values 9
3 Frequency and decibel Compared to NR 9
4 Noise rate applications 10
5 Sensors test error rates 22
6 Number of lost radio-messages 27
7 Average time from node to visualization 38

1 Introduction

1.1 Background and Motivations

Today, in developed countries, people spend more than 90% of their time indoor [1]. Comfortable indoor conditions can considerably improve our health, well-being, and performance. On the other hand, providing indoor comfort entails considerable energy consumption, and indeed buildings take a large share of the world's total energy. In Switzerland, for example, approximately 50% of the primary energy consumption is attributed to buildings [2].

To provide and increase indoor comfort, today's constructions increasingly rely on building automation systems (BAS). These systems use optimized schemes for ventilation, heating, lighting and air-conditioning. The used schemes are grounded in statistics and standard values like the size of the room or its ambient temperature[3]. For example, thermal comfort in buildings is controlled by simple dry bulb temperature settings. In addition, temperature sensors are not always placed at demanded spots around occupants. All these systems regulate the building based on average values that are not user-centred parameters. The emergence of advanced sensing and actuation systems allow, however, buildings to be more efficient and responsive to occupant's needs than ever before.

The opportunities offered by new technologies can modify the way in which we interact with a building. Considering that people spend a significant amount of time inside, how can we improve their life quality and ensure the best indoor user experience while reducing energy consumption, using new interactive technologies?

This project investigates this question from a technological point of view. More precisely, we aim at integrating a sensor network into the built environment of office workplace, which can also compile the collected data and compute the comfort level in the given context. This can be employed to assess a new energy saving strategy to see how it compares to the other energy systems in terms of occupants' comfort.

1.2 Overall Objectives

In order to understand and improve inhabitants' comfort, this project aims at understanding and sensing user's comfort data tracking while imposing the least possible constraints.

The first goal is to understand and sort out the existing studies about Human Indoor Comfort (HIC) and Human-Building Interactions (HBI). Built on top of the first parts results, we designed a sensor-system that collects data in order to measure the parameters that influence thermal, visual, acoustic and respiratory comfort.

In the second development iteration our sensor system includes a system of human-building interaction tracking. We conjecture that actions such as opening a window or manipulating the ventilation can complete our objective measurements towards a more context-aware sensing system.

We believe that the result of this work, coupled with a visual feedback system, can inform novel indoor user experience design as well as future energy saving strategies.

In summary, the three main goals of this work are as follows:

- Understanding current theories about human indoor comfort and human-building interactions with the goal of defining what we need to sense
- Designing a system to measure the parameters needed to compute HIC and HBI (sensor network, data collection, data storage)
- Testing the system in experimental environment and make preliminary analysis on the collect data

To conclude, this project is the first phase of a more user-centered project that will allows users to have personal comfort profiles and optimal comfort conditions at their work-space.

1.3 How this Report is Structured

This report explains the procedure that we followed to achieve our goals. It is divided into two parts: The first part focuses on the different theories about comfort and human-building interactions. It contains a literature review about the comfort models and theories. The second part describes the developed system to track comfort based on the theories of the first part of this work.

2 Literature Review on the Concept of Comfort

Broadly speaking, comfort can be defined as "a sense of physical or psychological ease, often characterized as a lack of hardship " [4]. The formal definition of comfort has been, however, under debate for more than half a century without an official consensus. Questions about comfort have been addressed by physiologists, computer-scientists, biologists, historians, sociologists, geographers, and urban planners; and in general they define it as a situation of absence of discomfort or a perception of wellbeing and aesthetics [5, 6].

The mentioned definition is easy to understand, but it is difficult to quantify and express in physical parameters. In our work we are not interested in answering the philosophical or psychological questions about comfort but more the technological question of how can we collect data about the inhabitant's comfort.

In this section we first give a short overview of existing research in the area of sensing human-comfort. Then we choose indexes, methods and techniques to measure an indoor human comfort and human-building interactions. The second part treats about Human-Building Interactions.

2.1 Understanding Indoor Human Comfort

Being comfortable involves all the dimensions of physical environments, and also depends on the person's psychological and behavioral aspects. Here, our goal is limited to summarizing only the standard methods or techniques that define the criteria for evaluating all parameters and that can have impact on indoor comfort.

In most studies we found four sub-comfort dimensions: Thermal, Visual, Acoustic, and Respiratory. These aspects are generally quite well known and many standards specify minimum and maximum thresholds for the relevant parameters (light, temperature, sound power, etc.).

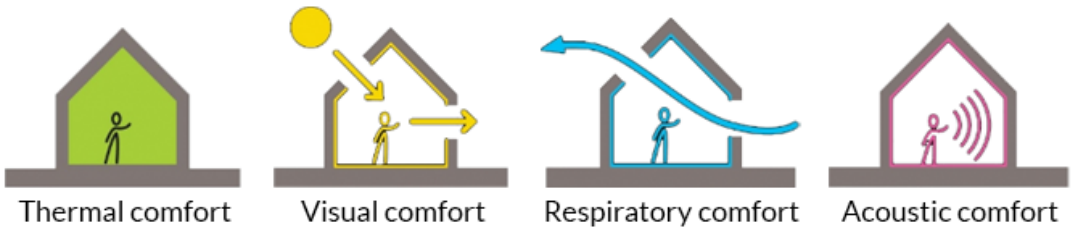

Figure 1: The four dimensions of comfort

2.1.1 Thermal comfort

People have always striven to create environments that are thermally comfortable. This can be traced in the history of architecture and building design. The ISO standards defines thermal comfort as "a condition of mind which expresses satisfaction with the thermal environment" [4]. This definition is mostly agreed upon. However, finding the

physical parameters that compose thermal comfort seems to be a challenge. If we take as an example two individuals: one is doing a physical activity outdoor and the other is sitting behind the computer in a room, the thermal environments are different but both might feel equally comfortable. This pronounces that thermal comfort is a combination of many physical parameters, and far more complicated that just the air temperature.

Researchers have been therefore interested in identifying these parameters that influence thermal comfort. At the beginning of the 20th century new techniques of cooling and heating appeared. The study of thermal comfort, related to temperature, began in England and the United States. One of the first results in this new field of research was a chart defining the comfort zone for most people in the United States proposed by the American Society of Heating and Ventilating Engineers [7]. Later they showed that this chart is applicable worldwide.

In 1932 Vernon and Warner, and in 1936 Bedford, conducted new empirical studies about comfort. In the 60s, Nevis and McNall used participants that rated their thermal sensation in relation to specific thermal situations. Based on these studies, Fanger defined in 1970 a comfort equation describing the comfort based on six parameters: activity, clothing level, air temperature, mean radiant temperature, air velocity and air humidity. This equation was later standardized and has became an universal comfort equation called Predicted Mean Vote (PMV) index. Many other methods for thermal comfort evaluation have been done at the same time, a comprehensive summary of which is provided by Olesen et al. [8].

Predicted Mean Vote (PMV)
The most known and standard indicator for thermal comfort is the Predicted Mean Vote (PMV). PMV (ISO 7730 standard) has become the most used index for measuring thermal comfort. More precisely, PMV is a prediction index that can be used to compute thermal comfort by using a model that can predict the mean value of the votes of a large group of people. It refers to a thermal scale that runs from Cold to Hot. This scale was developed by Fanger and adopted as an ISO standard (ASHRAE standard scale) [9]:

Value	Sensation
-3	Cold
-2	Cool
-1	Slightly cool
0	Neutral
1	Slightly warm
2	Warm
3	Hot

Table 1: Fanger's Thermal Comfort Scale

The recommended PMV value is between -0.5 and +0.5 for an interior space[10].

Parameters of the PMV
The PMV index computes the mean value of ratings of a group of people in a given environment. To compute the PMV we need four environmental variables, the person's activity and the clothing level:

1. Air temperature

2. Radiant temperature: the weighted average of all the temperatures from surfaces surrounding the occupant
3. Air velocity
4. Relative humidity: percentage of water vapour in the air
5. Metabolic rate (met): the energy generated from the human body (see appendix of met value table)
6. Clothing insulation (clo): the amount of thermal insulation the person is wearing

Theses parameters are proved [10] to influence the thermal sensation. It is worth mentioning that, not all the parameters have the same significance. Fanger's equation reveals that the temperature of the surfaces that surround an inhabitant has a big influence on the thermal sensation. The humidity on contrary has a small influence of thermal sensation.

The four first parameters are objective values that can be sensed with help of sensors in a room.

The two last parameters are more subjective values, and are often given as input.

The metabolic rate depends on the amount of muscular activity done by a user. The metabolism is measured in Met (1 Met = 58.15 W / m2 of body surface). Our metabolism is at its lowest while we sleep (0.8 Met) and at its highest during sport (up to 10 Met). A metabolic rate table done by the ASHRAE institution [9] can be found in Appendix A .

The second subjective parameter is the clothing level. Clothing reduces the body's heat loss. To measure clothing's insulation we take the Clo unit. The Clo value 0 corresponds to a naked person and someone with a business suit has a value of 1.0. The Appendix B contains a table of all corresponding values to compute the PMV.

Computing the PMV
If we express PMV as a mathematical expression [9, 10] we obtain the following expression:

$$PMV = [0.303e^{-0.036M} + 0.028]\{(M - W) - 3.96E^{-8} f_{cl}[(t_{cl} + 273)^4 - (t_r + 273)^4] - f_{cl} h_{cl}(t_{cl} - t_a) - 3.05[5.733 - 0.007(M - W) - p_a] - 0.42(M - W - 58.15) - 0.0173M(5.87 - p_a) - 0.0014M(34 - t_a)\}$$

with

$$f_{cl} = \begin{cases} 1.0 + 0.2 l_{cl} \\ 1.05 + 0.1 l_{cl} \end{cases}$$

$$t_{cl} = 35.7 - 0.0275(M - W) - r_{cl}\{(M - W) - 3.05[5.73 - 0.007(M - W) - p_a] - 0.42[(M - W) - 58.15] - 0.0173M(5.87 - p_a) - 0.0014M(34 - t_a)\}$$

$$R_{cl} = 0.155 l_{cl}$$

where e is the Euler's number (2.718), M is the metabolic rate (W/m^2), W is the effective mechanical power (W/m^2), P_a is the water vapor partial pressure, f_{cl} is the clothing surface area factor, t_{cl} is the clothing surface temperature, t_r is the mean radiant temperature, I_{cl} is the clothing insulation, t_a is the air temperature and h_{ac} is the convective heat transfer coefficient.

The water vapor partial pressure, P_a, and the effective mechanical power, W, are calculated by

$P_a = RH * exp[16.6536 - 4030.183/(t_a + 235)$

$W = \eta M$

where RH is measured in % and η is the effective utilization coefficient of the mechanical work. The clothing surface area factor f_{cl}, is determined as follows :

$h_c = 12.1(V)^{1/2}$

The clothing surface temperature, t_{cl}, is calculated by

$t_r = 35.7 - 0.028(m - W) - l_{cl} <= 0.78m$

The PMV is calculated in an iterative process. The PMV index establishes a quantitative expression about the rate of body heat storage and the thermal sensation. This value represents the sensation of the majority of the population in the same environment.

2.1.2 Visual Comfort

Visual comfort is defined as a feeling of well-being in relation to the light (artificial or natural source) while performing a specific task [11]. A good visual comfort ensures that people have enough light for their activities, without exposing the eyes to a higher light level than to which it can adopt.

The perception of light is one of the most important human senses. With our visual perception we can easily learn the space around us. The eye, acting as an interface with the environment is sensitive not only to the characteristics of light, but also to its variation and distribution[12].

Visual comfort is an objective notion that we can measure with quantifiable parameters but can also be influenced by subjective values such as age. Since 1900, different researchers focused their study on optimizing the best environment to have an optimal visual comfort[13]. They found out that visual comfort can depend on:

1. Physical parameters: illuminance which identifies luminance.
2. Environmental parameters: indoor or outdoor (people normally need more light outdoor)
3. Task: reading, office work, goods handling, etc.
4. Physiological parameters: age, sight, etc.
5. Sociological parameters: culture, education

Based on these parameters, researches created new metrics and indexes. Today a lot of metrics, standards and indexes exist to quantify visual comfort. The parameters that are commonly measured are as following:

1. the light level that is the amount of light on a surface (illuminance)
2. the light of day factor that is a ratio between the natural indoor illumination received at a point and the outside illumination
3. the Daylight Autonomy (DA) that represents a percentage of annual daytime hours that a given point in a space is above a specified illumination level
4. the light distribution and uniformity that characterize changes in the level of illumination
5. the color rendering that computes the perception of the colors to our eye
6. the glare
7. the shadows

Luminance, illuminance, glare and contrast are the factors that are the most noticeable by human eye. These are objective measures, but how people experience this light can be rather subjective. These factors are the most representative of visual comfort, and can be measured with the following tools:

1. Illuminance (in lux) can be measured easily with a luxmeter
2. Luminance (in candela per square-meter) need a lot of material to be measured (luminancemeter)
3. Glare (in UGR) can also be measured with a luminancemeter

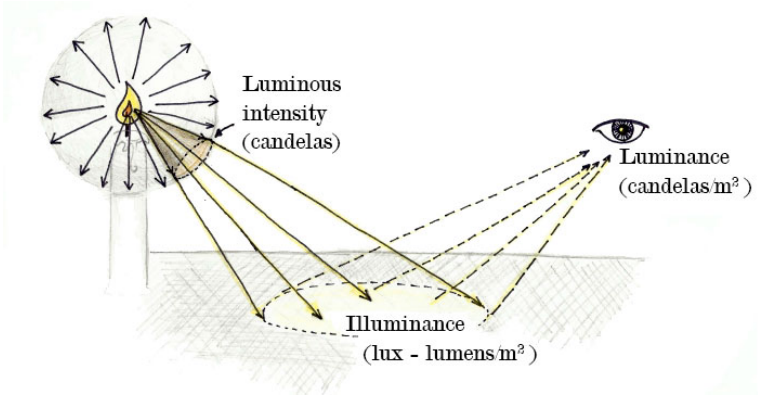

Figure 2: Measuring light levels

Due to the price and the complexity of measuring glare and luminance we decided to focus on illuminance. Illuminance is the amount of light falling on a surface (in our case on the working place) and is measured in lux. This value can indicate if the right level of light is provided for a certain task. A table in Appendix C done by the Illuminating Engineering Society gives us the different illuminance levels for different tasks. For example, if you work behind your computer you need 500 lux to be comfortable or if you do precision assembly you need 1500 lux.

2.1.3 Acoustic Comfort

Appropriate building acoustic is a major contributor to work performance and well-being. People are more productive when they are not distracted by outside noise. To guarantee a good acoustic comfort, the first step is to fix a maximal noise level: a maximal decibel level. But the sensibility of the human ear varies depending also on the frequency. For example 60 dB to 1000 Hz is more disturbing than 60 dB to 250Hz. One well-established way to determine local acoustic comfort is to use the Noise Rating curves. These curves are developed by the International Organization for Standardization [14] to determine the acceptable indoor environment for hearing preservation, speech communication and annoyance.

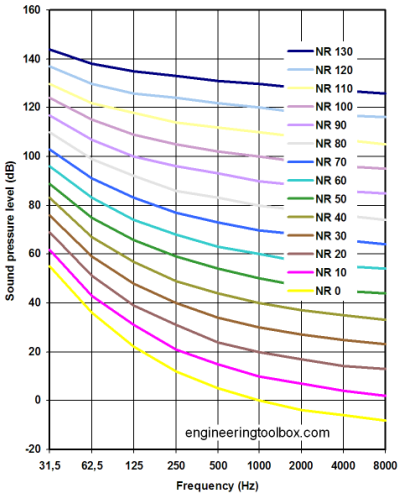

Figure 3: The Noise Rate curves

The curves correspond to different noise rates. To compute the Noise Rate we have to capture the frequency of local noises and the decibel level. The decibel level (dB) and the frequency form together an octave band. Each octave band is compared to the values in the NR curves (see Figure 3). For example the following values can be captured by an ordinary off-the-shelf microphone:

Frequency	63Hz	125Hz	250Hz	500Hz
Decibel level	74.1	75.3	68.9	59.6

Table 2: Frequency and decibel sensed test values

These values can be compared in the NR curves (see Figure 3) and we obtain the correspondences given in Table 3.

Frequency	63Hz	125Hz	250Hz	500Hz
Decibel level	74.1	75.3	68.9	59.6
NR	49	62	61	51

Table 3: Frequency and decibel Compared to NR

Table 4 shows us then in which range we are and if the acoustic environment is comfortable.

Noise rating	Application
NR 25	Concert halls, broadcasting and recording studios, churches
NR 30	Private dwellings, hospitals, theaters, cinemas, conference rooms
NR 35	Libraries, museums, court rooms, schools, flats, hotels, executive offices
NR 40	Halls, corridors, cloakrooms, restaurants, night clubs, offices, shops
NR 45	Department stores, supermarkets, canteens, general offices
NR 50	Typing pools, offices with business machines
NR 60	Light engineering works
NR 70	Foundries, heavy engineering works

Table 4: Noise rate applications

2.1.4 Respiratory Comfort

The CO2 (Carbon Dioxide) is the main indicator for the indoor air quality. It allows for measuring the indoor pollution which can vary with the presence of people[15] in a room.

The human body needs Oxygen and produces Carbon Dioxide. For example, an adult man at rest produces 20 liters per hour of Carbon Dioxide[15]. If the ventilation rate of an occupied room is insufficient, the air is quickly tainted. High level of CO2 in air causes a less active breathing and premature fatigue, and thus it is highly recommended to work in a ventilated room.

Carbon Dioxide concentration is given as a mole fraction. The mole fraction is a way to express the composition of a mixture with a dimensionless quantity. This ratio is indicated in millionths or ppm (parts per million).

Different studies and standards exist about the acceptable ppm values for CO2. The latest and most used standard is the ANSI/ASHRAE standards [10], which summarize the effects of increased CO2 level. With 1500 ppm the air is qualified as stale. The ASHRAE's standards recommend a level below 1000 ppm.

2.2 Human-Building Interactions

By nature, humans are lazy, tend to seek comfort and have their habits. By laziness we do mean the unwillingness to work, but the tendency to minimize efforts and maximize rewards. It's not a weakness but rather a biological story. We can say that a human is programmed to be energy efficient: they minimize efforts and maximize rewards.

In this section we want to build an understanding on how the occupants interact with their built environment to stay comfortable in it. In other words, to understand how, in interaction with the occupants, the buildings can provide maximal comfort while minimizing the environmental impact. The term human-building interactions has been only recently used for a few studies in the domain of building performance and civil engineering referring to the control of occupants on the ambient thermal conditions. We believe that the topic can be at the interest of several disciplines such as human-computer interaction, architecture and building performance with a much wider coverage.

There has been studies on three aspects of human presence in the building: (1) where the occupants are positioned in the building, (2) what they do, and (3) how they move in the building. An overview of these works is provided by Jeff Hsu et al. [16]. Rodden et al. [17] tries to define how interactive technology can be integrated into the built environment.

In our work, as the focus is on collecting comfort data, we only investigate the type of interaction between human and building that can have an impact on the users' comfort; examples are opening the window for fresh air (less CO2) or cooling down the ambient temperature.

3 Literature Review on Sensing Comfort

Today, building management systems are installed to control and monitor the buildings mechanical and electrical equipment such as ventilation, lighting, power systems, fire systems, and security systems. Commercial building management systems run on defined hours, maximum occupancy rates, and standardized comfort profiles[18]. However, what such systems have provided is far from optimal in terms of the comfort satisfaction for the occupants. Window blinds never work pertinent to the occupants needs, automatic ventilation do not take into account subjective parameters, and automatic light conditions always need human interventions.

To improve these systems a lot of studies tried to measure user's comfort and to define the factors that influence human indoor comfort. The recent studies in this field have in common the need of environmental data in order to compute comfort. These are variables like temperature, humidity, air velocity or activity level.

Wai et al. [19] build a sensor network for measurement of human thermal comfort feelings based on a well-recognized human thermal comfort index : the Predicted Mean Vote. They sensed the temperature of the surfaces, the air temperature, the air speed, the humidity and the metabolic rate. They performed an laboratory experiment to evaluate the feasibility of their system. A study was then carried out to evaluate the economic impact of the measurement system. They found out that their measurement system could create better thermal comfort environments with energy saving of 5.8%. This study focuses on thermal comfort.

Jazizadeh et al. [20] build a sensor network that computes occupants comfort profiles by using fuzzy rule-based descriptive and predictive model. Their solution focuses on occupant's comfort perception. The system collects subjective data from the user and objective data with a sensor network. An algorithm computes a user's profile based on subjective and objective data. In a second part, an actuation system regulates only the temperature of a room. The interesting thing in this study is that they use real human factors for the computation of the PMV (metabolic rate, activity or clothing level). Indeed their framework allows the user to give feed-backs in relation to subjective values. Here also the focus is on thermal comfort.

Qingyuan Zhu et al. [21] presents a new thermal comfort measurement system based on the PMV. The goal of the system is to display the PMV to the user on a web-page. They compare their values to the PMV values obtained from the ISO 7730 standard to show the correctness of their system. Subjective values like metabolic rate, activity or clothing level are constant and not real user feed-backs. Their system includes an air quality module: it's a great advantage of the two other studies because here they care about thermal comfort but also respiratory comfort.

Kumar et al. [22] build a sensor to compute the comfort of a room based on the PMV and air quality. They sense parameters such as: temperature, relative humidity, CO and CO2. The results of their study show that we can use the PMV as thermal comfort index in a smart sensor network. Here it's more a tool to validate the PMV index than a system to capture building data. In comparison to the two first studies, here we take care not only of thermal comfort but also of acoustic comfort.

Becerik-Gerber et al. [23] proposes an intermediary communication platform, which enables occupants to communicate their preferences to the building management system. The objective is to facilitate the communication between humans and buildings toward adaptive end user comfort management. They care about thermal comfort and respiratory comfort.

Wienold et al. [24] developed a model that predict glare at a workplace. The goal is to use CCD camera and detect with the help of models when glare appears at a working place. This study is very interesting but they need big infrastructures to capture data. They do not care about other comforts than visual comfort.

De Carli et al. [25] proposes an article about the influence of light in productivity. They analyze all different articles done in this research field. The interesting thing in this study is a table of "optimal light conditions". Here again they focus on visual comfort and not on global comfort.

All these works used the PMV for quantifying thermal comfort, without questioning if the sensors are close enough to the user and if there are other types of comfort that we can sense and compute. Some studies focused on visual comfort, other on thermal comfort or respiratory comfort. No study tried to compute a global comfort index. In addition, the user is generally completely over-looked.

In our research, the goal is first to take the inhabitant as an active part of the building management system and not as just a passive user. Second to capture thermal comfort but also new dimensions of the physical environment that surround a user. In addition, we want to know if the user is also visually, acoustically and respiratory comfortable.

4 An Inclusive Sensor-System: Concept Overview

As we have shown in the previous section, human indoor comfort is an important concern today, and a lot of recent researches strive to address it. So far, there is no commercial or academic measurement system to provide information about subjective human indoor comfort. In addition, most studies focused only on one type of comfort (thermal, acoustic, visual, or respiratory). In this chapter we present a concept of a new measurement system that will collect data related to thermal, acoustic, visual and respiratory comfort. The system will also include an "user dimension", i.e. comfort at the location of user (this will be described in due course in this Chapter). The product is two types of sensor combos, named "Comfort Box" and "Event Box" to measure comfort and human-building interactions events respectively.

4.1 Used Standards and Metrics

Our sensing system is based on the different indexes and standards described in Section 1. We use:

1. the Predicted Mean Vote (PMV) to compute the thermal comfort
2. the level of illumination to define the visual comfort
3. the Noise Rate scale to define the acoustic comfort
4. the CO2 level to define the respiratory comfort

4.2 Parameters to Capture

We use open source environmental sensors to capture relevant data. A sensor is a device that measures a physical quantity and converts it into a digital signal. For example, a gas sensor detects particular molecules and creates an electrical signal whose magnitude is proportional to the concentration of the gas.

A Comfort Box captures the following values:

1. Sound level
2. Presence of the user
3. Illuminance
4. Wind speed
5. CO2 level
6. Humidity and temperature

And, an Event Box captures the following values:

1. Surfaces temperature
2. Windows/doors status

4.3 The Concept of Comfort and Event Boxes

In a workplace, each desk is equipped with a Comfort Box and each surface with an Event Box. The Comfort Box records the values close to the user and the Event Box catches the status of the doors and windows and the temperature of the surfaces. Figure 4 shows an example of scenario how we dispatch the different sensors.

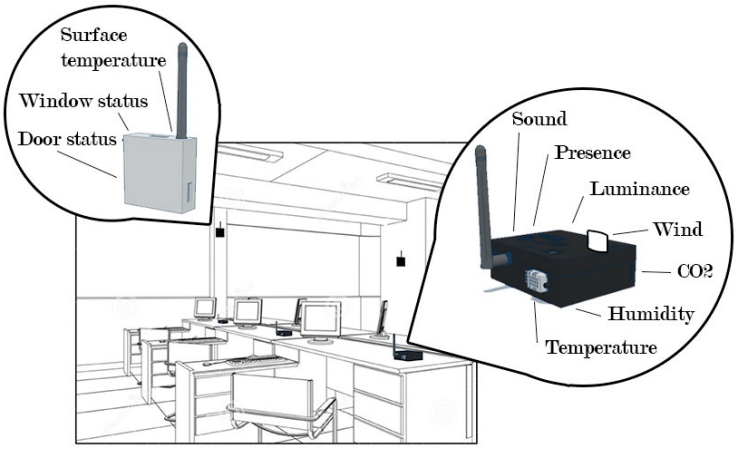

Figure 4: The Comfort Box on the desks and the Event Box on the walls

Both the Comfort Box and the Event Box have a stand-alone micro-controller and a communication chip to send data to a central server for the storage and the display of data.

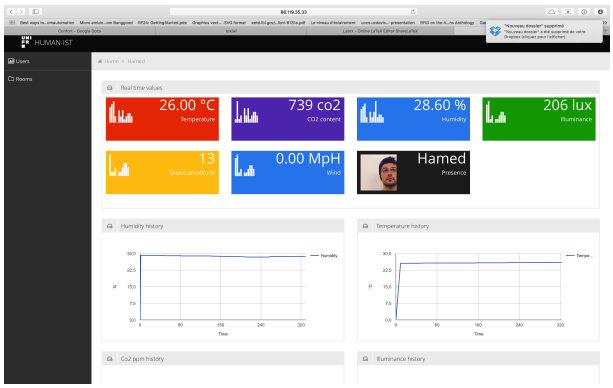

Figure 5: An overview of the real-time web-interface

In addition, to be able to view the collected data, we developed a web-based visualization tool that displays real-time values (Figure 5). This tool can display not only the current objective comfort values but also the user that is at the desk where the Comfort Box is located. The user's identification is done with an RFID tag assigned to the user.

5 An Inclusive Sensor-System: System Implementation Overview

Figure 6 illustrates the overall architecture of the system. In the first part we will explain how the sensors devices work (Comfort and Event Boxes) and how they communicate to a hub that sends the data to the Internet. In the second part, we will explain what is a broker and how we route the messages. In the last part, we will explain how we subscribe to the broker and how we store and visualize the data.

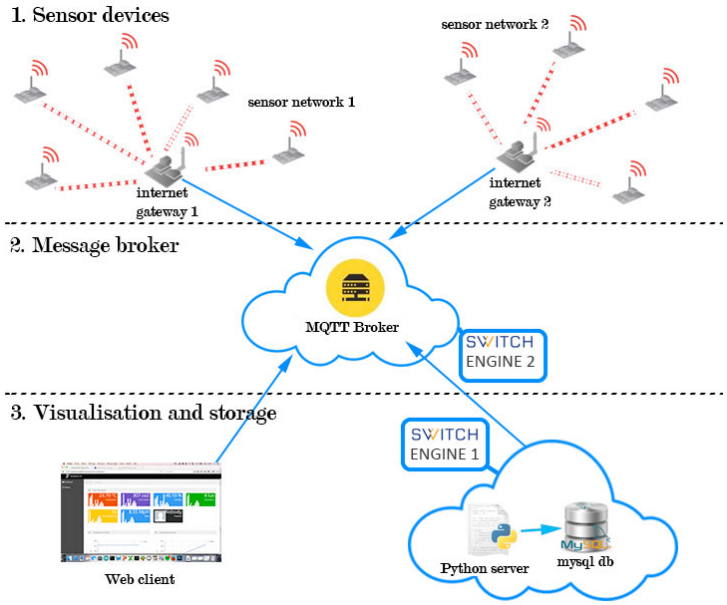

Figure 6: system architecture

In Figure 7 we can see the same process as in Figure 6 but in a linear way, starting with the sensor nodes that capture data and send them to the Internet Gateway through radio-messages. Then the Internet Gateway transmits the message over TCP/IP to a message broker. This broker is responsible to push the data to a visualization tool and a database. In the following sections we describe each of these steps in more details.

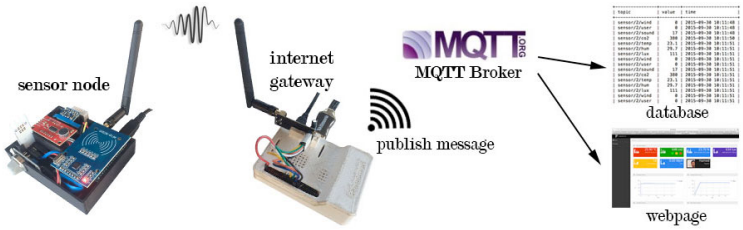

Figure 7: The Comfort Boxes on the desks and the Event Boxes on the walls

5.1 Sensor Network

The first part of our system consists of collecting data through sensor nodes. The Comfort and Event Boxes are based on an Arduino architecture [26]. And the sensors are Arduino-compatible. Once values are sensed by a sensor node, a message is sent, through radio-frequencies (2.400 - 2.4835GHz) to a central gateway. Figure 8 shows a sensor node with different sensors and a radio antenna which is used to send data to the central gateway.

Figure 8: Comfort Box sends a radio-message to the central gateway

The central gateway consists of a Raspberry Pi[27] (connected to Internet) that listens for radio-messages. Once a message is received, the central gateway publishes it to a message server (called a broker) using the MQTT protocol.

MQTT is a lightweight publisher/subscriber protocol based on top of TCP/IP. It is used for connections with remote locations where a "small code footprint" is needed. This protocol is explained in Section 6.4.

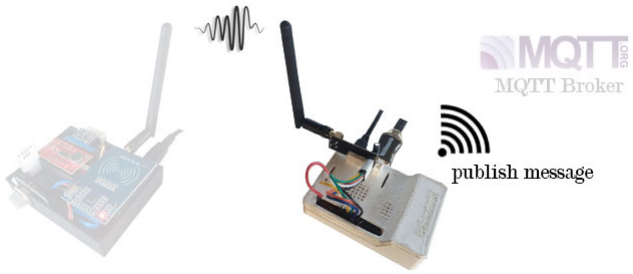

Figure 9: Central gateway runs as interface between radio and TCP/IP

5.2 Message Broker

The message broker is the heart of the publisher/subscriber protocol. The broker is responsible for receiving all the messages from the publishers, and then dispatching them to the clients that have subscribed for that type of message.

In our case the central gateways publish messages to the broker. The broker receives the messages (for example the temperature of sensor #2). Once filtered, the broker sends the message to the visualization system and to the database that have subscribed for all types of environmental data including the temperature of sensor #2.

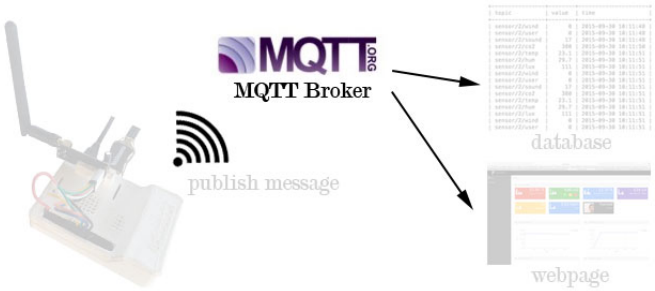

Figure 10: the gateway publish messages to the broker and clients consumes

The MQTT broker is hosted on a virtual machine[1]. We use the mosquitto[28] MQTT open source broker to route the messages.

5.3 Visualization and Storage

The third part of the system consists of storing and visualizing the sensed data. As we use the MQTT protocol the database and the webpage are subscribers to different topics (e.g. sensor/2/hum). The main advantage of this architecture is that the visualization is

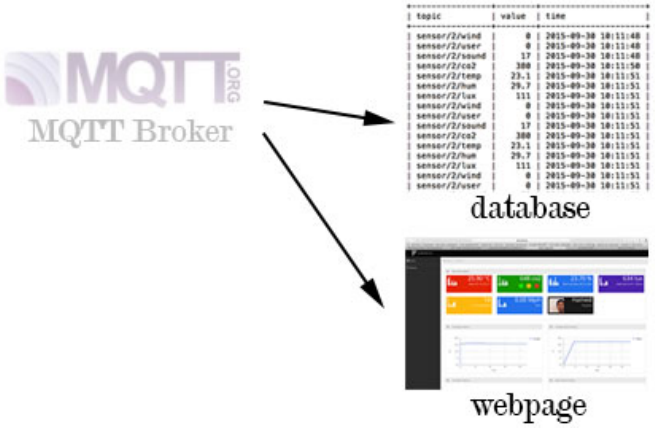

Figure 11: the broker forwards messages to the subscribed clients

done in real-time, and in parallel. We do not need to store the data to read after from the website, the two operations of storing and visualizing the data are done at the same time.

[1]SWITCH virtual machine hosted in Zurich

6 Implementation of the Sensor Network

In this section we focus on the first part of our system, namely the sensor network. We will explain the hardware choices, the architecture and the communication protocols.

6.1 Choice of Solution

Figure 12 demonstrates an overview of the possibilities to acquire a comfort sensor network.

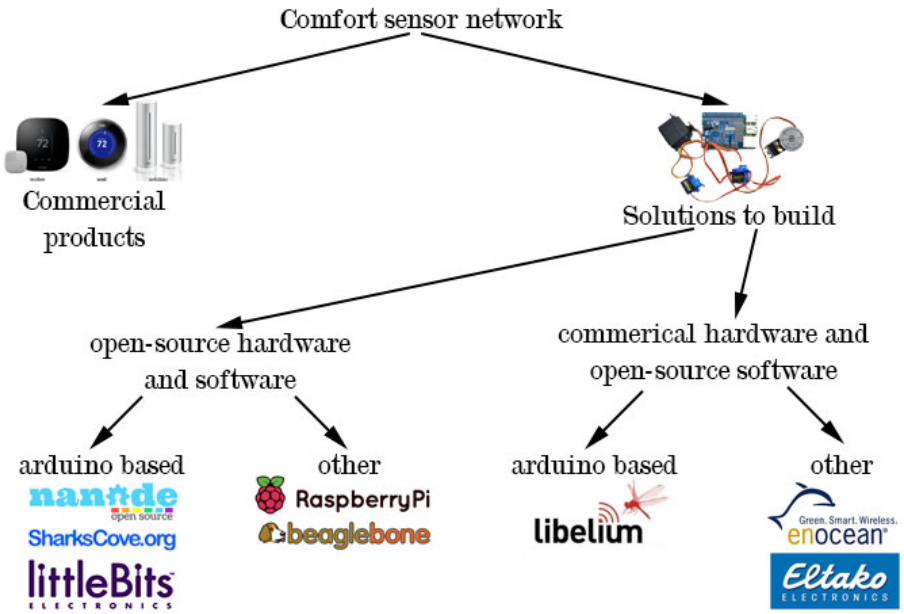

Figure 12: sensor network solutions overview

6.1.1 Commercial Products

First we researched about commercial ready-to-use solutions. Different companies offer devices that capture comfort data.

Pros: The advantage of such solutions is the facility of use. Often there is no need for further development or low-level modification.

Cons: These solutions are typically not customizable in terms of types of integrated sensors. Moreover, the collected data is stored in private commercial clouds and these solutions are quite expensive (e.g. a netatmo sensor box cost 169$).

These solutions were not adapted to what we needed.

6.1.2 Solutions to Build

In recent years, a new kind of micro-controllers was built for causal users, electronics enthusiasts and hobbyists. It became a popular tool for electronics enthusiasts (before Arduino) because it was easy to use and to program [29]. These boards support a lot of

input sensors (motors, led's,...) and open an infinite number of possibilities to develop new applications. Today, the most known implementation of easy-to-program boards are produced by Arduino[26]. Arduino is an open and versatile platform used for the development of electronic prototypes. The goal is to build quickly prototypes. A lot of military, scientific or business products exists because of Arduino. Even many commercial solutions appeared based on Arduino. In this section we will discuss the most known of them and try to compare them.

6.1.2.1 Open-Source Hardware and Software

Complete open source solutions like BeagleBone, Nanode, Arduino or Raspberry Pi offer powerful development platform and are quite less expensive than commercial solutions. A large community of developers exists around them which specially facilitates the debugging.

6.1.2.2 Commercial Hardware and Open-Source Software

Fully or partly commercial solutions offer great opportunities of development too. A lot of sensors-kit are pre-developed and the hardware is Arduino compatible. The negative point is that we have to buy their products, to be fully compatible, and for some solutions to be able to use the provide software.

6.1.3 Our Choice: Open Hardware, Open Software

Arduino-based platforms are powerful to develop complex and flexible projects. The community is big and a lot of research projects use this platform. Our choice is the Arduino compatible world and to take only open-source products.

6.2 Sensor Network Overview

The sensor network is composed of three parts. The first is the sensor nodes that contain the sensors and the micro-controller. The second part is the network that connects the sensor nodes to the central gateway. And the last is an Internet gateway that collects the data and sends it to Internet. In the next three sections we describe each of these parts in details.

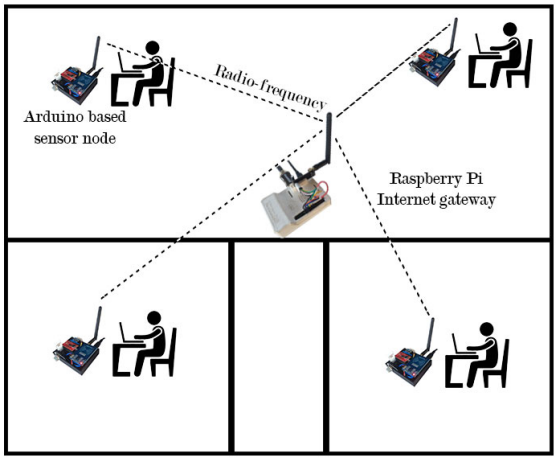

Figure 13: Sensor network with Internet hub

6.3 The Sensor Nodes

6.3.1 Hardware Choices

The main decision was to choose a development platform. After selecting the Arduino platform, choosing hardware components to prototype and build our first node was not too difficult. A lot of sensors, displays, modules and communication chips are compatible with the Arduino platform.

We used the Arduino Micro that offers enough memory, pins and power for 10 sensors. The Arduino Micro is based on the ATmega32U4 processor. It has 20 digital input/output pins, a 16 MHz crystal oscillator and a micro USB connection.

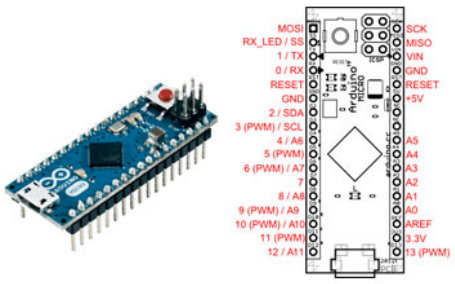

Figure 14: sensor networks solutions overview

Figure 15 shows the sensors we selected (from left to right: MQ135 gas sensor, SparkFun Sound Detector, Ardafruit Tsl007 luminosity sensor, DHT22 humidity/temperature sensor, ModerDevice Wind Sensor, RC555 RFID reader, TMP526 surface temperature sensor, doors status).

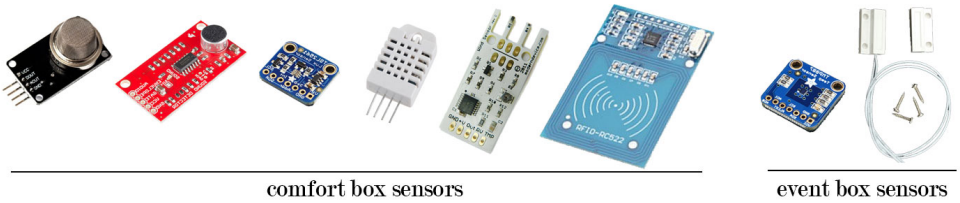

Figure 15: Selected sensors

We tested, calibrated and validated all the sensors that we received against professional devices (e.g. thermometer), which allowed us to choose the most accurate ones. During a test-phase we also measured the error rate for each sensor. Table 5 shows the error rate of our final choices.

Value	Error rate
T	0.49%
Hum	2.72%
Co2	4.66%
Lux	0.24%

Table 5: Sensors test error rates

Then we needed to chose an appropriate communication chip. We searched different low-cost transmission chips and compare them (you can find the comparison in Appendix D). We finally chose the Nordic nRF24L01+ communication chip. It's a highly integrated, ultra low power radio transceiver. This choice was made because it was the chip that is the most energy-saving.

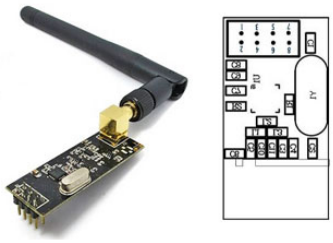

Figure 16: nRF24L01+ communication chip

All these components are Arduino compatible and programmable with the Arduino IDE that makes it easy to write code and upload it to the board.

6.3.2 Prototyping

Once the hardware selected we started to prototype our first sensor node. After wiring all the components together we obtained our two first prototypes. To prototype we required an USB connection to a PC to transfer the source code on the micro-controller. To test and calibrate the sensors we tested different source codes.

Figure 17 shows a prototype of the Comfort Box node and Figure 18 shows a prototype of the Event Box.

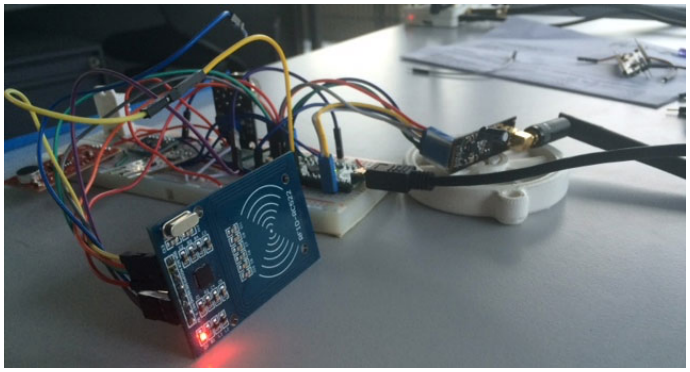

Figure 17: Picture of our first prototype after wiring

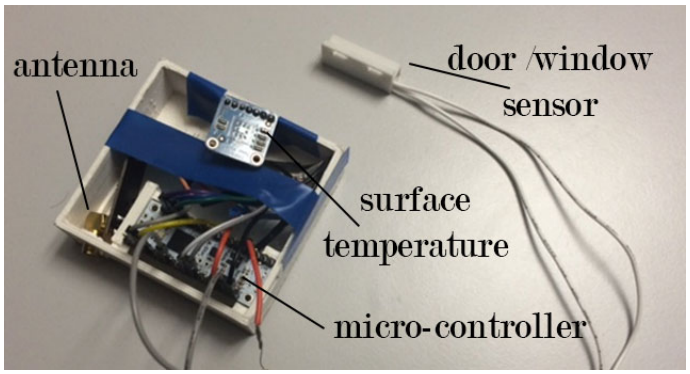

Figure 18: Picture of our first prototype after wiring

We decided during the project to focus on only one box : the Comfort Box. Therefore the Comfort Box will be improved but the Event Box will stay to a prototype status. You can find in Appendix E the wiring schema of the Comfort Box that we produced after the prototyping phase.

6.3.3 Code Implementation

The Arduino IDE is pretty easy to use. The first screen you see is a white window with several different shades. An Arduino project is called "sketch". The only thing that you need to know is that a "sketch" is composed of two main methods : *setup*() and *loop*(). The first one is called when you switch your micro-controller to initialize variables, pin or other stuff. The second method is the part of the code that loops back onto itself. It's the main part of the code. Fig. 19 shows a typical Arduino code execution.

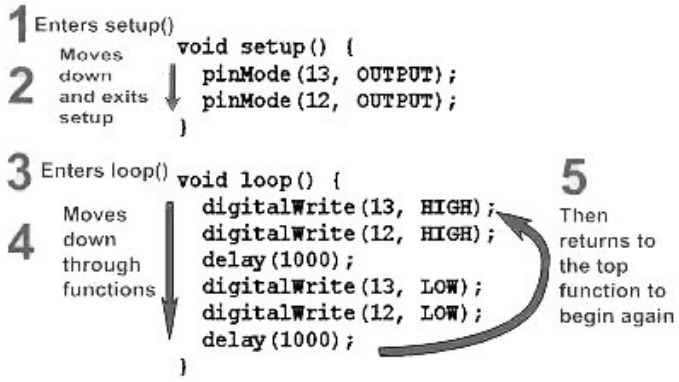

Figure 19: Example of a code execution

To program our nodes we follow the same approach. We have a setup method that initializes all the objects we need (ex: a temperature reader, RFID reader...) and we have a main method called loop. This method is executed each second and takes 400ms to be executed. At each call, the following executions are performed :

1. Read the humidity
2. Read the temperature
3. Read the illumination level (in lux)
4. Read the CO2 level (in PPM)
5. Read the wind speed (in MpH)
6. Read the sound level
7. Check if a user is next to the box
8. Send the data to the internet gateway

At the end of the execution a delay of 1 second is respected before the re-execution of the method.

The second node, composed of a surface temperature sensor and a door/window status sensor performs each second these operations:

1. Read the surface temperature
2. Check if the connected doors/windows are open or closed
3. Send the data to the Internet gateway

6.3.4 Printed Circuit Board Fabrication

After prototyping the boxes we decided to make the node more usable. We mean by usable to put it in a 3D printed plastic case and to avoid the wiring. To avoid the wiring we had to design and produce a Printed Circuit Board. A PCB supports electronic components using conductive tracks laminated onto a non-conductive substrate. Our PCB is double sided (two layers) for much higher density. The goal is to plug the sensors, the microcontroller and the antenna direct on the PCB without wires. We use the Eagle software to design the PCB. Fig. 20 shows the PCB model before the production.

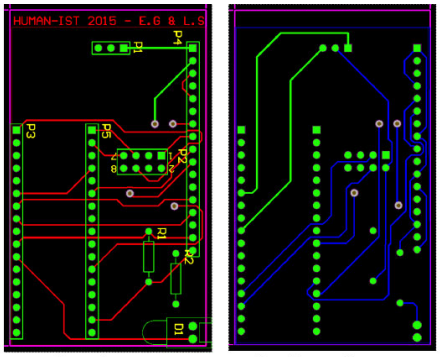

Figure 20: Example of a code execution

After designing the PCB you have to choose a method for creating the PCB. The choice was based on the availability of materials at the University. We use the UV etching method that produces finer and more precise circuits. To apply this method we needed a photosensitive laminated PCB card, an UV insulator and a transparent sheet. A chemistry process with UV and acid attacks the material that is not needed and leaves only the need tracks on the board. A good overview can be found on Wikihow[30]. Fig. 21 shows the

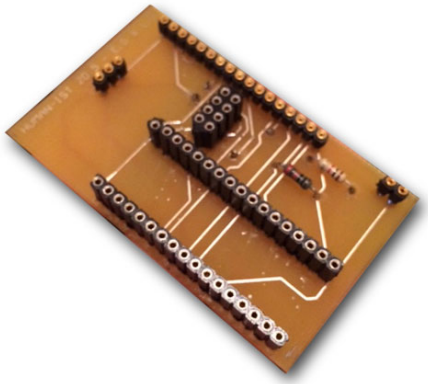

Figure 21: First produced PCB

first PCB we produced. We added headers to just plug the elements on it. Different resistances have been also added.

6.3.5 Case Printing

At this step, we have the sensors, the micro-controller and the PCB. At a software level we can capture data. The next step was to put this all together in a 3D-printed case.

Figure 22: Production steps for a Comfort Box

To design a nice 3D box model we used the online Tinkercad 3D editor[2]. We designed a case for the bottom and for the top. Inside we designed compartments for each sensor and a for the PCB. Once a final model done, we printed it with an Leapfrog Creart HS 3D printer [3]. We encourage a lot of problems and a lot of waste of time to master and calibrate the 3-D printer. We used PLA plastic fiber as plastic component. After the printing phase we assemble the components together and we obtained our first sensor node.

6.4 The Radio-Frequency Network

Once the nodes finished we interested us in transmitting the data to the Internet gateway. Arduino is compatible with a lot of communication chips and protocols. We finally choose the Nordic nRF24L01+ communication chip. It's a highly integrated, ultra low power radio-frequency transceiver. It transmits on the 2.4GHz band and can be easily connected to an Arduino micro-controller.

Concerning the implementation of the communication we used a Arduino library[4] that gives us divers to communicate with the antenna. This library implements an OSI Network Layer using nRF24L01 radio-chips. The network layer listens actively to up to 6 radios at once and is arranged in a star architecture : a node is the base and all other nodes are communicating with this main node. In our case the Internet

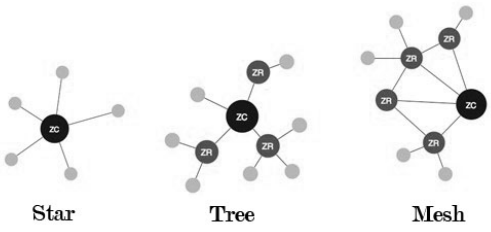

Figure 23: Different network configurations

gateway (Raspberry Pi) is the base node and the Comfort Boxes are the other nodes. To communicate, the chips send messages that are structured in a certain way. The nRF24L01+ has a simple packet structure that is formatted as in Fig. 24.

[2] https://www.tinkercad.com
[3] http://www.lpfrg.com/en/creatr-hs
[4] http://tmrh20.github.io/RF24Network/index.html

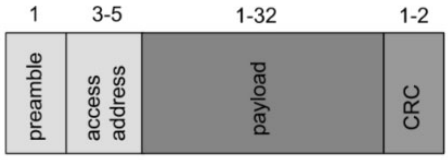

Figure 24: nRF24L01+ packet structure

A message starts with the preamble. This preamble is used by the radio-antenna to identify incoming packets. Second, the node address can be specified as 3 to 5 bytes. Then comes the packet payload of 32 bytes and to finish an optional CRC to check the integrity of the message.

When a messages arrives the radio chip checks the address and the CRC. When it's done the message is stored in a FIFO memory and transmitted to the micro-controller.

The message payload contains 32 bytes of data. These bytes contain the measurements from the sensors and they are structured as follow:

Listing 1: Python example

```
struct payload {
    unsigned co2: 16;
    unsigned user: 16;
    unsigned sound: 16;
    unsigned wind: 16;
    unsigned free: 16;
    unsigned lux: 16;
    unsigned hum: 12;
    unsigned temp: 12;
    unsigned id: 8;
} payload;
```

Each sensor value is represented in 16 bits, the node id takes 8 bits and the temperature and humidity are stored in 12 bits. The payload is encapsulated in the above message structure and sent to the Internet gateway.

To save energy we programmed a "sleep mode" on the communication chips. The radio chip is powered up, then it tries to send the packet. A retries system tries to send the message up to 15 times or until an acknowledgement message is received from the gateway. If no acknowledgement is received after 15 retries the radio is shutting down and the operation will be done at the next loop execution. Tests were performed during 1 hour to find the best radio configuration. The results are the following :

Configuration	packets send	packets received	PDR
1	243	238	0.9795
2	243	212	0.8725
3	243	197	0.8107

Table 6: Number of lost radio-messages

The configuration 1 retries to send the message 15 times and has a data rate of 250kbps. The configuration 2 retries to send the message also 15 times and has a data rate of 1Mbps. The last configuration has a data rate of 2Mbps. If we examine the packet delivery ratio (ratio of the number of delivered data packet to the destination) we clearly see that the configuration 1 is the best.

6.5 The Internet Gateway

To transmit radio-messages to the Internet we needed an Internet Gateway. Remember the architecture of the radio network, it's a star architecture. This means that all the children send their messages to a central node. This central node is the gateway and must be able to catch NRF24 radio messages and send them on the Internet.

To do this we chose to work with a Raspberry Pi. It's a micro-computer of credit card size developed in the UK. The model we chose has a Broadcom BCM2836 CPU and 1 GB of RAM. We chose to work with this computer because it's powerful for its size, its

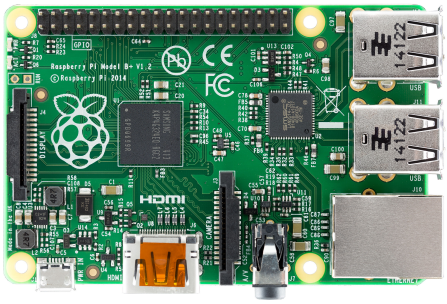

Figure 25: Raspberry Pi 2 B

cheap, can be connected to Internet, and we can wire an NRF24 antenna on it to capture the messages.

We install a Raspbian[5] distribution as operating system. We wired the NRF24 communication chip to the Raspberry Pi SPI pins. Once installed we downloaded the drivers to communicate with the radio antenna and used a C library to receive the radio-messages. A Wi-Fi dongle connected to Internet was installed to connect the Raspberry Pi to the Internet.

The C program runs in background and does the following instructions:

1. Configure the pins and initialize the objects
2. Check if new radio-messages are coming
3. If yes send the message to Internet
4. Send the acknowledgement message to the node
5. Continue at step 2
6. If no continue at step 2

The code of the Raspberry Pi Internet Gateway can be found in Appendix F.

[5]https://www.raspbian.org

7 Implementation of the Message Broker

In this section we will focus on the second part of our system, namely the message broker. We will explain the hardware choices, the architecture and the communication protocols.

7.1 State of the Art of the Device to Server Protocols

The Internet has changed the way people interact and work. The next wave of Internet is the Internet of Things : intelligent and connected devices by connecting physical objects to Internet. To interact with Internet and between them, these devices must work with speeds and scales. To communicate they need low-power and well organised communication protocols. The devices must be able to communicate data between them (D2D). The data must be sent to the server infrastructure (D2S) that has to be shared by the server (S2S) to visualisation tools, analysis programs or databases. In this part we are interested in

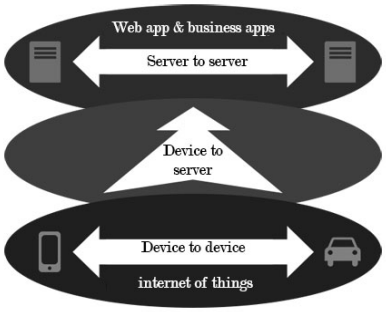

Figure 26: layers of IoT protocols

the middle part of Fig. 26 namely the device to server protocols. We found out the two most widely adopted D2S protocols :

1. MQTT: a publish-subscribe based "light weight" messaging protocol
2. XMPP : a communication protocol for message-oriented middle-ware based on XML

These are the two main used protocols to communicate from a device to a server. We found near to 10 implementations of each protocol. It's difficult to find a good implementation of a message protocol because on the top they do all the same thing. We focused on "real-time" values and a lightweight implementation.

The MQTT protocol is very lightweight. XMPP is a more complex protocol and has much overheads in handling messages. XMPP is based on XML and you have to parse it also. The advantage is that XMPP has a message structure and MQTT not. You have to supply the message structure by using MQTT. In addition, MQTT is a publish/subscribe protocol. XMPP is an instant messaging protocol. This means that you need an extension if you want to support publish subscribe.

To summarize, MQTT has less overhead, is more lightweight, supports pub/sub and no parser is needed. We chose to work with the MQTT protocol and to take his most famous implementation : the mosquitto[6] open source for the broker and the Paho[7]

[6]http://mosquitto.org
[7]http://www.eclipse.org/paho

open-source library for the client.

7.2 MQTT Protocol Mechanism

MQTT was developed by IBM and Eurotech in 1999 to monitor oil pipelines in the desert. The protocol had to be light-weight, bandwidth-efficient and to consume as less energy as possible to send data to a satellite.

The MQTT protocol is based on a publish and subscribe architecture. To compare, HTTP is based on a request/response architecture. The difference between them is that MQTT is event-driven and allows to push messages to clients. All the messages pass through a broker that acts as a router and dispatches all messages between the senders and the receivers. Each client that wants to receive messages related to a certain topic can subscribes to this topic and will receive all messages that are related to the latter. Different clients can be subscribers of the same topic and they can also publish messages. On Fig. 27 you can see how the Internet Gateway publishes a temperature value from a sensor to the broker and how clients subscribe to this topic. Each MQTT client has an

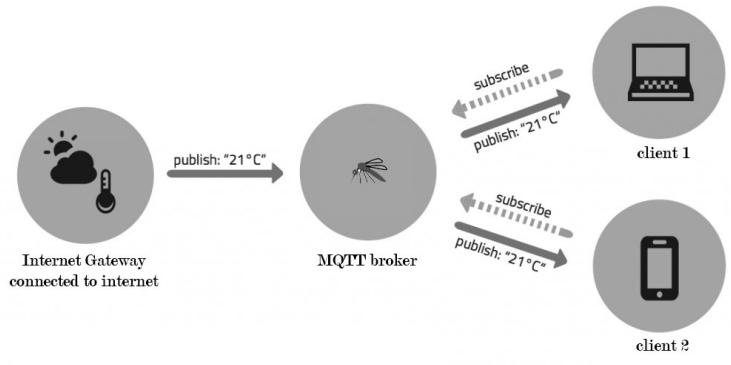

Figure 27: How does mqtt works

open TCP connection to the message broker. If the connection is interrupted, the broker buffers all messages and sends them to the client when the connection is reestablished.

As mentioned MQTT distributes messages with the help of topics. A topic is a string containing different levels. For example the string "room/1/sensor/2/humidity" concerns the room 1 with with the sensor 2 and we want the humidity. The client can choose to subscribe to the exact topic or to a more general topic. For example with the topic "room/+/sensor/2/humidity" the client subscribes to all rooms. We use the + sign to replace a level. If we want to replace more than one level we have to use the sign #. For example "room/#" subscribes to all messages that begin with room.

7.2.1 MQTT Broker Deployment on Server

After choosing our "device to server" protocol we installed a broker on a SWITCHengines[8] virtual machine. SWITCHengines provides compute and storage services to Swiss academics for free. We setup an Ubuntu machine with the following specifications: 2 GB of RAM, 20 GB of hard-drive and 86.119.35.147 as public IP.

[8]https://engines.switch.ch

We installed the mosquitto MQTT broker on it and opened the ports 1883 for the incoming MQTT messages, 9001 for the out-going messages through web-sockets. A full tutorial is available in Appendix G.

On the Raspberry Pi Gateway we installed also the mosquitto MQTT C library. And with the following command we send a message to the broker:

Listing 2: Publish MQTT message instruction

```
mosquitto_pub -h 86.119.35.147 -p 1883 -t sensor/2/hum -m "34.5\%";
```

The MQTT server is ready and at each received message he routes it to the clients that have subscribed to the topic. To subscribe to a topic in JavaScript the few following code lines are enough:

Listing 3: Subscription to an MQTT topic in JavaScript

```
// Create a client instance
client = new Paho.MQTT.Client("86.119.35.147", 9001, "0");
// set callback handlers
client.onMessageArrived = onMessageArrived;
// connect the client
client.connect({onSuccess:onConnect});
// called when the client connects
function onConnect() {
  client.subscribe("/sensor/2/hum");
}
// called when a message arrives
function onMessageArrived(message) {
  console.log("onMessageArrived:"+message.payloadString);
}
```

We had problems by enabling the websockets.

8 Implementation of the Database and Visualisation Tool

In this section we will explain the last part of our system : the visualisation and the database. First we will explain our technology choices and then how it works.

8.1 Technology Choices

8.1.1 Database Choice

The goal was to store the data and to visualize live data at the same time. The MQTT allows us to do both at the same time and we do not waste time in storing the data and request them from the visualization tool. One client will connect to the broker to visualize the data and the other client will connect to the broker to store the data.

First we interested us in choosing a database. There are two types of databases : relational and non-relational. Due to the non-complexity of our data this was not relevant. We decided to store each message in our database on a single row at each time. For this project we have a lot of data that arrives each second. If you take a sensor node it sends 7 values to the broker each second. We produced 3 sensors nodes so we had 21 rows to store each second. For our database prototype we found the following possibilities :

1. MySQL
2. PostgreSQL
3. MangoDB

MySQL is the most popular web-database. There is a lot of documentation, it's a relational database, it's performant and it's secure.

PostgreSQL is for more large-scale project and not really for protypes. The configuration is quite complex and a daemon is needed to access to the database.

MongoDB stores data in JSON-like documents that can vary in structure. MongoDB allows to create records without first defining the structure, such as the fields or the types of their values.

We chose MySQL because it's quick and quite easy to install for a prototype. A lot of scripts exists to interact with a MySQL database. MangoDB is interesting but for prototyping and to save time it's too complex to install.

8.1.2 Client Webpage Choice

We wanted to display the captured data in real time on a web interface which can be consulted by the user. The goal was to show which user is at which place and what are the values around him. To do this we chose to work with Javascript, the PAHO-MQTT client library to connect to the broker and Twitter Bootstrap for the responsive interface.

8.2 Database

8.2.1 Database Structure

MySQL works with tables. We designed a simple structure to store the messages from the broker. Each sensor value has a topic that represents the theoretical path. To each topic is attached a value that represents the sensed value. Each value is timestamped when it's sensed and we store it in a third column called time. Fig. 28 shows you an example of the created table with a few rows.

topic	value	time
sensor/2/wind	0	2015-09-30 10:11:48
sensor/2/user	0	2015-09-30 10:11:48
sensor/2/sound	17	2015-09-30 10:11:48
sensor/2/co2	380	2015-09-30 10:11:50
sensor/2/temp	23.1	2015-09-30 10:11:51
sensor/2/hum	29.7	2015-09-30 10:11:51
sensor/2/lux	111	2015-09-30 10:11:51
sensor/2/wind	0	2015-09-30 10:11:51
sensor/2/user	0	2015-09-30 10:11:51
sensor/2/sound	17	2015-09-30 10:11:51
sensor/2/co2	380	2015-09-30 10:11:51
sensor/2/temp	23.1	2015-09-30 10:11:51
sensor/2/hum	29.7	2015-09-30 10:11:51

Figure 28: SensorValues table example

We decided to store the whole path in a column because MySQL allows to do string queries which allows more flexibility because we do not need to manage other tables.

8.2.2 Database Access Object

To catch the messages from the broker and to store them in the database, we wrote a script that runs on the virtual machine. The script uses the PAHO MQTT client library [9]. This library allows us to connect and subscribe to an MQTT broker. The script we wrote acts as a MQTT client and is written in python. The Python script executes the following instructions :

1. Connect to the MQTT broker
2. Create database connection
3. Subscribe to all the topics
4. Start listening for messages
5. If a message arrives store it in the database
6. Continue at step 4

In Appendix H you can find the script that is running in background to catch messages. This latter is running on a virtual machine.

[9] http://www.eclipse.org/paho/

8.2.3 Database Deployment

Same as the broker we use a new SWITCH virtual machine. We setup an UBUNTU machine with the following specifications: 2GB of RAM, 40 GB of hard-drive and 86.119.35.33 as public IP.

On this server we first install MySQL. You can find a tutorial in Appendix I to install correctly MySQL. Once installed we created a new database with a blank table that contains the columns described in the subsection "Database Access Object". After the configuration we launched MySQL in background.

The next step is to upload our Python script explained in the previous subsection. We run the following command to allow the script to run in background.

Listing 4: Run python script in bakcground

```
1 nohup python python_database.py &;
```

MySQL and the DAO script are now running on the virtual machine. When messages arrive to the MQTT broker they are redirected to this machine and processed by the running script that stores the values in the database.

8.3 Client Website

The main goal of this project was to collect data and not to visualize them. We have treated this subject anyway to have a minimal graphical interface. In this section we will explain how we implement the user-interface. We developed a web-based interface that has the goal to render a live visualization of the environmental conditions that surround a user.

8.3.1 Concept Overview

Representing comfort data to the user is really tricky. What is the best way to represent the data and how present it? After discussing with different people we decided to adopt the following approach: display comfort data can be done by two different ways. The first way is to select the user and have access to the data that surround him. The second way is to select a room and have access to all Comfort Boxes present in the room and select one of them.

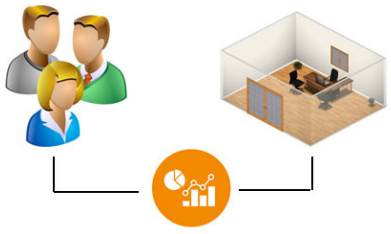

Figure 29: Ways to display the data

To do this we designed an user-interface on which we can select the users and visualize the rooms by clicking on the "room mode". On Fig. 30 you have an example of the first

entry point of our data. All users are displayed and the users that are close to a Comfort Box are mentioned as online.

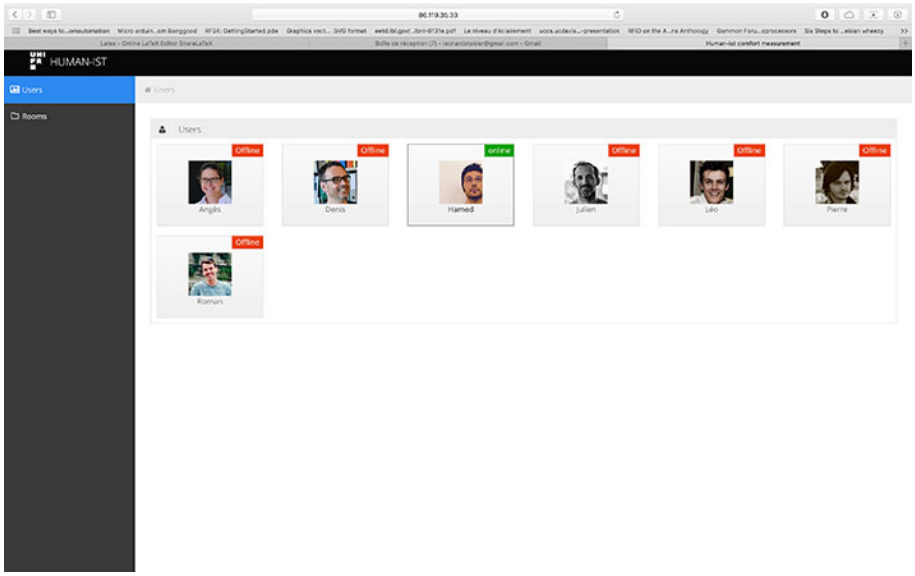

Figure 30: User-interface : display the users

On Fig. 31 you have an example of the second entry point of our data. We see each room of our building and the sensor-boxes that are inside. When the box is connected a green box appears. If a user is close to the box, his picture is displayed.

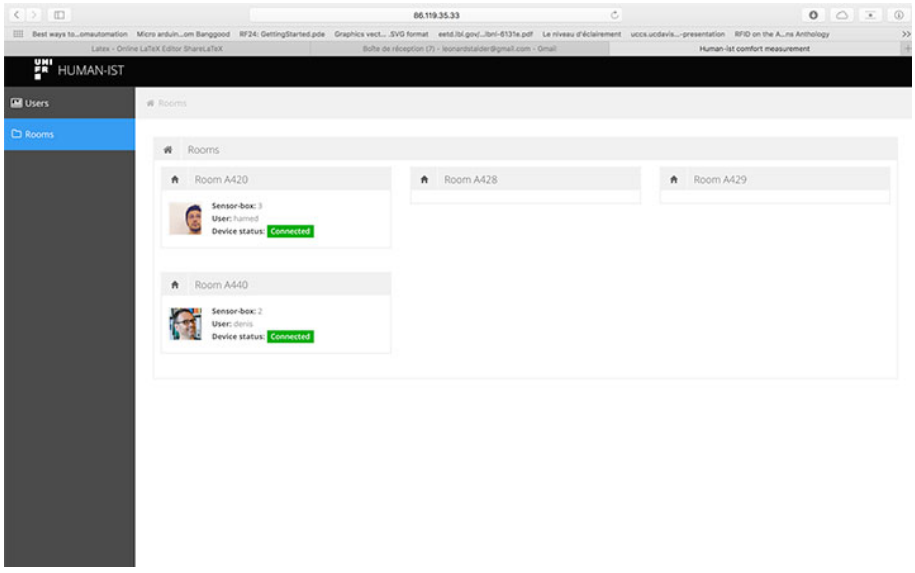

Figure 31: User-interface : display the rooms

These two screens are the main entry points to have access to environmental data.

From the two presented screens, we can show the details of a sensor node. We displayed the different current values that are sensed from a sensor node and the user that is close to the box. Different charts present the history of the values. On Figure 32 you can find the screen of the details of a node. The boxes represent the current values and the charts show the history of each value.

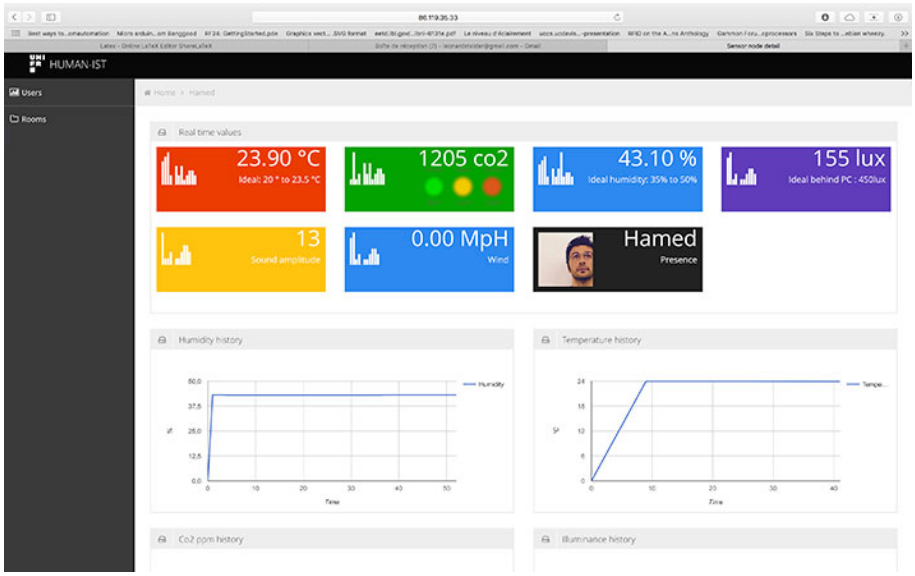

Figure 32: Sensor Values table example

The values are updated in real-time and this thanks to the architecture that we created.

8.3.2 Client Website Architecture

To develop our website we use JavaScript, HTML5 and CSS3. We do not use a server-side language.

We have 3 pages : index.html lands on the user-selection, rooms.html lands on the rooms-selection and sensor.html gives the details of a Comfort Box. A JavaScript script acts as MQTT client to render real-times values in the HTML. In Appendix J you can see a part of the script that subscribes to the broker.

8.3.3 Client Website Deployment

After finishing the web-project we decided to deploy it on the web. We used the same virtual machine that runs the python script and the database. We installed an Apache web-server and deployed the project. The website is online and can be consulted[10].

[10] `http://86.119.35.33/app/rooms.html`

9 Performance Tests

In this section we will show you the different tests we did on our system. Our system is a prototype but it's important to test it. The goal is to show that each step from the capture to the visualization and storage is done.

9.1 Performance Tests Environment

We tested our system in the Human-ist institute offices in Fribourg. The offices are composed of an open-space of 6 meters per 15 meters and two individual offices of 3 meters on 5 meters. The open-space regroups 10 work desks, the first individual office two work desks and the last one one work desk. We decided to produce 3 Comfort Boxes and to put them on an desk in each room. One Internet gateway is placed in the floor to capture the radio-messages.

We took time to make our system easy to deploy. Indeed, you just need to plug the devices to begin to use the system. First we plugged the Internet gateway in the middle of the rooms to an outlet. The Internet gateway contains scripts that connect automatically to the Internet, that run the MQTT client and that start to listen for incoming radio-messages. In the second phase we have to plug the three Comfort Boxes to outlets via an USB 2.0 cable. The system is really "plug and play" and can be easily deployed in an environment.

We placed the Comfort Boxes on desks that are often used to see the interactions with the user.

9.2 Performance Measurement Procedures

To show if our system was robust enough we decided to track all possible errors and to measure the network performances by defining these following measures :

1. We compute the time that it takes to a data to go from the sensor to the visualization tool
2. We log the possible interruptions of the system

To measure this we added a time-stamp when a message leaves the Comfort Box and another time-stamp when the same message arrives to a MQTT client. We compare then the two time-stamps.

To measure if the system has interruptions we configure a simple tracking system. The Internet gateway contains a try-catch statement and writes the errors in a log file if some errors appear. We do the same thing on the broker, the Python DAO and on the client side.

The final goal was to launch our system during 4 days and to see how the system behaves. During this test-phase we had to check if there were errors on our system.

9.3 Performance Results

9.3.1 Time from Sensor to Visualization

The first goal was to measure the "time to visualization". To measure this we plugged the Arduino-node to a computer. On this computer we ran a script that reads the serial ports. The script is available in Appendix K. The script reads the sensed value, adds a time-stamp and adds an id to the message. The message is then leaving to go to the broker. We do the same with the client: indeed we run a python script that subscribes to the broker. Once a message arrives we add a time-stamp. To finish we compare the two time-stamps and we have the time that the values go from the sensor to the client.

We obtained results by tracking 312 messages from Comfort Box 1 and we lost 0 messages. We computed the average time from the box to the visualization for Comfort Box 1 and obtained the following results in mm:ss:ms :

Average box 1
00:02:88

Table 7: Average time from node to visualization

This is a quite good result but variable depending the network quality and the client's connection. This value provides from our test environment but it can change with other configurations.

9.3.2 System Interruptions or Crashes

The second measure was to check if there are some errors in the log files. We had a problem with the Internet Gateway and could found hundreds of logs like :

Listing 5: Error message in log file

```
1 connect: Network is unreachable
```

This error appeared on the Internet gateway and showed that the Raspberry Pi couldn't connect to the Internet any more. The Internet connection from the Raspberry Pi to the network of the University.

This occurs occasionally at random time. After research we found out that the University network disconnects sometimes the Wifi connection. We can avoid this problem by connecting the Raspberry Pi to an RJ45 cable.

10 Applications

10.1 Research Tool

The first application of our system could be a research tool for better understanding human behaviors related to comfort. Indeed, we observed data that was captured during one day and tried to compare the different measures between them. The first observation showed us a correlation between humidity and temperature. This result is already known and not the most interesting. The main result we found is that the user stays during a long time in an uncomfortable situation before making an action to return to a comfortable situation. Figure 33 shows the evolution of some measured data during one day.

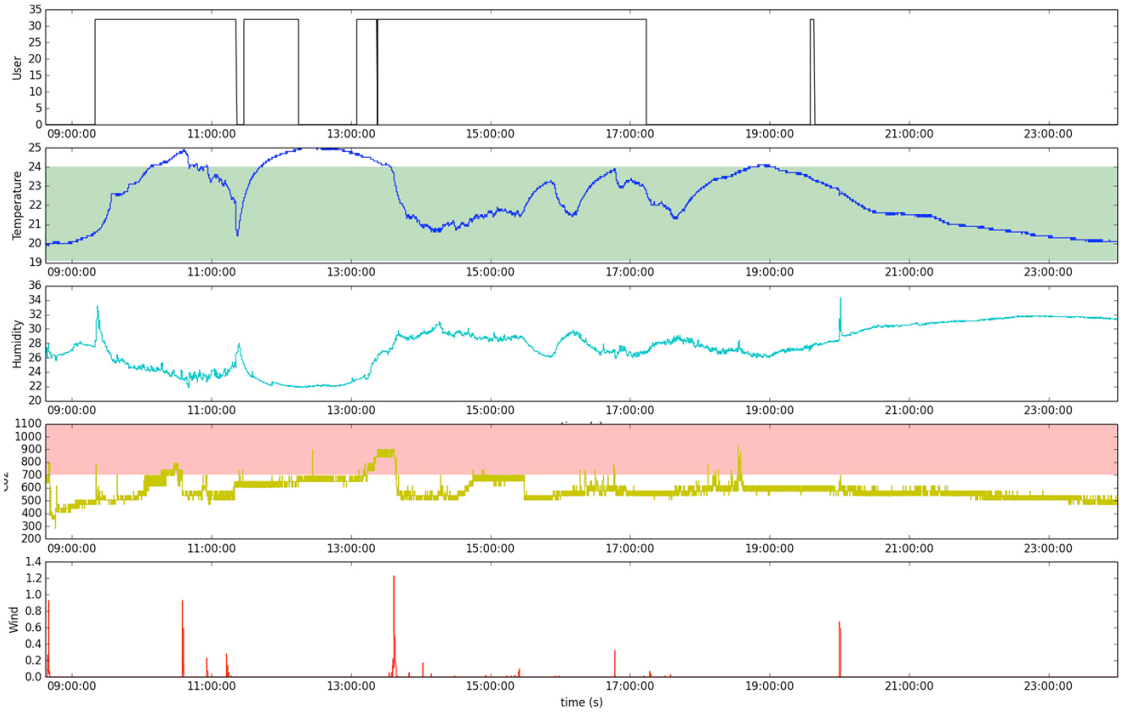

Figure 33: Some values of a day

The first sub-graph represents the presence of the user. If the value corresponds to 32, it means that an inhabitant is present.

The second and third sub-graphs represent the temperature and the humidity. We clearly see that these two values are correlated. The green area in the temperature sub-graph represents the optimum indoor temperature to work according to the World Health Organization. This range should ensure the health, safety, and welfare of the occupants.

The fourth sub-graph represents the CO2 tenure in PPM in the room. In red you see the limit corresponding to the air saturation.

The last sub-graph shows the air flows in the room. Each pic corresponds to a window-opening.

If we focus only the presence, temperature, CO2 and wind between 9 a.m. and 3 p.m. we obtain the Figure 34. We can clearly see that when a situation of discomfort appears the action to stay comfortable is not directly taken. Indeed, the inhabitant stays 30-40 minutes in a discomfort state before opening a window.

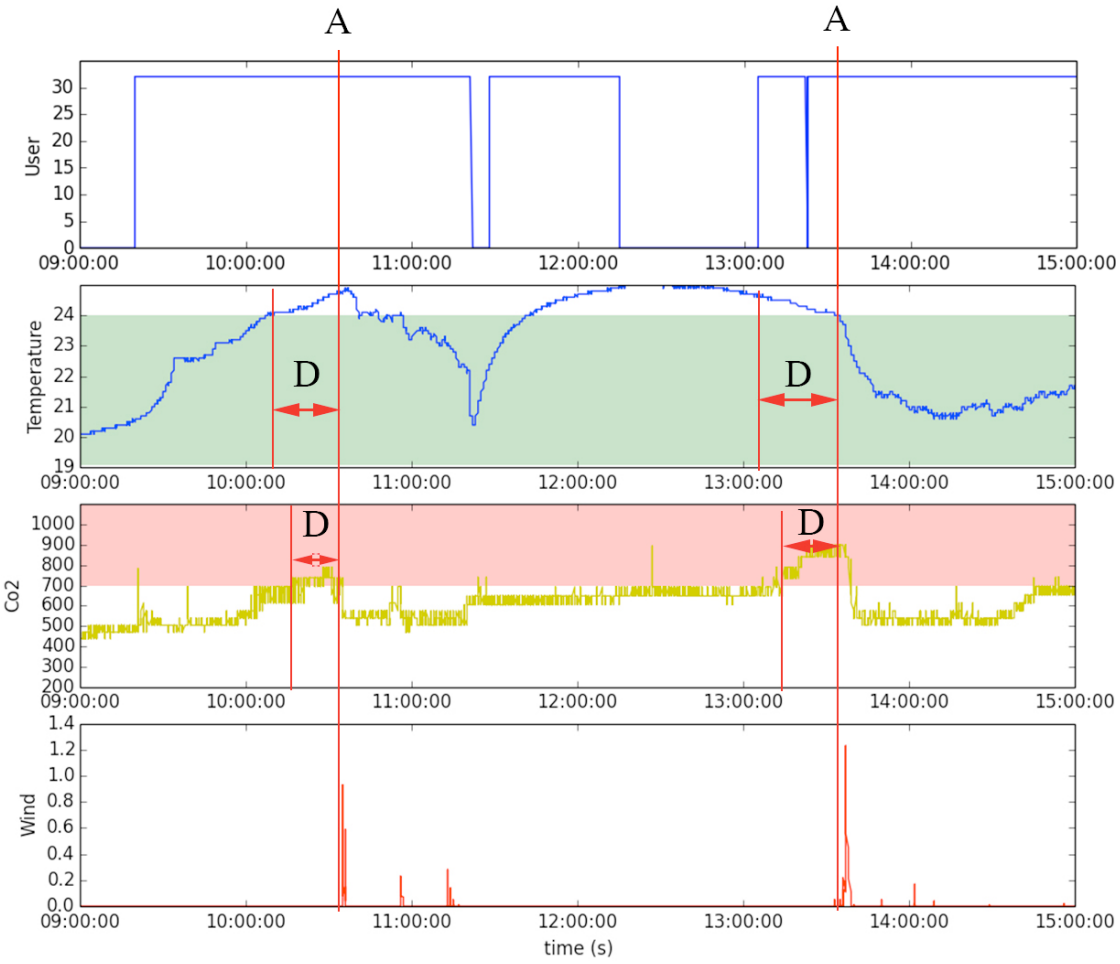

Figure 34: Some values of a day

On Figure 34 the D's represents the time during the user is uncomfortable. The A's represents the actions that are taken to improve his comfort.

In this section, we only analyze one day's data for one user. It would be interesting to do a bigger experiment with more Comfort Boxes. More results and more data will allow us to develop models to avoid this uncomfortable time.

On Figure 35 we build two correlation matrices to find relations between the data. The first matrix corresponds to the different sensed values when the user is present and the second matrix corresponds to the values when the user is not at his working place.

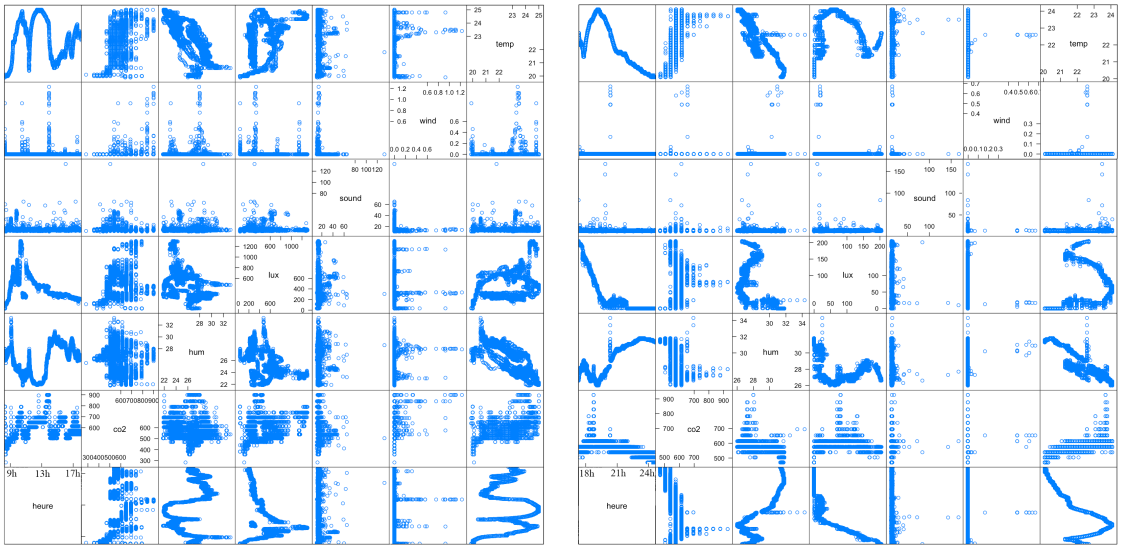

Figure 35: Left : correlation matrix when the user is present, right: correlation matrix when nobody is in the room

We can see that the environmental values change a lot when a user is present. When the user is not present we can observe that that evolution of the data is more linear. We can suppose two things: First, that the user isn't satisfied by the environmental conditions and tries to improve them constantly and second that the user interacts multiple times per day with the building.

To test these suppositions we could imagine linking our data-acquisition system with a building management system to minimize the user action's and improve the building intelligence.

10.2 Commercial Solution

The second idea was to produce a commercial product by improving some parameters and adding a user feedback and an user input system. The concept is to capture data, compute the indexes for thermal, visual, respiratory and acoustic comfort and then to display it directly on the Comfort Box. Then the user can choose to accept the feedback or refuse it on clicking on the comfort's dimension that he refuses.

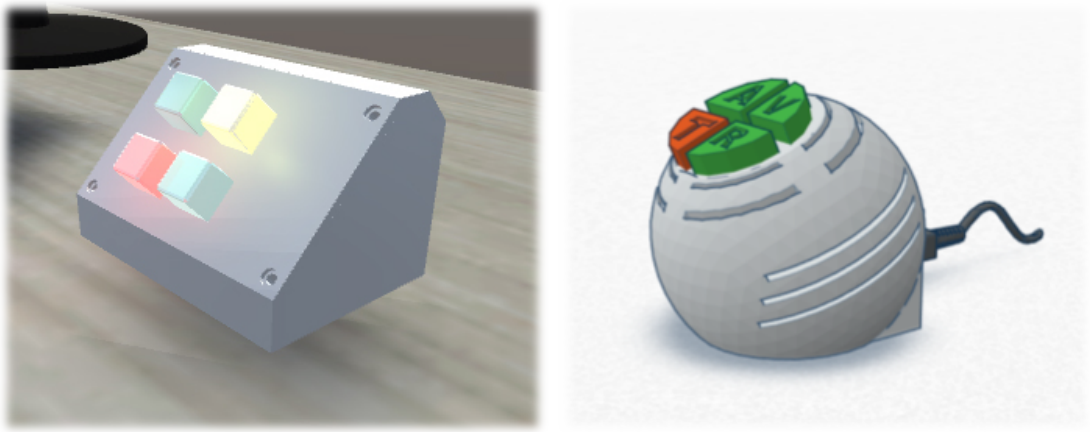

Figure 36: Concept of an improved Comfort Box

Figure 36 shows two examples of improved Comfort Boxes that could be used for commercial purposes. Each button corresponds to a dimension of the comfort (visual, thermal, acoustic and respiratory).

Behind each button light changes from red to green depending on the comfort level. If a button is red, it means that this dimension is not comfortable and the user should do an action to improve this dimension. If he does not agree with the feedback he can push the button and the system ignores the alert.

These solutions are interactive and should help the inhabitants to stay comfortable by warning them before they are uncomfortable. We can also imagine reusing the collected data to interact directly on the building and create a sort of smart building.

11 Conclusions

11.1 Contributions

The main goal of this project was to understand and capture human indoor comfort. As mentioned in the introduction we divided the project into three parts: The first was to understand human indoor comfort and human building interactions in order to define the parameters that we have to sense. The second part consisted in designing a system that collects data in order to compute human indoor comfort and human indoor interactions. The last part consisted in testing the system. Different studies and researchers have built systems to track comfort, but they are generally focused on thermal comfort and they do not care about visual, acoustic or respiratory comfort.

We proposed a complete acquisition system that allows to compute thermal, visual, respiratory and acoustic comfort. We also care about the user by tracking the presence near the devices that sense the comfort. This allows us in a future work to calculate user comfort profiles. To summarize, we wanted to compute the comfort level that surrounds an inhabitant. We build smart sensor nodes to put them on each work desk. The nodes contain different sensors to sense data about comfort but also an RFID tag to track the user's presence.

With this project, we tried to define personalized human indoor comfort by measuring it near to the inhabitant. We build on the top of comfort theories, a system from scratch by using the amazing Arduino-compatibles hardware. We printed boxes to hold the sensors, designed a printed circuit board and a network to store the data on a remote server. We also designed a live visualization tool to display the sensed data. The selection of the hardware and the electronic part of this project were a challenge that we had underestimated. Indeed, we encountered problems with the accuracy of the sensors and had to test a lot of different hardware pieces.

The entire system was tested and we just had a problem with the University WiFi network. It was a first prototype system and we found out that it was fully functional. We can improve the system and also add an embedded feedback functionality. Indeed, the user needs to connect to a website to have his information. During the test phase, we connect environmental indoor data.

To conclude, we could prove that the system has a usability in the indoor environment. Moreover, we could demonstrate that people tend to wait before improving their comfort level. Potential applications for such a technology are unlimited and yet to be explored.

11.2 Future Work

11.2.1 Hardware Improvements

The hardware choices were not easy and we made some mistakes with some sensors. Indeed, the CO2 sensor offered not the best accuracy and the RFID reader not an interesting reading range. We also wanted to improve the size of the box. Indeed, the actual size is quite big and we could improve it by doing it smaller. The printed circuit board could also be improved by adding headers to plug direct the sensors on it.

11.2.2 System Improvements

The radio-frequency network could be avoided. Indeed, we used it at the beginning of the project to save energy because we wanted autonomous sensors nodes. Since we decided that the sensors nodes were alimented with a cable we could imagine that a Wi-fi solution would be more appropriate. This solution would allow us to connect the nodes direct to the broker.

11.2.3 Concept Improvements

It would be interesting to add a user feedback directly on the Comfort Box. We can imagine an interface where the user can directly see if he is comfortable or not and some buttons or a touch-screen to give inputs on his feelings.

References

[1] Monika Frontczak, Pawel Wargocki, *Literature survey on how different factors influence human comfort in indoor environments*, Technical University of Denmark, 2800 Kgs. Lyngby, Denmark, 2010.

[2] Swiss Federal Office of Energy SFOE, *Energie-Forschung 2012. berblicksberichte der Programmleiter*, Swiss Federal Office of Energy SFOE, Muehlestrasse 4, Bern, 2013.

[3] K.L.Ku, J.S.Liaw, M.Y.Tsai,and T.S.Liu, *Energie-Forschung 2012. berblicksberichte der Programmleiter*, IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, 2015.

[4] Katharine Kolcaba *Comfort Theory and Practice: A Vision for Holistic Health Care and Research*, Springer 2002.

[5] Luian Zhang, Martin G. Helander *Identifying Factors of Comfort and Discomfort in Sitting*, Human Factors: The Journal of the Human Factors and Ergonomics Society, 1996.

[6] Roulet, C.A., et al. *Perceived health and comfort in relation to energy use and building characteristics.*, Building Research and Information, 2006.

[7] Bernard Nagengast *Early Twentieth Century Air-Conditioning Engeneering*, ASHRAE Journal, 1999.

[8] Bjarne W. Olesen *INTERNATIONAL STANDARDS FOR THE INDOOR ENVIRONMENT*, Technical University of Denmark, International Centre for Indoor Environment and Energy, Nils Koppels Alle, DTU 2004 .

[9] ASHRAE *ASHRAE Handbook*, ASHRAE, Atlanta, GA, 2005.

[10] Ashrae *Ventilation for acceptable indoor air quality*, ANSI/ASHRAE Standard 2013 .

[11] Magali BODART *ASSURER LE CONFORT VISUEL*, UCL Architecture et Climat 2013 .

[12] Taeyon Hwang, Jeong Tai Kim *Effects of Indoor Lighting on Occupants Visual Comfort and Eye Health in a Green Building*, Department of Architectural Engineering, Kyung Hee University, Yongin-si, Gyeonggi-do 446-701, Republic of Korea 2011 .

[13] CSTC *Le confort visuel et la normalisation* , CSTC 2003 .

[14] International Organization for Standardization *The Noise Rating - NR*, International Organization for Standardization (ISO/R 1996:1971 withdrawn) .

[15] Kristian Fabbri *Indoor Thermal Comfort Perception*, SpringerBriefs in Applied Sciences and Technology 2015 .

[16] Jeff Hsu, Prashanth Mohan, Xiaofan Jiang, Jorge Ortiz, Sushant Shankar, Stephen Dawson-Haggerty, and David Culler *HBCI: Human-Building-Computer Interaction*, University of California, Berkeley, 2010.

[17] Tom Rodden, Steve Benford *The evolution of buildings and implications for the design of ubiquitous domestic environments*, he University of Nottingham, 2003.

[18] KMC Controls *UNDERSTANDING BUILDING AUTOMATION AND CONTROL SYSTEMS*, KMC Controls 2011.

[19] Wai Leung Tse, Wai Lok Chan *A distributed sensor network for measurement of human thermal comfort feelings*, The Hong Kong Polytechnic University, Hung Hom, Hong Kong SAR, 2008.

[20] Farrokh Jazizadeh, Ali Ghahramani, Burcin Beceril-Gerber, Tatiana Kichkaylo, Michael Orosz *Human-Building Interaction Framework for Personalized Thermal Comfort-Driven Systems in Office Buildings*, Viterbi School of Engineering, University of Southern California, KAP 217, 3620 S. Vermont Ave., Los Angeles, 2014.

[21] Qingyuan Zhu, Jian Yi, Shiyue Sheng, Chenglu Wen, Huosheng Hu *A Computer-Aided Modeling and Measurement System for Environmental Thermal Comfort Sensing*, Xiamen University, Xiamen 361005, China, 2014.

[22] Anuj Kumar, I.P. Singh, S.K. Sudl *An approach towards development of PMV based thermal comfort smart sensor*, Instrument Design and Development Centre, Indiant Institute of Technology), New Delhi 110016, India, 2010.

[23] Burcin Becerik-Gerber, Farrokh Jazizadeh *Toward Adaptive Comfort Management in Office Buildings Using Participatory Sensing for End User Driven Control*, University of Southern California, 2012.

[24] Jan Wienold, Jens Christoffersen *Evaluation methods and development of a new glare prediction model for daylight environments with the use of CCD cameras*, Frauenhofer Institute for Solar Energy Systems, Freiburg, Germany 2006 .

[25] Michele De Carlik, Valeria De Giuli, Roberto Zecchin *Review on visual comfort in office buildings and influence of daylight in productivity*, Dipartimento di Fisica Tecnica, Padova 2008.

[26] Michael McRoberts *Beginning Arduino*, Technology in action 2010 .

[27] Alison Watson *The Big Book of Raspberry Pi*, Createspace 2013.

[28] Mosquitto community *An Open Source MQTT v3.1/v3.1.1 Broker*, http://mosquitto.org 2015.

[29] Harry Henderson *Encyclopedia of Computer Science And Technology*, New York, NY: Infobase Publishing, 2009.

[30] Wikihow Community *How to Create Printed Circuit Boards*, http://www.wikihow.com/Create-Printed-Circuit-Boards.

Appendices

We separate most code listings from the main document for clarity and readability. These appendices contain information for developers and persons interested in the technical part of this project.

A Met Value Table

This table gives metabolic rate values depending on the activity that is done. These values are used to compute the PMV.

Activity	w/m2	Met
Reclining	46	0.8
Seated relaxed	58	1.0
Clock and watch repairer	65	1.1
Standing relaxed	70	1.2
Sedentary activity (office, dwelling, school, laboratory)	70	1.2
Car driving	80	1.4
Graphic profession - Book Binder	85	1.5
Standing, light activity (shopping, laboratory, light industry) 93	1.6	
Teacher	95	1.6
Domestic work -shaving, washing and dressing	100	1.7
Walking on the level, 2 km/h	110	1.9
Standing, medium activity (shop assistant, domestic work)	116	2.0
Building industry -Brick laying (Block of 15.3 kg)	125	2.2
Washing dishes standing	145	2,5
Domestic work -raking leaves on the lawn	170	2.9
Domestic work -washing by hand and ironing (120-220 W/m2)	170	2.9
Iron and steel -ramming the mould with a pneumatic hammer	175	3.0
Building industry -forming the mould	180	3.1
Walking on the level, 5 km/h	200	3.4
Forestry -cutting across the grain with a one-man power saw	205	3.5
Agriculture -Ploughing with a team of horses	235	4.0
Building industry -loading a wheelbarrow with stones and mortar 275	4.7	
Sports -Ice skating, 18 km/h	360	6.2
Agriculture -digging with a spade (24 lifts/min.)	380	6.5
Sports -Skiing on level, good snow, 9 km/h	405	7.0
Forestry -working with an axe (weight 2 kg. 33 blows/min.)	500	8.6
Sports -Running, 15 km/h	550	9.5

B Clothing Insulation Table

This table gives the recommended clothing levels for different ensembles.

Ensemble Description	$l_{cl}(clo)$
Walking shorts, short-sleeved shirt	0.36
Trousers, short-sleeved shirt	0.57
Trousers, long-sleeved shirt	0.61
Same as above, plus suit jacket	0.96
Same as above, plus vest and T-shirt	0.96
Trousers, long-sleeved shirt, long-sleeved sweater, T-shirt	1.01
Same as above, plus suit jacket and long underwear bottoms	1.30
Sweat pants, sweat shirt	0.74
Long-sleeved pajama top, long pajama trousers, short 3/4 sleeved robe, slippers	0.96
Knee-length skirt, short-sleeved shirt, panty hose, sandals	0.54
Knee-length skirt, long-sleeved shirt, full slip, panty hose	0.67
Knee-length skirt, long-sleeved shirt, half slip, panty hose, long-sleeved sweater	1.10
Knee-length skirt, long-sleeved shirt, half slip, panty hose, suit jacket	1.04
Ankle-length skirt, long-sleeved shirt, suit jacket, panty hose	1.10
Long-sleeved coveralls, T-shirt	0.72
Overalls, long-sleeved shirt, T-shirt	0.89
Insulated coveralls, long-sleeved thermal underwear, long underwear bottoms	1.37

C Illuminance Levels for Different Tasks

This table gives the ideal illuminance levels for different tasks. This table is done by the Illuminating Engineering Society.

lux	Characteristics of Activity	Representative Activity
50	Interiors rarely used for visual tasks	Cable tunnels, nighttime sidewalk
100	Interiors with minimal demand for visual acuity	Corridors, changing rooms
200	Interiors with low demand for visual acuity	Foyers, entrances, dining rooms
300	Interior with some demand for visual acuity	Libraries, sports halls
500	Interior with moderate demand for visual acuity	Computer work, reading
750	Interior with demand for good visual acuity	Drawing offices, chain stores
1000	Interior with demand for superior visual acuity	Detailed electronics assembly
1500	Interior with demand for maximum visual acuity	Hand tailoring, precision assembly

D Communication Module Comparison

This table shows a comparison of the different communication chips that we could use for the sensor node.

Wi-Fi	ZigBee	nRF24
No gateway needed	Gateway needed	Gateway needed
Range: 100m (i: 10-30m)	Range: 100m (i: 10-20m)	Range: 300m (i: 30-100m)
Speed: max 54Mbps	Speed: max 2Mbps	Speed: max 11Mbps
Con: 380mA	Con: 8-12mA	1-260mA
Sleep mode: 12uA	Sleep mode: 22uA	Sleep mode : 2 uA
Max nodes: 2007	Max nodes: 1000	Max nodes : 65000
No micro-controller needed	Micro-controller needed	Micro-controller needed
8$	20$+25$	4$+25$

E Wiring Illustration of the Primary Sensor Node

This picture represents the wiring illustration of our different components. We do it to produce our Printed Circuit Board.

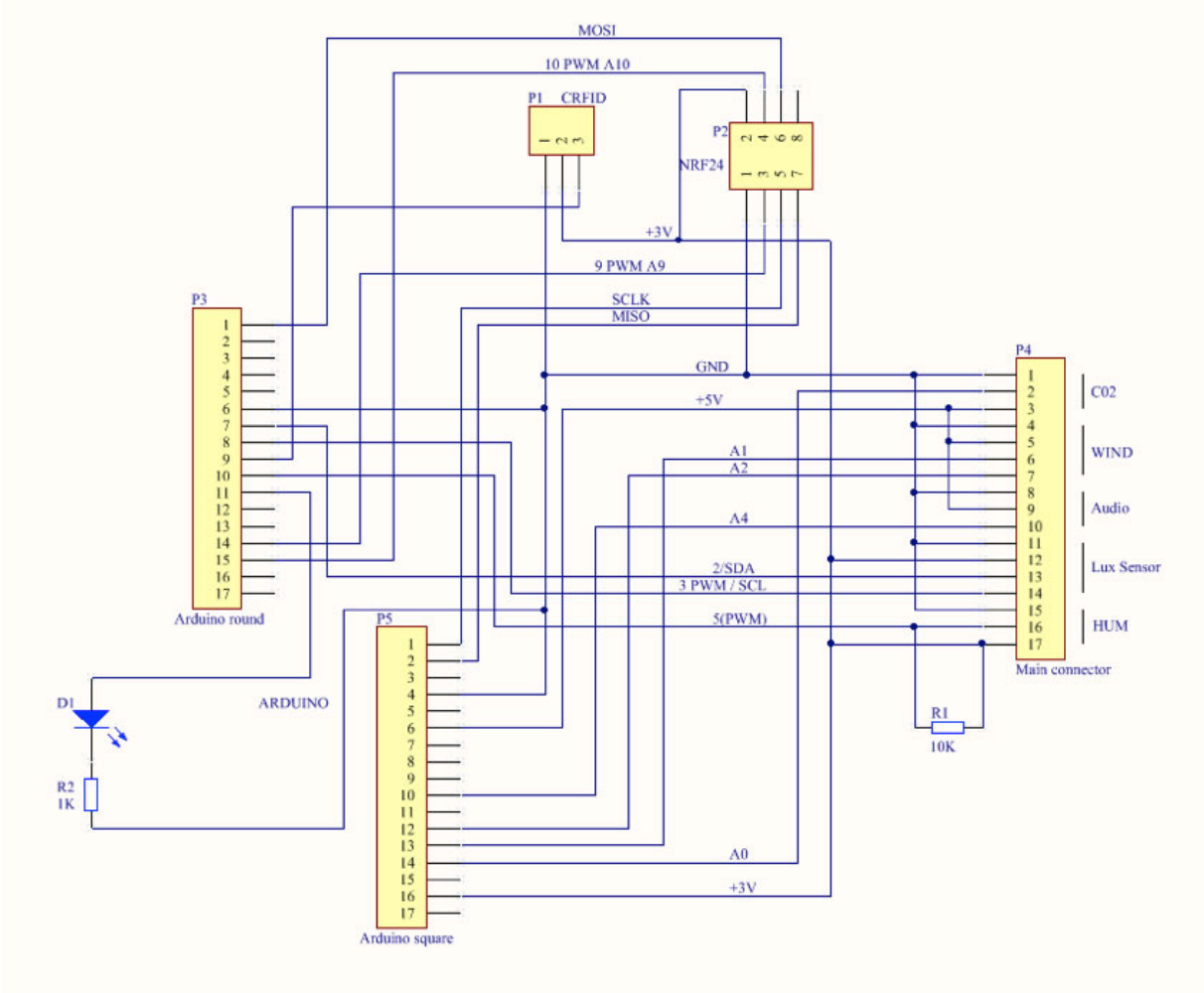

Figure 37: Wiring of the Comfort Box

F Node Code Example

```
// FILE: final_arduino_node1.ino
// AUTHOR: Leonard Stalder
// PURPOSE: comfort box code : senses data about comfort
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <RF24_config.h>
#include <printf.h>
#include "MQ135.h"
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_TSL2561_U.h>
#include "DHT.h"
#include <MFRC522.h>

#define RST_PIN   13    // pin reset for RFID
#define SS_PIN    4   // pin digital for RFID data
#define MQ135_PIN 0 // what pin we are connected for gaz sensor
#define DHTPIN 5 // what pin we are connected to for temperature
#define DHTTYPE DHT22   // type of temperature sensor DHT 22 OR 11
#define PIN_ANALOG_IN A4 // Define hardware connections for sound detector
#define analogPinForRV    A1   // change to pins you the analog pins are using
#define analogPinForTMP   A2 // temperature pin
#define NODEID 3 // unique node id

MFRC522 mfrc522(SS_PIN, RST_PIN); // Create MFRC522 object for RFID
DHT dht(DHTPIN, DHTTYPE); //temperature/humidity sensor object
Adafruit_TSL2561_Unified tsl = Adafruit_TSL2561_Unified(TSL2561_ADDR_FLOAT, 12345); //lux sensor object
MQ135 gasSensor = MQ135(MQ135_PIN); //gas sensor object
RF24 radio(9, 10); //Set up nRF24L01 radio on SPI bus plus pins 9 & 10
const uint64_t pipe_writing = 0xF0F0F0F0E1LL; //Radio address to write
const uint64_t pipe_reading = 0xF0F0F0F0D2LL; //Radio address to receive

// messages structure sended to gateway
struct payload {
  unsigned co2: 16;
  unsigned user: 16;
  unsigned sound: 16;
  unsigned wind: 16;
  unsigned free: 16;
  unsigned lux: 16;
  unsigned hum: 12;
  unsigned temp: 12;
  unsigned id: 8;
} payload;

const float zeroWindAdjustment =  .01; // Wind sensor calibaration near to zero = smaller value and vice-versa.
int TMP_Therm_ADunits;  //temp termistor value from wind sensor
float RV_Wind_ADunits;    //RV output from wind sensor
float RV_Wind_Volts;
unsigned long lastMillis;
int TempCtimes100;
float zeroWind_ADunits;
float zeroWind_volts;
float WindSpeed_MPH;
```

```
56
57 void setup() {
58   Serial.begin(9600);
59   //while (!Serial);
60   dht.begin();
61   tsl.begin();
62   tsl.enableAutoRange(true);
63   tsl.setIntegrationTime(TSL2561_INTEGRATIONTIME_101MS);
64   SPI.begin();       // Init SPI bus
65   mfrc522.PCD_Init();   // Init MFRC522
66   mfrc522.PCD_AntennaOff();
67   pinMode(6, OUTPUT);
68   radio.begin(); // Start radio module
69   radio.setAutoAck(1, 1);
70   radio.setPayloadSize(16);
71   radio.setChannel(90);
72   radio.setRetries(15, 15);
73   radio.setDataRate(RF24_250KBPS);
74   radio.setCRCLength(RF24_CRC_16);
75   radio.openWritingPipe(pipe_writing);
76   radio.openReadingPipe(1, pipe_reading);
77 }
78
79 void loop() {
80   float h = dht.readHumidity();
81   float t = dht.readTemperature();
82   payload.temp = (int)(t * 10);
83   payload.hum = (int) (h * 10);
84   payload.id = NODEID;
85   sensors_event_t event;
86   tsl.getEvent(&event);
87   float ppm = gasSensor.getPPM();
88   payload.co2 = ppm;
89   payload.lux = (int) event.light;
90   payload.free = 0;
91   payload.sound = 12;
92   payload.user = 0;
93   payload.wind = 0;
94   TMP_Therm_ADunits = analogRead(analogPinForTMP);
95   RV_Wind_ADunits = analogRead(analogPinForRV);
96   RV_Wind_Volts = (RV_Wind_ADunits *  0.0048828125);
97   TempCtimes100 = (0.005 * ((float)TMP_Therm_ADunits * (float)
         TMP_Therm_ADunits)) - (16.862 * (float)TMP_Therm_ADunits) + 9075.4;
98   zeroWind_ADunits = -0.0006 * ((float)TMP_Therm_ADunits * (float)
         TMP_Therm_ADunits) + 1.0727 * (float)TMP_Therm_ADunits + 47.172; //
         13.0C  553  482.39
99   zeroWind_volts = (zeroWind_ADunits * 0.0048828125) - zeroWindAdjustment;
100  WindSpeed_MPH =  pow(((RV_Wind_Volts - zeroWind_volts) / .2300) , 2.7265)
         ;
101
102  int sound_value = analogRead(PIN_ANALOG_IN);
103  payload.sound = sound_value;
104  payload.wind = (int)(WindSpeed_MPH * 100);
105  delay(100);
106  mfrc522.PCD_AntennaOn(); // set RFID reader to on
107  delay(10);
108  // Look for new cards
109  if (  mfrc522.PICC_IsNewCardPresent()) {
110    digitalWrite(6, HIGH);
111    if ( mfrc522.PICC_ReadCardSerial()) {
112      // Info about the rfid tag
```

```
113         byte x = *(mfrc522.uid.uidByte);
114         Serial.println(x);
115         payload.user = x;
116         mfrc522.PCD_AntennaOff(); // set RFID reader to on
117       }
118     } else {
119       digitalWrite(6, LOW);
120       Serial.println(" no ");
121       payload.user = 0;
122     }
123
124     delay(300);
125     radio.powerUp();
126     bool ok = radio.write( &payload, sizeof(payload));
127     delay(10);
128     radio.powerDown();
129     delay(1000);
130
131 }
```

G Install Mosquitto 1.4.2 with Websockets on Ubuntu

The first step consists in preparing the environment:

```
$ sudo apt-get update
$ sudo apt-get install build-essential python quilt devscripts python-
    setuptools python3
$ sudo apt-get install libssl-dev
$ sudo apt-get install cmake
$ sudo apt-get install libc-ares-dev
$ sudo apt-get install  uuid-dev
$ sudo apt-get install daemon
```

The second step consists of installing libwebsockets to enable web-sockets connexions to our server:

```
$ sudo wget http://git.libwebsockets.org/cgi-bin/cgit/libwebsockets/
    snapshot/libwebsockets-1.4-chrome43-firefox-36.tar.gz
$ tar zxvf libwebsockets-1.4-chrome43-firefox-36.tar.gz
$ cd libwebsockets-1.4-chrome43-firefox-36
$ mkdir build
$ cd build
$ cmake ..
$ sudo make install
```

The third step consists of installing mosquitto:

```
$ wget http://mosquitto.org/files/source/mosquitto-1.4.2.tar.gz
$ tar zxvf mosquitto-1.4.2.tar.gz
$ cd mosquitto-1.4.2
```

Go to config.mk and change the line $WITH\_WEBSOCKETS := no$ to $WITH\_WEBSOCKETS := yes$. Then execute the following commands

```
$ make
$ make install
$ cp mosquitto.conf /etc/mosquitto
```

The next step consists of the configuration of mosquitto. First you have to add the following lines to /etc/mosquitto/mosquitto.conf:

```
port 1883
listener 9001
protocol websockets
```

To finish just reboot your system and execute these commands :

```
$ mosquitto -c /etc/mosquitto/mosquitto.conf
```

Your broker is now running and can receive messages and resend messages through web-sockets.

H Python MQTT to MySQL Script

```
1  import paho.mqtt.client as mqtt
2  import MySQLdb
3
4  con = MySQLdb.connect(host="localhost", # your host, usually localhost
5                        user="root", # your username
6                        passwd="master-2015", # your password
7                        db="master2015") # name of the data base
8
9  # The callback for when the client receives a CONNACK response from the
       server.
10 def on_connect(client, userdata, rc):
11     print("Connected with result code "+str(rc))
12         # Subscribing in on_connect() means that if we lose the connection
               and
13         # reconnect then subscriptions will be renewed.
14     client.subscribe("sensor/+/#")
15
16 # The callback for when a PUBLISH message is received from the server.
17 def on_message(client, userdata, msg):
18     with con:
19         cur = con.cursor();
20         cur.execute("insert into SensorValues(topic, value, time) values('"
               +msg.topic+"',"+str(msg.$
21     print(msg.topic+" "+str(msg.payload))
22
23
24 client = mqtt.Client()
25 print("teat")
26 client.on_connect = on_connect
27 client.on_message = on_message
28 print("sdf")
29 client.connect(host="86.119.35.147", port=1883)
30
31 # Blocking call that processes network traffic, dispatches callbacks and
32 # handles reconnecting.
33 # Other loop*() functions are available that give a threaded interface and
       a
34 # manual interface.
35 client.loop_forever()
```

I Install MySQL

The first step consists in preparing the environment:

```
1 $ sudo apt-get update
2 $ sudo apt-get install mysql-server
```

This command installs all needed packages for MySQL. We are prompted to set a password and an user. We set as user "root" and as password "master-2015".

The second step consists in installing the python packages to communicate with the database.

```
1 $ sudo apt-get install python-mysqldb
```

The next goal is to create a new database user and a new table.

```
1 $ sudo mysql -u root -p
```

When launching MySQL you have to enter your password and your user-name. The next step is to create a database with the $CREATEDATABASE$ command and to create a new table.

J JavaScript MQTT client code example

Example of a script:

```
client = new Paho.MQTT.Client("86.119.35.147", 9001, "clientId");
// set callback handlers
client.onMessageArrived = onMessageArrived;
client.connect({onSuccess:onConnect});
// global variables

// called when the client connects
function onConnect() {
  client.subscribe("sensor12/hum");
}

// called when a message arrives
function onMessageArrived(message) {
  if (message.destinationName == "sensor12/hum")
    console.log(message.payloadString);
```

K Python Script Stores Arduino Time-stamp

This is a Python script that catches the serial messages from Arduino and applies a time-stamp:

```
from time import sleep
import datetime
import serial
ser = serial.Serial('/dev/tty.usbmodem1d11', 9600)
counter = 32 # Below 32 everything in ASCII is useless
while True:
      counter +=1
      ser.write(str(chr(counter))) # Convert the decimal number to ASCII
          then send it to the Arduino
      print str(datetime.datetime.now())
      print ser.readline() # Read the newest output from the Arduino
      sleep(.1) # Delay for one tenth of a second
      if counter == 255:
      counter = 32
```

The second script is running on a client computer. The difference between the two time-stamps are the time to client.

```
import paho.mqtt.client as mqtt
import datetime
import MySQLdb
def on_connect(client, userdata, rc):
    print("Connected with result code "+str(rc))
    client.subscribe("sensor/+/#")

# The callback for when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
    print str(datetime.datetime.now())
    print(msg.topic+" "+str(msg.payload))

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.connect(host="86.119.35.147", port=1883)
client.loop_forever()
```