Università degli Studi di Urbino Carlo Bo
Dipartimento di Scienze Pure e Applicate

Corso di Dottorato di Ricerca in: Scienze di Base e Applicazioni
Curriculum: Scienza della Complessità
Ciclo: XXXIV
Settore Scientifico Disciplinare: INF/ 01

Sustainable comfort in indoor environments: global comfort indices and virtual sensors

Coordinatore e supervisore:

Chiar.mo Prof. Alessandro Bogliolo

Dottorando:

Dott. Stefano Riffelli

Anno Accademico 2020/2021

Sustainable comfort in indoor environments: global comfort indices and virtual sensors

Stefano Riffelli

2022

to my family:
Angela, Davide and Giacomo

Contents

Chapter 1

Introduction

Comfort, as well as safety and energy saving, has always been one of the main concerns of the "home and building automation" field or of the wider "indoor environments" domain. The concept of human comfort, i.e., indoor environmental quality (IEQ), is based on the indoor environment perception through the occupants' senses. This is a particularly important aspect since it has been shown to affect the physical and mental status of the occupants (health and comfort). Researchers are placing more focus on the impact of indoor environmental quality on health, performance, and human comfort due to increased worries about socio-economic issues and the environmental sustainability of buildings [1]. Nowadays, building sector stakeholders are increasingly involved in systems capable of acquiring, storing and analysing building data through the Internet of things (IoT) [2]. The high flexibility of new embedded systems allows their application in fields such as indoor environmental quality (IEQ) management and energy savings. It is well-known that people spend many hours of their time indoors. According to a study conducted by the World Health Organization [3], the population in developed countries spends approximately 90% of its time in indoor environments such as homes, offices, schools, etc. In these types of environments, comfort is becoming increasingly important as there is now widespread evidence that it impacts health, well-being, and productivity [4, 5, 6]. It is widely accepted that the user's comfort or Indoor Environmental Quality (IEQ) [7, 8, 9] is made of four core parameters, also known as IEQ elements, IEQ factors or IEQ categories. These are *thermal comfort* [10, 11, 12, 13, 14], *indoor air comfort* or *indoor air quality (IAQ)* [15, 16, 17, 18], *acoustic comfort* [19, 20, 21], and *visual comfort* [22, 23, 24, 25, 26, 27]. Achieving IEQ high levels means preventing the occurrence of Sick Building Syndrome (SBS) [28, 29, 30], building-related diseases, as well as Multiple Chemical Sensitivity

(MCS) and other unrecognised controversial disorders [31]. The availability of a global comfort index (GCI) capable of capturing the many factors that affect the human perception of comfort and returning the quantitative estimate is relevant for many fundamental aspects. These aspects include health, productivity, building renovation, comfort prediction, energy efficiency, and generally understanding and acting on the improvable potential of an indoor environment. Several studies [32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47] identify weightings of these IEQ factors and/or propose an overall comfort index. These GCIs are often assessed either objectively through sensors or subjectively through surveys [48, 49, 50, 51, 21]. Assessing the impact of each IEQ category on overall comfort is challenging for multiple reasons [52].

The purpose of this research is to identify a methodology for predicting perceived comfort from measured physical parameters in a given indoor environment. Firstly, aspects of comfort are investigated in order to understand how it is possible to carry out a global comfort assessment in indoor environments. This study makes it possible to provide a review of the main comfort indices. Secondly, a wireless IEQ logger system is designed, comprising hardware, software components, and data analysis. The process of IEQ data collection and processing is divided into multiple steps: implementation, deployment, data collection, data analysis, model building and characterization. Eventually, a virtual sensor that estimates the perceived global comfort index, automatically choosing and characterizing the model with the lowest error, is developed. This is aimed at studying comfort in optimized control systems for sustainable buildings. The rest of the thesis structure is outlined below.

- **Chapter 2: Comfort.** Overview of the main aspects of comfort and survey of previous work on global comfort indices.

- **Chapter 3: Indoor Environmental Quality logger.** Hardware and software design of the IEQ logger. Design, implementation, deployment, and data collection phases are presented.

- **Chapter 4: Data processing and proposed model.** Description of the phases and models applied to perform the analysis of objective and subjective data. Results from different algorithms are obtained to identify the method that minimizes the distance between a predicted comfort index calculated from sensor data and a perceived comfort index based on questionnaires. Data analysis, model building and characterization phases are presented.

- **Chapter 5: Virtual Sensor.** Development of the virtual sensor, i.e. a software sensor that provides a perceived global comfort index (P-IGCI) estimate, automatically choosing and characterizing the model with the lowest error.
- **Chapter 6: Conclusions and future work.** Final considerations, directions and future work.

Chapter 2

Comfort

A number of overall comfort indices have been proposed by several studies adopting different methodologies. Therefore, a review is necessary in order to identify the best common aspects and the current direction in which they are headed. This chapter is aimed at identifying and analysing studies in order to:

- Understanding the comfort categories that have a major impact on IEQ / overall / global comfort index;
- Identifying the main techniques that have been used;
- Discussing open issues and envisioning a new approach.

The goals can be reached through a 4-step process. Step 1 implies identifying a methodology to be adopted in order to perform the research of the papers on this topic in the available literature (in the "Literature survey" section). Step 2 is an overview of the main concepts concerning the core aspects of comfort and the identification of the most significant indices for every specific comfort category (in each corresponding "IEQ Comfort Categories" section). Step 3 is the review of studies concerning the impact/contribution of different IEQ categories on global comfort; among these studies, those containing an explicit final formula for calculating an overall comfort index (in the "Indoor Environmental Quality and Global Comfort" section) are highlighted. Step 4 is a discussion about this review aimed at answering the pre-established targets (in the "Discussion on Global Comfort Indices" section). *Comfort* is defined as a specific condition of well-being. According to the sensorial perceptions of an individual in an environment, it is determined by temperature, air humidity, noise level and brightness detected within the environment. This definition highlights a distinction between thermal comfort, acoustic comfort and visual comfort.

The *Environmental comfort* is identified with the psychophysical well-being of people in an environment (home, office, museum, shopping centre, etc.); it is a feeling that depends on certain environmental conditions that are largely planned and therefore fall under the responsibility of the designer, for instance in the design, implementation and management phases of a *smart home* or, more in general, of a *smart/green building*.

In addition to energy efficiency in green buildings, many researchers have also become more focused on user comfort, using *thermal* and *visual comfort* in addition to *air quality* as the parameters of most interest [53] [54]. Furthermore, in accordance with the previous environmental comfort definition and considering the attention towards acoustically-isolated buildings (especially in big cities and metropolises), *acoustic comfort* must be added. Therefore, it is possible to identify four core parameters that define a user's comfort or IEQ:

- Thermal comfort
- Indoor air comfort or Indoor Air Quality (IAQ)
- Acoustic comfort
- Visual comfort

For each of these categories, the basics and main indices will be presented.

Anyway, this study part is not intended to provide a review of the indices corresponding to the specific comfort aspects or IEQ factors. Those comfort categories will be analysed only for a better comprehension of the global comfort indices. The focus of this chapter is a review specifically on these global indices and their IEQ comfort categories weighting.

2.1 Literature survey

This search was conducted through Google scholar database [55] by setting any date as the time interval. Most of the results is also drawn from other databases such as sciencedirect.com [56], mdpi.com [57], researchgate.net [58], and others. Research was specifically focused on the following keywords:
"building" OR "buildings" OR "indoor environment" OR "indoor environments" AND
"thermal" OR "heating" AND

"IAQ" OR "indoor air quality" AND
"acoustic" OR "aural" AND
"visual" OR "lighting" OR "luminous" OR "luminosity" AND
"comfort index".

Researches were also carried out with other similar words or synonyms, but this did not change the results number. According to these criteria, the found publications total number was 369. Figure 2.1 shows growing research focus on this topic.

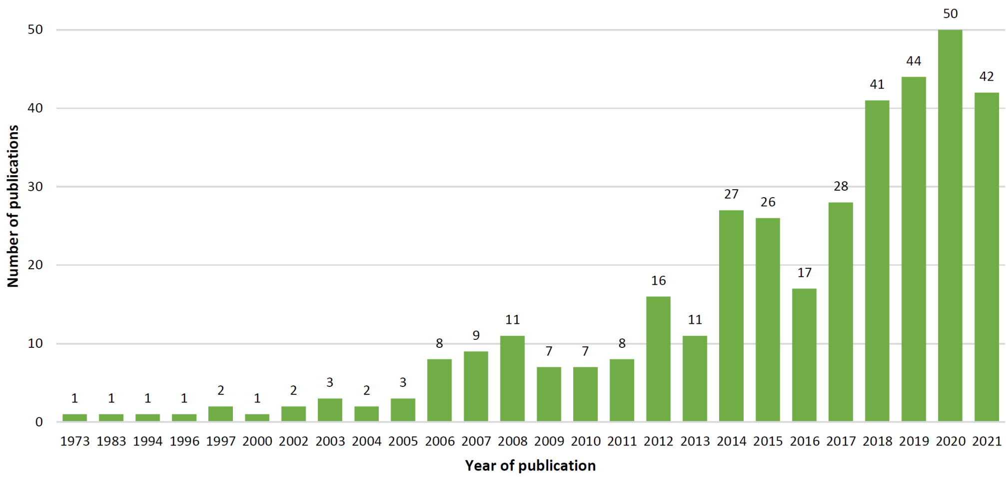


Figure 2.1: Publications number on the topic by year.

Global comfort indices considering two or more comfort categories are identified in the literature by other more specific names such as:

- Overall comfort index
- Global comfort index
- Combined comfort index
- IEQ index

Moreover, some indices do not explicitly refer to "comfort", or they are referred to by other words such as "satisfaction" or "dissatisfaction". Thus, other global indices can be found with other terms such as:

- Overall index
- Global index

- Combined index
- Satisfaction index
- Dissatisfaction index

Incorporating (one by one) these terms into the previous search, 85 effective results were found. Analysing the title and abstract of these studies, the articles selected for review were those:

- with an "assessment", "evaluation", "judgment", "rating", "weight", "influence", "impact" or generally a "correlation" between each IEQ category and the overall index
- with an explicit formula of the global index including the IEQ categories

Thanks to their references, other few articles of interest for the review were also found. In summary, 27 papers were reviewed:

- 25 papers provide 26 different IEQ comfort categories weighting;
- 10 papers provide 9 GCIs / IEQ indices with an explicit formula.

The comfort categories that will be discussed in the next section are often identified in the literature by other labels like "IEQ categories", "IEQ attributes", "IEQ factors", "IEQ aspects", "IEQ elements", "IEQ components", and "IEQ items". However, these always refer to the four categories "Thermal", "IAQ", "Acoustic" and "Visual Comfort".

In the next section, IEQ categories are presented to provide a better reading of the global comfort indices. Obviously, the most widely discussed comfort category will be "Thermal comfort". This is because it presents many aspects and influencing elements, it contains a number of subjective factors, and for years it has been considered the unique (or the most important) comfort category. At the end of each category, the main indices, methods or strategies are outlined.

2.2 Thermal comfort

ASHRAE (American Society of Heating Ventilation and Air-conditioning Engineers) defines thermal comfort as "that condition of mind that expresses satisfaction with the thermal environment and is assessed by subjective evaluation" [59]. In accordance with P. Ole Fanger's studies and theories, thermal comfort in a building depends on the relationships between the *subjective variables* and *environmental variables* [60].

Recent studies about building comfort highlight that, besides such variables, the feeling of comfort is closely linked to the individual's psychological, cultural, and social aspects; it is also linked to weather and his/her adaptation capacities. That is why it is not easy to quantify the state of well-being, as it has to at least consider the age, gender and health of people. This latest theory is known as the *adaptive method* and has been developed by researchers like G.S. Brager, R.J. de Dear, M.A. Humphreys, and J.F. Nicols [61] [62] [63]. Thermal comfort depends on:

- *Objective variables*: air temperature, mean radiant temperature, operating temperature, relative humidity and air speed;
- *Subjective variables*: external parameters (activity being performed influencing the metabolism, clothing insulation), organic factors (age, gender, specific physical characteristics), psychological and cultural factors.

In accordance with the social and environmental conditions, it is also possible to find different levels of acceptance of situations of discomfort. When living in a situation that extends for a long period, individuals can consider environmental situations that could be judged as uneasy in a different context, as "normal". In this context, we will limit our field of research to the description of objective variables (that is to say, environmental variables or physical parameters). The main variables that depend on both the internal and external climatic conditions of buildings and that affect thermal comfort are:

- Air temperature
- Mean radiant temperature
- Operating temperature
- Relative humidity of indoor air
- Air speed

Air temperature: average temperature of the surrounding air against location and time. The ASHRAE 55 standard says that the spatial average takes into account a number of factors for different parts of the body (e.g. ankle, head etc.), varying either for seated or standing occupants. Hence, the temporal average is calculated through 3 minute-intervals, with a minimum of 18 points with the same temporal distance. A dry-bulb thermometer is used to measure the temperature of the air. This is why it is called dry-bulb temperature. It is the most important factor in determining thermal comfort [59].

Mean Radiant Temperature: average weighted temperature of the surfaces that mark the boundaries of the environment, including the effect of incident solar radiation. It affects exchanges by radiation. It is calculated as the average temperature of the walls inside the room, including the ceiling and floor. Mean radiant temperature has the most impact on how heat is felt on a body in addition to air temperature, and if it experiences a cold surface, much heat is emitted through radiation, making it cold. Mean radiant temperature is affected by both temperatures and surrounding surface irradiances, plus the view factor. It could also depend on the surface size that is perceived by the element. The MRT that a person experiences when the sunlight streams in varies according to his/her body portions that are exposed to the sun. The most comfortable condition is considered the one corresponding to a 2 °C higher than the MRT air temperature. An MRT lower than 2 °C is also tolerable when the radiation emitted by the body is almost the same in all directions; this happens only if the surface temperature of the surrounding environment is uniform.

Operating temperature is also defined as the average of air temperature and mean radiant temperature to precisely assess heat exchanges by convection and radiation with a single value.

Relative Humidity of indoor air: RH is the ratio of the amount of water vapour in the air compared to the amount of water vapour that the air could hold at a specific temperature and pressure; therefore, it is measured in percentages. On the one hand, at high RH, the air comes close to the maximum water vapour that it can hold, so evaporation from the skin, and therefore heat loss, is decreased. On the other hand, environments characterised by very low RH are considered uncomfortable due to the effects exerted on the body. The range of recommended indoor humidity is 30-60% in air-conditioned buildings, [64] [65] but new standards, e.g. adaptive model, allow both lower and higher humidity levels, according to other thermal comfort factors.

Air Speed: when air moves, the air temperature may not be affected, but there are thermal effects and heat can be dissipated as follows through the surface of the epidermis:

- higher dissipation of heat as long as air temperature is not as high as the temperature of the epidermis;
- increased evaporation resulting in body cooling.

Air speed is the average to which a body is exposed with respect to location and time according to the ANSI/ASHRAE Standard 55, with air temperature equating to the time average and the surface area exposed to the air equating to the space average [59].

2.2.1 Thermal comfort indices

This section includes the main thermal comfort indices or corresponding models. They are: 2) the Predicted Mean Vote ("PMV") and Predicted Percentage of Dissatisfied ("PPD") (deriving from the Fanger model), 2) the adaptive thermal comfort model, and 3) bioclimatic charts and indices.

1) PMV and PPD.

The Predicted Mean Vote (PMV) and Predicted Percentage of Dissatisfied (PPD) are derived from the Fanger model. These two thermal comfort level indices come from the relationships between human body functions and the feeling of thermal comfort. They are also defined by the EN ISO 7730 standard [10].

The *PMV* is an index assessing the individual's state of well-being by personal and environmental variables. Hence, it is a mathematical function whose result is a number on a scale from -3 (when it is too cold) to +3 (when it is too hot); zero represents the thermal comfort state. Table 2.1 shows different comfort conditions with the corresponding PMV values. Since it is an average index referring to a group of individuals, if the PMV=0, it doesn't mean that the entire group has reached a well-being state.

Table 2.1: PMV values and comfort condition.

	PMV Index
+3	Hot
+2	Warm
+1	Slightly warm
0	Neutral (Comfort)
-1	Slightly cool
-2	Cool
-3	Cold

Fanger described the comfort criteria that have been defined by theoretical, experimental and statistical studies. To calculate PMV, once the air temperature (AT), mean radiant temperature (MRT), relative humidity (RH), air speed (AS), metabolic rate (MR), and clothing insulation (CI) data have been collected, Fanger's equations are applied [60]. The first 4 variables (AT, MRT, RH, AS) are environmental, while the final 2 (MR, CI) are physiological. Figure 2.2 shows the six variables required to calculate the PMV.

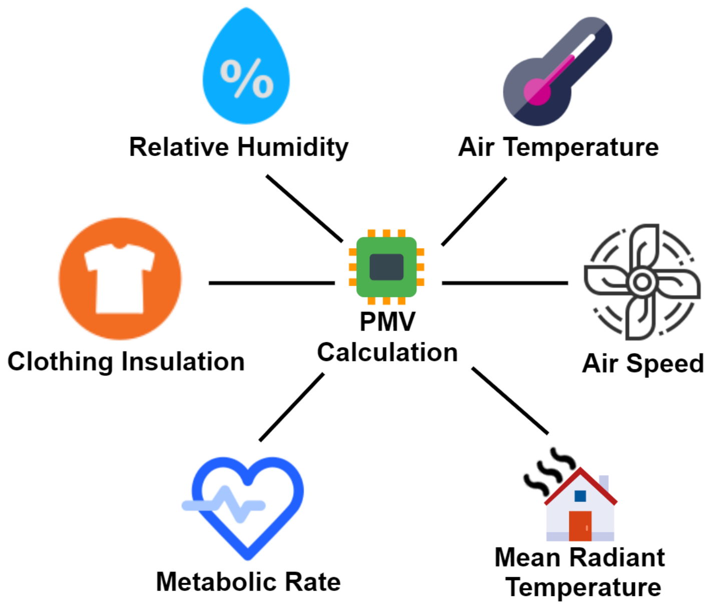


Figure 2.2: Six variables for PMV calculation.

Therefore, PMV is a function of these six variables:

$$PMV = f(AT, MRT, RH, AS, MR, CI) \tag{2.1}$$

The comfort zone is achieved if these parameters are merged to get a PMV of between -0.5 and +0.5 (ASHRAE 55 recommended limits) since thermal neutrality is represented by a PMV of 0 [59]. ISO 7730 [10] (EN 15251:2007 [66]) provides the formula for calculating PMV for a specified condition and expands that limit; it offers a number of different indoor environment ranges. The ISO standard defines the strict limit as -2 to +2, with old buildings ranging between -0.7 and +0.7 and new builds ranging between -0.5 and +0.5. In thermal comfort studies, the PMV is often compared with Thermal Sensation Vote (TSV) [67] [68]. Unlike PMV (which is derived from a mathematical formula), TSV is purely subjective, like the Thermal Comfort Vote (TCV). The TSV is normally assigned the same 7-level scale ranging from -3 to +3, just as with the PMV (see Table 2.1), whereas the TCV is normally based on a 4-level scale from -3 (very uncomfortable) to 0 (comfortable) [69].

Predicted Percentage of Dissatisfied (*PPD*) is an "index defining the quantitative prediction expressed as a percentage of thermally dissatisfied people determined from PMV" [59]. The Percentage of Dissatisfied Persons is always linked to a specific environment.

The relationship between the PMV and PPD is:

$$PPD = 100 \ - \ 95 \cdot exp(- \ 0.03353 \cdot PMV^2 \ - \ 0.2179 \cdot PMV^2) \qquad (2.2)$$

Equation 2.2 is plotted in Figure 2.3.

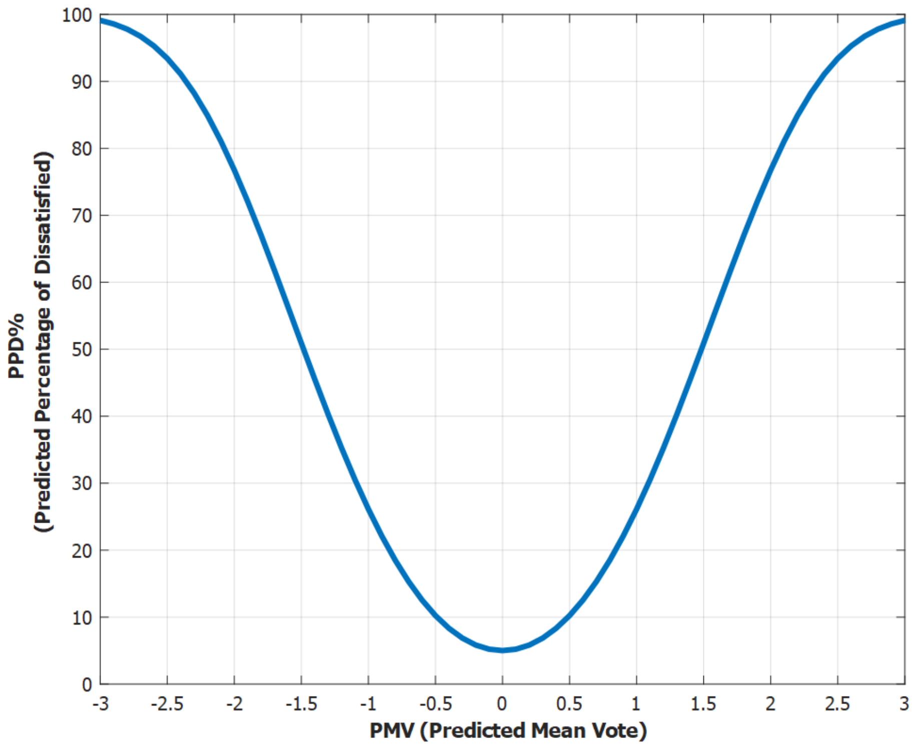


Figure 2.3: The relationship between the PPD and the PMV.

The graph shows that the PPD (function of the calculated PMV) can vary from 5% to 100%. Depending on the occupant's position in the building, the comfort values will be different. In general, in order to comply with the applicable standards, no point occupied in the area should have a PPD of higher than 20%. In accordance with the ISO 7730 standard [10], there are three recommended categories for buildings, while the fourth category is for values outside the criteria [70].

Table 2.2 shows the PMV/PPD percentage values as a function of the building category. Table 2.3 includes a short description [71] for each category.

Table 2.2: Building category, corresponding PMV and PPD% values.

Category (of building or room)	**PMV (Predicted Mean Vote)**	**PPD (Predicted Percentage of Dissatisfied)**
I	$-0.2 < \text{PMV} < +0.2$	< 6
II	$-0.2 < \text{PMV} < +0.2$	< 10
III	$-0.7 < \text{PMV} < +0.7$	< 15
IV	$\text{PMV} < -0.7$ or $\text{PMV} > +0.7$	> 15

Table 2.3: Building category and short description.

Category	**Description**
I	high level of expectation for spaces occupied by a very sensitive person
II	normal level of expectation to be used in new buildings and renovation
III	acceptable level of expectation to be used in existing buildings
IV	values outside the criteria for the above categories to be used only for a limited part of the year.

In recent years, many researchers have questioned the validity of such an approach since it doesn't take a number of important factors into account, including the climatic, cultural, social, and contextual ones. They have introduced the adaptation concept, explaining how each individual's context and thermal history can modify the occupant's expectations and thermal preferences.

2) Adaptive thermal comfort model.

In the adaptive thermal comfort model, the building occupant is not simply viewed as a passive subject, as appeared in the static model (Fanger

PMV); rather, he or she is an active agent interacting at all levels with the environment he/she is in. The adaptive comfort model suggests a correlation between the occupant's comfort temperature (Operative Temperature) within a building and the external air temperature (Prevailing Mean Outdoor Temperature).

The ASHRAE-55 Standard defines prevailing mean outdoor temperatures as adaptive model input variables. They can be obtained by calculating the arithmetic average of the daily outdoor mean temperatures that must be acquired within a range of 7 and 30 consecutive days before the target day [59]. Therefore, the adaptive model introduces both control and response algorithms in order to both improve the occupant's thermal comfort and reduce energy consumption. The basic idea is that, nowadays, it is well known that the human body (including the metabolic rate variation) adapts to both local and seasonal climates. Therefore, the occupants perceive different indoor temperatures as comfortable based on locations and seasons. The adaptive model is based upon measured correlations between the subjective comfort perception of people and indoor temperatures in hundreds of real buildings.

There are 3 types of adaptation:

- Behavioural adaptation: a set of changes performed by a person, with or without being aware of it to change the thermal balance in their bodies and relating to the person's cultural, technological and personal background;

- Physiological adaptation: stress can be reduced if the body is exposed for enough time to a certain environment, but if the standard climate is mild, it will have a negligible impact on how it is perceived;

- Psychological adaptation: past experience and assumptions can change stimulation perceptions and consequent reactions.

Among the three adaptive mechanisms, the behavioural mechanism gives people an active role in maintaining their own comfort since it is directly linked to the human body's thermal balance. Adaptive models of thermal comfort have been implemented into a number of standards like EN 15251 [66] and ISO 7730 [10]. Exact derivation methods and results can be slightly different compared to the ASHRAE 55 adaptive standard; however, they are equal from a material standpoint [7]. As an example, this is the correlation according to EN 15251[66]:

$$OT = 0.33 \cdot PMOT + 18.8 \ [^{\circ}C] \tag{2.3}$$

where OT is optimal daily / hourly Operating Temperature and PMOT is Prevailing Mean Outdoor Temperature. Unlike the Fanger model, the adaptive model considers a wider range of "comfortable" temperatures, with this, allowing a more flexible integration of passive cooling technologies.

3) Bioclimatic charts and indices.

According to the type of activity and relative clothing, bioclimatic analysis methods define the perimeter of the area of thermal well-being. This is to be understood as the combination of environmental and climatic factors in which the thermal feeling is judged as comfortable by over 80% of people.

The main charts are:

- The Olgyay bioclimatic chart (1969) [72]
- Psychrometric bioclimatic chart or Givoni–Milne Bioclimatic chart (1979) [73]

Olgyay collected the results of extensive research, tending to numerically determine the concept of well-being; in 1963, he was the first to place them in a single bioclimatic chart. Olgyay defined well-being as "the situation where no discomfort is felt". The Givoni-Milne chart predicts the indoor thermal comfort conditions of a building in accordance with the outside climate. This takes account of the linear relationship between temperature and vapour pressure of the outside air. The psychrometric chart overlaps the limits of both cooling and heating passive strategies.
Below is a list of the main bioclimatic indices [11]:

- Indices based on a *single element*, e.g. temperature, wind, moisture, pressure, etc.
- Indices based on *temperature and moisture*, e.g. Humidex index (heat index) [74] [75], Humiture Index [76], Apparent temperature [77] [75], Thom index THI (Temperature Humidity Index) [78] [75], etc.
- Indices based on *temperature and wind speed values*, e.g. Steadman Index [79], wind chill index [80] [81], etc.
- Thermal comfort with *more than one parameter* (temperature, humidity and wind speed), e.g. equivalent effective temperature (^{0}TEE) and radiation (^{0}TEER)
- Other bioclimatic indices: total index bioclimatic stress, skin temperature as an index of comfort, tonicity coefficient K_t, weather summer touristic index, Fanger's effort equation, weather classes [11].

The high quantity of bioclimatic indices proves that researchers are striving to express any possible link between the human body and climatic variations in one single formula. Nevertheless, it is evident that thermal comfort is affected by many parameters, including individual, social, physical, and geographical parameters [11]. Therefore, it is clear that a general planetary index for all these conditions does not exist. A single and acceptable formula cannot be defined for all types of conditions (climatic and geographical) to understand which thermal comfort level could suit certain health requirements.

In order to provide an overall picture, the main bioclimatic diagrams and indices were mentioned, but they will not be further discussed in this context since they are more related to the climate and/or to outdoor thermal comfort.

2.3 Indoor air comfort or Indoor Air Quality (IAQ)

For the purpose of quantifying indoor air comfort, reference is made to the Indoor Air Quality (IAQ) parameter. To achieve a good IAQ, it is necessary to monitor certain levels of pollutant concentrations in order to provide adequate ventilation or air recirculation. IAQ defines the indoor (and surrounding) air quality of buildings and structures, taking into account the healthiness and comfort in relation to the occupants. IAQ can be affected by different contaminants like carbon monoxide/dioxide, radon, ozone, cigarette smoke, dust, total volatile organic compounds (TVOCs), chemical substances or any other element that negatively impacts health.

The substances involved in the air quality assessment can be divided into 3 groups:

- Physical pollutants: radon, artificial mineral fibres, non-ionising electromagnetic fields
- Biological pollutants: viruses and bacteria, fungi and moulds, pollens, mites, and bacilli
- Chemical pollutants: divided into organic pollutants (volatile organic compounds) and inorganic pollutants (including carbon monoxide and dioxide, sulphur dioxide, nitrogen dioxide, ozone, etc.)

IAQ has become popular due to the greater awareness of health problems caused by mildew and since it triggers asthma and allergies. IAQ measures indoor air as it affects the potential comfort and health of people. A range of physical disturbances can be caused by biological, physical and chemical pollutants, On the contrary, the quality of the air can increase through the attentive selection of cleaning products and building materials, in addition to proper ventilation and specific air filters. The ideal condition is to have an IAQ that is greater than or equal to the outdoor air quality.

2.3.1 Air comfort indices

In order to achieve good indoor air quality, the main criteria is the minimum ventilation rate. The general requirements to achieve acceptable indoor air quality are similar for both residential and non-residential buildings in a number of standards (like EN16798-1: 2019. One or more of the following can be used as design parameters for indoor air quality:

- Method using the perceived quality of the air;
- Method using pollutant concentration criteria;
- Method using previously-defined ventilation air flow rates.

The designer has to choose between different categories of indoor air quality with each method and define which building category is to be used [7]. CO_2 is a surrogate of human-emitted indoor pollutants. It can be easily measured and correlated to metabolism functions. Atypically high indoor CO_2 levels can cause headaches, tiredness and reduced performance. The levels of outdoor CO_2 are around 400 ppm and the acceptability threshold for indoor CO_2 is 1000 ppm [17]. In most buildings, humans are the main source of indoor CO_2. Its levels prove the adequacy of air ventilation with respect to the number of occupants and their metabolic rates. To avoid any issues, the CO_2 concentration difference between the indoor and outdoor values must not exceed 600 ppm. The National Institute for Occupational Safety and Health (NIOSH) says that low rate ventilation is associated with a CO_2 concentration that exceeds 1000 ppm [16].

One of the best strategies in green buildings to obtain a good IAQ is the adequate use of HVAC (Heating, ventilation, and air conditioning) systems. HVAC is one of the main indoor environmental comfort technologies. The use of HVAC systems allows a good compromise to be reached between IAQ and thermal comfort. These systems can be used in both domestic and business frameworks, and prices are still reasonable for installation, commissioning, operations, maintenance and service.

In general, HVAC is an important part of smart homes, buildings and the like, where the health and safety conditions are regulated in terms of humidity and temperature, exploiting the fresh air coming from the outside.

Ventilation refers to the process of replacing/exchanging air in any space in order to provide high IAQ levels, including oxygen supply, temperature control and the removal of moisture, smells, fumes, heat, airborne bacteria, dust, CO_2, and other gases. Ventilation also removes unpleasant odours and excess moisture thanks to the fresh, outside air; it keeps the air circulation within buildings and prevents indoor air stagnation. The ventilation process includes both indoor air circulation and outdoor air exchange. It is also a decisive factor for having an acceptable IAQ. If necessary, buildings can be ventilated in various ways, including mechanical, forced and natural methods [15].

2.4 Acoustic comfort

In order to obtain good acoustic comfort, it is necessary to analyse the main noise sources and design the solutions for proper acoustic isolation. Acoustic comfort can be defined as the condition where an individual is not disturbed by other sounds/noises and his/her hearing system is not damaged by strong noise exposure. In most buildings, poor acoustic comfort is the most common source of disturbance. This is why it is important to ensure the utmost acoustic comfort during the design and construction of the building, and its performance in relation to noises from the outside and from the neighbouring flats.

When designing green buildings, it is necessary to consider the choice of materials, furniture, types of machinery, fixtures, coatings, etc. to ensure they do not cause noises within the building envelope and guarantee acoustic well-being.

In a confined environment, it is possible to distinguish between outdoor sources and indoor sources.

Outdoor noise sources generally include car traffic and the possible presence of industrial manufacturing activities near the building. The noise produced by these sources propagates through the air and enters the building through its envelope.

Indoor noise sources can be found both in the environment in question and in other neighbouring environments. These sources are:

- Installations (lifts, hoists, hydraulic installations, etc.)
- Appliances
- Radio-television devices

In this case, propagation occurs both through the air and the building's solid parts. Regulations generally take account of the different acoustic disturbance sources. Noise is distinguished as:

- Noise from walls and partitions between indoor units
- Noise from facades
- Noise from footsteps
- Noise from installations that work in constant and alternate modes

The standard EN 12354-part 5 [82] includes the guidelines for noise evaluation during the design phase. A service system emanating high levels of noise can create problems for occupants, compromising building usage. An A-weighted equivalent sound pressure level can be used to measure noise which has been normalised to take the sound absorption of the space into account [7].

2.4.1 Acoustic comfort indices

The acoustic comfort assessment criterion is based on the noise level concept. Hence, the acoustic comfort index depends directly on noise/sound levels. The sound pressure level (measured in decibel, dB) is the air pressure increase, on a logarithmic scale, against a still air situation. The "A-weighted" scale (dBA) is sometimes used to account for differences in how people respond to sound. A sound level of normal tolerability is defined to establish the purpose of the analysed environment and the activities performed there. This is a maximum noise threshold, considered as acceptable since it doesn't cause any discomfort. When this threshold is passed, well-being is lost. The noise source emission control is the fundamental strategy for noise pollution.

On average, in standard housing and, in general, in buildings, noise emissions are due to installations and car traffic. Such emissions can propagate mainly within the building itself, causing disturbance to occupants; in other cases, mainly at outdoor level, they lead to quality degradation in the surrounding environment.

In terms of acoustic emissions, the most relevant installation categories are: air conditioning systems, refrigerating equipment, air handlers, HVAC systems, furnaces and boilers, plumbing and drainage, lifts, hoists, escalators, etc. The aim is to reduce the noise emissions from the installations. The main reference strategies and technologies are:

- Choosing either silent or components that can be acoustically insulated;

- Installing the outdoor components in positions that are shielded from possible sensitive receivers that could be potentially disturbed (e.g. housing, schools, hospitals etc.);

- Soundproofing of technological areas.

As far as noisiness within a green building is concerned, the design process must include both technological and architectural solutions that can achieve the individual's acoustic well-being.

Indices, implemented to improve acoustic comfort, especially in the field of smart/green buildings, concern:

- *Noise Level*: noise due to conversations, noise due to footsteps, noise coming from outside, noise of the ventilation system equipment, noise of the lighting equipment, noise of the office equipment, noise of the furniture and doors;

- *Echo*: echoes in the work environment, echoes in meeting rooms, echoes in conference halls and echoes in the social areas;

- *Acoustic privacy*: acoustic privacy in the work environment and meeting rooms [20].

Several studies have been proposed by researchers to quantify the perception of noise levels in indoor environments. The main design standards are [43]:

- NC (noise criterion curves)

- NCB (noise criterion balanced)

- NR (noise rating)

- PNC (preferred noise criterion)

- RC (room criterion)

- Loudness, loudness level

More details on noise ratings can be found in Bies, Hansen, and Howard [83]. One of the most widely adopted indices for assessing acoustic comfort in this field is the A-weighted continuous equivalent sound pressure level (L_{eqA}), especially in office environments [36].

2.5 Visual comfort

In order to achieve visual comfort, it is necessary to have the correct light quantity, during both daytime and night-time, in order not to tire the eyes.

During daytime, a sufficient quantity of light must be able to enter. Therefore, the number of windows, window size and spacing, position of window shutters, glass selection, etc. have to be right. During both night-time and cloudy days, there has to be proper artificial light.

Nowadays, artificial light design is highly advanced, allowing us to benefit from a wide range of light sources. Two important indicators for artificial light are:

- *Colour rendering*: index measuring the light source capacity to return the real colour of the illuminated object;
- *Temperature*: measured in Kelvin [K], whereas low values will produce yellow-oriented colours, while high values will produce blue-oriented colours.

European standard EN 12665 [84] defines visual comfort as "a subjective condition of visual well-being induced by the visual environment" [26] and depends on:

- the physiology of the human eye,
- the physical quantities describing the amount of light and its distribution in space,
- the spectral emission of the light source.

Appropriate lighting has to be provided so that people can perform visual tasks efficiently and accurately. The level of visibility and comfort will depend on the type of workplace, the activities carried out and their duration (for instance, as specified in EN12464-1). Illuminance levels should be designed to incorporate daylight and electric light or a combination of both. For reasons of comfort and energy, the use of daylight is preferred in most cases. This will depend on factors like standard occupancy hours, autonomy (the portion of occupancy time in which there is enough daylight), location of the building (latitude), the number of daylight hours during the different seasons, etc. [7]

2.5.1 Visual comfort indices

Studies on visual comfort are usually based on assessing a number of specific factors that highlight the dependence between the light environment and the occupants' needs. These factors are:

- The quantity of light;
- The homogeneity of light;
- The light quality in colour rendering;
- The occupants' glare risk prediction.

Even though the above factors may be interdependent, only one of them is usually considered. There are more and more indices and metrics being described in the literature nowadays.

The quantity of light. A correct quantity of light results in good visibility and the proper performance of the occupants' activities. When the light is either too weak or too strong, it can create various problems. Illuminance is the physical quantity necessary to calculate the quantity of light reaching a certain spot over a surface. Illuminance can be either used directly or integrated with other indices where it is one of the source inputs.
The main indices for assessing the quantity of light are: Illuminance, Daylight Factor, Daylight Autonomy, Continuous Daylight Autonomy, Spatial Daylight Autonomy, Useful Daylight Illuminance, Frequency of Visual Comfort and Intensity of Visual Discomfort [26].

The homogeneity of light. Homogeneity refers to the uniform propagation of light over a certain surface area. It avoids any possible optical stress caused by the need for the eye to switch from weakly to strongly lit areas. In this way, the risk of vision problems can be easily reduced. The "illuminance uniformity" is the main index to be used to calculate light distribution.

The light quality in colour rendering. The available literature often says that people prefer having natural light in their living and working areas [85]. This results in a number of benefits for the general well-being of the occupants and has an impact on many aspects such as physiological, perceptive, psychological, and also economic [86] [25].
The main indices for assessing the quality of light are: CIE Colour Rendering Index, (General) Colour Quality Scale, Flattery Index, Colour Preference Index, Colour-Discrimination Index, Feeling of Contrast Index and Colour Rendering Capacity [26].

The occupants' glare risk prediction. Glare refers to light phenomena hindering the sight of the occupants in a luminous framework. This is caused by an excessive brightness level of either natural or artificial lighting. Glare can be generally defined as "luminance-produced feeling in a visual field, sufficiently higher than the luminance that requires the eyes adaptation; it causes problems, discomfort or loss in terms of visibility and visual performance" [22].

The main indices for assessing glare are: Luminance, Luminance ratio, British Glare Index, Visual Comfort Probability, CIE Glare Index, Discomfort Glare Index, New Discomfort, Glare Index, Unified Glare Rating, Discomfort Glare Probability, the simplification of Discomfort Glare Probability by Wienold et al., the simplification of Discomfort by Hviid et al, Glare Probability, Enhanced simplified Discomfort Glare Probability, Predicted Glare Sensation Vote, J-Index and Comparison of glare sensation scales [26].

This abundant quantity of visual comfort indices is used to assess certain characteristics of luminous environments or the human eye's perception of these environments. Building designers need help to figure out how new buildings should be designed to explicitly optimise the visual comfort for their occupants. Visual comfort factors should therefore be summarised with a multi-objective optimisation approach. Keeping this in mind, it is necessary to first detect, then identify, improve or develop reliable metrics [26].

2.6 Indoor Environmental Quality and Global Comfort

In this section, all reviewed articles fulfilling the review requirements are shown. This part represents the review core and it has been divided into two parts, in order to achieve the pre-established targets. The first part concerns the studies on "IEQ comfort categories weighting", while the second part selects the studies proposing an explicit formula on a "global comfort index".

2.6.1 IEQ Comfort Categories Weighting

All reviewed studies concerning IEQ comfort categories weighting are summarised in Table 2.4. All "weights" are obtained from different data analysis techniques such as Pearson correlation, Analytic hierarchy process (AHP), Multivariate linear/logistic regression, etc. Table 2.4 includes some annotations, namely:

- In "Marans and Yan, 1989" study [87], heating and drafts coefficients have been merged and considered in the thermal category.
- In some studies, certain categories and/or parameters are not considered for this analysis. For example, the EMF (electromagnetic fields) category (in "Chiang and Lai, 2002" study [34]), and the air velocity parameter (in "Marino et al., 2012" study [41]) have been removed.

Table 2.4: IEQ comfort categories studies and corresponding weighting.

Study(ies)	**Thermal**	**IAQ**	**Acoustic**	**Visual**
Marans and Yan, 1989 [87] (enclosed office)	0.74 (0.50+0.24)	0.6	0.48	0.51
Reffat and Harkness, 2001 [32] [33]	2.29	3.38	1.89	3.44
Chiang and Lai, 2002 [34]	0.208	0.290	0.203	0.164
Mui and Chan, 2005 [88]	0.42	0.09	0.28	discarded
Humphreys, 2005 [35]	0.67 (0.39+0.16+0.12)	0.36	0.13	0.05
Lai and Yik, 2007 [89] (commercial buildings - end users)	0.1127	0.6531 (0.2318+0.4213)	0.2341	-
Lai and Yik, 2007 [89] (commercial buildings - professionals)	0.2015	0.4233 (0.131+0.2923)	0.3752	-
Wong et al., 2008 [36]	6.09	4.88	4.74	3.70
Astolfi and Pellerey, 2008 [37] (renovated classroom)	0.50	0.32	0.39	0.29
Astolfi and Pellerey, 2008 [37] (non-renovated classrooms)	0.28	0.31	0.50	0.25
Choi et al., 2009 [38]	0.51	0.52	0.43	0.45
Lai et al., 2009 [39]	22.05	1.609	21.86	11.77
Lai and Yik, 2009 [90] (high-rise residential buildings)	0.3382	0.4313 (0.229+0.2023)	0.2305	-
Bluyssen et al., 2011 [40] (in summer)	0.577 (IAQ1)	0.510 (IAQ2)	0.482	0.450
Bluyssen et al., 2011 [40] (in winter)	0.529	0.408	0.491	0.441
Marino et al., 2012 [41] (in summer)	0.173	0.150	0.160	0.146
Marino et al., 2012 [41] (in winter)	0.189	0.150	0.160	0.146
Cao et al., 2012 [42]	0.316	0.118	0.224	0.171
Ncube and Riffat, 2012 [43]	0.30	0.36	0.18	0.16
ASHRAE/CIBSE/USGBC Performance Measurement Protocols (PMP), Kim and Haberl, 2012 [44] Heinzerling et al., 2013 [91] Hunn and Bochat, 2015 [92]	0.12	0.20	0.39	0.29
Fassio et al., 2014 [45] (11:30 am - Linear Regression)	0.33	0.10	0.18	0.38
Fassio et al., 2014 [45] (11:30 am - Logistic Regression0)	0.30	0.12	0.28	0.30
Loreti et al., 2015 [46]	0.21	0.19	0.30	0.17
Piasecki et al., 2017 [93] [94]	0.25	0.25	0.25	0.25
Buratti et al., 2018 [95]	34.5	-	35.7	30.1
Wei et al., 2020 [47] (Average of Green Building certification schemes)	0.27	0.34	0.17	0.22

- In "Mui and Chan, 2005" [88], the coefficient for the visual category was found to be negative. Therefore, PDVC (Percentage of dissatisfaction in visual comfort) was removed from the model by the authors.
- In "Humphreys, 2005" study [35], warmth, air movement, and humidity coefficients have been merged and considered in the thermal category.
- In "Lai and Yik" studies [89] [90], air cleanliness and odour coefficients have been merged and considered in the IAQ category.
- In "Bluyssen et al., 2011" [40], IAQ1 category has been considered in thermal category (as it contains temperature, air movement and humid/dry air quality). IAQ2 category has been considered in IAQ category (as it contains stuffy/fresh air quality).
- In "Wei et al., 2020" [47], the average of the following Geen Building schemes have been considered: BREEAM, KLIMA, DGNB, ITACA, LiderA, LEED, and NABERS. and odourless air quality).

Figure 2.4 shows the four IEQ category importance degrees for each study. A higher number corresponds to a higher ranking i.e. rank 1 = lowest importance, rank 4 = highest importance.

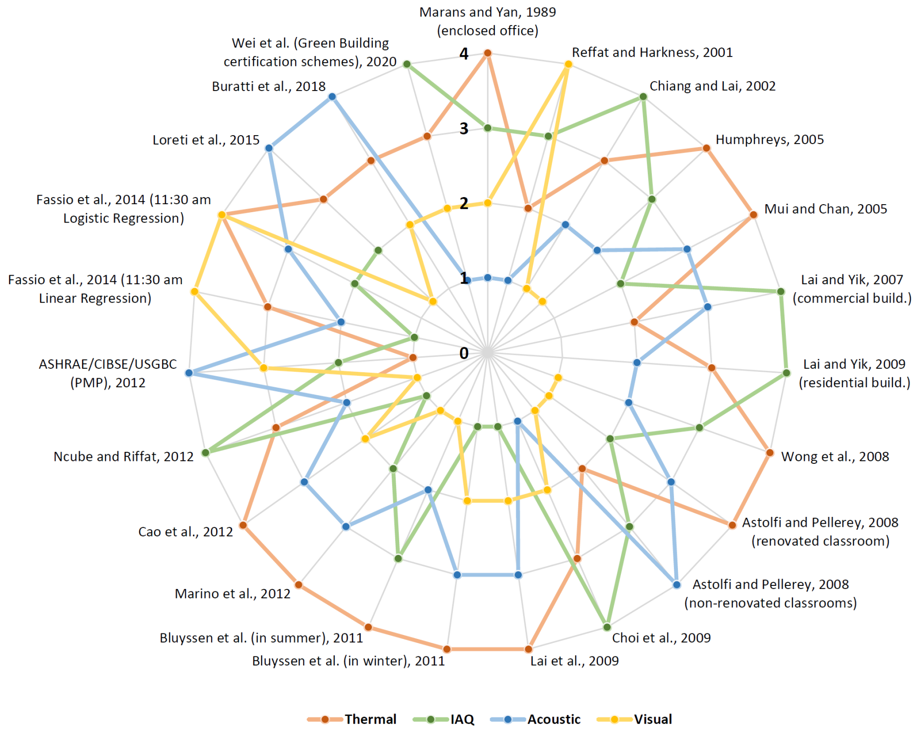


Figure 2.4: Summary radar chart of IEQ category ranking studies.

2.6.2 Global Comfort Indices

This section concerns some of the main comfort indices based on indoor environmental quality, and already covered in the available literature. These indices are global and, as such, are often made of sub-indices, relative to the different comfort categories and/or physical quantities. The studies selected for this revision include indices with an explicit final formula and that may allow evaluation of the assigned weight for each comfort category or physical quantity. The reviewed literature mainly deals with the following fields: air, environment, indoor and building quality. The selected studies cover a temporal range of the last 20 years. Older studies are often focused on thermal comfort only; moreover, they rarely jointly cover all the different comfort aspects for indoor contexts. The selected indices are included in Table 2.5, as follows:

Table 2.5: IEQ indices, corresponding year, author(s), and publication journal.

Index	Author(s)	Year	Published in
TEQE [32] [33]	Reffat and Harkness	2001	Journal of Performance of Constructed Facilities
$IEI_{(AHP)}$ [34]	Chiang and Lai	2002	Building and Environment
IEI [96]	Moschandreas and Sofuoglu	2004	Journal of the Air and Waste Management Association
I [35]	Humphreys	2005	Building Research and Information
PDIEQ [88]	Mui and Chan	2005	Architectural Science Review
S [42]	Cao et al.	2012	Building and Environment
IEQ_{index} [43]	Ncube and Riffat	2012	Building and Environment
DEQI [97]	Laskari et al.	2017	Indoor and Built Environment
I_{CC} [95]	Buratti et al.	2018	Building and Environment

For each of these IEQ indices, the following details will be presented:

- Comfort elements and/or categories taken into consideration;
- Formula to calculate the global index;
- Any data, methods, techniques and/or algorithms that have been used.

Total environmental quality evaluation (TEQE)

TEQE [32] [33] considers visual (lighting), acoustic, thermal comfort and indoor air quality (IAQ) in office buildings. It is obtained through the following formula:

$$TEQE = 3.44 \cdot lighting + 1.89 \cdot acoustics + 2.29 \cdot thermal + 2.38 \cdot IAQ \quad (2.4)$$

where assigned weights were calculated from 50 expert inputs.

Finally, statistical analyses were performed using SPSS (Statistical Package for Social Sciences).

The Indoor Environmental Index using analytic hierarchy process ($\mathbf{IEI_{(AHP)}}$)

$IEI_{(AHP)}$ [34] takes into account thermal comfort ($S_{ThermalComfort}$), acoustic comfort ($S_{Acoustics}$), visual comfort ($S_{Illumination}$), indoor air quality (S_{IAQ}) and electromagnetic fields (S_{EMF}). It is obtained through the following formula:

$$\begin{aligned} IEI_{(AHP)} &= 0.203 \cdot S_{Acoustics} + 0.164 \cdot S_{Illumination} + \\ &+ 0.208 \cdot S_{ThermalComfort} + 0.290 \cdot S_{IAQ} + 0.135 \cdot S_{EMF} \end{aligned} \quad (2.5)$$

The analytic hierarchy process (AHP) method is carried out to do the weighting.

The Indoor Environmental Index (IEI)

IEI [96] considers indoor air quality aspect (with IAPI - Indoor Air Pollution Index) and thermal comfort aspect (with IDI - Indoor Discomfort Index) in office buildings.

IAPI is obtained by measuring the concentrations of formaldehyde (HCHO), total volatile organic compounds (TVOC), carbon monoxide (CO), carbon dioxide (CO_2), particulate matter (PM10, PM2.5), bacteria, and fungi, whereas IDI is obtained by measuring levels of temperature and relative humidity.

IEI is the arithmetic mean between IAPI (Indoor Air Pollution Index) and IDI (Indoor Discomfort Index). It is obtained through the following formula:

$$IEI = \frac{IAPI + IDI}{2} \tag{2.6}$$

The used data comes from a study that measured the concentrations of the pollutants and simultaneously identified the symptoms of the office occupants, surveyed through questionnaires, along with the building characteristics, according to a standard protocol.

The index of overall comfort (I)

This index I [35] considers thermal comfort (with satisfaction code for warmth, air movement, humidity - S_w, S_{am}, S_h), acoustic comfort (with satisfaction code for noise – S_n), visual comfort (with satisfaction code for lighting - S_l) and indoor air comfort (with satisfaction code for air quality – S_{aq}). It is obtained through the following formula:

$$I = 1.24 + 0.39 \cdot S_w + 0.16 \cdot S_{am} + 0.12 \cdot S_h + 0.05 \cdot S_l + 0.13 \cdot S_n + 0.36 \cdot S_{aq} \tag{2.7}$$

where this has been achieved by the use of data from an environmental survey and multiple regression analyses.

Percentage of dissatisfaction in indoor environmental quality (PDIEQ)

PDIEQ [88] considers thermal comfort (percentage of dissatisfaction in thermal comfort - PDTC), acoustic comfort (percentage of dissatisfaction in aural comfort - PDAC), visual comfort (percentage of dissatisfaction in visual comfort - PDVC), indoor air quality (percentage of dissatisfaction in indoor air quality - PDIAQ). It is obtained through the following formula:

$$PDIEQ = 0.42 \cdot PDTC + 0.09 \cdot PDIAQ + 0.28 \cdot PDAC \tag{2.8}$$

where PDVC has been removed from the model because the range of illuminance gave no significant contribution. In this research, an IEQ logger was developed to measure the physical parameters. In addition, a questionnaire was given to obtain the subjective responses of the occupants and a multiple regression model was adopted.

Overall satisfaction (S)

This index S [42] considers thermal comfort (satisfaction with the indoor thermal environment - S_T), acoustic comfort (satisfaction with the acoustic environment - S_A), visual comfort (satisfaction with the luminous environment - S_L), indoor air quality (satisfaction with the indoor air quality - S_{IAQ}). It is obtained through the following formula:

$$S = 0.075 + 0.316 \cdot S_T + 0.118 \cdot S_{IAQ} + 0.171 \cdot S_L + 0.224 \cdot S_A \quad (2.9)$$

In this study, the satisfaction of the occupants with the indoor environment was investigated through questionnaires and multivariate linear regression.

The Overall IEQ index (IEQ_{index})

IEQ_{index} [43] considers thermal comfort (with thermal comfort index - TC_{index}), acoustic comfort (with the acoustic comfort index – ACc_{index}), visual comfort (with lighting index – L_{index}), indoor air quality (with indoor air quality index – IAQ_{index}). It is obtained through the following formula:

$$IEQ_{index} = 0.30 \cdot TC_{index} + 0.36 \cdot IAQ_{index} + 0.16 \cdot L_{index} + 0.18 \cdot ACc_{index} \quad (2.10)$$

This study adopted POM (Passive Observational Method), a correlational method involving both field measurements and questionnaires.

Dwelling Environmental Quality Index (DEQI)

DEQI [97] considers thermal comfort (with temperature and relative humidity sub-indices – S_T, S_{RH}) and indoor air comfort (with carbon dioxide sub-index – S_{CO2}). DEQI is the simple arithmetic mean of the three sub-indices s for temperature, RH (Relative Humidity) and CO_2 concentrations as defined by the equation:

$$DEQI = \frac{S_T + S_{RH} + S_{CO2}}{3} \quad (2.11)$$

Sub-indices are calculated by the following equation (based on the formula developed by Marino et al. [41]:

$$S_i = 100 \cdot f_{i,I} + 70 \cdot f_{i,II} + 35 \cdot f_{i,III} \quad (2.12)$$

where $f_{i,N}$ is an indicator measuring the persistence of the indoor environmental conditions that satisfy the requisites defining the N-th category of quality (please, refer to Table 2.3).

The recommendations for energy and design calculations in the European Standard EN15251:2007 were used to give the ranges of values for the various categories [66]. Homes used as part of the ICE-WISH project were employed to provide the indoor environment data [98].

Combined Comfort Index (I_{CC})

I_{CC} [95] considers thermal comfort (with predicted mean vote index - I_{PMV}), acoustic comfort (with sound index – I_S) and visual comfort (with visual comfort index – I_{VC}). It is obtained through the following formula:

$$I_{CC} = 0.35 \cdot I_{PMV} + 0.35 \cdot I_S + 0.3 \cdot I_{VC} \tag{2.13}$$

In this research, indices are mainly based on measurements, whereas index-weights are based on a questionnaire.

Other IEQ studies

These indices concern studies in which there is an explicit final global index formula covering various comfort categories. However, other relevant research on different case studies concerning IEQ factors and/or their relationships were found for this review. For the sake of completeness, these pertinent studies are mentioned below: Chiang et al. (2001) [99], Frontczak and Wargocki (2011) [100], Kim and de Dear (2012) [101], Catalina and Iordache [102], Sakhare and Ralegaonkar [103], Nimlyat and Kandar (2015) [104], Gadotti and Albatici (2016) [105], Ricciardi and Buratti (2018) [49], Nimlyat (2018) [106], Yang and Moon (2019) [107], Piasecki (2019) [108], Rohde et al. (2020) [109], Piasecki et al. (2020) [110], Tang et al. (2020) [111].

2.7 Discussion on Global Comfort Indices

Before reviewing, the basics and main indices of each comfort category were presented. For each of these IEQ elements, the main indices (or indicators) most commonly used in Green Buildings rating systems are listed below, in spreading order [105].

- Indicators for thermal comfort: operative temperature, humidity, PMV / PPD, thermal control, air velocity, room temperature, temperature differences between walls, room thermal capacity, Givoni comfort zone, and sunlight penetration ratio.

- Indicators for indoor air comfort (IAQ): formaldehyde concentration, CO_2 concentration, TVOC, low emitting materials, and air ventilation rate.

- Indicators for acoustic comfort: noise level, sound insulation, and reverberation time.

- Indicators for visual comfort: daylight factor, illuminance, sunlight availability, CRI, view out, lighting control, illuminance ratio, glare control, daylight uniformity, daylight illuminance, Equivalent Melanopic Lux.

These indices refer to a specific IEQ category provided a necessary overall picture to analyse and discuss global comfort indices. In order to discuss the importance of each specific IEQ category, all reviewed studies coefficients are approximate to 2 decimal points, reported on a percentage scale and presented below. Figure 2.5 summarises the proposed weightings in the reviewed research studies. Two studies ([96, 97]) are not in Figure 2.5 (such as in Table 2.4 and Figure 2.4), but they are in Table 2.5 because their corresponding formulas (see formulas 6 and 11) do not perform a "real" weighting but an arithmetic average.

Averaging over the analysed studies, the thermal category is considered as the most important and the visual category as the least important. The IAQ and acoustic categories have on average the same impact on global comfort. In detail, an average of all the analysed studies gives the following percentages:

- 30% (thermal)

- 26% (IAQ)

- 26% (acoustic)

- 23% (visual)

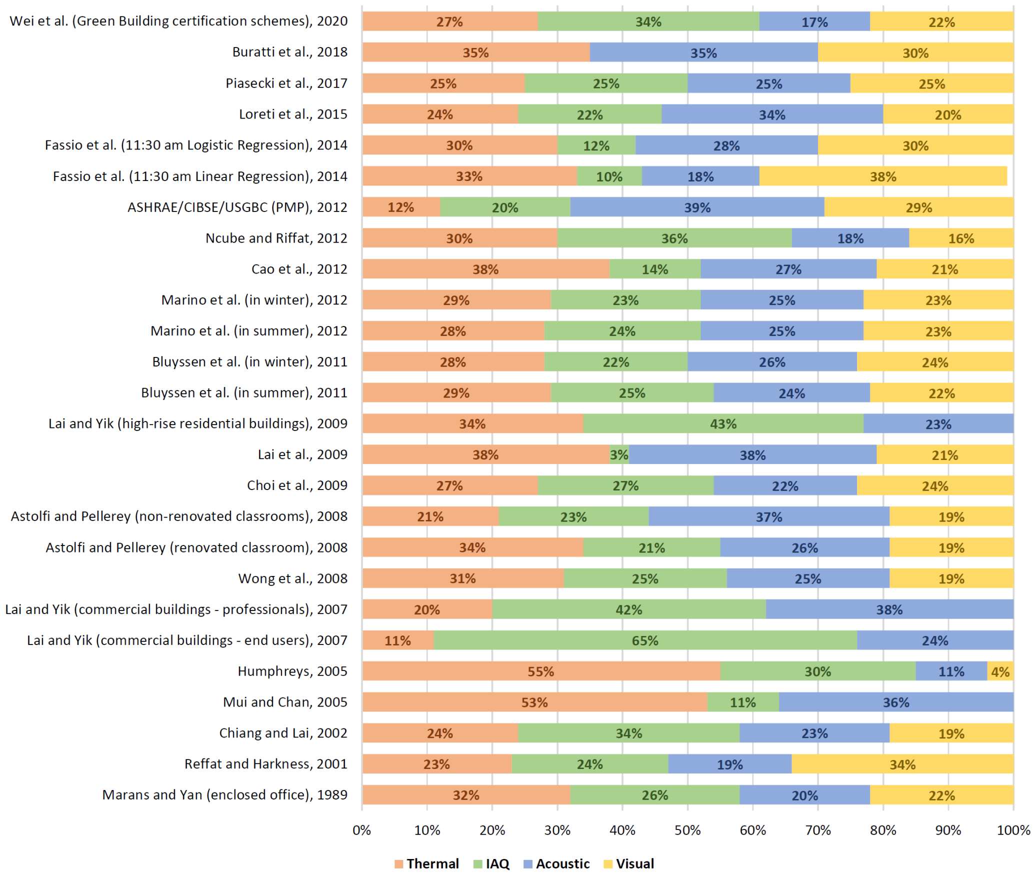


Figure 2.5: Percentage weightings in the reviewed research studies.

In Green Buildings schemes, IAQ category tends to have more weight than acoustic category (sometimes it is considered the most important or at least as important as thermal category). Moreover, each category can considerably vary from case to case. More precisely between the analysed studies, the variation is as follows:

- Thermal category from 11% to 55%.
- IAQ category from 3% to 65%.
- Acoustic category from 11% to 39%.
- Visual category from 4% to 38%.

Generally speaking, it is difficult to establish the IEQ category impact on overall comfort for different reasons:

- The non-independence between variables: physical environmental factors (such as sound level, temperature, illuminance, etc.) do not only influence the corresponding comfort category but (although to a lesser degree) also the other categories [107].
- The IEQ factors weightings change significantly based on the occupant's expectation/satisfaction for that corresponding IEQ factor [101]. Thus, if for instance, occupants are not satisfied with thermal comfort, this category becomes more relevant.
- The IEQ category weighting depends on the type of building type (e.g. commercial building, residential building, educational building, etc.).
- To determine various IEQ categories weightings, different models are adopted with different results.
- In general, the IEQ (like the IEQ-productivity belief [112]) depends not only on the building characteristics. It also depends on attitudinal-behavioural factors, social-influence factors and demographics (such as gender, age and cultural difference).

However, by analysing the reviewed studies, it is possible to define some *common aspects* of these global comfort indices and models:

- All mentioned indoor environmental quality indices consider different types of comfort. Nevertheless, the four most relevant comfort categories are the ones examined in this chapter (indoor air quality, thermal, acoustic and visual comfort);
- In the majority of such global comfort indices, each sub-index (or comfort category) is assigned a "relevancy weight" which are mainly weight-based indices. Several methods are used to assign these weights, but in any case, they are drawn from databases coming from surveys and/or expert input;
- The various indices that comprise global comfort are often assessed in either a completely objective way (through sensors) or a completely subjective way (through surveys), so these two aspects are not often analysed together.

In accordance with the above, an indoor global comfort index (IGCI) gathering the best aspects should have the following *requisites*:

- Considering only those indices that have a certain relevance with respect to the global one (i.e. thermal comfort, indoor air quality, acoustic comfort, visual comfort indices);
- Assigning certain importance to each index (for instance, "weighted" attribution through the acquisition of subjective data, such as questionnaires/feedback by either experts or occupants);
- Correlating subjective data to objective data;
- Tracking the index value variation according to the occupants' preferences and the measurable objective parameters (temperature, humidity, CO_2 levels, illuminance, noise level, etc.);
- Introducing an acceptability threshold for each index. In this way, even if only one of them falls below this threshold, the entire global comfort index must be heavily affected. The laws about this subject change and depend on the country where they are applied and on the use cases. This is why it is appropriate to let these choices be made freely.

This kind of global index has the following *pros and cons.*
Pros - Managing a global comfort index in a building allows:

- The quantification, in the most objective way possible, of the comfort aspects only of a certain building;
- Aspects to be included and/or improvement of a model already proposed for anyone who might study this subject;
- A parameter/objective reference to be obtained for the designer and/or builder, for the certifying authority and for the purchaser;
- Implementation of artificial intelligence algorithms for the optimal control of any actuators (e.g. shutter position in windows, cooling/heating, dimming for lighting level, etc.) based on index value and occupant feedback;
- Improvement of the occupants' quality of life;
- The majority of building types to be covered (e.g. green buildings, smart homes / smart buildings, educational buildings, offices, and other more specific cases).

Cons - This IGCI considers the more important aspects that create comfort, without other elements that might change the indoor environment quality perception. Some examples are:

- Considering attitudinal-behavioural factors, social-influence factors and demographics;
- Considering external factors, such as climate, within the model;
- The presence (or absence) of systems (such as safety / cleaning / gardening / entertainment systems);
- Security and safety in general;
- The use of specific building materials;
- Others aspects: potable water, electromagnetic frequency levels, smart speakers, ergonomics, aesthetics, etc.

On the one hand, integrating such elements could contribute to a more complete indoor global comfort index; on the other, the risk is that more subjectivity might enter the evaluation, with the subsequent decrease of the index objectivity.

By analysing the reviewed papers, different *techniques and models* are used to obtain the weights (or correlation coefficients) of each comfort category. In general, the most commonly adopted models in all these studies covering IEQ factors are [45]:

- Multivariate linear regression algorithm;
- Multivariate logistic regression algorithm;
- Multivariate linear regression algorithm based on dummy variables;
- Alternative algorithms.

Finally, other recent comfort-related studies are using artificial intelligence algorithms. This promising approach focused on them will be discussed below.

2.7.1 Comfort indices: future direction and Artificial Intelligence

Nowadays, most global comfort indices can help quantify the comfort of a certain indoor environment or building in addition to being capable of predicting comfort levels for several reasons. The main one is to be able to automatically control different actuators for improving the building performance (especially in terms of energy efficiency). Artificial intelligence in comfort prediction has been employed in several studies. The main techniques that have been employed are based on machine learning (ML) and concern:

- Artificial Neural Networks (ANN)
- Decision Trees (DT)
- Support Vector Machines (SVM)
- Bayes Networks (BN)
- General Linear Model (GLM)

In several studies, ANN is deployed to implement AI [113] [114] [115] [116] [117]. More specifically, some studies use models such as multilayer perceptron (MLP) [118] [119] and Neural Network Autoregressive with Exogenous Input (NNARX) [120] [121] to output indoor temperature and indoor relative humidity. Other studies, aimed at obtaining the Predicted Mean Vote (PMV) and the Thermal Sensation Vote (TSV) as outputs apply models like the back-propagation neural network (BPNN) [122] [123], the feed-forward neural network (FFNN) [124] [125] [126] [127], the radial basis function networks (RBFN) [128] and random forests (RFs) [129] [130]. However, these studies use artificial intelligence to predict quantities (or indices) often related to thermal comfort and rarely include the different IEQ aspects. Today, these algorithms must include as many comfort factors as possible (not only thermal comfort). The integration of IoT (Internet of Things) and WSN (Wireless Sensor Networks) has led to the widespread use of artificial intelligence algorithms, which normally require large amounts of data for proper processing. From the aforementioned studies, it can be concluded that, in this field, artificial intelligence allows the implementation of increasingly better predictive models. The implementation of methods that exploit AI in the prediction of comfort levels is also essential for energy efficiency in buildings [131]. For instance, this allows the well-known BPG (Building Performance Gap) to be reduced, i.e. the difference between the predicted and actual performance of a building.

The next chapter will describe the hardware and software architecture of the wireless IEQ (Indoor Environmental Quality) logger.

Chapter 3

Indoor Environmental Quality logger

Assessing the impact of each IEQ category on overall comfort is challenging for multiple reasons [52], as can be derived from Chapter 2. First, the physical environmental factors (such as CO_2 concentration, noise level, temperature, and illuminance) influence not only the corresponding comfort category but also the other categories, although to a lesser degree [107]. The IEQ factor weightings largely depend on the occupant's expectation and satisfaction toward the corresponding factor [101]. For example, if occupants are not satisfied with acoustic comfort, this category becomes more relevant. The IEQ category weightings also depend on building type (e.g. commercial, residential, and educational buildings) and other building-related factors (e.g. geographic location, ventilation system, public or private property, new or existing) [52], as well as seasonal changes and external climate [100]. Finally, IEQ (like the IEQ-productivity belief) is also affected by attitudinal and behavioural factors, social influence factors, and demographic aspects of the building occupants (such as gender, age and cultural difference) [112]. Different methods have been adopted to determine these weightings and have produced different results [52].

The purpose of this project is precisely to identify a methodology for predicting perceived comfort from measured physical parameters in a given indoor environment. A wireless IEQ logger system was designed to this aim, comprising hardware, software components, and data analysis. The idea is to assemble a hardware system that is expandable and has the necessary resources for autonomous data processing. For this reason, a microprocessor-based embedded system (Raspberry Pi) was chosen, rather than a micro-controller-based one (such as Arduino). The design choice of employing

Raspberry Pi allows other sensors to be easily connected, communicating via wireless, processing, sending, and receiving data in real-time. This choice allows the system to be easily scalable both for its current purpose and future. The final idea is to develop a device that provides a predicted global comfort index in real-time. This device will be called a "virtual sensor" (explained in Chapter 5). For the sake of simplicity, practicality and compactness, some environmental kits were considered, rather than individual sensors to be connected directly to the board. These kits are Metriful [132], OKdo air quality kit [133], and Enviro+ [134]. Metriful uses the MS430 all-in-one sensor. This is a very cheap sensor (it costs about 40 €) but it is currently out of stock on the market. The OKdo air quality kit adopts a "Base HAT" to connect the Asong AM2302 temperature and humidity sensor and the Sensirion SGP30 sensor to measure VOCs (volatile organic compounds) and eCO_2 (carbon dioxide equivalent). This complete kit costs about 50 €. Enviro+ (by Pimoroni) is the final choice, as it is currently available on the market (for about 50 €) and it is definitely one of the most complete models (see later for details).

3.1 Project architecture

The process of IEQ data collection and processing is divided into multiple steps.

- *Implementation*: the IEQ logger was built adopting the DIY philosophy (in Section 3.2). The main hardware components are an IEQ Control Unit and sensors measuring physical quantities associated with indoor environmental quality (i.e. thermal comfort, indoor air quality (IAQ), visual comfort, and acoustic comfort). The software system includes the sensors libraries and control unit, a database for data collection, the online questionnaires, and a graphical web interface.

- *Deployment*: the IEQ logger was positioned in a university classroom and registered 29 university lectures over the course of 3 months (in Section 3.3).

- *Data collection*: physical parameters measured by the sensors (objective data) and questionnaires filled by students (subjective data) were collected and stored in a MySQL database (in Section 3.4).

The data analysis, model building, and characterization phases will be described in the next chapters.

Table 3.1: Comfort categories and corresponding physical parameters and units.

Comfort category	**Physical parameter**	**Unit**
Thermal Comfort	Air Temperature	°C
	Relative Humidity	%
Indoor Air Quality (IAQ)	CO_2 concentration	ppm
Visual Comfort	Illuminance	lx
Acoustic Comfort	Noise level	dBA

3.2 Implementation

This section describes the hardware needed to build the complete IEQ logger system and the software architecture for proper data acquisition and storage provided by the sensors. Thermal comfort is measured with air temperature (in degrees Celsius) and relative humidity (as a percentage). IAQ is measured with CO_2 concentration (in parts per million). Visual comfort is measured with illuminance (in lux). Acoustic comfort is measured with noise level (in A-weighted decibels). Table 3.1 summarises the comfort categories with all corresponding measured physical parameters and units. The human ear is most sensitive to sound at frequencies between 1kHz and 4 kHz [135]. It reaches its maximum sensitivity in the 800 Hz to 2000 Hz frequency range, and it also strongly attenuates sounds below 400 Hz. Please note that the noise level is measured in dBA to take into account this human ear sensitivity.

The type of thermal comfort assessment depends on the adopted approach. The first approach consists in determining the PMV (Predicted Mean Vote) and PPD (Predicted Percentage Dissatisfied) indices according to the ISO 7730 standard [10] that defines them. The determination of the PMV and PPD indices is carried out through specific professional instrumentation, such as microclimatic control units based on "spot measurements". The instrumentation must conform to the requirements specified in the ISO 7726 standard [136]. The advantage of this methodology is the high measurement accuracy. The second approach is to determine the behaviour of parameters such as temperatures, relative humidity, and air velocity through a datalogger and "frequent measurements". By simulating different scenarios with the CBE Thermal Comfort Tool [137], it was possible to carry out several tests concerning the ASHRAE-55 [13] and EN-16798 [8] standards (both with "PMV method" and "adaptive method"). Given the few differences

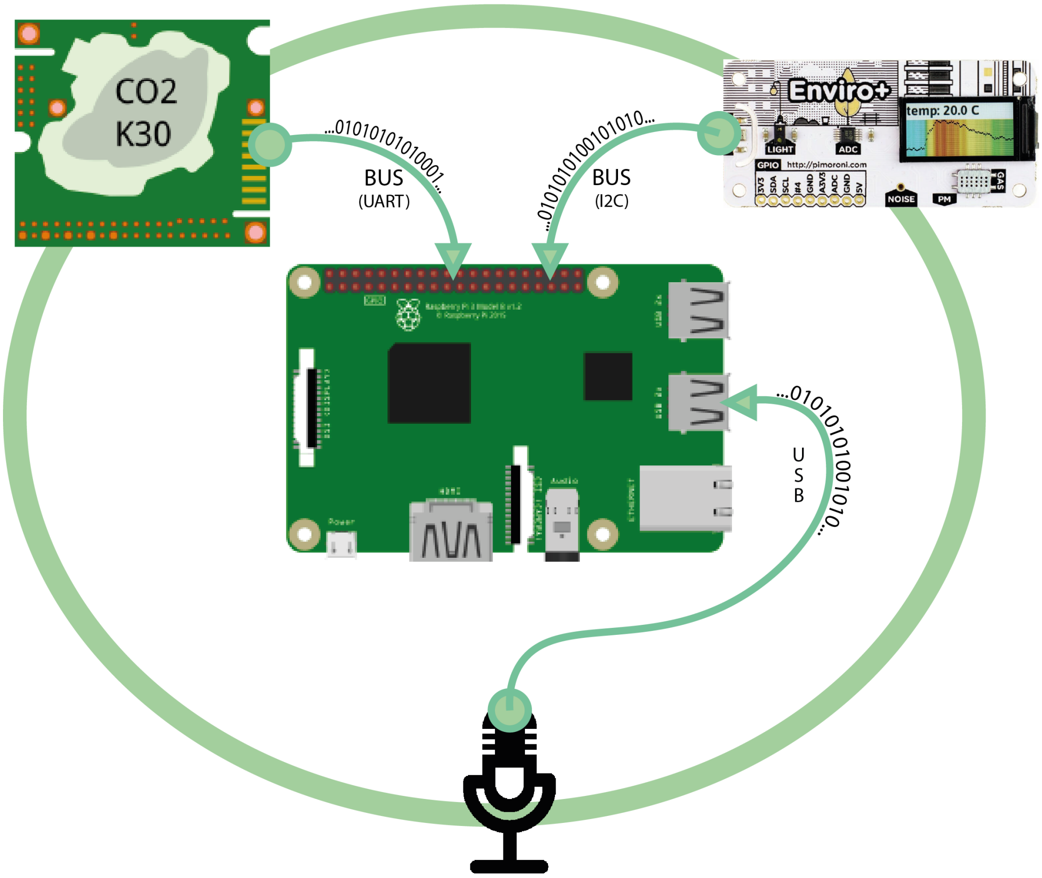


Figure 3.1: Wireless IEQ logger: hardware architecture.

(regarding this case study), it was decided to follow the second approach and directly investigate the air temperature and relative humidity. In summary, the adopted methodology for thermal comfort produces less precise measurements but is undoubtedly cheaper, simpler, more compact, and better in terms of interfacing.

3.2.1 Hardware

Raspberry Pi 3 Model B+ [138], Enviro+ by Pimoroni [134], K30 (CO_2 sensor) [139], and USB omnidirectional condenser microphone were adopted as hardware development of the IEQ logger. The hardware architecture is shown in Figure 3.1.

In addition, other hardware has been adopted, such as a "40-Pin cable" for connection between Raspberry Pi and Enviro+, "GPIO Pin header" to split the necessary wires for the K-30 sensor connection, and external box (ABS case). A common micro-USB power supply with an output voltage of 5V and a maximum current of 3A was employed. Other hardware was required exclusively for sensors calibration and will be described later. Figure 3.2 shows the sensors adopted by the system and their connections.

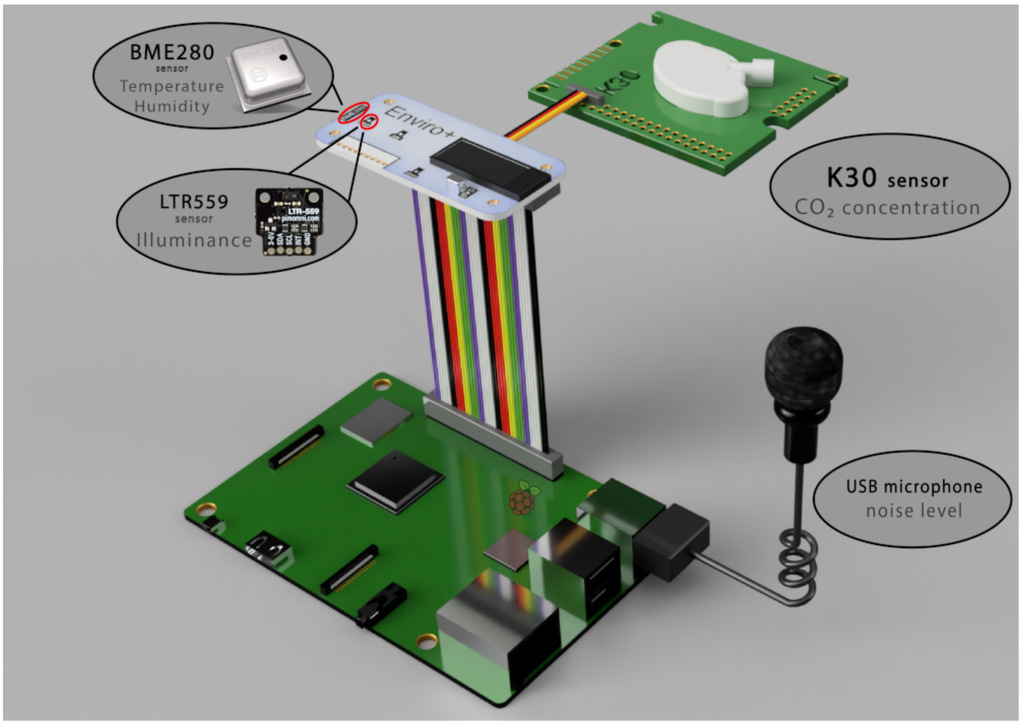


Figure 3.2: Sensors adopted by the system.

Table 3.2 summarises the sensors used to monitor the considered physical parameters, while Table 3.3 presents the sensors technical features.

Table 3.2: Physical parameters and corresponding sensors.

Physical parameter	**Sensor**
Air Temperature	BME280 sensor on Enviro+ board
Relative Humidity	BME280 sensor on Enviro+ board
Illuminance	LTR-559 sensor on Enviro+ board
CO_2	K-30 sensor
Noise level	USB omnidirectional condenser microphone

For more technical features and details, please refer to the corresponding data sheet for BME280 [140], LTR-559 [141], and K-30 [139]. The Enviro+

Table 3.3: Sensors technical features.

Technical Features	BME280	LTR-559	K-30	Microphone
Interface	I^2C (up to 3.4 MHz) SPI (up to 10 MHz)	I^2C (Fast Mode @ 400kbit/s)	I^2C UART	USB 2.0
Power supply	1.71-3.6 V	2.4-3.6 V	5-9 V (preferred operating range)	5 V
Operating range	-40...+85°C (temperature) 0...100% (rel. humidity)	0.01-64k lux (6 dynamic range)	0-10,000 ppm (total) 0-5,000 ppm (within specifications)	84dB (SNR)
Accuracy	±1.0°C (temperature) ±3% (rel. humidity)	-	±30ppm ±3% (of measured value within specifications)	Sensitivity range: within -3dB (at 1V)
Resolution	0.01°C (temperature) 0.008% (rel. humidity)	16-bit (effective resolution)	10mV (8.5 bits in the range 0-4 V)	-
Measurement / Response Time	Response Time ($\tau_{63\%}$): 1 s	Integration time: 50 ms Measurement time: 100 ms	Response Time ($T_{1/e}$): 20 s (diffusion time) Response Rate: 2 s	Frequency Response: 20Hz-16KHz
Dimensions	2.5 × 2.5 × 0.93 mm	2.4 × 3.9 × 1.3 mm	$\sim 57 \times 51 \times 14$ mm	$\sim 20 \times 5 \times 5$ mm
Other specifications	3 power modes: sleep, normal, forced	- Close to human eye spectral response - Immunity to IR / UV Light Source - Automatically rejects 50/60Hz lightings flicker	- Self-Diagnostics (complete function check at startup) - ABC (Automatic Background Calibration)	- Polar Pattern: Omni-directional - Impedance ≤2.2KΩ - Sensitivity: -30dB±3dB

board includes the following sensors: BME280 (temperature, pressure, humidity sensor), LTR-559 (light and proximity sensor), MICS6814 (analogue gas sensor), and SPH0645LM4H-B (MEMS microphone). The board also contains an ADS1015 analogue to digital converter (ADC), 0.96" colour LCD (16x8 mm), and a connector for particulate matter (PM5003) sensor. Finally, other features are a power supply of 5V, a 40-pin header Raspberry Pi models compatible (uses 16 GPIO pins), a communication interface I^2C, and dimensions of 65x30x8.5 mm. For more details, please refer to the board official website [134] and pinout [142]. The Enviro+ board by Pimoroni was mainly used to detect air temperature, relative humidity and light level (thanks to the BME280 and LTR-559 sensors). Currently, there is no full support for MEMS (Micro-Electro-Mechanical Systems) microphone, as the official Pimoroni website reports [143]). Furthermore, running several tests with the currently available libraries, the noise detection range is reduced to a few metres and therefore not very suitable for our purpose. For these reasons, a USB omnidirectional condenser microphone (by Gyvazla brand) was chosen to detect ambient noise. This is a low-cost microphone (it costs just 10 €) with good features for our study. The MICS6814 analog gas sensor [144] detects many different types of gas such as carbon monoxide CO, nitrogen dioxide NO_2, ethanol C_2H_5OH, hydrogen H_2, ammonia NH_3, methane CH_4, propane C_3H_8, and iso-butane C_4H_{10}. However, this sensor does not detect carbon dioxide CO_2. For this purpose, the K-30 sensor has been added to the system. This sensor measures real (not equivalent) CO_2. It is a mid-to-high-end sensor with a good price-performance ratio (it costs 60 €).

Figure 3.3 shows the instruments in operation during the calibration and testing phases. For the calibration and testing phases of the different sensors, the following instruments were adopted: Sound Level Meter,VLIKE VL6708-LCD (for USB microphone calibration), Netatmo NWS01-EC (for K30 CO_2sensor calibration), ThermoPro TP53 (for temperature and humidity sensor calibration BME280) and a consumer-grade smartphone with the corresponding app for brightness sensor.

A smartphone app like "Lux Ligh Meter" was used for lux calibration, measured by the LTR-559 sensor, and the "shift" was corrected via software. The sensor was tested with different light types (with a bulb dimmable in light colour and intensity).

The calibration of the temperature (in degrees Celsius) and the humidity (in percent), measured by the BME280 sensor, was performed via software. The sensor was tested in a room with an HVAC (Heating, Ventilation and Air Conditioning) system (in order to obtain different temperature/humidity conditions). The obtained readings were compared with the values shown on the ThermoPro TP53 display.

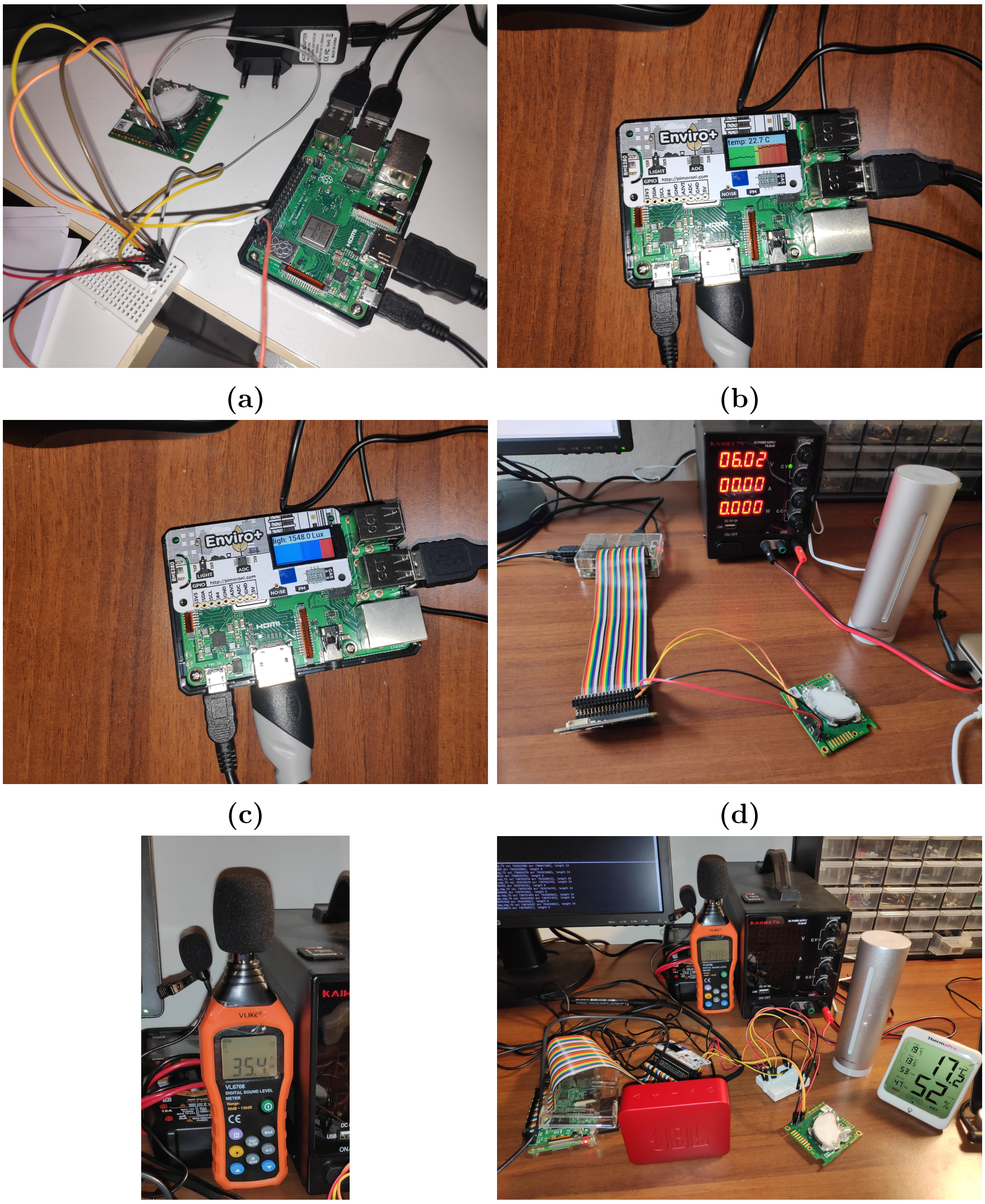
Figure 3.3: Instruments during calibration and testing phases.

Calibration of the dBA measured by the USB omnidirectional condenser microphone was performed via software. The microphone was placed close to the sound level meter. A sound generator (at different frequencies) was used to obtain different noise levels to compare with the values of the VLIKE VL6708 sound level meter and displayed on the LCD.

Lastly, the calibration of the CO_2 concentration was done via hardware. The sensor was placed in an outdoor environment (in fresh air corresponding to 400 ppm), and Din1 was connected to the ground for at least 8 seconds (as instructed in the datasheet [139]). In this way, the internal calibration code bCAL (background calibration) is executed. Then, simply by spending some time in a room, it was possible to compare the values between the K-30 sensor and Netatmo NWS01-EC.

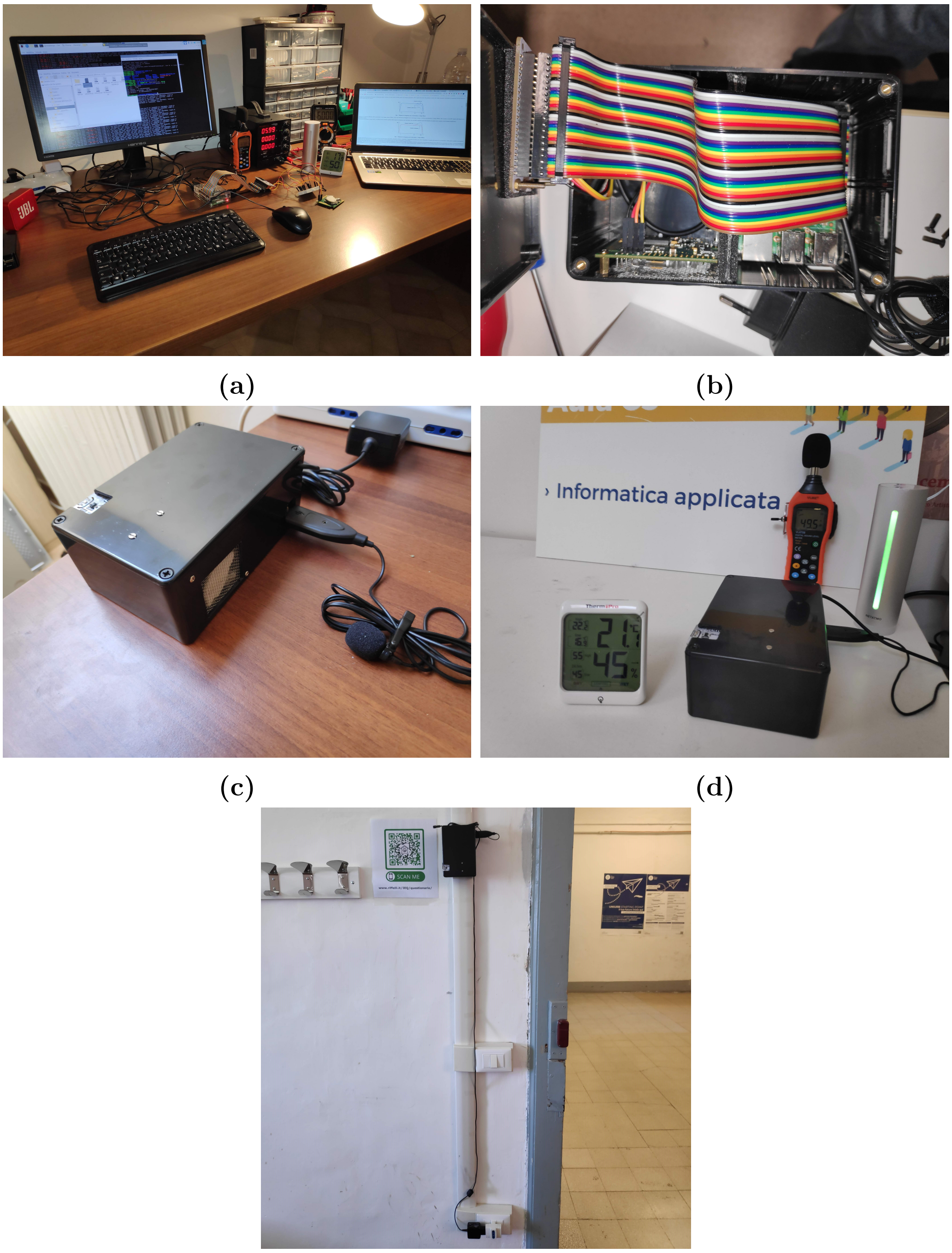
All sensors were tested in a values range suitable for an indoor environment under non-extreme conditions. For technical details, specifications and more information on these devices, please visit the corresponding web pages for VLIKE VL6708-LCD [145], Netatmo NWS01-EC [146], and ThermoPro TP53 [147].

In the following, different stages of the wireless IEQ logger hardware implementation are shown by an image gallery. In Figure 3.4, Part 1 of hardware implementation phases is presented: first tests with the K30 sensor (3.4a), first tests with the BME280 sensor on Enviro+ board (3.4b), first tests with the LTR559 sensor on Enviro+ board (3.4c), K30 sensor calibration through Netatmo NWS01-EC (3.4d), omnidirectional condenser microphone calibration through VLIKE VL6708 sound level meter (3.4e), and BME280 sensor calibration through ThermoPro TP53 (3.4f). In Figure 3.5, Part 2 of hardware implementation phases is presented: working space during the final calibration phase (3.5a), component installation inside the ABS case (3.5b), assembled external box (3.5c), testing period before final installation (3.5d), and wireless IEQ logger classroom installation (3.5e).



(a) (b) (c) (d) (e) (f)

Figure 3.4: Hardware implementation phases: Part 1



(a) (b) (c) (d) (e)

Figure 3.5: Hardware implementation phases: Part 2

3.2.2 Software

In structural terms, the software implemented for the IEQ logger system can be divided into three macro-blocks: i) Sensor Libraries, ii) Software Core, iii) API Service and Database. The software architecture is shown in Figure 3.6.

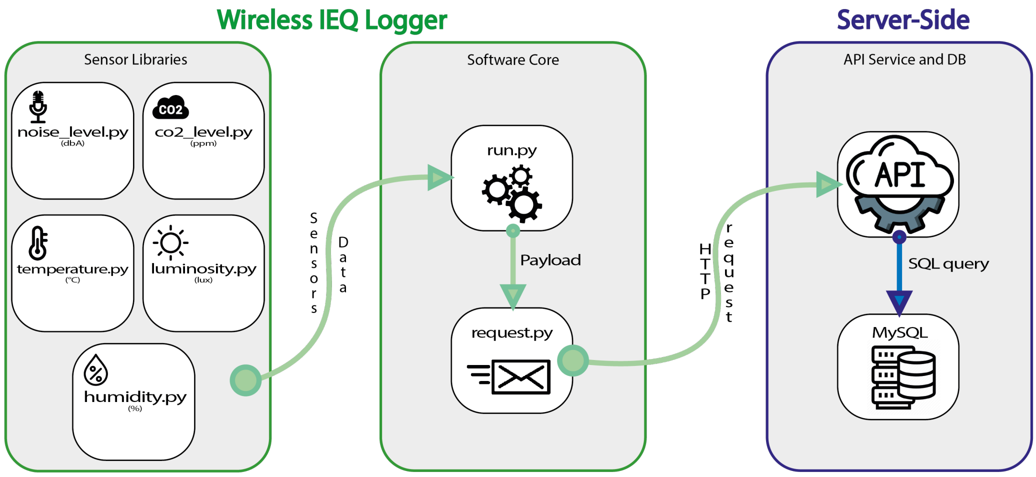


Figure 3.6: Wireless IEQ logger: software architecture.

The *Sensor Libraries* contains all the adopted libraries and implemented in Python programming language. Each library defines the methodologies for measurements from the corresponding sensors as shown in Table 3.4:

Table 3.4: Python files and corresponding sensors.

Python library file	Sensor
temperature.py	BME280 sensor on Enviro+ board
humidity.py	BME280 sensor on Enviro+ board
luminosity.py	LTR-559 sensor on Enviro+ board
co2_level.py	K-30 sensor
noise_level.py	USB omnidirectional condenser microphone

The *Software Core* represents the central part of the system. Inside it, run.py is the main Python script that calls up the previous sensors libraries. The purpose of run.py is to get the reliable value of each sensor from the various libraries and generate a "payload" (in JSON format). In addition to the sensor parameters, username and password are added at the payload beginning to perform operations on the online API (application programming interface). For security reasons, authentication is server-side, and it has been implemented in PHP scripting language. Furthermore, the run.py file uses the methods contained in the request.py file. This last file has the only aim of getting the "payload" as input and sending an HTTP request to the API server located on the Website (which will be discussed later)

The *Server-Side* includes the database and the required API services for interfacing. Each module provides a different service, such as generating a new record, getting one or more records from the database, and so on. The database is implemented in MySQL and mainly consists of records from questionnaires and measurements of all sensors stored in two different tables.

A website has been developed to implement the questionnaire and let the link be also reached via QR code (quick access from smartphones and tablets), collect questionnaires information into the database, and report the acquired measures in a user-friendly layout. The main components of the website are summarised in Figure 3.7.

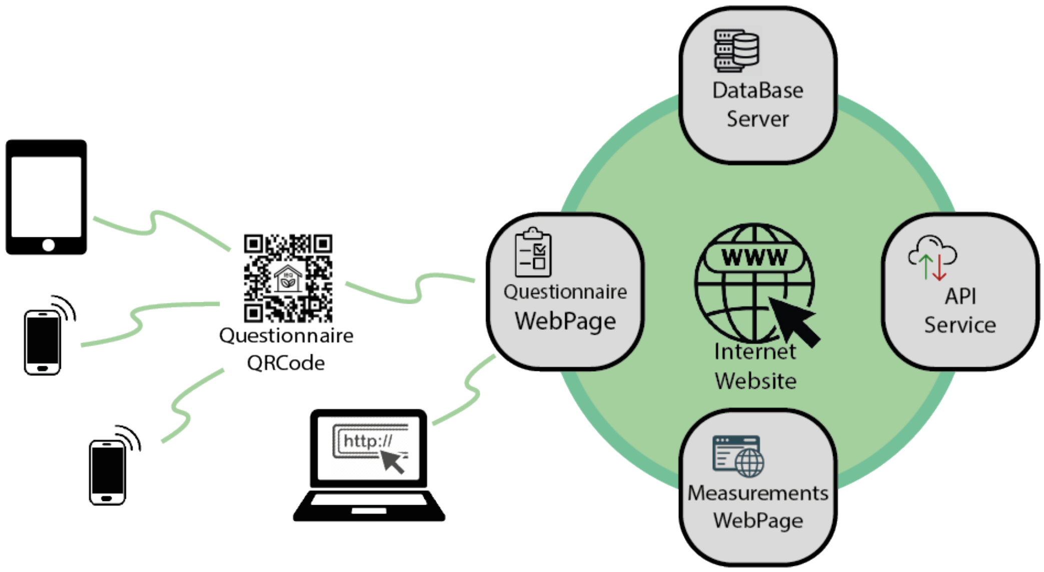


Figure 3.7: Website components.

A measurements webpage layout example is presented in Figure 3.8, while the questionnaire structure is described in Section 3.4.

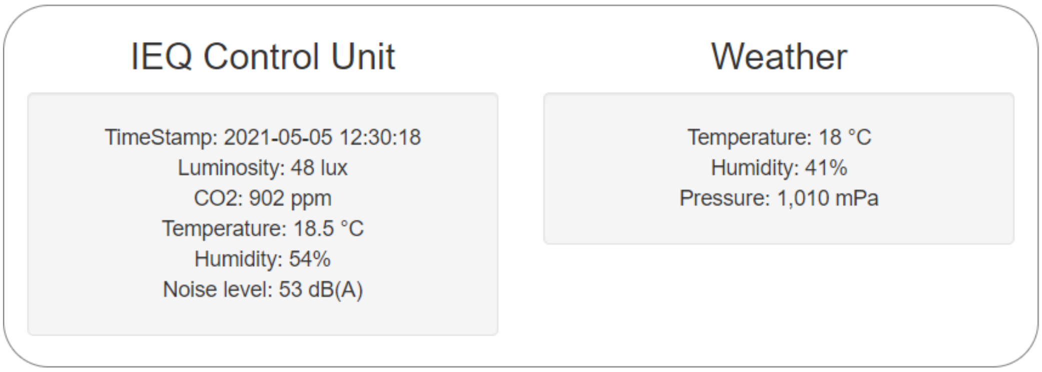


Figure 3.8: A measurements webpage layout example.

To simultaneously provide information on the outdoor conditions, a Weather section has been added (thanks to the openweathermap API service [148]).

Please refer to the corresponding Appendix B and Appendix C for further information on the files of both the IEQ logger and the website components.

3.3 Deployment

Wireless IEQ logger was installed in a classroom of the university, located on the second and top floor of "Collegio Raffaello" building. This classroom, normally occupied by students, has an area of about 70 m^2 (8.2 m × 8.5 m). The system was installed at the height of 1.6 metres from the floor and approximately halfway up one side of the classroom. This height was considered a reasonable average to measure also the CO_2 (which stratifies downwards), the brightness (considering blackboard, windows and eye-level) and the noise perceived by the students. The IEQ logger placement on an internal wall was chosen for practical reasons and to find a position that does not create an obstacle for people (both for the passage and view). According to the tests carried out before and during the calibration phase, this installation on the internal wall still guarantees a good temperature measurement. The chosen position met the following requirements:

- It was sufficiently far from radiators or windows, allowing a correct temperature and humidity measurement;
- It was at a medium height, in order to correctly measure the CO_2 concentration (corresponding approximately to the height of the air inhaled by people);
- It was in the middle of the side because it is optimal for the perceived noise level (not too close to the teacher's voice) and also to detect both

the artificial light (from neon) and natural light (from the windows at the bottom of the classroom);

- It was not too far from the wireless repeater (to ensure a good wireless signal);

Eventually, the position was also comfortable, being close to a socket. A picture of the classroom is shown in Figure 3.9 in order to provide a better idea of the investigated environment. The wireless IEQ logger installation in the classroom is shown in Figure 3.10.Finally, Figure 3.11 illustrates the university classroom plan with the wireless IEQ logger and wireless repeater locations.

Figure 3.9: A picture of the examined university classroom.

Figure 3.10: Wireless IEQ logger: system installation in the classroom.

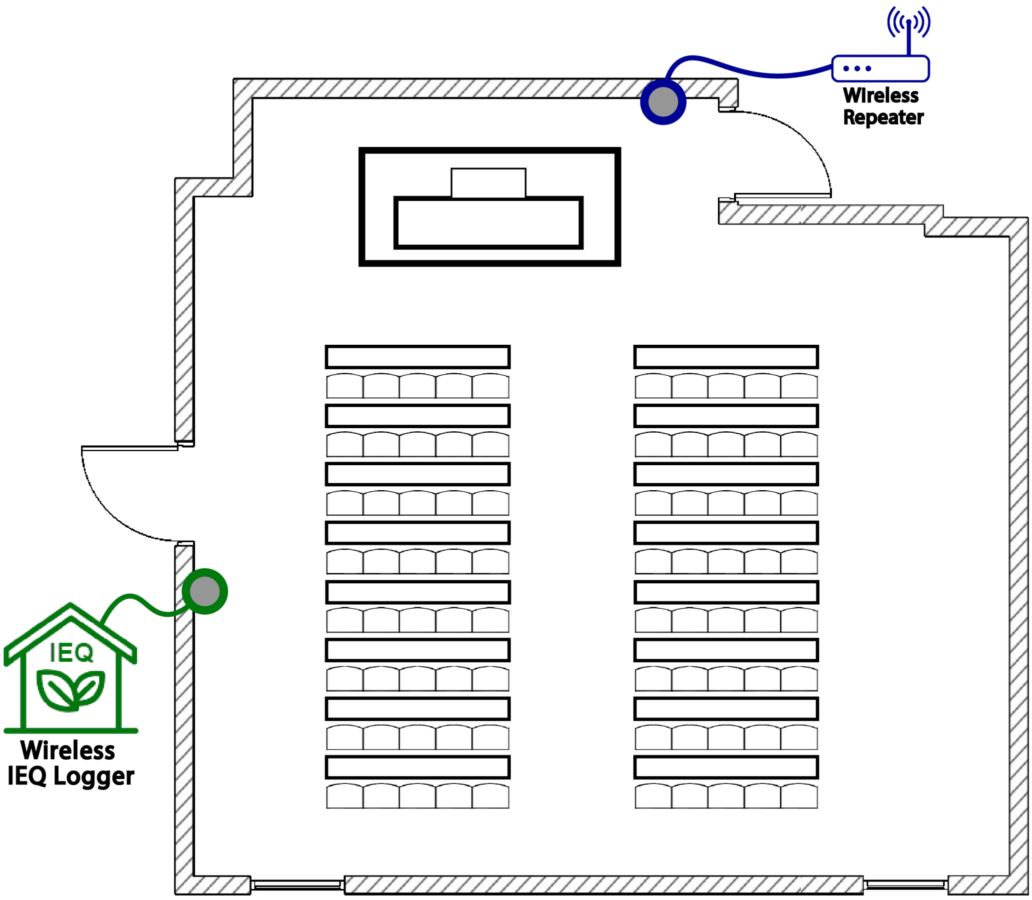


Figure 3.11: IEQ placement on the university classroom plan.

3.4 Data collection

The study was carried out during the three months of March, April and May 2021. More precisely, 85 complete questionnaires were collected from 3 March to 28 May and grouped into 29 sessions. A session is defined as a classroom lesson unit typically one hour long. 10% of the sessions, with either a few or only one questionnaire carried out improperly, were not included in the analysis described in the following section. Environmental parameter data provided by the sensors were recorded every 5 minutes. This time interval is adequate in order to avoid a data overload on the database, and it is however adjustable. In this way, within one hour sessions, there are 12 different recordings for each measured parameter. Figure 3.12 shows the parameters measured in a typical session. It is evident how each factor is affected by the occupants.

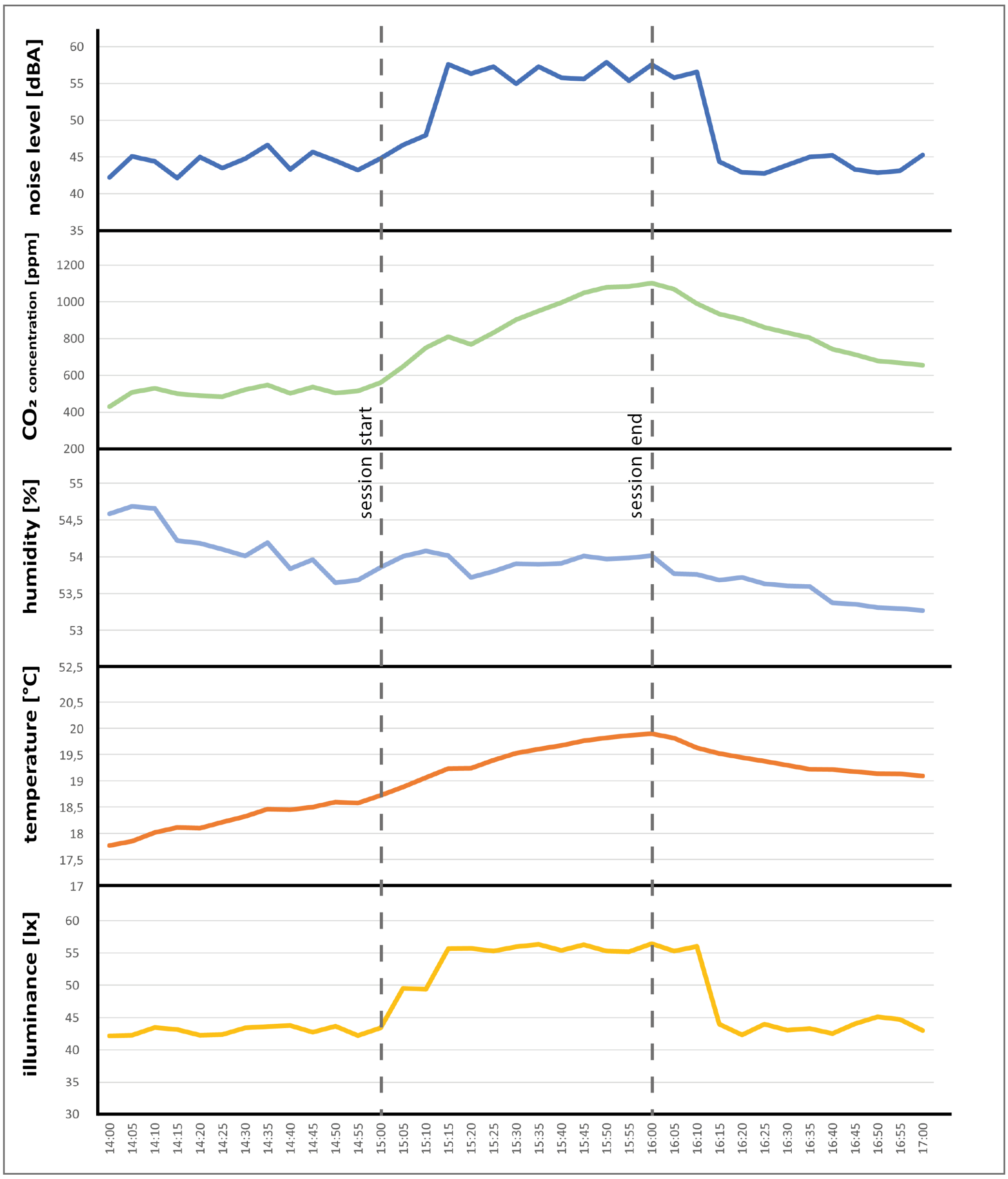


Figure 3.12: Measured physical parameters in a session example.

Sensor data are directly uploaded to the online database via a wireless connection. In case of connection problems, the data are locally stored (in a file on the microSD) and uploaded to the database as soon as the Internet connection is back. On the other hand, subjective data are collected by accessing the following online questionnaire. Graphical interface and sections of the questionnaire are shown in Figures 3.13 and 3.14.

IEQ QUESTIONNAIRE.
BASIC INFORMATION:
- Gender: Male | Female | Not declared
- Age: 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30+
- How do you rate **global** comfort in the classroom?
Very poor 1 | 2 | 3 | 4 | 5 Very good

COMFORT CATEGORIES IN THE LAST HOUR:
- How do you rate **thermal** comfort in the classroom?
Very poor 1 | 2 | 3 | 4 | 5 Very good
- How do you rate the **air quality** in the classroom?
Very poor 1 | 2 | 3 | 4 | 5 Very good
- How do you rate **visual** comfort in the classroom?
Very poor 1 | 2 | 3 | 4 | 5 Very good
- How do you rate **acoustic** comfort in the classroom?
Very poor 1 | 2 | 3 | 4 | 5 Very good

THERMAL COMFORT IN THE LAST HOUR:
- Thermal Sensation Vote:
Cold | Cool | Sufficiently cool | Neutral | Sufficiently warm | Warm | Hot
- Air movement is:
Too ventilated | Perfect | Not enough ventilated
- You are currently wearing:
Summer clothing | Standard clothing | Winter clothing

ACOUSTIC COMFORT IN THE LAST HOUR:
- You heard unwanted noises in the classroom for a:
Long time 1 | 2 | 3 | 4 | 5 Short time / not at all
- Does the noise allow you to understand what the teacher is saying?
Yes | Almost always | Almost never | No

VISUAL COMFORT IN THE LAST HOUR:
- The lighting in the classroom is:
Insufficient | Appropriate | Excessive

INDOOR AIR QUALITY IN THE LAST HOUR:
- How would you rate the air quality at the end of last hour?
Poor 1 | 2 | 3 | 4 | 5 Suitable
- Did you perceive odours from furniture, cleaning products, glues, adhesives, solvents, paints, cigarette smoke, printers and photocopiers?
Never 1 | 2 | 3 | 4 | 5 Often / Intensely

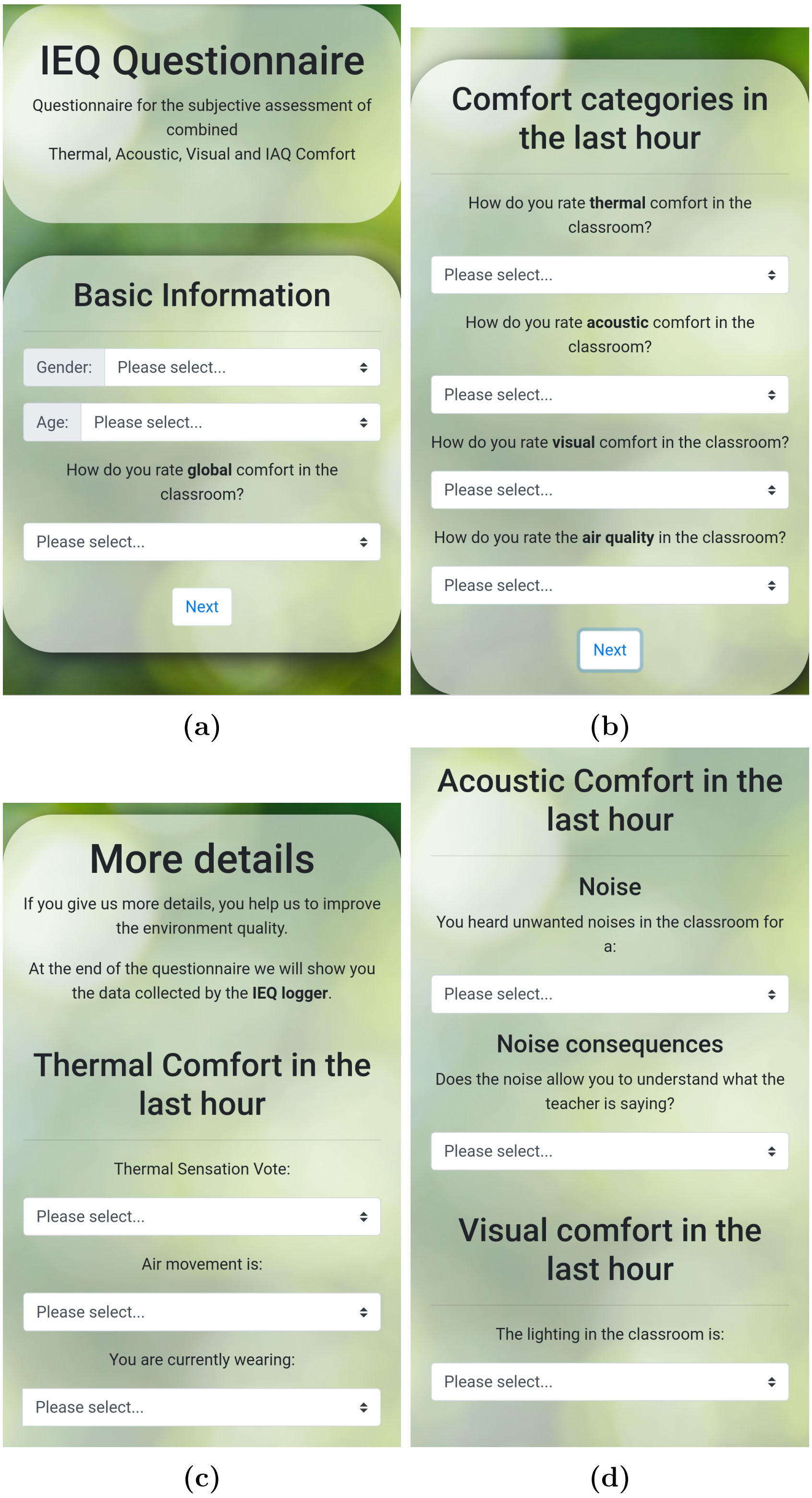
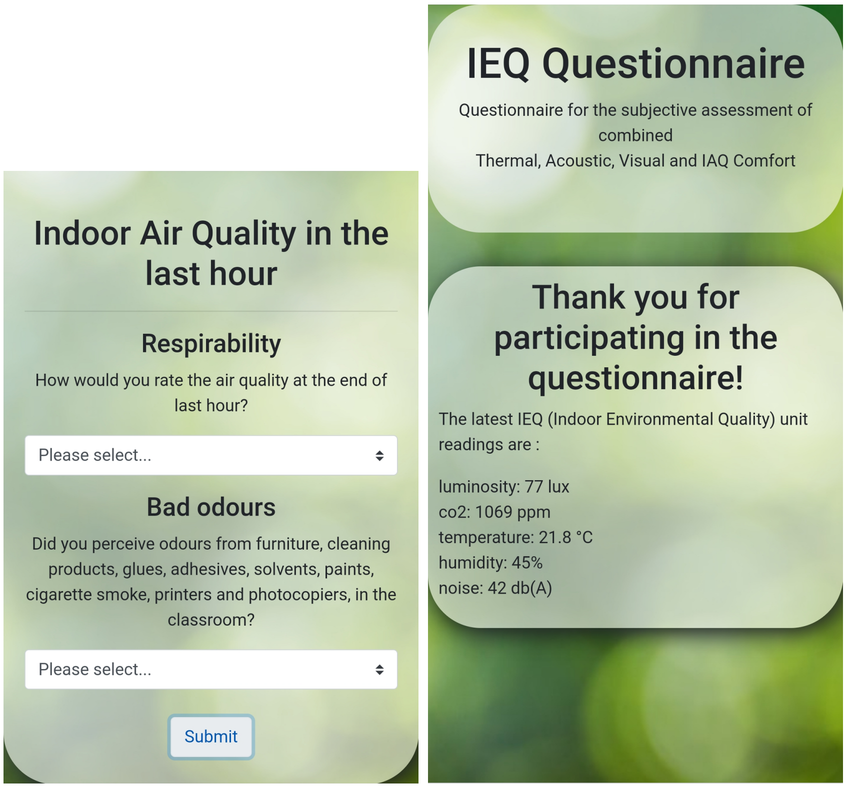


Figure 3.13: Graphical interface and sections of the questionnaire: Part 1



(a) (b)

Figure 3.14: Graphical interface and sections of the questionnaire: Part 2

The questionnaire is drawn for a university classroom [49], adopting the POE (Post-Occupancy Evaluation) method [48]. The response range for the comfort sensation was from 1 to 5 [21], where 1 is "very poor" and 5 is "very good". All participation was voluntary at the end of the lecture hour (session).

In this chapter, all wireless IEQ logger hardware and software has been described. Implementation, deployment and data collection phases have been presented. Now that all data have been collected, it is possible to perform data analysis, model building and characterization phases. The next chapter will present these phases.

Chapter 4

Data processing and proposed model

In this chapter, before proceeding with data analysis, a brief introduction to the MLR technique is presented (in Section 4.1). The final goal of this study is to identify a Predicted Indoor Global Comfort Index (P-IGCI) model starting from the measured physical quantities, firstly using the MLR (Multiple Linear Regression) methodology and finally checking whether a better model exists. To achieve that, data collected were analysed, and the correlation between the overall comfort (as stated in questionnaires) with the comfort categories and physical parameters was investigated (in Section 4.2). Then, the most suitable model for calculating a P-IGCI was identified and presented (in Section 4.3).

4.1 MLR

Multiple linear regression (MLR) analysis is a technique for analysing the linear relationship between a dependent variable (output/response variable) and two or more independent variables (inputs/predictors). The MLR can be adopted for two purposes: *explanatory*, that is, understanding and weighing the effects of independent variables on the dependent variable as a function of a given theoretical model; *predictive/estimative*, to identify a linear combination of independent variables to best predict/estimate the assumed value by the dependent variable.

In previous studies on comfort indices, MLR analysis is performed, or weights are assigned to different comfort categories in order to provide an indoor global comfort index (IGCI) [32, 34, 35, 88, 42, 43, 95]. From these studies, it is possible to generalise the formula for an IGCI:

$$IGCI = c + W_1 I_1 + W_2 I_2 + ... + W_n I_n \tag{4.1}$$

where c is the constant or intercept (which is zero when passing through the origin), I are the different comfort categories (expressed as indices of one or more physical parameters or as satisfaction/dissatisfaction indices), W are the corresponding weights, and n is the indices number taken into account.

In this study, objective data (from sensor measurements) and subjective data (from questionnaires) are averaged for every session. Specifically, the following objective and subjective data correspond to each session. The *objective data* are temperature average, humidity average, CO_2 concentration average, illuminance average, and noise level average. While the *subjective data* are thermal comfort question rate average, IAQ question rate average, visual comfort question rate average, acoustic comfort question rate average, and global comfort question rate average. Age and gender were not considered because the data collected were too homogeneous.

The objective data average was performed between collected measurements during the regarded session time interval, while the subjective data average was performed between questionnaires conducted at the end of every session. All these averages are thus pre-set as input for the MLR technique.

4.2 Data Analysis

The objective data collected for analysis can be summarized graphically in Figure 4.1, which shows the averages of the physical parameters per session. This figure consists of 5 histograms aligned with the X-axis. The X-axis represents the date and time of every session. The Y-axis represents the average measured over the session interval for each physical parameter.

Firstly, MLR was applied to investigate the relationship between the comfort categories and the global comfort question rate average, treated as Real Perceived Indoor Global Comfort Index (RP-IGCI). In this case, MLR is used for explanatory purposes, i.e. to understand and weigh the effects of each of the four categories on RP-IGCI reported in the questionnaires. The RP-IGCI stated in the questionnaires is the model's dependent variable, while the four comfort categories stated in the questionnaires are the independent variables.

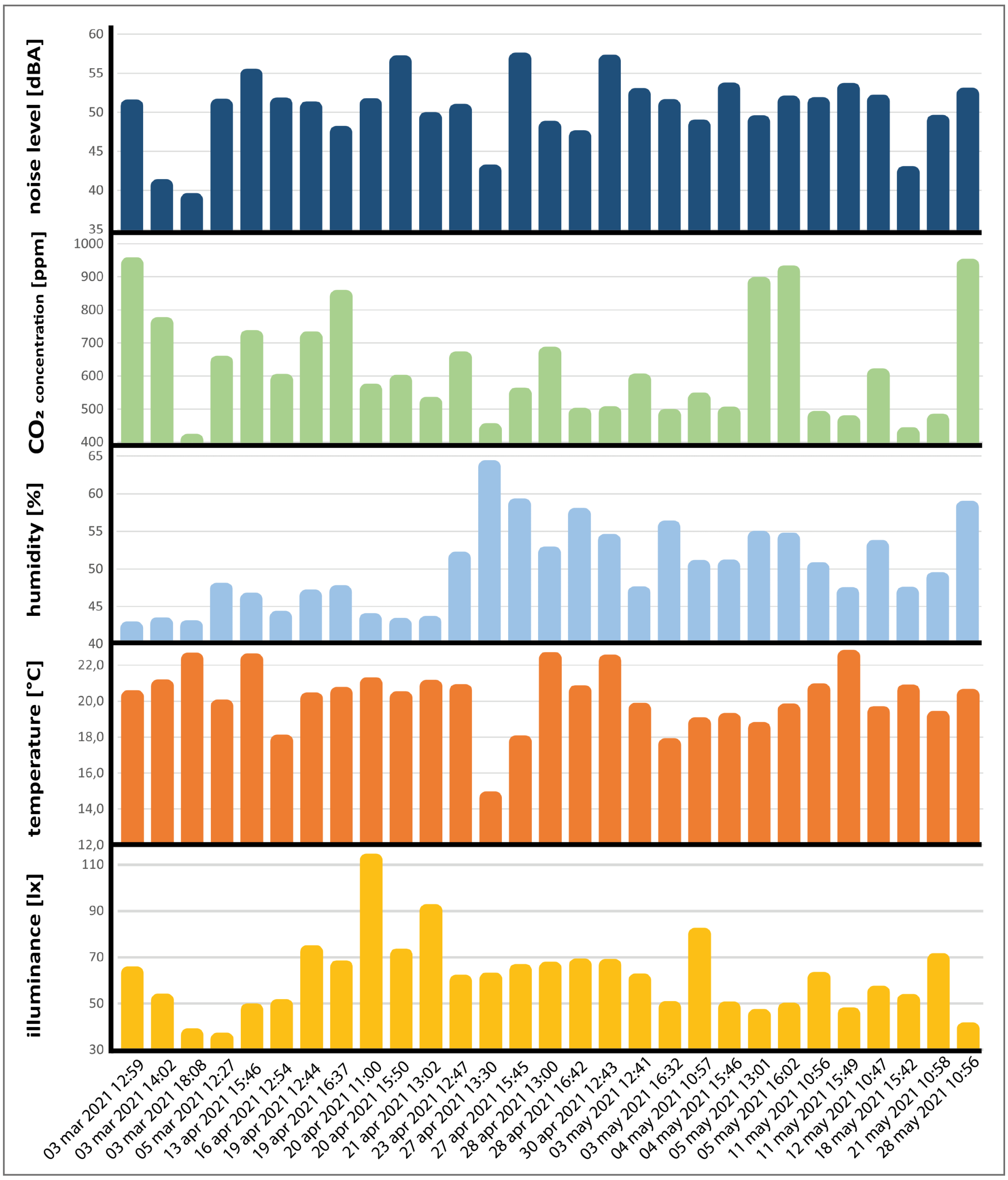


Figure 4.1: Physical parameter averages per session.

IBM SPSS software [149] was deployed to perform the MLR. The resulting standardised coefficients (beta) are 0.517 (for thermal comfort), 0.418 (for IAQ), 0.223 (for visual comfort), and 0.246 (for acoustic comfort) with the coefficient of determination R^2 equal to 0.74. These coefficients are reported on a percentage scale to illustrate the subjective impact of each comfort category on overall comfort:

- Thermal comfort: 37%
- IAQ: 30%
- Visual comfort: 16%
- Acoustic comfort: 17%

Secondly, MLR was applied to estimate the correlation between the physical parameters and the overall comfort average. Therefore, here MLR is used for predictive/estimative purposes, i.e., identifying a linear combination of objective variables to best predict the assumed value by the RP-IGCI. Thus RP-IGCI is held as the dependent variable, while the four main physical quantities measured (temperature, CO_2 concentration, illuminance, and noise level) are independent variables.

As mentioned above, all these variables are entered into the algorithm as averages within a lesson session (typically one hour). IBM SPSS software was deployed to perform the MLR. The resulting standardised coefficients (beta) are 0.377 (for temperature), -0.538 (for IAQ), -0.035 (for illuminance), and -0.022 (for acoustic comfort) with the coefficient of determination $R^2 =$ 0.49 and the Root Mean Square Error RMSE = 0.40 (Mean Square Error MSE = 0.16). By adding humidity in the MLR algorithm, the coefficient of determination does not change. For this reason, this parameter has been removed from the model, thus maintaining one physical parameter for each comfort category. This fact does not mean, in general, that humidity is not essential. On the contrary, as seen in Chapter 1, humidity, in combination with temperature, is one of the determining factors of thermal comfort. But evidently, in this specific case study, humidity is negligible compared to the other physical quantities. The XY scatter plots are presented below. These graphs show the relationship between the comfort category question rates (for all questionnaires) and the corresponding measured objective physical parameter like temperature (in Figure 4.2), CO_2 concentration (in Figure 4.3), illuminance (in Figure 4.4), and noise level (in Figure 4.5). A high comfort question rate corresponds to a high level of comfort, i.e. 5 equals the "very good" comfort answer from the questionnaire.

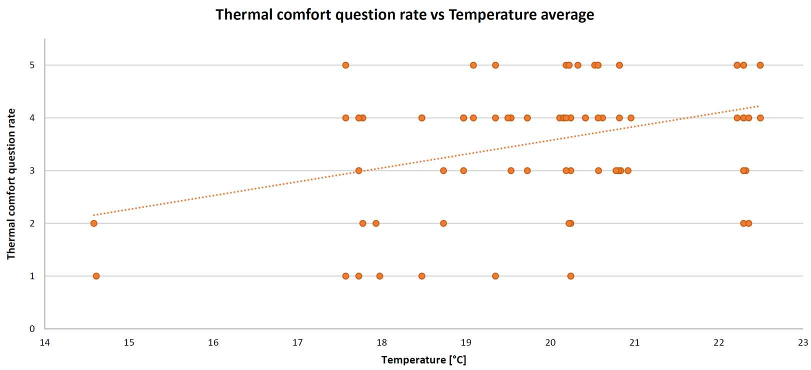


Figure 4.2: Thermal comfort XY scatter plot and trendline.

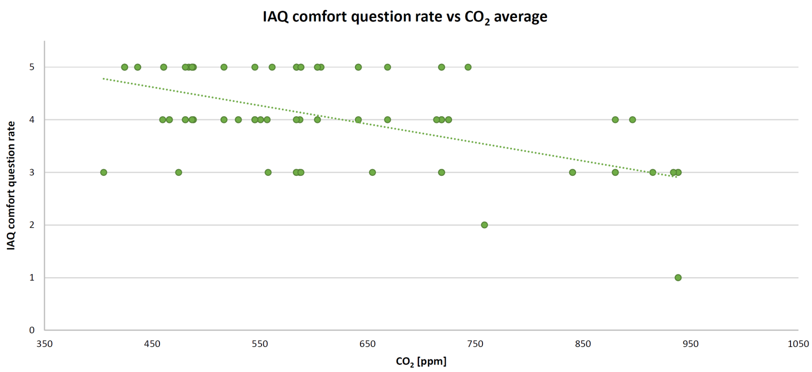


Figure 4.3: IAQ comfort XY scatter plot and trendline.

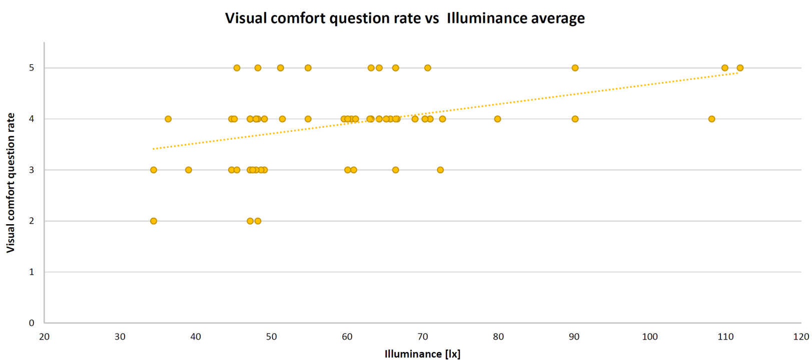


Figure 4.4: Visual comfort XY scatter plot and trendline.

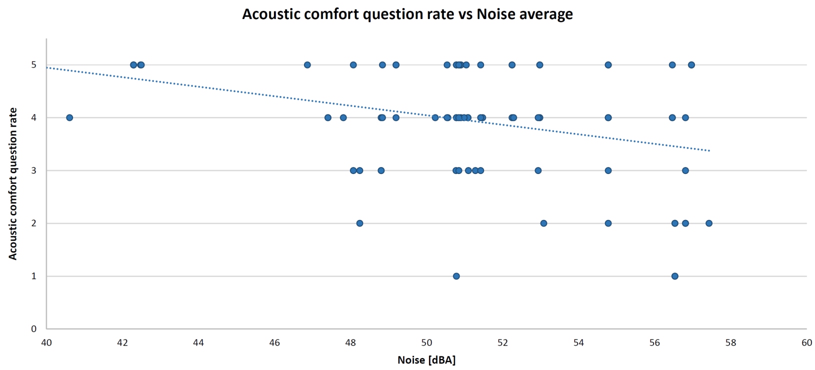


Figure 4.5: Acoustic comfort XY scatter plot and trendline.

For all measured physical quantities, a linear trendline was entered. Obviously, there is a satisfaction value range for temperature and illuminance (unlike noise level and CO2 concentration): high/low temperature and too much/too little illuminance generate dissatisfaction. For these reasons, in general, a polynomial (second-degree) trendline would be more suitable. However, in this specific case study, values corresponding to high temperatures (it was not summertime) or glaring light were never detected. Hence, for simplification, a linear relationship is also good.

4.3 Model building and characterization

This section aims to find a model that outputs a P-IGCI, derived from the measured physical quantities, as close as possible to the RP-IGCI, based on the questionnaires. Both of these indices are calculated over a session. The chosen algorithm is that returning the smallest possible RMSE. First, linear regression methods were investigated using SPSS, and the "Stepwise" method performed best. Stepwise criteria are a probability of F to enter ≤ 0.050 and a probability of F to remove ≥ 0.100. In this way, a model with "CO_2 concentration" and "Temperature" as "Variables Entered" is achieved. The coefficient of determination R^2 is 0.46 and RMSE is 0.38 (MSE is 0.14). The "Regression Learner" App in MATLAB [150] was then used to test the "Stepwise" method against other models. Temperature, humidity, CO_2 concentration, illuminance, and noise level were entered as "Predictors", and RP-IGCI is entered as "Response". The App tested 19 different algorithms and calculated the corresponding RSME, whose values are listed in Table 4.1. "Stepwise" was confirmed as the best performing model, not only among the linear regression methods but across all tested methods.

In the following, "Predicted vs Actual plot" (Figure 4.6), "Residuals plot" (Figure 4.7), and "Response plot" (Figure 4.8) are illustrated. A high indoor global comfort index either real perceived (RP-IGCI) or predicted (P-IGCI), equals a high level of comfort i.e. 5 corresponds to the maximum comfort.

Table 4.1: Tested algorithms and corresponding RMSE/MSE.

Model	**Method**	**RMSE**	**MSE**
Linear Regression	Linear	0.40	0.16
	Interactions Linear	0.40	0.16
	Robust Linear	0.42	0.18
	Stepwise Linear	**0.38**	**0.14**
Regression Trees	Fine Tree	0.58	0.34
	Medium Tree	0.51	0.26
	Coarse Tree	0.51	0.26
Support Vector Machines	Linear SVM	0.47	0.22
	Quadratic SVM	0.47	0.22
	Cubic SVM	0.56	0.31
	Fine Gaussian SVM	0.51	0.26
	Medium Gaussian SVM	0.51	0.26
	Coarse Gaussian SVM	0.47	0.22
Gaussian Process Regression	Rational Quadratic GPR	0.55	0.30
	Squared Exponential GPR	0.53	0.28
	Matern 5/2 GPR	0.53	0.28
	Exponential GPR	0.52	0.27
Ensembles of Trees	Boosted Trees	0.51	0.26
	Bagged Trees	0.48	0.23

In Figure 4.6 the observations "cloud" around the perfect prediction line is displayed. This figure also reveals that, in general, a good overall comfort was perceived (all points are in the range 2.5 - 5). In Figure 4.7, the P-IGCI residuals on the true response (RP-IGCI) are shown. All points are within the -1 and +1 range. This is a very good result since the scale is potentially between -5 and +5. The errors can be better highlighted in Figure 4.8. The error is minimum when RP-IGCI point coincides with P-IGCI point, while it is maximum when RP-IGCI point is far from P-IGCI point, i.e. when the red line is longer. The error ranges from a minimum of zero (in session 17, with RP-IGCI = P-IGCI = 3.67) to a maximum of less than 1 (in session 23 with RP-IGCI = 4.67 and P-IGCI = 3.74). The result is positive since it is a maximum error, and the RMSE is equal to 0.38 anyway.

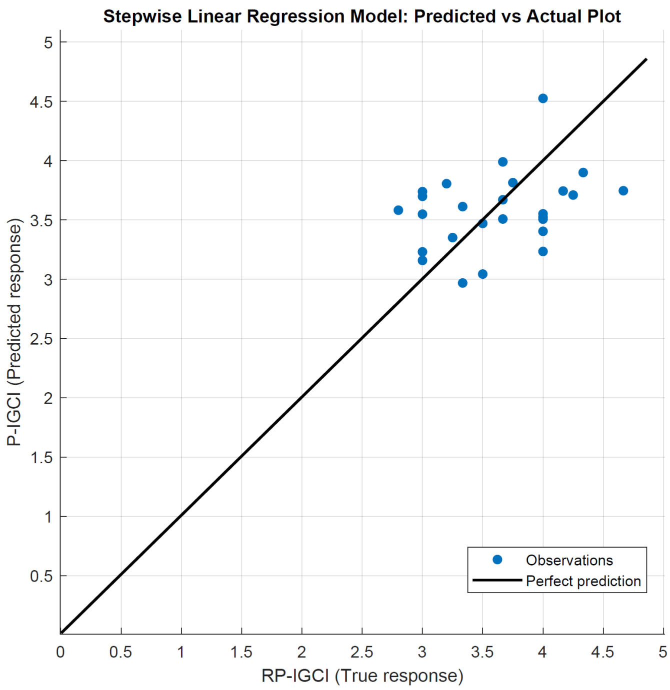


Figure 4.6: Predicted (P-IGCI) vs Actual (RP-IGCI) plot.

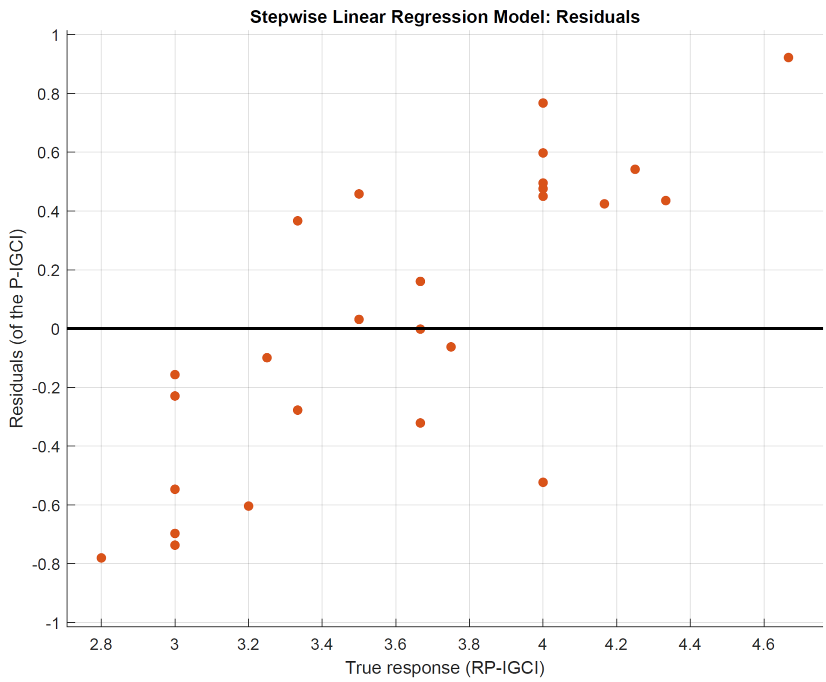


Figure 4.7: P-IGCI Residuals on true response (RP-IGCI).

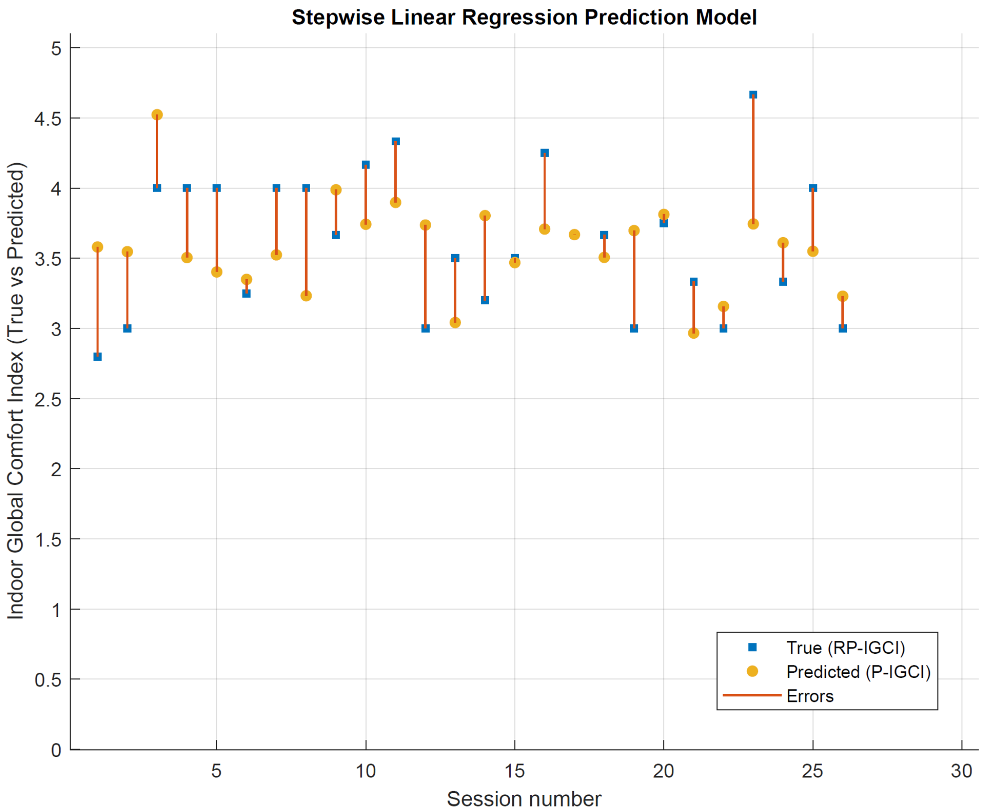


Figure 4.8: "True (RP-IGCI) vs Predicted (P-IGCI)" error for every session.

This means, for example, that if a session average comfort of 4.00 out of 5 is perceived, the model:

- In the worst case is hardly going to 3 or 5 (± 1);
- On average, it will return 4.38 or 3.62;
- In the best case, it coincides with 4.00.

This global picture is very good since these are estimates on subjective parameters and a 5-point comfort scale.

4.4 Out-Of-Sample validation

Out-Of-Sample (OOS) testing is one of the most widely accepted model validation techniques for assessing how statistical analysis results can generalise to an independent data set. Considering that classes started again in October, an OOS validation is performed on the collected data during this period. October is a reasonably 'similar' month to the period previously examined. On average, this month is neither excessively hot nor excessively cold. It is evident that the best model will most likely not be the linear model in both very hot periods (with windows open) and cold periods (with the heating on). However, the proposed model provides a global comfort estimate with a minimum error for the case study and in mild periods. The objective data collected for OOS validation can be summarised graphically in Figure 4.9, which shows the averages of the physical parameters per session. As for Figure 4.1, this chart consists of 5 histograms aligned with the X-axis. The X-axis represents the date and time of every session. The Y-axis represents the average measured over the session interval for each physical parameter.

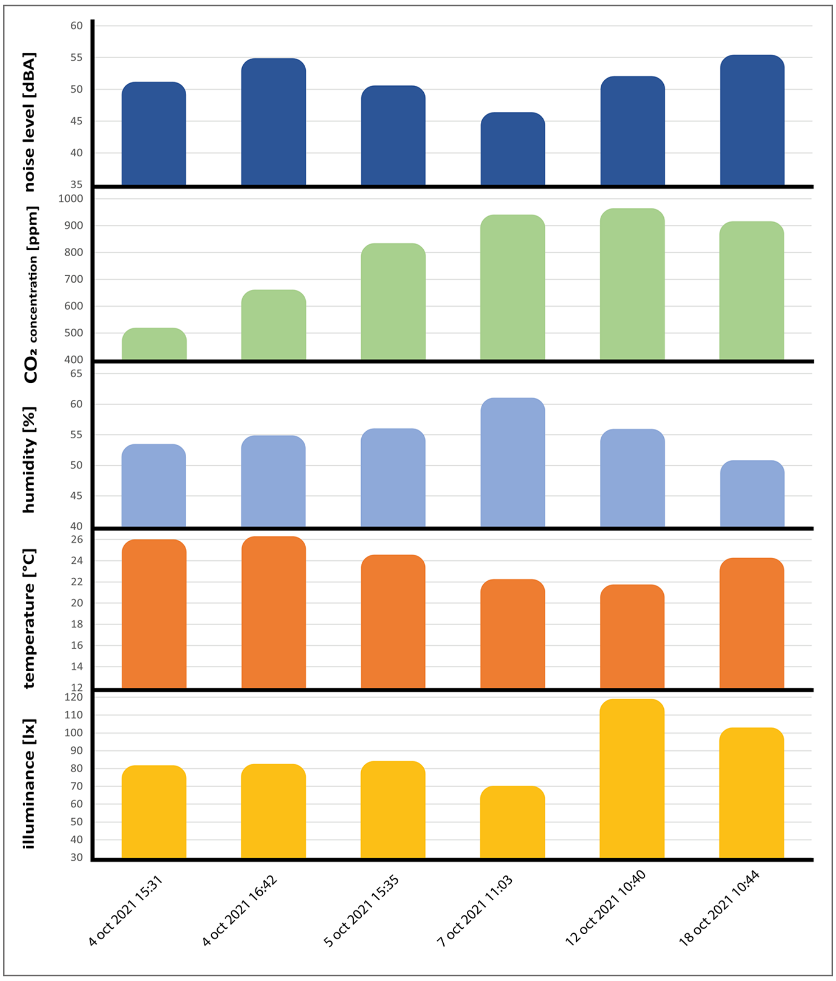


Figure 4.9: Physical parameter averages per session.

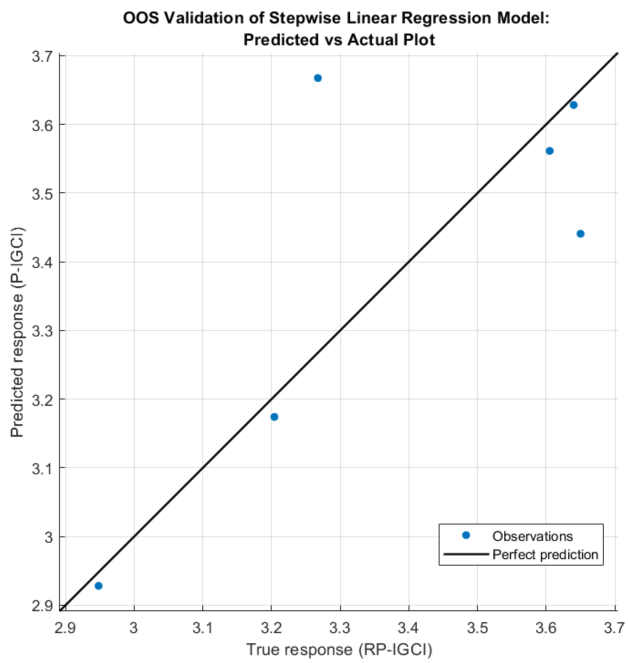


Figure 4.10: Predicted (P-IGCI) vs Actual (RP-IGCI) plot.

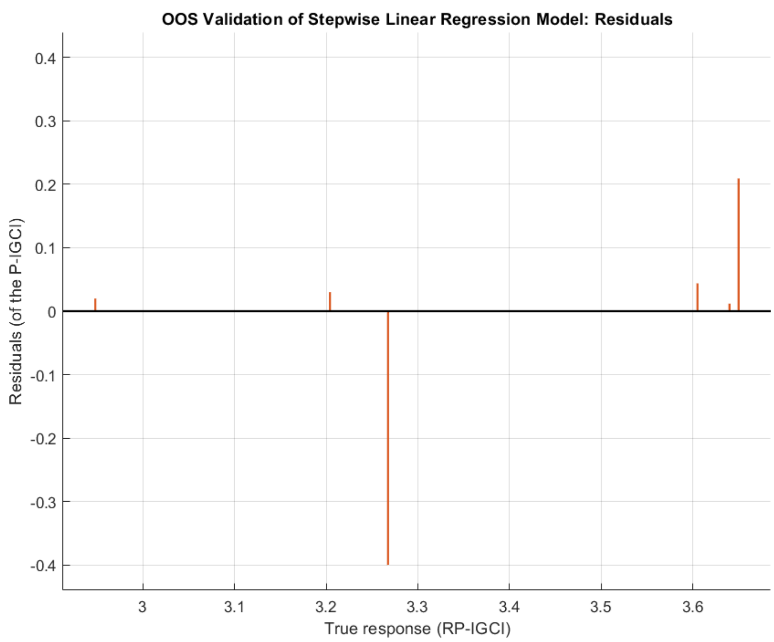


Figure 4.11: P-IGCI Residuals on true response (RP-IGCI).

In this chapter, several steps were taken to achieve the proposed model. Now, the idea is to develop an "automatism" of what has been done "manually" in this chapter. The next chapter proposes a development of the virtual sensor, i.e. a software sensor that provides a perceived global comfort index estimate, automatically choosing the lowest error model.

Chapter 5

Virtual Sensor

A virtual sensor is a pure software sensor that autonomously produces signals by combining and aggregating signals that it receives (either synchronously and asynchronously) from physical or other virtual sensors [151]. The basic idea is to develop a virtual sensor that provides the P-IGCI in real-time. In this chapter, the main steps, the software architecture, and the final graphic interface will be outlined.

5.1 P-IGCI virtual sensor: main steps

The main steps that the virtual sensor has to perform to generate the output (the Predicted-Indoor Global Comfort Index) are shown in Figure 5.1.
These steps are briefly described below.

- *Data Collection.* In this first step, all completed questionnaires (both complete and incomplete) are retrieved via the server APIs. For each questionnaire:
  - The objective measurements of the previous hour are retrieved;
  - A call to the API is launched;
  - The averages and standard deviations of the collected objective values are calculated. This last step is optional: it is only needed if the data is exported.
- *Session generation.* Questionnaires that have been completed correctly (i.e. all fields containing the required values) are selected. The questionnaires are grouped into sessions according to their filling in time. The time after completion of the first questionnaire is chosen as the session time range. For example, questionnaires completed at 15:00,

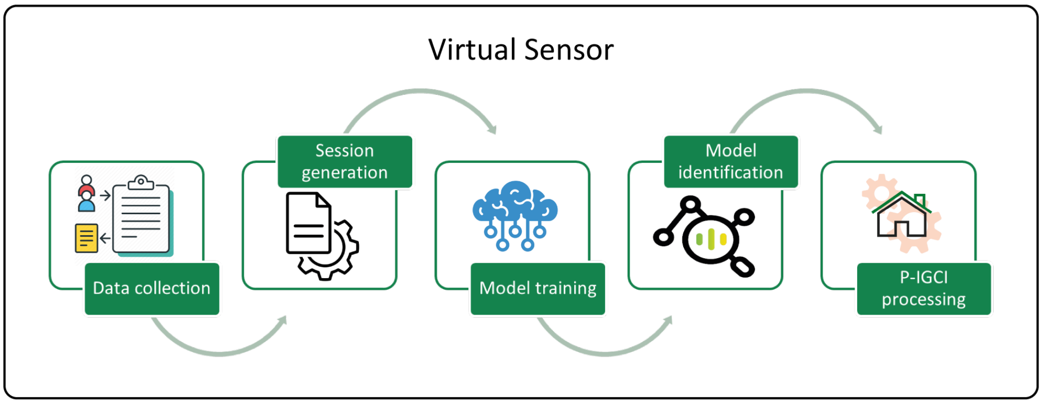


Figure 5.1: P-IGCI virtual sensor: main steps.

15:01, 15:05, 15:32 and so on (up to a 15:59 max.) will be grouped into the same session. This session indicates the lesson carried out the hour before, i.e. it refers to the objective parameters collected from 15:00 to 15:59. The average of each measured physical parameter in the session time range is calculated. Obviously, questionnaires carried out completely outside the session time range are discarded. The export of the sessions is performed on two files (in CSV extension): "input" and "output" for the MATLAB models. These exported files are then used for the MATLAB engine but can, of course, be used in other applications such as Microsoft Excel, OpenOffice Calc, etc.

- *Model training.* The MATLAB engine for running MATLAB files from Python is started. The data saved (in CSV files) on the MATLAB workspace are loaded. Nineteen regression models with different methods are trained (for more details, see Table 4.1. Finally, the RMSE is calculated for each trained model.

- *Model identification.* The search for the trained model with the lowest RMSE begins. Once the model is identified, it is saved in the disk as a MATLAB file. Additional information useful for the next steps is added to the created model. The MATLAB engine is stopped. This procedure only serves to create a model based on the data you want to provide as input; otherwise, the next step can be continued directly.

- *P-IGCI processing.* MATLAB engine is started from Python for running MATLAB files. The trained regression model is loaded. The last objective detection measured by the IEQ Logger is retrieved from the API (regardless of the input date). The model to predict the P-IGCI is adopted. The data to the database is sent via an http request. A history of calculated P-IGCIs is stored in the database. When the web page is loaded from a device, the last calculated P-IGCI will be returned.

5.2 P-IGCI virtual sensor: software architecture

The virtual sensor software architecture starts from the following assumptions: - adopting the MATLAB engine as the central part for processing the P-IGCI; - adopting the Python language for developing the core of the system and the related libraries. Using MATLAB provides the advantage of having applications such as "Regression Learner" and tools for creating and exporting data, graphs, etc. In addition, MATLAB provides a Python package that allows invoking its functions from Python. P-IGCI virtual sensor software architecture is shown in Figure 5.2.

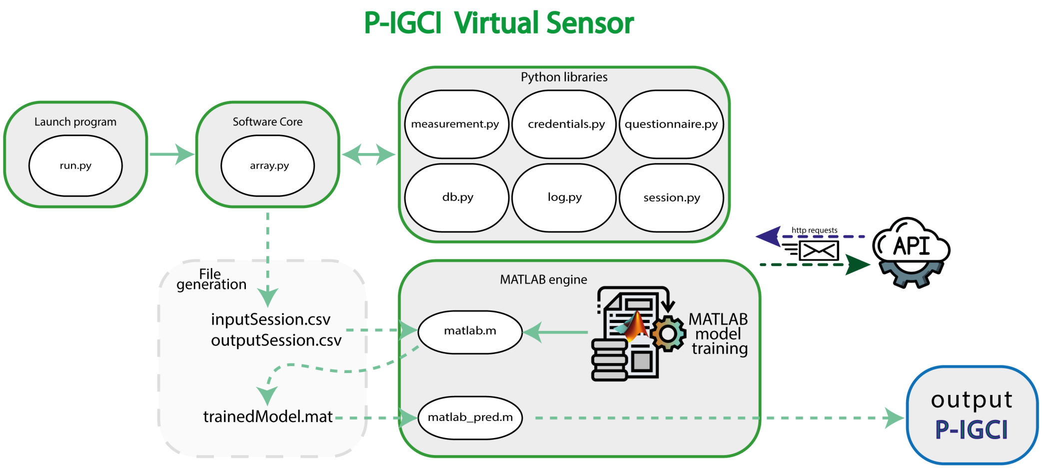


Figure 5.2: P-IGCI virtual sensor: software architecture.

The "Software core" is initiated by the "Launch program" and performs the main functions by communicating with the "Python libraries". The "Software core" purpose is to generate data to be used as inputs in the MATLAB engine. Both the Python libraries and the MATLAB engine use

APIs to communicate with the database. MATLAB engine will provide the best-trained model and finally the P-IGCI value based on the last collected physical parameters. In the following, the files represented in the virtual sensor architecture are described.

run.py
Run.py represents the project file to be run, where it is possible to choose the execution options on both questionnaires and sensor readings. This file is extremely flexible and can be changed according to needs. In order to automate the process previously seen in Figure 5.1, the file has been structured in 5 steps:

1. Data collection;
2. Session generation;
3. Model training;
4. Model identification;
5. P-IGCI processing.

In addition, this file can be replaced with a framework for automation as a web service. In this case, its purpose will be to constantly run the prediction algorithm of a P-IGCI. Thus, every time a new questionnaire is filled in, all regression models have to be retested to identify which one fits best.

array.py
This class represents the core software. It handles all the functions that can be done on the collected data, such as retrieving data, running statistics, saving to file, and starting the MATLAB engine. The class contains 2 arrays:

- arrayQuestionnaire is the set of all "questionnaire" objects;
- arraySession is the set of all instances of the class "sessions".

First of all, it is necessary to retrieve from the API service all the questionnaires that have been made in order to perform operations on them. Alternatively, it is possible to retrieve the questionnaires handled over a certain time range. For each questionnaire, the physical parameters of the previous hour (recorded in the database) are retrieved (thanks to the APIs). In this class, all methods to perform statistics on the measured physical parameters are implemented:

- Calculating averages;
- Calculating standard deviations;
- Performing linear regression.

This class also allows the questionnaires retrieved from the API server to be exported in CSV format. Some questionnaires may be incomplete due to the user's failure to submit answers when filling in the form. These are still exported but with the corresponding fields empty. The mechanism for generating sessions has been implemented with the following criteria:

- All questionnaires completed within the range of one hour are considered in the same session;
- Only questionnaires containing the corresponding measured physical parameters are considered;
- Within a session, the measured physical parameters are averaged.

The session array is then converted to Pandas DataFrame, and a table is saved in CSV format containing all the session parameters. This procedure generates the files:

- inputSessions.csv (containing the averages of all the physical parameters measured for each session);
- outputSessions.csv (containing the RP-IGCI values from questionnaires for each session).

These files with CSV extension are created to be either directly analysed or used as input to the matlab.m file (see later).

The array.py file needs "Python libraries" to perform all its functions. The main libraries are *questionnaire.py, session.py, credentials.py, db.py, log.py,* and *measurements.py*. These are briefly described below.

questionnaire.py
The class contains all the fields of the online questionnaire in the form of attributes. In addition, it has an array of measured physical parameters corresponding to the hour before the questionnaire completion.

session.py
The class contains the objective data (the measured physical parameters) and the subjective data (RP-IGCI) as attributes. It is needed to obtain the

sessions to be exported to a file and then processed in MATLAB. This class allows the measured physical parameters corresponding to each questionnaire to be averaged.

credentials.py
The constants concerning the authentication token and the web address of the APIs are stored in this file.

db.py
This file contains the *request* class and its objectives are:

- Retrieve all questionnaires from the database;
- Retrieve the questionnaires from the database within a time interval;
- Retrieve the physical parameters measured in the hour before the questionnaire completion.

The class provides communication with the APIs.
log.py
The file contains the program's standard terminal output. Different types of output will be obtained depending on the settings assigned. A constant is defined:

- "True" to log on screen;
- "False" to not log on screen.

In detail:

- LOG is used to print the results of operations on the screen;
- LOGINFO is used to print on the screen the information relating to what is being executed;
- ERROR is used to display errors detected by the code.

This file can be useful for future modifications, but it was especially helpful in the development phase.

measurements.py
This is the class that contains, as attributes, the various physical measured parameters identified with the corresponding timestamp. It also defines methods for displaying this information.

matlab.m
The file checks for the best regression method among all the exported functions by MATLAB's "Regression Learner" application (currently 19). These functions are derived from the trained models with the exported databases and adopt 5-fold cross-validation. This is certainly one of the most accredited and popular techniques for cross-validation, but as the data grows, it can take a longer time to run. This happens because the model has to be trained repeatedly. The file performs the following steps:

- Gets as input the two files in CSV format (inputSession.csv and OutputSession.csv);
- Converting these files into a formatted format for regression functions;
- Training the models by establishing the corresponding methods and RMSE;
- Identifying the model with the lowest RMSE as "valid";
- Exporting the obtained model structure to a file with a MAT extension (trainedModel.mat).

In this way, the best-trained model can also be reused in any program that runs MATLAB code.

matlab_pred.m
The file imports the trained model from matlab.m. It also retrieves the last detection performed by the IEQ Logger via an HTTP request to the APIs. The output is an array sent as a parameter to the "predictFcn" function. This function performs the data prediction starting from a trained model. The predicted value (P-IGCI) is added to the payload of an HTTP request and saved to the database via the APIs. The database keeps track of all predicted values.

For more information on each individual file, please refer to the Appendix D.

5.3 P-IGCI virtual sensor: graphical interface

The graphical interface of the virtual sensor (as for the measurements webpage and the questionnaire) was designed using the RWD (responsive web design) technique. In this way, it is graphically and automatically adapted on PCs, tablets, smartphones, etc. The language used for the implementation was PHP (see Appendix D). The predicted value (P-IGCI) saved in the database is called up via APIs. In addition, the best method identified is indicated. Figure 5.3 shows the final layout on a smartphone.

Figure 5.3: P-IGCI virtual sensor: main steps.

5.4 Virtual sensors for sustainable comfort

Building management system (BMS), also known as BCS (building control system) or BAS (building automation system), aims at covering comfort and energy. Therefore, the resulting high indoor global comfort index must be integrated with energy and economic saving strategies.

Building control schemes for indoor environments can be divided into conventional controllers and intelligent controllers. Conventional controllers include various systems, such as on/off switching controllers, thermostats, P (proportional), PI (proportional-integral) or PID (proportional-integral-derivative) controllers. Intelligent controllers, on the other hand, can be subdivided mainly into [152]:

- Learning-based methods (including artificial intelligence, fuzzy systems and neural networks)
- Model-based predictive control (MPC) methods (following the principles of the classical controls)
- Agent-based control systems (adopting MAST, multi-agent system technology).

Agents are generally virtual or physical entities that cooperate rationally in an environment with both perceiving and influencing qualities. Intelligent building control with a MAST structure allows [152]:

- The learning of occupancy trends in the building;
- The co-ordination of energy resources;
- The ability to respond to internal environmental conditions in real-time.

In this context, adopting the P-IGCI virtual sensors is very useful as they can:

- Collecting important objective information through the measured physical parameters;
- Collecting important subjective information through occupants' feedback;
- Providing the perceived global comfort level.

The final idea is to use several virtual sensors to monitor the indoor global comfort index within different building types. A subsequent analysis would allow understanding how energy-saving has an impact on the comfort in buildings such as smart homes, smart buildings and green buildings.

Chapter 6

Conclusions and future work

In this work, a comprehensive analysis of comfort in the indoor environment aims to define and measure an Indoor Global Comfort Index. This IGCI can be employed for different optimisation and energy-saving challenges. The task has been done by analysing the literature on the topic reviewing all previous work on the global comfort index, conducting critical analysis, and investigating the many aspects of comfort. A new methodology based on an integrated approach to choose and characterise a model for estimating the perceived global comfort index has been proposed. Finally, this procedure has been automated by implementing a predicted indoor global comfort index virtual sensor.

Chapter 2 outlined a survey of global comfort indices in indoor environments, often called "overall comfort index", "combined comfort index", "IEQ index", even without the term "comfort" or using other words such as "satisfaction"/"dissatisfaction". The first part presented an overview of the main concepts, factors, methodologies and indices of the individual comfort categories (also known as IEQ "factors", "aspects", "elements", etc.). This overview has made it possible to examine and discuss the global comfort indices proposed in the literature. Among them, the one to be chosen is strictly related to the case study that most closely resembles the real situation to be examined. For example, the choice could be based on:

- Building type: TEQE, [32] [33], IEI, [96], I, [35], PDIEQ, [88], IEQ $_{index}$ [43] for office buildings and DEQI [97] for dwelling or residential buildings;
- Geographical location and/or climate: $IEI_{(AHP)}$ [34] in Taiwan, PDIEQ [88] in Hong Kong, S [42] in Beijing and Shanghai, IEQ_{index} [43] in the UK, I [35] in Europe or I_{CC} [95] in temperate climates;

- Ventilation system: IEQ_{index} [43] for mechanically ventilated buildings or PDIEQ [88] for air-conditioned buildings;

- Other factors: public/private buildings, new/existing buildings, etc.

In Section 2.7, pre-established targets about Global Comfort Index have been reached:

- Understanding the comfort categories importance and their impact on the global IEQ index;

- Identifying common aspects of GCIs, IGCI requisites (with pros and cons), main weighting techniques, models, strategies and methodologies adopted in this research field;

- Evaluating the possibility of making use of machine learning.

EN15251 [66] was the first international standard to address indoor environmental parameters (acoustic comfort, lighting, air quality and heat), and provided a number of parameters to help design and assess energy performance in buildings. This standard was developed in 2007 to help implement the Energy Performance of Buildings Directive in Europe [7], and has now been updated and incorporated into the new standard EN16798-1: 2019. In this chapter, the focus has been intentionally set on the overall user comfort indices and their related components. For a better reading, the chapter has also included an initial overview of these four aspects, the corresponding indices (or methodologies) and the various strategies to achieve the best comfort in indoor environments. Generally speaking, in smart and green buildings (especially in smart homes), life quality can also vary in accordance with the possible presence of other artificial systems, such as safety systems (e.g. alarms and/or video-surveillance), cleaning systems (e.g. robot vacuum cleaner), appliances (low noise and high energy efficiency), gardening systems (e.g. watering systems and/or lawnmower robot), TV/Speakers, entertainment systems, etc. Moreover, for the same indoor environment, allowing occupants to change the environmental conditions tends to increase their satisfaction [153].

Chapters 3 and 4 presented a wireless IEQ logger with the DIY philosophy. A simple but comprehensive hardware and software implementation has been proposed. The system is designed to monitor all the main types of comfort that represent indoor environmental quality, i.e. thermal comfort, indoor air quality (IAQ), visual comfort, and acoustic comfort. The wireless IEQ logger hardware development was possible thanks to the employment of different sensors connected to the Raspberry Pi board. This board operates in the open-source ecosystem. Other employed hardware concerns instruments adopted for the calibration and testing phases of the different sensors. The structure of the questionnaire and, in general, of the entire software allowed the organised collection of several objective and subjective data. The specific case study concerns the logger's use in a university classroom. However, the system implemented is relatively low-cost and can be easily reproduced for applications in any indoor environment. The total cost of the IEQ logger was about 150 €. The price is about the same as a medium quality IEQ logger. The problem is that devices measuring all examined parameters can hardly be found on the market. For example, Netatmo NHC-IT [154] costs about 150 €, but does not measure illuminance. A professional air quality detector, such as Airthings Wave Plus [155], costs about 250 €. This device also measures other parameters such as Radon and TVOCs but does not measure noise and illuminance. Therefore, the Wireless IEQ logger developed is as cheap as a mid-range product for indoor air quality but measures all physical parameters related to IEQ categories. Furthermore, it has greater processing capacity, thanks to the Raspberry Pi, allowing for even upgrade capability. The methods adopted allowed the main objective to be achieved: identifying a P-IGCI model starting from the measured physical quantities. The MLR technique between subjective data allowed to detect the weights of the different comfort categories (thermal comfort 37%, IAQ 30%, visual comfort 16%, acoustic comfort 17%). A first predictive model was found through the MLR technique between objective data and overall subjective comfort (RP-IGCI). Finally, by testing and examining 19 different algorithms, the MLR model with the Stepwise method turns out to be the best one with the lowest RMSE/MSE. The SPSS (by IBM) and MATLAB (by MathWorks) software were of great help and fundamental importance to achieve these results. Interestingly, the physical quantities excluded from the model identified correspond to the comfort categories that also subjectively had the lower weight (i.e. visual comfort and acoustic comfort). This result, in general, does not mean that these comfort categories (or the corresponding physical parameters) are useless. The reasons why MLR with stepwise method discarded these two parameters are: i) Objective difficulty in measurement (e.g. voice of the teacher to be distinguished, light varying from the position in the room, etc.); ii) Always satisfactory levels: illuminance almost always above 50lx (as per EN 12464-1) and noise level always below 60dBA (as per World Health Organization Community Noise Guidance). In this regard, there are studies [39, 90] stating that the level of satisfaction with a comfort type influences the classification of that condition. In other words, the more dissatisfied people are with a condition, the more weight will be given to it; conversely, when people are satisfied with a certain condition, it is considered of less importance [100]. Finally, it is interesting to note that also in several studies on indoor environments [87, 34, 35, 38, 90, 43], IAQ and thermal comfort are considered the most relevant categories. This fact is even more evident in different Green Building certification schemes [47], in particular by KLIMA [156], LiderA [157], and NABERS [158].

Chapter 5 demonstrated how it is possible through a P-IGCI virtual sensor to automate what was done manually in Chapter 4. This software allows to predict a perceived indoor global comfort index and identify the best model to achieve it. Finally, the importance of virtual sensors for sustainable comfort was outlined in Section 5.4. Some future work is proposed below.

6.1 Future work

The future effort is mainly addressed towards two directions: i) Improving the mathematical model of the global comfort index; ii) Applying machine learning to provide advanced adaptive capabilities to the wireless IEQ logger.

Regarding global comfort indices, possible future research could be to analyse the four fundamental comfort parameters further, seeking and integrating new indices for the different comfort categories. This study would allow a better IGCI to be obtained that might include as many aspects as possible while maintaining a high objectivity level. Another option would be to describe the strategies and techniques to maintain good global comfort levels in buildings with maximum energy efficiency (e.g. integrating energy consumption analysis and using machine learning or, more generally, AI algorithms). This option would produce nZEB (nearly Zero Energy Building) and green buildings with maximum comfort.

Concerning wireless IEQ logger, future research could involve using artificial intelligence algorithms, such as machine learning techniques, to identify an increasingly accurate predictive model of global comfort. However, these techniques require large amounts of data to be efficient. In this case, more data could be collected by producing more wireless IEQ loggers, installing them in different classrooms and collecting data for a much longer period. A further possible investigation could concern an even more precise measurement of thermal comfort, i.e. following the ISO 7730 standard [10] and adopting the instrumentation required by the ISO 7726 standard [136]. The most commonly adopted physical parameters for each comfort category were considered in this research.

Finally, another option would be to collect other physical parameters by considering, for example, TVOCs (Total Volatile Organic Compounds) [159, 160, 161, 162], data from weather, and increasing the types of sensors to check if there is a considerable influence of these on the overall comfort.

Appendix A

Acronyms

AI Artificial Intelligence.

AHP Analytic Hierarchy Process.

ANN Artificial Neural Network.

AS Air Speed.

ASHRAE American Society of Heating Ventilation and Air-conditioning Engineers.

AT Air Temperature.

BAS Building Automation System.

BCS Building Control System.

BMS Building Management System.

BN Bayes Network.

BPG Building Performance Gap.

BPNN Back-Propagation Neural Network.

BREEAM Building Research Establishment Environmental Assessment Method.

CI Clothing Insulation.

CIBSE Chartered Institution of Building Services Engineers.

CIE International Commission on Illumination (Commission internationale de l'éclairage).

CRI Colour Rendering Index.

DEQI Dwelling Environmental Quality Index.

DGNB German Sustainable Building Council (Deutsche Gesellschaft für Nachhaltiges Bauen).

DT Decision Tree.

EMF ElectroMagnetic Field.

FFNN Feed-Forward Neural Network.

GCI Global Comfort Index.

GLM General Linear Model.

HVAC Heating, Ventilation and Air Conditioning.

IAPI Indoor Air Pollution Index.

IAQ Indoor Air Quality.

IDI Indoor Discomfort Index.

IEI Indoor Environmental Index.

IEQ Indoor Environmental Quality.

IGCI Indoor Global Comfort Index.

IoT Internet of Things.

LEED Leadership in Energy and Environmental Design.

MAST Multi-Agent System Technology.

ML Machine Learning.

MLP MultiLayer Perceptron.

MPC Model-based Predictive Control.

MR Metabolic Rate.

MRT Mean Radiant Temperature.

NABERS National Australian Built Environment Rating System.

NC Noise Criterion curves.

NCB Noise Criterion Balanced.

NIOSH National Institute for Occupational Safety and Health.

NNARX Neural Network Autoregressive with Exogenous Input.

NR Noise Rating.

OOS Out Of Sample.

OT Operating Temperature.

PDAC Percentage of Dissatisfaction in Aural Comfort.

PDIAQ Percentage of Dissatisfaction in Indoor Air Quality.

PDIEQ Percentage of Dissatisfaction in Indoor Environmental Quality.

PDTC Percentage of Dissatisfaction in Thermal Comfort.

PDVC Percentage of Dissatisfaction in Visual Comfort.

PI Proportional-Integral.

PID Proportional-Integral-Derivative.

PMOT Prevailing Mean Outdoor Temperature.

PMV Predicted Mean Vote.

PNC Preferred Noise Criterion.

PPD Predicted Percentage of Dissatisfied.

P-IGCI Predicted Indoor Global Comfort Index.

RBFN Radial Basis Function Network.

RC Room Criterion.

RF Random Forest.

RH Relative Humidity.

RP-IGCI Real Perceived Indoor Global Comfort Index.

SVM Support Vector Machine.

TCV Thermal Comfort Vote.

TEE Equivalent Effective Temperature.

TEER Equivalent Effective Temperature depending on Radiation.

TEQE Total Environmental Quality Evaluation.

THI Temperature Humidity Index.

TSV Thermal Sensation Vote.

TVOC Total Volatile Organic Compound.

USGBC United States Green Building Council.

WSN Wireless Sensor Network.

Appendix B

IEQ logger implementation

B.1 Software Core

B.1.1 *run.py*

```
    print("=========================================
    \n============== IEQ Logger ================
    \n======== Powered by Stefano Riffelli ========
    \n=========================================\n")

# Imports list.
from lib.luminosity import Light
from lib.co2_level import Co2
from lib.temperature import Temperature
from lib.humidity import Humidity
from lib.noise_level import Audio_processing
from lib.request import HTTPRequest
from lib.file import File
from lib.get_weather import Weather
from datetime import datetime
import json

# Opening File.
f = File("/home/PROJECT_PATH/BackupErrors.txt")

# Instances list.
# All the classes defined in the different files (inside
    ↪ the "lib" folder) are instantiated
```

```
luminosity = Light()
co2 = Co2()
temperature = Temperature()
humidity = Humidity()
audio = Audio_processing()
HTTPRequest = HTTPRequest()
# The city and its state are given to the constructor of
    ↪ the Weather class.
w = Weather("Urbino","Italy")

# Request parameters.
try:
    lux = luminosity.get_lux()
except:
    lux = -1
    print("Error getting lux")
try:
    co2_level = co2.get_co2()
except:
    co2_level = -1
    print("Error getting co2")
try:
    degrees = temperature.get_temperature()
except:
    degrees = -1
    print("Error getting temperature")
try:
    percentage = humidity.get_humidity()
except:
    percentage = -1
    print("Error getting humidity")
try:
    noise = audio.listen()
except:
    noise = -1
    print("Error getting db(A)")

# Debug print.
print("Mean lux : ", lux," lux" )
print("Co2 level : ", co2_level," ppm")
print("Mean degrees : ", degrees," °C")
print("Mean Humidity percentage : ", percentage, "%")
print("Mean Decibels : ", noise,"dB(A)")
```

```

# Preparing the payload, i.e. the string that will be sent via HTTPRequest to the API.
# The current date of the IEQ controller is defined.
now = datetime.now()
# The date is formatted in the same format as the SQL database.
now = now.strftime("%Y-%m-%d %H:%M:%S")
# The payload is assembled as an array: each "key" is associated with a value
# e.g. : {"key": value, "key": value, etc.}
# The username and password to access the service are entered (instead of XXX).
payload = {"username" : "XXX", "password": XXX, "timestamp": str(now), "luminosity": str(lux), "co2": str(co2_level), "temperature": str(degrees), "humidity": str(percentage), "noise": str(noise)}

# Checking for internet connection. If HTTPRequest.internet_on is True it send a store-request, else it write into file.
if(HTTPRequest.internet_on()):

    # Getting data from weather station's API (https://openweathermap.org/).
    text = w.get_temperature(now)
    # Sending weather data to IEQ Server Side API
    HTTPRequest.send_weather(text)

    # Recovering data not sent due to the lack of an internet connection.
    # Check for old backup:
    # If there are rows in the backup file, it sends them all and then sends the current detection;
    # else, it sends the current detection directly.
    while True:
        chunk = f.getAndDelete()
        if chunk == '':
            break
        else:
            try:
                # Sending IEQ Logger backup data (if exists) to the IEQ Server Side API.
```

```
                res = HTTPRequest.send_json(chunk)
                print(res)
            except:
                print("Connection Error")
    # Sending IEQ Logger data to the IEQ Server Side API.
    response = HTTPRequest.send_json(payload)

else:
    # If there is no internet connection, the payload is appended into the file.
    try:
        f.append(payload)
        print("Connection errors, saving on file")
    except:
        print("Failed writing on file")
```

B.1.2 *request.py*

```
#pip3 install datetime requests
import requests
from urllib.request import urlopen
from datetime import datetime

# This class is designed to send the Weather readings and data to the API.
# - send inputs the values separately (in the form of parameters) and composes the payload.
# - send_json takes the payload as direct input and sends it to the API
# - internet_on checks if the website www.riffelli.it is reachable
# - send_weather sends the data acquired from the weather API into the database
class HTTPRequest():

    def send(self, luminosity, co2, temperature, humidity, noise):
        now = datetime.now()
        now = now.strftime("%d/%m/%Y %H:%M:%S")
        response = []
```

```
        payload = {"username" : 'XXX', "password": 'XXX',
            ↪  "timestamp": now, "luminosity": luminosity
            ↪ , "co2": co2, "temperature": temperature, "
            ↪ humidity": humidity, "noise": noise}
        try:
            r = requests.post("http://www.riffelli.it/IEQ
                ↪ /API/add_module.php", data=payload)
            response[0] = "ok"
            response[1] = r
        except:
            response[0] = "Errore"
            response[1] = payload
        return response

    def send_json(self,text):
        r = requests.post("http://www.riffelli.it/IEQ/API
            ↪ /add_module.php", data=text)
        print(r.status_code)
        return r.status_code

    def internet_on(self):
        try:
            response = urlopen('http://www.riffelli.it/',
                ↪  timeout=10)
            return True
        except:
            return False

    def send_weather(self,text):
        r = requests.post("http://www.riffelli.it/IEQ/API
            ↪ /add_weather.php", data=text)
        return r.status_code
```

B.2 Sensor libraries

B.2.1 *co2_level.py*

```
import serial
import time

```

```
# This class acquires the ppm value of CO2 through the K30 sensor.
# The UART protocol is used via the serial port /dev/serial0 with baud rate = 9600.

class Co2:

    def __init__(self):
        self.ser = serial.Serial("/dev/serial0",baudrate=9600,timeout = .5)

    def get_co2(self):
        vett = [0] * 10
        for tmp in range(3):
            self.ser.flushInput()
            self.ser.write(serial.to_bytes([0xFE,0x44,
            0x00,0x08,0x02,0x9F,0x25]))
            time.sleep(.5)
            resp = self.ser.read(7)
            len(resp)
            high = resp[3]
            low = resp[4]
            co2 = (high*256) + low
            vett[tmp] = co2
        return co2
```

B.2.2 *temperature.py*

```
#!/usr/bin/env python3

import time
from bme280 import BME280

try:
    from smbus2 import SMBus
except ImportError:
    from smbus import SMBus

# This class acquires the °C value of temperature through the BME280 sensor.
```

```

class Temperature:
    # Defining protocols.
    BUS = SMBus(1)
    BME = BME280(i2c_dev=BUS)

    # Get CPU temperature and compensed for it.
    def get_cpu_temperature(self):
        with open("/sys/class/thermal/thermal_zone0/temp", "r") as f:
            cpu_temp = f.read()
            cpu_temp = int(cpu_temp) / 1000.0
        return cpu_temp

    # It returns temperature (with compensation).
    def get_temperature_compensated(self):
        # Temperature compensation value:
        # change this parameter to adjust the temperature.
        # Default value: 2.25
        factor = 2.25
        cpu_temps = [self.get_cpu_temperature()] * 5
        # It runs 3 cycles and averages the values it obtains during the 3 iterations.
        for x in range(3):
            cpu_temp = self.get_cpu_temperature()
            # Smooth out with some averaging to decrease jitter.
            cpu_temps = cpu_temps[1:] + [cpu_temp]
            avg_cpu_temp = sum(cpu_temps) / float(len(cpu_temps))
            raw_temp = self.BME.get_temperature()
            comp_temp = raw_temp - ((avg_cpu_temp - raw_temp) / factor)
            time.sleep(1.0)
        return comp_temp

    # It returns temperature (without compensation).
    def get_temperature(self):
        tmp = 0
        vett = [0] * 10
        #value for the calibration.
        calibration = 1
```

```
        for x in range(3):
            temperature = self.BME.get_temperature()
            vett[tmp] = temperature
            tmp = tmp +1
            time.sleep(1.0)
        return temperature - calibration

    def mean(self, vett, tmp):
        # The average (mean) is performed.
        sum = 0
        for x in vett:
            sum = sum + x
        mean = sum / tmp
        return mean
```

B.2.3 *humidity.py*

```
#!/usr/bin/env python3

import time
from bme280 import BME280

try:
    from smbus2 import SMBus
except ImportError:
    from smbus import SMBus

# This class acquires the % value of humidity through the
    ↪ BME280 sensor.

class Humidity:
    # Defining protocols.
    BUS = SMBus(1)
    BME = BME280(i2c_dev=BUS)

    # The average of 3 measurements is performed.
    def get_humidity(self):
        tmp = 0
        vett = [0] * 10
        try:
```

```
            for x in range(3):
                humidity = self.BME.get_humidity()
                vett[tmp] = humidity
                tmp = tmp +1
                time.sleep(1.0)
            return self.mean(vett, tmp)
        except:
            print("Error to get humidity")

    def mean(self, vett, tmp):
        # The average (mean) is performed.
        sum = 0
        for x in vett:
            sum = sum + x
        mean = sum / tmp
        return mean
```

B.2.4 *luminosity.py*

```
#!/usr/bin/env python3
'''
light.py - It returns the illuminance value.
'''
import time
try:
    # Transitional fix for breaking change in LTR559.
    from ltr559 import LTR559
    ltr559 = LTR559()
except ImportError:
    import ltr559

class Light:
    # The sensor must initially go to a steady-state.
    # For this reason, cycles are performed at 1-second
       ↪ intervals
    # and the third reading is taken as reliable.
    def get_lux(self):
        tmp = 0
        vett = [0] * 10
        try:
```

```
            for x in range(3):
                lux = ltr559.get_lux()
                vett[tmp] = lux
                tmp = tmp +1
                #The lux value is read if, and only if, more than one cycle is performed within one second.
                time.sleep(1.0)
        except:
          print("Error to get Luminosity")
        return lux
```

B.2.5 *noise_level.py*

```
import os
import sounddevice
import numpy as np
from scipy.io.wavfile import write
from scipy.io.wavfile import read
import scipy
from scipy.signal import bilinear
import errno
import pyaudio
import time

class Audio_processing:
     # First method to be executed, it define the adopted standard.
    def __init__(self):
        # 16 bit.
        self.FORMAT = pyaudio.paInt16
        # 1 means mono. If stereo, put 2.
        self.CHANNEL = 1
        # Use what you need.
        self.CHUNKS = [4096, 9600]
        # see self.CHUNKS.
        self.CHUNK = 9600
        # device index found by p.get_device_info_by_index(ii).
        self.index = 3
```

```
        '''
        Different mics have different rates.
        For example, Logitech HD 720p has rate 48000Hz
        '''
        self.RATES = [44300, 48000]
        self.RATE = self.RATES[1]

        '''
        Listen to mic
        '''
        self.pa = pyaudio.PyAudio()
        # Opening audio streaming.
        self.stream = self.pa.open(format = self.FORMAT,
                        channels = self.CHANNEL,
                        input_device_index = self.index,
                        rate = self.RATE,
                        input = True,
                        frames_per_buffer = self.CHUNK)

    # The method iterates for 3 seconds and returns the dBA average of this range.
    def listen(self):
        t_end = time.time() + 3
        tmp = 0
        vett_dba = [0] * 100
        while time.time() < t_end:

            # Recording data self.CHUNK.
            self.stream.start_stream()
            data = np.fromstring(self.stream.read(self.CHUNK,
            exception_on_overflow = False),dtype=np.int16)
            stream_data = data
            self.stream.stop_stream()

            # Mic sensitivity correction and bit conversion.
            # Mic sensitivity in dBV + any gain.
            mic_sens_dBV = 33.0
            # Calculating mic sensitivity conversion factor.
```

```
            mic_sens_corr = np.power(10.0,mic_sens_dBV
                ↪ /20.0)

            # USB=5V, so 15 bits are used (the 16th for
                ↪ negatives)
            # and the manufacturer microphone sensitivity
                ↪  corrections.
            data = ((data/np.power(2.0,15))*5.25)/
            (mic_sens_corr)

            # Computing FFT parameters.
            # Frequency vector based on window size and
                ↪ sample rate.
            f_vec = self.RATE*np.arange(self.CHUNK/2)/
            self.CHUNK
            # Low frequency response of the mic (mine in
                ↪ this case is 100 Hz)
            mic_low_freq = 100
            low_freq_loc = np.argmin(np.abs(f_vec-
                ↪ mic_low_freq))
            fft_data = (np.abs(np.fft.fft(data))
            [0:int(np.floor(self.CHUNK/2))])/self.CHUNK
            fft_data[1:] = 2*fft_data[1:]

            max_loc = np.argmax(fft_data[low_freq_loc:])+
            low_freq_loc

            # A-weighting function and application.
            f_vec = f_vec[1:]
            R_a = ((12194.0**2)*np.power(f_vec,4))/
            (((np.power(f_vec,2)+20.6**2)*np.sqrt
            ((np.power(f_vec,2)+107.7**2)*
            (np.power(f_vec,2)+737.9**2))*
            (np.power(f_vec,2)+12194.0**2)))
            a_weight_data_f = (20*np.log10(R_a)+2.0)+
            (20*np.log10(fft_data[1:]/0.00002))
            a_weight_sum = np.sum(np.power(10,
                ↪ a_weight_data_f/20)*
            0.00002)

            dba = 20*np.log10(a_weight_sum/0.00002)
            vett_dba[tmp] = dba
            tmp = tmp +1
```

```

        # The average (mean) is performed.
        sum = 0
        for x in vett_dba:
            sum = sum + x
        mean = sum / tmp
        return mean
```

B.3 Other libraries

B.3.1 *get_weather.py*

```
import requests
import json

#https://openweathermap.org/

class Weather:
    # First method to be executed: it define the adopted standard and the API key.
    def __init__(self,city,nation):
        self.city = city
        self.nation = nation
        self.unit = "metric"
        self.key = "1195bd976923aeedb30a29e697e82b47"
        self.weather_url = "http://api.openweathermap.org/data/2.5/weather?q="+self.city+","+self.nation+"&units="+self.unit+"&APPID="+self.key

    def get_temperature(self,timestamp):
        response = requests.get(self.weather_url)
        json_data = json.loads(response.text)
        res_json = {"timestamp":str(timestamp),
        "temp":str(json_data['main']['temp']),
        "humidity":str(json_data['main']['humidity']),
        "pressure":str(json_data['main']['pressure'])}
        return res_json
```

B.3.2 *file.py*

```
'''
Library for writing on backup file
'''

# Definitions for the backup file.
class File:

    def __init__(self,path):
        self.filepath =  path

    def append(self,text):
        text = text + "\n"
        file = open(self.filepath,"a")
        file.write(text)
        file.close()

    def write(self,text):
        text = text + "\n"
        file = open(self.filepath,"w")
        file.write(text)
        file.close()

    def getAndDelete(self):
        file = open(self.filepath, "r")
        row = file.readline()
        self.deleteRow(file,row)
        file.close()
        return row

    def deleteRow(self,lines,text):
        with open(self.filepath, "w") as f:
            for line in lines:
                if line.strip("\n") != text:
                    f.write(line)
        f.close()
```

Appendix C

Website components implementation

C.1 Main webpage - *index.html*

```
<!DOCTYPE html>
<html lang="it">
<head>
  <title>IEQ</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
  <style>
      .format{
          border: 3px solid white;
      }

      .alert{
          margin-bottom: 0px;
      }

```

```
      .well{
          background-color: rgba(255,255,255,0.3);
      }
      body{
          background: url("img/Sfondo.png");
          background-repeat: no-repeat;
          background-size: cover;
          background-position: center;
          background-attachment: fixed;
      }
      .opacity{
          background-color: rgba(255,255,255,0.5);
      }
    /* Set black background color, white text and some padding */
    footer {
      background-color: #555;
      color: white;
      padding: 15px;
    }
  </style>
</head>
<body onLoad="get_list();setInterval(function(){ get_list();}, 5000)">

<nav class="navbar navbar-inverse">
  <div class="container-fluid">
    <div class="navbar-header">
      <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#myNavbar">
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="http://www.riffelli.it/IEQ/">Wireless IEQ Logger</a>
    </div>
    <div class="collapse navbar-collapse" id="myNavbar">
      <ul class="nav navbar-nav">
        <li class="active"><a href="http://www.riffelli.it/IEQ/">Home</a></li>
        <li><a href="http://www.riffelli.it/IEQ/questionnaire/">Survey</a></li>
```

```
          <li><a href="http://www.riffelli.it/IEQ/prediction/">Virtual Sensor</a></li>
      </ul>
    </div>
  </div>
</nav>

<div class="container text-center">
  <div class="row ">
    <div class="col-sm-3 well ">
      <div class="well">
        <p><a href="http://www.riffelli.it/IEQ/IEQLoggerImages.html">Wireless IEQ Logger</a></p>
        <img src="/IEQ/img/profile1.png" class="img" height="100vh"  alt="Profile">
      </div>
      <div class="well">
        <p><strong>Fields</strong></p>
        <p>
        <span class="label label-success">Comfort</span>
        <span class="label label-success">Thermal Comfort</span>
        <span class="label label-success">Indoor Air Quality (IAQ)</span>
        <span class="label label-success">Acoustic Comfort</span>
        <span class="label label-success">Visual Comfort</span>
        <span class="label label-success">Indoor Global Comfort Index (IGCI)</span>
        <span class="label label-success">Indoor Environmental Quality (IEQ)</span>
        <span class="label label-success">Smart Buildings</span>
        <span class="label label-success">Green Buildings</span>
        <span class="label label-success">Artificial Intelligence (AI)</span>
        <span class="label label-success">Raspberry Pi</span>
        <span class="label label-success">Internet of Things (IoT)</span>
```

```
        <span class="label label-success">Sensors</span>
        <span class="label label-success">Matlab</span>
        <span class="label label-success">SPSS</span>
        <span class="label label-success">Python</span>
        </p>
      </div>
      <div class="alert alert-success fade in">
        <p><strong>Welcome!</strong></p>
       This is my website for Wireless IEQ (Indoor Environmental Quality) Logger Project <br>
      </div>
      <p style="color:steelblue"><b>Contacts:</b></p>
        <p><a href="mailto:riffelli@gmail.com">Stefano Riffelli</a></p>
              <p><a href="https://www.linkedin.com/in/stefano-riffelli-964410b5/">LinkedIn page</a></p>
      <p><a href="https://www.uniurb.it/">PhD Candidate at University of Urbino "Carlo Bo"</a></p>

    </div>
    <div class="col-sm-7">

          <div class="row">
            <div class="col-sm-12">
              <div class="panel opacity panel-default text-left">
                   <div  class="alert alert-success fade in">
                    <p><strong>LIST OF INTERNAL AND EXTERNAL MEASUREMENTS:</strong></p>
                </div>
              </div>
            </div>
          </div>

          <div id="list">
              <!--White space for measurements-->
          </div>
      </div>
  </div>
</div>
```

```

<footer class="container-fluid text-center">
  <p>IEQ</p>
</footer>
        <script>
            function get_list(){
                 $.ajax({
                    url : "http://www.riffelli.it/IEQ/API
                       ↪ /get_list.php?username=
                       ↪ raspberrypiIEQ&password
                       ↪ =123456789&page=0",
                    success : function (list) {
                        document.getElementById("list").
                           ↪ innerHTML = list;
                    },
                    error : function (request,status,
                       ↪ errors) {
                        console.log("Error: "+status);
                    }
                });
            }
        </script>
</body>
</html>
```

C.2 API (Application Programming Interface)

C.2.1 *add_ module.php*

```
<?php

/*
    add_module.php by Stefano Riffelli
    This file is used to add the measurements from IEQ
       ↪ Logger into database.
*/

require 'lib/dbcon.php';

//user login credentials
```

```
$internal_username = "XXX";
$internal_password = "XXX";

// getting credentials from request
$username = $_REQUEST['username'];
$password = $_REQUEST['password'];

// chencking credentials
if($internal_username == $username && $internal_password == $password){

    // getting param from http request
    $timestamp = $_REQUEST['timestamp'];
    $luminosity = $_REQUEST['luminosity'];
    $co2 = $_REQUEST['co2'];
    $temperature = $_REQUEST['temperature'];
    $humidity = $_REQUEST['humidity'];
    $noise = $_REQUEST['noise'];

    // assembling query
    $sql = "INSERT INTO `Sql315130_3`.`measurements` (`id`, `timestamp`, `luminosity`, `co2`, `temperature`, `humidity`, `noise`) VALUES (NULL,'";
    $sql .= $timestamp . "','";
    $sql .= $luminosity . "','";
    $sql .= $co2 . "','";
    $sql .= $temperature . "','";
    $sql .= $humidity . "','";
    $sql .= $noise . "')";

    // sending query and errors checking
    if($conn->query($sql)){
        header("HTTP/1.0 200 OK");
        echo("\nCorrectly entered value");
    }else{
        header("HTTP/1.0 500 Internal Server Error");
        die("Query error" . $conn->error);
    }
}else{
    header("HTTP/1.0 500 Internal Server Error");
}
?>
```

C.2.2 *add_prediction.php*

```
<?php
/*
    add_prediction.php by Stefano Riffelli
    This API inserts the P-IGCI calculated by the Virtual
      ↪ Sensor into the database.
*/

require 'lib/dbcon.php';

$PGCI = $_REQUEST['PGCI'];
$MODEL = $_REQUEST['MODEL'];

$sql = "INSERT INTO `predict` (`id`, `pgci`, `method`)
  ↪ VALUES (NULL, '".$PGCI."', '".$MODEL."')";

if($conn->query($sql)){
    #header("HTTP/1.0 404 Not Found");
    header("HTTP/1.0 200 OK");
    echo("\nDone");
}else{
    header("HTTP/1.0 500 record error");
    die("Error --> " . $conn->error);
}

?>
```

C.2.3 *add_weather.php*

```
<?php

/*
    add_weather.php by Stefano Riffelli
    This file is used to add the weather information
      ↪ sended by the IEQ Logger.
*/

require 'lib/dbcon.php';
```

```

//user login credentials
$internal_username = "XXX";
$internal_password = "XXX";

// getting param from http request
$timestamp = $_REQUEST[’timestamp’];
$temperature = $_REQUEST[’temp’];
$humidity = $_REQUEST[’humidity’];
$pressure = $_REQUEST[’pressure’];

// assembling query
$sql = "INSERT INTO ‘Sql315130_3‘.‘weather‘ (‘id‘, ‘timestamp‘, ‘temperature‘, ‘humidity‘,‘pressure‘) VALUES (NULL, ’".$timestamp."’, ’".$temperature."’, ’".$humidity."’,’".$pressure."’);";

// sending query and errors checking
if($conn->query($sql)){
    header("HTTP/1.0 200 OK");
    echo("\nCorrectly entered value");
}else{
    header("HTTP/1.0 404 Not Found");
    die("Query error" . $conn->error);
}
?>
```

C.2.4 *analysis.php*

```
<?php
/*
    analysis.php by Stefano Riffelli
    This file is used by P-IGCI Virtual Sensor to retrive data from database
*/

require ’lib/dbcon.php’;

//credentials for user authentication
$token1 = "XXX";
```

```

//credentials sent for client authentication
$userToken = $_REQUEST['token'];

//action you want to perform
$action = $_REQUEST['action'];

//output variable
$str ="";

// checking token
if($token1 == $userToken){
    if ($_SERVER['REQUEST_METHOD'] === 'GET') {
        if($action == "getQuestionnaire"){
            $sql = "SELECT * FROM `questionnaire` WHERE";
            if(isset($_REQUEST["startdate"]) and isset($_REQUEST["enddate"])){
                $sql .= "`timestamp` BETWEEN str_to_date('".$_REQUEST["startdate"]."','%Y-%m-%d %H:%i:%s') AND str_to_date('".$_REQUEST["enddate"]."','%Y-%m-%d %H:%i')";
            }else{
                $sql .= " 1 ";
            }
            $sql .= " ORDER BY `id` DESC";
            if($res = $conn->query($sql)){
                    $str .= ('{"questionnaire":[');
                    $i = 0;
                    while($rows = $res -> fetch_assoc())     {
                        if ($i > 0)
                            $str .= (",");
                        $j = 0;
                        $str .=  "{";
                        foreach($rows as $row => $row_value) {
                            if ($j > 0)
                                $str .= (",");
                            $str .=  '"'.$row.'":"'.$row_value.'"';
                            $j++;
                        }
```

```
                        $str .= "}";
                        $i++;
                    }
                    $str .= ("]}");
                    echo json_encode($str);
                }else{
                    echo json_encode("{}");
                }
        }
        if($action == "getQuestionnaireMeasurements"){

            $data1 = $_REQUEST['data1'];
            $data2 = $_REQUEST['data2'];
            $sql = "SELECT * FROM `measurement` WHERE `timestamp` BETWEEN str_to_date('".$data1."','%Y-%m-%d %H:%i:%s') AND str_to_date('".$data2."','%Y-%m-%d %H:%i')";
            if($res = $conn->query($sql)){
                    $str .= ('{"measurement":[');
                    $i = 0;
                    while($rows = $res -> fetch_assoc())     {
                        if ($i > 0)
                            $str .= (",");
                        $j = 0;
                        $str .= "{";
                        foreach($rows as $row => $row_value) {
                            if ($j > 0)
                                $str .= (",");
                            $str .= '"'.$row.'":"'.$row_value.'"';
                            $j++;
                        }
                        $str .= "}";
                        $i++;
                    }
                    $str .= ("]}");
                    echo json_encode($str);
                }
        }
    }else{
```

```
        header("HTTP/1.0 404 Not Found");
        die("Query error: " . $conn->error);
    }
}else{
    header("HTTP/1.0 404 Not Found");
}

?>
```

C.2.5 *dbcon.php*

```
<?php

/*
    dbcon.php by Stefano Riffelli
    This file provides the connection with the database
*/

// defining database informations
$servername = "XXX";
$username = "XXX";
$password = "XXX";
$dbname = "XXX";

// creating connections
$conn = new mysqli($servername, $username, $password,
   ↪ $dbname);
// Checking connection status
if ($conn->connect_error) {
    die("Connection error: " . $conn->connect_error);
}

?>
```

C.2.6 *get_last_rilevation.php*

```
<?php
```

```php

/*
    get_last_rilevation.php by Stefano Riffelli
    This file send to the client the last measurement wrapped in a HTML tag
*/

require 'lib/dbcon.php';

// assembling query
$sql = "SELECT * FROM measurements ORDER BY id DESC LIMIT 0, 1";

// checking for query errors
if($target = $conn->query($sql)){

        // assembling the html element and the
        $list = "";
        while($row = $target -> fetch_assoc()) {
            $list .= "<div class=\"row\">
                    <div class=\"col-sm-6\">
                        <div class=\"well\">
                        luminosity: ".number_format($row['luminosity']). " lux<br> co2: ".$row['co2']." ppm<br>temperature: ".number_format($row['temperature'],1)." °C<br> humidity: ".number_format($row['humidity'])."%<br> noise: ".number_format($row['noise'])." db(A)
                      </div>
                    </div>
                  </div>";
        }
        echo($list);
}else{
    header("HTTP/1.0 404 Not Found");
    die("Query error" . $conn->error);
}
?>
```

C.2.7 *get_last_rilevation_json.php*

```
<?php
/*
    get_last_rilevation_json.php by Stefano Riffelli
    This API is required for the Virtual Sensor.
    This API extracts the last record collected in the
       ↪ database corresponding to the measured physical
       ↪  parameters.
*/

require 'lib/dbcon.php';

$sql = "SELECT * FROM rilevazioni ORDER BY id DESC LIMIT
   ↪ 0, 1";

if($target = $conn->query($sql)){
        while($row = $target -> fetch_assoc())  {
            $json = '{"param":['.$row['luminosity'] . ','
               ↪  . $row['co2'] . ",".$row['temperature'
               ↪ ] . ',' . $row['humidity']. ',' . $row[
               ↪ 'noise'] .']}';
        }
        echo($json);
}else{
    header("HTTP/1.0 404 Not Found");
    die("Errore durante la query: " . $conn->error);
}

?>
```

C.2.8 *get_list*

```
<?php

/*
    get_list.php by Stefano Riffelli
    This file send to the client the last 10 measurement
       ↪ and its weather informations wrapped in an HTML
```

```
        tag
*/

require 'lib/dbcon.php';

//user login credentials
$internal_username = "XXX";
$internal_password = "XXX";

// getting credentials from request
$username = $_REQUEST['username'];
$password = $_REQUEST['password'];

// chencking credentials
if($internal_username == $username && $internal_password == $password){

    // assembling query
    $sql = "SELECT measurements.id,measurements.timestamp,measurements.luminosity,measurements.co2, measurements.temperature,measurements.humidity, measurements.noise , weather.temperature as weatherTemperature, weather.humidity as weatherHumidity,weather.pressure as weatherPressure FROM `measurements` INNER JOIN weather ON measurements.timestamp = weather.timestamp WHERE measurements.id> (SELECT `id` FROM `measurements` WHERE 1 ORDER BY `id` DESC LIMIT 1)-10 ORDER BY measurements.timestamp DESC";

    // checking for query errors
    if($target = $conn->query($sql)){

            // adding html tag to measurements informations
            $list = "";
            while($row = $target -> fetch_assoc()) {
                $list .= "<div class=\"row\" style='border: solid 1px;border-color: gray;border-radius: 30px;'>
                        <div class=\"col-sm-6\">
                            <h3>IEQ Control Unit</h3>
```

```
                                  <div class=\"well\">

                                  TimeStamp: ".$row['timestamp'
                                     ↪ ]."<br>Luminosity: ".
                                     ↪ number_format($row['
                                     ↪ luminosity']). " lux<br
                                     ↪ >CO2: ".$row['co2']."
                                     ↪ ppm<br>Temperature: ".
                                     ↪ number_format($row['
                                     ↪ temperature'],1)." °C<
                                     ↪ br>Humidity: ".
                                     ↪ number_format($row['
                                     ↪ humidity'])."%<br>Noise
                                     ↪  level: ".number_format
                                     ↪ ($row['noise'])." dB(A)
                              </div>
                          </div>

                          <div class=\"col-sm-6\">
                              <h3>Weather</h3>
                              <div class=\"well\">

                                  Temperature: ".number_format(
                                     ↪ $row['
                                     ↪ weatherTemperature'],1)
                                     ↪ ." °C<br>
                                  Humidity: ".number_format(
                                     ↪ $row['weatherHumidity'
                                     ↪ ])."%<br>
                                  Pressure: ".number_format(
                                     ↪ $row['weatherPressure'
                                     ↪ ])." mPa<br>
                                     ↪
                              </div>
                          </div>
                      </div>";
            }
            echo($list);

    }else{
        header("HTTP/1.0 404 Not Found");
        die("Query Error: " . $conn->error);
    }
```

```
}else{
    header("HTTP/1.0 404 Not Found");
}
?>
```

C.2.9 *insert_ questionnaire.php*

```
<?php

/*
    insert_questionnaire.php by Stefano Riffelli
    This file enables the questionnaire to be saved in
       ↪ the database. This file is called up 3 times
       ↪ per questionnaire. In this way, even partially
       ↪ completed questionnaires are saved.
*/

require 'lib/dbcon.php';

$question = $_REQUEST['question'];

//user login credentials
$internal_username = "XXX";
$internal_password = "XXX";

// getting credentials from request
$username = $_REQUEST['username'];
$password = $_REQUEST['password'];

// checking which question you arrived at
if($question == 1){
    $timestamp = $_REQUEST["timestamp"];
    $comfortGlobal = $_REQUEST['comfortGlobal'];
    $gender = $_REQUEST["gender"];
    $age = $_REQUEST["age"];

    //assembling query
    $sql = "INSERT INTO `Sql315130_3`.`questionnaire` (`
       ↪ id`, `cod`, `timestamp`, `gender`, `age`, `
       ↪ comfortGlobal`, `comfortThermal`, `
```

```
        ↪ comfortAcoustic`, `comfortVisual`, `airQuality
        ↪ `, `TSV`, `airMovement`, `clothing`, `
        ↪ unwantedNoise`, `noiseConsequences`, `lighting
        ↪ `, `respirability`, `badOdours`) VALUES (NULL,
        ↪ '".$userid."','".$timestamp."', '".$gender."',
        ↪ '".$age."', '".$comfortGlobal."', NULL, NULL,
        ↪ NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
        ↪  NULL, NULL);";

    if($conn->query($sql)){
        header("HTTP/1.0 200 OK");
        echo("\nCorrectly entered value");
    }else{
        header("HTTP/1.0 500 Internal Server Error");
        die("Query error" . $conn->error);
    }

// checking which question you arrived at
}else if($question == 2){
    $comfortThermal = $_REQUEST["comfortThermal"];  //
        ↪ question comfortThermal
    $comfortAcoustic = $_REQUEST["comfortAcoustic"];
        ↪ //question comfortAcoustic
    $comfortVisual = $_REQUEST["comfortVisual"];    //
        ↪ question comfortVisual
    $airQuality = $_REQUEST["airQuality"];  //question
        ↪ airQuality

    //assembling query
    $sql = "UPDATE `questionnaire` SET `comfortThermal`='
        ↪ ".$comfortThermal."',`comfortAcoustic`='".
        ↪ $comfortAcoustic."',`comfortVisual`='".
        ↪ $comfortVisual."',`airQuality`='".$airQuality."
        ↪ ' WHERE `cod` = '".$userid."'";

    if($conn->query($sql)){
        header("HTTP/1.0 200 OK");
        echo("\nCorrectly entered value");
    }else{
        header("HTTP/1.0 404 Not Found");
        die("Query error" . $conn->error);
    }

```

```
// checking which question you arrived at
}else if($question == 3){
    $TSV = $_REQUEST["TSV"];    //question TSV
    $airMovement = $_REQUEST["airMovement"];    //
       ↪ question airMovement
    $clothing = $_REQUEST["clothing"];  //question
       ↪ clothing
    $unwantedNoise = $_REQUEST["unwantedNoise"];    //
       ↪ question unwantedNoise
    $noiseConsequences = $_REQUEST["noiseConsequences"];
       ↪    //question noiseConsequences
    $lighting = $_REQUEST["lighting"];  //question
       ↪ lighting
    $respirability = $_REQUEST["respirability"];    //
       ↪ question respirability
    $badOdours = $_REQUEST["badOdours"];    //question
       ↪ badOdours

    //assembling query
    $sql = "UPDATE `questionnaire` SET `TSV`='".$TSV."',`
       ↪ airMovement`='".$airMovement."',`clothing`='".
       ↪ $clothing."',`unwantedNoise`='".$unwantedNoise.
       ↪ "',`noiseConsequences`='".$noiseConsequences."
       ↪ ',`lighting`='".$lighting."',`respirability`='"
       ↪ .$respirability."',`badOdours`='".$badOdours."'
       ↪  WHERE `cod` ='".$userid."'";

    if($conn->query($sql)){
        header("HTTP/1.0 200 OK");
        echo("\nCorrectly entered value");
    }else{
        header("HTTP/1.0 500 Internal Server Error");
        die("Query error" . $conn->error);
    }
}
?>
```

C.3 Questionnaire - *questionnaire.html*

```
<!DOCTYPE html>
```

```
<html lang="it">
<head>
    <title>IEQ Questionnaire</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">

    <style>

        .color {
          list-style-type: none;
          margin: 0;
          padding: 0;
          overflow: hidden;
          background-color: #333;
        }

        li {
          float: left;
        }

        li a {
          display: block;
          color: gray;
          text-align: center;
          padding: 14px 16px;
          text-decoration: none;
        }

        li a:hover:not(.active) {
          color : white;
```

```
        }

        .active {
          background-color: black;
        }

        body {
            background-image: url("../img/photo
              ↪ -1497211419994-14ae40a3c7a3.jpg");
            background-repeat: no-repeat;
            background-size: cover;
            background-attachment: fixed;
        }

        .container{
            padding: 10px;
            box-shadow: 5px 5px 20px;
            border-radius: 50px;
            background-color: rgba(255,255,255,0.60);
        }

        .jumbotron{
            background-color: rgba(255,255,255,0.60);
            border-radius: 50px;
        }

        #section1{
            display: none;
        }

        #section2{
            display: none;
        }
        #section3{
            display: none;
        }
        #section4{
            display: none;
        }

        .section{
            text-align: center;
            border-radius: 20px;
```

```
            padding: 10px;
            width: 100%;
        }

    </style>

</head>
<body>
    <ul class="color">
      <li><a href="http://www.riffelli.it/IEQ/">Wireless IEQ Logger</a></li>
      <li><a href="http://www.riffelli.it/IEQ/">Home</a></li>
      <li><a class="active" href="#">Survey</a></li>
      <li><a href="http://www.riffelli.it/IEQ/prediction/">Virtual Sensor</a></li>
    </ul>

<div class="jumbotron text-center">
    <h1>IEQ Questionnaire</h1>
  <div>Questionnaire for the subjective assessment of combined<br></div>
    <div>Thermal, Acoustic, Visual and IAQ Comfort</div><br>
</div>

<div class="container">

    <div id="section1" class="section">
        <!--Section body-->
        <h2>Basic Information</h2><hr>

        <!--Question 1-->

        <div class="input-group mb-3">
          <div class="input-group-prepend">
            <label class="input-group-text" for="sz11">Gender:</label>
          </div>
          <select class="custom-select" id="sz11">
            <option value="NAN" disabled selected>Please select...</option>
            <option value="male">Male</option>
```

```
            <option value="female">Female</option>
            <option value="not-declared">Not declared</
               ↪ option>
          </select>
        </div>
        <!--Question 2-->
        <div class="input-group mb-3">
          <div class="input-group-prepend">
            <label class="input-group-text" for="sz12">
               ↪ Age:</label>
          </div>
          <select class="custom-select" id="sz12" >
            <option value="NAN" disabled selected>Please
               ↪ select...</option>
            <option value="19">19</option>
            <option value="20">20</option>
            <option value="21">21</option>
            <option value="22">22</option>
            <option value="23">23</option>
            <option value="24">24</option>
            <option value="25">25</option>
            <option value="26">26</option>
            <option value="27">27</option>
            <option value="28">28</option>
            <option value="29">29</option>
            <option value="30">30+</option>
          </select>
        </div>
        <!--Question 3-->
        <p>How do you rate <strong>global</strong>
           ↪ comfort in the classroom?</p>
        <div class="input-group mb-3">
          <select class="custom-select" id="sz13" >
            <option value="NAN" disabled selected>Please
               ↪ select...</option>
            <option value="1">1 - VERY POOR</option>
            <option value="2">2</option>
            <option value="3">3</option>
            <option value="4">4</option>
            <option value="5">5 - VERY GOOD</option>
          </select>
        </div>

```

```html
    </div>

    <div id="section2" class="section">
        <!--Section body-->
        <h2>Comfort categories in the last hour</h2><hr>

        <!--Question 1 -->
        <p>How do you rate <strong>thermal</strong> comfort in the classroom?</p>
        <div class="input-group mb-3">
          <select class="custom-select" id="sz21" >
            <option value="NAN" disabled selected>Please select...</option>
            <option value="1">1 - VERY POOR</option>
            <option value="2">2</option>
            <option value="3">3</option>
            <option value="4">4</option>
            <option value="5">5 - VERY GOOD</option>
          </select>
        </div>

        <!--Question 2 -->
        <p>How do you rate <strong>acoustic</strong> comfort in the classroom?</p>
        <div class="input-group mb-3">
          <select class="custom-select" id="sz22" >
            <option value="NAN" disabled selected>Please select...</option>
            <option value="1">1 - VERY POOR</option>
            <option value="2">2</option>
            <option value="3">3</option>
            <option value="4">4</option>
            <option value="5">5 - VERY GOOD</option>
          </select>
        </div>

        <!--Question 3 -->
        <p>How do you rate <strong>visual</strong> comfort in the classroom?</p>
        <div class="input-group mb-3">
          <select class="custom-select" id="sz23" >
            <option value="NAN" disabled selected>Please select...</option>
```

```
            <option value="1">1 - VERY POOR</option>
            <option value="2">2</option>
            <option value="3">3</option>
            <option value="4">4</option>
            <option value="5">5 - VERY GOOD</option>
          </select>
        </div>

        <!--Question 4 -->
        <p>How do you rate the <strong>air quality</strong> in the classroom?</p>
        <div class="input-group mb-3">
          <select class="custom-select" id="sz24" >
            <option value="NAN" disabled selected>Please select...</option>
            <option value="1">1 - VERY POOR</option>
            <option value="2">2</option>
            <option value="3">3</option>
            <option value="4">4</option>
            <option value="5">5 - VERY GOOD</option>
          </select>
        </div>


    </div>

    <div id="section3" class="section">
        <!--Section body-->
        <h1>More details</h1>
        <p>If you give us more details, you help us to improve the environment quality.</p>
        <p>At the end of the questionnaire we will show you the data collected by the <strong>IEQ logger</strong>.</p>

        <br>
        <h2>Thermal Comfort in the last hour</h2><hr>

        <!--Question 1 -->
        <p class="text-center">Thermal Sensation Vote:</p>
        <div class="input-group mb-3">
            <select class="custom-select" id="sz31">
```

```
        <option value="NAN" disabled selected>Please select...</option>
        <option value="-3">Cold</option>
        <option value="-2">Cool</option>
        <option value="-1">Sufficiently cool</option>
        <option value="0">Neutral</option>
        <option value="+1">Sufficiently warm</option>
        <option value="+2">Warm</option>
        <option value="+3">Hot</option>
    </select>
</div>

<!--Question 2 -->
<p class="text-center">Air movement is:</p>
<div class="input-group mb-3">
  <select class="custom-select" id="sz32">
    <option value="NAN" disabled selected>Please select...</option>
    <option value="-3">Too ventilated</option>
    <option value="-2">Perfect</option>
    <option value="-1">Not enough ventilated</option>
  </select>
</div>

<!--Question 3-->
<p class="text-center">You are currently wearing:</p>
<div class="input-group mb-3">
  <div class="input-group-prepend">
  </div>
  <select class="custom-select" id="sz33">
    <option value="NAN" disabled selected>Please select...</option>
    <option value="summer-clothing">Summer clothing</option>
    <option value="standard-clothing">Standard clothing</option>
    <option value="winter-clothing">Winter clothing</option>
</select>
```

```
        </div>

        <h2>Acoustic Comfort in the last hour</h2><hr>

        <!--Question 1 -->
        <h4 class="text-center">Noise</h4>
        <p class="text-center">You heard unwanted noises in the classroom for a:</p>

        <div class="input-group mb-3">
          <select class="custom-select" id="sz34" >
            <option  value="NAN" disabled selected>Please select...</option>
            <option value="1">1 - Long time</option>
            <option value="2">2</option>
            <option value="3">3</option>
            <option value="4">4</option>
            <option value="5">5 - Short time / not at all </option>
          </select>
        </div>

        <!--Question 2 -->
        <h4 class="text-center">Noise consequences</h4>
        <p>Does the noise allow you to understand what the teacher is saying?</p>

        <div class="input-group mb-3">
          <select class="custom-select" id="sz35" >
            <option value="NAN" disabled selected>Please select...</option>
            <option value="yes">Yes</option>
            <option value="almost-always">Almost always</option>
            <option value="almost-never">Almost never</option>
            <option value="no">No</option>
          </select>
        </div>
        <br>

        <h2>Visual comfort in the last hour</h2><hr>

```

```html
<!--Question 1 -->
<p>The lighting in the classroom is:</p>

<div class="input-group mb-3">
  <select class="custom-select" id="sz36" >
    <option value="NAN" disabled selected>Please select...</option>
    <option value="insufficient">Insufficient</option>
    <option value="appropriate">Appropriate</option>
    <option value="excessive">Excessive</option>
  </select>
</div>

<br>
<h2>Indoor Air Quality in the last hour</h2><hr>

<!--Question 1 -->
<h4 class="text-center">Respirability</h4>
<p>How would you rate the air quality at the end of last hour?</p>

<div class="input-group mb-3">
  <select class="custom-select" id="sz37" >
    <option value="NAN" disabled selected>Please select...</option>
    <option value="1">1 - Poor</option>
    <option value="2">2</option>
    <option value="3">3</option>
    <option value="4">4</option>
    <option value="5">5 - Suitable</option>
  </select>
</div>

<!--Question 2 -->
<h4 class="text-center">Bad odours</h4>
<p>Did you perceive odours from furniture, cleaning products, glues, adhesives, solvents, paints, cigarette smoke, printers and photocopiers, in the classroom?</p>
<div class="input-group mb-3">
  <select class="custom-select" id="sz38" >
```

```
            <option value="NAN" disabled selected>Please select...</option>
            <option value="1">1 - Never</option>
            <option value="2">2</option>
            <option value="3">3</option>
            <option value="4">4</option>
            <option value="5">5 - Often / Intensely</option>
          </select>
        </div>
    </div>

    <div id="section4">
        <h2 class="text-center">Thank you for participating in the questionnaire!</h2>
        <p>The latest IEQ (Indoor Environmental Quality) unit readings are :</p>
        <div id="truereading"></div>

    </div>


    <ul class="pagination justify-content-center">
        <li class="page-item"><a class="page-link" id="next" href="javascript:void(0);">Next</a></li>
    </ul>


</div>
    <script>


        $(document).ready(function() {

            var now = getFormattedDate();
            var userCod = generateID();
            var main_page = 1;  //number of the main page
            var max_page = 4; //change this number to add nother div to the pagination
            setMainPage(main_page,max_page);
            var num = main_page;
```

```

            function generateID () {
              // Math.random should be unique because of
                 ↪ its seeding algorithm.
              // Convert it to base 36 (numbers + letters
                 ↪ ), and grab the first 9 characters
              // after the decimal.
              return '_' + Math.random().toString(36).
                 ↪ substr(2, 9);
            };

            function getFormattedDate() {
                var date = new Date();
                var month = date.getMonth() + 1;
                var day = date.getDate();
                var hour = date.getHours();
                var min = date.getMinutes();
                var sec = date.getSeconds();

                month = (month < 10 ? "0" : "") + month;
                day = (day < 10 ? "0" : "") + day;
                hour = (hour < 10 ? "0" : "") + hour;
                min = (min < 10 ? "0" : "") + min;
                sec = (sec < 10 ? "0" : "") + sec;

                var str = date.getFullYear() + "/" +
                   ↪ month + "/" + day + " " +  hour + "
                   ↪ :" + min + ":" + sec;
                return str;
            }

            function sendRequest(payload){
                $.ajax('http://www.riffelli.it/IEQ/API/
                   ↪ insert_questionario.php', {
                    type: 'POST',  // http method
                    data: payload,  // data to submit
                    success: function (data) {
                        //alert(data);
                    },
                    error: function (data) {
                            console.log("errore "+ data);
                    }
                });
```

```
                };

                $(document).on("click","#previous",function() {
                    var page = num - 1;
                    saveParam(num);
                    if(page >= 1 && page <= max_page){

                        var i = 1;
                        while(i <= max_page){
                            if(i != page){
                                $("#section"+i).css("display","none");
                            }else{
                                $("#section"+page).css("display","block");
                            }
                            i++;
                        }
                        num = num - 1;
                    }
                });

                $(document).on("click","#next",function() {
                    var page = num + 1;
                    var response = saveParam(num);
                    if(response && (page >= 1 && page <= max_page)){
                        var i = 1;
                            if(page == max_page){
                                $("#next").css("display","none");
                                $.ajax('http://www.riffelli.it/IEQ/API/get_last_rilevation.php', {
                                    success: function (data) {
                                        $("#truereading").append(data);
                                    },
                                    error: function (data) {
```

```
                                        console.log("errore "+data);
                                }
                            });
                        }
                        if(page == max_page-1){
                            $("#next").html("Submit");
                        }
                        while(i <= max_page){
                            if(i != page){
                                $("#section"+i).css("display","none");
                            }else{
                                $("#section"+page).css("display","block");
                            }
                            i++;
                        }

                        $("#section"+page).css("display","block");
                        num = num +1;
                    }
            });

            function saveParam(page){
                var payload;
                var response = true;
                if(page == 1){
                    var sz11 = $("#sz11").val();    //Gender
                    var sz12 = $("#sz12").val();    //Age
                    var sz13 = $("#sz13").val();    //Global Comfort question
                    if(sz11 == null || sz12 == null || sz13 == null){
                        response = false;
                    }else{
                        payload = {"question":"1","userid":userCod,"gender":sz11,"age":sz12,"timestamp":now,"comfortGlobal":sz13};
```

```
                    }

                }else if(page == 2){
                    var sz21 = $("#sz21").val();    // Thermal Comfort question
                    var sz22 = $("#sz22").val();    // Acoustic Comfort question
                    var sz23 = $("#sz23").val();    // Visual Comfort question
                    var sz24 = $("#sz24").val();    // Indoor Air Quality Comfort question
                    if(sz21 == null || sz22 == null || sz23 == null || sz24 == null){
                        response = false;
                    }else{
                        payload = {"question":"2","userid":userCod,"comfortThermal":parseInt(sz21),"comfortAcoustic":parseInt(sz22),"comfortVisual":parseInt(sz23),"airQuality":parseInt(sz24)};
                    }

                }else if(page == 3){
                    var sz31 = $("#sz31").val();    //TSV
                    var sz32 = $("#sz32").val();    // airMovement
                    var sz33 = $("#sz33").val();    // clothing
                    var sz34 = $("#sz34").val();    // unwantedNoise
                    var sz35 = $("#sz35").val();    // noiseConsequences
                    var sz36 = $("#sz36").val();    // lighting
                    var sz37 = $("#sz37").val();    // respirability
                    var sz38 = $("#sz38").val();    // badOdours
                    if(sz31 == null || sz32 == null || sz33 == null || sz34 == null ||
```

```
                    ↪  sz35 == null || sz36 == null
                    ↪ || sz37 == null || sz38 == null
                    ↪ ){
                    response = false;
                }else{
                     payload = {"question":"3","userid
                        ↪ ":userCod,"TSV":parseInt(
                        ↪ sz31),"airMovement":
                        ↪ parseInt(sz32),"clothing":
                        ↪ sz33,"unwantedNoise":
                        ↪ parseInt(sz34),"
                        ↪ noiseConsequences":sz35,"
                        ↪ lighting":sz36,"
                        ↪ respirability":parseInt(
                        ↪ sz37),"badOdours":parseInt(
                        ↪ sz38)};
                }
            }
            if(!response){
                alert("Enter all data correctly!");
            }else{
                sendRequest(payload);
            }
            return response;
        }

        function setMainPage(page,max_page){

            if(page >= 1 && page <= max_page){
                var i = 1;
                while(i <= max_page){
                    if(i != page){
                        $("#section"+i).css("display"
                            ↪ ,"none");
                    }else{
                        $("#section"+page).css("
                            ↪ display","block");
                    }
                    i++;
                }
            }
        }
    });
```

```

    </script>
    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script
  src="https://code.jquery.com/jquery-3.4.1.min.js"
  integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlBw8HfCJo="
  crossorigin="anonymous"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12.9/umd/popper.min.js" integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js" integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>

</body>
</html>
```

C.4 Virtual Sensor - *index.php*

```
<!doctype html>
<!--
    index.php by Stefano Riffelli
    This file represents the graphical interface of the P-IGCI Virtual Sensor.
-->
<html lang="it">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">  <meta charset="utf-8">
```

```
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>


    <title>IEQ Virtual Sensor</title>

    <style>
        body{
            background-image: url('./img/box2.jpeg');
            background-repeat: no-repeat;
            background-attachment: fixed;
            background-position: center;
            background-size: cover;
            color: black;
            }
        .left{
            text-align: left;
            float: left;
            width: 60%;
            height: 50px;
            padding: 3%;
            font-size: 22px;
            color: black;
        }
        .right{
            text-align: center;
            float: right;
            width: 40%;
            height: 50px;
            border: 3px solid black;
            font-size: 28px;
            border-radius: 30px;
            padding: 0%;
            color: black;
        }
        .gci-box{
            color: black;

```

```
        }
        .box2{
            padding: 3%;
        }

      </style>

  </head>
  <body>
      <nav class="navbar navbar-inverse">
          <div class="container-fluid">
            <div class="navbar-header">
              <button type="button" class="navbar-toggle"
                  ↪  data-toggle="collapse" data-target="
                  ↪ #myNavbar">
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
              </button>
              <a class="navbar-brand" href="http://www.
                  ↪ riffelli.it/IEQ/">Wireless IEQ Logger
                  ↪ </a>
            </div>
            <div class="collapse navbar-collapse" id="
                ↪ myNavbar">
              <ul class="nav navbar-nav">
                <li ><a href="http://www.riffelli.it/IEQ/
                    ↪ ">Home</a></li>
                <li><a href="http://www.riffelli.it/IEQ/
                    ↪ questionnaire/">Survey</a></li>
                  <li class="active"><a href="http://www.
                      ↪ riffelli.it/IEQ/prediction/">
                      ↪ Virtual Sensor</a></li>
              </ul>

            </div>
          </div>
        </nav>

      <?php
        require("../API/lib/dbcon.php");
        $sql = "SELECT `pgci`,`method` FROM `predict`
            ↪ WHERE 1 ORDER BY id DESC LIMIT 0,1";
```

```
        $res = $conn->query($sql) or die("Errore durante
           ↪ la query: " . $conn->error);
        if($row = $res -> fetch_assoc())    {
            $PGCI = $row['pgci'];
            $MODEL = $row['method'];
        }else{
            echo("Nothing to show...");
        }
        ?>

        <div class="container" >
            <div style=" width: 150px; position: absolute
               ↪ ; top: 55%;left: 50%; transform:
               ↪ translate(-50%,-50%); height: 50vh;">
                <div style="width: 250px; height: 0;
                   ↪ position: absolute; top: 50%;left:
                   ↪ 50%; transform: translate
                   ↪ (-50%,-50%); background-color: rgba
                   ↪ (180,180,180,0.8); border-radius:
                   ↪ 50px;">
                        <div class="alert alert-secondary
                           ↪ " style="width: 89vw; max-
                           ↪ width: 350px; position:
                           ↪ absolute; top: 50%;left:
                           ↪ 50%; transform: translate
                           ↪ (-50%,-50%); background-
                           ↪ color:rgba
                           ↪ (255,255,255,0.60) ;border-
                           ↪ radius: 20px; border: 5px
                           ↪ solid rgb(255,255,255);-
                           ↪ webkit-filter: grayscale
                           ↪ (20%);filter: grayscale
                           ↪ (20%);"><div class="box">
                            <h3 >P-ICGI Virtual Sensor
                              ↪ </h3>
                              <div>(Predicted-Indoor
                                 ↪ Global Comfort
                                 ↪ Index)</div>
                              <hr>
                              <div class="gci-box">
                                  <p class="left">
                                      Current <strong>P
                                         ↪ -IGCI</
```

```
                                        ↪ strong> :
                                    </p>
                                    <p class="right"> <b
                                        ↪ ><?php echo(
                                        ↪ round($PGCI,2))
                                        ↪  ?> / 5</b> </p
                                        ↪ >
                                </div>
                                <p class="box2">Best
                                    ↪ identified method :
                                    ↪  <b><?php echo(
                                    ↪ $MODEL) ?></b> <br
                                    ↪ ></p>
                                <hr>
                                <div style="font-size:
                                    ↪ 70%;">Last update :
                                    ↪  <?php echo(date("Y
                                    ↪ -m-d H:i")) ?></div
                                    ↪ >
                                <br>
                                <div  style="font-size:
                                    ↪ 70%; text-align:
                                    ↪ end;" >Powered by <
                                    ↪ strong>Stefano
                                    ↪ Riffelli</strong></
                                    ↪ div>
                        </div>
                    </div>
                </div>
            </div>

        </div>


    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS
        ↪ -->
    <script src="https://code.jquery.com/jquery-3.2.1.
        ↪ slim.min.js" integrity="sha384-
        ↪ KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/
        ↪ Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="
        ↪ anonymous"></script>
```

```
    <script src="https://cdnjs.cloudflare.com/ajax/libs/
       ↪ popper.js/1.12.9/umd/popper.min.js" integrity="
       ↪ sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/
       ↪ ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin
       ↪ ="anonymous"></script>
    <script src="https://maxcdn.bootstrapcdn.com/
       ↪ bootstrap/4.0.0/js/bootstrap.min.js" integrity=
       ↪ "sha384-JZR6Spejh4U02d8jOt6vLEHfe/
       ↪ JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
       ↪ crossorigin="anonymous"></script>
  </body>
</html>
```

Appendix D

Virtual sensor implementation

D.1 Launch program - *run.py*

```
#!/usr/bin/env python3
# Python 3.9.7

'''
    run.py
    Main method
'''

import sys
import os
from src import *
from src.lib import log

# checking python version
assert (sys.version_info[0] == 3), "Python version must be 3"


# main method
def main():
    print("IEQ automation")

    # DATA COLLECTION
    control.getData()  # use this syntax to take ALL questionnaires  and related measurements
```

```
    '''
    Otherwise you can use
    ontrol.getData(startdate="2021-3-3 12:00:00", enddate
       ↪ ="2021-4-28 17:00:00")
    to take questionnaire between two dates
    '''

    # calculating data average
    control.calculateAverages()

    # SESSION GENERATION
    try:
        control.generateSession()
    except Exception as e:
        log.error(e)

    # MODEL TRAINING AND MODEL IDENTIFICATION
    control.generateMatlabModel()

    # P-ICGI PROCESSING
    control.virtualSensor()


# Defining main as the main program
if __name__ == "__main__":
    main()
```

D.2 Software core - *array.py*

```
'''
    Class Control
    the class collects in a control all the
       ↪ questionnaires and for each questionnaire
'''

import os
import csv
import locale
import matlab.engine
from datetime import datetime, timedelta
```

```
import matplotlib.pyplot as plt
from scipy import stats
import numpy as np
from sklearn.model_selection import train_test_split
import pandas as pd
import statsmodels.api as sm
from src.lib.db import Request
from src.lib.log import log, error, logInfo
from src.lib.questionnaire import Questionnaire
from src.lib.session import Session


class Control:
    def __init__(self):
        self.arrayQuestionnaire = []
        self.arraySession = []
        self.workingDirectory = os.getcwd()
        self.outputPath = self.workingDirectory + '/
           ↪ export/output.csv'
        self.tmpPath = self.workingDirectory + "/export/
           ↪ tmp.csv"

    def addQuestionnaire(self, id, timestamp, gender, age
         ↪ , comfortGlobal, comfortThermal,
         ↪ comfortAcoustic,
                         comfortVisual,
                         airQuality, tsv, airMovement,
                            ↪ clothing, unwantedNoise,
                            ↪ noiseConsequences,
                            ↪ lighting,
                         respirability, badOdours):
        self.arrayQuestionnaire.append(
            Questionnaire(id, timestamp, gender, age,
               ↪ comfortGlobal, comfortThermal,
               ↪ comfortAcoustic,
                          comfortVisual, airQuality, tsv,
                             ↪  airMovement, clothing,
                          unwantedNoise,
                             ↪ noiseConsequences,
                             ↪ lighting, respirability,
                             ↪ badOdours))

    def getQuestionnaire(self, number):
```

```python
        return self.arrayQuestionnaire[number]

    def getArrayLen(self):
        return len(self.arrayQuestionnaire)

    # the method recovers from the API the carried out questionnaires, selects them and recovers the relative measurements,
    # compute the averages and the standard deviations and create the arrays on which to carry out operations and calculations
    def getData(self, **kargs):
        r = Request()
        # Check if a start date has been entered as an optional parameter
        # retrieving questionnaires from API
        logInfo("Getting data from Database...")
        '''
            if the start and end date are entered, the questionnaires
            between the two dates are recovered, otherwise all are retrieved.
        '''
        if len(kargs) == 2:
            if "startdate" in kargs and "enddate" in kargs:
                startdate = datetime.strptime(kargs['startdate'], '%Y-%m-%d %H:%M:%S')
                enddate = datetime.strptime(kargs['enddate'], '%Y-%m-%d %H:%M:%S')
                # retrieving questionnaires from the db
                response = r.getQuestionnaire(startdate=startdate, enddate=enddate)
        else:
            response = r.getQuestionnaire()

        logInfo("OK")
        logInfo("creating objects...")
        # Converting questionnaires from json to Questionnaire objects
        if 'questionnaire' in response:
            for element in response['questionnaire']:
                self.addQuestionnaire(
```

```
                    element['id'],
                    datetime.strptime(element['timestamp'
                        ↪ ], '%Y-%m-%d %H:%M:%S'),
                    element['gender'],
                    element['age'],
                    element['comfortGlobal'],
                    element['comfortThermal'],
                    element['comfortAcoustic'],
                    element['comfortVisual'],
                    element['airQuality'],
                    element['sensazioneTerminca'],
                    element['airMovement'],
                    element['clothing'],
                    element['unwantedNoise'],
                    element['noiseConsequences'],
                    element['lighting'],
                    element['respirability'],
                    element['badOdours']
                )
            logInfo("OK")
            # recovering the findings inherent in each
                ↪ questionnaire
            logInfo("Cross-referencing data with surveys
                ↪ ...")
            for x in list(
                    range(
                        self.getArrayLen())):  # cycle
                            ↪ from 0 to the length of the
                            ↪  object control. x
                            ↪ increments by 1 at each
                            ↪ cycle
                elem = self.getQuestionnaire(x)  # ask
                    ↪ the class to return questionnaire
                    ↪ number x
                logInfo("# Id questionnaire -> " + elem.
                    ↪ getIDQuestionnaire() + " Data --> "
                    ↪  + str(
                    elem.getJson()['timestamp']))
                logInfo(elem.getJson())
                res = r.getQuestionnaireMeasurements(elem
                    ↪ .getJson()['timestamp'])
                # search and retrieve from the database
                    ↪ the surveys included in the time
```

```
   ↪ interval (questionnaire time - 1
   ↪ hour)
# convert surveys into objects
for temp in res['measurement']:
    elem.addMeasurement(
        temp['id'],
        temp['timestamp'],
        temp['luminosity'],
        temp['co2'],
        temp['temperature'],
        temp['humidity'],
        temp['noise']
    )
# Display on screen the findings of each
   ↪ questionnaire

if (len(elem.arrayMeasurements) > 0):
    log("Hourly measurements: ")
    for y in list(range(elem.
       ↪ getMeasurementNumber())):
        log(elem.getMeasurements(y).
           ↪ to_string())

    # Average objective values
    log("Mean luminosity -->\t" + str(
       ↪ elem.getLuminosityAverage()))
    log("Mean co2 -->\t\t" + str(elem.
       ↪ getCO2Average()))
    log("Mean temperature -->\t" + str(
       ↪ elem.getTemperatureAverage()))
    log("Mean humidity -->\t" + str(elem.
       ↪ getHumidityAverage()))
    log("Mean noise -->\t\t" + str(elem.
       ↪ getNoiseAverage()))
    log("standard deviation luminosity
       ↪ -->\t" + str(elem.
       ↪ getDevLuminosity()))
    log("standard deviation co2 -->\t\t"
       ↪ + str(elem.getDevCo2()))
    log("standard deviation temperature
       ↪ -->\t" + str(elem.
       ↪ getDevTemperature()))
```

```
                    log("standard deviation humidity -->\t\t" + str(elem.getDevHumidity()))
                    log("standard deviation noise -->\t\t" + str(elem.getDevNoise()))
                else:
                    logInfo("\t\tnothing measurements to show")
        else:
            logInfo("No Object to create")
        logInfo("ok")

    # the method calculates the means and standard deviations for each questionnaire
    def calculateAverages(self):
        if self.getArrayLen() > 0:
            logInfo("Processing...")
            for x in list(
                    range(
                        self.getArrayLen())):  # cycle from 0 to the length of the  object control. x increments by 1 at each cycle
                elem = self.getQuestionnaire(x)  # ask the class to return questionnaire number x

                # Average objective values
                log("Mean luminosity -->\t" + str(elem.getLuminosityAverage()))
                log("Mean co2 -->\t\t" + str(elem.getCO2Average()))
                log("Mean temperature -->\t" + str(elem.getTemperatureAverage()))
                log("Mean humidity -->\t" + str(elem.getHumidityAverage()))
                log("Mean noise -->\t\t" + str(elem.getNoiseAverage()))
                log("standard deviation luminosità -->\t" + str(elem.getDevLuminosity()))
                log("standard deviation co2 -->\t\t" + str(elem.getDevCo2()))
```

```
                log("standard deviation temperatura -->\t" + str(elem.getDevTemperature()))
                log("standard deviation umidità -->\t\t" + str(elem.getDevHumidity()))
                log("standard deviation noise -->\t\t" + str(elem.getDevNoise()))
            logInfo("OK")
        else:
            error("Error in <Object>.calculateAverages() -->Empty Array\nUse First : <Object>.getData()")

    # the method receives as input the parameters x, y, the name of the graph and the name
    # of the two axes and generates a graph and saves it locally.
    def getLinearRegressionGraph(self, x, y, graphName, xlabel, ylabel):
        coef = np.polyfit(x, y, 1)
        poly1d_fn = np.poly1d(coef)
        plt.plot(x, y, 'yo', x, poly1d_fn(x), '--k')
        plt.title(graphName,
                  fontdict={'family': 'serif',
                            'color': 'darkblue',
                            'weight': 'bold',
                            'size': 18})
        plt.grid(True)
        plt.xlabel(xlabel, size=16)
        plt.ylabel(ylabel, size=16)

        def formatString(str):
            res = ""
            for x in str:
                if (x == " "):
                    res += "_"
                elif (x != "/"):
                    res += x
                else:
                    res += "-"
            return res

        plt.savefig("graph/" + formatString(graphName) + '.png')
```

```
        plt.show()
        plt.clf()

    def getLinearRegPVMVSTemp(self):
        if self.getArrayLen() > 0:  # check if data have
            ↪ already been entered
            temp = []
            pmv = []
            logInfo("fill dataset...")
            # create a json and then create a dataset
            for x in list(range(self.getArrayLen())):
                if not self.arrayQuestionnaire[x].
                    ↪ checkNone():
                    t1 = self.arrayQuestionnaire[x].
                        ↪ getTemperatureAverage()
                    t2 = self.arrayQuestionnaire[x].
                        ↪ getTSV()
                    if t1 != None and t2 != None:
                        temp.append(float(t1))
                        pmv.append(int(t2))

            logInfo("OK")
            self.getLinearRegressionGraph(temp, pmv, "PMV
                ↪  question VS Temperature Average", "
                ↪ Mean Objective Temperature [°C]", "PMV
                ↪ ")

        else:
            error("Error in <Object>.
                ↪ getLinearRegPVMVSTemp() --> Empy array\
                ↪ nUse first : <Object>.getData()")

    def getLinearRegAcusticNoise(self):
        if self.getArrayLen() > 0:  # check if data have
            ↪ already been entered
            x = []
            y = []
            logInfo("fill dataset...")
            for a in list(range(self.getArrayLen())):
                if not self.arrayQuestionnaire[a].
                    ↪ checkNone():
                    t1 = self.arrayQuestionnaire[a].
                        ↪ getNoiseAverage()
```

```
                    t2 = self.arrayQuestionnaire[a].getComfortAcustic()
                    if t1 != None and t2 != None:
                        x.append(float(t1))
                        y.append(float(t2))
            logInfo("ok")
            self.getLinearRegressionGraph(x, y, "Acustic Comfort question VS Noise Average", "Noise [dbA]", "Acoustic comfort question")

        else:
            error("Error in <Object>.getLinearRegAcusticNoise() --> Empty Array\nUse First : <Object>.getData()")

    def getLinearRegNoiseQuestVSNoise(self):
        if self.getArrayLen() > 0:  # check if data have already been entered
            x = []
            y = []
            logInfo("fill dataset...")
            for a in list(range(self.getArrayLen())):
                if not self.arrayQuestionnaire[a].checkNone():
                    t1 = self.arrayQuestionnaire[a].getNoiseAverage()
                    t2 = self.arrayQuestionnaire[a].getUnwantedNoise()
                    if t1 != None and t2 != None:
                        x.append(float(t1))
                        y.append(float(t2))
            logInfo("ok")
            self.getLinearRegressionGraph(x, y, "Noise question vs Noise average", "Noise [dbA]", "Consequence of noise question")

        else:
            error("Error in <Object>.getLinearRegNoiseQuestVSNoise() --> Empty Array\nUse First : <Object>.getData()")

```

```
    def getLinearRegNoiseConsQuestVSNoise(self):
        if self.getArrayLen() > 0:  # check if data have already been entered
            x = []
            y = []
            logInfo("fill dataset...")
            for a in list(range(self.getArrayLen())):
                if not self.arrayQuestionnaire[a].checkNone():
                    t1 = self.arrayQuestionnaire[a].getNoiseAverage()
                    t2 = self.arrayQuestionnaire[a].getNoiseConsequence()
                    if t1 != None and t2 != None:
                        x.append(float(t1))
                        y.append(float(t2))
            logInfo("ok")
            self.getLinearRegressionGraph(x, y, "Consequence of noise question vs Noise average", "Noise average [dbA]", "Noise question")

        else:
            error(
                "Error in <Object>.getLinearRegNoiseConsQuestVSNoise() --> Empty Array\nUse First : <Object>.getData()")

    def getRegLightquestVSLumAverage(self):
        if self.getArrayLen() > 0:  # check if data have already been entered
            x = []
            y = []
            logInfo("fill dataset...")
            for a in list(range(self.getArrayLen())):
                if not self.arrayQuestionnaire[a].checkNone():
                    t1 = self.arrayQuestionnaire[a].getLuminosityAverage()
                    t2 = self.arrayQuestionnaire[a].getLighting()
                    if t1 != None and t2 != None:
```

```
                        x.append(float(t1))
                        y.append(float(t2))
            logInfo("ok")
            self.getLinearRegressionGraph(x, y, "Lighting question vs Luminosity average", "Luminosity [lux]", "Lighting question ")

        else:
            error("Error in <Object>.getRegLightquestVSLumAverage() --> Empty Array\nUse First : <Object>.getData()")

    def getRegIAQcomfQuestVSCo2Avr(self):
        if self.getArrayLen() > 0:  # check if data have already been entered
            x = []
            y = []
            logInfo("fill dataset...")
            for a in list(range(self.getArrayLen())):
                if not self.arrayQuestionnaire[a].checkNone():
                    t1 = self.arrayQuestionnaire[a].getCO2Average()
                    t2 = self.arrayQuestionnaire[a].getIAQComfort()
                    if t1 != None and t2 != None:
                        x.append(float(t1))
                        y.append(float(t2))
            logInfo("ok")
            self.getLinearRegressionGraph(x, y, "IAQ comfort question vs CO2 average", "CO2 [ppm]", "IAQ comfort question")

        else:
            error("Error in <Object>.getRegIAQcomfQuestVSCo2Avr() --> Empty Array\nUse First : <Object>.getData()")

    def getRegRespirabilityQuestVSCO2Average(self):
        if self.getArrayLen() > 0:  # check if data have already been entered
```

```
            x = []
            y = []
            logInfo("fill dataset...")
            for a in list(range(self.getArrayLen())):
                if not self.arrayQuestionnaire[a].
                    ↪ checkNone():
                    t1 = self.arrayQuestionnaire[a].
                        ↪ getCO2Average()
                    t2 = self.arrayQuestionnaire[a].
                        ↪ getRespirability()
                    if t1 != None and t2 != None:
                        x.append(float(t1))
                        y.append(float(t2))
            logInfo("ok")
            self.getLinearRegressionGraph(x, y, "
                ↪ Respirability question vs CO2 average",
                ↪  "CO2 [ppm]", "Respirability question")

        else:
            error(
                "Error in <Object>.
                    ↪ getRegRespirabilityQuestVSCO2Average
                    ↪ () --> Empty Array\nUse First : <
                    ↪ Object>.getData()")

    # the method allows exporting questionnaires
        ↪ retrieved from the API
    def exportCSV(self):
        if self.getArrayLen() > 0:  # check if data have
            ↪ already been entered
            locale.setlocale(locale.LC_ALL, '')
            with open(self.tmpPath, 'w', newline='',
                ↪ encoding="utf-8") as file:
                writer = csv.writer(file, quoting=csv.
                    ↪ QUOTE_NONNUMERIC, delimiter=',')
                writer.writerow(
                    ["Timestamp", "comfortGlobal", "
                        ↪ comfortThermal", "
                        ↪ comfortAcoustic", "
                        ↪ comfortVisual",
                     "airQuality", "PMV", "Air speed", "
                         ↪ Clothing insulation", "Noise",
                         ↪  "Consequences of noise",
```

```
                    "Lighting", "Respirability", "Bad odors", "", "Clothing insulation (n)",
                    "Consequences of noise (n)", "Lighting (n)", "", "luminosity average μ [lux]",
                    "co2 average μ [ppm]", "temperature average μ [°C]", "humidity average μ [%]",
                    "noise average μ [dBA]", "", "luminosity var σ [lux]", "co2 var σ [ppm]",
                    "temperature var σ [°C]", "humidity var σ [%]", "noise var σ [dBA]"])
                for elem in self.arrayQuestionnaire:
                    if (not (elem.none or len(elem.arrayMeasurements) < 1)):
                        writer.writerow(
                        [
                        elem.timestamp,
                        elem.comfortGlobal,
                        elem.comfortThermal,
                        elem.comfortAcoustic,
                        elem.comfortVisual,
                        elem.airQuality,
                        elem.tsv,
                        elem.airMovement,
                        elem.reconvertClothing(elem.clothing),
                        elem.unwantedNoise,
                        elem.reconvertNoiseConsequence(elem.noiseConsequences),
                        elem.reconvertLuminosity(elem.lighting),
                        elem.respirability,
                        elem.badOdours,
                        "",
                        elem.clothing,
                        elem.unwantedNoise,
                        elem.lighting,
                        "",
                        elem.getLuminosityAverage(),
```

```
                        elem.getCO2Average(),
                        elem.getTemperatureAverage(),
                        elem.getHumidityAverage(),
                        elem.getNoiseAverage(),
                        "",
                        elem.getDevLuminosity(),
                        elem.getDevCo2(),
                        elem.getDevTemperature(),
                        elem.getDevHumidity(),
                        elem.getDevNoise()
                        ])

        else:
            error("Error in <Object>.exportCSV()--> Empty Array\nUse First : <Object>.getData()" )

    def mlr(self, csvPath):

        if csvPath == "":
            self.exportCSV()
            df = pd.read_csv(self.tmpPath)
        else:
            df = pd.read_csv(csvPath, delimiter=';')

        print(df.head())  # Show data

        X = df[['ThermalComfort', 'AcousticComfort', 'VisualComfort', 'IAQComfort']]  # independent variables
        y = df['GlobalComfort']  # dependent variables

        # to get intercept
        X = sm.add_constant(X)

        # fit the regression model
        reg = sm.OLS(y, X).fit()
        print(reg.summary())

    # the method generates the sessions from the questionnaires and the relative measurements
    # retrieved from the database
    def generateSession(self):
```

```
        arrayTmp = []
        arrayTmp2 = []
        # removing all questionnaire witout objectives data and with null param
        for elem in self.arrayQuestionnaire:
            if (not (elem.none or len(elem.arrayMeasurements) < 1)):
                arrayTmp.append(elem)
        arrayTmp.reverse()  # reverse array object's positions
        for a in arrayTmp:
            print(a.timestamp)
            print(str(a.getMeasurementNumber()))
            print("\n")
        # if returned array is not empty, do
        if len(arrayTmp) > 0:
            cont = 0

            x = arrayTmp.__getitem__(cont)
            while cont + 1 < len(arrayTmp):
                arrayTmp2 = []
                s = Session(x.timestamp)
                arrayTmp2.append(x)
                startdate = x.timestamp
                enddate = x.timestamp + timedelta(hours=1)
                cont += 1
                x = arrayTmp.__getitem__(cont)
                while startdate <= x.timestamp <= enddate and cont < len(arrayTmp):
                    arrayTmp2.append(x)
                    cont += 1
                    x = arrayTmp.__getitem__(cont)
                o = 0
                s.addArray(arrayTmp2)
                logInfo(s.to_string())
                self.arraySession.append(s)
            return self.__generateDatasetFromSession()
        else:
            error("Error in <Object>.generateSession() --> Array is empty\nUse First : <Object>.getData()")

```

```python
    def getSessionNumber(self):
        return len(self.arraySession)

    # private method to generate numpy array and csv from session array
    def __generateDatasetFromSession(self):
        if len(self.arraySession) > 0:
            pd.set_option('display.max_rows', 500)
            pd.set_option('display.max_columns', 500)
            pd.set_option('display.width', 1000)
            dfin = pd.DataFrame(columns=['Luminosity', "Co2", 'Temperature', "Humidity", "Noise"])
            dfout = pd.DataFrame(columns=["GC"])
            for x in self.arraySession:
                dfin.loc[dfin.shape[0]] = [x.luminosity, x.co2, x.temperature, x.humidity, x.noise]
                dfout.loc[dfout.shape[0]] = [x.gc]
            dfin = dfin.round(4)
            dfout = dfout.round(4)
            # saving csv on local path
            dfin.to_csv("export/inputSession.csv", index=False)
            dfout.to_csv("export/outputSession.csv", index=False)
            dfin = dfin.to_numpy()
            dfout = dfout.to_numpy()
            # return 2 numpy array
            return dfin, dfout

        else:
            raise Exception("Cannot generate dataset from Session: session array is empty")

    # method starts matlab engine and executes matlab file for model comparison and export of the best model
    def generateMatlabModel(self):
        logInfo("starting matlab engine...")
        eng = matlab.engine.start_matlab()
        res = eng.matlab(nargout=0)
        logInfo("Stopping matlab")
```

```
        eng.quit()

    # the method starts the matlab engine and executes
        ↪ the matlab file for
    # the prediction of a model taking as input the
        ↪ exported model and the last measurement by the
        ↪ API
    def virtualSensor(self):
        logInfo("starting matlab engine...")
        eng = matlab.engine.start_matlab()
        res = eng.matlab_pred(nargout=0)
        logInfo("Stopping matlab")
        eng.quit()
```

D.3 Python libraries

D.3.1 *credentials.py*

```
'''
    Class credentials
    the class provides the definition of the constants
'''

#Define API Token
TOKEN = "XXX"
#Define API link
APILINK = "http://www.riffelli.it/IEQ/API/"
```

D.3.2 *db.py*

```
'''
    Class db
    the class provide communication with API
'''

# Import Module
import requests
```

```
import json
from datetime import datetime
from .credentials import TOKEN, APILINK


# Define Class
class Request:
    def __init__(self):
        # Define API Token
        global TOKEN, APILINK
        self.TOKEN = TOKEN
        # Define API link
        self.LINK = APILINK + "analysis.php?token=" + self.TOKEN + "&&"

    # the method retrieves from the API all the questionnaires,
    # otherwise it can retrieve only those carried out in a given period
    def getQuestionnaire(self, **kargs):
        param = ""
        action = "action=getQuestionnaire"
        if "startdate" in kargs and "enddate" in kargs:
            param += "&&startdate=" + str(kargs['startdate'])
            param += "&&enddate=" + str(kargs['enddate'])

        print(self.LINK + action + param)

        try:
            response = requests.get(self.LINK + action + param)
            response.raise_for_status()

        except requests.exceptions.TooManyRedirects as error:
            print(error)

        if (response.status_code == 200):
            return json.loads(response.json())
        elif (response.status_code == 404):
            print("Result not found!")

```

```
    # the method retrieves from the API the relative
       ↪ measurements of each questionnaire according to
       ↪  the time of
    # compilation. the measurements are retrieved from
       ↪ the hour before to the questionnaire
       ↪ compilation time.
    def getQuestionnaireMeasurements(self, data2):
        data1 = datetime.timestamp(data2)
        data1 -= 3600
        data1 = datetime.fromtimestamp(data1)
        try:
            response = requests.get(
                self.LINK + "action=
                   ↪ getQuestionnaireMeasurements&&data1
                   ↪ =" + str(data1) + "&&data2=" + str(
                   ↪ data2))
            response.raise_for_status()
        except requests.exceptions.TooManyRedirects as
           ↪ error:
            print(error)

        if (response.status_code == 200):
            return json.loads(response.json())
        elif (response.status_code == 404):
            print("Result not found!")
```

D.3.3 *log.py*

```
'''
    Class log
    the class provides methods for displaying basic
       ↪ information and errors generated during
       ↪ execution on the screen

    Constant Definition, true = screen log  --- False =
       ↪ no log
        LOG is used to print the results of operations on
           ↪  the screen.
        LOGINFO is used to print on the screen the
           ↪ information relative to what is being
```

```
            ↪ executed.
        ERROR it is used to print on the screen the
            ↪ errors detected by the code.
'''

LOG = True
LOGINFO = True
ERROR = True


def logInfo(string):
    if LOGINFO:
        print(string)

def error(string):
    if ERROR:
        print("\33[1;31m"+str(string)+"\33[m")

def warning(string):
    if ERROR:
        print("\033[93m"+str(string)+"\31")

def log(string):
    if LOG:
        print(string)
```

D.3.4 *measurement.py*

```
'''
    Class measurement
    the class contains all the methods and attributes
        ↪ necessary for cataloguing the measurements made
        ↪  by the IEQ Logger
'''
class Measurement:
    def __init__(self, id, timestamp, luminosity, co2,
        ↪ temperature, humidity, noise):

        self.id = id
        self.timestamp = timestamp
```

```
        self.luminosity = float(luminosity)
        self.co2 = float(co2)
        self.temperature = float(temperature)
        self.humidity = float(humidity)
        self.noise = float(noise)

    def to_string(self):
        return "id --> " + str(self.id) + "\ttimestamp
            ↪ --> " + str(self.timestamp) +"\tluminosity
            ↪ --> "+ str(self.luminosity) + "\tco2 --> "
            ↪ + str(self.co2) + "\ttemperature --> " +
            ↪ str(self.temperature) + "\thumidity --> " +
            ↪ str(self.humidity) + "\tnoise --> " + str(
            ↪ self.noise)

    def getLuminosity(self):
        return self.luminosity
    def getCo2(self):
        return self.co2
    def getTemperature(self):
        return self.temperature
    def getHumidity(self):
        return self.humidity
    def getNoise(self):
        return self.noise
```

D.3.5 *questionnaire.py*

```
'''
    Classe Questionnaire
    contains all the methods and attributes necessary for
        ↪ cataloguing student questionnaires from the
        ↪ website www.riffelli.it/IEQ/questionnaire
'''
import statistics

from src.lib.measurement import Measurement


class Questionnaire:
```

```python
    def __init__(self, id, timestamp, gender, age, comfortGlobal, comfortThermal,
                 comfortAcoustic, comfortVisual, airQuality, tsv, airMovement,
                 clothing, unwantedNoise, noiseConsequences, lighting, respirability,
                 badOdours):
        self.arrayMeasurements = []  # Array of questionnaire surveys
        self.none = False  # Boolean variable, True if there are empty values in the questionnaire, else False
        self.idQuestionnaire = id
        self.timestamp = timestamp
        self.gender = self.insertNone(gender)
        self.age = self.insertNone(age)
        self.comfortGlobal = self.insertNone(comfortGlobal)
        self.comfortThermal = self.insertNone(comfortThermal)
        self.comfortAcoustic = self.insertNone(comfortAcoustic)
        self.comfortVisual = self.insertNone(comfortVisual)
        self.airQuality = self.insertNone(airQuality)
        self.tsv = self.insertNone(tsv)
        self.airMovement = self.insertNone(airMovement)
        self.clothing = self.insertNone(self.convertClothing(clothing))
        self.unwantedNoise = self.insertNone(unwantedNoise)
        self.noiseConsequences = self.insertNone(self.convertNoiseConsequences(noiseConsequences))
        self.lighting = self.insertNone(self.convertLuminosity(lighting))
        self.respirability = self.insertNone(respirability)
        self.badOdours = self.insertNone(badOdours)

    def addMeasurement(self, id, timestamp, luminosity, co2, temperature, humidity, noise):
```

```
        self.arrayMeasurements.append(Measurement(id,
           ↪ timestamp, luminosity, co2, temperature,
           ↪ humidity, noise))

    def getJson(self):
        return {
            "timestamp": self.timestamp,
            "comfortGlobal": self.comfortGlobal,
            "comfortThermal": self.comfortThermal,
            "comfortAcoustic": self.comfortAcoustic,
            "comfortVisual": self.comfortVisual,
            "airQuality": self.airQuality,
            "tsv": self.tsv,
            "airMovement": self.airMovement,
            "clothing": self.clothing,
            "unwantedNoise": self.unwantedNoise,
            "noiseConsequences": self.noiseConsequences,
            "lighting": self.lighting,
            "respirability": self.respirability,
            "badOdours": self.badOdours,
        }

    # the function inserts None instead of empty fields
    def insertNone(self, param):
        if param == "":
            self.none = True
            res = None
        else:
            res = param
        return res

    '''
        Get and Set methods
    '''

    def getMeasurements(self, number):
        return self.arrayMeasurements[number]

    def getMeasurementNumber(self):
        res = len(self.arrayMeasurements)
        return res

    def getTSV(self):
```

```
        return self.tsv

    def getComfortAcustic(self):
        return self.comfortAcoustic

    def getUnwantedNoise(self):
        return self.unwantedNoise

    def getNoiseConsequence(self):
        return self.noiseConsequences

    def getLighting(self):
        return self.lighting

    def getIAQComfort(self):
        return self.airQuality

    def getRespirability(self):
        return self.respirability

    def getIDQuestionnaire(self):
        return self.idQuestionnaire

    '''
        Average and standard deviation
    '''

    def getLuminosityAverage(self):
        sum = 0
        if (len(self.arrayMeasurements) > 0):
            for temp in self.arrayMeasurements:
                sum = sum + float(temp.getLuminosity())
            return sum / len(self.arrayMeasurements)
        else:
            return None

    def getCO2Average(self):
        sum = 0
        if (len(self.arrayMeasurements) > 0):
            for temp in self.arrayMeasurements:
                sum = sum + float(temp.getCo2())
            return sum / len(self.arrayMeasurements)
        else:
```

```
            return None

    def getTemperatureAverage(self):
        sum = 0
        if (len(self.arrayMeasurements) > 0):
            for temp in self.arrayMeasurements:
                sum = sum + float(temp.getTemperature())
            return sum / len(self.arrayMeasurements)
        else:
            return None

    def getHumidityAverage(self):
        sum = 0
        if (len(self.arrayMeasurements) > 0):
            for temp in self.arrayMeasurements:
                sum = sum + float(temp.getHumidity())
            return sum / len(self.arrayMeasurements)
        else:
            return None

    def getNoiseAverage(self):
        sum = 0
        if (len(self.arrayMeasurements) > 0):
            for temp in self.arrayMeasurements:
                sum = sum + float(temp.getNoise())
            return sum / len(self.arrayMeasurements)
        else:
            return None

    def getDevLuminosity(self):
        temp = []
        if (len(self.arrayMeasurements) > 0):
            for x in self.arrayMeasurements:
                temp.append(x.getLuminosity())
            return statistics.stdev(temp)
        else:
            return None

    def getDevCo2(self):
        temp = []
        if (len(self.arrayMeasurements) > 0):
            for x in self.arrayMeasurements:
                temp.append(x.getCo2())
```

```
            return statistics.stdev(temp)
        else:
            return None

    def getDevTemperature(self):
        temp = []
        if (len(self.arrayMeasurements) > 0):
            for x in self.arrayMeasurements:
                temp.append(x.getTemperature())
            return statistics.stdev(temp)
        else:
            return None

    def getDevHumidity(self):
        temp = []
        if (len(self.arrayMeasurements) > 0):
            for x in self.arrayMeasurements:
                temp.append(x.getHumidity())
            return statistics.stdev(temp)
        else:
            return None

    def getDevNoise(self):
        temp = []
        if (len(self.arrayMeasurements) > 0):
            for x in self.arrayMeasurements:
                temp.append(x.getNoise())
            return statistics.stdev(temp)
        else:
            return None

    def convertClothing(self, param):
        if (param == "summer-clothing"):
            res = 0.5
        elif (param == "standard-clothing"):
            res = 0.75
        elif (param == "winter-clothing"):
            res = 1
        else:
            res = None
        return res

    def convertNoiseConsequences(self, param):
```

```
        if (param == "yes"):
            res = 4
        elif (param == "almost-always"):
            res = 3
        elif (param == "almost-never"):
            res = 2
        elif (param == "no"):
            res = 1
        else:
            res = None
        return res

    def convertLuminosity(self, param):
        if (param == "insufficient"):
            res = -1
        elif (param == "excessive"):
            res = 1
        elif (param == "appropriate"):
            res = 0
        else:
            res = None
        return res

    def reconvertClothing(self, param):
        if (param == 0.5):
            res = "summer-clothing"
        elif (param == 0.75):
            res = "standard-clothing"
        elif (param == 1):
            res = "winter-clothing"
        else:
            res = None
        return res

    def reconvertNoiseConsequence(self, param):
        if (param == 4):
            res = "yes"
        elif (param == 3):
            res = "almost-always"
        elif (param == 2):
            res = "almost-never"
        elif (param == 1):
            res = "no"
```

```
        else:
            res = None
        return res

    def reconvertLuminosity(self, param):
        if (param == -1):
            res = "insufficient"
        elif (param == 1):
            res = "excessive"
        elif (param == 0):
            res = "appropriate"
        else:
            res = None
        return res

    def checkNone(self):
        return self.none

    def getClothing(self):
        res = ""
        if self.clothing is not None:
            res = self.clothing
        return res
```

D.3.6 *session.py*

```
'''
    Class Session
    the class stores the relevant questionnaires and
        ↪ performs operations on them
'''

class Session:
    # method to initialized the class
    def __init__(self, timestamp):
        self.timestamp = timestamp
        self.array = []

    # method to ad questionnaire array
    def addArray(self, array):
```

```
        self.array = array
        self.mean()

    # method to process the means of parameters
    def mean(self):
        if len(self.array) > 0:
            for i in range(0, 6):
                sum = 0
                for j in self.array:
                    sum += self.__switch(i, j)
                mean = sum / len(self.array)
                if i == 0:
                    self.luminosity = mean
                elif i == 1:
                    self.temperature = mean
                elif i == 2:
                    self.humidity = mean
                elif i == 3:
                    self.noise = mean
                elif i == 4:
                    self.co2 = mean
                elif i == 5:
                    self.gc = mean

    def __switch(self, i, j):
        if i == 0:
            res = j.getLuminosityAverage()
        elif i == 1:
            res = j.getTemperatureAverage()
        elif i == 2:
            res = j.getHumidityAverage()
        elif i == 3:
            res = j.getNoiseAverage()
        elif i == 4:
            res = j.getCO2Average()
        elif i == 5:
            res = int(j.comfortGlobal)
        return res

    # method to print the session values
    def to_string(self):
        string = ""
```

```
        string += "------------------------------------------
--------------------\n"
        for j in self.array:
            string += "timestamp -> " + j.timestamp.strftime('%Y-%m-%d %H:%M:%S') + "\n"
            string += "\t measurements number : " + str(j.getMeasurementNumber()) + "\n"

        string += "\n"
        string += "Mean Luminosity -> " + str(self.luminosity)
        string += "\n"
        string += "Mean Temperature -> " + str(self.temperature)
        string += "\n"
        string += "Mean Humidity -> " + str(self.humidity)
        string += "\n"
        string += "Mean Noise -> " + str(self.noise)
        string += "\n"
        string += "Mean Co2 -> " + str(self.co2)
        string += "\n"
        string += "Mean GC -> " + str(self.gc)

        return string
```

D.4 Matlab libraries

D.4.1 *matlab.m*

```
addpath('Matlab_lib')
inputSession = readtable('export/inputSession.csv');
outputSession = readtable('export/outputSession.csv');
filename = "export\matlabTrainedModel\trainedModel.mat";

inputSession = table2array(inputSession);
outputSession = table2array(outputSession);

% processing models
```

```
[linearRegressionModel, RMSElinearRegressionModel] = linearRegression(inputSession,outputSession)
[stepwiselinearRegressionModel, RMSEstepwiselinearRegressionModel]  = stepwiselinearRegression(inputSession,outputSession)
[robustlinearRegressionModel, RMSErobustlinearRegressionModel] = robustlinearRegression(inputSession,outputSession)
[interactionsLinearRegressionModel, RMSEinteractionsLinearRegressionModel]  = interactionsLinearRegression(inputSession, outputSession)
[fineTreeModel, RMSEfineTreeModel] = fineTree(inputSession,outputSession)
[mediumTreeModel, RMSEmediumTreeModel] = mediumtree(inputSession,outputSession)
[coarseTreeModel, RMSEcoarseTreeModel] = coarsetree(inputSession,outputSession)
[linearSvmModel, RMSElinearSvmModel] = linearsvm(inputSession,outputSession)
[quadraticsvmModel, RMSEquadraticsvmModel] = quadraticsvm(inputSession,outputSession)
[cubicsvmModel, RMSEcubicsvmModel] = cubicsvm(inputSession,outputSession)
[finegaussiansvmModel, RMSEfinegaussiansvmModel] = finegaussiansvm(inputSession,outputSession)
[mediumgaussiansvmModel, RMSEmediumgaussiansvmModel] = mediumgaussiansvm(inputSession,outputSession)
[coarsgaussiansvmModel, RMSEcoarsgaussiansvmModel] = coarsgaussiansvm(inputSession,outputSession)
[boostedtreesModel, RMSEboostedtreesModel] = boostedtrees(inputSession,outputSession)
[baggedtreesModel, RSMEbaggedtreesModel] = baggedtrees(inputSession,outputSession)
[squaredexponentialgprModel, RMSEsquaredexponentialgprModel] = squaredexponentialgpr(inputSession,outputSession)
[MaternGPRModel, RMSEMaternGPRModel] =  MaternGPR(inputSession,outputSession)
[ExponentialGPRModel, RMSEExponentialGPRModel] = ExponentialGPR(inputSession,outputSession)
```

```
[RationalQuadraticGPRModel, RSMERationalQuadraticGPRModel
    ↪ ] = RationalQuadraticGPR(inputSession,outputSession
    ↪ )

% search lower RMSE
min = RMSElinearRegressionModel;
res = "linearRegressionModel";
name = "Simple Linear Regression";
if RMSEstepwiselinearRegressionModel < min
    min = RMSEstepwiselinearRegressionModel;
    res = "stepwiselinearRegressionModel";
    name = "Stepwise Linear";
end
if RMSErobustlinearRegressionModel < min
    min = RMSErobustlinearRegressionModel
    res = "robustlinearRegressionModel";
    name = "Robust Linear Regression";
end
if RMSEinteractionsLinearRegressionModel < min
    min = RMSEinteractionsLinearRegressionModel;
    res = "interactionsLinearRegressionModel";
    name = "Interactions Linear Regression";
end
if RMSEfineTreeModel < min
    min = RMSEfineTreeModel;
    res = "fineTreeModel";
    name = "Fine Tree";
end
if RMSEmediumTreeModel < min
    min = RMSEmediumTreeModel;
    res = "mediumTreeModel";
    name = "Medium Tree";
end
if RMSEcoarseTreeModel < min
    min = RMSEcoarseTreeModel;
    res = "coarseTreeModel";
    name = "Coarse Tree";
end
if RMSElinearSvmModel < min
    min = RMSElinearSvmModel;
    res = "linearSvmModel";
    name = "Linear SVM";
end
```

```
if RMSEquadraticsvmModel < min
    min = RMSEquadraticsvmModel;
    res = "quadraticsvm";
    name = "Quadratic SVM";
end
if RMSEcubicsvmModel < min
    min = RMSEcubicsvmModel;
    res = "cubicsvmModel";
    name = "Cubic SVM";
end
if RMSEfinegaussiansvmModel < min
    min = RMSEfinegaussiansvmModel;
    res = "finegaussiansvmModel";
    name = "Fine Gaussia SVM";
end
if RMSEmediumgaussiansvmModel < min
    min = RMSEmediumgaussiansvmModel;
    res = "mediumgaussiansvmModel";
    name = "Medium Gaussian SVM";
end
if RMSEcoarsgaussiansvmModel < min
    min = RMSEcoarsgaussiansvmModel;
    res = "coarsegaussiansvmModel";
    name = "Coarse Gaussian SVM";
end
if RMSEboostedtreesModel < min
    min = RMSEboostedtreesModel;
    res = "boostedtreesModel";
    name = "Boosted Tress";
end
if RSMEbaggedtreesModel < min
    min = RSMEbaggedtreesModel;
    res = "baggedtreesModel";
    name = "Bagged tress";
end
if RMSEsquaredexponentialgprModel < min
    min = RMSEsquaredexponentialgprModel;
    res = "squaredexponentialgprModel";
    name = "Squared Exponential GPR";
end
if RMSEMaternGPRModel < min
    min = RMSEMaternGPRModel;
    res = "MaternGPRModel";
```

```
    name = "Matern 5/2 GPR";
end
if RMSEExponentialGPRModel < min
    min = RMSEExponentialGPRModel;
    res = "ExponentialGPRModel";
    name = "Exponential GPR";
end
if RSMERationalQuadraticGPRModel < min
    min = RSMERationalQuadraticGPRModel;
    res = "RationalQuadraticGPRModel";
    name = "Rational Quadratic GPR";
end

disp ("==================================================
========")
disp ("best model trained : " )
disp (name)
disp ("with RMSE : ")
disp (min)

trainedModel = eval(res);
trainedModel.title = name;
trainedModel.realRMSE = min;

disp ("==================================================
========")

save (filename,'trainedModel');

writematrix(res,"export/outputModelInfo.csv");
writematrix(min,"export/outputModelInfo.csv",'WriteMode',
   ↪ 'append');
```

D.4.2 *matlab_pred.m*

```
addpath('export/matlabTrainedModel/')
load('export\matlabTrainedModel\trainedModel.mat')


% getting last rilevation for prediction
```

```
headers = {'Content-Type' 'application/json'; 'Accept' 'application/json'};
options = weboptions('HeaderFields', headers, 'Timeout', 5000);
data = struct();
download  = webwrite('http://www.riffelli.it/IEQ/API/get_last_rilevation_json.php', data, options);


% decode json and create matrix
tmp = jsondecode(download);
param = cell2mat(struct2cell(tmp)); % convert cell to matrix
param = param.';    % transposed matrix

% predict data
res = trainedModel.predictFcn(param)  % predict data

%send data to the server

uri = matlab.net.URI('http://www.riffelli.it/IEQ/API/add_prediction.php');
res = webwrite(uri,'PGCI',num2str(res),'MODEL',trainedModel.title);
```

Acknowledgments

I would like to thank all the people and researchers who made possible the achievement of my research project.
I thank all the DiSPeA Department professors and researchers: Emanuele Lattanzi, Lorenz Cuno Klopfenstein, Saverio Delpriori and especially my tutor Alessandro Bogliolo, for tips, feedback and precious suggestions.
I thank all the professors and researchers of the Philosophy Department and the Synergia research group, in particular, Pierluigi Graziani and Vincenzo Fano. I thank all my PhD colleagues, in particular Annamaria Amura, Matteo Bischi, Gioele Bigini, Christel Sirocchi and the researcher Stefano Pagnotta (from CNR - Pisa).
I thank all my friends who have been close to me during this challenging period and, in particular, Giovanni Bezicheri, Claudio Colapinto, Matteo Falcioni and Rocco Nori.
I dedicate this PhD thesis to my family, whom I thank in a special way for supporting me during these demanding years of study gave me the strength to overcome difficult moments.

Bibliography

[1] Alan Hedge. “Where are we in understanding the effects of where we are?” In: *Ergonomics* 43.7 (2000), pp. 1019–1029. ISSN: 13665847. DOI: 10.1080/001401300409198.

[2] Daniela Pasini et al. “Exploiting Internet of Things and building information modeling framework for management of cognitive buildings”. In: *2016 IEEE International Smart Cities Conference (ISC2)*. IEEE. 2016, pp. 1–6.

[3] Dimosthenis A. Sarigiannis. *Combined or multiple exposure to health stressors in indoor built environments*. Tech. rep. October. WHO, 2014, p. 82. URL: http://www.euro.who.int/__data/assets/pdf_file/0020/248600/Combined-or-multiple-exposure-to-health-stressors-in-indoor-built-environments.pdf?ua=1.

[4] Jacqueline C. Vischer. “The concept of workplace performance and its value to managers”. In: *California Management Review* 49.2 (2007). ISSN: 00081256. DOI: 10.2307/41166383.

[5] Jacqueline C. Vischer. “Towards an environmental psychology of workspace: How people are affected by environments for work”. In: *Architectural Science Review* 51.2 (2008), pp. 97–108. ISSN: 17589622. DOI: 10.3763/asre.2008.5114.

[6] Jin Hui Woo. “Towards sustainable workplaces: Effects of indoor environmental quality on occupant comfort and work performance”. PhD thesis. Faculty of Built Environment. The University of New South Wales Sydney, Australia, 2010.

[7] Bjarne W Olesen. “Indoor Environmental Input Parameters for Design and Assessment of Energy Performance of Buildings Addressing Indoor Air Quality, Thermal Environment, Lighting and Acoustics”. In: *REHVA* (2015), pp. 17–23.

[8] EN 16798-1. *Energy performance of buildings - Ventilation for buildings - Part 1: Indoor environmental input parameters for design and assessment of energy performance of buildings addressing indoor air quality, thermal environment, lighting and acoustics - Module M1.* 2019.

[9] Sandra G.L. Persiani et al. "Biometric data as real-time measure of physiological reactions to environmental stimuli in the built environment". In: *Energies* 14.1 (2021). ISSN: 19961073. DOI: 10.3390/en14010232.

[10] ISO - International Organization for Standardization. *ISO 7730:2005. Ergonomics of the thermal environment — Analytical determination and interpretation of thermal comfort using calculation of the PMV and PPD indices and local thermal comfort criteria.* 2005. URL: https://www.iso.org/standard/39155.html (visited on 09/20/2021).

[11] Elena Teodoreanu. "Thermal Comfort Index". In: *Present Environment and Sustainable Development* 10.2 (2016), pp. 105–118. DOI: 10.1515/pesd-2016-0029.

[12] Peder Wolkoff. "The mystery of dry indoor air – An overview". In: *Environment International* 121.October (2018), pp. 1058–1065. ISSN: 18736750. DOI: 10.1016/j.envint.2018.10.053. URL: https://doi.org/10.1016/j.envint.2018.10.053.

[13] Refrigerating ASHRAE - American Society of Heating and Air - Conditioning Engineers. *ASHRAE Standard 55:2020. Thermal environmental conditions for human occupancy.* 2020. URL: https://www.ashrae.org/technical-resources/bookstore/standard-55-thermal-environmental-conditions-for-human-occupancy (visited on 09/20/2021).

[14] Peder Wolkoff, Kenichi Azuma, and Paolo Carrer. "Health, work performance, and risk of infection in office-like environments: The role of indoor temperature, air humidity, and ventilation". In: *International Journal of Hygiene and Environmental Health* 233.February (2021), p. 113709. ISSN: 1618131X. DOI: 10.1016/j.ijheh.2021.113709. URL: https://doi.org/10.1016/j.ijheh.2021.113709.

[15] ASHRAE. "Ventilation and Infiltration chapter". In: *Fundamentals volume of the ASHRAE Handbook.* American Society of Heating Refrigerating and Air-Conditionin, Atlanta, Ga., 2005.

[16] The National Institute for Occupational Safety and Health (NIOSH). *Indoor Environmental Quality.* 2013.

[17] Kerry Kilpatrick. *Sick Classrooms Caused by Rising CO2 Levels.* 2021. URL: https://energyalliancegroup.org/sick-classrooms-require-energy-efficient-solutions-2/ (visited on 05/28/2021).

[18] Matthias Richter et al. "Natural building materials for interior fitting and refurbishment—what about indoor emissions?" In: *Materials* 14.1 (Jan. 2021), pp. 1–14. ISSN: 19961944. DOI: 10.3390/ma14010234. URL: https://www.mdpi.com/1996-1944/14/1/234/htm.

[19] Federica Bettarello et al. "Indoor acoustic requirements for autism-friendly spaces". In: *Applied Sciences (Switzerland)* 11.9 (2021). ISSN: 20763417. DOI: 10.3390/app11093942.

[20] Deniz Artan, Esin Ergen, and Isilay Tekce. "Acoustical comfort in office buildings". In: *Proceedings of the Annual International Conference on Architecture and Civil Engineering*. Global Science and Technology Forum, May 2019, pp. 145–149. DOI: 10.5176/2301-394X_ACE19.605.

[21] Marco Caniato, Federica Bettarello, and Andrea Gasparella. "Indoor and outdoor noise changes due to the COVID-19 lockdown and their effects on individuals' expectations and preferences". In: *Scientific Reports* 11.1 (2021), pp. 1–17. ISSN: 20452322. DOI: 10.1038/s41598-021-96098-w. URL: https://doi.org/10.1038/s41598-021-96098-w.

[22] IESNA. *Lighting handbook.* New York (USA), 2000.

[23] Wonwoo Kim and Jeong Kim. "The Scope of the Glare Light Source of the Window with Non-uniform Luminance Distribution". In: *Indoor and Built Environment - INDOOR BUILT ENVIRON* 20 (2011), pp. 54–64. DOI: 10.1177/1420326X10389269.

[24] F Cantin and M-C. Dubois. "Daylighting metrics based on illuminance, distribution, glare and directivity". In: *Lighting Research & Technology* 43.3 (2011), pp. 291–307. DOI: 10.1177/1477153510393319. URL: https://doi.org/10.1177/1477153510393319.

[25] C Sapia. "Daylighting in buildings: Developments of sunlight addressing by optical fiber". In: *Solar Energy* 89 (2013), pp. 113–121. ISSN: 0038-092X. DOI: https://doi.org/10.1016/j.solener.2012.12.003. URL: http://www.sciencedirect.com/science/article/pii/S0038092X12004185.

[26] Salvatore Carlucci et al. "A review of indices for assessing visual comfort with a view to their use in optimization processes to support building integrated design". In: *Renewable and Sustainable Energy Reviews* 47.7491 (2015), pp. 1016–1033. ISSN: 18790690. DOI: 10.1016/j.rser.2015.03.062. URL: http://dx.doi.org/10.1016/j.rser.2015.03.062.

[27] CEN - Comité européen de normalisation (European Committee for Standardization. *EN 12665:2018. Light and lighting – Basic terms and criteria for specifying lighting requirements*. 2018. URL: https://standards.iteh.ai/catalog/standards/cen/43e4dbbf-7710-4b60-9872-ad27a25a4661/en-12665-2018 (visited on 09/20/2021).

[28] Peder Skov and Ole Valbjørn. "The "sick" building syndrome in the office environment: the Danish Town Hall Study". In: *Environment international* 13.4-5 (1987), pp. 339–349. ISSN: 0160-4120.

[29] Sherwood Burge et al. "Sick building syndrome: a study of 4373 office workers". In: *The Annals of Occupational Hygiene* 31.4A (1987), pp. 493–504. ISSN: 1475-3162.

[30] Dan Norbäck, Ingegerd Michel, and John Widström. "Indoor air quality and personal factors related to the sick building syndrome". In: *Scandinavian journal of work, environment & health* (1990), pp. 121–128. ISSN: 0355-3140.

[31] Francesco Chirico. "The «Indoor Environmental Quality» as an ergonomic risk factor at workplace : Proposal for a risk assessment strategy". In: *SSRN Electronic Journal* (2021). DOI: 10.2139/ssrn.3763343.

[32] Rabee M. Reffat and Edward L. Harkness. "Environmental comfort criteria: Weighting and integration". In: *Journal of Performance of Constructed Facilities* 15.3 (2001), pp. 104–108. ISSN: 08873828. DOI: 10.1061/(ASCE)0887-3828(2001)15:3(104).

[33] Rabee M. Reffat and Edward L. Harkness. "Expert System for Environmental Quality Evaluation". In: *Journal of Performance of Constructed Facilities* 15.3 (2001), pp. 109–114. ISSN: 0887-3828. DOI: 10.1061/(asce)0887-3828(2001)15:3(109).

[34] Che Ming Chiang and Chi Ming Lai. "A study on the comprehensive indicator of indoor environment assessment for occupants' health in Taiwan". In: *Building and Environment* 37.4 (2002), pp. 387–392. ISSN: 03601323. DOI: 10.1016/S0360-1323(01)00034-8.

[35] Michael A. Humphreys. "Quantifying occupant comfort: Are combined indices of the indoor environment practicable?" In: *Building Research and Information* 33.4 (2005), pp. 317–325. ISSN: 09613218. DOI: 10.1080/09613210500161950.

[36] L. T. Wong, K. W. Mui, and P. S. Hui. "A multivariate-logistic model for acceptance of indoor environmental quality (IEQ) in offices". In: *Building and Environment* 43.1 (2008), pp. 1–6. ISSN: 03601323. DOI: 10.1016/j.buildenv.2007.01.001.

[37] Arianna Astolfi and Franco Pellerey. "Subjective and objective assessment of acoustical and overall environmental quality in secondary school classrooms". In: *The Journal of the Acoustical Society of America* 123.1 (2008), pp. 163–173. ISSN: 0001-4966. DOI: 10.1121/1.2816563.

[38] Joon Ho Choi, Azizan Aziz, and Vivian Loftness. "Decision support for improving occupant environmental satisfaction in office buildings: The relationship between sub-set of IEQ satisfaction and overall environmental satisfaction". In: *9th International Conference and Exhibition - Healthy Buildings 2009, HB 2009*. 2009.

[39] A. C.K. Lai et al. "An evaluation model for indoor environmental quality (IEQ) acceptance in residential buildings". In: *Energy and Buildings* 41.9 (2009), pp. 930–936. ISSN: 03787788. DOI: 10.1016/j.enbuild.2009.03.016.

[40] Philomena M. Bluyssen, Myriam Aries, and Paula van Dommelen. "Comfort of workers in office buildings: The European HOPE project". In: *Building and Environment* 46.1 (2011), pp. 280–288. ISSN: 03601323. DOI: 10.1016/j.buildenv.2010.07.024. URL: http://dx.doi.org/10.1016/j.buildenv.2010.07.024.

[41] Concettina Marino, Antonino Nucara, and Matilde Pietrafesa. "Proposal of comfort classification indexes suitable for both single environments and whole buildings". In: *Building and Environment* 57.December 2002 (2012), pp. 58–67. ISSN: 03601323. DOI: 10.1016/j.buildenv.2012.04.012. URL: http://dx.doi.org/10.1016/j.buildenv.2012.04.012.

[42] Bin Cao et al. "Development of a multivariate regression model for overall satisfaction in public buildings based on field studies in Beijing and Shanghai". In: *Building and Environment* 47.1 (2012), pp. 394–399. ISSN: 03601323. DOI: 10.1016/j.buildenv.2011.06.022. URL: http://dx.doi.org/10.1016/j.buildenv.2011.06.022.

[43] Matiwaza Ncube and Saffa Riffat. "Developing an indoor environment quality tool for assessment of mechanically ventilated office buildings in the UK - A preliminary study". In: *Building and Environment* 53 (2012), pp. 26–33. ISSN: 03601323. DOI: 10.1016/j.buildenv.2012.01.003. URL: http://dx.doi.org/10.1016/j.buildenv.2012.01.003.

[44] Hyojin Kim and Jeff S. Haberl. "Field-test of the new ASHRAE/-CIBSE/USGBC performance measurement protocols for commercial buildings: Basic level". In: *ASHRAE Transactions* 118.PART 1 (2012), pp. 135–142. ISSN: 00012505.

[45] Francesco Fassio, Aldo Fanchiotti, and Roberto de Lieto Vollaro. "Linear, non-linear and alternative algorithms in the correlation of IEQ factors with global comfort: A case study". In: *Sustainability (Switzerland)* 6.11 (2014), pp. 8113–8127. ISSN: 20711050. DOI: 10.3390/su6118113.

[46] Lisa Loreti et al. "Overall indoor quality of a non-renewed secondary-school building". In: *Energy Procedia* 78 (2015), pp. 3126–3131. ISSN: 18766102. DOI: 10.1016/j.egypro.2015.11.768. URL: http://dx.doi.org/10.1016/j.egypro.2015.11.768.

[47] Wenjuan Wei et al. "Review of parameters used to assess the quality of the indoor environment in Green Building certification schemes for offices and hotels". In: *Energy and Buildings* 209 (2020), p. 109683. ISSN: 03787788. DOI: 10.1016/j.enbuild.2019.109683.

[48] Marielle Ferreira Silva et al. "Post-occupancy evaluation of residential buildings in Luxembourg with centralized and decentralized ventilation systems, focusing on indoor air quality (IAQ). Assessment by questionnaires and physical measurements". In: *Energy and Buildings* 148 (Aug. 2017), pp. 119–127. ISSN: 03787788. DOI: 10.1016/j.enbuild.2017.04.049.

[49] Paola Ricciardi and Cinzia Buratti. "Environmental quality of university classrooms: Subjective and objective evaluation of the thermal, acoustic, and lighting comfort conditions". In: *Building and Environment* 127 (2018), pp. 23–36. ISSN: 03601323. DOI: 10.1016/j.buildenv.2017.10.030. URL: https://doi.org/10.1016/j.buildenv.2017.10.030.

[50] Julia K. Day et al. "Blinded by the light: Occupant perceptions and visual comfort assessments of three dynamic daylight control systems and shading strategies". In: *Building and Environment* 154 (May

2019), pp. 107–121. ISSN: 03601323. DOI: 10.1016/j.buildenv.2019.02.037.

[51] Jihye Ryu et al. "Defining the thermal sensitivity (Griffiths constant) of building occupants in the Korean residential context". In: *Energy and Buildings* 208 (Feb. 2020), p. 109648. ISSN: 03787788. DOI: 10.1016/j.enbuild.2019.109648.

[52] Stefano Riffelli. "Global Comfort Indices in Indoor Environments: A Survey". In: *Sustainability* 13.22 (2021). ISSN: 2071-1050. DOI: 10.3390/su132212784. URL: https://www.mdpi.com/2071-1050/13/22/12784.

[53] Israr Ullah and Dohyeun Kim. "An improved optimization function for maximizing user comfort with minimum energy consumption in smart homes". In: *Energies* 10.11 (2017). ISSN: 19961073. DOI: 10.3390/en10111818.

[54] A. I. Dounis and C. Caraiscos. "Advanced control systems engineering for energy and comfort management in a building environment-A review". In: *Renewable and Sustainable Energy Reviews* 13.6-7 (2009), pp. 1246–1261. ISSN: 13640321. DOI: 10.1016/j.rser.2008.09.015.

[55] *Google Scholar*. URL: https://scholar.google.com/ (visited on 09/20/2021).

[56] *ScienceDirect.com | Science, health and medical journals, full text articles and books*. URL: https://www.sciencedirect.com/ (visited on 09/20/2021).

[57] *MDPI - Publisher of Open Access Journals*. URL: https://www.mdpi.com/ (visited on 09/20/2021).

[58] *Home Feed | ResearchGate*. URL: https://www.researchgate.net/ (visited on 09/20/2021).

[59] ASHRAE-55. *Thermal environmental conditions for human occupancy*. 2013.

[60] P Ole Fanger. *Thermal comfort: Analysis and applications in environmental engineering*. Vol. 3. McGraw-Hill, 1972, p. 181. DOI: 10.1016/s0003-6870(72)80074-7.

[61] Michael A. Humphreys and J. Fergus Nicol. "Understanding the adaptive approach to thermal comfort". In: *ASHRAE Transactions* 104.Pt 1B (1998), pp. 991–1004. ISSN: 00012505.

[62] Richard J. de Dear and Gail Schiller Brager. "Developing an adaptive model of thermal comfort and preference". In: *ASHRAE Transactions* 104.Pt 1A (1998), pp. 145–167. ISSN: 00012505.

[63] G. S. Brager and R. De Dear. "A standard for natural ventilation". In: *ASHRAE Journal* 42.10 (2000), pp. 21–23+25. ISSN: 00012491.

[64] Constantinos A. Balaras, Elena Dascalaki, and Athina Gaglia. "HVAC and indoor thermal conditions in hospital operating rooms". In: *Energy and Buildings* 39.4 (2007), pp. 454–470. ISSN: 03787788. DOI: 10.1016/j.enbuild.2006.09.004.

[65] Peder Wolkoff and Søren K. Kjærgaard. "The dichotomy of relative humidity on indoor air quality". In: *Environment International* 33.6 (2007), pp. 850–857. ISSN: 18736750. DOI: 10.1016/j.envint.2007.04.004.

[66] Bruxelles: European Standardisation. *EN 15251:2007. Indoor environmental input parameters for design and assessment of energy performance of buildings addressing indoor air quality, thermal environment, lighting and acoustics*. 2007.

[67] Baharuddin Hamzah et al. "Thermal comfort analyses of secondary school students in the tropics". In: *Buildings* 8.4 (2018), pp. 1–19. ISSN: 20755309. DOI: 10.3390/buildings8040056.

[68] Evandro Eduardo Broday et al. "The approximation between thermal sensation votes (TSV) and predicted mean vote (PMV): A comparative analysis". In: *International Journal of Industrial Ergonomics* 69.August 2017 (2019), pp. 1–8. ISSN: 18728219. DOI: 10.1016/j.ergon.2018.09.007. URL: https://doi.org/10.1016/j.ergon.2018.09.007.

[69] X. Zhou et al. "Experimental study of the influence of anticipated control on human thermal sensation and thermal comfort". In: *Indoor Air* 24.2 (2014), pp. 171–177. ISSN: 09056947. DOI: 10.1111/ina.12067.

[70] Andrea Ursini Casalena. *Indici di Comfort Termico*. 2009. URL: www.mygreenbuildings.org (visited on 04/09/2021).

[71] Petra Vladykova Bednarova. *Overview of EN 15251: Part 1 – Addressing indoor air quality, thermal environment, lighting and acoustic*. 2014. URL: www.swegonairacademy.com (visited on 04/09/2021).

[72] Victor Olgyay. "Design with Climate Princeton university press". In: *Princeton, new jersey* (1969).

[73] Murray Milne and Baruch Givoni. "Architectural design based on climate in Energy Conservation Through Building Design". In: *Energy conservation through building design* (1979), pp. 96–113.

[74] J. Masterson and F.A. Richardson. *Humidex, A Method of Quantifying Human Discomfort Due to Excessive Heat and Humidity.* Environment Canada, Atmospheric Environment, 1979, p. 45.

[75] G. Agostini et al. *Bioclimatologia umana.* UTET, 2005, p. 344.

[76] George A Winterling. "Humiture—Revised and Adapted for the Summer Season in Jacksonville, Fla." In: *Bulletin of the American Meteorological Society* 60.4 (Aug. 1979), pp. 329–330. ISSN: 00030007, 15200477. URL: http://www.jstor.org/stable/26218670.

[77] Robert G Steadman. "The assessment of sultriness. Part II: effects of wind, extra radiation and barometric pressure on apparent temperature". In: *Journal of Applied Meteorology* 18.7 (1979), pp. 874–885. ISSN: 0021-8952.

[78] Earl Crabill Thom. "The discomfort index". In: *Weatherwise* 12.2 (1959), pp. 57–61. ISSN: 0043-1672.

[79] Robert G Quayle and Robert G Steadman. "The Steadman wind chill: an improvement over present scales". In: *Weather and Forecasting* 13.4 (1998), pp. 1187–1193. ISSN: 1520-0434.

[80] Jean-Pierre Besancenot. "Premières données sur les stress bioclimatiques moyens en France". In: *Annales de geographie.* JSTOR, 1974, pp. 497–530. DOI: 10.3406/geo.1974.18951.

[81] E Teodoreanu. *Bioclimatologie umană.* Editura Academiei Române, 2002. ISBN: 9789732708255. URL: https://books.google.it/books?id=Ste5rQEACAAJ.

[82] Bruxelles: European Standardisation. *EN 12354-5:2009. Building acoustics - Estimation of acoustic performance of building from the performance of elements - Part 5: Sounds levels due to the service equipment.* 2009.

[83] David A Bies, Colin H Hansen, and Carl Q Howard. *Engineering noise control.* CRC press, 2017. ISBN: 1351228153.

[84] European Committee for Standardization. *EN 12665. Light and lighting – Basic terms and criteria for specifying lighting requirements.* 2011.

[85] Wonwoo Kim and Jeong Kim. "The Scope of the Glare Light Source of the Window with Non-uniform Luminance Distribution". In: *Indoor and Built Environment - INDOOR BUILT ENVIRONMENT* 20 (2011), pp. 54–64. DOI: 10.1177/1420326X10389269.

[86] F Cantin and M-C. Dubois. "Daylighting metrics based on illuminance, distribution, glare and directivity". In: *Lighting Research & Technology* 43.3 (2011), pp. 291–307. DOI: 10.1177/1477153510393319. URL: https://doi.org/10.1177/1477153510393319.

[87] Robert W. Marans and Xiao Ying Yan. "Lighting quality and environmental satisfaction in open and enclosed offices". In: *Journal of Architectural and Planning Research* 6.2 (1989), pp. 118–131. ISSN: 07380895.

[88] K. W. Mui and W. T. Chan. "A new indoor environmental quality equation for air-conditioned buildings". In: *Architectural Science Review* 48.1 (2005), pp. 41–46. ISSN: 17589622. DOI: 10.3763/asre.2005.4806.

[89] Joseph H.K. Lai and Francis W.H. Yik. "Perceived importance of the quality of the indoor environment in commercial buildings". In: *Indoor and Built Environment* 16.4 (2007), pp. 311–321. ISSN: 1420326X. DOI: 10.1177/1420326X07080463.

[90] Joseph H K Lai and Francis W H Yik. "Perception of importance and performance of the indoor environmental quality of high-rise residential buildings". In: *Building and Environment* 44.2 (2009), pp. 352–360. ISSN: 0360-1323.

[91] David Heinzerling et al. "Indoor environmental quality assessment models: A literature review and a proposed weighting and classification scheme". In: *Building and environment* 70 (2013), pp. 210–222. ISSN: 0360-1323.

[92] Bruce D. Hunn and Jim Bochat. "Measurement of commercial building performance". In: *ASHRAE Journal* 57.1 (2015), pp. 66–71. ISSN: 00012491.

[93] Michał Piasecki, Krystyna Kostyrko, and Sławomir Pykacz. "Indoor environmental quality assessment: Part 1: Choice of the indoor environmental quality sub-component models". In: *Journal of Building Physics* 41.3 (2017), pp. 264–289. ISSN: 1744-2591.

[94] Michał Piasecki and Krystyna Barbara Kostyrko. "Indoor environmental quality assessment, part 2: Model reliability analysis". In: *Journal of Building Physics* 42.3 (2018), pp. 288–315. ISSN: 1744-2591.

[95] C. Buratti et al. "A new index combining thermal, acoustic, and visual comfort of moderate environments in temperate climates". In: *Building and Environment* 139 (2018), pp. 27–37. ISSN: 03601323. DOI: 10.1016/j.buildenv.2018.04.038. URL: https://doi.org/10.1016/j.buildenv.2018.04.038.

[96] Demetrios J. Moschandreas and Sait C. Sofuoglu. "The indoor environmental index and its relationship with symptoms of office building occupants". In: *Journal of the Air and Waste Management Association* 54.11 (2004), pp. 1440–1451. ISSN: 21622906. DOI: 10.1080/10473289.2004.10470999.

[97] Marina Laskari, Stavroula Karatasou, and Mat Santamouris. "A methodology for the determination of indoor environmental quality in residential buildings through the monitoring of fundamental environmental parameters: A proposed Dwelling Environmental Quality Index". In: *Indoor and Built Environment* 26.6 (2017), pp. 813–827. ISSN: 14230070. DOI: 10.1177/1420326X16660175.

[98] Eur Soc Hous. *ICE-WISH. Demonstrating through intelligent control (smart metering, wireless technology, cloud computing, and user-oriented display information), energy and water wastage reductions.* 2016. URL: www.ice-wish.eu.

[99] C. M. Chiang et al. "A methodology to assess the indoor environment in care centers for senior citizens". In: *Building and Environment* 36.4 (2001), pp. 561–568. ISSN: 03601323. DOI: 10.1016/S0360-1323(00)00024-X.

[100] Monika Frontczak and Pawel Wargocki. "Literature survey on how different factors influence human comfort in indoor environments". In: *Building and Environment* 46.4 (2011), pp. 922–937. ISSN: 03601323. DOI: 10.1016/j.buildenv.2010.10.021. URL: http://dx.doi.org/10.1016/j.buildenv.2010.10.021.

[101] Jungsoo Kim and Richard de Dear. "Nonlinear relationships between individual IEQ factors and overall workspace satisfaction". In: *Building and Environment* 49.1 (2012), pp. 33–40. ISSN: 03601323. DOI: 10.1016/j.buildenv.2011.09.022. URL: http://dx.doi.org/10.1016/j.buildenv.2011.09.022.

[102] Tiberiu Catalina and Vlad Iordache. "IEQ assessment on schools in the design stage". In: *Building and Environment* 49 (2012), pp. 129–140. ISSN: 0360-1323.

[103] V V Sakhare and R V Ralegaonkar. "Indoor environmental quality: Review of parameters and assessment models". In: *Architectural Science Review* 57.2 (2014), pp. 147–154. ISSN: 0003-8628.

[104] Pontip Stephen Nimlyat and Mohd Zin Kandar. "Appraisal of indoor environmental quality (IEQ) in healthcare facilities: A literature review". In: *Sustainable Cities and Society* 17 (2015), pp. 61–68. ISSN: 2210-6707.

[105] Alessia Gadotti and Rossano Albatici. "A survey of evaluation methods used for holistic comfort assessment". In: *Proceedings - 9th International Windsor Conference 2016: Making Comfort Relevant* (2016), pp. 994–1006.

[106] Pontip Stephen Nimlyat. "Indoor environmental quality performance and occupants' satisfaction [IEQPOS] as assessment criteria for green healthcare building rating". In: *Building and Environment* 144 (2018), pp. 598–610. ISSN: 0360-1323.

[107] Wonyoung Yang and Hyeun Jun Moon. "Combined effects of acoustic, thermal, and illumination conditions on the comfort of discrete senses and overall indoor environment". In: *Building and Environment* 148.November 2018 (2019), pp. 623–633. ISSN: 03601323. DOI: 10.1016/j.buildenv.2018.11.040. URL: https://doi.org/10.1016/j.buildenv.2018.11.040.

[108] Michał Piasecki. "Practical implementation of the indoor environmental quality model for the assessment of nearly zero energy single-family building". In: *Buildings* 9.10 (2019), p. 214. ISSN: 20755309. DOI: 10.3390/buildings9100214.

[109] Lasse Rohde et al. "Determining indoor environmental criteria weights through expert panels and surveys". In: *Building Research & Information* 48.4 (2020), pp. 415–428. ISSN: 0961-3218.

[110] Michał Piasecki et al. "Implementation of the Indoor Environmental Quality (IEQ) Model for the Assessment of a Retrofitted Historical Masonry Building". In: *Energies* 13.22 (2020), p. 6051.

[111] Hao Tang, Yong Ding, and Brett Singer. "Interactions and comprehensive effect of indoor environmental quality factors on occupant satisfaction". In: *Building and Environment* 167 (2020), p. 106462. ISSN: 0360-1323.

[112] Chien Fei Chen et al. "The impacts of building characteristics, social psychological and cultural factors on indoor environment quality productivity belief". In: *Building and Environment* 185.August (2020), p. 107189. ISSN: 03601323. DOI: 10.1016/j.buildenv.2020.107189. URL: https://doi.org/10.1016/j.buildenv.2020.107189.

[113] Jin Woo Moon and Jong-Jin Kim. "ANN-based thermal control models for residential buildings". In: *Building and Environment* 45.7 (2010), pp. 1612–1625. ISSN: 0360-1323.

[114] Türkan Göksal Özbalta, Alper Sezer, and Yusuf Yıldız. "Models for prediction of daily mean indoor temperature and relative humidity: education building in Izmir, Turkey". In: *Indoor and Built Environment* 21.6 (2012), pp. 772–781. ISSN: 1420-326X.

[115] Jin Woo Moon, Sung-Hoon Yoon, and Sooyoung Kim. "Development of an artificial neural network model based thermal control logic for double skin envelopes in winter". In: *Building and Environment* 61 (2013), pp. 149–159. ISSN: 0360-1323.

[116] Arya Ashtiani, Parham A Mirzaei, and Fariborz Haghighat. "Indoor thermal condition in urban heat island: Comparison of the artificial neural network and regression methods prediction". In: *Energy and buildings* 76 (2014), pp. 597–604. ISSN: 0378-7788.

[117] Jin Woo Moon and Sung Kwon Jung. "Algorithm for optimal application of the setback moment in the heating season using an artificial neural network model". In: *Energy and Buildings* 127 (2016), pp. 859–869. ISSN: 0378-7788.

[118] Leopold Mba, Pierre Meukam, and Alexis Kemajou. "Application of artificial neural network for predicting hourly indoor air temperature and relative humidity in modern building in humid region". In: *Energy and Buildings* 121 (2016), pp. 32–42. ISSN: 0378-7788.

[119] Jin Woo Moon. "Integrated control of the cooling system and surface openings using the artificial neural networks". In: *Applied Thermal Engineering* 78 (2015), pp. 150–161. ISSN: 1359-4311.

[120] G Mustafaraj, J Chen, and G Lowry. "Thermal behaviour prediction utilizing artificial neural networks for an open office". In: *Applied Mathematical Modelling* 34.11 (2010), pp. 3216–3230. ISSN: 0307-904X.

[121] Giorgio Mustafaraj, Gordon Lowry, and Jie Chen. “Prediction of room temperature and relative humidity by autoregressive linear and nonlinear neural network models for an open office”. In: *Energy and Buildings* 43.6 (2011), pp. 1452–1460. ISSN: 0378-7788.

[122] Surat Atthajariyakul and Thananchai Leephakpreeda. “Neural computing thermal comfort index for HVAC systems”. In: *Energy conversion and management* 46.15-16 (2005), pp. 2553–2565. ISSN: 0196-8904.

[123] Jin Woo Moon. “Performance of ANN-based predictive and adaptive thermal-control methods for disturbances in and around residential buildings”. In: *Building and Environment* 48 (2012), pp. 15–26. ISSN: 0360-1323.

[124] Chengli Li, Qihui Zhang, and Tingting Mou. “The study of neural network in the application of pmv index”. In: *2010 International Conference on System Science, Engineering Design and Manufacturing Informatization*. Vol. 1. IEEE, 2010, pp. 289–292. ISBN: 1424486645.

[125] M Castilla et al. “Neural network and polynomial approximated thermal comfort models for HVAC systems”. In: *Building and Environment* 59 (2013), pp. 107–115. ISSN: 0360-1323.

[126] C Buratti, M Vergoni, and D Palladino. “Thermal comfort evaluation within non-residential environments: development of Artificial Neural Network by using the adaptive approach data”. In: *Energy Procedia* 78 (2015), pp. 2875–2880. ISSN: 1876-6102.

[127] Jörn von Grabe. “Potential of artificial neural networks to predict thermal sensation votes”. In: *Applied energy* 161 (2016), pp. 412–424. ISSN: 0306-2619.

[128] António E Ruano and Pedro M Ferreira. “Neural network based hvac predictive control”. In: *IFAC Proceedings Volumes* 47.3 (2014), pp. 3617–3622. ISSN: 1474-6670.

[129] Tanaya Chaudhuri et al. “Random forest based thermal comfort prediction from gender-specific physiological parameters using wearable sensing technology”. In: *Energy and Buildings* 166 (2018), pp. 391–406. ISSN: 0378-7788.

[130] Zi Wang et al. “Predicting older people’s thermal sensation in building environment through a machine learning approach: Modelling, interpretation, and application”. In: *Building and Environment* 161 (2019), p. 106231. ISSN: 0360-1323.

[131] Seyed Masoud Sajjadian, Mina Jafari, and Peer Olaf Siebers. "An artificial intelligence method for comfort level prediction". In: *Smart Innovation, Systems and Technologies*. Vol. 131. Springer, 2019, pp. 169–177. ISBN: 9783030042929. DOI: 10.1007/978-3-030-04293-6_17.

[132] *Indoor Environment Sensor | Metriful*. URL: https://www.metriful.com/ms430 (visited on 09/20/2021).

[133] *Okdo Air Quality Kit - OKdo*. URL: https://www.okdo.com/project/okdo-air-quality-kit/ (visited on 09/20/2021).

[134] *Enviro for Raspberry Pi - Pimoroni*. URL: https://shop.pimoroni.com/products/enviro (visited on 09/20/2021).

[135] *The Engineering ToolBox: Decibel A, B and C*. URL: https://www.engineeringtoolbox.com/decibel-d_59.html (visited on 09/20/2021).

[136] ISO - International Organization for Standardization. *ISO 7726:1998 - Ergonomics of the thermal environment — Instruments for measuring physical quantities*. 1998. URL: https://www.iso.org/standard/14562.html (visited on 09/20/2021).

[137] *CBE Thermal Comfort Tool*. URL: https://comfort.cbe.berkeley.edu/ (visited on 09/20/2021).

[138] *Raspberry Pi 3 Model B+ – Raspberry Pi*. URL: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/ (visited on 09/20/2021).

[139] *K-30 - CO2 Sensor, Datasheet*. URL: http://co2meters.com/Documentation/Datasheets/DS_SE_0118_CM_0024_Revised9%20(1).pdf (visited on 09/30/2021).

[140] *BME280, Datasheet*. URL: https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bme280-ds002.pdf (visited on 09/20/2021).

[141] *LTR-559, Datasheet*. URL: https://optoelectronics.liteon.com/upload/download/ds86-2013-0003/ltr-559als-01_ds_v1.pdf (visited on 09/20/2021).

[142] *Enviroplus at Raspberry Pi GPIO Pinout*. URL: https://pinout.xyz/pinout/1_wire (visited on 09/20/2021).

[143] *Pimoroni Enviro+*. URL: https://learn.pimoroni.com/article/getting-started-with-enviro-plus (visited on 09/20/2021).

[144] *MiCS-6814, Datasheet*. URL: https://www.sgxsensortech.com/content/uploads/2015/02/1143_Datasheet-MiCS-6814-rev-8.pdf (visited on 09/20/2021).

[145] *VLIKE 6708 LCD Digital Audio Decibel Sound Level Meter.* URL: https://www.amazon.com/VLIKE-Digital-Measurement-Measuring-Function/dp/B01N2RLJ32 (visited on 09/20/2021).

[146] *Specifications for Netatmo Weather Station.* URL: https://www.netatmo.com/en-gb/weather/weatherstation/specifications (visited on 09/20/2021).

[147] *ThermoPro TP-53, Manual.* URL: https://buythermopro.com/wp-content/uploads/2018/06/Thermopro-DE-EN-FR-IT-ES-TP-53-instruction-manual-20180424.pdf (visited on 09/20/2021).

[148] *Weather API - OpenWeatherMap.* URL: https://openweathermap.org/api (visited on 09/20/2021).

[149] *SPSS Software | IBM.* URL: https://www.ibm.com/analytics/spss-statistics-software (visited on 09/20/2021).

[150] *MATLAB | MathWorks.* URL: https://www.mathworks.com/products/matlab.html (visited on 09/20/2021).

[151] Sanem Kabadayi, Adam Pridgen, and Christine Julien. "Virtual sensors: Abstracting data from physical sensors". In: *Proceedings - WoWMoM 2006: 2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks.* Vol. 2006. IEEE, 2006, pp. 587–592. ISBN: 0769525938. DOI: 10.1109/WOWMOM.2006.115.

[152] Pervez Hameed Shaikh et al. "A review on optimized control systems for building energy and comfort management of smart sustainable buildings". In: *Renewable and Sustainable Energy Reviews* 34 (2014), pp. 409–429. ISSN: 13640321. DOI: 10.1016/j.rser.2014.03.027. URL: http://dx.doi.org/10.1016/j.rser.2014.03.027.

[153] Ken Parsons. "Design of the indoor environment". In: *Design and Management of Sustainable Built Environments.* Vol. 9781447147817. Springer, 2013, pp. 157–177. ISBN: 9781447147817. DOI: 10.1007/978-1-4471-4781-7_9.

[154] *Smart Indoor Air Quality Monitor | Netatmo.* URL: https://www.netatmo.com/en-eu/aircare/homecoach (visited on 09/20/2021).

[155] *Airthings Wave Plus | Smart radon and indoor air quality monitor.* URL: https://www.airthings.com/en/wave-plus (visited on 09/20/2021).

[156] *klimaaktiv - climate protection in Austria, klimaaktiv EN.* URL: https://www.klimaaktiv.at/english/ (visited on 09/20/2021).

[157] *LiderA – Sustainable assessment system.* URL: http://www.lidera.info/index.aspx?p=index&RegionId=3&Culture=en (visited on 09/20/2021).

[158] *NABERS International | NABERS.* URL: https://www.nabers.gov.au/about/nabers-international (visited on 09/20/2021).

[159] Silvia Vilčeková et al. "Investigation of indoor air quality in houses of Macedonia". In: *International Journal of Environmental Research and Public Health* 14.1 (Jan. 2017), p. 37. ISSN: 16604601. DOI: 10.3390/ijerph14010037. URL: https://www.mdpi.com/1660-4601/14/1/37/htm.

[160] Jesica Fernández-Agüera et al. "TVOCs and PM 2.5 in naturally ventilated homes: Three case studies in a mild climate". In: *Sustainability (Switzerland)* 11.22 (Nov. 2019), p. 6225. ISSN: 20711050. DOI: 10.3390/su11226225. URL: https://www.mdpi.com/2071-1050/11/22/6225/htm.

[161] Samuel Domínguez-amarillo et al. "Bad air can also kill: Residential indoor air quality and pollutant exposure risk during the covid-19 crisis". In: *International Journal of Environmental Research and Public Health* 17.19 (Sept. 2020), pp. 1–34. ISSN: 16604601. DOI: 10.3390/ijerph17197183. URL: https://www.mdpi.com/1660-4601/17/19/7183/htm.

[162] Michael S. Breen et al. "Integrating personal air sensor and gps to determine microenvironment-specific exposures to volatile organic compounds". In: *Sensors* 21.16 (Aug. 2021), p. 5659. ISSN: 14248220. DOI: 10.3390/s21165659. URL: https://www.mdpi.com/1424-8220/21/16/5659/htm.