UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

HUMAN-IST
HUMAN CENTERED
INTERACTION
SCIENCE
& TECHNOLOGY

Relationship between subjective comfort perception and smartphone sensor data

Roman Küpper[1]

January 3, 2016

Department of Informatics - Master Project Report

Département d'Informatique - Departement für Informatik • Université de Fribourg -
Universität Freiburg • Boulevard de Pérolles 90 • 1700 Fribourg • Switzerland

phone +41 (26) 300 84 65   fax +41 (26) 300 97 31   Diuf-secr-pe@unifr.ch   http://diuf.unifr.ch

[1] romanrick.kuepper@unifr.ch, Human-IST, DIUF, University of Fribourg

Abstract

We evaluated if a smartphone based sensing application is able to collect comfort related data and how the notification strategy influences its results. For this reason we built a hybrid smartphone application that is able to access the microphone and the ambient light sensor and collect user related information at the same time. The application is part of a system that is able to notify the user of the application through push notifications and save the collected data into a central database. The evaluation, of a two-day lasting experiment with twelve participants, showed that the system is able to collect comfort related data with an adequate quality. We could further show that there are significant quality differences depending on the notification strategy.

Keywords: Master thesis report, Human-IST Research Institute, Human comfort

Acknowledgements

I would like to thank:

Prof. Dr. Denis Lalanne for giving me the opportunity to work on this project and his helpful ideas.

Dr. Julien Nembrini for his technical advice, his motivation and thoughtful inputs in our numerous meetings.

Léonard Stalder for the inspiring dialogues and his help with the server implementation.

My Family for their continuous support through my academic career and their motivation during this project. Also my girlfriend and my close friends for listening and supporting me in the last few month.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

One of the key factors that determines human productivity and happiness is the personal level of comfort[12][40]. A person that feels comfortable works more efficiently and is overall happier than a person who is distressed[12]. But what causes someone to feel comfortable? Besides interpersonal relations and health related issues, one of the most important factors is the surroundings. In developed countries, the surroundings mostly consist of private indoor environments and offices[21]. With the wider distribution of HVAC[1] and lighting systems most of the factors that determine human comfort can be regulated. The regulations are made based on standards like the ASHRAE 55[10] and statistical analysis[24]. But even the most advanced HVAC systems are only as good as the information they get.

Sensing information about the surroundings is mostly done with the help of sensor boxes which are distributed over the location of interest[32]. This has two major disadvantages. The first one is the cost of this sensors. Producing or buying them at a large quantity is expensive. And the second one is the location of these sensors inside a room. Especially in large office-buildings it is difficult to place the sensors at the right spot[62]. A possible solution to both of this issues is the usage of smartphones respectively their sensors. This approach has the advantage that nearly all users already have their own personal sensor. At least in the industrialized countries, most of the people already own smartphones. Also are smartphones much closer to the person of interest. This could allow the HVAC system to gather much more precise information about the indoor environments and therefore to alter the conditions in a more comfortable direction.

Better regulation not only helps the people living or working inside the building but also the environment[17]. Studies have shown that in the United States 43 percent of the energy consumed in commercial buildings is used for heating, cooling and ventilation[31]. Mobile phones and their sensors could be used to build better sensing infrastructures at a larger scale and help to collect much more data and thereby save energy through better regulation. With the technical progress of modern mobile phones and their, sensors this approach of using smartphones as sensors could become even more important in the next few years. Connecting the perceived comfort with the subjective information gathered by a sensor could help to improve modern comfort prediction models and our understanding of human comfort in general.

1.2 Project Goals

With this project we try to connect purely sensor-based data with user-specific preferences. Modern smartphones come shipped with a lot of powerful sensors that could be used to gather subjective information on the surroundings while the user can enter his objective comfort perception through an graphical interface. The aim of this project is to investigate if there exists a relationship and if user and smartphone data could be used to collect information on human comfort.

The project has two major goals. One is the investigation of possibilities and challenges of a smartphone based sensing application. Here it should be examined how good the sensor data represents the perceived level of comfort. The second goal is an analysis of different notification approaches and their effect on the collected data and user satisfaction. This part of the project more aims on the user side and tries to find strategies that enhance the information exchange with the user.

To achieve this goals, we have built an application that is capable of collecting objective data from the sensors together with subjective information entered by the user. The subjective data is collected through a graphical user interface on a smartphone. The aim hereby was to get as much information as possible without interrupting the user more than necessary.

[1] Heating, Ventilation and Air Conditioning

While collecting the information from the user, the application accesses the build-in sensors of the smartphone. It was essential to collect data from the luminosity sensor and the noise sensor. These two are the most important values in terms of comfort that a smartphone can sense.

The user- and sensordata should be sent to a server where it gets saved into a database. To give the user an incentive to answer the survey, the data is directly visualized after it has been entered. In the last iteration the application is then tested in a group of twelve participants over two days. The test should give insights on the following variables:

- Quality of answers - correlation with sensor data
- Answer rate - depending on push strategy
- Quality of answers depending on the push strategy
- How disturbing is this kind of testing - survey evaluation

Furthermore the application should be able to work together with another project done by the Human-IST, in which an Inclusive Sensor System was built to compute human comfort based on the Predicted Mean Vote, the level of illumination, the noise rate scale and the CO_2 level. The data captured here should be compatible to the work of the other project and should enhance the thereby collected data with subjective user given input.

1.3 Structure

The report is structured into three parts. It starts with a chapter that explains the state of the art of the research on topics that are important for the framework of the project. It is itself divided into the sections human comfort, the experience sampling method, mobile sensing and push notifications. The section on human comfort explains the theory behind the four most important factors that influence comfort. The sections on experience sampling and mobile sensing describe how user and sensor related data can be collected with mobile devices and how the techniques evolved. The last section of the first part then ends with a theoretical foundation on the push notification approach.

The second part gives insights on how the whole system was built and designed. It starts with the description of the techniques used to build the application itself and then explains how the data is collected and stored.

In the last part, it is shown how we tested the system in an experiment and how we analyzed the collected data. We also give suggestions on how the project can be used in combination with external sensors to provide even more powerful solutions.

2 State of the Art

2.1 Human Comfort

Human comfort is a term that is widely used without having an adequate definition. The environment in which a human feels comfortable highly depends on factors that are fastly fluctuating and hard to grasp. So the easiest way to define it is probably through its opposite:

> ”Comfort is best defined as the absence of discomfort. People feel uncomfortable when they are too hot or too cold, or when the air is odorous and stale. Positive comfort conditions are those that do not distract by causing unpleasant sensations of temperature, drafts, humidity, or other aspects of the environment. Ideally, in a properly conditioned space, people should not be aware of equipment noise, heat, or air motion.” - [14]

The quote also shows another aspect of human comfort. There isn’t such thing as comfort in general. To be able to properly talk about comfort, the term needs to be further divided. Most of the studies on this matter divide it into thermal, visual, acoustic, and hygienic[2] comfort[13][20][40][47]. These different dimensions of comfort will be further discussed in the following chapter.

2.1.1 Thermal

As shown by Frontczak, Andersen and Wargocki [20], for most individuals all of the above mentioned dimensions of comfort are almost equally important. Nevertheless thermal comfort has a slightly higher importance than the others.

The thermal comfort range of humans is much narrower then the range of outside temperatures[13]. This makes it necessary for human beings to regulate the body heat through movement or transpiration if the difference between environmental temperature and optimal internal temperature gets too big. This disparity of temperature is perceived as uncomfortable.

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

Table 1: ASHRAE Thermal Sensation Scale

Computing the perfect thermal environment is difficult, since it depends on several internal and external factors. Surface temperature, relative humidity, metabolic rate, air velocity and clothing level are only a few factors that determine if someone feels hot or cold. The factors that influence human thermal comfort can be divided into personal and environmental factors. Metabolic rate and clothing level belong to the personal factors while air temperature, mean radiant temperature, operative temperature, air speed and relative humidity are counted among the environmental factors.

The most recognized model when it comes to discussing thermal comfort is the Predicted Mean Vote(PMV) developed by Fanger[19] in 1970. It uses Air temperature, Mean radiant temperature, Air speed, Humidity, Metabolic rate and Clothing level to predict the mean vote on comfort or the percentage of discontent people[19]. To collect information on the metabolic rate, Fanger [19]

[2] Hygienic comfort mostly investigates the impacts of different air qualities.

used a seven-point activity scale as seen in figure 2.

The PMV model was developed through studies in which the researchers adjusted the thermal conditions and simultaneously asked the participants how they felt[15]. To measure the subjective comfort, the seven-point ASHRAE thermal sensation scale as in Table 1 has been developed.

Activity	Met	Wm^{-2}
Lying down	0.8	47
Seated quietly	1.0	58
Sedentary activity(office, home, laboratory, school)	1.2	70
Standing, relaxed	1.2	70
Light activity, standing(shopping, laboratory, light industrie)	1.6	93
Medium activity, standing(shop assistant, domestic work, machine work)	2.0	116
High activity(heavy machine work, garage work)	3.0	175

Table 2: Activity scale by Fanger. Source: [19]

The PMV model by Fanger has a number of disadvantages and has been adapted multiple times in the last decades. One of the major drawbacks is the fact that it has been designed only for the application in fully controlled and purely ventilated buildings. Also it is not based on field studies. Fanger used climate chambers to test different climatic states on a set of test subjects[25].

2.1.2 Visual

Visual comfort plays another important role in the field of human comfort. Of the diverse fields covered by the term visual comfort, the one that interests us the most is lighting. Most indoor locations are illuminated with artificial light and lighting in general accounts for large percentages of the total energy consumption of buildings[41]. Also it can be measured with existing sensors in contrast to visual comfort given by interior design for example.

Illumination can have different effects on the comfort and well-being of a person. Experiments on the relationship between comfort and lighting have shown that illumination has a significant influence on our productivity and mood[35]. Knez[37] has demonstrated that proper lighting of an office space has a positive impact on the mood, an enhances the performance and leads to better problem solving abilities.

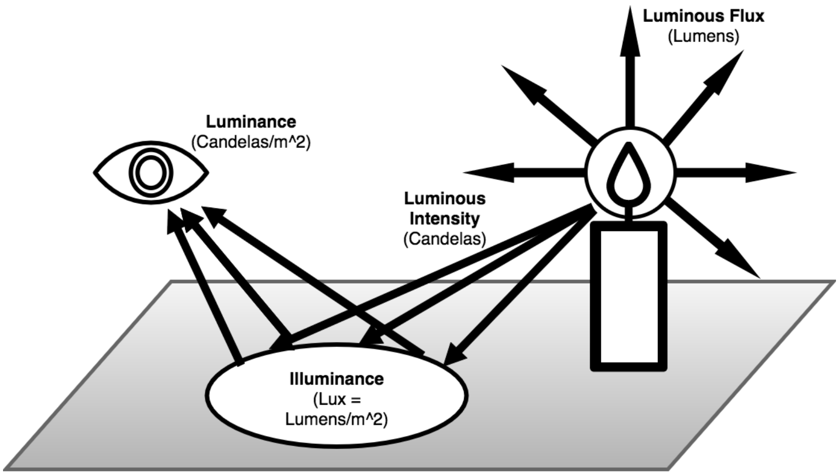


Figure 1: Basic metrics of light.[3]

When it comes to illumination, it is important to exactly distinct different forms of light. A light bulb with a certain number of Watts does not implicit a proper lighting of someones desk. Figure 1 shows the relationship between luminance, illuminance and luminous intensity. Luminous flux is the amount of light that a single lightsource produces and is measured in Watt or Lumen. 60 Watts equates to about 850 Lumens.

The intensity of the light is specified in Candelas and represents the amount of light that travels in a certain direction in the three-dimensional space.

Illuminace specifies the light that falls onto a surface and is measured in Lux or in Foot-candles in English speaking countries. When it comes to measuring visual comfort, this is the most common unit. Table 3 shows recommendations made by the Illuminating Engineering Society for different activities. This table gives an overview about which tasks require what amount of Lux[65].

Activity	**Lux**
Public spaces with dark surroundings	20-30-50
Simple orientation for short temporary visits	50-75-100
Working spaces where visual tasks are only occasionally performed	100-150-200
Performance of visual tasks of high contrast or large size	200-300-500
Performance of visual tasks of medium contrast or small size	500-750-1000
Performance of visual tasks of low contrast or very sm size	1000-1500-2000
Performance of visual tasks of low contrast or very sm size over a prolonged period	2000-3000-5000
Performance of very prolonged and exacting visual tasks	5000-7500-10000
Performance of very special visual tasks of extremely low contrast	10000-15000-20000

Table 3: Illuminating Engineering Society illuminance recommendations. Source: [65]

2.1.3 Acoustic

When trying to build comfortable surroundings, office planners and building designers often seem to forget the importance of acoustic comfort while concentrating on the visual appearance[47]. The influence of the acoustical environment on productivity and happiness seems to be often overlooked even for modern houses and office buildings.

Paradis [47] lists the following three items as the most common origins of noise problems:

- Too much noise outside the building entering the space
- Too much noise from adjacent spaces, and
- Lack of sound control in the space itself.

Too much noise from the outside is especially problematic in combination with badly ventilated and cooled indoor spaces. When the inhabitants have to open the windows in order to get cool and fresh air, the sound intensity even rises. But also noise from adjacent spaces is responsible for people to feel uncomfortable. This could happen when the noise isolation between rooms is bad. Another important reason is the lack of control. Lacking control is one of the most important reasons for feeling uncomfortable in general. Having control over the own environment can be seen as a factor as important as the physical environment itself[42].

But the right noise level also depends on the surroundings and should be adapted on the current occupancy. The HVAC system's noise level should be adapted to the surroundings. If the sound level is too low, unwanted conversations and noises from the outside can influence a persons

[3] Source: `http://sustainabilityworkshop.autodesk.com/buildings/measuring-light-levels`

productivity[47].

When speaking of noise we have to distinguish between the Sound Pressure Level(SPL), that can be measured by technical devices and the experienced loudness by human beings. The SPL can be measured in Decibel(dB) while the psychoacoustic loudness is measured in Phon. Figure 2 shows the relationship between sound pressure level, frequency and the perceived sound level in Phon.

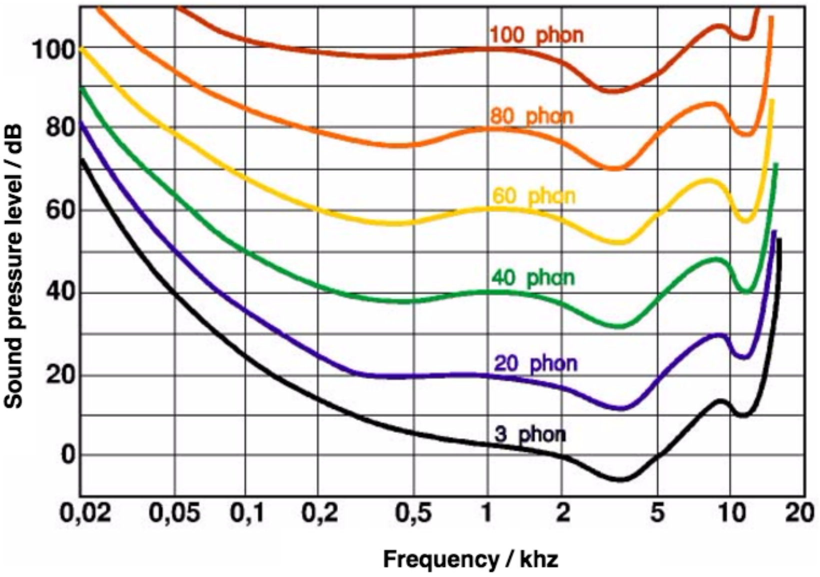


Figure 2: Relationship between SPL, Frequency and Phon. Source: [11].

Figure 2 shows that a noise with the same sound pressure level but with a different frequency can be felt to have the same loudness. A surrounding with noises of a very low or high frequency can be felt as really silent even though really high decibel values are measured[48]. But sound level and sound level pressure are not the only factors that influence the perceived loudness of a noise source. Also the difference between the current and the prior sound volume have an influence. Even quite tones can be felt as really loud when coming from an absolute silent environment. Certain noises like hammering or squeaking are perceived as louder than they actually are.

As a rule of thumb, it can be said that an increase or decrease of ten decibel results in a perceived sound level that is double respectively half as high as before. But changes in the SPL at a really high decibel value are perceived as much higher than in low ranges. Meaning that an one decibel increase from 100 to 101 dB is felt as a much bigger increase than from ten decibel to eleven.

2.1.4 Hygienic

Many studies have shown the importance of clean, temperate and healthy air[56]. Even though human beings are not capable of sensing the air quality directly, it has a crucial effect on our level of comfort. Working in a badly ventilated room for a longer period can lead to headaches and sickness[61].

There exists no sensor to sense air quality or CO_2 in modern phones nor is it planned to be implemented by one of the major smartphone manufacturers. We do not sense hygienic factors in our system. But due their important influence on comfort and they should be listed anyways. Collecting of data on this matter could be possible through external sensors like shown in chapter 10.

2.2 Experience Sampling Methods

2.2.1 Traditional

Experience sampling method(ESM) is a research methodology that is used to study human behavior[49]. With it's origins coming from Psychology, it is nowadays widely used among almost every field of research. Thanks to modern technology, ESM-based studies are nowadays easier to conduct than ever[55]. But even without the support of advanced hard- and software, ESM has interested various researchers for decades although it has been costly, time-consuming and difficult to conduct[55].

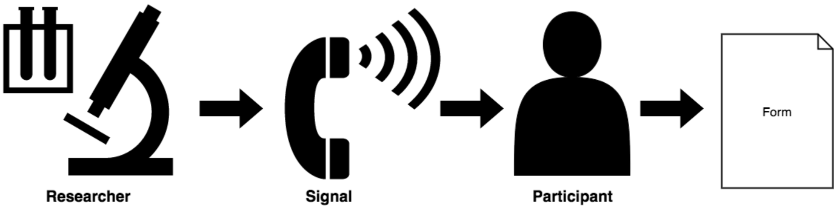


Figure 3: The diagram shows the traditional ESM notification approach

Figure 3 displays how the notification process works in the traditional way. The researcher decides when a signal is sent. He then triggers the signal which is sent to the participant and signaled with a pager, mobile phone or similar. The participant then manually fills out a form[55]. This is uncomfortable for the researcher who has to set the timer, send the signal and evaluate the form. And also for the participant who has to keep both - the timer and the form within reach.

ESM uses a technique of interrupting the participant in his daily routine and asking predefined questions. The participant answers right away or at least as fast as possible. In the traditional ESM approach, the notification of the participant happens via a pager, alarm clock or similar. When notified, the user has to fill out a series of predefined questions or write down his feelings or opinion on something. In recent years, even SMS were used as triggers[27].

2.2.2 mESM

The evolution of mobile-phones and the arise of smartphones made the execution of experience sampling for researchers and participants much easier[55]. It was characteristic for early ESM studies that they needed special devices[49]. Even the introduction to PDA's made it necessary for most of the participants to carry an additional device with them.

This situation rapidly changed with the introduction of the iPhone in 2007 and the rapid distribution of smartphones since then and lead to huge changes in ESM. Study design, data collection, notification and even the recruitment of participants changed in a radical way and can be summarized under the term of mobile Experience Sampling Method(mESM)[49].

Mobile Experience Sampling allows the researcher to conduct the survey directly on the participant's smartphone. There is no more need for a signaling device or a paper-based form. This has numerous advantages. Smartphones are already part of the normal life for most of the people. So the burden of carrying a special device and using it in public becomes redundant[49].

The mESM process is shown in Figure 4. The diagram shows that the smartphone is now in the center of the whole process. The researcher builds a mobile application on the computer and distributes it to the users smartphone. This can happen via email, and application stores or adhoc. The researcher can then send out the notifications via a computer at a given time or even send them out depending on signals sensed by the sensors of the smartphone[4]. The participant then

[4] The so-called event-contingent sampling isn't part of this paper

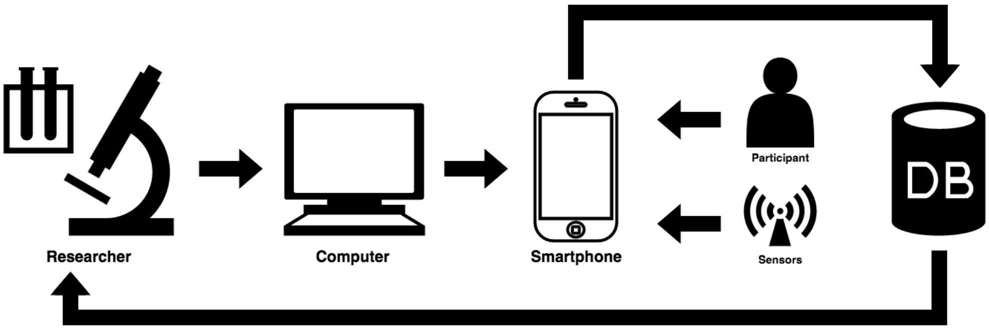


Figure 4: The diagram shows a general mESM approach

answers to the signals directly on his smartphone. This approach is known as signal-contingent sampling[53]. The various built-in sensors in modern smartphones make it possible to perceive information from the context of the user[49]. This leads to a whole new dimension of data.

The information collected by the smartphone is then sent, in intervals or at once, to a centralized database which can directly be accessed by the researcher. The process of transliterating is already done by the user.

2.2.3 Strengths and Weaknesses

Before conduction an ESM based studies, it is important to analyze its strengths and weaknesses. The ESM technique offers some unique methodological advantages over other research methods like questionnaires and interviews. In comparison to these retrospectively reported approaches, ESM allows to study behavior right at the spot[53]. The person of interest doesn't need to be in a lab-like environment[49]. And compared to observation based studies, the researcher doesn't need to follow the person of interest. This is especially useful in longitudinal studies where participants are surveyed over more than 24 hours.

Thanks to this comparatively direct method of observation, it is possible to gain deep insights of the participant and his surroundings. Technical advances and especially the wide distribution of smartphones made it possible that today, ESM studies can be conducted with comparatively little effort[55].

Besides all its benefits, when it comes to experience sampling, it is important to also consider the possible downsides. ESM requires the participants to actively take part in the surveying process. The longer the study goes, the harder it gets for the participants to follow the instructions. One of the pillars of this survey method is interrupting the user. And being interrupted multiple times a day over a longer period is something that even the most motivated people cannot stand very long. This could lead to an over- or under-representation of certain types of individuals[53].

Another possible issue is the fatigue-effect. Even if it takes only a minute to answer the survey, the total amount of time spent with answering can quickly grow to a decent number[55]. This could lead to a worsening of the quality of answers.

Finally, repeated measurement can cause reactivity effects[53]. This means that the fact that you know you are surveyed, and in case of ESM even are constantly reminded of this, your perception changes. In the framework of our experiment this could mean that participants start to pay more attention to their surroundings and personal comfort. In case of the push notifications, it is possible that they know when they usually get push notifications and start anticipating the interruption.

The strengths and weaknesses shown in this chapter lead us to the following conclusions:

- Number of participants should be large enough
- Provide an incentive for users to take part
- Make the survey as fast and easy to answer as possible
- Unobtrusive way of interrupting the user
- Different push strategies

This helps us to conduct the survey and evaluate the project. The results can be found in section 7.

2.3 Mobile Sensing

While Experience Sampling is more interested in the role of the user in the data mining process, Mobile Sensing mainly deals with the technological aspects. Thanks to the huge progress in mobile computing, it is easier than ever to collect large amounts of data with mobile sensing techniques. This progress has an impact on various fields of research and economy.

> Mobile phone sensing is an emerging area of interest for researchers as smart phones are becoming the core communication device in peoples everyday lives. Sensor enabled mobile phones or smart phones are hovering to be at the center of a next revolution in social networks, green applications, global environmental monitoring, personal and community healthcare, sensor augmented gaming, virtual reality and smart transportation systems. - [36]

Modern phones are shipped with a growing number of built-in sensors. Besides camera and microphone, sensors to capture movement, the global position and even the air pressure are widely spread among smartphones even in the low-cost segment. In recent years, most of the notable manufacturers of these phones ship them with additional software. This software allows the user to keep track of their position and even captures health related data and presents it in an understandable way[8] [23].

Collecting data just to track personal progress is known under the term of*personal sensing*[39]. It differs from the so called *community sensing* approach where a large group of participants is needed to collect meaningful information[39].

2.3.1 Special Devices

Prior to the introduction of the iPhone and smartphones in general, mobile sensing was done with special devices that had to be carried around by the participants. Even studies conducted in 2008, like the Mobile Sensing Platform project[18], proposed the development of external devices. These devices were shipped with sensors that every state-of-the-art smartphone nowadays is equipped with by default.

Lately, special devices have mainly been used to enhance the performance of the smartphone in terms of its sensing capabilities or to provide supplementary data through additional sensors. This allows to enhance the capabilities of the smartphone. Especially health related data often needs additional hardware to make meaningful predictions.

2.3.2 Smartphones

Sensor	iPhone 6S	Galaxy S6
Camera(multiple)	✓	✓
Microphone(multiple)	✓	✓
Three-axis gyro	✓	✓
Accelerometer	✓	✓
GPS	✓	✓
Proximity sensor	✓	✓
Ambient light sensor	✓	✓
Barometer	✓	✓
Finger scanner	✓	✓
Magnometer	✓	✗
Heart Rate	✗	✓
Hall sensor	✗	✓

Table 4: Comparison of different smartphone sensors. Source: [7], [54]

When talking about mobile sensing, it i important to specify about what context we speak. Since the mobile sensing we are interested in mostly takes place on smartphones, it is important how the term "smartphone" is exactly defined. The Oxford Dictionary defines it as:

> "A mobile phone that performs many of the functions of a computer, typically having a touchscreen interface, Internet access, and an operating system capable of running downloaded apps." - [16]

So modern smartphones are much more than enhanced mobile phones. They have a lot of similarities to personal computers, but fit into a person's pocket. This allows them to be carried with to almost every location. But that's not all. Besides their large touchscreens and fast access to mobile internet, most of the modern smartphones have several built-in sensors. These sensors allow the phone to be aware of its surroundings and computer scientists to access huge amounts of data.

Table 4 shows a compilation of the built-in sensors for the Apple iPhone 6S and the Samsung Galaxy S6. Both of them are the current "flagship-models" of their manufacturers who are respectively the most successful vendors of iOS and Android smartphones worldwide[29]. But not all of these sensors can be accessed through public APIs. During our project we encountered that the ambient light sensor of the Apple iPhone is not available in a public API. But at least for Apple there exists no official documentation that states if a sensor is accessible through an API or not.

2.4 Push Notifications

In the following section, we are going to take a look at the Apple Push Notification Service(APNs). Google offers a similar service for Android devices called Google Cloud Messaging[22] which we won't discuss in detail, since the general approach is the same for both. APNs allows developers to send custom message to a certain phones. These messages are known under the term push notification and are broadly used in all different kinds of applications.

2.4.1 Local and Remote Notifications

Notifications on a smartphone can be divided into local and remote. Even though there are a number of similarities, the process of triggering the notification is fundamentally different. While local notifications are scheduled and sent by the app itself, remote notification are sent by the APNs and pushed to the device from the outside[3]. The term *push notifications* usually refers to remote notifications. Both kinds of notifications are used to notify the user if a certain event has occurred. This is necessary if the users has the phone locked or if another application is in the foreground.

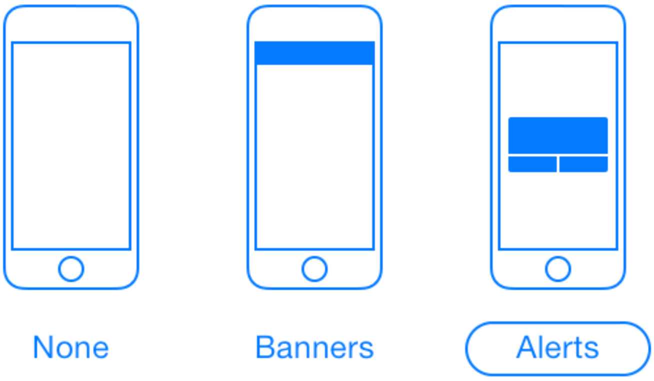


Figure 5: iOS menu with the different alert styles.

Figure 5 shows the different visual presentations of push notifications in iOS 7 and later. The user can chose if and how he is going to be notified by each application. While the *Banners* notification style is only shown for a few seconds and then disappears, the *Alerts* notification type offers asks the user to make a choice - usually between opening the application and dismissing the notification. To open a *Banner*, the user has to clock the notification before it disappears.

2.4.2 Push Notification Strategies

Research on push notifications is mostly done by private enterprises or blogs. We have not found any research paper or scientific publication on the matter of push notification strategies. But the recommendations found online mostly suggest that the notification strategy, at least for a commercial application, highly depends on the branch. News or financial related applications can notify the user a lot more often than applications from other sectors.

Most articles list four tips for using push notifications:

- Consider the user preferences and allow them to opt-out
- Find the right frequency and time
- Use Geodetection if possible
- Segment the audience

The first recommendation should prevent the user from deleting the application from his phone in order to get rid of the notifications. For our application we decided to not include a opt-out button. We feared that too much participants would use this option and lead to a much smaller dataset when the application is only used for two days. The second tip advises to find out, not only what a suitable amount of notifications is, but also when to send them. For some applications it is more suitable to send notification on a Monday morning instead of a Sunday afternoon[51].

Other source suggest to notify the user depending on its location. Thanks to modern indoor-location technologies it is possible to determine the user's position even inside of buildings. The last suggestion implies to use background information. This could mean to use data from a CRM or to detect and store how a single user reacts on notifications and use this information to create a custom strategy for each user.

3 Concept Overview

3.1 Rationale

The previous chapters have shown the importance of human comfort and what kind of sensing techniques are nowadays used to capture comfort related information. The measurement of human comfort is mainly based on two pillars. On the one hand, there is the gathering of objective data on environmental factors like air temperature, relative humidity, illuminance and noise. On the other hand, there are personal factors like light sensitivity, metabolic rate and clothing level but also body shape and personal preferences. While environmental factors always should be measured by sensors or technical devices, the personal factors are almost impossible to measure without human interaction.

Purely sensor based data can be used to compute predictive models on human comfort like the PMV or the PPD models by Fanger[19]. But it is important to realize that even though these models allow to predict a comfort zone for an average person, the predicted comfort zone can vastly differ from the comfort that a person experiences.

While research has investigated the single effect of thermal, visual and acoustic factors for the last decades, studies that tried to measure the combined effects haven't been made until recently. Studies that combine these factors have shown that they seem to influence each other[59].

Large projects like the European HOPE project, which was conducted between 2010 and 2013, have investigated the effects of various factors on perceived human comfort[12]. This project, which collected questionnaires from 5732 participants in 59 office buildings all over Europe, showed that perceived comfort is more than the addition of single comfort factors and deserves more and more selective research[12]. It is also suggested to add physiological, psychological and social aspects to the information collection process on human comfort[52].

3.2 Proof of Concept

The aim of this project evaluates if it is possible to collect human comfort related subjective and objective data with a standard smartphone. The project should serve as a proof of concept on how to interact with the user. This includes the design of a suitable interface as well as the right amount of interaction with the possible users.

To capture environmental data, the application should be able to collect reliable and comparable data from the outside. Therefore, mobile sensing techniques come to play. As shown in subsection 2.3, modern smartphones are capable of sensing various variables from the outside world. Sensors like an accelerometer, multiple microphones and a camera are standard equipment in phones of almost every manufacturer.

Besides the mobile sensing approach, the application should cover aspects of mobile Experience Sampling and collect subjective data at the same time. The user should be able to communicate personal impressions on certain variable important to human comfort.

The utilization of smartphones comes in handy as they are widely spread among most of the developed countries and allow to collect data with almost no restriction to physical space.

The project should also give insights on how to interact with the user. Different notification strategies will be tested on a set of users to determine the best frequency. Furthermore it will be analyzed how exact the data of the sensors represent the users impressions.

3.3 Parameters to Capture

The parameters we try to capture are shown in table 5. We focused on the variables with the biggest influence on human comfort. Since not all of these factors can be sensed with the built-in sensors, we could only gather objective data on the ones that have a corresponding sensor.

Objective Data (Mobile Sensing)	**Subjective Data** (Mobile Experience Sampling)
Sound volume	Acoustic comfort
Illuminance	Visual comfort
	Thermal comfort
	Current activity

Table 5: Parameters captured by the application.

This parameters are chosen because of their importance for human comfort together with the technical limitations of the smartphone as a sensor. Visual and acoustic signals can be collected directly with the built in sensors of a smartphone and can then be compared to the subjective impression of the user. Besides these variables, thermal comfort and the current metabolic rate are two of the most important factors that determine human comfort[20]. But depending on the research background, other variables can be tracked.

The parameters captured can be easily extended with the utilization of external sensors as shown in section 4.1. Without external sensors the smartphones capabilities are limited. To collect objective information it needs to be exposed to the same conditions that the user experiences. When placed inside the pocket or covered by something it is not possible to make assumptions on the surroundings.

This is especially problematic for longitudinal studies since accessing the phone's sensors over a longer period can quickly drain it's battery and make in therefore useless.

4 Project Overview

This chapter serves as an overview of the whole application and the corresponding back-end. The first section shows the architecture of the application and back-end and explains the notification sending routine. It shows the process from sending out a notification until the data is saved into the corresponding database and how it can be interpreted.

The second sections shows what design choices were made in case of the application itself. It also focuses on the collection of the data on the device and the user interaction that follows after the push notification is received.

4.1 System Architecture

Figure 6 shows the architecture of the project from triggering a notification until the storing of the received data into the database. To enhance the comprehensibility, some of the processes like the registration for the APNs are simplified.

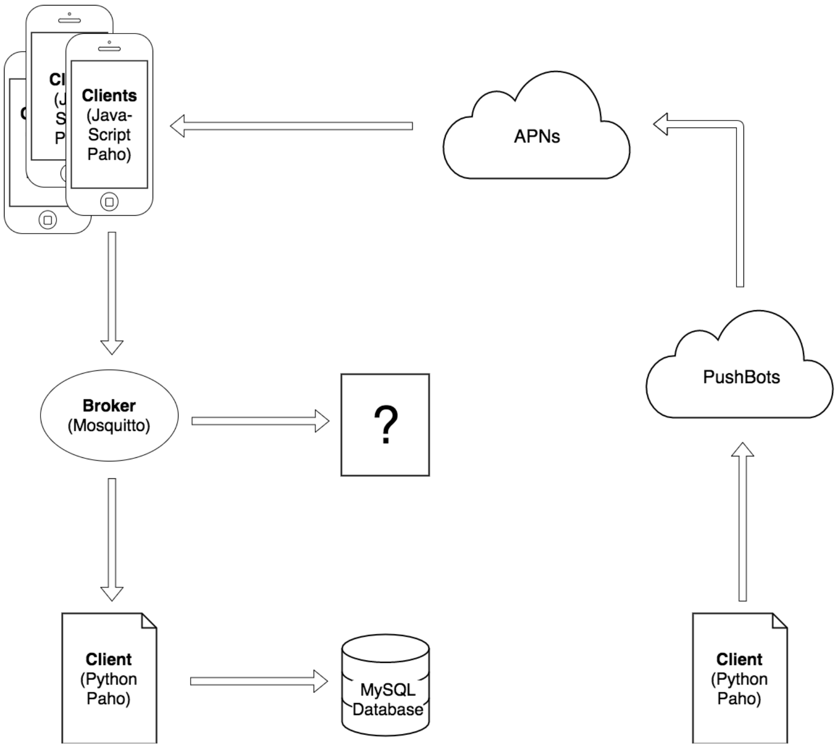


Figure 6: The overall system architecture

In a first step, the server automatically executes a script at a given time. The scheduling is done by a CRON job to allow the exact timing of the notification for every single device. The script then sends a message to the PushBots API. PushBots is a service that offers access to the Apple Push Notification Service directly from a webbrowser or via an API.

Once the message reaches the PushBots servers, it gets transmitted to APNs. The process is shown in more detail in chapter 5. APNs then sends a notification to the selected devices, where the user gets notified and asked to fill out a form.

The form and the data collected by the sensors then gets transmitted to a server via the MQTT protocol. The server is a SWITCH Machine running Ubuntu Linux and a Mosquitto[44] message broker. Mosquitto is built upon the publish/subscribe model which is further discussed in section

6.2 and listens to the messages sent by each phone.

In a fifth step, another script listens to the Mosquitto message broker and subscribes to the messages published by the smartphones. The messages then get saved into a Mysql database.

The system is highly extensible. Other services can easily subscribe to the message broker and collect the data collected by the phones. Or the service presented in this project could gather additional data through external sensors.

4.2 Design Choices

The design process of the application and the corresponding back-end can be divided into three parts. The first one covers the sending and receiving of a signal and the selection of a specific user. The second one treats the collection of data from the user and his sensors. And finally the storage of the data and how it can be connected to other data sources and consumers. The whole process is shown in the diagram in figure 7.

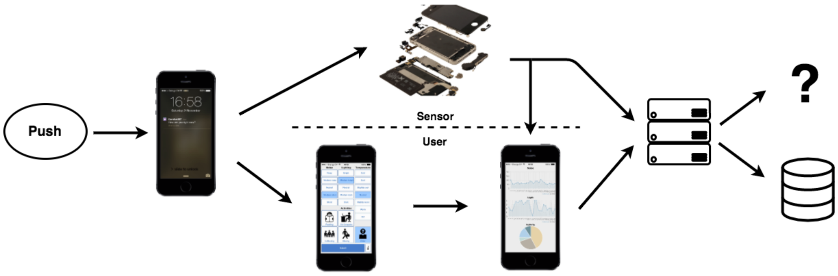


Figure 7: Process overview

The most important goal of the design process was to build an application that has a high usability, is easy to understand and should cause the least possible disturbance of the user. Besides that, the researcher should have a high flexibility on selecting the user and timing the experience sampling process.

4.2.1 Interrupting the User

The user should be interrupted certain times a day and asked to fill out the survey. This part resembles the *signal* in the traditional experience sampling process shown in figure 3.

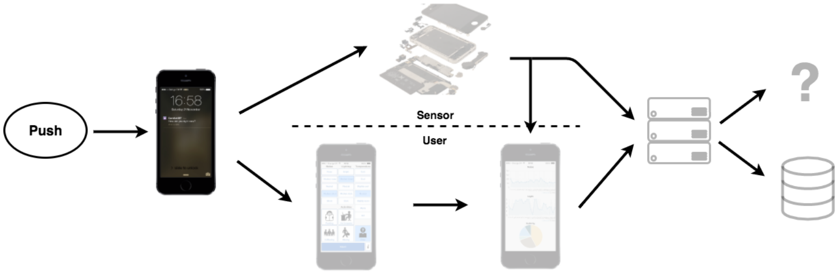


Figure 8: Design of the push mechanism

Since the user already carries a smartphone and takes the survey on the smartphone, it is obvious to use the smartphone as a signal device. When an application on a modern smartphone is asking for attention, it is usual to send a push notification to the user as discussed in section

6.4.1. This process is shown in figure 8.

When a notification arrives at the user and the phone is locked, the notification stays on the lock-screen until the user unlocks the device. He can directly access the device by selecting the notification or dismiss it by unlocking the device directly. If multiple notifications have arrived while the phone is locked, the user just has to answer once.

To make sure that every user answers the same way, the app has a built-in instructions page as seen in section 7.2.

4.2.2 Data Collection

Once the user got a notification and opens the application, a single form is shown. The user can select from multiple buttons what his or her perception of comfort is. The forms are assembled on a single page of the application to let the user select his current state in the fastest possible way. The interface is shown in section 4.3.

As soon as the user made a selection, the submit button collects the information of the form and saves it on the device. During the selection of the states, the application has access to the built-in sensors. This is shown in figure 9.

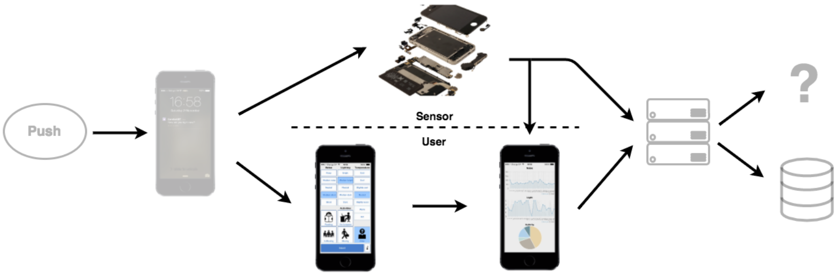


Figure 9: Collection of user and sensor data in the application

After selecting the states, a second screen is shown. This screen shows a graphical representation with the last recordings of the user and sensor data. This screen has two purposes - first of all it is impossible to close an application from within an application in iOS. And since it is impossible, the visualization of his data can serve as an incentive for the user to continue filling out the questionnaire and submitting data to the researcher.

The data that is used for the visualization is saved locally in a SQL database on the phone. This should minimize the amount of mobile data which the transmission of information from the app to the server needs.

4.2.3 Data Storage and Utilization

Once the user has submitted the form and the sensors have collected the data on the environment, the information has to be sent to a central server.

Since the data collection takes place on a mobile device, there is a good chance that the device is connected to the internet via a cellular network. So, to provide a good experience and minimize the costs, it is necessary to keep the amount of data sent and received as small as possible.

Besides the low data traffic, the system has to be seamlessly extensible. This should provide the whole system to access to external sensors without major changes in the systems architecture. The application, together with the data entered by the users, should also be able to serve as a sensor itself in the context of a bigger system which gathers information on comfort.

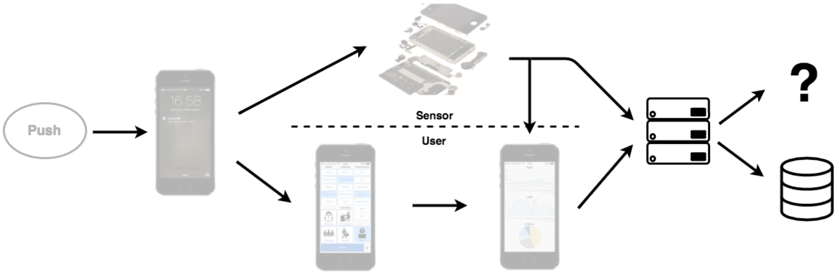
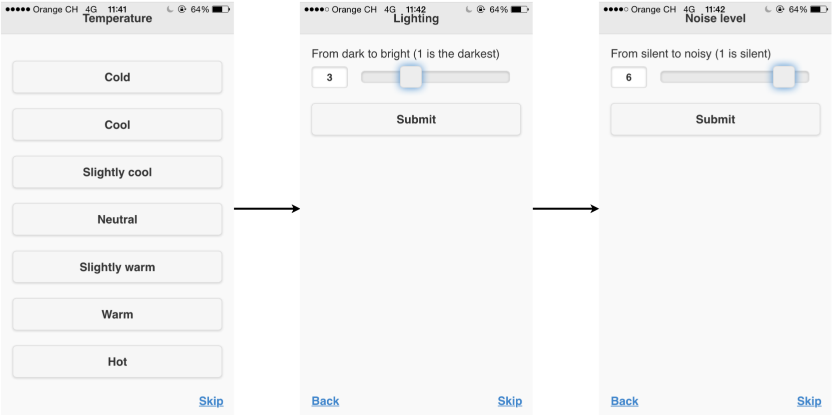


Figure 10: Transmission of the data

To satisfy this needs, we chose to build the data transmission upon the publish-subscribe pattern. This patterns is widely used for network communication in the context of the so called internet of things[28].

We chose the Message Queue Telemetry Transport (MQTT) protocol to fulfill our needs. The protocol is further explained in section 6.2. When implementing the transmission of the data to the server, we tried to adapt the message oriented design pattern of modern machine to machine communication. This means that a client publishes a single value on a certain topic.

Even though this is a great way of publishing information for the internet of things, it makes the analysis of the data unnecessary complicated. The data produced by the application gets saved into a database in a format shown in figure 22. When we tried to analyze this data later and compare the values entered by the user with the corresponding sensor values, we quickly realized that it is hard to combine the seven database entries into a single one. As you can see in figure 22, submitting the form leads to a unique combination of appID and time-stamp. But since the transmission of the data takes time, it is possible that a single submission has two time-stamps or that they get mixed up when two users submit the form at the same time. We also encountered that a single submission is saved multiple times into the database.


●●●●● Orange CH 4G 11:41 64%
Temperature

Cold

Cool

Slightly cool

Neutral

Slightly warm

Warm

Hot

Skip

●●●●○ Orange CH 4G 11:42 64%
Lighting

From dark to bright (1 is the darkest)

3

Submit

Back Skip

●●●●○ Orange CH 4G 11:42 64%
Noise level

From silent to noisy (1 is silent)

6

Submit

Back Skip

Figure 11: Interface of the prototype

This fact makes the analysis of the data time-consuming. To prevent this, we propose to adapt the messages on the purpose of the experiment. If it is only used for data analysis, a single entry with multiple values would be easier to parse. This approach could also be used in combination with the message oriented approach if needed. Further information on this matter can be found in section 6.2.

4.3 Evolution of the Interface

The interface is designed to allow the user the fastest possible entry of his data. We have built two fully functional prototypes and compared the ease of use of both. The interface of the first prototype is seen in figure 11. It consists of three different views, each collects the users perception of either temperature, lighting or noise. The views vary between forms with multiple buttons and ones with a slider plus text input field.

We found out that this approach has a major disadvantage. The user is asked various times during a day on his or her current perception of comfort. But even though it changes sometimes, chances are high that during a normal workday only some of the variables change while others stay the same. So it is useful to only alter the values that have changed. This is not handy with a multiple-view-form. Furthermore, we wanted to reduce the cognitive load that is needed to fill out the form. Both of this lead to the single-view interface seen in figure 12.

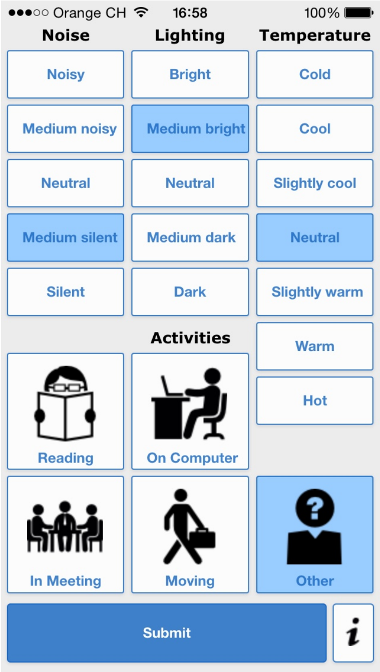


Figure 12: Input form on a single page

The form has four dimensions. Noise, Lighting, Temperature and Activities. The temperature dimension is based on the ASHRAE seven-point thermal sensation scale. The scale can be found in table 1. Since there does not exist a standard scale for measuring light and noise, we chose a five point liker-like scale to get information on noise and lighting. The activities scale is built upon usual office activities. There exist scales for different activity levels like the one proposed by Fanger[19], which is used to estimate different metabolic rates. The activity scale by Fanger can be seen in table 2. But since the Fanger scale doesn't distinguish between different kinds of indoor activities well enough, we decided to implement a customized activity scale with five different values to choose from. We chose values that would mean something to the user and cover the most important activities during a workday.

By pressing the "i" button on the lower right side of the interface, the user can get general information about the application and on how to use it. This should ensure that every participant has the same information on how to conduct the survey and that we can hereby reduce a possible bias of information.

5 Implementation: Mobile Device

5.1 Technologies

The decision to build a smartphone application usually directly leads to the first question - which mobile operating system/s (mobile OS) should be targeted. Android, iOS and Windows Phone are only a few of the candidates a developer has to choose from. Each of these operating systems is based on another programming language or at least offers different API's what leads to a large amount of additional work to make an application capable of running on multiple mobile OS.

Some developers try to avoid this by building their applications directly on the web. But this webapplications have a number of disadvantages. Even though HTML 5 together with local storage and browser application caching made it possible for webapplications to run offline, there are still a number of downsides on this approach. Purely web-based applications don't have merely the access to underlying APIs that native code offers[64]. Other downsides on this approach are the lack of native feeling when interacting with application and the absence of push notifications[5].

But in recent years projects like Apache Cordova[1] made it possible to build a single application that is capable of running on different devices. This so-called hybrid applications bridge the gap between native and web-applications. Built of web-technologies like HTML, CSS and JavaScript, these applications run native on the mobile device and allow access to features that until then have been exclusive for native applications.

Since one of the core features of our application is the access on the smartphone's sensors, which requires access to native APIs, only native and hybrid technologies could serve our purposes. This section will therefore highlight the differences between native and hybrid applications and illustrate the decision we made. It will further give insights on how hybrid applications work in detail and what is important to remember in their development process.

5.1.1 Native

Native applications have several advantages over hybrid- and webapplications. The most important one is the better performance. Native applications are compiled directly into machine code and are thereby faster than hybrid applications which run in a webview. Especially for graphic-intense applications, native code is surely the better choice.

Pros	**Cons**
Performance	Cost/Time
Platform specific design	Maintainability
Native feeling	Developer experience
App Store Regulations	Code reusability
No limitations	
Documentation	
Debugging	

Table 6: Pros and Cons of native applications.

A listing of the advantages and disadvantages can be found in table 6. The main arguments for native applications besides the better performance are the native feeling and the platform specific design. The design of menus can mostly be adopted directly from the operation system.

Besides that, native developed applications are more certain to be allowed in the corresponding application stores. Apple for example states that applications that lack the iOS specific look and

[5] Even though Google offers solutions to use push notifications in Chrome, there is to this day no solution for iOS.

feel can be rejected from the App Store[4] [60].

Another benefit is the better documentation, at least for the more popular operating systems like iOS, Android and Windows Mobile. Documentation and support is really advanced compared to hybrid application frameworks like Cordova and others.

Debugging is also a big advantage over hybrid apps. Xcode (Apple's IDE for developing iOS applications) for example has already quite advanced debuggers built in, which help the developer to track down bugs. Debugging hybrid applications on the other hand has limitations and requires additional development tools.

The most important factor to choose hybrid over native applications is the cost/time factor. To reach a majority of users, multiple versions in different programming languages have to be built. This is time and money consuming and leads to large code maintaining efforts. Developers need to be experienced in multiple programming languages and have knowledge about different tools and APIs.

5.1.2 Hybrid

The term *hybrid application* describes an approach to build "[...]cross-platform mobile applications and [...] implement it as a combination of native and web application technologies." [64]. With the development of a growing number of different mobile operating systems and their different programming languages, came the wish for an approach to develop software for multiple of those OS with a single framework. Hybrid applications serve as a possible solution to this problem.

Hybrid application frameworks can be further divided into ones that translate their code into the target's native language or those who run them within a native application shell. An example for a translating framework is Xamarin[6], that translates source code written in C# into native code for iOS, Android and Windows Mobile or Titanium[7].

Examples for frameworks that run in a native application shell are Adobe PhoneGap[8]/Apache Cordova[9], Sencha Touch[10] and Ionic[11] [12].

For this project we chose to use Apache Cordova. The main reasons were the comparatively good documentation and the existing experiences with web technologies of the author. Other important factors are the fact that the framework is open-source and its maturity(Version 5.4.0 to this day).

[6] `https://xamarin.com/`

[7] `http://www.appcelerator.com/`

[8] `http://phonegap.com/`

[9] `https://cordova.apache.org/`

[10] `https://www.sencha.com/`

[11] `http://ionicframework.com/`

[12] Ionic started as a front-end framework but nowadays offers additional functionality.

5.2 Apache Cordova

The development of Apache Cordova started in August 2008 under the name PhoneGap. It has been developed by the company Nitobi which has been purchased by Adobe in 2011[45]. At the same time, Nitobi announced that PhoneGap applied to become an open-source Apache project[13] and would be called Apache Cordova. The version distributed by Adobe is called Adobe PhoneGap or just PhoneGap and is based on Apache Cordova but adds additional functionality like the distribution platform PhoneGap Build[14].

Cordova/PhoneGap uses an approach of running in a native application shell. As seen in figure 13. When a Cordova application is invoked, it opens a native web view with a specific starting page, written with the earlier mentioned web technologies (HTML, CSS & JavaScript)[64].

With the web view in control, the user can interact with the underlying HTML and JavaScript to navigate to other pages or load additional packages. A native web view takes care of the rendering of web content - usually websites written in HTML[64]. Which web view is used depends on the operating system. For Android it is *android.webkit.WebView*[15], for iOS it is *UIWebView*[16] (using System/Library/Frameworks/UIKit.framework)[64].

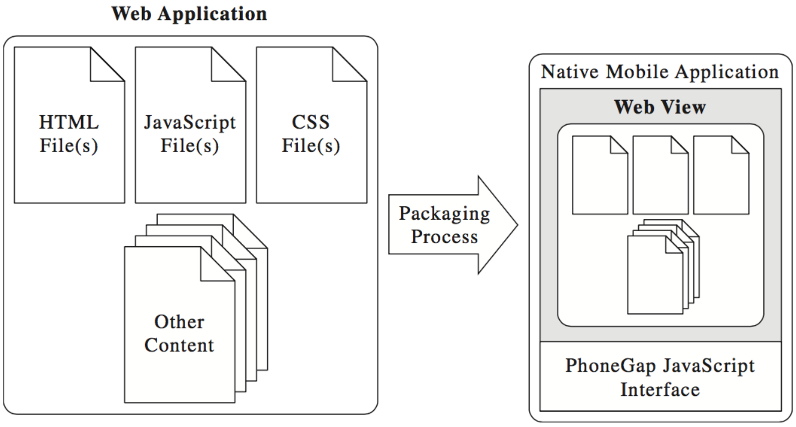


Figure 13: Apache Cordova Application Packaging Process. Source:[63]

At this point the application wouldn't offer any additional functionality compared to a normal web-application. What makes Cordova so powerful is its ability to connect to the native APIs of the different mobile operating systems. A single JavaScript interface comes along with native code libraries for a number of supported platforms[50]. Figure 14 gives an illustration of this process.

To this day, the Apache Foundation lists 796 Cordova specific plugins[17]. These plugins allow the application to access the camera and microphone or more advanced features like barcode-scanners and tracking of device motion. It is also possible to build and publish custom plugins as seen in section 5.2.1. In fact, most of the plugins for Cordova are open-source and therefore accessible by anybody.

[13] http://apache.org/

[14] https://build.phonegap.com/

[15] http://developer.android.com/reference/android/webkit/WebView.html

[16] https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIWebView_Class/

[17] http://cordova.apache.org/plugins/

But is has to be said that during our project we found out that not all of this plugins that well-maintained and do not function the way it is expected. Some of the code repositories on Github haven't been updated for months and are not compatible to current mobile OS or Apache Cordova versions. A negative example is the push-plugin with 74 open issues on Github[18].

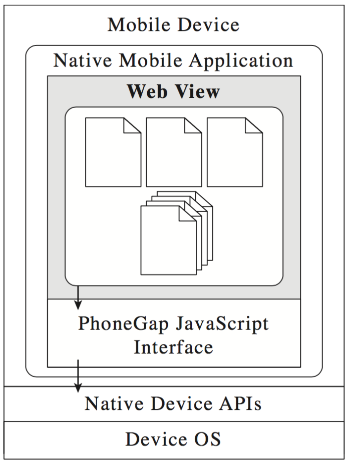


Figure 14: The Cordova JavaScript interface. Source:[63]

5.2.1 Custom plugins

As seen in the previous chapters, a Cordova plugin adds additional functionality to an application through a bridge between the applications JavaScript and native platform specific code. These plugins have to be built in a certain way and fulfill specifications given by the framework in order to work well. A typical structure for a plugin with support for iOS and Android looks like this:

- plugin.xml
- README.md
- src/
    - iOS/
        - <Objective-C/Swift source code>
    - Android/
        - <Java source code>
- www/
    - <JavaScript interface>

The plugin.xml contains information about the application itself like its name and description, under which license it has been published and information on supported platforms and configuration files.

The JavaScript part of the plugin serves as the front-facing interface[50]. This means that a single call can be translated into various native methods depending on the underlying operating system. Its syntax is shown in Listing 1. The first two parameters deal as success and error callback functions. Service and action define the native class and their according method. The array at the end of the function takes optional arguments.

[18] `https://github.com/phonegap/phonegap-plugin-push/issues`

Listing 1: The front-facing JavaScript interface

```
cordova.exec(function(winParam) {},
             function(error) {},
             "service",
             "action",
             ["firstArgument", "secondArgument", 42, false]);
```

The native counterpart of the JavaScript interface is shown in listing 2. Once the JavaScript fires off the plugin request on the native side, the appropriate method defined in the JavaScript interface is fired. The native code gets executed and saves the resulting object into an object of the type *CDVPluginResult* as seen in listing 2.

A result of *CDVCommandStatus_OK* will cause the JavaScripts success callback function to be executed or the error callback in case of an *CDVCommandStatus_ERROR* object.

Listing 2: Native connection to the JavaScript interface

```
1
2       - (void)myMethod:(CDVInvokedUrlCommand*)command
3       {
4           CDVPluginResult* pluginResult = nil;
5           NSString* myarg = [command.arguments objectAtIndex:0];
6
7           if (myarg != nil) {
8               pluginResult = [CDVPluginResult resultWithStatus:CDVCommandStatus_
                    OK];
9           } else {
10              pluginResult = [CDVPluginResult resultWithStatus:CDVCommandStatus_
                    ERROR messageAsString:@"Arg was null"];
11          }
12          [self.commandDelegate sendPluginResult:pluginResult callbackId:command.
                callbackId];
13      }
```

The native code in listing 2 for example saves a string with the message ”Arg was null” as seen on line ten into the CDVPluginResult object in case the variable ”pluginResult” on line four is ”nil”. Otherwise an object with status okay is sent.

5.3 Cordova Application

Besides the reasons to build the application with Apach Cordova, mentioned in section 5.1.2, the extensibility was an important factor. Applications built with Cordova can be easily extended in their functionality trough additional plugins. To do this, the code base for interface and visualization do not have to be changed. That approach allows a comparatively fast adaption to extended project objectives.

Another reason to build the application with Cordova was that it should be compatible with multiple smartphone operating systems. This goal had to be changed during the project due to the lack of time to build a native implementation for Android and Windows Mobile. During the process, we decided to concentrate on iOS. We ruled against Android because iOS is more popular among the members of the Human-IST who served as beta-testers of our application and more popular in Switzerland in general[9].

The following chapter gives insights on how the applications works. In a first section the JavaScript implementation of the app, the part that also contains the logic, is explained in detail. Section 5.4 then gives insights on the native plugin which was built to access the native sensors of the phone. The last sections deal with the data visualization inside the application and how the data is transmitted to the servers. The last chapter explains distinctive features of the debugging process for hybrid applications.

5.3.1 JavaScript Implementation

The whole applications consists of a configuration file in XML, HTML, CSS and JavaScript files and the according Objective-C classes in which the plugin is written. These files are then compiled to a native iPhone application which happens through services like PhoneGap Build or through the Command Line Interface(CLI) of Apach Cordova. The source three of our application is shown in figure 15.

```
- config.xml
- www/
    - index.html
    - visualization.html
    - css/
        - index.css
    - js/
        - index.js
        - dataviz.js
        - jquery.js
        - chart.js
        - [...]
- Plugins/
    - plugin.h
    - plugin.m
```

Figure 15: Structure of the Cordova application.

The user accesses the application through a HTML form with the name index.html. The navigation inside Cordova is similar to the navigation on a Website. So if the view redirects the user, a different .html page is rendered.

The backbone of every Cordova application is its main JavaScript file. This files gets loaded once the native code-base of Cordova is operational. It needs therefore to be packed inside a *deviceready eventlistener* that waits until the application is fully loaded. Besides the main index.js JavaScript file, the application consists of additional libraries like jQuery and Chart.js.

Figure 16 shows the flowchart that represents the main form of the application and the corresponding background processes.

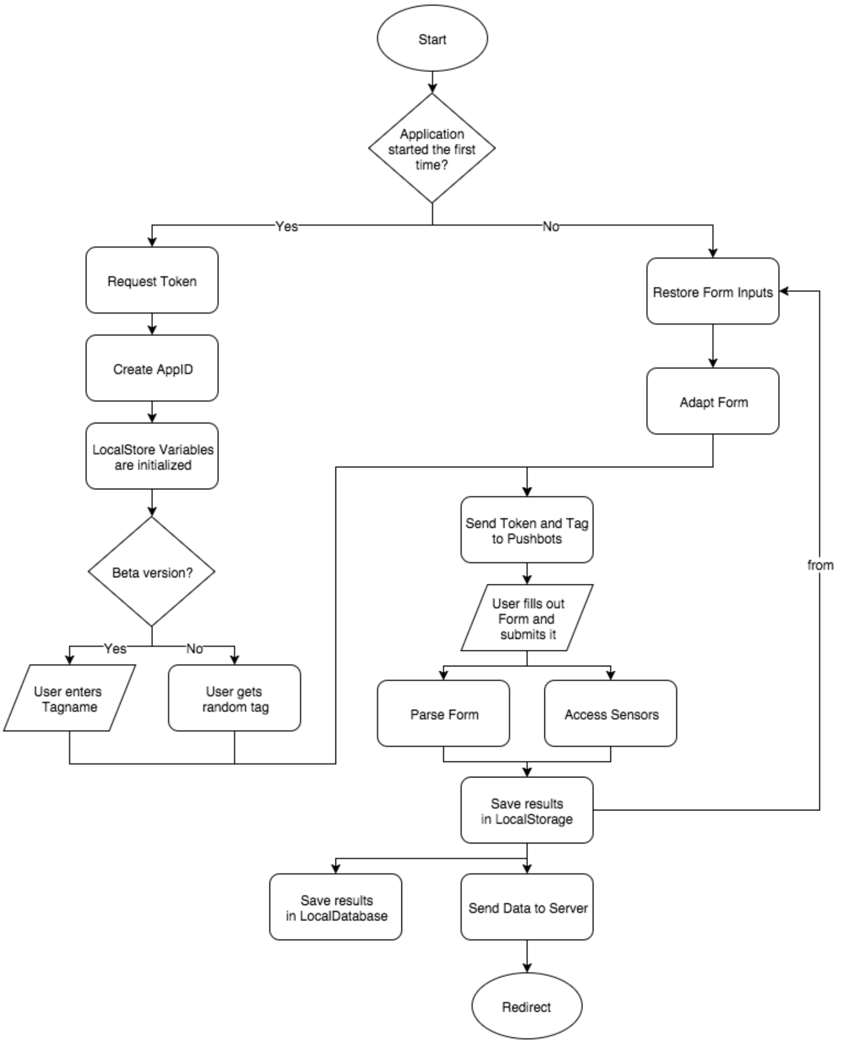

Start
Application started the first time?
Yes
No
Request Token
Create AppID
LocalStore Variables are initialized
Beta version?
Yes
No
User enters Tagname
User gets random tag
Restore Form Inputs
Adapt Form
Send Token and Tag to Pushbots
User fills out Form and submits it
Parse Form
Access Sensors
Save results in LocalStorage
from
Save results in LocalDatabase
Send Data to Server
Redirect

Figure 16: Flowchart of the first page of the application.

If the application starts for the first time, a few additional processes are necessary to provide the application with an unique Application Identifier(AppID) and a Tagname. The AppID is used to distinguish the data that the different participants send to the server and gets generated by a random ID generator.

This approach was chosen to make sure that even with a large number of participants, we could still distinguish the data received by each of them. The generator is shown in appendix A. The numbers and letters used in the IDlist together with an IDlength of eight characters lead to $(27+27+10)^8 = 281474976710656$ different possible IDs. Even though the Math.random methods' don't generate perfectly random numbers, it should make sure that with a sufficient probability, no ID is allocated twice[19].

The approach with the AppID generator is designed to distinguish different users without their intervention. For the beta version, we chose to let each participant enter their Name through a Popup at the first application launch. This allowed us to send push notifications to each participant independently. This approach was necessary since most of the participants that took part in the first experiment were located in the same office. Notifying half of the them at once would lead to a huge distraction. More information to this matter can be found in section 7.

In an experiment with more participants, instead of using a Tag entered by the user itself, it would be possible to use a random generator once again. For an experiment with two different groups, it therefore would be possible to tag them randomly with ones and zeros and then send the them different push notifications.

After this step, the process continues as if the application had been started before. The Token and the Tagname of the device are sent to the PushBots server. This happens even if the application had been launched before due to the PushBots plugin architecture. It further allows the future integration of a do-not-disturb button or similar.

Once the interface is fully loaded, the user can fill out the form shown in figure 12. We implemented an event listener in JavaScript, that added the class *active* as soon as a button is clicked. This is shown for the button with the *Value="Noisy"* in listing 3.

As soon as the user presses the submit button, two things happen at once. The JavaScript part of the applications parses the form and save the value of the field *val* into a variable. Since we have four different values in one form, the JavaScript also stores the *id* of according parent class which in this case is the *id="noise"*. Simultaneously the JavaScript interface makes a call to the native plugin shown in section 5.4.

The results are then saved into localStorage variables. This is necessary to make them available the next time the application is launched. Since the application lives inside a Web View, all variables get reset after the application is closed. The localStorage variables are then used to be sent to the server and to restore the form inputs as soon as the application gets opened up again.

The information captured is also saved into a local SQL database. This is necessary since the visualization on the second page of the application needs to get information from more than one point in time. The data-visualization on the smartphone is explained in more detail in section 5.3.2.

Listing 3: Extract of index.html

```
<div id="page1">
   <div id="formScales">
       <div class="formContent" id="noise">
          <p>Noise</p>
          <input type="button" id="nbtn1" value="Noisy" val="4" class="active"/
              >
          <input type="button" id="nbtn2" value="Medium noisy" val="3"/>
       [...]
   </div>
</div>
```

The *val* values of each input button get readout by a JavaScript function and saved into an array as soon as the user clicks the Submit button on the application.

[19] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/random

5.3.2 Data Visualization

The visualization directly on the phone is an important part of the application. It should give the user an incentive to answer to the push notifications and let him see his own results in an understandable form.

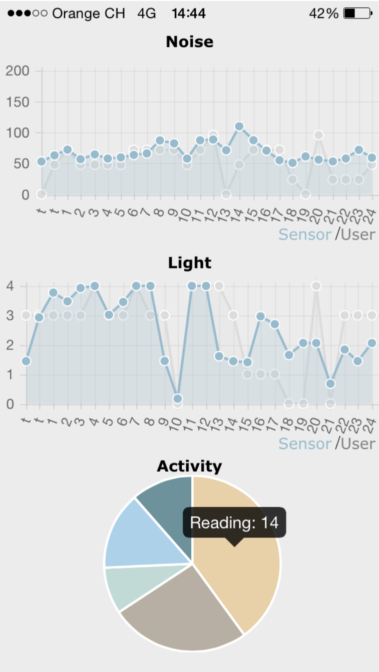

●●●○○ Orange CH 4G 14:44 42%
Noise
200
150
100
50
0
t t 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
Sensor /User
Light
4
3
2
1
0
t t 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
Sensor /User
Activity
Reading: 14

Figure 17: Data-visualization on the iPhone.

The visualization of the data of one specific user is shown in Figure 17. The view consists of three different diagrams. We decided to display the data of both sensors and the corresponding data entered by the user through the form. The temperature values are not shown. We decided not to present them to the user at the current state, since there is no corresponding sensor to which the data entered by the user could be compared to. The pie-chart on the bottom of the view represents the activities.

The representation of the acoustic and visual information is done by curve charts. The numbers on the x-axis each represent a single form submission with the corresponding value on the y-axis. We chose to show 26 datapoints, of which the first two are marked with a *t*, illustrating that they are test values, used as an example when explaining the application to the test users of our experiment. The number of 24 points is composed out of two sets of 12, which is the number of push notifications we sent per day during our two-day experiment. So a participant can see his progress.

Each chart shows a combination of user and sensor data. If the user clicks on one of the datapoints, a container with the exact data pops up as seen for the pie-chart in figure 17.

Storing information for a longer period on an application that is based on web-technologies is not that simple. For the visualization of our data, we need to store a growing number of different keys and values over multiple days. The normal approach would be the storage in an object of the type localStorage. But localStorage only supports strings, which means that the array would have to be converted into a string to be saved and later parsed to be used again. This approach is computationally intensive, especially with a growing amount of data.

To enhance the performance of the application, we implemented a Web SQL database inside the application. This database can be accessed in the regular SQL syntax as seen in listing 4 and provides the perfect solution to our problem.

Listing 4: Creating the Web SQL database

```
var db = window.openDatabase("localDB", "1.0", "Local DB", 10000000);
db.transaction(runTransaction, errorDB, successDB);
[...]
function runTransaction(t){
    t.executeSql('CREATE TABLE IF NOT EXISTS comfort (id unique, noiseS, noiseU
        , lightS, lightU, date)');
    t.executeSql("INSERT INTO comfort (noiseU, lightU, noiseS, lightS) VALUES (
        "+noiseUdb+", "+lightUdb+", "+noiseSdb+", "+lightSdb+")");
}
function errorDB(err){
    console.log('Error creating tables: '+err);
}
function successDB(){
    console.log('Successfully created tables');
    window.location.href="question.html";
}
```

The visualization itself was done with the JavaScript library chart.js[20]. The charts were rendered directly after the submission of the form.

5.3.3 Data Transmission

The transmission of the data takes place before the visualization of the charts inside the second view. As soon as the data is saved into the Web SQL database, it is sent to the server as seen in chapter 6.

We chose to send the data directly after the submission of the form. To submit the form, it is therefore necessary to have access to the internet. Otherwise, an error message is sent.

The first prototype of the application used AJAX(short for asynchronous JavaScript and XML) to transmit the data to the server. This approach used a POST request over HTTP to transport the data from the device to the server. The reason for changing to the MQTT protocol are listed in section 6.2.

To send MQTT messages from within a browser, it is recommended to use it inside the WebSocket protocol[26]. Websocket is supported by current versions of all major browsers including Apple's Safari, Google Chrome, Mozilla Firefox and Microsoft's Internet Explorer and their corresponding mobile web views.

The advantage of using WebSockets is the following: it has been in used for a longer time by all major web-browser producers and can be seen as a solid technology. But the biggest surplus is, that it is already implemented on all smartphone web browsers and hence Apache Cordova.

Figure 18 shows how the messages are sent and received with MQTT over WebSockets. Each MQTT message is enveloped inside a WebSocket frame. As a client, we implemented the open-source JavaScript client Eclipse Paho[21]. The Mosquitto MQTT Broker as well as the MQTT protocol itself is explained in chapter 6.2.

Our application is designed in a way that each variable is sent as a single message. This allows a bigger flexibility when subscribing to single variables. It could therefore be possible to only listen to data from the noise sensor. Exemplary the composition of the message for the illuminance

[20] http://www.chartjs.org/

[21] https://www.eclipse.org/paho/

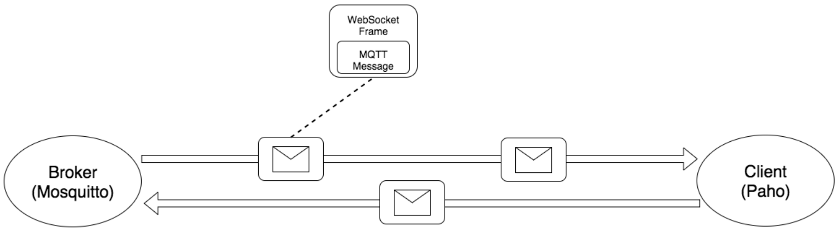


Figure 18: Message transfer for MQTT over WebSockets.

sensor is shown in listing 5.

Listing 5: Creating a MQTT message in Paho

```
1 function sendLightS() {
2   // get value from globalData array
3     lightSValue = globalData.LightS;
4     message = new Paho.MQTT.Message(lightSValue);
5     message.destinationName = "iphone/" + window.localStorage.getItem('appUID')
          + "/lightS";
6     client.send(message);
7 }
```

Every message contains the unique identifier *appUID* and the variable name *lightS* as a message destination name as seen in line 5. The messages each get sent once the user submits the form.

5.4 Native Plugin

Even though the Apache Project lists 796 different plugins for the Cordova framework, there is no working plugin that records values on noise and lighting.

Since the access to the values of the sensors is essential to our project, we decided to start with the development of a plugin at ourselves. The plugin should be able to access the microphone to measure external noise and the luminosity sensor to measure the illuminance.

5.4.1 Noise Metering

Several native decibel metering applications are available in Apple's AppStore. Studies on the quality of those applications made in 2014 showed that out of 130 iOS apps, only ten were classified as useful[34]. This shows that building a trustworthy decibel meter is a complex field. Also to provide valid data, each sensor has to be calibrated. In addition, the hardware used in the different versions of the iPhone differ from each other which makes it hard to develop reliable metering application that is easy to use for the user.

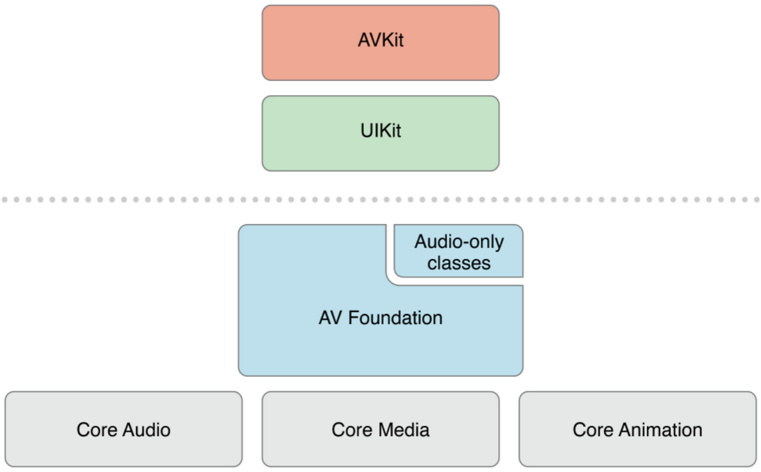


Figure 19: The AVFoundation framework. Source: [2]

The quality of the native plugin will therefore be analyzed in chapter 8 and compared with the inputs given by the users.

Accessing audiovisuel media on Apple's iOS can be done through the AVFoundation framework. This framework provides an Objective-C interface and allows the developer to, among other, capture information on the current audio-level[2].

The framework itself lies on top of the lower-level frameworks *Core Audio*, *Core Media* and *Core Animation* as seen in figure 19. It is itself one of the lower-level frameworks of iOS and provides a lot of functionality. The part of AV Foundation we are going to use is the part of the *Audio-only classes*.

The plugin itself relies on the *AVAudioRecorder* class which is part of the AVFoundation framework. Objects of this class can "[...]obtain input audio-level data that you can use to provide level metering"[6].

Listing 6 shows a small excerpt of the custom plugin. Line ten shows the conversion of the originally negative decibel values into positive ones. The *CDVPluginResult* object in line 13 and 14 provides access to the according variables by the JavaScript interface.

Listing 6: Excerpt of the noise plugin

```
1       recorder = [[AVAudioRecorder alloc] initWithURL:url settings:settings error
            :&error];
2       if (recorder) {
3           [recorder prepareToRecord];
4           recorder.meteringEnabled = YES;
5           [recorder record];
6
7           [recorder updateMeters];
8
9           float averageNS = [recorder averagePowerForChannel:0];
10          float averageNSPos = 120.0f + averageNS;
11          NSString *averageString = [NSString stringWithFormat:@"%f",
                averageNSPos];
12
13          CDVPluginResult* result = [CDVPluginResult resultWithStatus:
                CDVCommandStatus_OK messageAsString: averageString];
```

Apple provides several APIs and libraries to access its audiovisuel sensors. It is possible to directly collect information on the metered sound pressure level through the AV Foundation framework. But to translate this results into the psychoactive loudness of human beings, as seen in chapter 2.1.3, a deeper knowledge of acoustics would be needed.

5.4.2 Illuminace Metering

The best solution to measure illuminace with a smartphone is through its built-in ambient light sensor. But this is not as easy as it seems. To this day, Apple does not provide a public API to access any data from the ambient light sensor. Neither do they publish any documentation on the matter at all. Besides this, Apple states in its App Store Review Guidelines that "Apps that use non-public APIs will be rejected"[4].

So since one of the goals of the project was to build an application that could be published through the App Store, it is not possible to measure the illuminance through the ambient light sensor directly.

To get information on the luminosity anyways, there are mainly two possible approaches. Either accessing the phone's camera and compute the brightness of the picture taken or access an API that uses the ambient light sensor and try to approximate the illuminance from it. Both approaches have different advantages and disadvantages. Table 7 shows them for the camera.

Pros	**Cons**
Highly customizable	Delay until camera is functional
Public API	Imprecise data
Well documented	Energy consumption
	Camera needs to be placed in right angle
	Privacy issues

Table 7: Advantages and disadvantages of the camera as an ambient light sensor

Accessing the phone's camera takes time. Even on new smartphone models, it takes at least two seconds until the camera is fully functional. If the picture is taken, it has to be processed, which can be computationally intensive and therefore drain the battery if done often.

We chose to use the other option. It is possible through public APIs on the iPhone to access the current brightness of the screen. If the Auto-Brightness feature of the phone is enabled, the brightness is controlled directly through the ambient light sensor. Since this sensor is made for exactly this task, it responds fast and gives reliable information about the ambient light. The downsides are that this approach tends to give slightly too high values in average. This is due to the Auto-Brightness feature, which does not dim the background light as fast as it enlightens it. Also this approach can't differ between very bright lighting. This comes due to the fact that upon a certain point, the brightness of the display reaches a maximum. Despite these disadvantages, we thought that this approach fits the need for a fast and least disruptive way to sense the ambient light the best.

6 Implementation: Server-side

6.1 Overview

To store the data sent by applications, we needed a server that is reliable and able to handle multiple connections at once. The SWITCH foundation, which is a partner of the Swiss academic community[58], currently offers the free usage of its SWITCHengines called cloud based virtual machines. Since these machines fit our needs perfectly, we chose to use their services. To get access to this service we had to contact their service team via email and describe our project. Subsequently, we received access to a website where we could manage our virtual machines.

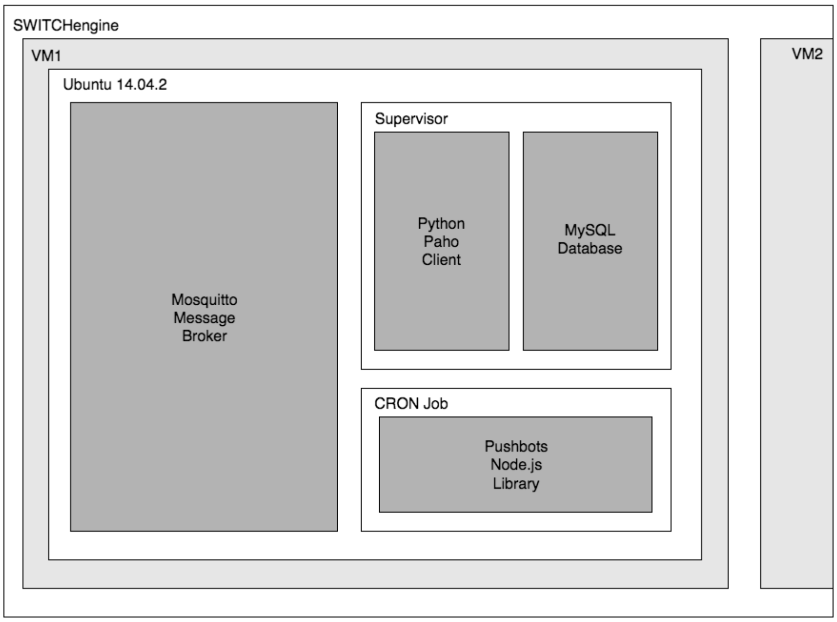


Figure 20: Different layers of Software on the Server.

We chose to install the Linux distribution Ubuntu 14.04.2 and connected to the system over the Secure Shell protocol(SSH) from the Terminal. The main task of the server would be to run a Mosquitto message broker and a MySQL database to store the incoming data. Also it should send out the push notifications to specific devices automatically.

To allow the application to connect to the server, we had to manually manage the firewall settings and open up the according ports. The TCP/IP ports 1883 and 8883 are currently reserved for the usage of MQTT. But since we are using MQTT over WebSockets, it is important to also open port 9001, which is the port associated with WebSockets.

Our machine is accessible under the IP 86.119.31.113 and is equipped with 4 gigabytes of RAM, four virtual CPUs and 20 gigabytes of disk storage. In the three month the server was running, we have not had a single downtime.

An overview of the different software-layers of the server is shown in figure 20. Inside a single SWITCHengine, we are capable of running different virtual machines(VMs). Each VM can be equipped with a custom operating system, which in our case is Ubuntu Linux. To handle the incoming MQTT messages, we installed the open-source MQTT message broker Mosquitto[22]. The task of the MQTT message broker is further explained in section 6.2.

[22] http://mosquitto.org/

The incoming messages have to be stored into a database so that we can analyze them later. To do so, we used the open-source Python MQTT client of the Eclipse Paho project[23]. The Python script is shown in Appendix K.

To make sure that the Paho client and the MySQL database work reliable, we installed the process control system Supervisor[24]. Supervisor makes sure that the Paho MQTT client and the database are restarted if they are shut down or an error occurs. Also it logs every error message into a logfile. The configuration file for the Paho client is shown in listing 7.

Listing 7: Supervisor: Paho script configuration

```
[program:databasePython]
command=python -u dataBaseScript.py
directory=/home/ubuntu
stdout_logfile=/home/ubuntu/databasePython_output.txt
redirect_stderr=true
autorestart=true
```

To send out the push notifications to each user, we made use of the PushBots Node.js library[25]. PushBots is a library that integrates and manages push notifications in Apache Cordova. It helps us to automate the notification sending process to each participant. The PushBots service is described in section 6.4.2.

In the free version of PushBots, scheduling notifications is not possible. So we built our own scheduling service base on the PushBots Node.js library. Messages can be sent through a single JavaScript file. Appendix N shows such a script for a single user with the tag: ”User1”. We created one script for every tag that is used in the survey.

To fire these scripts at a predefined time or interval, Unix systems are shipped with the CRON-daemon. With CRON, multiple scripts can be scheduled to run at a certain date or be repeated in intervals. The so called CRONjobs can be managed trough the CRONtab which is a file that contains all schedules in a certain syntax. Listing 8 shows an exemplary CRONjob for ”User1”.

Listing 8: CRONjobs for User1 on the random strategy

```
#User1 Random
12 8 22 10 * nodejs /home/ubuntu/pushbots/pushbots/user1.js
56 8 22 10 * nodejs /home/ubuntu/pushbots/pushbots/user1.js
13 10 22 10 * nodejs /home/ubuntu/pushbots/pushbots/user1.js
29 11 22 10 * nodejs /home/ubuntu/pushbots/pushbots/user1.js
19 12 22 10 * nodejs /home/ubuntu/pushbots/pushbots/user1.js
26 13 22 10 * nodejs /home/ubuntu/pushbots/pushbots/user1.js
[...]
```

The CRONjob shown in listing 8 is only so long for the random strategy. CRON does an excellent job in scheduling iterative commands. The same CRONjob for User1 with the regular strategy can be seen in listing 9. The syntax for this scheduling is ”Minute Hour Day_of_the_Month Month_of_the_Year Day_of_the_Week”. So a command scheduled for ”5 8-19 21 10 ∗” is executed every fifth minute of the hour on every hour between 8h and 19h on the 21th of the tenth month of the year. This allows to easily schedule even complex intervals.

Listing 9: CRONjobs for User1 on the random strategy

```
#User1 Regular
5 8-19 21 10 * nodejs /home/ubuntu/pushbots/pushbots/user1.js
```

[23] https://www.eclipse.org/paho/

[24] http://supervisord.org/index.html

[25] https://pushbots.com/developer/docs/nodejs

6.2 MQTT-Protocol

The MQTT protocol has been developed by Andy Stanford-Clark and Arlen Nipper in 1999 as an alternative to the existing protocols[33]. The protocol was designed from its beginnings as a lightweight and fast protocol, making it the perfect fit for devices with limited bandwidth and computing power or low energy consumption. It is described by the official MQTT 3.1.1 specifications like this:

> ”MQTT is a Client Server publish/subscribe messaging transport protocol. It is light weight, open, simple, and designed so as to be easy to implement. These characteristics make it ideal for use in many situations, including constrained environments such as for communication in Machine to Machine (M2M) and Internet of Things (IoT) contexts where a small code footprint is required and/or network bandwidth is at a premium.”
> - [46]

MQTT has several differences compared to HTTP. It uses the publish/subscribe pattern instead of request/response and is data-centric in contrast to the document-centric HTTP protocol. This allows MQTT to have really small headers and minimal package sizes. Packages of only two bytes are possible. Furthermore, is it a relatively simple protocol, having only a short list of message types (Connect, Publish, Subscribe, Unsubscribe and Disconnect) compared to the many return codes and messages in HTTP[38].

The properties that make it the perfect fit for Machine to Machine communication and the Internet of Things are the same that make it so advantageous for mobile communication. Facebook announced in 2012 to use MQTT for their mobile messenger because of its faster message delivery, smaller bandwith usage and smaller impact on the phones battery life[33].

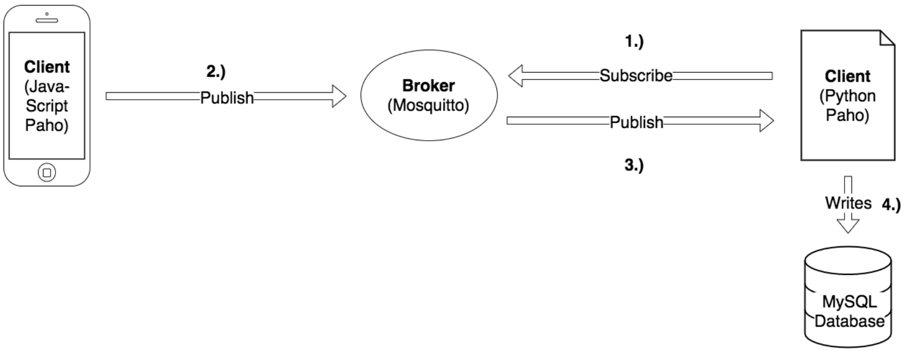


Figure 21: MQTT Publish/Subscribe in our System.

Figure 21 shows the publish/subscribe pattern used in MQTT for our system. A client can be a producer or receiver of information at the same time. To publish a message, the publisher does not have to know the receiver of the message. He just publishes a message with a certain topic to a defined message broker, which in our case is the Mosquitto broker on our virtual machine. Other clients can then subscribe to this topic on the message broker. In our system, this is the Python Script on our virtual machine. As soon as a message arrives at the broker, it gets published to every subscriber to this topic.

Listing 10: Most simple example of publish from the command line.

```
#Client1:
#publish
mosquitto_pub -t 'test/topic' -m 'Hello World!'
```

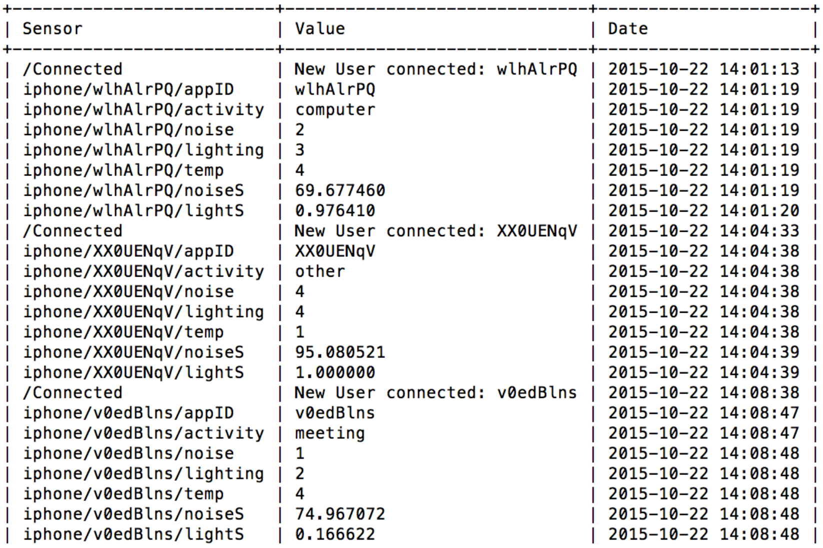
In the most simple case, this could look like the conversation in listings 10 and 11. The second client subscribes to a certain topic as seen in 11 on line three. Then, the first client publishes the message "Hello World!" on the topic "test/topic". The second client, that subscribed to this topic, then receives the message through the broker as seen on line four of listing 11.

Listing 11: Most simple example of a subscription from the command line.

```
1 #Client2:
2 #subscribe
3 mosquitto_sub -v -t 'test/topic'
4 >Hello World!
```

6.3 MySQL Database

Since the system introduced in this project is scaled to process smaller numbers of participants, a standard MySQL database is able to handle the data-storage with ease. Figure 21 shows how the data is written into the database by the Paho Python client shown in Appendix K. Another advantage of using a MySQL database is the ability to export it and use it with external programs for statistical analysis like R[26]. This can be seen in Chapter 8.



```
+--------------------------+------------------------------+---------------------+
| Sensor                   | Value                        | Date                |
+--------------------------+------------------------------+---------------------+
| /Connected               | New User connected: wlhAlrPQ | 2015-10-22 14:01:13 |
| iphone/wlhAlrPQ/appID    | wlhAlrPQ                     | 2015-10-22 14:01:19 |
| iphone/wlhAlrPQ/activity | computer                     | 2015-10-22 14:01:19 |
| iphone/wlhAlrPQ/noise    | 2                            | 2015-10-22 14:01:19 |
| iphone/wlhAlrPQ/lighting | 3                            | 2015-10-22 14:01:19 |
| iphone/wlhAlrPQ/temp     | 4                            | 2015-10-22 14:01:19 |
| iphone/wlhAlrPQ/noiseS   | 69.677460                    | 2015-10-22 14:01:19 |
| iphone/wlhAlrPQ/lightS   | 0.976410                     | 2015-10-22 14:01:20 |
| /Connected               | New User connected: XX0UENqV | 2015-10-22 14:04:33 |
| iphone/XX0UENqV/appID    | XX0UENqV                     | 2015-10-22 14:04:38 |
| iphone/XX0UENqV/activity | other                        | 2015-10-22 14:04:38 |
| iphone/XX0UENqV/noise    | 4                            | 2015-10-22 14:04:38 |
| iphone/XX0UENqV/lighting | 4                            | 2015-10-22 14:04:38 |
| iphone/XX0UENqV/temp     | 1                            | 2015-10-22 14:04:38 |
| iphone/XX0UENqV/noiseS   | 95.080521                    | 2015-10-22 14:04:39 |
| iphone/XX0UENqV/lightS   | 1.000000                     | 2015-10-22 14:04:39 |
| /Connected               | New User connected: v0edBlns | 2015-10-22 14:08:38 |
| iphone/v0edBlns/appID    | v0edBlns                     | 2015-10-22 14:08:47 |
| iphone/v0edBlns/activity | meeting                      | 2015-10-22 14:08:47 |
| iphone/v0edBlns/noise    | 1                            | 2015-10-22 14:08:48 |
| iphone/v0edBlns/lighting | 2                            | 2015-10-22 14:08:48 |
| iphone/v0edBlns/temp     | 4                            | 2015-10-22 14:08:48 |
| iphone/v0edBlns/noiseS   | 74.967072                    | 2015-10-22 14:08:48 |
| iphone/v0edBlns/lightS   | 0.166622                     | 2015-10-22 14:08:48 |
```

Figure 22: Database entries in the MQTT data format.

The structure of the database can be seen in Figure 22. In the first column we can see from which sensor the entry is. The data is structured in topics, which follows the data structure propositions given by MQTT. The Sensor column is therefore composed through entries like "Phone_Model/Application_ID/Sensor".

To select entries, we use SQL string queries as shown in listing 12. This approach is very similar to the one proposed by the MQTT publish/subscribe pattern which is shown in listing 10.

Listing 12: Select entries with certain appID or Date.

```
#Select all entries of an iPhone with an appID of "jbYITDpF":
select * from Comfort where Sensor like "iphone/jbYITDpF/%";

#Select all entries from the 22th of October 2015:
select * from Comfort where Date like '2015-10-22%';
```

[26] https://www.r-project.org/

6.4 Push Notifications

6.4.1 Apple Push Notification Service

The Apple Push Notification Service is responsible for transporting and routing a message from a given provider to a given device[5]. A message always consists of a payload, which is usually the message itself, and a device token, that specifies a particular device[5].

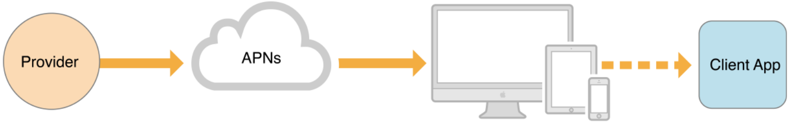


Figure 23: Pushing a remote notification from a provider to a client app. Source: [5]

The path of a notification from the provider to the client is shown in figure 23. It shows that it is not possible to contact a phone or an application running on a phone directly. A provider can only contact APNs, which then directs the message to an application with a certain device token.

Before an application can receive push notifications, it has to request the device token from APNs. This process is shown in figure 24.

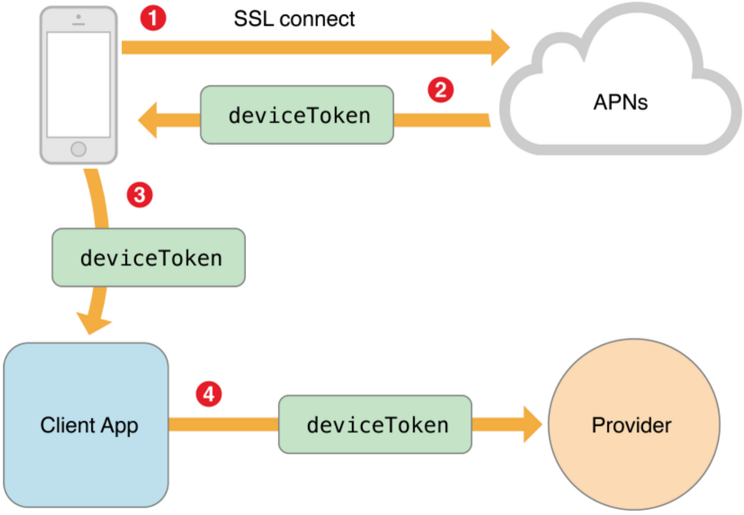


Figure 24: Sharing the device token. Source: [5]

The client device connects to APNs and receives a custom device token. This usually happens when the applications is invoked for the first time. The token is then sent to the client application which sends it to the provider. The provider then sends a notifications to the client application as seen in figure 23.

6.4.2 Push Notifications with Pushbots

The Cordova project lists several open-source and free to use push notification plugin. At the beginning we wanted to use one of these plugins in our application. But although these plugins are backed by a lot of developers, they still have a large number of bugs and are not easy to handle. The most famous plugin "PushPlugin"[27] lists 347 open issues to this day and the newer "Phonegap-

[27] `https://github.com/phonegap-build/PushPlugin`

Plugin-Push"[28] still has 78 open issues on GitHub. When testing the "Phonegap-Plugin-Push" plugin, we encountered so many bugs that we decided to use the free plan of a commercial solution instead.

We decided to use PushBots[29], a project with an integrated Apache Cordova plugin and a Library for sending the messages with Node.js. Sending a push notification through a third party service slightly differs from the approach shown in section 6.4.1 and is shown in Figure 25.

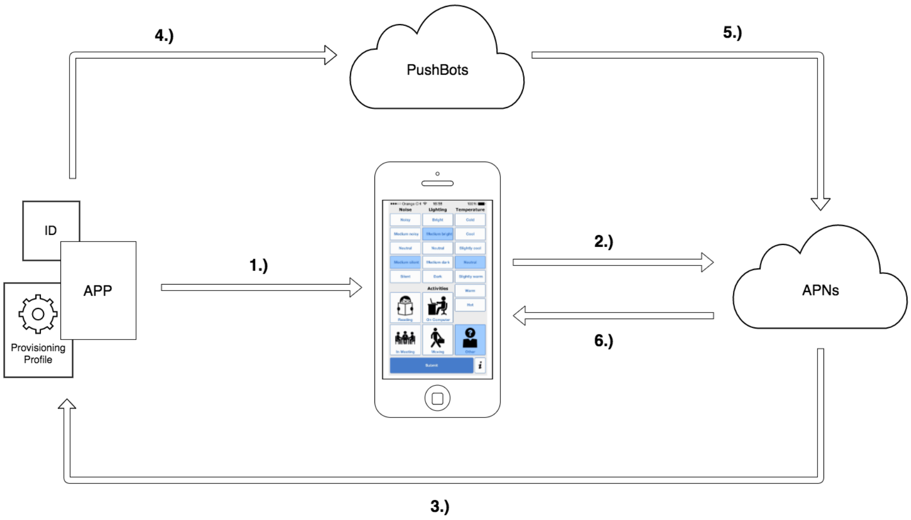


Figure 25: Push notification delivery with third party service. Source:[30]

The first step in Figure 25 is the following: The application sends a request to iOS to send a push notification. iOS then sends a request to APNs after which APNs sends back the device token to the application. Then the applications sends the device token to the PushBots server. After this has happened, you can see the device listed in the PushBots online dashboard. PushBots then is able to send a request (containing a message and a device token) to APNs, which then sends the notification to the device with the specified device token.

In section 6.1 we explained how it is possible to send scheduled messages with PushBots over their Node.js library. With the free plan it is possible to make one million API requests per application and sending up to 1.5 million notifications. The number of supported devices is not regulated.

[28] `https://github.com/phonegap/phonegap-plugin-push`
[29] `https://pushbots.com/`

7 Evaluation

7.1 Test Scenario

The testing of the system can be divided into two parts. One part evaluates how useful the concept of mining comfort related data with a smartphone is. This part covers the quality of data that is collected with the smartphone's sensors but also how the users perceived the usage of the application. The second part investigates in which way participants of mESM studies should be notified. We therefore developed two different notification strategies and analyzed if the strategy has an influence on the number and quality of answers.

We tested the application with twelve users mostly from the Human-IST department and friends over two days. To prevent the results from being biased, it was important that the participants did not know what we wanted to test. To ensure this, only people that weren't involved in the project were asked to join the experiment.

Most ESM and mESM studies use repeated measures as study design. This means that the participants get notified multiple times, during a predefined time period, about a certain state. In our experiment, we asked the user to give us information about his actual perception of comfort.

Due to the relatively small number of participants in the experiment, we chose to conduct a crossover study. Like this, every participant will be tested with the same notification strategy. This should reduce the influence of single users on the outcome of the experiment.

Since we imagined that some users might find the interruption annoying and submit less answers on the second day of the experiment, we expected the experiment to have a high order effect. To antagonize this problem, we used the counterbalancing technique. So half of the group started with the first strategy and the other one with the second strategy on day one. On the second day the strategy changed for each participant.

Name	Group	Push time
...	...	...
User 1	A	Tue Oct 13 15:04
User 1	A	Tue Oct 13 16:04
User 1	A	Tue Oct 13 17:04
User 1	A	Tue Oct 13 18:04
User 1	A	Tue Oct 13 19:04
User 2	B	Tue Oct 13 08:21
User 2	B	Tue Oct 13 09:06
User 2	B	Tue Oct 13 10:54
...	...	...

Table 8: Excerpt of timetable with groups and exact time when the push notification gets sent

Most of the researchers at the Human-IST work in a single bureau. If half the members would be notified at the same time, this could lead to a big disturbance. To prevent this, we decided to notify each user in the same frequency but shift the starting time of each user compared to the others as seen in table 8. The resulting Excel sheet with the notification time for each participant can be seen in figure G.

7.2 User Instructions

To make sure that all participants have the same knowledge about the experiment, it is important that all of them get the exact same instructions. So prior to the execution, everybody will get an email explaining how the experiment is conducted and what is important to remember. To prevent the users from being biased, a false reason for the experiment is communicated.

The instructions are sent via email at the day before the experiment starts. There will be no oral explanation to insure that the received information is the same. A shortened version of the instructions will be available inside the application and accessible via a button. Before the first push notification is sent, the participants are asked by the app if they have read and understood the instructions and that they have familiarized themselves with the interface. The instructions can be seen in Appendix H.

7.3 Distribution

The distribution of the application at a larger scale could be done through the Apple AppStore. But for the experiment conducted in the context of this project, we decided that an AdHoc distribution is sufficient. AdHoc describes the installation on a device that is in range of the developer. But the application is built to be published through the AppStore. To solve this problem, we passed on using unpublished APIs and designed the application according to Apples design guidelines.

Installing a custom application on an iOS device is not as straight-forward as it seems. Before an application can be installed, the developer has to create and a provisioning profile and let apple sign it. The process is shown in figure 26. The provisioning profile is coupled to a single application and a development or production certificate which has to be signed by the developer and apple. Also it contains a list of UDIDs for all applications on which the app will be installed. Each AppID has a corresponding APNs SSL certificate which is used by APNs to identify the applications to which a push notification is sent. In our case, we had to send the APNs certificate to PushBots where they used it to contact APNs.

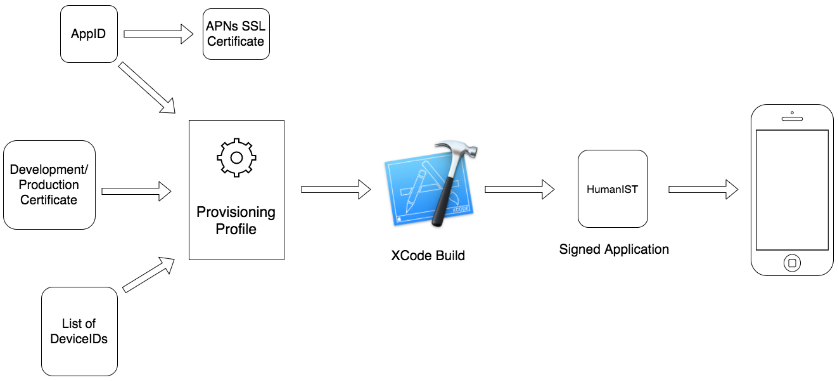


Figure 26: Apple application signing process.

After the provisioning profile is created and downloaded from Apple, the developer can build the application in XCode. This signed application can then be installed on each device whose UDID is specified in the provisioning profile. The transmission of the application can be done through iTunes or XCode.

7.4 Perception of the Users

After the participants conducted the survey, we asked them to fill out a short online survey on the experiment. Unfortunately, only seven of the twelve participants submitted an answer. The results are shown in table 9, 10 and 11.

	Very disturbing	Disturbing	Neutral	Little disturbing	Not disturbing	**Median**
Likert value	5	4	3	2	1	
Regular	1(1)	2	3	1	0	3.8
Random	0	0	2(1)	5	0	2.6

Table 9: User survey on the question on the push strategies.

One of the relationships we wanted to analyze in this project is if the push notification strategy has an influence on the number of answers a user submits. But it is also important to measure how comfortable the participation on the experiment is for the user. In our final survey, we asked the participants which strategy was more disturbing. The results are shown in table 9. One of the participants who had filled out the survey stated that he could not fill out the form often enough to make general assumptions on which day is more disturbing than the other. The answers given are shown in parenthesis and are not used in the statistical analysis of the survey.

We chose to use a Likert-scale with values ranging from ”Very disturbing” to ”Not disturbing”. To interpret a Likert scale statistically, we assigned values ranging from five to one to the verbal scale. To test if one of the strategies has a significant influence on the perception of the participants, we chose to conduct the Mann-Whitney U test. The Mann-Whitney U test, which is also known under the name Wilcoxon Rank sum test, should be chosen if the dependent variable is ordinal and the populations do not form the normal distribution. Also the observations should be independent and the independent variable should consist of two categorical, independent groups[57].

Listing 13: Results of the Exact Wilcoxon-Mann-Whitney Test.

```
data:  v by g (Random, Regular)
Z = -0.44721, p-value = 0.7619
alternative hypothesis: true mu is not equal to 0
```

The corresponding R-script can be found in Appendix B. Median values of 3.8 for the regular strategy and 2.6 for the Random strategy indicate that the random approach is slightly less disturbing than the regular. But the results of the Wilcoxon Rank sum test, which are shown in listing 13, show that this effect is not significant. The p-value of 0.7619 states that the results are not statistically relevant or interpretable. The main reason for this is the small sample size of only seven answers.

	User 3	User 4	User 5	User 6	User 9	User 11	User 12
Regular	(5)	4	3	3	4	3	2
Random	(3)	2	2	2	2	3	2

Table 10: User perception of push strategies per user.

The results for the single users are shown individually in table 10. The comparison with the total number of answers of the single participants, as seen in figure 27, does not give an insight on a correlation of this two values. We suggest to repeat the experiment with a larger number of participants to get statistically significant results.

The last question on the survey asked the participants about the usage of the system. Even though most of them stated that the system is easy to use, most of them would not use the system

	Strongly disagree	Disagree	Neutral	Agree	Strongly Agree
I would use the system frequently	2	2	0	3	0
I think the system is easy to use	0	1	1	1	4

Table 11: User survey on the usage of the system.

frequently. Some users reported that they have been highly interrupted in their daily routine. This shows that the user has to have a higher incentive on using the application or that the number of notifications should be reduced.

8 Quantitative Analysis

8.1 Overview

The two days of collecting data resulted in a dataset containing 1642 entries. An excerpt of the data can be seen in figure 22. But this dataset also contains entries for every connection to the server and some of them did not result in an actual submission of the form. Also entries for every AppID have been saved into the database. A total of 12 people took part in the experiment.

	Day 1	**Day 2**	**Combined**
Raw entries	991	651	1642
Entries without duplicates	968	640	1608
Statistical relevant	726	480	1206
Form submissions	121	80	201
Notifications sent	144	144	288
Submissions/N. sent	84,03%	55.56%	69,79%

Table 12: Received data per day.

Table 12 shows how the entries are composed. Each form submission resulted in eight database entries as seen in section 6.2. Without duplicates, 1608 entries can be found. Since some of the entries like the AppID or the connection of a user are redundant, this leads to 1005 statistically relevant datasets or 201 submissions of the form, each with five corresponding entries. As statistically relevant, we defined each of the four variables in form and the two variables sensed by the ambient light sensor and the microphone.

Figure 27 shows how many times each participant has answered each day. As we can see, there are large differences between single users. The person that answered the most, submitted the form 23 times in total, while the user with the least answers only filled out the form once.

Besides the gap between individual participants, there is a general gap between the first and the second day. As we can see in table 12, there is a decline from 121 answers on day one to 80 answers on day two which comes up to a decrease of 44 percent.

The participants of the experiment have been split up into two groups as seen in section 7 with different notification strategies for each group. The analysis of this different strategies is one of the main goals of this chapter.

8.2 Strategy and Answer Rate

The first hypothesis we are going to examine is that the different push strategies have an influence on the number of answers given by the user. Figure 28 shows the distribution of number of answers and if the corresponding participant is notified by the random or the regular push strategy. If the notification strategy is random, the value on the y-axis has a value of one, if it is regular, the y-axis value is zero. The size of the dots represents the number of participants with the same amount of answers and strategy. So on day two, three people that are notified with the regular strategy answered ten times.

To examine if there is a correlation between the push strategy and the number of answers, we decided to compute Spearman's rank correlation coefficient or Spearman's rho, which is an appropriate statistical method for computing the influence of a binary (strategy) variable on a metric (number of answers) variable.

The result for the first day can be seen in the first row of table 12. Even though there seems to be a small negative influence of the random strategy on the number of answers, the large p-value

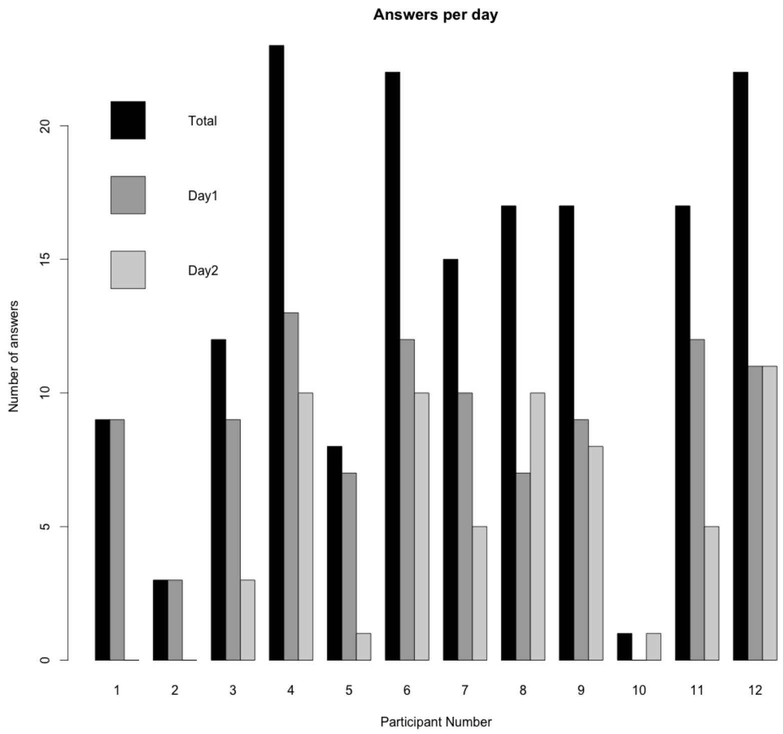


Figure 27: Answers per participant and day.

implies that the significance of the correlation is really low.

The output in the second row of table 12 shows the Spearman rank correlation coefficient for the second day. It suggests that there is a relatively strong negative correlation between the random strategy and the number of answers. But again with a really low significance as the p-value of 0.2088 implies.

The last row of table 12 lists the combination of both days. It shows a slighty negative correlation between the random strategy and the number of answers. The p-value of 0.2086 is almost the same as for the second day which shows that the significance of the correlation is low. The findings can be seen in more detail in Appendix C.

Period	**Spearman's rho**	**P-value**
Day 1	-0.048795	0.8803
Day 2	-0.391059	0.2088
Day 1 & Day 2	-0.2662416	0.2086

Table 13: Influence of the random strategy on the number of answers.

The findings in this section imply that there is a slightly negative correlation between sending push notifications randomly and the answer rates of the participants. But it is possible that other factors have an influence. Two of the participants stated that they were unable to answer to the notifications because of external factors. This fact, together with the already relatively small population, could lead to the big p-values and the associated little significance of the correlation coefficient.

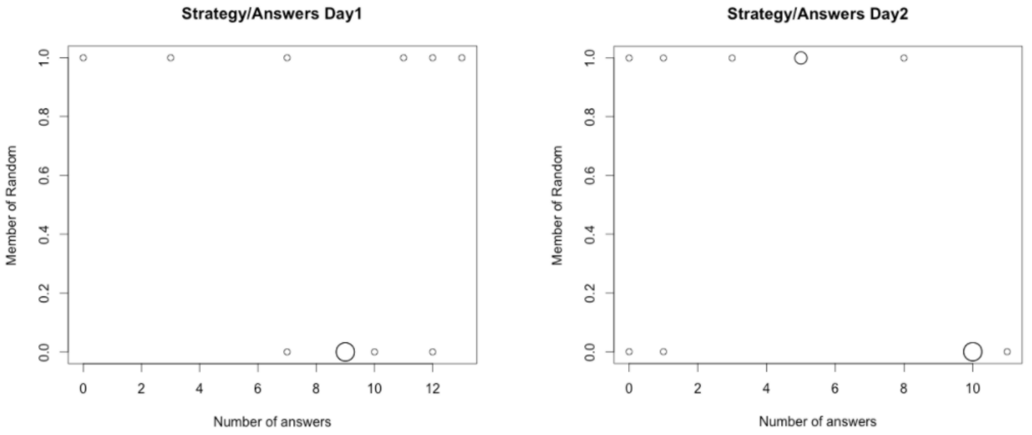


Figure 28: Number of answers to push strategy membership on day one and day two.

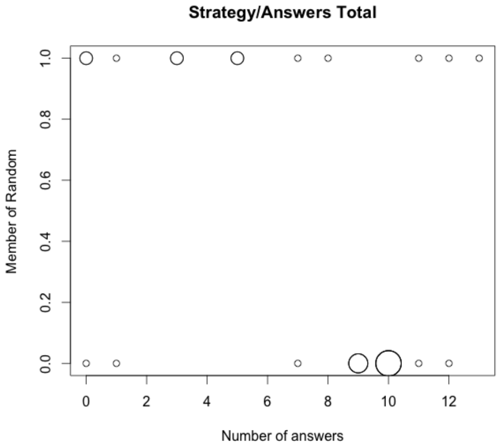


Figure 29: Number of answers to push strategy membership on both days.

8.3 External Factors

To examine if there exists an external influence on our dataset, we additionally performed tests to analyze if there is a difference between the random and the regular strategy. T-test can be used to make assumptions about how equal two given sets of data are. The results are shown in table 14. Since a regular t-test can only be performed on metric scaled data, the appropriate test for the user entered ordinal scaled data is the Wilcoxon rank sum test[43]. The results for the Wilcoxon test are shown in table 15.

	Random mean	**Regular mean**	**T-value**	**P-value**
Noise(Sensor)	75.19155	69.43483	0.34547	0.0097
Light(Sensor)	0.4367627	0.4033423	0.74979	0.4547
Answer Rate	5.666667	8.166667	-1.4576	0.1596

Table 14: Results of the t-test for the regular and the random strategy

The results shown in table 14 show that there is no significant difference between the means of the results of the random and the regular strategy. Even though the p-value for the noise sensor indicates a significant difference between the two means, the t-value of 0.34537 shows that this difference can be neglected. For the other two variables, the the p-value shows that there is no significant difference between the two means.

	W-value	P-value
Noise(User)	2330.0	0.6798
Light(User)	2281.5	0.8498

Table 15: Results of the Wilcoxon rank sum test for the regular and the random strategy

Table 15 shows the results of the Wilcoxon t-test for the data entered by the users. The extremely high p-values show that there is no significant difference between the two means of the data.

If the results in this section had shown a significant difference of population means, it would have been an indicator that there has been an external influence on the collected data. Since the data shows that there is no significant difference between the data collected with the random and the regular strategy, the findings support the results shown in section 8.2 and 8.5.

8.4 Quality of Sensor Values

In this section, the data collected by the microphone and ambient light sensor will be compared to the subjective information given by the users. Since we did not sense the temperature values or the activities, these information won't be evaluated in this project.

Dimension	Spearman's rho	P-value
Visual	0.4504185	2.199e-11
Acoustic	0.4862721	2.19e-13

Table 16: Influence of the random strategy on the number of answers.

To compare an ordinal scale (user values) with a metric scale (sensor values) we can again use Spearman's rank correlation coefficient. The values for all users can be seen in figure 30 and 31 . The plot shows for both variables that most of the values are recorded in the middle of the spectrum and besides a few outlier the sensors seem to reflect the users opinion quite well. This corresponds to the according correlation coefficients shown in listing table 16. The corresponding outputs are listed in appendix D.

With p-values of 2.199e-11 and 2.19e-13, both of them are highly significant and show a good correlation between the sensor and the opinion of the user. This shows that the sensors can be used to measure the human condition on these variables. Nevertheless, it has to be said that there exist differences between the users. While for some, the sensors agreed with the impression of the user, for others there was a bigger discrepancy. It is further possible, that the visualization of the data leads to a bias for certain users. If a users tries to match the answers given through the form with the sensor values or the other way around, it could have an influence on the findings.

8.5 Strategy and Quality of Answers

The second hypothesis to examine is whether or not the different strategies influence the quality of answers. To do so, we decided to only choose participants that answered at least half of the time in average. That meant that we did not include the answers given by the participants one, two, five and ten, as seen in figure 27.

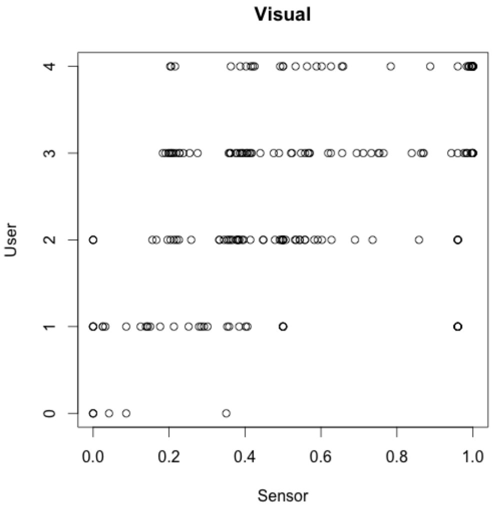


Figure 30: User and Sensor values for the visual dimension.

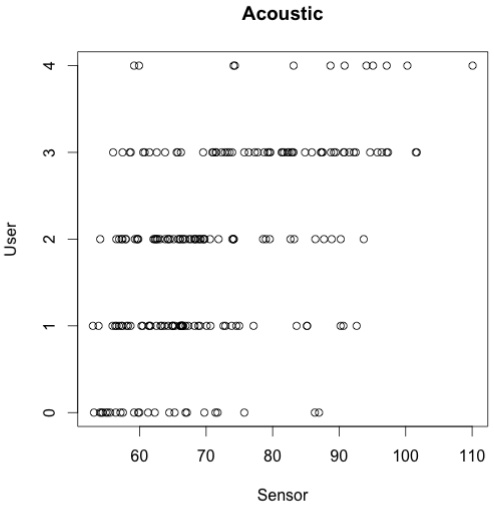


Figure 31: User and Sensor values for the acoustic dimension.

The answers of the remaining eight participants were divided into the groups *Random* and *Regular*. Half of the group received notifications with the random strategy on the first day and the other half on the second. This minimizes a possible bias of the order in which the two strategies are tested. The corresponding R-scripts for the analysis of the regular and random strategies can be found in Appendix I and J.

8.5.1 Regular Strategy

The distribution of answers for the regular strategy can be seen in figure 32 and figure 32. To evaluate the quality, the correlation of the sensor values with the user's impression will be evaluated. Notified with the regular strategy, the participants gave 79 answers in total which equates to 39.3 percent of all answers given on both days.

The correlation coefficients can be seen in table 17. Both values are significant even though

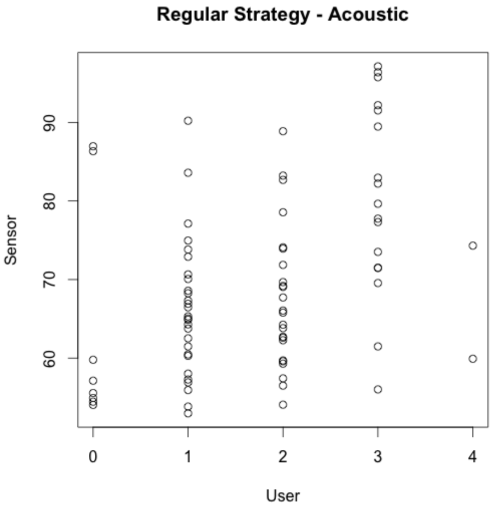


Figure 32: Plot of user and sensor values for the acoustic dimension of the regular strategy.

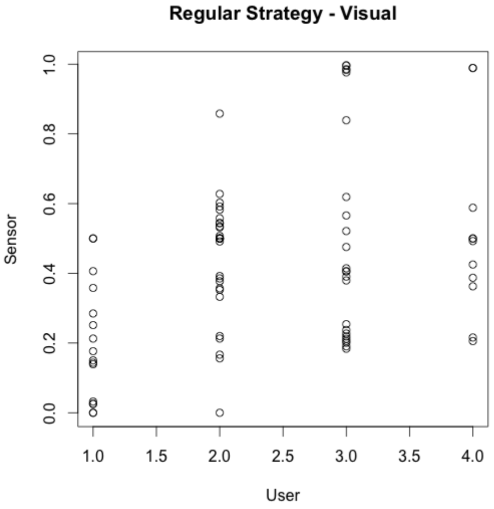


Figure 33: Plot of user and sensor values for the visual dimension of the regular strategy.

the p-values are smaller than in chapter 8.4, which partly is the result of the smaller dataset. The outputs are shown in Appendix E.

Strategy	**Spearman's rho**	**P-value**
Acoustic	0.4096094	0.0001779
Visual	0.3414737	0.002071

Table 17: Influence of the regular strategy on the quality of answers.

While the correlation coefficient for the acoustic quality is almost the same, the differences between the ambient light sensor and the participants perception of lighting are much bigger.

While looking at the data in detail, we found out that one of the users had a constant value for the ambient light sensor. This leads to the conclusion that this particular user had forgotten to turn on the automatic screen brightness feature. Without this user the rho equals 0.4054674.

8.5.2 Random Strategy

The plotting of the data is shown in figure 34 and figure 35. Compared to the the distribution of values in figure 32 and 33, the plots already show a much better correlation.

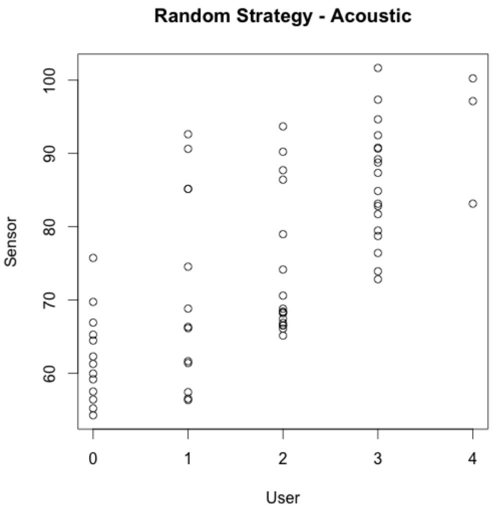


Figure 34: Plot of user and sensor values for the acoustic dimension of the random strategy.

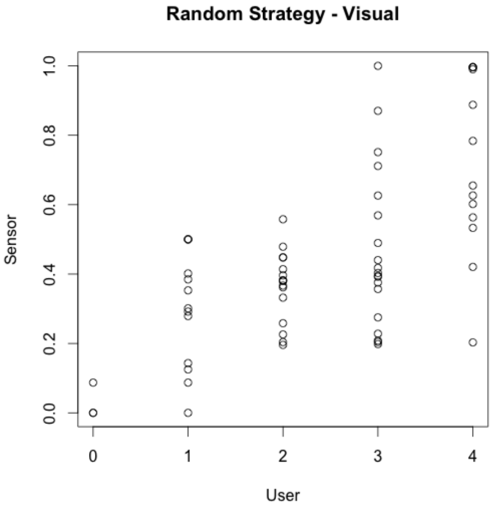


Figure 35: Plot of user and sensor values for the visual dimension of the random strategy.

The total amount of 64 answers for the random strategy is slightly smaller than for it's regular counterpart. This concurs with the findings in section 8.2.

The correlation coefficients with their according p-values are shown in table 18. Both correlation coefficients are much smaller than the corresponding values found with the regular strategy.

Strategy	**Spearman's rho**	**P-value**
Acoustic	0.7033729	9.105e-11
Visual	0.6023368	1.397e-07

Table 18: Influence of the random strategy on the quality of answers.

The p-values of 9.105e-11 and 1.397e-07, at the same time, show a much higher significance. Without the participant with the problematic display brightness setting, the correlation coefficient is even higher with a value of 0.6688247. Detailed outputs are shown in Appendix F.

Although, further research is still needed, the findings in this chapter indicate that the notification strategy has a significant influence on the quality of answers. It should be examined if the difference occurs in a larger population of participants and with another indicator for quality to make general assumptions.

9 Conclusion

The goal of the project was to examine if a smartphone application can be used to collect subjective and objective data on human comfort. As stated in the introduction, we therefore reviewed the current state of research on the topics human comfort, experience sampling and mobile sensing. The groundwork of this part of the paper was then used to build a smartphone application with a corresponding back-end by ourselves that was later tested in a two-days experiment. Besides this proof-of-concept, the experiment should give insights on how to interact with the participants of mobile surveys to get the best possible data.

The application and the corresponding back-end developed for this manner was tested without complications over the whole experiment. The server and database were working without complications and none of the users reported difficulties while using the application. The architecture of the system allows it to be scaled to serve large numbers of users and work together with other external systems.

The analysis of the collected data showed that the built-in smartphone sensors are capable of gathering information on lighting and noise. We expect that mobile sensing will playing an increasing role in sensing human comfort. The growing number of built-in smartphone sensors and people who use this devices, will help to make this approach even more powerful. This comes especially into play when being combined with external sensing systems.

During our research, we found out that the influence of different notification strategies on the quality and the amount of collected data is almost not investigated. We tried to explore the relationship between two different notification strategies and the quality and amount of answers given by the user. We therefore conducted a two-days lasting experiment using the repeated measures design. We chose the correlation between the information entered by the user and the sensor data as an indicator of quality. The results indicate that the strategy has a significant influence on the quality of answers but not on the amount. This findings could be used to gather better information in various fields of research and economy, where mobile experience sampling is used.

At the end of the experiment, we conducted a survey to see if the users felt that one of the strategies was more disturbing than the other, which could not be said with a satisfying significance. We suggest to repeat the experiment with the same application and an enhanced usage of the build in sensors together with a larger population of users.

10 Future Works

The project was designed as a proof-of-concept for a smartphone base comfort sensing application. Additionally, it should give insights on the way a researcher can interact with a possible user of the application. Future versions of the application could collect much more and more accurate data. Especially in the field of Noise metering. Karadous and Shaw[34] have shown that some sensing applications for iOS and Android are capable of sensing even smallest decibel changes.

When it comes to sensing of ambient light, the possibilities are more restricted through the limited publication of API's through the major smartphone manufacturers. But these libraries exist and can be used. If the targeted group of participants is small, such an application could be built without the goal of publishing it through an application store and the number of users is smaller than one hundred.

Further we see great chances for the combination of the application with an external sensing system. The advantages of these systems are that they can consist of a nearly endless number and combination of sensors that go way beyond the capabilities of modern smartphones. Figure 36 shows a possible application together with the sensor boxes built by Léonard Stalder for the Human-IST project "Sensing Human Comfort: An Inclusive Implementation of Indoor Environmental Data Collection".

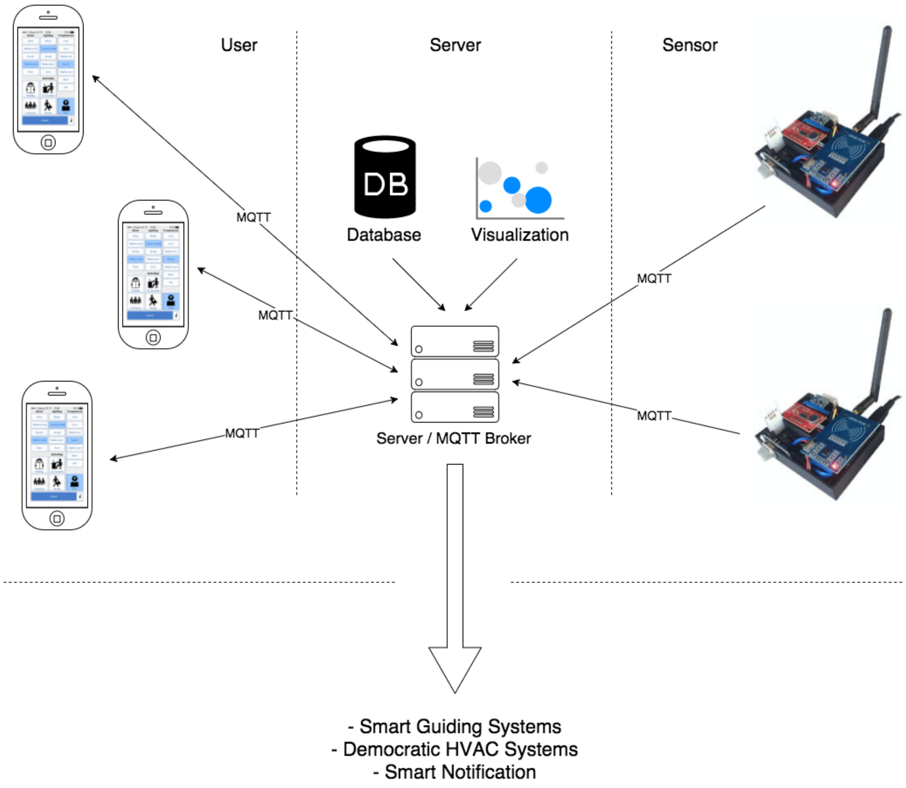


Figure 36: Possible application in an sensing framework with external sensors.

The architecture of the application presented in this project is widely build with a combination of these systems in mind. The transmission of data is built upon the MQTT protocol which makes it easy to combine the smartphone data with sensor data. Triggering the push notification can easily be done through a webapplication that sends them to specific users based on predefined conditions. The publish/subscribe pattern used by the application allows it to be used in multiple systems at once without producing additional data traffic for the user.

The combination of these two systems could be used to enhance both systems. The sensor systems could benefit from the smartphone application in multiple ways. If the users are able to deliver their clothing level or current activity, the system in general could adapt the model and make much more precise predictions on the users state of comfort. The close distance to the user can further enhance the precision of the data inside a room.

The smartphone application on the other hand can profit from sensors that aren't available on the phone. The sensor box presented in the Human-IST project comes with eight different sensors like humidity, wind and CO_2 sensors.

A combination of both opens completely new ways of interaction. If the sensors are placed in multiple rooms, they could send this information to the smartphone application which could then lead the user to the room with the most convenient temperature, humidity, lighting and so on for his own unique profile.

Other fields of application are democratic HVAC systems that could be regulated based on the user input and sensor values combined or smart notification systems that notify the user before he realized that he is outside it is comfort zone. The variables temperature and activity, that we collected in this project, could be used in this matter.

The experiment we conducted with twelve participants showed a big influence of the notification strategy on the quality of answers. This relationship should be further investigated and tested with a larger number of users. If the correlation can be confirmed in larger experiments, this could lead to profound changes in mobile experience sampling and mobile sensing.

References

[1] Apache. Apache cordova. https://cordova.apache.org/. Accessed: 2015-11-24.

[2] Apple. About avfoundation. https://developer.apple.com/library/prerelease/mac/documentation/AudioVideo/Conceptual/AVFoundationPG/Articles/00_Introduction.html#//apple_ref/doc/uid/TP40010188. Accessed: 2015-11-28.

[3] Apple. About local notifications and remote notifications. https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Introduction.html. Accessed: 2015-11-22.

[4] Apple. App store review guidelines. https://developer.apple.com/app-store/review/guidelines. Accessed: 2015-11-29.

[5] Apple. Apple push notification service. https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Chapters/ApplePushService.html. Accessed: 2015-11-10.

[6] Apple. Avaudiorecorder class reference. https://developer.apple.com/library/prerelease/mac/documentation/AVFoundation/Reference/AVAudioRecorder_ClassReference/index.html#//apple_ref/occ/instm/AVAudioRecorder/averagePowerForChannel:. Accessed: 2015-11-28.

[7] Apple. Compare iphone models. http://www.apple.com/iphone/compare/. Accessed: 2015-11-09.

[8] Apple. Health - an innovative new way to use your health and fitness information. http://www.apple.com/ios/health/. Accessed: 2015-11-10.

[9] Areppim. Mobile os percent market share switzerland as of june 2015. http://stats.areppim.com/stats/stats_mobiosxtime_ch.htm. Accessed: 2015-11-27.

[10] ASHRAE. Standard 55. *Thermal Environmental Conditions for Human Occupancy*, 2013.

[11] J. Blauert. Akustik 2. Lecture script at the Ruhr-Universität Bochum, 2005.

[12] Philomena M. Bluyssen, Myriam Aries, and Paula van Dommelen. Comfort of workers in office buildings: The European HOPE project. *Building and Environment*, 2010.

[13] Michael Boduch and Warren Fincher. Standards of Human Comfort - Relative and Absolute. *Meadows Foundation Funded Projects*, 2009.

[14] Vaughn Bradshaw. The building environment: Active and passive control systems. *John Wiley and Sons*, 2006.

[15] K.E. Charles. Fanger's thermal comfort and draught models. *IRC Research Report*, 2003.

[16] Oxford Dictionaries. Smartphone. http://www.oxforddictionaries.com/definition/english/smartphone. Accessed: 2015-11-09.

[17] Anastasios Dounis and Christos Caraiscos. Advanced control systems engineering for energy and comfort management in a building environment—a review. *Renewable and Sustainable Energy Reviews*, 2008.

[18] Tanzeem Choudhury et al. The mobile sensing platform: An embedded activity recognition system. *IEEE Pervasive Computing*, 2008.

[19] Poul O. Fanger. Thermal comfort. *Analysis and applications in environmental engineering.*, 1970.

[20] Monika Frontczak, Rune Vinther Andersen, and Pawel Wargocki. Questionnaire survey on factors influencing comfort with indoor environmental quality in danish housing. *Building and Environment*, 2012.

[21] Monika Frontczak and Pawel Wargocki. Literature survey on how different factors influence human comfort in indoor environments. *Building and Environment*, 2011.

[22] Google. Cloud messaging. `https://developers.google.com/cloud-messaging/`. Accessed: 2015-11-10.

[23] Google. Google fit. `https://fit.google.com`. Accessed: 2015-11-10.

[24] Frédéric Haldi and Darren Robinson. Modelling occupants' personal characteristics for thermal comfort prediction. *International Journal of Biometeorology*, 2010.

[25] Runa Tabea Hellwig. *Unterschiede zwischen frei und mechanisch belufteten Burogebauden aus Nutzersich*. PhD thesis, Technische Universitaät München, 2005.

[26] HiveMQ. Mqtt over websockets with hivemq. `http://www.hivemq.com/blog/mqtt-over-websockets-with-hivemq`. Accessed: 2015-11-29.

[27] Wilhelm Hofmann and Paresh Patel. Surveysignal: A convenient solution for experience sampling research using participants' own smartphones. *Reports and Communications*, 2015.

[28] Jan Holler, Vlasios Tsiatsis, Catherine Mulligan, Stefan Avesand, Stamatis Karnouskos, and David Boyle. *From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence*. Elsevier, 2014.

[29] IDC. Smartphone vendor market share, 2015 q2. `http://www.idc.com/prodserv/smartphone-market-share.jsp`. Accessed: 2015-11-10.

[30] Tomomi Imura. Sending ios push notifications via apns in javascript. `https://www.pubnub.com/blog/2014-12-22-sending-ios-push-notifications-via-apns-javascript-using/apns-phonegap/`. Accessed: 2015-12-04.

[31] Farrokh Jazizadeh, Ali Ghahramani, Burcin Becerik-Gerber, Tatiana Kichkaylo, and Michael Orosz. Human-Building Interaction Framework for Personalized Thermal Comfort-Driven Systems in Office Buildings. *Journal of Computing in Civil Engineering*, 2013.

[32] Jeongho Kang and Sekwang Park. Development of comfort sensing system for human environment. *Mechatronics*, 1998.

[33] Christian Karasiewicz. Why facebook is using mqtt on mobile. *The Mobile Frontier*, 2013.

[34] Chucri A. Kardous and Peter B. Shaw. Evaluation of smartphone sound measurement applications. *The Journal of the Acoustical Society of America*, 2014.

[35] Van Den Wymelenberg Kevin G. *Evaluating Human Visual Preference and Performance in an Office Environment Using Luminance-based Metrics*. PhD thesis, University of Washington, 2012.

[36] Wazir Zada Khan, Yang Xiang, Mohammed Y. Aalsalem, and Quratulain Arshad. Mobile phone sensing systems: A survey. *IEEE Communications Surveys and Tutorials*, 2013.

[37] Igor Knez. Effects of indoor lighting on mood and cognition. *Journal of Environmental Psychology*, 1995.

[38] Valerie Lampkin, Weng Tat Leong, Leonardo Olivera, Sweta Rawat, Nagesh Subrahmanyam, and Rong Xiang. *Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry*. IBM Redbooks, 2012.

[39] Nicholas D. Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T. Campbell. A survey of mobile phone sensing. *IEEE Communcations Magazine*, 2010.

[40] Adrian Leaman. Dissatisfaction and office productivity. *Facilities*, 1995.

[41] Friedrich Linhart and Jean-Louis Scartezzini. Evening office lighting e visual comfort vs. energy efficiency vs. performance? *Building and Environment*, 2011.

[42] Carol Lomonaco and Dennis Miller. Comfort and control in the workspace. *ASHRAE Journal*, 1997.

[43] Universität Zürich Methodenberatung. Datenanalyse: Zentrale tendenz. `http://www.methodenberatung.uzh.ch/datenanalyse/unterschiede/zentral.html#13`. Accessed: 2016-01-02.

[44] Mosquitto. Mosquitto. `http://mosquitto.org/`. Accessed: 2015-11-23.

[45] Giorgio Natili. *PhoneGap 3 Beginner's Guide*. Packt Publishing, 2013.

[46] Oasis open. Mqtt 3.1.1 specifications. `http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html`. Accessed: 2015-12-04.

[47] Richard Paradis. Acoustic comfort. *National Institute of Building Sciences: Whole Building Design Guide*, 2014.

[48] Willy Passchier-Vermeer and Wim F. Passchier. Noise exposure and public health. *Environmental Health Perspectives*, 2000.

[49] Veljko Pejovic, Neal Lathia, Cecilia Mascolo, and Mirco Musolesi. Mobile-based experience samping for behaviour research. *Pre-print*, 2015. arXiv:1508.03725.

[50] Adobe Phonegap. Plugin development guide. `http://docs.phonegap.com/en/edge/guide_hybrid_plugins_index.md.html#Plugin%20Development%20Guide`. Accessed: 2015-11-25.

[51] Erin Pierce. Mobile push notification strategy: Tips for engaging and retaining users. `http://blogs.adobe.com/digitalmarketing/digital-marketing/mobile-push-notification-strategy-tips-for-engaging-and-retaining-users/`. Accessed: 2015-11-02.

[52] Bluyssen PM, Fossati S, Mandin C, Cattaneo A, and Carrer P. Towards a new procedure for identifying causes of health and comfort problems in office buildings. *ISIAQ. 10th International Conference on Healthy Buildings 2012. Proceedings of a meeting held 8–12 July 2012*, 2012.

[53] Michaela Riediger. Experience sampling. *German Data Forum, Building on Progress. Expanding the research infrastructure for the social, economic, and behavioral sciences*, 2010.

[54] Samsung. Galaxy s6. `http://www.samsung.com/hk_en/consumer/mobile/smartphones/smartphones/SM-G9250ZWATGY`. Accessed: 2015-11-09.

[55] Christie N. Scollon, Chu Kim-Prieto, and Ed Diener. Experience sampling: Promises and pitfalls, strenghts and weaknesses. *Journal of Happiness Studies*, 2003.

[56] C. J. Simonson, M. Salonvaara, and T. Ojanen. The effect of structures on indoor humidity – possibility to improve comfort and perceived air quality. *Indoor Air*, 2002.

[57] Laerd Statistics. Mann-whitney u test using spss statistics. `https://statistics.laerd.com/spss-tutorials/mann-whitney-u-test-using-spss-statistics.php`. Accessed: 2015-12-05.

[58] SWITCH. About us. `https://www.switch.ch/about/foundation/`. Accessed: 2015-12-04.

[59] Dale Tiller, Lily M. Wang, Amy Musser, and Matthew Radik. Combined effects of noise and temperature on human comfort and performance. *Architectural Engineering*, 2010.

[60] Andrew Trice. Phonegap advice on dealing with apple application rejections. `http://www.adobe.com/devnet/phonegap/articles/apple-application-rejections-and-phonegap-advice.html`. Accessed: 2015-11-25.

[61] Phillip J. Walsh, Charles S. Dudney, and Emily D. Copenhaver. *Indoor Air Quality*. CRC Press, 1983.

[62] Danni Wang, Edward Arens annd Tom Webster, and Mingyu Shi. How the number and placement of sensors controlling room air distribution systems affect energy use and comfort. *Energy Systems Laboratory*, 2002.

[63] John M. Wargo. *PhoneGap Essentials Building Cross-Platform Mobile Apps*. Addison-Wesley, 2012.

[64] John M. Wargo. *Apache Cordova 3 Programming*. Addison-Wesley, 2013.

[65] Bill Williams. Footcandles and lux for architectural lighting - an introduction to illuminance. `http://www.mts.net/~william5/library/illum.htm`. Accessed: 2015-11-02.

Appendices

This part of the document contains code listings and tables that could not be placed inside the main text or would disturb the normal reading flow. Also, additional information on the statistical analysis and technical implementation of the application is shown.

A Application: Implementation of the unique ID generator

Listing 14: Implementation of the unique ID generator

```
        var IDlist = ’0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ’;
        var IDlength = 8;
        var createID = function() {
          var rtn = ’’;
          for (var i = 0; i < IDlength; i++) {
            rtn += IDlist.charAt(Math.floor(Math.random() * IDlist.length));
          }
          return rtn;
        }
        appID = createID();
        window.localStorage.setItem(’appUID’, appID);
```

B R Script: Mann-Whitney U Test on the User Perception of Strategy

Listing 15: Regular Strategy and Quality of Answers

```

#Which strategy is more disturbing?

#Very disturbing & Disturbing & Neutral & Little disturbing & Not disturbing
#       5               4          3              2                 1

Regular = c(0,2,3,1,0)
Random =  c(0,0,1,5,0)
wilcox.test(Regular,Random)

#Refactor to deal with Ties
install.packages("coin")
library(coin)
g = factor(c(rep("Regular", length(Regular)), rep("Random", length(Random))))
v = c(Regular, Random)
wilcox_test(v ~ g, distribution="exact")
```

C R Output - Detailed outputs for strategy and answer rate

Listing 16: Spearman's rho for strategy on the first day

```
data:  AnswersDay1 and Day1Group
S = 299.96, p-value = 0.8803
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
-0.048795
```

Listing 17: Spearman's rho for strategy on the second day

```
data:  AnswersDay2 and Day2Group
S = 397.84, p-value = 0.2088
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
-0.391059
```

Listing 18: Spearman's rho for strategy on both days combined

```
data:  RegRand and BinaryRandom
S = 2912.4, p-value = 0.2086
alternative hypothesis: true rho is not equal to 0
sample estimates:
       rho
-0.2662416
```

D R Output - Detailed outputs for the quality of the sensor-data

Listing 19: Spearman's rho for visual quality

```
data:  LightFrame$userL and LightFrame$sensorL
S = 732760, p-value = 2.199e-11
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.4504185
```

Listing 20: Spearman's rho for acoustic quality

```
data:  NoiseFrame$userC and NoiseFrame$sensorC
S = 705710, p-value = 2.19e-13
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.4862721
```

E R Output - Detailed outputs for the quality of sensors for the regular strategy

Listing 21: Regular strategy: Acoustic quality

```
data:  RegularDataframe2$RegularNoise2 and RegularDataframe2$RegularNoiseS2
S = 48506, p-value = 0.0001779
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.4096094
```

Listing 22: Regular strategy: Visual quality

```
data:  RegularDataframe2$RegularLighting2 and RegularDataframe2$RegularLightS2
S = 54105, p-value = 0.002071
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.3414737
```

F R Output - Detailed outputs for the quality of sensors for the random strategy

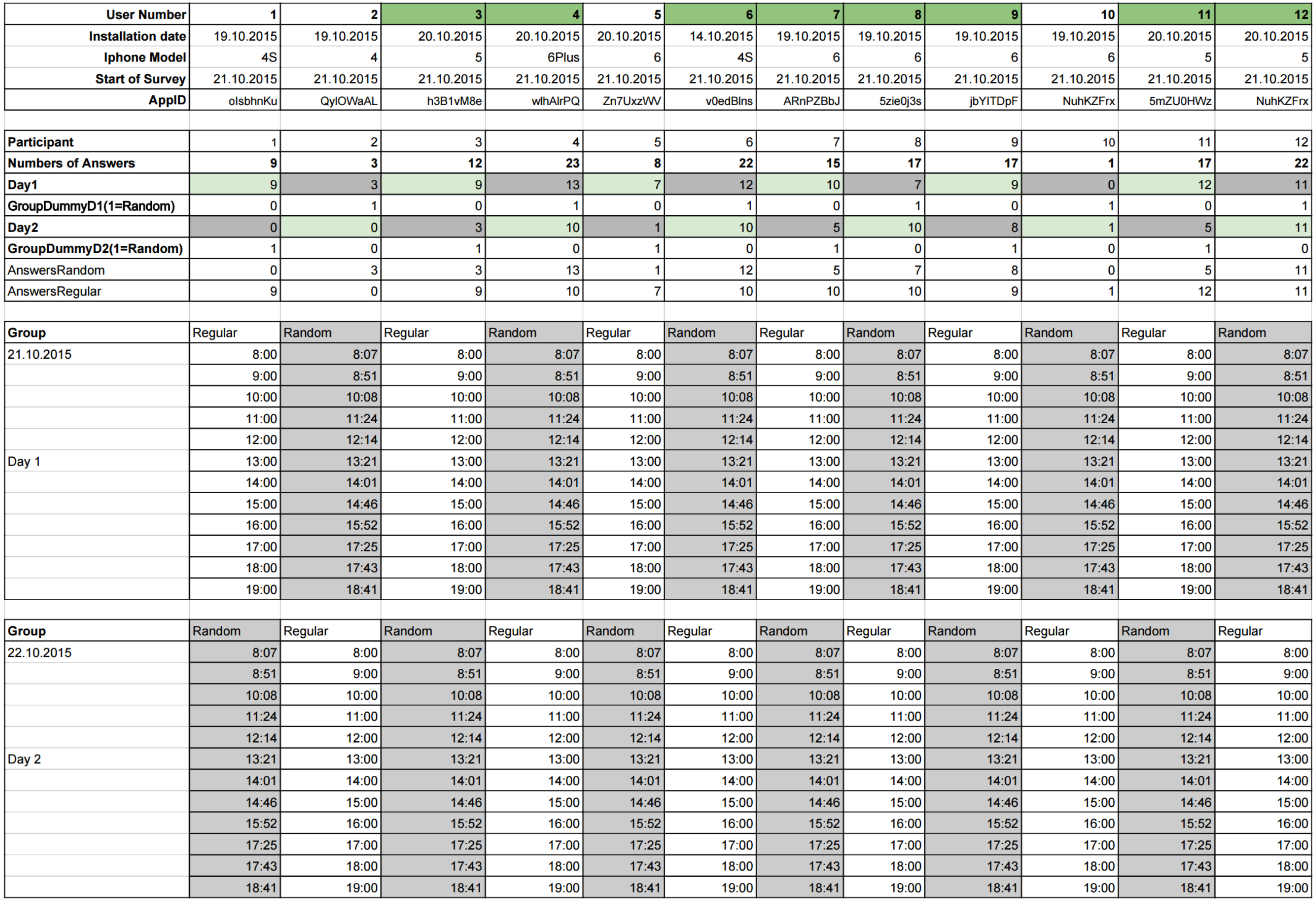
Listing 23: Random strategy: Acoustic quality

```
data:  RandomDataframe$RandomNoise and RandomDataframe$RandomNoiseS
S = 12957, p-value = 9.105e-11
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.7033729
```

Listing 24: Random strategy: Visual quality

```
data:  RandomDataframe$RandomLighting and RandomDataframe$RandomLightS
S = 17370, p-value = 1.397e-07
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.6023368
```

G Excel Table with User Information and the Exact Notification Date



User Number	1	2	3	4	5	6	7	8	9	10	11	12
Installation date	19.10.2015	19.10.2015	20.10.2015	20.10.2015	20.10.2015	14.10.2015	19.10.2015	19.10.2015	19.10.2015	19.10.2015	20.10.2015	20.10.2015
Iphone Model	4S	4	5	6Plus	6	4S	6	6	6	6	5	5
Start of Survey	21.10.2015	21.10.2015	21.10.2015	21.10.2015	21.10.2015	21.10.2015	21.10.2015	21.10.2015	21.10.2015	21.10.2015	21.10.2015	21.10.2015
AppID	olsbhnKu	QylOWaAL	h3B1vM8e	wlhAlrPQ	Zn7UxzWV	v0edBlns	ARnPZBbJ	5zie0j3s	jbYITDpF	NuhKZFrx	5mZU0HWz	NuhKZFrx

Participant	1	2	3	4	5	6	7	8	9	10	11	12
Numbers of Answers	9	3	12	23	8	22	15	17	17	1	17	22
Day1	9	3	9	13	7	12	10	7	9	0	12	11
GroupDummyD1(1=Random)	0	1	0	1	0	1	0	1	0	1	0	1
Day2	0	0	3	10	1	10	5	10	8	1	5	11
GroupDummyD2(1=Random)	1	0	1	0	1	0	1	0	1	0	1	0
AnswersRandom	0	3	3	13	1	12	5	7	8	0	5	11
AnswersRegular	9	0	9	10	7	10	10	10	9	1	12	11

Group	Regular	Random	Regular	Random	Regular	Random	Regular	Random	Regular	Random	Regular	Random
21.10.2015	8:00	8:07	8:00	8:07	8:00	8:07	8:00	8:07	8:00	8:07	8:00	8:07
	9:00	8:51	9:00	8:51	9:00	8:51	9:00	8:51	9:00	8:51	9:00	8:51
	10:00	10:08	10:00	10:08	10:00	10:08	10:00	10:08	10:00	10:08	10:00	10:08
	11:00	11:24	11:00	11:24	11:00	11:24	11:00	11:24	11:00	11:24	11:00	11:24
	12:00	12:14	12:00	12:14	12:00	12:14	12:00	12:14	12:00	12:14	12:00	12:14
Day 1	13:00	13:21	13:00	13:21	13:00	13:21	13:00	13:21	13:00	13:21	13:00	13:21
	14:00	14:01	14:00	14:01	14:00	14:01	14:00	14:01	14:00	14:01	14:00	14:01
	15:00	14:46	15:00	14:46	15:00	14:46	15:00	14:46	15:00	14:46	15:00	14:46
	16:00	15:52	16:00	15:52	16:00	15:52	16:00	15:52	16:00	15:52	16:00	15:52
	17:00	17:25	17:00	17:25	17:00	17:25	17:00	17:25	17:00	17:25	17:00	17:25
	18:00	17:43	18:00	17:43	18:00	17:43	18:00	17:43	18:00	17:43	18:00	17:43
	19:00	18:41	19:00	18:41	19:00	18:41	19:00	18:41	19:00	18:41	19:00	18:41

Group	Random	Regular	Random	Regular	Random	Regular	Random	Regular	Random	Regular	Random	Regular
22.10.2015	8:07	8:00	8:07	8:00	8:07	8:00	8:07	8:00	8:07	8:00	8:07	8:00
	8:51	9:00	8:51	9:00	8:51	9:00	8:51	9:00	8:51	9:00	8:51	9:00
	10:08	10:00	10:08	10:00	10:08	10:00	10:08	10:00	10:08	10:00	10:08	10:00
	11:24	11:00	11:24	11:00	11:24	11:00	11:24	11:00	11:24	11:00	11:24	11:00
	12:14	12:00	12:14	12:00	12:14	12:00	12:14	12:00	12:14	12:00	12:14	12:00
Day 2	13:21	13:00	13:21	13:00	13:21	13:00	13:21	13:00	13:21	13:00	13:21	13:00
	14:01	14:00	14:01	14:00	14:01	14:00	14:01	14:00	14:01	14:00	14:01	14:00
	14:46	15:00	14:46	15:00	14:46	15:00	14:46	15:00	14:46	15:00	14:46	15:00
	15:52	16:00	15:52	16:00	15:52	16:00	15:52	16:00	15:52	16:00	15:52	16:00
	17:25	17:00	17:25	17:00	17:25	17:00	17:25	17:00	17:25	17:00	17:25	17:00
	17:43	18:00	17:43	18:00	17:43	18:00	17:43	18:00	17:43	18:00	17:43	18:00
	18:41	19:00	18:41	19:00	18:41	19:00	18:41	19:00	18:41	19:00	18:41	19:00

H Instructions on how to use the ComfortIST application

The purpose of the survey is to find out if there is a relationship between objective data measured by the built in smartphone sensors and the perceived comfort. To collect this data you are asked several times a day to fill out a small form. This happens via push notifications on your smartphone. If you take part in the survey please keep the following points in mind:

- It is important for the survey that you answer as fast as possible. If you have **missed a single notification** please answer as soon as you can. If you have **missed more than one** notification please only answer once.
- To gather objective information, the application needs to collect data from the built-in sensors. Therefore, it accesses the luminosity sensor and the microphone. This information, together with the data entered into the form, is **sent to a server**.
- Please enter your **first name** starting with a capital letter the first time you've opened the application.
- To participate in the experiment it is important that you have turned on the **Auto-Brightness** feature of your iPhone and that you have enabled **Push Notifications**. The app also needs the permission to access the **microphone**. No audio files are saved or sent from the app. The microphone only measures the sound pressure.
- As soon as you have pressed the submit button, you get prompted to a page with a small visualization of the data you have entered so far. You can **leave the app** at any time by pressing the home button.
- The application sends push notifications **from 8 AM to 7 PM for two days**. If you like to leave the experiment, just delete the application from your device.
- If you have any questions, please feel free to write an email to **romanrick.kuepper@unifr.ch**.

I R Script: Regular Strategy and Quality of Answers

Listing 25: Regular Strategy and Quality of Answers

```

library(RMySQL)
mydb = dbConnect(MySQL(), user=’root’, password=’Unifr2020!’, dbname=’newComfort’, host=’localhost’)
dbListFields(mydb, ’Comfort’)

##############Regular:
#agnes: h3B1vM8e
#pierre: wlhAlrPQ d2
#michelle: v0edBlns d2
#marie: ARnPZBbJ
#carlo: 5zie0j3s d2
#rudi: jbYITDpF
#florian: 5mZU0HWz
#benji: NuhKZFrx d2

pierreD2 = dbGetQuery(mydb, "select * from Comfort where Sensor like ’iphone/wlhAlrPQ/noise’ and Date like ’2015-10-22 %’ or Sensor like ’iphone/wlhAlrPQ/noiseS’ and Date like ’2015-10-22 %’ or Sensor like ’iphone/wlhAlrPQ/lighting’ and Date like ’2015-10-22 %’ or Sensor like ’iphone/wlhAlrPQ/lightS’ and Date like ’2015-10-22 %’;")
michelleD2 = dbGetQuery(mydb, "select * from Comfort where Sensor like ’iphone/v0edBlns/noise’ and Date like ’2015-10-22 %’ or Sensor like ’iphone/v0edBlns/noiseS’ and Date like ’2015-10-22 %’ or Sensor like ’iphone/v0edBlns/lighting’ and Date like ’2015-10-22 %’ or Sensor like ’iphone/v0edBlns/lightS’ and Date like ’2015-10-22 %’;")
carloD2 = dbGetQuery(mydb, "select * from Comfort where Sensor like ’iphone/5zie0j3s/noise’ and Date like ’2015-10-22 %’ or Sensor like ’iphone/5zie0j3s/noiseS’ and Date like ’2015-10-22 %’ or Sensor like ’iphone/5zie0j3s/lighting’ and Date like ’2015-10-22 %’ or Sensor like ’iphone/5zie0j3s/lightS’ and Date like ’2015-10-22 %’;")
benjiD2 = dbGetQuery(mydb, "select * from Comfort where Sensor like ’iphone/NuhKZFrx/noise’ and Date like ’2015-10-22 %’ or Sensor like ’iphone/NuhKZFrx/noiseS’ and Date like ’2015-10-22 %’ or Sensor like ’iphone/NuhKZFrx/lighting’ and Date like ’2015-10-22 %’ or Sensor like ’iphone/NuhKZFrx/lightS’ and Date like ’2015-10-22 %’;")
marieD1 = dbGetQuery(mydb, "select * from Comfort where Sensor like ’iphone/ARnPZBbJ/noise’ and Date like ’2015-10-21 %’ or Sensor like ’iphone/ARnPZBbJ/noiseS’ and Date like ’2015-10-21 %’ or Sensor like ’iphone/ARnPZBbJ/lighting’ and Date like ’2015-10-21 %’ or Sensor like ’iphone/ARnPZBbJ/lightS’ and Date like ’2015-10-21 %’;")
rudiD1 = dbGetQuery(mydb, "select * from Comfort where Sensor like ’iphone/jbYITDpF/noise’ and Date like ’2015-10-21 %’ or Sensor like ’iphone/jbYITDpF/noiseS’ and Date like ’2015-10-21 %’ or Sensor like ’iphone/jbYITDpF/lighting’ and Date like ’2015-10-21 %’ or Sensor like ’iphone/jbYITDpF/lightS’ and Date like ’2015-10-21 %’;")
florianD1 = dbGetQuery(mydb, "select * from Comfort where Sensor like ’iphone/5mZU0HWz/noise’ and Date like ’2015-10-21 %’ or Sensor like ’iphone/5mZU0HWz/noiseS’ and Date like ’2015-10-21 %’ or Sensor like ’iphone/5mZU0HWz/lighting’ and Date like ’2015-10-21 %’ or Sensor like ’iphone/5mZU0HWz/lightS’ and Date like ’2015-10-21 %’;")
agnesD1 = dbGetQuery(mydb, "select * from Comfort where Sensor like ’iphone/h3B1vM8e/noise’ and Date like ’2015-10-21 %’ or Sensor like ’iphone/h3B1vM8e/noiseS’ and Date like ’2015-10-21 %’ or Sensor like ’iphone/h3B1vM8e/lighting’ and Date like ’2015-10-21 %’ or Sensor like ’iphone/h3B1vM8e/lightS’ and Date like ’2015-10-21 %’;")

RegularCharacter2 = rbind(pierreD2, michelleD2, carloD2, benjiD2, marieD1, rudiD1, florianD1, agnesD1)
RegularCharacter2
typeof(RegularCharacter2)

#Regular to data.frame
Regular2 <- transform(RegularCharacter2, Value = as.numeric(Value))
Regular2
```

```r
RegularValue2 <- Regular2$Value
RegularValue2

#### the data received at the 2015-10-21 12:14:03 is corrupted
#### delete corrupted data
RegularValue2 = RegularValue2[-257]
RegularValue2[257]
RegularValue2 = RegularValue2[-257]
RegularValue2[257]
RegularValue2 = RegularValue2[-257]
RegularValue2[257]
RegularValue2 = RegularValue2[-257]
RegularValue2[257]
RegularValue2 = RegularValue2[-257]
RegularValue2[257]
RegularValue2 = RegularValue2[-257]
RegularValue2[257]
RegularValue2 = RegularValue2[-257]
RegularValue2[257]
#####

RegularValue2
RegularValue2[250:270]

length(RegularValue2)

RegularNoise2 <- RegularValue2[seq(1, length(RegularValue2), 4)]
RegularNoise2
RegularLighting2 <- RegularValue2[seq(2, length(RegularValue2)-1, 4)]
RegularLighting2
RegularNoiseS2 <- RegularValue2[seq(3, length(RegularValue2), 4)]
RegularNoiseS2
#####Error correction
mean(RegularNoiseS2)
RegularNoiseS2[55]
RegularNoiseS2[55]= mean(RegularNoiseS2)
RegularNoiseS2
######
RegularLightS2 <- RegularValue2[seq(4, length(RegularValue2), 4)]
RegularLightS2

length(RegularNoise2)
length(RegularLighting2)
length(RegularNoiseS2)
length(RegularLightS2)

RegularDataframe2 <- data.frame(RegularNoise2,RegularNoiseS2, RegularLighting2, RegularLightS2)
RegularDataframe2
#Boxplot Regular
plot(RegularDataframe2)

#correlation Regular
CorRegularNoise2 <- cor.test (RegularDataframe2$RegularNoise2, RegularDataframe2$RegularNoiseS2, method = "spearman") #spearman with according p-value
CorRegularNoise2

CorRegularLight2 <- cor.test (RegularDataframe2$RegularLighting2, RegularDataframe2$RegularLightS2, method = "spearman") #spearman with according p-value
CorRegularLight2
```

J R Script: Random Strategy and Quality of Answers

Listing 26: Random Strategy and Quality of Answers

```

library(RMySQL)
mydb = dbConnect(MySQL(), user=’root’, password=’Unifr2020!’, dbname=’newComfort’, host=’localhost’)
dbListFields(mydb, ’Comfort’)

##############Regular:
#agnes: h3B1vM8e
#pierre: wlhAlrPQ d2
#michelle: v0edBlns d2
#marie: ARnPZBbJ
#carlo: 5zie0j3s d2
#rudi: jbYITDpF
#florian: 5mZU0HWz
#benji: NuhKZFrx d2

pierre = dbGetQuery(mydb, "select * from Comfort where Sensor like ’iphone/wlhAlrPQ/noise’ and Date like ’2015-10-21 %’ or Sensor like ’iphone/wlhAlrPQ/noiseS’ and Date like ’2015-10-21 %’ or Sensor like ’iphone/wlhAlrPQ/lighting’ and Date like ’2015-10-21 %’ or Sensor like ’iphone/wlhAlrPQ/lightS’ and Date like ’2015-10-21 %’;")
michelle = dbGetQuery(mydb, "select * from Comfort where Sensor like ’iphone/v0edBlns/noise’ and Date like ’2015-10-21 %’ or Sensor like ’iphone/v0edBlns/noiseS’ and Date like ’2015-10-21 %’ or Sensor like ’iphone/v0edBlns/lighting’ and Date like ’2015-10-21 %’ or Sensor like ’iphone/v0edBlns/lightS’ and Date like ’2015-10-21 %’;")
carlo = dbGetQuery(mydb, "select * from Comfort where Sensor like ’iphone/5zie0j3s/noise’ and Date like ’2015-10-21 %’ or Sensor like ’iphone/5zie0j3s/noiseS’ and Date like ’2015-10-21 %’ or Sensor like ’iphone/5zie0j3s/lighting’ and Date like ’2015-10-21 %’ or Sensor like ’iphone/5zie0j3s/lightS’ and Date like ’2015-10-21 %’;")
benji = dbGetQuery(mydb, "select * from Comfort where Sensor like ’iphone/NuhKZFrx/noise’ and Date like ’2015-10-21 %’ or Sensor like ’iphone/NuhKZFrx/noiseS’ and Date like ’2015-10-21 %’ or Sensor like ’iphone/NuhKZFrx/lighting’ and Date like ’2015-10-21 %’ or Sensor like ’iphone/NuhKZFrx/lightS’ and Date like ’2015-10-21 %’;")
marieD2 = dbGetQuery(mydb, "select * from Comfort where Sensor like ’iphone/ARnPZBbJ/noise’ and Date like ’2015-10-22 %’ or Sensor like ’iphone/ARnPZBbJ/noiseS’ and Date like ’2015-10-22 %’ or Sensor like ’iphone/ARnPZBbJ/lighting’ and Date like ’2015-10-22 %’ or Sensor like ’iphone/ARnPZBbJ/lightS’ and Date like ’2015-10-22 %’;")
rudiD2 = dbGetQuery(mydb, "select * from Comfort where Sensor like ’iphone/jbYITDpF/noise’ and Date like ’2015-10-22 %’ or Sensor like ’iphone/jbYITDpF/noiseS’ and Date like ’2015-10-22 %’ or Sensor like ’iphone/jbYITDpF/lighting’ and Date like ’2015-10-22 %’ or Sensor like ’iphone/jbYITDpF/lightS’ and Date like ’2015-10-22 %’;")
florianD2 = dbGetQuery(mydb, "select * from Comfort where Sensor like ’iphone/5mZU0HWz/noise’ and Date like ’2015-10-22 %’ or Sensor like ’iphone/5mZU0HWz/noiseS’ and Date like ’2015-10-22 %’ or Sensor like ’iphone/5mZU0HWz/lighting’ and Date like ’2015-10-22 %’ or Sensor like ’iphone/5mZU0HWz/lightS’ and Date like ’2015-10-22 %’;")
agnesD2 = dbGetQuery(mydb, "select * from Comfort where Sensor like ’iphone/h3B1vM8e/noise’ and Date like ’2015-10-22 %’ or Sensor like ’iphone/h3B1vM8e/noiseS’ and Date like ’2015-10-22 %’ or Sensor like ’iphone/h3B1vM8e/lighting’ and Date like ’2015-10-22 %’ or Sensor like ’iphone/h3B1vM8e/lightS’ and Date like ’2015-10-22 %’;")

RandomCharacter = rbind(pierre, michelle, carlo, benji, marieD2, rudiD2, florianD2, agnesD2)
RandomCharacter
typeof(RandomCharacter)

#Regular to data.frame
Random <- transform(RandomCharacter, Value = as.numeric(Value))
Random
```

```r
RandomValue <- Random$Value
RandomValue

RandomNoise <- RandomValue[seq(1, length(RandomValue), 4)]
RandomNoise
RandomLighting <- RandomValue[seq(2, length(RandomValue)-1, 4)]
RandomLighting
RandomNoiseS <- RandomValue[seq(3, length(RandomValue), 4)]
RandomNoiseS
RandomLightS <- RandomValue[seq(4, length(RandomValue), 4)]
RandomLightS

length(RandomNoise)
length(RandomLighting)
length(RandomNoiseS)
length(RandomLightS)

RandomDataframe <- data.frame(RandomNoise,RandomNoiseS, RandomLighting, RandomLightS)
RandomDataframe
#Boxplot Random
plot(RandomDataframe)

#correlation Random
CorRandomNoise <- cor.test (RandomDataframe$RandomNoise, RandomDataframe$RandomNoiseS, method = "spearman") #spearman with according p-value
CorRandomNoise

CorRandomLight <- cor.test (RandomDataframe$RandomLighting, RandomDataframe$RandomLightS, method = "spearman") #spearman with according p-value
CorRandomLight
```

K Server: Implementation of the Eclipse Paho Python client

Listing 27: Python Script that stores incoming MQTT messages into a MySQL database

```

import paho.mqtt.client as mqtt
import MySQLdb

con = MySQLdb.connect(host="localhost",
                      user="root",
                      passwd="Unifr2020!",
                      db="14Sept15")

# The callback for when the client receives a CONNACK response from the server.
def on_connect(client, userdata, rc):
    print("Connected with result code "+str(rc))
    client.subscribe("#")

# The callback for when a PUBLISH message is received from the server.
def on_message(client, userdata, msg):
    with con:
        cur = con.cursor();
        cur.execute("insert into Comfort(Sensor, Value, Date) values(’"+msg.topic+"’,’"+msg.payload+"’,CURRENT_TIMESTAMP);")
    print(msg.topic+" "+str(msg.payload))

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.connect(host="86.119.31.113", port=1883)

# Blocking call that processes network traffic, dispatches callbacks and
# handles reconnecting.
# Other loop*() functions are available that give a threaded interface and a
# manual interface.
client.loop_forever()
```

L Application: Implementation of the Eclipse Paho Javascript client

Listing 28: Paho client inside the application that posts the data to the server

```
var wsbroker = "86.119.31.113";
var wsport = 9001;
var client = new Paho.MQTT.Client(wsbroker, wsport,
                                  "myclientid_" + parseInt(Math.random() * 100, 10));

client.onConnectionLost = function (responseObject) {
    console.log("connection lost: " + responseObject.errorMessage);
};
client.onMessageArrived = function (message) {
    console.log(message.destinationName, ’ -- ’, message.payloadString);
};

var options = {
timeout: 3,

onSuccess: function () {
    console.log("mqtt connected");
    // Connection succeeded; subscribe to our topic, you can add multile lines of these
    //client.subscribe(’/World’, {qos: 1});

    // publish to a topic on connect
    message = new Paho.MQTT.Message("New User connected: " + window.localStorage.getItem(’appUID’));
    message.destinationName = "/Connected";
    client.send(message);
},
onFailure: function (message) {
    console.log("Connection failed: " + message.errorMessage);
}
};
function initClient() {
    client.connect(options);
}

/*function sendJSON() {
    console.log("sendJSON entered")
    jsonString = JSON.stringify(globalData);
    message = new Paho.MQTT.Message(jsonString);
    message.destinationName = "/Data";
    client.send(message);
    window.location.href="question.html";
}*/

function sendActivity() {
    console.log("sendActivity entered")
    activityValue = globalData.activity;
    message = new Paho.MQTT.Message(activityValue);
    message.destinationName = "iphone/" + window.localStorage.getItem(’appUID’) + "/activity";
    client.send(message);
}

function sendNoise() {
    console.log("sendNoise entered")
    noiseValue = globalData.noise;
    message = new Paho.MQTT.Message(noiseValue);
    message.destinationName = "iphone/" + window.localStorage.getItem(’appUID’) + "/noise";
    client.send(message);
}
```

M Application: Save variable in local SQL database

Listing 29: Open and fill local SQL database

```

        // *** start local database ***
            function addGlobalToLocalDB() {
                // *** start sql database ***
                    // create db with approx 10mb of storage
                    var db = window.openDatabase("localDB", "1.0", "Local DB", 10000000);
                    db.transaction(runTransaction, errorDB, successDB);

                        var noiseUdb = globalData.noise;
                        var lightUdb = globalData.lighting;
                        var noiseSdb = globalData.NoiseS;
                        var lightSdb = globalData.LightS;

                        function runTransaction(t){
                            t.executeSql(’CREATE TABLE IF NOT EXISTS comfort (id unique, noiseS, noiseU, lightS, lightU, date)’);
                            t.executeSql("INSERT INTO comfort (noiseU, lightU, noiseS, lightS) VALUES ("+noiseUdb+", "+lightUdb+", "+noiseSdb+", "+lightSdb+")");
                        }
                        function errorDB(err){
                            console.log(’Error creating tables: ’+err);
                        }
                        function successDB(){
                            console.log(’Successfully created tables’);
                            window.location.href="question.html";
                        }
                // *** end sql database ***
            }
        // *** end local database ***
```

N Server: Node.js script that notifies a user with a tag

Listing 30: PushBots Node.js script for User1

```
exports.api = require('./api');

var pushbots = require('pushbots');
var Pushbots = new pushbots.api({
        id:'5609620b1779591b758b4567',
        secret:'70efebad0128903db8f6e593a1bbdb08'
});
Pushbots.setMessage("How are you right now?" , [0] );
Pushbots.customNotificationTitle("comfortIST");

// push to user with certain tag
Pushbots.sendByTags(["User1"]);

//to push to all
Pushbots.push(function(response){
        console.log(response);
});
```