Universidade Federal de Santa Catarina
Centro de Blumenau
Departamento de Engenharia de
Controle e Automação e Computação

Enrique Adami Geske

Applications of IoT in Workspaces : Building of a Smart Desk

Blumenau
2019

Enrique Adami Geske

Applications of IoT in Workspaces : Building of a Smart Desk

Final work submitted in partial fulfillment of the requirements for the degree of BEng. in Automation and Control Engineering of the Universidade Federal de Santa Catarina.
Advisor: Prof. Dr. Daniel Martins Lima

Universidade Federal de Santa Catarina
Centro de Blumenau
Departamento de Engenharia de
Controle e Automação e Computação

Blumenau
2019

Enrique Adami Geske

Applications of IoT in Workspaces : Building of a Smart Desk

Final work submitted in partial fulfillment of the requirements for the degree of BEng. in Automation and Control Engineering of the Universidade Federal de Santa Catarina.

Examination Committee

Prof. Dr. Daniel Martins Lima
Universidade Federal de Santa Catarina
Orientador

Prof. Dr. Mauri Ferrandin
Universidade Federal de Santa Catarina

Prof. Dr. Daniel Alejandro Ponce Saldias
Universidade Federal de Santa Catarina

Blumenau, January 30, 2019

I dedicate this work to my family...
you know who you are.

Acknowledgements

I thank Dennis Stolze for the internship opportunity and all the support given in the development of this project.

I thank Prof. Daniel Martins Lima for accepting the difficult task of guiding me, not just in this occasion, and for doing it so well.

I thank my parents for the love and support given during all my life and extend it to my family, though far away, as I write this. This would not be possible without you.

To all my friends, you are very important to me and the support you give cannot be underestimated.

Finally, I thank Wesley Safadão for being the soundtrack of many intense nights writing this paper.

"To achieve great things, two things are needed;
a plan and not quite enough time."
(Leonard Bernstein)

Resumo

Fraunhofer Gesellschaft é uma organização de pesquisa com institutos espalhados por todo o globo e um deles é o Fraunhofer - Instituto de Engenharia Industrial, em Estugarda, Alemanha. Entre seus campos de pesquisa está Ambientes Cognitivos, do qual o projeto Office 21® faz parte. Este projeto de pesquisa conjunta busca responder as seguintes questões: como iremos trabalhar e viver no futuro? Onde o trabalho intelectual e de escritório irão ser desenvolvidos e como esses ambientes serão projetados para satisfazer suas necessidades? Como existe uma nova demanda por projetar e adaptar espaços de trabalho para melhorar o bem estar e a a performance dos trabalhadores, apoiado por pesquisas acerca de como as variáveis ambientais influenciam em fatores de cucesso, a solução encontrada foi construir uma estação de trabalho inteligente. Nomeada SmartDesk, o dispositivo é uma estação de trabalho totalmente controlável, usando Internet das Coisas como seu principal paradigma tecnológico. É equipada com um conjunto completo de sensores para o usuário poder acompanhar as variáveis que mais influenciam no seu bem estar, como temperatura, umidade e luminosidade e é controlado através de uma interface responsiva que pode ser acessada por smartphone, computador ou outro dispositivo similar. É descrito, nesse trabalho, desde a conceitualização e revisão bibliográfica, que valida este projeto como uma solução viável, até o planejamento e construção do protótipo. Contém também a discussão acerca de se o projeto foi ou não bem sucedido, tanto do ponto de vista do produto, como de uma prova de conceito para aplicações de IoT no espaço de trabalho.

Palavras-Chave: 1. SmartDesk. 2. Estação de Trabalho. 3. Internet das Coisas. 4. Rede de Sensores. 5. Dashboard.

Abstract

Fraunhofer Gesellschaft is a research organization with branches spread all around the globe and one of them is Fraunhofer - Institute for Industrial Engineering , in Stuttgart, Germany. Among its research fields is Cognitive Environments, to which the project Office 21® takes part. This joint-research project aims to answer these questions: how will we work and live in the future? Where will office and knowledge work take place, and how will these environments be designed to fulfill its needs? As there is a new demand for designing and adapting workspaces to improve the well being and performance of workers, backed by research on how the environmental variables influence these success factors, the solution found was building a smart workstation. Named SmartDesk, the device is a fully controllable workstation, using Internet of Things as its main technological paradigm. It is equipped with a set of sensors for the user to keep track of the variables that most influence on their well being, like temperature, humidity and light intensity and is controlled via a responsive interface that can be accessed via smartphone, computer or other similar devices. It is described, in this work, everything from the conceptualization and the bibliographical research that validates this project as a viable solution, to the designing and building of the prototype. It also contains the discussion on whether the project was successful or not, both from the product point of view as well as a proof of concept for applications of IoT in the workspace.

Keywords: 1. SmartDesk. 2. Workstation. 3. Internet of Things. 4. Sensor Network. 5. Dashboard.

List of figures

List of tables

Acronyms

UFSC	*Universidade Federal de Santa Catarina*
HTML	*Hypertext Markup Language*
CSS	*Cascading Style Sheet*
PHP	*Hypertext Preprocessor*
IAO	*institut für Arbeitswirtschaft und Organisation*
SQL	*Stuctured Query Language*
NoSQL	*Not only SQL*
IDE	*Integrated Development Environment*
GPIO	*General Purpose Input/Output*
I2C	*Inter-Integrated Circuit*
I/O	*Input/Output*
SCL	*Serial Clock*
SDA	*Serial Data*
USB	*Universal Serial Bus*
VNC	*Virtual Network Computing*
GND	*Ground*
PWM	*Pulse-Width Modulation*
DMX	*Digital Multiplexer*
LED	*Light-Emitting Diode*
RGB	*Red Green Blue*
DIY	*Do It Yourself*
IoT	*Internet of Things*
RFID	*Radio Frequency Identification*
CAD	*Computer Aided Design*
ZVE	*Zentrum für Virtuelles Engineering*
API	*Application Programming Interface*
GSM	*Groupe Special Mobile*
NFC	*Near Field Communication*
LTE	*Long Term Evolution*
AI	*Artificial Intelligence*
HTTP	*Hypertet Transfer Protocol*
SSID	*Service Set Identifier*
DVI	*Digital Visual Interface*
HDMI	*High-Definition Multimedia Interface*

Table of contents

1 Introduction

Times are changing and so are the ways people interact with the world around them. This fact reflects in every aspect of the human life, including work. Today, the knowledge and office work are becoming more and more important and necessary for every area. The worker of this kind is someone who has to think to do their job and their main asset is the information that they possess, using it for problem solving and spent good amounts of their time researching. That includes engineers, scientists, physicians, lawyers, academics and so on. Because of that, the technology that guides the interaction between the worker and his work tool needs to provide an easier and more comfortable experience for him. Not just that, but because of the tendency to shared workspaces and non-fixed workstations rises, the local environment has to adapt itself to the needs and preferences of every user.

Since the market keeps depending even more on the knowledge work, it's a responsibility of the companies to adapt themselves in order to become more attractive to the needs of these professionals. The office work has a huge parcel of contribution to the economy. Most of what is created nowadays relies on this environment, such as new products and services, organizational and logistic projects, marketing campaigns and market research. Specifically for this type of work, the demand is rising for more competence and independence of workers, making fixed workplace and work hours impracticable.

This scenario of changes in work and how it is developed induces the reflection about what is the real importance of the office as workspace and consequently, how those spaces have to be designed. That is the role of the project *Office 21*®, from *Fraunhofer - Institut für Arbeitwirtschaft und Organisation*.

One of the research fields at Fraunhofer IAO is Cognitive Environments, in which the project Office 21 takes part. It's main goal, in the current stage, is to create and apply new technologies into designing new products and workspaces with the goal of increasing productivity, comfort and other success factors. All of this is backed by years of own research that shows the importance of a workspace suitable for the type of work developed there. Because of that, in partnership with numerous companies, it proposes the creation of more adaptable workspaces.

Because of that, the project presented in this document has the goal of creating a solution for adaptable workspaces using the knowledge obtained during the Control and Automation Engineering course and also contemporary technological paradigms, like Internet of Things. In that sense, the project will not just be a product, but also a proof of concept for applications of IoT in the workspace.

The solution for that is creating a modern and powerful workstation, with features that enable it to be called "smart". The project, then, consists on building a fully controllable

desk, as an application of the Internet of Things, uniting the concepts of data acquisition, networks, databases, control and mechanics. It will be equipped with a set of sensors for the user to keep track of the variables that most influence on their well being, like temperature, humidity and light [9][10][2][3]. It will also allow the user to control various aspects of it remotely. The final product was conveniently named Smart Desk.

Regarding the structure, the device is a standing desk, which is able to change its height in order for the user to work while sitting or standing. It contains all the necessary parts necessary for any type of office work: screens, a lamp, and cables for connecting the laptop. A sideboard, which is the "wall" in which the screens and the lamp are attached, is also part of the package called SmartDesk and is also controllable as well as lockers and drawers for storing things.

There are sensors attached in order to measure both the quality of the environment, for example temperature, pressure and humidity, as important variables to be controlled, such as desk height and luminosity. All of this is controlled through a responsive interface that can be accessed via smartphone or computer and all of it is connected via a private network for the workstation. The sensor data is stored in a database for displaying purposes, analysis and usage in possible future upgrades and other implementations.

1.1 General Goal

The goal of this project is to create a better way for the worker to interact with his work environment in order to improve some success factors such as well being and performance. The prototype must be a fully functional and controllable workstation, that can provide the user real time information.

1.2 Specific Goals

- Instrumentation of the desk. Putting sensors and configuring them to get the ambient data, such as the temperature, humidity, pressure and light intensity.
- Store data from the work environment. The data is useful when shown to the user, that can track in real time and for analysis, in some future application.
- Create a device network of actuators in the workstation. All the controllable variables, such as desk and sideboard height, light intensity and the security will be available for controlling via the network.
- Design and program a user interface for controlling these devices. A dashboard in which every readable and controllable variable are disposed in a user friendly and intuitive manner.

1.3 Document Structrure

The next chapters describe everything from the planning to the construction and analysis of the results. The following chapter introduces Fraunhofer and Office 21, the place in which this work took place. Chapter 3 contains the bibliographical research and the conceptualization of the idea. Chapter 4 details the design of every part of the prototype and justify the choices for each one of them. Chapter 5 describes the building process of what was designed in the details and, finally, Chapter 6 discusses the results and whether the project was successful or not.

2 Fraunhofer IAO

Fraunhofer Gesellschaft is a research organization with branches spread all around the globe, including 72 in Germany, it's home and main country. It was founded in 1949, the same year as the federal republic of Germany, in a small office with 3 employees and it's name is a tribute to *Joseph von Fraunhofer* (1787-1826), a Munich researcher and inventor, who was one of the pioneers in the optics field.

It has a budget of over 2.3 billion euros and more than 25 thousand employees, making it the biggest organization for applied research in Europe. Its main fields of research are *Security and Protection, Mobility and Transport, Production and Supply of Services, Communication and Knowledge, Energy and* and *Health and Environment.* The partners and customers of the organization come from the public administration, industry and the service sector.

One of it's institutes, located in Stuttgart, is *IAO*, which stands for *Fraunhofer-Institut für Arbeitswirtschaft und Organisation*, or, in English, *Institute for Industrial Engineering.* The main focus of this institute is the interaction between humans and technology, approaching themes like virtual reality, adaptable cities, artificial intelligence, future of mobility and specially, cognitive environments. It is also a main part of the *Mass Personalization* center, which is a joint initiative between the Fraunhofer institutes in the city and the University of Stuttgart. One of it's buildings, *ZVE (Zentrum für Virtuelles Engineering)*, can be seen in Figure 1.

Figure 1 – ZVE (Photo: Christian Richter © Fraunhofer IAO)

2.1 Research and the project Office 21

The last one mentioned, *Cognitive Environments*, is the area that researches in what way people interact with the things around them in different situations and places, how it is changing, and how to improve these interactions, with great focus on themes like IOT, Big Data, AI and Automation, and how they can help accomplish important results in improving comfort, productivity and, nevertheless, financial results. It also searches how the environmental variables can interfere differently on each type of activity and how to take advantage of it. One of it's main projects approaches the theme *Future Workspaces*, which is the case of *Office 21*.

The nature of work has been changing in the past decades. These changes come as result of the innovation in the area of information and communication technologies (ICTs) and they appear in the pattern of work and also in the environment, thus being a responsibility of the employer to provide an appropriated platform for the flexibility it presents. The technology also expanded the time and space of work, making possible for people to work across countries, in different time zones, using the internet for example. Supporting all of this is the challenge for the companies in this century [10]. The premise is that in order to improve worker performance is important to have a suitable environment.

The joint-research project *Office 21* aims to answer these questions: how will we work and live in the future? Where will office and knowledge work take place, and how will these environments be designed and equipped to sustain performance, motivation and innovation? It mainly serves the purpose of examining how these types of work will change and determining what has to be done to improve the environment in which they occur. Contrary to what was thought, the changes on the dynamics of the work did not exclude or diminish the important role of the office as the main space, not being then, just a place for service provision, but also a place where workers can be inspired, comfortable and supported on these tasks. When designing environments, it is important to take into account how the physical variables interfere, improve or worsen, the quality of the work that is being made and the performance of the worker.

Curious enough, the first tests regarding the relation between environmental variables and performance were conducted in 1920 and were known as The Hawthorne Studies and researchers realized that any modification on the work environment, more or less temperature, humidity or light would make the workers more productive, thus actually attributing these improvements to the attention that they were given from the researchers [10]. As lighting was one of the variables approached, this now well-documented "experimenter effect", in which the presence of the researches had more influence than the environment itself, led to a significant decrease in researches about direct relation between lighting and performance at work. The researches, then, started to focus on what is the influence of lighting in the workers' performance through its effects in their well-being.

Taris and Schaufeli [11] say that work performance involves various factors surrounding the work being made. Performance can be classified within its relation to the job. If directly related to their job requirements, it is in-role and if it adds to the organization's goals, but not directly related to the job description, it is extra-role. It can be classified within the nature of the task, distinguishing what is done and how it is done from how much it achieved the goal.

Another interesting factor is that these behaviours are often related, thus being possible to infer that an individual is productive or counter-productive in both in-role and extra-role tasks. The effects of that is that the collective performance can be taken into account, and it makes sense, due to the fact that it is easier to identify productive groups doing larger tasks than individuals doing parts of it.

The relation between worker well being and performance is, from the point of view of Sonnentag and Frese [12] divided in three different perspectives. The individual perspective takes into account the abilities, preferences and personality of the worker, the second one, the situational perspective, examines the performance according to the environmental factors and how they improve it or not. The last one is the performance regulation perspective, which analyzes the sole process of work and how it can be modified to improve the performance[11]. Regarding lighting, the second perspective is the one that matters to this discussion, and relies on the individual approach to well-being.

The individual-oriented solutions aim for the well-being of the employee, reducing stress and creating a space for him to develop his skills. Providing the worker with a proper workspace that allows him to work in a comfortable manner has to be a priority in the organizational strategy if the goal is to increase productivity, as many studies show that some of the main causes of stress-related issues and lack of productivity are poorly designed workload and poorly designed environments.[10]

Since the result of previous researches show the importance of spatial-technological infrastructures and office environments on individual and organizational success indicators, the more the technology evolves, more opportunities appear and more forms of improvement become possible. Working in partnership with companies in order to design new products and environments with constant feedback, important points of improvement are tackled by the research. They are: communication, concentration, well-being, productivity, motivation, creativity and innovative ability of and in companies[13]. The challenge, then, is to discover specifically in what way the ideal condition can be created for the development of knowledge work and get the best possible satisfaction from the worker.

2.1.1 Office Analytics

The previously cited book Office Analytics [13], recently published by Fraunhofer IAO, that contains some advances so far in Office 21, shows the result of empirical data collected

over 2 years of research and presents some key points to be considered on approaching work types and environments, and the success factors that depend on them, as cited in the upper section. The data presented on this book served as the main theoretical basis for this project.

One of the important points brought up was the role of self-determination on motivation. The power of structuring your work make workers not just more motivated, but able to deal with stress in a much better way and builds a better work-life balance. That can be supported by the use of different workplaces (desks, or geographical locations), which is also a great driver for new ideas. Technology that did not exist a few years ago is one of the main responsibles for this flexibility and the so-called "job control" possiblity [10]. Different places have different characteristics such as offices have more than one role, and that has to be taken into account when designing workspaces. That can be seen through the fact that 48% of workers are not totally satisfied with their offices, which attests the opportunity for improvement. The goal in this case is to make the worker feel that the space suits him as much as possible in every one of them. [13]

2.1.2 Conclusion

This project, then, presents a contemporaneous necessity of designing and adapting workspaces. At first, the data is used as base to suggest a new idea for improving the workspace, what would be better and how to accommodate different kinds of work and workers. Then, the idea is formulated in the form of a service or product (in this case, product) and then, the concept, design, building and results. All the development of the project is described and discussed in the next chapters of this document. Further research about specific topics approached in the solution are also mentioned alongside the formulation of it, on the next chapters.

3 SmartDesk

The previous chapter showed the structure of the company where this work was developed. It also gave an idea of the project it takes part on, its premise and goal, as well as what is expected as result in this paper. The sole basis of the research was also cited to introduce the theoretical foundation upon which are justified the necessity and validity of the purposed solution.

This chapter presents the solution for the problem of designing future workspaces. It comprises the bibliographical research necessary to support the choices and the brief explanation of the solutions. The next section will detail some important concepts used for formulating the whole idea.

3.1 Concepts

In order to rethink the work environments of the future, it is necessary to be aware of the new technologies and technological paradigms. This is the era of Internet and its different applications, and, right now, it is being seen as not just a way of connecting people to people and accessing web content, but also as a way of connecting people with things and things with things [14].

Data right now is also of great value. It is the time in which companies are working with behaviour-oriented services more than ever. With data acquisition and analysis, it is possible to predict patterns, behaviours and find more suitable solutions for everyday problems and also offer more personalized content and products, all of this using the power of internet [15].

Hence, is of great importance the understanding of the concepts related with the statements above, Big Data and Internet of Things, which are the 21st century paradigms guiding the market and adding more value to products and services. To understand in which way that can help creating new technology for offices, its necessary, beforehand, to know the ideas behind them.

The concepts above mentioned are described and discussed in the subsections below. Other topics and concepts are also explained over the course of this project in the appropriated area. These are specific to the topic and are part of the bibliographical research contained in this chapter alongside the explanation of the solution.

3.1.1 Internet of Things (IoT)

Internet of things is one of the terms that have brought more attention to the tech industry lately. Even though it has no exact definition, it is a paradigm that involves

using the internet to create a network of machines and devices that can communicate to each other. This concept is becoming more and more important and is now seen as one of the main areas of future technology [14].

At low level, any device that has an on-off switch can be connected to a network and be controlled and managed remotely. Although it does not cover even the ground floor of the concept, it shows that the extent of the applicability is huge. Mainly what happens is, when devices can communicate with one another, it adds functionality and consequently, value. It has come to a time where, everything that can be connected, will be, and that does not include just computers and cellphones, but washing machines, lamps and even cars. All of that using information.

The analysis firm Gartner [16] previews that by 2020, 26 billion devices will be connected, and some believe that the number is actually much bigger, and so are the possibilities. But different from some really basic applications, as turning the lights on and off remotely, data is a major asset in others. An interesting (hypothetical) use case would be if, for example, someone wakes up at 6 am, the alarm on the cellphone rings and it also notifies the coffee machine to brew coffee for him, or his car, knowing via calendar where his meeting is, shows the best route to the place. Those are some examples that show the real power of intercommunication between devices and data usage, which explain how valuable the concept is nowadays.

An interesting factor is that various different communication pattern and technologies tend to be invented and coexist more and more, due to various applications. One important examples is the smartphone. A smartphone nowadays supports various types of wireless connections itself, for example, *Wifi*, *GSM*, *NFC*, *Bluetooh*, *LTE*, among others, as shown in Figure 2. This variety increases the capability of devices to be connected, examples such NFC allow people to connect, in close distance, to objects that don't even run on batteries or such, like external buttons, or use it as a port to pair devices via Bluetooth just by touching them, showing the means of interaction between these technologies [1].

Besides different means of communication, hardware is a big part of what is causing this exponential growth in *IoT*. Nowadays, everyone with a little knowledge and effort can build some kind of controller for other devices, something that sometime ago would not be possible with that much ease. This is due to the increasingly wide range of hardware, mainly prototyping boards, on the market today. Boards like *Arduino*, *Raspberry Pi*, *NodeMCU* and so on, are the basis of many of the advances happening today. Their programmability (some even are a fully functional computer that can run *Linux*) and large support community is pushing the *DIY (Do it Yourself)* culture in to the heads of people from a wide range of age and experience, making it possible to create projects ranging from a remote light switch to a fulling functional smart home.

A lot of applications require data, going from little to big amounts. This data can

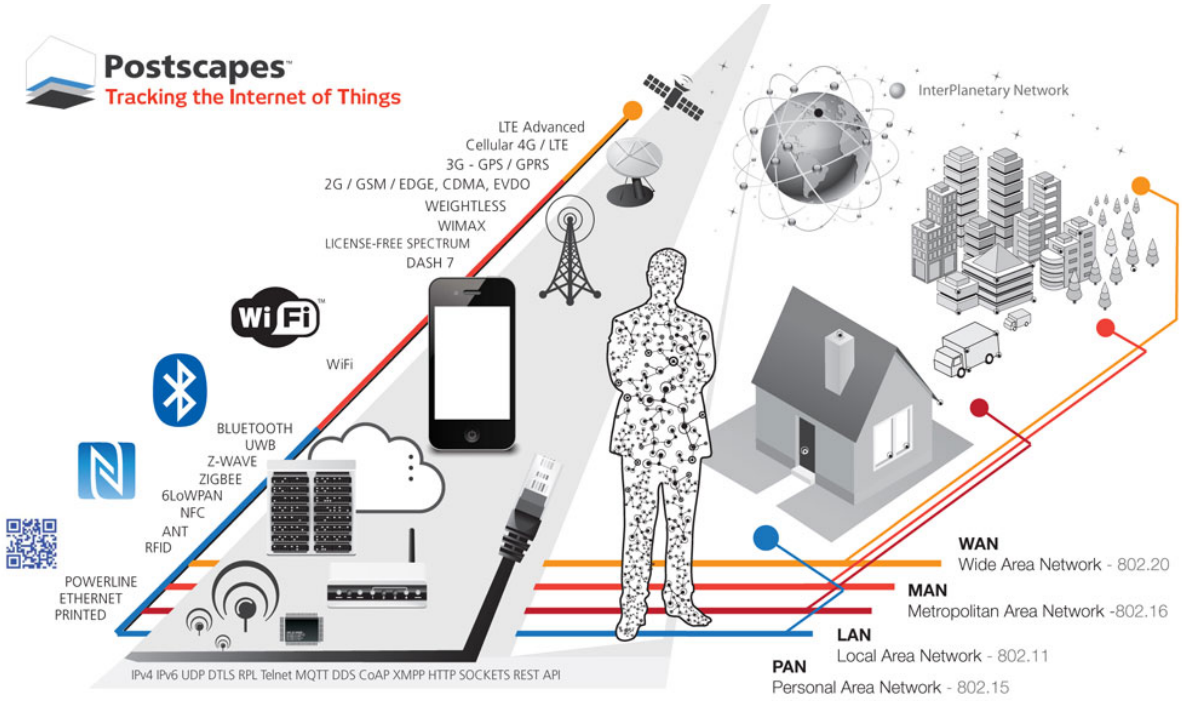


Figure 2 – Communication on IoT [1].

be personally set up, or acquired. Depending on the nature of this information, there are different ways of acquiring it. If is environmental or measurable data, it's common the use of sensors of different kinds, but if its personal data, besides the amount being much larger, it is taken from your computer, smartphone and other personal devices and represent a much bigger amount of data that is not necessarily measurable, which require a different kind of analysis. This last one is called *Big Data*, which is going to be explained next.

3.1.2 Big Data

Big Data got its name primarily for representing a huge amount of data that is characterized by exceeding the processing capability of regular databases not just for the amount of it, but for it format and speed. Even though it has no specific meaning, it is usually referred to the gathering and processing of huge amounts of different types of data [15].

The key factor here is: non-structured data. Around eighty percent of the data generated is not structured, that means it cannot be "interpreted" directly and that includes audio, images, texts messages and tweets, for example. Besides, its amount doubles every year. The classical tools for data storage and management just cannot handle this, and that's why, for a long time, these information have not been taken advantage of. Now, various tools are being used and developed for these applications, and that's the phenomenon of *Big Data*. These tools include NoSQL databases, which have characteristics like being schema-free, easy replication support, simple API (Application Programming

Interface), a huge amount of data and more. They rely on storing these non-structured dat types in collections and often use distributed processing [17].

Another factor that hugely contributed for the growth of big data is the range of devices being used nowadays, a thing that 10 years ago was not even close to what exists now. Devices such as smartphones, tablets, computers, wearables and more, are the main generators of these non-structured content that constitute the challenge.

The detail is that when correctly processed and analyzed, this data is of huge utility. These information can show tendencies and preferences, for example, and help on improving advertising or on offering more personalized services. That is a fraction of the applications, but these can give an idea of the Big Data power. Even though this project does not make use of big data, it offers some aspects of it in addition to the *IoT* principle, as will be explained further in this document.

3.2 Improving the Workspace

As previously mentioned, this project presents a contemporaneous necessity of designing and adapting workspaces. This means implementing technologies in order to create a more intelligent and suitable office for knowledge work. The paradigms explained in the previous section should possibly be part of the solution. As shown in the book *Office Analytics*, a great part of the workers are not satisfied with the place they work, evidencing the need for improvement. But what does that mean?

It not a major consense what represents an improvement, due to different types of workers and their personal needs and preferences. So the solution cant be a fixed one, but the possibility of interacting and modifying your work environment easily. The choice, then, based on that, is to attack the problem of the workspace adaptability. So to speak, it consists on making the workstation fit the worker more by being controllable and personalizable, and as a consequence, finding new ways of interacting with the work environment.

As an application of the concepts above explained, the project consists on creating a workstation that can be controlled through an *IoT* point of view and has useful features to improve the comfort of the worker and practicality in the interaction with the station. It was conveniently named *SmartDesk* and in Section 3.3 it is explained with more details.

3.3 The Solution

The so-called *SmartDesk* consists of a complete workstation, with a desk, which can change its height, screens, a lamp, lockers and a drawer, various outlets and a sideboard, which supports the screens, light and so on. It's also equipped with a full set of sensors and actuators, all functioning in the station's own wireless network, making it fully adaptable

and controllable. It also features a database which stores all the data from the sensors in order to provide a better view of the environmental variables.

As an application of *IoT*, all the functions of the workstation can be controlled remotely, using it's network. That means that the information acquired from the sensors can be monitored by distance and also the desk can be operated in the same way, which includes opening the drawer and lockers, controlling the lights and the screens and changing the height of the table and the sideboard.

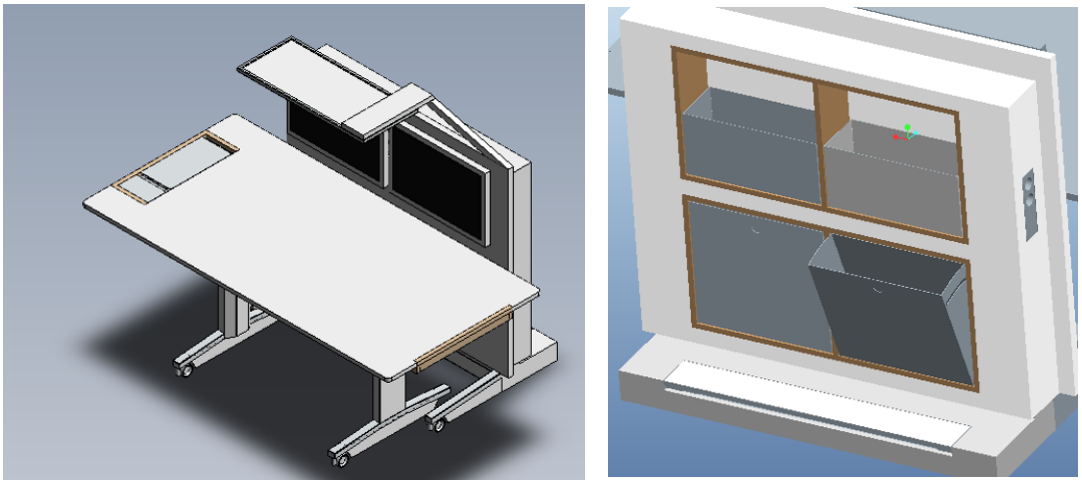
The *SmartDesk* takes advantage of a pre-existing structure of desk and sidebar not in use in the office. the whole project is built on top of it, thus making it useful again and serving as a proof of concept for a smart workstation. All of the features of this device as well as the description of the structure are listed and explained briefly on the next subsections.

3.3.1 The Structure

This project is possible because it is built upon an existing structure in the *ZVE* office. This structure was constructed as a modular workstation, composed by the sideboard and the desk, as shown in the *CAD* model on Figure 3a and Figure 3b. The first one, as mentioned, has the two screens attached to it and also the lamp, eliminating the need to an external lamp and saving room on the desk that would be used for the screens. On the back of this sidebar there are two tilting lockers and other two shelves. The base of the sideboard is built on motorized standing desk columns, which means that it can rise and go down.

The desk is also built on top of standing columns, thus being characterized as a standing desk. It contains a drawer on the right side of it, that can be controlled electronically, as the lockers on the sideboard. The role of the drawer is mainly to store the laptop, that via USB is connected to the screens and also via cable to the network. On the left side, it has some outlets, including power, *Ethernet* and *USB*, as well as a place to put objects with a magnetic cover. The desk can also turn in to a drawing board by flipping the top up.

Originally, the locking mechanism on the lockers and drawer was designed to work with *RFID - Radio Frequency Identification*, and that was the only thing left on the table that was properly working. Although it was working, none of the software code for these were available, so everything had to be coded and rewired from scratch. Basically, of the whole project, only the structure was already built, so, one of the objectives was to make it functional again. The next subsections discriminate everything that composes the final solution and the Chapter 4 approaches the project itself.



(a) Desk and Sidebar. (b) Back of Sidebar.

Figure 3 – *CAD* model of the existing structure.

3.3.2 Motors and height control

The contemporary worry about workers health is a true thing and is very present in the workspace engineering since the past decade. The most common form it takes is the working while standing paradigm. It is well documented that long periods of sitting as well as low amounts of exercise can lead to health conditions and thus also affect your work[18]. To improve this situation, standing desks in the workplace are nowadays the most feasible, modifiable, and acceptable environmental strategy to this end. Standing desks allow you to work while standing or sitting, through a motorized setup that changes the height of it.

Most researches were conducted on the effects of standing desks on students. The data found on them is that while standing, as expected, they burn more calories, but nothing was conclusive regarding the improvement on the academic performance. Nonetheless, the health effects are usually a long-term issue, not measurable in short periods of time. The plus side is that the presence of the desks did represent an improvement by reducing on the amount of time spent being sedentary [18].

One of the latest important publication on this subject, from Buckley and Hedge [19], consists on a set of guidelines for employers to promote the avoidance of sedentary behavior in the workplace. It shows, based on scientific evidence, that there is actually a specific amount of accumulated time that would be ideal for standing and doing light exercise (walking) during work. This amount is from two to four hours, and the recommendation is to start at two hours and progressing until reaching four hours. The research shows that the ideal is a combination of standing and seating during work, and that is achieved by the capability of altering the height of the workstation.

Both the table and the sideboard are built upon standing columns, which means that they are motorized aand capable of changing their height. Justified by the research

mentioned above, it is important for every workstation to have this capability and that is mostly why one of the implementations of the *SmartDesk* project is height control. As the height is an important quantitative variable, it must also be read, for that a sensor comes in handy, so the person controlling can know at what height the table is. Since the sideboard moves along and has no defined point of interest to measure the distance to, the sensor is only necessary on the desk.

Another feature is the synchronization between the desk and the sideboard, so, when the they move together. This is the only way it can provide comfort while changing work positions, as the sidebar carries the screens and it would not be comfortable to change the height of the two parts separately. As to how the height control works, there are modules developed exclusively to work with standing desks, so they are the most useful and practical for this task, meaning that the job here is just sending the signal to go up or down. Further technical specificities will be presented on the Chapter 4.

3.3.3 Lighting

One of the components of every workstation is a lamp. It is necessary for almost every kind of work, and on different levels. The importance of lighting in the workplace is undeniable, but it's effects on the mood and performance are an important subject. Though natural light is very important and one of the best sources of it, as mentioned before, the nature of modern work implies that artificial light is most of the times necessary, at least partially, due to office work, changing demographics, night shifts and so on [10].

As mentioned before, one of the perspectives regarding the relation between worker well-being and performance is the situational perspective. It studies how the environmental aspects affect the well-being and thus, the performance. As lighting is one of the main environmental aspects, it is useful to know what the research says about its effects on the worker.

At a basic level, it is well documented that jobs that require visual effort, when done in poor lighting, like using the computer or writing, can cause fatigue and in long term, do serious damage to the vision of the worker. It has a straight-forward relation with stress as it leads to the inability of doing the job itself. The least obvious data is that changes in the source of light (direct or indirect) have no conclusive effect on the well-being of the person and neither its intensity, being dependent, in fact, of the preference of every worker and also dependent on other factors, like age for example. This data is shown in the *Fraunhofer IAO* study *Lighting quality perceived in offices* [9][10].

One important aspect regarding the satisfaction and well-being, besides the light intensity and as pointed in different studies is the light temperature color. It was found that different demographics (mainly gender and age) usually have different responses to lighting color settings, and also that different color temperatures have different effects

on specific subjective measures, like alertness, fatigue, concentration and so on. It was also found that variable light settings had the effect of avoiding fatigue and improving performance, at least when tested in a controlled environment [10].

In other study, Küller [2] found out that the more colorful the environment is, the better the mood of the workers is, throughout the year (tested in different seasons), as seen in the Figure 4. An interesting factor in shown in another research is that the ability to control the lights fully improves both the worker's well-being and also his satisfaction with the overall situation of the lighting and that, depending on the setup of the office, being it a open space, small individual office or such, the color preferences change. One thing that is predominant is that the overall satisfaction with *LED* lights rather then other types is bigger. All of this point to the importance of being able to control the light, which is in accordance to the role of self determination to well-being.[9][10]

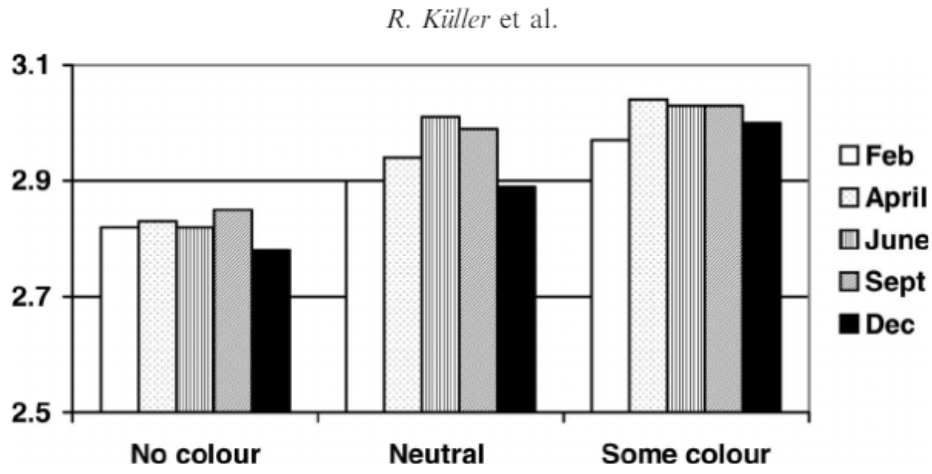


Figure 4 – Mood in relation to colorfulness throughout the year [2].

This research led to the choice of using, in the workstation, an *RGB LED* lamp. It fills all the requirements to improve the well-being of the worker, besides using less power than other types of lamps. The *RGB LED* means that it can change its color to any color in the *RGB* (Red, Green and Blue) scale, because it is composed of multiple red, green an blue *LED*s and it can display up to 16 million colors. The color will be chosen by the user, thus fitting his preference at the moment and also giving the power to compensate the possible lack of colorfulness in the bigger environment, which can significantly change the person's mood as seen in Figure 4.

The lamp will be controlled via a color picker and also an virtual "dimmer" to change the intensity of the light. Another important feature will be the luminosity sensor, which means that the user will be able to track the light intensity in the workstation in order to adjust it to his preferences, because, due to the fact that in different times of the day, there is need for more or less light, the worker will be able to regulate the desired amount of light in a broader sense.

3.3.4 Ambient sensors

For this part, the goal is, in principle, to read and store data from the workspace environment. Those consist in temperature, humidity and pressure. Mainly, those serve only for reading because, in principle, there is no way of controlling those variables workstation to workstation, and in the case of pressure, there is no way of controlling it at all in an office environment. When the data is read, it is then stored in the database and can be visualized by the user at any time.

Although most researches have some level of uncertainty, it is clear, also based on what was previously presented, that the environmental variables have an influence on the productivity of the worker in a high level. Some studies try to calculate the ideal temperature related to the productivity, and an article from Seppanen and Fisk [3] interpolates their data and show that, for example, even in different tasks, the performance increases with temperature up to $21-22\,°C$ and decreases from $23-24\,°C$. As stated in the study itself, the data is non-conclusive, but it shows an undeniable relation between these variables, as shown in Figure 5.

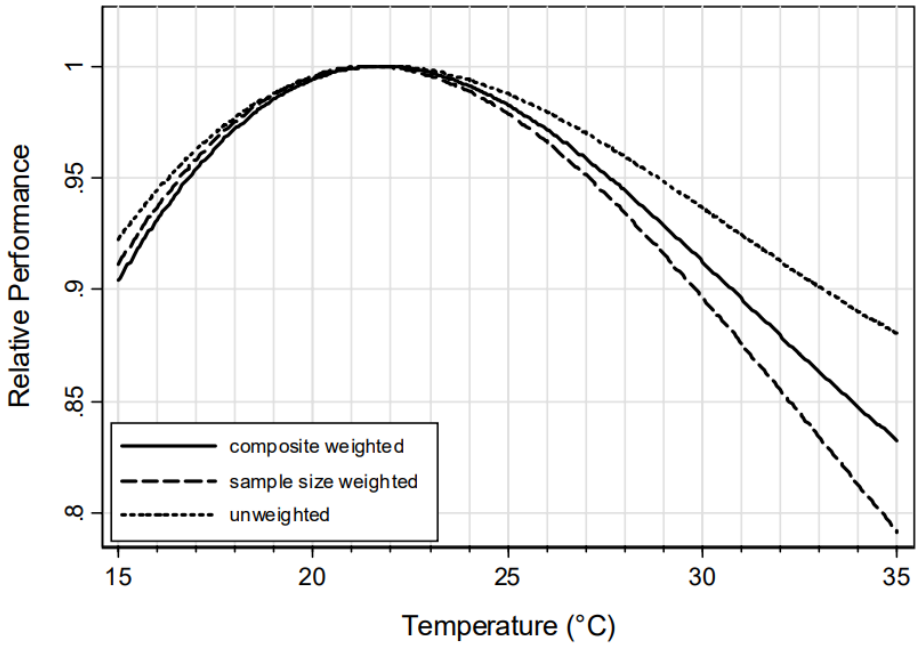


Figure 5 – Normalized performance vs. Temperature, with max. performance set to 1 [3].

The presented research support the need to measure the physical variables for maintaining a pleasant work environment. The main goal is to show the information to the user and, besides informing the worker about the quality of the environment, the data is stored, so it can be used in further analysis. The analysis of the data itself is not approached in this document, but the importance and validity of storing it was discussed in the last chapter.

3.3.5 Drawers and Lockers

As explained, the structure of the *SmartDesk* contains two tilting lockers on the back of the sideboard, which have the sole utility of storing things safely and on the desk there is a drawer that serves the purpose of putting the laptop inside and connecting it to the screens and the network. The lockers and the drawer are also electronic. The locking system is magnetic, which means that an electronic magnet secures it in place, making it controllable through a simple switch, that can be activated in many different ways.

In the original setup, the locks worked with *RFID*, which is a communication protocol and stands for Radio Frequency Identification. This protocol uses a card or tag with an identification number that can work as a "pass". In this project they will be controlled remotely through a web interface, just as the table and the lights, so the worker will be able to unlock all the three compartments by distance. The *RFID* readers, though existent in the structure, had to be coded from scratch, so its application is not trivial depending on the choice for controller.

3.3.6 Database

To store all the data that was obtained, a database is needed. A database is a collection of information that is organized in a way that it can be accessed, managed and updated. There are different types of database and the most used is relational. The most common interface for this type of database is *SQL*, which means *Structured Query Language*. It basically stores the information in tables and the information can be retrieved via query statements.

The main difference between relational databases and *NoSQL* databases, often known as "NotOnlySQL", is that in the *SQL* (relational), the data storage system is composed by tables, which contain certain amount of data in every cell. As usual, every table has a common structure for every row, being able to store data only in that structure, while *NoSQL* databases do not have this restrictions. There are different types of *NoSQL* databases and some use the concept of collections. These collections can store documents that do not have need to have the same structure, like *XML* and *JSON* files, which made easier some more modern applications, as shown in the Figure 6 [17].

Storing time-base information is an easy task in relational databases. Having the values attached to the time of post and just adding a line with it. Since the readings from the sensors are just numerical values and have a exact timestamp, the database does not have to support odd-structured data. The SQL database, then, was the choice for this project. The database can be local or in a external server, but for this application, storing it locally is better. The reason is because the only devices that need to have access to the data are the users in the local network, which will be further explained.

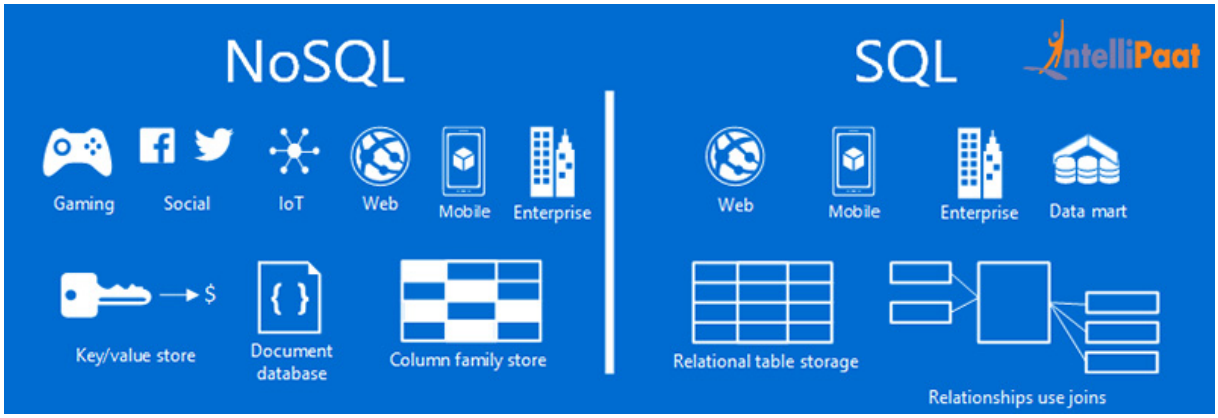


Figure 6 – Applications of SQL vs. NoSQL databases [4].

3.3.7 Network and interface

All the features mentioned above, by themselves, have no real utility if there is no way of displaying the data and controlling the desk. For that reason, all the controllers of the workstation are connected in their own network and so is the database. The devices have two possible roles in the network, which are client and server. Basically, the client access the server to send or retrieve information. As there are different devices for different functions, like, reading the sensors, controlling the desk motor and storing the database, they assume different roles in the whole network.

The main point is creating an accessible interface in which the user can both have the information from the sensors and control the workstation. Since it is all distributed trough the network, the only requirement would be the device having access to it. The best way to do it is with a responsive website, which can be accessed via computer, tablet or smartphone, being adaptable to the screen size and features, by resizing and reorganizing the content. The final product, then, is a responsive dashboard, which can display the content in the most pleasant way, as will be explained in the Section 4.5.

The device hosting the website would act as a server and so would the device in which the database is kept, making it easier for these both being the same device. The choice for the device will be described in the specific session, as well as the role of every node on the network. As consequence, the client will be the device used to interface with the desk, in which the website is accessed from.

3.3.8 Other implementations

By following the description above, the prototype can be built and be a powerful workstation by itself. The interesting thing is that it also creates a basis to easily implement other possible features. In other words, the existence of a sensor network, a controllable desk, a database and an interface opens the door for other functionalities. These functionalities range from a simple user database to a complete recognition system.

As of now, the project consists just on creating this platform and building the smart workstation prototype, as well as acknowledging that parts of the project can also be implemented separately, like a sensor box or a smart lighting with complete success. The SmartDesk would then not only serve as a more useful and suitable workstation, but as a platform for future developments on the *IoT* field.

3.4 Conclusion

This chapter consisted in describing the concept, part by part, of what is going to be the SmartDesk. It also contains the bibliographical research necessary to validate every aspect of the project, thus proving he need of what was chosen. The next chapter takes these concepts and turn it into a proper project, foreseeing how everything is going to be built.

4 Project

This chapter consists in detailing the project itself. It describes everything that is going to be done and what method it is going to use, what equipment, the function of them and why were they chosen. Section 4.1 shows an overview of every important part of both the table and the sideboard setup, and the following ones get into details of each one of them.

4.1 Overview

This section is in an overview of both the desk and the sideboard. It gives a more concrete view of the organization of the elements, of every important device in the workstation, not regarding the wiring but focusing on the "data" connections. It is divided in two subsections, one for the Desk and one for the Sideboard.

4.1.1 Desk

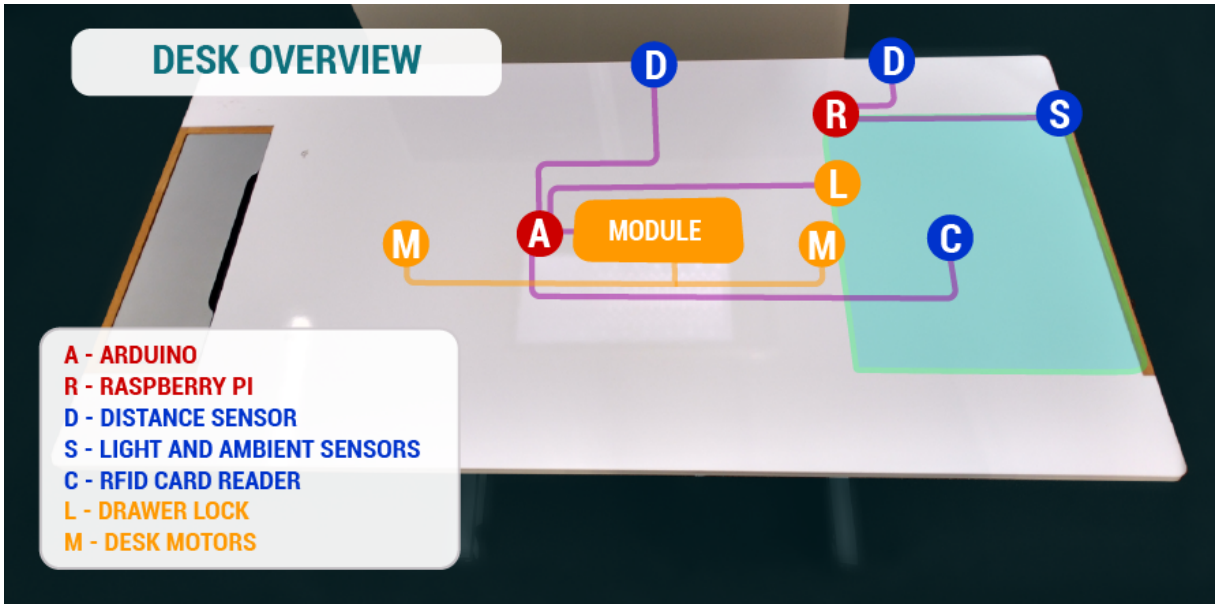


Figure 7 – Desk scheme.

In Figure 7, it is shown the desk and all the important devices there. The explanation for the figure can be found subsequently. First of all, the color separation consists in:

- Blue: input. All the sensors, independent on what device is using them and the RFID card reader all send information to a controller board.

- Orange: output. Just for the outputs in the desk itself, not the information transmitted. The drawer lock and the motors, alongside the module. The yellow line states that the module and the motor are the same output, because they just follow the commands from the Arduino.
- Red: controllers. The Arduino and the Raspberry Pi are the controllers that are going to be used in the desk setup and are going to be explained further. As shown, they have no connection between themselves. Another connotation of the red is that the devices are using wireless communication.
- Pink: to show what devices are connected to which one, since there are two different controllers with different functions on the desk.

The sections below extend on the information presented above, giving a complete description of every device used, such as the Arduino, Raspberry Pi and the sensors. The reasons for their choice is also discussed in the following sections.

4.1.2 Sideboard

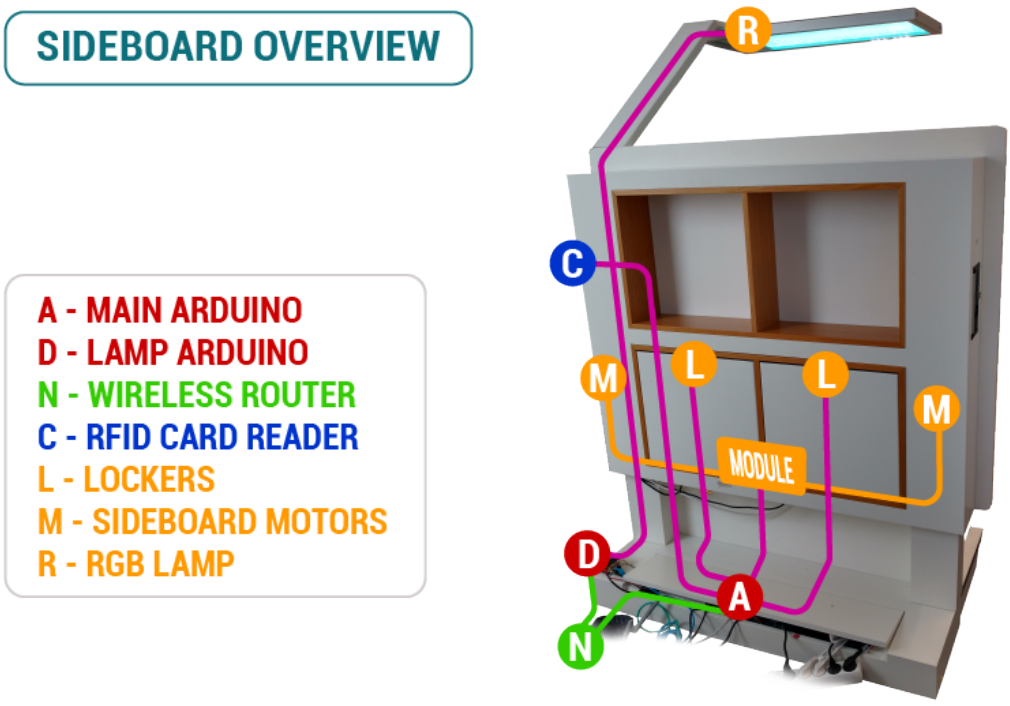


Figure 8 – Sideboard scheme.

Just like the the last subsection,there are explanations that need to be given regarding Figure 8. The color scheme is somewhat similar, but there are slight differences:

- Blue: input. The only input in the sideboard is the RFID card reader, and it also has the security purpose. In this case, the reader opens both lockers.
- Orange: output. The simpler outputs are the lockers, that just open and the motors, that go up and down. In the sideboard there is the light, that changes color, which means there are 16 Million possibilities of output, but is still and output.
- Red: controllers. Two different Arduinos, one for the main controller and one for the lamp. Both are connected to the network.
- Pink: also to show what devices are connected to which one, since there are two different controllers with different functions on the sideboard. As mentioned, one of the Arduinos take care of just the light.
- Green: network. Shows that there is a router and that both the Arduinos are connected to the network, but this time, via ethernet cable.

More details are discussed and referenced in the following sections.

4.2 Motors and Height Control

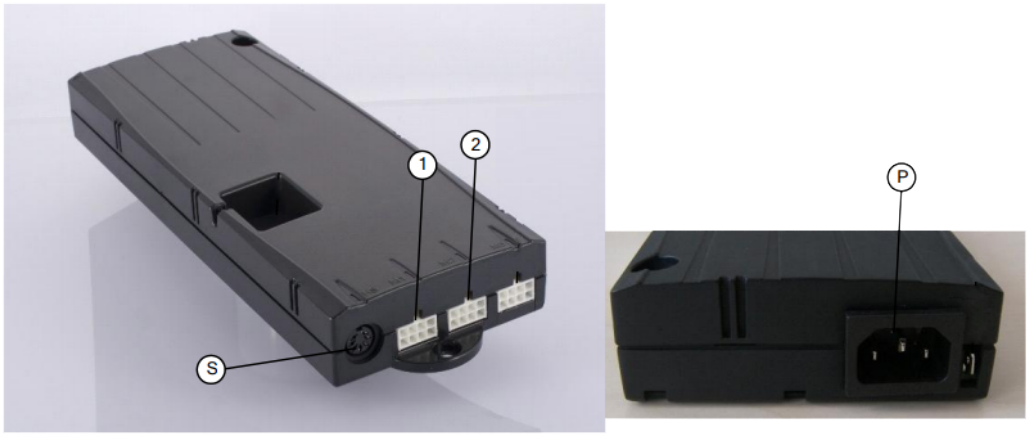
As mentioned before, for a standing desk to be able to work, there is a need for an activation circuit for the motors. As common as they are becoming right now, there are plenty of solutions for that in the market nowadays. The main one are the *Control Modules*, or *Control Boxes*. They are a complete device for attaching the motors, the power supply and the switch, doing the work of synchronizing the activation of the columns and, in some versions, showing the height and saving presets. They usually have a size that can fit every office desk, for example, the module from the company *LOGICDATA*, as shown in Figure 9a, similar to one used in the project, with approximately the length of 27 cm. Also, in Figure 9b, there is an example of these columns, that actually consist in three-stage linear actuators.

For this project, two of these modules were used, one for the desk and one for the sideboard. Each module has outputs for the motors, which are standardized connections for this purpose. Some modules offer three outputs and some offer two. For this application, only two outputs were needed, since both the desk and the sideboard were built upon two columns/legs. Figure 10 is a section taken from the *datasheet* of one of these devices that shows the connectors assignment.

As seen in Figure 10, the up/down switch is connected on the plug marked by the letter *S*. It is a 7-pin *DIN45329* connection, but it is imaginable that just a fraction of these pins are needed to make the table go up and down. As imagined, according to the datasheet [5] and through tests, it was found that only 3 pins are needed to control the

(a) *LOGICDATA* Control Module [20]. (b) Lifting columns [21].

Figure 9 – Control module and lifting columns.



① Motor socket 1 (M1)
② Motor socket 2 (M2)
S Handswitch socket (HS)
P Mains socket

Figure 10 – *LOGICDATA* Control Module connectors assignment (modified) [5].

motors, as shown in Figure 11. The more simpler hand switches, in fact, just use these three pins, but the better ones come with little screens to show the height, thus using the *SERIAL* pins and others.

To make the table go up, it is necessary to close the circuit between the *UP INPUT* and the *COMMON/+5V*, and, likewise, to put the table down, do the same thing with the *DOWN INPUT* and the *COMMON/+5V*. The solution to this part is, then, using relays to close the connections for up and down, then using a programmable board to

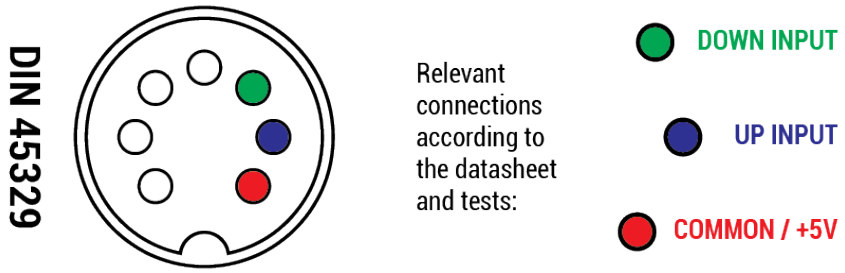


Figure 11 – Pins for controlling the motors.

send the commands to switch the state of the relays. Another part of the height control is being able to set a specific height for the table. This is only possible using a distance sensor. The description and information about this sensor will be further explained in the Section 4.4.1 and the application itself in the Chapter 5.

The chosen board for this application is an *Arduino*, which is simple, has enough processing to support all the code for this task and can also be used for other applications on the desk itself. The goal then, is to receive commands to go up, down, or stop, and make sure that the activation is flawless, avoiding that both the UP and Down relays are switched on at the same time, or that the code enters in a loop.

4.2.1 Arduino

Arduino is an open source prototyping board. It is based in an easy to use software and hardware. The basic principle is that it can read an input, a button, a signal from a sensor, a string through serial or internet, for example, process it and output something, a command, a led, a motor, etc. Since it is open source and well spread, there is a huge DIY community of students, hobbyists and professionals contributing with projects and support, as well as a huge variety of sensors and shields (that attach to the top of the board) for the Arduino. [22]

There is a huge variety of boards in the Arduino family, that differ in size, processing, memory and functions built in. One of the most common, and the one used in this project, is the Arduino UNO. It contains 20 I/O pins, being six PWM outputs and 6 analog inputs and uses the ATmega328P. In this case, only the digital outputs are used to switch the state of the relays. An example of the Arduino UNO board can be seen in Figure 12.

On each one of those pins, except for the GND and V+, can be set as input or output. In this case, when the command is sent, the output is turning a relay ON or OFF, which is connected to one of this pins. Since the idea is to control it over the network, the only way of receiving these commands is connecting the board to the network. To do that, it is necessary a shield, since the Arduino Uno does not have network connection capabilities by itself (the UNO does not, some do, like the Arduino Ethernet).

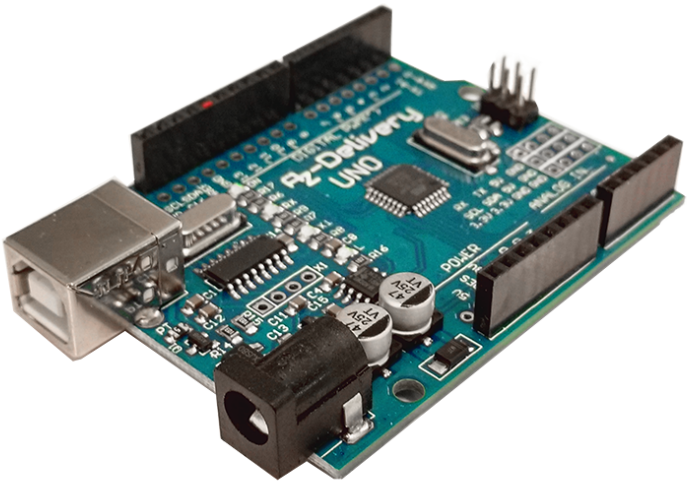


Figure 12 – Arduino board from manufacturer AZDelivery.

The two main options for the board are a WiFi Shield or an Ethernet Shield. The Ethernet Shield was the choice, mainly because it works with a cable, but can also be used wirelessly through external devices, like a mini router, which will be further explained. It also adds support to an SD card. Figure 13 is a picture of he Ethernet shield attached to the top of the Arduino Board. As is possible to see, the shield extends the I/O pins since it does not use any one of them.

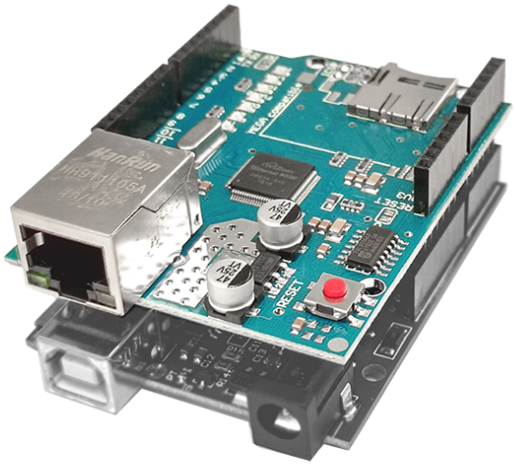


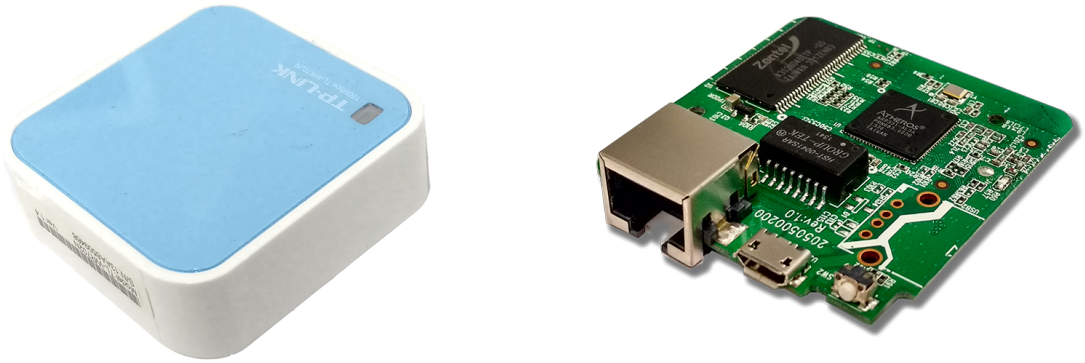
Figure 13 – Arduino with Eternet Shield.

As mentioned, even with the choice being the Ethernet Shield, for the desk control t would be useful to not have any internet cabling. Thus, the solution found for that was

using a Nano WiFi Router, specifically the *TP-LINK TL-WR702N*.

4.2.2 TP-LINK TL-WR702N - Nano Wifi Router

The *TP-LINK TL-WR702N* is the smallest WiFi router in the world, which can fit in the palm of your hand. It has a 150Mbps data rate and can be powered trough an USB connection. What makes it even more different is that it has multiple operating modes. That means it can work as a router, repeater, bridge, access point, but more important, as a client. Two pictures of the router can be seen in Figure 14 [23].



(a) Router enclosure. (b) Router inner circuit.

Figure 14 – *TP-LINK TL-WR702N* enclosure and inner circuit.

Working as a client means that it can connect to a Wireless network and then connect to the desired device via cable. That is specially practical for testing and debugging, since the connection to the network and the software in the Arduino are independent, and overall making it more robust.

The final setup for the height control is then:

- Router connected to the wireless network.
- Arduino, through ethernet shield, connected to the router.
- Module input connected to the the Arduino.
- Motors connected to the module output.

4.3 Light

As mentioned in the last chapter, an *RGB LED* lamp was chosen for this project, due to the fact that it fulfills all the requirements to improve the well-being of the worker and gives him more control of his own workstation. The lamp used in this project is shown in Figure 15. It consists basically of a LED strip wrapped around a translucent block of acrylic plastic, to spread it.

Figure 15 – Lamp from the SmartDesk structure.

To set a color for the lamp, it is necessary to set individual values (0 to 255) for each of the basic colors of the RGB system, red, green and blue. The combination of these colors is going to create the other ones. There are some ways to do that, one of them is doing that with 3 analog (PWM) outputs of some board like the Arduino. In this part, in fact, was also used an Arduino, but another solution was taken, in order to switch colors using just one output pin. For that was chosen to use the DMX512 protocol.

The DMX512, which means Digital Multiplexer, can transmit, in a small amount of time, 512 bits of data with a single channel. Each channel stands for a color, that can have values from 0 to 255, making it possible to control up to 170 RGB lights at once. [24] To use this with the Arduino, it needs a circuit that can convert this signal back to separate channels. In the Figure 16 it is possible to see a "shield" on top of the Arduino and ethernet shield.

This DIY shield basically consists in a integrated circuit that splits the signal from the Arduino output pin into two signals and then, as seen in Figure 16, the green circuits turn these signal into values for each color.

As explained in the precious section, to receive signals in the network is necessary an ethernet shield, if using an Arduino. That is also the case for the lighting, the chosen controller is also the Arduino and the Ethernet Shield for communication, but, this time, the router is not necessary since the lamp is attached to the sideboard.

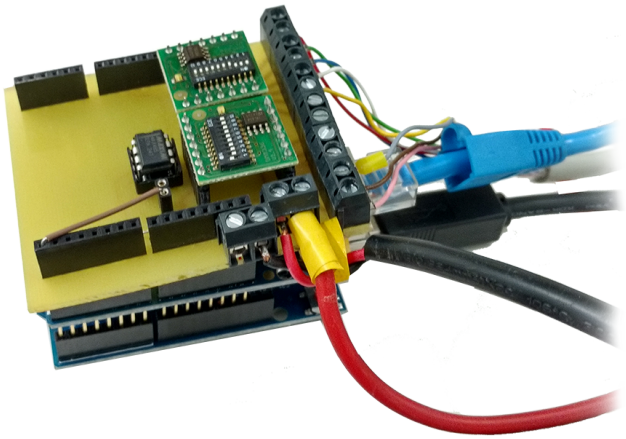


Figure 16 – Arduino, Ethernet Shield and DMX512 shield.

4.4 Data and Storage

This section discusses how the data will be acquired and stored. As mentioned before, the goal is to gather information about the quality of the work environment in order to improve the worker well-being. This data consists of:

- Light Intensity (LUX)
- Temperature (Celsius)
- Humidity (%)
- Pressure (Pa)
- Desk Height (cm)

The next subsections will explain everything about gathering and storing data. Subsection 4.4.1 gives details about every sensor used and how they work. In Subsection 4.4.2 it is talked about the device used for interpreting the sensors and communicating with the database and finally, in Subsection 4.4.3, it is discussed the database itself.

4.4.1 Sensors

Three sensors were used in this application, the *HCSR04*, for distance, the *TSL2561* for light intensity and the *BME280* for temperature, humidity and pressure. All of them work in a different form and use different types of connections/interfaces. An explanation about each one of them can be found following.

4.4.1.1 Distance - *HCSR04*

The HCSR04 is an affordable ultrasonic distance sensor. It works by sending an sound signal in a very high frequency (ultrasound) and then listening to it after it reflects. The time spent between sending the signal and it reaching back to the sensor can tell what is the distance of the object of interest.

Figure 17 – Ultrasonic sensor HCSR04.

It uses a simple I/O interface, where, the trigger pin is the output, that sends the signal to trigger the sound, and the echo pin is and input, that sends a signal when the sound is heard. It works in a range from $2 - 400\,cm$. Since it is an affordable sensor, the reading is not exact every time, so filters may be needed. Further explanation on it is found on Chapter 5.

4.4.1.2 Light - TSL2561

The TSL2561, as statet in the datasheet [25], is a light-to-digital converter that transforms light intensity to a digital signal output capable of direct I2C interface. In other words, it measures the level of light in the ambient and provides a digital signal. It combines two photodiodes, one broadband, ranging from the infrared spectrum to the visible light and the other infrared-responding in just one IC that can provide a near-photopic response, which means it approximates the human eye. The digital output, then, can be used to calculate the illuminance (LUX) through an empirical formula.

As mentioned, it uses the I2C bus connection. This interface allows for connecting multiple peripherals to be connected in the same 2-wire connection and have a 2-way communication. It works in a master-slave setup and can use multiple slaves ad multiple masters. The principle is that every slave in the bus has a unique address and only talks when the master addresses it [26].

The I2C bus is very useful because of the simplicity in its connection and the power of supporting multiple devices, that is why many sensors use this interface and come already with the necessary configuration to plug and use, through the SCL (Serial Clock)

Figure 18 – Light-to-digital converter TSL2561. Height = 2cm.

and SDA (Serial Data) pins. That was one of the reasons for the choice of the TSL2561 sensor and, as can be seen in Figure 18, has four necessary pins (Vcc, GND, SCL and SDA) and the last one is for changing the address.

4.4.1.3 Ambient - BME280

The BME280 is a digital humidity, pressure and temperature sensor. It is extremely compact once the module itself occupies a space of only 2.5mm x 2.5mm and has a very low power consumption. It is indicated for various applications such as home automation control, indoor navigation, health monitoring, IoT, outdoor navigation and so on, because of its high accuracy [27].

As it is a combo of 3 sensors from Bosch Sensortec, it is improved for the combined application. The humidity sensor works in a very wide temperature range while the pressure sensor is a very accurate absolute barometric pressure sensor. Both the temperature and pressure sensor have lower noise than their standalone versions and higher resolution [27].

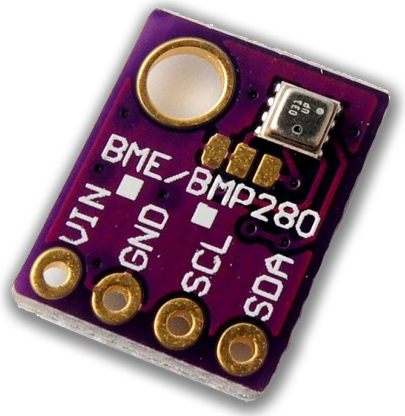


Figure 19 – Combo sensor (Pressure, Humidity and Temperature) BME280.

As seen in Figure 19, the sensor itself is inside the metal housing, but for usability, it is built in a bigger board. In this application it also uses the I2C bus for communication, as seen in the stamps above the pins, which mark the Vcc, GND, SCL and SDA connections. As previously mentioned, each device in the I2C bus has a different address, thus making this sensor and the light sensor share the same two pins.

4.4.2 Raspberry Pi

The Raspberry Pi is a single board computer, which means it has the functions of a normal computer but with a reduced configuration and size, designed for projects that require a bit more complexity than an Arduino could handle. As a normal computer it can be plugged in to a screen and also used with mouse and keyboard. The most common operating system, and provided by the Raspberry Pi Foundation (the creator of the board), used for the device is Raspbian, which is based in Debian, a Linux distribution [28].

It is a complete computer, so it is programmable in itself, in it's own interface, meaning that it is possible to install any IDE compatible with the operational system installed in it and also, the system can be accessed trough VNC, which is a remote viewer, thus being able to program in it remotely. On Figure 20 it is shown the model used in this project, which is the Raspberry Pi 3 Model B+. One of the main advantages is that it has built-in WiFi, thus avoiding the need for external shields or hats.

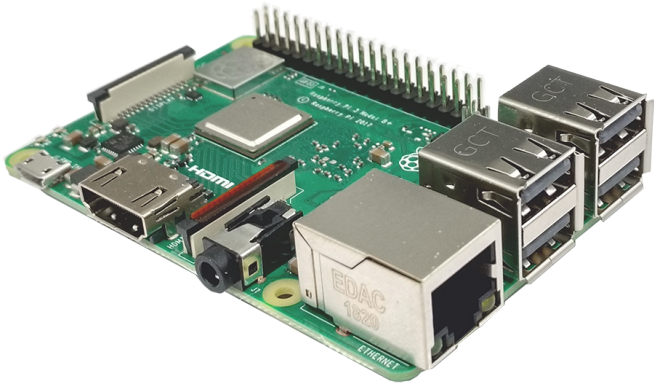


Figure 20 – Raspberry Pi 3 Model B+.

The Raspberry Pi 3 Model B+, as seen in the picture above, has mainly 4 USB ports, 1 audio jack, an HDMI port, an Ethernet port and, more importantly, the main difference and advantage compared to regular computers, which is the GPIO. GPIO stands for General Purpose Input-Output and are the metal pins. In other words, as the Arduino, it has programmable inputs and outputs. These GPIO pins are the main reason for the choice of the Pi, since the sensors can be plugged into them.

Three different sensors are used in this application, one uses regular I/O interface and two use I2C bus. Two of the GPIO pins of the Raspberry Pi are the I2C bus pins, the SCL and SDA, making the device completely useful for the set of sensors chosen. The role of this computer is to read the sensors, interpret the data, and then send it to a database, where the data is going to be stored in order to be read afterwards. For that a program is needed.

The chosen programming language for the main software running in the Raspberry Pi is Java. Since it is platform independent, well documented and has a great support for development in this board, such as libraries, IDE and a huge community of developers. Even though Python is the main development language, recommended by the RPi Foundation, since some of the sensors are have a bit of complex calculations, separating the tasks in classes seems like a wise choice and Java is perfect for it, since it is an object-oriented programming language. Also, the chosen IDE is NetBeans.

Regarding the program itself, the task for the software embedded in the Raspberry is basically:

1. Read the sensors.
2. Calculate the values for each reading (desk height, illuminance, temperature, pressure and humidity).
3. Connect to the database (located remotely in the network).
4. Write the values to the database alongside the time of the reading.
5. Repeat each X seconds.

The interval is not that relevant to the process itself, it relies basically on what can be handled without much effort from the Raspberry Pi. That is also the reason why the database is stored in a full size computer, because writing the data is not fast enough on the board.

4.4.3 Database

The choice for database, as stated, is the relational database, or SQL. The most common SQL database is MySQL. The choice for this one is based on the large support for this platform and how easy it is to use it. It is organized in tables, as mentioned before, so every database in the system has tables and every table has columns which separate the different variables and the rows that separate each registry/value.

The SQL database uses queries to open tables, insert, retrieve, modify and delete data. What the program described in the last subsection does is autonomously using queries to access the database and insert the sensor data alongside the time of the reading. That

is also one of the things the dashboard does, by showing the last registry, which will be explained in Section 4.5. Although both the website and the software on the Raspberry Pi are programmed in their own language, what they do is simply writing the queries.

The simplest way to store simple data such as the one from the sensors, which are just numerical values (doubles) is to put it all in the same table, since it speeds up the process of posting the information and retrieving it, which can be done in the same query. Below, in Table 1, it is possible to see the planned layout for the table carrying the sensor data.

Table 1 – Layout of table carrying the data from the sensors.

ID	Light (LUX)	Temp. (C)	Hum. (%)	Press. (mAtm)	Height (cm)	Time (date and time)
1	22.41	22.85	21.17	966.21	82.78	2018-12-18 16:53:28
2	22.56	22.86	21.15	966.22	82.75	2018-12-18 16:53:32
3	22.43	22.86	21.12	966.20	82.73	2018-12-18 16:53:36
4	22.56	22.85	21.17	966.22	82.75	2018-12-18 16:53:41
5	22.52	22.86	21.17	966.21	82.80	2018-12-18 16:53:45
6	22.70	22.86	21.14	966.24	82.61	2018-12-18 16:53:49

It is possible to observe some things about the fields. The ID field, for example, is automatically put in place by the database, it serves to identify each reading by its order and helps in sorting by order, making it easier to retrieve a specific registry or interval, which is vary useful for most applications. The pressure is stored in mili-atmospheres, or 10^{-3} atm. This is not a common unit, but there is no problem since the conversion is simple and can be simply done when retrieving the data to show in the desired interface.

Also, as stated, the data is stored in the form of double, which means it can have multiple decimal places, more than the two as exemplified in Table 1. It is not useful to have this many decimals (up to 16), and neither it is necessarily true. Taking the desk height, for instance, the readings are between 82.80 and 82.61 centimeters. That does not reflect the reality, which is that the table is not moving at all, it is just the noise of the sensor. The logical thing, in this case, would be to just use a 1 *cm* resolution, also because there is no need for more, and it would make no significant difference.

What can also be observed is that the readings are occurring each four seconds or so. That consists in the determined interval and the time it takes to read and post to the database. It is not possible to always be constant, but it revolves around this value. Since the interval is arbitrary, as mentioned in the Section 4.4.2, and for this application it is not necessary to be smaller than one second, it serves the purpose the best way possible.

4.5 Dashboard

Dashboards are mostly used for business and companies to track progress and analyze tendencies. They consist on a simple and visual way of displaying information to the user and takes his focus to the necessary and relevant data. They are linked to databases and

usually show data in the form of tables, graphs and key points. Dashboards usually are seen as websites or applications.

The name has come from the automotive dashboards, that basically take information from all the processes occurring in your vehicle and summarize them in a number of indicators, such as speed, temperature, RPM, gas and alerts, for example, the oil level, battery and so on. They serve the purpose of letting you conduce your vehicle with more information, thus more safely, as a business dashboard also will, to keep you focused on the important variables to maintain your business running [29].

It is being considered, in this case, also a broader definition, because obviously dashboards have applications other than business, as they are seen in health (Figure 21), weather, web insights and so on. The dashboard, then, is the center of information in which the user can better see what relevant information it may need, regardless of its nature. In this case, the environmental data, and the height of the desk.

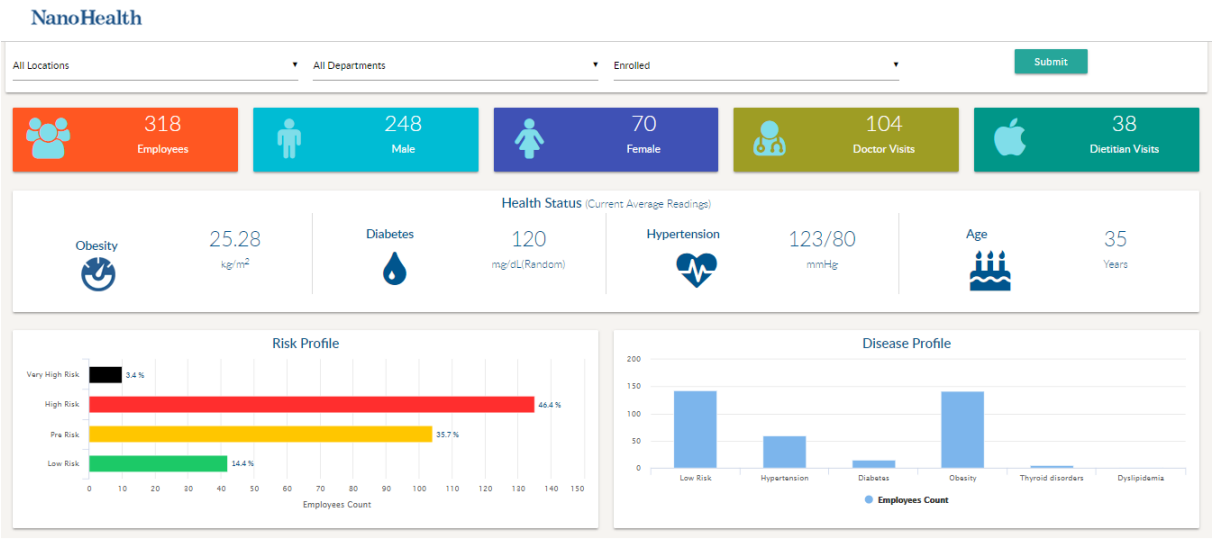


Figure 21 – Health Dashboard for Management [6].

As mentioned, dashboards tend to show information in graphs and tables. It is also possible to do that with this project and is a probable future implementation, but for this prototype, the focus is to show just the last reading, giving up-to-date information to the user. Also, the complete system actually consists in more than a dashboard, but rather a control center. The term dashboard will keep being used because of the intended organization of the interface, with blocks of interest.

4.5.1 Responsive Website

One of the mains reasons for the choice of building a dashboard is that its layout comes in handy when adapting for multiple devices. The doubt about whether to build an app or a website was solved by adhering to the multi-platform option. It was thought

that it would be much more useful to be able to access the dashboard from any capable device in the network, that is why a website was chosen.

The important part is the responsiveness. It serves no purpose having a website if, when accessed trough a smaller device, it does not adapt to the screen or supports the touch display. Responsiveness, in this case, means "responding" to the size of the screen, showing the content of the website in the best way possible. About the dashboard concept being useful for it, that means that the concept of having cards that shows specific information and have controls separate from one another makes it easier to reorganize the content, as illustrated in Figure 22.

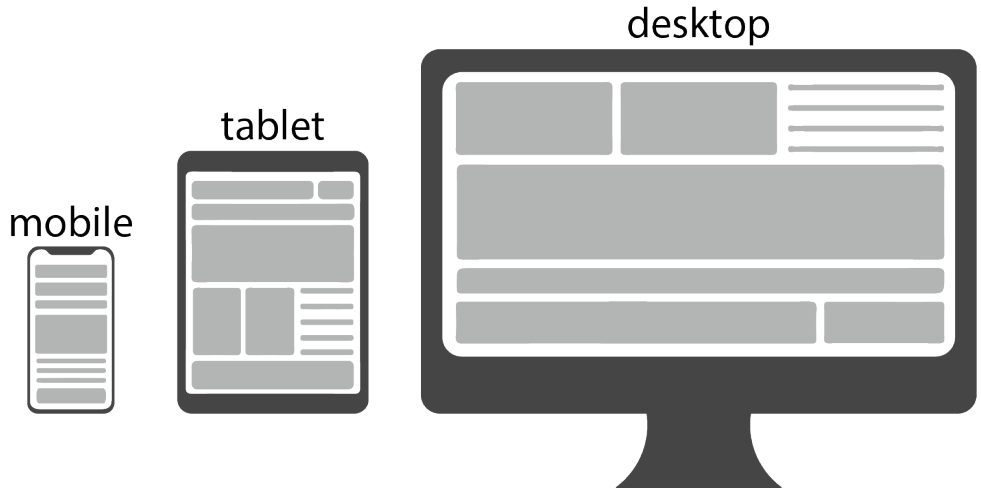


Figure 22 – Illustration of a responsive interface in three different screens.

To build a web application, even more a responsive one, differently from building a software, it is needed more than one language. Even really basic web pages usually use at least HTML, which means Hypertext Markup Language and it is what guides the structure of the website, links and so on, and the CSS, which means Cascading Style Sheet and, as the name accuses, it takes care of the style of every element.

In this case, more languages are needed, due to the functions of the website. These are PHP and Javascript. Presented below is an overview of the web development languages to be used in this project. [30]

- HTML - It is a markup language used to structure the page, which means putting the base content and giving semantic information about it.
- CSS - It gives the website information about the visual aspects of each element, in other words, how should they be displayed, using properties like color, size, relative position, visibility and many more.
- JavaScript - It is a scripting language. It makes the site dynamic and responds to actions from the user, having access to the whole page and being able to alter elements from the HTML, CSS, PHP and also sending requests.

- PHP - Means Hypertext Preprocessor. It is a server-side language and operates alongside the HTML code, changing it dynamically. It is often used for content management, which means databases and contact forms and posts.

The language responsible for giving the website its responsiveness is the CSS. It allows dynamic changes in the styles according to the size of the screen. One of the new features of this language is CSS Grid, which allows dynamic 2-dimensional structures to be built and it "substitutes" the HTML in the layout organization. Further information on that is found in Chapter 5.

The layout, then, as mentioned, is going to be separated in specific cards for each part of the workstation. The cards are an easier way of organizing the information, to better visualize the content, and they also can easily switch places when the screen have different sizes, thus supporting the responsiveness of the page.

4.5.2 Functions

As briefly mentioned, two are the main functions of the dashboard: showing the sensor data and controlling the workstation. For that, some coding is needed in order to make everything work. At first, PHP is used to retrieve the information from the database, locally, since the database is hosted by the same computer as the website. The data, then, is allocated in each respective card for a better visualization.

The second and equally important function is sending commands to the workstation, thus controlling it. For that, JavaScript is used, because it can send HTTP requests, and as the controlling devices like the Arduinos are connected to the network, by wire and wirelessly, they can receive the requests and act accordingly. That means that there will be buttons to make the desk and sideboard go up an down, to open the lockers and the drawer, to control the light intensity and also, for changing the color of the light, a color picker, which ultimately, sends the desired color to the Arduino controlling the lamp.

That resumes the dashboard in four cards, those are:

- Ambient: shows the ambient data. Has a bigger emphasis on the Temperature, once it is the most important variable of the three.
- Light Control: shows the light intensity and has the dimmer for the light and also the color picker.
- Height Control: shows the height of the desk and has the height controllers both for desk and sideboard separately and synchronized. Has a button for synchronization and a field to put a specific height for the desk to go to.
- Security: it is the card that controls the lockers and drawer. Basically has 3 buttons, one for each locker and one for the drawer.

4.5.3 Server

To host a website, being it on the network or the whole internet, the device which is hosting has to act as a server. That consists on providing access to your computer for the others in the network and handling the files in the domain. For that a HTTP Server is needed. The most common one used is Apache [31].

The solution found for this prototype was XAMPP, which is a platform Apache distribution that already contains PHP interpreters (and other script languages like Perl) and the MySQL database. That is actually the reason for the name, as it stands for (X Apache MySQL PHP Perl). Is one of the most common used servers because it is simple and easy to install, also offering a nice and clear interface for managing all the services, as seen in Figure 23 [32].

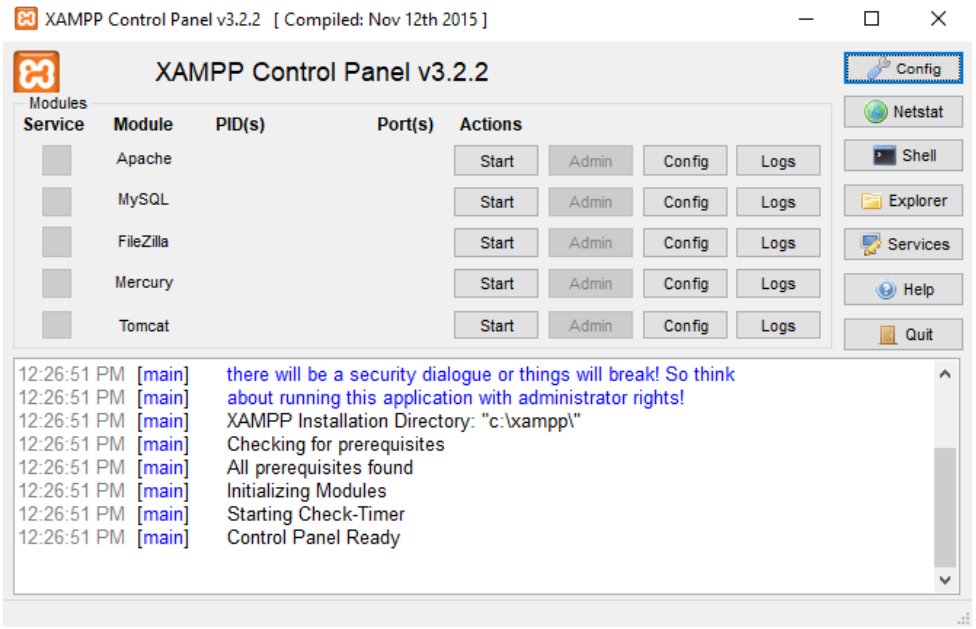


Figure 23 – XAMPP control panel.

4.6 Conclusion

This chapter described the whole project of the SmartDesk. Basically, it detailed what devices, softwares and languages are going to be used, and why. It also gave a better understanding of how every part of the workstation is going to work. The next chapter consists in the construction of this project. In other words, it will take what was planned in this chapter, execute and describes every aspect of the process.

5 The Build

This chapter describes the building of the project. The building consists in wiring, programming and configuring every device in the network. Chapter 4 described the project, in other words, everything that needed to be done, this part, then, tells how everything was done in the most detailed way.

5.1 Desk

In Chapter 4 every part of the whole workstation was explained. The wiring of the desk, though, was not mentioned. On the broader view, the power supply for the desk is divided in 3 different sources, 220 VAC, 12 VDC and 5 VDC. The first one feeds the outlets on the table, the module and, of course, the adapters for the other sources. The 12 VDC is for the RFID reader and the Lock (through a relay, because when fed 12 VDC it unlocks), and finally the 5 VDC or regular USB source feeds the Raspberry, the Arduino and consequently the relays and sensors, as seen in Figure 24.

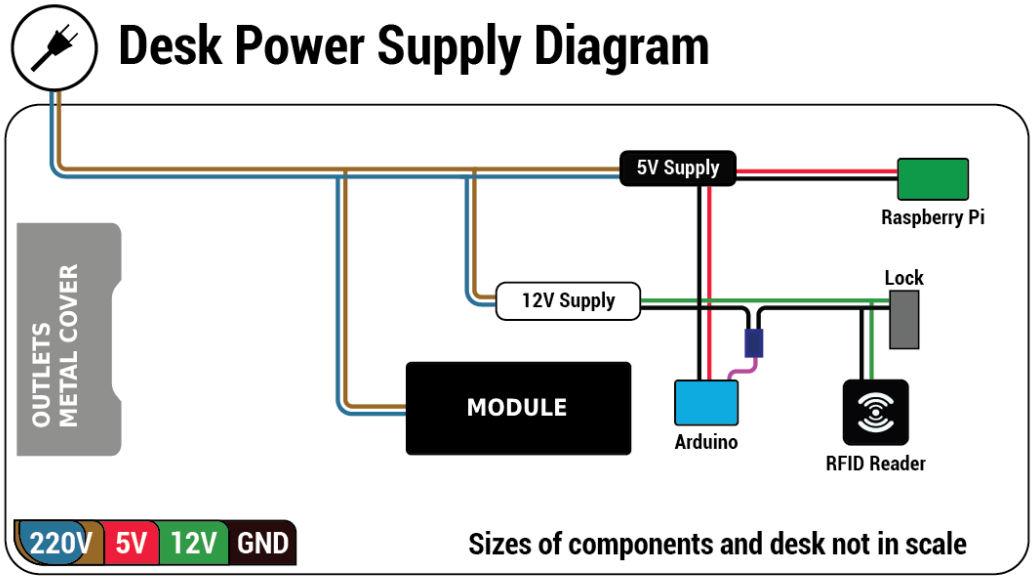


Figure 24 – Power Supply Diagram for the Desk.

The focus here are the two different cores of the desk. As explained before, there are basically two processing devices, the Arduino and the Raspberry Pi. For the sake of the explanation, the Arduino, which takes care of controlling the desk itself, is gonna be called the *Main Controller* and the Raspberry Pi that sends data from the sensors, the *Data Acquisition System*. The following subsections are divided between the two of them and the wiring schemes are expanded to the device level.

5.1.1 Main Controller

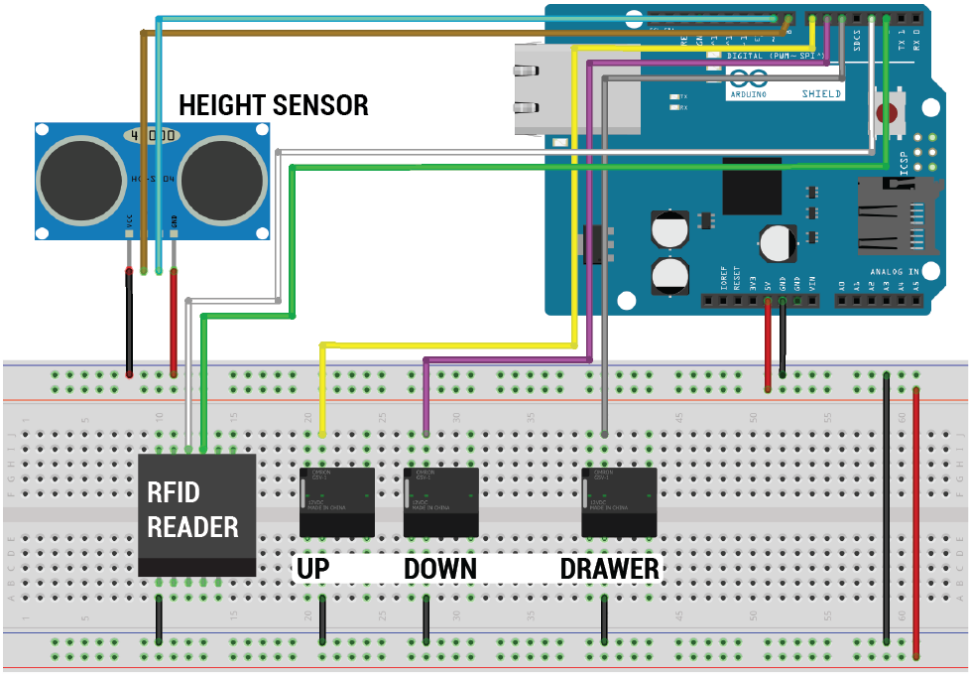


Figure 25 – Wiring Scheme for the Desk Arduino.

First of all, it is important to remind what functions the device assumes. As seen in Figure 25, there are three relays, one for the locker and two for controlling the desk height. There is also a RFID reader, which opens the drawer and a height sensor, for closed loop control of the height.

The connections to the Arduino, in the digital ports, are mostly arbitrary, aside from the two wires on the RFID reader, because they need ports that support interruption (2 and 3) for the Wiegand protocol, as going to be explained in Subsection 5.1.1.3. Some of the power sources are note represented in Figure 25, for example, the Arduino (USB) and the 12VDC for the RFID reader and also the drawer lock.

Another important part is the network connection, because the desk works by receiving HTTP requests from a client. The next subsections will detail every aspect of programming and configuring the device.

5.1.1.1 Ethernet and Routing

For the Arduino to be able to receive HTTP requests, it was needed to make it act as a server, so the client, whoever access the control panel (Dashboard) can send a request. The fist step was to configure the ethernet shield. Luckily there are some libraries for the Arduino that make this easier. Those are *Ethernet.h*, which in fact takes care of the communication and *SPI.h* which is necessary for the shield. As they are native to the Arduino IDE, the only necessity is to include them in the code.

```
#include <SPI.h>
#include <Ethernet.h>
```

The subsequent steps to configure the Arduino as a server is to set the addresses and the port. As an arduino code is divided in 3 parts, the first where the libraries are included and variables defined, the setup(), where the declared things are instanced and initiated before the loop, and the loop(), where things are repeated, these configurations must be done before the running code, and on the setup part the Ethernet communication needs to be started and after that, the server, as it follows:

```
byte mac[] = { 0xXX, 0xXX, 0xXX, 0xXX, 0xXX, 0xXX }; //MAC ADDRESS
byte ip[] = { 192, 168, 0, XXX}; // IP address in the network
byte gateway[] = { 192, 168, 0, 1 }; // Internet access via router
byte subnet[] = { 255, 255, 255, 0 }; //Subnet Mask
EthernetServer server(80); //Server Port

void setup(){
    Ethernet.begin(mac, ip, gateway, subnet); //Starts the Ethernet
    connection
    server.begin(); //Starts the server
}
```

The first value, the machine address, is the identification of each device, so it has to be different for each one in the network. The IP address is the one that makes the device be found in the network and it follows the layout of the subnet mask and the gateway. In this case, as it shows, the network has the address of 192.168.0.XXX, which means that the device has 255 possible addresses and in this case is fixed, although others may change their addresses.

On the setup it is possible to see that using these addresses the ethernet connection can be started. Also, as defined before, the port for the server is 80, then, after connecting to the network, the server is started. After that, the loop keeps the Arduino reading the requests from the client and interpreting them, if it matches any, then it calls the respective function. In technical terms, the text after the address is stored in an array of characters and those characters are compared with a list of possible words that can call functions, when there is a match, the function is called.

Choosing the arbitrary address of 150 (because it was in fact the one used in this device), any request from the client to the server on the Arduino consists in the URL that points to this address and a text thereafter that can be interpreted by the controller. An example that follows is the request sent for making the desk go down.

```
http://192.168.0.150/?DOWN
```

For the desk, since the connection to the network is wireless, the nano router was used. Configuring the device is fairly simple, but needs some attention to avoid confusion in the

network. The first step is to configure as a client, which means connecting to an existing network and using it as a wifi module for any ethernet capable device.

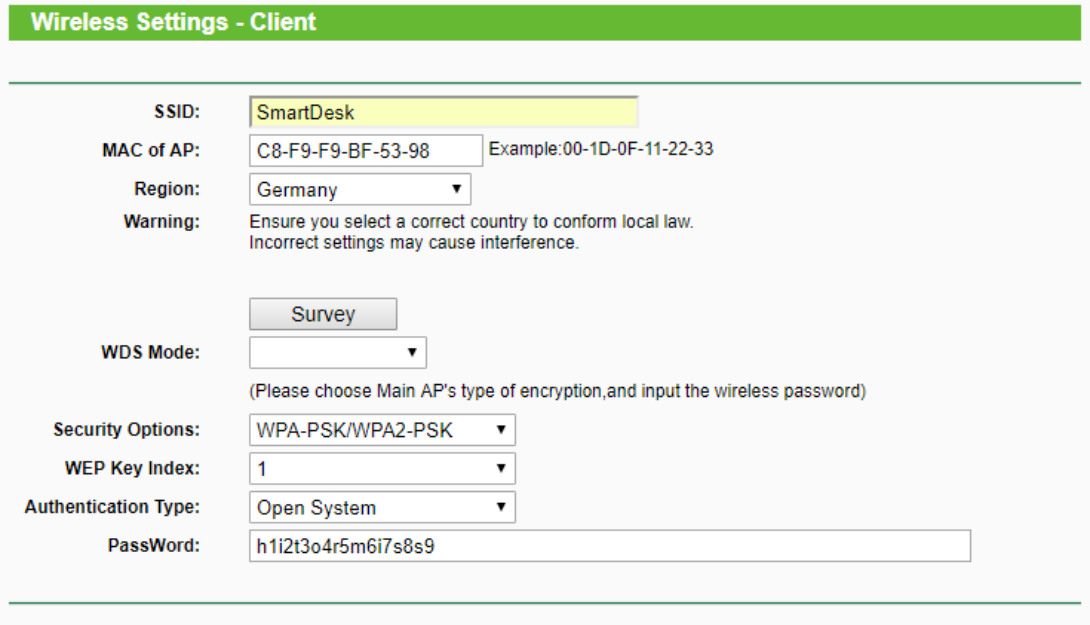


Figure 26 – Client mode configuration on the Nano Router.

The example in Figure 26 shows the router connecting to a network called SmartDesk, through a specific access point given its MAC address. It is possible to use the survey function, which lists all the reachable access points, and then the only necessary configuration in this part is putting the password, which was exactly what was done.

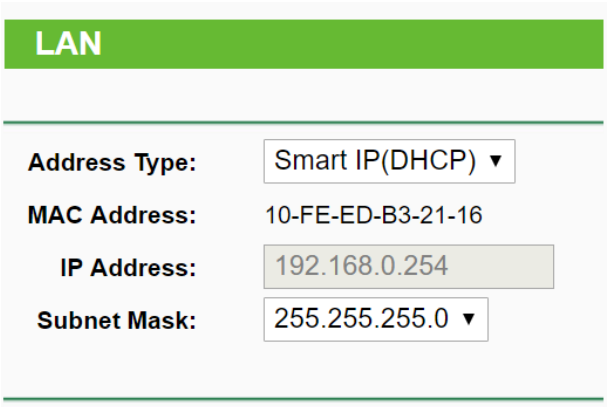


Figure 27 – LAN configuration on the Nano Router.

The next step is configuring the address of the nano router in the network. As shown in Figure 27, one of the options is to use DHCP, which stands for Dynamic Host Configuration Protocol, that chooses arbitrarily an address at each renewal of the addresses in the network. The other option is giving it a specific address, so it never changes. The last option was chosen for some reasons:

- The nano router configuration can be accessed trough the wireless network always from the same address. That is, not needing the effort to go to the list of DCHP addresses connected in the main router every time you want to know the nano router address.
- Every other device in the desk has a fixed address. It helps organizing and setting everything up faster when the network boots up.
- Guaranteeing that the router and the Arduino do not use the same address. It is important because the requests are sent to the Arduino through his address in the network. If the router has the same address there might be a confusion. Other advantage of having different addresses is that, if accessed, one shows the configuration of the router and the other one shows the client the "home page" of the server, even though it's a blank page.

5.1.1.2 Height Control

Following the connection to the network and the ability to receive requests from the client, the arduino has to use them to control all the things on the desk. In this section, the height control is explained. The first step is to set the outputs for the relays. So, two pins were chosen to be the up and down pins, and then they were set to be digital outputs, meaning they can only be on or off. The pin assignments might be different from the schematic, but as explained, do not matter this much.

```
1 int upPin = 5;
2 int downPin = 6;
3
4 void setup(){
5     pinMode(upPin, OUTPUT);
6     pinMode(downPin, OUTPUT);
7 }
```

This is the basic for controlling the height. The principle is that when the *upPin* is high, the desk goes up and when the *downPin* is high, it goes down. That mentioned, there are six functions for controlling the desk height according to the requests, those are: *?UP*, *?DOWN*, *?STOP*, *?RESET*, *?BOARD* and *?H=*.

The three main ones are *?UP*, *?DOWN* and *?STOP*, and the other ones are applications of them. In theoretical terms, it would be enough for the UP function to just turn on the upPin, and the DOWN function the downPin respectively, but, any mistake of clicking both or not stopping between one and the other would make both pins be high, thus leaving the system vulnerable to some problem in the operation of the desk. The solution is fairly simple and works perfectly for avoiding these situations

```
1 Up(){
```

```
2       digitalWrite(downPin, LOW);
3       digitalWrite(upPin, HIGH);
4   }
5   Down(){
6       digitalWrite(upPin, LOW);
7       digitalWrite(downPin, HIGH);
8       }
9   Stop(){
10      digitalWrite(downPin, LOW);
11      digitalWrite(upPin, LOW);
12      }
```

As can be seen, what the code does is make sure that every time one of the pins is on, the other one is off, except, of course, on the stop function, that turn both pins off and consequently making the desk stop.

The *?RESET* and *?BOARD* functions are very similar. One intends to put the desk in the highest point, to use it as a board (*?BOARD*) and the other (*?RESET*) to put it on the lowest, this one, for two reasons, synchronizing the desk and the sideboard, which means the sideboard also goes to its lowest point and resets. In fact, putting the desk to its lowest point is the factory designed way of resetting it.

Basically, the adopted method is to keep the desk, in these two functions, rising or going down for 10 seconds, because this time span guarantees that in whatever the height it is, it can reach its highest or lowest point. These functions implement passive countdowns, in other words, it counts on the background, not locking the system until the end of the count. These functions can also be stopped if wanted, just by moving the desk in some other direction or stopping it. The module both on the desk and sideboard guarantee that they do not exceed their course limits.

The last function, *?H=* is different. After the "=" comes a number, which is the desired height for the table. So, this function puts the table on the exact desired height. For that it uses the HCSR04 distance sensor. The easiest way to use the sensor is using a specific library.

```
1 #include <HCSR04.h>
2
3 int triggerPin = 13;
4 int echoPin = 12;
5 UltraSonicDistanceSensor distanceSensor(triggerPin, echoPin);
```

The code above shows how to use the sensor. Basically two pins are necessary, as mentioned in the last chapter. One to send the ultrasound signal and one to listen to it. The time between the sending and listening can be used to calculate the distance. In this case, when including the library, the only things necessary are defining which will be the pins (*echoPin* and *triggerPin*), and to read the distance, using the function *measureDistanceCm()*, which returns the distance in centimeters.

The function *?H=* compares the desired height with the current one. If it is lower, than the desk goes down, if it is higher, it goes up. Than, when it has less than $1cm$ of difference, it stops. This difference is put on due to the estimated delay between the command and the actual activation and deactivation of the motors. This concludes the height control functions. The method of sending the requests is going to be described in the Section 5.4.

5.1.1.3 RFID and Lock

The RFID reader works with the Wiegand Protocol. Briefly explaining, this protocol uses two data wires for reading cards, D0 and D1 and one common ground. It works with not only RFID cards, but with keypads, and whatever needed, but in this case, a card reader is being used. For that reason, a library was needed for the Arduino. The one used is originally designed by JP Liew [33] and supports various N-bit formats. It makes it very simple to use a card reader, as can be seen in the code below.

```
#include <Wiegand.h>
WIEGAND wg;
void setup(){
    wg.begin();
}
```

The standard pins for the data wires are pins 2 and 3. Since, for this protocol, the data pins need to support interruption, in the Arduino UNO there are no other options for that, which is not the case of the Mega, for example. Interruption is when, through a signal or change in the signal, the code interrupts to execute some function. The library wraps the code for the protocol, thus needing only to use the following function *getCode()* to read the code for the card.

The application for the reader, in the desk is opening the Drawer. As shown in the schematic, the drawer also works trough a relay, that, when activated, send current to the magnetic lock and then unlocks the drawer, making it possible to open it. There are two ways of opening the drawer, through an HTTP request (mainly in the dashboard) or with a valid card.

The drawer responds to the HTTP request *?OPEN* and to the card reader, if the RFID code matches one of the authorized codes, meaning it is a valid card from one of the rightful users. The opening function consists in turning the relay on for 10 seconds. As in the height control functions, it implements a passive countdown, meaning the main code still runs during the time. The 10 seconds are the time the user has to open the drawer, once it is open, it only locks again when it is closed, so the time is more of an arbitrary amount.

As could be seen, this is how the functions of the main desk controller were implemented in this project. In the next subsection (5.1.2) it is explained how the Raspberry

Pi was wired and coded for sending the sensor information to the database.

5.1.2 Data Acquisition System

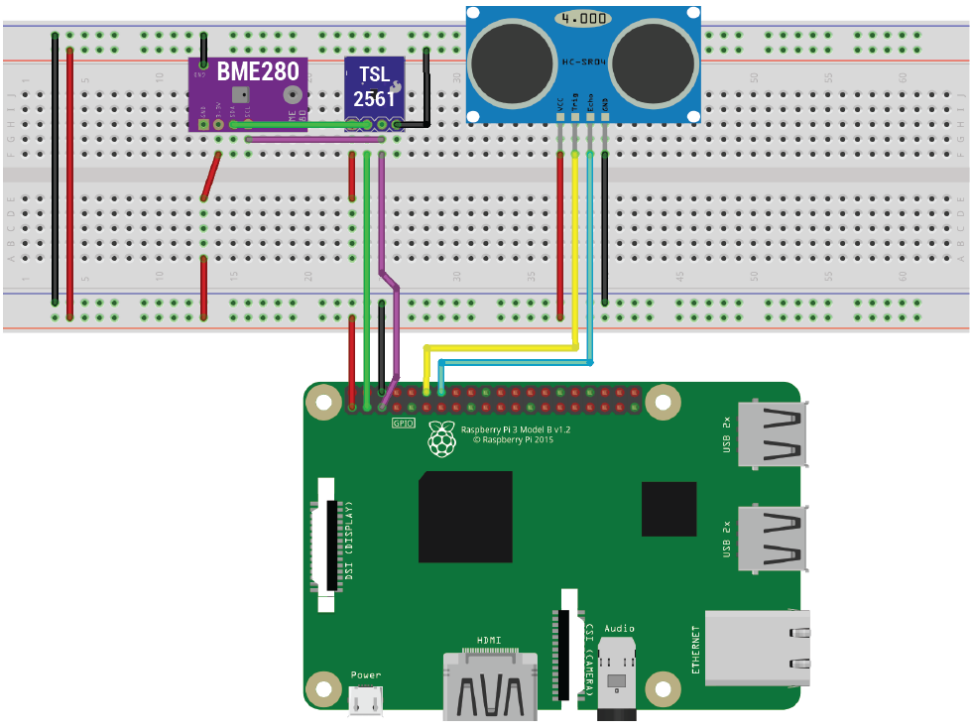


Figure 28 – Wiring Scheme for the Raspberry Pi

In Figure 28 the Raspberry Pi and the sensors attached to it are shown. As explained before, there are 3 sensors and 5 different variables being read. This device is different from the Arduino because it is in fact a computer, that can be programmed in any language and, as mentioned, was in Java. Java is an object-oriented language and that is why the code is separated in classes.

The Raspberry Pi runs the Raspbian OS. In this operational system, since it's linux-based, it is possible to install any system independent IDE and the chosen one was NetBeans. NetBeans has support for Java and a huge amount of libraries and it is easy for organizing in classes, debugging and testing.

The software has specific classes for each job, which are reading the sensors and posting to the database. For the sensors, there is a common class called *Sensor* and at least one specific to each sensor, that inherits the structure of this one. For the database there is a class with specific methods to connect and post to the database. This one relies on libraries from Java that give support for MySQL. For a better understanding of the structure, in Figure 29 there is a diagram of the most important classes and methods of the program.

As can be seen in Figure 29, there are different colors and shapes for different blocks. The square shapes are classes, and the ones with the rounded edges are methods. The

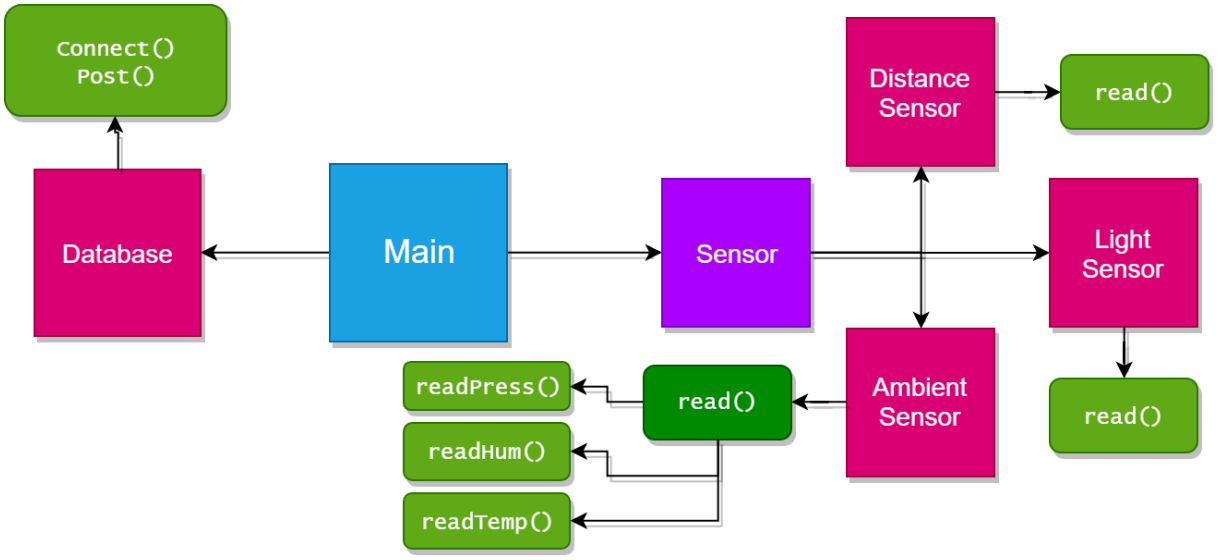


Figure 29 – Classes and methods diagram for the Raspberry Pi software.

blue block is the main class (as written on it), which is the class that uses all the others and execute the program. The *Sensor* class is in purple because it is an abstract class, but the pink ones are each one for a specific sensor and extend the purple, as the database one is for connecting the database, but does not extend any class from this project.

The green blocks are the methods for each pink class. The light green are the ones who return something and the dark green one, which is in fact inherited from the *Sensor* class is used for the individual readings of each value on the Ambient Sensor, which reads 3 variables. As the function of it all is sending the sensor information to the database, the steps for it would be as shown in the diagram in Figure 30.

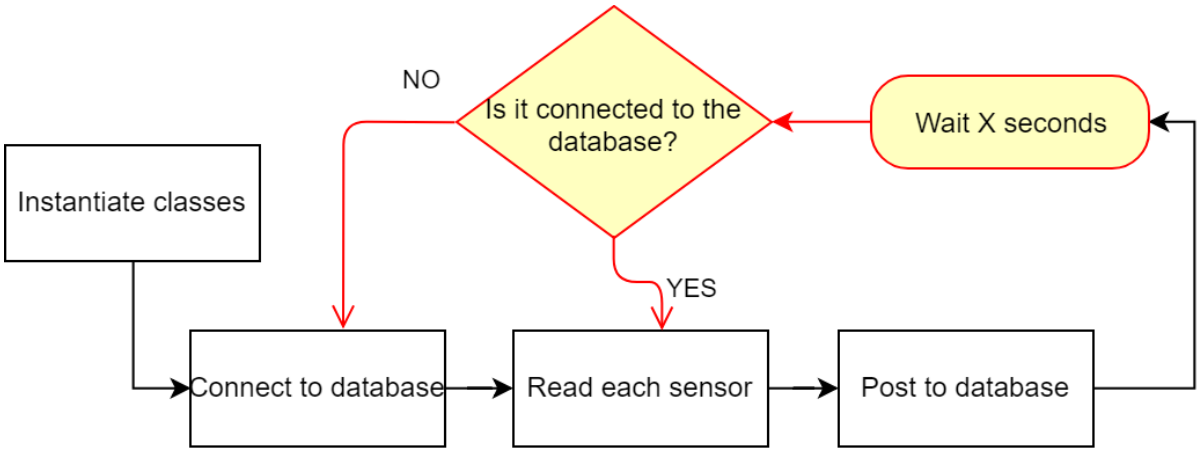


Figure 30 – Internal logic scheme for the Raspberry Pi program.

This is a summarized view of the structure of the program, but that is exactly what the object-oriented language offers, simplifying the organization of it all by packing the code in different classes, thus optimizing it all. Although the structure is straight forward, every individual sensor class has its specificities and resources, as well as the one for the

database. This will be described on the following subsections.

5.1.2.1 Reading

First of all, the structure of every reading of the sensors is based on the sensor class. It basically implements a method that reads the data and returns a double. Then, each specific sensor class (AmbientSensor, LightSensor and DistanceSensor) implements their own way of reading them, but always returning the reading.

What is common for every sensor, as mentioned before, is the GPIO, which is the set of input and output pins. The majority of the usual programming languages have support for this interface, but most of the times through external libraries and Java is no different than that. For java, the library for the I/O on the Pi is Pi4J. It provides a low-level integration and support for interruption. Since it allows to use the full capability of the GPIO, it also serves for the I2C bus, thus working with every sensor.

5.1.2.1.1 Distance Sensor

As stated before, the distance sensor is used to read the desk height. And works the same way it worked on the Arduino. This time, no library is used besides the Pi4J, instead, the calculation is done on the method *read()*, which already returns the reading.

To instantiate the function it is needed to inform the Trigger and Echo pins, which send and receive the ultrasonic signal. To read the distance is fairly simple, as follows:

```
1 DistanceSensor dist = new DistanceSensor(TriggerPin, EchoPin);
2
3 double Distance = dist.read();
```

Since the HCSR04 is an affordable sensor, sometimes it has wrong readings. To avoid them, a basic filter was formulated. The basic goal of this filter is guaranteeing that the function returns a valid value. A valid value is one inside the sensor reading range or the desk height, in this case, for the sake of testing, the sensor measure range was used. It returns a null value if a valid value is not found after a number of times.

5.1.2.1.2 Light Sensor

For both the light and ambient sensor, the interface used was the I2C bus. This, as explained, means that they share the wires, thus, not needing for any of them, to define which pins they are using. On the other hand, the device needs to be identified trough its address. The standard address for the TSL2561 is 0x39, but can be changed by connecting the specific pin to GND or V+. Changing the address is useful in the case of using more than one of the same sensor, once there can only be one device for each address.

For this sensor, besides reading the data, it has to be processed in a way it makes sense, approximating the light intensity itself. Because it uses 2 different devices, infrared and visible light, the response was tested and on the datasheet [25] there is the empirical formula that had to be implemented in order to achieve the desired reading. What follows is the comments for an example code in the datasheet.

```
1  //For Ch1/Ch0=0.00 to 0.50
2  // Lux/Ch0=0.0304-0.062*((Ch1/Ch0)^1.4)
3  //
4  // For Ch1/Ch0=0.50 to 0.61:
5  // Lux/Ch0=0.0224-0.031*(Ch1/Ch0)
6  //
7  // For Ch1/Ch0=0.61 to 0.80:
8  // Lux/Ch0=0.0128-0.0153*(Ch1/Ch0)
9  //
10 // For Ch1/Ch0=0.80 to 1.30:
11 // Lux/Ch0=0.00146-0.00112*(Ch1/Ch0)
12 //
13 // For Ch1/Ch0>1.3:
14 // Lux/Ch0=0
```

This calculation was then implemented in the method *read()*, on the *LightSensor* class. Alongside this method, there are registers and variables to control the gain, as well as other methods and constructors and the verbose mode. Though this class is well packed with features, the only method used for this project was *read()*.

5.1.2.1.3 Ambient Sensor

This sensor has a major difference compared to the others, which is that it reads more than one variable, being them the temperature, humidity and pressure. For that reason, the methods that return the desired reading are actually no the method read(), but rather three different ones, one for each variable, even though they depend on the original method.

The code for this sensor is actually an object-oriented adaptation of a code [34] designed for the sensors provided by Control Everything. It also uses data provided previously from the manufacturer in order to make the necessary calculations for the offsets and compensations. As it worked straight away, no modification to the calculations were done, just with the structure.

The test setup for the sensors is found on Figure 31. It follows the wiring shown in Figure 28. It consists of the Raspberry Pi attached to the bottom of the desk, the height sensor facing down and the light and ambient sensor side by side and facing up.

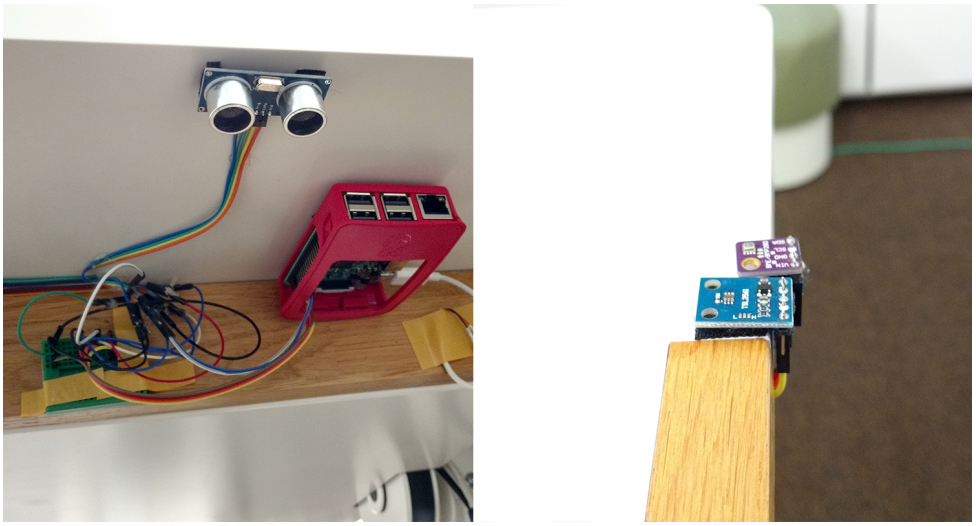


Figure 31 – Testing setup for the Raspberry Pi

5.1.2.2 Writing

For sending the data to the database, it was created a class with certain functions that connect and send the information. First of all, the library that allows the interaction between Java and MySQL is provided by the company itself and it is called *MySQL Connector J* [35]. This library allows the connection and interaction with the database through queries, that can go from creating a table to reading a specific cell in a specific row of a table.

Lots of constructors for the Database class were created for the purpose of testing. Those required from only the table name, to the user and password and the database name. Ultimately, for the final system, the simplest solution found was just putting the information already inside the class, ending up as something like the code below.

```
1 public class Database {
2
3     String url = "jdbc:mysql://192.168.0.102:3306/CognitiveEnvironments";
4     String user = "user1";
5     String password = "password1";
6     String table = "deskdata";
7     Connection con;
8 }
```

This data is necessary for the two main methods: Connect() and Post(). The method Connect() takes the string variables and the Connection one and creates a connection to the database. The other one, Post(), works differently. The main goal is posting a query statement to the database and the library is the interface that takes the query and effectively send it to the database, like this:

```
1 PreparedStatement post = con.prepareStatement("STATEMENT HERE");
2 post.executeUpdate();
```

The query statements, for once, are specific to the MySQL language, but follow a structure when handling data inside a table. They consist on the action, the table, the fields and the values. For example, if wanting to post some data into a specific table the statement would be:

```
"INSERT INTO deskdata (light, temp, hum, press, height, time) VALUES
    ("200", "23", "30", "969", "80", NOW())")
```

The line above is actually the structure of the query used in the method Post() in this project. the first five fields hold numerical values in the form of doubles and the last one is the timestamp, which is the date and time of the reading, so track of time can be kept.

5.2 Sideboard

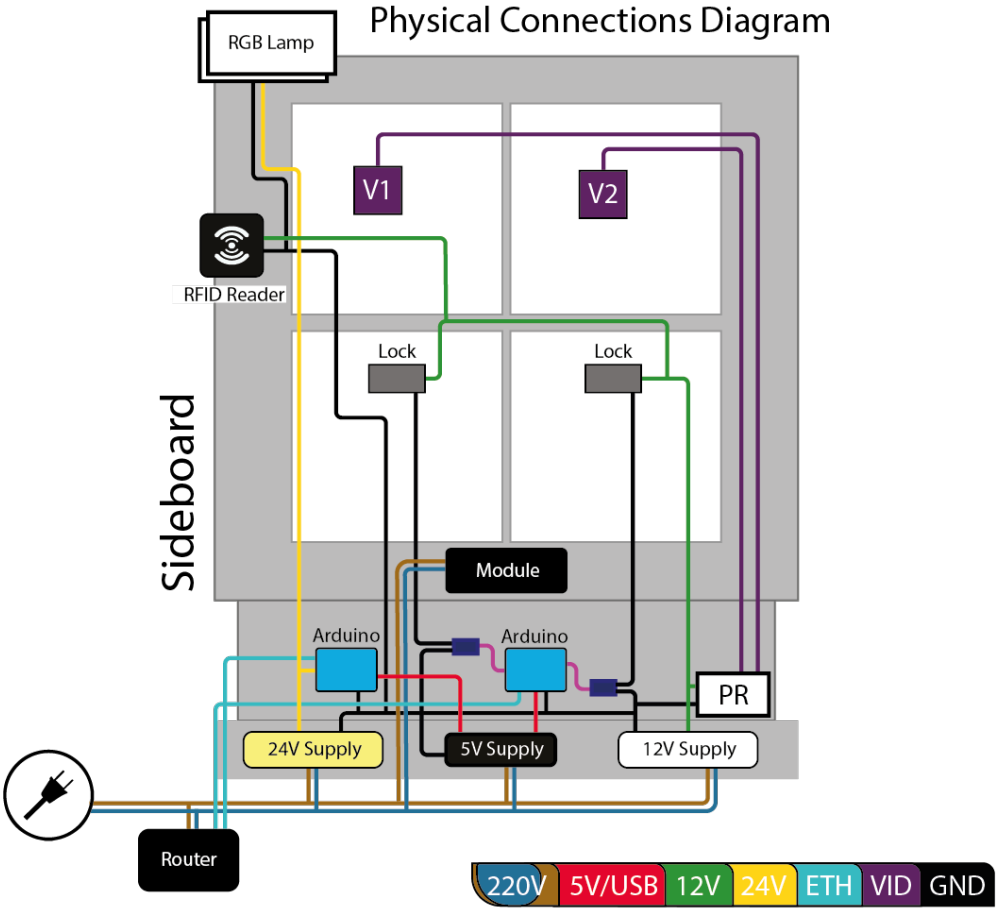


Figure 32 – Physical Connections Diagram for the Sideboard

As the diagram in Figure 32 shows, there are much more connections and sources on the sideboard, compared to the desk. Although there are some similarities, as in the main Arduino, there are more variables to be controlled and is also home to the network router.

Differently from the desk diagram, in this one it is important to also show the ethernet and video connections, as well as the 24 VDC power source for the light.

As can be seen, there are, in purple, two squares, V1 ad V2. They represent the two video monitors, and the device from which the video connection comes from, identified as PR, is the Port Replicator, further explained in this section. The router is the device that supports the network, as there is no need for wireless connections in the sideboard, the two arduinos connect via ethernet cable to it.

As in the desk, there is a 12 VDC power source, this time for one more lock. There is also a 24 VDC power source, as mentioned. This one is for the RGB lamp and the arduino shield, which is also powered at this voltage. All the DC sources share a ground for less interference.

The next subsections dissect the two main controllers, this time two arduinos with different functions, and the other two devices that have important roles on the project, which are the Router and the Port Replicator.

5.2.1 Light Control Arduino

As any of the other controllers managed via the dashboard, this one works also through HTTP requests from the network. It received the arbitrary address of 192.168.0.52, because it is easy to remember, since the sideboard main controller is 152 (which is going to be mentioned further ahead). The function of this unit is controlling the color and intensity of the light. For that it relies on two different types of requests, one that control the color and other that controls the intensity of the light.

The first, which controls the color, consists basically in the colors of the RGB scheme, disposed as follows: 192.168.0.52/?r=255&g=255&b=255. This example is the color white, which is all the values at the maximum, but as known, they can be from 0 to 255. That said, the Arduino has to read the request, which consists in a string, and separate it in each value for the red, green and blue colors. Being *url* the string taken from the request, the way the values were separated was the following:

```
1 red = url.substring(url.indexOf('=')+1, url.indexOf('&')).toInt();
2 green = url.substring(url.indexOf('g') + 2, url.lastIndexOf('&')).toInt();
3 blue = url.substring(url.indexOf('b') + 2).toInt();
```

The other type of request is the up and down, to control the intensity of the light. It makes the power go up or down by 10% each time, so all the colors are equally multiplied by this decimal number that goes from 0 to 1 in intervals of 0,1. The original colors are stored in a different variable in order for the color and the intensity to be really changed separately and to avoid resetting the intensity each time the color changes.

To effectively make use of the DMX protocol, a library was needed to make it easier. The DmxMaster library [36] was the chosen one. It reduces the job to only selecting a

pin, selecting a number of color channels and attributing a value for each one. All the processing and transformation to a 1-channel signal is done by the library. As follows:

```
#include <DmxMaster.h>

void setup() {
    DmxMaster.usePin(0);
}

void loop() {
// Setting the color to White
DmxMaster.write(1, 255); // R
DmxMaster.write(2, 255); // G
DmxMaster.write(3, 255); // B
}
```

The lamp working is shown in Figure 33. The color is majorly green with some blue in it. Basically, with the library and the circuit that decodes the DMX is very straight forward to work with the RGB lamp. The circuit was already built in the structure and the choice was to use it, as it is very practical, once it was found out what it effectively was. The end result is a lamp that can change its color and intensity in a extremely fast way and can be controlled remotely.

Figure 33 – RGB lamp showing a greenish color.

5.2.2 Main Arduino

The main controller of the sideboard is the arduino presented in Figure 34. It looks very similar to the one presented in Section 5.1, from the desk. The similarity is explained,

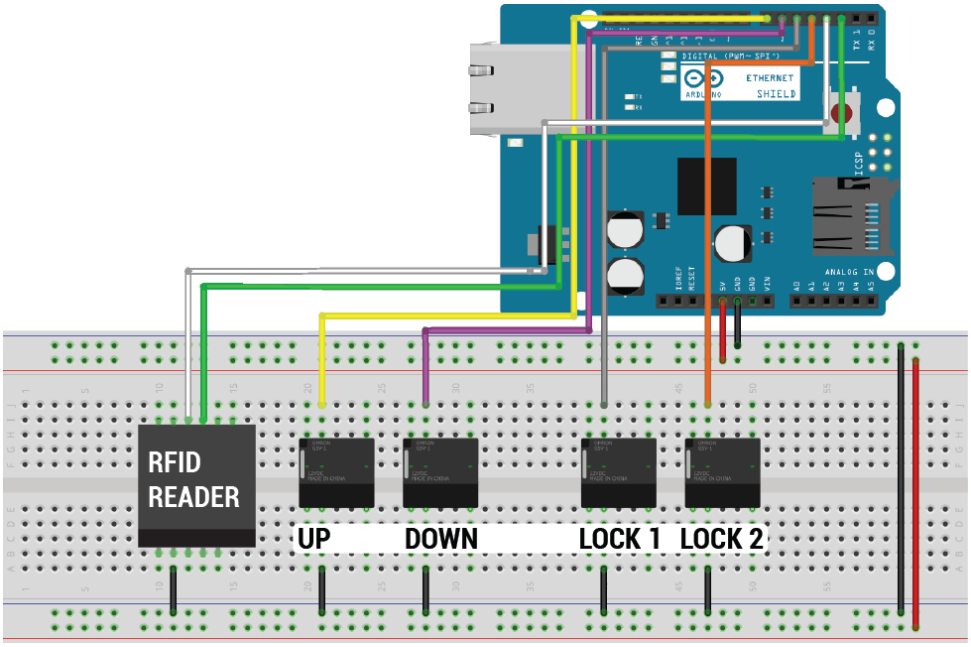


Figure 34 – Wiring scheme for the Arduino on the Sideboard

of course, through the functions they have in common, since this one is also responsible for controlling the height and opening lockers. There are actually two main differences between the two controllers, which is that this one does not make use of any sensor, for reasons previously pointed out and, instead of a drawer it has two lockers.

The program inside the Arduino is also almost the same as the desk one. As mentioned, for connecting everything in the network, it is necessary that every device has a different address and, since the other arduino took the address of 150, the chosen one for this was 152. Another difference regarding the network connection is that as the sideboard carries the router and is intended to not move around, there is no need for the Nano Router, since the connection is via cable.

About the main code, aside from the changes in the address, which are pretty straight forward, the limitations of the Arduino UNO were considered when working with the sideboard. As the Arduino UNO has only two pins that support interruption, no more than one RFID reader could be put in place there. As there were two lockers, the decision was that the reader would open both and, through the dashboard, they could be opened separately. The following example code shows how it turned out.

```
1 Open1(){
2     digitalWrite(l1Pin, HIGH);
3     delay (100);
4     digitalWrite(l1Pin, LOW);
5 }
6 Open2(){
7     digitalWrite(l2Pin, HIGH);
8     delay (100);
```

```
9       digitalWrite(l2Pin, LOW);
10  }
11
12  if(auth){
13      digitalWrite(l1Pin, HIGH);
14      digitalWrite(l2Pin, HIGH);
15      delay (100);
16      digitalWrite(l1Pin, LOW);
17      digitalWrite(l2Pin, LOW);
18  }
```

As may be noticed, the pins for the lockers are l1Pin and l2Pin and the process for opening the lockers is basically setting the output high for a tenth of a second, which is enough for the spring in the lock to push the locker away from the magnet. As this amount of time is not considerable, though it effectively stops the system from changes in the mean time, it was chosen to not do a passive timer. Also, to save time, the functions were not used in the RFID authentication, because they would double the time in which the system stops and also desynchronize the opening of the lockers.

The other function, aside from the normal control is *?RESET* and it works the same way it did on the desk. The reason is that the goal of this function is synchronizing the desk and the sideboard. Those conclude the functions of the devices in the sideboard. The next section approaches the other equipment and devices used in this project.

5.3 Additional Equipment

The physical part of this project would not be possible without some additional equipment not mentioned before. The reason for not mentioning is that it is neither built or coded here, is just ready-made but important for allowing the project to function. The first one is the Wireless router, without it the network cannot be created and the second one is a port replicator, that, as the name states, creates more ports.

The next subsections are going to approach the configuration of the router for the proper work of the network and also describe the port replicator and why in fact it is necessary for a workstation like this.

5.3.1 Router

For a network to exist, it needs a router, being it wired or not. Its job is to manage the traffic of data, creating routes for the information to go from a device to other. In this case, a domestic router was used, the most common one for small scale applications, and this one did not need anything more than that. The router model used was the TP-LINK AC750, which has two antennas, as shown in Figure 35.

Figure 35 – Router TP-Link AC750 [7].

There are two important things to configure in the network. The SSID and password are one and the other is the address reservation. As it works with DHCP, a new address is given to every device on each after each renewal interval. As there are some devices that have fixed addresses, like the arduinos, but there are some others that even though they do not have it in their own setup, it needs to be configures trough the router.

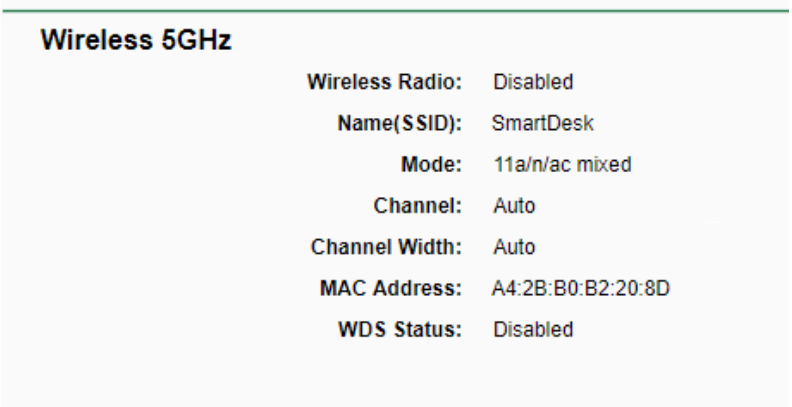


Figure 36 – Network basic settings.

In this case, the network was called SmartDesk, to be easy to find since it's the name of the project, as can be seen in Figure 36. Since the router is in fact one of the devices in the network, it also has an address, which is 192.168.0.1 (standard). There are two other addresses that need to be reserved (for this project), which is the server address, that contain the dashboard and so on, and the Raspberry Pi one, so it is always accessible. They were defined as:

- Server: 192.168.0.102
- Raspberry Pi: 192.168.0.104

The configuration could have been done differently, but there was no need for more than these settings. The next part explains the Port Replicator.

5.3.2 Port Replicator

As the SmartDesk is not resumed to sensors and the network, but rather a place to develop knowledge work, it needs all the resources of a workstation. That includes monitors (two, in this case), the ethernet connection, mouse and keyboard. Since the principle of it all is sharing, it is supposed that every person has his own computer, usually a laptop. This supposition is supported by the actual fact that in the office where the project was developed, everyone had its own device.

That, then, is the job for the Port Replicator. Since laptops in general have a limited amount of connectors, either for video, audio, USB, etc., it is not possible to plug two monitors, network, and all other devices directly to the computer. The port replicator, then, uses a USB interface to replicate those necessary ports. It contains usually audio, video (2x), USB and Ethernet, so the choice is arbitrary to what can be used.

Figure 37 – Lenovo USB Port Replicator [8].

The device used was a Lenovo Think Pad port replicator. As seen in Figure 37 it contais five USB 3.0 ports, two DVI, one RJ-45 (ethernet) and one audio I7O P2 jack. This PR is extremely powerful for its size because it uses the DisplayLink DL-3900 chip inside, which means that each monitor can support a resolution up to 2560:1600. It is an relatively older version, since it supports only DVI and the standard nowadays is HDMI or DisplayPort, but for this purpose it fitted perfectly.

This concludes the setup of the sideboard and consequently, the physical part of the project. The next section (Section 5.4) will discuss the building of the dashboard, and all the aspects involved, as well as end the chapter.

5.4 Dashboard

The dashboard layout, as mentioned, consists in four blocks that reorganize according to the screen size and also adapts to the input method, being it mouse or touchscreen. The full-sized layout proposal, with the description of each card is shown in Figure 38. It offers a lot of options while still being user friendly and simple. The choice to make it colorful and pretty come simply from the fact that it is much more valid to have a pleasant environment to work with in every aspect, and that extends to the interface.

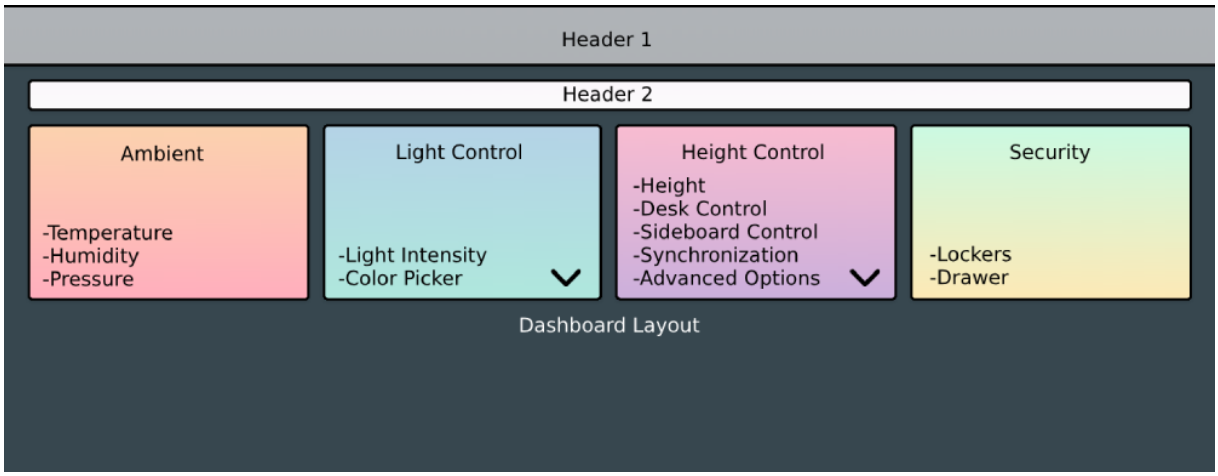


Figure 38 – Dashboard Layout.

Analyzing superficially, it is possible to see that there are two headers. Those are reserved spaces for possible applications such as menus and new functions the system might need in the future. In this project, they only contain the Fraunhofer logo and the title SmartDesk and Figure 38 is a section of the whole screen, just showing the cards and content. On the side of the functions, on the two middle cards, there is an arrow. This means that the card has a drop down menu, with more options, which is gonna be explained in specific subsection. In the following, the details about the functions of the dashboard and everything about its design is explained.

5.4.1 Functionality

Besides offering a pleasant view of the necessary information and a user-friendly interface, on the technical term, it has basically two roles. The first is to read the information sent by the Raspberry Pi to the database, which is the sensor data, and the second one

is sending the HTTP requests to control the height, the light and the security to the respective devices. For each one of these roles a different structure and language is needed, and that is explained in the next sections.

5.4.1.1 Database reading

The language used to read the database is PHP. The integration between the language and the MySQL database is native, thus not making it necessary any sort of library. The connection works fairly similarly to the one in Java, which consists in informing the database, username, password and address and then creating a connection with it. The difference, besides the language itself, is that in this case, since the database and the dashboard are on the server, the address is *localhost*.

As the goal is showing the last recorded piece of data and everything is done via query statements, after connecting to the database, a query is sent retrieving the last row of the table, as follows:

```
1 $sql = "SELECT id, light, height, hum, press, temp, time FROM deskdata
    ORDER BY id DESC LIMIT 1";
2 $result = mysqli_query($conn, $sql);
```

First of all, *$sql* is an arbitrary name for the variable that stores the query. What the query intends to do is to take the value from the fields mentioned (id, light, height, hum, press, temp, time) from table *deskdata* in the descendent order and stopping at the first row. In other words, it starts to read from the end of the table and just reads one row. Then, the variable *$result* stores what was returned from the function *mysql_query*, which is the value from each cell, linked to the field it came from.

After that, the data is separated in variables for each value and then it is filtered to the format it should have, in the sense of number of decimals, for example. The PHP code is then included in the index file. Since it loads every time the Dashboard is accessed, the page would need to be refreshed every time someone wanted new data. Since that is impractical and goes against the whole concept of the dashboard, a JavaScript was used to update just the fields where the variables are shown in a defined interval. For that each variable was saved in a separate PHP file and replaced content of an HTML field with a specific ID, as shown below:

```
1 setInterval(function(){
2   $('#temp').load('temp.php');
3   $('#humid').load('humid.php');
4   $('#pres').load('pres.php');
5   $('#light').load('light.php');
6   $('#height').load('height.php');
7   $('#time').load('time.php');
8 }, 3000);
```

The interval in this case is three seconds (3000 miliseconds), as shown above. Each ID(#) corresponds to a field in a specific card where the information is shown, and the files loaded have the *.php* extension. Using MySQL databases is fairly simple to retrieve the data. The details about how and where it is shown is found on the Subsection 5.4.2.

5.4.1.2 HTTP Requests

The HTTP requests are the way of sending information and commands to the other devices in the network. In this case, these commands are for controlling every controllable aspect of the workstation. There are many ways of sending HTTP requests. The chosen way was via JavaScript, because it allows an easier programming structure and do not explicitly open the URL.

There are actually two types of requests that are sent in this system. The first one is the static, that means that every time it is sent, the result is the same. One example is the ?UP request, which always makes the desk or sideboard go up. The other type is the dynamic one, that depends on a certain value set, and consists on sending the value, whatever it is, which is the case for the ?H=, which is the closed loop height control and the ?R= &G= &B= that controls the color of the light. The structure of a HTTP request in JavaScript can be seen below:

```
1 function commanddown()
2 {
3     var xmlHttp = new XMLHttpRequest();
4     xmlHttp.open( "PUT", "http://192.168.0.150/?DOWN", true ); // false for
      synchronous request
5     xmlHttp.send();
6     var xmlHttp2 = new XMLHttpRequest();
7     xmlHttp2.open( "PUT", "http://192.168.0.152/?DOWN", true );
8     xmlHttp2.send();
9 }
```

It creates a request when the function is activated trough a mouse or touch action, and then it sends it to the URL. The example above is a good one because it shows a function that sends two requests at the same time, to control both the desk and sideboard. The third value is set to true because otherwise it would expect the response to the request before sending the other one, and since the devices are not programmed to respond the requests, it would not work.

The requests are triggered by what are called Mouse Events. They are basically whatever the mouse does, click, hover, move and so on. The height control, for example, works in the following way: When the mouse clicks down on the UP arrow, it sends the ?UP request, and when the mouse releases it, it sends the ?STOP one, and the same thing for the DOWN button. Like shown in the excerpt of code below:

```
<div onmousedown="commanddown()" onmouseup="commandstop()" class="arrowdown
    ">
    <i class="material-icons">keyboard_arrow_down</i>
</div>
```

Another necessary feature for these controls to work in any device is the remapping of the Touch Events. These are any possible interaction between a finger and a touchscreen, such as touch, release and move or drag. The simplest solution is that, for each touch event, a equivalent virtual mouse event is created and the touch ignored, which, basically consists in replacing the touch with a correspondent mouse event. That was also done in JavaScript.

This subsection covered the basic functions of the dashboard and control system. The next subsection is responsible for indicating the placement of these controls and how it was made in order to be the most pleasant to the user.

5.4.2 Design

The design of the dashboard uses a newly added feature to the CSS language, the CSS Grid. Basically what it does is organize the content in a modifiable grid. The order of the content is then mandated by the CSS and not the HTML, which is a major advantage in designing responsive websites. Actually, the grid design is used both to make the organization of the page in responsive cards, as well as inside the cards.

Although CSS has a huge importance in allowing things to be pretty, in this case, it takes a step further, allowing the website to be responsive. Responsiveness is important for the user experience and for the sole purpose of the project, which is controlling and visualizing remotely, through any device. The way responsiveness was built in the page aligned with the CSS grid was by simply reorganizing the cards on the screen, which was done like this:

```
.grid{
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
    grid-gap: 1rem;
    grid-template-areas:
    "heading heading heading"
    ". menu ."
    ". ambsensor ."
    ". lightcontrol ."
    ". deskcontrol ."
    ". secbox .";
}
```

It basically defines that this style will be a grid, defines the column template, which means how many columns will there be and how will they divide their space in fractions.

Then the areas are defined, which is basically saying that a content identified with each of these names will take these specific areas when in this grid. Also, the dots mean empty spaces.

So it can change whenever the screen has a different size, it is necessary to change the grid with a @media tag, as shown on the excerpt below:

```
@media (max-width: 1020px) and (min-width: 630px) {
  .grid{
      grid-template-columns: 1fr 1fr 1fr 1fr;
      grid-template-areas:
      "heading heading heading heading"
      ". menu menu ."
      ". ambsensor lightcontrol ."
      ". deskcontrol secbox .";
 }}
```

As can be noticed, it changes when the screen follows the sizing described after the media tag. In this case, it is the layout for the tablet. It turns in to a four-column grid where the cards align in a 2-2 position. Those are the basic changes that give responsiveness to the page. In Figure 39, there is a comparison between the page in two different screens. It features the real design of the web page, with the intended colorfulness and inviting interface.

(a) Dashboard on Smartphone. (b) Dashboard on Tablet.

Figure 39 – Responsive web page in two different screens.

The CSS, as discussed, takes care of the style of the page. Actually, aside from the logos, no piece of style is image, but rather code, such as the gradients, shadows

and corners. Taking as an example, if wanted to build cards much like the ones in the dashboard, it would be necessary at least the code below.

```
.cardshape {
    border-radius: 10px;
    padding: 30px;
    padding-bottom: 0;
    box-shadow: 0px 2px 5px 0px rgba(0, 0, 0, 0.1);
}
.cardcolor {
    background-image: -ms-linear-gradient(90deg, #E0F7FA 0%, #F1F8E9 100%);
}
```

The first style is the card itself, the shape and shadow, and the second one is the color. That is done separately simply because every card has a different color but the same shape, so in order to not repeat information on the style sheet, this is a clever solution. Another cool feature that can be seen on Figure 39 and also on the next explanations, are the little icons on the side of the text and buttons. Those are also not images, but rather font packages, and can be re-sized without quality loss.

Regarding the content, there are two cards in this Dashboard that are fairly simple. They do not have drop down menus and neither have more than one function, just showing information or sending commands. Those are the Security and Ambient cards. On the other hand, the two middle ones (Light and Height Control) are more complex. Besides both showing information and also sending requests, they expand to show more options. On the next sections they will be detailed and seen on a closer look, starting from the simpler ones and ending on the more complex ones.

5.4.2.1 Ambient Card

One of the simplest one, its main focus is showing in the best way possible the environmental data. It basically shows 3 variables, the temperature, in celsius, the Humidity in percentage and the pressure in atm. The focus is on the most important of these variables, which is the temperature. It is what affects the most and is more perceivable, then it is justified for it to have a spotlight.

Everything about what is behind was already explained. It is just the information retrieved from the database organized in an HTML file, guided by the CSS file, as shown in Figure 40. Even though the page grid changes with the screen size, the grids inside the cards do not change.

5.4.2.2 Security Card

Basically the opposite of the Ambient card, it shows no data at all, but allows the user to control the lockers in the back of the sideboard and the drawer that stores the laptop

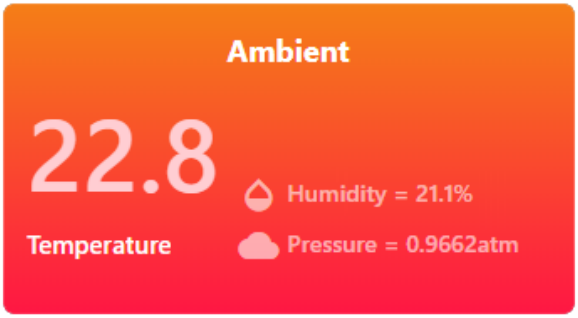


Figure 40 – Ambient Card.

in the desk. It has, then, three buttons, one for each function. The design of the card can be seen in Figure 41.

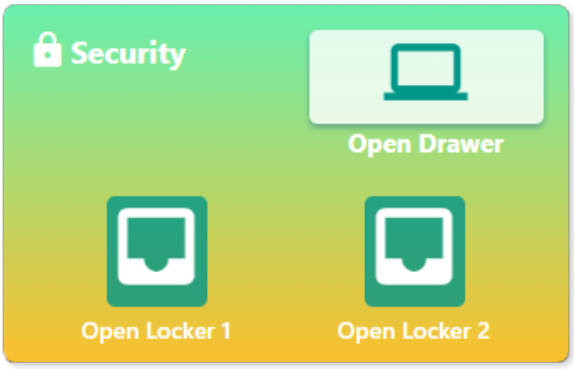


Figure 41 – Security Card.

As explained before, the buttons on the card send commands to the arduinos that control this on the desk and sideboard (addresses 150 and 152). For opening the drawer, the request is http://192.168.0.150/?OPEN and to open the both the lockers, being the first one, the one on the left and the second the one on the right, http://192.168.0.152/?OPEN1 and http://192.168.0.152/?OPEN2. In this case, the three commands are triggered by click. The next two cards are more complex in their functions and are going to be described next.

5.4.2.3 Height Control Card

This card controls the Height of the workstation. Though it may sound simple, there are lots of different functions that do it in different ways. The card itself shows the height of the desk, in centimeters, taken from the database, and has two buttons to control the height. Those buttons do that for the desk and the sideboard simultaneously, not depending on what point are they. The card has a simple design, as can be seen in Figure 42.

The buttons that control the height work on the following way: when the mouse presses down on any of the arrows, the desk and sideboard move in that way, and when

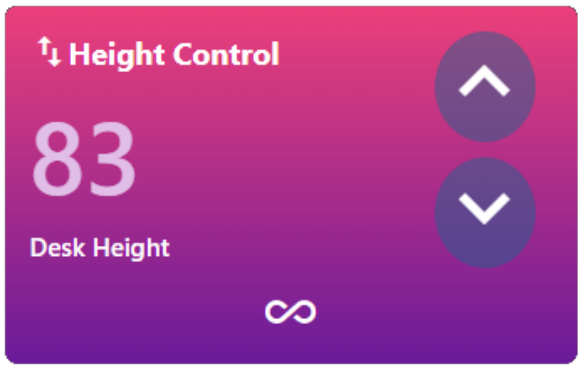


Figure 42 – Height Control Card.

it releases, it stops. That work with an application of the commands ?UP, ?DOWN and ?STOP.

On the bottom of the card, there is a "infinity" icon. It is actually a button that activates the advanced options for controlling the height. The card then expands down and shows other options, as seen in Figure 43

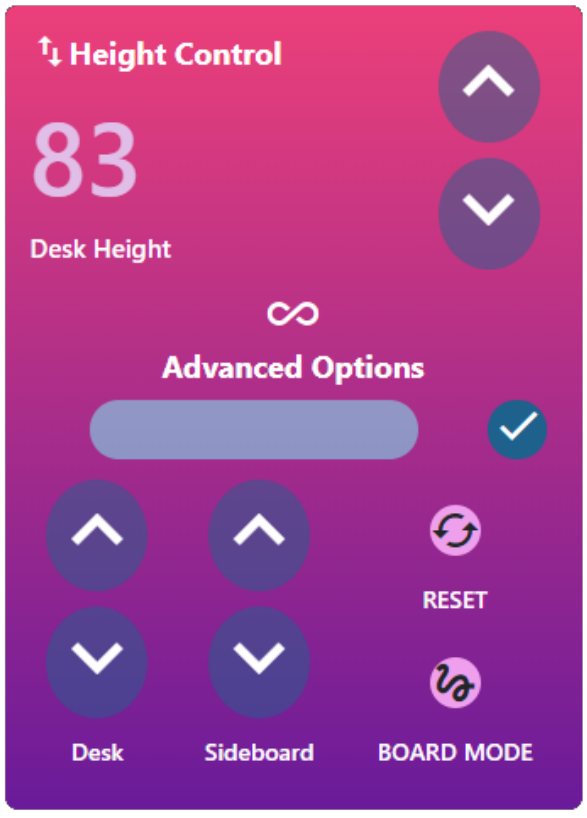


Figure 43 – Advanced Options of the Height Control Card.

The thing that draws more attention in the advanced option are the two height controllers side by side. As labeled below them and since the basic height controller works for the desk and sidebar together, these two are for controlling them separately. The

HTTP requests used are the same, but in this case they are sent to just one device in each controller. There are also two buttons on the side, the reset and board mode. Their function was already explained in Section 5.1, then, the buttons are put there because they are not meant to be used that frequently to be in the basic controllers.

The last piece part of the advanced options is the closed loop control. There is a text field and a button. This text field is for the user to put the desired height for the desk (it out of reach it is gonna be ignored) and then clicking to send the command. The request it sends was already explained on this section, but it is fairly simple. The job for processing it and in fact controlling it on closed loop is a responsibility of the desk's main controller.

It is valid to mention that in order to make the card drop down and expand to show the advanced options, besides the CSS, JavaScript is needed. What it does is, when the "infinity" symbol is clicked, it changes a CSS property that displays the content from "hidden" to "visible", and also animates the transition.

5.4.2.4 Light Control Card

The last card, then, is the Light Control Card. It looks, at first, very similar to the Height Control one. That is because it shows the light intensity on the same place the previous showed the height and has the same up and down buttons on the right, though they have a different function. The arrows on the right side, as shown in Figure 44 serve for changing the power of the lamp, as explained in the proper section.

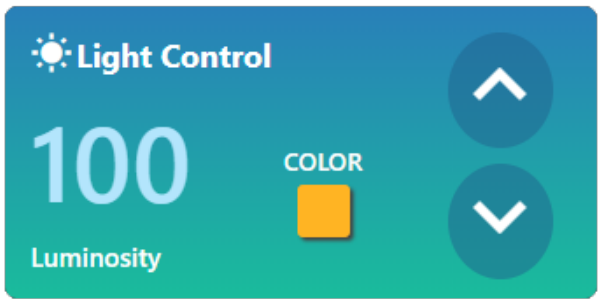


Figure 44 – Light Control Card.

In this case, actually, the most important control is being able to pick the color. For layout reasons, the color picker is not already opened in the basic setup of the block, but it appears when the color button is clicked. An interesting feature is that this button also shows the color that was picked for the lamp. When clicked, a huge color circle appears. Details are shown in Figure 45.

Much similar to the closed loop height field, this color picker works by giving the RGB values for the chose color and, when the button is clicked, it sends the information to the controller device. What happens is that, when the mouse hovers the image, a script

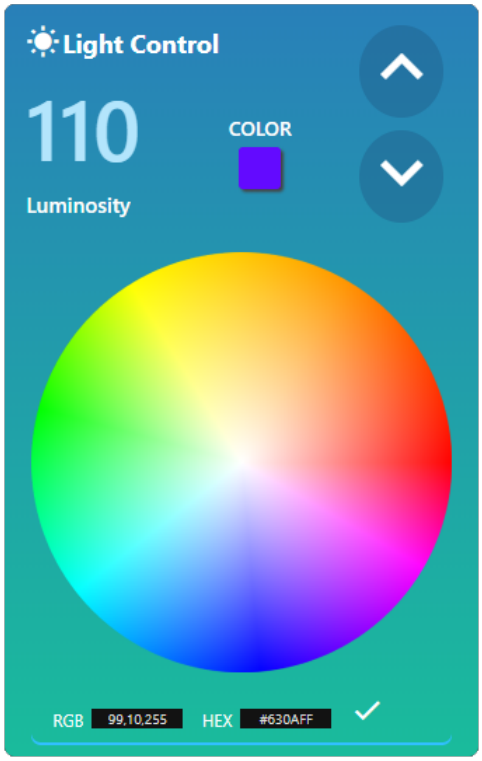


Figure 45 – Color Picker from the Light Control Card.

reads the color in the image and put it in the RGB field, that locks when clicked. That information is then sent to the arduino, that changes the color of the lamp.

Now, regarding the design, the only additions to what was already explained are the headers, the one with the Fraunhofer logo and the other one with SmartDesk written in it, shown in Figure 39. These four blocks combine, in a practical way, all the necessary information and control that a user needs to have in order to use the SmartDesk, so the interface turned out to be simple, pretty and powerful.

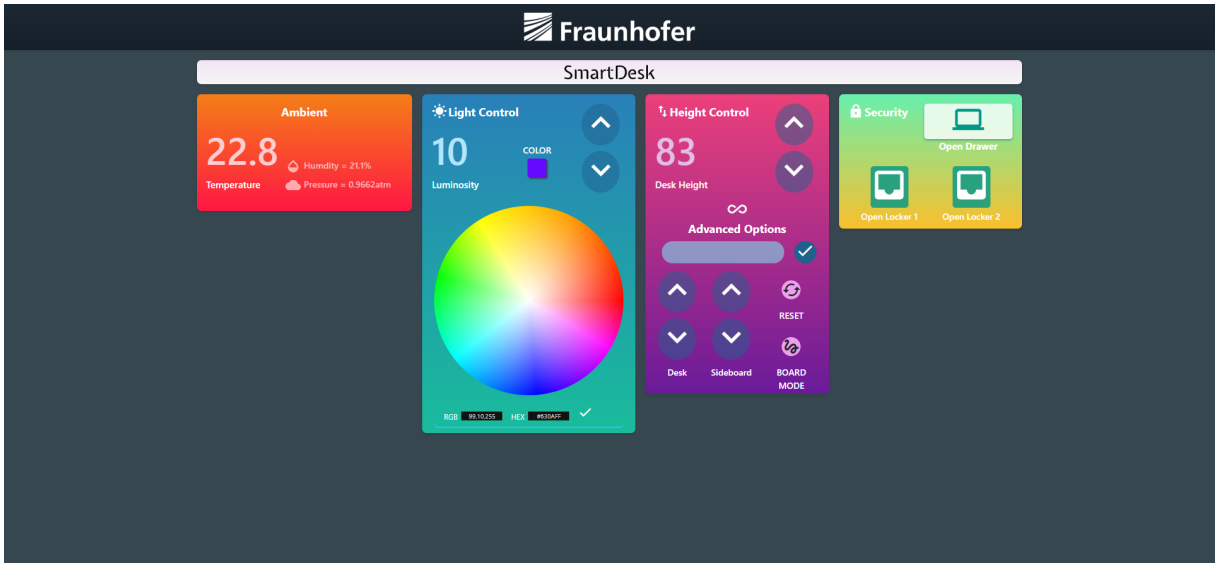


Figure 46 – Dashboard in big screens.

The whole design of the dashboard, at least for big screens, turned out almost identical to the layout shown in Figure 38. Figure 46 is a screenshot of the dashboard in use with the cards extended in a Full HD screen. That concludes the design of the dashboard for the purpose of the project.

5.5 Conclusion

This chapter consisted in explaining how every aspect of this project was built, coded and designed. From the relay on the Arduino to the color of a block on the dashboard, every relevant part was covered. The organization was different from the previous chapter, but that adds to a better understanding of the whole prototype. The overall functioning, the tests and the discussion about the success of the project will be found in Chapter 6.

6 Results

This chapter consists in showing the result of the project and evaluating its success through quantitative and qualitative measures. The results will be shown by first analyzing the functioning of each part of the project, if it works and, if not, why not. Then analyzing it as a whole, as a fully functional workstation, as it was intended. The methods will be discussed individually for each part.

After the analysis, a discussion is made, where the project itself is judged on its success or not, in a broader perspective than the fist part. Subsequently, the possibilities of other approaches or possibilities will be discussed. As it is in the prototype form, all the tests and pictures were taken in a test setup, even though it was being used as a workstation already. In Figure 47 is found the prototype in the test setup, actually being metalinguistically used to write this monography.

Figure 47 – Full functioning prototype.

6.1 Tests

The multi-disciplinarity of this project becomes a challenge when trying to evaluate it. The tests were chosen accordingly to the need for each part and just after that, for the whole prototype and in the most technical way possible. For example, testing the lamp is pretty straight forward, as it just has to work properly, but analyzing the satisfaction

of whomever is using the workstation takes time and method and none of them were available.

The decision was that, as the project is already backed by research in the well being of workers related to their workspace, it was assumed that, if matching the needs already discussed, the project would present itself as a good solution for that, and, then, using this cause-effect relation, if the qualitative and quantitative tests were to prove that the prototype is functional, it would match its requirements and consequently present itself as a good solution for the situation.

Following the subsections, each part is tested to evaluate its function. A "part" can be considered each independent section or point of interest in the project, soon to be pointed out when tested.

6.1.1 Lighting

When referring to the RGB lamp, there are two different ways of controlling it, even though they all resume effectively to the colors. The user can control the lamp color and the intensity, which, as said, after all is controlling the intensity of all colors altogether. For that, two different tests were done in order to assure the functioning of the lamp. Those are a Individual color test and a intensity test. These two tests are very simple, in fact, as mentioned, the lamp is pretty straight forward because it resumes to simply showing the desired colors.

The individual color test is basically testing individually, all the primary color of the RGB scale - red, green and blue - to see if they show correctly and if not, where the problem is. Luckily, all of the three color show perfectly on the lamp, as well as the mixture of them. The result can be seen on Figure 48, which contain all the 3 colors and one of the random color tested.

Figure 48 – Color test for RGB lamp.

Even though the pictures may not represent fully the real color being presented by the lamp, it's more than obvious that the lamp can show any of the primary color individually

and, as seen in Figure 48 and previous ones (Figures 47 and 33), it can also mix them.

For the intensity test, the process was also fairly simple, picking a random color and lowering its power. The values used were 50%, 30% and 10%. No bigger percentages were used (except 100%) because the difference would not be perceived by the camera, but the system, as mentioned works in intervals of 10%. The result is seen in Figure 49.

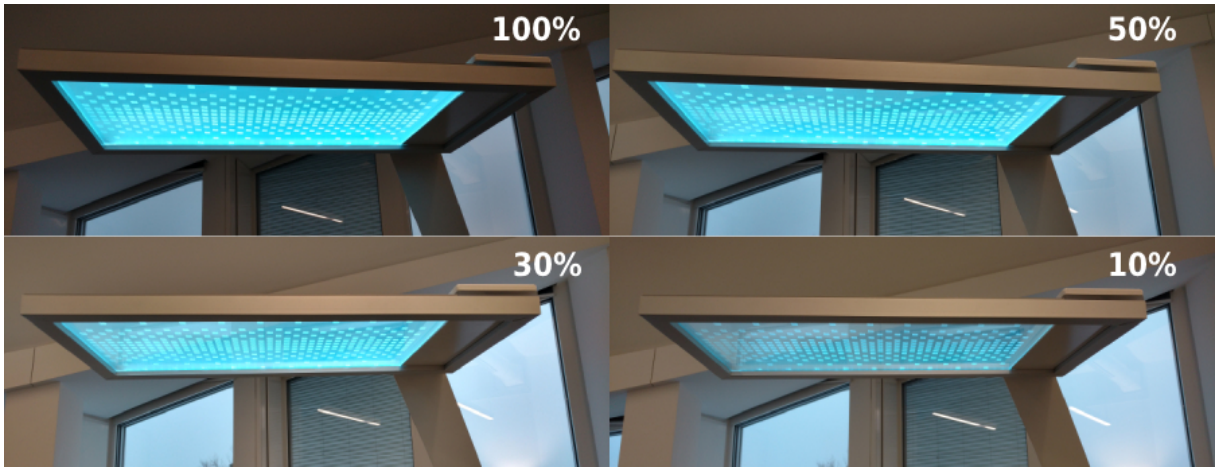


Figure 49 – Intensity test for RGB lamp.

It is valid to notice, that, in the 100% picture, the darkness of the rest of the environment is due to the lamp itself, sadly, it gets in the way of having a good digital representation of the real intensity of the light. Another thing, is that, even though it was not the case for this picture, there are possible distortions to the color when lowering down the power. For example, if a color has a value of 255, 10% of it would be 25.5, which is not reachable with a 256bits resolution. The solution is always approximating to the closest value, and other than that there is nothing to be done. The bright side is that it would hardly mean a noticeable distortion.

As a conclusion for this tests, is possible to see that the lamp is completely functional. That, aligned with the user's control of it is one successful result of this project. The other tests are presented in the next subsections.

6.1.2 Main Control

The main control refers to everything that is part of the structure itself, which are the the height control, the drawer and lockers. Basically, the movable parts. The height control is the sole thing that worked since day one. The only thing necessary for it to work (independent on the controlling device itself) was the module. This part, then, consists on basically giving an overview of the height control and the workstation's modes of operation and, after that, doing tests for the closed loop. These tests consisted in basically determining the best threshold for the control, which is going to be discussed at the end of this part.

Regarding what was built for controlling the height, the desk and the sideboard worked perfectly from the beginning. One problem found is that the desk can loose its wireless connection because of it's position, but that does not happen in the distance the desk is designed to be related to the sideboard. Another problem already fixed is explained on Chapter 5, which is that, for controlling the desk and sideboard simultaneously, the requests must be asynchronous, otherwise just one of the parts will move. Aside from that, everything worked and every work mode was possible to reach: the sitting, standing and board modes. As the sitting mode can already be seen in Figure 47, Figure 50 shows the desk in standing mode.

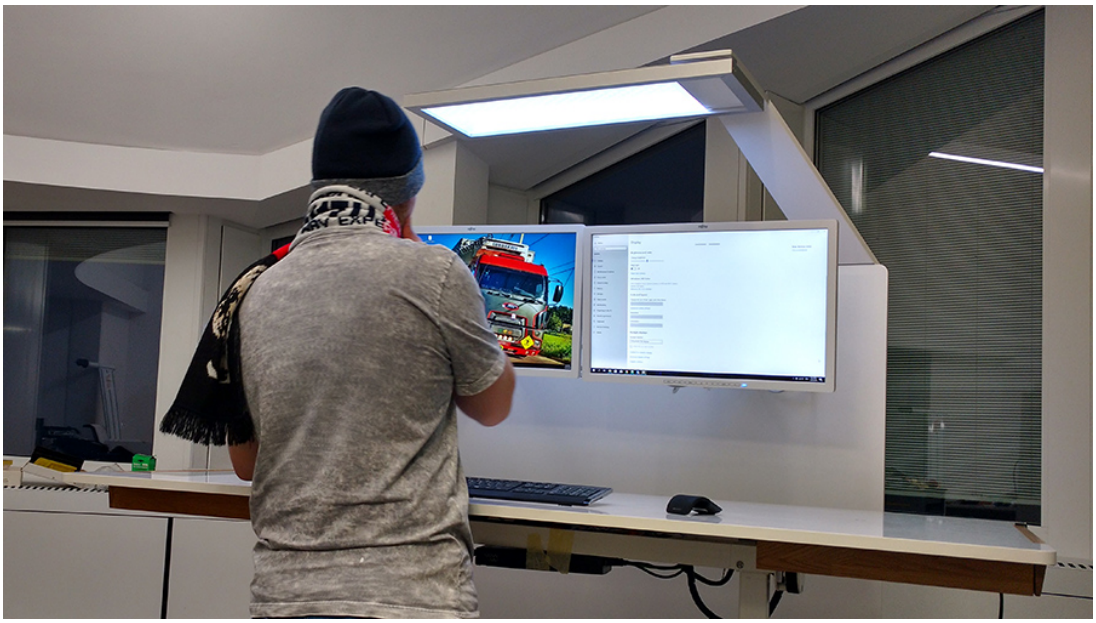


Figure 50 – SmartDesk being used in Standing Mode.

The desk height ranges from 70cm to 130cm approximately. The sideboard range was not measured. The reason for that is the inexistence of a specific point from where would be valid to measure from, such as the desk, that is measured from the bottom to the top. Though it was not measured, it is known that it is smaller than the desk's range. The fact is that, if the desk and sideboard are used synchronized, there is no point in going to the limit of the desks range, because few people are this tall to use it in a standing position this high, so, the only use for it is the board mode.

The board mode is applicable to the desk only, since there is no need for the sideboard to follow along. Figure 51 shows Mr. Dennis Stolze using the SmartDesk in board mode. In this specific example, the sideboard was, in fact, in a higher position, and, as can be seen, it exposes the wires in this height, because the bottom was design to be detachable.

The last situation about the height control, is that there is a delay between the control signal on the Arduino and the beginning of the desk's movement, and that is inherent to the module itself, not dependent on the request or the relays or anything of this sort. The existence of this delay means that the dynamic of the system is not instantaneous,

Figure 51 – Tests for the Board Mode.

in other words, if the command stop is sent, it is still going to take some time for the desk or sideboard to stop. In order to control the desk in closed loop, this delay has to be taken into account for the table to stop at the right place, in an acceptable range.

This acceptable range is considered to be 1 cm, or the reading error from the sensor, which was surprisingly not big, as seen further in this section. Through some tests, it was found that the optimal threshold is 1 centimeter away from the desired point, whether going up or down. When in closed loop, the sensor works very fast to secure that the measure is quite correct, so when the desk reaches the threshold for the desired point, it almost instantly sends the signal to stop.

There are actually two reasons for the acceptable range being considered 1 cm. First, because it does not represent a noticeable difference for the user and second, because the error on the reading from the sensor on the Raspberry, since has a reading interval bigger than one second, can get to the centimeter level, which was predicted when acquired.

For the lockers and drawer, the system also worked perfectly. In fact, the only problem with this part of the system is that the lockers unlock in the lack of power, contrary to the drawer, that needs to be powered up to unlock. In other words, it means that if the system is turned off, the lockers will open up. The open drawer can be seen in Figure 52. Also seen in the picture is a laptop inside it and an antenna symbol that indicates the position of the RFID reader.

Other important difference between the drawer and the lockers is that the drawer, if unlocked, needs to be manually opened and will also stay open if the power is off if not pushed inside, but if pushed, will lock again. The lockers have a spring that pushes them outside, so unlocking them means that the user will be able to open it for as long as he wants. Figure 53 shows the two lockers, both unlocked and one being opened. It

Figure 52 – Open drawer with laptop inside.

is interesting to notice the gap between the wooden frame and the top edge of the right locker, it means that it is unlocked, even though it is not being opened by the user.

Figure 53 – Lockers being opened.

Both RFID readers also worked with no problem. In this part, the solution was to just have a list of possible authorized users that could open all three compartments and the time it takes to run the list and find if the card matches any value is not even considerable, so the lockers and drawer open almost instantly. That was not a concern, but it would be a downside if happened. That concludes the main controller section.

6.1.3 Sensors

As known, there was a device (Raspberry Pi) designated for the sole function of gathering the data from the sensors and sending it to the database. It is important to analyze the success of it. Two points have to be addressed: the quality of the reading and the quality of the posting to database.

The posting of the data can be addressed first as it is quite simple. After putting some thought in what would be the necessary interval for the data to be updated, the decision was 4 seconds. The reason for that it that, from the five variables that were being read, three do not have a fast dynamic. Actually, humidity and temperature change very slowly and the pressure is almost static. From the two that can change their values faster, the height can already be controlled in closed loop and the light intensity is also controlled in ten different levels. That said, there is no need for less than that interval.

The code was presented in Chapter 5, and it showed effectively the delay between reading and posting being 4 seconds. Of course, the interval in posting is actually a bit bigger, because reading and posting take some time adding to these 4 seconds. The tests indicated that, with this interval, there was no computing nor network problem for the extended period of time (1+ hours) it was left working in many occasions. So, it can be indicated as a success.

In the realm of the quality of the reading, there was a worry about the height sensor having an error bigger than it should, for, as mentioned, being an affordable device. In fact, there were no problems about the temperature and pressure, as they indeed vary continuously up and down throughout small and big time intervals and the pressure would not have significant changes. The tests for the height sensor were the most basic as they can be. Leaving the desk static and reading the output for some periods of time. If the variation was acceptable, it would consist in a success.

Regarding what consists an acceptable range, it differs for every sensor, but what was basically decided for the height, is that it would have to stay static in the number format displayed in the dashboard. In Figure 42 it is possible to see that it displays only natural numbers. As that can be tricky and the real height can be close to a natural number itself, there might have changes in the displayed number without that much change in the reading itself, that is why the analysis was done based on the database. An example is found on Figure 54.

As can be seen, the reading actually crosses the interval a bit. That is not optimal but it was foreseen. It was chosen to highlight (grey) the pieces of data that cross the $\pm 0,5$ interval from the mean value (88,22) of the section, so it is possible to see that there is a difference between this and staying in the same numerical unit. These difference tends to be amplified during the usage of desk, due to shaking and bending. This part was not quite the success, but any alternative for the height reading would have a much

id	light	temp	hum	press	height	time
2003	10.845830387922113	23.13176626880304	23.234591111610136	966.8768591726299	98.4836941580756	2018-12-10 16:28:53
2004	11.105328598720465	23.13176626880304	23.256761612986775	966.850418694819	98.85060137457043	2018-12-10 16:28:59
2005	11.458774268843277	23.126713895634747	23.21813888051002	966.8685510237866	98.49264604810996	2018-12-10 16:29:03
2006	11.784438488160212	23.111556777876103	23.26854068102713	966.8968255866292	98.03089347079037	2018-12-10 16:29:07
2007	12.043873765732341	23.101452034158864	23.246722009577432	966.8802087098733	97.55658075601373	2018-12-10 16:29:11
2008	12.33117116891803	23.09134729160578	23.22490417186007	966.8900307496422	98.4291237113402	2018-12-10 16:29:15
2009	12.656764001978111	23.101452034158864	23.26334870938914	966.8273303187468	98.41926116838488	2018-12-10 16:29:19
2010	12.982278608621082	23.086294920765795	23.258332156568557	966.802406571291	97.57359106529209	2018-12-10 16:29:23
2011	13.335769026950842	23.086294920765795	23.258332156568557	966.9081607894674	97.9610824742268	2018-12-10 16:29:27
2012	13.727297237598279	23.07619017995894	23.258675816004256	966.8125485360209	98.45060137457044	2018-12-10 16:29:31
2013	14.08081099349869	23.08124255021685	23.214171277625766	966.8205370316392	98.90786941580755	2018-12-10 16:29:35
2014	14.467356442124013	23.066085440316236	23.281193249149606	966.8488079348448	98.06041237113402	2018-12-10 16:29:39
2015	14.886935398025514	23.066085440316236	23.286734995657746	966.822370240942	97.55211340206186	2018-12-10 16:29:43
2016	10.163167202343212	23.066085440316236	23.247942583093277	966.8488079348448	98.50338487972508	2018-12-10 16:29:47
2017	10.516822540490107	23.066085440316236	23.192524093923197	966.822370240942	97.51719931271478	2018-12-10 16:29:51
2018	11.035150251854532	23.061033070931444	23.19270002020735	966.8404995603711	98.84972508591065	2018-12-10 16:29:55
2019	11.520516653122218	23.05598070183769	23.198417698380357	966.7793167265704	97.97003436426115	2018-12-10 16:29:59
2020	12.034576022006224	23.05092833303497	23.265093227371725	966.7974458762906	98.44073883161512	2018-12-10 16:30:03

Figure 54 – Section from the information on the database.

bigger cost than this sensor and would not even guarantee the success. That said, the only effective downside is maybe seeing the height changing in one unit every few seconds in the database, for a possible data analysis in the future, it can be easily overseen.

6.1.4 Dashboard

As the control functions of the dashboard were already analyzed and also were the sensors, what is left to test is the usability of the dashboard in every device, that includes the responsiveness and the touch screen adaptability. There was no specific test routine for that, it just consisted in using the dashboard in different devices until finding any error. In Figure 55 it can be seen the dashboard in use on a computer.

Luckily, the dashboard was tested extensively during its programming, meaning that at each step it was usable, whether or not it had all the functions. This contributed to, in the end, not having any problems regarding its functioning. To illustrate that it can work in multiple devices, in Figure 56 it can be been being used in a smartphone. Obviously, since every worker has its own computer and every person has a smartphone nowadays, it is most likely that these two would be the kinds of devices most used for this purpose.

6.1.5 Conclusion

As pointed out, even though there were some minor things that did not work optimally, it was previously expected. As a full workstation there was no systematic test and measuring of subjective indexes, like well being, but all the normal day-to-day work was

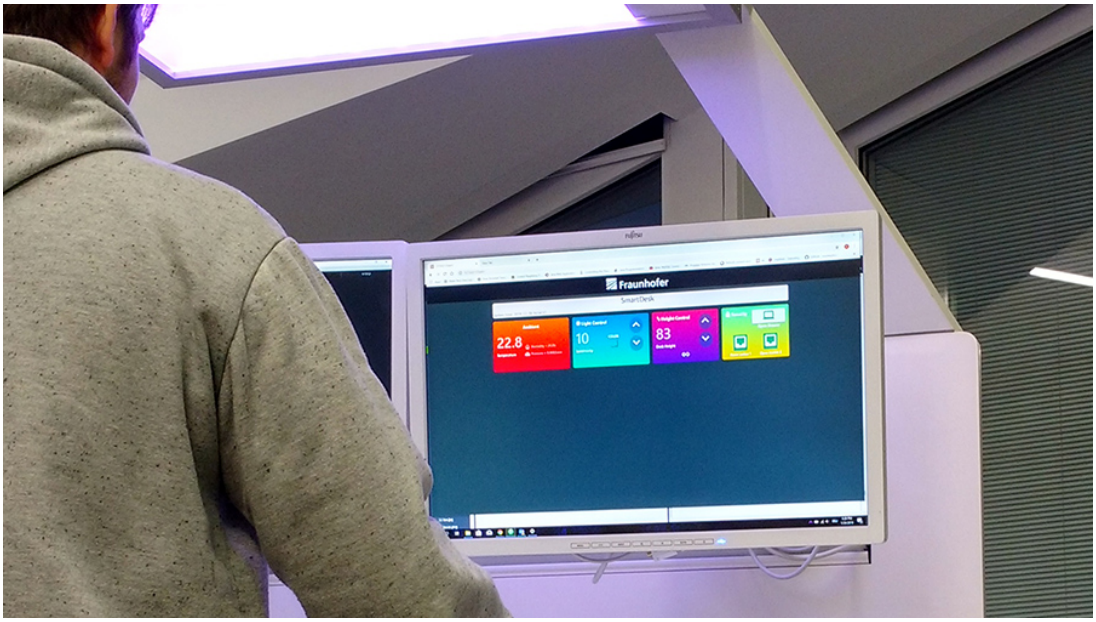


Figure 55 – Dashboard accessed via computer.

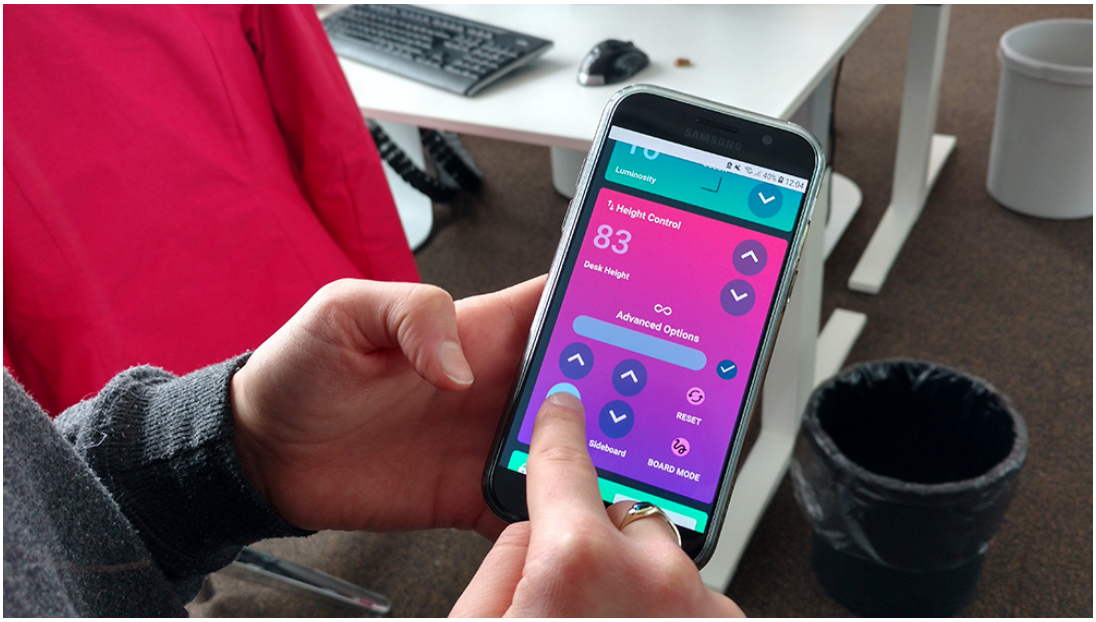


Figure 56 – Dashboard accessed via smartphone.

done in it, thus being a test. Actually, the desk was being used for testing ever since basic controls could be sent via the network, such as changing the color of the light even though it was not yet implemented on the dashboard. A photo of the SmartDesk being used before it was completely ready can be seen in Figure 57. It is possible to notice that the color picker was not implemented yet and instead there was a temporary switch in the dashboard.

Overall, everything worked as it should. In other words, from idealization, to the project and construction, everything followed the specifications and turned out as intended to: a smart workstation fully controllable and interactive. On the next section will be the

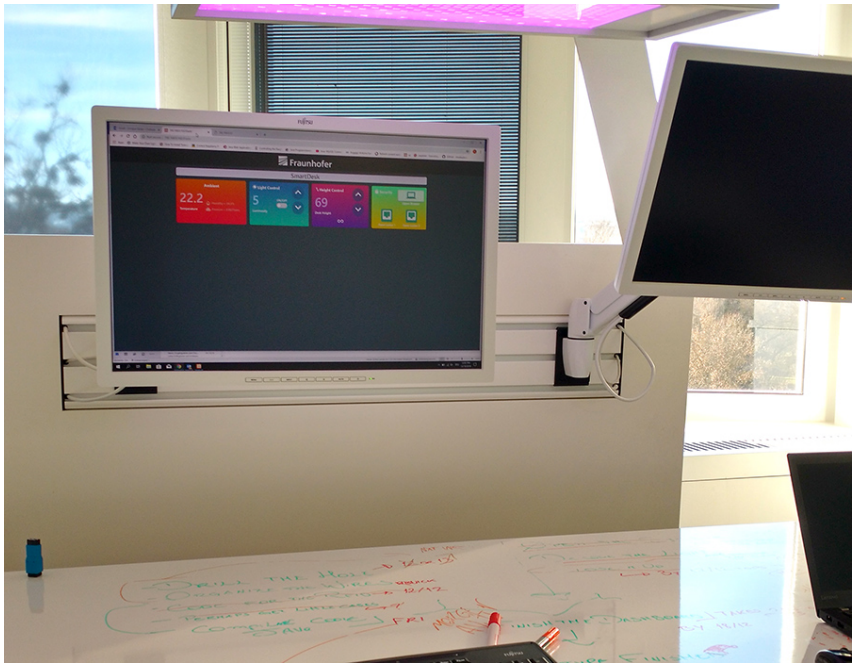


Figure 57 – SmartDesk being used during the build process.

discussion, where the success of the project will be evaluated as well as the possibilities for further improvement.

6.2 Discussion

This part consists in evaluating the success of the project. Does it solve the problem? Does it delivers what was proposed? What could have been done differently and what could be done further? These questions are put to discussion in this section.

From what was described so far in the results section, everything that was proposed for this project, is working and functional. The height control, the lamp, the dashboard and database and everything else that was, during this monography, described and explained. Hence, it validates the product SmartDesk as a functional and usable workstation. In other words, it "worked out".

It was mentioned in the beginning of this chapter, that as this project is backed on extensive research and designed as a viable solution for what was presented, if the prototype worked out as designed to, it would configure itself as successful for the purpose. That is in fact what happened. This project is able to provide a better tracking of the variables that influence the worker's well being, provide him more control of his work environment and also allow him to attain a healthier balance of different working positions. That is all explained in Chapter 3, and that is why this project can be considered successful.

Even though it is a successful project for itself, it is also a proof of concept for applications of IoT in the workspace. This means that, using this paradigm and following the line of what was built here, there are much more things that can be created. Specifically

for this project, there are many ways it could have been implemented, maybe using more or less controlling devices or even changing the way things were controlled and measured and still attaining the same or similar result, but all the choices that were made in this project are already justified in this work.

Using the IoT paradigm, so many things can be done, even as upgrades to what have been done already. A central login system would be an interesting implementation, even more if there was face recognition that automatically synchronized the user's preferences. Expanding on that, another application would be using concepts of Big Data to interpret user behavior and infer their preferences, creating a more personalized experience. That could be aligned with more sensors, for example, to measure gases in the air, and more actuators, for example motors, to change the angle of the monitors or move the desk around.

Those possible implementations are sure interesting and useful, but the existence of them, meaning that more can be done, does not exclude the importance of what has been done, on the contrary, it creates an evidence of the conceptual relevance of the project. Overall, and mainly considering the time it took, approximately 3 months from the idea to the full prototype, this project has succeeded in its goals.

7 Conclusion

This work was developed during an internship at Fraunhofer IAO, in Stuttgart, Germany and it had the goal of proposing a solution for this scenario of changes in the workplace, being adaptable and interactive, providing the user a more comfortable way of working and thus improving his well being. It consisted in a application of the Internet of Things as the technological paradigm responsible for these improvements.

The research indicated that there were many variables that could interfere, improve or worsen, the well being of workers in the workplace and consequently, their performance as well as some health benefits from changes in the work position. It is known that environmental variables affect the worker's well being. This variables are temperature, humidity and lighting, as well as the color of the environment. Also the role of self determination and being able to control your work and environment have a proven positive effect on motivation and dealing with stress, as well as for example, being able to change between sitting and standing to work can avoid health issues. All of that allied with the data about the satisfaction in the workspace brought up the necessity of improvement.

For that reason, the construction of what was called SmartDesk was proposed, a smart workstation, equipped with a set of sensors and actuators in order to act upon those researches and improve the experience of the worker. It was composed of a desk and a sideboard, which is the "wall" where the monitors stand. It allowed the user to read the environmental variables for him to track and attain the optimal environment, as well as controlling the light and its color and the height of the desk and sideboard. All of this through an interface over the network.

In this paper was described everything from the idea, to the design of it all and the construction. It was divided in different categories in each chapter, to cover all the possible aspects, but it came down to some controllers, that control the height and the security, one in for the desk and one for the sideboard, one that controls the light, and one that gets the data from the sensors and sends to the database. Another part was designing and coding the dashboard, that was the responsive web interface that allowed the data to be seen, taken from the database, and everything to be controlled.

It was defined that since the project took as basis all this research and purposed and justified the SmartDesk as a viable solution for the described situation, the prototype working out would, by consequence, configure the project as successful. After the results were shown, and all the analysis were done, it was established that, in fact, the project was successful not just as the product, but as a proof of concept for applications of IoT in the workspace and as an first step to bigger projects.

Concluding, it is necessary to mention the importance of this work for the graduation

as Control and Automation Engineer. Much of the knowledge used in this project was obtained during the course and thus making it possible for it to be made. There was also a lot of learning on the field that also contributed for the project, validating the experience of making this project alongside the internship.

References

1 IOT Technology Guidebook. Postscapes, 2018. Available at:<https://www.postscapes.com/internet-of-things-technologies/>.

2 KüLLER, R. et al. The impact of light and colour on psychological mood: A cross-cultural study of indoor work environments. *Ergonomics*, v. 49, p. 1496–507, 12 2006.

3 SEPPANEN, O.; FISK, W. J.; LEI, Q. Effect of temperature on task performance in office environment. 2006.

4 NOSQL vs. SQL – What is Better? Postscapes, 2018. Available at:<https://intellipaat.com/blog/nosql-vs-sql-what-is-better/>.

5 LOGICDATA. *Control Unit for an Electric Height-Adjustable Desk*. [S.l.], 2012. Rev. 0.

6 HEALTH Dashboard for Management. NanoHealth. Accessed: 2019-01-07. Available at:<https://d1upjxh66wqmf4.cloudfront.net/Website-Images/Corporate/Dashboard.png>.

7 AC750 Wireless Dual Band Gigabit Router. TP-Link, 2019. Accessed: 2019-01-20. Available at:<https://www.tp-link.com/pt/products/details/cat-9_Archer-C2.html>.

8 LENOVO'S USB 3.0 dual HD video port replicator. ExtremeTech, 2012. Accessed: 2019-01-20. Available at:<https://www.extremetech.com/deals/130533-et-deals-20-off-lenovos-usb-3-0-dual-hd-video-port-replicator>.

9 KELTER, J. *Wahrgenommene Lichtqualität im Büro - Phase 1: Auswertung Europa*. 2014. Available at:<https://www.zumtobel.com/PDB/Teaser/DE/Study_Office_Perceived_Lighting_Quality.pdf>.

10 SILVESTER, J.; KONSTANTINOU, E. Lighting, well-being and performance at work. *London: City University*, 2010.

11 TARIS, T. W.; SCHAUFELI, W. Individual well-being and performance at work: A conceptual and theoretical overview. Psychology Press, 2015.

12 SONNENTAG, S.; FRESE, M. Performance concepts and performance theory. *Psychological management of individual performance*, John Wiley & Sons, v. 1, 2003.

13 JURECIC, M.; RIEF, S.; STOLZE, D. *Office Analytics - Success Factors for Designing a Type-Based Work Environment*. [S.l.]: Fraunhofer IAO, 2018.

14 LEE, I.; LEE, K. The internet of things (iot): Applications, investments, and challenges for enterprises. *Business Horizons*, v. 58, n. 4, p. 431 – 440, 2015. ISSN 0007-6813. Available at:<http://www.sciencedirect.com/science/article/pii/S0007681315000373>.

15 SCHNEIDER, R. D. *Hadoop for dummies, special edition*. [S.l.]: John Wiley and Sons Inc, 2012.

16 GARTNER Says 4.9 Billion Connected "Things" Will Be in Use in 2015. *Gartner Says 4.9 Billion Connected "Things" Will Be in Use in 2015*, Gartner, Inc., Nov 2014. Available at:<https://www.gartner.com/newsroom/id/2905717>.

17 FOTACHE, M.; COGEAN, D. Nosql and sql databases for mobile applications. case study: Mongodb versus postgresql. *Informatica Economica*, v. 17, n. 2, 2013.

18 MINGES, K. E. et al. Classroom standing desks and sedentary behavior: A systematic review. *Pediatrics*, American Academy of Pediatrics, v. 137, n. 2, 2016. ISSN 0031-4005. Available at:<http://pediatrics.aappublications.org/content/137/2/e20153087>.

19 BUCKLEY, J. P. et al. The sedentary office: a growing case for change towards better health and productivity. expert statement commissioned by public health england and the active working community interest company. *Br J Sports Med*, BMJ Publishing Group Ltd and British Association of Sport and Exercise Medicine, p. bjsports–2015, 2015.

20 COMPACT - The Allrounder, a Product of LOGIC OFFICE LOGICDATA. <http://www.logicdata.net/blog/product/compact/>. Accessed: 2019-01-02.

21 LINEAR actuator electric height adjustable standing desk lifting columns. Hangzhou Lihi Eco-Tech. Accessed: 2019-01-02. Available at:<https://www.lihidesk.com/desk-frames/electric-standing-desk-lifting-columns/>.

22 ARDUINO - Introduction. Arduino. Accessed: 2019-01-03. Available at:<https://www.arduino.cc/en/Guide/Introduction>.

23 150MBPS Wireless N Nano Router - TP-Link. TP-Link. Accessed: 2019-01-04. Available at:<https://www.tp-link.com/en/products/details/TL-WR702N.html>.

24 ELATION. *DMX 101:A DMX 512 HANDBOOK*. [S.l.], 2008. Available at:<http://cdb.s3.amazonaws.com/ItemRelatedFiles/10191/dmx-101-handbook.pdf>.

25 TEXAS ADVANCED OPTOELECTRONIC SOLUTIONS INC. *TSL2560, TSL2561 LIGHT-TO-DIGITAL CONVERTER*. [S.l.], 2009. MARCH 2009. Available at:<https://www.tp-link.com/en/products/details/TL-WR702N.html>.

26 VALDEZ, J. B. J. *Understanding the I2C Bus*. [S.l.], 2015. Available at:<http://www.ti.com/lit/an/slva704/slva704.pdf>.

27 BOSCH SENSORTEC GMBH. *BME280: Final data sheet*. [S.l.], 2014. Revision 1.0. Available at:<https://www.embeddedadventures.com/datasheets/BME280.pdf>.

28 SCHMIDT, M. *Raspberry Pi: a quick-start guide*. [S.l.]: Pragmatic Bookshelf, 2014.

29 ULTIMATE Guide to Business Dashboards. Klipfolio. Accessed: 2019-01-07. Available at:<https://www.klipfolio.com/guide-to-business-dashboards>.

30 OVERVIEW of Web Programming Languages and Frameworks. Who Is Hosting This. Accessed: 2019-01-07. Available at:<https://www.whoishostingthis.com/compare/languages-and-frameworks/>.

31 APACHE - HTTP Server Project. Apache Foundation. Accessed: 2019-01-08. Available at:<https://httpd.apache.org/>.

32 XAMPP Apache + MariaDB + PHP + Perl. Apache Friends. Accessed: 2019-01-08. Available at:<https://www.apachefriends.org/index.html>.

33 LIEW, J. *Wiegand 4 bit, 8 bit, 26 bit, 32 bit and 34 bit library for Arduino*. 2018. Available at:<https://github.com/monkeyboard/Wiegand-Protocol-Library-for-Arduino>.

34 SINGH, Y. *BME280.java*. GitHub. Accessed: 2019-01-18. Available at:<https://github.com/ControlEverythingCommunity/BME280/blob/master/Java/BME280.java>.

35 MYSQL Connector J. MySQL, 2018. Accessed: 2019-01-18. Available at:<https://dev.mysql.com/downloads/connector/j/8.0.html>.

36 DMXMASTER. ThinkerKit, 2013. Accessed: 2019-01-20. Available at:<https://github.com/TinkerKit/DmxMaster>.