SLOVAK UNIVERSITY OF TECHNOLOGY
FACULTY OF CHEMICAL AND FOOD TECHNOLOGY

REGISTRATION NUMBER: FCHPT-5414-70732

DEVELOPMENT OF GRAPHICAL USER INTERFACE FOR INTELLIGENT ROOM

DIPLOMA THESIS

2017 **Bc. Jakub Jakabšic**

SLOVAK UNIVERSITY OF TECHNOLOGY
FACULTY OF CHEMICAL AND FOOD TECHNOLOGY

Registration number: FCHPT-5414-70732

DEVELOPMENT OF GRAPHICAL USER INTERFACE FOR INTELLIGENT ROOM

DIPLOMA THESIS

Study programme:	Automation and Information Engineering in Chemistry and Food Industry
Study field:	5.2.14. Automation
Workplace:	Institute of Information Engineering, Automation and Mathematics
Supervisor:	Ing. Martin Kalúz, PhD.

2017 **Bc. Jakub Jakabšic**

Slovak University of Technology in Bratislava
Institute of Information Engineering, Automation and Mathematics

Faculty of Chemical and Food Technology
Academic year: 2016/2017
Reg. No.: FCHPT-5414-70732

MASTER THESIS TOPIC

Student:	**Bc. Jakub Jakabšic**
Student's ID:	70732
Study programme:	Automation and Information Engineering in Chemistry and Food Industry
Study field:	5.2.14. Automation
Thesis supervisor:	Ing. Martin Kalúz, PhD.
Workplace:	Ústav informatizácie, automatizácie a matematiky

Topic: **Development of Graphical User Interface for Intelligent Room**

Language of thesis: English

Specification of Assignment:

This work deals with the design and development of Web-based user interface for control of an intelligent room, located at the Institute of Information Engineering, Automation, and Mathematics. The room is equipped with various types of sensors, actuators, and controllers, so climate in the room can be monitored and controlled.

The first stage of the work is devoted to studying of existing technical solution used in the room, as well as the participation on its further development. This includes the extensions of the sensor network, deployment of thermostatic actuator valves, and their integration into the control software of the room.

The second stage is dedicated to the development of the Web-based application that allows to visualize and measure the physical quantities with respect to a spatial arrangement of the room and a time of the measurement. The application also lets users control the heaters in the room.

Length of thesis: 50

Selected bibliography:

1. Margolis, M. *Arduino Cookbook: Recipes to Begin, Expand, and Enhance Your Projects.* Sebastopol: O´Reilly Media, 2013. 699 p. ISBN 978-1-449-31387-6.
2. Darie, C. – Brinzarea, B. – Chereches-Tosa, F. – Bucica, M. *AJAX a PHP tvoříme vysoce interaktívní webové aplikace: Profesionálně.* Brno : Zoner Press, 2006. 320 p. ISBN 80-86815-47-1.
3. Gilmore, W J. *Beginning PHP and MySQL: From Novice to Professional.* New York, USA: Apress, 2010. 824 p. ISBN 978-1-4302-3114-1.

Assignment procedure from:	13. 02. 2017
Date of thesis submission:	21. 05. 2017

Bc. Jakub Jakabšic
Student

prof. Ing. Miroslav Fikar, DrSc.
Head of department

prof. Ing. Miroslav Fikar, DrSc.
Study programme supervisor

Acknowledgement

I would like to express sincere gratitude to my supervisor, Ing. Martin Kalúz, PhD. for his guidance, help and overall positive attitude during the making of the project, and also for his help with finishing the thesis.

Jakub Jakabšic

Abstract

The thesis focuses on creating data acquistion architecture for a so called “Intelligent room” and the following creation of a Graphical User Interface (GUI) in the web environment for visualizing the physical quantities measured in the room. The thesis consists of five chapters. In the first two chapters, the thesis introduces the topic of Intelligent Environments and compares the produced solution with similar projects. Third chapter describes the used methods of data acquisition. The measuiring units are multiple types of sensors that measure temperature, air humidity and other quantities. The sensors are connected to a network of Arduino Yún microcontrollers, which contain an integrated computer with OpenWrt webserver. The data is saved into a MySQL database in periodic intervals, and it can be displayed through the interactive web visualization. The user interface is based on technologies such as PHP, SVG and JavaScript. Chapter no. four is devoted to describing all functions of the GUI for displaying the time course of measured information, and the basics of its source code. Fifth chapter briefly describes the possible expansions of the Intelligent room project, and its benefits towards research of automatic control.

Key words: Intelligent room; Arduino; MySQL; GUI; JavaScript

Abstrakt

Práca sa zaoberá vytvorením architektúry zberu dát v tzv. „Inteligentnej miestnosti“ a následne vytvorením grafického užívateľského rozhrania vo webovom prostredí na vizualizáciu nameraných údajov. Práca pozostáva z piatich kapitol. V úvodných dvoch kapitolách sa práca zaoberá témou inteligentných miestností a porovnáva vytvorené riešenie s podobnými projektami. Tretia kapitola opisuje využité metódy zberu dát. Meracími členmi sú rôzne druhy senzorov na meranie teploty, vlhkosti vzduchu a ďalších veličín. Senzory sú pripojené k sérii mikroovládačov typu Arduino Yún s integrovaným počítačom s webserverom OpenWrt. Dáta sa ukladajú v pravidelných intervaloch do MySQL databázy, a je ich možné zobraziť pomocou interaktívnej webovej vizualizácie. Užívateľské rozhranie využíva hlavne technológie PHP, SVG a JavaScript, a všetkým jeho funkciám na zobrazenie časového priebehu dát a zdrojovému kódu je venovaná štvrtá kapitola. Pomocou aplikácie je tiež možné priamo ovládať termostaty oboch radiátorov v miestnosti. V piatej kapiole sú stručne opísané možné budúce rozšírenia projektu Inteligentná miestnosť, a jeho prínos v oblasti výskumu automatického riadenia.

Kľúčové slová: Inteligentná miestnosť; Arduino; MySQL; GUI; JavaScript

Table of Contents

Table of Figures

List of Abbreviations

GUI	Graphical User Interface
IoT	Internet of Things
PHP	PHP: Hypertext Preprocessor
SQL	Structured Query Language
SVG	Scalable Vector Graphics
AJAX	Asynchronous JavaScript and XML
JSON	JavaScript Object Notation

Introduction

From the beginning of human history, people have developed more advanced technologies, and then used these technologies to their advantage and comfort. The importance of an individual's comfort inside their household, or the comfort of work environments, has been growing rapidly over the recent years. Starting by introducing electricity and water supply to homes, through central heating, air conditioning, automatic light settings, and finally home automation solutions, we've always put great effort into improving our environment.

The objective of the thesis is to create a user interface for an "Intelligent Room". In the context of objects, places, or software solutions, the term *Intelligent* doesn't mean sensible or conscious, but informs that the object in question is far from ordinary, and has the purpose to co-operate with humans towards a specific goal. In case of the Intelligent Room project, the goal is to conduct measurements of various quantities in the room, such as temperature and humidity, and then display the output in an interactive Graphical User Interface (GUI).

The biggest advantages of the GUI are its accessibility through the Internet, the transparency and simplicity of viewing information, but also the option to directly adjust the heating power of the radiators in the room. This one environment, in which we can control temperature actuators and then view the outcomes of the performed changes, might serve as an important tool for further research of air temperature control at the Institute.

In the thesis, we focus on explaining the importance and advantages of an Intelligent Environment. We then continue by comparing this Intelligent Room project with other similar projects, and introduce the term Internet of Things, which the project is closely related to. In the following chapters, we provide a list of all sensors that were installed in the room, and further describe the entire architecture of data acquisition, with not just the intention to clarify how it works, but also how it could be expanded in the future. We then proceed to describe the entire process of creating the GUI, focusing on details of every aspect of its source code. We conclude the thesis by analyzing the possibilities and outcomes of future research that could be conducted in the room.

1 Intelligent Room

There are many companies and products around the world, whose main objective is to optimize the cost and controllability of the features providing comfort in buildings. Starting from an adjustable thermostat, on which one can set the entire week‘s heating cycle, up to fully automated (autonomous) systems, which we call Intelligent Environments.

An Intelligent Environment is defined as a physical space with embedded systems, and information and communication technologies creating interactive spaces that bring computation into the physical world and enhance occupants’ experiences (Augusto C., 2013). However, whether or not we can call a room with some embedded systems devoted to enhance the comfort inside an Intelligent Environment is still ambiguous. As of yet, there are no international standards covering this topic.

The Intelligent room, whose creation, features, used technologies, and future usage are covered in the thesis, is located at the Institute of Information Engineering, Automation, and Mathematics (IAM) of the Faculty of Chemical and Food Technology of Slovak University of Technology in Bratislava. It serves as the Institute’s library and conference room, and has the seating capacity of approximately 20 people.

The reason why we are confident in calling the room an Intelligent Environment is that there have been installed many sensors and technologies for collecting information from these sensors, which give us a lot of data, based on which we can follow steps to change the room’s environment. Our objective was to obtain as much data as possible, store it in a database and then output this data in a transparent manner.

1.1 Optimal Room Environment

One of many reasons why Intelligent Environments exist is to enhance the comfort of people inside. However, it is quite difficult to define what are the optimal conditions (thermal conditions in particular), of an office or any work environment in general. It would appear reasonable to have the definition of optimal conditions based on some model of thermal comfort of a large number of people. Thermal comfort modeling has been the topic of studies during the last four decades. (Laftchiev E., Nikovski D., 2016). One of the most influential has been the model introduced by Dr. Povl Ole Fanger in 1967, which was adopted as an international standard (ISO 7730).

Studies have proven, that thermal conditions of an office environment have significant impact on work productivity, e.g. in a study by Hedge et al., 2005 in Laftchiev E., Nikovski D, it was found that lower temperature conditions of an office environment reduce the work output by as much as 46%.

By introducing technologies for both measurement and control of the room's temperature conditions, we are one step further to obtaining the optimal conditions inside the room, which would be based on a thermal conditions model, taking into account the temperature, humidity, number of people in the room, the freshness of air, air movement (e.g. by controlling fan rotation speed) etc.

The thesis focuses on building the first ground for an Intelligent Room: a structure of data-collecting nodes and sensors, which will be essential to any further studies regarding control of the room's actuators in order to reach the optimal conditions.

1.2 Data Acquisition

For any studies, which would need to analyze the course of temperature changes inside the room, or purely to satisfy the vast human curiosity, it is important to have as much data as possible. For an in-depth analysis, a single thermometer somewhere in the room is not nearly enough information to base assumptions on. It is important to know the source of heat-loss and its significance to the overall room temperature.

Based on these premises, we've decided to measure (among other quantities) the temperature of the inside layer of the room's walls. There is a total of 16 sensors installed throughout the room's walls, including the floor, ceiling and also insulation and outside facade of the building's outer walls, to measure the gradient of heat-loss through these walls. Here is a list of all measured quantities in the room:

- temperature and humidity of the air inside;
- temperature of the inner layer of five walls – one wall is separated by the entrance door;
- temperature of insulation and outside facade of the three outer walls;
- temperature of the floor and ceiling, measured twice in opposite corners of the room;
- air temperature near the two radiators;
- whether the entrance and the balcony door is opened or closed;
- number of people in the room.

Most parts of the data acquisition architecture were installed and functional previously – the Intelligent room existed and partially worked before our work on the project. As was mentioned before, our main objective was the creation of GUI for the measured data, although not all sensors that are currently working were functional originally. Many changes have been made to the original setup of data acquisition, including installing new cable endings on added sensors, and replacing the previous sensor connections to match the new ones. These new cable endings simplify the potential replacement of a faulty sensor, and also the addition of new ones.

More information about data acquisition can be found in Chapter 3.

1.3 Output of Data – GUI

After acquiring data from the sensors, the next step is to create an interface for browsing among this data. The User Interface shouldn't just be a set of tables that contain the measured values, but should provide interactive features for the user, so that they can browse, check, and compare various information as easily as possible.

It is important, that the GUI is easily accessible and that there is no need to install a specific software to gain the access. Furthermore, the interface for viewing measured data and also controlling the room's actuators, should be available on the Internet. Therefore, the created GUI is available as a PHP webpage, with interractive features of a client-side JavaScript code. Chapter 4 is dedicated purely to describing the features of the GUI, and explaining the programming background.

2 State of the Art

In this section, we focus on the current existing solutions of Inteligent Environments around the world, how they affect their occupants and compare their used technologies with the technologies used in this project.

The idea of a truly intelligent room, which would respond to verbal commands and understand and react to complex requests, such as “I’m feeling cold” or “It’s getting dark”, is not just part of science fiction anymore. The voice recognition technology has improved enormously during the past few years, and is actively used by many people in smartphone applications, to search for a specific phrase on the Internet or adjusting the phone’s settings on the user’s demand. The actions performed based on the request only concern the settings or commands of that specific device, whereas in an intelligent room, the processing application must control multiple actuators, and obtain data from various sources.

Already in 1998, on MIT in Massachusetts USA, a project with the name “Intelligent room” was created. Its purpose was to enable experiments with natural human-computer interactions during what is traditionally viewed as non-computational activity. (Coen M., 1998) More than on deterministic automatic control, it was focused on reseach of Artificial Intelligence (AI). The room included cameras, voice recognition systems and multiple monitors around the room, and had algorithms implemented to determine the direction where its current user is pointing. Their objective was that the computer system is constantly active in the room, activated by voice commands when required, or autonomously interacts with people according to their current activity, e.g. on the command “Computer, show me the weather forecast”, it would respond by displaying it on the monitor nearest to the user, or ideally in the future, when it would see a person lying down and closing their eyes, it will dim the lights and play some relaxing music.

Our “Intelligent room” project is not focused on human-computer interactions, but rather interactions between people and objects. The project is very closely tied with the term “Internet of Things” (IoT).

2.1 Internet of Things

IoT refers to a new direction, in which the Internet is expanding in the recent years – connection of *smart objects*. “Smart objects” is the term for everyday physical things, which are expanded by a small electronic device that is meant to provide local intelligence and connectivity to the cyberspace (Kopetz H., 2001). This small device refers to a computational component (e.g.

a microcontroller), which is attached to the physical thing and ultimately connects the physical world with the information world.

The novelty of the IoT is not in the functional capability of a smart object – there are already many devices (embedded systems) connected to the Internet – but in the expected size of billions of smart objects – a huge network of interconnected objects, which not only harvests information from the environment (sensors) and performs actions towards the physical world (commands, actuation, control), but also takes advantage of the current Internet standards to offer services for analytics, communication and information transfer. (Gubbi, J. et al., 2013). Figure 2.1 illustrates the idea of a complete and global IoT, which would influence all aspects of an individual's life, and also enhance the simplicity and comfort of the entire society.

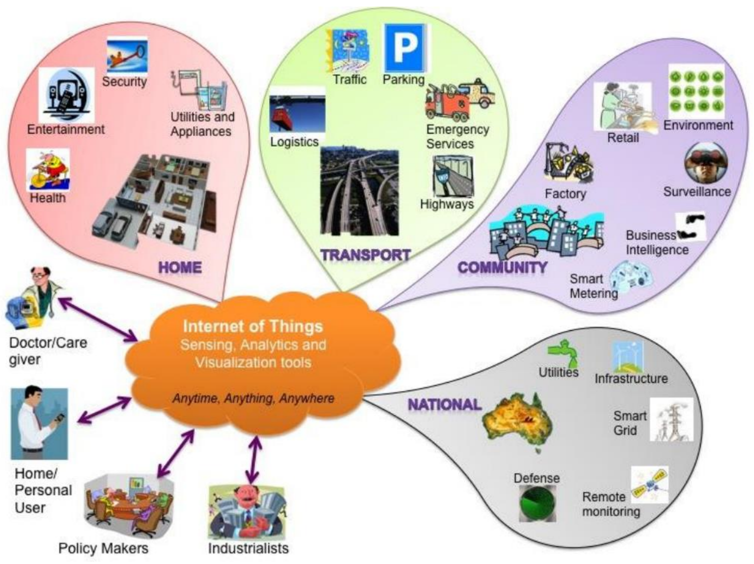


Figure 2.1 – Internet of Things schematic (Gubbi J. et al., 2013)

The IoT is an inevitable part of the human advancement in the future, as the number of smart objects is constantly growing, and the coverage and usability of wireless communication technology such as Bluetooth, Wi-Fi or telephonic data services are improved everyday.

A similar project to our Intelligent room also on Slovak University of Technology was conducted at the Faculty of Civil Engineering as a part of a larger project called “Systém monitorovania a riadenia vnútorného prostredia pre optimalizáciu spotreby energie”. (System to monitor and control indoor environment for energy consumption optimization). The final objective of this project was to ensure high comfort of the inner environment, while optimizing the energy consumption. Creation of the “Intelligent room” was the first step towards this goal. It measures data concerning both heating and cooling, ventilation and lighting. Sensors and actuators are connected to a BACnet IP module, and the application supports remote internet access, automatic control and alarm notification through email and SMS. The technologies have already been used for research of environment quality and intensity of ventilation in the school’s Library and Information centre (Hrčková L., Krajčík M., 2015).

As for non-research projects of Intelligent Environments, there are many companies around the world that develop and sell different kinds of home automation solutions, which are becoming more and more accessible to the general public, in terms of cost and complexity. Most of the solutions focus mainly on optimization of costs for heating and light sensitivity control, and most of them require a custom PC or smartphone application for adjusting the settings. However, for a single household, the price of these products usually exceeds the savings and advantages obtained by the solution, and it will take several years before such solutions become very common in homes. Still it is safe to say that Intelligent Environments are inevitably becoming an important part of human advancement.

3 Architecture of Data Acquisition

In this section, we describe the used technologies and architecture of obtaining information from the room, and how it is stored on the Internet for future visualization.

Before any information about the room's climate can be displayed for the user, we must first obtain the correct data from individual sensors, and save them into a database in a periodic interval. There are several steps that a value returned by a sensor must undergo before it is saved to its designated database column. The architecture and hierarchy of data acquisition is illustrated on Figure 3.1

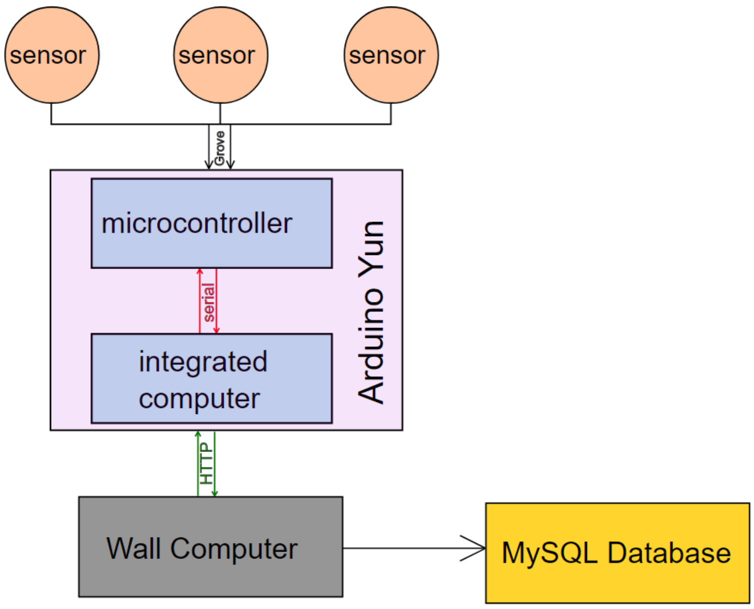


Figure 3.1 – Architecture of data acquisition

3.1 Sensors

Naturally, there are various types of sensors installed in the room, as they each serve their individual purpose. In this section, we provide a list of the various sensors used, their properties and their physical location in the room. Connecting sensors to their designated nodes has been a significant part of our work on the project, which required a respectable amount of time spent on cable work.

3.1.1 Digital Temperature Sensor DS18B20

This is the most used sensor in the entire room, as there are total 16 installed. They are installed in holes that were drilled into bricks forming the inside layer of the room's walls, as well as the floor and ceiling. Some of them are also installed inside the insulation and attached to the outside surface of certain walls.

The sensor consists of a water-proof probe with three wires forming just one cable with a Grove connector. The sensor is powered by either 3.0V or 5.5V and its wiring contains a pre-assembled pull-up resistor for digital signal conditioning. The sensor in this configuration can be directly connected to an Arduino board. It has a wide temperature range from -55°C to 125°C and accuracy of ±0.5°C. A picture of the DS18B20 sensor, which is mounted to the outside wall surface is displayed on Figure 3.2.

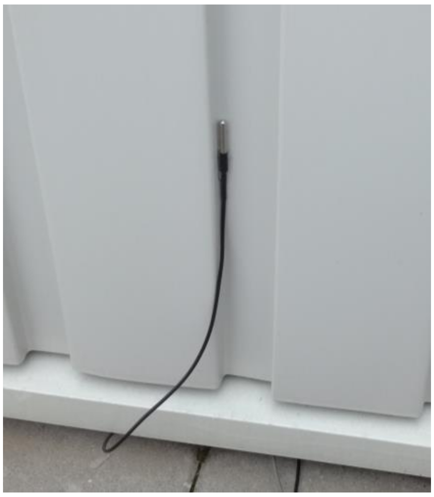


Figure 3.2 – Digital temperature sensor

3.1.2 Combined Air Temperature and Humidity Sensor

This sensor is used twice in the room, in parallel corners of the front wall, so that their distance is maximized subject to their connection to the Arduino nodes. It is connected via Grove port.

The sensor consists of a capacitive sensor element used for measuring relative humidity and a negative temperature coefficient (NTC) thermistor used for measuring temperature. The proposed accuracy of the measurements is ±0.3°C in temperature and ±2% in relative humidity.

The sensor is displayed on Figure 3.3 and is also visible connected to the data acquisition node on Figure 3.7.

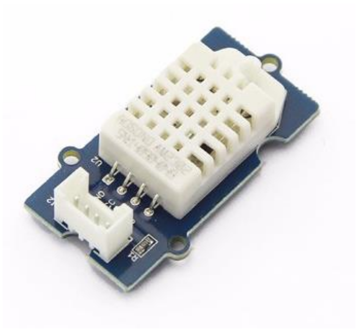


Figure 3.3 – Combined Air Temperature and Humidity Sensor

3.1.3 Photo Electric Counter (ER-SMS38491S)

This photoelectric counter uses a groove IR Optocoupler to detect if there is an obstacle located between the IR emiter and IR receiver. This sensor is installed twice, on both doors inside the room, to detect whether the door is opened or closed. We’ve crafted a small plastic obstruction, which was then attached to the door, while the sensor is located on the doorframe. When the door is closed, the plastic obstruction interrupts the IR signal and sensor reading provides a logic level 0. If the door is open, the sensor returns logic level 1. Figure 3.4 shows how the sensor is connected to the balcony door, in both open and closed state.

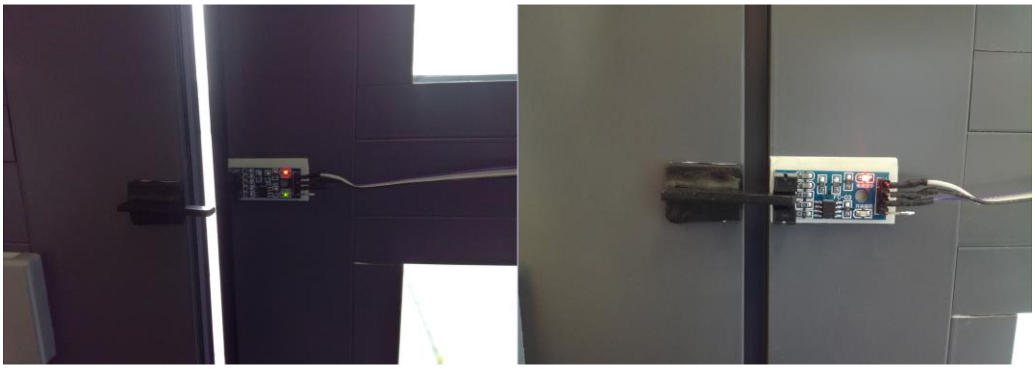


Figure 3.4 – Door opening sensor mechanism

3.1.4 Proximity Sensor (ER-SPGP2YIP)

This sensor is used to measure distance of objects between 10cm to 80cm from the sensor. There are total two sensors installed, which together serve the purpose of counting the number of people that are currently in the room.

The two sensors are mounted on the doorframe side by side, aiming to the space where people are expected to pass through. An obstacle is placed between the sensors to avoid a light of one sensor to be captured by the second one. Each sensor contains one IR emmiting diode and a position-sensible photo detector (PSD). The sensor works on triangulation principle, where the emmited beam reflects on the detected object and falls on the PSD, whose conductivity is based on where the beam landed. Output voltage is dependend on the conductivity, and the distance is calculated by an analog-digital converter.

The distance threshold is set up inside the microcontroller to capture any solid object (in this case a person) that passes through the door. The direction of person movement is determined by order in which the sensors are triggered. The mechanism is displayed on Figure 3.5.

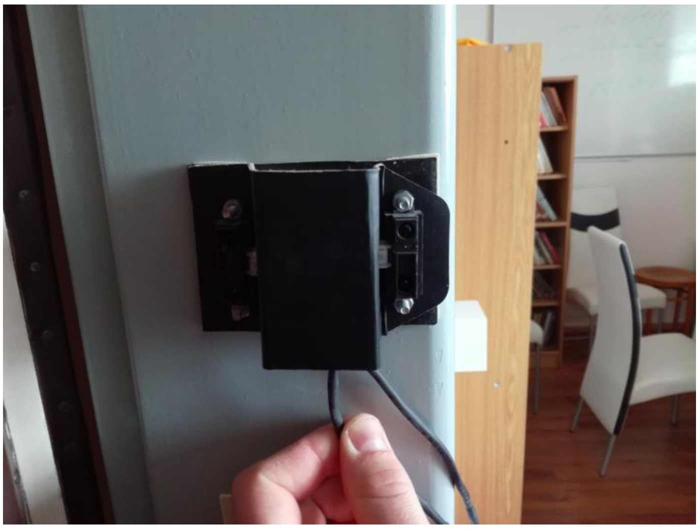


Figure 3.5 – Person counter mechanism

3.2 Microcontrollers – Arduino Yún

The sensors are connected to data acquisition nodes that are mounted on walls and spatially distributed inside the room. Each node consists of electronic development board Arduino Yún and Grove extension shield. Arduino Yún contains two independent processing units. First is an 8-bit Atmel ATMega32u4 microcontroller unit (MCU), which serves the purpose of controlling and capturing input-output electric signals, e.g. from connected sensors. The second is an integrated computer with Atheros AR9331 processing unit (CPU), which supports a Linux distribution based on OpenWrt named Linino OS.

To communicate with the computer, it is possible to either use the Linino‘s web interface, or a secure shell (SSH). Communication between the CPU and MCU is served via a UART serial interface (asynchronous serial communication), with the purpose of exchanging data.

The 8-bit microcontroller is programmed in the C language, through Arduino IDE environment. Arduino IDE, which stands for Integrated Development Environment, is an open-source program, available for download from the official Arduino site (*www.arduino.cc*), which is used for programming all types of Arduino boards. It has great options of importing additional libraries, and has a simple way of uploading a script to the specific type of Arduino board.

The integrated computer is basically a standard PC running on Linux OS, and may run various types of programs in different formats and languages. OpenWrt has a preinstalled Python interpreter, so we’ve decided to use a custom Python script to request data from sensors connected to the microcontroller. There are no other environments needed to be installed. Installing another computing environment could be beneficial in terms of performance, however, it might present a problem due to significant memory constraints of the integrated computer.

The board has built-in Ethernet interface, as well as a Wi-Fi module, both of which are supported by the Atheros AR9331 processor.

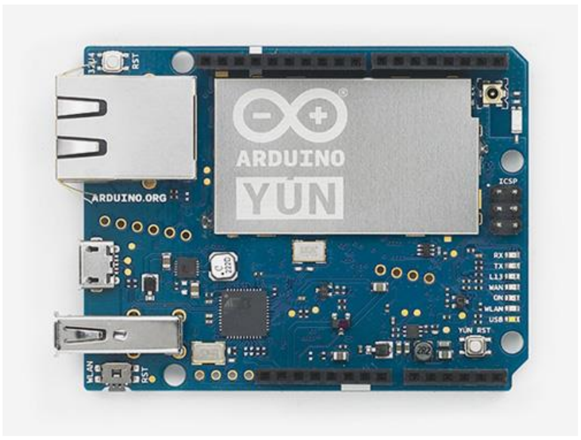


Figure 3.6 – Arduino Yún board

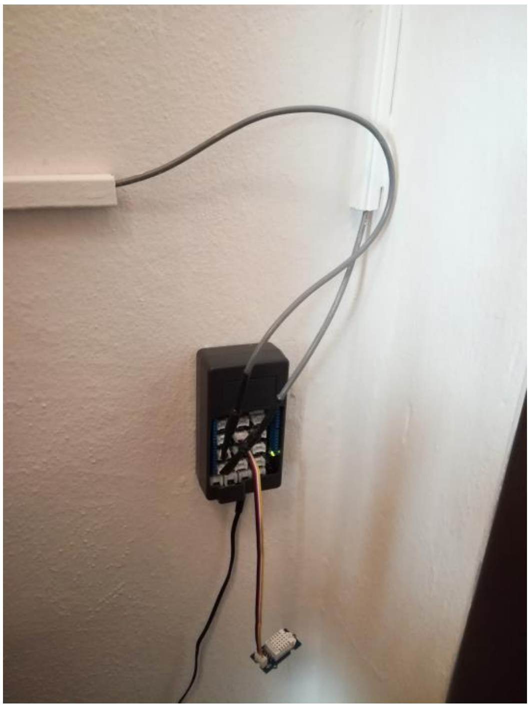


Figure 3.7 – Whole data acquisition node

3.3 Physical Location of Individual Units

In this section, we provide a more precise description of where individual nodes are located in the room, and which sensors are connected to them. Figure 3.8 shows the exact diagram of the Intelligent room architecture.

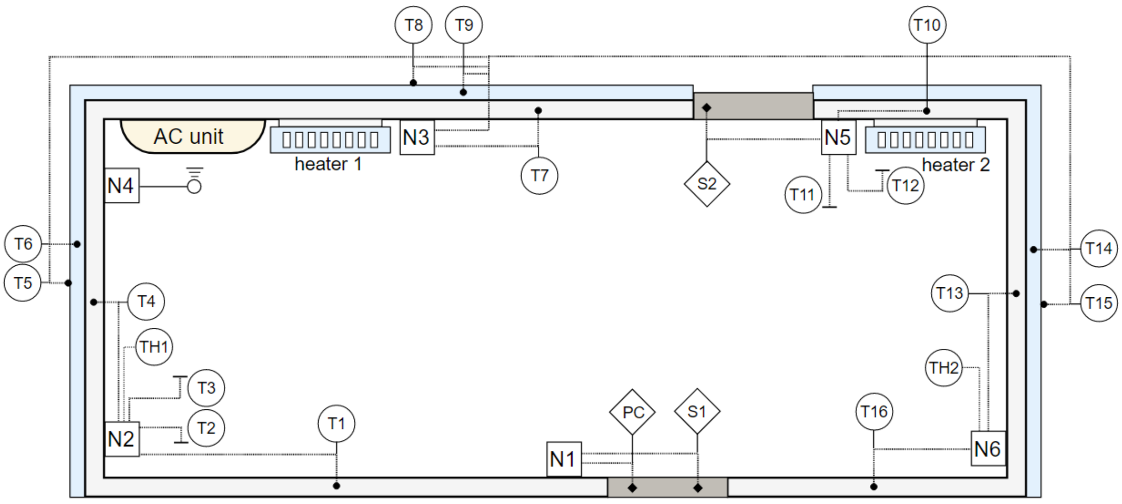


Figure 3.8 – Diagram of sensor and node location and connection

The diagram contains multiple types of symbols, which all follow the same pattern. “N” inside a square is the label for data acquisition and control nodes (mircrocontrollers), which are all mounted to the wall at the depicted locations. Sensors are marked by a circle or a diamond label, depending on the type of the sensor:

- T is the label for digital temperature sensor (16x)
- TH is the combined air temperature and humidity sensor (2x)
- S marks the sensor for door opening (2x)
- PC is the people counter (1x)

This diagram, as well as the so far unmentioned MS Excel table depicting the exact Grove ports number of individual connected sensors to the microcontrollers, are crucial for servicing a potential sensor or connection failure.

The blue outline of the diagram indicates that the actual room has three walls that don’t lead to another room, but are the outer walls of the building, and are naturally exposed to weather conditions. The second door (next to N5) leads onto the balcony, from where all the outer walls of the room are accessible for installing a temperature sensor onto them (T5, T8 and T15). It is also worth noting, that the room's ceiling is the floor of the above story's balcony, therefore it is an outer wall as well.

As the objective is to provide as much information about the room as possible, the fact that these four walls are the building's outer walls, might prove interesting when comparing their temperature with the remaining walls. It also opens opportunities for analysing the impact of weather conditions on the overall temperature in the room.

3.4 Communication Interface

Each computer of the Arduino Yún board is connected to the local computer network via wireless interface. The local network, along with the Wi-Fi, is managed by a router. Each node has its own fixed IP address within the local network. The nodes can be also accessed from wide area network (WAN) through the router's public address and specific port number.

As mentioned before, a Python script is constantly running on these computers, purpose of which is to await HTTP request in the correct form, which is a JSON array. When the HTTP request is received, and its value checked as the correct JSON form, the computer sends the request further to the Arduino board and awaits its response.

The response is based on the internal code of each Arduino board, as the number of connected sensors, and the overall response time may vary between the microcontrollers. On request, the sensors values are collected, mainly using Arduino IDE commands `analogRead()` and `digitalRead()`, or by a different command specified by the sensor's library. The response is, similarly to the request, in the JSON format, with the number of array elements matching the number of sensors connected to the microcontroller. These are designed to be recognized by the main PHP script, as it saves the returned data into the MySQL database accordingly.

3.5 Wall Computer

A PC has been placed on the wall near the room's entrance, which is essential to the whole process of data collecting. It is an industrial computer of the ECW-281B series, with 1.8GHz dual-core processor; 1GB RAM capacity, Ethernet and VGA monitor support. Currently, the computer runs on Windows OS, however, this might be changed to Linux in the future. It is displayed on Figure 3.9; next to it is the MAX! Cube device essential for controlling the heaters, which is further described in Chapter 4.3, and more to the left is the data acquisition node N1.

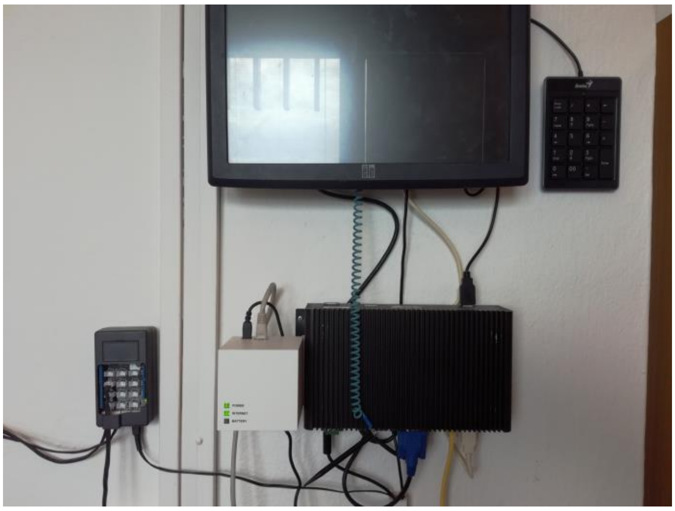


Figure 3.9 – Industrial computer attached to the wall

While this PC remains constantly active, there is an internal timer running, which runs a PHP script every specified number of minutes, which represents the sampling time of collecting data from the sensors. Currently, the period in which data is collected is set to 5 minutes, although this value might be changed to obtain data more often.

Purpose of the PHP script is to send an HTTP POST request onto the designated computers via CURL – native PHP library for communication through various web protocols, including HTTP and its certificates. By specifying the internal IP addresses of the Arduino computers, which are connected to the same network as the wall PC, we ensure that they receive the required request in the correct form, which is a JSON array in the format `{“getData” : 1}`.

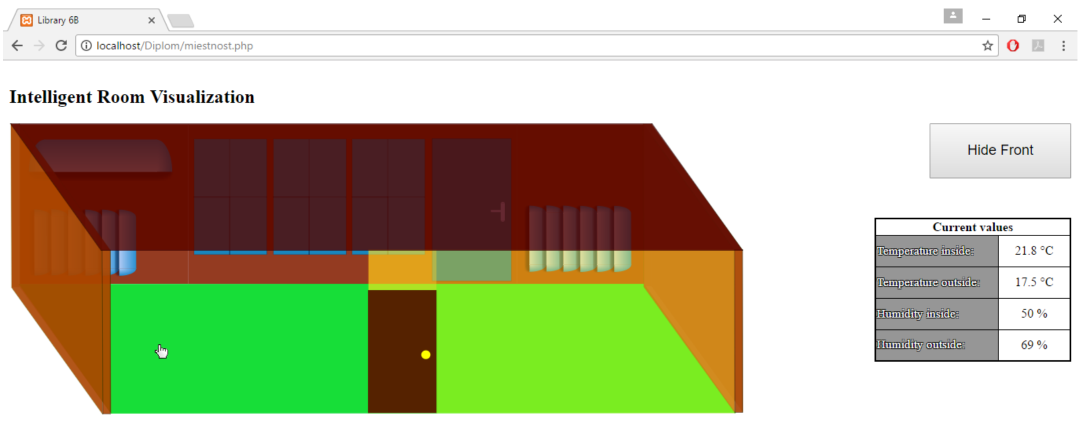
Each of the integrated computers is running a Web server (in Python), which receives the request and forwards it to the microcontroller through the serial link. The microcontroller then proceeds to read the request and collects data from the connected sensors. The obtained data is then transformed into a JSON array as well, the size and shape of which vary between the individual microcontrollers. The array is then transferred back to the Python script, which forwards it to the industrial PC as a response to the original HTTP request.

When data from all microcontrollers is obtained, the PHP script converts the received JSON format into a standard array object, and saves the value of its individual elements into MySQL database.

When data is stored inside the database, the next step of the application is to provide an efficient and attractive way to display it. The following chapter is devoted to describing both the frontend and backend of the user interface for browsing all collected data.

4 User Interface

As the objective of the thesis is to visualize information about the room, the most vital part is to create a graphical user interface (GUI) for displaying and control of individual features. It is required that the GUI is accessible through web and supported on various browsers.


Library 6B
localhost/Diplom/miestnost.php
Intelligent Room Visualization
Hide Front

Current values	
Temperature inside:	21.8 °C
Temperature outside:	17.5 °C
Humidity inside:	50 %
Humidity outside:	69 %

Figure 4.1 – Complete GUI

4.1 Application Structure

The GUI is built as a web application, whose structure consists of multiple files in different programming languages. The main page, which appears in the URL address of the user's browser, is in the PHP format. It consists of HTML elements, whose functional and dynamic properties are adjusted by JavaScript. However, the most important feature of the visualization is the main image (illustration) in the SVG format.

4.1.1 SVG Image

The first step of creating the user visualization was creation of the image representing the room in a vector graphic format. We've used the .svg format, which is a standardized form of the XML language. This means that the image is basically a text file, which consists of markup elements similar to HTML that can be assigned various attributes. It is standard for modern web browsers to be able to recognize and display an SVG file as image. However, the tree structure of the file, as well as attributes of the individual objects such as `class` or `id` are preserved by the browser, which allows the JavaScript code to access them and provide them with properties responsive to user interactions, such as hover or click, and also dynamically change their properties, e.g. change their fill color or stroke width.

To create the image, we've used the open source drawing software **Inkscape 0.92**, which allows the user to directly optimize the output SVG file for its use in web environment. It is also possible to directly access the source code of the image during the drawing process, which significantly simplifies the assignment of the `class` and `id` attributes to individual objects.

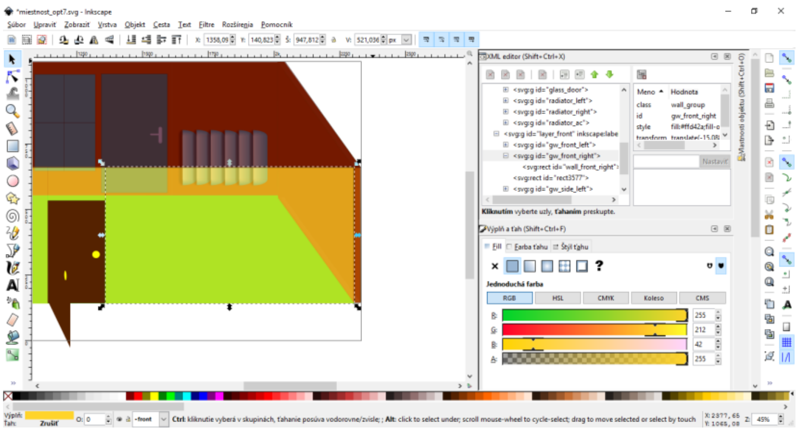

*miestnost_opt7.svg - Inkscape
Súbor Upraviť Zobraziť Vrstva Objekt Cesta Text Filtre Rozšírenia Pomocník
XML editor (Shift+Ctrl+X)
Výplň a ťah (Shift+Ctrl+F)

Figure 4.2 – Inkscape 0.92 drawing interface

Individual walls on the image are drawn using the `<rect>` element, which standardly represents a perpendicular rectangle. However, by adjusting the element's attributes, it is possible to rotate, turn and adjust the size of the element. The Inkscape environment makes the adjustment of these attributes very straightforward. In some cases, there are more objects used to represent one wall in the room.

All objects representing a single wall are put into a parent group element, to which we assign a common class `wall_group` and a unique `id`. Both of these attributes are used for interactions with the user's mouse. The rest of the objects forming the image are created in a similar way, and we choose an eligible color scheme to enhance the user experience.

To illustrate the entire room, we've elected to use a 3-dimensional projection from the right side, which can be seen in whole on Figure 4.1, covering major part of the screen. In the image's source code, the objects are merged into two main groups (layers) – `front` and `back`, due to the fact that the objects representing the front walls and ceiling overlap the remaining objects, and therefore prevent mouse interactions with them. By implementing the layers, the user has an option to hide (and subsequently show) all of the objects overlapping the ones in the back by clicking on a designated button located to the right of the image.This functionality is secured by a JavaScript function, which reads the current value of the CSS attribute `display` of the front layer group element, and changes it to the opposite, i.e. from the value `block` to `none` and vice versa. The view of just the back section of the room is illustrated on Figure 4.3.

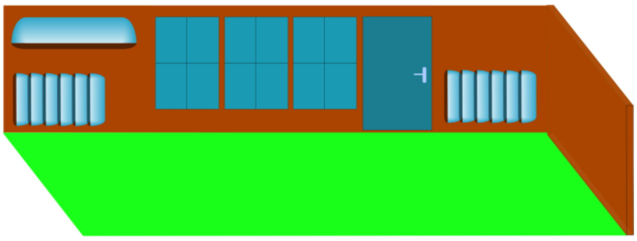


Figure 4.3 – Room illustration without the front section

4.1.2 Main Page – HTML and PHP

The source code of the main page is written in HTML, and contains and internal PHP script. Therefore, the URL of the website has .php suffix. The structure of the page is very simple, as it consists of only the heading and a few `<div>` elements, by which individual sections of the page are placed to their positions and sizes.

Content of the first block is filled by the SVG image's source code using the function `file_get_contents()` of the PHP language. The desired width and proportional height of the image are adjusted in CSS by changing properties of the `<svg>` element, i.e. the image's root element.

The remaining blocks that create the structure of the page form the right side of the interface – the toggling view button and the table, which displays the latest measured information about the room.

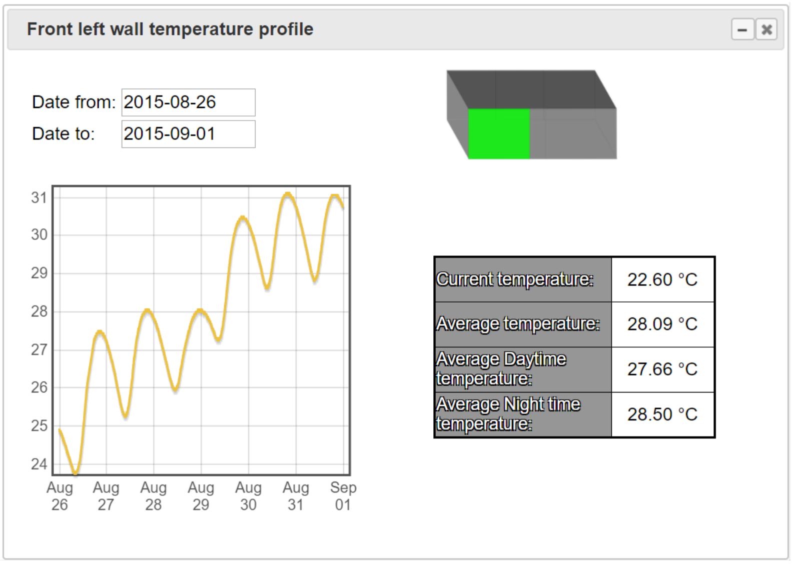
4.1.3 Source Code – JavaScript

The the source code of the application's interactive user environment is written in JavaScript. We mostly use functions of the jQuery library, with many of its extensions such as jQuery.ui and Flot.

The interaction of individual objects of the illustrations, such as walls and heaters, with the user's mouse is provided by selector for the joint class of all of the walls and heaters respectively, for which we call the jQuery functions `.mouseover()` and `.click()`.

After hovering over a certain wall, the original color and opacity of the SVG elements representing the wall is saved into global variables, along with the specific wall `id` parameter. The color is then changed to a defined value, which highlights the wall, and the cursor picture is changed to `pointer` with CSS. This encourages the user that clicking on the wall will trigger an event. After hovering over a different wall or moving the mouse from the image entirely, the color and opacity values are set to those saved in the global variables. This way, the image remains in the same color scheme, and only the hovered object is highlighted.

Clicking on a wall opens a modal window that shows a graph with the wall's temperature profile, along with a small table that shows the average values. The time interval, for which the values in the graph are displayed, can be changed by a calendar feature. The graph is made to support user interactions with the drawn data as much as possible. The source code and functions that create the modal window are further described in the following chapter (4.2). An example of the modal window that pops up after clicking on a wall is illustrated on Figure 4.4.


Front left wall temperature profile

Date from: 2015-08-26
Date to: 2015-09-01

Current temperature:	22.60 °C
Average temperature:	28.09 °C
Average Daytime temperature:	27.66 °C
Average Night time temperature:	28.50 °C

Figure 4.4 – Example of the displayed window for wall temperature visualization

4.1.4 MySQL Database

MySQL is an open-source database management system, which runs on the side of the server and uses standard SQL (*Structured Query Language*) commands. A database is a structured collection of data, in our case, of all measured variables in the room. The database currently consists of only one large table, where each row represents one measurement that includes the time of measurement in one column, and each measured value in its own corresponding column.

When displaying data from the databse, to distinguish which column represents which sensor, we use an MS Excel table that has been created and updated during the time the sensors were installed, and the source code of their connected Arduino boards adjusted.

4.2 Displaying Data from the Database

The main objective of the created visualization is to display collected data stored in the database, in a transparent manner to the user. In this chapter, we describe the methods that are used for this purpose.

Clicking on a wall on the illustration triggers a series of functions that create the modal window displaying all measured data of this specific wall. The unique identifier, by which the wall is distinguised from the others, is the `id` attribute of the SVG `<g>` (group) element representing the wall. The walls have various properties, which are important for us to know in order display all information about them. Based on just the `id` attribute, we already know the following:

- Whether there is just one, or three temperature sensors installed in this wall
- Name, or names of the corresponding columns in the database for this wall
- Description name of the wall, which appears in the modal window's title
- Filename of the small image, which illustrates the location of the wall

As these properties depend solely on the `id` of the wall element, and they are respectively required in different sections of the code, we've created a function which returns the currently prompted parameter value, based on the `id`. With these parameters accessible, we are ready to create the modal window.

4.2.1 Creation of Modal Window

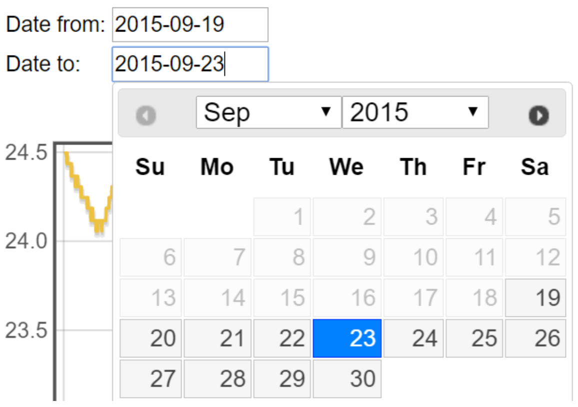
When creating the modal window, we start by creating a new `<div>` block on the page with *jQuery*, which has a dynamic `id` named `modal`. The contents of this block are filled by a 2x2 `<table>`, and we put an empty `<div>` block into each cell, which we are going to access later by their assigned `id`.

However, as the user can have multiple modal windows opened at the same time, these identifiers cannot be fixed, but must be unique in each modal window. For this purpose, a global variable called `figure_count` has been defined, whose (numeric) value increases with each click on a wall. Identifiers of the blocks inside individual table cells contain this number, and are therefore unique for the entire document.

4.2.2 Selecting the Date Range - jQuery DatePicker

The top left corner of the modal window contains a functionality, by which the user can select the date range, for which he wants the figure of the wall‘s temperature profile to be displayed. While searching for a handy tool for implementig such functionality, we’ve decided to use *DatePicker* – part of *jQuery.ui*, a freely available and scalable library for *jQuery*, which is mostly dedicated to enhance the user experience and simplify the web-development process.

As we need a range of dates for data display, the user must select both the starting and ending point of the desired interval. For layout purposes, we create a borderless table inside the top left cell of the parent element, which contains two `<input>` elements in separate rows, with a unique `id`. The `<input>` element is supported by the *DatePicker* library, therefore we may call the `.datepicker({options})` method to create a scalable *DatePicker* element. By clicking on such element, a small calendar for selecting a date pops up.



Su	Mo	Tu	We	Th	Fr	Sa
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

Figure 4.5 – jQuery DatePicker object

With the options parameter, we may adjust the object‘s properties, such as whether it is possible to change the month or year for the selected date, number of displayed adjacent months, and format of the date that is written in the input.

For these objects, we call the method `.on(“change”,{})`. At each change of the input elements, the function for drawing the figure of temperature profile – `PlotData()` – is called, which has many parameters including the starting and ending date. A function for displaying the average values in the given time interval and the most recent measured value, is called afterwards.

Changing a date in one of the inputs also affects the other one, as the result must be a valid date range. The *DatePicker* objects have a very convenient adjustable options – `minDate` and `maxDate`. By changing the “Date from” value, the `minDate` attribute of the “Date to” input is set to this date, and analogically for the opposite case. The effect is illustrated on Figure 4.5, where the invalid dates are blurred and unselectable.

With an interest to simplify the user experience with comparing the temperature values measured for different walls in the room, a change of date saves this date into a corresponding global variable. These two global variables are the default value for the *DatePicker* object when a new modal window is created. Therefore, if the user selects the desired time interval for browsing data, and opens a new window (by clicking on a different wall), the selected interval remains the same, and the user can easily compare the two temperature profiles. When all currently open windows are closed, values of the global variables are set to the previous default ones – yesterday’s and today’s date.

4.2.3 Displaying the Average Values Table

The function for creating the table that contains average temperature values within the selected time interval, and also the most recently measured value, is called when a new modal window is created, as well as at each change of the date interval as mentioned before. The parameters of the function, with which it’s called are: `id` of the wall object, dates from – to and `id` of the element, into which the output table is drawn.

Knowing the `id` of the wall element, we also know the name of its corresponding collumn in the database. Database data are accessed using the AJAX technology. AJAX stands for *Asynchronous JavaScript and XML*, and is called by the jQuery function `$.ajax()`, which asynchronously calls an external script with a specified URL – in out case a custom PHP script. It also sends specified data using the POST method, which are the necessary input data for the script – column name and the range of dates. The PHP language has a great synergy with the SQL language, or just tools and commands for overall work with a MySQL database. To obtain various average values within the time interval, we create a query on the database.

A query is just a custom string variable in the PHP script, but by calling the function `mysqli_query()`, it is translated into SQL and the desired data is extracted from the database. The query string is constructed with variables, therefore we obtain the desired data on each call of the PHP script based on the input parameters. Here’s the query used to obtain the average daytime temperature:

```
$sql = 'SELECT '. $column_name. ' AS "col",
EXTRACT(HOUR FROM time) AS "hrs" FROM sensors WHERE
(time > "'.$from.'" AND time < "'.$to.'") AND
(EXTRACT(HOUR FROM time) >=6 AND EXTRACT(HOUR FROM time) < 18)
ORDER BY time ASC ';
```

Using the function `mysqli_fetch_array()`, all of the measured temperature values between 6 a.m. and 6 p.m. of each day within the interval, are stored into an array. The single average value is then obtained by a simple loop through all of the array’s elements. The same procedure is then applied for obtaining the average night time temperature. The overall average is the mean value of these two.

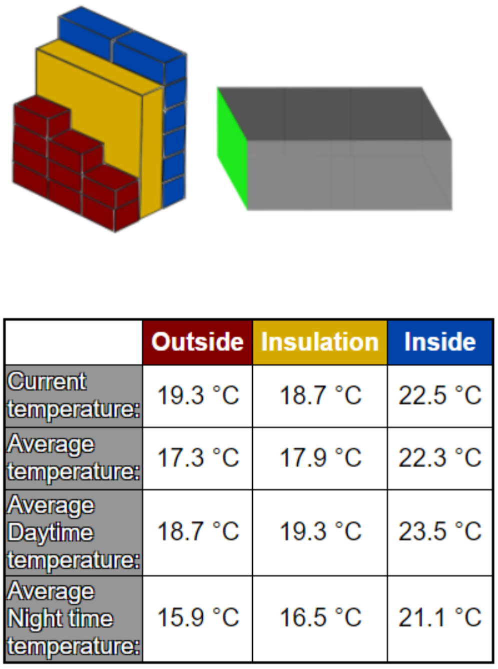
To obtain the most recently measured (current) temperature value, we use a similar query, but without the time condition, then order the output according to time, and we only take the last available value (condition `LIMIT 1` at the end of the SQL query).

The next step is to transfer the obtained results back to JavaScript, in order to display it in the table. We store all the desired results into an associative array, which also preserves the names of each value, in order to distinguish betwwen them. A PHP array object would be difficult to process in JavaScript, therefore we transform it into JSON format (JSON stands for *JavaScript Object Notation*), by calling the function `json_encode($result, JSON_FORCE_OBJECT).`

In the AJAX function, we can define the output type, so naturally we set it to JSON. The obtained output is then in the JavaScript object format, therefore we can access its individual elements associatively. All the extracted values are rounded to two floating points, and inserted into their designated cells of the Average values table.

4.2.3.1 Average Values for a Three-sensor Wall

In case the selected wall has three sensors installed, we use a modified function for creating the average values table, which has two additional parameters, and also calls a different PHP script. Such change is preffered due to the fact that we are requesting a set of three measured values instead of just one. To call the same script, which prompts the database and receives data three times in a row to obtain three average values, would be much less effective than creating a different script, which sends a query to the database to obtain values from three columns. All three average values are calculated within the same loop. The output format is also a JSON array, but with three times as many elements. An example of an average value table for a three-sensor wall is illustrated on Figure 4.6.



	Outside	Insulation	Inside
Current temperature:	19.3 °C	18.7 °C	22.5 °C
Average temperature:	17.3 °C	17.9 °C	22.3 °C
Average Daytime temperature:	18.7 °C	19.3 °C	23.5 °C
Average Night time temperature:	15.9 °C	16.5 °C	21.1 °C

Figure 4.6 – Average table for a three-sensor wall

4.2.4 Small Image in the Modal Window

When the user has opened multiple modal windows, it is in our best intentions to make the distinction between them as transparent as possible. Even though each window has its own title, we've put extra effort by placing a small image into the top right corner of the window, which illustrates the location of the selected wall.

The displayed image is a file in the .png format, a separate file for each clickable wall. These images were created by changing the color scheme of the original image in *Inkscape*, changing the overall size of the image in order to reduce the output file size, and saving them under their corresponding name directly in the .png format. The filename is directly linked to the `id` attribute of each wall, and is obtained during the process of creation of a new modal window via the function mentioned at the beginning of chapter 4.2.

For a wall with three sensors, the top right corner of the modal window is split into two columns, and contains an additional image. This image illustrates the meaning of why there are three columns in the average values table, as well as three lines in the graph. It represents a cross section of the individual layers of the wall – outside facade, insulation and inner layer – each containing a temperature sensor. The colors of the plotted lines, as well as headings of the average values table match the colors used in the image, which is possible as this image was also created using *Inkscape*, and so its individual elements and their colors are customizable.

4.2.5 Graph Plotting - Flot

The most important part of the modal window is the graph showing the temperature profile on the clicked wall in a selected time. For plotting the graph we are using the freely available jQuery library called **Flot** (www.flotcharts.org). The main purpose of this library and its methods is to plot graphs and support user interactions and customization options as much as possible. A great advantage of this library is its popularity among the users, who help create additional expansions, and there is always a vast documentation available for each usable functionality.

To use *Flot* on a page, the main document must have the main library, as well as all used addtional libraries, linked as a new script. These have been downloaded from the official *Flot* repository, which can be found on the official *Flot* website. The fundamental command for plotting a graph is:

```
$.plot( placeholder, data, options );
```

where **placeholder** is a fixed page element, where the graph should be drawn, which must have its width and height set to a certain value in CSS. It is recommended that this value should not be changed afterwards. The variable **data** denotes a two-dimensional array, which has many rows and two columns. Each row represents one data point, the first column depicts the x-axis coordinate and the second column the function value. The “data” parameter might also represent a structure variable, which contains several such arrays. This is used for plotting multiple lines into one graph, in our case for walls with multiple sensors. The last function parameter – **options** – is a structure variable in a standard format, by which the properties of the plot are customized, e.g. an option to zoom in and out of the plotted data, moving in the plot area by dragging the mouse in the direction of both x and y-axis (pan), properties of individual axis labels etc.

4.2.5.1 Obtaining Data for Plotting

The data plotted into the graph are obtained in a similar way as data for the average values table – by asynchronous request to an external script (AJAX). SQL query inside the external PHP script contains the corresponding column's name in the database for the selected wall, or names of three columns corresponding to a triple-sensor wall.

In this case however, we only use one script for both wall types, and distinguish between the cases based on a binary variable. This variable is obtained according to the selected wall's `id` attribute. The check whether a wall has one or three sensors installed is done twice, first in JavaScript before sending the AJAX request, which in case of a single-sensor wall will cause the remaining two column names to be left blank, as the script always receives values of three variables. The wall type is then checked again at the beginning of the PHP script, to verify whether the second and third received column names are relevant and to change the SQL query accordingly.

Besides the individual measured temperature values, we also prompt the database for the time when the data was measured. This information is then transformed into the `timestamp` format, which is an integer which represents, according to www.php.net, "a time measured in the number of seconds since the Unix Epoch (January 1 1970 00:00:00 GMT)". To transform the database value into the `timestamp` format, we use the function `strtotime()`.

Output of the external script is a two-dimensional array, where each row represents one measurement, so it either has two or four columns based on the wall type. To transfer the array back to JavaScript, it is encoded into the JSON format.

4.2.5.2 Editing Data into Correct Format

The next step after obtaining data from the database is to transform it from the JSON format into a standardized JavaScript array, which works for *Flot*, as mentioned previously – many rows and two columns, where each row is one datapoint. The time variable, stored in the first column of the array, which represents the plot x-axis, must be first transformed into JavaScript `timestamp`, which is different from the PHP `timestamp` format, as the time units are miliseconds in JavaScript, as opposed to seconds in PHP. Therefore, the integer value returned by the external script must be multiplied by 1000 before it is stored into the JavaScript array.

In the case of a three-sensor wall, this procedure remains the same, but the output are three arrays stored inside one structure variable.

4.2.5.3 Adjusting Graph Properties

Based on *Flot* documentation, we adjust individual properties of the graph by defining the options parameter of the plot function. The set options include:

- Changing the x-axis label, by denoting that the values of the x-axis are in the `timestamp` format, and specifying the timezone in which the data were measured.
- Zoom in and out from the graph by scrolling the mouse.
- Pan in the direction of both x and y-axis by pressing the mouse in the graph area and dragging it in the desired direction.

Sizes of how much does the user zoom in and out, and by how much do they pan the visible area are dependent on the amount of data points, or to be more exact, by the range of values on the x-axis. This value changes with each zoom into the graph, therefore so do the speed of zooming and paning, until they no longer change when the bottom limit has been reached. The top limit is the original size of the graph. The amount of labels and their text on the x-axis also change by zooming, from dates, to dates with time, up to just time in the 24-hour format. Double-clicking on any place in the graph calls the `PlotData()` function with the exact same parameters, which ultimately resets the zooming, while maintaining the same properties. These functionalities are included in the additional *Flot* library `flot.time`.

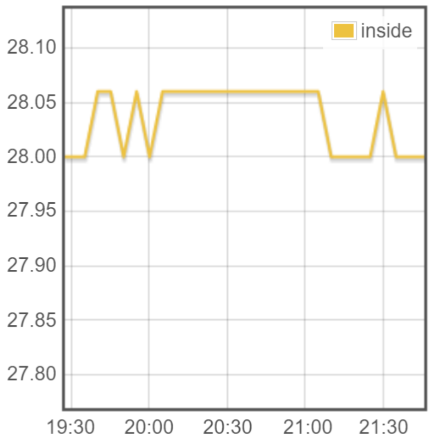


Figure 4.7 – Zoom in to the graph

4.2.5.4 Displaying a Single Datapoint

To enhance the transparency and simplicity of browsing the displayed data, the user has an option to hover over a plotted line in the graph, which makes a small banner pop up that contains information about the single measurement. The banner is created and positioned using the function

```
$(element).bind(“plothover”, function() );
```

which is included in the basic *Flot* library. It recognizes the position of the hovered data point according to the bottom left corner of the graph, and both x and y-values. We then create a new fixed element in the document denoted by a unique `id`, append it to the modal window element, and adjust its CSS position attributes according to the data point‘s position. An illustration of such banner is given on Figure 4.8.

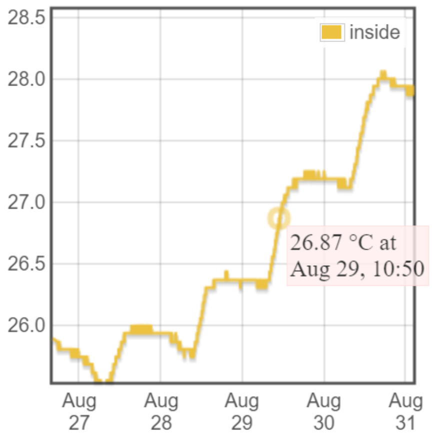
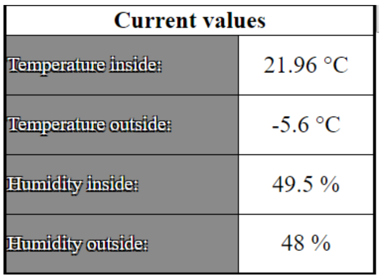


Figure 4.8 – Captioning a single data point

4.2.6 Current Values Table

As the objective is to visualize all available data about the room, and so far we’ve only covered the visualization of temperature profiles of walls, we now move to displaying the most recently measured information, such as current air temperature and humidity both inside and outside of the room, and whether the door leading to the hallway, or the door leading to the outside balcony, is open or closed.

The temperature values are showed inside a small table, located near the bottom right corner of the main image. The values are obtained in a similar fashion to the average values table for a wall – AJAX request to a designated PHP script. The request is sent on the event of the page being loaded, and the table is not further updated unless the user reloads the page. As there are two sensors that measure temperature and humidity inside the room, located in parallel corners of the room, the displayed value is the mean between the two acquired values.



Current values	
Temperature inside:	21.96 °C
Temperature outside:	-5.6 °C
Humidity inside:	49.5 %
Humidity outside:	48 %

Figure 4.9 – Current values table displayed on the main page

Information about the door being opened or closed is displayed directly on the main image, as the original SVG contains two objects representing each door either open or closed that overlap each other. After receiving information about each door being closed or open, one of the SVG elements is hidden using CSS. This value is also only updated every time the page is refreshed.

4.3 Controlling the Heaters

The web application serves not only as a way to view measured data, but it also allows the user to directly control the radiator thermostat units. The technology, which enables such functionalities, is the eQ-3 MAX! Cube LAN Gateway.

“MAX!” is the product name of a house solution by the brand eQ-3 for heating control, which we’ve decided to use for its cost and functional efficiency. The *MAX! Cube* (Figure 4.10) is the central element of the solution, which communicates wirelessly with other individual components – thermostats. The physical location of the MAX! Cube in the room is near the entrance, below the industrial PC on the wall. It is connected to the internet by Ethernet cable, and the USB power supply is connected to the wall PC.

Figure 4.10 – MAX! Cube device

The reason why the Cube is connected to the internet is that we’re using an artificial method of communication, called MAX! Home Automation, which was developed by Dmitry A. Kazakov (Kazakov, D. 2015), as opposed to the local MAX! software or the official smartphone application. The solution provides that while the Cube remains active and connected to the internet, it is possible to control individual thermostats, which are physically installed on the radiators (Figure 4.11), by HTTP GET requests. When internet connection is lost, or the Cube deactivated, the configuration of both thermostats remains the same.

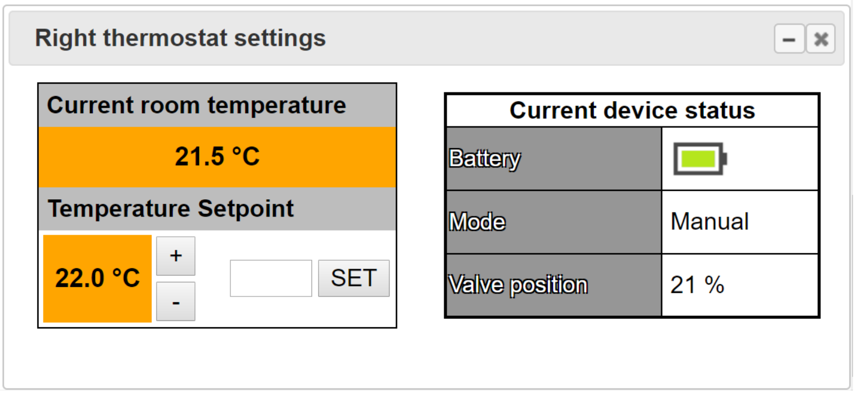
Figure 4.11 – MAX! Radiator thermostat

The HTTP request must be in a standard form, which specifies IP address of the integrated HTTP server of the application for all connected cubes, the RF (hexadecimal) address of the specific cube, as well as the RF address of the thermostat. There are multiple options of how to change the HTTP requests in order to obtain or set certain parameters value.

Similarly to viewing temperature profiles of walls, clicking on a radiator opens a new modal window, the contents of which, however, are very different. Left side of the window shows the air temperature measured by the thermostat, as well as a section for adjusting the temperature setpoint of both radiators in the room. To send the request, we use the command `$.get()`, where the address is the standard MAX! command. An example of the command to increase the temperature setpoint by 0.5 °C:

```
http://<ip-address>/set-manual?cube=116EA8&device=0F1D4A&temperature+=0.5
```

The right side of the window shows the current status of the clicked radiator's thermostat – its battery life, control mode (automatic / manual), and the valve position (0-100%). This information is obtained by an asynchronous request to an HTTP location specified by the device's address, which is a plain text in the JSON format. The JSON array contains all sorts of information, but we've decided to only display the mentioned.


Right thermostat settings

Current room temperature
21.5 °C
Temperature Setpoint
22.0 °C

Current device status	
Battery	🟩
Mode	Manual
Valve position	21 %

Figure 4.12 – Thermostat settings modal window

5 Further Approach

The subject of Intelligent environments is very vast and assorted. There are always many options how to proceed in creation of such a project. In this thesis, we‘ve successfully created a functional GUI for the Intelligent room, which periodically measures data and stores it in a database, using Arduino boards and other components, which we’ve described previously. However, it would be foolish to claim that the project is complete and ready for use or even sale. There is still a lot more work to be done around the room, concerning both data acquisition and the GUI, as by installing new sensors into the room, the source code of the interface must be adjusted accordingly. The current source code has not yet been tested for efficiency or security breach possibilities, and in the future of this project, adjustments are definitely going to be required. However, it fulfills all of its current tasks and the objectives of the thesis, and it‘s a sufficent tool for displaying all measured data and a manual control of the radiators.

Among other planned extentions for the project is the option to control another actuator for the temperature besides the radiators – the air conditioning unit (A/C). Implementation of this feature seems almost crucial for further research of air temperature control, as it enables direct action to decrease the temperature, rather than just increasing in case of the radiators. And besides all possible further research, it is still a convenient feature to control the temperature of an actual room through nothing but the internet browser.

Other actuators that might be added in the future are, for instance, ceiling fans or light intensity adjusters, automatic window openers etc. There truly are many ways to approach this project in the future, whether it’s temperature control towards a given setpoint, using all three of the actuators, optimization of energy consumptions for reaching a certain level of the room environment, or studying the sources and prevention of heat loss through the walls.

One of the most interesting possible further researches to be conducted in the room is the measurement and automatic control of human comfort level, which is based on human comfort modeling, which uses variables such as temperature, humidity, air freshness, air movement and light intensity, as was briefly described in Chapter 1. To conduct such research would require more than just the results of our project, and the Intelligent room has to be significantly expanded before it becomes possible. But maybe in the future, and maybe even thanks to this project, it will be.

6 Conclusion

Objectives of the thesis included participation on further development of the used technical solution in the intelligent room and development of a web-based Graphical User Interface for visualizing measured physical quantities.

The Intelligent room is a very interesting project at IAM in terms of monitoring and automatic control of room climate. Its main purpose is research and testing, which requires student participation on the project. However, the library serves mostly as a conference room for the employees of the Institute, and the provided technical solution might be eventually used to enhance the comfort level of the room's occupants.

Although many parts of the technical sphere of the room have been functional before our work on the project, we've succeeded in expanding the sensor network and therefore contributed significantly to the project's development. However, the most important contribution has been the creation of the GUI.

The created GUI works well for visualizing measured information in the room. It enables viewing of the time course of measured temperature of all the room's walls, in any desired interval, as well as the air temperature and humidity. There is an option to zoom, move and view individual measurements in the graph, which should fulfill all desires of a curious user. Although as mentioned in the thesis, the purpose of the GUI is not just replenishing curiosity, but it composes the first ground for further research in the room. It also allows the user to control the heating power of the two radiators in the room.

We've also described several possible researches that might be conducted in the room in the future, which will probably require expansions in the room's sensors and actuators. Therefore the thesis provided information how one should approach the expansion, and described the entire process of storing information measured by physical sensors into an information database in real time.

7 Resumé

Diplomová práca s názvom Development of Graphical User Interface for Intelligent Room sa venuje návrhu architektúry zberu dát a vytvoreniu uživateľského rozhrania pre inteligentnú miestnosť. Stručne opisuje problematiku inteligentných prostredí a produktov domovej automatizácie, a porovnáva vytvorené riešenie s ďalšími projektmi inteligentných miestností.

Inteligentná miestnosť sa nachádza na Ústave informatizácie, automatizácie a matematiky na FCHPT, STU v Bratislave, kde slúži ako knižnica a zasadačka pre zamestnancov. V miestnosti bolo nainštalovaných množstvo senzorov, ktoré merajú veličiny akými sú teplota a vlhkosť vzduchu, počet ľudí v miestnosti, ale aj teplotný profil vybraných stien, u ktorých sa meria teplota vonkajšieho povrchu, vrstvy izolácie a vnútra samotnej steny.

Jednotlivé senzory sú zapojené do siete mikroovládačov typu Arduino Yún, ktoré majú integrované dva procesory, jednotku mikroovládača (MCU) a procesnú jednotku (CPU), na ktorej beží distribúcia Linuxu s názvom OpenWrt. Na komunikáciu a výmenu dát medzi sebou využívajú procesory sériovú linku UART. OpenWrt obsahuje interpreter jazyka Python, čo umožnuje, aby v pozadí bežal skript v tomto formáte, ktorý po prijatí špecifickej požiadavky odošle požiadavku na MCU, ktoré získa dáta z jednotlivých pripojených senzorov a vráti ich ako pole vo formáte JSON. Túto požiadavku, ktorá spustí proces zberu dát, odosiela PHP skript bežiaci na priemyselnom počítači, ktorý sa nachádza na stene pri vstupe do miestnosti a obsahuje interný časovač, ktorý zaručuje spustenie skriptu v definovaných periódach. Ako odpoveď na HTTP POST požiadavku, ktorú odošle jednotlivým integrovaným počtačom mikroovládačov, očakáva pole vo formáte JSON. Jednotlivé dáta spracuje a uloží do MySQL databázy.

Na prezeranie namerných údajov uložených v databáze bolo vytvorené grafické užívateľské rozhranie (GUI). Je vytvorené ako webová aplikácia vo formáte PHP, ktorej interaktívne vlastnosti zabezpečuje JavaScript na strane klienta. Hlavnou súčasťou aplikácie je obrázok vo vektorovom formáte SVG, ktorý zobrazuje miestnosť v trojrozmernej projekcii sprava. Jednotlivé steny miestnosti zmenia farbu po nadídení myšou a po kliknutí na ne sa otvorí modálne okno, v ktorom sa nachádza graf s vykreslenými hodnotami nameraných údajov, tabuľka obsahujúca informácie o priemerných hodnotách, a funkcionalita na zmenu intervalu dátumov, pre ktorý sa zobrazujú namerané dáta v grafe. Na získanie údajov z databázy sa využívajú asynchrónne požiadavky (technológia AJAX) na interné PHP skripty. V práci sú popísané použité funkcie a metódy, ktoré zabezpečujú možnosti ako priblíženie do grafu, zobrazenie jedného konkrétneho nameraného bodu, postup a štruktúra vytvoreného modálneho okna a pod.

Významnou súčasťou práce je možnosť ovládať termostaty oboch radiátorov, ktoré majú na starosti kúrenie v miestnosti. Termostaty a ich riadiaca jednotka, ktorá je pripojená k internetu, sú značky MAX! a pomocou riešenia MAX! Home Automation, ktoré bolo vyvinuté dodatočne treťou osobou, je možné nastavovať a získať aktuálne hodnoty parametrov jednotlivých termostatov pomocou štandardizovaných HTTP GET požiadaviek. Podobne ako pri stenách miestnosti, kliknutie na ilustráciu radiátora otvorí modálne okno, v ktorom sa zobrazia všetky aktuálne údaje a ovládacie prvky pre žiadanú hodnotu teploty v miestnosti. Možnosť ovládať teplotu v miestnosti sa plánuje rozšíriť pomocou rozhrania pre ovládanie klimatizácie, preto bude možné teplotu meniť nielen smerom nahor, ale ju aj znižovať.

Touto diplomovou prácou projekt Inteligentnej miestnosti nie je ani zďaleka ukončený, ale práve naopak sa otvára veľké množstvo možností na výskum riadenia teploty vzduchu, skúmanie vplyvu počasia na teplotu vzduchu či stien miestnosti, minimalizáciu strát tepla do okolia a ďalších potenciálnych výskumov. Pre takéto projekty je vytvorená architektúra zberu dát a ich interaktívna vizualizácia základným bodom, od ktorého je možné sa odraziť.

Bibliography

AUGUSTO, J. C. 2013. Intelligent Environments: a manifesto. In *Human-centric Computing and Information Sciences.* [online]. 2013, vol. 3, no. 12 [cit. 2017-03-10]. Available on the Internet: <https://link.springer.com/article/10.1186/2192-1962-3-12/fulltext.html>. ISSN 2192-1962.

COEN, M. H. 1998. *Design Principles for Intelligent Envrioments.* Cambridge, MA: MIT Artificial Intelligence Lab, 1998. ISBN 0-262-51098-7.

GUBBI, J. et. al. 2013. Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions. [online]. In *Future Generation Computer Systems*, vol. 29, 2013, p. 1645-1660. [cit. 2017-04-05]. Available on the Internet: <https://arxiv.org/ftp/arxiv/papers/1207/1207.0203.pdf>

HRČKOVÁ, L., KRAJČÍK, M. 2015. Kvalita vnútorného prostredia a intenzita vetrania v knižnici SVF STU. [online]. In *ASB.sk* 2015. [cit. 2017-04-08]. Available on the Internet:< https://www.asb.sk/tzb/vetranie-a-klimatizacia/kvalita-vnutorneho-prostredia-a-intenzita-vetrania-v-kniznici-svf-stu)>.

KOPETZ, H. 2011. *Real-Time Systems. Design Principles for Distributed Embedded Applications. 2nd ed.* US: Springer, 2011. 378 p. ISBN 978-1-4419-8236-0.

KAZAKOV, D. 2015. MAX! Home Automation. version 1.9. In *dmitry-kazakov.de* [online]. 2015. [cit. 2017-04-05]. Available on the Internet: <http://www.dmitry-kazakov.de/ada/max_home_automation.htm>.

LAFTCHIEV, E., NIKOVSKI, D. 2016. *An IoT System to Estimate Personal Thermal Comfort. 3rd World Forum on Internet of Things (WF-IoT).* Reston, VA: IEEE, 2016, p. 672-677.

The PHP group. Function time(). [online]. In *php.net* [cit. 2017-04-15]. Available on the Internet: <http://php.net/manual/en/function.time.php>.

Attachments

Attachment A: CD medium – electronic version of the thesis and source files of the GUI