SILESIAN UNIVERSITY OF TECHNOLOGY

FACULTY OF AUTOMATIC CONTROL, ELECTRONICS AND COMPUTER SCIENCE

Internet Technologies – project work

Smartroom

December 2016

Table of Content

sadasd5.

1. Introduction

In the introduction, write a short description of the project, state what the project is designed to do and what are its general assumptions.

Smartroom is an intelligent platform that is used to control and monitor rooms of the user. Project was made both for subjects Internet Technologies and Microcontrollers Systems. Intention was to make stable- and safe-working platform that allows user to easy access and manage his own rooms from any place on the Earth. The only thing that registered user needs is an internet connection. We set the requirements a bit high. We had not experienced any webmaster exercises before, and now we intended to make a website that will have plenty of functionalities. So at the very beginning we were a bit afraid, but only for a little while. We were very motivated to make something useful and nice looking. We just wanted to learn something new and useful.

In the project we wanted to acquire data from real existing sensors (temperature, humidity, presence) and reed switches that will be controlled by microcontroller. Moreover, we wanted to implement ability to switch the light on and off. The data from sensors and devices is transferred through RS-232 to database that we created. To implement our website we decided to use Joomla! with its components and extensions. We were also obliged to set-up a localhost-server to work with. To do that we chose XAMPP with its Apache and MySQL components. Setting up a database server was also very important, to do that we used phpMyAdmin and MySQL Workbench tools. The internet site was made mainly to display the data from database and also to manipulate them (in case of light switching), in user-friendly way.

We allow the user to:
- Log in to enter our website
- Check the temperature in the room in selected time period
- Check the humidity in the room in selected time period
- Edit coordinates of an existing room
- Add new room
- Display position of a selected room on the map
- Easy-toggle between user's rooms

We ensure also that the data of users is protected as well as we could, by encrypting both data and tables that store data in created by us database.

2. Aim and scope of the project

Describe the aim and scope of the project and how the results may be used. Determine the requirements of the hardware or software needed for the project.

At the beginning we wanted to create a part of a intelligent house. We intended to combine real-existing hardware with software that will be able to manage and control applied microcontroller system. We decided to implement our idea for room not for whole house, because of that the size of the project will become enormously big and costs will definitely raise. We were also aware of that we should build-up our project in the way, which will make extension of a project simply.

We tried to implement system that will be user-friendly and easy accessible for registered users, remember that, it should be hard to crack-in. It was important, because data we manage is really private and sensitive.

Below we present stages we went through our project:

- Coming-up with an idea. Describing what exactly we intend to make. Identification of the problem and formulation of assumptions.
- Checking technologies/solutions that will be helpful during project.
- Choosing the most appropriate possibilities by consideration pro and contra arguments. Selection on the basis of established criteria and reasons for such solution.
- Implementation of the project. Writing scripts, discovering new technologies by practise, developing user interface.
- Start-up, verification and tests of the system. Finding out bugs and fixing them.
- Emphasization the possible ways of development of the project.
- Making conclusions after completion of the project.

While building up of our project we must have used a couple of software-tools and also some hardware to implement everything properly and to make it work. Below we insert a short description of each of the used elements, both hard- and software.

Hardware from the very beginning, just as we come up with an idea of making Smartroom©, was an integral part of the project. Each intelligent house and also intelligent room should has some specialist electronic devices to acquire data that could be later displayed and manipulated by an end-user. Description of each electronic part is not a topic of a report, so we will only mention names, and also functionalities of each element.

- Light bulb, that will be turned on and off by the user
- Humidity-, presence- and temperature-sensors, to acquire appropriate data. This data allow users to control the "state" of a room, by controlling temperature, humidity, and also checking if somebody is in the room or not.
- Reed switch, to check if windows (or doors) are opened.
- Microcontroller, to control all of the data and send them to the server via RS-232

Not only hardware were used during project, but also a couple of software tools, both freeware and developed completely by us:

- Joomla! with several extensions and additional modules – CMS(content management system) platform we used to develop our website. Joomla! allowed us to create website

just by adding articles. By using additional extensions and modules we could develop user interface, log-in form, write scripts (JavaScript or php) and include it into subpages.

- XAMPP – program that allowed us to set up localhost(via Apache) and also to manage database (via phpMyAdmin)
- MySQL Workbench – tool used to manage and create database
- C++ application – written by us, program that "push" data into database. At the beginning data was generated not acquired, but now application provide data acquisition.

While testing project on localhost all of the tools above are required, otherwise some functionalities could possibly not work correctly, or even the website will not open at all. Using Joomla! as main platform can simplify work in some aspects, but Joomla! is unfortunately far less portable then single .html or .php files. O the other hand, if we transfer our website on some Joomla! hosting, testing of the website will be quite easy. The only requirement for testing is account on our webpage and internet connection.

To sum up, costs of a single room includes only hardware, because the software is freeware, or was written by us. System developed by us is quite useful, because it really allows user to control and manage room. Therefore project that we have done, could be applicable i.e. for private clients to check living conditions in specified area of user's home, or even in big offices to monitor working conditions of each worker. After a couple of improvements our project could be also useful for gardeners.

2.1 Schedules

Below we present schedules of our work. First written at the beginning, and second that presents the current schedule that we applied.

2.1.1 Schedule approved at the initial classes:

W1 - Create a project schedule. Instalation and configuration Joomla! enviroment and data base.

W2 - Preparing interface, homepage and categories.

W3 - Registration for users and possibility to login for existing users.

W4 - Adding a control panel and its components.

W5 - Adding graphs.

W6 - Communications between database and our display system.

W7 - Adding Google Maps on website.

W8 - Refinement of our project and adding extra issues on the sites.

2.1.2 Current schedule (deviations are in *italic*)

W1 - Create a project schedule. Instalation and configuration Joomla! enviroment and data base.

W2 - Preparing interface, homepage and categories.

W3 - Registration for users and possibility to login for existing users.

W4 - Adding a control panel and its components.

W5 - Adding graphs. *Adding Google Maps on website.* – We were able to speed up a little. Because of that we could have included Google Maps a bit faster than we expected, but they were not 100% finished.

W6 - Communications between database and our display system. *Adding possibility to change the time period to be displayed.* – To make graphs more clear and user-friendly we added possibility to change the timeperiod of displayed temperature- and humidity- graphs

W7 – *Adding room editor and room creator. Completion of Google Maps Preparing final user interface, control panel interface improved.* – At the beginning we thought, that we will not be able to allow user to have more than one room. We had enough time to change it, and make a possibility to have more than one room per user in system. We also changed user interface completely and made it more clear and beautiful. We changed also control panel a bit, instead of displaying only text on/off, we included graphics. Google Maps were made 100% properly. It worked stable with nice interface.

W8 - Refinement of our project and adding extra issues on the sites. *Adding toggling between rooms*. – We allowed user to have more than one room, so we should have also implemented functionality as toggling between rooms of the user.

3. Software and hardware implementation

Due to the fact that our project contains both Internet Technologies and Microcontrollers Systems "parts", we divided this section into two parts: MS implementation and IT implementation.

3.1 MS implementation

This section is less important than the second one, but we decided also to include a little explanation of that how we implement issues such as i.e. sensors or lightning control.

At the picture below we present sample microcontroller system that will ensure all of the functionalities that we will describe in the section "IT implementation" depend in real data.
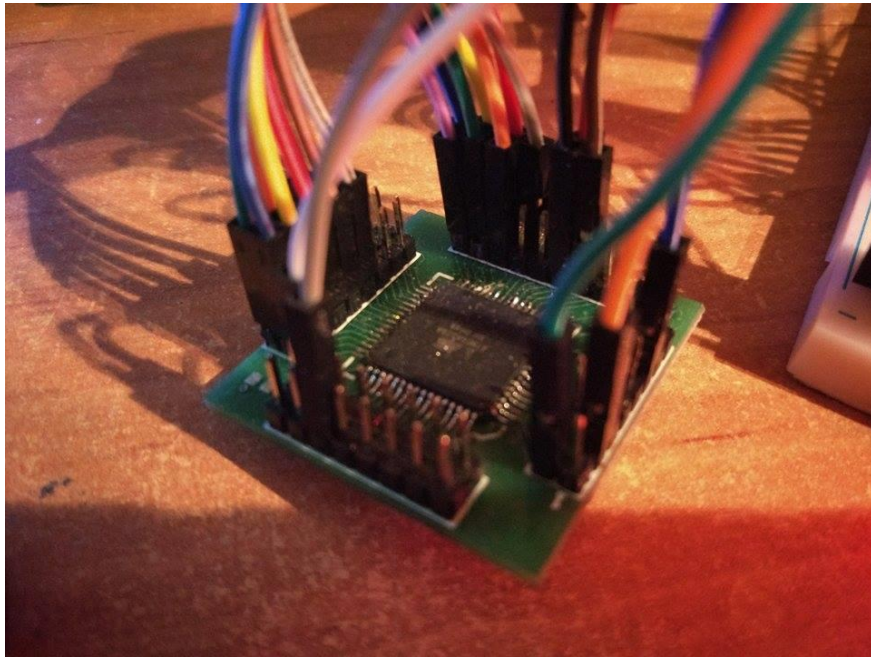
Picture above presents "brain" of our system: microcontroller ATmega128. We soldered it on an appropriate PCB to make usage of pins as easy as possible.
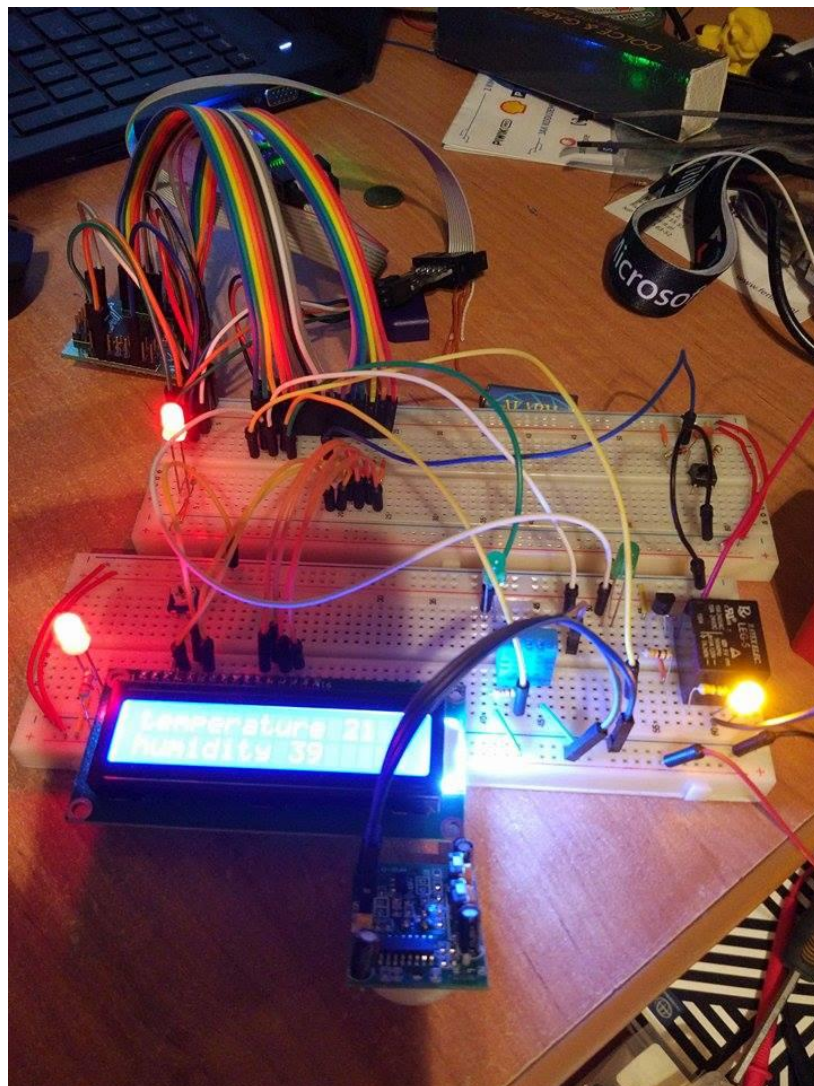
Picture above presents the whole microcontroller system. We can see reed switch, presence sensor, lightning control, humidity- and temperature-sensor. For testing purposes we included a display that shows current temperature and humidity values (temperature in Celsius degrees, humidity in percent).

Naturally, pictures above present prototype. All of the functionalities work, but the appearance can be definitely improved.

3.2 IT implementation

This is the main section of our report. In few following points we will describe briefly implementation of all functionalities from our website. We will also mention a bit environment we used that is called Joomla!.

3.2.1 Setting-up

Before we started to code we had to set up an environment. We chose Joomla! to work with. At the very beginning we were really sophisticated of Joomla!. It looked easy-to-handle for us as first time users. Problems appeared later, we will also describe it.

To install Joomla! we needed XAMPP – cross platform that contains MySQL and Apache contents. Thanks to XAMPP we were able to set up a localhost.
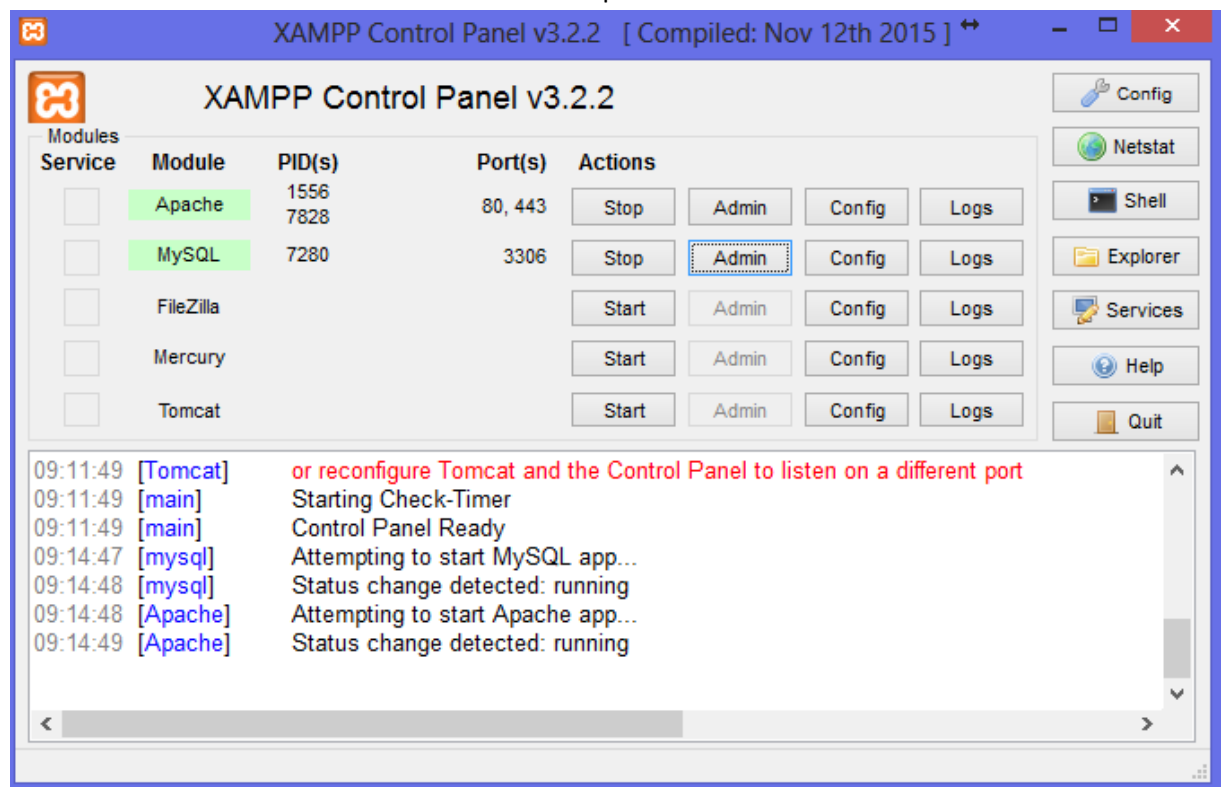

XAMPP

After setting up the localhost we could install Joomla!. As we mentioned before, Joomla! is CMS – content management system, that means Joomla! allows developer to easy manage content that is included into website. Really easy to include were: articles, photos, text.
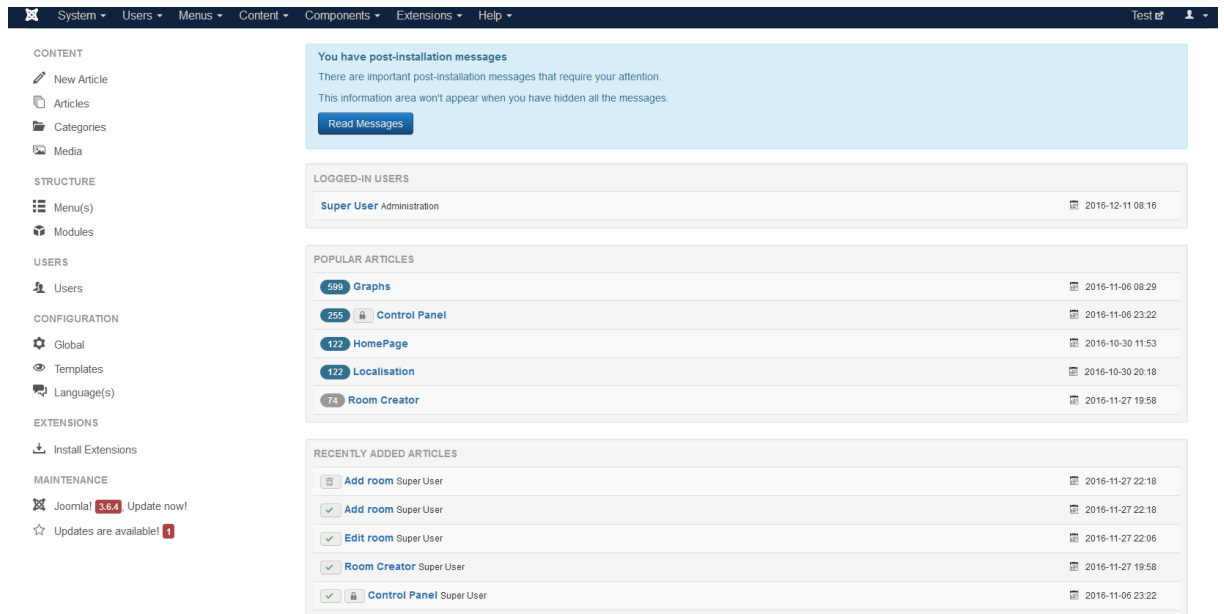
Joomla! workspace

During installation Joomla! installs its own database. That database could be really easy manage through MySQL Workbench or phpMyAdmin. What is remarkable, Joomla! adds its own prefix to each name of auto-generated tables. Because of that it is harder to crack into such database, and also whole internet site is safer.
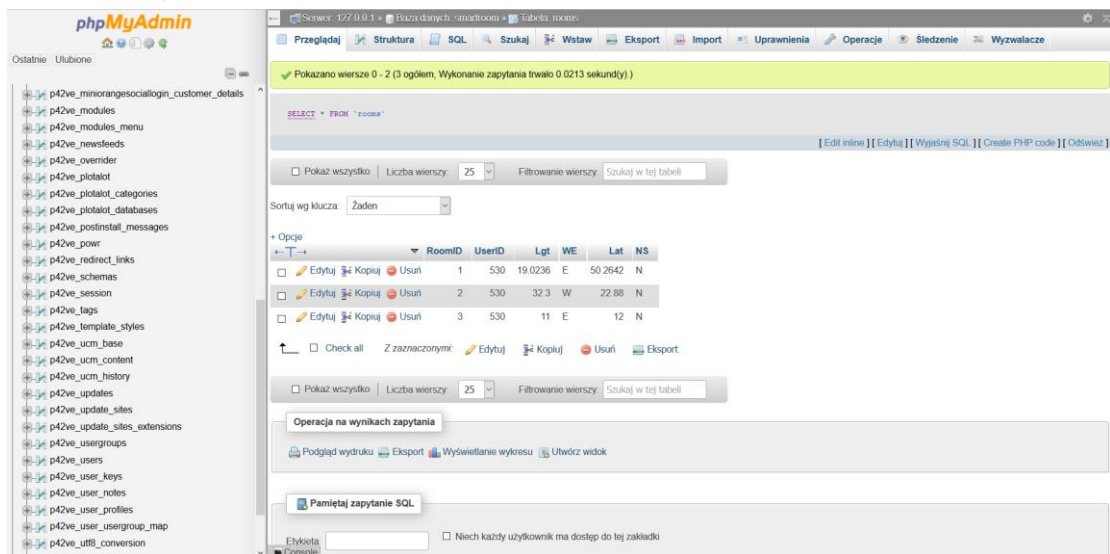
phpMyAdmin

Naturally, we should have add some tables created only by us, not by Joomla! itself. For that we used MySQL workbench. We used it also to set up relations between tables. In the ER Diagram below structure (tables, relations, columns names) of our database is shown. What is interesting, MySQL Workbench generates ready ERD by itself, does not matter if structure of database is complex or simple.
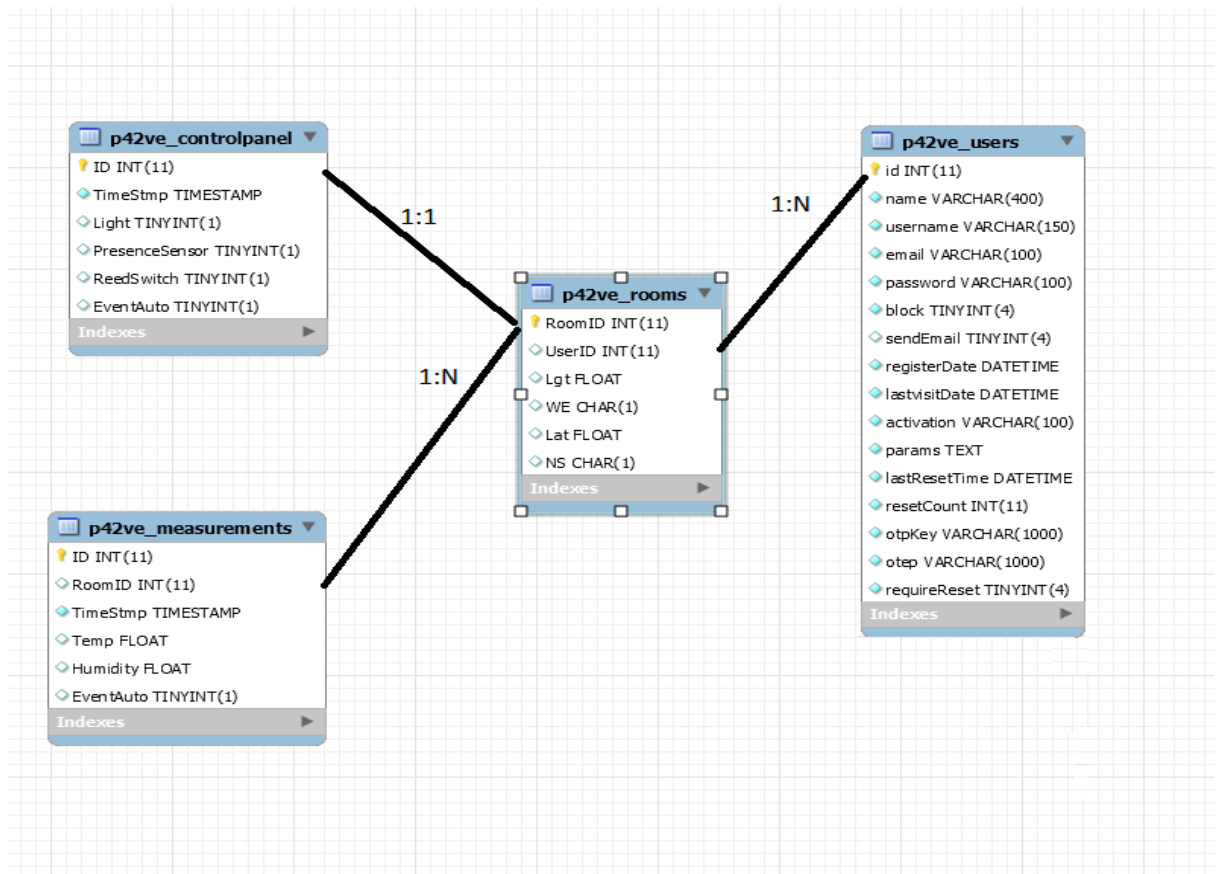
ERDiagram in MySQL Workbench

3.2.1.1 Handling with database

In our project one of the most important thing is to communicate fluently with database to perform operations such as adding records, updating records, etc. We applied two different approaches: connecting via website (Joomla! articles) and through C++ application.

Each article/functionality that requires connection with data base include code that is very similar to that code at the snippet below. It performs connection with database and also example operations. Syntax is similar to PHP, but it is not strict PHP. It is kind of Joomla! framework, that allows to connect to the database and perform some operations. The query itself is really similar to SQL syntax, but unfortunately for us there were a couple of annoying mistakes that we made, but we could not find because of poor Joomla! debugging.

```
// Get a db connection.
$db = JFactory::getDbo();

// Create a new query object.
$query = $db->getQuery(true);

// Select all records from the user profile table where key begins with "custom.".
// Order it by the ordering field.
$query->select($db->quoteName(array('user_id', 'profile_key', 'profile_value', 'ordering')));
$query->from($db->quoteName('#__user_profiles'));
$query->where($db->quoteName('profile_key') . ' LIKE '. $db->quote('\'custom.%\''));
$query->order('ordering ASC');

// Reset the query using our newly populated query object.
$db->setQuery($query);

// Load the results as a list of stdClass objects (see later for more options on retrieving data).
$results = $db->loadObjectList();
```

Second approach of connecting with database was to write C++ application that will push acquired data from sensors just into database of the website. Program was also useful for testing, when we needed a massive amount of data to check if we are able to draw plots of temperature and humidity from last month/week or even a year. Code snippet below presents .h file of a class that is a part of whole application.

```cpp
#pragma once
#include <stdio.h>
#include <my_global.h>
#include <mysql.h>
#include <string>
#include <stdio.h>
#include <iostream>
#include <string>

using namespace std;

class DBBClass
{
public:
    DBBClass(void);
    ~DBBClass(void);
    int Menu(int Err);

private:
    void finish_with_error(MYSQL* con);

    //!Methods returning int return 0 when finised without errors
    int Select();
    int Insert(string TableName);
    int InsertSampleMeasurements(string TableName, int N);
    int ConcatenateSampleMeasurements(string TableName, int N);
    int Info();
    int Drop(string TableName);
    int Create(string TableName);
    int CloseConnection(MYSQL* con);
    int Func, Rows;
    int Choice;
    bool InfoRunned;
    string query, TableName, DataBaseName, Pwd;
    MYSQL* mysql;
    MYSQL* con;
    MYSQL_RES* result;

};
```

3.2.2 User interface

Joomla! delivers a plenty of freeware templates. They are really similar to CSS files, but installation is far easier. We should have downloaded template that we liked the most and then install it into Joomla! environment. Naturally, we should have made a couple of changes. I.e. we must have changed the value of the background colour, because the login form just disappeared. Snip of the code is presented below.
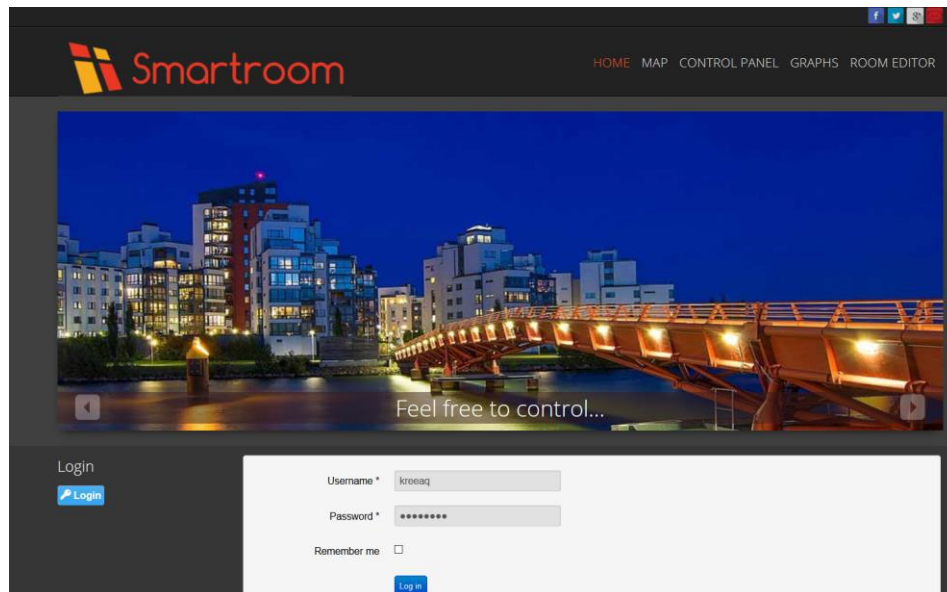

As the above image shows, Joomla! templates are just CSS files. Changes could be made in the same way as in "normal" CSS files.

In the picture below the front-end of the website is presented. Static navbar with logo, slideshow below and visible login form at the bottom.

Joomla! is perfect for websites that only functionality is to look nicely. Preparing interface is really fast and intuitive. Unfortunately much more problems with Joomla! appear when some useful non-templateable extentions should be coded.

3.2.3 Functionalities

Firstly we should clarify how exactly we could insert JavaScript, HTML, CSS, PHP into Joomla! articles.

As we mentioned before, Joomla! has a plenty of extensions and modules. The best of them are not free of charge unfortunately, but we found a couple of useful modules and extensions that are freeware and also useful. One of them is Sourcer, tool that allows to insert snip of codes written in JS, HTML, PHP, CSS just into articles, so just into subpage.


```
{source}
<!-- You can place html anywhere within the source tags -->


<script language="javascript" type="text/javascript">
    // You can place JavaScript like this

</script>
<?php
    // You can place PHP like this

?>
{/source}
```

Next useful tool was Slide Login Module. That tool allow user to log in and register to website. Naturally, the appearance of the login module is adjustable

3.2.3.1 Room Editor/Creator

As the name says, room editor and room creator allow user to create new and edit existing room. Here we can point out first huge problem with Joomla!. When we tried to implement form in HTML we was not able to send all of the data we wanted to be send, due to that fact records in database were not updated. Joomla! pointed out us that it is a problem, but did not point out the place in code, moreover, it did not point out the full description of a problem at all. Debugging in such conditions are really tough. Finally we found the error, it was something like dot instead of comma.


```
//Setting up a connection and performing query

$db = JFactory::getDbo();

$query = $db->getQuery(true);

$fields = array(

$db->quoteName('Lat') . ' = ' . $db->quote($lat),

$db->quoteName('NS') . ' = ' . $db->quote($latitude),

$db->quoteName('Lgt') . ' = ' . $db->quote($lgt),

$db->quoteName('WE') . ' = ' . $db->quote($longitude)

);

// Conditions for which records should be updated.
$conditions = array(

$db->quoteName('UserID') . ' = '.$db->quote($user->id),

$db->quoteName('RoomID') . ' = '.$db->quote($noroom)

);

$query->update($db->quoteName('p42ve_rooms'))->set($fields)->where($conditions);
$db->setQuery($query);

$result = $db->execute();
```

3.2.3.2 Control Panel

Now we enter the most important functionality in our project. Lightning control, sensors control, data acquisition etc. all of that things are part of control panel.

1. **Lightning**

   In this section we firstly used JavaScript to dynamically change state of light by pressing a button. We also had to use PHP because of the fact that we must have push data into database, so both JavaScript and PHP with implemented connection between both of them were used. Problem was, how to "connect" JS and PHP, as shown in the code snippet on the right, usage is quite simple, just echo with json_encode(). Image on the left is the snip form website, image on the right is snip from code.

```
<div style="text-align:center">
<img src=<?php if($LightState['0']>0){echo "
lightbulbON.jpg";}else{echo " bulbOFF.jpg";} ?> alt="Bulb"
style="width:150px;height:150px;text-align:left">

</div>

<script language="javascript" type="text/javascript">

function LightState() {

var LightState = <?php echo json_encode($LightState); ?>;

if (LightState == 1)

{<?php

$db = JFactory::getDbo(); $query = $db->getQuery(true);

| // Fields to update.
$fields = array( $db->quoteName('Light') . ' = ' . $db-
>quote('1') );

    // Conditions for which records should be updated.
$conditions = array( $db->quoteName('RoomID') . ' = '.$db-
>quote($UserRoomID));
$query->update($db->quoteName('p42ve_controlpanel'))-
>set($fields)->where($conditions);
$db->setQuery($query);
?>
else

{..}
window.location.reload(false);

</script>
```

2. Sensors section

Another dynamically changing values in our database are states of sensors: reed switch and presence sensor. The purpose is to check the state of the sensors once a three seconds to check the current state of sensors. We made it in JavaScript, because it is appropriate to use when something on the website changes dynamically. In the picture on the left snip from front-end website is shown, on the right snip of code from this section.


```
<?php

$db = JFactory::getDbo();
$query = $db->getQuery(true);
$query->select($db->quoteName(array('ReedSwitch')));
$query->from($db->quoteName('p42ve_controlpanel'));
$query->where($db->quoteName('RoomID')." = ".$db-
>quote($noroom));
$db->setQuery($query);
$SwitchState= $db->loadRow();

?>
<script language="javascript" type="text/javascript">
setInterval(function() {window.location.reload();},
3000);

</script>
```

3.2.3.3 Graphs

We wanted to attach some dynamically changing plots on our website. We considered using Joomla! extensions and Google Chart API. After massive amount of fruitless hours we decided to change the approach and to programmatically insert plots using Google Chart API. We allow user to change the time period of displayed plots. Options are: current/day/week/month/year. For testing that functionality really helpful was our C++ application, that could have push massive amount of sample data in a few seconds. User is also able to put a marker on the plot. We had also some problems with transferring data from database into Google Chart API (exactly JavaScript), so that it will work properly. Problem was solved through usage of php echo in appropriate way. Left hand side snip from webpage, right hand snip from code.
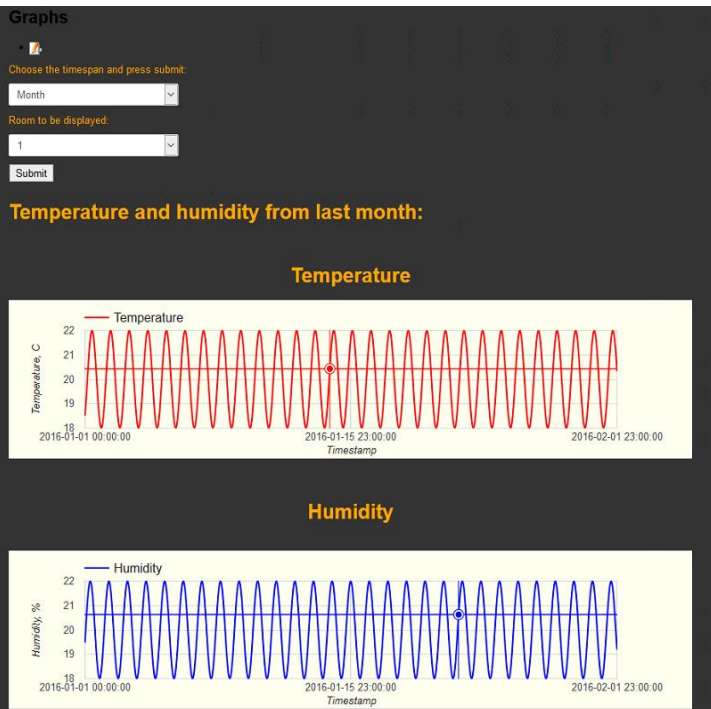

```php
for ($i=1; $i<=$NumOfRecords; $i++)

{
$query = $db->getQuery(true);
$query->select($db->quoteName(array('TimeStmp', 'Temp',
'Humidity')));
$query->from($db->quoteName('p42ve_measurements'));
$query->where($db->quoteName('ID')." = ".$db->quote($i)." AND ".$db-
>quoteName('RoomID')." = ".$db->quote($noroom));
$db->setQuery($query);

$TempTbl[$i] =$db->loadRow();
}
<html>
<head>
<script type="text/javascript"
src="https://www.gstatic.com/charts/loader.js"></script>
<script language="javascript" type="text/javascript">
 google.charts.load('current', {packages: ['corechart', 'line']});
google.charts.setOnLoadCallback(drawBasic);
function drawBasic() {

        var data = new google.visualization.DataTable();
        data.addColumn('number', 'Timestmp');
        data.addColumn('number', 'Temperature');

        data.addRows([
<?php

for($i = 1; $i<=$NumOfRecords; $i++)
{
echo "[{v:".$i.", f:'".$TempTbl[$i]['0']."'},
".$TempTbl[$i]['1']."],";
}?>   ]);
        var options = {
         colors : ['red'],
         hAxis: {
            title: 'Timestamp',
 ticks: [{v:1, f:'<?php echo $TempTbl[1]['0'] ?>'}, {v:<?php echo
$NumOfRecords/2 ?>, f:'<?php echo $TempTbl[$NumOfRecords/2]['0']
?>'}, {v:<?php echo $NumOfRecords ?>, f:'<?php echo
$TempTbl[$NumOfRecords]['0'] ?>'}],
```
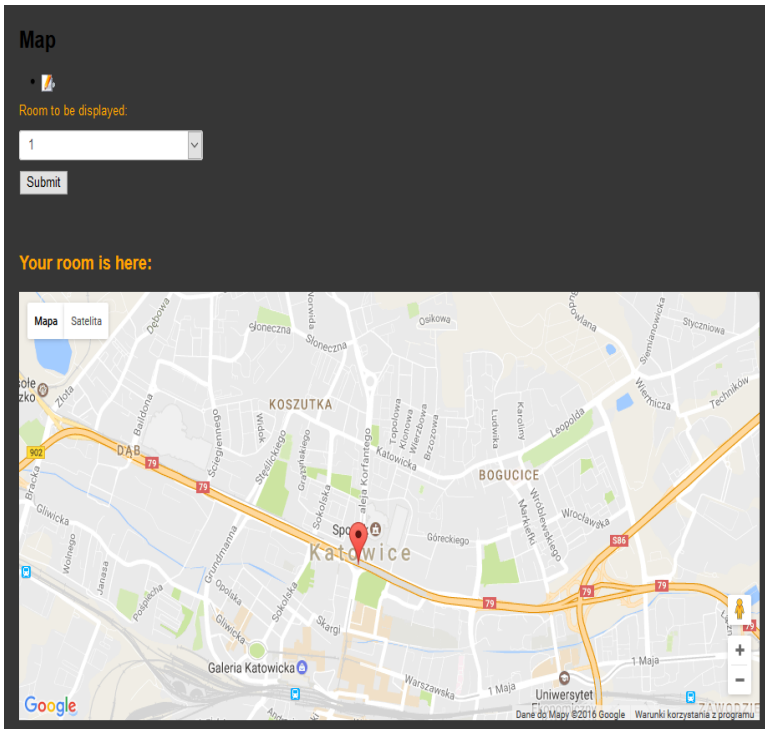
3.2.3.4 Localisation

We know that everybody know where his room is, but we just wanted to implement something "fresh". We use Google Maps everyday but we did not exactly know how it works. By implementing our maps section we have a slight knowledge about how Google Maps works and how to implement it. Here snip from website and snip from code are presented. We perform selecting latitude and longitude depending on that which user is logged in. Data from tables are calculated properly to set a marker in appropriate place on the globe. That functionality could be useful for user that has many room worldwide. Such a solution reduce chance of mistake by picking inappropriate room.

Like by using Google Chart, also by using Google Maps we should use JavaScript combined with PHP to extract appropriate data from database. We had no significant problem by implementing that functionality.

```
<!DOCTYPE html>
<html>
<head>
<style>
#map {
height: 400px;
width: 100%;
}
</style>
</head>
<body>
<h3 style="color:#FFA500">Your room is here:</h3>
<div id="map"></div>
<script>
function initMap() {
var uluru = {
lat: <?php echo $Lat; ?>,
lng: <?php echo $Lgt; ?>
};
var map = new google.maps.Map(document.getElementById('map'), {
zoom: <?php echo $Zoom; ?>,
center: uluru
});
var marker = new google.maps.Marker({
position: uluru,
map: map
});
}
</script>
<script async defer
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyD7TotheiQHyP9
pIoi7qyxlDvjrmxhm7C4&callback=initMap">
</script>
</body>
</html>
```

At the end of describing functionalities we have to add that each functionality has also ability of toggling between rooms of a user.

4. Summary

We are really happy to announce that we have filled all of the points from schedule. We even made a little bit more. We are really satisfied because we started from "level zero" and now we can say that we are able to write simple or intermediate-hard functionalities on internet sites.

There are of course a couple of things that we can include to our project, to improve and make it more attractive:
- Adding live streaming from room.
- Adding ''visiting history" of a room, to see when room was occupied.
- Adding some kind of mechanism to regulate the temperature.
- Attach more lights to be controlled
- Adding sensor that will measure light intensity

We could implement control of some devices like automatic garage door, or automatic curtain. This way we could make our project more like home automation. On the other hand, we could add some sensors useful for gardeners and make it more like platform for gardeners, where they will be able to check parameters important for plants. Devices delivered by us will measure that parameters, send it to database and then webpage could display all of the data in nice and tidy way.

5. Literature

It is required to reference the used literature/other sources of information, e.g. like this [1]. If anywhere were adopted drawings, texts, part of the program code, you must put this information in the list of references, and use a link in the text pointing to a position in the given literature [2].

1. Last name, first name, title, publisher, year of publication.
2. Website URL, with the date of download.

Note 1: The titles of the sections are only exemplary - do not adopt them literally!

Note 2: Full program code with possible database content, other supporting files and a description of the installation and requirements for the installation have to be delivered in an electronic form, together with the electronic version of this report! Provided software and given description should enable a test installation of the project. If the software and/or related environment are not standard, and it can make difficulties to test the project, then it should be clearly described and justified!