Research Article

Filip Pružinec, Peter Mego, and Renata Ďuračiová*

A low-cost web-based smart street lighting geographical information system for smaller towns and villages

<https://doi.org/10.1515/comp-2020-0205>

Received Mar 20, 2020; accepted Jul 18, 2020

Abstract: The use of LED lights in smart street lighting to save energy is efficient and very common in many developed cities. However, such a system results in higher initial costs, which may discourage smaller towns and villages from its purchasing. This paper therefore deals with the design and development of a smart street lighting control system for smaller cities and villages. Such a system is required to be economical from the perspective of development and production. Considering other requirements such as platform independence, high availability and broad support, we proposed to build a system as a web application using a custom light control service component. The result is a web application that consists of a user interface served by a web server and an application server used to communicate with light control service. In addition, the system is designed as a geographical information system to be easy to use for managing street lights in groups or individually, while displaying them on the map. The system allows automatic sensor-based light intensity modulation by default with the possibility of manual adjustment or override of the illumination. It also includes a device error notification system with a tool to navigate faulty devices. The presented system is a low-cost solution for intelligent street lighting control designed for smaller cities or villages. They can apply the designed architecture of the system and the specific technologies suitable because of their low-cost implementation.

Keywords: intelligent street lighting, smart city GIS, light control service, web map application

1 Introduction

Taking advantage of the LED lights in the street lighting systems is nowadays very common in the western part of Europe due to better efficiency, higher quality light source and lower maintenance costs. LED luminaires have started to replace sodium, mercury or filament lamps broadly in Slovakia in recent years due to European Union's efforts to reduce electricity consumption. An intelligent street lighting system can increase the efficiency of the energy consumption reduction by adapting the lighting to the actual illumination of the covered area. Advantages of smart street lighting using LED lights are lowering energy costs, light source quality improvement and better maintenance. Besides the already stated benefits, such a system can be extended with electric car chargers, Wi-Fi internet connection or a navigation system. However, these benefits come with a higher initial deployment costs, which can discourage smaller cities and villages from purchasing such a system. The aim of this paper is therefore to design a smart lighting system for small towns and villages. Due to its efficient use, we propose it as a geographic information system (GIS) with the possibility to visualize the lights on the map.

The idea of applying spatial data and GIS to create the low-cost electrification strategies is not new, globally it is presented in [1]. In paper [2] a street lighting infrastructure in a rural-urban setting using GIS is introduced. A case study presented in [3] proposed a GIS-based tool for the estimation of the potential of renewable energy production and reduction of the energy needs to the exploitation of renewable energy sources with respect to existing urban neighbourhoods. In addition, article [4] describes a method for measuring lighting parameters in various streets through the use of an aerial imagery of towns and cities. The results of that study confirmed that also the lighting measurements, luminaire position, and electricity consumption data can be integrated into GIS to improve energy saving. GIS and its advantages for better management of energy efficient public lighting is also described

*Corresponding Author: Renata Ďuračiová: Faculty of Civil Engineering, Slovak University of Technology in Bratislava, Slovakia; Email: renata.duraciova@stuba.sk

Filip Pružinec: Faculty of Civil Engineering, Slovak University of Technology in Bratislava, Slovakia; Email: filip.pruzinec@stuba.sk

Peter Mego: Megsys Europe s. r. o., Bratislava, Slovakia; Email: peter.mego@hotmail.com

in [5]. In addition, according to [5], much attention is being paid to improve the energy efficiency of public streets in most countries, as the street lighting is accountable for large electric power consumption as well as night light pollution [6]. Urban lighting infrastructures are also an important source of the energy consumption, accounting for some 2.3% of the global electricity consumption [7].

In terms of the technologies used in GIS for smart street lighting, for example, the Outdoor light desktop central management architecture using Internet of Things (IoT) and GIS technologies is described in [8]. The remote controlled smart LED lighting system is proposed, for example, in [9]. An example of park smart lighting is described in [10], a case study on a lighting system on an area not covered by the Internet is presented in [11]. An example of an implementation of smart lighting in buildings is presented in [12].

In the field of the lighting systems, Philips (www.lighting.philips.com) offers solution to connect luminaires to the cloud via the Starsense Wireless outdoor light Controller (OLC), which can be installed on the LED Starsense luminaires stand-alone lighting system (LumiMotion) specifically designed to minimize the light consumption without compromising people's safety. In addition, LonMark (www.lonmark.org) presents a system architecture for dynamic street-light control, which relies on key components such as light point controllers, gateways (OLC), segment controllers, meters, sensors, digital and analogue inputs and other devices (www.lonmark.org). Although most of the above described solutions reduce the energy consumption, they are still too expensive [8] and thus difficult to apply to smaller towns lighting. That is why we propose the low-cost web-based GIS for smart lighting for smaller towns and villages.

This paper specifically deals with the design and implementation of a smart street lighting system in Slovakia that should be economical from the perspective of development and production. Considering also other requirements such as platform independence, high availability and broad support, we designed the system as a web application using a custom light control service (LCS) component.

In the following chapters we describe the design, architecture, implementation and testing of the proposed system, which can form the basis for creating similar web-based smart lighting systems for smaller towns.

2 Design and architecture

The main requirement for a smart lighting system is that it should be an effective tool for street lighting management with a simple user interface (UI) that displays system devices on a map. Automatic modification of light intensity based on real time illumination should be provided together with the possibility of manual adjustment of light intensity.

In addition, the system should take advantage of open-source technologies and provide:

- user interface for efficient street lighting system management
- display of the devices on the map
- light intensity control based on various parameters
- notification system
- automatic functionality without the necessity of manual management
- available map services support - Google Maps (www.google.com/maps), Bing Maps (www.bing.com/maps), OpenStreetMap (www.openstreetmap.org/)
- ease of use
- simple extensibility and high availability.

Therefore, we designed a system consisting of three components: 1. a user interface, 2. a web and application server, 3. a light control service component (Figure 1). Individual components communicate in order to meet user's requirements, who manages the system. All parts of the system are sequentially described below.

Figure 1: The basic system architecture of the designed low-cost web-based smart lighting system

2.1 User interface

We proposed the web UI that can be displayed in all modern web browsers (*i.e.* Google Chrome, Mozilla Firefox, Opera, etc.) both on desktop and mobile devices. A user has a user account, to which all his application data and settings are bound. The UI consists of a map window and a settings panel. The map window should display street lights as point features in a base map. These devices can be distinguished by the colour of a group they belong to.

In addition, the UI should provide an ability to:

- select devices by location on the base map
- change the settings of the selected devices
- assign or change the group of the selected devices
- change the group settings in the settings panel
- delete the device groups.

Every change of settings causes a change in the system state. Multiple persons are expected to access the application from the same account, therefore it is necessary to display every change of the state in every running instance of the application with the same user account. The state of the system is stored globally on the server. Every change of the state must be sent to the server and every change received by the server must be displayed in the UI. During the state change, if an error occurred, an alert notification is displayed in the UI.

2.2 Server

The server component resolves requests from the client components – the UI and the LCS. The server should:

- manage the process of resolving client requests and maintain the system's global state stored in the database
- provide the UI in the form of HTML documents, JavaScript (JS) files, cascade style sheets (CSS) and other files to the browser
- provide a user authentication, based on which certain requests and usage of the application are allowed or prohibited
- mediate user requests to the LCS
- validate these requests
- store and provide the state of the request process
- alert the user with a notification on every error occurrence.

The server should receive some requests from the LCS periodically. These requests include the actual state of the devices. Based on the received state and user settings, the

server should compute desired light intensity for every device and respond in a predefined form.

Various information about the street lights such as groups, users, and state of the application, are stored in the database (DB). The global state of the application consists of three different states stored in the DB:

- real state
- optimistic state
- events.

Parameters of the devices after a successful state change operation are stored as the real state. The optimistic state, or the desired state, reflects the state of the devices set by the user, but not yet successfully achieved. This state changes every time a user changes the settings of any device or group. History of these changes should be stored as events in an order of occurrence. By using these three states we can show the user the desired state of his devices while the server and the LCS works on resolving his requests in the background. This approach frees the user to set his devices in the final desired state without waiting time after every change.

2.3 Light control service

The LCS is a service that accepts the device settings change requests and responds with a real state change information. This service should also send the real state of the devices to the server periodically and awaits the next state settings in the response. The real light intensity changes and the street light settings changes may not be successfully set for all devices in one request. Information emitted from LCS to the device network may not reach all the devices due to environmental causes. The LCS should no-

Figure 2: Designed street light consisting of an aluminium cast, AC/DC, a chip on board (COB) LED module, a light control module and a lens (Photo by Filip Pružinec)

Figure 3: The design of the system components - user interface, web and application server, light control service, with the used technologies and communication protocols

tify the server about the unsuccessful changes and their corrections.

The LCS consists of a system of gateways (GW) that communicate with the light control modules installed on the street lights. The street lights consist of an aluminium cast, AC/DC switching power supply operating in both constant voltage modes, a chip on board (COB) LED module, a light control module and a lens (Figure 2).

The GW is an industrial computer connected to the internet in order to communicate with the server and with a minimum of one universal asynchronous receiver transmitter (UART) interface. The GW listens to JSON messages from the server via the WebSocket (WS) protocol. The UART is used for communication with a radio frequency (RF) module. The RF module serves as a bidirectional access point for the radio modules installed in the light control modules. The GW communicates with the light control modules via a custom serial protocol (Figure 3). The light control modules adjust the light intensity of the street lights upon reception of the command from the GW. After a successful command handle, the light control modules respond back to the GW with a message containing the data about the actual state of the given lamp. The light control modules also forward GW messages to the other modules within a maximum possible range, depending on a signal strength, to reach the desired modules in a minimal number of transfers. The system, its components, and communication protocols are depicted in Figure 3.

3 Implementation

We propose to implement the UI and the web and application server using JavaScript programming language and a project starter kit ReactQL. ReactQL creates a code base for the server and the client using React.js (reactjs.org/), Apollo GraphQL (graphql.org/learn/), Koa (koa.js.com/), Redux (redux.js.org/) and other libraries. The presented components communicate using the Hypertext transfer protocol (HTTP) [13] and the WS protocol [14]. The UI is provided by the web server using HTTP and continues to communicate with the server via both HTTP and WS. The LCS is controlled by the application server via command messages sent over the WS connection (Figure 3).

As mentioned before, we created the UI codebase using ReqctQL starter kit that comes with a React.js root component, an Apollo GraphQL client side implementation and a Redux state storage implementation. React.js serves for building user interfaces based on components that are updated and rendered when the application data changes. To keep the application state in one place, we take advantage of the Redux library, also used by the Apollo GraphQL. The Apollo GraphQL is a set of tools for GraphQL query language that is used to create application programming interfaces (APIs). The Apollo GraphQL in combination with the Redux state and React.js provide a simple and efficient environment, which facilitates management of all processes concerned with data querying, storage, manipulation and view. By adding the OpenLayers (openlayers.org/) library for effective work with interactive web maps (Figure 3), we

Figure 4: Street light management user interface containing the map window displaying the devices and the settings panel

managed to create the UI for the web GIS application consisting of a map window and settings panel (Figure 4).

The map window displays the street lights on a base map as point features with a colour depending on the group they belong to (Figure 4.). It also provides basic web map functionality – zoom, pan and rotate. As a base map, we used OpenStreetMap Standard layer with an option to switch to BingMaps aerial imagery by clicking on the button in the bottom left corner of the UI (Figure 4.). In the initial start of the system all street lights belong to a single group with default settings. A list of groups is on the top part of the settings panel (Figure 4.) and it allows the user to extend the group to see the sublist of street light IDs belonging to the group. By selecting the settings button, which can be found to the right of the group name, a settings window of the group appears (Figure 5).

The group settings allow the user to modify the behaviour of the whole street lights group. The default behaviour is an *Automatic* behaviour (Figure 5) that orders the street lights to adjust the light intensity to the illumination of the area they are located in. A *Semi-Automatic* option lets us alter the automatic light intensity by multiplying the output percentage. A multiply coefficient ranges

from 50% to 150% and allows us to also dim the lights. An *Add* option (Figure 5.) can be used to prioritize a group against the other groups with similar light conditions. A *Manual* behaviour sets the light intensity directly to a specified value, overriding the *Automatic* behaviour completely. The group settings represent a powerful interface allowing the users to adjust the settings of the devices by their importance and spatial relations while taking advantage of the automatic light intensity adjustment using sensors. The group settings provide a possibility of manual override on specific events.

By using the map window (Figure 4), we can select street lights. Selected street lights will appear at the bottom part of the settings panel (Figure 4), allowing us to move them to an existing group, or create a new one. Selected street lights can also be set for the *Manual* behaviour directly, overriding the group settings. Deleting the groups (Figure 5) is allowed until a single group, which cannot be deleted, exists. While deleting a group that contains street lights, the user is prompted to select another group to transfer these lights into. All mentioned actions cause the change of the application state that is sent to the server via GraphQL queries. The change of the appli-

Group - default ●

Settings

Automatic ☒

Semi-Automatic

☐ Multiply 150 %

☒ Add + 20 %

Manual ☐

40 %

Group devices

💡	168.192.1.0
💡	168.192.1.5
💡	168.192.1.1
💡	168.192.1.2

DELETE GROUP SAVE

Figure 5: Group settings window

cation state is displayed instantly in all running instances of the application on receive of the WS messages containing the actual state received from the server. The UI sends four types of requests to the server: *changeGroup* that also serves for the group creation, *deleteGroup*, *changeGroupSettings*, and *setDeviceValue* used for the selected devices. All these requests have a predefined JSON form specified by GraphQL schema shared by both UI (GraphQL client) and the server.

The web and application server created by ReactQL starter kit contains the basic implementation of the GraphQL server in a Koa web framework. The server component is running using JavaScript runtime Node.js. Koa is used to create the HTTP server using Node.js HTTP API (nodejs.org). For the system data storage, the MySQL database management system (DBMS) (www.mysql.com/) is used and accessed via knex.js query builder. Knex.js is used by Objection.js library, which is an object-relational mapping (ORM) library introducing object-oriented approach into a relational database [15].

The server component manages the data flow and communication between the UI and the LCS. The LCS requests the server for light intensity values periodically via WS pro-

tol, sending the real state of street lights and the measured illumination converted to percentage values. An example of measured values can be seen in Figure 6. The measurement was performed during the morning.

The measured values represent the resistance of the photoresistor, which is connected to the voltage divider and the voltage is then measured by an analog-digital converter. These values are converted to percentage, while 100% represents the lowest and 0% the highest measured illumination. The server resamples these values considering the 20% minimum and 90% maximum in order to maximize the sensitivity during the night and day shifts. Afterwards, the server modifies these values depending on the street light settings set by the user and stored in the DB. Modified values are sent back to the LCS, that handles the street lights accordingly.

When the group settings or selected lights settings are changed, the server receives a *changeGroupSettings* or *setDeviceValue* request. These are the settings that have direct impact on resolving the periodical light intensity request of the LCS, while the *changeGroup* and *deleteGroup* requests only assign these settings to different street lights. Server's *changeGroupSettings* and *setDeviceValue* call resolves into an instant state change on the UI as well as an instant change of the optimistic state stored in the DB. Every change of the optimistic state is stored as a JSON data event in the DB. Afterwards, the server orders the LCS to change the behaviour of the corresponding lights. When the *changeGroup* functionality is called, the server changes the optimistic state in the DB, but does not request the LCS to set the lights immediately. The real state of the lights changes on the next periodical LCS request. As mentioned in section 2, the information emitted from the GW often does not reach the addressed lights on first attempt. The LCS response to the setting changes after a successful response of the light control modules or after a certain period of time, specifying the light modules were not successfully set. This type of event results in an error notification on the UI displayed in the error log window. The LCS sends further requests to the street light network until the error state is patched. Then the LCS notifies the server about the error state correction, that notifies the UI. Faulty devices remaining in the error log window can be navigated to via link on GoogleMaps web or mobile application. Another cause of the error state can be the LCS WS connection error caused by a power outage, a weak network connection, etc. In this case we take advantage of the stored event data. The user is allowed to continuously modify their settings, while the server waits for the LCS to reconnect. After the connection is restored, the server processes the events as

Figure 6: Measured values [%] from the test device with the light sensor during the change of the light condition in the morning

a user requests sequentially, resulting in the real state reflecting the optimistic state again.

4 Testing

The system was implemented according to the above described design and was deployed to a testing server. To verify system functionality, the testing LCS device was created and used (Figure 7).

Figure 7: Testing LCS device, from left to right: 2 photoresistors, Arduino MEGA (GW + light network emulator), led lights (connected to the cables on the right) (Photo by Filip Pružinec)

The system was tested according to the following proposed test scenarios:

A. Selected street lights settings

- 1) Turning on the testing device and waiting for the WS connection to establish

- 2) Changing light intensity on sensor using flash-light
- 3) Loading the web application
- 4) Setting all lights manually from 30% to 100% and back to 30%
- 5) Setting a randomly chosen light to 100%
- 6) Setting selected lights back to Automatic mode.

B. Group settings with manual override

- 1) Turning on the testing device and waiting for WS connection to establish
- 2) Changing light intensity on sensor using flash-light
- 3) Loading the web application
- 4) Setting default group settings to Manual 30%
- 5) Selecting three lights and change their group to a new group
- 6) Setting a new group to Semi-Automatic mode with + 50% Add coefficient
- 7) Setting the default group to Semi-Automatic mode with 50% Multiply coefficient
- 8) Moving two of the remaining three devices from the default group to a new group
- 9) Setting the default group to Automatic mode
- 10) Deleting the new group, while transferring devices to the default group
- 11) Selecting three out of six test lights and setting them to Manual 100%.

C. Dropout testing

- 1) Loading the web application
- 2) Setting the default group to Manual 100%
- 3) Turning on the testing device

Figure 8: Google Chrome DevTools performance test of the web UI

- 4) Setting the default group to Manual 30%
- 5) Turning off the testing device
- 6) Moving three lights into a new group and setting it to Semi-Automatic with Add +50% coefficient
- 7) Selecting one of the new groups lights and setting it to Manual 100%
- 8) Selecting next one of the new groups lights and moving it to the default group
- 9) Turning on the testing device.

During these scenarios, the test LCS device lights intensity was carefully observed. The Automatic and Semi-Automatic behaviour was verified by changing the light condition using a flashlight. Half of the test LED lights were responding correctly to the first sensor and the other half to the second. The system passed the test for dropout behaviour, correctly showing error notifications in the UI and resolving the error state after the LCS successfully reconnected.

The UI performance testing was performed using Google Chrome DevTools (developers.google.com/web/

Table 1: The results of testing of consumption (ILP – installed lamp power, CWC – consumption without control, DLP – default lamp power, CRP – consumption at regulated power)

Time	ILP [W]	CWC [W]	DLP [%]	CRP [W]
20	36	40	50	20
21	36	40	80	32
22	36	40	100	40
23	36	40	100	40
0	36	40	70	28
1	36	40	50	20
2	36	40	50	20
3	36	40	50	20
4	36	40	50	20
5	36	40	70	28
6	36	40	80	32

tools/chrome-devtools). The test results are shown in Figure 8.

The UI is fully loaded in generally 1.5 seconds which is comparable to other similar web map applications such as Google Maps (maps.google.com/) (3.7 s), Mapový klient ZBGIS® (zbgis.skgeodesy.sk/mkzbgis/) (2.6 s) or OpenStreetMap (www.openstreetmap.org/) (1 s).

The results of testing of consumption are shown in Table 1. According to the test results, consumption without control is 40 W, but the average consumption at the regulated power is only 27.27 W, which corresponds to an energy saving of 32% compared to the default settings.

5 Conclusions

The aim of this project was to design a system that helps manage street lights in smaller towns and villages while saving energy. The result is a design of the generally applicable system architecture, but also the proposal for the use of specific technologies suitable for their low-cost implementation. We designed to implement the system as a web application to ensure platform independence, high availability and broad support. During development of the system, requirements for the lower development and initial deployment costs were taken into account. We also recommend using available maps (e.g. OSM) as the base maps in the web-based GIS. We designed the user-friendly UI that requires only simple initial settings or even no settings to set the system to manage the lights.

An important result of this paper is also the complete proposal of the use of specific technologies, which will allow the low-cost implementation of the web-based system. In the presented solution, the system's UI and server components are based on ReactQL kit, written in Javascript with Node.js used as a JS runtime on server side. Data storage is provided by the MySQL database. The LCS communicates with the server via the WS protocol. The server is built on Koa framework and exposes a GraphQL API for the UI using the Apollo GraphQL server. The UI is rendered in the web browser using React and communicates with the server using the Apollo GraphQL client library. GraphQL queries run over HTTP, while the real time communication uses WS based GraphQL subscriptions. The map window embedded in the UI along with the spatial data operations are built with OpenLayers library. Choosing Apollo GraphQL with WS provides a real time communication between UI and LCS via the server component, while React seamlessly keeps the UI up to date with the state of the system. Taking advantage of the OpenLayers web mapping li-

brary, the modern web GIS application is presented to the user.

The system was tested via testing devices and specially designed test scenarios. Based on the test results, the system proved to be suitable for deployment. The results suggest that the proposed web-based street lighting system can reduce energy consumption by 30-40%.

Further development is possible while sustaining the ease of use, or even enhancing it with features such as SMS or email notifications. With higher investment, the system can become even more useful by extending the street light modules with additional sensors (humidity, wind speed, etc.) or by integrating electric car chargers, Wi-Fi network interface or the navigation system. In addition, with further development, initial setup can be automated, the position of the lamp modules can be computed using signal response time and GPS module integration. We assume that such web-based GIS can be a very useful part of smart city energy management.

Acknowledgement: This work was supported by grants no. 1/0300/19 and no. 1/0468/20 of the Slovak Research Grant Agency VEGA.

References

- [1] Mentis, D., Howells, M., Rogner, H., Korkovelos, A., Arderne, Ch., Zepeda, E., Siyal, S., Taliotis, C., Bazilian, M., De Roo, A., Tanvez, Y., Oudalov, A., Scholtz, E., *Lighting the World: the first application of an open source, spatial electrification tool (OnSSET) on Sub-Saharan Africa*, *Environmental Research Letters*, 2017, 12, DOI: <https://doi.org/10.1088/1748-9326/aa7b29>
- [2] Izdihar, R.P., Maryono, M., Widjonarko, W., Rahayu, S., *Street Lighting Infrastructure Assessment Using Discriminant and GIS Method on Mount Merapi Evacuation Road*, *IOP Conference Series: Earth and Environmental Science*, 2018, 123, 012017
- [3] Fichera, A., Gagliano, A., Nocera, F., Pagano, A., Volpe, R., Bisegna, F., *Application of a Geographical Information System to plan energy policy at a neighborhood scale*, 2018, DOI:10.1109/EEEIC.2018.8493723
- [4] Rabaza, O., Molero-Mesa, E., Aznar-Dols, F., Gómez-Lorente, D., *Experimental Study of the Levels of Street Lighting Using Aerial Imagery and Energy Efficiency Calculation*, *Sustainability* 2018, 10, 4365, DOI:10.3390/su10124365
- [5] Nowacka, A., Remondino, F., *Geospatial data for energy efficiency and low carbon cities - overview, experiences and new perspectives*, *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Volume XLII-4, 2018, ISPRS TC IV Mid-term Symposium “3D Spatial Information Science – The Engine of Change”, 1–5 October 2018, Delft, The Netherlands, DOI: <https://doi.org/10.5194/isprs-archives-XLII-4-467-2018>
- [6] Lobão, J.A., Devezas, T., Catalão, J.P.S., *Energy efficiency of lighting installations: software application and experimental validation*, *Energy Reports*, 2015, 1, 110-115
- [7] Butt, M.J., *Estimation of light pollution using satellite remote sensing and geographic information system techniques*, *GI-Science & Remote Sensing*, 2012, 4, 609-621
- [8] Escrivá, D.M.L., Berlanga-Llavori, R., Torres-Sospedra, J., *Smart Outdoor Light Desktop Central Management System*, *IEEE Intelligent transportation systems magazine*, Summer 2018, 2018, DOI: 10.1109/MITS.2018.2806643
- [9] Sung, W.-T., Lin, J.-S., *Design and Implementation of a Smart LED Lighting System Using a Self-Weighted Data Fusion Algorithm*, *Sensors*, 2013, 13, 16915-16939
- [10] Merrino, G., Bruneo, D., Distefano, S., Longo, F., Puliafito, A., Al-Anbuky, A., *A Smart City Lighting Case Study on an OpenStack-Powered Infrastructure*, *Sensors*, 2015, 7, 16314-16335
- [11] Leccese, F., Cagnetti, M., Trinca, D., *A Smart City Application: A Fully Controlled Street Lighting Isle Based on Raspberry-Pi Card, a ZigBee Sensor Network and WiMAX*, *Sensors*, 2014, 14, 24408–24424
- [12] Kumar, A., Kar, P., Warriar, R., Kajale, A., Panda, S.K. *Implementation of Smart LED Lighting and Efficient Data Management System for Buildings*, *Energy Procedia*, 2017, 143, 173-178
- [13] Fielding, R.T., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T., *Hypertext Transfer Protocol – HTTP/1.1.*, RFC Editor, 1999, <http://www.rfc-editor.org/rfc/rfc2616.txt>
- [14] Fette, I., Melnikov, A., *The WebSocket Protocol*. RFC Editor, 2011, <http://www.rfc-editor.org/rfc/rfc6455.txt>.
- [15] Barry, D. K., Dick, D., *Web Services, Service-Oriented Architectures, and Cloud Computing. The Savvy Manager’s Guide*. Second Edition, Elsevier, 2013