SERBIAN JOURNAL OF ELECTRICAL ENGINEERING
Vol. 16, No. 1, February 2019, 45-54

UDC: 007.5:64.06]:004.43 **DOI: https://doi.org/10.2298/SJEE1901045M**

Lighting Control Using Raspberry Pi and Oblo Living Home Automation System

Miloš Milošević[1], Nenad Četić[1], Jelena Kovačević[1], Tihomir Anđelić[1]

Abstract: Home automation systems are bringing comfort and safety into the life of a modern human. They are becoming more popular each day and most of the well-known software companies are fighting to offer their newest solution in this area. *OBLO Living* is a home automation system developed by the scientific-research institute "RT-RK". This paper presents one successful binding of the *Raspberry Pi* computer and the *OBLO living* system. The binding was made using a C++ application developed by the authors of this paper. The developed application is executed in real time on the *Raspberry Pi* platform. It supports a graphical user interface and its purpose is maintaining constant illumination of the room regardless of the daylight.

Keywords: Home Automation, Lighting Control, Oblo Living, Raspberry Pi.

1 Introduction

The *Raspberry Pi* is an inexpensive, fully customizable and programmable small computer with support for a large number of peripherals and network communication. The idea about developing a computer with such characteristics was born in 2005 at the St. John's College, Cambridge. Professor Eben Upton, the Director of Studies had noticed a drop in both, the number and the talent level of incoming computer engineers. The explanation for that phenomenon was found in the fact that technology designers were giving their best efforts to hide all the electrical components from a regular user. Professor Eben decided to design a cheap computer that will allow young people to see what is inside and inspire them to write programs. Since its official commercial release, back in February 2012, the *Raspberry Pi* has found use in many creative solutions. Its low price, compact dimensions and yet mighty hardware has made it a controller of choice in countless systems [2, 3].

[1]RT-RK Institute for Computer Based Systems, Narodnog fronta 23a, 21000 Novi Sad, Serbia;
E-mails: milos.milosevic@rt-rk.com, nenad.cetic@rt-rk.com, jelena.kovacevic@rt-rk.com, tihomir.andjelic@rt-rk.com

Raspberry Pi computers became so popular that, to date, nine different models were developed. These models have evolved throughout four generations. Hardware characteristics of each *Raspberry Pi* board depend on the generation the board was developed in. Nevertheless, all the boards are equipped with a processor and graphics chip, program memory and certain interfaces used for communication with external devices. The most powerful model, *Raspberry Pi 3 Model B+* (Fig. 1), has a 1.4GHz 64-bit quad-core processor with a graphics processing unit, 1GB of program memory and a sharp arsenal of supported communication protocols. Dual-band wireless LAN connection and Bluetooth 4.2 modules, make *Raspberry Pi 3 Model B+* an excellent choice for a wireless controller. The board also has general-purpose input/output (GPIO) pins which allow it to communicate with any nonstandard peripheral. When it comes to the audio and video connections, *Raspberry Pi* offers a 3.5mm composite video port and a standard full-size HDMI port. At the end, there are four standard USB ports [3, 4] .

Having all these connection ports and with the addition of a keyboard, a mouse and an external display, *Raspberry Pi* can operate as an ordinary personal computer.

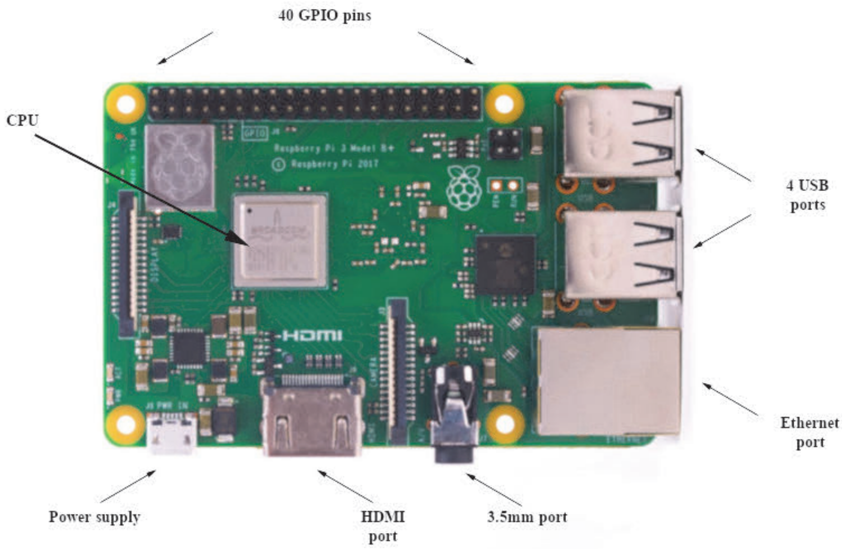


Fig. 1 – *Raspberry Pi 3 Model B+.*

Professor Upton's vision was that *Raspberry Pi* would allow young students to write programs that could control a microwave or manipulate a thermostat; at the same time, numerous companies were working on developing more complex solutions for controlling home appliances. Our laziness and desire for comfort have resulted in the integration of those solutions into a home automation system. A home automating system is a set of subsystems used to control security, surveillance, energy management, climate, lighting and other segments of residential space. Home automation systems mostly have a connection to the Internet and allow remote access and control of the devices in the house. Using a home automation system, the user should be able to set the air conditioning temperature hours before coming home, check if the stove was left turned on or if the door is locked. It is essential that a home automation system is easy to understand and operate with [5].

This paper will present one interesting binding of a *Raspberry Pi* computer and an *OBLO Living* home automation system. *OBLO Living* is one of the home automation solutions currently available on the market and it is developed by the scientific-research institute “RT-RK”. The binding was made using a C++ application developed by the authors of this paper. The developed application is executed in real time on a *Raspberry Pi* platform and its purpose is maintaining constant illumination of the room regardless of the daylight.

2 Oblo Living Home Automation System

From an engineer’s point of view, the *OBLO Living* home automation system is a control system consisted of a vast number of sensors, numerous actuators, a central control unit and supervisory controls. The central control unit, called an *OBLO gateway* or simply gateway, connects all the devices and is the brain of the system. On one side, it is wirelessly connected to all the sensors and actuators. On the other side, it is connected to the Internet and can be accessed by supervisory control units. Gateway gathers all the information from available sensors, runs multiple calculation loops and controls the behavior of all actuators. It stores all the relevant information about the system’s state and its history. The supervisory control units are used for interaction with the users. In the case of *OBLO Living*, supervisory control units are called *OBLO client applications* or simply client applications. They have a human understandable interface and present the information from a gateway in human-readable format. People use client applications to define the behavior of the whole system and set master parameters. Fig. 2 illustrates the relationship of entities in the *OBLO Living* home automation system [5 – 7].

OBLO gateway is the core of the system and any device, that has the access to the gateway’s network, can talk to it. A connected device would have to know the communication protocol the gateway is using and it had to be authorized by the gateway. The communication protocol used by the *OBLO Living* system is the MQTT (*Message Queuing Telemetry Transport*) protocol. It is used both in the communication between the gateway and client applications and between the nodes (which are sensors and actuators) and in the gateway [7, 8].

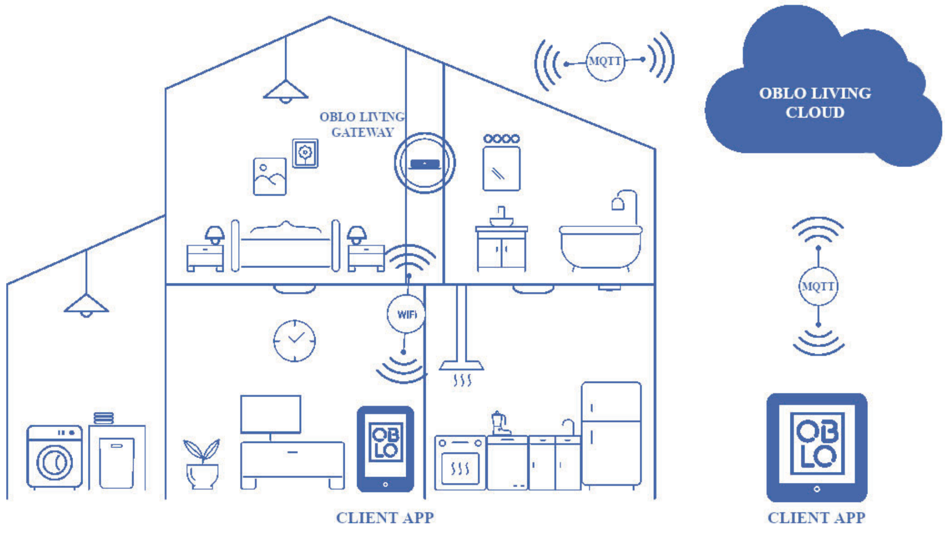


Fig. 2 – *Architecture of OBLO Living home automation system.*

MQTT protocol is a broker-based publishing/subscribing messaging protocol. It is designed to be open, simple and lightweight. These characteristics make it suitable for home automation systems. Any device that communicates using the MQTT protocol can subscribe to a certain topic. From that point on, that device will receive all the messages published on the subscribed topic. Any device communicating using MQTT can also publish a message on an arbitrary topic. All the devices in that system subscribed to that topic will receive the published message. The message format is not predefined and the communicating sides should know how to interpret the message [8].

The architecture of the *OBLO Living* home automation system had a major influence on the architecture of the developed application.

3 Problem Description

Within a home automation system, one of the subsystems is usually in charge of lighting control. This type of a subsystem is called a lighting control system and its purpose is to provide the right amount of light where and when it is needed. The amount of needed light is calculated according to a specified criteria, such as minimal energy consumption or time of the day. Lighting control systems can come both in open-loop and closed-loop modes. The simpler ones are operating in an open-loop mode. An example of an open-loop lighting system is a remote binary switch. When it comes to the advanced ones, they are capable of performing more complex operations, such as turning off the lights in the absence of the people in a particular room. Closed-loop lighting control system most often relies either on the photoelectric sensor or on the presence sensor. According to the information acquired from the sensors, they usually control light bulbs with adjustable illumination know as dimmers [9].

As mentioned before, this paper will present one closed-loop system used for maintaining constant illumination of the room regardless of the daylight. The desired system should use multiple photoelectric sensors for measuring the illumination of the room. The system's controller should be calculating the control signal and it should adjust the dimming level of available dimmers so that the illumination in the room stays constant. Daylight is a disturbance that cannot be controlled. The illustration of the problem's set-up is given in Fig. 3.

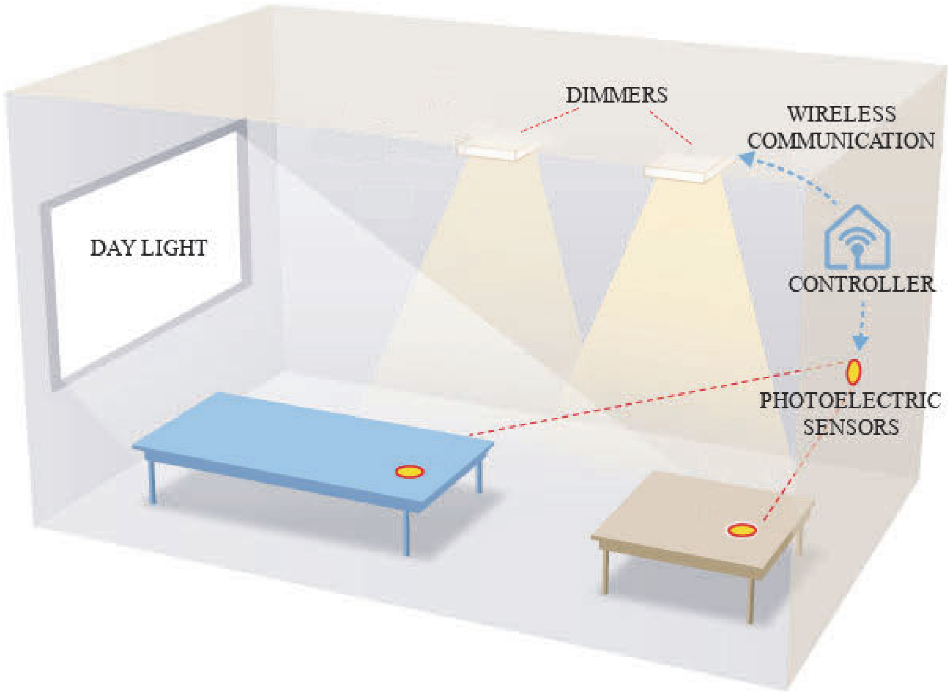


Fig. 3 – *Set-up of the desired system.*

4 Proposed Solution

The described problem can be solved using a *Raspberry Pi* computer as a controller and an *OBLO Living* home automation system as a mediator. The *OBLO Living* system already supports interaction with light sensors and dimmers. The *Raspberry Pi* can be a supervisory unit that will be running a client application which will be communicating with the user on one side and with the *OBLO gateway* on the other side. For the purpose of interaction with the user, the application should offer a graphical interface. The application should support two modes, automatic and manual. In the automatic mode, the user should be able to set the reference for the illumination in the room. In the manual mode, the user should be able to control the available dimmers in an open-loop manner. The application should display current illumination of the room, at all times.

5 Solution Implementation

The proposed application was implemented in the C++ programming language. It is a real-time multi-threaded application run on a *Raspberry Pi* computer. The main focus of the application was the calculation of the control signals used for adjusting the dimmers, but we could easily recognize a few other side tasks such as communication with the *OBLO gateway* or controlling the display. Having that in mind, the reasonable approach was to divide the application into five different modules. Each module has a distinct role and offers an external application programming interface (external API). Modules communicate between each other using only the external API and return values. The modules are:

– Main module,

– MQTT module,

– Display module,

– Control module,

– Storage module.

Main module is the coordinating module. It is responsible for passing all the information between other modules. Its most important role is to initialize all other modules on the application startup. In the case of any irregularity, the main module is responsible for reporting the error and shutting down the application.

MQTT module is responsible for the communication with the *OBLO gateway*. Its API offers a set of calls which is sufficient to serve the needs of all other modules. Also, its role is to listen to the events *OBLO Living* system is publishing and filter the ones that are of interest to the application.

Display module is in charge of the graphical interface and interaction with the user. When the user inputs a new value, it passes it to the main module. When a new measurement of the room illumination is acquired, this module shows it on the screen.

Control module holds the recipe for calculating the control signal. It is triggered each time either reference or measured values changed.

Storage module is used for holding the history of the system. This module was crucial for debugging and testing the system. The relationship between the modules is given in Fig. 4.

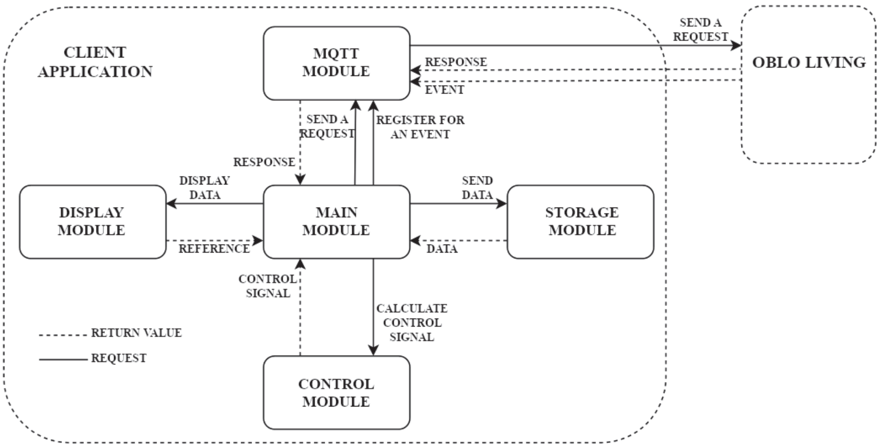
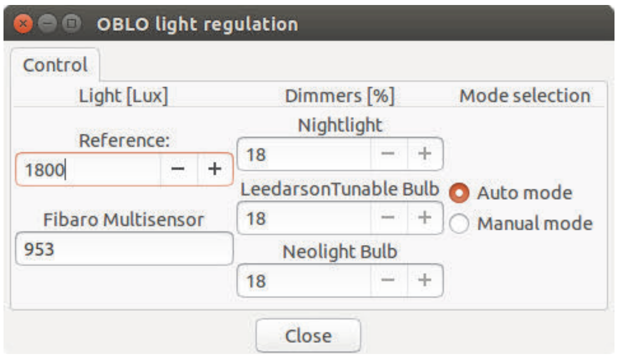


Fig. 4 – *Relationship of the modules in the developed application.*

At the application startup, the MQTT module tries to reach the *OBLO gateway* and authorize itself for communication. In the case of successful authorization, the MQTT module subscribes to all relevant events and retrieves the information about available dimmers and photoelectric sensors. According to the information about available dimmers and sensors, the display module is initialized and the graphical interface is displayed. Fig. 5 shows the graphic interface in the case when one photoelectric sensor and three dimmers were available. After the application setup, the execution is dictated either by user interaction with the display or by the change in the room.


OBLO light regulation

Control

Light [Lux] | Dimmers [%] | Mode selection

Reference: 1800

Fibaro Multisensor: 953

Nightlight: 18

LeedarsonTunable Bulb: 18

Neolight Bulb: 18

[✓ Auto mode] Manual mode

Fig. 5 – *Graphical interface of the application.*

In the manual mode, the application reacts to either a change of the measured illumination value or a user's change of the dimmers' control signal. If the measured value is changed, its value is updated on the screen. If the user inputs a new value for the dimmer's control, the input value is passed to the gateway. The gateway will apply it to the desired dimmer.

In the automatic mode, the application is a bit busier. It also reacts both to the change of the measured illumination value and the user's input. In either case, the values are passed to the control module for recalculation. After the new control signal is recalculated, it is displayed on the screen and passed to the gateway. Again, the gateway will apply it to the desired dimmer.

6 Solution Testing

The fact that the application was divided into multiple modules allowed us to easily include automated testing. A design approach which uses external API as the only cross-modular communication made the solution even more favorable to automated testing and enabled us to include three different levels of it:

– *Unit testing* was used to test modules as separate units. Unit tests were written along with each module's feature and they usually tested methods as standalone entities.

– *Integration testing* was used to confirm that cross-modular interaction was performed in the desired way.

– *System testing* provided information about the overall success rate of the application by comparing set reference and measured values.

Unit tests were run manually after each successful application build. The success rate of unit testing was at a high level. Failure of a unit test helped us in early defect detection and eased defect locating.

Integration tests were also run manually after each successful application build. The number of integration test cases was smaller than the number of test cases used in unit testing, but the failures were more common. Failure of integration test indicated that interaction between modules was broken.

One global system test was run manually after each major code change. Whenever it was possible, a system test was also run during the night and weekends. This test included taking the measure of the ambient lighting, comparing the measured value to the reference and tracking the time needed to achieve the measured value. The success rate of the system test was greatly influenced by the arrangement of objects in the room and by the distance between the dimmer and the sensor. In the dark, the success rate was higher due to the fact that the contribution of dimmers to the measured illumination was greater. In the presentence of dominant daylight, the success rate was lower.

7 Conclusion

This paper presented one enhancement of the *OBLO Living* home automation system. The enhancement was achieved by introducing the feedback loop and using the *Raspberry Pi* computer as an external controller. The developed application was used for maintaining constant illumination of the room regardless of the daylight. Closing the feedback loop enabled user to set the illumination reference and removed the burden of manual control every time the outside conditions change. With minor upgrades, the application could also be used for regulation in other areas of home automation.

As the daylight changing process is slow, the application had no problems compensating the change. In the case of step changes of either the reference or the measured signal, it took around a minute for the system to return to the steady state.

Limitations of the system were dictated by the room setup. The measured value could not be below the ambient illumination. The range of achievable values could be spread by introducing electrically controlled curtains or by adding more light bulbs that are dimmable. The curtains could help lower the ambient illumination and the dimmers could help raise maximal achievable illumination.

As a test, the application was continuously run for a week. As no error was reported, we can consider that the implementation was successful.

8 Acknowledgment

This research is a part of investigations performed within the scope of project TR-32029. The research was partially funded by the Serbian Ministry of Education, Science and Technological Development, which we are grateful for.

9 References

[1] M. Milošević, N. Četić, J. Kovačević, T. Anđelić: Lighting control in smart buildings, Proceedings of the 62nd Conference on Electronics, Telecommunications, Computers, Automatic Control and Nuclear Engineering (ETRAN 2018), Palics, Serbia, June 2018, pp. 159 – 162. (In Serbian)

[2] C. Severance: Eben Upton: Raspberry Pi, Computer, Vol. 46, No. 10, October 2013, pp. 14 – 16.

[3] V. Vujović, M. Maksimović: Raspberry Pi as a Sensor Web Node for Home Automation, Computers & Electrical Engineering, Vol. 44, May 2015, pp. 153 – 171.

[4] Raspberry Pi 3 Model B+, Raspberry Pi Foundation, December 2018,

Available at: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

[5] K. Gill, S.- H. Yang, F. Yao, X. Lu: A ZigBee-Based Home Automation System, IEEE Transactions on Consumer Electronics, Vol. 55, No. 2, May 2009, pp. 422 – 430.

[6] C. Gomez, J. Paradells: Wireless Home Automation Networks: A Survey of Architectures and Technologies, IEEE Communications Magazine, Vol. 48, No. 6, June 2010, pp. 92 – 101.

[7] OBLO Living, OBLO Living LLC, December 2018,
Available at: http://www.oblolivng.com/

[8] K. Tang, Y. Wang, H. Liu, Y. Sheng, X. Wang, Z. Wei: Design and Implementation of Push Notification System Based on the MQTT Protocol, International Conference on Information Science and Computer Applications (ISCA 2013), Tel-Aviv, Israel, October 2013, pp. 116 – 119.

[9] M. Miki, T. Hiroyasu, K. Imazato: Proposal for an Intelligent Lighting System, and Verification of Control Method Effectiveness, IEEE Conference on Cybernetics and Intelligent Systems, Singapore, Singapore, December 2004, pp. 520 – 525.