

Ernestas Juškevičius

# Smart home lighting system using IoT technologies

Bachelor's thesis

Information technology

Double degree



South-Eastern Finland  
University of Applied Sciences

<b>Author (authors)</b>	<b>Degree title</b>	<b>Time</b>
Ernestas Juškevičius	Bachelor of Engineering	April 2021
<b>Thesis title</b>		
Smart home lighting system using IoT technologies		49 pages 2 page of appendices
<b>Commissioned by</b>		
<b>Supervisor</b>  Timo Hynninen		
<p><b>Abstract</b></p> <p>The aim of this thesis was to use various web development, Internet of Things (IoT) and software development technologies to create a smart home lighting system prototype. System's prototype development using more affordable hardware and software appliances to control home 12 Volt lighting devices without abundance of features, while maintaining the aspect of continuous common communication between devices of IoT systems.</p> <p>The thesis is structured into three main content parts: Part 2, background research for hardware and software components, programming languages as well as Web system and IoT system security issues, other technologies used in development of such management systems. Part 3, system's hardware and software subsystem specification. Part 4, system's and its components design model, results and inspection of implemented prototype.</p> <p>The development of the project showed that IoT systems use a vast variety of different technologies, including many variables when generating use cases for wanted results. The created smart home lighting system prototype was a successful, cheaper and simpler alternative to the current smart home management system consumer market, however, it lacks the plug-and-play nature of such expensive and less accessible systems like Apple Homekit or Fibaro.</p>		
<b>Keywords</b>  Internet of Things, web system, single-board computing		

## **CONTENTS**

1 INTRODUCTION .....	5
2 RESEARCH .....	7
2.1 Task analysis .....	7
2.2 Internet of Things technology .....	7
2.3 Apache2 and Nginx Web servers .....	8
2.4 Single-board computers and microcontrollers .....	9
2.5 Programming technologies used to develop smart home management system.....	11
2.6.1 Web server back-end programming languages .....	11
2.6.2 Web server front-end programming languages.....	12
2.6.3 Programming languages for single-board computer and microcontroller tasks and data transfer .....	13
2.7 Security of web based IoT systems.....	14
3 SPECIFICATION .....	15
3.1 Projected object .....	15
3.2 Projected object functions .....	15
3.3 Requirements for subsystems of the designed object.....	15
3.3.1 Requirements for hardware subsystem.....	15
3.3.2 Requirements for the software subsystem and user interface .....	16
4 PROJECT DESIGN .....	17
4.1 Smart home lighting systems prototype specification, structure and its components.....	17
4.2 Raspberry Pi 3 B+ configuration and software installation .....	19
4.3 Database data model .....	20
4.4 Lighting unit and Arduino microcontroller design.....	23
4.5 Data communication between mariaDB database server and Arduino mega 2560 R3.....	28
4.6 Graphic user interface (GUI) model .....	31
4.6.1 System's logical model.....	31
4.6.2 User interface structure.....	36
4.7 Software architecture design.....	39
4.8 PROJECT RESULTS .....	43
5 CONCLUSIONS .....	47
REFERENCES.....	48

## Table of figures

Figure 1 Web Server software usage statistics (W3techs, 2021) .....	8
Figure 2 Usage statistic of server-side programming languages for websites (W3techs 2021) .....	12
Figure 3 Web application attack type statistics (Ptsecurity 2017) .....	14
Figure 4 System's principle scheme .....	18
Figure 5 Raspberry Pi's Raspbian OS Terminal .....	20
<i>Figure 6 System's DFD-0 diagram of data flows</i> .....	21
Figure 7 Database table „Logins“ structure .....	22
Figure 8 Database table „JudesioData“ structure .....	22
Figure 9 Database Table „ledBrightness“ structure .....	23
Figure 10 Lighting unit and Arduino microcontroller connection scheme.....	23
Figure 11 IRFZ44N MOSFET electrodes .....	24
Figure 12 HC-SR501 PIR motion sensor.....	25
Figure 13 Pin declaration.....	26
Figure 14 ASCII value assignment to variables .....	27
Figure 15 microcontroller's logical operation code .....	28
Figure 16 Software subsystem dataflow scheme .....	29
Figure 17 Python code „connect.py“ .....	29
Figure 18 Python code „ardtorp.py“ .....	30
Figure 19 System's Use Case Diagram.....	31
Figure 20 Login to the system activity diagram .....	33
Figure 21 Lighting unit parameter selection use case diagram .....	34
Figure 22 Lighting unit information review activity diagram .....	35
Figure 23 Logout from the system activity diagram .....	36
Figure 24 Login web page structure .....	36
Figure 25 Main page structure.....	37
Figure 26 Information review web page structure.....	39
Figure 27 System's software module diagram.....	40
Figure 28 Deployment diagram .....	42
Figure 29 Implemented Login page .....	43
Figure 30 Implemented Main page .....	44
Figure 31 Implemented Information review page .....	44
Figure 32 Executed „connect.py“ Python script .....	45
Figure 33 Executed „connect.py“ script with database record data print .....	45

## 1 INTRODUCTION

The contents of the chapter resolve around the relevance of the topic, research problem, work's aim and objective as well as workflow.

**Relevance of the topic.** Internet of Things enables various objects and devices that are used in daily life, automating them to transmit data and information over the network without requiring human supervision, thus facilitating the daily tasks of a modern-day man while reducing the resource use and providing him with valuable information while saving valuable time. IoT can be met almost anywhere, whether walking down the street through a smart pedestrian crossing, looking for a place to park a car while looking at a smartphone with an app that provides information on the status of a smart parking lot vacancies; these are some of the fields that help our daily lives. One of those fields is home automation. Smart home appliance systems allow the user to relax from their worries by connecting devices and objects to a network, controlling and managing them.

**Research problem.** Smart home management systems on the market are often presented as multi-functional, so their prices are high, management requires specific skills and access to the graphical user interface often requires platform specific applications. IoT and web development technologies could be used to create a system which would be accessible via a web browser from any device with internet connection to the private network to control and change parameters of the selected specific home appliances at the fraction of the cost.

**Objective.** Design and create a home lighting system prototype using Internet of Things and web development technologies, which would allow the user using a web browser through a private network to

change selected parameters and track information concerning lighting accessory operation and ability to track rough estimate of calories burned while using the stairs.

### **Thesis workflow.**

Research the current home automation system market, find out their advantages and disadvantages. Analyze web development and Internet of Things project programming languages as well as most occurring Web application security breaches. Select appropriate software and hardware technologies for the prototype. Setup a Web server and database for storing and sharing information between devices on the local network and management system components. Design and build a controlled lighting unit for a smart home lighting system. Prepare the single-board computer and microcontroller used in the control system prototype for seamless interaction and common communication. Implement a graphical user interface. Inspect the system's prototype.

## **2 RESEARCH**

Research chapter focuses on the task analysis, IoT technology implementation and definition for home appliance systems, various web development technology element analysis, hardware and software comparison and research used in both large and small scale IoT projects as well as commonly occurring web system security breach analysis.

### **2.1 Task analysis**

This thesis describes the development of a home lighting management system prototype using IoT technologies. The system requires the use of a variety of software and hardware technologies and resources to make it easy to use. The control system will be available to the user using a web browser on a computer or smartphone. The system prototype would allow the user to control unique, system-designed devices that can be controlled digitally and see their activity status in a graphical environment.

The prototype will use a variety of programming and scripting languages like: HTML5 hypertext markup language, CSS cascading template language, PHP hypertext dynamic interpretable programming language, Python3 interactive, open source programming language and various libraries for it, Arduino IDE microcontroller programming environment.

The system will be hosted on a physical single-board-computer, which will have a configurable web server. The connected user will be able to control the agreed and prepared lighting devices in the house. The system will be accessible online through a private network, however in order to protect the system from external malware, it is necessary to protect the database from SQL injections, eliminate unwanted connections and ensure data encryption. The microcontrollers of the managed and under development devices will be connected to single-board computer with a USB cable for data transfer.

### **2.2 Internet of Things technology**

Internet of Things (IoT) is an information technology structure that encompasses a network of interconnected objects that enables the transmission of information over the network without the need for human-computer interaction. A person with a heart monitoring implant or cattle with an injected ID chip are a great example of IoT in our lives (Alexander S. Gillis 2019).

In the consumer market for home automation, IoT technologies are reflected in smart home systems like Fibaro or Apple HomeKit, including devices such as:

- Thermostats
- Luminaires

- Security cameras
- Smart wall plugs
- Humidifiers

These devices in the home appliance management system can be controlled by other compatible devices on the network, such as Apple smartphones with the iOS operating system. However, the above-mentioned home systems are expensive due to the abundance of integrated functions, which are difficult to adapt for simpler home management.

### 2.3 Apache2 and Nginx Web servers

The implementation of a smart home system will require the use of a web server, which will be configured for the user's graphical interface where back and front end programming will be performed. When comparing Apache and Nginx servers, the following requirements are taken into account:

- Source code nature
- Easily accessible usage documentation
- User community scale
- Security

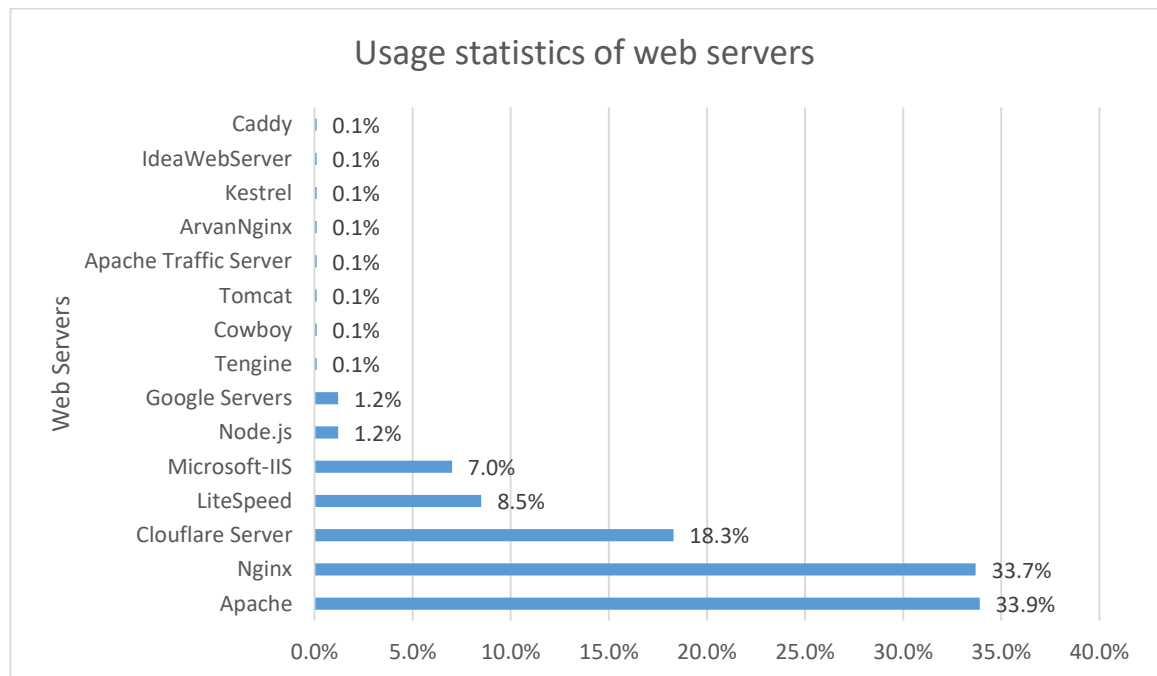


Figure 1 Web Server software usage statistics (W3techs, 2021)



**Apache** is an open source, HTTP web server for web development. According to Figure 1, Apache is used on 34% of all known websites. A web server that allows administrators to implement and publish websites, applications, or systems with a variety of content. It has official use and safety documentation and an extensive user-prepared documentation. Like many other software systems, the Apache web server has many security vulnerabilities that can be avoided by considering operating system preparation and proper software configuration (Krishnamurthy et al. 2008).

**Nginx** is an open source, HTTP web server, and a proxy server designed to compete with Apache, using an asynchronous, connection management algorithm. According to Figure 1, the server is used by 33.7% of all known sites. It handles static content more efficiently than Apache. Has official documentation.

## **2.4 Single-board computers and microcontrollers**

### **Single-board computers**

Single-board computers (SBCs) have integrated all or most of a computer's components. Light weight, compact size, often even more reliable than computers with composite boards and components. Because of these features, they are great for building IoT projects and systems (A. Gomez et al, 2015).

A smart home system can be designed and set up on its own, physical server, computer, or virtual private server (VPS). Since the system will be implemented in a local area network, a suitable single-board computer must be selected. When analyzing these devices, the following characteristics are taken into account (Table 1):

- Device price
- Computer size
- Processor computing power
- User community size
- Random Access Memory quantity
- Integration of a Wi-Fi module.

Table 1 Single-board computer characteristics

SBC	Characteristics					
	Price, eur.	Size, cm.	Processor	Community size	RAM	Wi-Fi module
Raspberry Pi 4 B	€63.00	8.6 x 5.7	1.5 GHz	High	2 GB	Yes
Raspberry Pi 3 b+	€49.90	8.5 x 5.6	1.4 GHz	High	2 GB	Yes
Orange Pi One	€25.50	6.8 x 4.2	1.2 GHz	Average	512 MB	No
ODROID XU4	€89.00	8.3 x 5.8	1.4 GHz	Low	2 GB	No

### Single-board microcontrollers

Single-board microcontrollers (SBMs) are controllers integrated in a single board that have all the components necessary for a control task: a microprocessor, digital input and output connectors and RAM. With their help, it is possible to design and control various devices such as motion sensors, magnetic field and sound sensors used in Internet of Things technology (Güven, Yılmaz et al 2017).

When developing devices controlled by the smart home system, and ensuring that new devices can be added to the system if necessary, these microcontrollers are necessary. Therefore, it is important to choose appropriately, when analyzing the devices and the following characteristics are taken into account (Table 2):

- Device price
- User community size
- The number of analog input ports
- The number of digital I/O ports
- Flash memory size
- Serial interface availability.

Table 2 Microcontroller characteristics

SBM	Characteristics					
	Price	Community size	Number of analog input ports	Number of digital I/O ports	Flash memory	Serial interface
Arduino Mega 2560 R3	€35.00	High	16	54	256 KB	USB
Arduino Uno Rev3	€20.00	High	6	14	32 KB	USB
Bluno Mega 1280	€52.40	Low	16	54	128 KB	UART
Adafruit Feather 32u4 Bluefruit	€44.70	Average	10	20	32 KB	USB

Each device has its pros and cons and is great for projects and systems of a relatively large scale, however, some devices offer an abundance of hardware components and computing power, increasing their cost.

## 2.5 Programming technologies used to develop smart home management systems

There are several programming languages and technologies that are used to develop smart home management and other IoT systems, to perform front-end and back-end programming, single-board computer work, microcontroller operations and tasks.

### 2.6.1 Web server back-end programming languages

Back-end programming consists of the functionality of a website, depending on the code of the program, the information is provided or transmitted on the website and stored in a database. When choosing a back-end programming language, following characteristics are taken into account:

- User community size
- Web server software support
- Maintained and supported versions
- Open source nature
- Security

Based on statistics from 2021 on the use of server-side programming languages (Figure 2), the top three are analyzed.

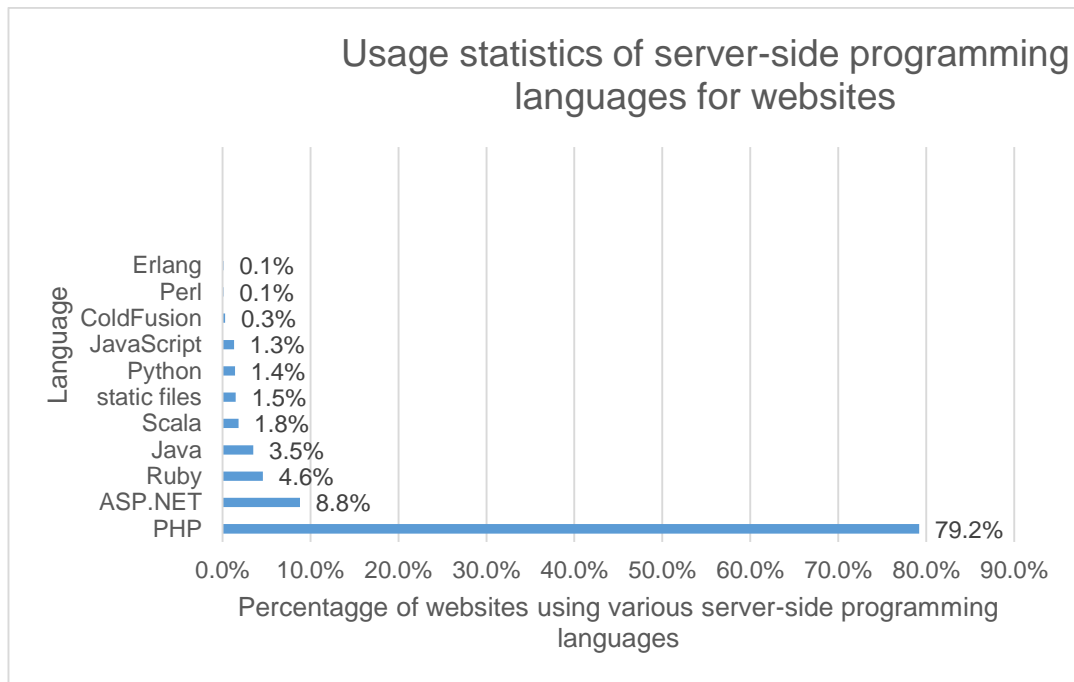


Figure 2 Usage statistic of server-side programming languages for websites (W3techs 2021)

Table 3 Top 3 server-side programming language characteristics

Programming language	Characteristics				
	Open source nature	Security	Maintained and supported versions	Web Server support	User community size, %
Php	Yes	High	7.3-8.0	Nginx and Apache	79.2
ASP.NET	Yes	High	< 5.x	Nginx	8.8
Ruby	Yes	High	2.6-3.0	Nginx and Apache	4.6

Php is by far the most used back-end Web Server programming language, and according to Table 3, is an open source nature while maintaining a variety of security features.

### 2.6.2 Web server front-end programming languages

External programming consists of elements that the user sees, including the design of the site and the content, interactive elements, or applications contained therein. Using these languages, a graphical user interface will be implemented that will provide managed elements to control devices in the system and allow user to see presented information about the device usage.

**HTML:** hypertext markup language for displaying content on a web site.

**CSS:** A cascading style template language for displaying content on a web site.

### **2.6.3 Programming languages for single-board computer and microcontroller tasks and data transfer**

It is necessary to choose the appropriate programming language in order to script computer and microcontroller tasks, transfer data between them. The following requirements are taken into account:

- Easily understandable syntax
- Open source nature
- User community size and available documentation
- Supported version

**Python** – open source syntax with similarities to written English language. Supports countless libraries created by official and users, performing various tasks. Used in various, small and large-scale projects or applications, perfect for IoT technology projects using Raspberry Pi devices (Dow, 2018). Latest released version is 3.9.2

**Perl** – open source, often used in important banking systems, network communication systems and devices. Syntax has similarities with many languages like: Awk, sed, C, but it is for this reason that the syntax is considered more complex. Perl has Python language features, but is seen as declining in competition with a newer edition with a larger user community of programming and scripting languages. Latest released version is 5.24.0

**Arduino IDE** – open source programming environment for writing work scenarios for Arduino single-board microcontrollers. Uses C and C programming language features and syntax. Supports many libraries for various tasks. Available for Windows, macOS, and Linux operating systems. Latest released version is 1.8.13

## 2.7 Security of web based IoT systems

Systems accessible via the Internet have countless loopholes that can be exploited by various attackers. An unprotected smart home system can cause extreme problems for the user as IoT devices and systems usually contain sensitive and/or private information. Considering an attack would happen at a Smart home system, the unauthorized attacker intervening in the system could use the information for its own needs and manage the devices in the system at their own accord. This can be assumed to endanger the health or life of the authorized system user. It is therefore necessary to protect the system against unwanted connections and other attacks that harm the system or its database. Based on Figure 3 Figure 1 statistics, some of the most common attacks directed at web systems are XSS (cross-site scripting), SQL injection, and Path Traversal.

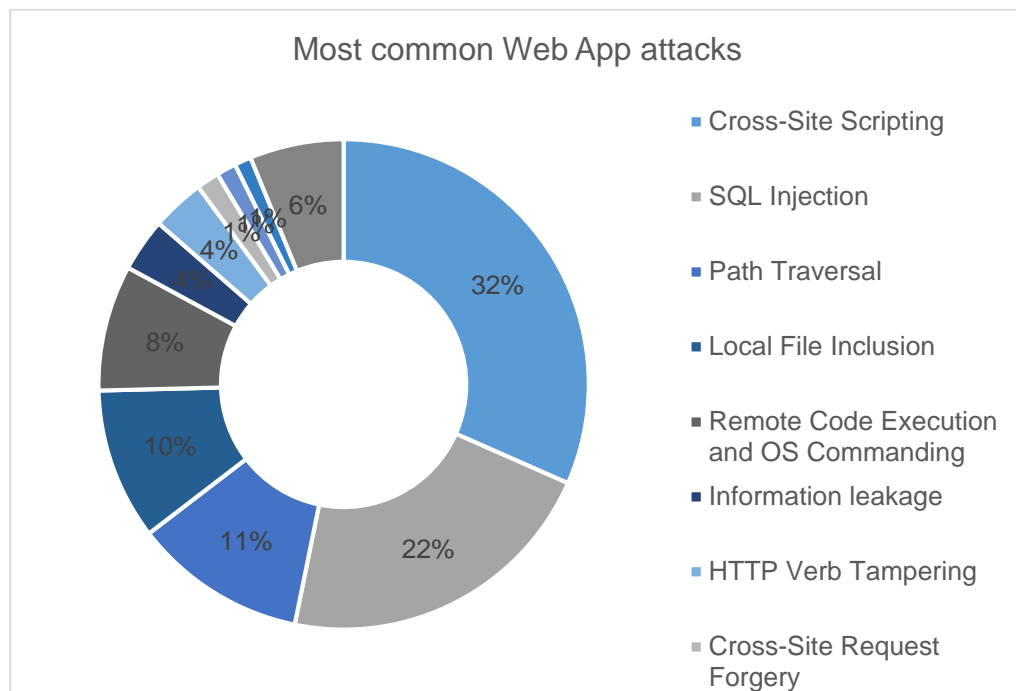


Figure 3 Web application attack type statistics (Ptsecurity 2017)

XSS or Cross-site scripting exploits web system by exploiting user interactions by returning malicious JavaScript to the users, executing the script inside its browser. In order to prevent at least some of the attacks, user input and actions should be filtered according to the actions that are necessary for the system to complete its tasks.

When it comes to SQL injection attacks, they usually target Web pages and web systems that use structured query language software to use databases and transfer information to a web page. The attack exploits the user input field on the page to send the SQL query (Ramoška, 2012). In this way, the attacker tries to

bypass the structure of the system and cause damage or access data for which it is not authorized. This type of attack is usually aimed at misappropriating information in database tables or selfish editing. SQL injections can be prevented by filtering all user input data in the website, not using database with administrator privileges as it might give the attacker full access to the database server, encrypt sensitive data, use prepared SQL query statements.

### **3 SPECIFICATION**

Chapter specifies hardware and software requirements for the development of the system as well as project's features.

#### **3.1 Projected object**

The purpose of the designed prototype is the control of a 12 V lighting unit using a web browser. Log in availability to the management system using a web browser. After logging into the system website, the user of the prototype – an authorized person whose house is equipped with the select lighting unit will be able to control its brightness and light up delay parameters, see its operational statistic information about it. The lighting unit connected to the microcontroller and designed will automatically illuminate the steps of the stairs of the house using motion sensors. Information between the microcontroller and the server will be transmitted via a USB serial port.

#### **3.2 Projected object functions**

Requirements for the functions of the developed system prototype:

1. Controlling brightness and light up delay time of LED strips through a graphical user interface implemented on a web server.
2. Rendering of information from a database server on a Web server site.
3. Accumulation of the lighting device's operation information into the database and its rendering on the website of the Web server.
4. Lighting systems prototype user authorization using logins.
5. Storage of information in a database.

#### **3.3 Requirements for subsystems of the designed object**

##### **3.3.1 Requirements for hardware subsystem**

Requirements for the technical characteristics of single-board computer:

- CPU: 1.4ghz 64bit
- Micro SD memory card: 16GB (at least)
- RAM: 1GB (at least)
- Ethernet port or Wi-Fi module

To take system's continuous and persistent operation and smooth user's experience into account, the SBC's hardware requirements are appropriately selected.

Requirements for the technical characteristics of microcontroller:

- High digital I/O port number
- USB serial connector support
- 128 KB flash memory (minimum)
- 5 V operating voltage

The amount of I/O ports ensures that the system could be scaled up if needed, more devices or sensors could be connected. Larger and more complex operational scripts could be implemented, thus, a bigger flash memory capacity is taken into account.

### **3.3.2 Requirements for the software subsystem and user interface**

- Raspbian buster operating system with graphical user interface
- Chromium, open source, low-resource web browser
- Apache Web Server
- MySQL type database server
- Text editor, for programming tasks

For smoother developing and maintenance experience, the software subsystem requirements are appointed, ensuring low resource usage and more efficient workflow.



## 4 PROJECT DESIGN

This chapter focuses on home lighting management system's software and hardware subsystem component design.

### 4.1 Smart home lighting system's prototype structure and its components

Based on the carried out background research and the specification, the following key components are chosen for the development of the smart home lighting system's prototype:

- **Raspberry Pi 3b 2GB single-board computer with 16GB memory micro SD card.**

Technical characteristics are suitable for use in small systems, the resources available in this SBC are sufficient to support the project web server and perform information transmission tasks, knowing that the system will be used by a single user.

- **Arduino Mega 2560 R3 single-board microcontroller.**

The selected microcontroller has 54 digital I/O connections, so the designed system will be able to install new managed devices if necessary, and the relatively large Flash 256KB memory of this microcontroller will ensure that the programmable script code will not run out of space. Due to the USB connection used in this microcontroller, it will not require a separate power supply, as power will be provided via USB. Arduino devices provide a great opportunity to develop various IoT technology projects at a low cost (Monk, 2016).

- **Apache2 Web Server.**

Apache web server was selected due to extremely detailed, official usage documentation (Httpd.apache, 2021), and large user community. Although not as efficient as Nginx, the Apache web server uses a generally small amount of resources and is highly configurable.

- **MariaDB database server.**

This database is based on the MySQL database. It is supported and updated to strengthen it against security breaches. The MariaDB database is supported by Raspbian Buster operating systems, which uses a small amount of system resources, making it perfect for use on small systems while maintaining stability.

All Web server front-end and back-end processes are written using HTML, CSS and PHP programming languages. All Raspberry Pi single-board computer and Arduino microcontroller data communication is scripted using Python and its libraries. All tasks are written with “GNU NANO” text editor used in Raspbian Buster and other UNIX type operating systems. This text editor is not modern by today’s standards, but is readily available for use in the Raspberry Pi command line while depleting minimal amount of resources, therefore there are no interferences during programming.

Arduino microcontroller tasks are scripted using Arduino IDE, which merges functions from both “C” and “C++” programming languages.

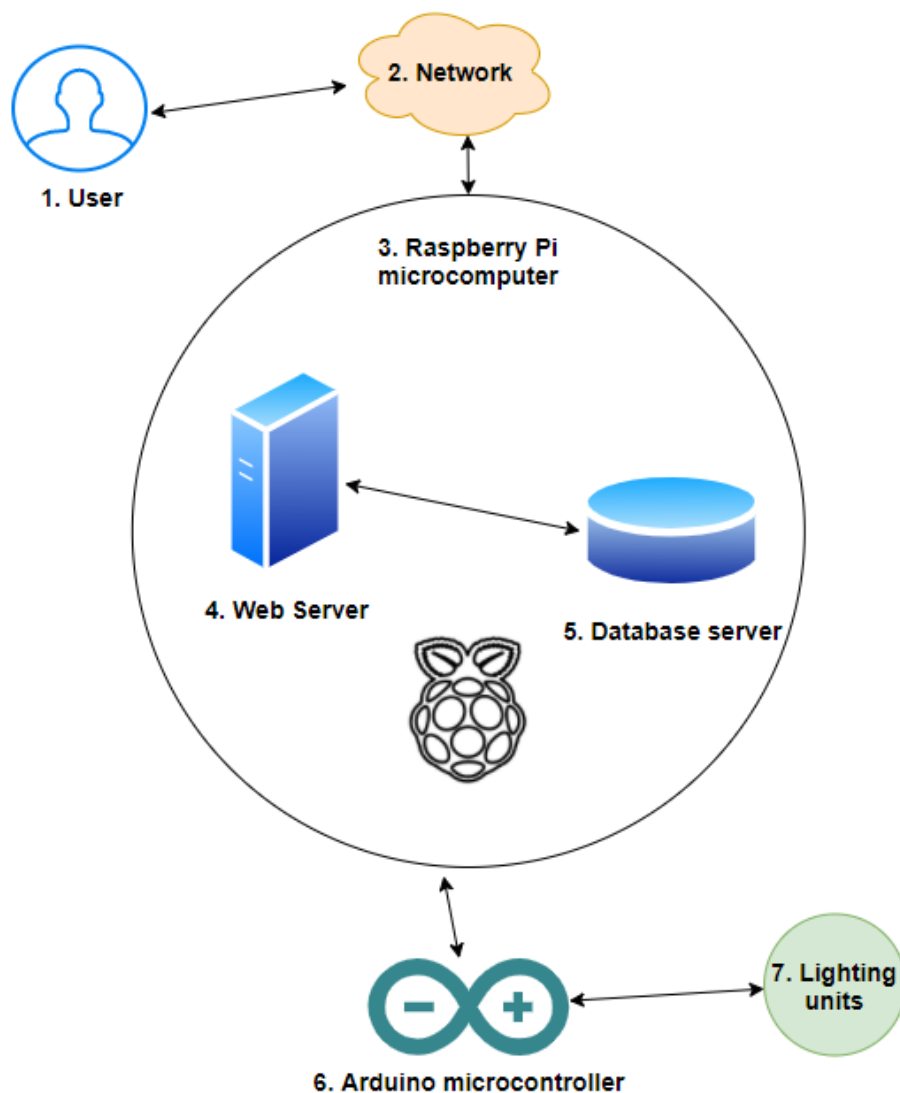


Figure 4 System's principle scheme

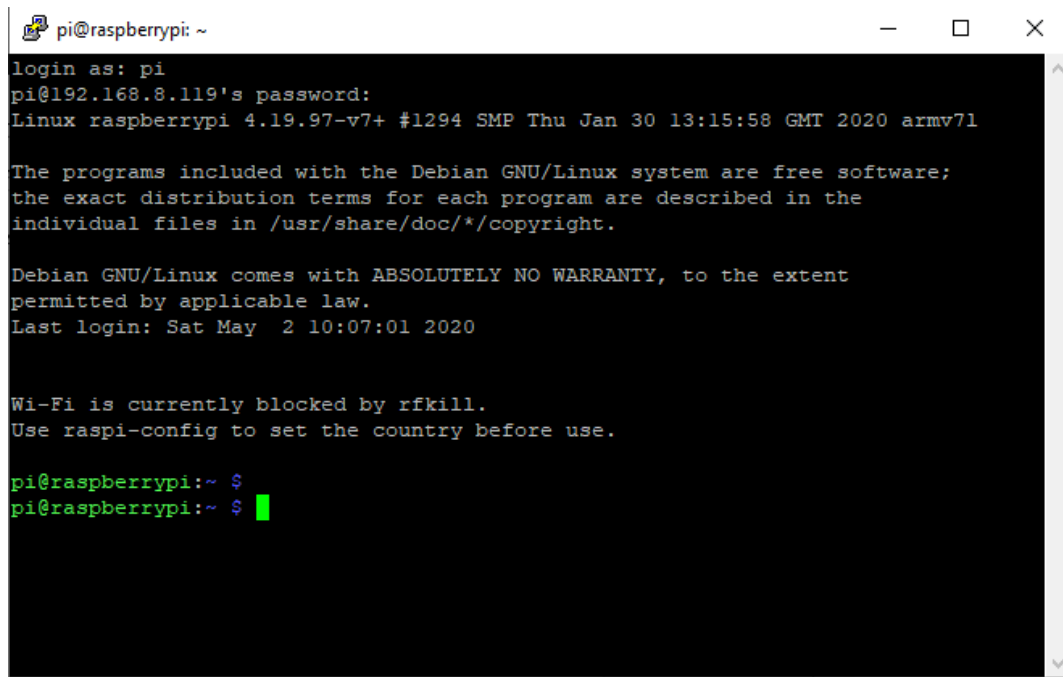
According to Figure 4, in the designed smart home lighting system, each component will transmit information to each other using various technologies for common communication. The user will need a smart device that could browse the web to access the system. Raspberry Pi will act as a host to both Web and Database servers. It will both collect and send digital data over the USB serial cable to the Arduino microcontroller.

Principle scheme components:

1. **User:** authorized system user who controls available parameters for the lighting unit.
2. **Network:** local area network where system prototype is implemented.
3. **Raspberry Pi single-board computer:** device on which software is installed and programmed: web server, database, software for communication between the single-board computer itself and the microcontroller in the system. A USB port is used for physical communication between the controller and computer.
4. **Web Server:** software to support the system and graphical user interface.
5. **Database server:** software to store information about the system, manageable lighting device parameters, user login data.
6. **Arduino microcontroller:** SBM used to read inputs and turn them into output signals.
7. **Lighting units:** Various devices for lighting, in this case, 12 V LED strips.

## 4.2 Raspberry Pi 3 B+ configuration and software installation

For the implementation of the project, the single-board computer requires the installation of the Raspbian buster operating system, which is based on Linux Debian. Therefore, the list of commands running on the terminal corresponds to other distributions of Debian operating systems. Using the mentioned terminal further software installation and configuration work is performed (Figure 5).

A screenshot of a terminal window on a Raspberry Pi. The window title is 'pi@raspberrypi: ~'. The terminal output shows the login process for user 'pi' at IP '192.168.8.119'. It displays the Linux version '4.19.97-v7+', the date and time 'Thu Jan 30 13:15:58 GMT 2020', and the architecture 'armv7l'. It also shows the Debian GNU/Linux license notice, the warranty disclaimer, the last login time 'Sat May 2 10:07:01 2020', and a message that Wi-Fi is blocked by rfkill. The prompt 'pi@raspberrypi:~ \$' is shown twice, with a green cursor on the second line.

```
pi@raspberrypi: ~
login as: pi
pi@192.168.8.119's password:
Linux raspberrypi 4.19.97-v7+ #1294 SMP Thu Jan 30 13:15:58 GMT 2020 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat May 2 10:07:01 2020

Wi-Fi is currently blocked by rfkill.
Use raspi-config to set the country before use.

pi@raspberrypi:~ $
pi@raspberrypi:~ $ █
```

Figure 5 Raspberry Pi's Raspbian OS Terminal

Advanced Package Tool (APT) is used inside the terminal in order to download and install needed software packages. The followed command is typed in the command line: **sudo apt-get install *selected package name***.

The following software packages will be installed on the Raspberry:

- Arduino IDE – official Arduino microcontroller programming environment.
- Apache2 – Web Server.
- MariaDB – Database server.
- Php – Php programming language compiler.
- Python-mysql.connect – python programming language library for MySQL database connection initialization.
- phpMyAdmin – database graphical user interface.

Once the select packages are installed, a data model can be projected for our mariaDB database server using the phpMyAdmin graphical user interface on a web browser.

### 4.3 Database data model

In order to implement common communication between systems components, it is necessary to ensure that the database server is designated to:

1. Store user login data that will allow them to log in to the system using a web browser.
2. Store information for changing the settings of a controlled lighting unit. This data is updated when the user sends the query using the graphical interface.

3. Store data intended to provide the user with information on the frequency of use of the controlled lighting unit in the graphical interface.

When using a database, efforts are made to avoid duplication of data (data redundancy). However, all tables in the database and their entities are not dependent on each other. The purpose of the tables is to implement a platform where various data could be transmitted between the system components and rendered on a graphical user interface for the user.

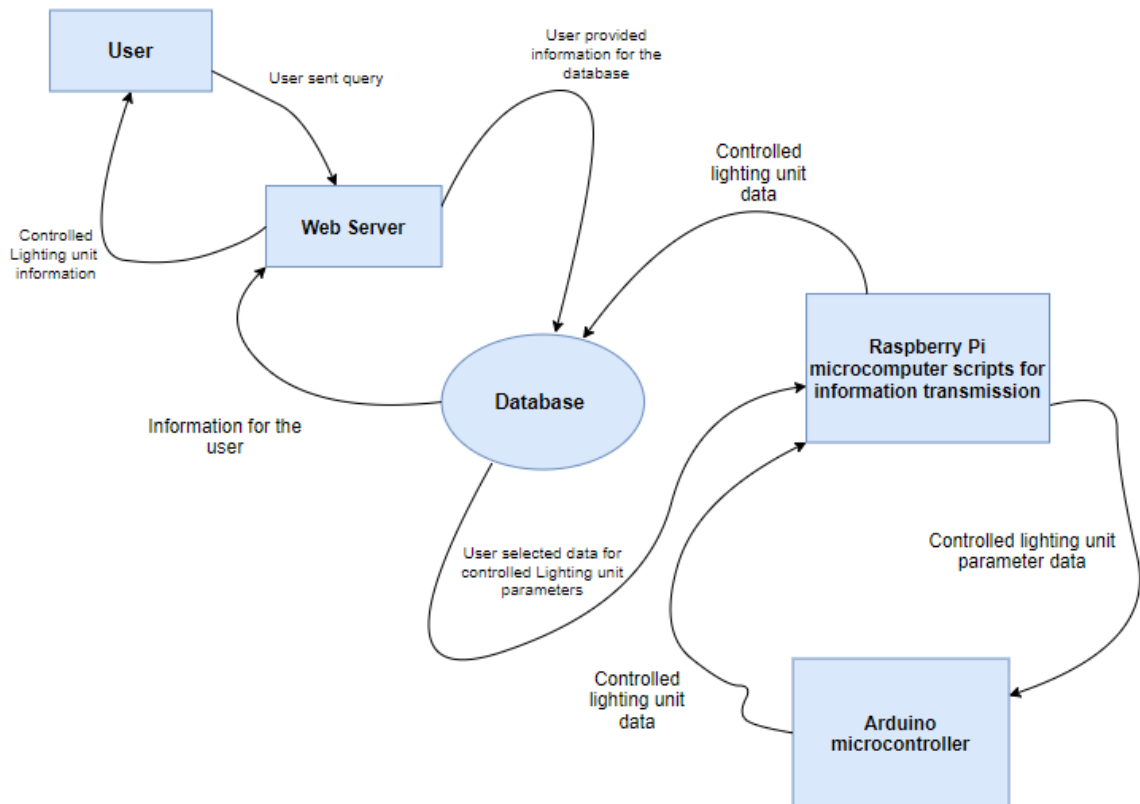


Figure 6 System's DFD-0 diagram of data flows

### Database normalization tables and their structure

Database tables are created accordingly to the data flow diagram (Figure 6).

## Table “Logins”

The "Logins" table used in the database is designed to store data for a user's login to the system. This will ensure the use of the authorized person system. Fields that make up the table (Figure 7):

1. “userID” – Primary key
2. “Username” – unique key, users name
3. “Password” – pre-generated password for the user to use while logging into the system

#	Pavadinimas	Tipas	Palyginimas	Atributai	Null	Nutylint	Komentari	Papildomai	Veiksmai
1	userID	int(11)			Ne	Jokio		AUTO_INCREMENT	Redaguoti Šalinti Pirminis Unikalus Indeksas Spatial Daugiau
2	Username	varchar(111)	utf8mb4_general_ci		Ne	Jokio			Redaguoti Šalinti Pirminis Unikalus Indeksas Spatial Daugiau
3	Password	longtext	utf8mb4_general_ci		Ne	Jokio			Redaguoti Šalinti Pirminis Unikalus Indeksas Spatial Daugiau

Figure 7 Database table „Logins“ structure

## Table “JudesioData”

The purpose of the table is to store information about the operation of the controlled lighting unit. Fields that make up the table (Figure 8):

1. “judesioID” – primary key
2. “Aptiktas” – operation description field
3. “Laikas” – automatic timestamp of the record.

#	Pavadinimas	Tipas	Palyginimas	Atributai	Null	Nutylint	Komentari	Papildomai	Veiksmai
1	judesioID	int(11)			Ne	Jokio		AUTO_INCREMENT	Redaguoti Šalinti Pirminis Daugiau
2	Aptiktas	varchar(111)	utf8mb4_general_ci		Ne	Jokio			Redaguoti Šalinti Pirminis Daugiau
3	Laikas	timestamp			Ne	current_timestamp()		ON UPDATE CURRENT_TIMESTAMP()	Redaguoti Šalinti Pirminis Daugiau

Figure 8 Database table „JudesioData“ structure

## Table “ledBrightness”

The purpose of the table is to store information for the settings of the controlled lighting device that the user will be able to change when logged into the system. Fields that make up the table (Figure 9):

1. “BrightID” – primary key
2. “Brange” – light intensity parameter of the controlled lighting unit
3. “Drange” – parameter of the controlled lighting unit light up time delay.

#	Pavadinimas	Tipas	Palyginimas	Atributai	Null	Nutylint	Komentari	Papildomai	Veiksmas
<input type="checkbox"/>	1	BrightID	int(11)		Ne	Jokio		AUTO_INCREMENT	Redaguoti Šalinti Daugiau
<input type="checkbox"/>	2	Brange	varchar(111) utf8mb4_general_ci		Ne	Jokio			Redaguoti Šalinti Daugiau
<input type="checkbox"/>	3	Drange	varchar(111) utf8mb4_general_ci		Ne	Jokio			Redaguoti Šalinti Daugiau

Figure 9 Database Table „ledBrightness“ structure

All database tables are created using the graphical user interface software package *phpMyAdmin*.

#### 4.4 Lighting unit and Arduino microcontroller design

For the implementation of this work, it was chosen to design and create a 12 step stair lighting device, which, after integrating PIR motion sensors, would sequentially light up the 12-volt LED strips in the stair steps, at the exact moment when the movement directed to the stairs is detected. Two LED parameters could be changed by the user, the LED brightness and sequential light up delay time. The total count of stair steps is used to generate a rough estimate of calories burned while using the stairs.

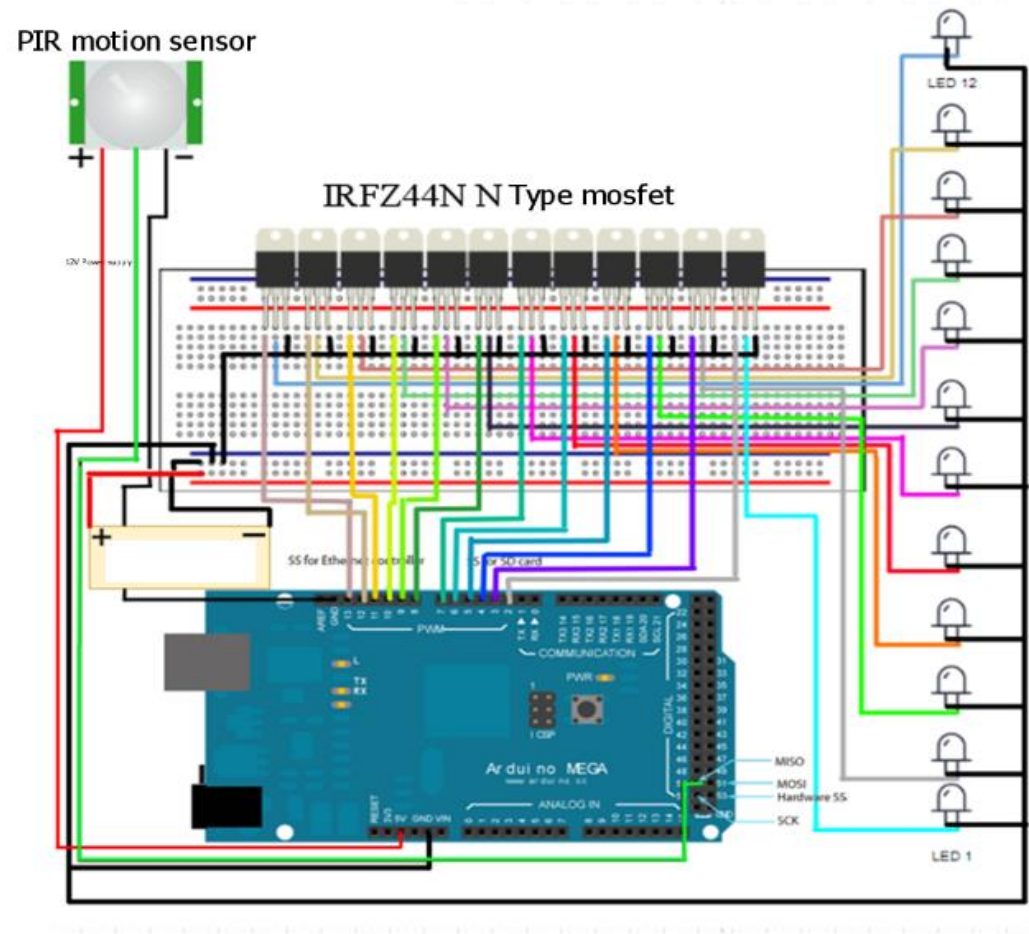


Figure 10 Lighting unit and Arduino microcontroller connection scheme

The diagram of the design stair lighting device shows the connection of all components used (Figure 10). The components included:

1. Twelve IRFZ44N N type MOSFETS
2. Twelve 12 Volt LED Strips
3. Arduino Mega 2560 R3 microcontroller
4. 12 Volt power supply
5. PIR motion sensor

A 12V power supply must be used to feed the 12 volt appliances. However, the working voltage of the Arduino mega 2560 R3 microcontroller is only 5 volts, therefore, in order to control 12 volt devices, semiconductor devices – transistors or relays – need to be inserted into the electrical circuit (Seedstudio, 2020). For this work, the use of type N conductive channel metal, oxide and semiconductor field transistors (MOSFET) IRFZ44N has been selected, which has three control electrodes – G “Gate”, D “Drain” and S “Source” (Figure 11).

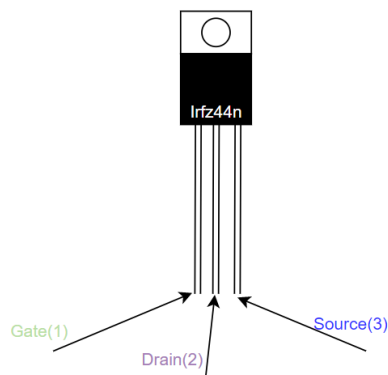


Figure 11 IRFZ44N MOSFET electrodes

### LED strips and IRFZ44N MOSFETS

The poles of the negative electric current source of all LED strips are separately connected to the MOSFET IRFZ44N Drain electrodes. The positive electric current poles of the LED strips are commonly connected to the 12-volt power supply positive pole. The MOSFET source electrodes are commonly connected to the negative pole of the power supply, and the remaining gate electrodes of the MOSFETS are connected in series with the Arduino mega 2560 R3 microcontroller PWM (Pulse-width modulation) ports labeled 2–13, which, depending on the transmitted signal complete the circuit between LED strips and the 12V power supply. IRFZ44N MOSFET will act as a switch, controlled by Arduino microcontroller, whose GND (ground) port is also connected to the 12V power supply negative pole in order to complete a common electric circuit.

### HC-SR501 PIR motion sensor and Arduino Mega 2560 R3

The purpose of the motion sensor is to ensure that led strips light up automatically, when user comes close to the stairs. After detecting movement,

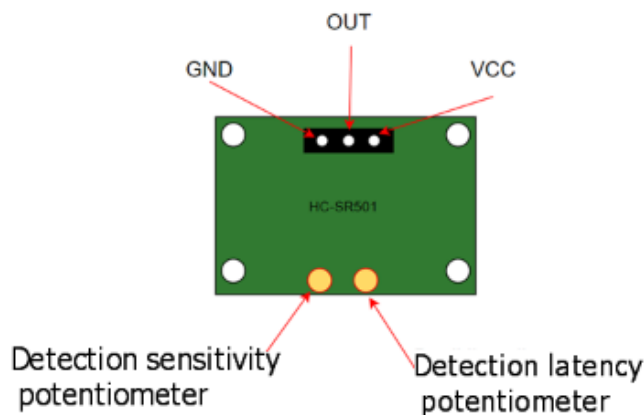


the sensor will transmit a HIGH signal to the Arduino microcontroller, which will initiate a sequential LED light-up of the stairs. The HC-SR501 motion sensor has a working electric voltage of 4.5V to 12V, so it can be fed using the 5V output port of the Arduino. The PIR motion sensor used has three pins GND - Ground, HIGH/LOW Dout - digital signal transmission pin and Vcc - positive electrical current connector (.). When using the OUT pin, depending on whether the sensor has detected movement, a HIGH or LOW signal will be transmitted to the controller.

*Table 4 PIR motion sensor and Arduino connection table*

HC-SR501 PIR pins	Arduino mega 2560 R3 ports
Out(Output)	Digital 50 (Input)
GND	GND (Output)
Vcc	5V (Output)

Table 4 represents the motion sensor and Arduino microcontroller pin connections correspondingly.



*Figure 12 HC-SR501 PIR motion sensor*

The sensor detection sensitivity and latency are adjusted physically using the potentiometers.

### **Arduino microcontroller code**

To program the logical work and parameter management scenario of the stair lighting device, Arduino IDE software is used in the Raspberry Pi SBC.

Arduino board is connected to the Raspberry Pi 3 B+ using a USB serial cable, which provides the needed 5 V power as well as ability to send and receive data from the computer.

### Variable and pin declaration

Based on the connection diagram, the program initially defines the connections of all IRFZ44N MOSFET gate electrodes used, PIR motion sensor's Dout to microcontroller ports, and other variables that are assigned as output in the void setup() function. Serial line bandwidth equal to 9600 bits per second (Figure 13).

```
int ledPin1 = 13;
int ledPin2 = 12;
int ledPin3 = 11;
int ledPin4 = 10;
int ledPin5 = 9;
int ledPin3 = 8;
int ledPin1 = 7;
int ledPin2 = 6;
int ledPin3 = 5;
int ledPin1 = 4;
int ledPin2 = 3;
int ledPin3 = 2;
int PIRpin = 50;
int pirState = LOW;
int val = 0;
int X;
int brightness;

void setup() {
  pinMode (ledPin1, OUTPUT);
  pinMode (ledPin2, OUTPUT);
  pinMode (ledPin3, OUTPUT);
  pinMode (ledpin...);
  pinMode (PIRPin, INPUT);

  Serial.begin(9600);
}
```

Figure 13 Pin declaration

### Serial line read and variable assignment using ASCII encoding and decoding

Arduino uses ASCII character encoding for electronic communication over the serial line, meaning that various encoded data could be either sent or received in order to control various signal outputs for the connected electronic appliances, in this case - 12 V LED strips and PIR motion sensor. Using function "void loop()" ensures that the information is transmitted or read in a real-time cycle. Depending on the information received, ASCII values are assigned to corresponding variables inside the code (Table 5).

Table 5 ASCII value to variable assignment table

ASCII values	Equivalent variable
118	brightness = 20;
108	brightness = 80;
109	brightness = 120;
107	brightness = 255;
104	X = 500;
111	X = 1000;
112	X = 1500;
116	X = 2000;

```
void loop(){
  if (Serial.available()){
    switch(Serial.read()){
      case 118:
        brightness = 20;

        break;
      case 108:
        brightness = 80;

        break;|
      case 109:
        brightness = 120;

        break;
      case 107:
        brightness = 255;
```

Figure 14 ASCII value assignment to variables

Switch statement is used to help assign each ASCII value read from the serial line to the corresponding variables.

### Lighting unit logical operation script

After variable assignment, using the same loop cycle, the logical operations are scripted. The variable “*brightness*” is used with “*analogWrite*” function, which sends the signal to the designated PWM pins, lighting up the corresponding LED strips. The variable “**X**” determines the latency it takes for each LED strip to light up.

Variable “**val**” using function “*digitalRead()*” receives information from PIR sensor, where “*HIGH*” is equal to motion detected and “*LOW*” – movement is absent. Using the “*If*” statement, depending whether the motion has been detected, a sequential LED light up is initiated.

```

val = digitalRead(PIRpin); // read input value
if (val == HIGH) { // check if the input is HIGH
    analogWrite(ledPin1, brightness);
    delay(X);
    digitalWrite(ledPin1, LOW);
    analogWrite(ledPin2, brightness);
    delay(X);
    digitalWrite(ledPin2, LOW);
    analogWrite(ledPin3, brightness);
    delay(X);
    digitalWrite(ledPin3, LOW); // turn LED ON
    if (pirState == LOW) {

        Serial.println("Motion detected!"); |

        pirState = HIGH;
    }
} else {
    digitalWrite(ledPin1, LOW);
    digitalWrite(ledPin2, LOW);
    digitalWrite(ledPin3, LOW); // turn LED OFF
    if (pirState == HIGH){
        // we have just turned of
        Serial.println("Motion ended!");
        // We only want to print on the output change, not state
        pirState = LOW;
    }
}
}

```

---

Figure 15 microcontroller's logical operation code

When the motion sensor senses the movement, a string “*motion detected*” is printed out into the serial line, where later it could be read by the Raspberry Pi and saved as an entry in the database server, from which it could be rendered for the user of the system (Figure 15).

#### 4.5 Data communication between mariaDB database server and Arduino mega 2560 R3

For data transfer and storage between the database and the Arduino, it will be necessary to write a Python script which would operate this task. The Python script is hosted and executed on the Raspberry Pi. The data will be transferred over a USB serial cable connected to the Arduino. Data flow scheme is provided below (Figure 16).

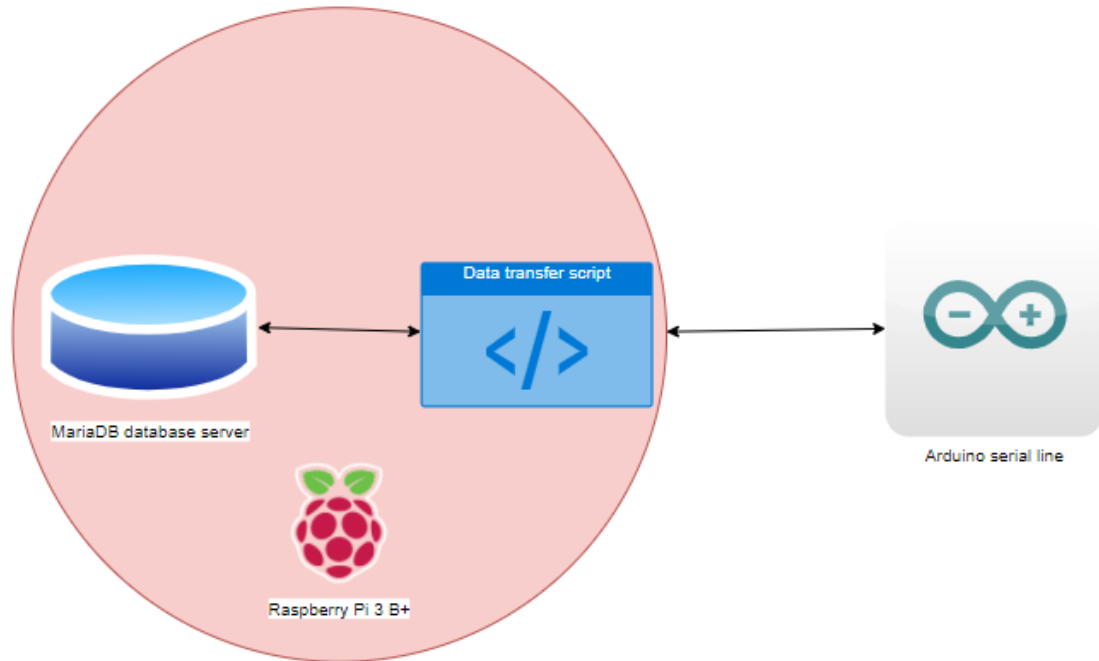


Figure 16 Software subsystem dataflow scheme

## Python function to send data to the microcontroller

```
def brightness():
    try:
        ser = serial.Serial('/dev/ttyACM0',9600)
        while 1:
            connection = mariadb.connect(user='VartotojasDB', password='w3b.s3rv', host='localhost', database='Duomb')
            cursor = connection.cursor(buffered=True)
            sql_select_brightness = ("SELECT Brange, Drange FROM `ledBrightness` WHERE `BrightID` = 3")
            cursor.execute(sql_select_brightness)
            brecords = cursor.fetchone()
            brightnessVar = brecords[0]
            delayVar = brecords[1]
            if delayVar == "500":
                delayVar = "h"
            if delayVar == "1000":
                delayVar = "o"
            if delayVar == "1500":
                delayVar = "p"
            if delayVar == "2000":
                delayVar = "t"
            #brightness
            if brightnessVar == "0":
                brightnessVar = "v"
            if brightnessVar == "1":
                brightnessVar = "l"
            if brightnessVar == "2":
                brightnessVar = "m"
            if brightnessVar == "3":
                brightnessVar = "k"
            time.sleep(1)
            print brightnessVar
            print delayVar
            ser.write(str(brightnessVar))
            ser.write(str(delayVar))
    except Error as e:
        print("Nepavyko nuskaityti brightness duomeni")
brightness()
```

Figure 17 Python code „connect.py“

## Function to send data from the database to the Arduino microcontroller

The script is started by declaring a new function “*brightness()*” (Figure 17). Inside the function the Serial line variable is declared, in this case “*ttyACM0*” port is used, where the Arduino is physically connected using a USB cable.

A “*while True*” loop is initiated which ensures that the code runs continuously and a connection to the mariaDB server is set, where SQL syntax queries will be sent to the existing table “*ledBrightness*”.

Based on MySQL reference manual Dev.mysql (2021), SQL query is generated: “*SELECT Brange, Drange FROM `ledBrightness` WHERE `BrighID` = 3*”. Query results are assigned to the “*brecords*” array. The first value being Brange is assigned to the “*brightnessVar*” variable and the second value of the array “*Drange*” is assigned to a new variable – “*delayvVar*”. The two new variables are then checked using “*if*” statements and their matches are assigned to characters whose ASCII encoding is in decimal form (Table 6).

Table 6 Database data to character variable assignment

Variable check statement	Variable assigned character	Assigned character ASCII encoding
if delayVar == "500":	h	104
if delayVar == "1000":	o	111
if delayVar == "1500":	p	112
if delayVar == "2000":	t	116
if brightnessVar == "0":	v	118
if brightnessVar == "1":	l	108
if brightnessVar == "2":	m	109
if brightnessVar == "3":	k	107

When the corresponding letters are assigned, they are sent using the function “*ser.write(str())*”.

## Python code to read data from the microcontroller

```
def motionDetection():  
while 1:  
    if (ser.in_waiting > 0):  
        lines = ser.readline()  
        if ("detected" in lines):  
            print lines  
            time.sleep(2)  
            motion = "was indeed detected"  
            print motion  
            SQL_query = ("INSERT INTO JudesioData (Aptiktas) VALUES ('Judesys')")  
            cursor.execute(SQL_query)  
            connection.commit()  
            lines = []  
            print(cursor.rowcount, "Atnaujinta.")  
            #connection.close()  
            lines = "beleka"  
except mysql.connection.Error as error
```

Figure 18 Python code „ardtorp.py“

## Function to send data from the Arduino microcontroller to the database

To read data from the microcontroller, function “*motionDetection()*” is declared (Figure 18). In the “*While True:*” loop a variable lines is created, which with the help of function “*ser.readline()*” reads the previously declared serial line. Using a simple “if” statement to check for a string, an operation can be initiated to send an SQL query to the database. If the string is correct to the condition, an SQL query “*INSERT INTO JudesioData (Aptiktas) VALUES ('Judesys')*” is performed. The result of the query is a new entry in the database table “*JudesioData*”.

## 4.6 Graphic user interface (GUI) model

### 4.6.1 System’s logical model

Main user of the system – authorized person. The user who successfully connects to the system using login details can change parameters of the lighting unit connected to the Arduino microcontroller:

1. LED strip brightness
2. LED strip light up latency (time delay).

Logged in user can also review information about the operation of lighting unit and finish his session by logging out of the system. System’s use case diagram provided below (Figure 19).

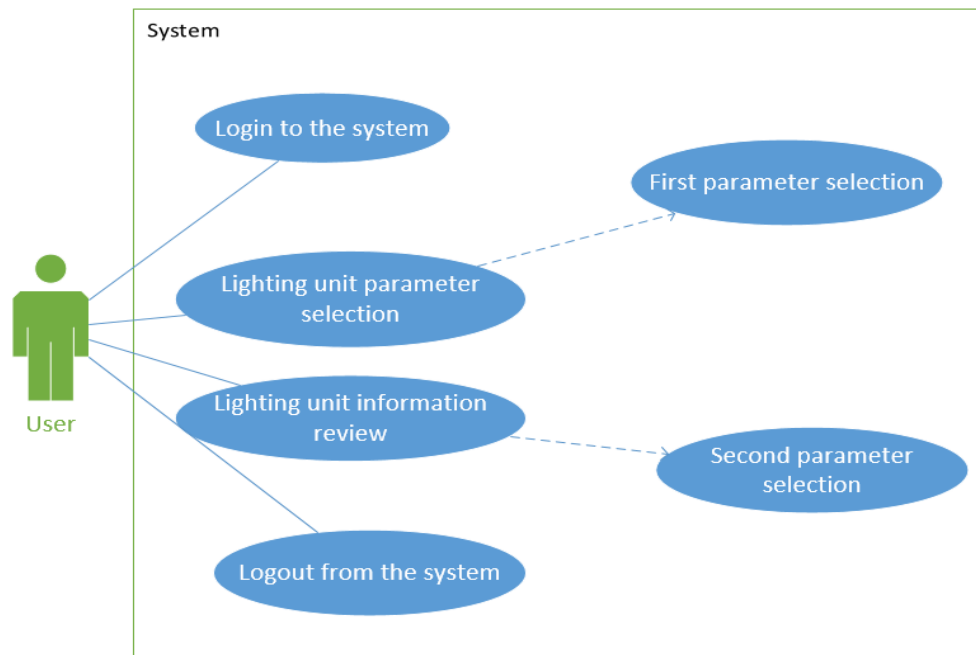


Figure 19 System’s Use Case Diagram

### **Login to the system:**

The process is designed to authenticate the user. It contains your login and password. After successful login, the user can continue to work in the system.

### **Lighting unit parameter selection:**

After successfully connecting, the user can change the parameters for the controlled lighting unit. There are two of these parameters.

- **First parameter selection:**  
The first parameter is to change the brightness of the LED stripes of the designed lighting unit.
- **Second parameter selection:**  
The second parameter is intended to change the time of illumination of the LED strips of the designed lighting unit.

### **Lighting unit information review:**

After successfully connecting, the user can view the information of the designed lighting device. Provided in textual form, the information includes:

- Number of how often the stairs were used in the past week.
- Number of calories roughly burned during the last week when using stairs.
- Number of times the lighting unit itself has activated during the past week.

### **Logout from the system:**

Disconnect an authenticated user from the system to complete the session.

### **Activity - Login to the system:**

In the provided Figure 20, it can be seen that the user after typing in the system's IP address into the web browser URL search bar is redirected to the login form page. In this form, the user must provide accurate login details: login and password. When the form is submitted, the system checks the data provided by the user with the data in the system database. If the data provided by the user does not match, the user receives a message that the submitted data was not correct and the user is redirected to re-fill in the login form. If the data retrieved in the form is correct, a login session is initiated that ensures that the system is used by an authorized person and that the user is redirected to the system home page.



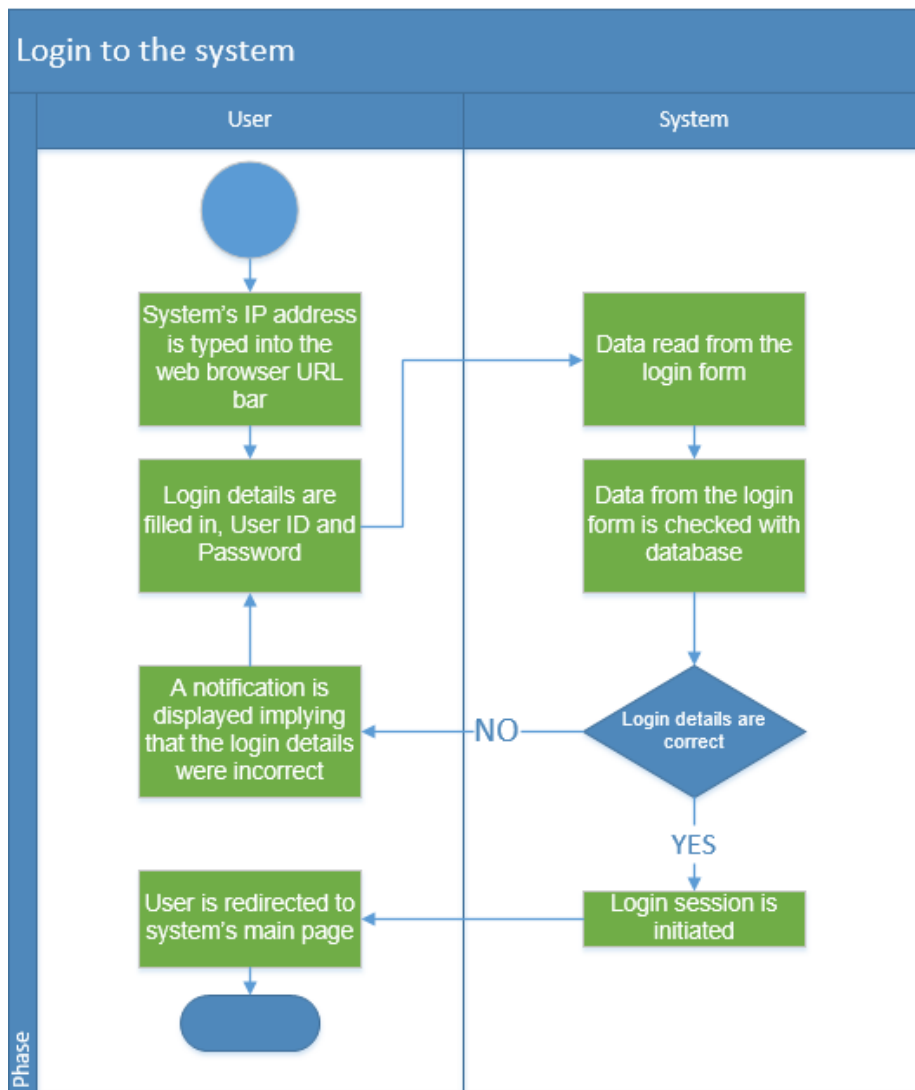


Figure 20 Login to the system activity diagram

**Activity - Lighting unit parameter selection:**

The following Figure 21 for changing settings for the controlled lighting unit shows that when a user logs on to the system home page, the system checks whether a sign-in session has been started. If the answer is no, the user is redirected to the filling in the system login form page, if the answer is yes, the user can continue working in the system and change the settings of the managed device. Session verification is designed to ensure that the system is used by an authorized user. The authorized user continues to change the first or second setting of the lighting unit, after providing the selected instructions, the system reads them and updates them to the corresponding device parameter data in the database. The user receives a message about the successfully changed settings.

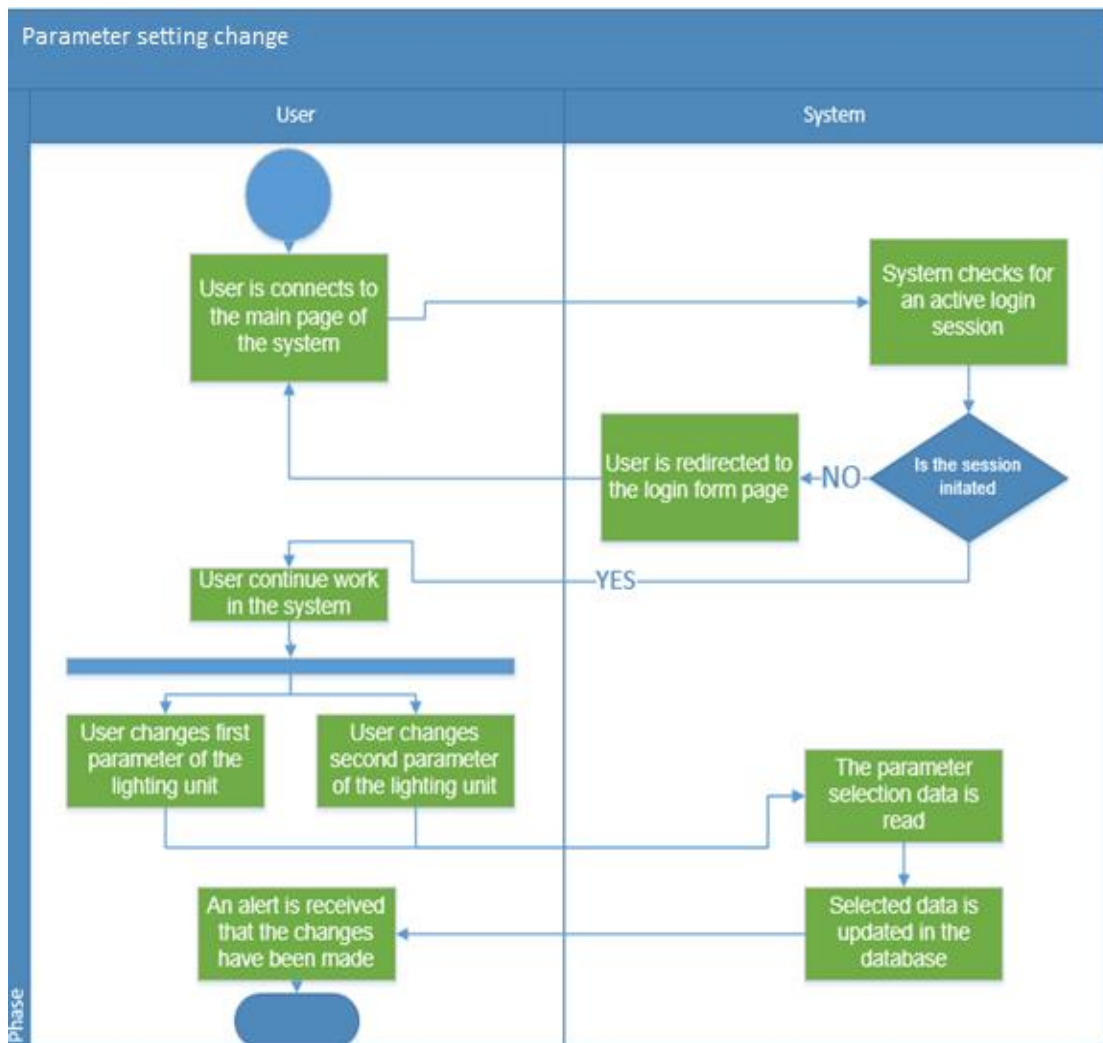


Figure 21 Lighting unit parameter selection use case diagram

### Activity - Lighting unit information review:

Figure 22 shows that just as in the previous activity diagram, a sign-in session check is initiated. If the answer is yes, the system sends a query to the database to retrieve last week's records of the lighting unit operation. As of the previous week, the following information is generated, which is provided to the user on the website in text form:

1. Number of times stairs were used in the last week – the system displays the message using the number of entries for the operation of the lighting unit stored in the database.
2. Number of calories roughly burned while using the stairs during the last week – the system displays the message using the formula:

$$\text{Burned calories} = (L \times K) \times S$$

Where, L – Number of stair steps;

K – Approximate number of calories burned when climbing one step of the stairs (0.17 cal);

S – Number of activation records of the lighting unit in the last week.

3. Number of lighting unit activation in the last week - the system displays the message using the number of trigger records of the lighting device in the database.

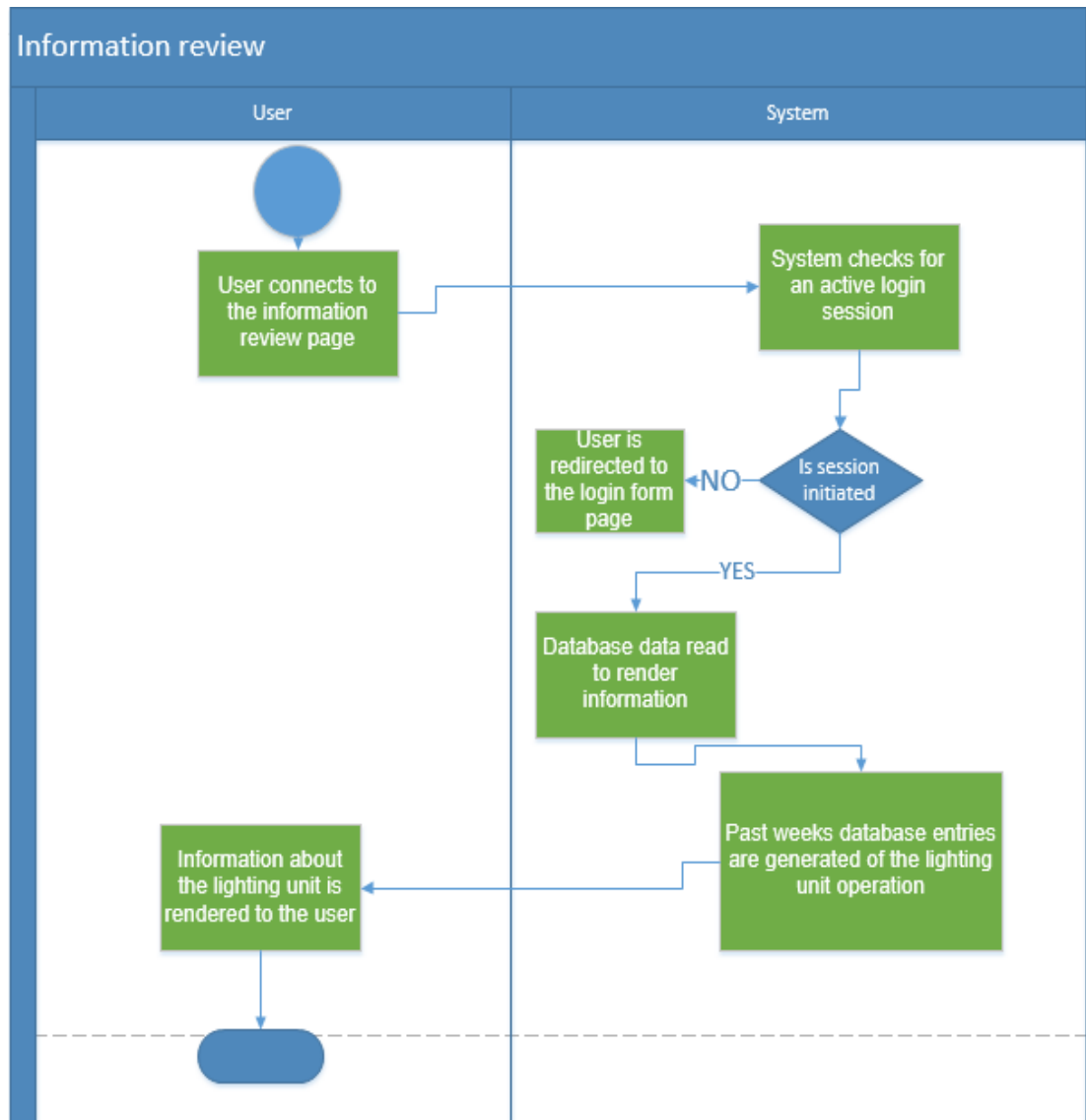


Figure 22 Lighting unit information review activity diagram

### Activity - Logout from the system:

Figure 23 shows that when a user clicks "Logout", the system stops the login session and directs the user to fill in the login form again. Stopping the session strengthens the security of the system, against potential attackers.

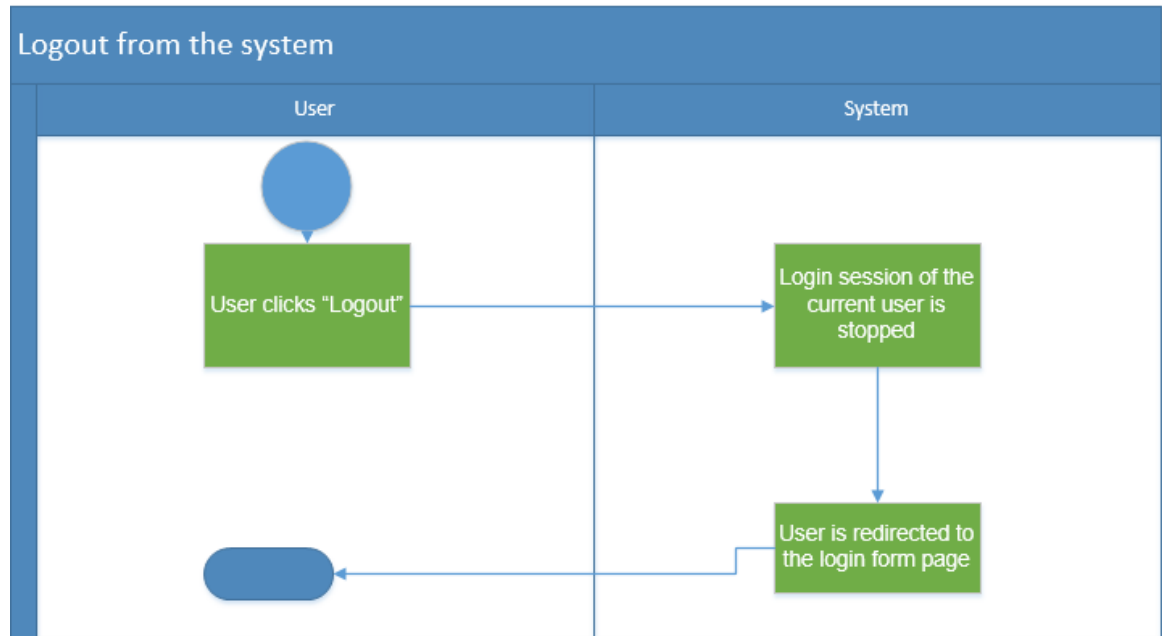


Figure 23 Logout from the system activity diagram

All activity diagrams describe system's GUI model behavioral process.

#### 4.6.2 User interface structure

A web browser is required to log in to the system. The user who entered the system address in the search box will be taken to the login form page of the system graphical interface.

##### Main graphical interface web page

When a user enters the system IP address in the browser, the following will be displayed until a successful login to the system is completed (Figure 24).

Username
Password
Button_1

Figure 24 Login web page structure

Graphical interface's login page form field specifications are mentioned below (Table 7).

Table 7 Login web page field specification

Field	Field type	Field characteristics	Field description
Username	Text	Input characters aren't hidden	Text input field where username data is typed in
Password	Password	Input characters are hidden	Password type field where user password is typed in
Button_1	Button	Type="submit"	Form send button

After a successful login to the system, the main page of the graphical user interface is opened (Figure 25).

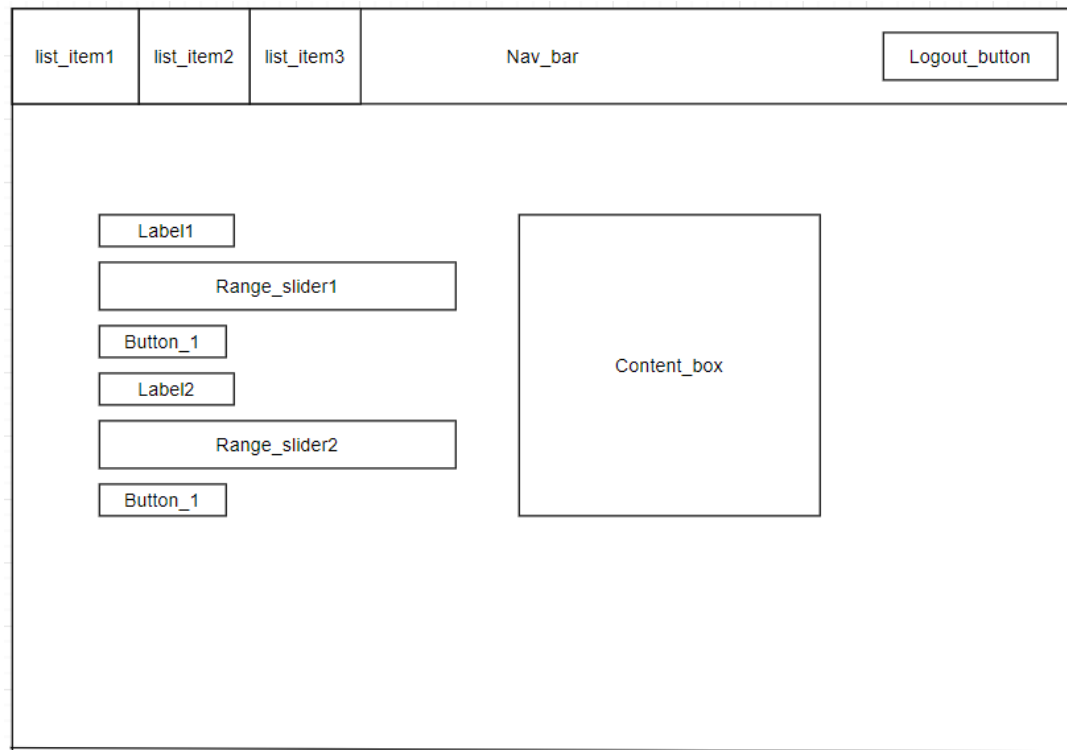


Figure 25 Main page structure

The main page uses a style template written in the CSS programming language as all other pages of the system user's graphical interface site, so the graphical styles of the pages on the interface site remain the same. The field specification table on the main graphical interface page explains the purpose of the site elements (Table 8).

Table 8 Main web page field specification

Field	Field type	Field characteristics	Field description
Nav_bar	Header	Three list items and one button	Systems page navigation bar
List_item1	Php script		Program code for real time printing of date and time
List_item2	href	Clickable	Main graphic user interface page link
List_item3	href	Clickable	Information review page link
Logout_button	button	Clickable	Session suspension button
Label1	text	Visible all the time	Text to describe lighting unit parameter range slider
Range_slider1	Range slider	Four space values	LED brightness parameter range slider
Button_1	button		Button to change settings into the database
Label2	text		Text to describe lighting unit parameter range slider
Range_slider2	Range slider	Four space values	LED light up latency parameter range slider
Button_2	button		Button to change settings into the database
Content_box	Php script		

On the main page of the system graphical interface, the user controls the lighting unit parameters by selecting the settings of the range sliders and pressing the confirmation button. The confirmation buttons initiate requests on the Web server to pass the selected information to the database. Pressing the "Information" item on the navigation bar redirects user to the informational page of the graphical interface, which is intended to provide the user with information about the statistics of the controlled lighting unit in the last week (Figure 26).

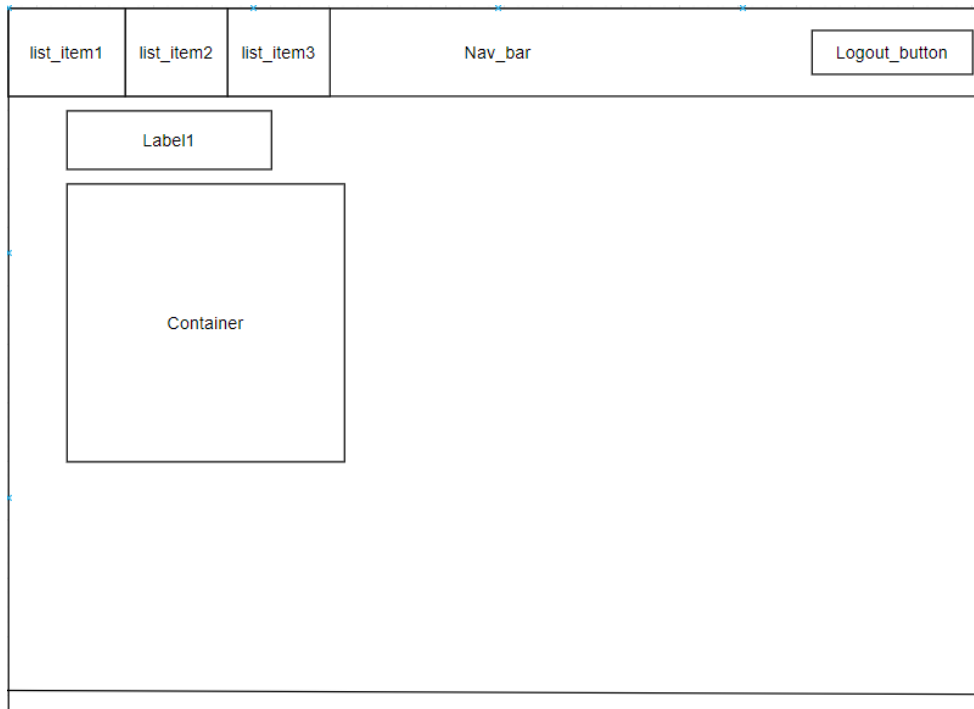


Figure 26 Information review web page structure

Graphical interface's information review page field specifications are mentioned below (Table 9Table 7).

Table 9 Information review web page field specification

Field	Field type	Field description
Label1	Text	Text field to describe information
Container	Php script	Field to generate SQL query information from the database

The navigation bar used on the main graphical page remains in the information review page.

#### 4.7 Software architecture design

All the software in the project is hosted on the Raspberry Pi single-board computer, where the Web server supports the lighting system's web pages that the user can use to connect to the system itself, change the controlled lighting unit parameters and view their operation information. All data is stored in the designated MariaDB database. For system implementation back-end Web Server programming language „PHP“ and interactive programming language „Python“ are used to create common communication between the database and the Arduino microcontroller. Software module diagram that makes up the system is provided below (Figure 27).

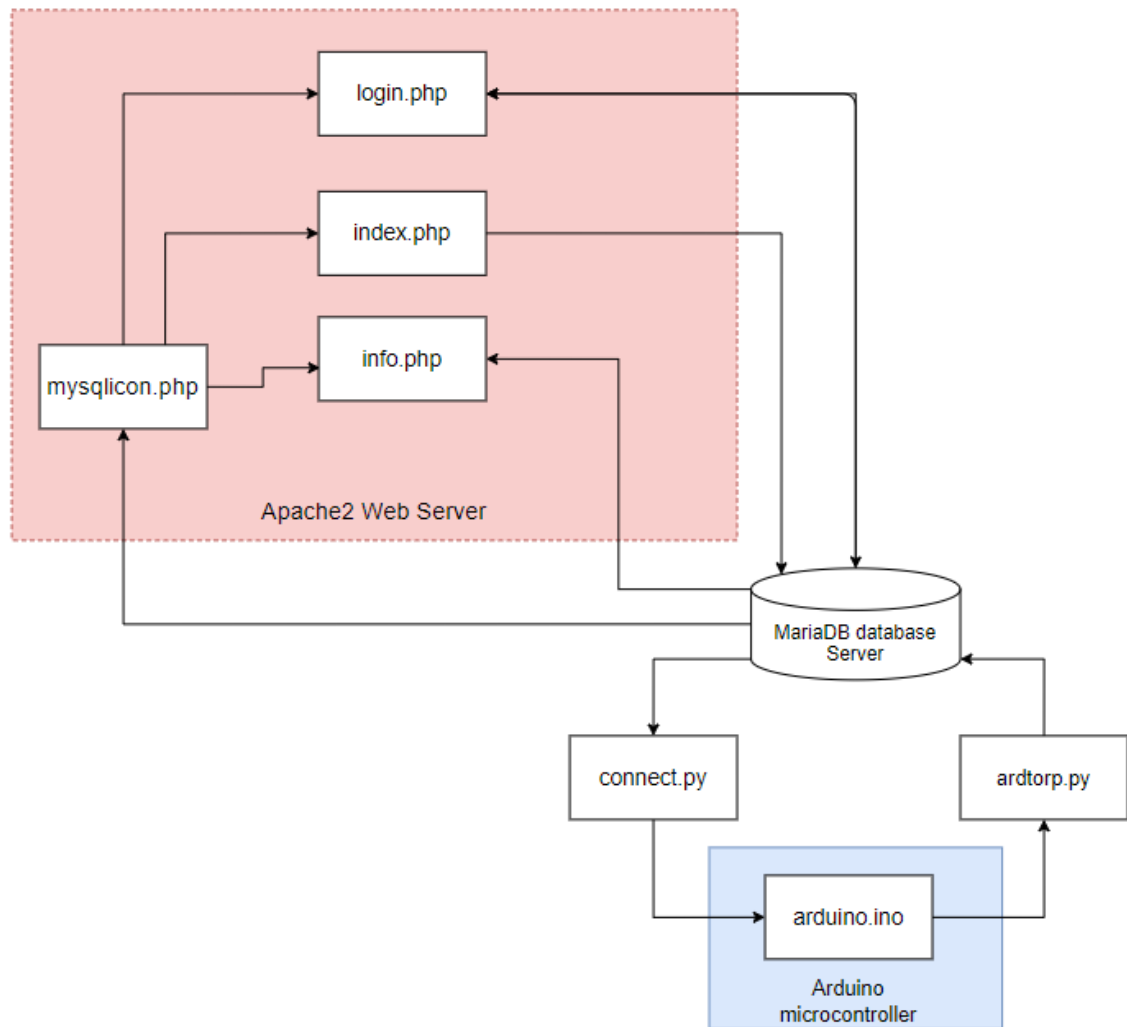


Figure 27 System's software module diagram

### System software module functions:

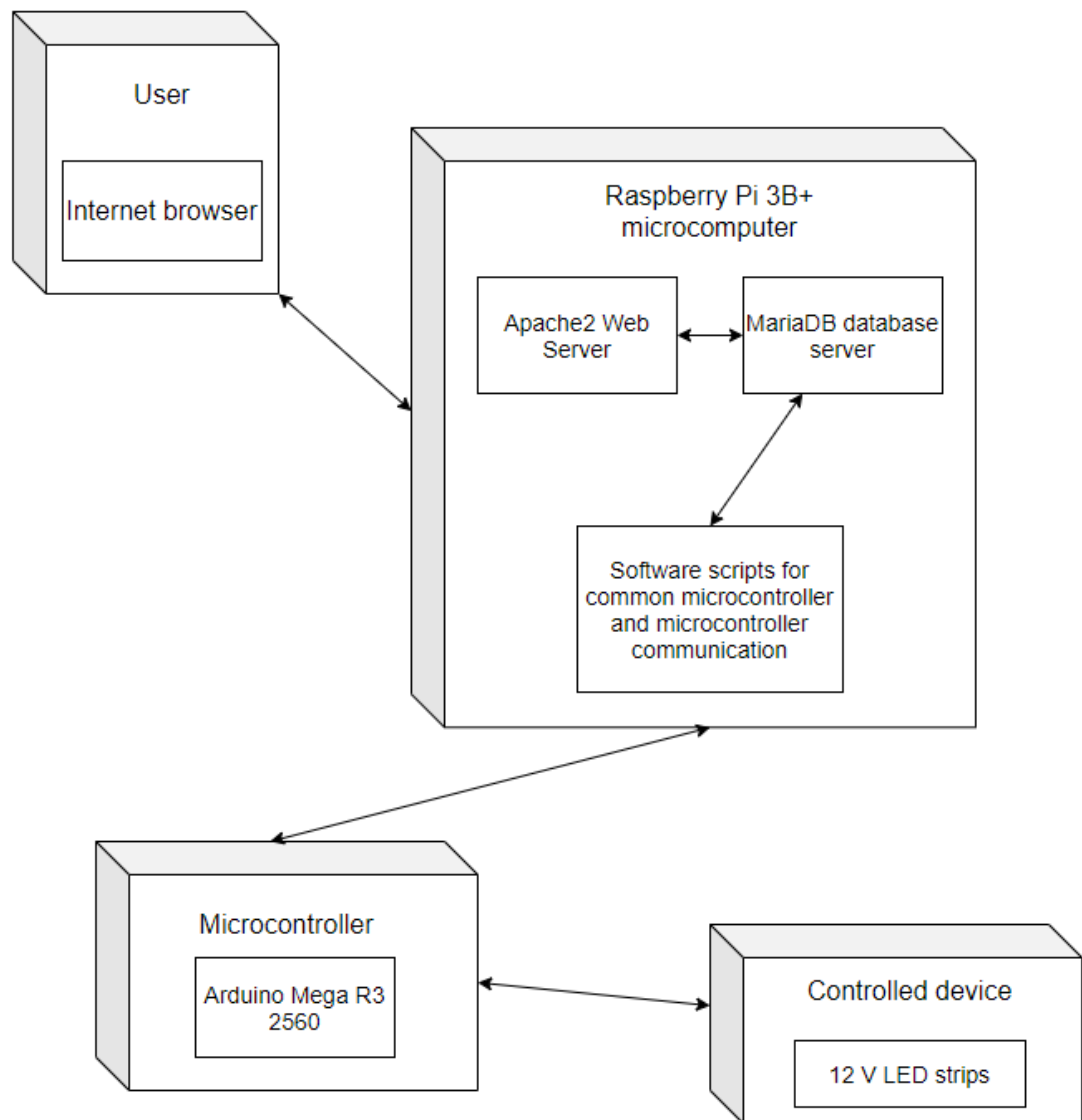
1. **Mysqlcon.php** - This module is a code written in PHP programming language, the purpose of which is to store information for the connection of a web server to the MariaDB database variables. The module stores the database address, database name, database user name, and password. By storing database login data in a separate, protected document that is not accessible to potential attackers or users through a web browser, protects the database and the data contained therein. The module is mapped to all other modules that initiate database connections.
2. **Login.php** – The purpose of the module is to exclude any unauthorised user from the control system. The login form in the module allows the user to log into the system using the login and password which is encrypted in the database using the PHP programming language bcrypt algorithm. After the user enters the login details and presses the login button, the login to the database begins, which checks whether the data filled in the user form



corresponds to those in the database and if the answer is yes, the user is redirected to the main module of the system – index.php.

3. **Index.php** - The main module of the smart home lighting system, in which the settings or parameters of the controlled lighting unit are changed. In the case of this project, the parameters of the system-designed stair lighting unit. After the user changes the selected parameters and presses the "change settings" button, the change data is updated in the database, where later this parameter data is transmitted to the Arduino microcontroller using a physical USB serial connection to the Raspberry Pi and a Python programming language written script – connect.py
4. **Connect.py** – Python programming language written module designed to transmit user-changed parameter instructions to the microcontroller using a serial USB connection. The script checks every 5 seconds that the database has not changed the corresponding parameter settings, converts the data from the database into simple characters that would be understandable to the Arduino in ASCII encoding form and sends it via Serial line. Module is hosted on the Raspberry Pi SBC.
5. **Ardtorp.py** – A Python script, hosted on the Raspberry Pi SBC used to read data from the Serial line that is physically connected to the Arduino microcontroller. The read data is correspondingly sent to MariaDB database table. In this case, information about lighting unit operation is communicated. Later this information is rendered graphically to the user using module – info.php
6. **Info.php** – Module used to render information from the database server about the controlled lighting unit. A running script connects to the database server, using an SQL query collects the corresponding data from the tables and renders it in a text form for user to review.
7. **Arduino.ino** - This module is the Arduino microcontroller's task script, controlling the lighting unit activity and parameters. After receiving information from the connect.py module via the serial line, decodes it and assigns the given values to the corresponding variables that change the parameters of the lighting unit. Reads digital data from the connected PIR motion sensor and accordingly lights up the LED strips by sending signal to the MOSFETS of the circuit, as well as send information to the Serial line that the sensor has activated. This information is then uploaded to the database using the ardtorp.py module.

All “.php” module codes are written using the php programming language syntax and technology documentation based on W3schools (2021). The “.py” module codes are written using Python technology methodical documentation base on Devguide.python (2021).



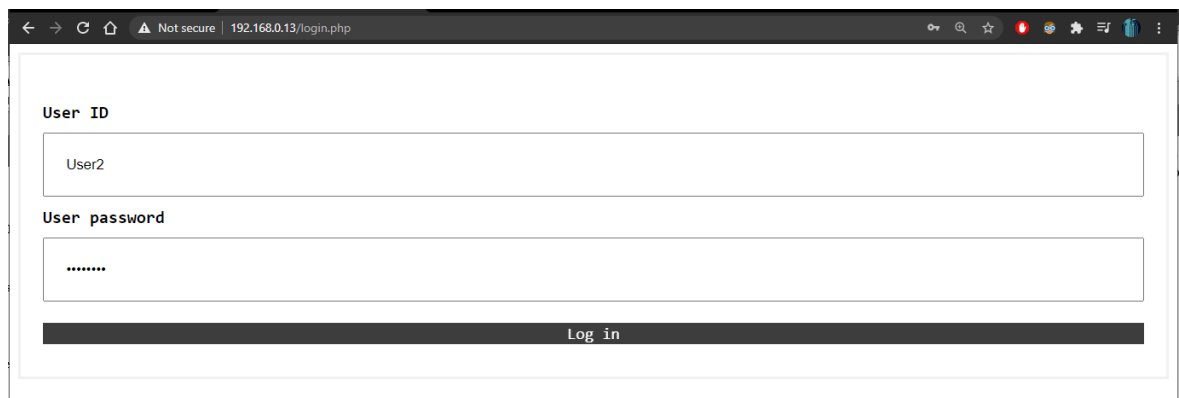
*Figure 28 Deployment diagram*

The deployment diagram shows the physical compatibility of the configuration between the software and the hardware. All components interact with each other to form a common communication of data flow (Figure 28).

## 4.8 PROJECT RESULTS

System's prototype example of the project has been developed using IoT technologies, single-board computers and microcontrollers, hardware and software components and other tools that make up the system.

Based on the project design part, user interface model and software architecture design a prototype of user's graphical interface and an information subsystem for common communication between system's components were developed on the Web Server. A prototype of the web management system is provided below.



The image shows a web browser window with the address bar displaying 'Not secure | 192.168.0.13/login.php'. The main content area of the browser contains a login form. The form has two input fields: the first is labeled 'User ID' and contains the text 'User2'; the second is labeled 'User password' and contains masked characters '.....'. Below these fields is a dark button labeled 'Log in'.

*Figure 29 Implemented Login page*

Figure 29 shows that when a user enters a system private IP address in a web browser on a local area network, the user is redirected to the initial page of the home lighting system, where the login data form is filled in for authorization.

After successfully filling in the login form, the user is redirected to the main page of the system (Figure 30). The parameters of the designed stair lighting unit are changed using the provided graphic interface range sliders. The header section of the page contains a navigation box that directs the user to the device's information review page. In this navigation box, the user can also finish working in the system by pressing the "Logout" button. After selecting the wanted parameters, the button "Change setting", the selected value are then updated in the database table correspondingly and sent to the Arduino microcontroller.



Figure 30 Implemented Main page

The provided illustration shows that when the user presses the "Information" button in the navigation box of the home page, the user is redirected to the information review page, information about the operation of the lighting unit operating in the system is provided (Figure 31). The information provided reflects the data for the last week. Therefore, if the device has not been used in the last week, the rendered database information does not contain records and displays the number "0" in their place.

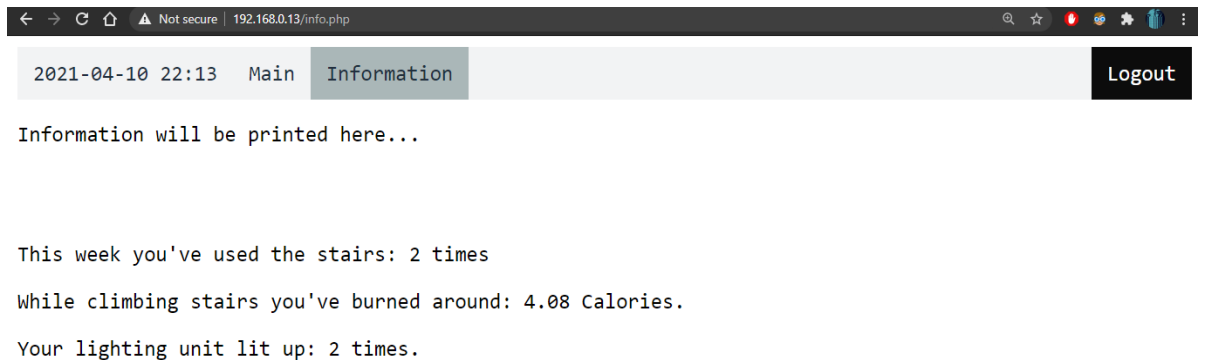


Figure 31 Implemented Information review page

On the Raspberry Pi we can examine the data communication python script (Figure 32). Using the terminal and typing in command: **python script\_filename.py**. The script is initiated, and the character prints show us that the database data read is working and data is being transmitted over the serial line to the Arduino microcontroller.



## **Design suggestions for future development**

The system is designed to be accessible only from private network, however it is possible to allow users be able to access it from public networks by implementing network address translation (NAT) rules to the router of which the system would be connected to. By allowing system to be accessed from public networks their vulnerabilities are easier to exploit, and without a doubt increase the risk of unwanted connections. Suggestions to increase security:

- Implement a secure and encrypted communication channel between client's browser and the server by using SSL/TLS certificate.
- Implement firewall rules of the router to block unwanted IP address connections.
- Implement firewall rules of the router to block connections with unwanted transport protocol (TCP) and User datagram protocol (UDP) port numbers.

Current smart home lighting system project's design uses a USB serial cable to both power the Arduino and transfer data to the Raspberry Pi, however an Arduino board with a Wi-Fi module could be used for data transfer, this would allow the Raspberry Pi to be placed anywhere in the house while still maintaining common communication with the Arduino board or other system's potential components.

## 5 CONCLUSIONS

1. Inspected the current market for smart home control systems Fibaro HomeCenter and Apple HomeKit, their identified advantages: systems are complete, have many functions; and disadvantages: extremely high prices, compatible only with specific devices.
2. Web development and IoT project programming languages and common security breaches of web systems and applications have been researched and inspected. Equipment chosen to achieve the work goal: Raspberry Pi 3b, Arduino Mega 2560 R3, Apache2 web server, mariaDB database, Php internal programming language, HTML programming language, Python programming language.
3. The selected Web server and database are setup and configured for storing and sharing information between devices and home lighting unit system components on the local network.
4. A 12 Volt LED strip stair lighting unit has been designed and developed to be controlled by the arduino microcontroller with the help of a PIR motion sensor and IRFZ44N N type MOSFETS. Written Python software scripts for single-board computer and controller.
5. A graphical user interface was designed and implemented onto the configured apache2 Web server, user is able to control two parameters of the controlled lighting unit by connecting to the system via a web browser, as well as review it's operational information.
6. System`s prototype task operation was inspected.

The home lighting system has been researched and designed, which automates lighting devices in user's home, thus providing additional comforts and the ability to monitor the operation of the lighting units in a web browser. An experimental prototype using IoT technologies, single-board computers and microcontrollers, computer network technologies, programming technologies has been developed. A prototype system has been designed to allow the installation of new controlled appliances if necessary.

## REFERENCES

Mohan Krishnamurthy, Eric S. Seagren, Raven Alder, Aaron W. Bayles, Josh Burke, Skip Carter, & Eli Faskha 2008. Chapter 11 - Apache Web Server Hardening. In How to Cheat at Securing Linux (pp. 383-401). Elsevier.

Ramoška, A. 2012. Apsaugos nuo SQL injekcijų el.verslo svetainėse metodikos sudarymas ir tyrimas: Magistro darbas. Kaunas: Kauno technologijos universitetas. Prieiga per eLABa – nacionalinė Lietuvos akademinė elektroninė biblioteka.

Dow, Colin 2018. Internet of things programming projects: Build modern IoT solutions with the Raspberry Pi 3 and Python. Birmingham: Packt Publishing.

Monk, Simon 2016. Programming Arduino: Getting started with Sketches (2nd ed.). New York, N.Y.: McGraw-Hill Education.

Devguide.python.org 2021. Python developer's guide. Available at: <https://devguide.python.org/> [Accessed 2021-02-05].

Httpd.apache.org 2021. Apache2 official Web Server documentation. Available at: <https://httpd.apache.org/docs/2.4/> [Accessed 2021-02-06].

Dev.mysql.com 2021. Official MySQL documentation. Available at: <https://dev.mysql.com/doc/> [Accessed 2021-02-07].

Dev.mysql.com 2021. Python and MySQL connection. Connecting to MySQL Using Connector/Python. Available at: <https://dev.mysql.com/doc/connector-python/en/connector-python-example-connecting.html> [Accessed 2021-02-15].

W3schools.com 2021. Php and other programming language online learning environment. PHP Tutorial w3schools.com. Available at: <https://www.w3schools.com/php/> [Accessed 2021-02-20].

Seedstudio.com 2020. Arduino High voltage device control. Available at: <https://www.seeedstudio.com/blog/2020/01/03/arduino-tutorial-control-high-voltage-devices-with-relay-modules/> [Accessed 2021-03-01].

W3Techs.com 2021. W3Techs provides survey information about the usage of technologies on the web. W3techs.com Usage statistics of web server.



Available at:[https://w3techs.com/technologies/overview/web\\_server](https://w3techs.com/technologies/overview/web_server)

[Accessed 2020-03-28].

Blog.ptsecurity.com 2017. *Most common Web system attacks. Positive Technologies – learn and secure*. Available at:

<http://blog.ptsecurity.com/2017/09/web-application-attack-statistics-q2.html>

[Accessed 2021-03-05]

Alexander S. Gillis 2019. Definition of Internet of Things (IoT). Available at:

<https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT>

[Accessed 2021-02-29].

Gómez, A., Cuiñas, D., Catalá, P., Xin, L., Li, W., Conway, S., & Lack, D. (2015). Use of single board computers as smart sensors in the manufacturing industry. *Procedia engineering*, 132, 153-159.

Güven, Yılmaz, et al. "Understanding the concept of microcontroller based systems to choose the best hardware for applications." (2017).