

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Дніпровський національний університет залізничного транспорту  
імені академіка В. Лазаряна

Кафедра «Електронні обчислювальні машини»

«ДО ЗАХИСТУ»

Завідувач кафедри

\_\_\_\_\_ (підпис) \_\_\_\_\_ (ПІБ)  
« \_\_\_\_\_ » \_\_\_\_\_ 20 \_\_\_\_ р.

**ДИПЛОМНА РОБОТА**  
на здобуття освітнього ступеня «магістр»

Галузь знань \_\_\_\_\_ 12 \_\_\_\_\_ Інформаційні технології \_\_\_\_\_  
(шифр) (назва)

Спеціальність \_\_\_\_\_ 123 \_\_\_\_\_ Комп'ютерна інженерія \_\_\_\_\_  
(код) (повна назва)

Тема Дослідження та розробка мікропроцесорної системи управління штучною екосистемою. Мобільна та ВЕБ версії програмної частини

Theme Research and development of the artificial ecosystem microprocessor control system. Mobile and WEB version of the software part

Керівник дипломного проекту	ст. викл. _____ (посада) (підпис)	Дзюба В.В. (ПІБ)
Консультант розділу з БЖД	К. Т. Н. _____ (посада) (підпис)	Музикін М.І. (ПІБ)
Нормоконтролер	К. Т. Н. _____ (посада) (підпис)	Шаповалов В.О. (ПІБ)
Студент групи	КС1921 _____ (група) (підпис)	Кирпа Д.Р. (ПІБ)
Student	Курпа Denys (family name)	

Дніпро  
2020

**Дніпровський національний університет  
залізничного транспорту ім. академіка В.Лазаряна**  
**Факультет** Комп'ютерних технологій і систем **кафедра** ЕОМ  
**Спеціальність** Комп'ютерні системи та мережі

**ЗАТВЕРДЖУЮ:**

**зав. кафедрою**

д.т.н., проф. \_\_\_\_\_ (Жуковицький І.В.)

« \_\_\_\_ » \_\_\_\_\_ 2020 р.

**ЗАВДАННЯ**

**до дипломного проекту студента КС 1921 групи  
Кирпи Дениса Романовича**

**1. Тема проекту (роботи):** Дослідження та розробка мікропроцесорної системи управління штучною екосистемою. Мобільна та ВЕБ версії програмної частини

затверджена наказом по університету № 57 ст. від «17» 01. 2020 р.

**2. Термін подання студентом закінченого проекту (роботи) – 10.12.2020 р.**

**3. Вихідні дані до проекту (роботи)** Теоретичні відомості сучасних штучних мікробіомів та екосистем,  
<https://www.w3schools.com/>

**4. Зміст розрахунково-пояснювальної записки (роботи)**

Вступ та постановка завдання

1. Огляд існуючих систем

2. Вибір програмних засобів для реалізації системи

3. Налаштування серверної частини

4. Побудова веб-версії клієнтської частини

5. Охорона праці та безпека в надзвичайних ситуаціях

6. Висновки

**5. Перелік креслень (з переліком обов'язкових креслень)**

Вибір програмних засобів системи

Структурна схема мікропроцесорної системи керування

Загальний алгоритм роботи системи

Структура програмної частини системи

Результати моделювання

## 6. Консультанти (з назвами розділів)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Охорона праці в надзвичайних ситуаціях			

7. Дата видачі завдання - \_\_\_\_\_  
Керівник проекту \_\_\_\_\_ (Дзюба В.В.)  
(підпис)  
Завдання прийняв до виконання \_\_\_\_\_ (Кирпа Д.Р.)  
(підпис)

## КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва розділів дипломного проекту (роботи)	Термін виконання розділів проекту (роботи)	Примітки
1	Вступ та постановка завдання	02.09.2020 р.	2%
2	Огляд існуючих систем	04.10.2020 р.	32%
3	Вибір програмних засобів для реалізації системи	03.11.2020 р	29%
4	Налаштування серверної частини	13.11.2020 р.	10%
5	Побудова веб-версії клієнтської частини	23.11.2020 р.	10%
7	Охорона праці та безпека в надзвичайних ситуаціях	08.12.2020 р.	10%
8	Висновки	10.12.2020 р.	2%

Студент-дипломник \_\_\_\_\_ (Кирпа Д.Р.)

Керівник проекту \_\_\_\_\_ (Дзюба В.В.)

**РЕФЕРАТ**

**Кирпа Д.Р. Дослідження та розробка мікропроцесорної системи управління штучною екосистемою. Мобільна та ВЕБ версії програмної частини.**

Дніпропетровський національний університет залізничного транспорту ім. академіка В. Лазаряна, кафедра електронних обчислювальних машин. – Дипломний проект. – 85 с., 22 рис., 2 табл., 28 джерел., 4 додатки.

В даному дипломному проекті було розроблено та створено клієнт-серверну систему, що дозволить користувачеві взаємодіяти із мікропроцесорною системою управління штучною екосистемою. Проведено огляд існуючих систем та засобів для розробки системи. Розроблено структурні схеми.

## **ESSAY**

**Kirpa D.R. Research and development of the artificial ecosystem microprocessor control system. Mobile and WEB version of the software part.**

Dnipropetrovsk National University of Retail Transport, Department of Electronic Counting Machines. - Graduation project. - 85 p., 22 pictures, 2 tables, 28 sources, 4 supplements.

In this diploma project has been developed a client-server system that allows users to interact with an artificial ecosystem microprocessor control system. Research carried out an overview of the existing systems and the tools for the system development. Block-diagrams was developed.

## ЗМІСТ

Вступ та постановка задачі .....	7
1 Огляд існуючих систем .....	8
1.1 Ninja Blocks .....	8
1.2 LS Monitoring .....	9
1.3 MasterKit .....	10
1.4 Аналіз представлених систем віддаленого керування мікропроцесорними системами .....	11
1.5 Висновки.....	12
2 Вибір програмних засобів для реалізації системи.....	14
2.1 Структурна схема програмних засобів системи.....	14
2.2 MQTT-Брокер Mosquitto .....	16
2.3 База даних InfluxDB .....	18
2.4 Організація передачі даних між Mosquitto та InfluxDB .....	18
2.5 Система аналітики та моніторингу метрик Grafana.....	19
2.6 Платформа Node.js.....	19
2.7 Побудування веб-сторінки.....	20
2.8 Фреймворк Express.js .....	20
2.9 Фреймворк MQTT.js.....	20
2.10 Висновки за розділом .....	20
3 Налаштування серверної частини .....	21
3.1 Встановлення та налаштування MQTT-брокера Mosquitto .....	21
3.2 Встановлення та налаштування бази даних InfluxDB .....	24
3.3 Встановлення та налаштування системи аналітики та моніторингу метрик Grafana.....	26

3.4	Налаштування передачі даних від MQTT-брокера Mosquitto до бази InfluxDB .....	30
4	Побудова веб-версії клієнтської частини .....	33
4.1	Аналіз можливостей програмного додатку .....	33
4.2	Встановлення та налаштування Node.js .....	34
4.4	Висновки за розділом .....	39
5	Охорона праці та безпека в надзвичайних ситуаціях.....	41
5.1	Вимоги безпеки при виконанні робіт на робочому місці.....	41
5.2	Шкідливі виробничі фактори на робочому місці.....	42
5.2.1	Мікроклімат робочого місця .....	43
5.2.2	Освітлення робочого місця .....	44
5.2.3	Шум робочого місця .....	45
5.2.4	Електрична та пожежна безпека .....	46
5.3	Дії працівників в аварійних ситуаціях .....	47
5.4	Висновки за розділом .....	49
	Перелік використаних джерел .....	51

## Вступ та постановка задачі

В сучасному суспільстві мережа Інтернет охоплює все більше і більше сфер нашого життя. Сьогодні, користувачами інтернету є не лише люди, а і велика кількість пристроїв починаючи від звичайних комп'ютерів та мобільних телефонів і закінчуючи автомобілями та цілими будинками. IoT – Internet of Things – саме так названо новий вектор розвитку мережі інтернет. Цей перспективний напрямок передбачає використання існуючої мережі Інтернет для налагодження зв'язку між звичайними речами, що обладнано вбудованими технологіями, що дозволяють «спілкуватися» одне з одним та з оточуючим середовищем. Але, якщо людина захоче взаємодіяти із такими пристроями так само як і вони між собою, стає очевидним що для реалізації такої задумки потрібно використовувати користувацький додаток, що буде встановлено на комп'ютері, смартфоні чи будь якому іншому пристрої користувача.

Задачею даного дипломного проекту є створення системи, що дозволить користувачеві взаємодіяти із мікропроцесорною системою управління штучною екосистемою. Таким чином, насамперед, слід ознайомитися із вже існуючими системами інтернету речей та в особливості - системами так званого «розумного будинку».

## 1 Огляд існуючих систем

### 1.1 Ninja Blocks

Ninja Blocks – це проект, що призначений для реалізації керування і контролю периферійними домашніми пристроями за допомогою модуля Ninja Block на базі одноплатного комп'ютера BeagleBone із використанням авторської плати на основі Atmega328 [4]. Особливістю цієї системи є те, що і апаратна і програмна частини мають відкриті початкові коди. Це означає, що для отримання пристрою не обов'язково використовувати готовий модуль від розробника – його можна зібрати самому. Це також означає що клієнтська та серверна частини можуть бути доповнені новим функціоналом завдяки користувачам.

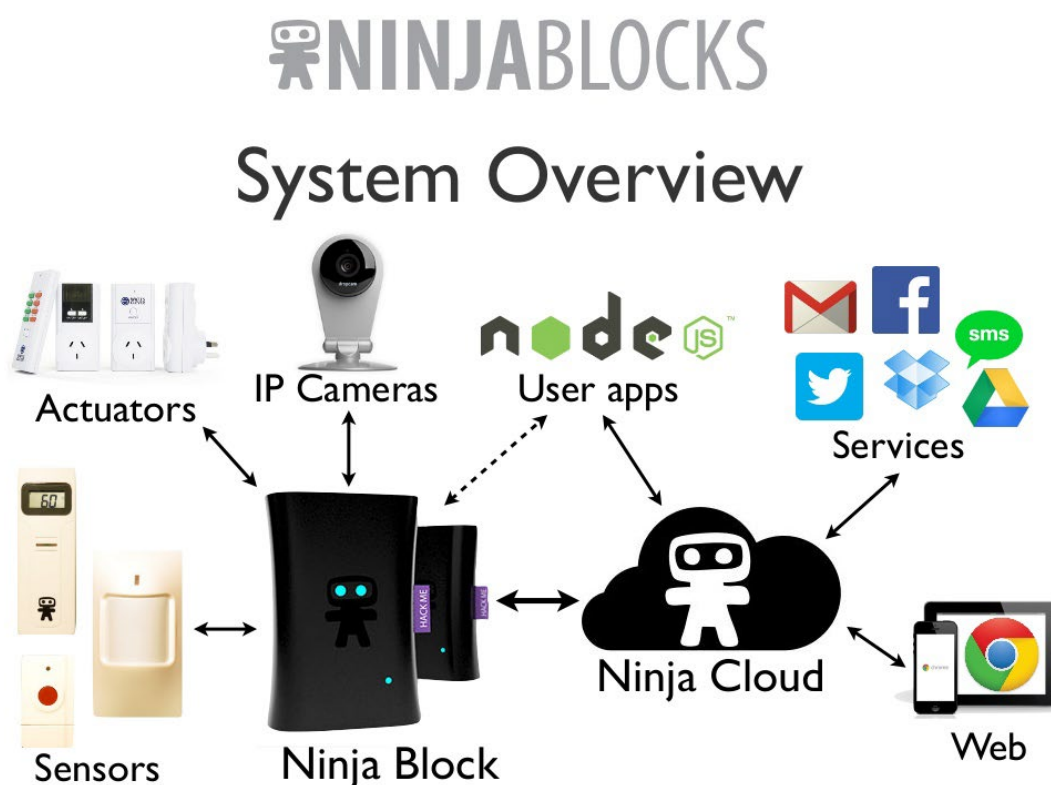


Рисунок 1 – Структурна схема системи NinjaBlocks

Як можна побачити на структурній схемі, центральною частиною цієї системи є сервіс Ninja Cloud, до якого у першу чергу пристрої Ninja Block, що з'єднано з різноманітними датчиками, камерами та перемикачами. Для керування пристроєм користувач може використовувати як веб-додаток або



різноманітні сервіси що підтримуються, так і власні користувацькі додатки написані мовою Node.JS. Користувацькі додатки, опціонально, можуть бути зв'язані із пристроєм напряму, не прибігаючи до використання сервісу Ninja Cloud, що дає можливість працювати системі без виходу до глобальної мережі, наприклад, у місцях що не підключено до мережі Internet.

## 1.2 LS Monitoring

LS Monitoring – це набір-коструктор, що побудовано на базі популярної платформи Arduino, та призначений для створення системи контролю температури з управлінням через мережу Internet [5]. Отриманий пристрій, за словами автора, може бути використано для контролю температури у погребях, теплицях, інкубаторах чи будь-якому іншому віддаленому об'єкті. Комплект готового пристрою продається на офіційному сайті компанії Lazy Smart, але користувач за бажанням може зібрати систему сумісну із хмарним сервісом LS Cloud самостійно, використовуючи велику кількість навчальних відеоуроків, що розміщено також на сайті авторів.

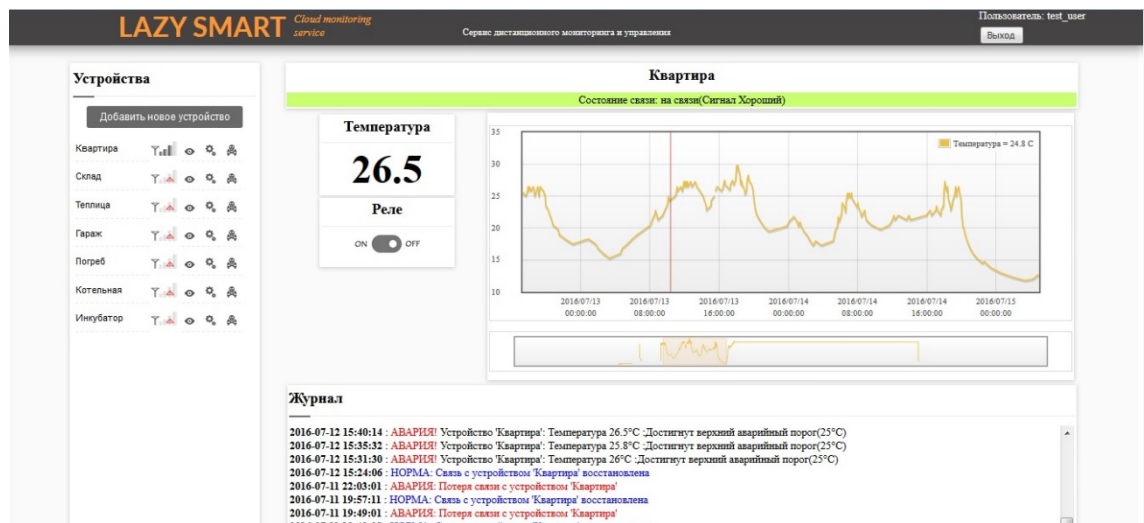


Рисунок 2 – Користувацький інтерфейс сервісу LS Cloud

Вигідною перевагою сервіса моніторингу LS Cloud є те що він є безкоштовним для користування, не прив'язаний до якогось конкретного типу пристрою, що у купі з розробленим API для доступу до сервісу дозволяє самостійно розробляти програми або прилади що будуть працювати з ним, виходячи за рамки пропонуємого компанією пристрою.

### 1.3 MasterKit

Система контролю та управління MasterKit розроблена одноіменною компанією і представляє собою набір модулів системи «розумний дім». Система є відкритою, а тому є можливість використання сторонніх модулів, що не передбачені виробником. Компанія пропонує готові пристрої, такі як розумні розетки, станції контролю якості повітря, модулі керування електроприборами, охоронні системи. [6]



Рисунок 3 – Готові пристрої компанії MasterKit

Також, пропонуються пристрої, для самостійної сборки, такі як розумні реле, що працюють у локальних або глобальних мережах, або модулібору показання лічильників. Усі пристрої для самостійної сборки включають вільно розповсюджуваний комплект розробника, що включає документацію та початкові коди.

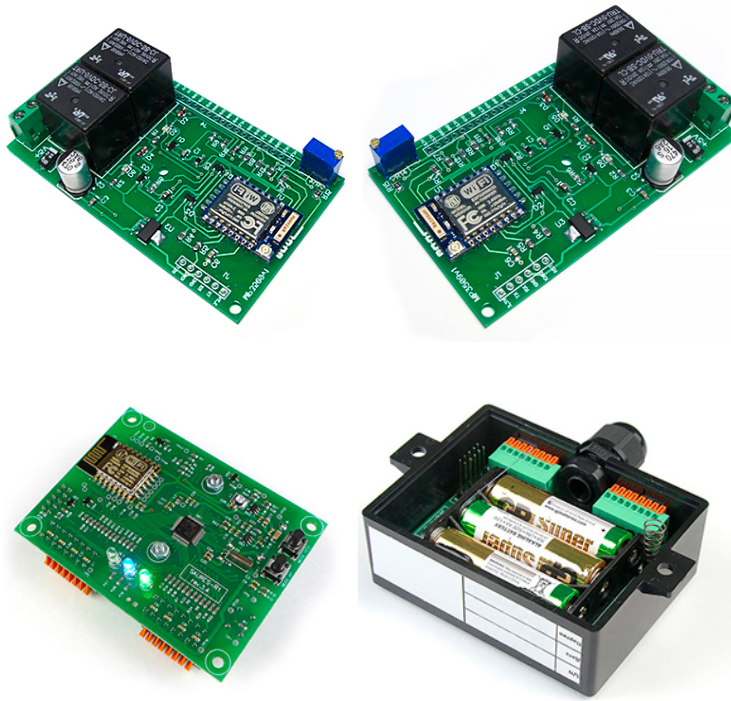


Рисунок 4 – Пристрої для самостійної сборки компанії MasterKit

Тим не менш, головним недоліком цієї системи є те, що усі пристрої працюють незалежно один від одного і не можуть бути зв'язані у єдину систему без додаткових пристроїв через використання різних протоколів та каналів передачі зв'язку. Усі готові рішення мають власний додаток під системи iOS або Android, що мають закриті початкові коди.


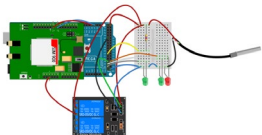

#### 1.4 Аналіз представлених систем віддаленого керування мікропроцесорними системами

Ознайомившись із різними існуючими продуктами, що призначено для керування мікропроцесорних систем, було виділено наступні критерії, якими може володіти розроблювана система:

- Кросс-платформеність – система керування має можливість надати користувачу можливість використання будь-якого пристрою;
- Простота керування – система має простий та зрозумілий інтерфес користувача, для збільшення охопленої аудиторії;
- Стандартизованість – система використовує розповсюджені канали та протоколи зв'язку для надання можливості легко розширити систему;

- **Захищеність** – система має інструменти захисту від несанкціонованого доступу зловмисників;
- **Відкритість** – система має відкриті похідні коди;
- **Розширюваність** – система має підтримку легкого додавання нових пристроїв
- **Вартість** – мінімально можлива вартість системи.

Таблиця 1 – Огляд систем за обраними критеріями

Система Критерії	NinjaBlocks 	LS Monitoring 	MasterKit 
Крос-платформеність	+	+	-
Простота керування	-	+	+
Стандартизованість	+	+	-
Захищеність	+	+	+
Відкритість	-	+	-
Розширюваність	+	+	-
Вартість	200\$	60\$	20-100\$

### 1.5 Висновки

В ході ознайомлення із існуючими на ринку системами керування пристроями побудованими на базі мікропроцесорів, було проведено аналіз їх можливостей, та вирішено, що розроблювана в ході дипломного проекту система, насамперед, має обов’язково відповідати таким критеріям як захищеність та простота керування, також, вважаю необхідними такі критерії

як розширюваність, та відкритість. У свою чергу менш значущими критеріями, якими система може і не володіти вважаю стандартизованість та кросс-платформеність. Виходячи з отриманих результатів, далі буде виконано вибір ПЗ, що буде використовуватися для побудування системи.

## 2 Вибір програмних засобів для реалізації системи

Для реалізації системи потрібні такі програмні засоби що дозволять:

- встановити зв'язок між мікропроцесорною системою та сервером для реалізації двонаправленої передачі даних;
- система управління базами даних, яка надасть можливість зберігати отримані дані;
- система аналітики та моніторингу що дозволить виводити у графічному вигляді дані збережені у базі;
- користувацький веб-додаток, що буде відображати дані із системи аналітики та надасть можливість користувачу керувати мікропроцесорною системою.

Також, в ході вибору програмного забезпечення було враховано що в пріоритеті стоять такі критерії як захищеність, простота керування, розширюваність, та відкритість.

### 2.1 Структурна схема програмних засобів системи

Із зазначених вище засобів на серверна частина має виконувати перші три завдання. У якості операційної системи для сервера використовується Ubuntu Linux 16.04.7. Враховуючи це, в ході дипломного проекту було використано наступне програмне забезпечення:

- MQTT-Брокер Mosquitto;
- База даних InfluxDB;
- Система аналітики та моніторингу метрик Grafana.

Обране програмне забезпечення ідеально відповідає поставленим задачам та розповсюджується як вільне програмне забезпечення, що дозволяє використати його у своєму проекті безкоштовно.

В ході вибору варіантів реалізації клієнтської частини, важливим фактором є те, що клієнтський додаток повинен мати можливість відправляти

MQTT-повідомлення для управління деякими пристроями штучної екосистеми. Тому, для створення додатку було вирішено використовувати платформу Node.js із використанням додаткових фреймворків Express.js та MQTT.js.

Більш детально із схемою взаємодії обраного програмного забезпечення можна ознайомитись на схемі нижче

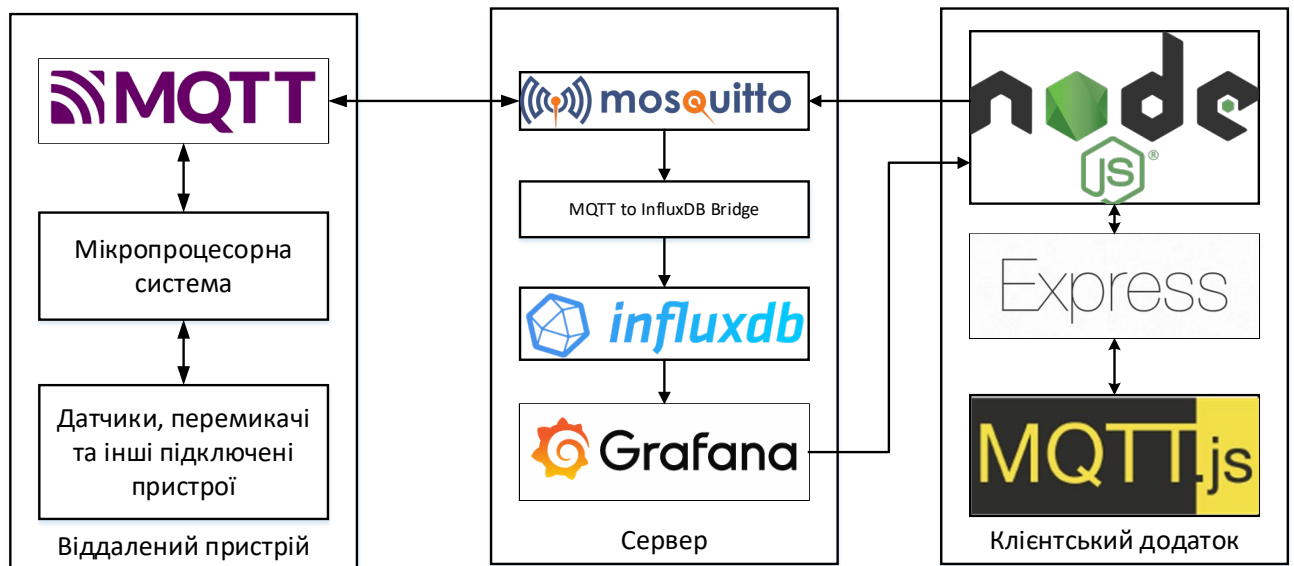


Рисунок 5 – Структурна схема програмної частини системи

У сукупності обране програмне забезпечення працює наступним чином:

- Мікропроцесорна система здійснює обмін даними з датчиками, перемикачами та ін.
- Мікропроцесорну систему за допомогою реалізації протокола MQTT на віддаленому пристрої з використанням MQTT-брокера Mosquitto підключено до сервера для організації прийому/передачі повідомлень;
- За допомогою скрипта MQTT To InfluxDB дані метрик, отримані з датчиків віддаленого пристрою передаються від MQTT-брокера Mosquitto до бази даних часових рядів InfluxDB;
- InfluxDB передає дані метрик до системи аналітики та моніторингу метрик Grafana;

- Клієнтський додаток, написаний з використанням платформи Node.js із використанням фреймворків Express.js та MQTT.js приймає дані метрик із системи Grafana та встановлює зв'язок з MQTT-брокером Mosquitto для організації прийому/передачі повідомлень за допомогою технології Websocket;

## 2.2 MQTT-Брокер Mosquitto

MQTT (Message Queuing Telemetry Transport) – це протокол TCP/IP для передачі повідомлень за моделю «видавець-підписник». Його було розроблено досить давно – у 1999 році Енді Стенфорд-Кларком із компанії IBM та Арленом Ніппером із Cirrus Link [7]. Ідея, що була основоположною при його розробці – підтримка зв'язку між пристроями у мережах з нестабільним з'єднанням або низькою полосою пропускання. Враховуючи це, протокол зроблено неперевантаженим та легким – він ідеально підходить для пристроїв що мають обмежену обчислювальну потужність та/або час автономної роботи. Саме завдяки цьому, приблизно з 2014 року, у момент зростання популярності «Інтернету речей», коли з'явилася потреба у легкому і простому протоколі, він почав широко використовуватися. Це не означає, що MQTT є єдиним протоколом такого роду, існують і схожі протоколи, наприклад Web Application Messaging Protocol, Streaming Text-Oriented Messaging Protocol або Alternative Message Queueing Protocol, але саме MQTT отримав найбільше розповсюдження.

Система зв'язку у протоколі MQTT побудовано з таких складових: сервер-видавець, сервер-брокер та один чи декілька клієнтів-підписників. Використовуються наступні поняття:

- Topic – тема, з якою відправлено/отримано повідомлення (message)
- Message – повідомлення, що містять деяку інформацію
- Publish – передача повідомлення брокеру
- Subscribe – підписка на отримання повідомлень від брокера



- QoS (Quality Of Service) – пріоритет передачі даних
- Retain – дублювання старих повідомлень новим

Порядок роботи наступний: сервер-видавець (publisher) надає запит на підключення до сервера-брокера (broker), у разі якщо брокер відповідає задовільно, видавець передає повідомлення (message) з визначеною темою (topic) і змістом до брокера. В свою чергу, клієнт-підписник (subscriber) також робить запит на підключення до сервера-брокера, і у разі якщо брокер відповідає задовільно, клієнт робить запит до брокера на підписку (subscribe) отримання повідомлень із визначеною темою. У разі надходження на брокер повідомлення з темою, яку на яку підписаний клієнт, повідомлення разом із його вмістом передається до клієнта. При всьому цьому, доставка кожного повідомлення контролюється за допомогою двох фалгів QoS та Retain.

QoS дозволяє публікувати повідомлення наступним чином:

0 – публікація повідомлень не більше одного разу – сервер передає повідомлення і забуває про нього, при цьому повідомлення можуть бути втрачені чи продубльовані;

1 – публікація повідомлень хоча-б раз – сервер передає повідомлення і отримувач підтверджує доставку, при цьому повідомлення можуть бути продубльовані, але доставка гарантована;

2 – публікація повідомлень лише раз – сервер забезпечує доставку, при цьому повідомлення не можуть бути втрачені чи продубльовані.

Retain, в свою чергу налаштовує дублювання старих повідомлень, що зберігаються на брокері до нових підписників.

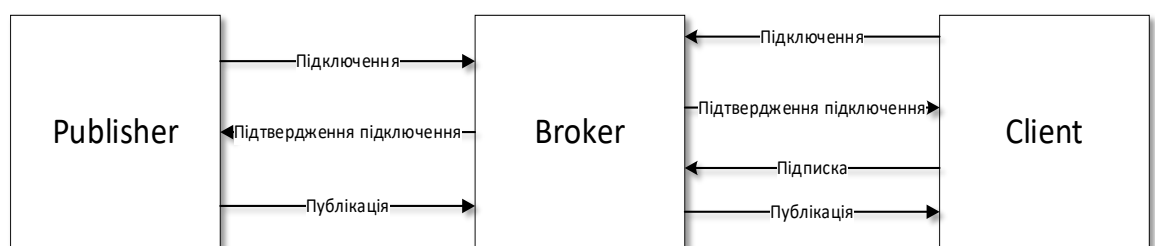


Рисунок 6 – Схема роботи протоколу MQTT

Виходячи з вищезазначеного, до одного брокера можуть бути підключені декілька видавників та клієнтів, що є дуже зручним. В свою чергу, сервер-видавець не потребує жодних налаштувань стосовно розміщення або кількості підписників та навпаки – у підписників немає потреби налаштування на конкретного видавника.

У ході виконання дипломного проекту буде використано MQTT-брокер Eclipse Mosquitto актуальної версії.

Вигідною відмінністю Mosquitto від інших є відкритість початкових кодів, що дає можливість скопіювати його з початкових кодів із ввімкненою опцією підтримки протоколу WebSocket, що в подальшому потребується для побудови веб-клієнту.

### 2.3 База даних InfluxDB

В ході вибору програмного забезпечення, що буде використовуватися на сервері, для зберігання даних, отриманих з пристрою було обрано базу даних InfluxDB.

InfluxDB – база даних для зберігання часових рядів з відкритим походним кодом. Це означає, що база створена для зберігання великих об'ємів даних з часовими мітками, що ідеально підходить для моніторингу та збору метрик з різноманітних датчиків в системах «розумного будинку» [8].

База даних не має зовнішніх залежностей і надає користувачу схожий на SQL синтаксис. На відміну від SQL, InfluxDB:

- не має операції update – для цього потрібно виконати вставку до бази із тим самим ключем (в якості ключа використовується часова мітка);
- підтримка фільтрації для видалення з використанням тегів або часу, але не стовпців;
- фільтрація при додаванні нового тегу для існуючих даних недоступний.

### 2.4 Організація передачі даних між Mosquitto та InfluxDB

Для організації передачі даних від MQTT-брокера Mosquitto до бази даних InfluxDB буде написано на мові Python скрипт, що виступає у ролі

«мосту». З похідним кодом цього скрипту можна ознайомитись у додатках до дипломного проекту.

## 2.5 Система аналітики та моніторингу метрик Grafana

Grafana – це система аналітики та моніторингу метрик з відкритим похідним кодом [9]. Завдяки тому, що Grafana підтримує базу даних InfluxDB у своєму стандартному вигляді, налаштування передачі метрик з бази виконується без використання стороннього програмного забезпечення. Також, варто відмітити що Grafana має API для використання графіків, що побудовано за допомогою цієї системи на зовнішніх веб-сторінках, що ідеально підходить для проекту.



Рисунок 7 – Користувачський інтерфейс система аналітики та моніторингу метрик Grafana

## 2.6 Платформа Node.js

Node.js – це платформа, з відкритим похідним кодом, що заснована V8 JavaScript Engine [10]. Головною особливістю Node є те, що на відміну від звичайного JavaScript, який використовується для обробки даних лише у браузері користувача, вона надає можливість виконувати скрипти, написані

мовою JavaScript на сервері та відправляти користувачеві лише результат їх виконання.

## 2.7 Побудування веб-сторінки

Особливістю побудування веб сторінки є те, що клієнт повинен працювати на безлічі різноманітних пристроїв під керуванням багатьох операційних систем. Таким чином, для однаково коректного відображення даних на різних пристроях потрібно використовувати адаптивний веб-дизайн.

## 2.8 Фреймворк Express.js

Express.js (також просто Express) – це відкритий фреймворк під ліцензією MIT спроектований для створення веб-додатків та API. Використовується у якості стандартного каркасу Node.js та має можливість виступати у ролі бекенда для набору MEAN (MongoDB, Express.js, Angular.js, Node.js) що використовується для розробки веб-додатків мовою Javascript [11].

## 2.9 Фреймворк MQTT.js

Для реалізації передачі MQTT-повідомлень напряду з веб-сторінки використовується фреймворк MQTT.js. Передача даних до брокера надходить із використанням протоколу WebSocket [12].

WebSocket – це незалежний протокол зв'язку, що засновано на протоколі TCP та призначено для обміну повідомленнями між браузером та веб-сервером в режимі реального часу [13].

## 2.10 Висновки за розділом

В ході виконання цього розділу дипломного проекту було проведено вибір програмного забезпечення для реалізації системи на основі критеріїв, що були отримані у попередньому розділі. Також, в ході вибору програмного забезпечення було описано структуру взаємодії цього програмного забезпечення в рамках системи. На цьому, вважаю можливим перехід до реалізації проекту.

### 3 Налаштування серверної частини

Налаштування серверної частини виконується за допомогою вбудованого у систему Ubuntu Linux 16.04.7 терміналу.

#### 3.1 Встановлення та налаштування MQTT-брокера Mosquitto

В ході проектування було визначено що MQTT-брокер повинен мати підтримку протоколу Websocket. Eclipse Mosquitto у стандартному вигляді підтримки цього протоколу немає, тому для отримання такого функціоналу потрібно скомпілювати його з похідних кодів.

Для того щоб розпочати компілювання брокера з похідних кодів встановлюю необхідні для компіляції пакети:

```
denissx@denissx-LIFEBOOK-E734:~$ sudo apt-get install cmake libwrap0-dev  
libc-ares-dev uuid-dev zlib1g-dev libssl-dev
```

- `cmake` це кросплатформена система автоматизації сборки ПЗ з похідних кодів, написаних на мові C
- `libwrap0-dev` включає пакет `tcp-wrappers`, що Eclipse Mosquitto потребує для компіляції;
- `libc-ares-dev` включає пакет `c-ares`, що Eclipse Mosquitto потребує для компіляції;
- `uuid-dev` включає пакет `libuuid`, що Eclipse Mosquitto потребує для компіляції;
- `zlib1g-dev` включає пакет `lib1g`, що потребує Eclipse Mosquitto для компіляції;
- `libssl-dev` встановлено через те, що він потребується для компіляції бібліотеки Websockets та Eclipse Mosquitto

Далі за допомогою наступних команд завантажую і розпаковую похідні коди бібліотеки Websockets та переходжу до директорії [15]:

```
denissx@denissx-LIFEBLOCK-E734:~$ cd ./Заврузки
```

```
denissx@denissx-LIFEBLOCK-E734:~/Заврузки$ wget  
https://github.com/warmcat/libwebsockets/archive/v1.6.2.tar.gz
```

```
denissx@denissx-LIFEBLOCK-E734:~/Заврузки$ tar -xzf v1.6.2.tar.gz
```

```
denissx@denissx-LIFEBLOCK-E734:~/Заврузки$ cd libwebsockets-1.6.2
```

Для компіляції бібліотеки створюю каталог build:

```
denissx@denissx-LIFEBLOCK-E734:~/Заврузки/libwebsockets-1.6.2$ mkdir build
```

```
denissx@denissx-LIFEBLOCK-E734:~/Заврузки/libwebsockets-1.6.2$ cd build
```

За допомогою наступних команд компілюємо та встановлюємо пакет:

```
denissx@denissx-LIFEBLOCK-E734:~/Заврузки/libwebsockets-1.6.2/build$  
cmake .. -DLIB_SUFFIX=64
```

```
denissx@denissx-LIFEBLOCK-E734:~/Заврузки/libwebsockets-1.6.2/build$ make
```

```
denissx@denissx-LIFEBLOCK-E734:~/Заврузки/libwebsockets-1.6.2/build$ sudo  
make install
```

Для перевірки інсталяції пакету використовую команду:

```
denissx@denissx-LIFEBLOCK-E734:~/Заврузки/libwebsockets-1.6.2/build$ pkg-  
config --modversion libwebsockets
```

Відповідь містить версію встановленого пакета:

1.6.2-

За допомогою наступних команд завантажую і розпаковую похідні коди MQTT-брокера Eclipse Mosquitto та переходжу до директорії:

```
denissx@denissx-LIFEBOOK-E734:~/Загрузки$ tar xzvf mosquitto-1.4.7.tar.gz
```

```
denissx@denissx-LIFEBOOK-E734:~/Загрузки$ cd mosquitto-1.4.7
```

За допомогою текстового редактора nano у файлі config.mk знаходжу параметр WITH\_WEBSOCKETS:=no та зміню його на WITH\_WEBSOCKETS:=yes

```
denissx@denissx-LIFEBOOK-E734:~/Загрузки/mosquitto-1.4.7$ nano config.mk
```

Для компіляції брокера створюю каталог build:

```
denissx@denissx-LIFEBOOK-E734:~/Загрузки/mosquitto-1.4.7$ mkdir build
```

```
denissx@denissx-LIFEBOOK-E734:~/Загрузки/mosquitto-1.4.7$ cd build
```

За допомогою наступних команд компілюю та встановлюю скомпільований пакет:

```
denissx@denissx-LIFEBOOK-E734:~/Загрузки/mosquitto-1.4.7/build$ cmake ..
```

```
denissx@denissx-LIFEBOOK-E734:~/Загрузки/mosquitto-1.4.7/build$ make
```

```
denissx@denissx-LIFEBOOK-E734:~/Загрузки/mosquitto-1.4.7/build$ sudo make install
```

Для перевірки коректності інсталяції запускаю пакет mosquitto за допомогою наступної команди:

```
denissx@denissx-LIFEBOOK-E734:~/Загрузки/mosquitto-1.4.7/build$ mosquitto -h
```

У відповіді отримую наступне повідомлення:

mosquitto version 1.4.7 (build date 2020-11-10 12:46:11+0200)

mosquitto is an MQTT v3.1 broker.

Usage: mosquitto [-c config\_file] [-d] [-h] [-p port]

-c : specify the broker config file.

-d : put the broker into the background after starting.

-h : display this help.

-p : start the broker listening on the specified port.

Not recommended in conjunction with the -c option.

-v : verbose mode - enable all logging types. This overrides any logging options given in the config file.

See <http://mosquitto.org/> for more information.

Для подальшого автоматичного запуску брокера із системою виконую наступну команду

```
denissx@denissx-LIFEBOOK-E734:~/Зарпузки/mosquitto-1.4.7/build$ sudo  
systemctl enable mosquitto
```

### 3.2 Встановлення та налаштування бази даних InfluxDB

Для встановлення та налаштування бази даних InfluxDB, для початку, додаю репозиторій до утиліти apt наступними командами:

```
denissx@denissx-LIFEBOOK-E734:~$ wget -qO-  
https://repos.influxdata.com/influxdb.key | sudo apt-key add -  
  
denissx@denissx-LIFEBOOK-E734:~$ source /etc/os-release
```



```
denissx@denissx-LIFEBOOK-E734:~$ echo "deb
https://repos.influxdata.com/debian $(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/influxdb.list
```

Далі, оновлюю apt з урахуванням нового репозиторію та встановлюю InfluxDB наступною командою:

```
denissx@denissx-LIFEBOOK-E734:~$ sudo apt update && sudo apt install -y
influxdb
```

Для налаштування, запускаю базу за допомогою команди та отримую відповідь:

```
denissx@denissx-LIFEBOOK-E734:~$ influx
```

```
Connected to http://localhost:8086 version 1.8.3
```

```
InfluxDB shell version: 1.8.3
```

```
>
```

За допомогою наступних команд створю базу даних mqtt та додаю користувача mqtt\_usr із повним доступом до цієї бази:

```
> create database mqtt
```

```
> use mqtt
```

```
> create user mqtt_usr with password 'mqtt_usr' with all privileges
```

```
> grant all privileges on mqtt to mqtt_usr
```

За допомогою наступних команд перевіряю наявність користувача:

```
> show users
```

```
user      admin
```

```
-----
```

```
mqtt_usr true
```

Для налаштування автоматичного запуску InfluxDB разом із системою використовую наступні команди:

```
denissx@denissx-LIFEBOOK-E734:~$ sudo systemctl unmask influxdb.service
```

```
denissx@denissx-LIFEBOOK-E734:~$ sudo systemctl start influxdb
```

```
denissx@denissx-LIFEBOOK-E734:~$ sudo systemctl enable influxdb.service
```

### 3.3 Встановлення та налаштування системи аналітики та моніторингу метрик Grafana

Аналогічно до InfluxDB додаю репозиторій Grafana до утиліти apt:

```
denissx@denissx-LIFEBOOK-E734:~$ wget -q -O -
```

```
https://packages.grafana.com/gpg.key | sudo apt-key add -
```

```
denissx@denissx-LIFEBOOK-E734:~$ echo "deb
```

```
https://packages.grafana.com/oss/deb stable main" | sudo tee
```

```
/etc/apt/sources.list.d/grafana.list
```

Далі, оновлюю apt вже з урахуванням репозиторію Grafana та встановлюю її наступною командою:

```
denissx@denissx-LIFEBOOK-E734:~$ sudo apt update && sudo apt install -y grafana
```

Для автоматичного запуску Grafana разом із системою використовую наступні команди:

```
denissx@denissx-LIFEBOOK-E734:~$ sudo systemctl unmask grafana-server.service
```

```
denissx@denissx-LIFEBOOK-E734:~$ sudo systemctl start grafana-server
```

```
denissx@denissx-LIFEBOOK-E734:~$ sudo systemctl enable grafana-server.service
```

Налаштування імпорту даних з InfluxDB до Grafana виконується за допомогою веб-інтерфейса останньої. Для цього, через веб браузер підключаємося до адреси `http://localhost:3000`

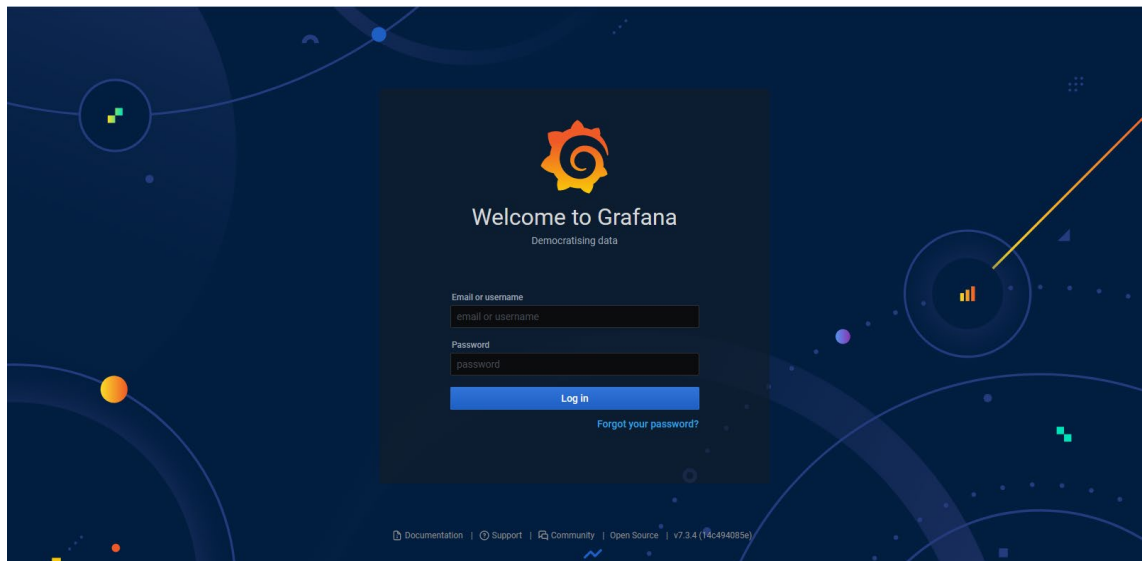


Рисунок 8 – Сторінка авторизації Grafana

Використовуючи стандартний логін `admin` з паролем `admin` входжу до системи. На наступному екрані система пропонує змінити стандартний пароль, змінюю його та натискаю Submit

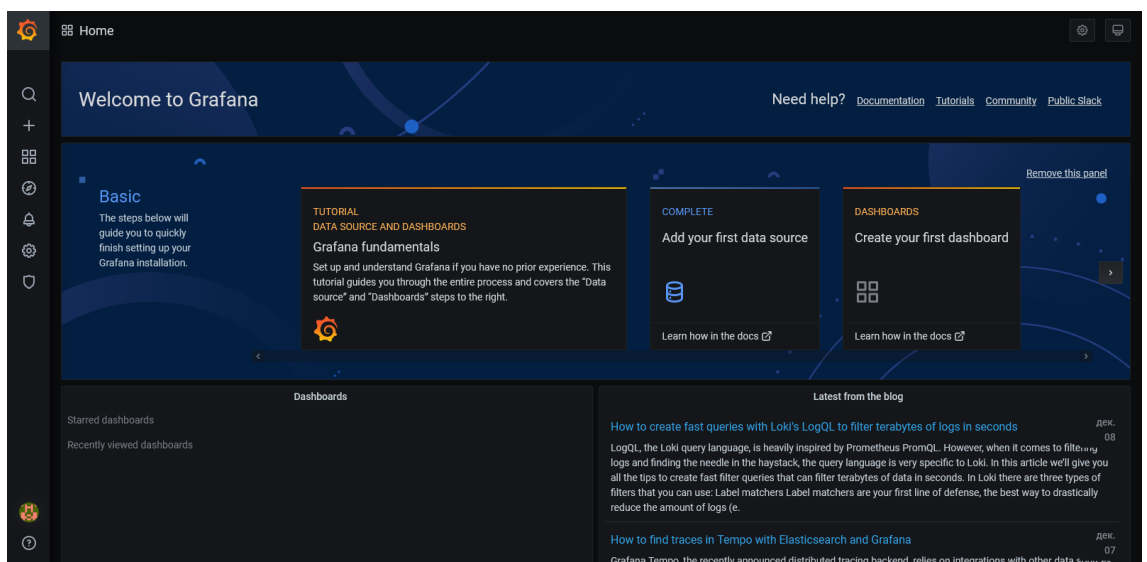


Рисунок 9 – Головний екран Grafana

У головному меню знаходжу пункт `Configuration => Data Sources`

Натиснувши кнопку Add Data Source шукаю InfluxDB та натискаю Select та виконую налаштування імпорту з бази даних mqtt, створений у InfluxDB

The image consists of four screenshots of the InfluxDB Data Sources configuration interface, arranged vertically. The first screenshot shows the 'Settings' tab for an InfluxDB data source. It includes fields for 'Name' (InfluxDB), 'Query Language' (InfluxQL), 'HTTP URL' (http://localhost:8086), 'Access' (Server (default)), and 'Whitelisted Cookies'. The second screenshot shows the 'Auth' section with options for 'Basic auth', 'TLS Client Auth', 'Skip TLS Verify', and 'Forward OAuth Identity'. The third screenshot shows the 'InfluxDB Details' section with fields for 'Database' (mqtt), 'User' (mqtt\_usr), 'Password' (masked), and 'HTTP Method' (GET). The fourth screenshot shows the 'Database Access' section with a warning message about database access and a 'Min time interval' field set to 10s. At the bottom, there are buttons for 'Save & Test', 'Delete', and 'Back'.

Рисунок 10 – Налаштування джерела даних InfluxDB

Налаштування виводу метрик проводиться наступним чином:

За допомогою пункту Create => Dashboard створюється дошка

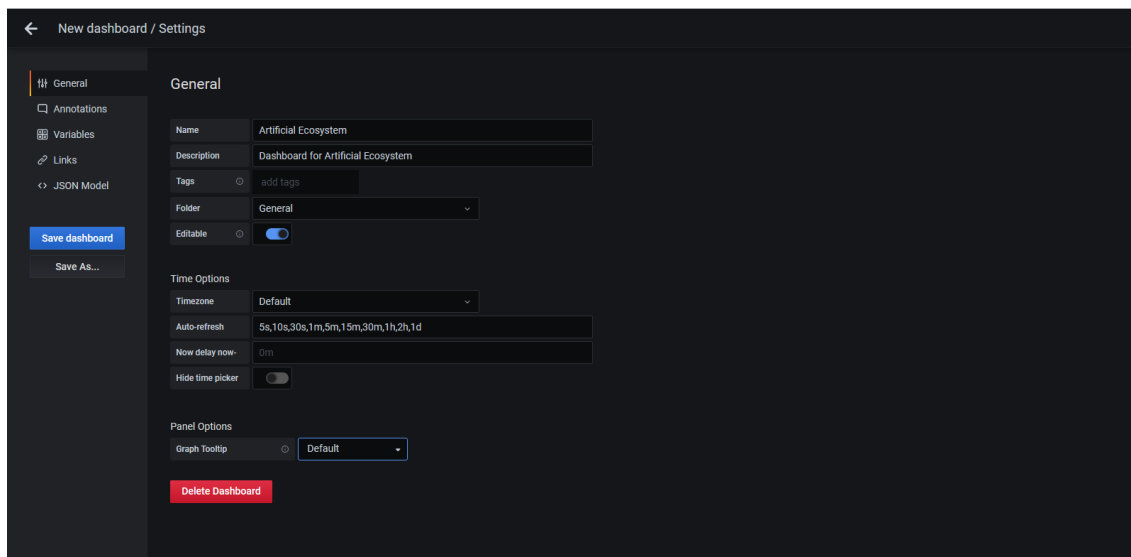


Рисунок 11 – Створення дошки у Grafana

Далі за допомогою пункта меню Add Panel створюється нова панель з вибором джерела даних InfluxDB, та налаштовується на отримання даних конкретного датчика з бази.

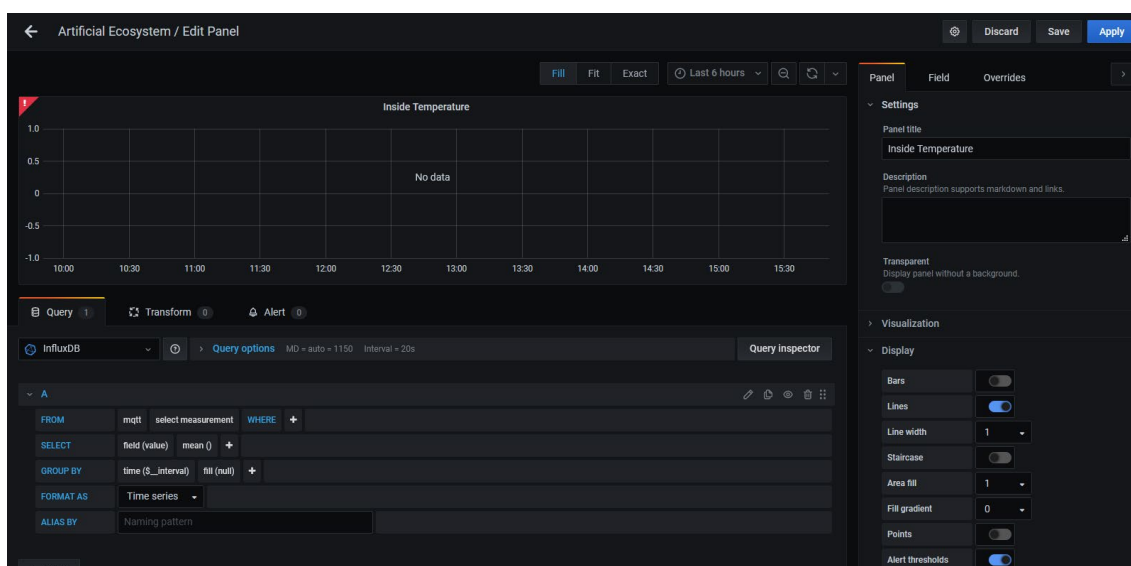


Рисунок 12 – Створення панелі у Grafana

Ці дії повторюються для усіх датчиків що використовуються у пристрої. В ході дипломного проектування буде використано такі метрики:

Для подальшого виводу метрик на веб-сторінку, потребується встановлення плагіну Grafana Image Renderer наступною командою

```
denissx@denissx-LIFEBOOK-E734:$ grafana-cli plugins install grafana-image-renderer.
```

Після встановлення плагіна потребується перезавантажити сервер за допомогою команди:

```
denissx@denissx-LIFEBOOK-E734:~$ sudo service grafana-server restart
```

Вивід метрик на сторонні ресурси у Grafana реалізовано за допомогою вбудовування сгенерованої сторінки до веб-додатку користувача.

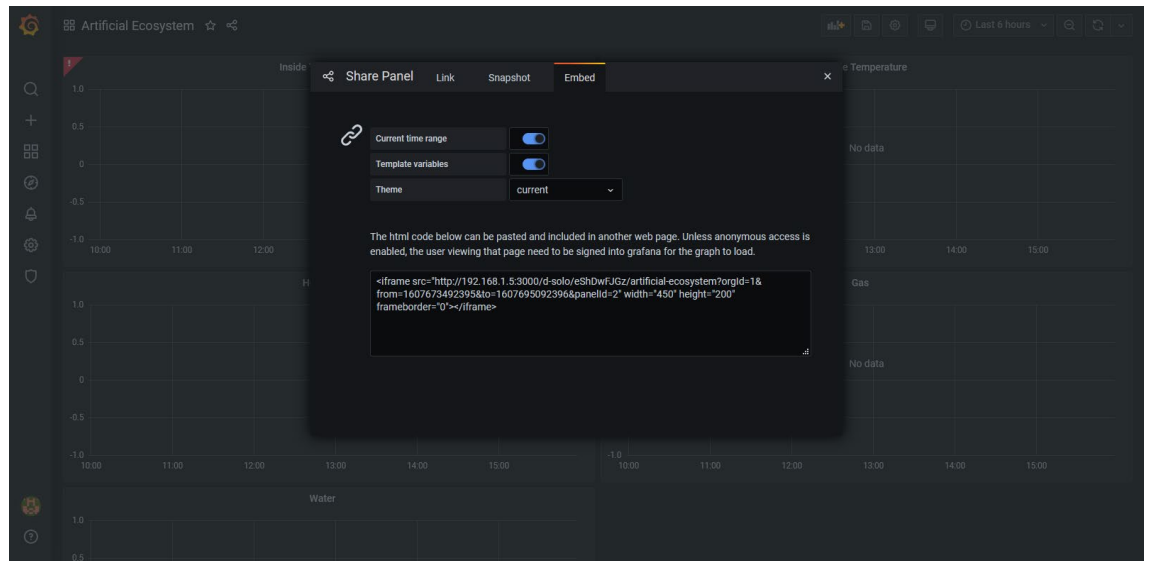


Рисунок 13 – Екран експорту панелі Grafana

### 3.4 Налаштування передачі даних від MQTT-брокера Mosquitto до бази InfluxDB

Для реалізації передачі даних від MQTT-брокера Mosquitto до бази InfluxDB було розроблено Python скрипт, що виступає у ролі MQTT-клієнта та передає дані до бази mqtt що знаходиться у InfluxDB. Ubuntu Linux 16.04.7 має підтримку Python 3 у стандартному пакеті поставки, але потребується встановлення менеджера інсталяції пакетів Python 3 – PIP. Виконується інсталяція за допомогою наступної команди:

```
denissx@denissx-LIFEBOOK-E734:~$ sudo apt install -y python3-pip
```

Для встановлення підготовленого пакету використано наступну команду:

```
denissx@denissx-LIFEBOOK-E734:~$ pip install mypackage --no-index --find-links file:///home/denissx/mqtttoinfluxdb
```

Далі, створюється конфігураційний файл `mqtt2influxdb.yml` у каталозі `/etc/denissx` та редагується за допомогою текстового редактора `nano`:

```
denissx@denissx-LIFEBOOK-E734:~$ sudo mkdir /etc/denissx
```

```
denissx@denissx-LIFEBOOK-E734:~$ sudo nano
```

```
/etc/denissx/mqtttoinfluxdb.yml
```

Файл налаштувань має наступний зміст:

```
mqtt:
  host: 127.0.0.1
  port: 1883

influxdb:
  host: 127.0.0.1
  port: 8086
  database: mqtt
  username: mqtt_usr
  password: mqtt_usr

points:
  - measurement: tempi
    topic: srv/+temp_i
    fields:
      value: $.payload
    tags:
      id: $.topic[1]
  - measurement: tempo
    topic: srv/+temp_o
    fields:
      value: $.payload
    tags:
      id: $.topic[1]
  - measurement: humidity
    topic: srv/+humidity
    fields:
      value: $.payload
    tags:
      id: $.topic[1]
  - measurement: gas
    topic: srv/+gas
    fields:
      value: $.payload
    tags:
      id: $.topic[1]
  - measurement: water
    topic: srv/+water
    fields:
      value: $.payload
    tags:
      id: $.topic[1]
```

Рисунок 13 – Конфігураційний файл скрипта-мосту (продовження)

Із похідними кодами скрипта можна ознайомитись у додатках.

### 3.5 Висновки за розділом

В ході виконання даного розділу дипломного проекту було виконано встановлення та налаштування програмного забезпечення, що використовується в якості серверної частини системи. Даний набір програмного забезпечення може працювати самостійно і автономно.



## 4 Побудова веб-версії клієнтської частини

### 4.1 Аналіз можливостей програмного додатку

В ході розробки додатку вирішено виконати аналіз можливостей веб-додатку та порівняти за обраними критеріями з потенціальними тонкими клієнтами для Google Android та Apple iOS.

Основними критеріями для порівняння було обрано такі властивості:

- швидкість роботи – наскільки швидко додаток працює;
- функціональність – кількість функціоналу, яка буде доступна користувачеві;
- простота взаємодії – придатність додатка для комфортного використання користувачем;
- час розробки – затрати часу на розробку;
- вартість розробки – затрати коштів на розробку;

Таблиця 2 – Порівняння клієнтів для різних платформ

Платформа Критерій	Android	iOS	Web
Швидкість роботи	+	+	-
Функціональність	+	+	+/-
Простота взаємодії	+	+	+
Час розробки	-	+	+
Вартість розробки	+	-	+

Детальніше за кожним пунктом:

- Тонкі клієнти під операційні системи Android та iOS будуть працювати у декілька разів швидше, ніж клієнт, що запущено у браузері на цих-же системах;

- Функціональність клієнта на даний момент не відрізняється на усіх платформах, але в перспективі Android та iOS мають більше потенційно реалізуємого функціоналу;
- У будь-якому клієнті взаємодія з користувачем повинна бути однаково зручна, але використання тонких клієнтів зменшує кількість дій, що потрібно зробити для користування клієнтом;
- Через фрагментованість системи Android час розробки сильно зростає бо змушує розробника враховувати велику кількість моделей пристроїв під управлінням цієї системи, котрі можуть дуже сильно відрізнятися одне від одного;
- Система iOS, на відміну від Android не має проблеми фрагментації пристроїв, але головним негативним фактором є необхідність капіталовкладень для початку розробки під цю систему.

## 4.2 Встановлення та налаштування Node.js

Для побудування веб-версії клієнтської частини було використано фронтенд, написаний за допомогою HTML+CSS із використанням Javascript та бекенд, що використовує платформу Node.js. За допомогою наступних команд було виконано інсталяцію Node.js та менеджера пакетів Node.js – npm:

```
denissx@denissx-LIFEBOOK-E734:~$ sudo apt install nodejs
```

```
denissx@denissx-LIFEBOOK-E734:~$ sudo apt install npm
```

Для перевірки коректності інсталяції використовую наступні команди, що виводять версію встановленого додатку та отримано відповіді:

```
denissx@denissx-LIFEBOOK-E734:~$ nodejs -v
```

```
v4.2.6
```

```
denissx@denissx-LIFEBOOK-E734:~$ npm -v
```

3.5.2

Для початку роботи з Node.js створюю директорію проекту та за допомогою менеджера npm встановлюю необхідні фреймворки:

```
denissx@denissx-LIFEBOOK-E734:~$ mkdir mqtt_app
```

```
denissx@denissx-LIFEBOOK-E734:~$ cd mqtt_app
```

```
denissx@denissx-LIFEBOOK-E734:~/mqtt_app$ npm install express --save
```

```
denissx@denissx-LIFEBOOK-E734:~/mqtt_app$ npm install mqtt --save
```

```
denissx@denissx-LIFEBOOK-E734:~/mqtt_app$ npm install body-parser --save
```

Таким чином, у директорії з'являються файл `package.json` що включає перелік модулів, що встановлено для проекту і директорію `node_modules` що включає усі необхідні модулі.

#### 4.3 Огляд отриманого додатку

В процесі розробки було отримано додаток під назвою `Artificial Ecosystem`

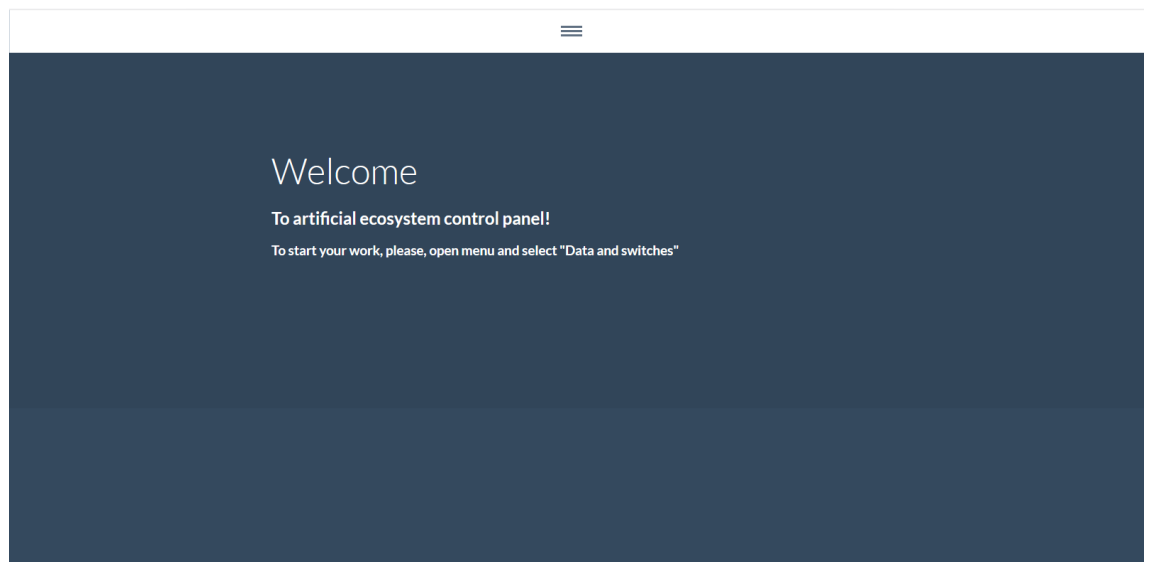


Рисунок 14 – Головна сторінка додатку на комп'ютері

У верхній частині екрану розташовано клавiшу меню, при наведенні на яку, буде з'являтися частина бокового меню

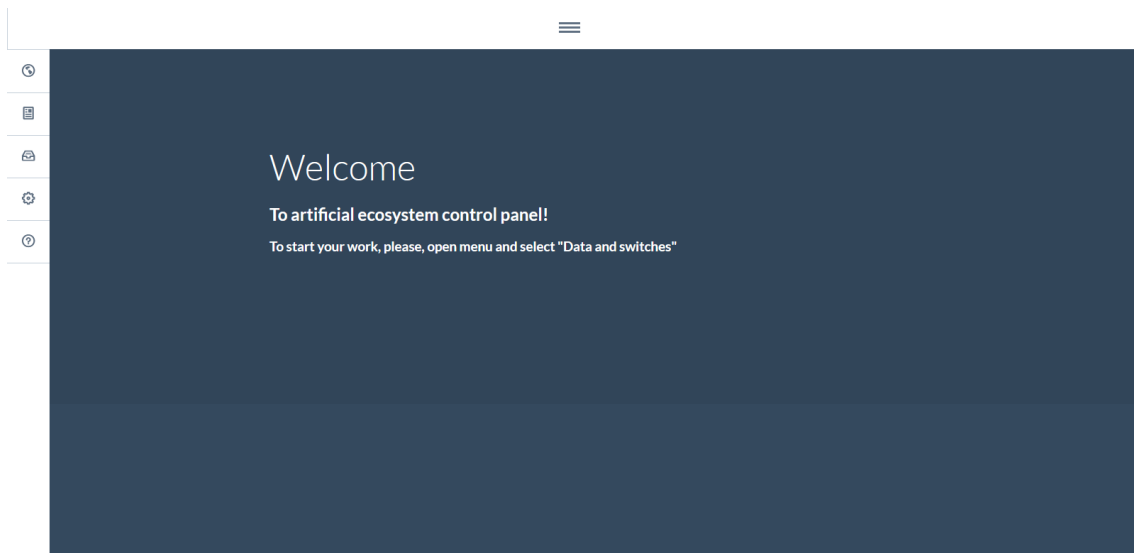


Рисунок 15 – Поява головного меню при наведенні на клавiшу меню

При натисканні на клавiшу меню відкриється

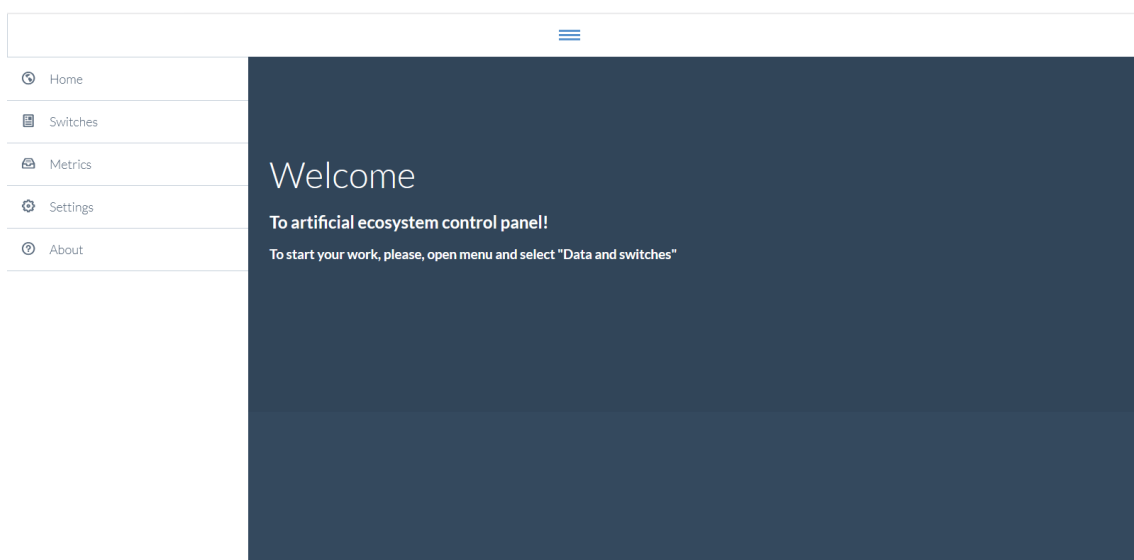


Рисунок 16 – Головне меню додатка

У відкритому меню представлено п'ять пунктів:

Home – представляє собою стартову сторінку додатку

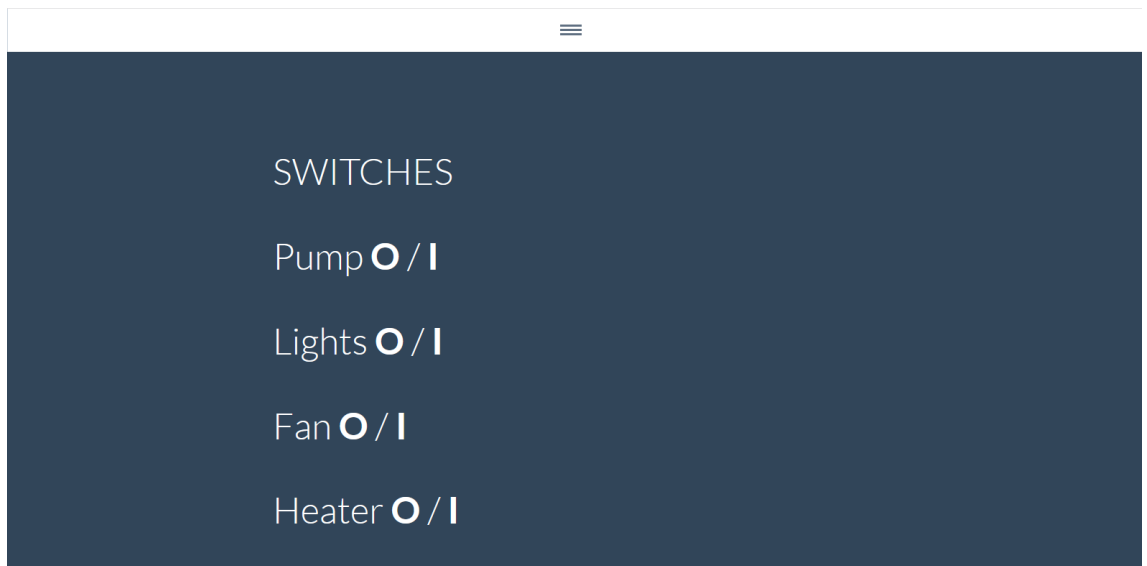


Рисунок 17 – Меню Switches

Пункт меню Switches меню до уваги користувача представлено перемикачі, що дають можливість примусово керувати пристроями штучної екосистеми, а саме водяною помпою, лампою освітлення, вентилятором та підігрівачем. Це може бути корисно наприклад, для перевірки пристроїв на роботоспособність або під час обслуговування.



Рисунок 18 – Меню Metrics

Меню Metrics відображає дані з системи аналітики та моніторингу метрик Grafana, що було отримано з датчиків зовнішньої та внутрішньої температури, датчику вмісту вуглекислого газу, датчику освітлення та датчику вологості ґрунту

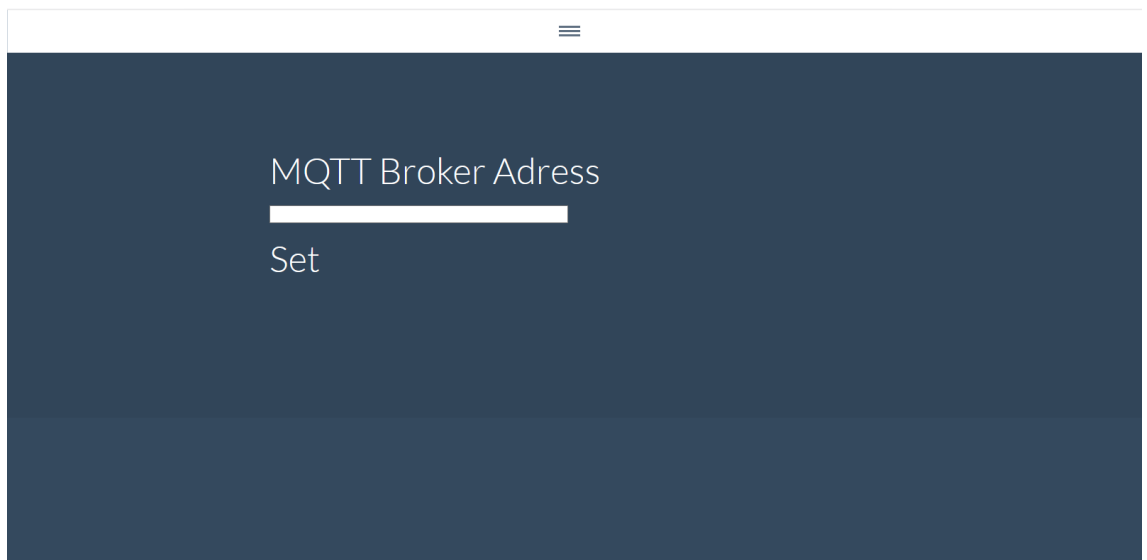


Рисунок 19 – Меню Settings

Пункт меню Settings дозволяє користувачеві встановити адресу MQTT-брокера з яким буде організовано обмін повідомленнями.

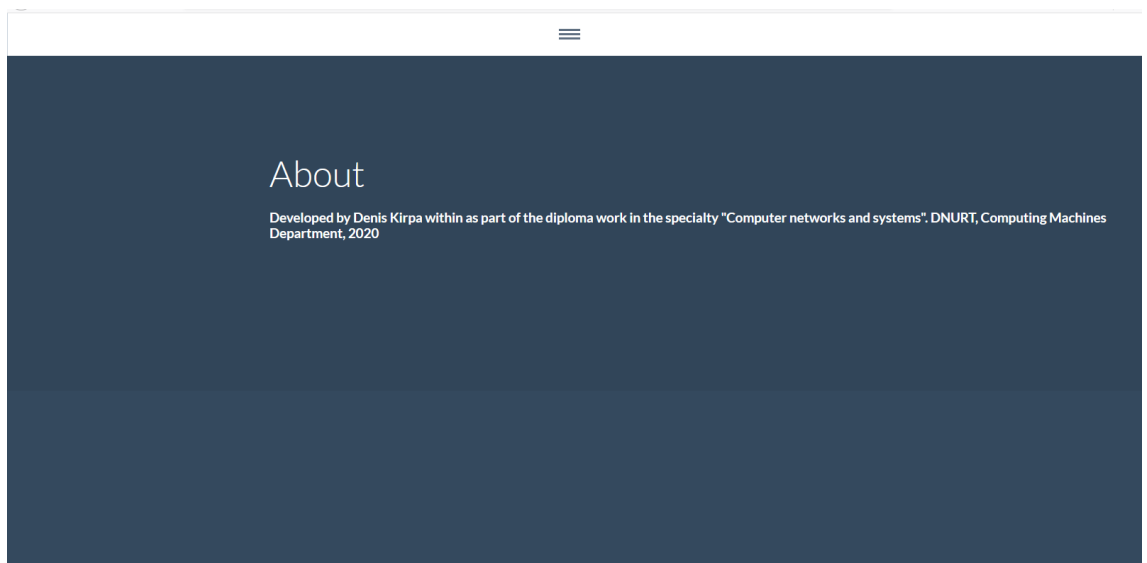


Рисунок 20 – Меню About

About Включає у себе короткі відомості про додаток.

Додаток створено з використанням адаптивного дизайну

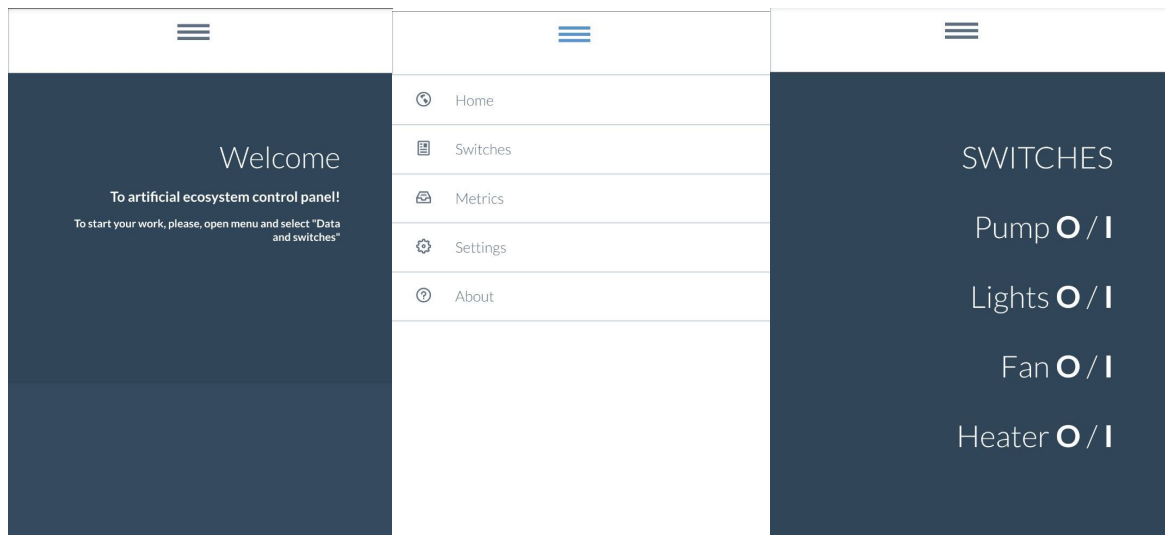


Рисунок 21 – Приклад відображення додатку на мобільному пристрої

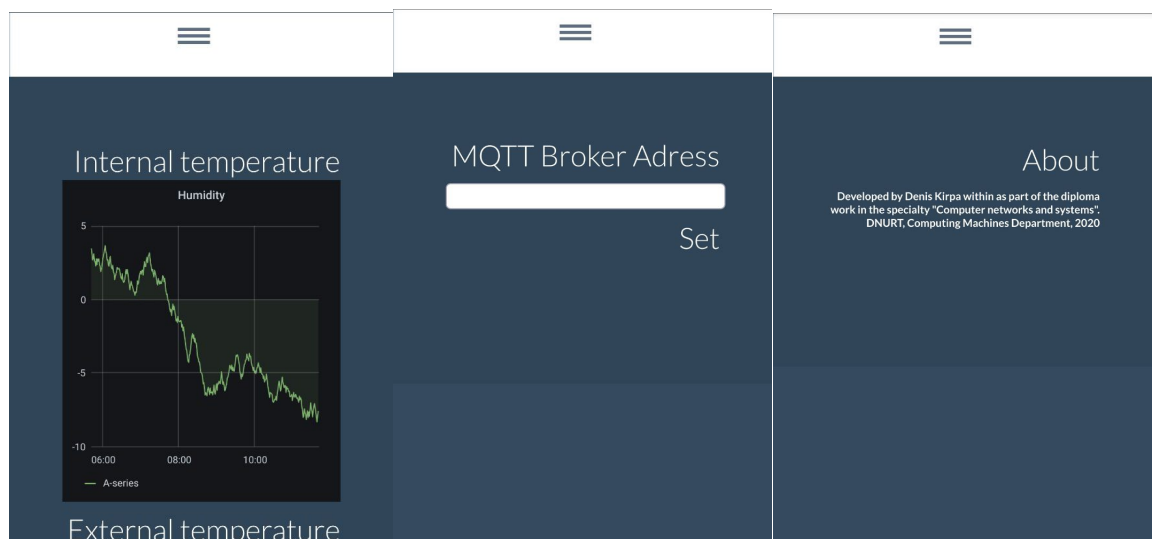


Рисунок 22 – Приклад відображення додатку на мобільному пристрої

Завдяки використанню адаптивного дизайну незалежно від операційної системи, браузера або розрішення екрану він буде мати однаково цілісний вигляд.

#### 4.4 Висновки за розділом

В ході виконання даного розділу дипломного проекту було створено веб-додаток, що повністю задовольняє поточним потребам користувачів. Від розробки тонких клієнтів під системи iOS та Android було вирішено відмовитися через більшу ефективність використання веб-клієнта. Веб-додаток було розроблено за допомогою HTML+CSS та Node через можливість

простого інтегрування із MQTT-Брокером Mosquitto та системою збору аналітики і метрик Grafana. Швидкість роботи додатка не так явно відрізняється від тонких клієнтів, а функціонування під усіма операційними системами що мають можливість встановлення сучасного веб браузерa є несумнівним плюсом.



## 5 Охорона праці та безпека в надзвичайних ситуаціях

В ході виконання дипломного проекту створюється клієнт-серверна система моніторингу та керування штучною екосистемою. З цього витікає те, що основним інструментом під час розробки та використання цієї системи буде комп'ютер. Комп'ютер може становити загрозу для життя користувача при неправильному користуванні. Саме тому, важливо притримуватися системи, спрямованої на запобігання шкідливого впливу на здоров'я та життя людини – Охорони праці.

Згідно із ст. 1 Закону України «Про охорону праці» [16] – охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності.

### 5.1 Вимоги безпеки при виконанні робіт на робочому місці

Згідно з НПАОП 0.00-1.28-10 «Про затвердження правил охорони праці під час експлуатації електронно-обчислювальних машин» [17] до роботи з комп'ютерною технікою не слід допускати персонал, що не пройшли інструктаж та перевірку знань з охорони праці, пожежної безпеки та правил експлуатації ЕОМ.

Комп'ютерна техніка працює від мережі змінного струму 220В, шкідливий вплив якого повинна враховувати охорона праці. Згідно з ГОСТ 12.1.030-81 ССБТ «Електробезпека. Захисне заземлення. Занулення» [18] для запобігання виникнення небезпеки протікання струму через неструмоведучі металеві частини корпусу здійснюється підключення до пристрою захисного. Захисне заземлення – це з'єднання металевих не струмоведучих частин електроустановок із землею через заземлюючі провідники та заземлювачі для створення між цими частинами і землею малого опору. Захисне заземлення

включає у свої конструкції металічні труб діаметром 25-50 мм, довжиною 2-3 м, що вбито на 4-6 м одна від одної і з'єднано між собою металічною трубою, остання проходить в приміщення, з'єднується з внутрішнім контуром заземлення.

## 5.2 Шкідливі виробничі фактори на робочому місці

Шкідливий виробничий фактор – це такий фактор, дія якого може призвести до захворювання або зниження працездатності в певних умовах.

В залежності від тривалості та рівня впливу шкідливі виробничі фактори можуть бути класифіковані як небезпечні. [19]

Небезпечний виробничий фактор – це такий фактор, дія якого може призвести до травм або погіршення здоров'я в певних умовах.

Шкідливі виробничі фактори, як правило, в основному розділяють на чотири групи:

- хімічні;
- біологічні;
- фізичні;
- психофізіологічні.

При роботі з комп'ютером можна виділити фактори з таких двох груп як фізичні і психофізіологічні:

- підвищений рівень електромагнітного випромінювання;
- підвищений рівень статичної напруги;
- підвищений рівень шуму та вібрації;
- підвищене значення рівня напруги для живлення комп'ютерів, замикання у ланцюзі якого може пройти через тіло людини;
- нервово-психічні перевантаження такі як: монотонність праці, розумове перенапруження, перенапруження органів зору

Види професійного ризику, можуть викликати психічні, нервові та серцево-судинні види захворювань через можливі стресові ситуації, що виникають під час роботи.

Виходячи з цього, доцільно передбачити планування оздоровчих заходів та поліпшення умов праці, що нададуть можливість зменшення ризику виникнення та розвитку психічних захворювань, розладів серцево-судинної системи, гіпертонії, неврозів та інших захворювань.

#### 5.2.1 Мікроклімат робочого місця

Згідно санітарних норм ДСН 3.3.6-042-99 «Державні санітарні норми мікроклімату виробничих приміщень» [20], є обов'язковим дотримання встановлених параметрів мікроклімату у приміщеннях, в яких встановлено комп'ютерне обладнання. Ці параметри залежать від характеру виробничого приміщення, характеру трудового процесу та пори року.

Розміри робочого місця визначаються згідно з вимогами ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами ЕОМ» [21]. Площа робочого місця повинна мати не менш ніж 6м<sup>2</sup>, а об'єм - не менш ніж 20м<sup>3</sup>.

Надзвичайно важливе значення надається місцю роботи персоналу. Для людини дуже важливим фактором є підтримка постійної температури тіла. Основним принципом створення мікроклімату є надання оптимальних умов для організації теплообміну тіла людини із навколишнім середовищем.

Комп'ютерна техніка, через особливості своєї конструкції, є джерелом виділення тепла, та завдяки цьому є основною зниження рівня вологості та підвищення температури у приміщенні. Також, комп'ютерна техніка, при взаємодії із побутовим пилом може викликати подразнення слизової оболонки очей, носа або шкіри, що може бути причиною виникнення запалювальних

процесів в організмі людини. Таким чином, для попередження цього використовуються наступні заходи:

- застосовується система вентиляції і кондиціонування повітря;
- проводиться вологе прибирання приміщень.

При проведенні вологих прибирань у приміщеннях де розташована комп'ютерна техніка слід бути обережним. Контакт техніки із рідинами може призвести до короткого замикання та ураження струмом.

#### 5.2.2 Освітлення робочого місця

Відповідно до ДБН В.2.5-28-2006 «Природне і штучне освітлення» [22] існують наступні види освітлення:

- природне;
- штучне;
- поєднане.

Природне освітлення - це денне світло що потрапляє до приміщення ззовні завдяки передбаченим конструкцією місцям. В основному, характеризується залежністю від часу дня, пори року і ряду інших чинників.

Штучне освітлення - це освітлення, що використовується замість природного у темний час доби або при умовах недостатнього значення світлового коефіцієнту, наприклад через короткий світловий день.

Поєднане освітлення - це природне освітлення, що використовується разом із штучним при умовах недостатнього значення світлового коефіцієнту.

Штучне освітлення поділяється на:

- робоче;
- аварійне;
- евакуаційне;
- охоронне.

Робоче освітлення поділяють на:

- загальне;
- комбіноване.

Приміщення в яких розташована комп'ютерна техніка повинні мати природне і штучне освітлення. При недостатньому освітленні людина може втрачати увагу, навантажувати очі, що може призвести до появи стомленості. Також, використання занадто яскравого освітлення може викликати подразнення очей або засліплення.

Некоректно направлене світло може дезорієнтувати людину, створюючи тіні та відблиски на робочому місці. Таким чином, однією з важливих складових у травм, захворювань та нещасних випадків є правильний розрахунок освітленості у приміщенні.

Вимоги до освітленості в приміщеннях при виконанні зорових робіт різної точності:

- високої точності - загальна освітленість - 300лк, комбінована - 750лк;
- високої точності - загальна освітленість - 200лк, комбінована - 300лк.

Основною гігієнічною вимогою є рівномірне освітлення усього поля зору людини через те що яскраве світло в районі периферійного зору збільшує напруженість очей що призводить до їх швидкої стомлюваності.

### 5.2.3 Шум робочого місця

Згідно ДСТУ 2867-94. «Шум. Методи оцінювання виробничого шумового навантаження. Загальні вимоги» [23] шум є фактором що може

погіршити умови праці, завдаючи шкоди на організм людини. В умовах тривалого впливу шуму людина може відчувати головні білі, стомленість, дратливість, біль у вухах тощо, що може привести до негативних наслідків.

Також, під впливом шуму на організм людини зменшується концентрація уваги, порушуються фізіологічні функції, у зв'язку з підвищеними енергетичними витратами і нервово-психічним напруженням з'являється втома.

Згідно ДСН 3.3.6.037-99. «Санітарні норми виробничого шуму, ультразвуку та інфразвуку» [24] рівень шуму на робочому місці програмістів не повинен перевищувати 50 дБ. Для зменшення рівня шуму приміщення рекомендовано оклеїти звукоізоляційними матеріалами.

#### 5.2.4 Електрична та пожежна безпека

Згідно ДСТУ 2267-93 «Вироби електротехнічні. Терміни та визначення» [25] технічне обладнання повинно відповідати I класу захисту за способом захисту людей від ураження електричним струмом.

Приміщення для роботи з технічним обладнанням відносять до категорії приміщень з підвищеною небезпекою, оскільки є можливість ураження електричним струмом. У приміщенні з технічним обладнанням на видному і доступному місці встановлюється аварійний резервний вимикач, що дозволяє повністю відключити електричне живлення приміщення, за винятком освітлення.

Засоби захисту від ураження електричним струмом:

- електрична ізоляція струмоведучих частин;
- захисне заземлення;
- захисне відключення.

Все що підлягає заземленню, повинно бути під'єднано до шини заземлення окремими провідниками.

Приміщення повинно відповідати нормативам з вогнестійкості будівельних конструкцій, плануванні будівель та оснащеністю пристроями протипожежного захисту відповідно до ДСТУ Б В.1.1-36:2016 «Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою» [26].

Профілактика пожежі передбачає забезпечення пожежної безпеки обладнання, електроустановок, систем опалення та вентиляції, запобігання утворення та внесення джерел запалювання, утворення горючого середовища. Система пожежного захисту передбачає застосування негорючих і важкозаймистих матеріалів, ізоляції займистого середовища, застосування засобів для гасіння пожежі, пожежної сигналізації та сповіщення про пожежу, застосування засобів захисту людей, організацію пожежної охорони об'єкта.

Згідно ДБН В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги» будівлі та приміщення, де розміщені робочі місця програмістів, мають бути не нижче II ступеня вогнестійкості [27].

### 5.3 Дій працівників в аварійних ситуаціях

Під час роботи з комп'ютером, найбільш вірогідними надзвичайними ситуаціями можуть бути «Виникнення пожежі» або «Ураження електричним струмом».

Згідно наказу «Про затвердження порядків надання домедичної допомоги особам при невідкладних станах» [28] при наданні домедичної допомоги постраждалим з опіками необхідно:

- переконатися у відсутності небезпеки та провести огляд постраждалого, встановити чи знаходиться людина у свідомості та перевірити дихання;
- викликати бригаду екстреної (швидкої) медичної допомоги;
- у разі відсутності дихання у постраждалого, розпочати проведення серцево-легеневої реанімації;
- у разі опіків від першого до другого ступеня необхідно охолодити місце опіку холодною водою, після цього накрити пошкоджену ділянку чистою вологою серветкою. Проколювати пухирі категорично забороняється у разі їх розриву необхідно накласти стерильну пов'язку;
- у разі опіків від третього до четвертого ступеня необхідно накласти на місце опіку стерильну пов'язку та за наявності ознак шоку надати постраждалому протишокове положення;
- забороняється використовувати мазі, гелі та інші засоби до прибуття бригади швидкої медичної допомоги;
- забезпечити постійний нагляд за постраждалим до приїзду бригади швидкої медичної допомоги;
- у разі погіршення стану постраждалого до приїзду бригади швидкої медичної допомоги повторно зателефонувати диспетчеру екстреної медичної допомоги.

Також, при наданні домедичної допомоги постраждалим при ураженні електричним струмом та блискавкою [28] необхідно:

- переконавшись у відсутності небезпеки у разі перебування постраждалого під дією струму, при можливості припинити його дію: вимкнути джерело струму, відкинути електричний провід за допомогою сухої дерев'яної палиці чи іншого електронепровідного засобу;



- провести огляд постраждалого, визначити наявність свідомості, дихання;
- викликати бригаду екстреної (швидкої) медичної допомоги;
- у разі відсутності дихання у постраждалого, потрібно розпочати проведення серцево-легеневої реанімації;
- у разі втрати свідомості постраждалим, при наявності дихання, необхідно надати постраждалому стабільного положення;
- необхідно обробити місця опіку стерильними пов'язками; забезпечити постійний нагляд за постраждалим до приїзду бригади швидкої медичної допомоги;
- у разі погіршення стану постраждалого до приїзду бригади швидкої медичної допомоги повторно зателефонувати диспетчеру екстреної медичної допомоги.

#### 5.4 Висновки за розділом

В даному розділі було розглянуто вимоги щодо робочого місця програміста-розробника. Проаналізовано вимоги щодо: мікроклімату, освітлення, шуму, електричної та пожежної безпеки, а також розглянуто дії, що проводяться в аварійних ситуаціях.

## ВИСНОВКИ

В результаті виконання дипломного проекту було створено клієнт-серверну систему, що дозволяє користувачеві взаємодіяти із мікропроцесорною системою управління штучною екосистемою. В ході розробки проаналізовано існуючі системи на ринку. Після виділення основних критеріїв якими повинна володіти система, виконано вибір необхідного для реалізації такої системи програмного забезпечення. Після налаштування серверної частини системи, під час розробки додатку, в результаті аналізу критеріїв що повинні бути притаманними тонким клієнтам для операційних систем iOS та Android а також веб-клієнту, було вирішено використовувати веб-клієнт через більшу ефективність.

## Перелік використаних джерел

1. Кантелон М., Хартер М., Головайчук Т., Райлих Н. Node.js в действии. 2-е издание: навч. посіб. Санкт-Петербург : Питер, 2018. 432 с.
2. П. Бэрри. Изучаем программирование на Python: навч. посіб. Москва : Издательство «Э», 2017. — 624 с.
3. Д. Дакетт. HTML и CSS. Разработка и дизайн веб-сайтов: навч. посіб. Москва: Ескімо, 2017. — 480 с.
4. Ninja Blocks: Connect your world with the web. by Ninja Blocks — Kickstarter. URL: <https://www.kickstarter.com/projects/ninja/ninja-blocks-connect-your-world-with-the-web> (дата звернення: 01.12.2020)
5. LS Constructor. URL: <http://lazysmart.ru/LSMonitor/> (дата звернення: 01.12.2020)
6. Wi-Fi устройства контроля и управления Мастер Кит. URL: <https://masterkit.ru/blog/articles/wi-fi-ustrojstva-kontrolya-i-upravleniya-master-kit> (дата звернення: 01.12.2020)
7. Протокол MQTT: концептуальное погружение / Хабр. URL: <https://habr.com/ru/post/463669/> (дата звернення: 01.12.2020)
8. InfluxDB: Purpose-Built Open Source Time Series Database | InfluxData. URL: <https://www.influxdata.com/> (дата звернення: 01.12.2020)
9. Grafana: The open observability platform | Grafana Labs. URL: <https://grafana.com/> (дата звернення: 01.12.2020)
10. Node.js. URL: <https://nodejs.org/uk/> (дата звернення: 01.12.2020)
11. Express - фреймворк веб-приложений Node.js. URL: <https://expressjs.com/ru/> (дата звернення: 01.12.2020)

12. MQTT.js · GitHub. URL: <https://github.com/mqttjs> (дата звернення: 01.12.2020)
13. W3Schools Online Web Tutorials. URL: <https://www.w3schools.com/> (дата звернення: 01.12.2020)
14. GitHub - warmcat/libwebsockets: canonical libwebsockets.org networking library. URL: <https://github.com/warmcat/libwebsockets> (дата звернення: 01.12.2020)
15. «Про охорону праці»: Закон України від 14.10.1992 № 2694-ХІІ.
16. НПАОП 0.00-7.15-18. «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями»: Наказ Міністерство соціальної політики Екранії від 14.02.2018 № 207
17. ДСТУ Б В.2.5-82:2016. «Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом»: Наказ Міністерства регіонального розвитку, будівництва та житлово-комунального господарства від 01.07.2016 р. № 204
18. ДСТУ 2293:2014. «Охорона праці. Терміни та визначення основних понять»: Наказ Мінекономрозвитку України від 02.12.2014 р. № 1429.
19. ДСН 3.3.6.042-99. «Державні санітарні норми мікроклімату виробничих приміщень»: Постанова Головного державного санітарного лікаря України від 01.12.1999 № 42.
20. ДСанПІН 3.3.2.007-98. «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» Постанова Головного державного санітарного лікаря України від 10 грудня 1998 р. № 7.

21. ДБН В.2.5-28:2018. "Природне і штучне освітлення": Наказ Міністерства регіонального розвитку, будівництва та житлово-комунального господарства України 03.10.2018 № 264

22. ДСТУ 2867-94. «Шум. Методи оцінювання виробничого шумового навантаження. Загальні вимоги».

23. ДСН 3.3.6.037-99. «Санітарні норми виробничого шуму, ультразвуку та інфразвуку».

24. ДСТУ 2267-93. «Вироби електротехнічні. Терміни та визначення».

25. ДСТУ Б В.1.1-36:2016. «Визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою»: Наказ Міністерства регіонального розвитку, будівництва та житлово-комунального господарства України від 15.06.2016 № 158.

26. ДБН В.1.1-7:2016. «Пожежна безпека об'єктів будівництва. Загальні вимоги» : Наказ Мінрегіону України від 31.10.2016 № 287.

27. «Про затвердження порядків надання домедичної допомоги особам при невідкладних станах»: Наказ Міністерства охорони здоров'я України 16.06.2014 № 398.