Co-funded by the Tempus Programme of the European Union

DesIRE

А.В. Пархоменко, А.В. Туленков, О.В. Соколянський,
Я.І. Залюбовський, А.В. Пархоменко

ПРОГРАМНО-АПАРАТНА ПЛАТФОРМА ДЛЯ НАВЧАННЯ ТЕХНОЛОГІЯМ ІНТЕРНЕТУ РЕЧЕЙ

Навчальний посібник

Запоріжжя
2017

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Запорізький національний технічний університет

Пархоменко А.В., Туленков А.В., Соколянський О.В., Залюбовський Я.І., Пархоменко А.В.

ПРОГРАМНО-АПАРАТНА ПЛАТФОРМА ДЛЯ НАВЧАННЯ ТЕХНОЛОГІЯМ ІНТЕРНЕТУ РЕЧЕЙ

Навчальний посібник

Запоріжжя
Дике Поле
2017

УДК 004.738.5:338.46:004.853
П-78

Рекомендовано до друку Вченою радою Запорізького національного технічного університету (протокол № 8 від 27 березня 2017 року)

Рецензенти:

Теслюк В.М., доктор технічних наук, професор кафедри систем автоматизованого проектування Національного університету «Львівська політехніка».

Тарасов О.Ф., доктор технічних наук, професор, завідувач кафедри комп'ютерних інформаційних технологій Донбаської державної машинобудівної академії

П-78 **Пархоменко А. В., Туленков А.В., Соколянський О.В., Залюбовський Я І., Пархоменко А.В.**
Програмно-апаратна платформа для навчання технологіям Інтернету речей: навчальний посібник.–Запоріжжя: Дике Поле, 2017. – 120 с.

ISBN 978-617-7433-29-2

Розглянуто особливості сучасних технологій та платформ Інтернету речей. Представлено програмно-апаратну платформу REIoT для навчання технологіям проектування вбудованих систем та систем типу Розумний будинок. Наведено практичні приклади розробки різних схем, пристроїв, а також програмного забезпечення для систем домашньої автоматизації та моніторингу ресурсів. Посібник призначений для студентів вищих навчальних закладів, які навчаються за спеціальностями «Комп'ютерні науки», «Інженерія програмного забезпечення», а також може використовуватись інженерами, аспірантами і студентами різних спеціальностей для отримання знань та практичного досвіду розробки вбудованих систем та реалізації технологій Інтернету речей та пристроїв.

Посібник видано за підтримки міжнародного проекту DESIRE «Розробка курсів з вбудованих систем з використанням інноваційних віртуальних підходів для інтеграції науки, освіти та промисловості в Україні, Грузії та Вірменії» (544091-TEMPUS-1-2013-1-BE-TEMPUS-JPCR) за програмою TEMPUS Європейської комісії.
Проект фінансується за підтримки Європейської Комісії. Зміст даної публікації/матеріалу відображає думку авторів і Європейська Комісія не несе відповідальність за використання інформації, що в ній міститься.

ISBN 978-617-7433-29-2

ЗМІСТ

ПЕРЕДМОВА/PREFACE

The goal of the international educational TEMPUS project DESIRE «Development of Embedded System courses with implementation of Innovative virtual approaches for integration of research, education and production in Ukraine, Georgia and Armenia» (http://tempus-desire.eu/) is to adapt current curricula from theoretical and desk-top application based curricula towards a practice oriented and embedded systems based education. Embedded systems (ES) design and production demands a very specific, qualitative and valuable knowledge growth in target HEIs, which ensures efficient implementation of high-skilled people in the labor market.

As known, ES can be used in conjunction with sensors and actuators for collecting the physical information and turning the collected or received data into suitable actions. Also embedded systems can use a range of technologies to connect with other devices or the Internet. That is why ES are the basis of Internet of Things (IoT) infrastructure.

In this case it is possible to distinguish several practically-oriented educational tasks: learning of ES software and hardware, analysis of existing approaches to ES design, studying of the principles of ES interaction and connection to the Internet

Often, the concept of IoT is inseparably connected with something smart: Smart House, Smart Transport, Smart City, Smart Businesses and so on. Technologies for creating a Smart House are interesting and useful for students, as they allow to make our life more comfortable, safe and to save on domestic energy needs.

REIoT complex was created by our partners from ZNTU for IoT technology studies and investigations. The set-up includes two parts - laboratory RELDES (REmote Laboratory for Development of Embedded Systems) and Smart House&IoT laboratory. Integration of this software/hardware platform into the educational process expands opportunities of distance learning and gives the students all advantages of remote experiments. Practically-oriented teaching methods based on REIoT set-up usage will provide students with the necessary knowledge and skills for the successful realization of IoT technologies in their future professional activities.

Ing. Dirk Van Merode MSc.
Coordinator of the DESIRE project,
TMMA, Belgium

ВСТУП

Сьогодні технології Інтернету речей (Internet of Things - IoT) значно розширюють можливості збору, аналізу та поширення даних, які людство може перетворити в інформацію та знання. Інтернет речей відкриває нові перспективи і дає більше можливостей для підвищенні ефективності економіки за рахунок автоматизації процесів в різних сферах діяльності [1]. На початку 2016 року основними сегментами для застосування IoT були обробна промисловість, енергетика і транспорт [2]. Зростає також вплив IoT на бізнес-діяльність компаній. Розумні, пов'язані продукти і дані, які вони генерують, суттєво перетворюють традиційні бізнес-функції [3].

Звичайно, є ще багато питань, які повинні бути вирішені для розвитку та впровадження IoT технологій: потреба у великій кількості нових унікальних IP-адрес, автономне живлення датчиків, сертифікація IoT-пристроїв, безпека, захист особистої інформації і т.д. [1]. Але вже сьогодні, завдяки IoT технологіям, світ починає взаємодіяти з фізичними та віртуальними "речами" і пристроями іншим способом. Інтернет став не лише середовищем спілкування і обміну інформацією між людьми, але також інструментом і технологією взаємодії між клієнтами, "речами" і пристроями. Саме тому, промисловість хоче ефективно проектувати, створювати і розгортати сучасні інтелектуальні зв’язані продукти і потребує відповідних фахівців з широким діапазоном знань і навичок бізнес-аналітики, апаратної інженерії, інформаційної безпеки та IoT. З цієї причини, компаніям необхідно більше і більше відповідних фахівців для розробки і впровадження нових технологій для ефективної взаємодії замовників і «речей». Багато компаній вже гостро відчувають нестачу таких фахівців.

Саме тому, інтеграція викладання IoT технологій в навчальний план підготовки IT-фахівців є актуальним завданням, оскільки дає реальні можливості для підвищення конкурентоспроможності студентів в умовах мінливого ринку праці.

Задача навчання IoT технологіям спрямована на формування знань студентів в області сучасного програмного і апаратного забезпечення IoT, а також практичних навичок застосування існуючих та розробки нових платформ і пристроїв.

Метою даної розробки є реалізація практично-орієнтованих підходів і методів у підготовці майбутніх IT-професіоналів на основі програмно-апаратної платформи для навчання технологіям Інтернету речей.

Запропонована програмно-апаратна платформа призначена для навчання студентів та проведення наукових досліджень в галузі IoT-технологій. Вона реалізована у вигляді REIoT комплексу та об'єднує кілька підсистем, що дозволяють створити справжній Інтернет речей для Розумного будинку.

REIoT - інтегрований комплекс, що об’єднує віддалену лабораторію RELDES (Remote Laboratory for Design of Embedded Systems) з лабораторним обладнанням Smart House& IoT (Smart House & Internet of Things). Обидві складові комплексу розроблені на кафедрі програмних засобів Запорізького національного технічного університету. Комплекс базується на платформах Arduino, Raspberry Pi та OpenHAB. OpenHAB REST API використовується для інтеграції віддаленої лабораторії RELDES з Smart House & IoT. Це дозволяє отримати віддалений доступ до лабораторних експериментів та їх станів, а також забезпечує поновлення статусу або відправку команд для експериментів.

Комплекс використовується для різних навчальних завдань в декількох режимах. У першому режимі комплекс забезпечує можливості проведення віддалених експериментів по кожній з підсистем окремо. Описи експериментів і результатів вимірювань доступні для студентів в цьому випадку. Інший режим дозволяє студентам визначити логіку роботи системи в цілому, здійснювати програмування і моніторинг процесів.

Використання REIoT комплексу в навчальному процесі дає студентам всі переваги дистанційних експериментів, а також можливості практичного вивчення різних платформ, датчиків, виконавчих пристроїв, протоколів та інтерфейсів для реалізації IoT технологій.

Практично-орієнтовані методи навчання, засновані на використанні комплексу REIoT, надають студентам необхідні знання та навички для реалізації своїх власних проектів, а також для успішного застосування IoT-технологій в майбутній професійній діяльності. Реалізація реальних проектів дає студентам цінний

практичний досвід в галузі IoT технологій, мотивацію до наукових досліджень та навички роботи в команді.

Міністерство освіти і науки України, Національна академія педагогічних наук України підтримують ідею розвитку віддалених лабораторій та відзначають такі розробки. У жовтні 2015 року під час Національного виставкового конкурсу «Видатні науково-практичні досягнення в освіті» команда розробників RELDES була нагороджена в номінації «Електронний освітній ресурс».

У березні 2017 року на міжнародній виставці "Сучасні заклади освіти - 2017" команда розробників лабораторії Smart House&IoT була нагороджена Золотою медаллю у номінації «Розробка та впровадження інноваційних проектів, тренінгових технологій, програм та рішень для осучаснення навчального процесу та підвищення рівня знань молоді».


ДИПЛОМ
нагороджується
ЗОЛОТОВО МЕДАЛЛЮ
Запорізький національний
технічний університет
номінація
«Розробка та впровадження інноваційних проектів,
тренінгових технологій, програм і рішень для осучаснення
навчального процесу та підвищення рівня знань молоді»
СВІТ

1. КОНЦЕПЦІЯ ПРОГРАМНО-АПАРАТНОЇ ПЛАТФОРМИ ДЛЯ НАВЧАННЯ ТЕХНОЛОГІЯМ ІНТЕРНЕТУ РЕЧЕЙ

1.1 Що таке Інтернет речей

Інтернет речей можна трактувати як мережу фізичних об'єктів, що містять вбудовану технологію, яка дозволяє цим об'єктам вимірювати параметри власного стану або стану навколишнього середовища, використовувати та передавати цю інформацію.

Перше завдання IoT - це віддалений моніторинг і управління набором взаємозалежних мережевих пристроїв, кожен з яких може взаємодіяти з об'єктами інфраструктури та фізичного середовища. Наприклад, датчик температури і вологості контролює мережу приладів, що керують системою клімату Розумного будинку (вікна, жалюзі, кондиціонери та ін.). Більш екзотичний приклад – датчик на руці власника Розумного будинку подає сигнал про психофізичний стан господаря всім розумним пристроям, що знаходяться в мережі; кожен з них реагує певним чином, в результаті чого змінюється освітленість, фонова музика, кондиціонування. Тут основна функція не аналітична, а саме керуюча. Друге завдання – це використання даних, одержуваних з кінцевих вузлів (смартпристроїв з можливістю підключення і зондування), для інтелектуального аналізу з метою виявлення тенденцій і взаємозв'язків, які можуть генерувати корисну інформацію для забезпечення додаткової вигоди, наприклад, заощадження ресурсів.

Особливу роль в Інтернеті речей відіграють засоби вимірювання, що забезпечують перетворення відомостей про зовнішнє середовище в машинозчитувані дані і, таким чином, наповнюють обчислювальне середовище значущою інформацією. Використовується широкий клас засобів вимірювання, від елементарних датчиків (наприклад, температури, тиску, освітленості), приладів обліку споживання (таких, як інтелектуальні лічильники) до складних інтегрованих вимірювальних систем. В рамках концепції Інтернету речей принциповим є об'єднання засобів вимірювання в мережі (такі, як бездротові сенсорні мережі, вимірювальні комплекси), за рахунок чого можлива побудова систем міжмашинної взаємодії.

Як особлива практична проблема впровадження Інтернету речей зазначається необхідність забезпечення максимальної автономності засобів вимірювання, насамперед, проблема енергопостачання датчиків. Знаходження ефективних рішень, що забезпечують автономне живлення сенсорів (використання фотоелементів, перетворення енергії вібрації, повітряних потоків, використання бездротової передачі електрики), дозволяє масштабувати сенсорні мережі без підвищення витрат на обслуговування (у вигляді зміни батарейок або підзарядки акумуляторів датчиків) [4-5].

Навчання IoT технологіям - непросте завдання. З одного боку, є багато веб-сайтів, вебінарів, документації і матеріалів по цій темі [6-14]. З іншого боку, навіть інтерпретація терміну «IoT» в різних роботах інколи значно відрізняється. Крім того, описана величезна кількість різних пристроїв і платформ для створення IoT-систем, але зазвичай досить проблематично для студентів і викладачів розібратися та знайти необхідну інформацію в цьому різноманітті. Таким чином, задача створення сучасної, зручної у використанні програмно-апаратної платформи для навчання IoT-технологіям та впровадження її в процес підготовки IT-фахівців є актуальною.

1.2 Інтернет речей та вбудовані системи

Так само, як інші фахівці в цій галузі [15], ми прийняли таке визначення IoT в якості основи: «Інтернет речей - взаємозв'язок однозначно ідентифікованих вбудованих обчислювальних пристроїв в рамках існуючої інфраструктури Інтернету». У цьому випадку можна виділити кілька практично-орієнтованих освітніх завдань: вивчення програмного і апаратного забезпечення вбудованих систем (ВС), аналіз існуючих підходів до проектування ВС, вивчення принципів взаємодії ВС та їх підключення до Інтернету [16, 17].

Як відомо, ВС можуть бути використані в поєднанні з датчиками та виконавчими пристроями для збору інформації та перетворення зібраної або отриманої інформації в дії. Також, ВС можуть використовувати ряд технологій для з'єднання з іншими пристроями або з мережею Інтернет (WiFi, Bluetooth, RFID, Ethernet, GSM, CDMA і т.д.) [18].

Найчастіше, поняття IoT нерозривно пов'язують з чимось Smart, або «розумним»: Розумний будинок, Розумний транспорт, Розумне місто, Розумний бізнес і т.д. [19-25].

Розумний будинок - єдина система управління в будинку або квартирі, що включає в себе датчики, керуючі елементи і виконавчі пристрої. Керуючі елементи приймають сигнали з датчиків і контролюють роботу виконавчих пристроїв, діючи відповідно до заданих алгоритмів та об'єднуючи такі підсистеми:

– генерації електроенергії (сонячна або вітрогенерація);

– контроля енергоспоживання, обмеження пікових навантажень і розподілу навантажень по фазах мережі живлення;

– управління джерелами резервного електроживлення (акумуляторними і дизель-генераторами);

– опалення будинку та постачання гарячої води (за допомогою котлів на різних видах палива або електроенергії, теплових насосів, геліосистеми);

– вентиляції, кондиціонування, рекуперації, пасивного охолодження;

– охоронної та пожежної сигналізації;

– контролю доступу;

– контролю аварійних ситуацій (витоку води, газу, аварії в електромережі);

– відеоспостереження (локальне та віддалене);

– управління внутрішнім і вуличним освітленням;

– мультируму та розподілу відео й аудіо;

– управління обігрівом зливової каналізації, сходів і доріжок;

– управління каналізаційною насосною станцією та системою автополиву зелених територій;

– управління воротами і шлагбаумами;

– управління шторами, ролетами і жалюзі;

– віддаленого моніторингу і управління всіма підсистемами будинку через Інтернет.

Технології створення Розумного будинку цікаві і корисні, оскільки вони дозволяють зробити наше життя більш комфортним, безпечним і забезпечити економію ресурсів. Саме тому, програмно-апаратну платформу для вивчення і дослідження IoT-технологій було реалізовано як REIoT комплекс, що включає в себе дві інтегровані

частини – віддалену лабораторію RELDES та віддалену лабораторію Smart House & IoT [26-30]. Інтерфейс лабораторії RELDES представлено на рис. 1.1.

Рисунок 1.1 – Інтерфейс віддаленої лабораторії RELDES

Інтеграція віддаленої лабораторії в навчальний процес розширює можливості дистанційного навчання і надає всі переваги віддаленого експерименту[31-39].

Концепція дизайну стенду Smart House & IoT лабораторії пройшла кілька етапів розвитку (рис. 1.2). Зрештою, ми використали два найбільш популярні вбудовані платформи для IoT-розумних пристроїв - Arduino та Raspberry Pi [15,40,41], а також платформу OpenHAB (Open Home Automation Bus). OpenHAB - це програмне забезпечення для інтеграції різних систем домашньої автоматизації і технологій в одне загальне рішення, яке передбачає всеосяжні правила автоматизації і пропонує єдині інтерфейси [41-44].

Архітектура REIoT-комплексу показана на рис. 1.3.

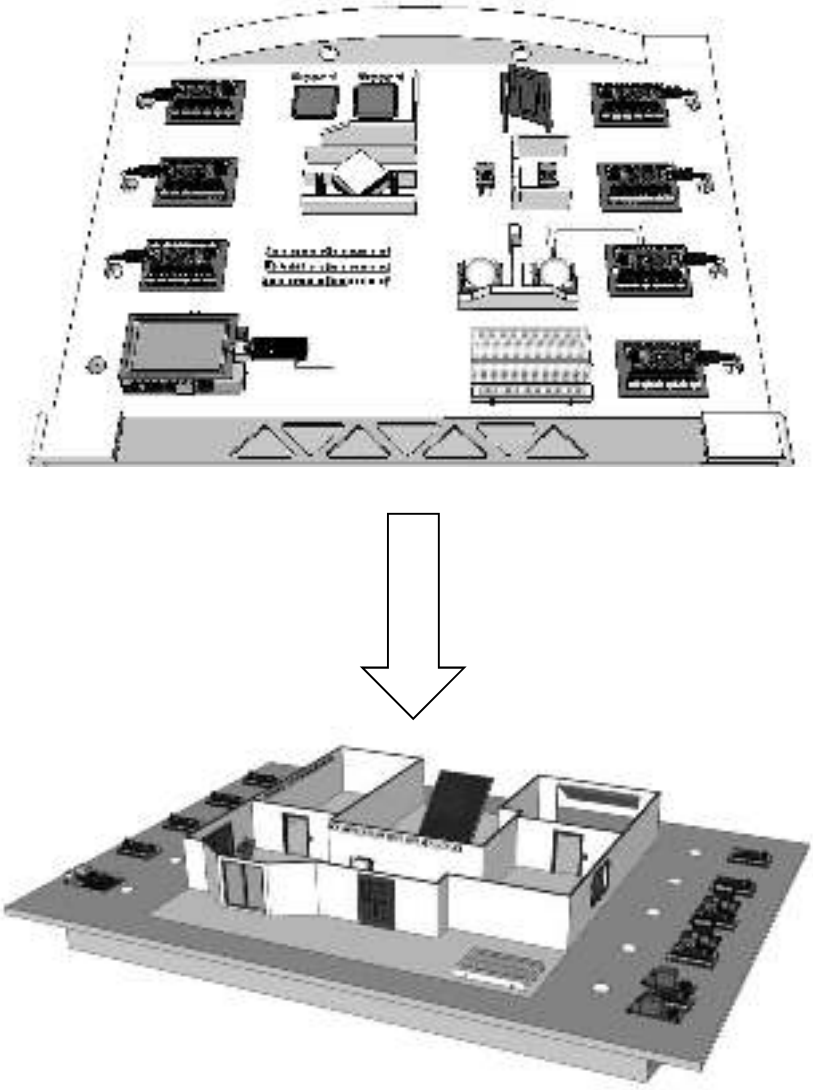

Рисунок 1.2 – Еволюція концепцій дизайну стенду лабораторії

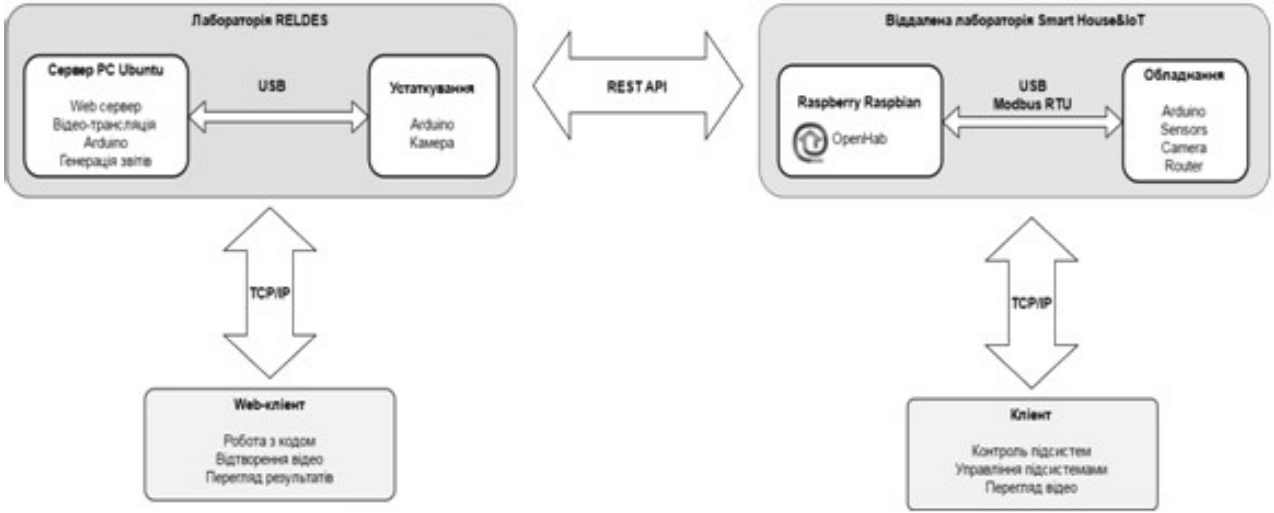

Рисунок 1.3 - Архітектура REIoT комплексу

1.3 Особливості реалізаціїї лабораторії Smart House & IoT

Набір лабораторії Smart House & IoT заснований на Arduino NANO V3 платах (рис. 1.4) та включає в себе наступні експерименти: Сонячна станція, Управління освітленістю, Клімат-контроль, Контроль доступу, Контроль безпеки, Контроль зони, Контроль присутності, Рекуперація та вентиляція, Контроль освітлення.

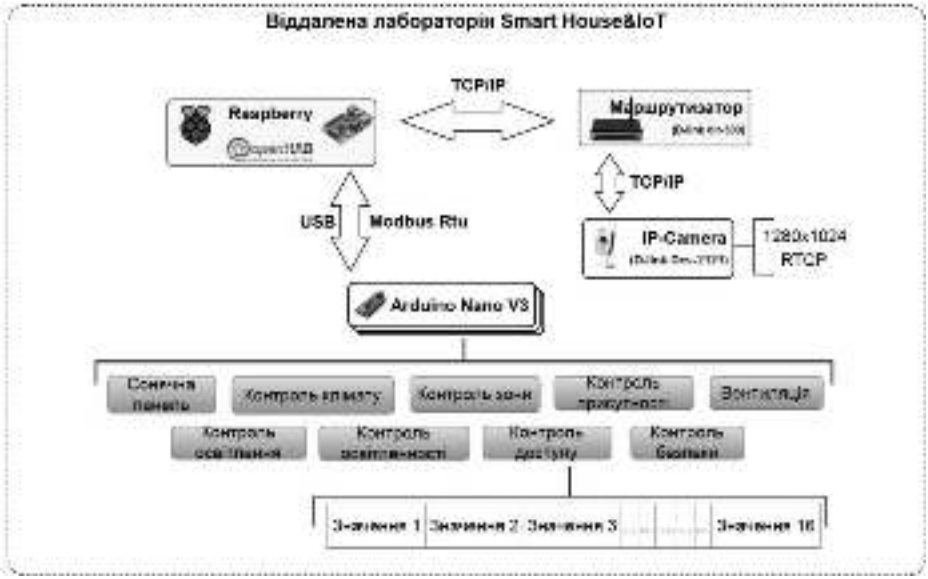

Рисунок 1.4 – Структура та складові Smart House & IoT лабораторії

Міні комп'ютер Raspberry Pi виконує роль сервера лабораторії зі встановленою платформою OpenHAB. Додаткові бібліотеки Modbus TCP Binding використовуються для його зв'язку з платами Arduino за USB інтерфейсом і протоколом Modbus RTU. Послідовна лінія RS-232 використовується для обміну даними між електронними пристроями.

Кожна плата Arduino містить програму, яка обробляє вхідні та вихідні дані. Arduino використовує відкриту бібліотеку Modbus Master-Slave. За допомогою цієї бібліотеки регістри знакового або беззнакового типу, які доступні для запису і зчитування, були створені

в кожній платі (підсистемі). Регістри містять 8 або 16 елементів, довжиною 16 бітів кожен. Таким чином, структура для обміну даними була створена. Кожна плата Arduino працює як Master.

Платформа OpenHAB зчитує або додає дані в регістри, коли опитує пристрої. Кожен елемент регістра корелюється з індивідуальними параметрами датчиків або виконавчих механізмів.

Лабораторія включає в себе IP-камеру D-Link DCS-2121, яка передає потокове відео для користувачів, щоб переглянути поточний стан експериментів. Ця IP-камера являє собою повну систему з вбудованим процесором і Web-сервером, який передає високоякісне відео з роздільною здатністю 1280 x 1024 пікселів і швидкістю 10 кадрів в секунду. IP-камера і комп'ютер з'єднані за допомогою кабелю Ethernet і взаємодіють з використанням протоколу TCP/IP. Маршрутизатор D-Link DIR-300 дозволяє підключатися до лабораторії через Wi-Fi, а також додавати пристрої, що використовують мережеві кабелі. Зовнішній вигляд лабораторного обладнання наведено на рис. 1.5.

Для інтеграції двох складових частин REIoT, а також для адміністрування Smart House & IoT лабораторії було використано OpenHAB REST API [45]. Щоб отримати доступ до експериментів лабораторії Smart House&IoT, адміністративна система лабораторії RELDES відправляє HTTP GET запит до API OpenHAB REST API і отримує результати в форматі JSON.

Отримавши список доступних в лабораторії Smart House & IoT експериментів, RELDES включає їх в загальний список експериментів (рис.1.6), після чого черга, статистичні дані та інші функції віддаленої лабораторії RELDES стають доступні для них.

Згодом, щоб провести експеримент, RELDES пересилає REST-методи в лабораторію Smart House & IoT, наприклад, HTTP PUT і HTTP GET запити використовуються для зміни типу та рівня освітленості і контролю результатів (рис.1.7).

Для того, щоб розпочати відео-трансляцію, ми використовували утиліту ffmpeg [46]. FFmpeg являє собою набір безкоштовних бібліотек з відкритим вихідним кодом, які дозволяють записувати, конвертувати і передавати цифрове аудіо та відео в різних форматах. Бібліотека ffmpeg захоплює відео з нашої камери з роздільною здатністю 1280 x 1024, кодує його в формат MPEG з 10 кадрів в секунду і бітрейт 800 Кбіт/с і після цього, використовує HTTP для

відправки на локальний сервер, який надсилає цей відеострім для кінцевого користувача.

Для того, щоб розділити відео на блоки (вирізати і вибрати частину відео для кожного експеримента), використовується фільтр "crop". В результаті, ми можемо отримати певні відео-фрагменти для кожного експеримента або для групи експериментів.

Рисунок 1.5 – Зовнішній вигляд стенду лабораторії Smart House&IoT

Рисунок 1.6 – Інтерфейс лабораторії RELDES з доданим списком експериментів лабораторії Smart House & IoT

1.4 Функціональні можливості лабораторії Smart House & IoT

Вивчення та практична реалізація віддалених експериментів лабораторії Smart House & IoT надає безліч можливостей для дослідження та вивчення технологій Розумного будинку.

Експеримент «Сонячна станція» призначений для вивчення основ сонячної енергії і принципів роботи сонячної батареї. Основними компонентами є: сонячна панель (6В, 300мА), літій-іонний акумулятор, зарядний пристрій для літій-іонних батарей, підвищуючий перетворювач напруги.

Експеримент «Клімат-контроль» призначений для вивчення основ кліматичного контролю з використанням цифрових датчиків температури і вологості DHT-11, а також аналогового датчику якості повітря MQ135.

Рисунок 1.7 – Веб-сторінка експерименту Управління освітленістю

Експеримент «Контроль зони», що заснований на модулі лазера (2-5мВт) та модулі фоторезистора, дозволяє контролювати стан периметра і реагує на разі його порушення.

Експеримент «Контроль присутності» призначений для комплексного вивчення принципів роботи систем освітлення та систем, які контролюють присутність людини в приміщенні. Експеримент заснований на використанні сервомотору, що рухає об’єкти, піроелектричного датчика і лінзи Френеля (піроелектричний датчик руху).

Експеримент «Рекуперація та вентиляція» дозволяє вивчати основи рекуперaційних установок та регулювання потоків повітря. Експеримент побудований з використанням драйвера електричних навантажень L298E, що використовує широтно-імпульсну модуляцію, який керує швидкістю вентиляції. Для підігріву радіатору, через який проходить повітря, використовується елемент Пельтьє (12В, 60Вт).

Визначення температури та вологості повітря у системі вентиляції відбувається за допомогою високоточного цифрового датчика температури і вологості DHT22.

Експеримент «Контроль освітлення» дозволяє виконувати контроль та регулювання рівня освітлення різних зон з використанням даних з датчиків-фоторезисторів у різних зонах.

Експеримент «Управління освітленістю» призначений для вивчення основ навантаження об'єктів дистанційного керування за допомогою широтно-імпульсної модуляції в мультиканалах. Основними компонентами є: світлодіодні стрічки, світлодіодні RGB стрічки і драйвер навантаження L298E.

Експеримент «Контроль доступу» використовує RFID-зчитувач карт і брелок RC522. Експеримент призначений для вивчення принципів побудови систем контролю доступу і систем авторизації.

Експеримент «Контроль безпеки» дозволяє вивчити принципи побудови систем безпеки, які реагують на виняткову ситуацію, наприклад, рух в контрольованій зоні. Підсистема може перебувати в стані контролю зон або стані датчиків індикації. Експеримент заснований на піроелектричних датчиках руху з використанням світлодіодного індикатора сигналу тривоги.

Кілька експериментів можуть бути виконані одночасно. В цьому випадку вивчаються принципи взаємодії між підсистемами, визначається логіка процесів, створюються ефекти, оцінюється реакція елементів і аналізуються отримані результати.

Керування віддаленою лабораторією реалізоване з використанням REST API взаємодії. OpenHAB має можливість стабільного керування за допомогою API команд заздалегідь відомим підключеним обладнанням. Кожне устаткування у експериментах має свій поточний статус, та розроблені команди для керування. За допомогою веб-додатку до лабораторії відправляються керуючі або статусні GET/POST запити, що оброблюються системою OpenHAB та приводять до дії необхідне устаткування або повертають для відображення інформацію про статуси сенсорів. Устаткування підвищеної безпеки має апаратні та автоматичні запобіжні заходи.

Можливо використовувати стандартні і створювати власні Actions в Scripts and Rules для виконання OpenHAB конкретних операцій. Наприклад, такі Actions, як Telegram, my.openHAB та інші можуть бути використані для повідомлення про події або зворотнього

зв'язку з Smart House&IoT лабораторією. Підключення до Telegram дозволяє відправляти повідомлення клієнтам Telegram від бот-клієнта (наприклад, відправка повідомлень користувачеві про включення або виключення вентиляції). З використанням my.openHAB користувачі можуть підключитися до OpenHAB REIoT комплексу з будь-якого пристрою з будь-якої точки планети (за умови підключення до Інтернет), щоб забезпечити доступ до інших користувачів, а також зберегти всі заходи і події в хмарі my.openHAB. У цьому випадку лабораторія Smart House & IoT виступає як автономна складова REIoT комплексу.

Адміністрування подій, виконаних OpenHAB може бути реалізовано за допомогою Mail-Control binding. Ця прив'язка забезпечує можливість прийому команд, надісланих електронною поштою в форматі JSON. Наступні типи команд можуть бути відправлені: decimal, HSB, increase – decrease, on – off, open – closed, percent, stop – move, string, up – down.

Крім того, можлива інтеграція з Google календарем для REIoT комплексу. Користувачі можуть створювати події та керувати системою відповідно до графіка (вкл/викл. освітлення, кондиціонування, відкривання/закривання дверей протягом заданого часу і т.д.).

Таким чином, набуваються знання і практичні навички з IoT-технологій шляхом виконання дистанційно керованих експериментів, вивчення описів цих експериментів, реалізації різних сценаріїв керування системою та програмування. Одним з практичних завдань є розробка настільних або мобільних додатків для підключення та управління OpenHAB.

1.5 Платформа OpenHAB

1.5.1 Загальна інформація про платформу

OpenHAB (Open Home Automation Bus) - це відкрита платформа для організації роботи Розумного будинку. Код проекту написаний на мові Java, оформлений у вигляді модульної системи OSGi (використовується Eclipse Equinox) і розповсюджується під ліцензією GPLv3. Для організації роботи web-інтерфейсу задіяний Jetty [41-43].

OpenHAB надає засоби для організації шини, що забезпечує узгоджену роботу різних систем, обладнання та інтерфейсів домашньої автоматизації, дозволяючи передавати через дану шину команди і отримувати інформацію про стан. При цьому openHAB не залежить від протоколів і устаткування, надаючи окремий рівень абстракції, що дозволяє взаємодіяти з різними типами пристроїв і програмного забезпечення. Для визначення керуючої логіки пропонується використовувати скрипти, написані на спеціальній предметно-орієнтованій мові програмування, розробленій за допомогою Eclipse Xtext.

Шина подій. Шина подій є основним сервісом OpenHAB. Всі модулі, які не вимагають відстеження стану, використовують цю шину для обміну інформацією про події з іншими модулями. Існує два основних типи подій:

- команди, які ініціюють будь-яку дію або зміну стану певного елемента або пристрою;
- оновлення статусу, які повідомляють про зміну стану певного елемента або пристрою (часто це відбувається у відповідь на будь-яку команду).

Усі біндінги (прив'язки) протоколів, що забезпечують зв'язок з реальними пристроями, повинні спілкуватися між собою через шину подій. Це гарантує існування зв'язку між модулями, що забезпечує динамічний характер OpenHAB. В якості фундаменту в OpenHAB використовується сервіс OSGi EventAdmin. Це легка pub/sub реалізація, яка ідеально відповідає всім вимогам.

Репозиторій. За допомогою сервісів, які не фіксують поточний стан, неможливо охопити всі функції. Тому, для OpenHAB також пропонується репозиторій елементів, що підключається до шини подій і відслідковує статус всіх елементів. Репозиторій елементів може використовуватися кожен раз, коли виникає необхідність отримати доступ до даних про поточний стан. Наприклад, в інтерфейсі в момент доступу користувача повинен відображатися поточний стан елементів. Також механізм виконання процесів автоматизації повинен завжди перебувати в курсі поточного стану. Репозиторій елементів дозволяє уникнути необхідності зберігати потрібні дані про стан безпосередньо всередині кожного модуля. Він також гарантує, що стани залишатимуться синхронізованими для всіх модулів, а також забезпечує можливість запису статусів в файловій системі або базі

даних, завдяки чому вони зберігаються навіть після перезавантаження системи.

OpenHAB має два різних вбудованих канали зв'язку: асинхронна шина подій та репозиторій з відстеженням станів, з якого можна зробити запит даних. Діаграма на рис.1.8 показує, як використовуються ці канали зв'язку.

Управління OpenHAB може здійснюватися через web-інтерфейс (рис.1.9), в якому допускається підключення додаткових віджетів (наприклад, віджет для вмикання/вимикання світла в кожній кімнаті або віджет для перегляду відео з камер спостереження). Додатково підготовлено кілька альтернативних систем управління, таких як клієнтські програми для платформ Android і iOS, що дозволяють управляти системою з мобільного телефону або планшету, а також оперативно отримувати повідомлення про події.

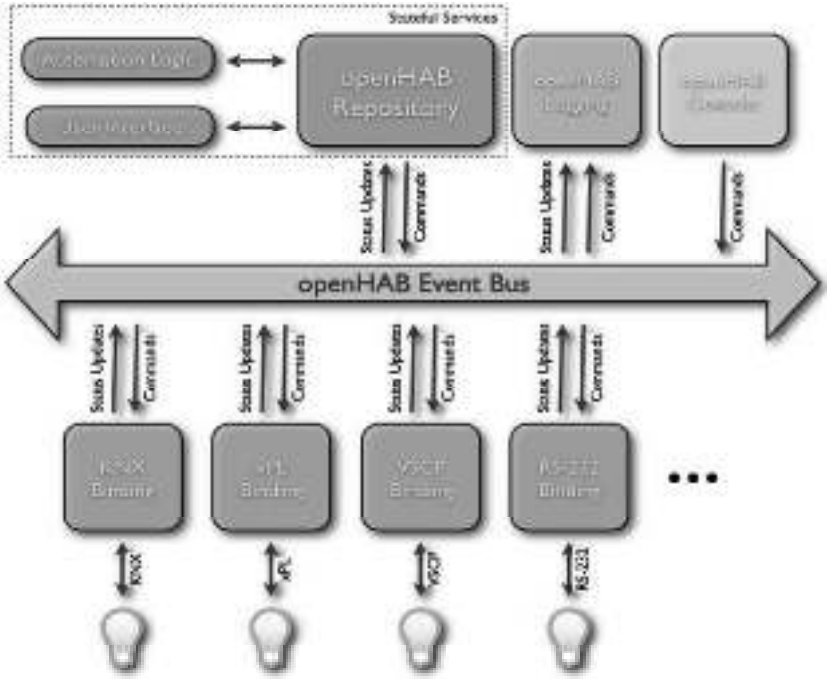

Рисунок 1.8 – Структура OpenHAB

Рисунок 1.9 – Інтерфейс OpenHAB

1.5.2 Сценарії OpenHAB (Scripts)

В openHAB використовується дуже потужна мова виразів, за допомогою якої створюються визначення сценаріїв. Сценарій або скрипт - це блок коду, який визначається користувачем і може викликатися і використовуватися в різних місцях.

Місцезнаходження файлу. Сценарії розміщуються в папці $ {openhab.home} / configurations / scripts. У комплекті з робочим середовищем йде демонстраційний файл з назвою demo.script. Ім'я файлу (без його розширення) є назвою сценарію, використовуваною для посилань. Сценарії також можна знайти всередині файлу правил, розміщеного в папці $ {openhab.home} / configurations / rules. Вони використовуються для визначення блоку виконання правила (EXECUTION_BLOCK). Кожне правило складається з двох частин: в одній містяться тригери дій, в інший - сценарії для їх виконання.

Кожне правило починає виконуватися лише при спрацьовуванні одного з тригерів, описаного між операторами when і then.

Синтаксис. Мова виразів, яка використовується для скриптів аналогічна тій, що використовується в Xtend. Синтаксис цієї мови дуже схожий на Java, але має багато корисних функцій, що дозволяють писати лаконічний код. Він особливо ефективний для обробки колекцій. Ідеальним варіантом для openHAB з технічної точки зору його робить той факт, що при цьому зникає необхідність в компіляції скриптів, оскільки вони можуть бути інтерпретовані прямо під час виконання.

Змінні і функції. Щоб за допомогою скриптів можна було зробити щось корисне, openHAB забезпечує доступ:

- до всіх певних елементів (завдяки чому ви легко можете отримати до них доступ по імені);
- до всіх пронумерованих статусів / команд, таких як ON, OFF, DOWN,INCREASE;
- до всіх стандартних дій для виконання різних операцій.

Комбінуючи ці функції, ви можете легко написати наприклад такий код:

```
if(Temperature.state < 20) {
sendCommand(Heating, ON)
}
```

Оператор повернення. Якщо ви хочете зупинити виконання правила або скрипта, необхідно визначити значення, що повертається. Якщо не вказати його явно, це може привести до того, що правило поведе себе непередбачувано, наприклад:

```
if(Temperature.state >= 20) return false
sendCommand(Heating, ON)
```

Виклик сценарію. Сценарій ідентифікується за його назвою. Якщо файл називається demo.script, ім'я скрипта буде просто demo. Кожен скрипт має значення, що повертається, яке являє собою результат останнього виразу, який міститься в ньому (може дорівнюватись null).

Сценарії можуть викликатися з різних місць:

- з правил через дію callScript("<scriptname>");
- з записів в календарі Google — треба просто вставити вираз > allScript("<scriptname>") в текст запису;
- з XMPP-консолі, для чого треба використати > allScript("<scriptname>").

Зверніть увагу, що після знака ">" можна вставити будь-який вираз, виклик скрипта - це лише один з варіантів. Отже, запит поточної температури в XMPP-консолі міг би виглядати так:

> Temperature.state [2]

Правила (Rules). Правила використовуються для автоматизації процесів. Кожне правило може бути спрацьоване шляхом запуску скрипта, який виконує будь-які види завдань, наприклад, вмикати освітлення, виставити таймери і так далі.

Розташування файлу. Правила розміщуються в папці $ {openhab.home} / configurations / rules. Середовище виконання вже поставляється з демо - файлом з ім'ям demo.rules.

Файл правил може містити кілька правил. Всі правила в файлі мають загальний контекст виконання, тобто вони можуть отримати доступ і обмінюватися змінними один одного. Тому, має сенс мати різні файли правил для різних сценаріїв використання або категорій.

Синтаксис. Файл правила є текстовим файлом з наступною структурою:

[Imports]
[Variable Declarations]
[Rules]

У розділі *Імпорт* містяться оператори імпорту, як і в Java. Подібно Java, вони виконують імпортовані типи доступу без необхідності використовувати повне ім'я для них. Приклад:

import org.openhab.core.library.types.*

Розділ *Оголошення змінних* можна використовувати для оголошення змінних, які повинні бути доступні для всіх правил в цьому файлі. Ви можете оголошувати змінні з / без початкових значень, що змінюються або тільки для читання. Приклад:

// зиінна з початковим значенням
var counter = 0
// тільки для читання
val msg = "This is a message"
// неініційована змінна
var Number x

У розділі *Правила* міститься список правил. Кожне правило має наступний синтаксис:

rule "rule name"
When

<TRIGGER CONDITION1> or
<TRIGGER_CONDITION2> or
<TRIGGER_CONDITION3>
...
Then
<EXECUTION_BLOCK>
end

Правило може мати будь-яку кількість умов. EXECUTION_BLOCK містить код, який повинен бути виконаний, коли умови запустяться.

Тригери. Кожне правило починає виконуватися лише при спрацьовуванні одного з тригерів, описаного між when і then. Тригер - це деяка подія. Є різні категорії правил тригерів:

- Item (-Event) - based triggers: вони реагують на події на шині подій openHAB, тобто команди і поновлення статусу для елементів;
- Time-based triggers: вони реагують в особливі моменти, наприклад, опівночі, кожну годину і т.д.;
- System-based triggers: вони реагують на певні системні статуси.

Google Calendar. Якщо потрібно керувати подіями в Google Calendar, які будуть виконуватися openHAB, потрібно налаштувати необхідні дані щоб отримати URL.

Конфігурація Календар подій. Назва події може бути якою завгодно, і опис події матиме команди для виконання.

Формат опису Календаря подій простий і виглядає наступним чином:

start {
send|update <item> <state>
}
end {
send|update <item> <state>
}
або
send|update <item> <state>

Команди в start секції будуть виконуватися в момент початку події, end секції на кінець події часу. Якщо цих секції немає, команди будуть виконуватися в момент початку події.

Як отримати URL календаря. OpenHAB повинен бути налаштований з календарем Google (GCal). URL повинен бути налаштований в openhab.cfg. Наступні випадки підтримуються:

- URL, ім'я користувача, пароль (адреса календаря);
- URL (приватна адреса).

Варіант 1. Отримання URL Google календаря використовуючи ім'я користувача/пароль:

- заповнити gcal: URL Gcal: ім'я користувача, Gcal: пароль в openhab.cfg;
- увійти в систему https://www.google.com/calendar/;
- натиснути "Налаштування" (в розділі "Мої календарі") -> <ваш календар>;
- знайти URL календаря в рубриці "Адреса календаря" -> "XML". URL буде виглядати так: https://www.google.com/calendar/feeds/user@gmail.com/public/basic;
- скопіювати даний URL і замінити "basic" на "full".

Остаточний URL повинен виглядати так:

https://www.google.com/calendar/feeds/user@gmail.com/private/full

Google додав нову функцію безпеки, яка повинна бути відключена для роботи цього методу. Перейти https://www.google.com/settings/security/lesssecureapps і дозволити доступ до менш захищених додатків.

Варіант 2. Отримання URL Google календаря без пароля (тільки для читання):

- тільки Gcal: URL повинен бути використаний в openhab.cfg
- увійти в систему https://www.google.com/calendar/
- натиснути "Налаштування" (в розділі "Мої календарі") -> <ваш календар>
- знайти URL календаря в рубриці "Закрита адреса" -> "XML". URL буде виглядати так:
- https://www.google.com/calendar/feeds/user@gmail.com/private-da2412ef24124f151214deedbeef1234/basic
- скопіювати дані URL і замінити "basic" на "full".

Остаточний URL буде виглядати так:

https://www.google.com/calendar/feeds/user@gmail.com/private-da2412ef24124f151214deedbeef1234/full

Імітація присутності. GCal може бути використаний для реалізації простої, але ефективної функції імітації присутності. За замовчуванням кожен запис розміщується зі зміщенням 14 днів (якщо ви хочете змінити зміщення, будь ласка, змініть параметр gcal-persistence: offset в openhab.cfg). Кожен запис календаря виглядає наступним чином:

Назва: [PresenceSimulation] <itemname>

Зміст: > if (PresenceSimulation.state == ON) sendCommand(<itemname>,<value>)

Для того, щоб використовувати імітацію присутності потрібно пройти через ці кроки налаштування конфігурації:

- переконатися, що вибран правильний реліз openHAB;
- налаштувати пакет Gcal шляхом додавання відповідної конфігурації в openhab.cfg. Всі записи починаються з gcal-persistence. Потрібно додати принаймні, ім'я користувача, пароль і URL;
- переконатися, що файл містить елементи, які належать до групи PresenceSimulationGroup.
- переконатися, що файл містить елементи під назвою PresenceSimulation, який згадується сценаріями, виконуваними в певний момент часу.

Необхідно також налаштувати GCAL зв'язування, щоб мати можливість читати записи з календаря і діяти у ньому.

Для активації імітації присутності просто встановити PresenceSimulation ON.

Дії (Actions). Доступний **с**писок основних дій:

1) Аудіо дії:

- setMasterVolume (float volume): встановлює гучність (обсяг в діапазоні 0-1);
- increaseMasterVolume (float percent): збільшує гучність на x відсотків;
- decreaseMasterVolume (float percent): зменшує гучність на x відсотків;
- float getMasterVolume (): повертає обсяг з діапазоном 0-1;
- playSound (String filename): повне або часткове відтворення файлу (повинен бути mp3 або WAV і перебувати в $ {openhab.home} / sounds);
- playStream (String url): відтворення аудіо потоку з даного URL;

- say (String text): промовляє даний текст через перетворення тексту в мову;
- say (String text, String voice): перетворення тексту в мову з заданим голосом.

2) HTTP дії:

- sendHttpGetRequest (String url): відправка запиту на GET-HTTP;
- sendHttpPutRequest (String url): відправка запиту PUT-HTTP;
- sendHttpPutRequest (String url, String contentType, String content): відправка запиту PUT-HTTP, з параметрами;
- sendHttpPostRequest (String url): відправка запиту POST-HTTP;
- sendHttpPostRequest (String url, String contentType, String content): відправка запиту POST-HTTP, з параметрами;
- sendHttpDeleteRequest (String url): зробити запит DELETE-HTTP.

3) Таймери.

4) Інші дії:

- callScript (String scriptName): викликає скрипт, який повинен знаходитися в папці конфігурації / скриптів;
- executeCommandLine (String commandLine): виконує команду в командному рядку;
- executeCommandLine (String commandLine, int timeout): виконує команду в командному рядку з тайм – ауту;
- transform (String type, String function, String value): застосовує перетворення даного типу з деякою функцією до значення. Повертає перетворене значення, або початкове значення, якщо перетворення не вдалося.

5) Astro дії. З Astro-Action можна розрахувати час сходу і заходу сонця.

6) Logging дії.

7) Twitter дії

Підключення до Twitter через цю дію:

- sendTweet (String message): посилає твіт через Twitter;
- sendDirectMessage (String recipient, String message): посилає пряме повідомлення через Twitter.

8) Погодні дії:

- getHumidex (double temperature, int hygro): обчислює індекс Humidex заданої температури в градусах Цельсія і гідрометрія (відносний відсоток). Повертає Humidex значення індексу;
- getBeaufortIndex (double speed): підраховує шкали Бофорта для заданої швидкості вітру в м/с;
- getSeaLevelPressure (double pressure, double temp, double altitude): розраховує тиск на рівні моря, враховуючи абсолютний тиск, температуру в градусах Цельсія і висоту в метрах. Повертає еквівалентний тиск на рівні моря.

9) Pushsafer дії.

Дія Pushsafer дозволяє повідомити IOS, Android і Windows.

1.5.3 Встановлення та налаштування OpenHAB

Для встановлення OpenHAB потрібно перейти на офіційну сторінку розробників: http://www.openhab.org/getting-started/. Після чого, завантажити основне ядро OpenHAB та пакет додаткових модулів за необхідністю. Слід зауважити, що для коректної роботи у системі повинна бути встановлена Java версії 1.7 або вище. Далі потрібно розпакувати архів у створену директорію. Для Windows розробники рекомендують - C:\openhab, а для Mac/Linux чи інших систем - /opt/openhab.

Щоб встановити додаткові модулі, розпакуйте zip файл у новий каталог (наприклад, all_addons) у каталозі openhab, який ви обрали вище. Знайдіть пакети (кожен пакет знаходиться в окремому файлі .jar), які відповідають технологіям або пристроям, якими ви володієте, а потім скопіюйте цей файл з розширенням .jar у каталог аддонів OpenHAB. Наприклад, якщо у вас є будь-які Z-Wave пристрої, вам потрібно скопіювати org.openhab.binding.zwave-<version>.jar в каталог аддонів. Більшості прив'язок потрібна деяка конфігурація в файлі openhab.cfg. Перегляньте цей файл і розкоментуйте та налаштуйте будь-які розділи, пов'язані із прив'язкою, яку ви обрали. Перевірте Wiki для більш докладної інформації про кожну прив'язку.

Для запуску серверу потрібно запустити на виконання start.bat сценарій (якщо ви працюєте у Windows) або start.sh (якщо ви на Linux або Mac). Через кілька секунд (залежно від швидкості вашого комп'ютера), OpenHAB повинен бути запущений.

Якщо щось не так, як планувалося, ви можете виконати сценарії налагодження замість звичайних (наприклад, start_debug.sh), який дасть вам більше інформації про будь-які помилки, які можуть статися.

Останній крок тестування роботи полягає в наступному. OpenHAB поставляється із вбудованим інтерфейсом. Він працює на всіх браузерах таких, як Chrome, Safari і т.д. У полі вводу URL у вашому браузері вкажіть: http://<openHAB address or hostname>:8080/openhab.app?sitemap=demo, і повинен з'явитися створений інтерфейс сайту. На місці "demo" в URL повинно бути ім'я вашого новоствореного файлу карти сайту.

Окрім основних функцій для підключення та роботи з різним обладнанням, OpenHAB також надає багато додаткових можливостей. Основні додаткові можливості OpenHAB для віддаленого керування і отримання повідомлень , а також способи їх підключення наступні.

Підключення до Telegram дозволить відправляти повідомлення клієнтам Telegram з бот-клієнта. Для підключення ботів до OpenHab потрібні маркери авторизації і ідентифікатори чатів. Згідно Telegram Bot API, щоб отримати токен авторизації і ідентифікатор чату потрібно виконати наступні кроки:

1) Створити бота і отримати токен:

– у клієнтському додатку Telegram почати чат з BotFather;

– написати /newbot для створення бота, ввести ім'я бота і отримати токен авторизації.

2) Створити чат з обраним користувачем і отримати ідентифікатор чату:

– почати чат з новим ботом і відправити йому повідомлення;

– відкрити браузер і відправити запит виду: https://api.telegram.org/bot<token>/getUpdates (де <token> це щойно отриманий токен авторизації);

– переглянути JSON результат і в ньому знайти значення поля result [0] .message.from.id - це і буде ідентифікатор чату.

Щоб підключити цих ботів до OpenHAB, потрібно додати інформацію в файл openhab.cfg, наприклад, для підключення двох ботів потрібно дописати в файл таку інформацію:

telegram: bots = bot1, bot2

telegram: bot1.chatId = 22334455

telegram: bot1.token = xxxxxxxxxxx

telegram: bot2.chatId = 654321
telegram: bot2.token = yyyyyyyyyyy

Тепер можна відправляти повідомлення з OpenHAB, для цього в .rules потрібно додати правило такого виду:

```
rule "Send telegram with Fixed Message"
when
        Item Foo changed
then
        sendTelegram ( "bot1", "item Foo changed")
end
```

Підключення my.openhab. Для можливості підключення до my.openhab в OpenHAB знаходяться два згенерованих файли, один з яких містить UUID користувача, а другий випадковий секретний ключ. Вони розташовані в директорії webapps/static всередині директорії OpenHAB з іменами uuid і secret. Потрібно перейти на сторінку реєстрації: https://my.openhab.org/login і ввести інформацію з файлів uuid і secret у відповідні поля. Після реєстрації для віддаленого доступу можна користуватися посиланням виду:

https://my.openhab.org/openhab.app?sitemap=yoursitemap

Оскільки my.openHAB не передає дані аутентифікації через Інтернет з'єднання з вашого OpenHAB з міркувань безпеки, потрібно відключити локальну аутентифікацію, для можливості відправлення запитів до OpenHAB. Для цього, необхідно в файлі openhab.cfg встановити security: option = EXTERNAL і налаштувати security: netmask = на маску локальної підмережі. Після цього my.openHAB матиме можливість робити запити, в той час як інші зовнішні запити будуть проходити перевірку автентичності користувачів, задану у файлі users.cfg.

Прив'язка MailControl дозволяє приймати команди, відправлені електронною поштою в JSON форматі.

Можуть бути відправлені команди наступних типів: decimal, HSB, increase – decrease, on – off, open – closed, percent, stop – move, string, up – down.

Щоб підключити прив'язку до OpenHAB потрібно додати інформацію в файл openhab.cfg.

Приклад конфігурації:

mailcontrol: username=email.address@some.com
mailcontrol: password = XXXXXXXXXX

mailcontrol: smtphost = smtp.mail.some.com
mailcontrol: smtpport = 465
mailcontrol: smtpauth = true
mailcontrol: smtpstarttls = true
mailcontrol: smtpsocketfactoryport = 995
mailcontrol: pop3host = pop.mail.some.com
mailcontrol: pop3port = 995
mailcontrol: pop3socketfactoryport = 995
mailcontrol: pop3socketfactoryclass = javax.net.ssl.SSLSocketFactory

Після налаштування конфігурації можна відправляти повідомлення-команди.

Приклади повідомлень:

- decimal: { "messageType": "110", "productVersion": "1.0", "itemCommand": { "timeSent": "0", "command": { "commandType": "DECIMAL", "value": " 1.2 "}," item_id ":" Item "}," senderEmail ":" email.address@some.com "};
- on - off: { "messageType": "110", "productVersion": "1.0", "itemCommand": { "timeSent": "0", "command": { "commandType": "ON_OFF", "value" : "ON"}, "item_id": "Item"}, "senderEmail": "email.address@some.com"};
- string: { "messageType": "110", "productVersion": "1.0", "itemCommand": { "timeSent": "0", "command": { "value": "someValue", "commandType": " STRING "}," item_id ":" Item "}," senderEmail ":" email.address@some.com "};
- up - down: { "messageType": "110", "productVersion": "1.0", "itemCommand": { "timeSent": "0", "command": { "value": "UP", "commandType" : "UP_DOWN"}, "item_id": "Item"}, "senderEmail": "email.address@some.com"}.

Підключення прив'язки до Google Calendar. Ця прив'язка дозволяє управляти подіями в Google Calendar, які будуть виконуватися в OpenHAB. OpenHAB повинен бути налаштований з обраним календарем Google, для цього URL повинен бути внесений в ваш openhab.cfg.

Для того, щоб отримати URL календаря google без передачі пароля потрібно виконати наступні кроки:

– у openhab.cfg потрібно використовувати тільки поле gcal: url;

– увійти в https://www.google.com/calendar/;

– натиснути "Settings" (під "My calendars") -> <your calendar>;

– знайти URL календаря в заголовку "Private Address" -> "XML". Отриманий URL буде виглядати приблизно так: https://www.google.com/calendar/feeds/user@gmail.com/private-da2412ef24124f151214deedbeef1234/basic;

– скопіювати даний URL і замінити в ньому "basic" на "full".

Кінцевий URL повинен виглядати так: https://www.google.com/calendar/feeds/user@gmail.com/private-da2412ef24124f151214deedbeef1234/full

1.6 Практичні завдання

1. Підготуйте презентацію на тему «Інтернет речей».
2. Виконайте огляд існуючих віддалених лабораторій в галузі електроніки, електротехніки та вбудованих систем. Підготуйте презентацію.
3. Виконайте віддалені експерименти лабораторії RELDES. Запропонуйте власні експерименти для розширення функціональних можливостей лабораторії.
4. Ознайомтесь з віддаленими експериментами лабораторії Smart House & IoT. Запропонуйте власні експерименти для розширення функціональних можливостей лабораторії.
5. Виконайте налаштування платформи OpenHAB на своєму комп'ютері або на платформі Raspberry Pi.
6. Створіть просту систему домашньої автоматизації: підключить декілька датчиків та виконавчих пристроїв до плати Arduino, виводьте дані з датчиків в OpenHAB, задайте правила та дії для керування вашою системою.

1.7 Перелік посилань

1. Что такое Интернет вещей, IoT [електронний ресурс]. – режим доступу: http://www.tadviser.ru/index.php/
2. Internet of Things (IoT) [електронний ресурс]. – режим доступу: http://www.cisco.com/web/solutions/trends/iot/overview.html
3. Porter M. How Smart, Connected Products Are Transforming Companies. Harvard In Business Review Articles / M. Porter,

J. Heppelmann. - [електронний ресурс]. – режим доступу: http://www.ptc.com/internet-of-things/harvard-business-review/download-article-2#sthash.L5wGKzcN.dpuf
4. Концепція Інтернет речей, аналіз ринку [електронний ресурс]. – режим доступу: http://www.tadviser.ru/
5. IoT Architecture [електронний ресурс]. - режим доступу: https://ru.coursera.org/learn/iot-augmented-reality-technologies/lecture/8ZlnC/iot-architecture.
6. Knowledge is Power [електронний ресурс]. – режим доступу: https://www.thingworx.com/resources/?topic=&type=white-papers
7. Чанг Дж.-М. Новые технологии - Интернет вещей и Дополненная реальность / Дж.-М. Чанг [електронний ресурс]. – режим доступу: https://ru.coursera.org/learn/iot-augmented-reality-technologies/
8. Purpose-built for the Internet of Things [електронний ресурс]. – режим доступу: http://www.thingworx.com/
9. Internet of Things Institute Learning Center [електронний ресурс]. – режим доступу: https://education.ioti.com/
10. Internet of Things Roadmap to a Connected World [електронний ресурс]. – режим доступу: http://web.mit.edu/professional/digital-programs/courses/IoT/
11. IEEE Internet of Things Webinars [електронний ресурс]. – режим доступу: http://iot.ieee.org/education/webinars.html
12. Vermesan O. Internet of Things - From Research and Innovation to Market Deployment / O.Vermesan, P.Friess // River Publishers (2014)
13. Qiu F. Overall Framework Design of an Intelligent Dynamic Accounting Information Platform Based on the Internet of Things. / F. Qiu //iJOE 12 (5), 14-16 (2016)
14. Cai K. Innovative Experimental Platform Design and Teaching Application of the Internet of Things / K. Cai, F. Tie, H. Huang, H. Lin, H. Chen //iJOE 11 (6), 28-32 (2015)
15. Stage 1 - Introduction to the Internet of Things: What, Why and How [електронний ресурс]. – режим доступу: http://www.codeproject.com/Articles/832492/Stage-Introduction-to-the-Internet-of-Things
16. Табунщик Г.В. Нові технології в підготовці фахівців з вбудованих систем / Г.В.Табунщик, А.В.Пархоменко // Тези

доповіді Міжнародної науково-практичної конференції «Інтернет-Освіта-Наука-2014» «ІОН-2014» (14 - 17 жовтня 2014, Вінниця). – Вінниця: ВНТУ, 2014. - С.248-250
17. Притула А.В. Практично-орієнтовані методи викладання в галузі вбудованих систем/ А.В.Притула, Г.В.Табунщик, А.В.Пархоменко // Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій: Тези доповідей VII Міжнародної науково-практичної конференції (17–19 вересня 2014 р., м. Запоріжжя). – Запоріжжя: ЗНТУ, 2014.-С.216-218
18. Internet of things – Overview [електронний ресурс]. – режим доступу: http://www.codeproject.com/Articles/833234/Internet-of-things-Overview
19. IoT Path to Product: Smart Home [електронний ресурс]. – режим доступу: http://www.codeproject.com/Articles/1119436/IoT-Path-to-Product-Smart-Home
20. Умный дом [електронний ресурс]. – режим доступу: https://ru.wikipedia.org/wiki/
21. Как работает умный дом [електронний ресурс]. – режим доступу: http://smarthouse.ua/ru/kak_rabotaet_umnyj_dom.html
22. Для чего нужен умный дом [електронний ресурс]. – режим доступу: http://home-sapiens.ru/dlya-chego-nuzhen-umnyiy-dom/
23. Электронный фундамент [електронний ресурс]. – режим доступу: http://janto.ru/repository/004/01.html
24. Мозг умного дома [електронний ресурс]. – режим доступу: http://janto.ru/repository/004/04.html
25. Датчики для “Умного Дома” [електронний ресурс]. – режим доступу: http://key.ru/reviews_items/datchiki-dlya-umnogo-doma/;
26. Parkhomenko A. Internet-Based Technologies for Design of Embedded Systems, / A. Parkhomenko, O. Gladkova, S. Kurson, A. Sokolyanskii, E. Ivanov //Journal of Control Science and Engineering, 13 (2), 2015. – pp..55-63
27. Parkhomenko A. Development and Application of Remote Laboratory for Embedded Systems Design. / A. Parkhomenko, O. Gladkova, E. Ivanov, A. Sokolyanskii, S. Kurson // iJOE, 11 (3), 2015. – pp.27-31
28. Parkhomenko A. Integrated Complex for IoT Technologies Study/ A.Parkhomenko, A.Tulenkov, A.Sokolyanskii, Y.Zalyubovskiy,

A.Parkhomenko / Online Engineering & Internet of Things, Book series Lecture Notes in Network and Systems, Vol. 22, Chapter No:31 // Springer International Publishing, 2017.- pp 322-330

29. Remote and virtual tools in engineering: textbook/general editorship Dr.Ing.Karsten Henke.–Zaporizhzhya: Dike Pole, 2016.– 250 p.
30. Parkhomenko A. Implementation of Reusable Solutions for Remote Laboratory Development / A. Parkhomenko, O. Gladkova, A. Sokolyanskii, V. Shepelenko, Y. Zalyubovskiy // iJOE – Volume 12, Issue 07, 2016.- pp. 24-29
31. Parkhomenko A. Integration of IoT Technologies to IT Professionals Educational Process/ A. Parkhomenko, A. Parkhomenko, Y. Zalyubovskiy, M. Kalinina // Proceedings of the International Symposium on Embedded Systems and Trends in Teaching Engineering, Nitra, Slovakia, 12-15 September, 2016.-pp.241-245.
32. Пархоменко А.В. Разработка программно-аппаратной платформы для обучения технологиям Интернета вещей и устройств/ А.В.Пархоменко, А.В.Туленков, О.А.Поздняков, Я.И.Залюбовский //Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій: Тези доповідей VIII Міжнародної науково-практичної конференції (21–23 вересня 2016 р., м. Запоріжжя). – Запоріжжя: ЗНТУ, 2016.-С. 297-299
33. Poliakov M. Hybrid models of studied objects using remote laboratories for teaching design of control systems /M. Poliakov, T.Larionova, G. Tabunshchyk, A. Parkhomenko, K. Henke // iJOE – Volume 12, Issue 09, 2016.- pp.7-13
34. Tabunshchyk G. Remote Experiments For Reliability Studies Of Embedded Systems. / G. Tabunshchyk, D. Van Merode, P. Arras, K. Henke //Proceedings of the International Conference on Remote Engineering and Virtual Instrumentation, Madrid, Spain, 2016. - pp.68-71.
35. Poliakov M. Remote laboratory for teaching of control systems design as an integrated system / M. Poliakov, T. Larionova, G. Tabunshchyk, A. Parkhomenko, K. Henke // Proceedings of the International Conference on Remote Engineering and Virtual Instrumentation, Madrid, Spain, 2016. - pp. 333-340.

36. Gilibert M. 80C537 Microcontroller Remote Lab for E-Learning Teaching./ M. Gilibert, J. Picazo, M. Auer, A. Pester, J. Cusidó, J. Ortega // iJOE 2 (4), 2006. - pp. 1-3
37. Guerra H. Remote Experiments as an Asset for Learning Programming in Python / H. Guerra, A. Cardoso, V. Sousa, L. Gomes //iJOE 12 (4), 2016. – pp. 71-73
38. Ozvoldová M. Integration of Online Labs into Educational Systems. / M. Ozvoldová, P. Ondrusek // iJOE 11(6), 2015. – pp.54-59
39. Wuttke H.-D. Remote and Virtual Laboratories in Problem-Based Learning Scenarios. / H.-D. Wuttke, R. Ubar, K. Henke // Proceedings of the IEEE International Symposium on Multimedia, Taichung, Taiwan, 2010. - pp.377-382.
40. Cvjetkovic V. Overview of Architectures with Arduino Boards as Building Blocks for Data Acquisition and Control Systems. / V. Cvjetkovic, M. Matijevic // iJOE 12 (7), 2016.- pp.10-17
41. Raspberry Pi https://www.raspberrypi.org/ Open HAB [електронний ресурс]. – режим доступу: http://www.openhab.org/
42. Configuring the OpenHAB runtime [Електронний ресурс]. - Режим доступу: https://github.com/openhab/openhab/wiki/Configuring-the-openHAB-runtime#individual-configuration.
43. OpenHAB - стань программистом собственного жилища [електронний ресурс]. - режим доступу: https://habrahabr.ru/post/232969/
44. REST API [електронний ресурс]. – режим доступу: https://github.com/openhab/openhab/wiki/REST-API
45. FFmpeg [електронний ресурс]. – режим доступу: https://www.ffmpeg.org/.

2 ПРИКЛАДИ ПРАКТИЧНО-ОРІЄНТОВАНИХ IoT ПРОЕКТІВ

2.1 Проект SMART LIFE

На сьогоднішній день, вартість пропонованих рішень для систем домашньої автоматизації зазвичай достатньо висока, виробники пропонують не тільки власні програмні рішення, але і обладнання, що використовує різні протоколи та апаратні засоби, часто з надлишковим функціоналом, складним налаштуванням та обслуговуванням. Тому, метою даного проекту є розробка структури, апаратного та програмного забезпечення системи домашньої автоматизації на основі доступних компонентів та безкоштовної програмної платформи.

В результаті виконання проекту створена opensource система SMART LIFE, яка виконує основні функції Розумного будинку та реалізована на доступних платформах Arduino та OpenHAB, проста в налаштуванні та обслуговуванні.

Opensource проект SMART LIFE об'єднує декілька підсистем для створення справжнього Інтернету речей в офісі, в будинку, в квартирі, на присадибній ділянці, тощо. Система об'єднує декілька підсистем (рис.2.1):

- підсистема керування ролетами;
- підсистема керування мікрокліматом;
- підсистема керування кондиціонуванням;
- підсистема контролю доступу;
- підсистема контролю присутності;
- підсистема керування освітленістю;
- підсистема керування вуликом;
- підсистема керування теплицею.

Підсистема керування ролетами виконує функції контролю доступу в приміщення через вікна або двері, захищені ролетами, в необхідних ситуаціях, або за командами користувача. В процесі досліджень було виявлено декілька можливих підходів до побудови апаратної частини підсистеми. Найбільш доцільним та високотехнологічним є автоматичне керування ролетами, яке надає змогу організувати роботу з декількома ролетами одночасно, не потребує постійного обслуговування, реагує на задані користувачем дії, а також забезпечує автоматичне формування зворотніх впливів на

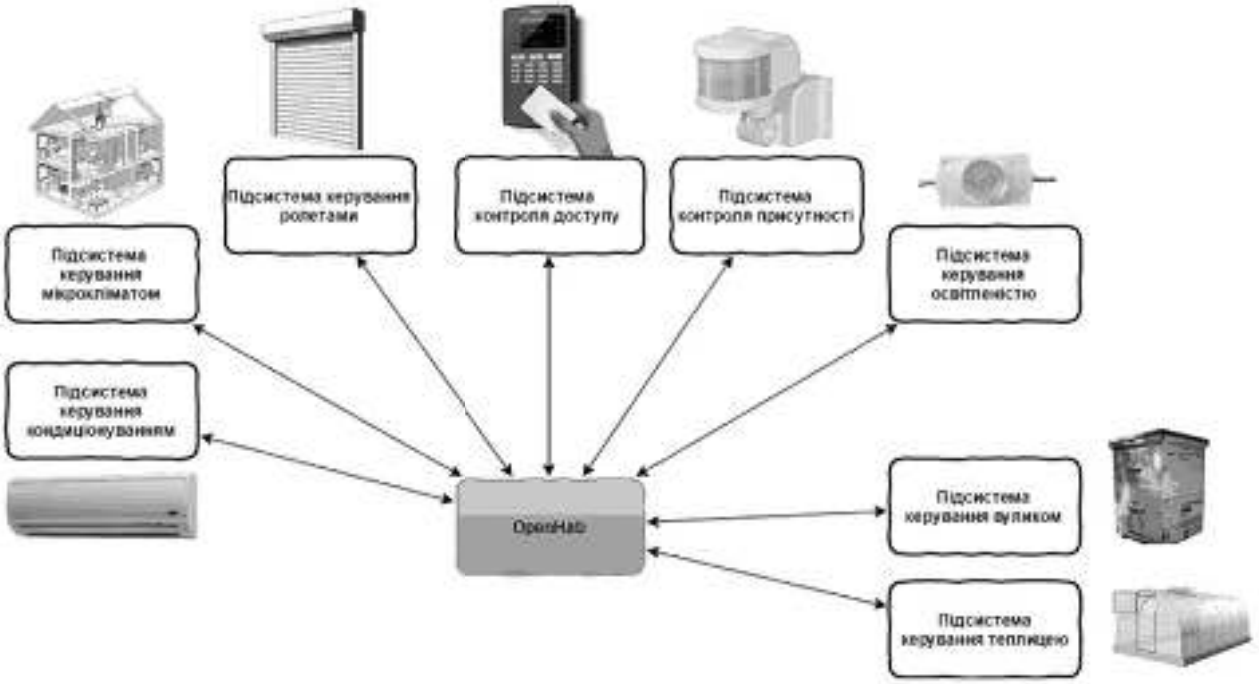

Рисунок 2.1 – Загальна структура системи SMART LIFE

виконавчі механізми у разі потреби (небезпечна або аварійна ситуація).

Після аналізу аналогічних підсистем керування ролетами, представлених на ринку автоматики, було зроблено висновок про відсутність рішень, які забезпечують сповіщення користувача про втручання у систему ролетів, з метою подальшого проникнення до приміщення.

З метою вибору компонентів підсистеми було досліджено механічні датчики з магнітокерованими феромагнітними контактами та напівпровідникові пристрої, що керуються напруженістю магнітного поля. Найбільш економічним та зносостійким виявився датчик Холла. Було обрано уніполярний датчик SS41 у корпусі TO92. Під час роботи датчика протікає електричний струм та при контакті з магнітом виникає поперечна різниця потенціалів, пропорційна напруженості магнітного поля.

Після аналізу модулів силового керування було обрано двоканальний модуль реле, з напругою 5V для Arduino з оптоелектронним керуванням. Оптоелектронна розв'язка забезпечує незалежність та завадостійкість контролеру керування. Для керування реле використовувались цифрові сигнали з контролеру керування. У якості контролеру для керування ролетами було обрано платформу Arduino Uno. Для реалізації програмної частини проекту використовувалося інтегроване середовище розробки Arduino IDE.

В результаті виконання роботи було створено прототип підсистеми керування ролетами, яка має наступні функції: відкриття та закриття ролет, обробка виключної ситуації, орієнтація положення ролет за допомогою датчиків Холла. Схема підключення компонентів представлена на рис. 2.2, прототип підсистеми на рис. 2.3, фрагмент тексту програми - на рис. 2.4. Надалі планується створення та використання інтерфейсу для підключення до однієї шини багатьох підсистем керування ролетами.

Підсистема керування мікрокліматом та кондиціонуванням.

Підсистема керування мікрокліматом реалізована на платформі Arduino Mega 2560 та під'єднаних до неї датчиків MQ135 та DHT11, що дозволяють зібрати дані про вологість та температуру повітря, а також концентрацію шкідливих газів у приміщенні. Схема підключення компонентів представлена на рис. 2.5, прототип підсистеми на рис. 2.6, фрагмент тексту програми - на рис. 2.7.

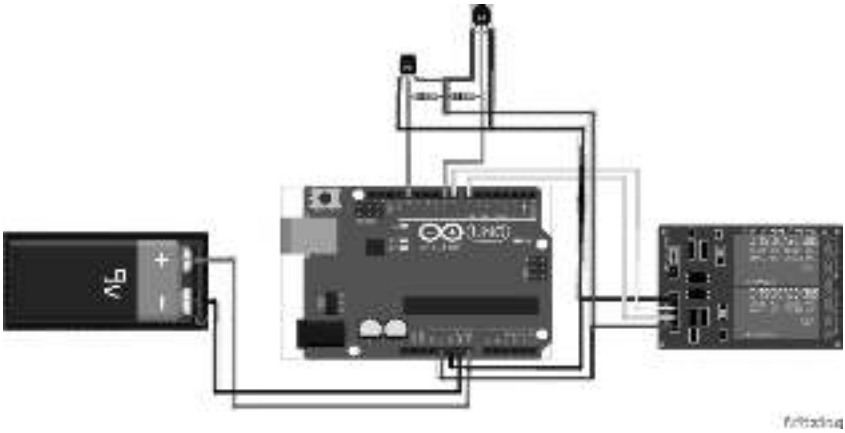

Рисунок 2.2 – Схема підключення компонентів
підсистеми керування ролетами

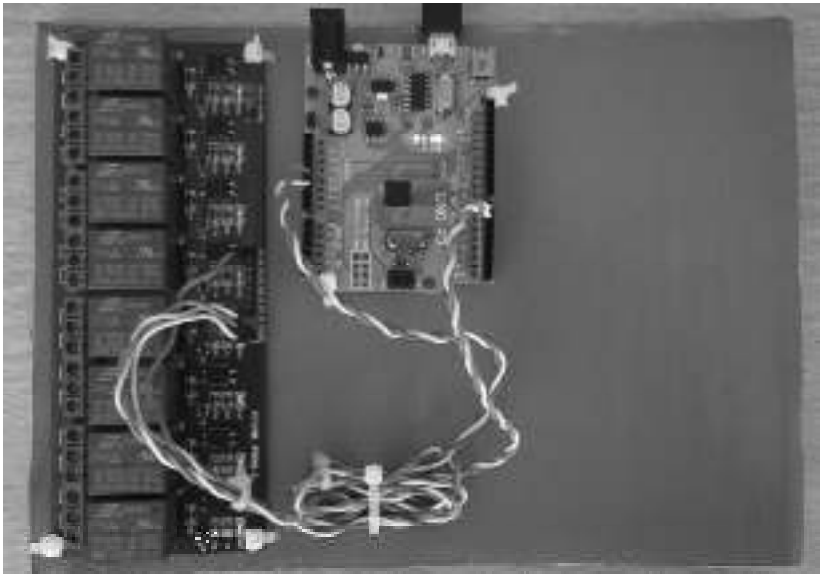

Рисунок 2.3 –Прототип підсистеми керування ролетами

Для реалізації підсистеми керування кондиціонуванням реалізовано схему, що містить IR-діод, інфрачервоний приймач TSOP4836, цифровий датчик температури DS18B20. Схема підключення компонентів представлена на рис. 2.8, прототип підсистеми на рис. 2.9, фрагмент тексту програми - на рис. 2.10. Передбачено віддалений режим керування, при якому користувач сам встановлює та перевіряє задані ним параметри мікроклімату, а також автоматизований режим, у якому користувач задає оптимальні параметри, а система сама їх регулює.

Рисунок 2.4 –Фрагмент тексту програми
для підсистеми керування ролетами

Підсистема контролю присутності та керування освітленістю. Дані підсистеми входять до складу системи SMART LIFE і надають можливість визначити наявність людини в приміщенні та створити потрібні умови освітленості. Використання цих підсистем дозволяє ефективно керувати енерговитратами, а також вирішувати задачі охорони об’єктів.

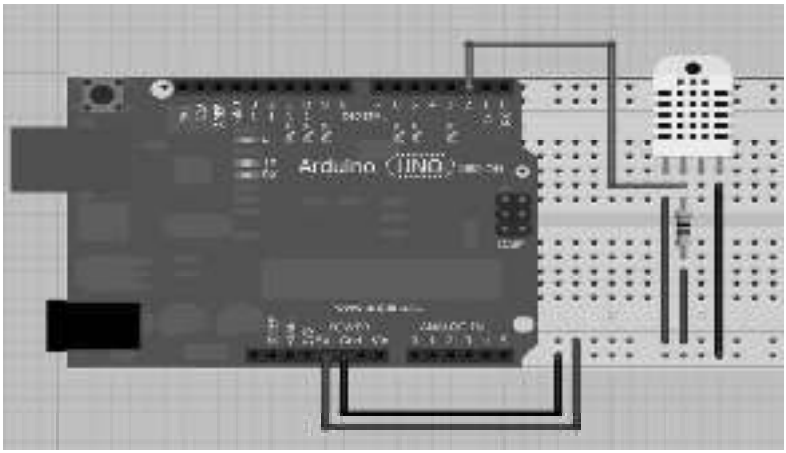

Рисунок 2.5 – Схема підключення компонентів
підсистеми керування мікрокліматом

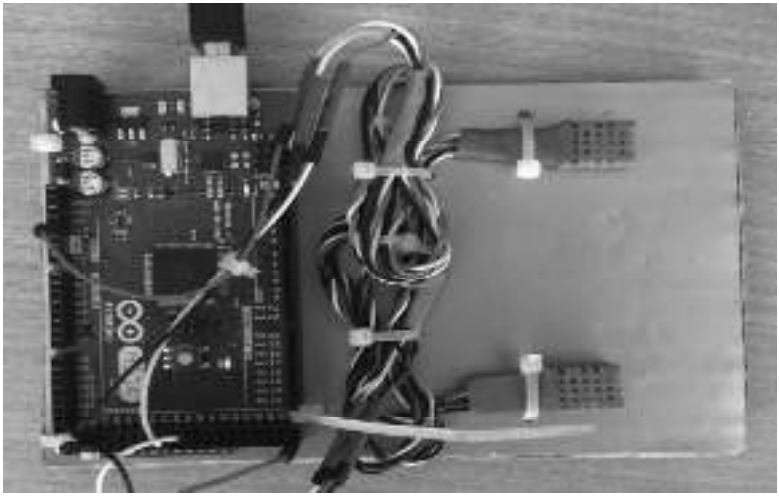

Рисунок 2.6 –Прототип підсистеми керування ролетами

Рисунок 2.7 –Фрагмент тексту програми
для підсистеми керування мікрокліматом

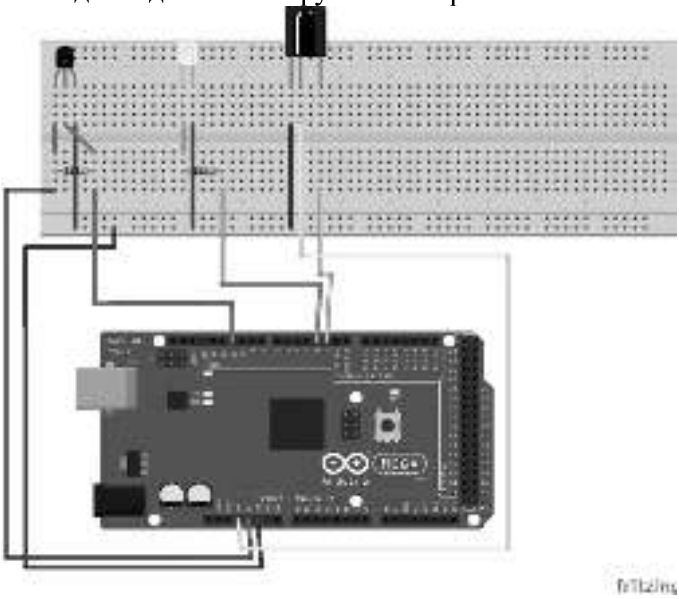

Рисунок 2.8 – Схема підключення компонентів
підсистеми керування кондиціонуванням

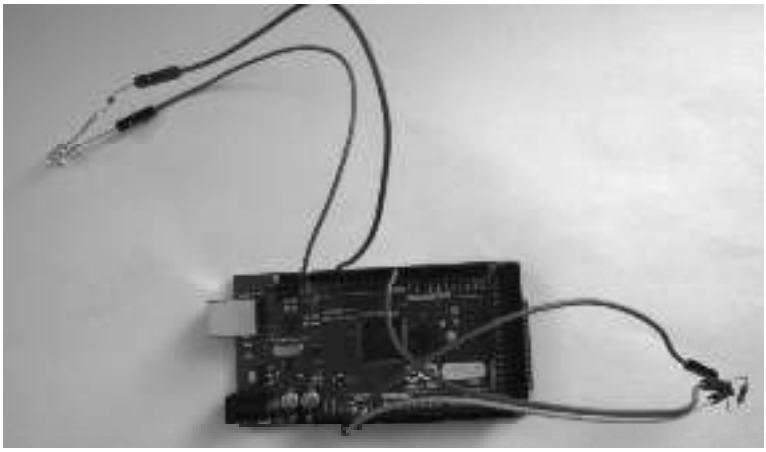

Рисунок 2.9 –Прототип підсистеми керування кондиціонуванням

Рисунок 2.10 –Фрагмент тексту програми
для підсистеми керування кондиціонуванням

Схема підсистеми контролю присутності створена на основі платформи Arduino Nano та піроелектричного інфрачервоного датчика руху PIR (HC-SR501). У середовищі розробки ArduinoIDE розроблено програму, яка виконує зчитування сигналів з датчика та виведення рядку інформації про рух у консоль. Після створення прототипу підсистеми та випробувань, було виявлено, що дана модель датчика не реагує на неживі предмети, а також тварин. Це дозволяє зробити

висновок про придатність даного датчика для систем освітлення та охорони, оскільки доцільно оцінювати саме присутність людини. Однією із головних проблем цього датчика є неможливість розпізнавання людини, якщо вона не рухається. Виявлено, що за умови, що у полі зору датчика знаходиться людина, котра не рухається (сидить, лежить, спить і т.д. і т.п.), то датчик не реагує на неї. Для розв'язку цієї задачі було проведено експеримент із вібрацією лінзи. Якщо, однократно, та із визначеною частотою вібрувати датчиком, покажчики з датчика оновлюються, це дає нам можливість заново визначити присутність людини. Реалізувати вібрацію лінзи можливо двома шляхами: за допомогою соленоїда (при подачі напруги на соленоїд, внутрішня частина витягується, при відключенні напруги внутрішня частина повертається у стартове положення, лінза вібрує, що дає змогу оновити покажчики на датчику); або за допомогою вібромотора (при подачі напруги на вібромотор (це потрібно робити однократно з визначеною частотою), лінза вібрує, тим самим оновлюючи покажчики на датчику). Схема підключення компонентів підсистеми контролю присутності представлена на рис. 2.11, прототип підсистеми на рис. 2.12, фрагмент тексту програми - на рис. 2.13.

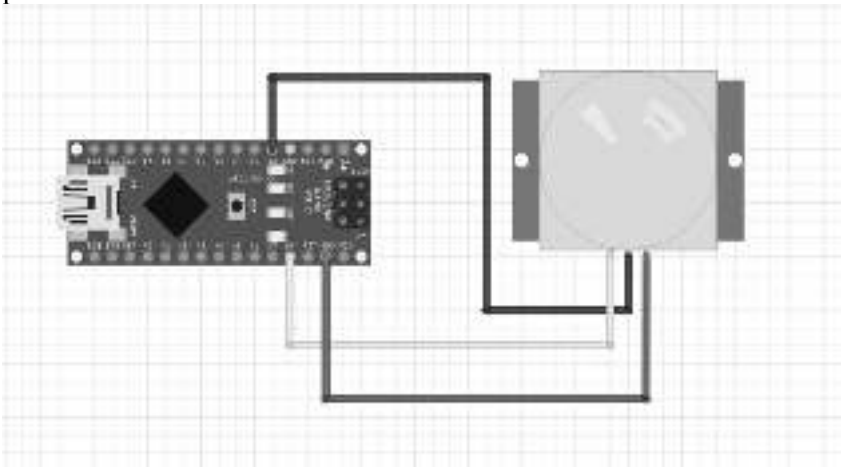

Рисунок 2.11 – Схема підключення компонентів підсистеми контролю присутності

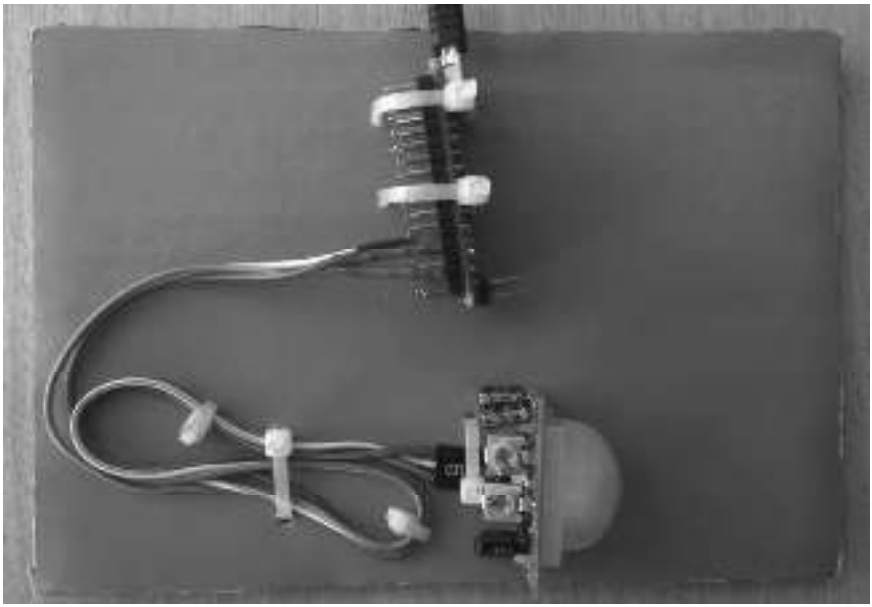

Рисунок 2.12 –Прототип підсистеми контролю присутності

Рисунок 2.13 – Фрагмент тексту програми
для підсистеми контролю присутності

Підсистема керування освітленістю може бути створена на основі відомих рішень компаній NooLite, Z-Wave, EnOcean, ZigBee. Використання даної підсистеми дає змогу контролювати включення/виключення джерел освітлення, створювати різні умови освітленості для певних зон в офісах та жилих приміщеннях в залежності від часу доби і яскравості природного світла. Це дозволить досягти значної економії енерговитрат, а також підвищити комфорт і працездатність користувачів. Схема підключення компонентів підсистеми керування освітленістю представлена на рис. 2.14, прототип підсистеми на рис. 2.15, фрагмент тексту програми - на рис. 2.16.

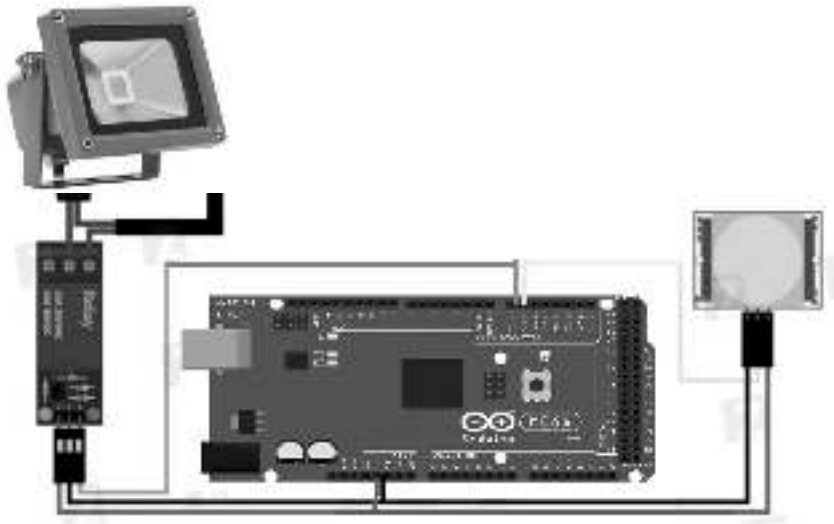

Рисунок 2.14 – Схема підключення компонентів
підсистеми керування освітленістю

Підсистема контролю доступу входить до складу системи SMART LIFE і виконує функції аутентифікації користувача для доступу в приміщення. В ході дослідження було виявлено декілька відомих технологій для реалізації даного завдання: розпізнавання користувача за біометричними даними (відбитки пальців, сітківка ока, голос) або з використанням смарт-карт.

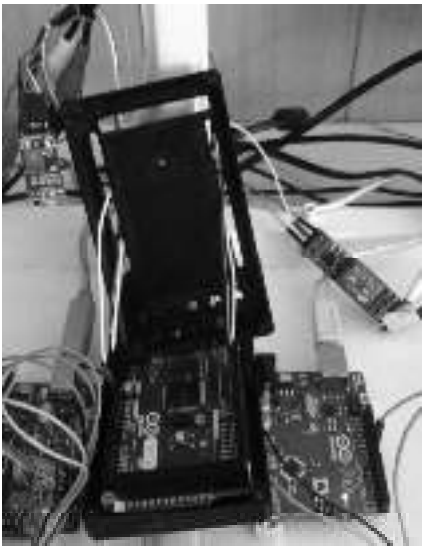

Рисунок 2.15 –Прототип підсистеми керування освітленістю

Рисунок 2.16 – Фрагмент тексту програми
для підсистеми керування освітленістю

Найбільш доцільною з точки зору мінімізації вартості та простоти реалізації є радіочастотна ідентифікація (RFID технологія), яка використовує радіочастотне електромагнітне випромінювання для читання/запису інформації на невеликий пристрій, який має назву смарт-карта. Завданням RFID системи є зберігання інформації про об'єкт з можливістю її зручного зчитування. Мітка може зберігати дані про тип об'єкта, вартість, вагу, температуру, а також будь-яку іншу інформацію в цифровій формі. RFID мітка дуже компактна і не вимагає джерела живлення, тому вона може бути захована в браслет, брелок, карту, телефон і навіть вшита під шкіру.

Розроблена підсистема складається з трьох базових компонентів: зчитувального пристрою, що має назву передавач/приймач (reader); антени, яка випромінює електромагнітні хвилі, що активізують RFID-мітку та дозволяють проводити запис і зчитування даних з цієї мітки; радіочастотних міток (смарт-карт) з вбудованою антеною, приймачем і передавачем. Схема підключення компонентів підсистеми контролю доступу представлена на рис. 2.17, прототип підсистеми на рис. 2.18, фрагмент тексту програми - на рис. 2.19.

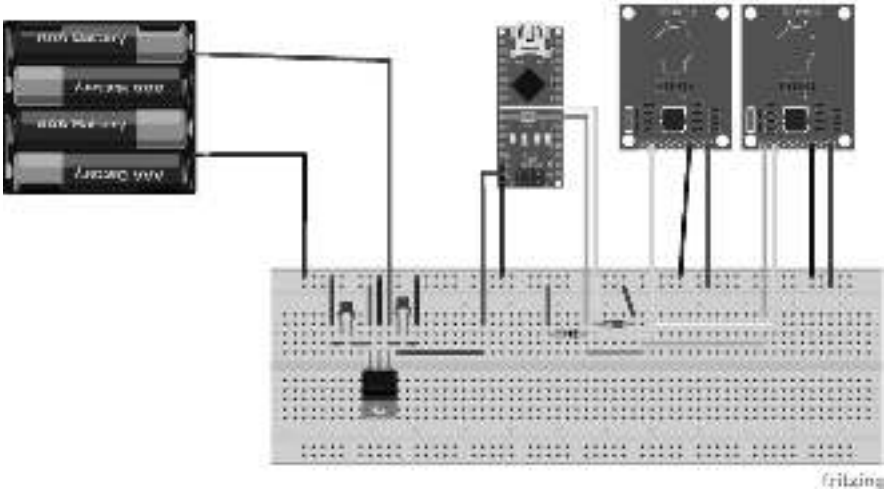

Рисунок 2.17 – Схема підключення компонентів
підсистеми контролю доступу

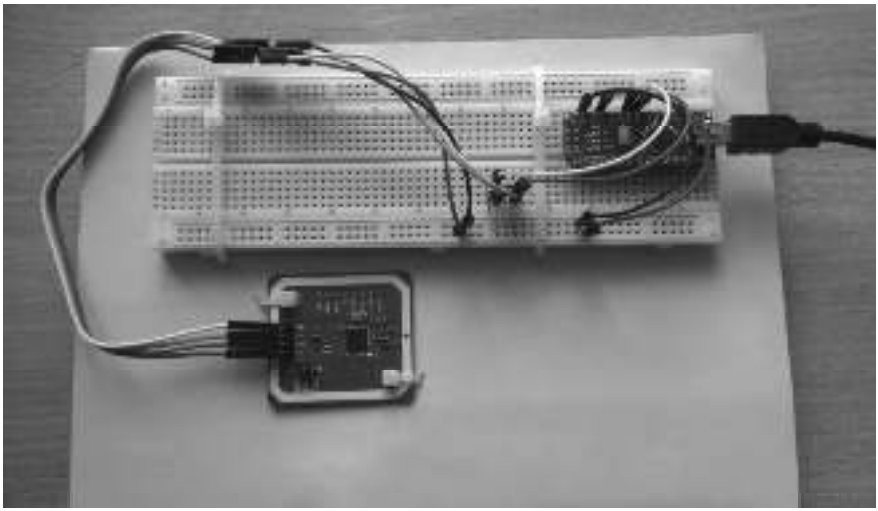

Рисунок 2.18 –Прототип підсистеми контролю доступу

Рисунок 2.19 – Фрагмент тексту програми
для підсистеми контролю доступу

Після дослідження ринку зчитувальних пристроїв, був обраний зчитувач на основі чіпу PN532 і антени, який підтримує три протоколи передачі даних High Speed UART, SPI, I2C і працює на частоті 13,56 МГц.

В ході розробки для взаємодії між платою і зчитувачем був обраний протокол I2C, який дозволить підключати відразу декілька пристроїв до однієї плати і взаємодіяти з ними на більшій відстані, ніж при використанні інших протоколів. Були протестовані декілька смарт-карт серії Mifare Classic, які мають 1 Кб пам'яті і містять по два ключі розміром в 48 біт на кожен сектор, а також підтримують апаратне шифрування Crypto-1.

У якості пристрою для управління було обрано платформу Arduino Nano. Для реалізації програмної частини проекту використовувалося інтегроване середовище розробки Arduino IDE.

Створений прототип підсистеми контролю доступу виконує наступні функції: зчитування даних з мітки, передача даних за допомогою I2C-протоколу до Arduino, виведення отриманої інформації на центральний модуль управління. Планується підключення і налагодження одночасної роботи декількох зчитувачів на різних дистанціях від блоку управління.

Підсистема курування вуликом. Як відомо, догляд за домашніми тваринами потребує часу, матеріальних витрат та певних зусиль. Якщо мова йде про бджіл, то найкращим місцезнаходженням пасіки є територія за містом, з доступом до великої кількості різноманітних рослин. Постійний контроль стану вулика дозволяє відслідковувати зміну кількості меду, а також показники внутрішнього мікроклімату. Але при цьому виникає гостра необхідність в дистанційному зборі ті аналізі даних.

Початковою стадією розробки підсистеми керування вуликом став аналіз вимог замовника. Були визначені наступні функціональні вимоги: можливість визначення ваги вулика; передача даних в систему openHUB; створення аналітичної інфографіки на сервері.

Базуючись на аналізі вимог, була спроектована архітектура підсистеми, що поєднує апаратну та програмну частини.

Наступним етапом створення «розумного вулика» став аналіз існуючих підходів та аналогів. Була виявлена тенденція збирання інформації про стан об’єктів шляхом зчитування даних з різноманітних датчиків з метою наступної обробки.

Вибір платформи засновувався на наступних характеристиках і властивостях: компактність, невисока ціна, великий набір периферійних пристроїв. Тому, була обрана проста у використання, а також популярна спеціалізована платформа Arduino Nano.

На етапі схемотехнічного проектування підсистема складається з наступних компонентів: платформа Arduino Nano з процесором ATmega328; аналогово-цифровий перетворювач HX711; тензодатчики ваги напівмостові.

Схема підключення компонентів підсистеми керування вуликом представлена на рис. 2.20, прототип підсистеми на рис. 2.21, фрагмент тексту програми - на рис. 2.22.

Принцип роботи тензодатчика заснований на зміні електричного опору резистора при його деформації під впливом сили, що була прикладена. Визначення ваги вантажу на вагах відбувається завдяки перетворенню сигналу, що надходить на аналоговий пін платформи, використовуючи коефіцієнт, який дозволяє отримати значення ваги в узгоджених одиницях. Регулювання точності перетворення можливе завдяки двом параметрам: коефіцієнту перетворення і кількості зчитувань, середнє значення яких буде виведено.

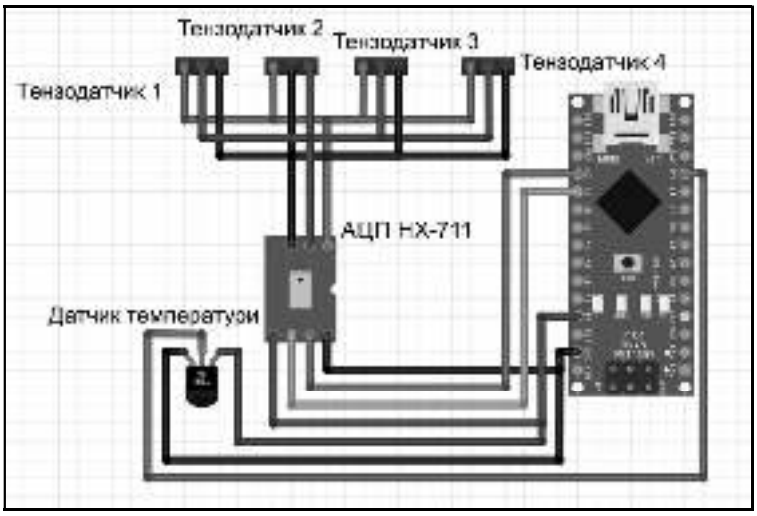

Рисунок 2.20 – Схема підключення компонентів підсистеми керування вуликом

Рисунок 2.21 – Набір тензодатчиків для прототипу підсистеми керування вуликом

Рисунок 2.22 – Фргмент тексту програми підсистеми керування вуликом

Підсистема керування теплицею здійснює контроль та управління мікрокліматом для забезпечення відповідних умов розвитку рослин. Представляє собою комплекс апаратних і програмних засобів, що дозволяє достовірно вимірювати показники температурно – вологісного режиму і на основі цього керувати виконавчими механізмами теплиці.

Дана підсистема виконує комплекс інформаційних і керуючих функцій, що забезпечують:

– задання добового циклу вологості і підтримання необхідного кліматичного режиму;

– контроль витрат води в каналі розпилення;

– збір, обробку та зберігання даних;

– представлення технологічної інформації в зручному для користувача вигляді;

– реєстрація подій і ведення журналу повідомлень (наприклад, при виході значення вологості за межі встановленого діапазону);

– підвищення продуктивності теплиці за рахунок автоматичної підтримки необхідних параметрів;

– віддалену відео трансляцію подій у теплиці.

Структура системи має два рівні. Нижній рівень представлений:

– регуляторами форсунок зволоження, відкриття клапана подачі води, включення і виключення опалювальної системи, і вентиляції;

– датчиками температури повітря, вологості грунту;

– мікроконтролерами моніторингу та управління регуляторами.

Верхній рівень – це операторська станція для збору та обробки даних. Функціональний склад підсистеми: плата ArduinoMEGA2560; DHT-11-датчик вологості та температури повітря; SRD-05VDS-модуль реле; YL-38-датчик вологості ґрунту; J34-електромагнітний клапан подачі води; резистор на 10кОм; блок живлення на 12В.

Схема підключення компонентів підсистеми керування теплицею представлена на рис. 2.23, прототип підсистеми на рис. 2.24.

Підсистема звукових повідомлень потрібна для вирішення задач інформування користувачів про події в розумному будинку або на присадибній ділянці. Наприклад, повідомлення про порушення периметру приватної території, зміну показників мікроклімату приміщення, перевищення рівня шкідливих викидів або продуктів згоряння в повітрі приміщення.

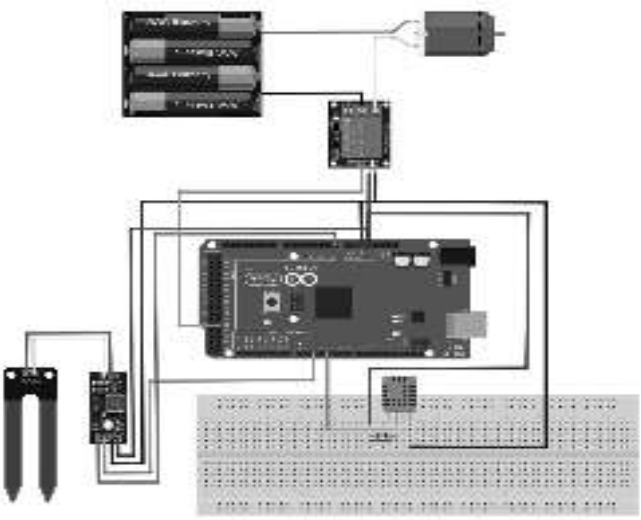

Рисунок 2.23 – Схема підключення компонентів
підсистеми керування теплицею

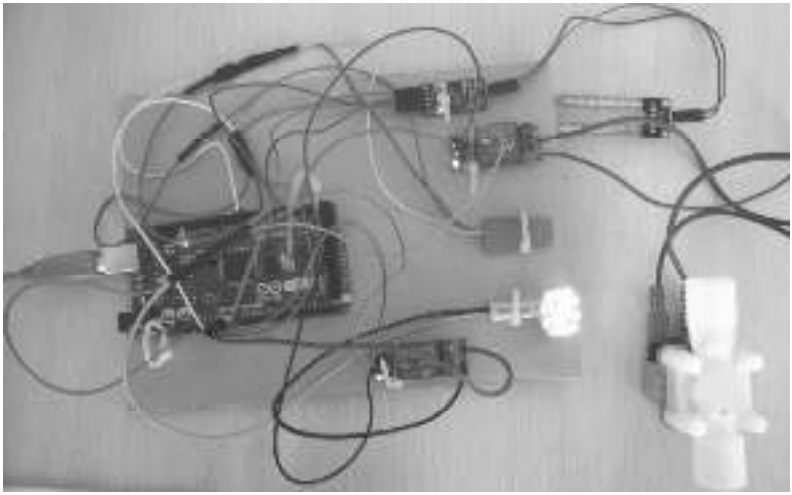

Рисунок 2.24 – Прототип підсистеми керування теплицею

Схема підключення компонентів підсистеми звукових повідомлень представлена на рис. 2.25, прототип підсистеми на рис. 2.26, фрагмент тексту програми на рис. 2.27.

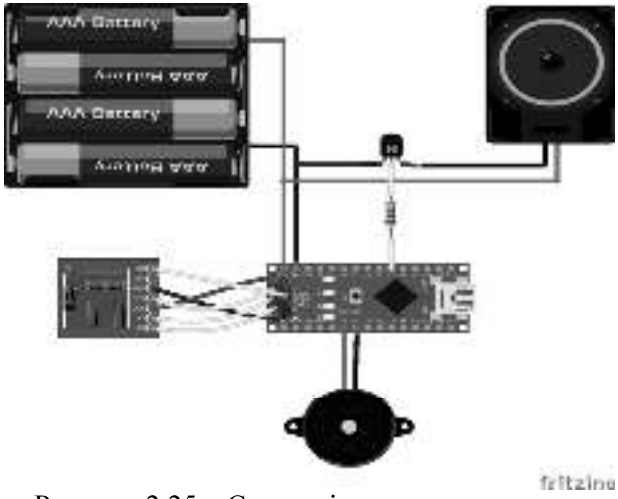

Рисунок 2.25 – Схема підключення компонентів підсистеми звукових повідомлень

Рисунок 2.26 – Прототип підсистеми звукових повідомлень

Рисунок 2.27 – Фрагмент тексту програми
підсистеми звукових повідомлень

Інтеграція підсистем системи SMART LIFE здійснюється на платформі OpenHAB (Open Home Automation Bus) (рис.2.28).

В результаті виконання роботи, розроблено систему домашньої автоматизації з використанням технологій IoT, а також прототип системи SMART LIFE.

Інтерфейс OpenHAB для розробленої системи наведено на рис. 2.29.

Команда розробників представлена на рис. 2.30.

Наступний етап – тестування системи в реальних умовах, на базі пасивного будинку в Київській області (http://ernst.kiev.ua/Jasnogorodka_ru.html) (рис. 2.31).

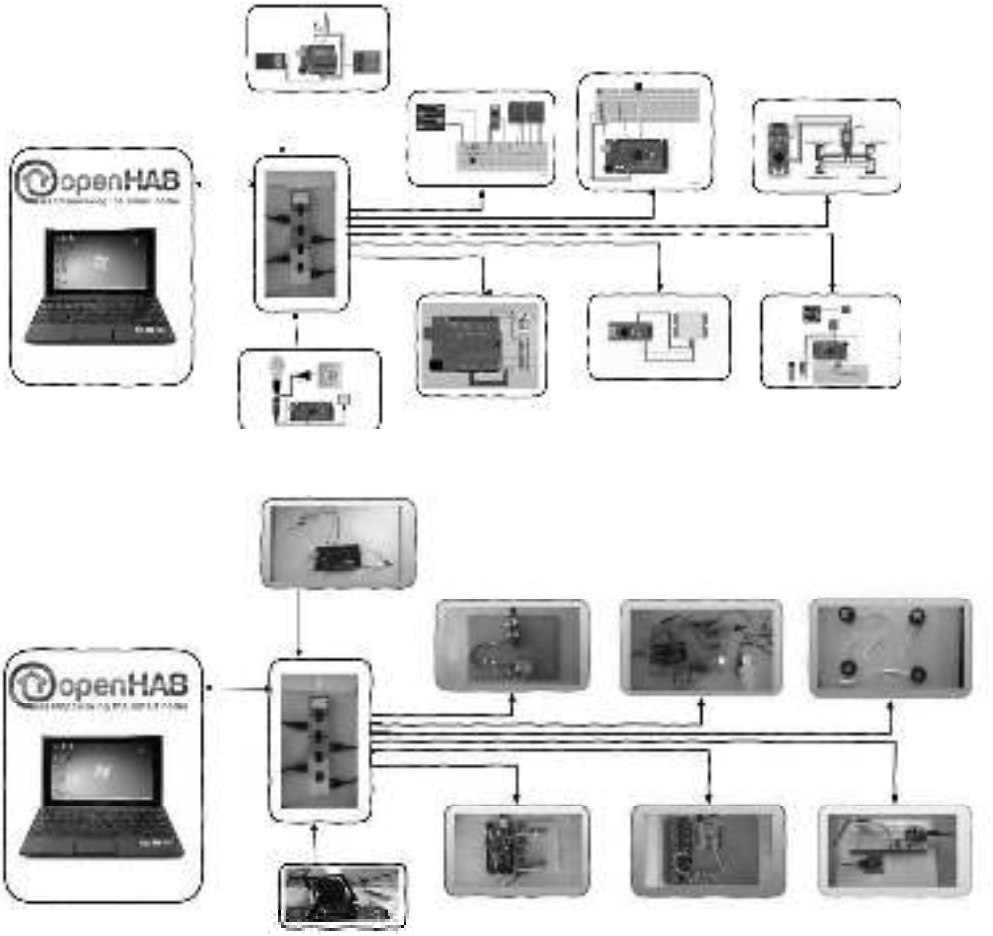

Рисунок 2.28 – Інтеграція підсистем на платформі OpenHAB

Рисунок 2.29 – OpenHAB інтерфейс системи SMART LIFE

Рисунок 2.30 – Команда розробників системи SMART LIFE

Рисунок 2.31 – Пасивний будинок

2.2 Система домашньої автоматизації

Система призначена для автоматизації процесу керування домашнім обладнанням та моніторингу використання ресурсів. Програмне забезпечення створене на основі платформ OpenHAB та Arduino Mega 2560 з використанням мови C++. Система є відкритою для додавання датчиків та виконавчих пристроїв, а також зміни інтерфейсу та підключення додаткових сервісів.

Система виконує наступні функції:

- управління кондиціюванням шляхом передачі керуючих сигналів (за умови контролю температури всередині приміщення);
- їизначення показників мікроклімату (температури та вологості всередині приміщення);
- відправка повідомлень про показники та стани системи на Telegram Messenger;
- виведення відомостей про прогноз погоди та астрономічний прогноз;
- контроль рівня освітлення та ввімкнення/вимкнення штучного освітлення за потреби;
- визначення рівня води в приміщенні (контроль затоплення).

Загальна структура розробленої системи представлена на рисунку 2.32.

В цілому система складається з комп'ютера з встановленим сервером OpenHAB та плати Arduino Mega 2560 з підключеними датчиками та виконавчими пристроями. Програмна частина складається з двох частин: програми для Arduino Mega 2560 та конфігураційних файлів для платформи OpenHAB.

Для передачі команд та обміну даними використовується послідовний порт, через який плата Arduino відправляє дані, отримані з підключених до неї датчиків та приймає команд від серверу. Сервер OpenHAB підключений до мережі Інтернет, що дозволяє забезпечити користувачу доступ до системи із будь якої точки світу за допомогою my.openhab, а також передачу повідомлень через Telegram Messenger.

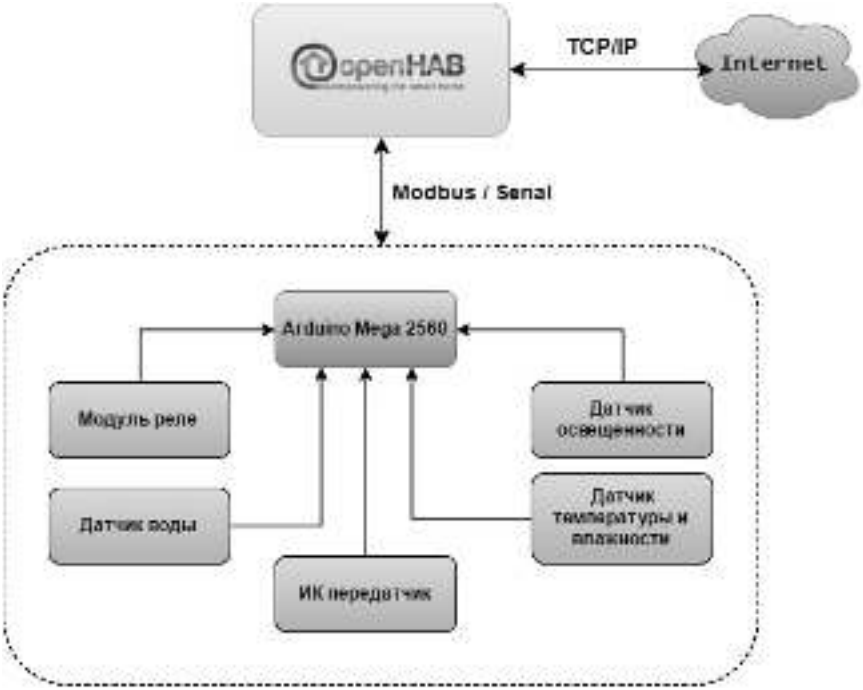

Рисунок 2.32 – Структура системи

Від датчиків, що підключені до плати Arduino, в систему надходять дані про рівень освітлення та води, а також про температуру та вологість в приміщенні. Система порівнює ці дані з заданими та виконує керування зовнішніми пристроями (системою освітлення за допомогою реле та кондиціонером шляхом безпосереднього посилання керуючих сигналів за допомогою інфрачервоного передавача). Крім того, система виводить отримані дані та стани зовнішніх пристроїв в інтерфейс та надсилає користувачу на Telegram Messenger.

Після запуску програми починається ініціалізація змінних та портів вводу-виводу. Потім програма переходить в головний цикл, у якому вона очікує команди від OpenHAB. Після отримання команди йде розпізнавання типу команди та відповідних дій.

Arduino Mega 2560 - це програмно-апаратна платформа з власним процесором і пам'яттю. До платформи Arduino були

підключені модуль реле, інфрачервоний передавач, а також наступні датчики (рис.2.33):

- аналоговий датчик рівня рідини (рис.2.34) із робочою напругою DC3-5V та током живлення до 20 мА;
- модуль датчика світла (рис.2.35) з пороговим компаратором, поріг спрацьовування якого регулюється змінним резистором. Основні характеристики: чутливий елемент - фоторезистор; вихід компаратора більше ніж 15 мА; регулювання порога спрацьовування змінним резистором; робоча напруга: від 3.3В до 5В; цифровий вихід компаратора (0 і 1).;
- датчик температури та вологості DHT11 (рис.2.36). Це цифровий датчик, що дозволяє калібрувати цифровий сигнал на виході. Складається з ємнісного датчика вологості і термістора, містить в собі АЦП для перетворення аналогових значень вологості і температури. Характеристики: робоча напруга: від 3.3В до 5В; визначення вологості 20-95% з 5% точністю; визначення температури 0-50 град. з точністю 2 град; частота опитування не більше 1 Гц (не більше одного разу за 1 сек.);

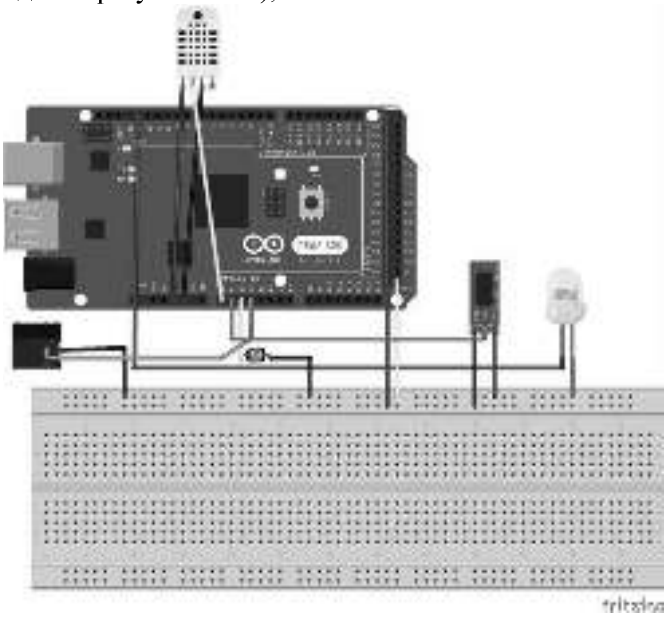

Рисунок 2.33 - Схема підключення компонентів

Рисунок 2.34 - Аналоговий датчик рівня рідини

Рисунок 2.35 - Модуль датчика світла

Рисунок 2.36 - Датчик температури та вологості DHT11

Після запуску програми, користувач бачить головну сторінку інтерфейсу системи (рис. 2.37), яка містить наступні елементи керування:

- Room - для переходу на сторінку управління обладнанням приміщення;
- Outside Temperature - для переходу на сторінку із більш детальною інформацією про погоду (рис. 2.38),
- Astronomical Data - для переходу на сторінку із астрономічними відомостями (рис. 2.39);
- Date - відображає поточну дату.

Рисунок 2.37 - Головна сторінка інтерфейсу

Рисунок 2.38 - Сторінка із інформацією про погоду

Сторінка із панеллю для керування обладнанням та інформацією з датчиків у приміщені зображена на рисунку 2.40.

Рисунок 2.39 - Сторінка із астрономічними даними

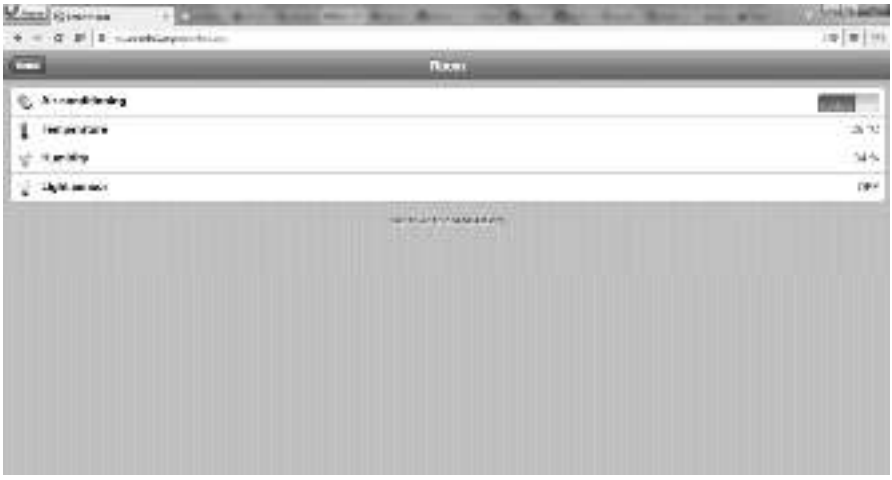

Рисунок 2.40 - Сторінка із панеллю керування обладнанням

У першому рядку є кнопка для ручного включення/виключення кондиціонеру. Однак, все керування кондиціонером зазвичай відбувається автоматично під час виконання власне написаних правил.

Для даної системи було написано декілька таких правил для керування кондиціонером.

Користувач може також керувати системою віддалено за допомогою мобільного додатку OpenHAB, завдяки підключенню до my.openhab (рис. 2.41).

Рисунок 2.41 - Головна сторінка у мобільному додатку.

Ця сторінка містить ті самі кнопки та поля, що і базовий інтерфейс системи.

Таким чином, користувач, використовуючи елементи керування інтерфейсу, може отримувати дані про про прогноз погоди та астрономічний прогноз, показники мікроклімату, контролювати рівень води та освітлення в приміщенні, вмикати/вимикати кондиціонер в ручному режимі. Крім того, передбачене вмикання/вимикання освітлення та кондиціонування в автоматичному режимі.

2.3 Система віддаленого моніторингу ресурсів

Система моніторингу ресурсів дозволяє користувачам аналізувати інформацію щодо водо- та енергопостачання, яка надходить з контролеру, підключеного до мережі Інтернет.

Клієнтська частина реалізовує наступні можливості:

- підключення группи користувачів;
- графічне відображення процесу використання води;
- відображення загальної кількості використаної води;
- відображення графіків використання води за добу, тиждень, місяць різними користувачами;
- відображення графіка залежності використання води в реальному часі;
- відображення графіка залежності температури води відносно часу;
- відображення загальної віртість використаної води за день, місяць, рік по кожному користувачу;
- відображення кількості використаної електроенергії по кожному користувачу;
- відображення даних (рік, місяць та час останього підключення) кожного користувача;
- відображення напруги та сили струму по кожному користувачу.

Вимоги до серверної частини:

- система повинна цілодобово збирати і обробляти дані, усуваючи людський фактор;
- оновлення даних повинно відбуватися кожні 15 секунд.

Вимоги до складу і параметрів технічних засобів, а також до інформаційної і програмної сумісності:

- не потребувати багато ресурсів ПК;
- працювати в будь-яких операційних системах;
- забезпечувати доступ користувачам для перегляду інформації через мережу Інтернет в будь-який час доби.

Додаткові вимоги:

- можливість додавання пристроїв збору даних, що працюють з різними параметрами;

- при нормальній роботі не потрібно ручних дій з боку користувача.

Переваги системи :

- дані з усіх витратомірів збираються в єдиному сховищі, яке знаходиться на сервері;
- система цілодобово збирає і обробляє дані, усуваючи людський фактор.

Система організована за принципом клієнт-серверної архітектури. Клієнт-сервер - обчислювальна або мережева архітектура, в якій завдання або мережеве навантаження розподілено між постачальниками послуг або серверами, і замовниками послуг або клієнтами. Фізично клієнт і сервер - це програмне забезпечення. Вони взаємодіють через Інтернет за допомогою мережевих протоколів і знаходяться на різних обчислювальних машинах. Сервер очікує від клієнтів запити і надає їм свої ресурси у вигляді даних і сервісних функцій.

Переваги архітектури:

- відсутність дублювання коду програми-сервера програмами-клієнтами;
- обробка виконується на сервері, отже вимоги до комп'ютерів, на яких запущено клієнт, знижуються;
- всі дані зберігаються на сервері, який, зазвичай, захищений набагато краще за більшість клієнтів;
- на сервері простіше організувати контроль повноважень, щоб дозволити доступ до даних тільки клієнтам з відповідними правами доступу.

Недоліки архітектури:

- непрацездатність сервера може зробити непрацездатною всю обчислювальну мережу;
- підтримка роботи даної системи вимагає фахівця - системного адміністратора;
- висока вартість обладнання.

Архітектура розроблюваної системи зображена на рис. 2.42.

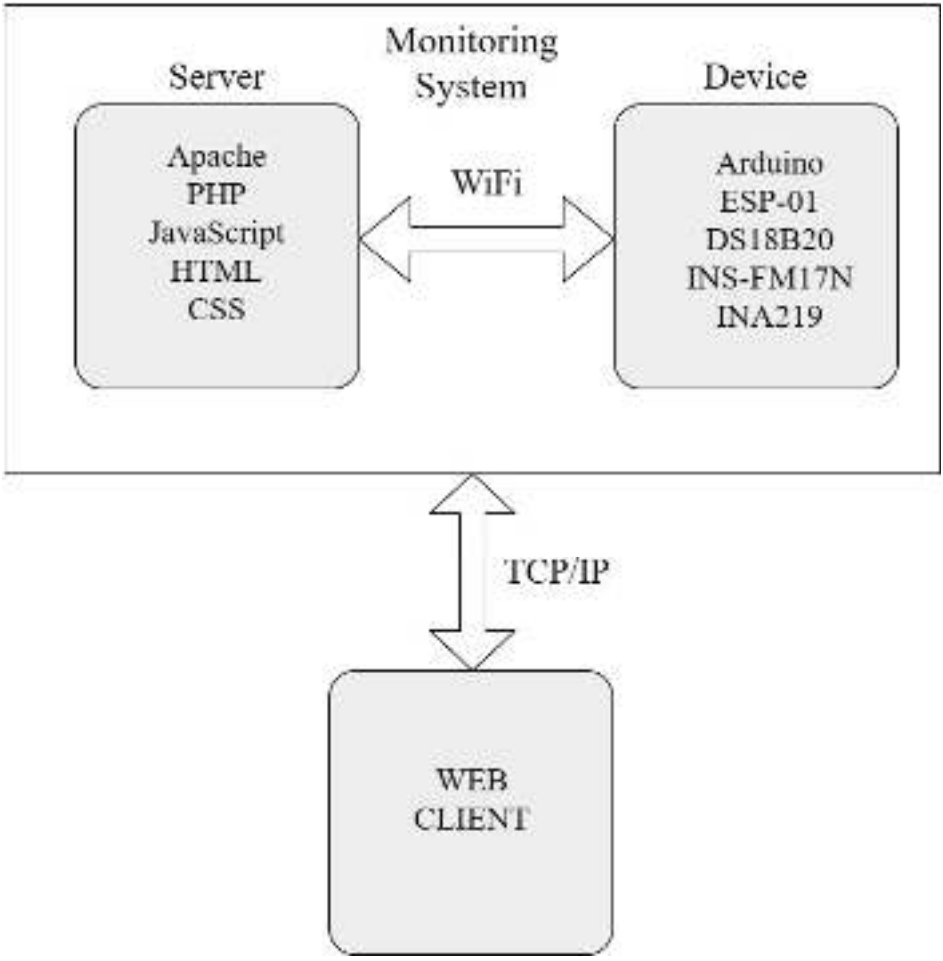

Рисунок 2.42 – Архітектура системи

Після детального огляду можливих компонентів систем, було вирішено обрати такі апаратні складові.

- Arduino Nano V3;
- WiFi модуль ESP8266-01 для передачі даних серверу;
- датчик потоку води YF-S201;
- датчик температури DS18B20 в захисному водонепроникному корпусі;
- цифровий датчик струму і напруги Zero CJMCU-219 на мікросхемі INA219.

Збір даних з датчиків виконує плата Arduino Nano V3 (рис. 2.43). Має невеликі розміри, що дуже доцільно, оскільки пристрій повинен бути компактним. Для використання контролера потрібно підключити його через USB - miniUSB кабель до USB порту комп'ютера. Альтернативою може послужити підключення зовнішнього блоку живлення або батареї, що теж є важливим для вибору, оскільки пристрій повинен мати можливість працювати від електромережі.

Рисунок 2.43 – Arduino Nano V3

WiFi модуль ESP8266-01 використовується для передачі даних серверу (рис. 2.44). Це мініатюрний WiFi модуль на базі новітньої мікросхеми ESP8266 з вбудованим стеком протоколу TCP / IP і управлінням AT-командами. Чіп створений для використання в розумних розетках, mesh-мережах, IP-камерах, бездротових сенсорах, електроніці, що носиться і таке інше. ESP8266 може стати мозком

майбутнього для реалізації технологій Інтернету речей. Передбачено два варіанти використання чіпа: у вигляді моста UART-WIFI, коли модуль на базі ESP8266 підключається до існуючого рішення на базі будь-якого іншого мікроконтролера і управляється AT-командами, забезпечуючи зв'язок рішення з інфраструктурою та реалізуючи нове рішення, яке використовує сам чіп ESP8266 в якості керуючого мікроконтролера. В нашому випадку ми обрали перший варіант, оскільки, можливості другого варіанту не задовольнять виконання функціональних можливостей системи.

Рисунок 2.44 – WiFi модуль ESP8266-01

Для реалізації функції зняття показань водного потоку для системи моніторингу ресурсів для було обрано датчик YF-S201 (рис. 2.45). Датчик потоку води складається з пластикового клапана з двома штуцерами з різьбою 1/2 дюйма, водного ротора і датчика Холла. Коли вода проходить крізь датчик, ротор обертається зі швидкістю, пропорційною швидкості потоку води. Датчик Холла фіксує кожен оборот і передає отриманий сигнал

Характеристики датчика потоку води :

- робоча напруга: 5-24В;
- максимальний струм: 15мА;
- маса: 43 гр.;
- вимірюваний діапазон: 1 ~ 30 літрів на хвилину;
- робоча температура 0 ° C ~ 80 ° C;
- температура вимірюваної рідини <120 ° C;

- робоча вологість 35% ~ 90%;
- робочий тиск ≤2МПа;
- температура зберігання -25 ° C ~ +80 ° C;
- вологість зберігання 25% ~ 90%;
- розміри 56x33x29мм.

Рисунок 2.45 – Датчик потоку води

Для вимірювання температури води використано датчик DS18B20. Датчик температури DS18B20 в захисному водонепроникному корпусі. Діапазон температур, вимірюваних датчиком знаходиться в межах -55C ... + 125C. Але оскільки захисна оболонка датчика зроблена з ПВХ, то рекомендується верхній діапазон виміру обмежити 100 градусами (рис. 2.46).

Рисунок 2.46 – Температурний датчик води DS18B20

Характеристики датчику температури :

- червоний дріт - VCC (живлення);
- зелений (Синій, Жовтий) провід - DATA (дані);

- жовтий (Чорний, Чорний) провід - GND (земля);
- робоча напруга дані / живлення від 3В до 5.5В;
- точність ± 0.5 ° С в діапазоні -10 ° С до + 85 ° С;
- робочий діапазон температур від -55 до 125 ° С;
- вибір 9 чи 12 бітної розрядності;
- інтерфейс 1-Wire;
- унікальний 64 бітний ID в кожному чіпі;
- паралельне включення сенсорів;
- зонд з нержавіючої сталі діаметром 6 мм і довжиною 50 мм;
- кабель діаметром 4 мм і довжиною 100 см.

Для вимірювання показників електроспоживання було обрано цифровий датчик Zero CJMCU-219, призначений для вимірювання таких параметрів постійного струму як напруга, струм і споживана потужність. Модуль виконаний на мікросхемі INA219 - вимірювачі струму і напруги з нульовим дрейфом і володіє малими розмірами і вагою при високій точності вимірювань. Мікросхема вимірює параметри протікання струму в будь-якому напрямку з автоматичним перемиканням полярності вимірювання. Застосувати модуль можна в системах, які контролюють процес заряду і розряду акумуляторних батарей, джерелах живлення з контролем напруги і споживаного навантаженням струму. Можливості зміни I2C адреси вимірювача дозволяють підключити на одну шину до 4-х таких пристроїв. Для збільшення точності вимірювань передбачений регістр калібрування (рис. 2.47).

Характеристики датчика струму і напруги:

- тип модуля: INA219;
- робоча температура: від -40С до 85С;
- дрейф в робочому температурному діапазоні: 100мкВ;
- роздільна здатність вимірювача: 12-біт;
- інтерфейс: I2C;
- швидкість інтерфейсу: 3,4МГц;
- максимальна вимірювана напруга: + -26 В;
- калібрування: калібрувальний регістр;
- внутрішні дані: вимірюваний струм і потужність;
- фільтрація: x128 відліків;
- напруга живлення: від 3В до 5В.

Рисунок 2.47 – Датчик струму і напруги

Схематичне підключення компонентів системи наведено на рис. 2.48.

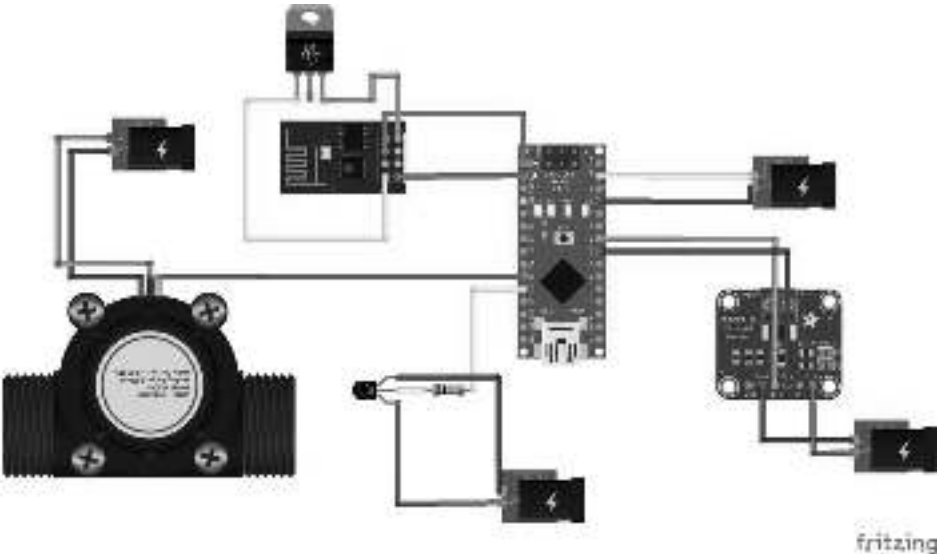

Рисунок 2.48 – Схематичне підключення компонентів системи

Загальний алгоритм роботи системи (рис.2.49) можна описати таким чином.

1. Підключення пристрою до мережі.
2. Збір даних з усіх датчиків.
3. Обробка даних на контролері, занесення у змінні показання.
4. Відправка даних по WiFi на роутер.
5. Передача даних на сервер.
6. Обробка та візуалізація даних на сервері.
7. Передача сформованої сторінки web-клієнту.

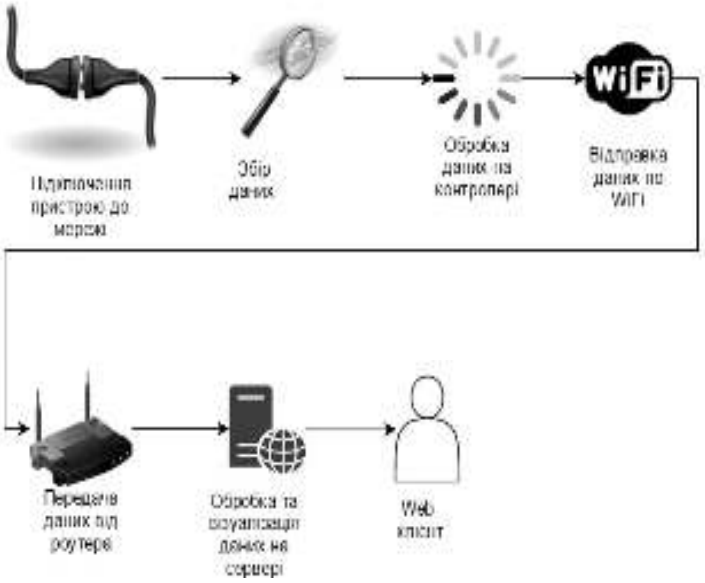

Рисунок 2.49 – Алгоритм роботи системи

Структура апаратної складової системи моніторингу ресурсів показує використовувані способи передачі даних та відношення між структурними компонентами системи (рис. 2.50). На рисунку 2.51 представлено інтерфейс системи.

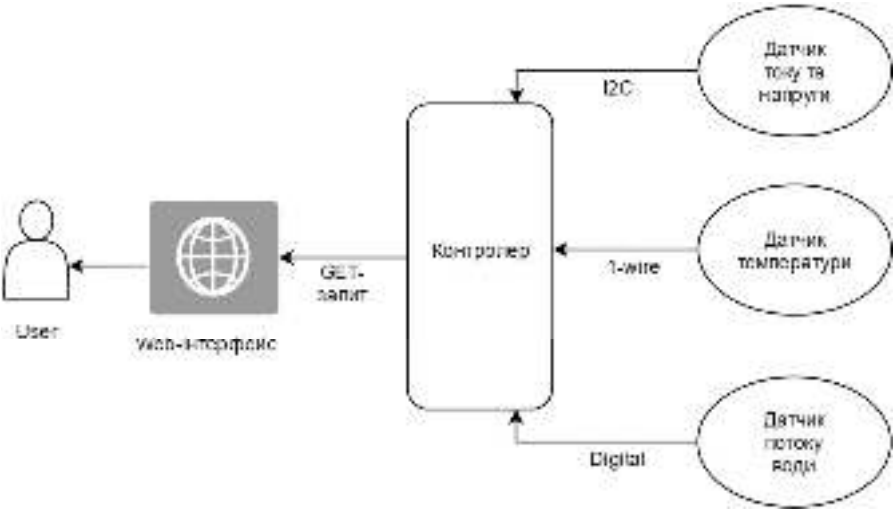

Рисунок 2.50 – Структура апаратної складової системи

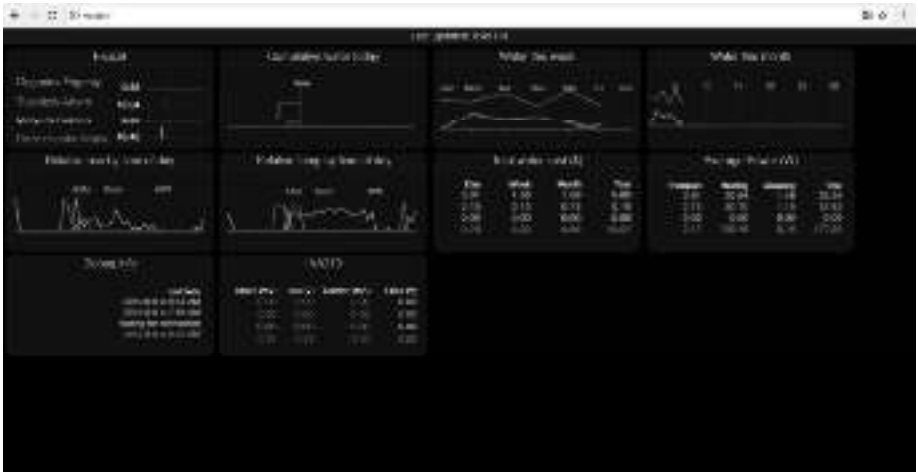

Рисунок 2.51 – Інтерфейс системи

Для збору даних використовується контролер підрахунку імпульсів. До одного такого контролеру збору може бути підключено до 8 витратомірів. Передбачається, що контролер збору буде змонтований поблизу витратомірів, а сам контролер знаходитись у зони дії WiFi мережі. Контролер являє собою схему на базі мікропроцесора, який виконує дві основні функції - підрахунок імпульсів від лічильників і зв'язок по мережі WiFi.

Кожному пристрою присвоюється унікальний ідентифікаційний ключ, котрий відповідає кожному користувачу з відповідним кольором (рис. 2.52).

Рисунок 2.52 – Список підключених користувачів

Дані, отримані контролером, передаються на сервер методом GET. Цей метод використовується для запиту вмісту зазначеного ресурсу. За допомогою методу GET можна також почати будь-який процес. У цьому випадку в тіло листа у відповідь слід включити інформацію про хід виконання процесу.

Клієнт може приймати параметри виконання запиту в URI цільового ресурсу після символу «>»:

GET / path / resource? Param1 = value1 & param2 = value2 HTTP / 1.1

А саме GET-запит із параметрами:

– userID – унікальний ключ користувача, котрий надається кожному пристрою;

– ticks – кількість імпульсів, отриманих від датчика потоку води;

– temp – температура води;

– secs – константа, оскільки оновлення виконується кожні 15 секунд;

– shunt – напруга на шунті;

– bus – напруга на шині;

– curr – поточна сила струму;

– load – загальна напруга.

Після того, як дані були отримані, параметри записуються у текстовий файл із назвою, відповідною до значення параметра userID. Якщо такого користувача немає або не було передано параметри, сервер виконує запис помилки у текстовий файл errLog.txt із поміткою дати та часу.

Отримані дані переходять до головного подання index.php, де спочатку йде статична, а потім динамічна частина. Отримані дані використовуються у динамічній частині. Ця частина в основному написана на JavaScript. Ця частина відповідає за обробку та візуалізацію даних. Вона розбита на функції, котрі виконують свою частину обробки. Список основних функцій :

– clearAndLoadAllUserData() – створює порожній масив allUsers та викликає функцію getUserData();

– getUserData() – додає дані у масив allUsers;

– updateUIWhenReady() – оновлює час останнього оновлення системи;

– parseDataAndAddToUser(data, user) – розбиває отримані параметри комою та додає їх у одноіменні змінні;

– updateUI() – оновлює час;

– function DataPt(epoch, fullDate, secs, ticks, temp, shunt, bus, curr, load) – округляє отримані дані.

2.4 Автоматизована система Розумний офіс

Система Розумний офіс має виконувати наступні функції: керування освітленістю, контроль доступу в приміщення за допомогою RFID міток, повідомлення у випадках затоплення, реакція

на випадкові ситуації та мережева взаємодія. Базове обладнання системи, надане замовником, наведене на рис. 2.53.

Система дозволяє через GSM модуль або по мережі TCP / IP передавати повідомлення при затопленні або спробі зламу, отримувати поточні показання з датчиків і зчитувачів, керувати освітленням і дверними замками.

Загальна архітектура системи (рис.2.54) включає в себе інтерфейсну плату, до якої підключаються периферійні пристрої: датчики руху, датчик води та RFID мітки. До модулю реле, розташованому на інтерфейсній платі, підключається кероване устаткування (лампи, замки). Крім того, на інтерфейсній платі розташовані Інтернет-модуль, який відображає дані з датчиків на сервері, а також GSM модуль, що відправляє повідомлення на заданий номер при виникненні виняткових ситуацій (злам, затоплення).

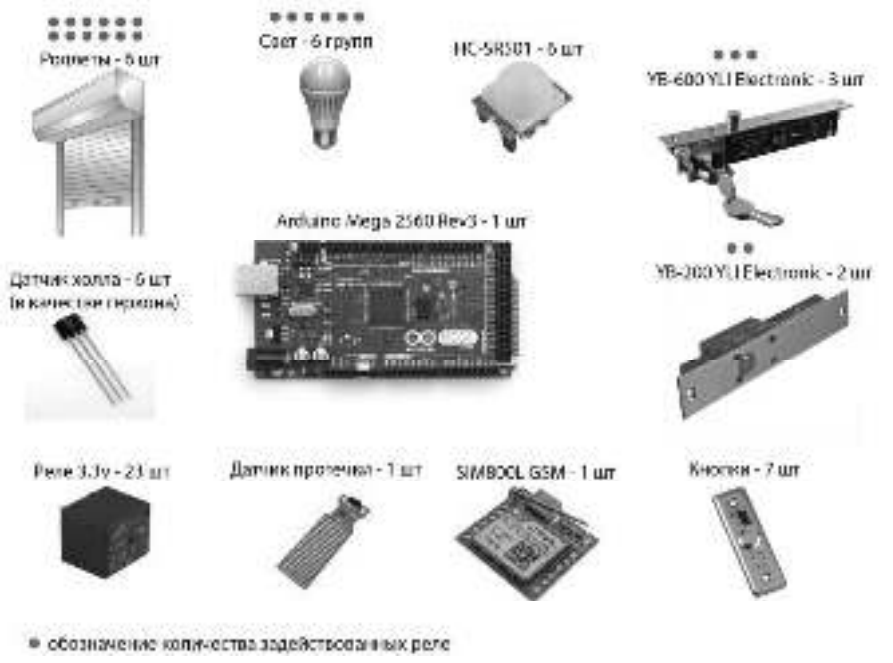

Рисунок 2.53 – Базове обладнання системи

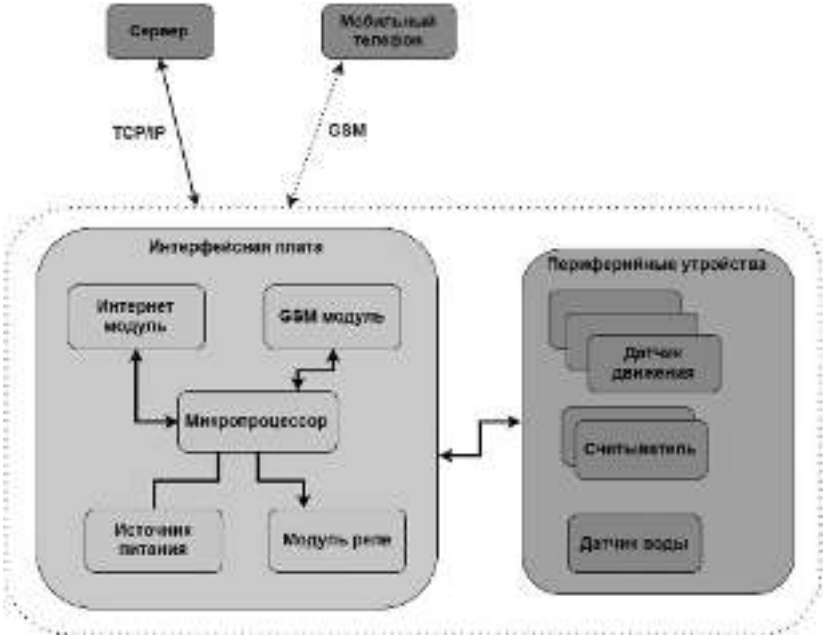

Рисунок 2.54 – Загальна архітектура системи

Дослідження ринку компонентів для реалізації даної задачі дозволило запропонувати складові прототипу системи (рис 2.55). Основою прототипу розробленої системи є плата Arduino Due, що виконує роль головного керуючого пристрою. До складу також входить Інтернет-модуль, що використовує SPI протокол і працює як HTTP сервер, а також GSM модуль SIM800L, що підключається по протоколу UART і передає повідомлення на заданий номер телефону. До прототипу підключені цифрові датчики руху та аналоговий датчик води. В якості зчитувачів використовуються модулі PN532, які мають можливість підключення по протоколам I2C, UART, SPI. Для забезпечення живлення плати використовується окремий перетворювач напруги. Для забезпечення нестандартного живлення GSM модуля (3.7 - 4.2В) використовується додатковий перетворювач напруги.

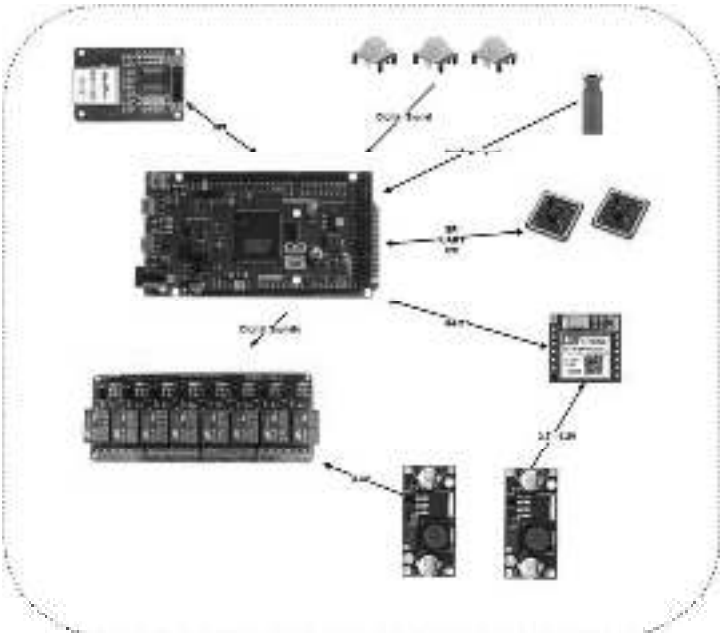

Рисунок 2.55 – Складові прототипу системи

Система може працювати в двох основних режимах. Звичайний режим - в цьому режимі система обробляє дані з периферійних пристроїв і на основі цих даних передає керуючі сигнали іншим пристроям. Режим сигналізації - після переходу в цей режим, система обробляє дані з датчиків і на разі виявлення руху в приміщенні передає GSM повідомлення про злам.

За результатами тестування прототипу системи, було прийнято рішення щодо розробки релізної версії пристрою для реалізації функцій системи Розумний офіс, що містить наступні складові:

• корпус пристрою;

• плата керування;

• комутаційна плата пристрою, яка містить всі необхідні компоненти та монтується в корпусі;

• набір необхідних датчиків та виконавчих пристроїв у монтажному вигляді.

• програмне забезпечення з можливістю поновлення.

Візуальне представлення концепції застосування проектованого пристрою наведено на рис. 2.56, модель складання на 2.57. Основними складовими пристрою є плата керування (Arduino) і комутаційна плата (рис.2.58), на якій розташовані необхідні модулі:

- модуль GSM SIM800L - для передачі інформаційних повідомлень за попередньо визначеними номерами;
- перетворювачі логичних рівнів Adafruit TXB0108 - виконують функцію перетворення напруги сигналів, отриманих з пристрою управління та їх передачу на модуль реле;
- перетворювачі напруги LM2596 - забезпечують живлення, необхідне для нормальної роботи модуля GSM і підключених датчиків.

Крім рознімів для розміщення модулів, комутаційна плата також містить клемні винтові розніми для підключення різних датчиків і зчитувачів і розніми для сигнальних шин з керуючого пристрою

Прототип системи був зібраний на макетній платі та пройшов тестування в реальних умовах (рис.2.59).

Схема електрична принципова (рис.2.60) містить необхідні з'єднання та виводи для комутаційного модуля, який забезпечить надійне з'єднання сигнальних та силових мереж для проведення випробувань. Посадкові розніми для модулей повинні відповідати розташуванню виводів на модулях. Клемні гвинтові розніми повинні бути розташовані вздовж контуру плати, відповідно до візуальної концепції плати. Лінії живлення 12В не повинні викликати перешкод в інформаційних лініях. Всі елементи мають бути розташовані з одного боку плати.

2.5 Система автоматизованого управління ролетами

Мета проекту – розробка системи автоматизованого управління ролетами з контролем положення ролети та руху її двигуна, реакцією на випадкові ситуації та мережевою взаємодією.

В результаті роботи створено програмно-апаратний комплекс, що дозволить дистанційно (по TCP/IP) приймати команди управління ролетою та отримувати її поточні стани. Для розробки системи використовувались наступні середовища розробки: Proteus, Arduino IDE, AltiumDesigner.

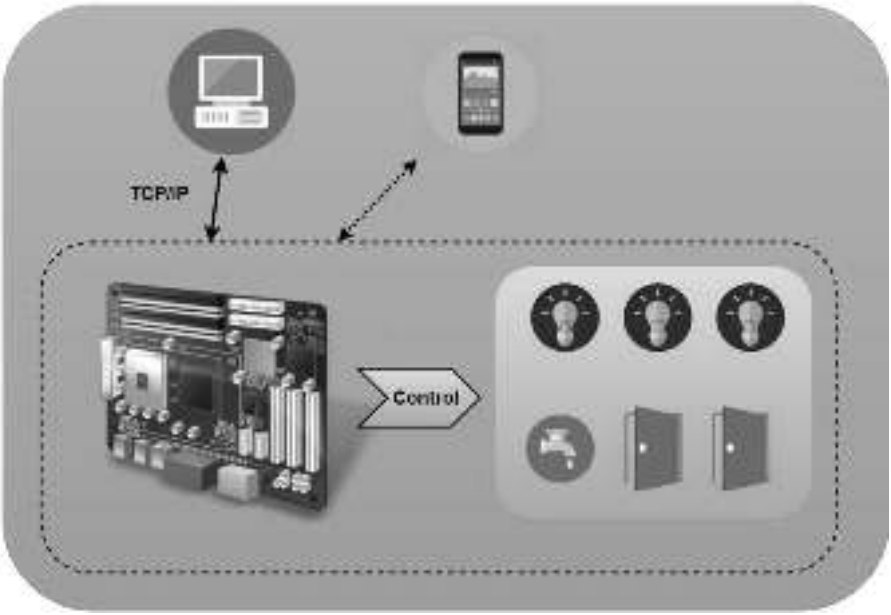

Рисунок 2.56 – Візуальне представлення концепції застосування проектованого пристрою

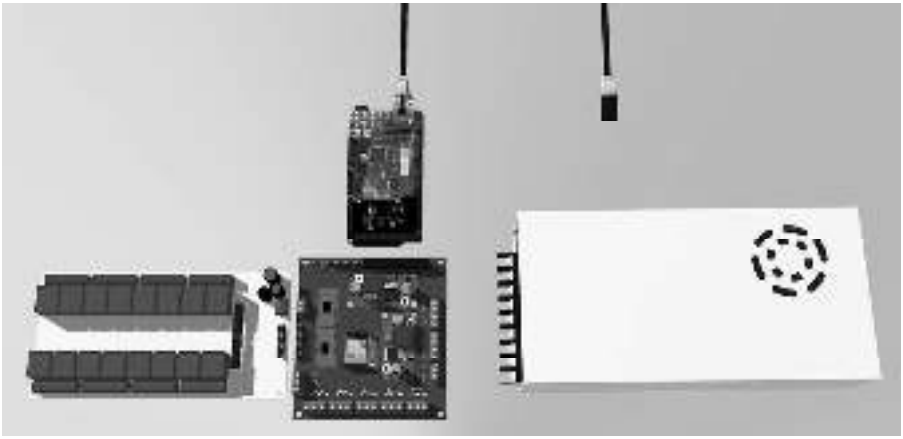

Рисунок 2.57 – Модель складання проектованого пристрою

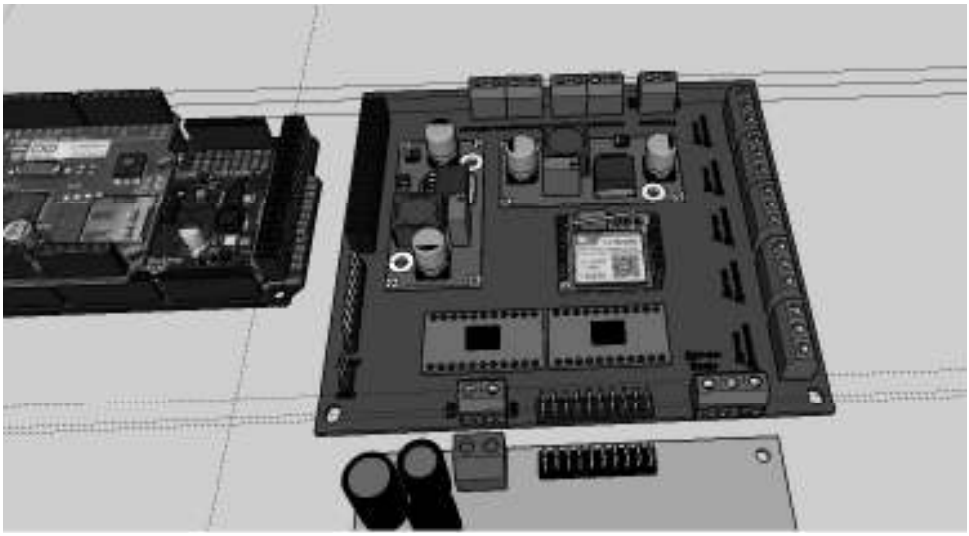

Рисунок 2.58 – Плата керування та комутаційна плата

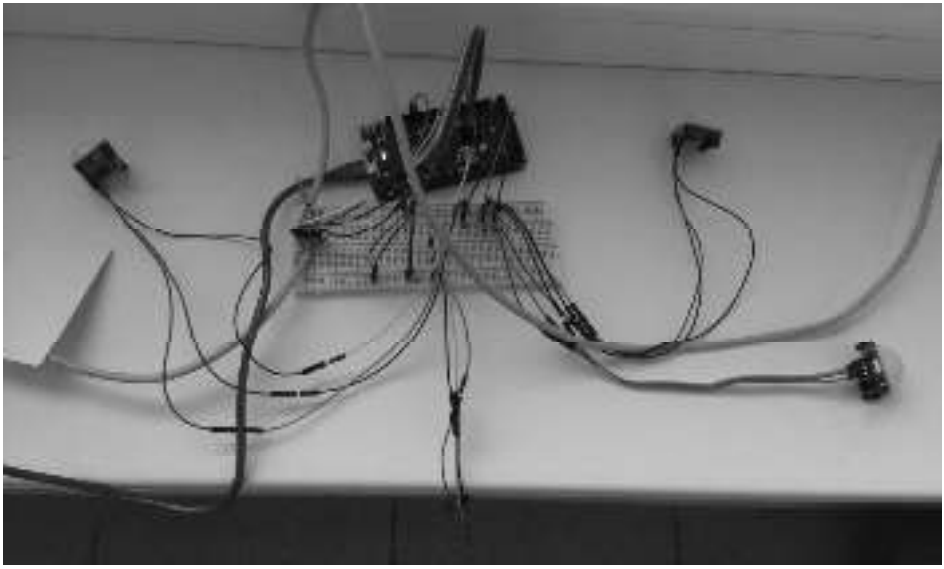

Рисунок 2.59 – Тестування прототипу системи

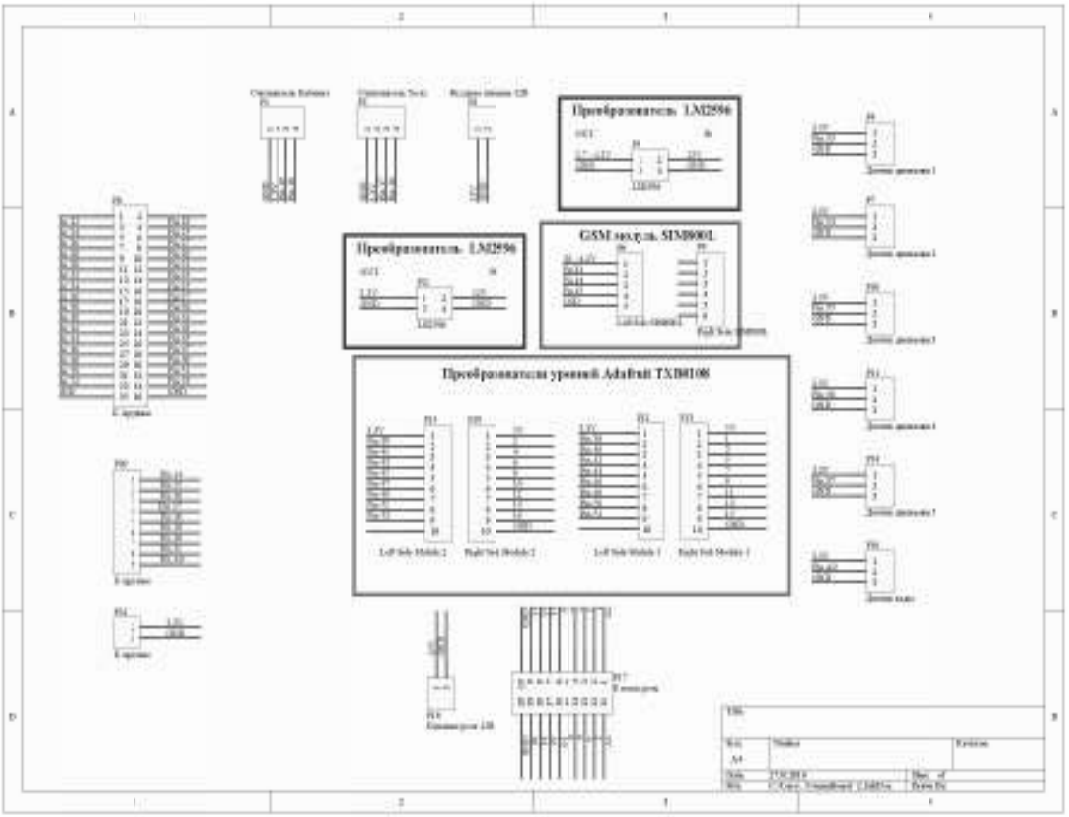

Рисунок 2.60 – Схема електрична принципова комутаційної плати

Система забезбечує обробку наступних виняткових систуацій:

- збій або знищення прошивки центрального мікроконтролера;
- збій або знищення прошивки мережевого мікроконтролера;
- недоступність даних з датчиків;
- відсутність зв'язку з зовнішніми пристроями керування;
- підрив ролети;
- заклинювання двигуна ролети або неповне закриття/відкриття ролети.

Архітектуру системи наведено на рис. 2.61. Для позначення різних типів зв’язків використано такі типи ліній: логічний (лінія зі стрілкою); фізичний (проста суцільна линия); TCP/IP & POE (пунктирна лінія).

Основою апаратної частини є модуль на основі двосторонньої друкованої плати з розташованими електрорадіоелементами та інтегральними мікросхемами, а також клемами та рознімом для підключення датчиків та перемикачів.

Призначення та функції основних складових системи:

• центральний мікроконтролер ATMEGA168PA (TQFP32) - дозволяє реалізувати основний алгоритм керування ролетою та забезпечує:

- передачу даних про стан ролети зовнішнім джерелам управління;
- зчитування та обробку станів датчиків, розташованих на ролеті та двигуні ролети;
- прийом та обробку керуючих команд від зовнішніх джерел управління;
- управління ролетами в ручному режимі за допомогою перемикача ручного управління;
- обробку та реакцію на виняткові ситуації;
- управління ролетами в автоматичному режимі;
- організацію захищеного доступу до управління ролетами.

• мережевий мікроконтролер ENC28J60 (SSOP28) – дозволяє реалізувати інтерфейс TCP/IP та забезпечує:

- підтримку мережі Ethernet;

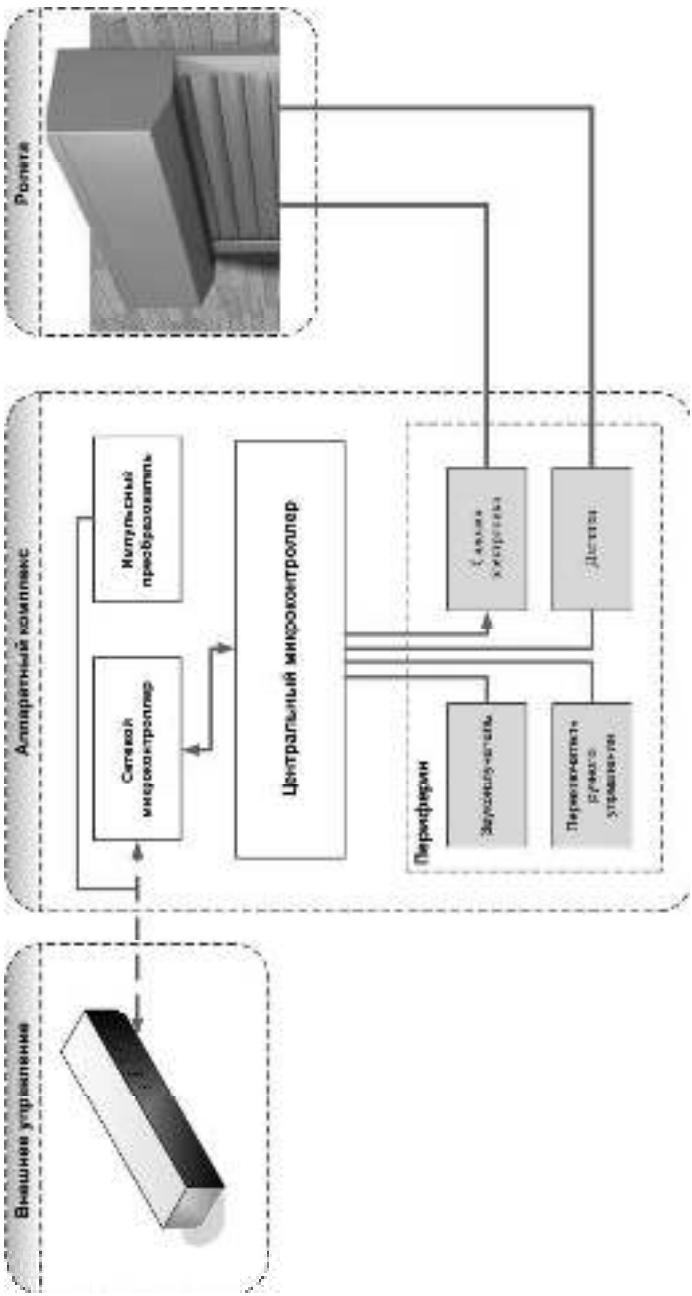

Рисунок 2.61 – Архітектура системи

- повторну передачу даних при помилці;
- фільтрування помилкових пакетів;
- підтримку мереж 10/100/1000 Base-T;
- взаємодію з центральним мікроконтролером за SPI інтерфейсом;
- спряження.

• імпульсний перетворювач 50В – 5В LM2596 (TO-220-5-TV) – дозволяє перетворити напругу в необхідну для електропостачання системи, використовує технологію POE;

• силова електроніка – реле електромеханічне – дозволяє реалізувати управління механічними пристроями ролети та забезпечує:

- захист опторозв'язкою центрального контролера від зворотньої напруги;
- комутацію двигуна ролети.

• датчики Холла, цифрові, уніполярні, SS41F - дозволяють реалізувати контроль стану ролети і електромеханічних пристроїв ролети та забезпечують:

- визначення положення ролети;
- визначення стану двигуна ролети.

• звуковипромінювач HCM1206BX, електромагнітний, генератор. 5 В, 30 мА, 85 дБ, 2.3 КГц - дозволяє реалізувати звукове сповіщення про виняткову ситуацію;

• перемикач ручного управління - дозволяє реалізувати управління ролетами (відкриття та закриття) в ручному режимі.

Програмна частина представлена розробленим програмним забезпеченням центрального мікроконтролера.

Прототип системи пройшов успішні випробування на реальному об'єкті (рис.2.62).

2.6 Автоматизована система управління вуликом

Система дозволяє користувачам збирати, обробляти, візуалізувати, зберігати інформацію про показники стану вулика (вагові та температурні) в реальному часі, та надає наступні функціональні можливості:

– зчитування інформації з датчиків температури та ваги;

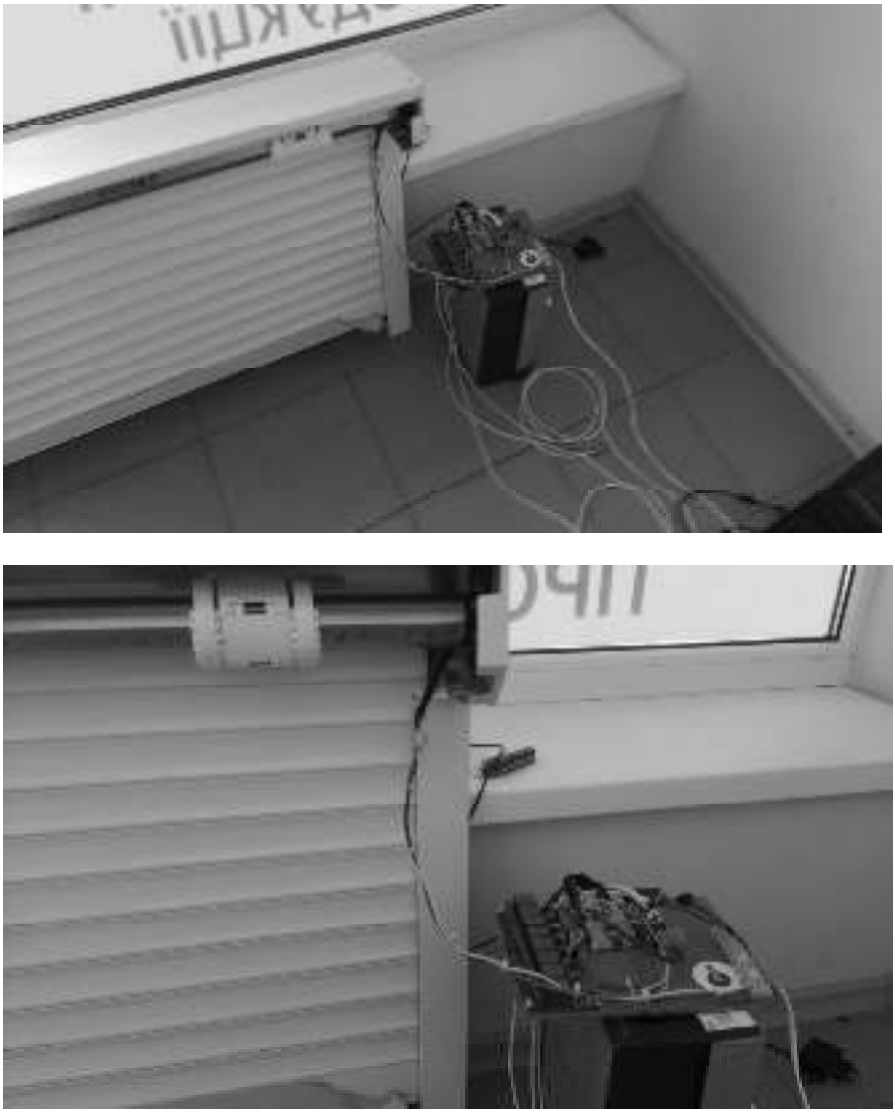

Рисунок 2.62 – Тестування прототипу системи

– відображення стану вулика у вигляді поточних значень та графіків;

– можливість представляти графіки у вигляді зображень;

– можливість зберігати значення відносно часу у базі даних;

– можливість доступу до результатів візуалізації стану через мережу Інтернет;

– можливість обнуління значення вагів;

– інтеграція апаратно-програмного продукту в систему домашньої автоматизації OpenHAB

До складу технічних засобів входить комп'ютер з доступом до мережі Інтернет, на якому запущена платформа OpenHAB, блок живлення 5В, чотири опорних датчики навантаження; сенсор температури; вимірюючий блок.

На рисунку 2.63 показана розроблена структура апаратного забезпечення системи. Основною складовою системи є набір з чотирьох тензодатчиків (рис.2.64) - тензорезисторів, що змінюють свій опір в залежності від сили (деформації), що була задіяна до них. Зміни опору вельми малі і вимагають прецизійних підсилювачів або підключення до аналогово-цифрового перетворювача (АЦП). Схема підключення наведена на рис. 2.20.

Аналоговий сигнал з датчиків проходить оцифрування за допомогою аналогово-цифропого перетворювача HX-711. Паралельно з цим збирається інформація температурним датчиком DS18B20 та передається мікроконтролеру на платі Arduino Nano, який є ініціатором збору всієї інформації та обробляє дані, що поступають й вираховує значення температури і ваги. Arduino Nano - це повнофункціональний мініатюрний пристрій на базі мікроконтролера ATmega328, адаптований для використання з макетними платами.

Після обробки інформації мікроконтролер використовує Ethernet-модуль ENC28J60 для передачі відповіді з даними про стан вулика на запит від хаба по протоколу ModBus TCP. Ethernet-модуль є інтерфейсом між платою Arduino Nano та роутером, який перенаправляє пакети запитів та відповідей на необхідні локальні адреси. Останнім пунктом у роботі підсистеми на апаратному рівні є сама система OpenHAB.

Схема підключення Ethernet-модуля наведена на рисунку 2.65.

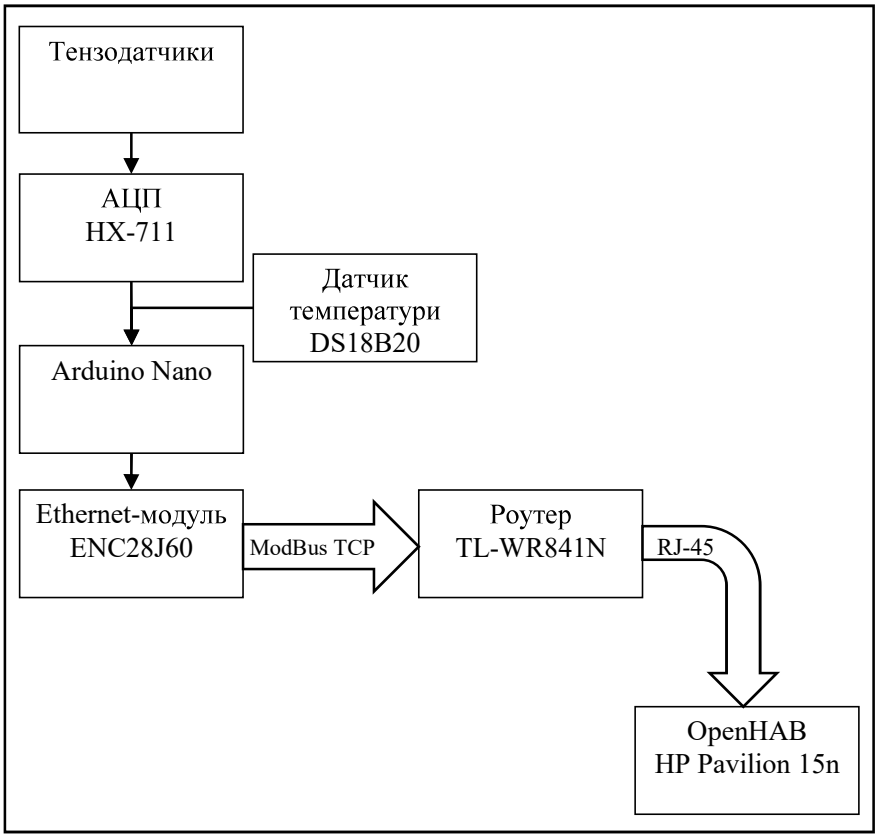

Рисунок 2.63 – Структура апаратного забезпечення

Рисунок 2.64 – Тензодатчик виявлення ваги CZL902

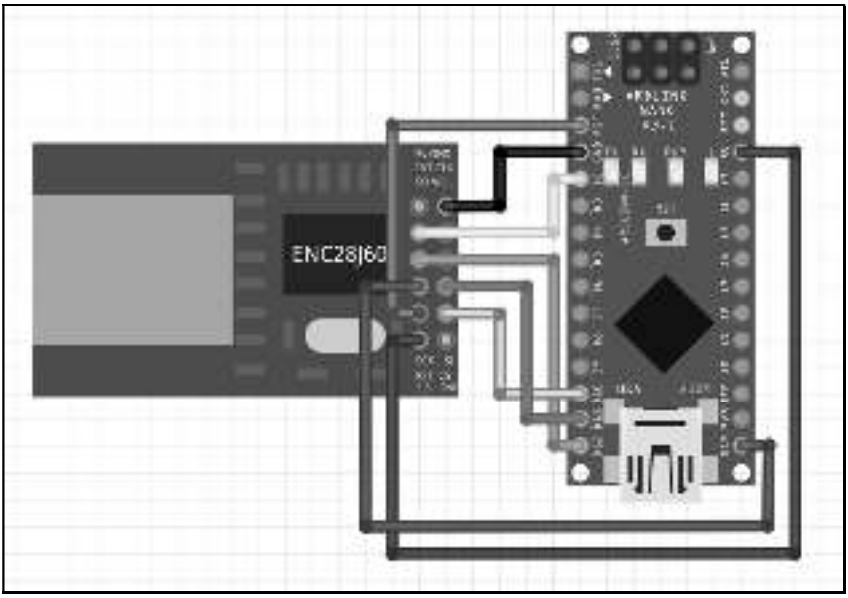

Рисунок 2.65 – Схема підключення Ethernet-модуля ENC28J60

Розроблене програмне забезпечення складається з двох частин: програми для мікроконтролера Arduino Nano та набору конфігураційних файлів.

Скетч Arduino являє собою програму, яка зчитує дані про стан вулика, зберігає їх та за запитом OpenHAB пересилає за допомогою протоколу Modbus.

Задачі системи автоматизації OpenHAB наступні:

– прийом відповіді та подальша обробка інформації про стан вулика на стороні клієнта;

– можливість зберігати значення відносно часу у базі даних;

– відображення стану вулика у вигляді поточних значень та графіків;

– можливість представляти графіки у вигляді зображень;

– можливість доступу до візуалізації стану через мережу Інтернет.

Прийом відповіді та подальша обробка інформації про стан вулика на стороні клієнта відбувається за допомогою біндингу Modbus.

Налаштування біндингу знаходяться в файлі openhab.cfg:

```
####################Modbus Binding ####################
#
# sets refresh interval to Modbus polling service.
# Value in milliseconds (optional, defaults to 200)
#modbus:<slave-type>.<slave-name>.<slave-parameter-name>=<slave-parameter-value>

modbus:poll=1500

# read Data from Arduino
modbus:tcp.slave1.connection=192.168.0.161:502
modbus:tcp.slave1.id=1
modbus:tcp.slave1.start=100
modbus:tcp.slave1.length=2
modbus:tcp.slave1.valuetype=int16
modbus:tcp.slave1.type=input
```

Ці записи означають наступне:

– опитування серверів відбувається кожні 1500 мілісекунд;

– з'єднання відбувається за адресою 192.168.0.161, при цьому використовується 502 порт;

– мікроконтролеру системи «Розумній вулик» присвоєно ідентифікаційний номер, який дорівнює 1;

– початок відліку відбувається з сотого реєстру;

– біндинг зчитує по 2 одиниці пам'яті з таблиці Input Registers у форматі 16 біт.

В файлі demo.items було підключено біндинг:

```
Number Temp_1_RAW "Temperature_RAW [%+.2f °C*100]" {modbus="slave1:0"}
Number Weight_1_RAW "Weigh_RAW [%.2f kg*100]" {modbus="slave1:1"}
```

Подальша обробка щойно прийнятих даних включає в себе ділення значень стану вагів та температурного сенсора на 100. Після цього відбувається оновлення цих значень в інтерфейсі користувача. Це було реалізовано завдяки створенню двох правил у файлі demo.rules:

```
rule "modbus Temp division"
when
        Item Temp_1_RAW received update
then
            var Number temp = (Temp_1_RAW.state as DecimalType)
            if(temp > 0x8000) {temp = temp - 0xFFFF } //to catch negative
    numbers
            temp = temp / 100
        postUpdate(Temp_1, temp)
end

rule "modbus Weight division"
when
        Item Weight_1_RAW received update
then
            var Number weight = (Weight_1_RAW.state as DecimalType)
            if(weight > 0x8000) {weight = weight - 0xFFFF } //to catch
    negative numbers
            weight = weight / 100
        postUpdate(Weight_1, weight)
end
```

Відображення стану вулика у вигляді поточних значень та графіків відбувається в інтерфейсі користувача (рис. 2.66, 2.67).

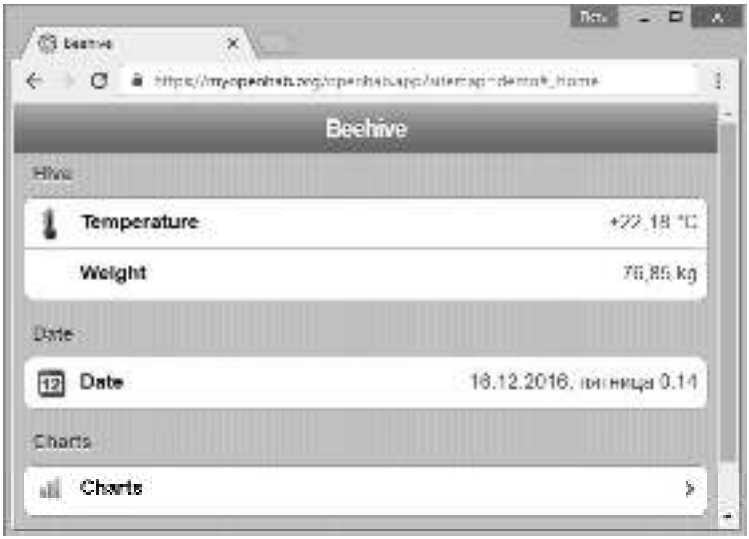

Рисунок 2.66 – Інтерфейс користувача

Для цього в файлі demo.sitemap була створена структура користувальницького інтерфейсу, яка описує що і в якому вигляді має бути представлено.

```
sitemap demo label="Beehive"
{
        Frame label="Hive" {
                Text item=Temp_1
                Text item=Weight_1
        }
        Frame label="Date" {
                Text item=Date
        }
         Frame label="Charts" {
            Text label="Charts" icon="chart" {
                        Chart item=Temp_1 period=d refresh=1000
                        Chart item=Weight_1 period=d refresh=1000
        }
        }
}
```

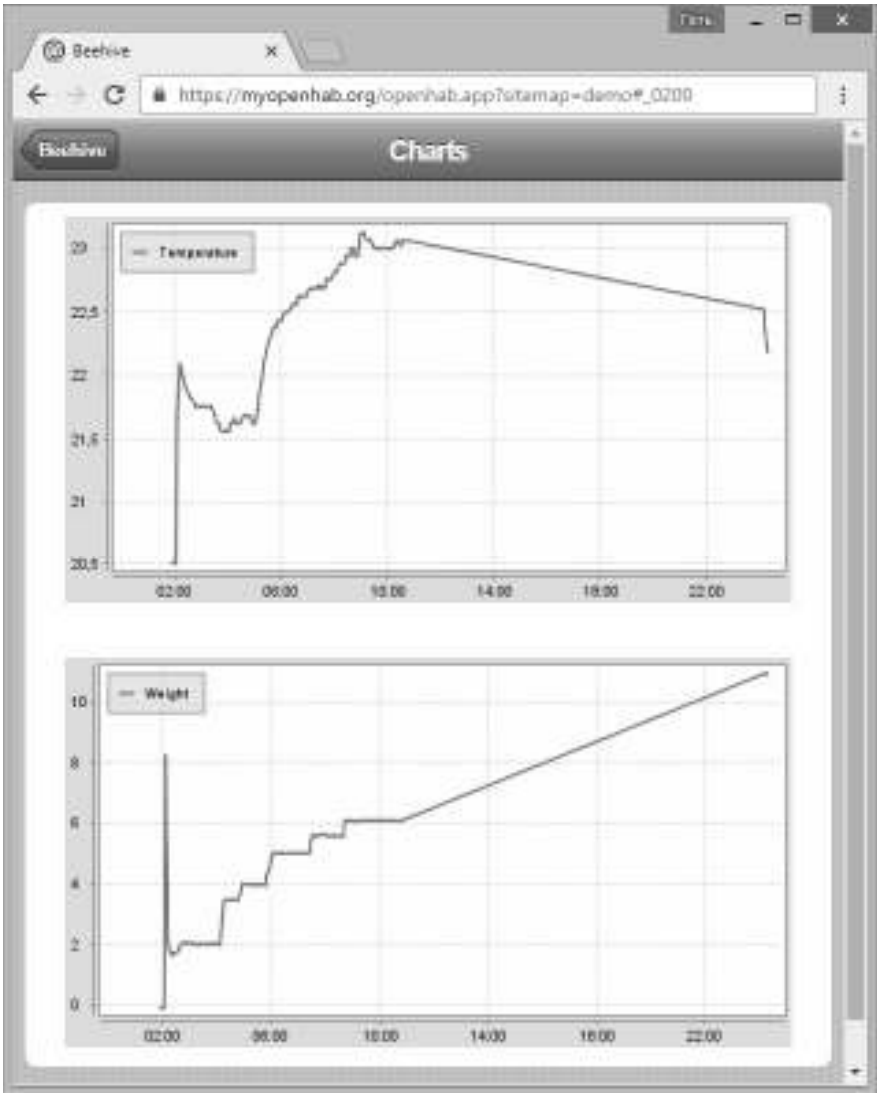

Рисунок 2.67 – Відображення графіків

В файлі demo.items визначені сутності предметів та їх формати.

```
Number Temp_1_RAW "Temperature_RAW [%+.2f °C*100]" {modbus="slave1:0"}
Number Temp_1 "Temperature [%+.2f °C]" <temperature>

Number Weight_1_RAW "Weigh_RAW [%.2f kg*100]" {modbus="slave1:1"}
Number Weight_1 "Weight [%.2f kg]"

DateTime Date "Date [%1$td.%1$tm.%1$tY, %1$tA %1$tk:%1$tM]" <calendar> {ntp="Europe/Kiev:en_EN"}
```

Після підключення біндинга openHAB Cloud Connector з'явилась можливість проводити моніторинг віддалено, через мережу Інтернет. Окрім підключення біндинга необхідно зареєструватися на сайті https://myopenhab.org. Форму реєстрації наведено на рисунку 2.68. При реєстрації заповнюються 4 поля: електронна адреса, пароль, OpenHAB UUID, OpenHAB Secret.

Рисунок 2.68 – Процес реєстрації

Авторизація відбувається завдяки останнім двом параметрам. Вони знаходяться в папці webapps/static. Кожного разу, коли користувач буде заходити з нового комп'ютера, йому доведеться ввести адресу sitemap через хостинг myopenhab.org: https://myopenhab.org/openhab.app?sitemap=demo.

Також у проекті була реалізована можливість зберігати значення відносно часу у базі даних. Для цього використовується сервіс зберігання стану rrd4j.

Для зберігання даних використовується циклічна база даних (ЦБД). Це означає, що обсяг інформації, що зберігається, не змінюється із часом, оскільки кількість записів постійна, а в процесі зберігання даних вони використовуються циклічно. Завдяки цьому вдається виконати вимогу щодо компактності програмного забезпечення.

Особливість цього сервісу полягає в тому, що він не може бути запитаним напряму. Розробники цю особливість не коментують, вказавши в документації, що не можуть дати відповіді на це питання.

Для встановлення цього пакету зберігання стану був встановлений біндинг, файл інсталяції мав назву org.openhab.persistence.rrd4j-1.8.3.jar.

Налаштування сервісу відбувається в двох файлах:

– ${openhab_home}/configurations/persistence/rrd4j.persist;

– ${openhab_home}/configurations/openhab.cfg.

rrd4j.persist – включає в себе таймери і стратегії для предметів. Для роботи rrd4j має використовуватись стратегія everyMinute, інакше дані не будуть зберігатися.

openhab.cfg – визначає яким чином вони будуть зберігатися в ЦБД.

ЦБД мають, так звані, «архіви» фіксованої довжини для зберігання значень. Одна ЦБД може мати (в цілому) декілька джерел даних, і кожне з цих джерел може мати декілька архівів. OpenHAB підтримує тільки одне джерело даних на кожну ЦБД, яке називається DATASOURCE_STATE.

Джерела даних бувають наступних типів:

– COUNTER – являє собою значення, що постійно збільшується;

– GAUGE – представляє значення, наприклад, показників з датчика температури;

– ABSOLUTE – майже теж саме, що і COUNTER, але враховує тільки різницю між старим та новим значеннями;

– DERIVE – схожий на лічильник, але може віднімати, отже мати від'ємні значення.

При створенні умов був написаний наступний фрагмент коду в файлі, що має назву rrd4j.persist:

```
Strategies {
  // for rrd charts, we need a cron strategy
  everyMinute : "0 * * * * ?"
}

Items {
  Temp_1,Weight_1 : strategy = everyMinute, restoreOnStartup

}
```

Однією з унікальних особливостей є те, що графіки тільки пакету rrd4j можуть бути представлені у вигляді зображень. Для того, щоб звернутися до регістрів пакету rrd4j треба зробити запит за звичайним посиланням, яке має наступний вигляд: http://<server>:<port>/rrdchart.png

Також є можливість через HTTP запит отримати більше параметрів при створенні графіка. Для цього використовуються наступні параметри:

– w: ширина зображення в пікселях (за замовчуванням 480);

– h: висота зображення в пікселях (за замовчуванням 240);

– period: часовий інтервал для вісі X; значення може бути: h,4h,8h,12h,D,3D,W,2W,M,2M,4M,Y (за замовчуванням "D", що означає відображення останніх 24 годин);

– items: перерахований через кому список назв предметів, що мають показуватися; обов'язковий, якщо не надається параметр "groups";

– groups: перерахований через кому список назв груп, члени яких мають бути показані; обов'язковий, якщо не надається параметр "items".

Працюючий запит та його результат наведено на рисунку 2.69:

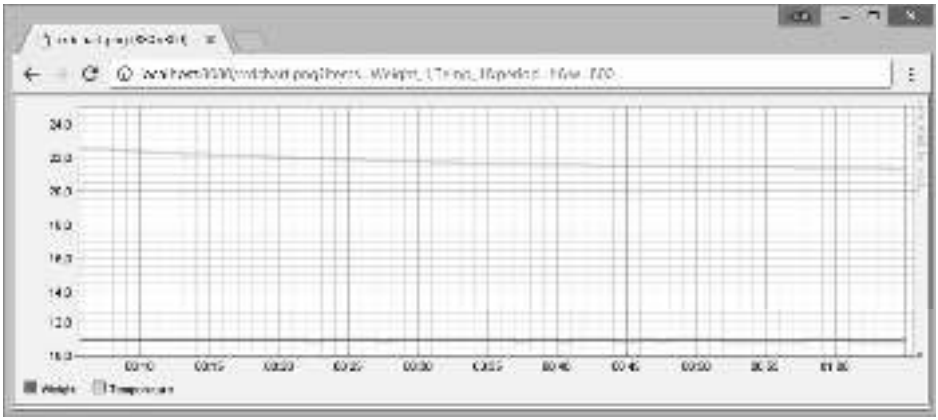

Рисунок 2.69 – HTTP запит графіка

2.7 Реалізація IoT проекту на платформі Thingworx

Платформа ThingWorx IoT від компанії PTC скорочує час, вартість і ризики побудови інноваційної технології «машина-машина» (M2M) і додатків Інтернету речей.

ThingWorx перша програмна платформа призначена для реалізації IoT технологій, розробки та впровадження інтелектуальних додаткі і використання пов'язаних з ними продуктів. Компанія запевняє, що ви будете економити час, гроші і зменшите свої ризики при розробці додатків Інтернету речей.

Платформа Thingworx поєднує у собі майже усі сучасні IoT технології та забезпечує:

– низькі витрати обчислювальних ресурсів;
– швидку передачу даних з пристрою до хмари;
– приватність даних;
– доступ до даних та обчислювальних ресурсів через Інтернет;
– розробку та збереження розроблених додатків у хмарі.

ThingWorx встановлює нові стандарти в IoT масштабованості, безпеці і платформі розширення.

Дві осовні складові платформи ThingWorx – сервер ThingWorx і компоненти ThingWorx Edge, включаючи Edge MicroServer (EMS) та різні комплекти для розробників програмного забезпечення (SDK).

Сервер здійснює аутентифікацію користувача та пристрою, виступає посередником в обміні даними між системами, людьми та речами в ландшафті рішення, а також, за необхідністю, управляє перетворенням даних, їх збереженням та бізнес-логікою в додатках для кінцевого користувача. EMS дозволяє пристроям безпечно обмінюватися даними з сервером ThingWorx і бути повноправною частиною ландшафту рішення. EMS не просто з'єднує дані, але і забезпечує їх попередню інтелектуальну обробку.

При роботі з ThingWorx виробник отримує можливість використовувати найкращі практики. Передусім, платформа дозволяє провести оцінку ризиків - спочатку базових, а потім і всіх інших. При цьому процедура оцінки повторюється на всіх етапах - від проектування до експлуатації. Також ThingWorx пропонує виробнику визначити вимоги безпеки, щоб гарантувати високий рівень безпеки готової продукції та виконання всіх вимог замовника до її захисту.

IoT платформа забезпечує повний дизайн додатків, їх виконання і розробку у «інтелектуальному» середовищі з наступними інноваційними функціями та можливостями:

– швидкість розробки - до10 разів швидше, на основі гнучкої моделі – моделі, яка вам подобається;

– масштабованість – можливість розвиватися і розробляти ваш додаток протягом певного часу;

– сучасна і повноцінна платформа;

– Mashup люди, системи та машини;

– Mashup Builder дає вам можливість швидко створювати функціональні інтерактивні додатки IoT в режимі реального часу. Це конструктор додатків наступного покоління, що скорочує час розробки;

– широкий спектр застосування розроблених додатків.

Опис платформи доступний на http://www.thingworx.com/iot-platform/. Приклади розробки проектів з використанням платформи Thingworx можливо знайти за посиланням: http://www.thingworx.com/academics/example-iot-projects.

Розглянемо приклад реалізації простої програми за допомогою на платформі Thingworx.

Передусім, потрібно зареєструватися, щоб отримати доступ до платформи Thingworx на офіційному сайті компанії PTC.

Для того, щоб зайти у середовище розробки Composer, треба відкрити браузер та перейти за посиланням: http://zntu.cloud.thingworx.com/Thingworx/Composer/. Перед користувачем відкриється вікно авторизації (рис. 2.70).

Після авторизації, відкриється діалогове вікно з нещодавно відкритими/створеними проектами (рис. 2.71). Ми можемо відкрити ці проекти знову, або закрити діалогове вікно і перейти до головного меню середовища.

Розробимо програму, яка буде контролювати температуру та вологість повітря у приміщенні за допомогою IoT технологій.

По-перше треба створити «Інтернет річ» (рис.2.72):

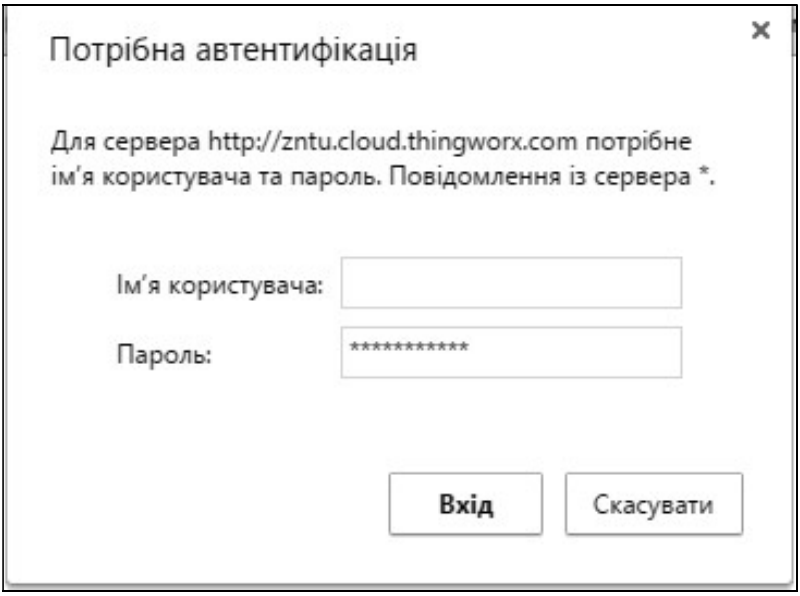

Рисунок 2.70 – Вікно авторизації доступу до середовища розробки Composer.

Рисунок 2.71 – Діалогове вікно з нещодавно відкритими проектами.

Рисунок 2.72 – Створення Інтернет-речі

– задаємо ім’я «речі»;
– указуємо річ шаблону – “GenericThing”;
– за бажанням можна указати «теги» та зробити опис «речі»;
– зберігаємо створену «річ».

Другий крок:

– переходимо до вкладки “Properties”, натискаємо клавішу редагування та створюємо нову властивість (рис. 2.73, 2.74). При створенні вказуємо назву та тип властивості;

– зберігаємо створені властивості.

Рисунок 2.73 – Створюємо властивість. Крок 1.

Третій крок:

– переходимо на домашню сторінку розробки та створюємо «mashup» (колаж/шаблон/шар) (рис.2.75). При створенні вказуємо тип «page» і шар «responsive»;

– у вкладці “info” вказуємо ім’я mashup (рис.2.76) й зберігаємо проект;

– відкриваємо створений mashup для редагування;

– робимо розмітку робочого місця за допомогою віджета “layout” (рис. 2.77, 2.78);

– додаємо до робочого місця віджет Gaudge, задаємо ім’я, стиль та шкалу;

– додаємо до робочого місця панель;

– додаємо до робочого місця віджет “Numeric Entry”. Вказуємо властивості;

– додаємо кнопку, вказуємо її ім’я;

Рисунок 2.74 – Створюємо властивість. Крок 2.

Рисунок – 2.75 – Створення mashup. Крок 1.

Рисунок – 2.76 – Створення mashup. Крок 2.

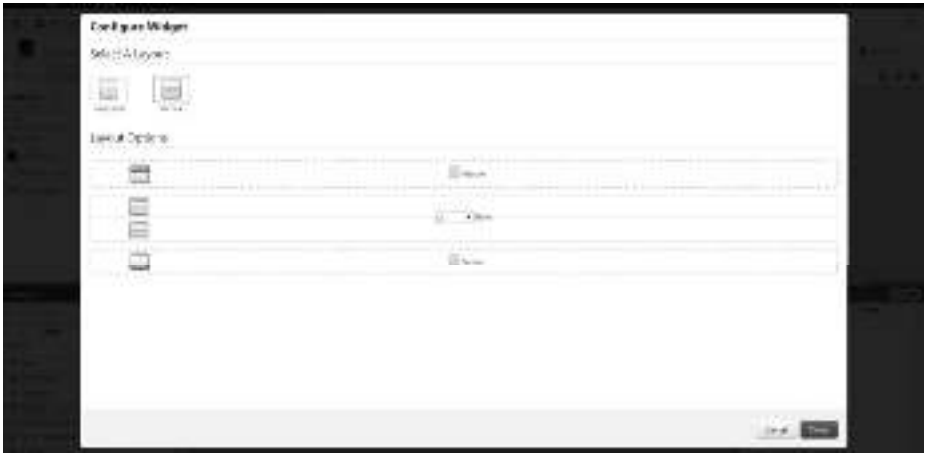

Рисунок 2.77 – Вибір конфігурації шару

– задаємо властивості елементам/додаємо раніше створену річ (рис. 2.79 - 2.82);

– зберігаємо та запускаємо проект (рис. 2.83).

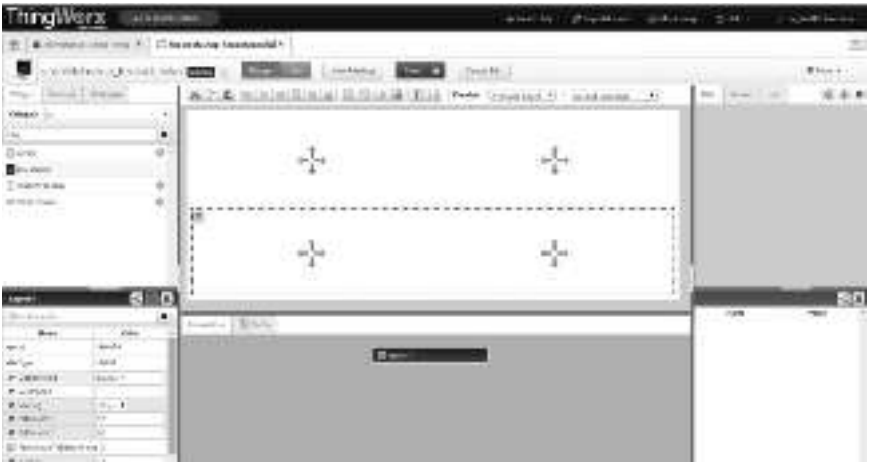

Рисунок 2.78 – Розмітка робочого місця

Рисунок 2.79 – Створення дизайну

Рисунок 2.80 – Додавання речі до проекту

Рисунок 2.81 – Додавання властивостей

Рисунок 2.82 – Зв’язки між компонентами

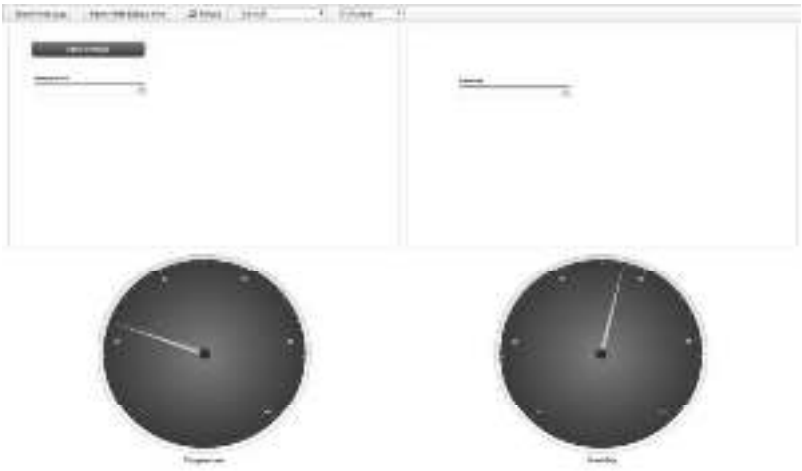

Рисунок – 2.83 –Інтерфейс програми

Розглянемо процес розробки програми для Android-пристрою. Використаємо один з прикладів Thingworx, який знаходиться за посиланням: http://www.thingworx.com/Academics/Example-IoT-Projects/Getting-Started-With-Android-Smartphones-And-ThingWorx.

Програма зчитує дані вільної RAM телефона, завантаженість процесору та геопозицію пристрою і відправляє дані до хмари.

По-перше:

– зберігаємо файли з сайту до свого смартфону;

– розархівуємо та встановимо додаток «APKGettingStartedWithThingWorx»;

– запускаємо щойно встановлений додаток (рис. 2.84).

Другий крок:

– заходимо до середовища проектування Composer та завантажуємо файл з розширенням “xml”(рис. 2.85);

– запускаємо щойно завантажений файл.

– водимо імей який вказаний в програмі на телефоні (рис. 2.86);

– на телефоні у запущеному додатку указуємо IP адресу хмари – 52.21.241.75 та ключ, який береться з вікна зчитування даних, та відправляємо дані через Інтернет(рис.2.87);

– переглядаємо результати у нашому машапі(рис.2.88).

Рисунок 2.84 – Запущений додаток

Рисунок 2.85 – Завантаження mashup

Рисунок 2.86 – Вікно зчитування даних

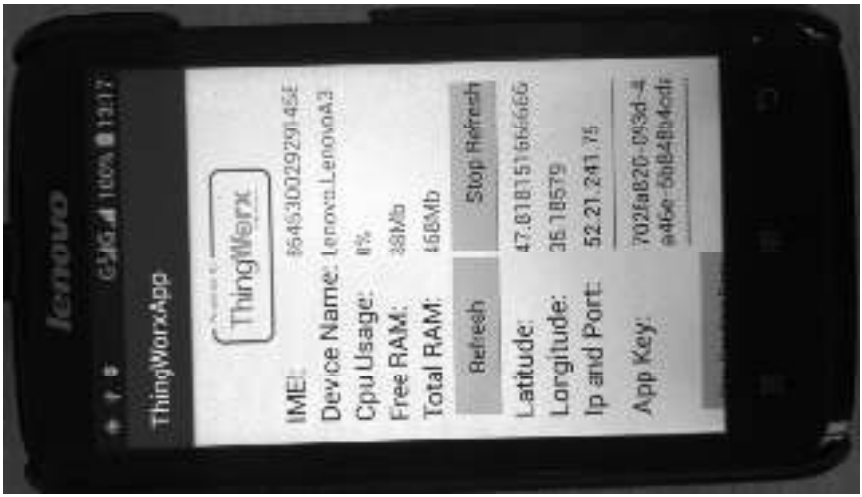

Рисунок 2.87 – Встановлення з’єднання з хмарою та відправка даних

Рисунок 2.88 – Результат роботи програми.

ПИТАННЯ ДЛЯ САМОПЕРЕВІРКИ

1. Що таке Інтернет речей?
2. Чому вбудовані системи можуть вважатися основою інфраструктури Інтернету речей?
3. Які технології можуть використовуватися для з'єднання вбудованих систем з іншими пристроями або з мережею Інтернет?
4. Назвіть типові підсистеми системи Розумний будинок.
5. Охарактеризуйте загальну структуру віддаленого лабораторного комплексу REIoT.
6. На яких програмних та апаратних платформах побудовано REIoT комплекс?
7. Які віддалені експерименти дозволяє виконати віддалена лабораторія RELDES?
8. Які віддалені експерименти дозволяє виконати віддалена лабораторія Smart House & IoT?
9. Назвіть особливості та переваги платформи Arduino. Які саме версії Arduino використано в REIoT комплексі?
10. Чому платформу Raspberry Pi можна назвати мінікомп'ютером?
11. Яким чином в лабораторії Smart House & IoT здійснено зв'язок Raspberry Pi та Arduino?

12. Для чого в програмах Arduino в лабораторії Smart House & IoT використано бібліотеку Modbus Master-Slave?
13. З якою метою в REIoT комплексі використовується OpenHAB REST API?
14. Яким чином реалізовано відео-трансляцію експериментів у REIoT комплексі?
15. Що таке шина подій OpenHAB?
16. Для чого використовується репозиторій елементів OpenHAB?
17. Що таке сценарій OpenHAB?
18. Де розташовуються сценарії OpenHAB та яким чином викликаються?
19. З якою метою створюються правила OpenHAB та яку структуру має файл правила?
20. Що таке тригер OpenHAB?
21. Як налаштувати календар подій в OpenHAB?
22. Як створити імітацію присутності в приміщенні засобами OpenHAB?
23. Які типи дій надає OpenHAB?
24. Назвіть основні етапи встановлення та налаштування OpenHAB.
25. Яким чином відбувається підключення до Telegram у OpenHAB?
26. Яким чином відбувається підключення до my.openhab у OpenHAB?
27. Які команди дозволяє приймати прив'язка MailControl у OpenHAB?
28. Розкрийте фізичні принципи функціонування різноманітних датчиків, що можуть використовуватися для створення систем типу Розумний будинок.
29. Розкрийте фізичні принципи функціонування різноманітних виконавчих механізмів, що можуть використовуватися для створення систем типу Розумний будинок.
30. Призначення GSM-модуля, WiFi- модуля, Ethernet- модуля.

АЛФАВІТНО-ПРЕДМЕТНИЙ ПОКАЖЧИК

Навчальне видання

Пархоменко Анжеліка Володимирівна,

Туленков Артем Вячеславович,

Соколянський Олександр Володимирович,

Залюбовський Ярослав Ігорович,

Пархоменко Андрій Валентинович

ПРОГРАМНО-АПАРАТНА ПЛАТФОРМА ДЛЯ НАВЧАННЯ ТЕХНОЛОГІЯМ ІНТЕРНЕТУ РЕЧЕЙ

Навчальний посібник

Технічний редактор Л. А. Рябоконь
Коректор Н. В. Чечеко
Художник А. П. Кондаков

Формат 60x84/16.
Папір офсетний. Гарнітура Times. Друк офсетний.
Підписано до друку 28.03.2017. Ум. друк. арк. 12,78.
Наклад 300 прим.

Видавництво «Дике Поле»
Україна, 69063, м. Запоріжжя, вул. Троїцька, 31-А.
Тел.: (061) 213-75-95; 213-75-05.
Свідоцтво суб'єкта видавничої справи ЗЗ № 004 від 23.08.2001 р.

Електронна адреса dikoepole.zp@gmail.com
Веб-сторінка http://www.dikoepole.zp.ua

Правове забезпечення
Юридична фірма «Щедрин і партнери»

Електронна адреса schedrin1959@gmail.com