

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
РОЗРОБКИ ГРАФІЧНОГО ІНТЕРФЕЙСУ ДЛЯ ARDUINO.
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS
FOR DEVELOPING A GRAPHICAL INTERFACE FOR ARDUINO.

Виконав: студент 2 курсу, групи КНТ-120 м
Спеціальності _____

121 Інженерія програмного забезпечення
(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Павленко В.М.

(прізвище та ініціали)

Керівник Пархоменко А.В.

(прізвище та ініціали)

Рецензент Зеленьова І.Я.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет ІРЕ, ФКНТ
 Кафедра програмних засобів
 Ступінь вищої освіти магістр
 Спеціальність 121 Інженерія програмного забезпечення
 (код і найменування)
 Освітня програма (спеціалізація) Інженерія програмного забезпечення
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
 “ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Павленка Владислава Михайловича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів розробки графічного інтерфейсу для Arduino. Research and Software Implementation of Methods for Developing a Graphical Interface for Arduino.

керівник проєкту (роботи) Пархоменко Анжеліка Володимирівна, к.т.н., доцент,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “5” листопада 2021 року № 435

2. Строк подання студентом проєкту (роботи) 10 грудня 2021 року

3. Вихідні дані до проєкту (роботи) рекомендована література, завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Дослідження особливостей реалізації сучасних систем домашньої автоматизації. 2. Аналіз методів та засобів прототипування СДА. 3. Розробка та дослідження програмного та апаратного забезпечення прототипу СДА

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-3 Основна частина	Пархоменко А.В., доцент		
Нормоконтролер	Дейнега Л.Ю., ст.викладач		

7. Дата видачі завдання “02” вересня 2021 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	2–3 тижні	Розділ 1
3	Дослідження та удосконалення методів, моделей та алгоритмів вирішення задачі.	4–5 тижні	Розділ 2
4	Розробка архітектури системи.	6 тиждень	Розділ 3
5	Розробка програмного забезпечення.	7–8 тижні	Розділ 3
6	Тестування та експериментальне дослідження програмного забезпечення.	9-10 тиждень	Розділ 3
7	Оформлення пояснювальної записки та документів до неї.	11-13 тижні	Додатки
8	Нормоконтроль та рецензування.	14 тиждень	
9	Захист роботи.	15 тиждень	

Студент

(підпис) Павленко В. М.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Пархоменко А.В.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
124 с., 69 рис., 17 табл., 2 дод., 71 джерел.

ІНТЕРНЕТ РЕЧЕЙ, СИСТЕМА ДОМАШНЬОЇ АВТОМАТИЗАЦІЇ,
БЕЗДРОТОВІ ТЕХНОЛОГІЇ, ХМАРНІ ТЕХНОЛОГІЇ, ПРОТОТИП,
МЕТОДОЛОГІЯ ПОВТОРНОГО ВИКОРИСТАННЯ, ГРАФІЧНИЙ
ІНТЕРФЕЙС, ARDUINO.

Об'єкт дослідження – процес домашньої автоматизації.

Предмет дослідження – методи та засоби розробки прототипів систем домашньої автоматизації (СДА).

Мета роботи – дослідження та розробка прототипу СДА на основі методології повторного використання та хмарних технологій для підвищення ефективності взаємодії компонентів СДА.

Матеріали, методи та технічні засоби: інтегровані середовища розробки Arduino IDE та Processing, мови програмування Wiring та Processing, бібліотека Firmata, хмарна платформа Microsoft Azure IoT.

Результати. Розроблено програмне забезпечення для організації графічного інтерфейсу між Arduino та сенсорами й актуаторами підсистем інтелектуального освітлення, контролю доступу та віддаленого моніторингу прототипу СДА.

Висновки. Розроблений прототип дозволяє виконати дослідження структурних та функціональних особливостей СДА на основі впровадження хмарних технологій та готових програмних і апаратних рішень.

Галузь використання – системи домашньої та офісної автоматизації.

ABSTRACT

Explanatory note to the master thesis: 124 pages, 69 figures, 17 tables, 2 appendixes, 71 sources.

INTERNET OF THINGS, HOME AUTOMATION SYSTEM, WIRELESS TECHNOLOGIES, CLOUD TECHNOLOGIES, PROTOTYPE, REUSE METHODOLOGY, GRAPHIC INTERFACE, ARDUINO.

The object of study is the home automation process.

The subject of the study are methods and tools for developing prototypes of home automation systems (HAS).

The goals of the master thesis are research and practical implementation of the HAS prototype based on reuse methodology and cloud technologies to increase the efficiency of user interaction with HAS subsystems.

Materials, methods, and tools: integrated development environments Arduino IDE and Processing, Wiring and Processing programming languages, Firmata library, cloud platform Microsoft Azure IoT.

Results. Software has been developed to organize the graphical interface between the Arduino and the sensors and actuators of the intelligent lighting subsystem, access control subsystem and remote monitoring subsystem of the HAS prototype.

Conclusions. The developed prototype allows to carry out research of structural and functional features of HAS on the basis of cloud technologies and ready software and hardware decisions application.

The fields of application are home and office automation systems.

ЗМІСТ

Перелік скорочень та умовних позначок.....	8
Вступ.....	9
1 Дослідження особливостей реалізації сучасних систем домашньої автоматизації.....	10
1.1 Аналіз концепції СДА.....	10
1.2 Вимоги до реалізації концепції СДА.....	15
1.3 Огляд тенденцій ринку СДА.....	16
1.4 Дослідження дротових та бездротових технологій для СДА.....	18
1.5 Дослідження архітектури СДА, інтегрованої в хмару.....	24
1.6 Постановка задач дипломної кваліфікаційної роботи.....	31
2 Аналіз методів та засобів прототипування СДА.....	33
2.1 Методи розробки прототипів СДА.....	33
2.2 Аналіз структурних та функціональних особливостей апаратно- програмної платформи Arduino.....	44
2.3 Огляд засобів розробки програмного забезпечення для Arduino	51
2.4 Дослідження патернів розробки графічного інтерфейсу Processing/Arduino.....	59
2.5 Розробка графічного інтерфейсу взаємодії між Arduino та Processing для СДА.....	63
2.6 Висновки до розділу 2.....	70
3 Розробка та дослідження програмного та апаратного забезпечення прототипу СДА.....	71
3.1 Аналіз вимог до прототипу СДА.....	71
3.2 Хмарна архітектура системи.....	73
3.3 Опис апаратних рішень.....	74
3.4 Схемотехнічне проектування та моделювання прототипу СДА.....	77
3.5 Розробка програмного забезпечення прототипу на основі методології повторного використання.....	81

3.6 Тестування прототипу СДА.....	85
3.5 Висновки до розділу 3.....	96
Висновки.....	97
Перелік джерел посилання.....	98
Додаток А Текст програми.....	105
Додаток Б Слайди презентації.....	116

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

HMI	– Human-Machine Interface;
IDE	– Integrated Development Environment;
IoT	– Internet of Things;
ISO	– International Organization for Standardization;
M2M	– Machine-to-Machine;
OSI	– Open Systems Interconnection;
SaaS	– Software as a Service;
SOAP	– Simple Object Access Protocol;
TSI	– Time Series Insights;
ОС	– операційна система;
ПЗ	– програмне забезпечення;
ПК	– персональний комп'ютер;
ПЛК	– промисловий логічний контролер;
СДА	– система домашньої автоматизації;
ШІМ	– широтно-імпульсна модуляція.

ВСТУП

Індустрія 4.0, або Четверта промислова революція – це безперервна автоматизація традиційних побутових та промислових практик із застосуванням сучасних інноваційних технологій. Комунікації між машинами (Machine-to-Machine, M2M) та Інтернет речей (Internet of Things, IoT) інтегровані для підвищення автоматизації, покращення зв'язку та виробництва розумних пристроїв. Цими пристроями можна керувати та використовувати їх для моніторингу, аналізу та діагностики проблем практично без втручання людини [1]. Індустрія 4.0, фактично, є рухом до цифровізації разом з такими технологіями, як штучний інтелект, робототехніка, блокчейн, 3D-друк, квантові обчислення та IoT. В свою чергу IoT описує вбудовані фізичні об'єкти (або групи таких об'єктів), з датчиками, можливостями обробки, програмним забезпеченням та іншими технологіями, які підключаються та обмінюються даними з іншими пристроями та системами через Інтернет чи інші мережі зв'язку [2].

Всього за пару років у домашньому середовищі швидко запровадили інтелектуальні цифрові технології з мережевою підтримкою. Одним з актуальних напрямів розвитку систем домашньої автоматизації (СДА) є проєктування та впровадження хмарної архітектури, частково побудованої на основі технології з відкритим кодом, а також широко доступних ресурсів. Зокрема, для реалізації СДА рекомендується використання мінікомп'ютера Raspberry Pi як основного контролера, а також програмно-апаратних платформ Arduino в якості вторинних контролерів для керування приладами та віддаленого моніторингу їх параметрів. Тому, тема магістерської роботи є актуальною.

Метою роботи є дослідження та розробка прототипу СДА на основі методології повторного використання та хмарних технологій для підвищення ефективності взаємодії компонентів СДА.

1 ДОСЛІДЖЕННЯ ОСОБЛИВОСТЕЙ РЕАЛІЗАЦІЇ СУЧАСНИХ СИСТЕМ ДОМАШНЬОЇ АВТОМАТИЗАЦІЇ

1.1 Аналіз концепції СДА

Розумний будинок або система домашньої автоматизації (СДА) – житло сучасного типу, організоване для проживання людей за допомогою високотехнологічних пристроїв. Під СДА слід розуміти систему, яка повинна вміти розпізнавати конкретні ситуації, що відбуваються в будівлі, та відповідним чином на них реагувати: одна з підсистем може управляти поведінкою інших за заздалегідь виробленими сценаріями [3].

Основною особливістю СДА є об'єднання окремих підсистем в єдиний керований комплекс (рис.1.1) [4].

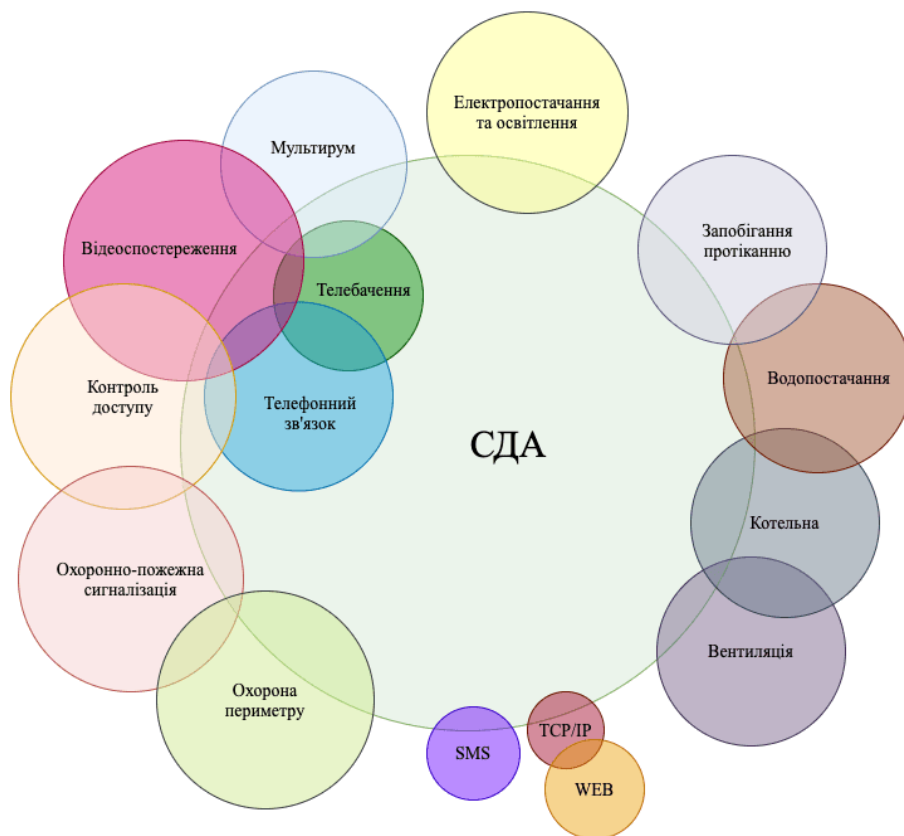


Рисунок 1.1 – Інтеграція підсистем в СДА

Важливою особливістю СДА є те, що це найбільш прогресивна концепція взаємодії людини з житловим простором, коли людина однією

командою задає бажану обстановку, а вже автоматика відповідно до зовнішніх і внутрішніх умов задає та відстежує режими роботи всіх інженерних систем і електроприладів. Достатньо вибрати один з сценаріїв і СДА налаштує роботу всіх підсистем відповідно до побажань користувача, часу доби, погоди, зовнішньої освітленості та т.і. [5].

Концепція СДА містить в собі такі положення [6]:

- створення інтегрованої системи управління будівлею – системи з можливістю забезпечення комплексної роботи всіх інженерних підсистем будівлі: освітлення, опалення, вентиляції, кондиціонування, водопостачання, контролю доступу та багатьох інших;
- відсутність обслуговуючого персоналу будівлі та передача функцій контролю й прийняття рішень підсистемам інтегрованої системи управління будівлею. У ці підсистеми якраз і закладається «інтелект» будівлі, тобто алгоритм дій у відповідь на зміну параметрів датчиків системи та інші події типу позаштатних ситуацій;
- забезпечення коректної роботи окремих підсистем у разі відмови загальної керуючої системи або інших частин системи;
- мінімізація вартості обслуговування й модернізації систем будівлі, що має забезпечуватися застосуванням загальних стандартів у побудові підсистем, автоматичне конфігурування та виявлення нових пристроїв і модулів при їх додаванні в систему [6].

Головні завдання які вирішують підсистеми СДА:

- управління світлом;
- управління вентиляцією та мікрокліматом;
- управління електроприладами;
- управління домашньою охоронною системою;
- контроль та моніторинг споживання електрики, води та газу;
- централізоване управління системами (домашнього кінотеатру, мультимедіа, тощо);
- управління поливом саду;

- запобігання витоку газу, води;
- управління приводами: жалюзі, ворота, штори;
- імітування присутності власника.

Основними складовими СДА є [7]:

- центральні процесори й хаби – забезпечують взаємодію безлічі підсистем СДА, моніторинг стану та управління цими підсистемами. Кожна підсистема може функціонувати автономно, на основі власних правил, але саме централізована архітектура СДА дозволяє правильно розв'язувати конфлікти між різними підсистемами та забезпечувати складні високорівневі алгоритми прийняття рішень;

- людино-машинні інтерфейси (Human-Mashine Interface, HMI), пульти та панелі управління – це основні компоненти, через які користувачі взаємодіють з СДА. Часто подібні пристрої можуть мати досить великі екрани та поєднувати в собі безліч функцій, виступаючи, наприклад, одночасно центральним процесором або монітором системи відеоспостереження, але можуть й мати досить вузьке утилітарне призначення;

- температурні та погодні датчики, метеостанції, що потрібні для організації правильної роботи багатьох підсистем, алгоритми яких залежать від погоди на вулиці (або навіть її прогнозу) або стану атмосфери в суміжних приміщеннях розумного будинку. Наявність подібних підсистем також дозволяє підвищити комфорт за рахунок різноманітних простих прийомів інформування користувачів про майбутні погодні умови (наприклад, світло в передпокої може змінювати свою колірну гаму залежно від дощу);

- дверна та віконна автоматика – це численні механізми та датчики, які автоматизують відкриття та закриття зовнішніх та внутрішніх дверей, вікон, штор, ролет;

- підсистеми зв'язку та комунікацій – надають інфраструктуру, яка забезпечує канали взаємодії різних підсистем та компонентів СДА. Це

можуть бути як структуровані кабельні системи (Ethernet, KNX, Modbus, HDMI та ін), так і бездротова інфраструктура (Wi-Fi, Z-Wave, ZigBee та ін.).

– допоміжне обладнання та датчики – використовуються для виконання якихось допоміжних функцій. Це можуть бути смарт-пилососи, керовані підйомники, вентиля, підсистеми стеження за басейном, фонтаном та ін. [7].

Таким чином, сучасна СДА за своєю функціональністю спрямована на вирішення трьох основних завдань – забезпечення комфорту, ресурсозбереження, безпеки. Важливою задачею є надання зручного інтуїтивно-зрозумілого інтерфейсу користувача з СДА та її окремими підсистемами.

Підсистема управління освітленням – призначена для управління приладами та механізмами, що регулюють освітлення в будинку і навколо нього. Управління освітленням дозволить користувачеві створювати відповідні сценарії для необмеженої кількості джерел світла з різною яскравістю, включати їх одночасно або із затримкою, імітуючи різні ефекти.

Використовуючи спеціальні світлові світлорегулятори можна не тільки міняти яскравість, на яку загоряється лампа при включенні, а й час, за яке буде досягнута ця яскравість [8].

Функція постійного контролю освітленості призначена в основному для офісних приміщень і дає можливість підтримувати задану користувачем освітленість робочої поверхні незалежно від зовнішніх умов освітлення [8].

Автоматичне включення зовнішнього освітлення в залежності від часу доби і присутності людей не тільки забезпечує додатковий комфорт, але і відлякує небажаних гостей [8].

Основні можливості підсистеми управління освітленням:

- автоматичний перехід освітлення в енергозберігаючий режим, в залежності часу доби та сезону;
- дистанційне керування освітленням в усіх кімнатах, із будь-якої кімнати;

- при комбінації з датчиками руху, автоматично включати/виключати світло у разі появи/відсутності людини в приміщенні;
- світлові сценарії – в залежності від сценарію, регулювати освітлення (приглушення світла, освітлення доріжок, фасаду будівлі) [9].

Підсистема управління мікрокліматом – призначена для керування кліматом в приміщеннях розумного будинку. Підсистема постійно вимірює температуру і вологість в кожній кімнаті, підтримуючи їх на заданому рівні, керуючи клапанами фанкойлу та кондиціонером) [8].

Налаштування всіх терморегуляторів на радіаторах, включення і виключення кондиціонера, регулювання температури і відсотка вологості повітря відбувається за допомогою всього однієї програми або програмної оболонки генерального модуля управління [10].

Сучасні моделі СДА дозволяють враховувати величезний спектр можливостей в налаштуваннях: контроль температури всередині і зовні будинку, корекція відсотка вологості в приміщенні, наявність протягів і відкритих вікон. Останні розробки смарт-систем надають ще один надзвичайно корисний функціонал – можливість вносити поправки в налаштування мікроклімату будинку при зміні погоди за вікном.

Крім того, кліматичний контроль в СДА має на увазі функції іонізації та озонування повітря. Система сама подбає про чистоту і свіжість вдихуваного вами повітря, наситивши його необхідними для хорошого самопочуття мікрочастинками [10].

Основні можливості системи управління мікрокліматом:

- оптимізація витрат на електрику, природний газ та воду;
- автоматично підтримує заданий температурний режим в різних зонах приміщення, а також регулює температуру в залежності від пори року, доби та зовнішньої температури повітря;
- детальна статистика споживання електроенергії, природного газу, води;

- управління системою опалення, вентиляцією, кондиціонуванням, теплими підлогами, радіаторами опалення, за допомогою керуючих панелей та мобільного застосунку;
- отримання інформації щодо температури та якості і складу повітря в різних зонах будівлі;
- кліматичні сценарії – в залежності від сценарію, регулювати клімат приміщень (піднімати температуру в певних приміщеннях, за певний час до повернення додому власника, режим «Запобігання замерзання» при від'їзду власника, вибір температурних програм: «Вихідні», «Відпустка», «Звичайний день»).

Підсистема імітування присутності – призначена для захисту будівлі від злодіїв, шляхом імітування присутності домовласника. Активація сценарію відбувається в момент, коли останній мешканець залишає свій будинок. Після активації починається імітація присутності людини: світло включається в різних кімнатах, штори закриваються, включається телевізор і аудіо-система, відтворюється гавкіт собаки. Також, під час проникнення злочинця на територію або в одну з кімнат власник буде автоматично сповіщений. В цей же час вбудована система відеоспостереження почне зйомку. Також, є можливість створити декілька різних сценаріїв імітації, для кожного дня [8].

1.2 Вимоги до реалізації концепції СДА

Як показали проведені дослідження, реалізація концепції СДА пов'язана з проблемами одночасного виконання кількох критичних вимог [11]:

- використання широкої номенклатури контролюючих і виконавчих пристроїв;
- забезпечення високого ступеня захисту даних;
- мінімізація витрат, пов'язаних зі споживанням ресурсів [11].

Залежно від вимог замовника, архітектура СДА, а також набір апаратних і програмних рішень може істотно відрізнятися. З одного боку, для створення систем домашньої автоматизації на ринку пропонується величезна кількість пристроїв, платформ і технологій. Основними вимогами до них є функціональна сумісність, гнучкість, безпека, простота установки і експлуатації та ін. Деякі компанії пропонують інтеграцію своїх розумних помічників в уже існуючі прилади (Alexa, Google Home). Інші працюють над новими розумними домашніми пристроями (Nest, Walmart, ABB, British Gas).

Існує безліч дротових і бездротових (KNX, Ethernet ZigBee, Z-Wave, Wi-Fi, Bluetooth) технологій для обслуговування розумних домашніх потреб. Є розумні домашні платформи (Wink, SmartThings від Samsung, HomeKit від Apple, Iris від Lowe) і рішення від постачальників безпеки (наприклад, Tor Home Assistant) [11].

З іншого боку, питання уніфікації та сертифікації, а також їх інтеграції в єдину систему, що легко налаштовується, залишаються відкритими. Наслідок для індустрії в цілому полягає в уповільненні темпів ринку, так як споживачі сумніваються в ефективності і безпеці подібних систем і не поспішають встановлювати їх [11].

Таким чином, створення систем типу Розумний будинок вимагає широкого спектру рішень щодо різноманітних контрольно-виконавчих пристроїв, а також технологій їх інтеграції в єдину систему з урахуванням вимог енергоефективності, безпеки, кібербезпеки та ін. [11].

1.3 Огляд тенденцій ринку СДА

У 2018 році обсяг європейського ринку пристроїв для розумного будинку склав рекордні 88,8 млн штук, збільшившись на 23,1% відносно 2017 року (за даними International Data Corporation (IDC)). Оцінки світового ринку систем «розумний дім» значно різняться, але більшість експертів відзначають позитивні темпи зростання. За деякими оцінками в США

впровадження СДА з 2017 по 2022 рр. становитиме 42% щорічного зростання. Світовий же ринок СДА в 2018 році оцінювався на рівні 76 млрд. доларів США. Прогнозується, що він до 2024 року складе 151,4 млрд. доларів США [12].

Про актуальність цього напрямку для України свідчить поява значної кількості українських компаній, що спеціалізуються в цій галузі. Зокрема в Запоріжжі з'явилися компанії 1M-Smart home, ТОВ "ПКФ МОТОР" (офіційний представник Legrand), Монада (офіційний представник Siemens), що пропонують обладнання для СДА та послуги з розроблення таких систем з урахуванням індивідуальних потреб замовників [11].

Поки що український ринок СДА знаходиться на етапі формування. Однак, в останні роки все ж спостерігається істотний прогрес в сфері створення СДА. Найпопулярнішими залишаються рішення для «розумного» витрачання ресурсів (електроенергія, вода, газ, тощо) і безпеки будинку. Причому, поняття безпеки стосується не тільки проникнення в житло сторонніх, але і виникнення там пожеж, протікання води, коротких замикань, чадного газу, диму або вогню. Іншою важливою складовою безпеки є кібербезпека та захист інформації СДА. І якщо питанням енергоефективності та безпеки подібних систем приділяється величезна увага, то питання кібербезпеки поки залишаються в тіні, хоча кібератаки останніх років показали небезпечність такого ставлення [11].

Основні проблеми для динамічного розвитку ринку СДА в Україні наступні:

- відносно велика вартість СДА з низькою купівельною спроможністю населення;
- погана обізнаність потенційних користувачів, щодо функцій СДА, їх характеристиках та переваг використання;
- побоювання в частині захисту персональних даних та забезпечення конфіденційності;

- соціальні бар'єри, пов'язані зі зміною звичайного способу життя, побоювання щодо використання складних розумних пристроїв;
- дворазове відставання від європейських країн за часткою охоплення Інтернет-технологіями населення країни [13].

1.4 Дослідження дротових та бездротових технологій для СДА

Аналіз показав, що існуючі технології дротового та бездротового зв'язку розроблялися до виконання різних завдань та працюють по-різному [14].

Розглядати ці технології потрібно крізь призму еталонної моделі Open Systems Interconnection (OSI) [15].

Міжнародна організація стандартизації (International Organization for Standardization, ISO) розробила модель взаємодії відкритих систем OSI ще у 80-х роках минулого століття. OSI є основою для координації розробки стандартів зв'язку та досі залишається еталонною моделлю для комунікаційних мереж. Структура OSI розглядає процес комунікацій у мережі як керовану ієрархію, що складається із семи рівнів (рис. 1.2). Кожен з них відповідає за певні функції та вирішує чітко визначені завдання в рамках одного рівня, а також взаємодії з рівнями, розташованими безпосередньо нижче та вище [15].

Бездротові протоколи зв'язку – це найпопулярніший вид протоколів, завдяки простому методу інсталяції устаткування. Бездротові протоколи зв'язку через відсутність дротів заощаджують простір, час та фінансові витрати [14].

Переваги бездротових протоколів:

- не потрібно прокладати дроти в будівлях з готовим ремонтом;
- більш економічний варіант, ніж дротовий;



Рисунок 1.2 – Комунікації у СДА згідно з моделлю OSI [15]

- зменшення кількості дротів;
- ринок містить багато систем з невисокою вартістю.

Недоліки:

- якість зв'язку радіоканалу;
- потребують елементи живлення від батарейок;
- якщо в системі присутній передавач, який живиться від мережі змінного струму, тоді потрібно буде тягнути дріт в монтажну коробку;
- кібер небезпечність [15].

ZigBee – один з найбільш широко використовуваних відкритих протоколів бездротових комунікаційних технологій в сучасних СДА. Використовується дуже часто через свою низьку ціну на модулі та високе енергозбереження. На окремому пристрої ZigBee здатен пропрацювати до 2 років без необхідності заміни елемента живлення.

Переваги ZigBee:

- зріла й поширена технологія;
- висока відмовостійкість і масштабованість;

- низьке енергоспоживання.

Недоліки ZigBee:

- використовує діапазон 2,4 ГГц, де сильні перешкоди від Wi-Fi, Bluetooth, мікрохвильових печей;
- вкрай погана сумісність між пристроями ZigBee різних виробників через занадто м'які умови сертифікації;
- проблеми з безпекою через недотримання виробниками вимог сертифікації [15].

Основний конкурент ZigBee – Z-Wave – є лідером серед технологій домашньої автоматизації за кількістю інстальованих пристроїв. Цей протокол бездротового зв'язку з низьким енергоспоживанням спеціально розроблений, щоб дати звичайним користувачам можливість ефективного та надійного віддаленого управління широким спектром датчиків і виконавчих пристроїв СДА. Пристрої з цим протоколом більш дорогі, ніж ZigBee, через високу вартість ліцензування мікросхем зв'язку [15].

Переваги Z-Wave:

- світовий лідер за поширеністю і кількістю інстальованих пристроїв в СДА;
- висока відмовостійкість і масштабованість;
- високий рівень безпеки;
- низький рівень споживання електроенергії;
- відмінна взаємна сумісність пристроїв Z-Wave різних виробників завдяки сертифікації;
- захищеність від впливу інших пристроїв Wi-Fi, Bluetooth, мікрохвильових печей та інших пристроїв, що працюють в діапазоні 2,4 ГГц [15].

Недолік Z-Wave полягає в тому, що для різних країн Z-Wave використовує різні радіодіапазони.

Протокол Wi-Fi дуже часто використовується в СДА, тому що Wi-Fi – найпоширеніша бездротова мережева технологія з мільярдами користувачів

по всьому світу. Зазвичай цей стандарт зв'язку застосовується для того, щоб з'єднати смартфон або планшет з уже готовою автоматизованою системою та керувати СДА – це більш зручно, ніж використовувати настінну сенсорну панель, пульт дистанційного керування або інтерпретатор мови [15].

Переваги Wi-Fi:

- широко поширений в мобільних пристроях та комп'ютерах;
- забезпечує високу швидкість передачі даних;
- має достатній радіус дії.

Недоліки Wi-Fi:

- високе енергоспоживання не дозволяє використовувати Wi-Fi в автономних датчиках і актуаторах СДА;
- мережева топологія «зірка» не гарантує відмовостійкості мережі;
- складний процес додавання в мережу деяких пристроїв;
- проблема з безпекою, такі як механізм управління ключами.

Протокол Bluetooth LE досить неоднозначний бездротовий протокол, тому що отримати доступ до приладів можна лише перебуваючи поруч з ними, а радіус дії менший за Wi-Fi. Bluetooth LE використовує діапазон 2,4 ГГц, тому є проблеми з перешкодами. Проте є і цікава у Bluetooth LE функція – це так звані маячки. Використовуючи функцію Bluetooth LE, маячки можуть змусити смартфони виконувати певні дії від push-повідомлень на основі визначення місця розташування до послуг точного позиціонування [15].

Переваги Bluetooth LE:

- висока швидкість передачі даних в мережі;
- помірне енергоспоживання в порівнянні з Wi-Fi.

Недоліки Bluetooth LE:

- використовує діапазон 2,4 ГГц, де сильні перешкоди від Wi-Fi, Bluetooth, мікрохвильових печей;
- недостатня надійність;

– малий практичний радіус дії (максимум до 10 м в приміщеннях), топологія «зірка» не дозволяє розширювати мережу за допомогою ретрансляторів [15].

В таблиці 1.1 представлено порівняльну характеристику основних характеристик бездротових протоколів [16].

Таблиця 1.1 – Порівняльна характеристика бездротових протоколів

Критерій	ZigBee	Z-Wave	Wi-Fi	Bluetooth LE
Дальність	100 м/Mesh	30 м/Mesh	До 100 м	80 м
Частота	915 МГц, 2.4 ГГц	900 МГц	2.4 ГГц, 5 ГГц	2.4 ГГц
Швидкість передачі	250 кбіт/с	10-100 кбіт/с	Макс. 7 Гбіт/с	<1 мбіт/с
Споживання енергії	Низьке	Низьке	Високе	Середнє
Автентифікація	Так	Так	Так	Ускладнена
Шифрування	Так	Так	Так	Так
Двонаправленість	Так	Так	Так	Так
Стандарт	ZigBee	Z-Wave	IEEE 802.11	Bluetooth 4.0
Маштабованість	Так	Обмежена	Так	Так

Дротові протоколи зв'язку – це протоколи, в яких всі керуючі елементи СДА: активатори, датчики, керуючі пристрої – пов'язані між собою єдиною провідною шиною. Роль шини може виконувати як спеціальний кабель, так і звичайна вита пара. По цій шині передаються команди виконавчих пристроїв.

Переваги дротових протоколів:

- передача сигналів по дроту більш надійна;
- достатня швидкість передачі сигналів;
- більше функціоналу та можливостей;
- довгий термін служби;

- різноманітність інтегрованих систем.

Недоліки дротових протоколів:

- дороге інсталування дротової системи для будівель с готовим ремонтом;
- перебитий інформаційний дріт виводить з ладу СДА, або її частину;
- після прокладки проводу, зрушити його буде практично неможливо [15].

Протокол SCS – провідна технологія від компанії Legrand на базі двох-провідної шини, яка призначена як для обміну інформації, так і для подачі живлення 27В, до якого паралельно приєднуються всі пристрої системи. Дана шина представляє собою кабель витой пари, при цьому вона поляризована, що виключає можливість переплутати полярність. За допомогою цього протоколу можна: управляти світлом, автоматизацією, поширенням звуку, енергоспоживанням, терморегуляцією, домофоном та системою сигналізації.

Протокол зв'язку KNX – надійна дротова технологія, яка зарекомендувала себе лідером по престижності на ринку домашньої автоматизації. Протокол дорогий та складний у налаштуванні, але він гарантує стабільність і надійність роботи системи в подальшому.

Протокол зв'язку RS485 – найпопулярніший стандарт в провідних пристроях СДА. В цьому протоколі використовується шина, по якій передаються тільки дані, а живлення подається окремо. Максимально можлива кількість пристроїв, з'єднаних однією шиною – 256, а максимальна довжина кабелю може досягати 1200 метрів.

Протокол зв'язку CAN – найчастіше використовують в автомобілебудуванні. Особливістю даного протоколу є залежність швидкості передачі даних від довжини дроту. Чим більша довжина, тим повільніше передається інформація. Завдяки цій особливості мало де можна зустріти використання цього протоколу у розробці СДА [15].

Таким чином, при реалізації СДА вибір протоколу є актуальною задачею, що має виконуватися з урахуванням цілого комплексу вимог щодо енергоефективності, вартості, надійності та ін.

1.5 Дослідження архітектури СДА, інтегрованої в хмару

Концепція IoT для СДА може бути реалізована шляхом передачі даних з використовуваних датчиків в хмарне сховище, де вони будуть оброблятися та зберігатися, а також шляхом передачі команд від користувача на кінцеві керовані пристрої. У свою чергу, користувачеві буде доступний інтерфейс (через браузер або мобільний застосунок), призначений для моніторингу даних із датчиків, а також управління пристроями. Таким чином, користувачі отримають інструмент віддаленого керування та моніторингу своєї СДА. Основні переваги використання хмарних технологій для СДА:

- у будь-який час та в будь-якому місці користувачеві буде доступна вся інформація про обстановку в СДА;
- у разі присутності серед мешканців будинку людей з особливими потребами через хмару можна забезпечити передачу даних з медичних пристроїв безпосередньо лікарю;
- СДА стає більш гнучкою, набагато легше буде інтегрувати нові пристрої лише забезпечивши їм доступ в Інтернет, а не перебудовуючи всю систему заново;
- через хмарний сервер можливе надання єдиного інтерфейсу управління підсистемами СДА у разі використання пристроїв від різних виробників із різними протоколами передачі [17].

Типове апаратне забезпечення СДА (рис. 1.3) зазвичай об'єднує такі компоненти:

- контролер (наприклад, ПК), на який встановлюється ПЗ для управління СДА;

- модулі (плати) розширення – спеціальні пристрої, до яких підключаються різні датчики та актуатори;
- кінцеве обладнання – датчики для відстеження різних параметрів та пристрої, керування якими здійснює СДА [18].

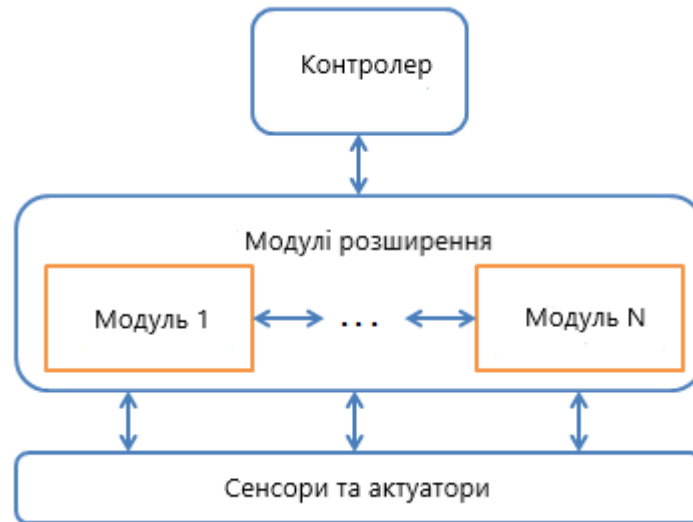


Рисунок 1.3 – Структура апаратного забезпечення СДА [18]

Здійснення функції віддаленого доступу для СДА можливе за допомогою застосування хмарних обчислень, коли користувачі забезпечуються доступом до мережесх обчислювальних ресурсів, сервісів та програм. Існує кілька моделей хмарних обчислень. Для віддаленого управління СДА більше підходить модель SaaS (Software as a Service, програмне забезпечення як послуга). Ця модель передбачає надання клієнту доступу до ПЗ через Інтернет. Основна перевага моделі SaaS для кінцевого користувача полягає у відсутності необхідності встановлення та оновлення ПЗ, також йому не потрібно дбати про працездатність обладнання, на якому функціонує програма [18].

При застосуванні хмарних обчислень у СДА можливі два варіанти. У першому випадку контролер (сервер) для керування пристроями СДА може бути розташований не в самому будинку (цю функцію візьме на себе хмара), завдяки чому управління СДА може здійснюватися з будь якої точки наявності доступу до Інтернету. При другому варіанті (рис. 1.4) контролер

може розташовуватися в будинку, але при цьому через хмару забезпечуватиметься лише віддалене управління – все ПЗ буде встановлено на сервері хмар. Крім того, у другому випадку від домашнього контролера буде потрібно лише функція для забезпечення модулям розширення доступу до Інтернету, що в свою чергу зменшує вимоги до технічних характеристик контролера. Також у разі впровадження віддаленого управління вже в існуючу СДА не знадобиться заміни жодного обладнання, достатньо лише забезпечити доступ контролера до хмарного серверу [18].



Рисунок 1.4 – Структура апаратного забезпечення СДА з функцією віддаленого керування [18]

Безпосередньо віддалене управління СДА можна здійснювати або через веббраузер, або через спеціальний мобільний застосунок. Багато сучасних пристроїв, що використовуються в СДА, як кінцеве обладнання, так і керуючі пристрої, працюють за власними протоколами передачі даних і, крім того, можуть взаємодіяти з Інтернет-сервісами тільки через свої API. Тому, часто немає можливості розширити СДА, або додати до неї пристрої, що працюють за іншими протоколами передачі даних.

Однак завдяки хмарному сервісу, який надаватиме загальний інтерфейс управління всіма системами, а різні пристрої взаємодіятимуть між собою через хмару, з'являється можливість використання пристроїв від різних виробників з різними протоколами передачі даних. Застосування хмарних технологій у СДА дозволить зробити їх набагато гнучкішими, а також скоротити витрати на обслуговування та розширення системи [18].

Для успішної взаємодії хмарного сервера з пристроями СДА обидва ці компоненти повинні «розмовляти» один з одним однією мовою. Найпростішим рішенням у цьому випадку є обмін даними через XML-повідомлення. Одним із протоколів, що використовують XML для обміну даними, є простий протокол доступу до об'єктів SOAP (Simple Object Access Protocol). Основною перевагою використання SOAP є те, що він здатний забезпечувати безперервну взаємодію вебсервісу з пристроями, що працюють за різними протоколами передачі даних. Інші переваги застосування формату SOAP перед іншими форматами передачі [18]:

- кодувати в XML структури даних з використанням SOAP так само легко, як і дані простих скалярних типів;
- при використанні SOAP-повідомлень надаються додаткові інструменти, що дозволяють легко додавати, наприклад, функції безпеки або трасування;
- існують набори інструментів SOAP для різних мов програмування.

Для СДА на основі хмарних технологій потрібно розглядати наступні рівні автоматизації [17]:

- віддалене управління та моніторинг через мобільні та стаціонарні пристрої (смартфони, планшети, ПК, ноутбуки);
- забезпечення віддаленого управління (хмарний рівень). Основний компонент – хмарний сервер, що призначений для зберігання даних про стан сенсорів та пристроїв. Він виступає також як посередник між віддаленими пристроями управління та домашнім сервером. Ця функція здійснюється

шляхом передачі команд, наприклад, з мобільних пристроїв у хмару, де вони обробляються та передаються далі на домашній сервер;

– внутрішнє управління, моніторинг та зв'язок із хмарою, основний компонент – домашній сервер. Він призначений для отримання команд від хмарного сервера та їх подальшої передачі на контролери, а також передачі у зворотному порядку даних із датчиків, що надходять на контролер. Через домашній сервер також забезпечується управління СДА по локальній мережі як з домашнього сервера або іншого домашнього комп'ютера, так і з мобільних пристроїв по Wi-Fi і Bluetooth (рис. 1.5);

- автоматичне управління;
- кінцеве обладнання [17].

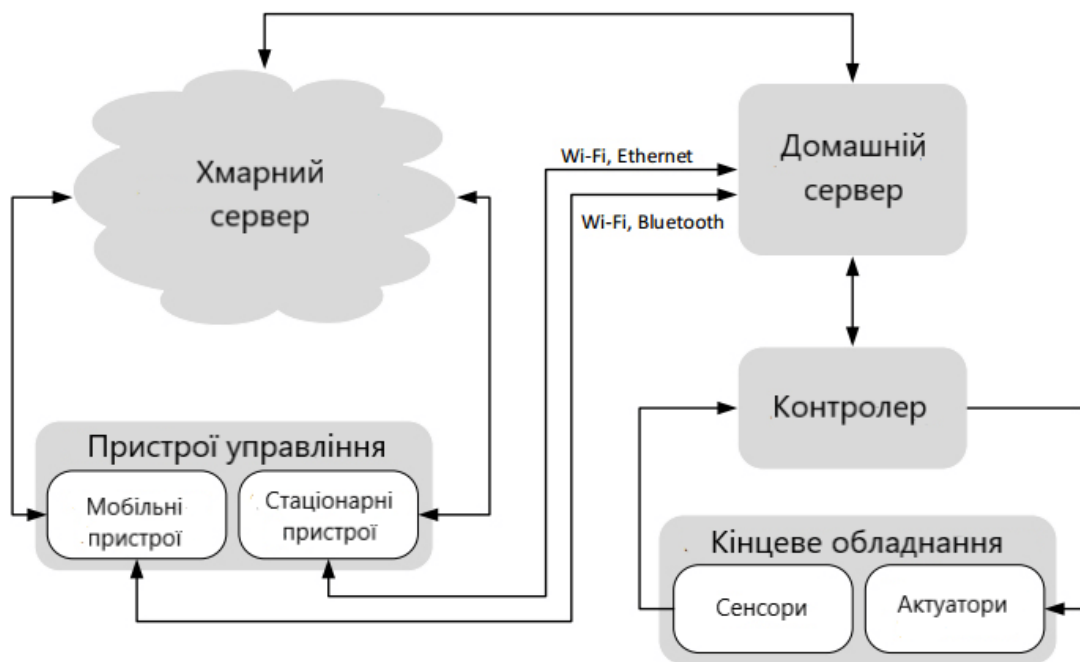


Рисунок 1.5 – Хмарна архітектура СДА [17]

На основі апаратної складової архітектури СДА, інтегрованої в хмару, програмна складова включає такі рівні [17].

- серверна частина хмари: вебсервер, БД, скрипти для обробки та передачі даних;

- клієнтська частина хмари, що надає графічний інтерфейс управління та моніторингу. Будується, наприклад, на трьох основних технологіях створення вебзастосунків: HTML, CSS та JavaScript;
- серверний домашній застосунок, комплектація якого подібна до серверної частини хмари;
- клієнтська домашня програма, комплектація якої аналогічна клієнтській частині хмари;
- мобільний застосунок;
- програма для контролера (наприклад Arduino) [17].

Алгоритм виконання функції віддаленого керування СДА [17] наведено на рисунку 1.6.

Етап 1. Початок роботи.

Етап 2. Встановлення з'єднання з хмарним сервісом із мобільного або вебзастосунка.

Етап 3. Авторизація користувача на хмарному сервісі. Відкриття сесії.

Етап 4. Перевірка підключення СДА до хмарного сервісу.

Етап 5. Завантаження графічного інтерфейсу для управління СДА.

Етап 6. Надсилання запиту на хмарний сервер про стан підсистем СДА (отримання даних із датчиків та передача інформації про стан керованих пристроїв).

Запит про стан систем здійснюється протягом усієї сесії. Залежно від типів параметрів, що відстежуються, встановлюється різний час для різних систем. Наприклад, дані з температурних датчиків можна запитувати раз на 30 хвилин, а дані про стан світильників – раз на кілька секунд.

Етап 7. Надсилання команди для управління пристроями, що входять до СДА. Отримання відповіді виконання команди.

Етап 8. Протоколювання подій на сервері хмари при кожному запиті.

Етап 9. Закриття сесії.

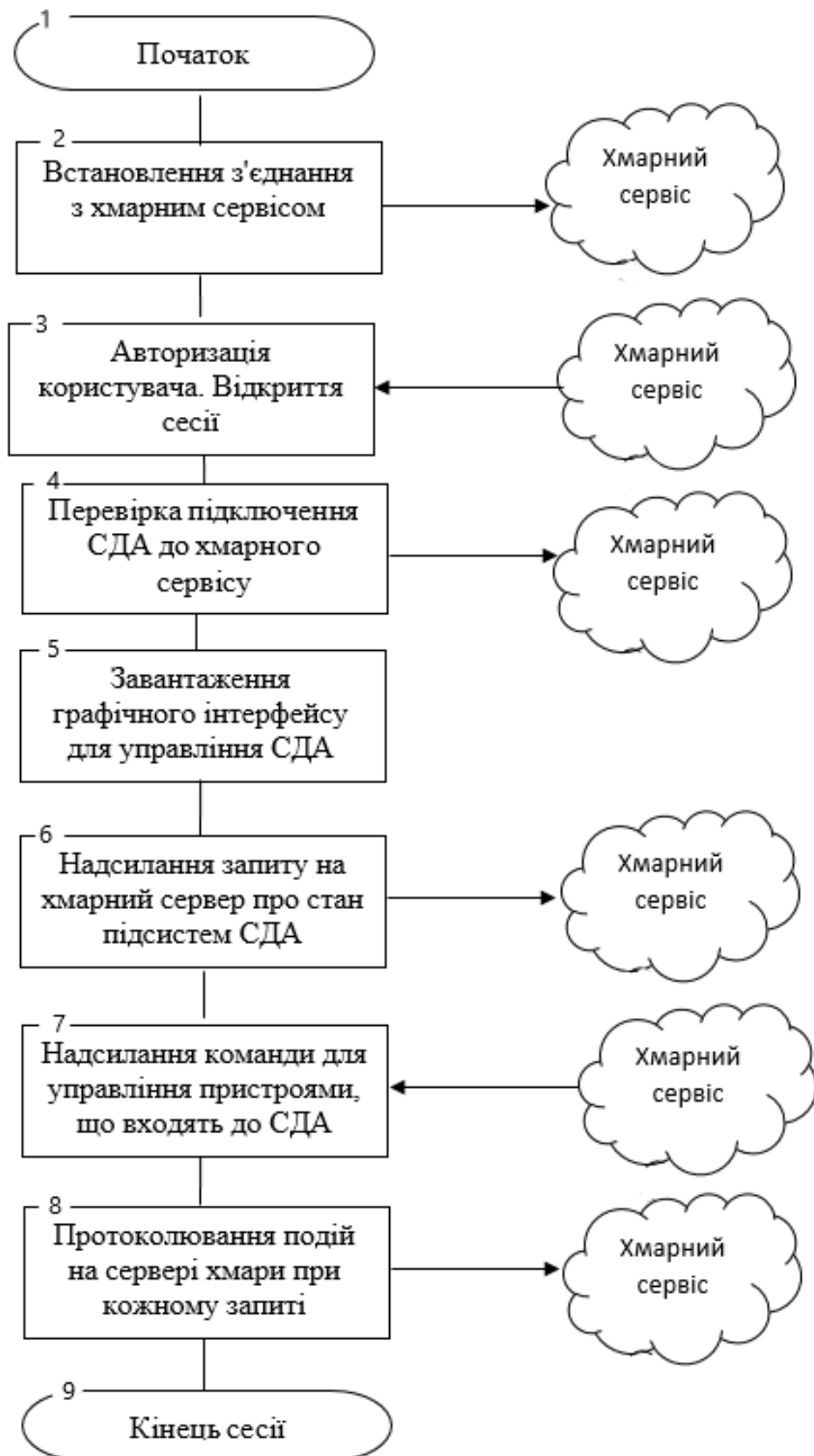


Рисунок 1.6 – Схема алгоритму функції віддаленого керування

Таким чином, робота через «хмару» позбавляє користувача багатьох проблем:

- не потрібно забезпечувати налаштування безпеки мережі;
- середовище для роботи з різним обладнанням розміщується в одному місці;
- підключення нових пристроїв не потребує додаткового втручання користувача в систему;
- виробники обладнання гарантують точну роботу серверів та безпеку особистої інформації клієнта [19].

Основний недолік полягає в тому, що більша частина взаємодії між пристроями здійснюється через Інтернет, який подекуди працює нестабільно і у разі обриву зв'язку, СДА буде частково паралізована на деякий час. Інший, фактор – ризик перехоплення даних. Гіпотетична небезпека полягає у захопленні злочинцями доступу до численних пристроїв користувачів та проведення DDoS-атак на великі ресурси у різних цілях – від комерційних до хуліганських [19].

1.6 Постановка задач дипломної кваліфікаційної роботи

Як показали проведені дослідження, можна виділити основні критерії оцінювання СДА:

- можливості розширення функціоналу та масштабування (наявність стандартних складових частин та можливість додавання обладнання);
- порядок підключення системних компонентів між собою (дротовий, бездротовий);
- спосіб управління (через ПК, смартфон, панель управління);
- канали зв'язку із користувачем (Інтернет, GSM, радіоканал);
- дальність дії сигналу;
- вартість;

– виробник та ін.. [20].

Для оцінки сукупності критеріїв доцільно створювати та досліджувати прототипи СДА на основі технології з відкритим кодом, а також широко доступних ресурсів.

Тому, тема магістерської роботи є актуальною.

Мета роботи – дослідження та розробка прототипу СДА на основі методології повторного використання та хмарних технологій для підвищення ефективності взаємодії компонентів СДА.

Задачі роботи:

- дослідити методи та засоби прототипування СДА;
- розробити хмарну архітектуру прототипу СДА;
- обрати апаратні та програмні засоби розробки;
- виконати програмну реалізацію;
- виконати дослідження віртуального та фізичного прототипів СДА.

2 АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ ПРОТОТИПУВАННЯ СДА

2.1 Методи розробки прототипів СДА

Створення СДА потребує широкого спектра знань та практичних навичок щодо різноманітних контрольно-виконавчих пристроїв, а також технологій їх інтеграції в єдину систему з урахуванням вимог енергоефективності, безпеки та кібербезпеки. Дослідити структурні та функціональні особливості СДА дозволяють їх прототипи, що можуть бути створені у вигляді стаціонарних демонстраційних стендів, що створили деякі компанії, або віддалених лабораторій [21].

Компанія Siemens пропонує мобільний демонстраційно-навчальний стенд GAMMA Training Kit GTK 5.1 (рис. 2.1), що підходить для організації мобільних класів, містить керуючий модуль та базову периферію для демонстрації принципів побудови та функціонування систем контролю приміщень та будівель [22] - [23].

Іншими прикладом є KNX-EIB Experimenter with Simulator від компанії Wuekro [24]. KNX-EIB Experimenter with Simulator (рис. 2.2) представляє собою готовий стенд для демонстрації роботи комунікаційної шини KONNEX (KNX), що є одним із варіантів реалізації мережі апаратних засобів у СДА.



Рисунок 2.1 – Стенд GAMMA Training Kit GTK [22]



Рисунок 2.2 – Wuekro KNX - EIB Experimenter with Simulator [24]

Компанія Legrand пропонує мобільний демонстраційний стенд, який можна побачити зокрема в шоурумі компанії в Запоріжжі (рис. 2.3). З використанням даного стенду проводяться тренінги-семінари, на яких обговорюються практичні питання: обладнання системи домашньої автоматизації MyHOME від Legrand (SCS-шина, пристрої управління, реле-активатори, блок живлення); програмне забезпечення MyHomeSuite та ScenarX; практичне створення проєкту (програмування команд, робота з блоком складних сценаріїв, програмування сенсорної панелі, створення проєкту на iPad) та багато іншого [25].



Рисунок 2.3 – Демонстраційний стенд MyHOME від Legrand [25]

Компанія GIRA, з якою активно співпрацюють українські архітектори, проектувальники та дизайнери, також має подібний стенд для демонстрації функціоналу продукції (рис. 2.4) [26].



Рисунок 2.4 – Демонстраційний стенд GIRA [26]

Корпорація Intel Corp. також створила прототип СДА (рис. 2.5 - 2.6), що базується на апаратурі компанії, з метою її просування [27].



Рисунок 2.5 – Демонстраційний стенд від Intel Corp. [27].

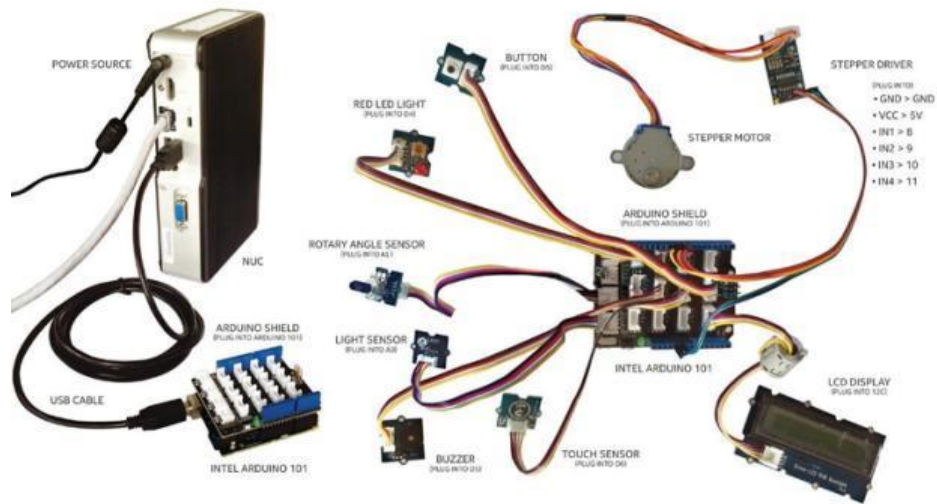


Рисунок 2.6 – Обладнання прототипу від Intel Corp. [27]

У проєкті використано наступне обладнання:

- ПК малого форм-фактору Intel® NUC Kit;
- плата Arduino;
- Intel IoT Commercial Development Kit.

Прототип використовує операційну систему (ОС) Intel® IoT Gateway Software Suite. Також для проєкту було створено декілька застосунків для збирання інформації з датчиків, керування електромеханічними пристроями, застосунок для адміністрування та мобільний застосунок для смартфона.

Головним недоліком розглянутих демонстраційних стендів є те, що вони здебільшого не орієнтовані на дослідження, а на демонстрацію готових рішень СДА з метою їх реклами та подальшого продажу. А також використання таких прототипів для навчання студентів є занадто дорогим. Тому, виші розробляють та впроваджують в освітній процес віддалені та віртуальні лабораторії, які дозволять студентам проводити онлайн експерименти з обладнанням для отримання практичних навичок в розробці СДА [23].

Smart Home System (A wireless, voice-controllable, household system) – віддалена лабораторія (ВЛ), що створена для вивчення системи голосового

керування об'єктами [28]. Систему розроблено у Корнелльському університеті в місті Ітака у шататі Нью-Йорк (рис. 2.7).

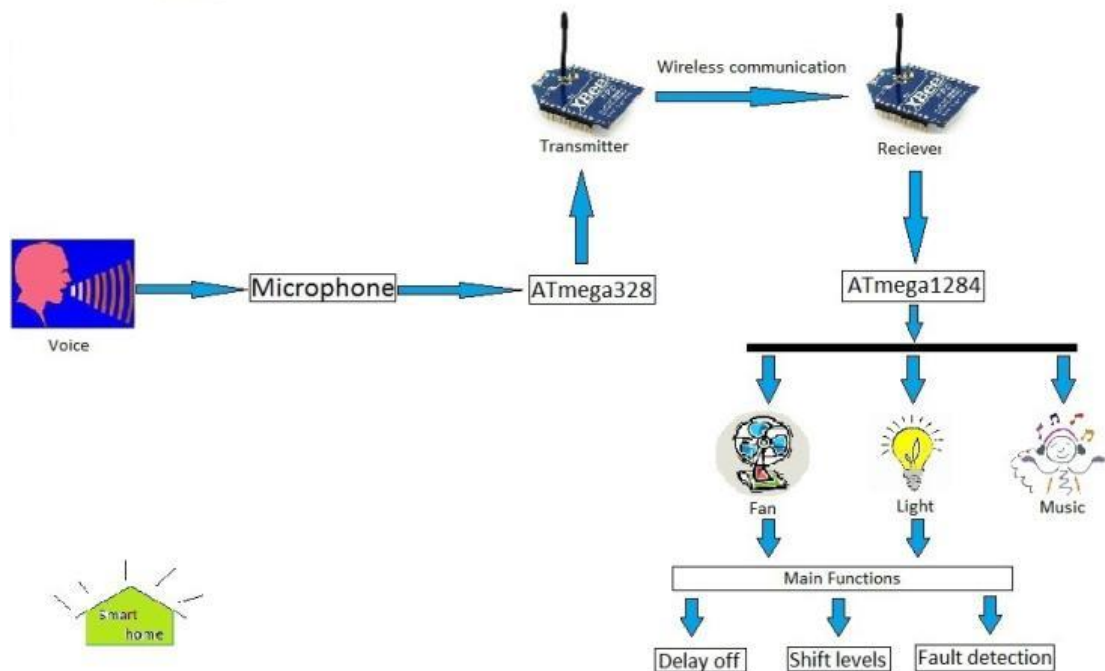


Рисунок 2.7 – Схема обладнання лабораторії голосового керування [28]

ВЛ використовує наступне обладнання:

- мікрофон;
- мікроконтролер (МК) Atmega328;
- МК Atmega1284;
- Wi-Fi модуль Xbee;
- LCD модуль.

Лабораторія має можливість керування вентиляцією, освітленням, відтворення музичних файлів за допомогою голосового керування [28].

RGB Led laboratory & Servomotor – ВЛ, що створена на кафедрі електричної та комп'ютерної інженерії університету дистанційної освіти у місті Мадрид [29]. У експериментах використовується наступне обладнання:

- Led куб;
- серводвигун;
- Arduino UNO;
- мінікомп'ютер Raspberry PI B+.

Лабораторія дозволяє проводити експерименти з вивчення принципів керування RGB світлодіодами у матриці та вивчати основи керування серводвигунами [29].

В роботі [30] представлено архітектуру ВЛ, яка підтримує розгортання багатьох експериментів на базі Arduino. Це ВЛ датчиків, що складається з одинадцяти датчиків і РК-дисплея, підключеного до Arduino MEGA, а також віддалена лабораторія 3D RGB LED, що складається з 16 світлодіодів RGB, підключених до Arduino UNO-сумісної плати. Результати виконання експериментів можна спостерігати в режимі реального часу через IP-камеру. Основним пристроєм цієї лабораторії є плата Arduino, яка з одного боку підключена до інших пристроїв, таких як світлодіодний куб (рис. 2.8), механічний кран, датчики, тощо, а з іншого боку — до мінікомп'ютера Raspberry Pi, яка діє як вебсервер. З'єднання між Raspberry Pi і платою Arduino здійснюється через порт USB. Кількість плат Arduino, які можна підключити до сервера Raspberry Pi, обмежена через кількість доступних портів USB [30].

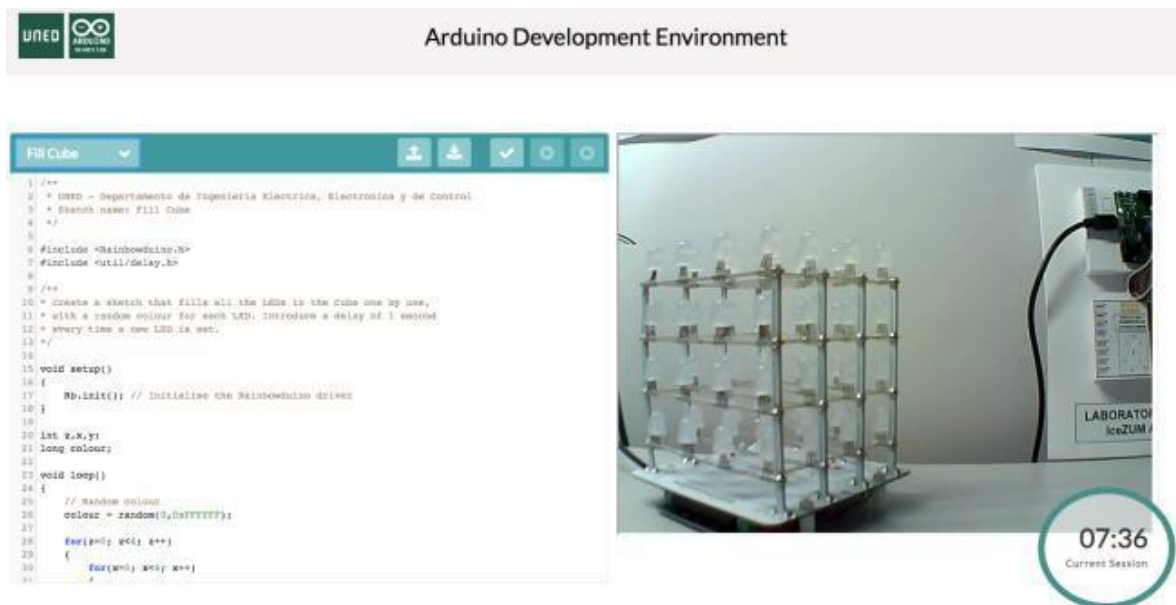


Рисунок 2.8 – Графічний інтерфейс експерименту з 3D світлодіодним кубом [30]

Іншим прикладом є ВЛ Smart House&IoT (рис. 2.9), створена на кафедрі програмних засобів Національного університету «Запорізька політехніка», що призначена для практичного вивчення різних платформ, пристроїв, протоколів та інтерфейсів для реалізації СДА та IoT технологій [31] - [32].

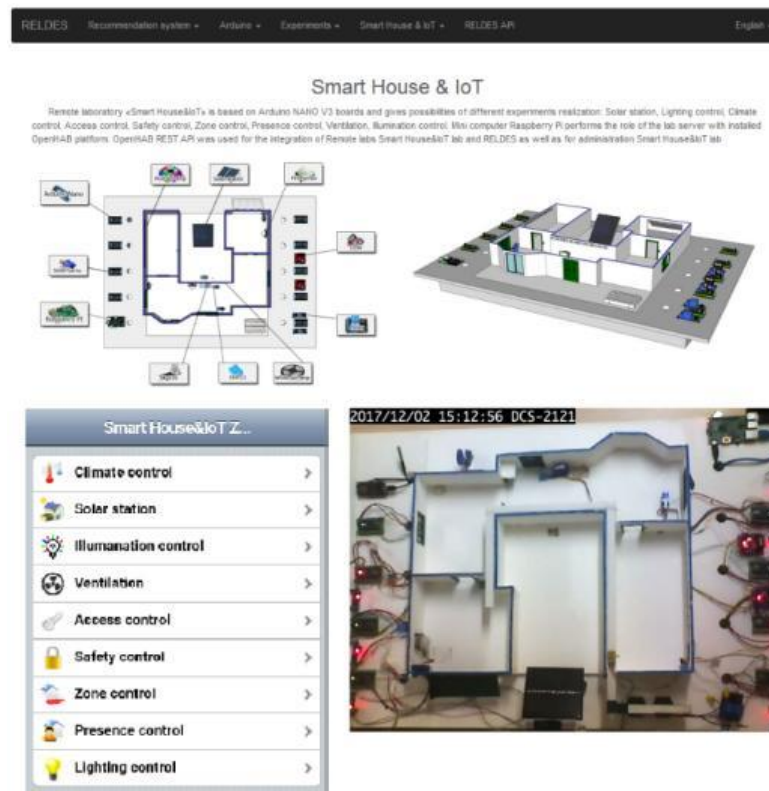


Рисунок 2.9 – Графічний інтерфейс ВЛ Smart House&IoT [32]

При цьому ставляться задачі вивчення мобільних та хмарних технологій, методів обміну даними та обробки великих масивів даних. Студенти набувають знання і практичні навички з реалізації сучасних цифрових систем та технологій шляхом виконання дистанційно керованих експериментів, вивчення описів цих експериментів, реалізації різних сценаріїв керування та програмування. Для навчання на основі віддаленої лабораторії розроблено декілька сценаріїв, що передбачають як індивідуальну роботу студентів з окремими експериментами, так і роботу в команді для визначення логіки роботи системи загалом, програмування і моніторингу складних процесів [32].

Структурну схему апаратного забезпечення ВЛ наведено на рисунку 2.10. Основою апаратного забезпечення лабораторії є мінікомп'ютер Raspberry Pi та плати Arduino [32].

ВЛ поєднує декілька підсистем для керування процесами: мікроклімату, освітлення, генерації сонячної енергії, вентиляції та кондиціонування, доступу, захисту даних та ін.

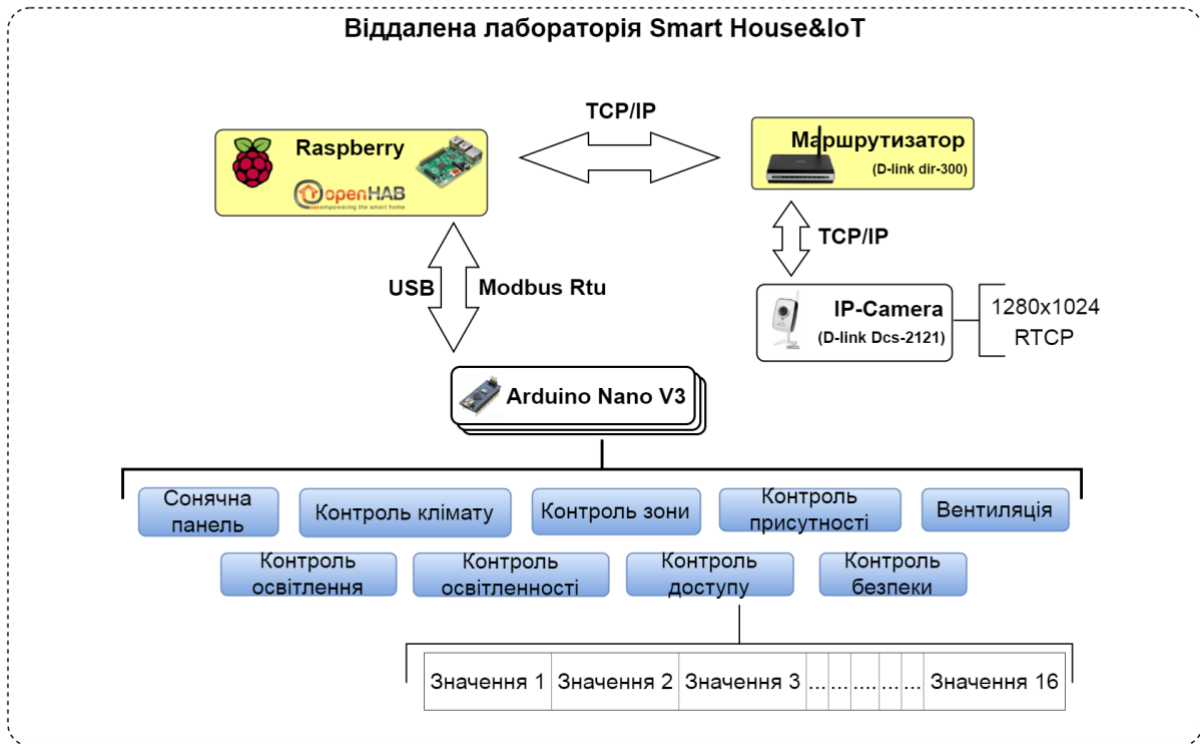


Рисунок 2.10 – Структура ВЛ Smart House&IoT [32]

Структура кожної підсистеми складається з плати Arduino Nano, що за допомогою внутрішньої програми оновлює регістри пам'яті, які відповідають за поточний стан окремого сенсору. Кожна окрема підсистема для зв'язку з сенсорами використовує цифрові, аналогові входи та виходи, необхідні для модулів інтерфейси та широтно-імпульсну модуляцію (ШИМ). Типову структуру підсистеми наведено на рисунку 2.11.

Плата Arduino виконує функції зчитування даних цифрових або аналогових сигналів з датчиків та занесення отриманих даних у комірку за номером.

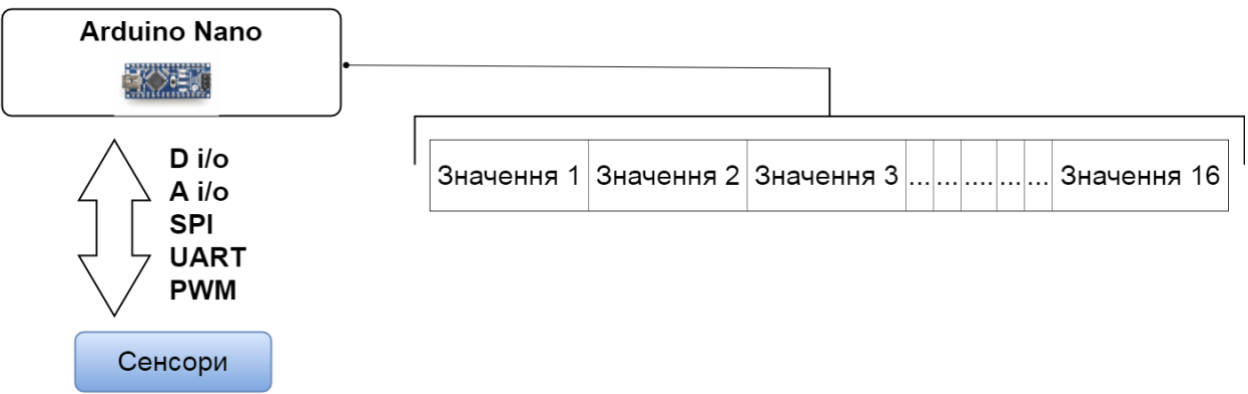


Рисунок 2.11 – Схема взаємодії з платою Arduino

СДА робить опитування комірок кожної підсистеми та вивід у інтерфейс відображення даних (рис. 2.12).

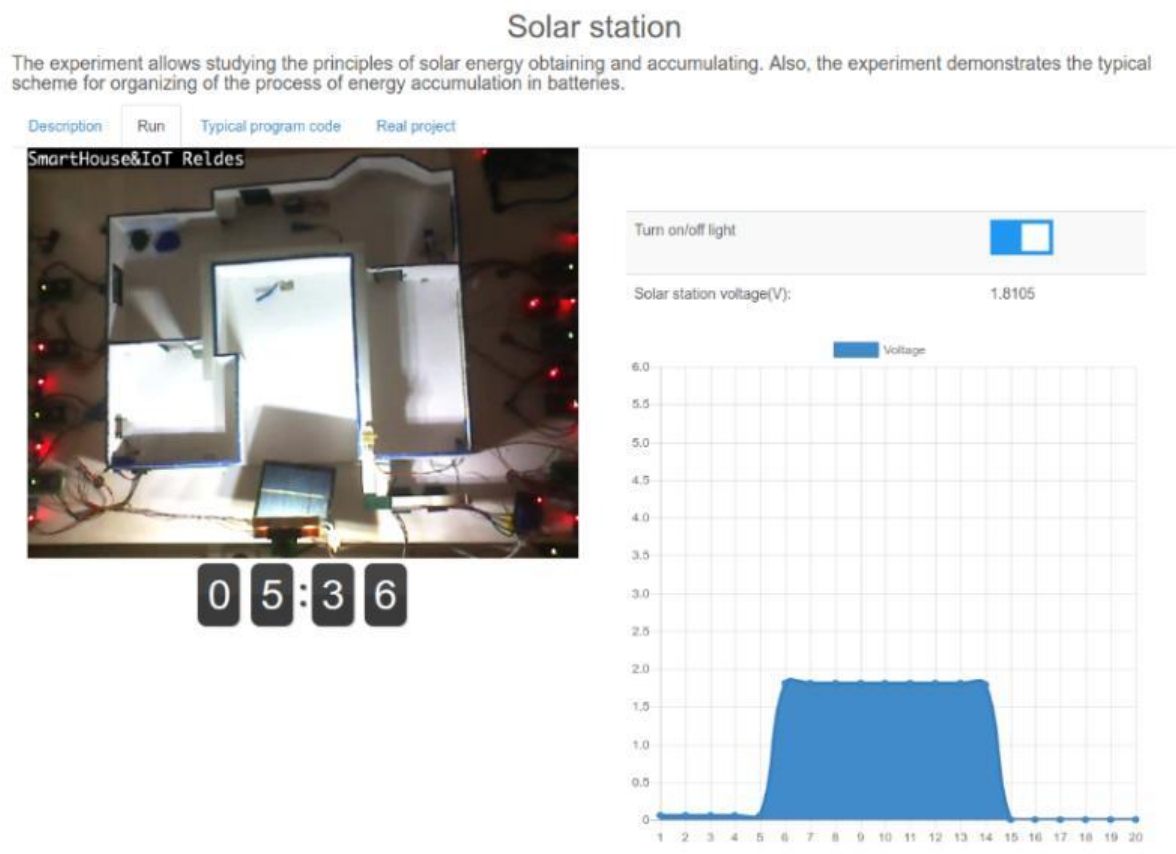


Рисунок 2.12 – Графічний інтерфейс експерименту ВЛ Smart House&IoT [31]

Для розробки керуючої програми для Arduino реалізовано спеціальний інтерфейс RELDES, що дозволяє вводити розроблений код програми та аналізувати помилки (рис. 2.13) [32].

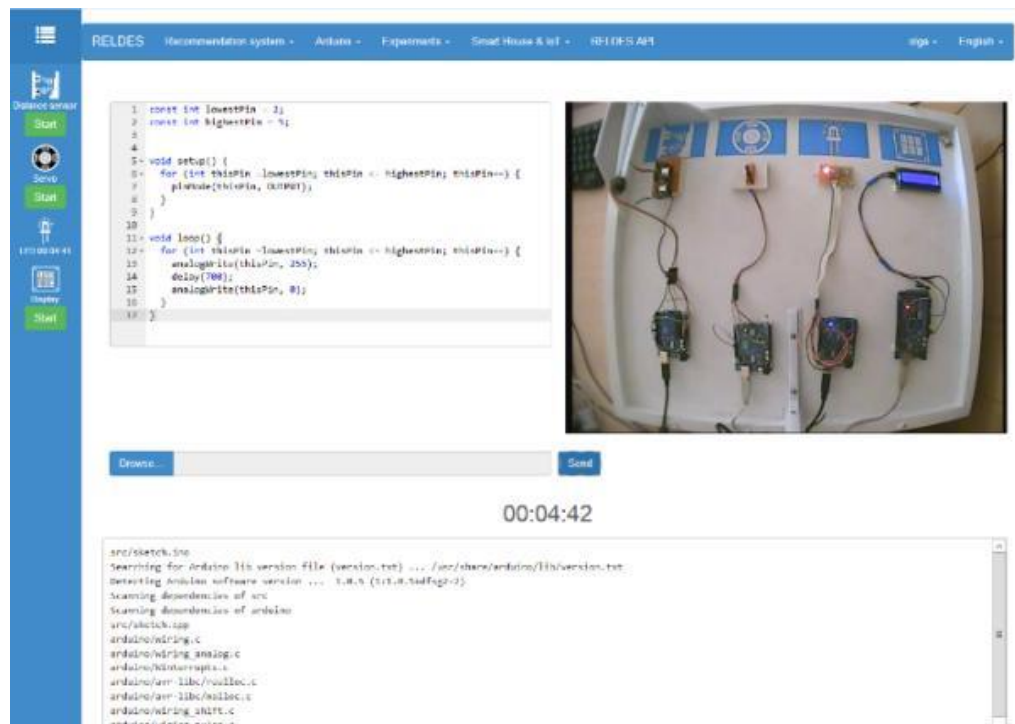


Рисунок 2.13 – Графічний інтерфейс експерименту ВЛ RELDES [32]

Діаграму діяльності користувача з лабораторією можна представити як наведено на рисунку 2.14.

Користувач на початку роботи виконує вибір методу навчання – теоретична або практична складова. При цьому йому надається можливість вивчення опису експерименту, схеми підключення компонентів, типового коду програми та рекомендацій щодо реалізації реального проекту [21].

Якщо студент обирає практичну складову, засобами віддаленої лабораторії RELDES виконується авторизація, резервування експерименту та старт експерименту (якщо експеримент зайнятий, користувач потрапляє у чергу на виконання). Для кожного експерименту виділяється певний час на виконання та пропонується декілька сценаріїв проведення [21].

Після вибору сценарію виконується запит щодо параметрів сенсорів та станів контрольно-виконавчих пристроїв. Засобами лабораторії RELDES формується запит, який перевіряється та відправляється до лабораторії Smart House&IoT, де відбувається виконання запиту та формування результату [21].

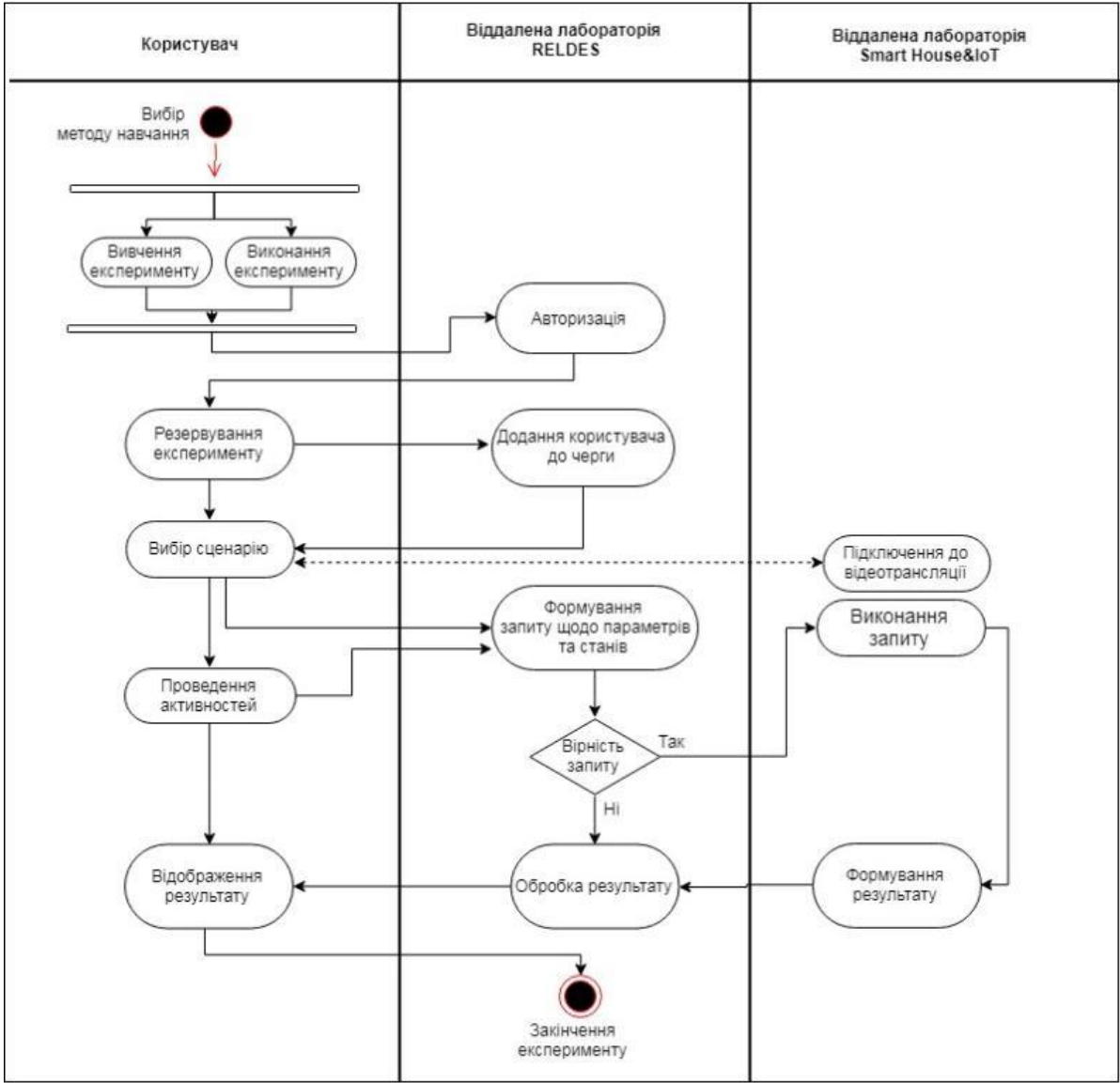


Рисунок 2.14 – Діаграма діяльності користувача з ВЛ [21]

Програмне забезпечення лабораторії RELDES виконує обробку отриманих результатів та відображення результатів користувачу у наочному вигляді на сторінці експерименту. Користувач може також змінювати стани контрольно-виконавчих пристроїв з метою відпрацювання алгоритмів керування системою [21].

Результати порівняльного аналізу прототипів СДА, побудованих на основі демонстраційних стендів та віддалених лабораторій, наведено в таблиці 2.1.

Таблиця 2.1 – Результати аналізу прототипів СДА

№ п/п	Назва стенду/лабораторії	Розробник	Режим доступу	Основне обладнання	Вартість
1	GAMMA Training Kit GTK 5.1	Siemens	офлайн	Siemens	+++ +
2	KNX-EIB Experimenter with Simulator	Wuekro	офлайн	Siemens	+++ +
3	Демостенд MyHOME	ПКФ МОТОР, Запоріжжя	офлайн	Legrand	+++
4	Демостенд GIRA	GIRA	офлайн	GIRA	+++
5	Макет Розумного будинку	Intel Corp.	офлайн	Intel Corp., Arduino	+++
6	ВЛ голосового управління	Корнельський університет, США	онлайн	ATmega	++
7	Smart House & IoT	НУ «Запорізька політехніка»	онлайн	Arduino, Raspberry	++
8	RGB Led laboratory & Servomotor	Університет дистанційної освіти, Мадрид	онлайн	Arduino	++

2.2 Аналіз структурних та функціональних особливостей апаратно-програмної платформи Arduino

На сьогоднішній день при розробці СДА та їх прототипів використовується безліч платформ [33] - [35], зокрема, Arduino (Mega, Uno, Leonardo, Mini, Yun та інші) та Raspberry Pi. Є ціла низка традиційних порівняльних факторів, які використовуються при виборі платформ. До них відносяться:

– параметри контролера (швидкість, енергоспоживання, периферійні пристрої на кристалі);

- доступні функції (в т.ч. підтримувані протоколи зв'язку, пам'ять та ін.);
- периферійні пристрої, що вбудовані на платформі;
- технічна підтримка (набір інструментів, необхідний для створення ПЗ, готові рішення).

Arduino Software – торгова марка апаратно-програмних засобів для побудови простих систем автоматики і робототехніки, орієнтована на непрофесійних користувачів. Програмна частина складається з безкоштовної програмної оболонки (Integrated Development Environment, IDE) для написання програм, їх компіляції та програмування апаратури. Апаратна частина являє собою набір змонтованих друкованих плат, що продаються як офіційним виробником, так і сторонніми виробниками.

МК для Arduino відрізняються наявністю попередньо прошитого в них завантажувача (bootloader). За допомогою цього завантажувача користувач завантажує свою програму в МК без використання традиційних окремих апаратних програматорів. Завантажувач з'єднується з комп'ютером через інтерфейс USB (якщо він є на платі) або за допомогою окремого перехідника UART-USB. Підтримка завантажувача вбудована в Arduino IDE і виконується в один клік мишею [36].

На випадок затирання завантажувача або покупки МК без завантажувача розробники надають можливість прошити завантажувач в МК самостійно. Для цього в Arduino IDE вбудована підтримка декількох популярних дешевих програматорів, а більшість плат Arduino має штирковий роз'єм для внутрисхемного програмування (ICSP для AVR, JTAG для ARM).

У лінійці пристроїв Arduino (рис. 2.15 - 2.18) в основному застосовуються МК Atmel AVR ATmega328, ATmega168, ATmega2560, ATmega32U4, ATTiny85 з частотою тактування 16 або 8 МГц. У старих виробках застосовувалися ATmega8, ATmega1280 і інші. Є також плати на процесорі ARM Cortex M. Сторонні розробники портували в Arduino підтримку популярного Wi-Fi МК ESP8266. Тепер компілювати і

завантажувати прошивку для ESP8266 зі своїми скетчами і підтримкою Wi-Fi можна прямо з Arduino IDE, отримуючи одноплатну схему з підтримкою мережі Wi-Fi [37].

В рамках співпраці зі сторонніми виробниками в Arduino IDE була включена підтримка деяких апаратних засобів Intel x86. Intel Galileo (процесор Intel Quark X1000 400 МГц), Intel Edison і Arduino 101 - Arduino-сумісні плати на Intel x86 архітектурі.



Рисунок 2.15 – Arduino Due



Рисунок 2.16 – Arduino Uno



Рисунок 2.17 – Arduino Mega 2560



Рисунок 2.18 – Arduino Leonardo

Результати аналізу плат Arduino наведено в таблиці 2.2 [36]. Плати механічно та електрично сумісні з периферійними платами Arduino. Плати функціонують під власною ОС Linux, поверх якої працює застосунок, що дозволяє завантажувати та виконувати скетчі Arduino. Мова програмування Arduino є стандартним C++ (використовується компілятор AVR-GCC) з деякими особливостями, які полегшують новачкам написання першої працюючої програми [37].

Таблиця 2.2 – Результати порівняльного аналізу плат Arduino для прототипування СДА [36]

Плата	Мікропроцесор	Робоча напруга, В	Цифрові піни	Аналогові піни	Живлення на Вхід/Вихід, мА	Оперативна пам'ять, Кб	Кабель	Підтримка Інтернет мережі
Uno	ATmega328	5	14	6	20	32	USB A-B	-
Leonardo	ATmega32u4	5	20	12	40	32	micro-USB	-
Micro	ATmega32u4	5	20	12	40	32	micro-USB	-
Nano	ATmega328	5	22	8	40	32	mini-B USB	-
Mini	ATmega328	5	14	6	20	32	USB-Serial	-
Due	Amel SAM3X8E ARM Cortex-M3 CPU	3.3	54	12	800	512	micro-USB	-
Mega	ATmega2560	5	54	16	20	256	USB A-B	-
M0	Amel SAMD21	3.3	20	6	7	256	micro-USB	-
Yun Mini	ATmega32u4	3.3	20	12	40	32	micro-USB	Ethernet/ Wifi
Uno Ethernet	ATmega328p	5	20	6	20	32	Ethernet	Ethernet
Tian	Amel SAMD21	5	20	0	7	16000	micro-USB	Ethernet/ Wifi
Mega ADK	ATmega2560	5	54	16	40	256	USB A-B	-
M0 Pro	Amel SAMD21	3.3	20	6	7	256	micro-USB	-
Industrial 101	ATmega32u4	5	7	4	40	16000	micro-USB	Ethernet/ Wifi
Uno Wifi	ATmega328	5	20	6	20	32	USB A-B	Wifi
Leonardo Ethernet	ATmega32u4	5	20	12	40	32	USB A-B	Ethernet
MKR1000	Amel SAMD21	3.3	8	7	7	256	micro-USB	Wifi

Програми, написані для Arduino (скетчі), зберігаються в файлах з розширенням `.ino`. Ці файли перед компіляцією обробляються препроцесором Arduino. Також існує можливість створювати і підключати до проєкту стандартні файли `C++`. Обов'язкову в `C++` функцію `main()` препроцесор Arduino створює сам, вставляючи туди необхідні «чорнові» дії. Програміст повинен написати дві обов'язкові для Arduino функції `setup()` і `loop()`. Перша викликається одноразово при старті, друга виконується в нескінченному циклі. В текст своєї програми (скетч) програміст не зобов'язаний вставляти заголовки використовуваних стандартних бібліотек. Ці заголовки доповнює препроцесор Arduino відповідно до конфігурації проєкту. Однак призначені для користувача бібліотеки потрібно вказувати [36].

Як показали проведені дослідження, вибір потрібної плати для розробки з безлічі варіантів, доступних на ринку, що мають певну кількість функцій і специфікацій є складною задачею. Методику вибору можна представити у вигляді алгоритму (рис. 2.19) [33].

Етап 1. Початок роботи.

Етап 2. Аналіз проєкту, який потрібно побудувати, і необхідних для нього компонентів.

Етап 3. Визначення правильного МК з необхідною кількістю контактів, портів вводу/виводу, інтерфейсом та іншими функціями, такими як вхідне живлення, швидкість, пам'ять тощо.

Етап 4. Створення списку плат розробників, які відповідають вимогам проєкту.

Етап 5. Визначення наявності інтегрованого середовища розробки та інструментів розробки, необхідних для перерахованих плат.

Етап 6. Перевка наявності відповідних симуляторів та компіляторів, оскільки деякі плати не дозволяють компіляцію GCC.

Етап 7. Перевірка наявності всіх засобів розробки.

Етап 8. Визначення плати відповідно до бюджету проєкту.

Етап 9. Кінець роботи.



Рисунок 2.19 – Схема алгоритму методики вибору плати для прототипу СДА

Таким чином, є ціла низка традиційних порівняльних факторів, які використовуються при виборі платформ. Для ефективного пошуку платформи для розробки прототипу СДА описано методику вибору, що враховує функціональні та фінансові вимоги до проєкту.

2.3 Огляд засобів розробки програмного забезпечення для Arduino

Оскільки одним з найбільш використовуваних компонентів прототипів СДА є плата Arduino, було проведено дослідження середовищ розробки саме для цієї платформи.

Найбільш популярними середовищами розробки програм для Arduino є Arduino IDE, Programino, B4R, CodeBlocks for Arduino, Visuino, S4A, ArduBlock та інші [38] - [39].

Arduino IDE – інтегроване середовище розробки Arduino, що надає текстовий редактор, менеджер проєктів, препроцесор, компілятор та інструменти для завантаження програми в МК [40]. Оболонка написана на Java на основі проєкту Processing, працює під Windows, Mac OS X і Linux (рис. 2.20). На відміну від онлайн-версії редактора коду (Arduino Web Editor), настільною версією можна користуватися за відсутності зв'язку з Інтернету [40].

Переваги Arduino IDE:

- доступність;
- зручний для використання та розуміння інтерфейс;
- програма сумісна з усіма версіями ОС Windows;
- наявність необхідних для роботи інструментів;
- кілька варіантів мов програмування;
- можливість поглибити знання мови C++;
- вбудований набір прикладів програм;
- функції збереження, експорту, перевірки, пошуку, заміни скетчів [40].

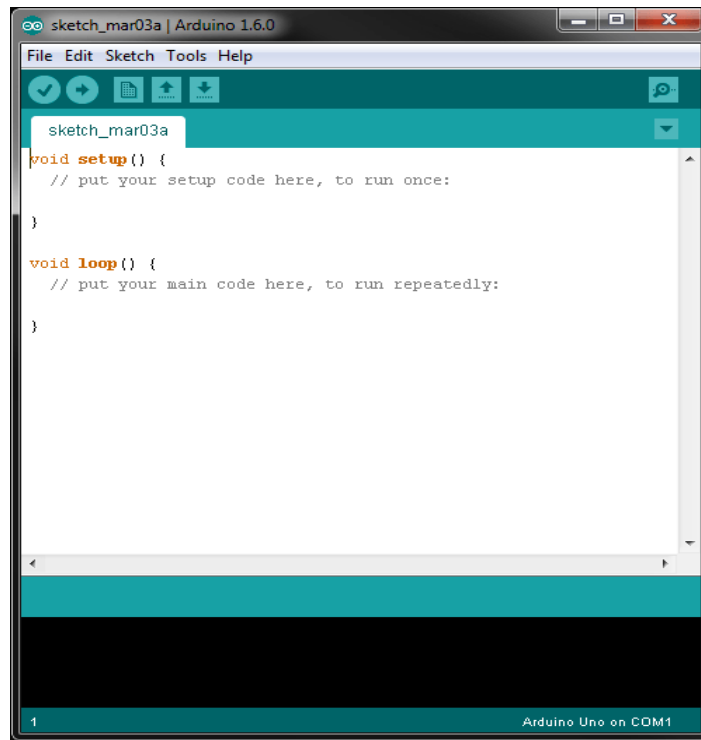


Рисунок 2.20 – Середовище розробки Arduino IDE

Недоліки Arduino IDE:

- програма не розрахована на роботу зі складними системами;
- робота окремих версій, за словами користувачів, характеризується нестабільністю [40].

Programino – це платне середовище розробки, але є можливість використовувати його 14 днів безкоштовно. Programino, як і деякі інші середовища розробки, вимагає попереднього встановлення Arduino IDE [38].

Мова, що використовується в Programino, така ж, як і в оригінальному Arduino IDE, що є великим плюсом цього середовища розробки.

Це IDE пропонує такий зручний спосіб швидкого розвитку як авто-addability коду. Тобто, непотрібно звертатися до довідників з описом команд і методів Arduino. У цьому середовищі розробки є кілька додаткових корисних інструментів, доступних у меню "Tools": блокнот, дизайнер LCD-символів, конвертер між DEC-BIN-HEX, термінал послідовного порту, аналоговий плотер та інші [38].

Інструмент Analog Plotter дозволяє візуалізувати те, що приходить в COM-порт від Arduino (рис. 2.21). Він може бути корисним, наприклад, для відображення показань під час роботи деяких аналогових датчиків: температури, вологості, тиску, освітлення та інших [38].

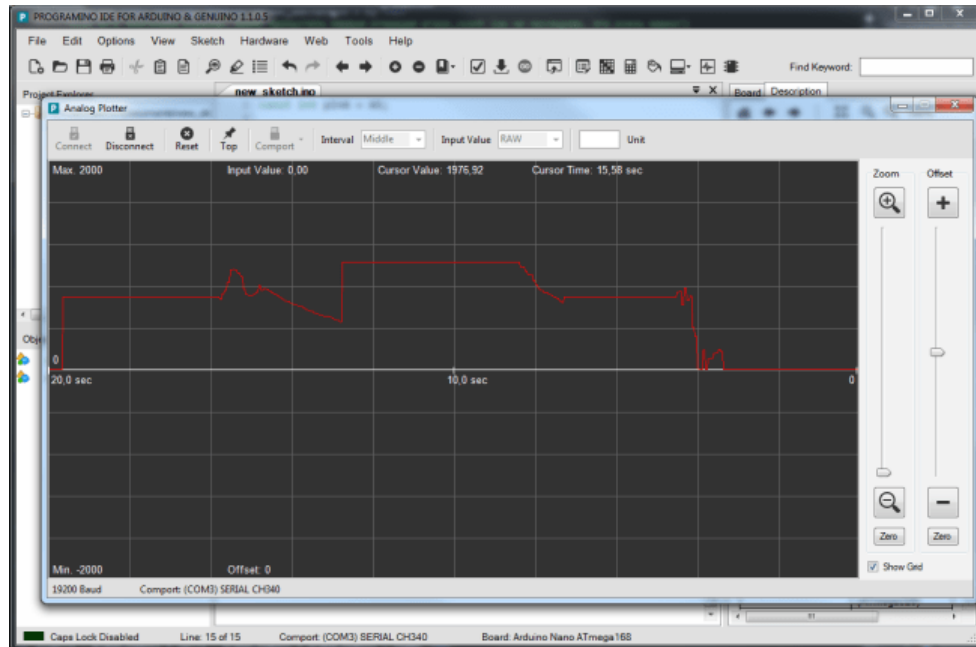


Рисунок 2.21 – Аналоговий плоттер в Programino [38]

B4R, або Basic for Arduino – це середовище розробки використовує мову Basic, замість C. Воно також підтримує функцію автодоповнення коду. Крім того, воно повністю безкоштовне [38]. При першому запуску середовище B4R також вимагає вказати шлях до директорії з Arduino IDE і додаткових нестандартних бібліотек і загальних модулів. Ці налаштування можна задати і пізніше через меню Tools/Configure, а також вибрати плату: Tools Board Selector (рис.2.22).

В даному середовищі розробки можна ставити точки зупину, що корисно в процесі налагодження, а також використовувати закладки для більш швидкої навігації по коду. Середовище використовує більш об'єктно-орієнтовану мову, ніж класична Arduino IDE, з іншим синтаксисом, тому, не можна одразу почати програмувати в ньому. Проте, зручність цього

середовища і наявність хорошого керівництва від розробників повністю окупає ці недоліки [38].

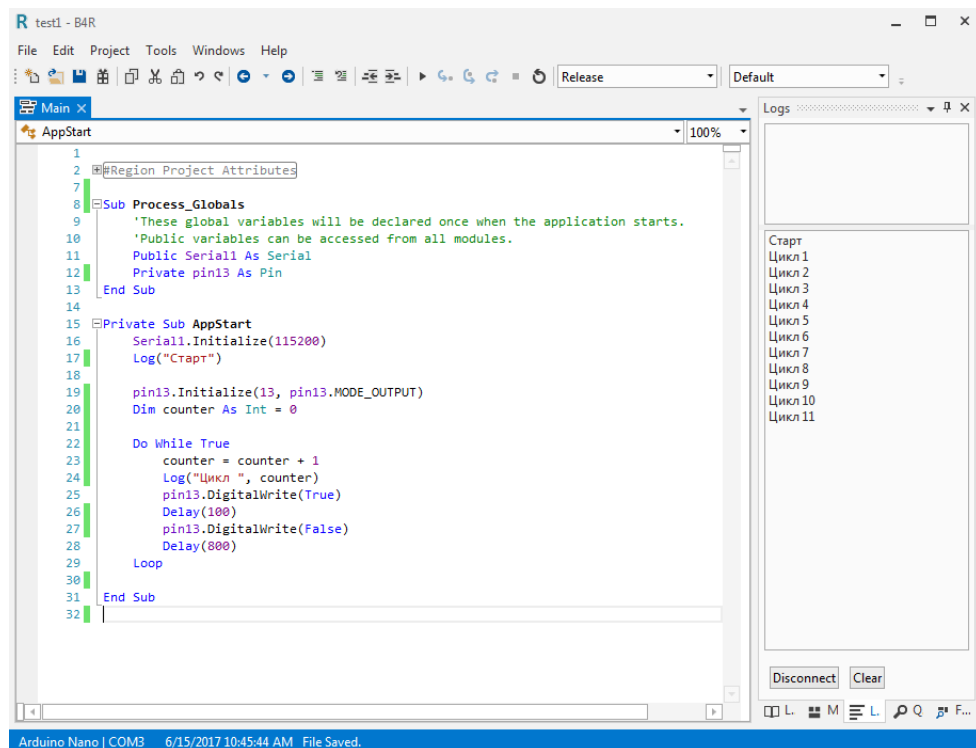


Рисунок 2.22 – Середовище розробки B4R [38]

CodeBlocks – середовище, що надає можливість писати код для МК і деяких інших платформ, а не тільки для Arduino [38]. На рисунку 2.23 зображено зовнішній вигляд Codeblocks for Arduino.

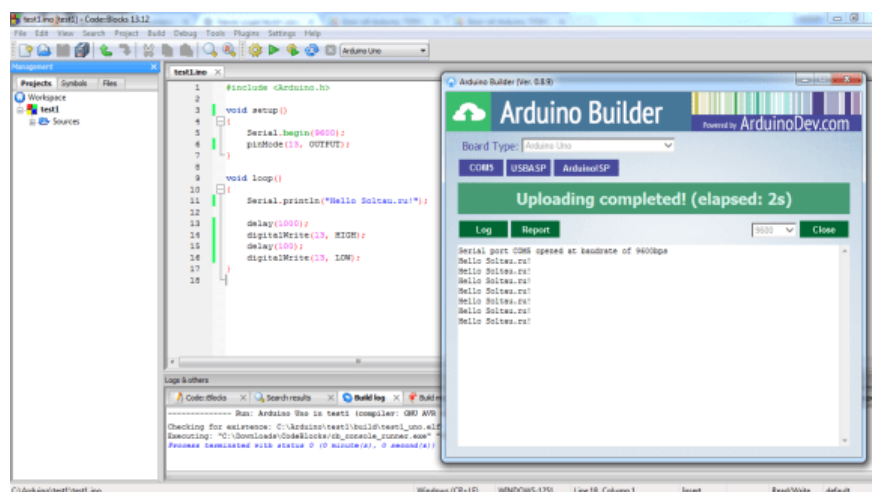


Рисунок 2.23 – Середовище розробки Codeblocks for Arduino [38]

Visuino – це безкоштовне графічне середовище, що працює на базі сумісних з Arduino промислових логічних контролерів (ПЛК) Controllino (рис. 2.24). Воно дає можливість створення складних систем автоматизації та IoT рішень, причому зробити це можна, просто переміщуючи та з'єднуючи візуальні блоки. Програмне середовище автоматично створює код для промислових контролерів [39].

До переваг Visuino відноситься великий набір компонентів для математичних та логічних функцій, сервоприводів, дисплеїв, Інтернету та ін. Коли ПЛК запрограмований, графічне середовище нагадує доступний спосіб підключення до контролера. Це може бути послідовний порт, Ethernet, Wi-Fi або GSM [39].

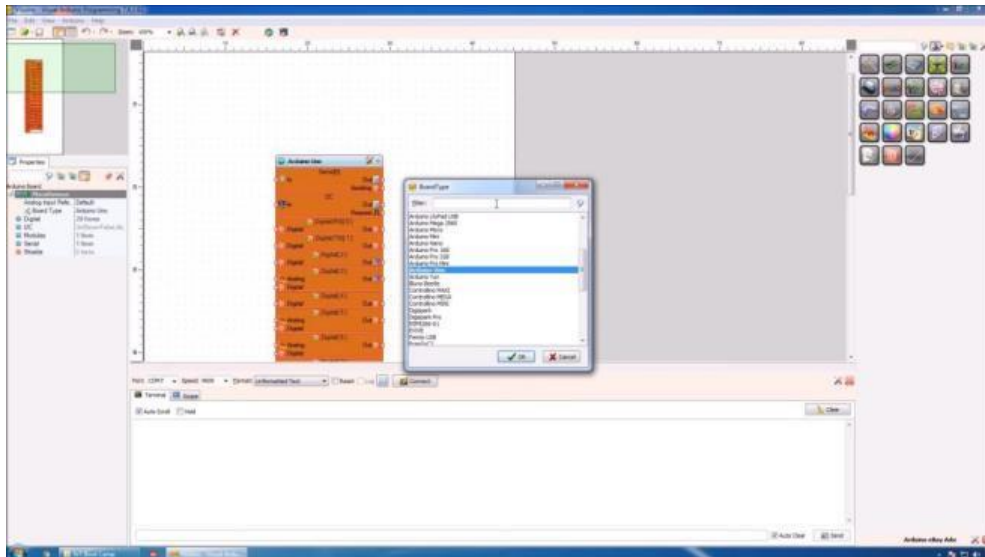


Рисунок 2.24 – Середовище розробки Visuino [39]

Scratch для Arduino (скорочено – S4A) – це об'єктно-орієнтоване середовище, в якому програми збираються з деталей, які можна переміщати, видозмінювати, змушувати взаємодіяти по-різному, що дозволяє програмувати Arduino (рис. 2.25). Основа Scratch – блоки команд, таких як сенсори, змінні, рух, звук, оператори, зовнішність, перо, контроль та інше. До плюсів системи відноситься те, що вона русифікована і повністю локалізована [39]. До недоліків – те, що Scratch не вміє прошивати Arduino, а лише керує МК через USB.

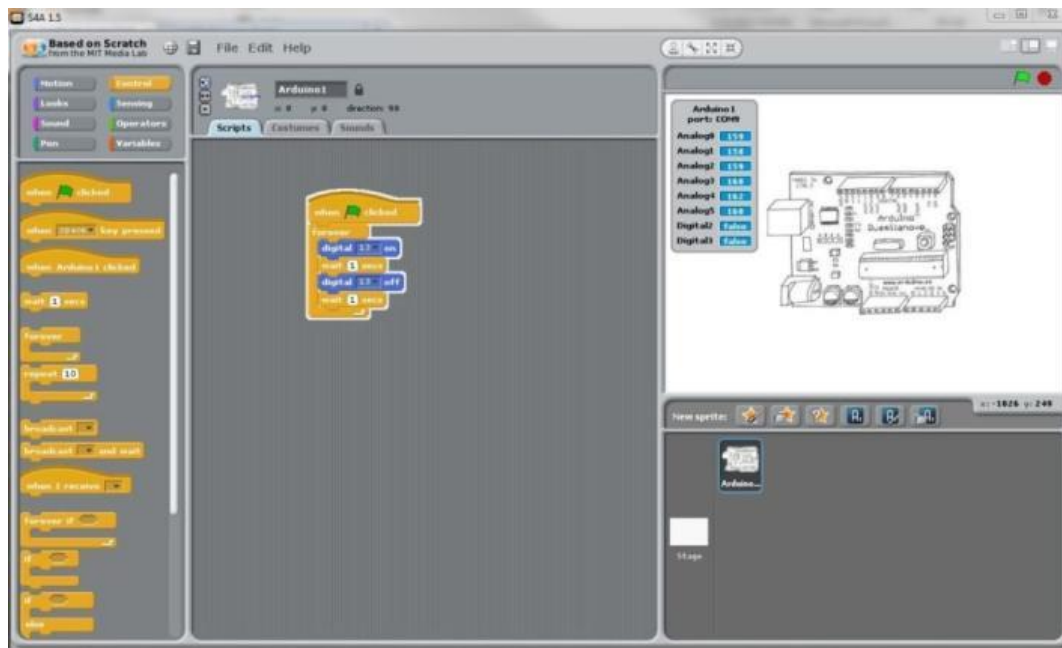


Рисунок 2.25 – Середовище розробки S4A [39]

ArduBlock – це графічне середовище і мова програмування для Arduino, призначений для легкого старту [39]. Середовище дуже просте у використанні та набрало велику популярність на інших платформах (рис. 2.26).



Рисунок 2.26 – Графічне середовище розробки ArduBlock [39]

FLProg – альтернативне середовище програмування для Arduino [41]. Програма FLProg дозволяє створювати прошивки для плат Arduino за допомогою графічних мов FBD і LAD, які є стандартом в області програмування промислових контролерів (рис. 2.27).

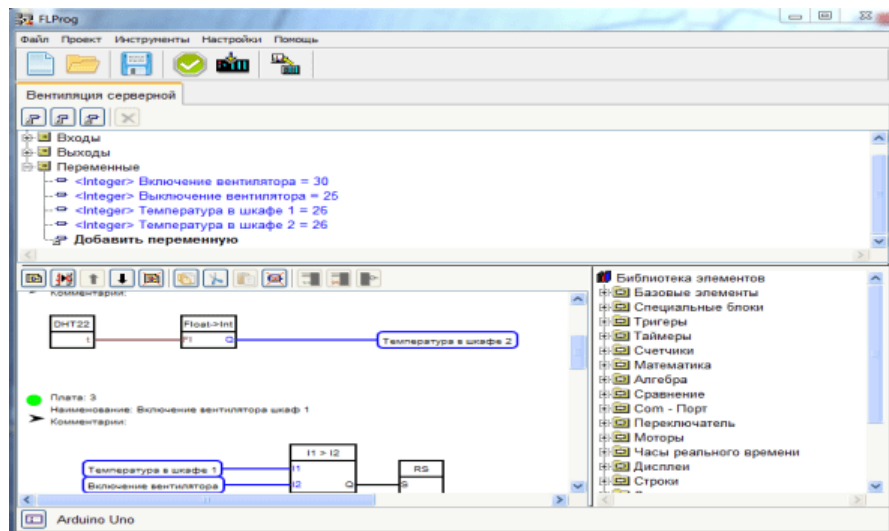


Рисунок 2.27 – Середовище програмування FLProg [41]

Avr-gcc – програмний засіб призначений для програмування плат, використовуючи ОС Linux. Використовується разом з утилітою AvrDude (рис. 2.28), що необхідна для завантаження прошивки у контролер. Перевагою програми є менший розмір скомпільованого коду, порівняно з Arduino IDE [42].

```

firmware — mc — 80×17
bash-3.2$ make
avr-gcc -Wall -Os -DF_CPU=16000000 -mmcu=atmega168 -c main.c -o main.o
avr-gcc -Wall -Os -DF_CPU=16000000 -mmcu=atmega168 -o main.elf main.o
rm -f main.hex
avr-objcopy -j .text -j .data -O ihex main.elf main.hex
avr-size --format=avr --mcu=atmega168 main.elf
AVR Memory Usage
-----
Device: atmega168

Program:  180 bytes (1.1% Full)
(.text + .data + .bootloader)

Data:      0 bytes (0.0% Full)
(.data + .bss + .noinit)

```

Рисунок 2.28 – Утиліта AvrDude

Arduino IDE для Microsoft Visual Studio і Atmel Studio (рис. 2.29) є вільним, повністю сумісним з програмуванням Arduino плагін для Microsoft Visual Studio і Atmel Studio. Використовує ті ж бібліотеки, вихідний код і інструменти розробки, що й Arduino IDE. Різниця полягає у забезпеченні більш професійного середовища розробки [43].

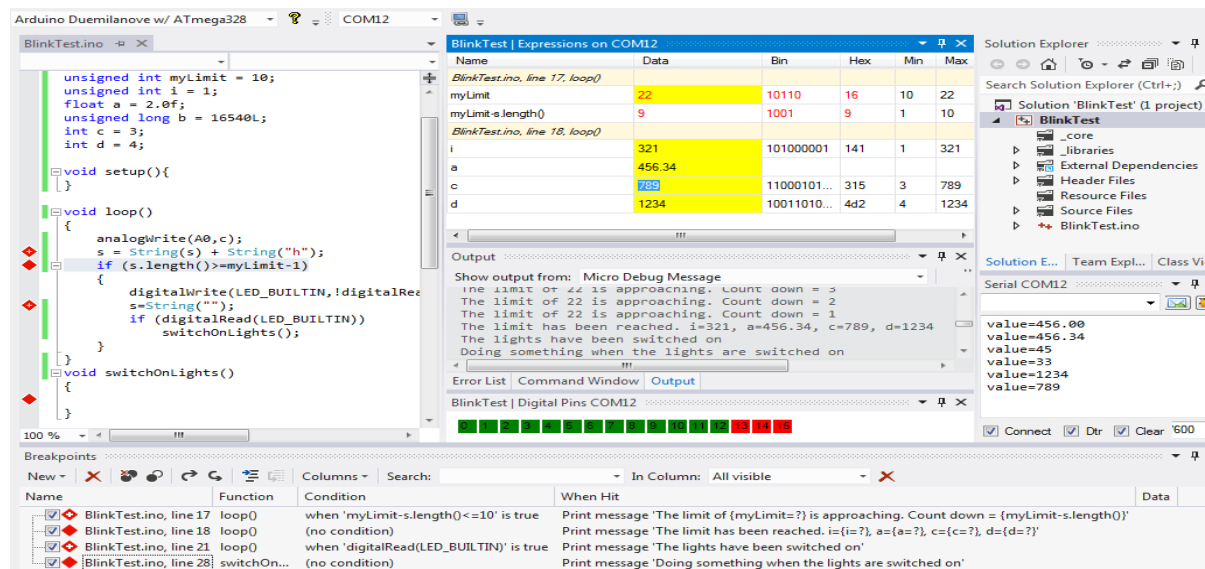


Рисунок 2.29 – Arduino IDE для Microsoft Visual Studio і Atmel Studio

Результати порівняльного аналізу деяких розглянутих середовищ розробки представлено в таблиці 2.3.

Таблиця 2.3 – Порівняльна характеристика популярних безкоштовних засобів розробки для платформи Arduino [38]

Характеристики	Середовище розробки			
	Arduino IDE	B4R	CodeBlocks	Processing
Простота	+	-	-	+
Підтримка	+	+/--	+/--	+
Мова програмування	C / Wiring	Basic	C	Processing, /Java
Доступність	Open-source	Open-source	Open-source	Open-source

В результаті проведених досліджень, для написання, відлагодження та завантаження коду може бути рекомендовано Arduino IDE, що являє собою сучасне повнофункціональне середовище розробки з сучасним інтерфейсом, мульти-панеллю IDE з інтегрованим представленням файлової системи, та призначене для розробки великих багатофайлових проєктів на основі репозиторію (інтеграція з Git) [40].

2.4 Дослідження патернів розробки графічного інтерфейсу Processing/Arduino

Processing — відкрита мова програмування, заснована на Java. Представляє собою легкий і швидкий інструментарій для створення візуалізацій за допомогою платформи Java Virtual Machine та дозволяє швидко створювати інтерактивні програми відображення зображень та анімації [44].

Особливості Processing:

- проста мова програмування для створення інтерактивних візуальних ефектів;
- базується на графічних можливостях мови Java та використовує її спрощений синтаксис;
- легкий та швидкий інструментарій для людей, які бажають програмувати зображення, анімацію та інтерфейси;
- створено вивчення основ комп'ютерного програмування у візуальному контексті;
- це відкритий проєкт, він абсолютно безкоштовний, а вихідний код є доступним для всіх бажаючих [44].

Processing та Arduino є спорідненими системами зі схожою ідеологією та реалізацією. Arduino використовує схожий, хоча і дещо урізаний у порівнянні з Processing синтаксис, лише орієнтований на роботу з МК. Офіційно мова програмування Arduino називається Wiring [45].

Мова Arduino (на основі Wiring) реалізована в C/C++, і тому має деякі відмінності від мови Processing, яка базується на Java [46]. Основні відмінності наведено в таблиці 2.4.

Таблиця 2.4 – Основні відмінності мовних конструкцій Arduino та Processing [46]

Мовна конструкція	Arduino	Processing
Масиви	<code>int bar[8];</code> <code>bar[0] = 1;</code>	<code>int[] bar = new int[8];</code> <code>bar[0] = 1;</code>
	<code>int foo[] = { 0, 1, 2 };</code>	<code>int foo[] = { 0, 1, 2 };</code> <i>or</i> <code>int[] foo = { 0, 1, 2 };</code>
Цикли	<code>int i;</code> <code>for (i = 0; i < 5; i++) { ... }</code>	<code>for (int i = 0; i < 5; i++) { ... }</code>
Виведення	<code>Serial.println("hello world");</code>	<code>println("hello world");</code>
	<code>int i = 5;</code> <code>Serial.println(i);</code>	<code>int i = 5;</code> <code>println(i);</code>
	<code>int i = 5;</code> <code>Serial.print("i = ");</code> <code>Serial.print(i);</code> <code>Serial.println();</code>	<code>int i = 5;</code> <code>println("i = " + i);</code>

Незважаючи на свою візуальну спеціалізацію, Processing має розвинені мережеві можливості і дозволяє отримувати інформацію з мережі, так і посилати її в мережу на різні ресурси та пристрої. А поєднання візуальних та мережевих можливостей Processing відкриває широкі перспективи побудови різних інтерактивних та аналітичних систем для різних сфер застосування [47].

Основні мережеві можливості Processing:

- робота в якості клієнта та підключення до інших систем на Processing;

- робота як вебсервер та обслуговування зовнішніх підключень;
- отримання даних та команд від контролерів у локальній мережі;
- створення візуальних інтерфейсів та мережеве управління підключеним обладнанням;

- візуалізація одержуваних через мережу даних;
- обмін даними з вебсайтами в Інтернеті;
- сніффінг та візуалізація мережевої активності [47].

Існує багато застосувань зв'язку між Processing та Arduino:

- управління МК з ПК;
- програмування МК;
- передача даних з МК на ПК;
- візуалізація даних і параметрів роботи як МК, так і ПК [48].

Зв'язати МК під управлінням Wiring та комп'ютерну програму під управлінням Processing можна декількома способами:

- по Serial;
- за допомогою спеціалізованого протоколу Firmata;
- по Ethernet мережі;
- різними бездротовими способами [48].

Варіант управління Arduino-контролером із скетчу на Processing може знадобитися для керування (за послідовним інтерфейсом) МК з ПК.

Оскільки взаємодія відбувається між двома системами, виконується розробка двох скетчів, які повинні працювати одночасно – один на контролері, інший – на комп'ютері.

В результаті, наприклад, за допомогою руху миші можливо керувати з комп'ютера включенням і вимкненням різних кольорів світлодіода на контролері Arduino (рис. 2.30) [44].

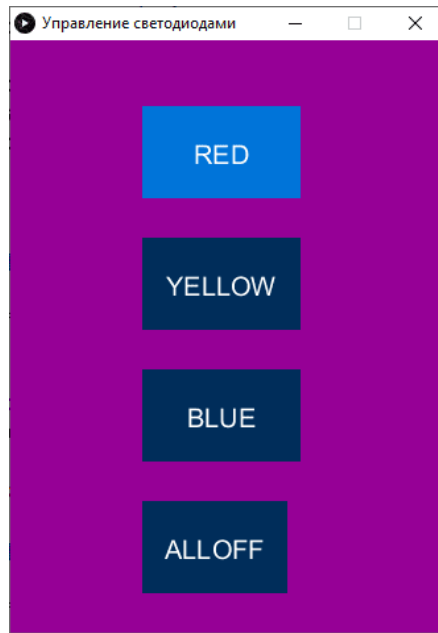


Рисунок 2.30 – Приклад 1 реалізації патерну Processing to Arduino

Можливо також реалізувати просте включення/виключення світлодіода (рис. 2.31). В результаті, коли вказівник миші знаходиться не над квадратом, квадрат виводиться сірим кольором, на контролер відправляється символ «L» і на контролері не горить світлодіод. Щойно тільки курсор миші з'являється над квадратом, квадрат стає червоним, на контролер посилається символ H і на платі загорається вбудований світлодіод [48].

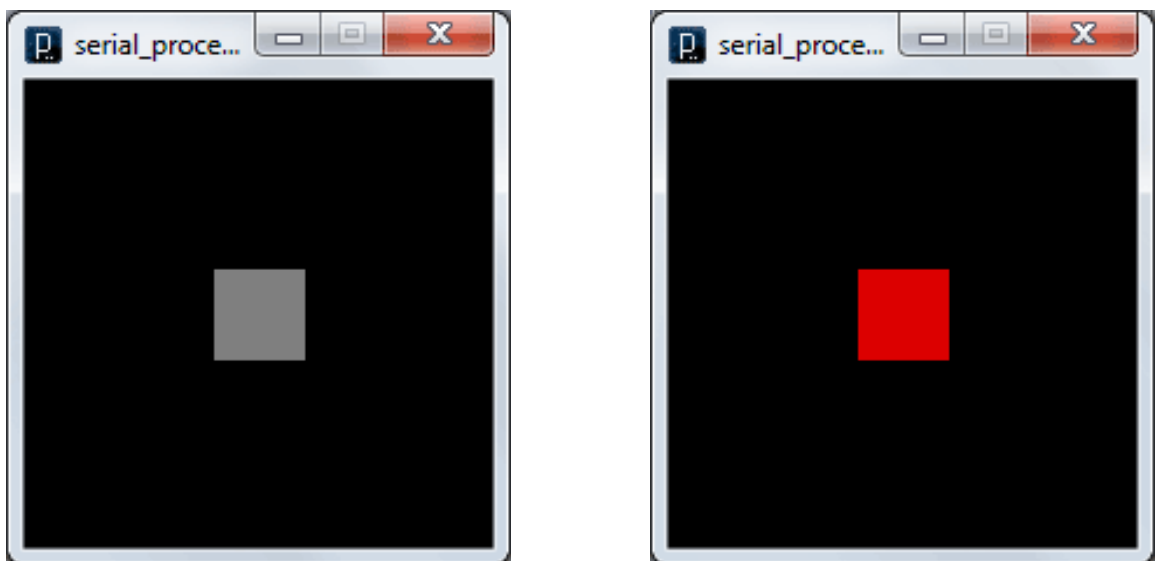


Рисунок 2.31 – Приклад 2 реалізації патерну Processing to Arduino

Arduino to Processing – протилежний варіант, що передбачає управління комп'ютером з контролера Arduino. Це теж дуже затребуваний патерн роботи системи, якому можна знайти безліч практичних застосувань [48].

Наприклад, контролер на Arduino передає керуючі байти (взяті ним із аналогового порту), а скетч на Processing використовує ці значення для встановлення яскравості фону графічного вікна (сірий). А потім передає нульовий байт та скетч на Processing відповідним чином змінює колір фону графічного вікна (чорний) (рис. 2.32).

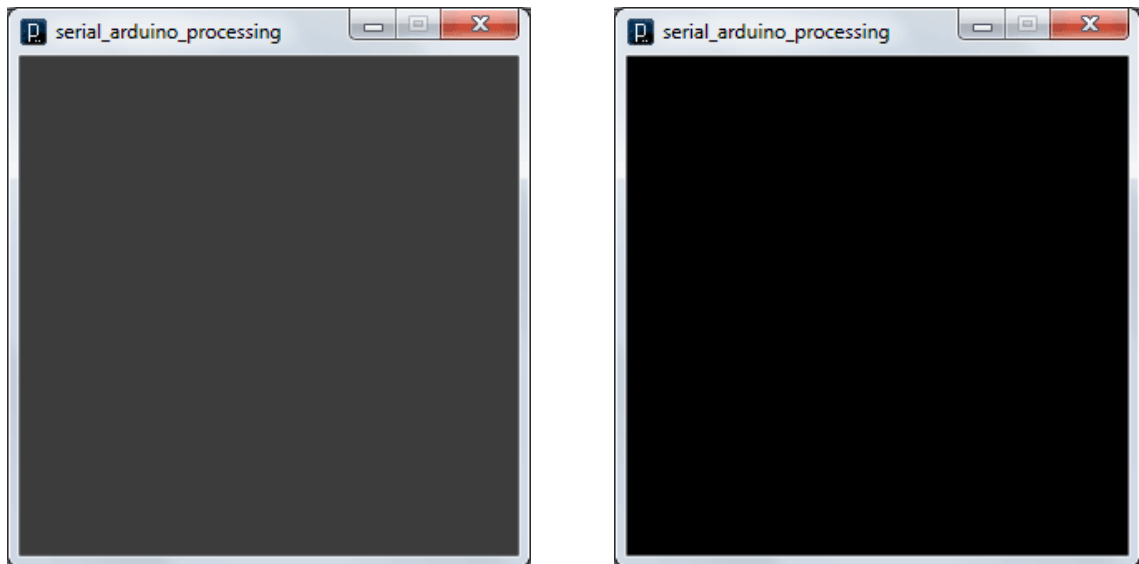


Рисунок 2.32 – Приклад реалізації патерну Arduino to Processing

2.5 Розробка графічного інтерфейсу взаємодії між Arduino та Processing для СДА

На основі описаних патернів, може бути реалізовано графічний інтерфейс взаємодії між контролером Arduino та актуаторами СДА, зокрема для прототипу підсистеми керування освітленням (вмикання/вимикання світлодіодного освітлення та керування жалюзі на основі серводвигунів).

RGB світлодіод підключимо до Arduino, як зображено на рисунку 2.33. Опір на кожному з резисторів становить 0.27 кОм.

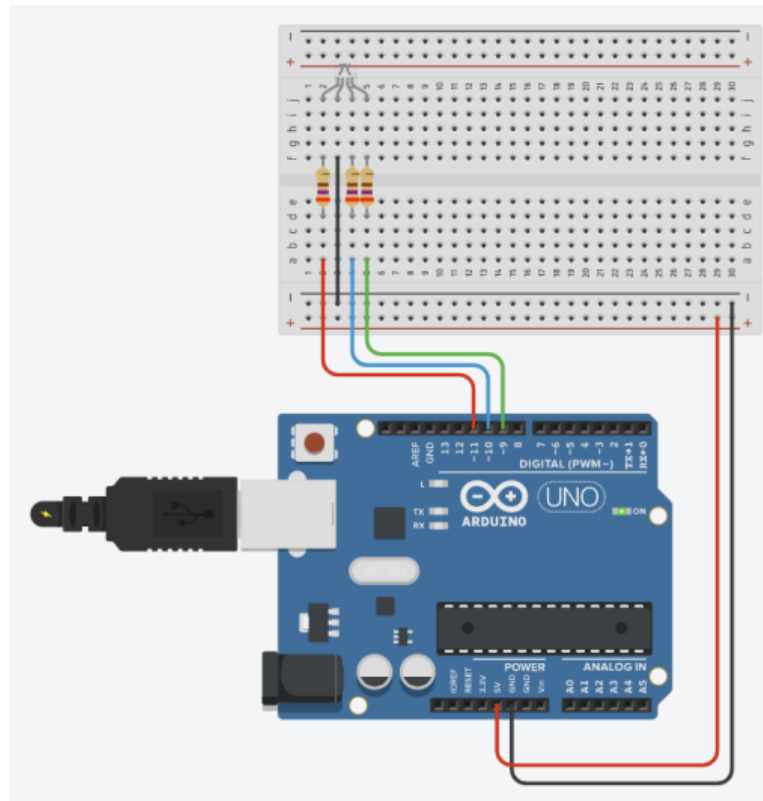


Рисунок 2.33 – Підключення RGB світлодіоду до Arduino

Програма для МК зчитує 3 цілих чисельних значення поспіль з послідовного порту, що відповідають кожному кольору RGB. Далі з урахуванням цих значень і загоряється світлодіод з потрібним кольором. Текст програми для Arduino наведено в Додатку А.1.

Створюваний графічний інтерфейс у Processing виглядає як вікно з трьома прямокутниками. Колір цих прямокутників різний та задається у змінній типу Color на початку програми, де може бути змінений. При наведенні курсору на прямокутник, його колір дещо змінюється (стандартний хід), а при натисканні – у монітор послідовного порту записується значення кольору відповідного прямокутника. Також значення кольору виводиться у консоль. Текст програми для Processing наведено в Додатку А.2.

Розроблене вікно зображене на рисунку 2.34. Стан вікна при наведенні на один з прямокутників зображено на рисунку 2.35. Результат послідовного натискання на кожен з прямокутників зображено на рисунку 2.36.

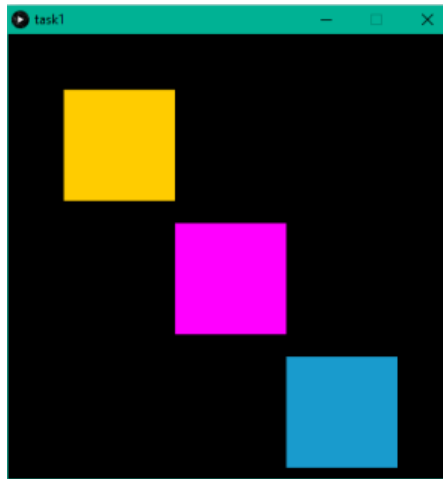


Рисунок 2. 34 – Розроблене у Processing вікно

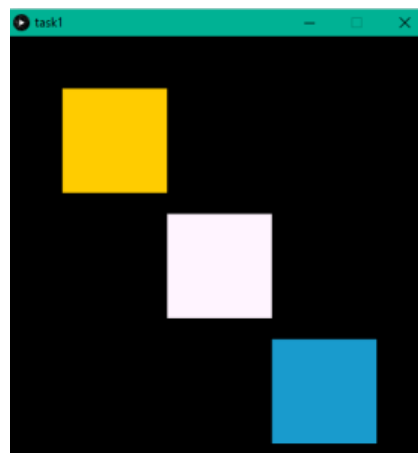


Рисунок 2.35 – Демонстрація роботи

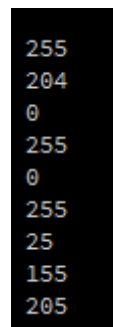


Рисунок 2.36 – Результат виведення значень кольору у монітор послідовного порту

Тепер вставимо отримані значення до монітору послідовного інтерфейсу у середовищі Tinkercad [49] - [51] (рис. 2.37). Модельований світлодіод світиться кольорами прямокутників з інтервалом 1 секунду (рис. 2.38).

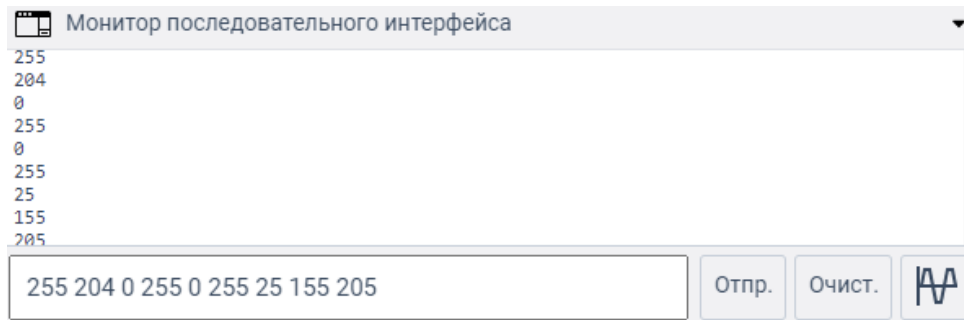


Рисунок 2.37 – Перевірка отриманих значень кольорів

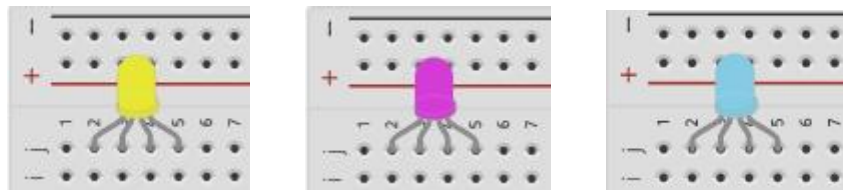


Рисунок 2.38 – Результат моделювання в Tinkercad

Для організації взаємодії між Processing та сервопривідом, під'єднаним до Arduino, створимо інтерфейс. Розробимо користувацький слайдер, положення якого на панелі, тобто його координата, буде позначати кут повороту сервоприводу. Отримаємо наступне вікно (рис. 2.39), в якому користувач сам може встановити кут повороту сервоприводу.

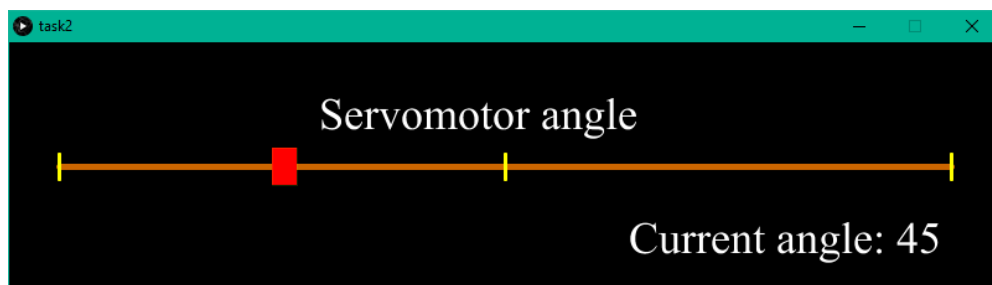


Рисунок 2.39 – Інтерфейс для повороту сервоприводу

Код програми наведено в Додатку А.3.

Паралельно, програма виводить у консоль кут повороту. Використаємо ці дані у Tinkercad (рис. 2.40) та перевіримо роботу на програмі з сервопривідом, що наведена у Додатку А.4.

Сервопривід рухається за вказаними координатами (рис. 2.41). Також, зберігаються інтервали часу між поворотами (рис. 2.42).

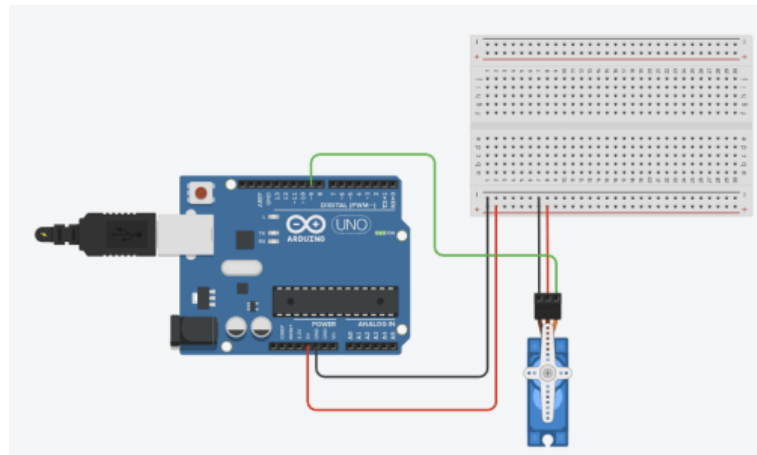


Рисунок 2.40 – Підключення сервоприводу до Arduino в Tinkercad

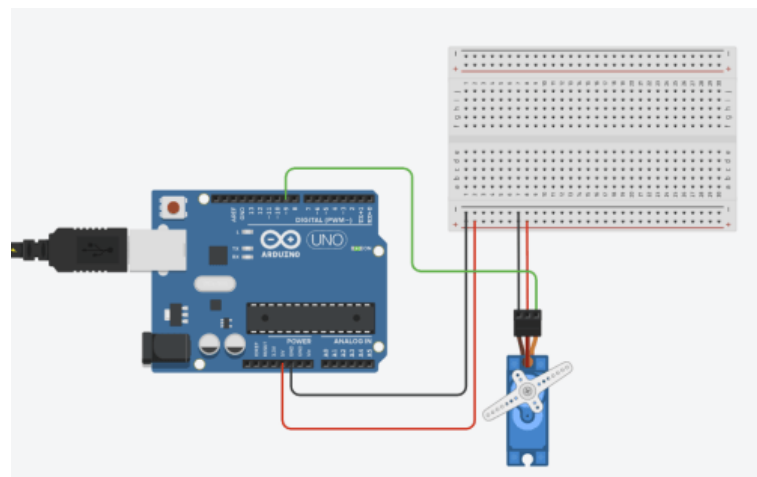


Рисунок 2.41 – Результат моделювання роботи серво в Tinkercad

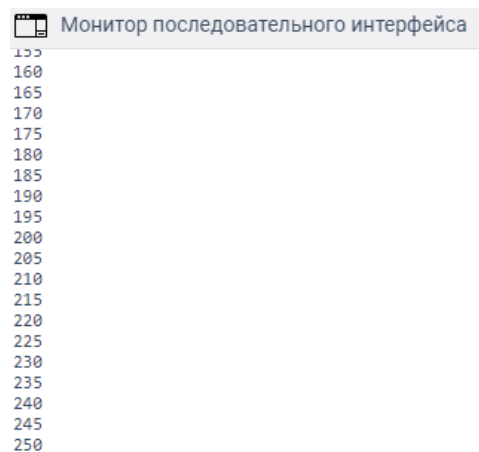


Рисунок 2.42 – Результат виведення значень у монітор послідовного порту

Для реалізації інтерфейсу у Processing за допомогою бібліотеки Firmata, необхідно завантажити до МК Arduino скрипт StandardFirmata, що містить саму бібліотеку. Для цього у Arduino IDE необхідно перейти Файл – Приклади – Firmata – StandardFirmata (рис. 2.43).

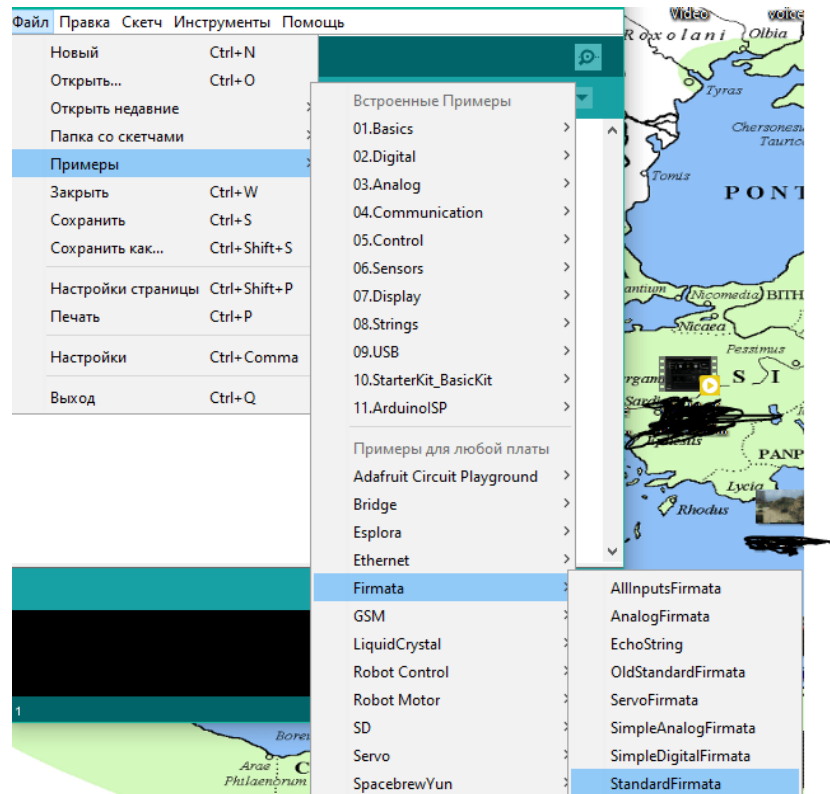


Рисунок 2.43 – Підключення сценарію Firmata

Також необхідно завантажити і бібліотеку Firmata до Processing, для використання її можливостей, як на рисунку 2.44.

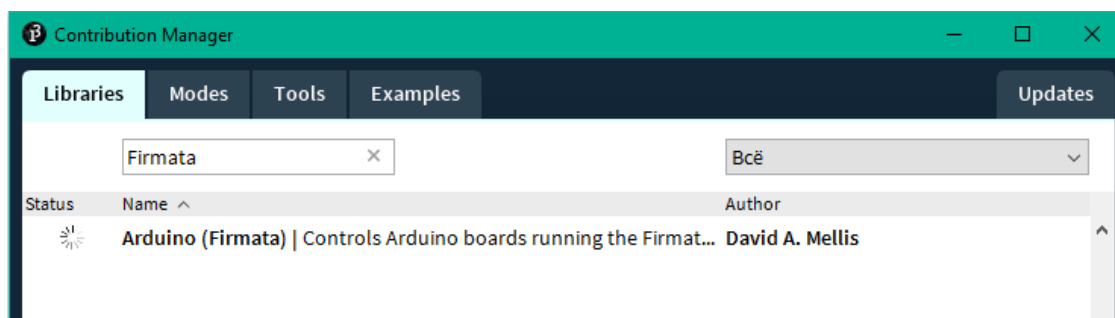


Рисунок 2.44 – Завантаження бібліотеки Firmata до Processing

Скористаємося розробленим «повзунком», представленим на рис.2.39. У якості ціни ділення візьмемо 1 відсоток, або 1 умовну одиницю. Будем зчитувати аналоговий сигнал потенціометру та приводити його до потрібного значення. У цьому допоможе функція `map`, код якої наведено в Додатку А.5.

Наприклад, для аналогового сигналу, як на рисунку 2.45, маємо наступне положення слайдеру, як на рисунку 2.46 (в інтервалі від 0 до 100 (0 до 1024) сигнал 266 займає 26 відсотків).

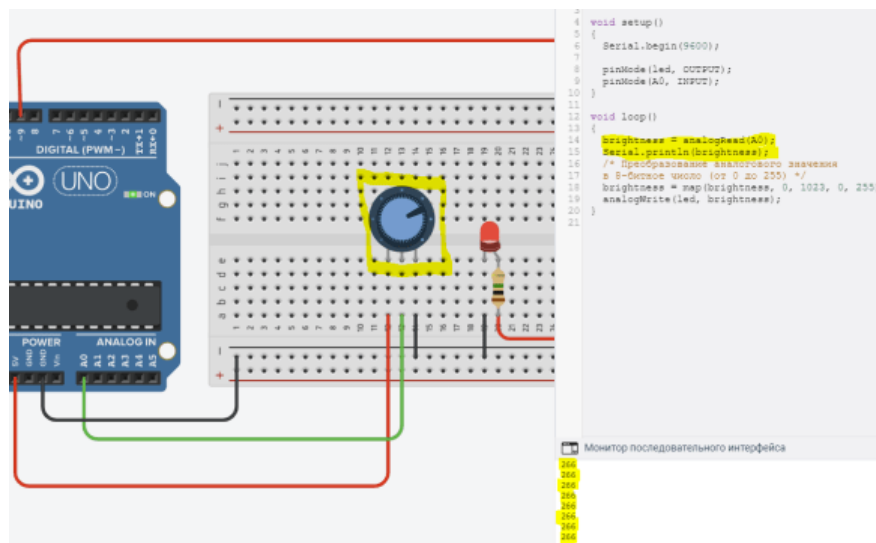


Рисунок 2.45 – Аналоговий сигнал потенціометра 266 одиниць

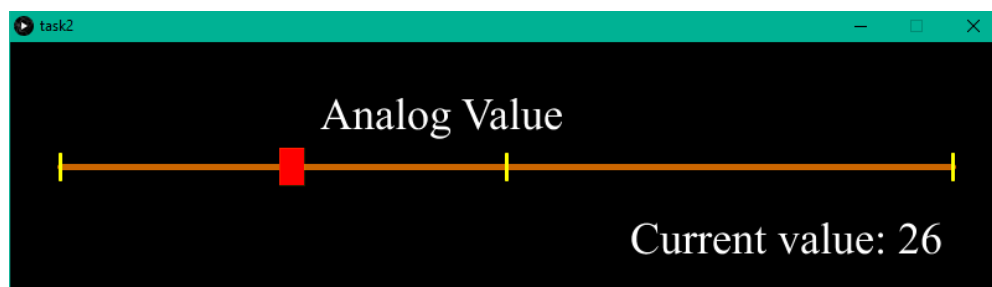


Рисунок 2.46 – Результуюче положення слайдеру

Для передачі інформації за допомогою Firmata можливо не використовувати слайдер, через те, що він має лише одну цифрову шкалу, коли, наприклад, для кольору потрібно усі 3. Тому замість повзунка було вирішено брати координати миші у вікні. При зміні положення курсору змінюється колір фону вікна, та у той же час змінюється колір RGB-

світлодіоду. Отримаємо результати як на рисунку 2.47. Текст програми наведено в Додатку А.6.

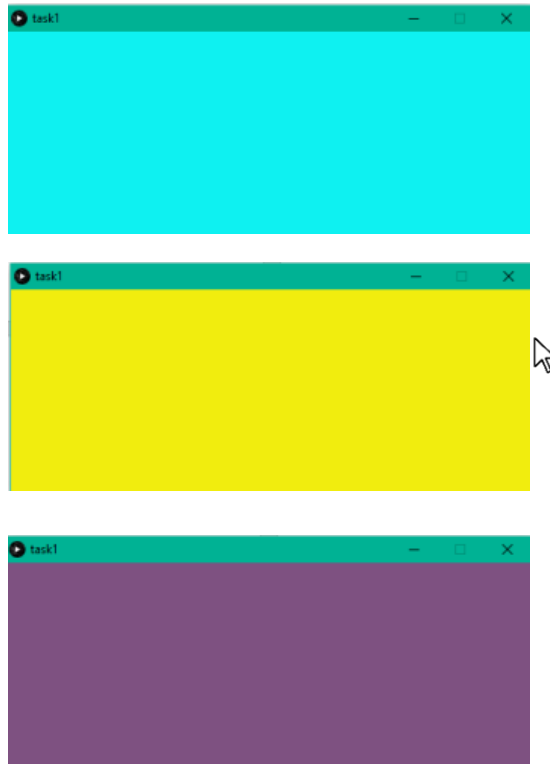


Рисунок 2.47 – Колір RGB-світлодіоду в залежності від положення курсору миші

2.6 Висновки до розділу 2

Як показали проведені дослідження, створення прототипів СДА на основі доступних апаратно-програмних платформ та середовищ розробки є актуальною задачею.

Апаратно-програмна платформа Arduino є однією з найбільш популярних та надає безліч можливостей. Використання патернів розробки графічного інтерфейсу в середовищі Processing дає можливість візуалізувати взаємодію Arduino to Processing та Processing to Arduino.

Розроблені інтерфейси можуть використовуватися для дослідження прототипів СДА на основі Arduino. Їх працездатність та коректність було перевірено шляхом моделювання в онлайн симуляторі Tinkercad.

З РОЗРОБКА ТА ДОСЛІДЖЕННЯ ПРОГРАМНОГО ТА АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ ПРОТОТИПУ СДА

3.1 Аналіз вимог до прототипу СДА

Проектований прототип СДА реалізує функції мережевої системи керування освітленням, контролю доступу та безпеки, моніторингу споживання енергії, вологості та температури. Він об'єднує три базові підсистеми: інтелектуальне освітлення, контроль доступу та віддаленого моніторингу. Моніторинг та керування здійснюються за допомогою застосунка, зв'язок якого з пристроями СДА є можливим на основі рішення Microsoft Azure IoT.

Специфікація проекту включає функціональні та нефункціональні вимоги до системи, а також вимоги до апаратного та програмного забезпечення. У таблиці 3.1 наведено функціональні вимоги до проекрованої системи, а у таблиці 3.2 – нефункціональні.

Таблиця 3.1 – Функціональні вимоги до прототипу СДА

№ п/п	Опис вимоги
1	2
1	Користувач повинен мати можливість контролювати температуру і показники вологості в будинку за допомогою клієнтського застосунка.
2	Користувач повинен мати можливість бачити стан світильників та входних дверей з можливістю контролювати їх за бажанням.
3	Користувачу потрібно буде використовувати логін Facebook для доступу до клієнтської програми для кожного входу.
4	Вебінтерфейс повинен забезпечувати представлення потоків даних, що існують усередині системи.
5	Клієнт повинен мати можливість налаштувати режими «Вдома», «Не вдома», «Сон» і «Пробудження» для пакетного керування домашніми пристроями.
6	Клієнт повинен мати можливість налаштувати та захистити свій пристрій у локальній мережі. На початку роботи відбувається авторизація на основі облікових даних користувача у Facebook та встановлюється зв'язок між клієнтською програмою та сервером.

Продовження таблиці 3.1

1	2
7	Мають бути передбачені повідомлення про проникнення сторонніх електронною поштою та SMS.
8	Домашні пристрої повинні контролюватися в будь-якому місці, де користувач має доступ до Інтернету, якщо домашній сервер також активний і підключений до Інтернет.
9	Передбачити альтернативний засіб керування, коли клієнтська програма недоступна або не бажана для використання.

Таблиця 3.2 – Нефункціональні вимоги до прототипу СДА

№ п/п	Опис вимоги
1	Інтерфейс клієнтського додатка повинен бути інтуїтивно зрозумілим та простим.
2	Слід передбачити можливість додавання компонентів для розширення функціональності.
3	Клієнтський застосунок має бути кросплатформним.
4	Оновлення пристроїв та зміни їх конфігурації мають бути реалізовані за допомогою технології хмара-пристрій.

Вимоги до апаратного та програмного забезпечення прототипу:

- основним контролером прототипу СДА має виступати мінікомп'ютер Raspberry Pi;
- в якості вторинних контролерів мають виступати плати Arduino;
- в якості хмарної платформи має використовуватися Microsoft Azure, що дає змогу представляти, тестувати, розгортати й керувати застосунками, системами та службами в хмарі та на периферії;
- прототип має інтегрувати три підсистеми: інтелектуальне освітлення, базова безпека доступу та віддалений моніторинг;
- мають бути реалізовані функції мережевої системи освітлення, контроль доступу безпеки, управління енергією, виявлення зломисників та звіт про вологість і температуру.

Розподілена СДА буде складатися з різних обчислювальних блоків (вузлів, кінцевих точок або пристроїв). Ці пристрої працюють разом для реалізації функцій автоматизації шляхом надсилання та отримання повідомлень. Пристрої об'єднуються в локальній мережі IP, мережі Bluetooth (BLE), мережі ZigBee, або можуть бути підключені через хмару [52]. Це дозволить вузловим пристроям працювати конструктивно та синхронізовано. Особливий акцент має бути зроблено на зниженні витрат на створення та дослідження прототипу.

Для створення прототипу має використовуватися 64-розрядний чотириядерний мінікомп'ютер Raspberry Pi 3 з тактовою частотою 1,2 ГГц, що працює під управлінням ОС Windows IoT [53], а також плати Arduino. Raspberry Pi виступатиме як домашній сервер на рівні, відповідальному за обробку, фільтрацію та завантаження даних пристроєм безпосередньо в хмару. Він також буде служити граничним пристроєм, який отримує керуючі сигнали з хмари та відповідно змінює стан інших підключених пристроїв.

Спеціальний програмний застосунок дозволить кінцевому користувачеві взаємодіяти з елементами керування та контролювати стани датчиків. Сигнали керування, які надсилаються з хмари на Raspberry Pi, можуть надходити від цієї програми або якогось попередньо визначеного тригера подій на платформі Azure.

3.2 Хмарна архітектура системи

Для реалізації проєкту було обрано хмарну архітектуру через популярність хмарних технологій у забезпеченні масштабованої цифрової інфраструктури, яка одночасно зменшує складність, покращує безпеку та забезпечує гнучкість в управлінні ресурсами.

Хмарні служби Microsoft Azure надають екосистему, яка забезпечує проведення обчислень, аналізу, обробки та зберігання даних, що необхідно для реалізації прототипу СДА. Прототип може бути реалізований шляхом

впровадження хмарної архітектури Microsoft Azure, що забезпечує послуги, необхідні для задоволення потреб проєкту (рис.3.1) [54].

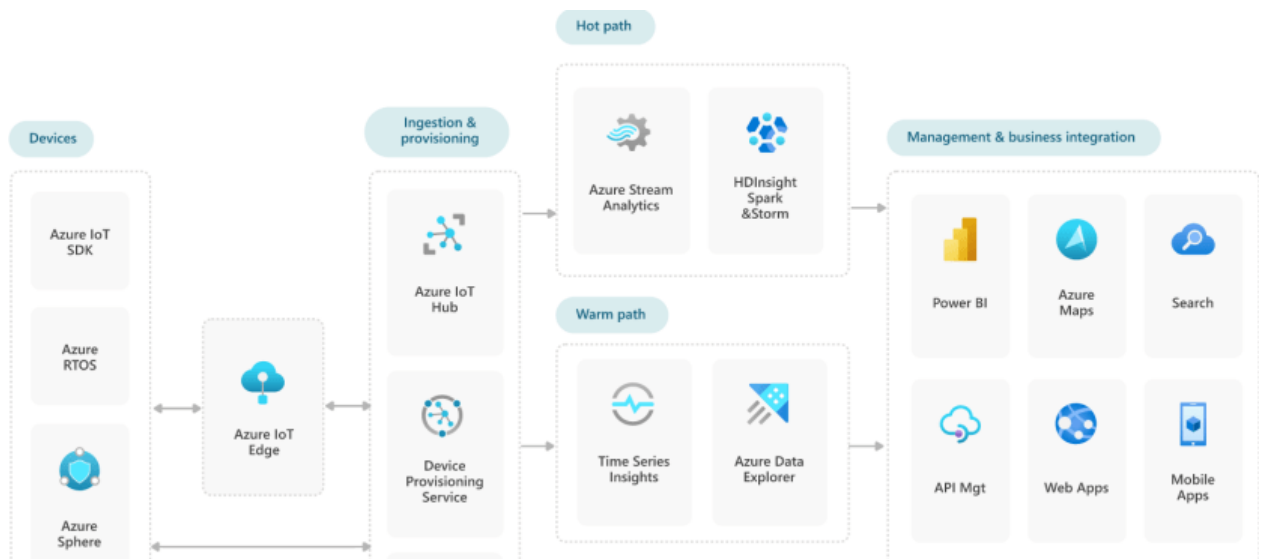


Рисунок 3.1 – Базова архітектура Azure IoT [54].

Концептуальну модель прототипу СДА наведено на рисунку 3.2. Вона демонструє взаємодію користувача з модельованою СДА, властивості її компонентів та причинно-наслідкові зв'язки, властиві системі для досягнення мети моделювання.

3.3 Опис апаратних рішень

Оригінальний план будинку було взято з джерела [55]. Для візуалізації плану помешкання використовувалося програмне забезпечення Edraw Max [56]. На рисунку 3.3 наведено додане розміщення основних компонентів СДА та їх з'єднання: 1–зовнішній світильник; 2–внутрішній світильник; 3–сенсор руху; 4–сенсор вологості; 5–Arduino; 6–Raspberry Pi; 7–реле (перемикач типу SPDT); 8–панель управління; 9–сенсор температури; 10–сервопривід для контролю дверей; 11–конектор.



Рисунок 3.2 – Концептуальна модель СДА

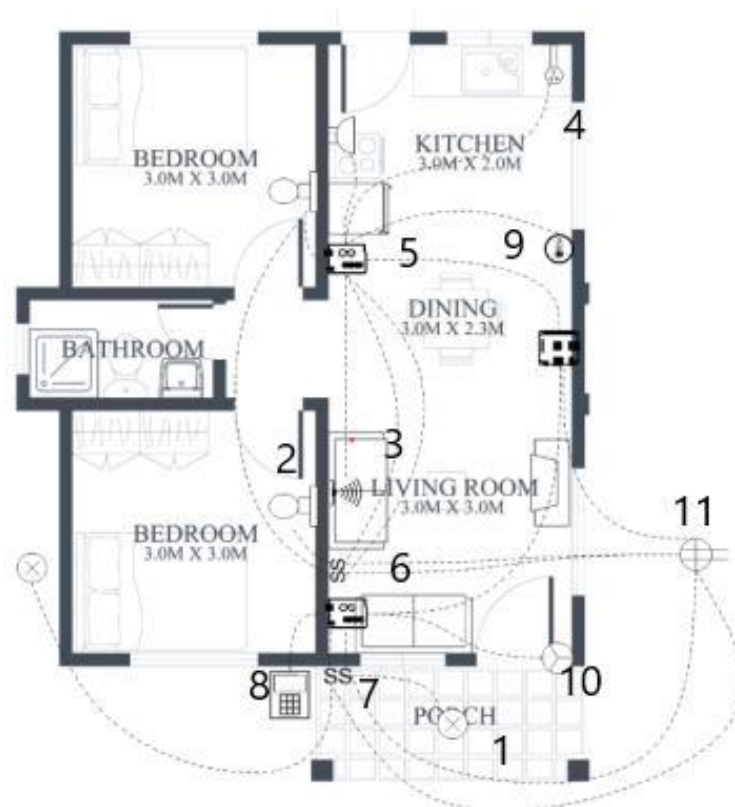


Рисунок 3.3 – Схема розміщення компонентів та топологія мережі СДА

В таблиці 3.3 наведено перелік компонентів апаратного забезпечення для прототипу СДА.

Таблиця 3.3 – Компоненти апаратного забезпечення прототипу СДА.

№ п/п	Найменування	Кількість
1	2	3
1	Raspberry Pi 3	1
2	Плата розширення 9 в 1	1
3	Плата Arduino	2
4	Резистор 2,2 кОм 1 Вт	10
5	Сервопривод Mg90s	2
6	Релейний двоканальний модуль	2
7	Комплект перемичок мама-мама	1
8	Комплект перемичок папа-папа	1
9	Комплект перемичок папа-мама	1

Продовження таблиці 3.3.

1	2	3
10	Монтажна плата	3
11	Двонаправлений логічний перетворювач рівня	3
12	UF5408 Діод	10
13	Модуль LCD 2004	1
14	Клавіатура 4x4	1
15	Подвійний перемикач живлення	3
16	Резистор 10КОм 1W	10
17	Транзистор BC547(NPN)	10
18	RGB світлодіод	18
19	Контактна колодка однорядна мама (40-контактів,2-мм)	2
20	Контактна колодка однорядна папа (40-контактів,2-мм)	10
21	Сенсор руху PIR	2
22	Магнітний вимикач дверей нормально відкритий	

3.4 Схемотехнічне проєктування та моделювання прототипу СДА

Схему управління СДА для зручності було розбито на дві підсистеми для управління зовнішніми та внутрішніми пристроями (рис. 3.4 - 3.5).

Підсистема управління зовнішніми пристроями складається з плати Arduino, до аналогових та цифрових контактів якої підключені сенсори та актуатори (клавіатура, серво, перемикач, РК екран, світлодіоди та ін.). GPIO (General-Purpose Input Output) контакти плати було використано як вхідні (INPUT_PULLUP) та вихідні для організації взаємодії з компонентами СДА. Для підсистеми керування зовнішніми пристроями використано цифрові контакти 0-7 (5V), 9, 10, 12,13 та аналогові контакти А0-А5. Для підсистеми управління внутрішніми пристроями використано цифрові контакти 2, 4, 5, 7, 9, 10, 11, 13 та аналогові контакти А2, А3.

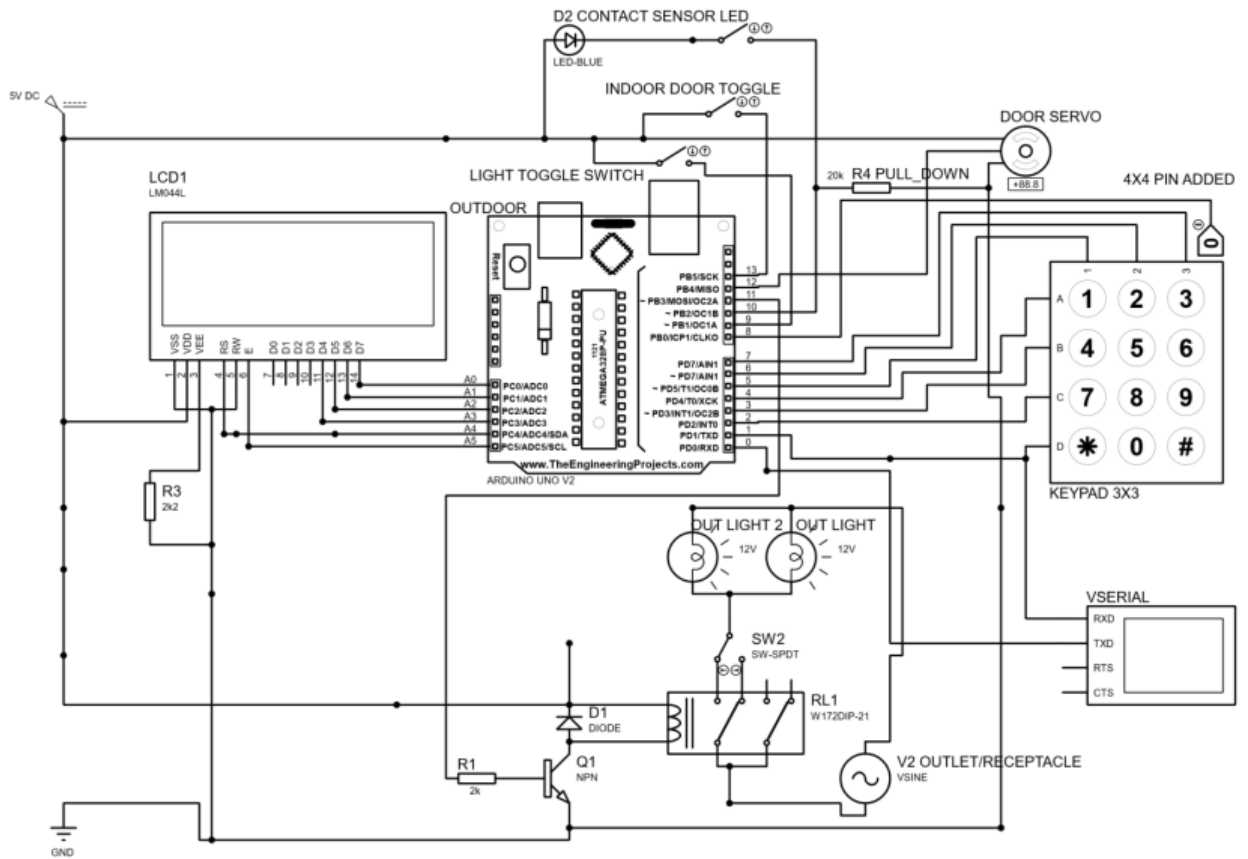


Рисунок 3.4 – Схема підсистеми управління зовнішніми пристроями

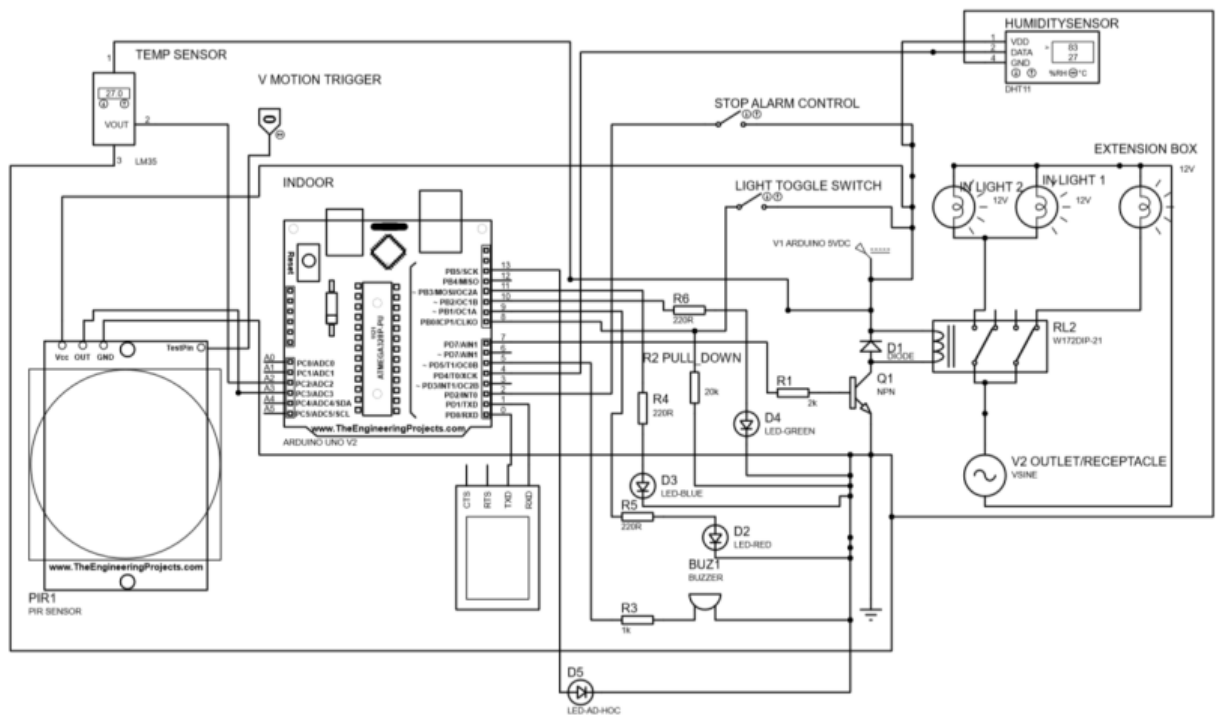


Рисунок 3.5 – Схема підсистеми управління внутрішніми пристроями

Реалізована схема послідовного зв'язку складається з двонаправленої передачі даних між Arduino і Raspberry Pi. Зв'язок реалізовувався через Serial інтерфейс за допомогою USB-кабелів, сумісних з Arduino. Це дозволило передавати дані в дуплексному режимі між Raspberry Pi та Arduino. Raspberry Pi зв'язується з Arduino, надсилаючи двозначне ціле число через послідовний порт до Arduino; потім Arduino зчитує дані та декодує відповідні інструкції за допомогою оператора switch у своїй програмі. Аналогічно, Arduino надсилає у відповідь дані на Raspberry Pi, надсилаючи трисегментний рядок через Serial інтерфейс на Arduino. Рядки розділяються за допомогою крапки з комою (;) і двокрапки (:).

Моделювання процесів управління для розроблених схем було проведено в середовищі Proteus (рис. 3.6 - 3.7). Proteus був обраний для моделювання схем через його точність, а також безліч бібліотечних пристроїв, роботу яких можливо моделювати [57] - [59]. Це дозволило розробити та візуалізувати підключення та функціонування електронних компонентів прототипу. На цьому етапі було перевірено розроблене ПЗ для МК Arduino, щоб переконатися, що помилки в коді відсутні.

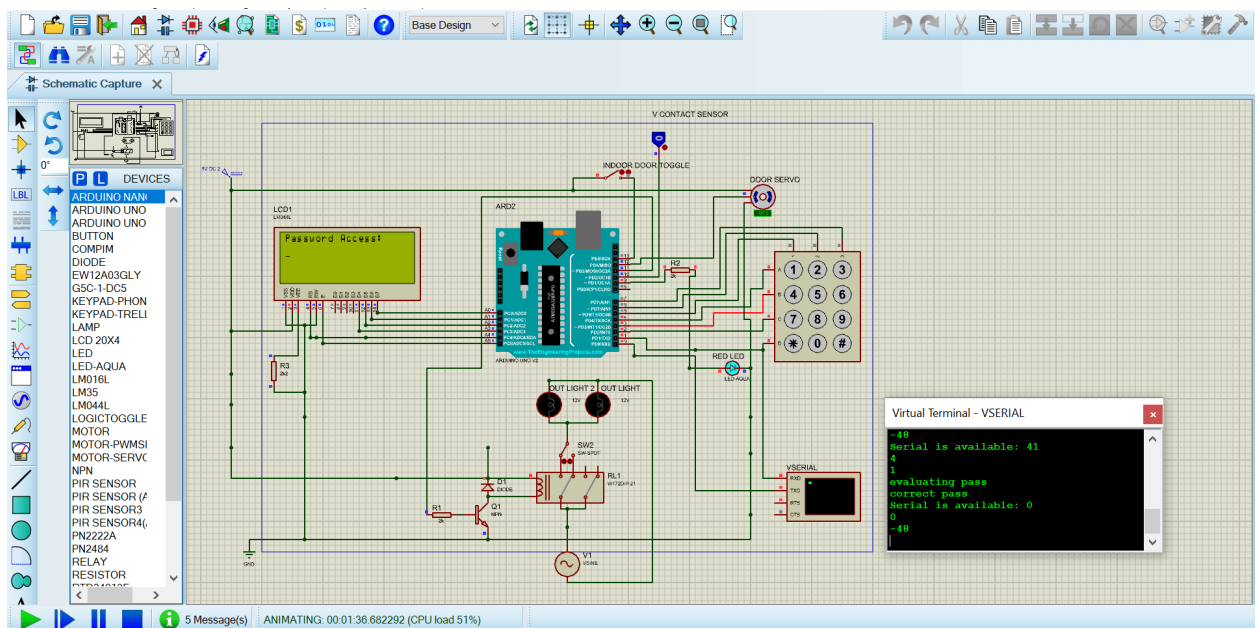


Рисунок 3.6 – Моделювання в Proteus схеми підсистеми управління зовнішніми пристроями

внутрішніми пристроями

Платформа Azure IoT використовувалася для телеметрії процесів «модельований пристрій-хмара» та «хмара-модельований пристрій». Azure IoT надала API, що забезпечує симуляції, пов'язані з IoT, без потреби у фізичних пристроях. Це дозволило прийняти рішення щодо правильності обрання апаратних компонентів та правильності функціонування підсистем управління.

В процесі моделювання було виявлено наступні особливості функціонування прототипу:

- після початку роботи або перезавантаження мінікомп'ютер Raspberry Pi не може надсилати команди або отримувати телеметричні дані, поки він не автентифікує клієнта в його локальній мережі;
- щойно встановлені клієнтські програми не можуть надсилати команди або отримувати телеметрію доки вони не будуть вперше автентифіковані шляхом підключення до Raspberry Pi в локальній мережі;
- клієнту знадобляться справжні облікові дані Facebook для використання програми в будь-який момент часу, а також логін буде

запитуватися для входу в систему (автоматичного або ручного) щоразу, коли клієнт знову відкриває клієнтську програму;

– доступ до серверу можна отримати лише через облікові дані Facebook, використані для автентифікації клієнта.

3.5 Розробка програмного забезпечення прототипу на основі методології повторного використання

Для реалізації прототипу СДА були використані програмні пакети, наведені в таблиці 3.4. Вибір обґрунтований задачами проєкту та необхідністю використання безкоштовного ПЗ. В таблиці 3.5 наведено онлайн застосунки та послуги, які використано для забезпечення функціональних вимог до проєктованого прототипу [60].

Таблиця 3.4 – Опис інструментарію розробки прототипу СДА

№ п/п	Назва програми	Розробник	Плагіни
1	Visual Studio Community	Microsoft Corporation	Visual Studio Build Tools 2017, Azure development .NET cross-platform development, Game development with Unity.
2	Visual Studio Code	Microsoft Corporation	C# support extension, Azure Tools extension, Debugger for Unity extension.
3	Arduino IDE	Arduino	-
4	Processing	Processing Foundation	-
5	Unity3d	Unity Technologies	Facebook SDK for Unity, Newtonsoft JSON for Unity, AdobeXD to Unity UI converter, M2MQTT for Unity, DoTween, Azure Functions for Unity.
6	VNC viewer	ReAlVnc	-
7	PuTTY	Simon Tatham	-
8	Edraw Max	Wondershare	-
9	Software Ideas Modeler	SoftwareIdeas	-
10	Adobe XD	Adobe Inc.	AdobeXD to Unity UI converter
11	Proteus	Labcenter Electronics	Arduino Library for Proteus V2.0, Proteus Libraries of Embedded Sensors.

Таблиця 3.5 – Онлайн-сервіси для дослідження прототипу СДА

№ п/п	Назва програми	Розробник	Опис
1	2	3	4
1	Twilio	Twilio	API зв'язку для SMS, голосового відео та автентифікації.
2	Facebook for Developers	Facebook	Платформа, яка надає API для підключення компаній та клієнтів.
3	Azure Functions	Microsoft Corporation	Хмарний сервіс на вимогу, який надає інфраструктуру, що постійно оновлюється та ресурси, що необхідні застосункам в хмарі.
4	Azure Logic App	Microsoft Corporation	Хмарна платформа для автоматизації робочих процесів, які інтегрують застосунки, дані, послуги та системи.
5	Azure Time Series Insights environment	Microsoft Corporation	Аналітична платформа IoT, що відстежує, аналізує та візуалізує дані аналітики промислового IoT у масштабі.
6	Azure IoT Hub	Microsoft Corporation	Керований сервіс для двостороннього зв'язку між пристроями IoT та Azure.
7	Azure Reource Group	Microsoft Corporation	Контейнер, який містить хмарні ресурси для рішень Azure.
8	Azure Application Insights	Microsoft Corporation	Розширюваний сервіс APM (Application Performance Management) для DevOps розробників.
9	Azure App Service plan	Microsoft Corporation	Визначає набір обчислювальних ресурсів для запуску вебзастосунків.
10	Azure Service Bus Namespace	Microsoft Corporation	Повністю керований брокер повідомлень корпоративного рівня з темами публікації-підписки та чергами повідомлень
11	Azure Storage Account	Microsoft Corporation	Надає унікальні дані простору імен Azure Storage і доступ з будь-якої точки світу.
12	Azure API connection	Microsoft Corporation	Надає попередньо вбудовані операції API для етапів робочого процесу.

Visual Studio Community – безкоштовне повнофункціональне розширюване середовище для створення сучасних програм для Android, iOS і Windows, а також вебзастосунків та хмарних служб [61].

Visual Studio Code (VS Code) – редактор вихідного коду для кросплатформної розробки веб та хмарних застосунків. Включає в себе

відладник, інструменти для роботи з розподіленою системою управління версіями Git, підсвічування синтаксису, технологію автодоповнення IntelliSense та засоби для рефакторингу. Надає широкі можливості для кастомізації: користувальницькі теми, поєднання клавіш і файли конфігурації. Розповсюджується безкоштовно, розповсюджується як програмне забезпечення з відкритим вихідним кодом, але готові збірки розповсюджуються під пропрієтарною ліцензією. Visual Studio Code заснований на фреймворку Electron, що розроблений GitHub та дозволяє розробляти нативні графічні програми за допомогою вебтехнологій, комбінуючи можливості Node.js для роботи з back-end та бібліотеки вебрендерінгу Chromium. Visual Studio Code реалізується через вебредактор Monaco, використовуючи Visual Studio Online [62].

Unity3D – середовище для створення простої та ефективної інфраструктури інтерфейсу користувача, що прискорює розробку базової конфігурації HMI. Дозволяє економити час завдяки стандартним елементам інтерфейсу, таким як регулятори, кнопки, текст, значки, списки прокручування, індикатори, зображення, джерела відеосигналу і маски інтерфейсу [63] - [65].

VNC viewer (VNC client) призначений для роботи з віддаленим комп'ютером по VNC. Ця програма передає на віддалений комп'ютер дані про натискання клавіш і рухи миші, зроблені користувачем, і показує інформацію, призначену до виведення на екран [66].

PuTTY — вільно розповсюджуваний клієнт для різних протоколів віддаленого доступу, включаючи SSH, Telnet, rlogin, який також надає можливість роботи через послідовний порт [67].

Edraw Max Pro – спеціалізована мультиплатформна програма для спрощення створення графіків, схем, діаграм, планів, креслеників та інших документів [56].

Software Ideas Modeler — це верифікатор CASE та UML, що підтримує всі 14 типів діаграм, зазначених в UML 2.5 [68].

Adobe XD (Adobe Experience Design) – програма для розробки інтерфейсів, яка підтримує створення прототипів на основі векторної графіки та вебверстки [69].

Програмне забезпечення клієнтського застосунку було розроблено з використанням AdobeXD, Unity3d і Visual Studio Code. Розроблений набір елементів UI Kit сумісний з AdobeXD було завантажено з [70]. Інформаційну структуру застосунку наведено на рисунку 3.8 [70].

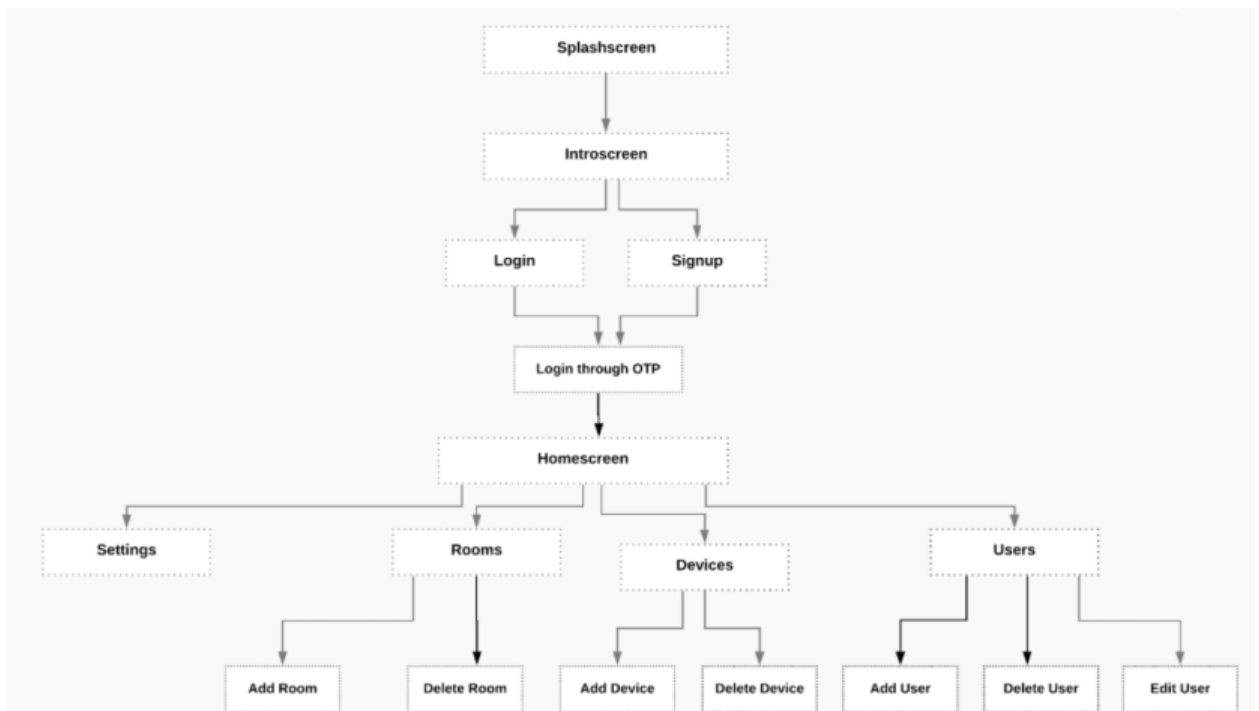


Рисунок 3.8 – Інформаційна структура клієнтського застосунку [70]

Для експорту набору елементів UI у зручний для Unity3d набір медіафайлів використовувався плагін [71]. Інтерфейс клієнтської програми наведено на рисунку 3.9.

Усі конфіденційні дані зберігаються в захищеному місці в хмарі та доступні лише з відповідними правами доступу автентифікованому застосунку клієнта. Таблиця 3.6 містить опис реалізованих методів автентифікації.

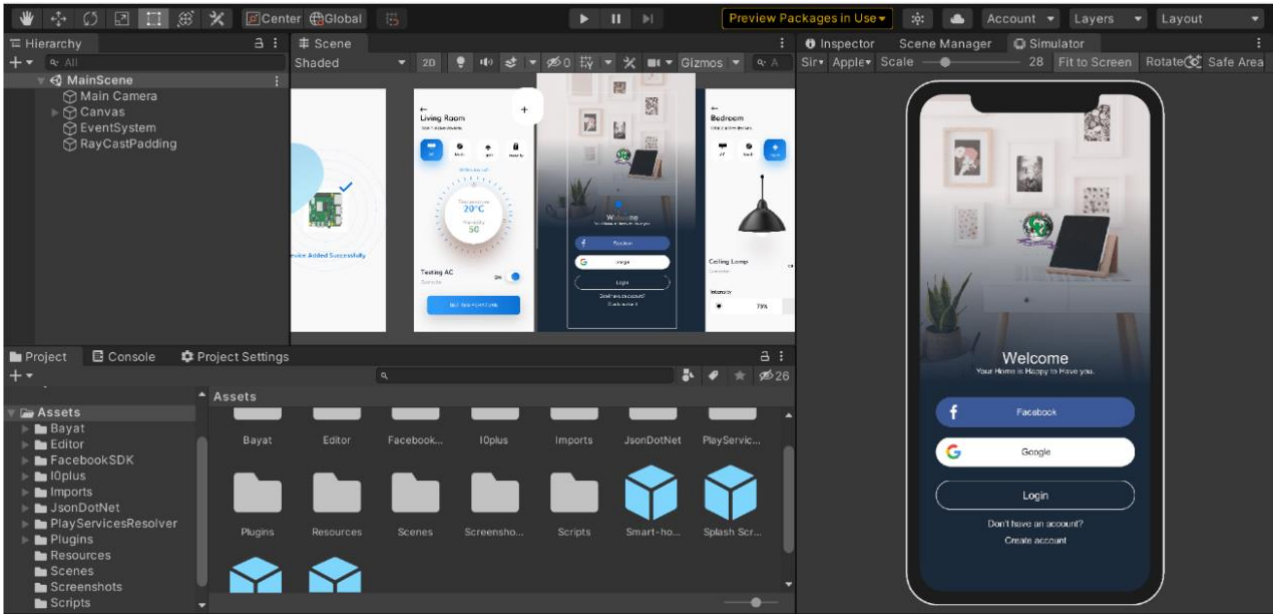


Рисунок 3.9 – Інтерфейс клієнтського застосунку

Таблиця 3.6 – Методи автентифікації

№п/п	Метод	Опис
1	Вхід користувача в програму	Застосунок скеровує користувача до <code>/.auth/login/<provider></code> .
2	Пост-автентифікація	<code>/.auth/login/<provider>/callback</code> .
3	Сеанс автентифікації	Застосунок додає автентифікований файл cookie до відповіді
4	Відображення автентифікованого контенту	Клієнтський застосунок автоматично включає файли cookie для автентифікації в наступні запити.

3.6 Тестування прототипу СДА

Дослідження прототипу СДА, який реалізує функції управління освітленням, контролю доступу, моніторингу показників температури та вологості, проводилися для віртуальної та фізичної складових прототипу с метою встановлення відповідності модельованих процесів реальним. Результати дослідження віртуального прототипу наведено в таблицях 3.7 та 3.8.

Таблиця 3.7 – Результати тестування віртуального прототипу для підсистеми управління зовнішніми пристроями

Вхідні дані (ціле число)	Послідовний вихід (рядок)	Очікуваний результат	Фактичний результат	Коментар
10	mydoorsensor; property2:False	Повертає результат у послідовний порт	Повертає результат у послідовний порт	Для емуляції стану датчика дверей використовується віртуальний тригер, встановлений на 0
20	mydoorcontroller; property2:0	Повертає результат у послідовний порт	Повертає результат у послідовний порт	-
30	-	Вимикає світло	Вимикає світло	-
31	-	Вмикає світло	Вмикає світло	-
32	light control accepted	Перемикає світло	Перемикає світло	-
40	wrong pass	Повертає серво на 0	Повертає серво на - 90	Віртуальний сервопривід відхилений від фази на -90 градусів порівняно з поворотом призначеним кодом.
41	correct pass	Повертає серво на 90	Повертає серво на 0	Віртуальний сервопривід відхилений від фази на -90 градусів порівняно з поворотом призначеним кодом.
42	default pass result	-	-	-

Таблиця 3.8 – Результати тестування віртуального прототипу для підсистеми управління внутрішніми пристроями

Вхідні дані (ціле число)	Послідовний вихід (рядок)	Очікуваний результат	Фактичний результат	Коментар
10	mytemperaturesensor; property2:27.34	Повертає результат у послідовний порт	Повертає результат у послідовний порт	-
20	myhumiditysensor; property2:83	Повертає результат у послідовний порт	Повертає результат у послідовний порт	-
30	mymotionsensor; property2:676	Повертає результат у послідовний порт	Повертає результат у послідовний порт	Віртуальний конвертор 3,3В DC був використаний для перемикавання
40	turn off light	Вимикає світло	Зміни не спостерігаються	Симулятор надто довго відповідав
41	turn on light	Вмикає світло	Зміни не спостерігаються	Симулятор надто довго відповідав
42	toggle light	Перемикає світло	Зміни не спостерігаються	Симулятор надто довго відповідав
50	toggle sound alarm	Перемикає стан звукового сигналу	Зміни не спостерігаються	-

Після створення фізичного прототипу було проведено тестування шляхом підключення Arduino через USB до ПК. Цей метод дозволив протестувати схему як ізольований блок без підключення Arduino до Raspberry Pi. Таблиці 3.9 та 3.10 показують результати дослідження фізичного прототипу для кожної з підсистем.

Таблиця 3.9 – Результати тестування фізичного прототипу для підсистеми управління зовнішніми пристроями

Вхідні дані (ціле число)	Послідовний вихід (рядок)	Очікуваний результат	Фактичний результат	Коментар
1	2	3	4	5
10	mydoorsensor; property2:False	Повертає результат у послідовний порт	Повертає результат у послідовний порт	Для емуляції стану датчика дверей використовується віртуальний тригер, встановлений на 0
10	mydoorsensor; property2: True	Повертає результат у послідовний порт	Повертає результат у послідовний порт	
20	mydoorcontroller; property2:0	Повертає результат у послідовний порт	Повертає результат у послідовний порт	
20	mydoorcontroller; property2:90	Повертає результат у послідовний порт	Повертає результат у послідовний порт	
30	-	Вимикає світло	Вимикає освітлювальні прилади	
31	-	Вмикає світло	Вмикає освітлювальні прилади	
32	light control accepted	Перемикає світло	Перемикає стан освітлювальних приладів	

Продовження таблиці 3.9

1	2	3	4	5
40	wrong pass	Повертає серво на 0	Повертає серво на -90	Віртуальний сервопривід відхилений від фази на -90 градусів порівняно з поворотом призначеним кодом.
41	correct pass	Повертає серво на 90	Повертає серво на 0	Віртуальний сервопривід відхилений від фази на -90 градусів порівняно з поворотом призначеним кодом.
42	default pass result	-	-	-

Таблиця 3.10 – Результати тестування фізичного прототипу для підсистеми управління внутрішніми пристроями

Вхідні дані (ціле число)	Послідовний вихід (рядок)	Очікуваний результат	Фактичний результат	Коментар
1	2	3	4	5
10	mytemperaturesensor; property2:27.34	Повертає результат у послідовний порт	Повертає результат у послідовний порт	-
20	myhumiditiesensor; property2:83	Повертає результат у послідовний порт	Повертає результат у послідовний порт	-
30	mymotionsensor; property2:False	Повертає результат у послідовний порт	Повертає результат у послідовний порт	Для датчика руху потрібна приблизно одна хвилина для калібрування

Продовження таблиці 3.10

1	2	3	4	5
40	turn off light	Вимикає освітлювальні прилади	Вимикає освітлювальні прилади	-
41	turn on light	Вмикає освітлювальні прилади	Вмикає освітлювальні прилади	-
42	toggle light	Перемикає світильники	Перемикає світильники	-
50	toggle sound alarm	Перемикає стан звукового сигналу	Перемикає стан звукового сигналу	-

Моніторинг телеметрії пристроїв проводився за допомогою Visual Studio Code, Azure IoT hub та Azure Function. Visual Studio Code використовувався для відображення логів консолі телеметрії у форматі JSON, які надсилаються до Azure IoT hub як засіб моніторингу безпосередньо на стороні СДА (рис. 3.10).

```

pi@raspberrypi:~$ mkdir ProjectReportTest
pi@raspberrypi:~$ cd ProjectReportTest
pi@raspberrypi:~/ProjectReportTest$ sudo chmod 755 ./SimulatedDevice
pi@raspberrypi:~/ProjectReportTest$ ./SimulatedDevice
Routing Practical: Simulated device
press Ctrl + C to cancel
We're on linux!
directory is /home/pi/.config/ControlFiles
back to main thread
waiting for Motion Sensor Calibration
Read House Temperature
Read House Humidity
Read House Motion Sensor State
Read House Door Sensor State
Read House Door Controller State
Data Received is:password Access:
Data Received is:Serial is available: 10
Data Received is:mydoorsensor;property2:True
Data Received is:Serial is available: 20
Data Received is:mydoorcontroller;property2:175
Data Received is:Serial is available: 30
Data Received is:Serial is available: 10
updated Temperature to 11.72
Data Received is:mytemperaturesensor;property2:11.72
Data Received is:Serial is available: 20
Data Received is:Serial is available: 0
updated Humidity to 67
Data Received is:myhumiditysensor;property2:67
Data Received is:Serial is available: 0
event sent
this is a normal message > Sent message: parameter 8/4/2021 10:35:01 PM : {"property2":false,"PartitionKey":"microcomp
uter","RowKey":"rpi123BoluM","deviceId":"myrpi","propertyLabel1":"activated","propertyLabel2":"connected","property1
":false,"Etag":"rpi123microcomputer","Misc":"awakeMode"}
event sent
  
```

Рисунок 3.10 – Моніторинг логів тестування телеметрії на стороні СДА

Функція Azure була використана як засіб віддаленого моніторингу логів (рис. 3.11). Основне завдання функції Azure полягало в тому, щоб зберігати дані телеметрії в сховищі таблиць Azure для асинхронного доступу клієнтських пристроїв, а також виконувати функцію зчитувача логів. Код на стороні пристрою був скомпільований за допомогою .NET5.0 та команди dotnet publish to Linux. Зібрані файли були перенесені на Raspberry Pi за допомогою команди "scp".

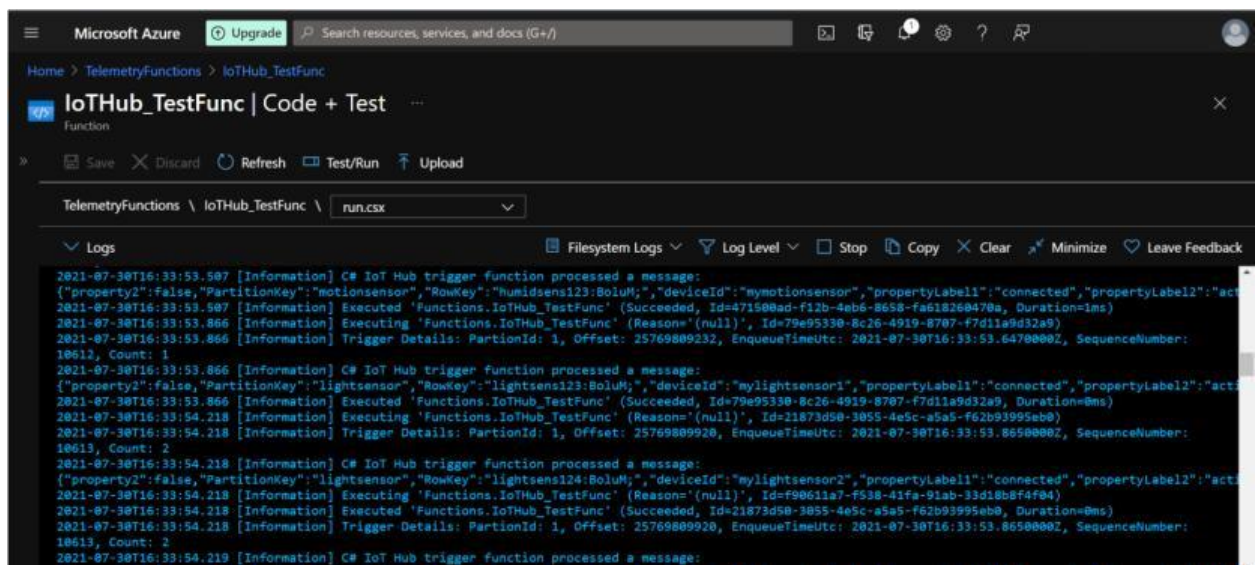


Рисунок 3.11 – Віддалений моніторинг логів тестування телеметрії

Хоча для відображення докладних логів використовувалася функція Azure, в той же час Azure IoT Hub надавав графічну інформацію про різні метрики: невдалі прямі виклики методів, успішні прямі виклики методів, доставлені повідомлення телеметрії, втрачені повідомлення та надіслані повідомлення від пристрою до хмари (рис. 3.12). Тип графіка був встановлений на лінійну діаграму; діапазон часу вимірювання (шкала X) був встановлений на 24 години; деталізацію за часом було встановлено на локальну, а Y-масштаб було встановлено автоматично засобами Azure IoT Hub. Також було протестовано дані часових рядів за допомогою Azure Time Series Insights (TSI), що створений як надійний сервіс для аналізу даних та візуалізації, а не лише для передачі даних з Azure IoT Hub (рис. 3.13).

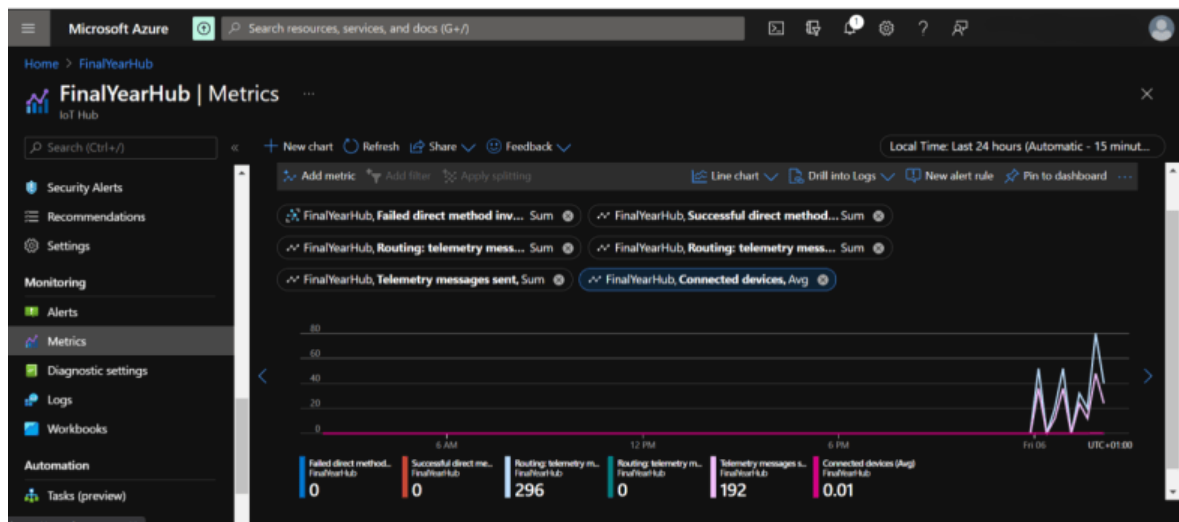


Рисунок 3.12 – Віддалений моніторинг метрик системи

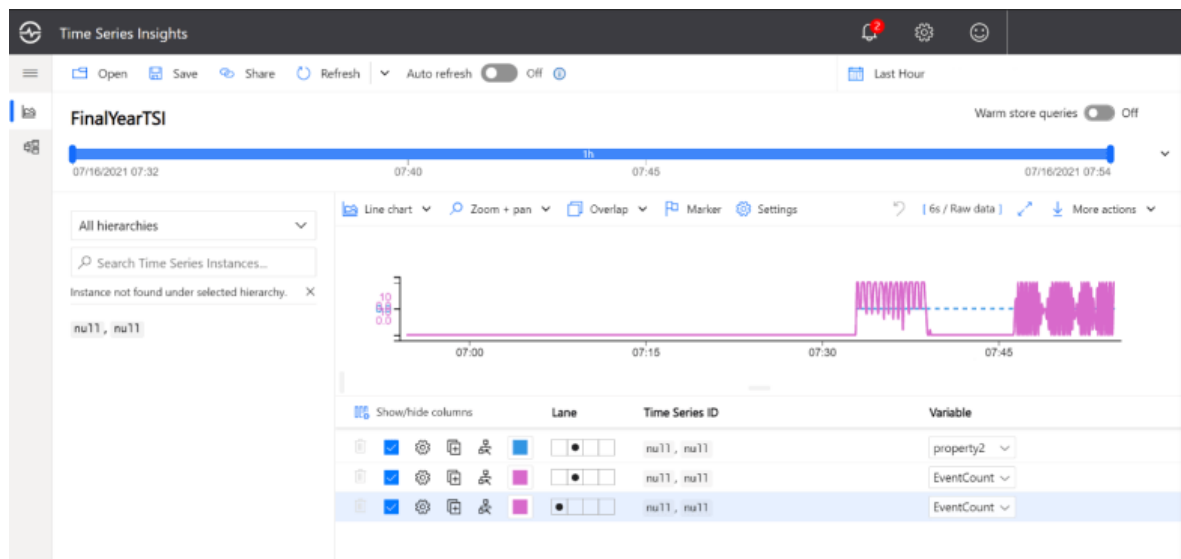


Рисунок 3.13 – Віддалений моніторинг даних часових рядів

Тестування пристроїв прямим методом було проведено за допомогою функції прямого виклику методу пристрою Azure IoT Hub Device (рис. 3.14). Мікроконтролерам Arduino були призначені постійні мітки портів, щоб забезпечити узгодженість розпізнавання Raspberry Pi, навіть якщо Raspberry Pi потребує перезапуску. Подібно до тестів послідовного порту, застосований метод тестування дозволив розглядати налаштування на стороні пристрою як ізольовані без необхідності застосування мобільного пристрою/планшета для використання прямих методів пристрою. У таблиці 3.11 наведено результати тестування пристроїв прямим методом.

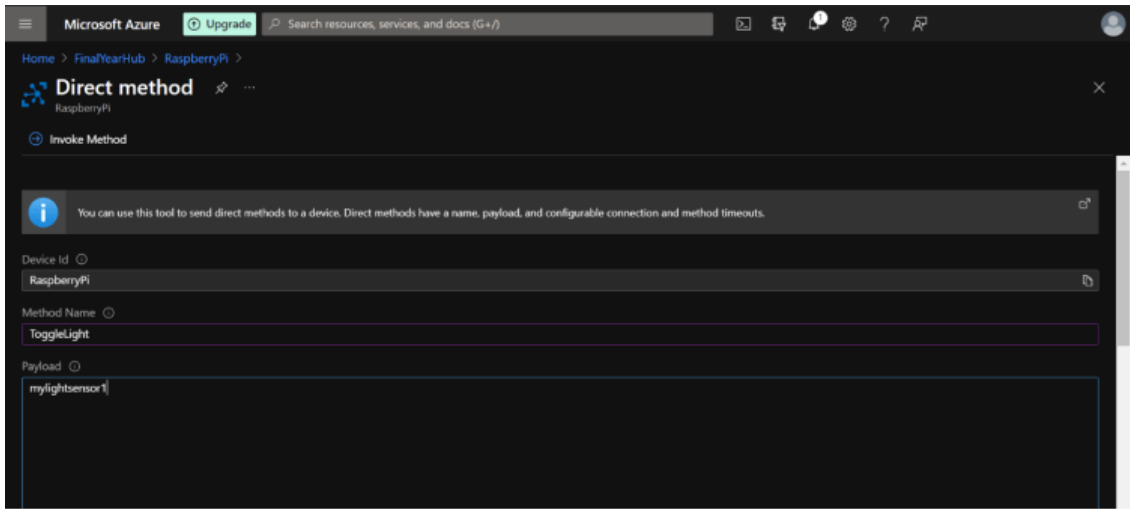


Рисунок 3.14 – Тестування пристроїв прямим методом Azure IoT-Hub

Таблиця 3.11 – Результати тестування пристроїв прямим методом

Назва методу	Корисне навантаження	Очікуваний результат	Фактичний результат	Коментар
1	2	3	4	5
ToggleLight	mylightsensor2	Перемикає зовнішні світильники	Зовнішні світильники переключені	
ToggleLight	Mylightsensor 1	Перемикає внутрішні світильники	Внутрішні світильники переключені	
TogglePresenceMode	1	Перемикає режим присутності	Режим присутності переключено	Режим присутності перемикався між всередині/зовні
ToggleSleep Mode	1	Перемикає режим сну	Увімкнено режим сну	Режим присутності перемикається між сон/не сон
ToggleDoor	1	Перемикає стан дверей	Стан дверей увімкнено	Зміни не спостерігаються
LockDoor	0	Встановлює стан дверей на заблоковані, якщо вони вже були заблоковані	Зміни не спостерігаються	
LockDoor	1	Встановлює стан дверей на заблоковані, якщо вони не були заблоковані	Стан дверей змінено на фізично заблокований	Раніше двері були у відкритому стані

Продовження таблиці 3.11

1	2	3	4	5
ToggleMotionSensor	1	Вимикає датчик руху	Зміни не спостерігаються	Функцію не реалізовано
ToggleExtensionBox	1	Перемикає стану поля розширення	Зміни не спостерігаються	Функцію не реалізовано

Система була налаштована на автоматичне надсилання SMS-повідомлень та сповіщень електронною поштою за попередньо налаштованою подією або умовним контекстом. SMS-повідомлення були реалізовані за допомогою Twilio API (рис. 3.15), а сповіщення електронною поштою – за допомогою Gmail SMTP. В таблиці 3.12 показані результати тестування сповіщень.

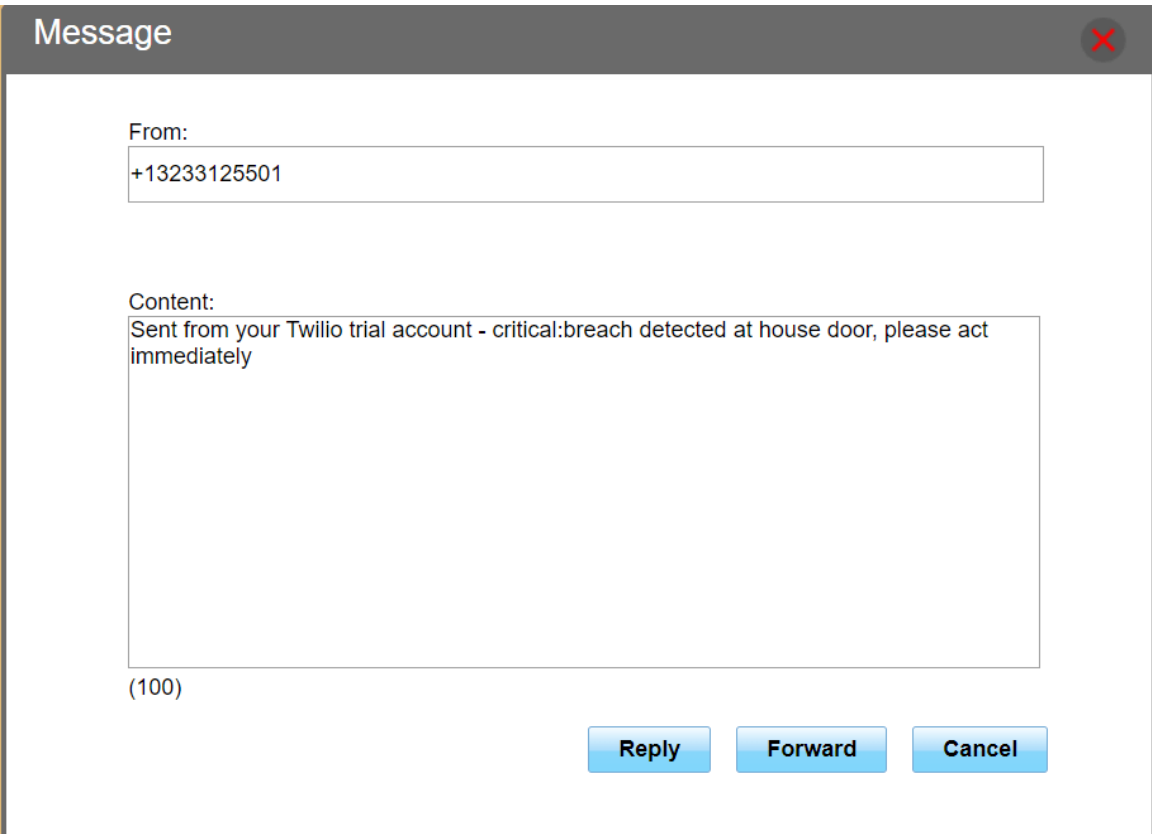


Рисунок 3.15 – Тестування SMS сповіщень

Таблиця 3.12 – Тестування сповіщень

№ п/п	Контекст сповіщення	Очікуваний результат	Фактичний результат	Коментар
1	Температура в кімнаті перевищує 40 °C	Оповіщення клієнта електронною поштою та SMS	Клієнтський пристрій успішно сповіщений електронною поштою та SMS	-
2	Відносна вологість повітря в приміщенні перевищує 55%	Оповіщення клієнта електронною поштою та SMS	Клієнтський пристрій успішно сповіщений електронною поштою та SMS	-
3	Датчик руху виявляє рух, коли клієнтський пристрій перебуває в режимі відсутності	Оповіщення клієнта електронною поштою та SMS	Клієнтський пристрій успішно сповіщений електронною поштою та SMS	Номер телефону користувача був збережений для SMS-повідомлення
4	Датчик дверей зафіксував, що двері відкриті, коли клієнтський пристрій перебуває в режимі сну	Оповіщення клієнта електронною поштою та SMS та автоматичне закриття дверей	Клієнтський пристрій успішно сповіщений електронною поштою та SMS	Режим присутності перемикався між режимом сну та не сну
5	Датчик руху та датчик дверей виявляють проникнення через двері	Оповіщення клієнта електронною поштою та SMS	Клієнтський пристрій успішно сповіщений електронною поштою та SMS	
6	Внутрішнє та зовнішнє освітлення увімкнено	Оповіщення клієнта електронною поштою та SMS	Клієнтський пристрій успішно сповіщений електронною поштою та SMS	Дані пристрою зберігаються як частина вимоги автоматичної автентифікації для наступного запуску програми.

Витрати на використання хмарної служби, відстежувалися за допомогою порталів Azure IoT і Twilio (рис. 3.16). Очевидно, що витрати на дослідження були незначними та склали біля 3,4 доларів.

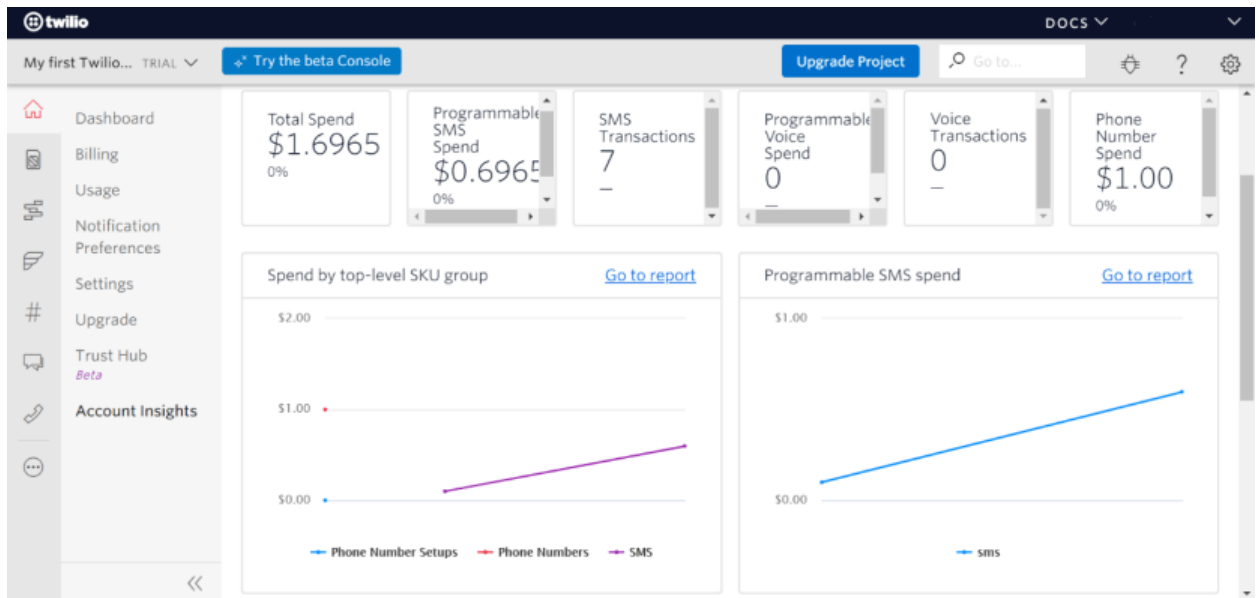


Рисунок 3.16 – Портал Twilio Cost Management

3.6 Висновки до розділу 3

В даному розділі відповідно до мети та задач магістерської роботи виконано аналіз вимог до прототипу СДА, розроблено хмарну архітектуру, обґрунтовано вибір апаратних та програмних засобів розробки.

Виконано дослідження віртуального прототипу СДА, а саме моделювання розробленого ПЗ для платформи Arduino в середовищі Proteus, а також тестування взаємодії прототипу СДА з хмарною платформою Azure IoT.

Порівняння результатів дослідження віртуального та фізичного прототипів СДА дозволили виявити особливості реалізації функцій управління та внести потрібні зміни в код програми.

ВИСНОВКИ

В результаті проведених досліджень показано актуальність розробки та дослідження віртуальних та фізичних прототипів СДА, а також впровадження хмарних технологій для удосконалення процесів управління процесами домашньої автоматизації.

В роботі розроблено програмне забезпечення графічного інтерфейсу Arduino/Processing. Проведено моделювання роботи підсистем управління зовнішніми та внутрішніми пристроями СДА на основі розробленого ПЗ в середовищах Tinkercad та Proteus VSM.

Розроблено концептуальну модель СДА та досліджено можливості реалізації її хмарної архітектури на платформі Azure IoT. Показано можливість повторного використання на основі методології reuse

Наукове значення роботи полягає в тому, що удосконалено методику розробки прототипу СДА, яка відрізняється від існуючих застосувань методології повторного використання апаратних та програмних рішень, а також хмарних технологій, що дозволяє створити недорогий повнофункціональний прототип СДА.

Практичне значення роботи полягає в тому, що розроблений на основі удосконаленої методики прототип СДА має менші, у порівнянні з аналогами, вартість та час розробки за рахунок використання доступних апаратних та готових програмних рішень та надає функції віддаленого моніторингу, масштабування та оновлення ПЗ СДА через хмару.

В майбутньому набір підсистем прототипу СДА може бути легко розширений для дослідження інших функціональних можливостей та процесів управління СДА.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Fourth industrial revolution [Electronic source]. – Access mode: https://en.wikipedia.org/wiki/Fourth_Industrial_Revolution 2021.
2. Internet of Things [Electronic source]. – Access mode: https://en.wikipedia.org/wiki/Internet_of_things.
3. Умный дом [Электрон. ресурс]. – Режим доступа: <https://biysk-tv.ru/raznoe-2/samostoyatel'naya-rabota-po-informatike-umnyj-dom-samostoyatel'naya-rabota-1-umnyj-dom.html>.
4. Умный дом и автоматизация инженерных систем — что общего? [Электрон. ресурс]. – Режим доступа: https://smarton.com.ua/smart_home/zachem_nugen_umniy_dom.
5. Умный дом [Электрон. ресурс]. – Режим доступа: <https://remont-f.ru/remont-kvartir-pod-kluch/home.php><https://remont-f.ru/remont-kvartir-pod-kluch/home.php>.
6. Шелухин, О. И. Интегрированная система управления информационной инфраструктурой «интеллектуального здания» / О. И. Шелухин, В. М. Артюшенко // Электротехнические комплексы и системы. – 2008. – №1,2. – Т.4. – С. 19-24.
7. Технологии и составные компоненты умного дома: Типовые подсистемы и модули [Электрон. ресурс]. – Режим доступа: <http://www.rovdo.com/components-smart-home>.
8. Умный дом. Основные концепции, техническое и программное обеспечение [Электрон. ресурс]. – Режим доступа: <https://www.stud24.ru/construction/umnyj-dom-osnovnye-koncepcii-tehnicheskoe/214761-628299-page2.html>.
9. Управление освещением и электропитанием [Электрон. ресурс]. – Режим доступа: https://smarton.com.ua/funkcii_umnogo_doma/upravlenie_electropitaniem_osvesh-eniem_v_umnom_dome/.

10. Климатический контроль в умном доме [Электрон. ресурс]. – Режим доступа: <https://gadgetpage.ru/smart-house/3007-klimaticheskij-kontrol-v-umnom-dome.html>.
11. Інтелектуальне середовище системи домашньої автоматизації / [А.В. Пархоменко, А.В. Туленков, О.В. Соколянський и др.] // Наукові праці ДонНТУ. – 2020. – №1 (30). – С. 101-111 (Серія «Інформатика, кібернетика та обчислювальна техніка»).
12. Аналіз ринку систем «Розумний дім» в Україні. 2019 рік [Електрон. ресурс]. – Режим доступу: <https://pro-consulting.ua/ua/issledovanie-rynka/analiz-rynka-sistem-umnyj-dom-v-ukraine-2019-god>.
13. Розум, який можна купити за гроші: аналіз ринку систем "Розумний дім" в Україні [Електрон. ресурс]. – Режим доступу: <https://pro-consulting.ua/ua/pressroom/um-kotoryj-mozhno-kupit-za-dengi-analiz-rynka-sistem-umnyj-dom-v-ukraine>.
14. Умный дом оборудование для системы | Цифровой дом – автоматика для Умного Дома и схематическое представление системы [Электрон. ресурс]. – Режим доступа: <https://smarton.com.ua/novosti-umnogo-doma/tpost/ktu5fma9m1-umnii-dom-oborudovanie-i-avtomatika-d>.
15. Z-Wave vs ZigBee, WiFi, Thread, Bluetooth BLE: выбираем протокол управления умным домом [Электрон. ресурс]. – Режим доступа: https://superhome.pro/z-wave-vs-zigbee-wifi-thread-bluetooth-ble-vybiraem-protokol-upravleniya-umnym-domom/#_yc8jpw1pzje2.
16. The features of wireless technologies application for Smart house systems / [A. Tulenkov, A. Parkhomenko, A. Sokolyanskii et. al.] Wireless systems: 4th IEEE International symposium within the IEEE International conferences on Intelligent data acquisition and advanced computing systems: proceedings. - Los Alamitos: IEEE, 2018. - P.1-6.
17. Николаев, П. Л. Архитектура интегрированной в облачную среду системы управления умным домом / П. Л. Николаев // Программные продукты и системы. – 2015. - № 2. – С. 65-69.

18. Николаев, П. Л. Применение облачных технологий в системах умного дома / П. Л. Николаев // Молодой ученый. – 2014. – № 13 (72). – С. 37-39.
19. Плюсы и минусы облачного сервера для умного дома [Электрон. ресурс]. – Режим доступа: <http://surl.li/awciq>.
20. ТОП 5: Рейтинг лучших систем «Умный дом» по производителям 2021 года [Электрон. ресурс]. – Режим доступа: <https://vencon.ua/articles/rejting-sistem-umnyy-dom-po-proizvoditelyam>.
21. Створення та використання віддаленої лабораторії як відкритого онлайн ресурсу / [А.В. Туленков, А.В. Пархоменко, О.В. Соколянський та ін.] // Вісник НУ «Львівська політехніка». – 2018. – №903. – С. 88-97. – (Серія «Інформатизація вищого навчального закладу»).
22. KNX training with the new GAMMA Training Kit 5.1. Modular, easy and individually configurable [Electronic source]. – Access mode: <https://docplayer.net/206260948-Knx-training-with-the-new-gamma-training-kit-5-1-modular-easy-and-individually-configurable-siemens-com-gamma.html>.
23. GAMMA Training Kit 5.1 – presentation with prices [Electronic source]. – Access mode: https://www.hqs.sbt.siemens.com/cps_product_data/data/en/support/gtk/GTK_5_1_information_EN_without_prices_2019_20.pdf.
24. W4006-0B – produkt – didacta 2018 [Electronic source]. – Access mode: <https://www.didacta-hannover.de/produkt/w4006-0b/2519507/U896355>.
25. Пархоменко А. Семінар-тренінг з розробки систем домашньої автоматизації [Електрон. ресурс] / А. Пархоменко. – Режим доступу: <http://www.zntu.edu.ua/seminar-trening-z-rozrobky-system-domashnoyi-avtomatyzaciyi>.
26. Первая Smart Home конференция | Gira [Электрон. ресурс]. – Режим доступа: <http://www.gira.ru/about/novosti/1390>.
27. IoT reference implementation: how to build the Smart Home prototype [Electronic source]. – Access mode: <https://software.intel.com/en->

us/articles/iot-reference-implementation-how-to-build-the-smart-home-prototype.

28. A wireless, voice-controllable, household system [Electronic source].
– Access mode:
https://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/f2012/jw937_sz369/jw937-sz369/jw937_sz369.html.
29. From RGB led laboratory to servomotor control with websockets and IoT as educational tool / [G. C. Fernandez, E. S. Ruiz, M. C. Gil et al.] // Remote Engineering and Virtual Instrumentation (REV): 12th international conference, Bangkok, Thailand, 25-28 February 2015: proceedings. – Los Alamitos: IEEE. - P. 32-36.
30. Remote experimentation through Arduino-based remote laboratories // [S. Martin, A. Fernandez-Pacheco, J. A. Ruipérez-Valiente et al.] // IEEE Revista Iberoamericana de Tecnologías del Aprendizaje.- 2021. - vol. 16 (2). - P. 180-186.
31. The application of the remote lab for studying the issues of Smart House systems power efficiency, safety and cybersecurity / [A. Parkhomenko, A. Tulenkov, A. Sokolyanskii et al.] // Smart Industry & Smart Education. – Cham: Springer. - 2019. – vol. 47. - P. 395-403 (Book series Lecture Notes in Network and Systems).
32. Integrated complex for IoT technologies study / [A. Parkhomenko, A. Tulenkov, A. Sokolyanskii et al.] // Online Engineering & Internet of Things. – Cham: Springer. - 2018. - vol. 22. - P. 322-330 (Book series Lecture Notes in Network and Systems).
33. How to select your next development board. Electronicsforu. [Electronic source]. – Access mode: <http://electronicsforu.com/buyers-guides/hardware-buyers-guide/select-next-development-board/3>.
34. Choosing the right embedded development board. Electronicsforu. [Electronic source]. – Access mode: <http://electronicsforu.com/buyers-guides/hardware-buyers-guide/embedded-development-board>.
35. Гид по выбору платформы разработки. Амперка [Электрон. ресурс]. – Режим доступа: <http://amperka.ru/page/development-board-guide>.

36. Arduino [Электронный ресурс] — Режим доступа: <https://uk.wikipedia.org/wiki/Arduino>.
37. Arduino [Electronic resource]. — Access mode: <https://store.arduino.cc/>.
38. Какие существуют среды разработки (IDE) для ARDUINO [Электрон. ресурс]. — Режим доступа: <https://soltau.ru/index.php/themes/kompyutery-i-programmy/item/465-kakie-sushchestvuyut-sredy-razrabotki-ide-dlya-arduino>.
39. Arduino: ТОП-3 графических сред программирования [Электрон. ресурс]. — Режим доступа: <https://ichip.ru/arduino-top-3-graficheskikh-sred-programmirovaniya.html>.
40. Arduino IDE [Электрон. ресурс]. — Режим доступа: <https://arduino-ide.com/>.
41. FLProg — альтернативная среда программирования Arduino. Описание проекта [Электрон. ресурс]. — Режим доступа: <https://habr.com/ru/company/flprog/blog/242335/>.
42. avr-gcc Atmel AVR микроконтроллеры вкл. Linux / Windows Arduino IDE [Электрон. ресурс]. — Режим доступа: <http://surl.li/awdyu>.
43. Visual Studio или Atmel Studio? [Электрон. ресурс]. — Режим доступа: <https://www.visualmicro.com/page/sysuserdevguide.aspx?doc=ru/Getting-started-which-IDE.html>.
44. Связываемся с Processing [Электрон. ресурс]. — Режим доступа: <http://developer.alexanderklimov.ru/arduino/processing.php>.
45. Processing и Arduino [Электрон. ресурс]. — Режим доступа: <https://robocraft.ru/blog/arduino/336.html>.
46. Arduino/Processing Language Comparison [Electronic source]. — Access mode: <https://www.arduino.cc/en/Reference/Comparison>.
47. Сетевые возможности Processing [Электрон. ресурс]. — Режим доступа: <https://electromicro.ru/resources/wiki/processing/processing3/>.

48. Processing и Ардуино. Работа по Serial [Электрон. ресурс]. – Режим доступа: <https://electromicro.ru/resources/wiki/processing/processing2/>.

49. Tinkercad [Electronic resource]. – Access mode: <https://www.tinkercad.com/>.

50. Tinkercad Arduino – лучший онлайн симулятор Ардуино на русском [Электрон. ресурс]. – Режим доступа: <https://arduino-master.ru/program/simulyator-arduino-tinkercad-circuits/>.

51. Описание для Ардуино виртуальный симулятор на русском. Tinkercad Dashboard – Начальная страница [Электрон. ресурс]. – Режим доступа: <https://sherstinka.ru/rezyume/opisanie-dlya-arduino-virtualnyi-simulyator-na-russkom-tinkercad/>.

52. Design, specification and implementation of a distributed Home automation system / [T. A. Abdulrahman, O. H. Isiwepeni, N. T. Surajudeen-Bakinde et. al.] // Procedia Computer Science. – 2016. - vol. 94. - P. 473-478.

53. Raspberry Pi documentation [Electronic source]. – Access mode: <https://www.raspberrypi.org/documentation>.

54. Azure IoT reference architecture [Electronic source]. – Access mode: <https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/iot>.

55. Small and simple house design with two bedrooms [Electronic source]. – Access mode: <https://ulrichome.com/home-plans/small-simple-house-design-two-bedrooms/>.

56. Edraw Max [Электрон. ресурс]. – Режим доступа: <https://soft-file.ru/edraw-max/>.

57. Proteus VSM [Электрон. ресурс]. – Режим доступа: <https://cxem.net/software/proteus.php>.

58. Shaw, A. Arduino library for Proteus V2.0 [Electronic source] / A. Shaw. - Access mode: <https://www.theengineeringprojects.com/2021/03/arduino-library-for-proteus-v2.html>.

59. Wilson, J. Proteus libraries of embedded sensors [Electronic source] / J. Wilson. – Access mode:

<https://www.theengineeringprojects.com/2020/07/proteus-libraries-of-embedded-sensors.html>.

60. Technical documentation [Electronic source]. – Access mode: <https://docs.microsoft.com/en-us/documentation/>.

61. Visual Studio Community [Electronic source]. – Access mode: <https://visualstudio.microsoft.com/ru/vs/community/>.

62. Visual Studio Code [Электрон. ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Visual_Studio_Code.

63. Unity3D для создания простой и эффективной инфраструктуры пользовательского интерфейса [Электрон. ресурс]. – Режим доступа: <https://russianblogs.com/article/43431013819/>.

64. Viganò, G. P. M2MQTT for Unity [Electronic source] / G. P. Viganò, F. Jasche, C. Pogolski. – Access mode: <https://github.com/gpvgiano/M2MqttUnity>.

65. Deadlyfingers, Azure functions for Unity [Electronic source]. – Access mode: <https://github.com/Unity3dAzure/AzureFunctions>.

66. Подключение к удаленному компьютеру по VNC [Электрон. ресурс]. – Режим доступа: <https://habr.com/ru/post/76343/>.

67. PuTTY [Электрон. ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/PuTTY>.

68. Software Ideas Modeler [Electronic source]. – Access mode: https://en.wikipedia.org/wiki/Software_Ideas_Modeler.

69. Adobe XD [Электрон. ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Adobe_XD.

70. Verma, S. Smart Home mobile app. [Electronic source] / S. Verma. – Access mode: <https://www.behance.net/gallery/91128375/Smart-Home-Mobile-App>.

71. Ito, H. AdobeXD to Unity UI converter [Electronic source] / H. Ito, KlingOne. – Access mode: <https://github.com/itouh2-i0plus/XuidUnity>.

ДОДАТОК А
Текст програми

A.1 Текст програми для Arduino/ RGB

```
int red = 11;
int blue = 10;
int green = 9;

int commandValue;
int rgbCounter = 0;

int RedValue = 0;
int GreenValue = 0;
int BlueValue = 0;

void setup()
{
    Serial.begin(9600);

    pinMode(red, OUTPUT);
    pinMode(blue, OUTPUT);
    pinMode(green, OUTPUT);
}

void loop()
{
    if (Serial.available()) {

        commandValue = Serial.parseInt();

        Serial.println(commandValue);

        switch (rgbCounter) {
        case 0:
            RedValue = commandValue;
            rgbCounter++;
            break;
            break;
        case 1:
            GreenValue = commandValue;
            rgbCounter++;
            break;
        case 2:
            BlueValue = commandValue;
            rgbCounter++;
            break;
```

```

    }
}

if (rgbCounter == 3)
{
    analogWrite(red, RedValue);
    analogWrite(green, GreenValue);
    analogWrite(blue, BlueValue);

    rgbCounter = 0;
    Serial.flush();
    delay(1000);
}

delay(20);
}

```

A.2 Текст програми Processing/ RGB

```

import processing.serial.*;

Serial serial; // создаем объект последовательного
порта
String received; // данные, получаемые с
последовательного порта

color c1 = color(255, 204, 0);
color c2 = color(255, 0, 255);
color c3 = color(25, 155, 205);

color c1Hover = color(255, 204, 244);
color c2Hover = color(255, 244, 255);
color c3Hover = color(244, 155, 203);

int FirstX = 50; int FirstY = 50;
int SecondX = 150; int SecondY = 170;
int ThirdX = 250; int ThirdY = 290;

int D = 100;

void setup()
{
    size (400,400);
}

```

```

String port = Serial.list()[0];
serial = new Serial(this, port, 9600);

smooth();
background(0);
}

void draw()
{
    background(0);

    strokeWeight(0);
    fill(c1);
    rect(FirstX, FirstY, D, D);

    strokeWeight(0);
    fill(c2);
    rect(SecondX, SecondY, D, D);

    strokeWeight(0);
    fill(c3);
    rect(ThirdX, ThirdY, D, D);

    mouseHover();
}

void mouseHover()
{
    if (mouseX >= FirstX && mouseX <= FirstX+D &&
mouseY >= FirstY && mouseY < FirstY+D){
        fill(c1Hover);
        rect(FirstX, FirstY, D, D);
    }

    if (mouseX >= SecondX && mouseX <= SecondX+D
&& mouseY >= SecondY && mouseY < SecondY+D){
        fill(c2Hover);
        rect(SecondX, SecondY, D, D);
    }

    if (mouseX >= ThirdX && mouseX <= ThirdX+D &&
mouseY >= ThirdY && mouseY < ThirdY+D){
        fill(c3Hover);
        rect(ThirdX, ThirdY, D, D);
    }
}

```

```

    }

    void mousePressed()
    {
        if (mouseX >= FirstX && mouseX <= FirstX+D &&
mouseY >= FirstY && mouseY < FirstY+D){

            int c1Red = int(red(c1)); println(c1Red);
serial.write(c1Red);
            int c1Green = int(green(c1));
println(c1Green); serial.write(c1Green);
            int c1Blue = int(blue(c1)); println(c1Blue);
serial.write(c1Blue);
        }

        if (mouseX >= SecondX && mouseX <= SecondX+D
&& mouseY >= SecondY && mouseY < SecondY+D){

            int c2Red = int(red(c2)); println(c2Red);
serial.write(c2Red);
            int c2Green = int(green(c2));
println(c2Green); serial.write(c2Green);
            int c2Blue = int(blue(c2)); println(c2Blue);
serial.write(c2Blue);
        }

        if (mouseX >= ThirdX && mouseX <= ThirdX+D &&
mouseY >= ThirdY && mouseY < ThirdY+D){

            int c3Red = int(red(c3)); println(c3Red);
serial.write(c3Red);
            int c3Green = int(green(c3));
println(c3Green); serial.write(c3Green);
            int c3Blue = int(blue(c3)); println(c3Blue);
serial.write(c3Blue);
        }
    }
}

```

A.3 Текст программы для Processing/ серво

```

import processing.serial.*;

Serial serial; // создаем объект последовательного
порта

```

```

String received; // данные, получаемые с
последовательного порта

PFont f;

int ScaleStartX = 40; int ScaleY = 100;
int ScaleEndX = 760;

int MarkLen = 20;

int SliderWidth = 20; int SliderHight = 30;
int SliderX = ScaleStartX;

boolean CanDrag = false;

void setup()
{
    size (800,200);

    String port = Serial.list()[0];
    serial = new Serial(this, port, 9600);

    smooth();
    background(0);

    f = createFont("Times New Roman",14); //Create
Font
}

void draw()
{
    background(0);

    textFont(f,36); // Specify
font to be used
    fill(255); // Specify
font color
    text("Servomotor angle",width/2-150,height/2-
30); // Display Text

    textFont(f,36);
    fill(255);
    text("Current angle: "
str(GetServoAngle()),width/2+100,height/2+70);

```

```

    strokeWeight(5);
    stroke(204, 102, 0);
    line(ScaleStartX, ScaleY, ScaleEndX, ScaleY);

    strokeWeight(3);
    stroke(255, 255, 0);
    line(ScaleStartX, ScaleY - MarkLen/2,
ScaleStartX, ScaleY + MarkLen/2);
    line(ScaleStartX + (ScaleEndX-ScaleStartX)/2,
ScaleY - MarkLen/2, ScaleStartX + (ScaleEndX-
ScaleStartX)/2, ScaleY + MarkLen/2);
    line(ScaleEndX, ScaleY - MarkLen/2, ScaleEndX,
ScaleY + MarkLen/2);

    if (CanDrag)
    {
        SliderX = mouseX;
        if (mouseX < ScaleStartX) SliderX =
ScaleStartX;
        if (mouseX > ScaleEndX) SliderX = ScaleEndX;
    }

    strokeWeight(0);
    fill(255, 0, 0);
    rect(SliderX-SliderWidth/2, ScaleY-
SliderHight/2, SliderWidth, SliderHight);

    println(GetServoAngle());
}

int GetServoAngle()
{
    return int((SliderX-ScaleStartX)*180/(ScaleEndX-
ScaleStartX)); // (SliderX-SliderWidth/2)/(ScaleEndX-
ScaleStartX);
}

boolean isMouseOnSlider()
{
    return SliderX-SliderWidth/2 <= mouseX &&
ScaleY-SliderHight/2 <= mouseY && SliderWidth +
(SliderX-SliderWidth/2) >= mouseX && SliderHight +
(ScaleY-SliderHight/2) >= mouseY;
}

```

```

void mousePressed()
{
    if (isMouseOnSlider()) CanDrag = true;
}

void mouseReleased()
{
    CanDrag = false;
}

```

A.4 Текст программы для Arduino/ серво

```

#include <Servo.h>

long  commandValue;  //  данные, поступающие с
последовательного порта
int ledPin = 13;
int ServoPos = 0;

Servo servo_9;

void setup() {
    servo_9.attach(9, 500, 2500);
    pinMode(ledPin, OUTPUT);  // режим на вывод
данных
    Serial.begin(9600);
}

void loop() {
    if (Serial.available()) {
        commandValue = Serial.parseInt();

        Serial.println(commandValue);

        if (commandValue != 0 && commandValue > 0) {
            ServoPos = commandValue*180/400;
            servo_9.write(ServoPos);
        }
    }
    delay(10);  // задержка перед следующим чтением
данных
}

```

A.5 Текст програми для функції map

```

import processing.serial.*;
import cc.arduino.*;

Arduino arduino;

int PotentiometerPin = 0;
int PotentiometerAnalogValue;
int AnalogValuePercent = 0;

PFont f;

int ScaleStartX = 40; int ScaleY = 100;
int ScaleEndX = 760;

int MarkLen = 20;

int SliderWidth = 20; int SliderHight = 30;
int SliderX = ScaleStartX;

void setup()
{
    size (800,200);

    arduino = new Arduino(this, Arduino.list()[0],
115200);
    print(Arduino.list()[0]);

    smooth();
    background(0);

    f = createFont("Times New Roman",14); //Create
Font
}

void draw()
{
    PotentiometerAnalogValue =
arduino.analogRead(PotentiometerPin);
    //AnalogValuePercent = 18; for test
    AnalogValuePercent =
int(map(PotentiometerAnalogValue, 0, 1023, 0, 100));

```

```

        SliderX          =          int(ScaleStartX          +
AnalogValuePercent*(ScaleEndX - ScaleStartX)/100);

        background(0);

        textFont(f,36);                                // Specify
font to be used
        fill(255);                                       // Specify
font color
        text("Analog      Value",width/2-150,height/2-30);
// Display Text

        textFont(f,36);
        fill(255);
        text("Current      value:      "              +
str(AnalogValuePercent),width/2+100,height/2+70);

        strokeWeight(5);
        stroke(204, 102, 0);
        line(ScaleStartX, ScaleY, ScaleEndX, ScaleY);

        strokeWeight(3);
        stroke(255, 255, 0);
        line(ScaleStartX,      ScaleY      -      MarkLen/2,
ScaleStartX, ScaleY + MarkLen/2);
        line(ScaleStartX + (ScaleEndX-ScaleStartX)/2,
ScaleY      -      MarkLen/2,      ScaleStartX      +      (ScaleEndX-
ScaleStartX)/2, ScaleY + MarkLen/2);
        line(ScaleEndX, ScaleY - MarkLen/2, ScaleEndX,
ScaleY + MarkLen/2);

        strokeWeight(0);
        fill(255, 0, 0);
        rect(SliderX-SliderWidth/2,                                ScaleY-
SliderHight/2, SliderWidth, SliderHight);

        PotentiometerAnalogValue          =
arduino.analogRead(PotentiometerPin);
        println(AnalogValuePercent);
    }

```

A.6 Текст програми передачі інформації за допомогою Firmata

```
import processing.serial.*;
```

```

import cc.arduino.*;

Arduino arduino;

int red;
int green;
int blue;

void setup() {
    size(512, 200);
    arduino = new Arduino(this, Arduino.list()[0],
115200);
    print(Arduino.list()[0]);

    arduino.pinMode(9, Arduino.OUTPUT);
    arduino.pinMode(10, Arduino.OUTPUT);
    arduino.pinMode(11, Arduino.OUTPUT);

}

void draw() {
    red = constrain(mouseX/2, 0, 255);
    green = constrain(255 - mouseY, 0, 255);
    blue = constrain(255 - mouseX/2, 0, 255);
    background(red, green, blue);
    arduino.analogWrite(9, red);
    arduino.analogWrite(10, green);
    arduino.analogWrite(11, blue);
}

if (rgbCounter == 3)
{
    analogWrite(red, RedValue);
    analogWrite(green, GreenValue);
    analogWrite(blue, BlueValue);

    rgbCounter = 0;
    Serial.flush();
    delay(1000);
}

delay(20);
}

```

ДОДАТОК Б
Слайди презентації

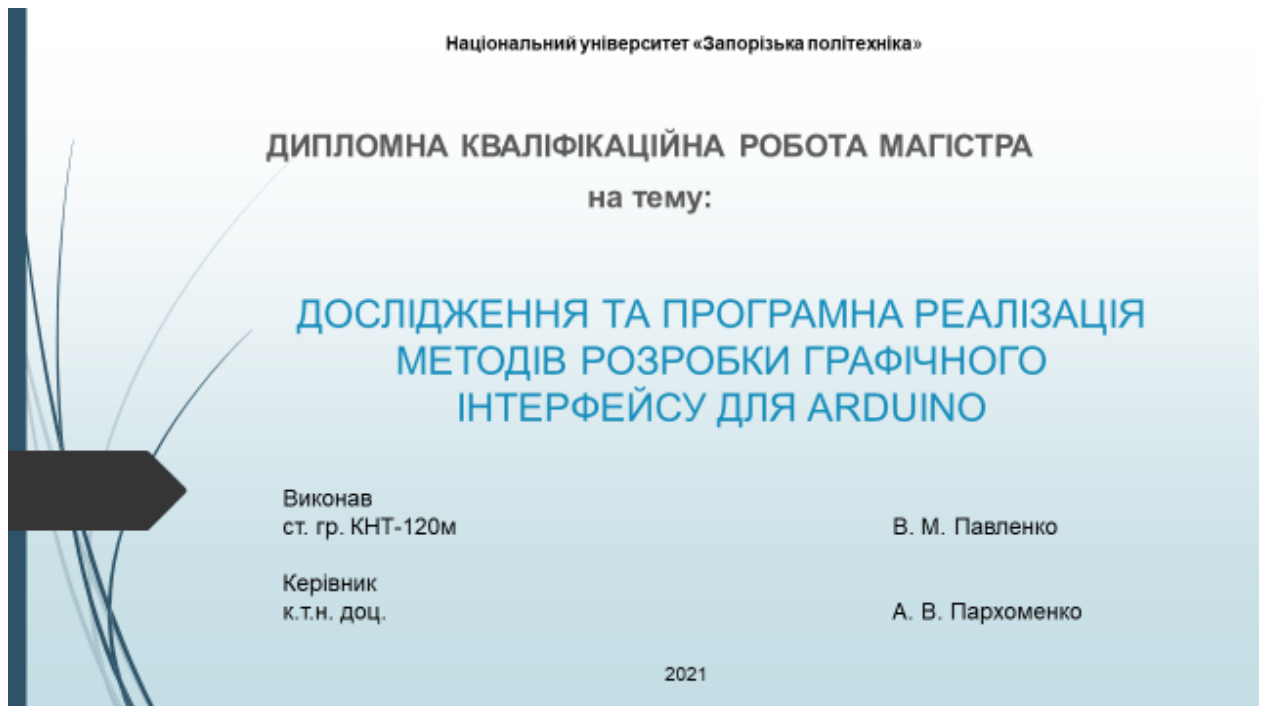


Рисунок Б.1 – Слайд 1



Рисунок Б.2 – Слайд 2

Мета та задачі роботи

3

Метою роботи є дослідження та розробка прототипу СДА на основі методології повторного використання та хмарних технологій для підвищення ефективності взаємодії компонентів СДА.

Задачі роботи:

- дослідити методи та засоби прототипування СДА;
- розробити хмарну архітектуру прототипу СДА;
- обрати апаратні та програмні засоби розробки;
- виконати програмну реалізацію;
- виконати дослідження віртуального та фізичного прототипів СДА.

Рисунок Б.3 – Слайд 3

Дослідження існуючих прототипів СДА

4

№ п/п	Назва стенду/ лабораторії	Розробник	Режим доступу	Основне обладнання	Вартість
1	GAMMA Training Kit GTK 5.1	Siemens	офлайн	Siemens	++++
2	KNX-EIB Experimenter with Simulator	Wuekro	офлайн	Siemens	++++
3	Демостенд MyHOME	ПКФ МОТОР, Запоріжжя	офлайн	Legrand	+++
4	Демостенд GIRA	GIRA	офлайн	GIRA	+++
5	Макет Розумного будинку	Intel Corp.	офлайн	Intel Corp., Arduino	+++
6	ВЛ голосового управління	Корнельський університет, США	онлайн	ATmega	++
7	Smart House & IoT	НУ «Запорізька політехніка»	онлайн	Arduino, Raspberry	++
8	RGB Led laboratory & Servomotor	Університет дистанційної освіти, Мадрид	онлайн	Arduino	++

Рисунок Б.4 – Слайд 4

Критерії та методика вибору платформи розробки

5

Критерії вибору:

- параметри контролера (швидкість, енергоспоживання, периферійні пристрої);
- доступні функції (в т.ч. підтримувані протоколи зв'язку, пам'ять та ін.);
- периферійні пристрої, що вбудовані на платформі;
- технічна підтримка (набір інструментів, необхідний для створення ПЗ, готові рішення).

Характеристики	Середовище розробки			
	Arduino IDE	B4R	CodeBlocks	Processing
Простота	+	-	-	+
Підтримка	+	+/-	+/-	+
Мова програмування	C / Wiring	Basic	C	Processing, /Java
Доступність	Open-source	Open-source	Open-source	Open-source



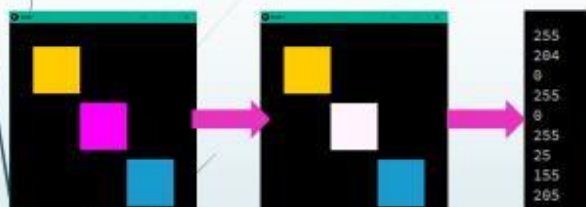
Узагальнена методика вибору платформи розробки

Рисунок Б.5 – Слайд 5

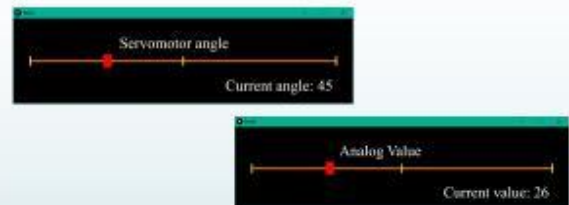
Розробка графічного інтерфейсу для Arduino

6

Processing to Arduino



Arduino to Processing



Результати моделювання в Tinkercad



Рисунок Б.6 – Слайд 6

Хмарна архітектура Microsoft Azure

7

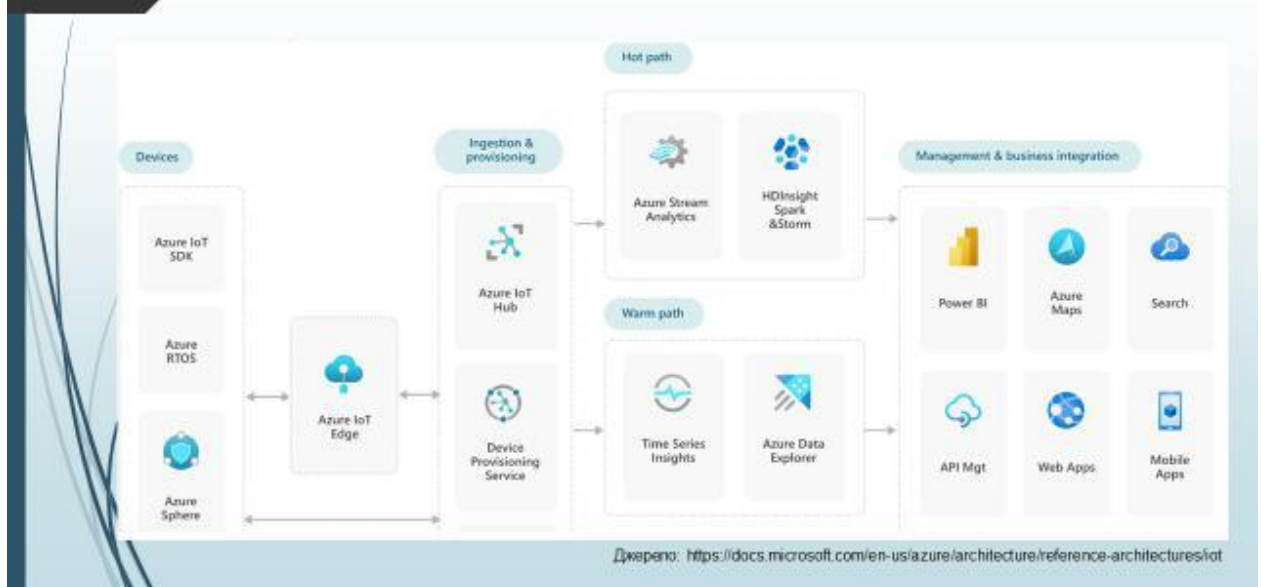


Рисунок Б.7 – Слайд 7

Концептуальна модель прототипу СДА

8

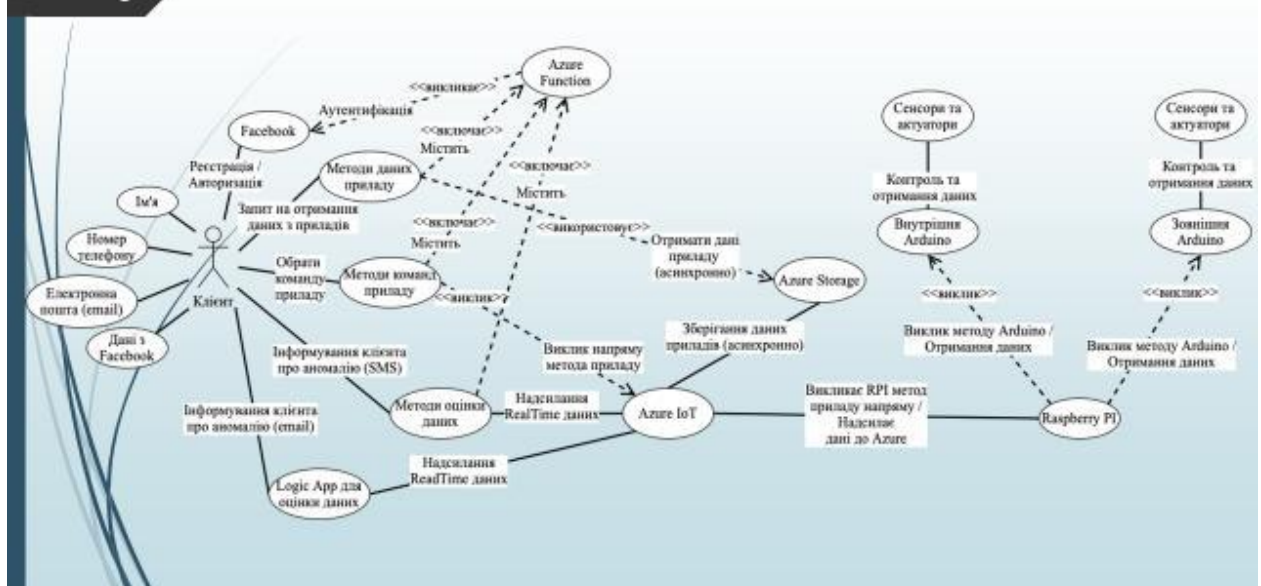


Рисунок Б.8 – Слайд 8

Компоненти мережі СДА та їх зв'язки

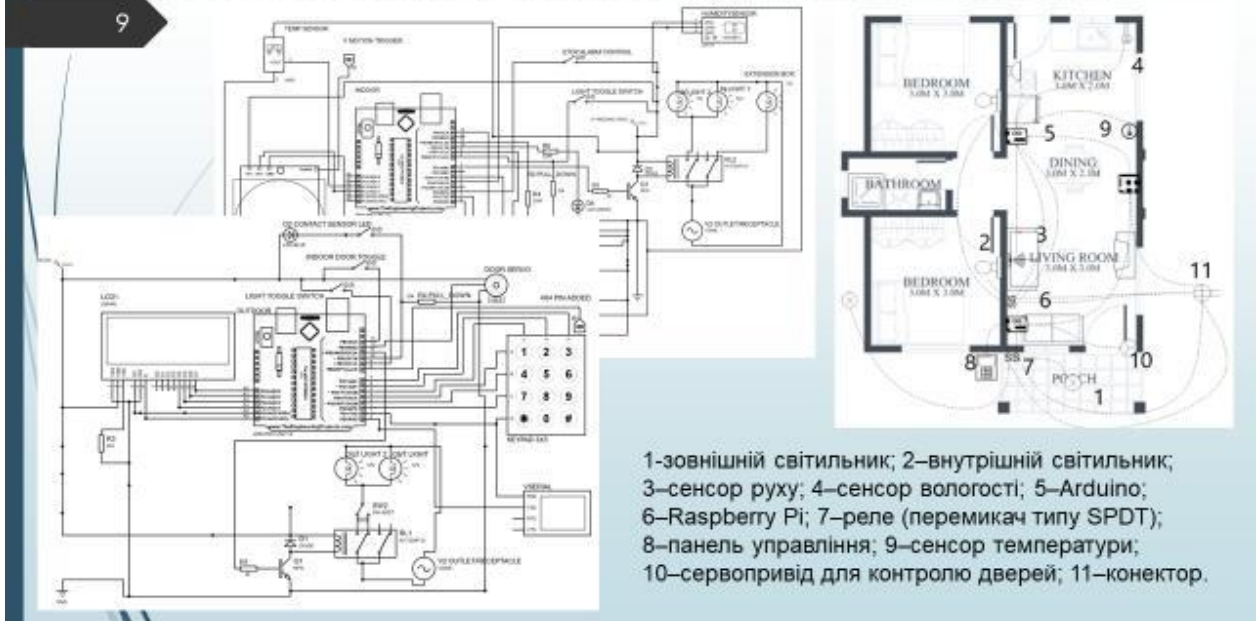


Рисунок Б.9 – Слайд 9

Моделювання процесів управління в Proteus

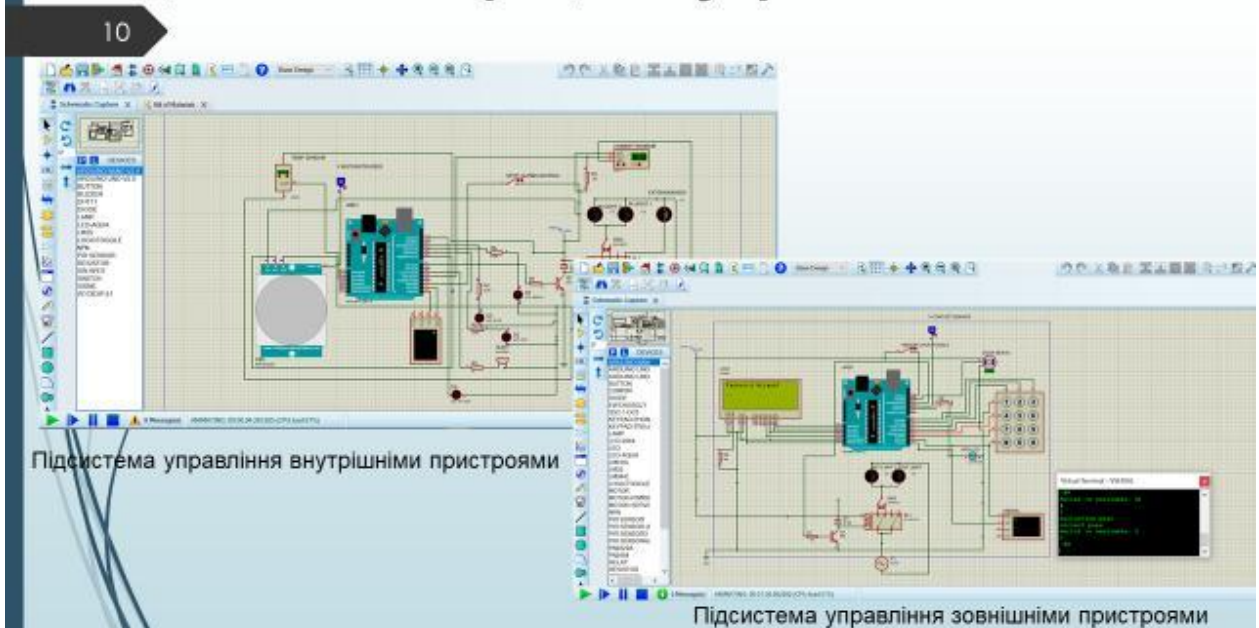


Рисунок Б.10 – Слайд 10

Розробка інтерфейсу клієнтського застосунку

11

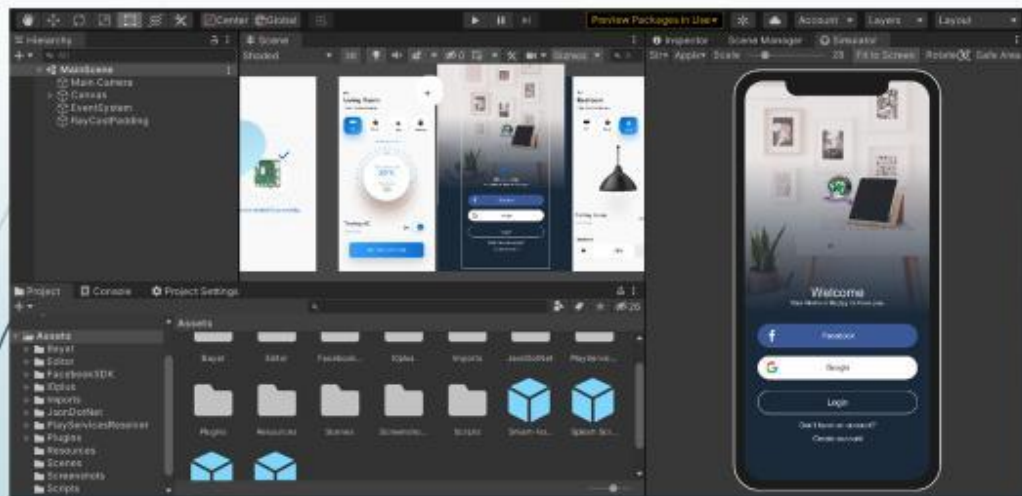


Рисунок Б.11 – Слайд 11

Інформаційна структура клієнтського застосунку

12



Рисунок Б.12 – Слайд 12

Тестування прототипу СДА

13



Моніторинг логів тестування телеметрії на стороні СДА



Віддалений моніторинг логів тестування телеметрії



Віддалений моніторинг метрик системи

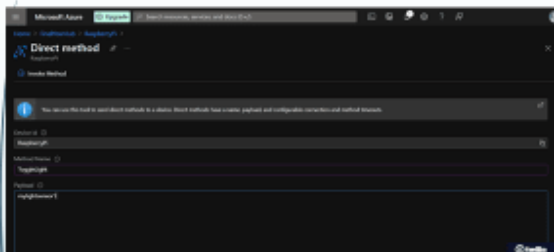


Віддалений моніторинг даних часових рядів

Рисунок Б.13 – Слайд 13

Тестування прототипу СДА

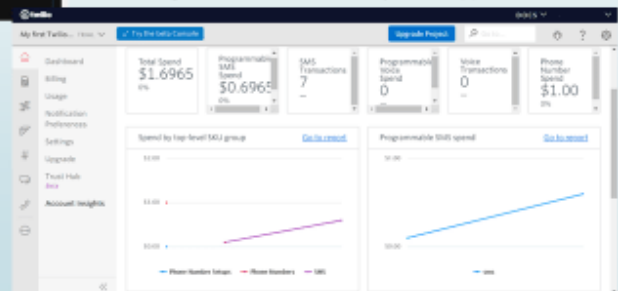
14



Тестування пристроїв СДА прямим методом

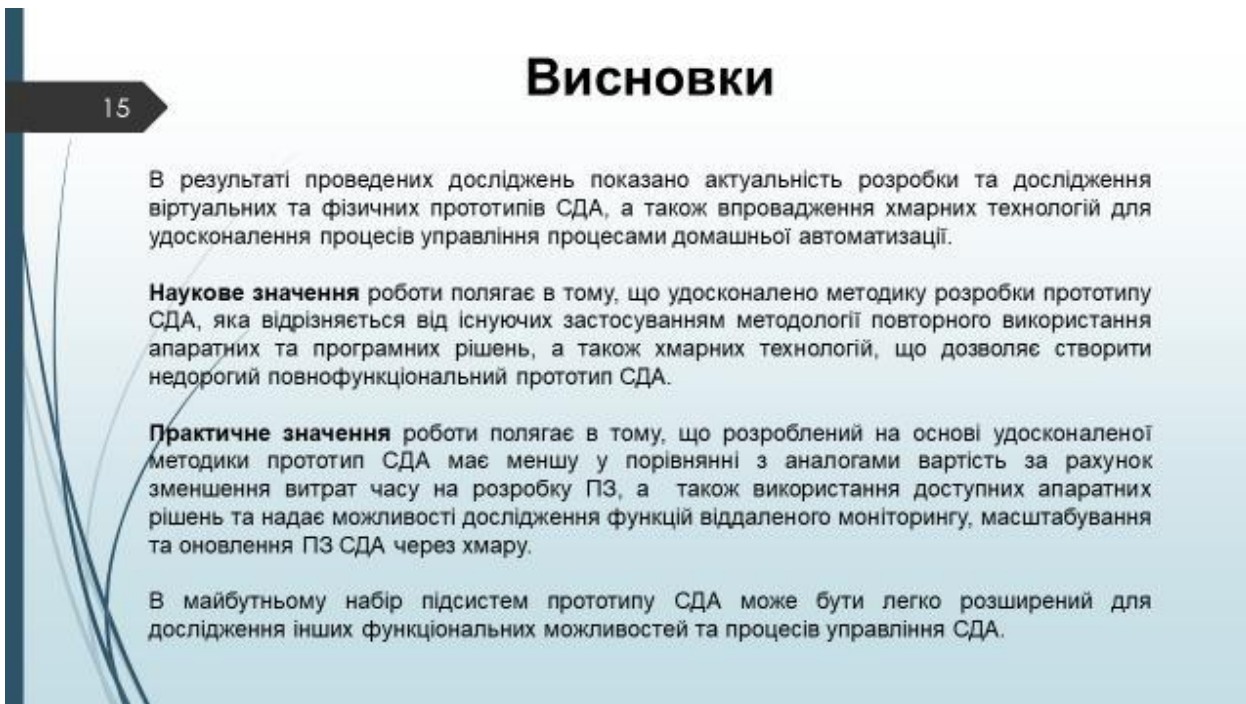


Тестування SMS сповіщень за допомогою Twilio API



Тестування витрат на порталі Twilio

Рисунок Б.14 – Слайд 14



Висновки

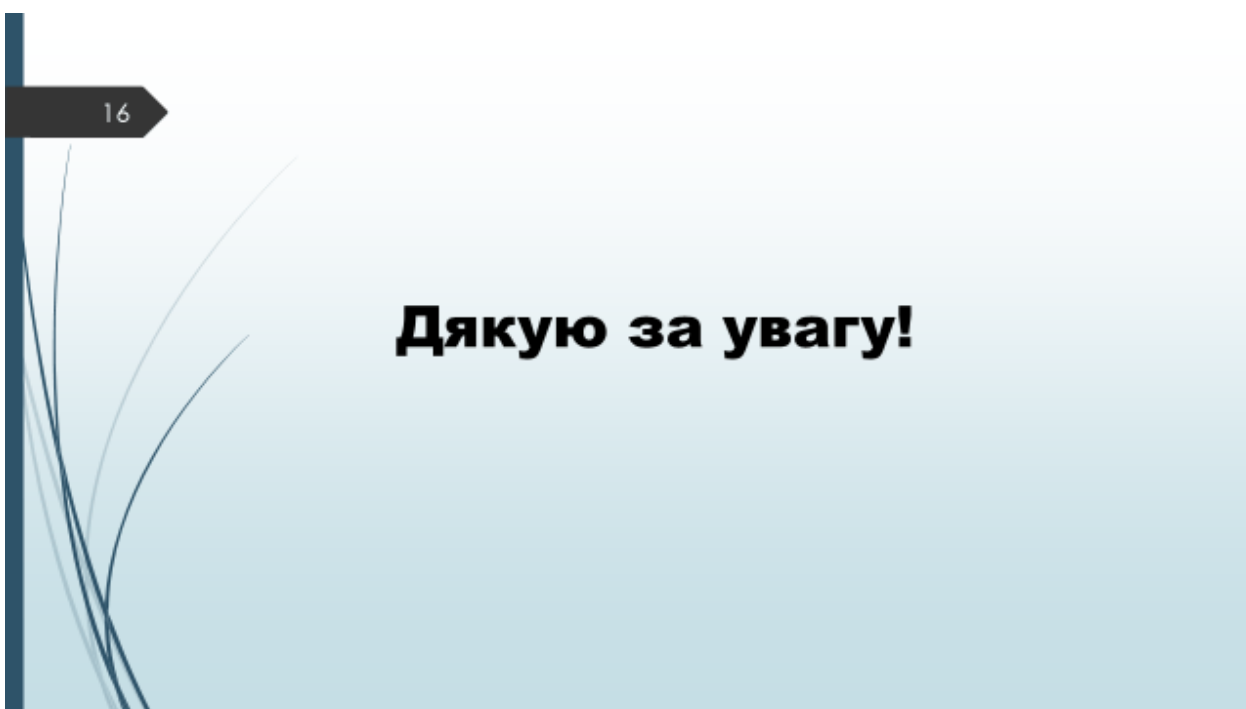
В результаті проведених досліджень показано актуальність розробки та дослідження віртуальних та фізичних прототипів СДА, а також впровадження хмарних технологій для удосконалення процесів управління процесами домашньої автоматизації.

Наукове значення роботи полягає в тому, що удосконалено методику розробки прототипу СДА, яка відрізняється від існуючих застосуванням методології повторного використання апаратних та програмних рішень, а також хмарних технологій, що дозволяє створити недорогий повнофункціональний прототип СДА.

Практичне значення роботи полягає в тому, що розроблений на основі удосконаленої методики прототип СДА має меншу у порівнянні з аналогами вартість за рахунок зменшення витрат часу на розробку ПЗ, а також використання доступних апаратних рішень та надає можливості дослідження функцій віддаленого моніторингу, масштабування та оновлення ПЗ СДА через хмару.

В майбутньому набір підсистем прототипу СДА може бути легко розширений для дослідження інших функціональних можливостей та процесів управління СДА.

Рисунок Б.15 – Слайд 15



Дякую за увагу!

Рисунок Б.16 – Слайд 16