Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Галузь знань 12 – Інформаційні технології

Спеціальність 123 –Комп'ютерна інженерія

на тему «Протокол для центрального контролера в кіберфізичній системі «розумний будинок»»

КвРКІ. 016006.20.01.12 ПЗ

Виконав: студент 2 курсу, група КІ2м-20-1 _____ Зелінський О.О.
Підпис Ініціали, прізвище

Керівник доктор техн. наук, професор _____ Савенко О.С.
Науковий ступінь, вчене звання Підпис Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КІІС, д.т.н., проф.
Т.О. Говорущенко
__ _____ 2022 р.

Хмельницький, 2022

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Освітній рівень МАГІСТР

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма ОСВІТНЬО-НАУКОВА ПРОГРАМА «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ

Зав. кафедри Т.О.Говорущенко

" 01 " 09 2021 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ)

Зелінському Олександру Олеговичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Протокол для центрального контролера в кіберфізичній системі «розумний будинок»

Керівник проекту (роботи) Савенко О.С., д.т.н., професор
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 06.01.2022 р. № 1

2. Строк подання студентом проекту (роботи) на кафедру 03.05.2022 р.

3. Вихідні дані до проекту (роботи) Завдання на дипломне проектування

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Аналіз відомих методів

Проєктування архітектури системи та протоколу взаємодії компонент

Реалізація кроків протоколу

Протокол та його технічне рішення

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прий
Нормоконтроль	Лисенко С.М., професор кафедри КІІС		
Антиплагіат	Нічепорук А.О., доцент кафедри КІІС		

7. Дата видачі завдання «_06_»_____09_____2021р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Прим
1	Вибір напряму дослідження та узгодження тематики ДРМ з керівником	05.09.2021	викон
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	05.10.2021	викон
3	Робота над розділом 1 – аналіз відомих моделей, методів за темою; постановка задачі	05.11.2021	викон
4	Робота над розділом 2 – розробка моделей для *вирішення поставленої задачі*	05.12.2021	викон
5	Робота над науковою статтею	05.01.2022	викон
6	Робота над розділом 3 – розробка методів для вирішення поставленої задачі	15.02.2022	викон
7	Робота над розділом 4 – проектування та розробка ПЗ для вирішення поставленої задачі, експериментальна частина	05.04.2022	викон
8	Оформлення пояснювальної записки згідно вимог	15.04.2022	викон
9	Попередній захист ДРМ	18.04.2022	викон
10	Захист ДРМ на засіданні ЕК	До 15.05.2022	

Студент _____ Зелінський О.О.
Підпис Ініціали, прізвище

Керівник проекту (роботи) _____ Савенко О.С.
Підпис Ініціали, прізвище

РЕФЕРАТ

Тема дипломної роботи: Протокол для центрального контролера в кіберфізичній системі «розумний будинок»

Автор роботи: Зелінський Олександр Олегович

Керівник роботи: д.т.н., професор, Савенко Олег Станіславович

Пояснювальна записка: 93 с., 4 рис., 1 табл., 1 дод., 79 джерел.

ПЕРЕЛІК КЛЮЧОВИХ СЛІВ (6-8) ЧЕРЕЗ КОМУ. Постійний центральний контролер, кіберфізична система, ключова відповідальність для РСС, протокол UPnP, розумний будинок.

Об’єктом дослідження є процес керування за протоколом для центрального контролера в кіберфізичній системі «розумний будинок».

Предметом дослідження є методи проєктування протоколів для центрального контролера в кіберфізичній системі «розумний будинок».

Метою дипломної роботи є проєктування протоколу для центрального контролера в кіберфізичній системі «розумний будинок».

Для розв’язання поставлених задач використовуються основні положення методи теорії комп’ютерних мереж, архітектури комп’ютерів, теорії множин.

Наукова новизна отриманих результатів:

− розроблено метод проєктування протоколів для центрального контролера в кіберфізичній системі «розумний будинок».

На основі проведених досліджень розроблено метод проєктування протоколів для центрального контролера в кіберфізичній системі «розумний будинок».

Практична значимість отриманих результатів полягає у розробленому протоколі для центрального контролера в кіберфізичній системі «розумний будинок», завданням якого має бути оптимізація роботи з додаванням нових можливостей. Завдяки удосконаленому варіанту центрального

мікроконтролера, система «розумний» дім буде функціонувати якісніше та вартість проєкту буде значно меншою.

За темою дипломної роботи опублікована одна публікація у Збірнику наукових праць за матеріалами XIII Всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2021». Хмельницький – 2021. – [79].

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ 5

ВСТУП 6

1 АНАЛІЗ ВІДОМИХ МЕТОДІВ 8

1.1 Аналіз предметної області 8

1.2 Огляд архітектур системи 12

1.3 Аналіз відомих рішень 14

1.4 Постановка задачі 25

1.5 Висновки 25

2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ СИСТЕМИ ТА ПРОТОКОЛУ ВЗАЄМОДІЇ КОМПОНЕНТ 28

2.1 Визначення області дослідження 28

2.2 Протокол Насріна і Редкліффа 29

2.3 Метод розробки протоколів 37

2.5 Протокол Robust 45

2.6 Висновок до другого розділу 53

3 РЕАЛІЗАЦІЯ КРОКІВ ПРОТОКОЛУ 56

3.1 Поєднання елементів інтернету речей 56

3.2 Проектування елементів протоколу 57

3.3 Подальший аналіз та надійність 63

3.4 Висновки 67

4 ПРОТОКОЛ ТА ЙОГО ТЕХНІЧНЕ РІШЕННЯ 68

4.1 Технічне рішення 68

4.2 Реалізація протоколу 70

4.3 Інтеграція протоколів 72
4.4 Висновки до четвертого розділу 74
ВИСНОВКИ 75
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ 77
ДОДАТОК А 86
ДОДАТОК Б 90
ДОДАТОК В 93

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

PCC - Постійний центральний контролер

AP – Точка доступу

RNDM – Надійна методологія розвитку мережі

МКБ – Переривчасті пристрої управління

OTA – Програмування по повітрю

VM – Віртуальна машина

UI – Інтерфейс користувача

HAS – Система домашньої автоматизації

M2M – Машина-машина

WPS – Бездротове захищене налаштування

UPnP – Universal Plug and Play

MQTT – Протокол телеметрії в черзі

COaP – Протокол обмеженого застосування

ВСТУП

Домашня автоматизація не є новою концепцією. Їй майже 50 років. Деякі з найбільш ранніх робіт сходять до 70-х років з такими протоколами, як X10 [1], [2]. Однак, незважаючи на десятиліття інтересу до впровадження автоматизації та комп'ютерного інтелекту додому, домашня автоматизація не наситила ринок і навіть не стала спільною рисою будинків. Не було досягнуто нічого подібного до повсюдності розумних будинків, очікуваних за останні 40-50 років.

Було кілька підходів до домашньої автоматизації в промисловості та дослідженнях. На сьогоднішній день найбільш поширеним було використання контролера, який розміщується в центрі домашньої системи і провід кожного пристрою назад до контролера. Це спостерігається, наприклад, у системах, що використовують X10 [1], [3] та KNX [4], [5]. Цей контролер є центром системи і дозволяє керувати будь-яким пристроєм з цього центрального місця розташування. Це може дозволити одному пристрою керувати іншим і може керуватися з інших місць, але весь контроль здійснюється через цей центральний контролер. Є кілька проблем з цими центральними контролерами. Вони фіксовані на місці і живляться постійно, вони можуть бути єдиною точкою невдачі для простих, але необхідних домашніх світильників, вони можуть бути дорогими, і їх часто важко переналаштувати. Вони можуть заблокувати користувача в конкретну лінійку виробників продукції, так як вони не сумісні з продукцією конкурентів, а це може збільшити вартість самих керованих пристроїв.

В цій роботі визначено топологію PCC, яка визначається як топологія з фіксованим контролером, який постійно знаходиться в центрі системи домашньої автоматизації. Хоча було багато спроб звести до мінімуму його проблеми, всі вони все ще належним чином описані як такі, що використовують PCC.

Актуальність роботи полягає в розробці протоколу для центрального контролера в кіберфізичній системі «розумний будинок».

Метою дипломної роботи є проєктування протоколу для центрального контролера в кіберфізичній системі «розумний будинок».

Поставлена мета досягається розв’язанням таких основних задач:

– проаналізувати відомі методи розробки протоколів для центрального контролера в кіберфізичній системі «розумний будинок»;

– удосконалити протокол для центрального контролера в кіберфізичній системі «розумний будинок»;

– удосконалити метод проєктування протоколу для центрального контролера в кіберфізичній системі «розумний будинок»;

– реалізувати протокол для центрального контролера в кіберфізичній системі «розумний будинок» згідно розроблених рішень.

Об’єктом дослідження є процес керування за протоколом для центрального контролера в кіберфізичній системі «розумний будинок».

Предметом дослідження є методи проєктування протоколів для центрального контролера в кіберфізичній системі «розумний будинок».

Наукова новизна отриманих результатів:

– розроблено метод проєктування протоколів для центрального контролера в кіберфізичній системі «розумний будинок».

Практична значимість отриманих результатів полягає у розробеному протоколі для центрального контролера в кіберфізичній системі «розумний будинок».

Для розв’язання поставлених задач використовуються основні положення методи теорії комп’ютерних мереж, архітектури комп’ютерів, теорії множин.

За темою дипломної роботи опублікована одна публікація у Збірнику наукових праць за матеріалами XIII Всеукраїнської науково-практичної конференції «Актуальні проблеми комп’ютерних наук АПКН-2021». Хмельницький – 2021. – [81].

1 АНАЛІЗ ВІДОМИХ МЕТОДІВ

1.1 Аналіз предметної області

Щоб пом'якшити вартість і енергоспоживання РСС, багато хто скористався більш розповсюджених одноплатформних комп'ютерів, таких як Raspberry Pi.

Ці одноплафорні рішення стають більш потужні, але ще не в точці суперництва на загальному ПК або сервері. Вони мають обмежені Вводи-виводи та розширюваності, як і раніше страждають від проблем, пов'язаних з гнучкістю, юзабіліті та блокуванням виробника.

У багатьох випадках система жорстко закодована за допомогою спеціального програмного забезпечення відповідно до конкретної конфігурації в використання, і будь-які зміни вимагають перепрограмування.

Бездротові інтерфейси надали багато популярних альтернатив для підключення пристроїв до системи домашньої автоматизації з метою підвищення гнучкості, оскільки пристрої можна легко переміщати, покладаючись лише на потужність, будь то мережа або акумулятор [13].

Wi-Fi, 802.11, є мабуть, першим найбільш очевидним вибором тут, і, безумовно, було зроблено багато роботи з його використанням[8], [14], [18]. Wi-Fi пропонує значні переваги домашньої автоматизації. По-перше, він пропонує гнучкість, як уже згадувалося, але він також пропонує досить просте налаштування і легко доступний, особливо в його тепер майже всюдисущій формі 2,4 ГГц. Є деякі проблеми з Wi-Fi, оскільки він не є найбільш енергоефективним бездротовим інтерфейсом [12] і працює переважно в дуже оспорюваному діапазоні 2,4 ГГц [19].

Bluetooth колись вважався життєздатним варіантом для домашньої автоматизації [6], [11], [20], і хоча він працює набагато краще, ніж Wi-Fi з точки зору споживання енергії, він не має діапазону або наскрізного використання Wi-Fi і набагато рідше зустрічається в більш пізніх дослідженнях.

LoRa - це набагато новіша технологія, яка пропонує відмінний асортимент, але менш добре підходить для домашнього середовища. Його діапазон набагато

перевищує потреби більшості домашніх додатків, хоча він показав сумісність в інших додатках для автоматизації та IoT [21] .

Як і LoRa, ZigBee пропонує бездротові інтерфейси в діапазонах нижчих частот, таких як 868 МГц, 915 МГц, але також працює в діапазоні 2,4 ГГц. ZigBee запропонував інтерфейс з низькою енергією на ранній стадії, перш ніж були доступні низькоенергетичні варіанти Bluetooth. Це було популярним рішенням для домашньої автоматизації [7], [10], [22]-[27], навіть для комерційних продуктів.

Мета тут полягає в тому, щоб зменшити витрати, скориставшись масштабами економіки. Виробник або інша третя сторона може розміщувати постійних центральних контролерів у великих центрах обробки даних і пропонувати послугу тисячам клієнтів за набагато нижчими цінами. Як і рішення типу Raspberry Pi, цей захід спрямований на зниження витрат, але без втрати обчислювальної потужності. Іншою перевагою є те, що витрати на час простою можна значно зменшити, збалансувавши навантаження, збалансувавши попит на послуги Центрального контролера. Однак ці підходи в кінцевому підсумку роблять невеликий, легкий локальний контролер, який координує дані всередині будинку і відправляє їх в хмару для обробки. Це додаткове обладнання пом'якшує переваги переходу в хмару в першу чергу.

У той час як традиційний PC пропонує єдину точку відмови, ця вразливість набагато гірша в хмарних підходах, де виробники повинні підтримувати центральні сервери, щоб система продовжувала функціонувати. Це вже призвело до того, що розумні пристрої втратили функціональність через те, що виробники закрили свої сервери [29]-[30]. Хоча це сприятлива модель для виробників, які можуть стягувати постійну плату за обслуговування.

Інші альтернативи PC та традиційні підходи PC дозволяють запускати пристрої в автономному режимі, але, якщо PC знаходиться в хмарі, коли підключення до Інтернету падає, вся функціональність втрачається і користувачі залишаються з деякими скороченими наборами функцій. Крім того, існують побоювання з приводу безпеки щодо того, хто може бачити ваші пристрої та їх дані, а також хто може отримати контроль.

Прийнявши PCC, багато хто намагався обійти ексклюзивність, яку виробники створюють навколо своєї продукції [31],[32]. Цей об’єднує PCC від різних виробників, об'єднаних ще більшим контролером, свого роду Super PCC, який може перекладати повідомлення між різними протоколами, що використовуються кожним виробником. Цей підхід (рисунок 1.1) був найбільш чудовим у роботі Miori та Russo і добре підсумовується, де їх DomoNet PCC переводить зв'язок, що дозволяє пристроям X10 взаємодіяти з пристроями KNX або Jini.

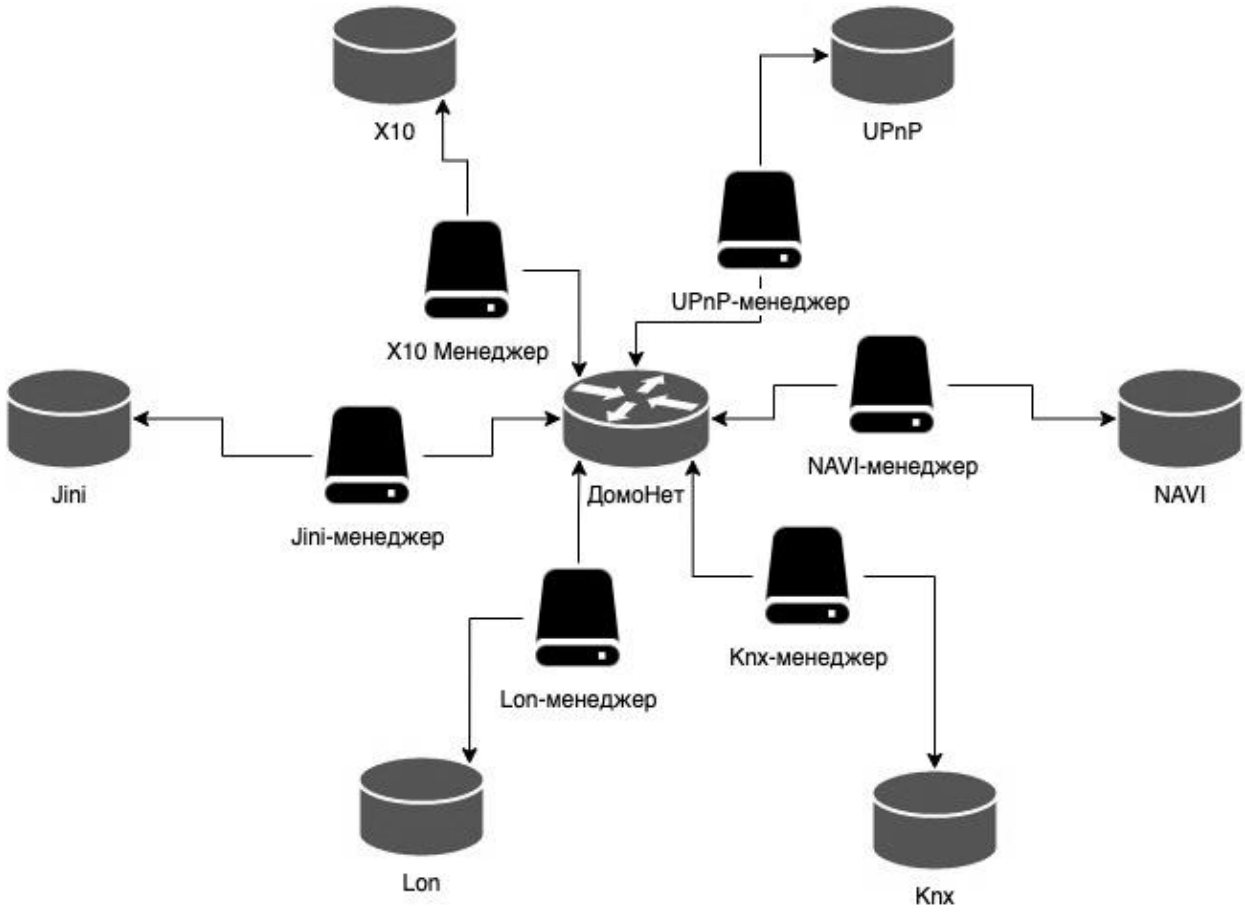


Рисунок 1.1 – Топологія DomoNet Miori та Russo для взаємодії в домашній автоматизації

Це далеко від бачення домашньої автоматизації, запропонованого цим дослідженням, в якому багато пристроїв від багатьох виробників можуть бути з'єднані разом. Деякі виробники розширюють асортимент своєї продукції, яка,

можливо, почалася з однієї розумної лампочки, але тепер включає в себе розумні вимикачі живлення і навіть кухонну техніку. Тим не менш, це мало що робить для поліпшення ситуації, оскільки він знову вводить стратегію блокування, коли користувачі, які купили один пристрій у виробника, заблоковані в асортименті цього виробника та його функціональних пропозиціях або стикаються з різким збільшенням складності та незручностями експлуатації та обслуговування декількох систем та їх додатків.

Ці комерційні розробки показали новий підхід до оперативної сумісності, який зараз пропонують Google, Amazon і Apple зі своїми голосовими помічниками. Вони надають виробникам можливість зробити свої розумні пристрої сумісними зі своїми голосовими помічниками, тим самим досягаючи сумісності між сумісними пристроями. Однак це знову з'єднує проблему PCC, при цьому всі три помічники залежать від хмари для операцій. Це викликає більше побоювань, але також пропонує меншу сумісність, ніж попередня робота.

Хоча було багато підтверджень недоліків PCC, про що свідчать різні спроби поліпшити традиційний підхід, слід визнати, що поточний масив варіантів дещо безладний. Вони часто залежать від жорстко закодованих рішень, які не будуть легко масштабуватися до масового виробництва або пропонувати доступні рішення. Багато додатків залишать споживачів збентеженими щодо того, які пристрої будуть або не будуть працювати разом. Крім того, PCC пропонує значні витрати, які зроблять їх непривабливими для користувачів, або вони знизять витрати на шкоду функціональності.

Тому варто розглянути нову топологію для систем домашньої автоматизації. Топологія без PCC була б бажаною, враховуючи всі недоліки традиційної топології на основі PCC та останні комерційні розробки, які також відійшли від звичайної концепції PCC.

Кращим і успішнішим підходом до домашньої автоматизації було б не використовувати PCC. Він буде більш орієнтованим на користувача, дозволяючи просту установку, налаштування, обслуговування та щоденне використання.

Споживачі повинні мати можливість купувати пристрої у будь-якого виробника і легко інтегрувати їх у власну систему самостійно. Це означає ліквідацію РСС для забезпечення кращого користувацького досвіду, шляхом:

1) зниження витрат, оскільки контролер не потрібно купувати і живити;

2) спрощення користувацького досвіду, оскільки потрібно менше підключення;

3) підвищення гнучкості, оскільки пристрої можуть бути додані або видалені, не перебуваючи в полоні одного виробника;

4) звільнення споживачів від вибору найкращих пропозицій з усіх виробників.

Це дослідження спрямоване на усунення РСС з системи домашньої автоматизації (HAS) і запропонувати систему, яка може бути недорогою, зручною і з відкритим вихідним кодом (для підтримки сумісності між виробниками). Однак видалення РСС не є тривіальним завданням, і не є кроком, який можна зробити без ретельного розгляду, оскільки РСС традиційно виконала кілька ключових обов'язків в рамках HASS.

1.2 Огляд архітектур системи

Засоби перерозподілу обов'язків РСС є, в значній мірі, метою і результатом цього дослідження, і тому дає деякі вказівки щодо того, як РСС може бути усунена. Рисунок 1.2 можна вважати чимось на зразок дорожньої карти для цієї роботи, оскільки він деталізує обов'язки, які необхідно усунути, і де це дослідження переклало ці обов'язки. Нова архітектура на Рисунку 1.2 піднімає прості пристрої віддаленого інтерфейсу до нової ролі ППУ. У цьому випадку ППУ показані у вигляді смартфонів, але можуть в рівній мірі включати настільні комп'ютери, ноутбуки, планшети і навіть смарт-годинники. Ці ППУ не схожі на РСС, оскільки вони пропонують лише інтерфейс до існуючої системи, але система не залежить від ППУ для роботи.

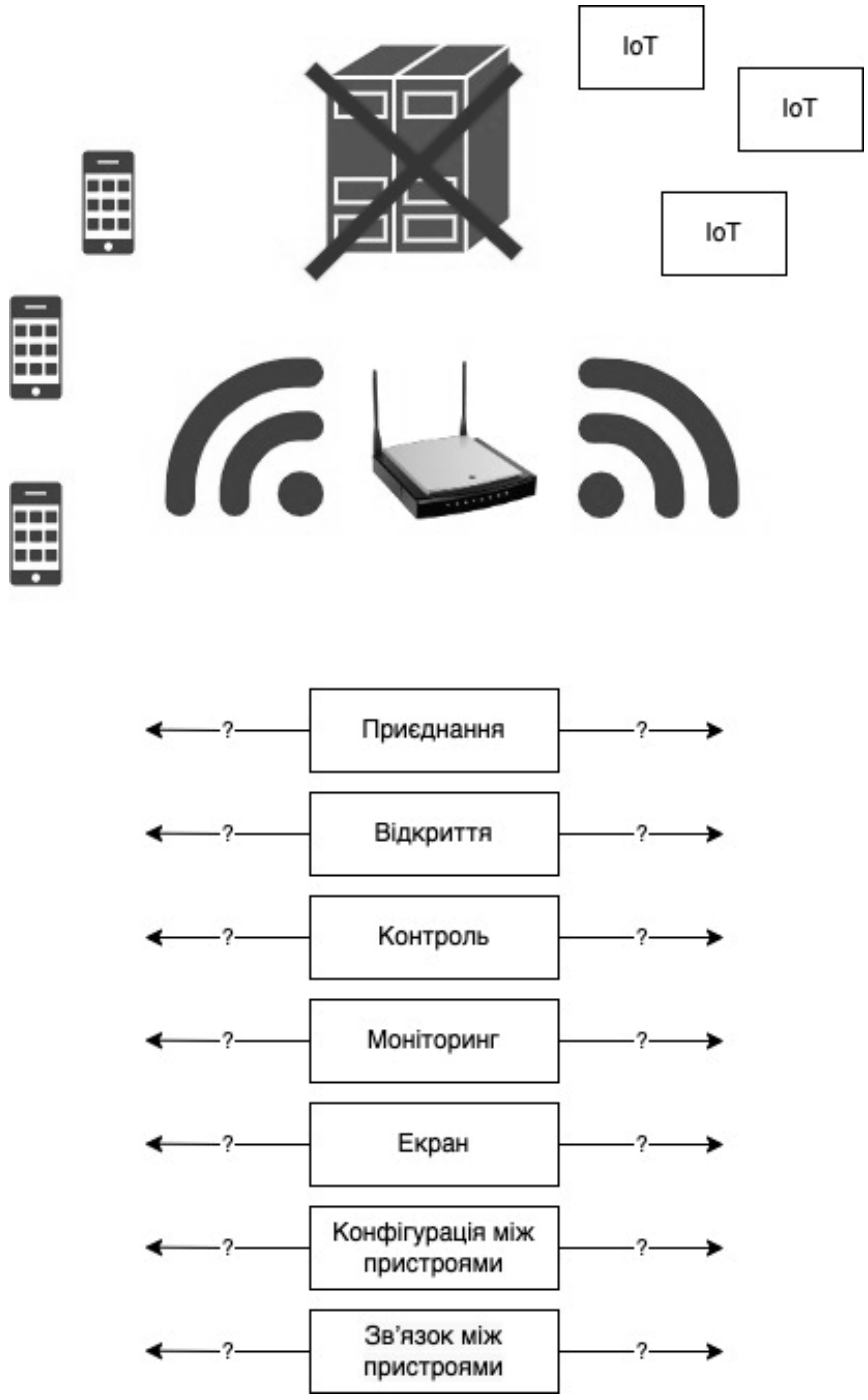


Рисунок 1.2 – Перерозподіл обов'язків постійного центрального контролера

Обов'язки РСС можна легко визначити, попередньо розглянувши її ключову функцію: управляти та інтегрувати розумні пристрої всередині будинку, а потім розглянути життєвий цикл об'єкта його функціонально: розумні IoT-пристрої. Життєвий цикл тут відносно простий, пристрій купується і спочатку повинен бути підключений до мережі. Традиційна система домашньої автоматизації буде покладатися на РСС на цьому етапі приєднання. Після того, як цей пристрій буде

підключено до мережі, тепер він повинен бути виявлений іншими пристроями, щоб дозволити з'єднання. Тут знову ж таки, РСС буде обробляти необхідні введення.

Нарешті, новий пристрій може бути пов'язаний з іншими пристроями, щоб дозволити зв'язок між пристроями, який часто називають M2M. Таким чином, наприклад, кондиціонер і контролер завіси можуть змінити свою поведінку на основі статусу іншого пристрою, такого як гаражні двері або датчик температури. Хоча це дослідження не пішло на великий розгляд цих останніх двох функцій, воно має їх твердо на увазі, пропонуючи альтернативи дисплею, моніторингу та контролю, таким чином, що дуже мінімальна робота над дизайном та деякі основні програмування необхідні для забезпечення цих функцій.

1.3 Аналіз відомих рішень

РСС є проблематичним, як було показано, і було розглянуто кілька різних підходів до вирішення цієї проблеми, але в кінцевому підсумку теза цього дослідження полягає в тому, що РСС можна видалити, а його ключові обов'язки перерозподіляються по новій комбінації МКБ та розумних пристроїв IoT без втрати функціональності. Це передбачає нову топологію для систем домашньої автоматизації (HAS) і вимагатиме нової парадигми взаємодії всередині системи.

Як зазначалося вище, основна увага буде приділена першим п'яти обов'язкам: приєднання, відкриття, моніторинг, відображення та контроль. Кожнен з цих обов'язків буде розглядатися по одному, при цьому відображення і контроль розглядаються разом, оскільки вони тісно пов'язані і дещо перекриваються з моніторингом.

Першою ключовою відповідальністю для РСС було приєднання, тобто прийняття нового пристрою та підключення його до HAS. Це, очевидно, життєво важливий елемент, без якого жодна система не могла б функціонувати. Він є частиною початкової конфігурації, налаштовуючи пристрій на осяжне майбутнє терміну служби пристрою. Нарешті, він контролює взаємодію в один з найбільш

вразливих часів для розумного пристрою і Системи, коли зловмисне втручання може знайти отвори, щоб скористатися, і поставити під загрозу безпеку Системи.

Приєднання викликає найбільше занепокоєння для HAS за допомогою Wi-Fi або інших бездротових протоколів, але деякий процес все ще може знадобитися для дротової системи. Однак це може бути в значній мірі покрито наступним розглядом обов'язків Discovery. Бездротові системи набагато менш тривіальні, оскільки фізичний транспортний носій є спільним і відкритим, тому облікові дані шифрування повинні бути обмінені. Цей обмін створює реальну проблему для обмежених пристроїв. Це особливо складно для недорогих пристроїв, де в бюджеті мало місця для додавання додаткових елементів управління інтерфейсом лише для полегшення того, що, по суті, є одноразовим завданням, яке не є частиною повсякденного використання пристрою.

Бездротове захищене налаштування (WPS) було однією з ранніх пропозицій, щоб спростити цей процес, користувачам просто довелося натиснути кнопку, як правило, на бездротовому маршрутизаторі, а потім підтвердити з'єднання на пристрої.

Однак цей протокол впав у немилість в останнє десятиліття через серйозні вразливості безпеки, які він вводить в мережу [36], [37].

Чотири основні підходи були помічені в дизайні пристроїв домашньої автоматизації. Вони присвячені IO, міст, постійний центральний контролер, і мінімалістичний підхід.

При такому підході додаткове обладнання вводу-виводу використовується виключно з метою безпечного приєднання до локальної мережі. Виділені протоколи вводу-виводу часто відрізняються від основного протоколу зв'язку (Wi-Fi). Наприклад, NFC [38], Bluetooth [11], [39] або повнофункціональний інтерфейс користувача (з екраном і клавіатурою або подібним управлінням входом) були включені в пристрої IoT, які в основному спілкуються за допомогою Wi-Fi, виключно для одноразової події приєднання.

Наприклад, Chen, Pan і Li [38] впровадили NFC-теги в пристроях, які можуть бути прочитані смартфоном, здатним NFC підключатися і приєднуватися до

пристроїв до мережі. Необхідність фізичного переміщення користувача до кожного пристрою для того, щоб прочитати тег NFC, робить Систему менш придатною для використання, ніж механічний перемикач. Хоча додавання цих тегів до пристрою має мінімальні витрати, це обмежує сумісність Системи з пристроями, що підтримують NFC. Нарешті, їх система все ще повністю залежить від постійного центрального контролера, щоб зробити з'єднання шляхом обробки даних, прочитаних смартфоном.

Інші, такі як Piyare і Tazil [11], використовували Bluetooth як звичайний протокол зв'язку, але тільки між телефоном і постійним центральним контролером. Перевага цього полягала в тому, що кінцеві пристрої не потребували апаратного забезпечення Bluetooth і тому були дешевшими, але це відбулося за додаткову вартість, складність і негнучкість постійного центрального контролера. Кумар і Лі [39], які також використовували Bluetooth, впровадили мікро-сервери на дошках Arduino з з'єднаннями Bluetooth і Ethernet. Bluetooth сильно обмежений своїми можливостями діапазону, і, хоча це може бути перевагою в будинку, обмежуючи зовнішні входи або атаки, великі будинки будуть боротися. Крім того, "безпека", що забезпечується обмеженим діапазоном Bluetooth, може бути відтворена на Wi-Fi або Ethernet з простими мережевими архітектурами.

Виділений io добре працює для великих, більш складних пристроїв, де додаткове обладнання для вводу-виводу є тривіальним або вже присутнє. Наприклад, такі пристрої, як розумні холодильники від LG або Samsung, які, по суті, вбудовують планшет у пристрій з повнофункціональним інтерфейсом користувача. Однак такий підхід не дуже добре масштабується до простих пристроїв, де це значно збільшить розмір, складність і вартість. Знову ж таки, дивлячись на розумні холодильники, io вже присутній, і будь-яка робота з розробки, необхідна для підтримки приєднання, є мінімальною вартістю до пристрою за 2000 доларів США+ . Однак додавання навіть кількох кнопок і РК-дисплея до розумного комутатора або вилки за 20 доларів матиме більш виражений вплив на вартість.

Другий підхід для вирішення проблеми приєднання до мережі знову використовує додаткове обладнання для перекладу між Wi-Fi і будь-яким іншим протоколом, створюючи таким чином міст. ZigBee був популярним протоколом у цьому підході [24], [26], [40], оскільки він забезпечив приєднання, властиве протоколу [41]. Таким чином, управління пристроєм підтримується пристроєм, що поєднується з таким протоколом, як ZigBee, підключеним до локального маршрутизатора.

Ян і Ден [24] назвали цей міст своїм домашнім шлюзом, який був оснащений модулями для Wi-Fi, Ethernet, GSM і ZigBee. Всі ці модулі підключені до вбудованого мікроконтролера для координації комунікацій. По суті, це PCC, розроблений для того, щоб зробити пристрої ZigBee більш доступними.

У комерційному плані найбільший успіх ZigBee був помічений в лінійці Hue Bulbs від Philips [28]. Найбільша проблема тут для споживачів полягає в тому, що недостатньо просто купити розумну лампочку і забрати її додому, вам також потрібно придбати міст Відтінок, який може коштувати більш ніж у два рази дорожче, ніж вартість лампочки.

Інші протоколи, такі як Insteon і Z-wave, також використовувалися [40], однак, як і ZigBee, в кінцевому рахунку покладатися на інші протоколи та інфраструктури. Ці вторинні протоколи просто об'єднують пристрій IoT до іншого основного протоколу. Це робиться для того, щоб уникнути створення можливостей в пристрої, обробки основного протоколу. Такий підхід лише сприяє підвищенню вартості та складності IoT-пристрої та мережі, в яких вони перебудовують [41].

Третій підхід приєднання до мережі використовує постійний центральний контролер, до якого фізично підключені всі пристрої. Це фізичне з'єднання пропонує невід'ємну перевагу безпеки і зменшує складність пристроїв, але воно жертвує гнучкістю, яку пропонує бездротове з'єднання.

KNX є одним з найбільш зрілих і успішних продуктів, що використовують цей підхід. Він побудований на декількох усталених протоколах і адаптований до таких ситуацій, як домашня автоматизація. Рішення KNX можуть бути дуже дорогими, і користувачі значною мірою залежать від експертів-техніків з

технічного обслуговування та модернізації установки. Лі і Хун [25] інтегрували систему KNX з пристроями ZigBee, намагаючись запропонувати більш дешевий шлях для оновлення системи KNX і запропонувати бездротові пристрої. Їхня робота справедливо визначила обмеження KNX, але намагалася врятувати стару ідею, а не переходити до нових рішень.

Більшість реалізацій центрального контролера, такі як KNX, є дротовим рішенням, але всі підходи покладаються на ПК або сервер, як постійний центральний контролер, який постійно підключається та живиться. Це також вимагає значної кількості обчислювальної потужності в кожному пристрої [42]. Це означає, що в якості рішення це дуже дорого: пристрої дорогі PCC дорогі, експлуатаційні витрати високі, а коригування складні і дорогі. Виробники швидко блокують користувачів, обмежуючи сумісність пристроїв, дозволяючи їм стягувати дуже високі ціни за прості потреби або незначні оновлення. Наприклад, інтерфейс користувача KNX може легко коштувати понад 1500 доларів [43].

Такі системи не легко змінюються, і користувачі швидко замикаються в єдину лінійку продуктів, збільшуючи витрати. У той час як окремі пристрої простіші, загальна складність системи збільшується, а її гнучкість знижується, ставлячи під загрозу будь-які вигоди для користувацького досвіду.

Нарешті, мінімалістичний підхід, запропонований Насріна і Редкліффа [44], використовує Wi-Fi і пов'язані з ним протоколи безпеки (наприклад, WPA, WPA2), але, на відміну від інших підходів, усуває необхідність складних і громіздких інтерфейсів користувача. Цей підхід використовує можливості точки доступу смартфона для встановлення початкового тимчасового, але безпечного підключення для передачі облікових даних для локальної домашньої мережі. Цей підхід вимагає мало або взагалі не вимагає регулювання існуючих керованих пристроїв і надзвичайно масштабується до простіших пристроїв, таких як вимикачі мережі.

Мінімалістичний підхід, з його протоколом, що обіцяє безкомпромісну безпеку за зниженою ціною і складністю, є найкращим підходом. Однак такий підхід не був реалізований. Насрін запропонував лише простий доказ концепції,

який не автоматизує функцію точки доступу телефону і вимагає уваги для поліпшення інтерфейсу користувача. Пропонований інтерфейс вимагає від користувача високої технічної компетентності і не спрощує процес і не підтримує позитивний користувальницький досвід. З такими змінами цей підхід буде негайно використаний для індустрії IoT.

Незалежно від того, чи використовується безпроводова або дротова мережа, пошук пристрою буде потрібно без РСС, і ця відповідальність не може бути знята разом з РСС. Відкриття пристроїв через мережу було активною областю досліджень протягом деякого часу, і існують різні методи, кожен з яких обслуговує унікальні проблеми різних середовищ. Середовищем, що представляє особливий інтерес для цієї тези, є будинок, який, як правило, має одну або кілька бездротових локальних мереж для обслуговування всього будинку і між варіантами Wi-Fi і Ethernet, бездротовий створює найбільше проблем. Таким чином, будь-яке рішення, достатнє для вирішення проблем Wi-Fi, повинно бути достатнім, без особливих зусиль, для задоволення потреб дротового рішення Ethernet. Оскільки існує кілька протоколів, які вже розглядають відкриття пристрою, наступний огляд буде згрупований в основному навколо різних базових протоколів, на яких побудована існуюча робота.

Протокол UPnP вже давно є улюбленим для домашньої автоматизації, особливо для простого протоколу виявлення послуг включені в його стек [45]- Чжан та ін.[46] об'єднали UPnP з OSGi для розробки архітектури системи управління, яка вела реєстр виявлених пристроїв, які потім можна було шукати, щоб знайти потрібний пристрій. Цей етап реєстрації є цінним, але система фокусується на більшій мережі з декількома контролерами і декількома домашніми шлюзами. Це не дозволяє домовласникам легко розробити власну систему або усунути постійного центрального контролера.

Hsu et al. [47] скористалися можливостями відкриття UPnP для розробки структури інтерфейсу для відкриття та управління пристроями в будинку. Фокус на зручному інтерфейсі похвальний і вкрай необхідний. Однак концепція все ще залежить від постійного центрального контролера.

Чжан та ін. використовували UPnP просто для виявлення пристроїв для своєї системи управління [48]. Велика частина протоколу залишається невикористаною і інтегрована лише для обслуговування події відкриття. Ці накладні витрати пом'якшуються за рахунок використання зовнішнього сервера. У той час як сервер використовується для створення «mash-up», де лампочка і датчик світла можуть бути практично об'єднані в одну розумну лампочку, вимоги до навичок користувача все ще високі, а постійний центральний контролер все ще необхідний.

Таким чином, хоча UPnP пропонує багато чого на шляху загальної структури та ідеології для відкриття пристрою, його архітектура залишає його залежним від постійного центрального контролера і залишає його непридатним для досягнення цілей життєздатної системи домашньої автоматизації.

Кім та інші звернулися до DPWS за протоколом виявлення пристрою при розробці домашнього шлюзу IoT [49]. Тим не менш, вони виявили, що він занадто вимогливий для більш обмежених пристроїв, і тому розробили свій власний спеціальний протокол для цих пристроїв, використовуючи пакети UDP. Цей протокол починається з рекламного повідомлення UDP від обмеженого пристрою, що попереджає шлюз для реєстрації пристрою. Цей протокол не дуже надійний і передбачає, що всі передачі є успішними. Крім того, для цього потрібна заздалегідь визначена інформація про пристрій, яка може бути пов'язана з зареєстрованим пристроєм.

У своїй роботі над розробкою структури додатків Kamilaris та ін. розробили спеціальний протокол виявлення, який вимагає, щоб нові пристрої рекламували себе, поки не будуть визнані [50].

Datta et al. [51], запропонували додаток для смартфонів, який реалізує структуру IoT для домашньої автоматизації і навіть продемонстрував, як це може бути реалізовано для персоналізованої охорони здоров'я[52]. Цей підхід, розглядає відкриття як процес відбору, щоб знайти відповідний ресурс для даного завдання, таким чином, що система в значній мірі статична, і всі пристрої вже відомі. Відкриття в сенсі нових доповнень пристрою - це ручний процес.

Більш пізнім доповненням до безлічі протоколів IoT був Протокол обмежених додатків (CoAP), проект RFC, опублікований у 2014 році [53]. Він має багато переваг, включаючи його зосередження на обмежених пристроях, що робить його вимоги до живлення та обчислень мінімальними [54]-[56].

Santos et al. [55], використовували CoAP для розробки системи IoT для датчиків особистого підключення здоров'я, призначених для моніторингу температури, ваги, рівня цукру в крові тощо. Все це координувалося на постійний центральний контролер зовнішньої обробки, що має сенс для застосування здоров'я але не підходить для домашніх застосувань.

Castro et al. [56], більше дивився на застосування CoAP через WAN та ітерацію з веб-браузерами, демонструючи масштабованість CoAP та його придатність для розлогих додатків.

CoAP пропонує метод, за допомогою якого UDP може бути достатньо надійним для більшості додатків. Discovery має два компоненти під CoAP: спочатку повинен бути виявлений сервер, а потім його ресурси. Для наших цілей відкриття серверів представляє інтерес. Це досягається за допомогою багатокислого пакету, на який відповідають видимі пристрої. Однак ці повідомлення вважаються "непідтвердженими", тобто пакет підтвердження не потрібен. Якщо є втрата пакета, то повідомлення про виявлення CoAP можуть бути втрачені або не охопити всі пристрої.

Після того, як пристрій підключений до мережі і відомий в системі, тобто пристрій був підключений і виявлений, за ним необхідно стежити. З цією метою залишається розглянути, як стан пристрою може бути відомий або як пристрій може контролюватися. Як і discovery, цей розділ буде вважати, що реалізація з використанням Wi-Fi матиме більше проблем і може бути легко адаптована до дротового дизайну Ethernet. Тому, якщо Wi-Fi є цільовим транспортним середовищем, вкрай важливо, розглядаючи відповідальність моніторингу, що частота, з якою здійснюються передачі, мінімізується як простір Wi-Fi, особливо діапазон 2,4 ГГц [19], дуже оспорюється і перевантажена. Крім того, в домашніх

умовах HAS навряд чи є єдиним домінуючим або навіть найважливішим користувачем простору Wi-Fi.

Домашня автоматизація вже давно є сферою пропрієтарних протоколів, але останнім часом відкриті стандарти набирають значних обертів. Двома основними гравцями, які слід розглянути для моніторингу, є MQTT [59], вперше стандартизований у 2014 році, та CoAP з проектом RFC, опублікованим у 2014 році [53].

MQTT - це легкий протокол для зв'язку між машинами (M2M) в Інтернеті. Він був реалізований в різних системах, орієнтованих на домашню автоматизацію. Багато в чому через обмежені обчислювальні вимоги, кілька додатків показали, що він може бути реалізований на недорогому обладнанні, такому як Raspberry Pi [8], [9], плати Arduino [8], [34] та ESP8266 [14], [16], [17].

Хоча всі ці системи залежать від всюди сущої домашньої бездротової мережі, в деяких дослідженнях розглядалося питання про використання інших протоколів фізичного рівня, таких як LoRa, для перенесення MQTT.

Хоча системи домашньої автоматизації на базі MQTT пропонують недороге обладнання, вони не є частиною більш широкого протоколу, який розглядає приєднання пристроїв та виявлення, і вони пов'язані з постійним центральним контролером. Jutadhamakorn et al. [9] продемонстрували, що цей підхід можна зробити досить добре масштабованим, але це досягається лише шляхом подальшого прийняття складності постійного центрального контролера до нехтування простим користувацьким досвідом. Ця залежність є не просто вибором дизайну, оскільки MQTT в основному базується на постійному центральному контролері у вигляді того, що він називає «брокерами».

Брокер є джерелом інформації для абонента. Брокеру надається інформація для публікації. Видавці надсилають дані брокера, пов'язані з темами, а Абоненти реєструють свій інтерес до теми, підписавшись на Брокера. Хоча це хороший підхід для скорочення мережевого трафіку і все ще підтримання зацікавлених сторін в актуальному стані, він повністю і принципово залежить від топології постійного центрального контролера.

Брокери в MQTT пропонують ряд переваг. Оскільки брокер функціонує так само, як буфер, абоненти можуть отримати доступ до останньої інформації від видавця без необхідності залишатися в Інтернеті. Це дозволяє пристрою Publisher переходити в режим сну, коли це не потрібно, і прокидатися лише за потреби програми, для якої він використовується. Крім того, видавцю не потрібно приймати з'єднання з декількох пристроїв; всі його повідомлення можуть перейти до брокера. Тим не менш, це відбувається за рахунок додаткового пристрою, щоб функціонувати як брокер, який знову виступає в якості PCC. У багатьох випадках використання MQTT брокера може бути не таким ефективним пакетом, як було б ідеально, оскільки один пакет надсилається кожному Абоненту, а також початковий пакет брокеру. Якщо все це надсилається через Wi-Fi, це може мати значний вплив на масштаб.

CoAP - це більш пізній протокол, який націлений на пристрої низької потужності з обмеженою обчислювальною потужністю. Хоча менше роботи розглядало застосування CoAP до систем домашньої автоматизації, його фундаментальна легка філософія корисна. CoAP використовувався як комунікаційний шар для досягнення сумісності між різними, часто застарілими протоколами [60], [61]. Інші системи зосереджені на безпеці пристроїв або використанні CoAP для Енергетичного менеджменту в будинку. Хоча всі ці системи використовують Wi-Fi і іноді Ethernet, Son та ін. продемонстрували, що CoAP працює над іншими протоколами, що не базуються на IP- пристроях.

Знову ж таки, всі ці системи в основному покладаються на постійного центрального контролера. Хоча CoAP може працювати за обмежених обставин без постійного центрального контролера, він призначений для роботи в широкій системі з набагато більшою інфраструктурою, ніж середній будинок. Таким чином, деякі з його більш ефективних функцій втрачаються, такі як використання посередників, які обмінюються кешованими даними з кількома зацікавленими клієнтами [62].

Як і MQTT, CoAP також має модель підписки, де клієнти можуть зареєструвати інтерес до певного статусу або «ресурсу». Цей процес відомий як

«спостереження» під CoAP і допомагає зменшити пакети, реєструючи інтереси клієнта та підтримуючи їх в актуальному стані, не запитуючи [62].

Було проведено деяке порівняння між MQTT і CoAP в домашніх умовах, виявивши, що CoAP є більш ефективним пакетом, але менш енергоефективним, ніж MQTT [63]. Однак головною проблемою цього дослідження залишиться ефективність пакетів і парадигма без РСС.

UPnP є набагато старішим стандартом [45], який був стандартизований у 2008 році і пропонує парадигму, яка не повністю залежить від РСС. UPnP використовує "Eventing" для публікації оновлень стану передплатників, коли змінюється змінна стану. Ці сповіщення про події можна надсилати як одно- або широкоформатні. Цей підхід не підходить для змінних, які можуть часто змінюватися, оскільки це буде генерувати більший мережевий трафік. UPnP також залишає користувачеві робити власні припущення щодо того, чи не означає відсутність оновлень, що Видавець зараз перебуває в автономному режимі або просто не змінився. Таким чином, UPnP Eventing не підходить для домашнього середовища, тому відповідно до UPnP, змінні повинні бути опитані вручну за допомогою команд запит.

Після того, як пристрій з'єднаний, виявлений і може контролюватися в мережі, останній крок до створення повністю автоматизованої системи полягає в тому, щоб дозволити взаємодії користувача з системою та її пристроями.

Хоча цей розділ зосереджений на управлінні пристроями кінцевим користувачем, необхідно, щоб пристрої могли контролювати та взаємодіяти один з одним, і це те, що передбачено в інших обов'язках конфігурації та зв'язку між пристроями.

Нарешті, можливість адекватного відображення і управління пристроєм вимагає, щоб пристрій і його можливості були адекватно зрозумілі, тобто забезпечити дисплей і управління пристроєм повинно бути адекватно описано якимось чином. Опис можна вважати ще однією ключовою відповідальністю Постійного центрального контролера, однак це передбачається можливістю відображення та керування, і тому зв'язок усіх трьох є ключовою функцією, що розглядається тут.

Відображення та керування пристроями IoT у системах домашньої автоматизації можна класифікувати за трьома основними типами: розміщений пристрій, розміщений зовні та користувацькі програми.

1.4 Постановка задачі

Для досягнення поставленої мети потрібно розв’язати такі основні задачі:

– проаналізувати відомі методи розробки протоколів для центрального контролера в кіберфізичній системі «розумний будинок»;

– удосконалити протокол для центрального контролера в кіберфізичній системі «розумний будинок»;

– удосконалити метод проєктування протоколу для центрального контролера в кіберфізичній системі «розумний будинок»;

– реалізувати протокол для центрального контролера в кіберфізичній системі «розумний будинок» згідно розроблених рішень.

1.5 Висновки

Приєднання не є областю великої уваги в літературі, але багато систем мають на увазі рішення, які були вивчені. Як було видно, пропонуються чотири основні підходи: виділений IO, Bridge, PCC і новий мінімалістичний підхід, запропонований Насріном і Редкліффом.

Виділений вводу-виводу займає один з двох шляхів, він або додає обладнання до пристрою, яке необхідне лише для процесу об'єднання та збільшує вартість, або використовує будь-яке обладнання, яке вже доступне, і робить все можливе, щоб полегшити приєднання, часто за рахунок користувацького досвіду.

Об'єднання трохи схоже на виділений io, оскільки вимагає придбання додаткового обладнання. На цей раз додане обладнання перетворюється між різними протоколами, що дозволяють з'єднання між Wi-Fi і ZigBee, наприклад.

Хоча це призвело до деякого комерційного успіху, це додаткова вартість, яка не пропонує ніякої додаткової цінності для клієнта, якщо міст не включений в більший PCC і який вже має значні недоліки.

PCC підходить до всіх, але відмовляється від зручності бездротових рішень і блокує користувачів в продуктовій лінійці виробника, що призводить до високих витрат і обмежень функцій. Модернізація та технічне обслуговування вимагають спеціалістів-техніків і лише збільшують витрати на рішення.

Мінімалістичні рішення Насріна і Редкліффа додають новий підхід до приєднання, який дозволив би приєднатися без PCC і без додаткових витрат. Він просто використовує апаратне забезпечення Wi-Fi, вже притаманне розумному пристрою Wi-Fi. Однак Насрін і Редкліфф не запропонували ніякої реалізації або доказів концепції, просто надавши теоретичний протокол. Крім того, запропонована парадигма користувача вимагатиме від користувачів високого рівня складної конфігурації. Якщо цей протокол може бути реалізований і розширений, це може виявитися життєздатною альтернативою для приєднання, яка є безпечною і не несе додаткових витрат.

Кілька протоколів були дуже успішними в моніторингу пристроїв для мереж IoT, і хоча MQTT, CoAP і UPnP відрізняються, кожен з них пропонує переваги для моніторингу комунікацій, включаючи низькі обчислювальні вимоги та ефективність пакетів. MQTT і CoAP сильно залежать від PCC за їх функції моніторингу, і хоча UPnP може функціонувати без PCC, його підхід до моніторингу не дуже підходить для часто мінливих даних. Жоден з цих протоколів не пропонує комплексне рішення для моніторингу в системі домашньої автоматизації (HAS), але вони пропонують цінні уроки. Необхідно буде розробити новий протокол Моніторингу, але він повинен засвоїти цінні уроки з цих протоколів. Було визначено три основні підходи до відображення та керування: розміщений пристрій, розміщення на зовнішніх та користувацьких програмах. У кожного з них було кілька проблем, що роблять їх непридатними для відображення та управління без PCC, деякі все ще залежать від PCC, інші не інтегрувалися добре в більш

широкий HAS, а деякі залишили користувача залежним від підключення до Інтернету, щоб використовувати свої пристрої.

Було відзначено, що було докладено мало зусиль для об'єднання Display з описом і управлінням пристроєм, але ті, хто доклав зусиль у цьому напрямку, підкреслили нереалізований потенціал XML для опису макетів інтерфейсу користувача, можливостей пристрою або способів управління пристроєм. Однак ніхто не зібрав все це разом, і це варто було б дослідити в процесі надання відповідного протоколу відображення та контроль.

2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ СИСТЕМИ ТА ПРОТОКОЛУ ВЗАЄМОДІЇ КОМПОНЕНТ

2.1 Визначення області дослідження

Першим обов'язком PCC є приєднання, тобто, як новий пристрій може бути підключений до домашньої мережі і стати доступним для HAS. Цей розділ зосереджується на цій відповідальності та розглядає перше дослідницьке питання: безпечне приєднання без постійного центрального контролера були показані для старих версій Android. Які зміни потрібні, щоб розширити це ширше?

Існує кілька підходів для приєднання до бездротових пристроїв, але всі вони мають значні проблеми і сильно залежать від PCC. Винятком став мінімалістичний протокол Насріна і Редкліффа, який був запропонований в теорії, але ніколи не був реалізований або оцінений.

Цей розділ починається з розгляду того, чи може протокол Насріна і Редкліффа запропонувати життєздатний метод приєднання до IoT-пристроїв без PCC. Це робиться, придивляючись уважніше до протоколу Насріна і Редкліффа і роблячи перші кроки до реалізації. Звідти досліджується і розглядається кілька поліпшень.

Реалізувавши протокол з декількома удосконаленнями, можна оцінити життєздатність протоколу. Ця оцінка визначає деякі значні обмеження, які в кінцевому підсумку роблять протокол нежиттєздатним для широкого прийняття на ринку.

Це дослідження спочатку було опубліковано автором в декількох роботах, і вони в значній мірі відтворені в змісті цього розділу. Перша стаття «Розширена реалізація нового протоколу приєднання до IoT. Він визначив протокол Насріна і Редкліффа як перспективний протокол приєднання, щоб замінити PCC, і продемонстрував, з удосконаленнями, що протокол може бути реалізований і запропонований зручний досвід.

Друга стаття ,"Оцінка та вдосконалення нового протоколу приєднання до IoT. Ця стаття знову визначила протокол Насріна і Редкліффа як перспективний

протокол приєднання і детально описала поліпшення, які були зроблені для парадигми користувача при реалізації протоколу. Крім того, він підняв розгляд сумісності протоколу, особливо зі смартфонами, і порівняв його з часткою ринку Android та iOS. Він прийшов до висновку, що потрібен більш універсально сумісний протокол.

Нарешті, «Універсальний протокол приєднання до IoT для diy-додатків» [35] був опублікований на Міжнародній конференції телекомунікаційних мереж та додатків (ITNAC) у 2017 році. Він також оцінив сумісність частки ринку в розширеному впровадженні Протоколу приєднання Насріна і Редкліффа і знову прийшов до висновку, що потрібен новий протокол і що переорієнтація Протоколу приєднання може досягти універсальної сумісності з будь-якими пристроями переривчастого управління з підтримкою Wi-Fi.

2.2 Протокол Насріна і Редкліффа

Протокол Насріна і Редкліффа призначений для підключення IoT -пристроїв до бездротових мереж, зберігаючи при цьому високий рівень безпеки. Він призначений для зниження витрат, використовуючи тільки апаратне забезпечення, вже необхідне для бездротового пристрою IoT.

Основна ідея та вимога до цього протоколу полягає в тому, що кожен бездротовий пристрій буде попередньо налаштований за допомогою унікального ідентифікатора набору послуг (SSID) та пароля виробником. З пристроєм можна зв'язатися лише в мережі, яка відповідає цим унікальним і секретним параметрам. Це найбільш легко досягається смартфоном, що діє в режимі гарячих точок.

Нарешті, третій етап показує, що пристрій IoT тепер може підключатися до локальної мережі Wi-Fi з обліковими даними, переданими через попереднє з'єднання на другому етапі. За допомогою смартфона знову в звичайному режимі Wi-Fi він може підключатися до локальної мережі Wi-Fi і через локальний маршрутизатор спілкуватися з пристроєм IoT і керувати ним.

Все це досягається без необхідності додаткового обладнання або складності в пристрої IoT. Замість цього він використовує тепер всюдисущий смартфон і його бездротові можливості AP.

Робота Насріна і Редкліффа продемонструвала, що коли смартфон і пристрій IoT підключаються через точку доступу смартфона, вони можуть передавати дані один одному за допомогою UDP. Насрін і Редкліфф також передбачили більш повну реалізацію запропонованого протоколу приєднання, додавши дві функції: перша для автоматизації завдання введення смартфона в режим AP і вихід з нього, а друга - автоматизація переналаштування ioT і облікових даних Wi-Fi для смартфонів для приєднання до локальної мережі Wi-Fi. Ці функції повинні бути реалізовані, щоб гарантувати, що протокол практично життєздатний.

Оцінюючи здатність сучасного обладнання підтримувати протокол Насріна і Редкліффа, необхідно було досягти чотирьох ключових функцій: по-перше, потрібно було змусити смартфон увійти в режим AP і вийти з нього. По-друге, конфігурація Wi-Fi смартфона повинна бути програмно налаштована, як в режимі Wi-Fi, так і в AP. По-третє, зв'язок між пристроєм IoT і смартфоном повинен бути досяжним. Четверте, облікові дані Wi-Fi пристрою IoT повинні бути програмно налаштовані.

Ці ключові функції були успішно зіставлені з операційною системою Android, в той час як пристрій IoT був представлений Raspberry Pi під управлінням Linux.

Наступна оцінка була зроблена для перевірки сумісності протоколу Насріна і Редкліффа з поточним обладнанням і виявлення інструментів, необхідних для реалізації протоколу:

Програмна реалізація Wi-Fi для смартфонів можлива за допомогою setWifiEnabled, логічного члена класу API android.net.wifi.WifiManager, який може вмикати або вимикати Wi-Fi. Хоча статус послуги Wi-Fi можна визначити за допомогою isWifiEnabled() або для більш детальної інформації, getWiFistate(). Метод getSystemService поверне клас WifiManager, щоб увімкнути вищезазначені функції.

Програмна конфігурація Wi-Fi може бути досягнута за допомогою функції-члена addNetwork() класу WifiManager і налаштування класу WifiConfiguration. Редагуючи клас WifiConfiguration і заповнюючи рядки SSID і PreshareKey, Wi-Fi можна налаштувати на потрібні налаштування. Для досягнення конфігурації в режимі AP необхідно використовувати Android 4.0 (API 14) або новішої версії, оскільки попередні версії не підтримують конфігурацію [64].

Зв'язок раніше був розроблений Насріном і Редкліффом за допомогою посилання UDP для доставки пакетів між пристроями. Ця комунікаційна ланка добре зрозуміла і інші можливі підходи рясніють, однак існуючої роботи достатньо.

Програмна конфігурація IoT Wi-Fi під Linux може бути досягнута шляхом редагування файлу конфігурації, wpa_supplicant.conf а потім перезапуску служби.

Всі ці функції були успішно реалізовані і інтегровані в функціональний додаток android, використовуючи інтерфейс, запропонований Насріном і Редкліффом, Raspberry Pi використовувався для представлення пристрою IoT, в даному випадку простого світла, яке можна повернути.

Ця оцінка для картографування протоколу Насріна і Редкліффа продемонструвала здатність ОС Android підтримувати протокол. Однак необхідна подальша оцінка для забезпечення того, щоб інші ОС були здатні підтримати протокол.

На даний момент обмеження запропонованого інтерфейсу Насріна стають очевидними. Він має ефективний інтерфейс, але досить вимогливий, що вимагає від користувача бути добре знайомим з конфігурацією своєї локальної мережі Wi-Fi. Інтерфейс також вимагав введення випадкових SSID і паролів, що збільшило ймовірність помилки користувача і таким чином створює поганий користувальницький досвід. Інтерфейсу також не вистачало будь-якої форми вказівки на прогрес, щоб запевнити користувача, що додаток працює над їхніми запитами, а не просто чекає подальших підказок або завис.

Спочатку був розроблений автоматизований метод введення попередньо налаштованих облікових даних ioT-пристрою з використанням QR-кодів. Замість

того, щоб виробник надав унікальний письмовий SSID і пароль, вони будуть надавати унікальний QR-код, який користувач буде сканувати, таким чином заощаджуючи на введенні тексту, помилках і часі.

По-друге, користувачі більше звикли приєднуватися до мереж Wi-Fi, вибираючи потрібний SSID зі списку.

Зараз користувачам потрібно було лише мати можливість ідентифікувати свій SSID (не вводити його) і вводити пароль. Це забезпечило звичний для користувача процес і вимагало лише мінімального знайомства з домашнім Wi-Fi.

По-третє, завантаження екранів - це простий, але ефективний спосіб заспокоїти користувачів, що додаток працює. Вони можуть бути використані для інформування про час очікування і якщо щось налаштовано неправильно. Якщо очікування користувача встановлюються рано, більш тривалий час очікування менше впливає на досвід користувача, тоді як несподівана бездіяльність швидко залишає користувачів невизначеними та стурбованими, швидко руйнуючи користувальницький досвід.

Доведення того, що мережа Насріна і Редкліффа працювала на практиці і внесла значні вдосконалення, що допомогло зробити протокол більш зручним для користувачів, але чи можна зробити користувальницький досвід ще простішим? Хоча вартість виключила додаткові апаратні рішення, такі як додавання NFC, ці реалізації можуть призвести до дуже простого інтерфейсу: користувач просто вибирає власну локальну мережу, вводить пароль і натискає кнопку «приєднатися».

Для того, щоб зберегти мінімалістичний підхід протоколу Насріна і Редкліффа, можна використовувати апаратне забезпечення, вже притаманне пристроям IoT, щоб спростити досвід користувача?

Можливо, можна використовувати ці апаратні компоненти, щоб спростити процес приєднання настільки, наскільки досяг NFC. Використовуючи протокол Насріна і Редкліффа, ці світлодіоди або зумери могли безпечно відправляти унікальний SSID і пароль IoT на смартфон, тим самим усуваючи стадію QR-коду. Це може зменшити витрати і спростити користувальницький досвід.

Замість того, щоб сканувати QR-код, користувачі запустять пристрій IoT, який почне блимати (для світлодіода) або відтворювати (для зумера) сигнал, закодований з локальним SSID і паролем. Потім користувачі можуть захопити цей сигнал з програми для смартфона, будь то наведення камери на світлодіод або утримання мікрофона до зумера. Це усуває потребу в QR-кодах, усуваючи витрати, заощаджуючи гроші виробників пристроїв. Крім того, користувачам не потрібно буде відстежувати кожен QR-код і пристрій, до якого належить кожен.

Такий підхід додає додатковий рівень безпеки, оскільки пристрій, що з'єднується, повинен бути присутнім не тільки в межах досяжності Wi-Fi, але і в межах досяжності зумера або світлодіода. Таким чином, ioT-пристрій зможе переконатися, що він спілкується з користувачем в досить близькому районі, щоб, ймовірно, бути власником пристрою.

У оригінальному протоколу Насріна і Редкліффа, якщо шкідливий користувач випадково знав початковий SSID IoT і пароль IoT і знаходився в діапазоні Wi-Fi IoT на момент живлення пристрою до приєднання, вони могли б спочатку приєднатися до пристрою або захопити домашній SSID і пароль. Однак з використанням допоміжного обладнання SSID і пароль можна отримати тільки в набагато ближчій близькості. Щоб зловмисники не вгадували облікові дані Wi-Fi або прогнозували їх на основі моделей виробників, пристрій IoT може додатково перевірити пристрій, випадково згенерувавши пароль, або після підключення до точки доступу він може запросити другий випадковий пароль, який буде закодований на світлодіоді або зумері. Процес, за допомогою якого ці паролі генеруються, не повинен бути надмірно складним, щоб вимагати значних накладних витрат, але потрібно лише уникати базової передбачуваності, уникаючи шаблонів, пов'язаних з виробництвом. Основна мета і цінність полягає в тому, що це не статичний пароль.

Ці опції, якщо це можливо, мають потенціал не тільки для спрощення користувацького досвіду, але і для зниження вартості IoT-пристроїв, а також підвищення безпеки заходу приєднання шляхом перевірки близькості користувача до пристрою IoT.

Реалізація світлодіода вимагатиме від користувача тримати камеру смартфона над миготливим світлодіодом, щоб отримати унікальний SSID та пароль для пристрою IoT.

Пристрій IoT може модулювати світлодіод на високій швидкості, але смартфон може контролювати такий світлодіод тільки за допомогою відеозйомки з частотою кадрів від 12 Гц до 60 Гц і більше, залежно від смартфона. Використовуючи критерії вибірки Nyquist, це означає, що швидкість передачі даних, в кращому випадку, буде від 6 Гц до 30 Гц. Хоча SSID може бути досить коротким, можливо, тільки дві літери, пароль повинен бути набагато довше. Для WPA2 було запропоновано, щоб пароль був як мінімум 12 випадкових символів, що дорівнює 78 бітам ентропії. Таким чином, потрібно мінімум 14 символів, 2 для SSID і 12 для пароля. З 7 бітами для кожного символу потрібно в цілому 98 біт. При частоті дискретизації 6 Гц користувачеві, таким чином, доведеться утримувати камеру над світлодіодним IoT протягом приблизно 17 секунд, 4 секунд для високошвидкісного смартфона. Ці часи засновані на розрахунках для базово-смугового сигналу. Ці часи можуть бути скорочені за рахунок більш складної модуляції та кодування, але це буде відбуватися за рахунок збільшення обчислювальних накладних витрат, які можуть бути нежиттєздатними для обмежених простих пристроїв.

Було б досить складним завданням розробити інтерфейс користувача, який допоміг би користувачеві впоратися з цими тривалістю часу. Сканування QR-коду, здавалося б, набагато простіше рішення для користувача.Реалізувавши протокол Nasrin і Radcliffe, ми в змозі точно зрозуміти його вимоги і розглянути сумісність протоколу з іншими операційними системами.

У попередніх версіях Android API були доступні для надання доступу до управління апаратним забезпеченням Wi-Fi, і, таким чином, маніфест програми повинен заявити, що цій програмі буде надано доступ до цих налаштувань. Чи можуть інші мобільні операційні системи надавати такий доступ або такий доступ блокується? Примітно, що iOS від Apple не дозволяє розробникам програмно налаштовувати режим точки доступу, навіть не вмикати або вимикати його. Деякий

доступ доступний для підключення та конфігурації Wi-Fi, але він дуже обмежений і жорстко контролюється. Хоча це було підтверджено ретельною оцінкою документації розробника, це також широко підтверджується офіційними та неофіційними форумами розробників.

Хоча Android займає велику частку світового ринку, на рівні 86,8% в 2017 році (останні дані, доступні при проведенні цієї роботи), не обходиться без конкурентів [65]. Незважаючи на набагато меншу частку Apple на світовому ринку, вона аж ніяк не є незначним конкурентом. Це з двох основних причин, Apple є набагато більш значущим гравцем на більш прибуткових ринках, і користувачі Apple, здається, мають і витрачають більше грошей.

Перша частка ринку Apple в Австралії (станом на 2017 рік) становить 39,1%, сильна конкуренція з Android становить 59,8%. Тенденції схожі у Великобританії, де Apple займає 40,4% ринку, а Android - 57,2%, а також в США, де Apple володіє лише 38,9%, а Android - 59,2%.

Тільки цей факт вказує на те, що пристрої Apple iOS не можуть бути проігноровані будь-яким HAS з прагненнями проникнення на ринок.

Пристрої Apple є загальними і не повинні ігноруватися як пристрій управління для HAS. Однак робота з впровадження протоколу Насріна і Редкліффа з удосконаленнями показала, що залежність протоколу від програмного доступу до апаратного забезпечення Wi-Fi робить його несумісним з цими пристроями iOS. Це ставить під загрозу мету цього нововведення створити відкритий протокол. HASS повинен бути відкритим не тільки для самих різних пристроїв різних виробників, але і для різних пристроїв управління.

Без можливості програмного інтерфейсу з апаратним забезпеченням Wi-Fi під iOS впевнений користувальницький досвід неможливий. Однак реалізувати протокол Насріна і Редкліффа з ручною конфігурацією Wi-Fi навіть не вдасться. Це пов'язано з тим, що користувачі не можуть налаштувати SSID точки доступу, яка створюється на основі імені пристроїв.

Крім того, не всі пристрої Android оснащені можливостями точки доступу, або вони були відключені. Тому в цілому нерозумно вважати, що будь-хто, хто хоче

налаштувати HAS, матиме доступ до сумісного обладнання. Таким чином, здавалося б, що, хоча протокол Насріна і Редкліффа може бути практично реалізований і з деякими вдосконаленнями, він може бути зручним для користувачів, він в кінцевому підсумку не в змозі вирішити доступність обладнання для середнього користувача.

Для вирішення цих проблем і досягнення загальної сумісності пропонується новий протокол. Це дозволить скористатися останніми розробками в області технологій, які зробили апаратне забезпечення, таке як ESP8266. Вони коштують всього кілька доларів, але здатні підтримувати режими Wi-Fi і точки доступу. Цей новий протокол покладає відповідальність за точку доступу на пристрій IoT, а не на пристрій управління (тобто смартфон). Таким чином, будь-який пристрій, здатний до зв'язку Wi-Fi, буде потенційним контролером для сумісних IoT-пристроїв. Модулі ESP8266 є лише одним із прикладів чіпів, які можуть бути використані, однак будь-який чіп Wi-Fi з налаштовуваним режимом точки доступу буде сумісний з цим протоколом. Керуючому пристрою знадобиться секретний пароль для пристрою IoT. Після підключення пристрій управління зможе безпечно транслювати локальні облікові дані Wi-Fi на пристрій IoT, всі захищені шифруванням WPA2, а не відкритою мережею. Після отримання цих облікових даних IoT і пристрій управління можуть приєднатися до локальної мережі Wi-Fi.

Це розглядає лише деталі послідовності об'єднання як такі, що мають відношення до пристрою IoT, він не деталізує, наприклад, як телефон повинен отримати локальні облікові дані Wi-Fi або SSID та пароль для пристрою IoT. Вони можуть бути досягнуті, як це було зроблено раніше при виконанні протоколу Насріна і Редкліффа.

Пристрій управління потрібен тільки для підтримки зв'язку UDP, що відкриває протокол для сумісності практично з будь-яким пристроєм з підтримкою Wi-Fi: телефоном, ноутбуком, настільним комп'ютером, навіть смарт-годинником, всім незалежним від виробника і операційної системи, навіть пристроями iOS. Це такий відкритий протокол, необхідний для успіху HAS.

Переорієнтований протокол залежить від дуже недорогих пристроїв IoT, здатних працювати в режимі AP (точки доступу) в домашній мережі з шифруванням WPA2. Нові пристрої ESP за ціною ~ $ 3 (USD), як видається, відповідають цій вимозі. Попередня оцінка пристрою ESP8266 дала багатообіцяючі результати. Стрес-тести показують, що пристрій, як видається, надійний пристрій, який здатний до значних навантажень трафіку. Він працює належним чином як пристрій Wi-Fi або точка доступу і є WPA2.

Після налаштування в режимі точки доступу пристрій був здатний відправляти і отримувати пакети UDP. Це дозволило всі необхідні комунікації, необхідні для реалізації цього нового протоколу.

Що стосується безпеки, то варто відзначити, що цей протокол сам по собі не забезпечує власну конфіденційність. Замість цього, це залишилося на рівні зв'язку даних із захистом WPA2, що означає, що він такий же безпечний, як і домашній Wi-Fi. Крім того, він стане більш безпечним понаднормово тільки в міру того, як нові технології безпеки стають доступними, такі як WPA3.

2.3 Метод розробки протоколів

Це дослідження показало, що існуючі протоколи виявлення недостатні, коли немає PCC, і тому був розроблений новий і надійний протокол. Тим не менш, це значна проблема для розробки нового надійного протоколу, який є надійним, оскільки легко зробити помилки або вставити помилки в дизайн. Для подолання цієї проблеми потрібна методика для зменшення кількості і величини недоліків. При розробці цього протоколу було встановлено, що жодна формальна методологія розробки не підходить для мінімізації недоліків дизайну для проектування протоколу. Хоча існує багато методологій розробки проектів, жодна з них не відповідає нішевій області розробки протоколів, і тому потрібна нова методологія.

Цей новий метод дозволить отримати більшу впевненість у розробці, впровадженні, тестуванні та використанні нового протоколу Discovery.

У цьому розділі спочатку буде розглянуто необхідність методології розробки мережевого протоколу, яка бере уроки з існуючих методів розробки, а потім запропонує надійну методологію розвитку мережі. Потім окреслюються вимоги до нового підходу до відкриття, і до цих вимог застосовується методологія надійного розвитку мережі, щоб запропонувати протокол Discovery, який потім буде реалізований та оцінений.

У розділі представлена нова методологія програмування і застосовується вона до внутрішнього середовища Wi-Fi. Процес, за допомогою якого концепція розробляється в мережевому протоколу є критичним, оскільки помилки тут можуть бути не виявлені до практичного використання і вимагають переробки може бути дорогим процесом. У цьому розділі розглядаються існуючі методології розробки та висвітлюються ключові сильні та слабкі сторони, які можуть бути використані для розробки методу, адаптованого до розробки мережевих протоколів.

Протягом багатьох років відомо, що помилки, допущені в процесі розробки, можуть не бути помічені до тих пір, поки протокол не буде реалізований, або навіть пізніше, коли він використовується на практиці. Такі помилки коштують дорого як з точки зору часу, так і грошей. Мінімізація помилок у всіх областях дизайну має вирішальне значення, і багато авторів запропонували формальні методології для досягнення цього на етапі розробки. Ми можемо скористатися багаторічним досвідом в області дослідження життєвого циклу проекту і засвоїти уроки з різних методологій, а потім адаптувати ці уроки до нішевої області розробки мережевих протоколів. Література в області життєвих циклів величезна, але мало присвячена специфіці розробки мережевих протоколів, які пропонують методологію, засновану в основному на моделі водоспаду.

Методологія, яка може перейняти досвід процесів життєвого циклу та створити індивідуальне рішення для розробки мережевих протоколів, буде набагато кращою, ніж спеціальний підхід, який, ймовірно, створить більше помилок, затримок розробки та перевитрат витрат.

Методологія розробки мережевого протоколу повинна починатися з чіткої мети щодо базової функціональності протоколу. Кінцевим виходом повинна бути

надійна конструкція протоколу, яка може бути передана відповідальним за його реалізацію. Для надійності повинен бути розроблений протокол, щоб вести себе надійно і належним чином за будь-яких умов. У мережевих протоколах мало допуску до помилок, тому тести та оцінки випадків використання повинні бути велика. Це говорить про те, що якісна документація має важливе значення не тільки для успішного впровадження, а також для того, щоб прищепити довіру до цих систем тестування та введення в експлуатацію та об'єднання протоколу по всьому світу. Розробка мережевих протоколів дещо унікальна, що на етапі розробки немає традиційного інтерфейсу клієнта, і, таким чином, вимоги керуються початковою концепцією і кінцевою метою.

З урахуванням ключових характеристик життєвого циклу мережевого протоколу тепер можна подивитися на існуючі методології, щоб побачити, які уроки можна засвоїти.

Життєвий цикл водоспаду [67] надає пріоритет плануванню на ранній стадії і сприяє суворій документації. Вони є важливими елементами в цілому і конкретно для розробки мережевих протоколів.

Життєвий цикл сприяє формалізованим оглядам з потенціалом для керівних принципів і контрольних списків для управління помилками, а також полегшує дозрівання менш досвідчених розробників і зменшує помилки, допущені менш досвідченим персоналом. Ключова проблема полягає в тому, що специфікації не тестуються до використання клієнтом, в цей час зміни дуже дорогі.

Життєвий цикл прототипу [68] пропонує корисний акцент на розстановці пріоритетів, починаючи з ключової функціональності в першу чергу і нарощуючи дизайн. Крім того, його ітераційний підхід дозволяє поліпшити ідею, а також повторне вивчення, що сприяє надійному дизайну.

Спіральна модель [69] використовує досвідченого інженера, щоб зробити сувору оцінку ризику, коли команда складається з членів з різноманітним досвідом. Це дає потенціал для розвитку досвідчених інженерів, оскільки менш досвідчені вчаться у більш досвідчених. Крім того, як і процес прототипу, спіральна модель

виграє від вдосконалення ітерації і додає додаткову можливість виявити помилки, особливо ті, що пов'язані з переглядами в різних аспектах дизайну.

V-модель [70], хоча і дуже схожа на модель Waterfall, пропонує цінний внесок у висвітлення взаємозв'язку між тим, де допускаються помилки і де вони будуть виявлені, і відповідним ступенем понесених витрат.

Модель Extreme (або XP) [71] розробляє тестові випадки до програмування, і це пропонує унікальну та ефективну стратегію мінімізації помилок та економічної ефективності.

Роботу Кеніга [66] слід похвалити за рідкісний розгляд життєвих циклів у розробці мережевого протоколу. Запропонована методологія добре продумана і багато в чому зобов'язана моделі водоспаду. Однак пізній розгляд обмежень у життєвому циклі Кеніга неминуче вимагатиме переробки конструкції, що збільшує витрати та часу.

Багато життєвих циклів проектування пропонують велику допомогу в розробці мережевих протоколів. Однак жоден з них не пропонує надійного способу переходу від початкової ідеї / концепції до фази перевірки. Це має вирішальне значення для надійного протоколу і таким чином, була розроблена наступна методологія, яка вільно базується на екстремальній моделі та життєвому циклі прототипу.

Ця методологія, розроблена для цього дослідження і в результаті цього дослідження, приймає ключові аспекти розробки мережевих протоколів, а також уроки з існуючих методологій, щоб розробити метод спеціально для проектування мережевих протоколів, який називається RNDM.

Крок 1. Наївний протокол: початкова концепція мережевого протоколу повинна бути узагальнена в наївному протоколі. Це повинно представляти мінімальний функціональний результат, бажаний для нового протоколу, і він повинен бути за своєю суттю оптимістичним щодо навколишнього середовища та всіх інших процесів. Очікується, що він не буде життєздатним рішенням і повинен уникати прогнозування будь-яких проблем в його роботі. Це запобігає веденню

аналізу в певному напрямку, який може звузити рішення, розглянуті на більш пізніх стадіях.

Крок 2. Модель: виберіть метод моделювання та інструмент. Існують різні способи представлення мережевих протоколів. Загальні методи включають діаграми відмов, FSM, SDL та мережі Петрі. FSM і Петрі мережі мають перевагу, що їх можна використовувати для перевірки того, що протокол працює в міру необхідності. Відмов діаграми є адекватними, коли протокол простий, наприклад, з відносинами господар-раб, де коливання часу не є значними. Будь-який з цих методів може бути використаний з RNDM в якості протоколу, який розробляється.

Крок 3. Оцінка: наївний протокол ретельно оцінюється методом, подібним до експериментів Геданкена[72]. Це етап генерації сценаріїв, на якому визначаються всі проблемні випадки використання. Одним із способів досягнення цього було б розглянути контрольний список відомих загальних проблемних сценаріїв або категорій. Це має велику перевагу для підтримки рівномірної та комплексної оцінки з менш досвідченим користувачем.

Однак, ця оцінка буде тільки так само добре, як контрольний список, і унікальні аспекти протоколу і його навколишнього середовища можуть бути проігноровані. Для того, щоб боротися з цим, можна також застосувати більш творчий підхід до мозкового штурму, відповідаючи на просте запитання: "Що може піти не так в будь-якому місці цього протоколу?" Цей спосіб сприяє досвідченому розробнику, в той час як перший дуже допомагає новачкові.

Комбінація контрольного списку та мозкового штурму може досягти найкращого з обох світів, але мозковий штурм слід застосувати в першу чергу, оскільки контрольний список може придушити творчу думку і здаватися всеосяжним. Після мозкового штурму, а потім чек-лист були застосовані, оцінка двох повинна дозволити доповнення до контрольного списку для майбутньої заявки. Таким чином, чек-лист зміцнюється з часом і досвід роботи в команді збільшується.

Контрольний список повинен застосовуватися строго. Це найкраще зробити, застосовуючи кожен елемент до кожної події або процесу в протоколі. Це найкраще

продемонструвати, оцінивши схему відмов: запуск контрольного списку для кожного вектора, враховуючи його початок (Tx) і кінець (Rx) і все, що між ними, а потім, запитуючи, що може піти не так між кожним вектором і наступним.

Крок 4. Підвищення: кожен проблемний випадок або сценарій повинен бути розглянутий в дизайні протоколу. Це дуже допомагає, якщо випадки пронумеровані і класифіковані. Після того, як ці проблеми були вирішені, більш надійний протокол знову моделюється та оцінюється (як у кроках 2 та 3). Це повторюється до тих пір, поки всі питання не будуть належним чином вирішені. Слід очікувати, що буде потрібно кілька ітерацій, оскільки деякі зміни створять свої проблемні сценарії, однак досвід зменшить частоту подібних проблем. Знову ж таки, простежуваність цих проблемних випадків має важливе значення, і детальний їх розгляд повинен гарантувати, що всі відомі проблеми простежуються шляхом розробки, впровадження та тестування.

Перевага цього підходу, (підсумована на малюнку 2.3), полягає в тому, що він не лише надає тестові випадки для тих, хто реалізує протокол у коді, але він також надає вичерпний опис того, як протокол повинен вести себе та за яких умов він, як відомо, працює, таким чином інформуючи тих, хто планує використовувати протокол у своїх системах.

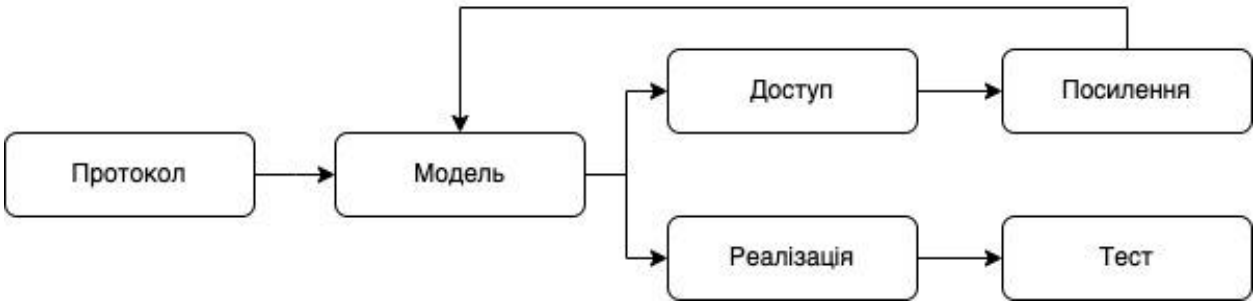


Рисунок 2.3 – Методологія розробки надійної мережі

RNDM може і має супроводжуватися стандартним тестуванням на проникнення та навантаженням для перевірки зроблені припущення та кількісно оцінити ефективність протоколу.

Крок 1 і 2: Наївний протокол і модель

Наївний протокол для цього рішення може бути описаний як наведений нижче, і був змодельований за допомогою діаграми послідовності.

1. МКБ транслює запит UDP на будь-які нові пристрої, щоб оголосити про себе, а потім чекає відповіді.

2. Кожен пристрій отримує цю трансляцію, і якщо вони раніше не оголосили про себе, вони відповідають пакетом UDP, що містить їх ідентифікатор, який буде їх MAC-адресою.

3. МКБ додасть будь-які нові пристрої, які відповідають його існуючому списку і завершать завдання.

Крок 3: Оцінка - Проблемні випадки використання

Кожен випадок використання констатує проблему та її можливі причини.

1. Загальні проблеми передачі при будь-якій передачі:

a. Не вдалося правильно ініціалізувати бездротову систему.

b. Немає підключення до мережі.

2. Загальні проблеми з прийомом для будь-якого прийому:

a. Втрата пакетів через погану потужність сигналу або випадкове зіткнення пакетів.

b. Втрата пакетів через навмисне глушіння.

c. Відсутність підключення до мережі.

3. Невдала трансляція ICD:

a. Бездротова передача ICD не досягає успіху через проблеми з ініціалізацією.

b. Відсутність підключення до мережі.

4. Пристрій не вдалося отримати:

a. Втрата пакетів через перешкоди або погану потужність сигналу.

5. Невдала відповідь:

a. Немає нових пристроїв.

b. UDP означає відсутність гарантії доставки.

6. Забуті пристрої:

a. Якщо пристрій відповідає, але не чується, він припускає, що він був знайдений, і не відповість знову. Таким чином, це неможливо виявити.

b. Якщо пристрій видалено на пульті дистанційного керування, як його можна повторно виявити без повторного налаштування пристрою?

c. Що робити, якщо пристрій з якоїсь причини потрібно скинути, наприклад, оновити прошивку, переналаштувати?

7. Втрачені пристрої:

a. Що відбувається, коли пристрій виходить в офф-лайн: аварійно завершує роботу, видаляється і т.д.? Це призведе до анулювання списку.

8. Кілька пультів:

a. Домашні мережі повинні мати можливість підтримувати кілька пультів, але в даний час пристрій оголосить про себе лише один раз.

9. Кілька мереж:

a. Що станеться, якщо користувач захоче використовувати один і той же пульт дистанційного керування в різних місцях в різних мережах?

10. Переналаштування мережі:

a. Користувачі можуть змінювати свій SSID або пароль

b. Користувачі можуть змінити свій ШЛЮЗ IP.

11. Безпека та видимість (тобто замки):

a. Чи повинен будь-який пульт дистанційного керування знати про будь-який пристрій? Це може бути краще оброблено на наступному шарі.

12. Зняття пристрою:

a. Як можна видалити пристрій з мережі?

13. Ігнорування пристрою:

a. 13.1.Чи повинен пульт дистанційного керування ігнорувати деякі пристрої? Як цього слід досягти?

14. Ігнорування запиту:

a. Чи повинен пристрій бути навмисно невідкритим?

Крок 4: Підвищення

Надійний протокол розроблений для вирішення проблемних випадків використання в попередньому розділі.

2.5 Протокол Robust

Протокол Robust виглядає наступним чином:

1. МКБ транслює пакет UDP на всі пристрої. Він буде включати в себе список перепису всіх відомих пристроїв.
2. Пристрої перевірять список перепису, і ті, кого немає в списку, або чиї дані застаріли, дадуть відповідь МКБ після затримки випадкової тривалості.
3. МКБ чекатиме 1 секунду, поки всі пристрої дадуть відповідь, а потім додадуть або оновлять усі пристрої відповіді до списку.
4. Кроки 1-3 будуть повторюватися, доки відповіді не будуть отримані протягом 1 секунди.
5. Якщо три послідовні спроби не отримують відповідей, процес завершується. В іншому випадку процес починається знову.
6. Протокол також може бути припинений, якщо буде зроблено 5 однакових трансляцій, щоб запобігти втечі.

Список перепису міститиме кожен пристрій:

1. MAC-адреса
2. Остання відома IP-адреса
3. Біт стану (онлайн/офф-лайн)
4. Біт видимості (рівень дозволів)
5. Дата останнього контакту

Ці параметри в цьому протоколі були обрані для досягнення балансу, де це можливо між конкуруючих проблем. Наприклад, більше спроб і більш тривалий час очікування або затримки часто дозволяють для більшого успіху передачі, однак, такі параметри також продовжать час циклу і сповільнять процес Discovery

і дратуватимуть користувачів. Ці значення можуть уточнюється далі шляхом детальної оцінки, але в даний час ці значення досягають корисного балансу.

Одне з великих значень RNDM полягає в тому, що він автоматично забезпечує комплексний набір сценаріїв тестових випадків, які можуть бути використані для перевірки дизайну протоколу на концептуальному рівні, в моделюванні або як реальний, фізичний експеримент.

Розроблений випробувальний стенд імітує МКБ за допомогою сценарію, що працює на віртуальній машині з дротовим підключенням до маршрутизатора Wi-Fi домашньої мережі. Бездротові пристрої IoT були представлені трьома модулями ESP-01 і 2 ESP-12.

Ці модулі ESP є членами сімейства чіпів ESP8266, які представляють чудову можливість для домашньої автоматизації, оскільки вони мають дуже низьку вартість та включений Wi-Fi на борту. Раніше вони були визначені як ідеальні кандидати від реалізації Універсального протоколу приєднання Стіна[35].

Оцінка складалася з трьох різних експериментів, кожен з яких розбився на кілька тестів з різними умовами. Потім ці тести повторювалися 200 або 1000 разів для статистичної строгості. В ідеалі результати цих експериментів можна було б порівняти з існуючими технологіями. Однак, через повсюдність топології постійного центрального контролера, не існує протоколу, що підтримує децентралізований підхід, з яким можна було б зробити змістовне порівняння.
Перший експеримент оцінив проблеми, викликані прикладами, що пакети можуть бути втрачені і порушити протокол. У цьому експерименті МКБ роблять єдину спробу виявити будь-які пристрої. Було проведено п'ять різних тестів: перший мав лише 1 пристрій ESP, другий мав 2 пристрої, а новий пристрій додавався до кожного тесту до п'ятого, з 5 пристроями. Кожен тест проводився 200 разів і був зафіксований відсоток переданих пакетів відповідей, успішно отриманих МКБ.

Результати цього експерименту, як і очікувалося, модулі, схоже, заважають один одному навіть з двома пристроями, причому майже 15% пакетів втрачається. Значно більше втрат спостерігається з більшою кількістю пристроїв, оскільки

перешкоди посилюються до такої міри, що для 5 пристроїв успішно передається менше 60% пакетів.

Другий експеримент складався з двох тестів. Тут високі показники трафіку можуть призвести до високих втрат пакетів. Це вирішується з випадковими затримками відповіді в запропонованому протоколі. В обох тестах МКБ транслює порожній список пристроїв (вказуючи на те, що всі пристрої повинні відповідати) на всі 5 пристроїв. У першому тесті 5 пристроїв запрограмовані на негайну реакцію, в той час як у другому тесті пристрої IoT запрограмовані на посів випадкового числа, використовуючи останній октет своєї MAC-адреси, а потім чекати рандомізований час між 0-200 мс (таким чином реалізуючи запропонований протокол). Кожен експеримент проводився 1000 разів. Результати показують частоту, з якою 0-5 пристроїв успішно виявляються з випадковою затримкою і без неї. Вони вказують на високу втрату пакетів без рандомізованої затримки, а це означає, що тільки 2-4 з 5 пристроїв можуть бути виявлені з будь-якою регулярністю, більшу частину часу виявляються тільки 3 з 5 пристроїв. Тут, як і в наступних цифрах, планки похибок були опущені для дуже невеликих зразків, які мають великі довірчі інтервали. З огляду на вбудовану ємність стандартів 802.11 для обробки зіткнення пакетів, включаючи CSMA / CA і MAC дивно, що так мало відповідей передаються успішно без випадкової затримки.

Тому варто провести подальше дослідження, щоб поставити під сумнів, чи стандарт повністю і належним чином реалізований. Додавання рандомізованої затримки значно покращило результати до такої міри, що всі 5 пристроїв відповіли і були виявлені успішно. Цей експеримент доводить корисність RNDM, оскільки вона змогла передбачити та ефективно вирішити цю проблему.

Третій експеримент оцінював здатність запропонованих протоколів впоратися з випадками використання, які стосуються втрати пакетів та обробки менш надійного характеру UDP. Це демонструє ефективність численних спроб, зроблених в рамках протоколу, щоб виявити всі пристрої. Цей експеримент складався з чотирьох тестів. У всіх тестах МКХ почав з трансляції порожнього списку пристроїв на 5 пристроїв IoT, потім наступна трансляція оновила список,

включивши всі пристрої, відповіді на які були отримані. Оскільки перераховані пристрої не зобов'язані відповідати, очікувалося, що з ростом списку менше пристроїв буде намагатися відповісти, що полегшить виявлення пристроїв через зменшення трафіку.

Кожен з чотирьох тестів проводився 1000 разів з 5 пристроями IoT. Перші два тести були проведені з випадковою затримкою і без неї, що спостерігалася в експерименті 2. Останні два тести були повторенням першого, але з додаванням рандомізованої втрати пакетів, досягнутих шляхом випадкового падіння ioT-пристроїв у 50% випадків. Це імітувало ефект галасливого середовища Wi-Fi, в якому пакети можуть бути втрачені з інших причин, ніж суперечка між пристроями IoT, що відповідають.

Отже, два сюжети, схожі на ті, що отримані в експерименті 2, і показують, що при випадковій затримці, потрібно лише одна спроба захопити всі 5 пристроїв, тоді як 2-3 частіше зустрічаються без затримки. Як і очікувалося, середня кількість спроб збільшується, але випадкова затримка все одно дає кращий результат в середньому 3 спробами, необхідними в порівнянні з 4 спробами без затримки.

Ці експерименти показують, що запропоноване надійне рішення успішно вирішує і обробляє розглянуті проблемні випадки використання. Показано, що протокол підходить для покращення успішної швидкості відповіді на відкриття відповідей від ненадійного пристрою 2-4 до надійного 5 з 5. Крім того, протокол, як показано, здатний відновлюватися від пропущених пакетів через зіткнення пакетів в результаті високого трафіку або одночасної передачі, при цьому, як правило, менше 5 спроб, необхідних для виявлення всіх пристроїв, навіть зі швидкістю падіння пакетів 50%. Крім того, він продемонстрував здатність цього протоколу знаходити пристрої без постійного центрального контролера, не ставлячи під загрозу надійність або функціональність.

Програмування великої кількості пристроїв створює кілька ключових проблем:

1. Організація яких пристроїв була запрограмована, а які ще чекають.
2. Координація даних коду та конфігурації програмується на кожному пристрої.

3. Час, необхідний для програмування всіх пристроїв.

4. Робочий час, щоб запрограмувати багато пристроїв і виправити помилки.

Ці виклики нелегко подолати, тому від масштабної фізичної реалізації мережевої системи часто відмовляються. Дослідження замість цього зосереджені на моделюванні і, можливо, результатах невеликих фізичних реалізацій. З мого досвіду мережевих досліджень, великомасштабна реалізація завжди знаходить значні наслідки, які не очевидні в іншому випадку, тому великомасштабна реалізація є обов'язковою для дослідження, щоб бути ефективними. З цих факторів робочий час, мабуть, є найбільш значущим, особливо коли деякі пристрої мають різні дані коду або конфігурації. Час є ще більшою проблемою, коли є кілька тестів, кожен з яких має різні дані коду або конфігурації.

Дослідження IoT створило нішеву область для програмування пристроїв. Невеликими прототипами можна керувати за допомогою простих ручних методів програмування, в той час як великомасштабні комерційні або промислові програми можуть дозволити собі накладні витрати на більш ресурсомісткі автоматизовані методи програмування.

Дослідницькі програми, часто вимагають більш незручних мереж середнього розміру, які вимагають складне управління конфігурацією і занадто великі для ручних методів, але недостатньо великі для більш вимогливі автоматизовані методи. Описаний тут метод долає всі ці проблеми і надає нішеве рішення для мереж дослідницького масштабу.

В даний час для мережевих додатків, таких як IoT або WSN є двома основними варіантами, доступними для програмних пристроїв, просте дротове програмування і повітряне програмування. Провідне програмування - це один пристрій за часом, який дуже повільний і вимагає багато зусиль, щоб відстежувати більше декількох пристроїв, а також яку версію коду вони використовують. Він часто використовується для дрібномасштабного тестування мережі. Провідне програмування часто віддає перевагу простому прототипуванню, оскільки мережа дуже мала, часто лише з одним або двома пристроями. Коли прийде час

протестувати більшу мережу, багато хто вибере моделі та симуляції, але налаштування цього може бути настільки ж тяжким, або навіть більше, ніж протоколи, які досліджуються, і вартість спеціалізованого програмного забезпечення може бути непомірною. Крім того, такий підхід може легко пропустити ефекти та явища, не передбачені дослідником, тому з цієї причини багато хто все ще віддають перевагу реальному тестуванню на апаратному забезпеченні, але відмовляються від великомасштабного тестування [32].

Бездротовий підхід до програмування «по повітрю» може заощадити багато часу і навіть поліпшити деякі з координації та організації пристроїв. Тим не менш, він вимагає, щоб пристрої були попередньо завантажені з підтримкою прошивки, яка може зайняти значний простір на обмеженому обладнанні, такому як ESP-01. Якщо пристрої не попередньо завантажені опорною прошивкою, то повільний і кропіткий дротовий метод необхідно знову використовувати, хоча б один раз, для налаштування всього. Крім того, пам'ять повинна бути відведена для зберігання цієї прошивки, а також залишити місце для поточного зображення і будь-якого майбутнього зображення. Вимоги до пам'яті можуть бути трохи зменшені за допомогою delta оновлення, але це корисно лише при зміні коду, але не повністю зміні програм.

Ці вимоги до пам'яті стають все більш проблематичними для обмежених додатків, де пристрої мають меншу потужність і зменшене сховище, тому пам'ять не може бути так легко відкладена для таких додатків. Крім того, обмежені програми часто реалізують більш повільні модулі пам'яті, щоб зменшити загальні витрати. Це часто можна зробити, не ставлячи під загрозу регулярну продуктивність, але це значно збільшує час програмування, особливо якщо застосовується індивідуальний підхід.

OTA має сенс, коли всі пристрої мають однаковий код. Коли пристрої мають різні конфігурації, як це часто буває для досліджень, то пристрої повинні бути запрограмовані один за одним, що може зайняти багато часу для великої мережі.

Ця методологія може бути застосована до будь-якої ситуації, коли потрібно запрограмувати кілька пристроїв, але ця робота була зосереджена на застосуванні

її до пристроїв IoT. Наступні процедури продемонструють, як цей конкретний додаток може бути реалізований і дасть уявлення про те, що буде потрібно для різних додатків.

Використовували Raspberry Pi 3 в якості контролера. Нижче наведено процедуру настроювання цього контролера:

Крок 1: Встановіть Raspbian на карту microSD відповідно до документації Raspberry Pi.

Крок 2: Дуже важливо налаштувати правильні параметри системи та інтерфейсу. Якщо цей крок не виконується належним чином, Raspberry Pi не буде функціонувати належним чином. У розділі Меню -> Налаштування виберіть параметри налаштування Raspberry Pi.

На старі змінюємо:

1. Змінюємо ім'я хоста на щось унікальне. У цьому прототипному середовищі використовувалися відповідні номери веж, тобто «Вежа1», «Вежа2».
2. Роздільна здатність: 1920 x 1080 (корисно для налагодження VNC)
3. На вкладці Інтерфейси змініть такі параметри:
4. SSH: Увімкнути (рекомендовано для налагодження та віддаленої конфігурації)
5. VNC: Увімкнути (рекомендовано для налагодження та віддаленої конфігурації)
6. Послідовний порт: Увімкнути (ВАЖЛИВО)
7. Послідовна консоль: вимкнути (ВАЖЛИВО)
8. Перезавантаження Raspberry Pi

Крок 3: Настроювання параметрів мережі. Оскільки модулі ESP-01 можуть бути використані для тестування Wi-Fi, рекомендується, щоб вежі були підключені до мережі управління. У цій прототипній системі кожен пристрій в мережі встановлюється зі статичним IP. Налаштуйте всі вежі та ПК з керуванням з унікальними статичними IP-адресами. Зауважте, що типовою конфігурацією raspberry Pi є підключення до 192.168.0.100, який слід призначити головному контролеру.

Крок 4: Встановіть програмне забезпечення.

Крок 5: Налаштуйте локальний контролер для запуску під час запуску;

sudo nano /home/pi/.config/lxsession/LXDE-pi/autostart

Потім:

@xscreensaver -no-splash

Далі:

@xterm -e /usr/bin/python /home/pi/ESP-Testbed/Tower/Pi_Client.py 192.168.0.100

Нарешті, зберігаємо файл і перезавантажуємо Raspberry Pi.

Регулярні операції дуже прості і можуть бути виконана з головного контролера, припускаючи, що всі вежі налаштовані і налаштовані правильно, і що всі цільові пристрої встановлені.

1. Всі файли зображень повинні бути попередньо скомпільовані. Це можна зробити в IDE, як Arduino IDE.

2. Файл конфігурації потрібно записати, щоб визначити, які файли слід запрограмувати на які пристрої.

3. Потім керування ПК запускається з такою командою:

./mgmt.py -configfile. YAML

4. Після завершення цього процесу результати програмування можна побачити у файлі журналу:

./Mgmt_PC/Журнал.html

5. Цільові пристрої тепер можна запускати і тестувати.

Досвід показав, що створення першого файлу конфігурації займає деякий час, але наступні файли конфігурації дуже прості у створенні. Для того, щоб протестувати і перевірити цей метод, було запрограмовано 39 цільових пристроїв. Цей випадок використання мав на меті оцінити кількість пристроїв, які можна було б легко розмістити на типовому споживчому маршрутизаторі Wi-Fi. Це тестування було проведено за допомогою 3 модулів Towers, кожен з яких має 13 цільових пристроїв ESP8266-01, всього 39 цільових пристроїв. Кожен цільовий пристрій був

завантажений простою програмою, яка б об'єднувала пристрій до попередньо визначеної бездротової мережі.

Для перевірки ємності внутрішньої мережі Wi-Fi кожному пристрою був відправлений пінг-пакет з ПК, підключеного до порту Ethernet того ж маршрутизатора Wi-Fi, який був підключений до цільових пристроїв. Ці пакети були відправлені на кожен цільовий пристрій по одному, щоб перевірити на підключення. Кожен пристрій був pinged з 500 мс тайм-аут з до 10 повторів. Цей тест був запущений, як все більше і більше пристроїв були додані в мережу. Пристрої були додані по три за раз, по одному на кожному модулі Tower. Потім результати тестів були проаналізовані, щоб побачити успішність запитів на пінг.

Результати показують, що після 15 пристроїв додаткові пристрої починають створювати проблеми в мережі і не всі пристрої можуть бути досягнуті. Це не є результатом того, що нові пристрої не визнаються, оскільки нові пристрої були повністю доступні. Тенденція до зниження відсотка доступних пристроїв була обумовлена збільшенням кількості пристроїв, оскільки маршрутизатор D-link, здавалося, підтримував лише 15 одночасних бездротових пристроїв. Тестування за допомогою високоякісного маршрутизатора Wi-Fi високої ємності, UniFi UAP-AC-Pro.

2.6 Висновок до другого розділу

Протокол Насріна і Редкліффа є одночасно новим і чудовим мережевим протоколом приєднання для пристроїв IoT. Ретельний аналіз показав, що протокол можна зіставити з можливостями операційної системи Android. У той час як фізична реалізація протоколу Насріна і Редкліффа, використовуючи смартфон Android і Raspberry Pi, довела, що протокол практично життєздатний, було запропоновано і впроваджено кілька поліпшень, щоб поліпшити легкість, з якою користувачі можуть приєднатися до пристроїв, використовуючи протокол Nasrin і Radcliffe.

Багато простих пристроїв IoT мають світлодіод або зумер під контролем мікропроцесора. У цьому розділі проаналізовано потенціал цих пристроїв для заміни частини QR-коду Протоколу приєднання та зроблено висновок, що такий підхід є життєздатним і може додати додаткової зручності.

Однак, розглядаючи сумісність протоколу Насріна і Редкліффа, стало зрозуміло, що залежність протоколу від пристроїв Android з можливостями гарячих точок означає, що він не зможе інтегруватися з великими сегментами ринку і буде перешкоджати його засвоєнню.

Тому був запропонований новий протокол, заснований на посиленій імплантації протоколу Насріна і Редкліффа. Ключовою особливістю цього протоколу є те, що відповідальність за функціональність точки доступу перекладається на пристрій IoT без додаткових витрат, враховуючи останні інновації в чіпах, таких як ESP8622.

Цей новий підхід забезпечує протокол приєднання до систем домашньої автоматизації без залежності до PCC, без додаткового обладнання, без підвищеної вартістості або складністі і без шкоди для безпеки. Будь-який пристрій з підтримкою Wi-Fi тепер потенційний контролер для події приєднання, незалежно від виробника або операційної системи.Коли не було знайдено відповідного методу для системи без PCC, це перейшло до розробки нового представленого протоколу. Взявши на себе це завдання, було продемонстровано, що не існує достатньої методології розробки, яка б відповідала конкретним потребам розробки мережевих протоколів.

Перед розробкою Протоколу Discovery цей розділ спочатку задокументував уроки, витягнуті з існуючих методологій розробки, а потім сформулював і запропонував методологію розвитку мережі надійності. Потім ця методологія була реалізована при розробці нового протоколу Discovery. Крім того, підкреслюючи значення RNDM, протокол Discovery був легко оцінений за допомогою проблемних випадків використання, виявлених у процесі розробки. Ця оцінка продемонструвала необхідність функцій протоколу, включаючи кілька трансляцій відкриття, а також рандомізовані затримки відповіді.

Експериментальні результати, отримані в цій роботі, демонструють не тільки успішність цього протоколу, але і життєздатність відкриття пристрою без постійного центрального контролера або будь-якої додаткової інфраструктури або послуг, на відміну від альтернативних підходів до відкриття. Цей розділ показав, що залежність систем домашньої автоматизації від PCC зовсім не є необхідною для надійного протоколу discovery.

Розробка цього протоколу також продемонструвала необхідність тестування на реальному обладнанні в масштабі, але це значні перешкоди, тому був розроблений і реалізований метод швидкого масового програмування трьох рівнів. Це змогло показати, що в масштабах більшість вітчизняних маршрутизаторів Wi-Fi будуть боротися за підтримку за межами 15 пристроїв. Це цінні реальні дані, які не будуть отримані за допомогою інших методологій тестування, таких як моделювання або емуляція.

3 РЕАЛІЗАЦІЯ КРОКІВ ПРОТОКОЛУ

3.1 Поєднання елементів інтернету речей

Після того, як пристрої з'єднуються і виявляються в мережі, наступне питання, яке слід розглянути, полягає в тому, як користувачі будуть взаємодіяти з цими пристроями для повсякденного використання. Продовжуючи парадигму, прийняту в попередніх розділах, переривчасті пристрої управління (МКБ) будуть основною точкою інтерфейсу, але для забезпечення цієї функціональності необхідні допоміжні системні архітектури та протоколи. Це звертає увагу на третє дослідницьке питання:

Без постійного центрального контролера, як інтерфейсні пристрої отримають уніфікований і функціональний інтерфейс для досягнення відображення і управління пристроєм.

Це дослідження показало, що існуюча робота, пов'язана з Display and Control, в кінцевому підсумку недорозвинена, і жоден цілісний підхід не об'єднав опис можливостей пристрою, управління, зв'язку та макета інтерфейсу користувача. Тим не менш, є проблеми, які слід враховувати в тому, як МКБ можуть підтримувати свої дисплеї в актуальному стані, оскільки стан пристроїв буде частиною дисплея, але також може інформувати про аспекти механізмів управління. Щоб впоратися з цими проблемами, цей розділ також розробляє і пропонує новий протокол зв'язку статусу для підтримки протоколу відображення та контролю. Цей протокол для зв'язку зі станом пристрою, або "моніторинг", дозволить кільком МКБ, щоб бути в курсі стану пристроїв в пакеті ефективно, щоб звести до мінімуму вплив і без того на переповнений простір 2,4 ГГц. Два терміни, "статус зв'язок" і "моніторинг", використовуються як взаємозамінні, оскільки один розглядає проблему з точки зору МКБ, а інший - з розумного пристрою.

Традиційно з використанням РСС, пристрої користувальницького інтерфейсу будуть йти до контролера безпосередньо для статусу будь-якого даного пристрою. РСС був джерелом «істини» для всіх пристроїв в Системі і мав найактуальніші дані про стан пристрою. Контролер також керуватиме інтервалами оновлення для

кожного пристрою та його різними значеннями стану. Тепер питання полягає в тому, як МКБ забезпечить йому дані про стан, які є достатньо актуальними для кожного конкретного пристрою та його застосування? І як додаткове ускладнення, як кілька МКБ підтримуватимуть сучасний статус, коли вони можуть бути скориговані іншим МКБ. Розглянемо розумний кондиціонер з користувачами дуелі над температурою, на яку повинен бути встановлений прилад. Як би один МКБ підтримував актуальне відображення заданої температури, якщо інший МКБ щойно змінив значення?

Будь-яке рішення цієї проблеми має звести до мінімуму мережевий трафік, оскільки домашнє середовище може швидко стати дуже зайнятим, оскільки все більше і більше ioT-пристроїв додаються до системи домашньої автоматизації. Одна мережа може легко складатися з 150 або більше пристроїв, і тому успішне рішення мінімізує пакети, навіть якщо вона підтримує своєчасний графік оновлення для декількох МКБ.

Це дослідження запропонувало чотири нові протоколи, які дозволять кільком МКБ підтримувати актуальну інформацію про стан з IoT Devices. Ці протоколи оцінюються і порівнюються один з одним, UPnP для ефективності пакетів і забезпечують життєздатне рішення з мінімальним трафіком, отже, мале порушення роботи домашньої мережі. Подальший аналіз розглянув заходи, які необхідно вжити для забезпечення надійне та зручне виконання протоколів.

3.2 Проектування елементів протоколу

Щоб забезпечити надійне та розумне рішення для підтримки даних про стан пристроїв у МКБ, проблема спочатку спрощується шляхом обмеження роботи протоколу до часу, коли МКБ спостерігають за станом конкретного пристрою, тобто активним. Це обмежує проблему створення протоколу, який може керувати оновленнями пристрою та встановлювати їх частоту. З цією метою пропонуються чотири потенційні рішення і досліджується їх поведінка. Основним дискримінаційним фактором між цими протоколами буде кількість пакетів,

згенерованих протоколом, оскільки домашнє бездротове середовище вже може бути зайнятим і не повинно завантажуватися без необхідності. Всі рішення будуть використовувати прості пакети UDP для зв'язку. Ці протоколи спрямовані на те, щоб скористатися ключовим розумінням від MQTT і CoAP, а саме реєстрації інтересів та ефективності посередників. Вони будуть зіставлені з теоретичним виконанням UPnP за допомогою простої команди опитування, виданої на бажаному інтервалі оновлення МКБ. Вони розраховуються, оскільки протокол такий простий.

Перший протокол, є найпростішим і призначений для функціонування як еталон, оскільки він мінімізує передачу пакетів, жертвуючи гнучкістю. У цьому протоколі пристрої IoT, які будуть спостерігатися, будуть постійно транслювати свій статус в мережу через фіксований інтервал, а зацікавлені МКБ можуть прослуховувати і записувати статус пристроїв. Однак, оскільки IoT-пристрій буде продовжувати транслювати свій статус, навіть якщо жоден пристрій не зацікавлений, він не є практичним рішенням, а мається на увазі лише як порівняльний орієнтир.

Другий протокол, робить МКБ більш активною частиною рішення, дозволяючи пристроям IoT передавати їх лише тоді, коли МКБ зацікавлене. Таким чином, МКБ повинні спочатку зареєструвати свій індивідуальний інтерес за допомогою пристрою IoT разом з інтервалом, з яким вони повинні бути оновлені, подібно до підписки або спостереження MQTT і CoAP. Крім того, це становитиме постійний інтерес, поки МКБ не оголосить, що більше не зацікавлений.

Третій протокол, зменшує навантаження, розміщене на пристрої IoT попереднім протоколом, маючи МКБ, щоб зробити більш загальну реєстрацію інтересу, просто повідомивши пристрій IoT про те, що «хтось» зацікавлений і як часто вони вимагають оновлення. Потім пристрій IoT відстежує лише те, скільки МКБ цікавить, і найкоротший інтервал оновлення. Потім він транслюється з таким інтервалом, поки кількість зацікавлених МКБ не досягне нуля.

Знову приймаючи модель реєстрації від MQTT і CoAP, цей протокол також дещо спирається на ідею проміжних кешів в CoAP. Цей підхід зменшує кількість відповідей, надісланих пристроєм IoT, але замість того, щоб використовувати

посередників, він надсилає пакет трансляції на всі МКБ, які можуть зберігати дані, якщо вони зацікавлені.

Четвертий протокол, мінімізує обчислювальний попит на пристрій IoT, вимагаючи від нього лише слухати та відповідати на запити на оновлення, а не відстежувати зацікавлені пристрої. МКБ все ще роблять індивідуальні запити на оновлення через регулярні проміжки часу, а пристрій IoT транслює свою відповідь. Однак, коли МКБ отримує оновлення, яке він не просив, він приймає оновлення, і скидає його інтервал таймера і збільшує інтервал трохи на фіксований інтервал зворотного виходу. Це дає час іншому пристрою запитувати його наступне оновлення до закінчення терміну дії таймера, і він запитує власне оновлення.

Для того, щоб оцінити продуктивність кожного з чотирьох протоколів, кожен з них був реалізований в скриптах C++ як для іоТ-пристроїв, так і для МКБ. Розроблена архітектура випробувального стенду була дуже ефективною і цілком може бути корисною для іншого аналізу протоколів.

Оцінка мережевого протоколу може бути досягнута декількома способами: розрахунок, моделювання, емуляція і реалізація. Аналітичне моделювання може дати ключове уявлення про продуктивність, наприклад, в [73], [74], де воно використовувалося для оцінки нових підходів до мережевих протоколів, таких як CSMA / CA, і для оцінки надійності систем, що використовують сервісно-орієнтовані архітектури. Однак розглянуті тут протоколи можуть включати п'ять або більше складних скінченних державних машин, що створило б надмірну складність в аналітичній моделі, і така модель все ще може пропустити важливі явища в реальному світі. Теорії черг можуть бути корисним підходом до багатьох ситуацій, подібних до роботи, проведеної тут. Однак занадто багато припущень, зроблених цими теоріями, такі як випадкові міжаррівні часи, порушуються реаліями IP-мереж і запропонованих протоколів. Ні моделювання, ні розрахунок не можуть легко запропонувати таке ж уявлення про поведінку протоколу в порівнянні з тим, коли він реалізується на реальному стеку TCP /IP. Цей урок вже був помічений несподівано, коли було виявлено, що типові домашні маршрутизатори можуть підтримувати лише 15 одночасних сеансів, проблема, не

передбачена моделюванням або теорією. Емуляція відповідає всім реальним правилам реалізації, але може використовувати різні апаратні та програмні засоби для зменшення зусиль з розробки. З цієї причини протоколи були емульовані за допомогою екземплярів VM Tiny Core Linux 9.0 [75] під управлінням скриптів C++ для кожного протоколу [76]. Ці скрипти включали діагностичний код для запису часу прибуття пакета, а також загальну кількість відправлених пакетів.

Шість віртуальних машин (див. рисунок 3.4) були використані для оцінки кожного протоколу, один для представлення пристрою IoT, чотири представляють МКБ, а один використовувався для координації всіх експериментів. Ці віртуальні машини, керовані та мережеві за допомогою GNS3 v2.1.8, який імпортував крихітні віртуальні машини ядра з VirtualBox v6.1.28 і надав перемикач Ethernet для підключення. Початкове тестування використовувало дротову мережу, вільну від суперечливих ефектів бездротових мереж, але в майбутньому роботі потрібно буде розглянути ці ефекти. Кожній з віртуальних машин була надана статична IP-адреса, запустили FTP-сервер, щоб спростити передачу файлів у віртуальну мережу, і запустили compiletc v0.1 для компіляції скриптів C++.

Кожен з чотирьох протоколів був запущений протягом 300 секунд з активністю лише 1 МКБ (тобто спостереження), а потім для 2 МКБ і так далі, до фінального тесту, який запустив всі 4 МКБ. Цей набір тестів повторювався тричі, кожен з яких мав різну комбінацію інтервалів оновлення, встановлених у МКБ. Інтервал за замовчуванням був встановлений на рівні 5 секунд, активних у першому раунді випробувань, в яких всі МКБ були встановлені на запити інтервалу за замовчанням. Другий раунд встановив перше оновлення МКБ на 10 секунд, а в третьому, він був встановлений на 1 секунду.

Протоколи 2-4 мають час налаштування, і вони не працювали нормально протягом перших кількох секунд тесту. Всі тести проводилися протягом 300 секунд, щоб звести до мінімуму вплив часу установки.

Замість того, щоб вручну налаштовувати мережу для кожного експерименту, який вимагав би налаштування та налаштування кількості активних пристроїв та їх окремих інтервалів оновлення, машина-координатор (див. рисунок 3.1)

використовувалася для автоматизації всіх експериментів для кожного протоколу. Після кожного експерименту координатор витягував кількість пакетів, переданих кожним пристроєм, і записував загальну суму. Це означало, що всі експерименти можуть бути завершені шляхом проведення чотирьох партійних експериментів, по одному для кожного протоколу. Цей експеримент був дуже успішним. Він виявився дуже ефективним і гнучким. Він дуже адаптований і може бути легко застосований до багатьох інших досліджень виконання протоколу.

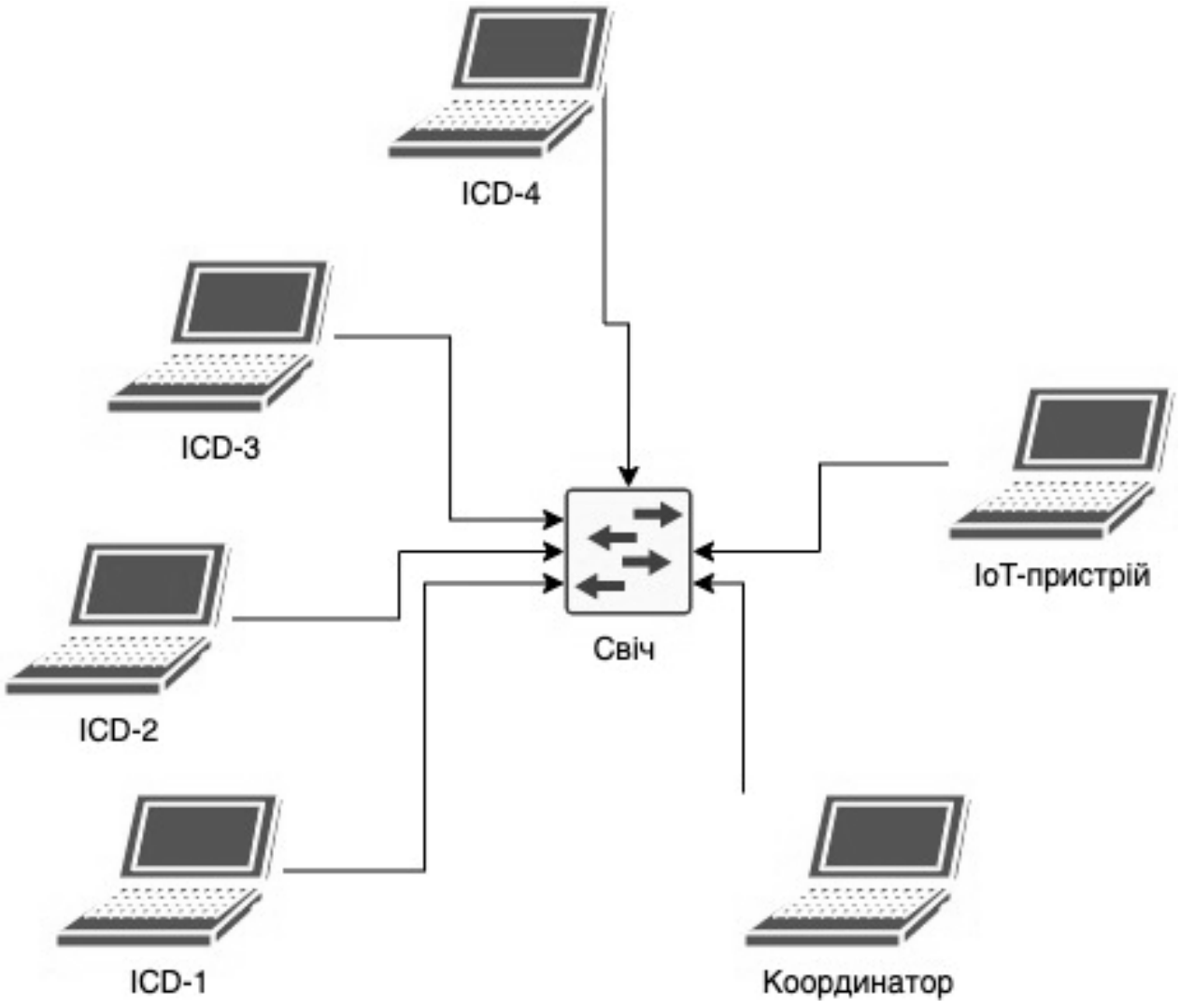


Рисунок 3.1 – Налаштування тестового стенду з ICD 1-4, підключеним до пристрою IoT і координатора експерименту через комутатор Ethernet в GNS3

У першому наборі тестів були всі пристрої, які очікували оновлень кожні 5 секунд. Результати 300-секундних тестів, який показує, скільки пакетів відправляється кожні 5 хвилин кожним протоколом в залежності від кількості активних пристроїв. Початкова оцінка результатів сприяла б непрактичному Протоколу 1 як найкращому виконавцю, але не слід забувати про близьке виконання Протоколу 3. Той факт, що Протокол 1 зберігатиме швидкість передачі, навіть якщо МКБ активний, показує, що продуктивність Протоколу 3 є явно вищою, враховуючи його граничне збільшення пакетів над Протоколом.

Протокол 4 зменшує вимоги, пред'явлені до пристрою IoT, але його поведінка приблизно в два рази перевищує Протокол 1 або 3. Нарешті, протокол 2 є проблематичним, оскільки він надзвичайно чутливий до числа активних МКБ.

У другому наборі тестів перший МКБ, який буде активований, потребував більш тривалого інтервалу оновлення в 10 секунд, в той час як всі інші МКБ залишалися на рівні 5 секунд. Непрактичний Протокол 1 має найбільшу кількість пакетів, переданих для 1 активного МКБ. Це підкреслює слабкість протоколу в його негнучкості, оскільки всі інші протоколи враховують бажаний інтервал оновлення МКБ. Навіть Протокол 4 перевершує непрактичний Протокол 1 у цьому тесті. Після цього, однак, додавання 2-го, 3-го і 4-го МКБ, тими ж інтервалами оновлення, що і пристрій IoT, бачить, що протокол 1 працює найкраще, за яким уважно стежить більш практичний протокол 3 (як зазначалося в попередньому наборі тестів).

Протокол 2 слідує тій же лінійній поведінці, на яку не впливає зміна частоти оновлення, за винятком загальної кількості передач. Протокол 3 і 4, присутній як більш постраждалий від більш тривалого інтервалу оновлення, демонструючи, що вони більш чутливі до інтервалу оновлення, ніж кількість активних МКБ.

Більш висока швидкість пакетів з Протоколу 4 підкреслює ціну, сплачену за знижене навантаження на пристрій IoT, ціну, яка перевищує раніше набагато слабший Протокол 2, який виграє від процесу реєстрації, навіть якщо він зобов'язаний вирішувати кожен МКБ окремо. Протокол 4 має набагато більш стабільну кількість пакетів, послідовно 598 пакетів. Це пов'язано зі значно коротшим інтервалом оновлення першого МКБ, що робить його чітким «лідером»

при ініціації оновлень трансляції з пристрою IoT. Коли є кілька МКБ з інтервалами оновлення в найкоротші терміни або поблизу них, значна кількість додаткових пакетів може бути згенерована на етапі суперечки, оскільки МКБ з'ясовують, який пристрій буде лідером, запитуючи оновлення, а який просто слухатиме відповіді на трансляцію. Цей процес суперечок також може бути знову запалений, якщо запити лідера коли-небудь відстають від МКБ, які приєднуються до лідера. З цієї причини інтервал зворотного відключення використовується для збільшення інтервалу оновлення пристроїв, коли вони визнають, що вони не в інтересах.

Результати, показані тут, мають мінімальну суперечку через використання 1,5-секундного інтервалу зворотного відкату. Більш низький інтервал в 1 секунду призвів до набагато менш послідовної поведінки з більшою кількістю подій суперечок і більш високою кількістю пакетів.

Ці результати добре порівнюються з прогнозованою продуктивністю UPnP за допомогою команд опитування. Всі чотири протоколи перевершують UPnP, незважаючи на те, що UPnP відповідає краще, ніж протокол 1, на зміни інтервалів оновлення. Хоча UPnP тісно відповідає Протоколу 4 лише для кількох МКБ, він не масштабується майже так само добре.

3.3 Подальший аналіз та надійність

Незважаючи на те, що протоколи виявилися по-різному ефективними пакетами, надійність і зручність використання протоколів заслуговують подальшого розгляду.

З метою аналізу надійності протоколів розглядаються потенційні проблемні випадки, які можуть виникнути при регулярному використанні цих протоколів. Другий і третій протоколи вимагають від МКБ зареєструвати свій інтерес, а потім оголосити, коли вони більше не зацікавлені в статусі пристроїв IoT. Однак, якщо пристрої не оголосять про це або оголошення якимось чином пропустить пристрій IoT, то ефективність пакетів протоколу буде негативно порушена. Наприклад, МКБ може просто і легко вийти за межі досяжності бездротової мережі і бути не в змозі

оголосити про свій втрачений інтерес до пристрою IoT. Механізм оренди повинен бути доданий до цих протоколів як всеосяжний, щоб запобігти дії Протоколу 2 і 3 більше, як Протокол 1, і постійно транслювати свій статус в мережу незацікавлених пристроїв. Самі МКБ повинні мати можливість запитувати цей період оренди, оскільки різні програми можуть мати різні вимоги, і ці протоколи можуть бути корисними для зв'язку ioT до ioT-пристроїв, таких як терморегулятор для обігрівача або датчик світла для контролерів завіси. Пакети все ще можуть транслюватися без присутності зацікавленого пристрою, якщо оренда занадто довга, а МКБ не повідомляють про свої змінені відсотки. Щоб уникнути цієї проблеми, пристрої IoT повинні обмежити прийнятні терміни оренди або відхилити занадто довгі запити, або просто застосувати встановлену максимально дозволену оренду.

Ще однією проблемою надійності є здатність МКБ запитувати інтервал оновлення з пристроїв IoT. Дуже короткі періоди оновлень можуть призвести до значного трафіку, можливо, до рівня атаки відмови в обслуговуванні. Знову ж таки, максимальна частота оновлення повинна бути обмежена IoT пристроями. Висока частота оновлення з МКБ буде скорочена до цієї межі.

Основне використання цих протоколів полягає в тому, щоб МКБ надавали користувачам актуальну інформацію про стан з пристроїв IoT. Хоча більшість протоколів на основі TCP/IP дуже чутливі з мінімальними затримками, важливо враховувати ефективність протоколу, щоб гарантувати, що поведінка протоколів не вплине на користувальницький досвід. Галіц запропонував багато гідних міркувань про те, як UIs можуть бути розроблені та реалізовані для оптимізації користувацького досвіду [77]. Основний внесок тут полягає в тому, що користувачі віддають перевагу послідовності в часі затримки над більш короткими затримками. Галіц пропонує загальне правило, згідно з яким відхилення затримок не повинно перевищувати половину середнього значення:

Затримки в оновленні елементів інтерфейсу користувача, які не підкоряються цьому правилу, можуть бути джерелом занепокоєння або тривоги у користувачів або можуть призвести до того, що користувачі взагалі нехтують елементом. Протокол, який використовується для оновлення МКБ, повинен дозволити МКБ

підтримувати частоту оновлення, яка відповідає правилу Галіца. Щоб оцінити здатність Протоколу 2-4 підтримувати це правило, затримка кожного пакета оновлення була записана проти очікуваного часу прибуття. Деякі протоколи, такі як 2 і 4, дозволяють надходити раннім пакетам, щоб ці пакети мали негативну затримку, але були скорочені і записані на нуль, оскільки ранні дані можуть бути буферизовані.

Затримки пакетів були зафіксовані з 4 зацікавленими МКБ, які оновлювалися з інтервалом в 5 секунд. Розподіл затримок пакетів для МКБ з найвищим співвідношенням стандартного відхилення до середнього значення для кожного протоколу.

Порівняння між різними протоколами мають обмежену цінність, оскільки розміри вибірки дуже різні через різну швидкість пакетів кожного протоколу. Однак всі вони досить близькі. В діапазоні від ~ 3000 до 0 мкм, з негативними затримками скорочується до 0 сек. Дані описують, як довго пакети затримувалися, але не кількість часу між пакетами. Щоб отримати час між пакети, 5 секунд повинні бути додані до затримки кожного пакета. Ці дані наведено в таблиці 3.1.

Всі три протоколи мали стандартне відхилення значно нижче середнього значення. Це означає, що всі розглянуті протоколи дозволять МКБ підтримувати дотримання правила Галіца.

Раннє оновлення стану може поставити під загрозу цю відповідність, але їх можна легко обробити за допомогою одноступеневої черги для буферизації даних, що дозволяє інтерфейсу користувача оновлюватися через власний визначений інтервал.

Таким чином, користувач не повинен бути роздратований змінними частотами оновлення дисплея, змушуючи їх ігнорувати елемент інтерфейсу користувача, що представляє дані стану.

Таблиця 3.1 – Стандартне відхилення та середнє значення часу між пакетом оновлення в (µs)

	STD(µs)	**Середнє значення(µs)**
Протокол 2	189.08	5000253.07
Протокол 3	420.14	5000279.08
Протокол 4	229.50	5000997.98

Оскільки зв'язок зі статусом для декількох МКБ тепер вирішений, кінцевим компонентом, який слід враховувати, є відображення та управління пристроями, незалежними від PCC. Для цього потрібно, щоб пристрої могли описувати свої можливості, як ними можна керувати та макет інтерфейсу користувача. Це дозволить МКБ представити інтерфейс користувача, який може відображати стан пристрою і дозволить користувачеві керувати або взаємодіяти з пристроєм.

Також є новий протокол, який дозволяє користувачам відображати і керувати домашніми пристроями IoT. Це робиться за допомогою одного файлу опису пристрою XML, який об'єднує опис можливостей пристрою, процедури команд і управління та макет UI. Використовуючи цей єдиний і компактний файл, що зберігається поза коробкою в пристроях IoT, МКБ можуть генерувати повністю функціонуючий інтерфейс користувача для відображення стану пристроїв IoT, а також надавати команди для управління пристроєм IoT. Це дозволить МКБ досягти

інтеграції plug-and-play для пристроїв IoT, при цьому багато пристроїв різних виробників контролюються одним додатком. Крім того, новий файл опису, що підсумовує кілька аспектів пристрою, дозволить в майбутньому розробити для розміщення зв'язку між пристроями всередині будинку від машини до машини.

3.4 Висновки

В результаті розроблений протокол для центрального контролера в кіберфізичній системі «розумний будинок», який створено згідно розробленого методу. Перевірено його надійність шляхом проведення необхідного аналізу.

4 ПРОТОКОЛ ТА ЙОГО ТЕХНІЧНЕ РІШЕННЯ

4.1 Технічне рішення

Як було описано, опис пристрою в тій чи іншій формі не є новою ідеєю, але жодна робота на сьогоднішній день не запропонувала рішення, яке дозволяє одному файлу опису дозволити МКБ відображати стан пристрою, представляти макет для інтерфейсу керування та координувати команди управління. Запропоноване рішення досягає цих ключових цілей, а також вимагає невеликих і простих пакетів і ніякої залежності від постійного центрального контролера. Передбачається, що це рішення стане стандартом для опису можливостей пристрою і того, як відображати і керувати пристроями, але для реалізації стандарту може бути розроблено кілька різних додатків, що дозволить клієнтам вибрати реалізацію, яка найкраще підходить для них. Виробники також можуть вносити заявки, які найкраще підходили до їхніх пристроїв, але які також з'єднуються з пристроями з інших виробників.

Пристрої будуть описані мінімалістичним описом XML-пристрою, який буде зберігатися на пристрої IoT. Це дозволить відкрити реалізацію інтерфейсів пристроїв, зберігаючи при цьому певний контроль над макетом і структурою UIs для виробників. Цей опис визначить всі елементи керування та відображення, які повинні бути присутніми в інтерфейсі користувача пристрою. Коли МКБ вперше виявить новий пристрій IoT в мережі, він отримає опис пристрою IoT, що дозволить МКБ генерувати інтерфейс користувача на основі опису.

МКБ будуть генерувати графічні інтерфейси з описів пристроїв XML за допомогою локального аналізатора словника та генератора, встановленого як частина програми ICD. Словник буде відкритим і загальнодоступним стандартом, який визначає, що кожен елемент як він і повинен бути описаний, в той час як генератор буде реалізацією конкретного визначення того, як елемент повинен з'явитися. Це схоже на концепцію, що використовується в HTML, де веб-сторінка використовує стандартні визначення для створення розділу тексту, який є "жирним", але окремі браузери визначать, як насправді виглядає жирний шрифт.

Це дозволить різним розробникам МКХ створювати власні теми та скіни, які можуть вибирати альтернативні колірні схеми або представлення елементів - наприклад, тумблерний елемент може бути представлений як перемикач або як кнопка, що змінює колір - але стандартний набір елементів дозволить виробникам контролювати основний макет і функціональність графічного інтерфейсу.

Кожен елемент в описі матиме ідентифікатор, який дозволить направити дані про стан пристрою на правильні елементи дисплея. Ідентифікатори також будуть використовуватися при активації елементів управління. МКБ надішле загальне контрольне повідомлення на пристрій IoT, що ідентифікує ідентифікатор елемента та будь-яке пов'язане з ним корисне навантаження, що дозволить МКБ залишатися сторонім щодо пристрою IoT або команди, які надсилає користувач. Таким чином, опис, заснований в основному навколо макета інтерфейсу користувача, також буде охоплювати управління пристроєм і можливості.

Пропонований протокол, використовує опис пристрою XML і вимагає від усіх пристроїв IoT зберігати цей файл і поширювати його на МКБ за запитом. Потім МКБ можуть використовувати цей опис і запускати його через словник для створення графічного інтерфейсу. Потім користувачі можуть взаємодіяти з графічним інтерфейсом, і запитувані дії будуть відправлені назад на пристрої IoT як загальні сповіщення, які потім можуть бути інтерпретовані у відповідні дії.

Абстрактний XML, визначає окремі елементи інтерфейсу користувача та групує їх на основі того, як вони повинні відображатися, а ідентифікатори використовуються, щоб допомогти додати порядок до цих груп. ICD приймає це визначення і генерує відповідний інтерфейс користувача, який поверне будь-яку взаємодію користувача як посилання на ідентифікатор елемента інтерфейсу користувача та будь-яке відповідне корисне навантаження, яке пристрій IoT може інтерпретувати та діяти відповідно. Таким чином, МКБ може залишатися в значній мірі сторонім щодо характеру та поведінки будь-якого пристрою, забезпечуючи при цьому зручний інтерфейс користувача та можливість керувати пристроєм.

Опис пристрою XML не містить змінних стану або значень, оскільки вони запитуються пізніше в окремому пакеті оновлення стану. Це дозволяє проводити

відмінність між регулярно мінливими значеннями та незмінними або менш змінними значеннями. Це зменшує розміри пакетів і, отже, ефірний час, що має вирішальне значення в сильно оспорюваному просторі Wi-Fi 2,4 ГГц [19]. (Хоча 5 ГГц Wi-Fi або Ethernet може полегшити цю проблему, кожен з них поставляється зі своїми власними обмеженнями) Ці пакети оновлення впорядковані (розділені комами) і можна декодувати з метою оновлення відповідних елементів інтерфейсу користувача. Знову ж таки, вся специфіка пристрою IoT обробляється самим пристроєм, в той час як МКБ просто діє на основі визначення кожного елемента, тому XML не визначає типи даних, оскільки визначення кожного елемента є стандартним і таким чином, мається на увазі тип даних.

Абстрактний XML може бути використаний для опису будь-якого пристрою і може вказувати будь-які gui_elements але, працюючи над публічним стандартом, буде доступний словник відомих gui_elements. Виробники можуть бути сертифіковані, як сумісні зі стандартом, використовуючи лише gui_elements, як описано в загальнодоступному словнику. Крім того, сертифіковані програми для МКБ повинні бути здатні належним чином обробляти всі зазначені словникові gui_elements.

4.2 Реалізація протоколу

Для підтримки цього протоколу були розроблені додатки для роботи на смартфонах під Android і iOS, а також кросплатформний електронний додаток для настільних комп'ютерів, ноутбуків і планшетів під управлінням Windows, macOS і Linux [78]. Всі ці програми підтримуються бібліотеками, які дозволяють абстрактний XML-файл з пристрою IoT читати і запускати через словник елементів і генерувати повністю функціональний інтерфейс користувача, який може відображати поточний стан пристрою і відправляти команди для управління пристроєм. Всі 12 елементів, ідентифікованих розробниками, були успішно інтегровані в ці бібліотеки.

Повна функціональність цього простого абстрактного визначення демонструється в повністю функціональних UI, які він може генерувати. Два повністю реалізованих XML абстрактних файлів від розробників були використані додатком Android для реалізації інтерфейсу Thermostat і інтерфейсу користувача камери безпеки.

Як уже згадувалося, простір Wi-Fi 2,4 ГГц сильно оскаржується в домашньому середовищі, і тому мінімізований ефірний час має вирішальне значення як для мінімізації використання цього простору, так і для максимального успішного використання. Запропонований протокол вже мінімізує вплив передач, обмежуючи їх одноразовими комунікаціями, коли МКБ вперше виявляє новий пристрій. Описи пристрою XML ще більше мінімізують цей вплив, використовуючи просту та коротку анотацію XML, як це представлено. Обидва мають розміри файлів менше 2 КБ.

Запропоноване рішення пропонує підхід, який забезпечить сумісність від усіх виробників, що відповідають вимогам, і керуватиме всіма пристроями з одного додатка.

Розмір програми має найбільше значення, якщо встановлюється кілька додатків і хоча всі розглянуті програми мінімальні в порівнянні з типовою доступністю пам'яті на смартфонів, сума багатьох додатків може вимагати дратівливого обсягу пам'яті і, можливо, часу процесора, якщо багато хто працює в той же час. Очевидно, що DHAP має набагато менший розмір завдяки тому, що велика частина інтерфейсу користувача генерується після установки і з додаванням XML-файлів з самого розумного пристрою. DHAP також може обробляти кілька пристроїв, і тому не займе багато часу процесора з кількома екземплярами.

Незалежність виробника є ключем до оперативної сумісності, і, хоча DHAP розроблений, щоб повністю задовольнити цей критерій, інші системи не є по-справжньому незалежними. Хоча вони дозволяють деяку сумісність з іншими пристроями за допомогою голосового помічника або будь-якої іншої узгодженої домовленості, все це вимагає спеціальної інтеграції. Примітно, що терморегулятор Nest широко сумісний з різними системами опалення та охолодження, але це не те

ж саме, що сумісність, коли розумний пристрій співпрацює з іншим. Наприклад, розумні перемикачі Belkin WeMo, що контролюють потужність лампи, телевізора або радіо, не демонструють сумісності.

Нарешті, залежність від постійного центрального контролера є ключовим питанням, і більшість виробників пропонують спосіб обійти використання PCC. Однак, як і було обговорено, це сталося ціною оперативної сумісності. Таким чином, в той час як Philips Hue все ще вимагає постійного центрального контролера, щоб мостові протоколи Wi-Fi і ZigBee, WeMo і Nest, та DHAP, можуть працювати без постійного центрального контролеру у деяких конфігураціях.

DHAP показує реальну обіцянку в наданні універсального протоколу для підключення розумних пристроїв в домашніх умовах. Об'єднання Display, Status and Control надає невелику програму, але дозволяє пристроям будь-якого сумісного виробника підключатися та керувати ними з однієї програми. DHAP дозволяє кросплатформний доступ і не залежить від постійного центрального контролера в будь-якій конфігурації.

4.3 Інтеграція протоколів

Попередня робота продемонструвала, що пристрої можуть бути з'єднані, виявлені, відображені та керовані без громіздкого PCC. Однак кожен з цих протоколів був продемонстрований лише ізольовано, і ще належить з'ясувати, чи можуть вони, по суті, інтегруватися в одне повністю кооперативне рішення.

У цьому розділі представлено повне рішення для усунення PCC, об'єднання всіх протоколів в одну реалізацію. Це демонструє сумісність кожного з протоколів один з одним і що один додаток може керувати всіма пристроями в типовому будинку.

Уявіть собі, що середній користувач щойно повернувся додому з новим недорогою розумною розеткою з підтримкою Wi-Fi. Він має домашню бездротову мережу та смартфон, ноутбук, планшет або інший пристрій з підтримкою Wi-Fi під управлінням програми, що дозволяє їй функціонувати як МКБ. Для початку новий

пристрій увімкнено, і він починається з протоколу приєднання, транслюючи свій заводський SSID за замовчуванням, а потім користувач завантажує програму на своєму МКБ, щоб приєднати цей новий пристрій до своєї мережі. Додаток запитує SSID і Пароль нового пристрою і, можливо, дає альтернативну можливість сканування QR-коду. Після того, як користувач надав ці облікові дані, МКБ приєднується до мережі, створеної їх новим пристроєм. Тут МКБ може направляти користувача шляхом встановлення деякої початкової конфігурації пристрою, принаймні імені та місцезнаходження пристрою, щоб допомогти користувачеві ідентифікувати пристрій пізніше. Коли конфігурація буде завершена, МКБ надішле пристрою облікові дані для домашньої мережі Wi-Fi, і обидва пристрої перейдуть до домашньої мережі.

Потім починається фаза відкриття. Користувач може мати додаткові пристрої в своїй мережі, але, принаймні, йому потрібно знайти цей новий пристрій на вже існуючому домашньому Wi-Fi. МКБ починає процес Discovery, отримуючи список всіх активних пристроїв в мережі. Потім МКХ попросить кожен пристрій по черзі про заголовок XML-файлу опису свого пристрою. Це дозволить МКХ представити виявлені пристрої користувачеві в організованому списку, відображаючи визначені користувачем імена та, можливо, пристрої групування на основі їх розташування.

Цей список є першим екраном, представленим користувачеві в протоколі відображення та керування. Коли пристрій вибрано, МКБ спочатку гарантує, що він має повний файл опису пристрою для пристрою, а потім використовувати його для створення відповідного інтерфейсу користувача, як зазначено виробником. МКХ також зареєструється у вибраному пристрої, щоб почати отримувати регулярні оновлення стану, які включатимуть усі відповідні значення, які повинні відображатися або виводитися елементами інтерфейсу користувача, визначеними файлом опису пристрою та згенеровані МКБ. Після отримання цієї реєстрації пристрої почнуть транслювати свій статус відповідно до Протоколу 3.

Пакети стану матимуть кілька полів, що описують поточний стан пристрою. Перше поле буде задокументувати номер поточної версії файлу опису пристрою. Я кщо це число не збігається з тим, що записано МКБ, МКБ має запросити оновлення

файлу опису. Це знадобиться лише для суттєвих змін значень, які не є важливими наприклад, ім'я пристрою або розташування.

Решта полів пакетів стану ідентифікують елемент інтерфейсу користувача та його поточне значення, яке буде прочитано МКБ, що використовується для оновлення інтерфейсу користувача.

Керуючі повідомлення будуть функціонувати так само, як оновлення стану. МКБ просто ідентифікує елемент інтерфейсу користувача, який був змінений, і нове значення або значення, які він отримав. МКБ не знає, що і як це змінить пристрій, так як пристрій відповідає за інтерпретацію цих нових значень. Таким чином, файл опису одного пристрою використовується для опису можливостей, верстки та команд управління пристрою.

Після того, як користувач закінчить дивитися на пристрій і / або керувати ним, він може залишити інтерфейс і повернутися до списку пристроїв. На даний момент МКБ повідомить пристрій про те, що він більше не зацікавлений в статусі пристрою і буде ігнорувати майбутні оновлення статусу. Якщо більше ніяких пристроїв не цікавить статус пристрою, то він перестане транслювати його статус.

4.4 Висновки до четвертого розділу

Реалізовано протокол для центрального контролера в кіберфізичній системі «розумний будинок» згідно розроблених рішень.

ВИСНОВКИ

Це дослідження показало, що постійний центральний контролер (PCC) може бути видалений з систем домашньої автоматизації без шкоди для функціональності системи. Крім того, це покращує загальну функціональність Системи. Нові та нові протоколи, вироблені в цій роботі, були розроблені для зниження витрат, складності та негнучкості, а також покращення користувацького досвіду та варіантів. Ці протоколи не тільки досягли цих цілей, але і дозволяють будь-якому виробнику реалізувати їх на будь-якому пристрої. Таким чином, ці протоколи дозволяють загальну сумісність.

Це дослідження поставило три ключові питання про наслідки видалення PCC з системи домашньої автоматизації. Перше питання: «Безпечне приєднання без постійного центрального контролера було продемонстровано для старих версій Android. Які зміни потрібні, щоб розширити це ширше?» Відповідь була дана в розділі 3. У цьому розділі було продемонстровано, що переорієнтація теоретичного протоколу приєднання, спочатку запропонованого Насріном і Редкліффом, дозволить інтелектуальним пристроям бути підключеними до мережі Wi-Fi без PCC і буде універсально сумісна з будь-яким пристроєм, який був включений Wi-Fi.

Також у розділі 3 було дано відповідь на друге питання: "Як пристрої в локальній мережі можуть бути виявлені пристроями інтерфейсу без постійного центрального контролера?". У цьому розділі представлено новий протокол виявлення пристроїв, який можна використовувати для виявлення розумних пристроїв у локальній мережі без PCC. Для розробки цього протоколу була запропонована і впроваджена нова методологія розвитку мережі Robustness (RNDM) для задоволення нішевих потреб розробки мережевих протоколів. Крім того, трирівневий метод швидкого масового програмування був розроблений разом з масштабованим випробувальним стендом, який дозволив швидко і легко програмується і тестується, з підтримкою масштабу набагато вище. Це дало реальні результати.

У розділі 4, він третє питання: "Без постійного центрального контролера, як інтерфейсні пристрої отримають єдиний і функціональний інтерфейс для досягнення відображення і управління пристроєм?" було отримано відповідь. Цей розділ показав, що новий, легкий абстрактний XML може бути використаний для визначення файлів опису пристрою, які описують можливості пристрою, макет інтерфейсу користувача та способи керування пристроєм. Це перший раз, коли ці три описи були об'єднані в один файл. Розділ визначив, що статус комерційний протокол був необхідний для відображення поточного стану IoT пристроїв на декількох пристроях інтерфейсу. Вивчаючи уроки з існуючих протоколів, таких як CoAP і MQTT, були запропоновані чотири протоколи і порівняні з UPnP для ефективності пакетів і найкращого кандидата, визначеного Протоколом 3.

Дослідження, проведене для відповіді на питання 3 , створило: 3 публікації [57], [58], новий протокол зв'язку статусу, новий абстрактний опис пристрою та новий протокол відображення та управління пристроями на інтерфейсі з використанням анотації до опису нового пристрою.

Всі протоколи, реалізовані для відповіді на ці питання, були інтегровані в три довідкові бібліотеки для Android, iOS і Electron і супроводжуються еталонними реалізаціями, що демонструють підтримку Android, iOS, Windows, macOS і Linux. Ці інтегровані реалізації демонструють, що нові протоколи, розроблені в цьому дослідженні, працюють разом і усувають необхідність PCC в системі домашньої автоматизації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. D. C. Campbell, GB6, D. R. Thompson, and GB6, United States Patent: 4200862 - Appliance control, 4200862, Apr. 29, 1980.

2. Dave Rye, Dave Rye @ X10: My life at x10, Oct. 01, 1999.

3. The Source for X10 & X10 Pro Genuine Products. URL: https://www.x10.com/ (дата звернення: 23.02.2022).

4. KNX Association KNX Association . https://www.knx.org/knx-en/for-professionals/index.php (дата звернення: 04.02.2022).

5. J. Lázaro, S. Abejón, A. Astarloa, F. Chamorro, and U. Bidarte. SoPC Implementation of the TP-KNX Protocol for Domotic Application. *2008 International Conference on Advances in Electronics and Micro-electronics*, Sep. 2008, pp. 115–120, doi: 10.1109/ENICS.2008.9.

6. R. A. Ramlee, M. A. Othman, M. H. Leong, M. M. Ismail, and S. S. S. Ranjit, Smart home system using android application. *2013 International Conference of Information and Communication Technology (ICoICT)*, Mar. 2013, pp. 277–280, doi: 10.1109/ICoICT.2013.6574587.

7. N. Dickey, D. Banks, and S. Sukittanon, Home automation using Cloud Network and mobile devices. 2012 Proceedings of IEEE Southeastcon, Mar. 2012, pp. 1–4, doi: 10.1109/SECon.2012.6197003.

8. A. Panwar, A. Singh, R. Kumawat, S. Jaidka, and K. Garg, Eyrie smart home automation using Internet of Things. *2017 Computing Conference*, Jul. 2017, pp. 1368–1370, doi: 10.1109/SAI.2017.8252269.

9. P. Jutadhamakorn, T. Pillavas, V. Visoottiviseth, R. Takano, J. Haga, and D. Kobayashi, A scalable and low-cost MQTT broker clustering system. *2017 2nd International Conference on Information Technology (INCIT)*, Nov. 2017, pp. 1–5, doi: 10.1109/INCIT.2017.8257870.

10. M. S. S. Rukmini and D. B. G. Devi, Remote control of appliances based on Raspberry pi. *2016 Second International Conference on Cognitive Computing and Information Processing (CCIP)*, Aug. 2016, pp. 1–4, doi: 10.1109/CCIP.2016.7802863.

11. R. Piyare and M. Tazil, Bluetooth based home automation system using cell phone. *2011 IEEE 15th International Symposium on Consumer Electronics (ISCE)*, Jun. 2011, pp. 192–195, doi: 10.1109/ISCE.2011.5973811.

12. C. Gray and L. Campbell, Should my toaster be polled? Towards an energy-efficient Internet of Things. *2016 26th International Telecommunication Networks and Applications Conference (ITNAC),* Dec. 2016, pp. 26– 31, doi: 10.1109/ATNAC.2016.7878777.

13. F. Viani, F. Robol, A. Polo, P. Rocca, G. Oliveri, and A. Massa, Wireless Architectures for Heterogeneous Sensing in Smart Home Applications: Concepts and Real Implementation, Proceedings of the IEEE, vol. 101, no. 11, pp. 2381–2396, Nov. 2013, doi: 10.1109/JPROC.2013.2266858.

14. M. Sutiono, H. Nugroho, and K. Karyono, ApplianceHub: A wireless communication system for smart devices (case study: Smart Rice Cooker), *2016 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET),* Oct. 2016, pp. 125–130, doi: 10.1109/ICRAMET.2016.7849597.

15. R. K. Kodali and S. Soratkal, MQTT based home automation system using ESP8266. *2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*, Dec. 2016, pp. 1–5, doi: 10.1109/R10-HTC.2016.7906845.

16. F. J. Bellido-Outeirino, J. M. Flores-Arias, E. J. Palacios-Garcia, V. Pallares-Lopez, and D. Matabuena-Gomez-Limon, M2M home data interoperable management system based on MQTT. *2017 IEEE 7th International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*, Sep. 2017, pp. 200–202, doi: 10.1109/ICCE-Berlin.2017.8210627.

17. J. Prabaharan, A. Swamy, A. Sharma, K. N. Bharath, P. R. Mundra, and K. J. Mohammed, Wireless home automation and security system using MQTT protocol, *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT),* May 2017, pp. 2043–2045, doi: 10.1109/RTEICT.2017.8256958.

18. A. G. Ismaeel and M. Q. Kamal, Worldwide auto-mobi: Arduino IoT home automation system for IR devices, *2017 International Conference on Current Research in Computer Science and Information Technology (ICCIT)*, Apr. 2017, pp. 52–57, doi: 10.1109/CRCSIT.2017.7965533.

19. F. den Hartog, A. Raschella, F. Bouhafs, P. Kempker, B. Boltjes, and M. Seyedebrahimi, A pathway to solving the Wi-Fi tragedy of the commons in apartment blocks, *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, Nov. 2017, pp. 1–6, doi: 10.1109/ATNAC.2017.8215382

20. R. A. Ramlee, D. H. Z. Tang, and M. M. Ismail, Smart home system for Disabled People via Wireless Bluetooth. *2012 International Conference on System Engineering and Technology (ICSET)*, Sep. 2012, pp. 1– 4, doi: 10.1109/ICSEngT.2012.6339347.

21. S. Spinsante et al., A LoRa enabled building automation architecture based on MQTT, *2017 AEIT International Annual Conference*, Sep. 2017, pp. 1–5, doi: 10.23919/AEIT.2017.8240560.

22. W. M. Khan and I. A. Zualkernan. SensePods: A ZigBee-Based Tangible Smart Home Interface. *IEEE Transactions on Consumer Electronics*. vol. 64, no. 2, pp. 145–152, May 2018, doi: 10.1109/TCE.2018.2844729.

23. M. B. Yassein, W. Mardini, and A. Khalil, Smart homes automation using Z-wave protocol. *2016 International Conference on Engineering MIS (ICEMIS)*, Sep. 2016, pp. 1–6, doi: 10.1109/ICEMIS.2016.7745306.

24. D. Yan and Z. Dan, ZigBee-based Smart Home system design. *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*, Aug. 2010, vol. 2, pp. V2-650-V2-653, doi: 10.1109/ICACTE.2010.5579732.

25. W. S. Lee and S. H. Hong, Implementation of a KNX-ZigBee gateway for home automation. *2009 IEEE 13th International Symposium on Consumer Electronics*, May 2009, pp. 545–549, doi: 10.1109/ISCE.2009.5156866.

26. N. Dou, Y. Mei, Z. Yanjuan, and Z. Yan, The Networking Technology within Smart Home System - ZigBee Technology, *International Forum on Computer*

Science-Technology and Applications, 2009. IFCSTA ’09, Dec. 2009, vol. 2, pp. 29–33, doi: 10.1109/IFCSTA.2009.129.

27. I. Krishna and K. Lavanya, Intelligent Home Automation System using BitVoicer. *2017 11th International Conference on Intelligent Systems and Control (ISCO)*, Jan. 2017, pp. 14–20, doi: 10.1109/ISCO.2017.7855973.

28. “Meethue,” Hue, 2019. URL: https://www2.meethue.com/en-au (дата звернення: 11.03.2022).

29. A. Dellinger, Social robot Jibo does one last dance before its servers shut down, Engadget, 2019. URL: https://www.engadget.com/2019/03/04/social-robot-jibo-shutting-down-message/ (дата звернення: 13.02.2022).

30. S. Gallagher, IoT garage door opener maker bricks customer’s product after bad review, Ars Technica, Apr. 04, 2017. URL: https://arstechnica.com/information-technology/2017/04/iot-garage-door-opener-maker-bricks-customers-product-after-bad-review/ (дата звернення: 25.01.2022).

31. V. Miori, L. Tarrini, M. Manca, and G. Tolomei, An open standard solution for domotic interoperability, IEEE Transactions on Consumer Electronics, vol. 52, no. 1, pp. 97–103, Feb. 2006, doi: 10.1109/TCE.2006.1605032.

32. K. M. Lee, W. G. Teng, and T. W. Hou, Point-n-Press: An Intelligent Universal Remote Control System for Home Appliances, IEEE Transactions on Automation Science and Engineering, vol. PP, no. 99, pp. 1–10, 2016, doi: 10.1109/TASE.2016.2539381.

33. T. B. Lee, Amazon admits that employees review ‘small sample of Alexa audio. Ars Technica, Apr. 11, 2019. URL: https://arstechnica.com/tech-policy/2019/04/amazon-admits-that-employees-review-small-sample-of-alexa-audio/ (дата звернення: 20.02.2022).

34. T. N. E. Steane and P. J. Radcliffe, An enhanced implementation of a novel IoT joining protocol. *2016 26th International Telecommunication Networks and Applications Conference (ITNAC)*, Dec. 2016, pp. 22–25, doi: 10.1109/ATNAC.2016.7878776.

35. T. N. E. Steane and P. J. Radcliffe, A universal iot joining protocol for DIY applications. *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*, Nov. 2017, pp. 1–3, doi: 10.1109/ATNAC.2017.8215360.

36. Wifi Protected Setup vulnerable, Computer Fraud & Security, vol. 2012, no. 1, p. 3, Jan. 2012, doi: 10.1016/S1361-3723(12)70004-0.

37. D. E. Goncharov, S. V. Zareshin, R. V. Bulychev, and D. S. Silnov, Vulnerability analysis of the Wifi spots using WPS by modified scanner vistumbler, *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, Jan. 2018, pp. 48–51, doi: 10.1109/EIConRus.2018.8317027.

38. L. Chen, G. Pan, and S. Li, Touch-driven interaction via an NFC-enabled smartphone. *2012 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, Mar. 2012, pp. 504–506, doi: 10.1109/PerComW.2012.6197548.

39. S. Kumar and S. R. Lee, Android based smart home system with control via Bluetooth and internet connectivity. The 18th IEEE International Symposium on Consumer Electronics (ISCE 2014), Jun. 2014, pp. 1–2, doi: 10.1109/ISCE.2014.6884302.

40. C. Withanage, R. Ashok, C. Yuen, and K. Otto, "A comparison of the popular home automation technologies. 2014 IEEE Innovative Smart Grid Technologies - Asia (ISGT ASIA), May 2014, pp. 600–605, doi: 10.1109/ISGT-Asia.2014.6873860.

41. C. Gomez and J. Paradells, Wireless home automation networks: A survey of architectures and technologies. IEEE Communications Magazine, vol. 48, no. 6, pp. 92–101, Jun. 2010, doi: 10.1109/MCOM.2010.5473869.

42. M. A. Zamora-Izquierdo, J. Santa, and A. F. Gomez-Skarmeta, An Integral and Networked Home Automation Solution for Indoor Ambient Intelligence. IEEE Pervasive Computing, vol. 9, no. 4, pp. 66–77, Oct. 2010, doi: 10.1109/MPRV.2010.20.

43. Operating panel for KNX home automation SP 5.1 KNX | Jung | SP 5.1 KNX | Alarm control panel eibabo.com. URL: https://www.eibabo.com/en/jung/operating-panel-for-knx-home-automation-sp-5.1-knx-eb12901615?utm_source=Portals&utm_medium=CPC&utm_campaign=eibabo-

COM_GoogleShopping_AU&gclid=Cj0KCQiA5bz-BRD-ARIsABjT4nge_KKG5ySLmkSA4ZKTtPuTXg62OM8l0JhfEgOlfMY0EyZd3MkggLkaAh31EALw_wcB (дата звернення: 01.02.2022).

44. S. Nasrin and P. J. Radcliffe, Novel protocol enables DIY home automation. *Telecommunication Networks and Applications Conference (ATNAC)*, 2014 Australasian, Nov. 2014, pp. 212–216, doi: 10.1109/ATNAC.2014.7020900.

45. UPnP Forum, "UPnP Device Architecture 2.0," 2015. URL: http://upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v2.0.pdf.

46. H. Zhang, F.-Y. Wang, and Y. Ai, An OSGi and agent based control system architecture for smart home. *Proceedings. 2005 IEEE Networking, Sensing and Control*, 2005., Mar. 2005, pp. 13–18, doi: 10.1109/ICNSC.2005.1461152.

47. C. W. Hsu, S. T. Cheng, and C. F. Chen, Widget-based framework for web service discovery on multiple home social network, *2011 IEEE International Conference on Granular Computing*, Nov. 2011, pp. 250–255, doi: 10.1109/GRC.2011.6122603.

48. J. Zhang, Z. Wang, Z. Yang, and Q. Zhang, Proximity based IoT device authentication, *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, May 2017, pp. 1–9, doi: 10.1109/INFOCOM.2017.8057145.

49. S. M. Kim, H. S. Choi, and W. S. Rhee, IoT home gateway for auto-configuration and management of MQTT devices. *2015 IEEE Conference on Wireless Sensors (ICWiSe)*, Aug. 2015, pp. 12–17, doi: 10.1109/ICWISE.2015.7380346.

50. A. Kamilaris, A. Pitsillides, and V. Trifa, The Smart Home Meets the Web of Things, Int. J. Ad Hoc Ubiquitous Comput., vol. 7, no. 3, pp. 145–154, May 2011, doi: 10.1504/IJAHUC.2011.040115.

51. S. K. Datta and C. Bonnet, Connect and Control Things: Integrating Lightweight IoT Framework into a Mobile Application. *2015 9th International Conference on Next Generation Mobile Applications*, Services and Technologies, Sep. 2015, pp. 66–71, doi: 10.1109/NGMAST.2015.23.

52. S. K. Datta, C. Bonnet, A. Gyrard, R. P. F. da Costa, and K. Boudaoud, "Applying Internet of Things for personalized healthcare in smart homes. *2015 24th*

Wireless and Optical Communication Conference (WOCC), Oct. 2015, pp. 164–169, doi: 10.1109/WOCC.2015.7346198.

53. Z. Shelby, K. Hartke, and C. Bormann, The Constrained Application Protocol (CoAP). URL: https://tools.ietf.org/html/rfc7252 (accessed Jul. 28, 2016).

54. M. Alaa, A. A. Zaidan, B. B. Zaidan, M. Talal, and M. L. M. Kiah, A review of smart home applications based on Internet of Things, Journal of Network and Computer Applications, vol. 97, pp. 48–65, Nov. 2017, doi: 10.1016/j.jnca.2017.08.017.

55. D. F. S. Santos, H. O. Almeida, and A. Perkusich, A personal connected health system for the Internet of Things based on the Constrained Application Protocol, Computers & Electrical Engineering, vol. 44, pp. 122–136, May 2015, doi: 10.1016/j.compeleceng.2015.02.020.

56. M. Castro, A. J. Jara, and A. F. Skarmeta, Enabling end-to-end CoAP-based communications for the Web of Things, Journal of Network and Computer Applications, vol. 59, pp. 230–236, Jan. 2016, doi: 10.1016/j.jnca.2014.09.019.

57. T. N. E. Steane and P. Radcliffe, Multiple Intermittent Controllers for IoT Home Automation. *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*, Nov. 2018, pp. 1–6, doi: 10.1109/ATNAC.2018.8615433.

58. T. N. E. Steane and P. J. Radcliffe, IoT status communication for home automation, International Journal of Computing, vol. 18, no. 3, pp. 240–248, Jan. 2019.

59. MQTT Version 3.1.1. URL: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html (дата звернення: 03.03.2022).

60. L. Mainetti, V. Mighali, and L. Patrono, An android multi-protocol application for heterogeneous building automation systems. *2014 22nd International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, Sep. 2014, pp. 121–127, doi: 10.1109/SOFTCOM.2014.7039071.

61. O. Bergmann, K. T. Hillmann, and S. Gerdes, A CoAP-gateway for smart homes. *2012 International Conference on Computing, Networking and Communications (ICNC)*, Jan. 2012, pp. 446–450, doi: 10.1109/ICCNC.2012.6167461.

62. K. Hartke, Observing Resources in the Constrained Application Protocol (CoAP). URL: https://tools.ietf.org/html/rfc7641#section-5 (accessed Aug. 03, 2018).

63. J. Joshi et al., Performance enhancement and IoT based monitoring for smart home. *2017 International Conference on Information Networking (ICOIN)*, Jan. 2017, pp. 468–473, doi: 10.1109/ICOIN.2017.7899537.

64. "<uses-sdk> | Android Developers." URL: https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#uses (дата звернення: 05.02.2022).

65. "IDC: Smartphone OS Market Share," URL: http://www.idc.com/promo/smartphone-market-share/os (дата звернення: 29.01.2022).

66. H. König, Protocol Engineering. Springer Science & Business Media, 2012.

67. W. W. Royce, Managing the development of large software systems: concepts and techniques. Proceedings of the 9th international conference on Software Engineering, 1970, pp. 328–338, URL: http://leadinganswers.typepad.com/leading_answers/files/original_waterfall_paper_winston_royce.pdf. (дата звернення: 17.02.2022)

68. J. Martin, Rapid Application Development. Macmillan Publishing Company, 1991.

69. B. W. Boehm, A spiral model of software development and enhancement. Computer, vol. 21, no. 5, pp. 61–72, May 1988, doi: 10.1109/2.59.

70. B. W. Boehm, Verifying and Validating Software Requirements and Design Specifications, IEEE Software; Los Alamitos, vol. 1, no. 1, pp. 75–88, Feb. 1984, doi: 10.1109/MS.1984.233702.

71. K. Beck, Extreme Programming Explained: Embrace Change. Addison-Wesley, 2000.

72. N. Kushik, J. López, A. Cavalli, and N. Yevtushenko, Improving Protocol Passive Testing through 'Gedanken' Experiments with Finite State Machines, *2016 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, Aug. 2016, pp. 315–322, doi: 10.1109/QRS.2016.43.

73. O. Tarasyuk, A. Gorbenko, V. Kharchenko, and T. Hollstein, Contention window adaptation to ensure airtime consumption fairness in multirate Wi-Fi networks, *The 10th International Conference on Digital Technologies 2014*, Jul. 2014, pp. 344–349, doi: 10.1109/DT.2014.6868737.

74. A. Gorbenko, A. Romanovsky, V. Kharchenko, and O. Tarasyuk, Dependability of Service-Oriented Computing: Time-Probabilistic Failure Modelling, Software Engineering for Resilient Systems, 2012, pp. 121–133.

75. Tiny Core Linux, Micro Core Linux, 12MB Linux GUI Desktop, Live, Frugal, Extendable. URL: https://distro.ibiblio.org/tinycorelinux/welcome.html (дата звернення: 07.03.2022).

76. T. N. E. Steane, tylersteane / dhap-iot-status-communication-for-home-automation — Bitbucket, DHAP - IoT Status Communication for Home Automation. URL: https://bitbucket.org/tylersteane/dhap-iot-status-communication-for-home-automation/src/master/ (дата звернення: 20.02.2022).

77. W. O. Galitz, The Essential Guide to User Interface Design: An Introduction to Guidesign Principles and Techniques. Hoboken, UNITED STATES: John Wiley & Sons, Incorporated, 2007.

78. T. N. E. Steane, A. Garipoli, D. Milner, and P. J. Radcliffe, Dynamic Device Display, Jul. 25, 2019. URL: https://github.com/TylerSteane/dynamic-device-display. (дата звернення: 23.01.2022)

79. Зелінський О. О., Савенко Б. О., Каштальян А. С. Розподілена нейромережна система виявлення комп’ютерних атак на основі шаблонів атак. УДК 004.49

ДОДАТОК А
(обов'язковий)

ЛІСТИНГ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПРОТОКОЛУ ДЛЯ ЦЕНТРАЛЬНОГО КОНТРОЛЕРА В КІБЕРФІЗИЧНІЙ СИСТЕМІ «РОЗУМНИЙ БУДИНОК»

```
#include "BedroomLamp.hpp"

const unsigned int BedroomLamp::RC_SWITCH_PROTOCOL = 1;
const unsigned int BedroomLamp::RC_SWITCH_LENGTH = 24;
// Intercepted codes
const unsigned long BedroomLamp::TURN_ON_CODE = 180356;
const unsigned long BedroomLamp::TURN_OFF_CODE = 180353;

BedroomLamp::BedroomLamp(uint8_t transmitterPin) : Lamp(transmitterPin) {
    Lamp::switchProtocol(RC_SWITCH_PROTOCOL);
}

void BedroomLamp::changeState(bool shouldTurnOn) {
    if (shouldTurnOn) {
        Lamp::turnOn(TURN_ON_CODE, RC_SWITCH_LENGTH);
    } else {
        Lamp::turnOff(TURN_OFF_CODE, RC_SWITCH_LENGTH);
    }
}
#include "EntryPoint.hpp"
#include "BedroomLamp.hpp"
#include "MqttClient.hpp"
#include "OtaServer.hpp"
#include "WifiClient.hpp"

MqttConfig *mqttConfig;
MqttClient *mqttClient;
OtaServer *otaServer;
WifiClient *wifiClient;
BedroomLamp *bedroomLamp;

void setup() {
    Serial.begin(BAUD_RATE);
    initNetwork();
    initSensors();
}

void initNetwork() {
    wifiClient = new WifiClient(DEVICE_IP, MASK, GATEWAY, WIFI_SSID, WIFI_PASSWORD);
    otaServer = new OtaServer();
    mqttConfig = new MqttConfig(MQTT_BROKER_IP, MQTT_BROKER_PORT, MQTT_USERNAME, MQTT_PASS
WORD, mqttCallback);
    mqttClient = new MqttClient(mqttConfig, wifiClient);
}
```

```
void initSensors() {
    bedroomLamp = new BedroomLamp(RF_TRANSMITTER_PIN);
}

void mqttCallback(const char *topic, const byte *payload, unsigned int length) {
    // mqtt payload processing
    // bool value = …
    bedroomLamp->changeState(value);
}

void loop() {
    otaServer->listen();

    if (!mqttClient->isConnected()) {
        mqttClient->connect([] { mqttClient->subscribe(Topic::DEVICE.c_str()); });
    }

    mqttClient->keepAlive();
}
```

```
@Injectable()
export class ASRService {
  @Inject()
  private readonly kaldiSocketService: KaldiSocketService
  @Inject(WINSTON)
  private readonly logger: Logger
  private readonly detector: HotwordDetector

  constructor() {
    this.detector = new HotwordDetector(DETECTOR_OPTIONS, [HOT_WORD_MODEL_OPTIONS], RECORD
ER_DATA, console)
      .on('error', err => this.logger.error(err))
      .on('hotword', (index, hotword, buffer) => {
        this.logger.info('[WAKE WORD DETECTED] %s', hotword)
        this.detector.stop()
        this.kaldiSocketService.startRecognition(this.start)
      })
  }

  public start = () => {
    this.detector.start()
  }
}

@Injectable()
export class KaldiSocketService {
  @Inject(WINSTON)
  private readonly logger: Logger
  private readonly recorder: AudioRecorder = new AudioRecorder(RECORDER_OPTIONS, console)
  private socketClient: WebSocket
  private pauseTimeout: NodeJS.Timeout | null = null

  public connect(): void {
```

```
  if (this.socketClient) {
    this.socketClient.removeAllListeners()
  }

  this.socketClient = new WebSocket(KALDI_SERVER_URL)
    .on('open', () => {
      this.logger.info('Connected to Kaldi server')
    })
    .on('close', () => {
      this.logger.info('Disconnected from Kaldi server')
    })
    .on('message', (data: string) => this.handleMessage(data))
}

public startRecognition(restartFn: () => void) {
  if (!this.isSocketAlive()) {
    this.connect()
  }

  if (this.recorder) {
    this.recorder.stop()
  }

  this.recorder
    .start()
    .stream()
    .on('error', err => this.logger.error(err))
    .on('data', chunk => this.sendChunk(chunk))
    .on('end', () => restartFn())
  this.startPauseTimer(this.stopRecognition)
}

public stopRecognition = (): void => {
  this.recorder.stop()
}

public sendChunk = (chunk: any): void => {
  this.socketClient.send(chunk)
}

public disconnect = () => {
  this.sendChunk('{"eof" : 1}')
  this.stopPauseTimer()
}

private async handleMessage(data: string): Promise<void> {
  const kaldiPayload: KaldiPayload = JSON.parse(data)

  if (kaldiPayload.result && kaldiPayload.result.length > 0) {
    this.logger.info('[FINAL TRANSCRIBE] %s', kaldiPayload.text)
    const match: BestMatch = compare(kaldiPayload.text, AVAILABLE_COMMANDS)
    const command: string = match.bestMatch.target
```

```
      this.logger.info('[MATCHING COMMAND] %s -
> %s', kaldiPayload.text, command, match.bestMatch.rating)

      if (match.bestMatch.rating > SIMILARITY_LEVEL && POWER_COMMANDS.includes(command)) {
        // publish MQTT message
        this.startPauseTimer(this.stopRecognition)
      } else if (command === STOP_WORD) {
        this.disconnect()
      }
    } else if (kaldiPayload.partial) {
      this.logger.info('[PARTIAL TRANSCRIBE] %s', kaldiPayload.partial)
      this.stopPauseTimer()
    }
  }

  private startPauseTimer = (callback: () => void): void => {
    this.pauseTimeout = setTimeout(callback, SILENCE_TIMEOUT)
  }

  private stopPauseTimer = (): void => {
    if (this.pauseTimeout) {
      clearTimeout(this.pauseTimeout)
      this.pauseTimeout = null
    }
  }

  private isSocketAlive(): boolean {
    return this.socketClient.readyState === this.socketClient.OPEN
  }
}
```

ДОДАТОК Б
(обов'язковий)

ФАХОВІ ПУБЛІКАЦІЇ

УДК 004.49

Зелінський О. О., Савенко Б. О., Каштальян А. С.

Хмельницький національний університет

РОЗПОДІЛЕНА НЕЙРОМЕРЕЖНА СИСТЕМА ВИЯВЛЕННЯ КОМП'ЮТЕРНИХ АТАК НА ОСНОВІ ШАБЛОНІВ АТАК

Розглянуто відомі рішення щодо систем виявлення комп'ютерних атак згідно їх шаблонів атак. В результаті проведеного дослідження було виокремлено відомих рішень та визначено стратегічне рішення, яке повинно базуватись на використанні нейромережного підходу. Запропоновано для виявлення комп'ютерних атак використовувати розподілену нейромережну систему, а в якості показників атак їх шаблони.

The known solutions for computer attack detection systems according to their attack patterns are considered. As a result of the study, known solutions were identified and a strategic decision was identified, which should be based on the use of a neural network approach. It is proposed to use a distributed neural network system to detect computer attacks, and their templates as indicators of attacks.

Сучасний стан розвитку інформаційних технологій, постійне стрімке підвищенням рівня інформатизації суспільства та недосконалі системи захисту інформації в комп'ютерних системах надають широкі можливості зловмисникам. Один з їх напрямів – це здійснення комп'ютерних атак. Ефективність таких атак залишається досить високою через недосконалість методів для їх виявлення. Також, реалізація розроблених методів в системах виявлення не враховує багато архітектурних особливостей.

Метою роботи є розробка нейромережної системи виявлення комп'ютерних атак на основі шаблонів атак, що забезпечить подальший розвиток напрямку з виявлення комп'ютерних атак та дозволить інтегрувати метод виявлення згідно шаблонів атак безпосередньо в систему виявлення, щоб врахувати архітектурні особливості системи.

Своє застосування та значний розвиток в задачах кібербезпеки знаходять також методи кластеризації. Varaprasad Rao at al. [1] пропонує модифікований k-means метод кластеризації із додатковими кроками препроцесінгу та нормалізації для виявлення патернів невідомих атак. В роботі [2] розглядається застосування fuzzy f-means кластеризація для виявлення ботнетів. Weng at al. [3]

пропонує систему виявлення аномалій на основі поєднання k-means кластеризації та ієрархічної клатеризації. Значний розвиток в системах виявлення втручання отримали deep learning методи [4]. Ці методи використовуються не тільки для безпосереднього виконання задач виявлення аномалій, а також для препроцесінгу даних, що актуально для даних, які змінюються в часі. Багато робіт проведено в напрямку отримання ознак часових рядів на основі підходу глибокого навчання.

Автоенкодери є одними з найбільш придатних моделей для задачі кластеризації, оскільки не потребують розмічених даних. Отримання ознак даних, зменшення їх розмірності та кластеризація можуть бути інтегровані в один процес. Незважаючи на широке дослідження вирішення задачі кластеризації та виявлення патернів в часових рядах, в тому числі для даних мережевого трафіку в системах виявлення втручання, досліджень, які відображають аналіз даних, зібраних приманками та мережами приманок порівняно незначна кількість. Тому важливо розробляти та вдосконалювати методи аналізу трафіку зловмисної активності для подальшого використання цієї інформації для запобігання атакам. Атаки різного типу, в тому числі DoS атаки, мають характерну поведінку в часі, тому доцільно виявляти патерни атак в динаміці, що передбачає аналіз багатомірних часових рядів.

Для виявлення патернів в багатомірному часовому ряді необхідно розбити його на сегменти. Є значна кількість підходів до сегментації, починаючи від простого використання вікна, що зсувається по осі часу до більш складних та якісних алгоритмів. Для виявлення патернів в часових рядах важко обрати однозначний підхід, оскільки для різних типів даних різні алгоритми та міри подібності можуть давати різні кінцеві результати. Тому, важливо обрати з існуючих такі, які забезпечать найкраще групування вихідних даних. А також необхідно використовувати відповідне представлення часових рядів, основною властивістю якого є те, що схожі часові ряди мають схожі представлення.

Представлення часових рядів отримується з використанням нейромережевого згорткового автоенкодера, який забезпечує зменшення розмірності та виявлення ознак часового ряду, необхідних для кластеризації.

Попередня обробка часових рядів виконується на основі нейромережевого згорткового автоенкодера. Автоенкодер складається з згорткового енкодера та згорткового декодера. Згортковий енкодер складається з стеку згорткових блоків.

Згортковий блок включає згорковий шар, випрямляючий лінійний шар (relu), шар об'єднання (pooling) та шар нормалізації.

Отже, запропоновано для виявлення комп'ютерних атак використовувати розподілену нейромережну систему, а в якості показників атак їх шаблони.

Перелік посилань

1. Rao M.V., Damodaram A., Charyulu N.C.B. (2012) Algorithm for Clustering with Intrusion Detection Using Modified and Hashed K – Means Algorithms. In: Wyld D., Zizka J., Nagamalai D. (eds) Advances in Computer Science, Engineering & Applications. Advances in Intelligent Systems and Computing, vol 167. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-30111-7_70.
2. S. Lysenko, O. Pomorova, O. Savenko, A. Kryshchuk and K. Bobrovnikova DNS-based Anti-Evasion Technique for Botnets Detection. Proceedings of the 8-th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Warsaw (Poland), September 24–26, 2015. – Warsaw, 2015. – Pp. 453–458.
3. F. Weng, Q. Jiang, L. Shi and N. Wu, "An Intrusion Detection System Based on the Clustering Ensemble," 2007 International Workshop on Anti-Counterfeiting, Security and Identification (ASID), 2007, pp. 121-124, doi: 10.1109/IWASID.2007.373710.
4. E. Min, X. Guo, Q. Liu, G. Zhang, J. Gui, J. Long. A Survey of Clustering with Deep Learning: From the Perspective of Network Architecture IEEE Access, vol. 6, pp. 39501-39514, 2018, doi: 10.1109/ACCESS.2018.2855437

Дані про авторів:

ПІБ автора	Телефон	Email
Зелінський Олександр Олегович		oleksandr.zelinskyi@astarta.ua
Савенко Богдан Олегович	+380638191228	savenko_bohdan@ukr.net
Каштальян Антоніна Сергіївна	+380974307661	yantonina@ukr.net

ДОДАТОК В
(обов’язковий)
ПРЕЗЕНТАЦІЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
Кафедра комп’ютерної інженерії та інформаційних систем

Протокол для центрального контролера в кіберфізичній системі «розумний будинок»»

Виконав роботу студент КІ2м-20-1
Зелінський Олександр Олегович
Науковий керівник, д.т.н., професор
Савенко Олег Станіславович

Актуальність роботи

- Незважаючи на десятиліття інтересу до впровадження автоматизації та комп'ютерного інтелекту додому, домашня автоматизація не заповнила ринок повністю і навіть не стала спільною рисою будинків. Не було досягнуто нічого подібного до повсюдності розумних будинків, очікуваних за останні 40-50 років.
- На сьогоднішній день найбільш поширеним було використання контролера, який розміщується в центрі домашньої системи. Наприклад, у системах, що використовують X10 та KNX. Є кілька проблем з цими центральними контролерами. Вони фіксовані на місці і живляться постійно, вони можуть бути єдиною точкою невдачі для простих, але необхідних домашніх світильників, вони можуть бути дорогими, і їх часто важко переналаштувати.
- Актуальність роботи полягає в розробці протоколу для центрального контролера в кіберфізичній системі «розумний будинок».

Мета, об’єкт, предмет.

Метою дипломної роботи є проєктування протоколу для центрального контролера в кіберфізичній системі «розумний будинок».

Об’єктом дослідження є процес керування за протоколом для центрального контролера в кіберфізичній системі «розумний будинок».

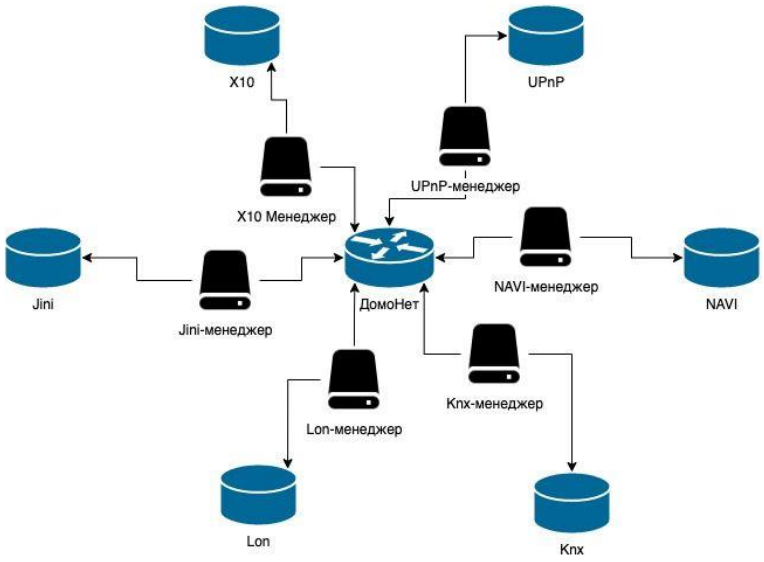
Предметом дослідження є методи проєктування протоколів для центрального контролера в кіберфізичній системі «розумний будинок».

Наукова новизна отриманих результатів: розроблено метод проєктування протоколів для центрального контролера в кіберфізичній системі «розумний будинок».

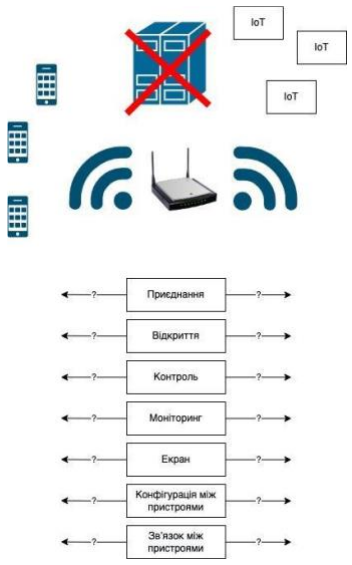
Практичне значення отриманих результатів та постановка задачі

- Практична значимість отриманих результатів полягає у розробленому протоколі для центрального контролера в кіберфізичній системі «розумний будинок».
- Для розв’язання поставлених задач використовуються основні положення методи теорії комп’ютерних мереж, архітектури комп’ютерів, теорії множин.

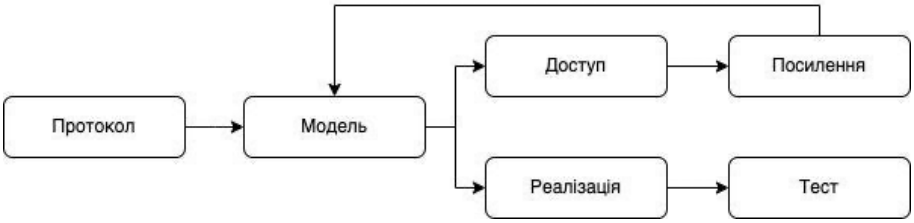
Топологія DomoNet Miori та Russo для взаємодії в домашній автоматизації



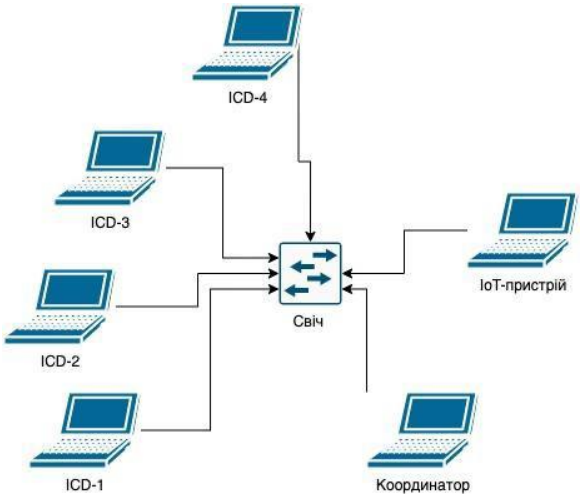
Перерозподіл обов'язків постійного центрального контролера



Методологія розробки надійної мережі



Налаштування тестового стенду з ICD 1-4, підключеним до пристрою IoT і координатора експерименту через комутатор



Стандартне відхилення та середнє значення часу між пакетом оновлення в (µs)

	STD(µs)	Середнє значення(µs)
Протокол 2	189.08	5000253.07
Протокол 3	420.14	5000279.08
Протокол 4	229.50	5000997.98

Висновок

- Це дослідження показало, що постійний центральний контролер (PCC) може бути видалений з систем домашньої автоматизації без шкоди для функціональності системи. Крім того, це покращує загальну функціональність Системи. Нові та нові протоколи, досліджені в цій роботі, були розроблені для зниження витрат, складності та негнучкості, а також покращення користувацького досвіду та варіантів.
- Це дослідження поставило три ключові питання про наслідки видалення PCC з системи домашньої автоматизації. Перше питання: «Безпечне приєднання без постійного центрального контролера було продемонстровано для старих версій Android. Які зміни потрібні, щоб розширити це ширше?». Було продемонстровано, що переорієнтація теоретичного протоколу приєднання, спочатку запропонованого Насріном і Редкліффом, дозволить інтелектуальним пристроям бути підключеними до мережі Wi-Fi без PCC і буде універсально сумісна з будь-яким пристроєм, який був включений Wi-Fi.
- Було представлено новий протокол виявлення пристроїв, який можна використовувати для виявлення розумних пристроїв у локальній мережі без PCC. Для розробки цього протоколу була запропонована і впроваджена нова методологія розвитку мережі Robustness (RNDM) для задоволення нішевих потреб розробки мережевих протоколів. Крім того, трирівневий метод швидкого масового програмування був розроблений разом з масштабованим випробувальним стендом, який дозволяє швидко і легко програмується. Це дало реальні результати.
- Було досліджено, що новий, легкий абстрактний XML може бути використаний для визначення файлів опису пристрою, які описують можливості пристрою, макет інтерфейсу користувача та способи керування пристроєм. Це перший раз, коли ці три описи були об'єднані в один файл. Додатково визначено, що статус “комерційний протокол” був необхідний для відображення поточного стану IoT пристроїв на декількох пристроях інтерфейсу.
- Всі протоколи було реалізовано, були інтегровані в три довідкові бібліотеки для Android, iOS і Electron і супроводжуються еталонними реалізаціями, що демонструють підтримку Android, iOS, Windows, macOS і Linux. Ці інтегровані реалізації демонструють, що нові протоколи, розроблені в цьому дослідженні, працюють разом і усувають необхідність PCC в системі домашньої автоматизації.

Дякую за увагу !

Tue May 03 14:41:04 EEST 2022, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальное совпадение с одним документом 0.0%

Словари проверки: en_US, ru_RU, ua_UA. **Ошибок в документах: 6%**

ID: 103248 Название: Протокол для центрального контролера в кіберфізичній системі "розумний будинок" Добавлено в БД: 2022-05-03 Авторы: Зелінський О.О. Руководители: Савенко О.С. Консультанты: Опоненты:	Документ		Суммарное совпадение по Базе Данных	
	Символы	Лексемы	Символы	Лексемы
	126305	928	568 (0%)	7 (1%)

Источник плагиата

ID	Описание	Наличие плагиата в документе	
		Символы	Лексемы

UNICHECK

Ім'я користувача:
Кафедра КІ

ID перевірки:
1011040663

Дата перевірки:
03.05.2022 15:31:47 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
03.05.2022 15:32:14 EEST

ID користувача:
100005591

Назва документа: Зелінський_Протокол для центрального контролера в кіберфізичній системі «розумний бу...

Кількість сторінок: 77 Кількість слів: 18072 Кількість символів: 136535 Розмір файлу: 335.50 KB ID файлу: 1010942022

0.68%
Схожість

Найбільша схожість: 0.6% з джерелом з Бібліотеки (ID файлу: 1010883069)

0.1% Джерела з Інтернету 2 ..... Сторінка 79

0.68% Джерела з Бібліотеки 95 ..... Сторінка 79

0.94% Цитат

Цитати 5 ..... Сторінка 80

Не знайдено жодних посилань

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 4

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА ДИПЛОМНУ РОБОТУ

Дипломник: Зелінський Олександр Олегович

Тема: Протокол для центрального контролера в кіберфізичній системі «розумний будинок»

Спеціальність: 123 «Комп`ютерна інженерія»

Обсяг дипломної роботи:

Кількість листів креслень – ; кількість сторінок записки 93

1. Короткий зміст роботи та прийнятих рішень У роботі запропоновано протокол для центрального контролера в кіберфізичній системі «розумний будинок».

2. Висновок про відповідність роботи дипломному завданню
Дипломна робота відповідає виданому завданню

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі проведено аналіз відомих методів. Досліджено відомі рішення та засоби в цій сфері. У другому розділі розглянуто особливості проєктування архітектури системи та протоколу взаємодії компонент. У третьому розділі проведена реалізація кроків протоколу. У четвертому розділі розроблено протокол та представлено його технічне рішення.

4. Позитивні сторони роботи: Запропоновані протокол для центрального контролера в кіберфізичній системі «розумний будинок», що дозволяє його застосування на практиці.

5. Негативні сторони роботи: В дослідженні було б доцільно провести експерименти для представленого протоколу для центрального контролера в кіберфізичній системі «розумний будинок».

6. Оцінка графічного оформлення та пояснювальної записки роботи: –

7. Відгук про роботу в цілому: В загальному робота виконана на достатньому рівні.

8. Інші зауваження: –

9. Оцінка дипломної роботи:

Розглянувши позитивні та негативні сторони представленої дипломної роботи вважаю, що робота заслуговує оцінки «добре» 4,5 (B)

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) Бездашнюк Леонід Петрович, зав. каф. ІПЗ, ХНУ

"___" _____ 2022р.

Завідувачу кафедри КІСП
д-р.техн.наук, проф. Говорущенко Т. О.

Зелінський Олександр Олегович
ПІБ здобувача вищої освіти

ФІТ, 2 курсу, групи КІ2м-20-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіатоповіщений (а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

25.04.2022
дата

підпис

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Протокол для центрального контролера в кіберфізичній системі «розумний будинок

Автор: Зелінський Олександр Олегович

Спеціальність: 123 – Компютерна інженерія

Освітня програма: освітньо-наукова

Науковий керівник: Савенко Олег Станіславович

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які *описують безпосередньо авторське дослідження, а кількість цитат перевищує* обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) окремі виявлені збіги є загальновживаними фразами, про що свідчить посилання системи на збіг з 10-15 джерелами на один фрагмент речення;

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 0,68% і адресується до 40 першоджерела, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ О. С. Савенко

Гарант ОП _____ О. С. Савенко

Завідувач кафедри КІСП _____ Т. *О.* Говорущенко