МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

першого (бакалаврського) рівня вищої освіти

на тему: **«Розробка мобільного додатку для керування «Розумним чайником» із використанням платформи Samsung SmartThings»**

Виконав: студент 2 курсу групи Іт-22сп

Спеціальності 126 «Інформаційні системи та технології»
(шифр і назва)
Цвик Назар Володимирович
(Прізвище та ініціали)

Керівник: к.т.н., в.о. доц. Татомир А.В.
(Прізвище та ініціали)

Рецензент: к.т.н., доцент Кригуль Р.Є.
(Прізвище та ініціали)

ДУБЛЯНИ-2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Перший (бакалаврський) рівень вищої освіти
Спеціальність 126 «Інформаційні системи та технології»

«ЗАТВЕРДЖУЮ»

Завідувач кафедри_____

д.т.н., проф. А. М. Тригуба
«____» _____2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу студенту

Цвику Назару Володимировичу

1. Тема роботи: «Розробка мобільного додатку для керування «Розумним чайником» із використанням платформи Samsung SmartThings»

Керівник роботи Татомир Андрій Володимирович, в.о. доцента
затверджені наказом по університету від 30.12.2022 року № 453/к-с.

2. Строк подання студентом роботи 10.06.2023 р.
3. Вихідні дані до роботи: *вимоги до мобільного додатку для керування «Розумним чайником»; методика проектування мобільного додатку; платформи для розробки мобільних додатків.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)
Вступ.
1. Аналіз предметної області.
2. Вибір інструментарію для реалізації проекту.
3. Розробка мобільного додатку для керування «Розумним чайником».
4. Практичне використання мобільного додатку.
5. Охорона праці.
Висновки та пропозиції.
Список використаної літератури.

5. Перелік ілюстраційного матеріалу (з точним зазначенням обов'язкових креслень): *особливості функціонування мобільного додатку для керування «Розумним чайником»; огляд аналогів мобільних додатків; результати вибору засобів реалізації проекту; планування робіт у проекті; результати розробки мобільного додатку для керування «Розумним чайником»; особливості реалізації та практичного використання мобільного додатку.*

6. Консультанти з розділів:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1, 2, 3, 4	*Татомир А.В., в.о. доцента кафедри ІТ*		
5	*Городецький І.М., доцент кафедри управління проектами та безпеки виробництва*		

7. Дата видачі завдання 30 грудня 2022 р.

Календарний план

№ з/п	Назва етапів кваліфікаційної роботи	Терміни виконання етапів роботи	Примітка
1	*Написання першого розділу*	*01.01-05.02.23*	
2	*Виконання другого розділу та аркушів ілюстраційного матеріалу до нього*	*06.02-05.03.23*	
3.	*Виконання третього розділу та аркушів ілюстраційного матеріалу до нього*	*06.03-15.04.23*	
4.	*Виконання четвертого розділу та аркушів ілюстраційного матеріалу до нього*	*16.04-12.05.23*	
5.	*Написання розділу «Охорона праці»*	*13.05-23.05.23*	
6.	*Завершення оформлення розрахунково-пояснювальної записки та аркушів ілюстраційного матеріалу*	*24-31.05.23*	
7.	*Завершення роботи вцілому*	*01 -10.06.23*	

Студент _____ Цвик Н.В.
(підпис)

Керівник роботи _____ Татомир А.В.
(підпис)

УДК 004.75

Розробка мобільного додатку для керування «Розумним чайником» із використанням платформи Samsung SmartThings.

Цвик Н.В. Кафедра ІТ – Дубляни, Львівський НАУ, 2023.

Кваліфікаційна робота: 61с. текст. част., 23 рис., 4 табл., 12 арк. ілюстраційного матеріалу, 20 джерел.

Подано особливості керування розумними пристроями із використанням платформи Samsung Smartthings. Виконано аналіз існуючих некомерційних систем «Розумним Дім». Проведено аналіз існуючих мобільних додатків для керування розумними пристроями. Здійснена ідентифікація ідеї проекту розробки мобільного додатку.

Подана архітектура системи «Розумний Чайник». Здійснено вибір пристроїв та їх основні функції. Проведена реєстрація створеного пристрою.

Подано основні етапи створення мобільного додатку для керування «Розумним Чайником». Здійснено розробку мобільного додатку. Проведена розробка модуля ініціалізації пристрою для підключення. Створено модуль для перевірки зовнішній подій. Здійснено додавання до профілю пристрою та використання кастомних Capabilities.

Розроблено заходи з охорони праці під час розробки мобільного додатку для керування «Розумним чайником».

ЗМІСТ

ВСТУП

У наш час цифрових технологій, «розумні» пристрої стають все більш поширеними і необхідними. Вони спрощують наше життя, дозволяючи керувати різними аспектами наших домівок із зручних мобільних додатків. Один з цих «розумних» пристроїв – «розумний чайник». Він може автоматизувати процес приготування чаю або кави, налаштовувати температуру води та багато іншого [10].

Кваліфікаційна робота «Розробка мобільного додатку для керування «Розумним чайником» із використанням платформи Samsung SmartThings» має на меті розробити додаток, який буде інтегрований із Samsung SmartThings, що являє собою платформу, яка дозволяє об'єднати весь ваш «розумний» дім під одним «дахом». Цей додаток не тільки забезпечить зручне керування «розумним чайником», але й дозволить його синхронізувати з іншими пристроями у вашому домі через платформу SmartThings.

На цей момент, існує велика потреба в таких рішеннях, що забезпечують комфорт і зручність для кінцевих користувачів. Використання технологій «Інтернету речей» (IoT) та штучного інтелекту (AI) в додатку дозволить максимально ефективно використовувати потенціал «розумного чайника».

У цій роботі пропонується розробити мобільний додаток для платформ Android і iOS, який буде мати інтуїтивно зрозумілий інтерфейс та гнучкі налаштування для максимального задоволення потреб користувачів. Це новий крок у впровадженні інноваційних технологій в повсякденне життя, що сприятиме більшій автоматизації домашніх процесів та підвищенню якості життя.

Наш проект «Розробка мобільного додатку для керування «Розумним чайником» із використанням платформи Samsung SmartThings» спрямований на створення інноваційного рішення, яке зробить наші домівки ще більш комфортними, автоматизованими та «розумними». Зазначений проект має

практичне скерування та дасть можливість зробити значний внесок у сферу IoT і «розумного» дому.

РОЗДІЛ 1.
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Особливості керування розумними пристроями із використанням платформи Samsung SmartThings

SmartThings – це передова технологія для керування підключеними пристроями, яка наближає майбутнє Інтернету речей (IoT). Його відкрита платформа вже підтримує тисячі пристроїв від сотень брендів і відкриває безмежні можливості для іноваторів і розробників, яким необхідно безперешкодно підключатися до ширшої екосистеми IoT (рис. 1.1).

Рис. 1.1. Технологія компанії Samsung SmartThings

Компанія прагне надавати розумні функції та покращувати взаємодію з користувачами для споживачів у всьому світі. Сьогодні мільйони людей у понад 200 країнах світу щодня використовують технологію SmartThings, щоб автоматизувати та контролювати кожен аспект свого розумного дому за

допомогою програми SmartThings та ряду інших продуктів Samsung (телефони, телевізори та цифрові пристрої).

SmartThings пропонує багато найбільш гнучких протоколів, включаючи новий стандарт IoT Matter. SmartThings, заснована в 2012 році у штаті Каліфорнія компанією Samsung, яка створює, підтримує і розробляє відкриті глобальні стандарти для Інтернету речей.

Samsung SmartThings – це платформа, створена для інтеграції та керування «розумними» пристроями. Вона дозволяє користувачам взаємодіяти і контролювати розумні пристрої за допомогою мобільного додатка, що доступний на платформах Android та iOS. Наступні особливості є характерними для керування розумними пристроями за допомогою платформи Samsung SmartThings.

SmartThings дозволяє користувачам контролювати всі їх розумні пристрої з одного місця, створюючи централізований «розумний» дім. Користувачі можуть вмикати та вимикати пристрої, налаштовувати їх та отримувати сповіщення про їх стан.

SmartThings сумісна з великою кількістю розумних пристроїв різних виробників. Це означає, що користувачі можуть використовувати цю платформу для керування різними пристроями в їхньому домі, незалежно від виробника.

За допомогою функцій автоматизації в SmartThings, користувачі можуть створювати різні сценарії та рутини для їхніх пристроїв. Наприклад, можна налаштувати чайник на автоматичне включення в певний час.

SmartThings має вбудовані функції безпеки, що дозволяють користувачам моніторити та контролювати безпеку свого дому. Крім того, платформа надає гарантії захисту даних користувачів.

SmartThings сумісна з голосовими помічниками, такими як Google Assistant та Amazon Alexa. Це дозволяє користувачам керувати своїми розумними пристроями за допомогою голосових команд, що робить процес взаємодії ще більш зручним і природним.

SmartThings може інтегруватися з іншими сервісами та платформами, такими як IFTTT (If This Then That), що розширює можливості автоматизації та персоналізації.

Керування розумними пристроями можливе з будь-якої точки світу, де є доступ до інтернету. Це створює додаткову зручність і гнучкість для користувачів.

SmartThings дозволяє отримувати сповіщення про стан пристроїв та відслідковувати їх роботу в реальному часі.

Використання платформи Samsung SmartThings для керування розумним чайником надає широкий спектр можливостей для користувачів. Від простого вмикання та вимикання пристрою до створення складних сценаріїв і рутин, використання AI для аналізу використання чайника та надання рекомендацій - все це можливо завдяки цій інноваційній платформі.

1.2. Аналіз існуючих некомерційних систем «Розумним дім»

Розглядаючи та аналізуючи некомерційні проекти, можна сказати, що існує велика кількість проектів з відкритим кодом (OSP), які охоплюють більшість сценаріїв, які можуть знадобитися користувачам. Однак навіть у цьому випадку можуть виникнути ситуації, коли існуючі проекти недостатні або недостатньо продуктивні для практичного використання, але головна перевага таких проектів полягає саме в тому, що вихідний код доступний будь-якому користувачеві. Це не тільки дозволяє користувачеві знати все, що робить його «розумний» пристрій, тому йому не потрібно турбуватися про приховані операції, які виробник може приховати, але це також дозволяє користувачеві редагувати програму таким чином, щоб робити те, що бажане для користувача, тобто надає користувачеві простий спосіб розробки нових пристроїв.

Звичайно, через недосвідченість у цій сфері користувачі можуть зіткнутися з великими труднощами, але навіть найменш обізнані користувачі

можуть досягти бажаних результатів, завдяки добре розвинутим технологіям та інструментаріям, які не тільки не відштовхує, але надихає на майбутні успіхи.

Загалом у некомерційних системах можна виділити два класи обладнання:

- ✓ обладнання, побудоване за допомогою проектувальників;
- ✓ пристрої, створені з коду.

До першої категорії відноситься проект Blynk (рис. 1.2) [9].

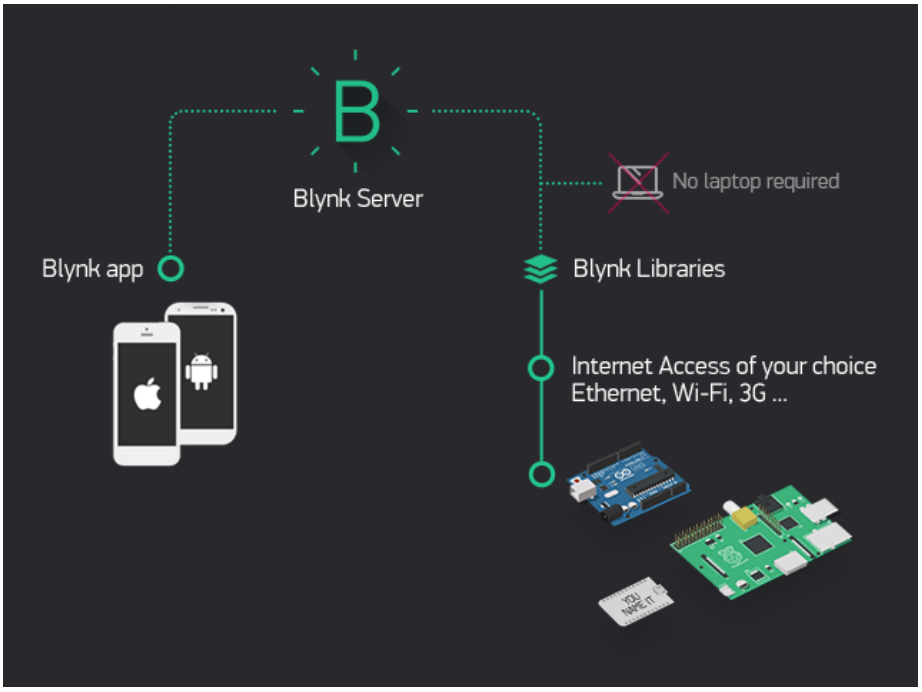

Рис. 1.2. Проект Blynk [9].

Для користувачів з нульовим досвідом програмування це має суттєву перевагу, оскільки не потребує програмування – користувач може досягти своїх цілей за допомогою програми-конструктора, яка охоплює велику кількість базових сценаріїв використання пристроїв розумного будинку.

Варто зазначити, що відсутність можливості програмування цих пристроїв також є недоліком проекту, оскільки це не тільки не дозволяє гнучко конфігурувати пристрої, але й погіршує розуміння системи користувачем. Крім того, недоліки включають створення окремих програм для кожного пристрою, що збільшує загальну кількість встановлених програм, які використовуються

для керування пристроями розумного дому, до кількості пристроїв у системі. Однак цей недолік полегшить проблему для користувачів, які вирішать просто спробувати автоматизувати невелику кількість домашніх процедур, для таких користувачів цей варіант ідеальний, оскільки він простий у використанні та недорогий, він має лише одну частину апаратного забезпечення [9].

Для другої категорії не знайдено жодного проекту, який сильно спрощує розробку цих самих пристроїв. При цьому більшість проектів мають самостійно розроблені програми для управління пристроями, тому користувачам необхідно мати мінімальний досвід програмування, щоб створювати потрібні пристрої на основі існуючих пристроїв. Звичайно, дуже рідко користувачеві потрібно внести серйозні зміни в програму, що можна пояснити використанням платформи Arduino із його програмним забезпеченням (рис. 1.3) [12].

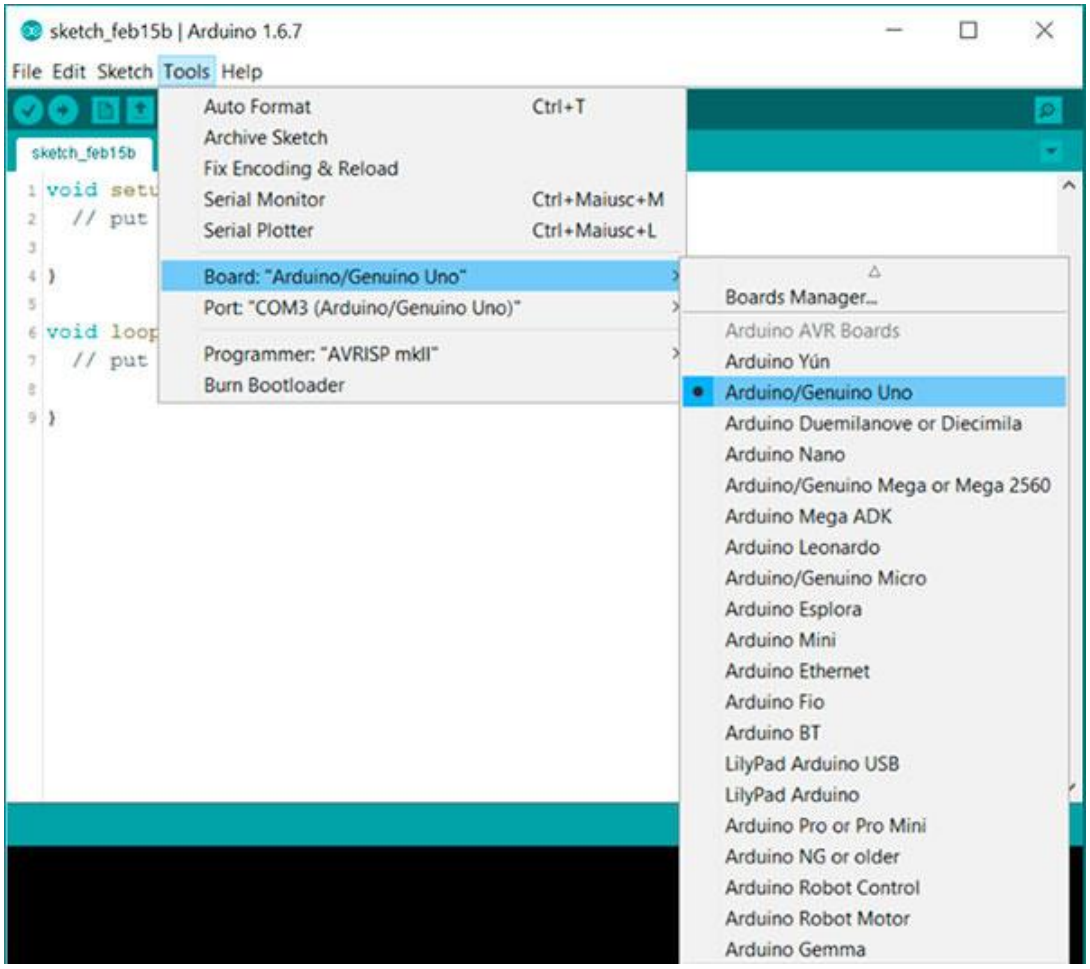

Рис. 1.3. Приклад програмним забезпеченням Arduino [12]

Це значно полегшує навантаження на нових розробників, створюючи один із найпростіших методів програмування мікроконтролерів, який породив велику спільноту Таким чином, у розробників багато проектів, що охоплюють більшість сценаріїв розумного будинку.

Існує два типи інтерфейсів керування пристроями розумного будинку:

- ✓ Веб-інтерфейс.
- ✓ Мобільні додатки.

При створенні системи, що складається лише з одного або невеликої кількості простих пристроїв, доцільно використовувати перший тип інтерфейсу [13].

Коли кількість або складність пристроїв збільшується, розробка першого типу інтерфейсу стає дуже складною, і зручність використання таких пристроїв також значно знижується. У цьому випадку мобільний додаток необхідно створити для такої популярної операційної системи, як Android. У цьому випадку підвищується не тільки зручність використання системи «розумний дім», але і гнучкість. Загалом можна виділити наступні переваги та недоліки некомерційних пристроїв системи «розумний дім».

До переваг некомерційних пристроїв системи «Розумний дім» належить:

- ✓ знизити вартість обладнання до вартості компонентів, оскільки немає комерційних компонентів;
- ✓ можливість обмінювати час розробки на будь-який рівень гнучкості;
- ✓ жодне пропрієтарне програмне забезпечення не гарантує, що користувачі можуть бути впевнені у своїй конфіденційності;
- ✓ користувачі обробляють дані безпосередньо в пристрої без використання стороннього сервера, тим самим знижуючи ризик витоку конфіденційності;
- ✓ дозволяє максимально налаштувати систему розумного дому.

До недоліків некомерційних пристроїв системи «Розумний дім» належить:

✓ підвищена складність створення системи розумний дім;

✓ збільшує час, необхідний для створення системи розумного будинку.

1.3. Аналіз існуючих мобільних додатків для керування розумними пристроями

Якщо говорити мобільні додатки для розумних пристроїв, то ними можна керувати через мобільний телефон, надаючи можливість управління пристроями на відстані. При цьому здійснюється покращення стану використання пристроїв, моніторинг стану пристроїв тощо.

Ось приблизний аналіз декількох популярних мобільних додатків у цій категорії.

Заслуговує на увагу Home Assistant (рис. 1.4).

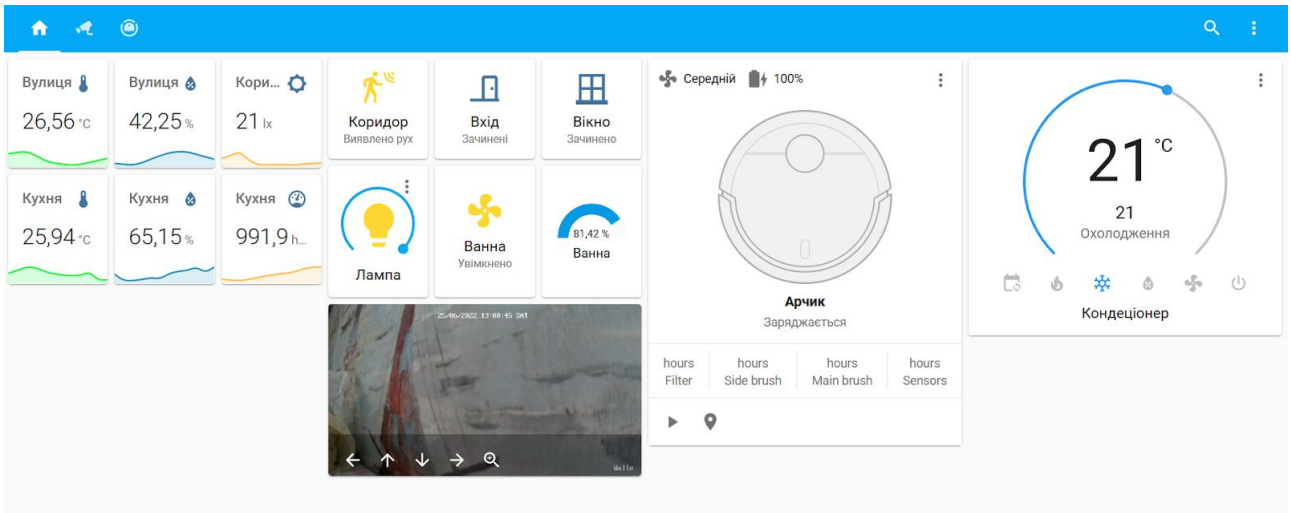

Рис. 1.4. Інформаційна панель (dashboard) Lovelace

Виробником є Home Assistant Community. Функціонал Home Assistant – дозволяє керувати багатьма розумними пристроями в домі, включаючи освітлення, термостати, камери тощо.

Призначення – керування розумними пристроями в домі.

Можливості – інтеграція з багатьма виробниками пристроїв, включаючи Google, Amazon, Philips Hue, Nest тощо. Можливість створення автоматизацій та сценаріїв. Віддалене керування.

Переваги – висока гнучкість і налаштування, велика кількість інтеграцій.

Недоліки – може бути складно для нових користувачів, потребує певного рівня технічних знань для налаштування.

Наступним мобільним додатком для керування розумними пристроями є OpenHAB (рис. 1.5).

Рис. 1.5. Інформаційна панель OpenHAB

Виробником є OpenHAB Foundation. Функціонал: OpenHAB – це мобільний додаток для автоматизації будинку та керування розумними пристроями. Він підтримує багато різних типів пристроїв і технологій.

Призначення – керування розумними пристроями в домі.

Можливості OpenHAB дозволяють керувати іншими розумними пристроями, створювати сценарії та автоматизацію, використовувати голосові помічники та віддалено керувати пристроями.

Переваги OpenHAB стосуються відкритого коду, що дозволяє користувачам і розробникам модифікувати та досягати його під свої потреби. Широка підтримка різних пристроїв і технологій.

Недоліки OpenHAB – може бути досить складним для нових користувачів, потім йому потрібен певний рівень технічних знань для його налаштування та використання.

Ці два додатки є відмінними прикладами мобільних додатків для керування некомерційними розумними пристроями. Обидва мають свої сили, але також вимагають певного рівня технічних знань для їх налаштування та використання.

Наступним розглянемо Domoticz для Android (рис. 1.6).

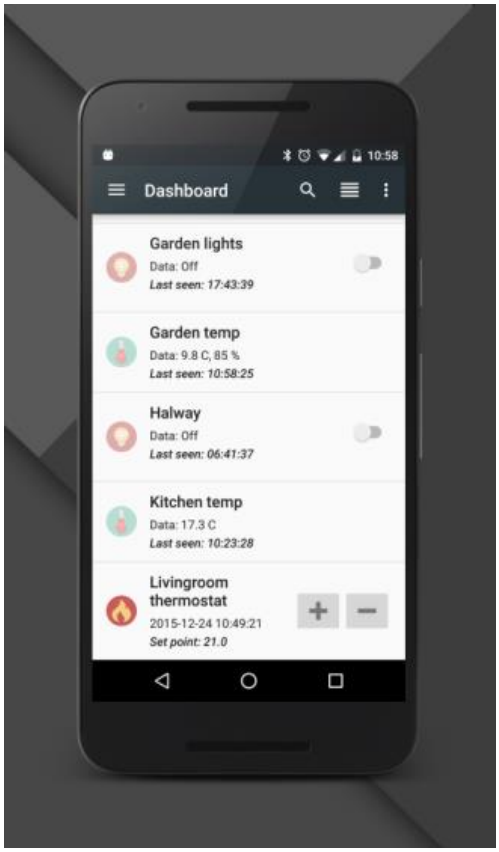

Рис. 1.6. Загальний вигляд вікна користувача Domoticz для Android

Виробником є компанія Domoticz.

Domoticz – це додаток для автоматизації будинку, який дозволяє контролювати різні розумні пристрої, включаючи освітлення, термостати, вимикачі, датчики і т.д.

Призначення – керування розумними пристроями в домі.

Domoticz підтримує багато протоколів і пристроїв, включаючи Z-Wave, Zigbee, MQTT, RFXCOM, 1-Wire та багато інших. Користувачі можуть створювати автоматизовані сценарії та сповіщення.

Переваги Domoticz – дуже функціональний і має великі можливості для налаштування. Через підтримку великої кількості протоколів він може інтегруватися з великою кількістю розумних пристроїв.

Недоліки – хоча Domoticz має багато функцій і можливостей, його інтерфейс може бути дещо складним для нових користувачів. Він також вимагає певного рівня технічних знань для налаштування.

Наступним розглянемо Node-RED (рис. 1.7).

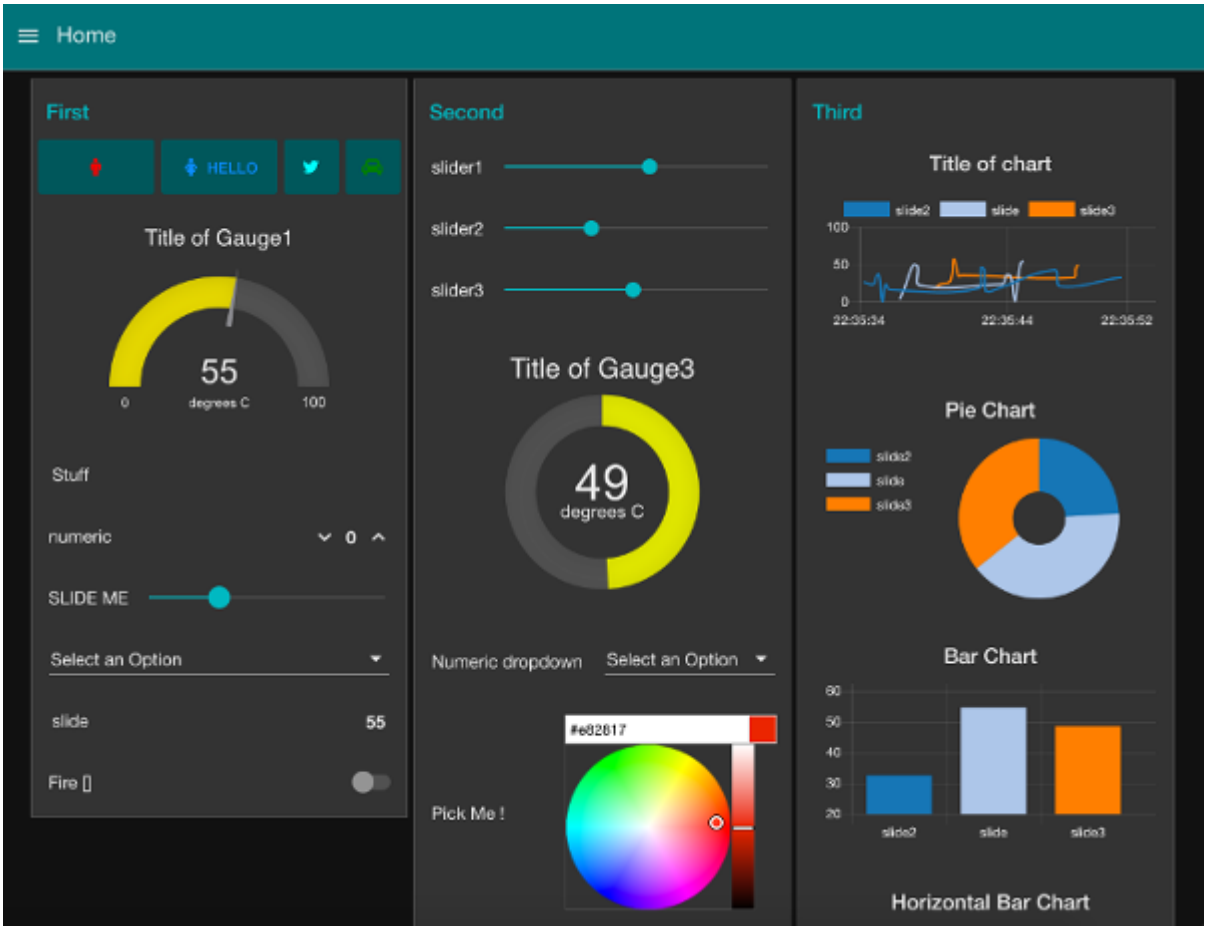

Рис. 1.7. Візуалізація Node-RED

Виробником є компанія Node-RED.

Функціонал Node-RED – це інструмент програмування для підключення апаратного та програмного забезпечення для Internet of Things (IoT). Він дозволяє створювати власні програми для автоматизації різних процесів.

Призначення – автоматизація роботи розумних пристроїв, керування ними, моніторинг.

Можливості Node-RED включають в себе багатий набір «вузлів» (програмних модулів) для взаємодії з іншими пристроями та сервісами. Це включає вузли для запитів HTTP, повідомлень MQTT, взаємодії з базами даних тощо.

Переваги Node-RED – дуже вдалий і потужний інструмент, який можна досягти під конкретні потреби користувача. Він має графічний інтерфейс для програмування, що спрощує створення програми.

Недоліки Node-RED – вимагає більше технічних знань і навичок програмування, ніж інші додатки в цьому списку. Хоча він і має графічний інтерфейс, нові користувачі можуть знайти його складним для використання.

Усі ці додатки надають широкий спектр можливостей для керування розумними пристроями. Вибір між ними залежить від ваших конкретних потреб та технічних знань.

Наступною розглянемо платформу SmartThings, яка розпочиналася як стартап. На той момент, коли вона з'явилася, в 2012 році в США, ще не було розумних пристроїв і різноманітності гаджетів у магазинах. Ідея прийшла засновнику стартапу, Алексу Хокінсону, після одного випадку, коли будинок його родини в Колорадо постраждав через те, що після тимчасового відключення електрики водопровідні труби сильно замерзли, їх прорвало, і весь поверх залило. Тоді він подумав, що міг би запобігти проблемі, якби знав про те, що відбувається в будинку.

Після безуспішного пошуку готового рішення Хокінсон та його колеги зробили прототип пристрою та розпочали кампанію на Kickstarter. Кампанія виявилася над-успішною, зібрали 1 200 000 $ замість запланованих 250 000 $ за 30 днів. Розпочався продаж хаба-концентратора з невеликим набором пристроїв

у комплекті: розумна розетка, датчик відкриття дверей, датчик руху, датчик присутності. А в 2019 році, вже під брендом Samsung SmartThings, вийшла третя версія хаба.

Рис. 1.8. Продукти кампанії Kickstarter

Творці стартапу вийшли з DIY-середовища, тому вони послідовно розвивають концепцію відкритого API для гіків-мейкерів, здатних на ранній стадії виступити як early adopters. У кампанії на Kickstarter був і шилд Arduino, і

різні хмарні модулі для безпроблемного підключення вашого стороннього пристрою. В Інтернеті можна знайти безліч саморобних пристроїв, сумісних зі SmartThings.

У нашій роботі пропонується за основу взяти для подальшого використання платформу Samsung SmartThings.

1.4. Ідентифікація ідеї проекту розробки мобільного додатку

Ідея проекту полягає у розробці мобільного додатка для керування «Розумним чайником» з використанням платформи Samsung SmartThings. «Розумний чайник» – це пристрій, який здатний підключитися до Інтернету та отримувати віддалені команди, що дозволяють користувачам керувати ним за допомогою мобільного додатка.

Мобільний додаток для керування «розумним чайником» може мати такі функції:

Увімкнення і вимкнення. Додаток може використовуватися для вмикання та вимикання чайника на відстані.

Регулювання температури. Додаток також може дозволити користувачам регулювати температуру чайника, що є особливо корисним для певних типів чаю, які потребують певного рівня температури.

Таймер. Користувачі можуть встановити чайник на певний час, щоб він автоматично вмикався і вимикався. Це особливо корисно для тих, хто хоче мати готовий чай, коли вони прокидаються або повертаються додому після роботи.

Сповіщення. Додаток може надсилати сповіщення на мобільний телефон, коли чайник припиняє кип'ятіння або коли вода охолоджується до встановленої температури.

Моніторинг води. Чайник може мати датчики для визначення рівня води в чайнику, і додаток може відображати цю інформацію, допомагаючи користувачам знати, коли потрібно додати воду.

Інтеграція з платформою Samsung SmartThings. Це дозволить керувати чайником разом з іншими розумними пристроями у вашому домі через один централізований інтерфейс. Це також може дозволити створювати автоматизовані сценарії – наприклад, можна налаштувати чайник на автоматичне ввімкнення, коли ваш розумний термостат покаже, що ви прокинулися вранці.

При розробці цього додатка потрібно забезпечити кілька речей.

По-перше, безпека є першим питанням для всіх розумних пристроїв, тому додаток повинен використовувати сильне шифрування для спілкування з чайником.

По-друге, додаток повинен бути простим у використанні, з чітким інтерфейсом, що дозволяє користувачам легко знаходити та виконувати потрібні дії.

В цілому ідея проекту має великий потенціал, особливо якщо зосередитись на створених корисних функціях, покращеній зручності використання та забезпеченні безпеки даних користувачів.

РОЗДІЛ 2.
ВИБІР АРХІТЕКТУРИ СИСТЕМИ «РОЗУМНИЙ ЧАЙНИК» ТА ПРИСТРОЇВ

2.1. Архітектура системи «Розумний чайник»

Типова система SmartThings складається з 3 основних частин:

- ✓ Пристрій – у нашому випадку це плата, яка відіграє роль «розумного чайника»;
- ✓ Мобільний додаток, з якого керуємо «чайником» та бачимо його показники;
- ✓ Хмара SmartThings.

Хмара SmartThings є деяким посередником між пристроєм та мобільним додатком.

На рис. 2.1, що представлено нижче видно, що архітектура SmartThings складається з більшої кількості задіяних систем, але мінімальний набір – це пристрій, хмара та мобільний додаток.

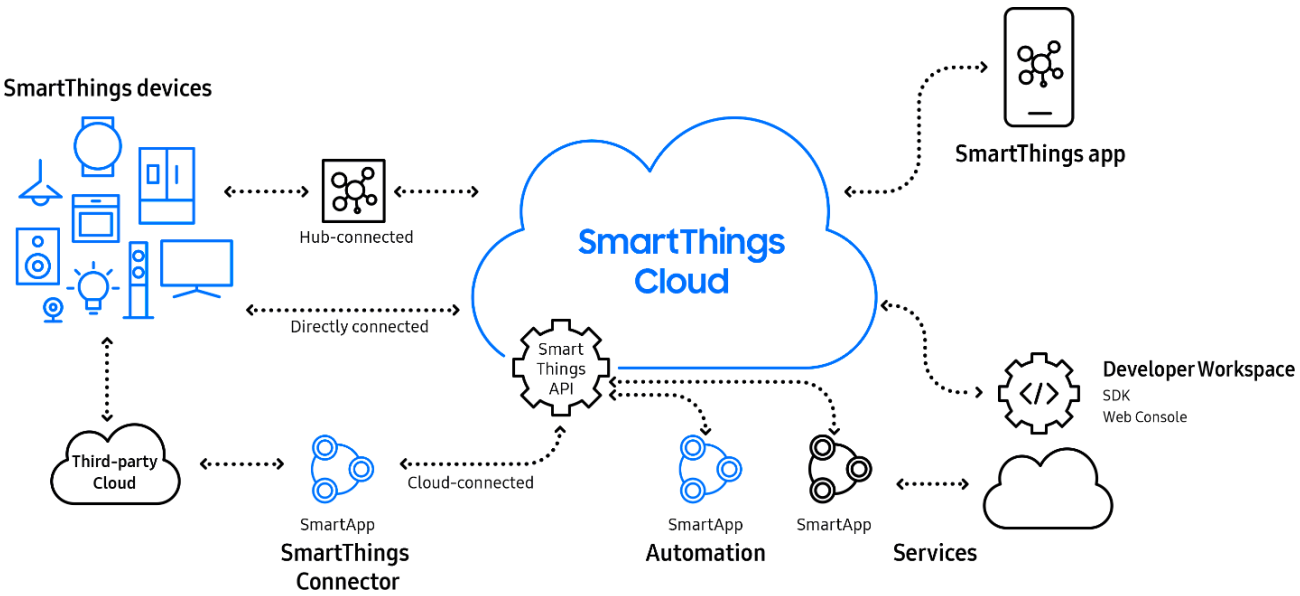

Рис. 2.1. Архітектура системи «Розумний чайник»

Пристрій збиратимемо із множини частин, які описано нижче. Зокрема, до пристрою входить плата мікроконтролера ESP8266 - у нашому випадку це Amperka Troyka Wi-Fi [20].

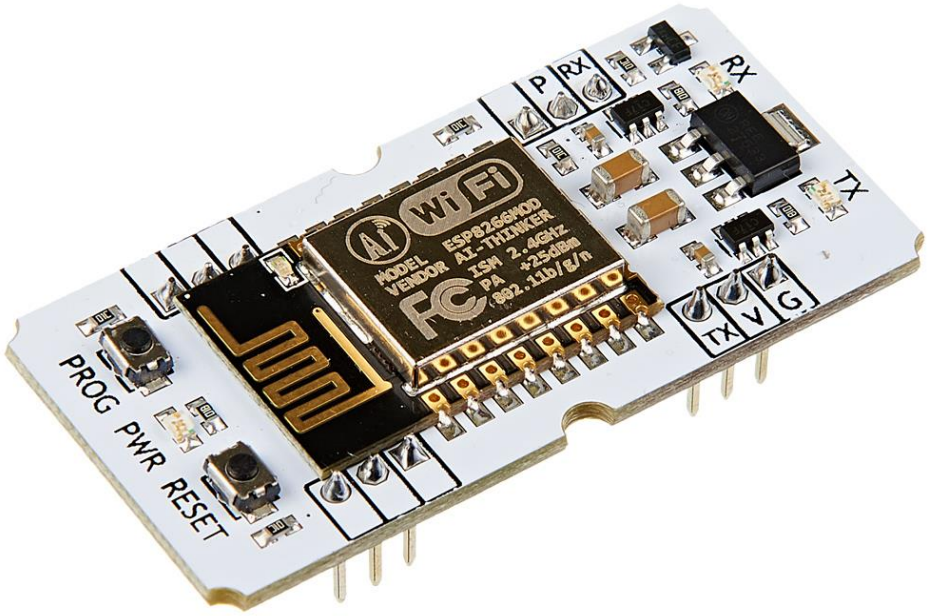

Рис. 2.2. Плата мікроконтролера ESP8266

Окрім того, до пристрою входить кнопка тактова, триколірний світлодіод, п'єзо-піщалка/зумер (активна), резистори 330 Ом, перемички та з'єднувальні дроти.

2.3. Вибір пристроїв та їх основні функції

Нами пропонується використовувати SmartThings Devices. При цьому можуть бути не тільки пристрої з комплекту SmartThings, а й безліч сторонніх зі списку елементів.

Якщо пристрою немає у списку офіційно підтримуваних, можна пошукати до нього Device Handler – він може бути наданий спільнотою розробників - або написати свій власний.

У нашому випадку найскладнішим буде датчик температури, а поки значення температури в «чайнику» генеруватимемо програмно.

Нижче у форматі таблиці наведено основні функції «Розумного чайника», з яких частин пристрою, хмари та телефону виконуватимуть ці функції та як відбуватиметься керування (табл. 2.1).

Таблиця 2.1 – Основні функції «Розумного чайника» та їх виконання

Основні функції	Частини пристрою, що виконуватимуть функції	Принцип керування
Увімкнути пристрій фізично	- джерело питання - інтерфейс пристрою для запитання - зумер - вбудований світлодіод - rgb світлодіод	- Модуль для живлення входить у потрібний інтерфейс. - Вбудований світлодіод світиться. - Rgb світить синім. - Зумер коротко пищить
Підключити телефон до пристрою	- телефон з виходом в інтернет - увімкнений пристрій - вбудований світлодіод	- Додавання пристрою. onboarding через UI - Під час підключення світлодіод блимає. - При підключенні світлодіод світить
Увімкнути/почати кип'ятіння з телефону	- UI телефону - датчик температури - rgb світлодіод - зумер	- Кнопка UI переходить у режим Увімкнена. - RGB Світлодіод світить червоним. - Нові дані з датчика температури відправляються в хмару і відображаються в UI. - Коли температура >= заданої температури кипіння. 2-3 секунди звучить зумер. і rgb світлодіод світить зеленим - Після сигналу зумера світлодіод перемикається на синій. кнопка запуску в UI в режимі вимкнена
Увімкнути/почати кип'ятіння з пристрою	- UI телефону - кнопка - rgb світлодіод - зумер	- Після натискання фізичної кнопки відбувається описане вище
Задати температуру кипіння з телефону	- UI телефону - rgb світлодіод	- у UI нове значення Heating temperature, - Світлодіод блимає синім.

На підставі даних таблиці 2.1 можна сказати, що запропонований «Розумний чайник» виконуватиме 5 функцій із своїми особливостями керування.

2.3. Реєстрація створеного пристрою

Робота починається в SmartThings Developer Workspace, де потрібно залогінитись за допомогою Samsung Account. Цей обліковий запис створюється безкоштовно, і для його створення не обов'язково мати телефон Samsung.

У Developer Workspace одразу пропонується створити перший проект. Погоджуємося із запропонованими, після чого із двох варіантів проектів потрібно вибрати Device Integration (рис. 2.3).

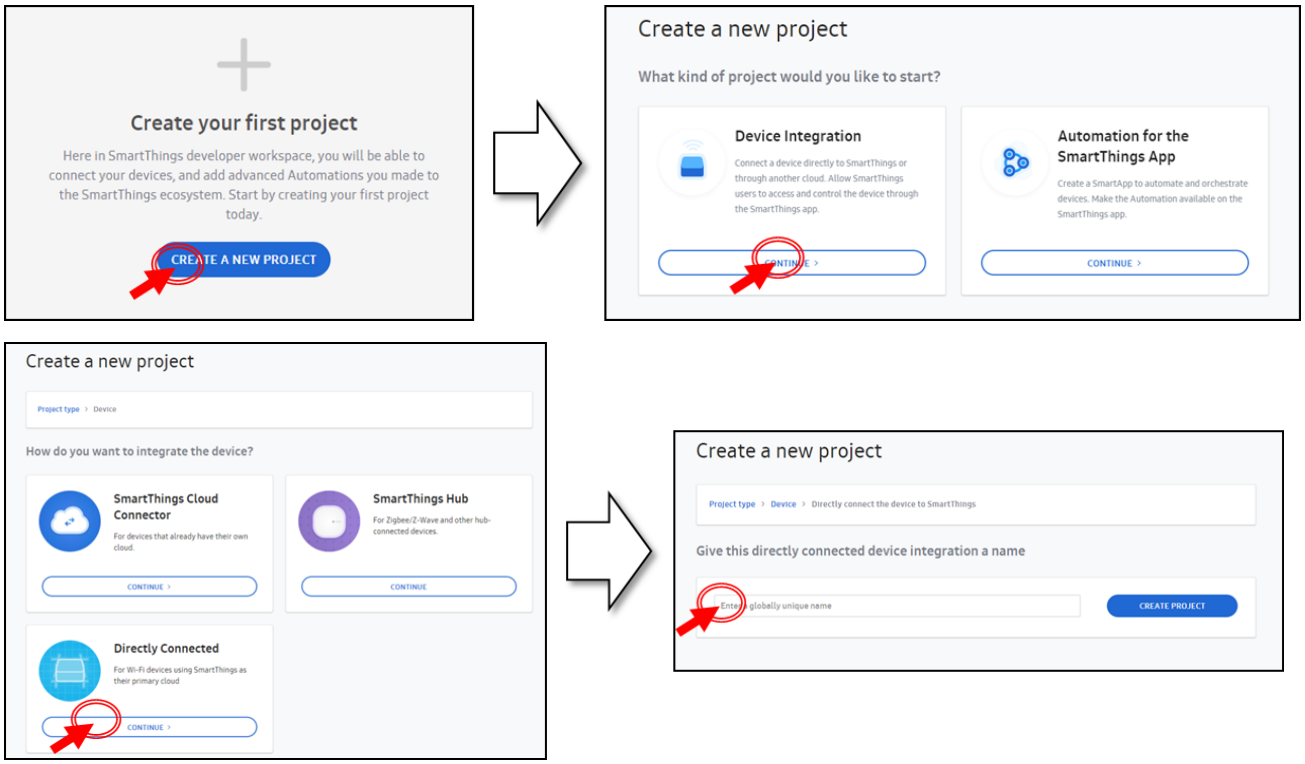

Рис. 2.3. Вибір варіанту проекту у Developer Workspace

На вибір буде дано три варіанти інтеграції пристрою:

✓ SmartThings Cloud Connector – для підключення пристроїв, що мають власну хмару, яка буде пов'язана з хмарою SmartThings.

✓ SmartThings Hub – для пристроїв ZigBee, Z-Wave, Bluetooth, і на цей момент недоступний, оскільки хаб офіційно поки не продається в Україні.

✓ Directly Connected – найпростіший варіант для підключення безпосередньо, без будь-яких посередників.

Вибираємо третій варіант. Придумайте назву проекту, наприклад, для простоти SmartKettle.

Додаємо профіль пристрою. Якщо говорити в термінах ОПП, то профіль пристрою – це такий «клас», а конкретний пристрій – «об’єкт класу».

Нам потрібно визначити функціональні характеристики пристрою в термінах SmartThings Capability (можливостей). Про те, які взагалі «можливості» пристрою існують, можна прочитати в інструкції.

Заповнюємо поля своїми значеннями. Наприклад, ім'я профілю можна вписати Test Switch Profile, опис – будь-який текст, тип пристрою – Switch, Vendor ID – будь-який текст. «Тип пристрою» впливає на іконку вашого пристрою та на інтерфейс користувача. У списку не так багато можливих типів пристроїв, тому, якщо не знайдете саме вашого – не засмучуйтесь і просто виберіть найближче та схоже. На останньому екрані (Dashboard Control) нічого не міняється.

Після створення профілю, слід натиснути кнопку «Add device profile to project» вгорі праворуч.

Тепер додаємо профіль аутентифікації (Onboarding profile). Якщо стисло, то в цьому профілі можна кастомізувати екрани, що з'являються при першому підключенні та логін. Введіть будь-яку назву та будь-яке тризначне число. Інші два екрани можна поки пропустити. На останньому екрані виберіть варіант підтвердження – Button Confirm, після натискання кнопки на пристрої.

Якщо кнопки немає (наприклад, ви використовуєте інший ESP-модуль), то можете вибрати варіант без аутентифікації – він називається Just Works, тобто «просто працює» без зайвих питань.

І знову ж таки, за аналогією з попереднім прикладом, не забудьте натиснути кнопку «Add device onboarding to project».

Залишиться лише натиснути кнопку Deploy. Після цього можна побачити пристрій у програмі SmartThings – але лише в режимі розробника (Developer Mode). Якщо Deploy не натискається, і буде написано «Please add a Model Name» – це означає, що не вказано Device Onboarding ID. Слід промотати сторінку нижче і ввести будь-яке ім'я на ваш розсуд.

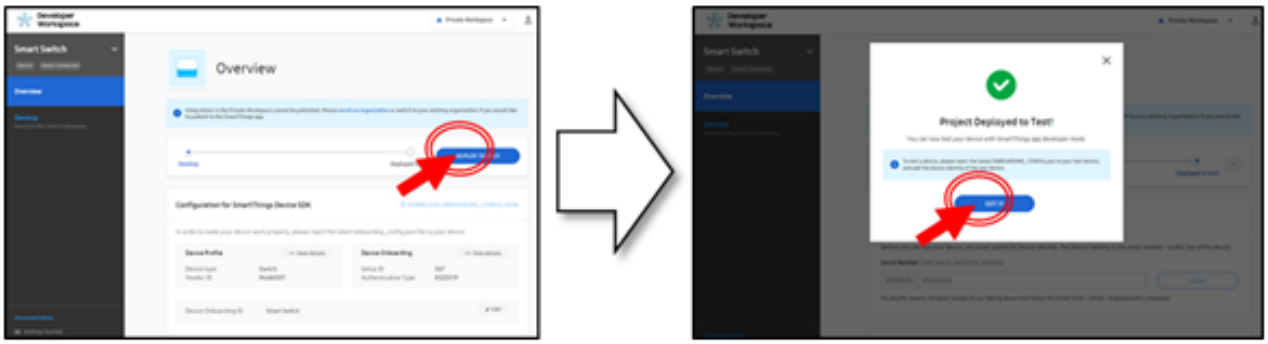

Рис. 2.4. Налаштування відображення пристрою у програмі SmartThings

Тепер можна почати тестування віртуального пристрою на вашому телефоні.

РОЗДІЛ 3.
РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ КЕРУВАННЯ «РОЗУМНИМ ЧАЙНИКОМ»

3.1. Основні етапи створення мобільного додатку для керування «Розумним чайником»

Розробка мобільного додатку для керування «Розумним чайником» передбачає наступні етапи. Насамперед передбачено, що весь процес створення робочого макету розбити на такі кроки:

1) налаштування середовища розробки;
   - ✓ завантажити sdk;
   - ✓ докачати підмодулі (iot-core та вибраний bsp);
   - ✓ згенерувати ключі;
   - ✓ встановити інструменти розробки;
2) реєстрація пристрою;
   - ✓ створити проект;
   - ✓ створити профіль пристрою;
   - ✓ створити схему авторизації;
   - ✓ зареєструвати тестовий пристрій (завантажити ключі);
   - ✓ завантажити onboarding_config.json;
3) розробити мобільного додатку;
   - ✓ створити новий проект;
   - ✓ додати інформацію про пристрій (2 json файли) ;
   - ✓ розробити прошивку;
   - ✓ зібрати та завантажити прошивку на пристрій;
4) тестування мобільного додатку;
   - ✓ включити у мобільному додатку режим розробника;
   - ✓ додати пристрій;
   - ✓ перевірити функції.

Наша робота скерована на розробку програми мобільного додатку та прошивки. Інші перераховані процеси вважаємо, що виконані попередньо і у цій роботі їх не розглядаємо. Детальна інформація про ці процеси є у інструкціях та інформація про них задокументована на сайті SmartThings [4].

3.2. Розробка мобільного додатку

Розробка програми для мобільного додатку включає роботу в workspace проекту і створення прошивки пристрою. При розробці прошивки працюємо з 4 основними частинами програми:

- ✓ capabilities;
- ✓ управління периферією;
- ✓ ініціалізація;
- ✓ бізнес-логіка взаємодії capabilities та периферії.

Основний об'єкт уваги при написанні коду прошивки – Capability. Capability – це можливість/функція системи. Capabilities надаються командою розробників SmartThings, але також можна створювати свої кастомні capabilities.

В інтерфейсі програми кожен Capability має своє відображення. Capability має задаємо ідентифікатор, ім'я, статус, атрибути/властивості та команди. Розробник може змінювати значення атрибутів у коді прошивки, і ці зміни будуть передані та помітні в інтерфейсі мобільного додатка. Також розробник може обробляти команди, які в тому ж інтерфейсі доступні для запуску/виконання.

Для кращого розуміння розберемо приклад можливості термостату задати значення нагріву – Capability Thermostat Heating Setpoint. Це Capability дозволяє встановити значення температури, до якої чайник повинен зігріти воду.

На рис. 3.1 показано, як додати Capability до проекту. При виборі вже можна зрозуміти, за що відповідає це Capability, які атрибути та команди він має.

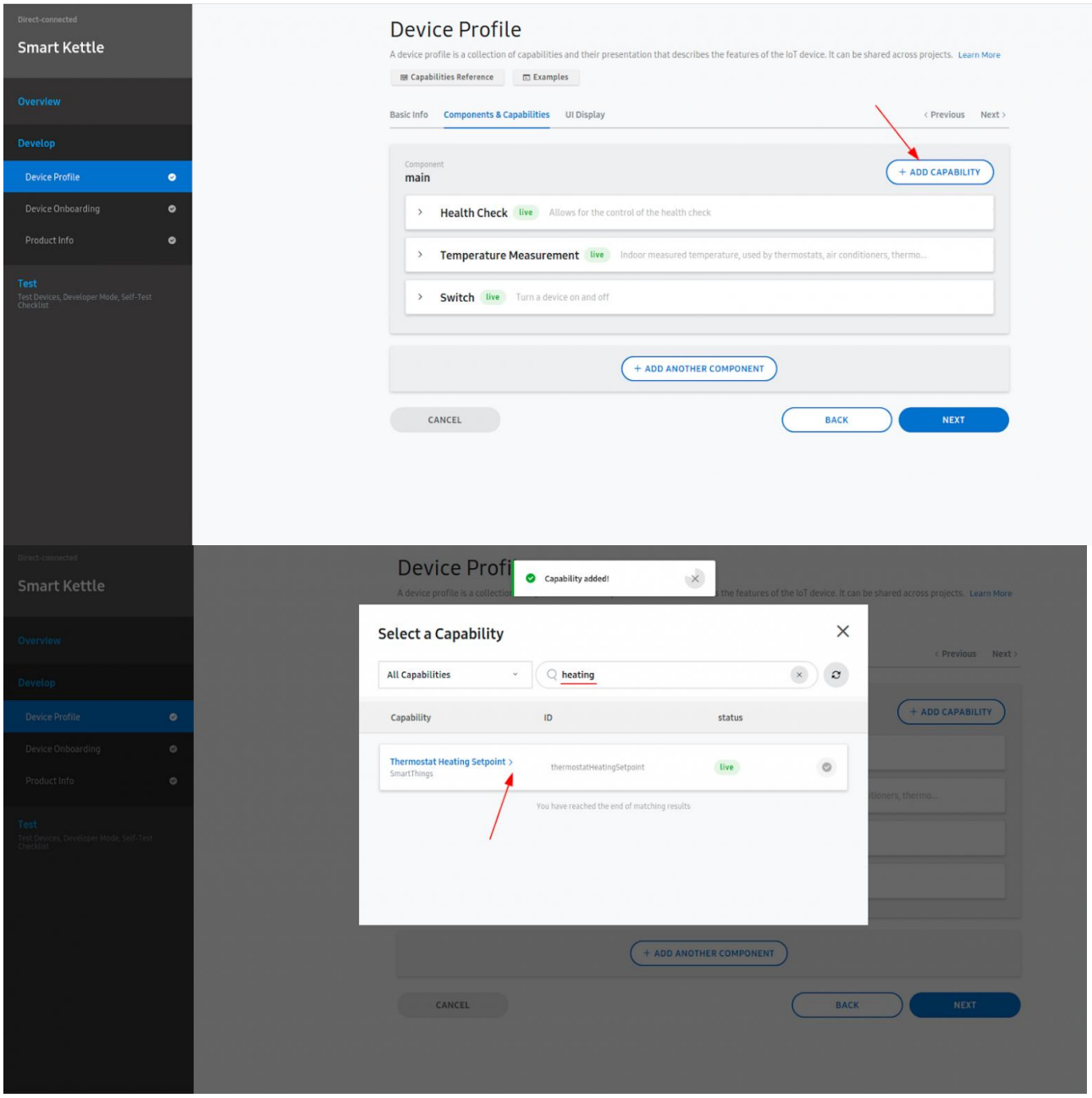

Рис. 3.1. Процес додавання Capability до проекту

На рис. 3.2, а показаний загальний вигляд вибраного пристрою в мобільному додатку зі списком всіх його можливостей. З цього екрана видно значення атрибутів усіх можливостей. На наступному рис. 3.2, б вибрано Capability Thermostat Heating Setpoint, і користувач може відправити команду зміни температури, задавши бажане значення температури в діалоговому вікні.

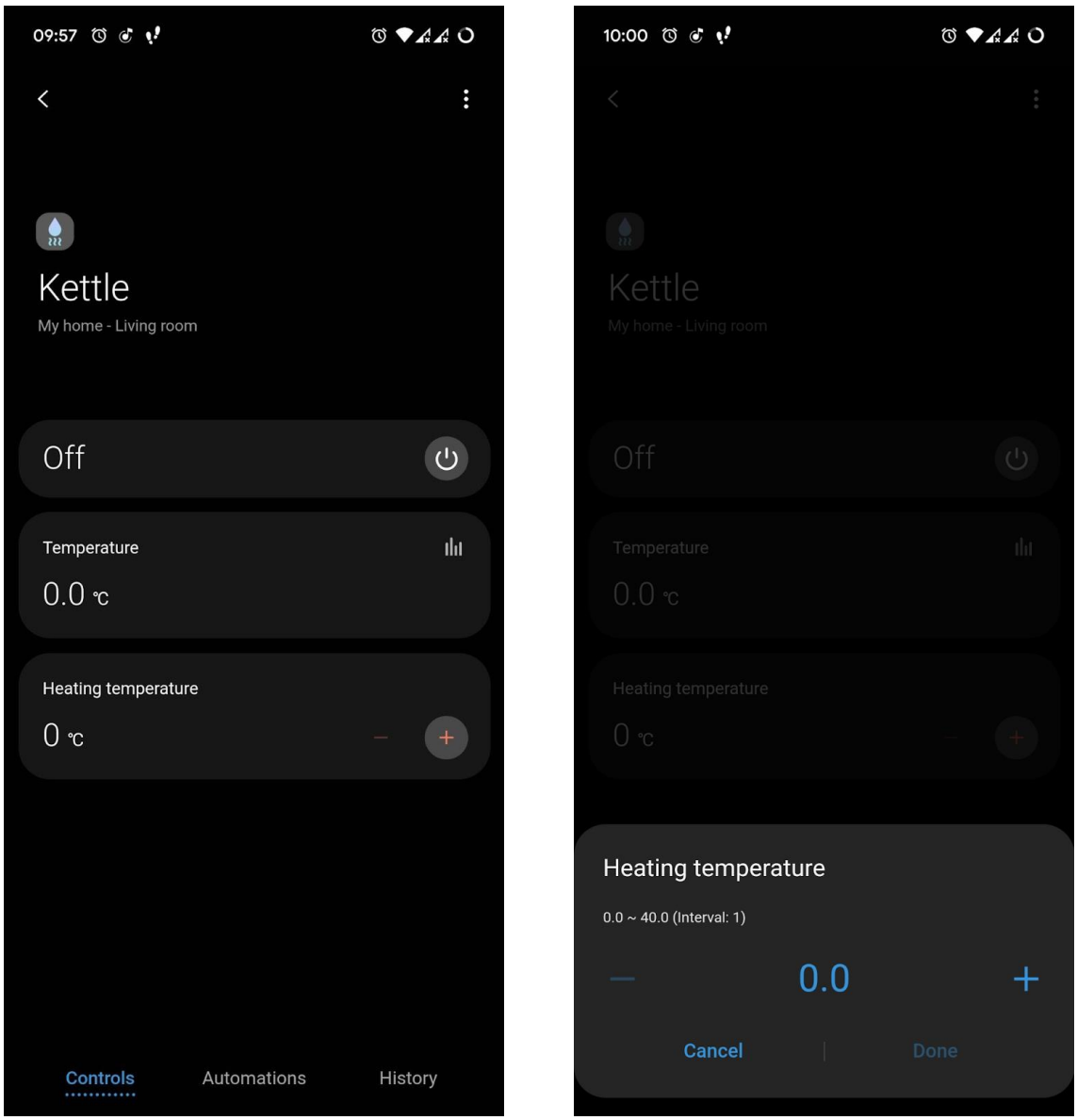

Рис. 3.2. Загальний вигляд вибраного пристрою в мобільному додатку та вікно користувача із вибором значення температури

На наступному рис. 3.3 подано код Capability.

```
} caps_helper_thermostatHeatingSetpoint = {
    .id = "thermostatHeatingSetpoint",
    .attr_heatingSetpoint = {
        .name = "heatingSetpoint",
        .property = ATTR_SET_VALUE_MIN | ATTR_SET_VALUE_MAX | ATTR_SET_VALUE_REQUIRED | ATTR_SET_UNIT_REQUIRED,
        .valueType = VALUE_TYPE_NUMBER,
        .units = {"F", "C"},
        .unit_F = "F",
        .unit_C = "C",
        .min = -460,
        .max = 10000,
    },
    .cmd_setHeatingSetpoint = { .name = "setHeatingSetpoint" }, // arguments: setpoint(number)
};
```

Рис. 3.3. Програмний код Capability

Capability має id = "thermostatHeatingSetpoint", атрибут heatingSetpoint з обов'язковим значенням, команда setHeatingSetpoint(setpoint) з обов'язковим аргументом.

Таким чином, Capability дозволяє абстрагуватися від того, як і яким залізом реалізовані можливості та взаємодіяти мобільному додатку з пристроєм на основі його можливостей/функціоналу.

Управління периферією виконується наступним чином. Тут описують всі функції управління периферією та їх реалізацію. Реалізація залежить від процесора та використовуваних під нього бібліотек та фреймворків. У нашому випадку використовується esp8266 та Espressif ESP8266 RTOS SDK від розробників esp. Написано весь код мовою C.

Після цього виконується ініціалізація. Перед тим як використовувати capabilities та периферію, слід їх проініціалізувати. У кожну Capability потрібно встановити колбек-функції на команди від користувача, значення атрибутів за замовчуванням та ін. Для периферії потрібно налаштувати всі піни і, якщо потрібно, встановити значення за промовчанням.

Бізнес-логіка описується в main.c і використовує функції, оголошені для capabilities та управління периферією. У бізнес-логіці опрацьовуються події. Є 2 джерела подій: сигнали від периферії, у тому числі дії користувача із залізом/пристроєм, та дія користувача в мобільному додатку.

Обробка подій із периферії відбувається в окремому потоці/таску. При спрацюванні певної події від периферії ми можемо змінити стан периферії або змінити стан Capability. Наприклад, після натискання на фізичну кнопку блимаємо світлодіодом. Або після натискання на кнопку відправляємо в хмару нове значення віртуальної кнопки, щоб інтерфейс стан кнопки теж змінилося.

Обробка подій з хмари/мобільної програми виконується наступним чином. Щоб обробити події з хмари, потрібно до кожного використовуваного Capability додати колбеки. Колбек – це функція, що викликається при отриманні команди Capability. Під час підготовки Capability ми реєструємо колбеки для кожної команди. У цих колбек-функціях ми можемо керувати

периферією або відправляти в хмару змінені значення атрибутів будь-якої Capability. Наприклад, при призначенні температури нагріву чайника з мобільного додатку помигати світлодіодом пристрою.

3.3. Розробка модуля ініціалізації пристрою для підключення

Пристрій SmartThings Direct Connect – це пристрій із підтримкою Wi-Fi, який використовує SmartThings Cloud як основну хмарну інфраструктуру. І цей пристрій буде спілкуватися за допомогою протоколу MQTT. Під час вибору мови написання коду вибрали мову С. У порівнянні із мовою Python враховано, що Python – це високорівнева мова програмування, яка не може мати прямого доступу до низькорівневих операторів і системних ресурсів, як це є у мові С.

Програма починається з функції app_main, на неї і звернемо спочатку увагу. Спочатку відбувається ініціалізація пристрою для підключення до хмари ST (рис. 3.4).

```
void app_main(void)
{
  unsigned char *onboarding_config = (unsigned char *) onboarding_config_start;
  unsigned int onboarding_config_len = onboarding_config_end - onboarding_config_start;
  unsigned char *device_info = (unsigned char *) device_info_start;
  unsigned int device_info_len = device_info_end - device_info_start;
  int iot_err;
  // st_dev.h
  ctx = st_conn_init(onboarding_config, onboarding_config_len, device_info, device_info_len);
  if (ctx != NULL) {
    iot_err = st_conn_set_noti_cb(ctx, iot_noti_cb, NULL);
    if (iot_err)
      printf("fail to set notification callback function\n");
  } else {
    printf("fail to create the iot_context\n");
  }
    /* . . . */
}
```

Рис. 3.4. Програмний код ініціалізації пристрою для підключення до хмари ST

У цих функціях використовуються константи та змінні, які потрібно оголосити глобально (рис. 3.5).

```
// onboarding_config_start is null-terminated string
extern const uint8_t onboarding_config_start[]   asm("_binary_onboarding_config_json_start");
extern const uint8_t onboarding_config_end[]     asm("_binary_onboarding_config_json_end");

// device_info_start is a null-terminated string
extern const uint8_t device_info_start[]     asm("_binary_device_info_json_start");
extern const uint8_t device_info_end[]       asm("_binary_device_info_json_end");

IOT_CTX* ctx = NULL;
```

Рис. 3.5. Програмний код оголошення глобальних констант та змінних

У функції st_conn_set_noti_cb можна встановити колбек для подій-оповіщень, таких як видалення пристрою або обмеження швидкості. Це не така важлива спочатку можливість, але встановити колбек варто. У нашому випадку для двох видів подій виводиться інформація про них (рис. 3.6).

```
static void iot_noti_cb(iot_noti_data_t *noti_data, void *noti_usr_data)
{
  printf("Notification message received\n");
  if (noti_data->type == IOT_NOTI_TYPE_DEV_DELETED) {
    printf("[device deleted]\n");
  } else if (noti_data->type == IOT_NOTI_TYPE_RATE_LIMIT) {
    printf("[rate limit] Remaining time:%d, sequence number:%d\n",
    noti_data->raw.rate_limit.remainingTime, noti_data->raw.rate_limit.sequenceNumber);
  }
}
```

Рис. 3.6. Програмний код встановлення функцій зворотнього виклику для подій-оповіщень

Список таких подій визначено у перерахунку enum iot_noti_type у st_dev.h. Далі в app_main викликається функція ініціалізації всіх Capability програми, capability_init. У нас три Capability: Switch, Temperature Measurement та Heating Setpoint. Крім виклику функції ініціалізації для кожного Capability можна додати додаткову колбек функцію обробки команд від користувача, яка, наприклад, може керувати периферією. Можна також встановити початкові значення атрибутів Capability (рис. 3.7).

```
static void capability_init()
{
  // викликаємо функцію ініціалізації Capability перемикача
  cap_switch_data = caps_switch_initialize(ctx, "main", NULL, NULL);
  if (cap_switch_data) {
    // встановлюємо додаткові колбеки функції
    cap_switch_data->cmd_on_usr_cb = cap_switch_cmd_cb;
    cap_switch_data->cmd_off_usr_cb = cap_switch_cmd_cb;
    // устанавливаем значение атрибута переключателя по умолчанию
    cap_switch_data->set_switch_value(cap_switch_data, caps_helper_switch.attr_switch.value_off);
  }
  // викликаємо функцію ініціалізації Capability показу температури
  cap_temperature_data = caps_temperatureMeasurement_initialize(ctx, "main", NULL, NULL);
  if (cap_temperature_data) {
    // встановлюємо значення та міру/unit температури за умовчанням
    cap_temperature_data->set_temperature_unit(cap_temperature_data, caps_helper_temperatureMeasurement.attr_temperature.unit_C);
    cap_temperature_data->set_temperature_value(cap_temperature_data, 0);
  }
  // викликаємо функцію ініціалізації Capability вибору температури нагрівання
  cap_heatingSetpoint_data = caps_thermostatHeatingSetpoint_initialize(ctx, "main", NULL, NULL);
  if (cap_heatingSetpoint_data) {
    // встановлюємо додаткову колбек функцію
    cap_heatingSetpoint_data->cmd_setHeatingSetpoint_usr_cb = cap_thermostat_cmd_cb;
    // встановлюємо міру/unit температури нагрівання
    cap_heatingSetpoint_data->set_unit(cap_heatingSetpoint_data, caps_helper_thermostatHeatingSetpoint.attr_heatingSetpoint.unit_C);
  }
}
```

Рис. 3.7. Програмний код встановлення початкових значень атрибутів Capability

```
void iot_gpio_init(void)
{
  // esp sdk specific
  gpio_config_t io_conf;
  // відключаємо переривання
  io_conf.intr_type = GPIO_INTR_DISABLE;
  // встановлюємо режим піна на вихід
  io_conf.mode = GPIO_MODE_OUTPUT;
  // вибираємо пін
  io_conf.pin_bit_mask = 1 << GPIO_OUTPUT_MAINLED;
  // вибираємо для піна pull-down
  io_conf.pull_down_en = 1;
  io_conf.pull_up_en = 0;
  // конфігуруємо пін вбудованого світлодіода
  gpio_config(&io_conf);
  // залишаємо ту ж конфігурацію, але застосуємо її для піна зумера
  io_conf.pin_bit_mask = 1 << GPIO_OUTPUT_BUZZER;
  // конфігуруємо цей пін
  gpio_config(&io_conf);
  // конфігуруємо піни rgb світлодіода
  io_conf.pin_bit_mask = 1 << GPIO_OUTPUT_RGBLED_R;
  gpio_config(&io_conf);
  io_conf.pin_bit_mask = 1 << GPIO_OUTPUT_RGBLED_G;
  gpio_config(&io_conf);
  io_conf.pin_bit_mask = 1 << GPIO_OUTPUT_RGBLED_B;
  gpio_config(&io_conf);
  // конфігуруємо пін кнопки-перемикача
  io_conf.intr_type = GPIO_INTR_ANYEDGE;
  io_conf.mode = GPIO_MODE_INPUT;
  io_conf.pin_bit_mask = 1 << GPIO_INPUT_SWITCH;
  io_conf.pull_down_en = 0;
  io_conf.pull_up_en = 1;
  gpio_config(&io_conf);
  // відключаємо сервіс переривань
  gpio_install_isr_service(0);
```

Рис. 3.8. Програмний код функції ініціалізації iot_gpio_init

Останньою функцією ініціалізації є iot_gpio_init, яка конфігурує та ініціалізує піни. Ця частина коду залежить від вибраного фреймворку/бібліотек під мікроконтролер. У нашому випадку код ініціалізації написано за допомогою ESP8266 RTOS SDK від розробників esp (рис. 3.8).

Після ініціалізації в app_main виконується підключення до хмари ST за допомогою функції connection_start (рис. 3.9).

```
static void connection_start(void)
{
  iot_pin_t *pin_num = NULL;
  int err;
  // process on-boarding procedure. There is nothing more to do on the app side than call the API.
  err = st_conn_start(ctx, (st_status_cb)&iot_status_cb, IOT_STATUS_ALL, NULL, pin_num);
  if (err) {
    printf("fail to start connection. err:%d\n", err);
  }
}
```

Рис. 3.9. Програмний код підключення до хмари ST

Залишається лише запуск потоку для обробки подій від периферії, який розберемо далі. Запускати потік можна за допомогою або ST SDK, або обрану RTOS безпосередньо. У нашому випадку можна використовувати FreeRTOS, але ми викличемо функцію iot_os_thread_create з ST SDK (рис. 3.10).

```
void app_main(void)
{
  /* . . . */
  // обробка подій від периферії в окремому потоці/таску
  iot_os_thread_create(app_main_task, "app_main_task", 2048, NULL, 10, NULL);
}
```

Рис. 3.10. Програмний код запуску потоку для обробки подій від периферії

3.5. Створення модуля для перевірки зовнішній подій

У app_main ми запустили функцію app_main_task в окремому потоці/таску. Всередині функції нескінченний цикл із перевіркою станів та подій периферії. Головне тут у правильному порядку керувати периферією та

відправляти у хмару значення тих Capability, які мають відображення у фізичному пристрої – це перемикач та температура.

Якщо було натиснуто фізичну кнопку, значить процес «кип’ятіння» запущено. Змінюється стан thermostat_enable і далі можна періодично набувати значення поточної температури. Використовується таймер iot_os_timer для отримання значень температури один раз на кілька секунд (рис. 3.11).

```
if (get_button_event()) {
  // змінити значення перемикача на увімкнене
  cap_switch_data->set_switch_value(cap_switch_data, caps_helper_switch.attr_switch.value_on);
  // відправити в хмару нове значення перемикача
  cap_switch_data->attr_switch_send(cap_switch_data);
  change_switch_state(get_switch_state());
  // допустити процес нагрівання/кип'ятіння
  thermostat_enable = true;
}
```

Рис. 3.11. Програмний код запуску процесу «кип’ятіння»

При отриманні температури її кількісне значення відправляється в хмару за допомогою структури/об'єкта Temperature Capability. Також змінюється колір світлодіода із синього на червоний.

```
// Якщо триває процес нагрівання і можна отримати значення температури
if (thermostat_enable && get_temperature_event(timer)) {
  // отримати значення температури
  temperature_value =  temperature_event(temperature_value);
  // вимкнути синій та увімкнути червоний колір світлодіода
  change_rgb_state(GPIO_OUTPUT_RGBLED_B, LED_GPIO_OFF);
  change_rgb_state(GPIO_OUTPUT_RGBLED_R, LED_GPIO_ON);
  // зберегти нове значення температури
  cap_temperature_data->set_temperature_value(cap_temperature_data, temperature_value);
  // відправити значення температури в хмару
  cap_temperature_data->attr_temperature_send(cap_temperature_data);
}
```

Рис. 3.12. Програмний код відправлення в хмару значення температури

Якщо температура більше температури кипіння, то процес кип'ятіння завершується. Змінюється стан thermostat_enable, колір світлодіода на зелений і стає доступним процес пищання зумера – змінюється стан buzzer_enable (рис. 3.13).

```
if (thermostat_enable && temperature_value >= heating_setpoint) {
    thermostat_enable = false;
    change_rgb_state(GPIO_OUTPUT_RGBLED_R, LED_GPIO_OFF);
    change_rgb_state(GPIO_OUTPUT_RGBLED_G, LED_GPIO_ON);
    temperature_value = 0;
    // зробити зумер доступним
    buzzer_enable = true;
}
```

Рис. 3.13. Програмний код доступу до процесу пищання зумера

Коли зумер пропищить, колір світлодіода змінюється із зеленого назад на синій, а значення атрибуту Switch Capability зміниться з увімкненого на вимкнений і відправиться в хмару для відображення в UI програми.

3.5. Додавання до профілю пристрою та використання кастомних capabilities

У профілі пристроїв Capability та його подання стають доступними для вибору не відразу. Іноді це займає кілька хвилин після створення, іноді більше години. Далі вибір не відрізняється від вибору стандартних Capability. Але спростити пошук можна, відзначивши у фільтрі My Capabilities елементи SmartThings (рис. 3.14).

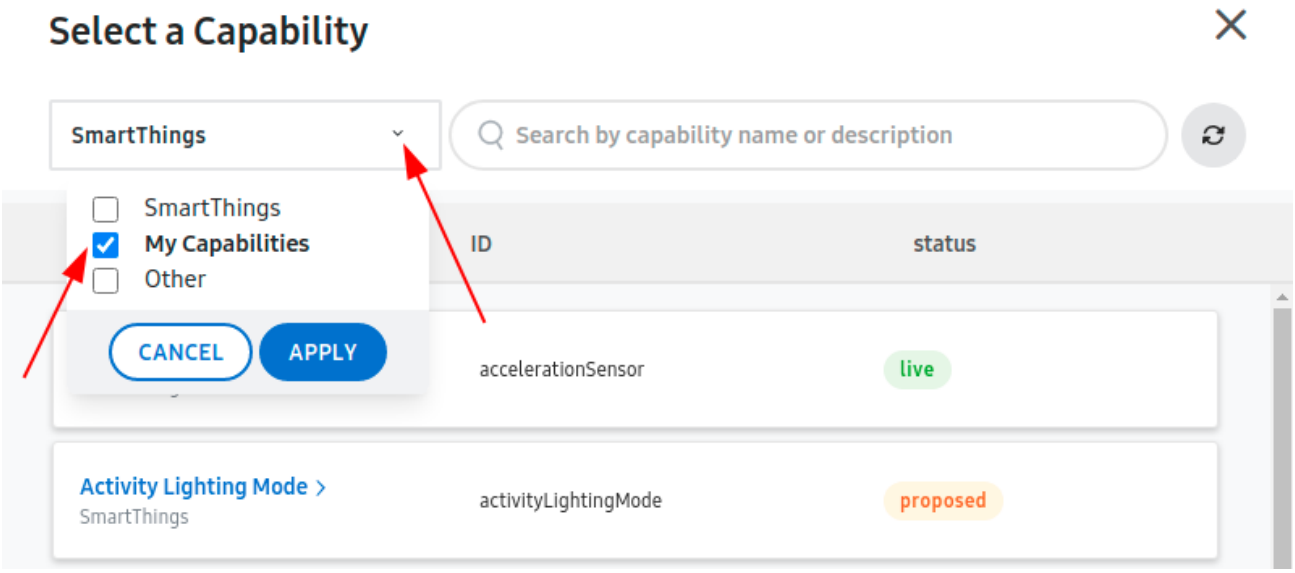

Рис. 3.14. Додавання до профілю пристрою My Capabilities

Використання кастомних capabilities у коді прошивки також відрізняється від використання стандартних capabilities. Головне тут правильно передавати ідентифікатор, назви атрибутів та команд. З цим допомагають заголовні файли-помічники.

Кожному Capability відповідає файл-помічник з ім'ям iot_caps_helper_{capabilityName}.h. Далі в коді відбувається звернення до структури файлу-помічника, яка містить правильні ідентифікатори, назву атрибута, команди та інші властивості Capability. При цьому можна створити такий файл-помічник для нашого кастомного Capability або замінити ідентифікатор Capability, який ми взяли за основу.

Другий варіант передбачає, що файлом-помічником базового Capability не зможемо скористатися, він перетвориться на файл-помічник нашого кастомного Capability. Нижче наведено вміст файлу-помічника iot_caps_helper_thermostatHeatingSetpoint.h із зміненими ідентифікатором, ім'ям атрибуту та назвою команди (рис. 3.15).

```
enum {
  CAP_ENUM_THERMOSTATHEATINGSETPOINT_HEATINGSETPOINT_UNIT_F,
  CAP_ENUM_THERMOSTATHEATINGSETPOINT_HEATINGSETPOINT_UNIT_C,
  CAP_ENUM_THERMOSTATHEATINGSETPOINT_HEATINGSETPOINT_UNIT_MAX
};

const static struct iot_caps_thermostatHeatingSetpoint {
  const char *id;
  const struct thermostatHeatingSetpoint_attr_heatingSetpoint {
    const char *name;
    const unsigned char property;
    const unsigned char valueType;
    const char *units[CAP_ENUM_THERMOSTATHEATINGSETPOINT_HEATINGSETPOINT_UNIT_MAX];
    const char *unit_F;
    const char *unit_C;
    const double min;
    const double max;
  } attr_heatingSetpoint;
  const struct thermostatHeatingSetpoint_cmd_setHeatingSetpoint { const char* name; } cmd_setHeatingSetpoint;
} caps_helper_thermostatHeatingSetpoint = {
  // замінюємо старий ідентифікатор нашим
  // .id = "thermostatHeatingSetpoint",
  .id = "titledouble20168.heatingSetpoint",
  .attr_heatingSetpoint = {
    // Якщо в нашій Capability назва атрибута відрізняється, то міняємо
    // .name = "heatingSetpoint",
    .name = "temperature",
    .property = ATTR_SET_VALUE_MIN | ATTR_SET_VALUE_MAX | ATTR_SET_VALUE_REQUIRED | ATTR_SET_UNIT_REQUIRED,
    .valueType = VALUE_TYPE_NUMBER,
    .units = {"F", "C"},
    .unit_F = "F",
    .unit_C = "C",
    .min = -460,
    .max = 10000,
  },
```

Рис. 3.15. Код вмісту файлу-помічника iot_caps_helper_thermostatHeatingSetpoint.h із зміненими параметрами

Виконана розробка мобільного додатку для керування «Розумним чайником» із використанням платформи Samsung SmartThings. Для пристрою «Розумний чайник» розроблено код прошивки пристрою, інтегрованого в платформу SmartThings. Окрім того обґрунтували структуру типового проекту та пов'язали Capability з можливостями пристрою. Це дозволяє інтегрувати пристрій розумного будинку в екосистему SmartThings, взаємодіяти з ним через мобільний додаток, включати його в сценарії роботи.

РОЗДІЛ 4.
ОХОРОНА ПРАЦІ

4.1. Аналіз умов праці та шкідливих виробничих чинників

В цьому розділі розглядається аналіз умов праці та пожежна безпека на робочому місці інженера-проектувальника систем «Розумний будинок». Згідно ГОСТу 12.0.003-74 у якому визначені фізичні небезпечні та шкідливі виробничі чинники було виділені ті чинники які присутні на робочому місці інженера-проектувальника систем «Розумний будинок»:

- підвищена чи занижена температура повітря робочої зони;
- підвищений рівень шуму на робочому місці;
- підвищений рівень електромагнітних випромінювань;
- підвищена напруженість електричного поля;
- підвищена напруженість магнітного поля;
- відсутність або нестача природного світла;
- недостатня освітленість робочої зони;
- підвищена яскравість світла;
- знижена контрастність.

В наш час технології безперервно і стрімко розвиваються і професії пов’язані з ІТ стали дуже поширеними. Серед них багато сидячої роботи за комп’ютером. Тому важливо забезпечити безпечні умови праці для робітників, що працюють увесь, або майже увесь час сидячи.

Безпечні та комфортні умови праці забезпечують продуктивну роботу та якісне виконання працівниками конкретної задачі. Безпека на робочому місці потрібна для того щоб запобігти травмам та хворобам працівників на робочих місцях, що в свою чергу також дає змогу підприємству працювати якісно.

4.2. Організація робочого місця під час створення мобільного додатку

Під час створення мобільного додатку для керування «Розумним чайником» із використанням платформи Samsung SmartThings важливим елементом є ергономіка робочого місця, що являє собою науку про зручність та організацію робочого простору для ефективної та комфортної праці, опираючись на психофізичні особливості організму людини (рис. 4.1).

Дотримання норм, які відзначені на рис. 4.1 знижує втомленість працівника, збільшує ефективність бізнес-процесу та зберігає здоров’я людей. Ергономіка забезпечує зниження навантаження на тіло людини, а нам відомо, що чим більше людина втомлюється, тим гіршою буде її продуктивність, що не вигідно ні працівнику, ні компанії в якій вона працює. Робоче місце повинно бути організоване відповідно стандартів, методичних вказівок та технічних вимог.

Не дивлячись на те, що робота за комп’ютером може здатися цілком безпечною, на відміну від підприємств з більш підвищеною небезпекою, в ній теж є свої нюанси, яких потрібно дотримуватись.

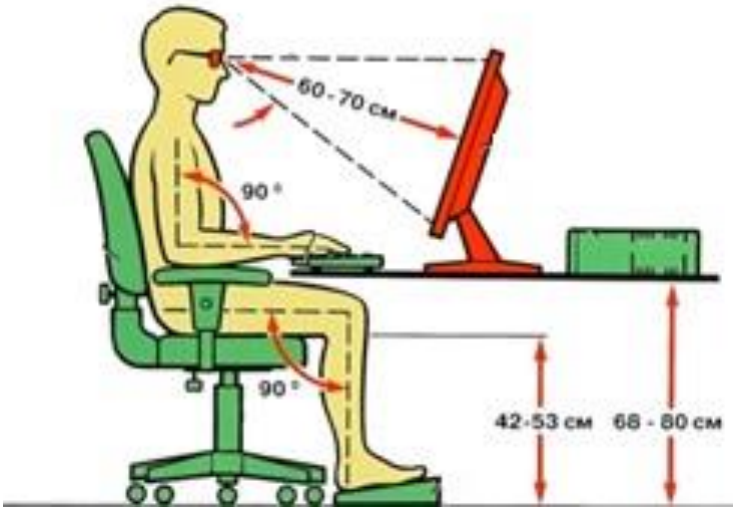

Рис. 4.1. Нормовані значення організації робочого місця під час створення мобільного додатку

Стандарт висоти письмового та комп’ютерного стола 68-80 см, при такій висоті у людини є достатньо міста для ніг. Монітор розміщується на відстані 60-70 см, клавіатура на 10-15 см і вона повинна дозволяти повністю розмістити лікті на столі. Відстань від підлоги до верхнього краю клавіатури 0,7-0,8 м. Відстань від підлоги до центру екрана 0,8-1,9 м.

Якщо недостатньо простору для розміщення усіх необхідних для роботи інструментів, це можна компенсувати розміщенням поряд тумб, стелажу та/або полиць. Усе це повинно бути розташовано по принципу «усе під рукою», що дозволить витрачати менше енергії та направити її на виконання робочого плану.

Якщо людина проводить за ПК більш ніж 6 годин на добу, є ризик розвитку захворювання опорної системи. Для того щоб мінімізувати ризики потрібно обладнати робоче місце спеціальним ортопедичним кріслом для роботи за комп’ютером. Воно оснащене спеціальним валиком у нижній частині спинки, який забезпечує підтримку попереку та повторює анатомічну будову тіла. Спинка крісла у робочому положенні фіксується під прямим кутом 90-95°. Кут між спиною та спинкою крісла 10°-30°. Відстань від підлоги до сидіння крісла 0,3750,5 м. Кут зору 15°-25°. Інформація про ергономіку описана у ГОСТі 12.2.032-78.

4.3. Створення мікроклімату на робочому місці

Для приміщення з ПК існують певні вимоги до вологості, температури та рівню пилу. Температура повинна бути 21…25 °С, відносна вологість – 40…60%, рівень аероіонів – от 400…600 до 50 000 (оптимальний – 1500…5000). Це є оптимальними умовами для комфортного теплового балансу температури тіла людини. На терморегуляцію організму людини також впливає вологість повітря.

Занадто низька вологість, яка менша 20%, викликає пересихання слизових оболонок, а саме дихальних шляхів та очей, а вища 85% ускладнює терморегуляцію. Також дуже важливою є оптимальна вологість, якщо вона вища за норму, то слабкішим стає електростатичне та електромагнітне поле, рівень випромінювання яких в приміщеннях з комп'ютером завжди високий.

Що стосується пилу в приміщеннях з ПК, він є не менш важливим, тому що організм людини погано реагує на велику запиленість. Пил в офісі відрізняється від природнього, він містить частки шкіри людини, будівельних матеріалів, клею, тканей меблів, а також бактерії та віруси. Такий пил може визвати як алергічну реакцію, так і захворювання дихальних шляхів.

Проблемою офісів з комп'ютерами полягає в тому, що через електромагнітне випромінювання пил не осідає на поверхні, він електризується від монітору та висить у повітрі, тому потрапляє на слизові оболонки людини та в легені. Через це вологе прибирання в офісі з ПК повинно проводитися від 3х разів на тиждень.

Також приміщення повинно провітрюватися. Усі заходи безпеки стосовно робочих місць з ПК описані у обов'язкових санітарно-епідеміологічних правилах та нормах – СанПіН 2.2.2/2.4.1340-30 «Гігієнічні вимоги до персональних електронно-обчислювальних машин та організації роботи».

Чи не найбільш важливим є освітлення приміщення та безпосередньо робочого місця, бо більшу частину інформації людина отримує через органи зору, від ступеня втоми очей залежить настрій та самопочуття людини.

Насамперед в приміщенні повинно бути штучне та природне освітлення. Для працівника робоче місце за комп'ютером повинно бути не менше 6 м², а об'єм – більше 20 м³. Має значення й обробка приміщення, а саме її коефіцієнт відображення. Нормою для стін є 0,5-0,6, для стелі 0,7-0,8, для підлоги 0,3-0,5. Для цього застосовують дифузно-видбивні комплектуючі.

Орієнтуватися тільки на природнє освітлення забороняється, але воно є оптимальним, бо більш сприятливе для зору людини. Робоче місце необхідно розмішувати біля вікна. Штучне освітлення використовують, коли природнього

недостатньо. Воно поділяється на загальне, яке використовує систему освітлення стелі, робоче – освітлення на робочому місці здійснюється за допомогою настінних, настільних світильників, та тих, що ставляться на підлогу. Існує документація ДБН В.2.528:20018 , в якій прописані норми та нормативи, які враховуються при організації освітлення при роботі з ПК. Для офісів спільного призначення з використанням комп'ютеру норма освітленості згідно з ДБН 300-500 лк.

Щоб отримати оптимальне освітлення робочого місця, а саме коефіцієнта освітленості потрібно потужність потоку світла розділити на площу. Яскравість освітлення поверхонь, які знаходяться у полі зору повинна бути до 200 кд/м². Яскравість відблисків на екрані монітора не повинна перевищувати 40 кд/м².

4.4. Зниження рівня шуму та забезпечення електробезпеки на робочому місці

Рівень шуму на робочому місці з комп'ютером неповинен перевищувати норм зазначених у СанПіН 2.2.4/2.1.8.562-96. Він складає не більше ніж 50 дБА. Знизити рівень шуму в приміщенні можна за допомогою звукопоглинаючих матеріалів з максимальним коефіцієнтом поглинання звуку в області частот 638000 Гц для обробки стін та стелі робочого приміщення. Джерелами шуму виступають:

- звуки, які доносяться з сусідніх приміщень або вулиці;
- технічні звуки. виникають у процесі функціонування обладнання. щоб мінімізувати шум від нього потрібно використовувати більш якісні пристрої.
- шум джерелом якого є людина. для зменшення шуму в приміщенні існують правила, які встановлює підприємство, порушуючи їх співробітник отримує попередження чи штраф.

На робочому місці працівника розміщуються монітор, клавіатура та системний блок. Коли дисплей включений створюється висока напруга на електронно-променевій трубці в декілька кіловат. Забороняється працювати за комп’ютером, якщо одяг або руки вологі, а також протирати його увімкненому стані.

Потрібно завжди слідкувати за цілісністю проводки, відсутності пошкоджень та наявність заземлення приекранного фільтра. В процесі роботи ПК на корпусах моніторів наведені токи статичної електрики, які при доторканні можуть призвести до розрядів. Вони хоч і не становлять небезпеки для людини, але можуть призвести до поломки комп’ютера.

4.5. Пожежна безпека на робочому місці

Пожежна безпека – комплекс заходів направлених на попередження виникнення випадкової або навмисної пожежі, обмеження та усунення його, якщо він виник та мінімізація наслідків цього явища. Для досягнення потрібного рівня безпеки про роботі з комп’ютером, у виробничому приміщенні повинні бути аптечки першої медичної допомоги, системи автоматичної пожежної сигналізації і вогнегасники. Якщо в приміщені працюють багато комп’ютерів, там повинен бути службовий вимикач, що дозволяє в разі необхідності вимкнути усе живлення кімнати. Пожежна безпека забезпечується пожежною профілактикою та активним пожежним захистом.

Переважна більшість людей гине через токсичність продуктів горіння, а саме отруєнням чадним газом, він більш інтенсивне реагує з гемоглобіном ніж кисень і у людини виникає кисневе голодування та порушення координації рухів. Оксид вуглецю має велику концентрацію в продуктах горіння, тому й створює підвищену небезпеку. Основним токсичними продуктами горіння є оксид сірки та вуглецю, газоподібні кислоти, а саме синильна та соляна, аміак,

альдегіди альфатичні. Чадний газ при концентрації 8-10% приводить до смерті через декілька хвилин.

Температура, яка перевищує 100 °С під час пожежі призводить до втрати свідомості людини і подальше загибелі через декілька хвилин. Така температура може викликати опіки шкіри. Небезпечною температурою вважається від 55 °С. До того ж вона викликає опіки другого ступеня при тривалості впливу 20 с, температура 70 °С завдає шкоди за 1 с.

Для забезпечення пожежної безпеки потрібно проводити бесіди з працівниками стосовно правил пожежної безпеки та не допускати дій, які можуть стати причиною пожежі. Також потрібне встановлення планів евакуації персоналу, технічне обслуговування вогнегасників. Зазвичай причинами пожежі на підприємствах з ПК стають електроприлади, куріння в невстановлених місцях, використання легкозаймистих речовин, порушення технологій, порушення правил використання електроприладів, закрите вентиляційний отвір в електроапаратурі та інше.

Потрібно слідкувати за чистотою приміщення. Сміття та горючі відходи потрібно регулярно утилізувати у спеціально виділене для цього місце.

Евакуаційні виходи, коридори, двері, сходини повинні бути порожніми, нічим не заставлені. Мебель та дроти не повинна бути перешкодою для евакуації людей в разі пожежі. Розташування електричних дротів повинно бути таким щоб вони не пошкоджувались і виключити ризик ураження робітників електричним струмом. По закінченню роботи потрібно вимкнути усі електроприлади та перевірити приміщення.

Для гасіння пожежі у приміщеннях використовують вогнегасники, які призначені для початкової стадії розвитку пожежі. Безпосередньо у приміщеннях з комп’ютерами використовують вогнегасник вуглекислотний ОУ-5, який призначений для гасіння різноманітних матеріалів, електричних установок, які знаходяться під напругою, ПК та оргтехніки. Хід дій такий, що при пожежі потрібно піднести вогнегасник якомога ближче до вогню , направити розтруб у вогнище, зірвати пломбу, далі відкрити вентиль,

натиснути на пусковий важіль, направити газ на вогонь. При цьому розтруб не можна тримати рукою під час його роботи, так як він має дуже низку температуру. Також використовують порошкові вогнегасники ОП-5.

У ДСТУ 3675-98 йдеться про пожежну техніку, вогнегасники переносні, загальні технічні вимоги та методи випробовування. Вогнегасники потрібно грамото розташовувати на підприємстві згідно норм та правил, встановити потрібну кількість та їх положення. Вогнегасники повинні бути на кожному поверсі у кількості не менше ніж 2, у кожного повинен бути сертифікат. Вони повинні бути легкодоступними та розташовуватися у виділених місцях близько до передбачуваного місця пожежі, а також біля евакуаційних шляхів та виходів з приміщення. Вогнегасник повинен бути у робочому стані з запломбованим запірно-пусковим пристроєм. Маса вогнегасника не повинна перевищувати 20 кг. Розташування від підлоги не більше ніж півтора метри до верхньої точки, якщо маса вогнегасника менша ніж 15 кг та один метри, якщо більша. Також можна установити вогнегасник на підставці та на підлогу з надійною фіксацією від падіння. При цьому вогнегасник не повинен заважати пересуванню працівників.

Щоб забезпечити якісний пожежний захист необхідно знати принцип припинення горіння.

Однією з важливих задач пожежної безпеки є забезпечення достатньої міцності будівельної конструкції та захисту приміщень від руйнувань в умовах дії високої температури при пожежі. Приміщення з ПК повинні бути першого та другого ступеня вогнестійкості, через свою велику вартість та категорію пожежної небезпеки.

ВИСНОВКИ І ПРОПОЗИЦІЇ

У наш час цифрових технологій, «розумні» пристрої стають все більш поширеними і необхідними. Вони спрощують наше життя, дозволяючи керувати різними аспектами наших домівок із зручних мобільних додатків. Один з цих «розумних» пристроїв – «розумний чайник». Він може автоматизувати процес приготування чаю або кави, налаштовувати температуру води та багато іншого.

Наш проект «Розробка мобільного додатку для керування «Розумним чайником» із використанням платформи Samsung SmartThings» спрямований на створення інноваційного рішення, яке зробить наші домівки ще більш комфортними, автоматизованими та «розумними». Зазначений проект має практичне скерування та дасть можливість зробити значний внесок у сферу IoT і «розумного» дому.

Використання платформи Samsung SmartThings для керування розумним чайником надає широкий спектр можливостей для користувачів. Від простого вмикання та вимикання пристрою до створення складних сценаріїв і рутин, використання AI для аналізу використання чайника та надання рекомендацій - все це можливо завдяки цій інноваційній платформі.

Нами виконано аналіз існуючих некомерційних систем «Розумним дім». Встановлено, що більшість проектів мають самостійно розроблені програми для управління пристроями, тому користувачам необхідно мати мінімальний досвід програмування, щоб створювати потрібні пристрої на основі існуючих пристроїв. До недоліків некомерційних пристроїв системи «Розумний дім» належить: підвищена складність створення системи розумний дім; збільшує час, необхідний для створення системи розумного будинку.

Нами виконано аналіз існуючих мобільних додатків для керування розумними пристроями. Заслуговує на увагу Home Assistant (рис. 1.4), мобільний додаток для керування розумними пристроями є OpenHAB (рис. 1.5), Domoticz для Android (рис. 1.6), додаток компанії Node-RED. Усі ці додатки

надають широкий спектр можливостей для керування розумними пристроями. Вибір між ними залежить від ваших конкретних потреб та технічних знань.

У нашій роботі пропонується за основу взяти для подальшого використання платформу Samsung SmartThings. В Інтернеті можна знайти безліч саморобних пристроїв, сумісних зі SmartThings, які потребують створення мобільних додатків для керуваннями ними.

Ідея проекту полягає у розробці мобільного додатка для керування «Розумним чайником» з використанням платформи Samsung SmartThings. «Розумний чайник» – це пристрій, який здатний підключитися до Інтернету та отримувати віддалені команди, що дозволяють користувачам керувати ним за допомогою мобільного додатка.

На рис. 2.1, що представлено нижче видно, що архітектура SmartThings складається з більшої кількості задіяних систем, але мінімальний набір – це пристрій, хмара та мобільний додаток. Пристрій збиратимемо із множини частин, які описано нижче. Зокрема, до пристрою входить плата мікроконтролера ESP8266 - у нашому випадку це Amperka Troyka Wi-Fi. Нами пропонується використовувати SmartThings Devices.

Нами у форматі таблиці наведено основні функції «Розумного чайника», з яких частин пристрою, хмари та телефону виконуватимуть ці функції та як відбуватиметься керування (табл. 2.1). На підставі даних таблиці 2.1 можна сказати, що запропонований «Розумний чайник» виконуватиме 5 функцій із своїми особливостями керування.

Робота починається в SmartThings Developer Workspace, де потрібно залогінитись за допомогою Samsung Account. Цей обліковий запис створюється безкоштовно, і для його створення не обов'язково мати телефон Samsung. Нами здійснено вибір варіанту проекту у Developer Workspace. Також налаштовано відображення пристрою у програмі SmartThings.

Нами представлено основні етапи створення мобільного додатку для керування «Розумним чайником». Наша робота скерована на розробку програми мобільного додатку та прошивки. Інші перераховані процеси

вважаємо, що виконані попередньо і у цій роботі їх не розглядаємо. Детальна інформація про ці процеси є у інструкціях та інформація про них задокументована на сайті SmartThings.

Розробка програми для мобільного додатку включає роботу в workspace проекту і створення прошивки пристрою. При розробці прошивки працюємо з 4 основними частинами програми.

На рис. 3.2, а показаний загальний вигляд вибраного пристрою в мобільному додатку зі списком всіх його можливостей. З цього екрана видно значення атрибутів усіх можливостей. На наступному рис. 3.2, б вибрано Capability Thermostat Heating Setpoint, і користувач може відправити команду зміни температури, задавши бажане значення температури в діалоговому вікні. На наступному рис. 3.3 подано код Capability, що дозволяє абстрагуватися від того, як і яким залізом реалізовані можливості та взаємодіяти мобільному додатку з пристроєм на основі його можливостей/функціоналу.

Нами здійснена розробка модуля ініціалізації пристрою для підключення. Пристрій SmartThings Direct Connect – це пристрій із підтримкою Wi-Fi, який використовує SmartThings Cloud як основну хмарну інфраструктуру. І цей пристрій буде спілкуватися за допомогою протоколу MQTT. Під час вибору мови написання коду вибрали мову С.

Програма починається з функції app_main, на неї і звернемо спочатку увагу. Спочатку відбувається ініціалізація пристрою для підключення до хмари ST (рис. 3.4). У цих функціях використовуються константи та змінні, які потрібно оголосити глобально (рис. 3.5). Окрім того виконано створення коду для решти функцій ініціалізації.

Також написано програмний код для створення модуля перевірки зовнішній подій, що лежить в основі ефективного керування «Розумним чайником».

Виконано додавання до профілю пристрою та використання кастомних Capabilities. Також нами запропоновано Нижче наведено вміст файлу-

помічника iot_caps_helper_thermostatHeatingSetpoint.h із зміненими ідентифікатором, ім'ям атрибуту та назвою команди (рис. 3.15).

Виконана розробка мобільного додатку для керування «Розумним чайником» із використанням платформи Samsung SmartThings. Для пристрою «Розумний чайник» розроблено код прошивки пристрою, інтегрованого в платформу SmartThings. Окрім того обґрунтували структуру типового проекту та пов'язали Capability з можливостями пристрою. Це дозволяє інтегрувати пристрій розумного будинку в екосистему SmartThings, взаємодіяти з ним через мобільний додаток, включати його в сценарії роботи.

У нашій роботі запропоновані заходи із забезпечення безпеки праці під час розробки мобільного додатку, що забезпечують покращення умов праці, мікроклімату на робочому місці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Барановський, А. В. Програмування мобільних додатків: навчальний посібник. А. В. Барановський, В. І. Горошко, В. О. Кондратюк. К.: НТУУ «КПІ», 2020. 234 с.

2. Волошин, Ю. М. Інтернет речей: технології та перспективи розвитку Львів: Новий Світ-2000, 2022. 320 с.

3. Геврик Є.О., Пешко Н.П. Гігієна праці на виробництві: Навчальний посібник для студентів вищих навчальних закладів. К.: Чельга, Ніка, Центр, 2004. 280 с.

4. Живіть розумно зі SmartThings [Електронний ресурс]. Режим доступу до ресурсу: https://www.samsung.com/ua/apps/smartthings/

5. Жидецький В.Ц. Охорона праці користувачів комп'ютерів. Львів: Афіша, 2000. 176 с.

6. Жидецький В.Ц., Джигерей В.С., Мельников О.В. Основи охорони праці. Підручник. Вид. 5-е, доп. Львів: Афіша, 2000. 350 с.

7. Катренко Л.А., Катренко А.В., Охорона праці в галузі комп'ютинґу: Підручник. За науковою редакцією В.В. Пасічника. Львів: «Магнолія 2006», 2012. 544 с.

8. Кузьменко, О. В. Основи створення розумного будинку: практичний посібник. Чернівці : Букрек, 2023. 156 с.

9. Проект Blynk [Електронний ресурс]. Режим доступу: https://shoorik007.github.io/

10. Технологія розумного будинку: як AI створює простір, комфортний для життя [Електронний ресурс]. – Режим доступу до ресурсу: https://www.everest.ua/tehnologiya-rozumnogo-budynku-yak-ai-stvoryuye-prostirkomfortnyj-dlya-zhyttya/

11. Чаплінський Ю.П. Мобільні інформаційні системи підтримки прийняття рішень. Наукова-технічна інформація. № 1. 2003. С. 22-26.

12. Arduino [Електронний ресурс]. Режим доступу: http://ipkey.com.ua/uk/faq/916-arduino.html

13. Arduino ESP8266 web server project [Електронний ресурс]. Режим доступу до ресурсу: https://create.arduino.cc/projecthub/harshmangukiya/create-esp8266-webserver-9c32ac

14. Johnson, R. Internet of things: a practical approach / R. Johnson. Boston: MIT Press, 2023. 452 p.

15. Openhab [Електронний ресурс]. Режим доступу до ресурсу: https://www.openhab.org/

16. Richardson, L. RESTful Web API: services for a changing world / L. Richardson, M. Amundsen. O'Reilly Media, 2021. 408 p.

17. Samsung developers. SmartThings Developers Documentation [Electronic resource]. Access mode: https://developer.samsung.com/smartthings

18. Smith, J. Mobile Application Development: A Comprehensive Guide / J. Smith. New York : Springer, 2022. 310 p.

19. Turner, R. Building smart homes using Internet of Things devices / R. Turner. New York: Apres, 2021. 240 p.

20. Wi-Fi (Troyka-модуль): розпинування, схема підключення та програмування [Електронний ресурс]. Режим доступу до ресурсу: http://wiki.amperka.ua/%D0%BF%D1%80%D0%BE%D0%B4%D1%83%D0%BA%D1%82%D1%8B:troyka-wi-fi

Додатки

Додаток А

Фрагменти програмного коду розробки мобільного додатку

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

#include "st_dev.h"
#include "caps_temperatureMeasurement.h"

static double caps_temperature_get_temperature_value(caps_temperature_data_t
*caps_data)
{
    if (!caps_data) {
        printf("caps_data is NULL\n");
        return -1;
    }
    return caps_data->temperature_value;
}

static void caps_temperature_set_temperature_value(caps_temperature_data_t
*caps_data, double value)
{
    if (!caps_data) {
        printf("caps_data is NULL\n");
        return;
    }
    caps_data->temperature_value = value;
}

static const char *caps_temperature_get_temperature_unit(caps_temperature_data_t
*caps data)
{
    if (!caps_data) {
        printf("caps_data is NULL\n");
        return NULL;
    }
    return caps_data->temperature_unit;
}

static void caps_temperature_set_temperature_unit(caps_temperature_data_t
*caps_data, const char *unit)
{
    if (!caps_data) {
        printf("caps_data is NULL\n");
        return;
    }
    caps_data->temperature_unit = (char *)unit;
}

static void
caps_temperatureMeasurement_attr_temperatureValue_send(caps_temperature_data_t
*caps_data)
{
    int sequence_no = -1;

    if (!caps_data || !caps_data->handle) {
        printf("fail to get handle\n");
        return;
    }

    ST_CAP_SEND_ATTR_NUMBER(caps_data->handle,
```

```
            (char *)caps_helper_temperatureMeasurement.attr_temperature.name,
            caps_data->temperature_value,
            caps_data->temperature_unit,
            NULL,
            sequence_no);

    if (sequence_no < 0)
        printf("fail to send temperature value\n");
    else
        printf("Sequence number return : %d\n", sequence_no);
}

static void caps_temperatureMeasurement_init_cb(IOT_CAP_HANDLE *handle, void
*usr_data)
{
    caps_temperature_data_t *caps_data = usr_data;
    if (caps_data && caps_data->init_usr_cb)
        caps_data->init_usr_cb(caps_data);
    caps_temperatureMeasurement_attr_temperatureValue_send(caps_data);
}

caps_temperature_data_t *caps_temperatureMeasurement_initialize(IOT_CTX *ctx,
const char *component, void *init_usr_cb, void *usr_data)
{
    caps_temperature_data_t *caps_data = NULL;
    caps_data = malloc(sizeof(caps_temperature_data_t));
    if (!caps_data) {
        printf("fail to malloc for caps_temperature_data_t\n");
        return NULL;
    }

    memset(caps_data, 0, sizeof(caps_temperature_data_t));

    caps_data->init_usr_cb = init_usr_cb;
    caps_data->usr_data = usr_data;

    caps_data->get_temperature_value = caps_temperature_get_temperature_value;
    caps_data->set_temperature_value = caps_temperature_set_temperature_value;
    caps_data->get_temperature_unit = caps_temperature_get_temperature_unit;
    caps_data->set_temperature_unit = caps_temperature_set_temperature_unit;
    caps_data->attr_temperature_send =
caps_temperatureMeasurement_attr_temperatureValue_send;
    caps_data->temperature_value = 0;
    if (ctx) {
        caps_data->handle = st_cap_handle_init(ctx, component,
caps_helper_temperatureMeasurement.id, caps_temperatureMeasurement_init_cb,
caps_data);
    }
    if (!caps_data->handle) {
        printf("fail to init temperatureSensor handle\n");
    }

    return caps_data;
}
```

```
#include "st_dev.h"
#include "caps/iot_caps_helper_switch.h"

#ifdef __cplusplus
extern "C" {
#endif

typedef struct caps_switch_data
{
    IOT_CAP_HANDLE* handle;
    void *usr_data;
    void *cmd_data;

    char *switch_value;

    const char *(*get_switch_value)(struct caps_switch_data *caps_data);
    void (*set_switch_value)(struct caps_switch_data *caps_data, const char
*value);
    int (*attr_switch_str2idx)(const char *value);
    void (*attr_switch_send)(struct caps_switch_data *caps_data);

    void (*init_usr_cb)(struct caps_switch_data *caps_data);

    void (*cmd_on_usr_cb)(struct caps_switch_data *caps_data);
    void (*cmd_off_usr_cb)(struct caps_switch_data *caps_data);
} caps_switch_data_t;

caps_switch_data_t *caps_switch_initialize(IOT_CTX *ctx, const char *component,
void *init_usr_cb, void *usr_data);
#ifdef __cplusplus
}
#endif
```

```
#include "caps/iot_caps_helper_thermostatHeatingSetpoint.h"
#include "st_dev.h"

#ifdef __cplusplus
extern "C" {
#endif
typedef struct caps_thermostatHeatingSetpoint_data
{
    IOT_CAP_HANDLE *handle;
    void *usr_data;
    void *cmd_data;

    double min;
    double max;
    double value;
    char *unit;

    void (*init_usr_cb)(struct caps_thermostatHeatingSetpoint_data *caps_data);

    double (*get_value)(struct caps_thermostatHeatingSetpoint_data *caps_data);
    void (*set_value)(struct caps_thermostatHeatingSetpoint_data *caps_data,
double value);
    double (*get_min)(struct caps_thermostatHeatingSetpoint_data *caps_data);
    void (*set_min)(struct caps_thermostatHeatingSetpoint_data *caps_data,
double value);
    double (*get_max)(struct caps_thermostatHeatingSetpoint_data *caps_data);
```

```
    void (*set_max)(struct caps_thermostatHeatingSetpoint_data *caps_data,
double value);
    const char *(*get_unit)(struct caps_thermostatHeatingSetpoint_data
*caps_data);
    void (*set_unit)(struct caps_thermostatHeatingSetpoint_data *caps_data,
const char *unit);

    void (*attr_setpointValue_send)(struct caps_thermostatHeatingSetpoint_data
*caps_data);
    void (*cmd_setHeatingSetpoint_usr_cb)(struct
caps_thermostatHeatingSetpoint_data *caps_data);
} caps_thermostatHeatingSetpoint_data_t;

caps_thermostatHeatingSetpoint_data_t
*caps_thermostatHeatingSetpoint_initialize(IOT_CTX *ctx, const char *component,
void *init_usr_cb, void *usr_data);
#ifdef __cplusplus
}
#endif
```

```
static void capability_init()
{
// викликаємо функцію ініціалізації Capability перемикача
cap_switch_data = caps_switch_initialize(ctx, "main", NULL, NULL);
if (cap_switch_data) {
// встановлюємо додаткові колбеки функції
cap_switch_data->cmd_on_usr_cb = cap_switch_cmd_cb;
cap_switch_data->cmd_off_usr_cb = cap_switch_cmd_cb;
// устанавливаем значение атрибута переключателя по умолчанию
cap_switch_data->set_switch_value(cap_switch_data,
caps_helper_switch.attr_switch.value_off);
}
// викликаємо функцію ініціалізації Capability показу температури
cap_temperature_data = caps_temperatureMeasurement_initialize(ctx, "main", NULL,
NULL);
if (cap_temperature_data) {
// встановлюємо значення та міру/unit температури за умовчанням
cap_temperature_data->set_temperature_unit(cap_temperature_data,
caps_helper_temperatureMeasurement.attr_temperature.unit_C);
cap_temperature_data->set_temperature_value(cap_temperature_data, 0);
}
// викликаємо функцію ініціалізації Capability вибору температури нагрівання
cap_heatingSetpoint_data = caps_thermostatHeatingSetpoint_initialize(ctx,
"main", NULL, NULL);
if (cap_heatingSetpoint_data) {
// встановлюємо додаткову колбек функцію
cap_heatingSetpoint_data->cmd_setHeatingSetpoint_usr_cb = cap_thermostat_cmd_cb;
// встановлюємо міру/unit температури нагрівання
cap_heatingSetpoint_data->set_unit(cap_heatingSetpoint_data,
caps_helper_thermostatHeatingSetpoint.attr_heatingSetpoint.unit_C);
}
}
```

```
void iot_gpio_init(void)
{
// esp sdk specific
gpio_config_t io_conf;
// відключаємо переривання
io_conf.intr_type = GPIO_INTR_DISABLE;
// встановлюємо режим піна на вихід
```

```
io_conf.mode = GPIO_MODE_OUTPUT;
// вибираємо пін
io_conf.pin_bit_mask = 1 << GPIO_OUTPUT_MAINLED;
// вибираємо для піна pull-down
io_conf.pull_down_en = 1;
io_conf.pull_up_en = 0;
// конфігуруємо пін вбудованого світлодіода
gpio_config(&io_conf);
// залишаємо ту ж конфігурацію, але застосуємо її для піна зумера
io_conf.pin_bit_mask = 1 << GPIO_OUTPUT_BUZZER;
// конфігуруємо цей пін
gpio_config(&io_conf);
// конфігуруємо піни rgb світлодіода
io_conf.pin_bit_mask = 1 << GPIO_OUTPUT_RGBLED_R;
gpio_config(&io_conf);
io_conf.pin_bit_mask = 1 << GPIO_OUTPUT_RGBLED_G;
gpio_config(&io_conf);
io_conf.pin_bit_mask = 1 << GPIO_OUTPUT_RGBLED_B;
gpio_config(&io_conf);
// конфігуруємо пін кнопки-перемикача
io_conf.intr_type = GPIO_INTR_ANYEDGE;
io_conf.mode = GPIO_MODE_INPUT;
io_conf.pin_bit_mask = 1 << GPIO_INPUT_SWITCH;
io_conf.pull_down_en = 0;
io_conf.pull_up_en = 1;
gpio_config(&io_conf);
// відключаємо сервіс переривань
gpio_install_isr_service(0);
// встановлюємо значення пінів за замовчуванням
gpio_set_level(GPIO_OUTPUT_MAINLED, LED_GPIO_ON);
gpio_set_level(GPIO_OUTPUT_BUZZER, LED_GPIO_OFF);
gpio_set_level(GPIO_OUTPUT_RGBLED_R, LED_GPIO_OFF);
gpio_set_level(GPIO_OUTPUT_RGBLED_G, LED_GPIO_OFF);
gpio_set_level(GPIO_OUTPUT_RGBLED_B, LED_GPIO_ON);
}
```