КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

До захисту допущено
Завідувач кафедри ІСТ
_____Олександр КУЧАНСЬКИЙ
(підпис) (ім'я, ПРІЗВИЩЕ)
"___"_____ 2022р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

спеціальності 126 «Інформаційні системи та технології»

освітньої програми «Програмні технології інтернет речей»

на тему: «Технологія створення датчиків вологості та рівнів pH в ґрунті із системою онлайн керування в мережі ZigBee»

Виконав: студент 4 курсу, групи IP-41
(шифр групи)

Андрій ШЕВЦОВ _____ _____
(Ім'я, ПРІЗВИЩЕ) (підпис)

Керівник кандидат технічних наук, доцент Сергій БРОНІН _ _____
(посада, науковий ступінь, вчене звання, Ім'я, ПРІЗВИЩЕ) (підпис)

Консультант нормо контроль _____ _____
(назва розділу) (посада, вчене звання, науковий ступінь, Ім'я, ПРІЗВИЩЕ) (підпис)

Рецензент доцент кафедри ІСТ ЧДТУ, к.т.н., Анаіт КАРАПЕТЯН _____
(посада, науковий ступінь, вчене звання, науковий ступінь, Ім'я, ПРІЗВИЩЕ) (підпис)

Засвідчую, що у пояснювальна записка не має запозичень з праць інших авторів без відповідних посилань.

Здобувач освіти __ __
(підпис)

Київ – 2022 року

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра Інформаційні системи та технології
Освітній рівень Бакалавр
Спеціальність 126 Інформаційні системи та технології
Освітня програма Програмні технології інтернет речей

ЗАТВЕРДЖУЮ
Завідувач кафедри,
професор
Олександр КУЧАНСЬКИЙ
_____
«__» ___ 2022 року

ЗАВДАННЯ
НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Здобувач освіти: Андрій ШЕВЦОВ

Група: ІР-41

1. **Тема кваліфікаційна робота бакалавра**: «Технологія створення датчиків вологості та рівнів pH в ґрунті із системою онлайн керування в мережі ZigBee». Затверджена протоколом засідання кафедри ІСТ №05/21_22 від 03.12.2021 року
2. **Строк подання студентом готової роботи** – «22» червня 2022 р.
3. **Вихідні дані до роботи**: дослідження в області автоматизації процесу аналізу стану ґрунту. Створення пристрою, який буде автоматично проводити знімок стану землі, в якій буде знаходитись, аналізуючи показники вологості та рівню pH за допомогою датчиків. Дані датчики будуть підключатися до контролеру і передавати показники в систему домашньої автоматизації і використовувати можливості ZigBee сервісів автоматизації.
4. **Зміст роботи**: РОЗДІЛ 1. АНАЛІЗ І ХАРАКТЕРИСТИКА СФЕРИ ЗАСТОСУВАННЯ СИСТЕМ КОНТРОЛЮ СТАНУ ҐРУНТУ. ПОСТАНОВКА ЗАДАЧІ (Опис об’єкту дослідження, проблематика визначення вологості ґрунту, аналіз pH рівню ґрунту, порівняння зі схожими технологіями); РОЗДІЛ 2. ПРОЕКТУВАННЯ СПАРЕНОГО ДАТЧИКУ ДЛЯ АНАЛІЗУ РІВНЮ ВОЛОГОСТІ ТА РІВНЮ РН ҐРУНТУ, СТВОЕРННЯ РЕАЛЬНОГО ПРОТОТИПУ ТА ЙОГО ПРОШИВКА (Проектування приладу, огляд та підбір

потрібного обладнання та зборка пристрою, створення прошивки для пристрою, прошивка мікроконтролеру, створення програматора для чіпів на базі esp8266, підключення пристрою та його прошивка); РОЗДІЛ 3. РЕАЛІЗАЦІЯ МЕРЕЖІ ZIGBEE ТА ЇЇ ІНТЕГРАЦІЯ В СЕРВІСИ ДОМАШНЬОЇ АВТОМАТИЗАЦІЇ (Огляд мережі передачі даних, підбір координатора для роботи в мережі, прошивка координатора, встановлення Home Assistant, MQTT брокера та інтеграція координатора, створення додаткового ZigBee маршрутизатора з 2 реле на сухих контактах для комутації постійного та змінного струму, та розширення загальної мережі ZigBee, фізичне підключення компонентів, прошивка пристрою, інтеграція в Home Assistant всіх наявних пристроїв, тестування мережі ZigBee та приладів всередині неї, тестування зв'язку, тестування підключення через проміжний маршрутизатор ZigBee, тестування показників, створення логіки роботи програми).

5. **Перелік графічного матеріалу**: електрична схема побудови приладу, фотографії готового виробу, приклади інших систем аналізу, модель роботи пристрою в мережі ZigBee, схема додаткового ZigBee маршрутизатора для розширення мережі, алгоритм роботи системи з датчиком та виконуючим пристроєм, алгоритм роботи телеграм-бота, діаграма діяльності, послідовності, класів, логічна модель бази даних, фізична модель бази даних, схема MVC роботи веб-додатку.

6. **Календарний план виконання роботи:**

Етапи виконання кваліфікаційної роботи бакалавра	**Термін виконання**	**Результат виконання**
1. Вибір тематики кваліфікаційної роботи бакалавра	01.09.2021-01.10.2021	виконано
2. Наказ про затвердження тем кваліфікаційної роботи бакалавра та призначення керівників	03.12.2021	виконано
3. Розробка плану кваліфікаційної роботи бакалавра і його погодження з керівником	25.12.2021	виконано
4. Написання I розділу кваліфікаційної роботи	19.03.2022	виконано
5. Написання II розділу кваліфікаційної роботи	25.04.2022	виконано
6. Написання III розділу кваліфікаційної роботи	29.04.2022	виконано
7. Підготовка висновків і пропозицій	30.04.2022	виконано
8. Попередній захист кваліфікаційної роботи	12.05.2022	виконано
9. Перевірка на плагіат	13.05.2022-15.06.2022	виконано

10. Нормоконтроль	02.06.2022- 06.06.2022	виконано
11. Рецензування кваліфікаційної роботи бакалавра і представлення роботи на кафедру в друкованому вигляді	15.06.2022	виконано
12. Захист кваліфікаційної роботи бакалавра	23.06.2021- 24.06.2021	

Дата видачі завдання «__» _____ 2022 р.

Керівник роботи: к. т. н., доцент Сергій БРОНІН _____ (підпис)

Завдання прийняв до виконання:

Здобувач освіти на освітньому рівні «бакалавр» 4-го курсу групи ІР-41

Андрій ШЕВЦОВ _____

(Власне Ім’я, ПРІЗВИЩЕ) (підпис)

АНОТАЦІЯ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра Інформаційних систем та технологій

Освітня програма «Програмні технології інтернет речей»

Кваліфікаційна робота бакалавра Андрія ШЕВЦОВА

Тема роботи: «Технологія створення датчиків вологості та рівнів pH в ґрунті із системою онлайн керування в мережі ZigBee».

Мета кваліфікаційної роботи бакалавра – створення пристрою для роботи в мережі ZigBee, який буде легко інтегруватись в сервіси автоматизації.

Об'єкт дослідження – прилад, який буде створено та інтегровано в систему домашньої автоматизації.

Предмет дослідження – впровадження нових IoT систем в сферу агрокультур.

Кваліфікаційна робота бакалавра складається зі змісту, вступу, основної частини, яка включає три розділи, висновків та списку використаних джерел. Всього 98 сторінок, 47 рисунків, 2 таблиці.

КЛЮЧОВІ СЛОВА: IoT, ZigBee, проектування, електрична схема, датчик, вологість ґрунту, cc2530, ESP8266, MQTT, ZigBee2MQTT, EasyEda, рівень pH ґрунту, мікроконтролер, автоматизація, PTVO, прошивка, мережа.

ABSTRACT

TARAS SHEVCHENKO NATIONAL UNIVERSITY OF KYIV

Faculty of Information Technologies

Department of Information Systems and Technologies

Educational Program "Software Technologies of the Internet of Things"

Qualification work of master Andrey SHEVTSOV.

Work topic: "Technology of creation of moisture and pH sensors in the soil with the system of online control in the ZigBee network".

The goal of the bachelor's qualification work is to create a device for working in the ZigBee network, which can be easily integrated into automation services.

The subject of the study is a device that will be created and integrated into the system of home automation.

The subject of the research is the implementation of new IT systems in the sphere of agricultural crops.

Bachelor's qualification work consists of the content, the introduction, the main part, which includes three sections, conclusions and the list of references. A total of 98 pages, 47 figures, 2 tables.

KEYWORDS: IoT, ZigBee, design, electric circuit, sensor, soil moisture, cc2530, ESP8266, MQTT, ZigBee2MQTT, EasyEda, pH level of soil, microcontroller, automation, PTVO, firmware, measure.

ЗМІСТ

ВСТУП

Сфера сільського господарства, впродовж століть розвитку, залишається однією з найголовніших у світі. Рослини, які зростають у ґрунті, напряму залежать від його стану. Головними аспектами, які характеризують стан ґрунту, є: відносна вологість та рівень вмісту кислот (рівень pH[]). Вологість ґрунту – головна умова, яка забезпечує правильний ріст культур і збільшення об’ємів врожайності. Рівень pH - це показник, який обумовлений наявністю в ґрунтових розчинах водневих іонів, що напряму впливає на активність у ґрунті елементів живлення і їхнє засвоєння рослинами. Зазвичай виміри цих показників проводились вручну, проходячи з приладами для вимірювання сектор за сектором. Це доставляє незручності для людини, яка здійснює дані заміри, є не дуже ефективним з огляду на час, який затрачується на аналіз кожного сектору і зняття показників може відбутися запізно, що негативно вплине на рослини. Саме для оптимізації аналізу ґрунту і було створено спарений датчик вологості ґрунту та рівню pH, щоб без постійного втручання людини проводити знімок стану ґрунту і передавати дані до потрібної системи автоматизації, наприклад: системи автоматичного поливу та системи автоматичного живлення рослин. Даний прилад буде використовувати інноваційну бездротову мережу передачі даних ZigBee, що дасть змогу збільшити одночасну кількість пристроїв в мережі та дозволить датчику працювати декілька років від 2 пальчикових батарейок.

Метою даної роботи є створення розумного приладу для аналізу вологості та рівню pH ґрунту, а також дослідження обладнання, яке використовується для аналізу стану ґрунту та для роботи в мережі ZigBee.

Об’єктом дослідження є прилад, який буде інтегровано в систему домашньої автоматизації.

Предметом дослідження даної роботи виступає впровадження нових IoT рішень в сферу агрокультур.

Методами дослідження є аналіз існуючих рішень автоматичного контролю стану ґрунту проектування та створення приладу для аналізу вологості та рівню pH ґрунту, інтеграції в систему домашньої автоматизації, проведення експериментів із готовою системою.

РОЗДІЛ 1. АНАЛІЗ І ХАРАКТЕРИСТИКА СФЕРИ ЗАСТОСУВАННЯ СИСТЕМ КОНТРОЛЮ СТАНУ ҐРУНТУ

1.1. Опис об’єкту дослідження

Одне з найголовніших досягнень сьогодення є бездротова передача інформації[1]. Даний метод передачі даних дав змогу значно збільшити можливості Інтернету речей, через що даний сектор почав стрімко зростати. Застосування бездротової передачі позбавляє небхідності прокладки кабелю між приладами, що дає змогу таким приладам не залежати від навколишнього середовища і працювати автономно.

Тому стало питання обрання протоколу для бездротової передачі даних, який не буде потребувати великих потужностей і буде енергоефективним, для автономної роботи без підзарядки на протязі декількох років.

В ролі технології для бездротової передачі було обрано проткол ZigBee[2]. ZigBee — це відкритий стандарт бездротової мережі, який підпадає під класифікацію бездротової персональної мережі (WPAN). Це стандарт для mesh[3] мереж, що дозволяє самостійно конфігурувати мережі, простота встановлення та налаштування. Він призначений для застосування з низьким енергоспоживанням, особливо для тих, які розраховані на дуже тривалий термін служби акумулятора. Він має низьку швидкість передачі даних 20-250 Кбіт/с, залежно від діапазону. Протокол розроблено з урахуванням безпеки.

Zigbee пристрої взаємодіють між собою в екосистемі розумного будинку, що використовують топологію осередкової мережі. Пристрої знаходять активні девайси на його околицях, ініціюють зв'язок без переривань.

З урахуванням можливотей проктолу ZigBee були розроблені та створені прилади, які використовуються в різних аспектах життєдіяльності людини. Головною сферою їх застосування є домашня автоматизація[4]. За допомогою даних приладів можна покращити звичні побутові речі, такі як:

- Віддалене керування освітленням за допомогою телефона або голосом.
- Аналіз стану приміщення за допомогою датчиків (датчики аналізу диму, води, бездротовий дифавтомат з енергомоніторингом).
- Передача показників виконуючим пристроям, такі як клапани для подачі води, чи кондиціонери та інше.

При побудові ZigBee мережі може бути використана технологія Wi-Fi. Так, для зв’язку датчиків з іншими приладами ZigBee та керування ними використовується координатор, який має бути встановлений на комп'ютер з доступом до маршрутизатора (зв'язок між комп'ютером та маршрутизатором може бути по LAN кабелю чи технології Wi-Fi).

Головним є те, що датчики додаються до сервісів домашньої автоматизації, де можна з іншого кінця світу керувати всіма підключеними приладами.

1.1.1. Проблематика визначення вологості ґрунту

Вимірювання вологості ґрунту є ключовим питанням для моніторингу та контролю розвитку сільськогосподарських культур за допомогою технології IoT. Для захисту крихкого середовища посушливих зон, що залежить від підземних вод, важливо стежити за вологістю ґрунту та випаровуванням ґрунтових вод.

Значення, яке вимірює ємкісний датчик вологості ґрунту - відсоток води в ґрунті відносно об’єму . Для його розрахунку датчик вимірює в «сфері впливу» навколо нього, припустимо, що це 1 м³. Якщо результат VWC = 2%, це означає, що з цього кубічного метра ґрунту 2% становить чиста вода. Тобто 2% від 1000 л, що еквівалентно 20 л. Ця вода, очевидно, розподіляється з самим ґрунтом, зволожуючи його. Це означає, що якби ви могли віджати ці 1000 літрів ґрунту, вийшло б 20 літрів чистої води.

Для виміру показників вологості датчик треба занурити повністю в воду, щоб активна зона вимірювання не контактувала з повітрям. Після занурення датчику на нього треба подати напругу, після чого треба зчитати напругу на контакті виходу. Показник напруги на виході і буде показником датчика вологості.

1.1.2. Аналіз pH рівню ґрунту

Концентрація іонів водню в ґрунті називається pH[5] і залежить від хімічних реакцій між компонентами ґрунту та водою. На pH ґрунту впливають різноманітні комбінації позитивно заряджених іонів (натрій, калій, магній, кальцій, алюміній, марганець і залізо) і негативно заряджених іонів (сульфат, хлорид, бікарбонат і карбонат). PH ґрунту безпосередньо впливає на концентрацію основних поживних речовин і форми мікроелементів, доступних для поглинання рослинами, і може призвести до дефіциту або токсичності. На рисунку нижче (рис 1.1) наведено шкалу вимірювання рівню pH і те, як вона впливає на рослини.

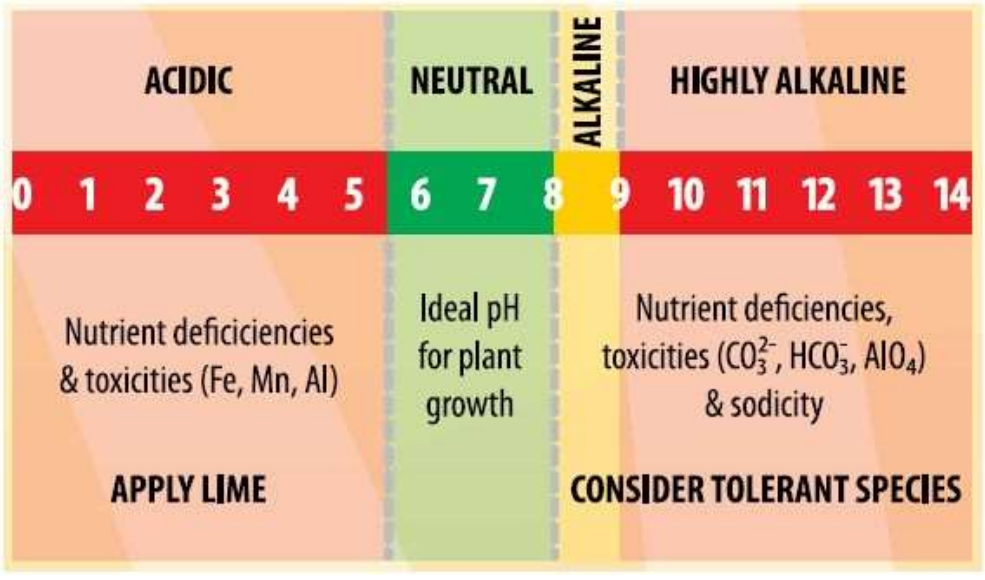


Рисунок 1.1 – шкала рівню pH

Вимірювання pH в ґрунтових системах створюють унікальні проблеми з точки зору інтерпретації отриманих значень. Принципи системи скляних/каломельних електродів обговорюються як фон для вимірювань pH

як у чистих розчинах, так і в ґрунтах. Детально обговорюється вплив потенціалу рідкого спаю та концентрації солі на значення pH суспензій ґрунтової води, з чого випливає, що поточна практика вимірювання pH ґрунту в перемішаних суспензіях ґрунту, ймовірно, призведе до найбільших помилок. Це пов’язано з великим потенціалом рідинного переходу невизначеної величини, що розвивається в таких системах.

1.2. Типи зв’язку у системах розумного аналізу ґрунту

У наш час вже створено багато різноманітних систем, які слідкують за показниками довкілля. Так для створення системи розумного аналізу грунту можна використовувати будь-яку мережу передачі даних. Кожна з наявних технологій передачі інформації несе у собі як і плюси, так і мінуси.

Для детального заглиблення у тему створення ZigBee пристроїв для розумного будинку треба спочатку розібратися в усіх методах доставки інформації. Адже передача інформації у просторі хоч і легка, на сьогодні, задача, але вимагає прискіпливої уваги, бо від обраної технології буде залежати складність розгортання повноцінної автоматизованої мережі, остаточка вартість, рівень складності експлуатації та інші фактори, які впливають на якість остаточного середовища IoT.

Всього, при виконанні дипломної роботи, було розглянуто 3 інші технології передачі інформації. Перші 2 – бездротові, а саме: Bluetooth та Wi-Fi, а третя це повне дротове з’єднання.

Так зі згаданих технологій виділяється провідне з’єднання пристроїв між собою. Даний метод характеризується низькою ефективністю, бо всі датчики, які будуть встановлені, треба живити та знімати з них показники, кожний датчик вимагає щонайменше 3 проводи. Провідні системи автоматизації тяжко масштабовані, бо вже встановлене обладнання треба буде демонтувати і проводити прокладання нових кабелів. Але, варто зазначити, що головним плюсом даної системи є відмовостійкість. Можна бути впевненим

що жоден з датчиків не втратить зв’язок з контролером або просто не розрядиться. Так, якщо вибір все ж таки пав на повне дротове з’єднання, то треба брати до уваги всі недоліки даного методу і що змінювати конфігурацію потім буде занадто складно.

Технологія Wi-Fi[6] є сьогодні найпопулярнішою, бо вона використовується в домашній локальній мережі для зв’язку пристроїв між собою. Головними плюсами даної технології є те, що швидкість обміну даними постійно наближується до показників дротового з’єднання, але при цьому пристрої можуть знаходись відносно віддалено один від одного. Головним плюсом даної технології є її розповсюдження, стандартизація та постійна оптимізація. Але за всі роки досі не було вирішено недолік, який несе в собі технологія Wi-Fi, це енергозатрати при передачі інформації. Дані затрати не суттєві якщо прилад підключено до мережі, але якщо живлення відбувається від акумулятора, то час життя такого пристрою буде не значним.

Тут і з’являється остання технологія - Bluetooth, яка була розглянута для порівняння із ZigBee і всіма іншими протоколами. Дана технологія являє собою щось середнє між технологію Wi-Fi та ZigBee. Вона характеризується не такими значними втратами енергії під час передачі даних, але при цьому швидкість і покриття не порівняти із технологією Wi-Fi. Bluetooth за останні роки отримав багато оновлень, такі як енергозберігаючий режим роботи та покращення зв’язку. Даний протокол в основному використовується для зв’язку із носимими пристроями, а також було помічено невелику кількість датчиків для домашньої автоматизації з мережею передачі Bluetooth, в основному в сфері систем розумного захисту, де кінцеві пристрої з Bluetooth живуть дуже довго через непостійність їх використання.

1.3. Огляд та аналіз подібних продуктів в даній сфері з іншими способами передачі інформації

Для чіткої постанови задачі дипломної треба розглянути рішення, які використовують схожі методи передачі інформації або алгоритми, які будуть корисними при проектуванні власного пристрою. З кожної системи можна підкреслити, що і як було зроблено неправильно і покращити це.

Порівняння з аналогами також дає розуміння, чого не вистачає на ринку автоматизації і що потребує покращення.

1.3.1. Wi-Fi датчик температури

Датчики температури Wi-Fi вимірюють температуру навколишнього середовища, бездротово надсилаючи цю інформацію в Інтернет через з'єднання WI-FI. Залежно від типу датчика температури дані можна переглядати на ПК за допомогою програмного забезпечення, мобільного додатка, комп'ютерного браузера або будь-якої комбінації цих трьох.

Використання та вимоги до бездротових датчиків температури різняться, як і тип пристроїв. Датчики температури Wi-Fi варіюються від простих, бюджетних пристроїв до професійних високоточних пристроїв.

Рисунок 2.2 – Wi-Fi температури повітря

Пристрої даної категорії (рис 1.2) можуть напряму підключатися до смартфонів, або підключатись до маршрутизатора і бути налаштованими для роботи у системі.

Wi-Fi є і залишається одним з лідерів серед інших протоколів. Але проблеми з його електроспоживанням дуже критичні, бо це найважчий серед протоколів бездротових мереж.

1.3.2. Дротові системи автоматизації

Для прикладу було обрані провідні системи KNX. Система автоматизації KNX є найнадійнішою системою автоматизації у світі. Побудована за стандартом KNX , вона вирішує головне завдання – скорочує витрати на електроенергію. Основні функції системи – це управління освітленням та функціями комфорту в автоматичному та ручному режимі.

В основі побудови системи лежить шина KNX. Завдання, які вона виконує, пов'язані з передачею команд від датчиків, сенсорів, вимикачів кнопок на виконавчі пристрої (актуатори) SpaceLink і далі на системи кондиціювання, вентиляції, опалення та освітлення.

Рисунок 1.3 – KNX система автоматизації

Дане рішення, зазначене на (рис 1.3) є еталоном в дротових системах автоматизації, але монтаж загальної системи вимагає занадто великих зусиль. Також слід зазначити, що телекомунікаційне обладнання, яке використовується при побудові рішення KNX, треба десь зберігати і відвести під це треба маленьке приміщення. Тому дана технологія стає менш популярною, бо складність створення системи не виправдовує кінцевого результату.

1.3.3. Bluetooth датчик вологості повітря

Для аналізу пристроїв, заснованих на технології передачі Bluetooth, було обрано датчик температури та вологості повітря Trax10219. Даний пристрій працює під протоколом Bluetooth Low Energy, що є наступною еволюційною сходинкою звичайного протоколу Bluetooth.

Trax10219 датчик якості повітря BLE, призначений для моніторингу клімату в приміщенні, пристрій вимірює якість повітря на основі концентрації летких органічних сполук і розраховує IAQ (якість повітря в приміщенні), що представляє якість повітря в приміщенні на основі запатентованого алгоритму Bosch.

Дані рішення добре підходять для аналізу стану приміщення, але життя таких датчиків від одного заряду не перевищує рік при постійній експлуатації.

1.3.4. Порівняння розглянутих рішень з ZigBee технологією

Нижче наведено таблицю порівняння між протоколами та технологіями передачі даних. Порівняння буде включати: частотні діапазони роботи, пропускна спроможність, розмір стеку повідомлення в кожного з протоколів, час неперервної роботи від акумулятор, максимальна кількість вузлів в мережі, діапазон дії мережі, область застосування а також складність монтажу повної системи автоматизації.

Таблиця 3.1 – Порівняння пристроїв з різними технологіями передачі даних

Технологія передачі даних	**ZigBee**	**Wi-Fi**	**Bluetoth**	**Дротове з'єднання**
1	2	3	4	5
Частотний діапазон, ГГц	2,4 – 2,483	2,4 – 2,483	2,4 – 2,483	220v – 59 Гц LAN – 100-500 МГц
Пропускна спроможність, кбіт/с	250	11 000	723	100 000 та більше
Розмір стеку протокола, кбайт	32-64	Більше 1000	Більше 200	Більше 1000
Час неперервної роботи от акумулятора однієї ємності, дні.	100-1000	0,5-5	1-10	Підключено постійно до мережі
Максимальна кількість вузлів в мережі	65 536	10	7	Необмежено
Діапазон дії, м	10-100	20-300	10-100	100
Область застосування	Віддалений моніторинг та керування	Передача мультимедійної інформації	Заміщення провідного з'єднання	Використання там, де потрібна найбільша відмовостійкість

Продовження таблиці зі сторінки 13				
Складність установки та налаштування від 1 до 5 (де 1 – найнижче, 5 – максимально складно)	3	1	2	5

Висновок до розділу 1

В даному розділі було підняте питання описання об’єкту. Було розглянуто тему вимірювання вологості та рівню pH грунту, а також їх проблематика. Також було розглянуто інші протоколи та технології передачі інформації. Було виведено плюси та мінуси кожної з технологій. Для покращення розуміння теми автоматизації та створення нових приладів було розглянуто схожі рішення на ринку, які використовують інші протоколи для відправки даних про стан датчиків. Так, ZigBee протокол є найслабшим з усіх якщо дивитися на пропускну спроможність чи розмір повідомлення. Але дана технологія виграє тим, що є найефективнішою серед інших. Так, усіх можливостей ZigBee протоколу вистачає щоб організувати повноцінну mesh-мережу та передавати дані про стан різних датчиків. Головна перевага ZigBee – це енергоефективність, що дає змогу простому датчику з 2 батарейками працювати декілька років без підживлення.

РОЗДІЛ 2. ПРОЕКТУВАННЯ СПАРЕНОГО ДАТЧИКУ ДЛЯ АНАЛІЗУ РІВНЮ ВОЛОГОСТІ ТА РІВНЮ РН ҐРУНТУ, СТВОЕРННЯ РЕАЛЬНОГО ПРОТОТИПУ ТА ЙОГО ПРОШИВКА

Основна частина дипломної роботи це технологія створення пристрою, який повинен містити датчики вологості та рівню pH в ґрунті. Технологія створення повинна включати в себе саме проектування приладу в середовищі для побудови електричних схем, після чого повинен проводитись підбір обладнання, яке в подальшому спаюється на друкованій платі текстоліту або з'єднується проводами, після побудови пристрою треба обрати спосіб прошивки, провести запис інформації на чіп та можна переходити к його інтеграції в сервіси домашньої автоматизації.

2.1. Проектування приладу

Процес побудови нового електронного пристрою завжди складатися з кількох кроків, перерахованих нижче:

- Етап ідеї. Записуються ідеї щодо технічних вимог.
- Етап дослідження. Проводиться пошук шляхів досягнення цілей і прийняття ключових рішень для виконуваного проекту.
- Етап розробки схеми. розробка електронної схеми. Вибір комплектуючих відповідно до технічних вимог.
- Етап налагодження та розрахунку. Використовуйте віртуальне середовище для моделювання схем і налагодження. Розрахувати оптимальні параметри для кожного компонента.
- Етап побудови самого пристрою. Використовується фабричний монтаж за замовленням або можна провести все вручну з куплених компонентів.

Перед самим проектуванням приладу треба прикинути які компоненти будуть використані і намалювати їх логічне з'єднання в будь-якому графічному редакторі. Дану схему було побудовано на листку, де схематично зображено розміщення компонентів і їх зв'язок між собою, тут розведено головний контролер, який з'єднується з датчиками, елементом живлення,

кнопками керування та світлодіодами для сповіщення. На даній схемі зображено які пристрої передають інформації на вхід і куди передає контролер інформацію на вихід.

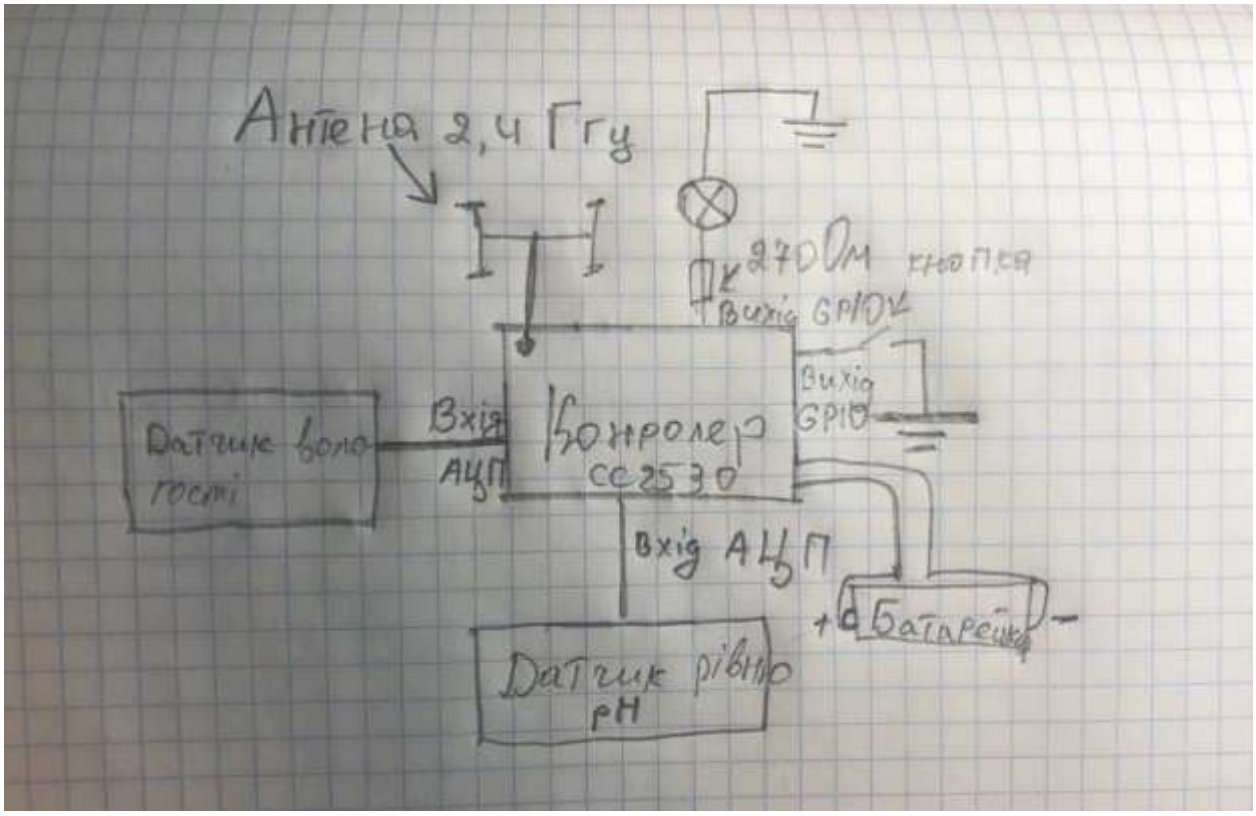


Рисунок 2.1 – Ескіз пристрою на бумазі

На схемі (рис 2.1) зображено логічне з'єднання компонентів, яке в подальшому буде використано для проектування у специфікованому середовищі.

Для проектування приладу було обрано веб-орієнтоване середовище автоматизації проектування електроніки EasyEda[9], в якому за допомогою редактор принципових схем отримується електрична конфігурація, з якою можна замовити виготовлення готових друкованих плат, або виготовити пристрій за допомогою підключення через макетну плату, на якій розводяться потрібні компоненти.

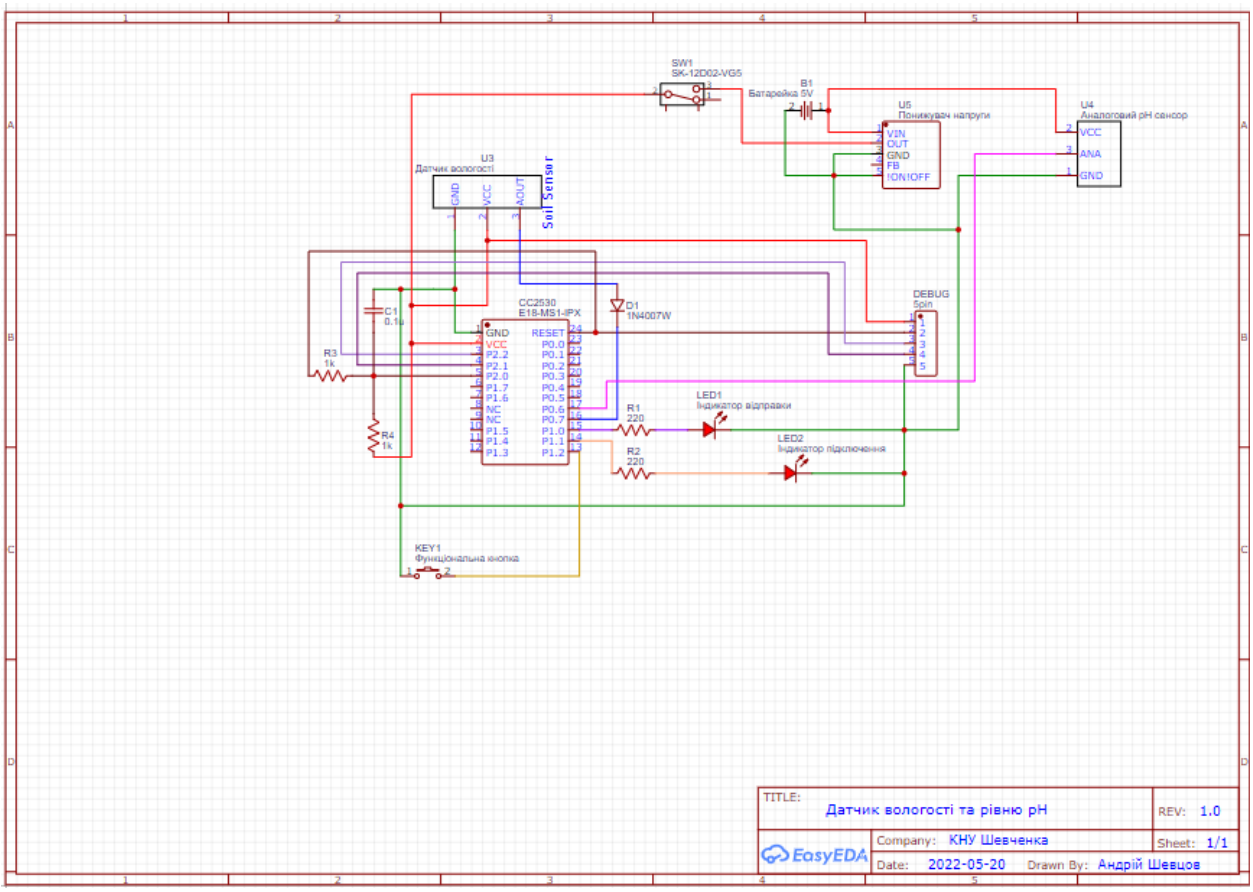


Рисунок 2.2 – Електрична схема спареного датчику вологості та рівню pH грунту

На даній схемі (рис 2.2) зображене послідовне з’єднання компонентів. Головним центром пристрою є чіп з вбудованим модулем радіопередачі. Присутні 2 світлодіоди для індикації стану: перший має бути індикатором підключення пристрою та відправки звітів, другий буде налаштовуваним, тому його можна використовувати щоб повідомляти про збої, чи коли низький заряд акумулятора. Для прошивки також відведено спеціальний роз’єм, за яким прилад буде підключатися до програматора, роз’єм цей містить вихід для скидання (Reset), RX та TX послідовної передачі інформації, а також живлення і мінус. Для роботи датчиків буде використано 3 стандартні батарейки, з напругою на виході кожною по 1,5 Вольта кожна, які в сумі дадуть напругу 4,5 Вольта. Дана напруга буде оптимальна для роботи датчиків, але занадто велика для контролеру, тому треба провести понижувач напруги, який змінить струм з 4,5 Воль до 3,3 Вольт, що є стандартом для сигналів, які

використовуються в електроніці. Також було вирішено додати порт, який через паралельне підключення через конденсатор та резистор до землі та живлення буде заведений до порту скидання (Reset), який в подальшому може бути використаний для віддаленого скидання пристрою в будь-який час.

Схема, в середовищі EasyEda перевірена в режимі симуляції на можливі замикання чи неправильні підключення. Проблем з логікою виявлено не було, тому було можна переходити до підбору обладнання та подальшої побудови пристрою.

2.2. Огляд та підбір потрібного обладнання та зборка пристрою

Для вирішення задачі побудови пристрою, який буде вимірювати показники вологості та рівень pH грунту, треба правильно обрати обладнання, яке буде задовольняти вимогам для приладів в мережі домашньої автоматизації. Головне, що треба обрати для подальшої побудови - це мікроконтролер, який виступає центром керування приладу. Для виконання дипломної роботи треба обрати чіп, який буде мати змогу передавати дані в мережі ZigBee, мати внутрішню пам’ять і порти для фізичного зв’язку з датчиками.

Першочергово було розглянуто чіпи серії CC253x, які можуть виконати всі потрібні функції і повністю задовольняють потребам для виконання дипломної роботи. Було розглянуто 3 чіпи CC253x:

- CC2530 - це справжнє рішення системи на чіпі для додатків IEEE 802.15.4, Zigbee та RF4CE. Це дозволяє будувати надійні мережеві вузли з дуже низькими загальними витратами на розрахунки. CC2530 поєднує в собі чудову продуктивність провідного радіочастотного трансивера з стандартним удосконаленим мікроконтролером 8051, внутрішньосистемною програмованою флеш-пам'яттю, 8-КБ оперативної пам'яті та багатьма іншими потужними функціями.
- CC2531 – це готове рішення з USB виходом для прямого підключення до комп'ютера. Взагалі даний чіп корисний для звичайної конфігурації у

якості координатора чи маршрутизатора. Тому його в основному обирають в якості проміжного пристрою.

- CC2538 – майже таке саме рішення, як і cc2530, але з процесором Cortex-M3 та 32Кб пам'яті. Має підсилювач cc2592, який розпаяний з головним чіпом на одній платі і дає підсилення для антени в районі 16 dbm.

Для використання у дипломній роботі було обрано чіп cc2530 (рис 2.3), він буде центром збору і відправки даних у мережу ZigBee. Сам модуль називається E18-MS1-PCB. До даного модуля є можливість підключення зовнішньої антени з підсилювачем, що значно розширить покриття мережі ZigBee.

Рисунок 2.3 – Чіп cc2530 для роботи в мережі ZigBee

Для керування пристроєм було додано тумблер включення/виключення (рис 2.4), а також функціональну кнопку, яку можна налаштувати при створені сценаріїв роботи системи.

Рисунок 2.4 – функціональні кнопки

Для вимірювання вологи було обрано готовий ємкісний датчик (рис 2.5), який за допомогою ємкісного методу вказує на зміну в кількості води навколо самого датчику.

Рисунок 2.5 – ємкісний датчик вологості

В якості сенсору для вимірювання показнику pH було обрано готове рішення, яке включає аналоговий перетворювач та вимірювальний щуп, який занурюється у землю (рис 2.6).

Рисунок 2.6 – аналоговий pH метр

Всі датчики використовують АЦП (аналогово цифрове пертворення) і прцюють на логічному рівні 3,3 Вольта, що одразу робить це сумісним із системою керування на базі чіпу cc2530.

Для індикації стану приладу було додано пару світлодіодів. Обидва є стандартними в електроніці, лише потребують резистор між ними та током, щоб вони не згоріли.

Для монтажу використано проектувальну плату (рис 2.7), на яку можна наносити компоненти і припаювати один до одного.

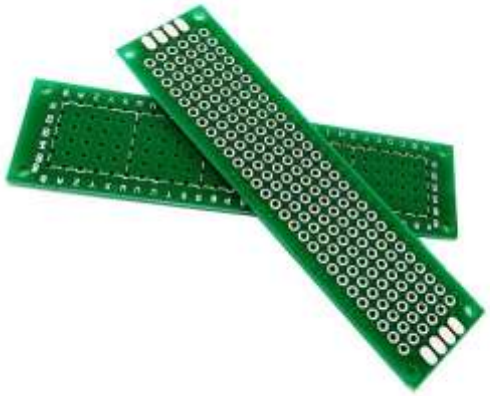


Рисунок 2.7 – макетна плата для монтажу електричних компонентів

В якості живлення для датчику було обрано варіант з двома змінними батарейками по 1,5 Вольт кожна (рис 2.8), які в сумі дають потрібну для живлення напругу 3 Вольти.

Рисунок 2.8 – елемент живлення 1,5 Вольт постійної напруги

2.3. Створення прошивки для пристрою

Прошивка – це набір команд та правил, за якою пристрій буде діяти та виконувати потрібні функції. Прошивка пристрою дозволяє наділити його новим функціоналом, якого раніше могло і не бути. Для створення прошивки для пристроїв на чіпах CC253x зазвичай використовують додаток PTVO, який

має основне вікно налаштувань, де зображено порті входу та виходу, поряд з якими є тумблер стану роботи, режим роботи, фізичний вихід на мікроконтролері, варіанти підключення до інших портів і зв’язування між ними.

Прошивка пристрою, створена в додатку PTVO, наведена на рисунку нижче (рис 2.9). На даному зображенні присутні ті параметри, які знадобляться при роботі з пристроєм.

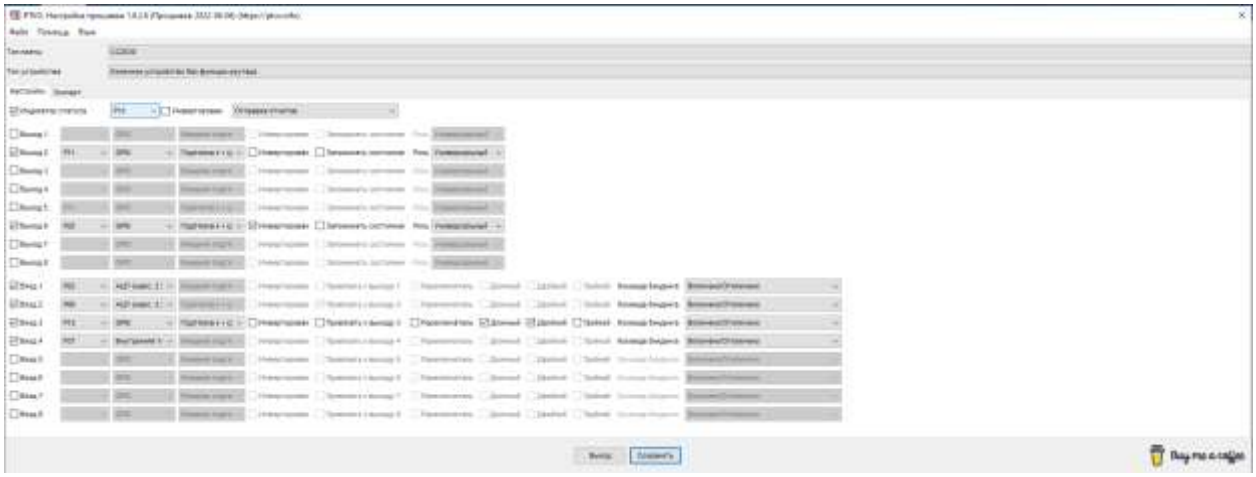


Рисунок 2.9 – приклад конфігурації прошивки в середовищі PTVO

Першочергово було вказано номер чіпу, який далі буде запрограмовано та тип пристрою. Всього в ZigBee існує 3 типи пристроїв:

Координатор – головний пристрій в мережі ZigBee. Він фізично підключається до комп’ютеру, де встановлено сервіс домашньої автоматизації та через MQTT брокера передає дані напряму в середину Home Assistant;

Маршрутизатор – пристрій, який може виконувати функції та взагалі працювати як кінцевий прилад, але буде використовувати додаткові можливості протоколу ZigBee, а саме можливість Mesh-мережі[], в якій маршрутизатори та кінцеві пристрої можуть самоорганізовуватись та змінювати маршрут трансляції повідомлення. Тобто кінцеві пристрої з датчиками можуть бути підключені до координатора через один чи декілька

маршрутизаторів ZigBee, щоб збільшити радіус дії мережі і гарантувати зону покриття пристрою.

Кінцевий пристрій без функції маршрутизатора – це вже виконуючий пристрій, який може лише підключитися до координатора чи маршрутизатора, та лише виконує дії, які закладено в прошивці. Кінцеві пристрої характеризуються малим електроживленням, яке потрібне лише для живлення датчиків під час опитування та для передачі повідомлення до контролюючого пристрою.

Для вирішення задачі дипломної роботи було створено конфігурацію, яка буде містити 2 порти вимірювання аналогового сигналу, який приходить від датчику вологості та рівню pH. Рівень сигналу для вимірювання сигналу АЦП складає 3,3 Вольта, що робить вимірювання показників датчиків не складною задачею.

Далі налаштовуємо інші порти для налаштувань роботи керувальної кнопки та індикації роботи пристрою за допомогою світлодіодів. Для роботи світлодіодів та функціональної кнопки було використано режим роботи GPIO.

GPIO - це абревіатура від General Purpose Input / Output, тобто це порти вводу/виводу, які налаштовуються на виконання різних функцій, отже, вони загального призначення, а не для конкретного використання. При обранні режиму роботи порту GPIO, треба вказувати за якою логікою створено фізичне з’єднання на платі. Існує 3 варіанти встановлення компонентів, які включають підтягування влаштованих резисторів або використовують опускання:

- Pull-up – дана функція включає внутрішній підтягуючий резистор. Для входів GPIO це означає, що натиснутим станом є сигнал LOW (підключений до землі).
- Pull-down – для входів GPIO це означає, що натиснутим станом є сигнал HIGH (підключений до +3,3 В через резистор 4700 Ом). Ви не можете

змішувати внутрішні резистори підтягування або опускання на одному порту.

- Зовнішній підтягуючий резистор – схожий на внутрішній підтягуючий резистор, але його слід додати зовні на друковану плату. Це головний недолік. Головна перевага, ви можете змішувати різні типи підтягування або опускання на одному порту. Типове значення резистора 4700 Ом.

Вибір налаштування залежить від вашого та використовуваного зовнішнього датчика чи іншої периферії. Для входів GPIO я віддаю перевагу «Pull-up», це дозволяє підключати кнопки, які заведені на землю і при натисканні її відбувається підтягування резистора і контролер визначає, що на порту присутній сигнал LOW, що і свідчить про натискання. Для зовнішніх датчиків рекомендовані схеми можна знайти в паспорті датчика.

Для індикації роботи пристрою було налаштовано 2 світлодіоди: перший для індикації підключення/відключення пристрою до координатора та сигналізує про відправку пакетів до пристрою керування, другий є налаштовуваним світлодіодом, алгоритм роботи якого налаштовується всередині середовища Home Assistant.

Далі при створені прошивки треба перейти до вікна «Експерт», яке включає всю базову інформацію про пристрій: виробник, ідентифікатор виробника, ідентифікатор моделі, ключ безпеки мережі (якщо потребується, захист налаштовується спочатку на координаторах та маршрутизаторах ZigBee, а потім вже ключ безпеки додається до прошивки кінцевого пристрою), інтервал відправки звітів (вказується в секундах), посилання на зображення пристрою та варіанти роботи та відправки сервісних повідомлень.

Варіант налаштування вікна «Експерт» в додатку PTVO наведено нижче [рис 2.10].

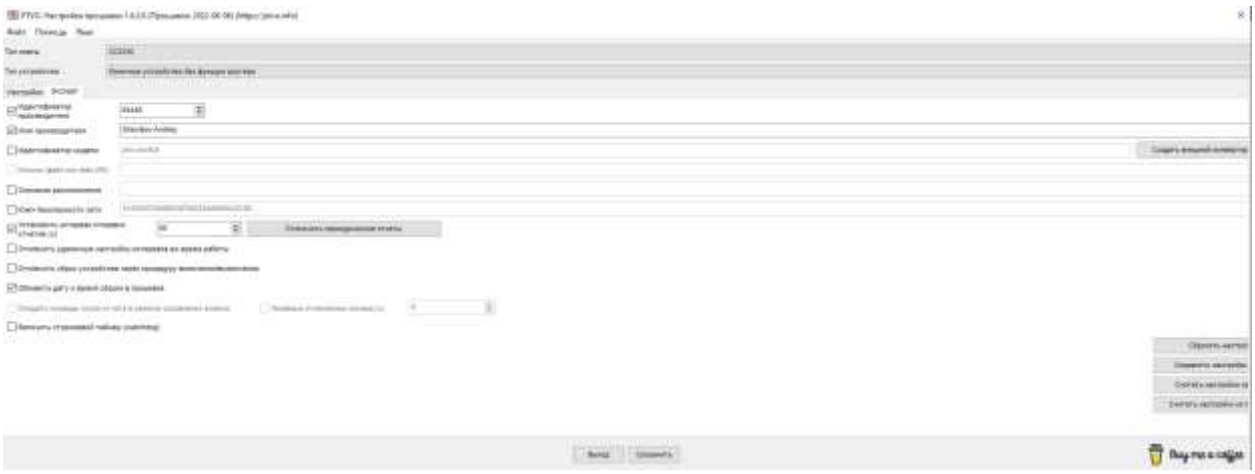


Рисунок 2.10 – налаштування паспортних даних пристрою

В варіанті прошивки для пристрою, розроблено в ході виконання бакалаврської роботи, вказано моє ім’я та прізвище в графі виробника, встановлено 60 секунд очікування між відправками повідомлень. Ключ безпеки мережі не було використано, бо пристрої працюють за дозволами з Home Assistant, тобто це значить що підключитися інший пристрій не зможе, якщо не буде встановлений дозвіл на нові підключення.

2.4. Прошивка мікроконтролеру

Для самого процесу прошивки чіпу CC2530 можна використовувати один з трьох варіантів завантаження коду в регістри пам’яті. Кожен з варіантів містить як і плюси, так і мінуси. Далі буде описано кожен з варіантів та зазначено, чому обрано окремі з ним під час виконання дипломної роботи:

Прошивка через офіційний CC Debugger від виробника чіпів ZigBee Texas Instruments. Даний пристрій чіпляє своєю простотою виконання процесу завантаження файлу до пам’яті контролера, де лише треба підключити контакти приладу до контактів контролеру, відкрити офіційний додаток, обрати файл прошивки у форматі hex, та почати процес завантаження. Мінусами даного методу є ціна виробу, бо офіційна конфігурація від Texas Instruments коштує декілька сотень доларів, а копія від китайських виробників коштує близько ста доларів. Тому даний пристрій краще обирати для підприємства, де він зможе покращити та прискорити сам процес прошивки,

але при домашньому використанні не потребується, бо існують інші способи перепрошивки.

Прошивка через контролер Raspberry pi, Arduino, NodeMcu. Даний спосіб майже такий же простий, як і прошивка через офіційний дебагер, але тут складність виникає лише в тому, щоб налаштувати контролер як пристрій для завантаження нового програмного забезпечення. Для цього контролер підключається по послідовному порту (UART) до комп'ютера, далі завантажується скетч, який буде містити конфігурацію для подальшої роботи. Далі треба фізично підключити пристрій для перепрошивки, завантаження файлу і початок прошивки через додаткові програмні рішення та командну строку. Даний спосіб краще обирати для присроїв, які раніше не піддавалися процесу прошивки. Його було обрано для налаштування роботи кінцевих пристроїв. Процес повної прошивки буде описано нижче, з вказанням портів підключення та всіх дій.

Прошивка через асинхронний послідовний порт та додаток SerialBootTool. Даний спосіб краще використовувати для роботи з пристроями, які містять USB з'єднання чи просте підключення через порти обміну інформації по портам RX та TX, які нагадують процес передачі інформації в радіомовленні, де TX - Tracieve (відправлення), RX - Recieve (отримання). Підключення RX та TX відбувається хрест нахрест з іншим пристроєм, тобто RX пристрою 1 підключається TX пристрою 2 і так само з іншим портом. Даний спосіб краще використовувати для заводських пристроїв, таких як координатор, так як вони йдуть одразу з USB виходом та мітять початковий варіант прошивки, в який записано влаштований бутлоадер, який можна використовувати для заливання оновленої прошивки в стик cc2531 або просто чіп cc2530 без використання CCDebugger або Arduino. Цей спосіб було обрано для прошивки координатор (процес прошивки буде описано далі).

2.4.1 Створення програматора для чіпів на базі esp8266

Для прошивки кінцевих пристроїв було взято контролер esp8266, який було перероблено і переналаштовано для потрібних задач.

Спочатку береться сам контролер та чіп, який буде прошито, та проводиться узгодження між їх портами. Так для підключення пристроїів було створено власний унікальний для даної системи роз'єм, який буде використано для живлення та завантаження інформації в регістри пам'яті чіпу CC2530.

Так контаки пристроїв підключаються як наведено в таблиці нижче [таблиця 2.1]:

Таблиця 2.1 – порти підключення для прошивки пристрою

Призначення	**ESP8266**	**CC2530**
1	2	3
Живлення	3,3	Vin
Земля	GND	GND
Сброс контролера перед прошивкою	D1 (GPIO 5)	Reset
Передача інормації	D2 (GPIO 4)	P21 (дані)
Контроль завантаження	D5 (GPIO 14)	P22 (годинник)

Далі буде наведено приклад виконання підготовки контролеру ESP8266 [рис 2.11], де буде підключено створений роз'єм. Дане підключення далі буде використано не лише для прошивки, а й для роботи пристрою від блоку живлення та для додаткового пристро, але про це трішки далі буде розказано детальніше.

Рисунок 2.11 – Роз’єм для прошивки пристрою, припаяний до контактів контролеру ESP8266

Роз'єм включає усі перераховані раніше порти для підключення, які було зібрано в джемпер з 4 контактів типу «мама» і додатково виведено 1 контакт для порту Reset, який обов’язково потрібен щоб скидати пристрій під час процесу прошивки та отримувати зворотню інормацію про готовність чіпу до прошивки.

Наступним кроком буде підключення пристроїв між собою та завантаження всього необхідного програмного забезпечення на пристрої.

2.4.2 Підключення пристрою та його прошивка

Щоб поєднати 2 пристрої, треба монтувати інший роз’єм на стороні пристрою, який буде піддаватися перепрошивці і прозводити підключення до ESP8266, як показано на фото нижче (рис 2.12).

Рисунок 2.12 – підключення готового пристрою для його подальшої прошивки

На фото (рис 2.12) можна побачити сам пристрій, який з’єднано зі створеним дебагером. Дане з’єднання буде пропрієтарним для даної системи і значно прискорить процес створення кастомного девайсу на чіпах ZigBee.

Після вдалого підключення пристроїв один до одного відбувається перевірка за допомогою мультиметра шляхом включення режиму продзвінки і доторкання щупів до сусідніх контактів. Якщо мультиметр буде вказувати на табло значення відмінні від 0, то буде наявне правильне підключення через додатковий опір чи контакту взагалі немає (контакту немає якщо мультиметр вказує на панелі значення рівне «1»). Після закінчення всіх перевірок можна переходити до наступного кроку, а саме завантаження скетчу на контролер ESP8266 та початку процесу перепрошивки.

Скетч, який буде завантажено на контролер ESP, буде містити алгоритм перезапису даних в пам’ять за адресою кожного з регістрів пам’яті чіпу cc2530.

Всього даний чіп включає 512 регистрів пам’яті, які треба послідовно перезаписати. Для цього було прописано сам алгоритм в файлі скетчу та порти, до відбувалося підключення.

Приклад коду скетчу для контролера ESP8266 буде наведно нижче (рис 2.13).

```
#define SBEGIN                0x01
#define SDATA                 0x02
#define SRSP                  0x03
#define SEND                  0x04
#define ERRO                 0x05
#define WAITING               0x00
#define RECEIVING             0x01

int DD = 14;
int DC = 4;
int RESET = 5;
int LED = 2;
```

Рисунок 2.13 – налаштування портів на контролері ESP8266

На даному зображенні (рис 2.13) проілюстровано вказання портів для фізичного з’єднання, а також список команд, які використовуються передачі по послідовному порту. Порти вказані згідно з раніше зазначеної документації по підключенню пристроїв.

```
}
write_flash_memory_block(rxBuf, addr, 512); // src, адреса та лічильник
if(Verify)
{
  unsigned char bank = addr / (512 * 16);
  unsigned int  offset = (addr % (512 * 16)) * 4;
  unsigned char read_data[512];
  read_flash_memory_block(bank, offset, 512, read_data); // Банк, Адреса, лічильник, пункт призначення
  for(unsigned int i = 0; i < 512; i++)
  {
    if(read_data[i] != rxBuf[i])
    {
      // Fail
      State = WAITING;
      Serial.write(ERRO);
      chip_erase();
      return;
    }
  }
}
```

Рисунок 2.14 – код для послідовного завантаження прошивки у контролер

На даному рисунку (рис 2.14) наведено фрагмент коду, який буде міститися в скетчі. Даний кусок коду буде послідовно перебирати дані з файлу прошивки і записувати в відповідний до ного регістр пам’яті. Всього для запису було побудовано цикл з 512 кроками виконання. Запис у пам’ять контролера буде прозводитись за допомогою внутрішньої функції Arduino IDE – Serial.write(ERRO);

Після завантаження скетчу до контролеру ESP8266 через послідовний порт та додаток Arduino IDE можна перехродити до процесу прошивки створеного раніше пристрою.

Для цього буде використано програмне забезпечення CCLoader, яке являє собою універсальний завантажувач даних. Його буде використано для передачі інформації з комп'ютера до контролера ESP8266, звідки код буде записано до регістрів пам'яті чіпу CC2530.

Перед початком процесу перепрошивки треба файл, отриманий з конфігуратора прошивок PTVO, конвертувати з шістнадцяткового формату у двійковий, бо прошивка зберігається в регістрах пам’яті саме в двійковому формати. Тому використовуємо утиліту «objcopy.exe» для конвертування файлів. Приклад команди для конвертування файлів буде зазначено на рисунку нижче [рис 2.15].

```
C:\Users\Господин\Desktop\Диплом>C:\MinGW\bin\objcopy.exe --gap-fill 0xFF --pad-to 0x040000 -I ihex прошивка_датчик_V4.3
.hex -O binary прошивка_датчик_V4.3.bin
```

Рисунок 2.15 – команда для конвертування шістнадцяткового файлу в двійковий

Дана команда виконана в командній строці Windows. Команда містить шлях до конвертору, який буде завантажено для виконання даної задачі, режим конвертування файлу (згідно документації), шлях до початкового фалу і шлях, де кінцевий файл буде створено.

Після конвертування фалу відбувається сам процес прошивки. Для цього треба перейти в диспетчер пристроїв на комп’ютері, перевірити чи є з’єднання з контролером, чи правильна швидкість виставлена (стандартна швидкість для таких з’єднань складає 115200 бод/с) і чи встановлено драйвер для даного контролеру. Після перевірки вводимо команду, яку буде приведено нижче (рис 2.16), в консоль і спостерігаємо за процесом прошивки.

```
C:\Users\Господин\Desktop\Диплом>C:\Users\Господин\Desktop\CCLoader_x86_64.exe 3 прошивка_датчик_V4.1.bin 0
```

Рисунок 2.16 – команда для завантаження прошивки через контролер ESP8266

Дана команда містить шлях до ПЗ CCLoader, який буде запущено для виконання даної задачі, номер послідовного порту, через який відбувається передача даних та шлях до файлу прошивки у двійковому форматі.

Після правильно введеної команди почнеться процес перевірки контролерів на доступність (спочатку перевіряється контролер ESP8266 на можливі наявні проблеми у підключенні, потім перевіряєтсья сам чіп CC2530 за тим самим алгоритмом), потім запуститься ПЗ для передачі інформації на контролер ESP8266, звідки дані будуть записані на чіп CC2530, а звітність про запис кожного з регістрів буде виведено у командну строку. Процес запису даних в 512 регістрів пам'яті та процес перевірки ESP8266 та CC2530 на доступність буде наведено на рисунку нижче (рис 2.17).

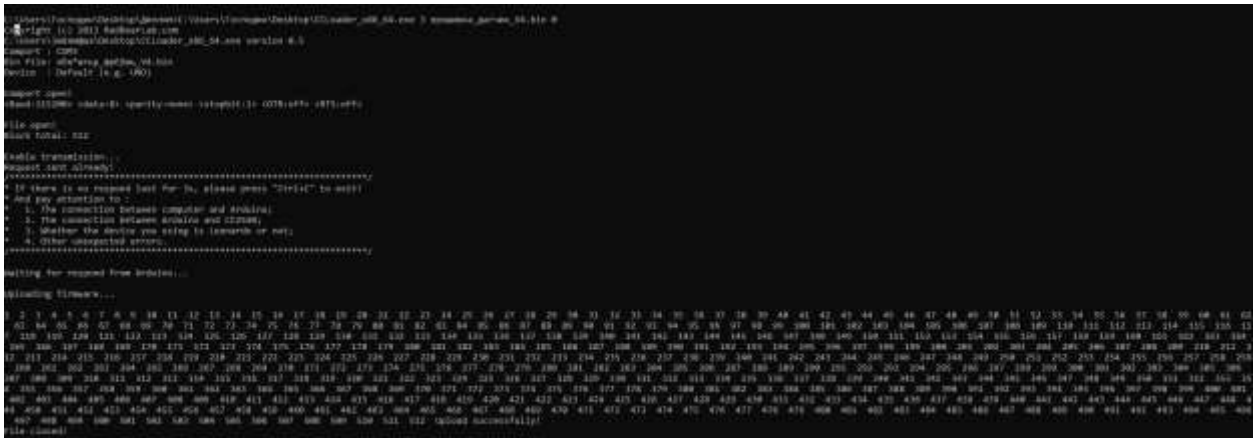


Рисунок 2.17 – приклад завантаженої прошивки і успішного звіту про це

Після успішньої перепрошивки повідомлення про це буде знаходитись в кінці звіту про завантаження. Якщо виникнуть проблеми під час запису або буде порушене з'єднання між пристроями, то процес прошивки зупиниться і буде виведено відповідне повідомлення. Як можна побачити на рисунку, проблем під час вконання не було і пристрій прошито успішно.

Після повного створення пристрою треба схематично зобразити його положення всередині мереж, щоб розуміти який вигляд буде мати кінцева мережа ZigBee [рис 2.18].

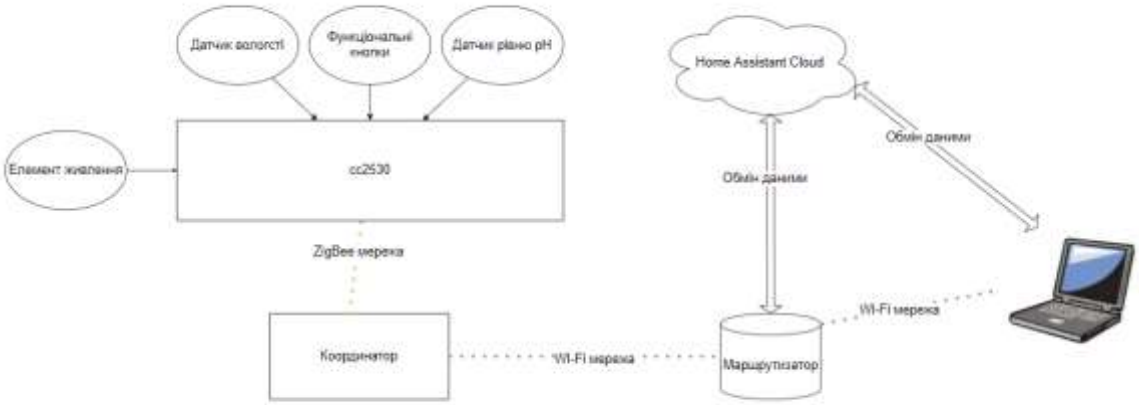


Рисунок 2.18 – схематичне зображення мережі передачі даних після інтеграції до неї ZigBee

Висновок до розділу 2

В даному розділі було досліджено шляхи створення пристроїв на базі протоколу передачі ZigBee. Так, було вигадано та спроектовано датчик, який буде за допомогою 2 сенсорів постійно аналізувати стан грунту та передавати дані про це на головний пристрій. Було розглянуто всі особливості побудови таких нових пристроїв і засвоїно на практиці все озвучене у даному розділі. Після проектування було створено прототип, який далі буде використано для інтеграції в сервіси автоматизації і для тестів.

Щоб пристрій працював коректно, його треба правильно прошити. В даному розділі розглядалися варіанти прошивки/перепрошивки пристрою а також методи створення нових кастомних прошивок.

Для завантаження прошивки було реалізовано пропрієтарне з’єднання контролерів між собою, щоб прискорити процес виготовлення і підготовки нових пристроїв.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ МЕРЕЖІ ZIGBEE ТА ЇЇ ІНТЕГРАЦІЯ В СЕРВІСИ ДОМАШНЬОЇ АВТОМАТИЗАЦІЇ

3.1 Огляд мережі передачі даних

ZigBee – це mesh-мережа, в якій може бути присутня велика кількість вузлів, кожен вузол у ній виступає проміжним маршрутизатором, отже може передавати дані до головного пристрою наскрізною пересилкою. Головним пристроєм в мережі ZigBee виступає координатор. Головна перевага mesh-мережі, яка буде розгортатися, що вона може самостійно організовуватися після відключення будь-якого пристрою. Тому якщо в мережі пропадає один вузол, інші пристрою одразу шукають інші шляхи для доставки повідомлень до координатора.

Сам координатор встановлюється в комп’ютер. Йому призначається номер порту та параметри підключення, такі як: швидкість (115200 бод/с), біти даних, стоп біти та керування потоком. Далі координатор інтегрується через MQTT брокер в Home Assistant і відбувається налаштування мережі.

3.1.1. Підбір координатора для роботи в мережі

Для роботи в мережі обов’язково потрібен координатор, який буде направляти потік даних в мережі і передавати дані від пристроїв до MQTT брокера. В якості координатора можна обрати той самий чіп, який було використано для створення спареного датчику вологості та рівню pH, а саме чіп cc2530, підключивши його порти Rx та Tx, а також живлення і землю до конвертора UART в USB - CH340G, який здатний конвертувати дані в потрібний для комп'ютера формат. Або можна обрати готовий пристрій на базі чіпу cc2531 з вбудованим USB виходом. Обидва варіанти однакові для мережі і ніяк не відразняються, але готовий варіант стіку вже скомпоновано в маленький корпус у вигляді шлешки і вже містить головне для перепрошивки через послідовний порт – бутлодер.

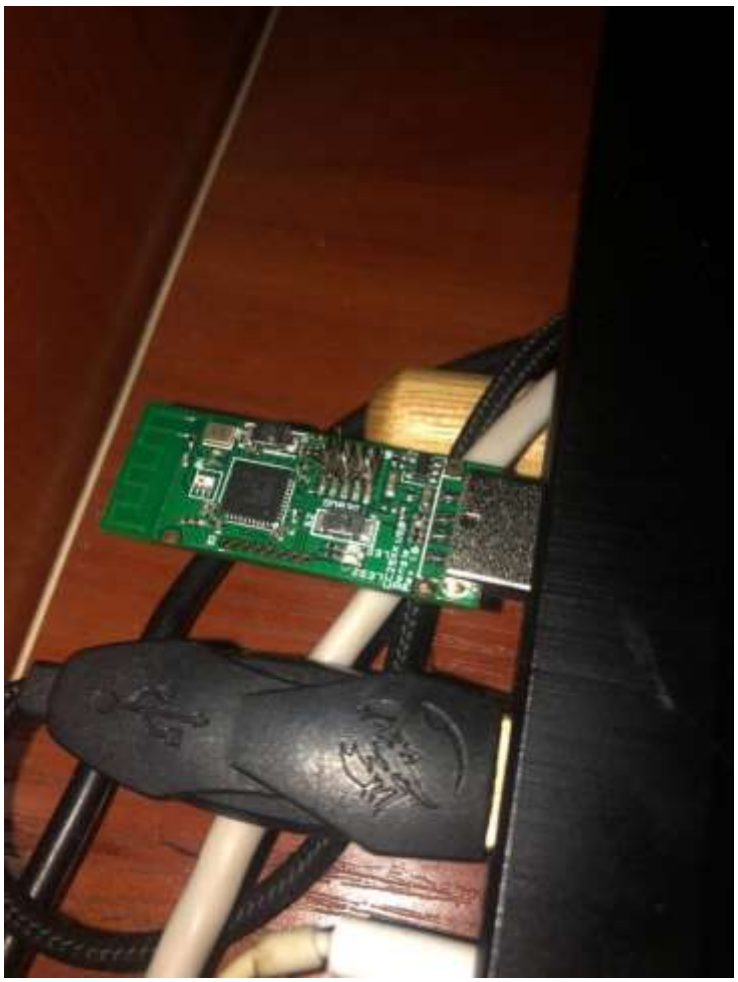


Рисунок 3.1 – USB ZigBee стік для старту роботи мережі

На фото (рис 3.1) зазначено копію оригінального ZigBee координатора від Texas Instruments, який містить піни для підключення офіційного CC Debugger, 2 функціональні кнопки, світлодіодну індикації підключення, сам чіп і вбудовану антену 2,4 Ггц.

Даний стік одразу після підключення ініціалізується в операційній системі Windows 10 і завантажує необхідне програмне забезпечення для роботи з послідовним портом, але не ідентифікується в системі як ZigBee координатор, тому що не має потрібної останньої версії програмного забезпечення.

3.1.2. Прошивка координатора

Для прошивки координатора використано вихідний код з репозиторію GitHub[15], який було підібрано згідно потрібних вимог. На репозиторію розміщено варіанти прошивки для пристроїв, щоб вони виконували функцію координатора, або якщо потрібно розширити мережу ZigBee, то можна прошити пристрій як маршрутизатор.

Головний пристрій прошивається прошивкою «CC2531ZNP-Prod». Дана прошивка розрахована на 15 прямих підключень ZigBee пристроїв і на 40 загальних пристроїв в мережі, підконтрольній даному координатору. Далі буде приведено приклади вибору версії прошивки для кожного з чіпів (рис 3.2).

Z-Stack	Device	Zigbee	Direct children	Routes	Notes
Z-Stack_Home_1.2 (default)	CC2531	1.2 HA	20	30/0	
	CC2530, CC2530 + CC2591, CC2530 + CC2592	1.2 HA	16	30/0	
Z-Stack_Home_1.2 (source_routing)	CC2531, CC2530, CC2530 + CC2591, CC2530 + CC2592	1.2 HA	5	40/40	
Z-Stack_3.0.x	CC2531	3.0	15	40/0	- Discussion #1445 - Max 40 Zigbee 3.0 devices
	CC2530, CC2530 + CC2591, CC2530 + CC2592	3.0	10	40/0	- Discussion #1445 - Max 40 Zigbee 3.0 devices
	CC2538 + CC2592	3.0	100	200/400	- Discussion #1568 - Max 200 Zigbee 3.0 devices
Z-Stack_3.x.0	CC2652P, CC2652R, CC2652RB, CC1352P-2	3.0	50	100/200	- Discussion #1429 - Max 200 Zigbee 3.0 devices

Рисунок 3.2 – перелік програмного забезпечення для перепрошивки пристрою в маршрутизатор чи координатор

Тут слід зазначити, що присутні прошивки під чіп CC2530, який було використано в якості базового для побудови спареного датчику. Отже для кожного пристрою можна завантажити вихідне ПЗ, яке перетворить його в координатор чи маршрутизатор.

Для прошивки даного пристрою використовується програмне забезпечення від Texas Instruments – SerialBootTool. Даний додаток спрощує процес завантаження двійкового файлу прошивки у пам’ять процесора. Після завантаження та відкриття треба вказати шлях до файлу прошивки (файл обов’язково має бути конвертований в двійковий формат!), зазначити потрібний COM порт та його швидкість, все інше залишається стандартним. Далі відбувається відкриття порту, після чого він стає недоступний для системи і задіяний в програмі. Далі стік треба перезавантажити просто вийнявши та вставивши його назад. Останньою дією буде натискання клавіші «завантажити зображення» і очікування записування даних у пам’ять.

Далі буде заначено приклад завантаження прошивки через додаток SerialBootTool версії 1.3.2, в якому обрані потрібні параметри для передачі (рис 3.3).

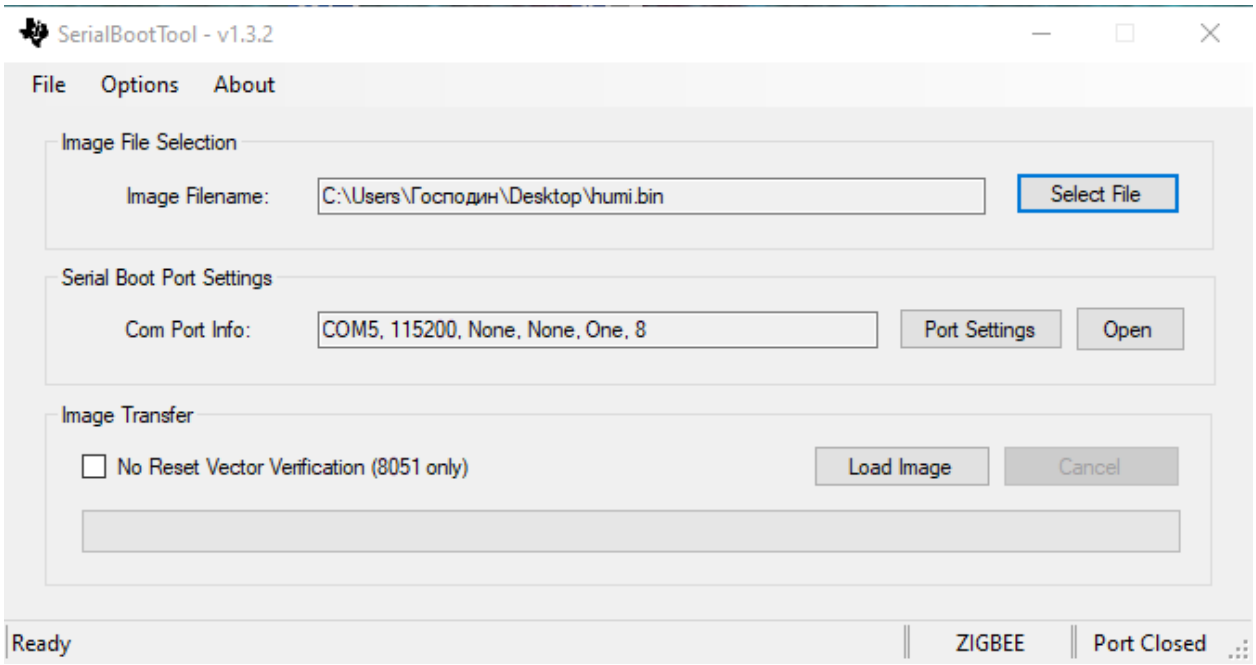


Рисунок 3.3 – Прошивка стіку через утиліту SerialBootTool

Після процесу прошивки стіку в якості координатора пристрій треба ще раз перезавантажити і він буде готовий до інтеграції в сервіси домашньої автоматизації.

3.1.3. Встановлення Home Assistant, MQTT брокера та інтеграція координатора

Варіантів встановлення Home Assistant існує 4:

- завантаження Lite версії, яка при відкритті завантажує сервіс автоматизації і надає обмежений функціонал
- встановлення Home Assistant допомогою Python 3.0 і розгортання локального серверу буде здійснено за допомогою його можливостей
- за допомогою Docker контейнерів[]
- встановлення та розгортання віртуальної машини VirtualBox[]

Для виконання дипломної роботи було обрано варіант з встановлення віртуальної машини VirtualBox, бо вона дозволяє створювати окрему віртуальну машину всередині Windows, надаючи їй потрібних ресурсів і постійно слідкуючи за її станом через системні утиліти. Так, після відкриття віртуальної машини, яку раніше було використано для лабораторних робіт в університеті, можна почати створення нового середовища під потрібну операційну систему. За замовчуванням Home Assistant розгортається на ядрі Linux. Кінцева операційна система завантажується з файлу з розширенням vdi, який розпаковується на базі раніше створеної ВМ, система буде називатися Home Assistant OS. Перед завантаженням треба додати одне відео ядро до ВМ та вказати в налаштуваннях номер послідовного COM порту, який буде через мережевий міст буде поєднано з віртуальним послідовним портом на віртуальній машині (рис 3.4). Далі відбувається запуск операційної системи (рис. 3.5).

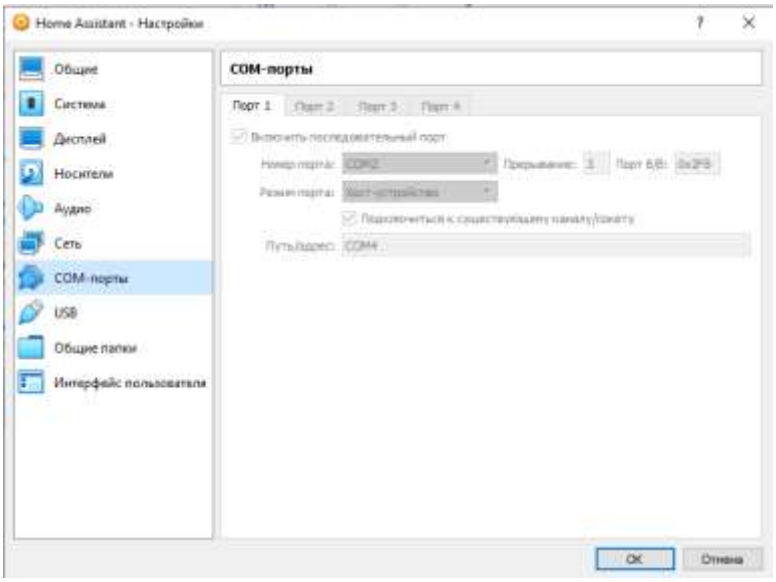


Рисунок 3.4 – Налаштування мосту для підключення стіку до віртуальної машини

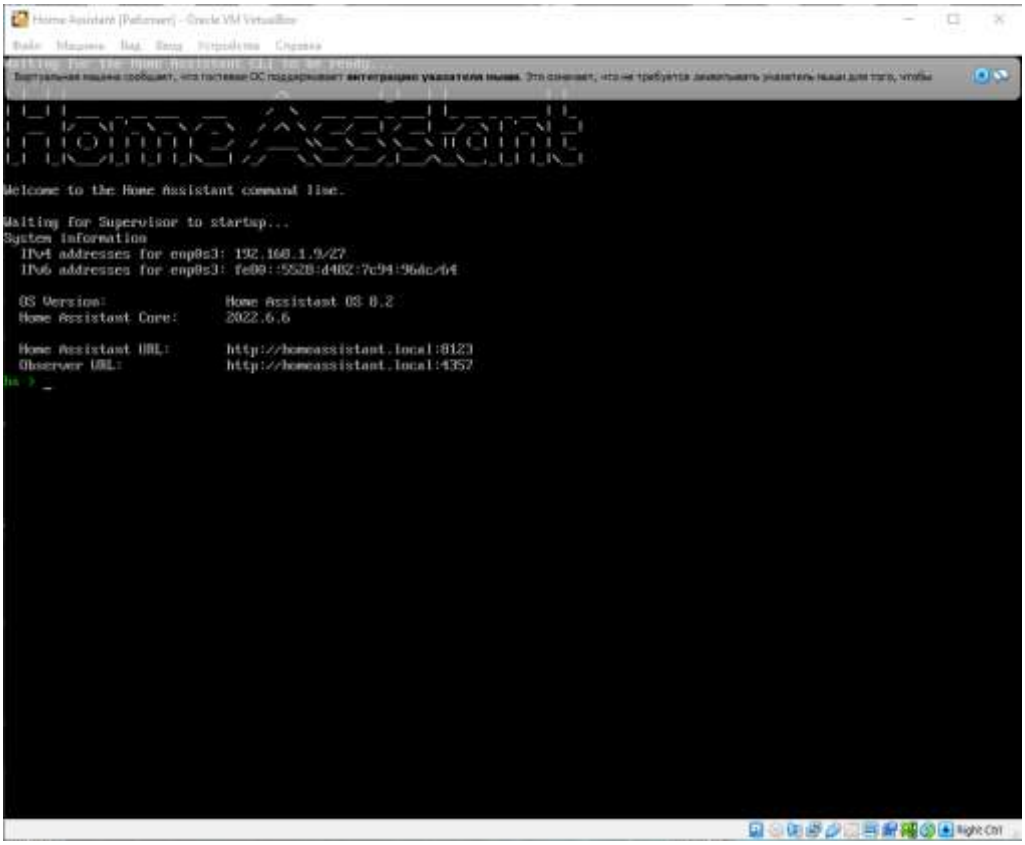


Рисунок 3.5 – Стартове вікно Home Assistant у консолі віртуальної машини

На рисунку нижче (рис 3.6) буде зображено стартова сторінка операційної системи Home Assistant всередині віртуальної машини. Для підключення до локального серверу потрібно ввести вказану у віртуальній машині адресу, яка

відправить на вікно початкового налаштування Home Assistant. Далі створюється новий профіль домашньої автоматизації, вказуються дані власника дому, домашня мережа (варто зазначити, що при першому завантаженні НА система визначає домашній маршрутизатор, його модель і дані про пропускну спроможність, а також пропонує додати пристрій до середовища домашньої автоматизації.

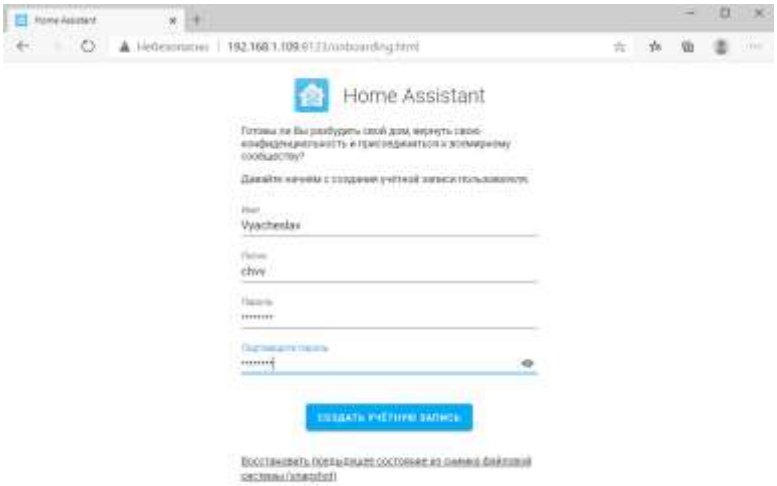


Рисунок 3.6 – Стартове вікно Home Assistant після виклику його через браузер за допомогою локальної адреси.

3.2 Створення додаткового ZigBee маршрутизатора з 2 реле на сухих контактах для комутації постійного та змінного струму, та розширення загальної мережі ZigBee

Для розширення загальної мережі і демонстрації можливостей ZigBee протоколу було вирішено створити додатковий пристрій, який буде виконуючим, дані з датчиків вологості та рівню pH будуть прямим чином впливати на роботу виконуючого пристрою. В якості виконуючого пристрою було рішено створити пристрій з бездротовим керуванням, який буде містити 2 реле на сухих контактах, антену для зв’язку, понижувач напруги, блок живлення, діод для індикації і чіп cc2530. Дані реле можуть комутувати змінну та постійну напругу. До контактів реле можна підключити світло, або клапан, або насос і тому побідне. Тобто можна підключити будь-який пристрій, який буде залежати від реле як від перемикача.

За для збільшення покриття домашньої мережі автоматизації дане реле потрібно прошити в якості маршрутизатора, так як воно може бути підключене до мережі або до акумуляторної батареї великої ємності, яка буде розташована на підлозі.

Для проектування виробу буде використано ті самі правила, що й в розділі 2.1, середовище для проектування залишилося без змін (EasyEda). Нижче буде наведено електрична схема створення маршрутизатора з 2 реле на сухих контактах (рис. 3.7).

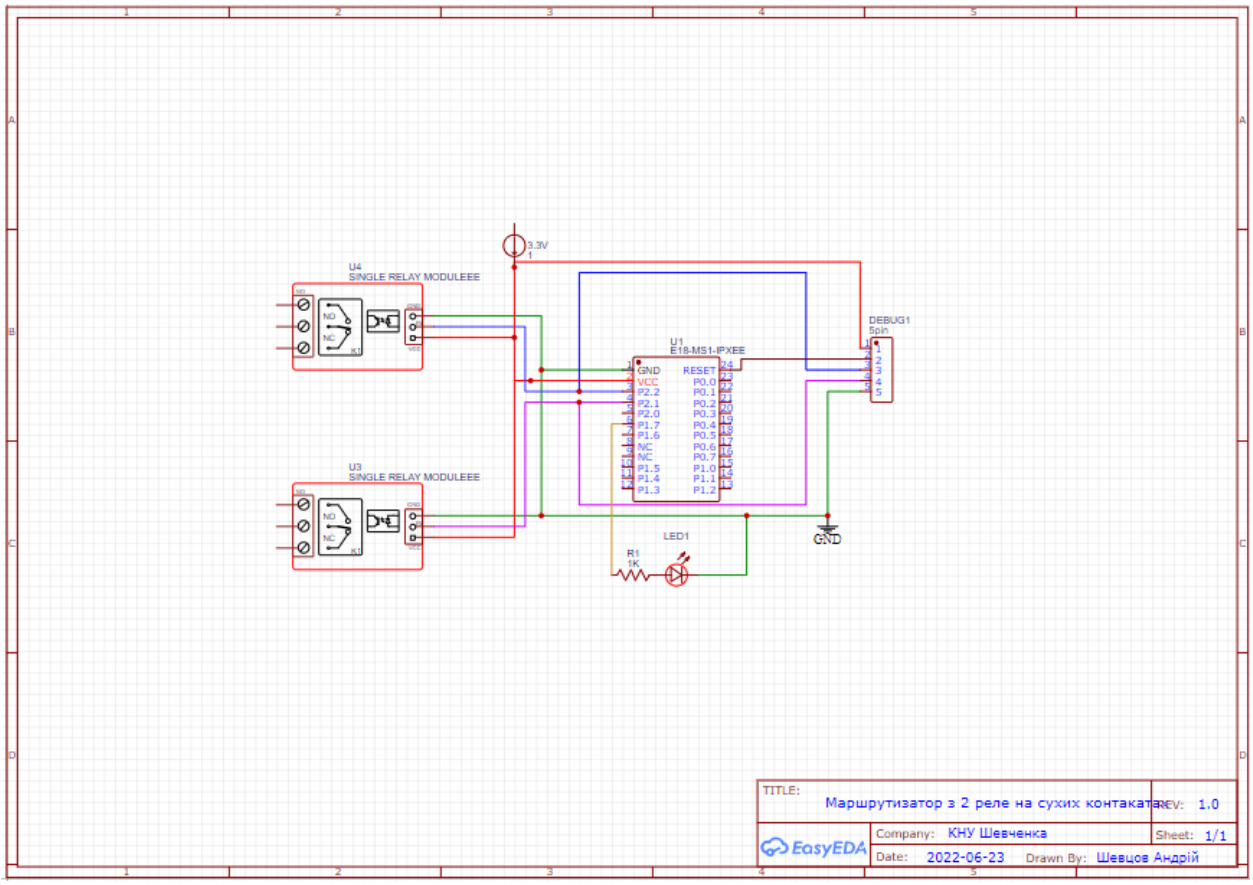


Рисунок 3.7 – Проектування додаткового маршрутизатора в середовищі EasyEda

Для прошивки використано той самий пропрієтарний роз’єм, який було використано для прошивки попереднього пристрою. Також було додано світлодіод для індикації та 2 реле на сухих контактах.

3.2.1 Фізичне підключення компонентів

Першочергово треба припаяти ніжки для подальшого монтажу елементів. Було обрано з’єднувальна пара ніжок, крок яких був у 2 рази більший ніж у пінів чіпу cc2530. Тому на один ряд пінів було припаяно 2 з’єднувальні ніжки одразу. Далі проводиться перевірка на коротке замикання та з’єднання елементів.

Рисунок підключення елементів до чіпу наведено нижче [рис. 3.8].

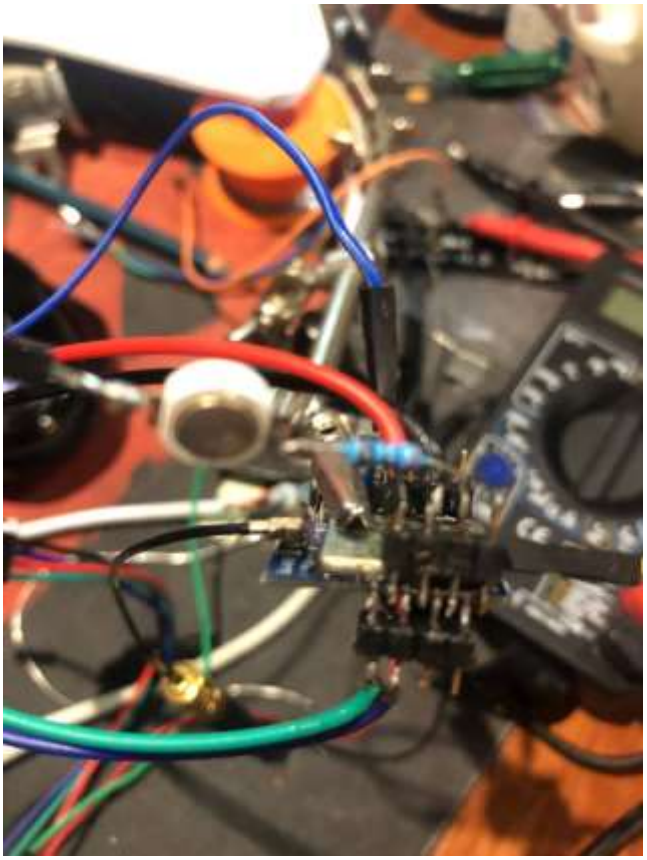


Рисунок 3.8 – Монтаж комплектуючих для нового маршрутизатора ZigBee

На даному рисунку (рис. 3.8) зображено чіп cc2530, на який вже припаяні з'єднувальні ніжки. До чіпу через дані ніжки підходить живлення 3,3 Вольта та мінус. Контакти p21 та p22 використовуються для заванатаженя прошивки та для керування станом реле. На контакті p17 виведено світлодіод для індикації підключення та стану роботи пристроя.

Фінальна частина приладу вміщається в маленьку коробку. На фото нижче (рис. 3.9) зображено фінальний вигляд пристрою у коробці з

підключеним живленням. Можна було б вибрати і менший розмір корпусу, але мене повністю задовольняє даний вибір.

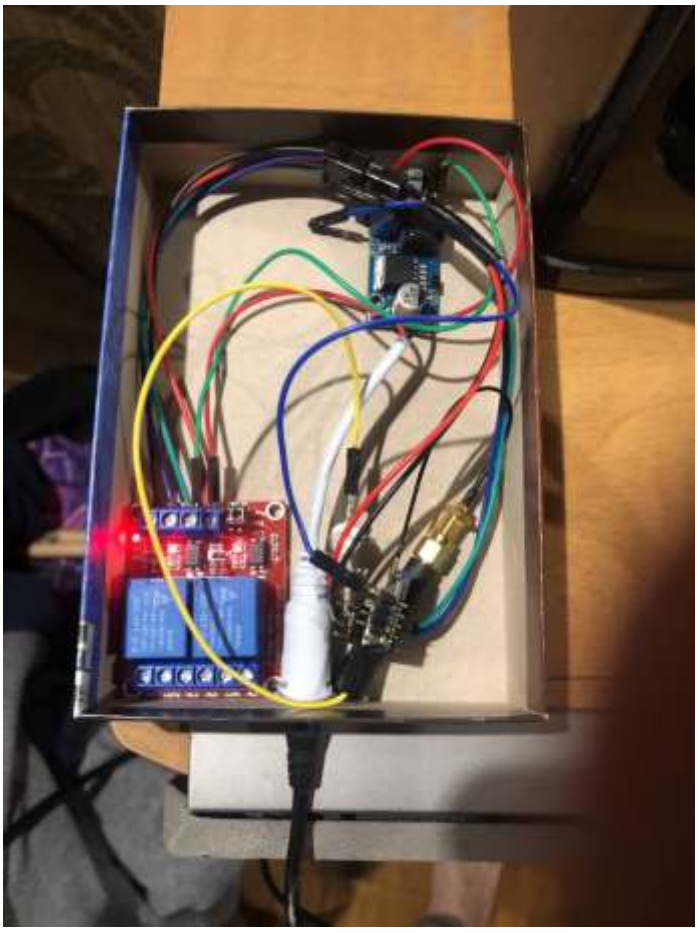


Рисунок 3.9 – Готовий маршрутизатор ZigBee

Після зборки потрібно налагодити прошивки пристрою та завантажити її.

3.2.2 Прошивка пристрою

Для прошивки обрана та сама схема, що була використана при прошивці спареного датчику вологості та рівню pH. Спочатку прошивка конфігурується в додатку PTVO. У конфігуратор було додано 2 порти GPIO для керування реле, 1 порт для світлової індикації та 1 віртуальний пін для внутрішньої температури.

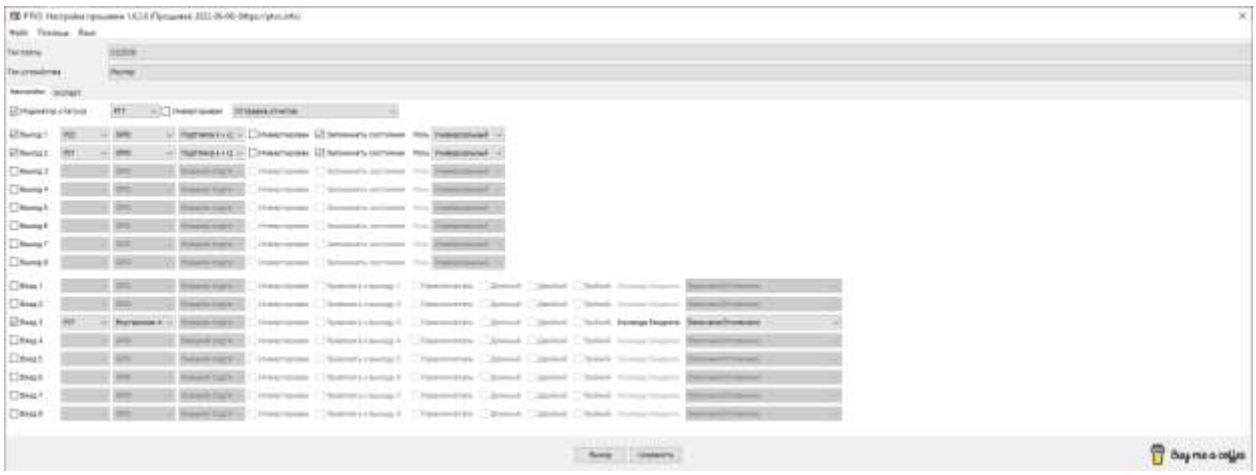


Рисунок 3.10 – Конфігурація прошивки для маршрутизатора з 2 реле на сухих контактах

У пристроя після монтажу є роз’єм для прошивки, який просто треба підключити до ESP8266 (треба не забути підключити окремий контакт для скидання пристрою). Потім за допомгою внутрішньої утиліти конвертуємо отриманий шістнадцятковий файл з конфігуратора в двійковий. Далі через програмне забезпечення CCLoader завантажується прошивка у регістри пам’яті процесору.

Нижче буде опубліковано фото успішної перепрошивки (рис. 3.11). Після закічення пристрій відключається від програматора та готове до підключення.

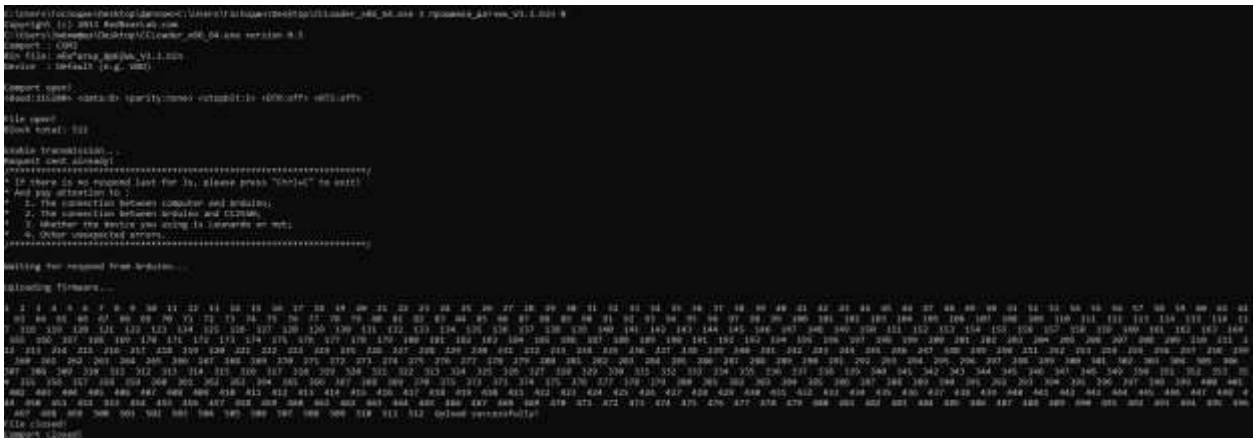


Рисунок 3.11 – Завантаження прошивки на маршрутизатор

3.3 Інтеграція в Home Assistant всіх наявних пристроїв

Для інтеграції пристроїв в середовище домашньої автоматизації Home Assistant потрібно встановити addons[] через магазин додатків HA. Даний магазин надає можливість встановлення додаткового програмного забезпечення за для збільшення можливостей домашньої автоматизації.

Так, для початку, треба встановити Mosquito MQTT брокер, який виступає програмним мостом для поєднання різних систем передачі інформації і дозволяє їм спілкуватися між собою. Так MQTT брокер виступає мостом між Home Assistant та технологією ZigBee. Для прямого зв'язку з MQTT приладів ZigBee використовують доповнення ZigBee2MQTT, який є мостом між девайсами мережі та брокером, а вже брокер зв'язує інші ZigBee прилади з середовищем Home Assistant.

Далі буде приведено приклад роботи мережі ZigBee і всіх її пристроїв із Mosquito MQTT брокером (рис. 3.12).

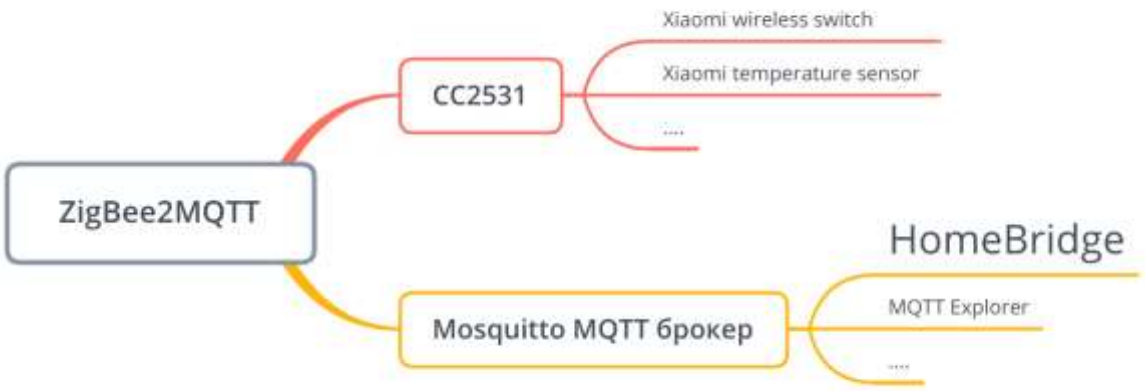


Рисунок 3.12 – Схема зв'язку MQTT брокера з пристроями ZigBee

Нижче буде зазначено всі доповнення, які були завантажені для інтеграції пристроїв та налагодження початкової автоматизації (рис. 3.13).

Рисунок 3.13 – Необхідні доповнення для створення домашньої ZigBee мережі

На даному знімку екрану зображено 5 доповнень, які знадобляться для інтеграції. Для початку за допомогою SSH терміналу ми підключаємося до Home Assistant та перевіряємо підключені потри за допомогою команди - “ls -l /dev/tty”, потім натискаємо клавішу Tab та бачимо список всіх зайнятих портів.

На рисунку нижче зображено список всіх зайнятих портів в віртуальній машині (рис. 3.14).

Рисунок 3.14 – Зайняті порти у Home Assistant

З цього списку ми бачимо один порт, який відрізняється від інших. Якщо в віртуальну машину не було нічого підключено окрім ZigBee координатора, то це буде його порт всередині образу віртуальної машини. Далі переходимо в конфігурацію ZigBee2MQTT та вказуємо в налаштуваннях даний порт, щоб додаток міг спілкуватися з координатором за обраним шляхом. Після цього запускаємо ZigBee2MQTT та переходимо у веб-інтерфейс. Тут ми бачимо місце в колонках під пристрої ZigBee. У правому верхньому куті натискаємо клавішу дозволу додавання нових пристроїв.

Далі треба взяти створені пристрої і перезавантажити їх, щоб при повторному запуску відбувся процес додавання нових зв’язків. Після декількох секунд ми бачимо як створене раніше двох-канальне реле та спарений датчик вологості та рівню pH підключилися до мережі. На рисунку

нижче буде продемонстровано список пристроїв, які було додано до Home Assistant.

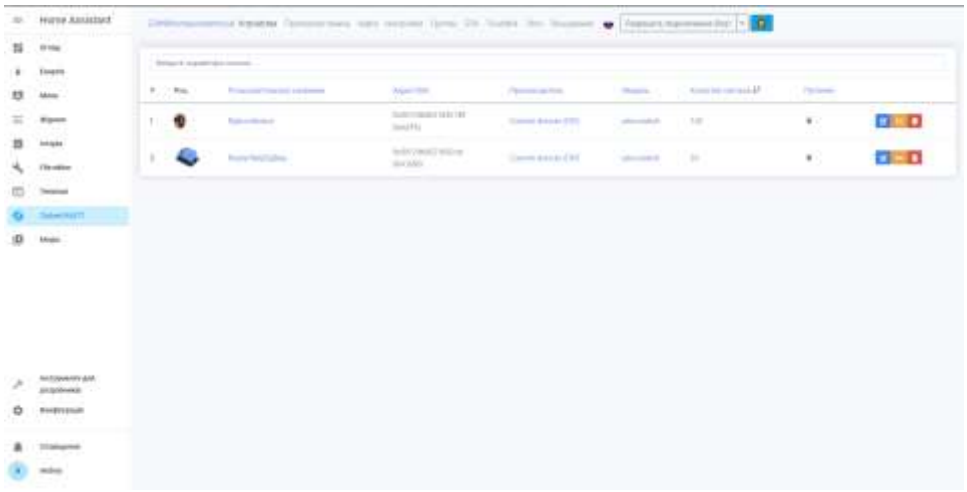


Рисунок 3.15 – Вікно ZigBee2MQTT, де вже додано 2 пристрої, створені раніше

Тут (рис. 3.15) ми бачимо 2 пристрої, які набули ознак, які їм було надано при процесі прошивки. Тут одразу можна перейти у вікно перегляду для кожного окремого пристрою.

На наступному рисунку (рис. 3.16) буде приведено приклад пристрою, який було додано до середовища автоматизації.

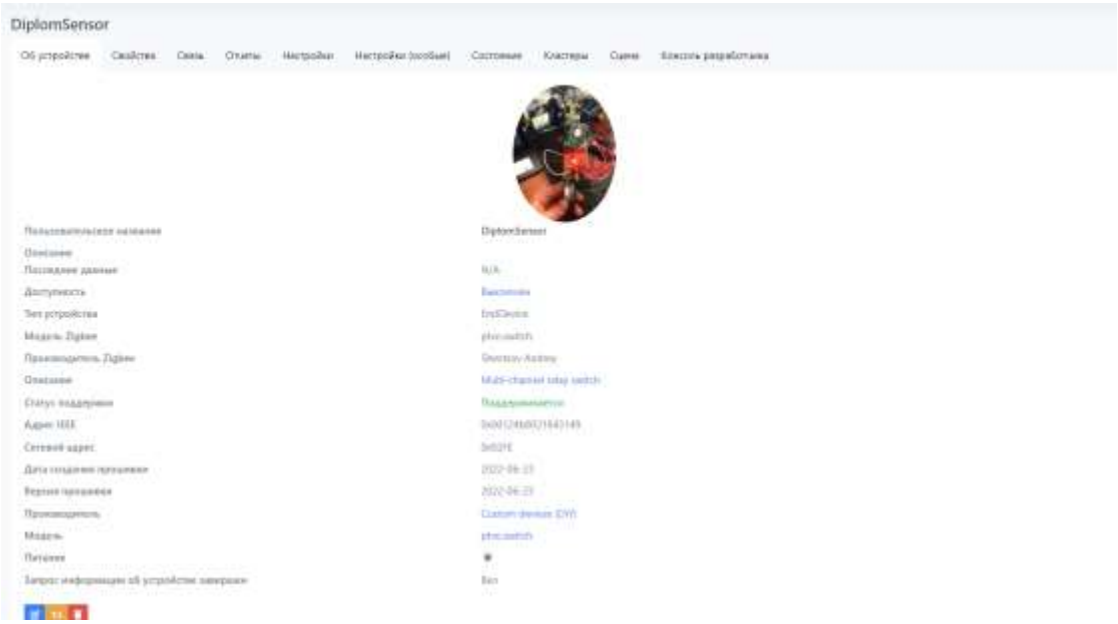


Рисунок 3.16 – Паспортні дані пристрою

На даному рисунку зображено початковий паспорт приладу, в якому вказана фотографія, модель, виробник, дата останньої прошивки та статус. Щоб переглянути значення можна перейти до панелі керування, чи обравши один з пристроїв, продивлятись стан датчиків та стан включення елементів.

На рисунку нижче (рис. 3.17) зображено панель керування окремим пристроєм, де вказуються показники з датчиків та можна змінювати стан обладнання: переключати реле, змінювати стан світлодіоду та продивлятися показники з датчиків.

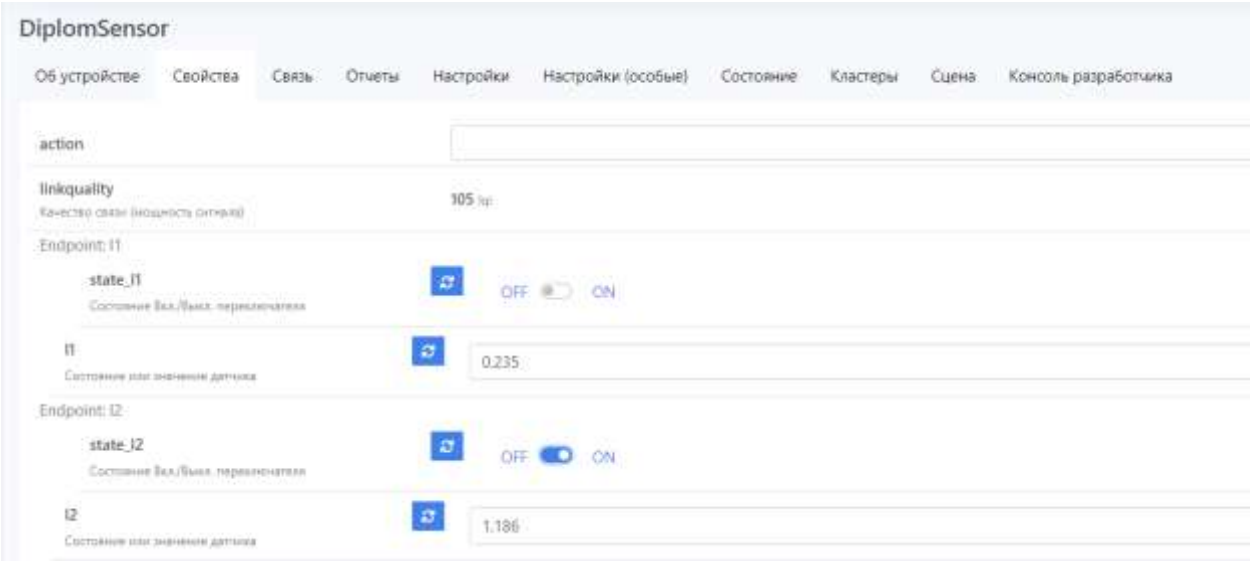


Рисунок 3.17 – Вікно перегляду показників з портів

Тут можна побачити, що світлодіод, який прив’язаний до входу l2, зараз знаходиться в активному стані. Показники сенсорів l1 = 0.235, l2 = 1.186.

Показники, які ми отримали від датчиків є вхідною напругою, який генерує датчик після перетворення струму всередині нього під дією зовнішніх чинників. Це ще не повні показники датчиків, бо їх значення спочатку треба калібрувати за допомогою різник тестів. Про калібрування і подальше налаштування датчику буде описано далі.

3.4. Тестування мережі ZigBee та приладів всередині неї

Для тестів із ZigBee мережею було взято 2 створені раніше пристрої і проведення з ними різних маніпуляцій. Так як обидва пристрої можна живити від двох звичайних батарейок, то тоді їх можна переміщувати в межах мережі як завгодно. Тому для розв'язання даної задачі було створено тест план, який допоможе краще орієнтуватися в можливостях протоколу ZigBee.

План буде включати 3 етапи:

1) Оцінка якості зв'язку у різних частинах квартири
2) Оцінка переданих показників
3) Тестування сенсорної мережі (коли при відключення головного пристрою чи іншого маршрутизатора кінцевий пристрій автоматично знаходить вузол, до якого зможе підключитися.

3.4.1. Тестування зв'язку

В першу чергу треба оцінити можливості використаних антен 2,4 Ггц з підсиленням 20 dBm, які було вмонтовано в датчик та маршрутизатор. Для цього беремо обидва пристрої і переміщуємо їх в зоні квартири. Для позначення місцеположення в квартирі буде продемонстровано приблизний план, який буде включати всі приміщення в квартирі а також положення кожного з пристроїв в певному випадку.

Нижче буде наведено пустий план квартири (рис. 3.18). В спальні вказано положення головного пристрою, який буде зберігати його під час всього тестування. Створення плану квартири було здійснено в веб-додатку RemPlanner, який дозволив швидко створити 2D план, на якому можна розміщувати прилади, потрібні для тестування якості зв'язку.

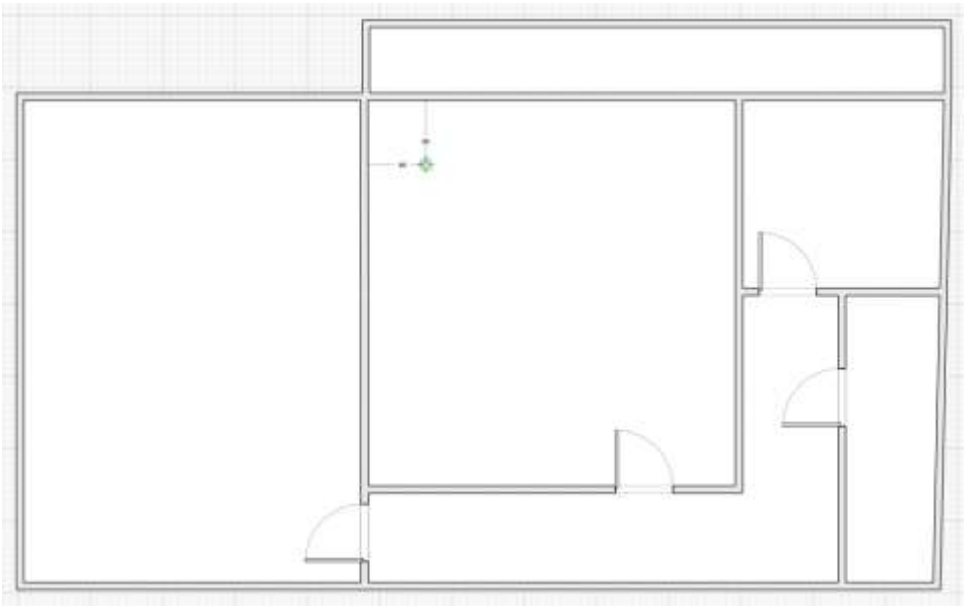


Рисунок 3.18 – План квартири

На даному зображені наведено план квартири, де зеленим кольором вказано положення координатора

Тест № 1.

Умова: реле на 2 канали та Датчик вологості та рівню pH знаходиться в кімніті приблизно поряд один з одним.

Далі буде зображено їх положення (рис. 3.19), виміряно приблизну відстань та записано стан сигналу на обох пристроях.

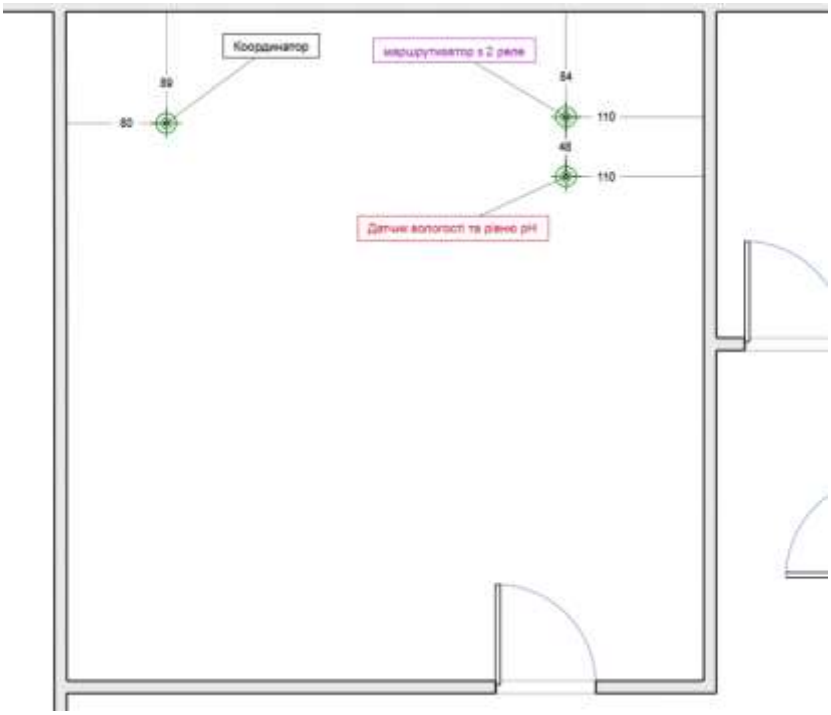


Рисунок 3.19 – Тестування зв’язку у приміщені, тест №1

Відстань складає приблизно 2,5-3 метри без перешкод.

Далі відкриваємо карту пристроїв у ZigBee2MQTT і бачимо, яка якість сигналу в обох пристроїв.

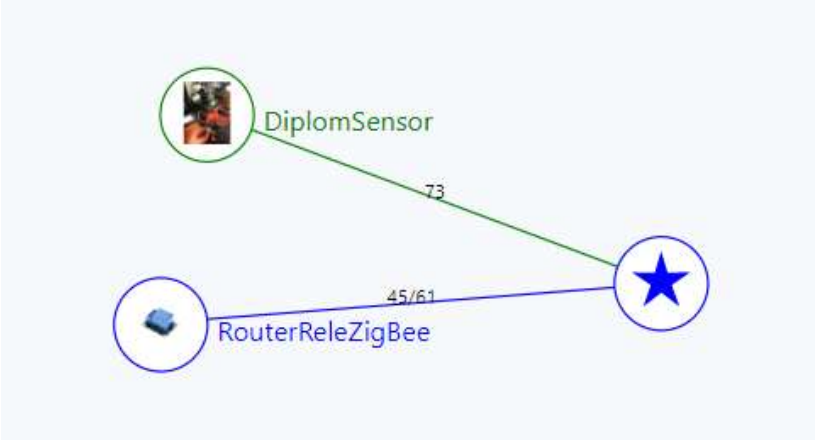


Рисунок 3.20 – Карта мережі ZigBee після змін

В датчику вологості та рівню pH грунту сигнал кращий, ніж у маршрутизатора з реле. Це може пояснюватись тим, що маршрутизатор з антено. Сховано в коробку, щоб він мав вигляд пристрою, тому коробка може створювати незначні похибки в зв’язку.

Для порівняння якості зв’язку буде обрано апаратний метод оцінки якості радіоканалу, якого відносять показник LQI [], який і вказується поряд з маршрутом на карті пристроїв ZigBee. Якість зв’язку датчику складає 73 lqi, тоді як у маршрутизатора цей показник складає 61 lqi.

Тест №2.

Умова: реле знаходиться в вітальні а датчик розташовано посеред ванної кімнати.

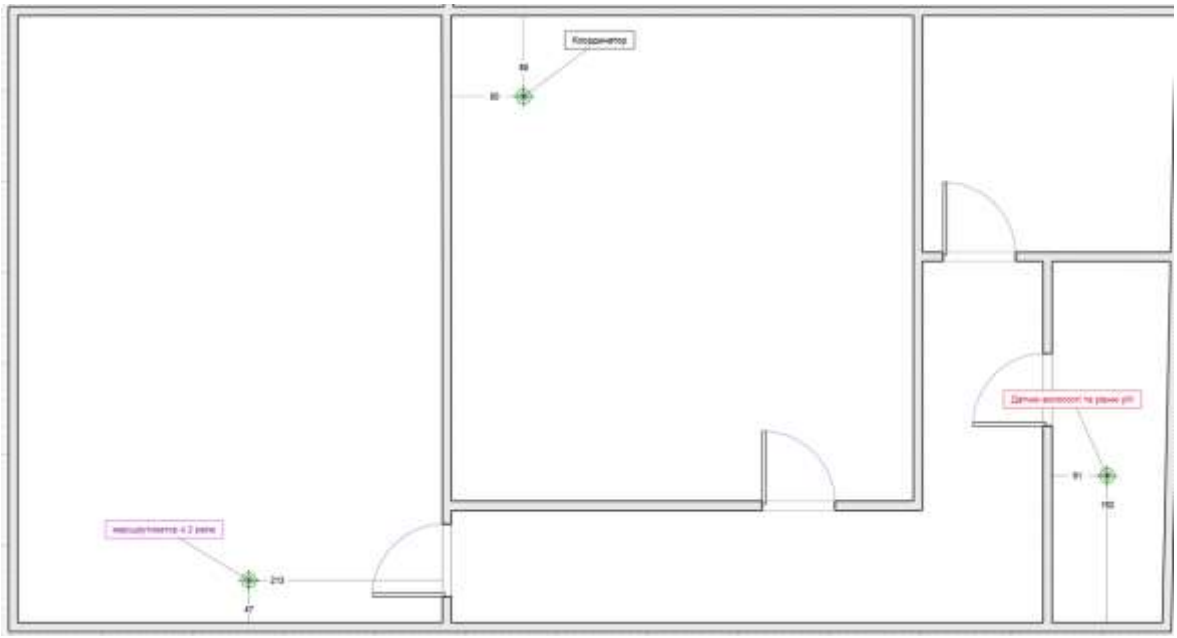


Рисунок 3.21 – Тестування зв’язку у приміщені, тест №2

Реальна відстань до датчику вологості – 5,5- метрів через 2 бетонні стіни.

Відстань до маршрутизатора складає 4,5 – 5 метрів через 1 бетонну стіну.

З даного тесту можна зробити висновок, що мережа ZigBee значно залежить від перешкод, які лежать від двома пристроями що спілкуються. При схожих відстанях обох пристроїв різниця в сигналі колосальна, це все через додаткову стіну.

Якість зв’язку датчику складає 34 lqi, тоді як у маршрутизатора цей показник складає 63 lqi.

Тест №3

Даний варіант тестування буде направлений на виявлення максимально доступної відстані з перешкодами для зв’язку. Коли зв’язок буде втрачено – це буде границя мережі ZigBee з одним координатором в якості приймача.

Маршрутизатор з 2 реле буде знаходитись на кухні, тоді як датчик буде знаходитись в крайній точці на балконі.

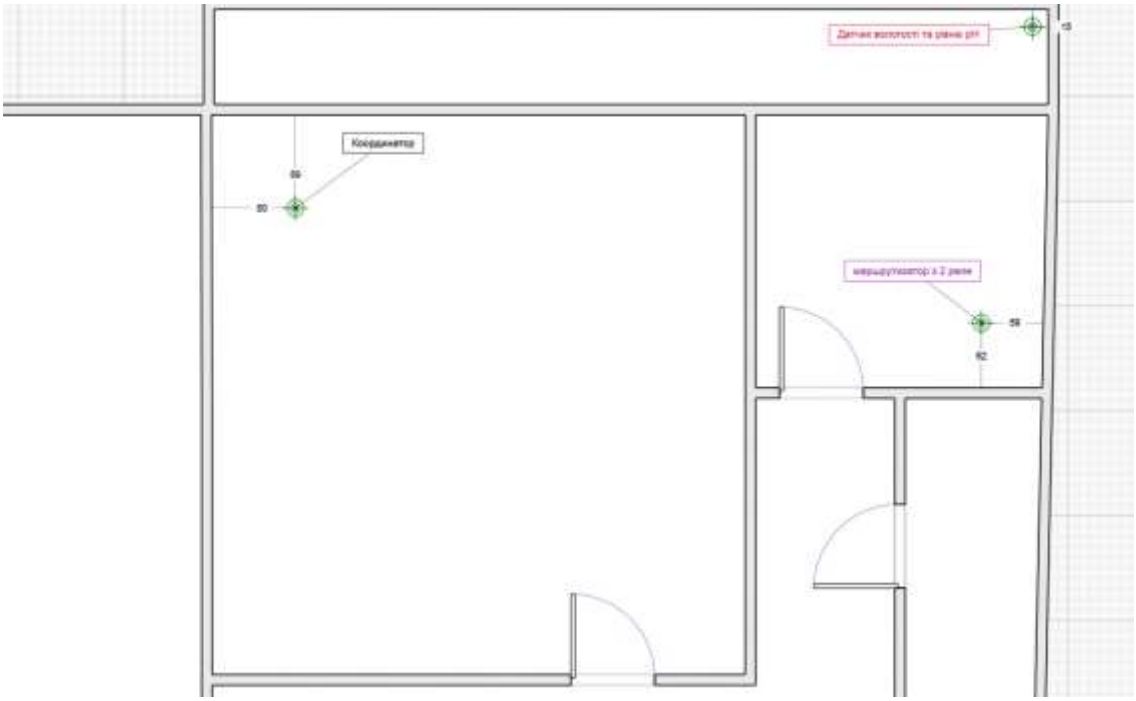


Рисунок 3.22 – Тестування зв’язку у приміщені, тест

Реальна відстань до датчику складає 6-6,5 метрів через 1 бетонну стіну та скло, а відстань до реле 5-5,5 метрів. При цьому якість сигналу датчику майже нульова, бо він вже знаходиться на краю мережі. В такому випадку повідомлення не відразу доходять і створюється затримка.

З цього можна зробити висновок, що дальність покриття мережі ZigBee повністю покриває потреби, коли розмір кварти 2-3 кімнати і пристрій знаходить у відносному центрі квартири. Якщо ж приміщення більше, то треба підключати проміжні маршрутизатори, бо можна поплатитися втратою даних з датчиків.

Нижче буде наведено карту покриття ZigBee мережі (рис. 3.23). Зеленим кольором вказано зону, де якість сигналу відмінна, оранжевим вказано зону

середньої якості зв’язку, червоним зазначено зону, де підключення вже не гарантовано і зв’язок з пристроєм може перериватися.

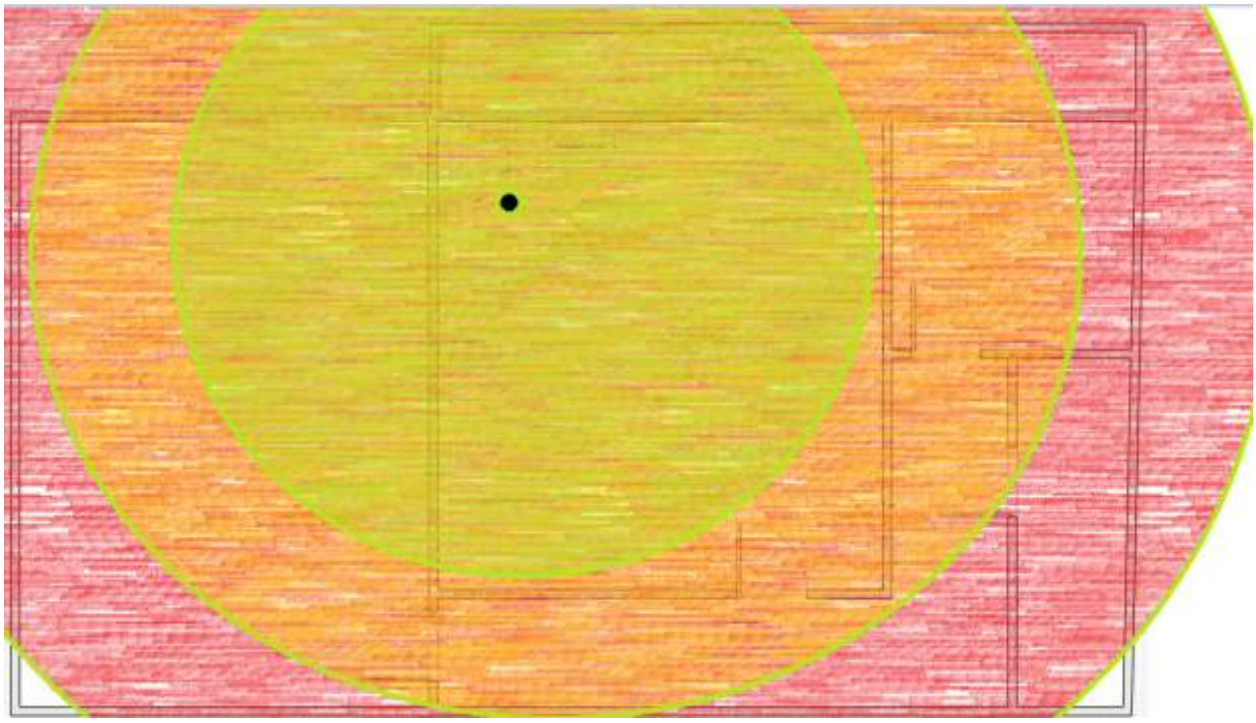


Рисунок 3.23 – Зона покриття ZigBee в квартирі. Зона розходиться навколо координатора у центрі

3.4.2 Тестування підключення через проміжний маршрутизатор ZigBee

Після тестування якості підключення треба перевірити одну з ключових особливостей mesh-мережі. Для цього треба створити умову, в якій маршрутизатор буде доступний для зв’язку з координатором, а датчик буде в зоні дії маршрутизатора, але до координатора зв’язок доставати не буде.

В квартирі, де проводяться досліди, ще з попереднього тесту було встановлено, що гранична точка початкової мережі лежить на краю балкона.

Тому для перевірки автоматичної конфігурації мережі було обрано розмістити маршрутизатор близько до виходу. Датчик було віднесено до під’їзду на приблизно 5-6 метрів від вхожу. З телефону було перевірено що одразу після виходу з квартири датчик зник з мапи і залишився лише маршрутизатор, підключений до координатора.

Після 5-10 секунд оновлень карта змінилась. Зміну карти наведено на рисунку нижче (рис. 3.24).

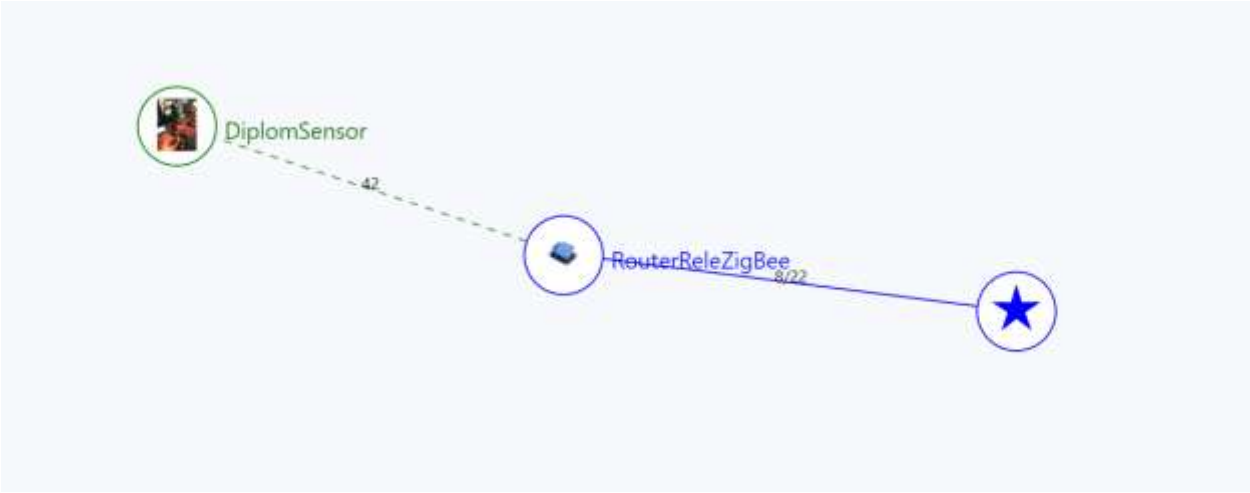


Рисунок 3.24 – З’єднання через проміжний пристрій (маршрутизатор)

На даній ілюстрації (рис. 3.24) можна побачити, що кінцевий пристрій самостійно приєднався до маршрутизатора в мережі, використовуючи його як міст між ним самим та координатором. Так якість зв’язку між датчиком та маршрутизатором складає 42 lqc, що є гарним показником на такій відстані. Тепер з двох проведених тестів можна судити про можливості мережі ZigBee і можна створювати промислову ZigBee мережу автоматизації, знаючи граничні значення мережі. В закритому середовищі дальність сигналу складає близько 6-7 метрів, в залежності від кількості перешкод, і на відтритому повітрі десь 10-15 метрів.

3.4.3. Тестування показників

Для тестування показників буде взято дані, які передає пристрій до координатора мережі, та порівняно з даними, які можна отримати вручну при вимірюванні напруги на виході датчиків.

Для калібрування датчиків їх треба занурювати в різні середовища, та робити висновки про пряму залежність чи залежність відносно деяких показників. Так датчик вологості тестується і калібрується за допомогою послідовного занурення датчику в сухе та вологе середовище і вимірювання

показників, які отримуються в результаті зняття показників. Датчик рівню pH калібрується за допомогою декількох середовищ (кисле, нейтральне та перенасичене). Так у нейтральному середовищі датчик має вказувати рівень pH = 7.0, в кислому середовищі близько 2,4 pH, а в перенасиченому рівень pH буде складати близько 14 pH.

Для початку треба занурити датчик вологості повністю у воду. Далі запишемо показники, які передасть датчик на комп'ютер і проведемо вимір за допомогою мультиметра.

Показник датчику вологості у воді на комп'ютері – 1,185 V.

Показник датчику вологості у воді, виміряний за допомогою мультиметра – 1,07.

Похибка складає (1,185-1,07)/1,185*100 = 9,7 %

Тепер проведемо перевірку датчику у сухому середовищі.

Показник датчику вологості в повітрі на комп’ютері – 2,753 V.

Показник датчику вологості в повітрі на мультиметрі – 2,4 V.

Похибка складає близько 12 %. Тобто АЦП процесора вимірює показання датчику з похибкою 10-12% від реальних показників в одному напрямку збільшення показника. З цього всього можна зробити висновок, що кореляція між відношеннями показників з комп’ютера та з мультиметра значна, тобто залежність майже пряма.

Далі проведемо дослідження датчику рівню pH. Для цього він занурюється у воду і калібрується так, щоб показник pH відповідав 7.0. Наступним кроком щуп приладу занурюється в кисле середовище, наприклад лимонну кислоту. Тепер датчик має вказувати показники близькі до 2,4 pH. Якщо все збігається, то датчик рівню pH налаштовано правильно.

3.5. Створення логіки роботи програми

Перед створенням сценаріїв роботи приладів та автоматизації треба відкалібрувати показники з датчиків у файлі конфігурації. Це робиться за допомогою додатку File editor. Далі відкриваємо файл у головному каталозі Home Assistant під назвою configuration.yaml. Цей файл виконується під час запуску Home Assistant і містить основні параметри запуску додатків та інтеграцій. В даному файлі треба створити конфігурацію для кожного з сенсорів, щоб конвертувати показники датчиків у відсотки. Після цього кожну з створених сутностей конвертування можна використовувати для подальшої передачі даних, для простого запису в базу даних, для передачі у стороні сервіси чи просто для виконання умов автоматизації. Наприклад: щоб реле включило роботу клапану водопостачання треба щоб показник вологості впав нижче 60%. Чи якщо вологість вище 80% буде включено світло та підогрів. Нижче буде наведено приклад коду для конвертування даних з аналогового датчику (рис. 3.25).

```
sensor:
  - platform: mqtt
    name: "Humi"
    state_topic: "zigbee2mqtt/DiplomSensor"
    value_template: '{{ (((2.8 - (value_json.voltage_12|float)) / 1.6) * 100 ) | round(0) }}'
    vailability_topic: "zigbee2mqtt/bridge/state"
    payload_available: "online"
    payload_not_available: "offline"
    json_attributes_topic: "zigbee2mqtt/DiplomSensor/attributes"
    icon: mdi:leaf
```

Рисунок 3.25 – Програмування датчиків вологості та рівню pH в середовище HA

Даний код містить платформу, в якій працює даний пристрій, шлях до пристрою всередині додатку ZigBee2MQTT. За вказаним шляхом обирається атрибут, який передається у форматі Json. Даний показник можна просто знімати, чи проводити математичні підрахунки для корекції показників датчиків. Так код для датчику може включати умови та порівняння.

Далі буде приведено датчик, який додано на головну панель поряд з виконуючими пристроями (рис. 3.26). Шкала відображення налаштована на якість показнику.

Рисунок 3.26 - Приклад відображення статистики та панелі керування у Home Assistant, де вказано показник з датчику у відсотках

Тепер дані, які отримує система з датчиків можна використовувати для побудови сценаріїв автоматизації або передавати в сторонні сервіси для аналізу стану грунту.

3.6. Створення сценарію роботи приладу в мережі ZigBee, автоматизації поливу та живлення рослин

Висновок до розділу 3

В даному розділі було розглянуто питання розширення мережі ZigBee за допомогою саморобного маршрутизатора з 2 реле. Даний пристрій зможе розширити покриття мережі, а також може виступати в якості виконуючого пристрою. Він так само додається до середовища автоматизації, як і кінцеві пристрої, але мастить додаткову функції маршрутизації, яка дозволяє підключати окремі пристрої до потрібних маршрутизаторів.

Після розширення мережі за допомогою новоствореного маршрутизатора усі пристрої було додано до сервісу Home Assistant, який надає безмежні можливості для автоматизації. Так, пристрої інтегровані в дане

середовище зв’язуються з сервером через MQTTброкера, який виступає мостовим з’єднанням між різними технологіями і сервісами.

Після підключення всіх пристроїв було розроблено короткий тест-план, щоб продемонструвати можливості ZigBee мережі і знайти її границю покриття всередині квартири.

Далі програмним шляхом було інтегровано та конвертовано датчики для аналізу стану грунту, після чого їх показники в реальному часі було виведено на панель керування Home Assistant.

ВИСНОВОК

1. У даній роботі було спроектовано та створено спарений ZigBee датчик вологості та рівню pH. Для даного приладу було описано послідовність дій для побудови та актуальність даного пристрою.
2. В першому розділі було описано область дослідження. Було описано що являє собою мережі ZigBee і які у неї плюси та мінуси. Також було проведено порівняння між пристроями, які використовують інші технології для зв’язку. Для кращого розуміння актуальності ZigBee технології її було порівняно з іншими конкурентними протоколами.
3. В другому розділі було виконано проектування приладу, який буде вимірювати рівень вологості грунту та показник pH автономно, не залежачи від живлення чи інших умов. Під час проектування була приведена характеристика обладнання та послідовність монтажу його на плату. Після повної зборки пристрою його було прошито під потрібні задачі.
4. В третьому розділі розглядалися методи побудови ZigBee мережі і проводився загальний її огляд. Так, для розширення покриття мережі було створено проміжний пристрій, який буде маршрутизувати трафік між пристроями навколо нього і зможе виконувати функції кінцевого пристрою за допомогою двох реле на сухих контактах, які використовуються для комутації змінного чи постійного струму.
5. Під час виконання дипломної роботи всі поставлені завдання були виконані. Було створено датчик та виконуючий пристрій, які можуть працювати автоматично без людського втручання.

Перелік використаних інформаційних джерел

1. Бездротові мережі передачі даних [Електронний ресурс] - Режим доступу: https://stud.com.ua/20630/informatika/bezdrotovi_merezhi_peredachi_danih/ (дата звертання: 11.05.2022)
2. Що таке Zigbee і чому це важливо для вашого розумного будинку?: https://xterm.com.ua/novosti/chto-takoe-zigbee-i-pochemu-eto-vazhno-dlia-vashego-umnogo-doma
3. Що таке Mesh-система? – Режим доступу: https://itc.ua/articles/about-mesh/
4. ДОМАШНЯ АВТОМАТИЗАЦІЯ [Електронний ресурс] / АТІЛОС – Режим доступу: https://www.larnitech.com/uk/solution/home-automation/
5. Водневий показник (pH-фактор) [Електронний ресурс] / Інтерв'ю з Dean Takahashi – Режим доступу: https://floragrowing.com/uk/encyclopedia/vodnevyy-pokaznyk-ph-faktor
6. Що таке Wi-Fi і як він працює? [Електронний ресурс] – Режим доступу: http://gsm-ka.com.ua/ua/chto-takoe-wi-fi-i-kak-on-rabotaet/
7. Методи вимірювання вологості грунту [Електронний ресурс] – Режим доступу: https://propozitsiya.com/ua/metody-vymiryuvannya-vologosti-gruntu
8. PTVO додаток [Електронний ресурс] – Режим доступу: https://ptvo.info/
9. ПЗ EasyEDA для створення електричних схем[Електронний ресурс] – Режим доступу: https://easyeda.com/ru
10. Home Assistant [Електронний ресурс] – Режим доступу: https://www.home-assistant.io/
11. Офіційний сайт Kodu Game Lab [Електронний ресурс] – Режим доступу: https://kodugamelab.com/
12. Офіційний сайт NeoAxis Engine [Електронний ресурс] – Режим доступу: https://www.neoaxis.com/
13. Mylopoulos, J. «Conceptual modeling and Telos1». In Loucopoulos, P.; Zicari. «Conceptual Modeling, Databases, and Case An integrated view of information systems development», New York: Wiley. pp. 49-68.
14. Alt, R., and Zimmermann, H. (2001) «Introduction to special section – business models», Electronic Markets, 11, 1, 3-9.

15. IDEF0 діаграма: приклади і правила побудови [Електронний ресурс] / Codoschool – Режим доступу: https://codoschool.ru/uk/services/idef0-diagramma-primery-i-pravila-postroeniya-metodologii-modelirovaniya.html

16. Діаграми декомпозиції: приклади і правила побудови [Електронний ресурс] / Студопедія – Режим доступу: https://studopedia.com.ua/1_162873_diagrami-dekompozitsii.html

17. Діаграма дерева вузлів: приклади і правила побудови [Електронний ресурс] / Студопедія – Режим доступу: https://studopedia.com.ua/1_162876_diagrami-dereva-vuzliv-i-FEO.html

18. Fischer, R. A., Campbell, A. J., Shofner, G. A., Lord, O. T., Dera, P., & Prakapenka, V. B. (2011). [Електронний ресурс] / Equation of state and phase diagram of FeO. Earth and Planetary Science Letters, 304(3-4), 496-502. – Режим доступу: https://doi.org/10.1016/j.epsl.2011.02.025

19. Тіллманн, Джордж (червень 1995) [Електронний ресурс] / Building a Logical Data Model – Режим доступу: https://web.archive.org/web/20080509063521/http://www.dbmsmag.com/9506d16.html

20. James Rumbaugh, Ivar Jacobson, Grady Booch (1999) «The unified modeling language reference manual», University of Toronto.

21. Benedikt Bollig (2006) «Formal Models of Communicating Systems», «Message Sequence Charts» pp. 91-95.

22. Діаграма класів: приклад побудови [Електронний ресурс] / Державний університет телекомунікацій – Режим доступу: https://dut.edu.ua/ua/news-1-626-8002-zastosuvannya-uml-chastina-3-diagrama-klasiv----class-diagram_kafedra-kompyuternih-nauk-ta-informaciynih-tehnologiy

23. Енциклопедія кібернетики : у 2 т. / за ред. В. М. Глушкова. — Київ : Гол. ред. Української радянської енциклопедії, 1973.

24. Офіційна сторінка PhpMyAdmin [Електронний ресурс] / Bringing MySQL to the web – Режим доступу: https://www.phpmyadmin.net/

25. Джон Дакетт «HTML и CSS. Разработка и дизайн веб-сайтов» (2011).

26. David Sawyer McFarland «CSS3: The Missing Manual, 3rd Edition» (2012).

27. Майк МакГрат «PHP7 для початківців с покроковими інструкціями» (2018).

28. Девід Фленаган «JavaScript: The Definitive Guide, 5th Edition» (2008)

29. Офіційна сторінка DataTables [Електронний ресурс] / DataTables – Режим доступу: https://datatables.net/

30. Model-View-Controller [Електронний ресурс] / Вікіпедія – Режим доступу: https://ru.wikipedia.org/wiki/Model-View-Controller – Загол. з екрану

31. Сайт WebGL [Електронний ресурс] / WebGL Overview - The Khronos Group Inc – Режим доступу: https://www.khronos.org/webgl/

ДОДАТОК А

КВАЛІФІКАЦІЙНА РОБОТИ

Тема: «Технологія створення датчиків вологості та рівнів pH в ґрунті із системою онлайн керування в мережі ZigBee»

Виконав: студент групи ІР-41
Шевцов Андрій
Керівник: к.т.н, доцент, Бронін Сергій

ВСТУП

Метою даної роботи є створення розумного приладу для аналізу вологості та рівню pH ґрунту, а також дослідження обладнання, яке використовується для аналізу стану ґрунту та для роботи в мережі ZigBee.

Об'єктом дослідження є прилад, який буде інтегровано в систему домашньої автоматизації.

Предметом дослідження даної роботи виступає впровадження нових IoT рішень в сферу агрокультур.

Методами дослідження є аналіз існуючих рішень автоматичного контролю стану ґрунту проектування та створення приладу для аналізу вологості та рівню pH ґрунту, інтеграції в систему домашньої автоматизації, проведення експериментів із готовою системою.

РИЗИКИ, ЯКІ ВИНИКАЮТЬ ПРИ НИЗЬКІЙ ВОЛОГОСТІ ҐРУНТУ

Важливою причиною втрати врожаю є зміна клімату, а саме: посушливість, екстремальні температури й їхній перепад, сильні зливи та урагани.

Україна вже відчула на собі наслідки кліматичних змін. Середня температура зросла на майже 1,5 °C за останні пару десятиліть, а кожен новий рік б'є температурні рекорди попереднього. Кількість днів із високими температурами (понад +30-35°C) у степовій зоні збільшилась із 30-40 до 50-65 днів щороку, а на півночі і заході – з 10 до 15-30 днів.

Аграрії відчувають зміни клімату «на собі». Збільшення частоти та інтенсивності посух стає одним із визначальних чинників врожайності культур, особливо у посушливих і напівпосушливих регіонах.

ВПЛИВ СОЛЕЙ НА РОСЛИНИ

Надлишкова концентрація солей:

- спричиняє як осмотичну дію, так і токсичну;
- порушує азотний обмін і зумовлює розпад білків;
- пригнічує ростові процеси;
- пригнічує діяльність ґрунтових мікроорганізмів (зокрема корисних);
- зумовлює уповільнене набухання і проростання насіння та, як наслідок, затримку сходів через «фізіологічну сухість» засолених ґрунтів.

До засолених ґрунтів відносять:

1) засолені ґрунти без солонцевого горизонту,
2) ґрунти з вираженим солонцевим горизонтом.

Отже, до першої групи відносяться солончаки та інші засолені ґрунти без солонцевого горизонту, а до другої – солонці та солонцюваті ґрунти.

ГОЛОВНІ КОМПОНЕНТИ СИСТЕМИ

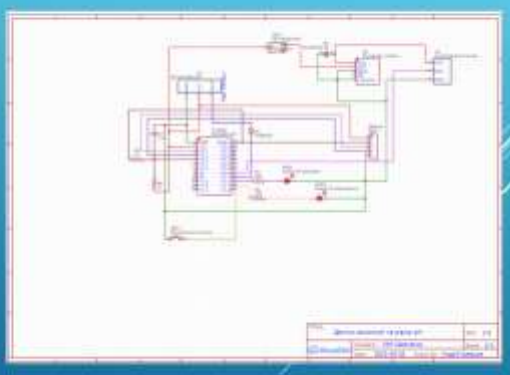
Датчик вологості ґрунту

Чіп cc2530

Датчик рівню pH

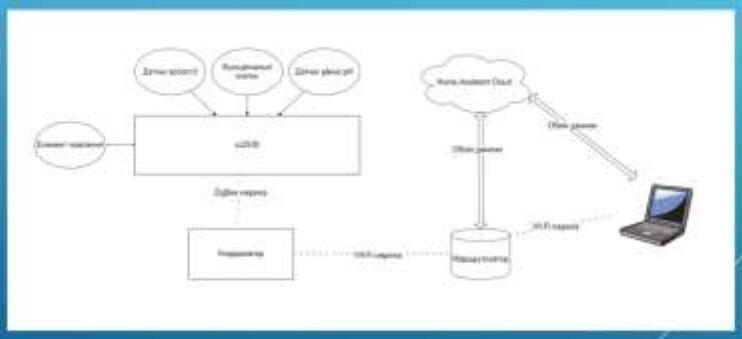
СХЕМА ПОБУДОВИ ДАТЧИКУ В СЕРЕДОВИЩІ EASY EDA

На даній схемі розведено 2 функціональні кнопки, перша заведена на звичайний порт входу/виходу і використовується для створення нового підключення до системи, другий підключено до порту скидання системи (reset). Для аналізу показників стану ґрунту було виведено 2 порти для двох датчиків: вологості та рівню pH (порти 6 та 5).



ГОТОВИЙ МАКЕТ ПРИЛАДУ ДЛЯ ТЕСТІВ

КОНЦЕПТУАЛЬНА СХЕМА МЕРЕЖІ

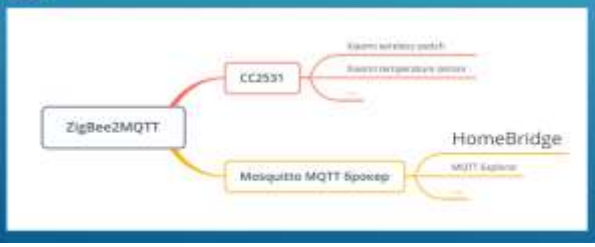


КООРДИНАТОР ТА ЙОГО ПРОШИВКА ДЛЯ ЗВ'ЯЗКУ В МЕРЕЖІ

Для прошивки було використано програмне забезпечення SerialBootTool, яке через послідовний порт дозволяє завантажувати файл прошивки у регістри пам'яті пристрою

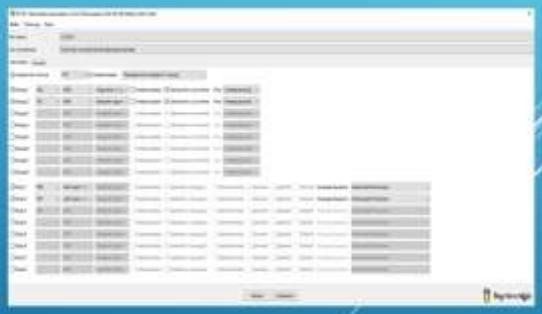
ПРИНЦИП ВЗАЄМОДІЇ ТЕХНОЛОГІЇ ZIGBEE2MQTT

Zigbee2mqtt – це міст, за допомогою якого ви можете керувати пристроями Zigbee через MQTT. Він працює на Node. js і дозволяє інтегрувати Zigbee пристрої у різні системи автоматизації: Home Assistant, Node Red тощо.

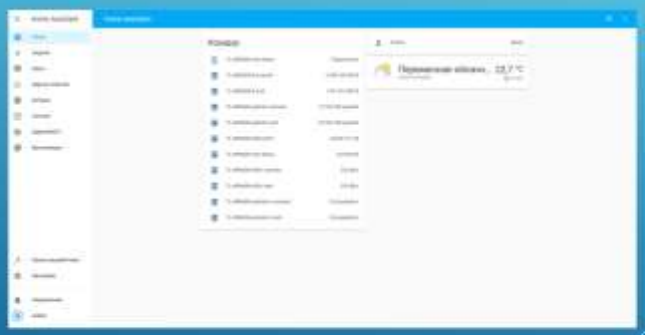


ПРОШИВКА ДАТЧИКУ

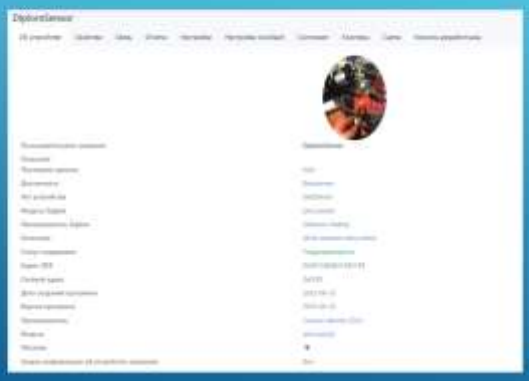
Прошивка відбувається шляхом підключення програматора до портів чіпу. Потім в графічному середовищі можна обирати кожний порт по черзі і додавати дії для нього.



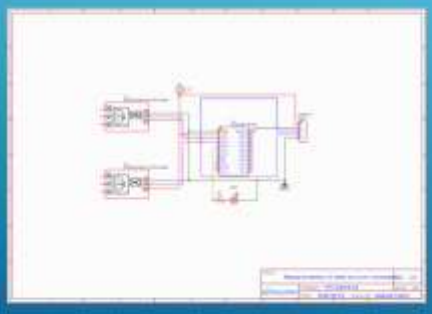
HOME ASSISTANT В ЯКОСТІ АВТОМАТИЗОВАНОЇ СИСТЕМИ



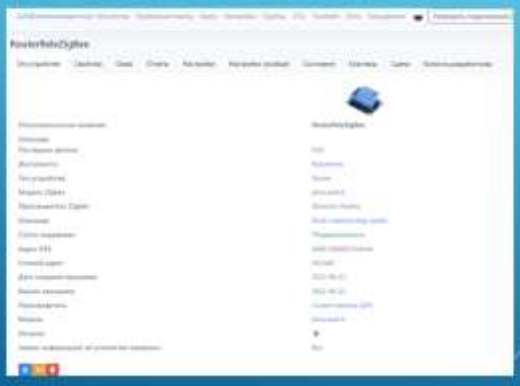
ПІДКЛЮЧЕННЯ ДО HOME ASSISTANT



СТВОРЕННЯ ДОДАТКОВОГО МАРШРУТИЗАТОРА З 2 РЕЛЕ НА СУХИХ КОНТАКТАХ ДЛЯ РОЗШИРЕННЯ МЕРЕЖІ



ІНТЕГРАЦІЯ МАРШРУТИЗАТОРА В HOME ASSISTANT



КАРТА ZIGBEE МЕРЕЖІ

ІНТЕГРАЦІЯ ДАТЧИКІВ ДО ОГЛЯДОВОГО МЕНЮ ТА КОНВЕРТАЦІЯ ДАНИХ, ОТРИМАНИХ З АНАЛОГОВОГО ДАТЧИКА

СТВОРЕННЯ СЦЕНАРІЮ АВТОМАТИЗАЦІЇ

ПОРІВНЯННЯ З АНАЛОГОМ

Wi-Fi датчик вологості

Технологія передачі даних	ZigBee	Wi-Fi
Частотний діапазон, ГГц	2,4 – 2,483	2,4 – 2,483
Пропускна спроможність, кбіт/с	250	11 000
Розмір стеку протоколів, кбайт	32-64	Більше 1000
Час неперервної роботи от акумулятора однієї ємності, дні	100-1000	0,5-5
Максимальна кількість вузлів в мережі	65 536	10
Діапазон дБ, м	10-100	20-300

ДОДАТКОВІ ДАТЧИК, ЯКІ МОЖНА ПІДКЛЮЧАТИ ДО СИСТЕМИ ЩОБ ДОДАТИ ФУНКЦІОНАЛУ

Датчик вологості повітря

Барометр

ВИКОРИСТАНІ ТЕХНОЛОГІЇ


ВИСНОВКИ
У ході виконання кваліфікаційної роботи було виконано:
1) Проектування пристрою для аналізу стану вологості грунту та рівню pH
2) Реалізація тестового зразка для повного ознайомлення з процесом розробки IoT пристроїв
3) Створено проект мережі з додатковим маршрутизатором
4) Інтегровано створені пристрої в середовище для домашньої автоматизації
5) Розроблено конвертор для аналогових значень датчиків
6) Протестовано загальну мережу та можливості mesh-мережі

ДОДАТОК Б

Код для файлу налаштування Home Assistant

```
default_config:


# Text to speech

# Loads default set of integrations. Do not remove.

mqtt:

  sensor:

    - name: "MySensor"

      state_topic: "zigbee2mqtt/0x00124b0021643149"

      unit_of_measurement: "%"

      value_template: "{{ (((2.8 - (value_json.l2|float)) / 1.6) * 100 ) | round(0) }}"

      availability_topic: "zigbee2mqtt/bridge/state"

      payload_available: "online"

      payload_not_available: "offline"

      json_attributes_topic: "zigbee2mqtt/0x00124b0021643149/attributes"

      icon: mdi:leaf


tts:

  - platform: google_translate


automation: !include automations.yaml
```

```
script: !include scripts.yaml

scene: !include scenes.yaml


sensor:
    - platform: mqtt
      name: "Humi"
      state_topic: "zigbee2mqtt/DiplomSensor"
      value_template: '{{ (((2.8 - (value_json.voltage_l2|float)) / 1.6) * 100 ) | round(0) }}'
      icon: mdi:leaf
    - platform: mqtt
      name: "humidityOf"
      state_topic: "zigbee2mqtt/DiplomSensor"
      unit_of_measurement: "%"
      value_template: "{{ (((2.8 - (value_json.voltage_l2|float)) / 1.6) * 100 ) | round(0) }}"
      availability_topic: "zigbee2mqtt/bridge/state"
      payload_available: "online"
      payload_not_available: "offline"
      json_attributes_topic: "zigbee2mqtt/DiplomSensor/attributes"
      icon: mdi:leaf
```

ДОДАТОК В

Код для створення дебагера на базі ESP8266

```
// Start addresses on DUP (Increased buffer size improves performance)
#define ADDR_BUF0              0x0000 // Buffer (512 bytes)
#define ADDR_DMA_DESC_0        0x0200 // DMA descriptors (8 bytes)
#define ADDR_DMA_DESC_1        (ADDR_DMA_DESC_0 + 8)

// DMA channels used on DUP
#define CH_DBG_TO_BUF0         0x01   // Channel 0
#define CH_BUF0_TO_FLASH       0x02   // Channel 1

// Debug commands
#define CMD_CHIP_ERASE         0x10
#define CMD_WR_CONFIG          0x19
#define CMD_RD_CONFIG          0x24
#define CMD_READ_STATUS        0x30
#define CMD_RESUME             0x4C
#define CMD_DEBUG_INSTR_1B     (0x54|1)
#define CMD_DEBUG_INSTR_2B     (0x54|2)
#define CMD_DEBUG_INSTR_3B     (0x54|3)
#define CMD_BURST_WRITE        0x80
#define CMD_GET_CHIP_ID        0x68
```

```
// Debug status bitmasks
#define STATUS_CHIP_ERASE_BUSY_BM   0x80 // New debug interface
#define STATUS_PCON_IDLE_BM         0x40
#define STATUS_CPU_HALTED_BM        0x20
#define STATUS_PM_ACTIVE_BM         0x10
#define STATUS_HALT_STATUS_BM       0x08
#define STATUS_DEBUG_LOCKED_BM      0x04
#define STATUS_OSC_STABLE_BM        0x02
#define STATUS_STACK_OVERFLOW_BM    0x01

// DUP registers (XDATA space address)
#define DUP_DBGDATA               0x6260  // Debug interface data buffer
#define DUP_FCTL                  0x6270  // Flash controller
#define DUP_FADDRL                0x6271  // Flash controller addr
#define DUP_FADDRH                0x6272  // Flash controller addr
#define DUP_FWDATA                0x6273  // Clash controller data buffer
#define DUP_CLKCONSTA             0x709E  // Sys clock status
#define DUP_CLKCONCMD             0x70C6  // Sys clock configuration
#define DUP_MEMCTR                0x70C7  // Flash bank xdata mapping
#define DUP_DMA1CFGL              0x70D2  // Low byte, DMA config ch. 1
#define DUP_DMA1CFGH              0x70D3  // Hi byte , DMA config ch. 1
#define DUP_DMA0CFGL              0x70D4  // Low byte, DMA config ch. 0
```

```
#define DUP_DMA0CFGH              0x70D5  // Low byte, DMA config ch. 0
#define DUP_DMAARM                0x70D6  // DMA arming register

// Utility macros
//! Low nibble of 16bit variable
#define LOBYTE(w)          ((unsigned char)(w))
//! High nibble of 16bit variable
#define HIBYTE(w)          ((unsigned char)(((unsigned short)(w) >> 8) & 0xFF))
//! Convert XREG register declaration to an XDATA integer address
//#define XREG(addr)       ((unsigned char volatile __xdata *) 0)[addr]
//#define FCTL             XREG( 0x6270 )
//#define XREG_TO_INT(a)     ((unsigned short)(&(a)))

// Commands to Bootloader
#define SBEGIN              0x01
#define SDATA               0x02
#define SRSP                0x03
#define SEND                0x04
#define ERRO                0x05
#define WAITING             0x00
#define RECEIVING           0x01
```

```
int DD = 14;

int DC = 4;

int RESET = 5;

int LED = 2;


/*****************************************************************************
************
 VARIABLES*/
//! DUP DMA descriptor
const unsigned char dma_desc_0[8] =
{
   // Debug Interface -> Buffer
   HIBYTE(DUP_DBGDATA),          // src[15:8]
   LOBYTE(DUP_DBGDATA),          // src[7:0]
   HIBYTE(ADDR_BUF0),            // dest[15:8]
   LOBYTE(ADDR_BUF0),            // dest[7:0]
   0,                            // len[12:8] - filled in later
   0,                            // len[7:0]
   31,                           // trigger: DBG_BW
   0x11                          // increment destination
};
//! DUP DMA descriptor
const unsigned char dma_desc_1[8] =
```

```
{
   // Buffer -> Flash controller
   HIBYTE(ADDR_BUF0),            // src[15:8]
   LOBYTE(ADDR_BUF0),             // src[7:0]
   HIBYTE(DUP_FWDATA),            // dest[15:8]
   LOBYTE(DUP_FWDATA),            // dest[7:0]
   0,                             // len[12:8] - filled in later
   0,                             // len[7:0]
   18,                            // trigger: FLASH
   0x42,                          // increment source
};

/**************************************************************************
********//**
* @brief    Writes a byte on the debug interface. Requires DD to be
*           output when function is called.
* @param    data    Byte to write
* @return   None.
**************************************************************************
***********/
#pragma inline
void write_debug_byte(unsigned char data)
{
```

```
    unsigned char i;
    for (i = 0; i < 8; i++)
    {
        // Set clock high and put data on DD line
        digitalWrite(DC, HIGH);
        if(data & 0x80)
        {
          digitalWrite(DD, HIGH);
        }
        else
        {
          digitalWrite(DD, LOW);
        }
        data <<= 1;
        digitalWrite(DC, LOW); // set clock low (DUP capture flank)
    }
}

/*****************************************************************
********//**
* @brief    Reads a byte from the debug interface. Requires DD to be
*           input when function is called.
* @return   Returns the byte read.
```

```
**********************************************************************
**********/
#pragma inline
unsigned char read_debug_byte(void)
{
   unsigned char i;
   unsigned char data = 0x00;
   for (i = 0; i < 8; i++)
   {
      digitalWrite(DC, HIGH);  // DC high
      data <<= 1;
      if(HIGH == digitalRead(DD))
      {
        data |= 0x01;
      }
      digitalWrite(DC, LOW);     // DC low
   }
   return data;
}

/**********************************************************************
********//**
* @brief    Function waits for DUP to indicate that it is ready. The DUP will
```

```
*           pulls DD line low when it is ready. Requires DD to be input when
*           function is called.
* @return   Returns 0 if function timed out waiting for DD line to go low
* @return   Returns 1 when DUP has indicated it is ready.
*****************************************************************
************/
#pragma inline
unsigned char wait_dup_ready(void)
{
   // DUP pulls DD low when ready
   unsigned int count = 0;
   while ((HIGH == digitalRead(DD)) && count < 16)
   {
      // Clock out 8 bits before checking if DD is low again
      read_debug_byte();
      count++;
   }
   return (count == 16) ? 0 : 1;
}

/*****************************************************************
********//**
* @brief    Issues a command on the debug interface. Only commands that return
```

```
*          one output byte are supported.
* @param    cmd            Command byte
* @param    cmd_bytes      Pointer to the array of data bytes following the
*                          command byte [0-3]
* @param    num_cmd_bytes  The number of data bytes (input to DUP) [0-3]
* @return   Data returned by command
**********************************************************************
***********/
unsigned char debug_command(unsigned char cmd, unsigned char *cmd_bytes,
                            unsigned short num_cmd_bytes)
{
   unsigned short i;
   unsigned char output = 0;
   // Make sure DD is output
   pinMode(DD, OUTPUT);
   // Send command
   write_debug_byte(cmd);
   // Send bytes
   for (i = 0; i < num_cmd_bytes; i++)
   {
      write_debug_byte(cmd_bytes[i]);
   }
   // Set DD as input
```

```
    pinMode(DD, INPUT);
    digitalWrite(DD, HIGH);
    // Wait for data to be ready
    wait_dup_ready();
    // Read returned byte
    output = read_debug_byte();
    // Set DD as output
    pinMode(DD, OUTPUT);

    return output;
}

/*****************************************************************************
********//**
* @brief    Resets the DUP into debug mode. Function assumes that
*           the programmer I/O has already been configured using e.g.
*           ProgrammerInit().
* @return   None.
*****************************************************************************
***********/
void debug_init(void)
{
    volatile unsigned char i;
```

```
    // Send two flanks on DC while keeping RESET_N low
    // All low (incl. RESET_N)
    digitalWrite(DD, LOW);
    digitalWrite(DC, LOW);
    digitalWrite(RESET, LOW);
    delay(10);   // Wait
    digitalWrite(DC, HIGH);                 // DC high
    delay(10);   // Wait
    digitalWrite(DC, LOW);                  // DC low
    delay(10);   // Wait
    digitalWrite(DC, HIGH);                 // DC high
    delay(10);   // Wait
    digitalWrite(DC, LOW);                  // DC low
    delay(10);   // Wait
    digitalWrite(RESET, HIGH);              // Release RESET_N
    delay(10);   // Wait
}

/**************************************************************************
********//**
* @brief    Reads the chip ID over the debug interface using the
*           GET_CHIP_ID command.
```

```
* @return   Returns the chip id returned by the DUP
*****************************************************************
************/
unsigned char read_chip_id(void)
{
   unsigned char id = 0;

   // Make sure DD is output
   pinMode(DD, OUTPUT);
   delay(1);
   // Send command
   write_debug_byte(CMD_GET_CHIP_ID);
   // Set DD as input
   pinMode(DD, INPUT);
   digitalWrite(DD, HIGH);
   delay(1);
   // Wait for data to be ready
   if(wait_dup_ready() == 1)
   {
     // Read ID and revision
     id = read_debug_byte(); // ID
     read_debug_byte();      // Revision (discard)
   }
```

```
    // Set DD as output
    pinMode(DD, OUTPUT);

    return id;
}

/*****************************************************************
********//**
* @brief    Sends a block of data over the debug interface using the
*           BURST_WRITE command.
* @param    src         Pointer to the array of input bytes
* @param    num_bytes   The number of input bytes
* @return   None.
*****************************************************************
************/
void burst_write_block(unsigned char *src, unsigned short num_bytes)
{
    unsigned short i;

    // Make sure DD is output
    pinMode(DD, OUTPUT);

    write_debug_byte(CMD_BURST_WRITE | HIBYTE(num_bytes));
```

```
    write_debug_byte(LOBYTE(num_bytes));
    for (i = 0; i < num_bytes; i++)
    {
        write_debug_byte(src[i]);
    }

    // Set DD as input
    pinMode(DD, INPUT);
    digitalWrite(DD, HIGH);
    // Wait for DUP to be ready
    wait_dup_ready();
    read_debug_byte(); // ignore output
    // Set DD as output
    pinMode(DD, OUTPUT);
}

/**************************************************************************
********//**
* @brief    Issues a CHIP_ERASE command on the debug interface and waits for it
*           to complete.
* @return   None.
**************************************************************************
**********/
```

```
void chip_erase(void)
{
   volatile unsigned char status;
   // Send command
   debug_command(CMD_CHIP_ERASE, 0, 0);

   // Wait for status bit 7 to go low
   do {
      status = debug_command(CMD_READ_STATUS, 0, 0);
   } while((status & STATUS_CHIP_ERASE_BUSY_BM));
}

/**************************************************************************
********//**
* @brief    Writes a block of data to the DUP's XDATA space.
* @param    address     XDATA start address
* @param    values      Pointer to the array of bytes to write
* @param    num_bytes   Number of bytes to write
* @return   None.
**************************************************************************
***********/
void write_xdata_memory_block(unsigned short address,
                              const unsigned char *values,
```

```
                        unsigned short num_bytes)
{
    unsigned char instr[3];
    unsigned short i;

    // MOV DPTR, address
    instr[0] = 0x90;
    instr[1] = HIBYTE(address);
    instr[2] = LOBYTE(address);
    debug_command(CMD_DEBUG_INSTR_3B, instr, 3);

    for (i = 0; i < num_bytes; i++)
    {
        // MOV A, values[i]
        instr[0] = 0x74;
        instr[1] = values[i];
        debug_command(CMD_DEBUG_INSTR_2B, instr, 2);

        // MOV @DPTR, A
        instr[0] = 0xF0;
        debug_command(CMD_DEBUG_INSTR_1B, instr, 1);
```

```
        // INC DPTR
        instr[0] = 0xA3;
        debug_command(CMD_DEBUG_INSTR_1B, instr, 1);
    }
}

/*****************************************************************************
********//**
* @brief    Writes a byte to a specific address in the DUP's XDATA space.
* @param    address     XDATA address
* @param    value       Value to write
* @return   None.
*****************************************************************************
*************/
void write_xdata_memory(unsigned short address, unsigned char value)
{
    unsigned char instr[3];

    // MOV DPTR, address
    instr[0] = 0x90;
    instr[1] = HIBYTE(address);
    instr[2] = LOBYTE(address);
    debug_command(CMD_DEBUG_INSTR_3B, instr, 3);
```

```
    // MOV A, values[i]
    instr[0] = 0x74;
    instr[1] = value;
    debug_command(CMD_DEBUG_INSTR_2B, instr, 2);

    // MOV @DPTR, A
    instr[0] = 0xF0;
    debug_command(CMD_DEBUG_INSTR_1B, instr, 1);
}

/**************************************************************************
********//**
* @brief    Read a byte from a specific address in the DUP's XDATA space.
* @param    address     XDATA address
* @return   Value read from XDATA
**************************************************************************
**********/
unsigned char read_xdata_memory(unsigned short address)
{
    unsigned char instr[3];

    // MOV DPTR, address
```

```
    instr[0] = 0x90;
    instr[1] = HIBYTE(address);
    instr[2] = LOBYTE(address);
    debug_command(CMD_DEBUG_INSTR_3B, instr, 3);

    // MOVX A, @DPTR
    instr[0] = 0xE0;
    return debug_command(CMD_DEBUG_INSTR_1B, instr, 1);
}

/*****************************************************************************
********//**
* @brief    Reads 1-32767 bytes from DUP's flash to a given buffer on the
*           programmer.
* @param    bank        Flash bank to read from [0-7]
* @param    address     Flash memory start address [0x0000 - 0x7FFF]
* @param    values      Pointer to destination buffer.
* @return   None.
*****************************************************************************
***********/
void read_flash_memory_block(unsigned char bank,unsigned short flash_addr,
                             unsigned short num_bytes, unsigned char *values)
{
```

```
unsigned char instr[3];
unsigned short i;
unsigned short xdata_addr = (0x8000 + flash_addr);

// 1. Map flash memory bank to XDATA address 0x8000-0xFFFF
write_xdata_memory(DUP_MEMCTR, bank);

// 2. Move data pointer to XDATA address (MOV DPTR, xdata_addr)
instr[0] = 0x90;
instr[1] = HIBYTE(xdata_addr);
instr[2] = LOBYTE(xdata_addr);
debug_command(CMD_DEBUG_INSTR_3B, instr, 3);

for (i = 0; i < num_bytes; i++)
{
   // 3. Move value pointed to by DPTR to accumulator (MOVX A, @DPTR)
   instr[0] = 0xE0;
   values[i] = debug_command(CMD_DEBUG_INSTR_1B, instr, 1);

   // 4. Increment data pointer (INC DPTR)
   instr[0] = 0xA3;
   debug_command(CMD_DEBUG_INSTR_1B, instr, 1);
```

```
    }
}

/*****************************************************************************
********//**
* @brief    Writes 4-2048 bytes to DUP's flash memory. Parameter \c num_bytes
*           must be a multiple of 4.
* @param    src         Pointer to programmer's source buffer (in XDATA space)
* @param    start_addr  FLASH memory start address [0x0000 - 0x7FFF]
* @param    num_bytes   Number of bytes to transfer [4-1024]
* @return   None.
*****************************************************************************
************/
void write_flash_memory_block(unsigned char *src, unsigned long start_addr,
                              unsigned short num_bytes)
{
    // 1. Write the 2 DMA descriptors to RAM
    write_xdata_memory_block(ADDR_DMA_DESC_0, dma_desc_0, 8);
    write_xdata_memory_block(ADDR_DMA_DESC_1, dma_desc_1, 8);

    // 2. Update LEN value in DUP's DMA descriptors
    unsigned char len[2] = {HIBYTE(num_bytes), LOBYTE(num_bytes)};
```

```
   write_xdata_memory_block((ADDR_DMA_DESC_0+4), len, 2);  // LEN, DBG
=> ram
   write_xdata_memory_block((ADDR_DMA_DESC_1+4), len, 2);  // LEN, ram
=> flash

   // 3. Set DMA controller pointer to the DMA descriptors
   write_xdata_memory(DUP_DMA0CFGH, HIBYTE(ADDR_DMA_DESC_0));
   write_xdata_memory(DUP_DMA0CFGL, LOBYTE(ADDR_DMA_DESC_0));
   write_xdata_memory(DUP_DMA1CFGH, HIBYTE(ADDR_DMA_DESC_1));
   write_xdata_memory(DUP_DMA1CFGL, LOBYTE(ADDR_DMA_DESC_1));

   // 4. Set Flash controller start address (wants 16MSb of 18 bit address)
   write_xdata_memory(DUP_FADDRH, HIBYTE( (start_addr)));//>>2) ));
   write_xdata_memory(DUP_FADDRL, LOBYTE( (start_addr)));//>>2) ));

   // 5. Arm DBG=>buffer DMA channel and start burst write
   write_xdata_memory(DUP_DMAARM, CH_DBG_TO_BUF0);
   burst_write_block(src, num_bytes);

   // 6. Start programming: buffer to flash
   write_xdata_memory(DUP_DMAARM, CH_BUF0_TO_FLASH);
   write_xdata_memory(DUP_FCTL, 0x0A);//0x06
```

```
    // 7. Wait until flash controller is done
    while (read_xdata_memory(DUP_FCTL) & 0x80);
}

void RunDUP(void)
{
  volatile unsigned char i;

  // Send two flanks on DC while keeping RESET_N low
  // All low (incl. RESET_N)
  digitalWrite(DD, LOW);
  digitalWrite(DC, LOW);
  digitalWrite(RESET, LOW);
  delay(10);   // Wait

  digitalWrite(RESET, HIGH);
  delay(10);   // Wait
}

void ProgrammerInit(void)
{
  pinMode(DD, OUTPUT);
```

```
  pinMode(DC, OUTPUT);
  pinMode(RESET, OUTPUT);
  pinMode(LED, OUTPUT);
  digitalWrite(DD, LOW);
  digitalWrite(DC, LOW);
  digitalWrite(RESET, HIGH);
  digitalWrite(LED, LOW);
}

void setup()
{
  ProgrammerInit();
  Serial.begin(115200);
  // If using Leonado as programmer,
  //it should add below code,otherwise,comment it.
  while(!Serial);
}

void loop()
{
  unsigned char chip_id = 0;
  unsigned char debug_config = 0;
```

```
unsigned char Continue = 0;
unsigned char Verify = 0;

while(!Continue)     // Wait for starting
{

 if(Serial.available()==2)
 {
   if(Serial.read() == SBEGIN)
   {
     Verify = Serial.read();
     Continue = 1;
   }
   else
   {
     Serial.read(); // Clear RX buffer
   }
 }
}

debug_init();
chip_id = read_chip_id();
```

```
if(chip_id == 0)
{
  Serial.write(ERRO);
  return; // No chip detected, run loop again.
}

RunDUP();
debug_init();

chip_erase();
RunDUP();
debug_init();

// Switch DUP to external crystal osc. (XOSC) and wait for it to be stable.
// This is recommended if XOSC is available during programming. If
// XOSC is not available, comment out these two lines.
write_xdata_memory(DUP_CLKCONCMD, 0x80);
while (read_xdata_memory(DUP_CLKCONSTA) != 0x80);//0x80)

// Enable DMA (Disable DMA_PAUSE bit in debug configuration)
debug_config = 0x22;
debug_command(CMD_WR_CONFIG, &debug_config, 1);
```

```
// Program data (start address must be word aligned [32 bit])
Serial.write(SRSP);    // Request data blocks
digitalWrite(LED, HIGH);
unsigned char Done = 0;
unsigned char State = WAITING;
unsigned char  rxBuf[514];
unsigned int BufIndex = 0;
unsigned int addr = 0x0000;
while(!Done)
{
  while(Serial.available())
  {
    unsigned char ch;
    ch = Serial.read();
    switch (State)
    {
      // Bootloader is waiting for a new block, each block begin with a flag byte
      case WAITING:
      {
        if(SDATA == ch)  // Incoming bytes are data
        {
```

```
    State = RECEIVING;
  }
  else if(SEND == ch)   // End receiving firmware
  {
    Done = 1;           // Exit while(1) in main function
  }
  break;
}
// Bootloader is receiving block data
case RECEIVING:
{
  rxBuf[BufIndex] = ch;
  BufIndex++;
  if (BufIndex == 514) // If received one block, write it to flash
  {
    BufIndex = 0;
    uint16_t CheckSum = 0x0000;
    for(unsigned int i=0; i<512; i++)
    {
      CheckSum += rxBuf[i];
    }
    uint16_t CheckSum_t = rxBuf[512]<<8 | rxBuf[513];
```

```
if(CheckSum_t != CheckSum)
{
  State = WAITING;
  Serial.write(ERRO);
  chip_erase();
  return;
}
write_flash_memory_block(rxBuf, addr, 512); // src, адреса та лічильник
if(Verify)
{
  unsigned char bank = addr / (512 * 16);
  unsigned int  offset = (addr % (512 * 16)) * 4;
  unsigned char read_data[512];
  read_flash_memory_block(bank, offset, 512, read_data); // Банк, Адреса, лічильник, пункт призначення.
  for(unsigned int i = 0; i < 512; i++)
  {
    if(read_data[i] != rxBuf[i])
    {
      // Fail
      State = WAITING;
      Serial.write(ERRO);
      chip_erase();
```

```
                return;
              }
            }
          }
          addr += (unsigned int)128;
          State = WAITING;
          Serial.write(SRSP);
        }
        break;
      }
      default:
        break;
    }
  }
}

digitalWrite(LED, LOW);
RunDUP();
}
```