Міністерство освіти і науки України
Одеський національний технологічний університет

Кафедра Автоматизації технологічних процесів і робототехнічних систем

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА
на тему Підвищення ефективності моніторингу умов вирощування рослин у закритому ґрунті

(назва кваліфікаційної роботи згідно наказу ОНТУ)

Здобувача (ки)Удовіка І.О.

(прізвище, ініціали)

2 Курсу АМЗ-20 групи

Керівник к.т.н., доцент Мазур О.В.

(посада, прізвище та ініціали)

Консультанти: к.т.н., доцент Мазур О.В.

(посада, прізвище та ініціали)

_____

(посада, прізвище та ініціали)

Кваліфікаційна робота допускається до захисту

Рішеннякафедривід 04.12 2023р.,протокол№ 1.

Завідувач(ка)кафедри АТПіРС _____ І.М Світий

(назва кафедри) (підпис) (Ім'я ПРІЗВИЩЕ)

Одеса-2023рік

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет	Автоматизації та робототехніки
Кафедра	Автоматизації технологічних процесів і робототехнічних систем
Ступінь вищої освіти	Магістр
Спеціальність	151 – Автоматизація та комп'ютерно-інтегровані технології
Освітня програма	Комп'ютерні системи та програмна інженерія в автоматизації

ЗАТВЕРДЖУЮ

Зав. Кафедри АТПіРС

В.А.Хобін

«_____» _____ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

Удовіка Ігоря Олександровича

1. Тема роботи «Підвищення ефективності моніторингу умов вирощування рослин у закритому грунті»

Затверджена наказом ОНТУ від 29.11.22 р. наказ №877-03

2. Термін здачі здобувачем закінченої роботи 14.12.2023

3. Вихідні дані до випускної роботи: матеріали виконаних індивідуальних завдань (ІЗ) виробничої практики, дипломної роботи бакалавра, дослідницької практики, курсових та самостійних робіт, виконаних відповідно до ІЗ.

4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити):

Вступ (актуальність роботи, зв'язок з науковими напрямками робіт академії, мета та задачі дослідження, об'єкт та предмет дослідження, методи дослідження, наукова новизна отриманих результатів, практичне значення отриманих результатів, апробація результатів роботи, публікації, структура та об'єм роботи).

Розділ 1. Аналіз існуючих систем моніторингу умов вирощування рослин у закритому ґрунті та постановка задач дослідження.

Розділ 2.Розробка архітектури та програмного забезпечення бездротової мережі контролю параметрів мікроклімату у культиваційних спорудах закритого ґрунту на базі мікроконтролерів Tehas Instruments CC1350 SimpleLink™ Ultra-Low-PowerDual-Band.

Розділ 3.Розробка методики вимірювання та програмного забезпечення спектрометра для контролю потужності спектральних складових фізіологічно активної радіації на базі комплекту оптичних сенсорівAS7265x виробництва ams-OSRAM AG.

Розділ 4. Розробка алгоритмів і програмного забезпечення системи аналізу та підтримки прийняття рішень для підвищення енергоефективності процесу опромінення рослин фізіологічно активною радіацією.

Додатки (допоміжні матеріали, ксерокопії програм конференцій, статей, патентів).

6. Консультанти по роботі, із зазначенням розділів роботи, що стосуються їх

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розробка архітектури та програмного забезпечення бездротової мережі контролю параметрів мікроклімату у культиваційних спорудах закритого ґрунту на базі мікроконтролерів Tehas Instruments CC1350 SimpleLink™ Ultra-Low-PowerDual-Band	Степанов М.Т. к.т.н. доцент каф. АТПіРС		
Розробка методики вимірювання та програмного забезпечення спектрометра для контролю потужності спектральних складових фізіологічно активної радіації на базі комплекту оптичних сенсорівAS7265x виробництва ams-OSRAM AG	Мазур О.В. к.т.н. доцент каф. АТПіРС		

7. Дата видачі завдання23.12.2022 .

Керівник МазурО.В.

Завдання прийняв до виконання УдовікаІ.О.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи магістра	Термін виконання етапів роботи	Примітка
1	Вступ та загальна характеристика роботи	01.05.2023	
2	Аналіз існуючих систем моніторингу умов вирощування рослин у закритому ґрунті та постановка задач дослідження	05.06.2023	
3	Розробка архітектури та програмного забезпечення бездротової мережі контролю параметрів мікроклімату у культиваційних спорудах закритого ґрунту на базі мікроконтролерів TehasInstruments CC1350 SimpleLink™ Ultra-Low-PowerDual-Band	01.08.2023	
4	Розробка методики вимірювання та програмного забезпечення спектрометра для контролю потужності спектральних складових фізіологічно активної радіації на базі комплекту оптичних сенсорівAS7265x виробництва ams-OSRAM AG	01.10.2023	
5	Розробка алгоритмів і програмного забезпечення системи аналізу та підтримки прийняття рішень для підвищення енергоефективності процесу опромінення рослин фізіологічно активною радіацією	01.12.2023	
6			

Здобувач-дипломник Удовіка І.О. Керівник роботи МазурО.В.

Несу відповідальність за ідентичність електронного та друкованого варіантів кваліфікаційної роботи, даю згоду на обробку персональних даних та не заперечую проти розміщення кваліфікаційної роботи на офіційних web-ресурсах ОНТУ.

Підтверджую, що в кваліфікаційній роботі відсутні порушення норм академічної доброчесності.

Здобувач-дипломник Удовіка І.О.

ПІБ Підпис

АНОТАЦІЯ

Удовіка І.О. «Підвищення ефективності моніторингу умов вирощування рослин у закритому грунті».

Кваліфікаційна робота магістра. – Одеса: ОНТУ 2023. 93– с. Бібліогр.:_22_. Іл.:_51_. Табл._9_.

Основною метою роботи є підвищення ефективності процесу моніторингу параметрів мікроклімату в культиваційних спорудах закритого ґрунту.

В теплицях активно використовуються системи автоматичного керування мікрокліматом, але в своїй більшості вони є системами керування об’єктами з розосередженими параметрами, а системи керування освітленням взагалі в більшості випадків є розімкнутою, тобто керування відбувається по моделі без зворотного зв’язку. Температура, вологість, вміст вуглекислого газу та світла розподіляються нерівномірно по площі теплиці, що може негативно впливати на якість та кількість вирощеної продукції, і питомі енергетичні витрати на одиницю продукції. Ця проблема поглиблюється за необхідності вирощування декількох культур в межах однієї теплиці.

Проведено аналіз існуючих систем моніторингу умов вирощування рослин у закритому ґрунті, виявлено основні недоліки та визначені шляхи їх вирішення.

Для підвищення ефективності моніторингу умов вирощування рослин у закритому ґрунті розроблені архітектура та програмне забезпечення бездротової мережі контролю розподілених параметрів мікроклімату.

Запропонована методика вимірювання та розроблене програмне забезпечення спектрометра для контролю потужності спектральних складових фізіологічно активної радіації на базі комплекту оптичних сенсорів AS7265x виробництва ams-OSRAM AG

Розроблене програмне забезпечення для інтеграції системи бездротової мережі теплиці в загальну систему автоматичного керування виробничими процесами на базі протоколів верхнього рівня архітектури OPC UA.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CSMA-CA - Carrier Sense Multiple Access With - Collision Avoidance

CoDeSys PLC - Controller Development System, Programmable Logic Controller

FTP - File Transfer Protocol

FWHM -Full Width at Half Maximum

HTTPs - Hyper Text Transfer Protocol secure

I2C – Inter-Integrated Circuit Protocol

IDE - Integrated Development Environment

IEEE - Institute of Electrical and Electronics Engineers

IIoT – Industrial Internet of Things

IoT – Internet of Things

IP – Internet protocol

MAC- Media Access Control

MCU – Micro Controller Unit

NPI - Network Processor Interface

OPC UA - Open Platform Communications Unified Architecture

OSI - Open Systems Interconnection

POP3 - Post Office Protocol version 3

PAR - Photosynthetic Active Radiation

PPF - Photosynthetic Photon Flux

PPFD - Photosynthetic Photon Flux Density

PSDU - Physical Layer Service Data Unit

REST - Representational State Transfer

SCADA - Supervisory Control And Data Acquisition

SCL - Serial Clock

SDA - Serial Data

SDK – Software Development Kit

SOAP - Simple Object Access Protocol

SMTP - Simple Mail Transfer Protocol

SPI – Serial Peripheral Interface

TCP – Transmission Control Protocol

UART – Universal Asynchronous Receiver/Transmitter

UDP – User Datagram

USB – Universal Serial Bus

WDSL - Web Services Description Language

ВСТУП

Різкий ріст населення планети обумовлює необхідність збільшення виробництва сільськогосподарської продукції, що в свою чергу обумовлює необхідність підвищення ефективності ведення сільського господарства. Це відбувається на фоні погіршення екологічних та кліматичних умов, що підвищує роль та частку вирощування сільськогосподарських культур в закритому ґрунті де є більш передбачувані та контрольовані кліматичні умови. Сьогодні, з появою нових світлодіодних світильників з'явились нові можливості підвищування ефективності вирощування рослин за рахунок керування рівнем потужності та спектром опромінення рослин враховуючий їх фізіологічні потреби.

Підтримка мікроклімату є досить складною задачею, вже більш ніж 50 років технології автоматичного керування працюють в культиваційних спорудах закритого ґрунту. Але й досі ефективність вирощування культур в теплицях залишається відносно низькою. Теплиця є досить дорогою спорудою, але для впровадження розподілених систем моніторингу параметрів мікроклімату останнього часу потрібні були такі фінансові витрати, що робили процес вирощування рослин в них не рентабельним. Теплиця с точки зору керування мікрокліматом є об'єктом з розподіленими параметрами, тому основні його показники будуть різними в різних частинах споруди. Так наприклад температура в різних зонах теплиці може різнитися на декілька градусів, що може призводити до зменшення врожайності та якості продукції рослинництва, чи просто до неефективних витрат на обігрів приміщення. Створення дротової мережі температурних датчиків, датчиків вологості та вмісту вуглекислого газу потребує капітальних витрат, що збільшує ціну та терміни окупності.

Ще гіршою є ситуація з моніторингом опромінення рослин фізіологічно активною радіацію. Фермери витрачають величезні гроші на облаштування теплиць ліхтарями різного типу та спектру, але не мають можливості

забезпечити ефективне освітлення рослин та скоротити витрати електрики на зайву роботу ліхтарів, бо без ефективного контролю це є нерозв'язним задаванням. Спектрометри дозволяють досить точно вирахувати рівень опромінення електромагнітною світловою енергією, але коштують дуже дорого в розрізі створення мережі спектральних датчиків. В свою чергу, люксметри не дозволяють отримати точного результату для системи керуванням зворотним зв’язком.

Але час не стоїть на місці, з’являються відносно дешеві технології спектрального аналізу. Розвиток мережевих технологій дозволяє змінити дорогі та не гнучкі для змін положень датчиків дротові мережі на нові бездротові системи. Це відкриває нові можливості суттєво підвищити ефективність моніторингу умов вирощування рослин у закритому ґрунті, та є перспективним напрямком науково-прикладних досліджень.

АНАЛІЗ ІСНУЮЧИХ СИСТЕМ МОНІТОРИНГУ УМОВ ВИРОЩУВАННЯ РОСЛИН У ЗАКРИТОМУ ҐРУНТІ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Визначення закритого ґрунту та класифікація культиваційних споруд закритого ґрунту.

Закритий ґрунт – це земельна ділянка, або спеціальне приміщення, обладнане для створення сприятливого мікроклімату, покращення природних умов з метою поза сезонного(або більш ефективного в межах сезону) вирощування рослинних культур.

Рис.1.1 - Теплиця

В закритому ґрунті вирощуються різноманітні овочеві культури: огірок, помідор, салат листовий та качаний, перець, баклажан, цибуля зелена, петрушка, селера, кріп, шпинат, редиска, кавун, диня, щавель, ревінь, спаржа, мангольд тощо. Окрім овочів, у теплицях вирощують і квіти, декоративні рослини, ягоди. Основними факторами інтенсифікації в тепличному овочівництві є, перш за все, параметри мікроклімату: добрива, харчування, сортовий склад, регулювання температури, освітлення та вологості в

оптимальних для кожної культури показниках. Регулювання мікроклімату при цьому проводиться в автоматичному та/або ручному режимах.

1.1.2 Класифікація культиваційних споруд закритого ґрунту.

Усі культиваційні споруди закритого ґрунту можна розділити на:

- утеплений ґрунт
- парники
- теплиці
- фітотрони/темниці(для вирощування рослин в умовах штучного освітлення).

1.1.2.1 Утеплений ґрунт - це штучне утворення для підвищення температури в зоні росту рослин, його умовно відносять також до культиваційних споруд закритого ґрунту. Раннє вирощування овочів та інших городніх культур відоме ще у XVI столітті. Тоді воно переважно зосереджувалося в Нідерландах. Не маючи дров, голландці вперше почали застосовувати гній для обігріву грядок. Утеплений ґрунт є найдавнішим типом культиваційних споруд. Це найпростіший і найбільш доступний для кожного овочівника спосіб вирощування ранніх овочів у ранньовесняний період. Для обігріву ґрунту використовують біопаливо, а також пару, гарячу воду або електричні нагрівальні прилади. Незважаючи на значний прогрес у розвитку суспільства, цей спосіб підігріву ґрунту та раннього вирощування овочів застосовується і до теперішнього часу.

1.1.2.2 Парник - неопалювана споруда для захисту культурних рослин від зовнішнього впливу. У широкому розумінні слова – це та ж теплиця, але відрізняється можливістю перебувати у теплиці людині на повен зріст. У той же час парники та плівкові тунелі мають лише таку висоту, яка дозволяє забезпечити нормальне зростання рослин. Вирощування буває подвійне: рослини висіють у парники і залишають у них до повного дозрівання або тільки протягом ранньої весни, а потім пересаджують на відкриті грядки.

1.1.2.3 Теплиця – найпоширеніший вид стаціонарних культиваційних споруд. Регулювання в теплицях параметрів мікроклімату, за яких вирощують культури (температура, харчування рослин, вологість, вміст вуглекислоти в повітрі і світло), забезпечує високу продуктивність отримання врожаїв культурних рослин.

1.1.2.4 Фітотрони/темниці - теплиці з повністю ізольованою системою створення мікроклімату. В останні часи за умов високої ціни генерації тепла та розвитку приборів штучного освітлення набувають розповсюдження вирощування рослин в спеціалізованих темних закритих приміщеннях. Завдяки капітальній формі споруд та майже повною ізоляцією від зовнішнього середовища вдається суттєво підвищити контрольованість параметрів мікроклімату та значно заощадити на опаленні та інших системах штучного створення мікроклімату.

1.1.2.5 Гроубокс (Growbox, англ.- ящик для вирощування Рис. 1.1.2.5) – обладнання підтримки локального мікроклімату для вирощування окремої рослини чи невеличкої групи рослин завдяки оснащенню системою освітлення, системою вентиляції, системою повітряної фільтрації, системою зволоження та контролю температури, та системою насичення вуглекислим газом. Дозволяє досить точно регулювати мікроклімат і підтримувати сприятливі умови середовища. Найчастіше в гроубоксах використовуються ґрунтові суміші або гідропонні системи. Є своєрідним окремим випадком споруди описаної в попередньому пункті в мініатюрі. Часто темниці проектують з використанням горубоксів як складових модулів всієї великої системи.

Рис. 1.1.2.5 - Гроубокс.

1.2 Параметри та характеристики контролю мікроклімату у культиваційних спорудах закритого ґрунту.

Мікроклімат – це кліматичні умови, що склалися в конкретному, штучно обмеженому просторі, наприклад у теплиці. Визначають мікроклімат більш ніж 20 показників, основні з них:

- температура
- вологість повітря
- вологість ґрунту
- вміст CO_2 в повітрі
- освітлення рослин в теплиці

1.2.1 Температурні потреби рослин дуже різні. По відношенню до комфортних теплових режимів рослини поділяють на такі групи:

Зимостійкі(або морозостійкі) – ці багаторічні культури можуть пристосовуватися до знижених температур і благополучно зимувати навіть без укриття. Це часник, ревінь, щавель, тощо.

Холодостійкі – одно, дво та багаторічні рослини, що витримують короткострокове похолодання до -5 °С або більш тривале охолодження до -2 °С. При цьому вони активно розвиваються при 18-23 °С, але в жарких умовах (вище 30 ˚С) почуваються некомфортно. Серед них: коренеплідні та капустяні культури, салати, цибуля, цибуля-порей, деякі бобові (горох) , тощо.

Теплолюбні – для таких рослин критичним буде тривале зниження температури повітря нижче 10-15 ˚С, а при 0 ˚С і менше настає загибель. Оптимальним вважається діапазон 20-30 °С, а з підвищенням температури до 35-40 °С їх зростання і розвиток припиняються. До цієї групи належать томати, огірки, перці, кукурудза, кабачки, тощо.

Жаростійкі – сприятлива для них температура 30 ˚С, максимальна – 40 ˚С. В їх числі: баклажани, дині, кавуни, тощо

Ця класифікація дозволяє визначити загальні потреби культури, що вирощується в теплі. Для формування більш детальної картини необхідно враховувати реакцію окремих сортів на добові коливання та співвідношення температури ґрунту та повітря, стійкість до екстремальних температур.

Клімат у нашому регіоні загалом характеризується досить тривалими та холодними зимами з обмеженою кількістю світла. Літо іноді спекотне та сухе. Теплиці витримують відмінності в температурі до 50 °С. Для підвищення температури внутрішнього повітря в теплицях використовуюсь системи штучного опалення. Для зниження температури повітря в теплицях влітку використовують системи кондиціювання, та додаткові теплозахисні та затемнювані екрани.

1.2.2 У закритому ґрунті необхідно збалансувати вологість ґрунту та повітря. Поглинання більшої частини води рослинами відбувається крізь кореневу систему та супроводжується транспірацією (випаровуванням) через листя та стебла. Інтенсивність цих взаємозалежних процесів коливається як у різні фази розвитку рослин, і протягом доби. Для забезпечення вологою в

теплицях рекомендується використовувати систему крапельного зрошення, що дозволяє подавати воду точково під корінь. В умовах штучного обігріву та високої сонячної активності відносна вологість повітря в теплиці знижується, а норми поливу збільшують через прискорення транспірації та висихання ґрунту. У прохолодну пору року необхідність у поливах скорочується. Не можна допускати надмірного ґрунтового зволоження, щоб уникнути кисневого голодування та розвитку кореневої гнилі. Надлишок вологи у повітрі сприяє появі грибкових захворювань. Зменшити вологість допомагає вентилювання, додатковий обігрів теплиці, кондиціювання, скорочення числа та норми поливів. Для поливу та обприскування не можна використовувати холодну воду, щоб не допустити розвитку хвороб. Вологість ґрунту має коригуватися залежно від температурного, світлового та поживного режиму на кожному етапі росту рослин.

1.2.3 Склад та стан повітряного середовища. Вуглекислий газ (CO_2), необхідний для життєдіяльності рослин. Низький рівень CO_2 пригнічує зростання та знижує врожайність. Рослини дуже швидко поглинають весь доступний обсяг CO_2. Відповідно, щоб уникнути занадто сильного падіння рівня CO_2 та врожайності використовують системи регулювання насиченістю CO_2 в повітрі теплеці шляхом штучного збагачення. Штучне насичення CO_2 відбувається або додаванням газу з балонів, або з використанням пристроїв відбору димових газів при процесі горіння та перерозподілу їх в повітряне середовище теплиці.

При відсутності можливості штучного збагачення рівня CO_2 організується процес вентиляції для забезпечення притоку CO_2 із зовнішнього середовища.

1.2.4 Освітлення грає одну із провідних ролей для життя рослин.

По перше світло забезпечує рослини енергією без якої неможливо їх харчування. Але не тільки інтенсивність світлового потоку має величезне вплив на життєдіяльність рослин.

1.2.4.1 Рослини також дуже чутливі та до тривалості освітлення. Вони використовують сонячне світло як джерело інформації. Так, співвідношення тривалості нічного та денного періоду для більшості рослинних організмів є орієнтиром у етапах їх розвитку (початок вегетації, цвітіння, періоду спокою тощо). Така реакція рослин на довжину дня та ночі, відома як фотоперіодизм, дозволяє культурам обрати оптимальний час для здійснення кожної фази своєї життєдіяльності.

Залежно від цієї реакції розрізняють рослини довгого дня, для яких потрібно світловий період не менше 12 - 18 годин на добу та рослини короткого дня, які задовольняються сонячним світлом протягом 8 – 12 годин. З допомогою скорочення або подовження освітлювального періоду можна регулювати початок та тривалість фаз життєдіяльності (вегетацію, цвітіння, плодоношення) рослин. У культур, що входять до групи рослин короткого дня, скорочення освітлювального періоду викликає прискорення переходу від вегетативної стадії розвитку до репродуктивної. Зворотна реакція спостерігається у рослин довгого дня: більш тривалий освітлювальний період стимулює ранній вступ у фазу цвітіння.

1.2.4.2 Крім фотоперіодизму рослини виявляють ще декілька основних реакцій на освітлення:

Фотосинтез – це процес, у якому з вуглекислого газу, води та світла утворюються органічні речовини. Променева енергія сонця перетворюється на хімічну енергію.

Фототропізм - це рух рослини в напрямку світла. Усі ми бачили кімнатні рослини, що нахиляються до вікна. Виробляються гормони росту, які змушують стовбурові клітини із сонячного боку зростати активніше, викликаючи нахил стовбура.

1.2.4.3 За вимогливістю до освітлення рослини діляться на

- світлолюбні (Геліофіти)

- тіньовитривалі (Сціогеліофіти)
- тіньолюбні (Сціофіти).

До першої групи відносяться культури, які добре ростуть і розвиваються під дією прямих сонячних променів або яскравого розсіяного світла, а на зменшення тривалості та інтенсивності освітлення реагують негативно. Зазвичай це рослини південних регіонів, де сонячна активність дозволяє їм отримувати щонайменше до 10 – 12 тисяч люксів. До цієї категорії входять більшість городніх культур і плодоносних дерев, цитрусові, пальми, сукуленти, бугенвілія, жасмин, гібіскус, гарденія, пасифлора, троянди та інщі.

Частина рослин, пристосованих як до розсіяного освітлення, і до періодичного або часткового затінення, утворюють групу тіневитривалих культур. Їхня потреба у світлі знаходиться в діапазоні від 5 - 10 тис. люксів. До тіньовитривалих відносяться плодово-ягідні чагарники, а також фенхель, хрін, естрагон, м'ята перцева, розмарин, базилік та ін.

Тіньолюбні рослини – це невибагливі до освітлення рослини, які віддають перевагу затіненим ділянки та болісно реагують на прямі сонячні промені. Необхідна кількість сонячного світла для них обмежується 2,5 тис. - 4 тис. люксів на рік. До представників тіньолюбних культур зараховують лимонник, актинідію коломікту, деякі сорти суниці садової, салату, м'яти, конвалія, барвінок та ін.

1.2.4.4 Недостатність чи відсутність освітлення дуже згубно позначаються на розвитку культур через деактивацію процесу фотосинтезу та, як наслідок, обмеженого синтезу органічних речовин. В результаті рослини виростають слабкими, і у них спостерігаються різні дефекти зростання та розвитку: витягнутість пагонів і міжвузлів, бліде забарвлення зеленої маси, зменшення розмірів листя, мізерність кольороутворення або повне відсутність цвітіння, пожовтіння та опадання нижнього листя тощо.

Хронічний дефіцит сонячної енергії призводить до загибелі рослин. Культури можуть відчувати нестачу світла при короткій тривалості світлового дня, а також при недостатній інтенсивності освітлення.

З іншого боку надлишок світла також призводить до негативних факторів: вигоряння хлорофілу (верхівки рослин біліють), світловим опікам, сповільненому зростанню.

1.3 Важливість спектральної складової при освітленні культиваційних споруд закритого ґрунту.

Вкрай важливим фактором розвитку рослин є склад спектра опромінення. Людини та рослини «бачать» по різному. Людське око найбільш чутливо в діапазоні 500 – 600 нм світлового спектру. Для рослин важливий набагато більший діапазон, саме цей діапазон формує фотосинтетичною активну радіацію.

1.3.1 PAR (Photosynthetic ActiveRadiation, англ.) або фотосинтетична активна радіація це світло в діапазоні від 400 до 700 нанометрів(нм) який використовується для фотосинтезу рослинами. PAR визначає тип світлових хвиль, що беруть участь у фотосинтезі. Саме кількість та спектральна якість світла в діапазоні PAR є однією з найважливіших характеристик для освітлення рослин(Рис. 1.3.1).

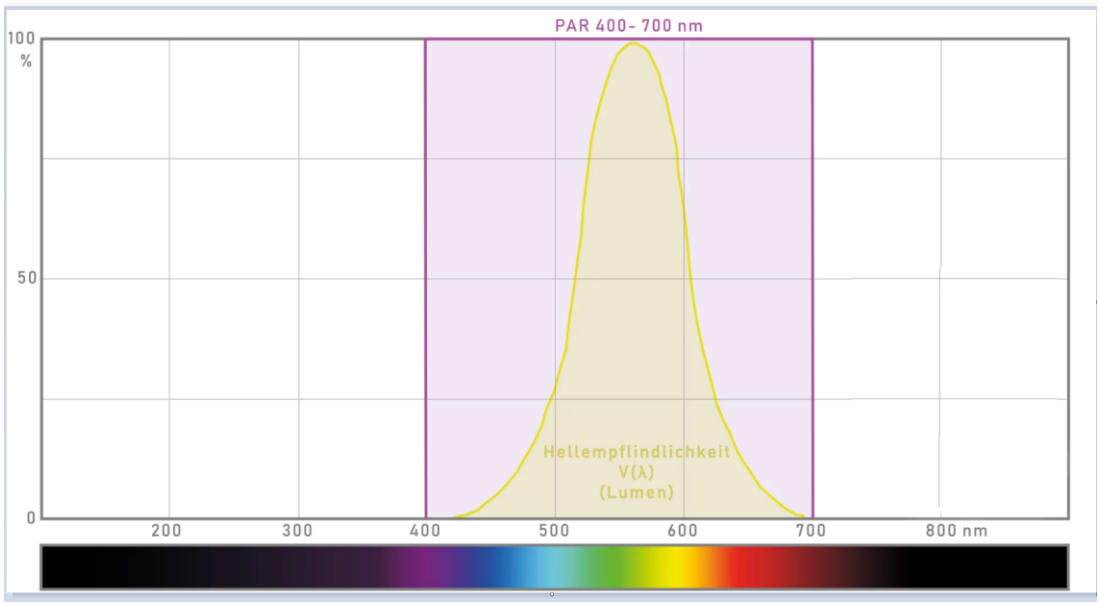


Рис. 1.3.1 - PAR та видима частина світла

1.3.2 Спектр світлової електромагнітної радіації по різному впливає на рослини:

- ультрафіолетове випромінювання зміцнює рослини, підвищує вміст вітамінів;
- фіолетові та сині промені активізують фотосинтез, прискорюють ріст рослин;
- Менш потрібне(на відміну від людського ока) але все ще важливе висвітлення теплиці зелено-жовтим світлом;
- світло помаранчевого та червоного спектру покращує розвиток кореневої системи, цвітіння культур;

В залежності від стадії розвитку рослинних культур їм потрібен світ з різною довжиною хвилі. Кожній фазі зростання відповідають свої потреби у спектральному складі світлових променів. Наприклад, для розвитку зеленої маси необхідне блакитне світло, а для зростання кореневої системи та в період підготовки до цвітіння в спектрі мають бути відтінки жовтого та червоного. Зелені промені стимулюють процеси фотосинтезу в листі з щільною структурою. Найбільш важливим у початковій стадії зростання культур є достатня кількість променів синього та фіолетового спектру з

довжиною хвиль 380 – 490 нм. Таке освітлення стимулює освіту білків та активізує вегетацію, внаслідок чого прискорюються ростові процеси культур. Це короткі хвилі з великим запасом енергії, які справляють великий вплив на зелені частини рослин. Від синього спектра залежать такі процеси: пророщування насіння, зростання пагонів, розвиток сім'ядолей і листових пластин, розвиток м'ясистого листя, попередження витягування побігів, які можуть стати тонкими стеблями, більш раннє формування бутонів і початок цвітіння.

Світлові хвилі довжиною 595 – 620 нм та 620 – 720 нм (помаранчевого та червоного спектру) критично важливі для здійснення фотосинтезу, тому найбільшу потребу в них рослини відчувають у період цвітіння та плодоношення. Під його дією проходять такі процеси, настає період цвітіння, подовжується термін цвітіння, починається формування плодів, розвиток насіння, покращуються смакові якості плодів.

Найменшу роль у життєдіяльності культур відіграють промені жовтого (довжина хвиль 490 – 565 нм) та зеленого (довжина хвиль 565 – 600 нм) спектра, але теж певною мірою потрібен рослинам.

Ультрафіолетові промені з довжиною хвиль 315-380 нм у великих дозах дуже шкідливі, але в обмежених кількостях захищають рослинний організм від патогенної мікрофлори, що підвищують холодостійкість рослин, перешкоджають їх «витягування».

За допомогою правильно підібраного спектрального освітлення можна стимулювати збільшення врожайності культур на 30%.

1.4 Освітлення в теплицях.

В промислових теплицях, де круглий рік вирощують квіти, тропічні фрукти та овочі, використовують штучне світло, щоб підтримувати

фотосинтез і відповідний температурний режим. При цьому важливо відтворювати не сам сонячний світ, а умови, в яких рослинним культурам буде комфортно. Важливо розуміти скільки світла потрібно рослини, бо рослини у стані стресу можуть плодоносити навіть швидше звичайного, але такий урожай значно поступається за смаковими якостями плодам, вирощеним у природних умовах.

Щоб компенсувати недостаню кількість сонячного світла, а подекуди його повну відсутність, для освітлення рослин використовують спеціалізоване освітлювальне обладнання.

1.4.1 Звичайні лампи розжарювання не підходять для застосування в теплицях, тому що їх світловий спектр є несприятливим для рослин і виділяють велику кількість тепла при високому рівні енергоспоживання.

Для штучного освітлення теплиць підходять кілька видів ламп: люмінесцентні, натрієві, галогенні, світлодіодні.

1.4.2 Люмінесцентні лампи, також відомі як лампи денного світла, можуть підходити для більшості теплиць. Їхній спектр близький до природного сонячного. Бувають люмінесцентні лампи холодного білого світла – ці лампи випромінюють більше світла у фіолетовому спектрі, викликаючи розвиток листя у рослин. Інший тип люмінесцентних ламп – лампи теплого білого світла, які дають більше світла в інфрачервоному спектрі, що прискорює процес цвітіння. Найбільш вигідний тип люмінесцентних ламп - це лампи природного світла, що містять весь сонячний спектр, проте вони дещо дорожчі. Недоліком всіх люмінесцентних ламп є низька освітленість, низький світловий потік, якого може бракувати деяким рослинам.

1.4.3 Натрієві лампи відносяться до газорозрядних ламп високого тиску. Їх світло має дуже високу інтенсивність, вони дають світловий потік, достатній навіть екзотичним рослинам. Натрієві лампи мають більше

червоного світла, тому вони сприяють прискоренню цвітіння. До того ж, натрієві лампи випромінюють достатньо тепла, що дозволить заощадити на опаленні теплиці.

1.4.4 Галогенові лампи також відносяться до газорозрядних ламп. Вони, також як натрієві, мають підвищену інтенсивність світлового потоку, але, на відміну від натрієвих, мають насичений фіолетовий спектр, тому галогенові лампи добре доповнюють натрієві в одній теплиці.

1.4.5 По мірі здешевлення, світлодіодні лампи набирають великої популярності як в побуті так і для освітлення теплиць. Для освітлення в побуті використовують світлодіодні лампочки з білим або жовтуватим світом, а для освітлення теплиць є спеціальні фітолампи. На ринку доступні декілька видів фітоламп:

- світлодіодні фітолампи одного кольору сперту: сині, червоні, зелені, ультрафіолетові та інфрачервоні;
- фітолампи, які поєднують в собі блакитний і червоний світ(бі-колор),
- мультиспектральні фітолампи - випромінювання синього, червоного, а також білого спектру;
- фітолампи повного спектру найточніше копіюють сонячне світло.

Світлодіодне освітлення рослин має такі переваги:

- висока енергоефективність;
- можливість використання в одному ліхтарі декількох діодів з різними спектрами випромінювання для формування необхідного спектрального складу;
- можливість керування потужністю аж до імпульсних режимів;
- можливість формування направленого випромінення;
- тривалий термін експлуатації ;
- малі габарити;

- можливість організації відводу тепла для його перерозподілу в межах теплиці.

1.5 Необхідність об'єктивного контролю освітлення та його автоматичним керуванням в теплицях.

Ефективне освітлення у теплицях залежить від багатьох факторів:

1.5.1 Алгоритму освітлення рослинної культури. Науковці розробляють світові рецепти для кожної культури та навіть сорту у межах однієї культури. Вони є частиною технологічних карт вирощуваних культур. Велика частина цієї роботи – підбір оптимального спектру опромінення для конкретної культури та сорту. Але лише цього не достатньо, в наш час ведуться багато досліджень в різних країнах світу стосовно розробки динамічних світлових режимів у розрізі зміни вимог до освітлення впродовж світлового дня та в розрізі потреб в сонячної радіації впродовж всього життєвого циклу рослини.

1.5.2 Джерел світлового опромінення. Якість освітлення залежить не лише від номінальних технічних характеристик ліхтаря. Залежно від виду джерела, виробника, якості, партії, строку використання, технічного обслуговування та способу встановлення інтенсивність світлового опромінення яке дійсно отримують рослини буде сильно різнитись. Коливання температури впливає на зміщення спектру світіння світлодіодних ліхтарів.

1.5.3 Взаємозв’язку з іншими параметрами мікроклімату. Кожен нормативний показник освітлення змінюється залежно від рівня CO_2 та рівня температури. Бо при низькій температурі та дефіциті вуглекислого газу падає ефективність фотосинтезу, тому зменшуються потреби в освітленні. До того, теплиця має розосереджені показники мікроклімату. В різних частинах теплиці температура, вологість, состав повітря може суттєво різнитись.

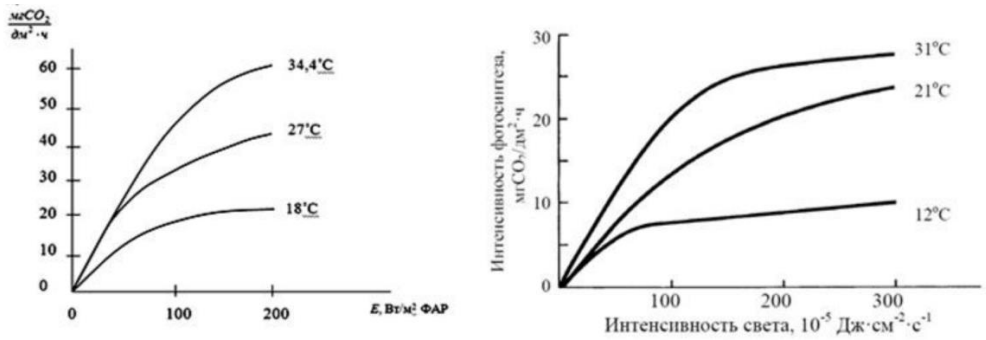


Рис. 1.5.3.1 – вплив температури на ефективність фотосинтезу та потреби в світловій енергії.

1.5.4 Природного освітлення теплиці, яке дуже залежить від пори року, часу в добі, орієнтації теплиці та матеріалу з якого вона зроблена, погодних умов.

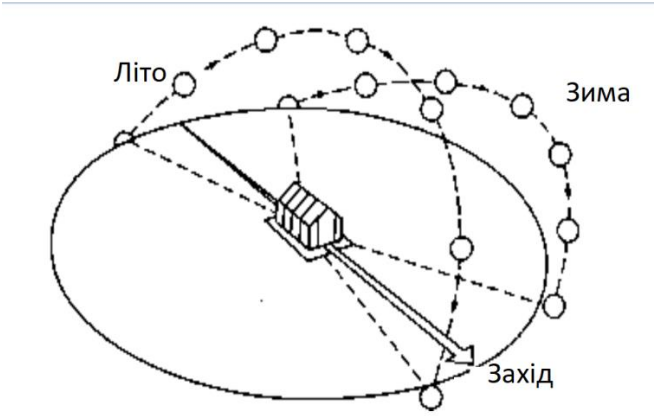


Рис. 1.5.4.1 Шлях сонця влітку та взимку та орієнтація теплиці

Точність вимірювання є ключовою та дуже складною задачею, яка впливає на врожай культури. Необхідно проконтролювати дотримання динамічного світового режиму технологічних карти сорту та виду вирощуваних культури з урахуванням якості та стана ліхтарів, динамічної зміни природного освітлення, та інших показників мікроклімату в теплиці. Стає очевидним що неможливо провести надійний розрахунок інтенсивності опромінення світовою радіацією лише на основі опису технічних характеристик ліхтаря та усереднених показників мікроклімату в теплиці.

Ефективний моніторинг освітлення допоможе забезпечити високій рівень якості врожаю шляхом опромінення рослин рівно необхідним рівнем

електромагнітної сонячної енергії та досягти суттєвої економії електричної енергії. Розглянемо методики вимірювання освітлення.

1.6 Теплиця - об’єкт керування з розподіленими параметрами мікроклімату.

Для отримання в ній гарного врожаю необхідно контролювати чимало параметрів мікроклімату. Всі вони дуже важливі бо найменше відхилення призведе до скорочення чи втраті врожаю. Але це ще тільки половина проблеми. Коли ми говоримо про контроль параметрів мікроклімату в теплиці, то в голові виникає картина (Рис. 1.6.1) коли контролюється кожен параметр мікроклімату для кожної рослини.

Рис. 1.6.1 - Точковий контроль параметрів мікроклімату в теплиці

Насправді, в реальному житті все ще працює дещо по іншому(Рис. 1.6.2)

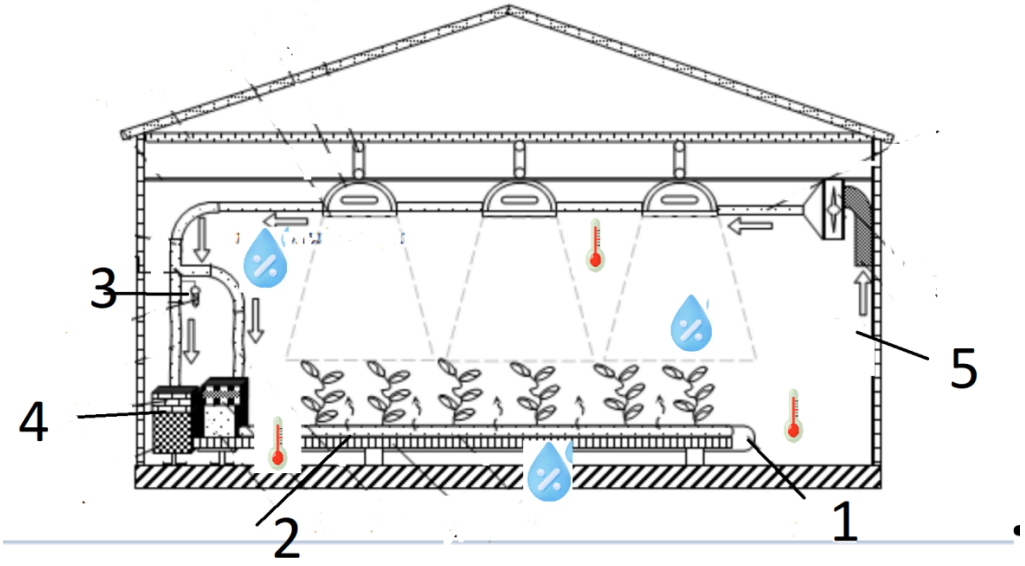


Рис. 1.6.2 – Контроль в умовах розподілених параметрів мікроклімату в теплиці

Рослини в зоні 1 та 2 скоріше за все мають різний температурний режим та мають доступ до різного по составу повітря. Біля рослин зони 2 знаходиться котел системи обігріву 4, там же датчик температури 3. З іншого боку рослини в зоні 1 знаходяться ближчий до витяжної труби системи вентиляції. Зовсім немає механізмів контроля освітлення. В кращому випадку на підприємстві є переносний люксметр або навіть спектрометр, які використовуються час від часу для приблизного розуміння якості освітлення в теплиці.

Для якісного контролю параметрів мікроклімату потребується розосереджена система датчиків яка включає:

1.6.1 Датчики температури. Потрібно два типу датчиків. Стельові інфрачервоні датчики, які зможуть заміряти температуру поверхні листя рослин. Інші теплові датчики потрібно розмістити, близько до поверхні грядок в безпосередній близькості до рослин (під листям).

1.6.2 Датчики вологості ґрунту, требо розмістити дуже близько до Корнієвої системи, особливо для тих теплиць, де працює крапельне зрошення безпосередньо під коріння рослин.

1.6.3 Датчики вологості повітря, розміщуються в безпосередній близькості до листви рослин.

1.6.4 За такою ж схемою, біля листви рослин розміщуються датчики CO_2 задля контролю составу повітря.

1.6.5 Датчики опромінення поверхні рослин фізіологічно активною радіацією. Датчики потрібно розмістити біля поверхні листя рослин.

Тільки маючи подібну систему датчиків можливо організувати ефективний контроль параметрів мікроклімату в теплиці, яка являється об’єктом з розподіленими параметрами мікроклімату. Виходячи з задач виникає два питання, які треба вирішити для досягнення поставленої мети.

По перше треба розробити архітектуру мережі датчиків. По друге є певна проблема з вимірюванням опроміненням поверхні рослин фізіологічно активною радіацією. Донедавна на ринку були присутні тільки дорогі спектрометри, зробити з яких систему датчиків було нерентабельно. Відносно дешеві люксметри не можуть забезпечити якісне виконання цієї задачі.

1.7 Постановка задач магістерської роботи.

Ця магістерська робота присвячена спробі вирішити питання сформульовані в попередньому пункті та ставить наступні задачі:

1.7.1 Розробка архітектури та програмного забезпечення бездротової мережі контролю параметрів мікроклімату у культиваційних спорудах закритого ґрунту та інтегрувати її в загальну систему автоматичного керування виробничими процесами на підприємстві.

1.7.2 Розробка методики вимірювання та програмного забезпечення спектрометра для контролю потужності спектральних складових фізіологічно активної радіації

Для початку треба розглянути існуючи на сьогодні підходи побудови бездротових мереж, методів інтеграції різноманітних систем автоматичного керування виробничими процесами на підприємстві та методів вимірювання освітлення та освітлення рослин.

Освітлення в нашому контексті є досить неточним терміном, так, як освітлення перш за все має на увазі чутливість людського ока, а для рослин важливий дещо ширший спектр. То правильний термін буде опромінення фізіологічно активною радіацією. Але в роботі будемо використовувати ці два терміни як взаємозамінні. Коли мова буде стосуватися людини чи техніки, то найчастіше буде використовуватися термін освітлення. Коли тема буде більш відносна до рослин, то буде використовуватись термін опромінення.

Почнемо з розгляду методів вимірювання освітлення.

1.8 Одиниці вимірювання освітлення та фізіологічно активної радіації.

Використовують різні величини, щоб показати ефективність або інші характеристики освітлення: "Ват", "Люмен", "Люкс", "PPF", "PPFD. Незважаючи на те, що всі ці терміни в тій чи іншій мірі виражають ефективність або кількість світла, що випромінюється сонцем, лампою, світлодіодом або іншим джерелом, лише частина з них дійсно важлива для оцінки освітлювання рослин.

1.8.1 Вимірювання освітленості в ватах поступово йде в минуле, коли основним джерелом штучного освітлення була лампа розжарювання. Тоді знаючи скільки ватт потребляє лампа можна було досить точно сказати скільки світла вона даватиме. Для лампи розжарювання кількість світла, яке вона виробляє, залежить від матеріалу нитi розжарювання та її температури. Але ці параметри в різних лампах відрізняються мало чим, тому лампи однієї потужності світили приблизно однаково.

1.8.2 Основним параметром будь-якого джерела світла є кількість світла, яке воно виробляє в одиницю часу. Ця величина називається

світловим потоком. Люмен(Lumen, Лм) це базова одиниця виміру світлового потоку, створена на основі чутливості людського ока. Світловий потік є загальною характеристикою, більш наглядною характеристикою є освітленість.

1.8.3 Освітленість - це інтенсивність світла, вилична, що відображає кількість світла, що потрапляє на одиницю площі. У міжнародній системі СІ одиницею вимірювання освітленості служить люкс (лк). Один люкс дорівнює одному люмену розподіленому на 1 метр квадратний.

1.8.4 PPF(Photosynthetic Photon Flux, англ.) – фотосинтетичний потік фотонів. Це абсолютна кількість фотонів у PAR діапазоні(400-700нм), що випромінює джерело світла кожну секунду. Для того щоб провести подібний вимір, потрібно використовувати так звану інтегральну сферу, яка реєструє і вимірює всі фотони, що випускаються лампою. Одиниця виміру – мікромолі за секунду (µmol/s).

1.8.5 PPFD(Photosynthetic Photon Flux Density, англ.) це щільність фотосинтетичного потоку фотонів. Показує кількість світла в PAR діапазоні (400-700нм), яке потрапляє безпосередньо на рослину. PPFD вимірюється в мікромолях на метр квадратний за секунду (µmol/m2/s).

Виходячи з опису мір стає зрозумілим, що PPFD це саме той вимір, який варто використовувати для оцінки інтенсивності опромінення рослин фізіологічно активно активною радіацією, до якого повинна привести методика яка розробляється в цій роботі.

Розглянемо способи вимірювання та методики переходу від одної міри до другої.

1.9 Способи вимірювання освітлення та фізіологічно активної радіації

Кількість Люменів та міру PPF коли річ йде про штучні промислові джерела світла можливо дізнатися з опису технічних характеристик приладу.

З певними умовами або припущеннями виробники декларують в документації на їхні прилади навіть освітленість чи PPFD. Воно може дати приблизний орієнтир на те що будуть отримувати рослини під тим чи іншим джерелом освітлення. Але ці значення PPFD указують виключно в центральній точці під лампою, при цьому часто не вказуючи висоту з якої вимір було зроблено. Такі дані дають дуже мало корисної інформації і найчастіше є сильно завищеними від реальної картини загалом. До того ж світло від приладу може підвергатися дії інших джерел. Для того щоб оцінити справжню ефективність лампи, необхідно робити виміри в кількох точках, при цьому вказуючи з якої висоти і в якій області на площі було здійснено вимір (тобто створити так звану PAR-карту far-map).

1.9.1 Для проведення цих замірів знадобиться спеціальний прилад– спектрофотометр. (Рис 1.9.1).

Рис. 1.9.1.1 Спектрометр.

На Рис. 1.9.1.2 наведені результати вимірів спектрів різних джерел світла. Фіолетовий графік – лампа розжарювання, зелений- ртутна економна лампа, сірий та синій -різні LED/

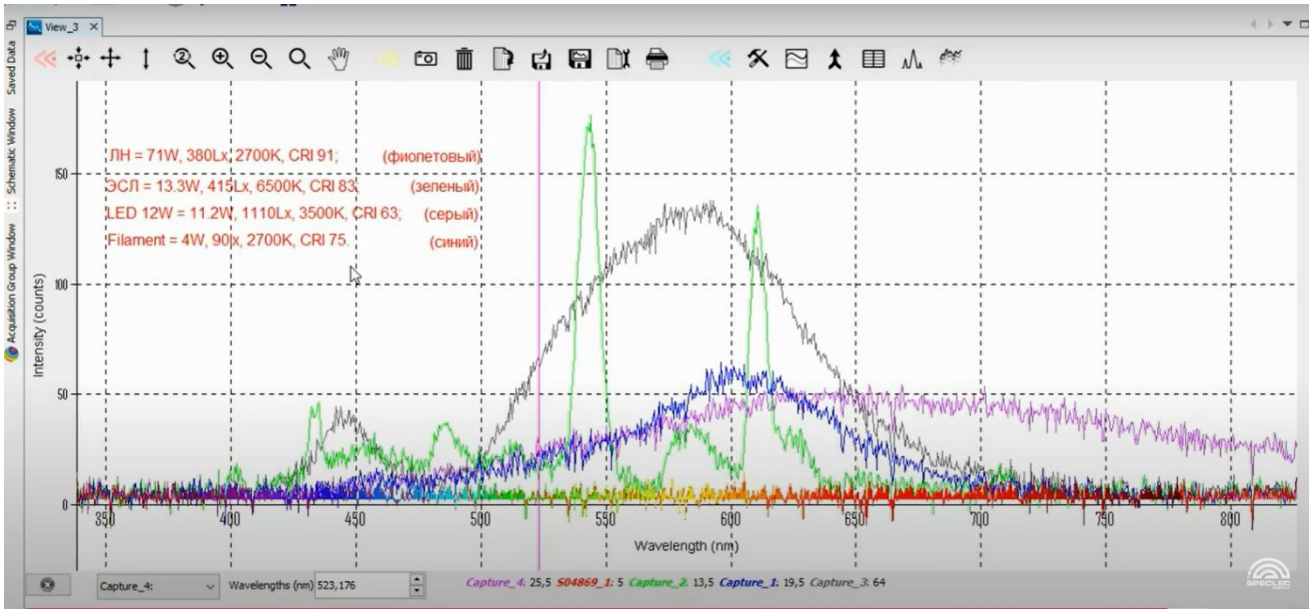


Рис. 1.9.1.2 Замір спектрів різних джерел освітлення

Ці пристрої дають точний результат та вже зарекомендували себе в сільському господарстві, але в них є великий недолік – ціна, кожен апарат коштує від 800 USD. Тому в сільському господарстві вони використовуються як переносні пристрої для тимчасової перевірки даних. Проектування системи моніторингу теплиці з розподіленою системою датчиків з використанням подібних пристроїв є недоцільною.

1.9.2 Використання люксів або люменів для вимірювань освітленості рослин дає результати, які не будуть відображати дійсної картини, але розглянемо цей спосіб для прикладу. Для початку треба провести замір освітленості.Її вимірюють спеціальними приладами – люксметрами

Рис. 1.9.2.1 – Люксметр.

Безумовно, маючи дані про люкси та враховуючи спектрограму джерела світла, можна зробити оцінку ефективності освітлення перерахувавши кількість люксів в приблизну міру інтенсивності опромінення в діапазоні FAR. Робиться це на основі розрахунку коефіцієнтів по спектрограми джерела світла. Для деяких відомих джерел можливо скористатися таблицею вже розрахованих коефіцієнтів. У таблиці нижче наведені коефіцієнти переведення з Ваттів у фотони спектру абсолютно чорного тіла, усіченого до діапазону 400-700 нм. У ній також наведені одиниці вимірювання світлової віддачі для кожного з джерел світла, а також частини спектру абсолютно чорного тіла, яка відповідає ФАР.

Таблиця 1.9.2.1 - Конвертування фотометричних в фотосинтетичні величини для різних джерел світла

Джерело випромінювання	з PPFD (мкмоль/м²/с) в PAR(Вт/м²)	з Люксів в PPFD (мкмоль/м²/с)	з Люксів в PAR(Вт/м²).
Сонце	0,219	0,019	$4,02*10^{-3}$
Холодна біла флуоресцентна	0,218	0,014	$2,93*10^{-3}$
ДНаТ	0,201	0,012	$2,45*10^{-3}$
Натрієва низького тиску	0,203	0,009	$1,92*10^{-3}$

Основна проблема при використанні люксів у вимірі інтенсивності світла в системах освітлення для садівництва полягає в недостатньому вимірювані люксметрами синього (400 – 500 нм) та червоного (600 – 700 нм) світла у видимому спектрі по причині низької чутливості проборів в цих діапазонах (Рис 1.9.2.2).

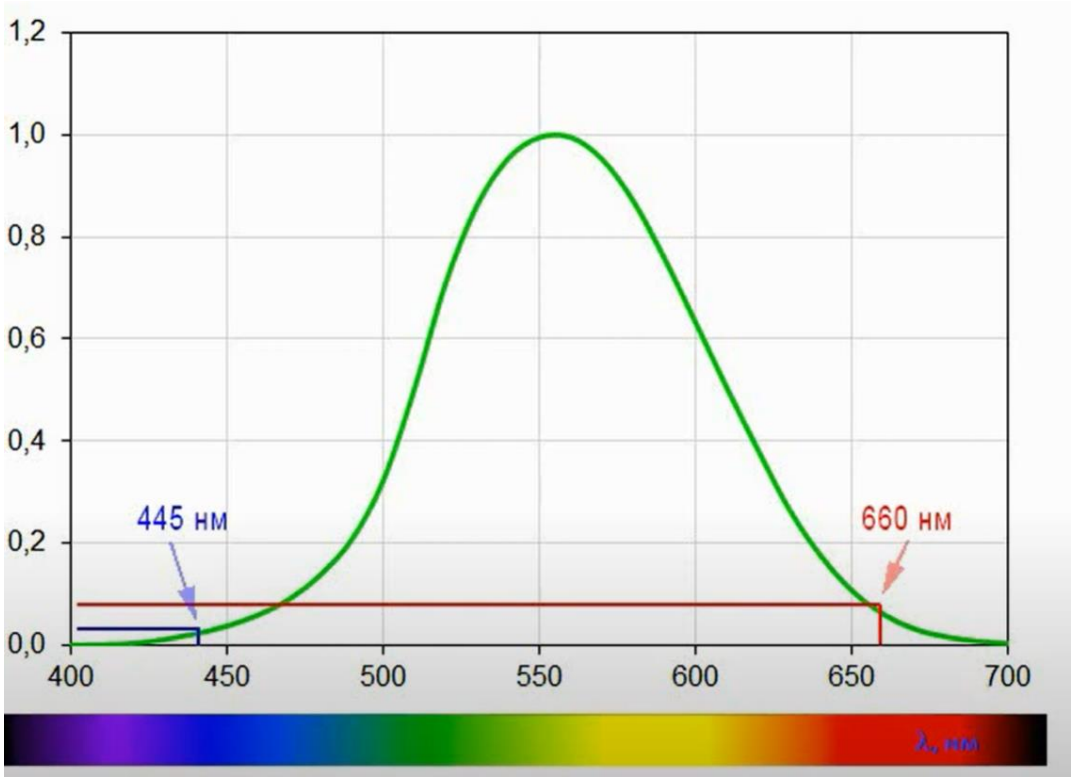


Рис 1.9.2.2 – Чутливість люксметру.

Так що ж робити якщо одні прилади вимірювання опромінення фізіологічно активною радіацією є дуже дорогими тому їх використання в промислових масштабах є необґрунтованим. Інші підходять по ціні, але вимірюють дещо інші величини, які дуже приблизно можна перерахувати для наших потреб. На щастя на ринку з’явилась сучасні недисперсійні спектрометри, які мають невелику ціну та дозволяють получити результати кращі ніж люксметри. Більш детальний опис буде надано у третьому розділі роботи.

1.10 Способи автоматичного керування виробничими процесами в культиваційних спорудах закритого ґрунту, концептуальний підхід «розумна теплиця».

Перейдемо до наступного питання роботи та розглянемо існуючі варіанти вирішення проблеми контролю параметрів мікроклімату в теплиці. Ця задача вирішується створенням системи розподілених датчиків, коли вони розміщаються в безпосередній близькості до рослин. Вона є підсистемою

концепту «розумна теплиця». Розумна теплиця (Рис. 1.10.1) — це теплиця, яка включає в себе мікроконтролери промислового інтернету речей IIoT(Industrial Internet of things, анг), сенсори та спеціальний додаток. IIoT і реалізує ту саму розподілену мережу датчиків. Стає питання чому тоді ця задача є предметом цієї роботи? До останнього часу ці технології були досить дорогими. Зараз з здешевленням ці підходи набирають оберти та все ще перебувають в стадії активного розвитку. Так бездротова передача даних дозволяє суттєво знизити витрати на дротові з'єднання, які вимагали не одну тисячу метрів кабелю. Тому є певний інтерес в проведенні науково-прикладних досліджень. Розглянемо цю технологію більш детально.

1.10.1 Комп'ютери, що керують підтримкою заданих параметрів мікроклімату та поливом, у теплицях застосовуються вже понад 50 років.

Рис. 1.10.1 Розумна теплиця

До 2022 року глобальний ринок IIoT для теплиць досяг $1,3 млрд. У 2016 році цей показник становив $680 млн. Раніше аграрій потребував більшої кількості робочих, які б контролювали потрібні умови в теплицях.

Тепер же більшість завдань можна вирішити за допомогою інтелектуальних датчиків і смартфона.

1.10.2 Як це працює? Мікродатчики можуть контролювати величезний масив факторів: освітлення, температуру, якість ґрунту, кількість пестицидів.

Датчики також фіксують дані про зростання рослин, зрошення, використання добрив. Зібрані дані відправляються на локальний або хмарний сервер. Консоль веб-адміністратора дозволяє аграрію налаштувати параметри системи та синхронізувати її з іншими рішеннями. Додаток на мобільному пристрої збирає звіти про стан культури та умови в теплиці, генерує попередження, якщо який-то показник не відповідає нормі.

Мікроконтролер— це основа будь-якої підключеної системи тепла. В залежності від числової потужності мікроконтролер може використовувати прошивку або класичну операційну систему для обробки даних датчиків і передачі їх на сервер. Якщо користувачеві не потрібні камери і складні сенсори — тоді можна вибрати простий контролер, який підтримує кілька типів датчиків, використовує радіо з'єднання, споживає мало енергії і може бути встановлений на ґрунті або прикріплений безпосередньо до рослин.

Можливо використання недорогих міні-комп’ютерів, таких як BeagleBoard, Arduino або Raspberry PI. Вони використовують вбудоване програмне забезпечення C++ для обробки даних датчиків і надають певні можливості для машинного навчання.

Мікроконтролери підключаються до електричної мережі, але потребують дуже мало електрики — 150 мА при активній передачі даних і всього 5 мА при сплячому режимі.

1.10.3 IIoT — це майбутнє сільського господарства. Інтернет промислових речей здатний збільшити виробництво продуктів харчування на 70%. Тому на рішення для агросфери вже приходить 6% всіх проектів інтернет-речей. Тільки за останні чотири роки було продано приблизно 600

мільйонів датчиків для сільського господарства. І значна частина їх припадає на розумні теплиці. IIoT обговорювалася ще в 1982 році, коли модифікований торговий автомат Coca-Cola в Університеті Карнегі-Меллона став першим пристроєм, підключеним до ARPANET (предок сучасного інтернету), здатним повідомляти про свої запаси та про те, коли були завантажені напої та холодні вони чи ні.

Концепція «Інтернету речей» IoTі сам термін вперше з'явилися у промові ПітераЛьюїса, опублікованої у вересні 1985 року. За словами Льюїса, «Інтернет речей - це інтеграція людей, процесів і технологій із підключеними пристроями та датчиками для забезпечення дистанційного моніторингу, стану, маніпулювання та оцінки тенденцій таких пристроїв».

Термін «Інтернет речей» незалежно та остаточно вивів у 1999 році Кевін Ештон з Procter&Gamble, пізніше Auto-ID Center Массачусетського технологічного інституту.

З того часу інтернет речей розвиваються навіть швидше ніж «людський» сегмент інтернету, та має дуже оптимістичні прогнози подальшого росту.

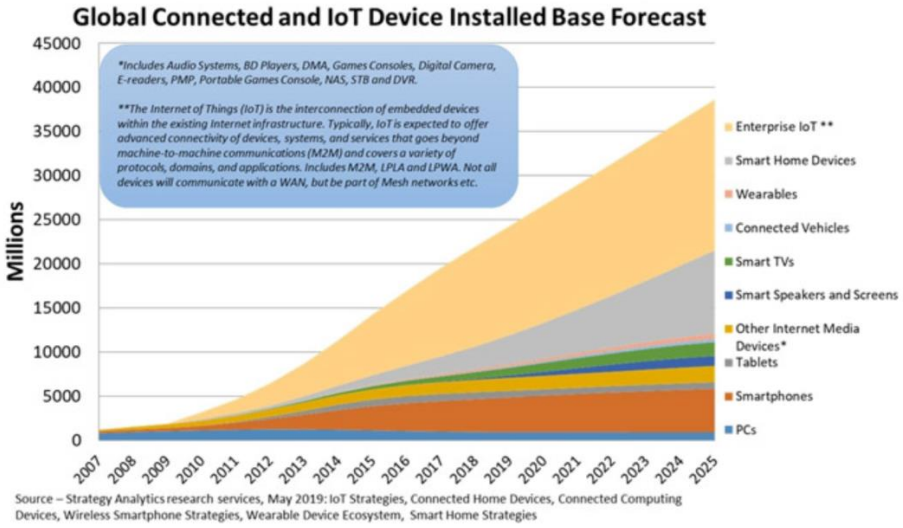


Рис. 1.10.3.1 Розвиток IoT [5]

Так, в 2021 году найбільша мережа світу(інтернет) вже налічувала 28 мільярдів пристроїв. Лише 13 мільярдів з них це пристрої безпосередньо кінцевого користувача, такі як смартфони, планшети, ноутбуки, персональні комп'ютери. 15 мільярдів - це автономні користувацькі або промислові пристрої широкого спектру призначення: логістика та транспорт, сільське господарство, промисловість, розумні дома, розумні міста, енергетичні мережі, тощо.

1.11 Способи інтеграція систем автоматизованого керування виробничими процесами на підприємстві.

Останнім питанням яке підлягає в цій роботі є розробка алгоритмів і програмного забезпечення системи аналізу та підтримки прийняття рішень для підвищення енергоефективності процесу опромінення рослин фізіологічно активною радіацією. Після реалізації системи контроля параметрів мікроклімату в теплиці є логічним розвивати систему далі. Для організації цієї взаємодії існує дуже зручна архітектура OPC UA (Open Platform Communications Unified Architecture, анл. Відкрита система комунікації, універсальна архітектура)— це остання за часом випуску специфікація OPC, яка представляє собою новий рівень відкритості, доступності та безпеки. Головними перевагами специфікації можна назвати наступні:

- Крос-платформна сумісність, тобто незалежність від ОС. Так як OPC UA не заснована не на технології Microsoft COM, то компоненти підтримують мультіплатформену реалізацію;
- Масштабованість від вбудованих систем до маінфреймів;
- Можливість обміну усіма типами даних: реального часу, тривоги та повідомлення, історія, файли;
- Доступність даних у виробничому контексті, тобто відповідно до моделі активів;
- Внутрішня система безпеки.

Більше деталей стосовно цієї архітектури та реалізації спілкування між бездротовою мережею теплиці та загальною системою автоматичного керування виробничих процесів на підприємстві доступно в 4 розділі роботи.

РОЗРОБКА АРХІТЕКТУРИ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ БЕЗДРОТОВОЇ МЕРЕЖІ КОНТРОЛЮ ПАРАМЕТРІВ МІКРОКЛІМАТУ У КУЛЬТИВАЦІЙНИХ СПОРУДАХ ЗАКРИТОГО ҐРУНТУ НА БАЗІ МІКРОКОНТРОЛЕРІВ TEHAS INSTRUMENTS CC1350 SIMPLELINK™ ULTRA-LOW-POWER DUAL-BAND.

2.1.1. Проектування та сегментів топології мережі. Розглянемо задачі кожного сегменту проектованої мережі.

Перша ланка сама теплиця. Щоб якісно контролювати параметри мікроклімату в розосередженій системі, якою є теплиця, треба якнайближче розмістити датчики до рослин. Кількість датчиків та особливості їх розміщення залежить від конструктивних особливостей теплиці, нормативів стосовно процесу проведення цих замірів, особливостей вирощування тих чи інших рослин і навіть сортів. Найвірогідніше систему розміщення датчиків знадобиться перепланувати в ході роботи в залежності від змін потреб виробничого процесу. Тому монтаж кабельної системи не є доречним. Знадобляться тисячі метрів кабелю, що потребує значних капітальних фінансових ресурсів на кожну теплицю. Технічне обслуговування цієї павутини кабелів буде ускладним завданням та теж потребуватиме немалих фінансових затрат. Тому зразу визначаємося що це буде бездротова мережа датчиків.

Планується що всі датчики теплиці будуть з’єднані з центральним шлюзом (топологія зірка), що об’єднає мережу теплиці з загальною мережею підприємства а далі за необхідності з любою глобальною мережею. В середні теплиці можливо передбачити можливість з’єднання з кожним датчиком по схемі точка-точка. Це необхідно для роботи персоналу безпосередньо в теплиці. Звісно до кожного датчика можливо добратися через центральну точку (шлюз) але можливість з’єднання з кожним датчиком окремо напряму дозволяє розшити інструментальний набір та гнучкість в процесі діагностики

проблем, технічного обслуговування, налаштування, тощо. Друга ланка мережі – загальна мережа підприємства. Особливості імплементації фізичної системи не має значення для цієї роботи та залежить від потреб та особливостей організації корпоративної мережі підприємства. Це може бути як бездротове з'єднання (локального покриття чи широкого мобільного), чи дротовим (металеві дроти чи оптичні), це все не є принципіальним для даної роботи. Важливим є лише робота протоколів високого рівня стосовно інтеграції мережі теплиць в загальні систему автоматичного керування виробничим процесами на підприємстві та можливість доступу к мережі датчиків з поза меж теплиці.

Загальна схема проекту мережі представлена на рисунку 2.1.1

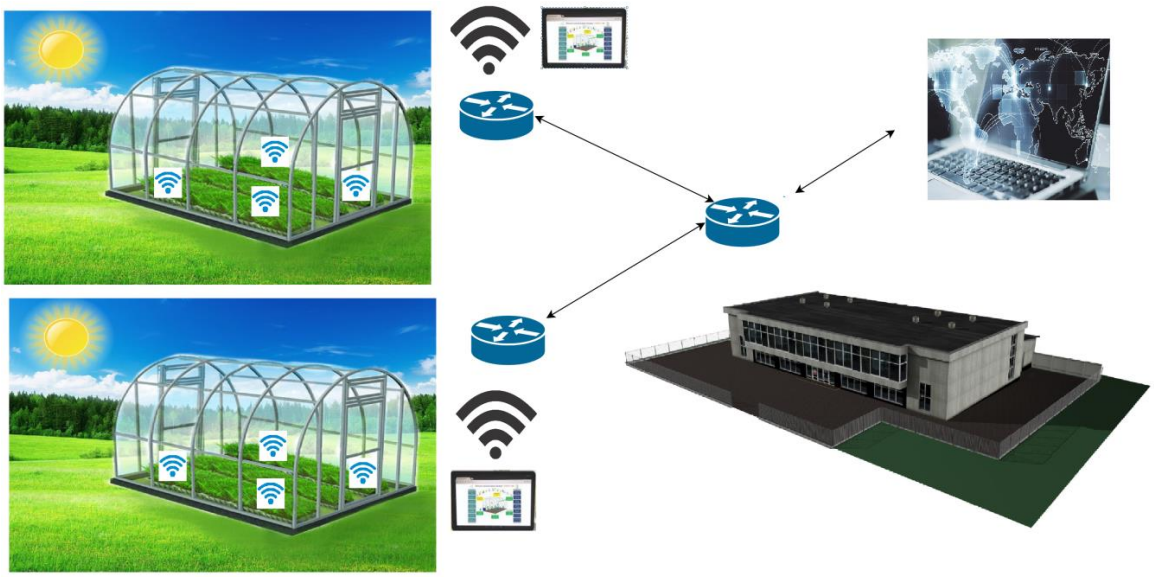


Рис. 2.1.1 – Загальна схема мережі

2.2 Визначення зі стеком використовуваних протоколів.

Вже згадувалось про фізичний рівень мережі та протоколи високих рівнів. Розглянемо ці поняття більш детально та продовжмо проектування нашої мережі. Взагалі, розробка кожної комп'ютерної мережі завжди базується на 7 рівнях базової відкритої мережевої моделі передачі данихOSI(The Open Systems Interconnection model, англ.).

Так інтернет працює на базі стеку протоколів TCP/IP (Transmission Control Protocol/Internet Protocol,англ. - протокол управління передачею/інтернет протокол).

TCP/IP– стек мережевих протоколів, який є реалізацією моделі OSI та у назві має перелік двох базових протоколів. Цей стек об'єднує деякі з 7 рівнів основної мережевої специфікації і в результаті описує 4 рівня (Рис.2.1.2.1)

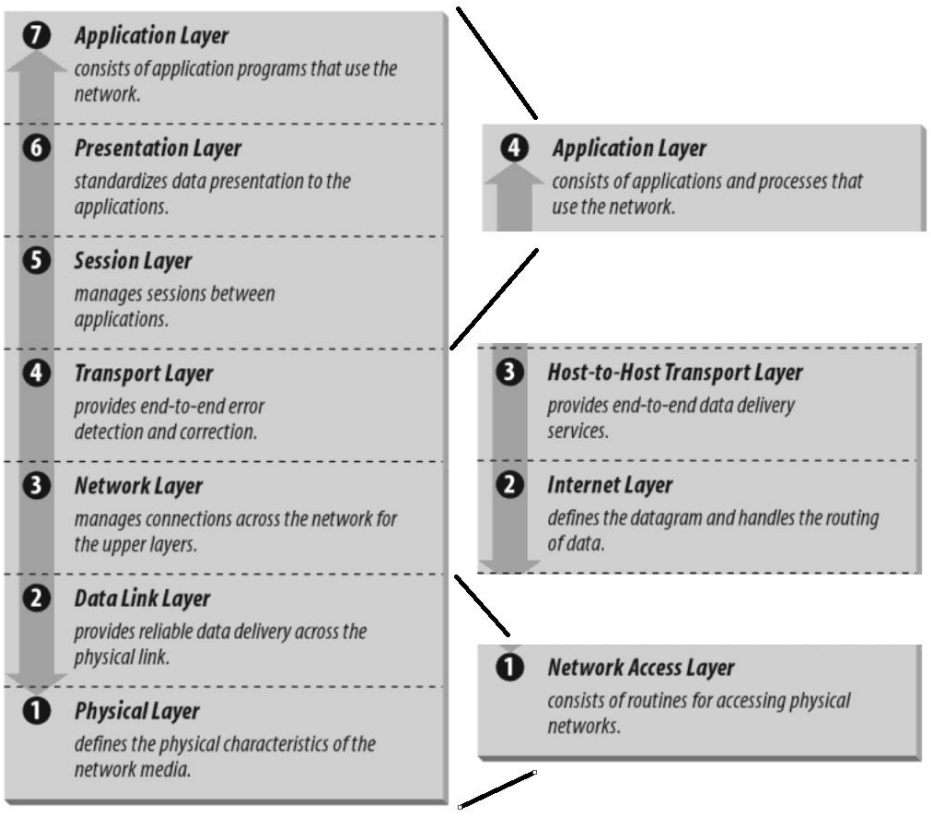


Рис. 2.2.1 - Реалізація стеку TCP/IP.[7], с 7-9

В подальшому будемо використовувати цю реалізацію як базову, тому розглянемо кожний рівень окремо та визначимся. Деталі реалізації стеку протоколів TCP/IP чи визначення принципів моделі OSI знаходяться поза межею цієї роботи та є загальнодоступним технічним матеріалом. Деякі деталі розкриваються в межах необхідності в подальшому в процесі проектування мережі.

Розглянемо ці 4 рівня в розрізі наших задач бездротової мережі. Дуже узагальнено можна сказати що задача мережі полягає в необхідності передати

зазначення якогось датчика до кінцевого користувача цих даних. На рисунку 2.2.2 можна побачити деталі цього процесу.

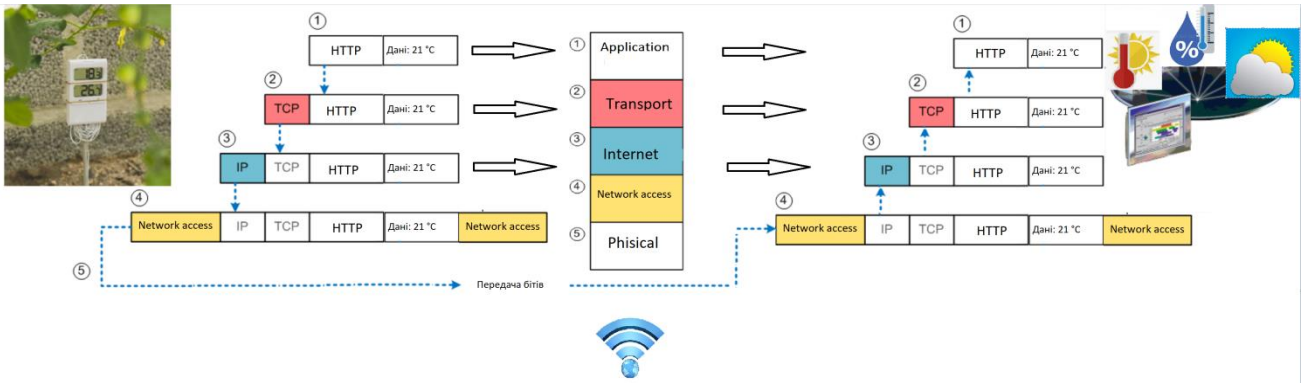


Рис. 2.2.2 - Узагальнена схема бездротової мережі теплиці

Наприклад треба передати дані температурного датчика.

2.2.1 Application рівень (англ.- Рівень додатків або прикладний рівень). Датчик виконує задачі серверу, поставляє дані для SCADA (Supervisory Control And Data Acquisition, англ.—диспетчерське управління та збір даних) системи (клієнт). На малюнку видна горизонтальна стрілка яка показує спілкування на рівні додатків між сервером та клієнтом. На цьому рівні зовсім немає значення як повідомлення знаходить свого адресата, як вони пересилаються по мережі, як забезпечується надійність передачі даних та безліч інших питань.Обмін повідомленнями між програмами працює з припущенням що адресат гарантовано «почує» повідомлення. На цьому етапі важливий лише інтерфейс спілкування – домовленість між клієнтом та сервером стосовно того що і в якій формі відправник перешле адресату, та як адресат відпрацює вхідне повідомлення

Існує безліч протоколів прикладного рівня HTTP, POP3, SMTP, FTP та багато інших. Дещо забігаючи попереду опишемо протоколи, які будуть використані в реалізації нашої мережі.

Для бездротового сегменту мережі немає сенсу застосовувати якийсь з наявних протоколів, та буде опрацьований власний підхід обробки даних на

цьому рівні. Деталі цього рішення будуть поясненні пізніше після обґрунтування вибору протоколів і технологій більш нижчих рівнів.

Центральний шлюз бездротової мережі теплиці для зв’язку з мережею підприємства(а можливо й далі з будь-якою глобальною мережею) буде використовувати протокол HTTP(s) (англ. - Hypertext Transfer Protocol secure протокол передачі гіпертексу, захищений). З самого початку цей протокол розроблявся для пересилки лише веб сторінок у вигляді HTMLдокументів. Але поступово розвився в протокол який лежить в основи обміну даними в Інтернеті ставши базою цілої ланки протоколів SOAP, REST, WDSL, та інших. Буква s в кінці декларує розширення протоколу для забезпечення безпечної пересилки даних Secure (англ.– безпечний*)*, описує механізм роботи з безпековими сертифікатами та шифруванням обміну даних.

Також плануємо використання протоколів високо рівня з уніфікованої архітектурної платформи відкритого спілкування OPC UA(*Open Platform Communications Unified Architecture,англ.*) задля інтеграції мережі теплиці в систему автоматичного керування виробничими процесами на підприємстві. Application протокол OPC UA працює поверх HTTP(s) або напряму поверх наступного транспортного рівня.

2.2.2 Транспортний рівень. Далі дані передаються по стеку до низу до транспортного рівня, але до переходу розширяються заголовками рівня.Задача транспортного рівня організувати пересилку даних а для цього необхідно ідентифікувати ці дві програми серед потенційної множини програм на кожній точці мережі. Другою важливою задачею транспортного рівня є забезпечення додаткової надійності передачі даних.

Для бездротового сегменту нашої мережі використання якогось існуючого протоколу транспортного рівня є недоцільним. Деталі такого рішення будуть викладені пізніше.

Стосовно комунікації центрального шлюзу мережі теплиці та корпоративною мережею підприємства(та потенційно глобальних мереж) важливо лише згадати що й HTTP(s) й application протоколи архітектурного рішення OPC UA найчастіше використовують протокол з гарантією доставки даних TCP (Transmission Control Protocol,англ.- протокол управління передачею). Але можлива конфігурація при якій OPC server буде використовувати менш надійний але більш швидкісний UDP (User Datagram Protocol, англ.— протокол датаграм користувача) .

2.2.3 Мереживний рівень. Разом з транспортними заголовками дані на більш нижчому рівні обгортаються в IP пакети (internet protocol,англ. - протокол міжмережевого спілкування). Задача цього рівня маршрутизація даних між різними мережами або різними сегментами однієї мережі. Вже частково зрозуміло чому в бездротовому сегменті мережі заголовки цього рівня також не важливі та можуть бути пропущені. Всі датчики теплиці будуть в одному сегменті фізичної бездротової мережі, то немає необхідності мати ці заголовки. На етапі ж переходу даних з центрального шлюзу бездротової мережі теплиці в корпоративну мережу підприємства, чи далі в будь-яку з глобальних мереж можливо додати необхідні заголовки декількох пропущених наразі рівнів.

2.2.4 Фізичний рівень. Нарешті переходимо до найнижчого фізичного рівня, в стеку TCP/IP він поширюється також на канальний рівень. Існує декілька технологій, які можуть підійти для реалізації бездротової мережі теплиці.

2.2.4.1 Bluetooth забезпечує високошвидкісну передачу даних відстань до 10 метрів по напрямку точка – точка, яку відносять до PAN(Personal Area Network, англ., мережа однієї людини), зазвичай використовується для бездротових навушників, маніпуляторів, клавіатур, тощо.

2.2.4.2 Bluetooth LE- Bluetooth з низьким енергоспоживанням LE (Low energy, англ). Bluetooth LE дозволяє значно скоротити енергоспоживання та витрати і при цьому підтримує такий самий діапазон підключень, як і класична технологія Bluetooth. Ця технологія швидко набуває популярності у сфері побутової електроніки, оскільки вона економічна і забезпечує тривалу роботу від акумулятора.

2.2.4.3 Wi-Fi/802.11. На сьогодні саме популярне бездротове з’єднання, дефакто Wi-Fi став стандартним варіантом бездротової мережі для дому та на підприємствах. Але цей варіант не підходить нам через обмежений діапазон дії та постійне споживання енергії. Для подолання цих недоліків було розроблене спеціальне розширення протоколу Wi-Fi HaLow.

2.2.4.4 IEEE-802.11Wi-Fi HaLow– це, по суті, оптимізована форма Wi-Fi для IoT яка працює в SUB-1Hz. Він пропонує низьке енергоспоживання, широке охоплення та збільшену кількість вузлів на точку доступу.

2.2.4.5 IEEE 802.15.4 Стандарт радіочастотних пристроїв для бездротового з'єднання з низьким енергоспоживанням. Працює в SUB-1Hz діапазоні, в залежності від країни працює на різних частотах.

2.2.4.6 Z-Wave Багатозв'язкова мережа, яка використовує низькоенергетичні радіохвилі для зв'язку між пристроями, де кожна точка може бути ретранслятором (крім пристроїв, що працюють на батарейках) для передачі сигналів, що забезпечує огинання перешкод і зон з поміхами. При запуску мережі кожен виконуючий компонент запам'ятовує оптимальний маршрут для доставки даних до необхідного отримання.

2.2.4.7 LoRaWAN Глобальні мережі великого радіусу дії (LoRaWAN) забезпечують зв'язок між мобільними захищеними двонаправленими пристроями на відстані від 500 м до більш ніж 10 км. Може працювати від акумуляторів.

Проаналізувавши плюси та недоліки кожної технології а також наявність на ринку обладнання було обрано стандарт IEEE 802.15.4. Для прямого зв’язку з датчиками доцільно використовувати BlueToothLE.

2.3. Стандарт IEEE 802.15.4

2.3.1 Загальні положення.

У порядку проектування бездротової мережі розглянемо більш детально стандарт IEEE 802.15.4.Стандарт 802.15.4 визначає фізичний рівень і рівень управління доступом до середовища MAC(Media Access Control, англ. – доступ до передаючою середи) моделі роботи мережі з відкритими системами OSI. Базовим стандартом із самими останніми є 802.15.4a/b, 802.15.4c для Китаю, 802.15.4d для Японії, 802.15.4e для промислових додатків і 802.15.4g для інтелектуальних інженерних мереж.

Базово стандарт побудований з припущенням, що типовий зв'язок буде здійснюватися на відстані близько 10 м, але стандарт не встановлює вимоги до потужності передатчика. Цей параметр регулюється нормативними документами в області радіозв'язку, специфічними для кожної держави. Найбільше поширення на ринку мають передатчики з потужністю 1 мВт, які забезпечують зв'язок на відстані до 10 м в приміщенні, а також передатчики з потужністю 10 мВт, що збільшують цю відстань до 80 м в приміщенні і до 1 км в умовах прямої видимості. Дальність зв'язку також залежить від використовуваного діапазону частот і застосування спеціальної конструкції антени.

Перша специфікації стандарту IEEE 802.15.4-2003, випущена в 2003 році. Вона декларує для Європистандарт, що містить 1 канал в діапазоні 868 МГц та швидкістю до 20 Кбіт/с, для Північної Америки - 10 каналів в діапазоні 915 МГц та швидкістю до 40 Кбіт/с. Срандарт в Китаї дозволяє використання 314-316 МГц, 430-434 МГц, And 779-787 МГц частотних

діапазонів. Можливо використання 16 каналів в діапазоні 2450 МГц та швидкістю до 250 Кбіт/с.

2.3.2 Мережеві топології. Стандарт передбачає підтримку трьох видів мережевих топологій: зіркову (*Star, англ.*) і однорангову (*Peer-to-Peer,англ.*) та змішану.

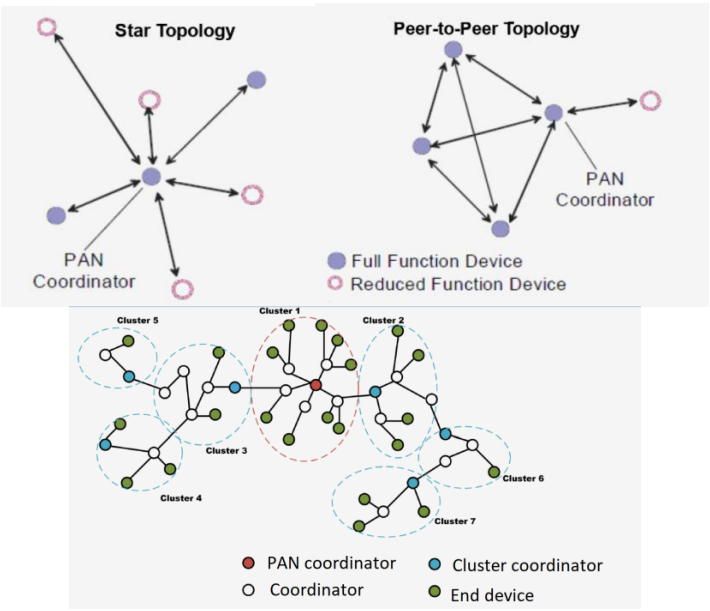


Рис. 2.3.2.1 Топології IEEE 802.15.4

Координатор повинен бути в мережі з будь-якою топологією, однак однорангова відрізняється тим, що будь-який пристрій може обмінюватися даними з будь-яким іншим, якщо він знаходиться в зоні досягнення радіозв'язку, у той час як у зірковій топології будь-який пристрій може взаємодіяти тільки з координатором. В роботі буде використана зіркова топологія.

2.3.3 Передача даних в умовах множинного доступу. Доступ до каналу IEEE 802.15.4 заснований на принципі CSMA-CA (Carrier Sense Multiple Access With Collision Avoidance, англ.- множинного доступу з прослуховуванням несучої хвилі та уникненням колізій). Підтримуються як мережі без маячків, так і з маячками.

У мережах без маячків колізії трапляються тільки з спеціалізованими jam-сигналами. Коли пристрій планує почати передачу, він надсилає мережу

спеціальний джем-сигнал і в якийсь час очікує аналогічних сигналів від інших джерел вилучень. Якщо jam-сигналів від інших передаючих пристроїв не відбувається, пристрій починає передачу. Якщо же з'являється «чужий» сигнал, то передатчик «засинає» на випадковий проміжок часу, а потім знову пробує почати передачу. У такому випадку одночасна передача може вийти тільки з одного пристрою, що підвищує продуктивність мережі. Однак необхідність очікування вільного каналу позначається на швидкості обміну повідомленнями.У мережах з маячками використовується слотовий (тактований) механізм CSMA-CA або ALOHA. За допомогою розсилки спеціальних фреймів-маячків координатор синхронізує передачу даних, тим самим зменшуючи появу колізій.

2.4 Фінальний вибір протоколів високих рівнів.

Ми вже визначилися, що для бездротового сегменту мережі є доцільним спростити навантаження на високих рівнях шляхом оптимізації.

Час розглянути це питання більш докладно. По-перше необхідно згадати що для стандарту IEEE 802.15.4 є декілька реалізацій протоколів високого рівня.

2.4.1 ZigBee – не розглядався як варіант, бо є комерційною реалізацією ZigBee Alliance, тому його використання може потягнути додаткові зайві затрати

2.4.2 6 LoWPAN (англ. IPv6 overLowpowerWirelessPersonal Area Networks) - цікавий стандарт, який можливо за необхідності буде використаний в майбутньому.Наразі було вирішено його не використовувати, розглянемо чому.

На кінцевої точці мережі в теплиці буде працювати лише ода програма. На додаток стандарт IEEE 802.15.4 забезпечує гарантію передачі даних на

канальному рівні. Тобто використання заголовків транспортного рівня не додає ефективної користі. Як вже було сказано, бездротовий сегмент окремої теплиці є окремою фізичною мережею, в рамках якого немає взаємодії з іншими мережами. Ця задача буде реалізовано в програмному забезпечені центрального шлюзу теплиці. Інтерфейс взаємодії клієнт-сервер рівня додатків ми можемо реалізувати самостійно. Плюсів немає, розглянемо мінуси.

2.4.3 Заголовки високих рівнів в фреймі.Заголовки високих рівнів використовують майже половину розміру 802.15.4 фрейму. У 802.15.4 максимальний розмір PSDU (Physical Layer Service Data Unit , англ. – одиниця даних фізичного рівня) становить 127 байтів. 25 байт займають заголовки канального рівня (без захисту). При додаванні захисту на рівні каналу передачі даних (AES-CCM-128) використовується ще 21 байт.Залишаться доступним лише 81. Враховуючи навантаження через заголовки IPv6 (40 байтів), та UDP (8 байтів) або TCP (20 байтів). В результаті залишається 33 байти для UDP або 21 байт для
TCP

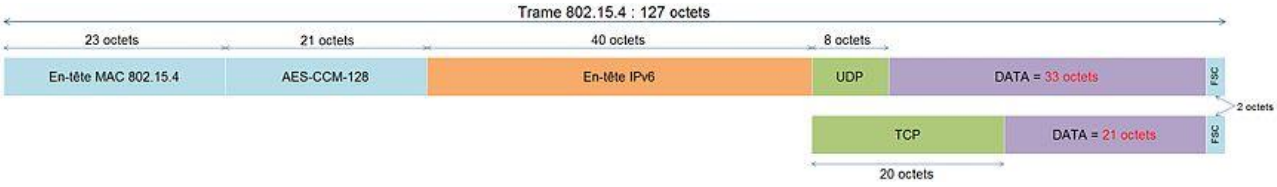


Рис. 2.4.3.1 - Інтегрування пакета IPv6 в кадр 802.15.4

На додачу є додаткове навантаження обладнання, треба фрагментувати пакет IPv6 на кілька кадрів 802.15.4, а віддалене обладнання має повторно зібрати всі отримані кадри 802.15.4, щоб відновити вихідний пакет IPv6. Ці завдання споживають ресурси (пам'ять та ЦП) і створюють затримку (буферизація, час створення/перевірки заголовків та інше).

2.5 Tehas Instruments, CC1350 SimpleLink™ Ultra Low Power Dual Band Wireless MCU.

Рис. 2.5.1 – Tehas InstrumentsCC1350 SimpleLink™ Ultra Low Power Dual Band Wireless MCU

2.5.1 Загальний опис. Компанія Tehas Instruments випустила бездротовий мікроконтролер CC1350, який повністю відповідає нашим потребам в реалізації бездротової мережі теплиці. Пристрій використовує кілька процесорних ядр, що з них відповідає за власну задачу, що дозволяє виконувати багато-поточні задачі.

Дводіапазонне радіо зі своїм власним мікроконтролером Cortex-M0 здатне працювати як в діапазоні нижче 1 Ггц, так і в діапазоні 2,4 ГГц.

У діапазоні до 1Ггц є вибір поміж433/868/932 МГц.Підтримуються фірмові формати передачі данихZigBee, а також стеки протоколів 802.15.4g, TI-MAC і 6LoWPAN. У діапазоні 2,4 ГГц мікросхема CC1350 працює в режимі Bluetooth Low Energy.

Центральна точка (концетратор) мережі датчиків приймає дані по радіоканалу 433/868/932 МГц і може ретранслювати прийняті дані в діапазоні 2,4 ГГц в режимі BLE Beacon.

2.5.2 Енергоспоживання. CC1350 має на борту два процесорних ядра для виконання коду користувача – високопродуктивне 32-бітне ядро Cortex-M3 і спеціалізований контролер датчиків, побудований на базі мікроспоживаючого 16-бітного RISC-ядра. CC1350 відрізняється рекордними показниками по енергоспоживанню – в режимі безперервного прийому споживання радіо становить 5,4 мА, в режимі передачі – 10,5 мА. Датчики

також можуть працювати з дуже низьким енергоспоживанням за допомогою конфігурації максимально часу сну.

2.5.3 Підтримка драйверів та програмних бібліотек. Системи керування живленням і годинником, а також радіосистемами пристрою CC1350 потребують спеціальної конфігурації і обробки програмним забезпеченням, яке було реалізовано в TI-RTOS. Повний TI-RTOS і драйвери пристроїв пропонуються безкоштовно у вихідному коді.

Для оцінки можливостей роботи в двох діапазонах у складі програмного забезпечення TI-RTOS 2.20 для CC13xx/CC26xx SimpleLink™ Wireless MCU наведено приклад дводіапазонного концентратора сенсорної мережі WSN DualModeConcentrator. TI рекомендує використовувати цю програмну структуру для розробки всіх програм на пристрої.

2.5.4 Технічні складові CC1350:

- Дводіапазонний (Sub-1 GHz та 2.4 GHz) мікроконтролер
- Продуктивне ядро ARM® Cortex®-M3, 48 МГц, 128KB Flash, 20+8KB SRAM
- Периферія може бути призначена на будь-який вихід мікросхеми
- 4 таймери; АЦП 12-біт; 2 х компаратора, джерело струму, UART, 2× SSI (SPI, MICROWIRE, TI), I2C, I2S, RTC, AES-128, TRNG, датчик температури
- Підтримка восьми сенсорних кнопок
- Вбудований у чіп DC-DC перетворювач
- Просте сполучення з мікросхемами розширення дальності зв'язку CC1190 та CC2592 Широкий діапазон робочої напруги: 1,8-3,8 В
- Споживання RX: 5.4 мА (Sub-1 GHz), 6.4 мА (Bluetoothlowenergy, 2.4 GHz); TX за +10 dBm: 13,4 мА (Sub-1 GHz), TX за +9 dBm: 22,3 мА (Bluetoothlowenergy, 2.4 GHz) Контролер датчиків при

зчитуванні АЦП 12-біт один раз на секунду: 0,95 µA Очікування (Standby): 0,7 µA (працює RTC, збереження RAM та CPU)

2.5.5. Можливості радіо модулю:

- Відмінна чутливість –124 dBm у режимі Long-RangeMode,
  - 110 dBm при 50 kbps (Sub-1 GHz)
  - 87 dBm у режимі Bluetoothlowenergy
- Програмована вихідна потужність
  - до +14 dBm у діапазонах Sub-1 GHz
  - до +9 dBm у діапазоні at 2.4 GHz (Bluetoothlowenergy)

2.5.6. Моделі використання CC1350. Інтеграція платформи SimpleLink дозволяє користувачам додавати будь-які комбінації пристроїв.Потенційно можливо використовувати смартфон як шлюз в глобальні мережі.Також є можливість з’єднання смартфона з напряму з будь-яким датчикам використовуючиBluetoothLE.

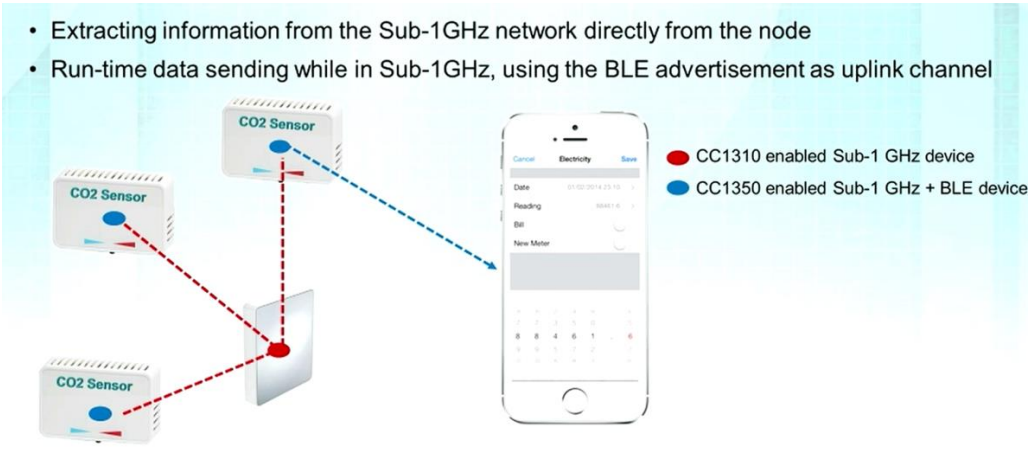


Рис. 2.5.6.1 – поєднання смартфону напряму з датчиком мережі

Але при рекомендованому підході основну мережу строять на SUB-1Ггцмодулі. Концентратор з’єднується з шлюзом в загальну мережу підприємства. BlueToothLEмережа використовується лише для прямого з’єднання датчика з мобільним пристроєм.

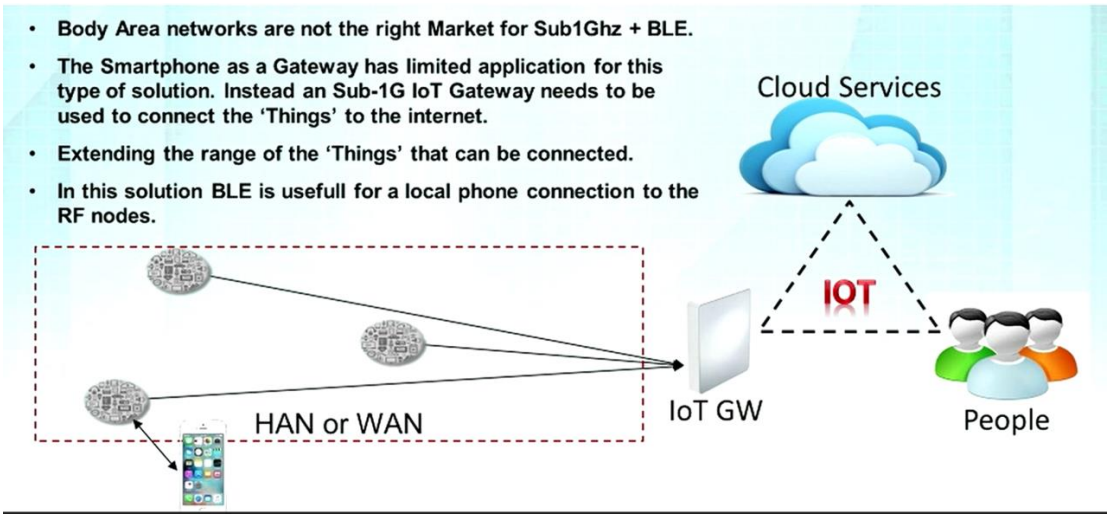


Рис 2.5.6.2 – Схема реалізації бездротової мережі.

2.5.7. Демонстраційні плати LaunchPad CC1350. Додатково виробник випускає та продає експериментальні плати де вже реалізовані демонстраційні можливості цього обладнання.

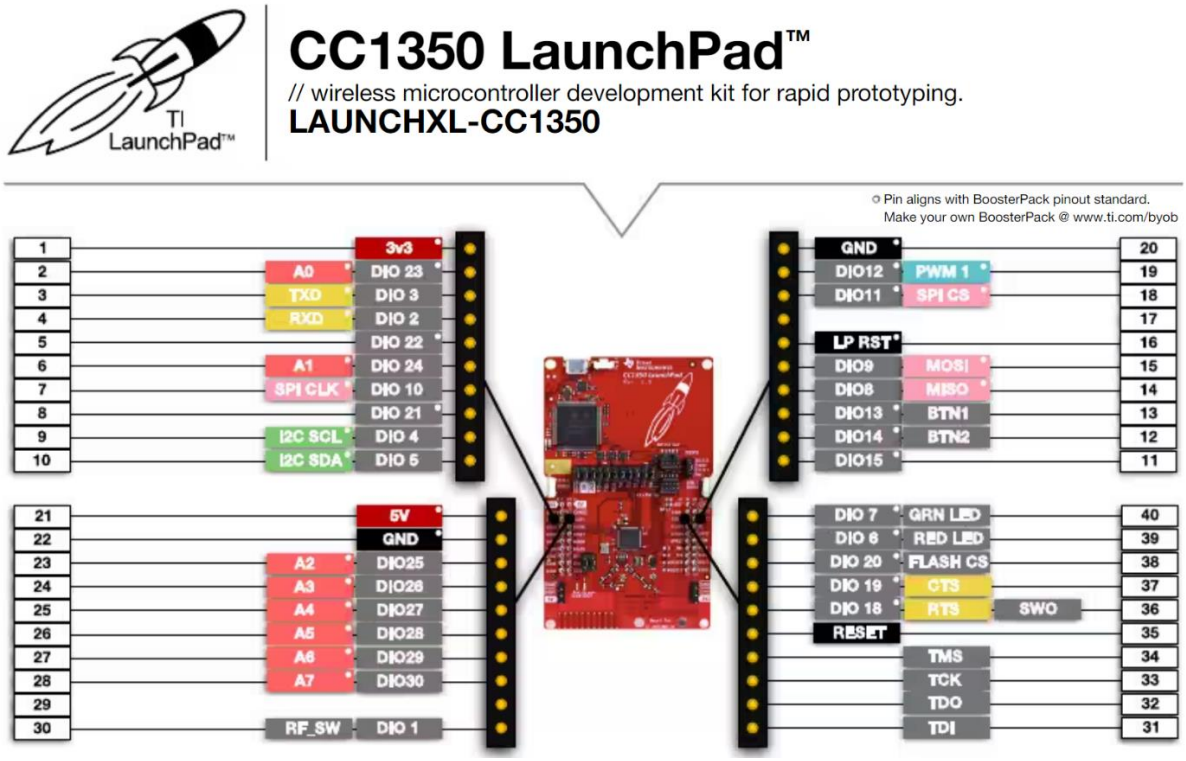


Рис 2.5.7.1 – Стартовий модуль для початку досліджень CC1350

2.6 Фінальна схема бездротової мережі теплиці. Підходимо к моменту коли ми маємо все необхідне для завершення проектування схеми нашої

мережі, з використанням конкретного обладнання та розумінням потоку даних через всі рівні протоколів мережевої архітектури. Рисунок 2.6.1 показує цю схему.

Бездротова технологія дозволяє розподілити різні датчикив залежності від нагальних технічних потреб. Нам знадобляться декілька видів датчиків. Температури: датчики під листом, в близькості до ґрунту, та інфрачервоні датчики над рослинами, які оцінюють температурний режим поверхні листя). Датчики вологи, як ґрунту в безпосередній близькості до рослин, так і вологості повітря. Датчики вмісту вуглекислого газу в повітрі. Спектральні датчики опромінення ФАР в безпосередній близькості до листа. Датчики з'єднаються з радіо модулем CC1350. Це дає нам можливість отримувати на мобільні пристрої по технології BlooeTooth LE по системі точка – точка. Також в теплиці монтується центральний модуль CC1350 виконуючий роль концентратора. Всі модулі CC1350 під'єднанні до датчиків передають дані на концентратор використовуючи прямий доступ до MAC рівня радіо середи SUB-1Hz. Концентратор пов'язаний за допомогою USB кабеля до Linux системи. Це може бути однопалатний комп'ютер Beagle Bone Black, чи більш стаціонарна система.

Linux система упаковує данні в стандартні пакети TCP/IP та пересилає (організовує маршрутизацію) їх далі використовуючи для цього HTTPs та OPC UA верхнього рівня.

В подальшому можлива імплементація програмного модуля CoDeSys PLC (Controller Development System Programmable Logic Controller , англ. Система розробки контролерів, програмований логічний контролер) для Linux/інші операційні системи, чи інша одноплатна комп'ютерна архітектура, чи навіть реалізація з використанням повноцінних PLC модулів промислових контролерів.

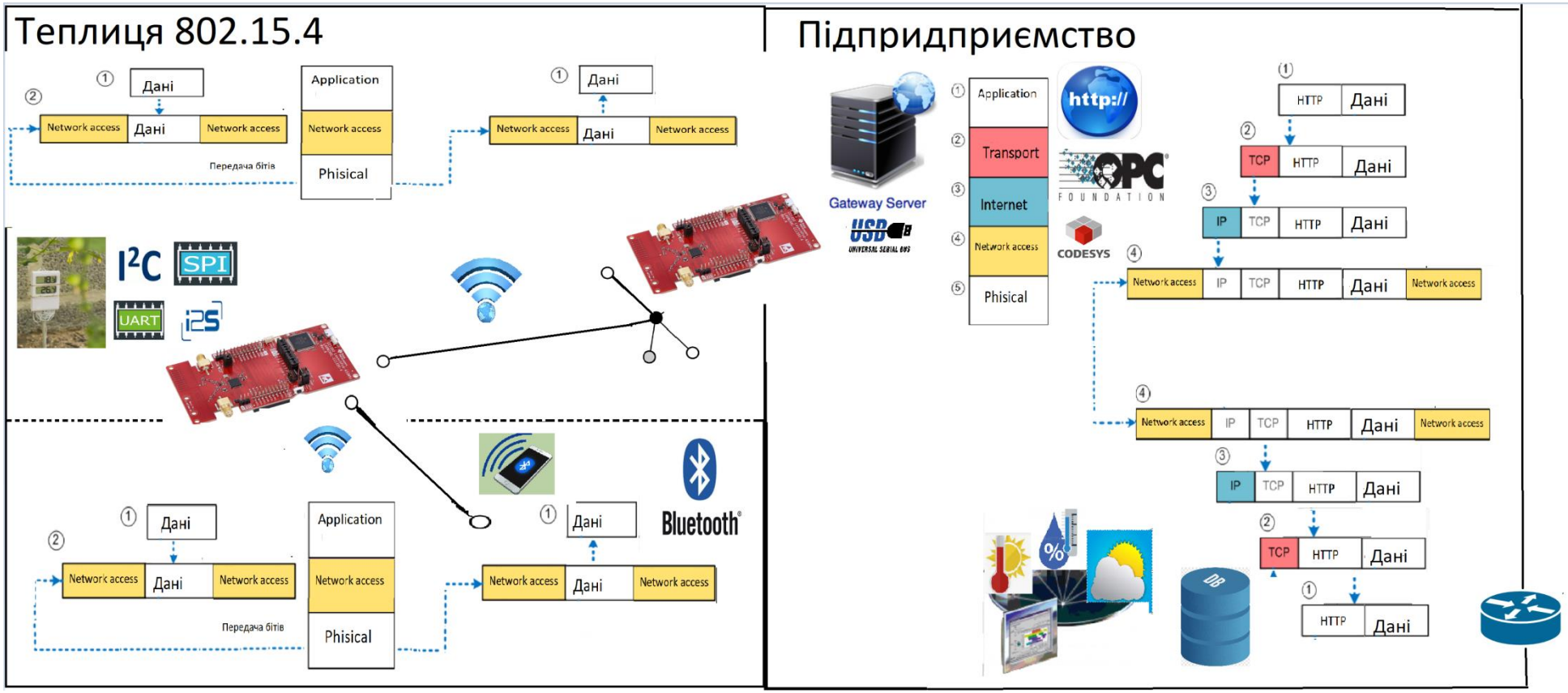


Рис 2.6.1 – Схема бездротової мережі на базі CC1350

2.7 Програмування модулів CC-1350 для реалізації бездротового сегменту мережі.

2.7.1 CCS IDE.

Для програмування плата CC1350 LaunchPad знадобиться CCS (Code Composer Studio) IDE (англ. integrated development environment — інтегрована середа розробки). За необхідності можливо скачати та встановити з офіціального сайту Tehas Instruments (потребує реєстрації). В роботі використовувалась версія 12.3.0.00005

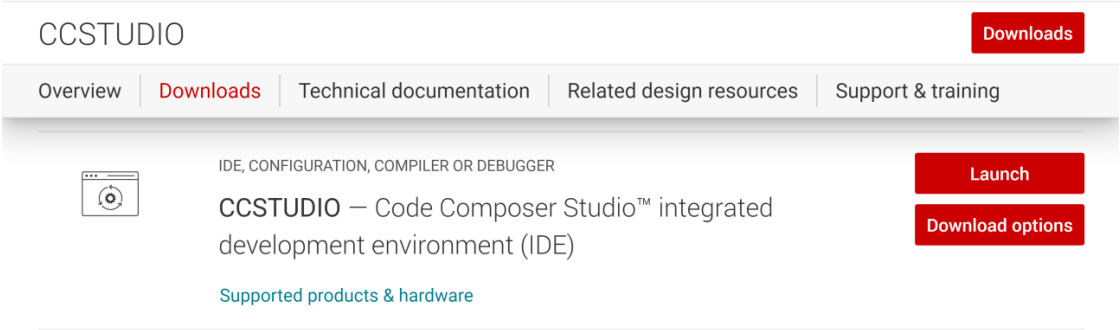


Рис 2.7.1.1 CCI IDE інсталяція на сайті виробника [18]

2.7.2 Імпорт sensor та coprocessor базових програмних шаблонів.

В колекціях програмних шаблонів – бібліотек можна знайти базові шаблони рішення, які рекомендовані виробником для подальшої розробки. Для реалізації наших задач знадобиться два шаблони. Імпортуємо co-processor та sensor програмні модулі для плат CC1350.

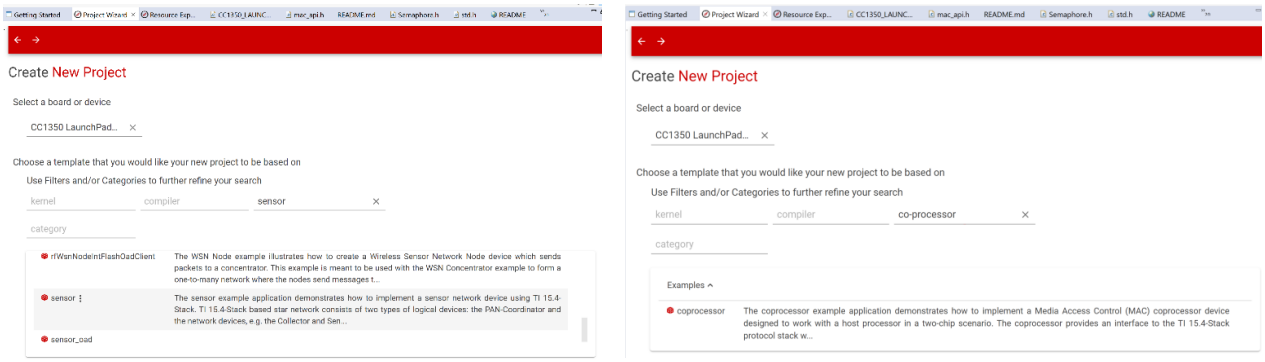


Рис. 2.7.2.1 – мінімально необхідні програмні модулі

2.7.3 Компіляція програмних модулів та прошивка плат CC1350.

Компілюємо обидва модулі: тиснемо кнопку build для кожного модулю.

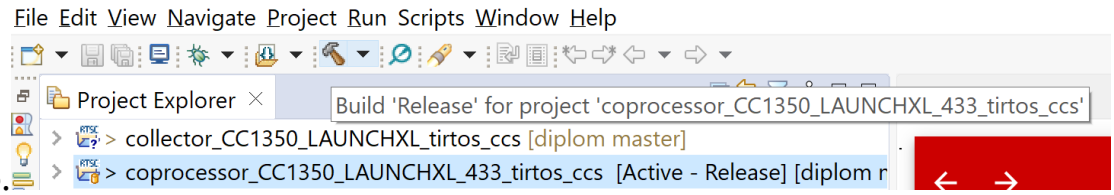


Рис. 2.7.3.1 – Компіляция програмних модулів

Далі знадобитеся щонайменше дві плати, одна для центрального модулю, друга для модулю датчика. З'єднуємо персональний комп'ютер з кожною платою за допомогою USB інтерфейсу. Прошиваємо кожну плату програмним кодом за допомогою кнопки flash

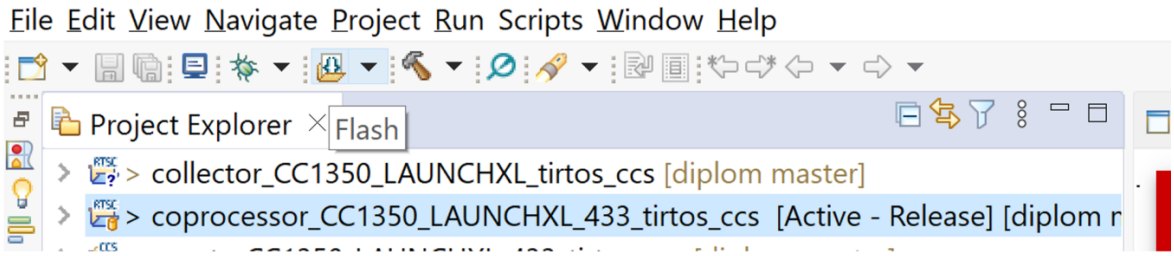


Рис. 2.7.3.2 – прошивка програмних модулів на плату CC1350

2.7.4 15.4-Stack Linux SDK. В якості базового рішення шлюзу в корпоративну мережу підприємства знадобиться Tehas Instruments 15.4-Stack Linux. Інсталяція також доступна на офіціальному сайті Tehas Instruments

Рис. 2.7.4.1 Інсталяція 15.4-Stack LinuxSDK сайті на сайті виробника [19]

Щоб інсталювати TI 15.4-Stack Linux Gateway треба запустити скачаний файл ti15.4stack_linux_x64_2_01_00_xx.run. Це потрібно робити на комп'ютері з операційною системою Linux. Була обрана остання версія Ubuntu Linux server. Для старту інсталяції необхідно дати право запуску:

$chmod +x ti154stack_linux_x64_4_40_00_03.run

Запускаємо інсталяцію:

$./ti154stack_linux_x64_4_40_00_03.run

Після успішної інсталяції буде создана папка *ti* (по замовчуванню, можливо задати іншу назву) з декількома програмами скомпільованими для звичайного процесора персонального комп'ютера та операційної системи Linux. Також присутні програмні модулі скомпільовані для процесора однопалатного комп'ютера Beagle Bone Black. Також присутній код програм на C++ для потенційної компіляції під інші архітектури.

2.7.5 NPI server.TI 15.4-Stack Linux SDK включає NPI server (network processor interface server,англ. - сервер мережевоїтрансляції). NPI Server це службова програма, яка перекладає USB/Seralпакети з та вTCP/IP та яка по суті також частиною координатора архітектури 802.15.4

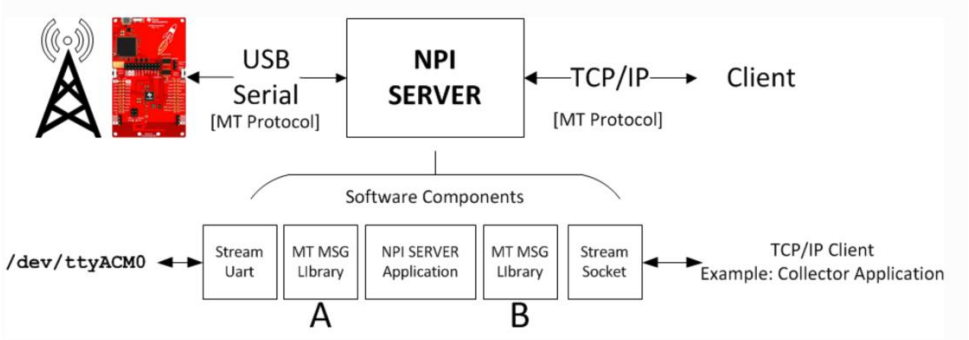


Рис. 2.7.5.1 схема роботи NPI серверу.

Запуск программного модулю collector. Він виконує задачі сервера NPI та частково coordinator з архітектури 802.15.4. По перше необхідно задати право запускати програмний модуль (робиться один раз).

$cd ti/ti154stack_linux_x64_4_40_00_03/example/collector/

Тепер безпосередньо запускаємо collector (плата CC1350 з програмним модулем coprocessor повинна бути під’єднана за допомогою USB кабелю):

$ sudo ./run_collector.sh

TI 15.4-Stack Linux також включає Web Server, що представляє базовий інтерфейс користувача реалізований за допомогою Node.JS

Запуск веб-серверу. Тут також необхідно задати право запускати програмний модуль (робиться один раз):

$ cd ti/ti154stack_linux_x64_4_40_00_03/prebuilt/gateway/

$ chmod +x run_gateway.sh

Далі запускаємо:

$ ./run_gateway.sh

2.7.6 Web server.

Webserver стартує на TCP порту 1310. Вже можна відкрити Web browser таполучити доступ к бездротової мережі.

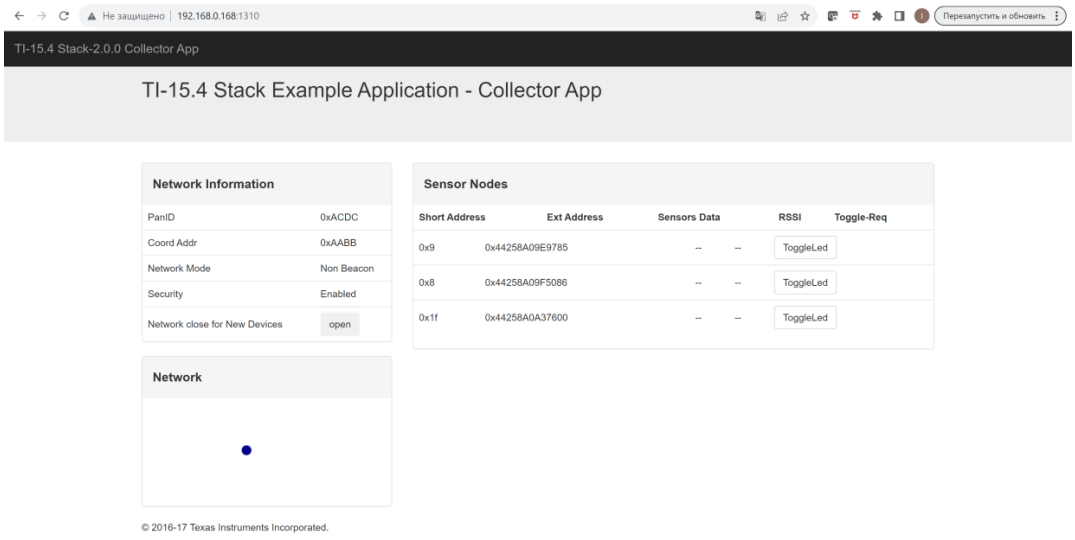


Рис. 2.7.6.1 Інтерфейс користувача для доступу в бездротову мережу.

Інтерфейс включає декілька областей. Інформацію про мережу: адреса мережі та координатору, чи використовується безпекові обмеження, та механізм маяків. Графічне відображення топології мережі. На даному етапі можливо бачити лише центральну точку – координатор. Область з переліком модулів датчиків. На даному етапі ми бачимо три сенсори які зараз не працюють(присутні в списку так як вже були доданні до мережі раніше). На даному етапі можна натиснути кнопку open в секції інформації мережі, це дозволить сенсорам доєднатися до мережі. Запускаємо всі доступні плати з програмним модулем sensor та просто зачекаємо. Через деякий час ми побачимо обновлення списку сенсорів та обновлення в мережевій топології

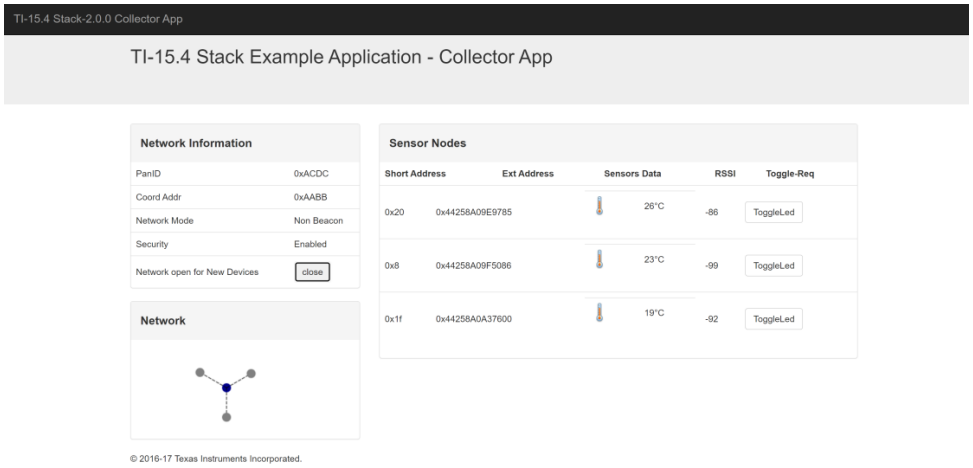


Рис. 2.7.6.2 Відображення змін в бездротовій мережі теплиці

2.8 Алгоритм роботи мережі на верхньому рівні

Як вже було вказано вище, дана реалізація бездротової мережі теплиці не використовує протоколів високого рівня. Дані передаються безпосередньо на канальний рівень архітектури IEEE 802.15.4. Тобто формат повідомлень, обробка помилок, конфігурація мережі та інше реалізується в програмних модулях – шаблонах, рекомендованих виробником як бази для реалізації мереж для власних потреб. Розглянемо деякі деталі цієї реалізації для подальшої модернізації та розширення функціоналу.

При старті концентратор знає свій адрес та адрес своєї мережі, та широкомовно надсилає цю інформацію в радіоканал. Має два статуси: 0 – мережа закрита для з'єднання сенсорів, 1- мережа відкрита. Цей статус регулюється програмою чи адміністратором мережі. Статус сенсора має дещо ширший діапазон. При подачі електроживлення девайс получає первинний статус 0 – не робочий, в очікувані завершення ініціалізації. Потім переходить в статус 1– старт в процесі, в цей момент йде пошук мережі для з'єднання. При знаходженні підходящої мережі переходить в статус 2 та починає процедуру з'єднання з мережею. При успішному з'єднанні, отримує статус 3 – працює у мережі. Статус 4 – це також «з'єднався та працює у мережі» але вже після повторного з'єднання при появі помилок в роботі мережі. Статус 5 означає якраз втрату з'єднання з концентратором. Статус 6 – отримана відмова в доступі до мережі.

2.9 Пакети даних верхніх рівнів мережі.

2.9.1 Пакет конфігурація сенсорів в мережі. Сенсор стартує роботу з конфігурацією по замовченню, але після з'єднання до мережі мусить її змінити на прийнятну для роботи в цій мережі. Концентратор шле конфігурацію сенсора в повідомленні типу 1.

Таблиця 2.9.1.1 - перелік полів в пакеті запита зміни конфігурації сенсором.

Назва поля	Опис	Довжина
Ідентифікатор команди	Тип повідомлення: Smsgs_cmdIds_configReq=1	1 байт
Формат даних сенсорів	Перелік даних необхідних для відсилки сенсором в повідомлені Smsgs_dataFields	2 байт
Інтервал відсилки даних сенсорів	Налаштування періоду відсиланняданих, долі секунди	4 байт

Інтервал зв’язку	Налаштування періоду через який сплячийпристрій зобов’язаний вийти назв’язок, долі секунди	4 байт

2.9.2 Пакет відповіді на пакет 1. Обробивши конфігураційний пакет та застосувавши конфігурацію сенсор шле концентратору відповідь, повідомлення типу 2.

Таблиця 2.9.2.1 - перелік полів в пакеті відповіді зміни конфігурації сенсором.

Назва поля	Опис	Довжина
Ідентифікатор команди	Тип повідомлення: Smsgs_cmdIds_configReq=2	1 байт
Формат даних сенсорів	Перелік запрошених даних які підтримуються сенсором	2 байт
Інтервал відсилки даних сенсорів	Налаштування періоду відсилання даних, долі секунди, 0 означає відключення відсилки даних	4 байт
Інтервал зв’язку	Налаштування періоду через який сплячий пристрій зобов’язаний вийти на зв’язок, долі секунди, 0 означає відсутність сплячого періоду	4 байт

2.9.3 Контроль стану сенсорів. Для моніторингу стану сенсору, використовуються контрольні повідомлення типу 3 та 4. 3 це запит концентратора сенсору, 4 – відповідь сенсора концентратору. Два дуже коротких повідомлення в 1 байт, обмін котрими повинен підтверджує робочій

стан сенсора. короткий запит від концентратору сенсору, для розуміння чи сенсор ще працює.

2.9.4 Пакет відправки даних сенсорів концентратору. Пересилка даних сенсора здійснюється повідомленням типу 5. В пакеті мають бути присутні всі данні, які були узгоджені для відсилки при обміну повідомленнями типу 1 та 2 в полі Frame Control.

Таблиця 2.9.4.1 – Перелік полів повідомлення даних сенсора концентратору.

Назва поля повідомлення	Опис	Довжина
Команда	Тип повідомлення, Smsgs_cmdIds_sensorData = 5	1 байт
Адреса відправника	Розширена Адреса	8 байт
Підтвердження набору даних	Бітова маска показниківсенсорів	2 байт
Датчик температури	Ambience temperature– температура повітря	2 байт
	Object temperature – температура об’єкта	2 байт
Датчик світла	Поле для данихсвітлового датчика. Виробник бачить тут OPT2001 light sensor	2 байт
Датчик вологості	температура датчика вологості Виробник пропонує TI HCD1000 humidity sensor	2 байт
	дані вологості. Виробник пропонує TI HCD1000 humidity sensor	2 байт
Статистика	загальне число намагань з’єднатись з мережею	2 байт
	загальне число невдалих намагань з’єднатись з мережею	2 байт
	загальне число намагань послати	2 байт

	повідомлення	
	загальне число відісланих повідомлень	2 байт
	загальне число отриманих трек повідомлень	2 байт
	загальне число отриманих намагань відсилки трек відповідей	2 байт
	загальне число успішних трек відповідей	2 байт
	загальне число отриманих конфігураційних повідомлень	2 байт
	…	Всього статистика займає 24 * 2 байт = 48 байт
Конфігураційна інформація	Налаштування періоду відсилання даних	4 байт
	Налаштування періоду через який сплячий пристрій зобов’язаний вийти на зв’язок	4 байт

2.9.5 Пакет управляючого повідомлення по зміні статусу сенсору. Повідомлення зміни стану пристроїв з’єднаних з кінцевою точкою мережі. Пакет типу 6 змінює стан світлодіоду на борту плати сенсора. Це дуже загальне управляюче повідомлення, зроблене як приклад для реалізації більш корисних управляючих повідомлень. Довжина пакета 1 байт.

2.9.6 Пакет відповіді на управляюче повідомлення типу 7. Несе в собі статус виповненої/невиповненої управляючої команди. Довжина пакета 2 байт

2.9.7 В колекції шаблонів існує ще декілька типів пакетів, також є можливість розробки власних. У таблиці 2.9.7.1 неведене неповний перелік основних шаблонних команд.

Таблиця 2.9.7.1- повідомлення в бездротовій мережі(перелік не повний)

Тип повідомлення	Опис
Smgs_cmdIds_broadcastCtrlMsg = 10,	Широкомовне повідомлення контрою сенсорів
Smgs_cmdIds_KeyExchangeMsg = 11,	Повідомлення обміну ключами

2.10 Коди помилок верхнього рівня.

В програмному шаблоні вже розроблений механізм обробки всебічних помилок. Частковий перечень кодів помилок наведений нижче у таблиці.

Таблиця 2.10.1 Частковий перелік помилок

Код помилки	Опис
ApiMac_status_success = 0,	Операція успішна
ApiMac_status_subSystemError = 0x25,	Помилка подсистеми
ApiMac_status_commandIDError = 0x26,	Помилка ідентифікатора команди
ApiMac_status_lengthError = 0x27,	Помилка довжини повідомлення
ApiMac_status_unsupportedType = 0x28,	Тип не підтримується

Підбиваючи результати цього розділу можемо сказати що маємо все необхідне для початку реалізації бездротової мережі теплиці. Деталі під’єднання конкретних датчиків до кінцевих точок мережі будуть розглянуті в наступному роздулі.

РОЗРОБКА МЕТОДИКИ ВИМІРЮВАННЯ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СПЕКТРОМЕТРА ДЛЯ КОНТРОЛЮ ПОТУЖНОСТІ СПЕКТРАЛЬНИХ СКЛАДОВИХ ФІЗІОЛОГІЧНО АКТИВНОЇ РАДІАЦІЇ НА БАЗІ КОМПЛЕКТУ ОПТИЧНИХ СЕНСОРІВ AS7265X ВИРОБНИЦТВА AMS-OSRAM AG

3.1 Сенсор AS7265x та його можливості спектрального аналізу світла

AS7265x складається з трьох сенсорних пристроїв AS72651(ведучий в I_2C з'єднанні), AS72652 і AS72653(відомі по I_2C).

Рис. 3.1.1 – сенсорний пристрій AS72651

Мультіспектральні датчики можуть бути використані для спектральної ідентифікації в діапазоні від видимого до NIR. Кожне з трьох сенсорних пристроїв має 6 незалежних вбудованих оптичних фільтрів, спектральний відгук яких визначається в діапазоні від 410 нм до 940 нм з FWHM(full width at half maximum, англ. - повна ширина на рівні половини висоти) 20 нм.

На ринку є два варіанти демонстраційних плат. Від виробника датчиків ams-OSRAM AG та Sparkfun.

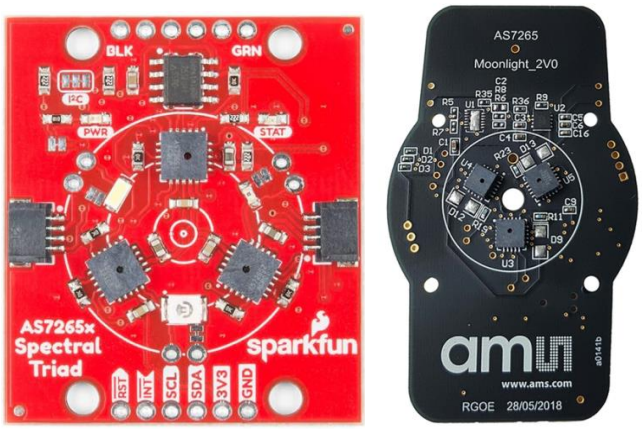


Рис. 3.1.2 – демонстраційні плати комплектів AS72651, AS72652 і AS72653

Технологіянанооптичних нанесених інтерференційних фільтрів дозволяє збирати показники потужності опромінення в розрізі 18 каналів.

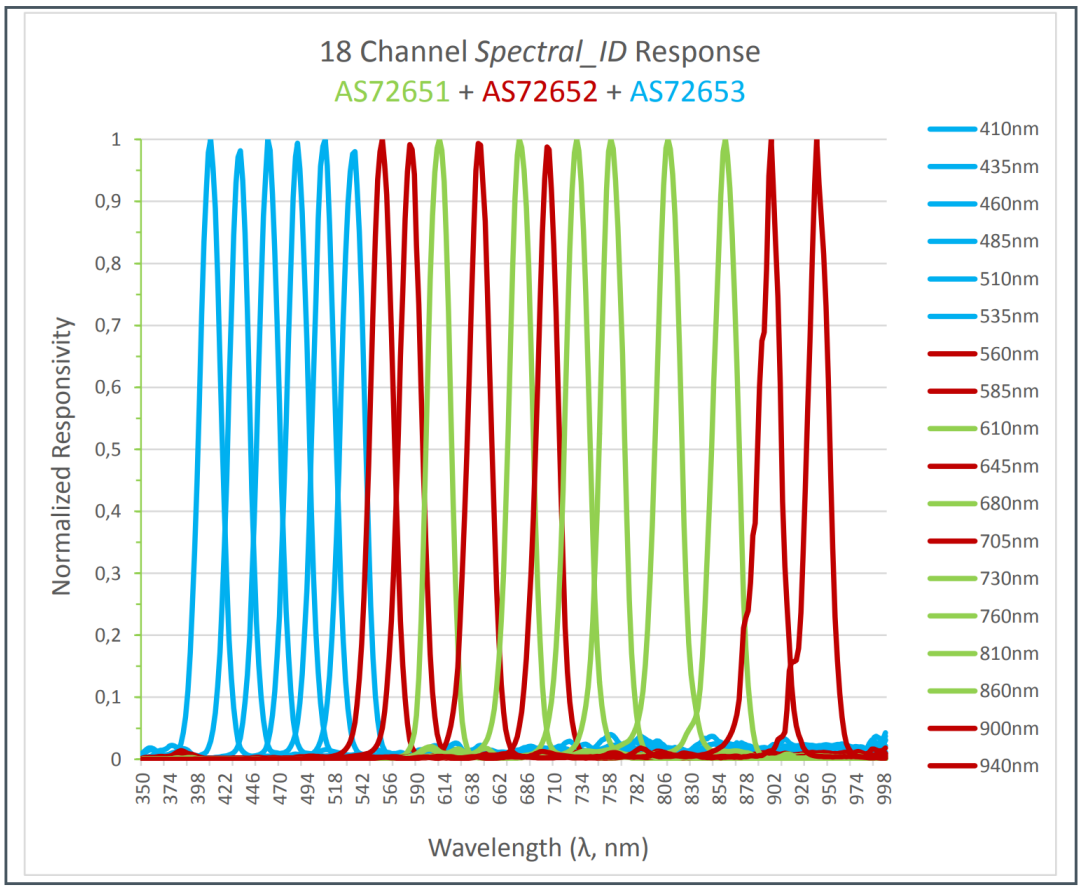


Рис. 3.1.3 Схема чутливості фотодачиківAS72651x

Датчик показує інтенсивність опромінення по кожному каналу мкВт/см²що фактично являється енергією фотона помножену на кількість

фотонів. Енергія фотону залежить напряму від дожени хвилі. Всі інші показники які впливають на цю міру є постійними.

$$E = \frac{hc}{\lambda}$$

h – постійна планка, $6.63 * 10^{-34}$

c – швидкість світла, $3 * 10^{8}$

λ – довжина хвилі

Тобто вирахувати кількість фотонів буде дорівнювати діленню показника датчика по кожному каналу на енергію фотона усереднене по цьому каналу. Для початку треба зчитати значення датчиків. Розглянемо цю задачу більш детально. По перше требо фізично з'єднати плату датчику з платою радіо модулю. Маємо тут два варіанти I_2C та UART.

3.2 Інтеграція плати AS7265x з Launchpad CC1350 по інтерфейсу I_2C

3.2.1 Фізичне з'єднання.

Живлення можна подавати окремо на кожну плату чи через вихід живлення з Launchpad CC1350. Маємо тут вихід землі (-) та 3.3/5В (+). Будемо використовувати з'єднання через Launchpad CC1350 3.3В, то поєднуємо дроти живлення відповідно малюнку. Живлення на Launchpad CC1350 3.3В буде подаватися через USB інтерфейс. Також требf поєднати контакти даних SDA (Serial Data, англ) на обох платах тактування SCL (Serial Clock. англ.)

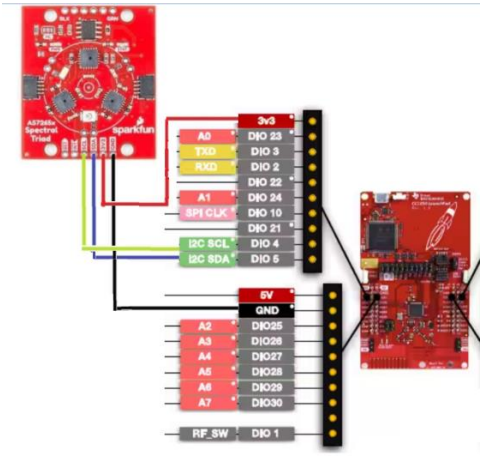


Рис. 3.2.1.1 Схема підключення по I_2C

3.2.2 Опис регістрів. Беремо зофіційної документації виробника опис віртуальних регістрів I_2C[20], с 9-12 які необхідні для вирішення наших задач.

Таблиця 3.2.2.1 - Набір віртуальних реєстрів I^2C для AS72651, який служить головним інтерфейсомз 3 пристроїв AS7265x[20], с 9-12.

Адрес	Опис
0x14	Каліброване значення датчика R, G, A
0x18	Каліброване значення датчикаS, H, B
0x1C	Каліброване значення датчикаT, I, C
0x20	Каліброване значення датчикаU, J, D
0x24	Каліброване значення датчикаV, K, E
0x28	Каліброване значення датчикаW, L, F
0x04	Конфігурація
0x05	Час інтегрування
0x06	Датчик температури
0x08	Сирі данні датчика R, G, A
0x0A	Сирі данні датчикаS, H, B
0x0C	Сирі данні датчикаT, I, C
0x0E	Сирі данні датчикаU, J, D
0x10	Сирі данні датчикаV, K, E
0x12	Сирі данні датчикаW, L, F

Більшість регістрів доступні тільки для читання – дані датчиків. Але час інтегрування та множник GAIN(частина регістра конфігурації) які доступні для запису, що дозволяє змінювати налаштування пристрою стосовно того як він обробляє замір даних. Час інтегрування – період за який проходить фактичний замір інтенсивності світла. Множник дозволяє змінити чутливість прибору.

Час інтеграції =<значення> * ~2,8 мс. Діапазон значень 1-255 (або ~2,8 мс – 714мс)

Можливі значення множника 0=1x, 1=3,7x,2=16x,3=64x

3.2.3 Реалізація в програмному забезпеченні зняття даних датчика AS7265x по інтерфейсу I_2C.

По перше підключити I2C бібліотеку, добавляємо файл заголовуів в список імпорту.

```
36 #include <stdint.h>
37 #include <stddef.h>
38 #include <unistd.h>
39
40 /* Driver Header files */
41 #include <ti/drivers/GPIO.h>
42 #include <ti/drivers/I2C.h>
43 #include <ti/display/Display.h>
```

Рис. 3.2.3.1 – підключення файлу заголовків I2C.h

Далі необхідно створюємо об’єкт для з’єднання з I_2C інтерфейсом

```
491     uint16_t         sample;
492     uint16_t         temperature;
493     uint8_t          txBuffer[1];
494     uint8_t          rxBuffer[2];
495     I2C_Handle       i2c;
496     I2C_Params       i2cParams;
497     I2C_Transaction i2cTransaction;
498     I2C_Transaction readI2cTransaction;
499
500
501     /* Call driver init functions */
502     Display_init();
503     //GPIO_init();
504     I2C_init();
505
506
507     I2C_Params_init(&i2cParams);
508     i2cParams.bitRate = I2C_3400kHz;
509     //i2cParams.transferMode  = I2C_MODE_BLOCKING;
510     i2c = I2C_open(Board_I2C_TMP, &i2cParams);
511
512     if (i2c == NULL) {
513         System_printf("Error Initializing I2C\n");
514     }
515     else {
516
517         System_printf("I2C Initialized!\n");
518     }
519     /* Common I2C transaction setup */
520     i2cTransaction.writeBuf   = txBuffer;
521     i2cTransaction.writeCount = 1;
522     i2cTransaction.readBuf    = rxBuffer;
523     i2cTransaction.readCount  = 2;
524
525
526
527     i2cTransaction.slaveAddress = AS7265X;
```

Рис. 3.2.3.2 – КонфігураціяI_2C з’єднання та створення точки доступу к даним

Вже можна створити з’єднання для подальшого обміну даними

```
527     i2cTransaction.slaveAddress = AS7265X;
528     if (!I2C_transfer(i2c, &i2cTransaction)) {
529         System_printf("Error. No sensor found!\n");
530
531     }
532     else {
533
534     System_printf("Slave address is %d\n", i2cTransaction.slaveAddress);
535
536     uint8_t value = virtualReadRegister(i2c, 0x49, AS7265X_DEV_SELECT_CONTROL);
537
538       if ((value & 0b00110000) == 0){
539           System_printf("Slave1 and 2 are not detected\n");
540       }
541       else {
542           System_printf( "Slave1 and 2 are detected\n");
543       }
```

Рис. 3.2.3.2 – встановлення з’єднання зI_2C, обробка можливих помилок з’єднання.

Тепер час створити механізми читання та запису в регістри.

```
uint8_t readRegister(I2C_Handle i2cBusDesciptor, uint_least8_t address, uint8_t registerName) {
    uint8_t         txBuffer[1];
    uint8_t         rxBuffer[1];
    I2C_Transaction i2cTransaction;
    i2cTransaction.writeBuf   = txBuffer;
    i2cTransaction.writeCount = 1;
    i2cTransaction.readBuf    = rxBuffer;
    i2cTransaction.readCount  = 1;
    i2cTransaction.slaveAddress = address;
    txBuffer[0] = registerName;

    if (!I2C_transfer(i2cBusDesciptor, &i2cTransaction)) {
        Display_printf(display, 0, 0, "Read register problem");
    }

    return rxBuffer[0];
}

uint8_t writeRegister(I2C_Handle i2cBusDesciptor, uint_least8_t address, uint8_t registerName, uint8_t registerValue) {
    uint8_t         txBuffer[2];
    I2C_Transaction i2cTransaction;
    i2cTransaction.writeBuf   = txBuffer;
    i2cTransaction.writeCount = 2;
    i2cTransaction.readBuf    = NULL;
    i2cTransaction.readCount  = 0;
    i2cTransaction.slaveAddress = address;
    txBuffer[0] = registerName;
    txBuffer[1] = registerValue;

    if (!I2C_transfer(i2cBusDesciptor, &i2cTransaction)) {
        Display_printf(display, 0, 0, "Write register problem");
        return false;
    }

    return true;
}
```

Рис. 3.2.3.3 – Читання з регістрів та запис.

Ці прямі інструкції можуть працювати нестабільно із-за появу помилок. Треба обгорнути в механізми надійного спілкування вводячи систему повторної обробки даних в випадку помилки.

```
uint8_t virtualReadRegister(I2C_Handle i2cBusDesciptor, uint_least8_t address, uint8_t virtualA
{

  uint8_t status;

  status = readRegister(i2cBusDesciptor, address, AS7265X_STATUS_REG);
  if ((status & AS7265X_RX_VALID) != 0) //There is data to be read
  {
    readRegister(i2cBusDesciptor, address, AS7265X_READ_REG); //Read the byte but do nothing wi
  }


  uint8_t attempt = 1;

  while (1)
  {
    if(attempt > maxAttempts) {
        Display_printf(display, 0, 0, "virtualReadRegister failed %d", virtualAddr);
        return(0); //Sensor failed to respond
        }
    status = readRegister(i2cBusDesciptor, address, AS7265X_STATUS_REG);

    if ((status & AS7265X_TX_VALID) == 0)
      break; // If TX bit is clear, it is ok to write
    sleep(AS7265X_POLLING_DELAY);
    attempt++;
  }
  writeRegister(i2cBusDesciptor, address, AS7265X_WRITE_REG, virtualAddr);

  attempt = 1;
  while (1)
  {
    if(attempt > maxAttempts)  {
        Display_printf(display, 0, 0, "virtualReadRegister 2 failed %d", virtualAddr);
        return(0); //Sensor failed to respond
    }
    status = readRegister(i2cBusDesciptor, address, AS7265X_STATUS_REG);
    if ((status & AS7265X_RX_VALID) != 0)
      break; // Read data is ready.
    sleep(AS7265X_POLLING_DELAY);
    attempt++;
  }
  uint8_t incoming = readRegister(i2cBusDesciptor, address, AS7265X_READ_REG);

  return (incoming);
```

Рис. 3.2.3.4 – Надійне читання з регістрів, реалізація повторів читання/запису при виявленні помилок

```
void virtualWriteRegister(I2C_Handle i2cBusDesciptor, uint_least8_t address, uint8_t virtualAddr, uint8_t dataToWrite)
{
  while (1)
  {
    if(attempt > maxAttempts) {
        unsigned long startTime1 = getTime();
        Display_printf(display, 0, 0, "virtualWriteRegister problem %d value %d", virtualAddr);
        return; //Sensor failed to respond, dataToWrite

    }
    status = readRegister(i2cBusDesciptor, address, AS7265X_STATUS_REG);
    if ((status & AS7265X_TX_VALID) == 0)
      break; // No inbound TX pending at slave. Okay to write now.
    sleep(AS7265X_POLLING_DELAY);
    attempt++;
  }
  writeRegister(i2cBusDesciptor, address, AS7265X_WRITE_REG, (virtualAddr | 1 << 7));

  attempt = 1;
  while (1)
  {
    if(attempt > maxAttempts) {
        Display_printf(display, 0, 0, "virtualWriteRegister problem 2 %d value %d, %d", virtualAddr, virtualAddr, attempt);
        return; //Sensor failed to respond
    }
    status = readRegister(i2cBusDesciptor, address, AS7265X_STATUS_REG);
    if ((status & AS7265X_TX_VALID) == 0)
      break; // No inbound TX pending at slave. Okay to write now.
    sleep(AS7265X_POLLING_DELAY);
    attempt++;
  }

  writeRegister(i2cBusDesciptor, address, AS7265X_WRITE_REG, dataToWrite);
}
```

Рис. 3.2.3.5 – Надійне запис в регістр, реалізація повторів читання/запису при виявленні помилок.

Тепер знаючи адреса потрібних регістрів та маючи механізм доступу можливо організувати зчитування даних.

```
float getCalibratedValue(I2C_Handle i2cBusDesciptor, uint_least8_t address, uint8_t calAddress, uint8_t device)
{
  selectDevice(i2cBusDesciptor, address, device);

  uint8_t b0, b1, b2, b3;
  b0 = virtualReadRegister(i2cBusDesciptor, address, calAddress + 0);
  b1 = virtualReadRegister(i2cBusDesciptor, address, calAddress + 1);
  b2 = virtualReadRegister(i2cBusDesciptor, address, calAddress + 2);
  b3 = virtualReadRegister(i2cBusDesciptor, address, calAddress + 3);

  //Channel calibrated values are stored big-endian
  uint32_t calBytes = 0;
  calBytes |= ((uint32_t)b0 << (8 * 3));
  calBytes |= ((uint32_t)b1 << (8 * 2));
  calBytes |= ((uint32_t)b2 << (8 * 1));
  calBytes |= ((uint32_t)b3 << (8 * 0));

  //System_printf("Debug value %d\n", calBytes);

  return (convertBytesToFloat(calBytes));
}
```

Рис. 3.2.3.6 – Читання каліброваних даних з пристрою AS7265x

3.3 Інтеграція плати AS7265x з Launchpad CC1350 по інтерфейсу UART

3.3.1 Фізичне з’єднання плат.

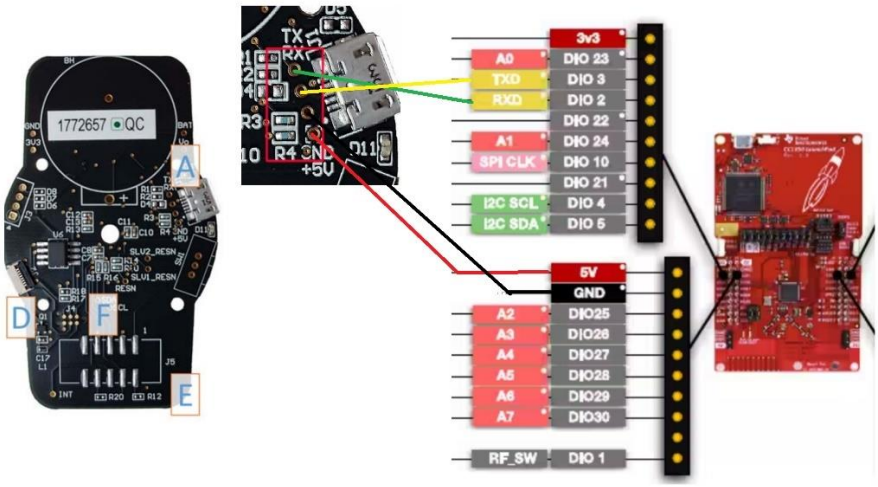


Рис. 3.3.1.1 Схема підключення по UART

Фізичне підключенняелектроживлення аналогічне I_2C але 5В замість 3.3В

Rx/Tx контакти UART ппідключаються в перехресному режимі.

3.3.2 AT команди. Спілкування по UART інтерфейсу працює використовуючи AT командиперелік яких є в офіційній документації виробника[21] с 44-47

Таблиця 3.3.1 – Список найбільш використовуваних AT командAS7265x[21], с 44-47.

AT команда	Опис команди
ATCDATA	Зчитує калібровані дані всіх датчиків по 18 каналам
ATINTTIME	Встановитичас інтеграції датчика Час інтеграції =<значення> * ~2,8 мс. Діапазон значень 1-255 або ~2,8 мс – 714мс
ATGAIN	Встановити посилення датчика: 0=1x, 1=3,7x,2=16x,3=64x
ATTEMP	Зчитує актуальну температура пристроюв градусах Цельсія
ATDATA	Зчитує дані всіх датчиків по 18 каналам

3.3.3 Реалізація в програмному забезпеченні зняття даних датчика AS7265x по інтерфейсу UART.

З UART також треба для початку створити точку доступу к інтерфейсу.

```
#include <stdint.h>
#include <stddef.h>

/* Driver Header files */
#include <ti/drivers/GPIO.h>
#include <ti/drivers/UART.h>

/* Example/Board Header files */
#include "Board.h"


void init_uart()
{
    UART_Handle uart;
    UART_Params uartParams;

    /* Call driver init functions */
    GPIO_init();
    UART_init();

    UART_Params_init(&uartParams);
    uartParams.baudRate = 115200;

    uart = UART_open(Board_UART0, &uartParams);


}
```

Рис. 3.3.3.1 точка доступу доUART

Після цього можливо обмінюватися даними з датчиком по інтерфейсу UART. Треба лише записати команду в буфер та створити буфер для прийому даних. Запуск обміну даними виглядатиме як показано нижче.

```
        UART_read(uart, &input, input.size());
        UART_write(uart, &output, 72);
```

Рис. 3.3.3.2 обмін даними поUART

3.4 Пересилка даних датчика AS7265x по бездротовій мережі теплиці.

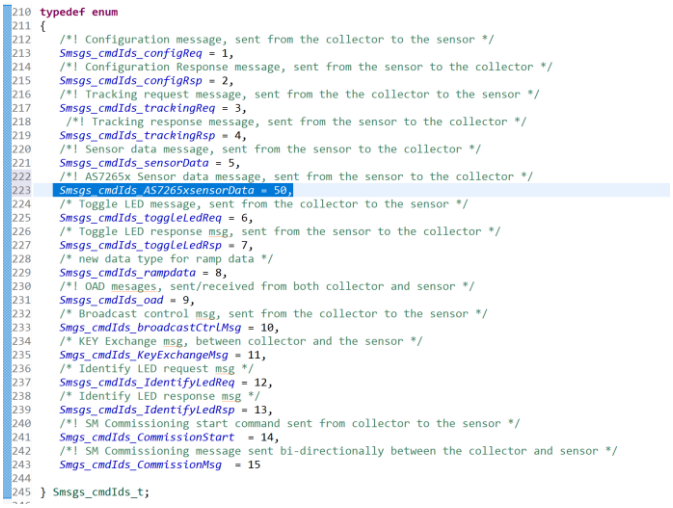
3.4.1 Пакет даних датчика AS7265x. Використаєм існуючий пакет типу 5 Smsgs_cmdIds_sensorData як базу для розробки нового пакета. Нехай це буде пакет типу 50 = Smsgs_cmdIds_AS7265xSensorData За для реалізації цієї задачі потребується реалізація нового пакету даних.

Таблиця 3.4.1.1 – структура пакету даних датчика AS7265x

Назва поля повідомлення	Опис	Довжина
Команда	Тип повідомлення, Smsgs_cmdIds_ AS7265xSensorData= 50	1 байт
Адреса відправника	Розширена Адреса	8 байт
Calibrated410	Відкалібровані дані каналу 410нМ	4 байт
Calibrated435	Відкалібровані дані каналу 435нМ	4 байт
Calibrated460	Відкалібровані дані каналу 460нМ	4 байт
Calibrated485	Відкалібровані дані каналу 485нМ	4 байт
Calibrated510	Відкалібровані дані каналу 510нМ	4 байт
Calibrated535	Відкалібровані дані каналу 535нМ	4 байт
Calibrated560	Відкалібровані дані каналу 560нМ	4 байт
Calibrated585	Відкалібровані дані каналу 585нМ	4 байт
Calibrated610	Відкалібровані дані каналу 610нМ	4 байт
Calibrated645	Відкалібровані дані каналу 645нМ	4 байт
Calibrated680	Відкалібровані дані каналу 680нМ	4 байт
Calibrated705	Відкалібровані дані каналу 405нМ	4 байт
Calibrated730	Відкалібровані дані каналу 730нМ	4 байт
Calibrated760	Відкалібровані дані каналу 760нМ	4 байт
Calibrated810	Відкалібровані дані каналу 810нМ	4 байт
Calibrated860	Відкалібровані дані каналу 860нМ	4 байт
Calibrated900	Відкалібровані дані каналу 900нМ	4 байт
Calibrated940	Відкалібровані дані каналу 940нМ	4 байт

3.4.2 Програмна реалізація обміну даними датчикаAS7265x

На обох точках мережі(концентратор та сенсор) треба додати підтримку нового формату даних.

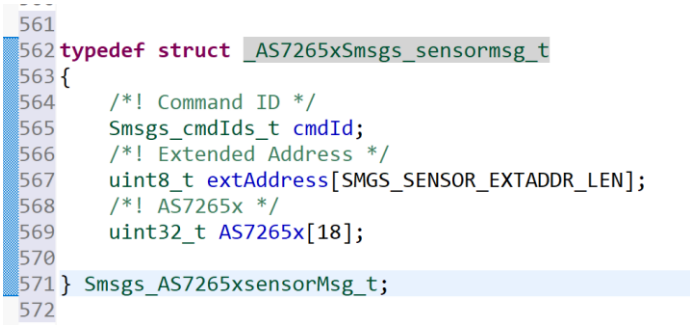


```
typedef enum
{
  /*! Configuration message, sent from the collector to the sensor */
  Smsgs_cmdIds_configReq = 1,
  /*! Configuration Response message, sent from the sensor to the collector */
  Smsgs_cmdIds_configRsp = 2,
  /*! Tracking request message, sent from the the collector to the sensor */
  Smsgs_cmdIds_trackingReq = 3,
   /*! Tracking response message, sent from the sensor to the collector */
  Smsgs_cmdIds_trackingRsp = 4,
  /*! Sensor data message, sent from the sensor to the collector */
  Smsgs_cmdIds_sensorData = 5,
  /*! AS7265x Sensor data message, sent from the sensor to the collector */
  Smsgs_cmdIds_AS7265xsensorData = 50
  /* Toggle LED message, sent from the collector to the sensor */
  Smsgs_cmdIds_toggleLedReq = 6,
  /* Toggle LED response msg, sent from the sensor to the collector */
  Smsgs_cmdIds_toggleLedRsp = 7,
  /* new data type for ramp data */
  Smsgs_cmdIds_rampdata = 8,
  /*! OAD mesages, sent/received from both collector and sensor */
  Smsgs_cmdIds_oad = 9,
  /* Broadcast control msg, sent from the collector to the sensor */
  Smgs_cmdIds_broadcastCtrlMsg = 10,
  /* KEY Exchange msg, between collector and the sensor */
  Smgs_cmdIds_KeyExchangeMsg = 11,
  /* Identify LED request msg */
  Smsgs_cmdIds_IdentifyLedReq = 12,
  /* Identify LED response msg */
  Smsgs_cmdIds_IdentifyLedRsp = 13,
  /*! SM Commissioning start command sent from collector to the sensor */
  Smgs_cmdIds_CommissionStart  = 14,
  /*! SM Commissioning message sent bi-directionally between the collector and sensor */
  Smgs_cmdIds_CommissionMsg  = 15

} Smsgs_cmdIds_t;
```

Рис. 3.4.2.1 декларація нового пакету даних датчика AS7265x на концентраторі та кінцевої точки мережі.

Також задаємо структуру нового пакету



```
561
562 typedef struct _AS7265xSmsgs_sensormsg_t
563 {
564     /*! Command ID */
565     Smsgs_cmdIds_t cmdId;
566     /*! Extended Address */
567     uint8_t extAddress[SMGS_SENSOR_EXTADDR_LEN];
568     /*! AS7265x */
569     uint32_t AS7265x[18];
570
571 } Smsgs_AS7265xsensorMsg_t;
572
```

Рис. 3.4.2.2 структура пакету даних датчика AS7265x

Обробка нового пакету даних на точцікінцевої точки мережі. В момент запуску збору даних треба визвати процедуру запиту даних AS7265x та відіслати на концентратор:

```
static bool sendAS7265xSensorMessage(ApiMac_sAddr_t *pDstAddr, Smsgs_broadcastcmdmsg_t *pMsg)
{
    bool ret = false;
    uint8_t *pMsgBuf;
    //len = 18 * 4 + 8 + 1 = 81
    uint16_t len = 81;

    pMsgBuf = (uint8_t *)Ssf_malloc(len);
    if(pMsgBuf)
    {
        uint8_t *pBuf = pMsgBuf;

        *pBuf++ = (uint8_t)Smsgs_cmdIds_AS7265xsensorData;

        memcpy(pBuf, pMsg->extAddress, SMGS_SENSOR_EXTADDR_LEN);
        pBuf += SMGS_SENSOR_EXTADDR_LEN;
        memcpy(pBuf, pMsg->AS7265x, 72);

        ret = Sensor_sendMsg(Smsgs_cmdIds_AS7265xsensorData, pDstAddr, true, len, pMsgBuf);
        Ssf_free(pMsgBuf);
    }

    return (ret);
}
```

Рис. 3.4.2.3 - Відсилка пакету даних датчика AS7265x з кінцевої точки мережі.

На концентраторі треба обробити новий пакет даних

```
844         switch(cmdId)
845         {
846             case Smsgs_cmdIds_configRsp:
847                 processConfigResponse(pDataInd);
848                 break;
849
850             case Smsgs_cmdIds_trackingRsp:
851                 processTrackingResponse(pDataInd);
852                 break;
853
854             case Smsgs_cmdIds_IdentifyLedReq:
855                 processIdendifyLedRequest(pDataInd);
856                 break;
857
858             case Smsgs_cmdIds_toggleLedRsp:
859                 processToggleLedResponse(pDataInd);
860                 break;
861
862             case Smsgs_cmdIds_sensorData:
863                 processSensorData(pDataInd);
864                 break;
865             case Smsgs_cmdIds_AS7265xsensorData:
866                             processAS7265xSensorData(pDataInd);
867                             break;
```

Рис. 3.4.2.4 обробка пакету даних датчика AS7265x на концентраторі.

Підсумки розділу.

В цьому розділі було розглянуто можливості датчику AS7265x та показано як за його допомогою можна получити лічильник фотонів якими опромінюються рослини закритого ґрунту. Також розглянуто інтерфейс спілкування по UART та I2C, що реалізує інтеграцію даних датчиків в бездротову мережу теплиці.

РОЗРОБКА АЛГОРИТМІВ І ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ АНАЛІЗУ ТА ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ ДЛЯ ПІДВИЩЕННЯ ЕНЕРГОЕФЕКТИВНОСТІ ПРОЦЕСУ ОПРОМІНЕННЯ РОСЛИН ФІЗІОЛОГІЧНО АКТИВНОЮ РАДІАЦІЄЮ

4.1 Схема взаємодії бездротової мережі теплиці з корпоративною мережею підприємства.

В розділі два, під час проектування бездротової мережі теплиці, вже була передбачена схема інтеграції мережі в систему автоматичного керування виробничими процесами на підприємстві на базі NPI server (Рис. 4.1.1). Розглянемо цю схему більш детально та продовжимо її проектування.

Для реалізації системи аналізу та підтримки прийняття рішень енергоефективності процесу опромінення рослин фізіологічно активною радіацією пропонується розробка трьох сервісів:

- OPC UA server
- Database server
- Web server

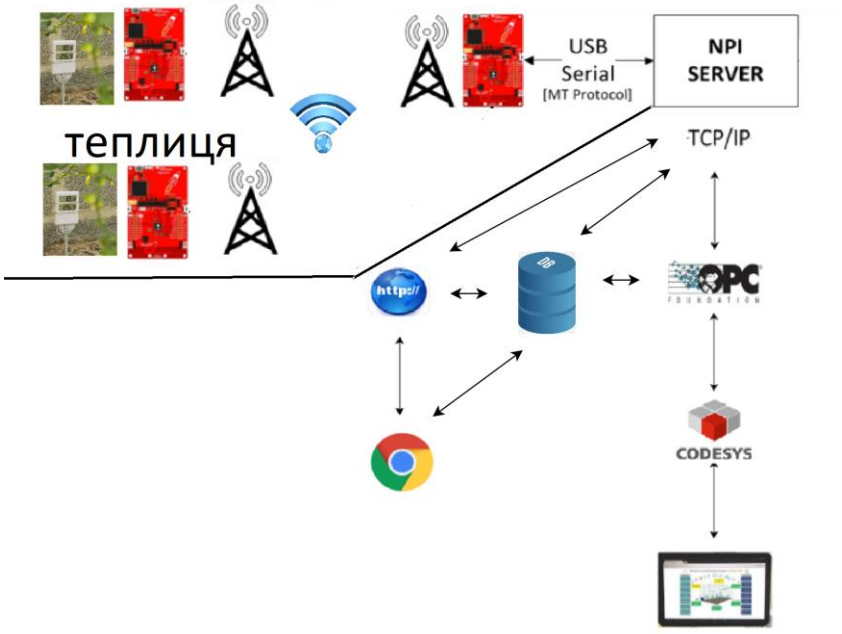


Рис. 4.1.1 – схема взаємодії бездротової мережі теплиці з корпоративною мережею підприємства.

NPI сервер виконую виконує частину задач концентратора, та є шлюзом в бездротову мережу теплиці. Всі пакети отримані концентратором, попадають на NPI сервер через USB, там вони обгортаються в TCP/IP заголовки та направляються в мережевий інтерфейс який з'єднаний з корпоративною мережею підприємства. Також система працює в зворотному напрямку. TCP/IP пакети корпоративної мережі, адресовані в бездротову мережу прибувають на мережевий інтерфейс NPI сервера, по USB потрапляють на плату CC1550, та відправляються на потрібний датчик.

4.2 Інтеграція бездротової мережі теплеці в систему автоматичного керування на базі OPC UA та CODESYS PLC

4.2.1 Програмна реалізація OPC UA server

Для розробки обрано мову програмування Python та програмна бібліотека [22], яка реалізує рівень додатків з протоколів верхнього рівня відкритої архітектури OPC UA. Для встановлення бібліотеки скористуємося командою:

pipinstall opcua

Далі розглянемо покроково реалізований програмний модуль OPC UA server.

4.2.1.1 Імпорт бібліотек та вхідні дані(Рис. 4.2.1.1)

```
# Імпорт необхідних бібліотек
from opcua import Server
import socket


# Вхідні дані
OPC_SERVER_URL = "opc.tcp://0.0.0.0:12345"
OPC_SERVER_NAMESPACE = "greenhouse"
NPI_GATEWAY_SERVER_ADDRESS = "192.168.0.168"
NPI_GATEWAY_SERVER_PORT = 5000

```

Рис. 4.2.1.1 - Імпорт бібліотек та вхідні дані

Для базової реалізації OPC server занадобиться всього дві бібліотеки.

- opcua – бібліотека по роботі в мережі на рівні додатків архітектури OPC UA.
- socket - стандартна бібліотека по роботі в мережі на транспортному рівні

Необхідно задати URL по якому буде доступний OPC UA server та адресу і порт на якому працює NPI server.

4.2.1.2 Ініціалізація OPC UA Server(Рис. 4.2.1.2)

```
# Ініціалізвція OPC Server
server = Server()
server.set_endpoint(OPC_SERVER_URL)
name_space_id = server.register_namespace(OPC_SERVER_NAMESPACE)

```

Рис. 4.2.1.2 - Ініціалізація OPC UA Server

Створюємо екземпляр OPC UA Server, прив’язуємо його до адреси, та задаємо іменовану групу змінних OPC

4.2.1.3 Ініціалізація OPC змінних (Рис. 4.2.1.3)

```
# Ініціалізація даних датчиків
# Датчик №1
objects = server.get_objects_node()
temperature_sensor = objects.add_object('ns=%s;s="TS1"' % name_space_id, "Temperature sensor 1")
temp = temperature_sensor.add_variable('ns=%s;s="TS1_Temperature"' % name_space_id, "Temperature value", 0)
led = temperature_sensor.add_variable('ns=%s;s="TS1_Led"' % name_space_id, "Temperature sensor led", True)
led.set_writable()

# Датчик №2
temperature_sensor2 = objects.add_object('ns=%s;s="TS2"' % name_space_id, "Temperature sensor 2")
temp2 = temperature_sensor2.add_variable('ns=%s;s="TS2_Temperature"' % name_space_id, "Temperature value", 0)
led2 = temperature_sensor2.add_variable('ns=%s;s="TS2_Led"' % name_space_id, "Temperature sensor led", True)
led2.set_writable()

# Датчик №3
temperature_sensor3 = objects.add_object('ns=%s;s="TS3"' % name_space_id, "Temperature sensor 3")
temp3 = temperature_sensor3.add_variable('ns=%s;s="TS3_Temperature"' % name_space_id, "Temperature value", 0)
led3 = temperature_sensor3.add_variable('ns=%s;s="TS3_Led"' % name_space_id, "Temperature sensor led", True)
led3.set_writable()
```

Рис. 4.2.1.3 - Ініціалізація OPC змінних

Добавляємо к server датчики за якими планується організувати моніторинг. Кожен датчик може видавати декілька значень параметрів мікроклімату, прив’язуємо ці дані до змінних OPC.

4.2.1.4 Реєстрація датчиків в системі (Рис. 4.2.1.4)

```
# Реєстрація датчиків в системі
devices = {'13480113190751 80': temp,
           '011811719075180': temp2,
           '133151112190751 80': temp3}

# Реєстрація коротких адресів 802.14.5
short_addresses = {}
```

Рис. 4.2.1.4 - Ініціалізація OPC змінних

У кожного датчика є свій унікальний MAC адрес прив’язуємо кожен датчик до OPC змінних. У мережах 802.15.4 після успішного з’єднання концентратор та кінцева точка мережі домовляються про тимчасовий короткий адрес кінцевої точки, створюємо змінну для зберігання цих коротких адресів.

4.2.1.5 Запускаємо OPC UA server(Рис 4.2.1.5)

```
try:
    # Старт OPC server
    print("starting OPC server")
    server.start()
    print("OPC server is online")
```

Рис 4.2.1.5 Запуск OPC UA server

4.2.1.6 Обробка вхідних повідомлень в циклі (Рис 4.2.1.6)

```
    # Обробка вхідних повідомлень
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((NPI_GATEWAY_SERVER_ADDRESS, NPI_GATEWAY_SERVER_PORT))
        while True:
            data = s.recv(1024)
            # Отриман пакет з NPI шлюзу, друкуємо вміст пакету
            print(f"Received package:{data!r}")
            package = bytearray(data)
            # друкуємо довжину пакету
            print("package length:%s" % len(package))
            # Перевірка типу пакту, потрібні лише дані датчиків
            if len(package) >= 88 and package[8] == 5:
                # Зчитуємо адресу датчика
                sensor_address = "%s%s%s%s%s%s%s%s" % (package[9], package[10], package[11], package[12],
                                                       package[13], package[14], package[15], package[16])
                sensor = devices.get(sensor_address)
                # Перівірка чи відомий в системі датчик
                if sensor is not None:
                    # Обновляємо зміні OPC даними датчіків
                    sensor.set_value(package[19])
                    short_address = short_addresses.get(sensor_address)
                    if short_address is not None or short_address != package[5]:
                        short_addresses[sensor_address] = package[5]
                else:
                    # друкуємо помилку відсутності регістрації
                    print("Not registered sensor: %s" % sensor_address)

            # Друкуємо змінні OPC
            print(str(temp.get_value()))
            print(str(temp2.get_value()))
            print(str(temp3.get_value()))
```

Рис 4.2.1.6 - Обробка вхідних повідомлень в циклі

Знаючи формат повідомлень в мережі в розділі два та три, можна організувати їх обробку для подальшої пересилки в форматі OPC UA. NPI server лише добавляє свої заголовки чи суфікси (тому довжина пакета дещо більша), але не змінює внутрішній состав повідомлення в бездротовій системі. Нас цікавить повідомлення з типом 5 – дані сенсору, получаємо необхідні дані з відповідних полів пакету та проставляємо їх в змінні OPC.

4.2.1.7 Завершення роботиOPC UA server(Рис 4.2.1.7)

```
83      finally:
84          # Завершення роботи OPC SERVER
85          server.stop()
86
```

Рис 4.2.1.7 Завершення роботи OPC UA server

4.2.2 Обмін даними між OPC Server та CoDeSys PLC.

4.2.2.1 По-перше інсталюємо PLC (Windows чи Linux, Beagle Bone Black, raspberry pi, або любий інший) та з'єднаємся з ним в CoDeSys IDE. В роботі використовувалась версія 3.5, Service Pack 19, Patch 2

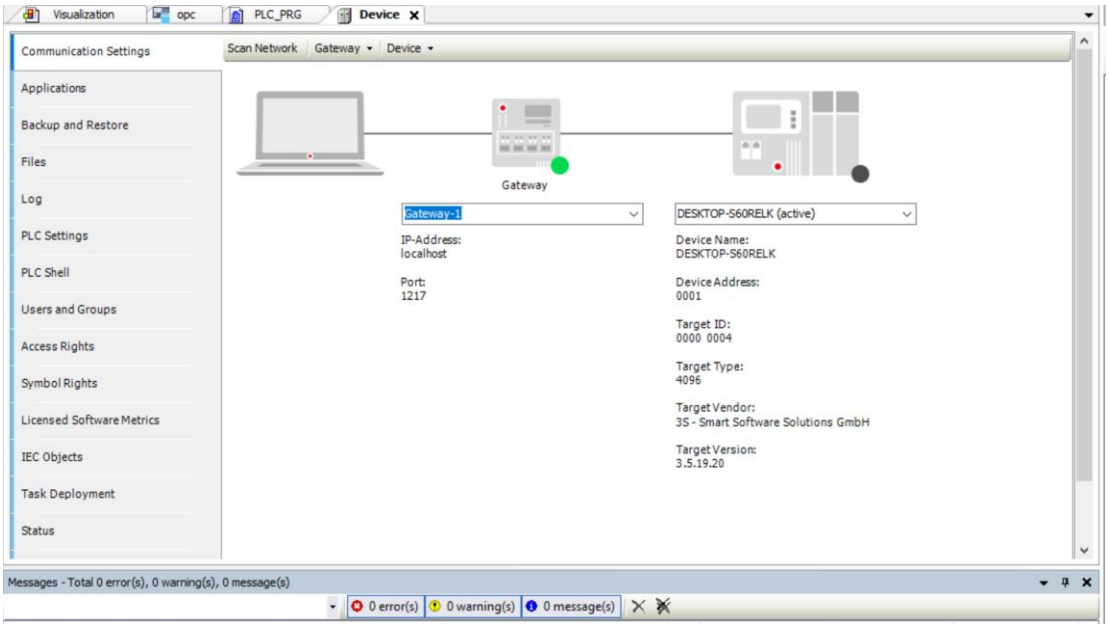


Рис 4.2.2.1 Windows CoDeSys PLC

4.2.2.2 Далі додаємо Data Source Manager, якщо його не має. Потім й сам Data Source, у контекстному списку типу ресурсу обираємо OPC UA server(Рис. 4.2.2.2)

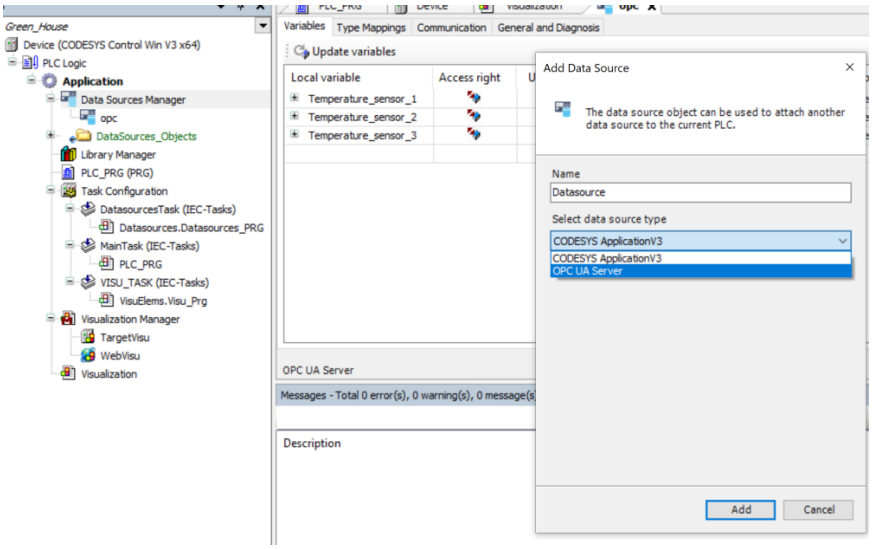


Рис 4.2.2.2 Створення OPC UA Server Data Source

4.2.2.3 Задаємо адресу OPC UA Server саме такий як був при створені серверу(см. Рис. 4.2.1.1), треба скопіювати константу OPC_SERVER_URL

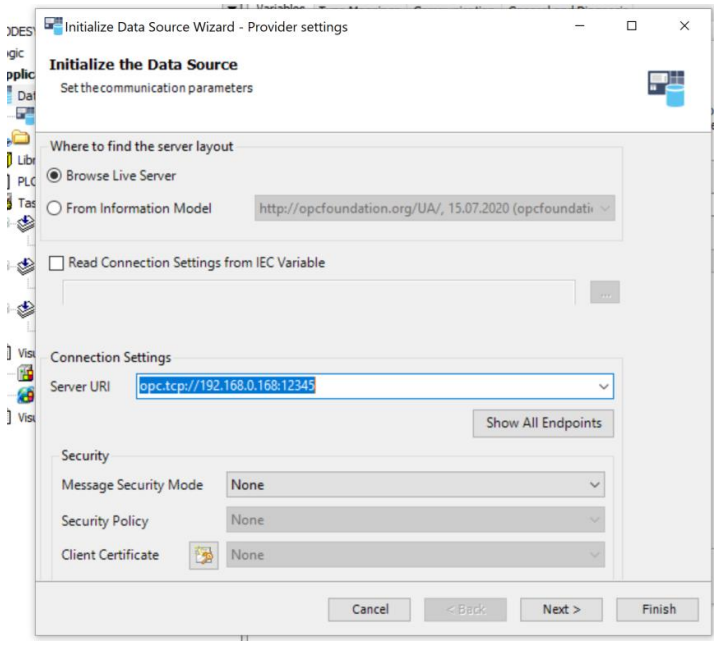


Рис 4.2.2.3 Конфігурація адресу OPC UA server

4.2.2.4 Тиснемо кнопку Finish та чекаємо створення з’єднання. Якщо з’єднання пройде без помилок, то буде можливо імпортувати змінні створені в OPC UA Server(см. Рис. 4.2.1.3)

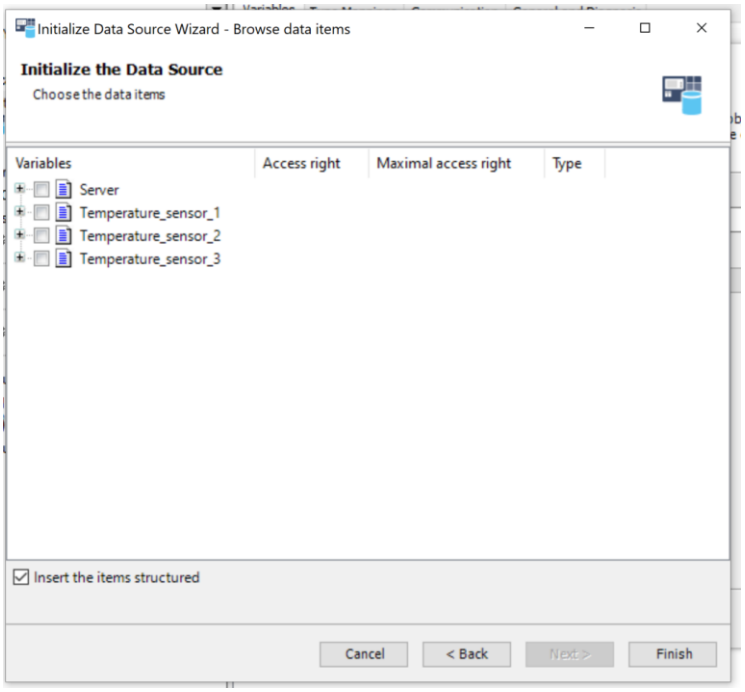


Рис. 4.2.2.4 - Імпорт управляючих змінних OPC

4.2.2.5 Створюємо віртуалізацію програмний контролеру, додаємо необхідні інструменти на панель візуалізації, пов’язуємо інструменти з змінними OPC.

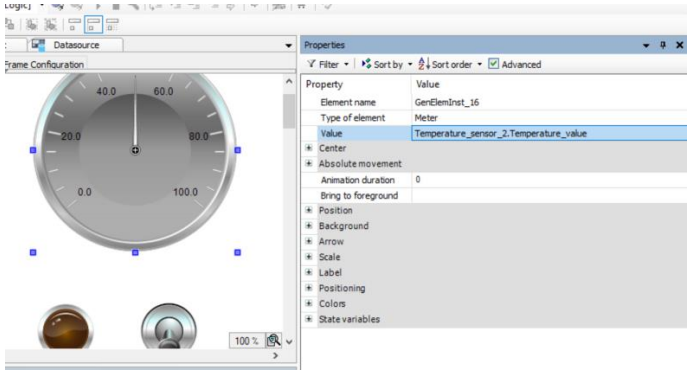


Рис - 4.2.2.5 Настройка інструментів візуалізації CoDeSys PLC

4.2.2.6 Завантажуємо зміну програмного коду в програмний контролер, та запускаємо його на виконування

Рис. 4.2.2.6 – Робота візуалізації CoDeSys PLC

Таким же чином в подальшому будуть імпортовані дані спектральних датчиків.

Цей приклад описує лише віртуалізацію, але таким же чином можливо запрограмувати не лише моніторинг параметрів мікроклімату з бездротової мережі теплиці а й створити повноцінний контролер, який вже буде регулювати системами контролю мікроклімату в теплиці.

Висновки розділу.

В виду браку часу не вдалося пропрацювати весь матеріал запланований на цей розділ. Але були зроблені базові речі, які дозволяють продовжити роботу поза межами цієї магістерської роботи. Була розроблена схема інтеграції обчислювальних можливостей бездротової мережі контролю параметрів мікроклімату в корпоративну мережу підприємства. На базі цієї інтеграції стало можливим обробка даних в системі розробки контролерів CoDeSys та інтеграція з SCADA системою. В майбутньому передбачена можлива повноцінна реалізація контролера(ів) для реалізації вже не тільки систем моніторингу але й повного циклу автоматичного керування. Система обміну даних з бездротовою мережею теплиці також дає можливість продовжити розробку бази даних системи прийняття рішень, та різних інтерфейсів користувача.

ВИСНОВКИ

В результаті виконання роботи було проаналізовано питанням підвищення врожайності культурних рослин в культиваційних спорудах закритого ґрунту та зменшення енерговитрат шляхом підвищення ефективності моніторингу в теплиці, як в об’єкті керування з розподіленими параметрами мікроклімату.

Були проаналізовані основні параметри мікроклімату в теплицях, які підлягають контролю. Розглянуті недоліки існуючих підходів організації моніторингу цих параметрів, особлива увага була переділена недолікам контроля опромінення рослин фізіологічно активною радіацією.

На базі цих недоліків було запропоноване ефективне рішення – створення бездротової мережі контролю за допомогою дводіапазонного радіо модулю CC1350. Спроектовані сегменти та топологія мережі, ретельно підібрані мережеві протоколи кожного рівня. На базі шаблонного рішення спроектовані алгоритм та протокол роботи мережі на рівні додатків. Розпочато розробку програмного забезпечення.

Задля подолання проблем відсутності якісного обладнання спектрального аналізу фізіологічно активної радіації та вимірювання потужності світлового опромінення для широкого промислового використання було розглянуто використання технічної новинки - недисперсійного спектрометра AS7265x в якості лічильника фотонів в діапазоні 385нм – 940 нм.

Реалізована інтеграція обчислюваних можливостей бездротової мережі контролю параметрів мікроклімату з SCADA системою за допомогою архітектури OPC UA.

Нажаль, вдалося завершити все заплановане. В виду браку часу була лише почато розробка програмного забезпечення всіх розділив цієї роботи, яке потребує ще багато циклів розробки для доведення до стану готового для практичного застосування на підприємстві. Система підтримки прийняття рішень також потребує допрацювання.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. TIBBITTS, T. W.; MORGAN, D. C.; WARRINGTON, I. J. Growth oflettuce, spinach, mustard, andwheatplantsunderfourcombinationsofhigh-pressuresodium, metalhalide, andtungstenhalogenlampsatequal PPFD. Journal ofthe American Society forHorticulturalScience, 1983, 108.4: 622-630.
2. CHAZDON, Robin L.; FETCHER, Ned. Light environmentsoftropicalforests. In: PhysiologicalecologyofplantsofthewetTropics: ProceedingsofaninternationalsymposiumheldinOxatepecand Los Tuxtlas, Mexico, June 29 toJuly 6, 1983. Springer Netherlands, 1984. p. 27-36.
3. К. Бекетт Рослини під склом Вибір обладнання Ведення тепличного господарства Вирощування рослин в теплицях 1988 198 с.
4. [Електрон. ресурс] wikiInternetofthingsURL: https://en.wikipedia.org/wiki/Internet_of_things
5. [Електрон. ресурс]helpnetsecurityInternetofthingsURLhttps://www.helpnetsecurity.com/2019/05/23/connected-devices-growth/
6. Parziale L. etal. TCP/IP tutorialandtechnicaloverview. – 2006
7. Hunt C. TCP/IP networkadministration. – " O'ReillyMedia, Inc.", 200
8. Ergen S. C. ZigBee/IEEE 802.15. 4 Summary //UC Berkeley, September. – 2004. – Т. 10. – №. 17
9. Yang D., Xu Y., Gidlund M. Wireless coexistencebetween IEEE 802.11-and IEEE 802.15. 4-based networks: A survey //International Journal ofDistributedSensor Networks. – 2011
10. Zheng J., Lee M. J. A comprehensive performance study of IEEE 802.15. 4 //Sensornetworkoperations. – 2006
11. Olsson J. 6LoWPAN demystified //Texas Instruments. – 2014
12. Lindh J. etal. Measuring CC13xx and CC26xx currentconsumption //Texas Instrument, Application Report. – 2019.

13. [Електрон. ресурс]CC1350 SimpleLink™ Ultra-Low-Power Dual-Band Wireless MCU datasheet. URL: https://www.ti.com/lit/ds/symlink/cc1350.pdf

14. [Електрон. ресурс]CC13x0, CC26x0 SimpleLink™ Wireless MCU TechnicalReferenceManual. URL: https://www.ti.com/lit/ug/swcu117i/swcu117i.pdf

15. [Електрон. ресурс]TI 15.4-Stack Linux Developer's Guide. URL: https://software-dl.ti.com/lprf/ti15.4stack_linux_x64/2.01.00.10/exports/docs/ti154stack-ldg/ti154stack-ldg/index.html

16. SCHWARZ, Michael H.; BÖRCSÖK, Josef. A surveyon OPC and OPC-UA: Aboutthestandard, developmentsandinvestigations. In: *2013 XXIV International Conference on Information, Communication andAutomation Technologies (ICAT)*. IEEE, 2013. p. 1-6.

17. VEICHTLBAUER, Armin; ORTMAYER, Martin; HEISTRACHER, Thomas. OPC UA integrationforfielddevices. In: 2017 IEEE 15th International Conference on Industrial Informatics (INDIN). IEEE, 2017. p. 419-424.

18. [Електрон. ресурс]Інсталяція CSS IDE на сайті Tehas Instruments.URL: https://www.ti.com/tool/CCSTUDIO#downloads

19. [Електрон. ресурс]Інсталяція 15.4 Staklinuxна сайті Tehas InstrumentsURLhttps://www.ti.com/tool/TI-15.4-STACK-GATEWAY-LINUX-SDK

20. [Електрон. ресурс]SparkfunAS7265xdatasheet URL: https://cdn.sparkfun.com/assets/c/2/9/0/a/AS7265x_Datasheet.pdf

21. [Електрон. ресурс]AMS-OSRAM AS7265x datasheethttps://ams.com/documents/20143/36005/AS7265x_DS000612_1-00.pdf

22. [Електрон. ресурс] OPC UA protocol URL https://github.com/FreeOpcUa/python-opcua

Додаток 1. Лістинг програми Sensor, C++ . Частина по роботі з регістрами I_2C

```
#include <stdint.h>
#include <stddef.h>
#include <unistd.h>

/* Імпорт драйверів плати CC1350 */
#include <ti/drivers/GPIO.h>
#include <ti/drivers/I2C.h>
#include <ti/display/Display.h>
#include <xdc/runtime/System.h>

#include "Board.h"

#define TASKSTACKSIZE       640

/*
 *  Декларація констант та адрес регістрів
 */
#define TMP006_REG          0x0001
#define TMP116_REG          0x0000

#define TMP006_ADDR         0x41;
#define TMP116_BP_ADDR      0x48;
#define TMP116_LP_ADDR      0x49;

#define AS7265X_HW_VERSION_HIGH 0x00
#define AS7265X_HW_VERSION_LOW 0x01


#define AS72651_NIR 0x00
#define AS72652_VISIBLE 0x01
#define AS72653_UV 0x02

#define AS7265X_STATUS_REG 0x00
#define AS7265X_WRITE_REG 0X01
#define AS7265X_RX_VALID 0x01
#define AS7265X_TX_VALID 0x02
#define AS7265X_READ_REG 0x02

#define AS7265X_R_G_A_CAL 0x14
#define AS7265X_S_H_B_CAL 0x18
#define AS7265X_T_I_C_CAL 0x1C
#define AS7265X_U_J_D_CAL 0x20
#define AS7265X_V_K_E_CAL 0x24
#define AS7265X_W_L_F_CAL 0x28

#define AS7265X_LED_CURRENT_LIMIT_12_5MA 0b00
#define AS7265X_LED_CURRENT_LIMIT_25MA 0b01
#define AS7265X_LED_CURRENT_LIMIT_50MA 0b10
#define AS7265X_LED_CURRENT_LIMIT_100MA 0b11

#define AS7265x_LED_WHITE 0x00
#define AS7265x_LED_IR 0x01
#define AS7265x_LED_UV 0x02
```

```
#define AS7265X_CONFIG 0x04
#define AS7265X_INTERGRATION_TIME 0x05
#define AS7265X_MEASUREMENT_MODE_6CHAN_ONE_SHOT 0b11

#define AS7265X_DEV_SELECT_CONTROL 0x4F
#define AS7265X_DEVICE_TEMP 0x06
#define AS7265X_LED_CONFIG 0x07

#define AS7265X_POLLING_DELAY 0.1

static Display_Handle display;
static uint8_t maxWaitTime  = 10000;
static uint8_t maxAttempts  = 30;


/* Читання даних до регістра */
uint8_t readRegister(I2C_Handle i2cBusDesciptor, uint_least8_t address, uint8_t
registerName) {
    uint8_t         txBuffer[1];
    uint8_t         rxBuffer[1];
    I2C_Transaction i2cTransaction;
    i2cTransaction.writeBuf   = txBuffer;
    i2cTransaction.writeCount = 1;
    i2cTransaction.readBuf    = rxBuffer;
    i2cTransaction.readCount  = 1;
    i2cTransaction.slaveAddress = address;
    txBuffer[0] = registerName;

    if (!I2C_transfer(i2cBusDesciptor, &i2cTransaction)) {
        Display_printf(display, 0, 0, "Read register problem");
    }

    return rxBuffer[0];
}
/* Запис даних до регістру */
uint8_t writeRegister(I2C_Handle i2cBusDesciptor, uint_least8_t address, uint8_t
registerName, uint8_t registerValue) {
    uint8_t         txBuffer[2];
    I2C_Transaction i2cTransaction;
    i2cTransaction.writeBuf   = txBuffer;
    i2cTransaction.writeCount = 2;
    i2cTransaction.readBuf    = NULL;
    i2cTransaction.readCount  = 0;
    i2cTransaction.slaveAddress = address;
    txBuffer[0] = registerName;
    txBuffer[1] = registerValue;

    if (!I2C_transfer(i2cBusDesciptor, &i2cTransaction)) {
        Display_printf(display, 0, 0, "Write register problem");
        return false;
    }

    return true;
}

unsigned long getTime() {
    time_t curtime;
    time(&curtime);
    return ctime(&curtime);
}
```

```
/* Читання даних до регістра з коректуванням помилок */
uint8_t virtualReadRegister(I2C_Handle i2cBusDesciptor, uint_least8_t address,
uint8_t virtualAddr)
{


  uint8_t status;


  status = readRegister(i2cBusDesciptor, address, AS7265X_STATUS_REG);
  if ((status & AS7265X_RX_VALID) != 0)
  {
    readRegister(i2cBusDesciptor, address, AS7265X_READ_REG);
  }

  uint8_t attempt = 1;

  /* Цикл підключення по I2C */
  while (1)
  {
    if(attempt > maxAttempts) {
        Display_printf(display, 0, 0, "virtualReadRegister failed %d", virtualAddr);
        return(0); // Не змогли приєднатися
        }
    status = readRegister(i2cBusDesciptor, address, AS7265X_STATUS_REG);
    if ((status & AS7265X_TX_VALID) == 0)
      break; // приєдналися
    sleep(AS7265X_POLLING_DELAY);
    attempt++;
  }

  writeRegister(i2cBusDesciptor, address, AS7265X_WRITE_REG, virtualAddr);

  attempt = 1;

  while (1)
  {
    if(attempt > maxAttempts)  {
        Display_printf(display, 0, 0, "virtualReadRegister 2 failed %d",
virtualAddr);
        return(0); // Не змогли приєднатися
    }
    status = readRegister(i2cBusDesciptor, address, AS7265X_STATUS_REG);
    if ((status & AS7265X_RX_VALID) != 0)
      break; // приєдналися
    sleep(AS7265X_POLLING_DELAY);
    attempt++;
  }

  uint8_t incoming = readRegister(i2cBusDesciptor, address, AS7265X_READ_REG);
  return (incoming);
}

/* Запис даних до регістру з коректуванням помилок */
void virtualWriteRegister(I2C_Handle i2cBusDesciptor, uint_least8_t address, uint8_t
virtualAddr, uint8_t dataToWrite)
{
  uint8_t status;
  uint8_t attempt = 1;
```

```
  while (1)
  {
    if(attempt > maxAttempts) {
        unsigned long startTime1 = getTime();
        Display_printf(display, 0, 0, "virtualWriteRegister problem %d value %d",
virtualAddr);
        return; // датчик не відповідає

    }
    status = readRegister(i2cBusDesciptor, address, AS7265X_STATUS_REG);
    if ((status & AS7265X_TX_VALID) == 0)
      break;
    sleep(AS7265X_POLLING_DELAY);
    attempt++;
  }

  writeRegister(i2cBusDesciptor, address, AS7265X_WRITE_REG, (virtualAddr | 1 << 7));

/* Цикл підключення по к I2C slave датчики  */
  attempt = 1;
  while (1)
  {
    if(attempt > maxAttempts) {
        Display_printf(display, 0, 0, "virtualWriteRegister problem 2 %d value %d,
%d", virtualAddr, virtualAddr, attempt);
        return; // датчик не відповідає
    }
    status = readRegister(i2cBusDesciptor, address, AS7265X_STATUS_REG);
    if ((status & AS7265X_TX_VALID) == 0)
      break;
    sleep(AS7265X_POLLING_DELAY);
    attempt++;
  }
  writeRegister(i2cBusDesciptor, address, AS7265X_WRITE_REG, dataToWrite);

}

/* Перемикання датчика, є три датчика на платі */
void selectDevice(I2C_Handle i2cBusDesciptor, uint_least8_t address, uint8_t
deviceNumber) {
    virtualWriteRegister(i2cBusDesciptor, address, AS7265X_DEV_SELECT_CONTROL,
deviceNumber);


uint8_t getHardwareVersion(I2C_Handle i2cBusDesciptor, uint_least8_t address) {
    return virtualReadRegister(i2cBusDesciptor, address, AS7265X_HW_VERSION_LOW);
}
// датчик не відповідає
void setBulbCurrent(I2C_Handle i2cBusDesciptor, uint_least8_t address, uint8_t
current, uint8_t device)
{
  selectDevice(i2cBusDesciptor, address, device);

  if (current > 0b11)
    current = 0b11;
  uint8_t value = virtualReadRegister(i2cBusDesciptor, address, AS7265X_LED_CONFIG);
  value &= 0b11001111;
  value |= (current << 4);
  virtualWriteRegister(i2cBusDesciptor, address, AS7265X_LED_CONFIG, value);
}
```

```
/* Включити LED на борту датчика */
void enableBulb(I2C_Handle i2cBusDesciptor, uint_least8_t address, uint8_t device)
{
  selectDevice(i2cBusDesciptor, address, device);


  uint8_t value = virtualReadRegister(i2cBusDesciptor, address, AS7265X_LED_CONFIG);
  value |= (1 << 3);
  virtualWriteRegister(i2cBusDesciptor, address, AS7265X_LED_CONFIG, value);
}

/* Виключити LED на борту датчика */
void disableBulb(I2C_Handle i2cBusDesciptor, uint_least8_t address, uint8_t device)
{
  selectDevice(i2cBusDesciptor, address, device);

  uint8_t value = virtualReadRegister(i2cBusDesciptor, address, AS7265X_LED_CONFIG);
  value &= ~(1 << 3);
  virtualWriteRegister(i2cBusDesciptor, address, AS7265X_LED_CONFIG, value);
}

/* Зчитати дані темометру на борту датчика */
uint8_t getTemperature(I2C_Handle i2cBusDesciptor, uint_least8_t address, uint8_t
deviceNumber)
{
  selectDevice(i2cBusDesciptor, address, deviceNumber);
  return (virtualReadRegister(i2cBusDesciptor, address, AS7265X_DEVICE_TEMP));
}

/* Задати конфігурацію часу інтегрування */
void setIntegrationCycles(I2C_Handle i2cBusDesciptor, uint_least8_t address, uint8_t
cycleValue)
{

  virtualWriteRegister(i2cBusDesciptor, address, AS7265X_INTERGRATION_TIME,
cycleValue);
}

bool dataAvailable(I2C_Handle i2cBusDesciptor, uint_least8_t address)
{
  uint8_t value = virtualReadRegister(i2cBusDesciptor, address, AS7265X_CONFIG);
  return (value & (1 << 1)); //Bit 1 is DATA_RDY
}

/* Задати конфігурацію множника */
void setGain(I2C_Handle i2cBusDesciptor, uint_least8_t address, uint8_t gain)
{
  if (gain > 0b11)
    gain = 0b11;

  uint8_t value = virtualReadRegister(i2cBusDesciptor, address, AS7265X_CONFIG);
  value &= 0b11001111;
  value |= (gain << 4);
  virtualWriteRegister(i2cBusDesciptor, address, AS7265X_CONFIG, value);
}

/* Задати конфігурацію способу вімиру */
void setMeasurementMode(I2C_Handle i2cBusDesciptor, uint_least8_t address, uint8_t
mode)
```

```
{
  if (mode > 0b11)
    mode = 0b11;


  uint8_t value = virtualReadRegister(i2cBusDesciptor, address, AS7265X_CONFIG);
  value &= 0b11110011;
  value |= (mode << 2);
  virtualWriteRegister(i2cBusDesciptor, address, AS7265X_CONFIG, value);
}

void takeMeasurements(I2C_Handle i2cBusDesciptor, uint_least8_t address)
{
  setMeasurementMode(i2cBusDesciptor, address,
AS7265X_MEASUREMENT_MODE_6CHAN_ONE_SHOT);
  uint8_t attempt = 1;
  while (dataAvailable(i2cBusDesciptor, address) == false)
  {
    if(attempt > maxAttempts) return;
    sleep(AS7265X_POLLING_DELAY);
    attempt++;
  }

}


float convertBytesToFloat(uint32_t myLong)
{
  float myFloat;
  memcpy(&myFloat, &myLong, 4); //Copy bytes into a float
  return (myFloat);
}

/* Задати дані спектральних датчиків, відкалібровані значення */
float getCalibratedValue(I2C_Handle i2cBusDesciptor, uint_least8_t address, uint8_t
calAddress, uint8_t device)
{
  selectDevice(i2cBusDesciptor, address, device);

  uint8_t b0, b1, b2, b3;
  b0 = virtualReadRegister(i2cBusDesciptor, address, calAddress + 0);
  b1 = virtualReadRegister(i2cBusDesciptor, address, calAddress + 1);
  b2 = virtualReadRegister(i2cBusDesciptor, address, calAddress + 2);
  b3 = virtualReadRegister(i2cBusDesciptor, address, calAddress + 3);

  //Channel calibrated values are stored big-endian
  uint32_t calBytes = 0;
  calBytes |= ((uint32_t)b0 << (8 * 3));
  calBytes |= ((uint32_t)b1 << (8 * 2));
  calBytes |= ((uint32_t)b2 << (8 * 1));
  calBytes |= ((uint32_t)b3 << (8 * 0));

  //System_printf("Debug value %d\n", calBytes);

  return (convertBytesToFloat(calBytes));
}

float getCalibratedA(I2C_Handle i2cBusDesciptor, uint_least8_t address)
{
  return (getCalibratedValue(i2cBusDesciptor, address, AS7265X_R_G_A_CAL,
AS72653_UV));
```

```
}
float getCalibratedB(I2C_Handle i2cBusDesciptor, uint_least8_t address)
{
  return (getCalibratedValue(i2cBusDesciptor, address, AS7265X_S_H_B_CAL,
AS72653_UV));
}
float getCalibratedC(I2C_Handle i2cBusDesciptor, uint_least8_t address)
{
  return (getCalibratedValue(i2cBusDesciptor, address, AS7265X_T_I_C_CAL,
AS72653_UV));
}
float getCalibratedD(I2C_Handle i2cBusDesciptor, uint_least8_t address)
{
  return (getCalibratedValue(i2cBusDesciptor, address, AS7265X_U_J_D_CAL,
AS72653_UV));
}
float getCalibratedE(I2C_Handle i2cBusDesciptor, uint_least8_t address)
{
  return (getCalibratedValue(i2cBusDesciptor, address, AS7265X_V_K_E_CAL,
AS72653_UV));
}
float getCalibratedF(I2C_Handle i2cBusDesciptor, uint_least8_t address)
{
  return (getCalibratedValue(i2cBusDesciptor, address, AS7265X_W_L_F_CAL,
AS72653_UV));
}

float getCalibratedG(I2C_Handle i2cBusDesciptor, uint_least8_t address)
{
  return (getCalibratedValue(i2cBusDesciptor, address, AS7265X_R_G_A_CAL,
AS72652_VISIBLE));
}
float getCalibratedH(I2C_Handle i2cBusDesciptor, uint_least8_t address)
{
  return (getCalibratedValue(i2cBusDesciptor, address, AS7265X_S_H_B_CAL,
AS72652_VISIBLE));
}
float getCalibratedI(I2C_Handle i2cBusDesciptor, uint_least8_t address)
{
  return (getCalibratedValue(i2cBusDesciptor, address, AS7265X_T_I_C_CAL,
AS72652_VISIBLE));
}
float getCalibratedJ(I2C_Handle i2cBusDesciptor, uint_least8_t address)
{
  return (getCalibratedValue(i2cBusDesciptor, address, AS7265X_U_J_D_CAL,
AS72652_VISIBLE));
}
float getCalibratedK(I2C_Handle i2cBusDesciptor, uint_least8_t address)
{
  return (getCalibratedValue(i2cBusDesciptor, address, AS7265X_V_K_E_CAL,
AS72652_VISIBLE));
}
float getCalibratedL(I2C_Handle i2cBusDesciptor, uint_least8_t address)
{
  return (getCalibratedValue(i2cBusDesciptor, address, AS7265X_W_L_F_CAL,
AS72652_VISIBLE));
}

float getCalibratedR(I2C_Handle i2cBusDesciptor, uint_least8_t address)
{
```

```
  return (getCalibratedValue(i2cBusDesciptor, address, AS7265X_R_G_A_CAL,
AS72651_NIR));
}
float getCalibratedS(I2C_Handle i2cBusDesciptor, uint_least8_t address)
{
  return (getCalibratedValue(i2cBusDesciptor, address, AS7265X_S_H_B_CAL,
AS72651_NIR));
}
float getCalibratedT(I2C_Handle i2cBusDesciptor, uint_least8_t address)
{
  return (getCalibratedValue(i2cBusDesciptor, address, AS7265X_T_I_C_CAL,
AS72651_NIR));
}
float getCalibratedU(I2C_Handle i2cBusDesciptor, uint_least8_t address)
{
  return (getCalibratedValue(i2cBusDesciptor, address, AS7265X_U_J_D_CAL,
AS72651_NIR));
}
float getCalibratedV(I2C_Handle i2cBusDesciptor, uint_least8_t address)
{
  return (getCalibratedValue(i2cBusDesciptor, address, AS7265X_V_K_E_CAL,
AS72651_NIR));
}
float getCalibratedW(I2C_Handle i2cBusDesciptor, uint_least8_t address)
{
  return (getCalibratedValue(i2cBusDesciptor, address, AS7265X_W_L_F_CAL,
AS72651_NIR));
}

/* Запуск роботи по I2C регістру */
void read AS7265XSensor()
{

    uint16_t         sample;
    uint16_t         temperature;
    uint8_t          txBuffer[1];
    uint8_t          rxBuffer[2];
    I2C_Handle       i2c;
    I2C_Params       i2cParams;
    I2C_Transaction i2cTransaction;
    I2C_Transaction readI2cTransaction;


    Display_init();
    I2C_init();


    I2C_Params_init(&i2cParams);
    i2cParams.bitRate = I2C_3400kHz;
    //i2cParams.transferMode  = I2C_MODE_BLOCKING;
    i2c = I2C_open(Board_I2C_TMP, &i2cParams);

    if (i2c == NULL) {
        System_printf("Error Initializing I2C\n");
    }
    else {

        System_printf("I2C Initialized!\n");
    }

    i2cTransaction.writeBuf   = txBuffer;
```

```
    i2cTransaction.writeCount = 1;
    i2cTransaction.readBuf    = rxBuffer;
    i2cTransaction.readCount  = 2;



    i2cTransaction.slaveAddress = TMP116_LP_ADDR;
    if (!I2C_transfer(i2c, &i2cTransaction)) {
        System_printf("Error. No sensor found!\n");

    }
    else {

    System_printf("Slave address is %d\n", i2cTransaction.slaveAddress);

    uint8_t value = virtualReadRegister(i2c, 0x49, AS7265X_DEV_SELECT_CONTROL);

      if ((value & 0b00110000) == 0){
          System_printf("Slave1 and 2 are not detected\n");
      }
      else {
          System_printf( "Slave1 and 2 are detected\n");
      }
    setBulbCurrent(i2c, 0x49, AS7265X_LED_CURRENT_LIMIT_12_5MA, AS7265x_LED_WHITE);
    setBulbCurrent(i2c, 0x49, AS7265X_LED_CURRENT_LIMIT_12_5MA, AS7265x_LED_IR);
    setBulbCurrent(i2c, 0x49, AS7265X_LED_CURRENT_LIMIT_12_5MA, AS7265x_LED_UV);

    disableBulb(i2c, 0x49, AS7265x_LED_WHITE); //
    disableBulb(i2c, 0x49, AS7265x_LED_IR);
    disableBulb(i2c, 0x49, AS7265x_LED_UV);
//
    setIntegrationCycles(i2c, 0x49, 0x3b);
    setGain(i2c, 0x49, 0b10);
    System_printf("UV sensor temperature: %d (C)\n", getTemperature(i2c, 0x49,
AS72653_UV));

          System_printf(" %f, %f, %f, %f, %f, %f, %f, %f, %f, %f, %f, %f, %f, %f, %f,
%f, %f, %f\n", getCalibratedA(i2c, 0x49), getCalibratedB(i2c, 0x49),
getCalibratedC(i2c, 0x49), getCalibratedD(i2c, 0x49), getCalibratedE(i2c, 0x49),
getCalibratedF(i2c, 0x49), getCalibratedG(i2c, 0x49), getCalibratedH(i2c, 0x49),
getCalibratedR(i2c, 0x49), getCalibratedI(i2c, 0x49), getCalibratedS(i2c, 0x49),
getCalibratedJ(i2c, 0x49), getCalibratedT(i2c, 0x49), getCalibratedU(i2c, 0x49),
getCalibratedV(i2c, 0x49), getCalibratedW(i2c, 0x49), getCalibratedK(i2c, 0x49),
getCalibratedL(i2c, 0x49));
          System_printf("--------------------------------FAR-------------------------
-----------");
          sleep(5);
          System_printf(" 410 nM %f\n", getCalibratedA(i2c, 0x49));
          System_printf(" 435 nM %f\n", getCalibratedB(i2c, 0x49));
          System_printf(" 460 nM %f\n", getCalibratedC(i2c, 0x49));
          System_printf(" 485 nM %f\n", getCalibratedD(i2c, 0x49));
          System_printf(" 510 nM %f\n", getCalibratedE(i2c, 0x49));
          System_printf(" 535 nM %f\n", getCalibratedF(i2c, 0x49));
          System_printf(" 560 nM %f\n", getCalibratedG(i2c, 0x49));
          System_printf(" 585 nM %f\n", getCalibratedH(i2c, 0x49));
          System_printf(" 610 nM %f\n", getCalibratedR(i2c, 0x49));
          System_printf(" 645 nM %f\n", getCalibratedI(i2c, 0x49));
          System_printf(" 680 nM %f\n", getCalibratedS(i2c, 0x49));
          System_printf(" 705 nM %f\n", getCalibratedJ(i2c, 0x49));
          System_printf(" 730 nM %f\n", getCalibratedT(i2c, 0x49));
          System_printf(" 760 nM %f\n", getCalibratedU(i2c, 0x49));
```

```
            System_printf(" 810 nM %f\n", getCalibratedV(i2c, 0x49));
            System_printf(" 860 nM %f\n", getCalibratedW(i2c, 0x49));
            System_printf(" 900 nM %f\n", getCalibratedK(i2c, 0x49));
            System_printf(" 940 nM %f\n", getCalibratedL(i2c, 0x49));

    }
    if (i2c != NULL) {
        I2C_close(i2c);
        System_printf("I2C closed!\n");
    }
    System_printf("COOOL END\n");
       return (NULL);
```

Додаток 2. Лістинг програми Sensor, C++ . Частина по відправці даних на концентратор.

```
/* Структура для пересилки даних */
typedef struct _AS7265xSmsgs_sensormsg_t
{
    /*! Command ID */
    Smsgs_cmdIds_t cmdId;
    /*! Extended Address */
    uint8_t extAddress[SMGS_SENSOR_EXTADDR_LEN];
    /*! AS7265x */
       uint32_t AS7265x[18];

/* Заповнення структури */
static void processSensorMsgEvt(void) {
              Smsgs_AS7265xsensorMsg_t sensor;
              sensor.AS7265x[0] = getCalibrated410(i2c, 0x49);
              sensor.AS7265x[1] = getCalibrated410(i2c, 0x49);
              sensor.AS7265x[2] = getCalibrated410(i2c, 0x49);
              sensor.AS7265x[3] = getCalibrated410(i2c, 0x49);
              sensor.AS7265x[4] = getCalibrated410(i2c, 0x49);
              sensor.AS7265x[5] = getCalibrated410(i2c, 0x49);
              sensor.AS7265x[6] = getCalibrated410(i2c, 0x49);
              sensor.AS7265x[7] = getCalibrated410(i2c, 0x49);
              sensor.AS7265x[8] = getCalibrated410(i2c, 0x49);
              sensor.AS7265x[9] = getCalibrated410(i2c, 0x49);
              sensor.AS7265x[10] = getCalibrated410(i2c, 0x49);
              sensor.AS7265x[11] = getCalibrated410(i2c, 0x49);
              sensor.AS7265x[12] = getCalibrated410(i2c, 0x49);
              sensor.AS7265x[13] = getCalibrated410(i2c, 0x49)
              sensor.AS7265x[14] = getCalibrated410(i2c, 0x49)
              sensor.AS7265x[15] = getCalibrated410(i2c, 0x49);
              sensor.AS7265x[16] = getCalibrated410(i2c, 0x49);
              sensor.AS7265x[17] = getCalibrated410(i2c, 0x49);
              sendAS7265xSensorMessage(ApiMac_sAddr_t *pDstAddr,
Smsgs_AS7265xsensorMsg_t *pMsg)


                memset(&sensor, 0, sizeof(Smsgs_sensorMsg_t));

/* відправка даних на концентратор*/
static bool sendAS7265xSensorMessage(ApiMac_sAddr_t *pDstAddr,
Smsgs_AS7265xsensorMsg_t *pMsg)
{
```

```
    bool ret = false;
    uint8_t *pMsgBuf;
    //len = 18 * 4 + 8 + 1 = 81
    uint16_t len = 81;

    pMsgBuf = (uint8_t *)Ssf_malloc(len);
    if(pMsgBuf)
    {
        uint8_t *pBuf = pMsgBuf;

        *pBuf++ = (uint8_t)Smsgs_cmdIds_AS7265xsensorData;

        memcpy(pBuf, pMsg->extAddress, SMGS_SENSOR_EXTADDR_LEN);
        pBuf += SMGS_SENSOR_EXTADDR_LEN;
        memcpy(pBuf, pMsg->AS7265x, 72);

        ret = Sensor_sendMsg(Smsgs_cmdIds_AS7265xsensorData, pDstAddr, true, len,
pMsgBuf);
        Ssf_free(pMsgBuf);
    }

    return (ret);
}
```

Додаток 3. Лістинг програми Collector, C++ . Частина по отриманню даних колектором(частина концентратору) з сенсору.

```
/* колбек прибуття вхідного повідомлення на колекторі */
        switch(cmdId)
        {
            case Smsgs_cmdIds_configRsp:
                processConfigResponse(pDataInd);
                break;

            case Smsgs_cmdIds_trackingRsp:
                processTrackingResponse(pDataInd);
                break;

            case Smsgs_cmdIds_IdentifyLedReq:
                processIdendifyLedRequest(pDataInd);
                break;

            case Smsgs_cmdIds_toggleLedRsp:
                processToggleLedResponse(pDataInd);
                break;

            case Smsgs_cmdIds_sensorData:
                processSensorData(pDataInd);
                break;

           case Smsgs_cmdIds_AS7265xsensorData:
                processAS7265xSensorData(pDataInd);
                break;

            case Smsgs_cmdIds_rampdata:
                Collector_statistics.sensorMessagesReceived++;
                break;
#ifdef FEATURE_SECURE_COMMISSIONING
            case Smgs_cmdIds_CommissionMsg:
```

```
                /* Process Security manager commissioning data */
                SM_processCommData(pDataInd);
                break;
#endif /* FEATURE_SECURE_COMMISSIONING */

            default:
                /* Should not receive other messages */
                break;
        }
    }
}


/* обробка вхідного AS7625x повідомлення на колекторі */


static void processAS7265xSensorData (ApiMac_mcpsDataInd_t *pDataInd)
{
    Smsgs_sensorMsg_t sensorData;
    uint8_t *pBuf = pDataInd->msdu.p;

    memset(&sensorData, 0, sizeof(Smsgs_sensorMsg_t));

    /* Parse the message */
    sensorData.cmdId = (Smsgs_cmdIds_t)*pBuf++;

    memcpy(sensorData.AS7265x, pBuf, SMGS_SENSOR_EXTADDR_LEN);
       pBuf += SMGS_SENSOR_EXTADDR_LEN;


    memcpy(sensorData.extAddress, pBuf, SMGS_SENSOR_EXTADDR_LEN);
       pBuf += SMGS_SENSOR_EXTADDR_LEN;



/* Відправка вхідного повідомлення AS7625x в TCP/IP інтерфейс NPI севера*/
    Csf_deviceSensorDataUpdate(&pDataInd->srcAddr, pDataInd->rssi,
                               &sensorData);

    processDataRetry(&(pDataInd->srcAddr));
}
```

Додаток 4. Лістинг програми Server OPC UA Python 3.8.10

```
# Імпорт необхідних бібліотек
from opcua import Server
import socket


# Вхідні дані
OPC_SERVER_URL = "opc.tcp://0.0.0.0:12345"
OPC_SERVER_NAMESPACE = "greenhouse"
NPI_GATEWAY_SERVER_ADDRESS = "192.168.0.168"
NPI_GATEWAY_SERVER_PORT = 5000
```

```
# Ініціалізвція OPC Server
server = Server()
server.set_endpoint(OPC_SERVER_URL)
name_space_id = server.register_namespace(OPC_SERVER_NAMESPACE )

# Ініціалізація даних датчиків
# Датчик №1
objects = server.get_objects_node()
temperature_sensor = objects.add_object('ns=%s;s="TS1"' % name_space_id, "Temperature sensor 1")
temp = temperature_sensor.add_variable('ns=%s;s="TS1_Temperature"' % name_space_id, "Temperature value", 0)
led = temperature_sensor.add_variable('ns=%s;s="TS1_Led"' % name_space_id, "Temperature sensor led", True)
led.set_writable()

# Датчик №2
temperature_sensor2 = objects.add_object('ns=%s;s="TS2"' % name_space_id, "Temperature sensor 2")
temp2 = temperature_sensor2.add_variable('ns=%s;s="TS2_Temperature"' % name_space_id, "Temperature value",
0)
led2 = temperature_sensor2.add_variable('ns=%s;s="TS2_Led"' % name_space_id, "Temperature sensor led", True)
led2.set_writable()

# Датчик №3
temperature_sensor3 = objects.add_object('ns=%s;s="TS3"' % name_space_id, "Temperature sensor 3")
temp3 = temperature_sensor3.add_variable('ns=%s;s="TS3_Temperature"' % name_space_id, "Temperature value",
0)
led3 = temperature_sensor3.add_variable('ns=%s;s="TS3_Led"' % name_space_id, "Temperature sensor led", True)
led3.set_writable()

# Реєстрація датчиків в системі
devices = {'1348011319075180': temp,
        '011811719075180': temp2,
        '13315111219075180': temp3}

# Реєстрація коротких адресів 802.14.5
short_addresses = {}


try:
    # Старт OPC server
    print("starting OPC server")
    server.start()
    print("OPC server is online")
    # ОБробка вхідних повідомлень
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((NPI_GATEWAY_SERVER_ADDRESS, NPI_GATEWAY_SERVER_PORT))
        while True:
            data = s.recv(1024)
            # Отриман пакет з NPI шлюзу, друкуємо вміст пакету
            print(f"Received package:{data!r}")
            package = bytearray(data)
            # друкуємо довжину пакету
            print("package length:%s" % len(package))
            # Перевірка типу пакту, потрібні лише дані датчиків
            if len(package) >= 88 and package[8] == 5:
                # Зчитуємо адресу датчика
                sensor_address = "%s%s%s%s%s%s%s%s" % (package[9], package[10], package[11], package[12],
                                                       package[13], package[14], package[15], package[16])
                sensor = devices.get(sensor_address)
                # Перівірка чи відомий в системі датчик
                if sensor is not None:
```

```
                # Обновляємо змінні OPC даними датчіків
                sensor.set_value(package[19])
                short_address = short_addresses.get(sensor_address)
                if short_address is not None or short_address != package[5]:
                    short_addresses[sensor_address] = package[5]
            else:
                # друкуємо помилку відсутності регістрації
                print("Not registered sensor: %s" % sensor_address)

        # Друкуємо змінні OPC
        print(str(temp.get_value()))
        print(str(temp2.get_value()))
        print(str(temp3.get_value()))
finally:
    # Завершення роботи OPC SERVER
    server.stop()
```