Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Прикладних інформаційних технологій та електроінженерії
(повна назва факультету)
Комп'ютерно-інтегрованих технологій
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: **Розробка автоматизованої системи керування боксом вирощувальної системи**

Виконав(ла): студент(ка) IV курсу, групи КТ-41
спеціальності *151 – Автоматизація та комп'ютерно-інтегровані технології*
(шифр і назва спеціальності)

	(підпис)	Хомин І.Б. (прізвище та ініціали)
Керівник	(підпис)	Дідич І.С. (прізвище та ініціали)
Нормоконтроль	(підпис)	Чихіра І.В. (прізвище та ініціали)
Завідувач кафедри	(підпис)	Микитишин А.Г. (прізвище та ініціали)
Рецензент	(підпис)	Марущак П.О. (прізвище та ініціали)

Тернопіль
2023

АНОТАЦІЯ

Кваліфікаційна робота складається з пояснювальної записки та графічної частини.

Об’єм графічної частини кваліфікаційної роботи становить __ слайдів. Об’єм пояснювальної записки складає __ друкованих сторінок формату А4 (210×297), об’єм додатків – __ друкованих сторінок формату А4.

У роботі було розроблено систему керування боксом вирощувальної ділянки.

Для цього було проаналізовано можливі технології для створення вирощувальних комплексів. На основі проведено аналізу було прийнято остаточну структуру системи.

Далі було описано структуру системи та алгоритм її роботи. Для обраної струтури системи було вибрано всі апаратні засоби та розроблено схему електричну принципову всіх з’єднань системи.

Після технічної реалізації системи було створено програмне забезпечення для керування боксом, а також мобільний додатко для можливості контролювання процесу в режимі реального часу.

Впровадження результатів роботи дозволить розширити сферу застосування вирощувальних боксів для забезпечення потреб населення.

Ключові слова: КОНТРОЛЕР, КЕРУВАННЯ, БОКС, ВИРОЩУВАЛЬНА СИСТЕМА.

ЗМІСТ

ВСТУП

Сільськогосподарська техніка завжди розвивалася, охоплюючи промислові зміни, які відбувалися відповідно до технологічної еволюції людини. Сьогодні Industry 4.0 представляє собою четверту промислову революцію, яка веде до Agriculture 4.0. Ця остання еволюція в основному визначається поєднанням багатьох нових технологій, таких як Інтернет речей, передова електроніка та робототехніка, великі дані та штучний інтелект. Таким чином, нова екосистема Agriculture 4.0 характеризується керуванням фермою в режимі реального часу, високим ступенем автоматизації та інтелектуальним прийняттям рішень на основі даних.

Нова концепція фабрики розумних рослин є реальним втіленням цієї нової сільськогосподарської парадигми. У цій роботі запропоновано нову платформу автоматизації та інтелекту, яка визначається як базовий шаблон апаратного забезпечення та структура програмного забезпечення, на яких будуються інформаційні та комунікаційні системи, які служать основою для різноманітних послуг. Метою цієї платформи є забезпечення більшості функцій, необхідних цим сучасним фермам. Точніше, головною метою є організація комунікацій і ролей кожного реалізованого модуля.

Для цього реалізовано перевірку концепції апаратного забезпечення моніторингу та мінімальний життєздатний продукт програмного забезпечення керування. Більше того, цей проект був перевірений у реальних експериментах із культивування завдяки виготовленому на замовлення лотку для культивування. Завдяки багатогалузевому підходу та вивченню модульності систем було показано, що проведена розробка вдосконалює наявні на даний момент рішення щодо технічної документації, сумісності з периферійними та хмарними обчисленнями, а також гнучкості апаратного та програмного забезпечення.

1. АНАЛІТИЧНА ЧАСТИНА

1.1. Огляд існуючих рішень в області створення вирощувальних боксів

Сільськогосподарська техніка завжди розвивалася, охоплюючи промислові зміни, які відбувалися відповідно до технологічної еволюції людини. По-перше, ці методи перейшли від ручних традиційних методів ведення сільського господарства до кінця 19 століття до перших машин, використаних у Сільському господарстві 2.0 після першої промислової революції.

Потім методи ведення сільського господарства приєдналися до світу інформаційно-комунікаційних технологій (ІКТ), розвиваючись у сільське господарство 3.0 паралельно з промисловістю 2.0 та промисловістю 3.0, які створили такі галузі, як вбудовані системи, розробка програмного забезпечення та відновлювана енергетика.

Галузь сільського господарства розвивається від свого поточного стану (Сільське господарство 3.0) до майбутньої форми (Сільське господарство 4.0) завдяки триваючій четвертій промисловій революції (Промисловість 4.0).

Нинішня еволюція в основному характеризується злиттям багатьох нових технологій, таких як Інтернет речей (IoT), передова електроніка та робототехніка, великі дані, штучний інтелект (AI) і технології блокчейн (рис. 1.1). Сільське господарство 4.0 (A4) визначається як стале та інтелектуальне промислове сільське господарство, яке досягається за допомогою детального збору, обробки та аналізу просторово-часових даних з усіх аспектів сільськогосподарської галузі в реальному часі.

Таким чином, ця нова сільськогосподарська екосистема характеризується управлінням фермою в режимі реального часу, високим ступенем автоматизації та інтелектуальним прийняттям рішень на основі даних, що може значно підвищити продуктивність, ефективність ланцюга

постачання продовольства, безпечність харчових продуктів та ефективність використання природних і відновлювані ресурси.

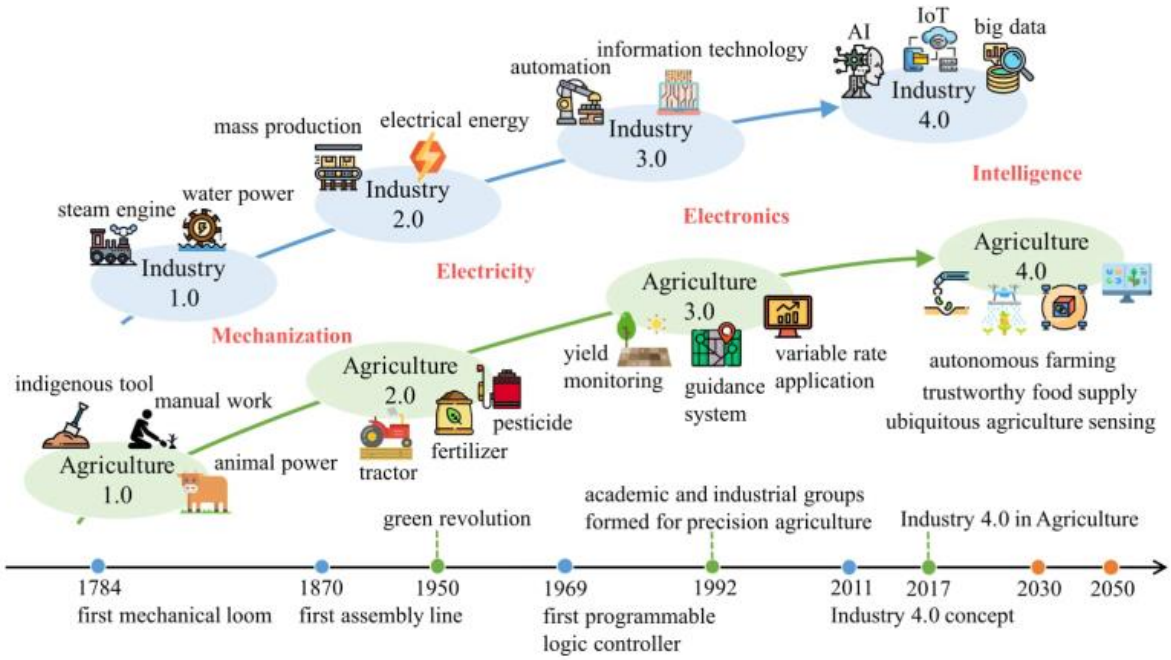


Рисунок 1.1 - Паралельна еволюція промисловості та сільського господарства відповідно до технічного прогресу людини.

Сільське господарство 4.0 є основою останніх проблем. Це один із найбільш вивчених векторів удосконалення через екологічні причини, необхідність наявності їжі, умови безпеки працівників, дослідження фенотипу та генотипу, розробки систем машинного навчання, еволюцію керування даними та мережами або покращення здоров’я людини.

Ця нова сільськогосподарська парадигма впливає на всі методи ведення сільського господарства: від вирощування у відкритому грунті до сільського господарства з контрольованим середовищем (CEA). Крім того, одна з них виділяється завдяки своїй рівновазі компромісів і, отже, потенційно є найцікавішою системою A4: Plant Factory (PF) і її наступник Smart Plant Factory (SPF).

Щоб зрозуміти чому, ось чотири з дев’яти очікуваних кінцевих функцій SPF з книги під назвою «Розумна фабрика рослин: закриті вертикальні ферми наступного покоління», яка визначила цю концепцію:

1. Внесок у вирішення проблем харчування, трилема ресурсів і середовища на особистому, місцевому, регіональному, національному та глобальному рівнях.

2. Внесок у покращення якості життя фізично, розумово та духовно, на додаток до продовольчої безпеки, включаючи стабільне постачання поживної та здорової їжі.

3. Енергоавтономні, екологічно стійкі та економічно життєздатні SPFs, які досягають найвищої продуктивності та якості продукції з мінімальним споживанням ресурсів і максимальним використанням сонячної енергії, енергії біомаси, рідини, теплової та механічної енергії, що призводить до мінімальних витрат виробництва та викидів відходів.

4. SPF, що складається з модулів системи вирощування (CSM), об’єднані один з одним і відкриті для більшості користувачів SPF через Інтернет. Стандартизація підрозділів апаратного та програмного забезпечення CSM для відповідності різноманітним типам CSM.

Як можна зрозуміти з першої очікуваної кінцевої функції у списку вище, розв’язання екологічної трилеми (рис. 1.2) є причиною, чому ця аграрна революція є основою багатьох останніх проблем. Ця тристороння дилема виникає, коли виробництво харчових рослин спрямоване на високу врожайність і високу якість, бажаючи зберегти середовище вирощування та споживаючи мінімальну кількість ресурсів.

Наразі ця трилема не досягла рівноваги, тому переважають її небажані альтернативи: нестабільне постачання продовольства, виснаження ресурсів і погіршення навколишнього середовища.

Дійсно, у цьому ж довіднику стверджується, що «щоб допомогти вирішити цю трилему, необхідно розробити трансдисциплінарні методології, засновані на нових концепціях, за допомогою яких урожайність і якість їжі

значно покращуються з меншими ресурсами. споживання та погіршення навколишнього середовища порівняно з поточними системами рослинництва».

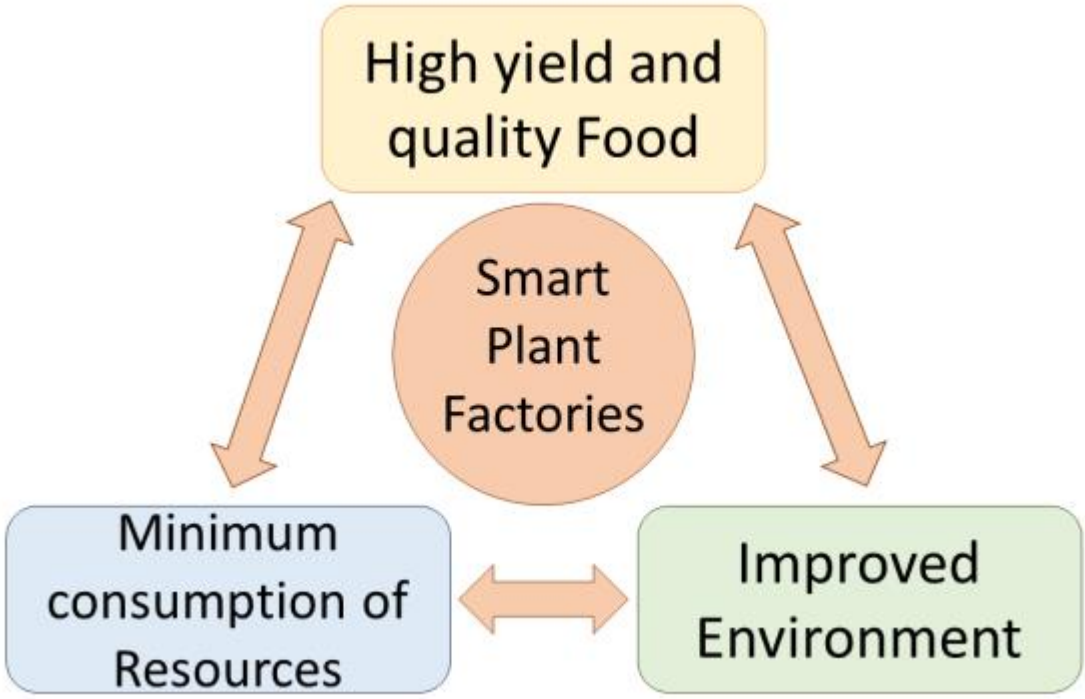


Рисунок 1.2: Представлення трилеми «Ресурси-Їжа-довкілля».

Сільське господарство завжди було, проте багатодисциплінарна галузь (наприклад, біологія, агрономія, хімія тощо) та послідовні промислові революції принесли різні рівні інженерії в галузі інтересів.

Крім того, поточна зміна парадигми сільського господарства поступово інтегрує більше програмного та апаратного забезпечення ІКТ (рис. 1.3).

Ці SPF є основною темою в усьому звіті. Точніше, платформа автоматизації та інтелекту (AIP), яка використовується для створення цих структур, є центром інтересів цієї роботи. Вибір у цьому дослідженні був зроблений завдяки потенційним покращенням, які ці нові технології (наприклад, великі дані, штучний інтелект, Інтернет речей тощо) можуть привнести в сільськогосподарський сектор шляхом вирішення багатьох проблем, які зараз постають перед нами (рис. 1.4 і 1.5).

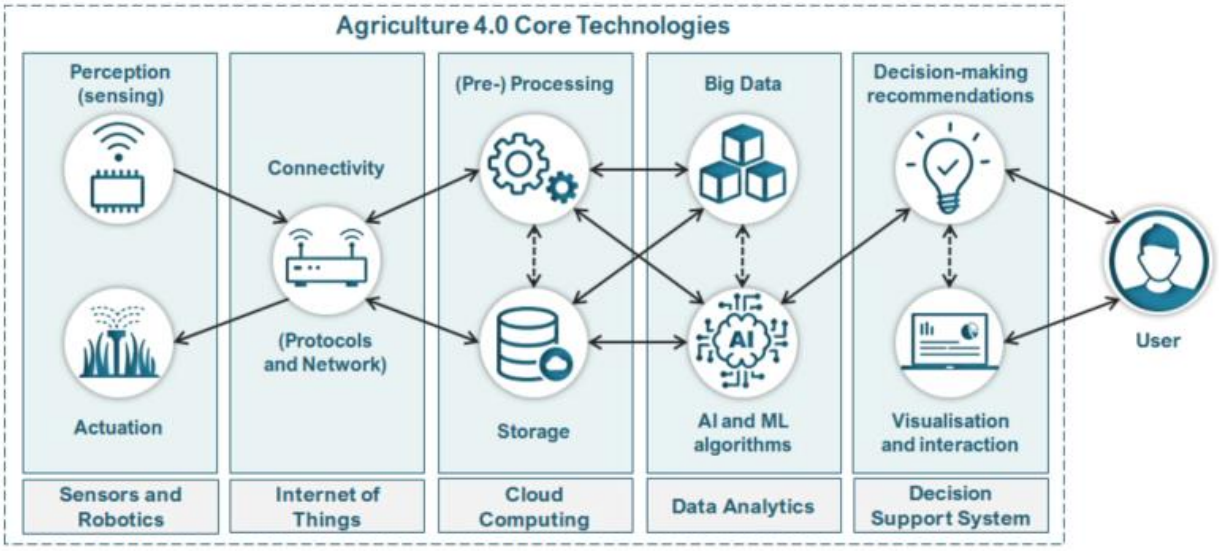


Рисунок 1.3 - Основні ІКТ-технології Agriculture 4.0 та їх взаємодія.

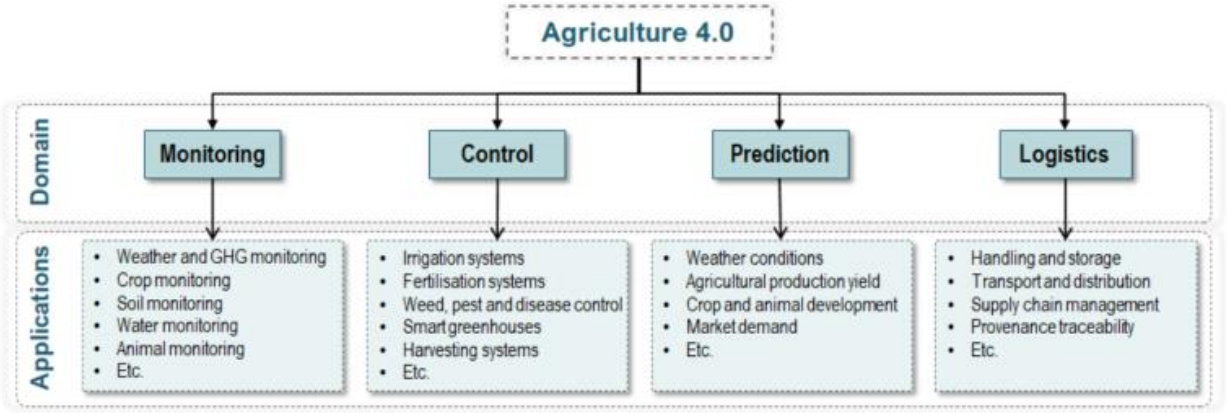


Рисунок 1.4 - Основні ІКТ-технології Agriculture 4.0 та їх потенційне використання.

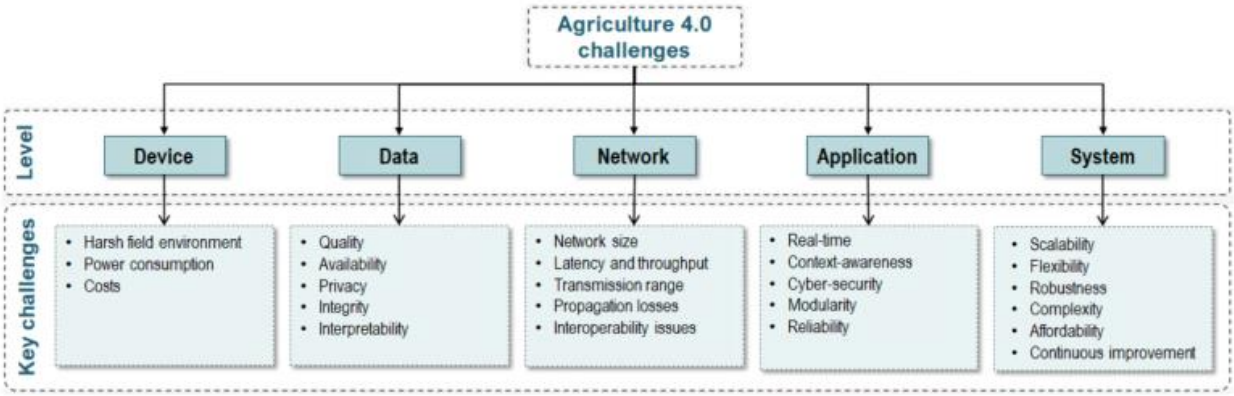


Рисунок 1.5 - Ключові проблеми Agriculture 4.0.

Враховуючи широкий спектр A4, ця робота не може охопити кожну сферу досліджень у рамках SPF. Сфера інтересів, якою є AIP для SPF, також була обрана через відсутність наукової літератури щодо цієї теми. Насправді у світі SPF домінують підприємства, які приховують свої технічні розробки та продають лише кінцеві продукти чи розроблені послуги. На щастя, можна знайти деякі наукові статті про SPF, але їх основна сфера інтересів часто набагато більш зосереджена, ніж корпоративні проекти.

Більшість звітів не надають багато технічних деталей щодо впровадженого та використовуваного AIP. Крім того, доступні деякі роботи любителів, які висвітлюють розробку такої платформи, але часто страждають від недостатньої наукової строгості. Точніше, ця магістерська робота спрямована на вивчення, розробку та тестування нового AIP для SPF. Ця розробка апаратного та програмного забезпечення повинна забезпечувати більшість функцій, які потрібні SPF, і таким чином покращити наявні на даний момент рішення. Для цього аналізуються споживчі технології та оглядається наукова, корпоративна та любительська література.

Крім того, розроблено мінімально життєздатний продукт (MVP) програмного забезпечення для керування та підтвердження концепції (POC) апаратного забезпечення моніторингу. Організація цієї роботи описана далі. Теоретичне нагадування вперше подано в другому розділі цієї роботи. Його перші розділи представляють важливі сільськогосподарські концепції, а останній розділ розглядає сучасні розробки з різних літературних джерел.

З усією попередньою інформацією, викладеною в третьому розділі, представлено позиціонування, цілі та методологію цієї магістерської роботи. Після цих пояснень у четвертому розділі описуються концепції, які використовуються в апаратних і програмних розробках, а також огляд літератури. Потім у трьох наступних розділах представлені розробки апаратного забезпечення, реалізації програмного забезпечення та досягнуті результати. Нарешті, ретроспективне обговорення, перспективний аналіз і висновок завершують цей звіт.

Передумови Оскільки галузь інтересів цієї магістерської роботи є багатогалузевою, цей розділ виступає одночасно як нагадування теорії та роз’яснення концепцій, які беруть участь у цьому проекті. Сучасний сільськогосподарський сектор фактично складається з безлічі підгалузей і перегруповує різні методи ведення сільського господарства. Іменник «сільське господарство» можна визначити як «науку, мистецтво або практику обробітку ґрунту, виробництва врожаю та розведення худоби».

На відміну від традиційного сільського господарства, сучасне землеробство використовує технологічні досягнення для покращення свого виробництва. Більш конкретно, цей звіт цікавить сфери контрольованого сільського господарства (CEA), системи гідропоніки (HS) і вертикального землеробства (VF). Ці три парадигми є підгалузями сучасного сільського господарства, які нещодавно були вдосконалені завдяки галузям інформаційно-комунікаційних технологій (ІКТ).

Крім того, вони були об’єднані разом під концепцією Plant Factory (PF) і Smart Plant Factory (SPF), які за своєю суттю включають використання ІКТ. Усі ці методи ведення сільського господарства (та інші, подібні чи дуже різні) спрямовані на підвищення ефективності (з різними значеннями, наприклад, економічно, екологічно тощо). Однак ця доповідь є дисертацією з інформатики, орієнтованої на інтелектуальні системи. Таким чином, це означає, що дискусії в основному зосереджені на стороні ІКТ сучасного сільського господарства.

Зважаючи на це, все ще важливо підкреслити переваги, недоліки та поточні дослідження цієї сучасної агрономічної концепції, як PF. Крім того, описано три вищезазначені парадигми, щоб краще зрозуміти масштаби сучасних систем сільського господарства, що будуються з урахуванням CEA, HS та VF.

1.2. Сільське господарство з контрольованим середовищем.

Сільське господарство з контрольованим середовищем (CEA) є першою парадигмою ведення сільського господарства, яку використовує ця дипломна робота. Абревіатура в значній мірі зрозуміла, але, щоб було зрозуміліше, CEA перегруповує всі методи вирощування рослин, для яких деяка частина культурного середовища контролюється людиною.

Основна форма CEA, якою є теплиці, еволюціонувала до складних систем, які керують багатьма аспектами середовища вирощування. Початкові системи CEA контролювали лише кілька параметрів навколишнього середовища, таких як температура повітря, вода для поливу або вплив сонячного світла. Сьогодні, крім високотехнологічних теплиць (рис. 1.6), передові системи CEA будують у транспортних контейнерах або встановлюють у сучасні міські будівлі. Як зазначено в тому ж документі, «ці структури [системи CEA] використовують природне або штучне освітлення, в якому створюються оптимальні умови росту для вирощування садових культур або для програм дослідження рослин.

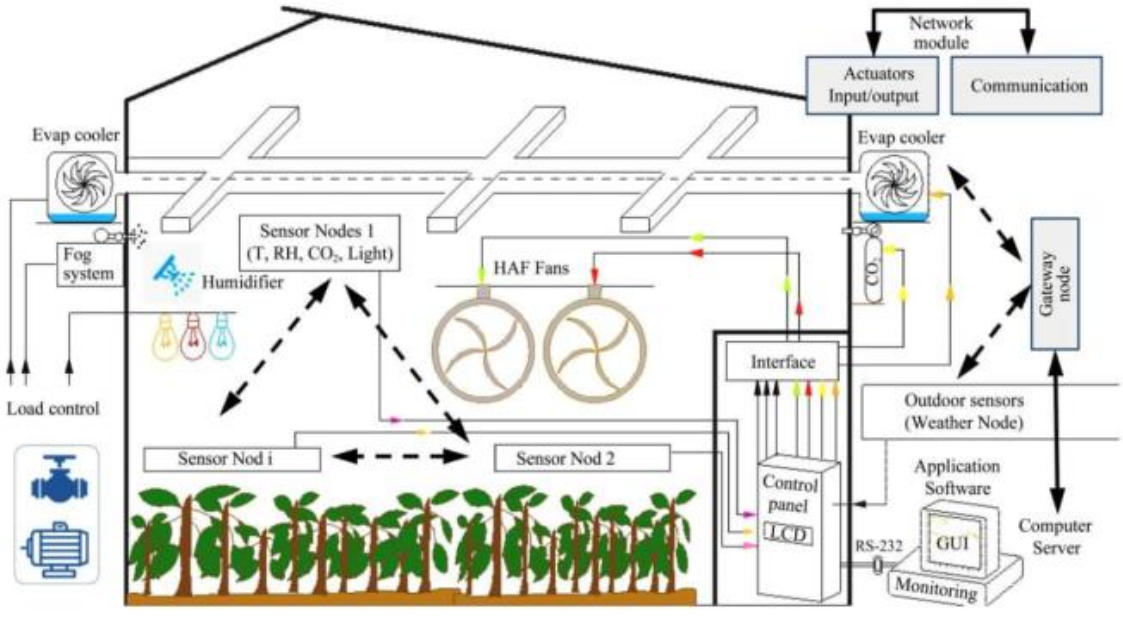


Рисунок 1.6 - Схематичне зображення сучасної теплиці.

Вони також забезпечують більшу передбачуваність, знижують собівартість виробництва та підвищують урожайність». Крім того, вони можуть запропонувати цілорічне виробництво, краще використовувати простір і захистити рослини від погодних умов або фауни.

На жаль, ці інфраструктури не є безпроблемними. Поряд з економічними проблемами, через витрати на будівництво (наприклад, матеріал для покриття) та експлуатаційні витрати (наприклад, контроль освітлення та температури або системи зрошення), основні проблеми пов’язані з кількістю, сортом, кількістю та навантаженням на вирощені культури. до вирощування у відкритому грунті. Крім того, великі безперервні потреби в енергії таких систем часто є найбільш негативним недоліком систем CEA. Нарешті, дослідження, проведене в Детройті (США) у 2017 році, вивчає стійкість і доцільність ферм CEA порівняно з традиційним землеробством, включаючи кілька аспектів.

Їхній головний висновок полягає в тому, що «загальна реструктуризація сільськогосподарської системи, яка використовує традиційні, органічні, міські та CEA у відповідних кількостях і з увагою до відновлюваних джерел енергії, є ключем до сталого майбутнього, а також зміна моделей розподілу таким чином, щоб продукти транспортуються здебільшого до місцевих та регіональних районів».

1.3. Системи гідропоніки

Другою парадигмою землеробства, яку стосується ця теза, є система гідропоніки (HS). Його можна визначити як техніку культивування рослин, яка використовує поживний розчин з інертним середовищем або без нього, що забезпечує механічну підтримку. Таким чином коріння можна безпосередньо занурити в поживний розчин або виростити в інертному середовищі (яке не є ґрунтом). Головні принципи HS полягають у тому, що

живильний розчин має підтримуватися при найкращій температурі, водночас насичений киснем, і рослини мають отримувати через свою кореневу систему необхідні їм поживні речовини. ГС можна класифікувати як з відкритим або замкнутим контуром, а також як активні або пасивні.

Використовується кілька варіантів системи: система припливу та відпливу, техніка поживної плівки, техніка глибокого потоку, крапельна система, глибоководна культивація тощо. Крім того, гідропоніка була виведена в кілька варіантів: аерогідропоніка, аеропоніка, аквапоніка, біопоніка тощо. Основні відмінності між цими варіантами полягають у тому, як виробляються поживні речовини та як вони транспортуються до коренів рослин (рис. 1.7). Як зауваження, різноманітні варіанти системи та варіанти техніки можна поєднувати, тому HS вважаються дуже гнучкими. Головною сильною стороною ГС є контроль над поживними речовинами розподілу, покращення здоров'я рослин, що призводить до меншого використання пестицидів, більшої врожайності та кращої якості, а також доступу до коріння.

Крім того, залежно від обраної техніки та системи, HS може бути більш екологічно чистим, використовувати менше ресурсів і створювати можливості для сільського господарства в екстремальних погодних умовах.

Як завжди, HS мають свої межі: відсутність буферної ємності від середовища (менша толерантність до помилок), керування температурою (складніше в зонах з жарким кліматом), особливі вимоги до конструкції, необхідні деяким рослин (наприклад, морква), а також економічний баланс між вартістю та виробництвом. Існують рішення для всіх цих недоліків, але вони часто збільшують витрати на будівництво та експлуатацію HS. Нарешті, кілька HS вже використовуються або зараз проходять тестування. У статті було проведено дослідження про тестовий стенд у Катарі. Вони розробили систему, яка могла б вирощувати рослини в цьому складному кліматі, одночасно вивчаючи ціну, енергоспоживання, потенціал автоматизації та стійкість.

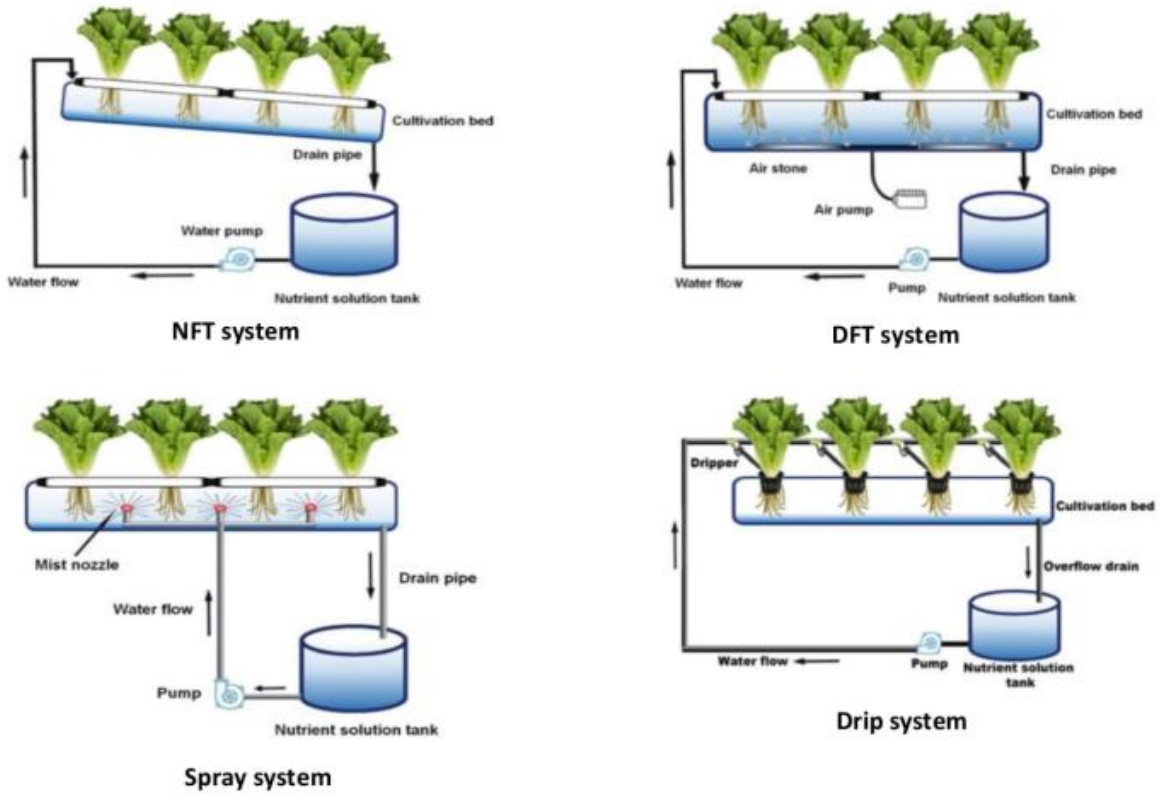


Рисунок 1.7 - Схематичне зображення різних варіантів системи гідропоніки: NFT відповідає техніці поживної плівки, DFT — техніці глибокого потоку, а Spray — техніці аерогідропоніки.

Їхній головний висновок такий: «Ця робота відкриває значні можливості для людей, які живуть у регіоні Перської затоки, виробляти їжу відповідно до своїх потреб».

1.4. Вертикальне землеробство.

Останньою парадигмою землеробства, необхідною для цієї дипломної роботи, є вертикальне землеробство (VF). Це метод ведення сільського господарства в приміщенні, який забезпечує незвичайне сільськогосподарське середовище, не обмежуючись двовимірним простором: рослини розміщують на доріжках для вирощування, які складають один за одним один за одним. Такі, як CEA та HS, системи VF з часом розвивалися, і тепер вони мають різні розміри та форми (рис. 2.3). Найпростіші (і часто

найменші) — це конструкції, які можна повісити на стіну або піраміду, як модулі. Найдосконаліші (і часто найбільші) використовують контейнери CEA або полиці, складені на кілька поверхів. Основними перевагами систем VF є їх економія простору та масштабованість завдяки їхній тривимірній (3D) формі. Крім того, їх вертикальна складова також може бути використана для виділення.

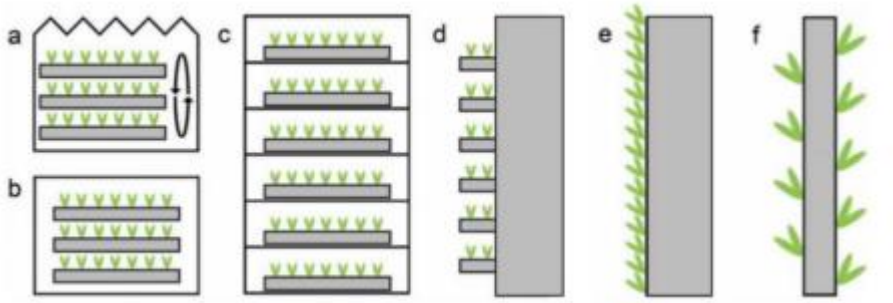


Рисунок 1.8 - Схематичне зображення різних варіантів вертикального землеробства: a - горизонтальні системи в штабелі з обертанням за рівнем; b - горизонтальні системи в штабелі з контрольованим середовищем; c - багатоповерхові вежі; d - вирощування культур на балконах; e - зелені стіни (які можуть бути модифіковані у форму піраміди); f - циліндричні вертикальні одиниці росту.

Новий простір для вирощування на невикористаній території (наприклад, на стінах будівлі). Цікаво, що їхні основні недоліки з часом змінилися: «Більшість авторів у минулому прийшли до висновку, що технологія недостатньо розвинена, щоб створити стабільний клімат для рослин, про який відомо недостатньо, тоді як більшість авторів сьогодні роблять висновок, що вертикальні ферми не є економічно доцільними або досить прибутковими, щоб залишатися в живих без будь-яких великих початкових інвестицій або коштів на етапі їх використання».

Оскільки VF і CEA часто використовуються разом, поточні проблеми у сфері VF дуже схожі на ті, що виникають у сфері CEA. Нарешті, дослідження здійсненності ВФ в Оклахома-Сіті (США) від 2020 року показало, що

«Більшість тверджень, викладених у розділі 2 [тобто ВФ має позитивний екологічний, економічний, соціальний і політичний вплив] цього дослідження, позначені як правильні, що й очікувалося, оскільки вони базуються на певному роді логічного мислення. Однак, як згадувалося раніше, одне твердження, яке часто використовується у верхній частині списку основних переваг вертикальної ферми, насправді не відповідає дійсності згідно з цим дослідженням: слід землі [через виробництво відновлюваної енергії]».

Цей висновок показує, що VF не є ідеальним рішенням, а радше світом компромісів.

1.5. Рослинні фабрики.

Навіть якби можна було використовувати кожну з них окремо, три сільськогосподарські парадигми: CEA, HS і VF часто використовуються разом. Насправді, як було сказано у вступі до глави 2, це призвело до концепції Plant Factory (PF). Ці закриті вертикальні ферми описані в книзі Toyoki Kozai «Plant factory – An indoor вертикальна фермерська система для ефективного виробництва якісних продуктів харчування».

Цей опис відноситься до ПФ як до теплоізольованих і герметичних рослинницьких виробничих приміщень, які мають структуру, схожу на склад, і в яких кілька культурних полиць розташовані вертикально і зрошуються поживним розчином. У сукупності ці агротехніки поєднують свої плюси без додаткових мінусів. Крім того, навіть створюються нові переваги та посилюються оригінальні.

Наприклад, гербіциди та пестициди більше не потрібні, використання простору можна значно оптимізувати, нітрати або поживні речовини витрачаються даремно, а рослини повністю ізольовані від екстремальних і незручних погодних умов і погодних змін. Нарешті, за умови правильного проектування PF має інші основні переваги вищого рівня: його можна

побудувати де завгодно, оскільки не потрібні ні сонячне світло, ні ґрунт, висока ефективність використання ресурсів (води, вуглекислого газу, добрив тощо) може бути досягнута з мінімальними викиди забруднюючих речовин у зовнішнє середовище (рис. 2.4), а вироблені овочі мають довший термін зберігання.

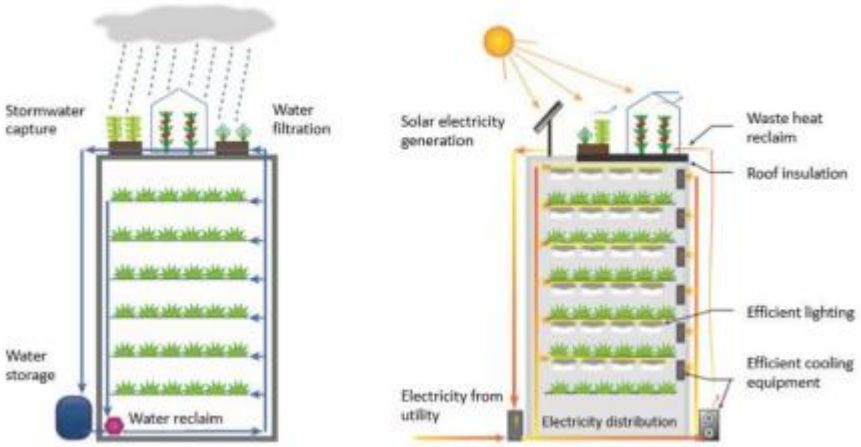


Рисунок 1.9 - Приклад проекту фабрики рослин із замкнутою системою зрошення та системами збору природних ресурсів.

Однак важливо зазначити, що головною проблемою такого типу інфраструктури є економічна рівновага через високі витрати на будівництво та експлуатацію. На щастя, у тій самій книзі можна прочитати, що надія зменшити вплив цього основного недоліку є великою: «Початкові інвестиції можна значно зменшити завдяки кращому дизайну. На щастя, позитивним моментом є те, що витрати на виробництво з кожним роком зменшуються, оскільки накопичується досвід роботи та управління. Електроенергія, праця та матеріали (насіння, добрива, упаковка, доставка тощо) становлять подібні пропорції виробничих витрат. Серед загального споживання електроенергії на освітлення припадає 70-80%, а решту припадає на кондиціонери, насоси та вентилятори.

Існує великий потенціал для зниження вартості освітлення шляхом розробки більш ефективної системи освітлення. Інші підходи до зниження виробничих витрат включають збільшення кількості вертикальних ярусів,

скорочення періоду культивування за допомогою оптимального контролю навколишнього середовища, правильний план виробництва для забезпечення цілорічного виробництва без втрат часу, збільшення щільності посадки та зменшення втрат виробництва». Переглядаючи літературу, про яку йдеться вище, можна побачити, що ці три парадигми сільського господарства часто неявно змішуються.

Концепція PF складається з шести основних структурних елементів рис. 1.10):

1. Споруда, схожа на склад, із теплоізоляційною та майже герметичною оболонкою.

2. Стелажна система для багатошарових культур, де кожне культурне ложе може бути обладнане різними системами контролю (наприклад, освітлення, потік повітря тощо).

3. Система охолодження/опалення (кондиціонери повітря/теплові насоси), яка в основному використовується для регулювання температури (для видалення тепла, що виділяється освітлювальним обладнанням) і осушення повітря (для повторного використання водяної пари, що виділяється рослинами).

4. Система контролю якості повітря, яка в основному використовується як блок доставки вуглекислого газу, який покращує фотосинтез рослин.

5. Система доставки поживного розчину, яка використовується для зрошення рослин.

6. Система контролю навколишнього середовища, яка має принаймні включати блоки керування потенціалом водню (pH) і електропровідністю (EC).

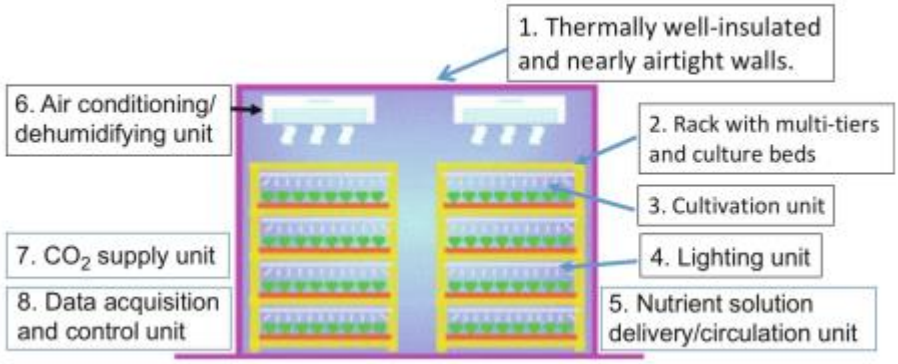


Рисунок 2.5 - Схематичний вигляд концепції Plant Factory з її шістьма основними структурними елементами.

Крім того, добре спроектована PF повинна бути побудована та експлуатована для досягнення цих чотирьох основних цілей:

1. Максимізація придатних для використання та реалізована кількість рослин при використанні мінімальної кількості ресурсів.

2. Збереження найвищої «резефективність використання наших ресурсів» (метрика, яка визначає протоколи, яких слід дотримуватися, щоб якомога ефективніше використовувати різні ресурси).

3. Мінімізація викидів забруднюючих речовин у навколишнє середовище.

4. Мінімізація витрат при досягненні трьох попередніх цілей.

2. ПРОЄКТНА ЧАСТИНА

2.1. Модуль системи культивування

Парадигмою землеробства, яка використовується та вивчається в цій роботі, є концепція SPF. Як було сказано раніше, вона об’єднує сільськогосподарські парадигми HS, VF і CEA. Однак одна ключова концепція, згадана у вступі, необхідна для визначення SPF: модуль системи вирощування (CSM).

CSM — це теоретична концепція з книги Тойокі Козаї «Фабрика розумних рослин: наступне покоління закритих вертикальних ферм». Цей дизайн описує фізичне розташування зростаючої системи та її зв’язок з апаратними компонентами та програмним стеком.

CSM повинен бути легким і простим за структурою. Кожен CSM самостійно займається низькорівневими вимірюваннями, контролем і обробкою інформації». Іншими словами, CSM є базовим компонентом рослинної фабрики для вирощування овочів, яка повинна бути розроблена для штабелювання, адаптації та вдосконалення. Ця концепція модуля системи вирощування є досить важливою, навіть якщо CSM, який розглядається в цій дипломній роботі, не реалізує всі функції, пояснені в оригінальному описі Toyoki Kozai.

Це пов’язано з тим, що ціль цієї магістерської роботи полягає у вивченні платформи автоматизації та інтелекту (AIP) SPF. Проте всі нереалізовані функції все ще повністю сумісні з майбутніми версіями CSM завдяки його модульній конструкції (наприклад, для додавання апаратних давачів) і розробленому AIP (наприклад, для додавання програмного забезпечення для моніторингу).

Щоб бути більш точним, ось список характеристик CSM. Цей список також показує, які можливості наразі підтримуються дизайном CSM,

розробленим для цього проекту (символ X), і які функції можуть бути додані в майбутніх версіях зростаючої системи (• символ).

X Масштабованість:

X Модульність: зробіть кожен компонент CSM максимально незалежним.

X Підключення: стабілізуйте та зробіть надійним кожне з'єднання в межах CSM та між CSM.

X Можливість оновлення: можливе поетапне вдосконалення.

• Керованість:

X вимірювання: дозволяє вимірювати різні параметри системи.

X Зберігання: визначте стратегію зберігання для роботи онлайн і офлайн.

X Обробка: створіть інструменти аналізу для використання зібраних даних.

X Контроль: дозволяє керувати системою через систему керування/моніторингу.

• Фенотипування: фенотипуйте рослини завдяки системі контролю/моніторингу.

• Адаптивність:

X Відкритість: вивчайте відкритість безпеки, стандартизованого протоколу та інтерфейсу прикладного програмного забезпечення.

X Ремонтопридатність: можливість легкого ремонту та заміни.

• Стійкість: оцініть життєвий цикл CSM.

• Продуктивність:

• Витрати на виробництво: розраховуються як добуток економічної вартості одиниці ($/кг) і споживання елемента ресурсу на кг продукції.

• Ефективність витрат: обчислюється співвідношенням продажів (S) до собівартості виробництва (C), значення S можна розрахувати як добуток економічної вартості (U), обсягу продукції (P) і (1,0- L), де L – коефіцієнт втрат продукції.

• Період окупності: визначте продуктивність за елементом ресурсу та витрату елемента ресурсу на кг продукту для вивчення точки беззбитковості.

Як було сказано на початку цього розділу, всі функції CSM наразі не вивчені. Наприклад, ні система освітлення, ні схема повітряного потоку не вивчалися під час розробки запропонованого CSM. Однак, як згадувалося раніше, систему було розроблено таким чином, щоб бути сумісною майже з будь-якою майбутньою модифікацією. Якщо поглянути на приклад конструкції повітряного потоку, то цю модульну систему можна трансформувати в будь-який тип CSM, визначений у довідковій книзі Тойокі Козаї (типи від A до D з їхніми розгалуженнями, рис. 2.1). Крім того, така ж історія щодо моніторингу факторів навколишнього середовища, контролю надходження ресурсів, розрахунків норм видобутку, аналізу фенотипічних ознак плану та факторів ефективності використання ресурсів.

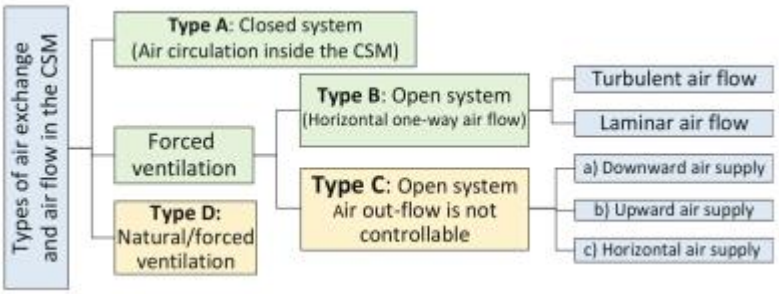


Рисунок 2.1 - Типи CSM з урахуванням схеми повітряного потоку.

Нарешті, щодо взаємодії CSM і балансу між хмарними, локальними та периферійними обчисленнями, архітектура, визначена в довідковій книзі Toyoki Kozai (рис. 2.2), може бути повністю реалізована AIP, розробленим для цього дисертація магістра.

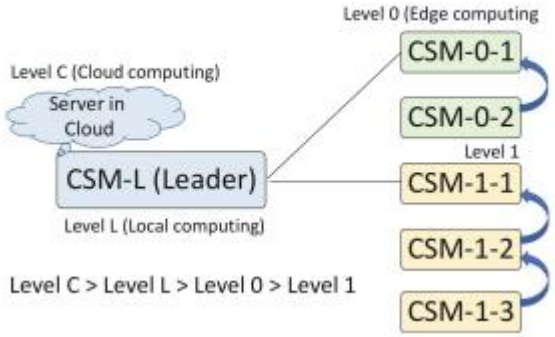


Рисунок 2.2 - Логіка взаємозв’язків CSM.

2.2. Автоматизація та інтелектуальна платформа.

Для підтримки цієї концепції CSM необхідно створити інфраструктуру ІКТ. Для цієї дисертації ця система ІКТ називається Платформою автоматизації та інтелекту (AIP). Він складається з чотирьох елементів: електроніки, CSM, SPF і контролю доступу людини. Це можна побачити на рис. 2.3.

Насправді пізніше буде видно, що логічні організації апаратного та програмного забезпечення дуже схожі на цю фізичну конструкцію. Крім того, кольори та назви, які використовуються в кожній діаграмі стилів універсальної мови моделювання цього звіту, відповідають один одному. Це було зроблено, щоб допомогти зрозуміти, які елементи є частиною яких інших діаграм і навпаки.

Метою AIP є організація комунікацій і ролей кожного модуля ІКТ. Для цього необхідно розробити архітектуру, щоб зробити систему масштабованою та керованою. Крім того, визначені модулі повинні знати, яка їх роль і з якими іншими модулями вони повинні спілкуватися. У

контексті цієї магістерської роботи метою розробки є створення інфраструктури ІКТ, що підтримує систему керування SPF, до якої має бути легко додавати автоматизацію та вивчати когнітивні алгоритми, зберігаючи при цьому масштабованість і керованість класичної системи керування SPF.

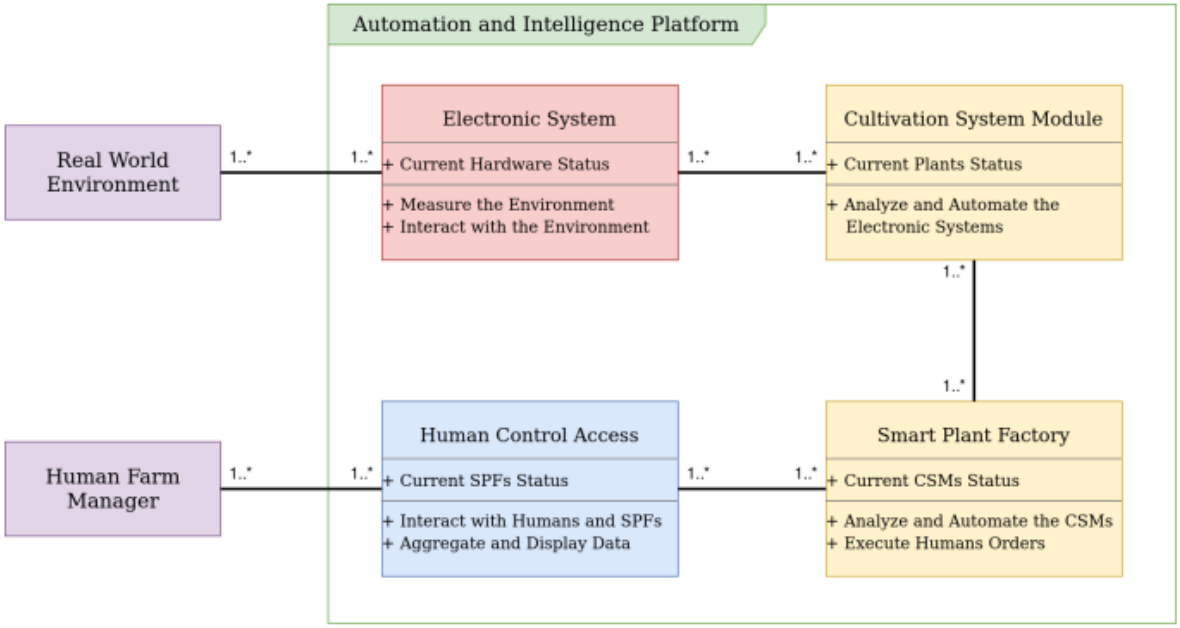


Рисунок 2.3 - Високорівнева діаграма компонентів реального SPF AIP.

Кожен компонент є концептуальним елементом, необхідним для реальної реалізації SPF AIP. Кожен із цих компонентів складається з апаратних і програмних частин: жовті складають електронну систему керування (EMS), синє поле — систему керування (доступ до неї можна отримати через панель управління), червоний компонент — електронні вузли, а зовнішні елементи мають фіолетовий колір.

Щоб краще зрозуміти концепцію AIP, описуються три основні елементи дизайну AIP для SPF: інформація, фізичне середовище та організація простору.

Ці три елементи є взаємозалежними, тому AIP має бути спроектовано відповідним чином шляхом вивчення кожного з трьох елементів. Точніше, з одного боку, інформаційна система (збір, розповсюдження та аналіз)

залежить від середовища, оскільки її джерела інформації (наприклад, давачі, зонди тощо) залежать від середовища.

З іншого боку, інформаційна система залежить від просторової організації ПФ, оскільки вона повинна враховувати цю організацію для аналізу зібраної інформації. Тоді організація простору залежить від інформаційної системи для отримання точних даних і правильної передачі рухів. І просторова організація також залежить від фізичного середовища, оскільки його апаратне забезпечення залежить від фізичного середовища. Нарешті, фізичне середовище залежить від двох інших фундаментальних елементів, щоб розвиватися найкращим чином. Як зауваження, ці взаємодії представлені на рис. 2.4.

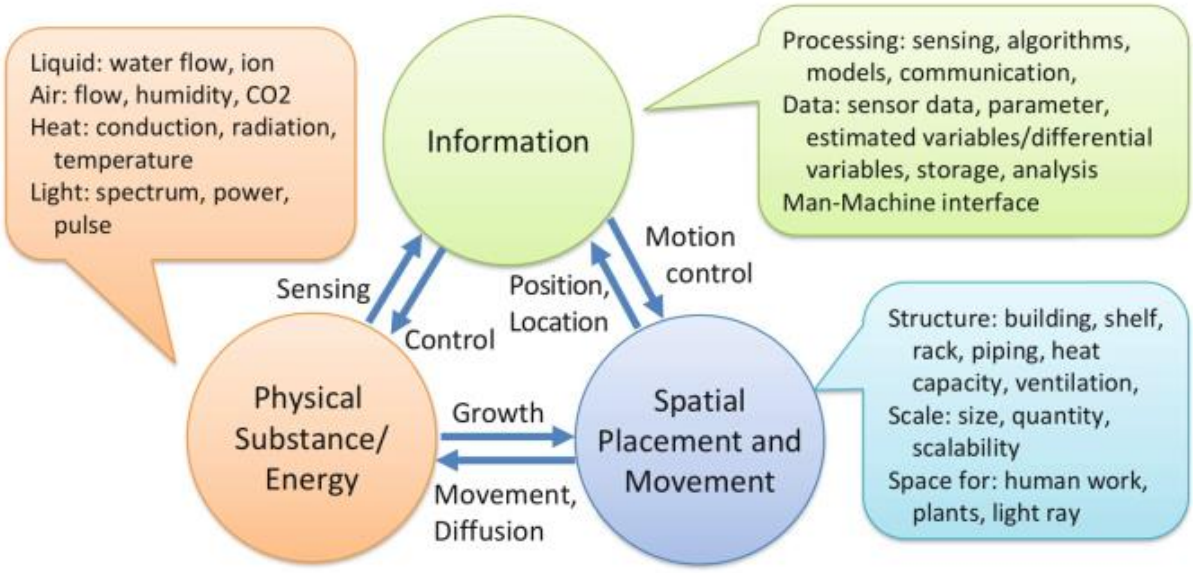


Рисунок 2.4 - Три основні елементи дизайну AIP та їх взаємозалежність.

З огляду SOTA, визначення AIP і опису, наведеного вище, основними цілями AIP низького рівня є:

Інтерфейс необхідного обладнання для моніторингу і контролювати фізичне середовище.

Збирайте, розповсюджуйте та зберігайте дані моніторингу та контролю фізичного середовища.

Реалізуйте деякі алгоритми прийняття рішень і взаємодійте з ними (незалежно від того, чи є ви зі світу когнітивних обчислень).

Забезпечте інтерфейс для втручання людини (з можливостями введення/виведення даних).

Крім того, для цієї роботи були визначені деякі додаткові цілі для вдосконалення розроблених на даний момент систем контролю та моніторингу:

Мати модульну конструкцію з точки зору апаратної підтримки, реалізації алгоритмів прийняття рішень та розробки програмного забезпечення. Бути надійним, автономним і зручним у використанні.

2.3. Перевірка апаратного забезпечення.

Щоб реалізувати апаратну сторону AIP, потрібно зробити деякі вибори. У цьому розділі розглядається апаратне забезпечення, яке використовується в літературі, пояснюються переваги чи недоліки кожного підходу та обґрунтовується вибір компонентів, зроблених для розробленого AIP. Важливо розуміти масштаби вибору обладнання. Насправді ці рішення впливають на програмну частину AIP, а також на кінцеві можливості AIP. Крім того, зроблений вибір також має бути сумісним з усіма цілями низького та високого рівня AIP.

Кожен з наступних параграфів присвячено одній дилемі апаратного забезпечення, і кожен пункт розглядається в три етапи. Для початку наводиться деяка теорія про відповідну сферу інтересів, потім виконується огляд відповідної літератури, і, нарешті, обґрунтовується вибір, зроблений у контексті цієї роботи.

Далі аналізуються такі теми: тип процесора, характеристики обчислювальної плати та вимоги до бажаних давачів і приводів. Що стосується типу блоку обробки, доступні дві можливості: мікроконтролери та

мікропроцесори. Крім того, існує кілька сімейств кожного типу блоку обробки.

Для мікроконтролерів це: вбудовані мікроконтролери, мікроконтролери зовнішньої пам’яті, 8-розрядні мікроконтролери, 16-розрядні мікроконтролери, 32-розрядні мікроконтролери тощо. Що стосується мікропроцесорів, то ці сімейства: мікропроцесори зі складним набором інструкцій, мікропроцесори з інтегральними схемами для спеціального застосування, мікропроцесори зі скороченим набором інструкцій, мікропроцесори з цифровими сигналами тощо.

Для розробки апаратної сторони AIP інтерес зосереджений на вбудованих мікроконтролери та мікропроцесори скорочення набору команд. Насправді дилема полягає у виборі між двома різними екосистемами: блоками мікроконтролерів (MCU) (наприклад, Arduino1) і одноплатними комп’ютерами (SBC) (наприклад, Rapsberry Pi2). Мікропроцесор можна описати як керуючий блок мікрокомп’ютера, який загорнутий у маленьку мікросхему, виконує арифметичні логічні операції та спілкується з іншими пристроями, підключеними до нього.

Це єдина інтегральна схема, в якій поєднані кілька обчислювальних функцій. Зі свого боку, мікроконтролер — це мікросхема, оптимізована для керування електронними пристроями, яка містить пам’ять, процесор і програмований вхід/вихід (I/O). Усі ці компоненти зберігаються в одній інтегральній схемі, яка призначена для виконання конкретного завдання та виконання однієї конкретної програми. Таким чином, основними відмінностями високого рівня (цікавими в контексті цієї роботи) між цими двома обчислювальними пристроями є потужність обробки, енергоспоживання та функціональні можливості.

Мікропроцесори часто є потужнішими, ніж мікроконтролери, але споживають більше енергії, але сумісні з системами з більшою ефективністю (наприклад, з високою швидкістю протоколів мережі та пам’яті, мовою

програмування з вищим рівнем абстракції тощо). У розглянутій літературі використовуються обидва блоки обробки.

MCU віддають перевагу для самодостатніх систем, SBC використовують для систем, які більше залежать від обчислень, а гібридні підходи створюються для більш складних систем.

Що стосується апаратної сторони AIP, необхідно взяти до уваги два фактори: мережеву та обчислювальну потужність. Як буде видно в наступних розділах цієї роботи, розроблений AIP має багатовузлову архітектуру, і ці вузли повинні спілкуватися через локальну мережу. З цієї причини всі MCU без підтримки Wi-Fi або Ethernet не є ідеальними.

Крім того, потрібна сумісність парадигм периферійних обчислень і хмарних обчислень. Таким чином, це змушує зробити вибір на користь SBC. Крім того, обчислювальний блок має бути сумісним із фреймворками ML (наприклад, Tensorflow3) і мовами програмування, які пропонують необхідні функції (наприклад, Python4).

Це також змушує зробити вибір на користь SBC. Таким чином, остаточним вибором є використання сімейства Raspberry Pi, оскільки воно задовольняє всі вимоги та оскільки реалізацію, розроблену для цієї платформи, можна легко перенести на більш потужні SBC, якщо це необхідно (що не стосується початкових розробок для MCU). .

Що стосується функцій обчислювальної плати, сімейство Raspberry Pi (RPI) пропонує кілька варіантів: RPI 4, RPI Zero 2(W), RPI Compute Module 4 і RPI Pico. Однак останнє не має значення через попередній вибір, оскільки RPI Pico є MCU. Більше того, RPI Compute Module 4 — це обчислювальна плата, яка не має портів вводу-виводу, і тому потрібна дочірня плата вводу-виводу, щоб бути корисною в цій розробці.

Щоб уникнути цієї проблеми, цей SBC не розглядається як життєздатна альтернатива на цьому етапі розробки. Крім того, RPI 4 повністю ідентичний RPI Compute Module 4, але має всі необхідні порти введення/виведення. При аналізі наведених вище статей, які використовують SBC, використовується

лише варіант B сімейства RPI (поточний RPI 4 є наступником цього варіанту).

Тому вибір може здатися очевидним, але слід взяти до уваги ще один факт: RPI Zero 2(W) було випущено після дати публікації всіх цих статей. Крім того, попередник цієї дошки був не дуже переконливим, і тому він не використовувався ні в одному з попередніх побачений розвиток подій.

Для поточних розробок потреба в обчислювальних потужностях наразі не надто висока. Основна відмінність між RPI 4 і RPI Zero 2(W) полягає в їхній обчислювальній потужності та вводу/виводу. Оскільки доступних портів вводу/виводу RPI Zero 2(W) достатньо для потреб цього проекту (тобто використовуються лише порти GPIO) і що обчислювальна потужність обох моделей достатня, варіант Wi-Fi RPI Zero 2W вибрано для розробки AIP. Цей вибір можна побачити на рис. 2.5.

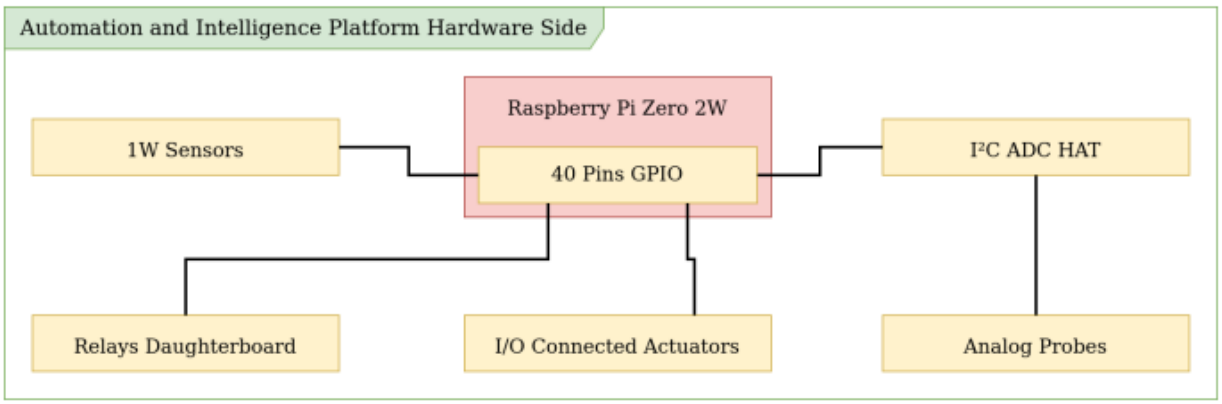


Рисунок 4.5: Апаратна частина AIP.

Що стосується вимог до бажаних давачів і приводів, найважливішим аспектом є сумісність із якомога більшою кількістю пристроїв. На перший погляд, платформа RPI вже добре підходить, оскільки наявні SBC сумісні з більшістю використовуваних протоколів зв'язку (наприклад, I²C, UART, 1W тощо).

Крім того, у випадках, коли протокол пристрою не підтримується, апаратне забезпечення, підключене зверху (HAT) або зовнішня плата може використовуватися для інтерфейсу цього конкретного пристрою.

Переглядаючи літературу, головна тенденція, яку можна знайти, полягає в тому, що в більшості проектів використовувалися давачі та виконавчі механізми рівня замовника.

Таким чином, вибір RPI Zero 2W все ще залишається апаратною платформою для розробки AIP. Цей пристрій відповідає всім необхідним вимогам і є чудовою відправною точкою для реалізації всіх бажаних функцій AIP. Цей вибір та інші обговорюються в розділі 5, де розглядається розробка апаратного забезпечення AIP.

Щоб завершити цей розділ про вибір апаратного забезпечення, можна пояснити останній цікавий момент. У сучасному світі жодна велика електронна платформа не є непридатною. Усі популярні на даний момент рішення широко використовуються та сумісні практично з будь-яким потрібним електронним пристроєм. Наприклад, обрана платформа RPI повинна забороняти використання давачів 5 В, але за допомогою перетворювача логічного рівня це обмеження не існує.

Рішення, прийняті в цьому розділі, очевидно, є результатом глибоких досліджень і, таким чином, є найкращим можливим вибором для поточного розвитку. Однак на момент написання статті лінійка Arduino Pro була оновлена і тепер містить плату Protenta-x8, яка безпосередньо інтегрує мікроконтролер і мікропроцесор. Таким чином, цей тип плати є новою життєздатною альтернативою використанню гібридної системи як з платами Arduino, так і з RPI.

Основне заключне зауваження полягає в тому, що серед популярних платформ жодна з них не була б поганим вибором, оскільки недоліки кожної платформи можна врівноважити підключенням деяких інших типів компонентів (наприклад, HAT або щитів, периферійних або хмарних обчислювальних серверів тощо).

Нарешті, при проектуванні апаратної системи виникають дві дилеми: однорідність чи неоднорідність і загального призначення чи спеціалізована. Для кожної дилеми обидва підходи мають свої плюси та мінуси, і жоден

насправді не кращий за інший. У цій роботі акцент робиться на однорідності та загальному призначенні системи. Таким чином, це справедливо для вибору апаратного забезпечення, і це бажано, оскільки розроблений AIP призначений для розвитку та відкритий для вдосконалень.

2.4. Огляд програмного забезпечення.

Що стосується апаратної сторони AIP, для реалізації його програмних компонентів потрібно зробити деякі вибори. У цьому розділі розглядається програмне забезпечення, розроблене в літературі, пояснюються альтернативні варіанти та обґрунтовуються варіанти реалізації розробленого AIP. Подібно до вибору обладнання, описаного в попередньому розділі, використовувані інструменти розробки програмного забезпечення впливають на кінцеві можливості AIP. Однак зроблений вибір також має бути сумісним з усіма цілями низького та високого рівня AIP.

Крім того, вибір програмного забезпечення має бути придатним для використання на вибраних апаратних компонентах і з ними. На щастя, ця проблема сумісності вже була вирішена під час перевірки апаратного забезпечення. Кожен з наступних абзаців зосереджений на одній програмній дилемі, і кожен пункт розглядається в три етапи. Спочатку наводиться деяка теорія щодо відповідної сфери інтересів, потім виконується огляд відповідної літератури, і, нарешті, обґрунтовується вибір, зроблений у контексті розробки програмного боку AIP.

Теми, які аналізуються далі, це: структура програмування, мережевий протокол і технологія баз даних (БД), а також формати даних. По-перше, що стосується вибору електронного середовища програмування, найважливішим рішенням, яке потрібно прийняти, є вибір мови програмування. Єдиним реальним обмеженням, яке є проблемним, є сумісність мови програмування з апаратним забезпеченням.

Що стосується обраного обчислювального блоку, то RPI Zero 2W сумісний з усіма популярними мовами програмування (і майже з будь-якою існуючою мовою). Це одна з сильних сторін цієї платформи. Проте, що стосується давачів і приводів, які використовуються, ймовірно, що їхні бібліотеки постачальників або виробників доступні лише для платформи Arduino і, зрештою, для Raspberry Pi.

На щастя, бібліотеки пристроїв Arduino можна використовувати на обладнанні RPI завдяки спеціалізованій структурі (тобто RasPiArduino5). Однак цей вибір нецікавий, оскільки використання такого фреймворку лише збирає недоліки кожної платформи, не маючи відповідних переваг. Щоб отримати максимальну віддачу від обраного обладнання, мова Python є найкращим вибором: вона дуже потужна, має величезну підтримку спільноти та дуже популярна для інтерфейсу електронних компонентів.

Ці причини пояснюють, чому ця мова програмування завжди використовувалася в літературі, коли проект базується на RPI. При використанні Python у контексті цієї магістерської роботи не потрібно використовувати особливу структуру програмування. Таким чином, електронною мовою програмування, обраною для розробки програмного забезпечення AIP, є Python.

Цей вибір можна побачити на рис. 2.6. По-друге, щодо структури програмування для інформаційної панелі управління, потрібно зробити кілька варіантів: інтерфейс і бекенд. На щастя, ці два рішення можна прийняти абсолютно незалежно. Щодо інтерфейсної частини наразі лідерами є два фреймворки: Flutter та Ionic React.

Обидва надають необхідні інструменти для створення нативних, кросплатформних, адаптивних і динамічних програм. Основні відмінності високого рівня між ними полягають у даті випуску (тобто Flutter є новішим, ніж Ionic React) та їх логіці (тобто Flutter — це все в одному пакеті, тоді як Ionic React використовує кілька підсистем для створення нативної програми).

Для серверної частини може працювати майже будь-яка мова програмування, оскільки доступні різні фреймворки.

Для цього проекту розробки AIP обрано фреймворки Flutter і Django. Обидва були обрані через їхній фактичний та ефективний підхід, коли мова заходить про створення веб- і мобільних додатків, а також через їх сумісність із пізнішими варіантами програмування (наприклад, обраною технологією бази даних). Як пояснюється далі в цій роботі, не було реалізовано жодного повноцінного бекенда, але було розроблено повний інтерфейс. Ці варіанти можна побачити на рис. 2.6. По-третє, для рішення мережевого протоколу вибір є більш вирішальним.

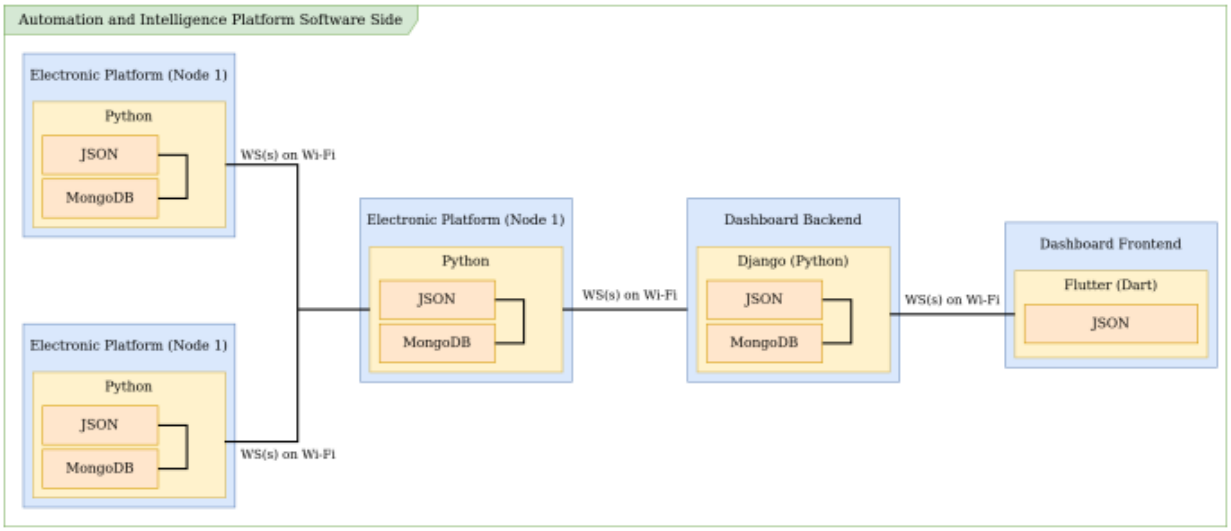


Рисунок 2.6 - Програмна сторона AIP.

Особливості проекту впровадження AIP: двонаправлений зв'язок та автоматизація у серверному вузлі. Для першого фактично необхідно, щоб вузли, які становлять програмну частину AIP, могли обмінюватися повідомленнями та інформацією з усіма іншими вузлами. Таким чином, це означає, що потрібно використовувати кілька односторонніх каналів зв’язку або потрібна єдина двостороння система зв’язку.

Що стосується другої особливості, автоматизації сервера, це означає, що може бути корисним контроль над комунікаційним сервером, а також над клієнтами. Якщо така функція можлива, деякі автоматизовані або когнітивні

обчислювальні алгоритми можуть використовуватися на самому сервері для моніторингу та вдосконалення AIP.

На додаток до цих двох особливостей також необхідно зробити кілька варіантів. Оскільки поточний проект контролює як апаратну, так і програмну частину AIP, потрібно вибрати мережевий протокол і протокол мережевого зв’язку. З літератури, майже нічого не сказано про використовувані мережеві протоколи зв'язку. Два основних пояснення полягають у тому, що або розроблена система є одновузловою, або використовуваний мережевий протокол також містить протокол зв’язку за замовчуванням (наприклад, Zigbee, Z-Wave тощо).

Основна тенденція, виведена з літератури, полягає в тому, що домінуючим мережевим протоколом є Wi-Fi, а єдиним згаданим протоколом зв’язку є MQTT. Для цієї магістерської роботи вибір мережевого протоколу досить простий. Зважаючи на те, що розробки відбуваються в обмеженому середовищі, використання мережі 4G або 5G LTE не є потрібним.

Крім того, обидва протоколи Zigbee і Z-Wave мають низьку енергоємність і низьку швидкість передачі даних, і цей проект може потребувати великої пропускної здатності для передачі зображень або даних хмари точок. Таким чином, найкращим вибором є використання мережевого протоколу Wi-Fi. На жаль, він не містить протоколу двонаправленого зв’язку за замовчуванням, тому потрібно вибрати конкретний.

Доступні кілька варіантів, але лише ці три протоколи зв’язку цікаві для цього проекту: HTTP(s), MQTT і WS(s) (Websockets). Вибір MQTT – це простий вибір, він ідеально підійде для поточних потреб. Однак контроль над автоматизацією сервера (брокера MQTT) неможливий. Вибір HTTP(s) дозволяє налаштувати автоматизацію сервера, але був розроблений для одностороннього зв’язку. Це правда, що можна здійснювати двонаправлений обмін даними через HTTP(s), але останній вибір кращий.

Найкращим вибором тут є мережевий протокол WS(s). Насправді він використовує веб-сокети для створення двонаправленого каналу зв’язку між

двома клієнтами. Завдяки цьому функціонуванню не потрібен сервер (за винятком відкриття з'єднання, яке використовує систему рукостискання HTTP(s), а автоматизацію всіх клієнтів можна налаштувати. Остаточним вибором для мережевого стека, таким чином, є протокол зв'язку WS(s).

Ці варіанти можна побачити на рис. 2.6. По-четверте, щодо технології БД, поряд із вибором форматів даних, необхідно зробити два основні вибори: формат даних, який використовується в усьому проекті (від електронного пристрою до налаштувань користувача), і система зберігання даних (наприклад, база даних, озеро даних). тощо). У вже проаналізованих статтях абсолютно нічого не сказано про використовуваний формат даних.

Насправді жодному іншому проекту не потрібно було про це піклуватися, оскільки жоден із них не спілкується через веб-сокети. Щодо системи зберігання даних, певна інформація міститься в наукових статтях, де розглядаються системні архітектури та хмарні інфраструктури, але вони тут не цікаві, оскільки цей проект розробки AIP не використовує ці попередньо визначені архітектури та інфраструктури.

Вибір формату даних досить простий. Переглядаючи вимоги до системи, можна побачити, що весь обмін інформацією в рамках проекту вимагав деякої інформації про походження повідомлення, призначення, вміст тощо. Це тому, що використання мережевого протоколу зв'язку WS змушує розробити система маршрутизації повідомлень для передачі повідомлень від вузла до вузла. Таким чином, очевидним вибором є використання формату даних JSON, оскільки він структурований, типізований і добре інтегрується з іншими варіантами програмного забезпечення. Що стосується вибору системи зберігання даних, доступні три основні концепції: бази даних, озера даних (або океани даних) і сховища даних. БД — це набір даних, який дозволяє системі зберігати, аналізувати та взаємодіяти з даними. Існує два типи БД (тобто реляційна та нереляційна), але обидва можуть бути структурованими або напівструктурованими відповідно до збережених даних.

Сховище даних — це вдосконалена система БД, яка зберігає високоструктуровану інформацію з різних різнорідних джерел і оптимізована для аналітичних операцій. Озеро даних — це сховище даних, яке зберігає дані в оригінальному необробленому форматі. Щоб вибрати одну з цих трьох систем, найкраще проаналізувати розроблений AIP.

У цьому випадку, враховуючи, що джерело даних зібрано разом у системі вузла AIP, і враховуючи, що використовуваний формат даних стандартизований для веб-сокетів, найкращим вибором буде використання нереляційної бази даних, такої як MongoDB14. Крім того, цей вибір дозволяє використовувати ту саму систему зберігання даних на всіх вузлах розробленого AIP, що забезпечує надійність AIP, дозволяючи тимчасово зберігати дані в локальному вузлі, якщо трапляється збій мережі.

Остаточним вибором є використання формату даних JSON із базою даних MongoDB для кожного вузла AIP. Ці рішення можна побачити на рис. 2.6. Щоб завершити цей розділ про вибір програмного забезпечення, слід пояснити додатковий факт. Що стосується апаратних рішень, то жодна популярна нині система не могла бути неправильним вибором. Рішення, прийняті в цьому розділі, очевидно, є результатом глибоких досліджень і, таким чином, є найкращим можливим вибором для поточного розвитку, але вибір протоколу HTTP(s) замість протоколу WS(s) не порушить систему.

Це пояснюється тим, що більшість доступних на даний момент платформ можна зламати, щоб змусити їх працювати належним чином, навіть якщо вони не призначені для виконання деяких операцій. Єдиним реальним досі актуальним обмеженням при виборі програмних компонентів є апаратна сумісність. Зручність широкої сумісності платформи RPI також є однією з причин її вибору.

3. СПЕЦІАЛЬНА ЧАСТИНА

3.1. Розробка апаратного забезпечення

У цьому розділі детально розглядаються апаратні вузли. Це означає, що побудований Proof Of Concept (POC) деталізований і показаний більш просунутий дизайн. Цю додаткову апаратну систему вивчали, але не створювали під час розробки апаратної частини AIP.

Перш ніж переглядати зроблені апаратні розробки, важливо зробити коментар. Як зазначено у вступі (Розділ 1), ця магістерська робота зосереджена на стороні ІКТ SPF. Ця спрямованість впливає на розроблені електронні системи. Насправді, бачачи, що основною метою SPF є виробництво рослин, забезпечуючи їм найкраще можливе середовище, електронні системи повинні бути корисними для досягнення цієї мети. Однак через фокус цього проекту актуальність кожного апаратного компонента глибоко не вивчена.

Для реалізації апаратної частини AIP було побудовано POC електронної системи. Ця збірка має на меті виконати всі основні вимоги AIP. Крім того, будучи першою розробленою системою, ця POC також має на меті перевірити зроблений вибір апаратного забезпечення. На рис 3.1 представлена схема електричних з’єднань цього першого електронного вузла, а на рис 3.2 показана схема з’єднання цього самого вузла. Як видно на цих схемах, два Raspberry Pis (RPI) мають спільні давачі та виконавчі механізми. Це не через обмеження обчислювальної потужності чи відсутність контактів введення/виведення загального призначення (GPIO), а через фізичну відстань між різними пристроями.

Насправді давачі води та реле розташовані біля резервуару для води, а інші давачі – біля лотка для культивування. Це зауваження важливе, оскільки означає, що апаратне забезпечення має бути модульним. Крім того, як буде

видно в огляді програмного забезпечення, ця модульність повинна бути (і є) у програмній частині AIP.

Використані одноплатні комп'ютери (SBC) – це Raspberry Pi Zero 2W. Перший пов'язаний з 2 давачами вологості та температури, 2 водонепроникними давачами температури та декількома давачами, прикріпленими зверху (HAT), які мають давач температури та вологості, а також детектор інтенсивності світла та давач барометричного тиску.

Другий використовується для керування 4 реле HAT і 8 каналами аналого-цифрового перетворювача (АЦП) HAT. Система освітлення 220 В змінного струму та насос поживного розчину підключені через 4 реле HAT, а 3 аналогові давачі якості води підключаються через 8 каналів ADC HAT.

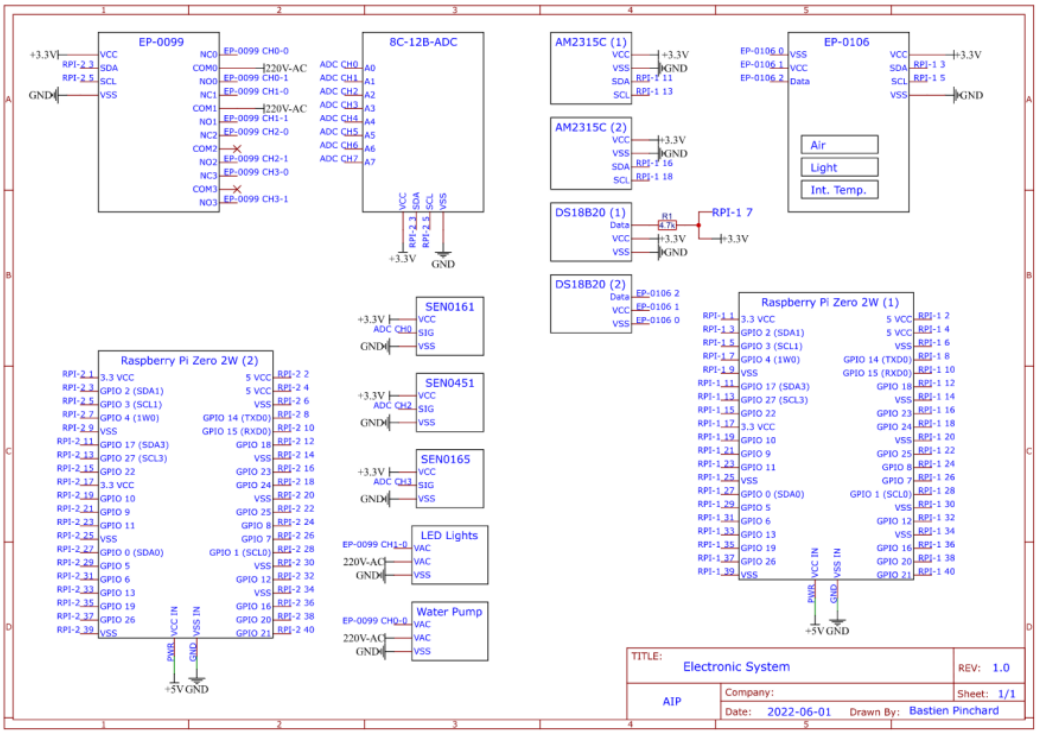


Рисунок 3.1 - Схема електричних з'єднань апаратного вузла РОС, що реалізує електронну систему AIP (апаратна сторона AIP).

Давачі вологості та температури (AM2315C) є цифровими та використовують протокол зв'язку I2C. Їх адреса за замовчуванням 0x38, і її не можна змінити, це означає, що потрібно використовувати кілька каналів I²C або мультиплексор I²C. Водонепроникні давачі температури (DS18B20,

рис. С.2) є цифровими та використовують протокол зв’язку за 1 дротом. Необхідно використовувати підтягуючий резистор 4,7 К, але кілька таких давачів можна об’єднати разом і підключити до одного контакту вводу/виводу. Кілька давачів HAT (EP-0106) використовує цифровий протокол зв’язку I²C із незмінною адресою за замовчуванням 0x17. Корисні давачі, присутні на цій платі: вбудована температура та вологість, барометр (з температурою та вологістю), рівень освітленості та зовнішня водонепроникна температура. 4 реле HAT (EP-0099) також використовує цифровий протокол зв’язку I²C із діапазоном адрес за замовчуванням від 0x10 до 0x13. Це означає, що 4 HAT можна об’єднати в стек для керування до 16 реле.

Ця система дуже зручна з боку програмування, але не дуже довговічна з боку електроніки. Це пов’язано з тим, що гвинти недоступні, коли HAT встановлено разом з іншим обладнанням. 8-канальний ADC HAT (8C-12B-ADC) також використовує цифровий протокол зв’язку I²C із незмінною адресою за замовчуванням 0x04. Він має 12-бітну точність, внутрішній годинник реального часу, і його можна використовувати в режимі порівняння для порівняння результатів двох каналів.

Давач pH (SEN0161) — це аналоговий давач, який включає дочірню плату, яка перетворює вимірювання давача в аналоговий сигнал. Цей зонд якості води кількісно визначає потенціал водню (pH) розчину, у який він занурений. Цей зонд потрібно відкалібрувати перед використанням і його вимірювання можуть дрейфувати понаднормово. EC-зонд (SEN0451) також є аналоговим давачом, який включає подібну дочірню плату з тією ж функціональністю, що й дочірня плата pH-зонда. Цей зонд якості води кількісно визначає електропровідність (EC) розчину, у який він занурений. Цей зонд також потрібно відкалібрувати перед використанням, тому його вимірювання з часом можуть відхилятися.

Давач ОВП (SEN0165) подібний до двох попередніх давачів: він аналоговий і постачається з подібною дочірньою платою. Крім того, його

також потрібно відкалібрувати, і його вимірювання можуть з часом змінюватися. Цей зонд якості води вимірює окислювально-відновний потенціал (ОВП) розчину, у який він занурений.

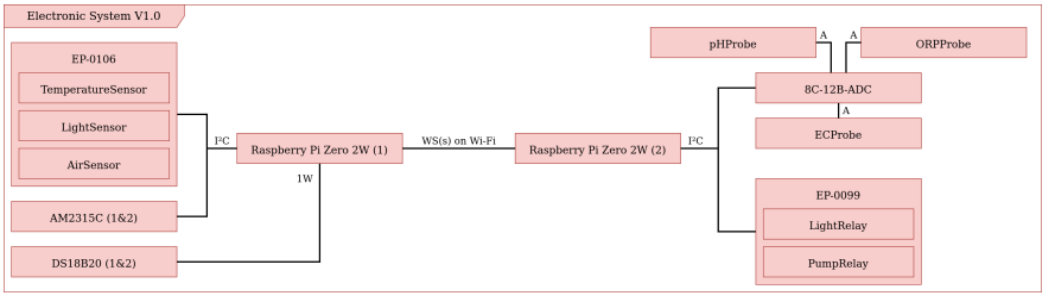


Рисунок 3.2 - Схема підключення апаратного блоку POC, що реалізує електронну систему AIP (апаратна сторона AIP).

Використаний протокол зв'язку відображається для кожного з'єднання: A (аналоговий), D (цифровий), I²C, ШІМ або 1 Вт. Цілі електронних систем, які складають апаратну частину AIP, полягають у тому, щоб якомога більше усунути взаємодію людини та фізичної людини з CSM та SPF. Для цього електронна система збирає дані про середовище, щоб мати можливість перевірити його оптимальність і змінити його стан. З цієї причини, чим більш модульна апаратна система, тим вона краща, тому що нові вимірювання та елементи керування завжди будуть реалізовані в AIP. Поточна система не має повних функцій, але є модульною.

Вже реалізовані системи контролю: управління потоком води, активність освітлення (увімкнення/вимкнення), вимірювання якості води та стан навколишнього середовища. Наступними цілями для системи контролю є: управління освітленням (увімкнення/вимкнення, рівень затемнення та температура Кельвіна), управління якістю води (вимірювання та модифікації) та управління навколишнім середовищем (вимірювання та модифікації). Іншим недоліком поточного POC є використання HAT.

Навіть якщо вони дійсно корисні для програмування та крафтової сторони реалізації проекту, вони не є ідеальними компонентами, оскільки їм бракує важливої модульності. Насправді використовувані HAT є дуже

гнучкими (що важливо при впровадженні першого РОС), але не є модульними. Наприклад, АЦП постачається з вбудованим мікроконтролером (наприклад, STM32F030F4P6TR або MM32F031F6P6).

Наявність цього мікроконтролера робить НАТ дійсно гнучким, роблячи його інтерфейс доступним, але його обчислювальні можливості марні для цього проекту, оскільки використання мікропроцесорів було віддано перевагу. Крім того, той самий НАТ не є модульним, тому що кілька АЦП не можуть використовуватися в одній системі (оскільки форм-фактор збірки не дозволяє використовувати мультиплексор I^2C і оскільки адреси I^2C АЦП не можна змінювати).

Реальність така, що, як і для всіх інших давачів, приводів і зондів, ці компоненти були обрані для розробки апаратної збірки РОС через їх доступність (стосовно ціни, необхідної пайки тощо) і гнучкості. Це причина, чому деякі вдосконалені конструкції показані нижче. Однак навіть із цими недоліками ці апаратні компоненти ідеально підходять для розробки поточного РОС, який підтримує апаратну сторону AIP.

3.2. Удосконалена конструкція.

Щоб подолати обмеження збірки апаратного забезпечення РОС, була розроблена більш досконала концепція системи керування. Як було сказано раніше, цілі, які повинна виконувати ця краща система контролю, це: управління освітленням (рівень затемнення та температура Кельвіна), управління якістю води (вимірювання та модифікації) та управління навколишнім середовищем (вимірювання та модифікації).

Очевидно, що системи моніторингу, представлені в попередній спробі, слід зберегти. Ці системи моніторингу включають: керування потоком води (вмикання/вимкнення), активність освітлення (увімкнення/вимкнення), вимірювання якості води (наприклад, pH, EC, ORP тощо) і стан навколишнього середовища (наприклад, температура повітря). і вологість

тощо). Більше того, ця передова концепція має усунути обмеження модульності першого POC, не використовуючи HAT із надлишковими мікроконтролерами та розробляючи модульну збірку.

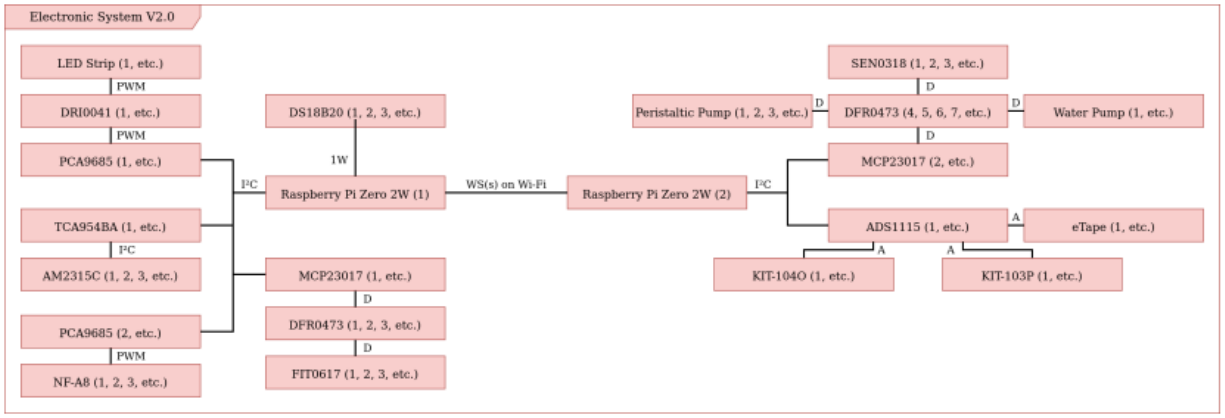


Рисунок 3.3 - Схема підключення вдосконаленого апаратного вузла для майбутньої версії електронної системи AIP (апаратна сторона AIP).

Використаний протокол зв’язку відображається для кожного з’єднання: A (аналоговий), D (цифровий), I²C, ШІМ або 1 Вт. На рис. 3.4, 3.5 і 3.6 представлені схеми підключення цієї нової електронної системи, а на рис. 3.3 показана схема з'єднання того самого вузла. Як видно на цих малюнках, основний акцент був зроблений на розширюваності апаратного забезпечення та модульності.

Насправді, на додаток до того факту, що в цій другій версії використовується набагато більше електронних компонентів, потрібно менше контактів RPI GPIO і відсутні конфлікти адрес I²C (таким чином усувається необхідність використання кількох шин I²C). Ці особливості та реалізовані функції розглядаються в наступних параграфах.

Стосовно першої сторінки цієї нової схеми з’єднання електронної системи (рис 3.4), можна знайти три основні вдосконалення на додаток до RPI, який уже використовувався в першій системі.

По-перше, водонепроникні давачі температури (DS18B20) все ще присутні, але з’єднані разом завдяки використанню шини 1 Вт. Подібним чином давачі температури та вологості (AM2315C) також присутні, але

підключені до мультиплексора I²C (TCA9548A), який дозволяє використовувати кілька пристроїв з однаковою адресою.

Третє вдосконалення стосується системи освітлення. Ця нова електронна система використовує ШІМ-драйвер (PCA9685) і драйвер двигуна постійного струму (DRI0041) для керування та модуляції деяких регульованих світлодіодних стрічок.

Це освітлювальне обладнання здатне контролювати інтенсивність світла завдяки використанню ШІМ та температуру світла завдяки двоканальній конструкції світлодіодної стрічки. На другій сторінці схеми електропроводки нової електронної системи (рис. 3.5) помітні дві особливості. По-перше, повторне використання драйвера ШІМ (PCA9685) у поєднанні з вентиляторами дозволяє системі трохи більше контролювати середовище.

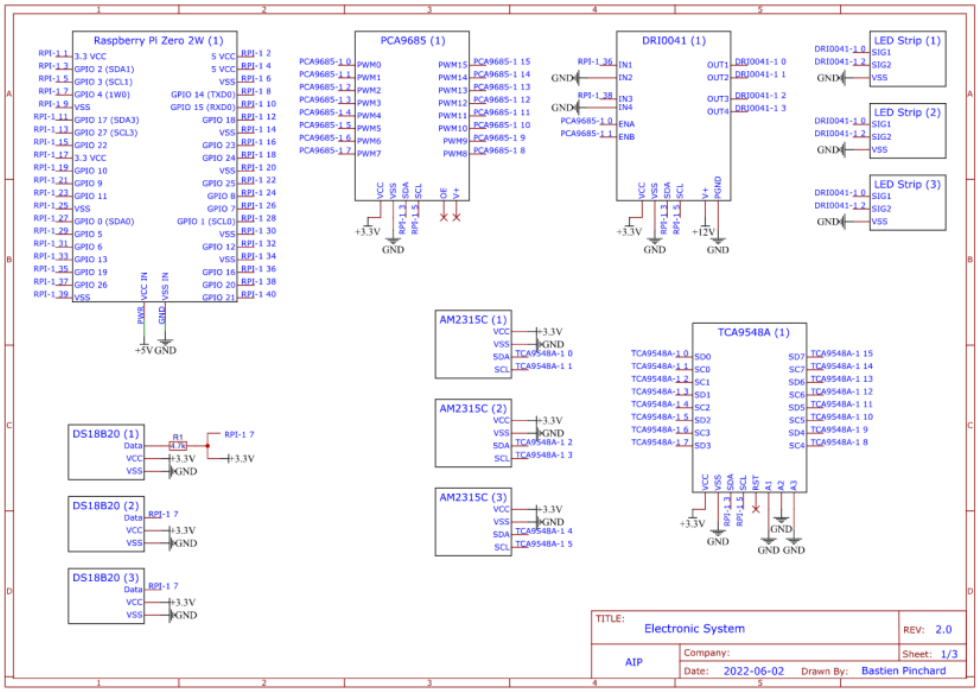


Рисунок 3.4 - Перша сторінка схеми електричних з’єднань розширеного обладнання для майбутньої версії електронної системи AIP (апаратна сторона AIP).

Можна уявити собі використання цієї системи в закритій камері культивування, щоб почати краще контролювати культурне середовище. Потім, друга особливість стосується управління потоком води.

Додавання електромагнітних клапанів (FIT0617), керованих за допомогою силових реле (DFR0473) і GPIO.

Розширювач I²C (MCP23017) робить цю систему більш модульною щодо розподілу поживного розчину. На рис. 3.6 представлена третя й остання сторінка схеми підключення нової електронної системи. Ця остання частина містить три цікаві моменти. На додаток до використання другого RPI, як у оригінальній конструкції, повторне використання розширювача GPIO I²C (MCP23017) для керування реле живлення (DFR0473), які самі керують перистальтичними насосами і водяний насос дозволяє цій новій електронній системі дозувати деякі поживні суміші для регулювання поживного розчину.

Крім того, використання зовнішнього давача рівня рідини (SEN0368) робить контроль запасів поживних сумішей зручним.

По-третє, останнім цікавим пунктом цієї сторінки зі схемою підключення є включення I²C АЦП (ADS1115) для контролю вимірювань, зроблених давачами якості води (KIT-103P, KIT-104O) і занурений давач рівня рідини.

На завершення цього огляду передового електронного дизайну заслуговують на увагу два основні моменти. Перше стосується можливості розширення системи. Як видно на рис. 3.4, 3.5 і 3.6, кожен компонент позначено індексом у дужках. Це не є незначним, оскільки це означає, що було вивчено використання кількох компонентів для кожного пристрою.

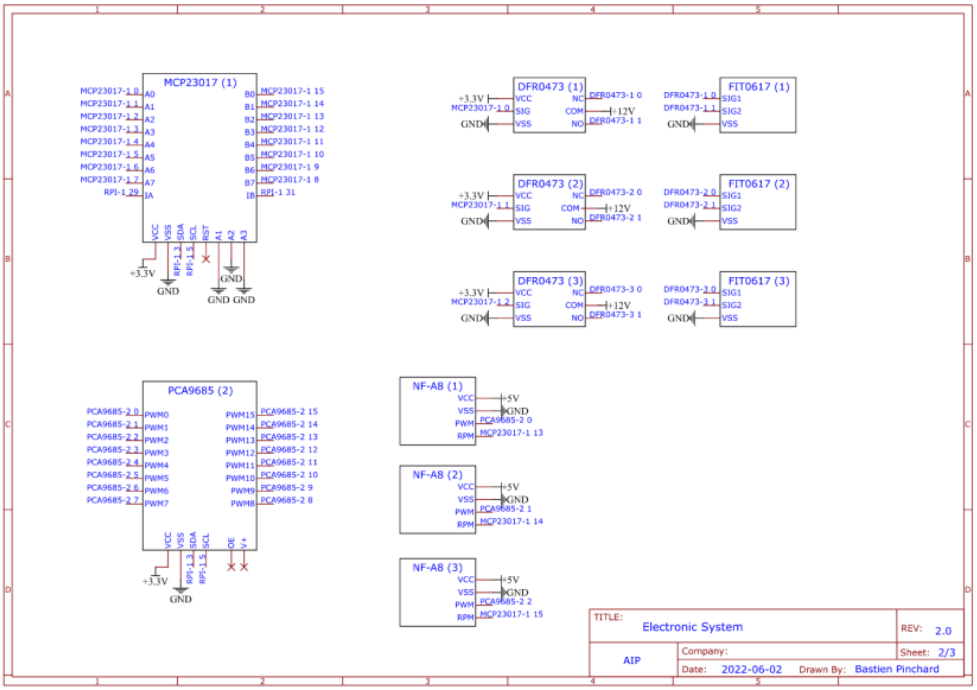
Завдяки тому факту, що всі «плати контролерів» (тобто мультиплексор I²C, драйвер ШІМ, розширювач GPIO I²C і АЦП I²C) використовують протокол I²C і мають настроювану адресу, кожну з них можна об’єднати разом і використовувати в той самий час. Це важлива функція, оскільки вона дозволяє розширювати розроблені на даний момент системи.

Другий цікавий аспект цього передового електронного дизайну полягає в його модульності. Насправді, завдяки можливості розширення багато

контактів GPIO все ще доступні для додавання нових функцій до системи керування.

Крім того, всі підсистеми (наприклад, управління освітленням) незалежні одна від одної. Таким чином, можна скласти ідеальну систему для кожної культуральної камери (наприклад, можна додати модуль температури та вологості першого RPI з першої сторінки схеми підключення до другого RPI з третьої сторінки тієї ж схеми).

Нарешті, як бонус, якщо виникне деяка відсутність контактів GPIO, можливе використання розширювача GPIO I²C (оскільки його зроблено для керування силовим реле). Крім того, для інтерфейсу деяких пристроїв 5 В з RPI 3,3 В можна використовувати безпечний двонаправлений перетворювач логічного рівня I²C (BSS138).



Рсунок 3.5 - Друга сторінка схеми електричних з’єднань розширеного апаратного вузла для майбутньої версії електронної системи AIP (апаратна сторона AIP).

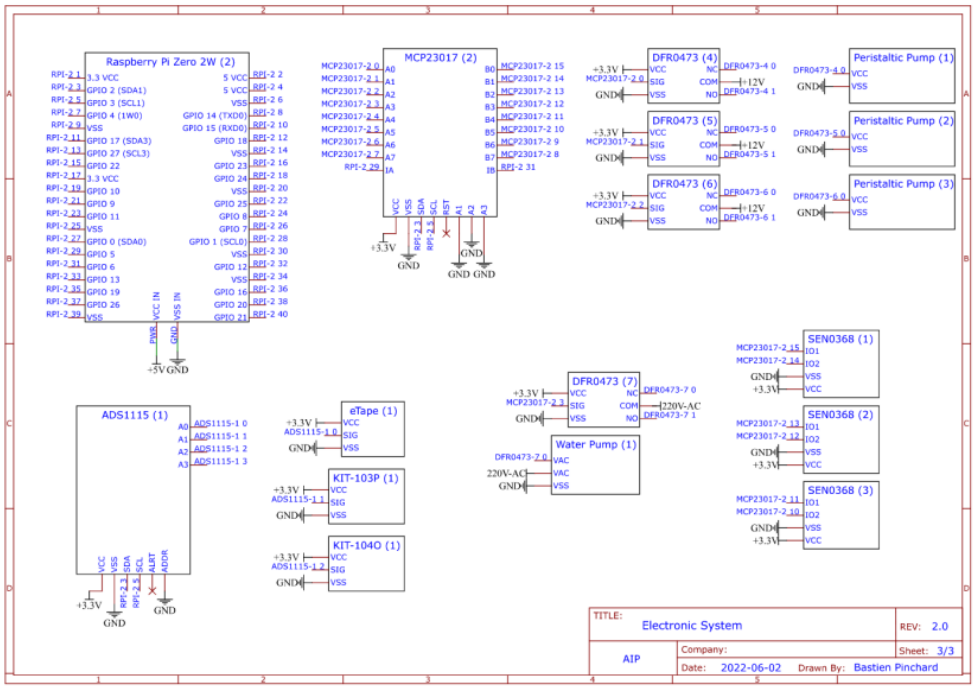


Рисунок 3.6 - Третя сторінка схеми електричних з’єднань розширеного апаратного вузла для майбутньої версії електронної системи AIP (апаратна сторона AIP)

3.3. Розробка програмного забезпечення.

Електронна система управління (EMS) і панель управління (MD) є двома доповнювальними програмними компонентами, які реалізують програмну частину AIP. Цей розділ присвячено огляду мінімально життєздатного продукту (MVP) керуючого програмного забезпечення. Обидві сторони цього MVP детально описуються з точки зору організації високого рівня та впровадження низького рівня.

Навіть якщо AIP має мати систему керування людиною (тут, MD), EMS є найоригінальнішим програмним забезпеченням у цій роботі. Насправді EMS містить ключові концепції AIP (наприклад, визначення середовища, алгоритми автоматизації тощо). На рис. 3.7 показано характеристики та взаємодію між цими програмними компонентами.

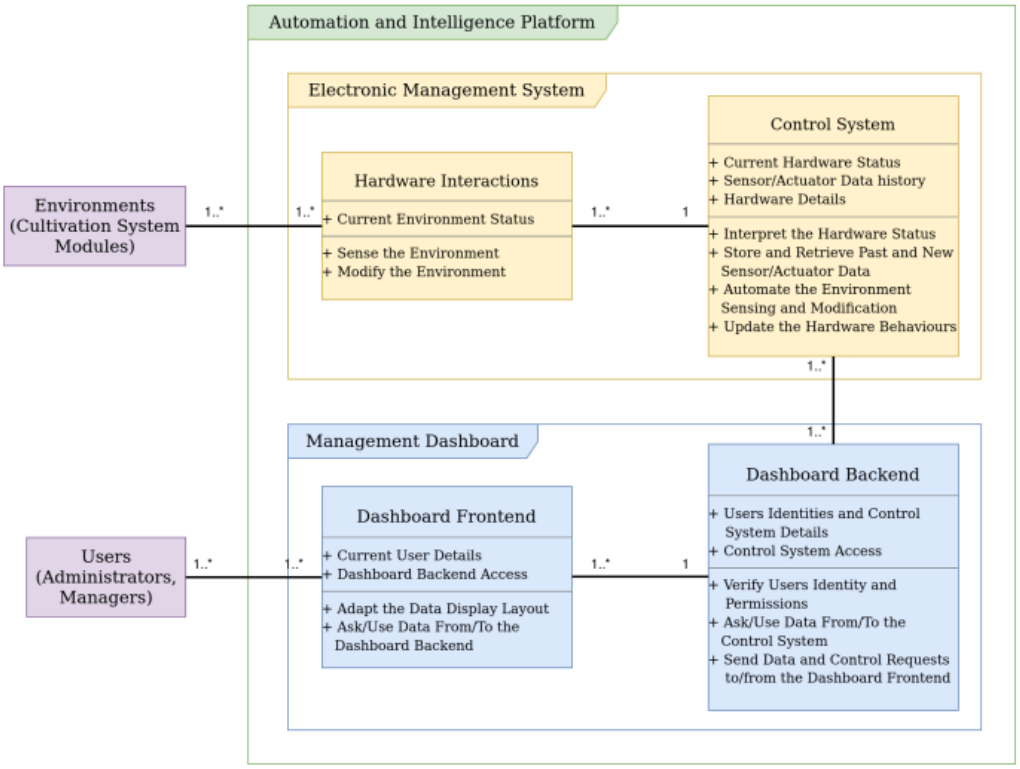


Рисунок 3.7 - Високорівнева діаграма компонентів програмного забезпечення AIP.

EMS та її компоненти позначено жовтим кольором, MD та його інтерфейсна/завершальна частини представлені синім кольором, сума цих двох систем, що склала AIP, позначена зеленим кольором, а зовнішні елементи позначено фіолетовим кольором. Однією цікавою особливістю програмної сторони цього AIP є те, що він був створений для підтримки змінної кількості CSM і кількох користувачів (тобто людей, які керують Smart Plant Factory). Крім того, кілька SPF можуть працювати паралельно, і один користувач може мати доступ до кожного SPF на MD.

Крім того, система керування та серверна панель приладів можуть відповідно обслуговувати декілька апаратних компонентів та інтерфейс панелі приладів. Це стало можливим завдяки модульності забезпечується об’єктно-орієнтованим підходом, який використовується як для EMS, так і для MD.

Електронна система управління (EMS) складається з двох основних компонентів: системи керування та програмного забезпечення взаємодії апаратного забезпечення. Перш ніж описувати кожну реалізацію, можна перерахувати основні технічні деталі.

Стандартний модуль abc для абстрактних класів. Модулі argparse та configparser для аналізу аргументів командного рядка та аналізу конфігураційного файлу. Унікальна база даних MongoDB (модуль pymongo) із колекцією для кожного реалізованого пристрою та модулем json для ефективного аналізу зв'язку. Модуль asyncio, щоб зробити виконання асинхронним. Модуль websockets для обробки двонаправленого зв'язку.

На додаток до цих варіантів було використано дві концепції програмування для покращення модульності системи та підтримки об'єктно-орієнтованого підходу. Ці дві парадигми програмування — це абстракція та успадкування.

У цьому проекті ці дві концепції використовувалися разом для створення класів (тобто вихідного коду взаємодії апаратного забезпечення), які побудовані на одному базовому класі (тобто вихідному коді системи керування) для визначення нових функціональних можливостей, зберігаючи ту саму базову поведінку.

Алгоритми автоматизації AIP і системи когнітивних обчислень можуть надходити з двох різних джерел: хмари або периферійних обчислень. У цьому проекті це EMS, який відповідає за з'єднання електронних компонентів разом із хмарою (якщо така є). Завдяки цьому спеціальному EMS, AIP, розроблений під час цієї магістерської роботи, сумісний з двома парадигмами: він може досягати хмари, щоб запитати, які дії виконати, і він може обчислювати свою наступну дію незалежно завдяки периферійним обчисленням.

Як показано далі, система керування відповідає за обидві парадигми, тоді як вихідний код взаємодії апаратного забезпечення стосується лише периферійних обчислень.

Система керування є найцікавішою частиною програмного забезпечення, оскільки вона була розроблена з нуля для підтримки AIP. Як видно на рис 3.8, система складається із семи абстрактних класів із точним деревом успадкування. Ця специфічна структура робить вихідний код більш компактним, придатним для багаторазового використання та гнучким.

Кожен клас відповідає за поставлені завдання, а реалізації взаємодії апаратного забезпечення значно вдосконалені та легші для розробки завдяки абстрактним класам найнижчого рівня (тобто VSensor і VActuator).

Крім того, чотири з цих семи абстрактних класів контролюють міжкомпонентний двонаправлений зв’язок. Їх можна побачити в помаранчевих прямокутниках на рис. 3.8.

Мережевий зв’язок є важливою частиною EMS, оскільки він дозволяє використовувати декілька апаратних систем, на яких програмні компоненти можуть бути розгорнутими. Крім того, кожен із цих чотирьох компонентів представляє окремий рівень керування, на якому можна аналізувати, агрегувати, перевіряти та передавати інформацію.

Таким чином, ця структура робить EMS дуже модульною та дозволяє точно контролювати апаратні компоненти. Нарешті, дивлячись на рис 3.8 чотири класи, відповідальні за зв’язок, мають таку саму структуру, що й чотири основні компоненти SPF.

Ця подібність потрібна для забезпечення однорідності системи, і це показує, що максимальний рівень керованості доступний з цією структурою EMS.

Ці чотири класи (наприклад, VHuman, VFarm, VCsm і VDevice) не є єдиними доступними рівнями для прийняття рішень.

Фактично, загальний базовий клас (тобто VClass) також можна використовувати для реалізації глобальної системи прийняття рішень, а два дочірні класи (тобто VSensor і VActuator) також мають однакові можливості. Ці локальні обчислювальні потужності часто називають периферійними обчисленнями.

Однак, як згадувалося вище, ці класи також сумісні з парадигмою хмарних обчислень.

Насправді, завдяки класу VClass можна було легко налаштувати HTTP-з'єднання з хмарою та надати доступ до кожного пристрою на основі дочірніх класів. Як примітка, схема іменування, яка використовується в EMS, досить проста: кожна назва класу має префікс «V» (що означає «Віртуальний»), щоб вказати на те, що вони є абстрактними класами, і суфікс класу імена посилаються на компонент, який вони обслуговують (наприклад, клас VFarm — це абстрактний клас, створений для об'єкта керування SPF).

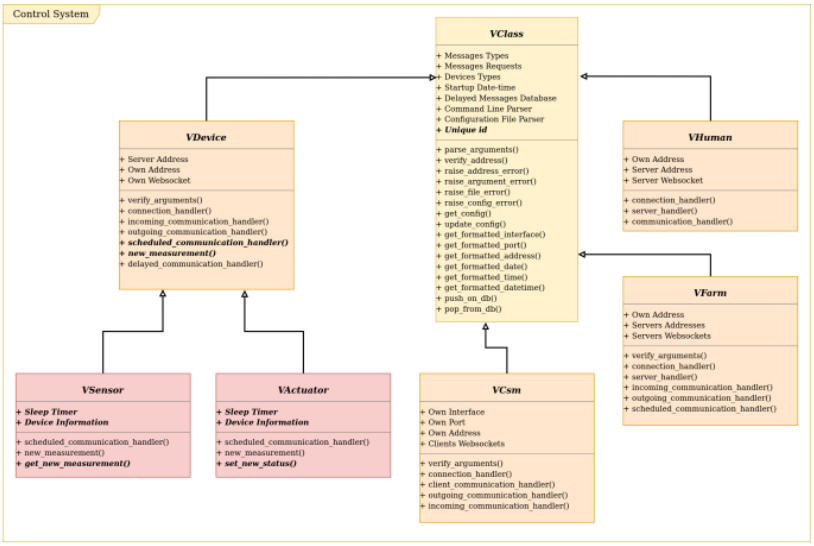


Рисунок 3.8 - Діаграма класів електронної системи керування, яка входить до складу EMS.

Рівні успадкування позначені кольором від жовтого до червоного (від чистого програмного забезпечення до програмного забезпечення, яке взаємодіє з апаратним забезпеченням), а абстрактні компоненти написані жирним курсивом. Абстрактний клас VClass є базовим класом для кожного пристрою EMS. Цей клас є абстрактним, оскільки його прямі дочірні класи є абстрактними (наприклад, VDevice, VCsm, VFarm, VHuman) і тому, що він вимагає від них реалізації поля ідентифікації пристрою.

Клас VClass реалізує аналізатор командного рядка, аналізатор конфігураційного файлу, тимчасову базу даних MongoDB і деякі службові

функції. Усі ці функції забезпечують загальний інтерфейс для кожного класу пристроїв. Ця стандартизація гарантує, що кожна система побудована на однакових засадах і що всі вони взаємосумісні.

Наприклад, визначення синтаксичного аналізатора командного рядка та аналізатора конфігураційного файлу забезпечують дві важливі функції для кожного підкласу. Крім того, визначення тимчасової бази даних і її функцій push/pop гарантує, що кожен підклас може мати доступ до такої бази даних у разі збою мережі (тобто зберігати повідомлення та надсилати їх, як тільки буде встановлено з'єднання з мережею). Особливістю цього класу VClass є те, що змінні класу використовуються для визначення макросів для мережевих комунікацій.

Ці макроси використовуються всіма дочірніми класами та забезпечують ефективний розбір повідомлень websocket і пов'язують кодову базу шляхом зміцнення логіки. Ці макроси стосуються типу повідомлення та запитів, але також визначають доступні типи пристроїв. Усі вищезазначені деталі можна побачити в псевдокоді, наданому в лістингу 3.1.

Клас VDevice безпосередньо успадковує абстрактний клас VClass. Цей клас VDevice також є абстрактним, оскільки він визначає два абстрактні методи, які мають бути реалізовані в прямих дочірніх класах (VSensor, VActuators тощо). Ці два методи забезпечують загальний інтерфейс для запиту нового вимірювання та обробки запланованих зв'язків.

Перший необхідний для стандартизації дії отримання нового вимірювання, оскільки давачі та виконавчі механізми мають протилежне функціонування: виконавчі механізми не можуть виконувати вимірювання, але можуть змінювати свій стан, а давачі не можуть змінювати свій стан, але можуть вимірювати своє середовище. Другий абстрактний метод також необхідний для того, щоб кожен дитячий клас міг визначити власний розклад надсилання нових даних і те, які дані потрібно надсилати.

```
# Libraries includes
import pymongo # For the database
import datetime # For time formatting
import argparse # For command line arguments
import configparser # For the configuration file

# Classes includes
from abc import ABC # For abstract class error management

class VClass(ABC):
    def __init__(self) -> None:
        # Set-up the basic class variables
        self.startup_datetime = None
        self.parser = None
        self.config = None
        self.delayed_db = None
        # Define the message types
        self.GET = 'GET'
        self.POST = 'POST'
        # Define the message requests
        self.NEW_MEASUREMENT = 'NEW MEASUREMENT'
        self.SCHEDULED_MEASUREMENT = 'SCHEDULED MEASUREMENT'
        self.DEVICE_INFO = 'DEVICE INFO'
        self.DEVICE_CONFIGURATION = 'DEVICE CONFIGURATION'
        self.DEVICE_CALIBRATION = 'DEVICE CALIBRATION'
        # Define the device types
        self.SENSOR = 'SENSOR'
        self.ACTUATOR = 'ACTUATOR'
        self.FARM = 'FARM'
        self.CSM = 'CSM'
        self.HUMAN = 'HUMAN'

    def __call__(self) -> None:
        self.parse_arguments()
        self.startup_datetime = self.set_datetime()

    def parse_arguments(self) -> None:
        # Define the command line parser
    # Verify the user-given addresses

    def verify_address(self, server_address) -> bool:
        # Verify if the address match a specific pattern

    def raise_errors(self, **kwargs) -> None:
        # Raise the wanted error for the parser, config-parser, address or file

    def get_config(self, file_path) -> dict:
        # Parse, evaluate, and return a configuration file

    def update_config(self, file_path, fields) -> None:
        # Update a configuration file

    def get_formatted_interface(self, address) -> str:
        # Format and return the interface number

    def get_formatted_port(self, address) -> str:
        # Format and return the port number

    def get_formatted_address(self, interface, port) -> str:
        # Format and return the websocket address

    def get_formatted_date(self) -> str:
        # Format and return the current date

    def get_formatted_time(self) -> str:
        # Format and return the current time

    def get_formatted_datetime(self) -> str:
        # Format and return the date and time

    def push_on_db(self, message) -> pymongo:
        # Push a new record on the object database (create the database if necessary)

    def pop_from_db(self) -> dict:
        # Pop the first message pushed on the database (signal when the database is/becomes empty)
```

Лістинг 3.1 - Псевдокод абстрактного класу VClass

Клас VDevice також реалізує керування веб-сокетами та обробники зв’язку завдяки асинхронним примітивам (тобто async і await). Для мережевого компонента клас VDevice — це клієнт, який може надсилати й отримувати повідомлення, підключаючись до сервера.

Крім того, цей клієнт може справлятися з помилками мережі та автоматично повторно підключатися до сервера, коли це доступно. Ці комунікації керуються асинхронно та в різних обробниках завдань. Усі ці особливості можна побачити в псевдокоді, представленому в лістингу 6.2.

```python
# Libraries includes
import abc # For abstract methods
import asyncio # For asynchronous communication handling
import websockets # For bidirectional communications

# Custom classes includes
from vclass import VClass # For inheritance

class VDevice(VClass):
    def __init__(self) -> None:
        super().__init__()
        # Define the network addresses
        self.server_address = ''
        self.own_address = ''
        # Prepare the client web-socket creation
        self.own_websocket = None

    def __call__(self) -> None:
        try:
            # Launch the infinite loop
            super().__call__()
            self.verify_arguments()
            asyncio.run(self.connection_handler())
        except KeyboardInterrupt:
            # Cleanly stop the infinite loop

    def verify_arguments(self) -> None:
        # Verify the provided command line arguments and raise errors if needed

    async def connection_handler(self) -> None:
        # Connect to the server (CSM) (with automatic reconnection)
        async for self.own_websocket in websockets.connect(self.server_address):
            # Launch the network listening and message sending
            task1 = asyncio.create_task(self.incoming_communication_handler())
            task2 = asyncio.create_task(self.scheduled_communication_handler())
            task3 = asyncio.create_task(self.delayed_communication_handler())
            tasks = [task1, task2, task3]
            try:
                await asyncio.gather(*tasks)
            except websockets.ConnectionClosed:
                # Clean the connection closing and try to reconnect

    async def incoming_communication_handler(self) -> None:
        async for message in self.own_websocket:
            # Analyze the incoming communications and execute the correct followup

    async def outgoing_communication_handler(self, message) -> None:
        if self.own_websocket:
            # Send the message to the correct destination
            await self.own_websocket.send(json.dumps(message))
        else:
            # Store the message if the connection is unavailable
            self.push_on_db(message)

    @abc.abstractmethod
    async def scheduled_communication_handler(self) -> None:
        # Abstract interface to periodically send messages

    @abc.abstractmethod
    async def new_measurement(self) -> json:
        # Abstract interface for asking a new measurement

    async def delayed_communication_handler(self) -> None:
        # Send the messages stored during the connection loss
        while message is not None:
            await self.outgoing_communication_handler(message)
            message = self.pop_from_db()
```

Лістинг 6.2: Псевдокод абстрактного класу VDevice

Клас VCsm є останнім абстрактним класом, який використовується для реалізації концепції Cultivation System Module. Цей клас обробляє керування веб-сокетами та керує зв’язками. Що стосується класу VDevice, комунікації управляються асинхронно та в різних обробниках завдань. Щодо сторони мережевих комунікацій, клас VCsm діє як сервер, який приймає підключення та маршрутизує зв’язок, якщо це необхідно.

Крім того, цей сервер керує своїми відомими клієнтами та може справлятися з перебоями в мережі. Система керування клієнтами створена зі словником, щоб відстежувати веб-сокети клієнтів разом із їхньою

інформацією. За допомогою цього словника сервер може аналізувати інформацію про клієнтів, щоб знати, якому клієнту(ам) слід надсилати повідомлення. Завдяки цим функціям і базовим утилітам VClass базова поведінка CSM можна успадкувати від цього класу для будь-яких користувацьких реалізацій CSM. Як і для всіх абстрактних класів EMS, цей клас VCsm може бути доповнений алгоритмом автоматизації та системами аналізу даних.

Клас VFarm також є останнім абстрактним класом, який використовується для реалізації концепції Smart Plant Factory. Цей клас обробляє керування веб-сокетами та керує зв’язками. Що стосується раніше представлених абстрактних класів, комунікації управляються асинхронно та в різних обробниках завдань. Щодо комунікаційної сторони, клас VFarm є клієнтом, який може підключатися до кількох серверів (тут, деяких об’єктів VCsm) через мережу одночасно, щоб об’єднати їх усіх в одну систему керування.

Дійсно, завдяки цій особливості об’єкти VFarm можна використовувати для передачі повідомлень між системами. Крім того, цей багатосерверний клієнт має можливість керувати списком відомих серверів і автоматично підключатися до них у разі збоїв у мережі. Той факт, що цей клієнт може бути підключений до кількох серверів, не є тривіальним. Ця особливість реалізується шляхом створення нового веб-сокета для кожного сервера та відстеження доступних на даний момент серверів.

Подібно до класу VCsm, клас VFarm може працювати з мережевими помилками. Базова поведінка SPF може бути успадкована від цього класу для будь-якої спеціальної реалізації SPF.

Абстрактний клас VHuman — це інтерфейс командного рядка керування для підключення до електронної системи керування через екземпляр VCsm.

Клас VSensor є першим членом останнього рівня успадкування. Це також останній абстрактний рівень, необхідний електронній системі управління.

Клас VActuator є другим членом останнього рівня успадкування. Він дуже схожий на клас VSensor, але присвячений приводам.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ХОРОНИ ПРАЦІ

4.1 Долікарська допомога при шоку

Розглянемо порядок, що визначає механізм надання домедичної допомоги при підозрі на шок не медичними працівниками.

Шок – це стан між життям та смертю; загальний тяжкий розлад життєво важливих функцій організму, спричинений порушенням нервової регуляції життєво важливих процесів; характеризується розладами гемодинаміки, дихання, обміну речовин.

Ознаки шоку у постраждалого:

- бліда, холодна і волога шкіра;
- слабкість;
- неспокій;
- сухість в роті, відчуття спраги;
- часте дихання (більш ніж 20 вдихів за хвилину);
- порушення свідомості; непритомність.
- Причинами виникнення шоку можуть бути:
- зовнішня кровотеча;
- внутрішня кровотеча;
- травми різного ґенезу;
- опіки;
- серцевий напад тощо.

Послідовність дій при наданні домедичної допомоги постраждалим при підозрі на шок не медичними працівниками:

1) переконатися у відсутності небезпеки;

2) провести огляд постраждалого, визначити наявність свідомості, дихання;

3) викликати бригаду екстреної (швидкої) медичної допомоги;

4) якщо у постраждалого відсутнє дихання, розпочати проведення серцево-легеневої реанімації;

5) усунути причину виникнення шокового стану: зупинити кровотечу, іммобілізувати перелом тощо;

6) надати постраждалому протишокове положення:

а) перевести постраждалого в горизонтальне положення;

б) покласти під ноги постраждалого ящик, валик з одягу тощо таким чином, щоб ступні ніг знаходились на рівні його підборіддя;

в) підкласти під голову постраждалого одяг/подушку;

г) вкрити постраждалого термопокривалом/покривалом;

7) забезпечити постійний нагляд за постраждалим до приїзду бригади екстреної (швидкої) медичної допомоги;

8) при погіршенні стану постраждалого до приїзду бригади екстреної (швидкої) медичної допомоги повторно зателефонувати диспетчеру екстреної медичної допомоги.

4.2 Розробка, оформлення кімнати для психологічного розвантаження працівників

Напружений ритм життя шкільних працівників, інтенсифікація їх праці на тлі низької рухової активності породжують відомий дисонанс між вимогами, що пред'являються до інтелекту, емоційній сфері, і порівняно малої фізичним навантаженням. Робота нервової системи в подібному режимі часто веде до підвищеного напрузі, невміння розслаблятися, виходити з напруженого стану, знаходити психічну рівновагу. У більшості випадків у людей, схильних "хворобам століття" - неврозів, гіпертонії та ішемічної хвороби серця, - можна фіксувати підвищену м'язову напруженість, втрату навику довільного розслаблення м'язів. Крім того, інтенсивне навчання деяких предметів викликає необхідність зняття психічної напруги. Все це ставить перед психологічною службою школи нагальну задачу створення кабінету психологічного розвантаження (КПР).

Кабінет психологічного розвантаження в школі працює в п'яти режимах:

- Психологічна розвантаження співробітників і школярів після напруженої роботи в кінці робочого (навчального) дня або в спеціально відведений для цього час.
- Психологічний настрій (мобілізація) тих співробітників і школярів, які насилу включаються в напружений ритм роботи на початку робочого дня, навчання навичкам мобілізації в стресі (контрольна, іспит і т.п.).
- Зняття психологічного навантаження викладачів і школярів відповідно до курсу, призначеним психотерапевтом.
- Психопрофілактична робота з практично здоровими вчителями та школярами (навчання методам релаксації, медитації, аутогенного

тренування, навичкам безконфліктного спілкування, тренінг спілкування і т.д.).

- Забезпечення процесу інтенсивного навчання, включаючи методи суггестопедии, релаксопедії, гіпнопедії, а також використання кімнати психологічного розвантаження як експериментальної бази для розробки нових методів навчання.

Питання про можливість і необхідність відвідування сеансів психологічного розвантаження вирішується співробітниками психологічної служби на основі даних психодіагностики в залежності від характеру впливу. Для індивідуальної роботи відводиться від 5 до 30 хвилин на одну людину, на групу - 60 хвилин. При наявності в КПР 12-15 місць його пропускна здатність становить 60-80 чоловік у зміну, а курсове лікування можуть отримати одночасно до 200 чоловік, оскільки заняття проводяться два-три рази на тиждень. При проведенні занять інтенсивного навчання пропускна можливість КПР знижується, однак особи, які проходять інтенсивний курс, одночасно випробовують і психопрофілактичний вплив.

До облаштування КПР пред'являються певні технічні вимоги. Кабінет повинен складатися з двох зв'язаних між собою кімнат. Перша кімната є одночасно і робочим кабінетом психологічної служби. Сюди винесена вся апаратура, обслуговуюча сеанси психотерапії і заняття інтенсивного навчання. Крім того, з операторської через спеціальне дзеркальне скло з одного боку можна проводити невключенное спостереження за поведінкою відвідувачів в психотерапевтичному залі. Такий зал обладнується 10-15 м'якими кріслами з високими підголовниками і вмонтованими в них роз'ємами для підключення індивідуальних навушників. Площа залу повинна бути не менше 40 кв. м, стелі повинні бути досить високими, щоб відвідувачі не відчували себе в тісноті і щоб в затемненому залі у них виникало почуття усамітнення.

Інтер'єр кабінету психологічного розвантаження повинен викликати у відвідувачів позитивні емоції, надавати сприятливий вплив на організм

людини. Шумоізольовані стіни КПР повинні бути блакитного або світло-зеленого кольору. В якості будівельного матеріалу використовуються перфопліти або акустична штукатурка, в декоративній обробці застосовуються шкірозамінник, дерматин та інші матеріали, за допомогою яких можна створити затишок, що позитивно впливає на настрій людини.

Психотерапевтичний зал також повинен бути обладнаний автоматичною системою затемнення вікон, екраном, світломузичним пристроєм, акустичними колонками, апаратами для іонізації, зволоження та кондиціонування повітря, великим акваріумом з підсвічуванням і технічними засобами управління станом людини (ТСУС), запропонованими С. М. Зоріна. Будучи головною частиною керованої цветозвукового середовища, ТСУС являє собою поліфункціональну систему для реалізації специфічних аудіовізуальних впливів з метою управління увагою, релаксацією, активізацією, а також для зниження рівня антісуггестівних бар'єрів. У систему ТСУС входить два функціональних блоку:

- Установка керованого колірного клімату. Вона має вигляд рами з алюмінієвого сплаву П-образного профілю, розташованої по периметру кімнати у стелі. П'ять груп ламп накачування, змонтовані на цій рамі, дозволяють здійснювати управління яскравістю і спектральним складом освітлюваної аудиторії. Всі світильники спрямовані в стелю, щоб забезпечувати м'який, розсіяне світло в залі. Освітлення психотерапевтичного залу може змінюватися як вручну оператором, так і автоматично, за заздалегідь розробленою для кожного виду впливу програмою.

- Светодінамінеская система (СДС). На відміну від установки керованого колірного клімату, що змінює лише яскравість і кольоровість освітлення, СДС дозволяє здійснити на екрані синтез керованих параметрів світлодинамічних символів. Ці символи, змінюючи за бажанням оператора або за заданою програмою свої обриси, колір, яскравість, насиченість, швидкість і спрямованість руху, можуть з'єднуватися в складні, що

розвиваються у часі динамічні композиції, що мають багатопланове застосування.

Все управління психотерапевтичним сеансом здійснюється з операторської кімнати, де на стелажах розташовані стереомагнітофони з Мікшерський пультом, діа- та кінопроектори, що забезпечують за допомогою спеціально підібраних слайдів, кінозарісовок і музики емоційно-естетичний вплив на людину.

.

ВИСНОВКИ

У роботі було розроблено систему керування боксом вирощувальної ділянки.

Для цього було проаналізовано можливі технології для створення вирощувальних комплексів. На основі проведено аналізу було принято остаточну структуру системи.

Далі було описано структуру системи та алгоритм її роботи. Для обраної струтури системи було вибрано всі апаратні засоби та розроблено схему електричну принципову всіх з’єднань системи.

Після технічної реалізації системи було створено програмне забезпечення для керування боксом, а також мобільний додатко для можливості контролювання процесу в режимі реального часу.

Впровадження результатів роботи дозволить розширити сферу застосування вирощувальних боксів для забезпечення потреб населення.

БІБЛІОГРАФІЯ

1. Open phytotron: A new iot device for home gardening. URL : https://orbi.uliege.be/bitstream/2268/248995/1/abdelouhahid2020.pdf.

2. Smart nest box: Iot based nest monitoring in artificial cavities. URL : https://orbi.umons.ac.be/bitstream/20.500.12907/42296/1/CommNet_2020__Smart_Nest_Box___Camera_Ready.pdf.

3. Wireless sensor network for agricultural environment using raspberry pi based sensor. URL : nodeshttps://www.researchgate.net/publication/322815903_Wireless_sensor_network_for_agricultural_environment_using_raspberry_pi_based_sensor_nodes.

4. Smart system for bicarbonate control in irrigation for hydroponic precision farming. URL : https://www.mdpi.com/1424-8220/18/5/1333.

5. Design, construction and testing of iot based automated indoor vertical hydroponics farming test-bed in qatar. URL : https://www.researchgate.net/publication/345146174_Design_Construction_and_Testing_of_IoT_Based_Automated_Indoor_Vertical_Hydroponics_Farming_Test-Bed_in_Qatar.

6. Personal food computer: A new device for controlled-environment agriculture. URL : https://www.researchgate.net/publication/317650227_ Personal _Food_Computer_A_new_device_for_controlled-environment_agriculture.

7. Plant growth prediction based on hierarchical auto-encoder. URL : https://ieeexplore.ieee.org/document/9748287.

8. Smart Plant factory : The next generation indoor vertical farms. URL : https://link.springer.com/book/10.1007/978-981-13-1065-2

9. А.Г. Микитишин, М.М. Митник, П.Д. Стухляк, В.В. Пасічник Комп’ютерні мережі. Книга 1. [навчальний посібник] (Лист МОНУ №1/11-8052 від 28.05.12р.) - Львів, "Магнолія 2006", 2013. – 256 с.

10. А.Г. Микитишин, М.М. Митник, П.Д. Стухляк, В.В. Пасічник Комп’ютерні мережі. Книга 2. [навчальний посібник] (Лист МОНУ №1/11-11650 від 16.07.12р.) - Львів, "Магнолія 2006", 2014. – 312 с.

11. Микитишин А.Г., Митник, П.Д. Стухляк. Комплексна безпека інформаційних мережевих систем: навчальний посібник – Тернопіль: Вид-во ТНТУ імені Івана Пулюя, 2016. – 256 с.

12. Микитишин А.Г., Митник М.М., Стухляк П.Д. Телекомунікаційні системи та мережі : навчальний посібник для студентів спеціальності 151 «Автоматизація та комп’ютерно-інтегровані технології» – Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2017 – 384 с.