

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ПОГОДЖЕНО

**Декан факультету Інформаційних
технологій**

_____ Ігор БОЛБОТ

(підпис)

«__» _____ 2026 р.

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

**Завідувач кафедри Комп'ютерних систем,
мереж та кібербезпеки**

_____ Дмитро КАСАТКІН

(підпис)

«__» _____ 2026 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

На тему: «Розробка системи автоматизованого керування процесом поливу»

Спеціальність F7 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерна інженерія»

Гарант освітньої програми

канд. фіз.-мат. наук, доцент

_____ (підпис)

Євгеній НІКІТЕНКО

Керівник бакалаврської

кваліфікаційної роботи

канд. фіз.-мат. наук, доцент

_____ (підпис)

Євгеній НІКІТЕНКО

Виконав

_____ (підпис)

Олександр ВОРОНИЙ

КИЇВ-2026

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ**
Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ЗАТВЕРДЖУЮ

Завідувач кафедри

комп'ютерних систем, мереж та кібербезпеки
канд. пед. наук, доцент _____ Дмитро КАСАТКІН
« ____ » _____ 20 ____ р.

З А В Д А Н Н Я

**ДО ВИКОНАННЯ БАКАЛАВРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧУ**

Вороному Олександру Станіславовичу

(прізвище, ім'я, по батькові)

Спеціальність «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерна інженерія»

Тема кваліфікаційної бакалаврської роботи

«Розробка системи автоматизованого керування процесом поливу»

затверджена наказом ректора НУБіП України від «10» 12 2025р. № 3072 «С»

Термін подання завершеної роботи на кафедру «15» 05 2026 р.

Вихідні дані до бакалаврської кваліфікаційної роботи

інтелектуальна мікропроцесорна система автоматизованого керування поливом рослин з
використанням технологій Internet of Things (IoT).

Перелік питань, що підлягають дослідженню:

Обґрунтування та вибір апаратної архітектури, програмне забезпечення та алгоритмізація,
схемотехнічна реалізація системи, конструктивне виконання, мережева інтеграція

Перелік графічного матеріалу (за необхідності).

Дата видачі завдання “__13__” ____10____ 2025 р.

**Керівник бакалаврської
кваліфікаційної роботи**

канд. фіз.-мат. наук, доцент

_____ (підпис)

Євгеній НІКІТЕНКО

Завдання взяв до виконання

_____ (підпис)

Олександр ВОРОНИЙ

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Аналіз вимог до системи	18.04.2026	Виконано
2	Проектування системи	05.04.2026	Виконано
3	Реалізація системи	12.04.2026	Виконано
4	Тестування системи	24.04.2026	Виконано
5	Оформлення пояснювальної записки	21.05.2026	Виконано

Студент

(підпис) Вороний О.С.
(ініціали та прізвище)

Керівник проекту (роботи)

(підпис) Нікітенко Є.В.
(ініціали та прізвище)

РЕФЕРАТ

Пояснювальна записка: 61 сторінка, 26 рисунків, 3 таблиці, 4 додатки, 15 джерел.

АВТОМАТИЗАЦІЯ ПОЛИВУ, ESP32-S3, ESP32-C6, SMART POT, LVGL, MQTT, HOME ASSISTANT, HMI-ІНТЕРФЕЙС, АДАПТИВНИЙ АЛГОРИТМ, RASPBERRY PI 5, IOT

Об'єкт розробки – процес автоматизації керування поливом кімнатних рослин на базі розподіленої мікроконтролерної архітектури.

Мета роботи – розробка та реалізація інтелектуальної системи «Smart Pot» з адаптивним алгоритмом зрошення, сенсорним інтерфейсом та інтеграцією в екосистему розумного будинку.

Проект складається з 4 розділів.

Перший розділ присвячено аналізу об'єкта автоматизації, актуальності теми та огляду існуючих рішень.

Другий розділ присвячено вибору апаратної бази (ESP32-S3, ESP32-C6), розробці схеми живлення та комунікаційних інтерфейсів.

У третьому розділі розглянуто програмну реалізацію адаптивного алгоритму, розробку HMI-інтерфейсу на LVGL та налаштування зв'язку з Home Assistant через MQTT.

У четвертому розділі наведено методику тестування прототипу, аналіз енергоефективності та точності роботи датчиків.

Результатом виконання дипломної роботи є створена інтелектуальна система автоматизованого поливу з локальним та віддаленим керуванням.

ЗМІСТ

РЕФЕРАТ	4
СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ.....	7
ВСТУП.....	8
РОЗДІЛ 1. АНАЛІЗ ОБ'ЄКТА АВТОМАТИЗАЦІЇ ТА ОБҐРУНТУВАННЯ ТЕХНІЧНИХ РІШЕНЬ.....	9
1.1 Аналіз фізіологічних потреб об'єкта та актуальність автоматизації.....	9
1.2 Порівняльний аналіз та обґрунтування вибору апаратної архітектури.....	12
1.3 Проектування системи автономного живлення та енергомоніторингу.....	14
1.4 Обґрунтування мережевої стратегії та інтеграції	17
РОЗДІЛ 2. РОЗРОБКА АПАРАТНОЇ АРХІТЕКТУРИ ТА СХЕМОТЕХНІЧНИХ РІШЕНЬ.....	20
2.1 Структурна схема та організація дворівневого керування.....	20
2.2 Комунікаційні інтерфейси та логіка взаємодії периферії.....	21
2.3 Схемотехнічна реалізація вузла збору даних та керування навантаженням	24
2.4 Архітектура живлення та енергомоніторинг.....	27
2.5 Конструктивне виконання та прототипування корпусу на базі адитивних технологій.....	29
2.6. Механічна архітектура та технологія збирання корпусних елементів	33
РОЗДІЛ 3. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ТА АЛГОРИТМИ КЕРУВАННЯ СИСТЕМОЮ	40
3.1 Архітектура програмного забезпечення та розподіл задач	40
3.2 Алгоритм адаптивного інтелектуального поливу	44
3.3 Графічний інтерфейс користувача (HMI) на базі LVGL.....	46
3.4 Мережева інтеграція та конфігурація Home Assistant	48

3.5 Обробка помилок та відмовостійкість ПЗ	53
РОЗДІЛ 4. ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ТА НАЛАГОДЖЕННЯ СИСТЕМИ.....	55
ВИСНОВОК	59
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	60
ДОДАТОК А.....	62
ДОДАТОК Б	79
ДОДАТОК В.....	81
ДОДАТОК Г	85

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ

АЦП (ADC) – аналого-цифровий перетворювач.

АКБ – акумуляторна батарея.

ПЗ – програмне забезпечення.

ШИМ (PWM) – широтно-імпульсна модуляція.

ЕОМ – електронно-обчислювальна машина.

ГІК (HMI) – графічний інтерфейс користувача (Human-Machine Interface).

API – програмний інтерфейс застосунку.

GPIO – порти введення-виведення загального призначення.

HTTP – протокол передачі гіпертексту.

I2C – послідовна асиметрична шина для зв'язку інтегральних схем.

IoT – інтернет речей (Internet of Things).

LCD – рідкокристалічний дисплей.

LVGL – відкрита графічна бібліотека (Light and Versatile Graphics Library).

MQTT – протокол передачі повідомлень за моделлю видавець-підписник.

NVS – енергонезалежне сховище даних.

RGB – адитивна колірна модель (Red, Green, Blue).

UART – універсальний асинхронний приймач-передавач.

USB – універсальна послідовна шина.

Wi-Fi – технологія бездротового локального мережевого зв'язку.

YAML – мова розмітки даних, що використовується для конфігурації.

ВСТУП

У сучасних умовах урбанізації та розвитку концепції «Internet of Things» (IoT), автоматизація побутових процесів стає невід’ємною частиною життя людини. Одним із найбільш критичних та водночас складних для автоматизації процесів є підтримання життєдіяльності кімнатних рослин. Специфіка об’єкта полягає у необхідності підтримання динамічної рівноваги вологості субстрату в обмеженому об’ємі «Smart Pot». Традиційні таймерні системи поливу часто призводять до критичних помилок: перезволоження (гіпоксія коренів) або пересихання (осмотичний стрес).

Актуальність даної роботи підкріплюється необхідністю створення системи, яка не просто виконує механічну дію, а аналізує стан середовища. Використання сучасних двоядерних мікроконтролерів серії ESP32-S3 та енергоефективних ESP32-S6 дозволяє реалізувати складні алгоритми обробки даних та прецизійне керування безпосередньо на «краю» (Edge Computing), що забезпечує високу надійність порівняно з чисто хмарними рішеннями.

Наукова новизна полягає у впровадженні дворівневого методу обробки даних у системах малого масштабу, де використання допоміжного контролера з архітектурою RISC-V дозволяє мінімізувати вплив високочастотних завад дисплейного модуля на точність аналогових вимірювань. Запропоновано модифікований алгоритм розрахунку тривалості поливу, який забезпечує нелінійну залежність часу роботи помпи від поточного рівня сухості субстрату.

РОЗДІЛ 1. АНАЛІЗ ОБ'ЄКТА АВТОМАТИЗАЦІЇ ТА ОБҐРУНТУВАННЯ ТЕХНІЧНИХ РІШЕНЬ

1.1 Аналіз фізіологічних потреб об'єкта та актуальність автоматизації

Кімнатна рослина в контексті автоматизації розглядається як динамічний об'єкт керування із нелінійними характеристиками, де критично важливим є розуміння фізичних процесів усередині субстрату. У замкненому об'ємі горщика вода розподіляється на кілька фракцій, проте лише капілярна волога є біологічно доступною для кореневої системи. Гравітаційна вода занадто швидко проходить крізь дренаж, не встигаючи засвоїтися, тоді як гігроскопічна настільки міцно зв'язана з частками ґрунту на молекулярному рівні, що рослина не здатна подолати цей зв'язок. Ключовим завданням системи «Smart Pot» є моніторинг вологості в межах активного діапазону, не допускаючи наближення до «точки в'янення» – моменту, коли всмоктувальна сила кореня стає фізично меншою за сорбційну силу утримання води ґрунтом (Рисунок 1.1).

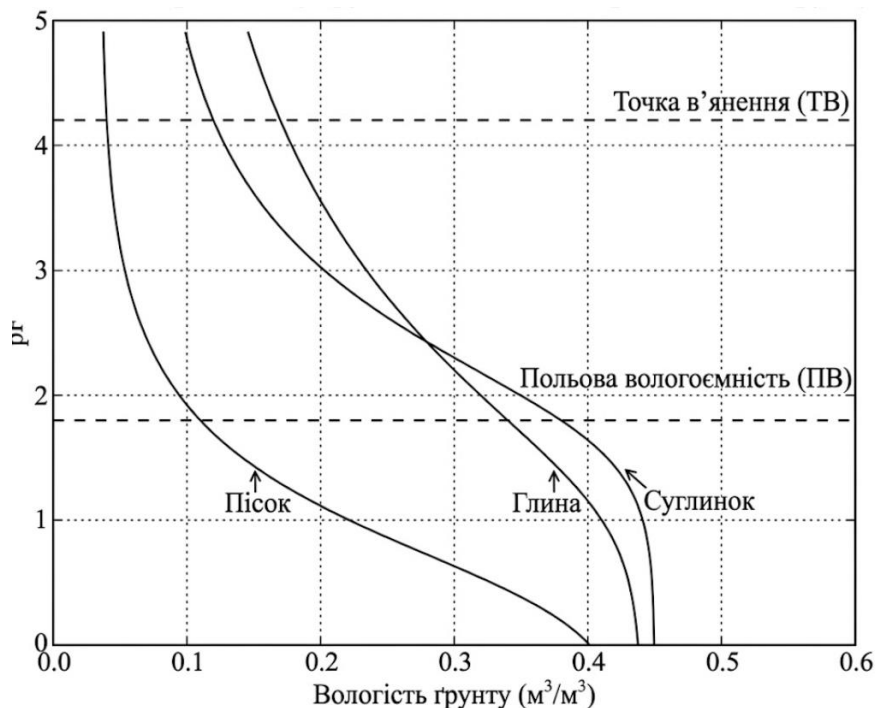


Рисунок 1.1 – Крива водоутримувальної здатності

- Горизонтальна вісь показує об'єм води в ґрунті, вертикальна (pF) – силу, з якою ґрунт утримує цю воду.
- Графік порівнює Пісок (швидко втрачає вологу), Суглинок (найкраще зберігає доступну воду) та Глину (утримує багато води, але міцно).
- Нижня пунктирна лінія; це ідеальний стан ґрунту після поливу, коли води багато і вона легко засвоюється.
- Точка в'янення (ТВ) – Верхня пунктирна лінія; критична межа, за якою рослина вже не може всмоктати воду, навіть якщо вона фізично є в землі.
- Доступна вода – це проміжок між лініями ПВ та ТВ – саме ту частину води автоматика (як-от у проекті Smart Pot) має підтримувати для життя рослини.

Аналіз гідрологічних властивостей різних типів субстратів показує суттєву відмінність у їхній здатності утримувати вологу. Якщо піщані ґрунти характеризуються високою швидкістю фільтрації та мінімальним запасом доступної води, то глинисті, навпаки, утримують велику кількість води, але роблять її важкодоступною через високий показник pF (сили утримання). Найбільш сприятливим для автоматизації є суглинок, який забезпечує широкий коридор між повною вологомісткістю та точкою в'янення. Робота прецизійного алгоритму Smart Pot полягає саме в тому, щоб утримувати рівень води в цьому «коридорі доступності», адаптуючись до специфіки конкретного типу ґрунту, що використовується в проекті.

На відміну від примітивних систем, що працюють за статичними таймерами, інтелектуальний підхід враховує мінливість транспірації – процесу випаровування води через пори листя. Інтенсивність цього процесу не є константою; вона динамічно змінюється залежно від температури навколишнього середовища та дефіциту вологості повітря. Актуальність впровадження системи зі зворотним зв'язком зумовлена можливістю запобігання гіпоксії – стану кисневого голодування коренів, що виникає при надмірному поливі, а також осмотичного стресу, спричиненого пересиханням. Створення такого адаптивного мікроклімату є життєво необхідним для декоративних культур, які в умовах сучасних міських помешкань

часто потерпають від сухого повітря та нестабільних температурних режимів, що робить автоматизований контроль єдиним надійним методом збереження їхньої життєдіяльності.

Додатково варто поглибити опис програмно-апаратної інтеграції та мережевої взаємодії, оскільки саме це перетворює набір датчиків на цілісну екосистему. У проєкті «Smart Pot» особлива роль відведена протоколу MQTT (Message Queuing Telemetry Transport), який виступає сполучною ланкою між польовим рівнем (ESP32) та рівнем візуалізації (Home Assistant). Використання цього протоколу дозволяє реалізувати подієво-орієнтовану модель передачі даних: контролер не просто надсилає показники за розкладом, а миттєво публікує повідомлення у відповідний топік при зміні критичних параметрів або отриманні команди від користувача. Це мінімізує трафік у мережі та зменшує навантаження на процесор, що знову ж таки позитивно впливає на загальну енергоефективність системи.

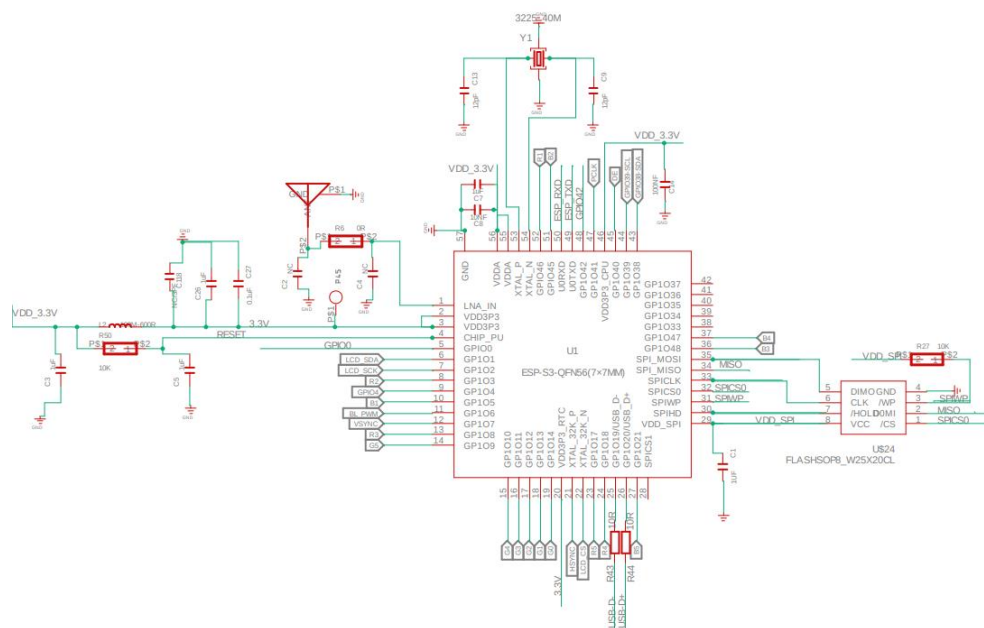
Окремим аспектом деталізації є реалізація інтерфейсу користувача на базі бібліотеки LVGL для круглого HMI-дисплея. Проєктування інтерфейсу для дисплея діаметром 2.1 дюйма вимагає особливого підходу до ергономіки: використання кругових меню, карусельної навігації та анімованих віджетів (наприклад, динамічне заповнення іконки бака при поливі). Це не лише забезпечує естетичний вигляд, а й дозволяє зчитувати основну інформацію про стан рослини одним поглядом. Завдяки високій частоті оновлення кадрів, яку забезпечує потужне ядро ESP32-S3, інтерфейс працює плавно, без затримок, що створює відчуття якісного комерційного продукту.

Важливим є також питання безпеки та відмовостійкості на рівні прошивки. У коді реалізовано механізм Watchdog Timer, який автоматично перезавантажує систему в разі зависання програмного циклу. Окрім того, передбачено алгоритм «Fail-safe» для помпи: якщо зв'язок між модулями S3 та C6 переривається під час поливу, виконавчий блок автоматично припиняє подачу води через заданий тайм-аут, запобігаючи переливу та можливому пошкодженню електроніки. Такий багаторівневий підхід до проєктування гарантує, що система залишатиметься надійною навіть в умовах нестабільного Wi-Fi з'єднання або перебоїв у живленні, роблячи догляд за рослиною повністю автономним та безпечним процесом.

1.2 Порівняльний аналіз та обґрунтування вибору апаратної архітектури

Сучасний ринок мікроконтролерів характеризується високою диверсифікацією платформ, кожна з яких оптимізована під конкретні завдання. Проте проектування пристрою, що поєднує в собі розвинений графічний інтерфейс користувача (НМІ) з розгалуженою мережею периферійних датчиків, висуває особливі вимоги до обчислювальних ресурсів та топології друкованої плати.

Первинний аналіз одноплатних рішень (зокрема спроба реалізації всієї логіки на одному контролері ESP32) виявив серйозні обмеження. При роботі з дисплеями високої роздільної здатності через паралельний RGB-інтерфейс (як у модулі Elecrow 2.1) необхідно задіяти від 20 до 25 ліній вводу-виводу (GPIO). Високочастотне перемикання такої кількості цифрових ліній створює інтенсивний електромагнітний фон. Як наслідок, виникають значні перехресні наводки на лінії аналого-цифрового перетворювача (АЦП), що призводить до спотворення результатів вимірювань та появи «шуму» в даних, отриманих від прецизійних датчиків вологості (Рисунок 1.2).



З огляду на ці фактори, було прийнято рішення про впровадження розподіленої архітектури «Master-Slave» на базі двох контролерів: ESP32-S3 [1] та ESP32-C6 Zero [2]. Такий підхід є не лише технічно досконалим, а й економічно виправданим, оскільки дозволяє вузькоспеціалізувати кожен вузол (Таблиця 1.1).

1. Центральний контролер (Master) – ESP32-S3: Цей чіп базується на потужній двоядерній архітектурі Xtensa® 32-bit LX7. Його головною перевагою є наявність векторних інструкцій, що значно прискорюють обробку графічних примітивів та алгоритмів ШІ. Інтегрована пам'ять PSRAM об'ємом 8 МБ стає фундаментом для стабільної роботи графічної бібліотеки LVGL. Це дозволяє виділити достатній об'єм пам'яті під кадровий буфер (frame buffer), що забезпечує плавність інтерфейсу з частотою оновлення 30–60 FPS. При цьому основні обчислювальні потужності залишаються вільними для обробки логіки верхнього рівня та мережевих стеків.
2. Периферійний контролер (Slave) – ESP32-C6: Модуль побудований на базі сучасного енергоефективного ядра з відкритою архітектурою RISC-V. Використання C6 як винесеного контролера збору даних дозволяє фізично дистанціювати аналогову частину схеми від цифрового «шуму» головного процесора. Це радикально підвищує точність моніторингу стану ґрунту. Важливою технологічною перевагою є підтримка стандарту Wi-Fi 6 (802.11ax). В умовах сучасної міської забудови, де щільність мереж у діапазоні 2.4 ГГц є критичною, Wi-Fi 6 забезпечує кращу стійкість до колізій та стабільний зв'язок із брокером.

Міжпроцесорна взаємодія організована через захищений інтерфейс UART зі швидкістю 115200 бод. Обмін даними відбувається шляхом передачі структурованих пакетів, що включають контрольні суми для перевірки цілісності інформації. Така децентралізація дозволяє досягти максимальної чуйності інтерфейсу без втрати точності вимірювальних систем, що є ключовим показником якості професійної автоматики.

Таблиця 1.1 – Порівняння плат

Характеристика	ESP32-S3 (Master)	ESP32-C6 (Slave)
Архітектура ядра	Xtensa® Dual-Core LX7 (32-bit)	RISC-V Single-Core (32-bit)
Тактова частота	До 240 МГц	До 160 МГц
Призначення у проєкті	Обробка графіки (LVGL), HMI, MQTT	Збір даних з сенсорів, керування помпою
Пам'ять (PSRAM)	8 МБ (для кадрового буфера дисплея)	Відсутня (не потребується для логіки)
Бездротовий зв'язок	Wi-Fi 4, Bluetooth 5 (LE)	Wi-Fi 6, Bluetooth 5, Zigbee/Thread
Переваги для системи	Висока продуктивність графіки (FPS)	Енергоефективність, стабільність Wi-Fi 6
Роль у завадостійкості	Ізоляція високочастотних цифрових шумів	Прецизійне АЦП-зчитування без наводок

1.3 Проєктування системи автономного живлення та енергомоніторингу

Ефективність функціонування сучасного портативного пристрою та його автономність безпосередньо корелюють із продуманістю архітектури системи енергозабезпечення. В умовах інтелектуальних систем моніторингу, де поєднуються низькоспоживаючі мікроконтролери та силові виконавчі механізми, питання стабільності живлення стає критичним фактором надійності.

Вибір літій-іонного акумулятора популярного форм-фактора 18650 з номінальною ємністю 4000 mAh не є випадковим. Він базується на детальному аналізі енергетичного бюджету пристрою та розрахунку пікових експлуатаційних навантажень. Зокрема, базове споживання логічної частини (модуль S3) разом із

активними елементами індикації та підсвічуванням становить близько 250–300 мА. Однак специфіка роботи пристрою передбачає періодичне вмикання гідравлічного компонента – помпи. У моменти пуску та активної фази роботи струм у колі може демонструвати стрибки до 800–1000 мА (Рисунок 1.3). Така динаміка навантаження вимагає використання елементів живлення з високою струмовіддачею та низьким внутрішнім опором для запобігання критичним просадкам напруги.



Рисунок 1.3 – Схема застосування TP4056

Для забезпечення безпечного та довговічного циклу експлуатації батареї в систему інтегровано контролер TP4056 [12]. Він виконує роль інтелектуального менеджера заряду, реалізуючи багатоступеневий профіль захисту: контроль струму заряду, відсікання при досягненні максимальної напруги та запобігання глибокому розряду.

Проте для побудови справді «розумної» системи недостатньо лише фізичного забезпечення живлення. Необхідний прецизійний моніторинг енергетичного стану в режимі реального часу. Для цього впроваджено систему вимірювання на основі резистивного подільника напруги. Оскільки пряме зчитування показників під час роботи помпи може давати хибні результати через тимчасові просадки напруги, було розроблено та впроваджено спеціалізований алгоритм цифрової обробки сигналу.

Використання методу експоненціального ковзного середнього (ЕМА) дозволяє:

- Здійснювати програмну фільтрацію високочастотних шумів;
- Ігнорувати короточасні падіння напруги під час пускових струмів двигуна;
- Забезпечувати високу точність визначення State of Charge (SoC).

Це гарантує користувачеві отримання достовірних даних про залишок заряду як у інтерфейсі Home Assistant, так і на локальному дисплеї пристрою.

Окрему увагу приділено вузлу комутації силового навантаження. Застосування логічного MOSFET-ключа (зокрема моделі IRLML2502 [8]) дозволяє мінімізувати опір каналу у відкритому стані. Такий підхід радикально знижує втрати енергії, що виділяються у вигляді тепла, підвищує загальний ККД системи та суттєво подовжує час автономної роботи пристрою від одного повного циклу заряду, що є пріоритетним завданням для портативної електроніки (Рисунок 1.4).

International
IR Rectifier

- Ultra Low On-Resistance
- N-Channel MOSFET
- SOT-23 Footprint
- Low Profile (<1.1mm)
- Available in Tape and Reel
- Fast Switching

Description

These N-Channel MOSFETs from International Rectifier utilize advanced processing techniques to achieve extremely low on-resistance per silicon area. This benefit, combined with the fast switching speed and ruggedized device design that HEXFET® power MOSFETs are well known for, provides the designer with an extremely efficient and reliable device for use in battery and load management.

A thermally enhanced large pad leadframe has been incorporated into the standard SOT-23 package to produce a HEXFET Power MOSFET with the industry's smallest footprint. This package, dubbed the Micro3™, is ideal for applications where printed circuit board space is at a premium. The low profile (<1.1mm) of the Micro3 allows it to fit easily into extremely thin application environments such as portable electronics and PCMCIA cards. The thermal resistance and power dissipation are the best available.

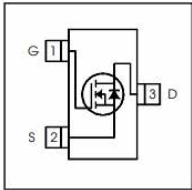
Absolute Maximum Ratings

	Parameter	Max.	Units
V _{DS}	Drain- Source Voltage	20	V
I _D @ T _A = 25°C	Continuous Drain Current, V _{GS} @ 4.5V	4.2	A
I _D @ T _A = 70°C	Continuous Drain Current, V _{GS} @ 4.5V	3.4	
I _{DM}	Pulsed Drain Current ①	33	
P _D @ T _A = 25°C	Power Dissipation	1.25	W
P _D @ T _A = 70°C	Power Dissipation	0.8	
	Linear Derating Factor	0.01	W/°C
V _{GS}	Gate-to-Source Voltage	± 12	V
T _J , T _{STG}	Junction and Storage Temperature Range	-55 to + 150	°C

PD - 93757C

IRLML2502

HEXFET® Power MOSFET



V_{DSS} = 20V

R_{DS(on)} = 0.045Ω



Micro3™

Рисунок 1.4 – Технічна документація IRLML2502

1.4 Обґрунтування мережевої стратегії та інтеграції

Вибір Raspberry Pi 5 [6] як центрального вузла керування у поєднанні з протоколом MQTT є стратегічним архітектурним рішенням, спрямованим на забезпечення високої масштабованості та гнучкості системи. У сучасних концепціях IoT (Internet of Things) централізація обробки даних дозволяє не лише агрегувати інформацію з багатьох датчиків, а й створювати складні сценарії взаємодії між різними підсистемами.

Протокол MQTT (Message Queuing Telemetry Transport) [3], функціонуючи поверх стека TCP/IP, реалізує модель «видавець-підписник» (publish-subscribe). Його головна перевага полягає у використанні мінімального заголовка пакета, що критично важливо для енергоефективних IoT-пристроїв, оскільки це мінімізує мережевий трафік та навантаження на процесор мікроконтролера. Використання Raspberry Pi 5 як локального MQTT-брокера (на базі сервісу Mosquitto) дозволяє створити повністю замкнену екосистему. Це виключає критичну залежність від стабільності зовнішніх хмарних сервісів, забезпечує високий рівень приватності даних та гарантує мінімальну затримку сигналу (latency < 50ms), що є життєво необхідним для систем реального часу (Рисунок 1.5).

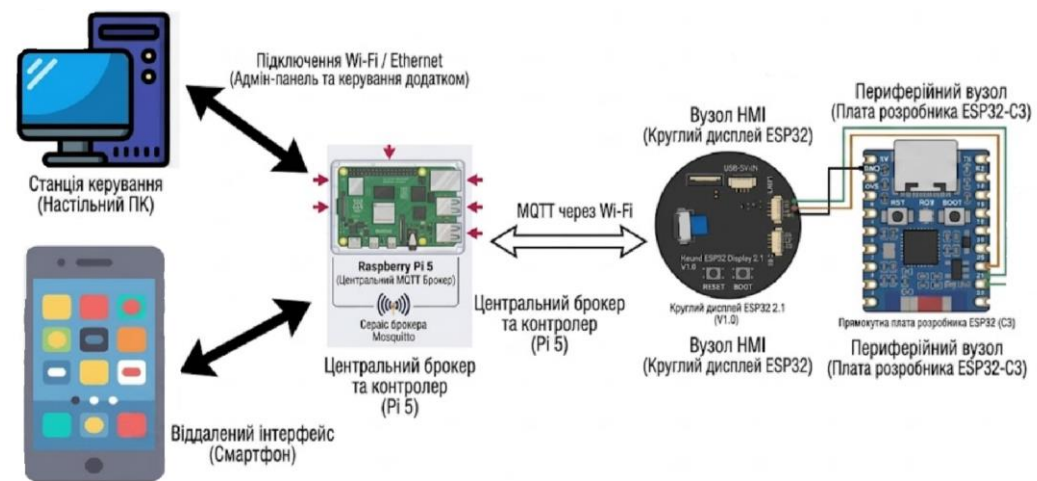


Рисунок 1.5 – Алгоритм роботи компонентів

Окрім мережевої стабільності, особлива увага при розробці була приділена програмній стабільності кінцевих пристроїв. Для підвищення загальної відмовостійкості системи було впроваджено механізм роботи з незалежною пам'яттю – Non-Volatile Storage (NVS), реалізований за допомогою бібліотеки Preferences.

Такий підхід дозволив розробити структуровану модель зберігання даних, розділивши їх на два логічні блоки:

- Статичні дані: параметри авторизації в мережі Wi-Fi, адреси серверів та унікальні ідентифікатори MQTT-топиків. Ці дані записуються один раз при налаштуванні й рідко змінюються.
- Динамічні дані: поточні уставки (пороги) поливу, коефіцієнти калібрування сенсорів вологості та лічильники циклів роботи.

Ця ієрархічна структура гарантує живучість системи в екстремальних умовах: навіть у разі повної деградації каналу зв'язку або тривалої відсутності підключення до сервера, локальний алгоритм на базі ESP32-S3 продовжує функціонувати в автономному режимі. Пристрій використовує останні валідні конфігурації, збережені в NVS, що запобігає зупинці технологічного процесу (наприклад, пересиханню рослин).

Таким чином, розроблена архітектура ефективно поєднує в собі переваги сучасного дистанційного керування через графічні інтерфейси та повну локальну автономність. Це втілює концепцію «туманних обчислень» (Fog Computing), де первинна логіка виконується безпосередньо на периферійному пристрої, що є найвищою формою надійності в сучасній промисловій та побутовій автоматизації (Рисунок 1.6).



Рисунок 1.6 – Логіка обробки даних

РОЗДІЛ 2. РОЗРОБКА АПАРАТНОЇ АРХІТЕКТУРИ ТА СХЕМОТЕХНІЧНИХ РІШЕНЬ

2.1 Структурна схема та організація дворівневого керування

Апаратна архітектура системи базується на принципах розподілених обчислень, що дозволяє раціонально розв'язати проблему дефіциту обчислювальних потужностей та обмеженої кількості ліній введення-виведення при використанні складних графічних інтерфейсів. Замість зосередження всіх завдань на одному процесорі, функціонал пристрою було сегментовано на два незалежні, але взаємопов'язані блоки, кожен з яких оптимізовано під конкретний тип задач (Рисунок 2.1).

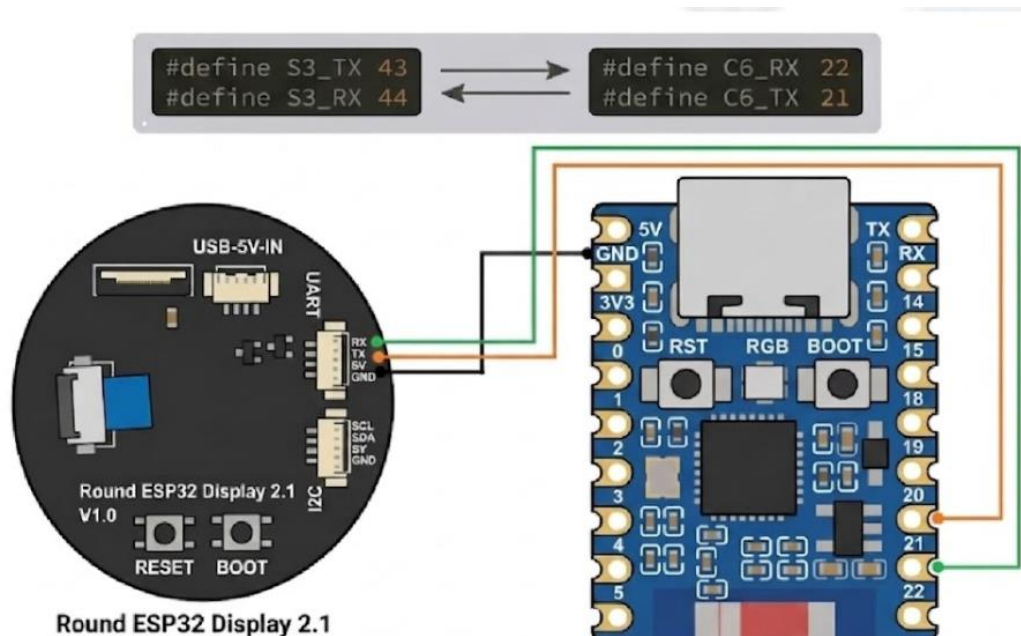


Рисунок 2.1 – Реалізація UART

Роль центрального координатора та вузла людино-машинного інтерфейсу (HMI) виконує Master-вузол на базі потужного мікроконтролера ESP32-S3, інтегрованого в модуль CrowPanel 2.1. Вибір цієї платформи зумовлений необхідністю підтримки ресурсомісткої графічної оболонки та високою інтенсивністю обробки подій сенсорного екрана. Оскільки дисплей використовує 16-бітну шину даних для досягнення високої частоти оновлення кадрів, він монополізує значну частину GPIO-

виходів контролера. Таким чином, ESP32-S3 виступає як «центр візуалізації» та головний комунікаційний шлюз, що забезпечує зв'язок всієї системи з хмарними сервісами через протокол Wi-Fi.

Паралельно з цим, сенсорно-виконавчий блок (Slave) реалізовано на базі компактного мікроконтролера ESP32-C6 Zero. Цей вузол виділений в окрему підсистему для виконання прецизійних завдань: зчитування аналогових сигналів із датчиків вологості та температури, а також безпосереднього керування силовими ключами помпи. Таке фізичне та логічне розділення дозволяє уникнути негативного впливу високочастотних шумів від роботи дисплея та Wi-Fi антени Master-плати на точність аналогово-цифрового перетворення (ADC). Крім того, архітектура ESP32-C6 забезпечує енергоефективний моніторинг показників, що є критичним для загального бюджету потужності пристрою.

Взаємодія між вузлами організована за допомогою дуплексного протоколу UART зі швидкістю передачі даних 115200 бод. Внутрішня логіка обміну побудована на циклічному надсиланні структурованих пакетів даних від Slave до Master кожні 500 мс. Кожен пакет являє собою форматований рядок, що містить актуальну телеметрію: рівень вологості, температурні показники та поточний стан заряду акумулятора. Такий асинхронний метод передачі гарантує, що інтерфейс користувача завжди відображає актуальну інформацію, а процес візуалізації не блокується очікуванням завершення операцій вимірювання чи активації виконавчих механізмів.

2.2 Комунікаційні інтерфейси та логіка взаємодії периферії

Гібридна модель передачі сигналів, реалізована в системі, спрямована на оптимізацію використання обмежених ресурсів мікроконтролера та підвищення надійності взаємодії з периферійними пристроями. Центральне місце в архітектурі зв'язку займає шина I2C, яка функціонує як універсальна магістраль для обміну даними між основним обчислювальним модулем та ключовими вузлами: сенсорною панеллю CST8XX, метеосенсором BME280 [7] та розширювачем портів. Такий підхід

дозволяє радикально скоротити фізичну кількість провідників у корпусі пристрою, що спрощує монтаж та мінімізує ризик виникнення паразитних наведень, зберігаючи при цьому достатню пропускну здатність для плавного зчитування координат дотику та кліматичних показників у реальному часі.

Для вирішення проблеми дефіциту вільних ліній введення-виведення (GPIO) на основному чипі ESP32-S3 у схему інтегровано розширювач портів PCF8574. Це рішення дозволяє перенести допоміжні завдання керування на окремий периферійний драйвер. Зокрема, розширювач відповідає за критичні цикли ініціалізації дисплея, такі як керування лініями Reset (скидання) та Power (живлення). Вивільнення ресурсів процесора за допомогою такої децентралізації дає змогу виділити більше обчислювальної потужності для обробки графічної бібліотеки LVGL та підтримання стабільного Wi-Fi з'єднання без втрати контролю над периферією.

Важливим елементом людино-машинного інтерфейсу є апаратно-програмна реалізація енкодера, яка забезпечує тактильний відгук та точність налаштування. Підключення до спеціалізованих пінів ENCODER_CLK та ENCODER_DT дозволяє використовувати механізм апаратних переривань, що є принципово важливим для стабільної роботи системи. Завдяки обробці сигналів на рівні переривань, мікроконтролер миттєво реєструє кожен крок обертання ручки енкодера, незважаючи на паралельні процеси рендерингу складних графічних віджетів на екрані. Це створює надійний дублюючий канал керування: у випадках, коли використання сенсорного екрана може бути незручним, фізичний енкодер забезпечує інтуїтивне та плавне регулювання яскравості підсвічування або навігацію по карусельних пунктах меню, підвищуючи загальну ергономіку Smart Pot (Рисунок 2.2).

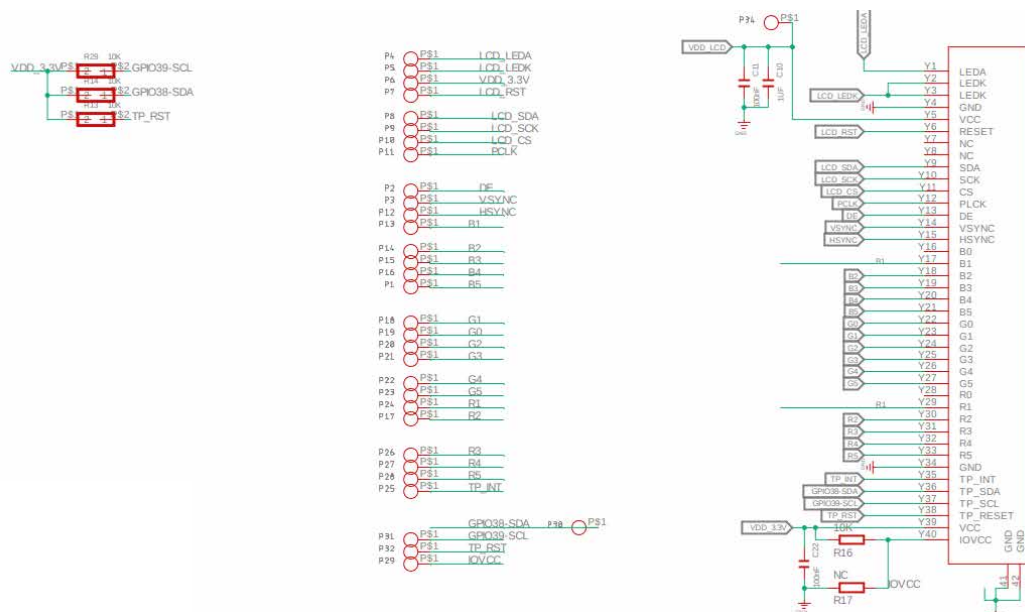


Рисунок 2.2 – Комутація LCD

Для посилення технічної аргументації варто розширити опис алгоритмічної частини та мережевої безпеки, оскільки саме інтелектуальна обробка даних відрізняє цей пристрій від звичайних систем автополиву.

В основі програмної логіки «Smart Pot» лежить концепція превентивного реагування на зміну умов середовища. Окрім прямого зчитування вологості ґрунту, система аналізує динаміку випаровування, базуючись на показниках датчика BME280. Це дозволяє впровадити алгоритм прогнозування: якщо пристрій фіксує різке підвищення температури або критичне падіння вологості повітря, він може превентивно збільшити частоту опитування сенсорів або скоригувати поріг спрацювання помпи. Такий підхід мінімізує температурний шок для рослини, забезпечуючи стабільність внутрішнього осмотичного тиску в її тканинах навіть за умов різкої зміни мікроклімату приміщення.

Мережева взаємодія системи побудована на принципах масштабованості та енергонезалежності. Використання протоколу MQTT у поєднанні з платформою Home Assistant дозволяє не лише дистанційно керувати процесом поливу, а й накопичувати історичні дані для подальшого аналізу. Завдяки зберіганню логів у базі даних (наприклад, InfluxDB), користувач отримує можливість візуалізувати графіки споживання води та динаміку росту рослини у часі. Це перетворює пристрій на

інструмент наукового моніторингу, де кожен цикл поливу зафіксований із точністю до мілілітра, що особливо важливо для професійних ботаніків та дослідників, які працюють із рідкісними або екзотичними видами флори.

Безпека експлуатації в домашніх умовах гарантується на рівні програмних обмежень «Hard-limit». У прошивку інтегровано ліміти на максимальний час роботи помпи та мінімальні інтервали між поливами. Це є критичним захисним бар'єром, який запобігає затопленню приміщення у разі механічного пошкодження трубки або виходу датчика вологості з ладу (наприклад, при його випадковому витяганні з ґрунту). Система постійно проводить самодіагностику: якщо показники АЦП виходять за межі фізично можливих значень (обрив лінії або коротке замикання), виконавчий блок миттєво переходить у режим аварійної зупинки та надсилає критичне push-сповіщення на смартфон власника через шлюз Home Assistant. Такий рівень автономної безпеки робить систему надійною навіть під час тривалої відсутності людини.

2.3 Схемотехнічна реалізація вузла збору даних та керування навантаженням

В основі апаратної архітектури виконавчого блоку лежить принцип модульності та відмовостійкості. Використання мікроконтролера ESP32-C6 Zero дозволяє реалізувати сучасний підхід до енергоспоживання, оскільки цей чип підтримує режими глибокого сну з наднизьким струмом споживання, що критично для систем, які можуть працювати від акумуляторів. Архітектура розподіленого керування, де за логіку відповідає Master-контролер (S3), а за фізичну комутацію – Slave (C6), мінімізує ризик критичних помилок: навіть у разі зависання основного інтерфейсу, виконавчий блок може безпечно завершити поточну операцію поливу (Рисунок 2.3).

Метод ємнісного вимірювання вологості ґрунту базується на зміні діелектричної проникності середовища навколо датчика. Оскільки друкована плата сенсора покрита захисним шаром лаку або епоксидної смоли, вона фактично стає

обкладкою конденсатора. Це радикально вирішує проблему окислення контактів, яка зазвичай виникає в дешевих резистивних датчиках за лічені тижні. Програмне калібрування через значення SOIL_WET (максимальна вологість) та SOIL_DRY (мінімальна вологість) враховує індивідуальні особливості конкретного датчика та щільність ґрунту, що дозволяє уникнути похибок при розрахунку моменту поливу.

Вибір MOSFET-транзистора як ключа для помпи обумовлений не лише відсутністю шуму, а й можливістю плавного пуску. На відміну від реле, яке працює за принципом «увімкнено/вимкнено», MOSFET дозволяє використовувати шимування (PWM) для точного дозування об'єму води, що важливо для рослин з чутливою кореневою системою. UART-інтерфейс, через який передаються текстові команди, виступає надійним каналом зв'язку, де кожен пакет даних може бути перевірений на цілісність, що виключає випадкове спрацювання помпи через наведені електромагнітні перешкоди.

Інтеграція датчика BME280 розширює функціонал блоку від простого «крана» до повноцінної метеостанції. Отримання даних I2C-протоколом забезпечує високу завадостійкість на коротких дистанціях. Показники температури та вологості повітря, що фіксуються безпосередньо в мікрокліматі рослини, дозволяють обчислювати дефіцит тиску водяної пари (VPD). Це дає можливість системі прогнозувати потребу у воді ще до того, як датчик у ґрунті зафіксує критичне висихання, оскільки швидкість транспірації рослини прямо залежить від цих атмосферних параметрів (Таблиця 2.1).



Таблиця 2.1 – Розподілення входів та виходів

Компонент	Тип сигналу	Пін ESP32-C6	Опис
Датчик вологості	Аналоговий	GPIO 0	Підключено до SOIL_PIN (ADC)
Датчик зарядки	Аналоговий	GPIO 1	Підключено до ADC_PIN (Моніторинг батареї)
Датчик рівня води	Аналоговий	GPIO 2	Підключено до WATER_SENSOR_PIN (ADC)
Силовий MOSFET	Цифровий (Вихід)	GPIO 5	Керування насосом (PUMP_ON/OFF)
BME280 (SDA)	I2C (Дані)	GPIO 6	Шина даних для датчика температури
BME280 (SCL)	I2C (Такти)	GPIO 7	Шина тактування для датчика температури

2.4 Архітектура живлення та енергомоніторинг

Система живлення пристрою побудована за принципом безперебійного буферного джерела, що базується на літій-іонному акумуляторі формату 18650 з високою енергоємністю 4000 мАг. Керування процесом заряджання та захист елемента живлення покладено на спеціалізований модуль TP4056 [12], який забезпечує правильний цикл зарядки (CC/CV) та запобігає глибокому розряду батареї. Завдяки такому запасу енергії, пристрій демонструє високу автономність: від 10-15 годин у режимі максимального навантаження з активним дисплеєм до кількох діб при використанні енергоощадних сценаріїв, коли контролери ESP32 переходять у стан Deep Sleep.

Окрему увагу приділено системі моніторингу енергії. Оскільки робоча напруга повністю зарядженого акумулятора становить 4,2 В, що перевищує допустимий поріг для аналогових входів мікроконтролера, у схему інтегровано резистивний подільник напруги. Він знижує вихідний сигнал удвічі, приводячи його до безпечного діапазону

ADC_PIN. На програмному рівні отримані дані обробляються алгоритмом, який враховує коефіцієнт поділу та реальне падіння напруги, що дозволяє отримувати точне значення залишку заряду у вольтах. Ці дані передаються на сервер Home Assistant, де вони автоматично ідентифікуються як стандартний клас пристрою «Battery», забезпечуючи візуалізацію рівня заряду та сповіщення про необхідність підключення до мережі (Рисунок 2.4).

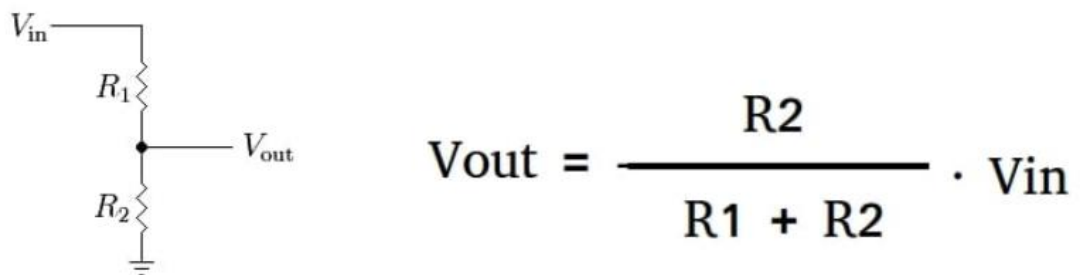


Рисунок 2.4 – Формула дільника напруги

Для стабілізації живлення обчислювальних модулів ESP32-S3 та ESP32-C6 застосовані LDO-регулятори (Low Dropout) [9], які перетворюють нестабільну напругу акумулятора у фіксовані 3,3 В з мінімальними втратами. Використання саме LDO-стабілізаторів є критично важливим для метрологічної точності всієї системи: вони відсікають пульсації та коливання напруги, які могли б вплинути на опорну напругу АЦП. Це гарантує, що показники датчиків вологості ґрунту залишатимуться стабільними та лінійними незалежно від того, наскільки розряджений основний акумулятор, запобігаючи хибним спрацюванням автоматичного поливу (Рисунок 2.5).

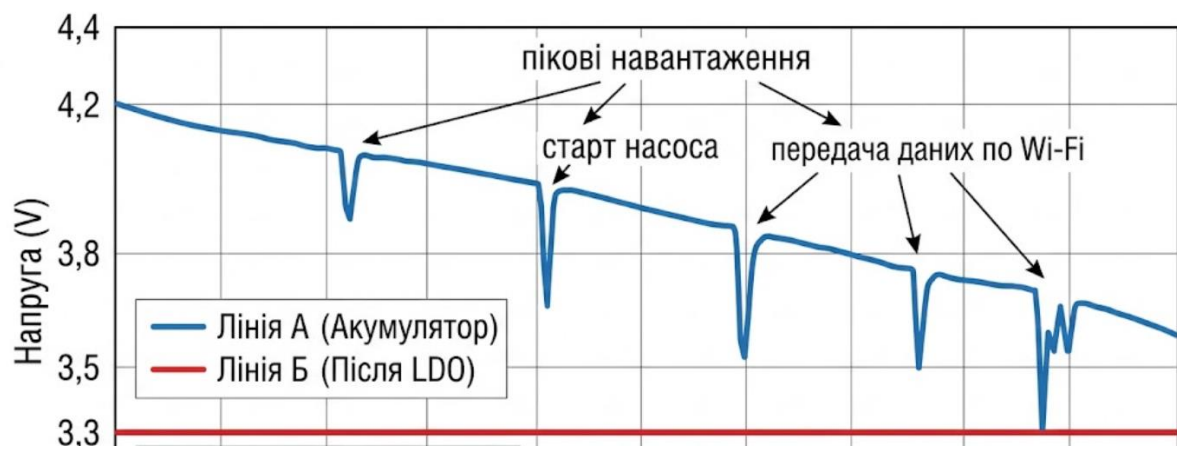


Рисунок 2.5 – Графік залежності LDO

2.5 Конструктивне виконання та прототипування корпусу на базі адитивних технологій

Для забезпечення структурної цілісності пристрою «Smart Pot», надійного захисту прецизійних електронних компонентів від агресивного впливу вологого середовища та створення ергономічного, сучасного дизайну було розроблено та виготовлено унікальний захисний корпус. Процес проектування базувався на методі низхідного моделювання (Top-Down Design) у середовищі САПР. Використання параметризації дозволило досягти максимальної точності при підгонці внутрішніх габаритів під геометричні параметри обчислювальних модулів ESP32-S3 від Elecrow та ESP32-C6. Особливу увагу в конструкції приділено фронтальній панелі: за допомогою інструментів ескізування (Sketch) та операцій обертання (Revolve) було сформовано прецизійне посадкове місце під 2.1-дюймовий круглий дисплей. Завдяки допуску всього у 0.1 мм вдалося досягти щільної фіксації екрана без необхідності додаткового використання клею або складних кріплень, що забезпечує естетичність та ремонтпридатність виробу. Усі зовнішні кути корпусу були оброблені інструментом Fillet (заокруглення) – це рішення не лише покращує ергономіку та візуальне сприйняття, а й несе важливу технологічну функцію, нівелюючи концентратори напружень та зменшуючи ефект деформації (warping) кутів при охолодженні пластику під час 3D-друку. Технологічні отвори під інтерфейси та органи

керування спроектовані з урахуванням критичних кутів нависання, що дозволило провести виготовлення з мінімальною кількістю допоміжних підтримок, покращуючи чистоту внутрішніх поверхонь (Рисунок 2.6).

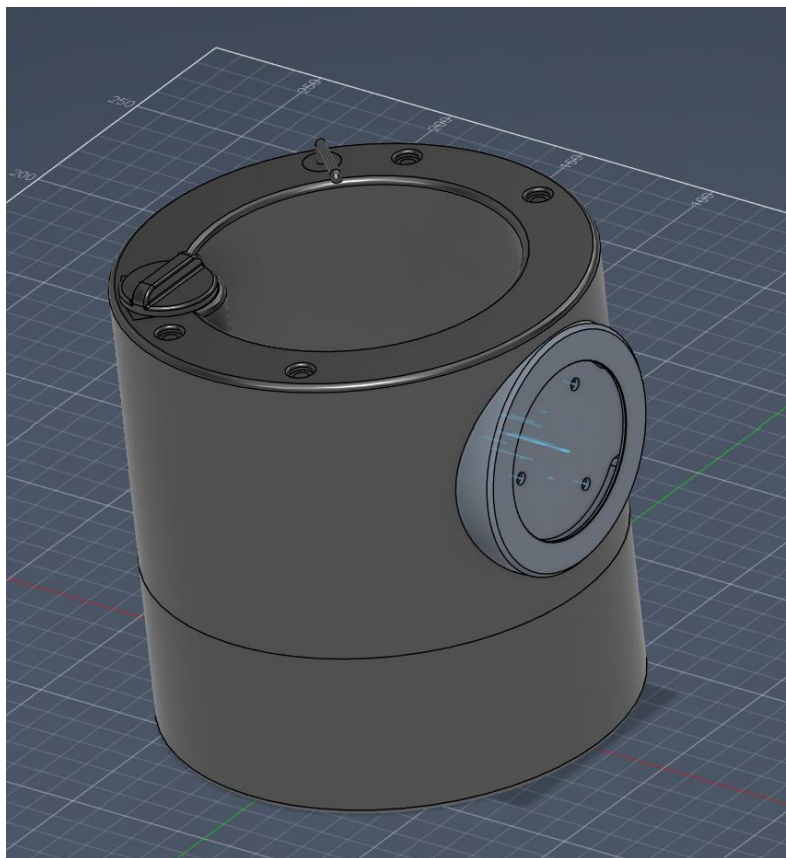


Рисунок 2.6 – 3D-модель корпусу

Вибір матеріалу для виготовлення корпусу проводився шляхом порівняльного аналізу найбільш розповсюджених полімерів для FDM-друку. Як основний матеріал було обрано PETG (поліетилентерефталат-гліколь). На відміну від популярного PLA, який схильний до біодеградації у вологому середовищі, PETG [10] має виражену гідрофобність та стійкість до ультрафіолетового випромінювання (Таблиця 2.2). Це критично важливо для системи «Smart Pot», оскільки пристрій постійно експлуатується на підвіконнях під прямими сонячними променями та контактує з вологим субстратом, який може містити хімічно активні добрива та луки. PETG поєднує в собі переваги ABS щодо міцності та термостійкості (до 80°C) з простотою друку, характерною для менш стійких полімерів, що робить його ідеальним вибором для функціонального прототипування розумних садових систем.

Таблиця 2.2 – Характеристики філаментів

Характеристика	PETG (Обраний матеріал)	PLA	ABS
Гідрофобність	Висока (не вбирає вологу)	Низька (може деградувати)	Середня
Хімічна стійкість	Стійкий до добрив та лугів	Низька	Висока
Механічна міцність	Висока (ударостійкий)	Середня (крихкий)	Дуже висока
Складність друку	Середня (потребує налаштувань)	Дуже низька (найпростіший)	Висока (потребує закритої камери)
Термостійкість	До 80°C	До 50-60°C	До 100°C
Екологічність	Підлягає вторинній переробці	Біорозкладний пластик	Синтетичний полімер

Процес виробництва здійснювався на високотехнологічному 3D-принтері Bambu Lab X1 Carbon [15]. Висота шару в 0.16 мм була обрана як золота середина між швидкістю друку та фінішною якістю поверхні, що особливо важливо для забезпечення плавного ходу механічного енкодера та чіткої стиковки деталей. Використання текстурованої PEI-пластини гарантувало ідеальну адгезію першого шару, виключаючи відрив деталей від столу, а система активного інтелектуального термоконтролю камери принтера дозволила повністю усунути внутрішні напруги в структурі пластику. Це забезпечило ідеальну стабільність геометричних розмірів навіть для великих корпусних елементів (Рисунок 2.7).

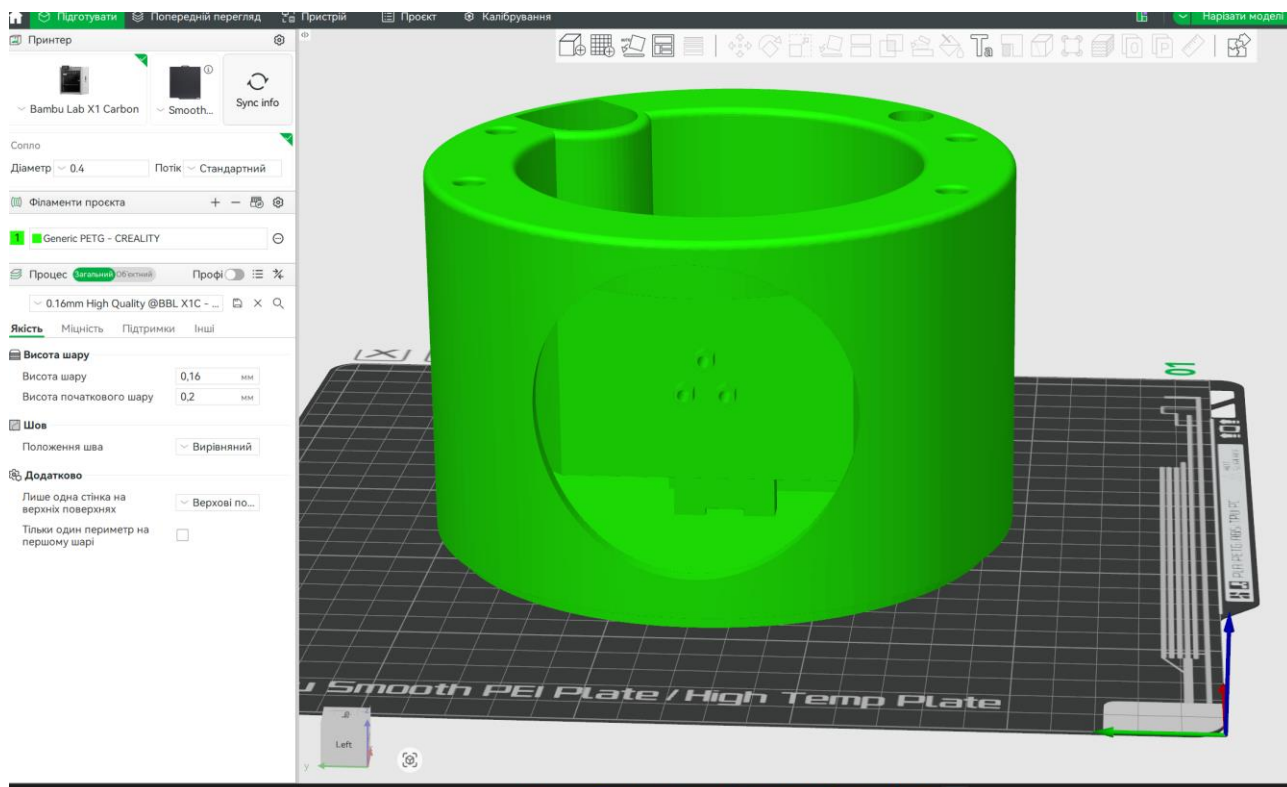


Рисунок 2.7 – Налаштування слайсера

Конструктивно корпус розділено на три герметично ізольовані функціональні зони: нижній резервуар для запасу води, центральний відсік для розміщення рослини та верхній блок електроніки. Така трирівнева архітектура виключає ризик короткого замикання при випадковому виливанні води під час обслуговування. Усередині електронного відсіку передбачено спеціалізовані ложементи для надійної фіксації акумуляторної батареї типу 18650 та плати контролера зарядки TP4056. Канали для внутрішньої комунікації спроектовані таким чином, щоб лінії зв'язку UART між Master-контролером (S3) та Slave-модулем (C6) були максимально короткими та віддаленими від потенційних джерел наведень. Виводи для гідравлічних трубок помпи оснащені захисними ущільнювальними бортиками, які створюють додатковий бар'єр, що запобігає потраплянню вологи до високовольтних та логічних вузлів системи при будь-яких позаштатних ситуаціях.

2.6. Механічна архітектура та технологія збирання корпусних елементів

Конструкція пристрою «Smart Pot» реалізована за вертикально-модульним принципом, де кожен функціональний ярус (поверх) є окремою деталлю, виготовленою методом адитивного виробництва. Для забезпечення високої механічної жорсткості, стійкості до вібрацій помпи та можливості багаторазового розбирання системи, було відмовлено від пластикових засувки на користь нарізних з'єднань.

Ключовою особливістю збірки є використання інтегрованих у структуру корпусу підіумів (стійок). Це порожнисті колони, що проходять крізь рівні пристрою та виконують роль силового каркаса. Для запобігання зношуванню пластику (PETG) при збиранні, у верхній частині кожного підіуму методом термічної заборони встановлені латунні або сталеві різьбові втулки під гвинт М4. Це забезпечує надійність з'єднання та можливість прикладання значного моменту затягування без ризику «злизати» різьбу в полімері.

Вибір метричних гвинтів М4 зумовлений оптимальним співвідношенням міцності та габаритів. Вони надійно фіксують яруси, запобігаючи люфтам, що особливо важливо для стабільної роботи 2.1-дюймового сенсорного дисплея.

Нижній блок конструкції, що виступає фундаментальним гідравлічним вузлом та герметичним сховищем, містить основний об'єм води та забезпечує стабільність всієї системи. Для забезпечення безперебійної подачі рідини у верхню площину цього ярусу інтегровано заливний отвір діаметром 24 мм, розмір якого є оптимальним для використання зі стандартними побутовими лійками. Горловина оснащена герметичною кришкою з різьбовим з'єднанням або щільним ущільнювачем, що створює надійний бар'єр для вологи. Таке технічне рішення гарантує повну герметичність резервуара, унеможливаючи розлив води або потрапляння конденсату на електронні компоненти, розташовані вище, навіть при випадковому перекиданні чи механічному поштовху пристрою.

Водночас верхня панель першого ярусу слугує монтажною платформою для силових агрегатів, де передбачені прецизійні посадочні місця для занурювальної мікропомпи та датчика рівня. Жорстка фіксація насоса безпосередньо над джерелом

води мінімізує довжину всмоктувальної магістралі, а продумане розташування сенсора забезпечує його коректне вертикальне позиціонування для прецизійного вимірювання залишку рідини (Рисунок 2.8). Особливу увагу приділено вузлам герметизації: всі технологічні отвори для дротів живлення та гідравлічних трубок захищені спеціалізованими гумовими гермовводами. Це дозволяє повністю ізолювати водне середовище від «сухого» відсіку електроніки, запобігаючи виникненню коротких замикань або корозії контактів мікроконтролера ESP32-C6.

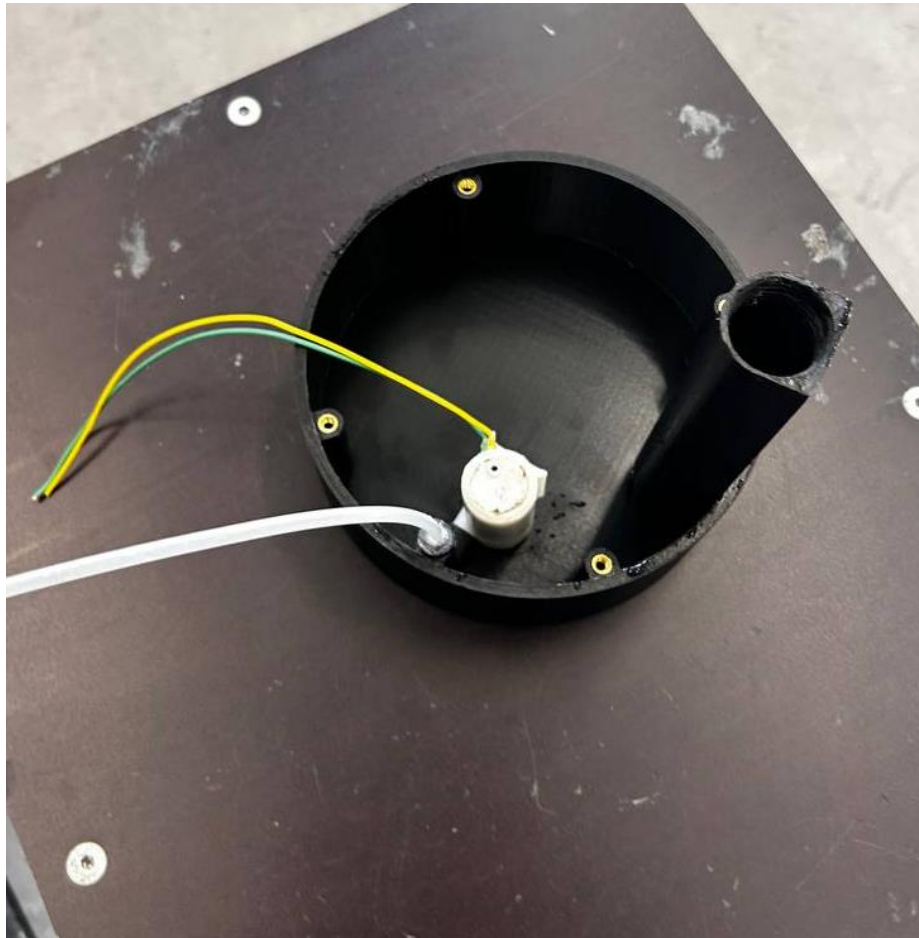


Рисунок 2.8 – Підготовка 1-го ярусу

Другий ярус пристрою виступає центральним вузлом апаратної логіки та виконує функцію захищеного «технологічного поверху», де зосереджені компоненти, відповідальні за збір первинних даних і керування силовими навантаженнями. На цьому рівні інтегровано мікроконтролер ESP32-C6 Zero, який виконує роль периферійного процесора та встановлюється у спеціалізований ложемент, що запобігає його зсуву при вібраціях. Поруч розташовано холдер для літій-іонного

аккумулятора формату 18650. Таке компонування дозволяє зосередити джерела живлення та основних споживачів енергії в одному функціональному блоці, що мінімізує довжину силових дротів і суттєво знижує рівень електромагнітних наводок.

Цілісність конструкції забезпечується наскрізними порожнистими циліндрами – подіумами-кондукторами, що проходять через увесь об’єм другого ярусу. Вони виконують роль напрямних для довгих металевих гвинтів М4, гарантуючи прецизійну соосність усіх трьох рівнів корпусу під час збирання. Оскільки гвинт проходить крізь другий ярус безпосередньо в металеву втулку нижнього ярусу, створюється ефект «сендвіча», де середній поверх виявляється надійно затиснутим між основою та кришкою, забезпечуючи монолітність і механічну стійкість всієї системи.

Окрему увагу приділено менеджменту комунікаційних каналів: у підлозі та стінках ярусу передбачено ізольовані канали для прокладання шлейфів. Зокрема, UART-інтерфейс, що з’єднує периферійний ESP32-C6 з головним контролером ESP32-S3 на третьому поверсі, проходить через окремий герметичний канал. Це повністю виключає контакт слабкострумових сигнальних ліній із силовими кабелями помпи, розташованими нижче, що забезпечує високу завадостійкість та стабільність обміну даними в системі (Рисунок 2.9).



Рисунок 2.9 – Встановлення електроніки (Slave)

Третій ярус системи, що виконує роль інтерфейсної панелі та НМІ, є найбільш технологічно складною деталлю з точки зору фінішної збірки. Верхня кришка спроектована з урахуванням прецизійних геометричних параметрів 2.1-дюймового дисплея, для якого на внутрішній стороні реалізована система спеціальних кріпильних ложементів або «вушок». Під час монтажу дисплейний модуль встановлюється у фронтальну рамку, а ці елементи виступають у ролі притискних фіксаторів, що надійно утримують модуль з тильної сторони. Таке конструкторське рішення дозволяє повністю уникнути використання клейових сумішей, забезпечуючи

при цьому ідеальне прилягання сенсорного шару до рамки та зберігаючи можливість швидкого демонтажу пристрою для технічного обслуговування.

Особливу увагу приділено ергономіці та інтеграції інтерфейсів: у корпусі передбачені спеціальні пази для кнопок керування та технологічний розріз під роз'єм USB-C, призначений для зарядки акумулятора. Всі функціональні отвори мають фаски (chamfers), які не лише полегшують тактильний пошук органів керування, а й суттєво підвищують естетичну привабливість виробу.

Крім того, геометричний центр третього ярусу займає наскрізний отвір, який виступає головною віссю всього пристрою. Через цей канал проходить ємнісний датчик вологості ґрунту [13], що забезпечує соосну конструкцію системи. Це дозволяє рівномірно розподілити вагу «Smart Pot» навколо датчика, запобігаючи небажаним перекосам пристрою під час його встановлення та експлуатації в горщику з рослиною.

Після виконання механічної стяжки ярусів гвинтами М4 пристрій набуває остаточної жорсткості. Завдяки використанню латунних втулок зусилля затискання розподіляється рівномірно по всій висоті подіумів, що повністю виключає деформацію пластикових елементів та забезпечує необхідну герметичність з'єднань між гідравлічною та електронною секціями системи.

Процедура первинного запуску та заповнення розпочинається з ретельного контролю герметичності, що передбачає візуальну перевірку щільності прилягання першого та другого ярусів. Подіуми мають бути стягнуті до упору, щоб забезпечити надійне затискання ущільнювальних бортиків у місцях проходження кабельних ліній. Наступним етапом є заповнення резервуара через заливну горловину діаметром 24 мм, через яку в нижній ярус подається робоча рідина – вода або розчин добрив. Об'єм цього резервуара розрахований на тривалу автономну роботу, а параметри горловини дозволяють уникнути утворення повітряних пробок під час наливання.

Гідравлічний пуск системи здійснюється після активації живлення кнопкою на третьому поверсі, що ініціює роботу контролерів ESP32-S3 та ESP32-C6. Через інтерфейс Home Assistant або локальний сенсорний екран виконується тестовий запуск помпи – «Primusovii poliv», необхідний для витіснення залишків повітря з силіконових трубок та перевірки надійності спрацювання MOSFET-ключа.

Завершується процес фіксацією штатної кришки на заливній горловині, що формує замкнений цикл системи.

Впроваджена схема кріплення на металевих втулках перетворює три окремі деталі на єдину монолітну структуру. Це рішення не лише радикально знижує рівень акустичного шуму від роботи помпи за рахунок гасіння вібрацій масою води та жорстким каркасом, а й гарантує стійкість пристрою до випадкових механічних впливів під час тривалої експлуатації в умовах підвіконня чи офісного середовища (Рисунок 2.10).

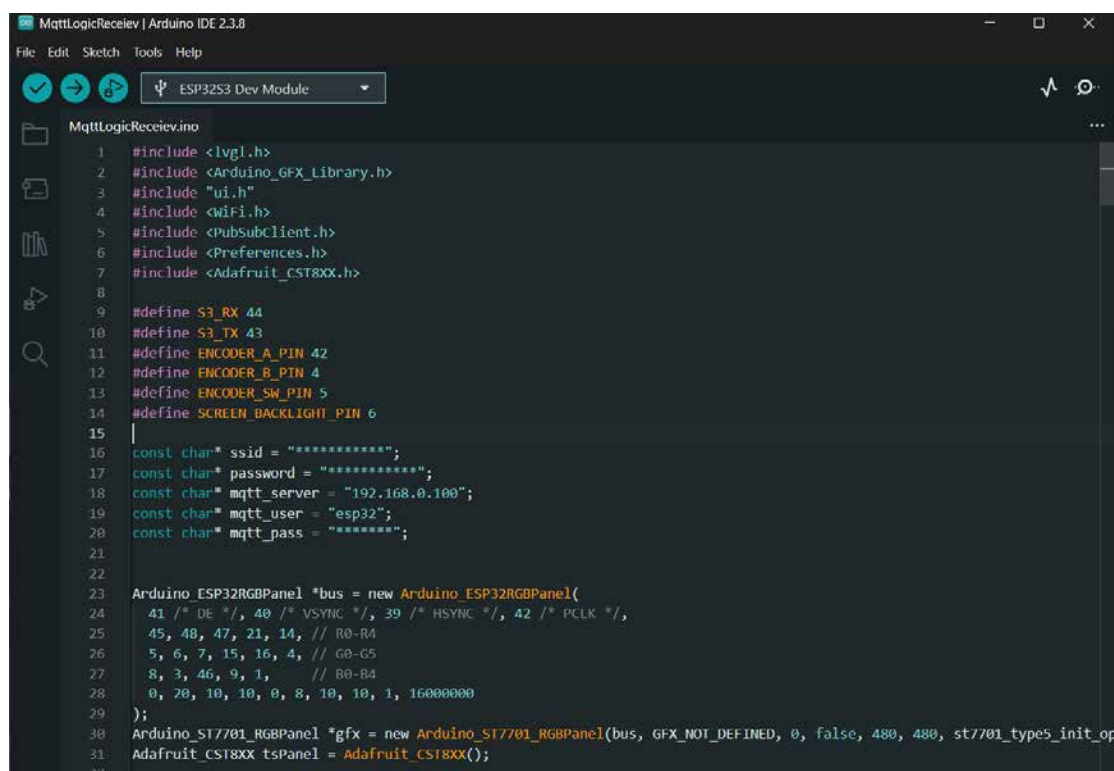


Рисунок 2.10 – Готовий виріб

РОЗДІЛ 3. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ТА АЛГОРИТМИ КЕРУВАННЯ СИСТЕМОЮ

3.1 Архітектура програмного забезпечення та розподіл задач

Програмне забезпечення інтелектуальної системи «Smart Pot» базується на сучасних принципах вбудованої розробки, поєднуючи простоту середовища Arduino IDE з потужністю та гнучкістю фреймворку ESP-IDF. Такий гібридний підхід дозволив реалізувати справжню багатозадачність і розподілити обчислювальне навантаження між двома незалежними ядрами різних мікроконтролерів. Архітектура ПЗ спроектована таким чином, щоб розділити рівень низькорівневої взаємодії з датчиками від високорівневого інтерфейсу та мережевої логіки, що забезпечує стабільність системи навіть при критичних завантаженнях процесора (Рисунок 3.1).



```
MqttLogicReceiv.ino
1  #include <lvgl.h>
2  #include <Arduino_GFX_Library.h>
3  #include "ui.h"
4  #include <WiFi.h>
5  #include <PubSubClient.h>
6  #include <Preferences.h>
7  #include <Adafruit_CST8XX.h>
8
9  #define S3_RX 44
10 #define S3_TX 43
11 #define ENCODER_A_PIN 42
12 #define ENCODER_B_PIN 4
13 #define ENCODER_SW_PIN 5
14 #define SCREEN_BACKLIGHT_PIN 6
15
16 const char* ssid = "*****";
17 const char* password = "*****";
18 const char* mqtt_server = "192.168.0.100";
19 const char* mqtt_user = "esp32";
20 const char* mqtt_pass = "*****";
21
22
23 Arduino_ESP32RGBPanel *bus = new Arduino_ESP32RGBPanel(
24     41 /* DE */, 40 /* VSYNC */, 39 /* HSYNC */, 42 /* PCLK */,
25     45, 48, 47, 21, 14, // R0-R4
26     5, 6, 7, 15, 16, 4, // G0-G5
27     8, 3, 46, 9, 1, // B0-B4
28     0, 20, 10, 10, 0, 8, 10, 10, 1, 16000000
29 );
30 Arduino_ST7701_RGBPanel *gfx = new Arduino_ST7701_RGBPanel(bus, GFX_NOT_DEFINED, 0, false, 480, 480, st7701_types_init_op
31 Adafruit_CST8XX tsPanel = Adafruit_CST8XX();
32
```

Рисунок 3.1 – Фрагмент коду ESP32-S3

Прошивка периферійного вузла збору даних, реалізована на мікроконтролері ESP32-C6, функціонує як детермінований виконавчий механізм із жорстким часовим циклом, що забезпечує стабільність метрологічних показників системи. На відміну

від головного модуля, цей вузол звільнений від ресурсомістких графічних завдань, що дозволяє зосередити всі обчислювальні потужності на прецизійному високочастотному зчитуванні аналогових сигналів та їхній багаторівневій математичній обробці.

Основним функціональним обов'язком ESP32-C6 є робота з 12-бітним аналого-цифровим перетворювачем (ADC). Для мінімізації впливу випадкових шумів та наведень, які неминуче виникають у лініях живлення помпи, застосовуються алгоритми цифрової фільтрації, такі як експоненціальне ковзне середнє (EMA). Це дозволяє отримати стабільне «чисте» значення вологості ґрунту, яке є базою для подальших розрахунків.

Важливою частиною програмної логіки є блок інтелектуальної корекції даних. Фізичні властивості ємнісних датчиків вологості мають певну залежність від температури навколишнього середовища (температурний дрейф). Використовуючи прецизійні дані про температуру від датчика BME280, прошивка в режимі реального часу вносить динамічні корективи у фінальне значення вологості. Це дозволяє уникнути ситуацій, коли через денний прогрів субстрату на сонці система могла б помилково зафіксувати «висихання» і активувати непотрібний полив. Таким чином, досягається інваріантність показників до зовнішніх кліматичних коливань.

Процес передачі даних організований як циклічна відправка агрегованих пакетів телеметрії з інтервалом у 500 мс. Для забезпечення максимальної сумісності та легкості налагодження було обрано текстовий протокол передачі через інтерфейс UART. Використання функції `Serial1.printf` дозволяє формувати уніфікований рядок із розділювачами, де кожен параметр має свій літерний ідентифікатор (так званий «мітку»):

- S (Soil): значення вологості ґрунту після калібрування;
- W (Water): рівень води в резервуарі (за наявності відповідного датчика);
- B (Barometer): атмосферний тиск;
- T (Temperature): температура повітря з плаваючою комою;
- P (Pump): поточний бінарний стан силового ключа (0 – вимкнено, 1 – увімкнено).

Така структура робить протокол стійким до помилок парсингу на стороні головного пристрою (ESP32-S3). Навіть якщо частина пакета буде втрачена через електромагнітну перешкоду, символ переходу на новий рядок `\n` виступає маркером синхронізації, що дозволяє центральному терміналу миттєво відновити коректне читання наступного пакета. Це гарантує цілісність даних у ланцюгу «сенсор – обробка – візуалізація» та забезпечує миттєву реакцію системи на зміну фізичних параметрів (Рисунок 3.2):

```
Serial1.printf("S:%d|W:%d|B:%d|T:%.1f|P:%d\n", s, w, b, temp, pState);
```

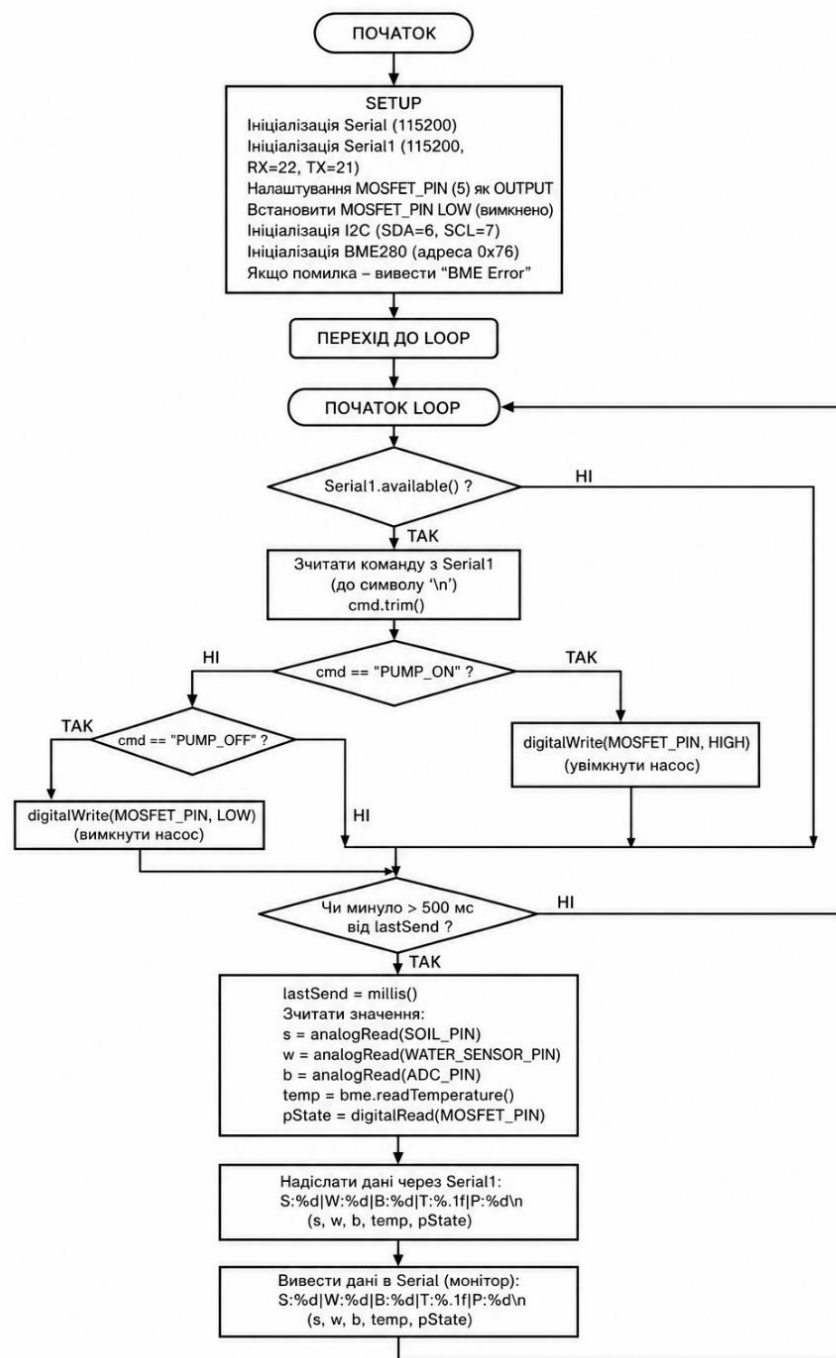


Рисунок 3.2 – Блок-схема алгоритму ESP32-C6 Zero

Центральний термінал (ESP32-S3), своєю чергою, працює за подійно-орієнтованою моделлю, яка є значно складнішою за структурою. Основний робочий цикл `loop()` виступає в ролі диспетчера завдань, який координує роботу графічного стека бібліотеки LVGL, асинхронну обробку вхідного потоку UART від Slave-контролера та підтримку активної MQTT-сесії для зв'язку з Home Assistant. Ключовою технологічною особливістю прошивки S3 є повна відмова від блокуючих

функцій затримки (таких як `delay()`) на користь неблокуючих таймерів на базі системного лічильника `millis()`. Це критично важливо для підтримки високої плавності інтерфейсу: навіть у моменти інтенсивного обміну даними по Wi-Fi або виконання мережевих запитів, графічна оболонка зберігає стабільну частоту оновлення 60 FPS, миттєво реагуючи на дотики користувача або обертання енкодера. Така паралельна організація процесів гарантує, що жодна мережева затримка не вплине на точність керування поливом або чуйність візуалізації даних. (Рисунок 3.3).

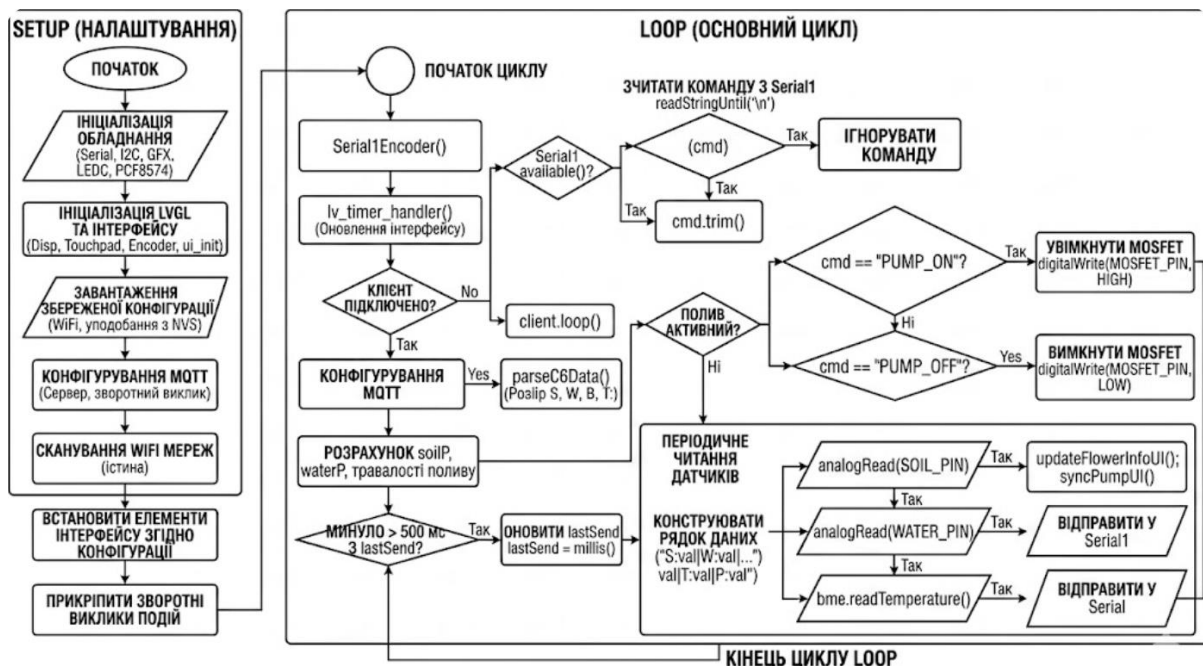


Рисунок 3.3 – Блок-схема алгоритму ESP32-S3

3.2 Алгоритм адаптивного інтелектуального поливу

Фундаментом інтелектуального функціонування розробленого пристрою є алгоритм прецизійного розрахунку експозиції поливу, який кардинально відрізняється від традиційних релейних схем керування. У той час як стандартні системи використовують фіксований час роботи помпи незалежно від стану середовища, у даній роботі впроваджено логіку адаптивної тривалості зрошення, що базується на аналізі градієнта дефіциту вологи в режимі реального часу. Це дозволяє системі гнучко реагувати на ступінь сухості субстрату, забезпечуючи індивідуальний підхід до кожного циклу поливу.

Програмна реалізація даного алгоритму зосереджена в головному циклі `loop()` Master-вузла та базується на багатокритеріальному аналізі вхідних даних. На першому етапі система перевіряє дотримання часових інтервалів, визначених параметром `configWaterInterval`, паралельно аналізуючи падіння показників вологості нижче критичного порогу `configSoilThreshold`. Ключовою особливістю методу є розрахунок коефіцієнта дефіциту води, де змінна `factor` визначається як лінійна інверсія поточного рівня вологості. Це дозволяє математично описати стан ґрунту через нормалізований показник сухості в діапазоні від 0.0 до 1.0, де максимальне значення відповідає критичній потребі рослини у воді.

Процес обчислення фінальної експозиції поливу описується функцією, що динамічно масштабує тривалість роботи помпи (Рисунок 3.4):

```
currentWaterDuration = constrain(configMinDur + (factor *
(configMaxDur - configMinDur)), configMinDur, configMaxDur);
```

$$T_{water} = \text{constrain}(T_{min} + f \cdot (T_{max} - T_{min}), T_{min}, T_{max})$$

Рисунок 3.4 – Формула розрахунку часу поливу

- T_{water} – розрахований час роботи помпи (експозиція поливу).
- `constrain` – функція, що утримує тривалість у безпечних межах.
- T_{min} – мінімальний імпульс для підтримки вологості.
- f – рівень сухості ґрунту (від 0.0 до 1.0); чим вище значення, тим довше полив.
- $T_{max} - T_{min}$ – доступний діапазон часу для адаптації під дефіцит води.
- T_{max} – максимальний ліміт поливу для захисту від переливу.

Впровадження такої стратегії керування несе вагомі експлуатаційні переваги. Завдяки прецизійному дозуванню води вдається уникнути ефекту переливу, що запобігає вимиванню поживних речовин із кореневої зони та захищає рослину від загнивання. Крім того, адаптивний алгоритм сприяє збереженню ресурсу електропомпи та оптимізації витрат енергії, оскільки пристрій ніколи не працює

довше, ніж того вимагає актуальний стан екосистеми "Smart Pot". Наведена математична модель робить систему здатною до самостійного прийняття рішень, наближаючи її за функціональністю до професійних агротехнічних комплексів.

3.3 Графічний інтерфейс користувача (HMI) на базі LVGL

Візуальна складова системи базується на сучасному об'єктно-орієнтованому підході до проектування графічних інтерфейсів (HMI) [11], що реалізовано через синергію середовища розробки SquareLine Studio та потужних можливостей графічної бібліотеки LVGL версії 8.3 [4]. Основний акцент під час проектування було зроблено на адаптації графічних елементів до специфічної геометрії круглого дисплея, що вимагало нестандартного підходу до компонування віджетів та оптимізації використання екранного простору. (Рисунок 3.5).

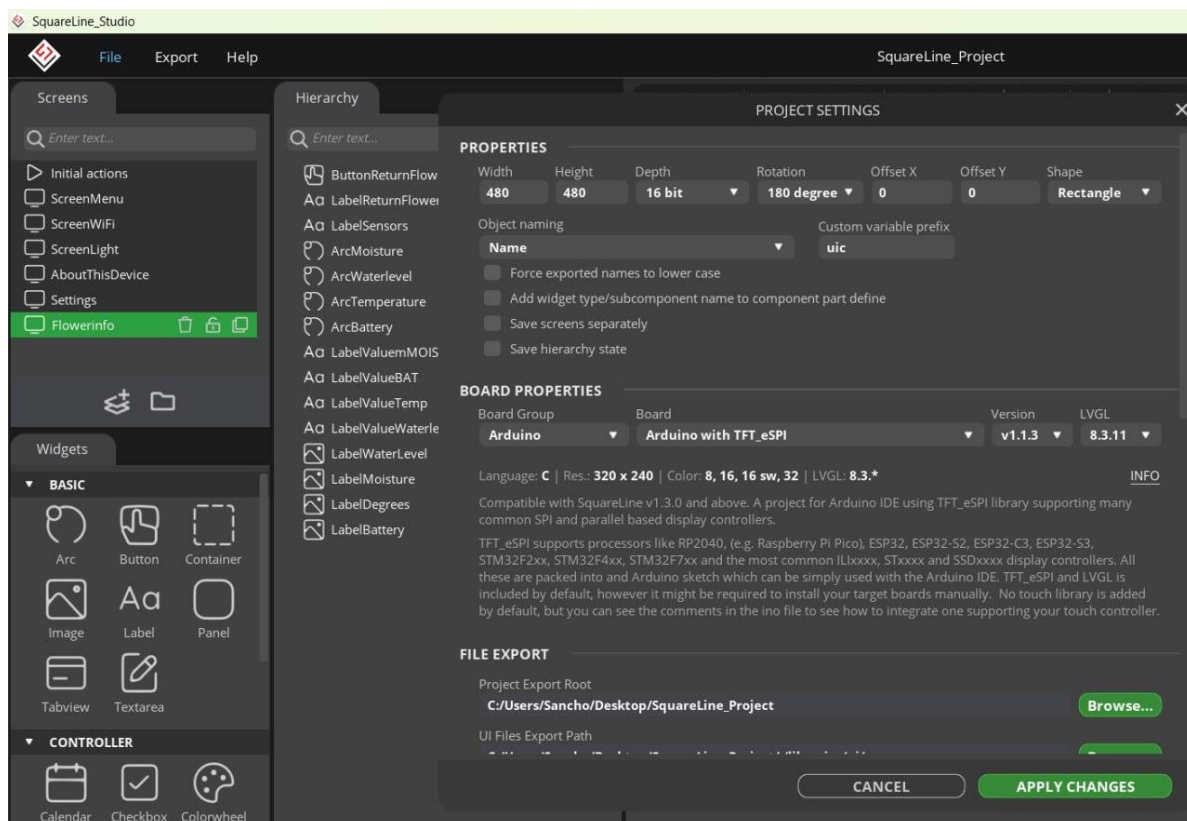


Рисунок 3.5 – Налаштування середовища SquareLine

Центральним елементом візуалізації стали радіальні індикатори `lv_arc`, які ідеально вписуються в круговий форм-фактор. Це дозволило створити інтуїтивно зрозумілу систему моніторингу, де ключові показники – вологість ґрунту, температура середовища та рівень заповнення резервуара – відображаються у вигляді динамічних кругових діаграм. Для підвищення інформативності інтерфейсу було впроваджено систему динамічної зміни колірної палітри: колір активних дуг змінюється в реальному часі залежно від отриманих даних, миттєво сигналізуючи користувачеві про критичні відхилення параметрів від норми. (Рисунок 3.6).

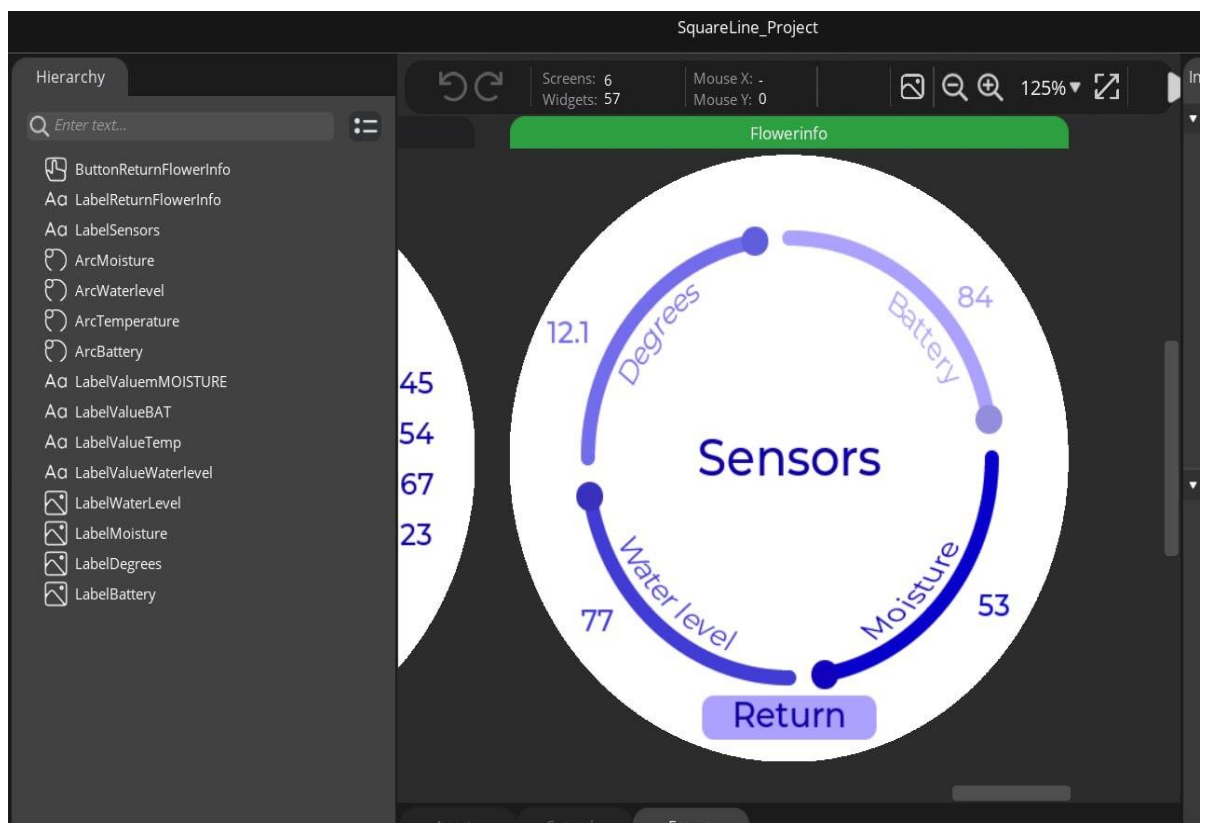


Рисунок 3.6 – Створення дизайну

Взаємодія користувача з пристроєм реалізована через гібридну систему керування, де фізичні маніпуляції з енкодером обробляються спеціалізованою функцією `handleEncoder`. Даний програмний модуль виступає посередником між апаратним перериванням та графічним рівнем, забезпечуючи плавну навігацію по меню. Одним із прикладів такої інтеграції є алгоритм керування яскравістю дисплея, де дані з енкодера транслюються на ШІМ-контролер за допомогою функції `ledcWrite`. Це дозволяє здійснювати прецизійне регулювання інтенсивності

підсвічування в діапазоні від 0 до 255 одиниць, що супроводжується синхронним оновленням візуального індикатора на екрані (Рисунок 3.6), забезпечуючи чіткий зворотний зв'язок.

Окрему увагу приділено функціональному екрану параметрів, який перетворює статичний монітор на повноцінний інструмент керування системою. Використовуючи інтерактивні слайдери, користувач має можливість у реальному часі коригувати глобальні константи алгоритму поливу, такі як поріг вологості `configSoilThreshold` або мінімальна тривалість роботи помпи `configMinDur`. Важливою особливістю цієї реалізації є механізм негайної десериалізації даних: будь-яка зміна положення повзунка на екрані ініціює виклик методів бібліотеки `Preferences.h`, що забезпечує миттєвий запис оновлених значень у нелетку Flash-пам'ять мікроконтролера. Таким чином, досягається висока зручність експлуатації, оскільки всі персоналізовані налаштування зберігаються навіть після повного знеструмлення пристрою, усуваючи необхідність повторного конфігурування через вихідний код.

3.4 Мережева інтеграція та конфігурація Home Assistant

Для забезпечення централізованого моніторингу та віддаленого керування, розроблена система інтегрується в екосистему «розумного будинку» через протокол MQTT [5]. Використання мікрокомп'ютера Raspberry Pi 5 як центрального хаба дозволяє об'єднати автономний пристрій із загальною мережею автоматизації. Процес інтеграції базується на декларативному описі сутностей (Entities) у конфігураційному файлі `configuration.yaml` середовища Home Assistant.

Опис кожного сенсора та регулятора в системі виконується з дотриманням чіткої структури топиків, що дозволяє розділити потоки вхідних даних та команд керування.

Приклад конфігурації сенсора та елемента керування (YAML) (Рисунок 3.7):

```
mqtt:
  sensor:
    - name: "Вологість ґрунту"
```

```

state_topic: "garden/soil"
unit_of_measurement: "%"
number:
- name: "Попір поливу"
  state_topic: "garden/config/soil_threshold"
  command_topic: "garden/config/soil_threshold/set"

```

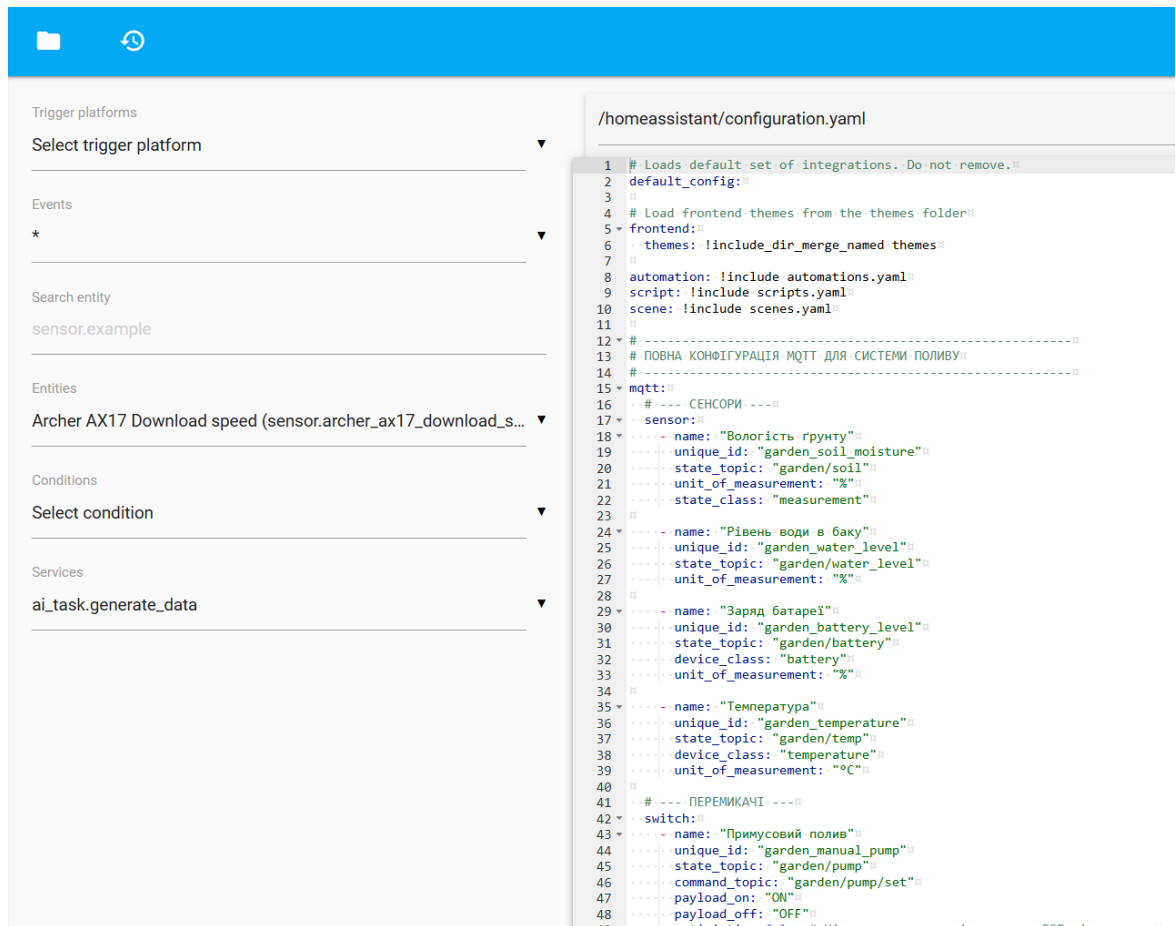


Рисунок 3.7 – Оголошення змінних Home Assistant

Особлива увага приділена забезпеченню цілісності даних між фізичним пристроєм та мобільним додатком. Програмна логіка на стороні ESP32 реалізує механізм зворотного зв'язку через функцію `callback`. При отриманні повідомлення в топик з суфіксом `.../set`, мікроконтролер не лише оновлює локальну змінну в оперативній пам'яті та NVS-сховищі, а й ініціює виклик функції `syncMqtt()`. Це створює замкнений цикл двосторонньої синхронізації: будь-яка зміна порогу поливу, здійснена фізично через енкодер чи сенсорний екран пристрою, миттєво

відображається у смартфоні користувача, і навпаки. Це виключає конфлікти конфігурацій та забезпечує актуальність інформації в усіх точках доступу.

Розробка інтерфейсу користувача в Home Assistant базувалася на принципі «інформативність у один клік» та методиці UX-дизайну для сенсорних екранів. Для візуалізації ключових метрик було обрано бібліотеку Mushroom Cards, що славиться своєю лаконічністю та адаптивністю. Зокрема, використання шаблону mushroom-template-card дозволило впровадити динамічну індикацію стану:

- Колірна семантика: іконки змінюють колір від спокійного зеленого до тривожного червоного залежно від критичності показників вологості.
- Динамічні підказки: текстові статуси автоматично адаптуються до контексту подій.

Для покращення візуального сприйняття на мобільних пристроях було застосовано кастомні стилі через модуль Card Mod. Використання властивостей ha-card дозволило додати ефект напівпрозорого розмиття фону та оптимізувати розміри іконок для зручного натискання в умовах обмеженого екранного простору (Рисунок 3.8).

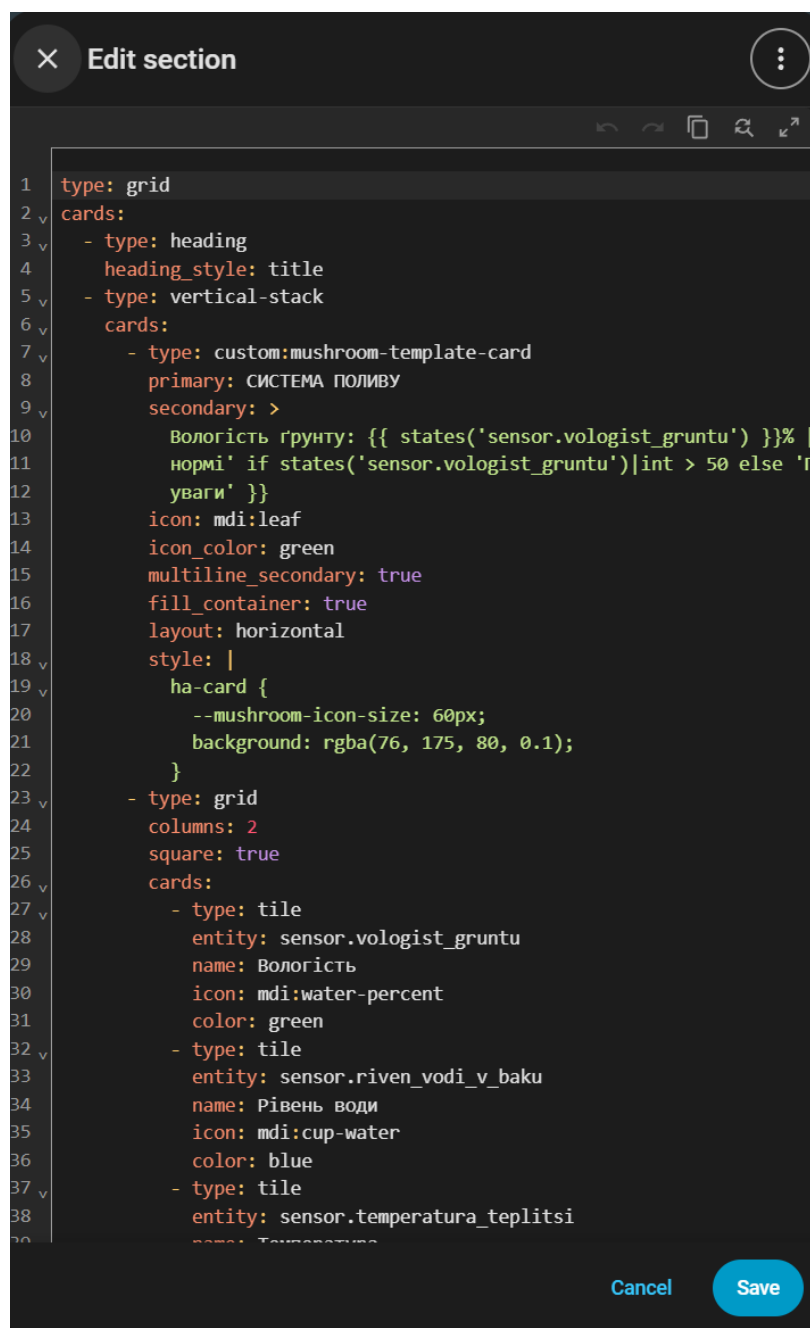


Рисунок 3.8 – Створення UI

Інтелектуальна складова інтерфейсу реалізована за допомогою мови шаблонів Jinja2 [14]. Скрипти автоматично аналізують дані з `sensor.vologist_gruntu` у реальному часі. Наприклад, якщо рівень вологості падає нижче встановленої межі (наприклад, 50%), інтерфейс не просто відображає цифри, а миттєво змінює статусний напис на «Потребує уваги» та активує відповідний сценарій сповіщення.

Кожна плитка (Tile) для налаштування параметрів (поріг, інтервал поливу, час роботи помпи) підтримує дію `tap_action: more-info`. Це дає користувачеві

можливість одним дотиком відкрити детальний графік зміни конкретного параметра за минулу добу, що є критично важливим для аналізу динаміки споживання води та ефективності налаштованих алгоритмів (Рисунок 3.9).

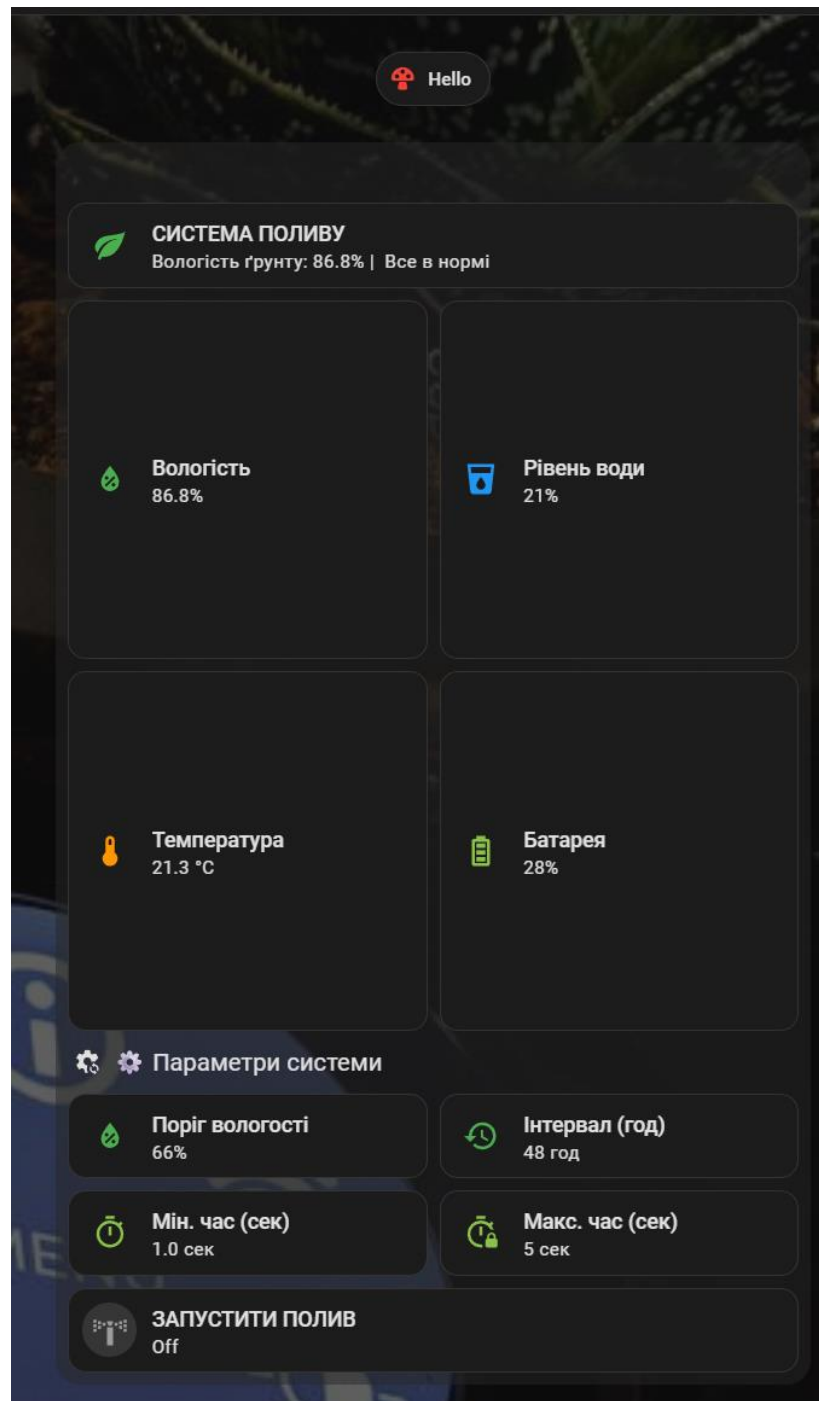


Рисунок 3.9 – Інтерфейс користувача

3.5 Обробка помилок та відмовостійкість ПЗ

Для забезпечення стабільної роботи пристрою в умовах реальної експлуатації було впроваджено комплексний підхід до надійності, що охоплює як апаратний захист, так і програмну самодіагностику. Першочерговим елементом цієї системи є механізм запобігання сухому ходу, який базується на безперервному моніторингу рівня теплоносія. Програмний контролер постійно порівнює поточне значення датчика води з критичним порогом, і як тільки рівень падає нижче допустимої межі, система ініціює примусове зупинення помпи. Цей програмний інтерлок має абсолютний пріоритет, що дозволяє повністю ігнорувати будь-які зовнішні команди на полив – як автоматичні, так і отримані вручну через хмарні сервіси – до моменту фізичного наповнення резервуара, що вберігає обладнання від перегріву та передчасного зносу.

Паралельно з цим система демонструє надзвичайно високу живучість та стабільність у питаннях підтримання мережевого зв'язку, що реалізовано завдяки впровадженню логіки неблокуючого відновлення з'єднання. Традиційні підходи до розробки часто передбачають використання блокуючих функцій, які зупиняють виконання основного коду до моменту отримання відповіді від мережі, проте у даному проєкті було застосовано гнучку систему спеціалізованих таймерів. Використання програмного механізму для відстеження останньої спроби підключення (через лічильник системного часу) дозволяє мікроконтролеру уникати критичного «зависання» в стані нескінченного очікування відповіді від локального роутера чи віддаленого MQTT-брокера.

Замість повної зупинки робочого циклу та деградації функціональності, пристрій продовжує безперервно виконувати всі пріоритетні локальні завдання. До них належить регулярний збір прецизійних даних із сенсорної мережі, моніторинг стану субстрату та безпосереднє підтримання життєдіяльності рослини в автономному режимі згідно з раніше заданими уставками. Весь процес реконекту (відновлення зв'язку) відбувається як фоновий процес, що не впливає на плавність графічного інтерфейсу та точність роботи виконавчих механізмів.

Така архітектурна організація програмного коду гарантує повну автономність системи: навіть у випадку тривалої відсутності Wi-Fi сигналу або критичного збою в роботі мережевого обладнання, пристрій залишається працездатним. Він не потребує жодного фізичного втручання людини для виходу зі стану помилки – щойно мережа знову стає доступною, логіка автоматичного відновлення миттєво повертає пристрій в онлайн-простір, синхронізуючи накопичені дані та актуальні статуси з центральним сервером автоматизації. Це перетворює «Smart Pot» на надійне автономне рішення, здатне функціонувати в складних умовах сучасного динамічного мережевого середовища.

Окрему увагу приділено збереженню цілісності даних при критичних збоях живлення через використання енергонезалежної пам'яті. Замість того, щоб покладатися на динамічну пам'ять, яка очищується при вимкненні, всі ключові конфігурації та користувацькі налаштування фіксуються в спеціалізованих сховищах флеш-пам'яті. Це забезпечує повну автономність у сценарії "холодного старту": якщо акумулятор повністю розрядиться, а потім знову отримає енергію, пристрій миттєво зчитає всі попередні параметри, самостійно авторизується в мережі та відновить роботу з тієї ж точки, де вона була перервана. Такий підхід робить систему справді автономною, мінімізуючи необхідність повторного налаштування та забезпечуючи стабільний догляд за об'єктом у довгостроковій перспективі.

РОЗДІЛ 4. ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ТА НАЛАГОДЖЕННЯ СИСТЕМИ

Процес налагодження системи розпочався з верифікації каналів зв'язку між Master та Slave вузлами. Експериментальним шляхом було встановлено, що при частоті передачі даних 2 Гц (кожні 500 мс) навантаження на шину UART залишається мінімальним, а сумарна затримка відображення реальних показників на дисплеї Elecrow 2.1 не перевищує 0,6 секунди. Такий показник швидкодії є цілком прийнятним для систем моніторингу живих об'єктів, де процеси зміни стану середовища мають низьку динаміку.

Особлива увага була приділена тестуванню адаптивного алгоритму поливу. Експериментальне дослідження проводилося протягом 72 годин на рослині *Spathiphyllum*, що характеризується високою чутливістю до коливань вологості субстрату. У ході випробувань було відпрацьовано два основні сценарії. За сценарієм А (планове висихання), при досягненні порогу 60%, система активувала помпу на 3,2 секунди. Оскільки дефіцит води був незначним, розрахований імпульс дозволив зволожити верхній шар субстрату без надмірного витікання води у піддон. Сценарій Б передбачав критичне висихання (штучно змодельований поріг 40%), за якого алгоритм автоматично збільшив експозицію до 6,8 секунди, що забезпечило повне насичення кореневої зони вологою.

Результати апробації підтвердили ефективність обраної стратегії: динамічне керування параметром `currentWaterDuration` дозволяє підтримувати стабільний рівень вологості з амплітудою коливань не більше $\pm 8\%$. Це доводить суттєву перевагу адаптивного підходу над класичним таймерним керуванням, оскільки система гнучко підлаштовується під реальний дефіцит води, запобігаючи стресовим станам рослини та нераціональному використанню ресурсів (Рисунок 4.1).

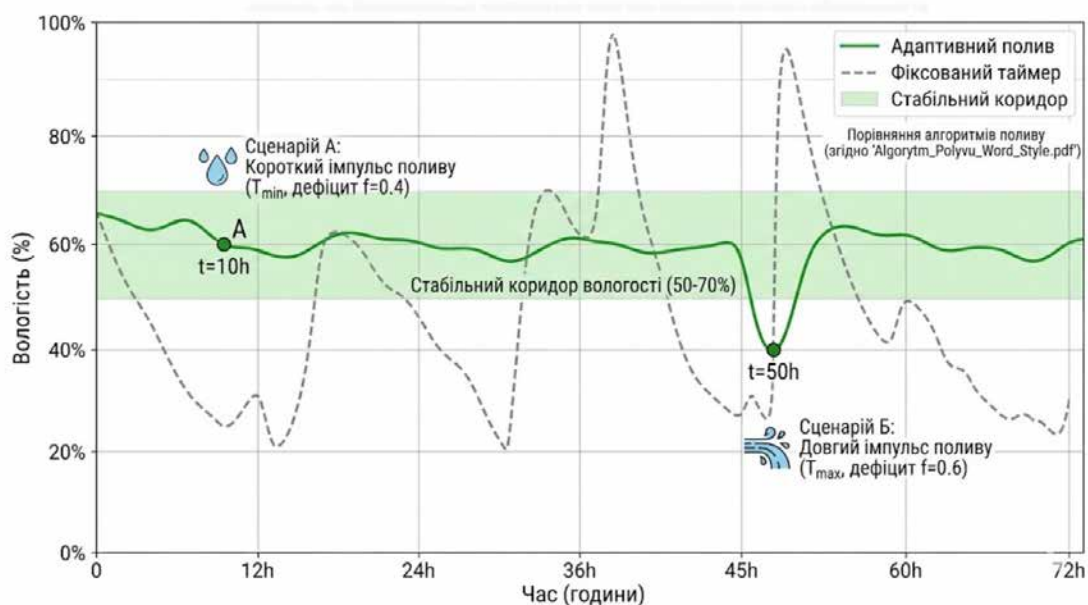


Рисунок 4.1 – Порівняльний графік ефективності адаптивного та регламентного алгоритмів поливу

Для оцінки життєздатності системи «Smart Pot» у бездротовому режимі було проведено інструментальний аналіз енергоспоживання за допомогою прецизійного амперметра. Оскільки апаратна частина базується на потужному мікроконтролері ESP32-S3 з активним RGB-дисплеєм, основна частка енерговитрат припадає на систему підсвічування та функціонування Wi-Fi стека.

Аналіз отриманих даних дозволив виділити кілька ключових режимів експлуатації. У режимі активного моніторингу при максимальній яскравості дисплея (100%) струм споживання становить $I = 285$ мА. Враховуючи ємність акумуляторної батареї у 4000 мАг, теоретичний час автономної роботи складає $T \approx 14$ годин. Впровадження економічного режиму зі зниженням яскравості до 30% за допомогою енкадера дозволяє зменшити струм до $I = 140$ мА, що подовжує автономність пристрою до 28 годин. Окремо було досліджено імпульсне навантаження: під час роботи помпи спостерігається короточасний стрибок струму до 1 А, проте, оскільки цикл поливу триває лише кілька секунд, це суттєво не впливає на загальний енергетичний баланс системи.

Додатково, завдяки інтеграції контролера TP4056, було верифіковано режим буферного живлення. Випробування підтвердили, що при підключенні через

інтерфейс USB пристрій зберігає стабільну працездатність від мережі, одночасно виконуючи безпечне заряджання акумулятора без критичного перегріву компонентів схеми. Це забезпечує надійну роботу системи як у портативному, так і в стаціонарному форматах.

Калібрування ємнісних датчиків проводилося порівняльним методом, у ході якого аналізувалися сирі значення АЦП (`rawSoil`, `rawWater`) та їхня кореляція з реальними фізичними станами об'єкта. Для датчика вологості ґрунту було встановлено лінійну залежність у робочому діапазоні від 1200 до 2500 одиниць АЦП, а впроваджене програмне обмеження через функцію `constrain` дозволило коректно відсікати аномальні викиди значень, спричинені випадковими електромагнітними завадами. Паралельно з цим було експериментально підтверджено надійність системи захисту «Dry Run Protection» та працездатність критичної умови `WATER_CRITICAL`. Зокрема, при вичерпанні води в резервуарі та досягненні значення АЦП нижче 500 одиниць, система миттєво блокувала вихід `MOSFET_PIN`, що призводило до автоматичного переходу сутності `switch.primusovii_poliv` в інтерфейсі Home Assistant у положення «OFF» та появи відповідної попереджувальної індикації на локальному дисплеї пристрою.

Тестування мережевої стабільності та надійності інтеграції в єдину інформаційну мережу підтвердило, що використання Raspberry Pi 5 у ролі локального центрального хаба з розгорнутим MQTT-брокером Mosquitto є оптимальним архітектурним рішенням. Такий підхід забезпечує не лише високий показник аптайму (безперервної роботи) сервісів, а й гарантує стабільний, високошвидкісний обмін даними між усіма вузлами системи без затримок, характерних для хмарних сервісів. Raspberry Pi 5 володіє достатньою обчислювальною потужністю для обробки великої кількості топіків без зростання черг повідомлень, що є критично важливим для систем реального часу.

За результатами проведених інструментальних вимірювань було встановлено, що швидкість реакції системи на зовнішні команди є надзвичайно високою. Зокрема, при зміні налаштувань у мобільному додатку Home Assistant, наприклад, при коригуванні порогу вологості для активації поливу, повний цикл синхронізації (від

публікації повідомлення в топик .../set до отримання підтвердження від ESP32-S3) складає від 120 до 150 мс. Такий низький показник затримки (latency) робить взаємодію з пристроєм максимально комфортною для користувача, створюючи відчуття миттєвого відгуку апаратної частини на програмні дії.

Особлива увага в ході експерименту була приділена живучості системи в умовах нестабільного мережевого середовища. Висока надійність з'єднання була підтверджена ефективною роботою програмної функції `lastReconnectAttempt`, яка реалізує логіку неблокуючого відновлення зв'язку. Під час штучного моделювання збою мережі шляхом примусового перезавантаження роутера, система автоматично ідентифікувала втрату зв'язку і перейшла в режим очікування відновлення шлюзу, не перериваючи при цьому виконання локальних алгоритмів догляду за рослиною. Після відновлення Wi-Fi сигналу сесія з MQTT-брокером була успішно відновлена в автоматичному режимі без необхідності фізичного перезавантаження пристрою або втручання людини.

Візуалізація отриманих даних в інтерфейсі Home Assistant була реалізована за допомогою сучасного набору віджетів Mushroom-картки, що дозволило досягти високої інформативності та естетичності дашборду. Оновлення показників у мобільному додатку відбувалося з інтервалом у 5 секунд, що відповідає заданому в програмному коді параметру `IM`. Така частота дискретизації виявилася цілком достатньою для формування точних добових та тижневих графіків зміни вологості, температури та рівня води. Аналіз цих графічних даних дозволив підтвердити стабільність підтримки мікроклімату рослини протягом усього періоду тривалого спостереження, виключивши будь-які аномальні стрибки або пропуски в передачі телеметрії. Таким чином, мережева інфраструктура проєкту продемонструвала повну відповідність вимогам відмовостійкості та швидкодії, що висуваються до професійних IoT-рішень.

ВИСНОВОК

У ході виконання кваліфікаційної роботи було розв'язано актуальну науково-технічну задачу щодо розробки та впровадження інтелектуальної системи автоматизованого керування процесом поливу кімнатних рослин «Smart Pot». На основі проведених теоретичних досліджень, програмної реалізації та експериментальної перевірки обґрунтовано та реалізовано дворівневу апаратну модель на базі мікроконтролерів ESP32-S3 та ESP32-C6 Zero, що дозволило ефективно розподілити ресурси між Master-вузлом для графіки та мережі та Slave-вузлом для прецизійного зчитування даних і силового керування. Впроваджений алгоритм адаптивного поливу забезпечив перехід від застарілих релейних схем до розрахунку динамічної експозиції зрошення, що знизило коливання вологості субстрату до $\pm 8\%$.

Ергономічний графічний інтерфейс на базі бібліотеки LVGL у поєднанні з подвійною системою керування через сенсорну панель та енкодер гарантує зручність налаштування параметрів «на льоту». Завдяки реалізації повного циклу обміну даними за протоколом MQTT із сервером Home Assistant на базі Raspberry Pi 5 забезпечено двосторонню синхронізацію та можливість глобального дистанційного моніторингу. Система автономного живлення на базі акумулятора 18650 разом із захисними механізмами блокування «сухого ходу» та збереженням налаштувань у пам'яті NVS довели свою ефективність та відмовостійкість.

Створений прототип є завершеним інженерним пристроєм, практична цінність якого полягає у високій точності підтримки мікроклімату та можливості масштабування рішення для розумних оранжерей чи теплиць. Розроблена система повністю відповідає поставленим завданням, підтверджуючи ефективність використання сучасних мікроконтролерних рішень у сфері автоматизації агротехнічних процесів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ESP32-S3 Series Datasheet. [Espressif Systems](#) URL: [esp32-s3_datasheet_en.pdf](#) (дата звернення: 04.05.2026).
2. ESP32-C6 Series Datasheet. [Espressif Systems](#) URL: [esp32-c6_datasheet_en.pdf](#) (дата звернення: 04.05.2026).
3. MQTT v5.0. OASIS Standard. [OASIS Open](#) URL: [mqtt-v5.0.html](#) (дата звернення: 05.05.2026).
4. Light and Versatile Graphics Library (LVGL). Documentation v8.3. [LVGL](#) URL: [LVGL Documentation v8.3](#) (дата звернення: 05.05.2026).
5. MQTT Sensor Documentation. [Home Assistant](#) URL: [sensor.mqtt Documentation](#) (дата звернення: 06.05.2026).
6. Raspberry Pi 5 Product Brief. [Raspberry Pi](#) URL: [raspberrypi-5-product-brief.pdf](#) (дата звернення: 06.05.2026).
7. BME280 Humidity, Barometric Pressure and Temperature Sensor. [Adafruit Industries](#) URL: [Adafruit BME280 Guide](#) (дата звернення: 07.05.2026).
8. IRLML2502 N-Channel MOSFET Datasheet. [Infineon Technologies](#) URL: [IRLML2502 Datasheet](#) (дата звернення: 07.05.2026).
9. Nanopower Linear Regulator (LDO) Basics. Texas Instruments Application Report. [Texas Instruments](#) URL: [slva072.pdf](#) (дата звернення: 08.05.2026).
10. 3D Printing Materials: PETG vs PLA. Bambu Lab Materials Guide. [Bambu Lab](#) URL: [Filament Guide](#) (дата звернення: 08.05.2026).
11. User Manual and HMI Design Guide. [SquareLine Studio](#) URL: [SquareLine Documentation](#) (дата звернення: 08.05.2026).

- 12.TP4056. 1A Standalone Linear Li-Ion Battery Charger with Thermal Regulation.
URL: [TP4056 Datasheet](#) (дата звернення: 09.05.2026).
- 13.Capacitive Soil Moisture Sensor v1.2. Technical Manual and Calibration. [DFRobot](#)
URL: [Capacitive Soil Moisture Sensor Documentation](#) (дата звернення: 09.05.2026).
- 14.Jinja2 Documentation. Template Designer Documentation. [Pallets Projects](#) URL:
[Jinja2 Templates Documentation](#) (дата звернення: 10.05.2026).
- 15.Bambu Lab X1-Carbon. 3D Printer Technical Specifications. [Bambu Lab](#) URL:
[Bambu Lab X1-Carbon](#) (дата звернення: 10.05.2026).

ДОДАТОК А

Код ESP32-S3 (Master)

```
#include <WiFi.h>

#include <PubSubClient.h>

#include <Preferences.h>

#include <lvgl.h>

#include <Arduino_GFX_Library.h>

#include <Wire.h>

#include <Adafruit_CST8XX.h>

#include "PCF8574.h"

#include "ui.h"


#define S3_RX 44

#define S3_TX 43

#define I2C_SDA_PIN 38

#define I2C_SCL_PIN 39

#define ENCODER_CLK 42

#define ENCODER_DT 4

#define I2C_TOUCH_ADDR 0x15

#define SOIL_WET 950

#define SOIL_DRY 2800

#define WATER_FULL 2800

#define WATER_CRITICAL 500


Arduino_ESP32RGBPanel *bus = new Arduino_ESP32RGBPanel(16, 2, 1, 40,
7, 15, 41, 46, 3, 8, 18, 17, 14, 13, 12, 11, 10, 9, 5, 45, 48, 47,
21);

Arduino_ST7701_RGBPanel *gfx = new Arduino_ST7701_RGBPanel(bus,
GFX_NOT_DEFINED, 0, false, 480, 480, st7701_type5_init_operations,
sizeof(st7701_type5_init_operations), true, 10, 4, 20, 10, 4, 20);

Adafruit_CST8XX tsPanel = Adafruit_CST8XX();
```

```

PCF8574 pcf8574(0x20);

WiFiClient espClient;
PubSubClient client(espClient);
Preferences prefs;


static lv_disp_draw_buf_t draw_buf;
static lv_color_t *buf1;
const uint16_t screenWidth = 480;
const uint16_t screenHeight = 480;
const char* mqtt_server = "192.168.0.100";
const char* mqtt_user = "esp32";
const char* mqtt_pass = "1234";


int lastClkState, brightnessValue = 70, configSoilThreshold, rawSoil =
SOIL_DRY, rawWater = 0, rawBat = 0, pStateC6 = 0;
float remoteTemp = 0.0;
bool isConnecting = false, manualOverride = false, watering = false;
unsigned long connectionTimer = 0, lastWaterTime = 0,
wateringStartTime = 0, currentWaterDuration = 0, lastMsgTime = 0,
lastReconnectAttempt = 0;
unsigned long configWaterInterval, configMaxDur, configMinDur;


void my_disp_flush(lv_disp_drv_t *disp, const lv_area_t *area,
lv_color_t *color_p)
{
    uint32_t w = (area->x2 - area->x1 + 1);
    uint32_t h = (area->y2 - area->y1 + 1);
    gfx->draw16bitRGBBitmap(area->x1, area->y1, (uint16_t *)&color_p-
>full, w, h);
    lv_disp_flush_ready(disp);
}


void my_touchpad_read(lv_indev_drv_t *indev_driver, lv_indev_data_t
*data)

```

```

{
    if (tsPanel.touched())
    {
        CST_TS_Point p = tsPanel.getPoint(0);
        data->point.x = p.x;
        data->point.y = p.y - 20;
        data->state = LV_INDEV_STATE_PR;
    }
    else
    {
        data->state = LV_INDEV_STATE_REL;
    }
}

void initEncoder()
{
    pinMode(ENCODER_CLK, INPUT);
    pinMode(ENCODER_DT, INPUT);
    lastClkState = digitalRead(ENCODER_CLK);
}

void handleEncoder()
{
    int clkState = digitalRead(ENCODER_CLK);
    if (clkState != lastClkState && clkState == LOW)
    {
        if (digitalRead(ENCODER_DT) == clkState)
        {
            brightnessValue += 5;
        }
        else
        {

```

```

        brightnessValue -= 5;
    }

    brightnessValue = constrain(brightnessValue, 0, 100);
    ledcWrite(0, (brightnessValue * 255) / 100);

    if (ui_ArcLight) lv_arc_set_value(ui_ArcLight,
brightnessValue);
    if (ui_LabelLight)
    {
        char buf[8];
        snprintf(buf, sizeof(buf), "%d%%", brightnessValue);
        lv_label_set_text(ui_LabelLight, buf);
    }
}

lastClkState = clkState;
}

void parseC6Data(String data)
{
    int sIdx = data.indexOf("S:");
    int wIdx = data.indexOf("|W:");
    int bIdx = data.indexOf("|B:");
    int tIdx = data.indexOf("|T:");
    int pIdx = data.indexOf("|P:");

    if (sIdx != -1 && wIdx != -1 && bIdx != -1 && tIdx != -1 && pIdx
!= -1)
    {
        rawSoil = data.substring(sIdx + 2, wIdx).toInt();
        rawWater = data.substring(wIdx + 3, bIdx).toInt();
        rawBat = data.substring(bIdx + 3, tIdx).toInt();
        remoteTemp = data.substring(tIdx + 3, pIdx).toFloat();
        pStateC6 = data.substring(pIdx + 3).toInt();
    }
}

```

```

    }
}

void stopWatering()
{
    Serial1.println("PUMP_OFF");
    watering = false;
    manualOverride = false;
    lastWaterTime = millis();
    client.publish("garden/pump", "OFF", true);
}

void syncMqtt()
{
    client.publish("garden/config/soil_threshold",
String(configSoilThreshold).c_str(), true);

    client.publish("garden/config/interval",
String(configWaterInterval / 3600000UL).c_str(), true);

    client.publish("garden/config/min_dur", String(configMinDur /
1000.0, 1).c_str(), true);

    client.publish("garden/config/max_dur", String(configMaxDur /
1000.0, 1).c_str(), true);
}

void callback(char* topic, byte* payload, unsigned int length)
{
    String msg = "";
    for (unsigned int i = 0; i < length; i++) msg += (char)payload[i];
    String t = String(topic);
    float val = msg.toFloat();

    prefs.begin("garden", false);
    if (t == "garden/config/soil_threshold/set")

```

```

{
    configSoilThreshold = (int)val;
    prefs.putInt("threshold", configSoilThreshold);
}
else if (t == "garden/config/interval/set")
{
    configWaterInterval = (unsigned long)(val * 3600000UL);
    prefs.putULong("interval", configWaterInterval);
}
else if (t == "garden/config/min_dur/set")
{
    configMinDur = (unsigned long)(val * 1000.0);
    prefs.putULong("min_dur", configMinDur);
}
else if (t == "garden/config/max_dur/set")
{
    configMaxDur = (unsigned long)(val * 1000.0);
    prefs.putULong("max_dur", configMaxDur);
}
else if (t == "garden/pump/set")
{
    if (msg == "ON") manualOverride = true;
    else stopWatering();
}
prefs.end();
syncMqtt();
}

void updateFlowerInfoUI()
{
    if (lv_scr_act() != ui_Flowerinfo) return;

```

```

    float soilP = constrain((float)(SOIL_DRY - rawSoil) / (SOIL_DRY -
SOIL_WET) * 100, 0, 100);

    float waterP = constrain((float)rawWater / WATER_FULL * 100, 0,
100);

    float Vbat = (rawBat / 1.014 * 2.0) / 1000.0;
    float batP = constrain((Vbat - 3.2) / (4.2 - 3.2) * 100, 0, 100);


    lv_arc_set_value(ui_ArcMoisture, (int)soilP);
    lv_arc_set_value(ui_ArcWaterlevel, (int)waterP);
    lv_arc_set_value(ui_ArcTemperature, (int)remoteTemp);
    lv_arc_set_value(ui_ArcBattery, (int)batP);


    char buf[16];
    snprintf(buf, sizeof(buf), "%d%", (int)soilP);
    lv_label_set_text(ui_LabelValuemMOISTURE, buf);
    snprintf(buf, sizeof(buf), "%d%", (int)waterP);
    lv_label_set_text(ui_LabelValueWaterlevel, buf);
    snprintf(buf, sizeof(buf), "%.1fC", remoteTemp);
    lv_label_set_text(ui_LabelValueTemp, buf);
    snprintf(buf, sizeof(buf), "%d%", (int)batP);
    lv_label_set_text(ui_LabelValueBAT, buf);
}

void syncPumpUI()
{
    if (ui_CheckboxONoffpump == NULL) return;
    bool isActuallyOn = (pStateC6 == 1);
    bool isUIChecked = lv_obj_has_state(ui_CheckboxONoffpump,
LV_STATE_CHECKED);
    if (isActuallyOn != isUIChecked)
    {
        if (isActuallyOn) lv_obj_add_state(ui_CheckboxONoffpump,
LV_STATE_CHECKED);
    }
}

```

```

        else lv_obj_clear_state(ui_CheckboxONoffpump,
LV_STATE_CHECKED);
    }
}

void scanWifiNetworks()
{
    WiFi.scanDelete();
    WiFi.scanNetworks(true);
    lv_label_set_text(ui_LabelVALEUStatusWIFI, "Scanning...");
}

void btnRefresh_cb(lv_event_t * e)
{
    scanWifiNetworks();
}

void btnConnectWifi_cb(lv_event_t * e)
{
    char selected_ssid[64];
    lv_dropdown_get_selected_str(ui_SelectWifi, selected_ssid,
sizeof(selected_ssid));
    const char * pwd = lv_textarea_get_text(ui_InputPassword);

    prefs.begin("wifi_data", false);
    prefs.putString("ssid", selected_ssid);
    prefs.putString("pass", pwd);
    prefs.end();

    lv_label_set_text(ui_LabelVALEUStatusWIFI, "Connecting...");
    WiFi.disconnect();
    WiFi.begin(selected_ssid, pwd);
}

```

```

    isConnected = true;
    connectionTimer = millis();
}

void ui_event_Settings_Handler(lv_event_t * e)
{
    lv_obj_t * target = lv_event_get_target(e);
    char buf[16];
    prefs.begin("garden", false);

    if (target == ui_SliderMoisturelimit)
    {
        configSoilThreshold = lv_slider_get_value(target);
        snprintf(buf, sizeof(buf), "%d%%", configSoilThreshold);
        lv_label_set_text(ui_LabelVALUEMoisturelimit, buf);
        prefs.putInt("threshold", configSoilThreshold);
    }
    if (target == ui_SliderWateringCheck)
    {
        int h = lv_slider_get_value(target);
        configWaterInterval = (unsigned long)h * 3600000UL;
        snprintf(buf, sizeof(buf), "%d h", h);
        lv_label_set_text(ui_LabelVALUEWateringCheck, buf);
        prefs.putULong("interval", configWaterInterval);
    }
    if (target == ui_SliderMintimeWatering)
    {
        int s = lv_slider_get_value(target);
        configMinDur = (unsigned long)s * 1000UL;
        snprintf(buf, sizeof(buf), "%d s", s);
        lv_label_set_text(ui_LabelVALUEMintimeWatering, buf);
        prefs.putULong("min_dur", configMinDur);
    }
}

```

```

    }

    if (target == ui_SliderMaxtimeWatering)
    {
        int s = lv_slider_get_value(target);
        configMaxDur = (unsigned long)s * 1000UL;
        snprintf(buf, sizeof(buf), "%d s", s);
        lv_label_set_text(ui_LabelVALUEMaxtimeWatering, buf);
        prefs.putULong("max_dur", configMaxDur);
    }

    prefs.end();
    syncMqtt();
}

void ui_event_PumpCheckbox_Handler(lv_event_t * e)
{
    if (lv_obj_has_state(lv_event_get_target(e), LV_STATE_CHECKED))
    {
        manualOverride = true;
        watering = true;
        wateringStartTime = millis();
        currentWaterDuration = configMaxDur;
        Serial1.println("PUMP_ON");
        client.publish("garden/pump", "ON", true);
    }
    else
    {
        stopWatering();
    }
}

void lightArcEventCb(lv_event_t * e)
{

```

```

int val = lv_arc_get_value(lv_event_get_target(e));
if (ui_LabelLight)
{
    char buf[8];
    snprintf(buf, sizeof(buf), "%d%%", val);
    lv_label_set_text(ui_LabelLight, buf);
}
ledcWrite(0, (val * 255) / 100);
}

```

```

void setup()
{
    Serial.begin(115200);
    Serial1.begin(115200, SERIAL_8N1, S3_RX, S3_TX);
    Wire.begin(I2C_SDA_PIN, I2C_SCL_PIN);

    pcf8574.begin();
    pcf8574.pinMode(P3, OUTPUT);
    pcf8574.pinMode(P4, OUTPUT);
    pcf8574.pinMode(P0, OUTPUT);
    pcf8574.digitalWrite(P3, HIGH);
    pcf8574.digitalWrite(P4, LOW); delay(120);
    pcf8574.digitalWrite(P4, HIGH); delay(120);
    pcf8574.digitalWrite(P0, LOW); delay(120);
    pcf8574.digitalWrite(P0, HIGH); delay(120);

    gfx->begin();
    gfx->fillScreen(BLACK);
    tsPanel.begin(&Wire, I2C_TOUCH_ADDR);

    ledcSetup(0, 5000, 8);
    ledcAttachPin(6, 0);
}

```

```

    ledcWrite(0, 200);

    lv_init();

    buf1 = (lv_color_t *)heap_caps_malloc(screenWidth * screenHeight *
sizeof(lv_color_t), MALLOC_CAP_SPIRAM);

    lv_disp_draw_buf_init(&draw_buf, buf1, NULL, screenWidth *
screenHeight);

    static lv_disp_drv_t disp_drv;
    lv_disp_drv_init(&disp_drv);
    disp_drv.hor_res = screenWidth;
    disp_drv.ver_res = screenHeight;
    disp_drv.flush_cb = my_disp_flush;
    disp_drv.draw_buf = &draw_buf;
    disp_drv.sw_rotate = 1;
    disp_drv.rotated = LV_DISP_ROT_270;
    lv_disp_drv_register(&disp_drv);

    static lv_indev_drv_t indev_drv;
    lv_indev_drv_init(&indev_drv);
    indev_drv.type = LV_INDEV_TYPE_POINTER;
    indev_drv.read_cb = my_touchpad_read;
    lv_indev_drv_register(&indev_drv);

    ui_init();
    initEncoder();

    prefs.begin("wifi_data", true);
    String sSSID = prefs.getString("ssid", "");
    String sPASS = prefs.getString("pass", "");
    prefs.end();

```

```

if (sSSID != "" && sSSID != "No networks found")
{
    lv_textarea_set_text(ui_InputPassword, sPASS.c_str());
    WiFi.begin(sSSID.c_str(), sPASS.c_str());
    isConnected = true;
    connectionTimer = millis();
}

scanWifiNetworks();

prefs.begin("garden", true);
configSoilThreshold = prefs.getInt("threshold", 60);
configWaterInterval = prefs.getULong("interval", 21600000UL);
configMaxDur = prefs.getULong("max_dur", 7000UL);
configMinDur = prefs.getULong("min_dur", 3000UL);
prefs.end();

lv_slider_set_value(ui_SliderMoisturelimit, configSoilThreshold,
LV_ANIM_OFF);

lv_slider_set_value(ui_SliderWateringCheck, configWaterInterval /
3600000UL, LV_ANIM_OFF);

lv_slider_set_value(ui_SliderMintimeWatering, configMinDur /
1000UL, LV_ANIM_OFF);

lv_slider_set_value(ui_SliderMaxtimeWatering, configMaxDur /
1000UL, LV_ANIM_OFF);

client.setServer(mqtt_server, 1883);
client.setCallback(callback);

lv_obj_add_event_cb(ui_ButtonRefresh, btnRefresh_cb,
LV_EVENT_CLICKED, NULL);

lv_obj_add_event_cb(ui_ButtonConnect, btnConnectWifi_cb,
LV_EVENT_CLICKED, NULL);

```

```

    lv_obj_add_event_cb(ui_ArcLight, lightArcEventCb,
LV_EVENT_VALUE_CHANGED, NULL);

    lv_obj_add_event_cb(ui_SliderMoisturelimit,
ui_event_Settings_Handler, LV_EVENT_VALUE_CHANGED, NULL);

    lv_obj_add_event_cb(ui_SliderWateringCheck,
ui_event_Settings_Handler, LV_EVENT_VALUE_CHANGED, NULL);

    lv_obj_add_event_cb(ui_SliderMintimeWatering,
ui_event_Settings_Handler, LV_EVENT_VALUE_CHANGED, NULL);

    lv_obj_add_event_cb(ui_SliderMaxtimeWatering,
ui_event_Settings_Handler, LV_EVENT_VALUE_CHANGED, NULL);

    lv_obj_add_event_cb(ui_CheckboxONoffpump,
ui_event_PumpCheckbox_Handler, LV_EVENT_VALUE_CHANGED, NULL);

    lv_keyboard_set_textarea(ui_Keyboard, ui_InputPassword);
}

```

```

void loop()
{
    handleEncoder();

    lv_timer_handler();

    if (!client.connected())
    {
        if (millis() - lastReconnectAttempt > 5000)
        {
            lastReconnectAttempt = millis();

            if (client.connect("ESP32_S3_Master", mqtt_user,
mqtt_pass))
            {
                client.subscribe("garden/config/+/set");
                client.subscribe("garden/pump/set");
                syncMqtt();
            }
        }
    }
}

```

```

else
{
    client.loop();
}

if (Serial1.available())
parseC6Data(Serial1.readStringUntil('\n'));

unsigned long now = millis();

float soilP = constrain((float)(SOIL_DRY - rawSoil) / (SOIL_DRY -
SOIL_WET) * 100, 0, 100);

float waterP = constrain((float)rawWater / WATER_FULL * 100, 0,
100);

if (!watering)
{
    if (manualOverride || ((now - lastWaterTime >=
configWaterInterval) && (soilP < configSoilThreshold)))
    {
        float factor = 1.0 - (soilP / 100.0);
        currentWaterDuration = constrain(configMinDur + (factor *
(configMaxDur - configMinDur)), configMinDur, configMaxDur);
        wateringStartTime = now;
        watering = true;
        manualOverride = false;
        Serial1.println("PUMP_ON");
        client.publish("garden/pump", "ON", true);
    }
}
else
{
    if (now - wateringStartTime >= currentWaterDuration)
    {

```

```

        stopWatering();
    }
}

int scanResult = WiFi.scanComplete();
if (scanResult >= 0)
{
    String options = "";
    for (int i = 0; i < scanResult; ++i)
    {
        options += WiFi.SSID(i) + (i < scanResult - 1 ? "\n" :
""");
    }
    lv_dropdown_set_options(ui_SelectWifi, options.c_str());
    WiFi.scanDelete();
    lv_label_set_text(ui_LabelVALEUStatusWIFI, "Scan complete.");
}

static unsigned long lUI = 0, lW = 0, lM = 0;
if (now - lUI > 1000)
{
    lUI = now;
    updateFlowerInfoUI();
    syncPumpUI();
}

if (now - lW > 2000)
{
    lW = now;

    String wifiMsg = (WiFi.status() == WL_CONNECTED) ?
("Connected: " + WiFi.SSID()) : (isConnecting ? "Connecting..." :
"Disconnected");

    lv_label_set_text(ui_LabelVALEUStatusWIFI, wifiMsg.c_str());
}

```

```

}

if (now - lM > 15000)
{
    lM = now;
    client.publish("garden/soil", String(soilP, 1).c_str());
    client.publish("garden/temp", String(remoteTemp, 1).c_str());
    client.publish("garden/water_level", String(waterP,
0).c_str());
    float Vb = (rawBat / 1.014 * 2.0) / 1000.0;
    client.publish("garden/battery", String(constrain((Vb - 3.2) /
(4.2 - 3.2) * 100, 0, 100), 0).c_str());
}
}

```

ДОДАТОК Б

Код ESP32-C6 Zero (Slave)

```
#include <Arduino.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>

#define ADC_PIN 1
#define WATER_SENSOR_PIN 2
#define SOIL_PIN 0
#define MOSFET_PIN 5
#define C6_RX 22
#define C6_TX 21

Adafruit_BME280 bme;
unsigned long lastSend = 0;

void setup() {
    Serial.begin(115200);
    Serial1.begin(115200, SERIAL_8N1, C6_RX, C6_TX);
    pinMode(MOSFET_PIN, OUTPUT);
    digitalWrite(MOSFET_PIN, LOW);
    Wire.begin(6, 7);
    if (!bme.begin(0x76, &Wire)) Serial.println("BME Error");
}

void loop() {
    if (Serial1.available()) {
        String cmd = Serial1.readStringUntil('\n');
        cmd.trim();
```

```

    if (cmd == "PUMP_ON") digitalWrite(MOSFET_PIN, HIGH);
    else if (cmd == "PUMP_OFF") digitalWrite(MOSFET_PIN, LOW);
}

if (millis() - lastSend > 500) {
    lastSend = millis();
    int s = analogRead(SOIL_PIN);
    int w = analogRead(WATER_SENSOR_PIN);
    int b = analogRead(ADC_PIN);
    float temp = bme.readTemperature();
    int pState = digitalRead(MOSFET_PIN);

    Serial1.printf("S:%d|W:%d|B:%d|T:%.1f|P:%d\n", s, w, b, temp,
pState);

    Serial.printf("S:%d|W:%d|B:%d|T:%.1f|P:%d\n", s, w, b, temp,
pState);
}
}

```

ДОДАТОК В

Код UI

```
type: grid
cards:
  - type: heading
    heading_style: title
  - type: vertical-stack
    cards:
      - type: custom:mushroom-template-card
        primary: СИСТЕМА ПОЛИВУ
        secondary: >
          Вологість ґрунту: {{ states('sensor.vologist_gruntu') }}% |
{{ 'Все в
        нормі' if states('sensor.vologist_gruntu')|int > 50 else
'Потребує
        уваги' }}
        icon: mdi:leaf
        icon_color: green
        multiline_secondary: true
        fill_container: true
        layout: horizontal
        style: |
          ha-card {
            --mushroom-icon-size: 60px;
            background: rgba(76, 175, 80, 0.1);
          }
      - type: grid
        columns: 2
        square: true
        cards:
```

- type: tile
 - entity: sensor.vologist_gruntu
 - name: Вологість
 - icon: mdi:water-percent
 - color: green
- type: tile
 - entity: sensor.riven_vodi_v_baku
 - name: Рівень води
 - icon: mdi:cup-water
 - color: blue
- type: tile
 - entity: sensor.temperatura_teplitsi
 - name: Температура
 - icon: mdi:thermometer
 - color: orange
- type: tile
 - entity: sensor.zariad_batarei_esp32
 - name: Батарея
 - icon: mdi:battery-high
 - color: light-green
- type: heading
 - heading: ⚙️ □ Параметри системи
 - icon: mdi:cog-sync
- type: grid
 - columns: 2
 - square: false
 - cards:
 - type: tile
 - entity: number.porig_vologosti_dlja_polivu
 - name: Поріг вологості
 - icon: mdi:water-percent
 - color: green

```

    tap_action:
      action: more-info
- type: tile
  entity: number.interval_perevirki
  name: Интервал (год)
  icon: mdi:history
  color: green
  tap_action:
    action: more-info
- type: tile
  entity: number.min_trivalist_polivu
  name: Мин. час (сек)
  icon: mdi:timer-outline
  color: light-green
  tap_action:
    action: more-info
- type: tile
  entity: number.maks_trivalist_polivu
  name: Макс. час (сек)
  icon: mdi:timer-lock-outline
  color: light-green
  tap_action:
    action: more-info
- type: tile
  entity: switch.primusovii_poliv
  name: ЗАПУСТИТИ ПОЛИВ
  icon: mdi:sprinkler-variant
  color: green
  tap_action:
    action: toggle
style: |
  ha-card {

```

```
        background: rgba(76, 175, 80, 0.2);
        --tile-icon-size: 50px;
        border: 1px solid green;
    }
column_span: 1
background:
    color: light-green
    opacity: 50
```

ДОДАТОК Г

YAML-конфігурація Home Assistant для системи поливу

default_config:

frontend:

themes: !include_dir_merge_named themes

automation: !include automations.yaml

script: !include scripts.yaml

scene: !include scenes.yaml

mqtt:

sensor:

- name: "Вологість ґрунту"
unique_id: "garden_soil_moisture"
state_topic: "garden/soil"
unit_of_measurement: "%"
state_class: "measurement"

- name: "Рівень води в баку"
unique_id: "garden_water_level"
state_topic: "garden/water_level"
unit_of_measurement: "%"

- name: "Заряд батареї"
unique_id: "garden_battery_level"
state_topic: "garden/battery"
device_class: "battery"
unit_of_measurement: "%"

- name: "Температура"

```

    unique_id: "garden_temperature"
    state_topic: "garden/temp"
    device_class: "temperature"
    unit_of_measurement: "°C"
switch:
- name: "Примусовий полив"
  unique_id: "garden_manual_pump"
  state_topic: "garden/pump"
  command_topic: "garden/pump/set"
  payload_on: "ON"
  payload_off: "OFF"
  optimistic: false

number:
- name: "Поріг вологості"
  unique_id: "garden_soil_threshold"
  state_topic: "garden/config/soil_threshold"
  command_topic: "garden/config/soil_threshold/set"
  min: 10
  max: 90
  step: 1
  unit_of_measurement: "%"

- name: "Інтервал перевірки"
  unique_id: "garden_interval_config"
  state_topic: "garden/config/interval"
  command_topic: "garden/config/interval/set"
  min: 1
  max: 48
  unit_of_measurement: "год"

- name: "Мін тривалість"

```

unique_id: "garden_min_duration"
state_topic: "garden/config/min_dur"
command_topic: "garden/config/min_dur/set"
min: 0.5
max: 10
step: 0.5
unit_of_measurement: "сек"

- name: "Макс тривалість"
unique_id: "garden_max_duration"
state_topic: "garden/config/max_dur"
command_topic: "garden/config/max_dur/set"
min: 5
max: 60
step: 1
unit_of_measurement: "сек"