МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВОЛИНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ЛЕСІ УКРАЇНКИ
Кафедра комп'ютерних наук та кібербезпеки

На правах рукопису

МАРКІТАН ВОЛОДМИР ОЛЕГОВИЧ

РОЗРОБКА ТА ДОСЛІДЖЕННЯ МЕРЕЖЕВОГО МЕНЕДЖЕРА ДЛЯ ДИСТАНЦІЙНОГО АВТОМАТИЗОВАНОГО 3D-ДРУКУ

Спеціальність: 122 Комп'ютерні науки
Освітньо-професійна програма: Комп'ютерні науки та інформаційні технології
Кваліфікаційна робота на здобуття освітнього ступеня «магістр»

Науковий керівник:
ГОЛОВІН МИКОЛА БОРИСОВИЧ,
кандитат фізико-математичних наук,
доцент кафедри комп'ютерних наук та кібербезпеки

РЕКОМЕНДОВАНО ДО
ЗАХИСТУ
Протокол № _____
засідання кафедри _____
від_____2025 р.
Завідувач кафедри

(_____) _____
(підпис) ПІБ

ЛУЦЬК – 2025

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ ТА ТЕРМІНІВ 4
ВСТУП 5
РОЗДІЛ 1: ТЕОРЕТИЧНІ ТА МЕТОДОЛОГІЧНІ ЗАСАДИ АВТОМАТИЗАЦІЇ 3D-ДРУКУ 7
1.1 Аналіз сучасного стану ринку адитивних технологій та FDM-друку 7
1.1.1 Технологічні основи методу пошарового наплавлення 7
1.1.2 Загальні тенденції ринку та перехід до цифрового виробництва 9
1.1.3 «Феномен Bambu Lab» та особливості закритих програмно-апаратних середовищ 11
1.1.4 Проблема автоматизації пост-обробки 13
1.2 Принципи побудови хмарних систем керування IoT-пристроями 14
1.3 Дослідження протоколів взаємодії та структури даних платформи Bambu Lab 17
1.4 Огляд та порівняльний аналіз існуючих систем автоматизації 3D-друку. 19
1.5 Технологічні принципи реалізації автоматичного очищення 23
1.5.1 Термодинаміка адгезії 23
1.5.2 Принцип механічного змітання 24
1.5.3 Роль гравітаційного впливу 25
РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА МЕРЕЖЕВОГО МЕНЕДЖЕРА «CORELOOP» 26
2.1 Постановка задачі та формування функціональних вимог до системи 26
2.2 Проєктування та реалізація апаратних модифікацій принтера 27
2.3 Методологія розробки та життєвий цикл проєкту 30
2.4 Загальна архітектура та логіка функціонування системи 32
2.5 Обґрунтування вибору інструментальних засобів та технологій реалізації 36
2.5.1 Засоби реалізації серверної частини 36
2.5.2 Засоби реалізації клієнтської частини 37
2.5.3 Протоколи обміну даними 38
2.5.4 Обґрунтування вибору інструментальних засобів розробки 38
2.6 Особливості програмної реалізації 40

2.6.1 Реалізація підсистеми моніторингу та низькорівневої комунікації..... 40
2.6.2 Реверс-інжиніринг протоколу та реалізація MQTT-сніфера 43
2.6.3 Програмна реалізація алгоритму автоматизації CoreLoop 44
2.6.4 Програмна реалізація механічного очищення робочої зони 46
2.6.5 Управління чергою завдань та реалізація прямого запуску друку 49
2.6.6 Реалізація клієнтського інтерфейсу та підсистеми лайвстримінгу..... 53
2.7 Організація тестування та налагодження програмного комплексу..... 58
2.8 Аналіз отриманих результатів та рекомендації щодо впровадження 59
ВИСНОВКИ..... 61
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 63
ДОДАТКИ..... 66
Додаток А 66
Додаток Б..... 68

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ ТА ТЕРМІНІВ

IoT	– Інтернет речей
FDM	– Fused Deposition Modeling
MQTT	– Message Queue Telemetry Transport
HTTP	– HyperText Transfer Protocol
FTP	– File Transfer Protocol
МП	– Мова Програмування
ПЗ	– Програмне Забезпечення
ТЗ	– Технічне Завдання
JSON	– JavaScript Object Notation
PY	– Python: Інтерпретована МП

ВСТУП

Актуальність теми. Стрімкий розвиток цифрового виробництва та впровадження концепції «Промисловість 4.0» докорінно змінюють підходи до створення фізичних об’єктів. Технології 3D-друку вийшли за межі лабораторій прототипування та інтегруються у виробничі процеси, забезпечуючи можливість децентралізованого виготовлення деталей «на вимогу». У сучасних умовах 3D-принтер перетворюється з ізольованого периферійного пристрою на повноцінний вузол IoT, ефективність якого прямо залежить від рівня його автономності.

Сучасні FDM-системи, зокрема обладнання Bambu Lab, досягли високого рівня продуктивності, достатнього для дрібносерійного виробництва. Однак їх ефективність суттєво обмежується необхідністю фізичної присутності оператора для обслуговування між циклами (знімання деталі, запуск нового файлу). Існуючі програмні рішення орієнтовані переважно на хмарний моніторинг і не вирішують проблему повної автоматизації у локальному режимі без доступу до інтернету. У зв’язку з цим, розробка системи, здатної автоматизувати повний цикл «черга–друк–очищення» без втручання людини, є критично важливим завданням для мінімізації простоїв обладнання.

Мета даної роботи полягає у розробці програмного забезпечення для дистанційного керування чергою друку 3D-принтера та автоматичного очищення робочої зони для підвищення рівня автономності обладнання.

Завдання. Для реалізації поставленої мети нам потрібно виконати наступне:

- Провести аналіз предметної області щодо дистанційного керування роботою 3D-принтера, зокрема дослідити протоколи взаємодії (MQTT, FTP) у закритих програмно-апаратних середовищах;

- Обґрунтувати вибір програмного середовища та інструментальних засобів для розробки веб-застосунку;

- Реалізувати серверну і клієнтську частину системи з урахуванням вимог безпеки, цілісності даних та ефективності (включаючи реверс-інжиніринг пакетів керування);

- Створити інтерфейс користувача, який забезпечує зручну взаємодію, керування чергою завдань та візуалізацію телеметричних даних у реальному часі;

- Провести тестування розробленої системи для перевірки її працездатності, надійності, зокрема алгоритмів автоматичного очищення та відповідності поставленим вимогам.

Об’єкт дослідження – процеси дистанційного керування та моніторингу адитивного обладнання (FDM 3D-принтерів) у локальних мережах.

Предмет дослідження – методи та програмні засоби, а саме: веб-технології, протоколи IoT, алгоритми обробки G-коду, що забезпечують автоматизацію циклу друку та взаємодію з пропрієтарним обладнання.

Наукова новизна – у роботі використано методи системного аналізу для проєктування архітектури ПЗ; методи реверс інженірингу для аналізу закритих мережевих протоколів; об’єктно-орієнтоване програмування для реалізації серверної частини; емпіричні методи для тестування алгоритмів механічного очищення

Наукова новизна одержаних результатів полягає у вдосконаленні методу керування 3D-принтерами із закритою архітектурою шляхом прямої маніпуляції пакетами протоколу MQTT, що дозволило реалізувати функціонал автоматичного запуску локальних файлів, недоступний у штатних засобах виробника.

Практичне значення роботи полягає у створенні діючого програмно-апаратного комплексу «CoreLoop», який дозволяє перетворити настільний 3D-принтер на автоматизовану виробничу чарунку, здатну працювати в режимі 24/7, що підтверджено результатами апробації.

РОЗДІЛ 1: ТЕОРЕТИЧНІ ТА МЕТОДОЛОГІЧНІ ЗАСАДИ АВТОМАТИЗАЦІЇ 3D-ДРУКУ

1.1 Аналіз сучасного стану ринку адитивних технологій та FDM-друку

1.1.1 Технологічні основи методу пошарового наплавлення

Технологія пошарового наплавлення (FDM) становить технологічну основу сучасного сегменту доступного цифрового виробництва. Фізична сутність процесу базується на контрольованій зміні агрегатного стану термопластичного матеріалу під впливом термічної енергії з подальшим його формоутворенням. Процес розпочинається з механічної подачі полімерної нитки-філаменту до нагрівального блоку екструдера, де відбувається фазовий перехід матеріалу з твердого стану у в’язко-текучий. Критично важливим параметром на цьому етапі є реологія розплаву, яка визначає здатність матеріалу проходити через каліброване сопло та зберігати геометричну форму після нанесення на робочу платформу.

Базовий цикл FDM-друку включає такі етапи:

- Слайсинг – 3D-модель розбивається на горизонтальні шари ПЗ яке генерує G-код — набір команд для керування координатами та екструзією;
- Нагрів та екструзія – термопластик подається механізмом подачі (екструдером) у нагрівальний блок (hotend), де переходить у в’язко-текучий стан (рис. 1.1);
- Шарування – розплавлений матеріал вичавлюється через сопло (nozzle) на робочу поверхню, де за допомогою вентилятора миттєво охолоджується та твердне, утворюючи фізичний шар об’єкта;
- Зміна висоти – після завершення шару друкувальна голівка піднімається на величину товщини шару, і процес повторюється. [1]

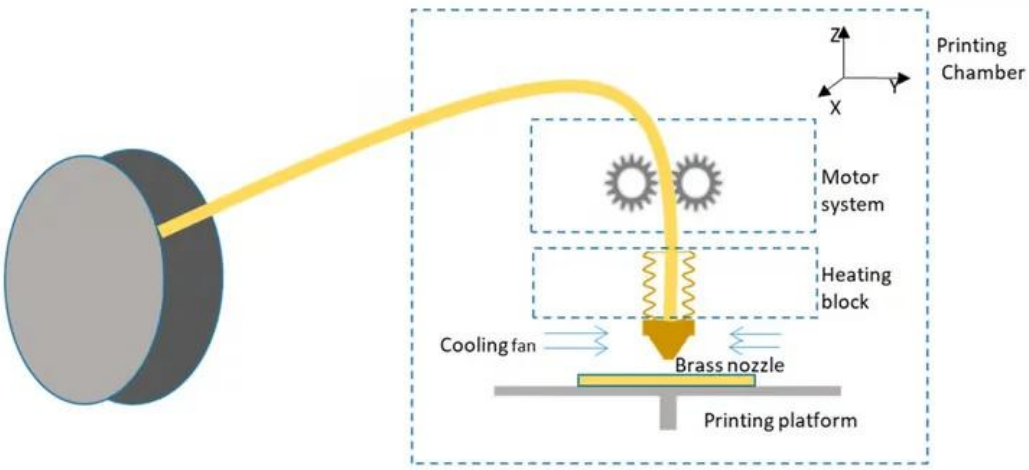

Рисунок 1.1 – Принцип роботи FDM-екструдера. [2]

Формування тривимірного об’єкта здійснюється шляхом послідовної укладки шарів матеріалу в площині XY з кроковим зміщенням по осі Z. Міцність кінцевого виробу безпосередньо залежить від якості міжшарової адгезії, яка, у свою чергу, визначається температурним градієнтом між щойно нанесеним шаром та попереднім, вже частково охолодженим шаром. Саме термодинаміка процесу охолодження є ключовим фактором, що впливає як на геометричну точність (відсутність деформацій усадки), так і на можливість автоматизації процесу знімання деталі. Для забезпечення надійної фіксації виробу під час друку використовуються спеціалізовані покриття робочої платформи, які мають властивість змінювати адгезійну здатність залежно від температури: при високих температурах адгезія максимальна, а при охолодженні до кімнатної температури відбувається самовільне відшаровування деталі або значне зменшення сили зчеплення.

З точки зору механіки та кінематики, досліджуване обладнання (Bambu Lab A1 Mini) належить до класу декартових принтерів (Cartesian) з консольною архітектурою. На відміну від рамних конструкцій типу CoreXY, де робочий стіл рухається виключно вертикально, у даній схемі реалізовано кінематику рухомого столу що переміщується вздовж осі Y. Друкувальна голівка рухається вздовж осі X по консольній балці, яка, у свою чергу, піднімається вздовж осі Z (рис. 1.2). Така архітектура накладає специфічні обмеження на динаміку

процесу: через інерційність рухомого столу виникають додаткові вимоги до алгоритмів керування прискореннями для уникнення вібрацій (ghosting). Проте, у контексті задачі автоматичного очищення робочої зони, така конструкція відкриває специфічні можливості: використання інерції самої деталі при різкому русі столу або комбінований вплив руху столу та друкувальної голівки для ефективного видалення готового виробу за межі робочої зони.

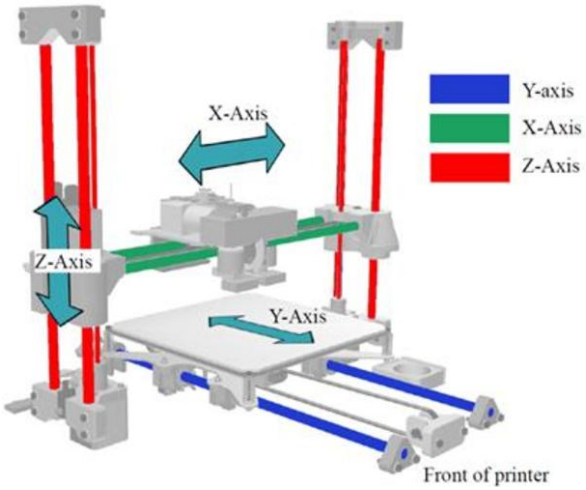

Рисунок 1.2 – Кінематична схема FDM-принтера з рухомим робочим столом (Cartesian XZ-Head / Y-Bed). [3]

1.1.2 Загальні тенденції ринку та перехід до цифрового виробництва

Сучасний етап розвитку світової економіки характеризується глобальною трансформацією виробничих ланцюгів, де адитивні технології відіграють роль каталізатора переходу до децентралізованого виробництва. Згідно з аналітичними даними провідних консалтингових агенцій, ринок адитивного виробництва демонструє стабільну тенденцію до зростання, трансформуючись із нішевого інструменту для швидкого прототипування (Rapid Prototyping) у повноцінну технологію прямого цифрового виробництва (Direct Digital Manufacturing). Цей зсув зумовлений необхідністю скорочення логістичних витрат та часу виведення продукту на ринок, що стало особливо актуальним в умовах порушення глобальних ланцюгів постачання останніми роками.

Динаміка ринку підтверджує стрімке зростання інтересу до адитивних технологій. За даними аналітичного звіту Grand View Research, обсяг світового ринку 3D-друку демонструє стабільний ріст і, згідно з прогнозами, до 2030 року може досягти показника у 88,3 млрд доларів США (рис. 1.3). [4]

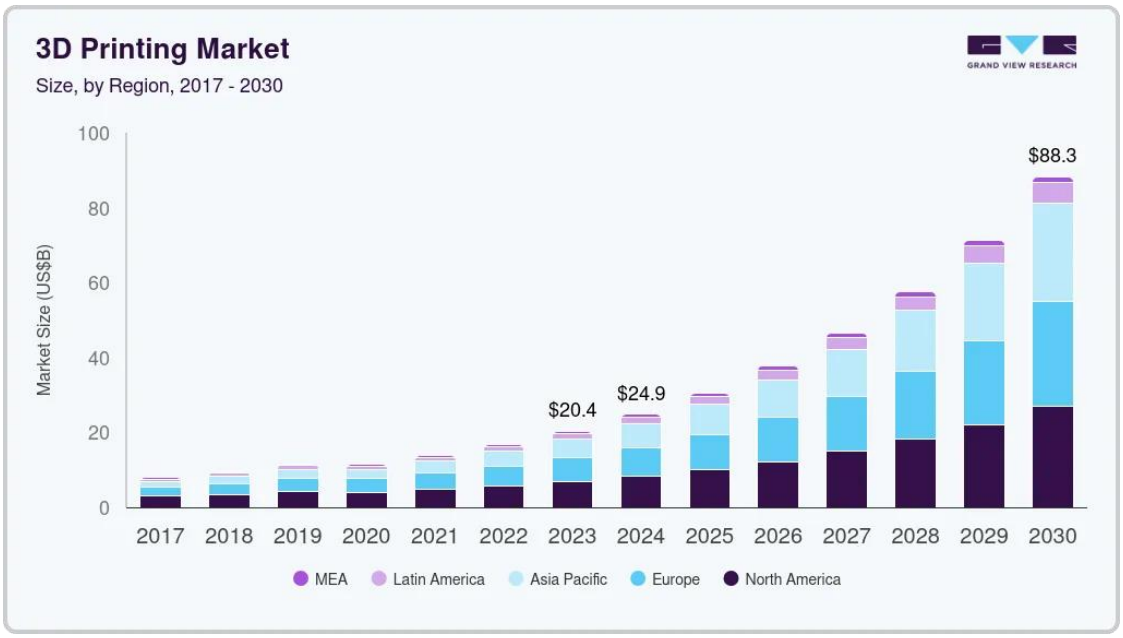

Рисунок 1.3 – Динаміка та прогноз зростання світового ринку адитивного виробництва у 2017–2030 рр. (за даними Grand View Research). [4]

Як видно з діаграми, ринок Європи, Азії та Північної Америки та займає значну частку, що свідчить про високий рівень технологічної готовності цих регіонів. Особливу увагу привертає сегментація ринку настільних FDM-систем. Якщо раніше цей сегмент чітко поділявся на аматорські пристрої низької надійності та дорогі промислові установки, то сьогодні сформувався потужний обладнання класу «prosumer». Цей клас пристроїв, до якого належить і досліджувана модель, забезпечує точність та повторюваність результатів, співставну з промисловими зразками, при значно нижчій капітальній вартості. Це призвело до появи феномену мікро-ферм 3D-друку — невеликих виробничих кластерів, що складаються з десятків уніфікованих принтерів, здатних забезпечувати дрібносерійне виробництво пластикових деталей обсягом від сотень до тисяч одиниць на місяць.

Впровадження концепції «Промисловість 4.0» у цей сегмент передбачає повну цифровізацію виробничого циклу. 3D-принтер перестає бути ізольованим пристроєм, стаючи інтелектуальним терміналом мережі Інтернету IoT. Важливим стає не лише сам факт виготовлення деталі, але й збір телеметричних даних, прогнозування відмов обладнання, облік витратних матеріалів та інтеграція з ERP-системами підприємства. Однак, незважаючи на цифрову зрілість, фізична складова процесу залишається вузьким місцем, оскільки більшість рішень на ринку досі орієнтовані на ручну працю оператора для обслуговування пристрою між циклами друку.

1.1.3 «Феномен Bambu Lab» та особливості закритих програмно-апаратних середовищ

Поява на ринку адитивних технологій компанії Bambu Lab спричинила фундаментальну зміну парадигми користувацького досвіду, яку галузеві експерти часто порівнюють із впливом компанії Apple на ринок смартфонів. До цього моменту сегмент настільних 3D-принтерів був чітко поляризований: з одного боку знаходилися доступні, але технічно недосконалі набори для самостійної збірки (DIY-kits) на базі відкритих прошивок Marlin, а з іншого — дорогі промислові рішення із закритим вихідним кодом. Bambu Lab вдалося інтегрувати передові алгоритми керування рухом, раніше доступні лише у складних конфігураціях прошивки Klipper, у споживчий продукт формату «Plug-and-Play». Для принтерів серії A1 Mini, попри їх спрощену консольну конструкцію, було імплементовано систему активного шумозаглушення двигунів та автоматичного калібрування екструзії за допомогою датчиків вихрових струмів, що дозволило досягти швидкостей друку, які раніше були доступні лише на дорогих кінематичних схемах. [5]

Однак технологічна довершеність апаратної частини була досягнута ціною повної герметизації програмного середовища. На відміну від традиційного для спільноти 3D-друку підходу Open Source, де користувач має повний доступ до модифікації прошивки та конфігураційних файлів, платформа Bambu Lab

побудована за принципом «Walled Garden». Центральним елементом керування виступає хмарний сервіс виробника, через який маршрутизуються всі команди від мобільного додатку Bambu Handy або слайсера Bambu Studio до кінцевого пристрою. Така архітектура гарантує високу стабільність та простоту для пересічного користувача, проте створює суттєві обмеження для інтеграції принтера у сторонні автоматизовані системи.

З технічної точки зору, взаємодія в межах цього середовища базується на модифікованих версіях стандартних IoT-протоколів. Передача команд керування та отримання телеметрії здійснюється через протокол MQTT із використанням шифрування TLS/SSL. При цьому структура корисного навантаження (payload) повідомлень є закритою і не документованою виробником публічно. Це створює ситуацію, за якої розробка власного програмного забезпечення вимагає проведення реверс-інжинірингу мережевого трафіку для виявлення ідентифікаторів команд, необхідних для запуску друку, зупинки процесу або зміни температурних режимів.

Особливої уваги заслуговує реалізація локального режиму роботи (LAN Mode), який дозволяє керувати принтером без підключення до мережі Інтернет. У цьому режимі принтер виступає локальним MQTT-брокером, до якого можна підключитися, маючи код доступу. Проте функціонал LAN-режиму є штучно обмеженим порівняно з хмарним з’єднанням: відсутня історія друку, обмежені можливості лайвстримінгу з камери спостереження та ускладнений процес оновлення прошивки. Крім того, передача файлів G-коду на принтер у локальному режимі здійснюється через застарілий протокол FTP, що вимагає реалізації окремого клієнтського модуля у проєктованій системі керування. [6]

Для задач автоматизації виробничого циклу закритість архітектури створює додаткові виклики. Принтер не надає нативного API для виконання кастомних скриптів після завершення друку, окрім тих, що прописані безпосередньо у G-коді (End G-code). Це означає, що логіка перевірки успішності очищення столу та прийняття рішення про запуск наступного файлу повинна бути винесена на зовнішній керуючий сервер. Розробка такого зовнішнього

менеджера, який би емулював поведінку офіційного додатку, виступаючи для принтера авторизованим клієнтом, є єдиним можливим шляхом створення автономної виробничої комірки на базі обладнання Bambu Lab без втручання в його апаратну частину.

1.1.4 Проблема автоматизації пост-обробки

Незважаючи на стрімкий прогрес у швидкості наплавлення матеріалу, який у сучасних принтерах досягнув високих показників об’ємної витрати, загальна продуктивність адитивного виробництва залишається суттєво обмеженою фактором дискретності процесу. У класичному сценарії використання FDM-принтера виробничий цикл завершується моментом охолодження готового виробу. Подальші дії — зняття деталі, візуальний контроль очищення платформи та запуск наступного файлу — повністю залежать від наявності та кваліфікації оператора. Це створює фундаментальну диспропорцію: час безпосереднього машинного друку типової деталі скоротився з 4–5 годин до 40–60 хвилин, тоді як час очікування обслуговування (наприклад, у нічний час або вихідні дні) залишається незмінним, що критично знижує показник загальної ефективності обладнання.

Процес пост-обробки в контексті FDM-друку включає етапи охолодження, відокремлення виробу від платформи, очищення сопла та перевірки готовності до наступного циклу. Традиційні методи автоматизації цих процесів, запозичені з промислової робототехніки, такі як використання зовнішніх маніпуляторів або конвеєрних стрічок, є економічно невиправданими для настільних принтерів бюджетного сегменту, до якого належить A1 Mini. Вартість роботизованої руки часто перевищує вартість самого принтера в кілька разів, а встановлення конвеєрної стрічки вимагає суттєвого втручання в конструкцію, що призводить до втрати гарантії та зниження жорсткості системи.

В умовах обмеженого бюджету та необхідності збереження штатної конструкції принтера найбільш перспективним напрямком є реалізація так званого «програмного автоматичного скидання». Цей метод базується на

використанні власних кінематичних можливостей принтера: друкувальна голівка, яка має достатню жорсткість та потужність приводів, використовується як штовхач для зміщення охолодженої деталі за межі робочої зони. У випадку з консольною кінематикою, додатковим фактором сприяння є інерція самого столу, різкий рух якого по осі Y може допомогти відокремити деталь.

Головною перешкодою для масового впровадження такого методу є відсутність уніфікованих програмних інструментів. Стандартні слайсери (Bambu Studio, OrcaSlicer) генерують G-код лише для друку, залишаючи етап завершення на рівні "вимкнути нагрів та двигуни". Реалізація автоматичного скидання вимагає динамічного втручання в G-код, точного моніторингу температури столу, оскільки передчасна спроба зсунути деталь може пошкодити покриття PEI або механіку принтера та системи зворотного зв’язку для підтвердження успішності операції. Саме розробка зовнішньої системи керування, яка б взяла на себе логіку координації цих процесів, здатна вирішити проблему "останньої милі" в автоматизації FDM-друку без залучення дороговартісного обладнання.

1.2 Принципи побудови хмарних систем керування IoT-пристроями

Проєктування сучасних автоматизованих систем керування виробничим обладнанням базується на архітектурних патернах IoT. У контексті керування 3D-принтерами, які є типовими представниками кіберфізичних систем, ключовим завданням є забезпечення надійного двонаправленого зв’язку між фізичним пристроєм, що генерує потік телеметричних даних, та віддаленим сервером керування. Специфіка адитивного обладнання накладає жорсткі вимоги до комунікаційних протоколів: вони повинні забезпечувати мінімальну затримку для оперативного реагування на аварійні ситуації, підтримувати роботу в умовах нестабільного мережевого з’єднання та мінімізувати навантаження на вбудований контролер принтера.

Традиційна модель веб-взаємодії, побудована на протоколі HTTP, використовує архітектуру «клієнт-сервер» із синхронним механізмом «запит-відповідь». У такій схемі для отримання актуальних даних про температуру

сопла або прогрес друку клієнт змушений постійно надсилати запити до пристрою (polling). Такий підхід є неефективним для IoT-систем, оскільки створює надлишковий мережевий трафік та значне обчислювальне навантаження на пристрій, більшість ресурсів якого має бути спрямована на керування кінематикою в реальному часі.

Альтернативою, що стала стандартом де-факто в індустрії IoT, є використання асинхронних протоколів обміну повідомленнями, зокрема MQTT. Архітектура MQTT базується на патерні «Publish-Subscribe», який розриває пряму залежність між відправником даних (Publisher) та отримувачем (Subscriber). Центральним елементом такої системи виступає брокер повідомлень — проміжний програмний компонент, який приймає дані від принтера та маршрутизує їх усім зацікавленим клієнтам (веб-додаткам, базам даних, аналітичним сервісам) (рис. 1.4). Використання MQTT дозволяє принтеру відправляти дані лише за фактом їх зміни (event-driven approach), що кардинально знижує навантаження на мережу. Крім того, протокол підтримує механізми контролю якості обслуговування, гарантуючи доставку критично важливих команд керування навіть при короткочасних розривах зв’язку. [7]

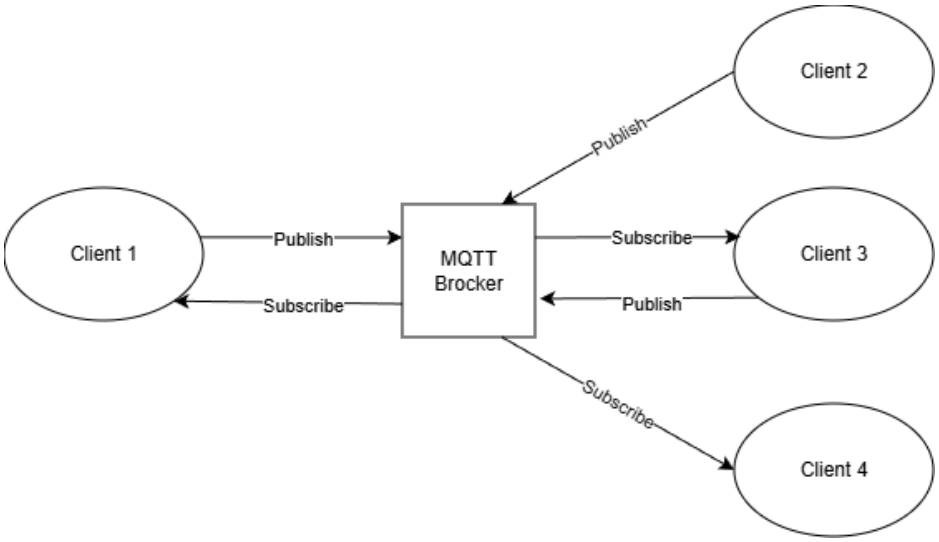

Рисунок 1.4 – Архітектура обміну даними за MQTT патерном «Publish-Subscribe».

Для забезпечення взаємодії з кінцевим користувачем через веб-інтерфейс застосовується комбінація протоколів прикладного рівня. Для передачі статичного контенту, авторизації користувачів та завантаження файлів моделей (G-коду) оптимальним є використання архітектурного стилю REST (Representational State Transfer). REST API забезпечує стандартизований інтерфейс для виконання операцій створення, читання, оновлення та видалення даних (CRUD) поверх протоколу HTTP. Однак, зважаючи на stateless-природу HTTP (відсутність збереження стану з'єднання), він не підходить для відображення динамічних процесів у реальному часі, таких як рух друкувальної голівки на візуалізації.

Для вирішення задачі реактивного оновлення інтерфейсу використовується протокол WebSocket. На відміну від HTTP, WebSocket встановлює постійне повноцінне TCP-з'єднання між клієнтом та сервером. Це дозволяє серверу ініціювати передачу даних клієнту миттєво після їх отримання від MQTT-брокера, без необхідності очікування запиту від користувача. Така зв'язка технологій (MQTT на рівні «пристрій-сервер» та WebSocket на рівні «сервер-клієнт») формує високоефективний транспортний канал для систем моніторингу реального часу. [8]

Важливим аспектом при проєктуванні хмарної архітектури є забезпечення безпеки передачі даних. Оскільки 3D-принтери часто працюють з інтелектуальною власністю (моделями прототипів), канали зв'язку повинні бути захищені криптографічними протоколами транспортного рівня (TLS/SSL). Автентифікація пристроїв та користувачів у сучасних системах реалізується через механізми токенів (наприклад, JWT — JSON Web Token), що дозволяє уникнути передачі облікових даних у відкритому вигляді при кожному запиті. [9]

Таким чином, побудова ефективного веб-менеджера для автоматизації 3D-друку вимагає інтеграції різнорідних протоколів у єдину інформаційну систему: легковагого MQTT для комунікації з обладнанням, надійного REST API для бізнес-логіки та швидкого WebSocket для візуалізації процесів. Такий підхід

забезпечує масштабованість системи та можливість її розширення для керування групами пристроїв.

1.3 Дослідження протоколів взаємодії та структури даних платформи Bambu Lab

Для реалізації зовнішнього керування принтерами Bambu Lab необхідно провести детальний аналіз мережевих інтерфейсів та логіки обміну даними, які закладені виробником. Оскільки компанія не надає офіційної публічної документації для розробників (SDK), дослідження базується на методах аналізу мережевого трафіку та вивченні роботи відкритого програмного забезпечення спільноти. Архітектура зв'язку принтерів Bambu Lab є гібридною і передбачає два основні канали комунікації: канал керування та телеметрії (Command & Control) та канал передачі файлів (File Transfer).

Основним транспортним протоколом для керування станом принтера є MQTT, який функціонує поверх захищеного з'єднання TLS/SSL. У ході аналізу встановлено, що принтер виступає у ролі локального MQTT-сервера (брокера) на порту 8883. Це відрізняє його від класичної IoT-схеми, де пристрої підключаються до зовнішнього хмарного брокера. У локальному режимі (LAN Mode) автентифікація клієнта відбувається без використання сертифікатів центру сертифікації (CA), а через механізм «username/password», де ім'ям користувача завжди виступає «bblp», а паролем — унікальний код доступу, який генерується принтером і відображається в його налаштуваннях. Це критично важливий аспект для проєктування системи безпеки веб-менеджера, оскільки код доступу повинен зберігатися у зашифрованому вигляді. [10]

Структура інформаційного обміну реалізована через систему MQTT-топіків. Принтер підписується на топік запитів **device/{serial_number}/request**, куди керуючий додаток надсилає команди. Відповіді та звіти про статус публікуються у топік **device/{serial_number}/report**. Важливою особливістю реалізації протоколу в принтерах Bambu Lab є те, що принтер не надсилає повний знімок стану постійно. Повне дерево даних надсилається лише при первинному

підключенні або за спеціальним запитом, а в решту часу пристрій транслює лише інкрементальні оновлення (delta updates). Це вимагає від розроблюваного веб-менеджера реалізації локального сховища станів, яке б оновлювалося на основі вхідних повідомлень. [11]

```
{
  "print": {
    "gcode_state": "RUNNING",
    "mc_percent": 45,
    "mc_remaining_time": 1240,
    "nozzle_temper": 220.5,
    "bed_temper": 60.0,
    "fan_gear": 255,
    "sequence_id": "20045",
    "task_name": "Gridfinity_Box.gcode",
    "ams": {
       "ams_exist_bits": "1"
    },
    "layer_num": 145
  }
}
```


Рисунок 1.5 – Фрагмент структури JSON-повідомлення телеметрії принтера.

Аналізуючи JSON-відповіді із даними про телеметрію (рис. 1.5), можемо встановити що об’єкт **print** містить параметри поточного завдання, серед яких критичними для задачі авто-очищення є **gcode_state** (статус виконання), mc_**percent** (відсоток завершення) та **bed_temper** (температура столу).

У ході аналізу технічних форумів та документації спільноти було знайдено команду **gcode_line**. Вона дозволяє вставляти окремі команди G-коду безпосередньо в потік виконання без необхідності генерації нового файлу, що створює технічну базу для реалізації динамічного керування кінематикою принтера через веб-інтерфейс.

Другим компонентом комунікаційної архітектури є протокол передачі файлів. Для завантаження моделей на SD-карту принтера використовується

протокол FTPS (FTP over SSL) на порту 990 з обов’язковим використанням пасивного режиму. Після завантаження файлу його запуск ініціюється окремою командою через канал MQTT. Таким чином, проведене дослідження підтверджує можливість повної програмної автоматизації керування принтером через використання пари протоколів MQTT та FTP, що стане основою архітектури розроблюваного веб-додатку.

1.4 Огляд та порівняльний аналіз існуючих систем автоматизації 3D-друку

Ринок програмного забезпечення для керування 3D-принтерами є висококонкурентним і пропонує спектр рішень від простих слайсерів до комплексних систем керування виробничими фермами. Для коректного формулювання технічного завдання на розробку власного веб-менеджера ми можемо сміливо оцінити функціонал провідних рішень та виявити їх обмеження в контексті повної автоматизації циклу на обладнанні Bambu Lab:

Базовим інструментарієм будь-якого власника принтера Bambu Lab є нативне програмне забезпечення: десктопний слайсер **Bambu Studio** та мобільний додаток **Bambu Handy**. Це глибоко інтегровані продукти, що працюють через хмарну інфраструктуру виробника. [12]

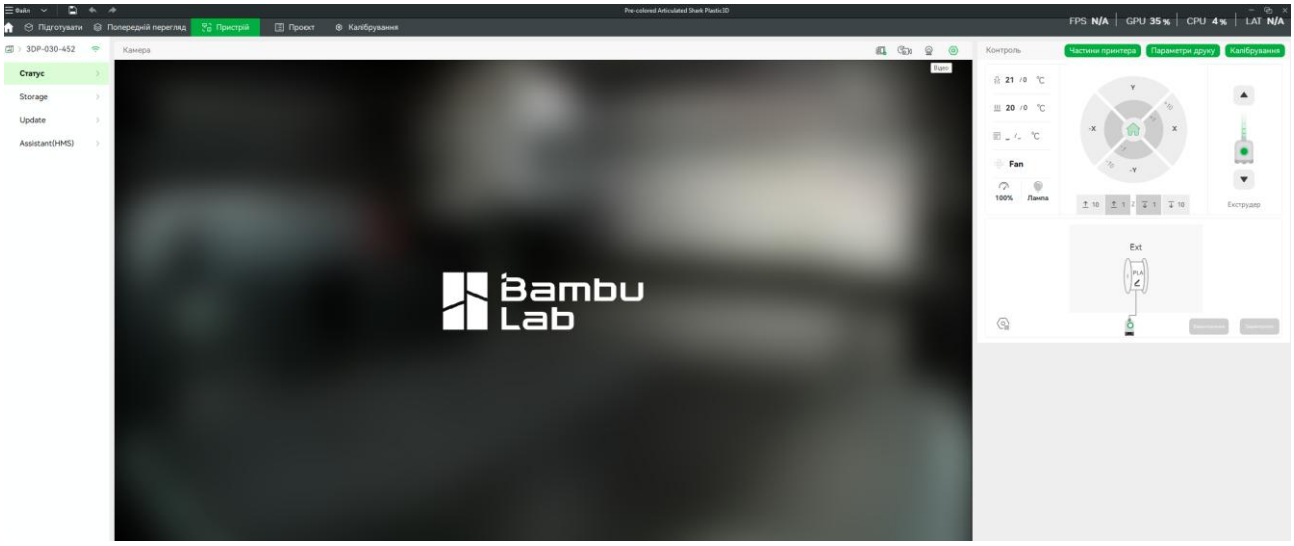

Рисунок 1.6 – Інтерфейс керування принтером у Bambu Studio.

Розглянемо основні параметри, особливості Bambu Studio/Bambu Handy:

- Програмний комплекс забезпечує повний цикл підготовки моделі: від слайсингу до відправки на друк;
- Нативна підтримка всіх апаратних функцій, включаючи калібрування потоку, активне гасіння вібрацій та керування системою зміни кольорів AMS;
- Можливість базового редагування моделі;
- Доступ до лайвстримінгу з камери в реальному часі;
- Отримування push-сповіщень про статус друку та його зміни;
- Великий маркетплейс із готовими деталями та рішеннями від користувачів платформи.

Серед недоліків можна зустріти наступне:

- Відсутня концепція автоматичної черги друку, яка б самостійно запускала наступний файл;
- Відсутності можливості виконання кастомних сценаріїв після друку (окрім стандартного G-коду), що унеможливлює реалізацію логіки перевірки очищення столу.

У відповідь на запит бізнес-користувачів компанія випустила спеціалізоване рішення — **Bambu Farm Server & Manager**. Це програмне забезпечення, призначене для централізованого керування великою кількістю пристроїв у локальній мережі або через хмару. [13]

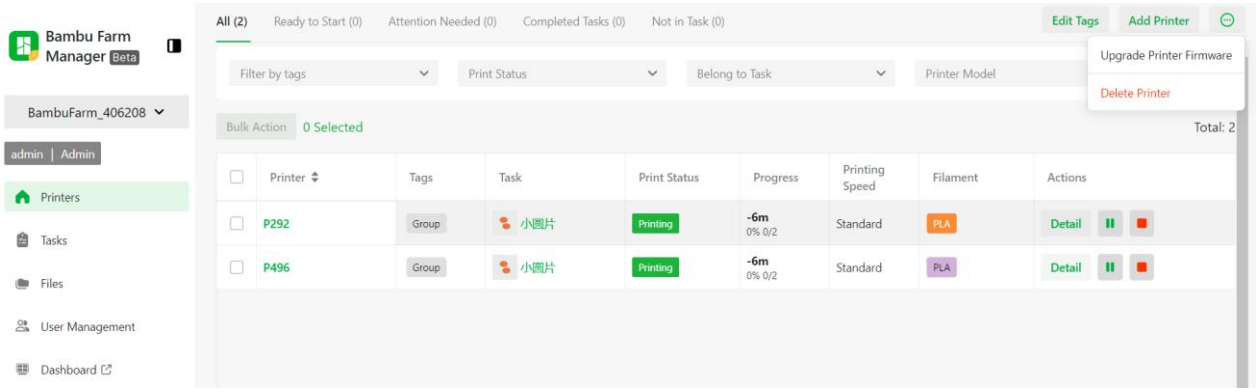

Рисунок 1.7 – Інтерфейс моніторингу групи принтерів у Bambu Farm Manager.

Основні характеристики застосунку **Bambu Farm Server & Manager**:

- Можливість завантажити один файл G-коду та відправити його одночасно на групу обраних принтерів (наприклад, запустити друк однієї деталі на 10 машинах в один клік);

- Централізований моніторинг, а саме відображення статусів усіх пристроїв на єдиній інформаційній панелі з можливістю фільтрації за групами;

- Збір даних про успішність друку, витрати матеріалу та час роботи кожного окремого принтера, що дозволяє проводити аналіз ефективності виробництва;

- Серверна частина може бути розгорнута локально, що зменшує залежність від стабільності зовнішнього інтернет-з’єднання.

Серед недоліків зазначмо наступне:

- Так само як із **Bambu Studio**, система розроблена виключно для керування процесом друку. Вона не містить інструментів для запуску скриптів очищення після завершення завдання;

- Логіка роботи передбачає обов'язкове підтвердження оператором готовності принтера до наступного циклу. Система не може самостійно визначити, чи дійсно стіл є чистим.

- Орієнтованість виключно на обладнання Bambu Lab, що унеможливлює використання у фермах, де є принтери інших брендів).

OctoPrint є найбільш популярною платформою з відкритим вихідним кодом для віддаленого керування 3D-принтерами. Система зазвичай розгортається на базі Raspberry Pi, і виступає проміжним хостом між принтером та мережею.

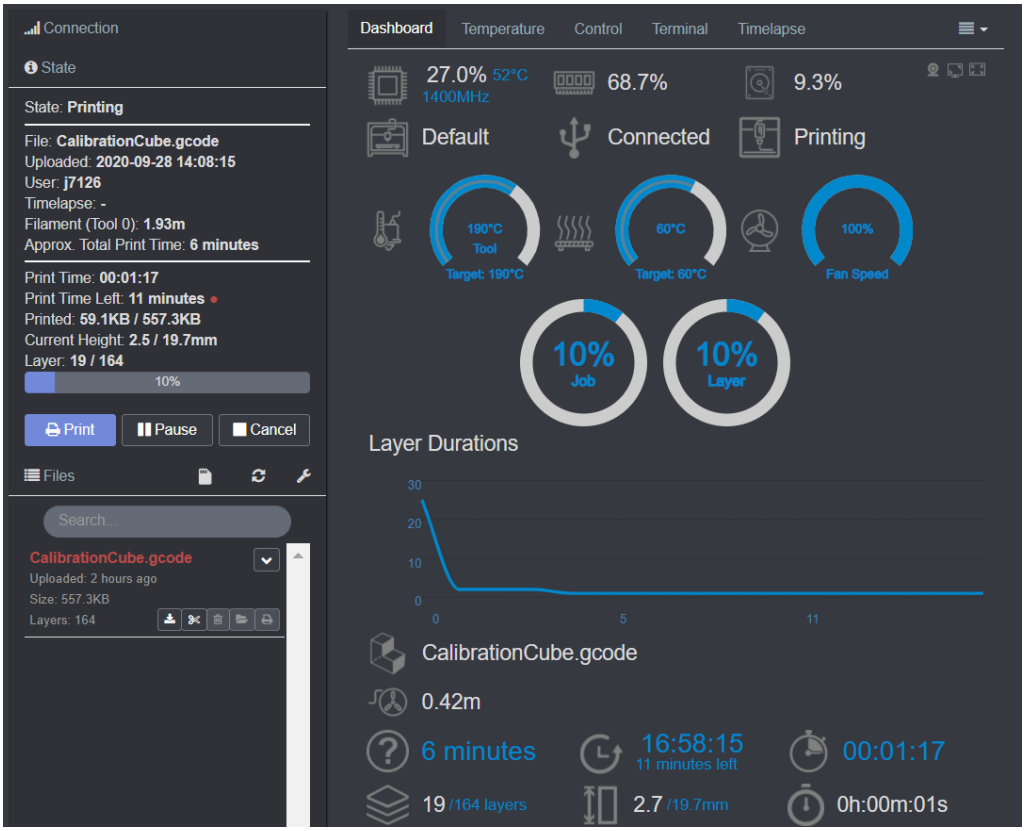

Рисунок 1.8 – Головна панель керування веб-інтерфейсу OctoPrint. [14]

Основні характеристики та функціональні можливості OctoPrint:

- Модульна архітектура платформи підтримує розширення функціоналу через систему плагінів, що дозволяє реалізувати широкий спектр задач: від створення таймлапсу до інтеграції із зовнішніми системами сповіщень;

- Серверна частина може бути розгорнута локально, що зменшує залежність від стабільності зовнішнього інтернет-з'єднання.

- Вбудований візуалізатор G-коду надає можливість переглядати траєкторію руху друкувальної голівки пошарово в реальному часі безпосередньо у вікні браузера;

- Забезпечується прямий доступ до консолі керування принтером, що дозволяє відправляти та діагностувати «сирі» команди G-коду та відповіді прошивки;

- Веб-інтерфейс надає повний доступ до керування принтером через будь-який пристрій у локальній мережі без необхідності встановлення спеціалізованого клієнтського програмного забезпечення.

Виявлені недоліки в контексті роботи із Bambu Lab A1 Mini:

- ПЗ покладається на USB-інтерфейс передачі даних, підтримка якого у принтерах від Bambu Lab є майже під нуль обмеженою на рівні прошивки, що ускладнює пряме керування рухом;

- Існуючі методи інтеграції через протокол MQTT часто працюють у режимі обмеженого доступу, дозволяючи лише зчитувати телеметрію, але не гарантуючи надійного керування запуском друку деталей чи завантаженням файлів;

- Впровадження системи вимагає додаткових апаратних витрат на придбання мікрокомп'ютера для кожного принтера, що знижує економічну ефективність масштабування порівняно з централізованим програмним сервером.

1.5 Технологічні принципи реалізації автоматичного очищення

Забезпечення безперервного виробничого циклу на обладнанні класу Bambu Lab A1 Mini, яке використовує кінематику руху столу, вимагає вирішення проблеми фізичного видалення готової продукції. На основі аналізу механіки процесу FDM-друку та існуючих рішень спільноти (зокрема проєктів 3DQue та JobOx) можна виділити три ключові фактори автоматизації: термодинамічний, механічний та гравітаційний.

1.5.1 Термодинаміка адгезії

Фундаментальною умовою відокремлення деталі є руйнування адгезійного зв'язку між полімером та поверхнею платформи (PEI). Цей процес базується на різниці коефіцієнтів теплового розширення матеріалів. При охолодженні робочої зони нижче температури склування пластику (для PLA < 45°C) виникають внутрішні напруження зсуву, що призводить до ефекту самовільного відшаровування. Тому програмний алгоритм керування повинен базуватися на моніторингу температури столу, забороняючи будь-які механічні дії до завершення фази охолодження.

1.5.2 Принцип механічного змітання

Навіть після термічного відшаровування деталь часто залишається на платформі через залишкову адгезію або силу тертя. Аналіз конструкції друкувальної голівки показує, що використання штатного сопла для силового зміщення виробу є ризикованим через малу площу контакту та загрозу механічного пошкодження хотенду .

Для вирішення цієї проблеми у спільноті набув поширення метод встановлення активних насадок-скребків. Як показано на рис. 1.9, це спеціалізована деталь, що монтується на корпус друкувальної голівки. Вона розташовується нижче рівня сопла і має широку плоску грань, що дозволяє працювати за принципом бульдозерного відвалу. Це дозволяє безпечно трансформувати крутний момент сервоприводів осей X/Y у лінійне зусилля для відділення деталі від поверхні столу.

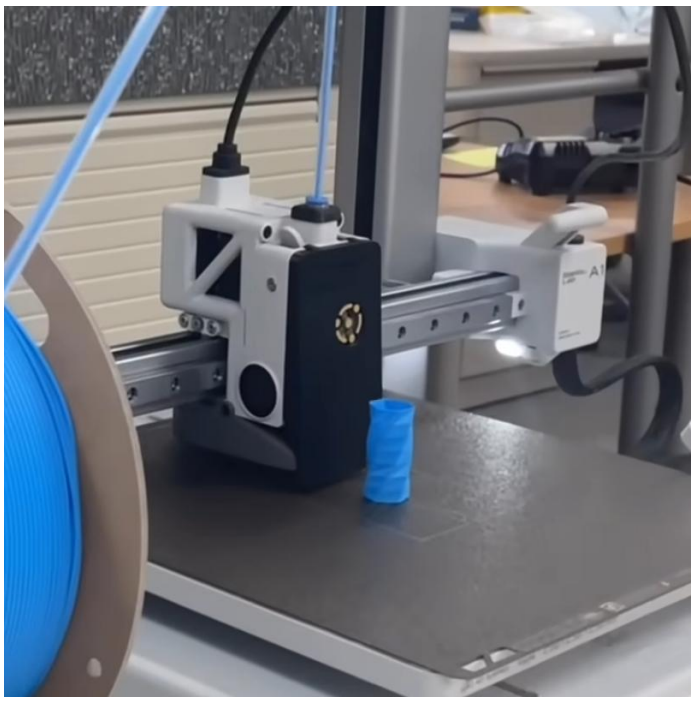

Рисунок 1.9 – Приклад використання модифікованої насадки-скребка для автоматичного очищення платформи для Bambu Lab A1. [15]

1.5.3 Роль гравітаційного впливу

Специфіка принтерів з рухомим столом полягає в тому, що вектор інерції деталі при русі столу спрямований горизонтально. Для підвищення надійності видалення доцільно ввести в систему вертикальну складову сили. Зміна просторової орієнтації принтера (нахил корпусу вперед) дозволяє використати силу тяжіння як допоміжний фактор. Згідно з законами динаміки, на похилій площині сила тертя спокою зменшується, що сприяє легшому сковзуванню деталі у накопичувач після застосування механічного імпульсу від штовхача (рис. 1.10).

Рисунок 1.10 – Реалізація гравітаційного методу автоматизації(від 3DQue): використання похилих опор для пасивного видалення деталей. [16]

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА МЕРЕЖЕВОГО МЕНЕДЖЕРА «CORELOOP»

2.1 Постановка задачі та формування функціональних вимог до системи

Головною метою практичної частини роботи є створення спеціалізованого програмно-апаратного комплексу «CoreLoop», призначеного для автоматизації процесів дрібносерійного виробництва на базі 3D-принтера Bambu Lab A1 Mini. Система розробляється як альтернатива стандартному мобільному додатку Bambu Handy, який орієнтований на поодинокий друк і вимагає ручного втручання оператора після завершення кожного завдання.

На основі аналізу предметної області, проведеного у першому розділі, було сформульовано ключові вимоги до функціоналу проєктованої системи.

Враховуючи специфіку закритого середовища Bambu Lab, система не повинна виконувати функції слайсингу (підготовки G-коду). Натомість, вона має оперувати готовими файлами формату .gcode.3mf, попередньо згенерованими у середовищі Bambu Studio. Основне завдання комплексу — забезпечити автономний конвеєрний запуск цих файлів із автоматичним очищенням робочої зони.

Система повинна реалізувати модуль автоматизації виробничого циклу, який працює за принципом скінченного автомата. Цей модуль має забезпечувати безперервну ініціалізацію наступного файлу з черги одразу після успішного завершення попереднього. Критично важливою вимогою є реалізація інтелектуального алгоритму очищення платформи, який базується на термодинамічних показниках. Програмне забезпечення повинно блокувати будь-які механічні дії до моменту охолодження робочої поверхні до безпечного порогу (40-45°C у залежності від типу філаменту), що гарантує самовільне відшаровування деталі та запобігає пошкодженню PEI-покриття.

Для ефективного керування потоком завдань необхідно розробити динамічну чергу друку з підтримкою пріоритетів. Система має дозволяти

користувачеві завантажувати готові файли на SD-карту принтера через протокол FTP та формувати список відтворення. Важливою є функція пакетної обробки, яка дозволяє вказати необхідну кількість копій для одного файлу. Алгоритм черги повинен автоматично відстежувати кількість вже виготовлених копій та надавати пріоритет завершенню поточної партії перед переходом до наступного унікального файлу.

Інтерфейс користувача повинен забезпечувати моніторинг стану обладнання в режимі реального часу через протокол WebSocket. Оператор має отримувати актуальні дані про цільові та поточні температури сопла і столу, швидкість обертання вентиляторів, а також візуалізувати процес через потокове відео (MJPEG stream) з камери принтера. Також необхідна наявність системного терміналу для трансляції логів сервера та діагностичних повідомлень.

Окремий блок вимог стосується низькорівневого керування апаратною частиною. Система повинна надавати можливість ручного переміщення друкувальної голівки по осях X, Y, Z із програмним обмеженням лімітів переміщення для запобігання зіткненням. Для забезпечення безпеки автоматичного очищення необхідно реалізувати модифіковану процедуру паркування, яка переміщує голівку в координати (0,0,0) замість центру столу, уникаючи колізій із готовими виробами. Окрім цього, передбачається пряме керування периферійними пристроями, такими як вентилятор охолодження хотенду, а також LED-освітленням робочої камери.

Таким чином, розроблюваний комплекс поєднує функції файлового менеджера, планувальника завдань та контролера автоматики, перетворюючи настільний 3D-принтер на автономну виробничу одиницю.

2.2 Проєктування та реалізація апаратних модифікацій принтера

Для фізичної реалізації алгоритмів автоматичного очищення («Auto-Loop») стандартної кінематики принтера Bambu Lab A1 Mini недостатньо. Необхідним етапом роботи стала розробка та виготовлення додаткових навісних

елементів, які перетворюють друкувальну голівку на активний маніпулятор (штовхач) та змінюють просторову орієнтацію пристрою.

Для фізичної реалізації алгоритмів автоматичного очищення («Auto-Loop») стандартної кінематики принтера Bambu Lab A1 Mini недостатньо, оскільки базове сопло не призначене для силового впливу на деталі. Тому необхідним етапом роботи стала розробка та виготовлення додаткових навісних елементів, які перетворюють друкувальну голівку на активний штовхач та змінюють просторову орієнтацію пристрою.

За основу конструкції виконавчого органу було обрано відкрите інженерне рішення «FarmLoop Stage 1» [17], яке являє собою змінну фронтальну панель для друкувальної голівки. У середовищі Bambu Studio було проведено модифікацію вихідної CAD-моделі для адаптації під конкретні виробничі умови (рис. 2.1). Основні зміни стосувалися геометрії контактної площини: було змінено кут атаки штовхача для кращого зчеплення з моделями різної висоти та мінімізації ризику заклинювання деталі під соплом.

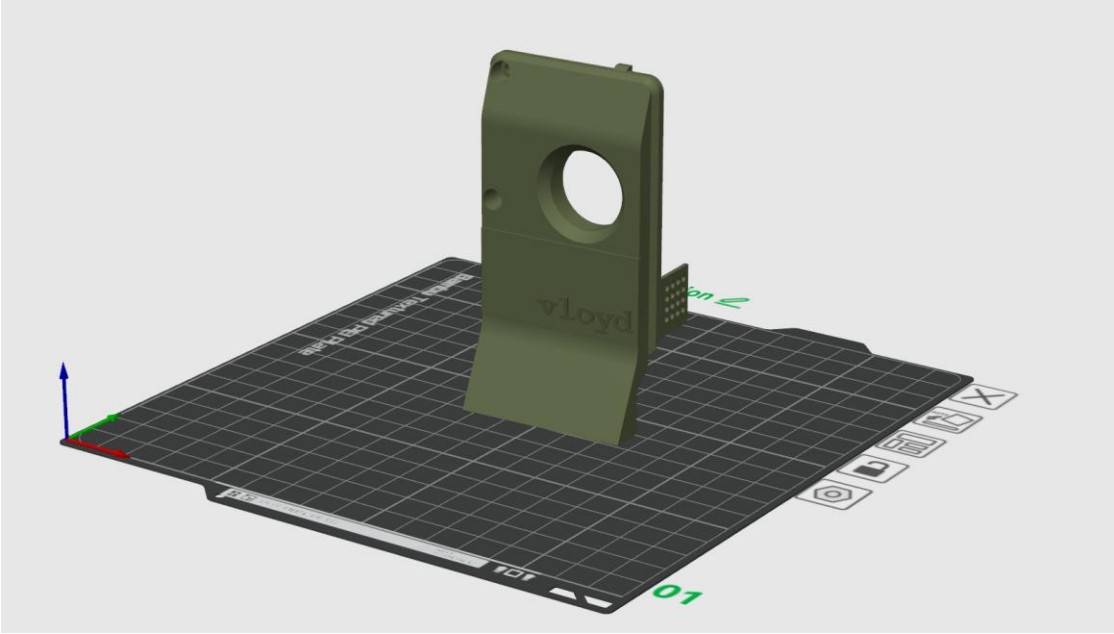

Рисунок 2.1 – Процес підготовки та модифікації моделі насадки-штовхача у середовищі Bambu Studio.

Виготовлення апаратних компонентів здійснювалося методом FDM-друку. Враховуючи умови експлуатації, для друку було обрано поліетилентерефталат-гліколь (PETG). Цей вибір зумовлений його термостійкістю, оскільки насадка знаходиться у безпосередній близькості до нагрівального блоку (Hotend), температура якого може досягати 250°C. PETG зберігає геометричну стабільність при температурах до 80°C, на відміну від PLA, який деформується вже при 55°C. Друк виконувався із заповненням 40% та збільшеною кількістю стінок для забезпечення механічної жорсткості.

Монтаж готової насадки здійснюється на штатні посадкові місця друкувальної голівки замість заводської декоративної заглушки, з використанням гвинтів блоку хотенду. Це дозволяє передавати значні зусилля від сервоприводів осі Y без люфтів. Додатково для забезпечення пасивного видалення деталей принтер встановлено на спеціалізований похилий стенд (рис. 2.2), що створює нахил робочої площини вперед і дозволяє використовувати силу тяжіння для переміщення відшарованих деталей у накопичувач.

Рисунок 2.2 – Загальний вигляд реалізованої апаратної частини системи «CoreLoop»: встановлена насадка-штовхач та похилий стенд.

2.3 Методологія розробки та життєвий цикл проєкту

При розробці програмно-апаратного комплексу «CoreLoop» було обрано ітераційну модель розробки. Враховуючи специфіку роботи з IoT-пристроями та необхідність фізичного тестування механіки, класична каскадна модель виявилася неефективною, оскільки вимоги до апаратної та програмної частини змінювалися в процесі дослідження протоколів принтера.

Ітераційна модель є гнучкою методологією, яка дозволяє поступово вдосконалювати продукт через серію циклів-спринтів. Кожна ітерація є завершеним міні-проєктом, що включає етапи аналізу, проектування, реалізації, тестування та розгортання

Процес розробки системи «CoreLoop» базувався на етапах, відображених на структурній схемі (рис. 2.3).

Рисунок 2.3 – Принципи та етапи ітераційної моделі розробки. [18]

Розглянемо реалізацію ітераційного підходу стосовно даного проєкту детальніше:

- Дослідження – на цьому етапі проводився аналіз закритих протоколів платформи Bambu Lab. Було сформовано технічне завдання та визначено обмеження локального режиму (LAN Mode), що вплинуло на вибір архітектури (відмова від хмарного API на користь прямого MQTT);
- Апаратне проєктування – цей етап включав CAD-моделювання насадки-штовхача та стенду. Завдяки ітераційному підходу було виготовлено кілька прототипів: перші версії тестувалися на сумісність з геометрією хотенду, після чого вносилися корективи в кут нахилу та товщину стінок деталей;

- Розробка бекенду – на цьому етапі було реалізовано скінченний автомат, який забезпечує логіку безпечного охолодження та відправку G-коду на скиданні даних. Тестування проводилося на ізольованих скриптах;

- Інтеграція та інтерфейс – фінальний етап об'єднав апаратну та серверну частини під керуванням веб-інтерфейсу. Було додано функції візуалізації (лайвстримінг, графіки температур) на основі зворотного зв'язку від попередніх тестів.

Використання ітераційної моделі надало проєкту наступні переваги:

- Гнучкість, а саме можливість швидко змінювати параметри G-коду «удару» або інші кодові зміни, чи апаратні зміни, такі як конструкцію насадки без переробки всієї архітектури системи;

- Тестування кожного модуля окремо (спочатку механіка, потім софт) дозволило уникнути критичних збоїв, таких як фізичне пошкодження принтера через некоректні команди руху;

- Система залишалася працездатною на кожному етапі, починаючи від простого скрипта очищення і закінчуючи повноцінним веб-менеджером.

До викликів обраної моделі можна віднести складність планування ресурсів (витрати пластику на прототипи та час на реверс-інжиніринг), проте варто врахувати що робота велась в умовах експериментальної розробки .

Загалом, ітераційна модель є найбільш виправданим способом розробки ПЗ, так як забезпечує гнучкість, високу якість та відповідність продукту. Кожна ітерація-спринт дозволяє поступово вдосконалювати ПЗ, отримувати зворотний зв'язок та швидко реагувати на зміни. Такий підхід забезпечує створення надійного та зручного інструменту, який відповідає сучасним вимогам та реальним очікуванням користувачів.

2.4 Загальна архітектура та логіка функціонування системи

Перед переходом до детального опису програмної реалізації необхідно розглянути загальну архітектуру комплексу «CoreLoop» та алгоритми взаємодії його компонентів. Система спроектована за принципами класичної трирівневої

архітектури, що забезпечує модульність, масштабованість та чітке розмежування зон відповідальності.

Структурна схема інформаційної взаємодії компонентів системи наведена на рис. 2.4.

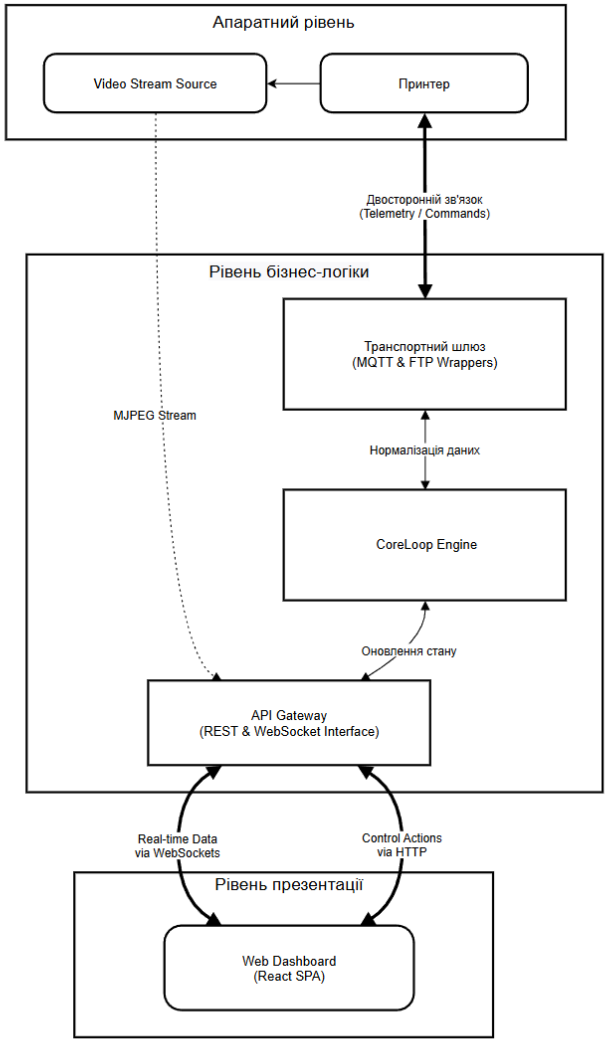

Рисунок 2.4 – Архітектурна схема проєкту «CoreLoop».

Як видно зі схеми, архітектура комплексу складається з трьох основних рівнів:

- Апаратний рівень – це фізичний рівень, до якого належать безпосередньо 3D-принтер Bambu Lab A1 Mini та джерело відеосигналу. Головною функцією цього рівня є виконання механічних операцій та генерація

первинних даних (телеметрії, відеопотоку). Взаємодія з вищим рівнем відбувається через мережеві інтерфейси;

- Рівень бізнес-логіки – це центральний елемент системи, який виконує роль "мозку" комплексу. Він складається з трьох ключових модулів транспортного шлюзу, CoreLoop Engine, API Gateway. Транспортний шлюз відповідає за абстрагування низькорівневих протоколів. Він містить обгортки-враппери для MQTT та FTP, забезпечуючи двосторонній зв'язок із принтером. Шлюз приймає "сирі" дані, нормалізує їх та передає далі. Модуль, що реалізує основні алгоритми автоматизації. У CoreLoop Engine працює скінченний автомат, який приймає рішення про запуск наступного друку, моніторить температуру та керує процесом очищення. Він отримує нормалізовані дані від шлюзу та оновлює внутрішній стан системи. API Gateway є інтерфейсом для зовнішньої взаємодії. Він організовує передачу даних на клієнтську частину через REST API (для команд керування) та WebSocket (для потокової передачі стану в реальному часі). Також через цей модуль маршрутизується MJPEG-лайвстримінг;

- Рівень презентації – це клієнтська частина, реалізована у вигляді веб-дашборду. Вона відповідає за візуалізацію даних для оператора та прийом команд користувача. Завдяки технології Single Page Application, цей рівень миттєво реагує на зміни, отримані від API Gateway, без перезавантаження сторінки.

Ключовим елементом інтелектуальної складової системи є модуль CoreLoop Engine. Оскільки процес автоматичного циклу є нелінійним і залежить від багатьох умов (температура, наявність файлів, помилки), він реалізований як скінченний автомат. Логіка переходів між станами системи детально зображена на рис. 2.5.

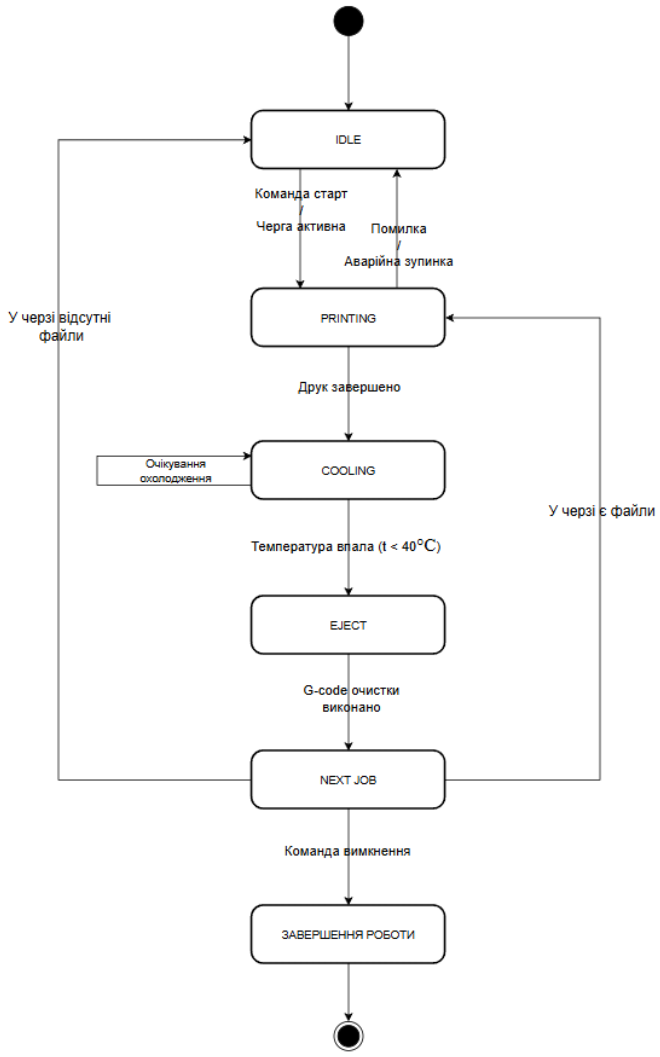

Рисунок 2.5 – Діаграма станів CoreLoop Engine.

Алгоритм передбачає наступні ключові стани та логіку переходів:

- IDLE – початковий стан системи. Перехід до активної роботи відбувається лише за умови отримання команди «Старт» та наявності активної черги файлів;

- PRINTING – активна фаза виконання завдання. У разі успішного завершення система переходить до охолодження. У випадку передбачено аварійне повернення у стан IDLE для запобігання пошкодженню обладнання;

- COOLING – після виконання друку, система переходить у стан очікування, періодично перевіряючи температуру робочої поверхні. Перехід до наступного етапу відбувається виключно при досягненні безпечного порогу в 40°C що гарантує безпечне відшаровування деталі;

- EJECT – виконання спеціалізованого G-code сценарію для механічного видалення готового виробу зі столу;

- NEXT JOB – вузол прийняття рішень після очищення столу. Система аналізує стан черги друку. Якщо у черзі є файли, відбувається зациклення алгоритму — повернення до стану PRINTING, або у стан IDLE за відсутності черги.

Підсумовуючи, запропонована архітектура системи «CoreLoop» вирішує ключову проблему синхронізації різнорідних процесів. Чітке розмежування рівнів абстракції дозволяє незалежно модифікувати апаратну частину або інтерфейс користувача без необхідності переписання ядра бізнес-логіки.

Використання моделі скінченного автомата для модуля керування гарантує детерміновану поведінку системи: принтер ніколи не опиниться у невизначеному стані, а критичні операції, такі як механічне очищення, будуть виконані лише за суворого дотримання умов безпеки. Такий підхід перетворює набір розрізнених скриптів на надійний промисловий інструмент, здатний працювати в автономному режимі тривалий час. Сформована архітектура стає фундаментом для вибору конкретних програмних засобів та написання коду, що буде детально розглянуто в наступних підрозділах.

2.5 Обґрунтування вибору інструментальних засобів та технологій реалізації

2.5.1 Засоби реалізації серверної частини

Реалізація серверної складової системи «CoreLoop» висуває специфічні вимоги до платформи розробки. Оскільки система повинна одночасно утримувати постійне з'єднання з 3D-принтером через протокол MQTT, виконувати фонові алгоритми автоматизації та миттєво відповідати на HTTP-запити користувача, критичним фактором вибору стала підтримка асинхронного вводу-виводу.

Проаналізувавши доступні варіанти, як основну мову програмування було обрано Python (версія 3.10+). Цей вибір зумовлений не лише розвиненою

розвиненим набором бібліотек для роботи з IoT та мережевими протоколами, але й нативною підтримкою асинхронності що дозволяє ефективно керувати сотнями конкурентних з'єднань на одному потоці процесора. Для побудови архітектури бекенду було сформовано наступний технологічний стек:

- **FastAPI** було використано як основний веб-фреймворк. На відміну від класичних Flask або Django, FastAPI базується на стандарті ASGI. Це критично важливо для реалізації WebSocket-з'єднань, через які транслюється телеметрія в реальному часі без блокування основного потоку виконання; [19]
- **Asyncio** виступає «двигуном» системи, дозволяючи виконувати фонові задачі такі як моніторинг температури, логіка Auto-Loop паралельно з обробкою запитів API; [20]
- **Paho-MQTT** використовувалась як бібліотека клієнта для протоколу MQTT. Вона забезпечує реалізацію патерну «Publish-Subscribe», підтримку QoS та автоматичне відновлення з'єднання при розривах мережі; [21]
- **Bambulabs_api** є спеціалізованою бібліотека-обгортка, яка спрощує взаємодію з принтерами Bambu Lab, надаючи зручні методи для парсингу «сирих» JSON-повідомлень та керування підсистемами, такими як світло, вентилятори, рух. [22]

2.5.2 Засоби реалізації клієнтської частини

Головним завданням клієнтського інтерфейсу є надання оператору актуальної інформації про стан виробничого процесу з мінімальною затримкою. Враховуючи динамічний характер даних (зміна температури, координат, прогресу друку відбувається декілька разів на секунду), використання класичного підходу з перезавантаженням веб-сторінки було визнано неефективним.

Для забезпечення користувацького досвіду, наближеного до нативних десктопних додатків, було обрано архітектурний підхід Single Page Application. Це дозволяє завантажувати оболонку додатку лише один раз, а надалі оновлювати лише змінні дані через фонові запити. Для реалізації цього підходу було обрано наступний набір інструментів:

- **React.js** було обрано для побудови інтерфейсу на основі компонентів. Це дозволяє створювати модульний код, який легко масштабувати та підтримувати, забезпечуючи швидку реакцію інтерфейсу на дії користувача; [23]

- **Zustand** використовується як легковаговий менеджер глобального стану додатку. Він виступає «єдиним джерелом правди», миттєво синхронізуючи дані телеметрії, отримані через WebSocket, з усіма віджетами дашборду без зайвого ускладнення коду; [24]

- **Tailwind CSS** застосовано як utility-first фреймворк, що дозволив швидко реалізувати адаптивний дизайн із підтримкою темної теми без необхідності написання та підтримки громіздких каскадних таблиць стилів. [25]

2.5.3 Протоколи обміну даними

Архітектура інформаційної взаємодії системи «CoreLoop» базується на трьох основних протоколах, кожен з яких вирішує специфічні задачі транспортування даних у розподіленій системі:

- **MQTT** використовується для двостороннього обміну даними між сервером та фізичним принтером. Перевагою протоколу є мінімальний розмір заголовка пакета та робота за моделлю «Publish-Subscribe», що дозволяє отримувати телеметрію в реальному часі з мінімальними затримками та гарантує доставку команд керування;

- **WebSocket** забезпечує канал постійного зв'язку між сервером та веб-інтерфейсом користувача. На відміну від класичних HTTP-запитів, цей протокол підтримує повнодуплексний режим, дозволяючи серверу ініціювати відправку оновлених даних клієнту миттєво при зміні статусу друку або температури;

- **FTP** застосовано для маніпуляцій з файловою системою принтера, зокрема для завантаження файлів формату «.3mf» на SD-карту та отримання списку моделей. Вибір зумовлений тим, що це єдиний відкритий спосіб передачі файлів у локальному режимі роботи принтера.

2.5.4 Обґрунтування вибору інструментальних засобів розробки

У процесі розробки серверної частини програмно-апаратного комплексу «CoreLoop» було обрано PyCharm як основне інтегроване середовище розробки

(IDE). Це рішення було прийнято завдяки численним перевагам, які пропонує цей інструмент для створення складних асинхронних додатків на мові Python. PyCharm є потужним та гнучким інструментом, який забезпечує зручну роботу з віртуальними середовищами та зовнішніми залежностями. [26]

Серед ключових можливостей, які стали у пригоді під час розробки, варто відзначити глибоку інтеграцію з бібліотеками FastAPI та Paho-MQTT. Вбудовані інструменти аналізу коду дозволяють автоматично перевіряти відповідність стандартам PEP8, що є критично важливим для підтримки чистоти архітектури проєкту. Не менш важливою перевагою PyCharm є його потужна система дебагу, яка дозволяє розробнику зупиняти виконання програми в реальному часі, інспектувати стан змінних та відстежувати поведінку скінченного автомата і MQTT-з'єднань, що значно економить час при пошуку логічних помилок у роботі алгоритмів автоматизації.

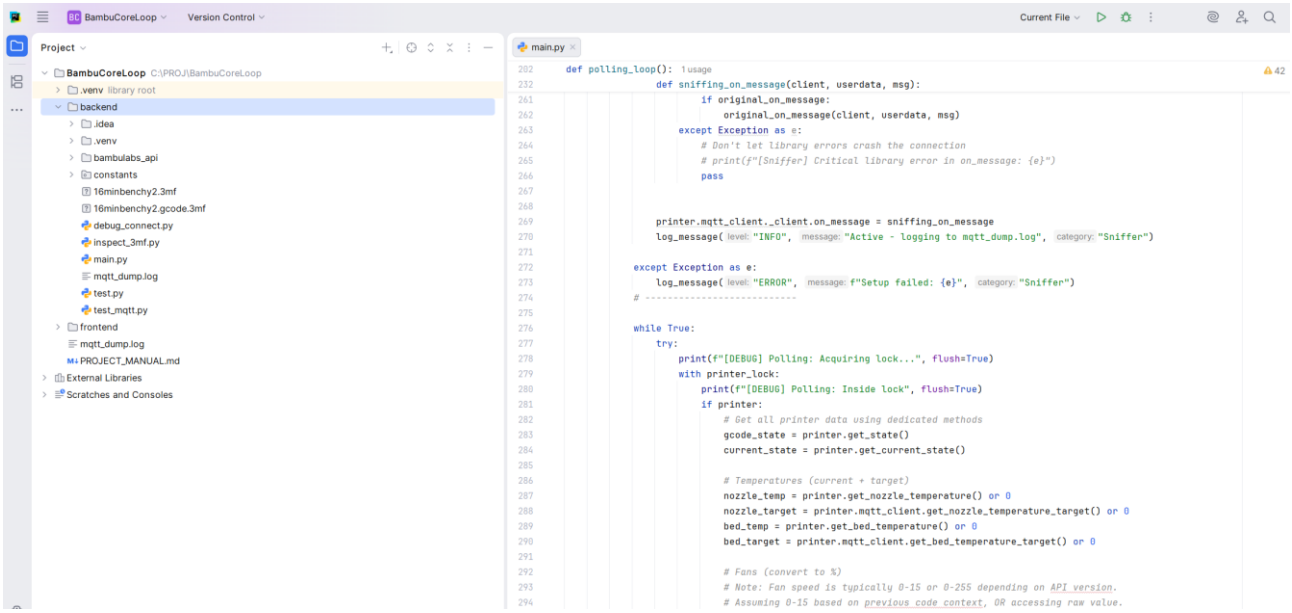

Рисунок 2.6 – Інтерфейс IDE PyCharm.

Для реалізації клієнтського веб-інтерфейсу було обрано редактор коду Visual Studio Code (VS Code). Він став незамінним інструментом завдяки своїй легкості та широкій бібліотеці розширень для веб-розробки. VS Code забезпечує відмінну підтримку синтаксису JSX та бібліотеки React, надаючи інструменти

інтелектуального автодоповнення. Крім того, наявність інтегрованого термінала дозволяє ефективно керувати пакетами npm та запускати локальний сервер розробки без необхідності перемикання між вікнами, що дозволяє зосередитися на безпосередній реалізації функціоналу інтерфейсу. [27]

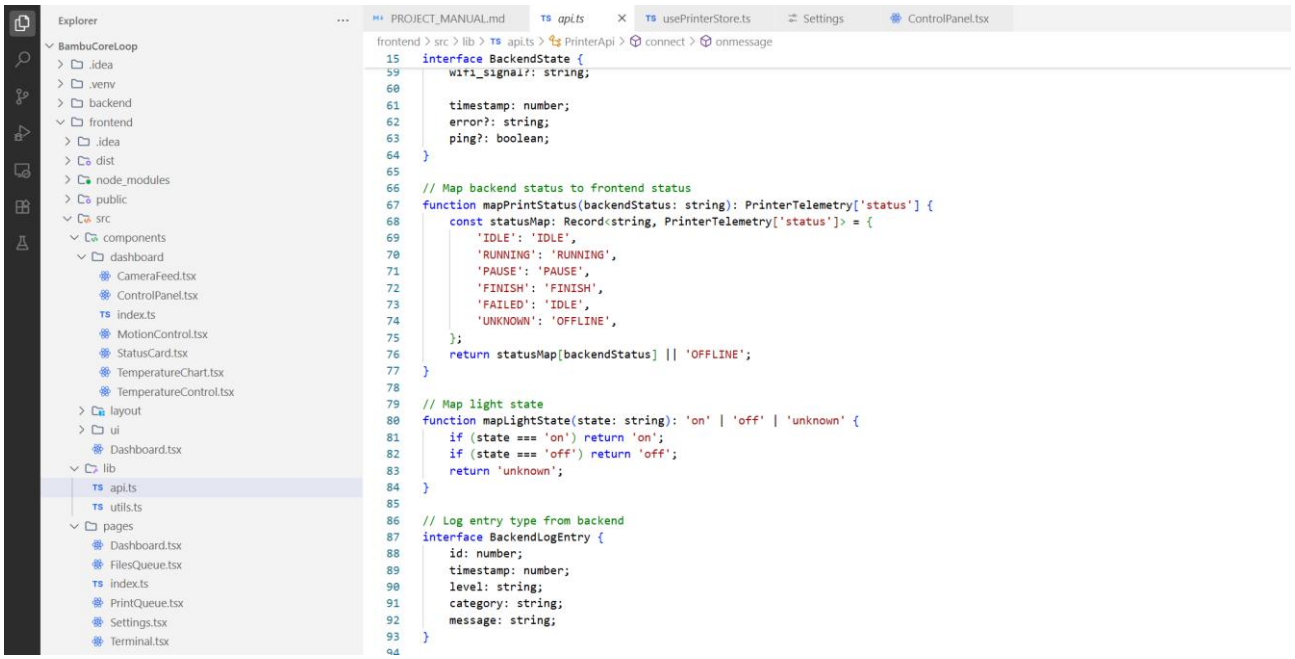

Рисунок 2.7 – Інтерфейс редактору коду Visual Studio Code.

2.6 Особливості програмної реалізації

2.6.1 Реалізація підсистеми моніторингу та низькорівневої комунікації

Фундаментом стабільної роботи системи «CoreLoop» є підсистема комунікації, яка забезпечує безперервний обмін даними між сервером керування та 3D-принтером. Враховуючи, що бібліотека bambulabs_api розроблена для синхронного виконання, а веб-сервер FastAPI працює в асинхронному режимі (Event Loop), було реалізовано гібридну архітектуру з використанням окремого системного потоку (threading.Thread).

Основним механізмом підтримки з'єднання є функція polling_loop, яка працює у фоновому режимі та ізольована від обробки HTTP-запитів користувачів. Це дозволяє уникнути блокування інтерфейсу при затримках мережі.

```python
def polling_loop():  1 usage
    """Background thread that polls printer state every 2 seconds."""
    global printer, latest_state

    consecutive_failures = 0
    MAX_CONSECUTIVE_FAILURES = 60 # Increased tolerance for temporary glitches

    # Outer loop to handle reconnection attempts
    while True:
        log_message( level: "INFO",  message: f"Connecting to printer at {IP}...",  category: "Polling")

        try:
            with printer_lock:
                printer = bl.Printer(IP, ACCESS_CODE, SERIAL)
                try:
                    printer.connect()
                except TimeoutError:
                    log_message( level: "WARN",  message: "Connection timed out (Printer might be offline)",  category: "Polling")
                    printer = None
                    time.sleep(5)
                    continue
                except Exception as e:
                    log_message( level: "ERROR",  message: f"Connection failed: {e}",  category: "Polling")
                    printer = None
                    time.sleep(5)
                    continue

            log_message( level: "INFO",  message: "Connected! Starting polling loop...",  category: "Polling")
            consecutive_failures = 0 # Reset counter on successful connection
```

Рисунок. 2.8 – Реалізація механізму підключення та політики відновлення з'єднання.

Як видно з рис. 2.8, використання конструкції while True на верхньому рівні функції гарантує, що процес моніторингу є нескінченним. У випадку виникнення виключення TimeoutError або будь-якої іншої мережевої помилки, система не завершує роботу аварійно, а переходить у режим очікування (time.sleep(5)) і повторює спробу підключення.

Для запобігання стану гонитви коли веб-сервер намагається відправити команду одночасно з оновленням статусу, використано примітив синхронізації printer_lock (Threading Lock). Усі операції з об'єктом printer виконуються виключно всередині контекстного менеджера with printer_lock.

```python
while True:
    try:
        print(f"[DEBUG] Polling: Acquiring lock...", flush=True)
        with printer_lock:
            print(f"[DEBUG] Polling: Inside lock", flush=True)
            if printer:
                # Get all printer data using dedicated methods
                gcode_state = printer.get_state()
                current_state = printer.get_current_state()

                # Temperatures (current + target)
                nozzle_temp = printer.get_nozzle_temperature() or 0
                nozzle_target = printer.mqtt_client.get_nozzle_temperature_target() or 0
                bed_temp = printer.get_bed_temperature() or 0
                bed_target = printer.mqtt_client.get_bed_temperature_target() or 0

                # Fans (convert to %)
                # Note: Fan speed is typically 0-15 or 0-255 depending on API version.
                # Assuming 0-15 based on previous code context, OR accessing raw value.
                # Let's try to be safe and check attributes or use mqtt_client direct access if needed.

                # Using methods from previous successful version:
                part_fan_raw = printer.mqtt_client.get_part_fan_speed() or 0
                aux_fan_raw = printer.mqtt_client.get_aux_fan_speed() or 0

                # Convert 0-255 (typical for mqtt) to percentage? Or is it 0-15?
                # Bambu usually sends 0-15 via some endpoints but MQTT often has raw.
                # Let's assume standard logic:
                part_fan_percent = int((part_fan_raw / 15) * 100) if part_fan_raw <= 15 else int((part_fan_raw / 255) * 100)
                aux_fan_percent = int((aux_fan_raw / 15) * 100) if aux_fan_raw <= 15 else int((aux_fan_raw / 255) * 100)
```

Рисунок. 2.9 – Використання printer_lock для запобігання ламання стану принтера.

Збір даних телеметрії виконується шляхом агрегації показників у єдиний словник latest_state. Це дозволяє нормалізувати дані, отримані з різних джерел і привести їх до зручного для фронтенду формату (рис. 2.11).

```python
is_active = str(gcode_state) == "RUNNING"

latest_state = {
    "status": "connected",
    "print_status": str(gcode_state),
    "printer_status": str(current_state),
    # Temperatures
    "nozzle_temp": nozzle_temp,
    "nozzle_target": nozzle_target,
    "bed_temp": bed_temp,
    "bed_target": bed_target,
    # Progress
    "print_percentage": print_percentage,
    "remaining_time": remaining_time if is_active else 0,
    "current_layer": current_layer if is_active else 0,
    "total_layers": total_layers if is_active else 0,
    # File
    "file_name": file_name,
    "subtask_name": subtask_name,
    # Controls
    "light_state": light_state,
    "print_speed": print_speed,
    # Info
    "wifi_signal": wifi_signal,
    # Fans (0-100%)
    "part_fan_speed": part_fan_percent,
    "aux_fan_speed": aux_fan_percent,
    # Auto-loop state
    "auto_loop_enabled": auto_loop_enabled,
    "auto_loop_state": auto_loop_state,
    "timestamp": time.time(),
}
```

Рисунок. 2.11 – Агрегація та нормалізація даних телеметрії.

Особливістю реалізації є конвертація "сирих" значень, таких як швидкість вентиляторів, яка може надходити у форматі 0-255 або 0-15, у відсотки. Це спрощує відображення даних на графічному інтерфейсі користувача.

2.6.2 Реверс-інжиніринг протоколу та реалізація MQTT-сніфера

У процесі розробки було виявлено, що офіційна документація API не описує команди запуску друку файлів, які вже знаходяться на SD-карті принтера (команда project_file). Для дослідження структури цих пакетів було розроблено програмний модуль «Sniffer» інтегрований безпосередньо в код бекенду.

Сніфер підписується на wildcard-топік device/{serial_ number}/#, що дозволяє отримувати абсолютно всі повідомлення, якими обмінюється принтер, включаючи команди від офіційного хмарного сервісу. Для цього було застосовано техніку «Monkey Patching» — динамічної підміни методу on_message бібліотеки Paho-MQTT.

```
# --- MQTT SNIFFER SETUP ---
try:
    # Subscribe to wildcard topic to capture ALL traffic (requests and reports)
    # pattern: device/{serial}/#
    sniffer_topic = f"device/{printer.serial}/#"
    printer.mqtt_client._client.subscribe(sniffer_topic)
    log_message( level: "INFO",  message: f"Sniffer subscribed to {sniffer_topic}",  category: "Sniffer")

    # Monkey-patch the on_message callback
    original_on_message = printer.mqtt_client._client.on_message

    def sniffing_on_message(client, userdata, msg):
        try:
            topic = msg.topic

            # Try decode payload, handle binary safely
            try:
                payload_str = msg.payload.decode('utf-8')
            except UnicodeDecodeError:
                # It might be an image chunk or binary data
                payload_str = f"<BINARY DATA len={len(msg.payload)}>"

            # Log to file
            with open("mqtt_dump.log", "a", encoding="utf-8") as f:
                timestamp = time.strftime("%Y-%m-%d %H:%M:%S")
                f.write(f"[{timestamp}] TOPIC: {topic}\n")
                f.write(f"{payload_str}\n")
                f.write("-" * 50 + "\n")
            if "request" in topic and "project_file" in payload_str:
                 log_message( level: "INFO",  message: f"CAPTURED REQUEST: {payload_str[:200]}...",  category: "Sniffer")

        except Exception as e:
            # Don't spam main logs with file errors, maybe print once
            print(f"[Sniffer] Log error: {e}")

        try:
            if original_on_message:
                original_on_message(client, userdata, msg)
        except Exception as e:
            log_message( level: "ERROR",  message: f"Original handler error: {e}",  category: "Sniffer")

    printer.mqtt_client._client.on_message = sniffing_on_message
    log_message( level: "INFO",  message: "Sniffer active - logging to mqtt_dump.log",  category: "Sniffer")
```

Рисунок. 2.12 – Програмний код перехоплювача MQTT-повідомлень.

Цей код виконує роль проміжної ланки. Він перехоплює вхідне повідомлення, записує його у файл mqtt_dump.log, а потім передає оригінальному обробнику (original_on_message), щоб не порушувати роботу основної бібліотеки.

Результат роботи сніфера виводиться в реальному часі в термінал адміністратора на веб-дашборді, що дозволяє аналізувати трафік без доступу до серверної консолі.

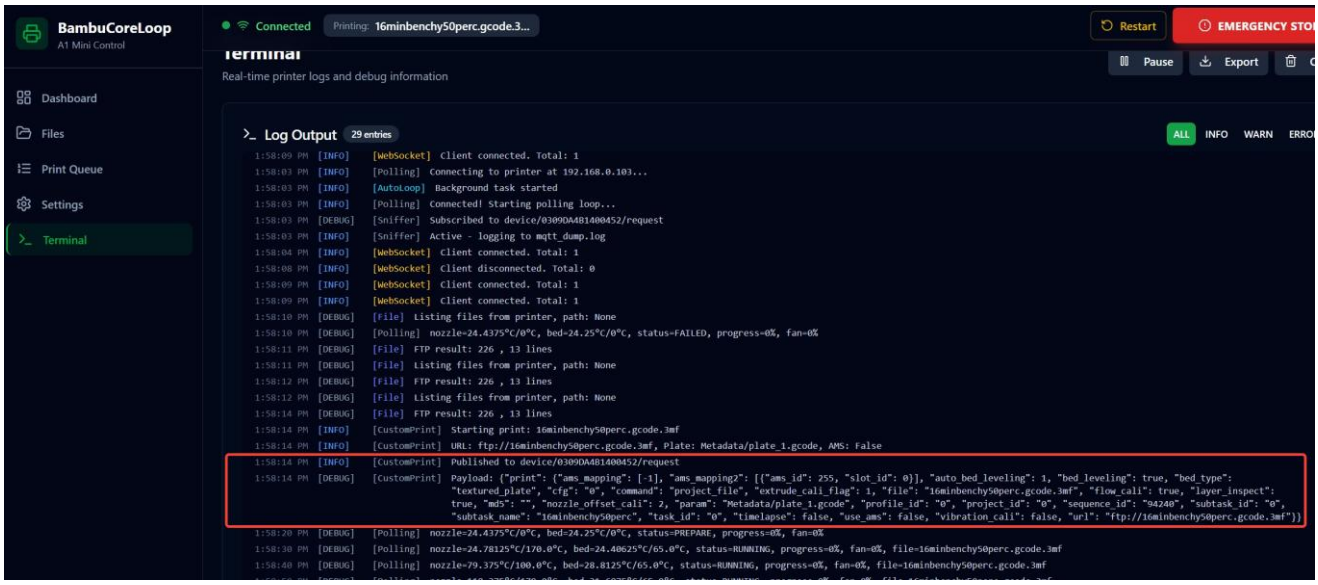

Рисунок. 2.13 – Інтерфейс системного терміналу з логами роботи сніфера.

Як видно на рис. 2.13, система логує події підключення, а також перехоплені запити. Червоним виділено запит на відправку файлу на друк, на основі чого було виділено структуру JSON-об'єкта для команди друку, що буде розглянуто в пункті 2.6.5.

2.6.3 Програмна реалізація алгоритму автоматизації CoreLoop

Ядром системи є модуль автоматизації CoreLoop, який реалізує логіку скінченного автомата. Цей модуль працює як окрема асинхронна задача (asyncio.Task) паралельно з веб-сервером.

Функція auto_loop_task циклічно перевіряє стан принтера та глобальну змінну auto_loop_state. Логіка передбачає перехід між станами: IDLE → COOLING → EJECTING → COMPLETED .

Розглянемо реалізацію ключового етапу — переходу від завершення друку до контрольованого охолодження:

```
while True:
    try:
        if auto_loop_enabled:
            with printer_lock:
                if printer is None:
                    await asyncio.sleep(5)
                    continue

                state = str(printer.get_state())
                bed_temp = printer.get_bed_temperature() or 0

            # State machine for auto-loop
            if auto_loop_state == AUTO_LOOP_STATE_IDLE:
                # Check if print just finished
                if state == "FINISH":
                    log_message( level: "INFO",  message: f"Print finished! Starting cooldown from {bed_temp}°C",  category: "AutoLoop")
                    auto_loop_state = AUTO_LOOP_STATE_COOLING
                    cooling_state["is_cooling"] = True
                    cooling_state["start_temp"] = bed_temp
                    cooling_state["start_time"] = time.time()
                    cooling_state["last_temp"] = bed_temp
```

Рисунок. 2.14 – Алгоритм моніторингу охолодження та прогнозування часу.

Коли система фіксує статус FINISH, вона ініціалізує змінні для розрахунку динаміки охолодження. На відміну від простих систем, що чекають фіксований час, CoreLoop аналізує реальну температуру столу (bed_temp).

Алгоритм також вираховує швидкість охолодження (cooling_rate) — на скільки градусів падає температура за секунду. Це значення було визначено як 0.5°C за секунду. Це дозволяє прогнозувати час, що залишився до досягнення цільової температури 40°C, при якій відбувається ефект самовільного відшаровування деталі.

```
elif auto_loop_state == AUTO_LOOP_STATE_COOLING:
    # Update cooling rate estimation
    if cooling_state["last_temp"] > bed_temp:
        elapsed = time.time() - cooling_state["start_time"]
        if elapsed > 0:
            temp_drop = cooling_state["start_temp"] - bed_temp
            cooling_state["cooling_rate"] = temp_drop / elapsed
    cooling_state["last_temp"] = bed_temp

    # Check if cooled enough
    if bed_temp <= COOLING_TARGET_TEMP:
        log_message( level: "INFO",  message: f"Bed cooled to {bed_temp}°C. Ejecting part...",  category: "AutoLoop")
        auto_loop_state = AUTO_LOOP_STATE_EJECTING
        cooling_state["is_cooling"] = False
```

Рисунок. 2.15 – Алгоритм моніторингу охолодження та прогнозування часу.

Інформація про прогрес охолодження транслюється на клієнтську частину, де відображається у вигляді інтерактивного статус-бару в черзі друку (рис 2.16).

Рисунок. 2.16 – Візуалізація стадії охолодження та прогнозу часу.

На рисунку видно статус «Cooling Down for Ejection», поточну температуру поверхні (60°C) та розрахунковий час (6 хв 39 с).

Після досягнення безпечної температури виконується перехід у стан EJECTING, де система відправляє спеціалізований G-код для механічного очищення столу.

2.6.4 Програмна реалізація механічного очищення робочої зони

Критичним етапом автоматизованого циклу є фізичне видалення готового виробу з робочої платформи. Оскільки конструкція 3D-принтера Bambu Lab A1 Mini не передбачає вбудованих механізмів очищення, було розроблено програмний метод що перетворити друкуючу головку на повноцінний маніпулятор-штовхач.

Реалізація цього методу базується на використанні мови G-code — стандартизованої мови керування верстатами з ЧПК. Це дозволяє отримати прямий контроль над кінематикою принтера, оминаючи стандартні обмеження прошивки, які зазвичай запобігають зіткненням. Сценарій очищення зберігається у константі CLEAR_BED_GCODE.

```
# G-code for clearing the bed after print
CLEAR_BED_GCODE = """
M104 S0
M140 S0
G90
G1 X90 Y180 Z3 F6000
G1 Z30 F3000
G28
"""
```

Рисунок. 2.17 – G-code сценарій для механічного видалення деталі.

Алгоритм дії сценарію наступний:

- Підготовка – повне відключення нагрівальних елементів сопла та столу (команди M104, M140). Це є необхідною умовою для безпеки та переходу пристрою в режим енергозбереження перед очікуванням наступної задачі;
- Позиціонування – друкуюча голова переміщується у точку з координатами X90 Y180 (центр задньої частини платформи) та опускається на висоту Z3 (3 мм над поверхнею). Це значення висоти розраховано експериментальним шляхом: воно є достатньо низьким для надійного зачеплення деталі за основу, проте залишається безпечним для текстурованого PEI-покриття столу;
- Удар – команда G1 Y0 F12000 ініціює різкий рух голови до переднього краю столу з максимальною швидкістю (200 мм/с). Завдяки попередньому етапу охолодження (див. п. 2.6.3), адгезія деталі до поверхні є мінімальною, тому вона легко відшаровується під дією механічного імпульсу;

- Паркування та перекалібрування – фінальна команда G28 (Auto Home) є критично важливою для надійності системи. Оскільки під час фізичного удару об деталь крокові двигуни можуть пропустити кроки (через механічний опір), фактичні координати голови можуть відрізнятися від програмних. Процедура паркування змушує принтер заново знайти «нульові точки» осей за допомогою кінцевих вимикачів, що відновлює точність позиціонування перед початком наступного друку.

Виклик цього сценарію відбувається всередині основного циклу автоматизації auto_loop_task. Система чекає досягнення цільової температури і лише тоді відправляє команду на виконання через асинхронний інтерфейс.

```
# Check if cooled enough
if bed_temp <= COOLING_TARGET_TEMP:
    log_message( level: "INFO",  message: f"Bed cooled to {bed_temp}°C. Ejecting part...",  category: "AutoLoop")
    auto_loop_state = AUTO_LOOP_STATE_EJECTING
    cooling_state["is_cooling"] = False

    # Execute clearing G-code
    with printer_lock:
        if printer:
            log_message( level: "INFO",  message: "Sending bed clear G-code",  category: "GCode")
            printer.gcode(CLEAR_BED_GCODE)

    # Wait for ejection to complete
    await asyncio.sleep(20)
```

Рисунок. 2.18 – Інтеграція процедури очищення в цикл автоматизації.

Використання await asyncio.sleep(20) після відправки команди гарантує, що система керування не намагатиметься розпочати новий друк, доки принтер фізично виконує маневри очищення та паркування.

Працездатність розробленого алгоритму була перевірена експериментально на серії тестових друків. На рис. 2.19 продемонстровано фізичний результат виконання скрипта очищення.

Рисунок. 2.19 – Робота системи механічного очищення: а) модель на робочій поверхні перед виконанням удару; б) стан платформи після виконання зштовхування.

Як видно з ілюстрацій, друкуюча голова успішно змістила готовий виріб за межі робочої зони, а подальша процедура паркування повернула осі у вихідне положення (0,0,0), підготувавши пристрій до наступного циклу виробництва без участі оператора.

2.6.5 Управління чергою завдань та реалізація прямого запуску друку

Оскільки 3D-принтер за своєю природою є пристроєм, що виконує одну задачу за раз для реалізації потокового виробництва було розроблено програмну надбудову — Менеджер черги.

Структура даних черги реалізована з використанням класу із даними PrintQueueItem (рис 2.20). Це дозволяє зберігати не лише шлях до файлу, а й метадані: необхідну кількість копій, кількість вже виготовлених деталей та поточний статус. Такий підхід дозволяє реалізувати сценарій «Друк серії», коли один файл друкується багаторазово без втручання оператора.

```python
@app.post("/api/queue/add")
async def add_to_queue(request: AddToQueueRequest):
    """Add a file to the print queue."""
    global print_queue

    if request.quantity < 1:
        raise HTTPException(status_code=400, detail="Quantity must be at least 1")

    # Extract filename from path
    file_name = request.file_path.split('/')[-1] if '/' in request.file_path else request.file_path

    # Create queue item
    item = PrintQueueItem(
        id=str(uuid.uuid4()),
        file_path=request.file_path.lstrip('/'),
        file_name=file_name,
        quantity=request.quantity,
        printed=0,
        status="PENDING",
        added_at=time.time()
    )

    with queue_lock:
        print_queue.append(item)
        print(f"[Queue] Added: {file_name} x{request.quantity}")

    return {
        "success": True,
        "item": asdict(item),
        "message": f"Added {file_name} x{request.quantity} to queue"
    }
```

Рисунок. 2.20 – Створення об’єкта PrintQueueItem та добавлення його в чергу.

Однак, найскладнішою технічною проблемою етапу реалізації стала відсутність у публічному API методів для запуску друку файлів, що знаходяться на локальній SD-карті, в режимі LAN. Стандартні методи бібліотеки bambulabs_api орієнтовані на хмарну взаємодію і формують посилання на AWS-сервера, які принтер не може обробити без доступу до інтернету.

Для вирішення цієї задачі було розроблено функцію custom_start_print, яка емулює мережеву поведінку оригінального ПЗ Bambu Studio (рис 2.21).

```python
def custom_start_print(  3 usages
    printer_instance,
    filename: str,
    plate_number: int | str,
    use_ams: bool = True,
) -> bool:
    if isinstance(plate_number, int):
        plate_location = f"Metadata/plate_{int(plate_number)}.gcode"
    else:
        plate_location = plate_number

    clean_filename = filename.lstrip('/')
    # CONSTRUCT URL - format: ftp://filename
    url = f"ftp://{clean_filename}"
    # Generate subtask name from filename (remove extensions)
    subtask_name = clean_filename.replace( old: '.gcode.3mf',  new: '').replace( old: '.3mf',  new: '')
    seq_id = str(int(time.time() * 1000) % 100000)
    payload = {
        "print": {
            "command": "project_file",
            "file": clean_filename,
            "param": f"Metadata/plate_{int(plate_number)}.gcode",
            "profile_id": "0",
            "project_id": "0",
            "sequence_id": seq_id,
            "subtask_name": subtask_name,
            "url": url,
            "use_ams": use_ams,
            "bed_leveling": True,
            "flow_cali": True,
        }
    }

    # Use the internal _client from the printer's mqtt_client to publish directly
    try:
        if not printer_instance.mqtt_client.is_connected():
            log_message( level: "ERROR",  message: "MQTT not connected",  category: "CustomPrint")
            return False

        topic = f"device/{printer_instance.serial}/request"
        # Access protected _client - hacky but necessary
        printer_instance.mqtt_client._client.publish(topic, json.dumps(payload))
        log_message( level: "INFO",  message: f"Published to {topic}",  category: "CustomPrint")
        log_message( level: "DEBUG",  message: f"Payload: {json.dumps(payload)}",  category: "CustomPrint")
        return True
```

Рисунок. 2.21 – Реалізація функції запуску друку для локального режиму.

Аналіз коду демонструє три критичні аспекти реалізації:

- Формування URL – так як принтер очікує специфічний формат ftp://ім'я_файлу. Якщо передати звичайний шлях або HTTP-посилання, прошивка принтера відхиляє запит.

- Використання унікального ідентифікатора (sequence_id) так як було виявлено, що принтер кешує ідентифікатори останніх команд. Якщо надіслати команду з тим самим ID у випадку повторного друку деталі, вона буде

проігнорована. Використання мітки часу time.time() гарантує унікальність кожного запиту.

- Пряма ін'єкція в MQTT, оскільки бібліотека-обгортка не підтримує такий формат пейлоаду, використано прямий доступ до внутрішнього клієнта (_client.publish) для відправки «сирого» JSON-пакета у топік request.

Інтеграція цієї функції з менеджером черги відбувається в автоматичному циклі. Коли система фіксує завершення попереднього циклу (див. п. 2.6.4), вона перевіряє лічильники черги:

```
# Check queue and start next print
next_item = None
with queue_lock:
    # 1. Update the counters for the item that just finished
    for item in print_queue:
        if item.status == "PRINTING":
            item.printed += 1
            if item.printed >= item.quantity:
                item.status = "DONE"
                log_message( level: "INFO",  message: f"Completed all copies of {item.file_name}",  category: "AutoLoop")
            # If not done, it remains PRINTING (ready for next loop)
            break

    # 2. Select next item
    # Priority: Finish current batch (PRINTING)
    for item in print_queue:
        if item.status == "PRINTING":
            next_item = item
            break

    # If no unfinished batch, find first PENDING
    if next_item is None:
        for item in print_queue:
            if item.status == "PENDING":
                next_item = item
                break

if next_item:
    log_message( level: "INFO",  message: f"Starting next print: {next_item.file_name}",  category: "AutoLoop")
    with queue_lock:
        next_item.status = "PRINTING"

    with printer_lock:
        if printer:
            log_message( level: "INFO",  message: f"Auto-starting: {next_item.file_path}",  category: "Print")
            success = custom_start_print(printer, next_item.file_path,  plate_number: 1, use_ams=False)
            if success:
                log_message( level: "INFO",  message: "Print started successfully",  category: "AutoLoop")
                # Wait for state to change from FINISH
                auto_loop_state = AUTO_LOOP_STATE_COMPLETED
```

Рисунок. 2.22 – Логіка пріоритезації завдань у черзі.

Алгоритм надає пріоритет завершенню поточної серії, і лише після друку всіх копій переходить до наступного файлу зі статусом PENDING. Це забезпечує логічну цілісність виробничого процесу.

2.6.6 Реалізація клієнтського інтерфейсу та підсистеми лайвстримінгу

Завершальним рівнем архітектури системи «CoreLoop» є клієнтський веб-інтерфейс, який забезпечує візуалізацію процесів та надання команд керування. Розробка велася з використанням бібліотеки React.js, а для обміну даними обрано протокол WebSocket, що дозволило відмовитися від класичної моделі «запит-відповідь» на користь подвійного керування івентами.

Організація миттєвого обміну даними через WebSocket-и

Замість постійного опитування сервера (Polling), яке створює зайве навантаження на мережу, реалізовано механізм «Server-Push». Сервер самостійно ініціює оновлення даних на клієнті у момент їх зміни. Для цього у реалізовано ендпоінт /ws, який підтримує постійне з'єднання з браузером.

```python
# --- WebSocket Endpoint ---
@app.websocket("/ws")
async def websocket_endpoint(websocket: WebSocket):
    """WebSocket endpoint for real-time printer state streaming."""
    await websocket.accept()

    async with websocket_lock:
        connected_websockets.append(websocket)

    log_message( level: "INFO",  message: f"Client connected. Total: {len(connected_websockets)}",  category: "WebSocket")

    try:
        # Send current state immediately
        if latest_state:
            await websocket.send_json({"type": "state", **latest_state})

        # Send log history
        with log_buffer_lock:
            logs_copy = list(log_buffer)
        await websocket.send_json({"type": "log_history", "logs": logs_copy})

        # Keep connection alive and handle pings
        while True:
            try:
                # Wait for any message with timeout
                data = await asyncio.wait_for(websocket.receive_text(), timeout=30.0)
                # Echo back as pong or just ignore
            except asyncio.TimeoutError:
                # Send ping to keep alive
                try:
                    await websocket.send_json({"ping": True})
                except Exception:
                    break
    except WebSocketDisconnect:
        pass
    finally:
        async with websocket_lock:
            if websocket in connected_websockets:
                connected_websockets.remove(websocket)
        log_message( level: "INFO",  message: f"Client disconnected. Total: {len(connected_websockets)}",  category: "WebSocket")
```

Рисунок. 2.23 – Реалізація WebSocket-сервера для синхронізації стану.

При першому підключенні клієнт миттєво отримує повний зліпок стану системи (latest_state) та історію логів. Це забезпечує відображення актуальної інформації без затримок інтерфейсу при перезавантаженні сторінки.

Реалізація підсистеми лайвстримінгу (MJPEG)

Для трансляції відео у браузер без використання важких протоколів було застосовано технологію MJPEG . Серверний ендпоінт /api/stream реалізовано як асинхронний генератор, що формує безперервний потік HTTP-відповіді типу multipart/x-mixed-replace.

Важливим технічним нюансом є отримання кадру (get_camera_frame) за межами блокування printer_lock. Це запобігає ситуації, коли повільна робота камери могла б заблокувати основний цикл опитування телеметрії.

```python
# --- Camera Stream Endpoint ---
/api/stream
@app.get("/api/stream")
async def camera_stream():
    """Stream camera frames as MJPEG."""

    def generate_frames():
        """Generator that yields JPEG frames."""
        while True:
            try:
                # Capture printer reference safely
                local_printer = None
                with printer_lock:
                    local_printer = printer

                # Perform camera operations OUTSIDE the lock
                # This prevents blocking the main polling loop during frame retrieval
                if local_printer and local_printer.camera_client_alive():
                    frame_b64 = local_printer.get_camera_frame()
                    if frame_b64:
                        frame_bytes = base64.b64decode(frame_b64)

                        yield (
                            b"--frame\r\n"
                            b"Content-Type: image/jpeg\r\n\r\n" + frame_bytes + b"\r\n"
                        )
                else:
                    # If camera not ready, wait a bit
                    time.sleep(1)

            except Exception as e:
                # Log only occasionally to avoid spam
                pass

            time.sleep(0.1)  # ~10 FPS

    return StreamingResponse(
        generate_frames(),
        media_type="multipart/x-mixed-replace; boundary=frame"
    )
```

Рисунок. 2.24 – Реалізація WebSocket-ендпоінта та реалізація лайвстримінгу із камери принтера.

Головна панель керування (Dashboard)

Інтерфейс «Dashboard» об'єднує всі потоки даних: відео, графіки температур, статус виконання завдання та елементи керування.

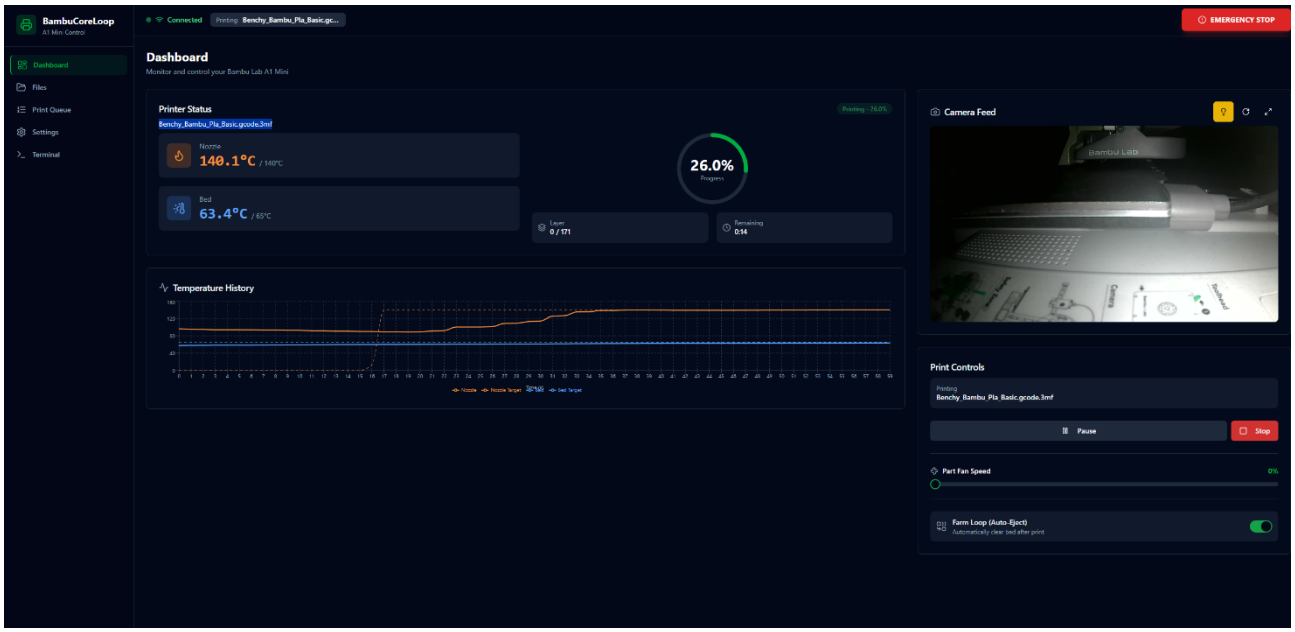

Рисунок. 2.25 – Головна панель керування системою.

Як видно з рисунку, інтерфейс надає оператору повну картину виробничого процесу, включаючи розрахунковий час завершення друку.

Файловий менеджер та інтеграція з чергою

Для керування бібліотекою моделей розроблено модуль «File Library». Він не лише візуалізує вміст SD-карти, але й дозволяє оператору завантажувати нові файли .3mf безпосередньо через веб-інтерфейс, використовуючи механізм «Drag & Drop».

На стороні сервера обробку вхідних файлів виконує ендпоінт /api/files/upload. Його реалізація вирішує дві важливі інженерні задачі: валідацію формату та санітизацію імен файлів.

```python
async def upload_file_endpoint(
    file: UploadFile = File(...),
    folder: Optional[str] = Query( default: None, description="Target folder (e.g., 'model')")
):
    """
    Upload a 3MF file to the printer.
    Optionally specify a target folder.
    """
    global printer

    # Validate file extension
    if not file.filename or not file.filename.lower().endswith('.3mf'):
        raise HTTPException(status_code=400, detail="Only .3mf files are allowed")
    safe_filename = re.sub( pattern: r'[^\w\-_\.]', repl: '_', file.filename.replace( old: ' ', new: '_'))
    # Ensure it still ends with .3mf
    if not safe_filename.lower().endswith('.3mf'):
        safe_filename = safe_filename + '.3mf'
    # Build full path with folder if specified
    if folder:
        folder = folder.strip('/')
        upload_path = f"{folder}/{safe_filename}"
    else:
        upload_path = safe_filename
    try:
        # Read file content
        content = await file.read()
        file_size = len(content)
        if file_size == 0:
            raise HTTPException(status_code=400, detail="File is empty")
        # Check printer connection first
        with printer_lock:
            if printer is None:
                raise HTTPException(status_code=503, detail="Printer not connected")
        print(f"[Files] Uploading to '{upload_path}' ({file_size} bytes)...")
        # Use separate FTP client to avoid breaking printer connection
        from bambulabs_api.ftp_client import PrinterFTPClient
        from constants.env import IP, ACCESS_CODE
        try:
            file_io = BytesIO(content)
            ftp_client = PrinterFTPClient(IP, ACCESS_CODE)
            result = ftp_client.upload_file(file_io, upload_path)
            print(f"[Files] Upload result: {result}")

            if result is None:
                raise HTTPException(status_code=500, detail="Upload returned no result - check FTP connection")
```

Рисунок. 2.26 – Реалізація завантаження файлів із санітизацією імен.

Як видно із коду, перед збереженням файлу виконується його перейменування за допомогою регулярного виразу re.sub(r'[^\w\-_\.]', '_', ...). Це критично важливо, оскільки вбудована операційна система принтера некоректно обробляє імена файлів із пробілами або спецсимволами при спробі запустити їх на друк через FTP-протокол.

Після валідації файл передається на принтер через тимчасове FTP-з'єднання, яке, як і у випадку читання списку, закривається одразу після завершення передачі даних.

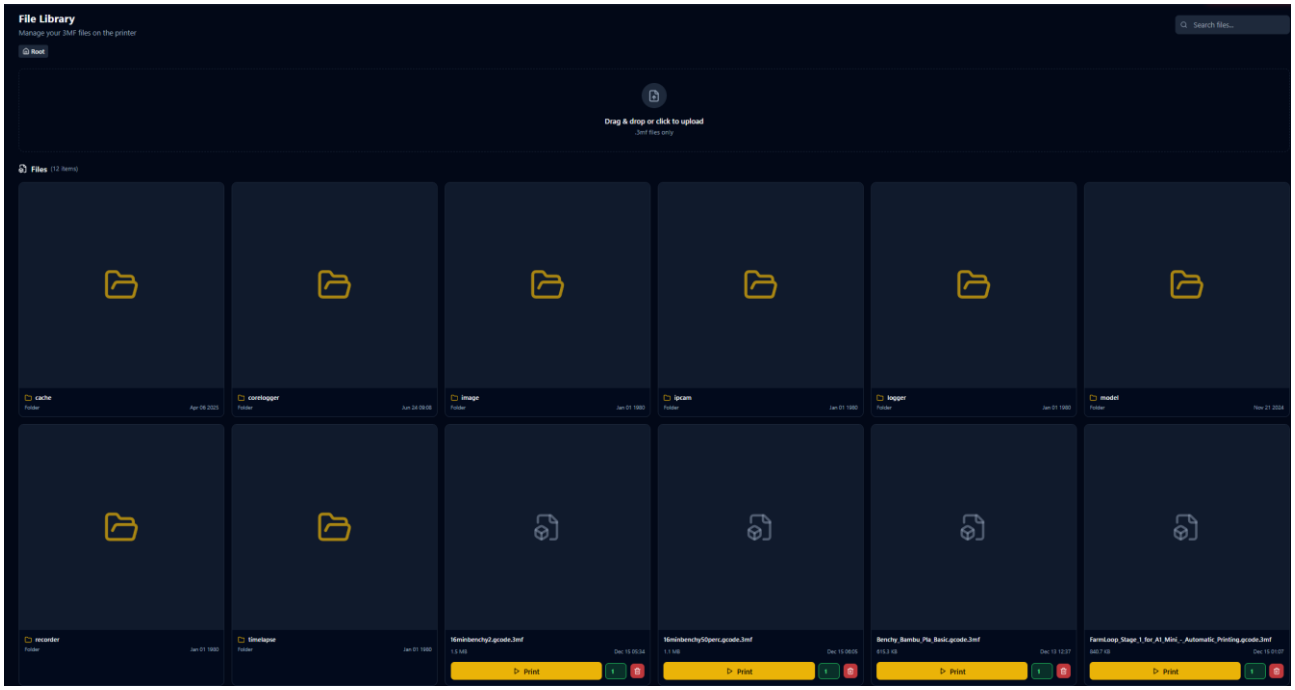

Рисунок. 2.27 – Інтерфейс файлового менеджера із можливістю завантажити файл та поставити на друк/в чергу модель.

Діагностика та безпечне керування механікою

Для контролю за роботою алгоритмів автоматизації в інтерфейс інтегровано системний термінал, який відображає логи сервера в реальному часі.

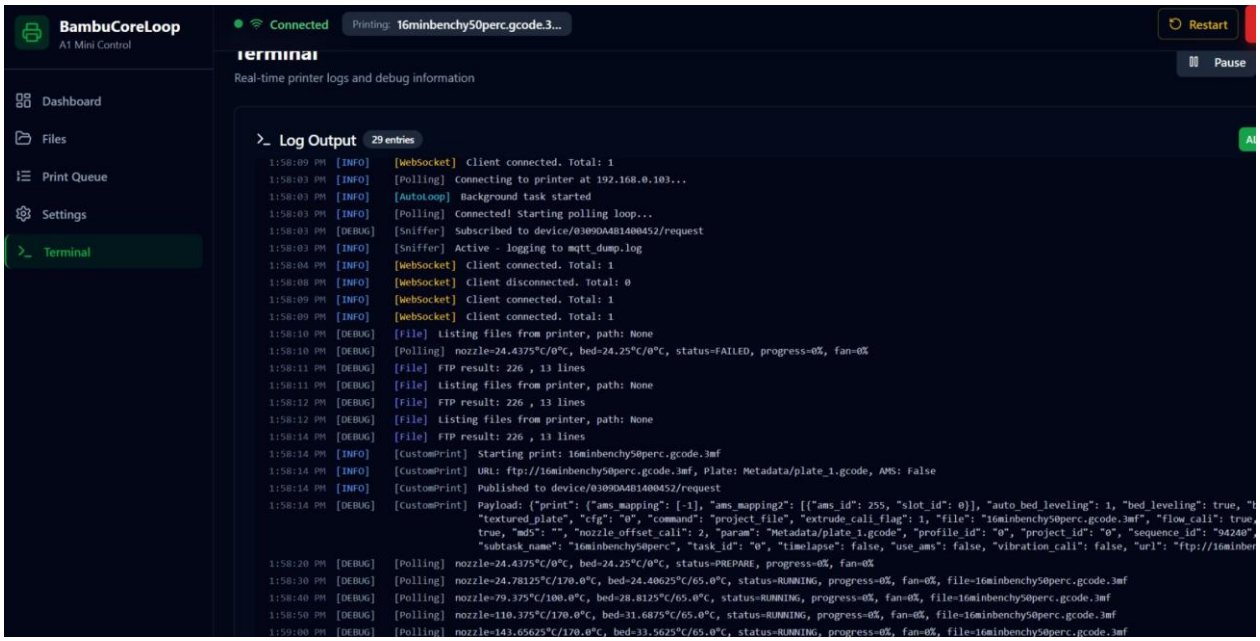

Рисунок. 2.28 – Системний термінал із логами роботи із можливістю фільтрування повідомлень.

Окремий модуль відповідає за ручне переміщення осей та можливістю задання температури для хотенду та друкувальної поверхні. Для запобігання пошкодженню обладнання, інтерфейс враховує програмні обмеження встановлені на сервері а також ліміти для досліджуваної моделі у вигляді 180x180x180мм.

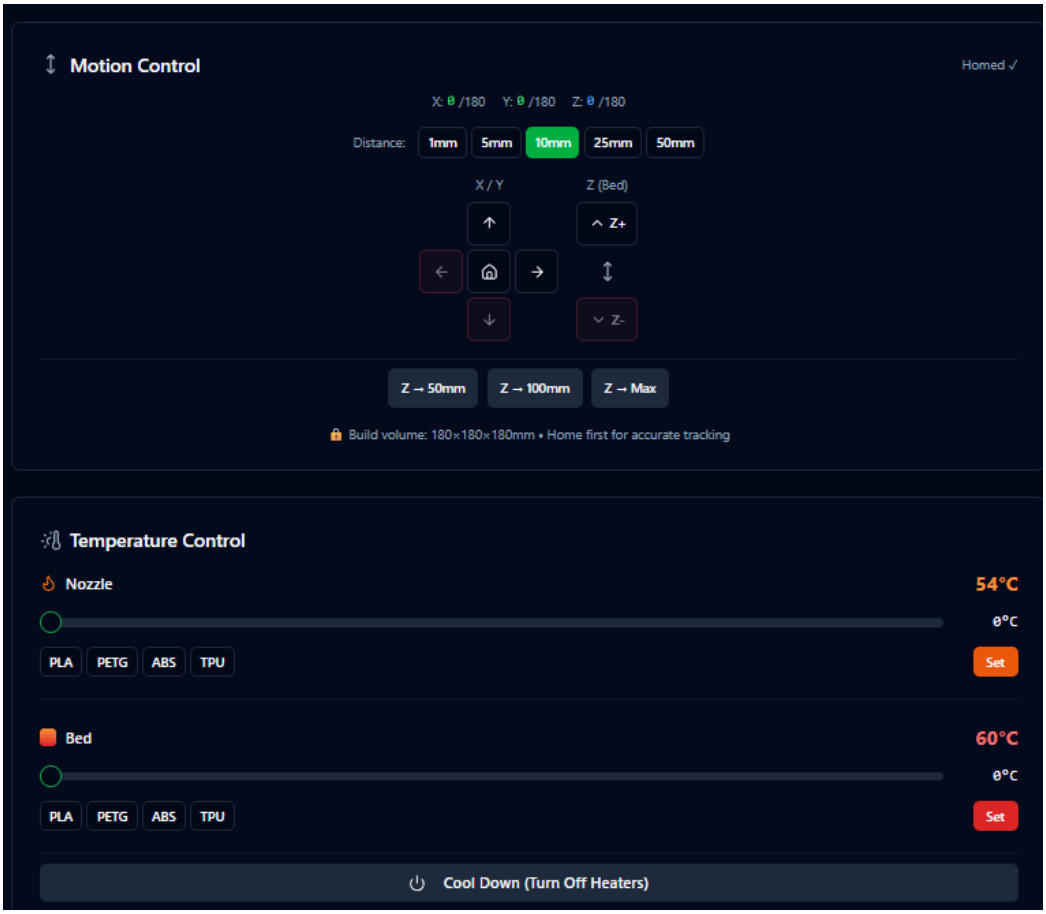

Рисунок. 2.29 – Інтерфейс ручного керування кінематикою принтера.

2.7 Організація тестування та налагодження програмного комплексу

Процес розробки системи «CoreLoop» супроводжувався етапами безперервного тестування окремих модулів та інтеграційним налагодженням. Враховуючи специфіку роботи з фізичним обладнанням, помилки в програмному коді могли призвести до механічного пошкодження компонентів 3D-принтера, тому було застосовано комплексну стратегію перевірки

працездатності, що охоплювала як віртуальні симуляції, так і натурні випробування.

Для верифікації серверної частини та REST API активно використовувався інструментарій Postman. Це дозволило перевірити коректність обробки HTTP-запитів на додавання файлів у чергу, керування температурними режимами та завантаження файлів через FTP-протокол ще до підключення реального фронтенду. Паралельно з цим, для незалежного моніторингу трафіку між сервером та принтером застосовувався MQTT Explorer. Цей інструмент надав можливість перевірити точність даних, перехоплених розробленим сніфером, та підтвердити, що критично важливі команди запуску друку надсилаються у валідному JSON-форматі, який відповідає вимогам прошивки пристрою. Налагодження клієнтського інтерфейсу та стабільності WebSocket-з'єднання здійснювалося за допомогою вбудованих засобів налагодження веб-браузера, що дозволило оптимізувати передачу відеопотоку MJPEG та усунути затримки візуалізації.

Найбільш відповідальним етапом стало тестування сценарію механічного очищення столу. Налагодження G-коду проводилося методом покрокового наближення для мінімізації ризиків. Спершу виконувалася «суха» симуляція без нагріву та без деталі на робочій поверхні, що дозволило переконатися у відповідності координат руху габаритам робочої зони принтера. Наступним кроком став запуск скрипта при кімнатній температурі з розміщеною тестовою деталлю для перевірки точності позиціонування штовхача по осі Z. Лише після успішного проходження цих етапів було проведено серію повноцінних автоматичних циклів, які підтвердили надійність алгоритму скидання деталі після охолодження.

2.8 Аналіз отриманих результатів та рекомендації щодо впровадження

В результаті виконання роботи було створено та успішно впроваджено програмно-апаратний комплекс, який ефективно вирішує проблему простою обладнання. Аналіз роботи системи продемонстрував значне підвищення автономності 3D-принтера, дозволяючи йому працювати в режимі 24/7 без

необхідності фізичного втручання оператора, допоки в системі подачі наявний витратний матеріал. Впровадження алгоритмів автоматизації суттєво скоротило час простою між циклами друку, зменшивши його з потенційних годин очікування оператора до приблизно 15 хвилин, необхідних для природного охолодження платформи. Окрім того, розроблений веб-інтерфейс розширив базові можливості обладнання, надавши користувачу функціонал прямого завантаження та запуску файлів без використання хмарних сервісів виробника.

На основі проведених експериментів було сформовано ряд технічних рекомендацій для успішної експлуатації системи. Ключовою умовою надійної роботи механізму автоматичного скидання є використання текстурованих PEI-пластини. Експериментально доведено, що саме цей тип покриття забезпечує необхідну зміну адгезивних властивостей: надійне утримання деталі при високих температурах та її легке відшарування при охолодженні. Використання гладких поверхонь або додаткових адгезивів може перешкодити успішному виконанню процедури очищення. Також було встановлено, що температурний поріг у 40°C є оптимальним тригером для безпечного видалення виробів із пластику PLA, тоді як для матеріалів з вищою температурою склування, таких як PETG або ABS, цей параметр може потребувати програмного коригування. З метою безпеки також рекомендовано забезпечити вільний простір перед принтером для встановлення накопичувального кошика, щоб уникнути застрягання готових виробів на виході з робочої зони.

ВИСНОВКИ

У магістерській кваліфікаційній роботі вирішено актуальне науково-прикладне завдання автоматизації циклу дрібносерійного виробництва на базі настільних FDM 3D-принтерів із закритим програмно-апаратним середовищем. Шляхом створення програмно-апаратного комплексу «CoreLoop» досягнуто мети роботи — підвищення рівня автономності обладнання та мінімізації часу простою:

- Проведено аналіз інфраструктури Bambu Lab та встановлено наявність штучних обмежень функціоналу в локальному режимі. Найбільш ефективним шляхом інтеграції визначено використання гібридної схеми зв'язку через протоколи MQTT та FTP;
- Здійснено реверс-інжиніринг протоколів та розроблено програмний модуль-сніфер, який декодує структуру закритих пакетів керування. Це дозволило реалізувати функцію запуску друку файлів з локального сховища без використання хмарних сервісів виробника, що є унікальною особливістю розробленої системи;
- Створено архітектуру веб-менеджера на базі мікрофреймворку FastAPI та бібліотеки React.js. Застосування асинхронного підходу та технології WebSocket забезпечило реактивність інтерфейсу та можливість моніторингу телеметрії в реальному часі;
- Розроблено та впроваджено сценарій на мові G-code, який використовує друкувальну голову як механічний маніпулятор. Алгоритм базується на термодинамічних показниках: система автоматично блокує механічні дії до охолодження платформи до 40°C, що гарантує безпечне відшаровування деталі без пошкодження обладнання;
- Спроєктовано та виготовлено апаратні модифікації, зокрема насадку-штовхач для екструдера та похилий стенд. Ці елементи фізично забезпечують видалення готової продукції з робочої зони під дією сили тяжіння;

- Впроваджено систему керування чергою з логікою тиражування, яка дозволяє задавати кількість копій для однієї моделі. Програма автоматично відстежує статус виконання та ініціює наступний цикл друку одразу після завершення процедури очищення.

Результати практичних випробувань підтвердили працездатність комплексу. Час простою між циклами друку скоротився до технічно необхідного мінімуму (часу охолодження), що дозволяє використовувати обладнання в режимі 24/7 без постійної присутності оператора. Система рекомендована для впровадження у приватних принт-фермах та лабораторіях прототипування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Етапи FDM 3D-друку. *3DWAY – 3D-принтери, 3D-сканери та комплектуючі.* URL: https://3dwayshop.com.ua/etapy-3d-druku/.
2. Investigation and Optimization of Effects of 3D Printer Process Parameters on Performance Parameters. MDPI. URL: https://www.mdpi.com/1996-1944/16/9/3392.
3. Illustration of Cartesian printer. (Cartesian3D, 2019). https://www.researchgate.net/. URL: https://www.researchgate.net/figure/llustration-of-Cartesian-printer-Cartesian3D-2019_fig4_350051043.
4. Additive Manufacturing Market (2024 - 2030). grandviewresearch.com. URL: https://www.grandviewresearch.com/industry-analysis/additive-manufacturing-market.
5. A1 mini. *Bambu Lab Wiki*. URL: https://wiki.bambulab.com/en/a1-mini/
6. How to enable LAN Mode on Bambu Lab printers. *Bambu Lab Wiki*. URL: https://wiki.bambulab.com/en/knowledge-sharing/enable-lan-mode.
7. Учасники проектів Вікімедіа. MQTT – Вікіпедія. *Вікіпедія*. URL: http://uk.wikipedia.org/wiki/MQTT.
8. Contributors to Wikimedia projects. WebSocket - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/WebSocket.
9. Explaining the "Auth System" in laymans terms. *Bambu Lab Community Forum*. URL: https://forum.bambulab.com/t/explaining-the-auth-system-in-laymans-terms/136629/68?page=4.
10. FTP and MQTT password after update. *Bambu Lab Community Forum*. URL: https://forum.bambulab.com/t/ftp-and-mqtt-password-after-update/5200/12.
11. How to publish MQTT messages. *https://www.reddit.com/*. URL: https://www.reddit.com/r/BambuLab/comments/13hsoq0/how_to_publish_mqtt_messages/.
12. Software Bambu Studio | Bambu Lab. *bambulab.com*. URL: https://bambulab.com/en/download/studio.

13. Bambu Farm Manager Quick Start Guide. *Bambu Lab Wiki*. URL: https://wiki.bambulab.com/en/software/bambu-farm-manager.

14. OctoPrint-Dashboard. *OctoPrint Plugin Repository*. URL: https://plugins.octoprint.org/plugins/dashboard/.

15. 3D Printer Auto Bed Clearing - (Bambu A1 Example). *Infinity Flow 3D*. URL: https://infinityflow3d.com/blogs/3d-printer-automation/3d-printer-auto-bed-clearing-bambu-a1-example.

16. Auto Ejection. *3DQue - Automation for 3D Printers and Print Farms*. URL: https://www.3dque.com/products/auto-ejection.

17. FarmLoop Stage 1 for A1 Mini - Automatic Printing by 3D Farmers MakerWorld: Download Free 3D Models. *MakerWorld*. URL: https://makerworld.com/en/models/1255869-farmloop-stage-1-for-a1-mini-automatic-printing.

18. Ітеративна модель (iterative model). *https://qalight.ua*. URL: https://qalight.ua/baza-znaniy/iterativna-model-iterative-model/.

19. FastAPI. *FastAPI*. URL: https://fastapi.tiangolo.com/.

20. Asyncio Asynchronous I/O. *Python documentation*. URL: https://docs.python.org/uk/3/library/asyncio.html.

21. Eclipse Paho | The Eclipse Foundation. *Projects Gateway | The Eclipse Foundation*. URL: https://eclipse.dev/paho/.

22. bambulabs-api. *PyPI*. URL: https://pypi.org/project/bambulabs-api/.

23. React. *React*. URL: https://react.dev/.

24. Introduction - Zustand. *Poimandres documentation*. URL: https://zustand.docs.pmnd.rs/getting-started/introduction.

25. Tailwind CSS - Rapidly build modern websites without ever leaving your HTML. *Tailwind CSS - Rapidly build modern websites without ever leaving your HTML*. URL: https://tailwindcss.com/.

26. JetBrains. PyCharm: the Python IDE for Professional Developers by JetBrains. JetBrains. URL: https://www.jetbrains.com/pycharm.

27. Microsoft. Visual Studio Code - The open source AI code editor. *Visual Studio Code - The open source AI code editor*. URL: https://code.visualstudio.com/.

28. Image file type and format guide - Media | MDN. *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Image_types.

29. Van Rossum G., Drake F. L. Python 3 Reference Manual. Scotts Valley : CreateSpace, 2009.

30. Nordquist T. MQTT Explorer. *MQTT Explorer*. URL: http://mqtt-explorer.com/.

31. Перевірка команди MQTT не вдалася. Як це виправити?. *Reddit.com*. URL: https://www.reddit.com/r/BambuLab/comments/1llmpkt/mqtt_command_verification_failed_how_to_fix/?tl=uk.

32. Bambu Lab X1 X1C MQTT. *Home Assistant Community*. URL: https://community.home-assistant.io/t/bambu-lab-x1-x1c-mqtt/489510/754.

33. Drogon-code. Cant start print A1mini · Issue #156 · BambuTools/bambulabs_api. *GitHub*.
URL: https://github.com/BambuTools/bambulabs_api/issues/156.

34. MQTT - BambuLab A1 Mini - 3D-Druck Forum. *3D-Druck Forum*. URL: https://forum.drucktipps3d.de/forum/thread/33551-mqtt-bambulab-a1-mini/.

35. Scheduled/Automated Print Job Start · greghesp/ha-bambulab · Discussion #628. *GitHub*. URL: https://github.com/greghesp/ha-bambulab/discussions/628.

36. MQTT with P1S. *Bambu Lab Community Forum*. URL: https://forum.bambulab.com/t/mqtt-with-p1s/102463.

37. FTP for ipcam videos (or download from Bambu Studio). *Bambu Lab Community Forum*. URL: https://forum.bambulab.com/t/ftp-for-ipcam-videos-or-download-from-bambu-studio/172734.

38. We can now connect to FTP on the P1 and A1 Series. *Bambu Lab Community Forum*. URL: https://forum.bambulab.com/t/we-can-now-connect-to-ftp-on-the-p1-and-a1-series/6464.

ДОДАТКИ

Додаток А

ТЕХНІЧНЕ ЗАВДАННЯ

ВСТУП

Найменування програмного продукту Програмно-апаратний комплекс автоматизації 3D-друку «CoreLoop».

Призначення застосування – Система призначена для власників 3D-принтерів Bambu Lab (модель A1 Mini), операторів «принт-ферм» та мейкерів, які прагнуть організувати безперервний цикл дрібносерійного виробництва. Продукт дозволяє автоматизувати запуск завдань з черги та очищення робочої поверхні без фізичного втручання людини.

ПРИЗНАЧЕННЯ І ПІДСТАВИ ДЛЯ РОЗРОБКИ

Призначенням даної роботи є створення спеціалізованого програмно-апаратного комплексу «CoreLoop», спрямованого на повну автоматизацію процесів дрібносерійного виробництва на базі FDM 3D-принтерів із закритою архітектурою. Підставою для розробки виступає необхідність мінімізації виробничих простоїв, зумовлених потребою у постійному фізичному втручанні оператора для обслуговування обладнання між циклами друку, а також обмеженість функціоналу штатного програмного забезпечення виробника, яке не підтримує автоматичний запуск черги завдань та механічне очищення платформи у локальному режимі роботи. Реалізація системи дозволить перетворити настільний принтер на автономну виробничу одиницю, здатну функціонувати в режимі 24/7 без хмарних сервісів.

ВИМОГИ ДО ПРОГРАМИ

Вимоги до функціональних характеристик:

- відображення телеметрії в реальному часі (температура сопла/столу, прогрес, залишковий час, статус вентиляторів) з оновленням через WebSocket;

- трансляція відеопотоку з камери принтера у веб-інтерфейс (MJPEG) з мінімальною затримкою;

- можливість додавання файлів (.gcode.3mf), зміни їх порядку, видалення та налаштування кількості копій;

- реалізація замкненого циклу: друк → охолодження до безпечної температури → механічне очищення (G-code) → запуск наступного файлу;

- завантаження файлів на SD-карту принтера через веб-інтерфейс (drag-and-drop) з автоматичною валідацією імен;

- можливість ручного переміщення осей (X, Y, Z), паркування та керування нагрівом.

Вимоги до надійності:

- автоматичне відновлення з'єднання з принтером у разі обриву мережі;

- блокування механічних дій очищення, якщо температура столу перевищує 40°C;

- коректна обробка аварійних станів принтера.

Умови експлуатації:

- програма повинна працювати на операційних системах Windows, а також може бути портовано на платформи Linux, macOS;

- програма повинна бути легкою у використанні, з інтуїтивно зрозумілим інтерфейсом;

Вимоги до технічних засобів та середовища:

- мінімальні вимоги до процесора: Intel Core i3 або еквівалентний;

- мінімальні вимоги до оперативної пам'яті: 4 ГБ;

- серверна частина повинна підтримувати Python 3.10+, FastAPI, Uvicorn;

- сучасний веб-браузер (Chrome, Firefox, Safari) з підтримкою JavaScript ES6+;

- 3D-принтер Bambu Lab A1 Mini, встановлена модифікація насадки-штовхача на екструдер, похилий стенд;

- локальна мережа (Wi-Fi 2.4 GHz) із доступом до принтера по порту 8883 (MQTT) та 990 (FTPS).

Додаток Б

Інструкція користувача

Для початку роботи з програмно-апаратним комплексом «CoreLoop» необхідно попередньо підготувати апаратну частину. Користувач повинен переконатися, що 3D-принтер Bambu Lab A1 Mini встановлено на спеціалізований похилий стенд, а на блок екструдера надійно змонтовано роздруковану насадку-штовхач. Перед запуском також варто розмістити перед принтером накопичувальний кошик для готових виробів та перевірити, що зона викиду деталей вільна від сторонніх предметів.

Наступним кроком є налаштування мережевого з’єднання. На сенсорному екрані принтера необхідно перейти в налаштування та активувати режим локальної мережі (LAN Mode). Після активації на екрані з’явиться IP-адреса та код доступу, які потрібно зберегти для подальшого внесення у конфігураційний файл сервера. Важливо, щоб керуючий комп'ютер та принтер знаходилися в одній Wi-Fi мережі.

Запуск програмного забезпечення виконується через командний рядок. Користувач відкриває термінал у папці з проєктом та вводить команду для ініціалізації сервера: **uvicorn main:app --host 0.0.0.0 --port 8000**. Після успішного старту бекенду доступ до графічного інтерфейсу здійснюється через веб-браузер за адресою локального хоста (порт 3000). Про успішне підключення сигналізуватиме статус «Connected» зеленого кольору у верхньому лівому куті панелі керування та поява зображення з камери.

Робота з чергою друку починається у вкладці «Files». Користувач завантажує підготовлені файли формату .3mf шляхом перетягування їх у відповідну зону інтерфейсу. Після завантаження файлу його можна додати до черги друку, вказавши необхідну кількість копій у спливаючому вікні. Система автоматично сформує список завдань у вкладці «Print Queue», де можна змінювати їх пріоритетність.

Для експлуатації в автоматичному режимі необхідно перевести перемикач «Auto-Loop» у положення активності та натиснути кнопку запуску. Система

самостійно завантажить файл, виконає друк та перейде в режим охолодження. Користувачеві забороняється торкатися робочої поверхні до завершення цього етапу. Як тільки температура столу опуститься нижче безпечного порогу в 40°C, принтер автоматично виконає процедуру механічного очищення та розпочне друк наступного виробу з черги.

У разі виникнення аварійних ситуацій, таких як збій друку або застрягання пластику, передбачено кнопку екстреної зупинки «EMERGENCY STOP» у верхньому правому куті екрана, яка миттєво відключає нагрівачі та двигуни. Для відновлення роботи та ручного звільнення робочої зони слід скористатися меню керування рухом у налаштуваннях системи.

АНОТАЦІЯ

Маркітан В.О – Розробка та дослідження мережевого менеджера для дистанційного мережевого 3D-друку – Рукопис.

Кваліфікаційна робота за спеціальністю 122 Комп'ютерні науки. - Волинський національний університет імені Лесі Українки, Луцьк. - 2025р.

Магістерська кваліфікаційна робота присвячена вирішенню актуального науково-прикладного завдання автоматизації дрібносерійного виробництва на базі настільних FDM 3D-принтерів із закритою архітектурою на прикладі Bambu Lab A1 Mini.

У роботі проведено комплексний аналіз протоколів взаємодії обладнання в локальній мережі та здійснено реверс-інжиніринг пакетів керування MQTT. На основі отриманих даних спроєктовано та розроблено програмно-апаратний комплекс «CoreLoop», який складається з серверної частини, реалізованої мовою Python на базі фреймворку FastAPI, клієнтського веб-інтерфейсу, використовуючи React.js та апаратних модифікацій принтера (насадка-штовхач, похилий стенд).

Ключовим результатом роботи є реалізація унікального алгоритму керування динамічною чергою друку, що дозволяє ініціювати запуск файлів з локального сховища без використання хмарних сервісів виробника. Впроваджено систему автоматичного очищення робочої зони на основі спеціалізованого G-коду, який активується за умовою досягнення безпечної температури платформи. Експериментальні дослідження підтвердили надійність системи та її здатність забезпечувати безперервний виробничий цикл у режимі 24/7, мінімізуючи час простою обладнання та необхідність втручання оператора.

Ключові слова: 3D-друк, автоматизація, IoT, MQTT, реверс-інжиніринг, G-code, веб-інтерфейс, Bambu Lab, скінченний автомат.

ANNOTATION

Markitan V.O – Development and research of a network manager for remote network 3D printing – Manuscript.

Qualification work for the specialty 122 Computer Sciences. – Lesya Ukrainka Volyn National University, Lutsk. – 2025.

The master's thesis is devoted to solving the urgent scientific and applied problem of automating small-batch production based on desktop FDM 3D printers with a closed architecture, especially for Bambu Lab A1 Mini.

The thesis conducts a comprehensive analysis of equipment interaction protocols within a local network and performs reverse engineering of MQTT control packets. Based on the obtained data, the «CoreLoop» hardware-software complex was designed and developed. The system consists of a backend server implemented in Python using the FastAPI framework, a client web interface via React.js, and hardware modifications to the printer.

The key result of the work is the implementation of a unique dynamic print queue control algorithm, which allows initiating file launches from local storage without relying on the manufacturer's cloud services. An automatic workspace clearing system was introduced based on a specialized G-code script, which triggers only after the platform reaches a safe temperature threshold. Experimental testing confirmed the system's reliability and its capability to maintain a continuous 24/7 production cycle, significantly minimizing equipment downtime and the need for operator intervention.

Keywords: 3D printing, automation, IoT, MQTT, reverse engineering, G-code, web interface, state machine.