Національний університет «Полтавська політехніка імені Юрія Кондратюка»

Навчально-науковий інститут інформаційних технологій і роботехніки

Кафедра автоматики, електроніки та телекомунікацій

Пояснювальна записка

до кваліфікаційної роботи
магістра
(ступінь вищої освіти)

на тему Розробка програмного забезпечення післяпроцесної обробки об'єктів адитивного виробництва

Виконав: студент 6 курсу, групи 601-ТТ
Спеціальність
172 Телекомунікації та радіотехніка
(шифр і назва напряму підготовки, спеціальності)

Зуб С.В.
(прізвище та ініціали)

Керівник Сокол Г.В.
(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

м. Полтава – 2022

Національний університет «Полтавська політехніка імені Юрія Кондратюка»
Інститут Навчально-науковий інститут інформаційних технологій і робототехніки
Кафедра Автоматики, електроніки та телекомунікацій
Ступінь вищої освіти Магістр
Спеціальність 172 «Телекомунікації та радіотехніка»

ЗАТВЕРДЖУЮ

Завідувач кафедри автоматики, електроніки та телекомунікацій

_____ О.В. Шефер
“ ___ ” _____ 2022 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Зубу Сергію Васильовичу

1. Тема проекту (роботи) **«Розробка програмного забезпечення післяпроцесної обробки об’єктів адитивного виробництва»**
керівник проекту (роботи) Сокол Галина Вікторівна, к.т.н., доцент.
затверджена наказом вищого навчального закладу від “12 ” 08 2022 року №544 фа

2. Строк подання студентом проекту (роботи) 07.12.2022 р.

3. Вихідні дані до проекту (роботи) FDM, PETG, G-code, STL, ESP32x, верстат S1, технічне завдання на розробку, наукові публікації та інтернет-джерела з питань розробки програмного забезпечення та мобільного додатку.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1) Адитивне виробництво, його післяпроцесна обробка та огляд і аналіз предметної області (загальні відомості про адитивне виробництво і його післяпроцесну обробку, основні відомості по esp32)
2) Проектування програмного забезпечення (аналіз та вибір мов програмування для серверної частини, проектування серверної частини, проектування клієнтської частини)
3) Практична частина розробки програмного забезпечення.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових плакатів):
   1) Адитивне виробництво
   2) Метод пошарового наплавлення (Fused Deposition Modeling, FDM)
   3) Модифікація апаратного сегменту
   4) Модифікація цифрового сегменту
   5) Аналіз та вибір мов програмування
   6) Аналіз та проектування серверної частини ESP32
   7) Візуалізація веб-інтерфейсу
   8) Візуалізація інтерфейсу мобільного додатку
   9) Візуалізація інтерфейсу програмного забезпечення ОС Windows

6. Дата видачі завдання 01.09.2022 р.

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів магістерської роботи	Термін виконання етапів роботи		
1	Аналіз напрямків використання адитивного виробництва в інтересах телекомунікацій.	13.09.22		15%
2	Аналітичний огляд з предметної області. Постановка задачі досліджень.	27.09.22	I	30%
3	Проектна частина.	10.10.22		40%
4	Розробка серверної частини і програмування МК.	17.10.22		50 %
5	Розробка програмного забезпечення і мобільного додатку.	25.10.22	II	60%
6	Оформлення магістерської роботи.	07.11.22		70%
7	Отримання допуску до захисту та подача роботи в ДЕК.	07.12.22	III	100%

Магістрант _____ Зуб С. В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Сокол Г. В.
(підпис) (прізвище та ініціали)

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ

FDM – Fused Deposition Modeling

ASTM – American Society for Testing and Materials

ISO – International Organization for Standardization

CJP – ColorJet Printing

MJP – MultiJet Printing

SLA – Laser Stereolithography

SLS – Selective Laser Sintering

SLM – Selective Laser Melting

DMP – Direct Metal Printing

CAD – Computer Aided Design

LTCC – Low Temperature Co-fired Ceramics

STL – STereoLithography

ПЗ – Програмне забезпечення

PETG – Поліетилентерефталат-Гликоль

PET – Поліетилентерефталат

ABS – Акрилонитрил бутадієн стирол

PLA – Полілактид

МК – Мікропроцесор

ШІМ – Широтно-імпульсна модуляція

TSMC – Taiwan Semiconductor Manufacturing Company

ЧПУ – Числове програмне управління

RTOS – Real-time operating system

MIMO – Multiple Input Multiple Output

IoT – Internet of Things

IIoT – Industrial Internet of Things

МОР – Мастильно-охолоджуючі рідини

HTTP – HyperText Transfer Protocol

URI – Uniform Resource Identifier

HTML – HyperText Markup Language

ОС	–	операційна система
IDE	–	Integrated Development Environment
API	–	Application Programming Interface
JWT	–	Json Web Token
ООП	–	Об’єктно орієнтоване програмування
DTO	–	Data Transfer Object
MVVM	–	Model View ViewModel
МК	–	Мікроконтроллер
STA	–	Station mode
AP	–	Access point
ПК	–	Персональний комп’ютер

ВСТУП

Як відомо, застосування 3D-друку в електроніці можна розділити на два напрямки: 3D-друк власне електронних компонентів і виробництво допоміжної оснастки для електроніки. Значні переваги технологій адитивного виробництва перед традиційними забезпечують: виготовлення недорогих пристроїв при малих серіях, створенні макетів та дослідних зразків; значне спрощення зборки, зменшення маси, кількості з’єднань та ін., а також підвищує надійність системи в цілому. Одним з поширених методів адитивного виробництва є FDM, що забезпечує економічне ефективність процесу проектування до початку масового виробництва.

Однак, під час такого 3D-друку модель формується шар за шаром, тобто необхідний попередній шар для формування нового. В залежності від складності 3D-моделі та специфіки технологій 3D-друку є необхідність використовувати спеціальні супорти, що безпосередньо впливають на якість кінцевих виробів (разом з нульовим шаром вони підлягають видаленню). Все це може призвести до нерівності поверхні друкованої 3D-моделі.

Метою роботи є підвищення ефективності технологічного процесу адитивного виробництва за рахунок модернізації, удосконалення і доповнення післяпроцесної обробки. Основна задача досліджень полягає в розробці програмного забезпечення і веб-інтерфейсу для прискорення та спрощення використання методів післяпроцесної обробки адитивного виробництва.

В роботі запропоновано розробку ПЗ МК ESP32 та ПЗ для клієнтської частини під керування ОС Windows та Android. При цьому необхідність роботи ПЗ МК як у автономному режимі, так і у ручному з під системи Windows та Android. Ці ПЗ розробляються для уніфікації роботи станка з ЧПУ не лише для післяпроцесної обробки адитивного виробництва, а і для самого адитивного виробництва і не тільки. Для обґрунтування висунутих в роботі положень представлено результати розроблених пропозицій.

Все це свідчить про актуальність теми дипломної роботи та розробка

пропозицій щодо використання даних ПЗ для фінішної обробки об’єктів адитивного виробництва лазерних оптичних модулів та сучасних технологій інфокомунікацій для їх управління.

Кінцевою метою дослідження є підвищення ефективності технологічного процесу післяпроцесної обробки адитивного виробництва за рахунок створення та удосконалення програмного забезпечення мікроконтролера та програмного забезпечення для популярних операційних систем.

Для досягнення зазначеної мети необхідно вирішити наступні задачі.

1. Провести аналіз роботи з МК ESP32 та реалізації сервер-клієнтської частини.
2. Спроектувати серверну частину МК.
3. Спроектувати WEB-інтерфейс для МК.
4. Визначитися з концепцією розробки ПЗ.
5. Розробити ПЗ для ОС Windows.
6. Розробити додаток для ОС Android.
7. Виконати технічне обґрунтування прийнятих рішень.
8. Розглянути перспективи подальшого розвитку.

Об’єкт дослідження – процес післяпроцесної обробки адитивного виробництва.

Предмет дослідження – програма мікроконтролера, програма для розробки програмних забезпечень.

Метод дослідження – пошуковий, інформаційний, аналітичний та експериментальний.

Структура дипломної роботи логічно пов’язана з задачами досліджень і містить перелік скорочень, умовних позначень, вступ, три розділи основної частини, висновки, список використаних джерел, додаток.

РОЗДІЛ 1.
АДИТИВНЕ ВИРОБНИЦТВО, ЙОГО ПІСЛЯПРОЦЕСНА ОБРОБКА ТА ОГЛЯД і АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Загальні відомості про адитивне виробництво і метод пошарового наплавлення (FDM)

Як всім відомо, для виготовлення окремих електронних, і не тільки, компонентів все частіше засовуються технології адитивного виробництва , що мають значні переваги перед традиційними.

Адитивне виробництво - узагальнена назва технологій, які передбачають виготовлення виробу за даними тривимірної цифрової моделі методом пошарового додавання (англ. Add - додавати, звідси і назва) матеріалів, проте словосполучення «3D-друк» більш поширене і фактично стало стандартом. У адитивних процесах об’єкт створюється шляхом укладання безперервних шарів матеріалу до завершення всього проекту. Кожен із цих шарів можна розглядати як вузько зрізану горизонтальну ділянку даного об’єкта. Загальна схема адитивного виробництва представлена на рис. 1.1.

Рис. 1.1. Схема адитивного виробництва

До адитивного виробництва відносяться 7 відмінних процесів. Вироби створюються шляхом: екструзії; розбризкування (струменеві напилення); УФ-затвердіння; ламінування; сплаву матеріалів.

Базові технології, які застосовуються при створенні виробів на адитивних установках:

– ColorJet Printing (CJP) – це технологія в основі якої лежить пошарове склеювання і фарбування композитного порошку на основі гіпсу або пластику;

– MultiJet Printing (MJP) – технологія заснована на багатоструменевому моделюванні за допомогою фотополімерного або воскового матеріалу ;

– Laser Stereolithography (SLA) – стереолітографія, використовуючи джерело світла — лазер або проектор — для затвердіння рідкої смоли в затверділий пластик.;

– Fused deposition modeling (FDM) - метод нанесення розплавленого матеріалу за заздалегідь встановленим алгоритмом, шар за шаром. Використовувані матеріали є термопластичні полімери і мають форму нитки.

– Selective Laser Sintering (SLS) – селективне лазерне спікання дрібнодисперсного порошкового (зазвичай, металевого) матеріалу з допомогою лазера. до утворення фізичного об’єкта за заданою CAD-моделю;

– Selective Laser Melting / Direct Metal Printing (SLM/DMP) – розроблена для використання лазера з високою щільністю потужності для плавлення і сплавлення металевих порошків разом [2] .

В цілому, 3D-друк дозволяє: виробляти недорогі по собі вартості пристрої як при малій кількості, так і при створенні макетів та дослідних зразків; відчутно спрощує перехід від класичної планарної компоновки електронних пристроїв до об’ємної; спростити процес зборки, зменшити габарити та масц, кількість зайвих з’єднань, та також збільшує вцілому надійність системи. До того ж, можна використовувати різні матеріали для виготовлення компонентів, наприклад: струмопровідні чорнила та полімери, металеві порошки, синтезовані діелектрики та ін.

Моделювання методом пошарового наплавлення Fused Deposition Modeling (FDM) полягає в пошаровому накладанню розплавленого пластика, який подається через сопло екструдера, який переміщається по програмно заданій траєкторії. З мірою подачі розплавленого матеріалу він охолоджується і застигає, формуючи шар, який буде основою для наступного шару. Весь цей процес повторюється шар за шаром до повного закінчення друку моделі [3].

Під час FDM-друку пластик лягає на нижній шар і навколо нього. Кожен шар формується з певним віддаленням від попереднього. Завдяки такому принципу модель (рис. 1.2) може друкуватися з кутами до 45^0. Але якщо кут нахилу більше 45^0, то обов'язково потрібно використовувати супорти. Видимий приклад для використання супортів під час FDM-друку наведений на рис. 1.8. Необхідність використання супортів можна пояснити на прикладі літери Y, H, і T (рис 1.4). Верхня частина букви Y друкується без проблем. Не перешкоджає навіть те, що верхня частина літери спрямована вгору і в сторону, її кут менше 45^0, то ж додаткові супорт структури не потрібні. Літера H – складніший випадок, але якщо центральна частина літери тонша 5 мм, то можна спробувати надрукувати її без супортів. Але якщо більше 5, додаткові супорти необхідні для уникнення провисання. Відповідно для літери T – супорти однозначно необхідні.

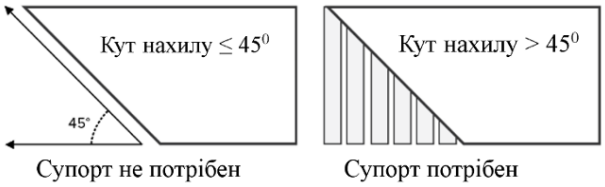
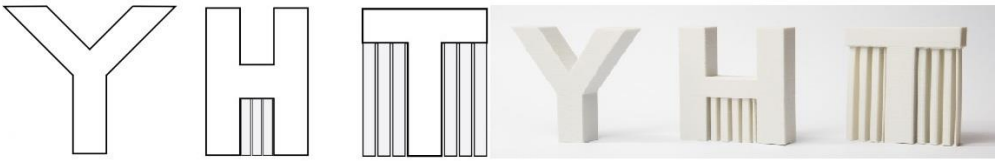


Рис. 1.2. Правило використання супортів

а) б)

Рис. 1.3. Супорти для FDM: а) – «акордеон»; б) – «дерево»



а) б)

Рис. 1.4. Обґрунтування необхідності супортів: а) – літери Y,H і T з супортами (світло-сірі лінії); *б) – надруковані з супортами літери Y, H і T*

Виходячи з попередніх обґрунтувань, недоліком використання супортів при 3Д-друці – необхідність фінішної (післяпроцесної) обробки. Та частина деталі, яка перебувала в контакті з супортами може бути пошкоджена. Якість цієї поверхні буде значно гірше і потребує додаткової обробки. Також для використання додаткових підтримок у виді супортів, буде використовуватися додатковий матеріал. Окремо варто врахувати час, який знадобиться для видалення супортів і пост обробки виробу.

1.2. Післяпроцесна обробка об’єктів адитивного виробництва

У нинішній період параметри STereoLithography (STL) [4] (Файл моделювання стереолітографії) набуває завжди чималої популярності. Його розробка проводилася для стереолітографії (об’ємної літографії) і призначалася з метою швидкого прототипування і формування візуалізації тривимірних моделей. Параметри файлу STL застосовує серію зв’язаних трикутників (рис. 1.5) з метою відтворення геометрії площини твердої моделі. Теоретично, з підняттям роздільної здатності буде використовуватись більша кількість трикутників, ліпше апроксимуючої поверхні 3D-моделі, проте також збільшується обсяг файлу STL [5].

Рис. 1.5.Приклад 3D-моделі у форматі STL

Всі нинішні програми CAD (Computer Aided Design) дають можливість експортувати свої параметри файлів в STL. Потім 3D-модель перетворюється на механічну мову (G-code) за допомогою процесу, що називається «нарізкою», а потім модель готова для друку. Якщо експортувати файл у дуже малому розширенні, модель матиме зримі трикутники під час друку на його поверхні (рис.1.6-1.8). Зазвичай таке не бажано, але для створення «низькополігональних» моделей з цифровим зовнішнім виглядом можна використовувати.

Кожен комплект CAD містить різні види укладання STL, проте більшість застосовує 2 ключові параметри: височина хорди і ракурс. Рівень хорди (рис. 1.9) - це максимальний проміжок серед поверхнею початкової 3D моделі і поверхнею файлу STL, що дозволяє скористатися програмне забезпечення. Застосування найменшої височини хорди допоможе найбільш чітко продемонструвати кривизну площини.

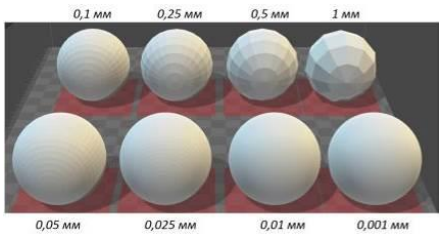


Рис. 1.6. Максимальне лінійне відхилення

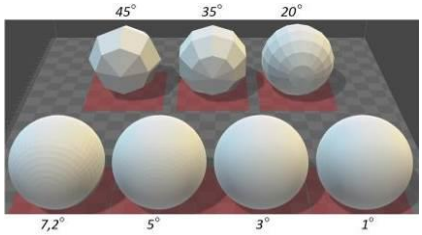


Рис. 1.7. Максимальне кутове відхилення

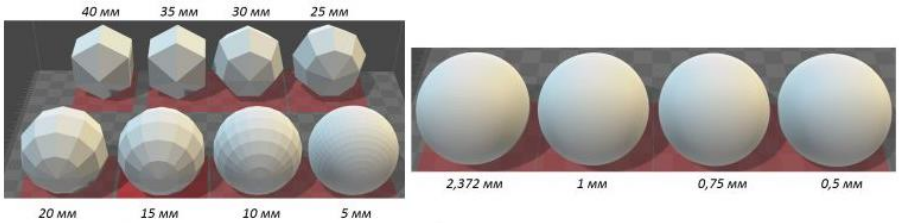


Рис. 1.8. Максимальна довжина ребра

Рекомендоване значення для височини хорди є 1/20 від товщини пласта 3D-друку і не нижче 0,001 мм. Це завжди призводить до формування STL-файлу разом з досконалою точністю з метою багатьох доповнень 3D-друку. Експорт разом з мінімальним допуском не впливає на якість друку, як і більшість звичайних 3D-принтерів не готові відтворити настільки великий рівень деталізації.

У свою чергу однокутовий доступ (рис. 1.10) обмежує ракурс серед нормалями трикутників, що розташовуються поруч. Значення за умовчанням часто є 15^0. Окремі програми також встановлюють цей доступ як значення від 0 до 1. Якщо з метою більш рівної поверхні не потрібно більша установка, то рекомендоване значення за умовчанням 15^0 (або 0). З метою стандартної діяльності проекту-слайсера необхідно, для того щоб обсяг STL-файлу ніяк не перевищував 70 МБ. Інакше, у праці слайсера можуть виникнути проблеми. Найкращий обсяг файлу 5-6 МБ [6].

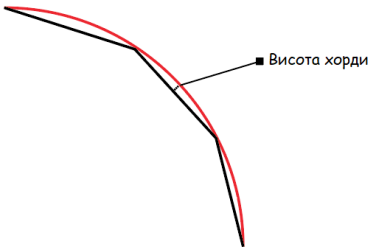


Рис. 1.9. Висота хорди

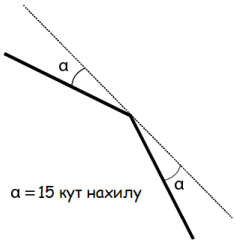


Рис. 1.10. Кутовий допуск

1.3. Рекомендації щодо практичної реалізації післяпроцесної обробки об’єктів адитивного виробництва

1.3.1. Вибір прототипу інструментарію для післяпроцесної обробки

З метою практичного здійснення пропонованого розкладу, у якості прототипу підібрано конструктор-комплект 2D-верстата лазерного гравіювання в основі верстата S1 (рис. 1.15). Властивості лазерного модуля (гарантує регулювання фокуса), наведені у табл. 1.4, а верстата у табл. 1.5.

Рис. 1.15. 2D-верстат

Таблиця 1.4

Характеристики лазерного модуля

Потужність	3000 мВт	TTL драйвер	кулер охолодження
Фокусна відстань	до 30 мм	Довжина хвилі	405 нм (фіолетовий лазер)
Робоча напруга	12 В	Робочий струм	до 2 А
Розмір лазера	86x33x33 мм	Скляна лінза в регулювальному кільці для фокусування	
Вага	140 г.		
Довжина кабелю	~ 50 см		

Таблиця 1.5

Характеристики верстату S1

Робоча зона / площа	395x285 мм	ПЗ: VigoEngraver (на диску в комплекті), підтримує Benbox, Ribs, Laserax
Габарити верстата	526x485x193 мм	

Оскільки в початковому варіанті зразок ніяк не пристосований для постобробки, у роботі запропоновано його модифікація.

1.3.2. Модифікація механотроніки

У цілях здійснення діяльності з 3D-моделями приєднана третя вісь зсуву (Oz) - рис. 1.15. Крім цього, побудовані кабель-канали; крайні вимикачі згідно з осями Ox і Oy, а також замінена провідна система. З метою продування робочої площини на модулі лазера змонтований кулер.

Як результат, пристрій містить можливість зміни модулів на третій осі зсуву (Oz) - рис. 1.16.а. при таких модифікаціях є можливість встановлювати наступні модулі: лазерний - рис 1.16.б, електрошпиндель з метою фрезерувальної і свердловинної обробки - рис. 1.16.в, а також 3D-руки або екструдери з метою виробництва 3D-моделей.

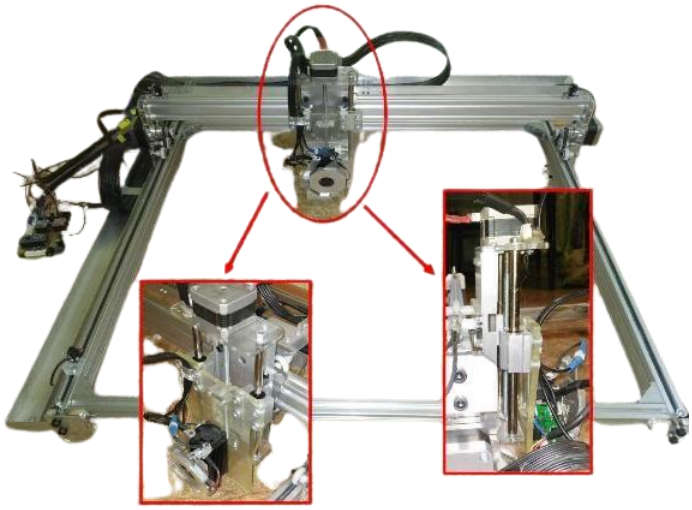
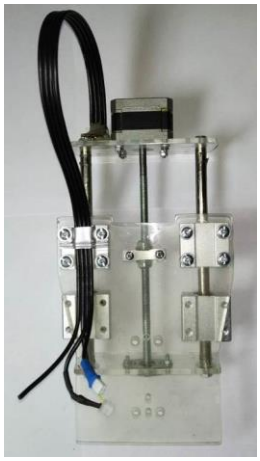
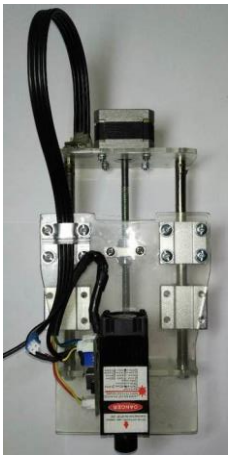


Рис. 1.15. Модифікації верстату S-1



а)



б)

в)

Рис. 1.16. Модифікація механотроніки верстату S-1: а) – вісь зсуву (Oz); б) –лазерний модуль; в) –шпиндель

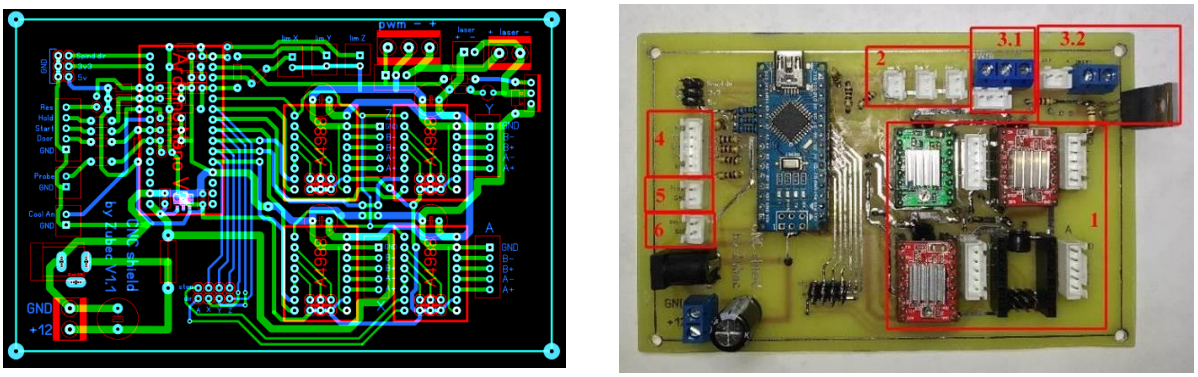
1.3.3. Модифікація цифрового сегменту

Зроблені зміни механічної частини викликають відповідне поліпшення цифрового сектора оснащення.

З цією метою в початковому етапі розроблено плату з метою розширення мікроконтролера (МК) Arduino Nano на базі Atmega328p (рис. 1.17). Ця оплата дає можливість здійснити такі включення:

- 4-х драйверів крокових моторів (крім існуючої підтримки трьох осей (Ox, Oy, Oz) можливо реалізується також допоміжна 4-а лінія зсуву - Oa);
- кінцевиків 3-х осей (Ox, Oy, Oz);
- компонентів, що контролюється ШІМ (широтно-імпульсна модуляція) або перетвореним сигналом ШІМ у 12-вольтову систему;
- приєднання клавіш з метою керування верстатом (Reset, Hold, Start, Door);
- приєднання Z-щупу, з метою побудови карт висот;
- приєднання керування охолодження (пневмообдув лазера або екструдера, охолодження фрези або свердла рідиною та ін.).

В повному обсязі, подібна модернізація дозволила користуватися досить розширеним ПЗ для модулів за основою яких є МК Arduino.



а) б)

Рис. 1.17. Плата розширення для МК Arduino Nano: а) – макет, який розроблений за допомогою ПЗ Sprint-Layout 6.0; б) – практична реалізація

Для керування не зміненим верстатом S-1 не потрібно велика обчислювана продуктивність. Проте при зробленій модернізації збільшується навантаження на МК. Тож, у другому етапі розроблена перехідна оплата (рис. 1.18), яка дає змогу змінити Arduino Nano на ESP32 (к. Espressif Systems,

Піднебесна), що містить більший перелік можливостей і web-інтерфейс, а також застосовує Bluetooth (рис. 1.19) та /або Wi-Fi (рис. 1.20).

Як встановлено, ESP32x - це ряд є заступником МК ESP8266. Дешеві ESP32 разом з невисоким енергоспоживанням, що передає концепцію в процесорі разом з вбудованими Bluetooth і Wi-Fi, контролерами і антенами. Дані модулі будуються к. TSMC згідно з науково-технічним процесом 40 нм. У ESP32 застосовується МК разом з ядром Tensilica Xtensa LX6 (є види разом з 2-ма або одним ядром). В концепцію вбудований радіочастотний тракт: перетворювач з метою симетрування, інтегровані антенні комутатори, радіочастотні елементи, аудіопідсилювач малошумливий, підсилювач потужності, фільтри а також модулі управління живленням.

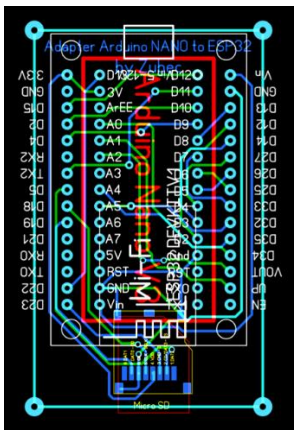


Рис. 1.18. Макет перехідної плати «Adapter Arduino Nano to ESP32», що розроблена в Sprint-Layout 6.0

Bluetooth
Class-1, class-2 and class-3 transmitter without external power amplifier
Enhanced power control
+10 dBm transmitting power
NZIF receiver with -98 dBm sensitivity
Adaptive Frequency Hopping (AFH)
Standard HCI based on SDIO/SPI/UART
High speed UART HCI, up to 4 Mbps
BT 4.2 controller and host stack
Service Discover Protocol (SDP)
General Access Profile (GAP)
Security Manage Protocol (SMP)
Bluetooth Low Energy (BLE)
ATT/GATT
HID
All GATT-based profile supported
SPP-Like GATT-based profile
BLE Beacon
A2DP/AVRCP/SPP, HSP/HFP, RFCOMM
CVSD and SBC for audio codec
Bluetooth Piconet and Scatternet

Рис. 1.19. Bluetooth ESP32

Wi-fi
802.11 n (2.4 GHz), up to 150 Mbps
802.11 e: QoS for wireless multimedia technology
WMM-PS, UAPSD
A-MPDU and A-MSDU aggregation
Block ACK
Fragmentation and defragmentation
Automatic Beacon monitoring/scanning
802.11 i security features: pre-authentication and TSN
Wi-Fi Protected Access (WPA)/WPA2/WPA2-Enterprise/Wi-Fi Protected Setup (WPS)
Infrastructure BSS Station mode/SoftAP mode
Wi-Fi Direct (P2P), P2P Discovery, P2P Group Owner mode and P2P Power Management
UMA compliant and certified
Antenna diversity and selection
Compliant with Bluetooth v4.2 BR/EDR and BLE specification

Рис. 1.20. Характеристики Wi-Fi ESP32

Узагальнені характеристики ESP32 на ведено в табл. 1.6.

Таблиця 1.6

<table>
<tr><td colspan="4">Робочі характеристики</td></tr>
<tr><td colspan="2">Робоча напруга від 2,2 в до 3,6 В
Робоча темп. від -40 ° С до + 125 ° С</td><td colspan="2">Максимальна потужність передачі 19,5 дБм @ 11b, 16,5 дБм @ 11 г, 15,5 дБм @ 11n</td></tr>
<tr><td colspan="4">Максимальна швидкість передачі даних 150 Мбіт/с при 11n HT40, 72 Мбіт/с при 11n HT20, 54 Мбіт/с при 11g і 11 Мбіт/с при 11b</td></tr>
<tr><td colspan="2">Стійка пропускна здатність UDP 135 Мбіт/с</td><td colspan="2">Мінімальна чутливість приймача-98 дБм</td></tr>
<tr><td colspan="4">Склад</td></tr>
<tr><td>Мікроконтролер і управління</td><td colspan="3">Tensilica Xtensa LX6 двоядерний (або одноядерний) 32-розрядний процесор, з тактовою частотою 160 або 240 МГц і продуктивністю до 600 DMIPS (Dhrystone MIPS)</td></tr>
<tr><td colspan="2">Співпроцесор з ультранизьким енергоспоживанням</td><td colspan="2">Пам’ять: 520 КБ пам’яті SRAM</td></tr>
<tr><td>Бездротовий зв’язок</td><td colspan="3">Wi-Fi: 802.11 b/g/n, Bluetooth: v4.2 BR / EDR and BLE</td></tr>
<tr><td>Периферійні інтерфейси:</td><td colspan="3">12-розрядний АЦП до 18 каналів; 2 × 8 бітових ЦАП;
10 × портів для підключення ємнісних датчиків (що вимірюють ємність GPIO); 4 × SPI майстер-інтерфейс (проводові пристрої); 2 × I²C майстер-інтерфейс; 3 × UART інтерфейс</td></tr>
<tr><td colspan="2">SD/SDIO/CE-ATA/MMC/eMMC хост-контролер</td><td colspan="2">Датчик температури</td></tr>
<tr><td colspan="2">SDIO / SPI слейв-контролери (ведені пристрої)</td><td>CAN bus 2.0</td><td>Датчик Хола</td></tr>
<tr><td colspan="4">Ethernet MAC interface з виділеним DMA і IEEE 1588 Precision Time Protocol support</td></tr>
<tr><td colspan="4">ІК пульт дистанційного керування</td></tr>
<tr><td colspan="4">Аналоговий передпідсилювач низького енергоспоживання</td></tr>
<tr><td colspan="4">Можливість підключення двигунів і світлодіодів через ШІМ-вихід</td></tr>
<tr><td>Безпека</td><td colspan="3">Підтримуються всі функції безпеки стандарту IEEE 802.11, в тому числі WFA, WPA / WPA2 і WAPI; безпечне завантаження; шифрування флеш-диска; 1024-бітовий ключ, до 768 бітів для клієнтів; криптографічне апаратне прискорення: AES, SHA-2, RSA, криптографії на основі еліптичних кривих (ECC), апаратний генератор випадкових чисел при включеному WiFi або Bluetooth, інакше використовується генератор псевдовипадкових чисел</td></tr>
<tr><td>Управління живленням</td><td colspan="3">лінійний регулятор з низьким рівнем падіння напруги; індивідуальний харчування для RTC; споживання 5-2,5 мкА в режимі «глибокий сон»; пробудження по перериванню від GPIO, таймера, вимірюванню АЦП, переривання ємнісного сенсорного датчика</td></tr>
</table>

Для двох МК використовувалося ПЗ LaserGrbl (рис. 1.21). Воно являє собою трансформацію популярної проекту Grbl (автор – Simen Svale Skogsrud), що містить відкритий код і спочатку була винайдена для платформи Arduino. Grbl пристосовано з метою використання в сотнях проектів, у тому числі верстати разом з числовим програмним управлінням (ЧПУ), наприклад: лазерні різаки, автоматичні рукописні машини, плотери, 3D-принтерів і автоматизованих токарних станків і т.п. ПЗ Grbl розпочалося використовуватися в 2009 р., на основі якого створено багато проектів для різних платформ: Windows, Linux і IOS. Принцип діяльність цих проектів подібне, вони посилають завдання у вигляді G-кодування. Grbl розроблений мовою С, застосовуючи всі характерні риси Atmega328p Arduino з метою

здійснення точних розрахунків і асинхронної роботи. Він здатний підтримувати темп кроку 30 кГц. У нинішній період Grbl підтримує лише 3 осі: X, Y і Z.

Оскільки ПЗ LaserGrbl мала можливість гравіювати тільки в двох вимірах, для МК сформовані спеціальні зміни Grbl-кодування (рис. 1.22). вони надали допоміжні функції для додаткового управління третьою віссю.

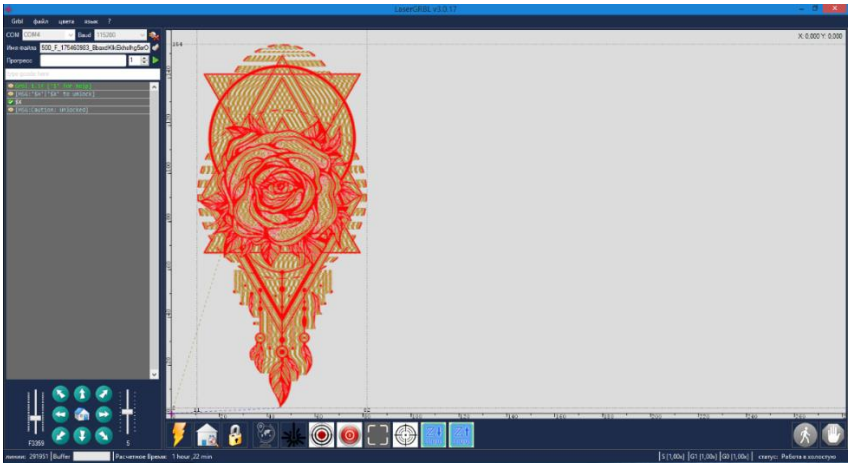


Рис. 1.21. Інтерфейс LaserGrbl V3.0.17

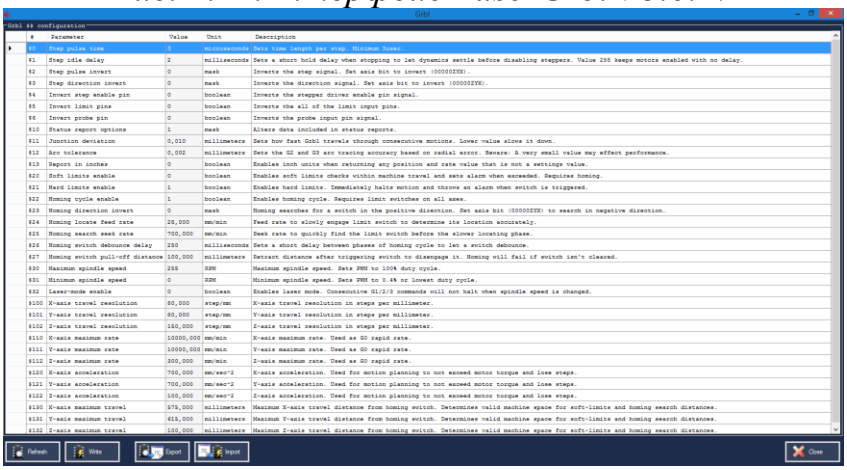


Рис. 1.22. Конфігурації Grbl

У зазначених конфігураціях записані ключові опції з метою покращеного верстата S-1:

- період регулювання сили лазера з 0% аж до 100%;
- налаштування ступеня сигналу у виході ШІМ з 0,02 до 5 В;
- інтегровані суворі межі робочої поверхні (підтримка встановлених кінцевиків);

- встановлена максимальна швидкість (мм/хв), з якою пересувається кожна вісь;
- найбільше збільшення швидкодії (мм/с2);
- максимальний довжина пересування усіх вісей.

Сутність такого розкладу полягає в тому, що Grbl віддає змінений G-code з комп'ютера або накопичувача (з мікроконтролера ESP32) в МК, що у свою чергу здійснює встановлені вказівки.

При цьому, G-код містить окремі особливості. Незважаючи на єдину регламентацію, G-код містить величезну масу реалізацій і розширень, які додаються, розробниками обладнань з використанням ЧПУ, що не перешкоджає йому зберігатися основним стандартом в сфері.

Таким чином, схема разом із застосуванням G-кодування містить строгу структуру. Команди керування поєднуються в кадри – група команд. Кадр закінчується знаком переходу рядка (CR/LF) і здатний мати абсолютно встановлений номер, що починається з літери N, за винятком коментарів та першого кадру. Головний (а в деяких випадках також і завершальний) кадр включає тільки єдиний рекомендаційний символ «%». Закінчується програма командами M02 чи M30.

Команди в кожному кадрі виконуються в той же час, отже процедура команд у кадрі не обумовлюється. Проте, в основному, мається на увазі, що основними вказуються підготовчі вказівки, потім команда координатного пересування, в майбутньому - вибір систем обробки і технічні команди.

У програмі виклик підпрограми виконується розпорядженням M98 разом із приписом невід'ємного параметра ім'я підпрограми P. Приклад призову O112: M98 P112. Допустимий ряд заклику підпрограми розпорядження кількості закликів підпрограми додаванням необов’язкового параметру L, наприклад подвійний заклик підпрограми 112: M98 P112 L2, що ж, наприклад, може бути корисно при описанні виконання другого проходу для чистового проходу після попередньої обробки. Поруч спущеному параметрі L підпрограма викликається одноразово.

Приклад проекту вирізування 2 прямокутних отворів 10×20 міліметрів разом з координатами нижнього кута отворів x = 57, y = 62, x = 104, y = 76 в листяній болванці товщиною 5 міліметрів разом із закликом підпрограми яка описує вирізання отворів

(Фрагмент програми)

G00 X57 Y62 (позиціонування по X, Y на 1-е отвір)

M98 P112 (вирізання 1-го отвору)

G00 X104 Y76 (позиціонування по X, Y на 2-е отвір)

M98 P112 (вирізання 2-го отвору)

...

M02 (Кінець програми)

...

(Тіло підпрограми)

O112 (Метка підпрограми, номер 112)

G00 Z1 (Підведення інструмента на висоту 1 мм над поверхнею заготовки зі швидкістю холостого переміщення)

G01 F40 Z-5.5 (Врізання інструменту на глибину 5,5 мм в заготовку зі швидкістю 40 мм / хв)

G91 (Перехід в відносну систему координат, в цій системі спочатку X = 0, Y = 0)

G01 F20 X10 (Вирізання 1-й боку прямокутника зі швидкістю 20 мм / хв)

Y20 (Вирізання 2-й боку прямокутника зі швидкістю 20 мм / хв)

X0 (Вирізання 3-й боку прямокутника зі швидкістю 20 мм / хв)

Y0 (Вирізання 4-й боку прямокутника зі швидкістю 20 мм / хв)

G90 (Перехід в абсолютну систему координат, відновлення поточних координат до переходу в відносну систему)

G00 Z5 (Підйом інструменту на висоту 5 мм над поверхнею заготовки зі швидкістю холостого переміщення)

M99 (Повернення в викликала програму або підпрограму)

...

M30 (Кінець інтерпретується коду програми. Після виконання цієї команди покажчик номера кадру встановлюється на 1-й рядок програми і виконання програми зупиняється)

Ключові (іменовані в стандарті підготовчими) вказівки мови починаються з букви G:

- Пересування робочих органів обладнання разом із встановленою швидкістю (прямолінійне та кільцеве)
- Виконання стандартних послідовностей (подібних, як переробка отворів і нарізка)
- Керування параметрами приладу, режимами розташування та робочої площини

Таблиця. 1.7

Підготовчі (основні) команди

Коди	**Опис**
G00-G03	Позиціонування інструменту
G17-G19	Перемикання робочих площин (XY, ZX, YZ)
G20-G21	Не стандартизовано
G40-G44	Компенсація розміру різних частин інструменту (довжина, діаметр)
G53-G59	Перемикання систем координат
G80-G85	Цикли свердління, розточування, нарізування різьблення
G90-G91	Перемикання систем координат (абсолютна, відносна)

Введення модуля ESP32 в структуру зміненого верстата, що використовує Grbl, містить наступні переваги:

- швидше (експериментальним шляхом визначено – згідно з останнім критерієм, в 4 рази) кроку згідно Grbl;
- найменші витрати речовини;
- більш значний розмір флеш-пам'яті і ОЗП (допускається користуватися більш значний кеш-пам'ять планувальника і додати значніше функцій);
- I/O містить приблизно таке ж число контактів, так само як і Arduino Nano, що є унікальною основою з метою Grbl;
- підвищена кількість переферійних приладів (наприклад, значніше таймерів і нарощених функцій, ніж Nano);
- інтегровані Bluetooth та Wi-Fi;

- RTOS – спеціалізовані функції допускається доповнювати, не впливаючи на ефективність концепції управління переміщенням.

1.3.4. Оцінка впливу лазерного випромінювання на PETG

Відповідно до п. 1.2 з метою післяпроцесної обробки адитивного виготовлення прообразу фрактального антенного компонента, зробленого з PETG, у процесі вивчення проаналізовано вплив лазерного випромінювання на джерело з фокусною відстанню 50 міліметрів. Був використаний лазерний модуль, що включає промінь потужністю 3000 мВт і довжиною хвилі 450 нм. Продуктивність випромінювання змінювалася з 0 до 100% крок різниці був 10%, і тривалість випромінювання була: 1 і 3 с. Отримані результати представлені в табл. 1.8 а також на рис. 1.23.

Таким чином, найкращий вид обробки PETG цим лазерним модулем відповідає випромінюванню у з фокусною відстанню 50 міліметрів, тривалістю в 1 с та при силі випромінювання лазера в 60-80%. Дані налаштування дають можливість обробляти матеріал зі швидкістю 1 мм/с.

Таблиця 1.8

Оцінка впливу оптичного випромінювання на PETG

Потужність випромінювання		Тривалість випромінювання	
%	мВт	1 секунда	3 секунди
10%	300	Не впливає на пластик	
20%	600	Не впливає на пластик	
30%	900	Не впливає на пластик	Мало відчутно впливає на пластик
40%	1200	Мало відчутно впливає на пластик	
50%	1500	Мало відчутно впливає на пластик	Оптимальна потужність для обробки
60%	1800	Оптимальна потужність для обробки	
70%	2100	Оптимальна потужність для обробки	Занадто плавить пластик
80%	2400	Оптимальна потужність для обробки	Занадто плавить пластик
90%	2700	Занадто плавить пластик	
100%	3000	Занадто плавить пластик	

Рис.1.23. Вплив лазерного випромінювання на тестовий зразок PETG

Після отримання раціонального виду обробки PETG була зроблена обробка прообразу фрактального антенного компонента, зробленого з блакитного PETG. Обробка здійснюється зі швидкістю 5 та 15 кроків/мм.

На фото видно як лазер обробив верхній шар (з правої сторони) (рис. 1.24, рис. 1.25).

Таким чином було згладжена нерівність після 3d друку прототипу фрактального антенного елементу.

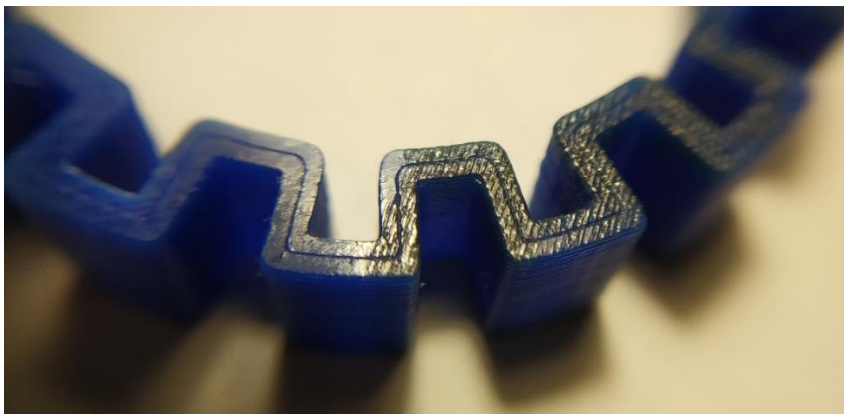


Рис. 1.24. Оброблений прототип фрактального антенного елементу з кроком в 5 кроків/мм

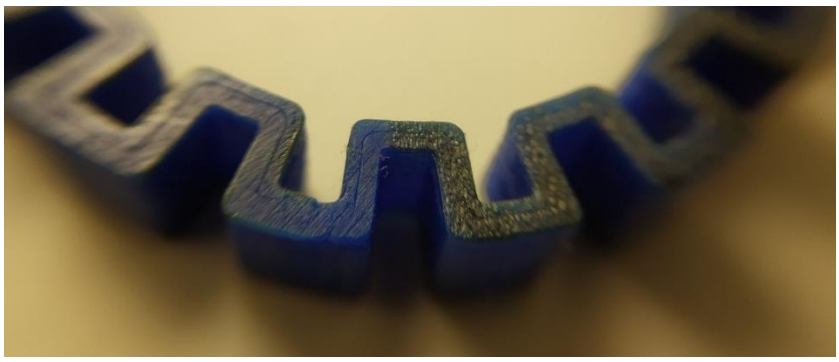


Рис. 1.25. Оброблений прототип фрактального антенного елементу з кроком в 15 кроків/мм

1.4. Використання модуля ESP32

ESP32 являє собою серію недорогих, малопотужних мікроконтролерів system on a chip з вбудованим Wi-Fi та дворежимним Bluetooth. У серії ESP32 використовується або мікропроцесор Tensilica Xtensa LX6 як у двоядерному, так і в одноядерному виконанні, або двоядерний мікропроцесор Xtensa LX7, або одноядерний мікропроцесор RISC-V. В комплект поставки входять вбудовані антенні перемикачі, радіочастотний балун, підсилювач потужності, малошумливий приймальний підсилювач, фільтри і харчування.-модулі управління. ESP32 створений і розроблений Espressif Systems, китайською компанією з Шанхаю, і проводиться TSMC з використанням їх 40-нм технологічного процесу. Він є наступником мікроконтролера ESP8266.

У нинішній період пристрій функціонує разом з МК ESP32, що містить у борту пристрій бездротового взаємозв'язку ESP-WROOM32 (рис. 1.26) з наступними еталонами: Wi-Fi 802.11 b/g/n, Bluetooth v4.2 BR / EDR and BLE. Ці зразки в нинішній період застарілі і відповідно до більш новими мають низьку швидкість передачі і невелику безпеку відомостей.

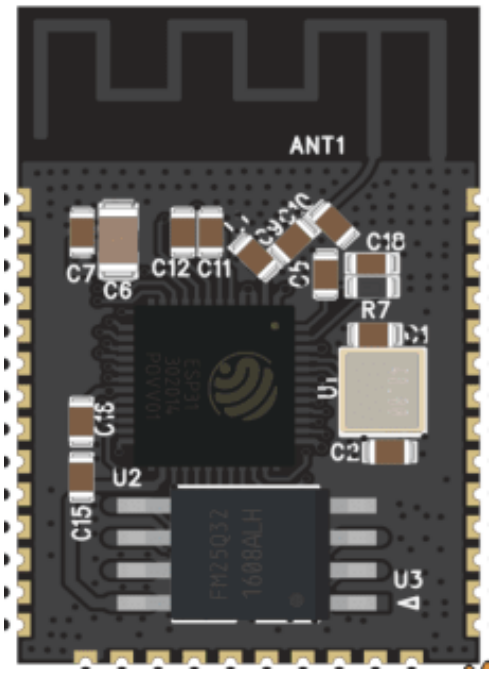


Рис. 1.26. Модуль ESP-WROOM32

Основні характеристик ESP32 наведені в таблиці 1.9.

Таблиця. 1.9

Процесори	Двоядерний (або одноядерний) 32-розрядний мікропроцесор Xtensa LX6 з частотою 160 або 240 МГц і продуктивністю до 600 DMIPS.
	Співпроцесор з наднизьким енергоспоживанням (ULP)
Пам'ять	320 КБ ОЗУ, 448 КБ ПЗУ
Бездротове підключення	Wi-Fi: 802.11 b/g/n
	Bluetooth: v4.2 BR / EDR і BLE (спільно використовує радіо з Wi-Fi)
Периферійні інтерфейси	34 × програмовані графічні процесори
	12-розрядний SAR-АЦП до 18 каналів
	2 × 8-розрядних ЦАП
	10 × сенсорні датчики (GPIO з ємнісним датчиком)
	4 × SPI майстер-інтерфейсу (провідні пристрої)
	2 × I2S майстер-інтерфейсу
	2 × I2C майстер-інтерфейсу
	3 × UART інтерфейсу
	Хост-контролер SD / SDIO / CE-ATA / MMC / eMMC
	Ведений контролер SDIO/SPI
	Інтерфейс Ethernet MAC з виділеним DMA і запланованої підтримкою протоколу точного часу IEEE 1588
	CAN-шина 2.0
	Інфрачервоний пульт дистанційного управління (TX/ RX, до 8 каналів)
	Можливість підключення двигунів і світлодіодів через ШІМ-вихід
	Датчик Холла
	Аналоговий передпідсилювач низького енергоспоживання
Безпека	Підтримуються всі функції безпеки стандарту IEEE 802.11, в тому числі WFA, WPA/WPA2 і WAPI
	Безпечне завантаження
	Флеш-шифрування
	1024-бітний OTP, до 768-біт для клієнтів
	Криптографічне апаратне прискорення: AES, SHA-2, RSA, криптографія на еліптичних кривих (ECC), генератор випадкових чисел (ГВЧ)
Керування живленням	Лінійний регулятор з низьким рівнем падіння напруги
	Індивідуальне живлення для RTC
	споживання 5-2,5 мкА в режимі «глибокий сон»
	Пробудження по перериванню від GPIO, таймера, вимірювання АЦП, переривання ємнісного сенсорного датчика
	Робоча напруга від 2,2—3,6 В

Зіставлення даних Wi-Fi: 802.11n (WiFi 4), що сьогодні встановлений і Wi-Fi: 802.11 ax (Wi-Fi 6), що в даний період найбільш новий, занесений в таблицю 1.10.

Таблиця. 1.10

	802.11 n (Wi-Fi 4)	802.11 ax (Wi-Fi 6)
Дата релізу	2009	2019
Робоча частота	2.4 ГГц & 5ГГц	2.4 ГГц & 5ГГц, spanning to 1ГГц – 7ГГц
Ширина смуги каналу	20МГц, 40МГц	20МГц/40МГц @ 2.4 ГГц, 80МГц, 80+80 мгц & 160МГц @ 5ГГц
Кількість піднесучих	64, 128	64, 128, 256, 512, 1024, 2048
Рознесення піднесучих	312.5 кГц	78.125 кГц
Тривалість символу і циклічного префікса	3.6 мкс (короткий захисний інтервал) 4 мкс (довгий захисний інтервал)	12.8 мкс (0.8/1.6/3.2 мкс циклічний префікс)
Тип модуляції і швидкість кодування	64-QAM, 5/6	1024-QAM, 5/6
Кодування сигналу	6 Біт на символ	10 Біт на символ
Швидкість передачі даних	від 54 Мбіт/с до 600 Мбіт/с (max 4 просторових потоку)	600 Мбіт/с (80 мгц, 1 просторовий потік) 9607.8 Мбіт/с (160МГц, 8 просторових потоків)
Технології MIMO	SU-MIMO-OFDM	MU-MIMO-OFDMA

З таблиці 1.10 помітно, Wi-Fi: 802.11 ax (Wi-Fi 6) в рази швидше за власного самого колишнього предка Wi-Fi: 802.11n (Wi-Fi 4).

Порівняння даних Bluetooth 4.2, що визначено і Bluetooth 5.0, що в даний період найбільш завершальний, записані в таблицю 1.11.

Таблиця. 1.11

Технічні характеристики або особливості	Bluetooth 4.2	Bluetooth 5.0
Дата релізу	04.12.2013	16.06.2016
Робоча частота	2.4 до 2.485 ГГц	2.4 до 2.485 ГГц
Швидкість	24 Мб/с	48 Мб/с
Дальність дії	100 м	300 м
Потрібна потужність	Висока	Низька
Ємність повідомлення	Невелика, приблизно 31 байт	Близька 255 байт
Надійність роботи в умовах перевантаженого середовища	Менше	Більше
Термін служби акумулятора	Менше	Більше
Контроль безпеки	Гірше	Краще
Теоретична пропускна здатність	1 Мбіт/с	2 Мбіт/с, дає приблизно 1.6 Мбіт/с
Надійність	Мала	Висока
Підтримка пристроїв IoT	Ні	Так

З таблиці 1.1 помітно що Bluetooth 5.0 найшвидший приблизно вдвічі за Bluetooth 4.2, найбільш оберігаємо як від потрапляння, як і від втрат відомостей, а також мінімальна споживча продуктивність, що підвищує незалежність приладів.

Також прийматиме вигляд включення верстата для «Інтернет речей».

1.5. Вибір середовищ програмування

Більшість робочого часу програміста витрачається на написання коду, тому редактори коду є невід'ємним засобом у роботі.

Існує два основних типи редакторів: IDE і «легкі» редактори. Багато людей використовують один інструмент кожного типу.

Терміном IDE (Integrated Development Environment, «інтегроване середовище розробки») відноситься до потужного редактора з багатьма функціями, які працюють протягом усього проекту. Як випливає з назви, це не просто редактор, а набагато більше.

IDE завантажує проект (може складатися з багатьох файлів), дозволяє перемикатися між файлами, пропонує автоматичне доповнення коду в проекті (а не лише відкритого файлу), а також інтегрується з системами контролю версій (наприклад, git), середовищем для тестування та іншими інструментами на рівні всього проекту.

«Легкі» редактори не такі потужні, як IDE, але вони швидкі, зручні та прості у використанні. Основною роботою їх є швидке відкриття та редагування потрібного файлу.

Основною відмінністю «легкого» редактора та IDE полягає в тому, що IDE працює з проектами будь-якого розміру, тому завантажує при запуску більше даних та при необхідності аналізує структуру проекту, тощо. Якщо розробник має справу лише з одним файлом, відкрити його у «легкому» редакторі набагато швидше.

На практиці «легкі» редактори можуть мати безліч плагінів, включаючи автодоповнення й аналізатори синтаксису на рівні директорії, тому межа між

IDE та «легкими» редакторами стирається.

Прикладами «легких» редакторів є:

1. Atom (багатоплатформовий, безкоштовний).
2. Sublime Text (багатоплатформовий, умовно-безкоштовний).
3. Notepad ++ (Windows, безкоштовний).

В ході роботи буде необхідність у використанні кількох редакторів, для розробки різних платформ та операційних систем.

Так для програмування, розробки серверної та клієнтської частини МК ESP32 знадобиться Arduino IDE та Visual Studio IDE.

Для розробки ПЗ операційної системи Windows, буде використовуватись Visual Studio IDE.

Для розробки додатка для ОС Adndoid використовується Android Studio IDE.

Arduino IDE — інтегроване середовище розробки для Windows, MacOS та Linux, розроблене на C та C++, призначене для створення та завантаження програм на Arduino-сумісні плати, а також на плати інших виробників.

Вихідний код середовища випущений під загальнодоступною ліцензією GNU версії 2. Підтримує мови C і C++ з використанням спеціальних правил структурування коду. Arduino IDE надає бібліотеку програмного забезпечення з проекту Wiring, яка надає безліч загальних процедур введення та виведення. Для написаного користувачем коду потрібні лише дві базові функції для запуску ескізу та основного циклу програми, які скомпільовані та пов'язані із заглушкою програми main () у виконувану циклічну програму з ланцюжком інструментів GNU, також включеною до дистрибутиву IDE. Використовує програму avrdude для перетворення коду, що виконується, в текстовий файл у шістнадцятковому кодуванні, який завантажується в плату Arduino програмою-завантажувачем у вбудованому програмному забезпеченні плати.

Зі зростанням популярності Arduino інші постачальники в якості програмної платформи почали впроваджувати компілятори та інструменти з відкритим вихідним кодом (ядра), які можуть створювати і завантажувати

ескізи в інші мікроконтролери, що не підтримуються офіційною лінійкою мікроконтролерів Arduino.

Visual Studio — це інтегроване середовище розробки (IDE) від Microsoft. Він використовується для розробки комп'ютерних програм, а також веб-сайтів, веб-додатків, веб-сервісів і мобільних додатків. Visual Studio використовує платформи розробки програмного забезпечення Microsoft, такі як Windows API, Windows Forms, Windows Presentation Foundation, Windows Store і Microsoft Silverlight. Він може створити як рідний, так і керований код.

Visual Studio містить редактор коду, який підтримує IntelliSense (компонент завершення коду), а також рефакторинг коду. Вбудований налагоджувач працює як налагоджувач на рівні джерела, так і на рівні машини. Інші вбудовані інструменти включають профайлер коду, конструктор для створення додатків графічного інтерфейсу користувача, веб-дизайнер, конструктор класів і конструктор схем бази даних. Він приймає плагіни, які розширюють функціональні можливості майже на всіх рівнях, включаючи додавання підтримки систем контролю джерел (наприклад, Subversion і Git) і додавання нових наборів інструментів, таких як редактори та візуальні дизайнери для доменно-спеціальних мов або наборів інструментів для інших аспектів розробки програмного забезпечення. життєвий цикл (наприклад, клієнт Azure DevOps: Team Explorer).

Visual Studio підтримує 36 різних мов програмування та дозволяє редактору коду та налагоджувачу підтримувати (різним ступенем) майже будь-яку мову програмування, якщо існує спеціальна служба для цієї мови. Вбудовані мови включають C, C++, C++/CLI, Visual Basic .NET, C#, F#, JavaScript, TypeScript, XML, XSLT, HTML і CSS. Підтримка інших мов, таких як Python, Ruby, Node.js і M, доступна через плагіни. Java (і J#) підтримувалися в минулому.

Найпростіша версія Visual Studio, версія Community, доступна безкоштовно. Слоган видання Visual Studio Community: «Безкоштовна, повнофункціональна IDE для студентів, розробників із відкритим кодом і

індивідуальних розробників».

Visual Studio Code (VS Code) – текстовий редактор, розроблений Microsoft для Windows, Linux та macOS. Позиціонується як «легкий» редактор коду для кросплатформної розробки веб- та хмарних програм. Включає в себе відладчик, інструменти для роботи з Git, підсвічування синтаксису, IntelliSense та засоби для рефакторингу. Має широкі можливості для кастомізації: теми користувача, поєднання клавіш і файли конфігурації. Поширюється безкоштовно, розробляється як програмне забезпечення з відкритим вихідним кодом, але готові зборки поширюються під ліцензією проприетарной.

Visual Studio Code створений на основі Electron и реалізується через веб-редактор Monaco, розроблений для Visual Studio Online.

Android Studio – інтегроване середовище розробки (IDE) для платформи Android, представлене на конференції менеджером по продукції корпорації Google – Еллі Паверс. 8 грудня 2014 року компанія Google випустила перший стабільний реліз Android Studio 1.0.

Android Studio замінила плагін ADT для платформи Eclipse. Це середовище базується на вихідному коді продукту IntelliJ IDEA Community Edition, розроблений компанією JetBrains. Android Studio розвивається як частина відкритої моделі розробки та розповсюджувалася під ліцензією Apache 2.0.

Бінарні складання підготовлені для Linux (для тестування використаний Ubuntu), macOS і Windows . Це середовище не лише надає інструменти для розробки додатків для смартфонів і планшетів, але і для смарт-годинників на базі Wear OS, телевізорів (Android TV), окулярів Google Glass і інформаційно-розважальних систем автомобіля (Android Auto). Для програм, розроблених спочатку за допомогою Eclipse та ADT Plugin, підготовлені інструменти для автоматичного імпорту існуючих проектів у Android Studio.

Середовище розробки підходить для виконання типових завдань, які вирішуються в процесі розробки додатків для платформи Android . У тому числі у середовище включені засоби для спрощення тестування програм на

сумісність з різними версіями платформи та інструменти для проектування застосунків, що працюють на пристроях з екранами різної роздільності (планшети, смартфони, ноутбуки, годинники, окуляри тощо). Окрім можливостей присутніх в IDEA, Android Studio також має ряд додаткових функцій, таких як нова уніфікована підсистема створення додатків, тестування та розгортання на основі інструменту збірки Gradle, яка підтримує використання безперервних засобів інтеграції

Для прискорення розробки додатків передбачено набір стандартних елементів інтерфейсу та візуальний редактор для компонування елементів, на якому можна легко переглядати різні стани інтерфейсу програми (наприклад, можна подивитися як інтерфейс буде виглядати для різних версій Android і розмірів екрану). Для створення нестандартного інтерфейсу існує майстер, що підтримує використання шаблонів, на основі яких можна створити власні елементи дизайну. У IDE присутні вбудовані функції завантаження типових прикладів коду з GitHub.

Він також містить передові інструменти рефакторингу, розроблені для платформи Android, такі як перевірка сумісності з попередніми версіями, виявлення проблем з продуктивністю, моніторинг використання пам’яті та оцінки юзабіліті. Також можна відмітити ряд функцій в Android Studio такі як:

1. У редакторі присутній режим швидкого внесення правок.
2. Система підсвічування, статичного аналізу та виявлення помилок розширена підтримкою Android API.
3. Інтегрована підтримка оптимізатора коду ProGuard.
4. Вбудовані засоби генерації цифрових підписів.
5. Надано інтерфейс для управління перекладами на інші мови.

1.6. Загальна постановка задачі

Проведений аналіз шляхів підвищення ефективності технологічного процесу адитивного виробництва елементів антенних систем за рахунок удосконалення післяпроцесної обробки, визначив основну задачу досліджень, яка полягає в розробці пропозицій щодо використання для фінішної обробки об’єктів адитивного виробництва лазерних оптичних модулів та сучасних технологій інфокомунікацій для їх управління.

Для повного підвищення ефективності необхідне вдосконалення не лише апаратної частини, а і програмної. Щоб повною мірою використати післяпроцесну обробку необхідно зробити програмне удосконалення не лише самого МК ESP32 керування станка з ЧПУ, а й розробити ПЗ та мобільний додаток сумісні з програмним забезпеченням даного МК.

Відповідно до розв’язуваної задачі, в роботі необхідно вирішити наступні задачі.

1. Провести аналіз роботи з МК ESP32 та реалізації сервер-клієнтської частини.
2. Спроектувати серверну частину МК.
3. Спроектувати WEB-інтерфейс для МК.
4. Визначитися з концепцією розробки ПЗ.
5. Розробити ПЗ для ОС Windows.
6. Розробити додаток для ОС Android.
7. Виконати технічне обґрунтування прийнятих рішень.
8. Розглянути перспективи подальшого розвитку.

1.7. Висновок за першим розділом

На даний, час, адитивне виробництво (3D-друк) забезпечує пошарове створення об'ємних виробів з різних матеріалів. У деяких областях, зокрема – в машинобудуванні, це вже реалізовано: переважна більшість 3D-принтерів орієнтовані на друк матеріалом одного типу, наприклад – термопластичними полімерами або металами, чого цілком достатньо для виробництва механічних деталей. Як тільки виникає необхідність виробництва продукції, що складається з різнотипних матеріалів, виникає потреба в більш складному спеціалізованому обладнанні. Особливо це стосується електроніки.

Розглянута оптимізація післяпроцесної обробки результатів адитивного виробництва дозволяє здешевити етап проектування НВЧ-компонентів. Даний підхід базується на використанні лазеру. При цьому, високотемпературний вплив лазерного випромінювання забезпечує видалення матеріалу.

Метою подальших робіт є удосконалення та розробка програмної частини для використання на різних платформах. При цьому запрограмувати МК ESP32, який керує станком з ЧПУ для необхідних потреб.

В даному розділі, на достатньому рівні, було розглянуто, описано та проаналізовано всі складові мети кваліфікаційної роботи. У наступному розділі пропонується провести проектування серверної та клієнтської частин МК ESP32, необхідних ПЗ та додатку сумісництва з серверною частиною МК ESP32.

РОЗДІЛ 2.
ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1. Аналіз та вибір мов програмування

2.1.1. Вибір мови програмування для ESP32

На сьогоднішній день є велика кількість різноманітних мікроконтролерів, які доступні як і для індивідуального використання і навчання з їх роботою, так і для масового виробництва і крупних проектів і розробок. Звичайно ж мікроконтролери в незапрограмованому і не підключеному виді не мають ніякого інтересу. Тому для МК пишуть програмний код, який предназначений для певної роботи МК. Тобто програмний код керує роботою МК.

Мікроконтролер ESP32 має в собі можливість розгортання внутрішнього сервера, для веб-інтерфейсу, то ж для програмування цього МК використовується не одна мова програмування, а певна сукупність мов, яка між собою взаємодіє. Основна мова для МК це вбудована мова програмування. Вбудована мова програмування — це мова програмування, яку розробники використовують у вбудованих системах. Загалом мови пропонують низькорівневий доступ до апаратного забезпечення пристрою. Розробники використовують кілька поширених мов програмування для вбудованих систем.

Розробники використовують різноманітні мови програмування в мікроконтролерах. Найбільш використовувані мови включають C, C++, Python, MicroPython і Java.

Мови програмування, які часто використовуються у програмуванні МК мають деякі ключові переваги. Більшість мов також мають недоліки. Ось переваги та недоліки популярних вбудованих мов програмування:

1. **C**

- **Основи:** C, розроблений на початку 1970-х років, є скомпільованою мовою, яка служить основою для багатьох інших мов.
- **Переваги:** C є ефективною та широко використовуваною мовою програмування. За оцінками галузі, 80% вбудованих систем та мікроконтролерів використовують мову програмування C.
- **Мінуси:** від розробників потрібно розуміти та використовувати технічні методи кодування, які можуть бути складними.

2. **C++**

- **Основи:** ця скомпільована мова містить більшість або всі елементи C, а також інші можливості. Мова була розроблена частково для системного програмування.
- **Плюси:** C++ може бути таким же ефективним, як використання C, але він має бібліотеку стандартів, яка може заощадити час програмістів на написання коду.
- **Мінуси:** складна мова, яку важко вивчити.
- **Коментар експерта:** «Якщо ви будете розглядати C++, у нього так багато можливостей. Але в мові є нюанси, якщо ви не використовуєте його підмножини, ви дійсно можете знизити продуктивність мікроконтролера. Отже, потрібно враховувати це», «У вас є всі ці функції, але це лише справжня частина, яку ви можете використовувати як вбудований розробник. Інші функції можуть уповільнити ваш код, зробити його наче роздутим і просто працювати не дуже добре».

3. **Python**

- **Основи:** Python, розроблений на початку 1980-х (і названий на честь британського комедійного загону «Monty Python»), є популярною мовою програмування. Він чудово підходить для машинного навчання, штучного інтелекту (AI) і аналізу даних, але ви можете використовувати його в багатьох інших програмах.

- **Переваги:** мова є відкритим вихідним кодом, вільна для використання, її легко вивчати, читати та писати.
- **Мінуси:** ця мова не є детермінованою — не для операційних систем реального часу (RTOS).

4. **MicroPython**

- **Основи:** ця мова є версією Python, оптимізованою для мікроконтролерів.
- **Плюси:** MicroPython також є відкритим вихідним кодом, безкоштовним у використанні та простим у вивченні.
- **Мінуси:** код не такий швидкий і може використовувати більше пам'яті порівняно з C або C++
- **Коментар експерта:** Beningo каже, що переваги MicoPython включають «усі бібліотеки, фреймворки, існуючий код і той факт, що всі його знають. Це легко навчитися. Я навіть чув про учнів початкової школи, які писали код на Python».

5. **Java**

- **Основи:** Java — це ефективна мова загального призначення, яка широко використовується для програм, що працюють в Інтернеті. У вбудованих системах Java найкраще підходить для тих, хто працює на ОС Android.
- **Плюси:** після написання у вбудованій системі код переноситься на інший пристрій і є досить надійним.
- **Мінуси:** Мова може бути складною, що призводить до проблем з продуктивністю, включно з графічним інтерфейсом користувача (GUI). Ви не можете використовувати Java у системах реального часу.

6. **JavaScript**

- **Основи:** JavaScript — це текстова мова програмування, яка використовується в деяких вбудованих системах.
- **Плюси:** Хороший варіант, якщо ваша система заснована на HTML5 і потребує значної мережі та графіки.

- **Мінуси:** низька ефективність роботи та може бути складно підтримувати.

7. **Rust**

- **Основи:** у 2010 році співробітник Mozilla розробив цю високорівневу мову програмування, призначену для продуктивності та безпеки.
- **Плюси:** Rust допомагає захищати код із меншою кількістю помилок.
- **Мінуси:** збірка Rust потребує часу. Мова ще не стандартизована та не використовується багато.
- **Коментар експерта:** «Rust все ще дуже новий. Знадобиться більше часу, поки він справді проникне на ринок вбудованих систем на рівні системи. Але його потенціал очевидний, і він повинен бути в кожному стратегічному списку спостереження».

Вибір мови програмування для мікроконтролерів.

Вибираючи мову програмування для мікроконтролерів, потрібно розуміти сильні та слабкі сторони кожної мови. Не менш важливо знати обмеження та цілі вашої системи.

Для багатьох мікроконтролерів найкращим вибором буде C або C++. Частково це тому, що вони є «компільованими» мовами та надзвичайно ефективними. У скомпільованих мовах машина безпосередньо перекладає код, що означає, що мова є швидкою та стабільною.

Пристрої з мінімальною пам’яттю або потужністю часто вимагають такої ефективності. Тому програмісти часто використовують у цих пристроях C або C++. C або C++ часто використовуються в мікроконтролерах і вбудованих пристроях, які використовують реальні операційні системи. Ці системи також вимагають швидкості та ефективності, які забезпечують C і C++. Близько 80% усіх мікроконтролерів використовують мову програмування C.

Але ваша вбудована система також може працювати так само добре або краще з «інтерпретованою» мовою програмування. З інтерпретованою мовою вбудований пристрій не перекладає код безпосередньо. Замість цього код виконує інша програма-інтерпретатор, яка працює на пристрої. Для деяких

вбудованих систем переваги інтерпретованих мов будуть критичними. На відміну від скомпільованих мов, інтерпретовані мови легко переносяться з однієї ОС на іншу. Програмістам їх також набагато легше вивчати, читати та писати.

Python і JavaScript є прикладами інтерпретованих мов програмування. Оскільки інтерпретовані мови потребують додаткової обробки, вони часто працюють повільніше, ніж скомпільовані мови, хоча їхня швидкість покращується. Якщо ваш вбудований пристрій використовує машинне навчання, Python і MicroPython мають значні переваги [7].

Також є необхідність визначитись з мовою програмування для вбудованої серверної частини мікроконтролера ESP32.

Серверне елемент містить найбільш високу відповідальність в першу чергу за єдність, захищеність і обробку індивідуальних даних. Отже наявна потреба у наданні необхідного ступеня інтересу із метою встановлення цільової мови програмування і інструментів розробки. На даний етап часу є певний вибір серед варіантів. Важливо оглянути кожен з них.

При підборі базової платформи серверної частки – основний інтерес прикутий до найвідоміших мов PHP, Python, Java. Вони мають великий списком випробуваних за роки бібліотеки і фреймворки. Проведем більш детальний аналіз кожної з мов.

PHP – скриптова мова програмування, що була створена з метою виконання алгоритмів для генерування HTML-сторінок на стороні веб-сервера. PHP виступає як одна з найбільш поширених мов у галузі розробки веб-програм. PHP забезпечена підтримкою у більшій мірі за рахунок хостинг-провайдерів. Дана мова програмування представляє собою продукт з відкритим кодом. Скрипт проходить процедуру інтерпретації веб-сервером у HTML-код, який далі буде передано під виглядом відповіді на запит клієнта. Однією з відмінностей від не менш популярної скриптової мови JavaScript є те, що користувач не має змоги переглядати виконуваний код, тому що браузер отримує готовий html-код. Це виступає досить серйозною перевагою

у плані безпеки, але негативно впливає на інтерактивність сторінок. Наразі немає жодних заборон для використання PHP з метою генерації JavaScript логіки, яка може бути інтерактивною та виконуються вже на стороні клієнта [8].

Основні переваги PHP:

1. Орієнтація на Web-розробку – PHP створювався, розвивався і підтримується як мова для створення Web-сайтів. Багато конструкцій і процесів ухвалення рішень, створені для зручності роботи в Web-середовищі.

2. Кросплатформеність – PHP перенесений на всі основні операційні системи: можна розробляти сайт в Windows, Mac OS X, а експлуатувати на Linux-сервері. Складнощі перенесення будуть мінімальні і нівелюватися мовою.

3. Безкоштовність – PHP є розробкою зі світу вільного програмного забезпечення, не буде потрібно платити ні за саму мову, ні за більшість супутніх програм (редактори, Web-сервери, бази даних). Більшість програмних продуктів матимуть доступний для вивчення і модифікації вихідний код [9].

Основні недоліки PHP:

1. Має слабкі засоби для роботи з винятками.

2. Глобальні параметри конфігурації впливають на базовий синтаксис мови, що ускладнює настройку сервера і розгортання додатків.

3. Об’єкти передаються за значенням, що бентежить багатьох програмістів, які звикли до передачі об’єктів по посиланню, як це робиться в більшості інших мов.

4. Веб-додатки, написані на PHP, часто мають проблеми з безпекою [10].

Наступною не менш популярною мовою для створення серверних частин додатків є Python.

Python – інтерпретована об’єктно-орієнтована мова програмування високого рівня з динамічною типізацією. До плюсів Python відносяться його простота, велика кількість бібліотек для використання в самих різних

областях, безкоштовність і підтримка спільнотою програмістів. Але є певна кількість недоліків, які можуть поставити під сумнів вибір такої мови.

Недоліки мови Python:

1. Низька швидкість виконання програм, у порівнянні з іншими мовами.

2. Копіювання коду. При копіюванні коду з іншого ресурсу, в деяких випадках, він може копіюватися без збереження відступів. Тому код буде не валідним, а програмісту доведеться додавати табуляцію в кожний рядок. Для вирішення цієї проблеми потрібно або використовувати спеціальні IDE, або додавати в Ваш редактор плагіни для python.

3. Конвертація програми на python в exe. Програми на python мають розширення *py. З метою використання без інтерпретатора, наприклад, на Windows, виникає необхідність у конвертації файлу з розширенням *exe (для цього можна використовувати додаток py2exe). ПЗ у результаті даної процедури може мати значне збільшення у розмірі. Скрипт який займає 30-40 Кб після конвертації може сягати вражаючих 50 Мб. Як наслідок з обробки та вирізання необов'язкових компонентів досягається зменшення розміру до 10-20 Мб, але результат буде гірше, ніж у аналогічного проекту, зробленого, наприклад на C++[11].

4. Складність роботи з Unicode та кирилицею.

Останньою мовою яку варто розглянути є Java.

Java – об'єктно-орієнтована мова програмування зі строгою типізацією даних. Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно-орієнтованих програм. Ряд дій, які в C/C++ повинні здійснювати програмісти, доручено віртуальній машині. Передусім Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням, що в порівнянні, наприклад, з C++ зменшує швидкість роботи програм. За

необхідності таких дій Java дозволяє викликати підпрограми, написані іншими мовами програмування [12].

Переваги Java:

1. Переносимість. Програми, написані на Java, після одноразової трансляції в байт-код можуть бути виконані на будь-якій платформі, для якої реалізована віртуальна Java-машина.

2. Безпека. Функціонування програми повністю визначається (і обмежується) віртуальної Java-машиною.

3. Надійність. У мові Java відсутні механізми, які потенційно призводять до помилок: арифметика покажчиків, неявне перетворення типів з втратою точності та інші. Присутній суворий контроль типів, обов'язковий контроль виняткових ситуацій.

4. Збирач сміття. Звільнення пам'яті при роботі програми здійснюється автоматично за допомогою «збирача сміття», тому програмувати з використанням динамічної пам'яті простіше і надійніше[13].

5. Наявність фреймворків для розробки enterprise проектів.

Недоліки Java:

1. Код потребує контейнера для виконання, що впливає на швидкість роботи.

2. Потребує більше оперативної пам'яті ніж інші мови програмування.

В результаті розгляду даних мов програмування, опираючись на особливості мети дипломної роботи логічним рішенням є обрати PHP оскільки вона має спеціалізовані фреймворки, є найпопулярнішою мовою програмування в enterprise проектах, а також має досить низьких поріг входження, що в свою чергу значно полегшує роботу спеціалісту, а також зменшує затрати часу на розробку базових складових проекту.

2.1.2. Вибір мови програмування програмного забезпечення ОС Windows

Середовище розробки Visual Studio.NET передбачає потужні і зручні ресурси написання, виправлення, компіляції, налагодження і запуску програм, що використовують .NET – сумісні мови. Організація Microsoft підключила до платформи ресурси розробки чотирьох мов: C#, VB.NET, C++ і J++. Мова програмування C# є повнофункціональним об'єктно-спрямованим стилем, що утримує завжди три "стовпи" об'єктно-орієнтованого програмування: інкапсуляцію, успадкування і поліморфізм. Він має чудову підтримку елементів, надійний і стійкий внаслідок застосування "збирання сміття", обробки висновків, безпеки видів. Відштовхуючись від характеру встановленої проблеми, промислових умов і отриманого інструменту була підібрана сфера дослідження Visual Studio.NET і програмування C# [14].

C# (сі Шарп) – це об'єктно-спрямоване програмування. Воно було винайдено в 2001 р., інженерами під керівництвом Андерса Хейлсберга в компанії Microsoft. У цей період є чотири версії мови «сі Шарп».

Найменування «Сі Шарп» (з англ. sharp - дієз) відбувається з музичної нотації, де ознака символ, що додається до основного позначення нотки, означає збільшення відповідного цій нотці звуку на звук. Це так само найменуванню мови C++, де «++» означає, що ж змінна повинна бути підвищена на 1.

Для кількості свідомо значущих висновків, що виконані фірмою Microsoft у складі програмування C#, можна зарахувати такі: - компонентно-спрямований аспект для програмування (що характерний і з метою ідеології Microsoft .NET в повному обсязі); - якості так само як метод інкапсуляції відомостей (характерно також у повному для ООП); - переробка подій (присутні розширення, в цій частині в частки обробки висновків, зокрема, диспетчер try); - уніфікована концепція типізації (відповідає ідеології Microsoft .NET у повному обсязі); - учасники (delegate – формування покажчика на функцію у стилях C і навіть C++); - індексатори (indexer – оператори індексу з метою призову для складових класу-контейнера); -

переміщені оператори (формування ОВП); - Менеджер foreach (переробка всіх компонентів класів-колекцій, подібність Visual Basic); - апаратура boxing і unboxing з метою перетворення видів; - властивості (спосіб оперування метаданими у COM-модифікації); - прямокутні масиви (комплект компонентів разом з допуском згідно з номером індексу і рівним числом стовпчиків і рядків) [15].

Переваги сі-шарпа:

- C# створювався паралельно з каркасом Framework .Net і повною мірою враховує всі можливості - як FCL, і CLR;
- C# є повністю об'єктно-орієнтованою мовою, де навіть типи, вбудовані в мову, представлені класами;
- C# є потужною об'єктною мовою з можливостями спадкування та універсалізації;
- C# є спадкоємцем мов C/C++, зберігаючи найкращі риси цих популярних мов програмування. Спільний із цими мовами синтаксис, знайомі оператори мови полегшують перехід програмістів від C++ до C#;
- Внаслідок скелету Framework .Net, що став надбудовою над операторною концепцією, розробники програмного забезпечення C# отримують ці ж переваги діяльність разом з умовною машиною, що і розробники програмного забезпечення Java. Результативність кодування навіть збільшується, тому що справна сфера CLR передбачає собою програма перехідного складу, в такому випадку період так само як умовна Java-автомобіль представляється інтерпретатором байт-кодування;
- Потужна бібліотека каркасу підтримує зручність побудови різних типів додатків на C#, дозволяючи легко будувати Web-служби, інші види компонентів, досить просто зберігати та отримувати інформацію з бази даних та інших сховищ даних;[16].

2.1.3. Вибір мови програмування додатка Android

В даний час Java є найбільш популярною мовою програмування для мобільного розробки на Android. Хоча Google активно просуває Kotlin як мову, який повинен буде замінити Java. Також додатки під Android пишуть і на інших мовах.

Нижче коротко описані мови програмування, які використовуються для розробки Android:

Java є офіційною мовою для розробки Android і підтримується Android Studio.

Kotlin є офіційним і самим останнім введеним мовою Android; він схожий на Java, але в багатьох відносинах трохи легше.

Kotlin нещодавно з'явився як "інша" офіційного мови для вивчення Android. Так само як і Java, Kotlin функціонує в умовній машині Java. Повністю поєднуємий разом з Java і не викликає ніяких перешкод або збільшення обсягу файлів.

Основа відмінності полягає в тому, що Kotlin потребує менше «шаблонного» кодування, в такому випадку приймати найлегша з метою читання концепція. Також ліквідує подібні похибки, так само як особливість нових пунктів, але також позбавляє Вас від потреби завершувати рядок пунктом разом з комою. Це прекрасний мова програмування, якщо ви тільки вчитеся створювати програмні забезпечення для Android [17].

Kotlin є більш легкою відправною точкою для початківців, і той факт, що можна використовувати Android Studio, є великим плюсом.

Як і Java, C і C++, Kotlin – це статично типізована мова. Він підтримує як об'єктно-орієнтоване, і процедурне програмування. За аналогією з вищезгаданими мовами, основний код Kotlin-програми пишеться у функції main, якою передається масив аргументів командного рядка.

Майбутнє Kotlin

Kotlin це наступний етап розвитку Java, з якою він повністю сумісний. Це робить його відмінним інструментом для мобільних та ентерпрайз-додатків.

Оскільки Kotlin тепер є офіційною мовою Android, можна не боятися, що, вивчивши його, ви залишитеся без роботи.

2.2 Аналіз розробки Веб-інтерфейсу

2.2.1 Взаємодія Веб-сервера з Веб-інтерфейсом

Веб-сервер ESP32 представляє апаратне забезпечення та мікропрограму ESP32. Апаратне забезпечення веб-сервера ESP32 зберігає/надає програмне забезпечення веб-сервера та необхідні файли (наприклад, сторінку HTML, таблицю стилів CSS, сценарії Java, шрифти, відео, аудіо, зображення тощо) «веб-клієнту» за запитом.

Сервер HTTP — це програмне забезпечення, яке розуміє URL-адреси (веб-адреси) і протокол HTTP. Веб-браузер використовує протокол HTTP для перегляду веб-сторінок.

Доступ до HTTP-сервера можна отримати через доменне ім’я або IP-адресу. Сервер HTTP зберігає та доставляє вміст цих розміщених веб-сайтів на пристрій кінцевого користувача (настільний ПК, ноутбук, планшет, мобільний телефон тощо)

«Статичний веб-сервер» надсилає розміщені веб-сторінки/файли HTML «веб-клієнту».

«Динамічний веб-сервер» оновлює розміщені веб-сторінки/файли HTML, а потім надсилає до «веб-клієнта».

Зазвичай ESP32 є «динамічним веб-сервером», який оновлює веб-сторінки/файли HTML даними підключених датчиків у реальному часі (наприклад, температура, вологість, тиск, ключова подія, GPIO, ADC, DAC, системні параметри тощо), а потім надсилає до «веб-клієнта» [18].

«Веб-клієнт» — це не що інше, як веб-браузер (Google Chrome, Microsoft Edge, Firefox, Safari, Internet Explorer, Opera тощо)

Веб-сервер підтримує протокол передачі гіпертексту (HTTP), є «текстовим» (усі команди є простим текстом, зрозумілим людині), «без стану» (ні сервер, ні клієнт не запам’ятовують попередні зв’язки) протоколу.

Зазвичай «веб-клієнти» надсилають запит HTTP до «веб-сервера», а потім «веб-сервер» відповідає на запит (рис. 2.1).

«Веб-сервер» також може заповнювати дані у «веб-клієнті» за допомогою механізму сервера.

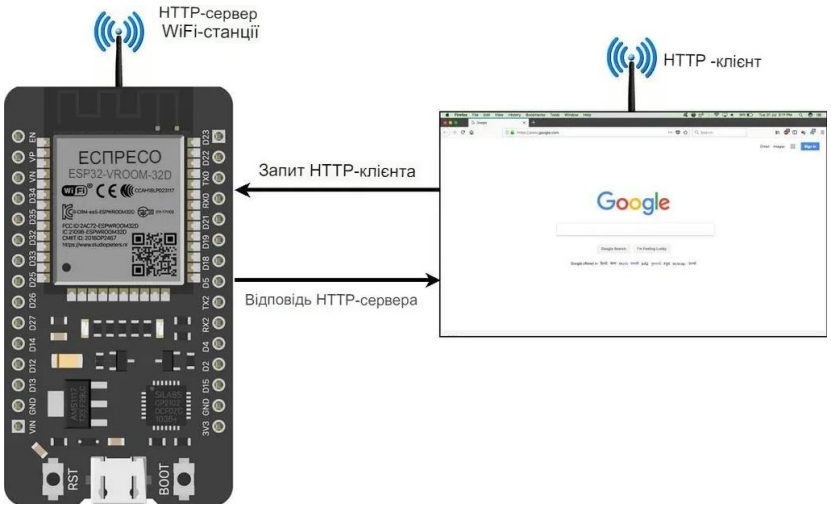


Рис. 2.1. Взаємодія Веб-сервера з Веб-інтерфейсом

2.2.2 HTML

HTML — це мова розмітки, яка визначає структуру вашого вмісту. HTML складається з серії елементів, які ви використовуєте для включення або обгортання різних частин вмісту, щоб він виглядав певним чином або діяв певним чином. Об’ємні теги можуть створювати гіперпосилання на слово чи зображення в іншому місці, виділяти слова курсивом, збільшувати або зменшувати шрифт тощо [19].

Основні частини елемента такі:

- *Початковий тег*: складається з назви елемента, укладеної у відкриваючі та закриваючі кутові дужки. Це вказує, де елемент починається або починає діяти .
- *Закриваючий тег*: це те саме, що відкриває тег, за винятком того, що він містить косу риску перед назвою елемента. Це означає, де закінчується елемент .
- *Вміст*: це вміст елемента, який у даному випадку є лише текстом.

- *Елемент*: відкриваючий тег, закриваючий тег і вміст разом складають елемент.

Елементи також можуть мати такі атрибути:

Атрибути містять додаткову інформацію про елемент, яку ви не бажаєте відображати у фактичному вмісті. Тут class — це назва атрибута, а editor-note — значення атрибута. Атрибут class дозволяє надати елементу неунікальний ідентифікатор, який можна використовувати для націлювання на нього (і будь-які інші елементи з таким же значенням class) з інформацією про стиль та іншими речами. Деякі атрибути не мають значення, наприклад required.

Атрибути, які встановлюють значення, завжди мають:

- Пробіл між ним і назвою елемента (або попереднім атрибутом, якщо елемент уже має один або кілька атрибутів).
- Ім’я атрибута, після якого стоїть знак рівності.
- Значення атрибута, укладене в лапки, що відкривають і закривають.

Є три теги HTML, необхідні для кожної сторінки. Це <html>, <head> і <body>. Щоб вказати веб-сторінці, що ви використовуєте HTML, кожна сторінка відкриватиметься за допомогою <html> і закриватиметься за допомогою </html>. Тег <head> містить метадані, не видимі на сторінці, але важливі для функціональності, а тег <body> вказує, де знаходиться основний вміст сторінки. Кожен із цих необхідних елементів можна використовувати лише один раз на сторінці. Нижче на рисунку 2.2 наведено базову структуру звичайних тегів HTML [19].

```
<!DOCTYPE html>

<html lang="en">
    <head>
        <meta
            http-equiv="Content-Type"
            content="text/html;
            charset=UTF-8"
            />
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <meta name="viewport" content="width=device-width,initial-scale=1" />
        <title>grblesp</title>
```

Рис. 2.2.1. Приклад використання базових тегів

```
<body>
        <div id="loadingmsg" style="display: none">
            <center><h3>Loading....</h3></center>
        </div>
        <div id="main_ui" class="container-fluid hide_it" style="display:
            block">
            <nav
                class="navbar navbar-default navbar-static-top"
                role="navigation"
                style="margin-bottom: 0; padding-right: 2em">
                <div class="navbar-header">
                    <span class="navbar-brand"><span translate=""
                            english_content="ESP3D for">Web-GRBL для</span>
                        <span id="fwName">ESP32</span>
                    </span>
                </div>
```

Рис. 2.2. Приклад використання базових тегів

Загальні елементи HTML

Інші несуттєві елементи HTML можна використовувати кілька разів на веб-сторінці, щоб визначити її структуру та підкреслити певні частини тексту. Загальні елементи HTML перелічені нижче та зображені на рисунку 2.3:

- , щоб додати курсив/виділення
- <head> мета-текст не відображається на веб-сторінці
- <header> визначає заголовок сторінки або розділу веб-сторінки
- <body> тіло сторінки
-
, щоб вставити розрив рядка
- <audio> вбудовування звукового вмісту
- <video> вбудовування відеовмісту
- <button> для використання кнопок, які можна натиснути
- , щоб визначити/поділити розділ
- <img> для вставки зображень
- для представлення списку
- , щоб визначити впорядкований список
- , щоб визначити невпорядкований список
- <u>, щоб підкреслити текст

Ще одна поширена функція HTML – це вбудовування CSS, Javascript або PHP. Ці файли створюють динамічні веб-сторінки, які додають яскравості та інтерактивності базовому скелету HTML [20].

```
<div class="panel panel-default">
                        <div class="panel-heading">
                            <div class="checkbox">
                                <label><input
                                        type="checkbox"
                                        id="show_commands_panel"
                                        onclick="prefs_toggledisplay(&#39;show_commands_panel&#39;)"
                                        />
                                    <span
                                        translate=""
                                        english_content="Show
                                        commands panel">Показати панель команд</span></label>
                            </div>
                        </div>
                        <div class="panel-body" id="cmd_preferences">
                            <div class="checkbox">
                                <label><input type="checkbox"
                                        id="preferences_autoscroll" />
                                    <span translate=""
                                        english_content="Autoscroll">Автопрокрутка</span></label>
                            </div>
                            <div id="verbose_mode_prefs_check" class="checkbox">
                                <label><input type="checkbox"
                                        id="preferences_verbose_mode" />
                                    <span
                                        translate=""
                                        english_content="Verbose
                                        Mode">Verbose Mode</span></label>
                            </div>
                        </div>
                    </div>
                </div>
                <div class="modal-footer">
                    <div class="pull-left" id="preferencesdlg_upload_msg">
                        <span translate="" english_content="Saving">Збереження</span>
                         <progress
                            name="prg"
                            id="preferencesdlg_prg"
                            max="100"></progress>  <span
                            id="preferencesdlg_upload_percent">0</span>%
                    </div>
                    <span class="pull-right">
                      <button
                            class="btn btn-warning"
                            onclick="closePreferencesDialog()"
                            translate=""
                            english_content="Cancel">
                            Скасувати
                        </button> </span><span class="pull-right">  </span>
                    <span class="pull-right">
                      <button
                            class="btn btn-primary"
                            onclick="SavePreferences()"
                            translate=""
                            english_content="Save">
                            Зберегти
                        </button></span>
                </div>
            </div>
```

Рис. 2.3. Приклад використання загальних тегів

2.2.3 CSS та JavaScript

CSS (Cascading Style Sheets) — це код, який стилізує веб-вміст. Основи CSS пояснюють, що вам потрібно для початку. Як і HTML, CSS не є мовою програмування. Це також не мова розмітки. CSS — це мова таблиць стилів. CSS — це те, що ви використовуєте для вибіркового стилізації елементів HTML.

Вся структура називається набором правил. (Термін набір правил часто називають просто правилом.) Назви окремих частин:

Селектор

Це назва елемента HTML на початку набору правил. Він визначає елемент (елементи), до якого потрібно стилізувати (у цьому прикладі елементи <p>). Щоб стилізувати інший елемент, змініть селектор.

Декларація

Це єдине правило, як color: red;. Він визначає, до яких властивостей елемента потрібно додати стиль.

Властивості

Це способи, за допомогою яких можна стилізувати елемент HTML. (У цьому прикладі color є властивістю елементів <p>.) У CSS ви вибираєте, на які властивості ви хочете вплинути в правилі.

Вартість майна

Праворуч від властивості — після двокрапки — вказано значення властивості. Це вибирає один із багатьох можливих виглядів для даної властивості. (Наприклад, крім red існує багато значень color.)

Інші важливі частини синтаксису:

Окрім селектора, кожен набір правил має бути укладено у фігурні дужки. ({})[21]

У кожній декларації ви повинні використовувати двокрапку (:), щоб відокремити властивість від її значення або значень.

У кожному наборі правил ви повинні використовувати крапку з комою (;), щоб відокремити кожне оголошення від наступного.

JavaScript — це мова програмування, яка додає інтерактивності сайту. Це відбувається у поведінці відповідей при натисканні кнопок або при введенні даних у формах; з динамічним стилем; з анімацією тощо[22].

Сам JavaScript відносно компактний, але дуже гнучкий. Розробники створили різноманітні інструменти на основі основної мови JavaScript, щоб розблокувати величезну кількість функціональних можливостей з мінімальними зусиллями. До них належать:

- Програмні інтерфейси браузера (API), вбудовані у веб-браузери, що забезпечують такі функції, як динамічне створення HTML і встановлення стилів CSS; збирання та маніпулювання відеопотоком із веб-камери користувача або генерування 3D-графіки та зразків аудіо.

- Сторонні API, які дозволяють розробникам включати функціональність у сайти інших постачальників вмісту, наприклад Twitter або Facebook.
- Фреймворки та бібліотеки сторонніх розробників, які можна застосувати до HTML, щоб прискорити роботу зі створення сайтів і програм[23].

2.3 аналіз та проектування серверної частини ESP32

2.3.1 аналіз роботи сервера на ESP32

ESP32 являє собою мікроконтролер з вбудованими модулями Wi-Fi і Bluetooth. Дуже простий у використанні, він легкий і має обсяг пам'яті і обчислювальних потужносте більше, ніж у Arduino. Це робить його ідеальною дошкою для навчання програмування, розробки підключених об'єктів або серверів.

Основою для створення підключених об'єктів є підключення їх до мережі, такий як Wi-Fi. Бездротовий передавач та антена, вбудований у мікроконтролер, що дозволяють підключатися до Інтернету. Завдяки цьому можна створити сервер, на якому розміщено веб-інтерфейс для віддаленого керування мікроконтролером ESP32 [24].

Як правило, веб-сервер на базі ESP32 в локальній мережі виглядає приблизно так: це ESP32, що працює в режимі веб-сервера і підключений до роутера по WiFi. Комп'ютер, смартфон або планшет теж підключені до роутера (через Wi-fi або Ethernet-кабель). Таким чином, ESP32 і пристрої, що знаходяться в одній і тій же мережі (рис. 2.4).

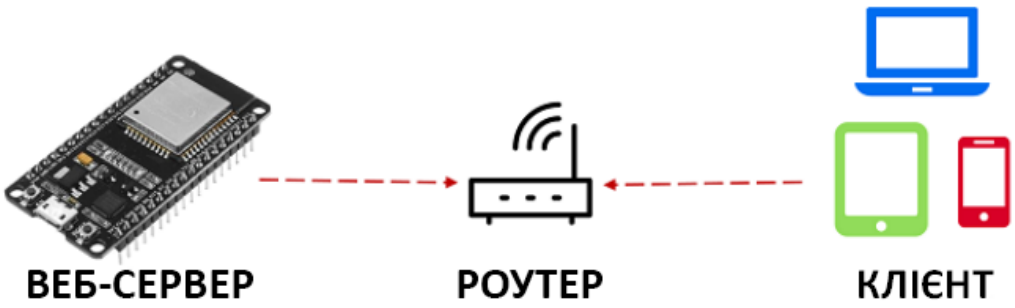


Рис. 2.4. Мережа з ESP32 та клієнтським пристроєм

При введенні IP-адресу ESP32 надсилається HTTP-запит. Після цього ESP32, як правило, відповідає за допомогою повідомлення-відповіді, що містить значення з датчика, і HTML-код, який потрібно показати на веб-сторінці, або щось інше запрограмоване (рис. 2.5).

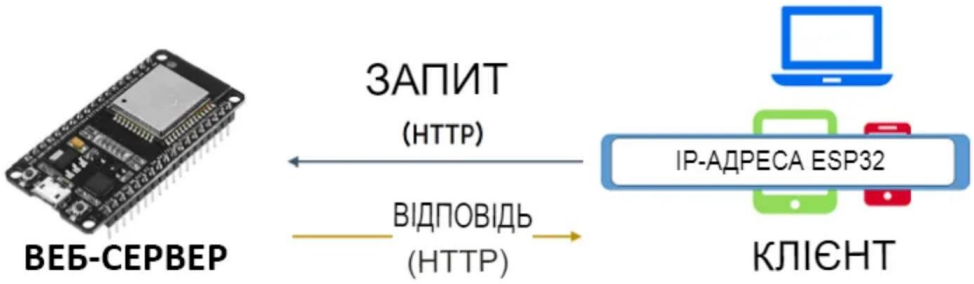


Рис. 2.5. Запит – відповідь клієнт-сервера

HTTP - сервер

Компонент HTTP Server надає можливість полегшеного запуску веб-сервера на ESP32. Нижче наведені докладні інструкції по використанню API, що надається HTTP-сервером:

- httpd_start(): Створює екземпляр HTTP-сервера, виділяє для нього пам'ять / ресурси в залежності від вказаної конфігурації і виводить дескриптор для екземпляра сервера. Сервер має як прослуховування сокет (TCP) для HTTP-трафіку, так і керуючий сокет (UDP) для керуючих сигналів, які вибираються циклічно в циклі виконання завдань сервера. Пріоритет завдання і розмір стека настроюються при створенні екземпляра сервера шляхом передачі структури httpd_config_t в httpd_start(). Трафік TCP аналізується як HTTP-запити, і, в залежності від запитаного URI, викликаються зареєстровані користувачем обробники, які повинні відправляти назад пакети відповідей HTTP [25].
- httpd_stop(): Це зупиняє сервер з наданими дескриптором і звільняє всю пов'язану пам'ять / ресурси. Це блокує функція, яка спочатку сигналізує про зупинення серверної завдання, а потім очікує завершення завдання. Під час зупинки завдання закриє всі відкриті з'єднання, видалить зареєстровані обробник URI і скине всі дані контексту сеансу на порожні.

- httpd_register_uri_handler(): Обробник URI реєструється шляхом передачі об'єкта типу httpd_uri_tstructure, який має елементи, включаючи uriім'я, methodтип (наприкладHTTPD_GET/HTTPD_POST/HTTPD_PUT, і т. д.), покажчик на функцію типу esp_err_t *handler (httpd_req_t *req)i user_ctx покажчик на дані користувача контексту [26].

Нище на рисунку 2.6. приведено приклад коду серверної частини МК ESP32.

```
bool Web_Server::begin(){

    bool no_error = true;
    _setupdone = false;
    Preferences prefs;
    prefs.begin(NAMESPACE, true);
    int8_t penabled = prefs.getChar(HTTP_ENABLE_ENTRY, DEFAULT_HTTP_STATE);
    _port = prefs.getUShort(HTTP_PORT_ENTRY, DEFAULT_WEBSERVER_PORT);
    prefs.end();
    if (penabled == 0) return false;
    _webserver= new WebServer(_port);
#ifdef ENABLE_AUTHENTICATION
    const char * headerkeys[] = {"Cookie"} ;
    size_t headerkeyssize = sizeof (headerkeys) / sizeof (char*);
    _webserver->collectHeaders (headerkeys, headerkeyssize );
#endif
    _socket_server = new WebSocketsServer(_port + 1);
    _socket_server->begin();
    _socket_server->onEvent(handle_Websocket_Event);
     Serial2Socket.attachWS(_socket_server);
    _webserver->on("/",HTTP_ANY, handle_root);
    _webserver->onNotFound (handle_not_found);
    _webserver->on("/login", HTTP_ANY, handle_login);
    _webserver->on ("/command", HTTP_ANY, handle_web_command);
    _webserver->on ("/command_silent", HTTP_ANY, handle_web_command_silent);
    _webserver->on ("/files", HTTP_ANY, handleFileList, SPIFFSFileupload);
    _webserver->on ("/updatefw", HTTP_ANY, handleUpdate, WebUpdateUpload);
```

Рис. 2.6. Фрагмент коду серверної частини ESP32

ESP32 може використовуватись як режим клієнта (станції) (STA) так і точки доступу (AP).

Режим станції (STA).

ESP32 Коли модуль ESP8266 підключений до доступної мережі Wi-Fi (тобто маршрутизатора, точки бездротового доступу, яку ви створили), до якої потрібно підключитися, і її спосіб, повідомила станція. (STA).

У режимі STA ESP32 отримує IP від маршрутизатора, якого він підключений [27].

У режимі STA ESP32 отримує IP від маршрутизатора, якого він підключений. З цим IP-адресом він може настроїти веб-сервер і віддавати веб-сторінки на всі підключені пристрої в існуючій мережі Wi-Fi (рис. 2.7).

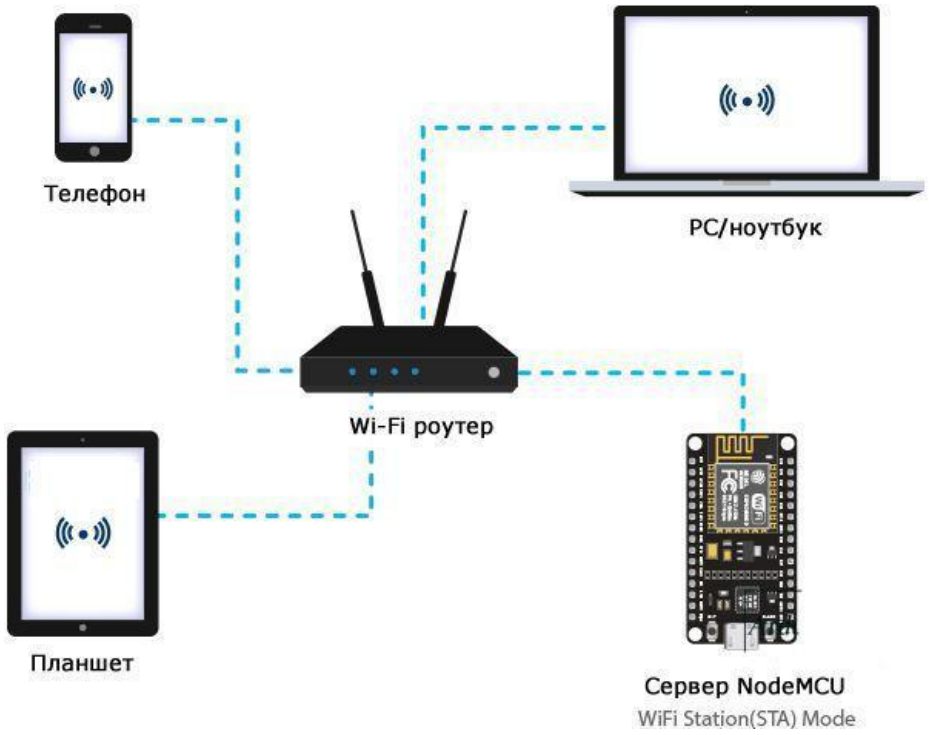


Рис. 2.7. Режим клієнта ESP32

Режим точки доступу (AP).

ESP32 (ESP8266), який створює власну мережу Wi-Fi і діє як концентратор (як Wi-Fi маршрутизатор) для одного або декількох пристроїв, називається точкою доступу (AP). На відміну від Wi-Fi роутера, він не має інтерфейсу для провідної мережі. Отже, такий режим роботи має назву Soft Access Point (soft-AP). Також максимальна кількість пристроїв, які можуть підключитись до нього, обмежена п'ятьма [28].

ESP32, який створює власну мережу Wi-Fi і діє як концентратор (як Wi-Fi маршрутизатор)

У режимі AP ESP32 створює нову мережу Wi-Fi та встановлює для неї SSID (ім'я мережі) та IP-адресу. З цим IP-адресом він може віддавати веб-сторінки на всі підключені пристрої у власній мережі (рис. 2.8).

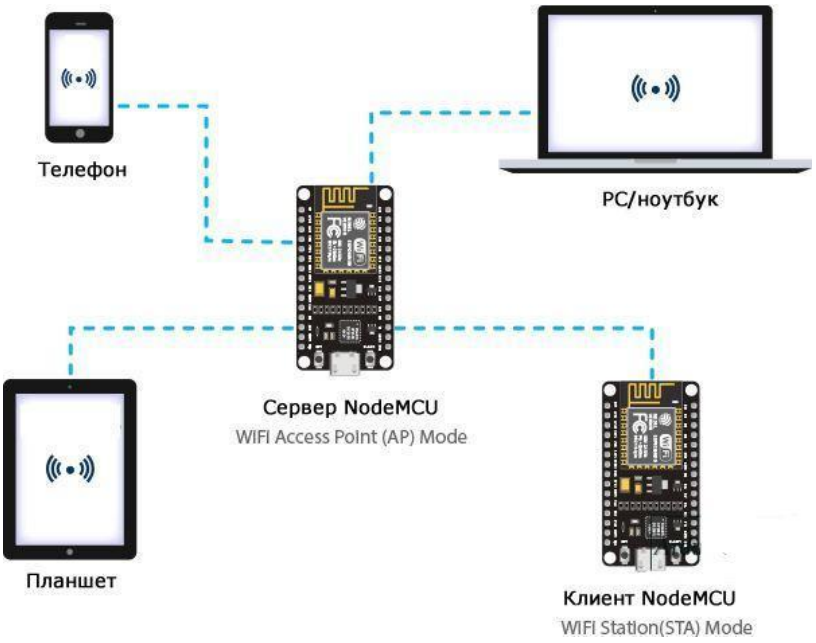


Рис. 2.8. Режим точки доступу ESP32

2.3.2 проектування серверної частини ESP32

Отже, сам мікроконтролер ESP32 програмується на мові C++, і для спрощення роботи веб-сервера на його базі для самого сервера також була обрана мова C++. Для початку роботи, в коді потрібно включити необхідні для подальшої роботи файли та бібліотеки (рис 2.9).

Далі назначаються змінні, ініціалізується сервер «_socket_server->begin();», та програмно включається підтримка веб-сервера в МК ESP32 «_webserver->on» (рис 2.10).

Також нам для роботи і зберігання файлів з G-кодом необхідний вільний простір, фізично додано можливість підключення SD-карти, то ж була зроблена і програмна реалізація включення SD-карти. Також налаштуємо режим станції (STA) (рис 2.11).

У корні (root) веб-сервера виконується операція для екземпляру класу String, що містить відомості про шлях до файлу динамічного веб-інтерфейсу String path = "/index.html";, який буде розглядатися в наступному розділі. Ця операція виконуються міжплатформенним способом. З цього ж файлу берем заголовок contentType який повідомляє, який буде тип переданого контенту.

Також створюється умова при наявності "/index.html"; буде кореневою сторінкою за замовчуванням (рис 2.12).

```
#include "config.h"

#if defined (ENABLE_WIFI) &&  defined (ENABLE_HTTP)

#include "wifiservices.h"

#include "grbl.h"

#include "commands.h"
#include "espresponse.h"
#include "serial2socket.h"
#include "web_server.h"
#include <WebSocketsServer.h>
#include "wificonfig.h"
#include <WiFi.h>
#include <FS.h>
#include <SPIFFS.h>
#ifdef ENABLE_SD_CARD
#include <SD.h>
#include "grbl_sd.h"
#endif
#include <Preferences.h>
#include "report.h"
#include <WebServer.h>
#include <ESP32SSDP.h>
#include <StreamString.h>
#include <Update.h>
#include <esp_wifi_types.h>
#ifdef ENABLE_MDNS
#include <ESPmDNS.h>
#endif
#ifdef ENABLE_SSDP
#include <ESP32SSDP.h>
#endif
#ifdef ENABLE_CAPTIVE_PORTAL
#include <DNSServer.h>
const byte DNS_PORT = 53;
DNSServer dnsServer;
#endif
```

Рис. 2.9. Включення бібліотек та необхідних файлів

```
Web_Server web_server;
bool Web_Server::_setupdone = false;
uint16_t Web_Server::_port = 0;
long Web_Server::_id_connection = 0;
uint8_t Web_Server::_upload_status = UPLOAD_STATUS_NONE;
WebServer * Web_Server::_webserver = NULL;
WebSocketsServer * Web_Server::_socket_server = NULL;
#ifdef ENABLE_AUTHENTICATION
auth_ip * Web_Server::_head = NULL;
uint8_t Web_Server::_nb_ip = 0;
#define MAX_AUTH_IP 10
#endif
Web_Server::Web_Server(){

}
Web_Server::~Web_Server(){
    end();
}

long Web_Server::get_client_ID() {
    return  _id_connection;
}

bool Web_Server::begin(){

    bool no_error = true;
    _setupdone = false;
    Preferences prefs;
    prefs.begin(NAMESPACE, true);
    int8_t penabled = prefs.getChar(HTTP_ENABLE_ENTRY, DEFAULT_HTTP_STATE);
    _port = prefs.getUShort(HTTP_PORT_ENTRY, DEFAULT_WEBSERVER_PORT);
    prefs.end();
    if (penabled == 0) return false;
    _webserver= new WebServer(_port);
#ifdef ENABLE_AUTHENTICATION
    const char * headerkeys[] = {"Cookie"} ;
    size_t headerkeyssize = sizeof (headerkeys) / sizeof (char*);
    _webserver->collectHeaders (headerkeys, headerkeyssize );
#endif
    _socket_server = new WebSocketsServer(_port + 1);
    _socket_server->begin();
    _socket_server->onEvent(handle_Websocket_Event);
     Serial2Socket.attachWS(_socket_server);
    _webserver->on("/",HTTP_ANY, handle_root);
    _webserver->onNotFound (handle_not_found);
    _webserver->on("/login", HTTP_ANY, handle_login);
    _webserver->on ("/command", HTTP_ANY, handle_web_command);
    _webserver->on ("/command_silent", HTTP_ANY, handle_web_command_silent);
    _webserver->on ("/files", HTTP_ANY, handleFileList, SPIFFSFileupload);
    _webserver->on ("/updatefw", HTTP_ANY, handleUpdate, WebUpdateUpload);
```

Рис. 2.10. Ініціалізація сервера

```
#ifdef ENABLE_SD_CARD
    //Direct SD management
    _webserver->on("/upload", HTTP_ANY, handle_direct_SDFileList,SDFile_direct_upload);
    //_webserver->on("/SD", HTTP_ANY, handle_SDCARD);
#endif

#ifdef ENABLE_CAPTIVE_PORTAL
     if(WiFi.getMode() != WIFI_STA){
        // if DNSServer is started with "*" for domain name, it will reply with
        // provided IP to all DNS request
        dnsServer.start(DNS_PORT, "*", WiFi.softAPIP());
        grbl_send(CLIENT_ALL,"[MSG:Captive Portal Started]\r\n");
        _webserver->on ("/generate_204", HTTP_ANY,  handle_root);
        _webserver->on ("/gconnectivitycheck.gstatic.com", HTTP_ANY, handle_root);
        //do not forget the / at the end
        _webserver->on ("/fwlink/", HTTP_ANY, handle_root);
     }
#endif
```

Рис. 2.11. Включення SD-карти та режим станції (STA)

```
void Web_Server::handle_root()
{
    String path = "/index.html";
    String contentType =  getContentType(path);
    String pathWithGz = path + ".gz";
    //if have a index.html or gzip version this is default root page
    if((SPIFFS.exists(pathWithGz) || SPIFFS.exists(path)) && !_webserver->hasArg("forcefallback") && _webserver->arg("forcefallback")!="yes") {
        if(SPIFFS.exists(pathWithGz)) {
            path = pathWithGz;
        }
        File file = SPIFFS.open(path, FILE_READ);
        _webserver->streamFile(file, contentType);
        file.close();
        return;
    }
    //if no lets launch the default content
    _webserver->sendHeader("Content-Encoding", "gzip");
    _webserver->send_P(200,"text/html",PAGE_NOFILES,PAGE_NOFILES_SIZE);
}
```

Рис. 2.12. Корінь (root) веб-сервера

Прописуємо функції обробки запитів веб-команд та відправка відповідей назад (рис 2.13). Ці функції надають змогу “спілкуватись” через веб-інтерфейс з веб-сервером і функціями самого МК ESP32.

```
void Web_Server::handle_web_command ()
{
    level_authenticate_type auth_level = is_authenticated();
    String cmd = "";
    if (_webserver->hasArg ("plain") || _webserver->hasArg ("commandText") ) {
        if (_webserver->hasArg ("plain") ) {
            cmd = _webserver->arg ("plain");
        } else {
            cmd = _webserver->arg ("commandText");
        }
    } else {
        _webserver->send (200, "text/plain", "Invalid command");
        return;
    }
    //if it is internal command [ESP32]<parameter>
    cmd.trim();
    int ESPpos = cmd.indexOf ("[ESP");
    if (ESPpos > -1) {
        //is there the second part?
        int ESPpos2 = cmd.indexOf ("]", ESPpos);
        if (ESPpos2 > -1) {
            //Split in command and parameters
            String cmd_part1 = cmd.substring (ESPpos + 4, ESPpos2);
            String cmd_part2 = "";
            //only [ESP800] is allowed login free if authentication is enabled
            if ( (auth_level == LEVEL_GUEST)  && (cmd_part1.toInt() != 800) ) {
                _webserver->send (401, "text/plain", "Authentication failed!\n");
                return;
            }
            //is there space for parameters?
            if (ESPpos2 < cmd.length() ) {
                cmd_part2 = cmd.substring (ESPpos2 + 1);
            }
            //if command is a valid number then execute command
            if (cmd_part1.toInt() != 0) {
                ESPResponseStream espresponse(_webserver);
                //commmand is web only
                COMMANDS::execute_internal_command (cmd_part1.toInt(), cmd_part2, auth_level, &espresponse);
                //flush
                espresponse.flush();
            }
            //if not is not a valid [ESP32] command
        }
    } else { //execute GCODE
        if (auth_level == LEVEL_GUEST) {
            _webserver->send (401, "text/plain", "Authentication failed!\n");
            return;
        }
        //Instead of send several commands one by one by web  / send full set and split here
        String scmd;
        String res = "";
        uint8_t sindex = 0;
        scmd = get_Splited_Value(cmd,'\n', sindex);
        while ( scmd != "" ){
        if ((scmd.length() == 2) && (scmd[0] == 0xC2)){
              scmd[0]=scmd[1];
              scmd.remove(1,1);
            }
        if (scmd.length() > 1)scmd += "\n";
        else if (!is_realtime_cmd(scmd[0]) )scmd += "\n";
        if (!Serial2Socket.push(scmd.c_str()))res = "Error";
        sindex++;
        scmd = get_Splited_Value(cmd,'\n', sindex);
        }
        _webserver->send (200, "text/plain", res.c_str());
    }
}
```

Рис. 2.13. Обробка запитів веб-команд і надсилання відповідей

2.4. Висновок за розділом

Як результат роботи проведеної у другому розділі кваліфікаційної роботи було визначено основні набори інструментів, основи роботи певних мов програмування та властивості їх функцій, а також певні нюанси з їх роботою для практичної реалізації запланованих проектів. Проведено проектування серверної частини та взаємодії її з мікроконтролером. Як основну мову для мікроконтролера та серверної частини було обрано C++, для веб інтерфейсу HTML, CSS та JavaScript, для ПЗ під ОС Windows – C#, а для реалізації мобільного додатку Kotlin.

Отже, маємо проведену на довільному рівні роботу щодо аналізу кожної зі складових проекту по розробці ПЗ для різних платформ для керування післяпроцесної обробки адитивного виробництва. Та проектування серверної частини мікроконтролера ESP32. У наступному розділі буде наглядно показано використання цих результатів у ході розробки веб-інтерфейсу та програмних забезпечень.

РОЗДІЛ 3.
ПРАКТИЧНА ЧАСТИНА РОЗРОБКА ПРОГРАМНИХ ЗАБЕЗПЕЧЕННЬ

3.1 Розробка Веб-інтерфейсу

Першим етапом реалізації розробки програмних забезпечень є розробка веб-інтерфейсу. На базі якого будуть розроблятися наступні програмні забезпечення. Тому вважаю за потрібне описати процес реалізації веб-інтерфейсу починаючи з основного розділу і його складових.

Головним функціоналом веб-інтерфейсу є підготовка верстату до роботи, а саме встановлення початкової позиції робочого інструменту та включення необхідних функцій, і відправка файлу з G-кодом на флеш накопичувач, з якого мікроконтролер зчитує цей файл та виконує роботу відповідно до нього.

Сам веб-інтерфейс представлений на рисунку 3.1.

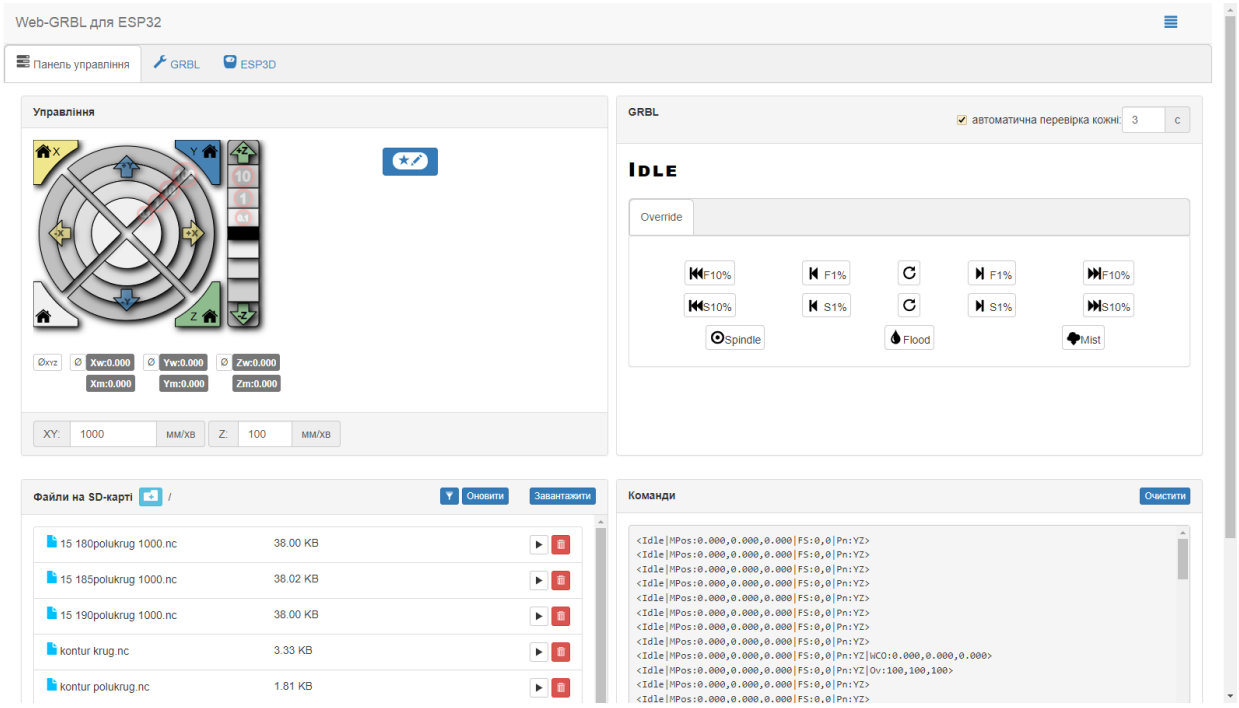


Рис. 3.1. Основна панель веб-інтерфейсу

Також необхідний додатковий функціонал для налаштування мікроконтролера, а саме мережеві налаштування ESP32 (рис 3.2) та налаштування конфігурацій GRBL (рис 3.3), опис яких є в першому розділі роботи.

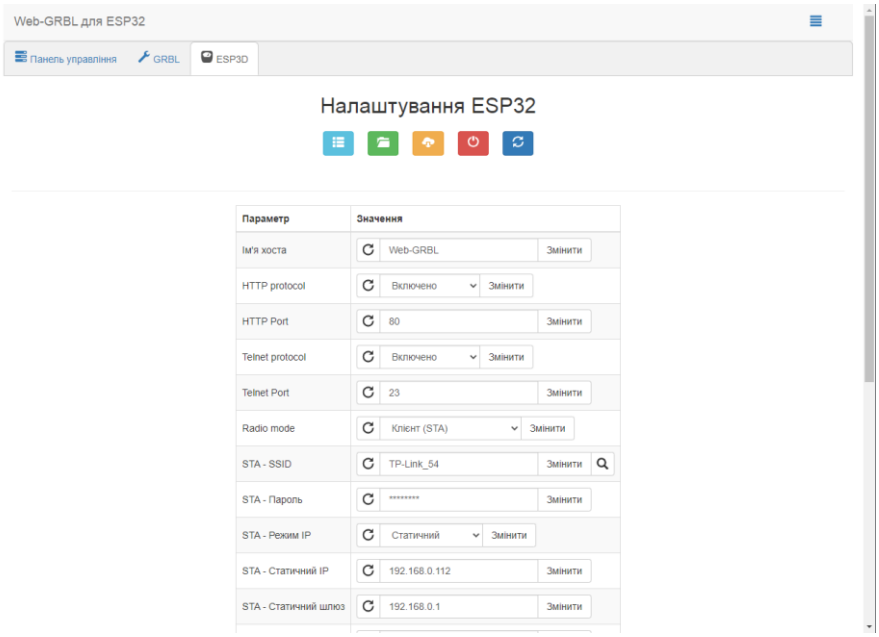


Рис. 3.2. Веб налаштування мікроконтролера

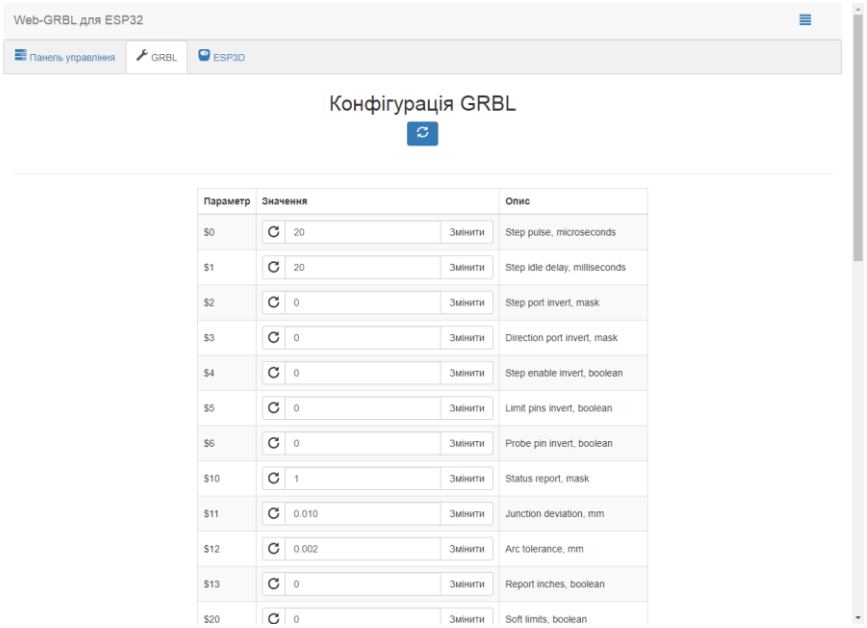


Рис. 3.3. GRBL конфігурації

У веб-інтерфейсу є верхня панель, яка дає можливість обирати між основною панеллю, налаштуванням GRBL конфігурацій та веб налаштуваннями. Кожен з варіантів відкривається як панель, а не як нова сторінка, що дає можливість кожного разу після вибору панелі не переобирати налаштування.

Основна панель розділена елементом поділу вмісту HTML ( ) для спрощення використання інтерфейсу, та розширення його можливостей. Блоки панелі представленні на рисунку 3.4.

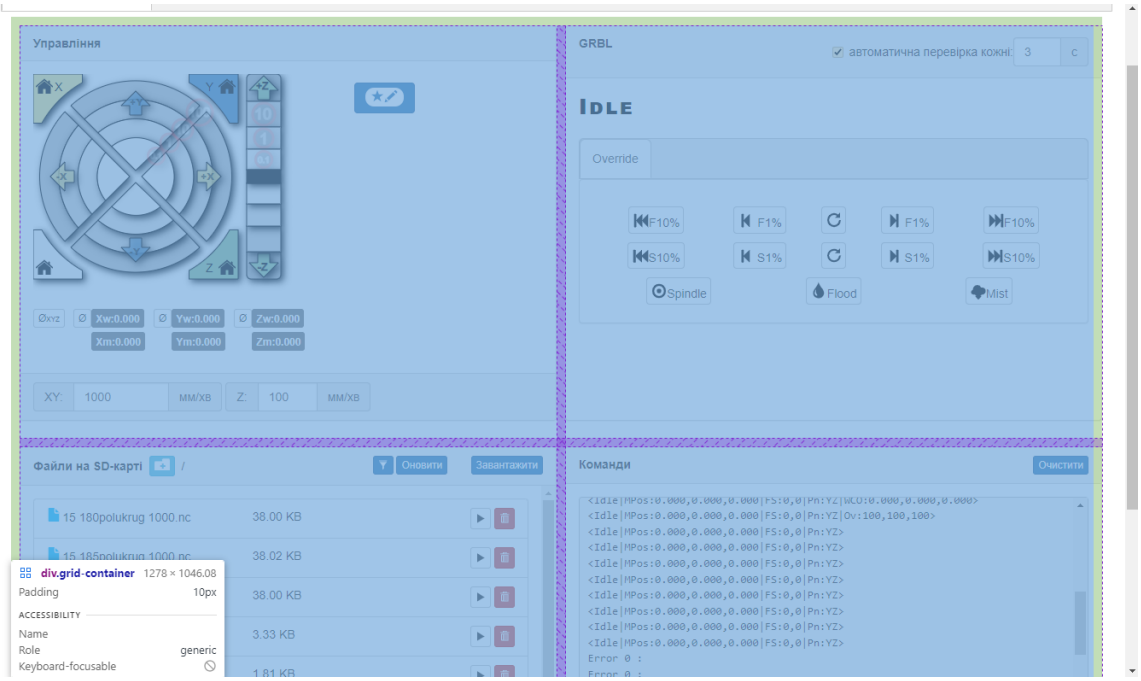


Рис. 3.4. Блоки основної панелі

І кожен блок має в собі ще поділені блоки (рис. 3.5) і так далі до кожного окремого елементу на сторінці, наприклад кнопка обнулення вісей на сторінці (рис. 3.6) або кнопка додавання файлів на флеш накопичувач (рис. 3.7 а) і кнопка запуску роботи файлу (рис 3.7 б).

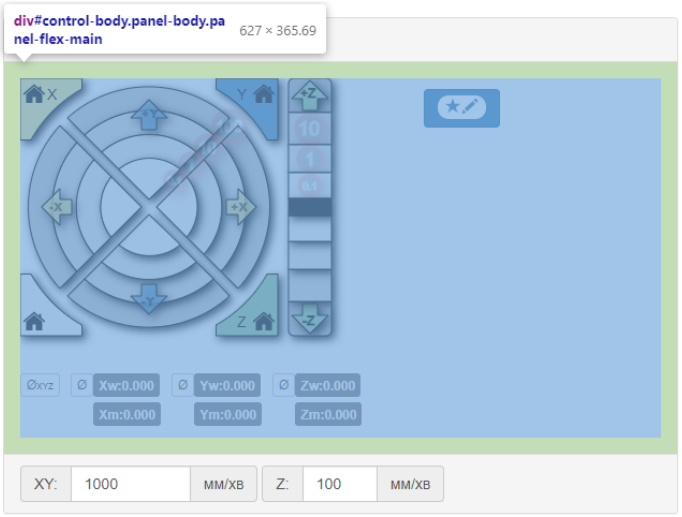


Рис. 3.5. Підблок основної панелі

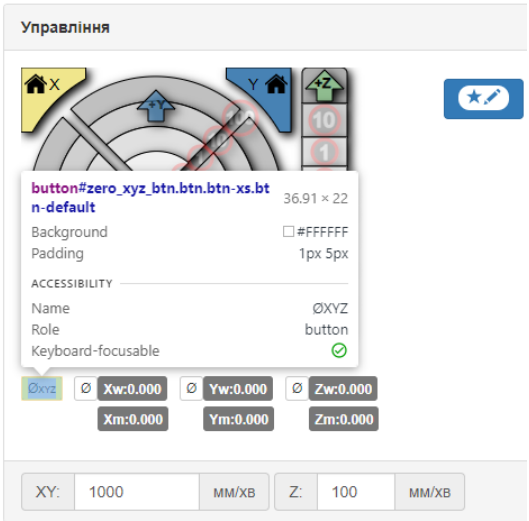
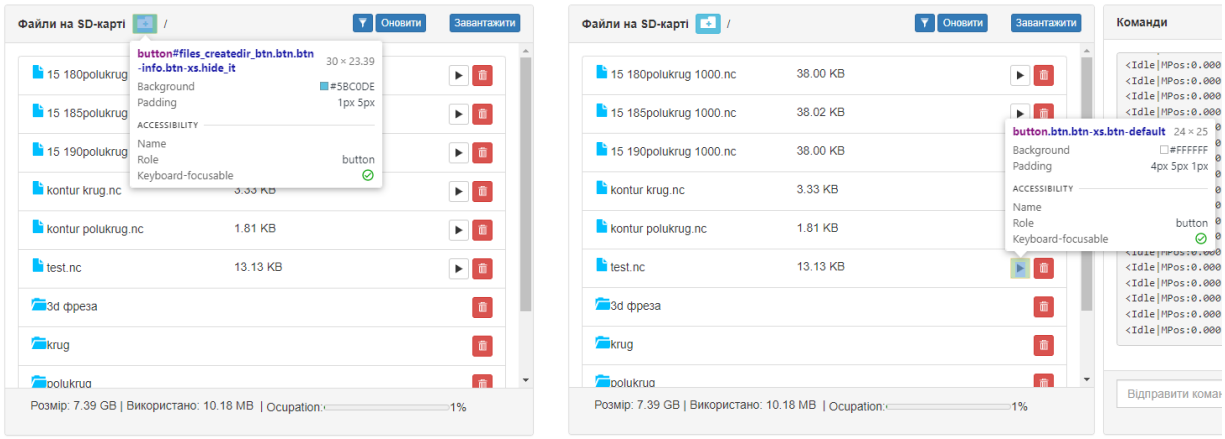


Рис. 3.6. Кнопка обнулення



а) б)

Рис. 3.7. Кнопка додавання файлів та запуск роботи з файлу

При використанні мікроконтролером режиму клієнта, в налаштуваннях необхідно встановити статичну IP-адресу (рис. 3.8.), для входу через веб інтерфейс і для підключення до мікроконтролера з мобільного додатку та програмного забезпечення, які будуть описані дальше.

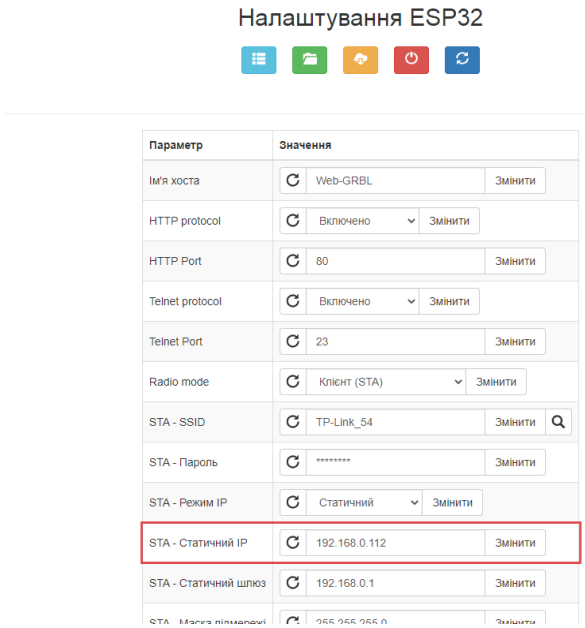


Рис. 3.8. Встановлення IP-адреси

Функціонал Веб-інтерфейсу обширніший від аналогічних програмних забезпечень для верстатів з ЧПУ на базі мікроконтролерів Arduino та ESP32. А саме: відправка, видалення та запуску робочих файлів з G-кодом; вбудоване веб налаштування мікроконтролера та налаштування GRBL конфігурацій; додавання своїй макросів для керування верстатом з ЧПУ; оновлення прошивки та веб-інтерфейсу через сам веб-інтерфейс.

3.2 Розробка мобільного додатку

3.2.1 Структура мобільного додатку

Для розповсюдження додатку та створення умов зручного користування виконаємо визначення цільової платформи для клієнтської частини. Найбільш поширена мобільна ОС в даний проміжок часу – це Android. Таким чином вона обрана у якості цільової платформи для реалізації мобільного додатку.

Відповідно до попередніх розділів проект побудований в ПЗ Android Studio. Будова проекту в Android Studio представляє собою набір модулів. За замовчуванням створюється базовий модуль. Кожний з них складається з двох базових блоків. Перший відповідає за класи, тобто логіку. Другий відповідає за ресурси та розмітку візуальних елементів. Після створення пустого проекту з використанням вбудованого механізму потрібно почати з визначенням структури директорій та архітектурним шаблоном проектування. Як результат код проекту буде розподілений на окремі складові для даних, графічного інтерфейсу, бізнес логіки та інші. Такий підхід надає розробнику зручну можливість розробки кожної складової проекту окремо. Також це добре тим, що такі структурні елементи можна в майбутньому замінити більш сучасною реалізацією. Головне організовувати залежності не на рівні реалізації, а на рівні протоколу. В даному випадку виділимо наступний список складових:

1. Ін’єкція залежностей.
2. Бізнес логіка.
3. Графічний інтерфейс.
4. Навігація.
5. Утиліти.

Частина проекту, що відповідає за файли ресурсів та розмітку не потребує втручання, адже жорстко визначається системою IDE та є незмінною.

Таким чином визначено, що основна структура проекту мобільного додатку створена на основі MVVM шаблону та складатися з вище перелічених частин.

3.2.2 Графічний інтерфейс мобільного додатку

З метою підключення до саме нашого мікроконтролера по мережі на початковому екрані встановлений контроль доступу, а саме для входу необхідно ввести IP-адресу даного мікроконтролера, яка встановлена у веб налаштуваннях. Тож для даного функціоналу спроектований елемент для введення адреси. У проектах під ОС Android стандартом розробки закладено, що на кожну задачу має бути створений окремий фрагмент. У нашому випадку одна задача. Для підключення наявні такі компоненти, як поле для введення тексту, кнопка підтвердження відправки даних на перевірку та індикатор прогресу. Використовуючи вбудовані засоби Android IDE, а саме xml розмітку було виконано проектування макету. Результат зображено на рисунку 3.9, для темної та світлої тем смартфона.

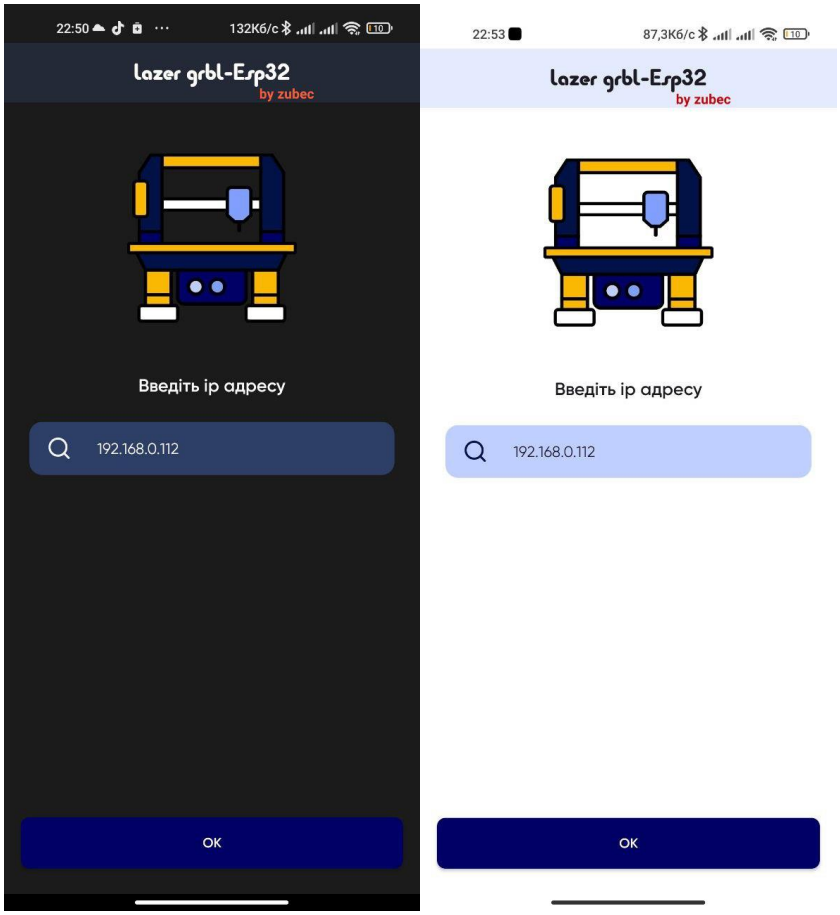
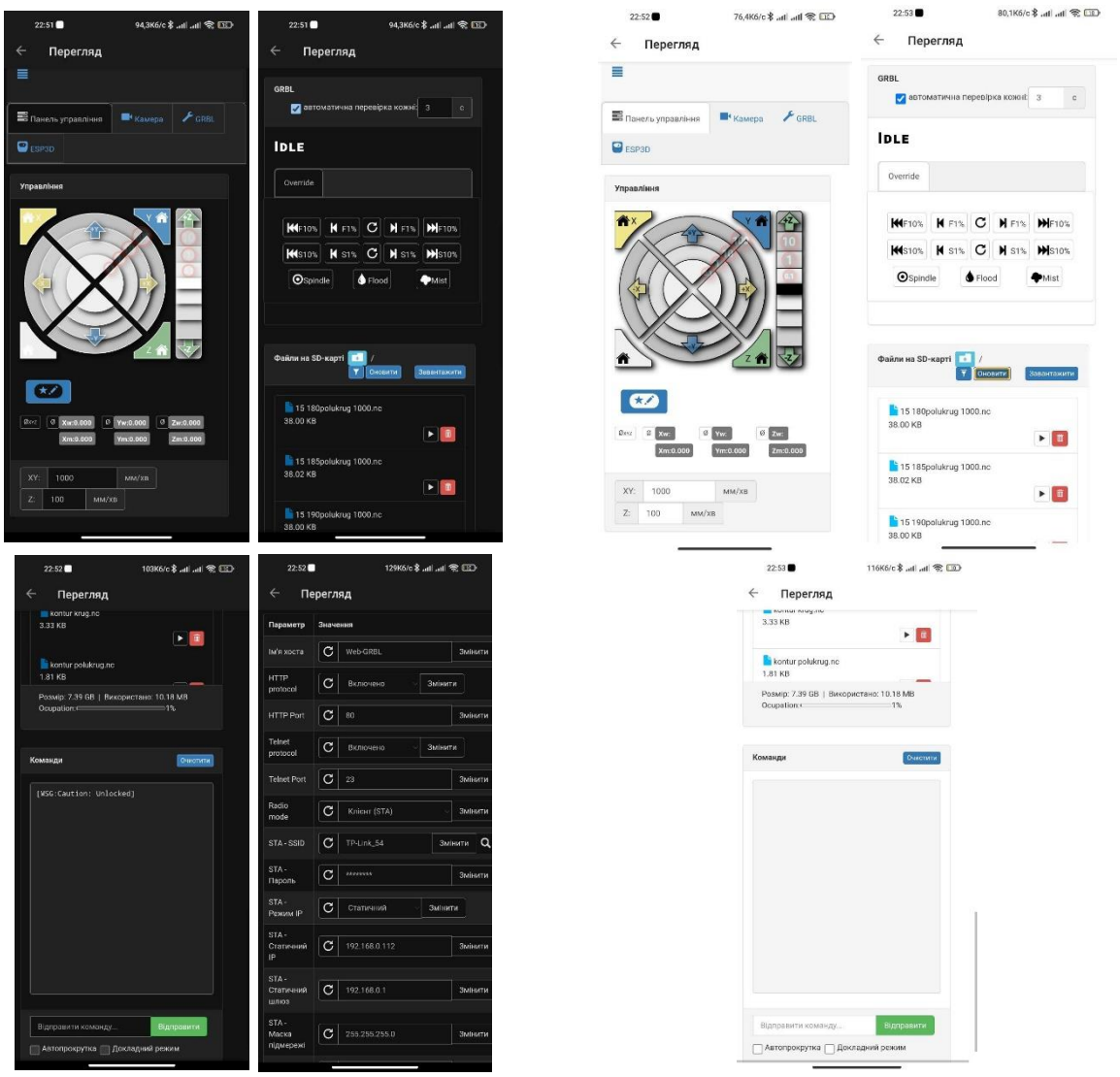


Рис. 3.9. Макет дизайну для підключення

Подальшим кроком є головний екран, який аналогічний до головної панелі веб-інтерфейсу. Базовою складовою тут є клас WebView з розширенням View класу Android. Таким чином клас розміщується на весь екран. Завдяки

цьому класу, є можливість повністю скопіювати функціонал веб-інтерфейсу, що надає простоту в переході між пристроями. На рисунку 3.10 зображено спроектований макет головного фрагмента для темної (3.10. а) та світлої (3.10. б) тем смартфона.



а) б)

Рис. 3.10. Макет головного екрану

3.2.3 Класи візуального елементу «фрагмет» та бізнес логіка «ViewModel»

Першим етапом реалізації мобільного додатку є впровадження базових класів. Мобільний додаток для ОС «Android» має три базові складові. Однією із них є графічний інтерфейс. Розпочнемо саме з нього.

Основна маса додатків будується з дотриманням підходу «Single Activity», що означає наявність єдиного класу для екрану з перемиканням візуальних фрагментів на ньому. Тож базовим компонентом графічного інтерфейсу є фрагмент. Він взаємодіє з класами бізнес логіки, зв'язування візуальних компонентів, обробником подій навігації. Тож вказуємо їх як «Generic» типи нашого фрагменту. Їх також необхідно ініціалізувати. Для цього клас зроблений абстрактним та виконано декларацію абстрактного методу ініціалізації для кожного з об'єктів. Далі виконується прив'язка панелі інструментів до компоненти навігації. На цьому імплементація базового фрагменту є закінченою. Було дотримано принципи SOLID та створено код який має залежність від протоколу, а не від реалізації. Таким чином клас легко впроваджувати у будь який інших проект. Структуру коду створеного класу зображено на рисунку 3.11.

```
abstract class BaseFragment<VM : BaseViewModel, VB : ViewBinding, NH : NavigationHandler> :
    Fragment() {

    lateinit var navController: NavController
    protected lateinit var binding: VB
    protected abstract val viewModel: VM
    protected abstract val navigationHandler: NH

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {...}

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {...}

    private fun setupNavigation() {...}


    protected fun hideKeyboard() {...}

    protected abstract fun provideViewBinding(): VB

    protected abstract fun init()
}
```

Рис. 3.11. Структура базового класу фрагмента

Шар бізнес логіки несе одну з найважливіших ролей в ПЗ. Адже саме тут виконується керування різного роду механізмами для роботи з даними. З метою повторного використання коду програмістами було розроблено

механізм ViewModel, що призначений для передачі команд в зону графічного інтерфейсу. У даний клас буде запроваджено 4 додаткові складові, а саме:

1. Механізм подій навігації.

2. Команди відображення повідомлення про завантаження.

3. Команди відображення повідомлень про помилку чи важливу інформацію.

Усі вказані елементи для команд реалізуються з використанням механізму повідомлень «MutableLiveData». Саме цей механізм містить у собі логіку передачі даних в головний потік. В основі цього механізму закладено шаблон проектування «Наглядач». Структуру коду імплементованої бази для ViewModel зображено на рисунку 3.12.

```
package com.zub.webviewexample.core.viewmodel

import ...

abstract class BaseViewModel : ViewModel() {

    protected val mNavigationEvent = SingleLiveEvent<NavigationEvent>()
    val navigationEvent: LiveData<NavigationEvent>
        get() = mNavigationEvent

    protected val mLoadingEvent = MutableLiveData<LoadingEvent>()
    val loadingEvent: LiveData<LoadingEvent>
        get() = mLoadingEvent

    protected fun showLoading() = mLoadingEvent.postValue(LoadingEvent.ShowLoading)

    protected fun hideLoading() = mLoadingEvent.postValue(LoadingEvent.HideLoading)

}
```

Рис. 3.12. Структура базової ViewModel

3.3 Розробка програмного забезпечення для персональних комп'ютерів

3.3.1 Структура програмного забезпечення

На даний момент найбільш поширена ОС для персональних комп'ютерів є Windows, актуальна версія якої є Windows 10. Та все ж ПЗ має містити сумісність зі старішими версіями ОС. То ж в ході розробки ПЗ це було враховано.

Відповідно до попередніх розділів проект був побудований в ПЗ Visual Studio. Будова проекту в Visual Studio представляє собою візуальний

конструктор (Windows Forms), що забезпечує один з самих ефективних способів створення програмних забезпечень. За замовчуванням створюється форма. Форма є користувацьким інтерфейсом Windows.

Зазвичай програма на основі конструктора Windows Forms будується шляхом додавання елементів керування до форм і створення коду для реагування на дії користувача, такі як клацання миші або натискання клавіш. До форми є можливість додавання необхідних для проекту елементів керування, наприклад: кнопки, поле введення, зображення, текстові поля, списки, що розкриваються і т.д.

При виконанні користувачем будь-якої дії з формою або одним з елементів керування створюється подія. Програма реагує на ці події, як задано в коді, та обробляє події при їх виникненні.

То ж створення ПЗ спрощується завдяки такому конструктору, та надає більше можливостей і функцій ПЗ за менший час розробки.

3.3.2 Графічний інтерфейс програмного забезпечення

Аналогічно до мобільного додатку, в ПЗ для ПК необхідний контроль доступу, тобто вхід по введеному IP-адресу мікроконтролера, який встановлений у веб налаштуваннях. Тож для даного функціоналу в першій формі були додані такі елементи керування, як поле для введення тексту, кнопка підтвердження дії, кнопка виходу та необхідні підказки. Також додано кілька візуальних елементів, текст та зображення. Результат зображено на рисунку 3.13.

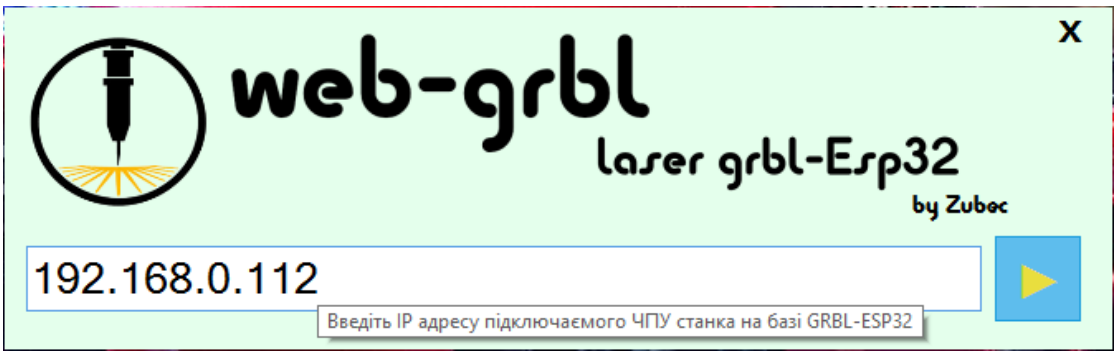


Рис. 3.13. Форма входу

Подальшим кроком є головний екран, а саме друга форма, яка аналогічна до головної панелі веб-інтерфейсу та головного екрану мобільного додатку. Основним елементом керування є WebView2. Завдяки цьому елементу керування, є можливість повністю скопіювати функціонал веб-інтерфейсу, що надає простоту в переході між пристроями. На рисунку 3.14 зображено спроектовану форму головного фрагменту.

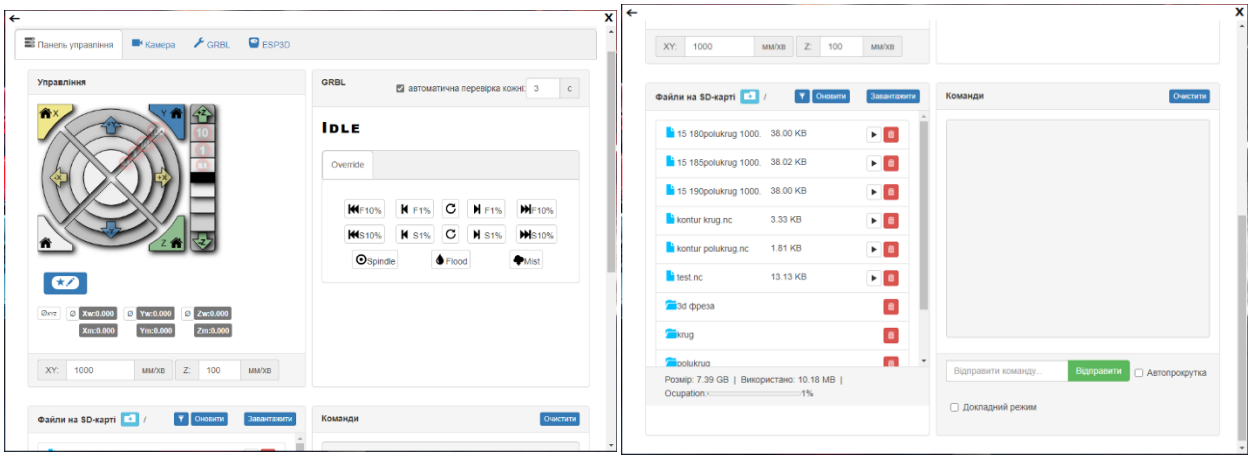


Рис. 3.14. Головна форма

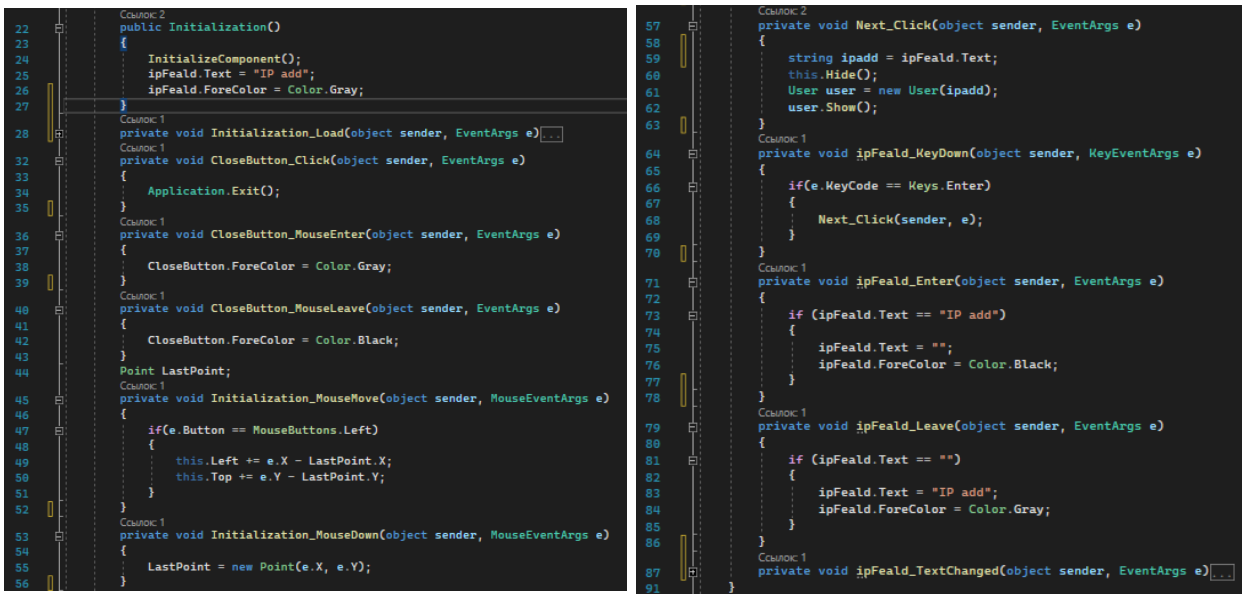
3.3.3 Створення подій для елементів керування

Для форма входу необхідні наступні події:

- Для натискання кнопки «далі» – перевірка введеної адреси та підключення по заданій адресі.
- Для натискання клавіші «Enter» – та ж сама дія що і для натискання кнопки.
- Для натискання кнопки «X» – закриття програми.
- Для курсора миші при наведенні на функціональні кнопки та поля – зміна виду курсора
- Для текстового поля – відправка введеного адресу на наступну форму для верифікації.
- При наведенні на функціональні кнопки – зміна їх відтінку для візуального контроля.

- При натисканні на функціональні кнопки – зміна їх відтінку для візуального контроля.

Всі ці події були виконані в коді та даний код зображений на рисунку 3.15.



```
Ссылок: 2
public Initialization()
{
    InitializeComponent();
    ipFeald.Text = "IP add";
    ipFeald.ForeColor = Color.Gray;
}
Ссылок: 1
private void Initialization_Load(object sender, EventArgs e)...
Ссылок: 1
private void CloseButton_Click(object sender, EventArgs e)
{
    Application.Exit();
}
Ссылок: 1
private void CloseButton_MouseEnter(object sender, EventArgs e)
{
    CloseButton.ForeColor = Color.Gray;
}
Ссылок: 1
private void CloseButton_MouseLeave(object sender, EventArgs e)
{
    CloseButton.ForeColor = Color.Black;
}
Point LastPoint;
Ссылок: 1
private void Initialization_MouseMove(object sender, MouseEventArgs e)
{
    if(e.Button == MouseButtons.Left)
    {
        this.Left += e.X - LastPoint.X;
        this.Top += e.Y - LastPoint.Y;
    }
}
Ссылок: 1
private void Initialization_MouseDown(object sender, MouseEventArgs e)
{
    LastPoint = new Point(e.X, e.Y);
}
```

```
Ссылок: 2
private void Next_Click(object sender, EventArgs e)
{
    string ipadd = ipFeald.Text;
    this.Hide();
    User user = new User(ipadd);
    user.Show();
}
Ссылок: 1
private void ipFeald_KeyDown(object sender, KeyEventArgs e)
{
    if(e.KeyCode == Keys.Enter)
    {
        Next_Click(sender, e);
    }
}
Ссылок: 1
private void ipFeald_Enter(object sender, EventArgs e)
{
    if (ipFeald.Text == "IP add")
    {
        ipFeald.Text = "";
        ipFeald.ForeColor = Color.Black;
    }
}
Ссылок: 1
private void ipFeald_Leave(object sender, EventArgs e)
{
    if (ipFeald.Text == "")
    {
        ipFeald.Text = "IP add";
        ipFeald.ForeColor = Color.Gray;
    }
}
Ссылок: 1
private void ipFeald_TextChanged(object sender, EventArgs e)...
}
```

Рис. 3.15. Події форми входу

Також для головної форми необхідні відповідні для неї події. А сааме:

- Для натискання кнопки «X» – закриття програми.
- Для натискання стрілки назад – повернення до попередньої форми.
- Для елементу «WebView2» – надання IP-адреси з форми входу та весь функціонал.
- Для курсора миші при наведенні на функціональні кнопки та поля – зміна виду курсора
- При наведенні на функціональні кнопки – зміна їх відтінку для візуального контроля.
- При натисканні на функціональні кнопки – зміна їх відтінку для візуального контроля.

Події для головної форми були виконані в коді та даний код зображений на рисунку 3.16.

```
public partial class User : Form
{
    private String httpProtocolPrefix = "http://";
    private String httpsProtocolPrefix = "https://";
    private String url = "";
    Ссылок: 1
    public User(String url)
    {
        this.url = url;
        InitializeComponent();
    }
    Ссылок: 1
    private void User_Load(object sender, EventArgs e)
    {
        initBrowser();
    }
    Ссылок: 1
    private async Task initizated()
    {
        await webView21.EnsureCoreWebView2Async(null);
    }
    Ссылок: 1
    public async void initBrowser()
    {
        await initizated();
        String targetUrl = url;
        if (!url.Contains(httpsProtocolPrefix) || !url.Contains(httpProtocolPrefix)) {
            targetUrl = httpProtocolPrefix + url;
        }
        webView21.CoreWebView2.Navigate(targetUrl);
    }
    Ссылок: 1
    private void CloseButton_Click(object sender, EventArgs e)
    {
        Application.Exit();
    }
    Ссылок: 1
    private void CloseButton_MouseEnter(object sender, EventArgs e)
    {
        CloseButton.ForeColor = Color.Gray;
    }
```

```
    Ссылок: 1
    private void CloseButton_MouseLeave(object sender, EventArgs e)
    {
        CloseButton.ForeColor = Color.Black;
    }
    Point LastPoint;
    Ссылок: 1
    private void User_MouseMove(object sender, MouseEventArgs e)
    {
        if (e.Button == MouseButtons.Left)
        {
            this.Left += e.X - LastPoint.X;
            this.Top += e.Y - LastPoint.Y;
        }
    }
    Ссылок: 1
    private void User_MouseDown(object sender, MouseEventArgs e)
    {
        LastPoint = new Point(e.X, e.Y);
    }
    Ссылок: 0
    private void textBox1_TextChanged(object sender, EventArgs e)...
    Ссылок: 1
    private void back_Click(object sender, EventArgs e)
    {
        this.Hide();
        Initialization initialization = new Initialization();
        initialization.Show();
    }
    Ссылок: 1
    private void back_MouseEnter(object sender, EventArgs e)
    {
        back.ForeColor = Color.Gray;
    }
    Ссылок: 1
    private void back_MouseLeave(object sender, EventArgs e)
    {
        back.ForeColor = Color.Black;
    }
```

Рис. 3.16. Події головної форми

3.4 Висновки за розділом

У результаті виконання третього розділу кваліфікаційної роботи було виконано реалізацію веб-інтерфейсу, мобільного додатку для ОС Android та ПЗ для ОС Windows на довільному рівні. Веб-інтерфейс розроблений для використання на любій платформі за допомоги любого браузера без будь-яких перешкод. В мобільному додатку та ПЗ був використаний ідентичний функціонал веб-інтерфейсу, та зроблена можливість входу по IP-адресі.

Весь функціонал всіх трьох інтерфейсів був розширений та спрощений в порівнянні з подібними ПЗ для станків з ЧПУ на базі мікроконтролерів Arduino та ESP32, та має можливість в подальшому розвитку.

ВИСНОВКИ

Розроблені в ході роботи програмні забезпечення та веб-інтерфейс для післяпроцесної обробки адитивного виробництва дозволяє пришвидшити, здешевити та оптимізувати післяпроцесну обробку. Зроблена можливість використання ПЗ на різних платформах, з ідентичним внутрішнім інтерфейсом.

З метою практичної реалізації було розроблено програмне забезпечення для мікроконтролера ESP32, який є основою верстата з ЧПУ, та відштовхуючись від розробленого вбудованого веб-сервера МК був зроблений веб-інтерфейс з функціоналом який обширніший та простіший в порівнянні з подібними готовими рішеннями для даної платформи. Завдяки розширеним функціям через веб-інтерфейс є можливість зміни веб-налаштувань МК ESP32, налаштувань конфігурацій GRBL, додавання, видалення та запуск робочої програми (G-кода) з додаткового флеш-накопичувача. А також для подальшого апгрейду, є можливість оновлення прошивки мікроконтролера та самого веб-інтерфейсу.

Основні функції мобільного додатку для операційної системи Android та програмного забезпечення для ОС Windows аналогічні функціям, які є у веб-інтерфейса, тим самим спрощення для користувача у використанні любої платформи для післяпроцесної обробки.

Подальші дослідження доцільно спрямувати в доповненні програмних забезпечень та веб-інтерфейсу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Розробка програмного забезпечення післяпроцесної обробки об’єктів адитивного виробництва / Сокол Г. В., Зуб С. В. // Інформаційно-керуючі системи на залізничному транспорті : Матеріали 35-ї МНПК опубліковані в журналі «Інформаційно-керуючі системи на залізничному транспорті» №3 (Додаток), 2022. – С. 36-37.

2. Нісан А. Друк металопорошковими композиціями: можливості у приладобудуванні / Нісан А. // Вектор. 2018. – № 37. – С. 22-27.

3. Реалізація етапу видалення супортів в об’єктах адитивного виробництва НВЧ-компонентів / Слюсарь І.І., Слюсар В.І., Зуб С.В., Шуть В.В. // Електронні та мехатронні системи: теорія, інновації, практика: Зб. наук. праць за матеріалами V Всеукр. НПК, 08.11.2019 р. – Полтава: НУ «Полтавська політехніка імені Юрія Кондратюка», 2019. – С. 53-57.

4. Слюсарь І.І. Післяпроцесна обробка результатів адитивного виробництва антенних елементів. / Слюсарь І.І., Слюсар В.І., Зуб С.В. // Проблеми інфокомунікацій: Матеріали 3-ої Всеукраїнської НТК. – Полтава: НУ «Полтавська політехніка імені Юрія Кондратюка»; К.: НТУ; Х.: НТУ «ХПІ»; К.: ДУТ; Х.: УкрДУЗТ; Мінськ: БНТУ; Полтава: ВКСС ВІТІ, 2019.

5. 3D-друк STL-файлів [Електронний ресурс] – Режим доступу до ресурсу: https://3dprinter.ua/3d-pechat-stl-fajlov-poshagovoe-rukovodstvo/.

6. Побудова 3D-моделі, придатної для 3D-друку [Електронний ресурс] – Режим доступу до ресурсу: https://3ddevice.com.ua/blog/3d-printer-obzor/3d-modelling-for-3d-printing/.

7. Embedded Software Programming Languages [Електронний ресурс] – Режим доступу до ресурсу: https://www.qt.io/embedded-development-talk/embedded-software-programming-languages-pros-cons-and-comparisons-of-popular-languages

8. PHP [Електронний ресурс] – Режим доступу до ресурсу: https://cutt.ly/oy639nR.

9. Що таке PHP 7? Можливості для програміста [Електронний ресурс].

– 2018. – Режим доступу до ресурсу: https://cutt.ly/Gy68cSd.

10. PHP, Ruby, Python – коротка характеристика трьох мов програмування [Електронний ресурс] – Режим доступу до ресурсу: https://www.internet-technologies.ru/articles/php-ruby-python-harakteristika-yazykovprogrammirovaniya.html#header-9428-2.

11. Мова програмування Python [Електронний ресурс] – Режим доступу до ресурсу: https://learn-code.ru/yazyki-programmirovaniya/python.

12. Java [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Java.

13. Переваги та недоліки Java-технологій [Електронний ресурс] – Режим доступу до ресурсу: https://studopedia.org/8-126786.html.

14. Вибір мови програмування [Електронний ресурс] – Режим доступу до ресурсу: https://prog.bobrodobro.ru/5183

15. Загальні відомості про платформу .NET [Електронний ресурс] – Режим доступу до ресурсу: https://learn.microsoft.com/ru-ru/dotnet/framework/get-started/overview

16. Обґрунтування вибору мови програмування [Електронний ресурс] – Режим доступу до ресурсу: https://studfile.net/preview/6009882/page:7

17. Як вибрати мову програмування для створення Андроїд — додатку [Електронний ресурс] – Режим доступу до ресурсу: https://blogchain.com.ua/iak-vibrati-movy-programyvannia-dlia-stvorennia-android-dodatki/

18. ESP32 – HTTP Web Server – HTML – CSS [Електронний ресурс] – Режим доступу до ресурсу: https://www.instructables.com/ESP32-HTTP-Web-Server-HTML-CSS-Simple-Counter-As-t/

19. HTML basics [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics

20. What is HTML? An Introduction [Електронний ресурс] – Режим доступу до ресурсу: https://codeinstitute.net/global/blog/what-is-html-and-why-

should-i-learn-it/

21. CSS basics [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics

22. How to link JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://codeinstitute.net/global/blog/how-to-link-javascript-to-html/

23. JavaScript basics [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics

24. ESP32:Приклади/Веб-сервер на базі ESP32 [Електронний ресурс] – Режим доступу до ресурсу: https://wikihandbk.com/wiki/ESP32:Приклади/Веб-сервер_на_базі_ESP32:_керування_вихідними_контактами#Веб-сервер_на_базі_ESP32

25. ESP-IDF Programming Guide [Електронний ресурс] – Режим доступу до ресурсу: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/

26. HTTP Server [Електронний ресурс] – Режим доступу до ресурсу: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/protocols/esp_http_server.html

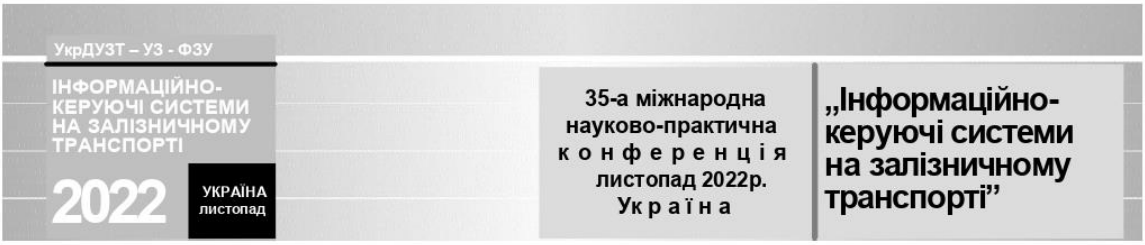
27. Teaching how to build a web server with ESP8266 module in Arduino [Електронний ресурс] – Режим доступу до ресурсу: https://roboeq.ir/blog/آموزش-ماژول-با-سرور-وب-ساخت-esp8266/

28. Веб-сервер ESP32 (ESP8266) серед Arduino IDE [Електронний ресурс] – Режим доступу до ресурсу: https://arduino-tex.ru/news/15/urok-1-veb-server-esp32-esp8266-v-srede-arduino-ide.html

29. Широкосмугові антени на основі кільцевої геометрії / Слюсарь І.І., Слюсар В.І., Зуб С.В., Телешун Д. Ю. // Збірник наукових праць. Системи управління, навігації та зв'язку. Том 2 № 60 Полтава: НУ «Полтавська політехніка імені Юрія Кондратюка», 2020. [Електронний ресурс] – Режим доступу до ресурсу: https://doi.org/10.26906/SUNZ.2020.2/.

ДОДАТОК А
ВІДОМОСТІ ПРО АПРОБАЦІЇ

Міністерство освіти і науки України
Акціонерне товариство „Українська залізниця”
Транспортна академія України
Федерація залізничників України
Український державний університет залізничного транспорту



Оргкомітет:
Голова:
Панченко С. В., д.т.н., ректор Українського державного університету залізничного транспорту

Члени оргкомітету: Бабаєв М. М., д.т.н. (Україна), Бунчуков О. А. (Україна), Бутько Т. В., д.т.н. (Україна), Гаврилюк В. І., д.ф-м.н. (Україна), Гончаренко В. І. (Україна), Доценко С. І ., д.т.н. (Україна), Жуковицький І. В., д.т.н. (Україна), Каргін А. О., д.т.н. (Україна), Климаш М. М., д.т.н. (Україна), Збігнев Лукасік, д.т.н. (Польща), Марек Мезитис, д.т.н. (Латвія), Мойсеєнко В. І ., д.т.н. (Україна), Приходько С. І., д.т.н. (Україна), Рубан І. В., д.т.н. (Україна), Самсонкін В. М., д.т.н. (Україна), Серков О. А., д.т.н. (Україна), Скалозуб В. В., д.т.н. (Україна), Терещенко Ю. М. (Україна), Трубчанінова К. А., д.т.н. (Україна), Тьєрі Хорсін (Франція), Шиш В. О., к.т.н. (Україна), Штомпель М. А., д.т.н. (Україна)

2022 р.
11 листопада

м. Харків,
Україна

ТЕЗИ СТЕНДОВИХ ДОПОВІДЕЙ ТА ВИСТУПІВ УЧАСНИКІВ КОНФЕРЕНЦІЇ

HIGHLIGHTS OF REPORTS AND PRESENTATIONS OF PARTICIPANTS TO THE CONFERENCE

СПИСОК АВТОРІВ

радіозв'язку від 0,2 до 5 км [3].

Застосування чисто детерміністських методів для розрахунку напруженості поля вимагає використання цифрових карт місцевості та не завжди виправдано із-за відсутності інформації про характеристики відбиття сигналів від різних типів забудови.

В роботі проведено аналіз особливостей використання цих рекомендацій ITU-R та виконаний порівняльний розрахунок напруженості поля за цими моделями.

Список використаних джерел

1. Recomendation ITU-R P.1546-3. Method for point-to-area predictions for terrestrial services in the frequency range 30 MHz to 4000 MHz. 2019. – 57 p.
2. Recomendation ITU-R P.1812-6. A path-specific propagation prediction method for point-to-area terrestrial services in the frequency range 30 MHz to 6 000 MHz 2021. – 34 p.
3. Ikegami F. Theoretical prediction of mean field strength on urban mobile radio / F. Ikegami // IEEE Trans. On Antenn. And Propag, 1991. – Vol.39, №3.

Приходько С. І., д.т.н., професор,
Штомпель М. А., д.т.н., професор
(УкрДУЗТ)

УДК 621.391

ЗАСТОСУВАННЯ ХМАРНИХ ТЕХНОЛОГІЙ ПРИ ПОБУДОВІ ІНФОКОМУНІКАЦІЙНИХ СИСТЕМ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ

Постійне зростання обсягів даних та впровадження новітніх інформаційних послуг обумовлює необхідність пошуку інноваційних підходів до побудови інфокомунікаційних систем залізничного транспорту. Аналіз наявних підходів та сучасних концепцій у галузі інфокомунікацій показав, що перспективним напрямом розвитку інфокомунікаційної інфраструктури залізниць є застосування хмарних технологій [1].

На даний момент широке поширення отримали ряд хмарних провайдерів, серед яких найбільш вагомим та функціональним є провайдер Amazon Web Services (AWS). Даний провайдер пропонує значну кількість мережевих сервісів, сервісів зберігання даних, сервісів захисту інформації тощо. Інфокомунікаційні системи залізничного транспорту можуть розглядатися як різновид корпоративних мережевих систем, що дозволяє застосовувати наявні типові та стандартизовані рішення щодо їх побудови та модернізації на базі хмарних сервісів AWS [2, 3].

У роботі проведено аналіз наявних мережевих та супутніх сервісів AWS, на основі якого запропоновані технічні рішення щодо удосконалення підходів до реалізації інфокомунікаційних систем залізничного транспорту. Представлено архітектуру сегменту інфокомунікаційної інфраструктури для ділянки залізниці, що побудована з використанням обраних сервісів AWS. Також у роботі визначено особливості застосування хмарних технологій при побудові інфокомунікаційних систем на ділянках залізниць та наведено відповідні практичні рекомендації.

Список використаних джерел

1. Воробієнко, П.П. Телекомунікаційні та інформаційні мережі / П.П. Воробієнко, Л.А. Нікітюк, П.І. Резніченко. – К., 2010. – 708 с.
2. Linthicum, David S. Cloud-Native Applications and Cloud Migration: The Good, the Bad, and the Points Between / David S. Linthicum // IEEE Cloud Computing. – 2017. – Vol. 4, No. 5. – P. 12-14.
3. Takabi, H. Security and Privacy Challenges in Cloud Computing Environments / H. Takabi, J. B. D. Joshi, G.-J. Ahn // IEEE Security & Privacy. – 2010. – Vol. 8, No. 6. – P. 24-31.

Сокол Г. В., к.т.н., доцент,
Зуб С. В., магістрант
(Національний університет «Полтавська політехніка імені Юрія Кондратюка»)

УДК: 621.9.015

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ПІСЛЯПРОЦЕСНОЇ ОБРОБКИ ОБ'ЄКТІВ АДИТИВНОГО ВИРОБНИЦТВА

Сьогодні технології адитивного виробництва все частіше використовуються в різних сферах, зокрема: виготовлення електронних компонентів, виготовлення моделей НВЧ-компонентів, створення механізмів та інше. Адитивні технології мають значні переваги перед традиційними. А саме, 3D-друк дає можливість: спростити зборку, зменшити масу, кількість з'єднань, ущільнень та кріплень, а також підвищує надійність системи в цілому; значно спрощує перехід від класичної планарної компоновки електронних пристроїв до об'ємної; виробляти недорогі пристрої як при малих серіях, так і при створенні макетів та дослідних зразків.

До найбільш поширених технологій адитивного виробництва слід віднести: стереолітографія (Laser Stereolithography, SLA); метод пошарового наплавлення (Fused Deposition Modeling, FDM) і селективне лазерне спікання (Selective laser sintering, SLS). З точки зору економічної ефективності, до початку масового виробництва доцільно використовувати FDM. Однак, під час такого 3D-друку

модель формується шар за шаром, тобто необхідний попередній шар для формування нового. В залежності від складності 3Dмоделі та специфіки застосування технологій 3D-друку, може з'явитися необхідність використовувати спеціальні супорти. Важливо розуміти, що вони безпосередньо впливають на якість кінцевих виробів, так як вони разом з нульовим шаром підлягають видаленню. Все це може призвести до нерівності поверхні друкованої 3D-моделі. Відповідно, виникає необхідність оптимізації післяпроцесної обробки результатів адитивного виробництва.

Для вирішення цього завдання доцільно використовувати лазер. При цьому, високотемпературний вплив лазерного випромінювання забезпечує видалення матеріалу. В якості прототипу обраний набір для зборки 2D-верстату лазерного гравірування (рис. 1). Через те, що у початковому вигляді він не пристосований для постобробки адитивного виробництва, була виконана його модифікація (рис. 2). При цьому, додана третя вісь зсуву (Oz), встановлено кабель-канал і кінцеві вимикачі Ox і Oy, за допомогою розробленого електронного сегменту реалізоване управління мікроконтроллером ESP32 (замість базового мікроконтроллера Arduino Nano), замінена кабельна система, для продуву робочої поверхні на драйвер лазеру змонтований вентилятор.

Рис. 1. 2D-верстат

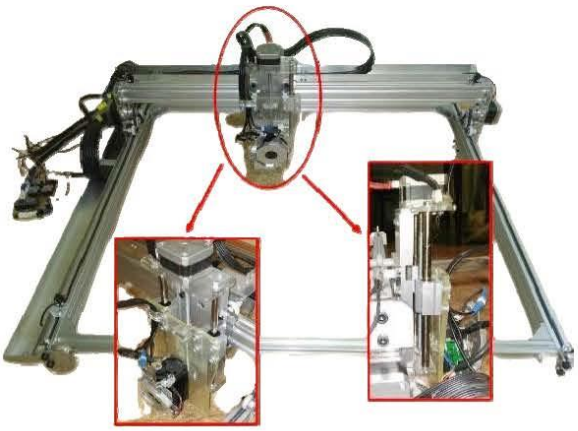


Рис. 2. Прототип 3D-верстату

Після апаратної модифікації зросло навантаження на базовий мікрокотроллер Arduino Nano, через що і був замінений на більш потужний мікроконтроллер ESP32 (к. Espressif Systems, Піднебесна), що містить більший перелік можливостей, застосовує Bluetooth та /або Wi-Fi і має в собі вбудований Web-сервер, який надає змогу розробити Web-інтерфейс та підключення додаткових програмних забезпечень для спрощення користувацького керування, як в домашній мережі, так і віддалено.

Таким чином, напрямком подальшої роботи є програмування мікроконтроллера ESP32 для наших цілей, проектування Web-інтерфейсу для використання вбудованим Web-сервером ESP32, розробку програмного забезпечення для операційних систем Android та Windows, задля спрощення користувацького використання та можливасті віддаленого доступу керування верстатом, та його налаштування.

Список використаних джерел

1. Реалізація етапу видалення супортів в об'єктах адитивного виробництва НВЧ-компонентів / Слюсарь І.І., Слюсар В.І., Зуб С.В., Шуть В.В. // Електронні та мехатронні системи: теорія, інновації, практика: Зб. наук. праць за матеріалами V Всеукр. НПК, 08.11.2019 р. – Полтава: НУ «Полтавська політехніка імені Юрія Кондратюка», 2019. – С. 53-57.
2. Слюсарь І.І. Післяпроцесна обробка результатів адитивного виробництва антенних елементів. / Слюсарь І.І., Слюсар В.І., Зуб С.В. // Проблеми інфокомунікацій: Матеріали 3-ої Всеукраїнської НТК. – Полтава: НУ «Полтавська політехніка імені Юрія Кондратюка»; К.: НТУ; Х.: НТУ «ХПІ»; К.: ДУТ; Х.: УкрДУЗТ; Мінськ: БНТУ; Полтава: ВКСС ВІТІ, 2019
3. 3D Принтери [Електронний ресурс] // 3D принтер в Україні. – 2018. – Режим доступу до ресурсу: https://3dprinter.ua/products/3d-printers/.

ДОДАТОК Б

АРХІТЕКТУРНА ЧАСТИНА ВИХІДНОГО КОДУ МІКРОКОНТРОЛЕРА ESP32

Grbl_Esp32.ino

```
#include "src/Grbl.h"
void setup() {
    grbl_init();
}
void loop() {
    run_once();
}
```

Grbl.h

```
// Grbl versioning system
const char* const GRBL_VERSION       = "1.3a";
const char* const GRBL_VERSION_BUILD = "20211103";
#include <Arduino.h>
#include <EEPROM.h>
#include <driver/rmt.h>
#include <esp_task_wdt.h>
#include <freertos/task.h>
#include <Preferences.h>
#include <driver/timer.h>
#include "Config.h"
#include "NutsBolts.h"
#include "Defaults.h"
#include "Error.h"
#include "WebUI/Authentication.h"
#include "WebUI/Commands.h"
#include "Probe.h"
#include "System.h"
#include "GCode.h"
#include "Planner.h"
#include "CoolantControl.h"
#include "Limits.h"
#include "MotionControl.h"
#include "Protocol.h"
#include "Uart.h"
#include "Serial.h"
#include "Report.h"
#include "Pins.h"
#include "Spindles/Spindle.h"
#include "Motors/Motors.h"
#include "Stepper.h"
#include "Jog.h"
#include "WebUI/InputBuffer.h"
#include "Settings.h"
#include "SettingsDefinitions.h"
#include "WebUI/WebSettings.h"
#include "UserOutput.h"
#include <Wire.h>
#include "SDCard.h"
#ifdef ENABLE_BLUETOOTH
#include "WebUI/BTConfig.h"
#endif
#ifdef ENABLE_WIFI
#    include "WebUI/WifiConfig.h"
#    ifdef ENABLE_HTTP
#        include "WebUI/Serial2Socket.h"
#    endif
```

```
    #    ifdef ENABLE_TELNET
    #        include "WebUI/TelnetServer.h"
    #    endif
    #    ifdef ENABLE_NOTIFICATIONS
    #        include "WebUI/NotificationsService.h"
    #    endif
    #endif
    #include "I2SOut.h"
    void grbl_init();
    void run_once();
    void machine_init();                      // weak definition in Grbl.cpp
    void display_init();                      // weak definition in Grbl.cpp
    void user_m30();                          // weak definition in Grbl.cpp/
    void user_tool_change(uint8_t new_tool);  // weak definition in Grbl.cpp
    bool user_defined_homing(uint8_t cycle_mask);  // weak definition in Limits.cpp
    bool cartesian_to_motors(float* target, plan_line_data_t* pl_data, float*
position);
    bool kinematics_pre_homing(uint8_t cycle_mask);
    void kinematics_post_homing();
    bool limitsCheckTravel(float* target);  // weak in Limits.cpp; true if out of range
    void motors_to_cartesian(float* cartestian, float* motors, int n_axis);  // weak
definition
    MACRO_BUTTON_2_PIN is defined
    void user_defined_macro(uint8_t index);
```

WebSettings.h

```
    namespace WebUI {
        extern StringSetting* wifi_sta_ssid;
        extern StringSetting* wifi_sta_password;
    #ifdef ENABLE_WIFI
        extern EnumSetting*   wifi_sta_mode;
        extern IPaddrSetting* wifi_sta_ip;
        extern IPaddrSetting* wifi_sta_gateway;
        extern IPaddrSetting* wifi_sta_netmask;
        extern StringSetting* wifi_ap_ssid;
        extern StringSetting* wifi_ap_password;
        extern IPaddrSetting* wifi_ap_ip;
        extern IntSetting* wifi_ap_channel;
        extern StringSetting* wifi_hostname;
        extern EnumSetting*   http_enable;
        extern IntSetting*    http_port;
        extern EnumSetting*   telnet_enable;
        extern IntSetting*    telnet_port;
    #endif
    #ifdef WIFI_OR_BLUETOOTH
        extern EnumSetting* wifi_radio_mode;
    #endif
    #ifdef ENABLE_BLUETOOTH
        extern StringSetting* bt_name;
    #endif
    #ifdef ENABLE_AUTHENTICATION
        extern StringSetting* user_password;
        extern StringSetting* admin_password;
    #endif
    #ifdef ENABLE_NOTIFICATIONS
        extern EnumSetting*   notification_type;
        extern StringSetting* notification_t1;
        extern StringSetting* notification_t2;
        extern StringSetting* notification_ts;
    #endif
    }
```

ДОДАТОК В

АРХІТЕКТУРНА ЧАСТИНА ВИХІДНОГО КОДУ ПРОГРАМИ WINDOWS

Initialization.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Security.Cryptography.X509Certificates;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Microsoft.Web.WebView2.Core;
using Microsoft.Web.WebView2.WinForms;

namespace Web_GRBL
{
    public partial class Initialization : Form
    {
          public Initialization()
        {
            InitializeComponent();
            ipFeald.Text = "IP add";
            ipFeald.ForeColor = Color.Gray;
        }
        private void Initialization_Load(object sender, EventArgs e)
        { }
        private void CloseButton_Click(object sender, EventArgs e)
        { Application.Exit(); }
        private void CloseButton_MouseEnter(object sender, EventArgs e)
        { CloseButton.ForeColor = Color.Gray;  }
        private void CloseButton_MouseLeave(object sender, EventArgs e)
        { CloseButton.ForeColor = Color.Black; }
        Point LastPoint;
        private void Initialization_MouseMove(object sender, MouseEventArgs e)
        {  if(e.Button == MouseButtons.Left)
            {   this.Left += e.X - LastPoint.X;
                this.Top += e.Y - LastPoint.Y;   }   }
        private void Initialization_MouseDown(object sender, MouseEventArgs e)
        {  LastPoint = new Point(e.X, e.Y); }
        private void Next_Click(object sender, EventArgs e)
        {   string ipadd = ipFeald.Text;
            this.Hide();
            User user = new User(ipadd);
            user.Show(); }
           private void ipFeald_KeyDown(object sender, KeyEventArgs e)
        { if(e.KeyCode == Keys.Enter)
            { Next_Click(sender, e); }  }
        private void ipFeald_Enter(object sender, EventArgs e)
        { if (ipFeald.Text == "IP add")
            {  ipFeald.Text = "";
                ipFeald.ForeColor = Color.Black; }
            ;   }
        private void ipFeald_Leave(object sender, EventArgs e)
        { if (ipFeald.Text == "")
            { ipFeald.Text = "IP add";
              ipFeald.ForeColor = Color.Gray; }  }
    }
}
```

User.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Microsoft.Web.WebView2.Core;
using Microsoft.Web.WebView2.WinForms;

namespace Web_GRBL
{  public partial class User : Form
    {   private String httpProtocolPrefix = "http://";
        private String httpsProtocolPrefix = "https://";
        private String url = "";
        public User(String url)
        {  this.url = url;
            InitializeComponent(); }
        private void User_Load(object sender, EventArgs e)
        { initBrowser(); }
        private async Task initizated()
        { await webView21.EnsureCoreWebView2Async(null); }
        public async void initBrowser()
        {  await initizated();
            String targetUrl = url;
            if (!url.Contains(httpsProtocolPrefix) || !url.Contains(httpProtocolPrefix))
{       targetUrl = httpProtocolPrefix + url;             }
            webView21.CoreWebView2.Navigate(targetUrl);           }
        private void CloseButton_Click(object sender, EventArgs e)
        { Application.Exit(); }
        private void CloseButton_MouseEnter(object sender, EventArgs e)
        {  CloseButton.ForeColor = Color.Gray; }
        private void CloseButton_MouseLeave(object sender, EventArgs e)
        {   CloseButton.ForeColor = Color.Black;    }
       Point LastPoint;
        private void User_MouseMove(object sender, MouseEventArgs e)
        {  if (e.Button == MouseButtons.Left)
            {  this.Left += e.X - LastPoint.X;
                this.Top += e.Y - LastPoint.Y; }  }
        private void User_MouseDown(object sender, MouseEventArgs e)
        {   LastPoint = new Point(e.X, e.Y);  }
        private void textBox1_TextChanged(object sender, EventArgs e)
        {   }
        private void back_Click(object sender, EventArgs e)
        {  this.Hide();
            Initialization initialization = new Initialization();
            initialization.Show(); }
        private void back_MouseEnter(object sender, EventArgs e)
        {  back.ForeColor = Color.Gray; }
        private void back_MouseLeave(object sender, EventArgs e)
        {  back.ForeColor = Color.Black; }
        private void webView21_Click(object sender, EventArgs e)
        {    }
    }
}
```

ДОДАТОК Г

АРХІТЕКТУРНА ЧАСТИНА ВИХІДНОГО КОДУ МОБІЛЬНОГО ДОДАТКУ

BaseFragment

```
     package com.zub.webviewexample.core.view
import android.content.Context.INPUT_METHOD_SERVICE
import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.view.inputmethod.InputMethodManager
import androidx.appcompat.widget.Toolbar
import androidx.fragment.app.Fragment
import androidx.navigation.NavController
import androidx.navigation.Navigation
import androidx.navigation.ui.NavigationUI
import androidx.viewbinding.ViewBinding
import com.zub.webviewexample.R
import com.zub.webviewexample.core.navigation.NavigationHandler
import com.zub.webviewexample.core.viewmodel.BaseViewModel

abstract class BaseFragment<VM : BaseViewModel, VB : ViewBinding, NH :
NavigationHandler> :
    Fragment() {
    lateinit var navController: NavController
    protected lateinit var binding: VB
    protected abstract val viewModel: VM
    protected abstract val navigationHandler: NH
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View {     binding = provideViewBinding()
                  return binding.root }
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        setupNavigation()
        init()
    }
    private fun setupNavigation() {
        navController = Navigation.findNavController(requireView())
        navController.addOnDestinationChangedListener { _, _, _ -> hideKeyboard() }
        viewModel.navigationEvent.observe(viewLifecycleOwner) {
            navigationHandler.handle(navController, it)
        }
        val toolbar = requireView().findViewById<Toolbar>(R.id.toolbar)
        toolbar?.let {
            NavigationUI.setupWithNavController(it, navController)
        }
    }
    protected fun hideKeyboard() {
        val inputMethodManager =
      context?.getSystemService(INPUT_METHOD_SERVICE) as InputMethodManager?
     inputMethodManager?.hideSoftInputFromWindow(activity?.currentFocus?.windowToken, 0)
    }
    protected abstract fun provideViewBinding(): VB
    protected abstract fun init()
}
```

BaseViewModel

```
     package com.zub.webviewexample.core.viewmodel
import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel
import com.zub.webviewexample.core.model.LoadingEvent
import com.zub.webviewexample.core.navigation.NavigationEvent
import com.zub.webviewexample.core.utils.SingleLiveEvent
abstract class BaseViewModel : ViewModel() {
    protected val mNavigationEvent = SingleLiveEvent<NavigationEvent>()
    val navigationEvent: LiveData<NavigationEvent>
        get() = mNavigationEvent
    protected val mLoadingEvent = MutableLiveData<LoadingEvent>()
    val loadingEvent: LiveData<LoadingEvent>
        get() = mLoadingEvent
    protected fun showLoading() = mLoadingEvent.postValue(LoadingEvent.ShowLoading)
    protected fun hideLoading() = mLoadingEvent.postValue(LoadingEvent.HideLoading)
}
```

SetupUrlFragment

```
     package com.zub.webviewexample.feature.view
import android.text.InputFilter
import androidx.core.widget.addTextChangedListener
import com.zub.webviewexample.core.view.BaseFragment
import com.zub.webviewexample.databinding.FragmentSetupUrlBinding
import com.zub.webviewexample.feature.navigation.FeatureNavigationHandler
import com.zub.webviewexample.feature.viewmodel.FeatureViewModel
import org.koin.android.ext.android.inject
import org.koin.androidx.viewmodel.ext.android.viewModel
class SetupUrlFragment :
    BaseFragment<FeatureViewModel, FragmentSetupUrlBinding, FeatureNavigationHandler>()
{   override val viewModel: FeatureViewModel by viewModel()
    override val navigationHandler: FeatureNavigationHandler by inject()
    override fun provideViewBinding() = FragmentSetupUrlBinding.inflate(layoutInflater)
    override fun init(): Unit = with(binding) {
        submitButton.setOnClickListener {
            val url = addressEditText.text.toString()
            viewModel.connectTo(url)
        }  applyTextFilter()     }
    private fun applyTextFilter() = with(binding) {
        addressEditText.addTextChangedListener {
            val isEmpty = it?.isEmpty() ?: true
            submitButton.isEnabled = !isEmpty   }    }   }
```

WebViewFragment

```
     package com.zub.webviewexample.feature.view
import android.annotation.SuppressLint
import android.graphics.Bitmap
import android.webkit.WebResourceError
import android.webkit.WebResourceRequest
import android.webkit.WebView
import android.webkit.WebViewClient
import androidx.activity.OnBackPressedCallback
import androidx.core.view.isInvisible
import androidx.core.view.isVisible
import androidx.navigation.fragment.navArgs
import com.zub.webviewexample.R
import com.zub.webviewexample.core.utils.showShortToast
import com.zub.webviewexample.core.view.BaseFragment
import com.zub.webviewexample.databinding.FragmentWebViewBinding
import com.zub.webviewexample.feature.navigation.FeatureNavigationHandler
```

```
import com.zub.webviewexample.feature.viewmodel.FeatureViewModel
import org.koin.android.ext.android.inject
import org.koin.androidx.viewmodel.ext.android.viewModel
class WebViewFragment :
    BaseFragment<FeatureViewModel, FragmentWebViewBinding, FeatureNavigationHandler>() {
    override val viewModel: FeatureViewModel by viewModel()
    override val navigationHandler: FeatureNavigationHandler by inject()
    private val args: WebViewFragmentArgs by navArgs()
    private val webClient = object : WebViewClient() {
        override fun shouldOverrideUrlLoading(view: WebView?, request:
WebResourceRequest?) = false
        override fun onPageStarted(view: WebView?, url: String?, favicon: Bitmap?) {
            super.onPageStarted(view, url, favicon)
            viewModel.loadingStartedCommand()   }
        override fun onPageFinished(view: WebView?, url: String?) {
            super.onPageFinished(view, url)
            viewModel.contentLoadedCommand()   }
        override fun onReceivedError(
            view: WebView?,
            request: WebResourceRequest?,
            error: WebResourceError?
        ) {  super.onReceivedError(view, request, error)
            viewModel.errorOccurredCommand()  }     }
    private val backPressedCallback = object : OnBackPressedCallback(true) {
        override fun handleOnBackPressed() = with(binding) {
            if (webView.canGoBack()) webView.goBack()
            else viewModel.navigateBackCommand()   }    }
    override fun provideViewBinding() = FragmentWebViewBinding.inflate(layoutInflater)
    @SuppressLint("SetJavaScriptEnabled")
    override fun init() {
        binding.webView.apply {
            settings.javaScriptEnabled = true
            webViewClient = webClient
            loadUrl(args.url)   }
        requireActivity().onBackPressedDispatcher.addCallback(
            viewLifecycleOwner,
            backPressedCallback
           viewModel.state.observe(viewLifecycleOwner, ::onStateChanged)
    }
    private fun onStateChanged(state: FeatureViewModel.State) {
        when (state) {
            FeatureViewModel.State.Content -> onContent()
            FeatureViewModel.State.Error -> onError()
            FeatureViewModel.State.Loading -> onLoading()
        }      }
    private fun onContent() = with(binding) {
        progressBar.isVisible = false
        webView.isInvisible = false
    }
    private fun onError() = with(binding) {
        progressBar.isVisible = false
        webView.isInvisible = false
        showShortToast(getString(R.string.resource_loading_error))
    }
    private fun onLoading() = with(binding) {
        progressBar.isVisible = true
        webView.isInvisible = true
    }
    override fun onDestroy() {
        super.onDestroy()
        backPressedCallback.remove()
    }
}
```

ДОДАТОК Д

SECTION 3.

PRACTICAL PART OF SOFTWARE DEVELOPMENT

3.1 Development of the Web interface

The first stage of implementation of software development is the development of a web interface. On the basis of which the following software will be developed. Therefore, I consider it necessary to describe the process of implementing the web interface starting with the main section and its components.

The main functionality of the web interface is to prepare the machine for work, namely, setting the initial position of the working tool and enabling the necessary functions, and sending the G-code file to the flash drive, from which the microcontroller reads this file and performs work in response to it.

The web interface itself is presented in Figure 3.1.

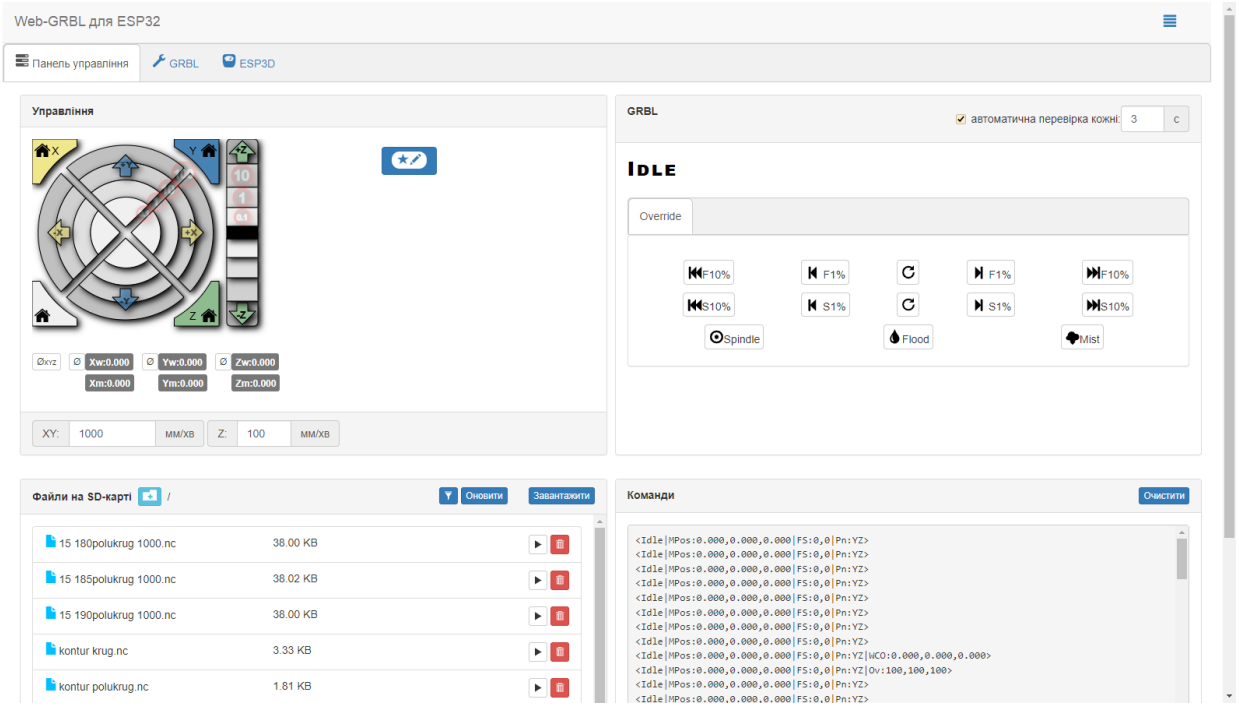


Fig. 3.1. The main panel of the web interface

Additional functionality is also required for configuring the microcontroller, namely ESP32 network settings (Fig. 3.2) and GRBL configuration settings (Fig. 3.3), which are described in the first section of the work.

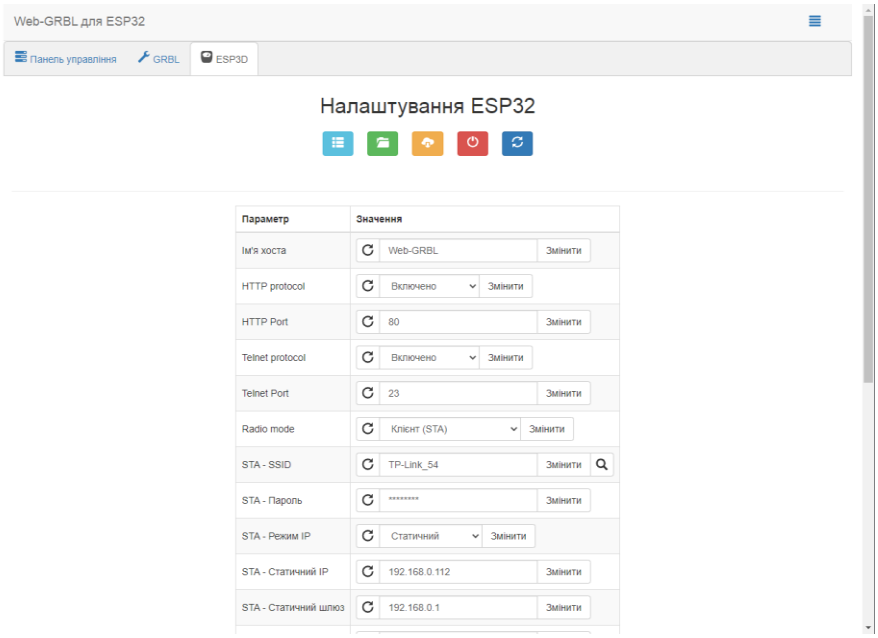


Fig. 3.2. Microcontroller web configuration

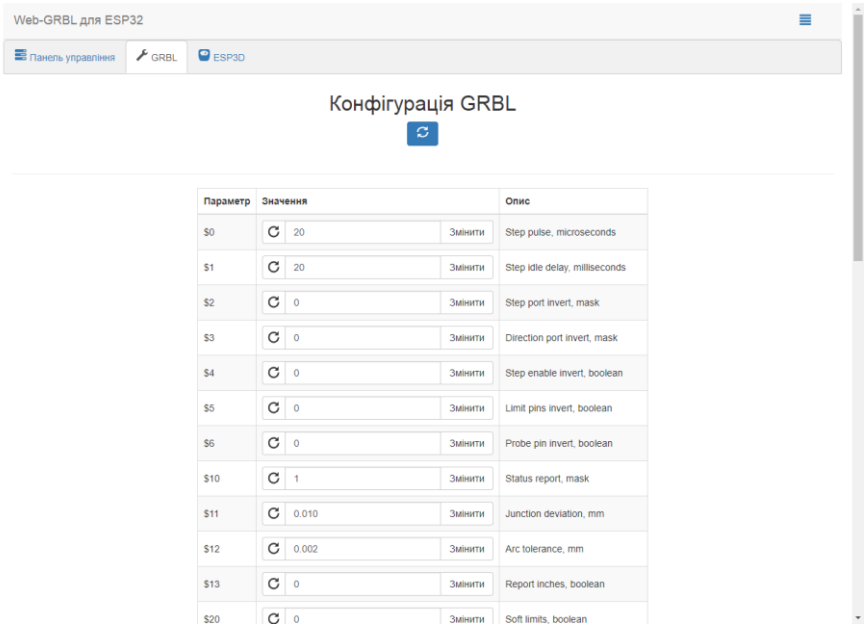


Fig. 3.3. GRBL configuration

The web interface has a top panel that allows you to choose between the main panel, GRBL configuration settings, and web settings. Each of the options opens as a panel, not as a new page, which makes it possible not to reselect the settings each time after selecting a panel.

The main panel is separated by an HTML content division element ( ) to simplify the use of the interface and expand its capabilities. Panel blocks are presented in Figure 3.4.

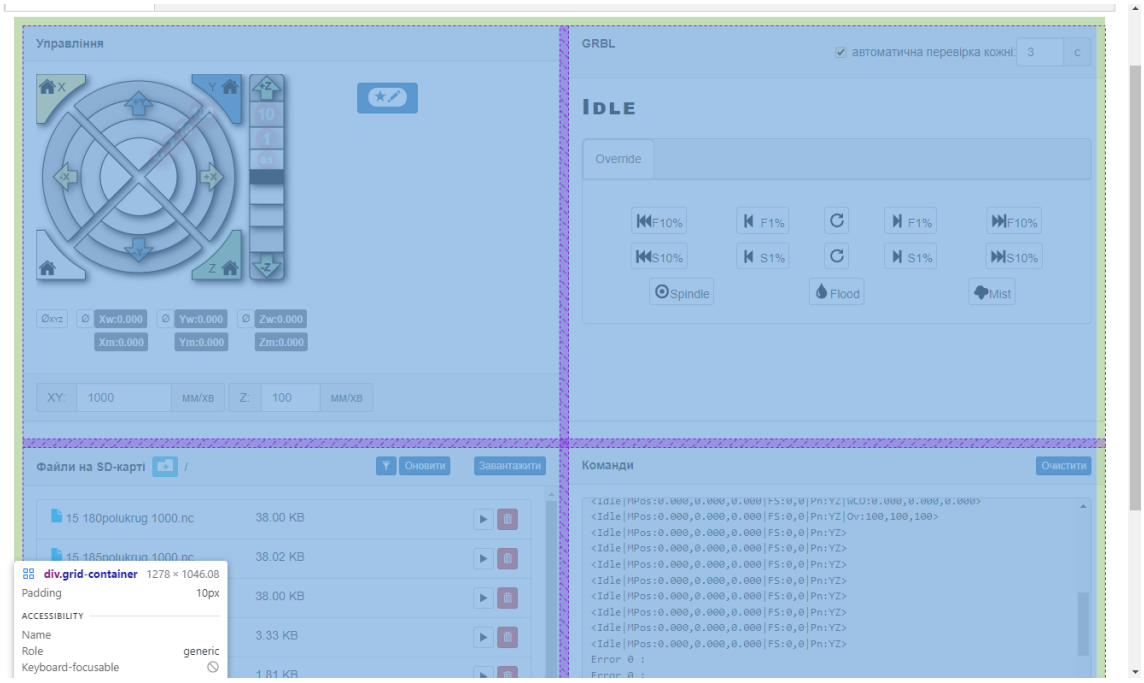


Fig. 3.4. Blocks of the main panel

And each block contains further divided blocks (Fig. 3.5) and so on to each individual element on the page, for example, the button for zeroing the axes on the page (Fig. 3.6) or the button for adding files to the flash drive (Fig. 3.7 a) and the button starting the file operation (Fig. 3.7 b).

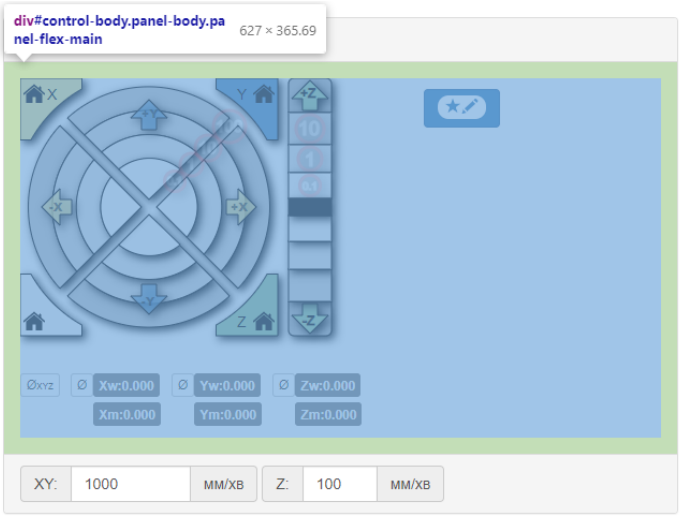


Fig. 3.5. Main panel sub-unit

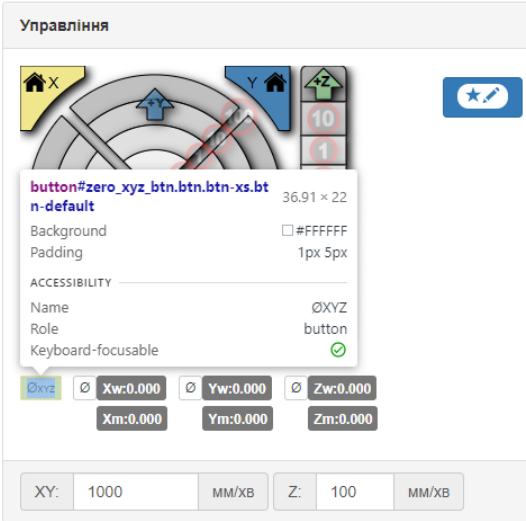
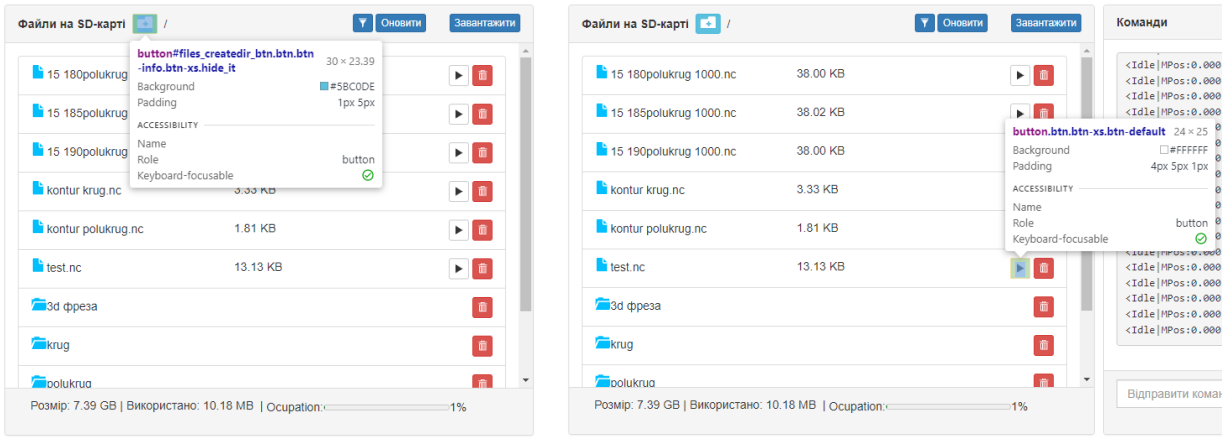


Fig. 3.6. Reset button



a) b)

Fig. 3.7. The button to add files and start work from a file

When the microcontroller uses the client mode, a static IP address must be set in the settings (Fig. 3.8.) for logging in through the web interface and for connecting to the microcontroller from the mobile application and software, which will be described later.

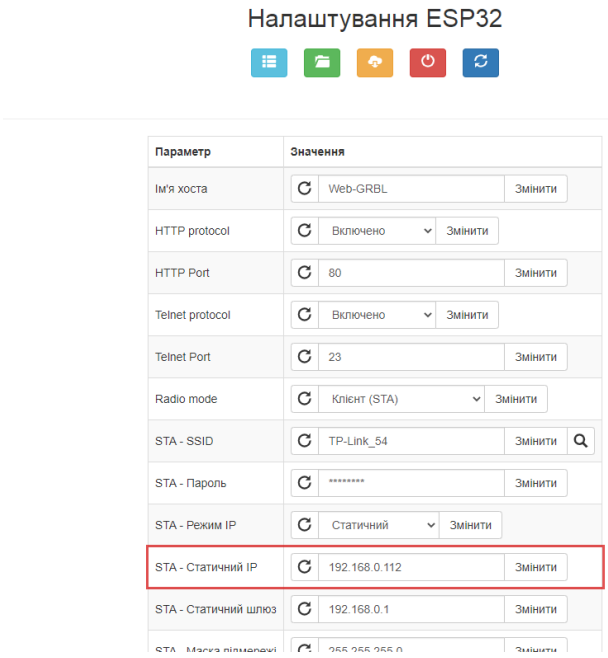


Fig. 3.8. Setting the IP address

The functionality of the Web interface is more extensive than similar software for CNC machines based on Arduino and ESP32 microcontrollers. Namely: sending, deleting and starting working files with G-code; built-in web configuration of the microcontroller and GRBL configuration settings; adding macros to control the CNC machine; firmware and web interface updates through the web interface itself.

3.2 Development of a mobile application

3.2.1 Structure of the mobile application

To distribute the application and create conditions for convenient use, we will determine the target platform for the client part. The most common mobile OS in this period of time is Android. Thus, it is chosen as the target platform for the implementation of the mobile application.

According to the previous sections, the project is built in the Android Studio software. The project structure in Android Studio is a set of modules. By default, a base module is created. Each of them consists of two basic blocks. The first is responsible for classes, that is, logic. The second is responsible for the resources and markup of visual elements. After creating an empty project using the built-in engine, you need to start by defining the directory structure and architectural design pattern. As a result, the project code will be divided into separate components for data, graphical interface, business logic and others. This approach provides the developer with a convenient opportunity to develop each component of the project separately. It is also good that such structural elements can be replaced by a more modern implementation in the future. The main thing is to organize dependencies not at the implementation level, but at the protocol level. In this case, we highlight the following list of components:

1. Dependency injection.
2. Business logic.
3. Graphical interface.
4. Navigation.
5. Utilities.

The part of the project responsible for resource files and markup does not require intervention, because it is rigidly defined by the IDE system and is immutable.

Thus, it is determined that the basic structure of the mobile application project is created on the basis of the MVVM template and consists of the above-listed parts.

3.2.2 Graphical interface of the mobile application

In order to connect to our microcontroller via the network, access control is installed on the initial screen, namely, to enter, you need to enter the IP address of this microcontroller, which is set in the web settings. So, for this functionality, an address input element is designed. In projects under the Android OS, the development standard stipulates that a separate fragment must be created for each task. In our case, one task. For the connection, there are such components as a text input field, a confirmation button for sending data for verification, and a progress bar. Using the built-in tools of the Android IDE, namely xml markup, the design of the layout was performed. The result is shown in Figure 3.9, for the dark and light themes of the smartphone.

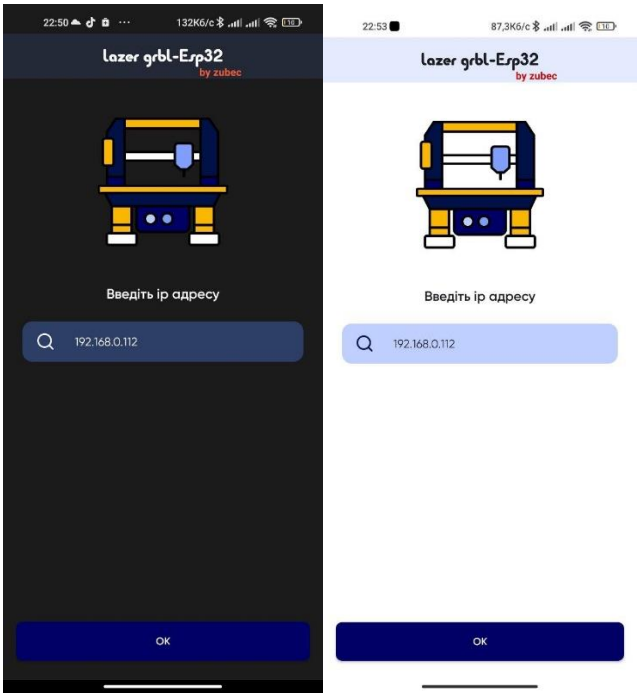


Fig. 3.9. Design mockup for connection

The next step is the main screen, which is similar to the main panel of the web interface. The basic component here is the WebView class with the View extension of the Android class. In this way, the class is placed on the entire screen. Thanks to this class, it is possible to completely copy the functionality of the web interface, which provides simplicity in the transition between devices. Figure 3.10 shows the designed layout of the main fragment for the dark (3.10. a) and light (3.10. b) smartphone themes.

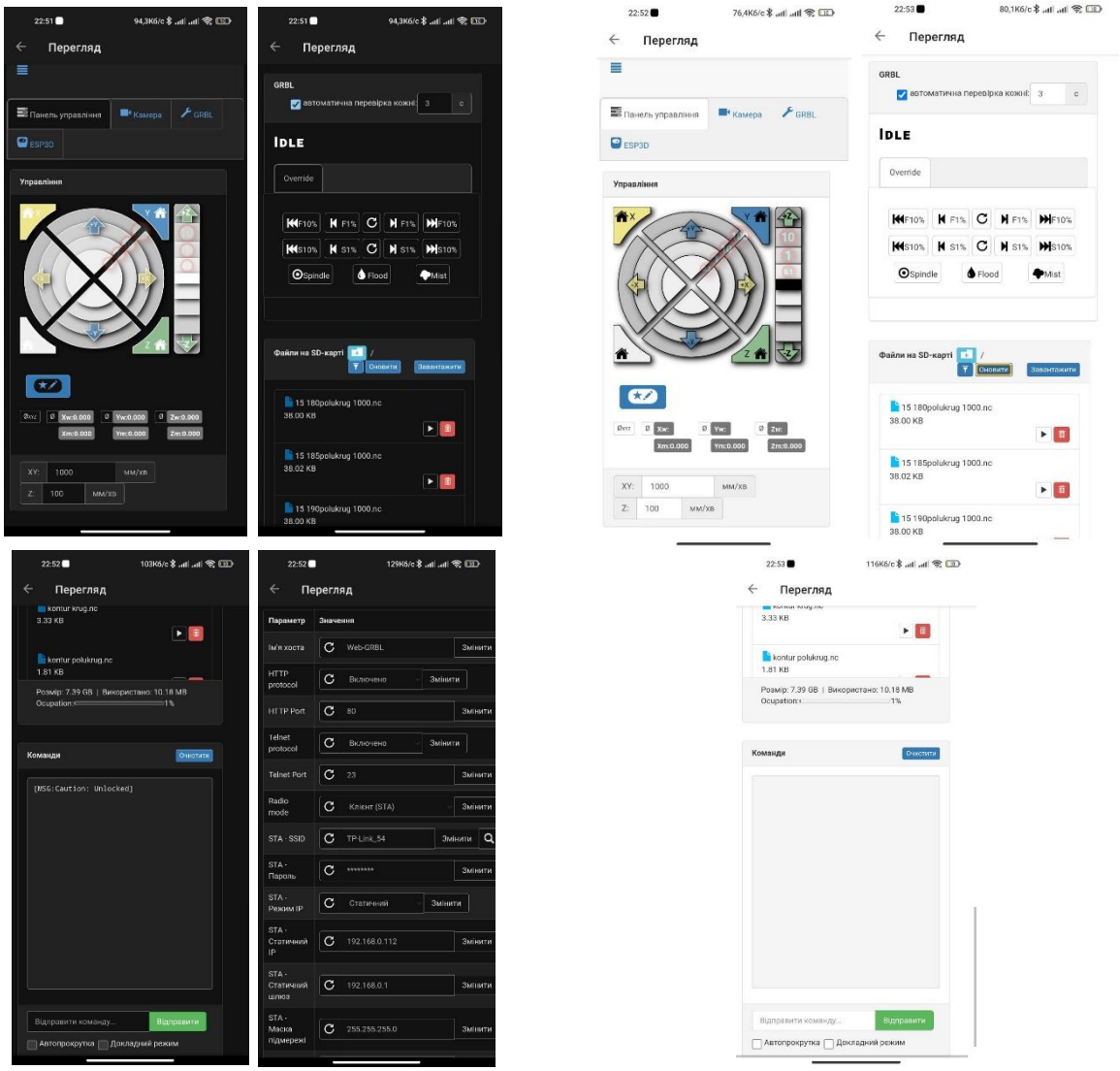


Fig. 3.10. Main screen layout

3.2.3 Classes of visual element fragments and business logic of ViewModel

The bulk of applications are built following the "Single Activity" approach, which means having a single class for the screen with switching visual fragments on it. So the basic component of the GUI is a fragment. It interacts with classes of business logic, binding of visual components, navigation event handler. So we specify them as "Generic" types of our fragment. They also need to be initialized. For this, the class is made abstract and the declaration of the abstract initialization method is performed for each of the objects. Next, the toolbar is bound to the navigation component. This completes the implementation of the basic fragment.

SOLID principles were followed and code was created that is protocol-dependent, not implementation-dependent. In this way, the class is easy to implement in any other project. The code structure of the created class is shown in Figure 3.11.

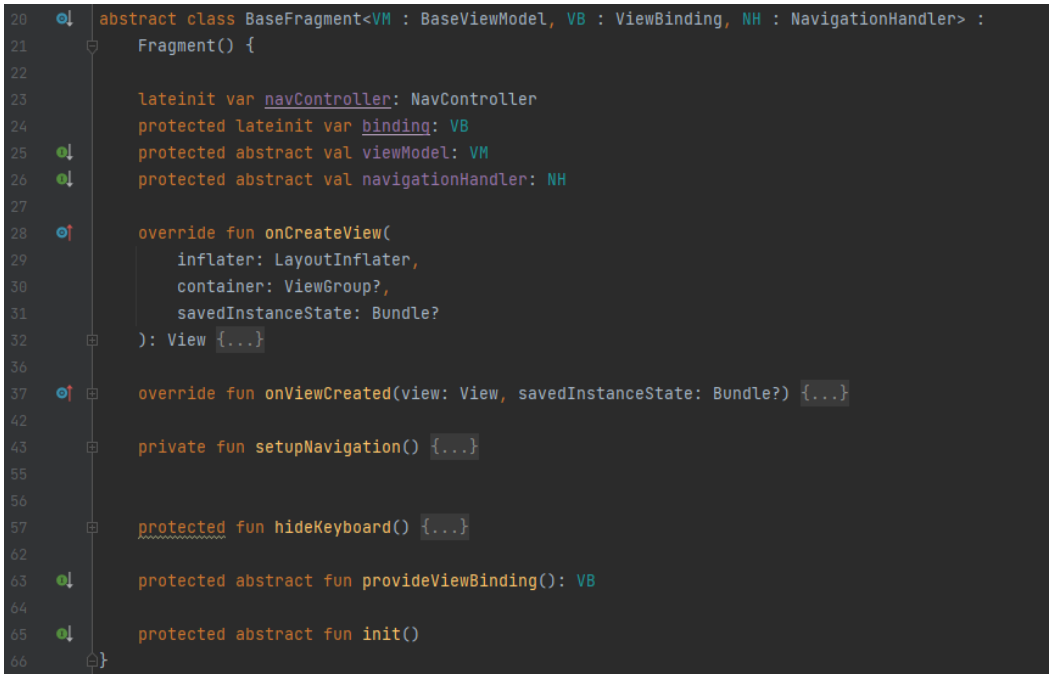


Fig. 3.11. The structure of the fragment base class

The business logic layer plays one of the most important roles in software. After all, it is here that various mechanisms for working with data are managed. In order to reuse the code by programmers, the ViewModel mechanism was developed, which is designed to transfer commands to the graphical interface area. 4 additional components will be introduced into this class, namely:

1. Navigation event mechanism.
2. Commands for displaying the loading message.
3. Commands for displaying error messages or important information.

All the specified elements for commands are implemented using the "MutableLiveData" message mechanism. It is this mechanism that contains the logic of transferring data to the main stream. At the heart of this mechanism is the Overseer design template. The code structure of the implemented base for ViewModel is shown in Figure 3.12.

```
package com.zub.webviewexample.core.viewmodel

import ...

abstract class BaseViewModel : ViewModel() {

    protected val mNavigationEvent = SingleLiveEvent<NavigationEvent>()
    val navigationEvent: LiveData<NavigationEvent>
        get() = mNavigationEvent

    protected val mLoadingEvent = MutableLiveData<LoadingEvent>()
    val loadingEvent: LiveData<LoadingEvent>
        get() = mLoadingEvent

    protected fun showLoading() = mLoadingEvent.postValue(LoadingEvent.ShowLoading)

    protected fun hideLoading() = mLoadingEvent.postValue(LoadingEvent.HideLoading)

}
```

Fig. 3.12. The structure of the base ViewModel

3.3 Development of software for personal computers

3.3.1 Software structure

At the moment, the most common OS for personal computers is Windows, the current version of which is Windows 10. However, the software must be compatible with older versions of the OS. This was also taken into account during the development of the software.

According to the previous sections, the project was built in Visual Studio software. The structure of the project in Visual Studio is a visual designer (Windows Forms), which provides one of the most effective ways of creating software. By default, a form is created. A form is a Windows user interface.

Typically, a Windows Forms application is built by adding controls to forms and creating code to respond to user actions such as mouse clicks or keystrokes. It is possible to add control elements necessary for the project to the form, for example: buttons, input fields, images, text fields, drop-down lists, etc.

When the user performs any action on the form or one of the controls, an event is generated. The program responds to these events as specified in the code and handles the events as they occur.

At the same time, the creation of software is simplified thanks to such a designer, and it provides more opportunities and functions of the software for less development time.

3.3.2 Graphical software interface

Similarly to the mobile application, access control is required in the PC software, i.e., login via the entered IP address of the microcontroller, which is set in the web settings. So, for this functionality in the first form, such controls as a text input field, a confirmation button, an exit button and necessary tooltips were added. Also added several visuals, text and images. The result is shown in Figure 3.13.

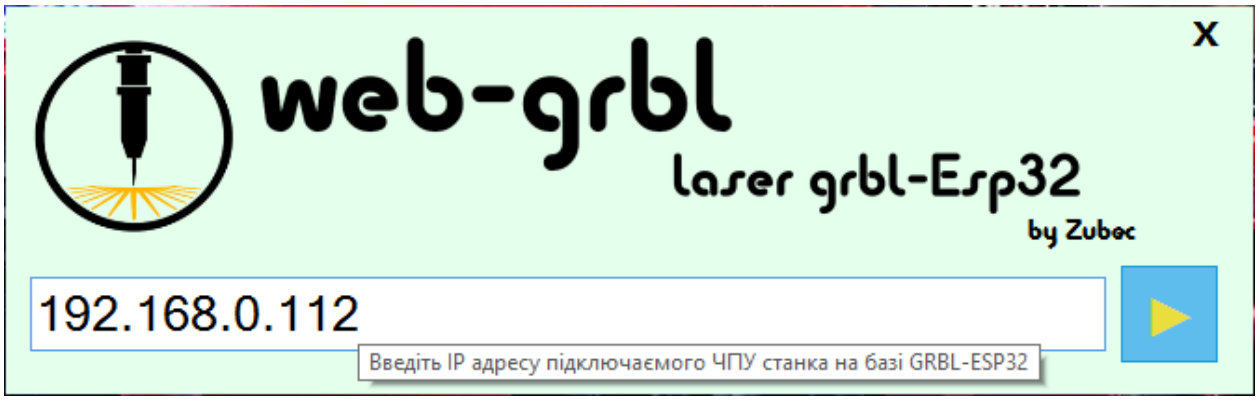


Fig. 3.13. Login form

The next step is the main screen, namely the second form, which is similar to the main panel of the web interface and the main screen of the mobile application. The main control is WebView2. Thanks to this control, it is possible to fully copy the functionality of the web interface, which provides ease of transition between devices. Figure 3.14 shows the projected shape of the main fragment.

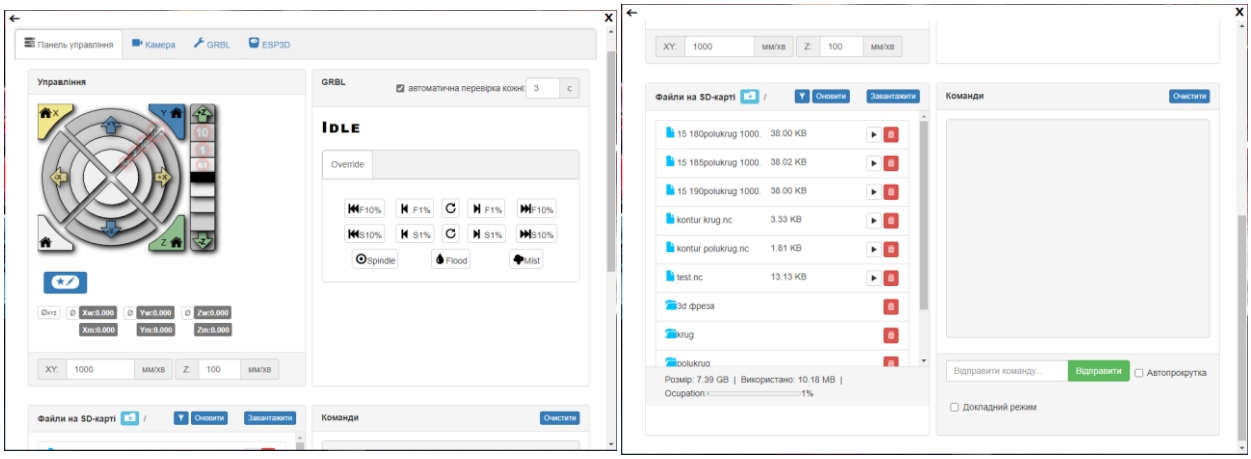


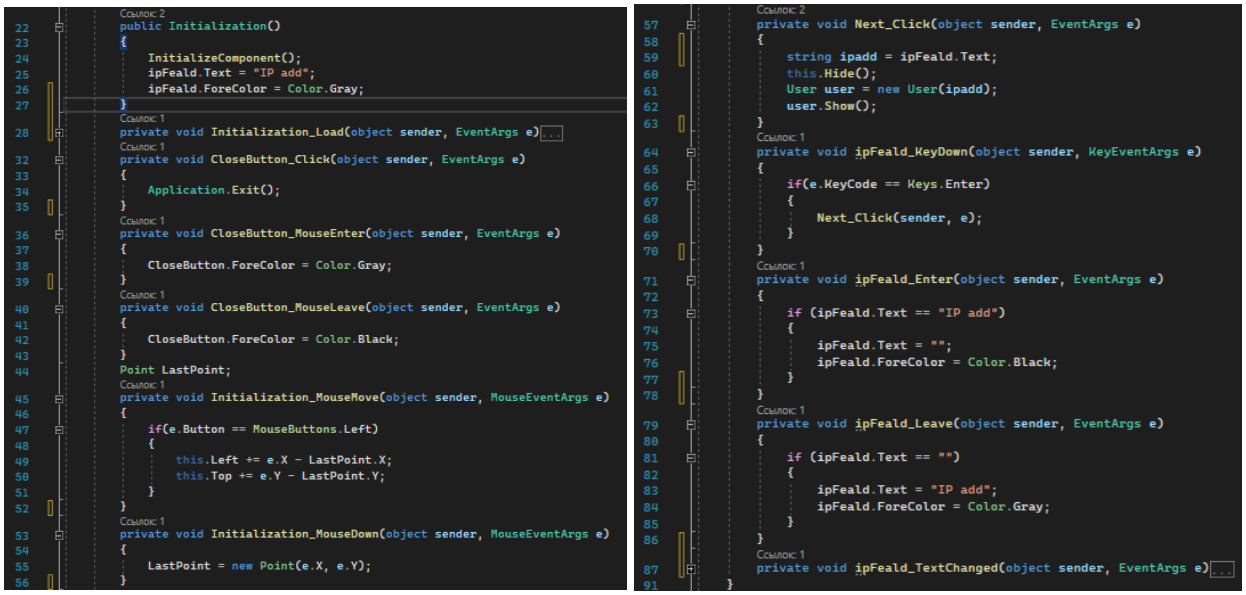
Fig. 3.14. The main form

3.3.3 Creating events for controls

The following events are required for the login form:

- To press the "Next" button ¬– verification of the entered address and connection at the given address.
- To press the "Enter" key - the same action as for pressing the button.
- To press the "X" button - closing the program.
- For the mouse cursor when hovering over functional buttons and fields – changing the cursor's appearance
- For a text field – sending the entered address to the next form for verification.
- When hovering over functional buttons – changing their shade for visual control.
- When pressing the functional buttons – changing their shade for visual control.

All these events were performed in code and this code is shown in Figure 3.15.



```
Ссылок: 2
public Initialization()
{
    InitializeComponent();
    ipFeald.Text = "IP add";
    ipFeald.ForeColor = Color.Gray;
}
Ссылок: 1
private void Initialization_Load(object sender, EventArgs e)...
Ссылок: 1
private void CloseButton_Click(object sender, EventArgs e)
{
    Application.Exit();
}
Ссылок: 1
private void CloseButton_MouseEnter(object sender, EventArgs e)
{
    CloseButton.ForeColor = Color.Gray;
}
Ссылок: 1
private void CloseButton_MouseLeave(object sender, EventArgs e)
{
    CloseButton.ForeColor = Color.Black;
}
Point LastPoint;
Ссылок: 1
private void Initialization_MouseMove(object sender, MouseEventArgs e)
{
    if(e.Button == MouseButtons.Left)
    {
        this.Left += e.X - LastPoint.X;
        this.Top += e.Y - LastPoint.Y;
    }
}
Ссылок: 1
private void Initialization_MouseDown(object sender, MouseEventArgs e)
{
    LastPoint = new Point(e.X, e.Y);
}
```

```
Ссылок: 2
private void Next_Click(object sender, EventArgs e)
{
    string ipadd = ipFeald.Text;
    this.Hide();
    User user = new User(ipadd);
    user.Show();
}
Ссылок: 1
private void ipFeald_KeyDown(object sender, KeyEventArgs e)
{
    if(e.KeyCode == Keys.Enter)
    {
        Next_Click(sender, e);
    }
}
Ссылок: 1
private void ipFeald_Enter(object sender, EventArgs e)
{
    if (ipFeald.Text == "IP add")
    {
        ipFeald.Text = "";
        ipFeald.ForeColor = Color.Black;
    }
}
Ссылок: 1
private void ipFeald_Leave(object sender, EventArgs e)
{
    if (ipFeald.Text == "")
    {
        ipFeald.Text = "IP add";
        ipFeald.ForeColor = Color.Gray;
    }
}
Ссылок: 1
private void ipFeald_TextChanged(object sender, EventArgs e)...
}
```

Fig. 3.15. Login form events

Also, the main form needs events corresponding to it. Namely:

- To press the "X" button - closing the program.
- To press the back arrow - return to the previous form.
- For the "WebView2" element – provision of the IP address from the login form and all functionality.

- For the mouse cursor when hovering over functional buttons and fields – changing the cursor's appearance
- When hovering over functional buttons – changing their shade for visual control.
- When pressing the functional buttons – changing their shade for visual control.

The events for the main form were implemented in code and this code is shown in Figure 3.16.

```
public partial class User : Form
{
    private String httpProtocolPrefix = "http://";
    private String httpsProtocolPrefix = "https://";
    private String url = "";
    Ссылок: 1
    public User(String url)
    {
        this.url = url;
        InitializeComponent();
    }
    Ссылок: 1
    private void User_Load(object sender, EventArgs e)
    {
        initBrowser();
    }
    Ссылок: 1
    private async Task initizated()
    {
        await webView21.EnsureCoreWebView2Async(null);
    }
    Ссылок: 1
    public async void initBrowser()
    {
        await initizated();
        String targetUrl = url;
        if (!url.Contains(httpsProtocolPrefix) || !url.Contains(httpProtocolPrefix)) {
            targetUrl = httpProtocolPrefix + url;
        }
        webView21.CoreWebView2.Navigate(targetUrl);
    }
    Ссылок: 1
    private void CloseButton_Click(object sender, EventArgs e)
    {
        Application.Exit();
    }
    Ссылок: 1
    private void CloseButton_MouseEnter(object sender, EventArgs e)
    {
        CloseButton.ForeColor = Color.Gray;
    }
    Ссылок: 1
    private void CloseButton_MouseLeave(object sender, EventArgs e)
    {
        CloseButton.ForeColor = Color.Black;
    }
    Point LastPoint;
    Ссылок: 1
    private void User_MouseMove(object sender, MouseEventArgs e)
    {
        if (e.Button == MouseButtons.Left)
        {
            this.Left += e.X - LastPoint.X;
            this.Top += e.Y - LastPoint.Y;
        }
    }
    Ссылок: 1
    private void User_MouseDown(object sender, MouseEventArgs e)
    {
        LastPoint = new Point(e.X, e.Y);
    }
    Ссылок: 0
    private void textBox1_TextChanged(object sender, EventArgs e)
    Ссылок: 1
    private void back_Click(object sender, EventArgs e)
    {
        this.Hide();
        Initialization initialization = new Initialization();
        initialization.Show();
    }
    Ссылок: 1
    private void back_MouseEnter(object sender, EventArgs e)
    {
        back.ForeColor = Color.Gray;
    }
    Ссылок: 1
    private void back_MouseLeave(object sender, EventArgs e)
    {
        back.ForeColor = Color.Black;
    }
```

Fig. 3.16. Events of the main form

3.4 Conclusions by section

As a result of the implementation of the third section of the qualification work, the implementation of the web interface, mobile application for Android OS and software for Windows OS was carried out at an arbitrary level. The web interface is designed to be used on any platform using any browser without any obstacles. The mobile application and software used the identical functionality of the web interface, and made it possible to log in by IP address.

The entire functionality of all three interfaces has been expanded and simplified compared to similar software for CNC machines based on Arduino and ESP32 microcontrollers, and has the potential for further development.

ДОДАТОК Е

ГРАФІЧНІ МАТЕРІАЛИ

Метою роботи є підвищення ефективності технологічного процесу адитивного виробництва за рахунок модернізації, удосконалення і доповнення післяпроцесної обробки.

Основна задача досліджень полягає в розробці программного забезпечення і веб-інтерфейсу для прискорення та спрощення використання методів післяпроцесної обробки адитивного виробництва.

Для досягнення зазначеної мети необхідно вирішити **наступні задачі**:

1. Провести аналіз роботи з МК ESP32 та реалізації сервер-клієнтської частини.
2. Спроектувати серверну частину МК.
3. Спроектувати WEB-інтерфейс для МК.
4. Визначитися з концепцією розробки ПЗ.
5. Розроити ПЗ для ОС Windows.
6. Розробити додаток для ОС Android.
7. Виконати технічне обґрунтування прийнятих рішень.

Об’єкт дослідження – процес післяпроцесної обробки адитивного виробництва.

Предмет дослідження – программа мікроконтроллера, програма для розробки програмних забезпечень.

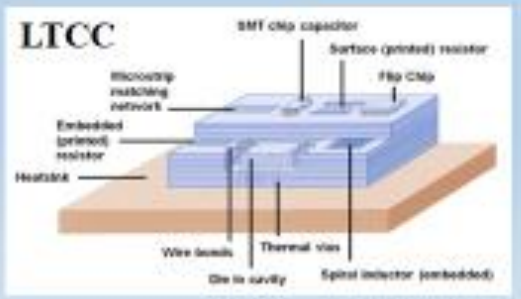
Адитивне виробництво – це процес виготовлення тривимірного об'єкту з цифрового файлу. Воно називається «адитивним», оскільки в загальному випадку технологія передбачає накладання послідовно тонких шарів матеріалу. Таким способом можна виготовляти деталі складних форм, що неможливо зробити традиційними методами лиття та механічної обробки або іншими методами.

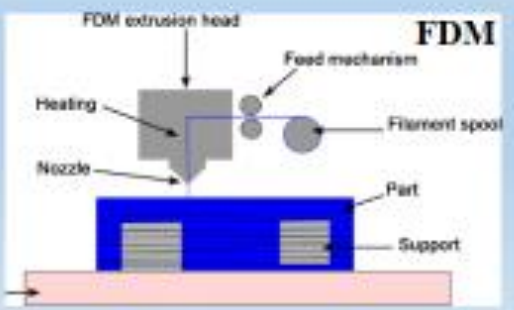
ТЕХНОЛОГІЇ АДИТИВНОГО ВИРОБНИЦТВА НВЧ-КОМПОНЕНТІВ

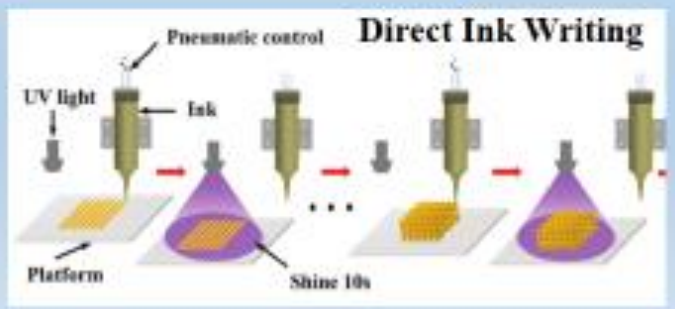
Використання в якості діелектричного матеріалу низькотемпературної керамікі з мінімальними втратами (**Low Temperature Co-fired Ceramics, LTCC**)

Метод пошарового наплавлення (**Fused Deposition Modeling, FDM**)

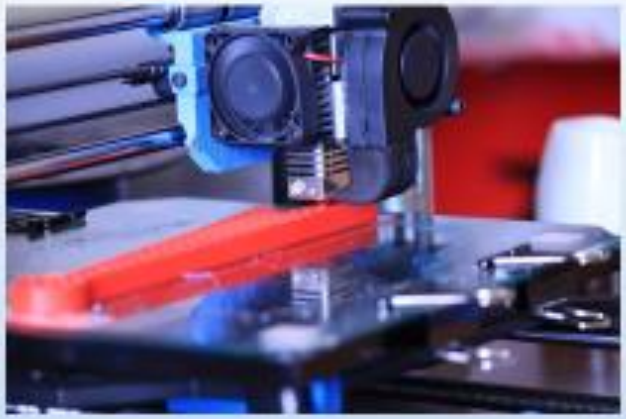
Робокастинг (**Direct Ink Writing, DIW**).





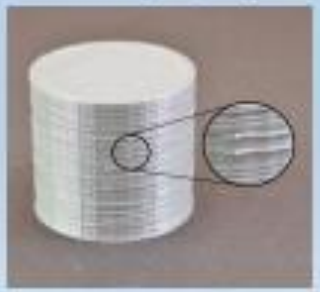


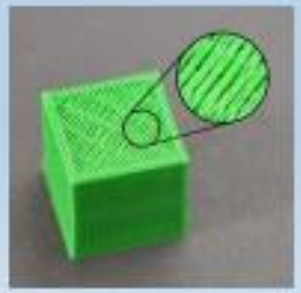
Метод пошарового наплавлення (Fused Deposition Modeling, FDM)



Також в залежності від складності 3D-моделі та специфікацій застосування технологій 3D-друку виникає необхідність у використанні спеціальних супортів.

Під час такого 3D-друку модель формується шар за шаром, тобто необхідний попередній шар для формування нового.





Один з основних недоліків використання супортів – необхідність фінішної (постпроцесної) обробки

Модифікація апаратного сегменту

Набір для зборки 2D-верстату лазерного гравірування на базі верстату S1

Вісь зсуву (Oz)

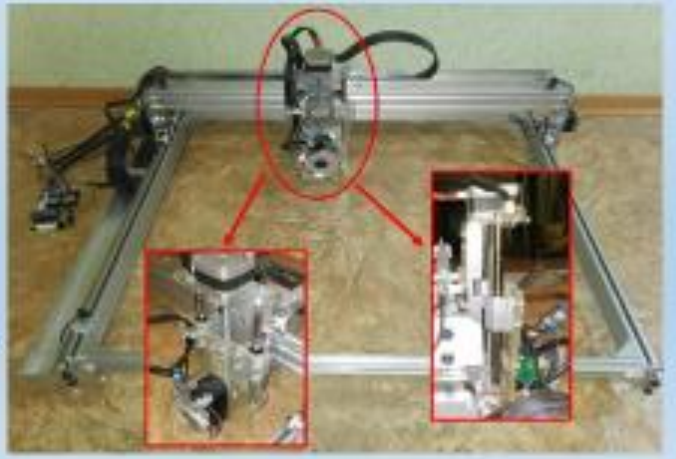
Лазерний модуль

Шпиндель

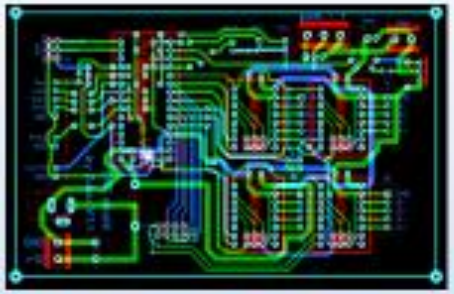
У початковому вигляді він непристосований для постобробки

Виконана його модифікація

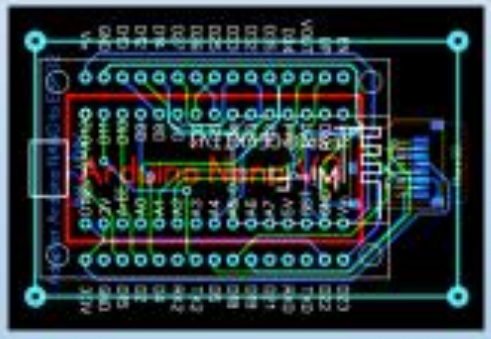
Додана третя вісь зсуву (Oz), встановлено кабель-канал і кінцеві вимикачі Ox і Oy, замінена кабельна система, для продуву робочої поверхні на драйвер лазеру змонтований вентилятор, забезпечена сумісна робота з програмою LaserGRBL, запропонован заміна мікроконтролера на більш потужніший і з можливістю роботи по Wi-Fi



Модифікація цифрового сегменту



Макет, який розроблений за допомогою ПЗ Sprint-Layout 6.0



Макет перехідної плати «Adapter Arduino Nano to ESP32», що розроблена в Sprint-Layout 6.0

Практична реалізація плата розширення для МК Arduino Nano та ESP32 з перехідною платою

Arduino Nano

ESP32

Аналіз та вибір мов програмування

Мова програмування для ESP32

Серверна частина: PHP та C++

Веб-інтерфейс: HTML та CSS

Програмне забезпечення та веб-інтерфейс для МК ESP32 були написані в ПЗ Visual Studio Code

Мова програмування мобільного додатку ОС Android

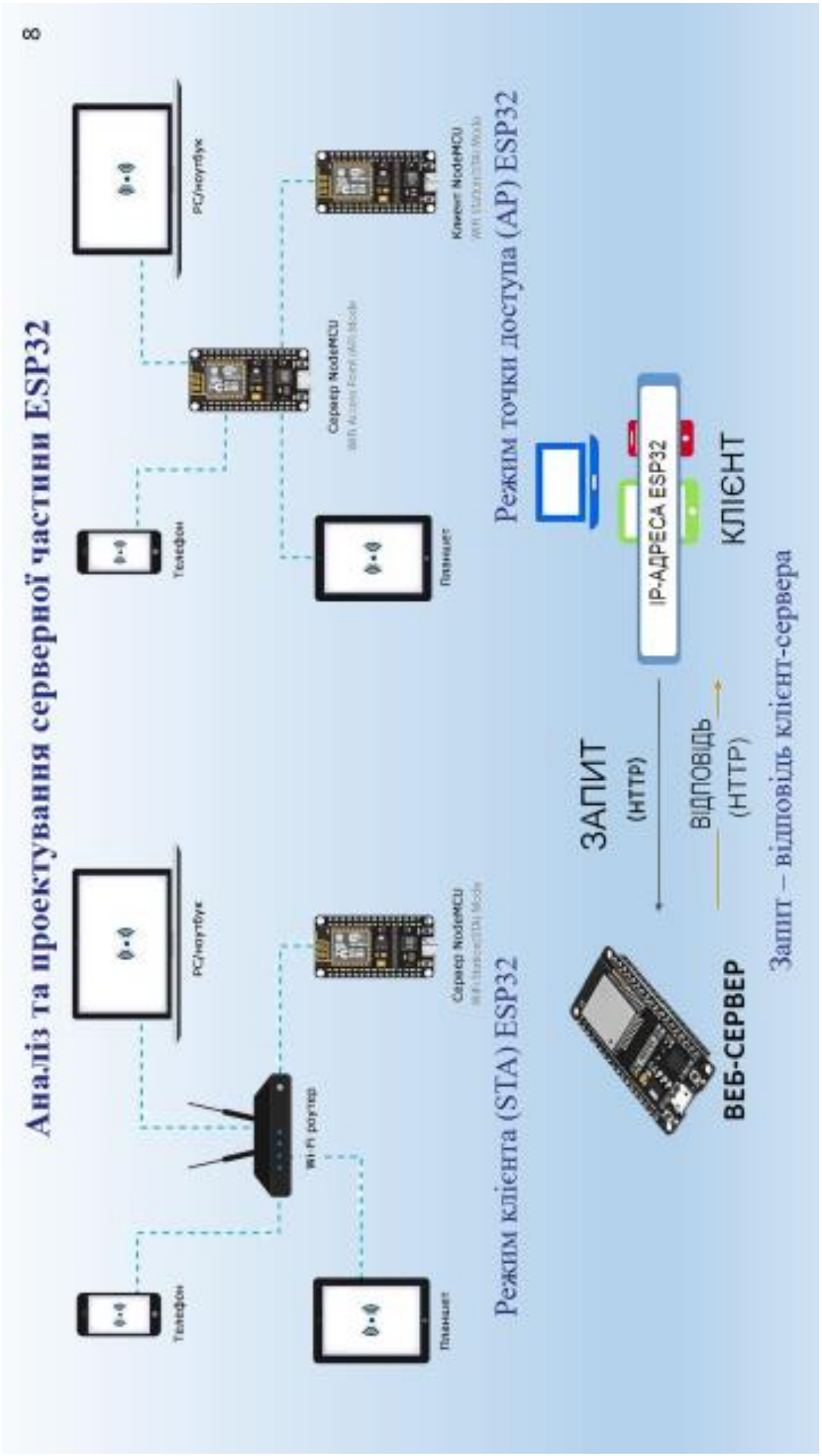
Kotlin

Мобільний додаток був розроблений в ПЗ Android Studio

Мова програмування ПЗ ОС Windows

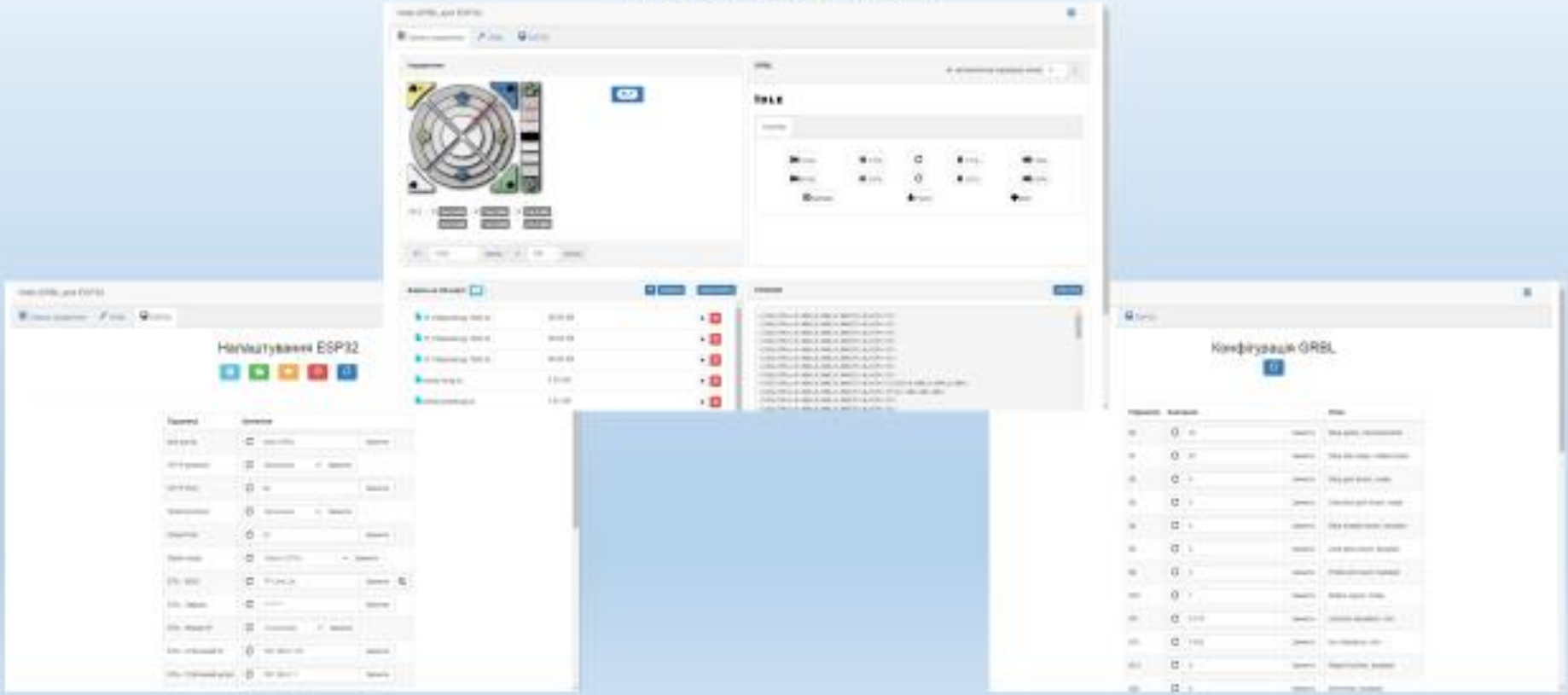
C#

Програмне забезпечення було розроблено в ПЗ Visual Studio


8
Аналіз та проектування серверної частини ESP32
PC/ноутбук
Телефон
Wi-Fi роутер
Сервер NodeMCU
Планшет
Режим клієнта (STA) ESP32
Режим точки доступа (AP) ESP32
ЗАПИТ
(HTTP)
ВІДПОВІДЬ
(HTTP)
IP-АДРЕСА ESP32
ВЕБ-СЕРВЕР
КЛІЄНТ
Запит – відповідь клієнт-сервера

Візуалізація веб-інтерфейсу

Функціонал веб-інтерфейсу більш обширніший ніж подібні інтерфейси та ПЗ для даної конфігурації верстата.



Візуалізація інтерфейсу мобільного додатку

Функціонал мобільного додатку аналогічний з функціоналом веб-інтерфейсу

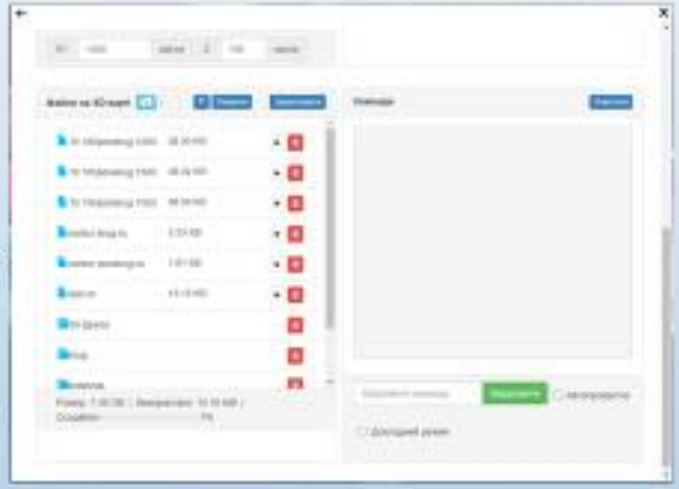
Макет дизайну підключення до верстату для темної та світлої теми смартфона

Макет головного екрану керування верстатом та налаштування

Візуалізація інтерфейсу програмного забезпечення ОС Windows

Функціонал ПЗ ОС Windows аналогічний з функціоналом веб-інтерфейсу

Форма входу



Форма головного екрану

Висновки

01 Проаналізовані та обрані мови програмування та середовища розробки

02 Проведено аналіз роботи мікроконтролера та його програмування

03 Спроектовано і реалізовано серверну та клієнтську частини мікроконтролера ESP32

04 Розроблено мобільний додаток ОС Android

05 Спроектовано та реалізовано та програмне забезпечення ОС Windows

Дякую за увагу!

web-grbl

laser grbl-Esp32

by Zubec