Зміст

[ВСТУП 4](#_Toc11224248)

[1 ОГЛЯД ІСНУЮЧИХ ТЕХНІЧНИХ РІШЕНЬ СИСТЕМИ КІНЦЕВОЇ ОБРОБКИ ДЕТАЛЕЙ З ПЛАСТИКУ ABS 5](#_Toc11224249)

[1.1 Опис Print Product CRYSTAL 6](#_Toc11224250)

[1.2 Опис Skymaker Magic Box 7](#_Toc11224251)

[2 ВИБІР ТЕХНІЧНИХ ЗАСОБІВ ДЛЯ КОНТРОЛЮ ПРОЦЕСІВ АЦЕТОНОВОЇ БАНІ 8](#_Toc11224252)

[2.1 Вибір мікроконтролера 9](#_Toc11224253)

[2.2 Вибір програмних засобів для програмування контролера 11](#_Toc11224254)

[2.3 Вибір програмних засобів для створення додатку до мобільної операційної системи Android 13](#_Toc11224255)

[2.2 Вибір схемо-технічних рішень 20](#_Toc11224256)

[3 РОЗРОБКА ПРОГРАМИ ДЛЯ МІКРОКОНТРОЛЕРА 22](#_Toc11224257)

[3.1 Перелік об’єктів, змінних та констант в програмі 23](#_Toc11224258)

[3.2 Перелік використаних функцій, та методів класу в програмі 24](#_Toc11224259)

[4 РОЗРОБКА ТА РОЗРАХУНКИ ДЛЯ ПРИНЦИПОВОЇ СХЕМИ З’ЄДНАНЬ МІКРОКОНТРОЛЕРА 27](#_Toc11224260)

[4.1 Розробка принципової схеми 27](#_Toc11224261)

[5. РОЗРОБКА ДОДАТКУ ДО ОПЕРАЦІЙНОЇ СИСТЕМИ ANDROID 32](#_Toc11224262)

[5.1 Опис класу MainWindow 32](#_Toc11224263)

[5.1.2 Методи класу MainWindow 34](#_Toc11224264)

[5.2 Опис класу BluetoothConnection 36](#_Toc11224265)

[5.2.1 Властивості класу BluetoothConnection 36](#_Toc11224266)

[5.2.2 Методи класу BluetoothConnection 37](#_Toc11224267)

## ВСТУП

Кінцева обробка ABS моделей пластику відіграє важливу роль в процесі виготовлення пластикових виробів, оскільки вона впливає на тактильне та візуальне сприйняття деталі, моделі, виробу тощо.

Оскільки принцип роботи 3D принтера полягає в пошаровому створенні моделі, то в результаті друку, на створеній моделі можна візуально побачити, і тактильно відчути надруковані принтером шари. Одним із способів усунення дефектів друку являється обробка парами ацетону, цей спосіб називається ацетоновою банею.

Ацетонова баня – метод пост обробки об'єкта друку, в якому виріб поміщають в середовище насичених парів того, чи іншого хімічного з'єднання для впливу на модель, після чого остання змінює свої властивості. Ацетонова баня для ABS робить деталь гладкою та глянсовою.

Для реалізації ацетонової бані треба підтримувати ацетон на необхідному рівні температури, та також необхідно циркулювати пари у середовищі обробки деталі.

Метою дипломного проектування є розробка пристрою для підтримування необхідних умов для обробки пластикової деталі, і також відлік часу до закінчення процесу. Пристрій повинен керуватися за допомогою програмного додатку для мобільної операційної системи Android.

При цьому необхідно розглянути і провести порівняльний аналіз існуючих технічних рішень.

# 1 ОГЛЯД ІСНУЮЧИХ ТЕХНІЧНИХ РІШЕНЬ СИСТЕМИ КІНЦЕВОЇ ОБРОБКИ ДЕТАЛЕЙ З ПЛАСТИКУ ABS

Багато студій які спеціалізуються на виробництві 3D моделей, також приватні особи які захоплюються 3D моделюванням, майстерні на виробництві які виробляють деталі, які використовуються у подальшому процесі виробництва, для усіх цих прикладів, а також багатьох інших є актуальним питання обробки пластику.

Існує декілька методів кінцевої обробки пластику:

* Полірування

Треба зачистити оброблювану поверхню шкурним матеріалом, змоченою ганчіркою видалити залишки зайвого пластику з деталі, обробити суху деталь на полірувальному апарату. До плюсів можна занести відсутність необхідності в розчинниках, та економічність, до мінусів : безпосередня участь людини, та також на оброблену поверхню погано наноситься фарба

* Ґрунтовка і фарбування

Після зачистки деталі наждачним папером, треба нанести аерозольну ґрунтовку, яка забезпечить гладку поверхню для подальшого нанесення фарби. Після нанесення ґрунтовки можна наносити фарбу, наносити треба в 2-4 шари, також після того як фарба висохне її треба надати процедурі поліровки. Цим способом можна досягти відмінних результатів, але він потребує ручної праці, також ґрунтовка і фарба збільшують розміри моделі, завдяки високоякісним ґрунтовці і фарбі, цей метод не можна назвати економічним.

* Обробка ацетоном

Для цього методу буде потрібна посудина з ацетоном, в котру занурюється модель, після цього модель треба висушити. Цей метод не витрачає багато часу, але дуже негативно сказається на деталізації моделі, також на моделі можуть залишитися білі розводи .

* 4. Ацетонова баня

При цьому методі використовуються пари ацетону, яких можна досягти підігрівши ацетон до певної температурі. До плюсів цього методу можна віднести кінцевий результат оброблювальної поверхні, швидкість процедури, та посередня участь людини в процесі обробки, до мінусів : негативно сказається на деталізації, міцності моделі, та також небезпечність займання ацетону.

Є декілька готових рішень які автоматизують процес описаний в методі ацетонової бані, а також в методі обробки ацетону, вони знижують небезпечність займання ацетону, одними з таких рішень являються Print Product CRYSTAL, а також Skymaker Magic Box.

## 1.1 Опис Print Product CRYSTAL

Пристрій Print Product Crystal призначений для пост обробки виробів, надрукованих за технологіями FDM і FFF. Установка підтримує найширший діапазон розчинників у порівнянні з аналогами: всі розчинники для ABS, де-лімонен, дихлоретан, діхлорметан, мурашину кислоту, сольвент. Обробка виробів відбувається парами розчинника, а не спреєвим розбризкування, як у інших установок. Розчинник нагрівається тенами, випаровується, і його пари обробляють виріб. Усередині камери установки розташовані вентилятори, які розганяють пари розчинника по всьому простору камери, дозволяючи їм обробити виріб з усіх можливих сторін.

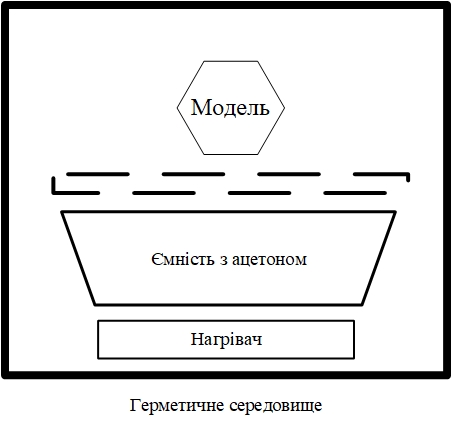


Рисунок 1.1 – Структурна схема ацетонової бані Print Product Crystal

## 1.2 Опис Skymaker Magic Box

Система Skymaker Magic Box призначена для обробки надрукованих пластикових моделей. Рекомендується використовувати ацетон як розчинник ABS пластику. Обробка поверхні відбувається кропленням поверхні моделі в короткий період часу. Завдяки системі збору ацетону, яка розташована під панеллю з оброблювальною моделлю, розчинник може використовуватися по декілька разів, що безумовно зменшує коштовність однієї процедури.

Також варто зазначити що процедура відбувається за значно коротший відрізок часу, від 10 до 60 секунд, що в декілька десятків разів менше в порівнянні з ацетоновою банею.

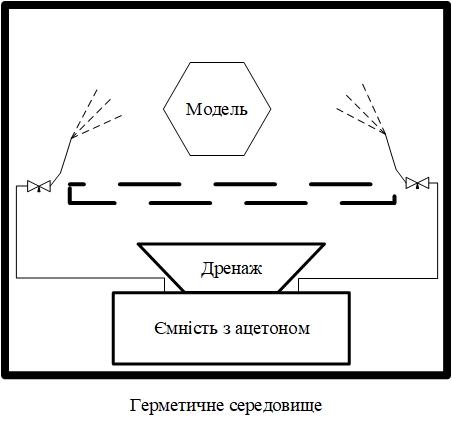


Рисунок 1.2 – Структурна схема ацетонової бані Skymaker Magic Box

# 2 ВИБІР ТЕХНІЧНИХ ЗАСОБІВ ДЛЯ КОНТРОЛЮ ПРОЦЕСІВ АЦЕТОНОВОЇ БАНІ

Для реалізації системи яка буде забезпечувати контроль процесів які відбуваються в ацетоновій бані, потрібен контролер, який буде керувати температурою нагрівача, відслідковувати температуру, вмикати, або вимикати вентилятор, також контролер повинен мати можливість бездротового зв’язку для обміну інформацією зі смартфоном на базі операційної системі Android.

## 2.1 Вибір мікроконтролера

Порівнюючи декілька контролерів різних специфікацій таких як ESP8266, та ESP32, було обрано плату побудовану на ESP32,оскільки цей контролер має досконаліші характеристики процесора, пам’яті, більшу різноманітність портів вводу/виводу, та також більш поширені можливості стосовно бездротового з’єднання.

Також має місце згадати про комерційні проекти котрі успішно використовують мікроконтролери на основі ESP32.

Особливості ESP32 включають в себе наступне:

* Процесори:
  + Процесор: Xtensa двоядерний (або одноядерної) 32-розрядний LX6 мікропроцесор, що працює на 160 або 240 МГц і виконує до 600 DMIPS
  + Ультра низька потужність (ОТП) співпроцесор
* Пам’ять: 520 Кб пам’яті SRAM
* Бездротовий зв’язок:
  + Wi-Fi: 802.11 b/g/N
  + Bluetooth: В4.2 БР/EDR і БЛЕ
* Периферійні інтерфейси:
  + 12-розрядний АЦП до 18 каналів
  + 2 × 8-біт ЦАП
  + 10 × сенсорних датчиків (ємнісних датчиків і контролерів)
  + Датчик температури
  + 4 × СВО
  + 2 × i2s для інтерфейсів
  + 2 × з I2C інтерфейси
  + 3 × UART з
  + SD/SDIO/CE-ATA/MMC/eMMC хост-контролер
  + SDIO/SPI підпорядкований контролер
  + Ethernet Mac інтерфейс з виділеними DMA і стандарти IEEE 1588 точного часу за протоколом підтримки
  + CAN bus 2.0
  + (передавач/приймач, до 8 каналів)
  + Можливість підключення двигунів та світлодіодів через ШІМ-вихід
  + Ультра низька потужність аналоговий передпідсилювач
* Безпека:
  + Стандарт IEEE 802.11 підтримує всі функції безпеки, у тому числі АБФ, захист WPA/WPA2 і ВАПІ
  + Безпечне завантаження
  + Шифрування флеш
  + 1024-бітний ключ, до 768 біт для клієнтів
  + Криптографічне апаратне прискорення: AES, SHA-2, RSA, криптографії на основі еліптичних кривих (ЕСС), генератор випадкових чисел (ГВЧ)
* Управління живленням:
  + Внутрішній низький регулятор відключення
  + Індивідуальний енергетичний домен для RTC
  + 5 мкА струм режиму "глибокий сон"
  + Прокидання з переривання від GPIO, таймера, вимірювання АЦП, переривання ємнісного сенсорного датчика

## 2.2 Вибір програмних засобів для програмування контролера

Існує багато програмних засобів, та мов програмування які можуть надавати можливість програмувати контролери сімейства ESP, а конкретно в цьому випадку контролер ESP32.

Мови програмування, платформи та середовища, що використовуються для програмування ESP32:

* Arduino IDE з ESP32 Arduino Core
* Espressif IoT Development Framework — Офіційна Espressif розробка для ESP32.
* Espruino — JavaScript SDK і прошивка майже замінює Node.js.
* Lua RTOS дляESP32
* Mongoose OS — Операційна система для підключених продуктів на мікроконтролерах;
* PlatformIO Ecosystem і IDE
* Pymakr IDE — IDE призначений для використання з пристроями Pycom;
* Simba Embedded Programming Platform
* Whitecat Ecosystem Blockly заснована на Web IDE
* MicroPython
* Zerynth — Python для IoT і мікроконтролерів, включаючи ESP32.

Як можна побачити засобів для реалізації задуманої ідеї дуже багато, є підтримка дуже популярних язиків програмування таких як: JavaScript, Python, C, C++. Також є можливість програмувати мікроконтролер використовуючи дуже просту платформа назва якої Whitecat Ecosystem Blockly, котра потребує лише будування блок-схеми алгоритму, чи використовування програмної мови Lua , але все ж таки не дає повного контролю над процесами.

Але все ж таки найбільш поширеними мовами програмування мікроконтролерів є C та С++. З перерахованих вище є декілька середовищ програмування які підтримують С та С++, найбільш популярною із впевненістю можна назвати Arduino IDE, котра вдячна своєму успіху дуже простій процедурі налаштування, та ненагромадженому інтерфейсу, та також дуже потужній підтримці зі сторони інших розробників, але існують й інші не менш вдалі пропозиції, одною з котрих являється PlatformIO.

PlatformIO – це програмний продукт з відкритим вихідним кодом, які націлений на розробку програм для мікроконтролерів та IoT пристроїв.

Перевагами PlatformIO над Arduino IDE, є можливість великого вибору налаштувань, але в той самий час і простота використання, швидкий доступ до всіх потрібних бібліотек, та налаштувань для різноманітних платформ. Також треба зазначити що PlatformIO має зручні інструменти, такі як : підказки можливих команд які можуть знадобитися під час написання програмного коду; створення готових кодових конструкцій, таких як цикли, умовні оператори, тіло функцій; підкреслювання синтаксичних помилок, та можливі засобі для їх виправлення; інструмент налагодження програми, цих та багатьох інших інструментів дуже бракує в Arduino IDE, що робить це рішення недостатньо функціональним в порівнянні з аналогом, та не зручним для розробки більш-менш професійних проектів. Саме тому в процесі розробки використовувалась PlatformIO.

## 2.3 Вибір програмних засобів для створення додатку до мобільної операційної системи Android

Android — операційна система і платформа для мобільних телефонів та планшетних комп'ютерів, створена компанією Google на базі ядра Linux. Підтримується альянсом Open Handset Alliance (OHA).

Хоча Android базується на ядрі Linux, він стоїть дещо осторонь Linux-спільноти та Linux-інфраструктури. Базовим елементом цієї операційної системи є реалізація Dalvik віртуальної машини Java, і все програмне забезпечення і застосування спираються на цю реалізацію Java.

Java Virtual Machine (скорочено Java VM, JVM) - віртуальна машина Java - основна частина виконує системи Java, так званої Java Runtime Environment (JRE). Віртуальна машина Java виконує байт-код Java, попередньо створений з вихідного тексту Java-програми компілятором Java (javac). JVM може також використовуватися для виконання програм, написаних на інших мовах програмування. Наприклад, вихідний код на мові Ada може бути откомпилирован в байт-код Java, який потім може виконатися за допомогою JVM.

JVM є ключовим компонентом платформи Java. Так як віртуальні машини Java доступні для багатьох апаратних і програмних платформ, Java може розглядатися і як сполучна програмне забезпечення, і як самостійна платформа. Використання одного байт-коду для багатьох платформ дозволяє описати Java як «скомпільовано одного разу, запускається всюди» (compile once, run anywhere).

Віртуальні машини Java зазвичай містять Інтерпретатор байт-коду, однак, для підвищення продуктивності в багатьох машинах також застосовується JIT-компіляція часто виконуваних фрагментів байт-коду в машинний код.

Програми, призначені для запуску на JVM, повинні бути скомпільовані в стандартизованому переносимому довічним форматі, який зазвичай представляється у вигляді файлів «.class». Програма може складатися з безлічі класів, розміщених в різних файлах. Для полегшення розміщення великих програм, частина файлів виду «.class» може бути упакована разом в так званому «.jar» -Файл (скорочення від «Java Archive»).

Віртуальна машина JVM виконує файли «.class» і «.jar», емулюючи дані в них інструкції наступними шляхами:

* інтерпретування;
* Використання JIT-компілятора (наприклад, такого як «HotSpot» від Sun Microsystems).

У наші дні, JIT-компіляція використовується в більшості JVM для досягнення більшої швидкості. Існують також ahead-of-time компілятори, що дозволяють розробникам додатків перекомпіліровать файли класів в рідній для конкретної платформи код.

Як і більшість віртуальних машин, Java Virtual Machine має stack-орієнтовану архітектуру, властиву мікроконтролерів і мікропроцесорів.

JVM - екземпляр JRE (Java Runtime Environment), що вступає в дію при виконанні програм Java; по завершенні виконання, цей екземпляр видаляється складальником сміття. JIT - та частина віртуальної машини Java, яка використовується для прискорення виконання програм; JIT одночасно компілює ті частини байт-коду, функціональність яких аналогічна, - що скорочує час, необхідний для проведення компіляції.

Додатки під операційну систему Android є програмами в нестандартному байт-коді для віртуальної машини Dalvik, для них був розроблений формат настановних пакетів .APK. Для роботи над додатками є безліч бібліотек: Bionic (бібліотека стандартних функцій, несумісна з glibc); мультимедійні бібліотеки на базі PacketVideo OpenCORE (підтримують такі формати, як MPEG-4, H.264, MP3, AAC, AMR, JPEG і PNG); SGL (движок двомірної графіки); OpenGL ES 1.0 ES 2.0 (движок тривимірної графіки); Surface Manager (забезпечує для додатків доступ до 2D / 3D); WebKit (готовий движок для веб-браузера; обробляє HTML, JavaScript); FreeType (движок обробки шрифтів); SQLite (легка СУБД, доступна для всіх додатків); SSL (протокол, що забезпечує безпечну передачу даних по мережі). У порівнянні зі звичайними додатками Linux, додатки Android підкоряються додатковим правилам [63]: Content Providers - обмін даними між додатками; Resource Manager - доступ до таких ресурсів, як файли XML, PNG, JPEG; Notification Manager - доступ до рядка стану; Activity Manager - управління активними додатками.

Google пропонує для вільного скачування інструментарій для розробки (Software Development Kit), який призначений для x86-машин під операційними системами Linux, macOS (10.4.8 або вище), Windows XP, Windows Vista і Windows 7. Для розробки потрібно JDK 5 або більше новий.

Розробку додатків для Android можна вести на мові Java (не нижче Java 1.5). Існує плагін для Eclipse - Android Development Tools (ADT), призначений для Eclipse версій 3.3-3.7. Також існує плагін для IntelliJ IDEA, який полегшує розробку Android-додатків, і для середовища розробки NetBeans IDE, який, починаючи з версії NetBeans 7.0, перестав бути експериментальним, хоч поки і не є офіційним. Крім того, існує Motodev Studio for Android - комплексна середовище розробки на базі Eclipse, що дозволяє працювати безпосередньо з Google SDK.

У 2009 році на додаток до ADT був опублікований Android Native Development Kit (NDK) - пакет інструментаріїв і бібліотек, що дозволяє реалізувати частину програми на мові С / С ++. NDK рекомендується використовувати для розробки ділянок коду, критичних до швидкості.

У 2013 році Google представила нову середу розробки Android Studio, засновану на IntelliJ IDEA від JetBrains.

У 2013 році відбувся реліз Embarcadero RAD Studio - XE5. Можливість розробки нативних додатків для платформи Android. Процес створення Android програми не вимагає додаткових пристроїв, крім, власне, Android-пристрої (в принципі, можна обійтися і емулятором).

Для розробки додатків для ОС Android, існує багато мов програмування, та платформ для їх застосування. Перелічимо декілька з них:

1. Java – офіційна мова програмування, яка підтримується середовищем розробки Android Studio. Велика кількість програмних додатків до Android створена саме на Java. При розробці на Java для Android використовуються не лише Java-класи, але й також файли маніфесту на мові XML, які представляють операційній системі інформацію про додаток, і також системи автоматичної збірки Gradle, Maven та Ant, вказівки до котрих пишуться на мовах Groovy, POM и XML.
2. Kotlin – ця мова позиціонується розробниками Android як другий офіційний язик програмування для Android після Java, але більш спрощеним для сприйняття. Kotlin зворотно сумісний з Java, та не викликає зниження потужності та збільшення розміру файлів. Відмінність від Java в тому, що ця мова потребує меншого службового втручання розробника, завдяки цьому більш легка для читання. Також можна сказати що Kotlin менш вимогливий до синтаксису, та пробачає такі поширені помилки як забутий знак «;».
3. Lua  — швидка і компактна скриптова мова програмування, розроблена підрозділом Tecgraf Католицького університету Ріо-де-Жанейро (Computer Graphics Technology Group of Pontifical Catholic University of Rio de Janeiro in Brazil). Є вільно-поширюваною, з відкритим сирцевим кодом на мові Сі.За можливостями, ідеологією і реалізацією, мова найближча до JavaScript, проте Lua відрізняється могутнішими і набагато гнучкішими конструкціями, спроектованими з метою «не плодити сутності понад необхідне». Хоча Lua не містить поняття класу і об'єкта в явному вигляді, механізми об'єктно-орієнтованого програмування з підтримкою прототипів (включаючи множинне успадкування) легко реалізуються з використанням метатаблиць, які також дозволяють перевантаження операцій, тощо. Реалізована модель ООП (як і в JavaScript) — прототипна. Lua значно простіша за Java, та з допомогою Corona SDK робить працю з цією мовою легкою.
4. JavaScript – за допомогою Adobe Phone Gap також можна створювати Android-додатки. Phone Gap, дозволяє розробляти додатки до Android, за допомогою технологій web-розробки. Зроблений додаток відображатиметься через WebView, але в обгортці мобільного додатку.
5. С/С++ - низькорівнева мова яка також підтримується Android Studio. та також завдяки іншим фреймворкам надає можливість створювати додатки для Android. Завдяки Android NDK, котрий дозволяє запускати додаток не через Java Virtual Machine, а безпосередньо через сам пристрій. Це також означає що є можливість користуватися зовнішніми бібліотеками С/С++. Одним з таких прикладів являється Qt, який надає дуже великий діапазон бібліотек з функціями і класами, які полегшують розробку програм на С/С++, та надають доступ до багатьох інших інструментів, таких як створення графічного інтерфейсу. За допомогою Qt можна створювати інтерфейс декількома засобами, одним з котрих являється Qt Widgets, що дозволяє в короткий проміжок часу створити простий на вигляд, але тим не менш функціональний інтерфейс, за основу котрого взята технологія слотів-сигналів. Другим засобом для створення інтерфейсу являється Qt Quick в основі цього засобу лежить технологія QML. QML-документ являє собою дерево елементів. QML елемент, так само, як і елемент Qt, являє собою сукупність блоків: графічних (таких, як rectangle, image) і поведінкових (таких, як state, transition, animation). Ці елементи можуть бути об'єднані, щоб побудувати комплексні компоненти, починаючи від простих кнопок і повзунків і закінчуючи повноцінними застосунками, що працюють з інтернетом. QML елементи можуть бути доповнені стандартними JavaScript-вставками шляхом вбудовування .js файлів. Також вони можуть бути розширені C++ компонентами через Qt framework. Також ці два засоби можуть бути об’єднані в одному проекті.

Завдяки перевагам які надає технологія Qt, та властивостям мови програмування С++, було обрано використовувати цю зв’язку для створення програмного додатку для операційної системи Android.

# 3 ВИБІР СХЕМО-ТЕХНІЧНИХ РІШЕНЬ

Оскільки мікроконтролер ESP32 потребує живлення від 3.3 до 5 В, то він не має можливості живити такі пристрої як нагрівач, та вентилятор. Тому було вирішено використовувати транзистори в ключовому режимі, та керувати відкриття транзистору за допомогою ШІМ-модуляції.

Широтно-імпульсна модуляція ( PWM - Pulse Width Modulation ) - це спосіб завдання аналогового сигналу цифровим методом , тобто з цифрового виходу, що дає тільки нулі і одиниці отримати якісь плавно змінюються величини.

При ШІМ ми подаємо на вихід сигнал, що складається з високих і низьких рівнів, тобто нулів і одиниць. А потім це все пропускається через інтегруючу ланцюжок. В результаті інтегрування на виході буде величина напруги, що дорівнює площі під імпульсами. Змінюючи шпаруватість (відношення тривалості періоду до тривалості імпульсу) можна плавно змінювати цю площу, а значить і напруга на виході. Таким чином, якщо на виході суцільні 1, то на виході буде напруга високого рівня (наприклад, 5 вольт), якщо нулі, то нуль. А якщо 50% часу буде високий рівень, а 50% низький то 2,5 вольт.

Для вибору необхідного нагрівача нам потрібно підібрати нагрівач який зможе забезпечувати максимальну температуру в 56 ℃ на протязі тривалого часу.

Для цих цілей підійде керамічний нагрівач, який потребує живлення 12 В. Нагрівальний елемент знаходиться в герметичній металевій капсулі. За рахунок більшої ефективності керамічного нагрівального елемента, по відношенню до резистора, температурні коливання мають меншу амплітуду. Нагрівач потребує живлення в 12 В, має внутрішній опір 3,5 Ом, та може споживати до 40 Вт електроенергії.

Так як струм на нагрівачу буде обмежено до 2 А , в такому випадку знадобиться радіатор котрий зможе рівномірно розсіяти тепло під ємністю з ацетоном.

Pн = I \* V (1)

Pн = 2 \* 12 = 24 Вт

(2)

Тобто, нагрівач потребує радіатор з тепловим опором в 2.3 К/Вт.

Також для побудування пристрою знадобиться вентилятор, котрий буде циркулювати пари ацетону по всій камері, тим самим обробляючи ацетоном поверхню моделі з усіх сторін. Але так як система працює безпосередньо з ацетоном, який має властивості горіння, то необхідно обрати вентилятор який побудований на основі безколекторного електричного двигуна. Оскільки електричні колекторні двигуни завдяки механізму з щітками можуть утворювати іскри, а так як вентилятор повинен знаходитися в одному середовищі з парами ацетону, то це може призвести до вибуху цих самих парів, саме тому потрібен вентилятор з безколекторним двигуном, котрий позбавлений таких недоліків.

Щодо вибору транзисторів за допомогою котрих буде керуватися нагрівач, та вентилятор, то він має відповідати таким вимогам:

1. Іе> 2 A
2. Uке  > 12 В
3. Uбе  > 5 В

Одним з багатьох транзисторів які задовольняли цим вимогам була модель 2SD965, яка має наступні характеристики:

* Напруга колектор-база: < 40 В
* Напруга колектор-емітер: < 20 В
* Напруга емітер-база: < 7 В
* Постійний струм колектора: < 5 А
* Піковий струм колектора: < 8 А
* Коефіцієнт h21e (β) : 180

## 3.1 Розробка та розрахунки функціональної схеми з’єднань мікроконтролера

### 3.1.1 Розробка функціональної схеми

Оскільки живлення нагрівача, та вентилятора відбувається від джерела напруги 12 В, а мікроконтролер, в свою чергу живиться від джерела в 5 В, то схема побудована на транзисторах, які працюють в режимі ключа, де на колектор подається 12 В напруги, и а напруга на базі керується мікроконтролером.

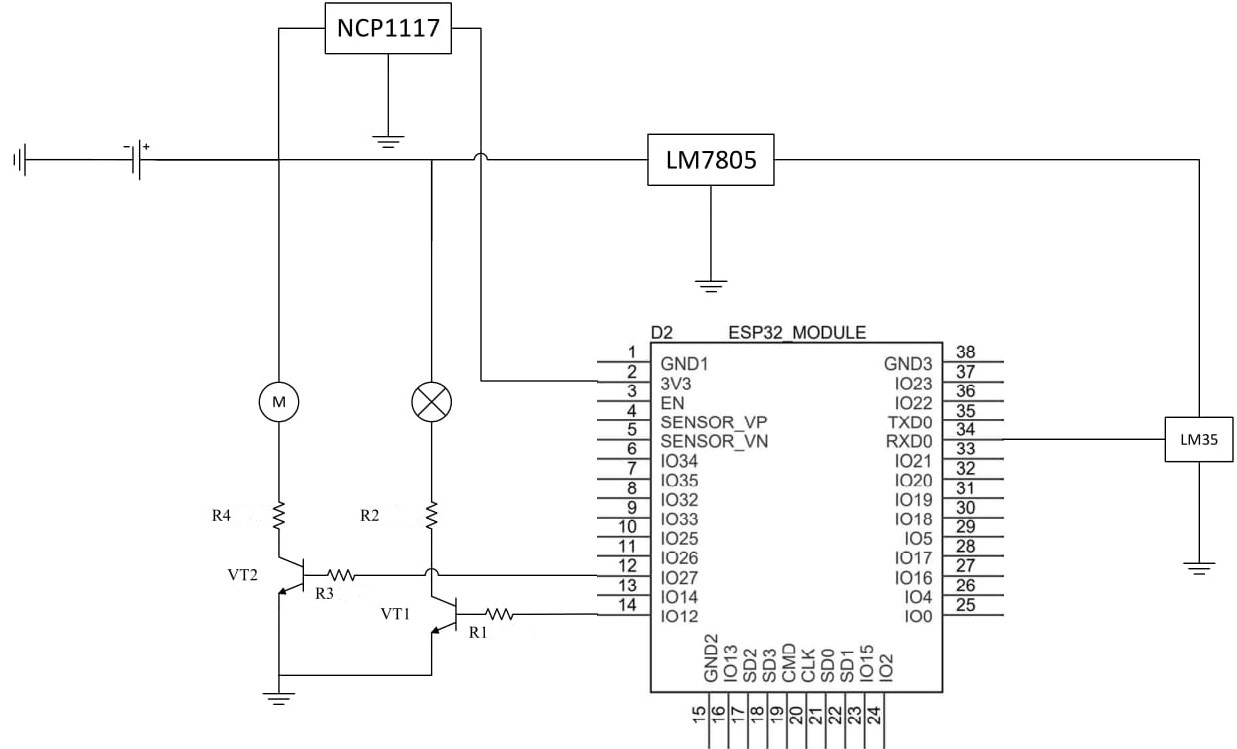


Рисунок 4.1 Принципова схема пристрою

Оскільки було обрано транзистор КТ8101, то розрахунки будуть проводитися опираючись на наступні характеристики:

1. Іе> 16 A
2. Uкб < 200 В
3. Uке  > 160 В
4. h21э = 20

Розрахунок транзистора VT1:

Розрахунки повільні задовольняти наступних умов:

1. Uк = 12 В
2. I­к = 2 А

Оскільки напруга на базі буде в діапазоні 0-5 В, середнє значення в 3 В.

Тоді :

Підберемо найближчі існуючі номінали:

Розрахунок транзистора VT2:

Розрахунки повільні задовольняти наступних умов:

1. Uк = 12 В
2. I­к = 0,14 А

Оскільки напруга на базі буде в діапазоні 0-5 В, середнє значення в 3 В.

Тоді :

Підберемо найближчі існуючі номінали:

Промоделюючи схему з отриманими розрахунками, отримаємо наступний результат.

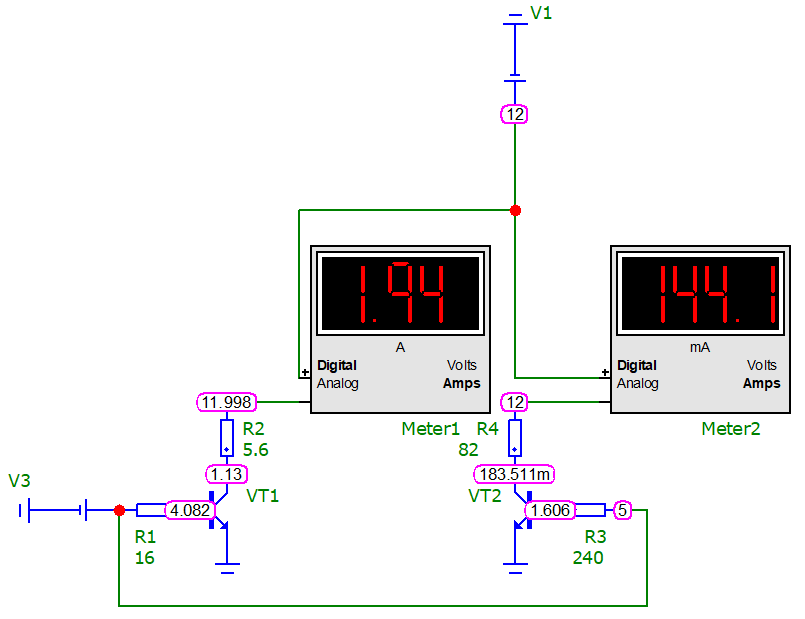


Рисунок 4.2 Модель розрахованої схеми в MicroCap

Як можна побачити відображена модель задовольняє усім умовам.

# 4 РОЗРОБКА ПРОГРАМИ ДЛЯ МІКРОКОНТРОЛЕРА

Центральним пристроєм у розробці є мікроконтролер ESP32, який служить для контролювання процесів які відбуваються в середовищі ацетонової бані. Додатковим, але не менш важливим, є додаток для операційної системи Android, через котрий відбувається налаштування характеристик процесу обробки пластика, та у реальному часі відображається важливі показники, такі як температура в середовищі з ацетоном, та час до закінчення процесу.

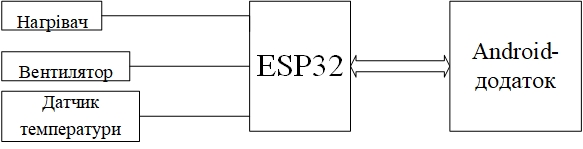


Рисунок 3.1 Структурна схема ацетонової бані

Структурно програма має дві необхідні функції:

1. setup()
2. loop()

Вони відрізняються тим, що функція setup() запускається лише один раз, в самому початку роботи контролера, а функція loop() запускається одразу після завершення виконування setup(),та запускається циклічно, тобто після того як всі команди функції loop() було виконано, то вона запускається знову, і знову, аж до самого кінця роботи контролера.

Як можна зрозуміти з назви функції setup() (з англійської мови setup – налаштування), у цій функції прийнято створювати об’єкти, ініціалізувати змінні, та константи які будуть використовуватися в процесі виконання програми в функції loop() (з англійської мови loop – коло).

## 3.1 Перелік об’єктів, змінних та констант в програмі

В програмі використовувалися наступні об’єкти, та змінні з константами:

* BluetoothSerial ESP\_BT – об’єкт класу BluetoothSerial за допомогою якого здійснюється з’єднання з клієнтом, та відбувається обмін інформацією, та командами.
* hw\_timer\_t \*timer – покажчик класу hw\_timer\_t, за допомогою котрого програма оперує таймером. Таймер потрібен для відліку часу який був заданий для виконання процесу обробки ацетоном.
* String command – об’єкт класу String, який представляє текстовий рядок, який за допомогою описаних в класі методів полегшує процес обробки тексту.
* int temp – змінна типу int, яка містить температуру яка була задана з Android-додатку, використовується для порівняння значень з датчика температури.
* int duration – змінна типу int, яка містить значення часу в хвилинах, отриманого з Android-додатку.
* int tempSensor - змінна типу int, яка містить значення отримане з датчика температури, значення може коливатися в інтервалу від 0 до 4096, потім це значення, за допомогою формули, конвертується у градуси Цельсія.
* int seconds - змінна типу int, еквівалент змінної duration, але містить секунди, замість хвилин.
* bool processRunning - змінна типу bool, яка може зберігати лише значення true чи false, вона містить інформацію про стан процесу обробки, якщо processRunning – true, це значить що обробка в процесі виконання.
* bool fanRunning - змінна типу bool, яка зберігає інформацію про стан вентилятора, якщо fanRunning – true, це значить що вентилятор працює.

## 3.2 Перелік використаних функцій, та методів класу в програмі

Функції, та методи класів які використовуються в процесі виконання програми:

* void IRAM\_ATTR onTimer() – функція виконується кожного разу коли таймер відраховую одну секунду.
* void checkConnection() – при визові функції вона буде виконуватися, доки не буде під’єднаний Bluetooth-клієнт, після цього вона відсилає повідомлення на Android-додаток, який буде зрозумілий як готовність контролера до прийняття команд.
* void setup() – функція налаштування, вона викликається один раз в самому початку роботи програми.
* void loop() – функція яка циклічно повторюється до завершення програми, основний процес виконується в цій функції.
* Serial.begin(unsigned long baud) - метод класу Serial, який вмикає передачу інформації через послідовний порт, та задає частоту опитування.
* ESP\_BT.begin(String localName) – метод класу BluetoothSerial, який приймає об’єкт типу String на вхід, та вмикає Bluetooth на контролері, та надає Bluetooth-пристрою назву яка вказана в аргументі.
* ESP\_BT.hasClient() = метод класу BluetoothSerial, який повертає значення типу bool, в залежності від того чи під’єднано Bluetooth-клієнт.
* size\_t Print:: println(const String &s) – метод класу Print, який приймає константне посилання типу String, як аргумент, та друкує його в потік повідомлень, у цій програмі клас Print унаслідується двома класами Serial та BluetoothSerial.
* hw\_timer\_t \* timerBegin(uint8\_t timer, uint16\_t divider, bool countup) – функція яка приймає на вхід три параметри: timer – порядок таймера(потрібен в випадку існування декількох таймерів); divider – значення на яке ділиться частота вбудованого в контролер таймера; countup – дозволяє відрахування, якщо true.
* void timerAttachInterrupt(hw\_timer\_t \*timer, void (\*fn)(void), bool edge) - функція яка приймає на вхід три параметри: \*timer – вказівник на таймер над котрим потрібно зробити дію; void (\*fn)(void) – вказівник на функцію котра буде відпрацьовувати при відрахуванні таймером потрібного часу.
* void timerAlarmWrite(hw\_timer\_t \*timer, uint64\_t alarm\_value, bool autoreload) - функція яка приймає на вхід три параметри: \*timer – вказівник на таймер над котрим потрібно зробити дію; alarm\_value – дільника на котрий ділиться частота таймера, autoreload – містить значення bool, яке надає можливість таймеру на повторення, якщо дорівнює true.
* void timerAlarmEnable(hw\_timer\_t \*timer) - функція яка приймає на вхід вказівник на таймер, на запускає його.
* void pinMode(uint8\_t pin,uint8\_t mode) - функція яка приймає на вхід два значення: pin – номер контакту; mode – режим роботи для контакту з номером pin.
* uint16\_t analogRead(uint8\_t pin) – функція яка повертає значення з аналогового контакту, номер якого зберігається в pin.
* int Serial::available() – метод класу Stream, від якого унаслідується клас BluetoothSerial, повертає значення типу int, в залежності від того чи доступне нове повідомлення в послідовному порту Bluetooth.
* int Serial::read() – метод класу Stream, від якого унаслідується клас BluetoothSerial, повертає значення типу int, яке відповідає літері англійського алфавіту.
* unsigned char String::endsWith(const String &suffix) – метод класу String, який приймає константне посилання на об’єкт String, як аргумент, та повертає значення unsigned char, в залежності від того чи співпадає кінець рядка із рядком suffix.
* void String::setCharAt(unsigned int index, char c) - метод класу String, який приймає два параметри: index – номер в рядку на який треба замінити символ; c – символ на котрий змінюється символ під номером index.
* long String::toInt() const – метод класу String, який повертає конвертоване значення рядку в тип даних int.
* int String::lastIndexOf(char ch) const - метод класу String, який приймає змінну типу char,як аргумент, та повертає індекс останнього символу який дорівнює ch
* unsigned char String::startsWith(const String &prefix) - const - метод класу String, який приймає константне посилання на об’єкт типу String, та повертає значення типу unsigned char, в залежності від того чи дорівнює початок рядка з рядком на котрий вказує prefix.
* void String::trim() - метод класу String, який прибирає усі пробіли в рядку.
* void digitalWrite(uint8\_t pin, uint8\_t val) – функція, яка приймає два значення, як аргументи, та подає на контакт з номером pin, значення яке залежить від змісту змінної val.

# 5 РОЗРОБКА ДОДАТКУ ДО ОПЕРАЦІЙНОЇ СИСТЕМИ ANDROID

Так як управління контролером має бути реалізовано через Android-додаток, то було вирішено створити програму за допомоги фреймворка Qt, використовуючи мову програмування С++.

При розробці на Qt варто знати про таку технологію як сигнали-слоти, ця технологія дозволяє з’єднати методи класу с якоюсь подією, де подія виступає сигналом, а метод – слотом, також до одного сигналу може буди приєднано декілька слотів, так само як і з’єднання одного слоту з декількома сигналами. Тому в переліку методів будуть переважати слоти.

Програма керується двома класами, кожен клас фактично відповідає за відображення вікна, та усіх процесів які в ньому відбуваються.

Оскільки класи два, то й вікон інтерфейсу також два.

Перелік класів у додатку:

* MainWindow
* BluetoothConnection

## 5.1 Опис класу MainWindow

Класс MainWindow використовується для задання параметрів для процесу обробки пластику, також цей клас керує інтерфейсом в головному вікні програми.

5.1.1 Властивості класу MainWindow

Перелік властивостей класу MainWindow:

* Ui::MainWindow \*ui – вказівник типу MainWindow,через нього відбуваються усі маніпуляції з інтерфейсом, такі як змінення надписів, форм об’єктів, та їх властивостей.
* BluetoothConnection window – об’єкт класу BluetoothConnection, який використовується для виклику другого вікна з інтерфейсом.
* QBluetoothSocket \*socket - об’єкт класу QBluetoothSocket, цей клас використовується для маніпуляцій через протокол зв’язку Bluetooth, за допомогою цього об’єкту програма отримує інформацію з контролера, та відсилає команди необхідні для керування процесом.
* QString addressToConnect – об’єкт класу QString, який містить MAC-адрес пристрою Bluetooth, з котрим відбувається обмін інформацією, та керування.
* QString command - об’єкт класу QString, котрий містить команди які відправляються до мікроконтролера з метою управління.
* QString receivedInfo = об’єкт класу QString, в котрий записуються отримані від мікроконтролера дані.
* QTimer \*tmr – вказівник об’єкту QTimer, за допомогою якого відбувається керування таймером. Таймер використовується для перевірки з’єднання смартфону з мікроконтролером, через заданий проміжок часу.
* bool errorShowed – змінна типу bool, яка містить інформацію щодо відображення повідомлення про помилку, для запобігання повторного відображення.
* bool running - змінна типу bool, яка містить інформацію про стан процесу обробки пластику.
* bool paused – змінна типу boll, яка містить інформацію про те, чи знаходиться процес на паузі.

### 5.1.2 Методи класу MainWindow

Перелік методів класу MainWindow:

Методи:

Публічні:

* **MainWindow**(QWidget \*parent = *nullptr*) – метод є конструктором класу MainWindow, в цьому методі ініціалізуютсья об’єкти, такі як socket,\*tmr, також відбувається з’єднання слотів з потрібними сигналами.
* ~**MainWindow**() - метод є деструктором класу MainWindow, в цьому методі звільнюється пам'ять яка була виділена для динамічних об’єктів, та також переривається Bluetooth з’єднання, якщо таке мається.

Приватні:

* void **connectToDevice**(*const* QString& str) – в цьому методі відбувається з’єднання Android-смартфону з мікроконтролером через протокол Bluetooth, та також задається властивості для цього з’єднання.

Слоти:

* void **on\_StartStopButton\_clicked**() – цей слот з’єднано з сигналом який визивається у випадку натискання на кнопку Start/Stop в інтерфейсі. При натисканні на Start посилаються відповідні команди, які ініціюють початок обробки на мікроконтролері, у випадку натиску на Stop відправляються команди які завершують процес обробки.
* void **updateState**() - цей слот з’єднано з сигналом таймера, котрий надсилається у випадку коли таймер відраховує заданий час. Цей слот виконує перевірку наявності з’єднання з мікроконтролером, у випадку коли з’єднання було перервано відображується відповідне повідомлення.
* void **getAddress**(*const* QString& str) - цей слот з’єднано з сигналом котрий надсилає MAC-адресу Bluetooth-пристрою до якого треба приєднатися. Цей метод приймає цей адрес, та ініціює процедуру з’єднання через метод **connectToDevice**()
* void **controllerReader**()- цей слот з’єднано з сигналом котрий надсилається кожного разу як з мікроконтролера приходить нове повідомлення з даними. Цей метод приймає отримані дані, виконує процедуру парсинга, та на основі отриманих значень виводить інформацію на елементи інтерфейсу.
* void **on\_PairButton\_clicked**() - цей слот з’єднано з сигналом котрий надсилається у випадку натиску на кнопку Pair, при цьому показується друге вікно з активними пристроями Bluetooth.
* void **on\_pauseButton\_clicked**() - цей слот з’єднано з сигналом котрий надсилається у випадку натиску на кнопку Pause, та надсилає команду паузи на мікроконтролер.

## 5.2 Опис класу BluetoothConnection

Клас BluetoothConnection використовується для знаходження доступних для з’єднання Bluetooth-пристроїв, також використовується для керування інтерфейсу в другому вікні.

### 5.2.1 Властивості класу BluetoothConnection

Перелік властивостей класу BluetoothConnection:

* QString addressToConnect – об’єкт класу QString, який містить інформацію про адресу приєднаного пристрою.
* Ui::BluetoothConnection \*ui – вказівник, за допомогою якого здійснюються зміни об’єктів інтерфейсу.
* QBluetoothDeviceDiscoveryAgent \*agent – вказівник класу QBluetoothDeviceDiscoveryAgent, за допомогою якого знаходяться доступні для приєднання Bluetooth-пристрої, та також надає інформацію про ці пристрої.
* *struct* **BlDevice**{QString name;QString address;} – структура BlDevice, використовується для збереження інформації про знайдені Bluetooth-пристрої.
* std::vector<BlDevice> devices – вектор типу BlDevice,в якому зберігаються усі знайденні Bluetoth-пристрої;
* BlDevice discovered – об'єкт типу BlDevice, використовується для заповнення вектору devices.

### 5.2.2 Методи класу BluetoothConnection

Перелік методів класу BluetoothConnection:

Публічні методи:

* *explicit* **BluetoothConnection**(QWidget \*parent = *nullptr*) – метод є конструктором класу BluetoothConnection, запускається процедура пошуку доступних Bluetooth-пристроїв, також відбувається з’єднання слотів з потрібними сигналами.
* ~***BluetoothConnection***()-метод є деструктором класу MainWindow, в цьому методі звільнюється пам'ять яка була виділена для динамічних об’єктів, та також зупиняється процедура пошуку Bluetooth-пристроїв.

Приватні методи:

* void **sendAddress**() – цей метод відправляє сигнал з MAC-адресою, який приймається у класі MainWindow.

Слоти:

* void **deviceDiscovered**(*const* QBluetoothDeviceInfo &device) – цей метод виконується коли знаходиться новий Bluetooth-пристрій, він приймає об’єкт класу QBluetoothDeviceInfo як параметр, та заповнює список знайдених пристроїв.
* void **on\_exitButton\_clicked**() – цей метод виконується у випадку натискання на кнопку Exit, він скриває вікно з пошуком Bluetooth-пристроїв.
* void **on\_connectButton\_clicked**() - цей метод виконується у випадку натискання на кнопку Connect, викликає метод **sendAddress**().
* void **on\_listWidget\_itemClicked**(QListWidgetItem \*item)- цей метод виконується у випадку натискання на один з найдених Bluetooth-пристроїв, ініціалізує змінну addressToConnect адресою обраного Bluetooth-пристрою.
* void **on\_refreshButton\_clicked**() - цей метод виконується у випадку натискання на кнопку Refresh, при цьому очищається список знайдених пристроїв, та перезапускається процедура пошуку.