МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Інститут інженерії та інформаційних технологій

Кафедра комп'ютерної інженерії та електромеханіки

ДИПЛОМНА БАКАЛАВРСЬКА РОБОТА (ПРОЄКТ)

на тему

ЗАРЯДНИЙ ПРИСТРІЙ ДЛЯ АВТОМОБІЛЬНИХ АКУМУЛЯТОРІВ

Виконав: студент групи БКІ-19

спеціальності

123 Комп'ютерна інженерія

Максим КОШЕЛЮК

Керівник к.т.н., доц.
Дмитро СТАЦЕНКО

Рецензент
(прізвище та ініціали)

Київ 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ
Інститут інженерії та інформаційних технологій
Кафедра комп'ютерної інженерії та електромеханіки
Спеціальність 123 « Комп'ютерна інженерія »
Освітня програма « Комп'ютерні системи та мережі »

ЗАТВЕРДЖУЮ

Завідувач кафедри КІЕМ

_____проф. Злотенко Б.М.

"_____" _____2023 року

З А В Д А Н Н Я
НА ДИПЛОМНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Кошелюк Максим Васильович
(прізвище, ім"я, по батькові)

1. Тема дипломної бакалаврської роботи Зарядний пристрій для автомобільних акумуляторів

Науковий керівник роботи Стаценко Дмитро Володимирович к.т.н., доцент

затверджені наказом вищого навчального закладу від 15.03.2023 № 75-уч.

2. Строк подання студентом роботи 1 червня 2023 року

3. Вихідні дані до дипломної бакалаврської роботи:: технічне завдання, технічна та патентна література

4. Зміст дипломної бакалаврської роботи (перелік питань, які потрібно розробити): Аналіз існуючих зарядних пристроїв для машинних акумуляторів, огляд та аналіз інструментів для створення зарядного пристрою для автомобільних акумуляторів

5. Дата видачі завдання 10.03.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів	Примітка
1	Вступ	06.03.2023	
2	Розділ 1	17.03.2023	
3	Розділ 2	29.03.2023	
4	Розділ 3	12.04.202 3	
5	Висновки	18.05.202 3	
6	Оформлення дипломного проекту (роботи) (чистовий варіант)	26.05.202 3	
7	Здача дипломного проекту (роботи) на кафедру для рецензування (за 14 днів до захисту)	28.05.202 3	
8	Перевірка дипломного проекту (роботи) на наявність ознак плагіату (за 10 днів до захисту)	01.06.202 3	
9	Подання дипломного проекту (роботи) на затвердження завідувачу кафедри (за 7 днів до захисту)	04.06.202 3	

Студент _____ Максим КОШЕЛЮК
(підпис) (прізвище та ініціали)

Науковий керівник роботи _____ Дмитро СТАЦЕНКО
(підпис) (прізвище та ініціали)

Рецензент _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кошелюк М.В. Зарядний пристрій для автомобільних акумуляторів. – Рукопис.

Дипломна робота бакалавра за спеціальністю 123 – Комп'ютерна інженерія освітньої програми «Комп'ютерні системи та мережі». – Київський національний університет технологій та дизайну, Київ, 2023 рік.

Робота присвячена дослідженню і розробленню зарядного пристрою для автомобільних акумуляторів.

В першому розділі в повному обсязі описано та проаналізовані існуючі зарядні пристрої для автомобільних акумуляторів.

В другому розділі проаналізовано інструменти для виконання роботи.

В третьому розділі описана розробка проекту, також представлений виготовлений прилад та інструкція щодо використання.

Пояснювальна записка виконана в текстовому редакторі Microsoft Word,в роботі використані програми sPlan, Sprint Layout, IAR, Proteus.

Ключові слова: зарядний пристрій, трансформаторний зарядний пристрій, регулювання вихідних параметрів, мікроконтролер, дисплей.

ABSTRACT

Koshelyuk M.V. Battery charger for car batteries.– Manuscript.

Bachelor's thesis in specialty 123 - Computer Engineering of the educational program "Computer Systems and Networks". - Kyiv National University of Technologies and Design, Kyiv, year 202X.

The work is devoted to the research and development of a battery charger for car batteries.

The first chapter fully describes and analyzes existing battery chargers for car batteries.

The second chapter analyzes the tools for the job.

The third chapter the development of the project is described, and a manufactured device and instructions for use are also presented.

The explanatory note was made in the Microsoft Word text editor, and the sPlan, Sprint Layout, IAR, Proteus programs were used in the work.

Keywords: battery charger, transformer charger, regulation of output parameters, microcontroller, display.

ЗМІСТ

ВСТУП

Актуальність роботи. Автомобільний акумулятор є деталлю автомобіля, яка забезпечує енергією всі електронні прилади та системи в автомобілях. Довгострокове використання автомобільного акумулятора, без відповідного обслуговування, може призвести до появи проблем пов'язаних із запуском двигуна та відсутності енергії для електричних систем.

Один із способів збереження та підтримки нормальної роботи акумулятора - це регулярне заряджання. Зарядка акумулятора передбачає передачу енергії з зовнішнього джерела до акумулятора. Регулярне заряджання забезпечує тривалий термін служби акумулятора та забезпечує надійність всього електричного обладнання. . Тому, зарядний пристрій – важливий елемент в арсеналі власника автомобіля.

Об'єктом дослідження є процеси керування трансформаторним зарядним пристроєм.

Предметом дослідження є підходи та алгоритми керування системами автомобільного акумулятора.

Метою досліджень є методи аналізу та синтезу розробка зарядного пристрою для автомобільних акумуляторів на мікроконтролері STM8S005K6, який має такі параметри, як напругу зарядки від 0 до 18 В та струм зарядки від 0 до 5 А. Процес зарядки можна контролювати через дисплей, який показує напругу і струм, що були встановлені, поточну напругу і струм на акумуляторі, температуру регулюючого транзистора, а також напругу і струм на акумуляторі.

Методологічною і теоретичною основою дослідження є принципи роботи акумуляторів та їх зарядки, електротехніка та електроніка, а також програмування мікроконтролерів.

Наукова новизна полягає в тому, що дана розробка базується на використанні мікроконтролеру STM8S005K6, що дає можливість контролювати процес зарядки та отримувати розширену інформацію про стан акумулятора.

Практична значимість даної розробки полягає у можливості ефективної

та точної зарядки автомобільних акумуляторів. Зарядний пристрій на мікроконтролері STM8S005K6 може бути корисним для автосервісів та власників автомобілів, оскільки він дозволяє точно контролювати процес зарядки акумулятора та уникнути його перезарядки або перевантаження, що може призвести до його пошкодження. Зарядний пристрій може забезпечити довгий термін експлуатації автомобільного акумулятора і зменшити ймовірність витрат на його заміну. Крім того, розробка може мати практичне застосування в інших галузях, де використовуються акумулятори з подібними параметрами.

Структура та обсяг роботи. Дипломна робота бакалавра складається зі вступу, 3 розділів та висновків по них, загальних висновків, списку використаних джерел та додатків. Основний текст роботи викладений на 69 сторінках, містить 31 рисунок, список джерел з 30 найменувань. Загальний обсяг роботи, враховуючи додаток, складає 83 аркуша.

РОЗДІЛ 1.
АНАЛІЗ ІСНУЮЧИХ ЗАРЯДНИХ ПРИСТРОЇВ ДЛЯ МАШИННИХ АКУМУЛЯТОРІВ

1.1 Різновиди зарядних пристроїв

Простий зарядний

Пристрій даного типу зазвичай не змінює потужність залежно від часу заряджання або рівня заряду акумулятора. Ця простота означає, що такий зарядний пристрій недорогий, водночас він має певні недоліки. Здебільшого, ретельно розроблений простий зарядний пристрій потребує більше часу, щоб повністю зарядити акумулятор, оскільки він налаштований на використання нижчої (тобто безпечнішої) швидкості заряджання. Ці зарядні пристрої також відрізняються тим, що вони можуть подавати сталу напругу або однаковий струм до батареї.

Прості зарядні пристрої, що живляться від мережі змінного струму, зазвичай мають набагато більший пульсуючий струм і пульсуючу напругу, ніж інші типи зарядних пристроїв, оскільки вони недорого влаштовані та виготовлені. Переважно, коли пульсуючий струм перебуває у межах зазначеного виробником акумулятора рівня, пульсуюча напруга також буде в межах зазначеного в настановах рівня. Найбільший пульсуючий струм для типової батареї VRLA 12 В 100 А·г становить 5 ампер. Поки пульсуючий струм не є надмірним (втричі - вчетверо перевищує рекомендований виробником рівень), очікуваний термін служби батареї VRLA з пульсуючим зарядом буде в межах 3% відхилення, від терміну служби батареї із звичним заряджанням постійним струмом [1].

Швидкісний зарядний пристрій

Вони використовують схеми керування для стрімкого заряджання акумуляторів, не пошкоджуючи жодних його елементів. Схема керування може бути вбудована в батарею або у зовнішній зарядний пристрій, чи розподілена між ними. Більшість таких зарядних пристроїв мають охолоджувальний вентилятор, який допомагає підтримувати температуру елементів на безпечному рівні.

Значна кількість швидкісних зарядних пристроїв також можуть працювати як стандартні зарядні пристрої, якщо використовуються зі стандартними елементами,

які не мають особливої схеми керування.

Триступеневий зарядний пристрій

Щоби пришвидшити час заряджання та забезпечити безперервне заряджання, “розумний” зарядний пристрій виконує вбудовану програму для визначення рівня заряду та стан батареї і застосовує 3-етапну схему заряджання. У наведеному нижче описі передбачається використання непроникної свинцево-кислотної тягової батареї за температури 25 °C. Перший етап називається «об'ємним поглинанням»; потужність зарядного пристрою обмежує постійний і високий зарядний струм . Коли напруга на акумуляторі досягає напруги виділення газів (2,22 вольта на елемент), зарядний пристрій перемикається на другий ступінь коли напруга залишається постійною (2,40 вольта на елемент). Струм зменшуватиметься з підтриманням сталої напруги, і водночас струм досягне менше ніж 0,005°C, зарядний пристрій перейде на третій етап, вихідна напруга зарядного пристрою буде підтримуватися постійною на рівні 2,25 вольти на елемент. Оскільки на третьому етапі, зарядний струм дуже невеликий — 0,005°C, під такою напругою акумулятор може підтримуватися на повному заряді, водночас врівноважуватиметься саморозряд [2].

Зарядний пристрій з індукційним живленням

Індуктивні зарядні пристрої для заряджання акумуляторів використовують електромагнітну індукцію. Зарядна станція надсилає електромагнітну енергію через індуктивний зв’язок до електричного пристрою, який накопичує енергію в батареях. Це досягається без потреби металевого з'єднання (контактів) між зарядним пристроєм і акумулятором. Індуктивні зарядні пристрої зазвичай застосовуються в електричних зубних щітках та інших приладах, які використовуються у ванних кімнатах. Оскільки немає відкритих електричних контактів, відсутні ризики ураження електричним струмом. Зараз вони застосовуються для заряджання смартфонів [3].

Розумний зарядний пристрій

Розумний зарядний пристрій може реагувати на стан акумулятора та відповідно змінювати показники його заряджання, тоді як класичні зарядні пристрої подають сталу напругу, крізь незмінний опір. Його не варто плутати з “розумною” батареєю, яка містить комп’ютерний чіп і в цифровому вигляді

передає дані про стан батареї “розумному” зарядному пристрою. Для використання “розумної” батареї потрібен “розумний” зарядний пристрій.

Деякі “розумні” зарядні пристрої також можуть заряджати стандартні батареї, у яких немає внутрішньої електроніки.

Вихідний струм “розумного” зарядного пристрою залежить від стану акумулятора. Інтелектуальний зарядний пристрій може підтримувати напругу, температуру або час заряджання акумулятора, щоби визначити найкращий струм заряджання або припинити його.

Для нікель-кадмієвих і нікель-метал-гідридних батареї напруга батареї повільно зростає під час процесу заряджання, доки батарея не буде цілком заряджена. Після цього напруга знижується, і це вказує “розумному” зарядному пристрою, що батарея повністю заряджена. Такі зарядні пристрої часто позначаються як зарядний пристрій ΔV, “дельта-V” або інколи “дельта-пік:”, і це вказує на те, що вони відстежують зміну напруги. Це може також призводити до того, що навіть інтелектуальний зарядний пристрій не відчує, коли акумулятори вже цілком заряджені, і продовжить заряджання, тож це може загрожувати перезаряджанням акумуляторів. Багато інтелектуальних зарядних пристроїв використовують різні системи вимкнення, аби запобігти перевантаженню батареї [4].

1.2 Огляд автомобільних акумуляторів

Автомобільні акумулятори

Сучасні технології збільшили термін служби автомобільного акумулятора з 2 до 5-7 років. Для вибору зарядного пристрою (ЗП) власник машини повинен точно знати параметри свого автомобільного акумулятора.

Номінальна електрична ємність АКБ – кількість енергії, яку може віддати повністю заряджений акумулятор при розряді в строго обумовлених умовах. Величина ємності вказується у позасистемних одиницях – А·ч (ампер-годину).

Рис1.1 Номінальна електрична ємність різних видів транспорту

Залежно від застосовуваних добавок в матеріал електродів, сучасні стартерні автомобільні акумулятори ділять на наступні типогруппи:

- Sb – сурм'янисті, свинцево-кислотні, з рідким електролітом – 12 В – обслуговуються;
- AGM – з гелієвим електролітом або з всмоктує склоподібним матеріалом –15 В;
- Ca / Ca – кальцієві – 16 В – не обслуговуються;
- Ca +, Ca / Sb – гібридні – згідно з паспортом – малообслуговувані.

Всі автомобільні акумулятори заряджаються за допомогою постійного струму, трьома способами: 1) струмом постійної сили, 2) при постійній напрузі, 3) комбінованим (оптимальним) – спочатку, постійним струмом при змінній напрузі, і в кінці процесу під постійною напругою зі спадаючої силою струму.

1.3 Огляд існуючих зарядних пристроїв для автомобільних акумуляторів

Принцип дії зарядних пристроїв однаковий. Отримуючи живлення ззовні, прилад випрямляє і одночасно знижує значення сили і напруги струму майже до необхідних величин для автомобільних акумуляторів. Відновлюючи акумуляторний заряд батареї, мета не тільки в самому запуску роботи мотора, але і його пуску без ривків, плавному набору оборотів до номінальної швидкості, а також забезпечення повільної рівномірної зупинки його роботи.

Залежно від використовуваного джерела живлення ЗП для автомобільних акумуляторів бувають наступних різновидів:

- від прикурювача – 12 В;
- імпульсні AT I ATX блоки – від 6 до 24 В;
- від електромережі – 220 В;

- автономні (з вбудованим акумулятором);
- на сонячних батареях;
- здвоєні системи для важкої техніки.

Вибір типу – трансформаторний або імпульсний. Класичні трансформаторні ЗП найбільш надійні, але дуже громіздкі. Переважно купувати трансформаторну модель з автоматичною системою управління. Імпульсні або інверторні моделі зручні не тільки повною автоматизацією процесу, а й компактними габаритами.

Вибір моделі – передпускова (Підзарядний) або пуско-зарядні. Підзарядні моделі ЗП заряджають акумулятор довго, але в заощаджуєчому режимі і максимально зберігають властивості батареї. Пуско-зарядні варіанти працюють як в звичайному, так і в прискореному режимі. Вони здатні зарядити повністю використану батарею і запустити автомобільний двигун навіть без наявності робочого акумулятора [5].

Функція експрес-відновлення. Наявність даної функції дозволяє за допомогою методу десульфатації короткочасними імпульсами, повністю очистити пластини акумулятора від сульфату свинцю і відновити заводську ємність батареї.

Заряд в автоматичному режимі. Повна автоматизація процесу зарядки має незаперечні переваги, серед яких важливими є відсутність витрат часу та виключення помилок, що можуть бути спричинені людським фактором. Мікропроцесор точно вимірює параметри напруги і струму, встановлює необхідні параметри зарядки, які залежать від стадії зарядки, та контролює температурні показники [6].

Наявність “зимового” режиму і дисплея. Функція температурної компенсації – інноваційна технологія, яка дозволяє заряджати абсолютно “холодну” АКБ, чим суттєво економить час. Переваги наявності рідкокристалічного дисплея з нічним підсвічуванням не вимагають пояснень.

Рівень класу захисту. Один з найважливіших елементів це автоматична блокада включення, яка запобігає неправильному підключенню клем до пристрою і небезпечним ситуаціям. Якщо клеми неправильно підключені, зарядний пристрій може блокувати своє включення для запобігання можливого ураження електричним струмом або інших небезпечних ситуацій [7].

Друга важлива функція - автоматичне відключення пристрою при досягненні повної зарядки. Це забезпечує оптимальний стан заряду батареї, запобігає її перевантаження і витраті енергії, що дозволяє збільшити термін експлуатації пристрою.

Захист від короткого замикання, перепадів напруги і стрибків в електромережі також є важливим елементом зарядного пристрою. Ці захисні функції забезпечують безпеку при зарядці і захищають пристрій від можливих пошкоджень, що можуть виникнути внаслідок негативних впливів на напругу і струм в електромережі [8].

1.4 Огляд предметного середовища

На сьогоднішній день одним з найбільш потужних і гнучких засобів розробки електронних схем є мікроконтролери. При роботі з мікроконтролерами необхідно враховувати такий фактор - коли розробляється система на основі мікроконтролера, то створюються не тільки апаратні засоби, що реалізуються відповідним підключенням мікроконтролера до зовнішніх пристроїв. Окрім цього розробник повинен забезпечити виконання багатьох системних функцій, які в традиційних мікропроцесорних системах забезпечуються за допомогою операційної системи та спеціальних периферійних мікросхем.

Відмінності в архітектурі процесорів можуть істотно позначитися на їхній продуктивності при виконанні різних завдань.

STM8S005K6 є низькопотужним мікроконтролером з ядром STM8S, призначеним для малих програм.. Він може працювати з напругою від 2,95В до 5,5 В і містить 32-кБ пам'яті програми, 2-кБ оперативної пам'яті і вбудований периферійний блок. Для розробки проектів на базі STM8S005K6 можна використовувати спеціальні інтегровані середовища розробки, такі як STM8S-Discovery або STM8S-ezrin. Доступна детальна документація та SDK для STM8S005K6 на офіційному сайті STM.

При розробці проектів, що використовують STM8S005K6, першою задачею є завантаження функціональних вимог до пристрою. Цей мікроконтролер може використовуватися для управління реле, сигналізації, контролю рівня води тощо.

Для розробки проекту можна використовувати мову програмування C або використовувати STM8CubeMX, який генерує код на C [9].

Якщо потрібно розробити систему контролю рівня води, можна включити прилад STM8S-Discovery і датчик Ultrasonic Range Sensor HC-SR04 для вимірювання висоти води. Потім проект можна програмувати та відлагоджувати за допомогою STM8S-ezrin.

Після розробки та тестування програмного коду проект слід перевірити на можливі помилки та оптимізувати для покращення продуктивності та мінімізації споживання електроенергії. Найбільш популярні бібліотеки програмного забезпечення, такі як STM8S Standard Peripheral Libraries, часто використовуються для розробки з STM8S005K6, оскільки вони допомагають зробити розробку швидше та менш проблемною [10].

Визначення послідовності елементів, їх параметрів та сполучень один з одним - це основний етап створення електричної схеми. Її можна зобразити від руки на папері або створити за допомогою спеціальної програми для комп'ютера.

Для створення електричної схеми необхідно спочатку застосувати тип системи, яку буде моделювати схему - це може бути, наприклад, схема з'єднання двох пристроїв, схема електромережі чи схема керування.

Далі необхідно застосувати всі елементи, які будуть використані в схемі, їх тип, параметри та місце розташування в схемі. Зазвичай в електричних схемах використовуються такі елементи, як резистори, конденсатори, індуктивності, діоди, транзистори та інші електронні компоненти.

Після цього необхідно відобразити всі з'єднання між елементами та встановити напрямок потоку електронів через схему. Це допоможе вирішити правильну роботу системи та зменшити кількість помилок [11].

При розробці процесу електричної схеми необхідно не тільки отримати параметри конкретних елементів схеми, але також отримати всі фактори, які впливають на роботу системи - це можуть бути електромагнітні поля, температурні умови та інші параметри. Усі елементи схеми повинні мати унікальні мітки та позначення, які допоможуть легко ідентифікувати кожен елемент під час роботи зі схемою.

Отже, створення електричної схеми - це складний процес, який потребує розуміння функціональності кожного елемента схеми, їх правильного сполучення з одним та дотриманням відповідних норм і стандартів.

ВИСНОВОК ДО РОЗДІЛУ 1

У цьому розділі було розглянуто та проаналізовані види автомобільних акумуляторів, їх властивості та способи зарядки. Було встановлено, що для подовження терміну експлуатації автомобільного акумулятора, що зазвичай становить 5-7 років, кожному водію рекомендовано мати пристрій для зарядки. Було розглянуто різноманітні види зарядних пристроїв для автомобілів, а також їх особливості.

Розглянуті основні технології розробки зарядних приладів, принципи будування електричних схем. Мікроконтролер STM8S005K6, є гнучким та потужним інструментом для розробки електронних схем. При розробці системи на основі мікроконтролера необхідно забезпечити виконання багатьох системних функцій, які в традиційних мікропроцесорних системах забезпечуються за допомогою операційної системи та спеціальних периферійних мікросхем. STM8S005K6 може використовуватися для різних цілей, таких як управління реле, контроль рівня води тощо. Програмування відбувається за допомогою мови C або STM8CubeMX, яка генерує код на C. Для відлагодження проекту можна використовувати STM8S-ezrin або інші доступні розробки інтегрованого середовища. При розробці використовуються бібліотеки програмного забезпечення, такі як STM8S Standard Peripheral Libraries, щоб зробити розробку швидше і з меншим ризиком помилок.

Отже, в даному розділі розглянуто важливість зарядки автомобільних акумуляторів та їх види, а також особлива увага приділена особливостям зарядних пристроїв для автомобілів. Варто зазначити, що існують різні технології розробки зарядних пристроїв, включаючи використання мікроконтролерів, керування реле чи контроль рівня води. Окрім того, описано переваги програмування STM8S005K6 мовою C або за допомогою STM8CubeMX, а також використання бібліотек програмного забезпечення, що допомагають прискорити та зробити розробку більш продуктивною.

РОЗДІЛ 2.
ОГЛЯД ТА АНАЛІЗ ІНСТРУМЕНТІВ ДЛЯ СТВОРЕННЯ ЗАРЯДНОГО ПРИСТРОЮ ДЛЯ АВТОМОБІЛЬНИХ АКУМУЛЯТОРІВ

2.1 Етапи створення приладу зарядки автомобільного акумулятора

Враховуючи вимоги до основного та додаткового функціоналу, вказані в Розділі 1, створений прилад буде живитись від мережі змінного струму з напругою 220В, базуватиметься на мікроконтролері STM8S005K6 і керується за допомогою кнопок управління.

Прилад повинен давати можливість користувачу встановлювати параметри за допомогою кнопок, відображати поточний стан приладу на дисплеї.

Для досягнення мети та уникнення помилок у зарядного пристрою необхідно було чітко розуміти наступні дії та не пропустити жодного етапу. Для цього було запропоновано розробити план розробки, який включав в себе пункти. Було проаналізовано багато відкритих ресурсів, які вже містять список етапів, але було важливо скласти власний список.

1) Затвердження вимог проекту ;
2) Дослідження ринку;
3) Вибір мікроконтролера ;
4) Створення електричної схеми ;
5) Розробка ;
6) Тестування та виправлення помилок ;

(1) Початковий етап процесу розробки приладу передбачає затвердження вимог проекту. Тобто прилад має бути зручним і зрозумілим у використанні для звичайного користувача при цьому виконувати свій функціонал.

(2) На етапі дослідження ринку було зрозуміло, які прилади є, їх плюси та мінуси. Цей етап дав змогу розробнику зрозуміти загальний вигляд приладу та основний функціонал.

(3) Мікроконтролер - це інтегральна мікросхема, яка містить у собі ядро, пам'ять та ряд периферійних пристроїв, додаткових для керування електронними процесорами. Вибір мікроконтролера залежить від конкретної задачі, яку потрібно

розв'язати.

Основні критерії при виборі мікроконтролера:

- Максимальна тактова частота і продуктивність, які необхідні для розв'язання завдань.
- Кількість і тип периферійних пристроїв, що підтримуються мікроконтролером.
- Обсяг вбудованої пам'яті для зберігання програмного та даних.
- Наявність програматора та навчальної документації, яка допоможе ознайомитися з роботою мікроконтролера.

На сьогоднішній день найбільш поширеними серед мікроконтролерів є виробники: Atmel, Microchip, NXP, STM, TI та інші. Їх відмінності полягають у характеристиках інтерфейсів, структурі та типі програмованих.

Після відбору мікроконтролера та розробки програми, його необхідно було програмувати. Для програмування мікроконтролера було використано спеціальний програмний засіб, IAR Embedded Workbench.

4)Створення електричної схеми - це процес розробки просторового плану електричних з'єднань та елементів, які необхідні для створення інженерної системи. Основні компоненти електричної схеми включають джерела електроенергії, провідники, роз'єднувальні пристрої, інструменти контролю та захисту, а також споживачів електроенергії.

Після врахування всіх вимог та деяких параметрів, розроблюється електрична схема, яка буде максимально ефективною та простою у використанні. Основними етапами створення електричної схеми є визначення джерел електроенергії, розробка елементів на схемах, роз’єднувальних пристроїв та інших елементів, планування розташування споживачів електроенергії, розробка систем контролю та захисту. Усі параметри та елементи електричної схеми потрібно виконати з точки зору їх взаємодії та безпеки в роботі. Після створення електричної схеми відбувається перевірка на збіг із вимогами інженерного проекту та коригування.

(5Після визначення типу пристрою, мікроконтролера та складання схеми, необхідно було написати прошивку для мікроконтролера та перейти до тестування.

(6) Тестування та виправлення помилок - це важливий етап проекту, який виконувався в програмі Proteus. Ця програма дозволяє віртуально перевірити

результати та в разі виявлення помилок їх виправити. Після успішної перевірки було зроблено розводку плати за допомогою програми Sprinter Layout, і складено всі елементи на плату відповідно до схеми.

2.2 Вибір та аналіз доступних технологій та програмного забезпечення

Для вирішення поставленої мети дипломної роботи обрано розробити трансформаторний зарядний пристрій на базі мікроконтролера STM8S005K6.

IAR Embedded Workbench програма для написання прошивки для мікроконтролера. Обрано мову програмування для написання внутрішньої програми для мікроконтролера С, це найпоширеніша мова для написання програм для STM мікроконтролерів . Для складання електричної схеми використана програма sPlan 7.0. Для розробки друкованих плат використано програму Sprint Layout. А для відлагодження та первиного перегляду результату зібраного приладу використано програму Proteus, яка дає змогу віртуально змоделювати прилад склавши його електричну схему віртуально.

2.3 Мікроконтролери

Мікроконтролер – це спеціалізований мікроелектронний програмований прилад, що призначений для використання у керуючих пристроях, системах передачі даних та системах керування технологічними процесами.

Мікроконтролери використовують у побутовій техніці, медичних приладах, системах керування ліфтами, телефонах, раціях та інших засобах зв’язку, електронних музичних інструментах та автомагнітолах, комп’ютерній периферії (клавіатурах, джойстиках, принтерах, тощо), світлофорах, автоматичних воротах та шлагбаумах, інтерактивних дитячих іграшках, автомобілях, локомотивах та літаках, роботах та промислових верстатах[12].

Рис. 2.1 Сфери використання мікроконтролерів.

Мікроконтролери також широко застосовуються в автомобільній електроніці. Наприклад, автомобіль «Peugeot 206» має на борту 27 мікроконтролерів, а в автомобілях високого класу, як наприклад «BMW» сьомої серії, використовується понад 60 мікроконтролерів. Вони керують вприском палива, жорсткістю адаптивної підвіски, світлотехнікою, двигунами двірників, склопідіймачів та дзеркал заднього виду, тощо (рис.2.2).

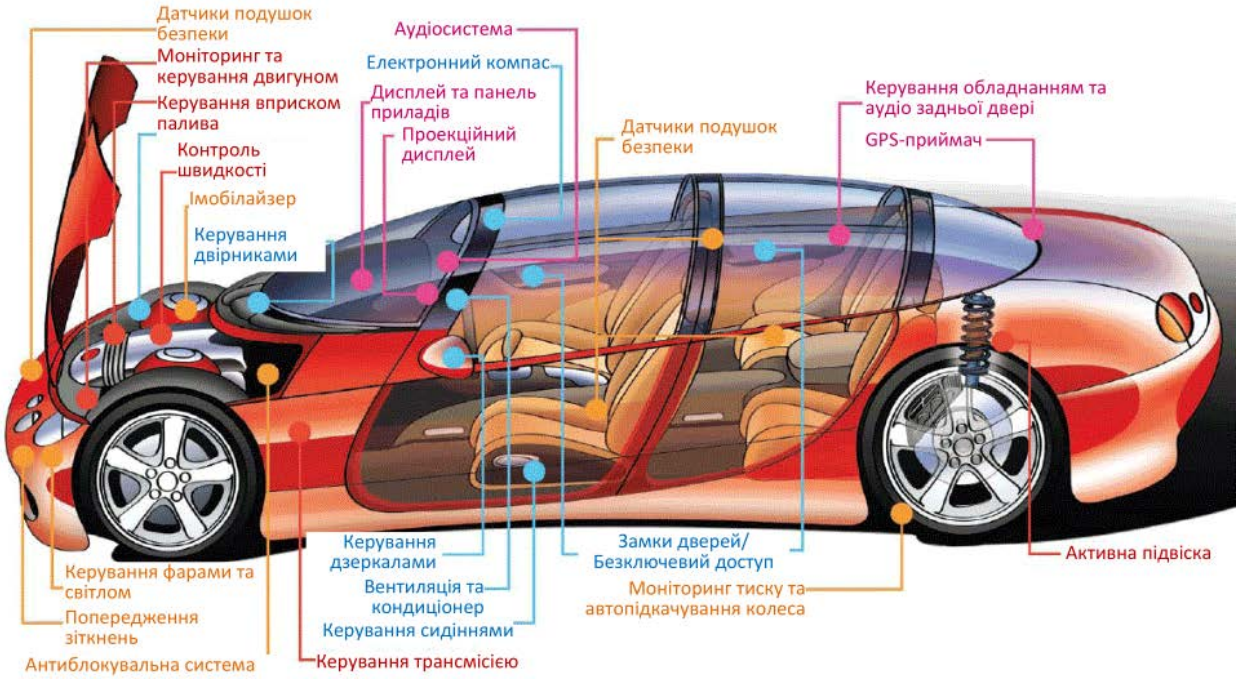

Рис. 2.2 Використання мікроконтролерів в автомобільний електроніці

Мікроконтролер, на відміну від мікропроцесора, зазвичай має порівняно невелику розрядність (8 – 16 бітів) та багатий набір команд маніпулювання окремими бітами. Бітові команди дають змогу керувати дискретним обладнанням (підняти/опустити шлагбаум, увімкнути/вимкнути лампу, нагрівач, запустити/зупинити двигун, відкрити/закрити клапан, тощо). Наявність можливості оперувати окремими бітами, вводити та виводити дискретні сигнали називають «бітовим процесором».

Ще одна з основних відмінностей мікроконтролера від мікропроцесора полягає у тому, що у складі мікросхеми контролера є усі елементи для побудови системи керування. Всередині мікроконтролера є пам'ять даних (оперативна пам'ять), пам'ять програм (постійна пам'ять), генератор тактових імпульсів, таймери, лічильники, паралельні та послідовні порти. Система мінімальної конфігурації на основі мікроконтролера може складатися з блока живлення, безпосередньо мікросхеми контролера та кількох пасивних елементів (резисторів, конденсаторів та кварцового резонатора).

Типова архітектура мікроконтролера складається з системи синхронізації та керування (1), арифметико-логічного пристрою (2), регістрів загального призначення (3), пам’яті даних (4) та пам’яті програм (5), портів (6), функціональних пристроїв (таймерів, лічильників, широтно-імпульсних модуляторів, інтерфейсів) та регістрів їх настроювання (7)(рис.2.3).

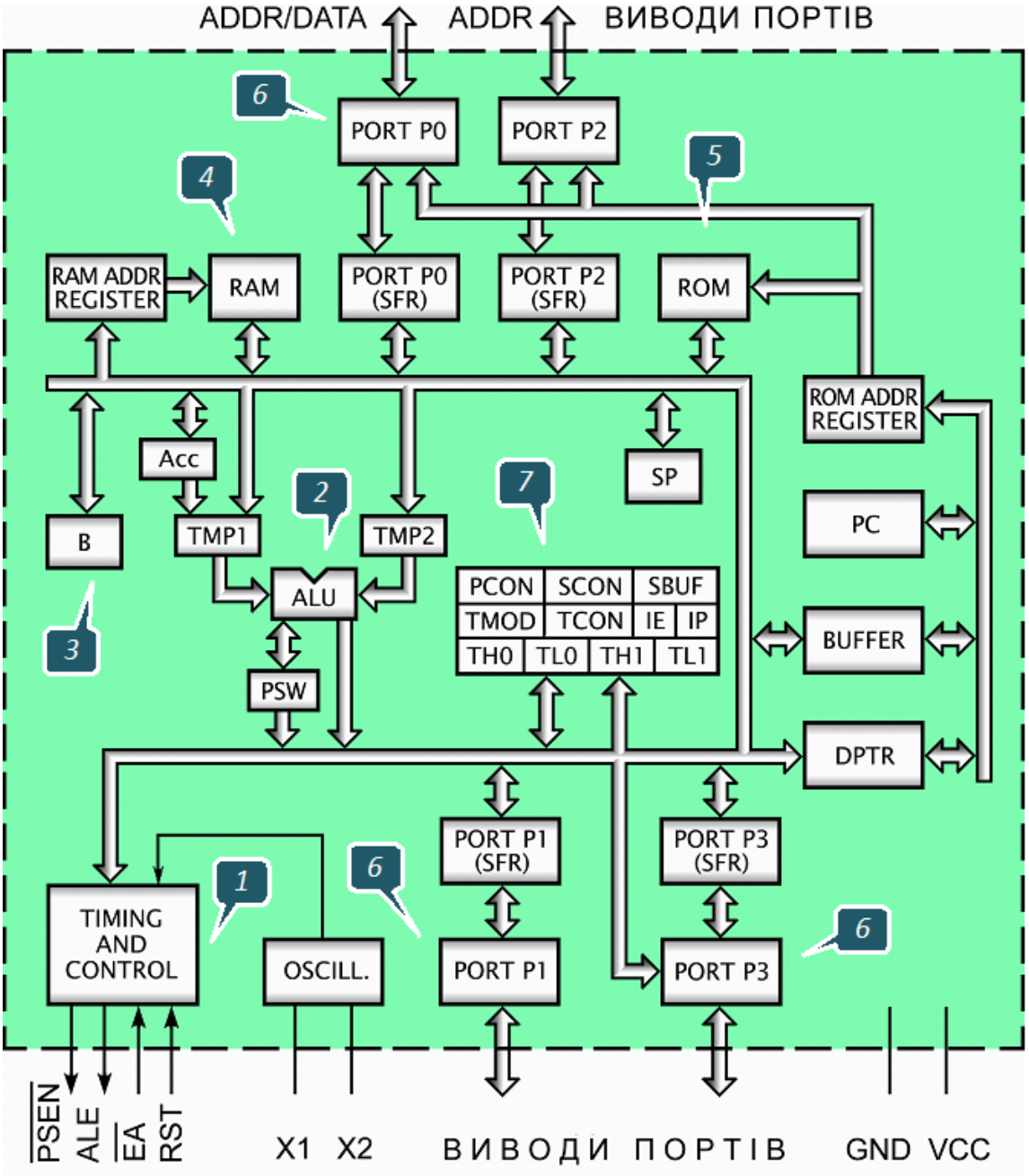

Рис. 2.3 Архітектура типового мікроконтролера.

Програми для мікроконтролерів створюють у спеціальних інтегрованих інструментальних середовищах (англ.: Integrated Development Environment, IDE) мовами Асемблера (машинних команд) або С.

Мікроконтролери стали невід'ємною частиною сучасного технологічного середовища.

2.4 Основні визначення та класифікація мікроконтролерів

МПС (Мікропроцесорні системи) на основі МП (мікропроцесор) частіше за все використовуються в якості вбудованих систем для вирішення завдань управління об’єктом. Важливою особливістю даного застосування є робота в реальному часі, тобто забезпечення реакції на зовнішні події протягом певного часового інтервалу. Такі вбудовані вузькоспеціалізовані керуючі МПС, виконані у вигляді окремих мікросхем, які працюють у реальному масштабі часу, називають мікроконтролерами.

Мікроконтролер (МК) – це функціонально закінчена МПС, виготовлена на одній НВІС (надвеликій інтегральній схемі). Мікроконтролер містить у собі: процесор, ОЗП, ПЗП, порти вводу-виводу для підключення зовнішніх пристроїв, модулі вводу аналогового сигналу АЦП, таймери, контролери переривання, контролери різних інтерфейсів і т.д. Найпростіший МК представляє собою ВІС площею не більше 1 см2 і всього з вісьмома виходами.

Контролери, як правило, створюються для рішення якоїсь окремої задачі або групи близьких задач. Вони зазвичай не мають можливостей підключення додаткових вузлів і пристроїв, наприклад, великої пам’яті, засобів вводу/виводу. Їх системна шина часто недоступна користувачеві. Структура контролера проста і оптимізована під максимальну швидкодію. У більшості випадків програми, що виконуються, зберігаються в постійній пам’яті і не змінюються. Контролери можуть бути виготовлені у різних варіантах конструкції, але часто зустрічаються у вигляді одноплатних пристроїв.

Основні переваги застосування систем з МК:

1. Значно підвищується гнучкість;
2. Істотно знижується вартість;
3. Знижується час розробки та модифікації;
4. Підвищується надійність системи за рахунок скорочення кількості корпусів і з’єднань.

Розрізняють наступні типи мікроконтролерів:

1) **Периферійні** (інтерфейсні) МК призначені для реалізації найпростіших МП систем управління. Вони мають малу продуктивність і малі габаритні розміри. Зокрема можуть використовуватися периферійними пристроями ЕОМ (клавіатура, миша і т.п.). До них відносяться: AVR-Atmel, PIC – Micro Chip, VPS – 42 (Intel).
2) **Універсальні 8-розрядні** МК призначені для реалізації МП систем малої і середньої продуктивності. Мають просту систему команд і велику номенклатуру вбудованих пристроїв. Основні типи: MCS – 51 (Intel), Motorola HC05 – HC012 та 96н..
3) **Універсальні 16-розрядні** МК призначені для реалізації систем реального часу середньої продуктивності. Структура і система команд адаптовані на швидку реакцію за зовнішніми подіями. Найбільше використання мають в системах управління електродвигунами. До типових 16-розрядних МК відносяться: MCS96/196/296 (Intel), C161-C167 (Siemens, Infineon), HC16 Motorola та 96н.
4) **Спеціалізовані 32-розрядні** МК реалізують високопродуктивну архітектуру і призначені для систем телефонії, передачі інформації, телебачення і інших, що вимагають високошвидкісної обробки інформації.
5) **Цифрові сигнальні процесори** (DSP – Digital Signal Processor) призначені для складної математичної обробки вимірюваних сигналів у режимі реального часу. Широко використовуються в телефонії та зв'язку.

Основні відмінності DSP: підвищена розрядність оброблюваних слів (16,32,64 біта) і висока швидкість у форматі з плаваючою точкою (16 flops). Виробники: Texas Instruments (TMS 320 та 97н.), Analog Device (ADSP 2181 і 97н.) .

У сучасних МК використовуються такі типи системи команд процесорів:

- **RISC** – (Reduce Instruction Set Commands) архітектура зі скороченим набором команд.
- **CISC** – (Complex Instruction Set Commands) традиційна архітектура з розширеним набором команд.
- **ARM** – (Advanced RISC – machine) вдосконалена RISC архітектура.

Головне завдання RISC архітектури забезпечення найвищої продуктивності процесора. Її характерними ознаками є:

- Мала кількість команд процесора (кілька десятків);
- Кожна команда виконується за мінімальний час (1-2 машинних цикла) ;
- Максимально можлива кількість регістрів загального призначення процесора (кілька тисяч);
- Збільшена розрядність процесора (12,14,16 біт).

Сучасна RISC архітектура включає, як правило, тільки останні 3 пункти, тому що за рахунок щільності компонування ВІС стало можливим реалізувати велику кількість команд. У сучасних 32-розрядних МК використовують ARM архітектуру (розширена RISC архітектура з суперскороченням команд THUMB).

2.4.1 Мікроконтролери сімейства AVR

AVR – найпоширеніша виробнича лінія серед інших мікроконтролерів корпорації Atmel. Atmel представила перший 8-розрядний мікроконтролер в 1993 році і з тих пір безперервно удосконалює технологію. Прогрес технології з новим ядром полягає у зниженні питомого енергоспоживання (мА/МГц), розширення діапазону напруги живлення (до 1,8 В) для продовження ресурсів батарейних систем, збільшенні швидкодії до 16 млн. операцій за секунду (конвеєризація), використанням часових емуляторів і відладчиків, реалізації функції самопрограмування, удосконалення і розширення кількості периферійних модулів, вбудуванням спеціалізованих пристроїв (радіочастотний передавач, USBконтролер, драйвер рідинно-кристалевого індикатора РКІ, програмована логіка, контролер DVD, пристрої захисту даних і ін.) [13].

AVR- мікроконтролери є досить простими у використанні та програмуванні, що дозволяє розробникам легко створювати ефективні проекти. Крім того, AVR-мікроконтролери мають високу продуктивність та енергоефективність, що дозволяє їх використовувати у широкому спектрі застосувань.

Доступність великої кількості інструментальних засобів для проектування, таких як компілятори, середовища розробки та програматори. Ці інструменти постачаються як безпосередньо від компанії-виробника, так і від відкритих

спільнот. Це дозволяє розробникам швидко та легко розпочинати роботу з AVR-мікроконтролерами та створювати високоякісні проекти.

Загалом, успіх AVR-мікроконтролерів можна пояснити поєднанням кількох факторів, включаючи їх простоту та продуктивність, доступність інструментів для проектування та широкий спектр застосувань.

Іншою особливістю AVR-мікроконтролерів, яка сприяла їх популяризації, є використання RISC-архітектури, що характеризуються потужним набором інструкцій (118-133), кожна з яких має довжину в одне слово (16 біт) і більшість яких виконуються за один машинний цикл. Це означає, що при рівній частоті тактового генератора вони забезпечують продуктивність в 12 (6) разів більше продуктивності попередніх мікроконтролерів на основі CISC-архітектури (наприклад, MCS51). З іншого боку, у рамках одного застосування із заданою швидкодією, AVR-мікроконтролер може тактуватися в 12 (6) разів меншою тактовою частотою, забезпечуючи однакову швидкодію, але при цьому споживаючи набагато меншу потужність. Таким чином, AVR мікроконтролери представляють ширші можливості з оптимізації відношення продуктивності до енергоспоживання, що особливо важливо при розробці додатків з батарейним живленням.

Мікроконтролери забезпечують продуктивність до 16 млн. оп. у секунду і підтримують FLASH-пам’ять (ППЗП) програм різної ємкості: 1…256 кбайт. Велика швидкодія обумовлена дворівневим конвеєром при виконанні команд. Під час першого машинного циклу відбувається вибірка команди з пам'яті програм і її декодування, а під час другого циклу ця команда виконується, також паралельно відбувається вибірка і декодування другої команди і так далі.

AVR-архитектура (рис. 2.4) оптимізована під язик високого рівня Сі, а більшість представників сімейства megaAVR містять 8-канальний 10 розрядний АЦП, а також сумісний з IEEE 1149.1 інтерфейс JTAG або debugWIRE для вбудованої відладки.

Крім того, всі мікроконтролери megaAVR з флэш-памятью ємкістю 16 кбайт і більш можуть програмуватися через інтерфейс JTAG.

Арифметико-логічний пристрій (АЛП), що виконує всі обчислення,

безпосередньо підключений до 32 робочих регістрів, об'єднаних у реєстровий файл. Завдяки цьому АЛП виконує одну операцію (читання вмісту регістрів, виконання операції і запису результату назад у реєстровий файл) за один машинний цикл.

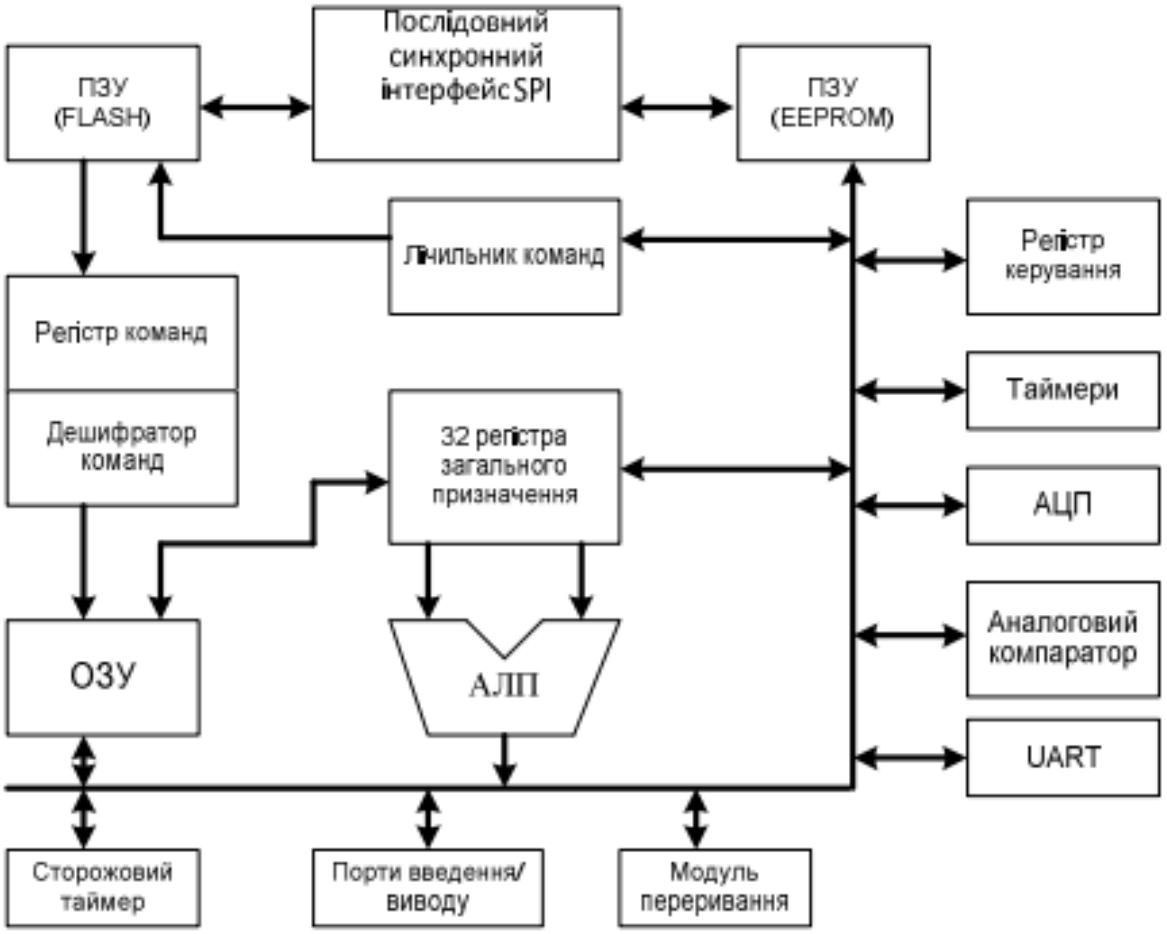

Рис. 2.4. Архітектура ядра мікроконтролерів AVR

У мікроконтролерах AVR практично всі команди (за винятком команд, у яких одним з операндів є 16нрозрядна адреса) займають одну комірку пам'яті програм. Організація пам'яті мікроконтролерів AVR сімейства Classic виконана по Гарвардській архітектурі, у якій розділені не тільки адресні простори пам'яті програм і пам'яті даних, але також і шини доступу до них. Причому пам'ять даних складається із трьох областей: реєстрова пам'ять, статичне ОЗП і пам'ять на основі EEPROM. У зв'язку з тим, що реєстрова пам'ять перебуває в адресному просторі ОЗУ, про ці дві області пам'яті звичайно говорять як про одну. Кожна з областей (ОЗП і EEPROM) розташована у своєму адресному просторі.

В рамках базової RISC-архитектури AVR-мікроконтролери розділяються на три сімейства:

- Classic AVR – базова лінія мікроконтролерів;
- Mega AVR – мікроконтролери для складних застосувань, що вимагають великого об'єму пам'яті програм і даних;
- Tiny AVR – (маленький) недорогі мікроконтролери з 8 вивідного виконання.

Мікроконтролери AVR фірми “Atmel” є 8-ми розрядними мікроконтролерами що мають пам'ять програм (FLASH), пам'ять даних EEPROM (електрично стирається, перепрограмовується статичний запам’ятовуючий пристрій ЕСППЗП) і різноманітні периферійні пристрої, склад яких змінюється від моделі до моделі. Мікроконтролери виготовляються за малоспоживаючою К-МОП-технологією, яка у поєднанні з вдосконаленою RISC-архітектурою дозволяє досягти якнайкращого співвідношення показників швидкодія/енергоспоживання.

Характерні особливості:

- Продуктивність, що наближається до 1 MIPS/МГц;
- Вдосконалена AVR RISC архітектура;
- Роздільні шини пам'яті команд і даних (гарвардська архітектура), 32 регістри загального призначення;
- Система двобайтових команд складається з 118(133) базових операцій;
- Вбудовані аналоговий компаратор, сторожовий таймер, порти SPI і UART, таймери/лічильники;
- Flash ПЗП програм, з можливістю внутрісистемного перепрограмування і завантаження через SPI послідовний канал, 1000 циклів стирання/запис EEPROM даних, з можливістю внутрісистемного завантаження через SPI послідовний канал, 100000 циклів стирання/запис. Блокування режиму програмування; Повністю статичні прилади працюють при тактовій частоті від 0 Гц до 20 Мгц;
- Діапазон напруги живлення від – 1,8 В до 6,0 В.

Додаткові функції:

- Скидання по включенню живлення (установка у нульовий стан);
- Вбудований годинник реального часу з окремим кварцевим резонатором;
- Три режими енергозбереження:
- Активний режим 6.4 мА;
- Режим холостого ходу пасивний (idle) 1.9 мА;
- Режим низького споживання стоповий <1 мкА;
- Зв'язок з навколишнім середовищем через 32 програмованих лінії ВЕДЕННЯ/ВИВЕДЕННЯ, конфігуруванні у чотирьох портах (A, B, C, D).

2.4.2 Мікроконтролери сімейства PIC

Основним призначенням мікроконтролерів PIC, як випливає з абревіатури PIC (Peripheral Interface Controller), є виконання інтерфейсних функцій. Цим пояснюються особливості їх архітектури:

– RISC-система команд, що характеризується малим набором одноадресних інструкцій (33, 35 або 55), кожна з яких має довжину в одне слово (12, 14 або 16 біт) і більшість виконується за один машинний цикл. У системі команд відсутні складні арифметичні команди (множення, ділення), гранично скорочений набір умовних переходів;

– висока швидкість виконання команд: при тактовій частоті 20 Мгц час машинного циклу складає 200 нс (швидкодія дорівнює 5 MIPS –млн. операцій/сек);

– наявність потужних драйверів (до 24 мА) на лініях портів введення/виведення;

– низька споживана потужність;

– орієнтація на цінову нішу гранично низької вартості, що визначає використання дешевих корпусів з малою кількістю виводів (8, 14, 18, 28), відмова від зовнішніх шин адреси і даних (окрім PIC17C4X), використання спрощеного механізму переривань і апаратного (програмно недоступного) стека.

Мікроконтролери сімейства PIC16CXXX, виконані за технологією HCMOS (high performance complementary metal oxide semiconductor) являють собою 8-

розрядні мікроконтролери на основі RISC-процесора, виконані за гарвардською архітектурою. Мають вмонтований ПЗП команд об'ємом від 0,5 до 4 Кслів (розрядність слова команд дорівнює 12 - 14 біт).

Пам'ять даних PIC-контролерів організована у виді реєстрового файлу об'ємом 32 - 128 байт, у якому від 7 до 16 регістрів відведено для управління системою та обміну даними з зовнішніми пристроями.

Одним з основних переваг цих пристроїв є дуже широкий діапазон напруг живлення (2 - 6 В). Струм споживання на частоті 32768 Гц складає менше 15 мкА, на частоті 4 МГц - 1 - 2 мА, на частоті 20 МГц 5 - 7 мА і у режимі мікроспоживання (режим SLEEP) - 1 - 2 мкА. Випускаються модифікації для роботи в трьох температурних діапазонах: від 0 до +70°С, від -40 до +85°С та від -40 до +125°С.

Кожен з контролерів містить універсальні (від 1 до 3) сторожові таймери, а також надійно збудовану систему ініціалізації при увімкненні живлення. Частота внутрішнього тактового генератора задається або кварцовим резонатором, або RC-ланкою у діапазоні 0 - 25 Мгц. PICконтролери мають від 12 до 33 ліній цифрового вводу-виводу, причому кожна з них може бути незалежно настроєна на ввід або вивід [14].

Обмеженість ресурсів мікроконтролерів PIC робить проблематичним їх програмування на мовах високого рівня. Особливості архітектури мікроконтролерів PIC виправдані надзвичайно низькою ціною, тому ці вироби (особливо сімейства PIC16) вельми популярні. У даний час їх використовують навіть замість логічних ІС середнього ступеня інтеграції. Але реалізувати всі переваги цих мікроконтролерів можна тільки за наявності засобів програмування і налаштування, адекватних за ціною і функціональними можливостями вирішуваним завданням [15].

Порівнюючи розглянуті мікроконтролери, можна відзначити наступні особливості:

- У мікроконтролерів сімейства MCS-51 коротка команда

виконується за 12 тактів генератора (один машинний цикл);

- У PIC контролера коротка команда виконується за 8

тактів генератора (два машинний цикл), тобто у конвеєрному

потоці одна команда – це один машинний цикл - 4 такти;

- У AVR контролера коротка команда виконується за 1 такт генератора (один машинний цикл).

У даний час при розробці нових мікропроцесорних систем найчастіше вибирають шлях використання мікроконтролерів (приблизно у 80% випадків). При цьому мікроконтролери застосовуються або самостійно, з мінімальною додатковою апаратурою, або у складі складніших контролерів з розвиненими засобами введення/виведення. За найбільш серйозних виробників на світовому ринку Flashмікроконтролерів вважаються Motorola, Renesas, Microchip, ST Micro, Philips і Atmel.

2.5 Програмування мікроконтролерів. Вибір середовища розробки.

Зазвичай програмування мікроконтролерів виконують на мові асемблера або с хоча також існують компілятори для інших мов для відлагодження програм використовують вбудовані інтерпретатори basic і fortran а також програмні симулятори які імітують роботу мікроконтролера для розробки програм на персональному комп’ютері можна використовувати спеціальні програми які імітують роботу мікроконтролера або внутрішньо схемні емулятори які є електронними пристроями що можна під’єднати замість мікроконтролера до вбудованого пристрою що розробляється також можна використовувати інтерфейс jtag для програмування та відлагодження програм на мікроконтролерах [16].

2.5.1 Arduino IDE

Arduino IDE є безкоштовним та відкритим середовищем розробки для програмування мікроконтролерів. Воно підтримує широкий спектр мікроконтролерів, зокрема, серію Arduino та багато інших платформ, таких як ESP8266 та ESP32.Одним з головних переваг Arduino IDE є його простий та

зрозумілий інтерфейс, що дозволяє швидко розпочати розробку проектів. У середовищі є можливість роботи з більшістю основних мов програмування, таких як C, C++, та Java.

Для розробки програм використовується мова Wiring, що базується на мові C++, але має спрощений синтаксис для розробки програм для мікроконтролерів. Окрім того, в середовищі є можливість використовувати бібліотеки, що значно спрощує розробку проектів.

Arduino IDE також має можливість симуляції та налагодження коду, що дозволяє швидко виявляти помилки та відлагоджувати програми. Крім того, у середовищі є можливість розробки графічного інтерфейсу за допомогою бібліотек, що дозволяє створювати проекти зі складними інтерфейсами користувача.

Однак, варто зазначити, що Arduino IDE має обмежені можливості налаштування та дебаггінгу, що може бути проблемою для деяких проектів. Також, середовище підтримує в основному мікроконтролери від Arduino та ESP, тому, якщо ви працюєте з іншими платформами, можливо, вам доведеться шукати інші середовища розробки.

2.5.2 Atmel Studio

Atmel Studio є інтегрованою середою розробки для програмування мікроконтролерів від компанії Atmel. Вона має безкоштовний та відкритий код та підтримує широкий спектр мікроконтролерів, включаючи AVR та SAM дивізії компанії Atmel.

Atmel Studio має багато інструментів та можливостей для розробки програм для мікроконтролерів. Його інтерфейс є інтуїтивно зрозумілим та простим у використанні. Для розробки програм використовується мова програмування C, C++ та ASM.

Одним з основних переваг Atmel Studio є можливість використання багатьох плагінів та інструментів розробки, що робить його одним з найбільш гнучких середовищ розробки для мікроконтролерів. Крім того, Atmel Studio має інструменти для роботи з віртуальними пристроями, що дозволяє симулювати роботу програми без використання фізичного мікроконтролера.

Також, Atmel Studio має вбудований дебаггер, що дозволяє легко відлагоджувати програми та шукати помилки. У середовищі є можливість налаштування дебаггера та додавання власних плагінів для розширення його можливостей.

Однак, Atmel Studio має дещо складніший інтерфейс порівняно з іншими середовищами розробки, що може потребувати додаткового часу для оволодіння ним. Крім того, середовище не підтримує мікроконтролери від інших виробників, окрім компанії Atmel.

2.5.3 Keil µVision

Keil µVision є інтегрованою середою розробки для програмування мікроконтролерів, що була розроблена компанією ARM. Вона підтримує багато мікроконтролерів, включаючи ARM Cortex-M, ARM7 та ARM9, і є одним з найбільш популярних середовищ розробки для програмування мікроконтролерів.

Keil µVision має дуже зручний та легкий у використанні інтерфейс, що дозволяє швидко створювати та редагувати програми. Інтегроване середовище також містить декілька інструментів для розробки програм, включаючи редактор коду, дебагер, інструменти для відлагодження, симулятор та інші.

Keil µVision підтримує мови програмування C та C++. Воно також має вбудовані бібліотеки для підтримки різних функцій, таких як робота з периферійними пристроями, робота з мережевими протоколами та інші.

Ще однією перевагою Keil µVision є підтримка різних типів мікроконтролерів, що робить його дуже гнучким та високопродуктивним. Крім того, Keil µVision має велику базу користувачів, що забезпечує доступ до великої кількості документації та підтримки.

2.5.4 IAR Embedded Workbench

IAR Embedded Workbench є інтегрованою середовищем розробки для програмування мікроконтролерів від шведської компанії IAR Systems. Це потужне інструментальне забезпечення, яке дозволяє розробникам створювати програми для

різних мікроконтролерів, включаючи ARM, Renesas і Texas Instruments.

IAR Embedded Workbench забезпечує інтуїтивно зрозумілий інтерфейс користувача, що дозволяє легко створювати, налагоджувати та відлагоджувати програми для мікроконтролерів. Ця інтегрована середовище розробки має багато корисних інструментів, таких як редактор коду з підсвічуванням синтаксису, функції автодоповнення та аналізу статичного коду, що робить процес програмування більш ефективним та швидким.

Однією з переваг IAR Embedded Workbench є те, що вона має вбудовані оптимізаційні та налагоджувальні інструменти, які дозволяють збільшити продуктивність та ефективність програм для мікроконтролерів. Наприклад, вона має вбудований оптимізатор, який може зменшити розмір програми та збільшити її швидкість роботи.

Іншою перевагою IAR Embedded Workbench є те, що вона підтримує багато мікроконтролерів від різних виробників. Вона має готові конфігурації для більш ніж 8000 різних мікроконтролерів, що дозволяє розробникам швидко створювати програми для нових пристроїв без необхідності вручну налаштовувати драйвери та інші параметри.

Однак, IAR Embedded Workbench є комерційним продуктом, тому вона не є безкоштовною.

2.5.5 MPLAB X IDE

MPLAB X IDE - це інтегрована середовище розробки від Microchip Technology для програмування мікроконтролерів PIC, AVR і SAM. Це потужне інструментальне забезпечення дозволяє розробникам легко створювати, відлагоджувати та тестувати програми для різних мікроконтролерів.

MPLAB X IDE забезпечує користувачам інтуїтивно зрозумілий інтерфейс користувача, який дозволяє легко створювати нові проекти, додавати та видаляти файли та виконувати інші операції з програмним кодом. Серед інших корисних функцій MPLAB X IDE можна виділити можливість виконання аналізу статичного коду, редагування коду з підсвічуванням синтаксису, автоматичне завершення коду та багато іншого.

MPLAB X IDE також забезпечує розробникам можливість відлагоджувати та тестувати програми за допомогою вбудованого візуального середовища, що дозволяє здійснювати налагодження по кроку, контролювати виконання коду та перевіряти стан регістрів та пам'яті мікроконтролера.

Окрім цього, MPLAB X IDE має широкий спектр інструментів та плагінів, що дозволяють розробникам легко і швидко інтегрувати їхні програми з різними пристроями та додатками. Наприклад, це може бути інструменти для профілювання продуктивності, симуляції схем, плагіни для роботи з бібліотеками та інші інструменти.

Узагалі, MPLAB X IDE є дуже потужним та високоефективним інструментом для розробки програм для мікроконтролерів, який забезпечує користувачам широкі можливості для розробки та тесту.

2.6 sPlan

Редактор схем sPlan розробляється з початку 2000-х німецькою фірмою АВАКОМ. Програма підтримується і зараз, на даний момент остання версія sPlan 7.0. Наприкінці 2014 року компанія заявила, що веде розробки нової версії sPlan 8.0. З документацією по всіх функціях програми можна ознайомитись, використовуючи вбудовану довідку (англійською мовою).

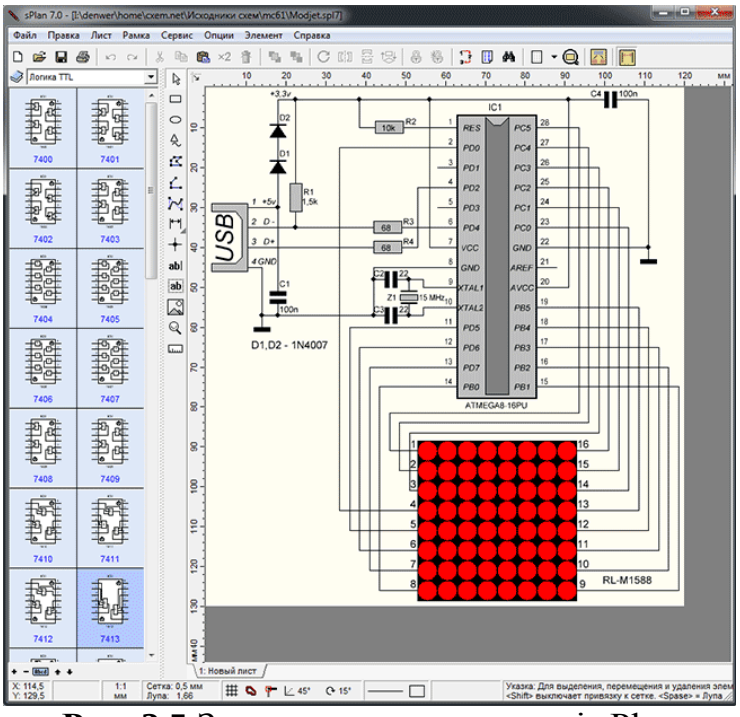

Рис. 2.5 Зразок схеми в програмі sPlan

sPlan дуже зручний у використанні. Додані компоненти просто «перетягуються» з панелі, що знаходиться зліва, праворуч від якої знаходиться панель інструментів для малювання ліній і різних геометричних форм, додавання написів, вставки растрових зображень і т.д. Нумерація компонентам може надаватися як автоматично, так і вручну. Режим можна вибрати правою кнопкою миші через властивості компонента (Properties), у тому ж таки вікні редагуються додаткові атрибути, у тому числі текстове поле (в ньому можна написати, наприклад, номінал даної деталі і т.д.). У комплект програми включено велику кількість готових бібліотек електронних компонентів, можливе створення та збереження власних шаблонів компонентів. Редактор може працювати з кількома сторінками (у вигляді вкладок), є функції експорту документа до графічного файлу та друку з попереднім переглядом. В останній версії зручно реалізовано масштабування за допомогою коліщатка миші, можливий друк великих аркушів на звичайному принтері, розрахованому на аркуш формату А4.

Програма платна, у безкоштовній версії відключені функції збереження, експорту та друку файлів. Пропонується також окрема безкоштовна програма для перегляду та друкування файлів (sPlan Viewer).

Програма працює у середовищі Windows NT, 2000, XP, Vista, 7, 8, 10 x32/64, до конфігурації комп'ютера якихось специфічних вимог немає.

2.7 Proteus

Proteus Design Suite – пакет програм для автоматизованого проектування (САПР) електронних схем. Розробка компанії Labcenter Electronics (Велика Британія).

Пакет є системою моделювання, що базується на основі моделей електронних компонентів, прийнятих в PSpice. Відмінною рисою пакету PROTEUS VSM є можливість моделювання роботи програмованих пристроїв: мікроконтролерів, мікропроцесорів, DSP та ін. Причому в Proteus повністю реалізована концепція наскрізного проектування, коли інженер змінює щось у логіці роботи схемотехніки і програмний пакет тут же «підхоплює» дані зміни в системі трасування. Бібліотека компонентів містить довідкові дані.

Додатково до пакету PROTEUS VSM входить система проектування друкованих плат.

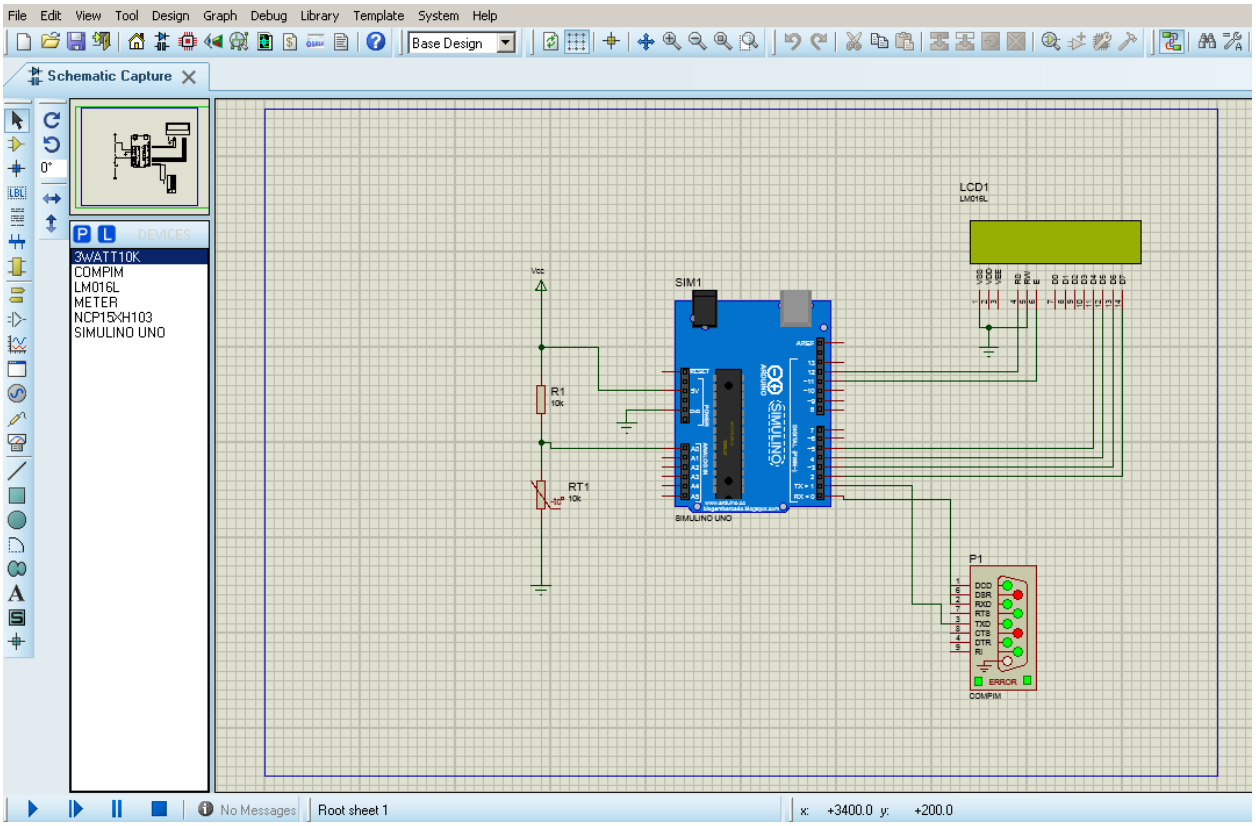

Рис. 2.6 Зразок схеми в програмі Proteus

Пакет Proteus складається з двох частин, двох підпрограм: ISIS – програма синтезу та моделювання безпосередньо електронних схем та ARES – програма розробки друкованих плат. Разом із програмою встановлюється набір демонстраційних проектів для ознайомлення.

Також до складу восьмої версії входить середовище розробки VSM Studio, що дозволяє швидко написати програму для мікроконтролера, що використовується в проекті, та скомпілювати.

Пакет є комерційним. Безкоштовна ознайомча версія характеризується повною функціональністю, але немає можливості збереження файлів.

Примітною особливістю є те, що ARES можна побачити 3D-модель друкованої плати, що дозволяє розробнику оцінити свій пристрій ще на стадії розробки.

Система підтримує підключення нових елементів (SPICE) та підключення різних компіляторів (PICOLO, ARM-подібні, AVR і далі).

2.8 Мова програмування мікроконтролера

Програмування для мікроконтролерів, як і програмування для універсальних комп'ютерів пройшло великий шлях розвитку від програмування в машинних кодах до застосування сучасних інтегрованих систем написання програм, налагодження та програмування мікроконтролерів. В даний час вихідний текст програми пишеться на одній з мов програмування.

Процес перетворення операторів вихідної мови програмування в машинні коди мікропроцесора називається трансляцією вихідного тексту. В даний час ручна трансляція програм практично не використовується. Трансляція здійснюється спеціальними программамі- трансляторами.

Існує два великі класи програм-трансляторів: компілятори и інтерпретатори. При використанні компіляторів весь вихідний текст програми перетворюється в машинні коди, і саме ці коди записуються в пам'ять мікроконтролера. При використанні інтерпретатора в пам'ять мікроконтролера записується вихідний текст програми, а телеканал передається при зчитуванні чергового оператора. Природно, що швидкодія інтерпретаторів набагато нижче в порівнянні з компіляторами, т. К. При використанні оператора в циклі він транслюється багаторазово. Однак при програмуванні на мові високого рівня обсяг коду, який потрібно зберігати у внутрішній пам'яті може бути значно менше в порівнянні з виконуваним кодом. Ще однією перевагою застосування інтерпретаторів є легка переносимість програм з одного процесора на інший [17].

2.8.1 Мова програмування С

Програмування мікроконтролерів на С є одним з найпоширеніших способів розробки вбудованих систем. С є мовою програмування високого рівня, яка дозволяє програмістам писати код, який може бути ефективним та оптимізованим для вбудованих систем, де ресурси обмежені.

У програмуванні мікроконтролерів на С використовуються спеціальні компілятори, які перетворюють код на мові С в бінарний код, який може бути виконаний на мікроконтролері. Для розробки програм на С для мікроконтролерів

можуть використовуватися різні інструменти, такі як текстові редактори, інтегровані середовища розробки (IDE) та інші.

У програмуванні мікроконтролерів на C зазвичай використовуються різні бібліотеки, які надають програмістам доступ до різних функцій, таких як взаємодія з портами введення-виведення (I/O), таймерами, перериваннями та іншими функціями, які можуть бути використані для розробки вбудованих систем [18].

ВИСНОВОК ДО РОЗДІЛУ 2

Важливим компонентом електроніки є зарядні пристрої, які забезпечують енергією батареї різних електронних пристроїв. Для забезпечення ефективної та стабільної роботи електронної техніки, обрано трансформаторний зарядний пристрій.

Трансформаторний зарядний пристрій має кілька переваг. По-перше, він забезпечує стабільний струм та напругу, що дозволяє заряджати акумулятори різних типів та ємностей. По-друге, трансформаторний зарядний пристрій є більш надійним та безпечним виконанням, оскільки не має відкритих контактів і не виробляє небезпечних електричних іскр. Також, трансформаторний зарядний пристрій забезпечує більш ефективний перехід від мережевої напруги до напруги заряду, що забезпечує швидке та ефективне заряджання батарей.

Отже, обрано трансформаторний зарядний пристрій, оскільки він забезпечує стабільну та безпечну роботу заряджання батарей, що позитивно впливає на продуктивність та якість життя.

STM8S005C6/K6 є мікроконтролером з фірмовою архітектурою STM8 від компанії STMicroelectronics. Цей мікроконтролер є досить популярним в області вбудованих систем завдяки своїм характеристикам та вартості.

Використання даного мікроконтролера забезпечує наступні переваги: продуктивність, надійність, зниження вартості системи та короткі цикли розробки.

Основні причини, чому обрано STM8S005C6/K6 для зарядного пристрою, наступні:

Низька вартість: STM8S005C6/K6 є досить дешевим мікроконтролером, що дозволяє знизити загальну вартість проекту.

Низька споживана потужність: цей мікроконтролер має дуже низьку споживану потужність, що дозволяє використовувати його в батарейних пристроях.

Має вбудовані апаратні модулі для роботи з PWM (модуляція ширини імпульсу), ADC(аналого-цифровий перетворювач) та таймерами, що дозволяє значно спростити розробку зарядних пристроїв.

Програмне забезпечення для складання електричних схем обрав sPlan 7, яке використовується інженерами та розробниками в галузі електротехніки та

електроніки. Основною причиною, чому обрано sPlan 7 для складання електричних схем, була:

Легкість використання та інтуїтивно зрозумілий інтерфейс: програма має зрозумілий та зручний інтерфейс, що дозволяє швидко та легко створювати та редагувати електричні схеми.

Програмне забезпечення для легкого і швидкого проектування друкованих плат обрано Sprint-Layout. Основною перевагою Sprint-Layout є інтуїтивно зрозумілий інтерфейс, що включає лише необхідні інструменти для підготовки друкованих плат .

І для програмування мікроконтролерів було обрано програмне забеспечення IAR Embedded Workbench. Програма буде написана на мові С.

РОЗДІЛ 3.
РОЗРОБКА ЗАРЯДНОГО ПРИСТРОЮ

3.1 Розробка принципової електричної схеми

Розробка принципової електричної схеми створення схеми з'єднання між електричними компонентами та пристроями в електричному колі. Основна мета розробки електричної схеми полягає в забезпеченні правильної та безпечної роботи в усій електричній системі.

Основні кроки при розробці електричної схеми:

1. Визначення типу та призначення електричної системи.
2. Створення набору схематичних елементів.
3. З'єднання елементів між собою.
4. Оформлення кінцевого варіанту електричної схеми.

Важливим етапом при розробці електричної схеми є відповідність необхідним стандартам та нормам безпеки. Розробка електричної схеми включає в себе також перевірку електричної відповідності всіх елементів вимогам до електричної схеми, проведення електричних розрахунків, що забезпечує безпечну роботу системи та її ефективність [19].

3.1.1 Моделювання основних керуючих вузлів ЗП в Proteus

В основній моделі рис3.1 ЗП є три основних керуючих вузли :

1) Основний регулюючий – на цей вузол сходяться сигнали з вузлів, а саме: керування вихідною напругою, керування вихідним струмом, обмеження вихідного струму на рівні 5,5 А. Рис. 3.2
2) Керування вихідною напругою – підтримує задану напругу контролером на виході ЗП. Рис. 3.3
3) Керування вихідним струмом - підтримує заданий струм контролером на виході ЗП. Рис. 3.4

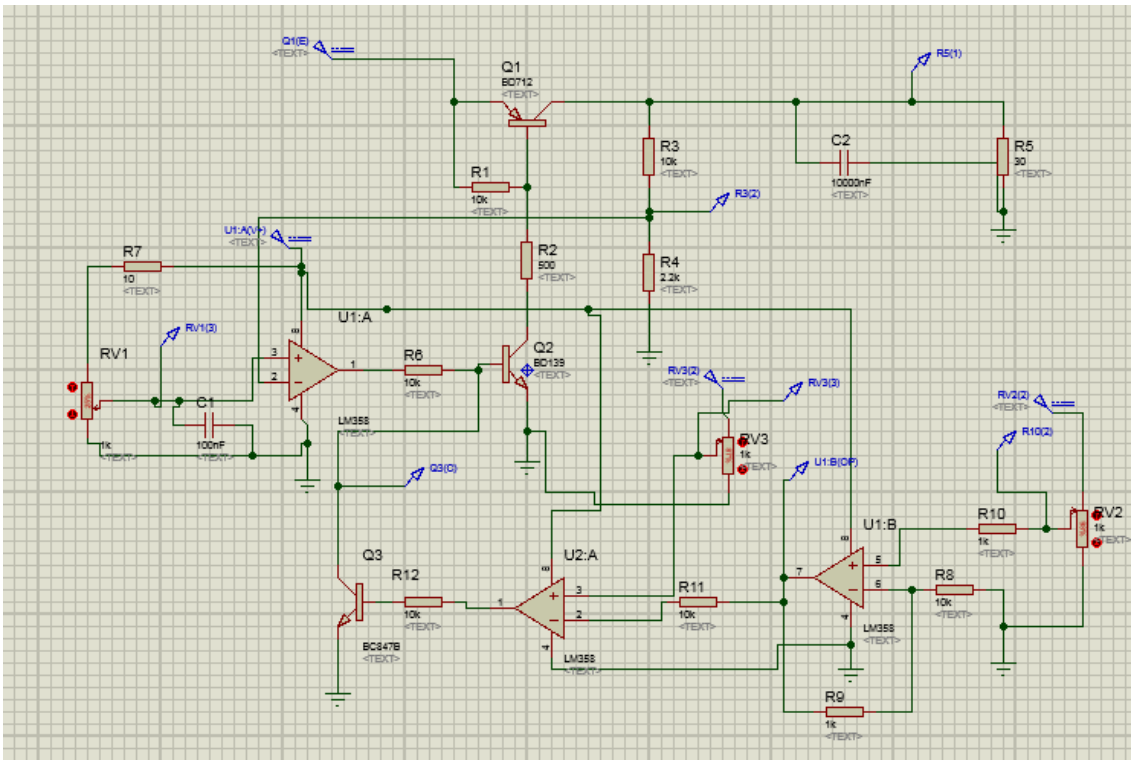

Рис. 3.1 Модель керування вихідним струмом та напругою в програмі Proteus

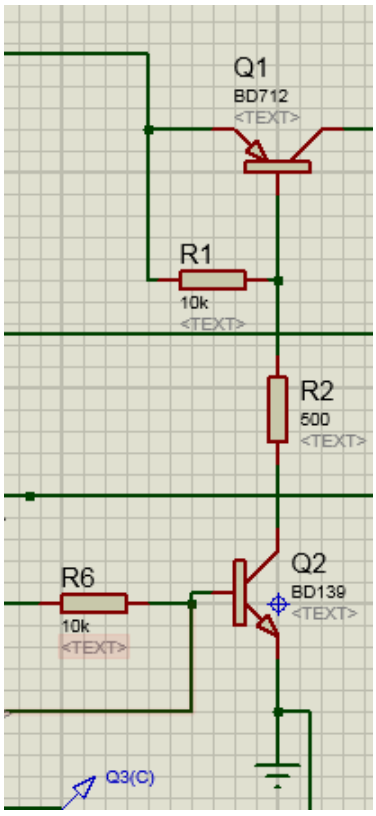

Рис. 3.2 Основний регулюючий вузол в моделі Proteus

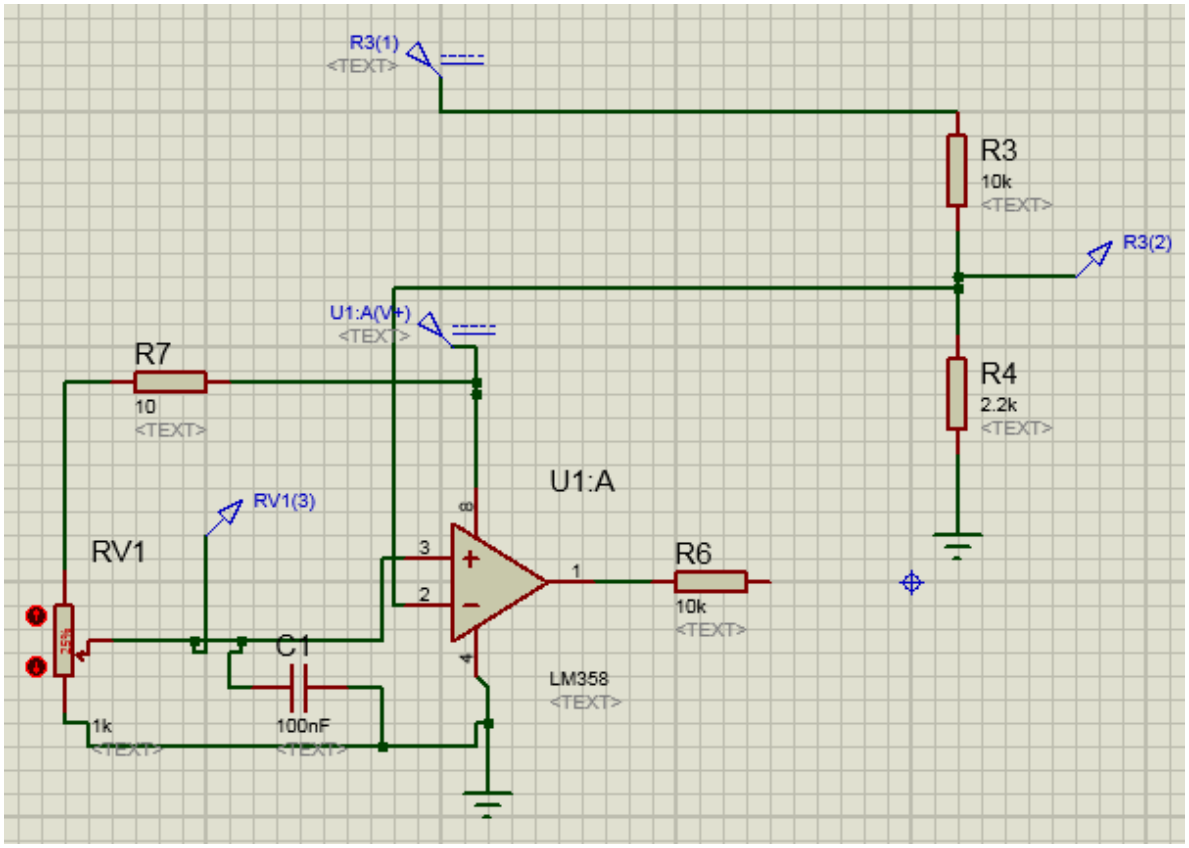

Рис. 3.3 Модель керування вихідною напругою

в програмі Proteus

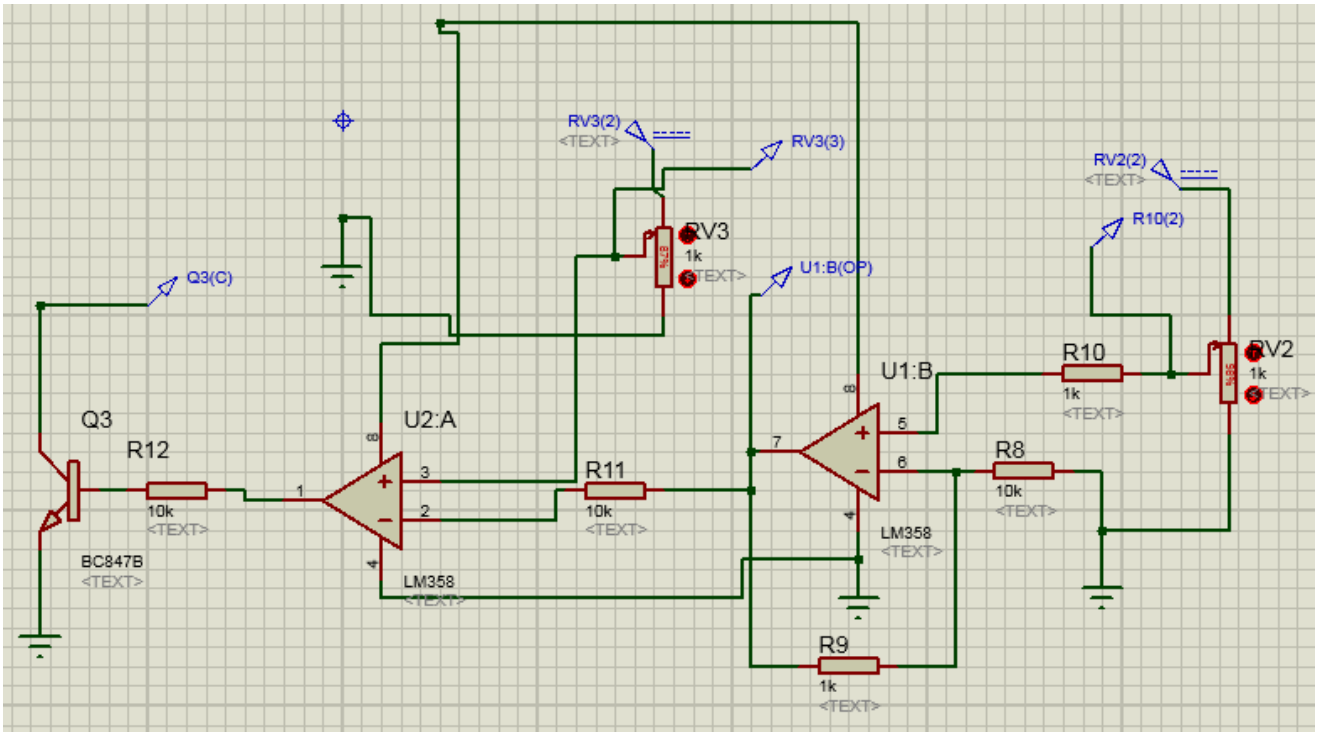

Рис. 3.4 Модель керування вихідним струмом

в програмі Proteus

В вузлах використовуються операційні підсилювачі для порівняння напруги і струму на виході ЗП з еталонними які встановлює мікроконтролер. Залежно від потенціалів на входах операційних підсилювачів міняється потенціал на їх виходах. Сигнали надходять з мікроконтролера та датчика струму або напруги, в залежності від вузла [20].

3.1.2 Електрична принципова схема ЗП

Електрична схема побудована на базі мікроконтролера STM8S005K6, який керує вузлами ЗП. Схема містить такі вузли: трансформаторний випрямляч змінної напруги, дисплейний модуль індикації, клавіатура керування, стабілізатори живлення модулів регулювання та мікроконтролера, датчик струму на ефекті Холла, регулюючий вузол по напрузі і струму з температурним захистом [21].

Джерелом живлення ЗП є трансформаторний мостовий виправляч. Ця напруга використовується для зарядки АКБ та живлення всіх вузлів ЗП. Напруги +5В та +12В формуються інтегральними стабілізаторами LM7805 та LM7812. В якості дисплейного модуля використано символьний LCD 1602 на базі контролера HD44780, який підключено по чотирьохпровідному інтерфейсу. Підсвічування дисплею виконано на LM317 по схемі стабілізатора струму. Для встановлення вихідних параметрів ЗП використовується триклавішна клавіатура з задіянням двох портів МК [22].

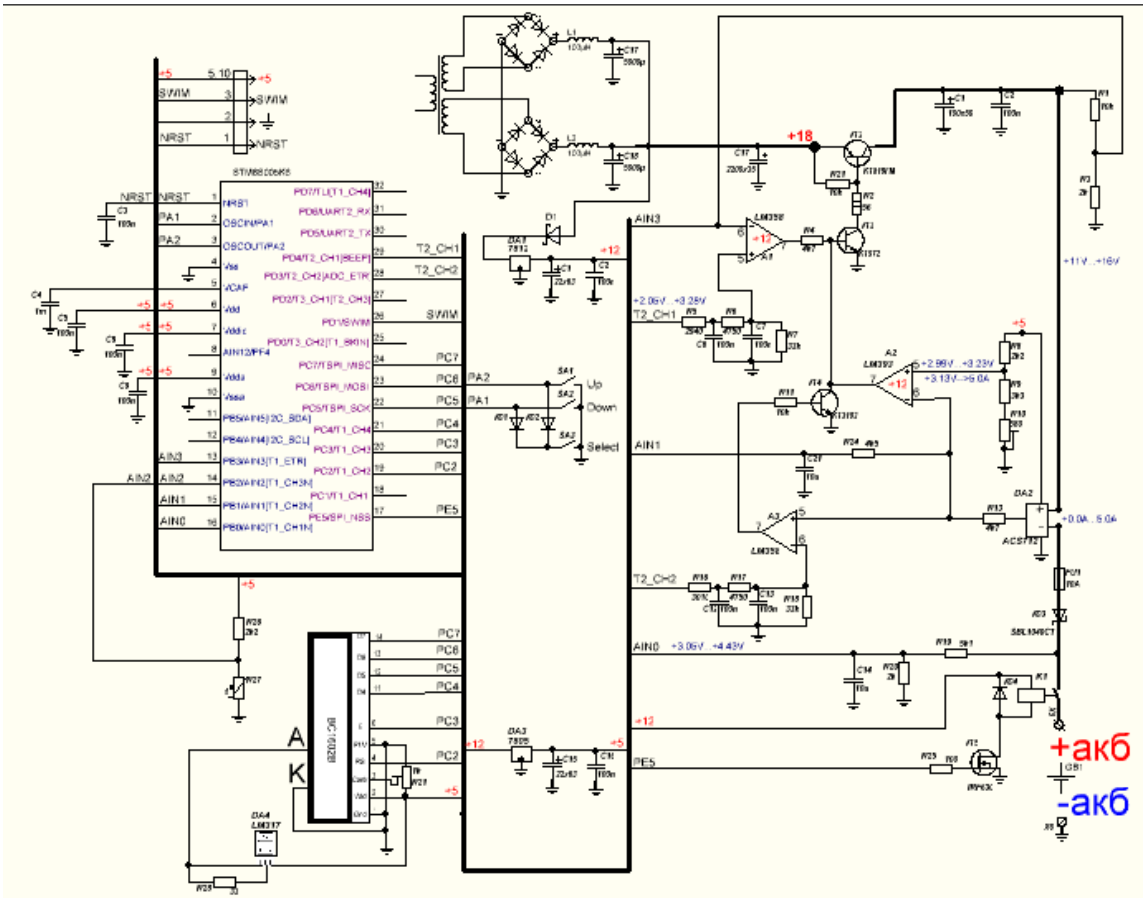

Рис. 3.5 Схема електрична принципова ЗП

Силовий регулюючий вузол побудований на PNP транзисторі КТ818ГМ, який встановлено на радіатор з активним охолодженням. Також використовується терморезистор для моніторингу температури і аварійного відключення акумулятора у разі перевищення температури. Операційні підсилювачі LM358 встановлюють вихідні параметрами за рахунок подачі на один із входів поточного значення параметра, а на другий вхід керуючого широтно-імпульсного сигналу від МК через RC-фільтр. Для узгодження вихідних рівнів сигналів використовуються резистивні дільники [23].

Підключення акумулятора у зворотній полярності не приведе до негативних наслідків для ЗП за рахунок встановленого потужного діода Шоткі в колі заряду, але це може привести до виходу з ладу самого акумулятора . Додатковий захист від нештатних ситуацій забезпечує плавкий запобіжник на 10А та тригерний захист на компараторі LM393 [24].

3.2 Мікроконтролер STM8S005K6

8-розрядні мікроконтролери STM8S005C6/K6 пропонують 32 Кбайт програмної флеш-пам'яті, а також 128 байт даних EEPROM. У довідковому посібнику сімейства мікроконтролерів STM8S (RM0016) вони називаються пристроями середньої щільності.

Усі пристрої лінійки STM8S005C6/K6 забезпечують наступні переваги: продуктивність, надійність, зниження вартості системи та короткі цикли розробки.

Ефективність і надійність пристрою забезпечуються справжніми даними EEPROM, що підтримують до 100 000 циклів запису/стирання, вдосконаленим ядром і периферійними пристроями, створеними за найсучаснішою технологією з тактовою частотою 16 МГц, надійними входами/виходами, незалежними сторожовими таймерами з окремим годинником. джерело та система безпеки годинника [25].

Вартість системи знижується завдяки високому рівню системної інтеграції з внутрішніми генераторами тактового сигналу, сторожовим таймером і скиданням.

Загальна архітектура сімейних продуктів із сумісною розпіновкою, картою пам’яті та модульними периферійними пристроями забезпечує масштабованість додатків і скорочує цикли розробки.

Особливості:

- Ядро
    - Максимальна частота цп : 16 МГц
    - Розширене ядро STM8 з гарвардською архітектурою та 3-ступеневим конвеєром
    - Розширений набір інструкцій

- Память
    - Flash/EEPROM середньої щільності
        - Пам'ять програм: 32 Кбайт флеш-пам'яті; збереження даних 20 років при 55 °C після 100 циклів
        - Пам'ять даних: 128 байт справжніх даних EEPROM; витривалість до 100 тисяч циклів запису/стирання
    - Оперативна пам'ять: 2 Кбайт

- Тактування, скидання та керування живленням
  - Робоча напруга від 2,95 В до 5,5 В
  - Гнучке керування тактуванням, 4 джерела головного тактування
    - Малопотужний кристалічний резонаторний генератор
    - Вхід зовнішнього тактового генератора
    - Внутрішній, настроюваний користувачем RC 16 МГц
    - Внутрішній малопотужний 128 кГц RC
  - Моніторинг тактування
  - Управління живленням
    - Режими низького енергоспоживання (очікування, активна зупинка, зупинка)
    - Включення тактування окремо для кожного периферійного модуля.
    - Постійно активний, низький рівень споживання живлення та живлення в ресеті
- Керування перериваннями
  - Вкладений контролер переривань з 32 перериваннями
  - До 23 зовнішніх переривань на 6 векторах
- Таймери
  - 2x 16-бітні таймери загального призначення з 2+3 каналами CAPCOM (IC, OC або PWM)
  - Розширений таймер керування: 16-бітний, 4 канали CAPCOM, 3 комплемитарні виходи, вставка мертвого часу та гнучка синхронізація
  - 8-розрядний базовий таймер з 8-розрядним попереднім дільником
  - Таймер автоматичного пробудження
  - Віконний та незалежний сторожовий таймери
- Комунікаційні інтерфейси
  - UART з тактовим виходом для синхронної роботи, SmartCard, IrDA, LIN
  - Інтерфейс SPI до 8 Мбіт/с
  - Інтерфейс I^2C до 400 Кбіт/с
- Аналого-цифровий перетворювач (АЦП)

- 10-розрядний АЦП, ± 1 LSB з до 7 мультиплексованих каналів, режим сканування та аналоговий сторожовий таймер

- введення/виведення
  - До 25 входів/виходів на 32-контактному корпусі
  - Надзвичайно надійна конструкція вводу/виводу, стійка до інжекції струму
- Підтримка для розробки
  - Вбудований однопровідний інтерфейсний модуль (SWIM) для швидкого програмування на кристалі та ненав'язливого налагодження

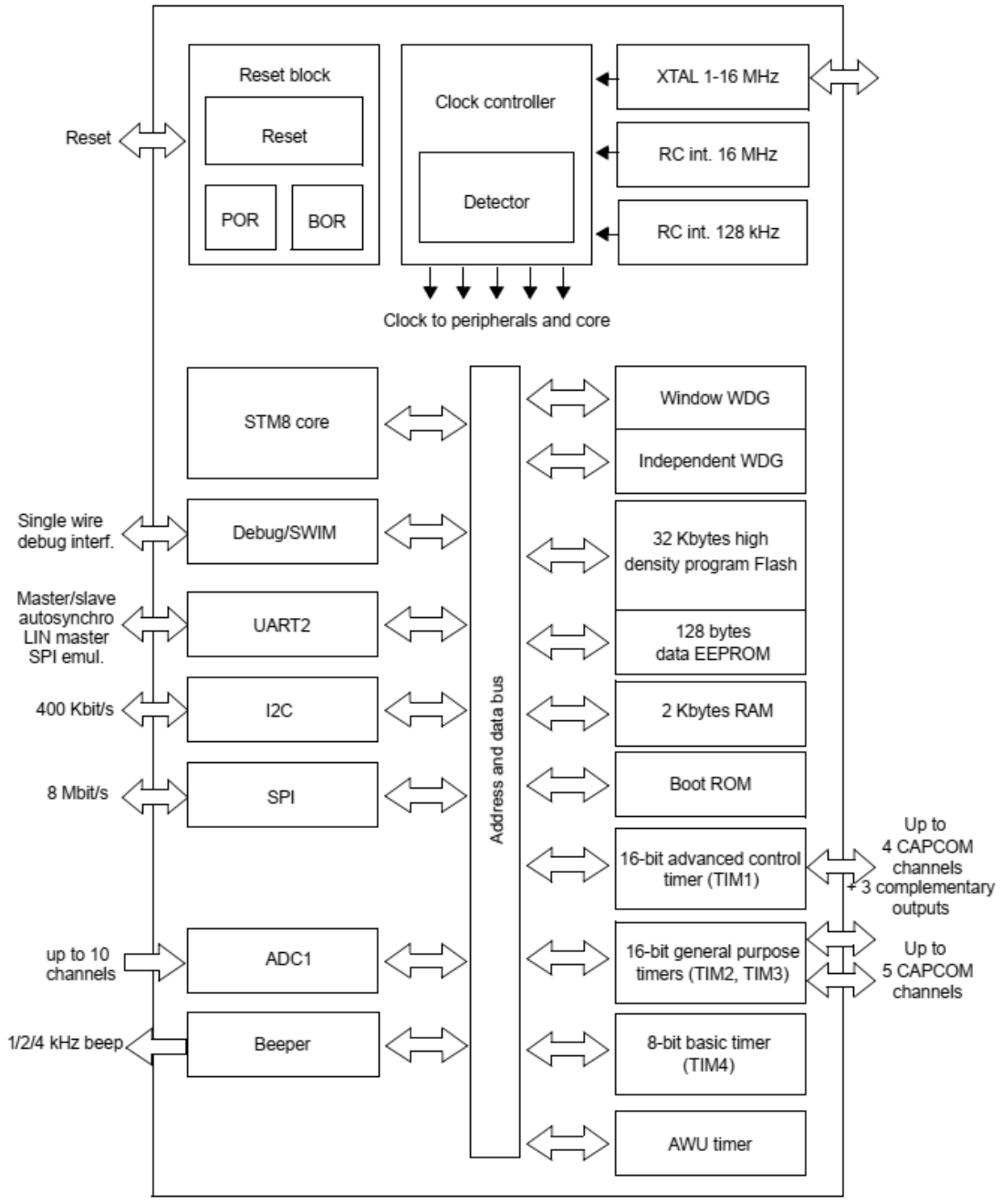

Рис 3.6 блок-схема мікроконтролера STM8S005K6

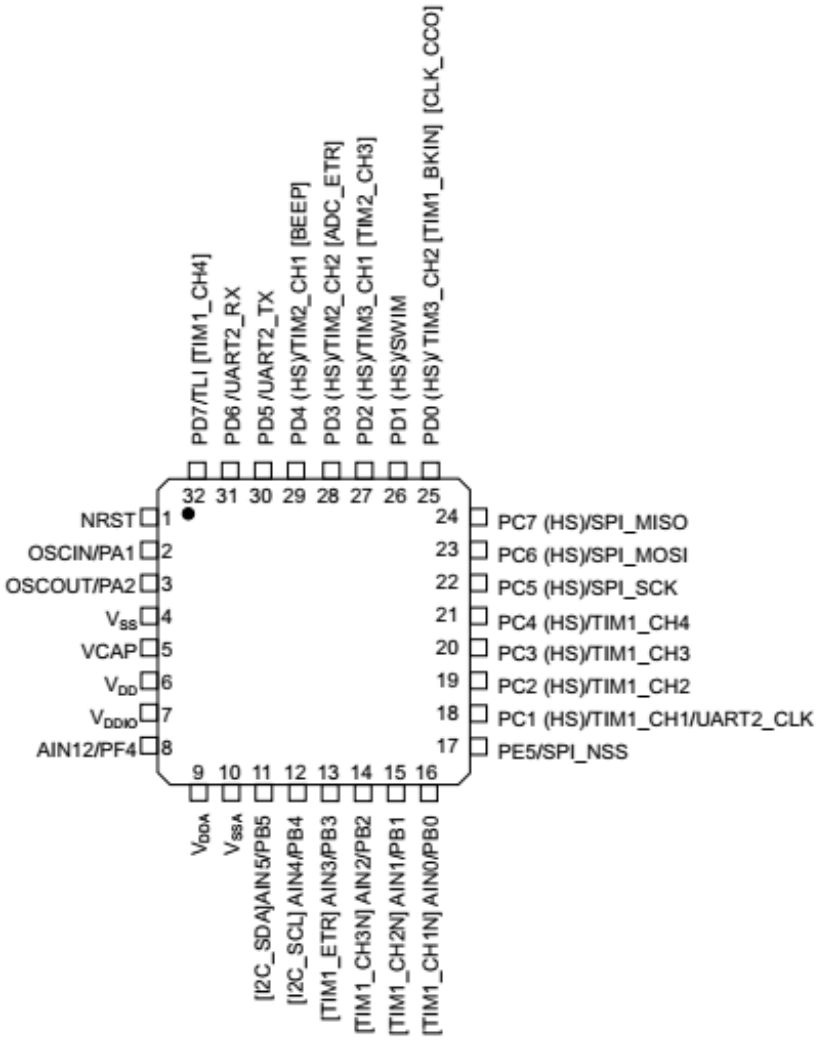

Рис 3.7 мікроконтролер в корпусі LQFP-32

Загалом, цей мікроконтролер має дуже широкий вибір можливостей для розробки вбудованих систем. Особливість ядра виконана в гарвардській архітектурі та 3-ступеневому конвеєрі, що дозволяє забезпечити швидку та ефективну роботу. Пам'ять Flash/EEPROM середньої щільності дозволяє зберегти програмне забезпечення та дані, і має високу витривалість та надійність. Керування тактуванням дозволяє гнучко досягати роботи мікроконтролера, включаючи режими низького енергоспоживання. Керування перериваннями та таймери роблять цей мікроконтролер дуже гнучким та можуть розробникам створювати різноманітні застосунки. Інтерфейси комунікації та АЦП можуть взаємодіяти із зовнішніми пристроями та отримувати дані з різних джерел [26].

3.3 Розробка програмного забезпечення для STM8S005K6

Для зручної розробки програмного забезпечення використовується стандартна периферійна бібліотека STM8S/A STM8S_StdPeriph_Lib від компанії STMicroelectronics.

Периферійна бібліотека STM8S/A - це стандартний набір функцій для програмування мікроконтролерів STM8S і STM8A в середовищі обраної розробки. Ця бібліотека надає функції для управління основними периферійними пристроями, такими як GPIO, таймери, переривання, інтерфейси зв'язку, АЦП [27].

STM8S/A - це дуже потужне рішення з великим набором периферійних функцій, а периферійна бібліотека STM8S/A дозволяє використовувати ці функції за допомогою кількох простих викликів функцій. Наприклад, якщо вам потрібно налаштувати пін на виході GPIO, скористайтеся відповідною функцією з бібліотеки [28].

В галузевих вбудованих системах STM8S/A підтримується широкий спектр різних розробних середовищ, таких як IAR Embedded Workbench, COSMIC CXST8 та ST Visual Develop (STVD) та ін. Периферійна бібліотека STM8S/A розроблена так, щоб працювати з будь-яким із цих середовищ. Вона є однією з найбільш популярних периферійних бібліотек для мікроконтролерів STM8S/A і її досить легко знайти в Інтернеті [29].

Задіяні такі блоки МК :

- Внутрішній тактовий RC генератор на 16МГц
- Аналогово-цифровий перетворювач
- Порти вводу/виводу
- Таймер 2 і 4
- Інтерфейс програмування МК SWIM

Програма складається з двох частин - налаштування периферії та головний безкінечний цикл роботи з периферією МК.

Робота з кожним задіяним блоком МК виконана в окремих файлах, які підключаються за допомогою дерективи компілятора include.

```
#include "stm8s.h"
#include "sub.h"
#include "hd44780.h"
#include "keyboard.h"
#include "timer2.h"
#include "timer4.h"
#include "adc.h"
```

Рис 3.8 Перелік підключених заголовних файлів

stm8s.h - стандартна периферійна бібліотека STM8S/A

sub.h – загальні процедури

hd44780.h – Бібліотека дисплею 1602

keyboard.h – робота з клавіатурою

timer2.h – Робота з таймером 2

timer4.h – робота з таймером 4

adc.h – Налаштування та робота з аналого-цифровим перетворювачем

Налаштування периферії проходить таким чином:

- налаштувати тактування від внутрішнього 16 MHz high-speed internal RC oscillator.
- налаштувати вихід керування відключення акумулятора від зарядного пристрою (ЗП)
- Відключити акумулятор від ЗП
- Timer4 як системний тік кожні 50мкс для програмних затримок
- Timer2 в ШИМ-режим для керування струмом і напругою

  канал1 – напруга, канал2 – струм
- Встановити початкові параметри на виході
- Налаштування АЦП для вимірювання параметрів
- Налаштування клавіатури
- дозволити переривання
- налаштування дисплея в 4-рьох провідний режим (очистка, дисплея, вивести стартові заставку на дисплей, затримка 2с)

А основний цикл працює:

Виводить напругу та струм виставлені і виміряні на дисплей і показує на дисплеї температуру вихідного керуючого транзистора у відносних одиницях. У разі перегріву транзистора ЗП програмо відмикається від електричного кола. Якщо

напруга була занадто велика – пробило транзистор, ЗП відключається від акумулятора. Якщо акумулятор підключений і заряджається то на дисплеї світиться 1, в іншому випадку 0.

Реалізовано управління кнопками та можливість поміряти напругу на акумуляторі знявши напругу зарядки .

3.4 Опис програмного коду

Timer4.c

Налаштовуємо Т4 на виклик переривання кожні 50мкс для формування затримок при тактовій частоті МК 16 000 000 Гц. Встановлюємо прескалер на 16 (TIM4_PRESCALER_16) - буде 1 000 000 Гц.

Лічильник T4_counter = 49, таймер рахує до (49+1)= 50.

Частота спрацювання Т4 1 000 000 / 50 = 20 000Гц, що відповідає 50мкс.

З інтервалом 50мкс викликається переривання від таймера Т4, яке описане в файлі stm8s_it.c.

```
//переривання від Т4 кожні 50мкс
 INTERRUPT_HANDLER(TIM4_UPD_OVF_IRQHandler, 23)
 {
        systic ++;
        TIM4_ClearITPendingBit(TIM4_IT_UPDATE);
}
```

Рис 3.9 Переривання кожні 50мкс

На базі змінної systic програмно формуються затримки, які описані в файлі sub.c.

```
void DelayUs(uint16_t del){
//меньш ніж 50мкс бути не може бо Т4 спрьовую кожні 50мкс
        uint8_t tmp;
        if (del < 50)
         del = 50;
L1:
        tmp = systic;
        while(tmp == systic){
        }
        del -= 50;
        if (del < 50)
         return;
        else
   goto L1;
}
```

Рис 3.10 Програма мікросекундних затримок

Timer2.c

Налаштовуємо Т2 для роботи в режимі двоканального широтно-імпульсного сигналу з частотою 32КГц для керування вихідною напругою та струмом пристрою. Функції встановлення вихідної напруги та струму рис 3.13, 3.14.

```
void Timer2_pwm_U_U(float Uset){
  //програмне обмеження напруги на виході
        if (Uset > 17)
         Uset = 17;
        //set pwm width
        ch1_pwm_U = (uint16_t)(Uset * pwmKU);
        //виставити на каналі1 потрібну скважність ШИМ
        TIM2_SetCompare1(ch1_pwm_U);
}
```

Рис3.11 Встановлення напруги на виході

```
void Timer2_pwm_I_I(float Iset){
  //програмне обмеження струму
  //наш давач струму не може міряти більше ніж 5А
        if (Iset > 4.95)
         Iset = 4.95;
        //set pwm width
        ch2_pwm_I = (uint16_t)(Iset * pwmKI) + 305;
        //виставити на каналі2 потрібну скважність ШИМ
        TIM2_SetCompare2(ch2_pwm_I);
}
```

Рис3.12 Встановлення струму на виході

adc.c

Робота аналого-цифрового перетворювача реалізована в 4-ьох канальному режимі.

Входи для роботи АЦПР:

- GPIOB GPIO_PIN_0 - напруга на АКБ
- GPIOB GPIO_PIN_1 - поточний струм яким заряджається АКБ
- GPIOB GPIO_PIN_2 - температура вихідного регулюючого транзистора
- GPIOB GPIO_PIN_3 - поточна напруга на виході регулюючого транзистора

```
uint16_t ADC_Get_ADC(uint8_t channel){
uint16_t v1;
ADC1->CSR &= (uint8_t)(~ADC1_CSR_CH);
switch(channel){
  case 0:
    ADC1->CSR |= (uint8_t)(ADC1_CHANNEL_0);
  break;
  case 1:
    ADC1->CSR |= (uint8_t)(ADC1_CHANNEL_1);
  break;
  case 2:
    ADC1->CSR |= (uint8_t)(ADC1_CHANNEL_2);
  break;
  case 3:
    ADC1->CSR |= (uint8_t)(ADC1_CHANNEL_3);
  break;
}
ADC1_ClearITPendingBit(ADC1_IT_EOC);
ADC1_StartConversion();
while((ADC1->CSR & ADC1_IT_EOC) != ADC1_IT_EOC){
}
v1 = ADC1_GetConversionValue();
return v1;
}
```

Рис 3.13 Читання АЦП по вибраному каналу

keyboard.c

Реалізовано опитування клавіатури яка підключена до двох виводів мікроконтролера, з програмним захистом від брязкоту контактів. Процедура обробки нажимання трьох кнопок розділяючи їх на короткі і довгі.

Якщо клавіша нажата менше 2мс то це вважається брязкотом і вона не обробляється. Короткому нажиманню відповідає інтервал до 500мс, від 500мс до 3000мс – довге нажимання. Все що більше 3000мс ігнорується.

```
#define key_no          0x06    // 0b00000110
#define key_down        0x04    // 0b00000100
#define key_up          0x02    // 0b00000010
#define key_select      0x00    // 0b00000000
#define key_down_long   0x84    // 0b10000100
#define key_up_long     0x82    // 0b10000010
#define key_select_long 0x80    // 0b10000000
```

Рис 3.14 Бітові маски кодів клавіш

```
void FloatToStr(float num, char* buf, uint8_t len){
        uint16_t tmp;
  buf[0] = ' '; buf[1] = '0'; buf[2] = '.'; buf[3] = '0'; buf[4] = '0'; buf[5] = 0;
        if (num < 0.01)
         goto end_;
        if (num >=10.0){
                tmp = (int)(num/10.0);
          buf[0] = '0' + tmp;
                num-= tmp*10;
        }
        tmp = (int)(num);
        buf[1] = '0' + tmp;
        num -= tmp;
        tmp = (int)(num*10.0);
        buf[3] = '0' + tmp;
        buf[4] = '0' + (int)(num*100.0) - tmp*10;
end_:;
        if (buf[0] == '0')
         buf[0] = ' ';
        if (len == 6)
   return;
}
```

Рис 3.15 Конвертація числа в стрічку символів

Для виводу на дисплей реалізована процедура конвертації числових значень параметрів в текстову стрічку в файлі sub.c. Рис3.15.

3.5 Створення друкованої плати

Для створення друкованої плати необхідно зробити трасування друкованої плати. Трасування друкованої плати - це процес створення маршруту провідників на поверхні друкованої плати, який з'єднує електричні компоненти і забезпечує передачу сигналів. Це важливий етап у процесі проектування електронних пристроїв, де точність та якість трасування можуть суттєво впливати на роботу пристрою. Точне трасування може забезпечити стабільність і підвищення продуктивності приладу, а також додаткові функції, такі як економія електроенергії та поліпшення сигналу [30].

Це зручно зробити в програмі Sprint-Layout(Рис3.16).

Етапи створення друкованої плати:

1) Трасування друкованої плати
2) Друк плати на труйному принтері на прозорій плівці Lomond.

3) Нанесення фоторезистивної плівки на фольгований стеклотекстолит.
4) Експозиція в ультрафіолетовому світлі стеклотекстолітової плати з фоторезестивною плівкою через роздрукованим раніше шаблоном.
5) Змивання незасвічених ділянок фоторезисту в розчині кальцинованої соди.
6) Травлення плати розчином хлорного заліза.
7) Підготовка друкованої плати до встановлення елементів(Свердління, лудіння)
8) Монтаж друкованої плати.

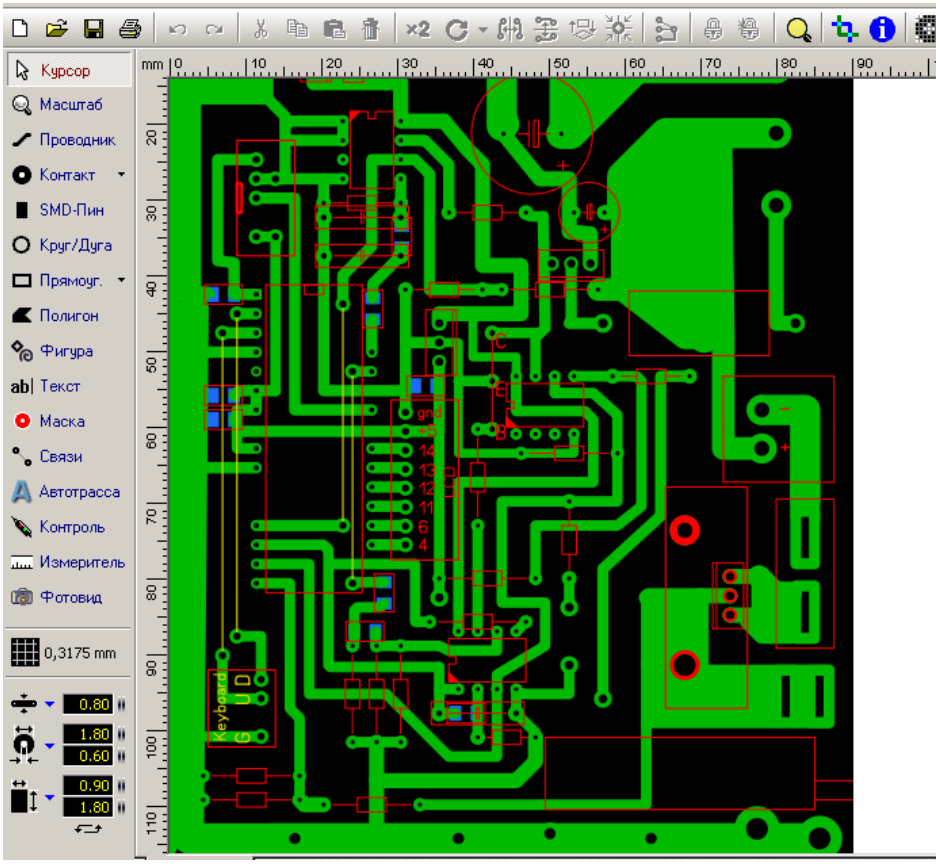

Рис3.16 Трасування друкованої плати

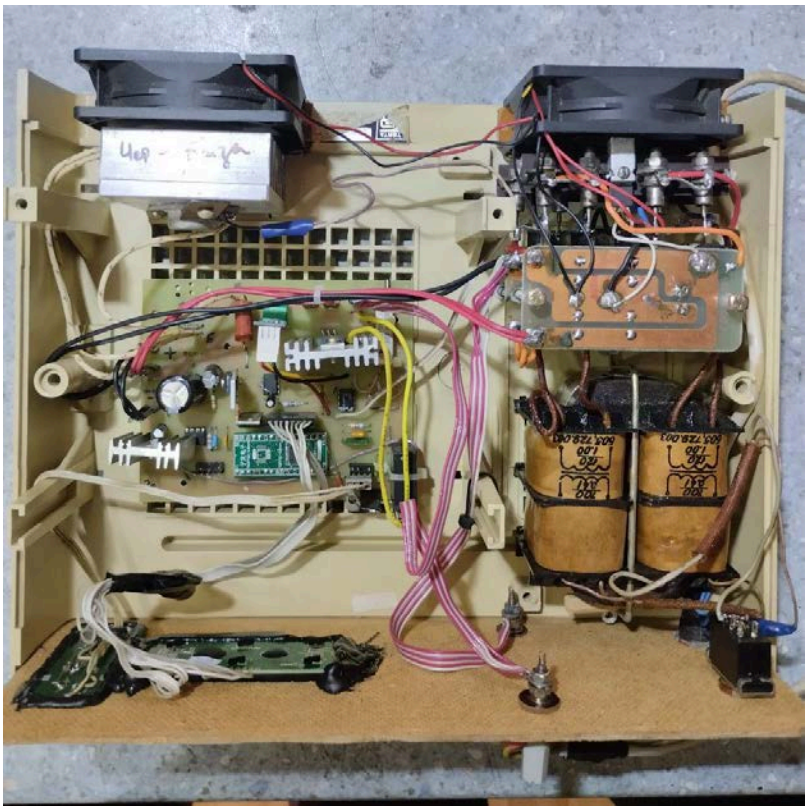

Рис 3.17 Виконаний монтаж в корпус

Після виготовлення плати, було вмонтовано плату та інші компоненти в корпус приладу, забезпечення надійної фіксації та з'єднання їх між собою. Рис 3.17

3.6 Підключення АКБ

Зарядний пристрій має три кнопки: up, down, select, power.Також дисплей на який виводиться все потрібна інформація. До двох виводів плюс там мінус потрібно підключити акумулятор (Рис3.18)

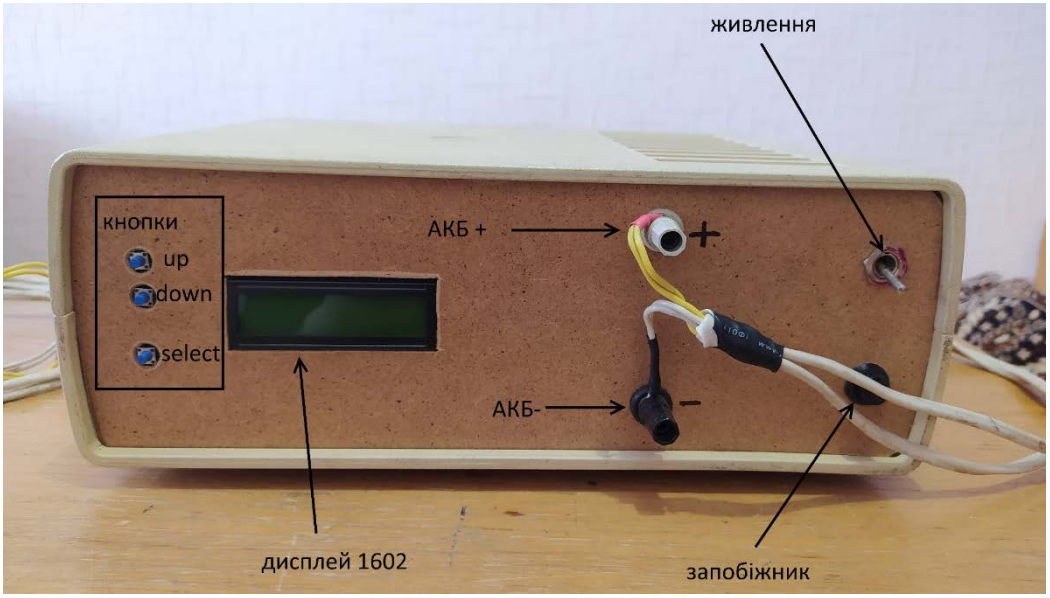

Рис3.18 Вигляд ЗП АКБ

Перш ніж підключати акумулятор необхідно ввімкнути прилад та подивитися які параметри виставлені, для того щоб не зіпсувати прилад чи акумулятор.

Рис 3.19 Дисплей ЗП не підключеного до акумулятора

За замовчуванням коли вмикається прилад там налаштовані параметри для заряджання машинного акумулятора(Рис3.19), проте їх можна змінити на потрібні для різних акумуляторів за допомогою кнопок керування. Якщо налаштовані параметри підходять для вас то можна підключати акумулятор плюс до плюса, мінус на мінус рис 3.20.

Рис 3.20 Акумулятор підключений до ЗП

Рис 3.21 Значення дисплею

На дисплеї в першому стовпчику відображається задана на вихідному каналі ЗП напруга та струм. В другому стовпчику показується поточна напруга на АКБ та поточний струм який споживає АКБ. У третьому стовпчику відображається символ (U/I), що вказує на те, який параметр можна змінювати на вихідному каналі ЗП, показник того, чи підключений АКБ (1/0) та температура транзистора відображається у відносних одиницях рис 3.21.

3.7 Встановлення необхідних параметрів

Після того як ми підключили ЗП до живлення, нам необхідно змінити вихідну напругу. Для цього необхідно подивитися на дисплей, якщо він показує U то натискаючи кнопки UP і Down буде змінюватися напруга відповідно збільшуватись або зменшуватись. Якщо ж дисплей показує I то необхідно один раз натиснути кнопку select і прилад перейде в режим налаштування вихідної напруги (рис3.22, рис3.23).

Для налаштування струму така ж сама послідовність дій, лише для налаштування струму дисплей має показувати I.

Рис3.22 Встановлення необхідної напруги

Рис3.23 Встановлення необхідного струму

Рис3.24 Режим вимірювання напруги на акумуляторів

Також затримавши кнопку select пристрій перейде в режим вимірювання напруги на акумуляторі рис 3.24. Затримавши ще раз цю кнопку поверне в режим заряджання.

ВИСНОВОК ДО РОЗДІЛУ 3

Отже, в даному розділі було досліджено електричну принципову схему та її головні вузли, що є важливим етапом в розробці будь-якої електронної пристрою. Було визначено необхідний перелік елементів, які будуть використовуватися для створення пристрою.

Також було розглянуто мікроконтролер STM8S005K6 та його основні блоки програмного забезпечення. Детально описано процес створення друкованих плат, який дозволяє виготовити електронний пристрій з високою точністю та якістю. Загалом, отримані знання дозволять ефективно розробити та виготовити електронний пристрій згідно з вимогами та стандартами.

В даному розділі описана інструкція для користувача ЗП. Наведено інформацію про кнопки управління та спосіб застосування.

Отже, можна зробити висновок, що зарядний пристрій є досить простим у використанні, має зручні кнопки керування та обов’язковий дисплей, який відображає всю необхідну інформацію. Перед підключенням акумулятора необхідно переконатися, що параметри зарядного пристрою виставлені правильно для даного типу акумулятора. Для зміни вихідної напруги та струму необхідно просто користуватися кнопками керування. Таким чином, зарядний пристрій є надійним та зручним пристроєм для заряджання різних типів акумуляторів.

ВИСНОВКИ

Під час виконання дипломної роботи було створено зарядний пристрій для машинних акумуляторів. Даний пристрій дає змогу користувачу безпечно зарядити машинний акумулятор, поміряти його напругу. В ньому реалізований користувацький інтерфейс та зручні кнопки управління за допомогою який зручно налаштовувати необхідні параметри.

В першому розділі було визначено що зарядка автомобільного акумулятора є дуже важливою його процедурою для продовження терміну служби. Для забезпечення правильної зарядки необхідно звернути відповідний тип і модель зарядного пристрою, а також врахувати його особливості. Використання мікроконтролерів, зокрема STM8S005K6, є досить цікавим варіантом розробки зарядних пристроїв для автомобілів. Програмування STM8S005K6 за допомогою мови C або STM8CubeMX забезпечує більшу гнучкість та продуктивність розробки. Бібліотеки програмного забезпечення, такі як STM8S Standard Peripheral Libraries, можуть зробити розробку швидкою та з меншими тривогами. У загальному, в даному розділі йдеться не тільки про автомобільних акумуляторів та їх властивості, але й про способи їх зарядки та можливості використання мікроконтролерів для розробки зарядних пристроїв.

У другому розділі було охарактеризовано та проаналізовано обраний проект. Було описано етапи створення зарядного пристрою та створено власний план. Також було проаналізовано доступні технології та програмне забезпечення для вирішення поставленого завдання та обрано найбільш підходящі. був обраний трансформаторний зарядний пристрій та мікроконтролер STM8S005C6/K6 для розробки проекту зарядного пристрою.

Обрані програми та пристрої допомогли створити ефективний та безпечний зарядний пристрій, який може позитивно вплинути на якість життя та продуктивність роботи.

У третьому розділі проведено дослідження електричного принципу та

схеми головного вузла, яка є обов’язковою складовою процесу розробки будь-якого електронного пристрою. Окреслено список деяких елементів, які використовуються для створення пристрою, також висвітлено основні блоки програмного забезпечення мікроконтролера STM8S005K6. Більш докладно

розповідається про процес створення друкованих плат, який дозволяє виготовити електронний пристрій з високою точністю та якістю. Освоєні знання дозволяють ефективно розробити та виготовити електронний пристрій, відповідним вимогам та стандартам. Також було продемонстровано роботу зарядного пристрою, користувацький інтерфейс. Розказано як виставляти необхідні параметри та перейти в інший режим роботи приладу. У розділі представлений готовий прилад, розроблений протягом всієї дипломної роботи. Надано загальні вказівки щодо використання.

Отже, під час виконання дипломної роботи було розроблено ефективний та безпечний зарядний пристрій для машинних акумуляторів з користувацьким інтерфейсом та зручними кнопками управління. Дослідження електричного принципу та створення друкованих плат дозволили створити електронний пристрій з високою точністю та якістю. Використання мікроконтролера STM8S005K6 з бібліотеками програмного забезпечення дозволило розробити пристрій з більшою гнучкістю та продуктивністю розробки. Готовий прилад може позитивно вплинути на якість життя та продуктивність роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Billings K., Pressman A. I. Switching Power Supply Design, 3rd Ed. 3-тє вид. McGraw-Hill Professional, 2008. 744 с.
2. Designing Control Loops for Linear and Switching Power Supplies. Artech House Publishers, 2012.
3. Кобзарьов, Ю. В. Теорія та практика конструкції пристроїв індуктивної зарядки високовольтних акумуляторів: Ю. В. Кобзарьов. – Київ : НТУУ "КПІ", 2009. – 294 с.
4. Maniktala S. Switching power supply design & optimization. McGraw-Hill Professional, 2004. 386 с.
5. Маляр В. С. Теоретичні основи електротехніки. Львів: Львівська політехніка, 2018. – 416 с.
6. Матвієнко М.П. Основи електроніки. Підручник. Київ: Ліра-К, 2021.-360 с.
7. Winfield H. The art of electronics. 3-тє вид. Cambridge : Cambridge University Press, 2015. 1224 с.
8. Коман Б. Основи комп'ютерної електроніки / Б. Коман, М. Мисько. – Львів: ЛНУ, 2019. – 430 с.
9. Barr M. Programming Embedded Systems in C and C ++. O'Reilly Media, Inc., 1999. 194 с.
10. Mazidi M. A., Naimi S., Naimi S. Avr Microcontroller and Embedded Systems: Pearson New International Edition. Pearson Education, Limited, 2015. 749 с.
11. Predko M. Programming and customizing the PIC microcontroller. 3-тє вид. New York : McGraw Hill, 2008. 1263 с.
12. Rizvi S. R. Microcontroller Programming: An Introduction. Taylor & Francis Group, 2016. 546 с.
13. Mazidi M. A., Naimi S., Naimi S. Avr microcontroller and embedded systems the: using assembly and C. Pearson Education, Limited, 2011. 791 с.
14. Sanchez J. Microcontroller programming: The microchip PIC. Boca Raton, FL : CRC Press, 2007. 804 с.
15. McKinlay R., Causey D., Mazidi M. A. PIC Microcontroller and Embedded Systems: Using Assembly and C for PIC18. MicroDigitalEd, 2016. 630 с.
16. Barr M. Programming embedded systems: With C and GNU development tools. 2-

ге вид. Sebastopol, CA : O'Reilly, 2007. 301 с.
17. Zurawski R. Embedded Systems Handbook: Embedded Systems Design and Verification. Taylor & Francis Group, 2018. 666 с.
18. Massa A., Barr M. Programming Embedded Systems: With C and GNU Development Tools, 2nd Edition. O'Reilly Media, Inc., 2006. 301 с.
19. Scherz P. Practical Electronics for Inventors. 2-ге вид. McGraw-Hill/TAB Electronics, 2006. 952 с.
20. Схемотехніка: Пристрої цифрової електроніки /Рябенький В.М. та ін. Київ 399с.
21. Etc M. N. Power Electronics: Converters, Applications and Design. 3-тє вид. NJ : Wiley, 2003. 824 с.
22. Bates D. J., Malvino A. P. Electronic Principles. McGraw Hill Higher Education, 2007.
23. Buchla D. Electric Circuits Fundamentals. 7-ме вид. Not Avail, 2006. 400 с.
24. Maksimovic D., Erickson R. W. Fundamentals of Power Electronics. Springer, 2012. 908 с.
25. Матюшов М.В. Початок роботи з мікроконтролерами stm8. Київ, 2016. - 208 с.
26. Mazidi M. a. The 8051 microcontroller and embedded systems: Using Assembly and C. 2-ге вид. Upper Saddle River, N.J : Pearson/Prentice Hall, 2006. 626 с.
27. Grace T. Programming and interfacing ATMEL AVR microcontrollers. 2016. 272 с.
28. Wright C., Sloss A., Symes D. ARM system developer's guide: designing and optimizing system software. Elsevier Science & Technology Books, 2004. 689 с.
29. Berger A. Embedded systems design: An introduction to processes, tools, and techniques. Lawrence, Kan : CMP Books, 2002. 237 с.
30. Brown M. Power supply cookbook. Boston : Butterworth-Heinemann, 1994. 238 с.

ДОДАТКИ

Аркушів 14

Київ 2023

ДОДАТОК А

Зарядний пристрій для зарядки машинних акумуляторів

main.c

```
#include "stm8s.h"

#include "sub.h"
#include "hd44780.h"
#include "keyboard.h"
#include "timer2.h"
#include "timer4.h"ë
#include "adc.h"

/* Private defines -----------------------------------------------------------*/
/* Private function prototypes -----------------------------------------------*/
/* Private functions ---------------------------------------------------------*/

void main(void)
{
 static uint8_t ui;
 uint16_t Tvt;
 bool K1on = TRUE;
 float Utmp = 0;
 CLK_HSIPrescalerConfig(CLK_PRESCALER_HSIDIV1);
 GPIO_Init(GPIOE, GPIO_PIN_5, GPIO_MODE_OUT_PP_LOW_FAST);
 GPIO_WriteLow(GPIOE, GPIO_PIN_5);
 Timer4_Init();
 Timer2_Init();
 U = 14.6;
 I = 0.5;
 Timer2_pwm_U_U(U);
 Timer2_pwm_I_I(I);
 ADC_Init();
 Keyboard_Init(&key_previous);
 enableInterrupts();
 HD44780_Init();
 HD44780_Clear();
 HD44780_Out(0, 2, "Charger V1.1");
 HD44780_Out(1, 2, "KWWSoft 2022");
 DelayMs(2000);
 HD44780_Clear();
 ui = 0;
 HD44780_Out(0, 13, "U");

 while (1)
  {
    HD44780_Out_Float(0, 0, U, 6);
    HD44780_Out_Float(0, 6, ADC_Get_U(), 6);

    HD44780_Out_Float(1, 1, I, 5);
    HD44780_Out_Float(1, 7, ADC_Get_I(), 5);

    Tvt = ADC_Get_ADC(2);
    HD44780_Debug_Out_Int_Dec_5(1, 11, Tvt);
    if (Tvt < 520){
      K1on = FALSE;
    }
    else{
      K1on = TRUE;
    }

    if (ADC_Get_U() > 17.5){
       K1on = FALSE;
    }
    else{
       K1on = TRUE;
    }
```

```
if (K1on){
  GPIO_WriteHigh(GPIOE, GPIO_PIN_5);
  HD44780_Out(0, 15, "1");
}
else {
  GPIO_WriteLow(GPIOE, GPIO_PIN_5);
  HD44780_Out(0, 15, "0");
}
switch(Keyboard_Key(&key_previous)){
  case key_no:
  break;

  case key_down:
    if (ui == 0){
      if (U > 0){
        U -= 0.1;
      }
     Timer2_pwm_U_U(U);
   }
   else{
      if (I > 0){
        I -= 0.5;
      }
     Timer2_pwm_I_I(I);
   }
  break;

  case key_up:
    if (ui == 0){
      if (U < 17){
        U += 0.1;
      }
     Timer2_pwm_U_U(U);
   }
   else{
      if (I < 5){
        I += 0.5;
      }
     Timer2_pwm_I_I(I);
   }
  break;

  case key_up_long:
  break;

  case key_select:
    if (ui == 0){
      ui = 1;
     HD44780_Out(0, 13, "I");
   }
   else{
      ui = 0;
     HD44780_Out(0, 13, "U");
   }
  break;

  case key_select_long:
    if (Utmp == 0){
      Utmp = U;
      U = 0;
      Timer2_pwm_U_U(U);
    }



   else{
      U = Utmp;
      Utmp = 0;
      Timer2_pwm_U_U(U);
```

```
        }
      break;
    }
  }



}
```

sub.c

```
#include "stm8s.h"
//#include "delay.h"
#include "sub.h"
#include <assert.h>
//*******************************************************************************
uint8_t systic;
//*******************************************************************************
//*******************************************************************************
void FloatToStr(float num, char* buf, uint8_t len){
       //len = 6
       //00.00
       //
       //len = 5
       //0.00
       //
       //buf len = 5+zero = 6
       uint16_t tmp;
  buf[0] = ' '; buf[1] = '0'; buf[2] = '.'; buf[3] = '0'; buf[4] = '0'; buf[5] = 0;
       //
       if (num < 0.01)
        goto end_;
       //x0.00
       if (num >=10.0){
              tmp = (int)(num/10.0);
         buf[0] = '0' + tmp;
              num-= tmp*10;
       }
       tmp = (int)(num);
       buf[1] = '0' + tmp;
       num -= tmp;
       tmp = (int)(num*10.0);
       buf[3] = '0' + tmp;
       buf[4] = '0' + (int)(num*100.0) - tmp*10;
end_:;
       if (buf[0] == '0')
        buf[0] = ' ';
       //00.00 len=6
       if (len == 6)
   return;
  //0.00  len=5
       buf[0] = buf[1];
       buf[1] = '.';
       buf[2] = buf[3];
       buf[3] = buf[4];
       buf[4] = 0;
}




//*******************************************************************************
void DelayUs(uint16_t del){
        uint8_t tmp;
       if (del < 50)
```

```
        del = 50;
L1:
       tmp = systic;
       while(tmp == systic){
       }
       del -= 50;
       if (del < 50)
        return;
       else
   goto L1;
}
//*******************************************************************************
void DelayMs(uint16_t del){
       if (del < 1)
        del = 1;
L1:
       DelayUs(1000);
  del --;
       if (del > 0)
   goto L1;
}
```

stm8s_it.c

```
/* Includes ------------------------------------------------------------------*/
#include "stm8s_it.h"
extern uint8_t systic;              //внутренний счетчик который идет +1 каждые 50мкс


/** @addtogroup Template_Project
  * @{
  */

/* Private typedef -----------------------------------------------------------*/
/* Private define ------------------------------------------------------------*/
/* Private macro -------------------------------------------------------------*/
/* Private variables ---------------------------------------------------------*/
/* Private function prototypes -----------------------------------------------*/
/* Private functions ---------------------------------------------------------*/
/* Public functions ----------------------------------------------------------*/


/**
  * @brief TRAP Interrupt routine
  * @param  None
  * @retval None
  */
INTERRUPT_HANDLER_TRAP(TRAP_IRQHandler)
{
}

/**
  * @brief Top Level Interrupt routine.
  * @param  None
  * @retval None
  */
INTERRUPT_HANDLER(TLI_IRQHandler, 0)

{
}

/**
  * @brief Auto Wake Up Interrupt routine.
  * @param  None
  * @retval None
  */
INTERRUPT_HANDLER(AWU_IRQHandler, 1)
{
}
```

```
/**
  * @brief Clock Controller Interrupt routine.
  * @param  None
  * @retval None
  */
INTERRUPT_HANDLER(CLK_IRQHandler, 2)
{
}

/**
  * @brief External Interrupt PORTA Interrupt routine.
  * @param  None
  * @retval None
  */
INTERRUPT_HANDLER(EXTI_PORTA_IRQHandler, 3)
{
}

/**
  * @brief External Interrupt PORTB Interrupt routine.
  * @param  None
  * @retval None
  */
INTERRUPT_HANDLER(EXTI_PORTB_IRQHandler, 4)
{
}

/**
  * @brief External Interrupt PORTC Interrupt routine.
  * @param  None
  * @retval None
  */
INTERRUPT_HANDLER(EXTI_PORTC_IRQHandler, 5)
{
}

/**
  * @brief External Interrupt PORTD Interrupt routine.
  * @param  None
  * @retval None
  */
INTERRUPT_HANDLER(EXTI_PORTD_IRQHandler, 6)
{
}

/**
  * @brief External Interrupt PORTE Interrupt routine.
  * @param  None
  * @retval None
  */
INTERRUPT_HANDLER(EXTI_PORTE_IRQHandler, 7)
{
}

/**
  * @brief SPI Interrupt routine.
  * @param  None
  * @retval None
  */
INTERRUPT_HANDLER(SPI_IRQHandler, 10)
{
}

/**
  * @brief Timer1 Update/Overflow/Trigger/Break Interrupt routine.
  * @param  None
  * @retval None
  */
INTERRUPT_HANDLER(TIM1_UPD_OVF_TRG_BRK_IRQHandler, 11)
```

```
{
}

/**
  * @brief Timer1 Capture/Compare Interrupt routine.
  * @param  None
  * @retval None
  */
INTERRUPT_HANDLER(TIM1_CAP_COM_IRQHandler, 12)
{
}

/**
  * @brief Timer2 Update/Overflow/Break Interrupt routine.
  * @param  None
  * @retval None
  */
 INTERRUPT_HANDLER(TIM2_UPD_OVF_BRK_IRQHandler, 13)
 {
  /* In order to detect unexpected events during development,
     it is recommended to set a breakpoint on the following instruction.
  */
 }

/**
  * @brief Timer2 Capture/Compare Interrupt routine.
  * @param  None
  * @retval None
  */
 INTERRUPT_HANDLER(TIM2_CAP_COM_IRQHandler, 14)
 {
  /* In order to detect unexpected events during development,
     it is recommended to set a breakpoint on the following instruction.
  */
 }

/**
  * @brief Timer3 Update/Overflow/Break Interrupt routine.
  * @param  None
  * @retval None
  */
 INTERRUPT_HANDLER(TIM3_UPD_OVF_BRK_IRQHandler, 15)
 {
 }

/**
  * @brief Timer3 Capture/Compare Interrupt routine.
  * @param  None
  * @retval None
  */
 INTERRUPT_HANDLER(TIM3_CAP_COM_IRQHandler, 16)
 {
  /* In order to detect unexpected events during development,
     it is recommended to set a breakpoint on the following instruction.
  */
 }

/**
  * @brief I2C Interrupt routine.
  * @param  None
  * @retval None
  */
INTERRUPT_HANDLER(I2C_IRQHandler, 19)
{
}

/**
  * @brief UART2 TX interrupt routine.
  * @param  None
  * @retval None
```

```
  */
 INTERRUPT_HANDLER(UART2_TX_IRQHandler, 20)
 {
 }

/**
  * @brief UART2 RX interrupt routine.
  * @param  None
  * @retval None
  */
 INTERRUPT_HANDLER(UART2_RX_IRQHandler, 21)
 {
 }

/**
  * @brief ADC1 interrupt routine.
  * @par Parameters:
  * None
  * @retval
  * None
  */
 INTERRUPT_HANDLER(ADC1_IRQHandler, 22)
 {
 }

/**
  * @brief Timer4 Update/Overflow Interrupt routine.
  * @param  None
  * @retval None
  */
//переривання від Т4 кожні 50мкс
 INTERRUPT_HANDLER(TIM4_UPD_OVF_IRQHandler, 23)
 {
       systic ++;
       TIM4_ClearITPendingBit(TIM4_IT_UPDATE);
}

/**
  * @brief Eeprom EEC Interrupt routine.
  * @param  None
  * @retval None
  */

INTERRUPT_HANDLER(EEPROM_EEC_IRQHandler, 24)
{
}
```

HD44780.c

```
#include "stm8s.h"
#include "HD44780.h"
#include "sub.h"
#include <assert.h>
//*****************************************************************************
char digits[]={'0','1', '2', '3','4','5','6','7','8','9','A','B','C','D','E','F'};
char str[6];
//*****************************************************************************
//define port and pins fol lcd control
//port and pin RS
#define hd44780_rs_port           (GPIOC)
#define hd44780_rs_pin            (GPIO_PIN_2)
//port and pin E
#define hd44780_e_port            (GPIOC)
#define hd44780_e_pin             (GPIO_PIN_3)
//port and nibble DATA
//  HD44780_DATA GPIO_PIN_LNIB                GPIO_PIN_HNIB
//     D7           P3                        P7
//     D6           P2                        P6
```

```
//     D5            P1                          P5
//     D4            P0                          P4
#define hd44780_data        (GPIOC)
#define hd44780_nibble              GPIO_PIN_HNIB

//*****************************************************************************
void WriteLowNibble(unsigned char cmd){
       unsigned char oldV = GPIO_ReadOutputData(hd44780_data);
       GPIO_Write(hd44780_data, (oldV & 0xF0) | (cmd & 0x0F));
       GPIO_WriteHigh(hd44780_e_port, hd44780_e_pin);
       DelayMs(1);
       GPIO_WriteLow(hd44780_e_port, hd44780_e_pin);
       DelayMs(1);
}
//*****************************************************************************
void WriteHighNibble(unsigned char cmd){
       unsigned char oldV = GPIO_ReadOutputData(hd44780_data);
       GPIO_Write(hd44780_data, (oldV & 0x0F) | ((cmd << 4) & 0xF0));
       GPIO_WriteHigh(hd44780_e_port, hd44780_e_pin);
       DelayMs(1);
       GPIO_WriteLow(hd44780_e_port, hd44780_e_pin);
       DelayMs(1);
}
//*****************************************************************************
void SendByte(unsigned char cmd){
       if (hd44780_nibble == GPIO_PIN_LNIB){
              WriteLowNibble(cmd >> 4);
              WriteLowNibble(cmd);
       }
       else{
              WriteHighNibble(cmd >> 4);
              WriteHighNibble(cmd);
       }
}
//*****************************************************************************
void SendCommand(unsigned char cmd){
       GPIO_WriteLow(hd44780_rs_port, hd44780_rs_pin);
       SendByte(cmd);
       DelayUs(50);
}
//*****************************************************************************
void SendData(unsigned char cmd){
       GPIO_WriteHigh(hd44780_rs_port, hd44780_rs_pin);
  SendByte(cmd);
       DelayUs(50);
}
//*****************************************************************************
void HD44780_Clear(void){
       SendCommand(0x1);
}
//*****************************************************************************
void SetLine(int line){
       if (line)     {//1
              SendCommand(0xC0);
       }
       else   {
              SendCommand(0x80);
       }
}
//*****************************************************************************
void HD44780_Out(int line, int pos,  char *str){
       SetLine(line);
       while(pos > 0){
        SendCommand(0x14);
        pos --;
       }
       while(*str)   {
              SendData(*str++);
       }
}
```

```
//*****************************************************************************
void HD44780_Out_Float(int line, int pos,  float num, uint8_t len){
 char buf[6];
 FloatToStr(num, buf, len);
 HD44780_Out(line, pos, buf);
}
//*****************************************************************************
void HD44780_CursorOFF(void){
 SendCommand(0x0C);
}
//*****************************************************************************
void HD44780_CursorSteady(void){
 SendCommand(0x0E);
}
//*****************************************************************************
void HD44780_CursorBlinking(void){
 SendCommand(0x0F);
}
//*****************************************************************************
void HD44780_Init(void){

  GPIO_Init(hd44780_data, hd44780_nibble, GPIO_MODE_OUT_PP_LOW_FAST);
  GPIO_Init(hd44780_rs_port, hd44780_rs_pin, GPIO_MODE_OUT_PP_LOW_FAST);
  GPIO_Init(hd44780_e_port, hd44780_e_pin, GPIO_MODE_OUT_PP_LOW_FAST);
  //
  GPIO_WriteLow(hd44780_e_port, hd44780_e_pin);
       GPIO_WriteLow(hd44780_rs_port, hd44780_rs_pin);
       DelayMs(20);
       SendCommand(0x03);
       DelayMs(5);
       SendCommand(0x03);
       DelayMs(1);
  SendCommand(0x03);
       SendCommand(0x02);
       SendCommand(0x28);
       SendCommand(0x0c);
       SendCommand(0x06);
       HD44780_Clear();
}
//*****************************************************************************
void HD44780_Debug_Out_Int_Dec_5(int line, int pos,  unsigned int num){

str[0] = ' '; str[1] = ' '; str[2] = ' '; str[3] = ' ';str[4] = ' ';str[5] = 0;
              if (num < 10){
                     str[4] = '0' + num;
              }
              else if (num < 100){
                     str[4] = '0' + num % 10;
                     str[3] = '0' + num / 10;
              }
              else if (num < 1000){
                     str[4] = '0' + num % 10;
                     str[3] = '0' + (num / 10) % 10;
                     str[2] = '0' + num / 100;
              }
              else if (num < 10000){
                     str[4] = '0' + num % 10;
                     str[3] = '0' + (num / 10) % 10;
                     str[2] = '0' + (num / 100) % 100;
                     str[1] = '0' + num / 1000;
              }
              else {
                     str[4] = '0' + num % 10;
                     str[3] = '0' + (num / 10)%10;
                     str[2] = '0' + (num / 100)%100;
                     str[1] = '0' + (num / 1000)%1000;
                     str[0] = '0' + num / 10000;
              }
 HD44780_Out(line, pos, str);
}
```

```
//*****************************************************************************
```

KEYBOARD.c

```
#include "stm8s.h"
#include "sub.h"
#include "keyboard.h"
#include <assert.h>
//*****************************************************************************
void Keyboard_Init(char *key){
 GPIO_Init(GPIOA, GPIO_PIN_1|GPIO_PIN_2, GPIO_MODE_IN_PU_NO_IT);
 *key = key_no;
}
//*****************************************************************************
/*
опитування клавіатури з програмним захистом від дрижання
//бітові маски для порту опитування
key_no          0x06    // 0b00000110
key_down      0x04    // 0b00000100
key_up        0x02    // 0b00000010
key_select    0x00    // 0b00000000

довге нажимання аналізується програмно через затримки опитування
*/
char Keyboard_Key(char *last){
 static char tmp;
 unsigned int count = 0;
 tmp = GPIO_ReadInputData(GPIOA) & key_no;
 if (tmp == key_no){
  *last = key_no;
  return key_no;
 }

 if (tmp == *last ){
        DelayUs(100);
        return key_no;
 }

 do{
  count++;
  DelayUs(400);
       if (count > 1300){
              if (count > 8000){
                        *last = key_no;
                     return key_no;
              }
              switch(tmp){
                     case key_down:
                      *last = key_down;
                     return key_down_long;
                     case key_up:
                      *last = key_up;
                     return key_up_long;
                     case key_select:
                      *last = key_select;
                     return key_select_long;
              }
       }

 }while( (GPIO_ReadInputData(GPIOA) & key_no) != key_no);
 if (count < 5){
  return key_no;
 }
 else{
       *last = tmp;
  return tmp;
 }
}
```

```
//*****************************************************************************
```

timer2.c

```
#include "stm8s.h"
#include "timer2.h"
#include "adc.h"
#include <assert.h>
//*****************************************************************************
static uint16_t ch1_pwm_U;  //voltage
static uint16_t ch2_pwm_I;  //current
//*****************************************************************************
void Timer2_Init(void){
//T2 в ШИМ режимі з використанням виходу по 2-х каналах
//Channel1 керує напругою на виході
//Channel2 керує струмом на виході
// Time base configuration
  TIM2_DeInit();
  TIM2_TimeBaseInit(TIM2_PRESCALER_1, timer2_counter);
  ch1_pwm_U = 10;
  ch2_pwm_I = 10;
  /* PWM1 Mode configuration: Channel1 */
  TIM2_OC1Init(TIM2_OCMODE_PWM1, TIM2_OUTPUTSTATE_ENABLE, ch1_pwm_U, TIM2_OCPOLARITY_HIGH);
  TIM2_OC1PreloadConfig(ENABLE);
  /* PWM1 Mode configuration: Channel2 */
  TIM2_OC2Init(TIM2_OCMODE_PWM1, TIM2_OUTPUTSTATE_ENABLE, ch2_pwm_I, TIM2_OCPOLARITY_HIGH);
  TIM2_OC2PreloadConfig(ENABLE);
  TIM2_ARRPreloadConfig(ENABLE);
  /* TIM2 enable counter */
  TIM2_Cmd(ENABLE);
}
//*****************************************************************************
void Timer2_pwm_U_U(float Uset){
       if (Uset > 17)
        Uset = 17;
       ch1_pwm_U = (uint16_t)(Uset * pwmKU);
       TIM2_SetCompare1(ch1_pwm_U);
}
//*****************************************************************************
void Timer2_pwm_I_I(float Iset){

       if (Iset > 4.95)
        Iset = 4.95;
       //set pwm width
       ch2_pwm_I = (uint16_t)(Iset * pwmKI) + 305;
       TIM2_SetCompare2(ch2_pwm_I);
}
//*****************************************************************************
```

timer4.c

```
#include "stm8s.h"
#include "sub.h"
#include "timer4.h"
#include <assert.h>
//*****************************************************************************
#define T4_prescaler TIM4_PRESCALER_16
#define T4_counter   49
//*****************************************************************************

//це відповідає 50мкс
void Timer4_Init(void){
       // Time base configuration
       TIM4_DeInit();
       systic = 0;
       TIM4_TimeBaseInit(T4_prescaler, T4_counter);
       TIM4_ARRPreloadConfig(ENABLE);
```

```
        TIM4_ClearFlag(TIM4_FLAG_UPDATE);
        TIM4_ClearITPendingBit(TIM4_IT_UPDATE);
        TIM4_ITConfig(TIM4_IT_UPDATE, ENABLE);
        TIM4_ClearITPendingBit(TIM4_IT_UPDATE);
        TIM4_Cmd(ENABLE);
}
//*****************************************************************************
```

adc.c

```
#include "stm8s.h"
#include "sub.h"
#include "adc.h"
#include <assert.h>
//*****************************************************************************
uint16_t tmpI16;
uint8_t tmpI8;
//volatile static af_U[5];
//volatile static af_U_c;
//volatile static af_I[5];
//volatile static af_I_c;
//*****************************************************************************
void ADC_Init(void){
 /*
 налаштувати входи для роботи АЦПР
 GPIOB GPIO_PIN_0  - напруга на акб
 GPIOB GPIO_PIN_1  - поточний струм яким заряджається АКБ
 GPIOB GPIO_PIN_2  - температура виходного регулюючого транзистора
 GPIOB GPIO_PIN_3  - поточна напруга на виході регулюючого транзистора
 */
  GPIO_Init(GPIOB, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3,
                                        GPIO_MODE_IN_FL_NO_IT);
  ADC1_DeInit();
  ADC1_Init(ADC1_CONVERSIONMODE_SINGLE,
            ADC1_CHANNEL_0|ADC1_CHANNEL_1|ADC1_CHANNEL_2|ADC1_CHANNEL_3,
            ADC1_PRESSEL_FCPU_D8, ADC1_EXTTRIG_TIM, DISABLE, ADC1_ALIGN_RIGHT,
                                        ADC1_SCHMITTTRIG_ALL, DISABLE);
}
//*****************************************************************************
uint16_t ADC_Get_ADC(uint8_t channel){

uint16_t v1;
ADC1->CSR &= (uint8_t)(~ADC1_CSR_CH);
switch(channel){
  case 0:
    ADC1->CSR |= (uint8_t)(ADC1_CHANNEL_0);
  break;
  case 1:
    ADC1->CSR |= (uint8_t)(ADC1_CHANNEL_1);
  break;
  case 2:
    ADC1->CSR |= (uint8_t)(ADC1_CHANNEL_2);
  break;
  case 3:
    ADC1->CSR |= (uint8_t)(ADC1_CHANNEL_3);
  break;
}
ADC1_ClearITPendingBit(ADC1_IT_EOC);
ADC1_StartConversion();
while((ADC1->CSR & ADC1_IT_EOC) != ADC1_IT_EOC){
}
v1 = ADC1_GetConversionValue();
return v1;
}
//*****************************************************************************
float ADC_Get_I(void){
        tmpI16 = ADC_Get_ADC(1);
```

```
  return ((float)tmpI16 - 513) * 0.036;
}
//*****************************************************************************
float ADC_Get_U(void){
       tmpI16 = ADC_Get_ADC(0);
  return (float)tmpI16 * 0.01697;
}
//*****************************************************************************
float ADC_Get_Uinput(void){
       tmpI16 = ADC_Get_ADC(3);
  return (float)tmpI16 * 0.027514;
}
//*****************************************************************************
void ADC_Get_ADC_String(uint8_t channel, char *buf){
 float tmpF;
  switch(channel){
              case 0:
               tmpF = ADC_Get_U();
              break;
              case 1:
               tmpF = ADC_Get_I();
              break;
              case 2:
               tmpF = 0.00;
              break;
              case 3:
               tmpF = ADC_Get_Uinput();
              break;
       }
       FloatToStr(tmpF, buf, 6);
}
//*****************************************************************************
```