Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «**МЕТОД ТА МІКРОПРОЦЕСОРНІ ЗАСОБИ ВИМІРЮВАННЯ ПАРАМЕТРІВ ЕЛЕКТРОННИХ КОМПОНЕНТІВ**»

Виконав: студент 2 курсу, групи 1КІ-23м
Спеціальності 123 Комп'ютерна інженерія
Мушинський В.Є. _____
Керівник к.т.н., доц. каф. ОТ
Богомолов С.В. _____
"__" _____ 2024р
Опонент д.ф. (PhD), доц. каф. ПЗ
Катєльніков Д. І. _____
"__" _____ 2024р.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.
«____» _____ 2024 р.

ВНТУ 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо-кваліфікаційний рівень магістр
Спеціальність 123 Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри
обчислювальної техніки
проф., д.т.н. О. Д. Азаров
вересня 2024 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Мушинському Вадиму Євгеновичу

Тема роботи «Метод та мікропроцесорні засоби вимірювання параметрів електронних компонентів», керівник роботи Богомолов Сергій Віталійович, к.т.н., доцент, затверджені наказом вищого навчального закладу від 18.09.2024 року № 247.

Строк подання студентом роботи 09.12.2024 р.

3 Вихідні дані до роботи: технічний опис мікропроцесорних платформ Arduino, технічна документація на електронні компоненти.

Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): огляд та аналіз методів та мікропроцесорних засобів вимірювання електронних компонентів, основні параметри, вимірювання параметрів, методи вимірювання, існуючі мікропроцесорні засоби, методи, алгоритми та вимірювання параметрів електронних компонентів, методи визначення типу електронного компонента, алгоритм вимірювання параметрів**Ошибка! Закладка не определена.**, методика вимірювання, проектування апаратної та мікропроцесорної системи та розроблення програмного забезпечення, розробка структурної схеми, вибір електронних компонентів, розробка схеми

електричної принципової, розробка алгоритму роботи, бібліотеки для реалізації коду, лістинг програми, інструкція користувача, висновки.

Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): блок схема алгоритму, схема електрична структурна, схема електрична принципова .

Консультанти розділів роботи представлено в табл. 1.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
	Богомолов С.В., к.т.н., доц. каф. ОТ		
	Небава М.І., к.е.н., проф. каф. ЕП ВМ		

Дата видачі завдання 18.09.2024р.

8 Календарний план наведено в табл. 2.

Таблиця 2 — Календарний план

з/п	Назва етапів виконання магістерської роботи	Строк виконання етапів роботи	Прим-
	Постановка мети та задач роботи	20.10.24	
	огляд та аналіз методів та мікропроцесорних засобів вимірювання електронних компонентів	30.10.24	
	Вибір моделі мікроконтролера	04.11.24	
	Проектування апаратної частини	08.11.24	
	Розробка електричної принципової схеми	19.11.24	
	Проектування програмної частини	24.11.24	
	Розрахунок економічної частини роботи	01.12.24	
	Оформлення пояснювальної записки та ілюстративного матеріалу	10.12.24	
	Аналіз виконання роботи, висновки, додатки		
	Перевірка якості виконання магістерської роботи та усунення недоліків		

Студент _____ Мушинський В.Є.

Керівник роботи _____ Богомолов С.В.

АНОТАЦІЯ

УДК 621.317:004.35:621.3.049.77

Мушинський В.Є. Методи та мікропроцесорні засоби вимірювання параметрів електронних компонентів. Магістерська кваліфікаційна робота зі спеціальності 123 – Комп'ютерна інженерія, освітня програма — Комп'ютерна інженерія. Вінниця: ВНТУ, 2024, 144 стор.

На укр.мові, бібліогр.: 25 назв, рис 46, табл.10.

Магістерська кваліфікаційна робота виконана з метою дослідження та удосконалення методів та мікропроцесорних засобів вимірювання параметрів електронних компонентів. В роботі було досліджено методи вимірювання та розробка приладу для вимірювання параметрів електронних компонентів.

У роботі проведений огляд та аналіз основних параметрів та методів вимірювання, також наведено існуючі мікропроцесорні системи та постановка задачі проектування. Розглянуто та вибрано методи визначення типу електронного компонента, алгоритм вимірювання параметрів, методика вимірювання. Також розроблено структурну схему та схеми електричної принципової. Проведений вибір електронних компонентів для вимірювальної системи. Вибрана методика вимірювання, також розроблений алгоритм роботи приладу. Вибрано бібліотеки для реалізації лістингу програми, також наведено середовище розробки скетчу, а також інструкцію користувача.

Ключові слова: основні параметри, електронні компоненти, мікропроцесорна система, мікропроцесор, Arduino, вимірювання.

ABSTRACT

Mushinsky V. Methods and microprocessor means of measuring parameters of electronic components. Master's qualification work in the specialty 123 - Computer Engineering, educational program - Computer Engineering. Vinnytsia: VNTU, 2024, 145 pp.

In Ukrainian. Bibliography: 25 titles, Fig. 46, Table 10.

The master's qualification work was carried out with the aim of researching and improving methods and microprocessor means of measuring parameters of electronic components. The work investigated measurement methods and developed a device for measuring parameters of electronic components.

The work reviews and analyzes the main parameters and measurement methods, also presents existing microprocessor systems and sets the design task. Methods for determining the type of electronic component, the algorithm for measuring parameters, and the measurement method are considered and selected. A structural diagram and schematic diagrams of the electrical circuit are also developed. Electronic components for the measuring system are selected. The measurement method is selected, and the device operation algorithm is also developed. Libraries are selected for the implementation of the program listing, the sketch development environment is also provided, as well as the user manual.

Keywords: basic parameters, electronic components, microprocessor system, microprocessor, Arduino, measurements.

ЗМІСТ

ВСТУП

У сучасному світі електроніка є однією з найпотужніших галузей науки і техніки, яка має застосування в різних сферах - від мікроелектроніки до автомобільної та медичної техніки. Від правильного визначення їх параметрів залежить надійність і якість роботи електронних компонентів, таких як конденсатори, резистори та інші. Це вимагає використання сучасних методів вимірювання та мікропроцесорних засобів, які забезпечують високу точність і швидкість аналізу.

Метою роботи є дослідження методів і мікропроцесорних пристроїв вимірювання параметрів електронних компонентів, а також розробка мікропроцесорної вимірювальної системи. Особлива увага приділяється огляду таких параметрів, як ємність, еквівалентний послідовний опір (ESR), напруга та термічна стабільність, їх методи вимірювання та вплив помилок і шуму на результати.

Актуальність теми обумовлена підвищеними вимогами до продуктивності та надійності електронних пристроїв, що залежить від точного визначення параметрів компонентів. Удосконалення методів вимірювання та розвиток мікропроцесорів сприяють підвищенню ефективності проектування електронних систем.

Об’єктом — це процеси вимірювання параметрів електронних компонентів.

Предметом є методи та мікропроцесорні пристрої для вимірювання параметрів електронних компонентів.

Завдання роботи:

— аналіз основних параметрів електронних компонентів, таких як ємність, напруга, ESR та оцінка їх впливу на роботу електронних систем;

— огляд методів вимірювання параметрів компонентів та їх придатність для різних застосувань;

— вивчення сучасних мікропроцесорних вимірювальних приладів та їх характеристик;

— створення структурної схеми мікропроцесорної вимірювальної системи, підбір компонентів і створення електричних схем;

— створення алгоритму роботи та програмного забезпечення;

— дослідження методів вимірювання параметрів та підготовка інструкцій для користувачів.

Наукова новизна:

— Розроблено нові методи вимірювання параметрів електронних компонентів з використанням мікропроцесорних засобів, що підвищують точність і швидкодію вимірювань.

— Запропоновано підходи для автоматичного вимірювання як активних, так і пасивних електронних компонентів.

Практична цінність:

— Результати можуть бути використані для створення автоматизованих приладів контролю параметрів електронних компонентів.

— Запропоновані методи забезпечують підвищення ефективності процесів тестування та діагностики електронних компонентів.

Апробація

Результатів наукової роботи було проведено на науковій конференції «Молодь в науці: дослідження, проблеми, перспективи (МН–2025)», доповідь на тему: «Метод та мікропроцесорні засоби вимірювання параметрів електронних компонентів» Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/22203

1 ОГЛЯД ТА АНАЛІЗ МЕТОДІВ ТА МІКРОПРОЦЕСОРНИХ ЗАСОБІВ ВИМІРЮВАННЯ ЕЛЕКТРОНИХ КОМПОНЕНТІВ

1.1 Основні параметри

Резистор (від лат. resisto — «опираюся»), — це пасивний елемент електричного кола, призначений для забезпечення певного електричного опору. Головною характеристикою резистора є його електричний опір. Вони є найпоширенішими пасивними елементами електронної апаратури та виконують функції навантаження, споживача, дільника напруги в колах живлення, а також використовуються як елементи фільтрів, шунти й елементи для формування імпульсів. У разі лінійної характеристики сила струму, що проходить через резистор, залежить від напруги і описується законом Ома. Закон Ома формулюється наступним чином: сила струму в провіднику прямо пропорційна прикладеній до нього напрузі та обернено пропорційна його опору.

Математично закон Ома записується як (1.1):

$$I = U/R, \tag{1.1}$$

де I — сила струму (в амперах),

U — напруга (у вольтах),

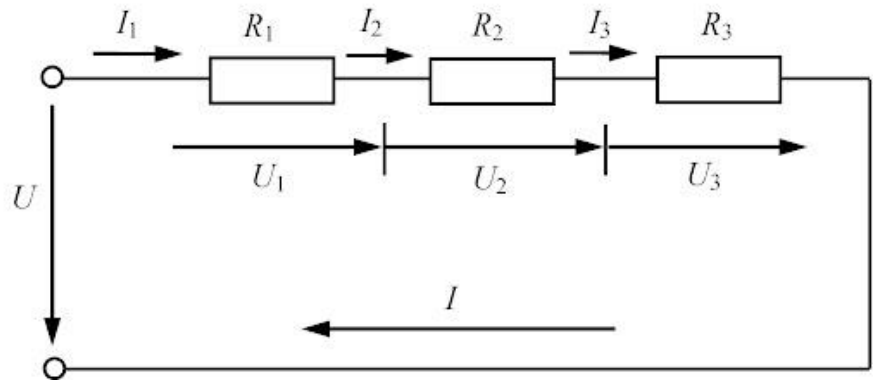
R — опір (в омах).

Резистори характеризуються номінальним опором (від часток Ома до 1000 ГОм), допустимим відхиленням від номіналу (0,001–20%), максимальною потужністю розсіювання (від сотих часток Вт до кількох сотень Вт), граничною напругою і температурним коефіцієнтом опору.

Залежно від призначення, резистори поділяють на дві групи: загального призначення та спеціального призначення. До спеціальних належать високоомні, високовольтні, високочастотні та прецизійні резистори.

Також існують основні види з'єднань резисторів використовуються для досягнення різних електричних характеристик у схемах і включають послідовне, паралельне та змішане з'єднання.

Послідовне з'єднання резисторів зображено на рисунку 1.1

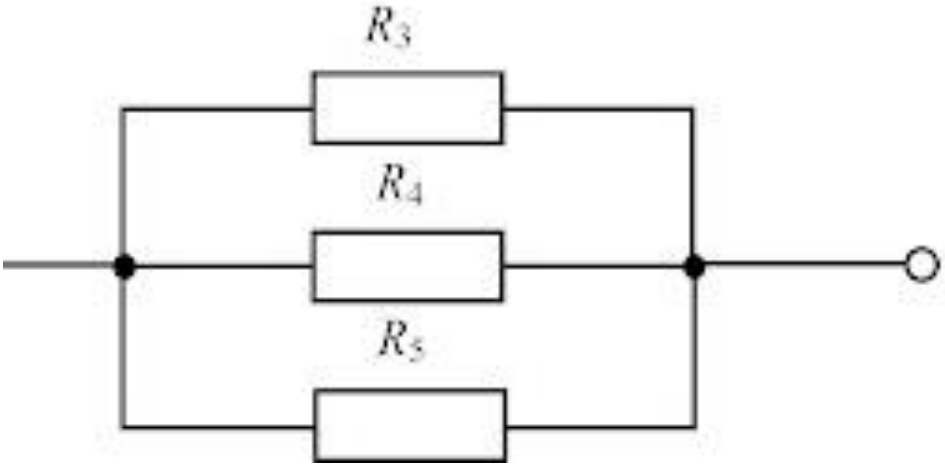


Рисунку 1.1 — Послідовне з’єднання резисторів

При послідовному з'єднанні резисторів загальний опір еквівалентної схеми дорівнює сумі опорів усіх резисторів. Обчислюється наступною формулою (1.2):

$$R = R_1+R_2+\ldots+R_n =\sum R_i, \quad (1.2)$$

Паралельне з'єднання резисторів зображено на рисунку 1.2

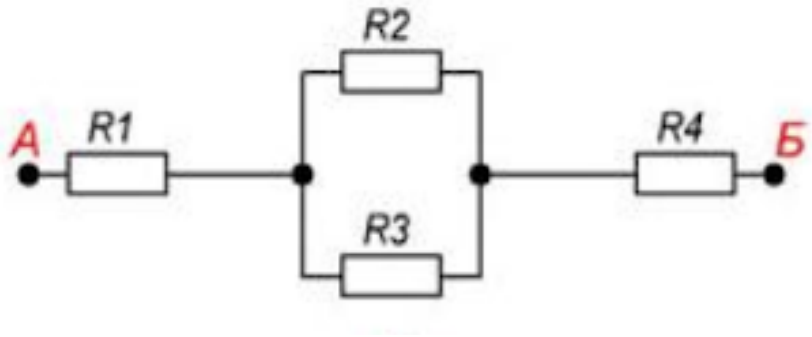


Рисунку 1.2 — Паралельне з’єднання резисторів

Для паралельного з'єднання резисторів загальна провідність (обернена величина еквівалентного опору) дорівнює сумі провідностей усіх резисторів. Зображено в формулі (1.3)

$$1/R = 1/R_1 + 1/R_2 + \ldots. + 1/R_n, \quad (1.3)$$

Змішане з'єднання резисторів зображено на рисунку 1.3



Рисунку 1.3 — Змішане з’єднання резисторів

У схемі зі змішаним з'єднанням резистори розташовані в два паралельні блоки. Перший блок складається з послідовно з'єднаних резисторів R1 та R2 із загальним опором R1 + R2, а другий блок — з резистора R3. Загальна провідність для такої схеми обчислюється за формулою:

$$1/R = (1/(R_1 + R_2)) + 1/R_3, \quad (1.4)$$

Загальний опір схеми можна визначити за рівнянням:

$$R = (R_3(R_1+R_2))/R_1 + R_2 + R_3, \quad (1.5)$$

Резистори мають надзвичайно широке застосування і є одними з найпоширеніших компонентів електронних схем. Їхня основна функція —

обмеження сили струму, але вони також часто використовуються для розподілу напруги. У схемах резистори застосовують для регулювання сили струму на інших елементах, демпфування коливань у фільтрах тощо.

В електронних схемах резистори виконують важливі функції, зокрема забезпечують зміщення робочої точки транзисторних каскадів через подільники напруги або струму, здійснюють сумування струмів і напруг, узгодження вхідного і вихідного опору каскадів. Також, у поєднанні з реактивними елементами (L і C), вони утворюють фільтрувальні, блокувальні та корекційні ланки для частотних і перехідних характеристик. Найбільш поширеними є подільники і регулятори напруги. Сфера використання резисторів щороку розширюється: від низьковольтних портативних пристроїв до високовольтних промислових установок.

Резистори можна знайти в побутовій техніці, медичному та вимірювальному обладнанні, системах автоматизації, блоках живлення, високочастотних лініях, хвилеводах, робототехніці, автомобільній техніці, теле- і радіоапаратурі, відеообладнанні тощо.

Конденсатор (від англ. capacitor) — це система з двох або більше електродів (обкладок), розділених діелектриком, товщина якого менша за розмір обкладок. Така система має електричну ємність і здатна зберігати заряд. Конденсатор є пасивним компонентом електронних пристроїв і часто використовується в схемах для блокування постійного струму при одночасному пропусканні змінного.

При подачі напруги на обкладки конденсатора відбувається накопичення електричного заряду. Після відключення від джерела напруги заряд залишається на обкладках завдяки електростатичним силам. Якщо конденсатор не заряджений, заряди на обох обкладках мають однакову величину, але протилежні за знаком. Ємність конденсатора характеризує його здатність зберігати електричний заряд. Ємність конденсатора визначається за формулою:

$$C = Q/U, \quad (1.6)$$

де C — ємність конденсатора у фарадах;

Q — електричний заряд, що накопичений на одній з обкладок в кулонах;

U — електрична напруга між обкладками у вольтах.

Ємність конденсатора вимірюється у фарадах, однак ця одиниця є досить великою, тому в практичному використанні частіше застосовують її підодиниці. Зазвичай ємність конденсаторів виражається в пікофарадах (пФ), нанофарадах (нФ), мікрофарадах (мкФ) або міліфарадах (мФ), що відповідно менші за фарад у 10^{12}, 10^{9}, 10^{6} і 10^{3} разів.

Основною характеристикою конденсатора є його електрична ємність, яка визначає, скільки електричного заряду він може накопичити. Номінальна ємність конденсатора зазначається у фарадах (F) або її підодиницях, таких як пікофаради (pF), нанофаради (nF) та мікрофаради (μF).

Значення ємності можуть варіюватися від дуже малих (пікофаради) до сотень мікрофарад. У більшості електронних пристроїв та схем застосовуються конденсатори з ємністю в діапазоні від пікофарадів до мікрофарадів. Однак, для деяких спеціальних застосувань використовуються конденсатори з ємністю у десятки фарад, такі як суперконденсатори або електролітичні конденсатори високої ємності.

Ємність, зазначена на конденсаторі, відома як номінальна. Реальна ємність може дещо відрізнятися від номінальної через допустиме відхилення, яке визначає клас точності конденсатора. Існує 11 класів точності, що визначаються допустимим відхиленням ємності (див. таблицю 1.1).

Справді, для досягнення більшої ємності конденсатори можна з’єднувати паралельно. При паралельному з'єднанні загальна ємність групи конденсаторів дорівнює сумі ємностей усіх конденсаторів у цій групі (рисунок 1.4).

Таблиця 1.1—Клас точності конденсаторів і допустиме відхилення ємності

Клас точності						I	II	III	IV		VI
Відхилення, %	± 0,01	± 0,2	± 0,5	± 1	± 2	± 5	± 10	± 20	-10 + 20	-20 +30	-20 +50

Це пояснюється принципом накопичення заряду: у паралельному з'єднанні заряди на кожному конденсаторі накопичуються окремо. Отже, загальний заряд групи є сумою зарядів на кожному з конденсаторів, що в підсумку забезпечує сумарну ємність, рівну сумі ємностей усіх конденсаторів у паралельному з'єднанні.

$$1/C = 1/C_1 + 1/C_2 \ldots + 1/C_n, \qquad (1.7)$$

Загальну ємність паралельно з’єднаних конденсаторів можна розрахувати за формулою (1.7). Наприклад, якщо з’єднати два конденсатори з ємністю 10 мкФ та 5 мкФ, загальна ємність становитиме 15 мкФ (10 мкФ + 5 мкФ). Цей метод можна використовувати для з’єднання більшої кількості конденсаторів, що дає змогу досягати значних ємностей для особливих потреб (рис.1.4).

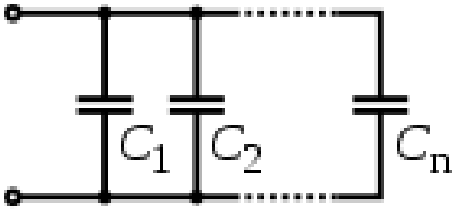


Рисунок 1.4 — Паралельно з’єднанні конденсатори

При послідовному з'єднанні конденсаторів заряди на них будуть рівними, оскільки струм, що проходить через кожен конденсатор, залишається однаковим. Загальна ємність групи послідовно з’єднаних конденсаторів визначається іншим підходом: тут напруга розподіляється по кожному конденсатору відповідно до його ємності. Для визначення загальної ємності групи в послідовному з’єднанні використовується обернене значення суми обернених значень ємностей кожного конденсатора.

$$C = C_1 + C_2 + \ldots + C_n, \tag{1.8}$$

Послідовно з'єднані конденсатори розраховуються за рівнянням (1.8). Послідовне з’єднання конденсаторів зображено на рисунку 1.5

Рисунок 1.5 — Послідовно з’єднанні конденсатори

При послідовному з’єднанні загальна ємність конденсаторів завжди буде меншою, ніж найменша ємність серед окремих конденсаторів у групі. Важливо також відзначити, що таке з'єднання знижує ризик пробою: кожен конденсатор витримує лише частину загальної напруги, тому на окремі конденсатори можна подавати менші напруги, що підвищує надійність всієї системи.

Таким чином, послідовне з’єднання конденсаторів має свої переваги та обмеження, які варто враховувати під час проєктування конденсаторних груп.

Основні характеристики конденсаторів є робоча напруга визначає максимальну напругу, яку конденсатор може витримати без пошкоджень, і зазвичай позначається у вольтах (В). Конденсатор повинен мати достатню

напругостійкість для уникнення пробою або пошкодження. Ця характеристика може вказуватися як максимальна постійна, змінна напруга або пробивна напруга. Діапазон робочих температур температурний інтервал, у якому конденсатор зберігає ефективність. Вихід за межі цього діапазону може призвести до зміни або погіршення характеристик, наприклад, для електролітичних конденсаторів через властивості електроліту. Тангенс кута втрат (tan δ) показник ефективності конденсатора, що відображає його здатність зберігати енергію. Нижчий тангенс кута втрат означає менші втрати енергії, що є важливим для застосувань із високими вимогами до ефективності, як у силовій електроніці Довговічність тривалість, протягом якої конденсатор зберігає свої характеристики. Деякі конденсатори, наприклад, електролітичні, мають обмежений термін служби через старіння електроліту. Довговічність може виражатися у "годинах роботи" або "циклах роботи", що визначає період експлуатації до заміни або втрати ефективності.

Ці параметри можуть значно варіюватися залежно від типу конденсатора (наприклад, керамічний, електролітичний, плівковий), розміру та призначення. Правильний вибір конденсатора з урахуванням цих характеристик забезпечує відповідність вимогам системи, оптимізуючи її ефективність та надійність.

1.2 Вимірювання параметрів

В електронних пристроях використовуються конденсатори різних типів і призначень, зі значеннями ємності від приблизно 1 пФ до 1000 мкФ. На високих та надвисоких частотах вимірюванню можуть підлягати й малі міжелектродні ємності електронних компонентів, а також паразитні ємності між елементами схеми (ємності монтажу).

Допустима похибка вимірювання ємностей залежить від їх застосування. Конденсатори, що є частиною коливальних контурів, повинні мати визначену ємність з точністю не менше 1%. У випадку блокувальних, розділових або

зв'язкових конденсаторів допустима значна (до 20-50%) варіація ємності, тому їх можна вимірювати більш простими методами (рис. 1.6).

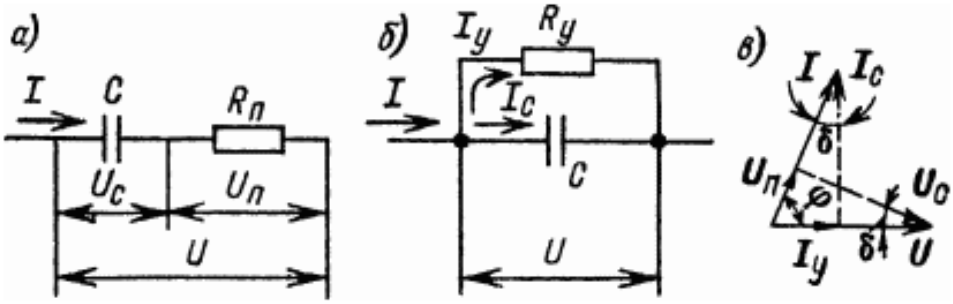


Рисунок 1.6 — Еквівалентні схеми (а, б) та векторна діаграма (в) ланцюга з конденсатором

У кожному конденсаторі, підключеному до електричного кола, виникають енергетичні втрати, які здебільшого зумовлені властивостями діелектрика, а також недосконалістю ізоляції між виводами. З урахуванням цих втрат еквівалентна схема конденсатора може бути подана у двох варіантах: як ємність С, з'єднана послідовно з опором втрат Rp (рис. 1.6, а), або як та ж ємність С, шунтована опором витоку Ru (рис. 1.6, б). Для переходу від однієї еквівалентної схеми до іншої та перерахунку активного опору застосовують формулу:

$$R_y = 1/((2*\pi*f*C)^2 * R_n), \quad (1.9)$$

де f - частота струму в ланцюзі конденсатора.

З векторної діаграми на рис. 1.6, в, яка підходить для обох варіантів еквівалентних схем, видно, що через наявність втрат у колі з конденсатором фазовий зсув між струмом I і напругою U завжди менший за 90°. Втрати в конденсаторі зазвичай характеризуються кутом втрат δ=90°−ϕ, який визначається згідно з позначеннями на рис. 1.6 за такою формулою:

$$tg\delta = 1/(2*\pi*f*C*R_y), \quad (1.10)$$

Втрати в конденсаторі інколи виражаються коефіцієнтом потужності cos або струмом витоку I_y, визначеним за стандартних умов. Для більшості конденсаторів ці втрати дуже малі ($tg\delta<0,001$), тому їх можна вважати.

$$tg\delta = \sin(90^{\circ}-\varphi)= \cos\varphi \quad (1.11)$$

Електронні пристрої використовують різні типи конденсаторів, серед яких найбільші втрати енергії характерні для електролітичних і паперових конденсаторів, обмежених зазвичай до низькочастотних застосувань. У деяких методах вимірювання втрати в конденсаторі визначаються одночасно з вимірюванням його ємності. Важливо врахувати, що з підвищенням частоти втрати можуть значно зростати (що відповідає зростанню Rп і зменшенню Ry, тоді як ємність при цьому майже не змінюється. На дуже високих частотах можливе помітне зростання ефективної ємності конденсатора через індуктивність його обкладок і підвідних провідників.

Параметри конденсатора (C, Rн, Ry, δ) залежать від умов експлуатації, таких як температура, вологість, атмосферний тиск і напруга. Для критичних застосувань випробування конденсаторів проводять не тільки на робочих частотах, а й в умовах, наближених до експлуатаційних. Найпростіші перевірки конденсаторів можливі без спеціальних приладів: за допомогою омметра чи пробника легко визначити коротке замикання або пробій між обкладками (варто пам'ятати, що пробій може проявлятися лише при високій напрузі, близькій до робочої). Відсутність обриву в неелектролітичних конденсаторах ємністю від 0,01 мкФ можна перевірити, включивши конденсатор у ланцюг змінного струму, наприклад, послідовно з лампою чи гучномовцем. Нормальне або слабке свічення свідчить про відсутність обриву.

Конденсатор із великим опором витоку може зберігати заряд тривалий час; це дозволяє оцінити якість конденсаторів ємністю понад 0,01 мкФ.

Підключення омметра до конденсатора призведе до короткочасного відхилення стрілки через зарядний струм, після чого при великому опорі витоку вона повернеться до початкового положення. Якщо кожне повторне підключення омметра викликає відхилення стрілки, це свідчить про малий опір витоку. Для перевірки витоку конденсаторів ємністю понад 100 пФ можна використовувати головні телефони, з'єднані послідовно з батареєю. При малому опорі витоку підключення викличе клацання у телефонах, тоді як хороший конденсатор викличе клацання тільки при першому підключенні. Виміряти опір витоку можна індукторними або електронними мегомметрами.

Електролітичні конденсатори слід підключати з урахуванням полярності. Вимірювання опору витоку рекомендується проводити через 10 хвилин після подачі напруги, коли заряд завершено. Для вимірювання параметрів конденсаторів застосовують методи вольтметра-амперметра, прямого вимірювання за допомогою мікрофарадметрів, порівняння (заміщення), мостові та резонансні методи. Напруга, що подається на конденсатор під час випробувань, не повинна перевищувати максимально допустиму робочу. Якщо під час випробувань конденсатор зарядився до високої напруги, після завершення тесту його потрібно розрядити, наприклад, через кнопку, з'єднану паралельно конденсатору.

Резистори описуються номінальним значенням електричного опору (від кількох Ом до 1000 ГОм), допустимим відхиленням (від 0,001% до 20%), максимальною потужністю, яку вони можуть розсіювати (від часток Вт до кількох сотень Вт), граничною електричною напругою та температурним коефіцієнтом електричного опору.

Резистори класифікуються за видом резистивного матеріалу на такі категорії:

— дротяні резистори — це відрізок дроту з високим питомим опором, намотаний на неметалевий каркас. Вони можуть мати значну паразитну індуктивність;

— плівкові металеві резистори — представляють собою тонку металеву плівку з високим питомим опором, нанесену на керамічне осердя, на кінцях якого встановлені металеві ковпачки з дротяними виведеннями. Це найбільш поширений тип резисторів;

— металофольгові резистори — використовують тонку металеву стрічку як резистивний матеріал;

— вугільні резистори — можуть бути плівковими або об'ємними, і в них застосовується графіт з високим питомим опором;

— напівпровідникові резистори — використовують опір слабко легованого напівпровідника, що може призводити до значної нелінійності вольт-амперної характеристики. Вони переважно використовуються в інтегральних мікросхемах, де інші типи резисторів складно застосувати.

За характером зміни опору резистори поділяються на:

— резистори сталого опору;

— регульовані резистори змінного опору (потенціометри);

— підналагоджувані резистори змінного опору.

За способом монтажу резистори можуть бути:

— для навісного монтажу (з дротяними виводами);

— для поверхневого монтажу (SMD — Surface Mount Device);

— комбінації резисторів в одному загальному блоці, зазвичай у мініатюрному виконанні (збірки, мікромодулі, матриці, мікросхеми).

За типом вольт-амперної характеристики резистори діляться на:

— лінійні резистори;

— нелінійні (напівпровідникові) резистори:

— варистори — опір змінюється залежно від прикладеної напруги;

— терморезистори — опір залежить від температури;

— фоторезистори — опір змінюється в залежності від освітленості;

— тензорезистори — опір залежить від деформації резистора;

— магніторезистори — опір змінюється в залежності від величини магнітного поля.

Останнім часом постійні резистори все частіше маркують кольоровим кодом. Маркування наносять на циліндричну поверхню резистора у вигляді точок або кругових смуг (поясків). Воно вказує на номінальний опір резистора та допустиме відхилення його опору від номінального значення. Номінальний опір виражається в омах двома або трьома цифрами (у випадку трьох цифр остання не дорівнює нулю) і множником 10^n, де n — будь-яке ціле число від мінус 2 до плюс 9.

Для резисторів з номінальним опором, вираженим двома цифрами та множником, кольорова маркування складається з чотирьох знаків або трьох, якщо допустиме відхилення становить +20% (таке відхилення не позначається маркуванням).

Маркувальні знаки зсуваються до одного з торців резистора. Першим вважається знак, розташований ближче до торця. Якщо довжина резистора не дозволяє зсунути маркування до одного з торців, останній знак роблять у 1,5 рази більшим за інші. Маркувальні знаки розташовують на резисторі зліва направо в наступному порядку: перший знак — перша цифра, другий знак — друга цифра, третій знак — множник. Це є номінальне опір. Четвертий знак вказує на допустиме відхилення опору.

Для резисторів з номінальним опором, вираженим трьома цифрами та множником, кольорова маркування складається з п'яти знаків:

— перші три знаки — це три цифри номіналу;

— четвертий знак — множник;

— п'ятий знак — допустиме відхилення опору.

Кольори маркувальних знаків та відповідні їм числа номіналу і допуска наведені на рисунку.

Перші дві цифри представляють значення в омах, а остання — кількість нулів. Це маркування використовується для номінального ряду E24 з допуском 1% та 5% для типорозмірів 0603, 0805 та 1206. Буква R виконує роль десяткової коми.

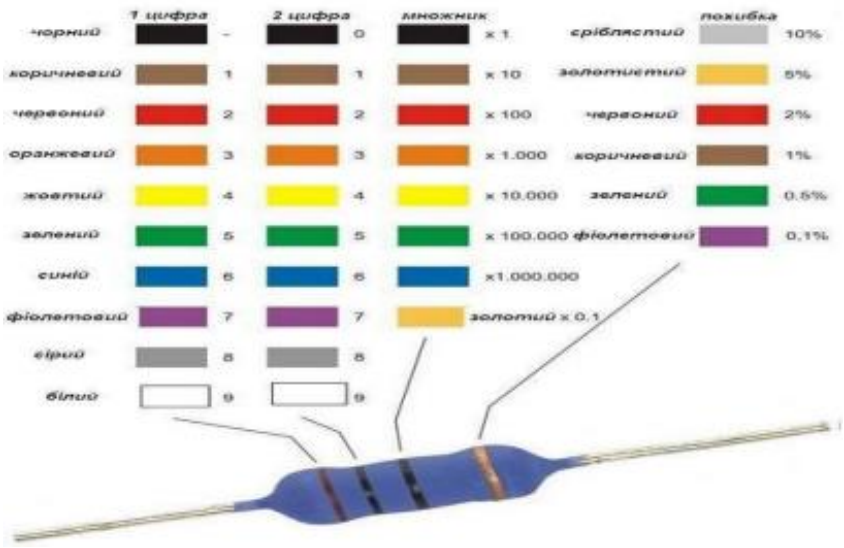


Рисунок 1.7 — Кольорове маркування резистора та відповідні кольори

Приклади:

— 220 = 22 x 100 = 22Ω;

— 471 = 47 x 101 = 470Ω;

— 102 = 10 x 102 = 1000Ω;

— 3R3 = 3.3Ω;

Перші три цифри вказують на значення в омах, а остання — на кількість нулів. Це маркування застосовується до резисторів з ряду номіналів E96 з допуском 1% для типорозмірів 0805 та 1206. Буква R також вказує на десяткову кому. Приклади:

— 4700 = 470 x 100 = 470Ω;

— 2001 = 200 x 101 = 2000Ω;

— 1002 = 100 x 102 = 10000Ω;

— 15R0 = 15.0Ω;

У випадку п'ятисмугового маркування перші три смуги відповідають опору, четверта — множнику, а п'ята — допуску. Якщо на резисторі лише три смуги, це означає, що його допуск становить 20%, і всі смуги вказують лише на опір. Шоста смуга, якщо вона присутня, вказує на температурний коефіцієнт опору (ТКС).

1.3 Методи вимірювання

У сучасній електроніці точність вимірювання параметрів таких компонентів, як резистори і конденсатори, дуже важлива для забезпечення надійної роботи електричних кіл. У цьому параграфі ми розглянемо методи вимірювання опору резисторів і ємності конденсаторів. Кожен із цих методів має свої особливості та переваги, які впливають на вибір підходу в залежності від умов і необхідної точності.

Існує кілька способів вимірювання ємності конденсаторів, серед яких:

— метод вольтметра-амперметра;

— метод безпосередньої оцінки;

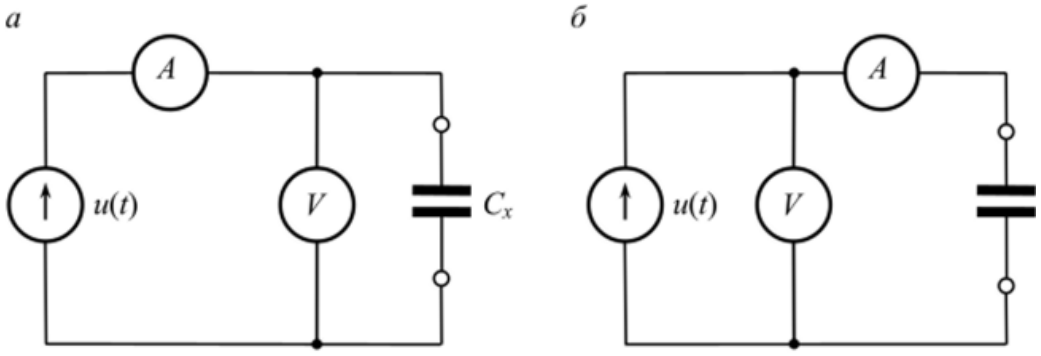
— мостовий метод;

— резонансний метод;

— метод дискретного відліку.

Метод вольтметра-амперметра ілюструється на рисунку 1.8, де представлені схеми вимірювання ємності за допомогою вольтметра (а) і амперметра (б).

Суть цього методу полягає у використанні показників приладів, які вимірюють змінний струм (амперметр) і напругу (вольтметр), для обчислення точного значення ємності конденсатора Cx, підключеного до вимірювальної схеми.



а - методом вольтметра б – методом амперметра

Рисунок 1.8 — Схеми вимірювання ємності

$$C_x = 1/(2\pi C_x), \quad (1.12)$$

де f - частота сигналу,

Cx - ємність конденсатора.

Схема, представлена на рисунку 1.3а, призначена для вимірювання ємностей, у яких опір (Xз) значно менший за вхідний опір вольтметра (Xз << Rv). Цей метод зазвичай використовується для визначення великих ємностей. У свою чергу, схема на рисунку 1.3б підходить для вимірювання менших ємностей, у яких опір (Xз) значно більший за опір амперметра (Xз >> RA). Опір у цьому випадку розраховується з урахуванням частоти сигналу відповідно до формули (1.4). Цей метод дозволяє вимірювати ємності в діапазоні від 1000 пікофарад до 100 мікрофарад. Таким чином, ці схеми вимірювання ємності надають можливість точно визначати значення ємності конденсаторів залежно від їх опору та характеристик вимірювальних приладів.

Метод безпосередньої оцінки є одним з найбільш простих і поширених способів вимірювання ємності конденсаторів. Він базується на використанні відомого резистора та вимірюванні часу зарядки або розрядки конденсатора через цей резистор.

Основні етапи методу безпосередньої оцінки для вимірювання ємності конденсатора включають:

— підготовка;

— зарядка конденсатора;

— розрядка конденсатора;

— вимірювання часу;

— обчислення ємності.

Процес починається з підключення резистора до конденсатора. Значення резистора має бути відомим і обиратися так, щоб зарядка або розрядка конденсатора тривала від кількох секунд до кількох хвилин. Важливо, щоб резистор мав достатньо великий опір, щоб уникнути надмірних струмів, але не був занадто великим, щоб не сповільнювати процес зарядки або розрядки.

Далі необхідно підключити джерело живлення до конденсатора, щоб зарядити його до заданого рівня напруги. Це можна здійснити, підключивши позитивний полюс джерела живлення до однієї з пластин конденсатора, а негативний полюс - до іншої. Після цього слід зафіксувати час зарядки.

Після зарядки джерело живлення відключається, і обмежувальний резистор підключається до конденсатора. Важливо запам'ятати початковий час розрядки.

Запустіть таймер або скористайтеся іншим способом вимірювання часу. Зафіксуйте час, необхідний для зміни напруги на конденсаторі від початкового значення до певної відсоткової частки від нього (наприклад, 63,2% для однієї часової константи конденсатора).

Використовуючи виміряні значення часу зарядки та розрядки, а також відоме значення резистора, можна обчислити ємність конденсатора за допомогою відповідних формул. Наприклад, для зарядки конденсатора можна використовувати формулу (1.13).

Даний метод є досить простим і зручним для вимірювання ємності конденсатора, особливо в умовах, коли доступ до більш точних вимірювальних пристроїв обмежений. Однак слід враховувати, що точність цього методу може бути знижена через наявність паразитних опорів і ємностей у схемі, які можуть впливати на час зарядки або розрядки конденсатора.

$$C = t/R, \tag{1.13}$$

де C — ємність,

t — час зарядки,

R — значення резистора.

Для розрядки використовується формула:

$$C = (t/R)/\ln(2) \tag{1.14}$$

де ln — натуральний логарифм.

Мостовий метод є ще одним популярним способом вимірювання ємності конденсаторів. Він базується на порівнянні значень ємностей шляхом балансування мостової схеми або вимірювання різниці фаз між різними гілками моста. Цей підхід дозволяє досягти високої точності в вимірюваннях.

Основні етапи мостового методу для вимірювання ємності конденсатора включають:

— підготовка мостової схеми;

— початкове налаштування;

— зміна ємності;

— балансування моста;

— вимірювання ємності.

Спочатку потрібно побудувати мостову схему, яка включає резистори і вимірювальні прилади (наприклад, вимірювальний міст або цифровий мультиметр) для вимірювання різниці напруг або фаз між гілками моста. Одна з гілок повинна містити конденсатор, ємність якого потрібно виміряти.

Наступним кроком є початкове налаштування мостової схеми, що передбачає встановлення початкових значень резисторів або фазових зсувів так, щоб досягти нульової різниці напруг або фаз між гілками моста.

Після цього змініть ємність конденсатора, який вимірюється, наприклад, шляхом переключення на інший конденсатор або регулювання його ємності за допомогою змінного конденсатора або потенціометра.

Далі виконайте процедуру балансування моста, змінюючи значення резисторів або фазових зсувів до досягнення нульової різниці напруг або фаз між гілками. Це свідчить про те, що міст збалансовано, і тепер можна визначити значення ємності конденсатора.

Після балансування моста виміряйте значення резисторів або фазових зсувів, які були налаштовані для досягнення балансу. Використовуючи відповідні формули або критерії балансу моста, можна обчислити ємність

конденсатора. Різні формули можуть бути використані в залежності від конкретного типу мостової схеми.

Наприклад, у стандартних схемах, за умови, що міст у балансі (тобто напруги в двох гілках моста рівні), можна застосувати таку формулу для визначення ємності конденсатора (1.15):

$$C = (R_1 * R_3)/ R_2, \quad (1.15)$$

де R1, R2 і R3 - значення резисторів у мостовій схемі.

У цій схемі коригують фазові зсуви для досягнення балансу моста. Коли міст знаходиться в стані балансу, що свідчить про рівність фаз у двох гілках, можна застосувати наступну формулу для обчислення ємності конденсатора (1.16).

Наведенні формули є загальними прикладами, і конкретні рівняння можуть змінюватися в залежності від типу мостової схеми та використовуваного вимірювального обладнання. При застосуванні мостового методу важливо дотримуватися інструкцій і рекомендацій виробника пристрою для коректного налаштування та вимірювання ємності конденсатора.

$$C = (1/(2\pi fR))*(R_1/R_2), \quad (1.16)$$

де f — частота сигналу, що використовується в мостовій схемі,

R — значення резистора у гілці моста, підключеному до конденсатора,

R1 , R2 — значення резисторів у гілках моста, підключених до вимірювальних приладів.

Мостовий метод забезпечує високу точність вимірювання ємності конденсаторів і може бути використаний для визначення як малих, так і великих значень ємності. Точність вимірювання може залежати від якості

компонентів мостової схеми та точності використовуваних вимірювальних приладів.

Резонансний метод вимірювання ємності конденсатора ґрунтується на вимірюванні резонансної частоти LC-кола, де L — індуктивність, а C — ємність конденсатора. Цей метод використовує принцип резонансу, коли імпеданс кола досягає мінімального значення, а амплітуда напруги або струму досягає свого максимального рівня.

Основні етапи резонансного методу для вимірювання ємності конденсатора включають:

— підготовка резонансного кола;

— налаштування генератора сигналу;

— виявлення резонансної частоти;

— обчислення ємності.

Спочатку необхідно побудувати LC-коло, що складається з індуктивності L та конденсатора C, підключених послідовно. Індуктивність може бути реалізована, наприклад, у вигляді котушки або обмотки.

Далі потрібно підключити генератор сигналу до кола і налаштувати його на пошук резонансної частоти. Це можна зробити, поступово змінюючи частоту генератора та спостерігаючи за зміною амплітуди струму або напруги в колі.

Під час налаштування генератора сигналу слід визначити частоту, при якій амплітуда струму або напруги в колі досягає максимального значення. Це і буде резонансною частотою LC-кола.

Знаючи індуктивність (L) та резонансну частоту (f), ємність конденсатора можна обчислити за допомогою наступної формули:

$$C = 1/(4\pi^2 f^2 L), \quad (1.17)$$

де π — математична константа,

f — резонансна частота,

L — індуктивність.

Цей метод забезпечує високоточне і зручне вимірювання ємності конденсатора, але точність результатів залежить від якості компонентів схеми і вимірювальних приладів. Метод дискретного відліку (або метод зарядки-відліку) є одним із способів вимірювання ємності конденсаторів, що базується на вимірюванні часу, необхідного для заряджання або розряджання конденсатора через відомий резистор.

Основні етапи методу дискретного відліку для вимірювання ємності конденсатора:

— підготовка кола;

— зарядка конденсатора;

— вимірювання часу;

— розрахунок ємності.

Для початку конденсатор підключають до відомого резистора, утворюючи RC-коло. Це коло може заряджатися або розряджатися через перемикач або керуючий елемент.

Після замикання перемикача або зміни стану розпочинається процес заряджання конденсатора через резистор. Фіксується час, необхідний для досягнення напруги на конденсаторі певного значення (наприклад, 63,2% від максимальної напруги).

Записують час зарядки конденсатора до цього рівня напруги.

Ємність конденсатора C розраховують за формулою з використанням відомого значення опору R і виміряного часу t:

$$C = t/(R*\ln 1-1/e), \quad (1.18)$$

де e — математична константа,

ln — натуральний логарифм.

Метод дискретного відліку дозволяє вимірювати ємність конденсатора без необхідності складної апаратури. Водночас необхідно враховувати опір проводів і внутрішній опір джерела живлення, оскільки ці фактори можуть вплинути на точність вимірювань.

Існує декілька методів визначення опору резистора, такі як:

— мостовий;

— амперметра і вольтметра;

— омметра;

Існують різні методи для вимірювання опору провідників. Один із способів — використовувати амперметр і вольтметр. Якщо I — сила струму в амперах, що показує амперметр, а U — напруга в вольтах на кінцях провідника, тоді опір провідника в омах можна обчислити за формулою R=U/I. Ще простіший метод — це безпосереднє вимірювання омметром.

Для точнішого вимірювання опорів використовують метод порівняння з еталонними значеннями опорів (мостовий метод). Мостова схема (міст Уітстона) складається з чотирьох опорів R1, R2, Rx, і R, з’єднаних у формі чотирикутника. Точки A, B, C, і D на рисунку 1.9 є вершинами цього чотирикутника, а протилежні вершини з'єднані діагоналями. Опори R1, R2, Rx, і R, розміщені між сусідніми вершинами, утворюють плечі містка.

Назва "мостове коло" пояснюється тим, що діагоналі, подібно до мостів, з'єднують протилежні вершини. В одну з діагоналей включено джерело живлення ε, а в іншу — гальванометр G. Гальванометр — це високочутливий електровимірювальний прилад для визначення малих струмів, напруг та зарядів. Шкала гальванометра має нуль посередині, що дає змогу фіксувати струми в протилежних напрямках.

Опір, що вимірюється (Rx), розміщують у ділянці AB, а в ділянку BD підключають магазин опорів, що дозволяє з високою точністю підбирати еталонний опір R. Метод полягає в тому, що, знаючи три значення опорів R, R1, і R2 (позначені сірим кольором на рис. 1), можна обчислити четвертий опір Rx (позначений білим).

Для розуміння роботи цієї схеми замкнемо ключ (див. рис. 1.9). Після цього від джерела живлення через точку А потече струм I, який розділиться на два струми I1 і I2.

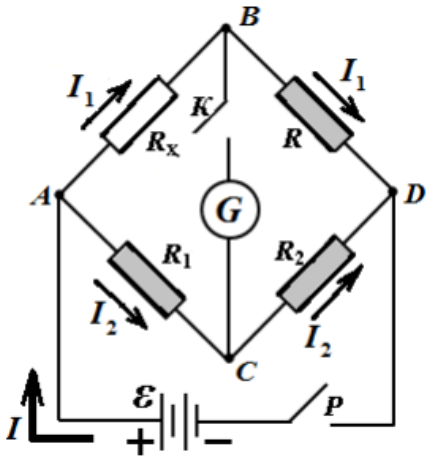


Рисунок 1.9 — Мостовий метод вимірювання опору

При замиканні кнопки К через гальванометр також протікатиме струм, напрямок якого залежить від того, яка з точок (В або С) має вищий потенціал. Особливий інтерес викликає ситуація, коли струм у діагоналі ВС дорівнює нулю. У такому випадку кажуть, що міст «збалансований». Розглянемо, за яких співвідношень між опорами резисторів можливий баланс моста.

Оскільки струму на ділянці ВС немає, через ділянки АВ і BD проходить однаковий струм, який позначимо як I1. Струм через ділянку ACD позначимо як I2. Відсутність струму на ділянці ВС свідчить про те, що потенціали точок В і С рівні, тобто φB = φC. Це означає, що від точки А (потенціал якої φA) до точок В і С потенціал знижується на однакову величину.

$$\varphi_A - \varphi_B = \varphi_A - \varphi_B, \tag{1.19}$$

або

$$U_{AB} = U_{AC} \tag{1.20}$$

Використовуючи закон Ома, можна написати

$$I_1 R_x = I_2 R_1, \tag{1.21}$$

Оскільки ділянки BD і CD зустрічаються в одній точці D (з потенціалом φD), падіння потенціалу на цих ділянках буде ідентичним.

$$\varphi_B - \varphi_{\mathbf{D}} = \varphi_C - \varphi_D, \tag{1.22}$$

Тобто

$$U_{BD} = U_{CD}, \tag{1.23}$$

Використовуючи закон Ома, можна написати

$$I_1 R = I_2 R_2, \tag{1.24}$$

Поділивши почленно рівності (1.21) і (1.24), отримаємо умову балансу моста:

$$R_x/R = R_1/R_2, \tag{1.25}$$

Звідки

$$R_x = R(R_1/R_2), \tag{1.26}$$

Для зручності вимірювань ділянку ACD (R1 і R2) замінюють реохордом, який складається з каліброваного сталевого або ніхромового дроту,

натягнутого вздовж лінійки зі шкалою AD (див. рис. 1.10). На реохорді переміщується рухомий контакт C, який розділяє дріт на два опори R1 і R2.

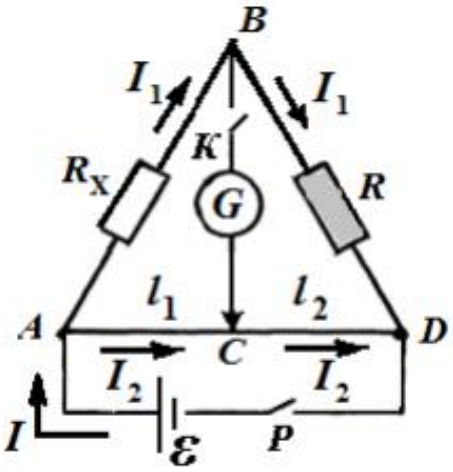


Рисунок 1.10 — Встановлення реохорда на ділянку AD

Оскільки опір однорідного дроту пропорційний його довжині, опори R1 і R2 будуть пропорційні довжинам ділянок AC (l1) і CD (l2) відповідно. Таким чином, для невідомого опору Rx, замінюючи відношення опорів на відношення довжин згідно з формулою (1.26), ми отримуємо робочу формулу:[5]

$$R_x = R(l_1/l_2) \qquad (1.27)$$

Отже, вимірювання опорів за допомогою мостової схеми зводиться до вимірювання довжин.

Наступний метод, це вимірювання способом омметра

Цей метод полягає у використанні омметрів, які побудовані на основі схем послідовного або паралельного з'єднання магнітоелектричного вимірювального механізму з приладом, що має вимірювальний опір. Приклади таких схем представлені на рисунку 1.11.

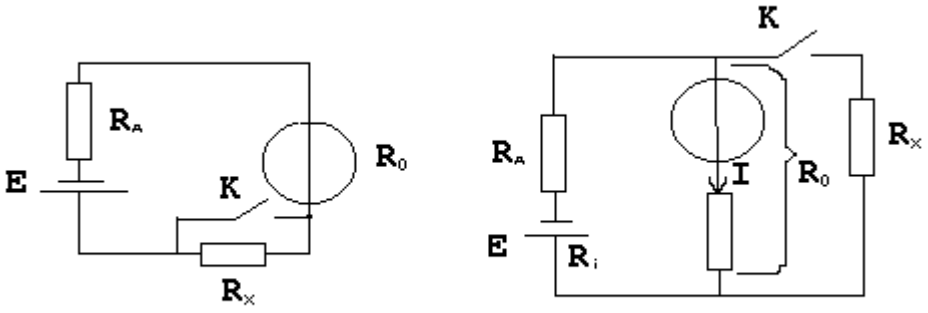


Рисунок 1.11 — Схеми послідовного та паралельного з’єднання

Тут Ro – опір обмотки вимірювального механізму омметра, Rд – опір додаткового резистора, що входить до схеми омметра, Ri – внутрішній опір джерела живлення Е омметра, Rx – вимірювальний опір, а ключ К. Послідовна схема (а) використовується для вимірювання великих опорів (100 Ом і більше), тоді як паралельна схема (б) призначена для вимірювання малих опорів. Відхилення показів омметра є функцією вимірювального опору, але також залежить від значення ЕРС джерела живлення. Щоб усунути вплив значення Е на покази омметра, в його конструкції передбачено регулювальний пристрій (Rд), за допомогою якого в послідовній схемі при закритому ключі К стрілку омметра встановлюють у нульове положення, а в паралельній схемі (б) при відкритому ключі стрілку омметра фіксують на відмітці «∞».

Дуже великі опори (понад 1 МГом), такі як ізоляційні, вимірюються за допомогою мегометри, які складаються з генератора постійного струму підвищеної напруги (250, 500, 1000, 2500 В) та логометричного вимірювача. Генератор може приводитися в дію вручну або живитися від мережі чи автономного джерела живлення.

Вимірювання опору за допомогою амперметра і вольтметра використовується для вимірювання малих і середніх опорів. При виборі вимірювальних приладів рекомендується віддавати перевагу магнітоелектричним приладам, оскільки вони споживають менше енергії з

кола і не піддаються впливу зовнішніх магнітних полів, що позитивно позначається на їх точності.

Важливо зазначити, що для вимірювання постійного струму, який перевищує 10 А, доцільно використовувати шунти або вимірювальні трансформатори струму. У цьому випадку струм первинної обмотки вимірювального трансформатора (ВТС) повинен бути рівним або більшим за вимірюваний струм, тоді як струм вторинної обмотки має відповідати межі вимірювання амперметра. У лабораторних ВТС номінальні струми можуть становити:

— I1H = 0,5; 1; 2; 2,5; 5; 7,5; 10; 15; 20; 25; 50; 75; 100; … 3000 А;

— I2H = 5 А (іноді 1 А).

Що стосується вимірювання напруги, якщо вона становить 600 В, вольтметр підключається безпосередньо до вимірювального кола паралельно навантаженню. У випадку, коли напруга менша за 1000 В, допускається підключення вольтметра послідовно з додатковим опором. Якщо ж напруга перевищує 1000 В, вольтметр слід підключати через вимірювальний трансформатор напруги (ВТН). У цьому випадку важливо враховувати співвідношення між первинною та вторинною напругою ВТН, яке визначається формулою:

$$U_1 = k * U_2, \tag{1.28}$$

де U1 і U2 – напруга первинної та вторинної обмоток відповідно,

k – коефіцієнт трансформації.

Таким чином, метод вимірювання опору за допомогою амперметра і вольтметра є ефективним і точним способом, який дозволяє отримувати достовірні результати при дотриманні всіх необхідних умов і рекомендацій. Це особливо важливо в лабораторних умовах, де точність вимірювань має критичне значення для подальшого аналізу та досліджень.[6]

1.4 Існуючі мікропроцесорні засоби

Мікропроцесорні засоби для вимірювання параметрів електронних компонентів є важливими інструментами в галузі електроніки та розробки пристроїв. Вони забезпечують зручність, точність та автоматизацію вимірювальних процесів, що робить їх популярними серед інженерів та ентузіастів електроніки.

Одним із найвідоміших мікропроцесорних засобів є Arduino (див. рисунок 1.12).

Arduino – це відкрите апаратне та програмне забезпечення, яке пропонує простоту використання та гнучкість для реалізації різноманітних проектів. Використовуючи Arduino разом із відповідними датчиками, модулями та програмним забезпеченням, можна вимірювати ємність, опір, індуктивність та інші характеристики електронних компонентів.

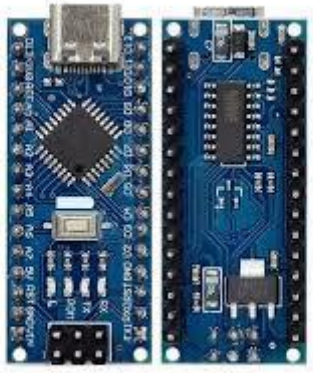


Рисунок 1.12 — Плата Arduino

Arduino має велику спільноту користувачів, що сприяє постійному розвитку нових бібліотек, програмних розширень та проектів. Існує безліч бібліотек для Arduino, які дозволяють проводити вимірювання параметрів електронних компонентів. Наприклад, бібліотека Arduino CapacitanceMeter дає змогу вимірювати ємність конденсаторів, а бібліотека Arduino ESRMeter – визначати еквівалентний серійний опір (ESR) конденсаторів.

Окрім Arduino, існують й інші мікропроцесорні засоби для вимірювання параметрів електронних компонентів. Наприклад, Raspberry Pi є ще одним популярним мікрокомп'ютером, який можна використовувати для вимірювання характеристик компонентів (див. рисунок 1.13). Raspberry Pi має більш потужний процесор і розширені можливості в порівнянні з Arduino, що дозволяє виконувати складніші вимірювання та аналізи. Завдяки своїй архітектурі та підтримці різноманітних операційних систем, Raspberry Pi може бути використаний для реалізації більш комплексних проектів, включаючи обробку даних, зберігання інформації та інтеграцію з іншими системами.

Також крім мікропроцесорних систем можна використати ситеми ПЛІС (програмованих логічних інтегральних схем). Основні типи систем, які можна використати для створення приладу, який буде вимірювати параметри електронних компонентів.

Рисунок 1.13 — Raspberry Pi 4

FPGA (Field-Programmable Gate Array) показано на рисунку 1.14.

FPGA складається з великої кількості програмованих логічних блоків, з’єднаних матрицею програмованих взаємозв’язків. Системи FPGA завдяки своїй внутрішній структурі дозволяють створювати конфігурації від простих цифрових елементів (логічних вентилів, тригерів) до складних цифрових схем (процесорів, контролерів, фільтрів). Застосування в вимірювальних приладах ПЛІС є чудовим вибором для обробки сигналів у реальному часі. Вони можуть реалізовувати високоточні генератори сигналів, фільтри, аналого-цифрові

перетворювачі (АЦП), а також сенсорні інтерфейси та комунікаційні модулі для передачі даних.

Рисунок 1.14 — Плата на базі системи FPGA

У приладобудуванні ПЛІС часто використовуються для виконання складних обчислень, таких як обробка спектру, фільтрація сигналу та перетворення Фур’є.

Переваги:

— висока швидкість, завдяки паралельній обробці сигналу і можливості роботи на високих частотах;

— гнучкість, FPGA можна програмувати кілька разів, що дозволяє пристрою адаптуватися до нових завдань;

— повна обробка сигналу в реальному часі, що важливо для точних вимірювальних приладів.

Недоліки:

— висока вартість, як самої ПЛІС, так і її розробки;

— розробка та програмування FPGA вимагає знання мов HDL (наприклад, Verilog або VHDL).

CPLD (Complex Programmable Logic Device)

Рисунок 1.15 — Плата на базі системи CPLD

CPLD відрізняються за архітектурою від FPGA, які містять менше логічних елементів, організованих у великий логічний масив, а не гнучку мережеву структуру, як FPGA. CPLD мають обмежену кількість програмованих блоків, що робить їх більш простими та гнучкими, але швидшими у виконанні простих логічних операцій.

Застосування в вимірювальних приладах CPLD зазвичай використовуються для виконання завдань комутації, таких як керування комутацією елементів схеми вимірювання або реалізація простих інтерфейсів. Вони можуть виконувати роль лічильників і генераторів імпульсів для стабілізації вимірювань і контролю режимів роботи приладу.

Переваги:

— низька вартість, порівняно з FPGA;

— низька затримка сигналу, через статичну архітектуру;

— низьке енергоспоживання, що робить CPLD придатними для портативних пристроїв.

Недоліки:

— менша гнучкість і продуктивність, через обмежену кількість логічних елементів, що робить їх непридатними для складної обробки сигналів;

— обмежені параметри конфігурації, через відсутність повної підтримки паралельної обробки.

SoC (System on Chip) з ядром FPGA

Рисунок 1.16 — Плата на базі системи SoC з ядром FPGA

SoC із вбудованим ядром FPGA — це комбінація багатоядерного процесора (зазвичай ARM) із ядром FPGA. Таке поєднання забезпечує потужну обчислювальну базу для роботи з програмним забезпеченням (на рівні процесора) і швидкої обробки спеціалізованих завдань за допомогою FPGA.

Застосування в вимірювальних приладах SoC із ядром FPGA забезпечує гнучкість інтеграції різноманітних вимірювальних функцій. Наприклад, на стороні процесора SoC дані можна обробляти та керувати ними, а на FPGA можна обробляти аналогові сигнали в реальному часі. Такі системи використовуються в складних вимірювальних пристроях, де потрібна багатозадачність і швидкість.

Переваги:

— інтеграція програмної та апаратної обробки, дозволяє швидко адаптуватися до змін вимірювальних завдань;

— висока продуктивність, завдяки поєднанню багатоядерного ЦП і FPGA;

— підходить для різних типів вимірювальних і контрольних інструментів, оскільки SoC можна перепрограмувати для різних завдань.

Недоліки:

— через необхідність поєднання програмного коду для процесора та конфігурації для FPGA;

— порівняно з FPGA або дискретними процесорами.

ASIC (Application-Specific Integrated Circuit)

ASIC — це спеціалізовані інтегральні схеми, призначені для виконання певних завдань. Постпродакшн не програмується, оптимізований для спеціальних високошвидкісних операцій.

Застосовувані вимірювальні прилади ASIC ідеально підходять для промислового застосування, де потрібна дуже висока точність і швидкість, наприклад, у комерційно доступних вимірювальних приладах. Він забезпечує мінімальну затримку сигналу та високу надійність для безперебійного тривалого використання. Після повної оптимізації вимірювальних функцій FPGA вони використовуються для масового виробництва пристроїв.

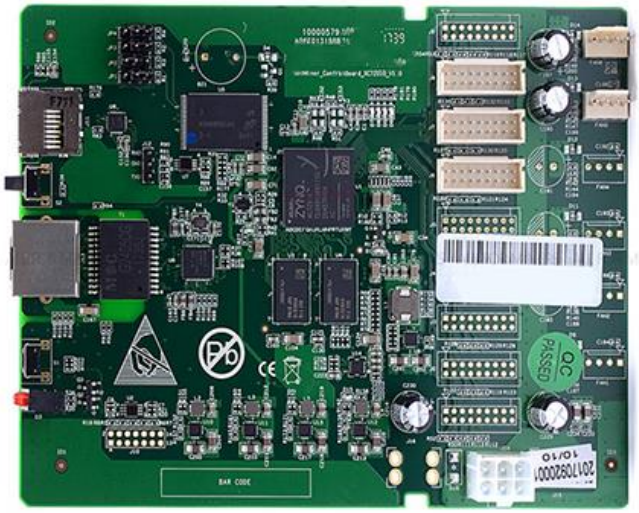


Рисунок 1.16 — Плата на базі системи ASIC

Переваги:

— висока швидкість і ефективність, завдяки оптимізованій структурі;

— низьке енергоспоживання, економічно ефективні ASIC;

— він підходить для великомасштабного виробництва, тому що це економічно доцільно для великомасштабного виробництва.

Недоліки:

— не можна використовувати ASIC, що обмежує їх використання;

— високі початкові витрати на розробку, роблять їх життєздатними лише для масового виробництва.

PLD (Programmable Logic Device).

PLD — це загальний термін, який включає програмовані логічні пристрої, які можуть змінювати свою конфігурацію для виконання певних завдань. PLD — це проста форма FPGA та CPLD, яка підходить для основних логічних операцій.

Використовувані вимірювальні прилади PLD можна використовувати для виконання простих логічних операцій, таких як перемикання елементів керування або зміна режимів вимірювання. Він може бути корисним у простих вимірювальних пристроях або як додатковий модуль у більш складних системах.

Переваги:

— простота використання та програмування, підходить для виконання простих логічних операцій;

— низька вартість робить їх доступними для простих інструментів;

— низьке енергоспоживання, важливо для портативних або переносних пристроїв.

Недоліки:

— не підходить для складної обробки сигналів;

— через відсутність паралельної обробки, PLD менш ефективні, ніж FPGA або SoC.

2 МЕТОДИ, АЛГОРИТМИ ТА ВИМІРЮВАННЯ ПАРАМЕТРІВ ЕЛЕКТРОННИХ КОМПОНЕНТІВ

2.1 Методи визначення типу електронного компонента

Визначення типу електронного компонента є важливим етапом у процесі тестування та діагностики електронних схем. Тестери, наприклад, на основі мікроконтролерів Arduino, використовують різні методи для визначення компонентів. Нижче наведено докладний опис основних методів визначення типу електронного елемента.

Метод вимірювання V-I характеристики базується на аналізі вольт-амперних характеристик електронного компонента. Цей метод передбачає подачу відомої напруги на компонент і вимірювання струму, що протікає через нього. На основі отриманих значень можна побудувати V-I характеристику, яка дозволяє визначити тип компонента.

Діод: для діодів напруга вимірюється в прямому та зворотному напрямку. Тестер подає напругу на анод діода та вимірює напругу на катоді. Якщо напруга на катоді нижче напруги на аноді, діод проводить струм, що свідчить про його активність. Однак, якщо струм не тече, це підтверджує, що діод працює належним чином. Якщо струм тече в протилежному напрямку, можливо, діод несправний або неправильно підключений.

У випадку транзисторів вимірюється напруга між базою, колектором і емітером. Тестер подає напругу на базу транзистора та вимірює напруги колектора та емітера. Якщо базова напруга перевищує напругу передавача, транзистор відкритий, що вказує на його тип (NPN або PNP). У випадку NPN-транзистора струм може протікати від колектора до емітера, але в PNP-транзисторі струм тече в протилежному напрямку.

Для визначення опору використовується метод вимірювання питомого опору. Тестер подає відому напругу на резистор і вимірює струм, що протікає через нього. Відповідно до закону Ома, опір можна розрахувати за формулою:

$$R = V/I, \quad (2.1)$$

де R — опір,

V — напруга,

I — електричний струм

Вимірювання опору дозволяє точно визначити номінал опору. Цей метод простий і ефективний для перевірки резистентності, оскільки не вимагає складних налаштувань.

Для визначення конденсаторів використовують електрометричний метод. Тестер вимірює час, необхідний для зарядки або розрядки конденсатора, дозволяючи розрахувати його ємність. Процес вимірювання ємності можна реалізувати таким чином:

Тестер подає напругу на один висновок, до якого підключений конденсатор.

Вимірюється час, протягом якого конденсатор заряджається до певної напруги.

Ємність визначається за формулою:

$$C=t/R, \quad (2.2)$$

де C — ємність,

t —час зарядки,

R — опір в ланцюзі.

Цей метод дозволяє точно визначити ємність конденсатора і його стан, що важливо для оцінки його працездатності.

Для визначення індуктивності використовують метод вимірювання індуктивності. Тестер вимірює час, за який індуктивність реагує на зміну струму. Процедуру індукційного вимірювання можна реалізувати наступним чином. Тестер подає напругу на один контакт, до якого підключено індуктор.

Вимірюється час, за який індуктивність реагує на зміну струму, що протікає через неї.

Індуктивність визначається за формулою:

$$L=V/(\Delta I/\Delta t), \quad (2.3)$$

де L — індуктивність,

V — напруга,

ΔI — зміна струму,

Δt — інтервал часу.

Даний метод дозволяє точно визначити складову індуктивності, що важливо для аналізу її характеристик в схемі.

Сучасні тестери напівпровідників можуть мати вбудовані функції автоматичного тестування, які дозволяють швидко та ефективно тестувати компоненти за заданими алгоритмами. Ці дії можуть включати комбінацію вищевказаних методів, що дозволяє скоротити час перевірки та підвищити точність результатів. Автоматичне тестування можна реалізувати за допомогою програмного забезпечення, яке відстежує процес тестування, аналізує результати та надає користувачеві інформацію про справність компонентів.

Використання цих методів в тестерах напівпровідників дозволяє точно визначити тип електронного компонента, його характеристики та стан. Кожен метод має свої переваги та обмеження, тому для досягнення найкращих результатів часто використовують комбінацію кількох методів. Ці процедури є основою роботи тестера та забезпечують його ефективність у вимірюванні та аналізі електронних компонентів. Вони також сприяють підвищенню надійності електронних пристроїв, що надзвичайно важливо в сучасній електроніці.

2.2 Алгоритм вимірювання параметрів

У даній роботі представлено алгоритм вимірювання параметрів електронних компонентів, реалізований для мікроконтролера ATmega328. Алгоритм складається з кількох кроків, які забезпечують точність і надійність вимірювань.

На першому етапі система ініціалізується та налаштовується. Послідовний (UART), ініціалізовано для виведення даних на зовнішній пристрій. РК-екран, регулюється для відображення результатів вимірювань. Вхідні/вихідні роз'єми, налаштовані на обробку аналогових і цифрових сигналів, що використовуються для вимірювання параметрів компонентів.

Перед початком вимірювань перевіряють напругу акумулятора. Зчитується значення напруги акумулятора. Відображається інформація про стан акумулятора (нормальний, низький або розряджений).

На наступному етапі система калібрується. Визначається еталонна напруга і внутрішній опір затворів, що використовуються для вимірювання.

Алгоритм виконує вимірювання різних електронних компонентів, таких як резистори, конденсатори, діоди та транзистори:

— опір вимірюється аналоговим зчитуванням значення затвора;

— ємність і еквівалентний послідовний опір (ESR) в конденсатора вимірюються шляхом заряджання та розряджання конденсаторів через відомі опори;

— визначається в діоді полярність (анод і катод) і виміряйте падіння напруги;

— визначається тип транзитора (NPN або PNP) і вимірюється коефіцієнт посилення (hFE) шляхом аналізу струмів між базою, колектором і емітером;

— отримання результатів

Результати вимірювання виводяться на РК-екран та інформація про знайдені компоненти, їх параметрах і значеннях відображається в зручному

для використання форматі.

Після закінчення вимірювань система повертається до початкового стану.

Представлений алгоритм забезпечує ефективне і точне вимірювання параметрів електронних компонентів, які можуть бути використані в різних наукових і практичних застосуваннях. Алгоритм реалізує комплексний підхід до вимірювання, який включає калібрування, перевірку напруги та обробку даних, що робить його надійним інструментом для електронних досліджень.

2.3 Методика вимірювання

Методика вимірювання визначає послідовність дій, алгоритми та принципи, за якими здійснюється процес визначення параметрів електронних компонентів. У цьому розділі розглянуто ключові підходи до виконання вимірювань, враховуючи особливості конструкції приладу, обрані методи обробки сигналів та програмну реалізацію. Описані методи забезпечують точність, повторюваність і стабільність отриманих результатів у межах заданого діапазону.

ESR (еквівалентний послідовний опір) — один із параметрів паразитних електролітичних конденсаторів, який останнім часом став популярним серед фахівців з ремонту електроніки. Вимірювачі ШОЕ разом із тестерами та мультиметрами стали необхідними інструментами для багатьох підприємців.

Цей параметр визначає опір, який виникає в конденсаторі при подачі змінного струму. Підвищення ESR може призвести до серйозних наслідків, таких як вихід пристрою з ладу, навіть якщо вимірювання ємності не показують проблем.

У той же час при практичному ремонті особливої точності вимірювання АТ зазвичай не потрібно. Навіть якщо є якась помилка в роботі датчика ESR, цього достатньо для визначення несправних компонентів. Такі вимірювання допомагають оцінити загальний стан конденсатора і його придатність для

подальшого використання в пристрої.

Однак у випадках, коли конденсатори керуються великими імпульсними струмами (наприклад, у фільтрах перетворювача), більш точний аналіз ESR може бути дуже важливим. Похибки вимірювань навіть у десятки чи сотні Ом можуть сильно вплинути на продуктивність таких компонентів.

На малюнку 2.1 показана залежність реактивного опору і паразитного виходу великого конденсатора від робочої частоти перетворювача (декілька десятків кГц). Ці значення набагато нижче в порівнянні з ESR, що безпосередньо впливає на загальний опір (опір) конденсатора змінного струму. При великих імпульсних струмах, що досягають десятків ампер, ESR розсіює реальну потужність, викликаючи нагрів електроліту електроліту і конденсатора.

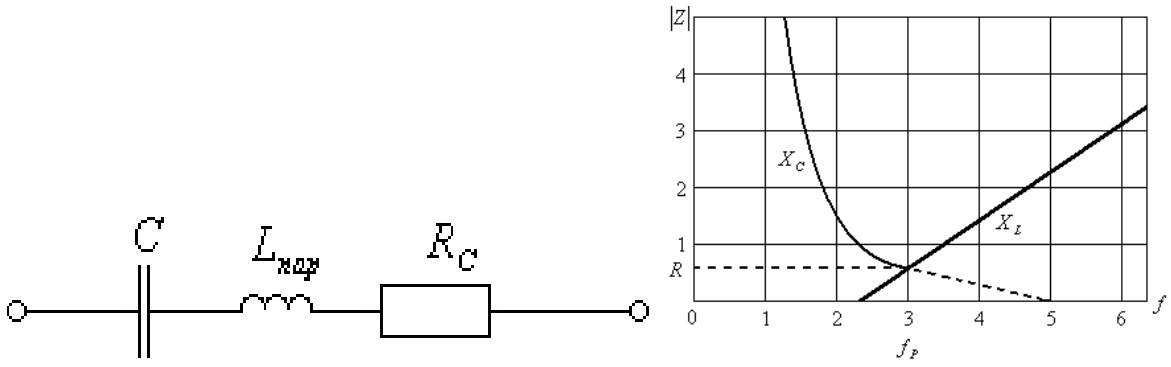


Рисунок 2.1 — Еквівалента схема параметрів конденсатора

Тангенс кута втрат (tan δ) є показником, який характеризує енергетичні втрати в електролітичних конденсаторах. Зазвичай виробники вказують значення tan δ для частоти 120 Гц у технічній документації. Однак при використанні конденсаторів у потужнострумових імпульсних перетворювачах (ПІП) ці дані можуть втрачати свою актуальність, оскільки робочі частоти ПІП значно вищі.

Для фільтрів вторинного випрямляча виробники пропонують конденсатори з низьким опором (Low Impedance) і надають значення імпедансу, виміряного при частоті 100 кГц, для кожного номіналу в технічних

таблицях. Активну складову опору (ESR) можна визначити, використовуючи формулу (2.4).

$$R = \sqrt{(Z^2 - X^2)}, \quad (2.4)$$

де Z - загальний опір, виміряний при частоті 100 кГц,

X - реактивний опір (імпеданс) конденсатора.

Наприклад, для конденсатора Jamicon серії WL ємністю 1000 мкФ та напругою 25 В, із реактивним опором Xc=0,0016 Ω, і загальним опором Z=0,04Ω, можна визначити значення ESR. Тангенс кута втрат R/Xc в цьому випадку дорівнює 39,97/1,6≈25. Хоча паразитна індуктивність у цих розрахунках не враховується, у певних ситуаціях її вплив може бути значним.

Пристрої для вимірювання ESR, що працюють на частотах від 40 до 100 кГц, є найбільш поширеними серед ремонтників електроніки. Однак для невеликих ємностей (менше 10 мкФ) їх точність може бути недостатньою, оскільки реактивний опір конденсатора на високих частотах може перевищувати значення ESR.

У таких випадках для більш точних вимірювань використовують прості зонди зі світлодіодним або стрілочним індикатором, а також цифрові ESR-метри різного рівня складності, які працюють на нижчих частотах. Це дозволяє коректно визначати ESR навіть для малих номіналів конденсаторів.

ESR та ємність є ключовими параметрами для оцінки якості та придатності конденсатора у пристрої. Врахування цих характеристик допомагає забезпечити належну роботу електроніки, що робить їх вимірювання важливим етапом під час ремонту та діагностики.

Для конденсатора, що заряджається від джерела постійного струму, можна застосувати залежність між напругою на конденсаторі (U) та зарядним струмом (I), яка випливає із закону Ома.

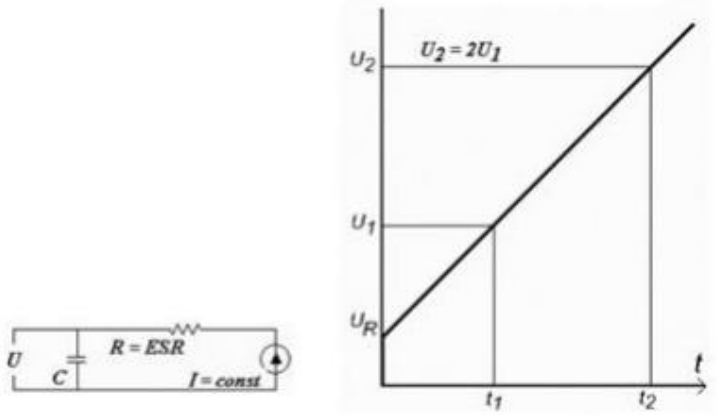


Рисунок 2.2 — Вимірювання ESR конденсатора

Згідно з цією залежністю, напруга на конденсаторі визначається як добуток зарядного струму на опір:

$$U = I \cdot R \tag{2.5}$$

Для конденсатора активний опір ESR характеризує внутрішні втрати, що виникають у процесі його роботи. З урахуванням цього, залежність можна записати, замінивши опір R на еквівалентний послідовний опір ESR:

$$U = I \cdot ESR \tag{2.6}$$

Таким чином, еквівалентний послідовний опір конденсатора (ESR) визначається як співвідношення між напругою на конденсаторі та струмом заряду:

$$ESR = U/I \tag{2.7}$$

Описаний метод може використовуватися для визначення ESR конденсатора за відомими значеннями напруги та струму заряду.

Існують різні підходи до вимірювання еквівалентного послідовного опору (ESR) конденсатора, що базуються на різних технологіях і алгоритмах. Один із них передбачає використання порогів напруги для управління таймерами та подальші математичні розрахунки. У цьому підході схема застосовує компаратори для відстеження напруги на конденсаторі (UR) під час заряду. Отримані значення обробляються мікроконтролером, який розраховує ємність і ESR конденсатора.

Цей метод має певні обмеження, особливо при вимірюванні малих ємностей, оскільки короткі імпульси під час вимірювання можуть впливати на точність результатів. Також важливо враховувати, що ESR, визначений за допомогою постійного струму, є лише відносним показником якості конденсатора, оскільки він не враховує індуктивні втрати, які проявляються при змінному струмі на високих частотах.

Для більш детальної оцінки конденсаторів, що працюють у високочастотних колах, застосовуються складніші методи. Наприклад, аналіз фазового зсуву дозволяє отримати точнішу картину поведінки компонентів на різних частотах і врахувати індуктивні ефекти.

Процес вимірювання з використанням аналого-цифрового перетворювача (АЦП) забезпечує точні дані щодо ESR та ємності конденсаторів. Основні кроки цього процесу:

— підключення конденсатора або зонда до входу АЦП мікроконтролера чи іншого пристрою, перевірка живлення;

— налаштування параметрів АЦП, вибір роздільної здатності, швидкості перетворення та інших параметрів для забезпечення точності вимірювань;

— ініціалізація АЦП, встановлення початкових значень і таймерів для підготовки до вимірювання;

— заряджання конденсатора, подача постійного струму або напруги на конденсатор до досягнення заданого рівня;

— вимірювання напруги, перетворення напруги на конденсаторі в цифрове значення за допомогою АЦП;

— розрахунок параметрів, математичні обчислення для визначення ESR або ємності;

— повторення вимірювань, перевірка кількох конденсаторів або спостереження за змінами в часі для одного зразка;

— аналіз результатів, оцінка якості конденсаторів, визначення несправностей та прийняття рішень щодо їх подальшого використання.

Цей метод є універсальним і може застосовуватися в різних сферах електроніки для оцінки стану компонентів і підтримки надійної роботи пристроїв.

Пристрій, який може вимірювати індуктивність (L), ємність (C) і опір (R) електронних компонентів. Вимірювання опору за допомогою вимірювача LCR використовує подібні принципи до вимірювання інших змінних, але є кілька особливостей, які слід враховувати під час вимірювання опору. Головною перевагою цього методу є те, що вимірювач дозволяє дуже точно вимірювати опір і враховувати особливі умови, які виникають при використанні компонентів зі змінним струмом.

У вимірювачі опір (R) вимірюється підключенням компонента до змінного струму, і струм змушує компонент виробляти відгук, який вимірюється приладом. Оскільки опір є складовою опору, вимірювач обчислює значення опору, аналізуючи цей опір за допомогою відомих методів і алгоритмів, щоб отримати точний результат.

Важливим аспектом є те, що вимірювач може вимірювати опір на різних частотах. На відміну від звичайних мультиметрів, які зазвичай працюють на постійній частоті або постійному струмі (DC), вимірювачі можуть працювати на змінній частоті, що дозволяє враховувати вплив індуктивності та ємності, а також реактивних і активних резистивних компонентів. Це особливо важливо, коли компонент працює від змінного струму (AC) або високої частоти, де можуть виникнути реактивні ефекти.

Опір — це комплексна змінна, яка поєднує активний опір (R) і реактивний опір, який залежить від індуктивності (L) або ємності (C). Вимірювач може виміряти загальний опір, а потім вивести ефективне значення опору.

3. ПРОЕКТУВАННЯ АПАРАТНОЇ ТА МІКРОПРОЦЕСОРНОЇ СИСТЕМИ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Розробка структурної схеми

Розробка структурної схеми є ключовим етапом у процесі проектування електронних пристроїв. Вона виконує кілька важливих функцій, зокрема визначає структуру та взаємозв'язки між компонентами пристрою, а також організовує ці компоненти в логічну систему. Цей етап передує безпосередньому створенню самого пристрою і дозволяє провести необхідні аналізи та планування.

Структурна схема вимірювача параметрів електронних компонентів була розроблена з метою забезпечення коректної роботи приладу. Вона охоплює визначення функціональних блоків, їх взаємозв'язків та фізичного розташування на схемі.

Першим етапом у створенні структурної схеми стало визначення функціональних блоків, які складають вимірювач. До цих блоків входять блок живлення, мікроконтролер, датчики, інтерфейсні модулі та інші елементи, необхідні для виконання заданих функцій. Кожен блок виконує свою специфічну роль у загальному функціонуванні вимірювача.

Після визначення функціональних блоків були встановлені зв'язки між ними. Це можуть бути сигнальні лінії, шини даних, лінії керування тощо, які забезпечують передачу сигналів і даних між компонентами. Ці зв'язки були ретельно проаналізовані та визначені для забезпечення належного функціонування пристрою.

На представленій схемі показано розташування різних компонентів, серед яких блок керування, платформа Arduino Nano, акумулятор або живлення від шнура живлення, рідкокристалічний дисплей та клемні колодки. Ці елементи становлять основу структурної схеми приладу для вимірювання параметрів конденсаторів.

Основним елементом структурної схеми є платформа Arduino Nano, яка виконує функції мікроконтролера та забезпечує управління всіма функціями пристрою. Для живлення платформи можна використовувати акумулятор 9В або блок живлення з кабелем type-C. Використання блоку живлення забезпечує більшу стабільність роботи приладу в порівнянні з батарейкою.

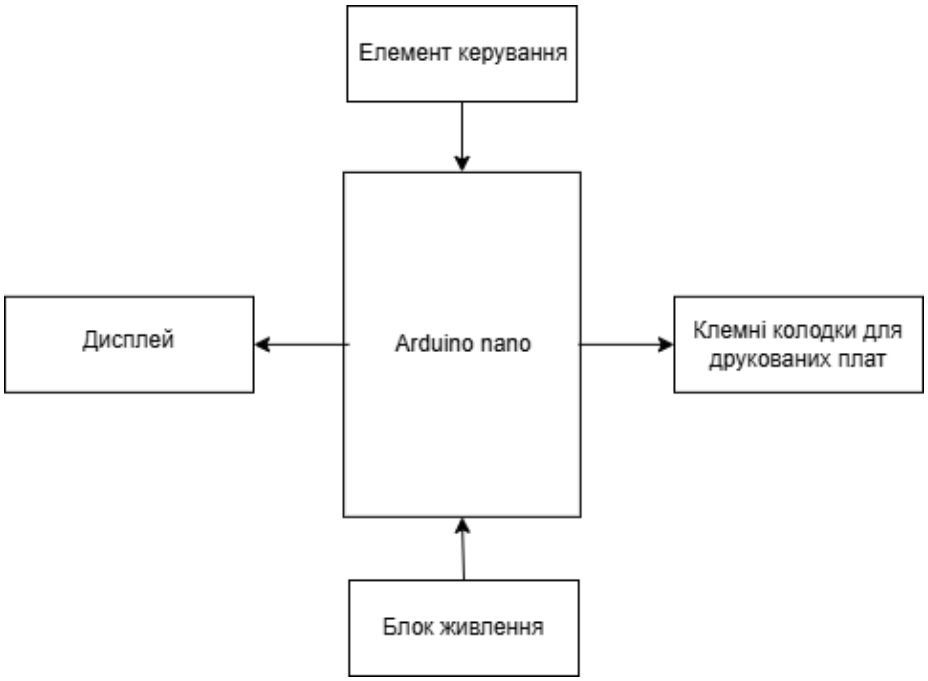


Рисунок 3.1 — Структурна схема пристрою

У схемі також присутній блок керування, що складається з однієї кнопки.

Крім того, до платформи Arduino підключається рідкокристалічний дисплей, який відображає інформацію про вимірювальний елемент або результати вимірювання. Цей дисплей забезпечує зручний моніторинг вимірювального процесу та отримання необхідних даних.

Також на схемі є клемні колодки, що забезпечує зручність підключення електронних компонентів для вимірювання їх параметрів.

Завдяки правильному розташуванню цих компонентів на схемі, пристрій може ефективно виконувати свої функції та відображення результатів на рідкокристалічному дисплеї. Це дозволяє користувачу швидко та зручно

отримувати інформацію про ємність та еквівалентний серійний опір конденсаторів.

3.2 Вибір електронних компонентів

Вибір електронних компонентів є дуже важливим кроком у розробці будь-якого пристрою. Правильний підбір компонентів забезпечує не тільки продуктивність, але й надійність і ефективність роботи системи.

Для реалізації пристрою знадобиться наступні компоненти Arduino nano, дисплей, живлення type-C, кнопка та клемна колодка.

Основним компонентом, який використовується в проекті, є плата Arduino Nano(зображена на рисунку 3.2). Характеристики плати наведені в таблиці 2.1. На платі розташовані 14 цифрових входів/виходів, з яких 6 може використовуватися як ШИМ — виходи. ШИМ-виходи (або PWM — Pulse Width Modulation) — це виходи, які використовують метод широтно-імпульсної модуляції для керування потужністю, що подається на навантаження. Вони генерують серію імпульсів змінної ширини, що дозволяє контролювати середнє значення вихідної напруги. Чим ширший імпульс, тим вище середня вихідна напруга. Окрім того, плата оснащена 8 аналоговими входами, що дають змогу зчитувати дані з аналогових датчиків.

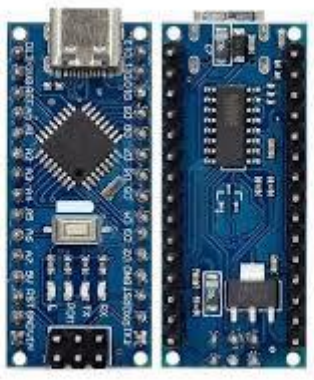


Рисунок 3.2 — Плата Arduino Nano

Arduino Nano можна живити через USB type-C, від зовнішнього джерела нестабілізованої напруги від 6В до 20 В (через вихід 30) або від стабілізованої напруги 5 В (через вихід 27). Плата автоматично вибирає джерело з більшою напругою, що спрощує її використання.

Живлення на мікросхему FTDI FT232RL подається тільки при підключенні Arduino Nano через USB. Тому, якщо пристрій працює від інших зовнішніх джерел (не через USB), вихід 3.3 В (який забезпечує FTDI) буде відключений. Це може спричинити мерехтіння світлодіодів RX і TX, якщо на контактах 0 і 1 присутній високий рівень сигналу.

Пам'ять для програм у мікроконтролері ATmega168 становить 16 КБ, з яких 2 КБ займає завантажувач. У ATmega328 обсяг пам'яті для програм становить 32 КБ, з яких також 2 КБ виділено під завантажувач. Крім того, ATmega168 містить 1 КБ оперативної пам'яті SRAM і 512 байт EEPROM, для роботи з якою використовується бібліотека EEPROM. Мікроконтролер ATmega328, своєю чергою, має 2 КБ SRAM та 1 КБ EEPROM.

За допомогою функцій pinMode(), digitalWrite() та digitalRead() кожен із 14 цифрових виходів Arduino Nano може працювати як вхід або вихід. Робоча напруга виходів становить 5 В, а максимальний струм, який може пропускати один вихід, дорівнює 40 мА. Всі виходи мають внутрішні підтягуючі резистори (за замовчуванням вимкнені) з номіналом від 20 кОм до 50 кОм. Деякі контакти Arduino виконують додаткові функції:

— контакти 0 (RX) та 1 (TX) служать для прийому (RX) і передачі (TX) даних через послідовний інтерфейс, вони з’єднані з відповідними контактами мікросхеми FTDI для USB-UART;

— контакти 2 та 3 можуть використовуватись як джерела переривань, що спрацьовують при різних умовах, таких як низький рівень сигналу, передній або задній фронт чи зміна сигналу;

— ШІМ контакти 3, 5, 6, 9, 10 та 11 дозволяють через функцію analogWrite() видавати 8-бітне аналогове значення у вигляді ШІМ-сигналу;

— контакти 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK) забезпечують зв’язок через інтерфейс SPI;

— вбудований світлодіод, підключений до цифрового контакту 13, він вмикається при подачі сигналу HIGH та вимикається при LOW.

Таблиця 3.1 — Характеристика Arduino Nano

Мікроконтролер	ATmega328
Робоча напруга	5В
Вхідна напруга (рекомендовано)	Від 7В до 12В
Вхідна напруга (обмеження)	Від 6В до 20В
Цифрові контакти введення/виведення	14 (з яких 6 використовуються як вихід ШИМ)
Аналогові входи	8
Постійний струм на контакт введення/виведення	40 мА
Флеш-пам'ять	16 КБ (ATmega168) або 32 КБ (ATmega328), з яких 2 КБ використовуються завантажувачем
SRAM	1 КБ (ATmega168) або 2 КБ (ATmega328)
EEPROM	512 байт (ATmega168) або 1 КБ (ATmega328)
Тактова швидкість	16 МГц

Arduino Nano також має 8 аналогових входів, які можуть представляти аналогову напругу у вигляді 10-бітного числа (до 1024 значень). Стандартний

діапазон вимірювання напруги — від 0В до 5 В, але його можна змінити, підключивши вихід AREF і скориставшись функцією analogReference(). Окрім основних функцій, деякі контакти мають ще додаткові можливості:

— I2C — контакти 4 (SDA) і 5 (SCL) забезпечують передачу даних через інтерфейс I2C (TWI) за допомогою бібліотеки Wire.

На платі є також кілька спеціальних контактів:

— AREF — забезпечує опорну напругу для аналогових входів, яку можна використовувати разом із функцією analogReference();

— Reset — подача низького рівня сигналу (LOW) на цей контакт призводить до перезавантаження мікроконтролера. Контакт часто використовується для кнопки скидання на платах розширення.

ATmega328 — це потужний 8-бітний мікроконтролер компанії Microchip, який має збалансовану продуктивність, мале енергоспоживання та достатню кількість периферійних модулів. Основні характеристики даного мікрокотролера наведенні нижче.

Особливості ATmega328:

— високопродуктивний 8-розрядний мікроконтролер AVR® з низьким енергоспоживанням;

— розширена архітектура RISC;

— 131 потужна інструкція – більшість виконання за один такт;

— 32 x 8 робочих регістрів загального призначення;

— повністю статична робота;

— пропускна здатність до 16 MIPS на частоті 16 мгц;

— вбудований двотактний множник;

— сегменти енергонезалежної пам'яті високої витривалості;

— 32 Кбайт внутрішньосистемної самопрограмованої флеш-пам'яті програм;

— 1 Кбайт EEPROM;

— 2 Кбайт внутрішньої SRAM;

— цикли запису/стирання — 10 000 спалахів/100 000 EEPROM;

— додатковий розділ коду завантаження з незалежними бітами блокування;

— внутрішньо системне програмування за допомогою програми завантаження на мікросхемі;

— справжня операція читання під час запису;

— блокування програмування для захисту програмного забезпечення;

— периферійні функції;

— два 8-бітних таймера/лічильника з окремим режимом попереднього масштабування та порівняння;

— один 16-бітний таймер/лічильник з окремим попереднім масштабувальником, режимом порівняння та захоплення режим;

— лічильник реального часу з окремим генератором;

— шість каналів ШИМ;

— 8-канальний 10-бітний АЦП у корпусі TQFP та QFN/MLF;

— вимірювання температури;

— програмований послідовний USART;

— послідовний інтерфейс головного/підлеглого SPI;

— байт-орієнтований 2-провідний послідовний інтерфейс (Phillips I2 C сумісний);

— програмований сторожовий таймер з окремим вбудованим генератором;

— вбудований аналоговий компаратор;

— переривання та пробудження після зміни PIN-коду;

— спеціальні функції мікроконтролера;

— скидання під час увімкнення живлення та програмоване виявлення перегорання;

— внутрішній калібрований генератор;

— зовнішні та внутрішні джерела переривань;

— шість режимів сну: режим очікування, зменшення шуму АЦП, енергозбереження, вимкнення живлення, режим очікування, і розширений

режим очікування atmega328p 8-розрядний мікроконтролер AVR із 32 Кбайтами в системі;

— введення/виведення та пакети;

— 23 програмованих лінії введення/виведення;

— 32-вивідний TQFP і 32-контактний QFN/MLF.

Робоча напруга від 2,7 до 5,5 В для ATmega328P.

Діапазон температур автомобіля від мінус 40°C до плюс 125°C.

Клас швидкості:

— від 0 МГц до 8 МГц при 2,7В до 5,5 В (автомобільний діапазон температур від мінус 40 °C до плюс 125 °C);

— від 0МГц до 16 МГц при 4,5В до 5,5 В (автомобільний діапазон температур від мінус 40 °C до плюс 125 °C);

— низьке енергоспоживання;

— активний режим: 1,5 мА при 3 В;

— режим вимкнення: 1 мкА при 3 В.

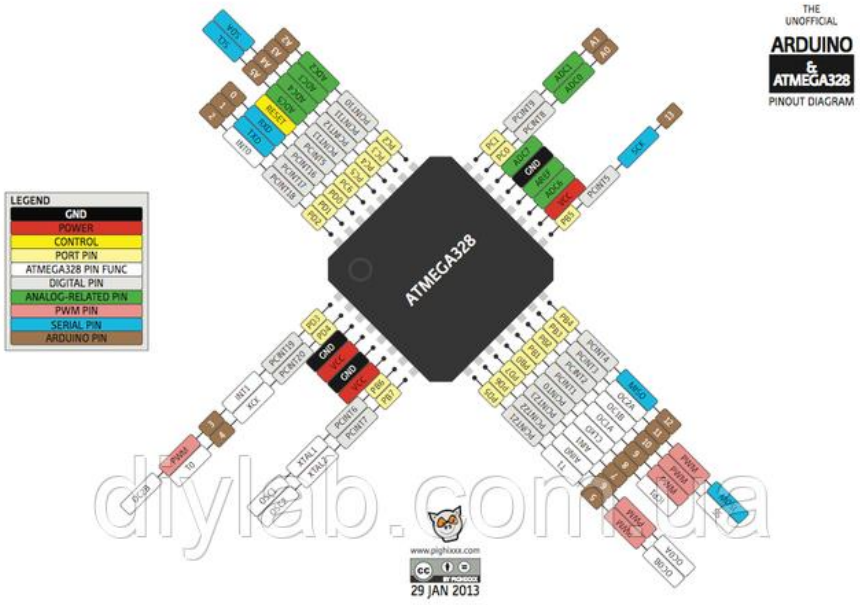


Рисунок 3.3 — Мікроконтролер ATMEGA328

На завершення, слід підкреслити, що для безпечного використання плати Arduino Nano необхідно дотримуватись встановлених обмежень щодо напруги

живлення. Напруга, що подається на плату, повинна бути в діапазоні від 7В до 12В. Використання нижчої напруги може призвести до нестабільної роботи, тоді як надмірна напруга може викликати перегрів і пошкодження регулятора напруги. Тому важливо бути уважним при виборі напруги живлення для вашого проекту.

В цілому, плата Arduino Nano з мікроконтролером ATmega328 є потужним і універсальним рішенням для вашого пристрою. Їхня комбінація дозволяє реалізувати різноманітні функції та ідеї, забезпечуючи високу продуктивність і надійність роботи.

Щоб підключити дисплей до плати Arduino Nano через інтерфейсний модуль I2C (IIC) або SPI, слід врахувати кілька важливих моментів.

Модуль інтерфейсу I2C/IIC/SPI для дисплея має чотири контакти: VCC, GND, SDA та SCL, схема яких представлена на рисунку 3.4.

Рисунок 3.4 — Модуль інтерфейсу I2C IIC SPI Arduino

Для забезпечення правильного живлення модуль і дисплей потрібно підключити наступним чином: контакт VCC модуля з’єднати з виводом VCC на платі Arduino, а контакт GND – з виводом GND. Це гарантує стабільне живлення обох пристроїв.

Лінії SDA (Serial Data) і SCL (Serial Clock) використовуються для передачі даних між дисплеєм і платою через протокол I2C. При підключенні дисплея через I2C контакт SDA модуля слід з’єднати з виводом SDA на платі Arduino, а SCL – з виводом SCL.

Це підключення забезпечує надійну передачу даних між платою Arduino та дисплеєм через інтерфейс I2C.

На рисунку 3.5 представлена схема підключення дисплея до плати Arduino через I2C-інтерфейс. Після коректного підключення дисплей можна використовувати для виведення різних даних, тексту, графіки тощо, залежно від вимог вашого проєкту та програмного коду, який створить користувач.

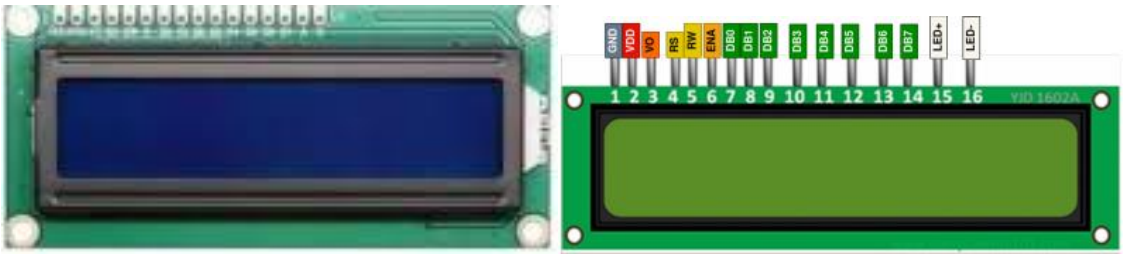


Рисунок 3.5 — Рідкокристалічний дисплей та призначення пінів

РК-дисплей, який використовується для відображення інформації, розташований на верхній кришці корпусу разом із кнопкою. Таке розміщення полегшує доступ до екрану та кнопки, забезпечуючи зручність користування та управління системою.

Для живлення плати можна використовувати батарею типу «Крона» на 9В або блок живлення 5В, показаний на рисунку 3.6. Обидва варіанти забезпечують необхідну напругу для стабільної роботи плати. Вибір між батареєю та блоком живлення залежить від ваших потреб та доступних ресурсів.

У цілому, ці компоненти та їх розміщення в системі сприяють оптимальній функціональності та зручності у користуванні пристроєм.

Також присутня клемна колодка на 3 піна, щоб вимірювати параметри елементів, які будуть вставлені в колодку, зображено на рисунку 3.7

3.3 Розробка схеми електричної принципової

Процес створення принципової схеми є одним з найважливіших етапів у виробництві електронного пристрою. У цьому розділі описано основні

принципи, методи та інструменти, необхідні для створення правильної схеми, яка забезпечує правильну роботу та взаємодію всіх компонентів.

Рисунок 3.6 — Блоки живлення для плати Arduino Nano

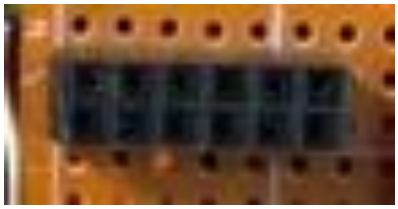


Рисунок 3.7 — Клемна колодка

При створенні системи важливо враховувати як експлуатаційні характеристики кожного елемента, так і їх розташування і підключення, що сприятиме підвищенню надійності майбутнього пристрою і максимальному енергоспоживанню.

Головну та важливу роль на схемі відіграє плата Arduino Nano. На рисунку 3.8 зображено схема електрична принципова плати Arduino. Також на рисунку зображено рідкокристалічний дисплей та плату LCM 1602 ICC. Відповідно плата LCM 1602 ICC під’єднана до плати Arduino використовуючи наступні піни A5, A4 та 5 вольт. Піни A5 та A4 використовуються для обміну інформації з платою LCM 1602 ICC, яка в свою чергу передає інформацію з плати на дисплей. Також зображено кнопка, для активації пристрою, та

виведення інформації на дисплей. До піна Ref під’єднаний конденсатор на 100 нано фарад.

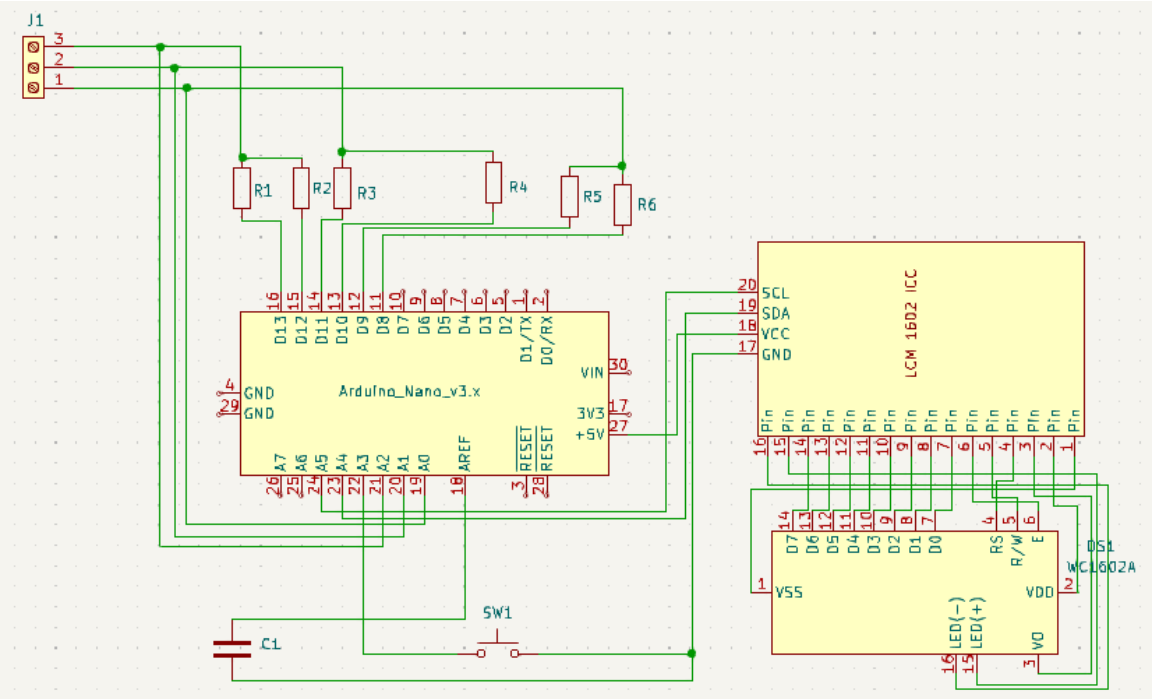


Рисунок 3.8 — Схема підлючення плати Arduino Nano

На схемі розташована клемна колодка, яка під’єднана до наступних пінів, A0, A1, A2, а також під’єднаний до пінів D8, D9, D10, D11, D12, D13 через резистори наступних опорів:

— пін D8 під’єднаний резистор R6 з опором 680 Ом;

— пін D9 під’єднаний резистор R5 з опором 470 кОм;

— пін D10 під’єднаний резистор R4 з опором 680 Ом;

— пін D11 під’єднаний резистор R3 з опором 470 кОм;

— пін D12 під’єднаний резистор R2 з опором 680 Ом;

— пін D13 під’єднаний резистор R1 з опором 470 кОм.

Також відбувається з’єднання резисторів між собою таких як: R1 та R2, R3 та R4, R5 та R6. Для налаштування чутливості, було вирішено з’єднати резистори паралельно.

Налаштування чутливості вимірювального контуру за допомогою паралельних резисторів є важливою технікою, особливо якщо прилад

призначений для точного вимірювання невеликих змін параметрів компонентів, таких як опір, ємність або індуктивність. Паралельне з'єднання резисторів дозволяє змінювати відповідний опір в ланцюзі, що в свою чергу впливає на чутливість приладу.

При паралельному з'єднанні резисторів загальний опір, що впливає на струм, що проходить через вимірювальний елемент, зменшується. Залежно від цілей вимірювання та діапазону значень вимірюваних компонентів можна вибрати відповідний загальний опір, який забезпечить більшу чутливість до невеликих змін. Наприклад, якщо в вимірюваному компоненті можна виявити невеликі зміни, зменшення відповідного опору через паралельне з’єднання допоможе зробити ці зміни більш помітними для вимірювального обладнання.

Приклад впливу на чутливість Паралельне з'єднання резисторів R5 (470 кОм) і R6 (680 Ом) дозволяє отримати еквівалентний опір, значно менше 470 кОм, але не такий малий, як 680 Ом. Це створює баланс, який дозволяє отримати бажаний діапазон напруги зменшення опору збільшує силу струму в колі, що у свою чергу, змінює вихідну напругу. Це покращує здатність точно зчитувати невеликі зміни у вхідному сигналі при зміні значень компонентів.

Підвищення чутливості при вимірюванні невеликих змін опору якщо схему налаштовано на вимірювання невеликих змін опору компонентів, паралельне з’єднання робить ці зміни помітними. Наприклад, якщо опір змінюється на кілька сотень Ом, ця зміна буде більш вираженою в загальному еквівалентному опорі, що робить вимірювання більш чутливим.

Використання паралельних з'єднань для калібрування паралельне з'єднання опорів у ланцюгах вимірювального обладнання також корисне для калібрування приладу. Регулюючи номінальний опір при паралельному з’єднанні, ви можете вибрати точний необхідний опір, який налаштує схему для визначення відповідного діапазону сигналу. Цей параметр часто необхідний для пристроїв, які мають виявляти дуже незначні зміни параметрів, наприклад, в аналогових датчиках або вхідних ланцюгах для вимірювання ємності чи індуктивності.

Тому паралельне з'єднання резисторів допомагає налаштувати чутливість вимірювального контуру таким чином, щоб він міг реагувати на найменші зміни, роблячи процес вимірювання більш точним і надійним.

3.4 Розробка алгоритму роботи

Для програмування платформи Arduino та опису функціонування пристрою необхідно розробити алгоритм, що визначатиме послідовність дій для досягнення необхідного результату.

Алгоритм роботи пристрою, який проектується, може бути таким:

— ініціалізація платформи Arduino;

— підключення електронного компонента;

— натискання кнопки;

— відображення необхідної інформації на підключеному дисплеї, що може включати текст, числа, графічні елементи тощо.

Спочатку здійснюється налаштування платформи Arduino, яке включає конфігурацію портів введення-виведення, ініціалізацію змінних і підключення необхідних бібліотек. Користувач підключає електронний компонент, наприклад конденсатор, резистор або діод, в клемну колодку . Далі натискає на кнопку. На екран виводиться інформація про компонент і його параметри.

Цей алгоритм можна відобразити у вигляді блок-схеми рисунок 3.9, що показує послідовність дій та їх взаємозв'язок. Запрограмування Arduino за цим алгоритмом дозволить коректно виконувати потрібні функції і забезпечити контроль роботи пристрою.

Лічильник на платформі Arduino використовує основну властивість конденсаторів, що називається постійною часу. Постійна часу визначає період, за який напруга на конденсаторі досягає 63,2% від своєї кінцевої величини при повному заряді.

Arduino може вимірювати ємність конденсатора, оскільки час заряджання конденсатора прямо залежить від його ємності за формулою:

$$TC = R \cdot C, \tag{3.1}$$

де TC — постійна часу конденсатора (в секундах),

R — опір кола (в Омах),

C — ємність конденсатора (в фарадах).

Рисунок 3.9 — Блок-схема алгоритму

Використовуючи рівняння:

$$C = TC/R\ , \tag{3.2}$$

Можна розрахувати значення ємності в фарадах.

При тестуванні відомий струм, що протікає через конденсатор протягом

короткого часу, тому конденсатор не встигає зарядитись повністю. Струм створює напругу на конденсаторі, що визначається добутком струму та еквівалентного серійного опору (ESR) конденсатора, а також невеликою напругою, спричиненою малим зарядом. Оскільки струм відомий, значення ESR можна обчислити, розділивши виміряну напругу на струм.

3.5 Бібліотеки для реалізації коду

Для керування дисплеєм і приймачем було підключено наступні бібліотеки в лістингу

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/sleep.h>
#include <stdlib.h>
#include <string.h>
#include <avr/eeprom.h>
#include <avr/pgmspace.h>
#include <avr/wdt.h>
#include <avr/interrupt.h>
#include <math.h>
#include <stdint.h>
#include <avr/power.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
```

Бібліотека LiquidCrystal.h використовується для відображення інформації на LCD-екрані. Вона надає необхідні функції та методи для управління символьними екранами, що дозволяє легко виводити текст, числа та інші символи.

Бібліотека Wire.h забезпечує зв'язок між мікроконтролером Arduino та пристроями через інтерфейс I2C. Вона містить функції для передачі та отримання даних по шині I2C, що дозволяє зручно підключати сенсори, дисплеї, модулі зовнішнього зберігання та інші пристрої до Arduino.

Бібліотека avr/eeprom.h призначена для роботи з даними, збереженими в EEPROM мікроконтролерів AVR. EEPROM (електрично стирабельна/програмована пам'ять) є неволатильним типом пам'яті, що зберігає дані навіть без подачі живлення. Ця бібліотека надає простий

інтерфейс для запису та зчитування даних, що дозволяє зберігати інформацію між різними сеансами роботи мікроконтролера.

Бібліотека avr/interrupt.h використовується для налаштування переривань на мікроконтролерах AVR. Переривання дозволяють прив'язувати дії до певних подій чи сигналів, що виникають на мікроконтролері. Бібліотека надає функції та макроси для реєстрації та обробки переривань, що дає можливість виконувати дії негайно після виникнення конкретних подій.

Бібліотека #include <avr/io.h> є основною для роботи з мікроконтролерами AVR і містить функції та макроси для доступу до регістрів вводу/виводу (I/O). Вона дозволяє програмісту взаємодіяти з портами, пін-контролерами, налаштуванням портів та іншими апаратними інтерфейсами, доступними на мікроконтролері. Використання цієї бібліотеки є обов'язковим при роботі з будь-яким чіпом AVR.

Бібліотека #include <util/delay.h> надає функції для створення затримок. Вона використовує функцію _delay_ms() для мілісекундних затримок і _delay_us() для мікросекундних. Затримки реалізуються на основі програмного циклу, тому вони не залежать від тактової частоти мікроконтролера, але можуть бути не зовсім точними, якщо тактова частота значно відрізняється від номінальної.

Бібліотека #include <avr/sleep.h> призначена для управління енергоспоживанням мікроконтролера через різні режими сну. Вона дозволяє програмісту переводити мікроконтролер у різні режими сну (наприклад, idle, power-down, standby), що допомагає зменшити споживання енергії під час бездіяльності, особливо в пристроях з обмеженим енергозабезпеченням.

Це стандартна бібліотека #include <stdlib.h>, яка надає основні функції для роботи з пам'яттю, виконання математичних операцій (наприклад, генерація випадкових чисел, перетворення типів), а також для роботи з рядками, масивами та іншими утилітами для маніпуляцій з даними.

Ця стандартна бібліотека #include <string.h> містить функції для роботи з рядками, включаючи функції для копіювання, порівняння, конкатенації

рядків та їх пошуку в масивах.

Бібліотека #include <avr/pgmspace.h>, що дозволяє працювати з програмною пам'яттю (Flash). Вона дає можливість зберігати великі масиви або рядки в постійній пам'яті мікроконтролера замість оперативної пам'яті, що особливо корисно в умовах обмеженої пам'яті. Ця бібліотека надає функції для зчитування даних з програмної пам'яті та створення масивів у цій пам'яті.

Ця бібліотека #include <avr/wdt.h> призначена для роботи з таймером watchdog (WDT), який використовується для забезпечення надійності програми. WDT — це таймер, який можна налаштувати на скидання мікроконтролера, якщо програма зависла або не відповідає вчасно. Це особливо корисно в пристроях, де важливо забезпечити безперебійну роботу системи.

Стандартна бібліотека #include <math.h>, що містить математичні функції для роботи з числами, такі як обчислення коренів, експоненційні функції, тригонометричні функції, логарифми та інші. Вона корисна для виконання складних математичних обчислень.

Ця бібліотека #include <stdint.h> визначає стандартні цілі типи даних з фіксованою довжиною, такі як int8_t, uint16_t тощо. Вона допомагає забезпечити переносимість програм, оскільки ці типи мають чітко визначену кількість бітів, що гарантує стабільність програм на різних платформах.

Бібліотека #include <avr/power.h> для управління споживанням енергії в мікроконтролерах AVR. Вона дозволяє програмно контролювати живлення пристроїв на мікроконтролері, вимикаючи непотрібні модулі або функ ції для зниження споживання енергії. Ця бібліотека також надає функції для управління частотою роботи системи.

Ці бібліотеки в сукупності забезпечують контроль над усіма аспектами функціонування мікроконтролера: від базових операцій вводу/виводу до складних функцій енергозбереження, математичних обчислень та роботи з пам'яттю. Також визначаються піни для підключення кнопок, а також надаються функції для налаштування, очищення та читання бітів на

відповідних пінах. Крім того, бібліотеки можуть містити визначення пінів для зарядки, розрядки або вимірювання аналогових сигналів.

3.6 Лістинг програми

Arduino IDE (інтегроване середовище розробки) — це програмне забезпечення, призначене для створення програм для платформи Arduino. Воно пропонує зручний інтерфейс, який дозволяє писати, компілювати та завантажувати код на платформи Arduino.

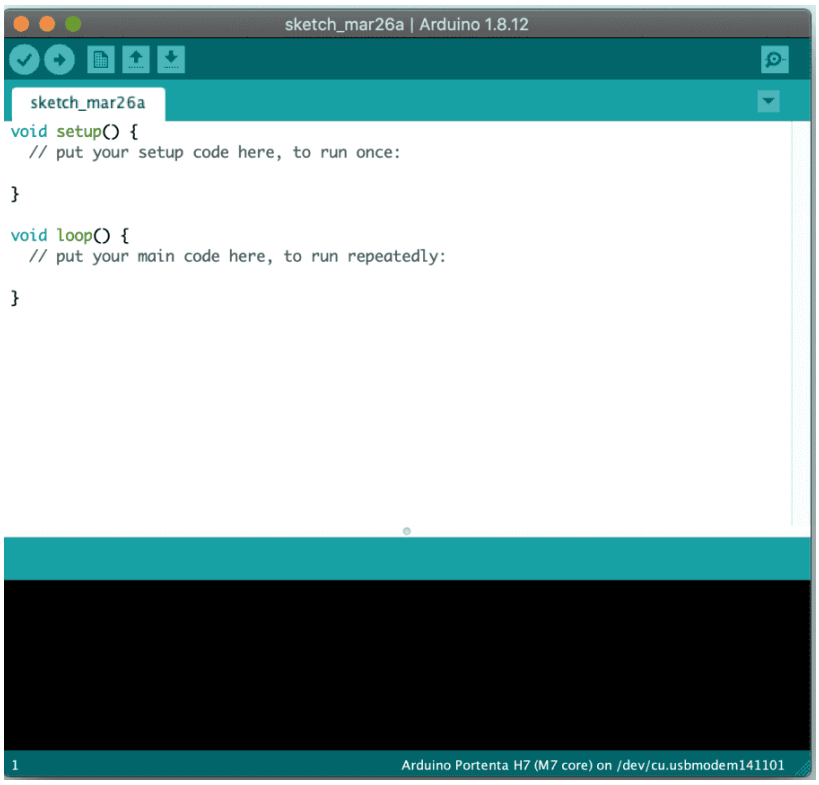


Рисунок 3.11 — Вигляд додатку

Основні характеристики та функціональні можливості Arduino IDE включають:

— написання коду;

— компіляцію;

— завантаження програм;

— монітор порту;

— бібліотеки;

— відлагодження.

Arduino IDE пропонує текстовий редактор, в якому можна створювати програмний код мовою Arduino. Цей редактор підтримує автодоповнення, вирівнювання, підсвічування синтаксису та інші корисні функції, які спрощують процес написання коду.

В IDE вбудована функція компіляції, яка перевіряє синтаксичну коректність коду та перетворює його на машинний код, зрозумілий платформі Arduino. Під час компіляції виявляються можливі помилки, що допомагає уникнути проблем у роботі програми.

Після успішної компіляції програми її можна завантажити на платформу Arduino. Arduino IDE забезпечує простий спосіб підключення Arduino до комп'ютера та завантаження програми через USB-порт. Завантажений код автоматично запускається на платформі.

Arduino IDE також має вбудований монітор порту, який дозволяє відстежувати вивід інформації з платформи Arduino. Це дуже корисна функція для відлагодження та виводу даних під час виконання програми.

IDE постачається з великою кількістю стандартних бібліотек, які містять готовий код для різних функцій. Ці бібліотеки значно спрощують розробку програм і дозволяють швидко виконувати різноманітні завдання, такі як робота з датчиками, керування моторами, взаємодія з бездротовими модулями тощо.

Arduino IDE підтримує прості засоби відлагодження, такі як вивід повідомлень на монітор порту, а також звичайні методи відлагодження мови C/C++, включаючи встановлення точок зупину та моніторинг змінних.

Arduino IDE є безкоштовним і доступним для завантаження з веб-сайту Arduino. Воно підтримує різні версії платформи Arduino та має активну спільноту користувачів, яка надає підтримку та допомогу в розв'язанні проблем і розробці програм. Це робить Arduino IDE популярним інструментом для початківців у світі мікроконтролерів та електроніки.

Для реалізації описаного алгоритму необхідно написати програмний код,

який називається "скетч" (або "скетч Arduino"). Для взаємодії з дисплеєм і передавачем потрібно використовувати певні бібліотеки, які слід підключити до проекту. Основні характеристики та функціональні можливості Arduino IDE включають:

— написання коду;

— компіляцію;

— завантаження програм;

— монітор порту;

— бібліотеки;

— відлагодження.

Arduino IDE пропонує текстовий редактор, в якому можна створювати програмний код мовою Arduino. Цей редактор підтримує автодоповнення, вирівнювання, підсвічування синтаксису та інші корисні функції, які спрощують процес написання коду.

В IDE вбудована функція компіляції, яка перевіряє синтаксичну коректність коду та перетворює його на машинний код, зрозумілий платформі Arduino. Під час компіляції виявляються можливі помилки, що допомагає уникнути проблем у роботі програми.

Після успішної компіляції програми її можна завантажити на платформу Arduino. Arduino IDE забезпечує простий спосіб підключення Arduino до комп'ютера та завантаження програми через USB-порт. Завантажений код автоматично запускається на платформі.

Arduino IDE також має вбудований монітор порту, який дозволяє відстежувати вивід інформації з платформи Arduino. Це дуже корисна функція для відлагодження та виводу даних під час виконання програми.

IDE постачається з великою кількістю стандартних бібліотек, які містять готовий код для різних функцій. Ці бібліотеки значно спрощують розробку програм і дозволяють швидко виконувати різноманітні завдання, такі як робота з датчиками, керування моторами, взаємодія з бездротовими модулями тощо.

Arduino IDE підтримує прості засоби відлагодження, такі як вивід повідомлень на монітор порту, а також звичайні методи відлагодження мови C/C++, включаючи встановлення точок зупину та моніторинг змінних.

Arduino IDE є безкоштовним і доступним для завантаження з веб-сайту Arduino. Воно підтримує різні версії платформи Arduino та має активну спільноту користувачів, яка надає підтримку та допомогу в розв'язанні проблем і розробці програм. Це робить Arduino IDE популярним інструментом для початківців у світі мікроконтролерів та електроніки.

Для реалізації описаного алгоритму необхідно написати програмний код, який називається "скетч" (або "скетч Arduino"). Для взаємодії з дисплеєм і передавачем потрібно використовувати певні бібліотеки, які слід підключити до проекту.

Код програми Arduino складається з двох основних функцій: setup() і loop(). Перед тим, як буде оголошено функцію setup(), потрібні бібліотеки, які використовуватимуться в програмі, збираються в пакет. Функція setup() викликається лише один раз, коли плата Arduino увімкнена або перезавантажена. Ця функція потрібна, навіть якщо вона не виконує жодних операцій, оскільки вона забезпечує початкові налаштування.

Функція loop() — це нескінченний цикл, який виконує команди у своєму тілі. Код у функції loop() працює безперервно, доки Arduino увімкнено. Це дозволяє реалізувати різні функції, такі як зчитування даних із датчиків, керування виходами чи обробка подій у реальному часі. Така структура програми дозволяє Arduino реагувати на зміни середовища та виконувати завдання в реальному часі, що робить його потужним інструментом для створення інтерактивних проектів.

```
void lcd_data(unsigned char temp1) {
  lcd.write(temp1);
  switch(temp1) {
    case LCD_CHAR_DIODE1: {
      uart_putc('>'); uart_putc('|'); break;
    }
    case LCD_CHAR_DIODE2: {
```

```
      uart_putc('|'); uart_putc('<'); break;
    }
    case LCD_CHAR_CAP: {
      uart_putc('|'); uart_putc('|'); break;
    }
    case LCD_CHAR_RESIS1:
    case LCD_CHAR_RESIS2: {
      uart_putc('R'); break;
    }
    case LCD_CHAR_U: {    // micro
      uart_putc('u');   // ASCII u
      break;
    }
    case LCD_CHAR_OMEGA: {  // Omega
      uart_putc('o');           // "ohm"
      uart_putc('h');
      uart_putc('m'); break;
    }
    default: {
      uart_putc(temp1);
    }
  }
}
```

В лістингу наведено функцію lcd_data, яка приймає один параметр типу unsigned char (названий temp1) і виконує дві основні задачі.

Виведення даних на LCD-екран.

```
cpp
VerifyOpen In EditorEditCopy code
1lcd.write(temp1);
```

Цей рядок коду використовує метод write об'єкта lcd для відображення символу temp1 на підключеному LCD-дисплеї. Це стандартний спосіб передачі символів на LCD-екран за допомогою бібліотеки, сумісної з Arduino.

Передача відповідного символу через UART: З використанням оператора switch, в залежності від значення temp1, виконується різна логіка. У кожному випадку символ або послідовність символів надсилаються через UART (універсальний асинхронний приймач/передавач) за допомогою функції uart_putc().

Оглянемо функції ,які описані в switch:

— в case LCD_CHAR_DIODE1, якщо temp1 дорівнює LCD_CHAR_ DIODE1, UART виведе символи > та |, що може представляти графічне зображення діода у текстовому форматі;

— в case LCD_CHAR_DIODE2,якщо temp1 дорівнює LCD_CHAR_D IODE2, UART виведе символи | та <, що є ще одним графічним зображенням діода, але з іншим напрямком;

— в case LCD_CHAR_CAP, якщо temp1 дорівнює LCD_CHAR_CAP, UART виведе два символи | і |, що може позначати графічне представлення конденсатора;

— в case LCD_CHAR_RESIS1: і case LCD_CHAR_RESIS2,якщо temp1 дорівнює LCD_CHAR_RESIS1 або LCD_CHAR_RESIS2, UART виведе символ R, який є загальноприйнятим позначенням резистора;

— в case LCD_CHAR_U, якщо temp1 дорівнює LCD_CHAR_U, UART виведе символ u (ASCII), що може означати мікро- (наприклад, мікрофаради в ємностях);

— в case LCD_CHAR_OMEGA, якщо temp1 дорівнює LCD_CHAR_OMEGA, UART виведе послідовність символів o, h, m, яка представляє слово "ohm" (Ом) — одиницю вимірювання опору;

— в default, якщо temp1 не відповідає жодному з вищезазначених випадків, функція передає значення temp1 без змін через UART.

Функція lcd_data виконує дві основні дії:

— відображає символ на LCD-дисплеї;

— передає текстове представлення цього символу або його графічної інтерпретації через UART, що дозволяє відобразити або використати його на іншому пристрої чи програмі, яка приймає дані з UART.

Це корисно для дублювання інформації: графічна частина відображається на LCD, а текстова — через UART, наприклад, для віддаленого моніторингу. Таким чином, користувач може отримувати інформацію в зручному для нього форматі, що підвищує ефективність роботи з даними.

4 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

4.1 Моделювання

ESR-тестер електронних компонентів — це універсальний пристрій, призначений для вимірювання, тестування та аналізу характеристик різноманітних електронних елементів, таких як резистори, конденсатори, діоди, транзистори, тиристори та інші напівпровідникові пристрої. Цей інструмент дозволяє зручно і точно визначати параметри компонентів, що є важливим як для професіоналів, так і для ентузіастів електроніки.

Призначення та функції ESR-тестера

ESR-тестер відіграє важливу роль у швидкій діагностиці стану електронних компонентів та визначенні їх основних характеристик. Він дозволяє:

— вимірювати параметри резисторів, такі як опір, розрядний струм, а також визначати тип резистора (постійний чи змінний);

— перевіряти конденсатори, включаючи вимірювання ємності, витоку струму та еквівалентного серійного опору (ESR);

— тестувати напівпровідникові пристрої, такі як діоди, транзистори та тиристори, з визначенням параметрів, наприклад, коефіцієнта підсилення транзисторів або падіння напруги на переході діода;

— визначати працездатність компонентів, наприклад, перевіряти їх на наявність короткого замикання або обриву.

Завдяки цим функціям, ESR-тестери є незамінними в багатьох сферах:

— для радіоаматорів, які займаються створенням і налаштуванням електронних схем;

— для інженерів-електронників, які проєктують, тестують і оптимізують електронні пристрої;

— для техніків і спеціалістів з ремонту, які проводять діагностику несправностей і відновлення роботи електронного обладнання;

— для навчальних цілей, де використання тестера допомагає студентам зрозуміти принципи роботи електронних компонентів.

Окрім високої точності вимірювань, ESR-тестери пропонують користувачеві зручний інтерфейс, інтуїтивно зрозуміле управління та часто підтримують функції автоматичного розпізнавання компонентів, що значно спрощує процес роботи.

Сучасні ESR-тестери також можуть інтегруватися з комп'ютерами або мобільними пристроями, що дозволяє зберігати результати вимірювань, створювати звіти та аналізувати дані в реальному часі. Ця універсальність робить їх важливим інструментом у будь-якому сучасному електронному середовищі.

На рисунку 4.1 зображений ESR-тестер, загальні виводи, які використовуються для вимірювання параметрів це два роз’єма для щупів, також є клемна колодка з трьох контактів, для виміру компонентів, які на 3 вивода, а також площадка для вимірювання smd елементів, зображено на рисунку 4.2.

Рисунок 4.1 — Загальний вигляд приладу

Також на корпусі розміщена кнопка. Коли користувач вставляє елемент в один із роз’ємів, натискаючи кнопку, на екрані висвічується, що за елемент був вставлений і його параметри приклад зображений на рисунку 4.3

Рисунок 4.2 — Роз’єми для щупів, клемна колодка, площадка для smd елементів

Рисунок 4.3 — Приклад використання приладу

4.2 Тестування

Прилад автоматично визначає тип компонента і параметр який потрібно вимірювати приклади наведенні на рисунках 4.4, 4.5, 4.6, 4.7. В залежності від елемента та кількості контактів можна використовувати різні площадки, можна використати площадку для вимірювання smd елементів, так і клемну колодку, або для щупів роз’єми.

Вимірювання параметрів різних елементів таких як:

— резисторів;

— конденсаторів;

— транзисторів;

— діодів.

Рисунок 4.4 — Елемент діод

Рисунок 4.5 — Елемент PNP транзистор

Рисунок 4.6 — Конденсатор

Рисунок 4.7 — Резистор

Для першого експерименту було використанні декілька видів резисторів, типи резисторів зображені в таблиці 4.1

Таблиця 4.1 — Резистори і їх параметри

Резистор	Номінальне значення резистора	Результат вимірювання
МЛТ 2	10кОм	10168Ом (10кОм)
резистор	1кОм	974Ом (0,974 кОм)
резистор	2,7 кОм	2701 Ом (2,7кОм)

Рисунок 4.5 — Резистор МЛТ 2

Рисунок 4.6 — Резистор

Після проведення експериментів було виконано вимірювання опору резистора, параметри резистора МЛТ 2 номінальне значення, яке зазначене на резисторі 10 кОм, при вимірювані параметра даного резистора прилад показав 10 кОм. Наступний резистор номінальне значення 1кОм, прилад виміряв 0,974 кОм.

Для наступного експерименту було залучено декілька видів конденсаторів: smd конденсатори, оксидно-електролітичний конденсатор та інші.

Використанні наступні smd конденсатори зображенні на рисунку 4.7. оксидно-електролітичний конденсатор, який використано в експерименті і зображений на рисунку 4.8. Та інші конденсатори, які використані в дослідженні зображені на рисунку 4.9

На рисунку зображено оксидно-електролітичний конденсатор номінальне значення 1 мкФ прилад виміряв 1,10 мкФ, наступний конденсатор номінальне значення 10 нФ прилад виміряв 9,73 нФ.

Конденсатори, які вимірювалися наступними. Перший конденсатор 1000 мкФ прилад виміряв 959,51 мкФ. Другий конденсатор 2200 мкФ, прилад виміряв 2110,58 мкФ.

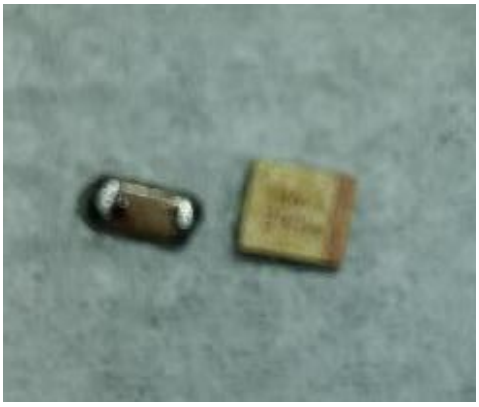


Рисунок 4.7 — smd конденсатори, які використовувалися в експерименті

Рисунок 4.8 — оксидно-електролітичний конденсатор, який використовувався в дослідженні

Рисунок 4.9 —конденсатори, які використовувався в дослідженні

Результати вимірювання параметрів конденсаторів показано на рисунку 4.10, рисунку 4.11 та рисунку 4.12.

Рисунок 4.10— Вимірювання ємності конденсаторів

Рисунок 4.11— Вимірювання ємності конденсаторів

Рисунок 4.12— Вимірювання ємності smd конденсаторів

Наступними конденсаторами були smd конденсатори, перший конденсатор ємністю 150 нФ прилад виміряв 151,72 нФ, другий конденсатор ємністю 10 мкФ, а прилад виміряв 9,5 мкФ.

В наступному дослідженні використовуються транзистори, види транзисторів зображені на рисунку 4.13

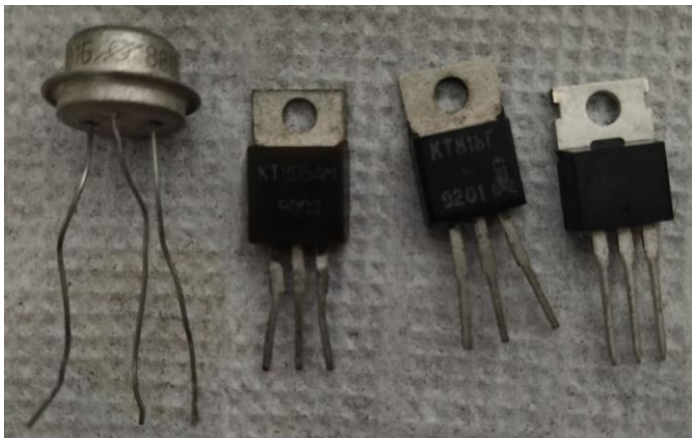


Рисунок 4.13 — Транзистори, які використанні для дослідження

Виміри транзисторів на приладі зображенні на рисунку 4.14

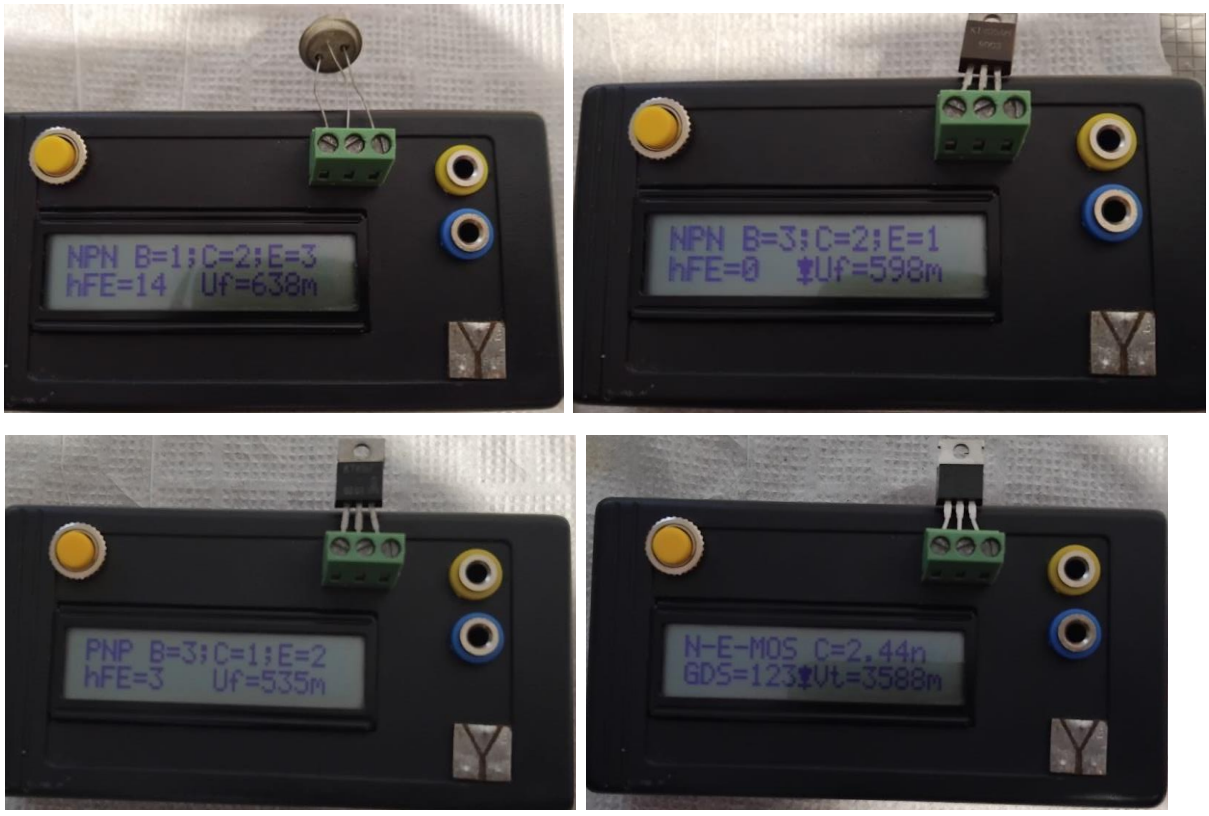


Рисунок 4.14 —Вимір транзисторів

В даному досліді були використані NPN,PNP, NEMOS, прилад автоматично визначить тип транзистора, також вимірює коефіцієнт підсилення струму, та його ємність.

Визначення приладом діода, та показання його параметрів на рисунку 4.16, на рисунку 4.15 показано діоди, які використовувалися в експерименті.

Вимірювання діодів відбувається в мілівольтах, прилад визначив, що це діод та виміряв його ємність.

Переваги застосування ESR-тестера:

— прилад автоматично виконує тестування без необхідності складного налаштування;

— результати вимірювань відображаються на екрані всього за кілька секунд;

— завдяки компактним розмірам, ECR-тестер зручно використовувати в будь-яких умовах;

— один пристрій підтримує тестування різних типів електронних компонентів;

— вартість ECR-тестерів робить їх привабливими для широкого кола користувачів.

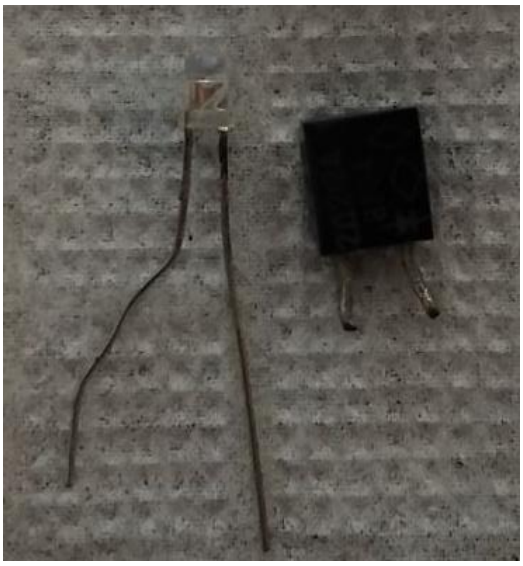


Рисунок 4.15 — Діоди

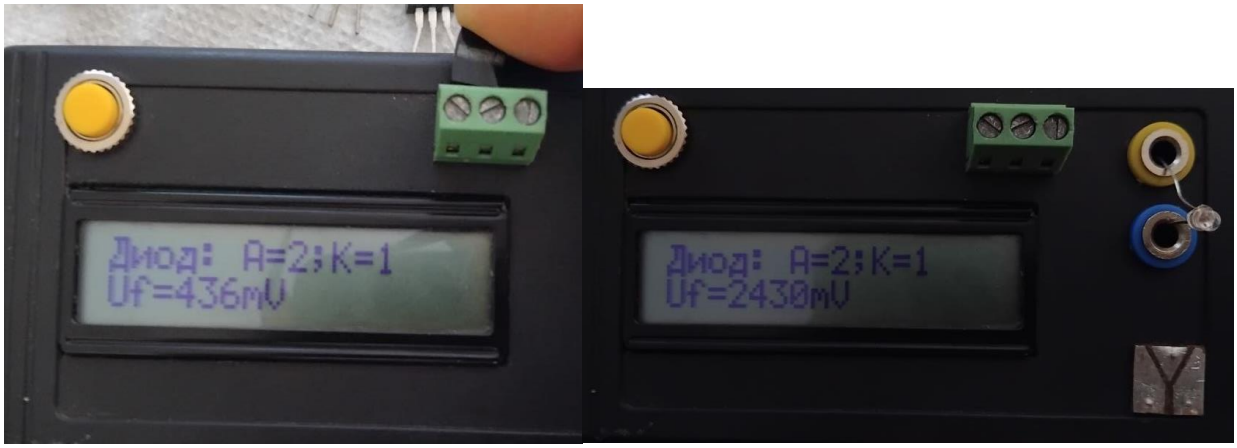


Рисунок 4.16 — Визначення елемента та його параметри

ESR-тестери вимірюють основні характеристики такі як:

— вимірювання резисторів у діапазоні від 0,1 Ом до кількох МОм;

— аналіз конденсаторів із діапазоном ємності від кількох пФ до десятків тисяч мкФ;

— Визначення ESR (еквівалентного серійного опору) з високою точністю для оцінки стану конденсаторів;

— відображення параметрів на РК- або OLED-дисплеї у зручному для користувача форматі;

— живлення від акумулятора чи батарей із можливістю зарядки через USB.

ECR-тестери знаходять широке використання:

— у діагностиці й ремонті електронного обладнання;

— під час монтажу та налагодження електронних схем;

— у навчальних цілях для студентів і радіоаматорів;

— для перевірки стану компонентів у застарілих пристроях.

ECR-тестери є універсальними й надійними інструментами для тестування та вимірювання параметрів електронних компонентів. Вони значно полегшують процес ремонту, налаштування та розробки електронних пристроїв.

4.3 Інструкція користувачу

Підготовка:

— переконайтеся, що прилад забезпечений живленням. Перевірте, чи підключена кабель типу type-C або акамулятор і чи працює воно справно;

— упевніться, що всі компоненти підключені правильно та надійно;

— перевірте з’єднання між Arduino, LCD-дисплеєм, кнопкою;

— увімкніть пристрій.

Процес вимірювання:

— вставте в клемну колодку елемент (резистор, конденсатор або діод), який потрібно виміряти;

— натисніть кнопку і прилад автоматично визначить, що вставив користувач і виміряє;

— дочекайтеся завершення вимірювання. Результати з’являться на рідкокристалічному дисплеї.

Отримання результатів:

— значення відобразиться на LCD-дисплеї. Зверніть увагу на отриманий результат.

Завершення роботи:

— вимкніть пристрій, витягнувши кабель;

— зберігайте пристрій у сухому та безпечному місці.

Примітки:

— використовуйте прилад відповідно до інструкцій та дотримуйтесь правил безпеки;

— перед вимірюванням обов’язково розряджайте конденсатор;

— у разі виникнення несправностей або неочікуваних результатів зверніться до посібника користувача або проконсультуйтеся з фахівцем.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Комерційний та технологічний аудит науково-технічної розробки

Метою даного розділу є проведення технологічного аудиту, в даному випадку нового методу та мікропроцесорного засобу вимірювання параметрів електронних компонентів. Особливістю розробки є покращення приладу для зручного вимірювання характеристик електронних компонентів (опору, ємності, ESR), з використанням доступного апаратного та програмного забезпечення.

Аналогом може бути LCR Meter DER EE DE-5000 – 180 $ (7200 грн.) або мультиметр UNI-T UT61E з функціями вимірювання ємності та опору – 100 $ або 4000 грн.

Для проведення комерційного та технологічного аудиту залучають не менше 3-х незалежних експертів. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, у відповідності із табл. 4.1.

Таблиця 5.1 — Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за 5-ти бальною шкалою)					
Критерій	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність в реальних умовах
2	Багато аналогів на малому ринку	Ринкові п Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку

Продовження табл. 5.1

Ринкові переваги					
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практик на здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так із комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження табл. 5.1

11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Усі дані по кожному параметру занесено в таблиці 5.2

Таблиця 5.2 — Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	ПІБ експертів		
	Експерт 1	Експерт 2	Експерт 3
	Бали		
Технічна здійсненність концепції	3	4	3
Наявність аналогів на ринку	4	3	4
Цінова політика	4	4	4
Технічні та споживчі властивості виробу	3	3	3
Експлуатаційні витрати	4	4	3
Ринок збуту	3	4	3
Конкурентоспроможність	4	4	4
Фахівці з технічної і комерційної реалізації	2	3	2
Фінансування	2	2	4
Матеріально-технічна база	3	3	3
Термін реалізації ідеї	4	4	4
Супровідна документація	4	4	4
Сума	40	42	41
Середньоарифметична сума балів	(40+42+41) / 3 = 41		

За даними таблиці 5.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 5.3.

Таблиця 5.3 — Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

Як видно з таблиці, рівень комерційного потенціалу розроблюваного нового виробу є високим, завдяки створенню доступних рішень для точного вимірювання електронних компонентів у лабораторних і побутових умовах, що актуально для інженерів, студентів та ремонтників електроніки.

5.2 Прогнозування витрат на виконання науково-дослідної (дослідно-конструкторської) роботи

5.2.1 Основна заробітна плата розробників, яка розраховується за формулою:

$$З_о = \frac{M}{T_р} \cdot t, \quad (5.1)$$

де М – місячний посадовий оклад конкретного розробника (дослідника), грн.;

$T_р$ – число робочих днів в місяці, 23 днів;

t – число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 5.1.

Таблиця 5.1 – Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	28000	1217,39	28	34086,957
Інженер	26000	1130,43	28	31652,174
Всього				65739,13

5.2.2 Витрати на основну заробітну плату робітників ($З_р$) розраховуються на основі норм часу, які необхідні для виконання даної роботи, розраховуються за формулою:

$$З_р = \sum_{1}^{n} t_i \cdot С_i \cdot К_с , \quad (5.2)$$

де t_i – норма часу (трудомісткість) на виконання конкретної роботи, годин;

n – число робіт по видах та розрядах;

$К_с$ – коефіцієнт співвідношень, який установлений в даний час Генеральною тарифною угодою між Урядом України і профспілками;

Сі – погодинна тарифна ставка робітника відповідного розряду, який виконує відповідну роботу, грн./год.

$С_i$ визначається за формулою:

$$С_i = \frac{М_м \cdot К_i}{Т_р \cdot Т_{з\,м}}, \quad (5.3)$$

де $М_н$ – мінімальна місячна оплата праці, грн., 8000 грн. у 2024 р.

$К_i$ – тарифний коефіцієнт робітника відповідного розряду;

$Т_р$ – число робочих днів в місяці, $Т_р$ = 23 дні;

$Т_{зм}$ – тривалість зміни, $Т_{зм}$ = 8 годин.

Погодинна тарифна ставка згідно чинного законодавства у грудні 2024 року = 48 грн./год.

Розрахунки заносимо до табл. 5.5.

Таблиця 5.5 – Витрати на основну заробітну плату робітників

Найменування робіт	Трудомісткість, год.	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн.	Величина оплати на робітника грн.
Заготівельні	1,5	3	1,35	64,8	97,2
Слюсарно-збиральні	5	3	1,35	64,8	324
Налагоджувальні та контрольні	1	5	1,7	81,6	81,6
Всього					502,80

5.2.2 Додаткову заробітну плату прийнято розраховувати як 10% від основної заробітної плати розробників та робітників:

$$З_д = (З_{о.роз} + З_{о.роб}) \cdot 10\% /100\% \quad (5.2)$$

$$З_д = (65739,13 + 502,80) \cdot 10\%/100\% = 6624,19 \text{ (грн.)}$$

5.2.3 Згідно діючого законодавства нарахування на заробітну плату складають 22 % від суми основної та додаткової заробітної плати.

$$Н_з = (З_{о.роз} + З_{о.роб} + З_{д.}) \cdot 22\%/100\% \quad (5.3)$$

$$Н_з = (65739,13 + 502,80 + 6624,19) \cdot 22\%/100\ \% = 15919,93 \text{ (грн.)}$$

5.2.4. Оскільки для розроблювального програмного продукту не потрібно витрачати матеріали та комплектуючи, то витрати на матеріали і комплектуючі дорівнюють нулю.

5.2.5 Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді амортизація обладнання, що використовувалась для розробки розраховується за формулою:

$$A = \frac{Ц}{Тв} \cdot \frac{t_{вик}}{12} \text{ [грн.].} \quad (5.4)$$

де Ц – балансова вартість обладнання, грн.;

Т – термін корисного використання обладнання згідно податкового законодавства, років

$t_{вик}$ – термін використання під час розробки, місяців

Розрахуємо, для прикладу, амортизаційні витрати на комп'ютер балансова вартість якого становить 25000 грн., термін його корисного використання згідно податкового законодавства – 2 роки, а термін його фактичного використання – 1,217 міс.

$$А_{обл} = \frac{25000}{2} \times \frac{1{,}217}{12} = 1268{,}12 \text{ грн.}$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до таблиці 5.6.

Амортизація обладнання, що використовувалось робітниками, розраховується аналогічно, результати розрахунків зведено в таблицю 5.7 і враховуються при розрахунку виробничої собівартості виробу.

Так як вартість ліцензійної ОС та спеціалізованих ліцензійних нематеріальних ресурсів, а також спеціалізованого обладнання менше 20000 грн, то даний нематеріальний актив не амортизується, а його вартість

включається у вартість розробки повністю, Вспец. обл.. = 11000 грн.

Таблиця 5.6 — Амортизаційні відрахування матеріальних і нематеріальних ресурсів для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортиза-ційні відрахуван-ня, грн.
Комп'ютер та комп'ютерна периферія	25000	2	1,217	1268,116
Офісне обладнення (меблі)	22000	4	1,217	557,971
Приміщення	850000	20	1,217	4455,956
Всього				6282,04

Таблиця 5.7 — Амортизаційні відрахування матеріальних і нематеріальних ресурсів для робітників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання		Аморти-заційні відрахування, грн.
			год.	міс.	
Комп'ютер	25000	2	1	0,0054	5,6612
Спеціалізоване обладнання (меблі)	22000	4	1	0,0054	2,4909
Приміщення	850000	20	7,5	0,0408	144,3614
Всього					152,5136

5.2.6. Витрати на комплектуючі, що були використані на виготовлення розраховуються за формулою

$$K = \sum_{i=1}^{n} H_i \cdot Ц_i \cdot K_i , \qquad (4.5)$$

де H_i — кількість комплектуючих i-го виду, шт.,

$Ц_i$ — роздрібна ціна комплектуючих i-го виду, грн.,

K_i — коефіцієнт транспортних витрат, K_i=1,1,

n — кількість видів матеріалів.

Проведені розрахунки зводимо до таблиці 4.8 без врахування транспортних витрат.

Таблиця 5.8 – Витрати на комплектуючі

Найменування комплектуючих	Кількість	Ціна за штуку, грн.
Arduino Nano	1	300
Рідкокристалічний дисплей LCM1602 IIC	1	200
Блок живлення Type-C (5 В)	1	250
Резистои 8 штук	8	30
Конденсатор	1	5
Всього		**995,00**

Витрати на комплектуючі, що були використані на розробку з врахуванням транспортних втрат:

$$Н = 995 \cdot 1{,}1 = 1094{,}50 \text{ (грн.)}$$

5.2.7 Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. При цьому тарифи на розподіл електроенергії у різних постачальників (енергорозподільних компаній), будуть різними. Крім того, розмір тарифу залежить від класу напруги (1-й або 2-й клас). Тарифи на розподіл електроенергії для всіх енергорозподільних компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п, \quad (5.6)$$

де В – вартість 1 кВт-години електроенергії для 1 класу підприємства з ПДВ в 2024 році для Вінницької області за даними Енера-Вінниця, В = (5635,47/1000)*1,2 = 6,76 грн./кВт;

П — встановлена середня потужність обладнання, кВт. П = 0,4 кВт;

Ф — фактична кількість годин роботи обладнання, годин.

$К_п$ — коефіцієнт використання потужності, $К_п$ = 0,8.

$$В_е = 0,8 \cdot 0,4 \cdot 8 \cdot 28 \cdot 10,42 + 0,8 \cdot 0,4 \cdot 2,0 \cdot 10,42 = 746,9056 + 6,669 = 753,57 \text{ (грн.)}$$

5.2.8 Інші витрати та загальновиробничі витрати.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.6)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

$$І_в = (65739,13 + 301,68) \cdot 60\% / 100\% = 39745,16 \text{ (грн.)}$$

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$H_{нзв} = (З_о + З_р) \cdot \frac{H_{нзв}}{100\%}, \quad (4.7)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$$Н_{нзв} = (65739,13 + 502,80) \cdot 110\% / 100\ \% = 72866 \text{ (грн.)}$$

5.2.9 Витрати на проведення науково-дослідної роботи.

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

$$В_{заг} = 65739,13+502,80+6624,19+15919,93+6282,04+ 152,5136+11000+$$
$$+1094,50+753,57+39745,16+72866 = 220679,97 \text{ грн.}$$

5.2.11 Розрахунок загальних витрат на науково-дослідну (науково-технічну) роботу та оформлення її результатів.

Загальні витрати на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються ЗВ, визначається за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} \text{ (грн)}, \quad (5.8)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η=0,1; технічного проектування, то η=0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7; впровадження, то η=0,9. Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка:

ЗВ = 220679,97 / 0,5 = 441360 грн.

5.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнювальним позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Для того, щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від можливого впровадження науково-технічної розробки необхідно:

— вказати, з якого часу можуть бути впроваджені результати науково-технічної розробки;

— зазначити, протягом скількох років після впровадження цієї науково-технічної розробки очікуються основні позитивні результати для потенційного інвестора (наприклад, протягом 3-х років після її впровадження);

— кількісно оцінити величину існуючого та майбутнього попиту на цю або аналогічні чи подібні науково-технічні розробки та назвати основних суб’єктів (зацікавлених осіб) цього попиту;

— визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями.

При розрахунку економічної ефективності потрібно обов’язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій

до отримання прибутку минає чимало часу. При оцінюванні ефективності інноваційних проектів передбачається розрахунок таких важливих показників:

— абсолютного економічного ефекту (чистого дисконтованого доходу);

— внутрішньої економічної дохідності (внутрішньої норми дохідності);

— терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково-технічних розробок, розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором можна об’єднати, враховуючи визначені ситуації з відповідними умовами.

5.3.1 Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (5.9)$$

де $\pm\Delta Ц_о$ – зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу;

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки;

$Ц_о$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_о = Ц_б \pm \Delta Ц_о$;

Цб – вартість програмного продукту у році до впровадження результатів розробки;

ΔN – збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт λ = 0,8333.

р – коефіцієнт, який враховує рентабельність продукту;

ϑ – ставка податку на прибуток, у 2024 році ϑ =18%.

Припустимо, що при прогнозованій ціні 2300 грн. за одиницю виробу, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на 100 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року – на 3000 шт., протягом другого року – на 2500 шт., протягом третього року на 2000 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

$$\Delta П_1 = (0*100 + (2300 + 100)*3000)* 0,8333* 0,2) * (1 - 0,18) = 942999,962 \text{ грн.}$$

$$\Delta П_2 = (0*100 + (2300 + 100)*(3000+2500)* 0,8333* 0,2) * (1 - 0,18) = 1803999,928 \text{ грн.}$$

$$\Delta П_3 = (0*100 + (2300 + 100)*(3000+2500+2000)* 0,8333* 0,2) * (1 - 0,18) = 2459999,902 \text{ грн.}$$

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 5206999,79 грн.

5.3.2 Розраховуємо приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.10)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної (науково-технічної) роботи, грн;

Т – період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ = 0,05…0,15;

t – період часу (в роках).

Збільшення прибутку ми отримаємо починаючи з першого року:

ПП =

(942999,962/(1+0,1)[1])+(1803999,928/(1+0,1)[2])+(2459999,902/(1+0,1)[3]) =

857272,69 + 1490909,03 + 1848234,34 = 4196416,061 грн.

Далі розраховують величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} * ЗВ, \quad (5.11)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}$=2...5, але може бути і більшим;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

PV = 2 * 441360 = грн.

Тоді абсолютний економічний ефект $Е_{абс}$ або чистий приведений дохід (NPV, Net Present Value) для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$Е_{абс} = ПП – PV, \quad (5.12)$$

$$Е_{абс} = 4196416,061 – 441360 = 3755056,13 \text{ грн.}$$

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів даної науково-дослідної (науково-технічної) роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність або показник внутрішньої норми дохідності (IRR, Internal Rate of Return) вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього використаємо формулу:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (5.13)$$

де $T_ж$ – життєвий цикл наукової розробки, роки.

$$Е_в = \sqrt[3]{(1 + 3755056,13/ 441360} - 1 = 1,119$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \tag{5.14}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні d = (0,09…0,14);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

$$\tau_{\min} = 0{,}14 + 0{,}05 = 0{,}19.$$

Так як $E_в > \tau_{min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в}, \tag{5.15}$$

$$T_{ок} = 1 / 1{,}119 = 0{,}89 \text{ р.}$$

Оскільки $T_{ок}$ < 3-х років, а саме термін окупності рівний 0,89 роки, то фінансування даної наукової розробки є доцільним.

Висновки до розділу: економічна частина даної роботи містить розрахунок витрат на розробку продукту, сума яких складає 441360 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений продукт за ціною дешевший за аналог і є висококонкурентоспроможним. Період окупності складе близько 0,89 роки.

ВИСНОВКИ

В результаті виконання магістерської роботи удосконалено методи та мікропроцесорні засоби, які забезпечують точні вимірювання параметрів електронних компонентів.

У першому розділі подано огляд та аналіз поточних методів і інструментів для вимірювання інженерних характеристик. Це полегшило визначення ключових цілей, методів вимірювання та вибір найбільш відповідних інструментів реалізації.

Друга частина присвячена методам, алгоритмам і методикам вимірювання ефективності виробництва. Методи та процедури вимірювань були розроблені для забезпечення ефективності та точності вимірювань.

На третьому етапі було виконано апаратне та системне проектування, а також розробка програмного забезпечення. Дизайн, вибір електроніки та електрична схема були розроблені для забезпечення продуктивності системи. Розроблені алгоритми та бібліотеки сприяли плавному процесу застосування, а в посібнику користувача детально пояснюється, як користуватися пристроєм.

Останній розділ проекту присвячено економічній частині, де було проведено економіко-технічний аналіз, який визначив фінансування наукових досліджень, а також розраховано техніко-економічну доцільність розробки на основі того, як це можна отримати в ринок. .

Таким чином, результати показують, що розроблений метод вимірювання є здійсненним та ефективним, який може бути широко використаний для вимірювання електронних компонентів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Резистор [Електронний ресурс]:Режим доступу: https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B7%D0%B8%D1%81%D1%82%D0%BE%D1%80 Дата доступу 01.11.2024

2. Конденсатори [Електронний ресурс]:Режим доступу: https://uk.wikipedia.org/wiki/%D0%95%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%B8%D1%87%D0%BD%D0%B8%D0%B9_%D0%BA%D0%BE%D0%BD%D0%B4%D0%B5%D0%BD%D1%81%D0%B0%D1%82%D0%BE%D1%80 Дата доступу 01.11.2024

2. Основи мікроелектроніки : навч. посіб. до лаб. практикуму / М. Є. Лещенко, І. К. Васильєва, О

3. М. Замірець, В. Є. Овчаренко. – Х. : Нац. аерокосм. ун-т "Харк. авіац. ін-т", 2010. – Ч. 1. – 64 с.)

4. Вимірювання опорів методом моста [Електронний ресурс]:Режим доступу: https://physics.nmu.org.ua/ua/To_students/Day_mode_of_study/Methodical_instructions_to_laboratory_works/Electricity_and_magnetism/3.30.pdf
Дата доступу 01.11.2024

5. Вимірювання опорів [Електронний ресурс]:Режим доступу: https://elib.lntu.edu.ua/sites/default/files/elib_upload/%D0%B3%D0%BE%D1%82%D0%BE%D0%B2%D0%B8%D0%B9%20%D0%91%D0%B0%D1%85%D0%BE%D0%B2%D1%81%D1%8C%D0%BA%D0%B8%D0%B9/page29.html
Дата доступу 01.11.2024

6. Measurement, Instrumentation, and Sensors Handbook Electromagnetic: Optical,Radiation, Chemical, and Biomedical Measurement / Edited By John G. Webster, Halit Eren - 2nd Edition. - Boca Raton, CRC Press, 2017. - 1921 p.

7. Imants Matiss / Capacitance Measurement Techniques – a New Challenge. - LAP LAMBERT Academic Publishing, 2015. - 224 p.

8. Kutz M. (Ed.) Handbook of Measurement in Science and Engineering. Volume 3. – John Wiley & Sons, Inc., Canada, 2016. — 818 p.

9. Ferri G., Stornelli V., Barile G. Electronic Interfaces for Differential Capacitive Sensors. - Gistrup: River Publishers, 2020. — 150 p.

10. Low Level Measurements Handbook. Precision DC Current, Voltage, and Resistance. - 7th edition. — New York: Textronix, 2014. — 244 p.

11. Pitard F.R. Theory of Sampling and Sampling Practice Boca Raton: CRC Press, 2019. — 727

12. Кохц Вимірювання, керування та регулювання за допомогою PIC мікроконтролерів / Дітер Кохц: МК Прес, 2007. - 299 с.

13. Яценков В.С. Мікроконтролери Microchip PIC із вбудованим малопотужним радіопередавачем / Яценков В.С. : Гор.лінія-Телеком, 2006. – 344 с.

14. Яценков В.С. Мікроконтролери Microchip з апаратною підтримкою USB/Яценков В.С.: Гор.лінія-телеком, 2008. - 400с.

15. Брей Застосування мікроконтролерів PIC18. Архітектура, програмування, побудова інтерфейсів із застосуванням С та асемблера / Барі Брей: МК-Прес, 2008 - 576 с.

17. Вілмсхерст Т. – Розробка вбудованих систем за допомогою мікроконтролерів PIC. Принципи та практичні приклади / Вілмсхерст Т.: МК-Прес, 2008. - 544 с.

18. Тимофєєв В. mpasm Як правильно оформлювати програми на асемблері для PIC-контролерів / Тимофєєв В. – Власне, 2010. - 40 с.

19. Capacitance measurement: Understand and use the right technique to dramatically improve results [Електронний ресурс] - Режим доступу: https://www.planetanalog.com/capacitance-measurement-understand-and-use-theright-technique-to-dramatically-improve-results/(дата звернення: 20.03.2023)

20. Raspberry Pi Pico. URL: https://www.tomshardware.com/reviews/raspberry-pi-pico-w (дата звернення: 20.03.2023).

21. Wemos D1 mini. URL: https://www.learnrobotics.org/blog/which-controller-should-i-use-wemos-d1-mini-or-nodemcu/ (дата звернення: 21.03.2023).

22. STM32. URL: https://www.classcentral.com/course/udemy-stm32-introduction-to-stm32-stm-electronics-55594 (дата звернення: 22.03.2023).

ДОДАТОК А

Міністерство освіти та науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ ВНТУ

д.т.н., проф.

_____О. Д. Азаров

“__” _____ 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

«Метод та мікропроцесорні засоби вимірювання параметрів електронних компонентів.»

08-54.МКР.013.00.000ТЗ

Науковий керівник

к.т.н., доц. каф. ОТ

_____ Богомолов С.В.

виконав:

магістрант 2 курсу,

_____ Мушинський В.Є.

Вінниця 2024

1 Підстава виконання магістерської кваліфікаційної роботи

1.1 Актуальність досліджень у цій сфері важко переоцінити, особливо в контексті покращення продуктивності електронних компонентів, таких як резистори та конденсатори. Сучасні напрямки досліджень зосереджені на вдосконаленні методів вимірювання, підвищенні точності та швидкості, а також на розробці нових матеріалів. Технічні рішення, зокрема в сенсорних технологіях і мікропроцесорних додатках, є критично важливими для еволюції електронних пристроїв, що відповідають зростаючим вимогам до ефективності та функціональності.

1.2 Аналіз існуючих аналогів виявляє обмеження в їх продуктивності. Пропозиції цього дослідження спрямовані на усунення цих недоліків через впровадження нових методів і алгоритмів, що перевершують поточні рішення. Параметри запропонованих рішень будуть порівнюватися, демонструючи їх переваги, такі як підвищена точність вимірювання. Формулювання завдань розробки включає визначення конкретних проблем у вимірюванні електронних компонентів та встановлення чітких цілей.

1.3 Наказ про затвердження теми МКР

2 Мета і призначенням МКР

2.1 Метою роботи є огляж методів і мікропроцесорних засобів вимірювання параметрів електронних компонентів. Основною метою є створення пристрою, який може точно вимірювати ці параметри, тим самим підвищуючи надійність і функціональність електронних систем.

2.2 Призначення розробки полягає в тому, щоб надати інженерам, студентам і технікам ефективні рішення для вимірювання електронних компонентів, таких як резистори та конденсатори та інші, що зрештою сприятиме покращенню дизайну та оцінки продукту в різних додатках. Це спрямовано на усунення існуючих прогалин у методах вимірювання,

гарантуючи, що нові рішення відповідають зростаючим вимогам до точності та ефективності в галузі електроніки.

3 Вихідні дані для виконання МКР

Технічний опис мікропроцесорних платформ Arduino, технічна документація на електронні компоненти.

4 Вимоги до виконання МКР

МКР повинна задовольняти таку вимогу — реалізувати можливості легкість у використанні та зменшення похибок вимірювань параметрів компонентів .

5 Етапи МКР та очікувані результати

Етапи роботи та очікувані результати приведено в табл. А.1.

6 Матеріали, що подаються до захисту МКР

До захисту МКР подаються: пояснювальна записка МКР, ілюстративні та графічні матеріали, протокол попереднього захисту МКР на кафедрі, відзив наукового керівника, відзив опонента, протоколи складання державних екзаменів, анотації до МКР українською та іноземною мовами, довідка про відповідність оформлення МКР діючим вимогам.

7 Порядок контролю виконання та захисту МКР

Виконання етапів розрахункової та графічної документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

8 Вимоги до оформлення МКР

8.1 При оформлюванні МКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— Методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп'ютерна інженерія». Кафедра обчислювальної техніки ВНТУ 2022.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ–03.02.02 П.001.01:21

Таблиця А.1 — Етапи МКР

з/п	Назва етапів виконання магістерської роботи	Строк виконання етапів роботи	.
	Постановка мети та задач роботи	21.10.24	
		25.10-30.10.24	
	огляд та аналіз методів та мікропроцесорних засобів вимірювання електронних компонентів	31.10-08.11.24	
	Вибір моделі мікроконтролера	09.11-15.11.24	
	Проектування апаратної частини	16.11-.20.11.24	
	Розробка електричної принципової схеми	21.11-25.11.24	
	Проектування програмної частини	26.11-31.11.24	
	Розрахунок економічної частини роботи	01.12-04.12.24	
	Оформлення пояснювальної записки та ілюстративного матеріалу	05.12.24	
	Аналіз виконання роботи, висновки, додатки		
	Перевірка якості виконання магістерської роботи та усунення недоліків		

ДОДАТОК Б

Лістинг програми

```
#include <avr/io.h>
#include <util/delay.h>
#include <avr/sleep.h>
#include <stdlib.h>
#include <string.h>
#include <avr/eeprom.h>
#include <avr/pgmspace.h>
#include <avr/wdt.h>
#include <avr/interrupt.h>
#include <math.h>
#include <stdint.h>
#include <avr/power.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
#define MCU atmega328p
#define F_CPU 16000000UL
#define LANG_ENGLISH
#define WITH_AUTO_REF
#define REF_C_KORR 12
#define REF_L_KORR 40
#define C_H_KORR 0
#define CAP_EMPTY_LEVEL 4
#define AUTOSCALE_ADC
#define REF_R_KORR 3
#define ESR_ZERO 20
#define PULLUP_DISABLE
#define BAT_POOR 6400

#ifndef ADC_PORT

#define ADC_PORT PORTC
#define ADC_DDR DDRC
#define ADC_PIN PINC
#define TP1 0
#define TP2 1
#define TP3 2
#define TPext 3
#define TPREF 4
#define TPBAT 5
#define R_L_VAL 6800
#define R_H_VAL 47000
#define R_DDR DDRB
#define R_PORT PORTB
#define ON_DDR DDRD
#define ON_PORT PORTD
#define ON_PIN_REG PIND
#define ON_PIN 18
#ifdef STRIP_GRID_BOARD
#define RST_PIN
#else
#define RST_PIN 17
#endif

#ifdef STRIP_GRID_BOARD
```

```
  #define HW_LCD_EN_PORT         PORTD
  #define HW_LCD_EN_PIN          5

  #define HW_LCD_RS_PORT         PORTD
  #define HW_LCD_RS_PIN          7

  #define HW_LCD_B4_PORT         PORTD
  #define HW_LCD_B4_PIN          4
  #define HW_LCD_B5_PORT         PORTD
  #define HW_LCD_B5_PIN          3
  #define HW_LCD_B6_PORT         PORTD
  #define HW_LCD_B6_PIN          2
  #define HW_LCD_B7_PORT         PORTD
  #define HW_LCD_B7_PIN          1
#else
  #define HW_LCD_EN_PORT         PORTD
  #define HW_LCD_EN_PIN          6

  #define HW_LCD_RS_PORT         PORTD
  #define HW_LCD_RS_PIN          7

  #define HW_LCD_B4_PORT         PORTD
  #define HW_LCD_B4_PIN          5
  #define HW_LCD_B5_PORT         PORTD
  #define HW_LCD_B5_PIN          4
  #define HW_LCD_B6_PORT         PORTD
  #define HW_LCD_B6_PIN          3
  #define HW_LCD_B7_PORT         PORTD
  #define HW_LCD_B7_PIN          2
#endif

#define U_VCC 5000

#define U_SCALE 4

#define R_ANZ_MESS 190


#if R_ANZ_MESS < ANZ_MESS
  #undef R_ANZ_MESS
  #define R_ANZ_MESS ANZ_MESS
#endif
#if U_SCALE < 0
  // limit U_SCALE
  #undef U_SCALE
  #define U_SCALE 1
#endif
#if U_SCALE > 4
  // limit U_SCALE
  #undef U_SCALE
  #define U_SCALE 4
#endif
#ifndef REF_L_KORR
  #define REF_L_KORR 50
#endif


#ifdef USE_EEPROM
  #define MEM_TEXT EEMEM
```

```
  #if E2END > 0X1FF
    #define MEM2_TEXT EEMEM
    #define MEM2_read_byte(a)  eeprom_read_byte(a)
    #define MEM2_read_word(a)  eeprom_read_word(a)
    #define lcd_fix2_string(a)  lcd_fix_string(a)
  #else
    #define MEM2_TEXT PROGMEM
    #define MEM2_read_byte(a)  pgm_read_byte(a)
    #define MEM2_read_word(a)  pgm_read_word(a)
    #define lcd_fix2_string(a)  lcd_pgm_string(a)
    #define use_lcd_pgm
  #endif

  #define MEM_read_word(a)  eeprom_read_word(a)
  #define MEM_read_byte(a)  eeprom_read_byte(a)

#else
  #define MEM_TEXT PROGMEM
  #define MEM2_TEXT PROGMEM
  #define MEM_read_word(a)  pgm_read_word(a)
  #define MEM_read_byte(a)  pgm_read_byte(a)
  #define MEM2_read_byte(a)  pgm_read_byte(a)
  #define MEM2_read_word(a)  pgm_read_word(a)
  #define lcd_fix2_string(a)  lcd_pgm_string(a)
  #define use_lcd_pgm
#endif


#define RH_OFFSET 3500

#define TP2_CAP_OFFSET 2

#define CABLE_CAP 3


#define PROCESSOR_TYP 328

// automatic selection of right call type
#if FLASHEND > 0X1FFF
  #define ACALL call
#else
  #define ACALL rcall
#endif

// automatic selection of option and parameters for different AVRs

#if PROCESSOR_TYP == 168
  #define MCU_STATUS_REG MCUCR
  #define ADC_COMP_CONTROL ADCSRB
  #define TI1_INT_FLAGS TIFR1
  #define DEFAULT_BAND_GAP 1070
  #define DEFAULT_RH_FAKT  884      // mega328 1070 mV
  #define LONG_HFE
  #define COMMON_COLLECTOR
  #define MEGA168A 17
  #define MEGA168PA 18

  #define PIN_RM 190
  #define PIN_RP 220
  #define CC0 36
```

```
  #define COMP_SLEW1 4000
  #define COMP_SLEW2 220
  #define C_NULL CC0+CABLE_CAP+(COMP_SLEW1 / (CC0 + CABLE_CAP + COMP_SLEW2))
  #define MUX_INT_REF 0x0e  // channel number of internal 1.1 V

#elif PROCESSOR_TYP == 328
  #define MCU_STATUS_REG MCUCR
  #define ADC_COMP_CONTROL ADCSRB
  #define TI1_INT_FLAGS TIFR1
  #define DEFAULT_BAND_GAP 1070
  #define DEFAULT_RH_FAKT  884      // mega328 1070 mV
  #define LONG_HFE
  #define COMMON_COLLECTOR
  #define PIN_RM 200
  #define PIN_RP 220
  #define CC0 36
  #define COMP_SLEW1 4000
  #define COMP_SLEW2 180
  #define C_NULL CC0+CABLE_CAP+(COMP_SLEW1 / (CC0 + CABLE_CAP + COMP_SLEW2))
  #define MUX_INT_REF 0x0e  // channel number of internal 1.1 V

#elif PROCESSOR_TYP == 1280
  #define MCU_STATUS_REG MCUCR
  #define ADC_COMP_CONTROL ADCSRB
  #define TI1_INT_FLAGS TIFR1
  #define DEFAULT_BAND_GAP 1070
  #define DEFAULT_RH_FAKT  884      // mega328 1070 mV
  #define LONG_HFE
  #define COMMON_COLLECTOR

  #define PIN_RM 200
  #define PIN_RP 220
  #define CC0 36
  #define COMP_SLEW1 4000
  #define COMP_SLEW2 180
  #define C_NULL CC0+CABLE_CAP+(COMP_SLEW1
  #define MUX_INT_REF 0x1e  /* channel number of internal 1.1 V */

#else
  #define MCU_STATUS_REG MCUCSR
  #define ADC_COMP_CONTROL SFIOR
  #define TI1_INT_FLAGS TIFR
  #define DEFAULT_BAND_GAP 1298   //mega8 1298 mV
  #define DEFAULT_RH_FAKT  740      // mega8 1250 mV
  #define LONG_HFE
  #define COMMON_COLLECTOR

  #define PIN_RM 196
  #define PIN_RP 240
  #define CC0 27
  #define COMP_SLEW1 0
  #define COMP_SLEW2 33
  #define C_NULL CC0+CABLE_CAP+(COMP_SLEW1 / (CC0 + CABLE_CAP + COMP_SLEW2))
  #define MUX_INT_REF 0x0e  /* channel number of internal 1.1 V */

  #ifndef INHIBIT_SLEEP_MODE
    #define INHIBIT_SLEEP_MODE  /* do not use the sleep mode of ATmega */
  #endif
#endif
```

```
#if PROCESSOR_TYP == 8
  // 2.54V reference voltage + correction (fix for ATmega8)
  #ifdef AUTO_CAL
    #define            ADC_internal_reference            (2560            +
(int8_t)eeprom_read_byte((uint8_t *)&RefDiff))
  #else
    #define ADC_internal_reference (2560 + REF_R_KORR)
  #endif
#else
  // all other processors use a 1.1V reference
  #ifdef AUTO_CAL
    #define           ADC_internal_reference           (ref_mv           +
(int8_t)eeprom_read_byte((uint8_t *)&RefDiff))
  #else
    #define ADC_internal_reference (ref_mv + REF_R_KORR)
  #endif
#endif


#ifndef REF_R_KORR
  #define REF_R_KORR 0
#endif
#ifndef REF_C_KORR
  #define REF_C_KORR 0
#endif

#define LONG_WAIT_TIME 28000
#define SHORT_WAIT_TIME 5000

#ifdef POWER_OFF
  // if POWER OFF function is selected, wait 14s
  // if POWER_OFF with parameter > 2, wait only 5s before repeating
  #if (POWER_OFF+0) > 2
    #define OFF_WAIT_TIME SHORT_WAIT_TIME
  #else
    #define OFF_WAIT_TIME LONG_WAIT_TIME
  #endif
#else
  // if POWER OFF function is not selected, wait 14s before repeat
measurement
  #define OFF_WAIT_TIME  LONG_WAIT_TIME
#endif


#define F_ADC 125000
#if F_CPU/F_ADC == 2
  #define AUTO_CLOCK_DIV (1<<ADPS0)
#endif
#if F_CPU/F_ADC == 4
  #define AUTO_CLOCK_DIV (1<<ADPS1)
#endif
#if F_CPU/F_ADC == 8
  #define AUTO_CLOCK_DIV (1<<ADPS1) | (1<<ADPS0)
#endif
#if F_CPU/F_ADC == 16
  #define AUTO_CLOCK_DIV (1<<ADPS2)
#endif
#if F_CPU/F_ADC == 32
  #define AUTO_CLOCK_DIV (1<<ADPS2) | (1<<ADPS0)
#endif
```

```
#if F_CPU/F_ADC == 64
  #define AUTO_CLOCK_DIV (1<<ADPS2) | (1<<ADPS1)
#endif
#if F_CPU/F_ADC == 128
  #define AUTO_CLOCK_DIV (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0)
#endif
//*************************************************************
#define F_ADC_F 500000
#if F_CPU/F_ADC_F == 2
  #define FAST_CLOCK_DIV (1<<ADPS0)
#endif
#if F_CPU/F_ADC_F == 4
  #define FAST_CLOCK_DIV (1<<ADPS1)
#endif
#if F_CPU/F_ADC_F == 8
  #define FAST_CLOCK_DIV (1<<ADPS1) | (1<<ADPS0)
#endif
#if F_CPU/F_ADC_F == 16
  #define FAST_CLOCK_DIV (1<<ADPS2)
#endif
#if F_CPU/F_ADC_F == 32
  #define FAST_CLOCK_DIV (1<<ADPS2) | (1<<ADPS0)
#endif
#if F_CPU/F_ADC_F == 64
  #define FAST_CLOCK_DIV (1<<ADPS2) | (1<<ADPS1)
#endif
#if F_CPU/F_ADC_F == 128
  #define FAST_CLOCK_DIV (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0)
#endif


#ifndef PIN_RP
  #define PIN_RP  220            // estimated internal resistance PORT to
VCC
                                 // will only be used, if not set before in
config.h
#endif
#ifndef PIN_RM
  #define PIN_RM  190            // estimated internal resistance PORT to
GND
                                 // will only be used, if not set before in
config.h
#endif


#define TxD 3   // TxD-Pin of Software-UART; must be at Port C !
#ifdef WITH_UART
  #define TXD_MSK (1<<TxD)
#else
  #define TXD_MSK 0xF8
#endif

#ifdef SWUART_INVERT
  #define TXD_VAL 0
#else
  #define TXD_VAL TXD_MSK
#endif


#ifdef INHIBIT_SLEEP_MODE
```

```
  // save memory, do not use the sleep mode
  #define wait_about5ms() wait5ms()
  #define wait_about10ms() wait10ms()
  #define wait_about20ms() wait20ms()
  #define wait_about30ms() wait30ms()
  #define wait_about50ms() wait50ms()
  #define wait_about100ms() wait100ms()
  #define wait_about200ms() wait200ms()
  #define wait_about300ms() wait300ms()
  #define wait_about400ms() wait400ms()
  #define wait_about500ms() wait500ms()
  #define wait_about1s() wait1s()
  #define wait_about2s() wait2s()
  #define wait_about3s() wait3s()
  #define wait_about4s() wait4s()
#else
  // use sleep mode to save current for user interface
  #define wait_about5ms() sleep_5ms(1)
  #define wait_about10ms() sleep_5ms(2)
  #define wait_about20ms() sleep_5ms(4)
  #define wait_about30ms() sleep_5ms(6)
  #define wait_about50ms() sleep_5ms(10)
  #define wait_about100ms() sleep_5ms(20)
  #define wait_about200ms() sleep_5ms(40)
  #define wait_about300ms() sleep_5ms(60)
  #define wait_about400ms() sleep_5ms(80)
  #define wait_about500ms() sleep_5ms(100)
  #define wait_about1s() sleep_5ms(200)
  #define wait_about2s() sleep_5ms(400)
  #define wait_about3s() sleep_5ms(600)
  #define wait_about4s() sleep_5ms(800)
#endif


#undef AUTO_RH
#ifdef WITH_AUTO_REF
  #define AUTO_RH
#else
  #ifdef AUTO_CAL
    #define AUTO_RH
  #endif
#endif

#undef CHECK_CALL
#ifdef WITH_SELFTEST
  // AutoCheck Function is needed
  #define CHECK_CALL
#endif

#ifdef AUTO_CAL
  // AutoCheck Function is needed
  #define CHECK_CALL
  #define RR680PL resis680pl
  #define RR680MI resis680mi
  #define RRpinPL pin_rpl
  #define RRpinMI pin_rmi
#else
  #define RR680PL (R_L_VAL + PIN_RP)
  #define RR680MI (R_L_VAL + PIN_RM)
  #define RRpinPL (PIN_RP)
```

```
  #define RRpinMI (PIN_RM)
#endif

#ifndef ESR_ZERO
  // define a default zero value for ESR measurement (0.01 Ohm units)
  #define ESR_ZERO 20
#endif

#ifndef RESTART_DELAY_TICS
  #define RESTART_DELAY_TICS 16384

#endif


#if EBC_STYLE == 123
  #undef EBC_STYLE
#endif


// self build characters
#define LCD_CHAR_DIODE1  1      // Diode-Icon; will be generated as custom
character
#define LCD_CHAR_DIODE2  2      // Diode-Icon; will be generated as custom
character
#define LCD_CHAR_CAP 3           // Capacitor-Icon;  will be generated as
custom character
#define LCD_CHAR_RESIS1 7       // Resistor left part will be generated as
custom character
#define LCD_CHAR_RESIS2 6      // Resistor right part will be generated as
custom character

#ifdef LCD_CYRILLIC
  #define LCD_CHAR_OMEGA  4       // Omega-character
  #define LCD_CHAR_U  5           // micro-character
#else
  #define LCD_CHAR_OMEGA  244     // Omega-character
  #define LCD_CHAR_U  228         // micro-character
#endif

#ifdef LCD_DOGM
  #undef LCD_CHAR_OMEGA
  #define LCD_CHAR_OMEGA 0x1e     // Omega-character for DOGM module
  #undef LCD_CHAR_U
  #define LCD_CHAR_U  5          // micro-character for DOGM module loadable
#endif

#define LCD_CHAR_DEGREE 0xdf      // Character for degree

#endif  // #ifndef ADC_PORT


#ifdef COMMON_COLLECTOR
  #if FLASHEND > 0x3fff
    #define COMMON_EMITTER
  #endif
#else
  #define COMMON_EMITTER
#endif
```

```
#define MAIN_C

#if defined (MAIN_C)
  #define COMMON

const           uint16_t           RLtab[]           MEM_TEXT           =
{22447,20665,19138,17815,16657,15635,14727,13914,13182,12520,11918,11369,1
0865,10401, 9973, 9577, 9209, 8866, 8546, 8247, 7966, 7702, 7454, 7220,
6999, 6789, 6591, 6403, 6224, 6054, 5892, 5738, 5590, 5449, 5314, 5185,
5061, 4942, 4828, 4718, 4613, 4511, 4413, 4319, 4228};

#if FLASHEND > 0x1fff
  //                                  {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,
12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30,
31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49,
50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,  64,  65,  66,  67,
68,  69,  70,  71,  72,  73,  74,  75,  76,  77,  78,  79,  80,  81,  82,
83,  84,  85,  86,  87,  88,  89,  90,  91 };
  const uint16_t LogTab[] PROGMEM = {0, 20, 41, 62, 83, 105, 128, 151, 174,
198, 223, 248, 274, 301, 329, 357, 386, 416, 446, 478, 511, 545, 580, 616,
654, 693, 734, 777, 821, 868, 916, 968, 1022, 1079, 1139, 1204, 1273, 1347,
1427, 1514, 1609, 1715, 1833, 1966, 2120, 2303, 2526 };
#endif

#ifdef AUTO_RH
  // resistor  470000 Ohm      1000 1050 1100 1150 1200 1250 1300 1350 1400
mV
  const uint16_t RHtab[] PROGMEM = { 954, 903, 856, 814, 775, 740, 707,
676, 648};
#endif

// with integer factors the ADC-value will be changed to mV resolution in
ReadADC !
// all if statements are corrected to the mV resolution.


// Strings in PROGMEM or in EEprom

#if defined(LANG_GERMAN)    // deutsch
   const unsigned char TestRunning[] MEM_TEXT = "Testen...";
   const unsigned char BatWeak[] MEM_TEXT = "gering";
   const unsigned char BatEmpty[] MEM_TEXT = "leer!";
   const unsigned char TestFailed2[] MEM_TEXT = "defektes ";
   const unsigned char Component[] MEM_TEXT = "Bauteil";
//   const unsigned char Diode[] MEM_TEXT = "Diode: ";
   const unsigned char Triac[] MEM_TEXT = "Triac";
   const unsigned char Thyristor[] MEM_TEXT = "Thyristor";
   const unsigned char Unknown[] MEM_TEXT = " unbek.";
   const unsigned char TestFailed1[] MEM_TEXT = "Kein,unbek. oder";
   const unsigned char OrBroken[] MEM_TEXT = "oder defekt ";
   const unsigned char TestTimedOut[] MEM_TEXT = "Timeout!";
   #define Cathode_char 'K'
 #ifdef WITH_SELFTEST
   const unsigned char SELFTEST[] MEM_TEXT = "Selbsttest ..";
   const unsigned char RELPROBE[] MEM_TEXT = "isolate Probe!";
   const unsigned char ATE[] MEM_TEXT = "Test Ende";
 #endif
#endif

#if defined(LANG_ENGLISH)   // english
```

```
  const unsigned char TestRunning[] MEM_TEXT = "testing...";
  const unsigned char BatWeak[] MEM_TEXT = "weak";
  const unsigned char BatEmpty[] MEM_TEXT = "empty!";
  const unsigned char TestFailed2[] MEM_TEXT = "damaged ";
  const unsigned char Component[] MEM_TEXT = "part";
  //const unsigned char Diode[] MEM_TEXT = "Diode: ";
  const unsigned char Triac[] MEM_TEXT = "Triac";
  const unsigned char Thyristor[] MEM_TEXT = "Thyristor";
  const unsigned char Unknown[] MEM_TEXT = " unknown";
  const unsigned char TestFailed1[] MEM_TEXT = "No, unknown, or";
  const unsigned char OrBroken[] MEM_TEXT = "or damaged ";
  const unsigned char TestTimedOut[] MEM_TEXT = "Timeout!";
  #define Cathode_char 'C'

  #ifdef WITH_SELFTEST
    const unsigned char SELFTEST[] MEM_TEXT = "Selftest mode..";
    const unsigned char RELPROBE[] MEM_TEXT = "isolate Probe!";
    const unsigned char ATE[] MEM_TEXT = "Test End";
  #endif
#endif


// Strings, which are not dependent of any language
const unsigned char Bat_str[] MEM_TEXT = "Bat. ";
const unsigned char OK_str[] MEM_TEXT = "OK";
const unsigned char mosfet_str[] MEM_TEXT = "-MOS";
const unsigned char jfet_str[] MEM_TEXT = "JFET";
const unsigned char GateCap_str[] MEM_TEXT = "C=";
const unsigned char hfe_str[] MEM_TEXT ="B=";
const unsigned char NPN_str[] MEM_TEXT = "NPN ";
const unsigned char PNP_str[] MEM_TEXT = "PNP ";

#ifndef EBC_STYLE
  const unsigned char N123_str[] MEM_TEXT = " 123=";
  //const unsigned char N123_str[] MEM_TEXT = " Pin=";
#else
  #if EBC_STYLE == 321
    const unsigned char N321_str[] MEM_TEXT = " 321=";
  #endif
#endif

const unsigned char Uf_str[] MEM_TEXT = "Uf=";
const unsigned char vt_str[] MEM_TEXT = " Vt=";
const unsigned char Vgs_str[] MEM_TEXT = "@Vgs=";
const unsigned char CapZeich[] MEM_TEXT = {'-',LCD_CHAR_CAP,'-',0};
const unsigned char Cell_str[] MEM_TEXT = "Cell!";
const unsigned char VCC_str[] MEM_TEXT = "VCC=";

#if FLASHEND > 0x1fff
  const unsigned char ESR_str[] MEM_TEXT = " ESR=";
  const unsigned char VLOSS_str[] MEM_TEXT = " Vloss=";
  const unsigned char Lis_str[] MEM_TEXT = "L=";
  const unsigned char Ir_str[] MEM_TEXT = "  Ir=";

  #ifndef WITH_UART
    //#define WITH_VEXT
  #endif
#else
  #ifndef BAT_CHECK
    #ifndef WITH_UART
```

```
      //#define WITH_VEXT
    #endif
  #endif
#endif

#ifdef WITH_VEXT
  const unsigned char Vext_str[] MEM_TEXT = "Vext=";
  #define LCD_CLEAR
#endif


const unsigned char VERSION_str[] MEM2_TEXT = "-- Diy Energy --";

const unsigned char AnKat[] MEM_TEXT = {'-', LCD_CHAR_DIODE1, '-',0};
const unsigned char KatAn[] MEM_TEXT = {'-', LCD_CHAR_DIODE2, '-',0};
const unsigned char Diodes[] MEM_TEXT = {'*',LCD_CHAR_DIODE1, ' ', ' ',0};
const unsigned char Resistor_str[] MEM_TEXT = {'-', LCD_CHAR_RESIS1,
LCD_CHAR_RESIS2,'-',0};

#ifdef WITH_SELFTEST
  const unsigned char URefT[] MEM2_TEXT = "Ref=";
  const unsigned char RHfakt[] MEM2_TEXT = "RHf=";
  const unsigned char RH1L[] MEM_TEXT = "RH-";
  const unsigned char RH1H[] MEM_TEXT = "RH+";
  const unsigned char RLRL[] MEM_TEXT = "+RL- 12 13 23";
  const unsigned char RHRH[] MEM_TEXT = "+RH- 12 13 23";
  const unsigned char RHRL[] MEM_TEXT = "RH/RL";
  const unsigned char R0_str[] MEM2_TEXT = "R0=";
  #define LCD_CLEAR
#endif

#ifdef CHECK_CALL
  const unsigned char RIHI[] MEM_TEXT = "Ri_Hi=";
  const unsigned char RILO[] MEM_TEXT = "Ri_Lo=";
  const unsigned char C0_str[] MEM_TEXT = "C0 ";
  const unsigned char T50HZ[] MEM_TEXT = " 50Hz";
#endif

#ifdef AUTO_CAL
  const unsigned char MinCap_str[] MEM2_TEXT = " >100nF";
  const unsigned char REF_C_str[] MEM2_TEXT = "REF_C=";
  const unsigned char REF_R_str[] MEM2_TEXT = "REF_R=";
#endif

#ifdef DebugOut
  #define LCD_CLEAR
#endif


const unsigned char DiodeIcon1[] MEM_TEXT = { 0x11, 0x19, 0x1d, 0x1f, 0x1d,
0x19, 0x11, 0x00 }; // Diode-Icon Anode left
const unsigned char DiodeIcon2[] MEM_TEXT = { 0x11, 0x13, 0x17, 0x1f, 0x17,
0x13, 0x11, 0x00 }; // Diode-Icon Anode right
const unsigned char CapIcon[]    MEM_TEXT = { 0x1b, 0x1b, 0x1b, 0x1b, 0x1b,
0x1b, 0x1b, 0x00 }; // Capacitor Icon
const unsigned char ResIcon1[]   MEM_TEXT = { 0x00, 0x0f, 0x08, 0x18, 0x08,
0x0f, 0x00, 0x00 }; // Resistor Icon1 left
const unsigned char ResIcon2[]   MEM_TEXT = { 0x00, 0x1e, 0x02, 0x03, 0x02,
0x1e, 0x00, 0x00 }; // Resistor Icon2 right
```

```
const unsigned char OmegaIcon[]  MEM_TEXT = { 0x00, 0x00, 0x0e, 0x11, 0x11,
0x0a, 0x1b, 0x00 }; // Omega Icon
const unsigned char MicroIcon[]  MEM_TEXT = { 0x00, 0x00, 0x0a, 0x0a, 0x0a,
0x0e, 0x09, 0x10 }; // Micro Icon

const unsigned char PinRLtab[] PROGMEM = { (1<<(TP1*2)), (1<<(TP2*2)),
(1<<(TP3*2))};  // Table of commands to switch the  R-L resistors Pin 0,1,2
const unsigned char PinADCtab[] PROGMEM = { (1<<TP1), (1<<TP2), (1<<TP3)};
// Table of commands to switch the ADC-Pins 0,1,2

/*
// generate Omega- and u-character as Custom-character, if these characters
has a number of loadable type
#if LCD_CHAR_OMEGA < 8
  const     unsigned     char     CyrillicOmegaIcon[]     MEM_TEXT     =
{0,0,14,17,17,10,27,0};  // Omega
#endif
#if LCD_CHAR_U < 8
  const unsigned char CyrillicMuIcon[] MEM_TEXT = {0,17,17,17,19,29,16,16};
// micro
#endif
*/

#ifdef AUTO_CAL
  //const uint16_t R680pl EEMEM = R_L_VAL+PIN_RP; // total resistor to VCC
  //const uint16_t R680mi EEMEM = R_L_VAL+PIN_RM; // total resistor to GND
  const int8_t RefDiff EEMEM = REF_R_KORR;     // correction of internal
Reference Voltage
#endif

const uint8_t PrefixTab[] MEM_TEXT = { 'p','n',LCD_CHAR_U,'m',0,'k','M'};
// p,n,u,m,-,k,M

#ifdef AUTO_CAL
  //const uint16_t cap_null EEMEM = C_NULL; // Zero offset of capacity
measurement
  const int16_t ref_offset EEMEM = REF_C_KORR;  // default correction of
internal reference voltage for capacity measurement
  // LoPin:HiPin                        2:1    3:1    1:2                  :
3:2                   1:3    2:3
  const        uint8_t        c_zero_tab[]        EEMEM        =        {
C_NULL,C_NULL,C_NULL+TP2_CAP_OFFSET,C_NULL,C_NULL+TP2_CAP_OFFSET,C_NULL,C_
NULL }; // table of zero offsets
#endif

const uint8_t EE_ESR_ZEROtab[] PROGMEM = {ESR_ZERO, ESR_ZERO, ESR_ZERO,
ESR_ZERO};  // zero offset of ESR measurement

// End of EEPROM-Strings

// Multiplier for capacity measurement with R_H (470KOhm)
unsigned int RHmultip = DEFAULT_RH_FAKT;


#else
  // no MAIN_C
  #define COMMON extern
  #ifdef WITH_SELFTEST
    extern const unsigned char SELFTEST[] MEM_TEXT;
    extern const unsigned char RELPROBE[] MEM_TEXT;
```

```
    extern const unsigned char ATE[] MEM_TEXT;
  #endif

  #ifdef AUTO_CAL
    //extern uint16_t R680pl;
    //extern uint16_t R680mi;
    extern int8_t RefDiff;
    extern uint16_t ref_offset;
    extern uint8_t c_zero_tab[];
  #endif

  extern const uint8_t EE_ESR_ZEROtab[] EEMEM;   // zero offset of ESR
measurement
  extern const uint16_t RLtab[];

  #if FLASHEND > 0x1fff
    extern uint16_t LogTab[];
    extern const unsigned char ESR_str[];
  #endif

  #ifdef AUTO_RH
    extern const uint16_t RHtab[];
  #endif

  extern const unsigned char PinRLtab[];
  extern const unsigned char PinADCtab[];
  extern unsigned int RHmultip;

#endif  // MAIN_C


struct Diode_t {
  uint8_t Anode;
  uint8_t Cathode;
  unsigned int Voltage;
};

COMMON struct Diode_t diodes[6];
COMMON uint8_t NumOfDiodes;

COMMON struct {
  unsigned long hfe[2];   // current amplification factor
  unsigned int uBE[2];    // B-E-voltage of the Transistor
  uint8_t b,c,e;    // pins of the Transistor
}trans;

COMMON unsigned int gthvoltage; // Gate-threshold voltage

COMMON uint8_t PartReady; // part detection is finished
COMMON uint8_t PartMode;
COMMON uint8_t tmpval, tmpval2;
COMMON unsigned int ref_mv;     // Reference-voltage  in mV units

COMMON struct resis_t{
  unsigned long rx;   // value of resistor RX
  #if FLASHEND > 0x1fff
    unsigned long lx;   // inductance 10uH or 100uH
    int8_t lpre;    // prefix for inductance
  #endif
  uint8_t ra,rb;    // Pins of RX
```

```
  uint8_t rt;     // Tristate-Pin (inactive)
} resis[3];

COMMON uint8_t ResistorsFound;  // Number of found resistors
COMMON uint8_t ii;    // multipurpose counter

COMMON struct cap_t {
  unsigned long cval;   // capacitor value
  unsigned long cval_max; // capacitor with maximum value
  union t_combi{
  unsigned long dw;   // capacity value without corrections
  uint16_t w[2];
  } cval_uncorrected;
  #if FLASHEND > 0x1fff
    unsigned int esr;   // serial resistance of C in 0.01 Ohm
    unsigned int v_loss;  // voltage loss 0.1%
  #endif
  uint8_t ca, cb;   // pins of capacitor
  int8_t cpre;      // Prefix for capacitor value  -12=p, -9=n, -6=u, -3=m
  int8_t cpre_max;    // Prefix of the biggest capacitor
} cap;

#ifndef INHIBIT_SLEEP_MODE
  // with sleep mode we need a global ovcnt16
  COMMON volatile uint16_t ovcnt16;
  COMMON volatile uint8_t unfinished;
#endif

COMMON int16_t load_diff; // difference voltage of loaded capacitor and
internal reference

COMMON uint8_t WithReference; // Marker for found precision voltage
reference = 1
COMMON uint8_t PartFound; // the found part
COMMON char outval[12];   // String for ASCII-outpu
COMMON uint8_t empty_count; // counter for max count of empty measurements
COMMON uint8_t mess_count;    // counter for max count of nonempty
measurements

COMMON struct ADCconfig_t {
  uint8_t Samples;    // number of ADC samples to take
  uint8_t RefFlag;    // save Reference type VCC of IntRef
  uint16_t U_Bandgap;   // Reference Voltage in mV
  uint16_t U_AVCC;    // Voltage of AVCC
} ADCconfig;

#ifdef AUTO_CAL
  COMMON    uint8_t    pin_combination;    //    coded    Pin-combination
2:1,3:1,1:2,x:x,3:2,1:3,2:3
  COMMON uint16_t resis680pl; // port output resistance + 680
  COMMON uint16_t resis680mi; // port output resistance + 680
  COMMON uint16_t pin_rmi;  // port output resistance to GND side, 0.1 Ohm
units
  COMMON uint16_t pin_rpl;  // port output resistance to VCC side, 0.1 Ohm
units
#endif

#if POWER_OFF+0 > 1
  COMMON unsigned int display_time; // display time of measurement in ms
units
```

```
#endif

// definitions of parts
#define PART_NONE 0
#define PART_DIODE 1
#define PART_TRANSISTOR 2
#define PART_FET 3
#define PART_TRIAC 4
#define PART_THYRISTOR 5
#define PART_RESISTOR 6
#define PART_CAPACITOR 7
#define PART_CELL 8

#define PART_MODE_N_E_MOS 2
#define PART_MODE_P_E_MOS 3
#define PART_MODE_N_D_MOS 4
#define PART_MODE_P_D_MOS 5
#define PART_MODE_N_JFET 6
#define PART_MODE_P_JFET 7

// Bipolar
#define PART_MODE_NPN 1
#define PART_MODE_PNP 2


// wait functions
#define  wait5s()    delay(5000)
#define  wait4s()    delay(4000)
#define  wait3s()    delay(3000)
#define  wait2s()    delay(2000)
#define  wait1s()    delay(1000)
#define  wait500ms() delay(500)
#define  wait400ms() delay(400)
#define  wait300ms() delay(300)
#define  wait200ms() delay(200)
#define  wait100ms() delay(100)
#define  wait50ms()  delay(50)
#define  wait40ms()  delay(40)
#define  wait30ms()  delay(30)
#define  wait20ms()  delay(20)
#define  wait10ms()  delay(10)
#define  wait5ms()   delay(5)
#define  wait4ms()   delay(4)
#define  wait3ms()   delay(3)
#define  wait2ms()   delay(2)
#define  wait1ms()   delay(1)
#define  wait500us() delayMicroseconds(500)
#define  wait400us() delayMicroseconds(400)
#define  wait300us() delayMicroseconds(300)
#define  wait200us() delayMicroseconds(200)
#define  wait100us() delayMicroseconds(100)
#define  wait50us()  delayMicroseconds(50)
#define  wait40us()  delayMicroseconds(40)
#define  wait30us()  delayMicroseconds(30)
#define  wait20us()  delayMicroseconds(20)
#define  wait10us()  delayMicroseconds(10)
#define  wait5us()   delayMicroseconds(5)
#define  wait4us()   delayMicroseconds(4)
#define  wait3us()   delayMicroseconds(3)
#define  wait2us()   delayMicroseconds(2)
```

```
#define  wait1us()   delayMicroseconds(1)


// LCD-commands
#define CMD_ClearDisplay         0x01
#define CMD_ReturnHome           0x02
#define CMD_SetEntryMode         0x04
#define CMD_SetDisplayAndCursor  0x08
#define CMD_SetIFOptions         0x20
#define CMD_SetCGRAMAddress      0x40    // for Custom character
#define CMD_SetDDRAMAddress      0x80    // set Cursor

#define CMD1_SetBias              0x10  // set Bias (instruction table 1,
DOGM)
#define CMD1_PowerControl        0x50  // Power Control, set Contrast C5:C4
(instruction table 1, DOGM)
#define CMD1_FollowerControl     0x60  // Follower Control, amplified ratio
(instruction table 1, DOGM)
#define CMD1_SetContrast          0x70  // set Contrast C3:C0 (instruction
table 1, DOGM)

// Makros for LCD
#define lcd_line1() lcd_set_cursor(0,0)  // move to beginning of 1 row
#define lcd_line2() lcd_set_cursor(1,0)  // move to beginning of 2 row
#define lcd_line3() lcd_set_cursor(2,0)  // move to beginning of 3 row
#define lcd_line4() lcd_set_cursor(3,0)  // move to beginning of 4 row

#define lcd_init()         lcd.begin(20,4)
#define lcd_command(value) lcd.command(value)
#define lcd_home()         lcd.home()

#define uart_newline()     Serial.println()


#ifndef INHIBIT_SLEEP_MODE
  // prepare sleep mode
  EMPTY_INTERRUPT(TIMER2_COMPA_vect);
  EMPTY_INTERRUPT(ADC_vect);
#endif

uint8_t tmp = 0;
//unsigned int PRR;

byte TestKey;
byte TestKeyPin = 17;  // A3

// begin of transistortester program
void setup()
{
  Serial.begin(9600);

  pinMode(TestKeyPin, INPUT);
   lcd.init();
 //lcd.begin(16,2);
 lcd.backlight();
 //lcd_init();  // initialize LCD

  lcd_pgm_custom_char(LCD_CHAR_DIODE1,  DiodeIcon1);   //  Custom-Character
Diode symbol >|
```

```
  lcd_pgm_custom_char(LCD_CHAR_DIODE2, DiodeIcon2);   // Custom-Character
Diode symbol |<
  lcd_pgm_custom_char(LCD_CHAR_CAP,    CapIcon);      // Custom-Character
Capacitor symbol ||
  lcd_pgm_custom_char(LCD_CHAR_RESIS1, ResIcon1);     // Custom-Character
Resistor symbol [
  lcd_pgm_custom_char(LCD_CHAR_RESIS2, ResIcon2);     // Custom-Character
Resistor symbol ]
  lcd_pgm_custom_char(LCD_CHAR_OMEGA,  OmegaIcon);    // load Omega as
Custom-Character
  lcd_pgm_custom_char(LCD_CHAR_U,    MicroIcon);  // load Micro as Custom-
Character
  lcd_home();

  lcd_string("   DIY ENERGY  ");
  lcd_set_cursor(1, 0);
  lcd_string("Component tester");

  //ON_DDR = 0;
  //ON_PORT = 0;

/*
  // switch on
  ON_DDR = (1<<ON_PIN);     // switch to output
  #ifdef PULLUP_DISABLE
    ON_PORT = (1<<ON_PIN);    // switch power on
  #else
    ON_PORT = (1<<ON_PIN)|(1<<RST_PIN);    // switch power on , enable
internal Pullup for Start-Pin
  #endif
*/

  // ADC-Init
  ADCSRA = (1<<ADEN) | AUTO_CLOCK_DIV;  // prescaler=8 or 64 (if 8Mhz clock)

  #ifdef __AVR_ATmega8__
    //#define WDRF_HOME MCU_STATUS_REG
    #define WDRF_HOME MCUCSR
  #else
    #define WDRF_HOME MCUSR
  #endif

/*
  tmp = (WDRF_HOME & (1<<WDRF));  // save Watch Dog Flag
  WDRF_HOME &= ~(1<<WDRF);    // reset Watch Dog flag
  wdt_disable();      // disable Watch Dog
*/

/*
  #ifndef INHIBIT_SLEEP_MODE
    // switch off unused Parts
    PRR = (1<<PRTWI) | (1<<PRTIM0) | (1<<PRSPI) | (1<<PRUSART0);
    DIDR0 = (1<<ADC5D) | (1<<ADC4D) | (1<<ADC3D);
    TCCR2A = (0<<WGM21) | (0<<WGM20);     // Counter 2 normal mode
    #if F_CPU <= 1000000UL
      TCCR2B = (1<<CS22) | (0<<CS21) | (1<<CS20); // prescaler 128, 128us
@ 1MHz
      #define T2_PERIOD 128
    #endif
    #if F_CPU == 2000000UL
```

```
      TCCR2B = (1<<CS22) | (1<<CS21) | (0<<CS20); // prescaler 256, 128us
@ 2MHz
      #define T2_PERIOD 128
    #endif
    #if F_CPU == 4000000UL
      TCCR2B = (1<<CS22) | (1<<CS21) | (0<<CS20); // prescaler 256, 64us @
2MHz
      #define T2_PERIOD 64
    #endif
    #if F_CPU >= 8000000UL
      TCCR2B = (1<<CS22) | (1<<CS21) | (1<<CS20); // prescaler 1024, 128us
@ 8MHz, 64us @ 16MHz
      #define T2_PERIOD (1024 / (F_CPU / 1000000UL)); // set to 128 or 64
us
    #endif
    sei();  // enable interrupts
  #endif
*/

  #define T2_PERIOD (1024 / (F_CPU / 1000000UL)); // set to 128 or 64 us

  //ADC_PORT = TXD_VAL;
  //ADC_DDR = TXD_MSK;

  if(tmp) {
    // check if Watchdog-Event
    // this happens, if the Watchdog is not reset for 2s
    // can happen, if any loop in the Program doen't finish.
    lcd_line1();
    lcd_fix_string(TestTimedOut); // Output Timeout
    wait_about3s();     // wait for 3 s
    //ON_PORT = 0;      // shut off!
    //ON_DDR = (1<<ON_PIN);   // switch to GND
    //return;
  }

  #ifdef PULLUP_DISABLE
    #ifdef __AVR_ATmega8__
      SFIOR = (1<<PUD);   // disable Pull-Up Resistors mega8
    #else
      MCUCR = (1<<PUD);   // disable Pull-Up Resistors mega168 family
    #endif
  #endif

  //DIDR0 = 0x3f;   // disable all Input register of ADC

/*
  #if POWER_OFF+0 > 1
    // tester display time selection
    display_time = OFF_WAIT_TIME;   // LONG_WAIT_TIME for single mode, else
SHORT_WAIT_TIME
    if (!(ON_PIN_REG & (1<<RST_PIN))) {
      // if power button is pressed ...
      wait_about300ms();      // wait to catch a long key press
      if (!(ON_PIN_REG & (1<<RST_PIN))) {
        // check if power button is still pressed
        display_time = LONG_WAIT_TIME;      // ... set long time display
anyway
      }
    }
```

```
  #else
    #define display_time OFF_WAIT_TIME
  #endif
*/

  #define display_time OFF_WAIT_TIME

  empty_count = 0;
  mess_count = 0;
}

void loop()
{
  // Entry: if start key is pressed before shut down
start:

  TestKey = 1;
  while(TestKey) {
    TestKey = digitalRead(TestKeyPin);
    delay(100);
  }
  while(!TestKey) {
    TestKey = digitalRead(TestKeyPin);
    delay(100);
  }
  lcd_clear();
  delay(100);

  PartFound = PART_NONE;  // no part found
  NumOfDiodes = 0;    // Number of diodes = 0
  PartReady = 0;
  PartMode = 0;
  WithReference = 0;    // no precision reference voltage
  ADC_DDR = TXD_MSK;    // activate Software-UART
  ResistorsFound = 0;   // no resistors found
  cap.ca = 0;
  cap.cb = 0;

  #ifdef WITH_UART
    uart_newline();   // start of new measurement
  #endif

  ADCconfig.RefFlag = 0;
  Calibrate_UR();   // get Ref Voltages and Pin resistance
  lcd_line1();    // 1 row

  ADCconfig.U_Bandgap = ADC_internal_reference;  // set internal reference
voltage for ADC

  #ifdef BAT_CHECK
    // Battery check is selected
    ReadADC(TPBAT);     // Dummy-Readout
    trans.uBE[0] = W5msReadADC(TPBAT);  // with 5V reference
    lcd_fix_string(Bat_str);    // output: "Bat. "

    #ifdef BAT_OUT

      cap.cval = (trans.uBE[0]*4)+BAT_OUT;      // usually output only 2
digits
```

```
      DisplayValue(cap.cval,-3,'V',2);       // Display 2 Digits of this
10mV units
      lcd_space();
    #endif

    #if (BAT_POOR > 12000)
      #warning "Battery POOR level is set very high!"
    #endif
    #if (BAT_POOR < 2500)
      #warning "Battery POOR level is set very low!"
    #endif

    #if (BAT_POOR > 5300)
      // use .8 V difference to Warn-Level
      #define    WARN_LEVEL    (((unsigned    long)(BAT_POOR+800)*(unsigned
long)33)/133)
    #elif (BAT_POOR > 3249)
      // less than 5.4 V only .4V difference to Warn-Level
      #define    WARN_LEVEL    (((unsigned    long)(BAT_POOR+400)*(unsigned
long)33)/133)
    #elif (BAT_POOR > 1299)
      // less than 2.9 V only .2V difference to Warn-Level
      #define    WARN_LEVEL    (((unsigned    long)(BAT_POOR+200)*(unsigned
long)33)/133)
    #else
      // less than 1.3 V only .1V difference to Warn-Level
      #define    WARN_LEVEL    (((unsigned    long)(BAT_POOR+100)*(unsigned
long)33)/133)
    #endif

    #define POOR_LEVEL (((unsigned long)(BAT_POOR)*(unsigned long)33)/133)

    // check the battery voltage
    if (trans.uBE[0] <  WARN_LEVEL) {

      // Vcc < 7,3V; show Warning
      if(trans.uBE[0] < POOR_LEVEL) {
        // Vcc <6,3V; no proper operation is possible
        lcd_fix_string(BatEmpty); // Battery empty!
        wait_about2s();
        PORTD = 0;      // switch power off
        return;
      }

      lcd_fix_string(BatWeak);    // Battery weak
    } else { // Battery-voltage OK
      lcd_fix_string(OK_str);     // "OK"
    }

  #else
    lcd_fix2_string(VERSION_str); // if no Battery check, Version .. in row
1
  #endif

  #ifdef WDT_enabled
    //wdt_enable(WDTO_2S);    // Watchdog on
  #endif

  #ifdef AUTO_RH
    RefVoltage();     // compute RHmultip = f(reference voltage)
```

```
  #endif

  #if FLASHEND > 0x1fff
    if (WithReference) {
      // 2.5V precision reference is checked OK
      if ((mess_count == 0) && (empty_count == 0)) {
        // display VCC= only first time
        lcd_line2();
        lcd_fix_string(VCC_str);      // VCC=
        DisplayValue(ADCconfig.U_AVCC,-3,'V',3);   // Display 3 Digits of
this mV units
        //lcd_space();
        //DisplayValue(RRpinMI,-1,LCD_CHAR_OMEGA,4);
        wait_about1s();
      }
    }
  #endif

  #ifdef WITH_VEXT
    // show the external voltage
    while (!(ON_PIN_REG & (1<<RST_PIN))) {
      lcd_line2();
      lcd_clear_line();
      lcd_line2();
      lcd_fix_string(Vext_str);     // Vext=
      ADC_DDR = 0;        // deactivate Software-UART
      trans.uBE[1] = W5msReadADC(TPext);  // read external voltage
      ADC_DDR = TXD_MSK;      // activate Software-UART

      #ifdef WITH_UART
        uart_newline();   // start of new measurement
      #endif

      DisplayValue(trans.uBE[1]*10,-3,'V',3); // Display 3 Digits of this
mV units
      wait_about300ms();
    }
  #endif
```

ДОДАТОК В

Схема електрична структурна

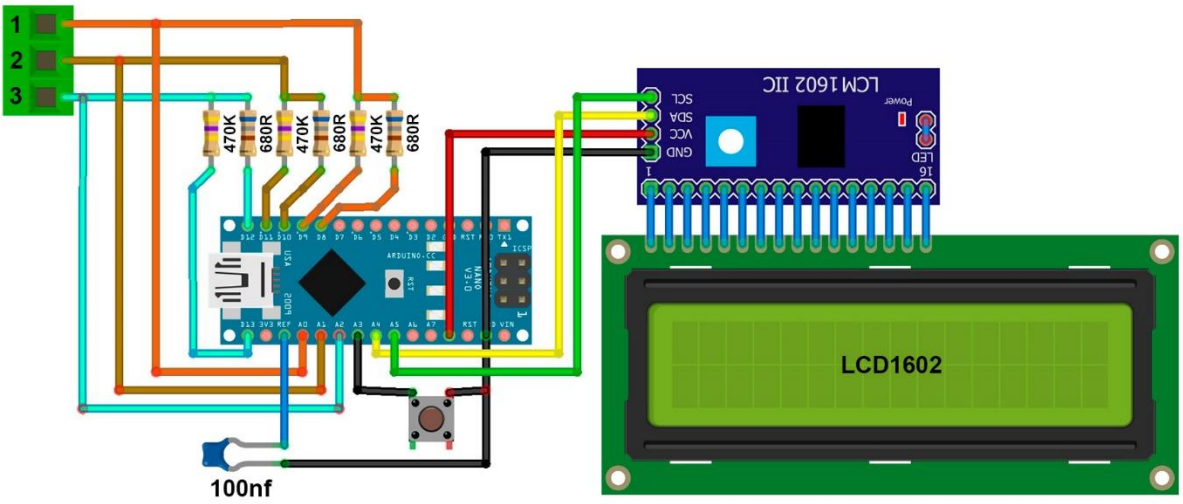
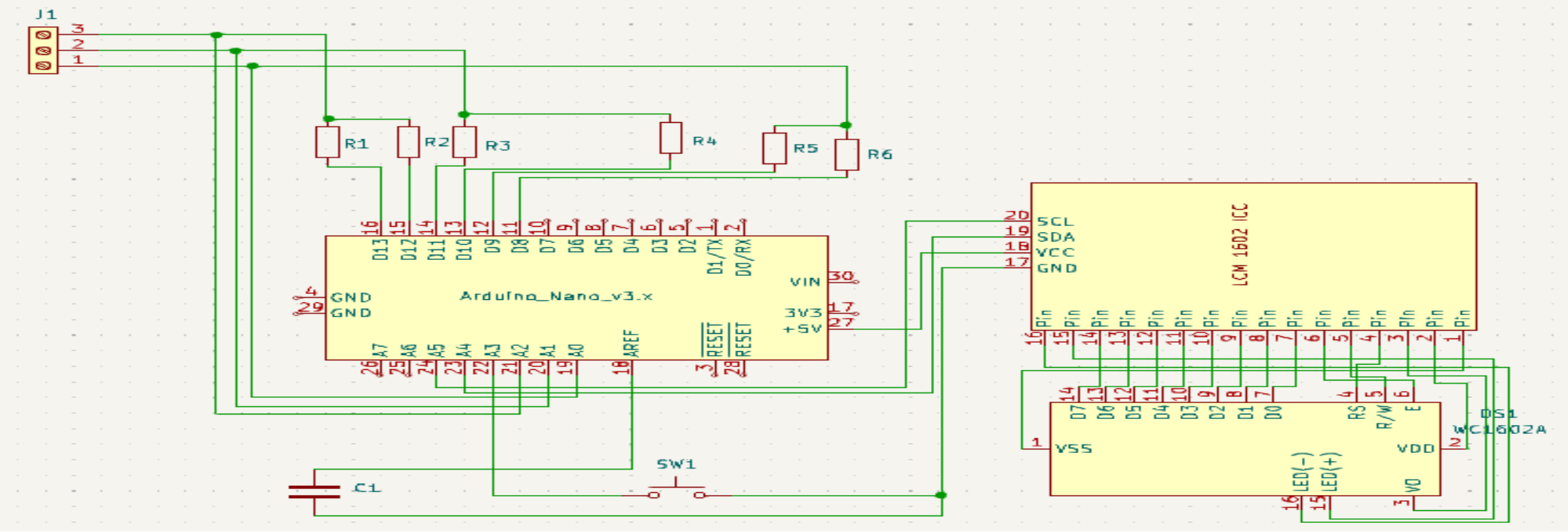


Рисунок В.1 — Схема електрична структурна

ДОДАТОК Г

Схема електрична принципова



					08-54.МКР.013.00.000 *Е3*
					Метод та мікропроцесорні засоби вимірювання параметрів електронних компонентів. Схема електрична принципова
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*	
Розробив		*Мушинський В.Є.*			
Перевірив		*Богомолов С.В.*			
Опонент		*Катєльніков Д.І*			
Н. Контроль		*Швець С.І.*			
Зав.кафедри		Азаров О.Д.			

Літ.	*Маса*	*Масштаб*
Арк. 139	*Аркушів* 1	
ВНТУ. гр. 1КІ-23м		

ДОДАТОК Д

Перелік елементів

Поз.	Найменування	Кіл.	Примітка
	Модулі		
	Модуль Arduino Nano	1	
	Резистори		
R1,3,5	470 кОм	3	
R2,4,6	680 кОм	3	
	Конденсатор		
C1	100nF	1	

Змн.	Лист	№ докум.	Підпис	Дата	08-54.МКР.013.00.000 ПЕЗ		
Розроб.		Мушинський В.Є			Метод та мікропроцесорні засоби вимірювання параметрів електронних компонентів. Перелік елементів	Літ.	Арк. 140 / Аркушів 1
Перевір.		Богомолов С.В.					
Опонент		Катєльніков Д.І.				ВНТУ, гр.1КІ-23м	
Н. Контр.		Швець С.І.					
Затверд.		Азаров О.Д.					

ДОДАТОК Е

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи:_____ Метод та мікропроцесорні засоби вимірювання параметрів електронних компонентів_____

Тип роботи:_____Магістерська кваліфікаційна робота_____

(кваліфікаційна робота, курсовий проект (робота), реферат, аналітичний огляд, інше (зазначити))

Підрозділ _____ кафедра обчислювальної техніки_____

(кафедра, факультет (інститут), навчальна група)

Керівник_____Богомолов С.В., доц. каф. ОТ_____

(прізвище, ініціали, посада)

Показники звіту подібності

Turnitin	
Оригінальність	87%
Загальна схожість	13%

Аналіз звіту подібності (відмітити потрібне)

- ✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
- ☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.
- ☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений (-на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається)

Автор _____ _____ Мушинський В.Є._____

(підпис) (прізвище, ініціали)

Опис прийнятого рішення

_____

_____

_____

_____

Особа, відповідальна за перевірку_____ _Захарченко С.М__

(підпис) (прізвище, ініціали)

Експерт _____ _____

(за потреби) (підпис) (прізвище, ініціали, посада)