Co-funded by the
Tempus Programme
of the European Union

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Запорізький національний технічний університет

Пархоменко А.В., Гладкова О.М.,
Залюбовський Я.І., Пархоменко А.В.

ІНЖЕНЕРІЯ ВБУДОВАНИХ СИСТЕМ

Навчальний посібник

Запоріжжя
Дике Поле
2017

УДК 004.31:621.3.049.77(075.8)
ББК 32.973.2я73
І-62

Рекомендовано до друку Вченою радою Запорізького національного технічного університету (протокол № 8 від 27 березня 2017 року)

Рецензенти:

Теслюк В.М., доктор технічних наук, професор кафедри систем автоматизованого проектування Національного університету «Львівська політехніка».

Тарасов О.Ф., доктор технічних наук, професор, завідувач кафедри комп'ютерних інформаційних технологій Донбаської державної машинобудівної академії

І-62 **Пархоменко А. В., Гладкова О. М., Залюбовський Я І. , Пархоменко А.В.**
Інженерія вбудованих систем: навчальний посібник.– Запоріжжя: Дике Поле, 2017. – 220 с.

ISBN 978-966-7433-15-5

Розглянуто особливості автоматизованого проектування апаратного та програмного забезпечення вбудованих систем з використанням середовищ Altium Designer, Proteus, Arduino IDE, Atmel Studio, Processing IDE, 123D Circuits. Наведено практичні приклади створення та дослідження різних схем та пристроїв. Посібник призначений для студентів вищих навчальних закладів, які навчаються за спеціальностями «Комп'ютерні науки», «Інженерія програмного забезпечення», а також може використовуватись інженерами, аспірантами і студентами різних спеціальностей для отримання знань та практичного досвіду розробки вбудованих систем.

Посібник видано за підтримки міжнародного проекту DESIRE «Розробка курсів з вбудованих систем з використанням інноваційних віртуальних підходів для інтеграції науки, освіти та промисловості в Україні, Грузії та Вірменії» (544091-TEMPUS-1-2013-1-BE-TEMPUS-JPCR) за програмою TEMPUS Європейської комісії.
Проект фінансується за підтримки Європейської Комісії. Зміст даної публікації/матеріалу відображає думку авторів і Європейська Комісія не несе відповідальність за використання інформації, що в ній міститься.

ISBN 978-966-7433-15-5

ЗМІСТ

ПЕРЕДМОВА/PREFACE

The goal of the international educational TEMPUS project DESIRE «Development of Embedded System courses with implementation of Innovative virtual approaches for integration of research, education and production in Ukraine, Georgia and Armenia» (http://tempus-desire.eu/) is to adapt current curricula from theoretical and desk-top application based curricula towards a practice oriented and embedded systems based education. Embedded systems (ES) design and production demands a very specific, qualitative and valuable knowledge growth in target HEIs, which ensures efficient implementation of high-skilled people in the labor market.

The curricula are developed according competences needed in local and international labor market. Competences are distilled from interviews with local and international enterprises in target countries and EU, academic staff and students. To come to the stated goals and competences, curricula are adapted and new course material is provided, supported with practical lab equipment, embedded hardware platforms, remote laboratories and design software for CAD/CAM/CAE.

Our partners from Zaporizhzhya National Technical University (ZNTU) implemented ES courses in the specialties «Engineering of Software» and «Computer Science». In the framework of new ES courses, developed in ZNTU, students deal with studying of modern remote and virtual tools for embedded systems design.

A new Laboratory of Embedded Systems is established at ZNTU, as one of the main results of DESIRE project, equipped with a modern server, state-of-the-art computers, interactive learning tools & educational set-up, several products for 3D-industry (3D-printer and 3D-scanner), Formula Flowcode Buggy and LEGO MINDSTORMS platforms, different boards – Arduino, Raspberry Pi, Altera Cyclone, as well as the CAD systems – Altium Designer and CREO.

Thanks to this project ZNTU has got new possibilities for spreading the ideas of practical-oriented engineering education and find new industrial partners for sustainable cooperation in education and research. I hope that this textbook will help to improve students' knowledge and skills in the field of ES design for their future successful career.

Ing. Dirk Van Merode MSc.
Coordinator of the DESIRE project,
TMMA, Belgium

ЧАСТИНА 1. ПРОЕКТУВАННЯ ВБУДОВАНИХ СИСТЕМ В СЕРЕДОВИЩІ ALTIUM DESIGNER

Вступ до частини 1

Світовий ринок вбудованих систем (ВС) постійно зростає, про що свідчать дослідження, проведені різними консалтинговими компаніями в останні роки [1,2,3]. Експерти прогнозують, що ринок ВС у найближчі 10 років зросте вдвічі [4]. З кожним роком вбудовані системи стають дедалі складнішими та вимагають більш високого рівня надійності та стійкості, при цьому ринкові умови вимагають постійного скорочення часу розробки проектів.

Вбудована система (англ. Embedded System) визначається сьогодні як спеціалізована обчислювальна система, яка в силу розв'язуваної задачі безпосередньо взаємодіє з фізичними об'єктами і процесами. До складу простої ВС входять [5]:

- мікропроцесорний модуль з пам'яттю;
- периферійна система (датчики, виконавчі елементи і контролери введення-виведення для зв'язку з об'єктом контролю/управління, пристрої людино-машинного інтерфейсу за необхідністю);
- система електроживлення;
- конструктив, що об'єднує (шасі, корпус);
- програмне забезпечення, що керує.

Вбудовані системи значно відрізняються від систем загального призначення, наприклад персональних комп'ютерів. Основними особливостями ВС вважаються [5-7]:

- робота в реальному масштабі часу;
- багатозадачність;
- обмежені ресурси;
- мінімізація вартості;
- програмно-апаратний дуалізм;
- різноманітні, часто складні, умови експлуатації;
- автономність роботи (відсутність оператора, обмеження електроживлення);
- високі вимоги щодо надійності і безпеки функціонування,

особливо коли ВС використовуються у відповідальних областях, де треба враховувати всі зовнішні фактори, які можуть вплинути на надійність (природні або людські);

- критичні застосування, пов'язані зі здоров'ям і життям людини.

ВС відносяться до категорії систем з переважно програмною реалізацією (Software-Intensive або Software-Dominated Systems). Дослідження проведені у 2013 році [8] компаніями UBM Tech Electronics та EE Times Group свідчать про те, що понад 60% ресурсів витрачається на розробку програмного забезпечення (ПЗ). Цей показник майже не змінився і протягом 5 років коливався в межах одного выдсотку. Термін «вбудоване програмне забезпечення» (Embedded Software) підкреслює особливі властивості такого ПЗ та потребує спеціальних технологій його створення [9-13].

Елементну базу ВС складають електронні, оптичні, механічні та інші фізичні компоненти (елементи, модулі, блоки), з яких складається фізична реалізація ВС. Сьогодні в переліку таких компонентів складні мікросхеми процесорів (мікропроцесори), контролерів, акселераторів, системні плати обчислювачів. В свою чергу, до складу таких елементів входять програмні засоби (завантажувачі, стеки протоколів та інші), що розміщуються у вбудованих блоках постійної пам'яті. Таким чином, навіть традиційне уявлення обчислювальної елементної бази виходить далеко за межі опису тільки конструкції і схемотехніки, торкаючись все більше питань системотехніки, програмування, архітектури [5].

Проблема проектування ВС полягає в постійному їх ускладненні за рахунок зростання вимог до функціональності виробів [14]. Існування різноманітної елементної бази збільшує простір проектних рішень, але в той же час ускладнює вибір найбільш ефективного варіанту реалізації. Це одночасно збільшує ризик виникнення помилки та час пошуку оптимального рішення, що є неприйнятним в умовах необхідності скорочення термінів розробки проекту.

Тому, актуальною є задача розвитку технологій повторного використання компонентів апаратного та програмного забезпечення при проектуванні ВС [15-17], рекомендаційних систем та систем прийняття проектних рішень для розробці ВС, а також розвиток віртуального та віддаленого інструментарію для систем автоматизованого проектування ВС [18-23].

При проектуванні ВС надзвичайно важливим етапом є процес розробки концепції та моделювання роботи пристрою. За допомогою ECAD-систем (Electronic Design Automation) можна віртуально збирати електронні схеми, використовуючи величезний набір елементів (в т.ч. мікроконтролерів), що існують в стандартних бібліотеках, а також проектувати друковані плати. Деякі з цих програм надають можливість моделювати роботу створеної системи, завантаживши в програму файл для мікроконтролера з «.hex» розширенням. Це дозволяє виявляти помилки, допущені при програмуванні або проблеми підключення компонентів без створення реального прототипу. Відомими та розповсюдженими на цей час ECAD-системами є наступні.

Proteus VSM – пакет програм для автоматизованого проектування електронних схем [24]. Розробка компанії Labcenter Electronics (Великобританія). Дозволяє зібрати схему будь-якого електронного пристрою і симулювати його роботу, виявляючи помилки, що допущені на стадії проектування.

Пакет Proteus складається з двох основних модулів:

- ISIS – графічний редактор принципових схем, який служить для введення розроблених проектів з подальшою імітацією і передачею для розробки друкованих плат в ARES;
- ARES – графічний редактор друкованих плат з вбудованим менеджером бібліотек, автотрасувальником Electra і автоматичною системою розстановки компонентів на друкованій платі. Крім цього, в ARES можна створити тривимірну модель друкованої плати.

AutoCAD Electrical – надає більшість функцій програмного забезпечення AutoCAD, крім того містить унікальні інструменти для автоматизації процесів створення схем, компонування креслень, генерації звітів та ін.[25]. Додаток працює як з цілими проектами, так і з окремими компонентами, наприклад, програмованими логічними контролерами. Проекти можуть включати в себе принципові схеми, схеми автоматизації, креслення компоновок, схеми з'єднань, монтажні плани, різноманітні звіти.

Altium Designer - система автоматизованого проектування від розробників легендарного P-CAD, що надає широкі можливості по створенню електронних пристроїв [26-29].

Altium Designer включає весь необхідний набір інструментів для створення, редагування та інших робіт на основі електричних і програмованих інтегральних схем.

Редактор схем дозволяє працювати з проектами будь-якого розміру та складності. Цифро-аналогове моделювання враховує майже всі реальні параметри і надає в розпорядження конструктора величезну кількість різних видів аналізу, включаючи аналіз перехідних процесів, частотний, температурний, шумів, передаточних функцій, Фур'є, методом Монте Карло.

Редактор друкованих плат програми містить унікальні засоби для автоматичного і інтерактивного розміщення компонентів. Постійно оновлюванні бібліотеки програми зберігають понад 90 тисяч компонентів. Багато з них мають моделі посадочних місць, IBIS і SPICE-моделі, а також 3D-моделі. Кожну з них можна створити в програмі самостійно з мінімальними витратами часу шляхом послідовного введення відомостей про компоненти.

EAGLE (англ. Easily Applicable Graphical Layout Editor) — система проектування електричних принципових схем і друкованих плат [30]. EAGLE користується популярністю серед західних радіоаматорів через свою безкоштовну ліцензію і доступність в Інтернет великої кількості бібліотек компонентів. Адаптована до систем Windows, Linux, Mac OS X.

123D Circuits - це онлайн-сервіс компанії Autodesk для проектування електронних схем і друкованих плат, що включає підтримку апаратно-обчислювальної платформи Arduino [31]. Основною особливістю 123D Circuits є імітація платформи Arduino з можливістю редагування програмного коду з браузера в візуальному режимі. Після закінчення роботи в графічній частині, програма автоматично генерує принципову електричну схему розроблювального пристрою і пропонує власний варіант розміщення компонентів на друкованій платі (з можливостями редагування). Працювати над проектами можна спільно з іншими користувачами, використовуючи єдину бібліотеку компонентів.

У цій частині будуть розглянуті інструменти системи Altium Designer, що надає проектувальникам весь спектр можливостей для проектування електронних схем, друкованих плат, а також програмованих логічних інтегральних схем (ПЛІС).

1 ЗНАЙОМСТВО З ІНТЕРФЕЙСОМ СИСТЕМИ АВТОМАТИЗОВАНОГО ПРОЕКТУВАННЯ ALTIUM DESIGNER

1.1 Основи інтерфейсу Altium Designer

В головному вікні програми (рис. 1.1) представлені сім груп елементів:

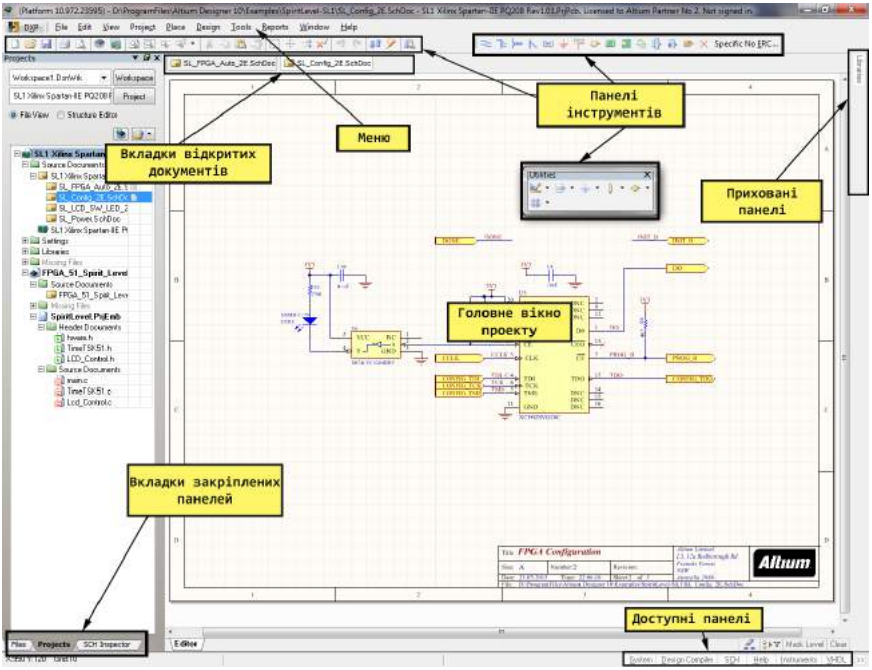


Рисунок 1.1 – Головне вікно програми Altium Designer

- рядок меню;
- головне вікно проекту;
- панелі інструментів;
- вкладки відкритих документів;
- приховані панелі робочої області;
- вкладки закріплених панелей робочої області;
- доступні панелі робочої області.

1.1.1 Панелі інструментів

Для перегляду списку доступних панелей (рис.1.2) або для створення власних панелей інструментів потрібно натиснути праву клавішу миші на вільному місці у рядку меню або з підменю View > Toolbars.

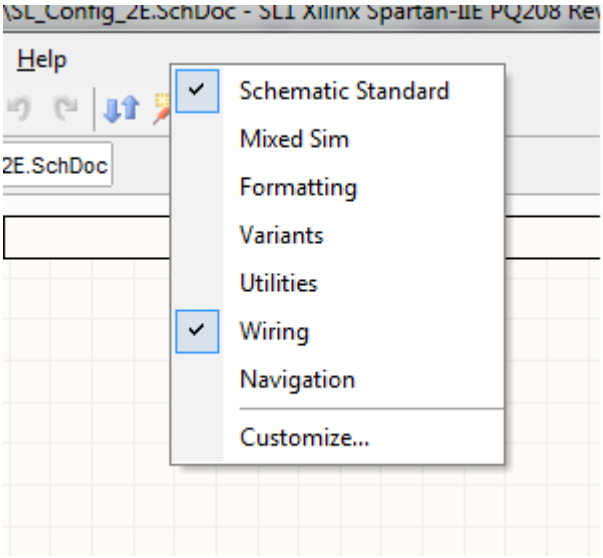


Рисунок 1.2 – Список доступних панелей інструментів

1.1.2 Панелі робочої області

Всі поточні доступні панелі робочої області (не плутати з панелями інструментів View > Toolbars) можна відкрити за допомогою рядка кнопок у правому нижньому куті вікна програми (на рис. 1.1 - це поле «Доступні панелі»). Кожна кнопка позначена ім'ям категорії панелей, які вона може відкрити.

Панелі робочої області також можна відкрити з підменю View > Workspace Panels (рис.1.3).

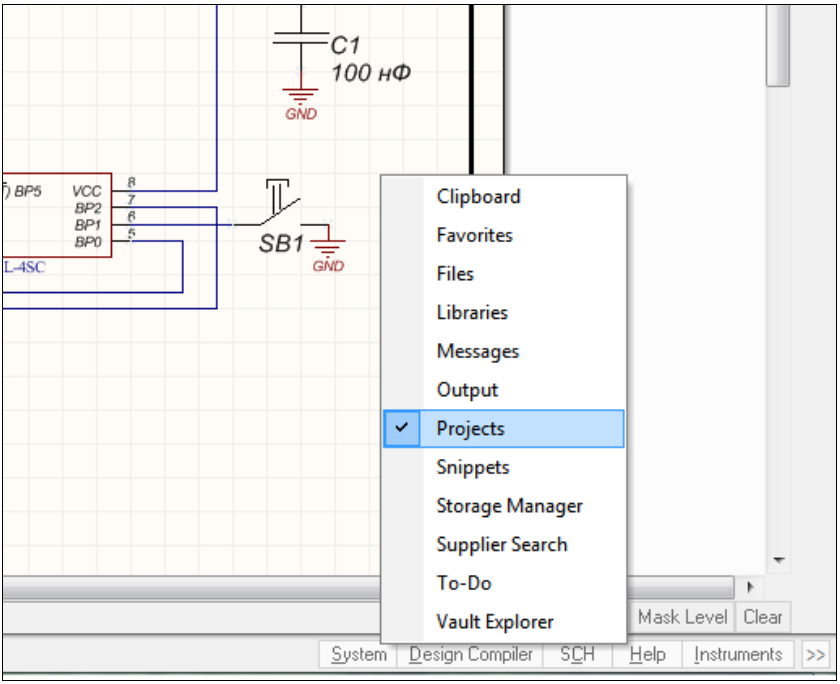


Рисунок 1.3 – Доступні панелі робочої області

1.1.3 Загальні графічні команди

Стандартні команди «малювання» доступні з панелі інструментів «Utilities» в випадаючому списку «Utility Tools» або підменю Place > Drawing Tools (рис.1.4). Піктограми команд «малювання» наведені в таблиці 1.1.

Перед тим, як розмістити вибраний об'єкт можна перевірити або задати його властивості, для чого необхідно натиснути клавішу Tab.

Редагування параметрів вибраного об`єкта здійснюється за декількома параметрами (рис.1.5).

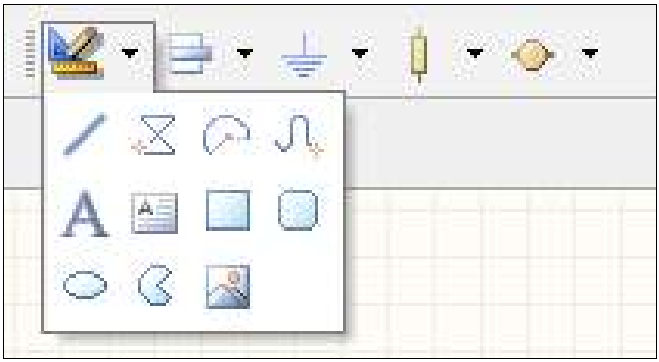


Рисунок 1.4 – Команди «малювання»

Таблиця 1.1 – Піктограми основних команд «малювання»

Піктограма	Назва команди	Призначення команди
	Place Line	Розміщення лінії
	Place Polygons	Розміщення полігону
	Place Electrical Arcs	Розміщення дуги
	Place Beziers	Розміщення сплайну (крива Безьє)
	Place Text String	Розміщення текстового рядка
	Place Text Frame	Розміщення текстової області
	Place Rectangle	Розміщення прямокутника
	Place Round Rectangle	Розміщення прямокутника із округленими кутами
	Place Ellipses	Розміщення еліпса
	Place Pie Chart	Розміщення сектора кола
	Place Graphic Image	Розміщення малюнку

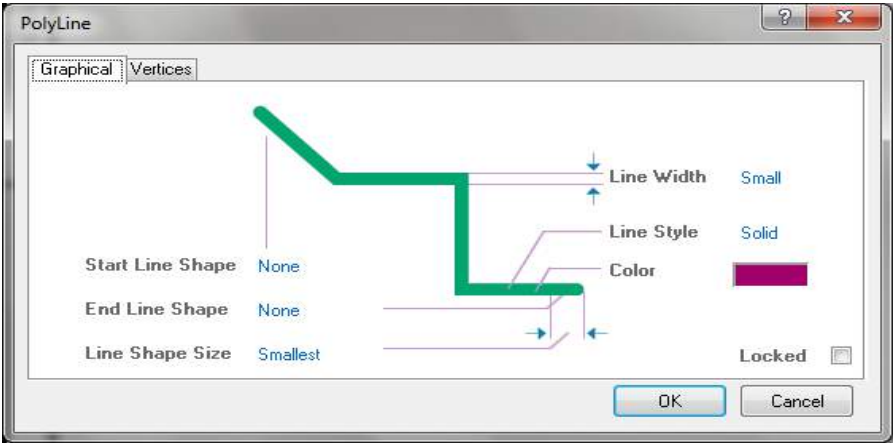


Рисунок 1.5 – Редагування параметрів вибраного об`єкта

Наприклад, для об'єкта PolyLine можна задати такі параметри:

- Start Line Shape – фігура, якою починається лінія;
- End Line Shape – фігура, якою закінчується лінія;
- Line Shape Size - розмір цих фігур;
- Line Width --ширина лінії;
- Line Style - стиль (суцільна, пунктирна, точкова);
- Color - колір.

Для зміни кута нахилу при малюванні потрібно натиснути клавішу SPACE, а для завершення малювання – праву клавішу миші.

1.1.4 Типи проектів. Створення проекту та додавання документів

Altium Designer підтримує різні типи проектів.

Проект плати – PCB Project (*.PrjPcb). Набір документів, необхідних для виготовлення друкованої плати. Електронна схема створюється у редакторі схем із бібліотечних символів, які розміщуються на аркуші та з’єднуються провідниками. Проект передається до редактору плат, де кожен компонент представляється як посадкове місце (корпус), провідники на схемі перетворюються у

лінії, що з'єднують, від виводу до виводу. Визначається кінцевий вигляд плати разом з фізичними шарами плати. Описуються правила проектування для виготовлення фотошаблону, такі, як ширина провідника і відстань між ними. Компоненти розташовуються у межах контуру плати та з'єднуються лініями зв'язку, які потім замінюються трасами вручну або автоматично.

Проект ПЛІС (Програмована логічна інтегральна схема) – FPGA (Field-programmable gate array) Project (* .PrjFpg). Набір документів, які можуть бути оброблені для програмування ПЛІС. Проект створюється за допомогою редактора схем та/або програмування мовою HDL (**H**ardware **D**escription **L**anguage) Це може бути VHDL (**V**ery high speed integrated circuits **HDL)** або Verilog HDL. Додаються файли обмежень в проект для опису вимог проекту таких, як програмований пристрій, внутрішній розподіл виводів для кіл і пристроїв, вимоги до швидкодії, визначення частот на виводах.

Інтегрована бібліотека - integrated library (* .IntLib). Ім'я файлу оболонки * .LibPkg; ім'я файлу бібліотеки * .IntLib. Умовні графічні відображення та посадкові місця компонентів формуються в редакторі бібліотек для створення інтегрованої бібліотеки. Символи компонентів для схеми створюються засобами бібліотечного редактора символів і для них визначається модельне уявлення. До символу можуть бути додані 4 типи моделей, наприклад, опис посадкового місця компоненту на платі, дані для схемного моделювання, моделювання цілісності сигналів і тривимірні моделі. Файли, які містять моделі, додаються в Integrated library Pakage (*.LibPkg) або визначаються шляхи пошуку для ідентифікації та їх розташування. Вихідні схемні бібліотечні символи і необхідні моделі потім компілюються у єдиний файл, який називається інтегрованою бібліотекою.

Вбудований проект – Embedded project (*.PrjEmb). Набір документів, необхідних для виробництва прикладного програмного забезпечення, яке може бути застосоване у частині керуючого процесора в електронному пристрої. Вихідний проект формується на мові С та/або асемблері.

Скрипт-проект – Script Project (* .PrjScr). Програмування в середовищі Altium Designer має за мету модифікацію об'єкта в інших відкритих проектах. Для управління використовується інтерфейс створення програм API (Application Program Interface).

Для створення проекту необхідно виконати наступне.

Використовуйте підменю File > New > Projects. Слід відзначити, що даний проектний файл існує лише у пам’яті при первинному створенні, тому використовуйте команди Save або Save As для збереження його з необхідною назвою на жорсткому диску. Імена файлів для проектів FPGA, Core та Embedded не повинні містити пропуски. Після створення та збереження проекту виникає необхідність додавання документів у поточний проект. Для цього потрібно виконати команду з підменю Project > Add New to Project та вибрати потрібний тип документу (рис.1.6). Перелік можливих типів документів, що підключаються до проекту:

- Schematic – схемний документ;
- PCB – файл друкованої плати;
- Schematic Library – бібліотека схемних символів (УГП - умовні графічні позначення);
- PCB Library – бібліотека топологічних посадкових місць (ТПМ);
- CAM Document – документ CAM-програми;
- Output Job File – файл вихідних даних для обробки;
- Database Link File – файл-вказівник зв’язку з базою даних;
- Text Document – текстовий документ;
- Other – інші документи.

На разі підключення уже існуючого документу необхідно вибрати команду з підменю Project > Add Existing to Project та вибрати потрібний документ. Якщо відкрито декілька проектів, то для додавання документу потрібно правою клавішею миші натиснути на потрібному проекті та вибрати потрібний пункт.

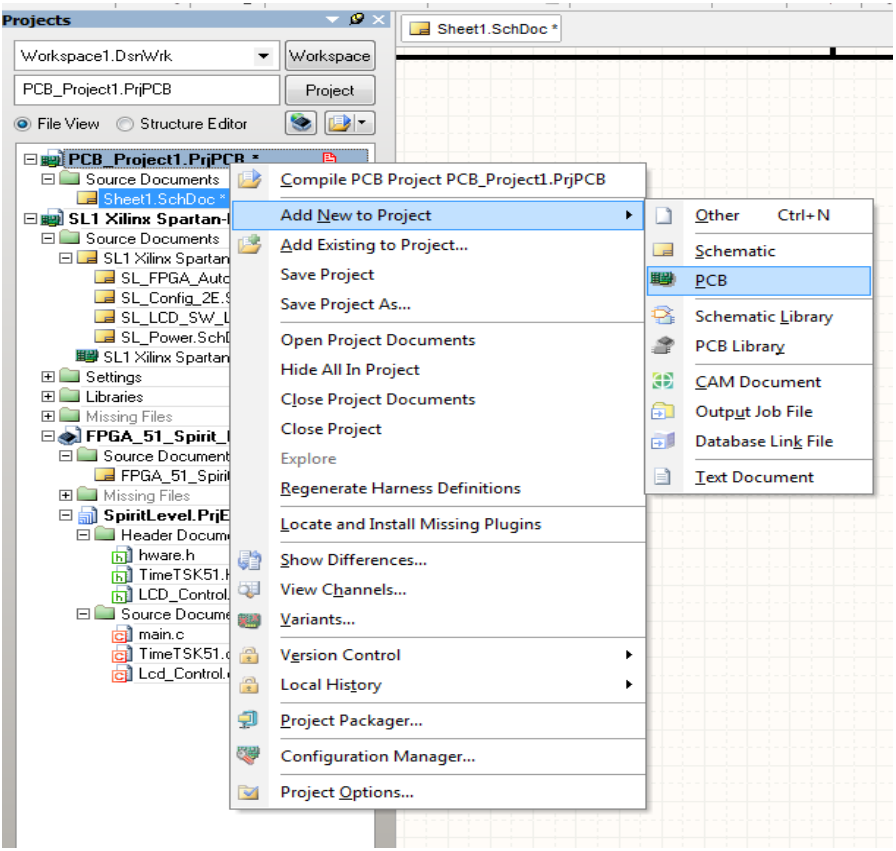


Рисунок 1.6 – Додавання нового документу до вибраного проекту

1.1.5 Основні терміни у Altium Designer

Найбільш часто використовувані терміни стосовно роботи у середовищі Altium Designer:

Component (Компонент) – загальне найменування об'єкта, який може бути застосований в проекті.

Symbol (Символ) – загальне найменування умовного графічного позначення (УГП) компонента, підготованого для розміщення на схемі. Символ може містити графічні об'єкти, які визначають

зовнішній вигляд, а також виводи, які визначають електричні струми підключення. В системі Altium Designer символ, по суті, є компонентом, оскільки є повністю завершеним об'єктом, який може бути використаний при створенні схем і до нього можуть бути підключені моделі різного типу.

Part (Частина, гейт, секція) – деякі компоненти, наприклад такі, як резисторні кола або реле можуть бути побудовані як серія окремих секцій, які в свою чергу можуть бути розміщені на схемі незалежно, при цьому на платі встановлені у вигляді єдиного корпусу.

Model (Модель) – уявлення компонента, який використовується в деякій практичній сфері діяльності. Так, для створення плати використовуються моделі посадкових місць, при моделюванні - моделі Spice.

Footprint (Топологічне посадкове місце - ТПМ) - найменування моделі, яка представляє компонент на заготівлі друкованої плати. Посадкове місце групує набір контактних майданчиків на платі і контур компонента.

Pad (Контактна площинка, КП) – зображення виводу елемента на платі.

Pin (Вивід) – зображення виводу елемента на схемі.

Бібліотека – файл містить набір компонентів і набір моделей.

Бібліотека моделей – файл містить набір моделей компонентів.

Бібліотека компонентів – файл містить набір схемних компонентів.

1.2 Практична робота с інтерфейсом

1.2.1 Редагування властивостей графічних об'єктів

Для того, щоб змінити параметри об'єкта, розміщеного раніше, необхідно навести на нього курсор і натиснути ліву кнопку миші, після чого в об'єкта з'явиться зелений контур і мітки редагування (рис.1.7). Щоб змінити форму об'єкта, потрібно навести курсор на зелену мітку редагування до появи діагонального курсора і, потім, переміщати із натиснутою лівою кнопкою миші. Для зміни положення

об'єкта, потрібно навести курсор на вільне від зелених міток місце на об'єкті до появи курсора у вигляді хрестика і, потім переміщати із натиснутою лівою кнопкою.

При переміщенні об'єкта, утримуючи клавішу Ctrl, він не відривається від своїх зв'язків.

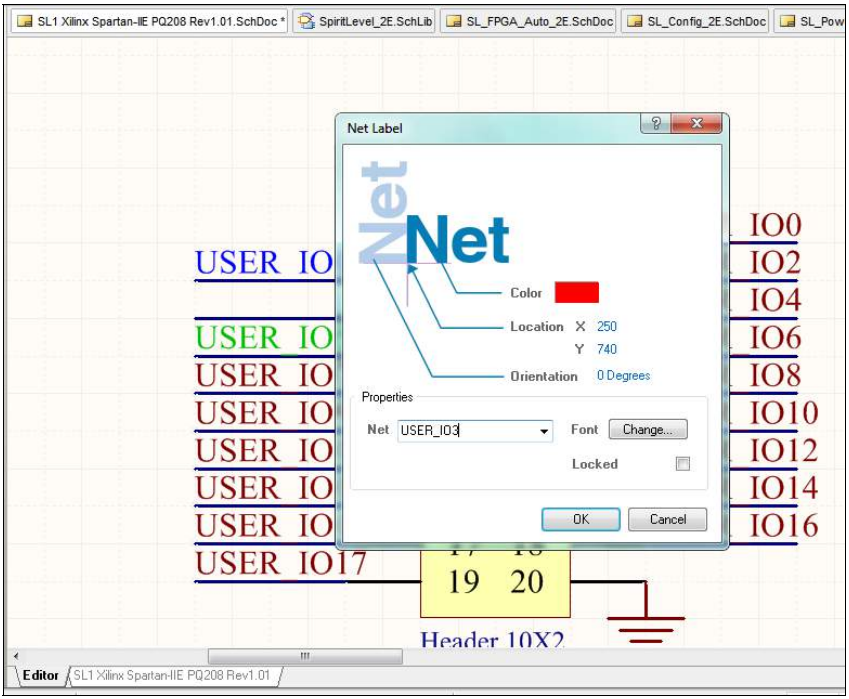


Рисунок 1.7 – Редагування мітки кола

Масштабування зображення здійснюється при прокручуванні колеса миші з натиснутою клавішею Ctrl, або при натисканні клавіш PageUp - збільшення масштабу, PageDown - зменшення.

Для редагування властивостей об'єкта потрібно виконати подвійне натиснення лівої кнопки на його зображенні або вибрати пункт Properties з контекстного меню при натисканні правої кнопки миші.

Редагування властивостей відразу декількох об'єктів неможливо виконати наведеними вище способами, це особливість даної системи проектування. Для зміни властивостей групи об'єктів (також, рекомендується застосовувати цей спосіб і для зміни властивостей одного об'єкта) використовується панель Inspector (рис.1.8), яка

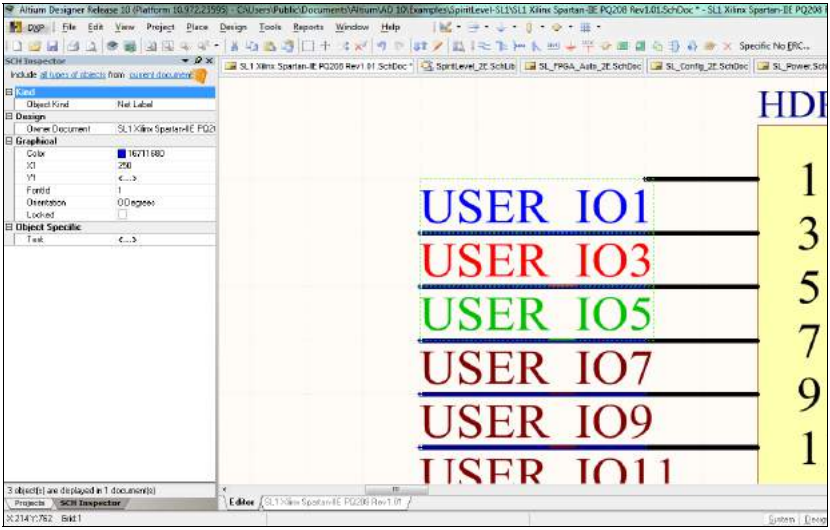


Рисунок 1.8 – Параметри виділених об’єктів

викликається натисканням клавіші F11. (Виділення декількох об'єктів необхідно виконувати, утримуючи SHIFT).

В панелі Inspector представлені властивості виділених об'єктів в табличному вигляді, доступні для редагування. У нижній частині панелі написано про те, скільки об'єктів вибрано в поточний момент. Якщо значення параметра відображається в таблиці (наприклад, Orientation - 0 Degrees), то це значення однакове для всіх обраних об'єктів, якщо в рядку параметра вказано <...> - значення цього параметра у різних об'єктів відрізняється.

У таблиці відображаються тільки ті параметри, які є у всіх виділених об'єктів.

Наприклад, встановимо для міток кіл, що мають синій, червоний і зелений кольори, такий же колір як і решта міток компонента.

В панелі Inspector також є вбудований фільтр (рис.1.10), в якому можна вказати, для яких типів об'єктів відображати властивості, а також область застосування (для поточного документа, для всіх відкритих документів, для всіх документів того ж проекту).

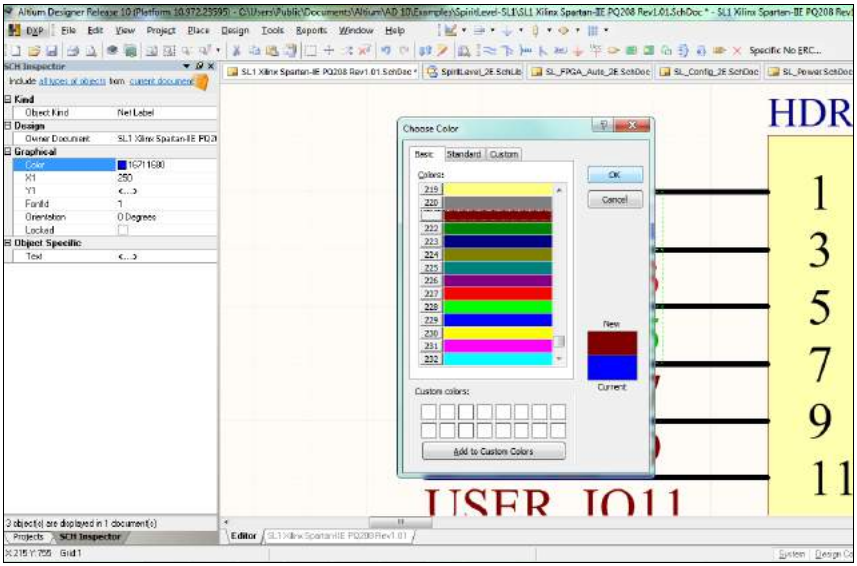


Рисунок 1.9 – Вибір кольору об’єктів

Рисунок 1.10 – Фільтр панелі Inspector

1.2.2 Пошук подібних об'єктів

Ця функція дозволяє виділити необхідні об'єкти відразу в декількох документах. Наприклад, відкриємо кілька документів з якого-небудь проекту, що є в стандартній бібліотеці прикладів

(Examples> SpiritLevel-SL1> SL1 Xilinx Spartan-IIE PQ208 Rev1.01.PrjPcb) і спробуємо виділити всі конденсатори у відкритих документах, для чого виділимо один з конденсаторів і викличемо його контекстне меню натисканням правої клавіші миші.

Вибравши пункт Find Similar Objects (рис.1.11) з контекстного меню, відкриється вікно, в якому зазначаються критерії виділення об'єктів (рис.1.12). В останній колонці вказується важливість критерію при пошуку:

- Any – критерій неважливий для пошуку;
- Same – критерій важливий для пошуку;
- Different – виключення, тобто вибрати все окрім цього.

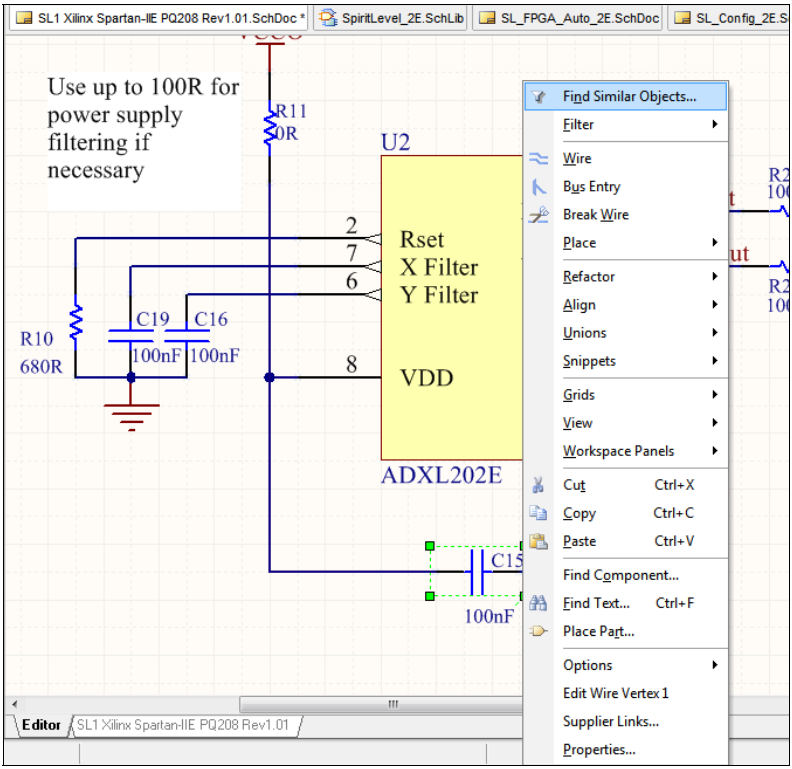


Рисунок 1.11 – Пошук подібних об’єктів

Щоб виділити конденсатори, вкажемо, що всі об'єкти, які ми хочемо виділити, мають умовне позначення, що починається на літеру C (наприклад C8 або C11). Для цього, в полі Component Designator вкажемо значення C*, де * - будь-яка кількість будь-яких символів і вкажемо, що цей критерій є важливий.

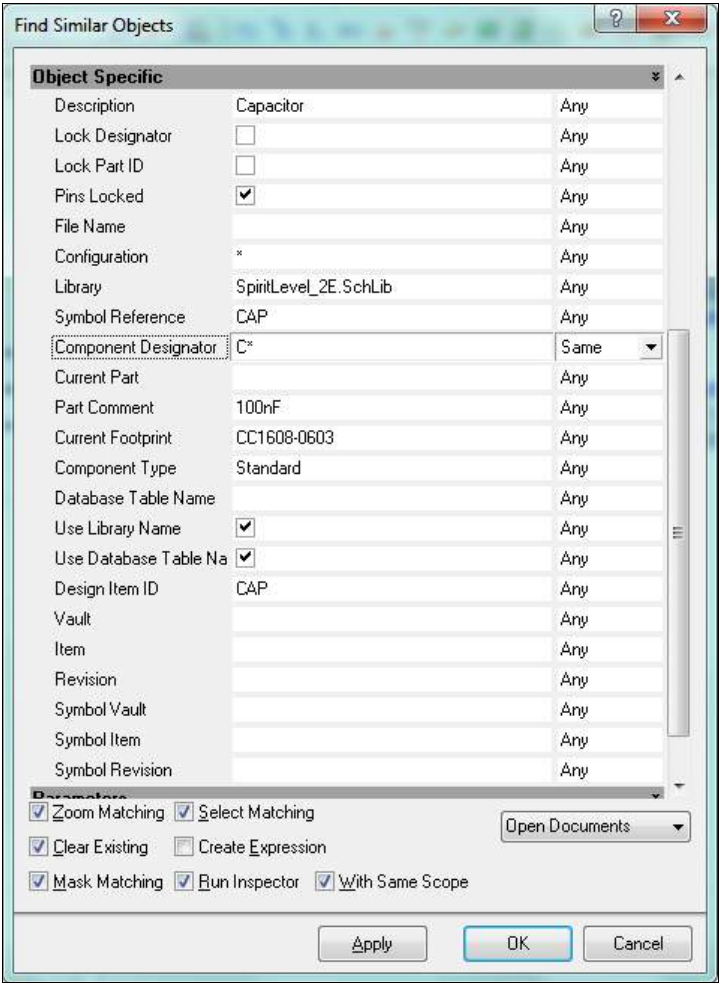


Рисунок 1.12 – Вибір критеріїв виділення об'єктів

У нижній частині вікна вказуються дії, які будуть виконані після пошуку об'єктів, а також область застосування пошуку:

- Zoom Matching - цей пункт говорить про те, що необхідно масштабувати отримані об'єкти;
- Select Matching - виділити об'єкти;
- Clear Existing - видалити попередні виділення;
- Mask Matching - включити маскування відфільтрованих об'єктів;
- Create Expression - сформувати фільтр для подальшого застосування.

Після натискання ОК, всі об'єкти, що не задовольняють критеріям пошуку, будуть затемнені, це говорить про те, що спрацювала маска. Для редагування властивостей всіх виділених об’єктів в панелі Inspector необхідно вказати область дії його фільтра, вказавши значення open documents.

Відключити маску можна кнопкою Clear в правому нижньому куті або клавішами SHIFT + C.

Пошук подібних об'єктів можна також здійснювати і для бібліотек.

2 СТВОРЕННЯ БІБЛІОТЕКИ ЕЛЕМЕНТІВ

2.1 Створення бібліотеки символів і посадкових місць

Для роботи з бібліотекою символів необхідно відкрити робочу панель (яка відкривається кнопкою SCH > SCH Library у правому нижньому куті) та розмістити її ліворуч від робочої області, після чого інтерфейс буде мати вигляд, показаний на рис. 2.1.

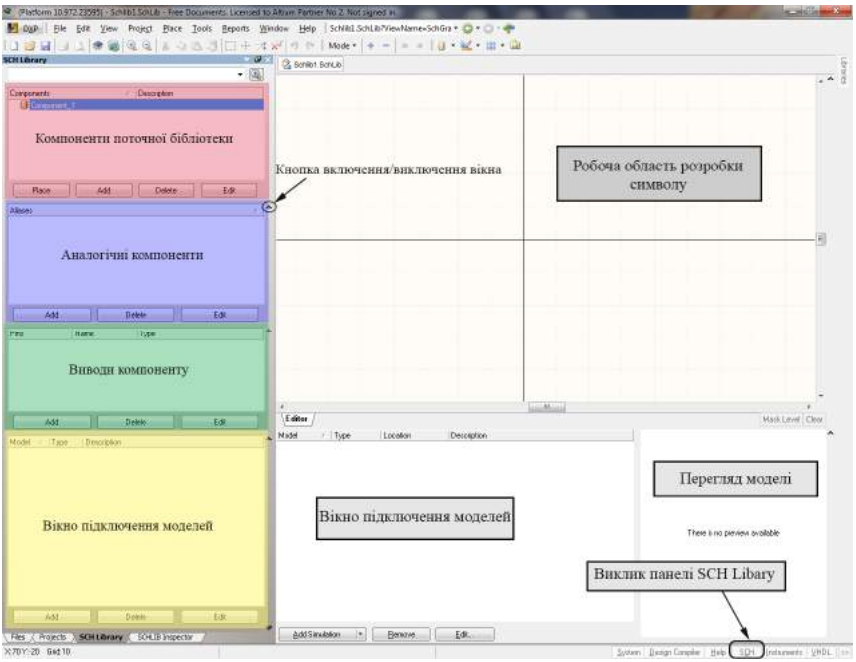


Рисунок 2.1 – Інтерфейс редактору компонентів

2.1.1 Початкові налаштування робочої області

Перш ніж розпочати розробку нового символу, потрібно налаштувати робочу область, тобто вибрати необхідні одиниці вимірювання та крок сітки. Дане налаштування здійснюється у вікні

Library Editor Options (рис. 2.2), яке викликається командою Tools > Document Options. У вікні на вкладці Units можна обрати одиниці вимірювання, вони можуть бути метричні або дюймові (у нашому випадку слід обрати міліметри).

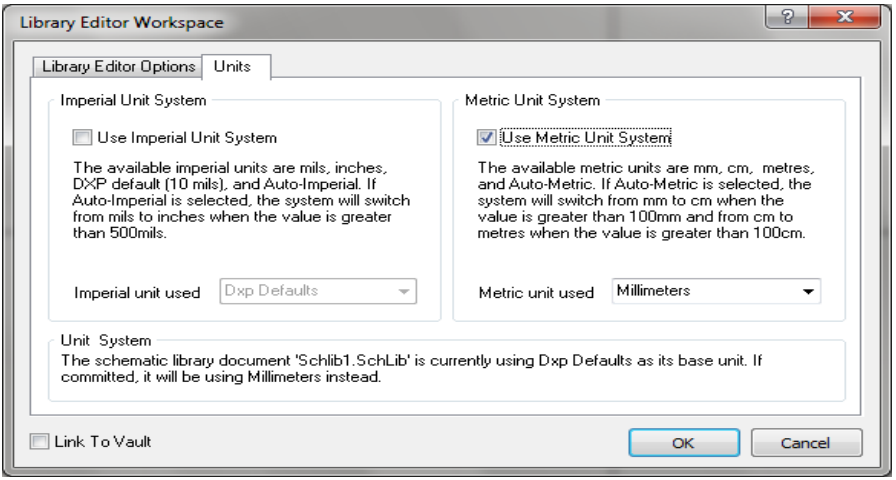


Рисунок 2.2 – Вибір одиниць вимірювання

У вкладці Library Editor Options (рис. 2.3) потрібно встановити крок сітки 2.5 мм, для двох видів сіток (Snap – сітка переміщення курсору у режимі графічної команди, Visible – сітка, яка відображається на екрані). Окрім сітки, у даній вкладці також можуть бути встановлені додаткові параметри.

2.1.2 Інтерфейс редактору посадкових місць. Параметри контактної площинки

Для роботи з редактором посадкових місць використовується панель PCB Library. Для відображення вказаної панелі її потрібно знайти у групі панелей PCB у правому нижньому куті екрану. Після чого інтерфейс програми набуде вигляду, як показано на рисунку 2.4.

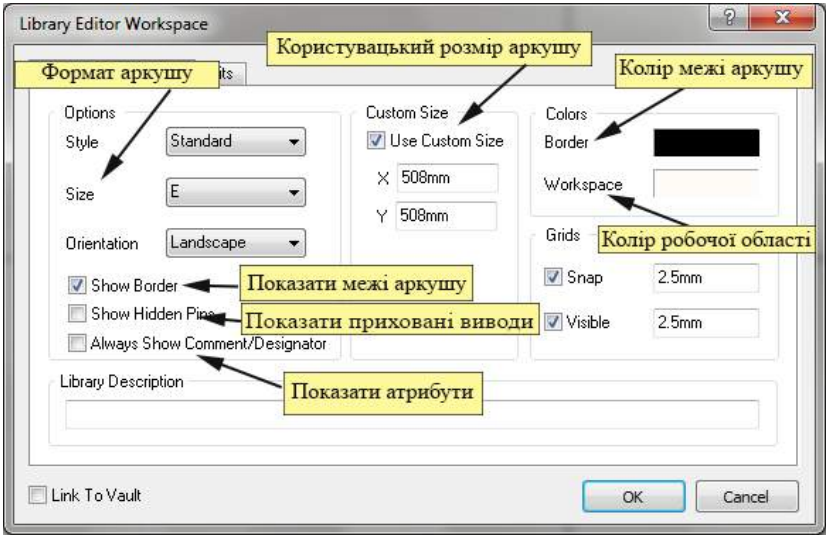


Рисунок 2.3 – Налаштування робочої області редактору символів

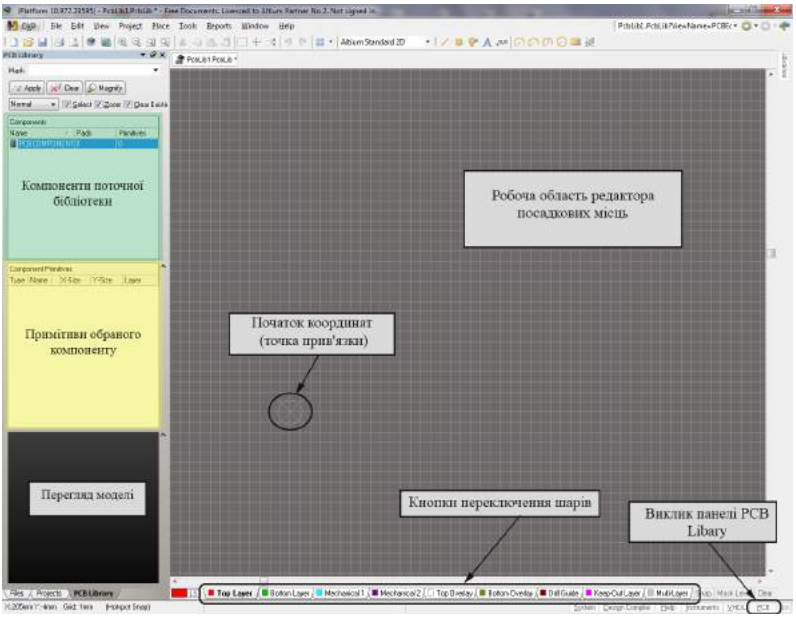


Рисунок 2.4 - Інтерфейс редактору посадкових місць

Переключення між шарами можна здійснювати за допомогою комбінації Ctrl + Shift + Scroll.

Після встановлення контактної площинки потрібно задати її параметри, натиснувши клавішу Tab, опис параметрів представлений на рисунку 2.5.

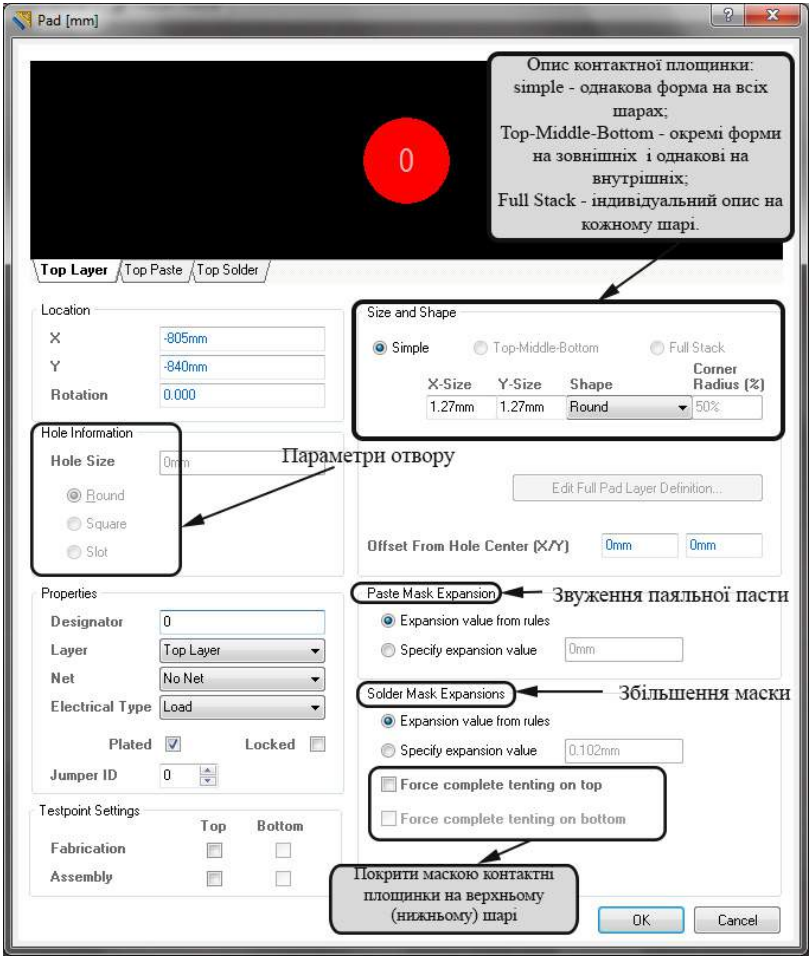


Рисунок 2.5 – Параметри контактної площинки

2.2 Практична робота з бібліотекою

2.2.1 Створення бібліотечного символу для резистора

Спочатку потрібно створити окрему бібліотеку символів за допомогою підменю File > New > Library > Schematic Library. Після створення, нову бібліотеку потрібно зберегти, це можна зробити за допомогою підменю File > Save або використавши комбінацію клавіш Ctrl + S.

Потрібно налаштувати робочу область. Встановити метричні одиниці вимірювання. Відкрити додаткову панель SCH Library за допомогою рядку кнопок у правому нижньому кутку вікна програми (рис. 2.6). У цій панелі зараз нам знадобляться лише вікна: Component (Компоненти або символи) та Pins (Виводи поточного компоненту), інші для зручності можна згорнути.

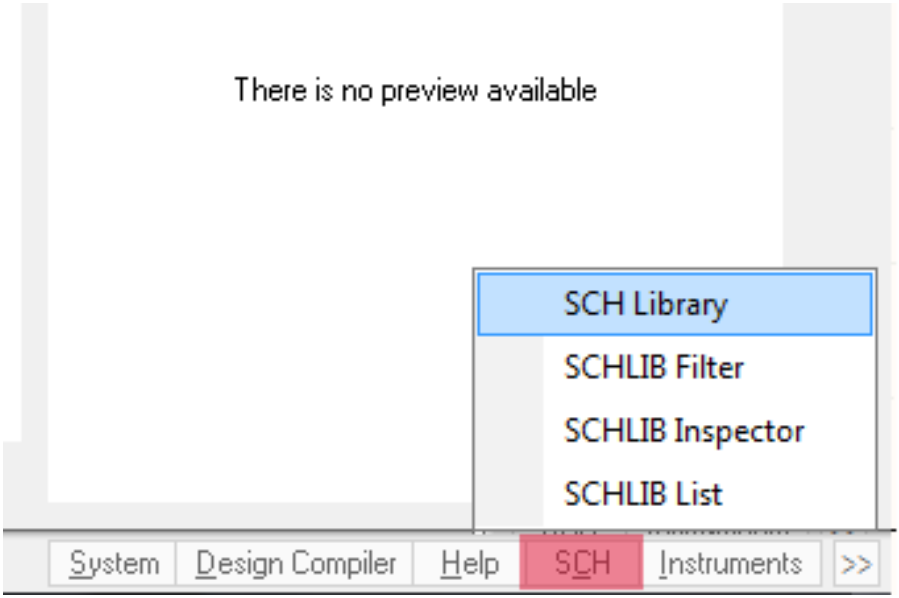


Рисунок 2.6 – Додавання панелі SCH Library

Створюємо новий компонент за допомогою клавіші Add у вікні Components або використавши підменю Tools > New Component та задаємо назву цього компонента – Resistor. Далі задаємо позиційне позначення, для чого натискаємо на клавішу Edit у вікні Components або через підменю Tools > Component Properties. У вікні, що з'явилося задаємо необхідні данні відповідно до рис. 2.7.

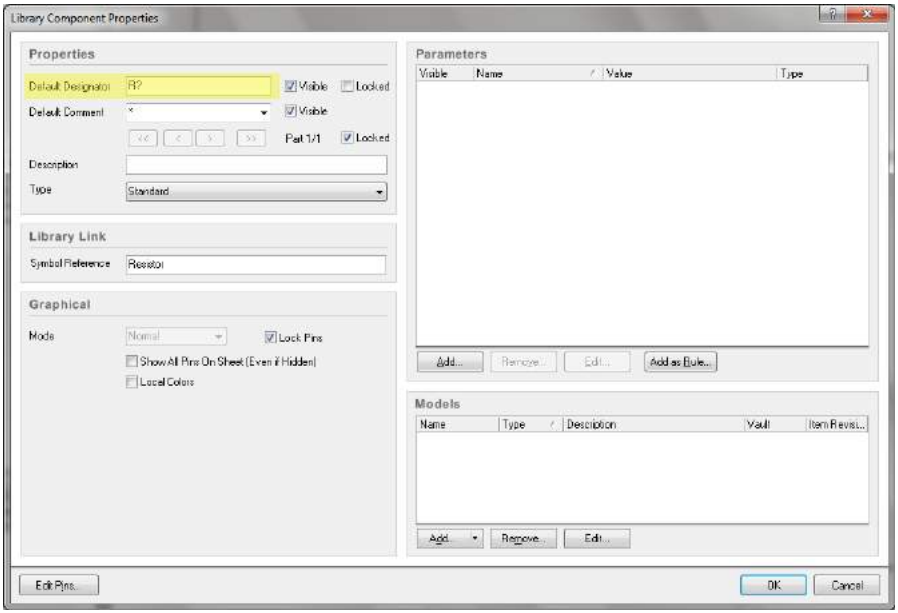


Рисунок 2.7 – Властивості компонента

У поле заносимо значення R?, де R – префікс, а ? згодом буде замінений у схемі на відповідну цифру.

Тепер потрібно встановити виводи компоненту. Спочатку встановлюємо крок сітки 2.5 мм за допомогою клавіші G, потім за допомогою підменю Place > Pin розміщуємо виводи компонента та змінюємо властивості кожного з них відповідно до рис.2.8, відкривши вікно за допомогою клавіші Tab.

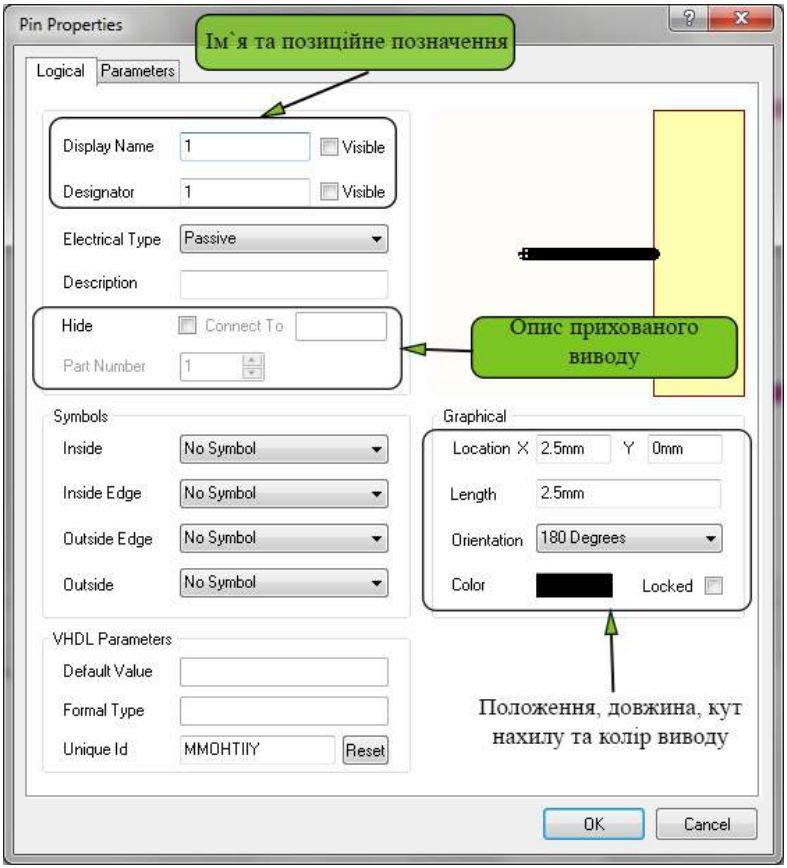


Рисунок 2.8 – Властивості виводу компонента

На виводі є електрична точка, яка відображена невеличким діагональним хрестиком (білого кольору). Вона позначає те місце, куди буде підводитися електричне з’єднання. У першого виводу електрична точка повинна бути ліворуч, у другого –праворуч.

Далі, попередньо встановивши крок сітки 0.5 мм, креслимо графіку (контур) компонента, використавши команду Place > Line (бажано, щоб колір ліній був відмінним від кольору виводів) за

розмірами, які відображені на рис. 2.9 та зберігаємо все комбінацією клавіш Ctrl + S.

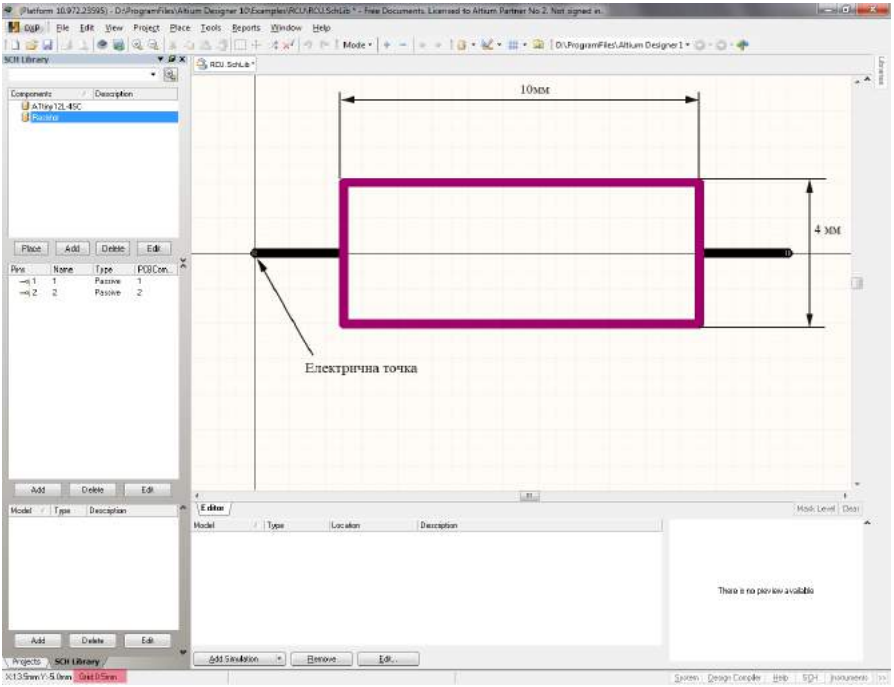


Рисунок 2.9 – УГП резистора

2.2.2 Створення топологічного посадкового місця для резистора

Спочатку потрібно створити окрему бібліотеку посадкових місць за допомогою підменю File > New > Library > PCB Library. Після створення, нову бібліотеку потрібно зберегти, це можна зробити за допомогою підменю File > Save або використавши комбінацію клавіш Ctrl + S.

Додаємо новий компонент, для цього натискаємо правою клавішею миші на вікні Components у панелі PCB Library і обираємо команду New Blank Component. Потім, подвійним натисканням на цей

компонент відкриваємо його властивості та у вікні, що з'явилося задаємо назву компонента – R_0805.

У центрі робочої області є точка початку координат, яка відображається, як на рисунку  . Перший вивід компонента будемо розміщувати у цю точку, для цього обираємо команду Place > Pad і клавішею Tab одразу задаємо необхідні параметри відповідно до рисунку 2.10 та розміщуємо компонент. Другий вивід розміщуємо праворуч на відстані 2 мм від першого.

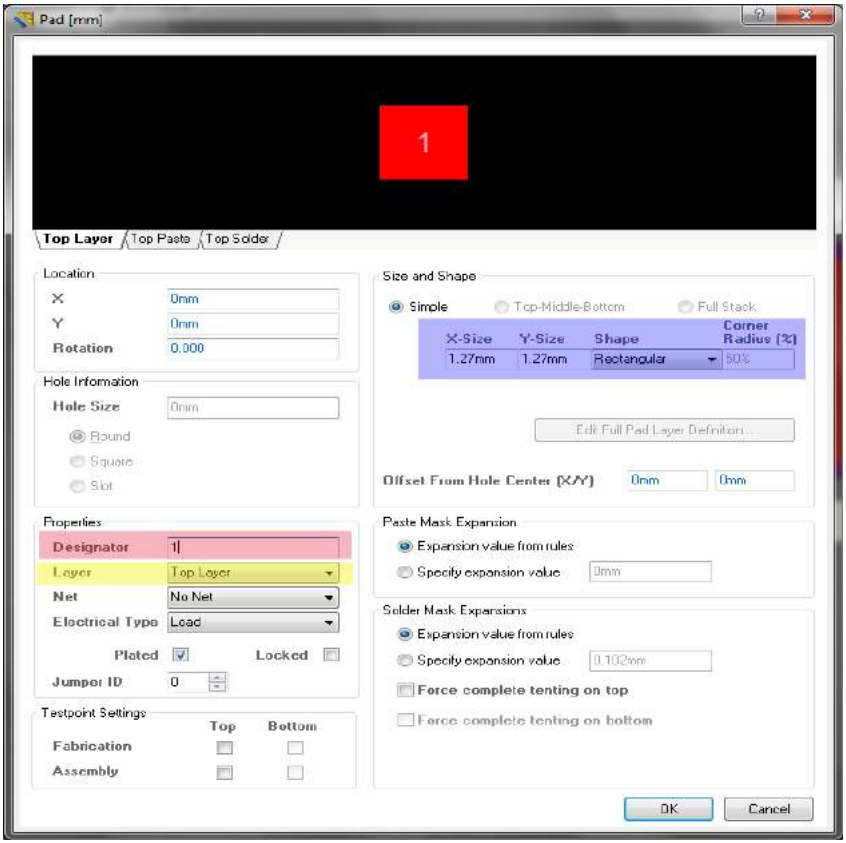


Рисунок 2.10 – Параметри контактної площинки резистора

Малюємо графіку компонента, спочатку обираємо потрібний шар – Top Overlay (для швидкого переходу між шарами можна використати комбінацію клавіш Ctrl + Shift + Scroll). Далі обираємо команду Place > Line та малюємо прямокутник навколо виводів компонента. Отримаємо результат, як на рис. 2.11 та зберігаємо все.

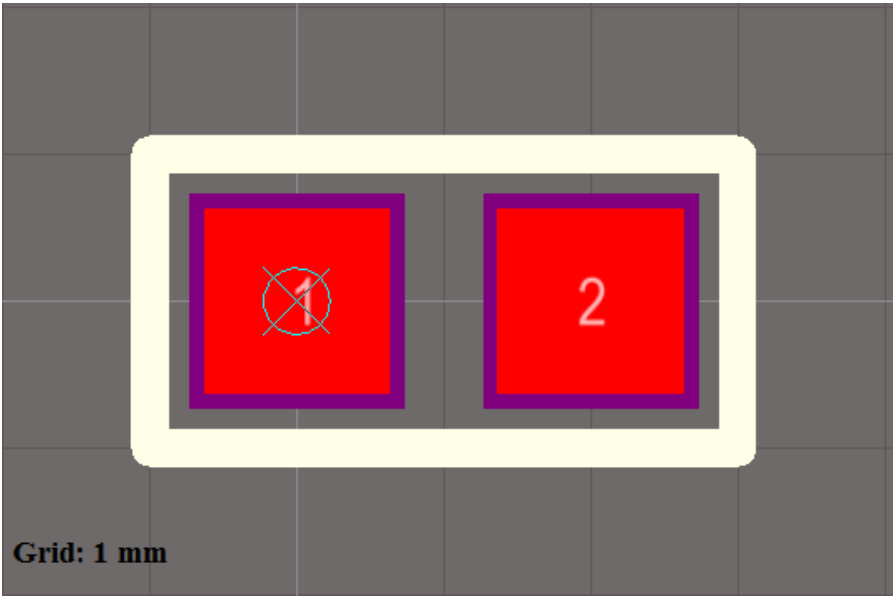


Рисунок 2.11 – ТПМ резистора

Тепер, потрібно з’єднати УГП та ТПМ резистора. Для цього, переходимо у вкладку з відкритим файлом бібліотеки символів з УГП резистора, та у вікні підключення моделей натикаємо на клавішу Add footprint. Далі, як показано на рис. 2.12 обираємо потрібний файл з ТПМ резистора та зберігаємо отриманий результат.

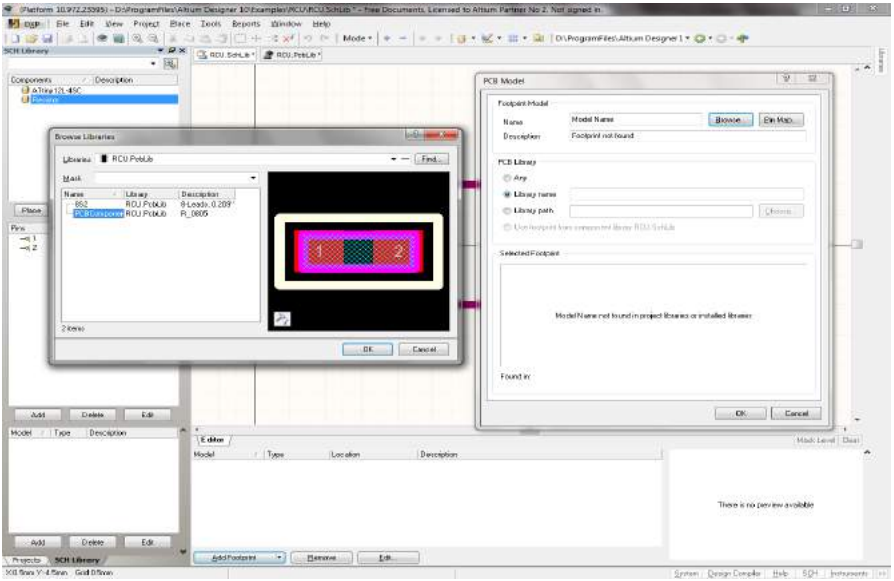


Рисунок 2.12 – З’єднання УГП та ТПМ резистора

2.2.3 Створення бібліотечного символу для мікросхеми ATtiny12

Відкривши бібліотеку символів, додаємо до неї новий компонент із назвою ATtiny12L-4SC та одразу встановлюємо позиційне позначення, для цього натискаємо на клавішу Edit у вікні Components і у поле Default Designator записуємо – D? та натискаємо ОК.

Тепер потрібно розмістити виводи та намалювати графіку компонента. Спочатку встановимо крок сітки 2.5 мм, потім натискаємо клавішу швидкого доступу P > Pin розміщуємо перший вивід у початок координат, і розміщуємо інші та малюємо графіку використавши клавішу швидкого доступу P > Rectangle відповідно до рис. 2.13.

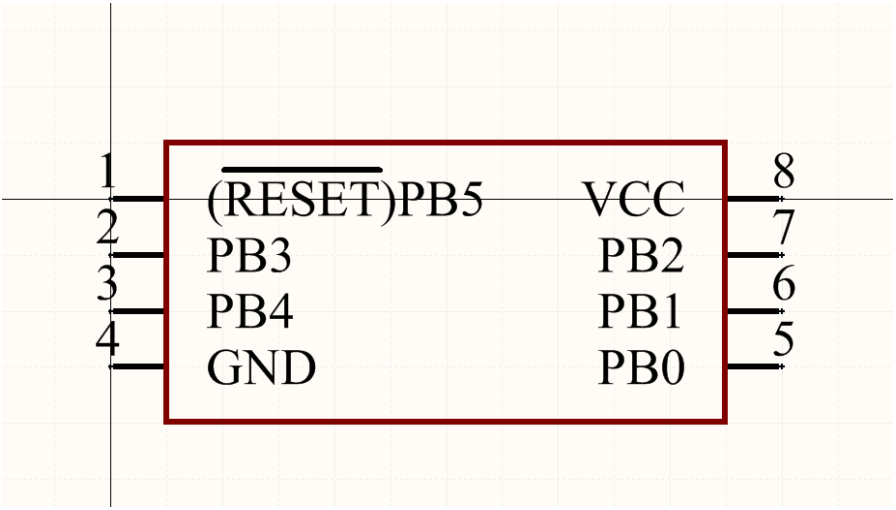


Рисунок 2.13 – УГП мікросхеми

Для позначення імен виводів потрібно викликати панель SCHLIB List у правому нижньому куті екрану. Вікно, що з'явилося, потрібно налаштувати на редагування відповідно до рис. 2.14 та задати імена виводів. Для інверсії символу є спеціальний символ - \. Зберігаємо цю бібліотеку.

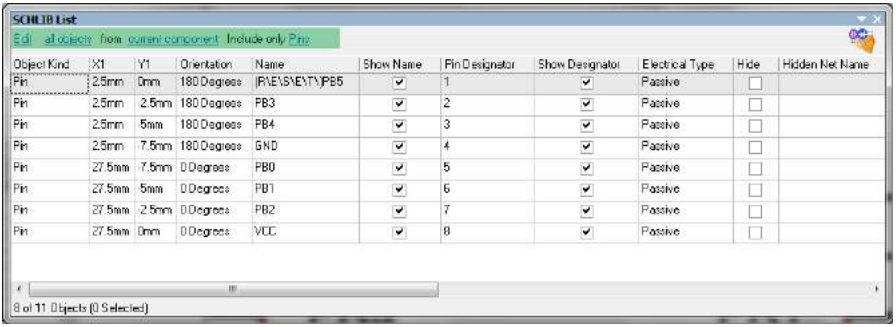


Рисунок 2.14 – Позначення імен виводів

2.2.4 Створення топологічного посадкового місця для мікросхеми ATtiny12

Для створення ТПМ мікросхеми ми будемо використовувати майстра для побудови корпусу елемента. Спочатку відкриємо бібліотеку посадкових місць, потім виберемо Tools > IPC Compliant Footprint Wizard. На першому кроці обираємо тип корпусу, у нашому випадку – SOP. На другому кроці вказуємо всі параметри корпусу, для нашої мікросхеми вони відображені на рис. 2.15.

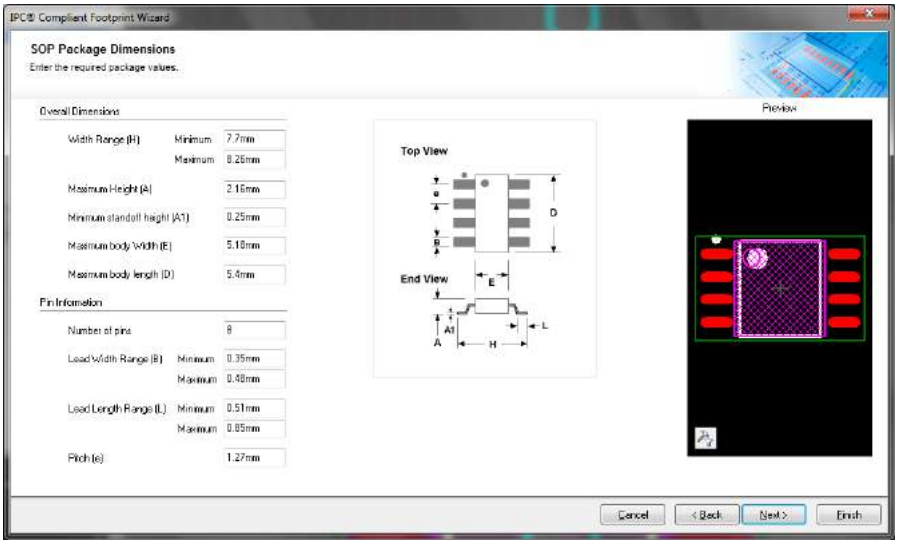


Рисунок 2.15 – Параметри корпусу мікросхеми

Далі, погоджуємося зі стандартними налаштуваннями, зупинившись лише на пункті SOP Courtyard, та задавши параметри з рис. 2.16.

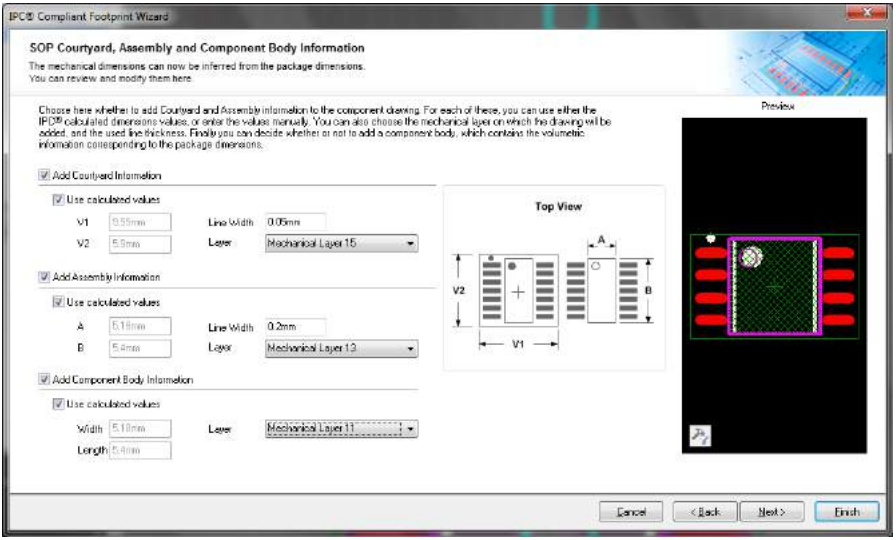


Рисунок 2.16 – Шари корпусу

У пункті Footprint Description задаємо ім’я корпусу – 8S2, а його опис – 8-Leads, 0.209" Body, Plastic Small Outline Package (EIAJ). На наступному кроці зберігаємо файл у потрібну папку (погоджуємося зберегти у поточну папку).

Після створення корпусу потрібно вивести на корпусі позиційне позначення. Для цього переходимо у шар Mechanical 1 і викликаємо команду Place > String та, після натискання клавіші Tab, задаємо параметри з рис. 2.17 та розміщуємо її. Зберігаємо все та підключаємо ТПМ до УГП мікросхеми так, як і для резистора.

Рисунок 2.17 – Параметри позиційного позначення

3 СТВОРЕННЯ ЕЛЕКТРИЧНОЇ СХЕМИ

Розглянемо створення файлу електричної схеми у складі проекту. Для цього виконаємо команду File > New > Project > PCB Project, після чого відразу його збережемо File > Save Project. Також важливо пам'ятати, що файли проекту можуть розміщуватися в пам'яті в різних місцях незважаючи на те, що в Altium (див. структуру проекту у вкладці Projects) вони будуть відображатися так, ніби вони знаходяться в рамках одного проекту. Тому, необхідно намагатися зберігати всі файли проекту в одній директорії фізичного диску.

Далі, додамо файл електричної схеми в проект командою File > New > Schematic або з контекстного меню проекту у вкладці Projects (рис.3.1), викликаного натисканням правої клавіші миші на проекті і збережемо його в папці проекту.

Тепер, необхідно підключити необхідний шаблон для аркуша схеми, викликавши команду Design > General Templates > Choose Another File і вибравши файл A4_1_portrait_ru.SchDot в папці Templates. У діалоговому вікні, що відкриється, запитується:

а) До яких документів застосувати цей шаблон:

- Just This Document – тільки до поточного документу;
- All schematic documents in the current project – до всіх схем поточного проекту;
- All open schematic documents – до всіх відкритих схем.

б) Що робити із його параметрами (спеціальними полями, що володіють властивостями автозаповнення):

- Do not update any parameters – не оновлювати параметри;
- Add new parameters that exist in the template only – додати нові параметри, не змінюючи значення і положення існуючих;
- Replace all matching parameters – оновити всі параметри.

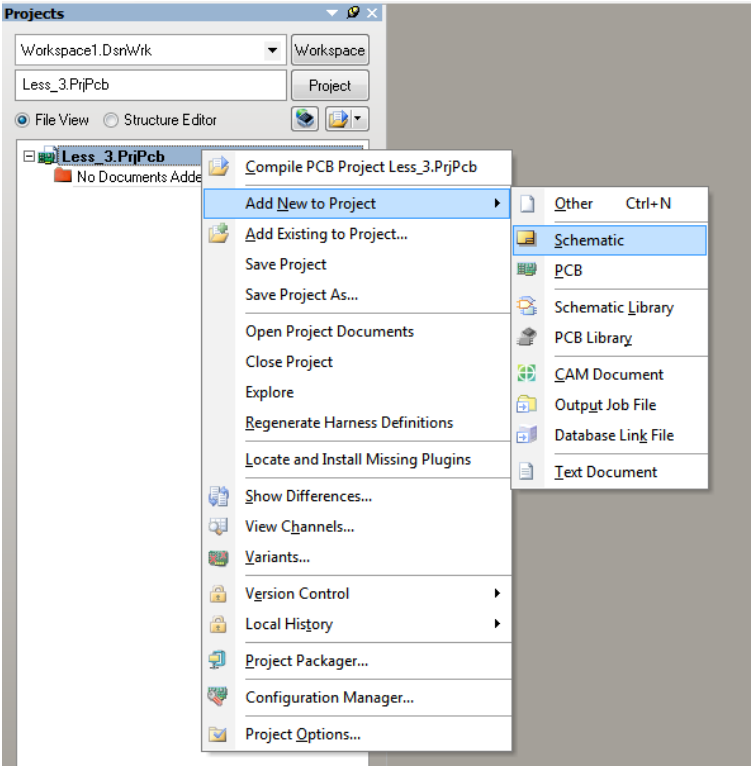


Рисунок 3.1 – Додавання файлу схеми у проект

Значення в цьому вікні необхідно встановити відповідно до рис. 3.2. Щоб створити схему необхідно використовувати бібліотеки компонентів. У Altium можна використовувати зовнішні і внутрішні бібліотеки.

Підключивши шаблон, заповнимо необхідні поля основного надпису. Для цього відкриємо вікно Design > Document Options і у вкладці Parameters (рис.3.3) для параметра Author, як значення вкажемо своє прізвище. Також вкажемо одиниці виміру – міліметри у вкладці Units.

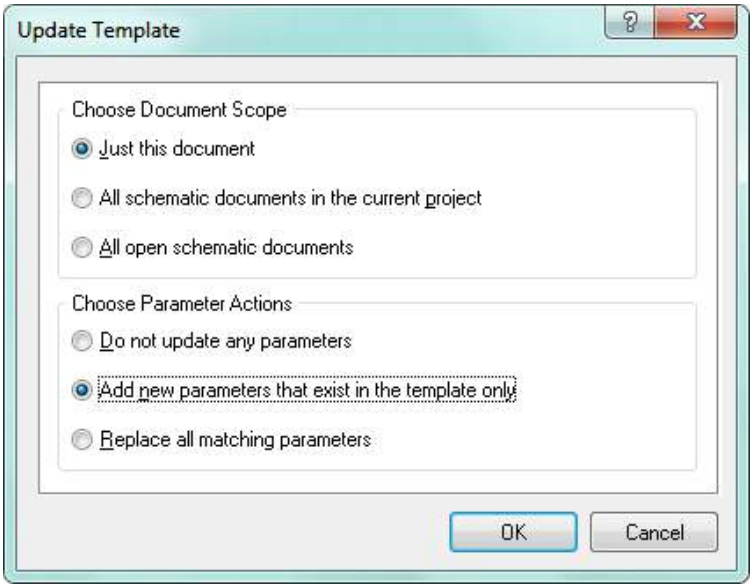


Рисунок 3.2 – Підключення шаблону

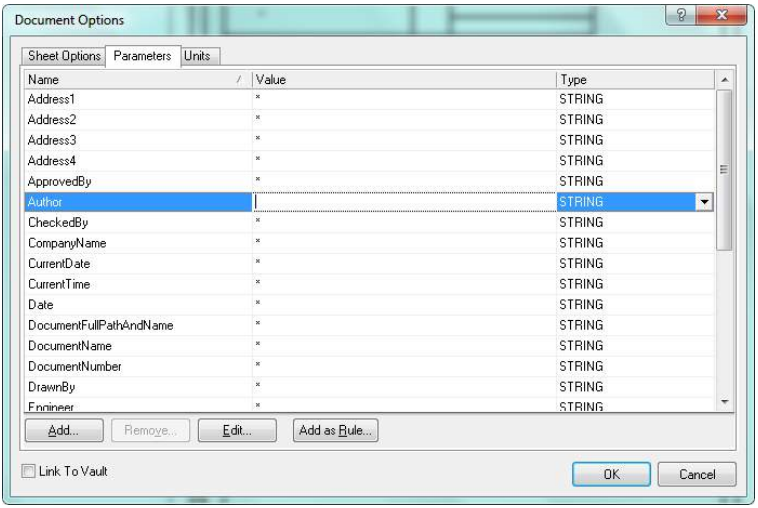


Рисунок 3.3 – Параметри схеми

Відкриємо панель Libraries (рис.3.4), де натисканням клавіші Libraries відкриємо вікно підключення бібліотек, що має 3 вкладки: Project - бібліотеки проекту (внутрішні бібліотеки); Installed - встановлені бібліотеки (зовнішні бібліотеки); Search Path - шлях для пошуку по бібліотеках (рис.3.5). Для зручності видалимо всі підключені бібліотеки у вкладці Installed за допомогою клавіші Remove, після чого встановимо потрібну зовнішню бібліотеку, натиснувши клавішу Install і вибравши файл RCU_source.IntLib. Потім, додамо в проект бібліотеку УГП (.SchLib), створену в розділі 2 за допомогою команди Project > Add existing to Project.

Використовуючи ці бібліотеки, зберемо схему відповідно до схеми на рис. 3.6, попередньо встановивши крок сітки 2.5мм. За допомогою команди Place можна почати розміщення обраного компонента, а клавішею SPACE повернути його.

Щоб малювати з'єднання (створювати електричні кола) необхідно використовувати спеціальну команду Place > Wire. Перемикання між режимами малювання з'єднання здійснюється клавішами SHIFT + SPACE. Якщо в режимі малювання навести курсор на електричне закінчення, то діагональний хрестик стане червоним, це говорить про те, що працює прив'язка до даного закінчення. Символи землі розміщуються командою Place > Power Port. (поле Style, вікна властивостей символу, повинне мати значення Power Ground).

Зібравши схему, пронумеруємо позиційні позначення елементів схеми (рис.3.7), використовуючи команду автоматичної нумерації Tools > Annotate Schematics Quietly.

Останнім етапом створення схеми є її перевірка на помилки (рис.3.8), яка викликається з підменю Project > Compile PCB Project (або з контекстного меню проекту). При наявності помилок відкриється вікно з їх описом.

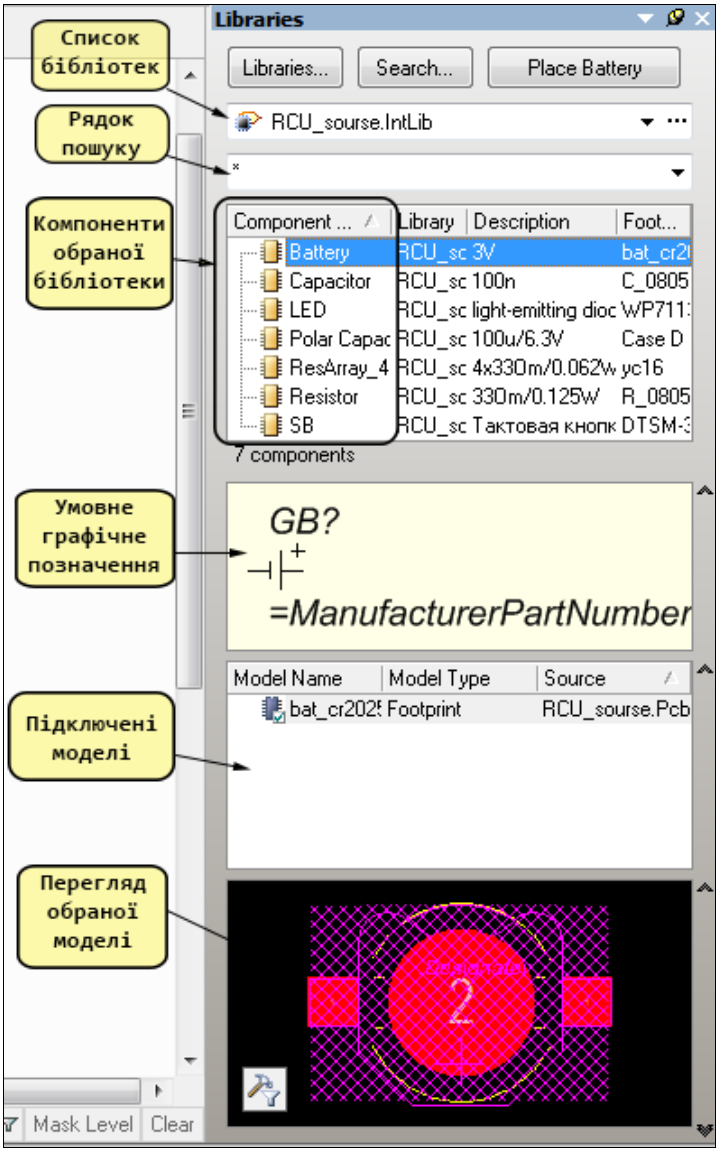


Рисунок 3.4 – Панель Libraries

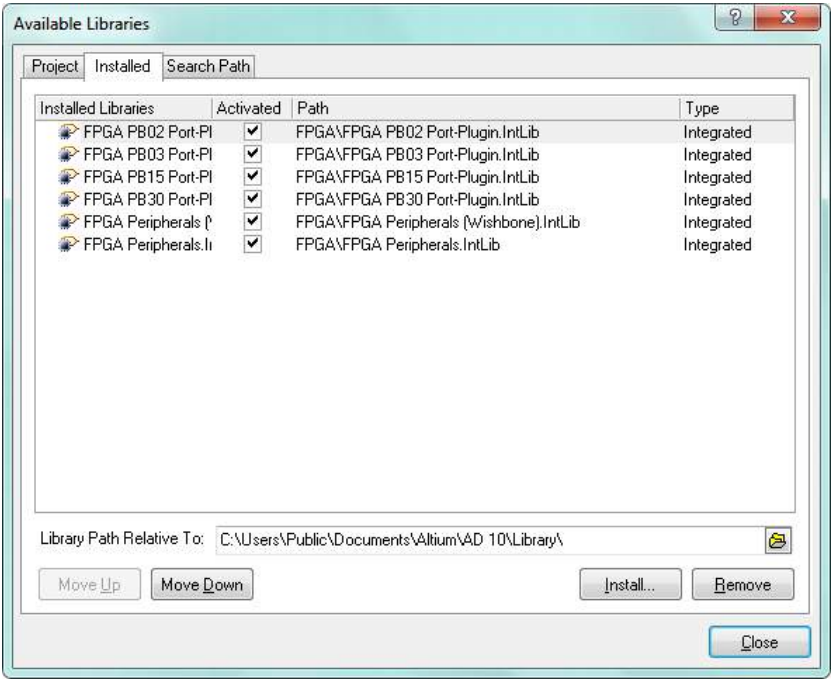


Рисунок 3.5 – Підключення бібліотек

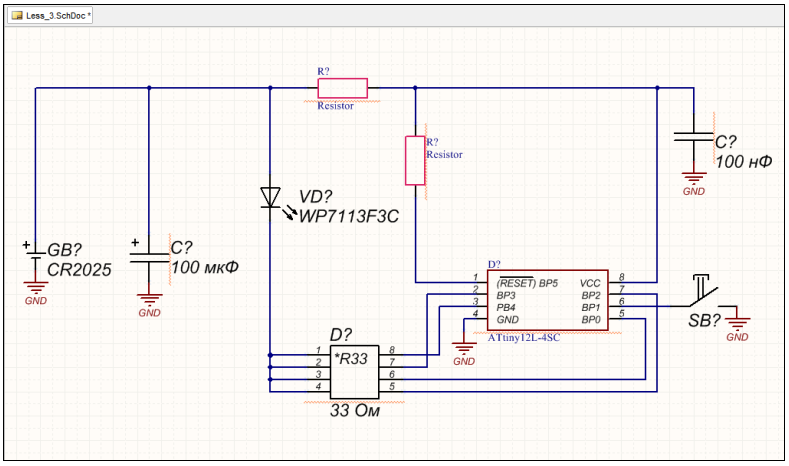


Рисунок 3.6 – Електрична схема

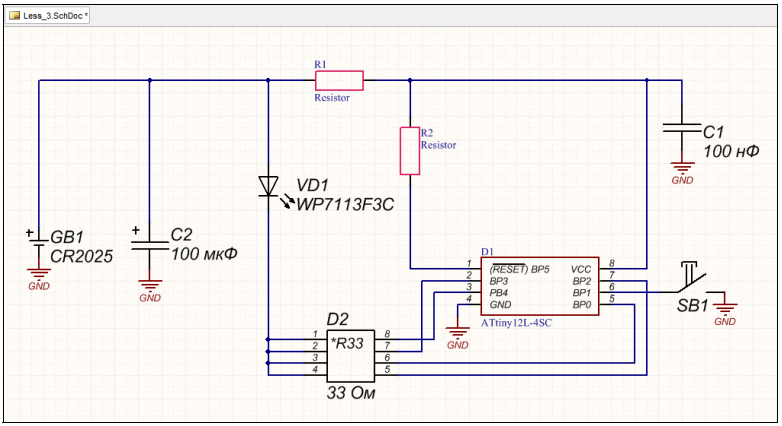


Рисунок 3.7 – Схема з позиційними позначеннями

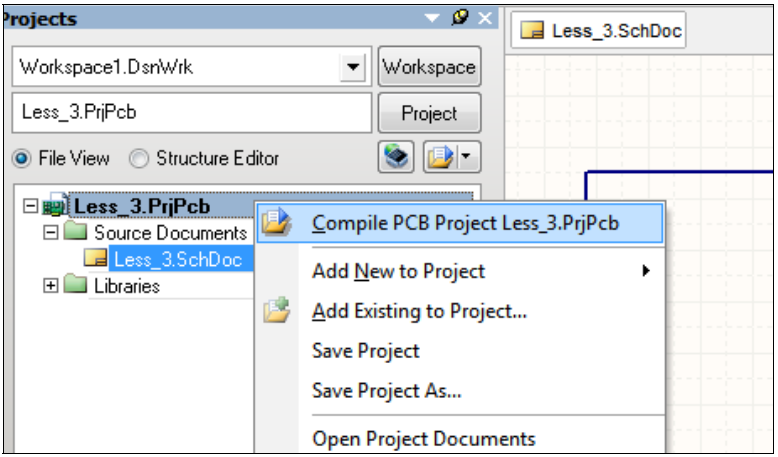


Рисунок 3.8 – Компіляція проекту

4 СТВОРЕННЯ ДРУКОВАНОЇ ПЛАТИ

4.1 Створення файлу друкованої плати

Створимо файл друкованої плати із вкладки Project > Add New to Project > PCB (або із контекстного меню) та збережемо його командою File > Save (або комбінацією клавіш Ctrl+S). Після чого, встановимо одиниці вимірювання міліметри у вікні Design > Board Options, у списку Range поля Snap Options встановимо значення 1x SnapGrid, а також крок сітки у 1мм за допомогою клавіші G.

Порядок роботи із друкованою платою:

а) розробка конструкції плати:
- створення контуру;
- створення структури (Опис стеку шарів);
- встановлення місць для кріплення;
- зазначення заборонених місць для трасування.

б) перенесення даних із схеми на плату;

в) створення правил проектування;

г) розміщення компонентів;

д) трасування (створення малюнку провідників, створення провідників між компонентами).

Із платою можна працювати у двох режимах 2D або 3D, переключення між якими відбувається за допомогою швидких клавіш 2 та 3 або командами View > Switch to 2D/3D. У режимі 3D плату можна обертати, утримуючи клавішу SHIFT та рухаючи натиснуту праву клавішу миші або, утримуючи клавішу SHIFT, використовувати кнопки управління поворотом, які розташовані на «сфері зі стрілками» (рис.4.1).

Повернути 3D модель у початкове положення можна клавішею 0 (нуль) або командою View > Zero Rotation. За допомогою комбінації клавіш V+B можна перевертати плату на 180 градусів (або View > Flip Board). Також є можливість вибрати режими відображення (рис. 4.2).

Тепер потрібно створити контур плати, в даному розділі ми будемо імпортувати вже створену реалістичну модель плати у форматі STEP. Відкриємо вікно 3D Body командою Place > 3D Body, за

допомогою якого можна створювати або додавати 3D моделі. У вікні, що з'явилося, виберемо тип моделі Generic STEP Model (рис.4.3), додамо клавішею Add у полі Snap Points точку із координатами (0;0;0) і натиснемо кнопку Link to Step Model.

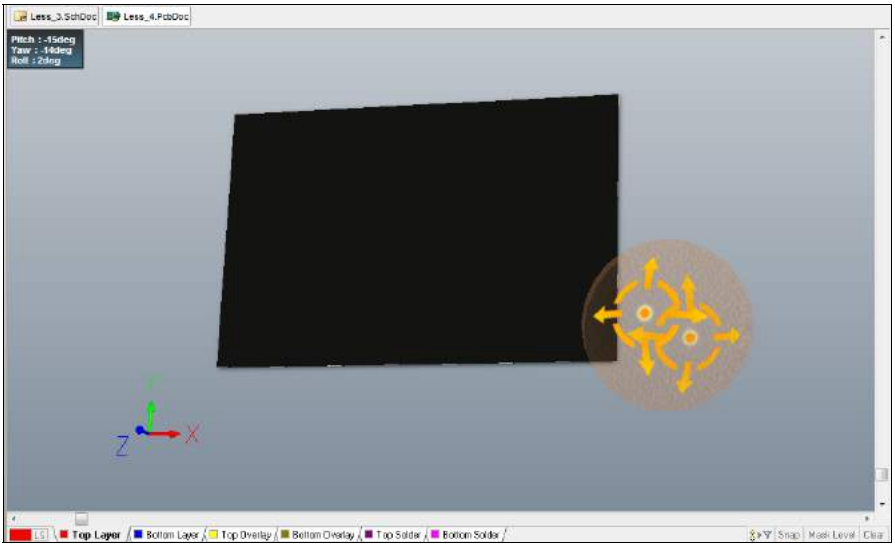


Рисунок 4.1 – Робота у режимі 3D

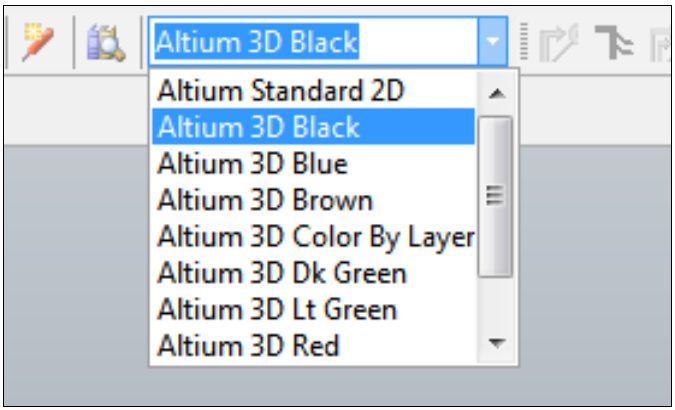


Рисунок 4.2 – Вибір режиму відображення

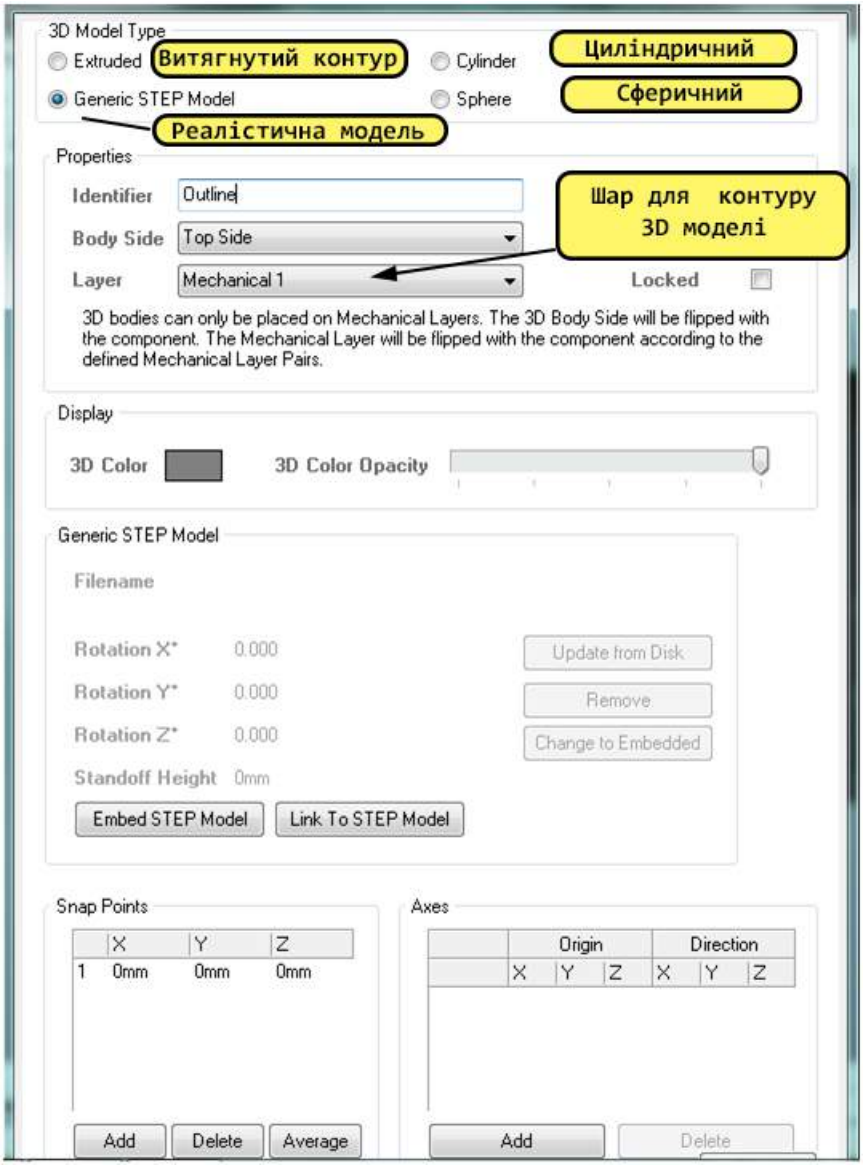


Рисунок 4.3 – Вікно додавання 3D моделі

Клавішею Add Directories викличемо вікно налаштувань, у якому вкажемо розташування папки із STEP моделями (рис.4.4). У списку виберемо модель Outline.stp і натиснемо OK. Далі, у діалоговому вікні 3D Body натиснемо OK і розмістимо модель у робочій області лівою клавішею миші, програма запропонує встановити ще одну модель у нашому випадку слід відмовитися клавішею Cancel. Слід зазначити, що при зміненні файлу моделі всі зміни автоматично переносяться у Altium Designer.

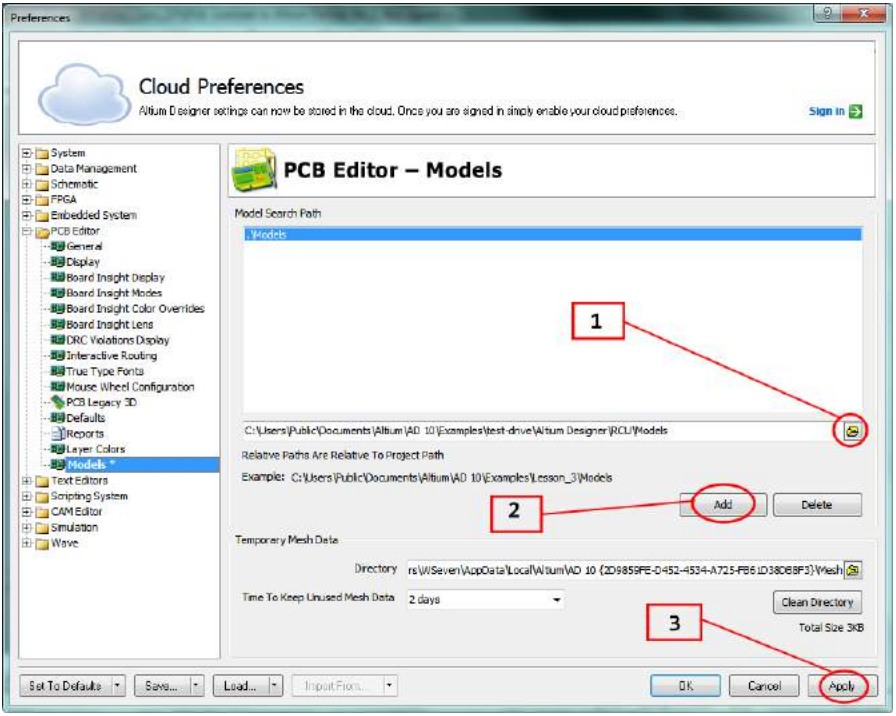


Рисунок 4.4 – Додавання папки із моделями

Для формування контуру (границь) плати використовується група команд меню Design > Board Shape серед яких найбільш потрібними є:

- Redefine Board Shape – малювання контуру плати вручну;

- Define from Selected Objects – формування контуру плати із виділених об’єктів;
- Define from 3D Body – формування контуру із тривимірної моделі (працює лише у тривимірному режимі);
- Define Board Cutout – формування вирізу на платі.

Командою Define from 3D Body (рис.4.5) вкажемо програмі, що контуром плати є наша модель: спершу потрібно вказати модель, а потім поверхню моделі. Після цього, з’явиться вікно, яке потрібно закрити. Область всередині контуру стане чорного кольору, а зовні сірою, що свідчить про коректне створення плати.

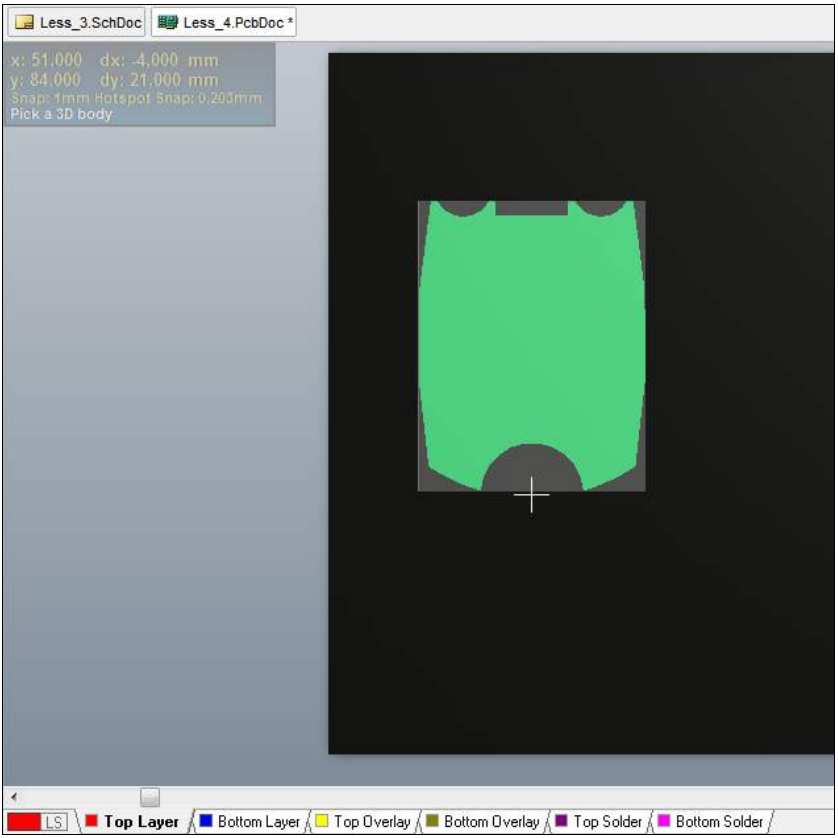


Рисунок 4.5 – Робота із командою Define from 3D Body

Створений контур плати служить лише обмеженням для програми Altium та вказує простір (границі, в яких можна розмістити) для розміщення компонентів і трасування провідників, але не є фізичним контуром плати, за яким виконується його обрізка. Тобто, контур цієї плати є віртуальним і не відображається на жоден з шарів.

Для створення фізичного контуру плати потрібно виконати одну з цих команд в підменю Design > Board Shape:

- Define from selected objects – побудувати із вибраних об’єктів. В цьому випадку у вибраному «механічному» (див. далі) шарі повинен бути попередньо створений замкнутий контур із ліній та/або дуг, який і стане фізичним контуром плати;
- Create Primitives From Board Shape – створити примітиви із контуру плати. В цьому випадку фізичний контур буде співпадати із віртуальним.

Перенесемо контури створеної нами плати на потрібний шар за допомогою команди Design > Board Shape > Create Primitives From Board Shape попередньо перейшовши у двовимірний режим (рис. 4.6). У діалоговому вікні треба вказати шар, до якого буде перенесений контур, товщину лінії контуру та додаткові параметри встановити згідно з рисунком 4.6.

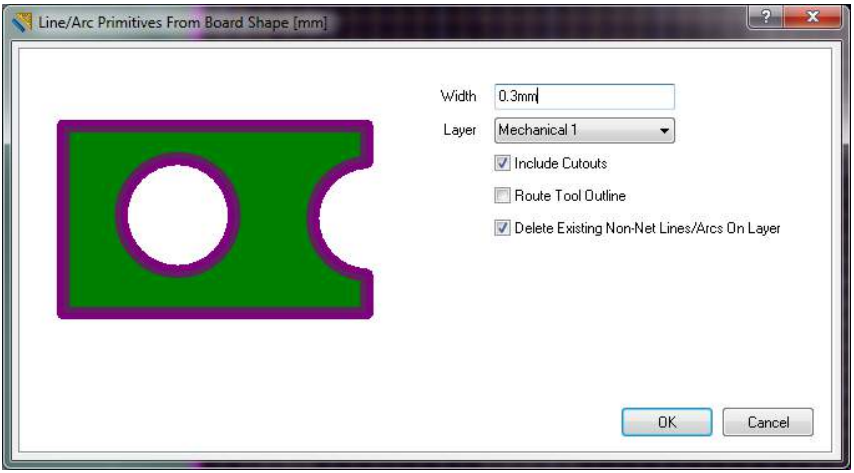


Рисунок 4.6 – Створення фізичного контуру

Далі перенесемо початок координат у точку (0;0;0) нашої моделі (рис.4.7), для цього скористаємося командою View > Origin > Set. Результат можно перевірити, перейшовши у режим 3D та порівнявши із рис. 4.8.

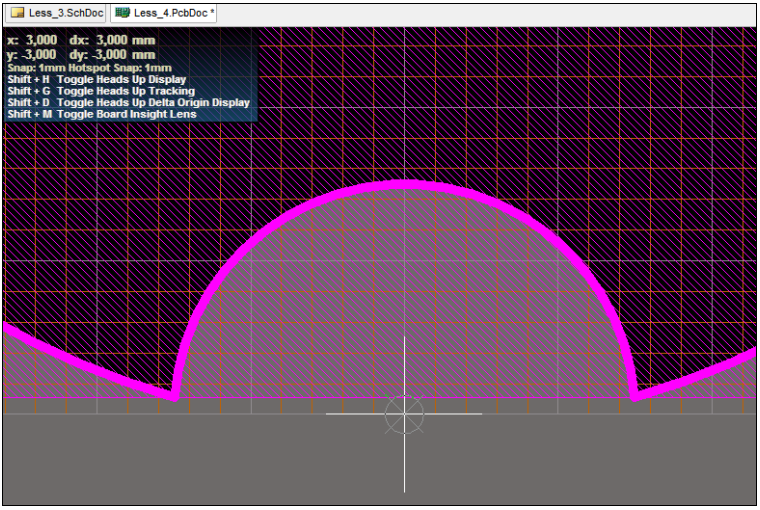


Рисунок 4.7 – Вибір нової точки координат

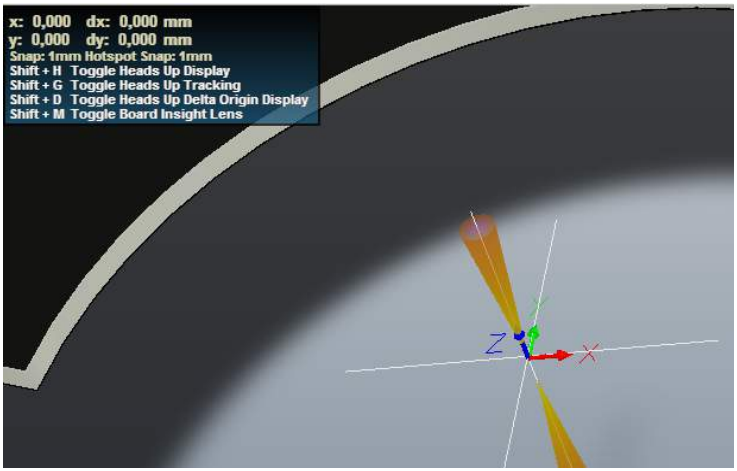


Рисунок 4.8 – Перегляд нової точки координат

Наступним кроком є створення структури плати. Додавання нових шарів і керування їх положенням в стеку шарів друкованої плати проводиться у вікні, яке викликається командою Design > Layer Stack Manager (рис.4.9).

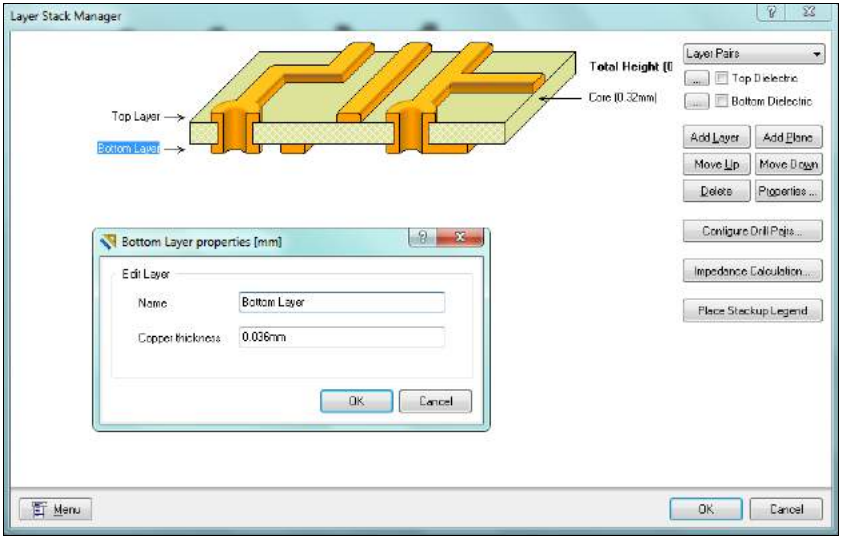


Рисунок 4.9 – Вікно керування стеком шарів

В правій частині вікна є набір команд для керування порядком розташування шарів. Для нашого прикладу структура плати не має значення, тому нічого змінювати не потрібно.

Перегляд всіх існуючих шарів проекту і керування їх відображенням виконується у вікні View Configurations, яке викликається командою Design > Board Layer&Colors або натисненням клавіші L. В даному вікні (рис.4.10) не можна додавати або видаляти шари, тут виконується лише керування видимістю шарів. Для редагування назви шару потрібно натиснути клавішу F2.

Шари розділені на групи:

- Signal Layers – призначені для створення малюнку провідників друкованої плати. Всього проект може містити до 32-х сигнальних шарів;

• Internal Layers – призначені для виконання провідників у вигляді металізованих полігонів (землі та живлення). Відображення інформації на цих шарах інверсне. Всього на платі може бути задіяно до 16 шарів живлення і землі. (Ці шари не призначені для прокладання провідників);

• Mechanical Layers – призначені для розміщення в них елементів збирання, контуру друкованої плати та інше (всього 16 шарів). Для використання шару спершу його потрібно активувати: поле Enable;

• Mask Layers – шари паяльної пасти і захисної маски (наприклад, Top Paste і Bottom Paste та Top Solder і Bottom Solder);

• Other Layers – додаткові шари, до яких відносяться шари із забороненими зонами та шари, що відображають отвори на платі;

• Silkscreen Layers – призначені для розміщення інформації про маркування та позначення контурів компонентів на платі (наприклад, Top Overlay; Bottom Overlay).

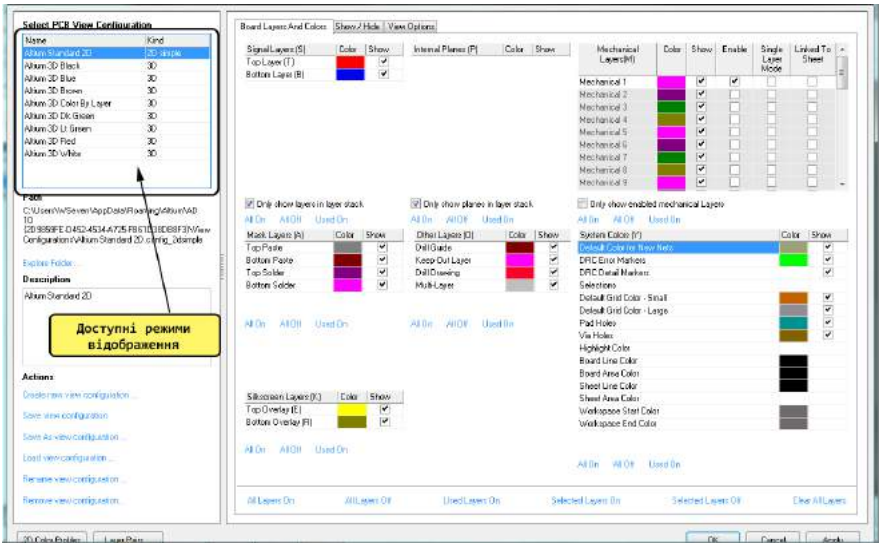


Рисунок 4.10 – Керування відображенням шарів плати

Щоб передати інформацію із схеми на плату, відкриємо потрібний аркуш електричної схеми і, знаходячись у редакторі цієї схеми, виконаємо команду Design > Update PCB Document … Відкриється вікно із списком відмінностей між схемою та платою (рис. 4.11). Попередньо вкажемо ім'я кола та лінію зв'язку, що входить до нього, командою Place > Net Label в редакторі схем згідно з рис. 4.12.

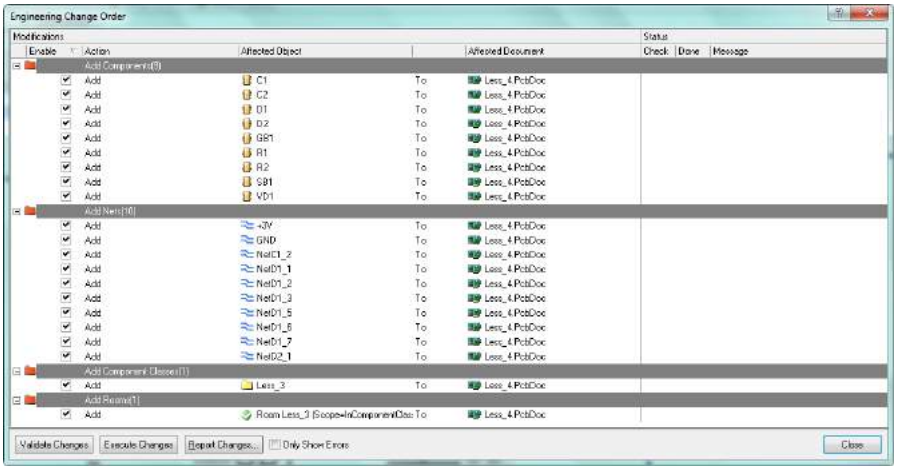


Рисунок 4.11 – Вікно із списком модифікацій плати

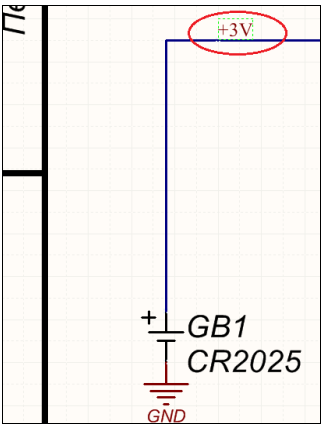


Рисунок 4.12 – Додавання імені кола

Перелік об’єктів, наявність яких звіряється в схемі та платі, задається у вікні Project > Project Options на вкладці Comparator. Спочатку командою Validate перевіряється можливість внесення змін (тобто пошук помилок), де головною проблемою може бути відсутність посадкових місць у того чи іншого компонента (як і у нашому випадку на рис. 4.13). При наявності помилок в колонці Status

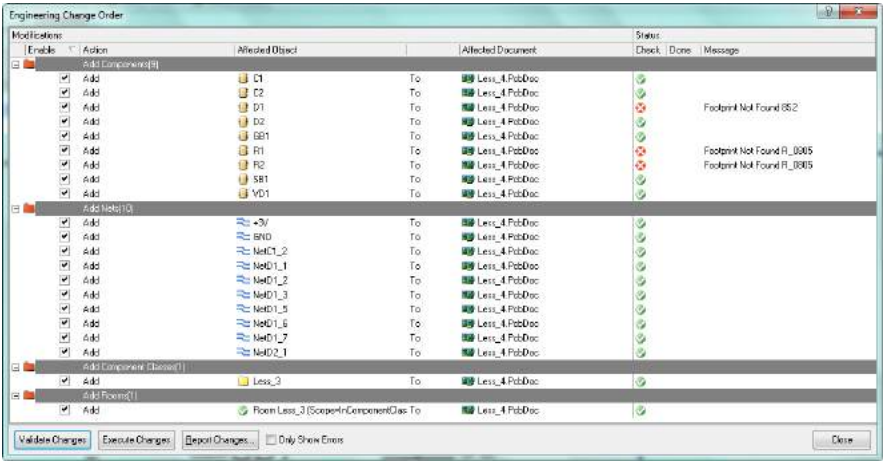


Рисунок 4.13 – Перевірка можливості внесення змін

вікна Engineering Change Order слід натиснути клавішу Close і виправити їх. Команда Execute завантажує дані із схеми до плати.

В нашому випадку при перевірці виявиться, що в проекті не вистачає бібліотеки посадкових місць для резистора та мікросхеми. Виправимо цю помилку, додавши файл RCU.PcbLib, створений нами у попередніх розділах командою Project > Add Existing to Project і спробуємо знову. Після виконання команди Execute відкриється редактор плат із перенесеними туди даними. Виділений на рис. 4.14 прямокутник - це кімната-регіон плати, за яким закріплений певний клас (група) компонентів. Клас компонентів додається згідно зі схемою. Кімната нам не знадобиться, тому видалимо її клавішею Del, не видаляючи при цьому компоненти.

Створення правил проектування або, інакше кажучи, конструктивних і технологічних обмежень проектування плати виконується у вікні Design > Rules за допомогою команди New Rule із контекстного меню обраного параметру (рис. 4.15).

Задамо правило для провідника (трасування), згідно з яким у кола із назвою +3V його ширина має бути 1мм. Для цього вкажемо параметри згідно з рис. 4.16. Створене нами правило має мати більш високий пріоритет ніж попереднє. Якщо це необхідно, скористуємось клавішею Priorities.

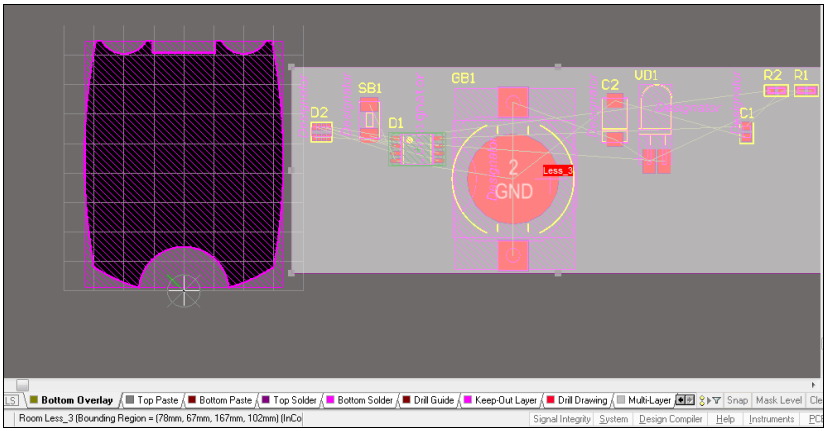


Рисунок 4.14 – Видалення кімнати

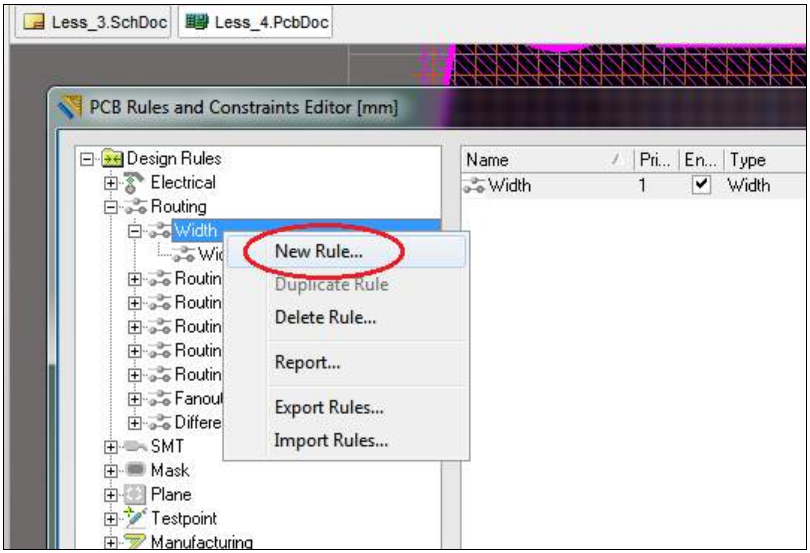


Рисунок 4.15 – Контекстне меню правила проектування

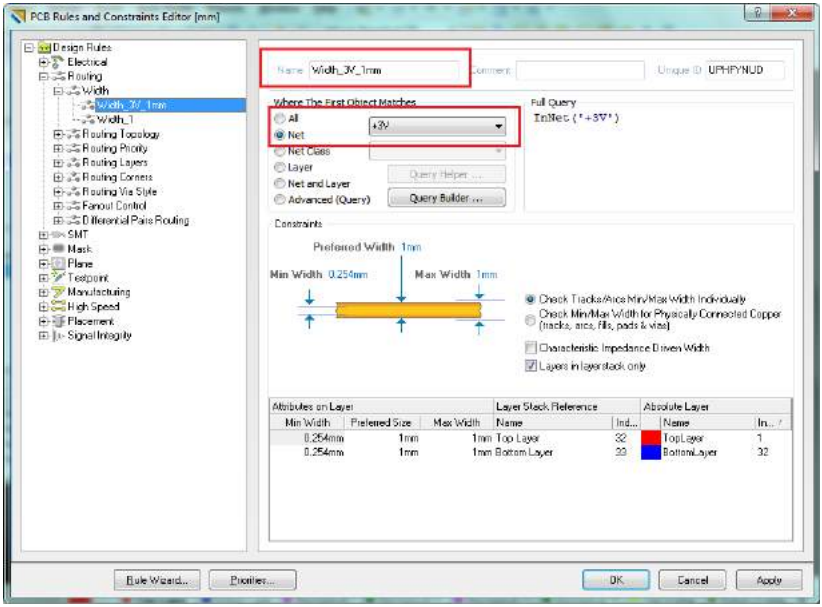


Рисунок 4.16 – Створення нового правила

Далі, потрібно розмістити компоненти, але перед цим вимкнемо функцію автопанарамування (рис.4.17), яка при переміщенні курсору в режимі редагування компонента (при переміщенні компонента) за межі робочого вікна автоматично переміщує видиму робочу область. Для того, щоб вимкнути цю функцію треба відкрити вкладку General групи налаштувань PCB Editor у вікно налаштувань DXP > Preferences і в полі Autopan Options для параметра Style вказати значення Disable.

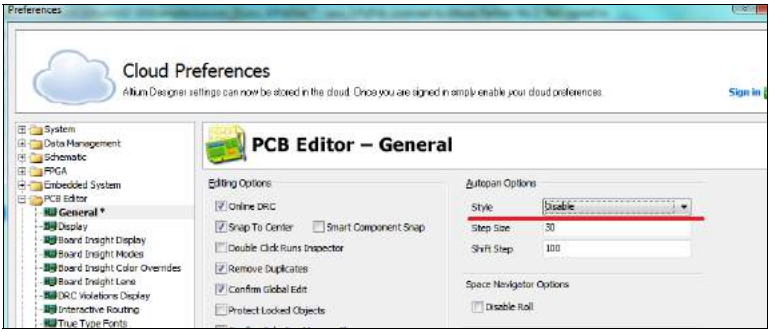


Рисунок 4.17 – Вимкнення автопанарамування

Розміщувати компоненти можна, як простим «перетягуванням» компонентів на потрібні місця так і задавши потрібні координати. Вибравши компонент лівою клавішею миші натиснемо клавішу J та виберемо пункт New Location у контекстному меню (рис.4.18).

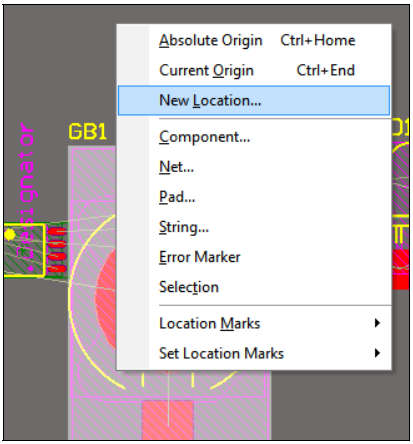


Рисунок 4.18 – Переміщення компонента

З’явиться вікно, у якому потрібно вказати нові координати для компонента. Так, перемістимо компонент GB1 в точку X - 0mm; Y – 19mm, SB1 X – 0mm; Y – 32.5mm, попередньо повернувши його клавішею SPACE та VD1 на позицію X – 0mm; Y – 37.5mm. Всі інші компоненти розмістимо перетягуванням згідно із рис. 4.19.

Для зручності скористаємось панеллю PCB (рис.4.20), в якій вкажемо режим роботи з компонентами та виберемо клас <Outside Board Components>, після цього у полі Components будуть відображатися лише ті компоненти, які знаходяться за межами плати.

Можливі наступні дії із обраними об’єктами:

- Mask (Dim) – варіант фільтрації: маска або затемнення;
- Select – виділення;
- Zoom – масштабування;
- Clear Existing – очистка попереднього виділення.

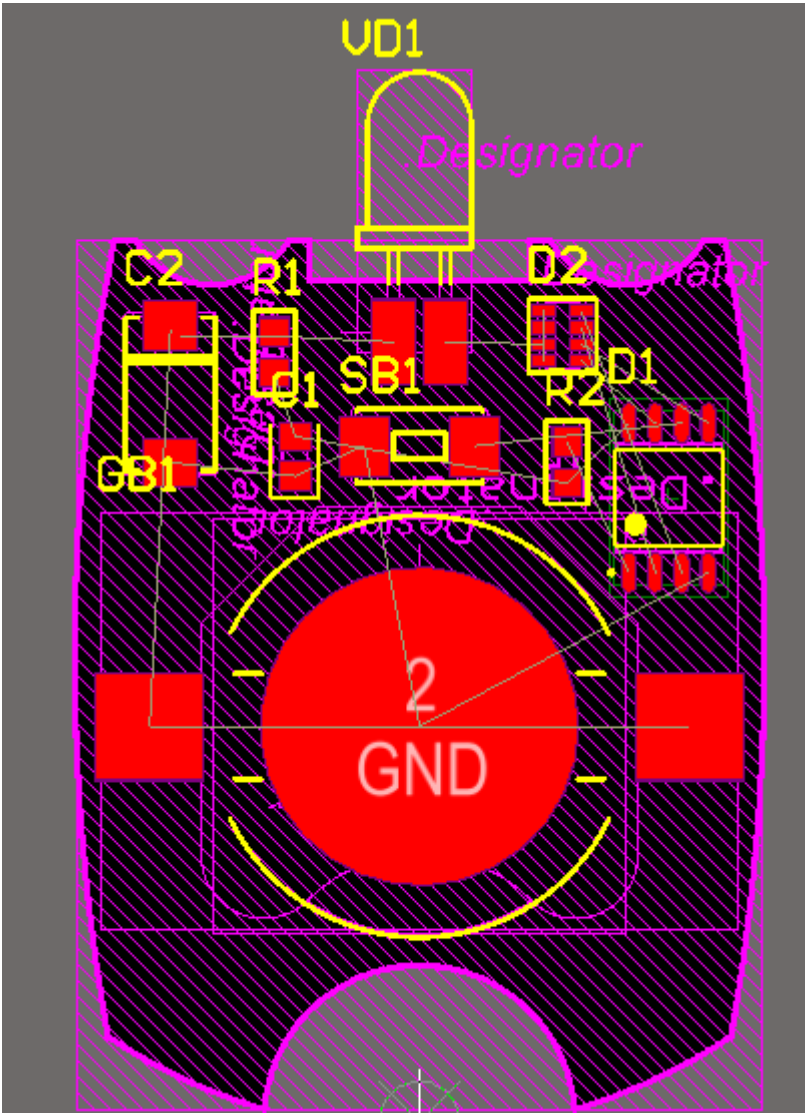


Рисунок 4.19 – Розміщення компонентів на платі

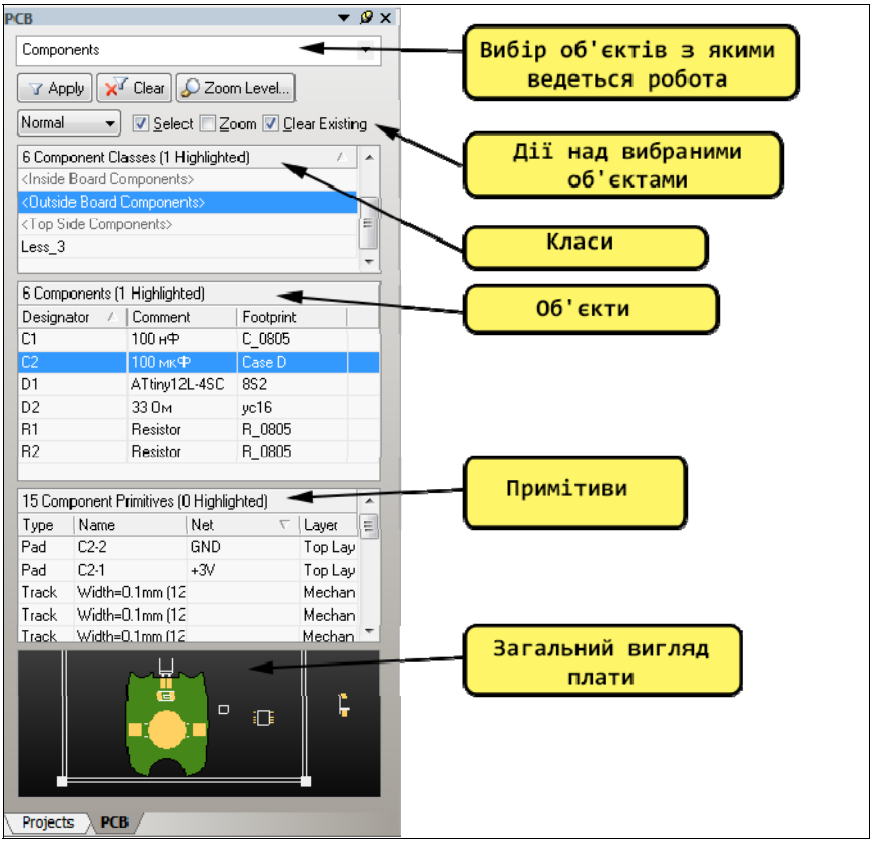


Рисунок 4.20 – Панель PCB

Останнім кроком є трасування. Встановимо крок сітки 0,1мм та перейдемо до шару Top Layer (Ctrl+Shift+ колесо миші). Аби малювати провідники між з'єднаними компонентами, скористаємось командою Place > Interactive Routing. При наведенні курсору на вивід компонента на місці курсору з'явиться коло - це говорить про те, що працює прив'язка до даного закінчення (рис. 4.21).

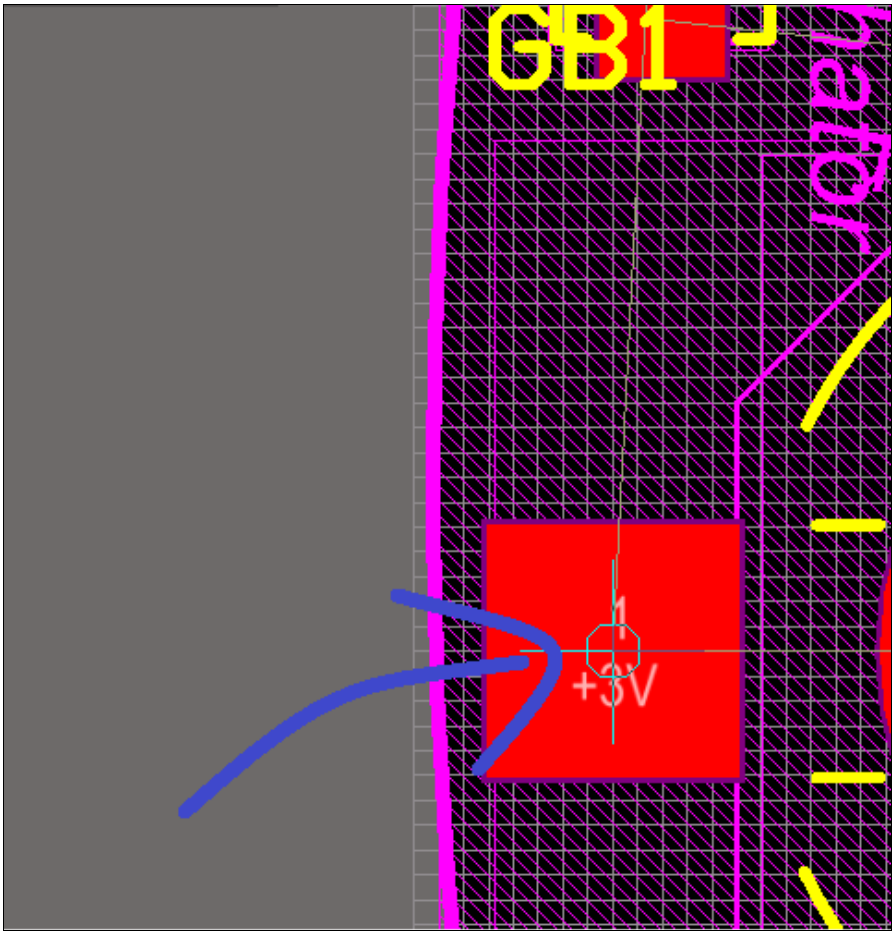


Рисунок 4.21 – Прив’язка до закінчення

Задамо налаштування для команди інтерактивного трасування, відкривши вікно налаштувань DXP > Preferences > PCB Editor > Interactive Routing і вказавши у розділі Interactive Routing Width/ Via Size Sources значення Rule Preferred. В цьому розділі вказується, які значення потрібно брати для ширини провідника та перехідного отвору. Ми вказуємо, що значення потрібно брати те, яке рекомендовано правилом проектування (рис. 4.22)..

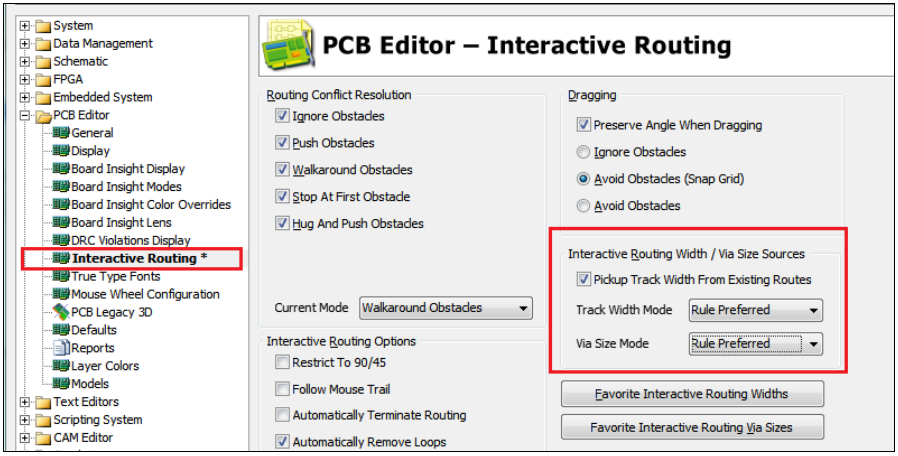


Рисунок 4.22 – Налаштування команди інтерактивного трасування

Для автоматичного завершення траси провідника потрібно натиснути Ctrl+ліва клавіша миші. Переключення між режимами трасування відбувається командою Shift+Space. Під час трасування бажано перейти до одношарового режиму відображення командою Shift+S, після чого на екрані буде відображено тільки активний шар, а інші, в залежності від налаштувань, будуть або напівпрозорими, або показані сірим кольором. Створимо структуру провідників, як на рис. 4.23.

Завершимо етап трасування створенням полігону (регіону плати, що покритий металевим покриттям), який буде з'єднувати всі виводи компонентів, що належать до кола GND. У вікні налаштувань, яке викличемо командою Place > Polygon Pour вкажемо значення, як на рисунку 4.24. Після чого розмітимо контурні лінії для нашого полігону. До створеного полігону (рис. 4.25) під'єднаємо закінчення, що залишилось та перезавантажимо його командою Polygon Actions > Repour, що знаходиться у контекстному меню, яке викликається натисканням правої клавіші миші на потрібному об'єкті (рис.4.26).

Якщо в панелі PCB обрати режим роботи із колами (Net), то у вікні Nets, для всіх кіл буде показано:

- Name – назва;
- Node – кількість вузлів;

- Routed – загальна довжина створеного провідника (траси);
- Un-Routed – довжина частини кола, що не має траси.

Так, дивлячись значення поля Un-Routed для всіх кіл, перевіримо чи є в них частини, для яких не створено траси (повинно бути значення 0).

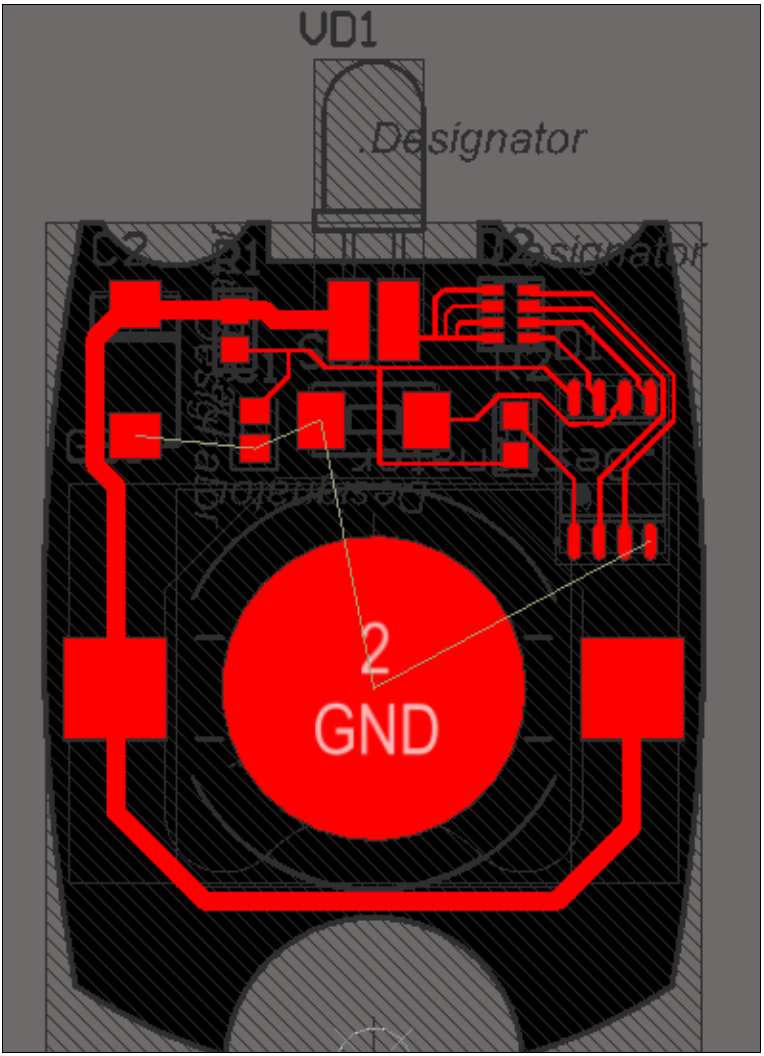


Рисунок 4.23 – Структура провідників плати

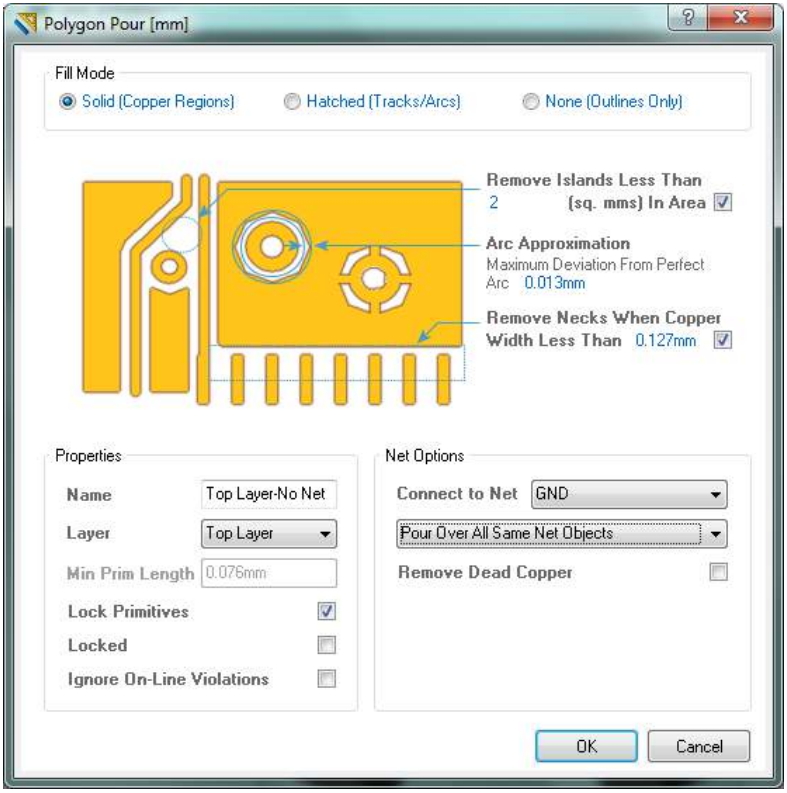


Рисунок 4.24 – Вікно налаштувань полігону

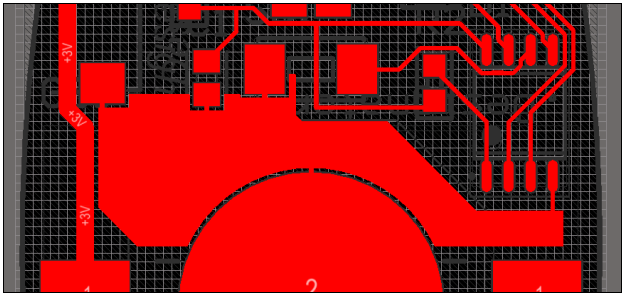


Рисунок 4.25 – Створений полігон

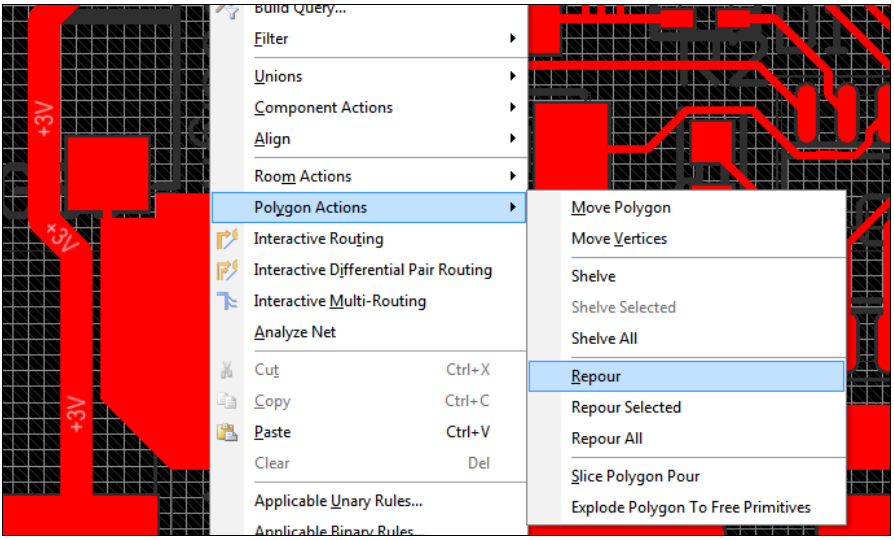


Рисунок 4.26 – Виклик команди Repour

4.2 Створення 3D моделей у бібліотеці посадкових місць

Спершу відкриємо створену раніше бібліотеку посадкових місць (яку ми додали до проекту) та виберемо компонент 8S2 в панелі PCB Library. Видалимо спрощену 3D модель, створену за допомогою майстра. Тепер вбудуємо (закріпимо) реалістичну модель до посадкового місця компонента бібліотеки, відкривши вікно роботи з 3D моделями командою Place > 3D Body. Натиснувши в ньому кнопку Embed STEP Model, виберемо модель із назвою ATTINY12L.step (рис.4.27). Далі натиснемо OK і розмістимо модель у будь-якому місці.

Після цього, перейдемо до режиму 3D та побачимо, що модель обернута відносно посадкового місця. Виправимо це, відкривши вікно властивостей подвійним натисканням лівої клавіші миші на моделі, та вкажемо значення повороту по вісі X - 90 градусів (рис.4.28). До речі, виконати масштабування на існуючих об'єктах можливо скориставшись командою View > Fit All Objects

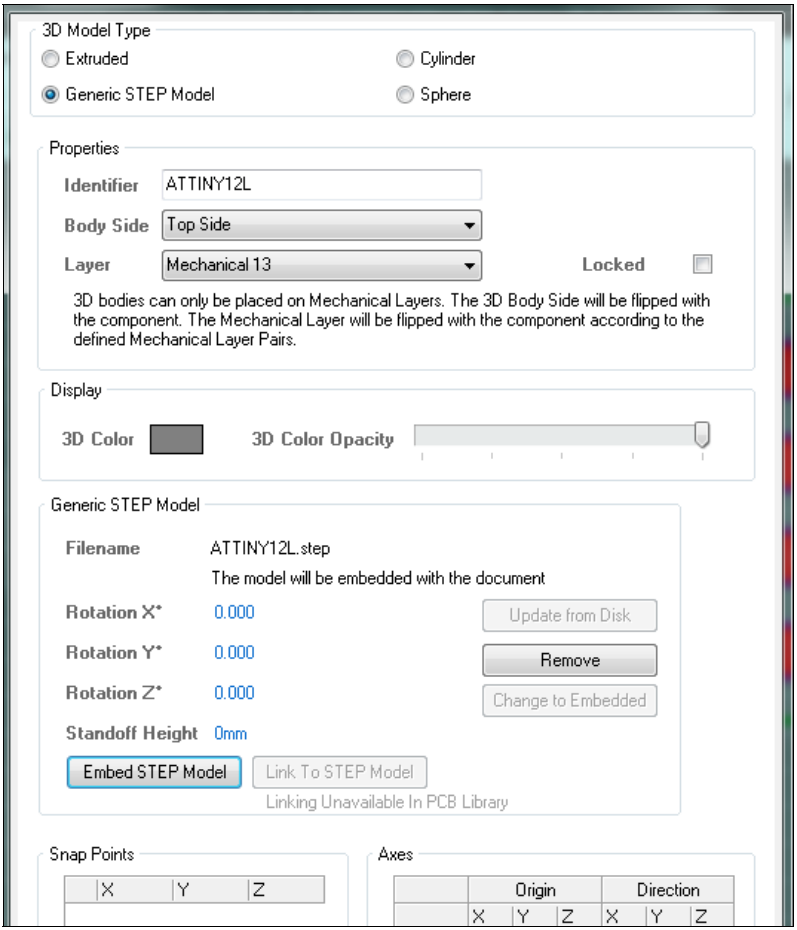


Рисунок 4.27 – Вибір 3D моделі, що буде вбудована

Поєднаємо центр моделі із центром посадкового місця командою Tools > 3D Body Placement > Position 3D Body. Після запуску команди курсор матиме вигляд маленького хрестика, ним необхідно вибрати модель. Тепер курсор прийме вигляд великого хреста (блакитного кольору), який рухається по невидимим вузлам моделі. Виберемо цим курсором центр нашої моделі, після чого

курсор знову змінить вигляд і тепер необхідно вказати на центр посадкового місця (рис.4.28).

У результаті модель буде розташовуватися нібито під посадковим місцем. Вкажемо висоту розташування моделі 2,3мм у вікні її властивостей (рис.4.29).

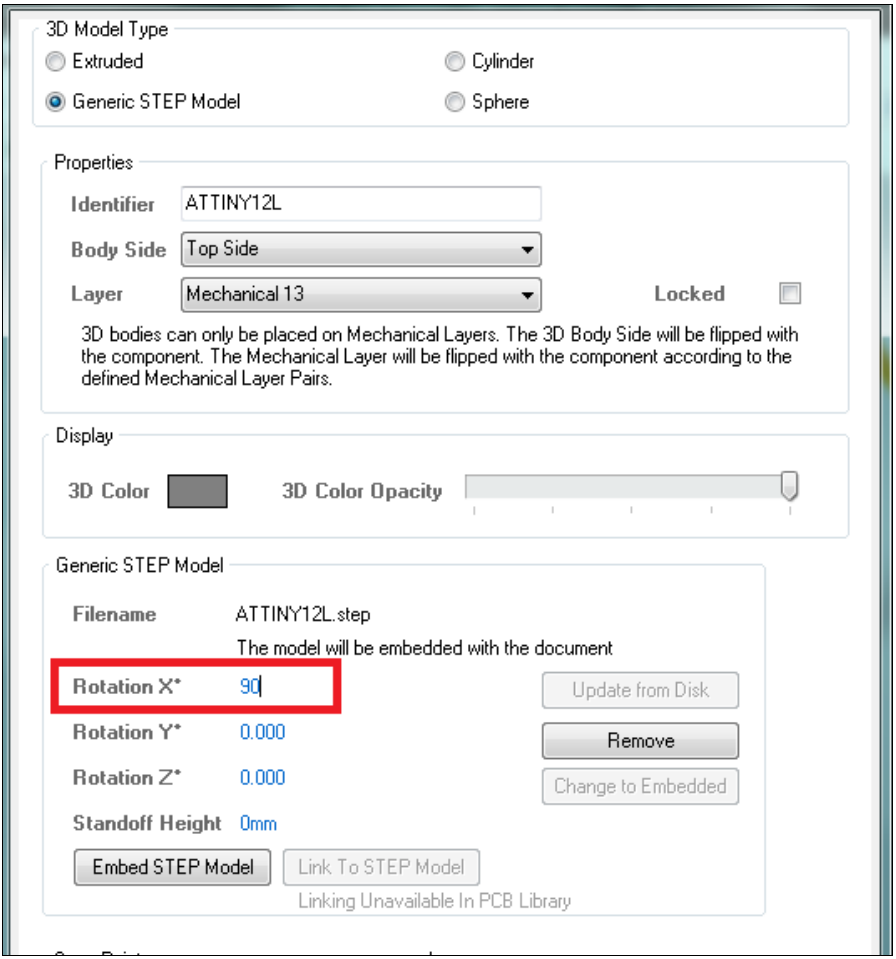


Рисунок 4.28 – Обертання доданої моделі

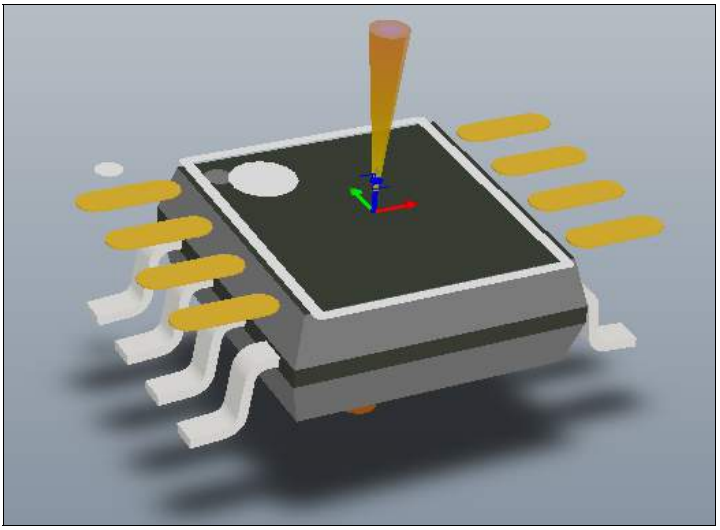


Рисунок 4.29 – Переміщення моделі на посадкове місце

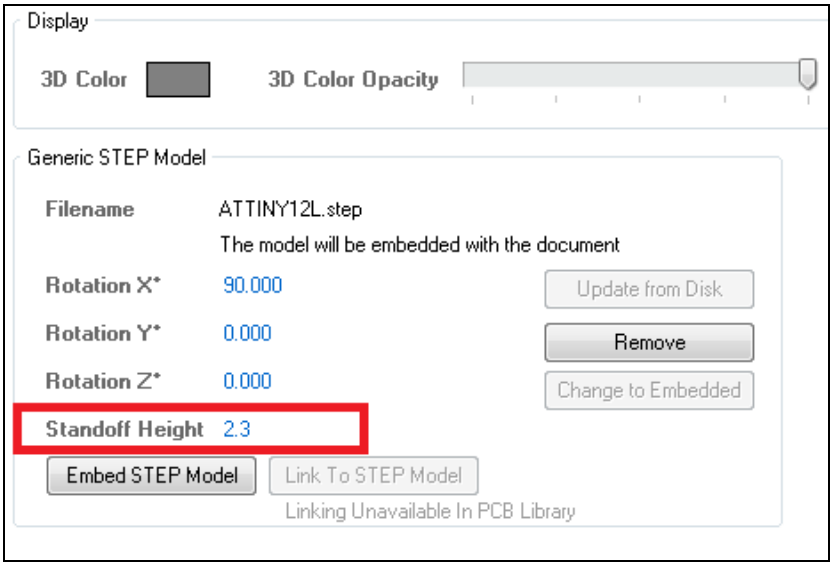


Рисунок 4.30 – Висота розташування моделі

Для компонента Резистор (R_0805) нашої бібліотеки створимо спрощену 3D модель. Перейшовши до режиму 2D, встановимо крок сітки 0,1мм та перейдемо до необхідного графічного шару (група Mechanical), у якому використовуючи команду Place > Line, намалюємо контур нашої моделі згідно рисунку 4.31.

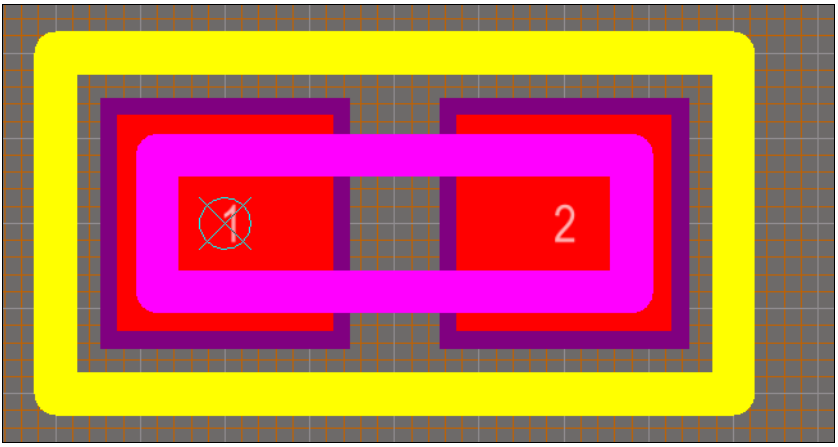


Рисунок 4.31 – Контур майбутньої моделі

Тепер вкажемо програмі, що створений нами контур є 3D моделлю. У вікні, яке викличемо командою Tools > Manage 3D Bodies for Current Component відображаються всі контури даного компоненту, серед яких методом перебору виберемо створений контур. В стовпчику Bode State вказується, чи застосовувати цей контур у якості 3D моделі, а у стовпчику Overall Height вказується, на яку довжину потрібно витягнути контур. Встановимо значення In Component R_0805 та 1 мм відповідно. Задамо бордовий колір для моделі (поле 3D Body Color) та у полі стовпчика Registration Layer вкажемо графічний шар, який буле зберігати цю 3D модель (рис.4.32).

Оскільки ми змінили компоненти бібліотеки, необхідно оновити компоненти, які знаходяться на платі. Перейдемо до файлу із платою та викличемо команду Tools > Update From PCB Libraries. З’явиться вікно, у якому питається, які шари плати необхідно оновити. Наступним відкриється вікно у якому детально описано на якому шарі та які зміни будуть внесені до плати (рис.4.33).

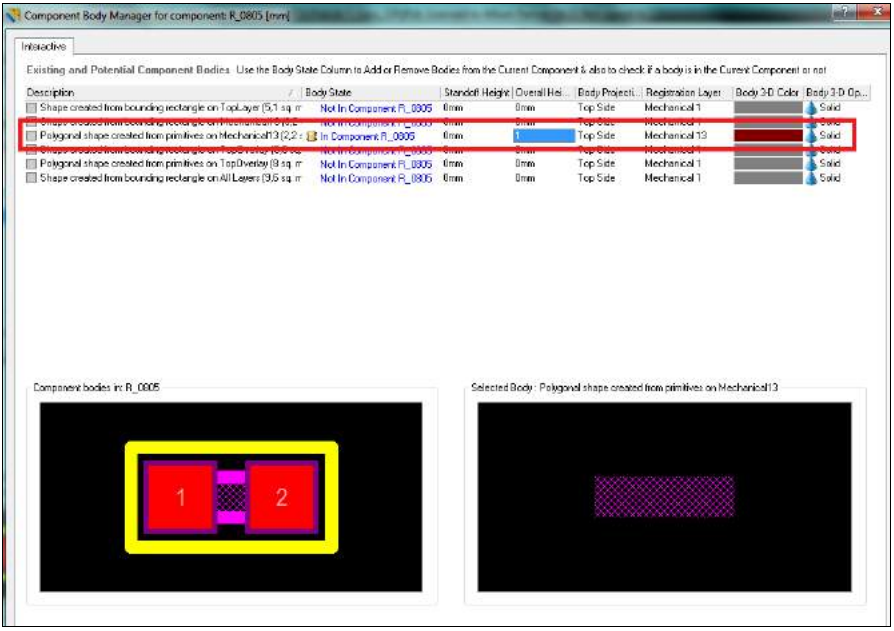


Рисунок 4.32 – Зазначення, що створений контур є 3D моделлю

Натиснемо клавішу Accept Changes, після чого відкриється вже знайоме вікно із списком модифікацій плати, де спершу натиснемо Validate Changes, а потім Execute Changes. У наступному вікні (рис.4.34) питається, які зміни дозволяється внести до плати. Зазвичай тут нічого міняти не потрібно тому натиснемо ОК, а потім закриємо вже не потрібні вікна.

Все, плата оновлена, це можна перевірити, перейшовши у режим 3D та подивившись на моделі компонентів (рис.4.35).

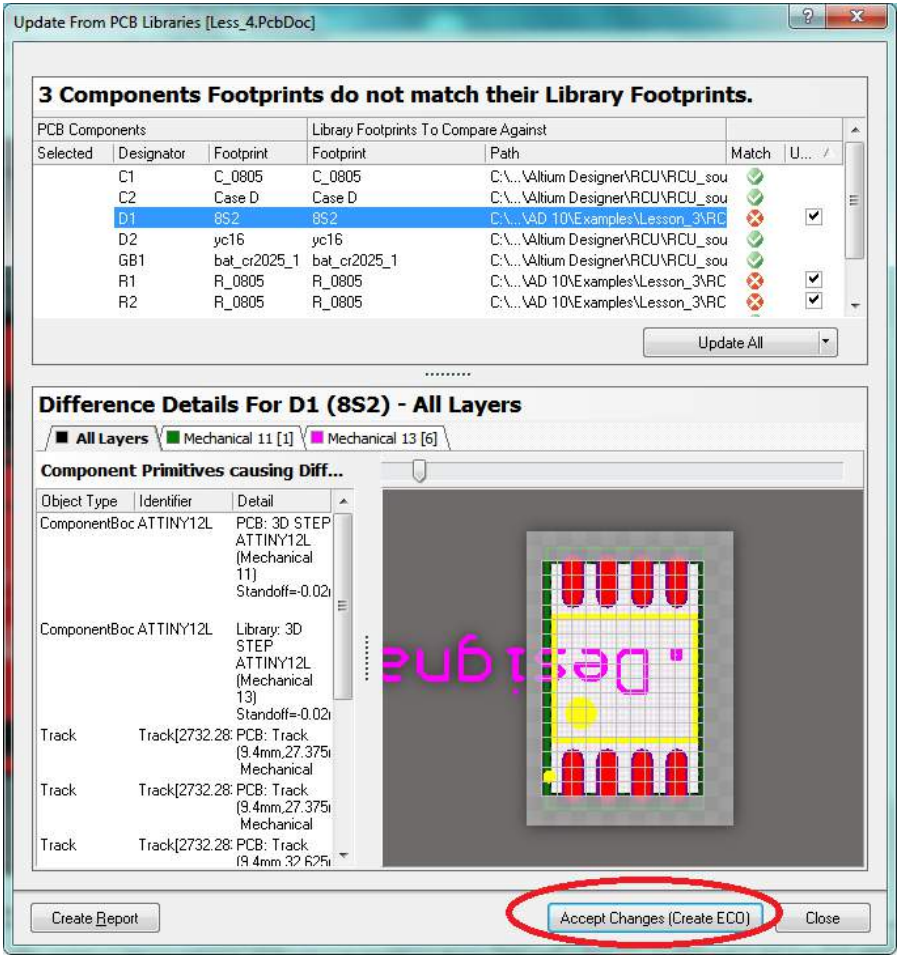


Рисунок 4.33 – Детальний список змін у платі

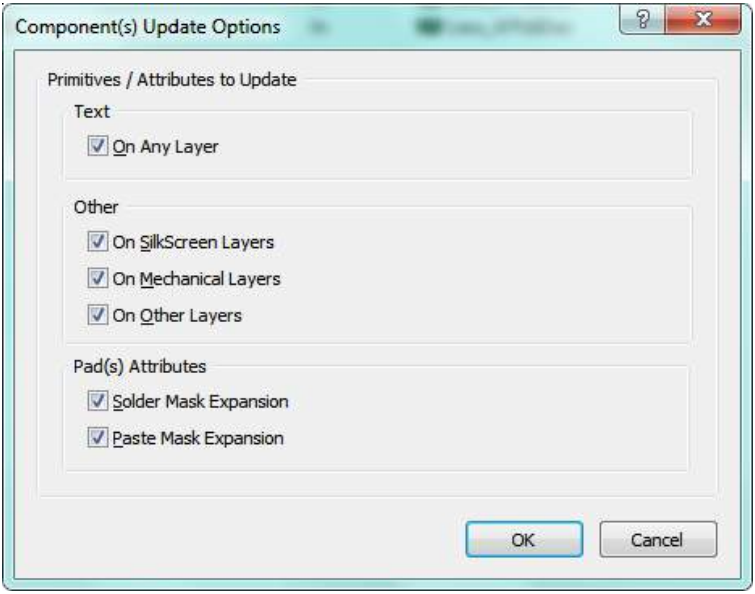


Рисунок 4.34 – Вибір можливих змін у платі

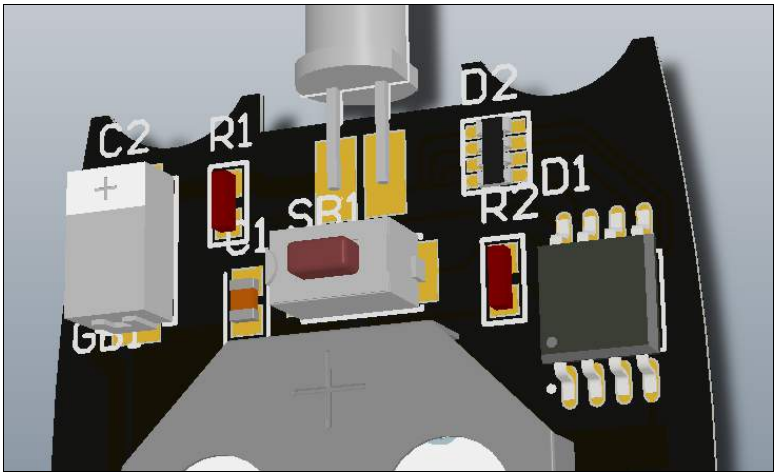


Рисунок 4.35 – Оновлена плата

5 ВСТАНОВЛЕННЯ ПРАВИЛ ПРОЕКТУВАННЯ

5.1 Створення правил для друкованої плати та принципової схеми

При розробці друкованої плати найбільш важливим етапом є встановлення правил проектування або, говорячи на мові інженера, конструктивних і технологічних обмежень проектування плати. Від встановлення правил залежить вся подальша робота над розробкою, тобто розміщення компонентів, трасування друкованих провідників і подальша верифікація проекту. Усі доступні у редакторі друкованих плат правила проектування діляться за функціональним призначенням на десять груп, кожній з яких у діалоговому вікні Design Rules виділена окрема вкладка.

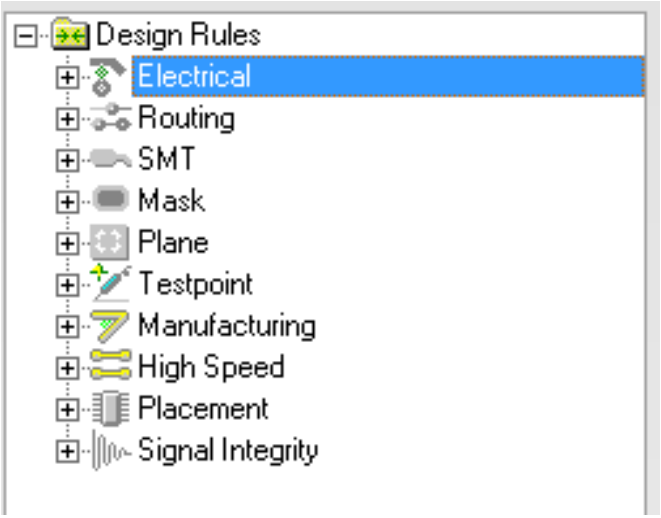


Рисунок 5.1 – Групи правил

У межах даного розділу будуть використані правила Clearance (група Electrical) та Length (група High Speed).

Clearance (проміжки) – визначає мінімально допустимий проміжок між будь-якими двома металізованими об'єктами на

сигнальному шарі. Це правило використовується для завдання відстані між провідниками на платі.

Length (обмеження довжини провідника) – визначає мінімальну та максимальну довжину провідника. Пріоритет має правило, яке визначає найменшу різницю між встановленими значеннями.

5.2 Практична робота з встановлення правил проектування

5.2.1 Створення правила друкованої плати

Спочатку відкриємо файл проекту DT01.PrjPcb, який знаходиться у папці Examples (…\Examples\Developer Tool - DT01), а тепер потрібно налаштувати проект. Встановимо метричну систему вимірювання, для цього скористаємось командою Design > Board Options. Також потрібно обрати, які правила будуть автоматично перевірятися під час роботи. Використаємо команду з меню Tools > Design Rules Check, виберемо пункт Rules To Check і поставимо відмітку у колонці Online навпроти потрібних правил (рис.5.2).

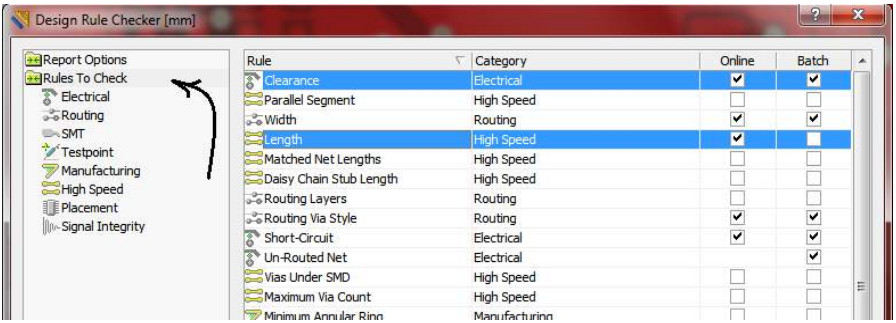


Рисунок 5.2 – Налаштування перевірки правил

Залишилось налаштувати відображення коментарів до правил, це можна зробити у меню DXP > Preferences > PCB Editor > DRC Violations Display (рис.5.3).

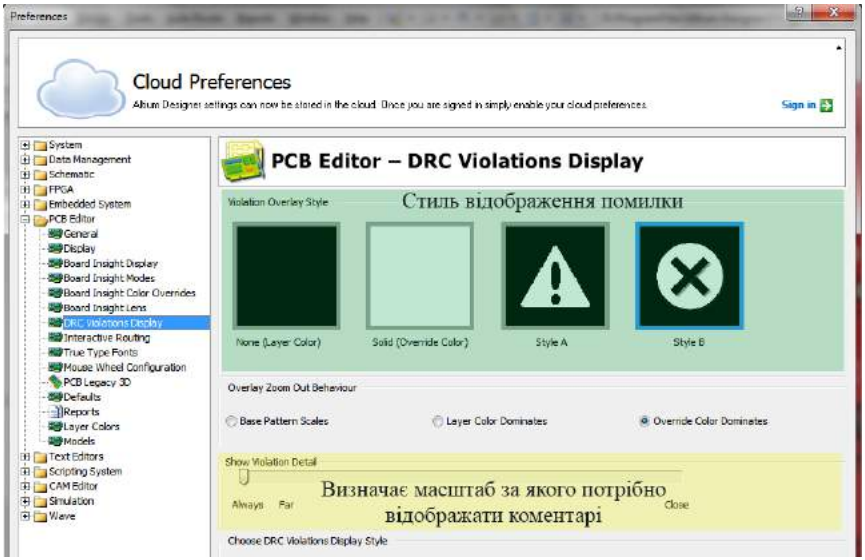


Рисунок 5.3 – Налаштування відображення коментарів

Викличемо вікно редактору правил за допомогою команди меню Design > Rules. Створимо нове правило. Для цього у потрібній категорії натискаємо правою клавішею миші та обираємо команду New Rule (рис. 5.4).

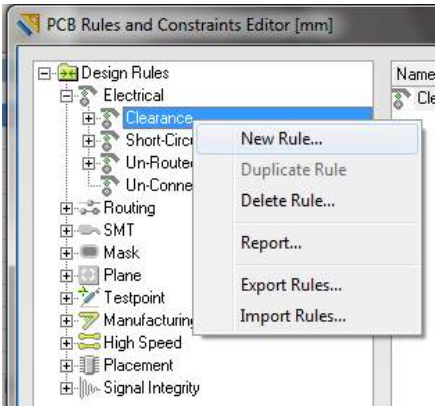


Рисунок 5.4 – Створення нового правила

У списку правил з’явилося нове правило, яке за замовчуванням має назву Clearance_1.

Перейдемо до редагування правила, вибравши його у списку та призначимо правилу унікальну назву. Назва правила задається у полі Name (рис. 5.5), слід відзначити, що правила повинні мати унікальні назви, інакше програма не збереже створені правила. У кожне ім’я будемо намагатись додати максимум інформації, для нашого прикладу задаємо ім’я – Clearance_GND_0.1mm.

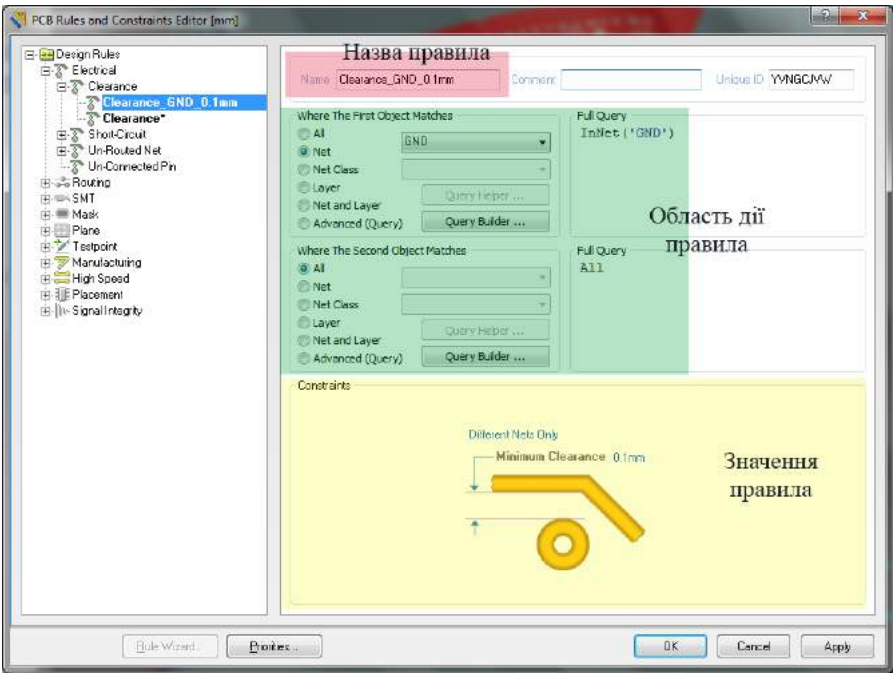


Рисунок 5.5 – Опис нового правила

Вкажемо область дії правила. Для обраного нами правила (Clearance) необхідно вказати, між якими електричними колами буде діяти вказане правило. Область дії може бути вказана вручну або за допомогою помічників Query Helper та Query Builder. Основною відмінність між цими помічниками є те, що Query Builder дає можливість створювати лише ті запити, які допустимі для даного виду

правила. При виборі області дії правила вручну можна обрати наступні варіанти:

- All - усі провідники плати;
- Net - одне обране електричне коло;
- Net Class - клас електричних кіл;
- Layer - усі кола розташовані на даному шарі;
- Net and layer - певне коло на даному шарі.

У нашому випадку задамо правило, яке вказує проміжок між колом GND та всіма іншими провідниками плати.

Для встановлення значення цього правила використовується нижня частина вікна PCB Rules (рис. 5.5). Задамо мінімальний проміжок між колом і всіма іншими провідниками плати – 0,1 мм.

Для правил можливо встановлювати пріоритети, оскільки області дії різних правил можуть перетинатися між собою. Так, наприклад, коло GND входить у наше правило, але також бере участь у базовому правилі, яке задає проміжки на всі електричні примітиви у платі. Для вирішення цієї проблеми потрібно встановити різні пріоритети, це можна зробити у спеціальному вікні Edit Rule Priorities, яке викликається клавішею Priorities у лівому нижньому куті екрану вікна PCB Rules. Правила виконуються у тому порядку, у якому вони перелічені в списку вікна Edit Rule Priorities, причому, якщо коло GND вже вказано у першому по порядку правилі, воно автоматично виключається з усіх наступних правил даної групи (рис.5.6).

Процедура створення правила завершується натисканням кнопки Apply у вікні створення правил.

Інколи виникає необхідність задати правило для окремого компонента, наприклад, щоб задати цей компонент, як виключення з інших правил. Це можна зручно зробити, натиснувши правою клавішею на потрібному виводі та обравши команду Find Similar Objects. У вікні, що з’явилося вкажемо, що критерій Component є важливим для пошуку і виберемо пункт Create Expression для генерації запиту (рис.5.8).

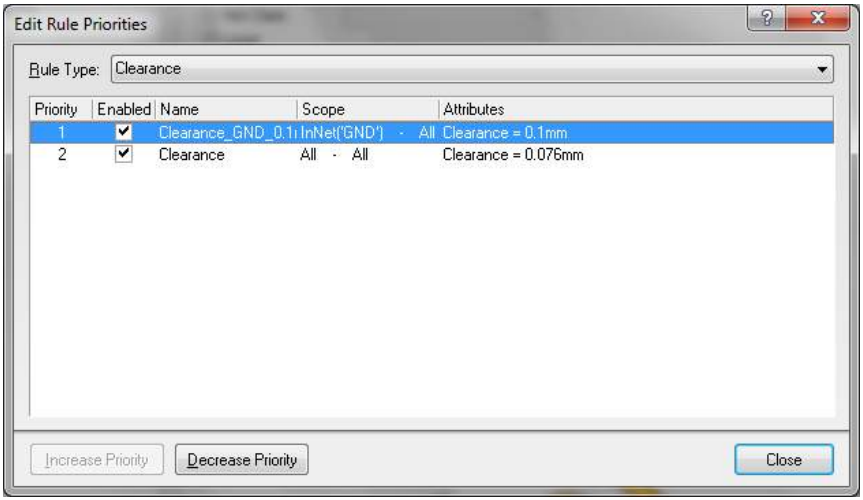


Рисунок 5.6 – Пріоритети виконання правил

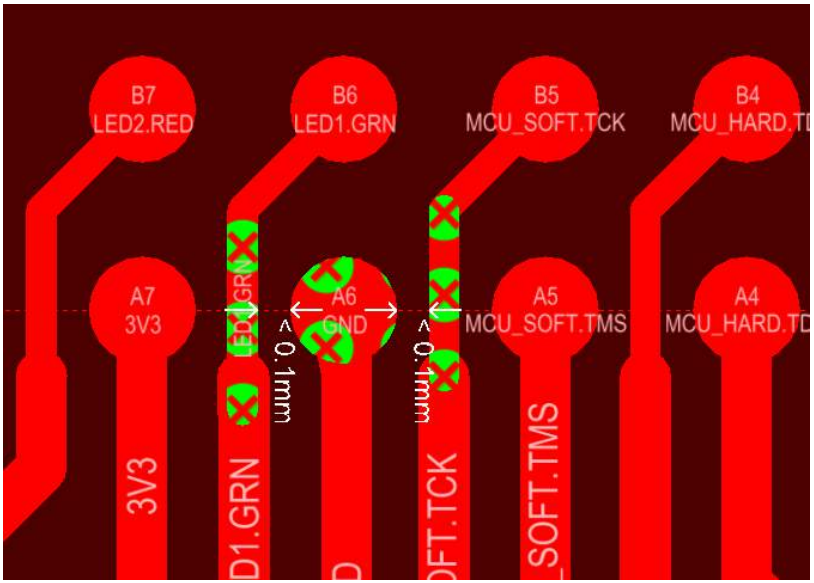


Рисунок 5.7 – Відображення місць, які не відповідають вимогам цього правила

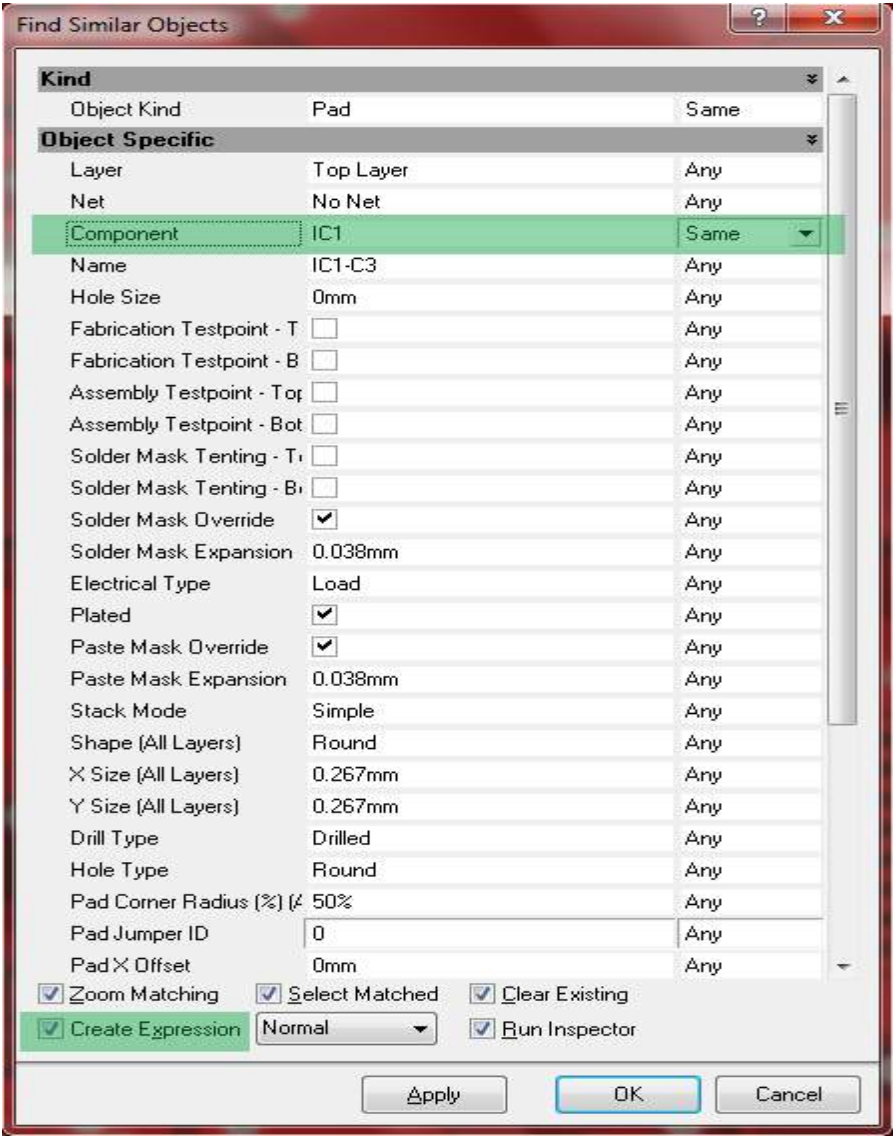


Рисунок 5.8 - Вибір критеріїв для виділення компонента

Далі з’явиться панель PCB Filter, у якій буде відображено згенерований запит і у робочому вікні на платі буде виділений обраний компонент. На основі згенерованого запиту можна одразу створити правило, для цього потрібно натиснути на клавішу Create Rule (рис. 5.9) і обрати тип правила. Потім з’явиться вікно редактору правил, у якому залишається задати лише значення цього правила (рис. 5.10).

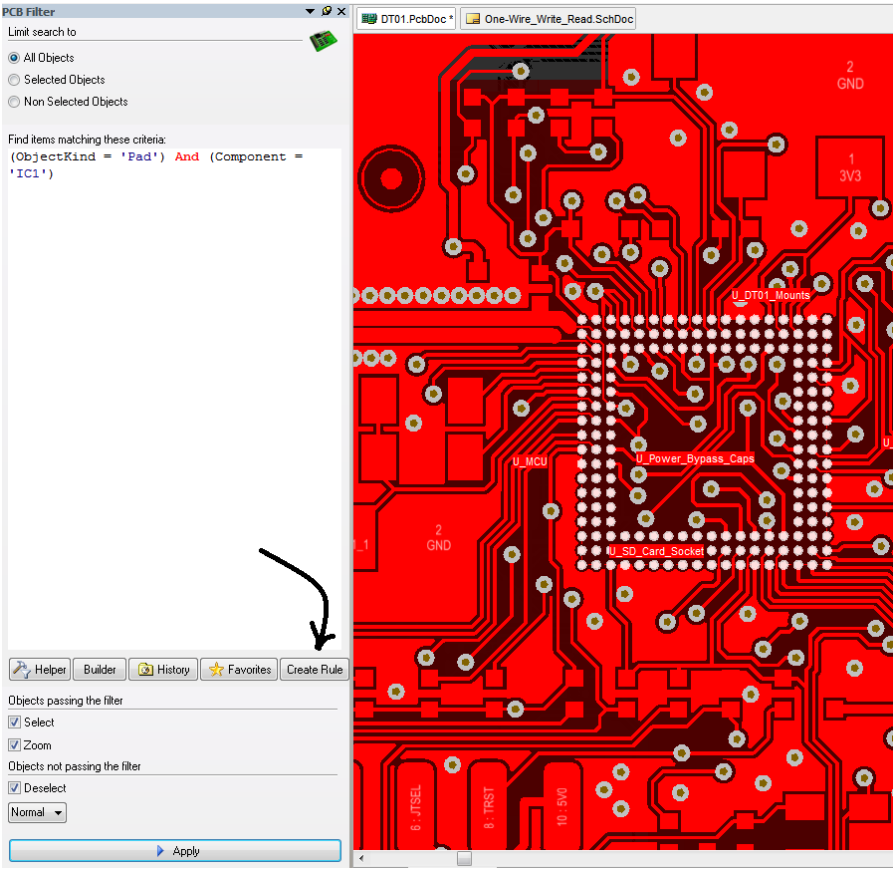


Рисунок 5.9 – Генерація запиту та виділення компоненту

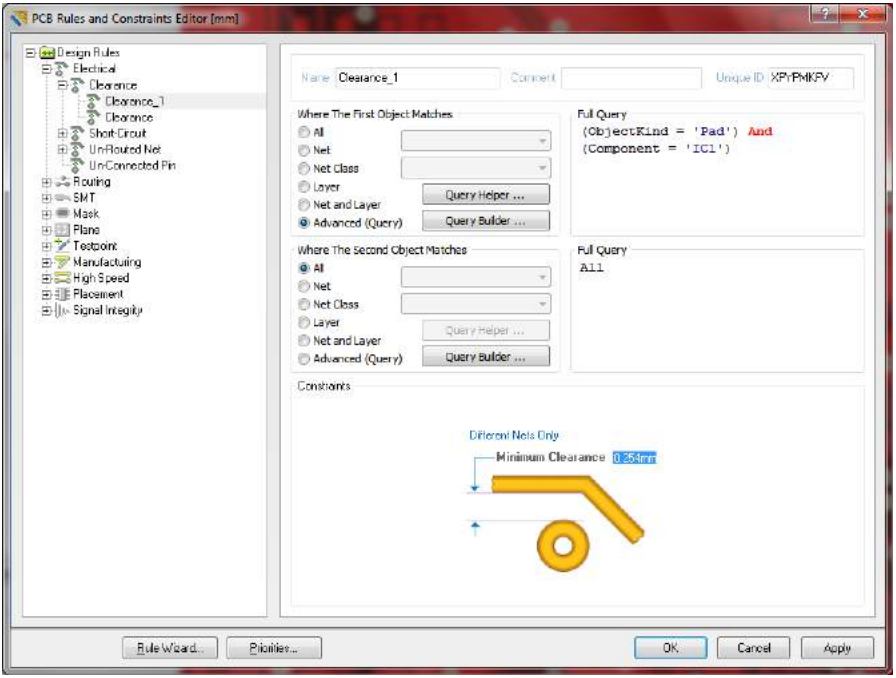


Рисунок 5.10 – Завдання значення правила, отриманого на основі запиту

5.2.2 Створення правила принципової схеми

Відкриємо файл проекту SL1 Xilinx Spartan-IIE PQ208 Rev1.01.PrjPcb, який знаходиться у папці Examples (…\Examples\SpiritLevel-SL1) і встановимо метричну систему вимірювання. Створювати правило будемо у схемі SL_LCD_SW_LED_2E.SchDoc, тому відкриємо її. Для створення правил проектування у принципових схемах використовують спеціальний інструмент – директиви, що знаходяться у меню Place > Directives (рис.5.11).

Створимо директиву (правило) яке буде задавати вирівнювання провідників у шині LEDS[7..0].

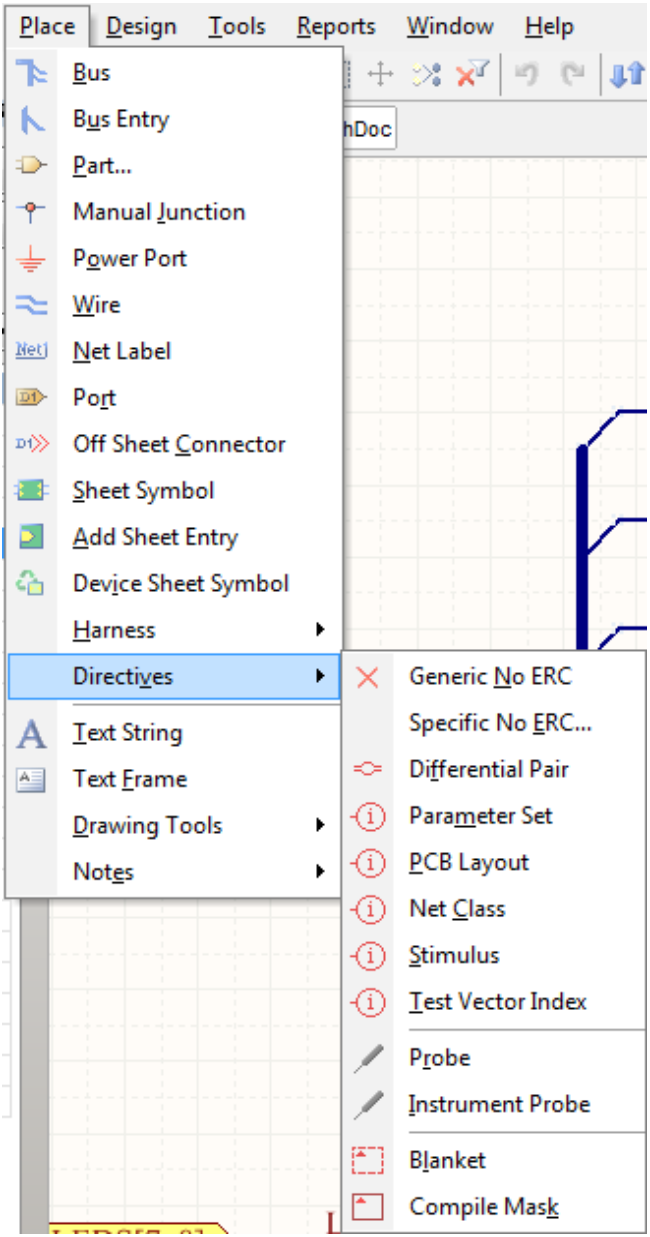


Рисунок 5.11 – Директиви для принципових схем

Для цього оберемо директиву Place > Directives > PCB Layout та розмістимо її на потрібній нам шині (рис. 5.12). За рахунок того, що ми розмістили директиву на шині, ми одразу вказали об’єкти правила. Об’єктами є усі кола, які входять у цю шину. Щоб вказати значення правила, натискаємо правою клавішею миші на директиві та обираємо команду Properties. Потім, заходимо у параметри правила, натиснувши на рядок Rule і для того, щоб задати значення, натискаємо на клавішу Edit Rule Values. Обираємо групу правил, у нашому випадку Length Constraint (група High Speed) і задаємо значення (див. рис. 5.13). Для того, щоб це правило відображалося у редакторі плат, потрібно його оновити командою Design > Update PCB Document, з’явиться вікно, у якому будуть відображені всі внесені зміни (рис. 5.14). Щоб передати зміни до плати, натискаємо на клавішу Execute Changes.

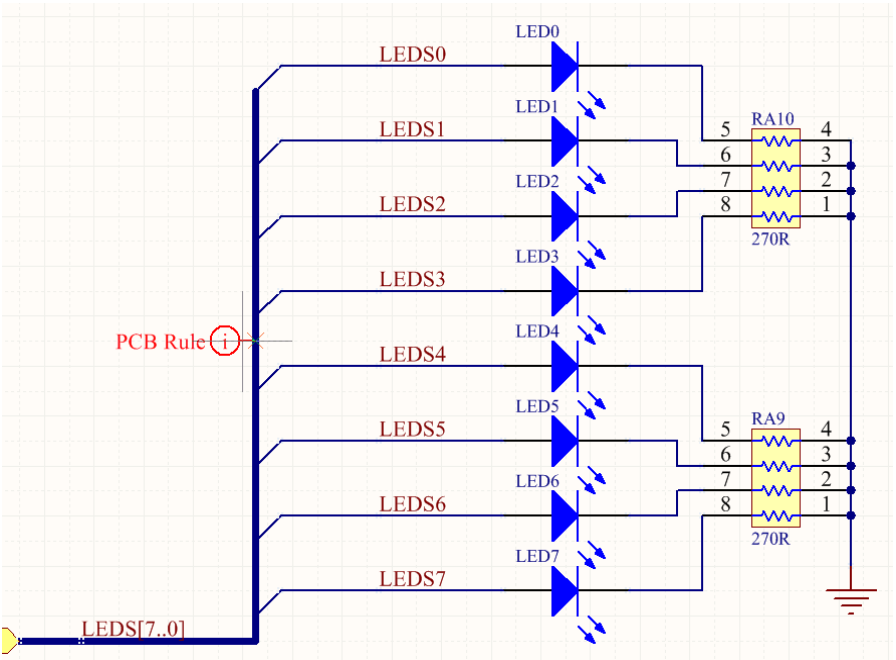


Рисунок 5.12 – Розміщення директиви

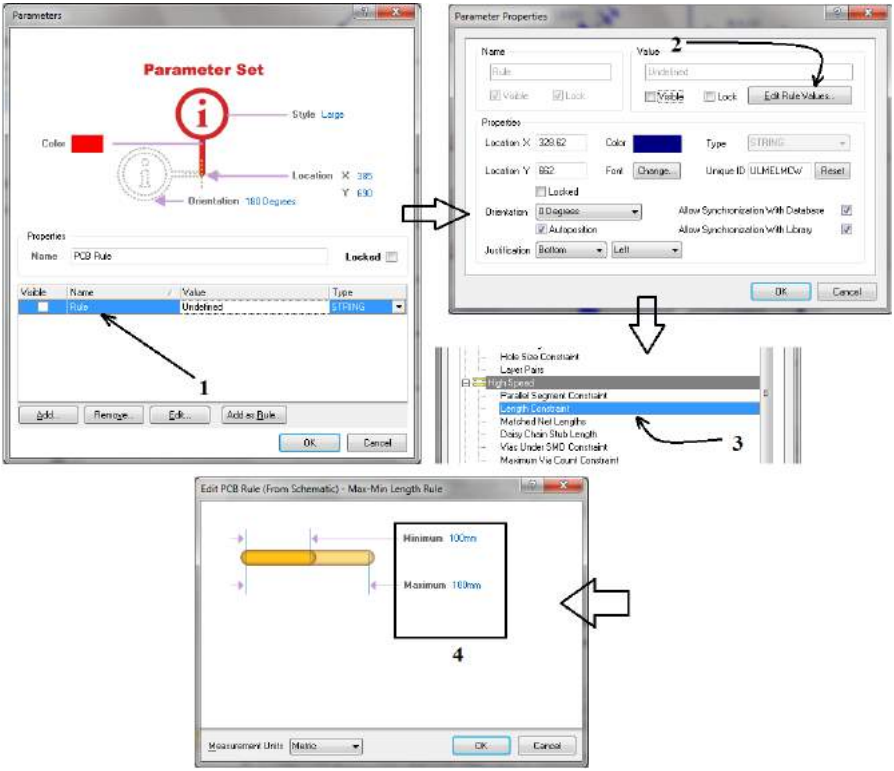


Рисунок 5.13 – Завдання значення правила

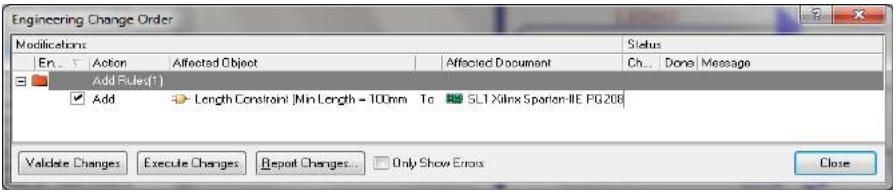


Рисунок 5.14 – Зміни, які будуть внесені до плати

6 МОДЕЛЮВАННЯ РОБОТИ СХЕМИ

6.1 Створення схеми для моделювання

Для схемотехнічного моделювання в пакеті Altium Designer використовується модуль Mixed SIM, що створений на основі програми-симулятора SPICE і є повнофункціональним аналогом пакету PSpice. Особливість Altium така, що наявність проекту для моделювання є обов'язковою умовою. Тому, створимо і збережемо проект друкованої плати File > New > Project > PCB Project, а потім додамо до цього проекту файл електричної схеми File > New > Schematic. За допомогою кнопки System в правому нижньому куті відкриємо панель Libraries, де користуючись однойменною клавішею підключимо наступні бібліотеки:

- Miscellaneous Connectors.IntLib (Altium/AD 10/Library)
- Miscellaneous Devices.IntLib (Altium/AD 10/Library)
- Simulation Sources.IntLib (Altium/AD 10/Library/Simulation) (рис.6.1)

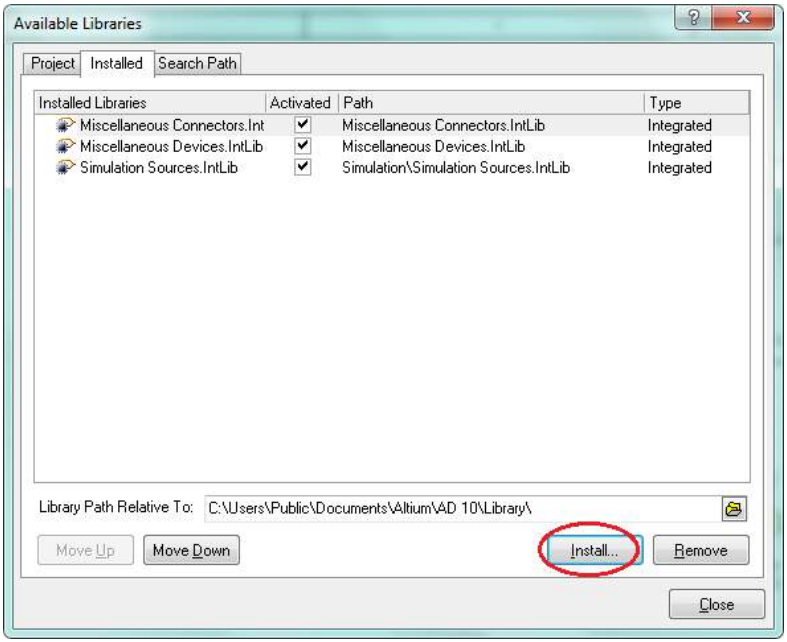


Рисунок 6.1 – Підключення зовнішніх бібліотек

Ці бібліотеки є стандартними і входять до стандартного комплекту програми. Праворуч від назв бібліотек у дужках вказано їх відносне розташування.

Також нам необхідно додати до проекту бібліотеку library.SchLib. Це можна зробити, виконавши команду Project > Add Existing to Project або відкривши бібліотеку командою File > Open і перетягнувши її до структури проекту у панелі Projects (рис.6.2).

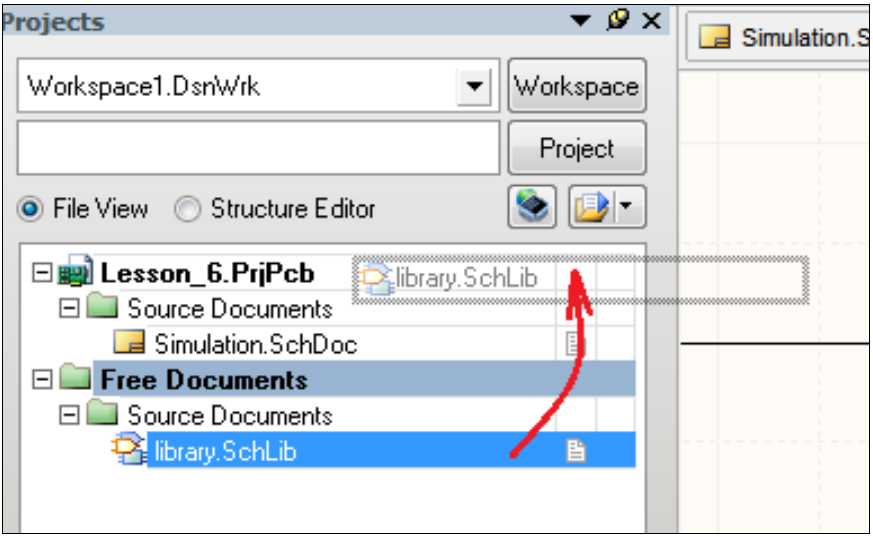


Рисунок 6.2 – Додавання бібліотеки до проекту

Далі розмістимо компоненти згідно з рис. 6.3. Резистор (Res2) та конденсатор (Cap) візьмемо із бібліотеки Miscellaneous Devices, а операційний підсилювач (UA741) із бібліотеки Library.

Відкриємо для будь-якого з резисторів вікно властивостей, викликавши контекстне меню і вибравши пункт Properties. У полі Models буде запис із типом Simulation, він вказує на модель формату SPICE, яка буде описувати даний компонент під час його симуляції (рис.6.4).

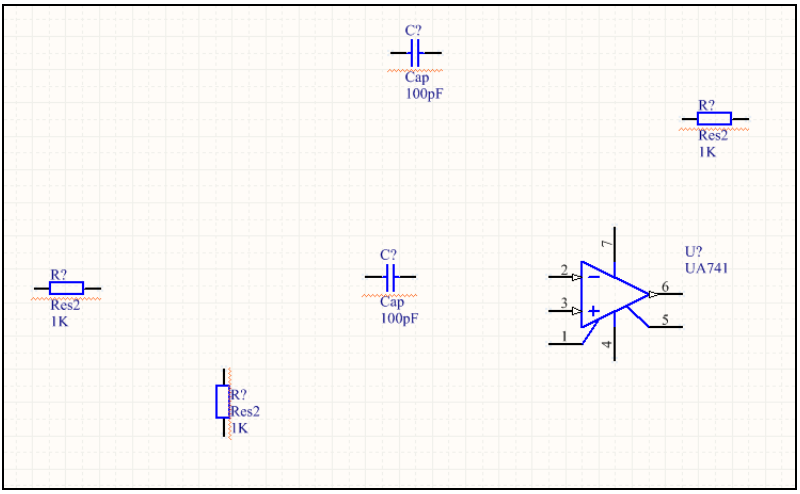


Рисунок 6.3 – Розміщення компонентів

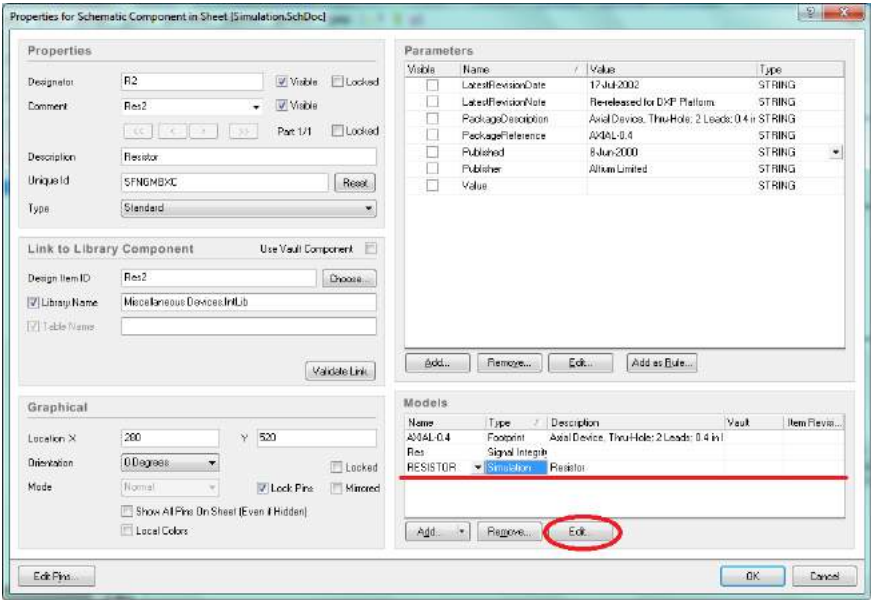


Рисунок 6.4 – Вікно властивостей компоненту

Виберемо його і натиснемо клавішу Edit, з’явиться вікно властивостей моделі (рис.6.5). Воно містить три вкладки:

- Model Kind – опис моделі; тут вказується одна із вбудованих моделей або дається посилання на користувацьку модель;
- Parameters – параметри (характеристики), які задаються лише для вбудованих моделей;
- Port Map – встановлення відповідності між виводами УГП компоненту та виводами моделі, що його описує.

Для моделювання можна використовувати вбудовані моделі. Це моделі, опис яких вже міститься у модулі схемотехнічного моделювання Altium Designer, або користувацькі моделі.

Оскільки модель операційного підсилювача не містить моделі, а наявність моделей у всіх компонентів схеми є необхідною умовою симуляції, підключимо її. Для цього, спершу додамо файл моделі UA741.ckt до складу проекту, це робиться аналогічно до додавання бібліотеки (дивись вище). Потім відкриємо вікно властивостей підсилювача і у полі Models натиснемо клавішу Add та виберемо тип Simulation. У вкладці Model Kind, вікна властивостей моделі, вкажемо тип моделі General, і підтип Spice Subcircuit, який свідчить про те, що модель братиметься із зовнішнього файлу. Після цього натиснемо клавішу Browse та у вікні, що з’явиться виберемо додану модель (рис. 6.6).

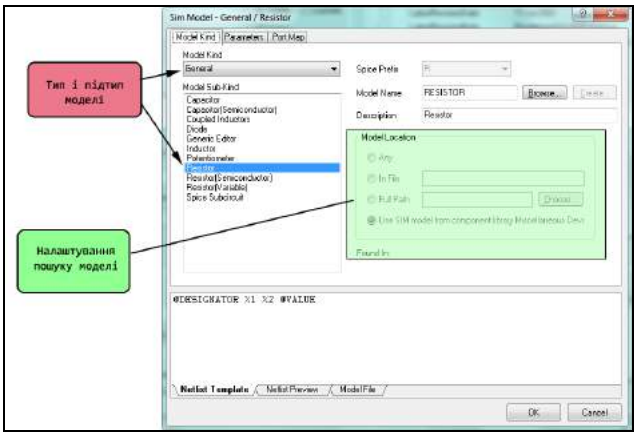


Рисунок 6.5 – Вікно властивостей моделі

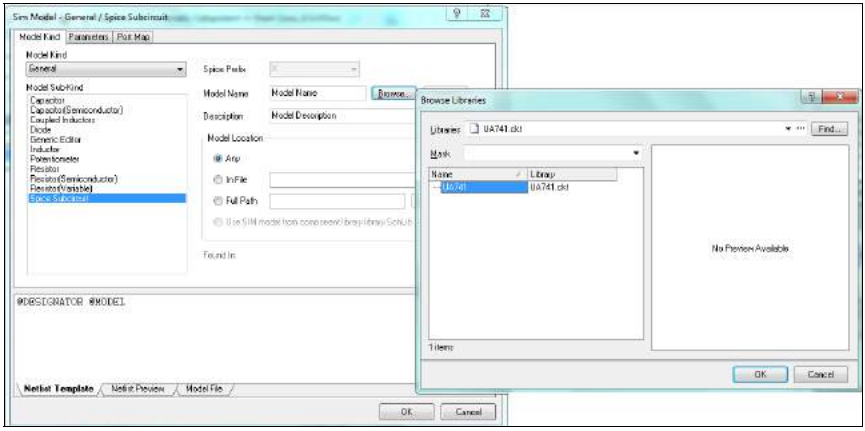


Рисунок 6.6 – Підключення моделі компоненту

Перевірити те, що модель була завантажена в програму можна відкривши поле-вкладку Model File, яка містить текст файлу SPICE моделі. Не закриваючи вікно властивостей моделі, перейдемо до вкладки і задамо відповідності між виводами як вказано на рис. 6.7 та натиснемо ОК.

Тепер використовуючи команду Place > Wire створимо кола між компонентами та розмістимо порти «землі» (нульовий потенціал), позитивного і від'ємного живлення відповідно до рисунку 6.8. Для цього будемо використовувати команди із панелі інструментів на рис. 6.9. Наявність «землі» для моделювання є обов'язковою.

На створені порти «землі» і живлення повинні подаватися якісь сигнали. В Altium для встановлення у системі моделювання напруг живлення, струмів і вхідних сигналів стандартної форми застосовуються спеціальні компоненти, що описують джерела постійних та змінних напруг і струмів. Ці компоненти знаходяться в стандартній бібліотеці Simulation Sources.IntLib.

Розмістимо два джерела напруги VSRC і одне джерело імпульсної напруги VPULSE, яке буде подавати сигнали на вхід нашої схеми. Перше джерело напруги формуватиме сигнали порту VCC, друге порту VEE, а від'ємного виводи обох компонентів підключимо до землі (рис.6.10). У полі Value вікна властивостей компоненту

(рис.6.11), для портів VCC та VEE встановимо значення +15V і -15V відповідно.

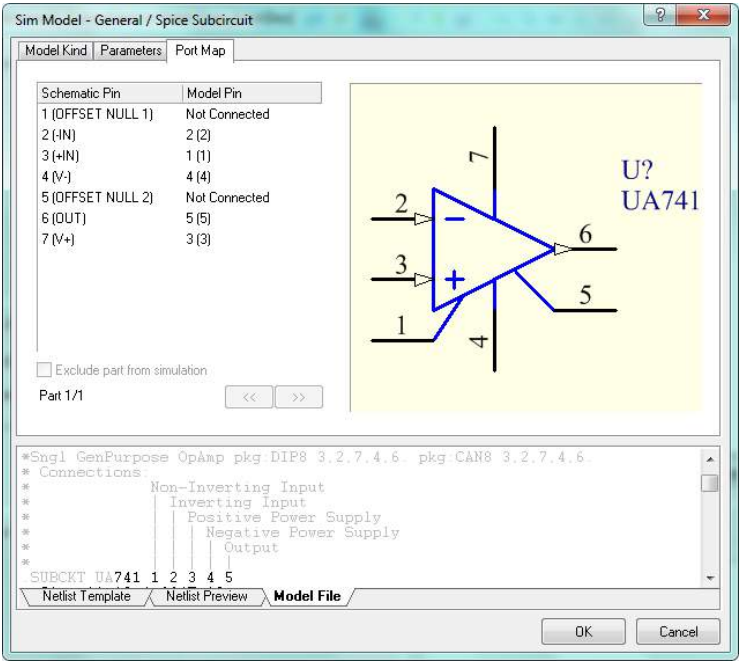


Рисунок 6.7 – Завдання відповідностей для виводів

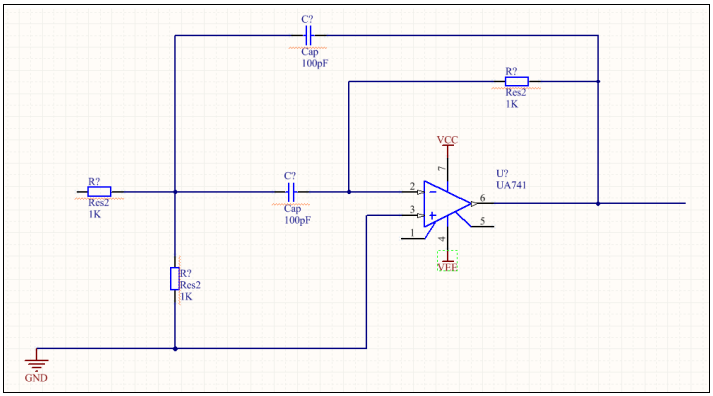


Рисунок 6.8 – Схема без джерел напруги та сигналів

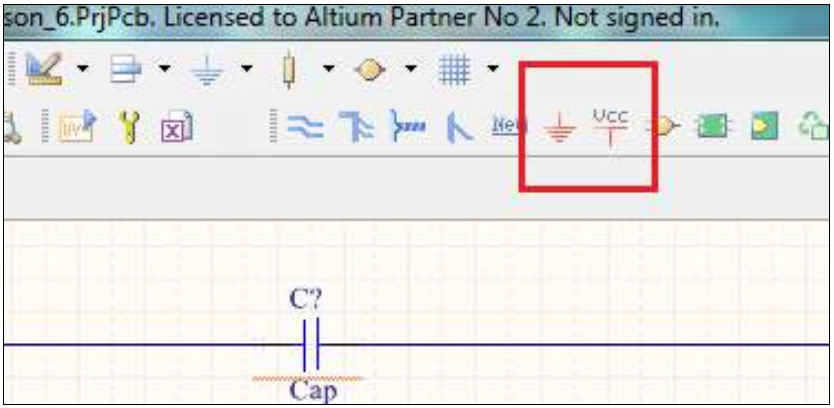


Рисунок 6.9 – Команди розміщення землі та живлення

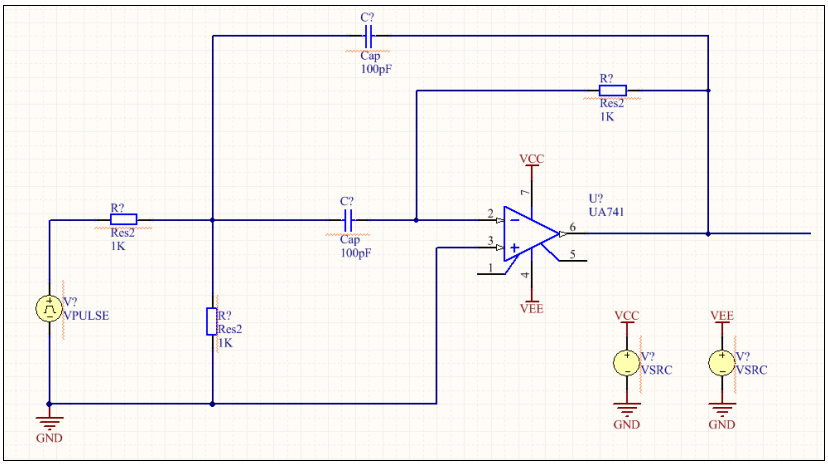


Рисунок 6.10 – Розміщення джерел напруги та сигналів

Для джерела імпульсних сигналів задамо параметри його моделі як на рис. 6.12. При завданні чисельних значень параметрів компонентів використовуються літерні множники, які набираються латинськими літерами. Допустимі множники наведені в табл.6.1.

Встановимо номінали компонентів згідно до рис. 6.13. Розділовим знаком у числах має бути тільки крапка.

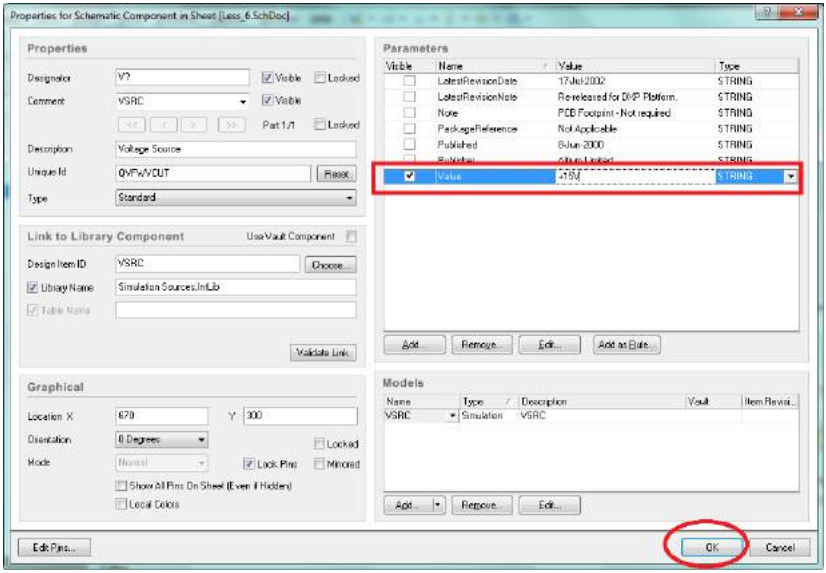


Рисунок 6.11 – Задання номіналу джерела напруги

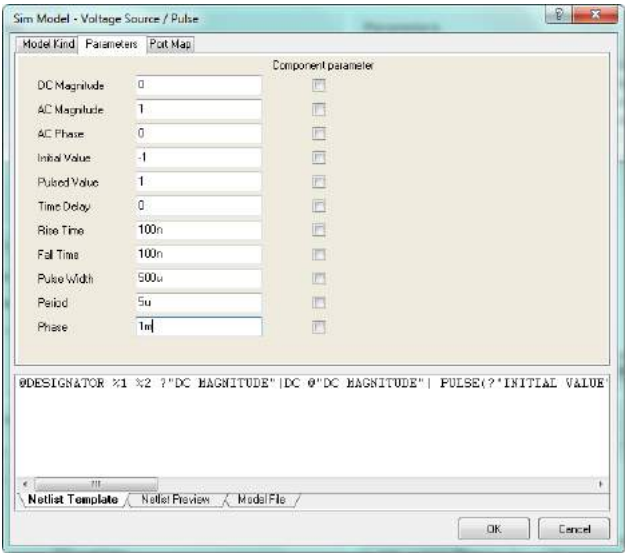


Рисунок 6.12 – Встановлення параметрів моделі

Таблиця 6.1 – Множники параметрів компонентів

Піктограма	Назва команди
T	10^{12}
G	10^{9}
Meg	10^{6}
K	10^{3}
mil	25.4^{-6}
m	10^{-3}
u	10^{-6}
n	10^{-9}
p	10^{-12}
f	10^{-15}

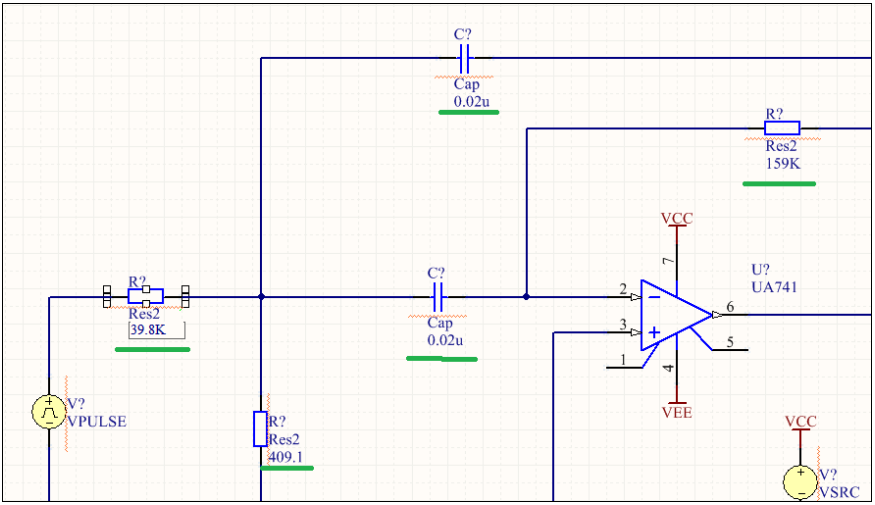


Рисунок 6.13 – Встановлення номіналів компонентів

Для зручності задамо назви тим колам, із яких ми будемо зчитувати показники. Зробимо це за допомогою команди Place > Net Label. Слід зауважити, що ім'я кола не може складатися лише з цифр. Пронумеруємо компоненти схеми командою Tools > Annotate Schematic Quietly. Якщо схема (рис.6.14) містить помилки, то схема не пройде компіляцію (перевірку на помилки), а успішна компіляція - це обов'язкова умова для проведення моделювання.

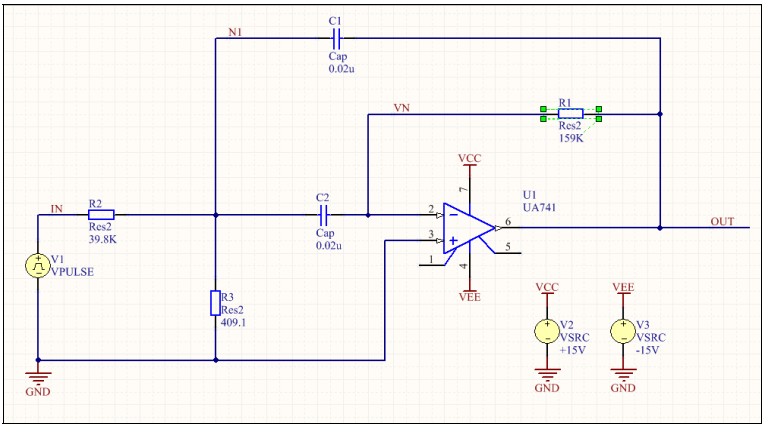


Рисунок 6.14 – Готова схема

6.2 Моделювання та обробка результатів

Наступним кроком є моделювання схеми. Відкриємо панель для моделювання Mixed Sim (рис.6.15), натиснувши праву клавішу миші на вільному місці у рядку меню і вибравши зі списку доступних панелей необхідну.

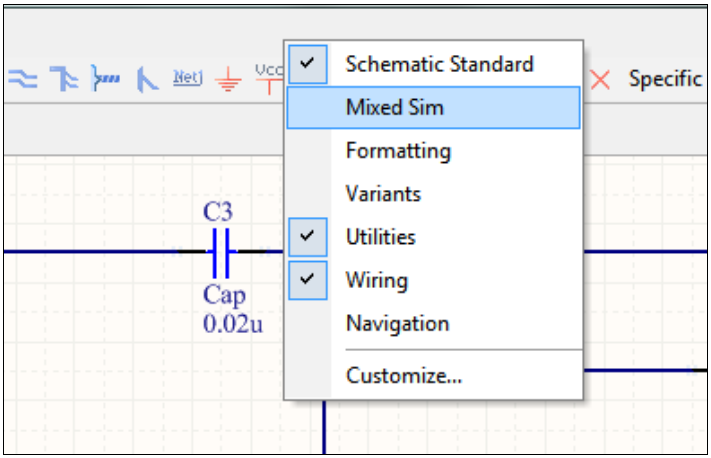


Рисунок 6.15 – Відкриття панелі для моделювання

Панель, що з’явиться має три команди:

- Run Mixed Signal Simulation – запуск процесу моделювання;
- Setup Mixed-Signal Simulation – налаштування моделювання;
- Generate XSpice Netlist – формування загальної моделі схеми, при цьому відбувається перевірка схеми.

Складемо завдання для моделювання, для цього відкриємо вікно Analyses Setup (рис.6.16) натисканням клавіші Setup Mixed-Signal Simulation. У цьому вікні задаються необхідні види аналізу і вибираються схемні змінні (сигнали), тобто напруга у вузлах схеми, струм у колах схеми, комплексні опори, потужність розсіювання на елементах схеми для їх зберігання в файлі результатів та графічного відображення. Проведемо частотний аналіз схеми – вкладка AC Small Signal Analysis, у полі Analyses/Options. В налаштуваннях вкажемо початкову частоту 1Гц та кінцеву частоту 5кГц і тип - лінійний тип зміни частоти.

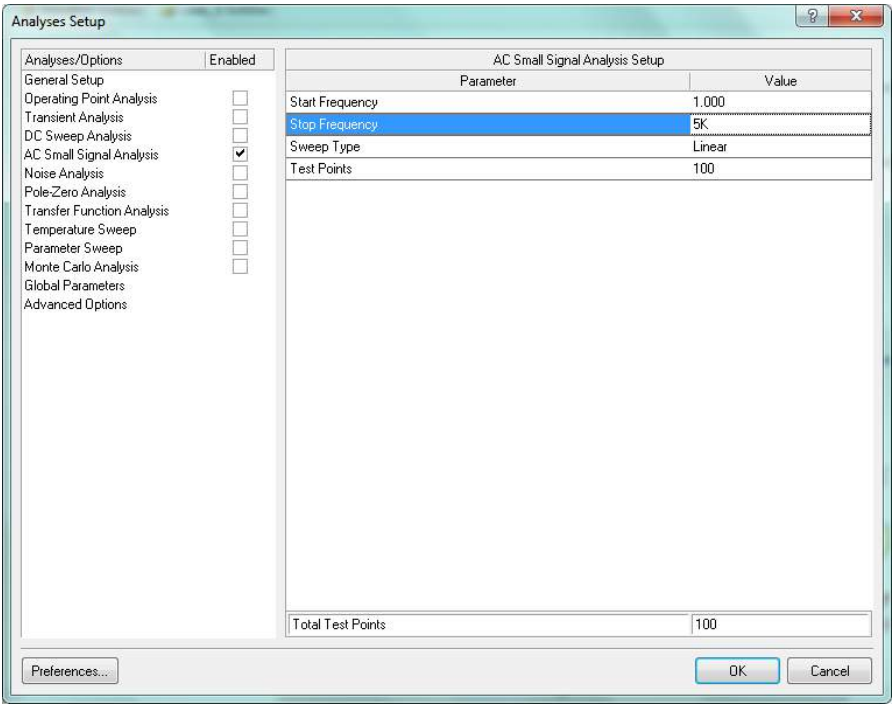


Рисунок 6.16 – Налаштування вибраного аналізу схеми

Після цього перейдемо до вкладки General Setup, яка містить загальні налаштування для вибраних видів аналізу, де у полі Collect Data For вказуються режим збереження даних щодо схемних змінних при моделюванні. Тобто, які схемні змінні та для яких елементів схеми будуть зберігатися дані при моделюванні. Наступне поле Sheets to Netlist визначає об'єкт моделювання – лист схеми або проект, а поле SimView Setup - чи потрібно зберігати налаштування виведення результатів. Нижче містяться два поля: Available Signals, у якому наведений перелік усіх схемних змінних, доступних відповідно до обраного режиму; та поле Active Signals, яке містить перелік сигналів, графіки яких будуть побудовані по закінченню моделювання.

Додаткові позначення в полі Available Signals:

- #branch – струм через джерело напруги;
- [i] – струм через двополюсний компонент;
- [p] – потужність розсіювання;
- [z] – модуль комплексного опору;
- [ib] – струм бази транзистора;
- [ic] – струм колектору транзистора;
- [ie] – струм емітера транзистора.

Встановимо значення цієї вкладки відповідно до рис. 6.17.

Закривши вікно натисканням клавіші ОК, запустимо моделювання командою Run Mixed Signal Simulation. По завершенню моделювання відкриється графік із амплітудно-частотною характеристикою нашої схеми (рис.6.18). Керування результатами моделювання можна здійснювати у спеціальній панелі Sim Data, яка відкривається однойменною клавішею у правому нижньому кутку.

В режимі роботи з результатами моделювання стають доступними три вкладки Chart, Plot і Wave. Вкладка Wave відповідає за графічне відображення сигналу; Plot – де це графічне відображення сигналу розміщується, тобто осі координат; та Chart – сторінку для відображення результатів аналізу (рис.6.19).

Наприклад, створимо новий графік (рис.6.20). Для цього виберемо пункт вкладки, після чого відкриється майстер створення графіків. На першому кроці треба вказати ім'я нового графіка, на другому - сітки, які необхідно відображати, а на третьому – сигнали для відображення на графіку.

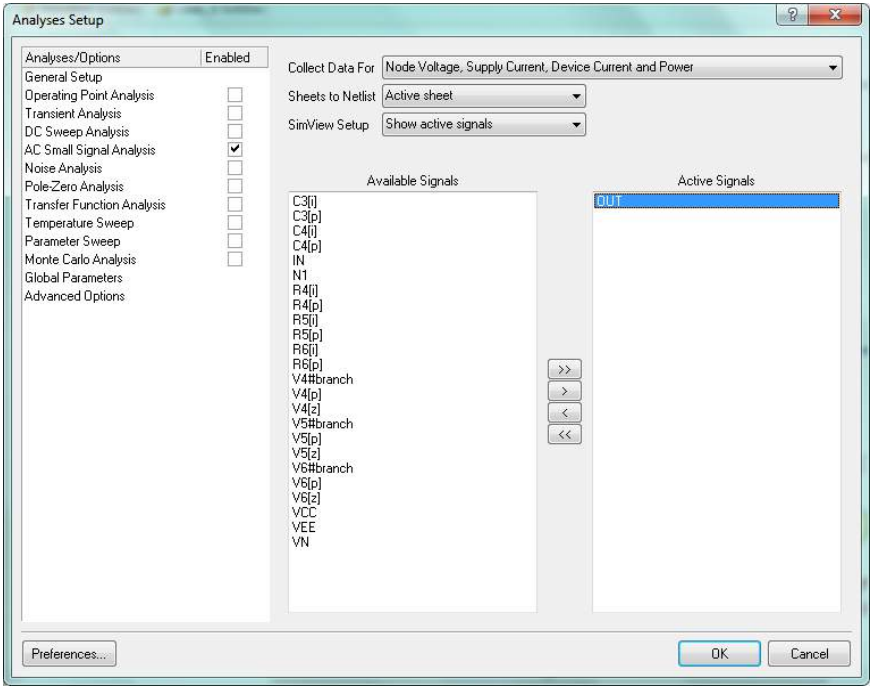


Рисунок 6.17 – Загальні налаштування моделювання

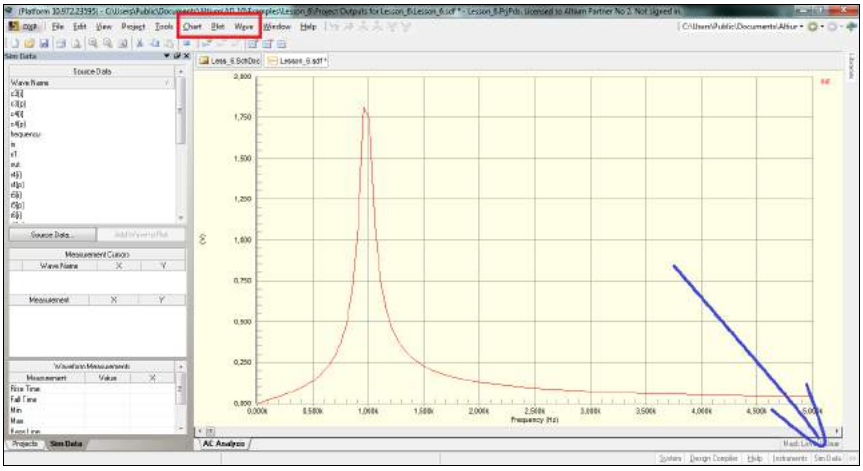


Рисунок 6.18 – Результат моделювання

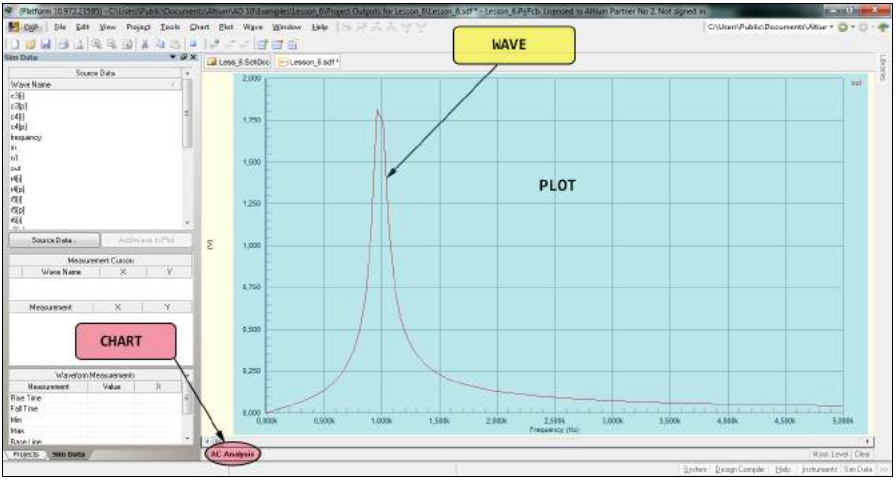


Рисунок 6.19 – Робота із результатами моделювання

Рисунок 6.20 – Додавання нового графіку

При додаванні сигналу вкажемо сигнал, але для відображення виберемо не просто його величину, а його фазове значення (рис.6.21).

З’явиться новий графік, на якому відображена фазо-частотна характеристика (рис.6.22).

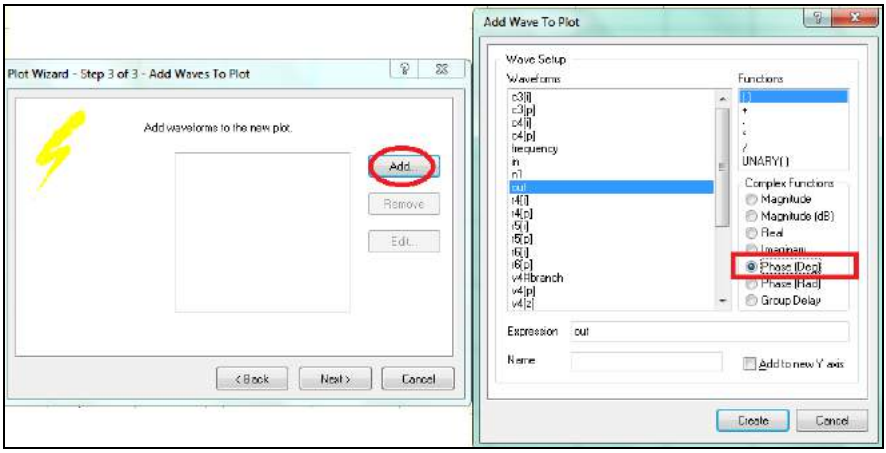


Рисунок 6.21 – Задання сигналу для побудови графіку

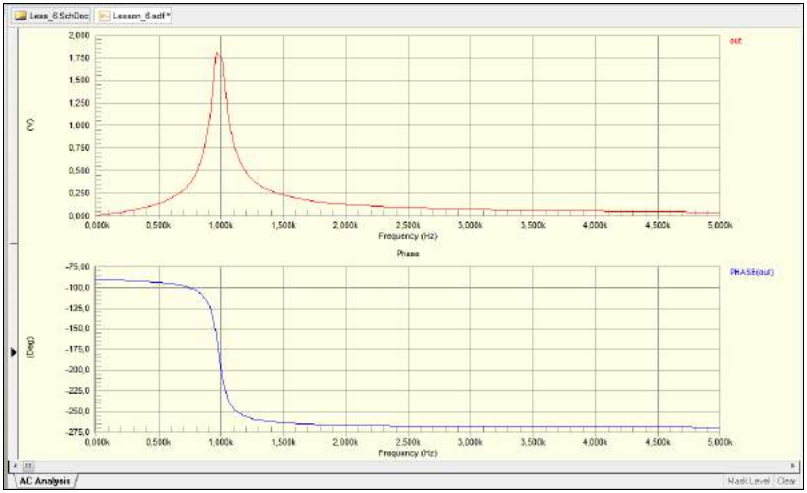


Рисунок 6.22 – Графіки моделювання

7 СТВОРЕННЯ ІНТЕГРОВАНОЇ БІБЛІОТЕКИ

7.1 Формування бібліотеки та додавання елементів

Для створення схеми достатньо мати бібліотеки компонентів у вигляді *.SchLib та *.PcbLib у структурі проекту, але в такому випадку ускладнюється процедура структурування бібліотек та подальшого їх використання всередині підприємства. Файли бібліотек символів та посадкових місць рекомендується об'єднувати у проект, який має назву інтегрованої бібліотеки. Рекомендується при створенні схем використовувати тільки інтегровані бібліотеки. Використання інтегрованої бібліотеки має наступні переваги: усі символи (УГП), Spice моделі та топологічні посадкові місця зберігаються у єдиному файлі, також є можливість компіляції бібліотеки, за рахунок чого досягається її відлагодження, крім того інтегровані бібліотеки можна використовувати для створення баз даних.

7.2 Практична робота з інтегрованою бібліотекою

Спочатку відкриємо файли RCU.SchLib та RCU.PcbLib, створені у попередніх розділах. Для створення інтегрованої бібліотеки виконаємо команду File > New > Project > Integrated Library, після чого у структурі панелі Project з'явиться новий документ, який потрібно одразу зберегти. Тепер потрібно додати до структури створеного проекту, раніше створені бібліотеки, шляхом їх переміщення у дереві панелі Project. Слід відзначити, що створений файл має розширення *.LibPkg, а не *.IntLib, тобто у даний момент ми маємо не саму інтегровану бібліотеку, а файл заготовки, із якого буде сформована інтегрована бібліотека у процесі компіляції.

Щоб скопіювати символ (УГП) з обраної бібліотеки у бібліотеку користувача, потрібно одночасно відкрити обидві бібліотеки. Відкрити обрану інтегровану бібліотеку можна командою File > Open та обравши потрібну бібліотеку. У нашому випадку …\Library\Miscellaneous Devices.IntLib (рис.7.1). При спробі це зробити на екрані з'явиться вікно, у якому пропонується виконати дві дії над бібліотекою Extract Sources (відкрити) або Install Libraries (встановити) (рис.7.2). В нашому випадку обираємо Extract Sources. В результаті у панелі будуть завантажені дві бібліотеки (символи та

посадкові місця), об'єднані файлом проекту Miscellaneous Devices.LibPkg.

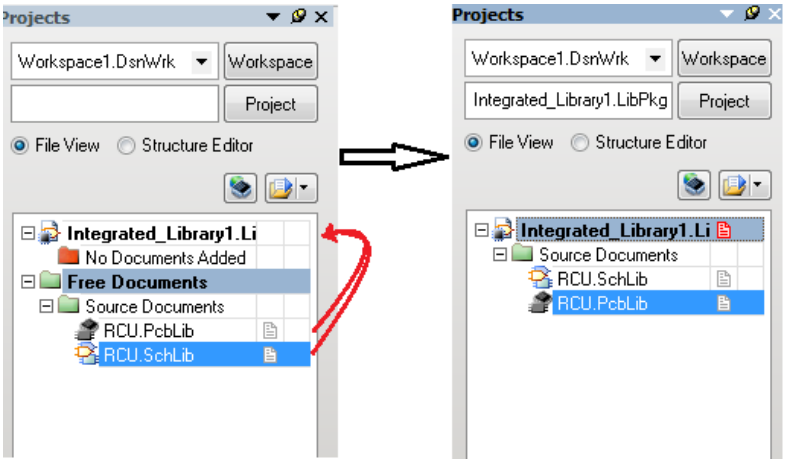


Рисунок 7.1 – Створення інтегрованої бібліотеки

Після цього відкриваємо з панелі Projects бібліотеку символів та знаходимо у ній компонент ADC – 8 (через панель SCH Library) та копіюємо його натиснувши на ньому правою клавішею миші і обравши команду Copy (рис.7.3). Далі відкриваємо бібліотеку користувача, і натискаємо правою клавішею миші у списку компоненті панелі SCH Library та обираємо команду Paste. За необхідності копіювання декількох компонентів одразу, їх слід виділяти із натиснутою клавішею Ctrl. При копіюванні символів з однієї бібліотеки до іншої слід не забувати, що топологічне посадкове місце при цьому не копіюється і його слід копіювати окремо.

Тепер потрібно скомпілювати проект. Для цього використаємо команду Project > Compile Integrated Library. В результаті компіляції програма виконає перевірку на відповідність виводів у УГП виводам у топологічному посадковому місці, а також створить вихідний файл формату *.IntLib, який і зберігаємо.

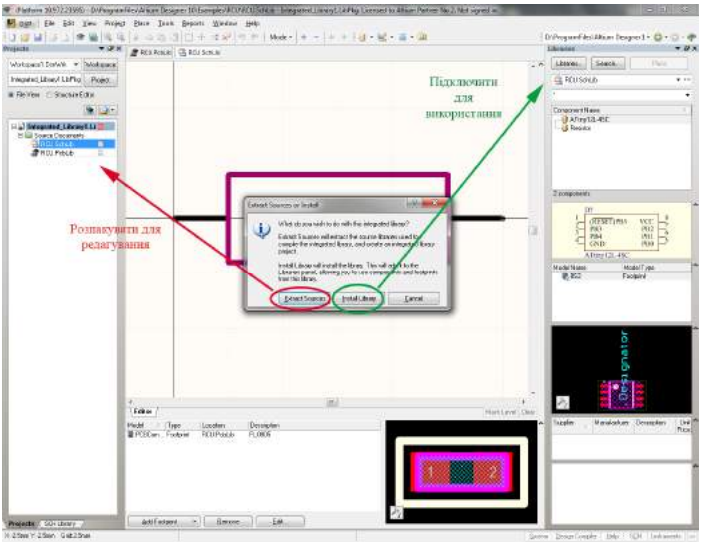


Рисунок 7.2 – Відкриття інтегрованої бібліотеки

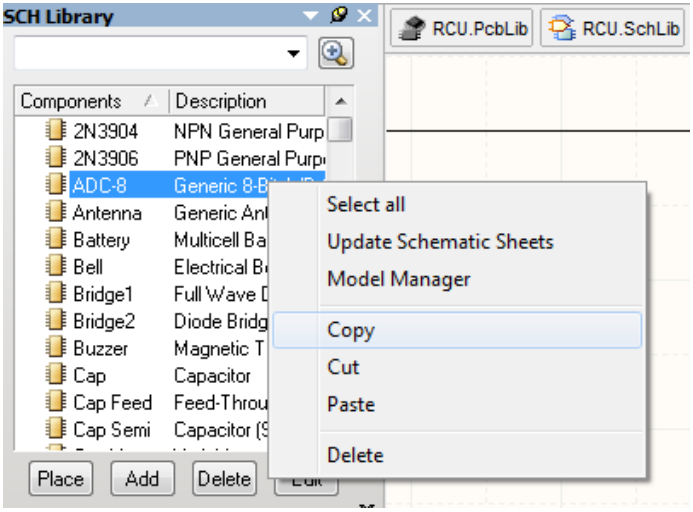


Рисунок 7.3 – Копіювання компонентів із бібліотеки

8 СТВОРЕННЯ ПЕРЕЛІКУ ЕЛЕМЕНТІВ (BILL OF MATERIALS). ДРУКУВАННЯ ДОКУМЕНТАЦІЇ

8.1 Формування документації

Результатом розробки принципової схеми, в загальному випадку повинні бути три пункти:

- інформація для розробки друкованої плати;
- креслення схеми;
- перелік елементів.

Інформація для плати закладена у самій схемі, і буде передана у файл плати на одному з перших етапів проектування плати. Два інших пункти можна отримати у Altium Designer.

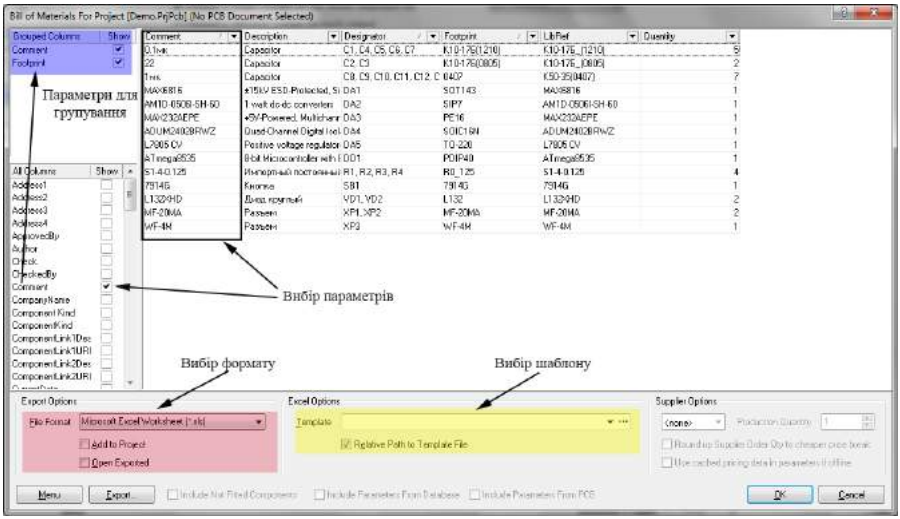


Рисунок 8.1 – Структура вікна налаштування звіту BOM

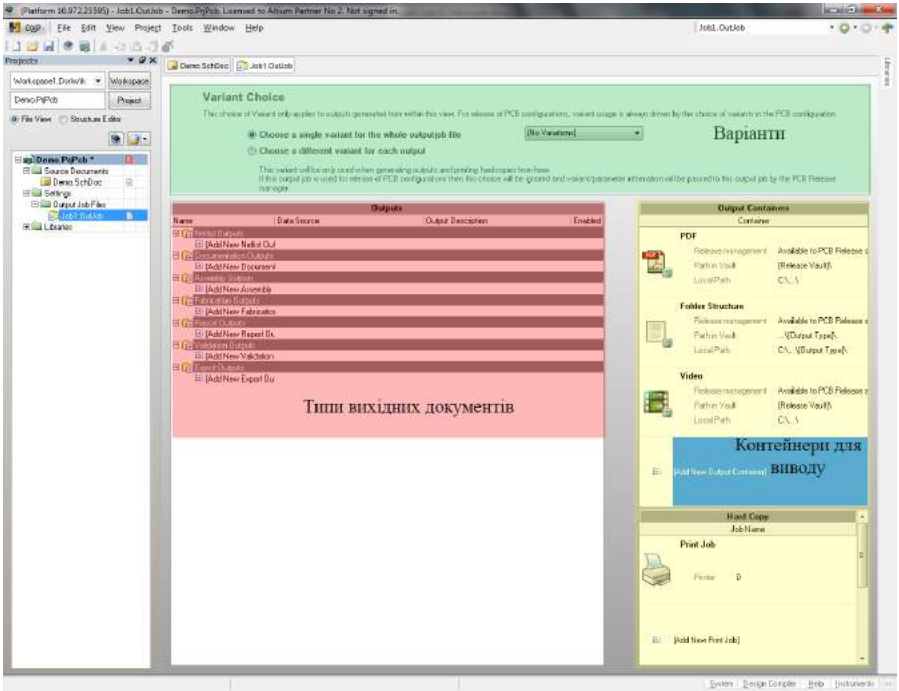


Рисунок 8.2 – Структура файлу Output Job File

8.2 Практична робота зі створення документації

8.2.1 Створення переліку елементів (Bill of Materials)

Спочатку потрібно відкрити файл проекту Demo.PrjPcb. Слід відзначити, що для створення переліку елементів потрібно мати правильно оформлені бібліотеки компонентів. У кожного компонента повинна бути введена інформація, яка описує всі його параметри, необхідні для створення переліку елементів. Щоб перевірити чи правильно заповнені параметри компонентів одразу для всього проекту можна використати команду Tools > Parameter Manager, у вікні, що з’явилося потрібно обрати параметри згідно з рис. 8.3.

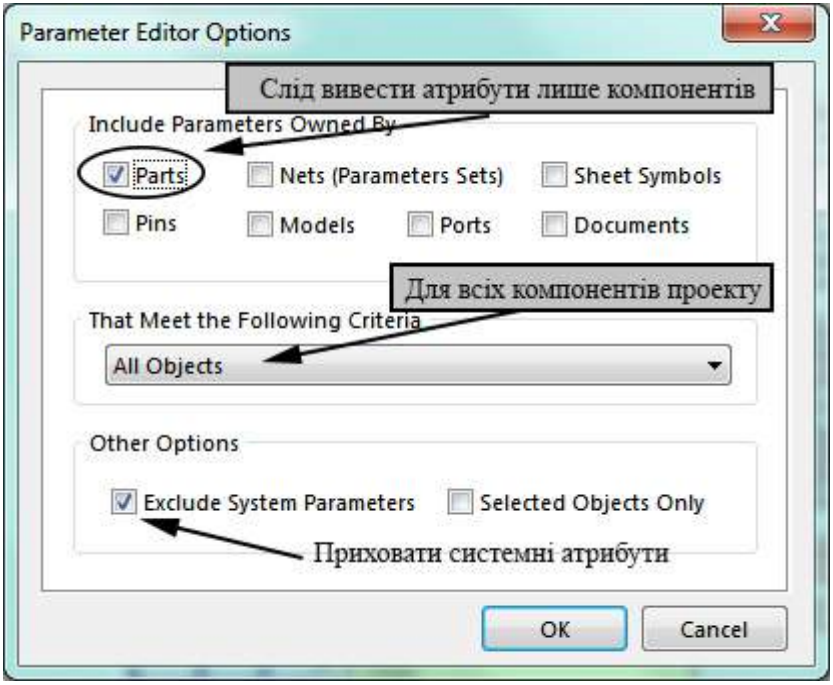


Рисунок 8.3 – Налаштування фільтру атрибутів

Потім з’явиться таблиця з параметрами компонентів, у якій можна зручно редагувати параметри одразу декількох компонентів (рис.8.4). Також потрібно перевірити поля основного надпису документа за допомогою команди Design > Document Options у вкладці Parameters.

Далі додамо файл вихідних даних для обробки в проект командою File > New > Output Job File або з контекстного меню проекту у вкладці Projects, викликаного натисканням правої клавіші миші на проекті і збережемо його в папці проекту.

Додаємо вихідний документ Report Outputs.

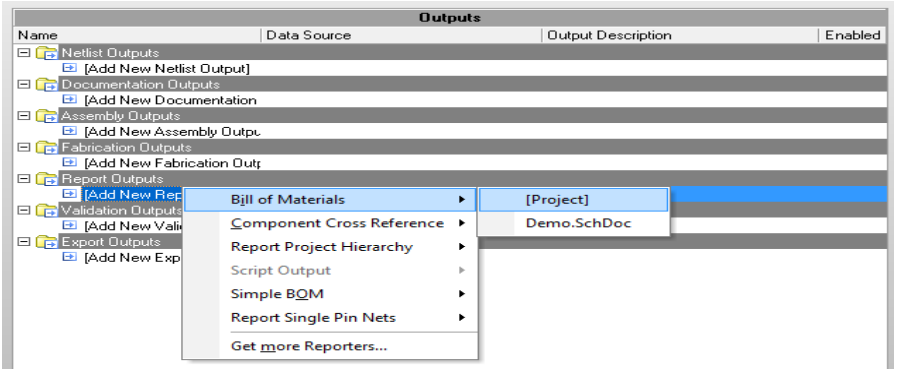


Рисунок 8.4 – Додавання переліку елементів для проекту

Після створення викликаємо вікно налаштування звіту BOM (рис.8.5). Для цього натискаємо правою клавішею миші на цьому документі та обираємо пункт Configure. У вікні налаштування обираємо потрібні параметри:

- Designator;
- Manufacturer;
- ManufacturerPartNumber;
- Power / Voltage;
- Quantity;
- Rem;
- TKE;
- Tolerance;
- Value.

По параметру Designator будемо проводити групування компонентів. Для цього перетягуємо потрібний параметр у вікно параметрів для групування. Також підключаємо шаблон Шаблон BOM_Родник.xls.

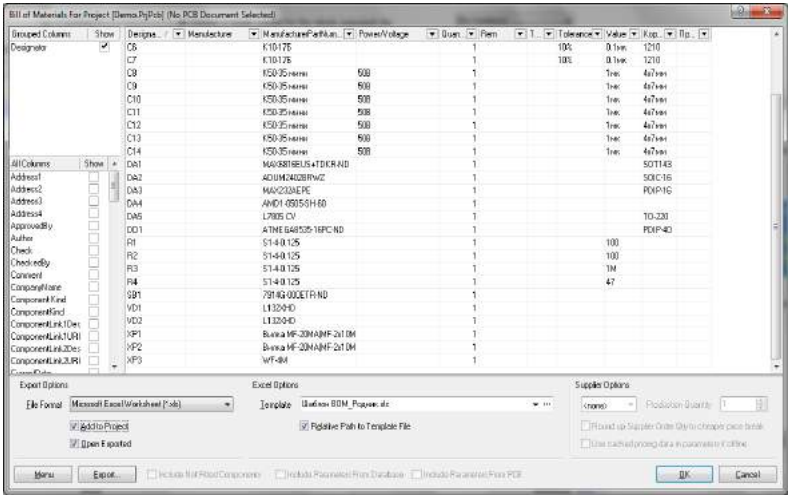


Рисунок 8.5 – Налаштування звіту BOM

Завершивши налаштування, виберемо контейнер для виводу Folder Structure та вказуємо, що звіт буде формуватися у папку (рис.8.6). Задаємо назву файлу та місце, де він буде збережений натисканням правої клавіші миші на потрібному контейнері виводу та обравши пункт Properties.

Для того, щоб перелік елементів зберігався разом з проектом у піддиректорії BOM потрібно натиснути на Release Managed і обрати Manually Managed (рис.8.7).

Для генерації переліку елементів натискаємо на клавішу Generate Content у вікні Folder Structure та зберігаємо отримані звіти.

8.2.2 Друкування принципової схеми

Відкриємо файл проекту RCU.PrjPcb та одразу додамо файл вихідних даних для обробки в проект командою File > New > Output Job File або з контекстного меню проекту у вкладці Projects, викликаного натисканням правої клавіші миші на проекті і збережемо його в папці проекту. У розділі Documentation Outputs натискаємо на клавішу Add New Document і обираємо вихідний документ Schematic Prints для всіх схем проекту (рис.8.8).

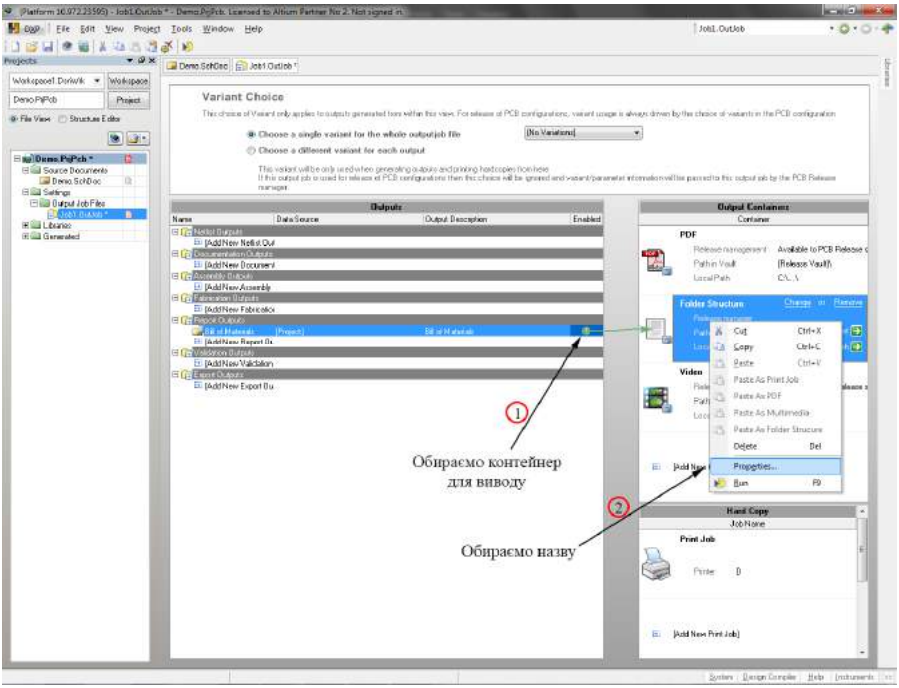


Рисунок 8.6 – Вибір потрібного контейнеру

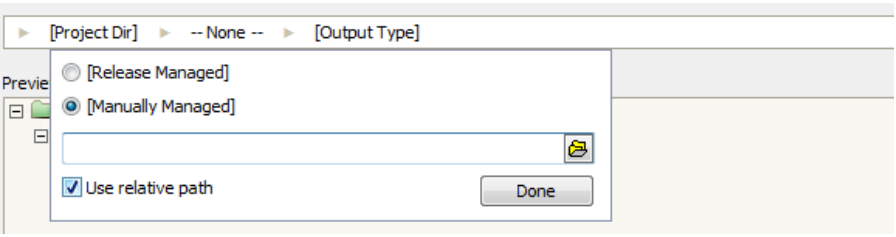


Рисунок 8.7 – Вибір місця збереження файлу

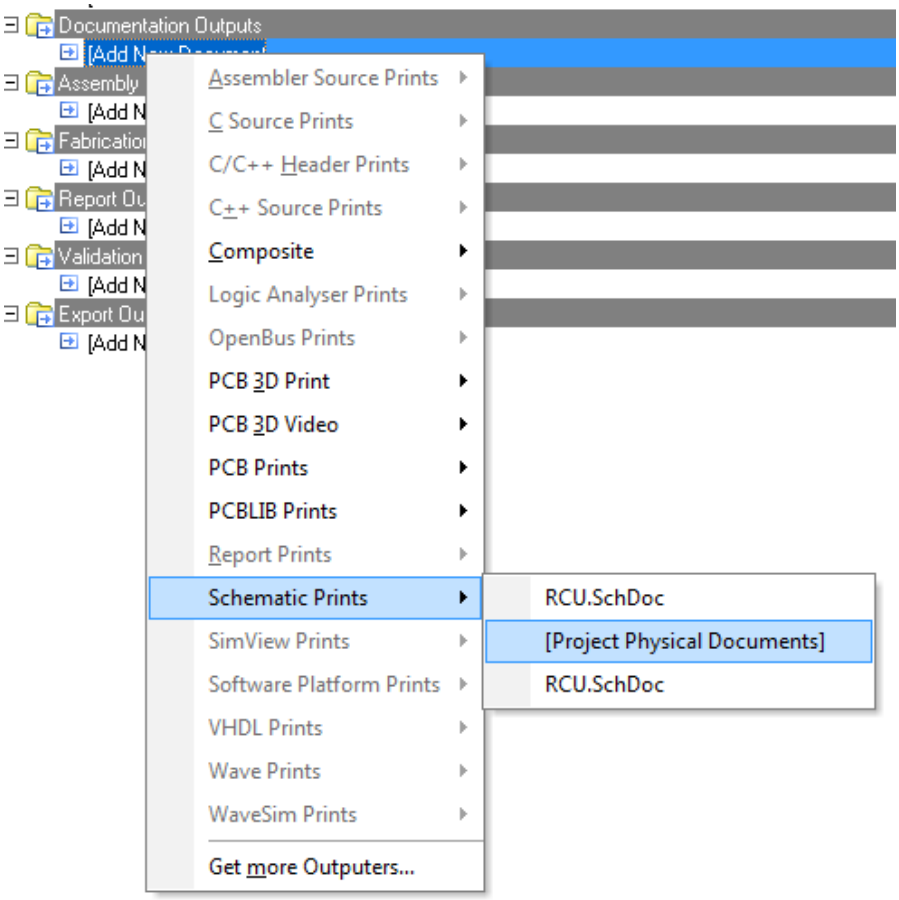


Рисунок 8.8 – Додавання вихідного документу

Далі вибераємо контейнер для виводу PDF та вказуємо, що звіт буде формуватися у PDF. Задаємо назву файлу та місце, де він буде збережений у властивостях контейнера, натиснувши на нього правою клавішею миші та обравши пункт Properties. Налаштовуємо параметри вихідного документу схеми згідно з рис. 8.9. Також налаштовуємо параметри друку, обравши пункт Page Setup. У вікні, що з'явилося обираємо розмір сторінки – A3, орієнтація сторінки – портретна, режим відображення кольорів – монохроматичний (чорно-білий).

Натискаємо на клавішу Generate Content у вікні New PDF та зберігаємо отриману схему.

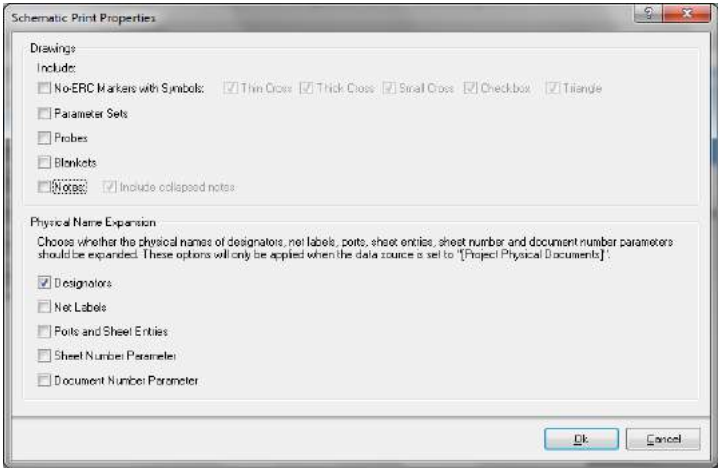


Рисунок 8.9 – Параметри документа схеми

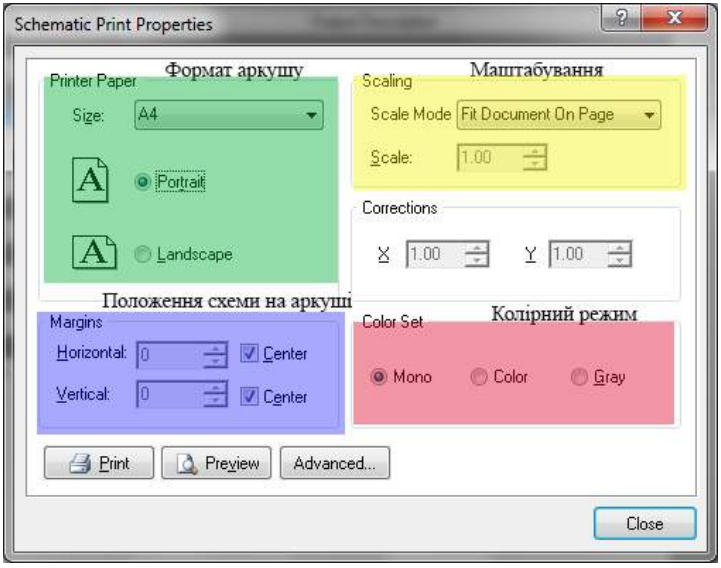


Рисунок 8.10 – Параметри друкування

8.2.3 Друк документації на друковану плату

Відкриємо файл проекту DT01.PrjPcb, який знаходиться у папці Examples (…\Examples\Developer Tool - DT01) та одразу додамо файл вихідних даних для обробки в проект командою File > New > Output Job File або з контекстного меню проекту у вкладці Projects, викликаного натисканням правої клавіші миші на проекті і збережемо його в папці проекту.

У розділі Documentation Outputs натискаємо на клавішу Add New Document і обираємо вихідний документ PCB Prints.

Налаштовуємо параметри вихідного документу (рис.8.11), видаляємо зайві та додаємо необхідні шари. Потім обираємо власну область для друкування Specific Area та натискаємо клавішу Define і задаємо потрібну область згідно з рис. 8.12.

Виберемо контейнер для виводу PDF та вказуємо, що звіт буде формуватися у PDF. Задаємо назву файлу та місце, де він буде збережений у властивостях контейнера, натиснувши на нього правою клавішею миші та обравши пункт Properties. Натискаємо на клавішу Generate Content у вікні New PDF та зберігаємо отриману схему.

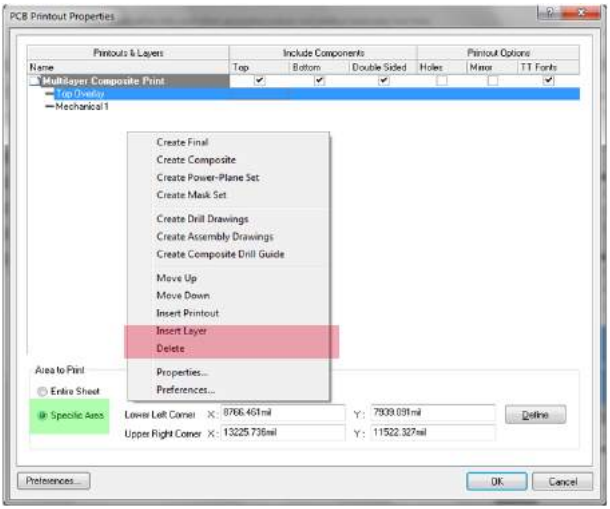


Рисунок 8.11 – Налаштування вихідного документу

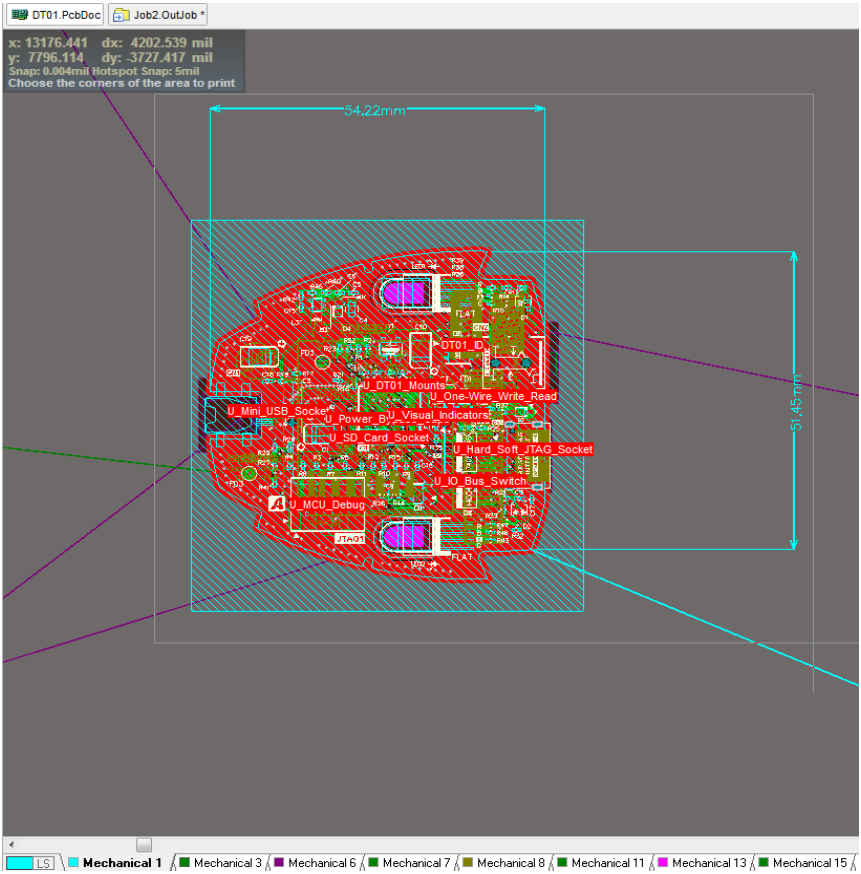


Рисунок 8.12 – Вибір області для друкування

ПИТАННЯ ДЛЯ САМОПЕРЕВІРКИ ДО ЧАСТИНИ 1

1. Що входить до складу простої ВС?
2. Назвіть особливості ВС. Чому ВС відносяться до категорії систем з переважно програмною реалізацією?
3. Які проблеми притаманні процесу проектування сучасних ВС?
4. Структура пакету програм Proteus VSM.
5. Порівняйте продукти компанії Autodesk: EAGLE, 123D Circuits та AutoCAD Electrical.
6. Призначення та функціональні можливості Altium Designer.
7. Групи елементів головного вікна програми Altium Designer.
8. Чим відрізняються панелі робочої області та панелі інструментів Altium Designer?
9. Яким чином в Altium Designer здійснюється редагування властивостей вибраного об`єкта?
10. Яка панель надає властивості виділених об'єктів в табличному вигляді?
11. Які команди малювання доступні користувачам Altium Designer?
12. Які типи проектів підтримує Altium Designer?
13. Які типів документів можливо підключити до проекту в Altium Designer?
14. Назвіть основні терміни, що використовуються при роботі з Altium Designer
15. Які види бібліотек підтримує Altium Designer?
16. Яким чином здійснюється пошук подібних об'єктів в Altium Designer та які критерії виділення можна застосовувати?
17. Які дії можливо виконати після пошуку подібних об'єктів в Altium Designer?
18. Яким чином обираються одиниці вимірювання та крок сітки?
19. Які параметри контактної площинки потрібно задати для проекту?
20. Назвіть послідовність дій зі створення бібліотечного символу (УГП).
21. Назвіть послідовність дій зі створення топологічного посадкового місця (ТПМ).
22. Як з’єднати УГП та ТПМ?
23. Яким чином можна додати файл електричної схеми в проект?
24. Яка команда відповідає за малювання з'єднань (створення електричних кіл)?
25. Як здійснюється нумерація позиційних позначень елементів схеми?
26. Як здійснюється перевірка схеми на помилки?

27. Назвіть порядок роботи із друкованою платою в Altium Designer
28. Яким чином можна переключатися між режимами 2D/3D при роботі з платою
29. Які типи 3D моделей підтримує Altium Designer
30. Які варіанти створення контуру плати можливо застосувати в Altium Designer? Що таке фізичний та віртуальний контур плати?
31. Яким чином можна додавати шари та керувати їх розташуванням для створення структури плати?
32. Які типи шарів можна створювати в Altium Designer
33. Яким чином відобразити дані схеми електричної принципової на плату?
34. Яка команда здійснює перевірку помилок при відображенні схеми електричної принципової на плату?
35. Назвіть послідовність створення правил проектування (конструктивних і технологічних обмежень). Як задається пріоритет правил.
36. Яким чином здійснюється розміщення компонентів на платі?
37. Які режими трасування провідників підримує Altium Designer?
38. Яка інформація для кіл доступна в режимі роботи із колами (Net)?
39. Як вбудувати (закріпити) реалістичну модель до посадкового місця компонента бібліотеки?
40. Послідовність створення спрощеної 3D моделі.
41. Які десять груп правил проектування доступні у редакторі друкованих плат
42. Послідовність створення правила для друкованої плати та для принципової схеми
43. Дайте характеристику модулю Mixed SIM
44. Які бібліотеки треба підключити до проекту перед моделюванням?
45. Правила створення схеми для моделювання.
46. Яким чином складається завдання для моделювання. Які графіки називаються АЧХ та ФЧХ
47. Переваги створення інтегрованої бібліотеки. Послідовність створення інтегрованої бібліотеки
48. Основні види вихідної документації.
49. Етапи створення переліку компонентів (BOM).
50. Етапи друкування документації на схему та друковану плату.

ЧАСТИНА 2. АПАРАТНО-ПРОГРАМНЕ ПРОЕКТУВАННЯ ВБУДОВАНИХ СИСТЕМ

Вступ до частини 2

З метою прискорення процесу проектування ВС сьогодні пропонуються та широко використовуються різноманітні готові платформи. Найбільш відомими є: Arduino, Raspberry Pi, Intel Galileo, Altera Cyclone та ін. Ці платформи оснащені всіма необхідними апаратними складовими, за необхідністю до них можна підключити додаткові модулі, які призначені для розширення базового функціоналу. Розглянемо деякі з них докладніше.

Raspberry Pi є повноцінним комп'ютером, але при цьому має розмір з кредитну картку. На ньому може бути встановлена операційних система (ОС) Debian, Fedora, Gentoo або ОС Raspbian, що заснована на Debian та оптимізована під Raspberry Pi. Він може виконувати безліч завдань, які виконують персональні компьютери, наприклад, виконувати функцію сервера Розумного будинку. Переваги: наявність повноцінної ОС, багатозадачність, висока швидкість роботи, велика кількість пам'яті. Недоліки: жорсткі вимоги до живлення, невеликий вихідний струм, відсутність аналого-цифрового перетворювача [32].

Intel Galileo - це Arduino-сумісна плата від Intel, заснована на x86-системі на чіпі (SoC) Intel Quark X1000 з тактовою частотою 400 МГц. Плата повністю сумісна з модулями (shields) від Arduino і підтримує, як 5-вольтові, так і 3,3-вольтові плати розширення. Стандартне оснащення плати включає в себе повнорозмірний рознім mini-PCI Express, порт Ethernet, рознім Micro-SD, шестиконтактний коннектор USB TTL UART 3,3 В, два USB порта і 8 МБ флеш-пам'яті. Плата не може живитися через USB-вхід. Переваги: наявність повноцінної ОС, висока швидкість роботи, велика кількість пам'яті. Недоліки: жорсткі вимоги до живлення, висока ціна [33].

Altera Cyclone - це ПЛІС від компанії Altera. На відміну від звичайних цифрових мікросхем, логіка роботи ПЛІС не визначається при виготовленні, а задається за допомогою програмування (проектування). Вони складаються з конфігурованих логічних блоків, подібних перемикачам з множиною входів і одним виходом (логічні вентилі або gates). У цифрових схемах такі перемикачі реалізують

базові виконавчі операції AND, NAND, OR, NOR і XOR. Основним недоліком таких систем є те, що програмування виконується на спеціальних мовах опису апаратури, таких як Verilog, VHDL, AHDL (Altera Hardware Description Languages) та ін. [34].

Arduino – це відкрита програмована апаратна платформа для роботи з різними фізичними об'єктами, що являє собою просту плату з мікроконтролером, а також спеціальне середовище розробки для написання програмного забезпечення мікроконтролера [35-37]. Проекти пристроїв, засновані на Arduino, можуть працювати самостійно, або ж взаємодіяти з програмним забезпеченням на комп'ютері (Flash, Processing, MaxMSP). Плати можуть бути зібрані користувачем самостійно або придбані готові. Програмне забезпечення доступне для безкоштовного скачування. Існує декілька версій платформ Arduino:

- Due – нова плата на базі ARM мікропроцесора 32bit Cortex-M3 ARM SAM3U4E;
- Leonardo – остання версія платформи Arduno на Atmega32u4 микроконтролері. Відрізняється рознімом microUSB, за розмірами збігається з UNO;
- Yun – нова плата з вбудованою підтримкою WiFi на базі Atmega32u4 та Atheros AR9331;
- Micro – нове компактне рішення на базі Atmega32u4;
- Uno – найпопулярніша версія базової платформи Arduino USB. Uno має стандартний порт USB. Arduino Uno багато в чому схожа з Duemilanove, але має новий чіп ATMega8U2 для послідовного підключення по USB і нове, більш зручне маркування входів/виходів. Платформа може бути доповнена платами розширення;
- Arduino Ethernet – контролер з вбудованою підтримкою роботи по мережі і з опціональною можливістю живлення по мережі за допомогою модуля POE (Power over Ethernet);
- Duemilanove – є передостанньою версією базової платформи Arduino USB. Підключення Duemilanove здійснюється стандартним кабелем USB. Після підключення вона готова до використання. Платформа може бути доповнена платами розширення;
- Diecimila – попередня версія базової платформи Arduino USB;
- Nano – це компактна платформа, що підключається до комп'ютера за допомогою кабелю USB Mini-B;

- Mega ADK – версія плати Mega 2560 з підтримкою USB host інтерфейсу для зв'язку з телефонами на Android і іншими пристроями з USB інтерфейсом;
- Mega2560 – нова версія плати серії Mega. Побудована на базі Atmega2560 з використанням чіпа ATMega8U2 для послідовного з'єднання по USB порту;
- Mega – попередня версія серії Mega на базі Atmega1280;
- Arduino BT - платформа з модулем Bluetooth для бездротового зв'язку і програмування. Сумісна з платами розширення Arduino;
- LilyPad- платформа, що розроблена для проектів, пов'язаних з одягом, іграшками, оскільки може зашиватися в тканину;
- Fio – платформа, що розроблена для бездротових застосувань. Fio містить рознім для радіо Xbee, рознім для батареї LiPo і вбудовану схему підзарядки;
- Mini – найменша платформа Arduino. Прекрасно працює як макетна модель, або, в проектах, де простір є критичним параметром. Платформа підключається до комп'ютера за допомогою адаптера Mini USB;
- Адаптер Mini USB – плата, яка конвертує підключення USB в лінії 5 В, GND, TX і RX для з'єднання з платформою Arduino Mini або іншими микроконтроллерами;
- Pro – платформа, розроблена для досвідчених користувачів, може бути частиною великого проекту. Вона дешевше, ніж Diecimila і може живитися від акумуляторної батареї, але в той же час вимагає додаткового складання і компонентів;
- Pro Mini – як і платформа Pro розроблена для досвідчених користувачів, яким потрібна низька ціна, менші розміри і додаткова функціональність;
- Serial – базова платформа з інтерфейсом RS232 для зв'язку і програмування. Плата легко збирається навіть початківцями;
- Serial Single Sided – платформа розроблена для ручного складання. Має трохи більший розмір, ніж Diecimila, але сумісна з платами розширення Arduino;
- USB Serial Light Адаптер – адаптер, що дозволяє підключати плати Arduino до комп'ютера для обміну даними і заливки скетчів. Зручний для програмування таких плат, як Arduino Mini, Arduino Ethernet і інших, які не мають свого розніму USB;

Arduino на відміну від інших розглянутих систем надає ряд переваг:

- низька вартість - у порівнянні зі схожими апаратними платформами, плати Arduino мають відносно низьку вартість (готові модулі Arduino коштують не дорожче 50 $), а можливість зібрати плату вручну дозволяє максимально заощадити кошти і отримати Arduino за мінімальну ціну;
- кросплатформеність - програмне забезпечення Arduino працює на операційних системах Windows, Macintosh OSX і Linux, в той час, як більшість подібних систем орієнтовані на роботу тільки в Windows;
- просте і зручне середовище програмування - середовище програмування Arduino зрозуміле і просте для початківців, але при цьому досить гнучке для просунутих користувачів. Воно засноване на середовищі програмування Processing, використання якого дозволяє створити графічний інтерфейс для взаємодії користувача з Arduino;
- розширюване програмне забезпечення з відкритим вихідним кодом - завдяки цьому досвідчені програмісти можуть змінювати і доповнювати його. Можливості мови Arduino можна також розширювати за допомогою C++ бібліотек. Оскільки мова Arduino заснована на мові AVR-C, просунуті користувачі, що бажають розібратися в технічних деталях, можуть легко перейти з мови Arduino на C або вставляти частини AVR-C коду безпосередньо в програми Arduino;
- розширюване відкрите апаратне забезпечення - пристрої Arduino побудовані на базі мікроконтролерів Atmel ATmega8 і Atmega168. Завдяки тому, що всі схеми модулів Arduino опубліковані під ліцензією Creative Commons, досвідчені інженери і розробники можуть створювати свої версії пристроїв на основі існуючих. І навіть звичайні користувачі можуть збирати дослідні зразки Arduino для кращого розуміння принципів їх роботи і економії коштів.

В даній частині будуть наведені теоретичні відомості та практичні приклади створення проектів ВС на основі платформи Arduino, а також різноманітних датчиків та виконавчих механізмів. При цьому будуть використані різноманітні середовища розробки - Proteus VSM, 123D Circuits, Processing, Arduino IDE, Atmel Studio.

9 ВІРТУАЛЬНЕ ПРОЕКТУВАННЯ ВБУДОВАНИХ СИСТЕМ В СЕРЕДОВИЩІ PROTEUS VSM

9.1 Світлодіод та його використання

Світлодіод, або світловипромінюючий діод (LED — Light Emitting Diode) - напівпровідниковий прилад, що випромінює некогерентне світло при пропусканні через нього електричного струму (рис.9.1). Робота його заснована на фізичному явищі виникнення світлового випромінювання при проходженні електричного струму через p-n-перехід. Колір світіння (довжина хвилі максимуму спектра випромінювання) визначається типом використовуваних напівпровідникових матеріалів, що утворюють p-n-перехід.

Рисунок 9.1 – Світлодіоди та індикатор на їх основі

Переваги:

- світлодіоди не мають скляних колб і ниток розжарювання, що забезпечує високу механічну міцність і надійність (ударна і вібраційна стійкість);
- відсутність розігріву та високих напруг гарантує високий рівень електро- і пожежобезпеки;
- безінерційність робить світлодіоди незамінними, коли потрібна висока швидкодія;
- мініатюрність;
- тривалий термін служби (довговічність);
- високий коефіцієнт корисної дії (ККД);
- відносно низькі напруги живлення та споживані струми, низьке енергоспоживання;
- велика кількість різних кольорів світіння, спрямованість випромінювання;

- регульована інтенсивність світіння.

Недоліки:

- відносно висока вартість;
- малий світловий потік від одного елементу;
- деградація параметрів світлодіодів з часом;
- підвищені вимоги до живлячого джерела.

У світлодіодів є декілька основних параметрів:

- тип корпуса;
- типовий (робочий) струм;
- зменшення (робоче) напруги;
- колір свічення (довжина хвилі, нм);
- кут розсіювання.

Загалом, під типом корпуса розуміють діаметр та колір колби (лінзи). Як відомо, світлодіод - напівпровідниковий прилад, який необхідно живити струмом. Струм, яким слід живити той чи інший світлодіод називається типовим. При цьому, на світлодіоді падає певна напруга. Колір випромінювання визначається як використовуваними напівпровідниковими матеріалами, так і легуючими домішками. Найважливішими елементами, використовуваними в світлодіодах, є: Алюміній (Al), Галій (Ga), Індій (In), Фосфор (P), що викликають свічення в діапазоні від червоного до жовтого кольору. Індій (In), Галій (Ga), Азот (N) використовують для отримання блакитного і зеленого світіння. Крім того, якщо до кристала, що викликає блакитне (синє) світіння, додати люмінофор, то отримаємо білий колір світлодіода. Кут випромінювання визначається характеристиками матеріалів, а також колбою (лінзою) світлодіода.

В даний час світлодіоди застосовуються в самих різних областях: світлодіодні ліхтарі, автомобільна світлотехніка, рекламні вивіски, світлодіодні панелі і індикатори, світлофори і т.п.

Оскільки світлодіод є напівпровідниковим приладом, то при включенні його в електричне коло необхідно дотримуватись полярності (рис.9.2). Світлодіод має два виводи, один з яких катод ("мінус"), а інший - анод ("плюс").

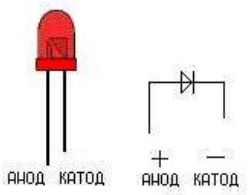


Рисунок 9.2 – Виводи світлодіода

Питання. Маємо один світлодіод, як його підключити правильно у найпростішому випадку?

Щоб правильно підключити світлодіод у найпростішому випадку, необхідно підключити його через струмообмежувальний резистор (рис.9.3).

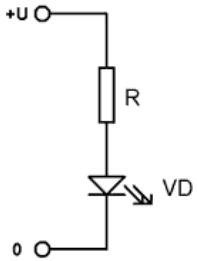


Рисунок 9.3 – Схема підключення світлодіода

Задача 1

Є блакитний світлодіод з робочою напругою 3В і робочим струмом 20 мА. Необхідно підключити його до джерела з напругою 5В. Визначити опір струмообмежуючого резистора (рис.9.4).

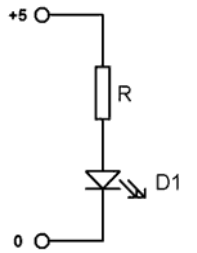


Рисунок 9.4 – Схема для Задачі 1.

У кожного світлодіода існує пряме падіння напруги (робоча напруга) при робочому стумі. Для світлодіодів одного кольору воно приблизно однакове: для червоних 1,8…2В, жовтих і зелених 2…2,4В, синіх і білих 3…3,6В. Ці дані є приблизними, для точного визначення краще дивитися у даташит (datashit) конкретного світлодіоду.

Розрахунок опору струмообмежувального резистора виконується за формулою:

$R = U_{гасяча} / I_{світлодіода}$

$U_{гасяча} = U_{живлення} – U_{світлодіода}$,

де $I_{світлодіода}$ – робочий струм, А;

$U_{світлодіода}$ – робоча напруга, В;

$U_{живлення}$ – напруга живлення, В;

$U_{гасяча}$ –кількість напруги, що гаситься на світлодіоді, В;

R – опір струмообмежуючого резистора, Ом.

Питання. Як підключити декілька світлодіодів?

Декілька світлодіодів підключаємо послідовно чи паралельно, розраховуючи необхідні опори.

Задача 2.

Є блакитні світлодіоди з робочою напругою 3В і робочим струмом 20 мА. Треба підключити 3 світлодіоди до джерела 15В (рис.9.5).

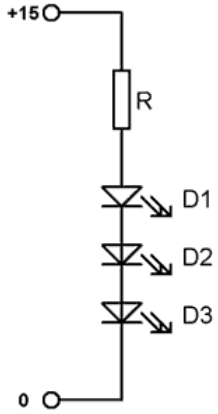


Рисунок 9.5 – Схема для Задачі 2.

Розрахунок аналогічний попередній задачі, але з урахуванням кількості світлодіодів:

$U_{гасяча} = U_{живлення} – N * U_{світлодіода}$,

де N – кількість світлодіодів.

Задача 3.

Нехай є 2 червоні світлодіоди з робочою напругою 1,8 В і робочим струмом 20 мА, та 2 жовті світлодіоди з робочою напругою 2,2 В і робочим струмом 20 мА. Треба підключити 4 світлодіоди до джерела 7 В (рис.9.6).

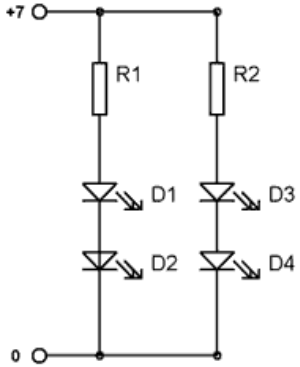


Рисунок 9.6 – Схема для Задачі 3.

9.2 Середовище віртуального проектування електронних схем Proteus VSM

PROTEUS VSM – середовище для проектування та симуляції роботи електронних схем. Відмінністю пакета Proteus VSM є можливість моделювання роботи програмованих пристроїв: мікроконтролерів (МК), мікропроцесорів, DSP (Digital signal processor) та ін..

PROTEUS VSM дозволяє достовірно моделювати і налагоджувати досить складні пристрої, в яких може міститися кілька МК одночасно і навіть різних родин в одному пристрої.

На рисунку 9.7 зображене головне вікно програми. Увесь робочий простір програми розподілено на декілька основних частин:

- вікно редактору схем (тут виконується синтез з окремих компонентів);
- вікно вибору об’єктів (доступні різні елементи в залежності від вибраного режиму);
- панель керування симуляцією (знаходиться у лівому нижньому куті, включає в себе такі команди: Пуск; Виконання одного такту, що включає симуляцію на час Single Step Time, який задається у розділі головного меню System>Set Animation Options; Пауза; Стоп).

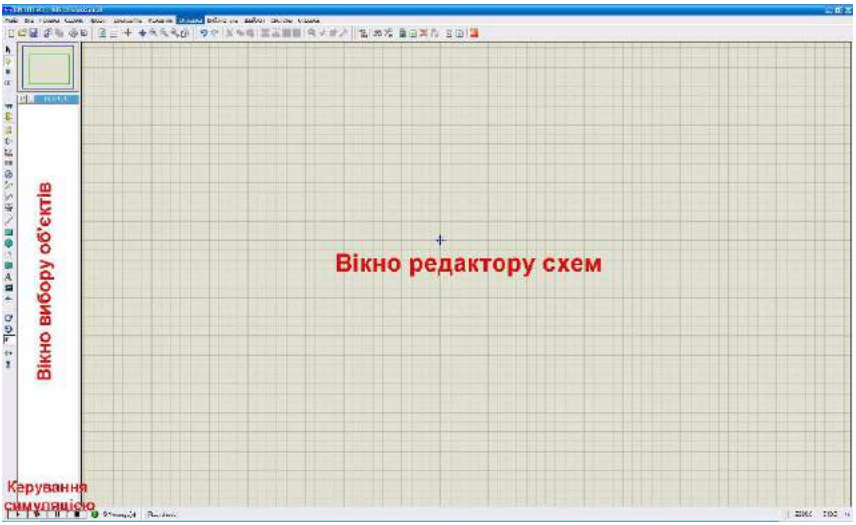
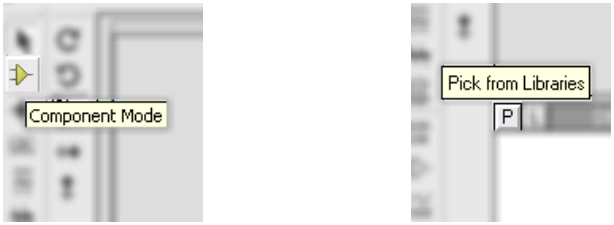


Рисунок 9.7 – Головне вікно Proteus VSM (ISIS)

9.3 Проектування віртуальної моделі електронної схеми та симуляція її роботи в Proteus VSM

Для того, щоб зібрати схему будь-якого пристрою, необхідно підготувати набір елементів, з яких буде складатися ця схема. Для цього переходимо в режим компонентів або Component Mode (рис. 9.8а) і клікаєм клавішу P, яка знаходиться під вікном перегляду поряд з клавішею L (рис 9.8б).



а) б)

Рисунок 9.8 – Панель інструментів

Перед нами з’являється менеджер компонентів, який надає на наш вибір всі елементи, які містяться в бібліотеці програми (рис 9.9).

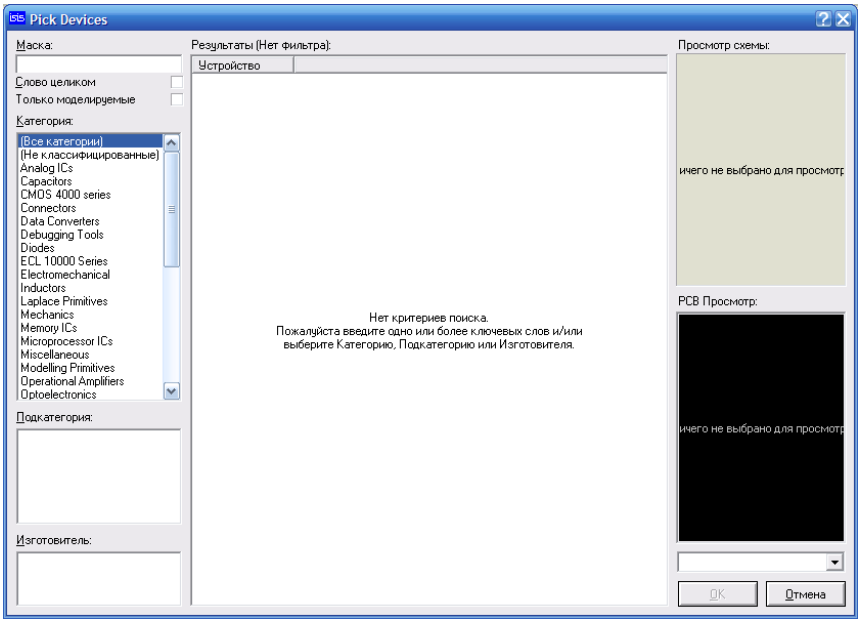


Рисунок 9.9 – Вікно менеджера компонентів

Треба користуватися рядком пошуку, який знаходиться у верхньому лівому куті. Коли потрібний компонент знайдений, подвійним кліком лівої кнопки миші (ЛКМ) по його назві додамо його до переліку використовуваних компонентів. Наприклад, створимо електронну схему, яка буде складатися з: світлодіода, резистора і

блоку живлення (батареї). У рядку пошуку (маска) вводимо перший елемент, який будемо додавати до схеми: led-green. У списку елементів з'являється світлодіод з такою назвою (рис.9.10), клікаємо по ньому подвійним кліком лівою кнопкою миші.

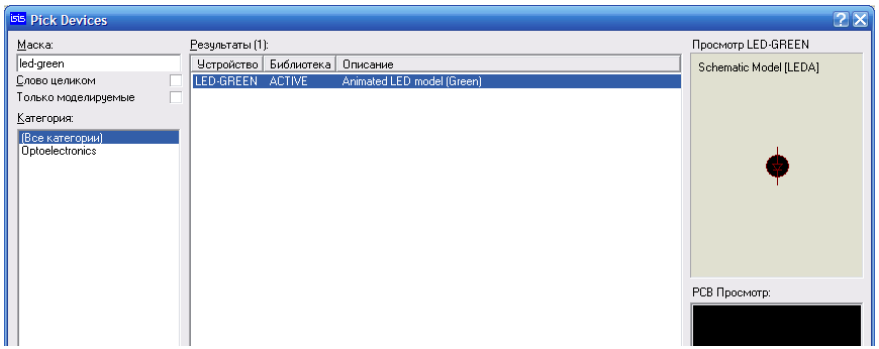


Рисунок 9.10 – Пошук компонентів

Далі так само виконуємо пошук за такими назвами: resistor, battery (бібліотека - Active), та по черзі подвійним кліком по цих елементах у списку додаємо їх до нашого проекту. Після того як ми вибрали всі компоненти, натискаємо на кнопку ОК. Тепер наше вікно вибору об'єктів буде містити всі чотири компоненти, які ми вибрали і виглядатиме як зображено на рисунку 9.11.

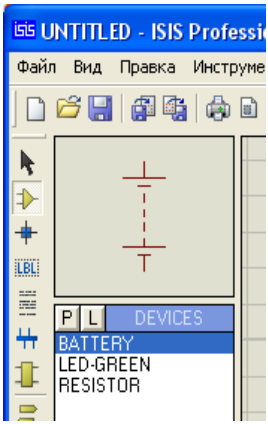
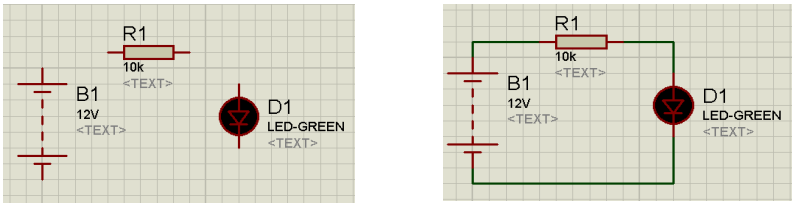


Рисунок 9.11 – Вікно вибору об'єктів

Для встановлення компоненту у вікні редактора схеми його необхідно вибрати зі списку і подвійним кліком лівої кнопки миші встановити у потрібному місці. До того, як встановити компонент на схемі, його можна попередньо розвернути у потрібне положення, яке можна контролювати у вікні перегляду.

Встановіть всі компоненти, як зображено на рисунку 9.12а, та з'єднайте їх, як на рис 9.12б. Для того, щоб з'єднати два компоненти між собою для початку виберіть інструмент Стрілка, потім на самій схемі наведіть курсор миші на кінець елемента, повинен з'явитися квадратний червоний контур, натискаємо і відпускаємо ЛКМ, і ведемо курсор до контакту іншого елементу, і на кінці іншого елемента також натискаємо ЛКМ, щоб завершити побудову з'єднання між двома елементами.



а) б)

Рисунок 9.12 – Розміщення та з'єднання компонентів електричної схеми

Після того, як електричне коло створено, міняємо характеристики наших компонентів. Зараз батарея живить схему 12В, а нам потрібно змінити живлення схеми на 3В. Для цього клікаємо двійним кліком ЛКМ на батареї і у полі Voltage встановлюємо 3V (рис. 9.13).

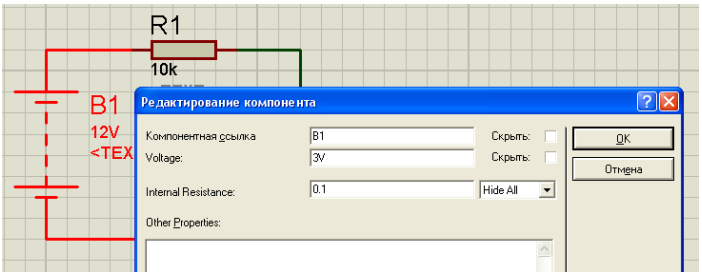


Рисунок 9.13 – Зміна параметрів живлення

Так само змінюємо параметри світлодіода. У нашому випадку маємо зелений світлодіод, тож встановлюємо у вікні параметрів світлодіода падіння напруги 2В та робочий струм 20мА (рис.9.14).

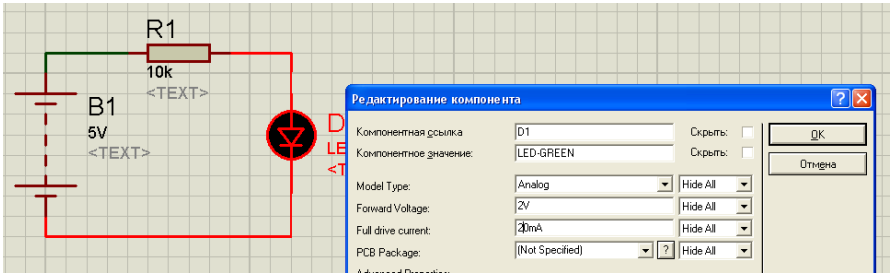


Рисунок 9.14 – Змінення параметрів опору

Далі виконуємо розрахунок опору, потрібного для нормальної роботи світлодіода. Для цього розраховуємо, скільки напруги гаситиметься на нашому світлодіоді: $U_{гасяча}$=3В-2В=1В. Згідно закону Ома, необхідний опір резистора: R=U/I=1В/0,02А=50Ом. У симуляторі ми можемо вказати величину опору 50Ом, але для реальної схеми виберемо резистор з стандартного ряду з опором 51Ом (рис.9.15).

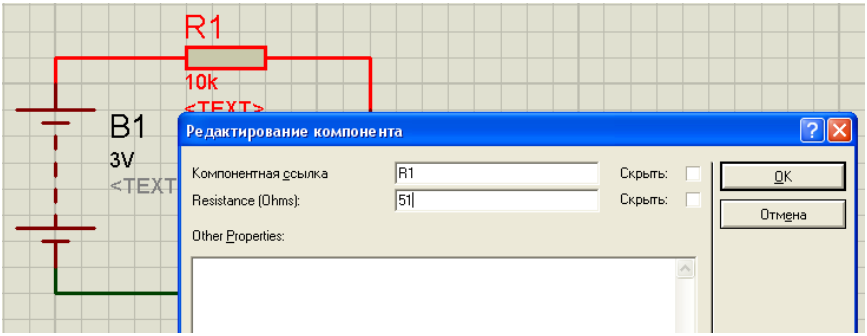


Рисунок 9.15 – Змінення параметрів опору

Далі, натискаємо в панелі керування симуляцією кнопку «Відтворити» (рис.9.16).

Рисунок 9.16 – Панель керування симуляцією

Тепер наша схема може віртуально відтворити включення світлодіоду (рис 9.17).

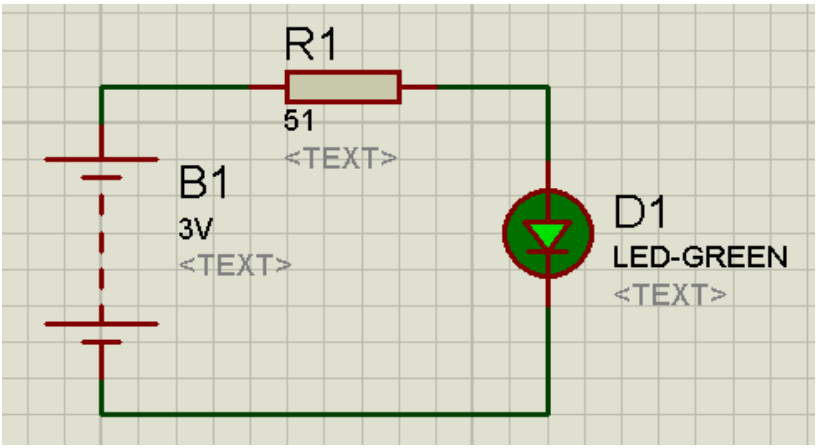


Рисунок 9.17 – Симуляція роботи світлодіода

9.4 Практична робота

Завдання 1. Виконати приклад, наведений у підрозділі 9.3.

Завдання 2. Розв’язати три задачі, які наводяться у підрозділі 9.1. Спроектувати віртуальні моделі всіх електронних схем та виконати симуляцію їх роботи в Proteus VSM.

10 ПРИНЦИПИ ВИМІРЮВАННЯ В ЕЛЕКТРОНІЦІ. РОБОТА З КНОПКОЮ ТА ЗМІННИМ РЕЗИСТОРОМ

10.1 Електровимірювальні прилади в середовищі Proteus

Для того, щоб переглянути, які вимірювальні прилади наявні в середовищі Proteus VSM, необхідно вибрати «Віртуальні інструменти» (рис. 10.1).

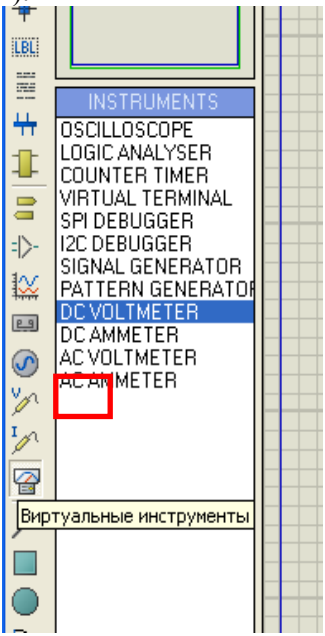


Рисунок 10.1- Електровимірювальні прилади в середовищі Proteus VSM

Розглянемо роботу таких приладів як: DC Voltmeter (вольтметр постійного струму), DC Ammeter (амперметр), Oscilloscope (осцилограф). AC вказує, що прилад використовується в електричних колах змінного струму.

Створимо електричну схему, як зображено на рисунку 10.2.

Щоб знайти елемент кнопка у бібліотеці Proteus, введіть push button.

10.2 Схеми з використанням кнопки

Кнопка – найпростіший електромеханічний пристрій для передачі електричного сигналу різним пристроям шляхом замикання або розмикання двох і більше контактів. В залежності від стану, в якому залишається кнопка після натискання на неї, вона буває двох видів:

- кнопки (перемикачі) з фіксованим положенням;
- тактові кнопки або нефіксовані, які повертаються у вихідне положення.

Встановіть живлення батареї 5В, показники зеленого світлодіода залиште незмінними та розрахуйте необхідне значення опору резистора.

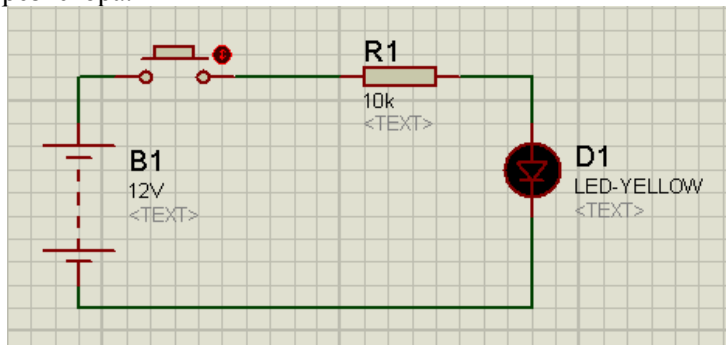
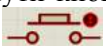


Рисунок 10.2 – Побудова електричної схеми

Після цього, натискаємо в панелі керування симуляцією кнопку «Відтворити».

Рисунок 10.3 – Панель керування симуляцією

Зараз наша схема може віртуально відтворити включення та виключення світлодіоду при натисканні на кнопку. Для цього просто наведіть курсор миші на елемент Кнопка у схемі і натисніть ЛКМ. Для

того, щоб затиснути кнопку, клікніть ЛКМ на коло з правої сторони елементу кнопка 

Для схеми на рис. 9.17 можна додати кнопку та підключіти DC Voltmeter, DC Ammeter, як зображено на рисунку 10.5. Ці вимірювальні прилади показують нам значення струму в електричному колі та значення напруги на світлодіоді, відповідно.

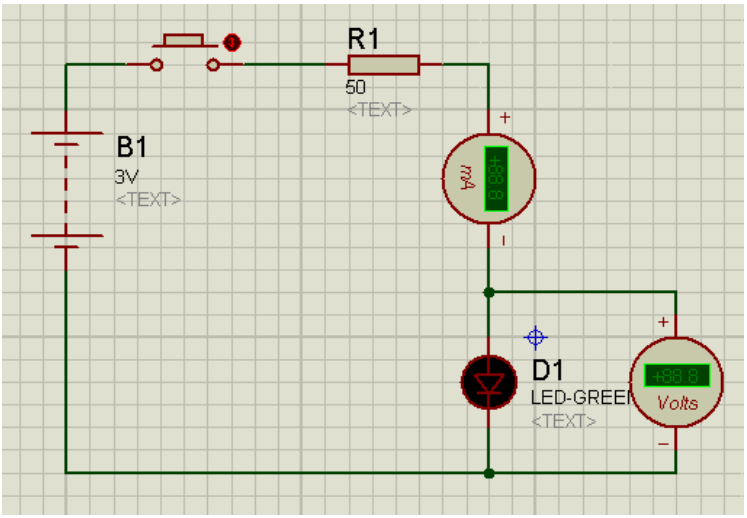


Рисунок 10.5 – Схема підключення вольтметра і амперметра

Щоб змінити відображення значень амперметра з Амперів на міліАмпери або мікроАмпери, у властивостях амперметра у полі Display Range встановіть необхідне значення (рис.10.6).

10.3 Практична робота з електровимірювальними приладами

Завдання 1 Спроектуйте електричну схему, яку зображено на рисунку 10.7. Виконайте 3 сценарії використання змінного резистору (мінімальний опір, середнє значення, максимальне) Змінний резистор або потенціометр у Proteus називається POT-HG.

Завдання 2. Спроектуйте схеми, представлені на рис.10.8-10.10 та визначте:

а) опір резисторів R1, R2;
б) опір R1 та силу струму (I);
в) опір R2 та напругу батареї (U).
Завдання 3. Розрахуйте силу струму для схеми на рисунку 10.5.

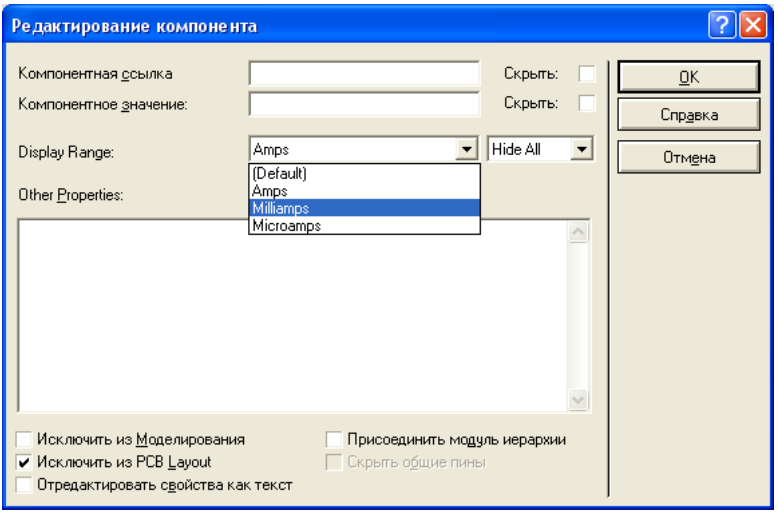


Рисунок 10.6 – Вікно властивостей амперметра

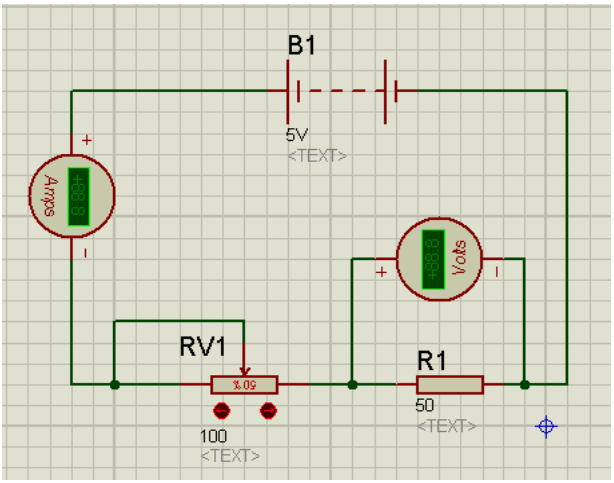


Рисунок 10.7 – Схема для Завдання 1.

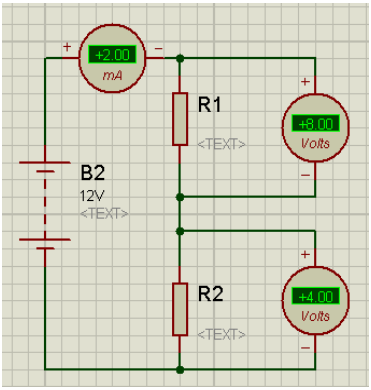


Рисунок 10.8 – Схема для Завдання 2а.

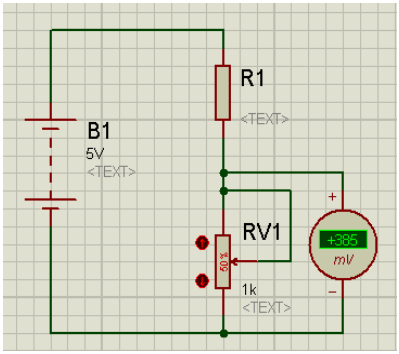


Рисунок 10.9 – Схема для Завдання 2б.

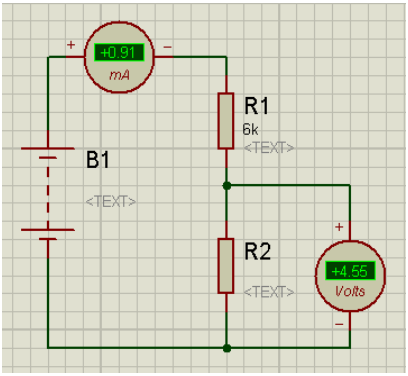


Рисунок 10.10 – Схема для Завдання 2в.

11 ПОЧАТОК РОБОТИ З ARDUINO. СИМУЛЯЦІЯ РОБОТИ ARDUINO

11.1 Arduino Uno

Пристрої на базі Arduino можуть отримувати інформацію про навколишнє середовище за допомогою різних датчиків, а також можуть управляти різними виконавчими пристроями (рис.11.1).

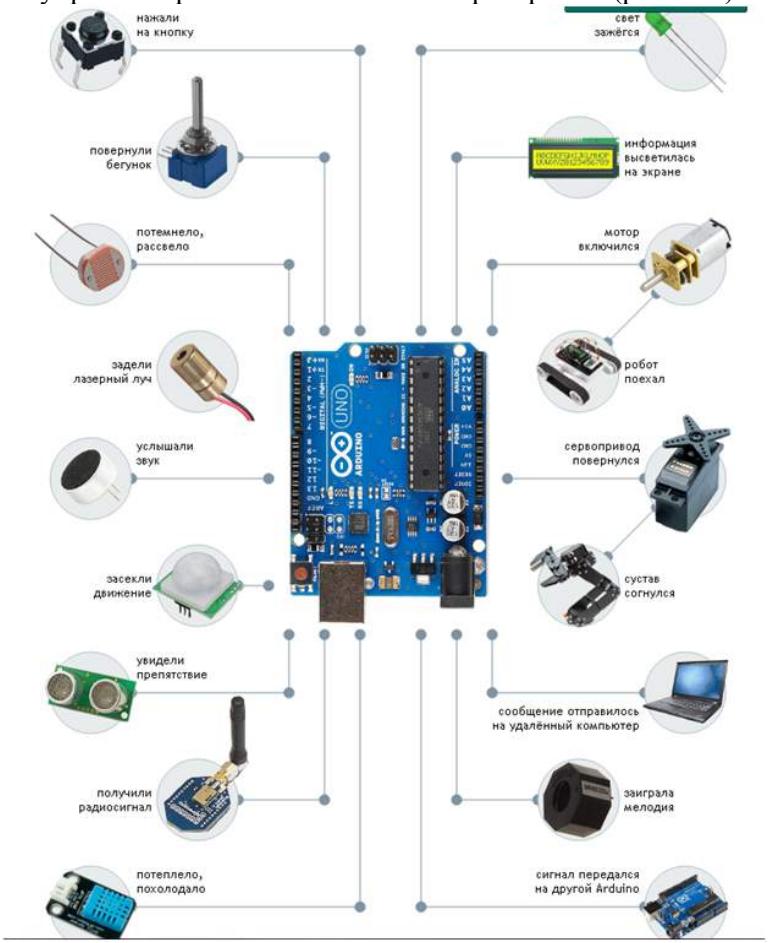


Рисунок 11.1 – Взаємодія Arduino з датчиками та виконавчими пристроями

Arduino Uno базується на мікроконтролері Atmel ATmega328 (рис. 11.2). Мікроконтролер на платі програмується за допомогою мови Arduino (заснований на мові Wiring) і середовища розробки Arduino (засноване на середовищі Processing).

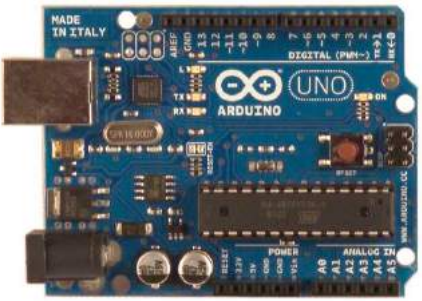


Рисунок 11.2 – Зовнішній вигляд Arduino Uno

Платформа має 14 цифрових входів/виходів (6 з яких можуть використовуватися як виходи широтно-імпульсної модуляції (ШІМ)), 6 аналогових входів, кварцовий генератор 16 МГц, рознім USB, силовий рознім, рознім ICSP і кнопку перезавантаження. Кожен з 14 цифрових виводів Uno може налаштований як вхід або вихід, використовуючи функції pinMode (), digitalWrite (), та digitalRead (). Кожен вивід має навантажувальний резистор (за замовчуванням відключений) 20-50 кОм і може пропускати до 40 мА. Для роботи необхідно підключити платформу до комп'ютера за допомогою кабелю USB, або подати живлення за допомогою адаптера AC/DC або батареї. Платформа може працювати при зовнішньому живленні від 6В до 20В. При напрузі живлення нижче 7В, вивід 5V може видавати менше 5В, при цьому платформа може працювати нестабільно. При використанні напруги вище 12В регулятор напруги може перегрітися і пошкодити плату. Рекомендований діапазон напруги від 7В до 12В.

Деякі виводи мають особливі функції:

- VIN - вхід використовується для подачі живлення від зовнішнього джерела (за відсутності 5В від розніму USB або іншого регульованого джерела живлення);
- 5V - регульоване джерело напруги, що використовується для живлення мікроконтролера і компонентів на платі. Живлення може

подаватися від виводу VIN через регулятор напруги, або від розніму USB, або іншого регульованого джерела напруги 5В;

- 3V3 - напруга на виводі 3.3В генерується вбудованим регулятором на платі. Максимальний споживаний струм 50мА;
- GND - виводи заземлення;
- послідовна шина: 0 (RX) і 1 (TX) - виводи використовуються для отримання (RX) і передачі (TX) даних TTL. Дані виводи підключені до відповідних рознімів мікросхеми послідовної шини ATmega8U2 USB-to-TTL;
- зовнішнє переривання: 2 і 3 - дані виводи можуть бути налаштовані на виклик переривання або на молодшому значенні, або на передньому чи задньому фронті, або при зміні значення. Детальна інформація знаходиться в описі функції attachInterrupt ();
- ШІМ: 3, 5, 6, 9, 10, і 11 - будь-який з виводів забезпечує ШІМ з роздільною здатністю 8 біт за допомогою функції analogWrite ();
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK) - за допомогою даних виводів здійснюється зв'язок SPI, для чого використовується бібліотека SPI;
- LED: 13 - вбудований світлодіод, підключений до цифрового виводу 13. Якщо значення на виводі має високий потенціал, то світлодіод горить.

На платформі Uno встановлені 6 аналогових входів (позначених як A0 .. A5), кожен роздільною здатністю 10 біт (тобто може приймати 1024 різних значення). Стандартно виводи мають діапазон вимірювання до 5В відносно землі, проте є можливість змінити верхню межу за допомогою виводу AREF і функції analogReference (). Деякі виводи мають додаткові функції. Наприклад: I2C: 4 (SDA) і 5 (SCL) - за допомогою виводів здійснюється зв'язок I2C (TWI), для створення якого використовується бібліотека Wire.

Додаткова пара виводів платформи:

- AREF - опорна напруга для аналогових входів. Використовується з функцією analogReference ();
- Reset - низький рівень сигналу на виводі перезавантажує мікроконтролер. Звичайно застосовується для підключення кнопки перезавантаження на платі розширення, що закриває доступ до кнопки на самій платі Arduino.

11.2 Середовище Arduino IDE

Arduino IDE (Integrated Development Environment) - це безкоштовне офіційне програмне забезпечення для програмування плати. Програма написана на Java на основі проекту Processing, працює під операційними системами Windows, Mac OS X і Linux. Програмування ведеться на мові C\C++, в якості компілятора використовується AVR-GCC.

Інтерфейс середовища розробки Arduino (рис.11.3) містить наступні основні елементи: текстовий редактор для написання програмного коду, область повідомлень, текстову консоль, панель інструментів і головне меню. Дане програмне забезпечення дозволяє комп'ютеру взаємодіяти з Arduino як для передачі даних, так і для прошивання коду в контролер.

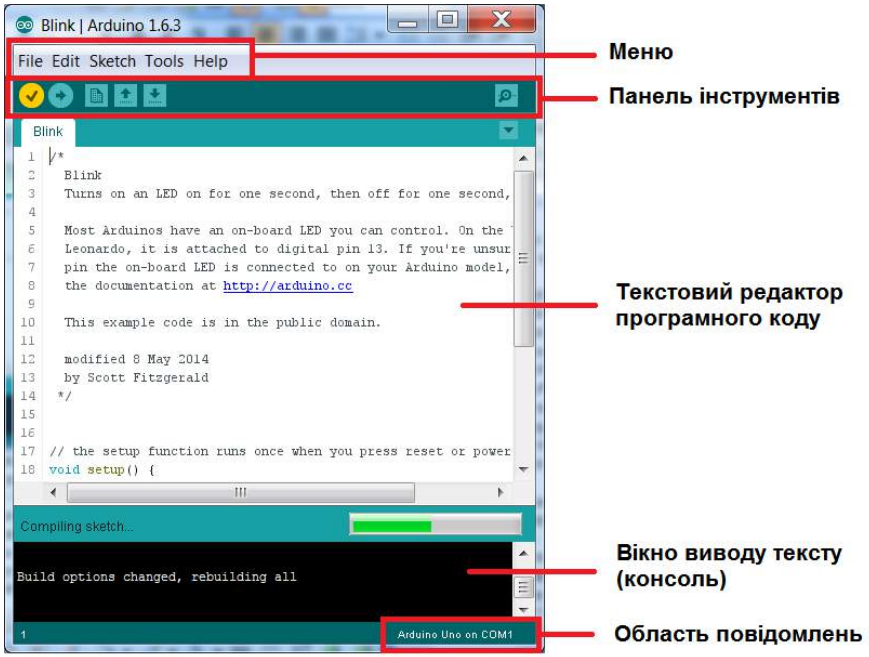


Рисунок 11.3 -Інтерфейс Arduino IDE

Програма, яка написана в середовищі Arduino, називається скетч. Скетчі створюються в текстовому редакторі та зберігаються в файлах з розширенням «.ino».

Вбудований текстовий редактор має стандартні інструменти копіювання, вставки та заміни тексту. Область повідомлень інформує користувача про події та помилки під час запису або експорту коду. Консоль відображає у вигляді тексту потік вихідних даних середовищ, що включає повні звіти про помилки та іншу інформацію.

У нижньому правому куті вікна програми показується модель поточної плати і послідовний порт, до якого вона підключена. Кнопки на панелі інструментів призначені для створення, відкриття, збереження і прошивки програм в пристрій. Окрема кнопка запускає програму SerialMonitor.

 Verify - перевірка програмного коду на помилки;

 Upload – скомпілювати програму та прошити її в МК;

 New - створення нового скетчу;

 Open - відкриття меню доступу до всіх скетчів вробочій папці;

 Save - збереження скетчу;

 Serial Monitor - відкриття програми для роботи з послідовним інтерфейсом.

Додаткові команди згруповані в п'ять меню: File, Edit, Sketch, Tools, Help. Доступність пунктів меню визначається роботою, виконуваною в даний момент.

Меню Edit (Правка):

- Copy for Forume - копіює в буфер обміну відповідний код скетчу з виділенням синтаксису для постингу на форумі;
- Copy as HTML - копіює код скетчу в буфер обміну, як HTML код для вбудовування в веб-сторінки.

Меню Sketch (Програма):

- Verify / Compile - перевірка скетчу на помилки;
- Show Sketch Folder – відкриває папку, яка містить файл поточного скетчу;

- Add File ... - додає вихідний файл в скетч (файл буде скопійований з поточного місця розташування). Новий файл з'являється в новій закладці у вікні скетчу;
- Include Library - додає бібліотеку в поточний скетч, вставляючи директиву #include в код скетчу.

Меню Tools (Інструменти):

- Auto Format - дана опція оптимізує код, наприклад, вирівнює відступи;
- Board - вибір модулі використовуваної плати;
- Serial Monitor – містить список всіх послідовних пристроїв (реальних і віртуальних), доступних в системі;
- Burn Bootloader - пункти даного меню дозволяють записати завантажувач (Bootloader) в мікроконтролер на платформі Arduino. Дана дія не потрібна в поточній роботі з Arduino, але стане в нагоді, якщо з'явився новий МК ATmega (без завантажувача). Перед записом рекомендується перевірити правильність вибору платформи з меню.

Середовищем Arduino використовується принцип Блокнота (Sketchbook): стандартне місце для зберігання програм (скетчів). Скетчі з блокнота відкриваються через меню File> Sketchbook або кнопкою Open на панелі інструментів. При першому запуску програми Arduino автоматично створюється директорія для блокнота. Розташування блокнота змінюється через діалогове вікно Preferences.

Середовище Arduino дозволяє працювати з програмами, що складаються з декількох файлів (кожен відкривається в окремій закладці). Файли коду можуть бути стандартними Arduino (без розширення), файлами C (.c.), файлами C ++ (.cpp) або заголовочними файлами (.h).

Перед завантаженням скетчу в МК, потрібно задати плату та послідовний порт в меню Tools> Board і Tools> Serial Port. В ОС Windows порти можуть позначатися як COM1 або COM2 (для пристроїв з послідовним інтерфейсом) або COM4, COM5, COM7 і вище (для плати з USB).

Після вибору порту і плати необхідно натиснути кнопку Upload на панелі інструментів або вибрати пункт меню File> Upload. Сучасні плати Arduino перезавантажуються автоматично перед завантаженням. На старих платформах необхідно натиснути кнопку перезавантаження. На більшості плат під час процесу будуть мигати

світлодіоди RX і TX. Середовище розробки Arduino виведе повідомлення про закінчення завантаження або про помилки.

При завантаженні скетчу використовується завантажувач (Bootloader) Arduino - невелика програма, що є в мікроконтролері на платі. Вона дозволяє завантажувати в МК програмний код без використання додаткових апаратних засобів. Завантажувач (Bootloader) активний протягом декількох секунд при перезавантаженні платформи і при завантаженні будь-якого з скетчів на виконання. Робота завантажувача розпізнається по мерехтінню світлодіода, підключеного до 13-го виводу контролера.

Бібліотеки додають функціональність скетчам, наприклад, при роботі з апаратною частиною або при обробці даних. Для використання бібліотеки необхідно вибрати меню Sketch> Include Library. Одна або кілька директив #include будуть розміщені на початку коду скетчу з подальшою компіляцією бібліотек разом зі скетчем. Завантаження бібліотек вимагає додаткового місця в пам'яті Arduino. Невикористані бібліотеки можна видалити з скетчу прибравши директиву #include.

11.3 Основні функції для роботи зі світлодіодами

Для роботи зі світлодіодом, підключеним до плати Arduino, необхідно знати і вміти використовувати такі функції і константи:

– оператор setup();
– оператор loop();
– функція pinMode()
– функція digitalWrite();
– функція delay();
– константи OUTPUT, HIGH, LOW.

Далі наведений код програми найпростішого прикладу мерехтіння вбудованим в плату Arduino світлодіодом, який підключено до 13 виводу.

```
void setup()
{ pinMode(13, OUTPUT);
 }
```

Ця функція виконується на початку роботи програми (після запуску мікроконтролера). Тобто, послідовно виконується кожна команда, яка знаходиться між фігурними дужками цієї функції.

Наприкінці кожного рядка необхідно поставити символ закінчення команди “;”. Тут функція setup містить одну єдину команду – pinMode (13, OUTPUT) . Ця команда налаштовує 13 порт Arduino, як вивід. Порт № 13 знаходиться на верхній колодці портів Arduino.

Після функції setup виконується функція loop.

```
void loop()
{
digitalWrite(13, HIGH); // вмикаємо світлодіод
 delay(1000); // чекаємо секунду
 digitalWrite(13, LOW); // вимикаємо світлодіод
 delay(1000); // чекаємо секунду
 }
```

На відміну від setup, функція loop постійно повторюється – як тільки послідовно виконані всі команди в дужках, функція запускається знову. Функція loop для цього прикладу складається з чотирьох команд:

На порт 13 подається напруга (5 В) – світлодіод вмикається.

Затримка до виконання наступної команди 1000 мілісекунд (одну секунду).

Порт 13 з’єднується з землею – світлодіод вимикається.

Ще одна затримка на 1 секунду.

Після виконання усіх чотирьох команд, знову виконується перша команда (включення світлодіоду) и так продовжується доти, доки Arduino включена або доки не буде натиснута кнопка RESET.

11.4 Симуляція роботи світлодіоду та Arduino в 123D Circuit

Сервіс 123D Circuits компанії Autodesk дає можливість створювати електронні схеми. Веб-додаток має підтримку платформи Arduino та дозволяє редагувати код, а також дає можливість у візуальному режимі будувати схеми і виконувати інтерактивну імітацію їх роботи у реальному часі. Працювати над схемами можна спільно з іншими людьми, використовуючи бібліотеку компонентів.

Для роботи з сервісом 123D Circuits перейдіть за посиланням http://123d.circuits.io/.

Після проходження реєстрації в 123D Circuits ви опинитесь на головній сторінці (рис. 11.4). Домашня сторінка показує список схем, які ви створили, а також нещодавно створені/редаговані.

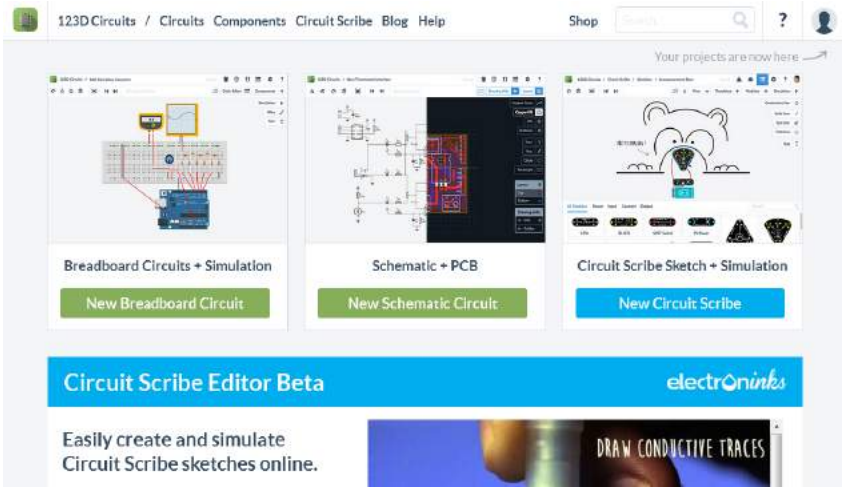


Рисунок 11.4 – Головна сторінка 123D Circuits

Щоб створити нову схему, клікніть на кнопку «+New», яка знаходиться у верхньому меню. Після, натисніть кнопку «New Electronics Lab» на правій панелі вибору. Далі вам буде представлена сторінка для побудови свого проекту з основними функціями та інструментами для роботи зі схемою та програмою. Веб-інтерфейс програмного забезпечення, представлений на рисунку 11.5, складається з таких основних блоків:

- робоча область створення схеми;
- панель керування проектом;
- меню вибору відображення схеми (Breadbord View, Schematics View, PCB View).

Для розуміння роботи в середовищі 123D Circuits, розробимо простий проект мерехтіння світлодіода. Створюємо схему за прикладом, наведеним на рисунку 11.6.

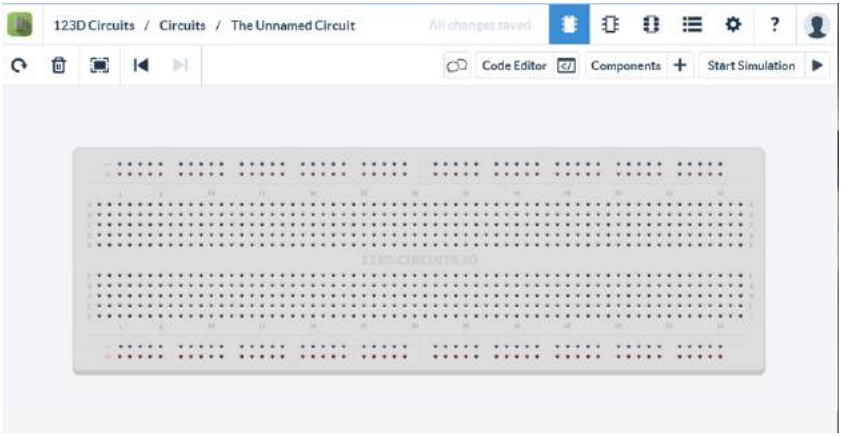


Рисунок 11.5 – Інтерфейс розробки проекту схеми

Клікаємо на кнопку Components, яка знаходиться на панелі керування проектом праворуч. У вікні компонентів, яке відкрилося, знаходимо плату Arduino Uno, обираємо її і розташовуємо на робочій поверхні. Далі обираємо LED (світлодіод) і розташовуємо його на Bredbord. Відкривається вікно налаштування параметрів світлодіода, де можемо задати його ім'я та колір. Після цього, встановлюємо резистор і також задаємо його параметри: ім'я та опір 100Ом. Коли всі компоненти встановлені, з'єднуємо їх.

Викличемо вікно редагування коду, клікнувши на кнопку «Code Editor», яка знаходиться праворуч на панелі керування проектом. У редактор копіюємо код:

```
int led = 8;
void setup() {
  pinMode(led, OUTPUT);
}
void loop() {
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```

Компілюємо і завантажуємо код, клікнувши на кнопку «Aplоad&Run».

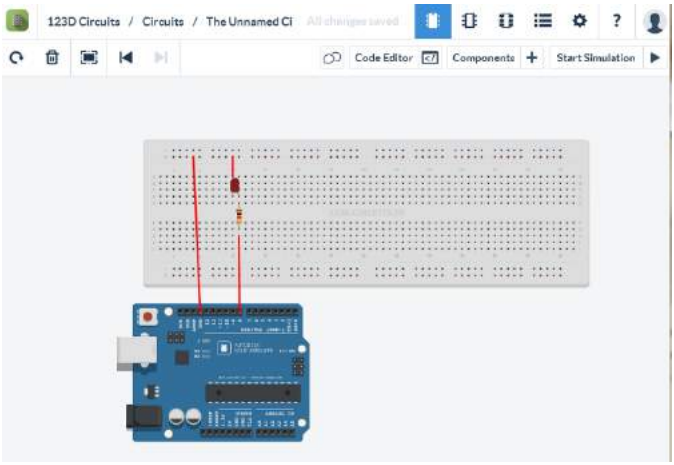


Рисунок 11.6 – Приклад схеми

11.5 Практична робота з симуляції проекту за допомогою 123D Circuits

Завдання 1. Розробіть у 123D Circuits проект підключення світлодіода. За допомогою мультиметра виконайте вимірювання напруги, яку дає плата Arduino Uno, встановіть відповідний резистор для обраного світлодіода.

Завдання 2. Розробіть проект мерехтіння 6 світлодіодів, використовуючи цикл FOR.

Завдання 3. Розробіть у 123D Circuits будь-який проект з підключенням осцилографа.

Завдання 4. Виконайте підключення плати Arduino до комп'ютера та напишіть програму мерехтіння світлодіоду, який встановлений за замовчанням на 13 виході плати. Для цього:

- запустіть на комп'ютері середовище розробки Arduino;
- створіть програмний код;
- підключіть плату Arduino через USB до ПК;
- натисніть кнопку Verify і переконайтесь, що у нижній частині вікна з'явився надпис Done Compiling. Це означає, що у написаній програмі не знайдено помилок;

- виберіть у Tools->Board ваш тип плати. Перевірте, чи правильно обрано USB-порт в Tools->Serial port. Після цього натисніть на кнопку Upload;
- якщо внизу з'явився надпис "Done uploading" – процес запису пройшов успішно;
- підключіть 3 світлодіоди (послідовно червоний, жовтий і зелений) до плати Arduino. Написати код згідно з алгоритмом, представленим на рисунку 11.7.

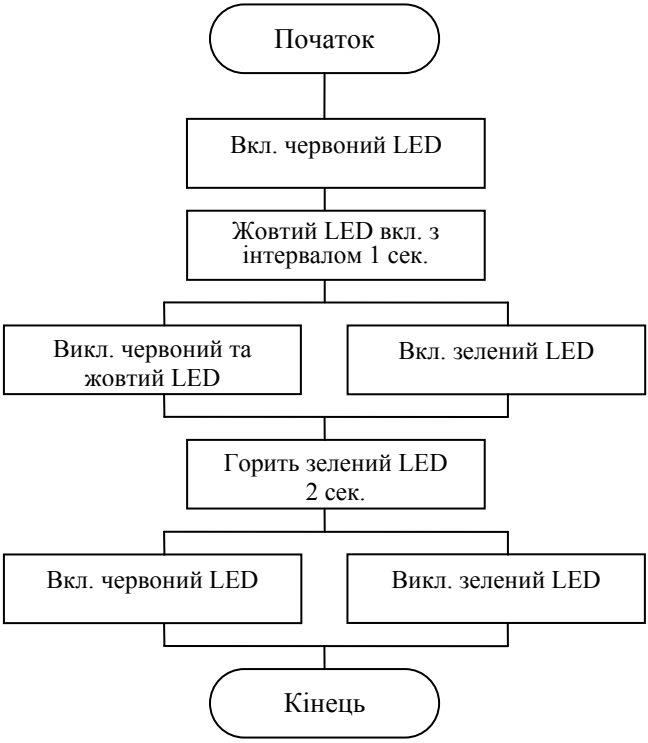


Рисунок 11.7 – Схема алгоритму роботи світлофора

12 СТВОРЕННЯ ПРОЕКТУ РОБОТИ ЗІ СВІТЛОДІОДАМИ З ВИКОРИСТАННЯМ СЕРЕДОВИЩ PROTEUS VSM ТА ATMEL STUDIO

12.1 Створення проекту програми для мікроконтролера у середовищі Atmel Studio

Atmel Studio – інтегроване середовище розробки (IDE) для розробки 8-ми і 32-х бітних додатків від компанії Atmel, що працює в операційних системах Windows NT/2000/XP/Vista/7 (рис.12.1). Atmel Studio містить асемблер і симулятор, що дозволяє відстежити виконання програми. Atmel Studio містить також менеджер проектів, редактор вихідного коду, інструменти віртуальної симуляції та внутрішньосхемного налагодження, дозволяє писати програми на асемблері або на C / C + + [38-39].

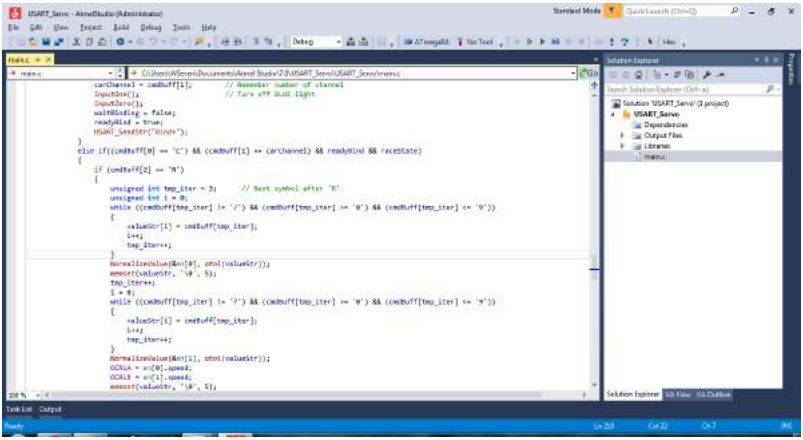


Рисунок 12.1 – Робоче середовище Atmel Studio

Після того, як ви встановите *Atmel Studio 6.2* на свій комп'ютер, створіть новий проект:

Відкрийте Atmel Studio 6.2 та виберіть **New Project**. У діалоговому вікні New Project, виберіть **GCC C Executable Project,** як шаблон (рис.12.2). Введіть ім'я проекту і вкажіть місце, де він буде зберігатися. Назвемо наш проект "BlinkLED". Зніміть прапорець

Create directory for solution для спрощення структури каталогів проекту. Клікніть **OK**.

У вікні Device Selection (рис. 12.3), оберіть необхідне ім'я пристрою AVR. Клікніть **OK** для створення нового проекту.

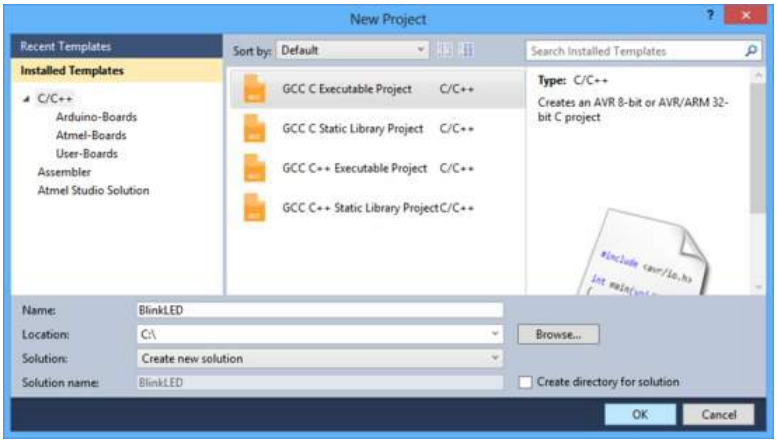


Рисунок 12.2 – Діалогове вікно New Project в Atmel Studio 6

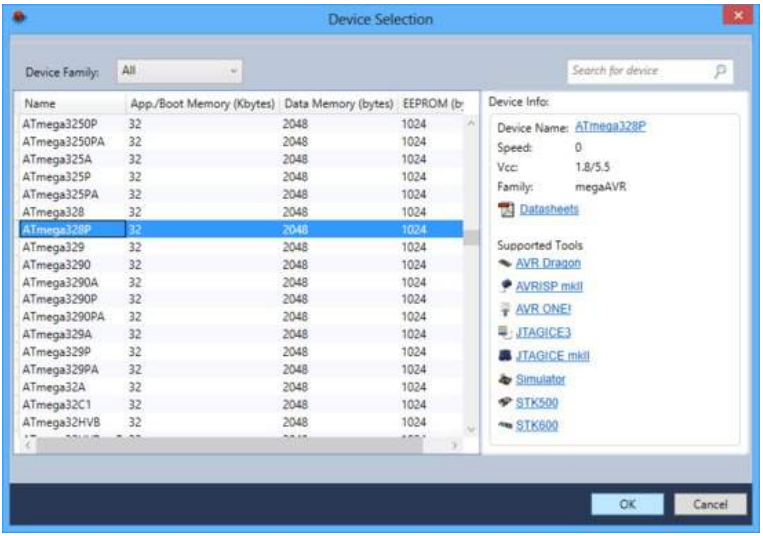


Рисунок 12.3 – Діалогове вікно Device Selection в Atmel Studio 6

Напишіть код у BlinkLED.cpp наведений нижче:

```
#define F_CPU 8000000    // частота AVR в Гц,
використовується для util/delay.h
#include <avr/io.h>
#include <util/delay.h>

int main() {
    DDRD |= (1<<DDD1);          // установка LED виходу
PD1 на вихід
    while (1) {
        PORTD |= (1<<PORTD1);   // PD1 включення
        _delay_ms(50);          // затримка на 50 мс
        PORTD &= ~(1<<PORTD1);  // PD1 виключення
        _delay_ms(100);         // затримка 100 мс
    }
}
```

На панелі інструментів клікніть кнопку **Build Solution** (або натисніть **F7**), щоб скомпілювати код (рис.12.4).

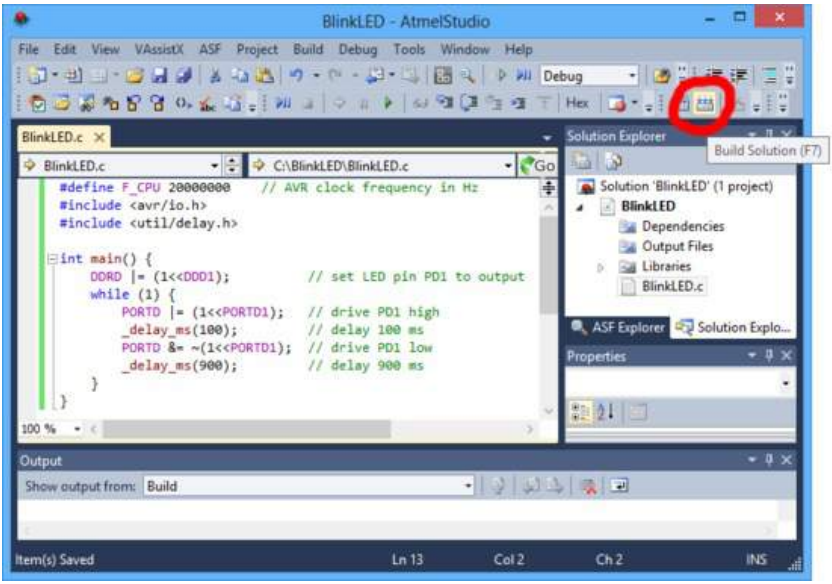


Рисунок 12.4 – Побудова проекту

Після компіляції перейдіть в директорію проекту і знайдіть файл BlinkLED.hex – це файл, який представляє собою довільні двійкові дані у текстовому вигляді. Він є стандартом де-факто при прошивці різноманітних мікросхем. Саме його необхідно буде завантажити у мікроконтролер схеми у Proteus VSM, яка буде побудована далі.

12.2 Симуляція роботи мікроконтролера за допомогою Proteus VSM

На рисунку 12.5 показане розведення виводів мікроконтролера ATMega8. Світлодіод ми будемо підключати до третього виводу мікроконтролера (PD1).

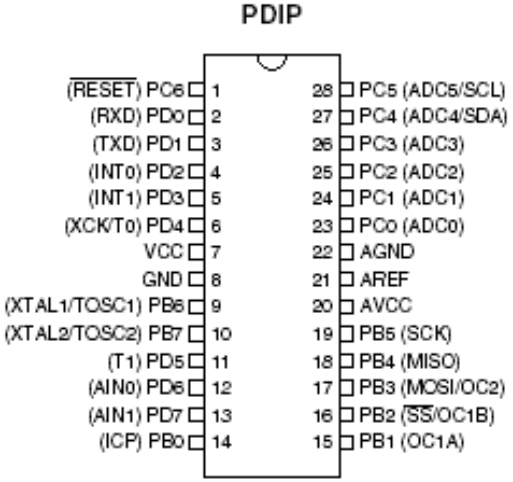


Рисунок 12.5 – Розташування виводів мікроконтролера ATMega8

Створіть новий проект у Proteus VSM та накресліть схему, як зображено на рисунку 12.6. У менеджері компонентів знайдіть елементи:

- світлодіод;
- мікроконтролер ATMega328P;
- резистор;
- заземлення.

Для того, щоб створити елемент «заземлення» (GROUND) перейдіть у панелі інструментів (знаходиться вертикально зліва) на вкладку Terminals Mode ( ).

Після з'єднання елементів у електричне коло, подвійним кліком клікніть на зображенні мікроконтролера. Відкриється вікно Edit Component (рис. 12.7), в якому у полі Program File вказуємо шлях до файлу BlinkLED.hex. Також перевірте, щоб була встановлена частота мікроконтролера така сама як і у програмному коді, тобто 8МГц.

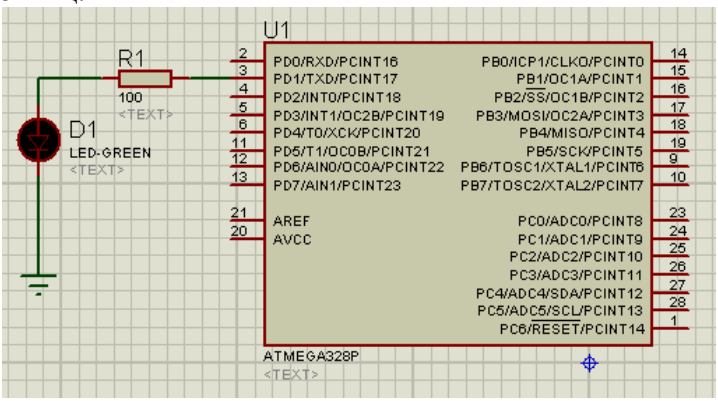


Рисунок 12.6 – Схема підключення

Після цього, на панелі керування симуляцією клікаємо на Play, і якщо все правильно було зроблено, спостерігаємо роботу схеми.

12.3 Практична робота з використання Atmel Studio

Розробіть програму і виконайте симуляцію мерехтіння трьох різних світлодіодів, підключених до мікроконтролера ATMega8. Схема підключення у середовищі Proteus VSM приведена на рис. 12.8.

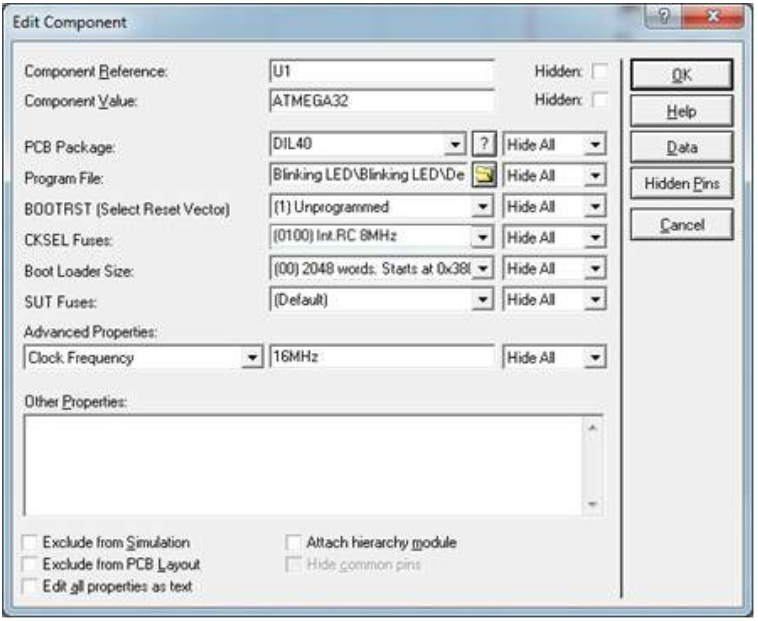


Рисунок 12.7 – Вікно Edit Component

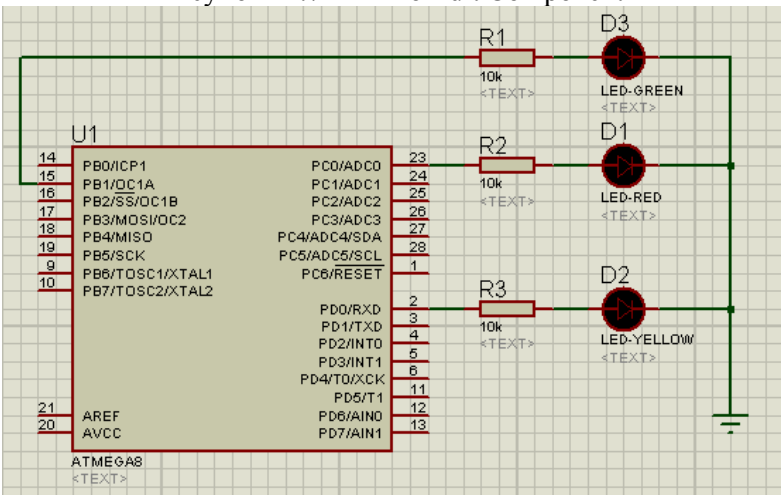


Рисунок 12.8 – Схема підключення світлофора

13 ПРОЕКТ НА ARDUINO З ПІДКЛЮЧЕННЯМ RGB СВІТЛОДІОДУ, КНОПКИ ТА ПОТЕНЦІОМЕТРУ

13.1 Особливості проектів з RGB світлодіодами

Розглянемо роботу RGB (Red, Green, Blue) світлодіоду з платою Arduino. Для управлінням кольором світлодіоду буде використовуватись функція analogWrite(). Якщо задіяти на платі контакти з відміткою «~», ми можемо регулювати напругу, яка подається на відповідний світлодіод.

RGB світлодіод має 4 виводи (рис.13.1).

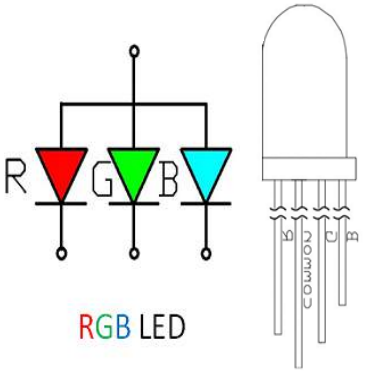


Рисунок 13.1 – RGB світлодіод

На перший погляд RGB світлодіод виглядає, як звичайний світлодіод, однак всередині RGB насправді 3 світлодіоди: один червоний, один зелений і один синій.

Три виводи, кожен окремо, підключені до позитивного виводу кожного одиночного світлодіоду усередині RGB світлодіоду та один з виводів підключається до від'ємного, який є спільним для виводу усіх трьох світлодіодів. До загального від'ємного виводу підключається заземлення (GROUND).

Шляхом контролю яскравості кожного індивідуального світлодіоду можна отримувати різні кольори, це схоже на змішування трьох кольорів фарби на палітрі. Жорсткий спосіб змінювати колір

RGB світлодіоду – це використовувати різне значення опору резисторів.

Розглянемо невеликий проект з RGB світлодіодом. Для кожного світлодіода потрібен відповідний резистор на 270 Ом, щоб запобігти можливості протікання занадто великих струмів. Ці резистори встановлюються в коло між катодами (червоний, зелений и синій) та керуючими пінами на Arduino (рис. 13.2).

Якщо використовується RGB світлодіод с загальним анодом, замість загального катода, тоді загальний пін світлодіода підключається до піна +5 V замість піна gnd.

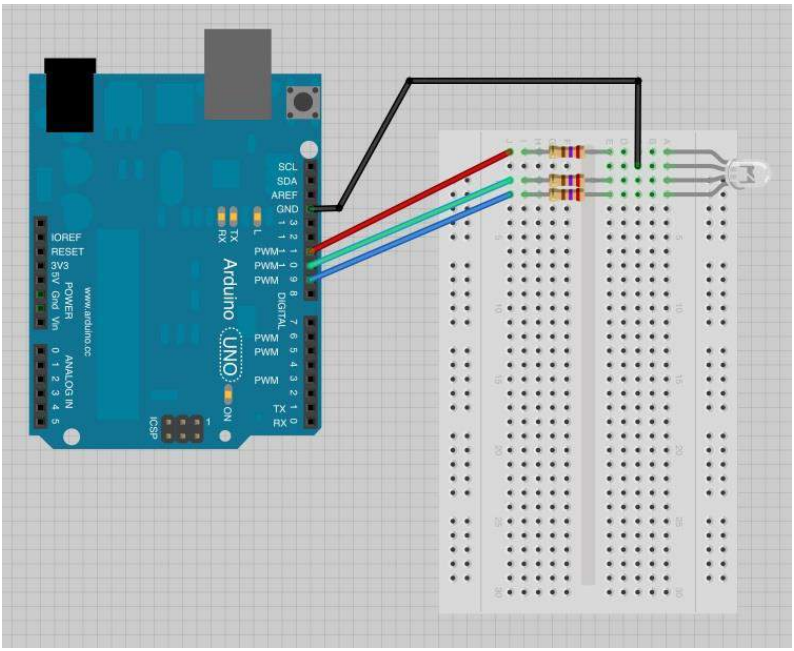


Рисунок 13.2 -Схема підключення

Але більш зручний спосіб – це використання потенціометра - змінного резистора з регульованим опором. Потенціометри використовуються в робототехніці як регулятори різних параметрів – гучності звуку, потужності, напруги, тощо.

Потенціометр має три контакти (рис. 13.3). Середній контакт йде на аналоговий вихід. Зовнішні контакти йдуть до піна із живленням и до землі.

Рисунок 13.3 – Зображення потенціометра

Для детального розуміння роботи потенціометра розглянемо приклад його роботи.

На рисунку 13.4 зображена схема підключення потенціометра до плати. У цьому випадку потенціометр підключений між землею і +5В потенціалами (оскільки плата Arduino живиться +5В), а движок сполучений з каналом аналого-цифрового перетворювача мікроконтролера. У такому разі можна регулювати вихідну напругу потенціометра в межах від 0 до 5В.

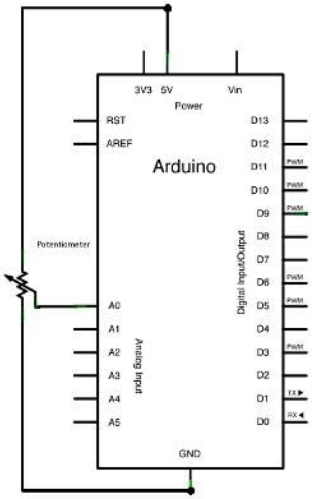


Рисунок 13.4 – Схема підключення потенціометра до плати Arduino

Код програми для роботи потенціометра:

```
void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(A0);  // отримуємо
поточне значення
  Serial.println(sensorValue, DEC); // виводимо
результат на монітор
}
```

При ініціалізації встановлюємо потрібну швидкість зв'язку: **Serial.begin(9600);**. Далі в циклі ми постійно зчитуємо дані, що поступають з потенціометра за допомогою методу analogRead(). Оскільки значення знаходитимуться в діапазоні від 0 до 1023, можемо використовувати тип int для змінної sensorValue.

Отриманий результат будемо виводити у вікно послідовного монітора в десятковому форматі.

В попередніх розділах вже було розглянуте загальне визначення кнопки. На рисунку 13.5 зображена схема підключення кнопки до піна плати Arduino.

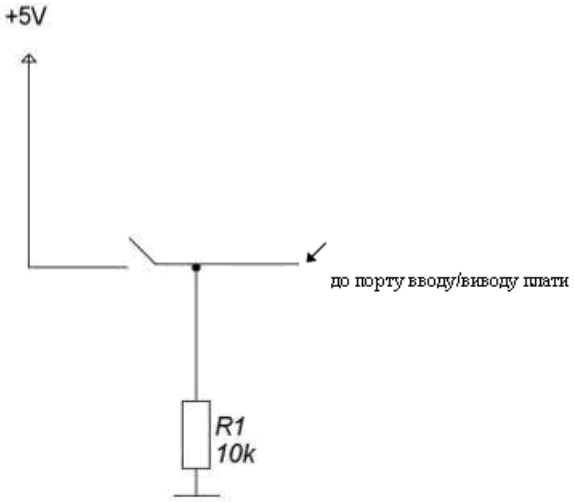


Рисунок 13.5 – Схема підключення кнопки

Пін плати на вході повинен мати стан 0 або 1. Коли стан «підвішений», тобто не визначений, на вході будуть збиратися різні зовнішні наводки (статичні, електричні, електромагнітні), що буде призводити до хибного спрацьовування кнопки. Щоб підвести пін до 0 або 1 використовують підтягуючі резистори. Вони бувають двох видів - верхньої або нижньої підтяжки. Верхні підключаються до плюса живлення, а нижні до мінуса.

Після підключення схеми переходимо до написання програми. Необхідно пін, до якого буде підключатися кнопка, ініціалізувати на вхід за допомогою команди: pinMode(buttonPin, INPUT); //де buttonPin – це номер піна, до якого підключена кнопка. Також необхідно об’явити змінну, в яку буде записуватися статус кнопки.

13.2 Практична робота с потенціометром та свідлодіодами

Завдання 1. Виконайте підключення до 5 порту звичайного світлодіода. Підключіть до 6 порту кнопку. Напишіть програму, щоб при натисканні кнопки світлодіод, підключений до 5 порту, вмикався, а вбудований світлодіод до 13 порту вимикався.

Завдання 2. При повороті ручки потенціометра змінювати яскравість світлодіоду. Схема підключення зображена на рисунку 13.6.

Завдання 3. Підключіть RGB світлодіод до плати та почергово змінюйте значення опору резисторів.

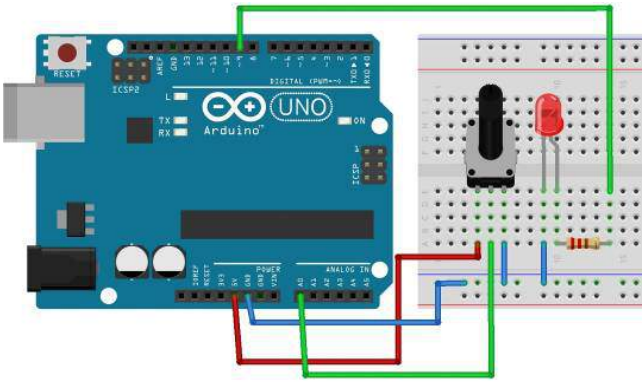


Рисунок 13.6 – Схема підключення до Завдання 2.

14 ПРОЕКТ З СЕРВОПРИВОДОМ НА ARDUINO

14.1 Особливості будови та різновиди серводвигунів

Сервопривід – привід з управлінням через від'ємний зворотній зв'язок, що дозволяє точно керувати параметрами руху. Сервопривід з мотором (серводвигун) призначений для приведення в рух пристроїв керування через обертання вихідного валу.

Сімейство серводвигунів різноманітне, його можна розподілити на декілька груп (рис.14.1). Відміні особливості обумовлені наступними факторами:

- конструкція двигунів (статор, ротор);
- необхідні системи регулювання;
- система зворотнього зв'язку (датчики).

Рисунок 14.1 – Огляд серводвигунів

Будова сервоприводу для радіокерованих моделей зображена на рисунку 14.2. Він складається з електродвигуна, укладеного в один корпус з редуктором і керуючою електронікою, яка найчастіше складається з потенціометра зворотнього зв'язку та плати управління. Серводвигун використовує три дроти для роботи. Плюсовий дріт для

живлення (зазвичай 4.8В-6В), мінусовий дріт та сигнальний дріт. Чорний(коричневий), червоний, білий(жовтий) – відповідно, земля, живлення та керуючий сигнал ШІМ.

Керуючий сигнал передає інформацію щодо положення вихідного валу. Вал пов'язаний з потенціометром, який визначає його положення. Контролер згідно опору потенціометра і значенням керуючого сигналу визначає, в який бік потрібно обертати мотор, щоб отримати потрібне положення вихідного валу.

Рисунок 14.2 – Конструкція сервоприводу

Керування сервоприводом відбувається шляхом подачі на нього прямокутних імпульсів по сигнальному виводу з частотою біля 50Гц, амплітудою не менше 3.3В (часто не менше 4.8В), шириною стандартно від 1000 до 2000мкс, що відповідає стандартним крайнім положенням (див. рис. 14.3). Зазвичай реальний діапазон може бути трохи ширшим (наприклад, 900-2100мкс), але це вже залежить конкретно від виробника сервоприводів.

14.2 Створення сервоприводу у Proteus

Ми перевіримо, як спроектувати сервопривід у Proteus ISIS, а також спроектуємо керуючий привід постійного струму за допомогою

логічних елементів. З початку, ми створимо двигун постійного струму, який присутній в Proteus і дуже простий у використанні. Ми будемо керувати їм шляхом подачі напруги на його обидві сторони, тобто прямим методом.

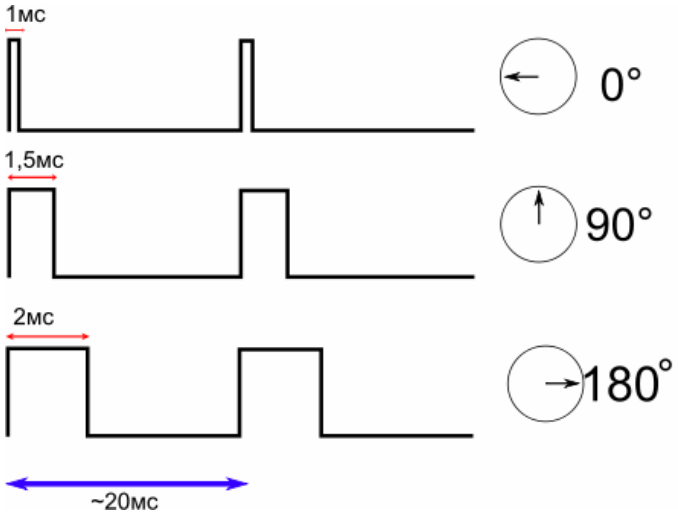


Рисунок 14.3 – Керуючі сигнали

Двигун постійного струму являє собою простий двигун, який потребує різної полярності на своїх двох кінцях. Якщо ця полярність в прямому напрямку, то двигун постійного струму рухається в одному напрямку, і якщо ми змінимо полярність, то двигун буде рухатися в протилежному напрямку. Отже, створимо DC Motor Drive Circuit в Proteus ISIS.

14.2.1 Простий DC Motor Drive Circuit в Proteus ISIS

Виберіть компоненти з бібліотеки Proteus: Motor, Logic State (рис.14.4). Logic State має два стани 1 і 0. Коли 0 це означає 0В і коли 1 – 5В. Тепер розробіть схему, як показано на рисунку 14.5.

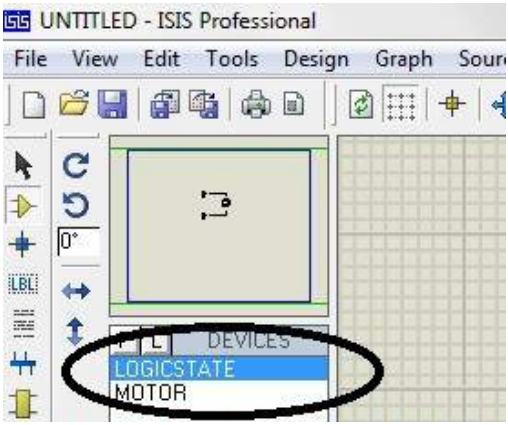
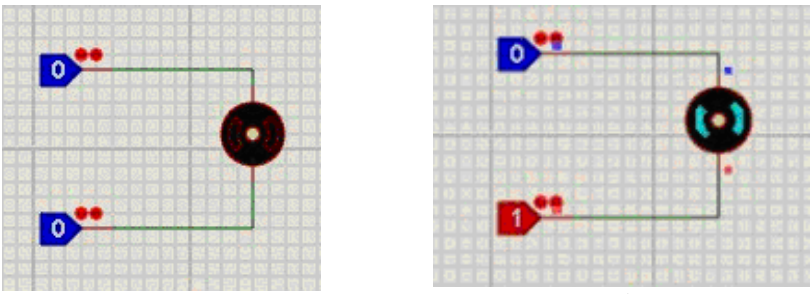


Рисунок 14.4 - Вибір компонентів з бібліотеки Proteus



а б

Рисунок 14.5 – Схема з різними станами для серводвигуна

Як бачимо, на схемі додано два логічні стан на обох сторонах двигуна. Напрямок двигуна буде залежати від цієї логіки. Таким чином, буде всього чотири стани:

1. Коли обидва стани 0, двигун не буде рухатися і залишатиметься нерухомими (рис.14.5а).
2. Коли обидва стани знаходяться на 1, також двигун не буде рухатися і залишатиметься нерухомими.
3. Двигун буде рухатися за годинниковою стрілкою, коли верхній стан 1, а нижній 0.
4. Двигун буде рухатися проти годинникової стрілки, коли верхній стан 0, а нижній 1 (рис.14.5б).

14.2.2 Керування сервоприводом у Proteus ISIS за допомогою ШІМ

Для керування сервоприводом нам потрібно формувати ШІМ з частотою 50Гц. При цьому для положення “0” довжина імпульсу повинна складати 1000 мікросекунд, а для положення “максимум” – 2000 мікросекунд. Середнє положення – 1500 мікросекунд (рис. 14.3). Розглянемо це конкретно на прикладі.

Відкриваємо Proteus ISIS та створюємо новий проект. У бібліотеці елементів знаходимо сервопривід та розміщуємо так, як зображено на рисунку 14.6. На ці три сервоприводи ми будемо відповідно надсилати сигнали різної довжини: 1мс, 1.5мс та 2 мс.

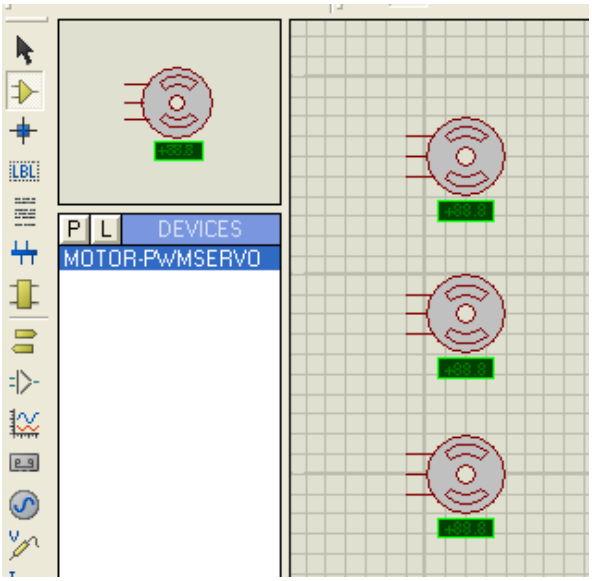


Рисунок 14.6 – Сервоприводи для проекту

Встановлюємо елемент живлення «POWER» та у його властивостях задаємо значення живлення +12В (рис.14.7)

Після встановлення заземлення електрична схема буде виглядати так, як зображено на рисунку 14.8.

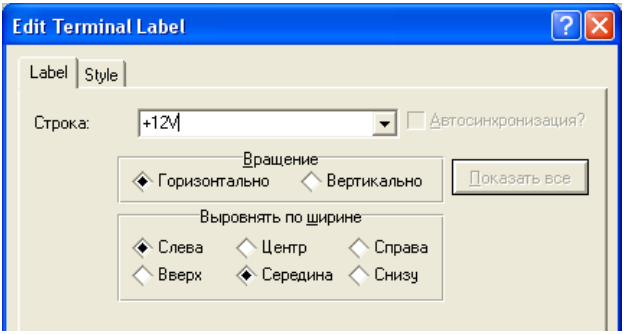


Рисунок 14.7 – Встановлення параметру елемента живлення

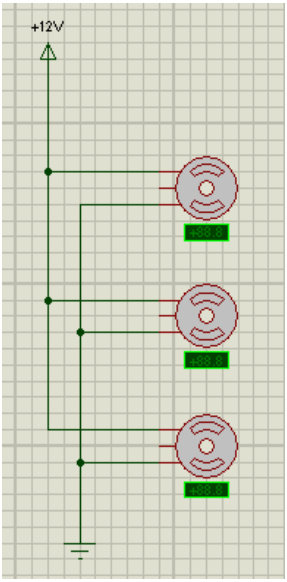


Рисунок 14.8 – Схема підключення сервоприводів

Для генерування сигналів потрібної частоти та довжини встановимо елементи, які будуть генерувати три різні сигнали на кожен з сервоприводів. Перейдіть у лівій паналі інструментів у режим Generator Mode та виберіть компонент PULSE. Встановіть три компоненти PULSE на схемі та з’єднайте їх як зображено на рисунку 14.9.

Для першого зверху генератора імпульсу встановіть параметр довжини (ширина) імпульсу 1мс та частота 50Гц (рис.14.10).

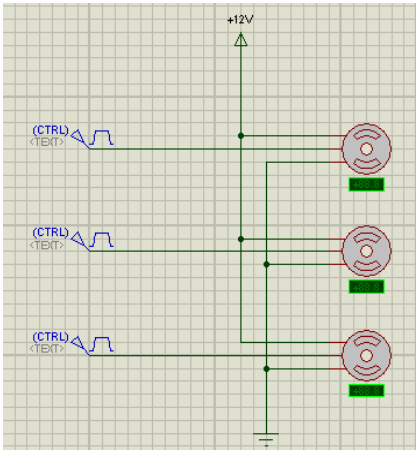


Рисунок 14.9 – Схема підключення генераторів сигналу

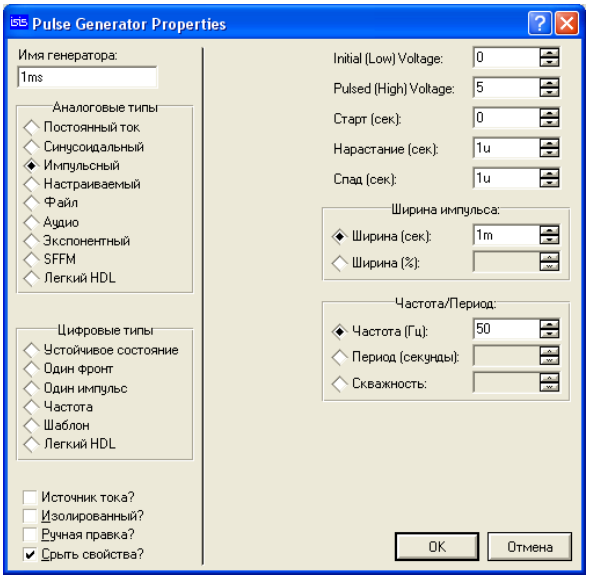


Рисунок 14.10 – Встановлення параметрів сигналу

Для двох наступних за аналогією з попереднім встановіть довжину імпульсу 1.5мс та 2мс відповідно. Коли схема буде завершена, запустіть симуляцію. У результаті симуляції верхній сервопривід повернеться у крайнє ліве положення, середній встановить нульове положення, а нижній повернеться у крайнє праве положення, як зображено на рисунку 14.11.

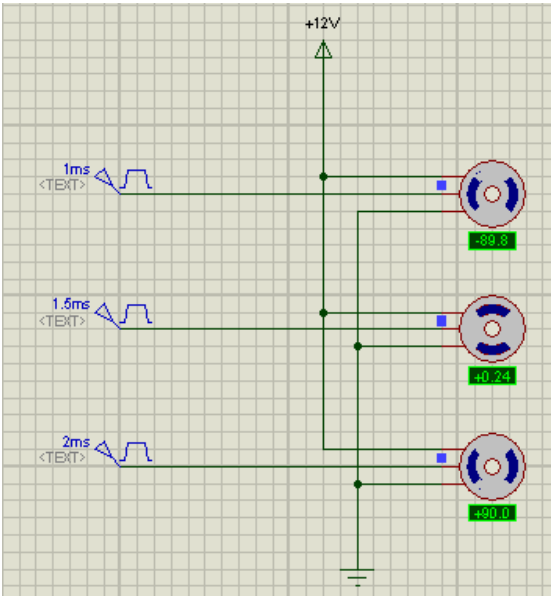


Рисунок 14.11 – Результат симуляції

14.3 Створення проекту з сервоприводом на Arduino

Для розглядання прикладу керування положенням сервоприводу за допомогою Arduino необхідна плата та сервопривід. Схема підключення зображена на рисунку 14.12. Як вже було сказано вище серводвигун має 3 дроти: живлення, заземлення, сигнальний. Дріт живлення, як правило, червоний, і повинен бути з'єднаній з піном 5V на платі Arduino. Дріт заземлення, зазвичай, чорного або коричневого кольору і повинен бути підключений до заземлення на платі Arduino. Сигнальний контакт, зазвичай, жовтого, білого або помаранчевого кольору, і повинен бути підключений до контакту 9 на платі Arduino.

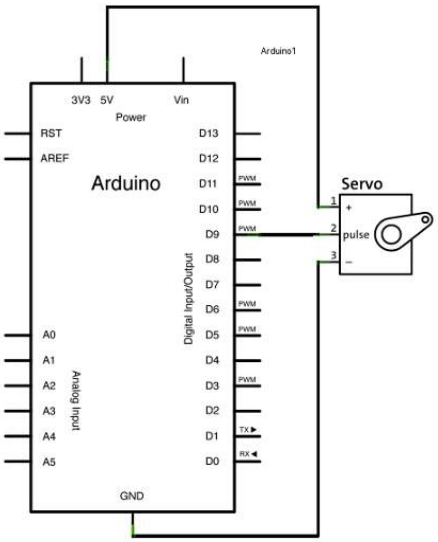


Рисунок 14.12 – Схема підключення сервоприводу до Arduino

Для написання програми у середовищі Arduino є спеціальна бібліотека для роботи з сервоприводом «Servo.h». Далі наведено простий приклад програми роботи з сервоприводом:

```
#include <Servo.h>

Servo myservo;  // створення об’єкта сервоприводу для керування серво; максимум можна створити 8 об’єктів сервоприводів

int pos = 0;    // змінна для збереження положення сервоприводу

void setup()
{
  myservo.attach(9);  // призначити серво, який підключений до 9 піна до об’єкту сервоприводу
}
```

```
void loop()
{
  for(pos = 0; pos < 180; pos += 1)  // рух від 0 до 180 градусів
  {                                  // з кроком в 1 градус
    myservo.write(pos);              // дає команду сервоприводу перейти на
позицію ‘pos’
    delay(15);                       // затримка 15мс для того, щоб сервопрівід
досяг необхідної позиції
  }
  for(pos = 180; pos>=1; pos-=1)     // рух від 180 до 0 градусів
  {
    myservo.write(pos);              // дає команду сервоприводу перейти на
позицію ‘pos
    delay(15);                       // затримка 15мс для того, щоб сервопривід
досяг необхідної позиції
  }
}
```

14.4 Практична робота з сервоприводом

Завдання 1. Підключити кнопку і сервопривід до плати Arduino. Написати програму, щоб при натисканні кнопки сервопривід рухався за годинниковою стрілкою. При відпусканні кнопки сервопривід зупинявся.

Завдання 2. Підключити кнопку і сервопривід до плати Arduino. Написати програму, щоб при натисканні кнопки сервопривід рухався проти годинникової стрілки. При відпусканні кнопки сервопривід зупинявся.

15 СТВОРЕННЯ ПРОЕКТУ НА ARDUINO З ВИКОРИСТАННЯМ РЕЛЕ

15.1 Особливості будови реле

Реле є найкращим способом управління будь-яким двигуном. У цьому розділі ми будемо розглядати, що таке реле, як його використовувати та яким чином контролювати реле в Proteus ISIS. Реле є одним з ключових компонентів будь-якого електронного пристрою або електричного кола і, зазвичай, є проблемою при проектуванні.

Спочатку, змоделюємо реле в простій схемі, в якій при запуску моделювання, реле автоматично активується, а після цього будемо керувати реле за допомогою логіки, тобто, коли ви подаєте + 5В до нього, то реле буде активуватися і коли ви даєте Gnd, то воно буде знеструмлене.

Для управління різними виконавчими пристроями, комутації кіл, управління приладами в електроніці активно застосовуються електромагнітні реле (рис.15.1).

Рисунок 15.1 – Зовнішній вигляд електромагнітного реле

Будова реле досить проста. Його основою є котушка, що складається з великої кількості витків ізольованого проводу. Всередину котушки встановлюється стрижень з м'якого заліза. В результаті виходить електромагніт. Також в конструкції реле присутній якір. Він закріплений на пружному контакті. Сам же пружний контакт закріплений на ярмі. Разом зі стрижнем і якорем ярмо утворює магнітопровід.

Якщо котушку підключити до джерела струму, то магнітне поле, що утворилося, намагнічує осердя. Він, в свою чергу, притягує якір. Якір укріплений на пружному контакті. Далі, пружний контакт

замикається з іншим нерухомим контактом. Залежно від конструкції реле, якір може по-різному механічно управляти контактами.

Внутрішня будова електромагнітного реле показана на рис. 15.2**.** У більшості випадків реле монтується в захисному корпусі. Він може бути як металевим, так і пластмасовим. Це реле без захисного корпусу. Як бачимо, реле має котушку, стрижень, пружний контакт, на якому закріплений якір, а також виконавчі контакти.

На принципових схемах електромагнітне реле позначається як на рис.15.3.

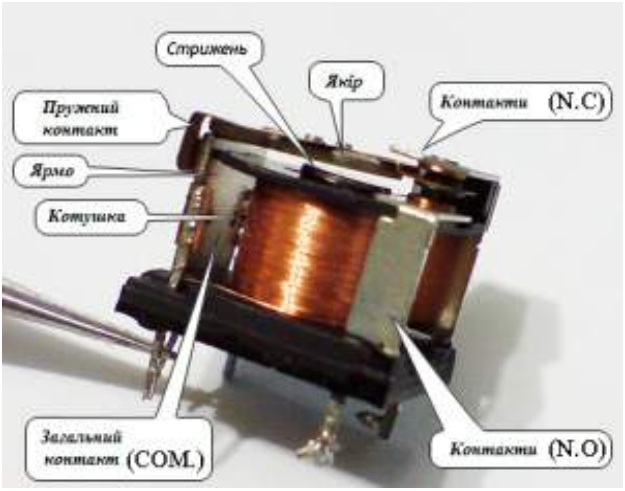


Рисунок 15.2 – Внутрішня будова реле

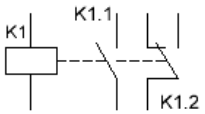


Рисунок 15.3 –Принципова схема електромагнітного реле

Умовне позначення реле на схемі складається наче з двох частин. Одна частина (**К1**) – це умовне позначення електромагнітної котушки. Вона позначається у вигляді прямокутника з двома виводами. Друга частина (**К1.1**; **К1.2**) – це групи контактів, якими

керує реле. Залежно від своєї складності реле може мати досить велику кількість комутованих контактів. Вони розбиваються на групи. Як бачимо, на позначенні зображені дві групи контактів (**К1.1** і **К1.2**).

Як тільки ми замкнемо керуюче коло вимикачем **SA1**, струм від батареї живлення **G1** надійде на реле **К1**. Реле спрацює, і його контакти **К1.1** замкнуть виконавче коло. На навантаження надійде напруга живлення від батареї **G2** і лампа **HL1** засвітиться. Якщо розімкнути коло вимикачем **SA1**, то з реле **К1** буде знята напруга живлення та контакти реле **К1.1** знову розімкнуться і лампа **HL1** вимкнеться.

Комутовані контакти реле можуть мати своє конструктивне виконання. Так, наприклад, розрізняють нормально-розімкнені контакти, нормально-замкнені контакти та контакти на перемикання (перекидні). Розберемося з цим детальніше.

Нормально розімкнені контакти – це контакти реле, які знаходяться в розімкненому стані доти, доки через котушку реле не потече струм. Говорячи простіше, коли реле вимкнено, контакти теж розімкнені. На схемах реле з нормально-розімкненими контактами позначається як на рисунку 15.4.

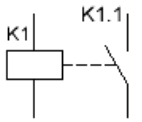


Рисунок 15.4 – Схема реле з нормально розімкненими контактами

Нормально замкнені контакти – це контакти реле, що знаходяться в замкненому стані, доки через котушку реле не почне текти струм. Таким чином, виходить, що при вимкненому реле контакти замкнені. Такі контакти на схемах зображують як на рисунку 15.5.

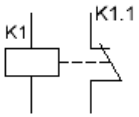


Рисунок 15.5 – Схема реле з нормально замкненими контактами

Перекидні контакти – це комбінація з нормально-замкнених і нормально-розімкнених контактів. У перекидних контактів є спільний провід, який переключається з одного контакту на інший (рис.15.6).

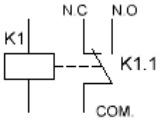


Рисунок 15.6 – Схема реле з перекидними контактами

Сучасні широко розповсюджені реле, як правило, мають перекидні контакти, але можуть зустрічатися і реле, які мають у своєму складі тільки нормально-розімкнені контакти.

В імпортних реле нормально-розімкнені контакти реле позначаються скороченням **N.O.** А нормально-замкнені контакти **N.C.** Загальний контакт реле має скорочення **COM.** (від слова *common* – *«загальний»*).

Зазвичай, розміри електромагнітних реле дозволяють наносити на корпус їх основні параметри. Як приклад, розглянемо імпортне реле Bestar BS-115C. На його корпусі нанесені такі написи, як на рис. 15.7.

Рисунок 15.7 – Зовнішній вигляд реле

COIL 12VDC – це номінальна напруга спрацьовування реле (12В). Оскільки це реле постійного струму, то вказано скорочене позначення постійної напруги (скорочення DC позначає постійний струм/напруга). Англійське слово *COIL* перекладається як «котушка», «соленоїд». Воно вказує на те, що скорочення 12VDC має відношення до котушки реле. Далі на реле вказані електричні параметри його контактів. Зрозуміло, що потужність контактів реле може бути різна. Це залежить як від габаритних розмірів контактів, так і від

використовуваних матеріалів. При підключенні навантаження до контактів реле потрібно знати потужність, на яку вони розраховані. Якщо навантаження споживає потужність більше тієї, на яку розраховані контакти реле, то вони будуть нагріватися, іскрити, “залипати”. Природно, це призведе до швидкого виходу з ладу контактів реле. Для реле, як правило, зазначаються параметри змінного і постійного струму, які здатні витримати контакти.

Так, наприклад, контакти реле Bestar BS-115C здатні комутувати змінний струм в 12А і напругу 120В. Ці параметри зашифровані в написі **12A 120VAC** (скорочення AC позначає змінний струм).

Також реле здатне комутувати постійний струм силою 10А і напругою 28В. Про це свідчить напис **10A 28VDC**. Це були силові характеристики реле, точніше його контактів.

Тепер звернемося до потужності, яку споживає реле. Як відомо, потужність постійного струму дорівнює добутку напруги (U) на струм (I): $P = U * I$. Візьмемо значення номінальної напруги спрацьовування (12В) і споживаного струму (30 мА) реле Bestar BS-115C і отримаємо його споживану потужність (англ. – *Power consumption*). Таким чином, потужність реле Bestar BS-115C становить 360мВт.

Є ще один параметр – це чутливість реле. За своєю сутністю, це і є потужність споживання реле у включеному стані. Зрозуміло, що реле, якому потрібно менше потужності для спрацьовування, є більш чутливим у порівнянні з тими, які споживають велику потужність. Такий параметр, як чутливість реле, особливо важливий для пристроїв з автономним живленням, оскільки включене реле витрачає заряд батарей. Наприклад, є два реле з споживаною потужністю 200мВт і 360мВт. Таким чином, реле потужністю 200мВт має більшу чутливість, ніж реле потужністю 360мВт.

15.2 Створення проектів з реле

15.2.1 Проста схема підключення Реле в Proteus ISIS

Відкрити Proteus ISIS и вибрати компоненти, які зображені на рисунку 15.8, з бібліотеки компонентів Proteus.

Встановіть та з’єднайте компоненти у схему, яка зображена на рисунку 15.9.

Рисунок 15.8 – Вікно вибору компонентів

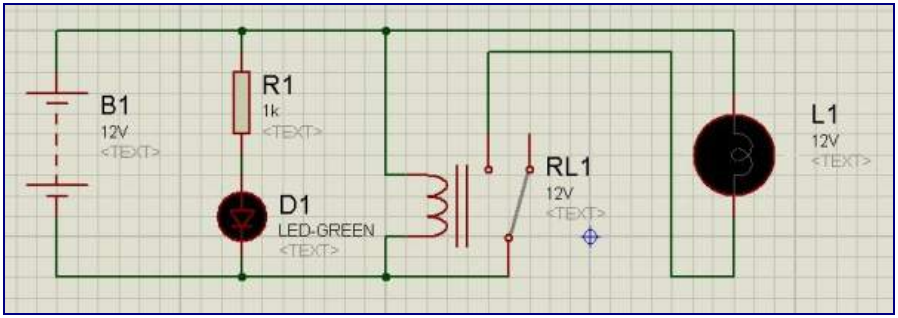


Рисунок 15.9 – З’єднання компонентів в схему.

Коли реле буде під напругою, лампа L1 буде світитись і коли реле знеструмлене, лампа залишиться вимкненою. Світлодіод D1 виконує роботу індикатора, який свідчитиме, що напруга в колі є.

Після розробки схеми, натисніть на кнопку запуска, і якщо все зроблено правильно, то лампа буде світитись як на рисунку 15.10.

Отже, тепер ви можете побачити, що світлодіодний індикатор також загорається. Також бачимо, що реле тепер з'єднане з другим терміналом і, таким чином замикає коло для лампи і лампа тепер також світиться.

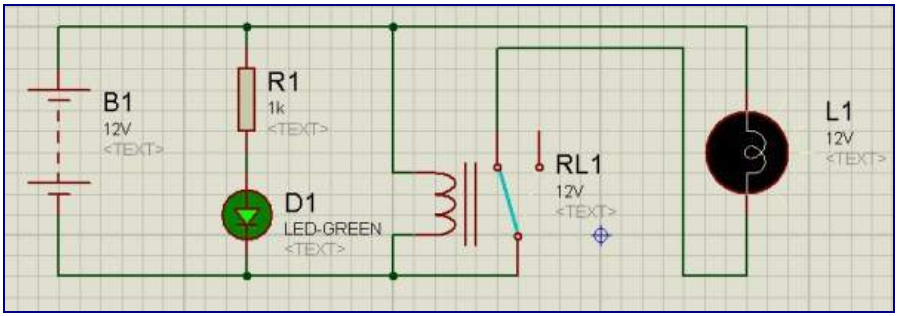


Рисунок 15.10 – Відстежування стану схеми

15.2.2 Керування реле за допомогою логіки в Proteus ISIS

У попередньому розділі ми бачили просту схему, яка керується вручну. Це означає, що для того, щоб включити або виключити його, потрібно включити або виключити живлення, але, зазвичай, потрібно, щоб реле знаходилось під автоматичним контролем якогось мікроконтролера. Зважаючи на те, що мікроконтролери зазвичай працюють від 5В, для того, щоб управляти реле 12В з використанням п’ятивольтового мікроконтролера, ми повинні використовувати транзистор. У цьому випадку, коли ви даєте + 5В, реле приводиться в дію, а коли ви даєте GND реле вимикається.

Знайдемо вказані компоненти (рис. 15.11) та спроектуємо схему (рис.15.12).

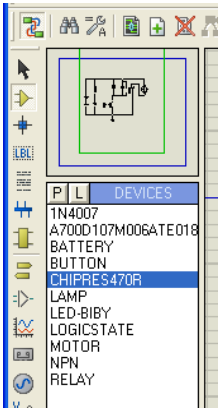


Рисунок 15.11 – Вікно пошуку компонентів

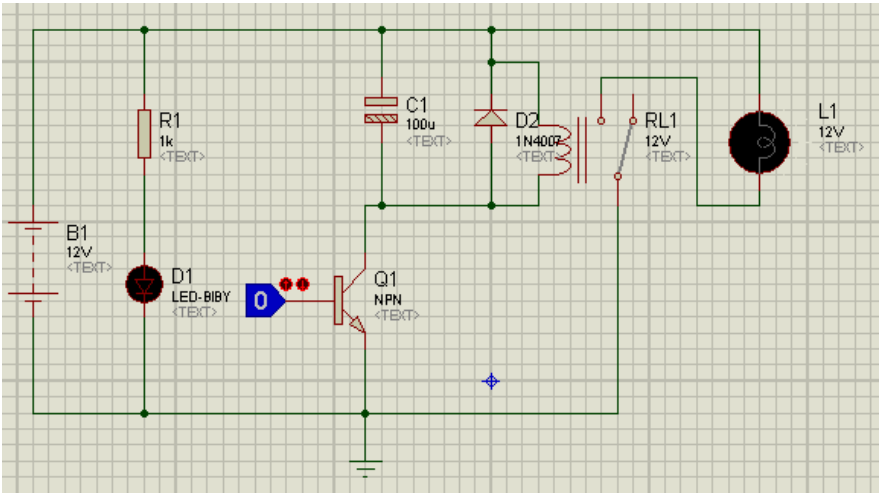


Рисунок 15.12 – Схема для створення

Ми можемо не використовувати мікроконтролер, замість нього будемо використовувати логічний стан, він буде працювати так само, як мікроконтролер. Таким чином, схема дуже схожа на просту схему, яку ми вже створювали в попередньому розділі. Єдина відмінність - тут присутній n-p-n транзистор.

У цьому випадку ми не забезпечуємо подачу живлення безпосередньо до реле, замість цього ми забезпечуємо його за допомогою цього транзистора. Коли логічний стан дорівнює нулю (це означає «землю»), транзистор не працюватиме, і живлення не може досягти реле. Коли ми задаємо логічну 1, що означає +5В на вході транзистора, тоді схема реле буде завершена і реле буде живитися.

Запускаємо симуляцію, вимкнений стан реле виглядає як на рис. 15.13. На даному рисунку можемо побачити, що світлодіодний індикатор працює тому, що живлення подається на схему, але лампа все ще вимкнена і на реле також не подається живлення, оскільки логічний стан 0.

Тепер натисніть на логічному стані та задайте логічну одиницю, тобто +5В, включений стан реле показано на рисунку 15.14.

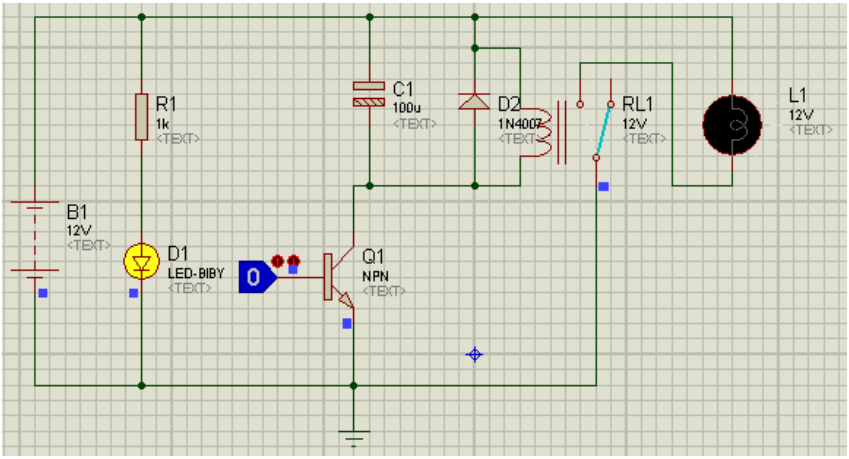


Рисунок 15.13 – Схема з вимкненим реле.

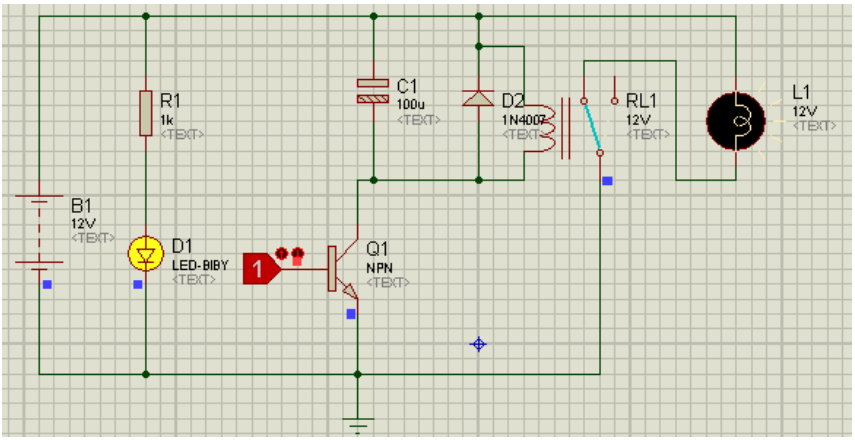


Рисунок 15.14 – Схема з включеним реле.

15.2.3 Робота Arduino з реле

Реле модуль має 3 виводи: VCC (+), GND (-), IN (вивід вхідного сигналу). Підключення модулю дуже просте (рис.15.15):

- VCC на + 5В на Arduino;
- GND на будь який з GND пінів Arduino;
- IN на будь який з цифрових входів/виходів Arduino.

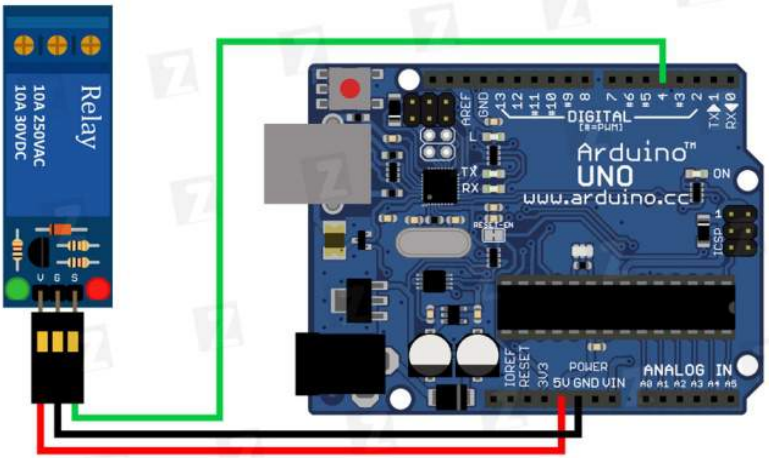


Рисунок 15.15 – Схема підключення до плати Arduino

Переходимо до створення програми. У даному прикладі реле буде включатись и виключатись з інтервалом у 2 секунди.

Приклад програмного коду:

```
// Реле модуль підключено до цифрового виводу 4
int Relay = 4;
void setup()
{
  pinMode(Relay, OUTPUT);
}
void loop()
{
  digitalWrite(Relay, LOW);    // реле вкл.
  delay(2000);
  digitalWrite(Relay, HIGH);   // реле викл.
  delay(2000);
}
```

15.3 Практична робота з реле

Завдання 1. Накресліть схему (рис.15.16) та знайдіть у ній помилки.

Завдання 2. Створіть проект на Arduino з використанням реле та промоделюйте його роботу в програмі 123D Circuit.

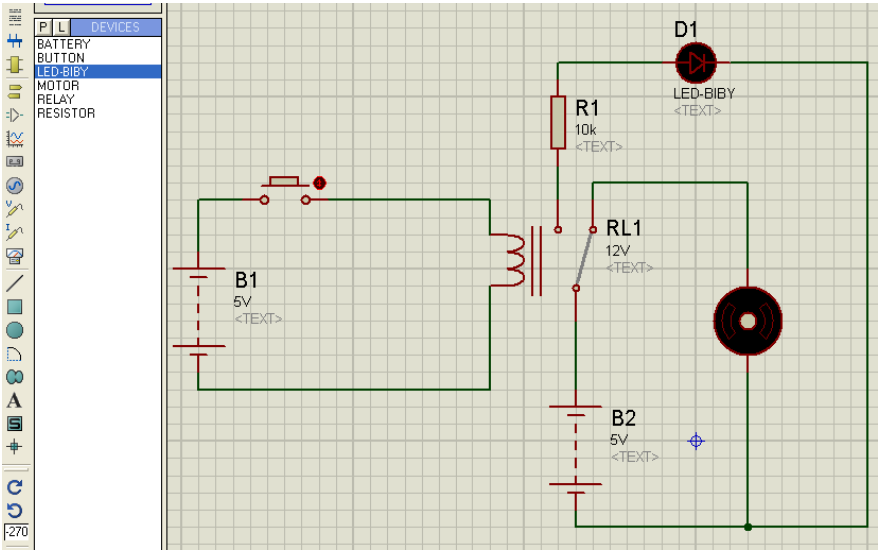


Рисунок 15.16 – Схема до виконання практичної роботи.

16. РОБОТА З БІБЛІОТЕКОЮ ARDUINO У PROTEUS. ПРОГРАМУВАННЯ LCD ДИСПЛЕЮ

16.1 Проект з підключенням плати Arduino та LCD дисплею у середовищі Proteus VSM

У папці Arduino Lib знаходяться два бібліотечних файли Arduino (ARDUINO.IDX, ARDUINO.LIB). Вставте ці два файли у бібліотечну папку Proteus, яка знаходяться: C: -> Program Files (x86) -> Labcenter Electronics -> Proteus 7 Professional -> LIBRARY

Відкриваємо Proteus ISIS та знаходимо у бібліотеці плату Arduino та LCD дисплей LM016L.

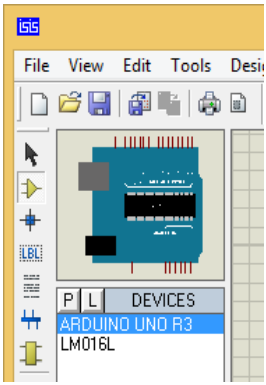


Рисунок 16.1 – Вибір плати з бібліотеки

З'єднуємо виходи Arduino з дисплеєм таким чином:

- LCD RS вихід до 12 виходу Arduino;
- LCD Enable до 11;
- LCD D4 до 5;
- LCD D5 до 4;
- LCD D6 до 3;
- LCD D7 до 2;
- LCD RS до терміналу GROUND.

Далі відкриваємо Arduino IDE. Для того, щоб бачити більш детально, куди зберігається файл з розширенням *hex, який нам знадобиться для симуляції роботи Arduino у Proteus перейдіть Файл-

>Налаштування. Налаштуйте Arduino IDE, щоб показувало докладний звіт при компіляції, поставивши прапорець у потрібних полях.

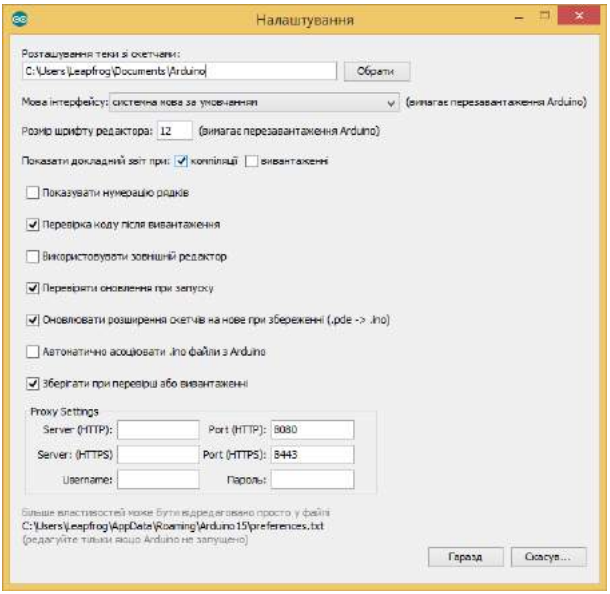


Рисунок 16.2 – Налаштування докладного звіту.

Копіюємо код, приведений нижче, зберігаємо та компілюємо:

```
// include the library code:
#include <LiquidCrystal.h>

// initialize the library with the numbers of the
interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("hello, world!");
}

void loop() {
```

```
  // set the cursor to column 0, line 1
  // (note: line 1 is the second row, since counting
begins with 0):
  lcd.setCursor(0, 1);
  // print the number of seconds since reset:
  lcd.print(millis() / 1000);
}
```

У вікні результатів компіляції знаходимо теку, у якій зберігається файл з розширенням *.hex (рис.16.3).

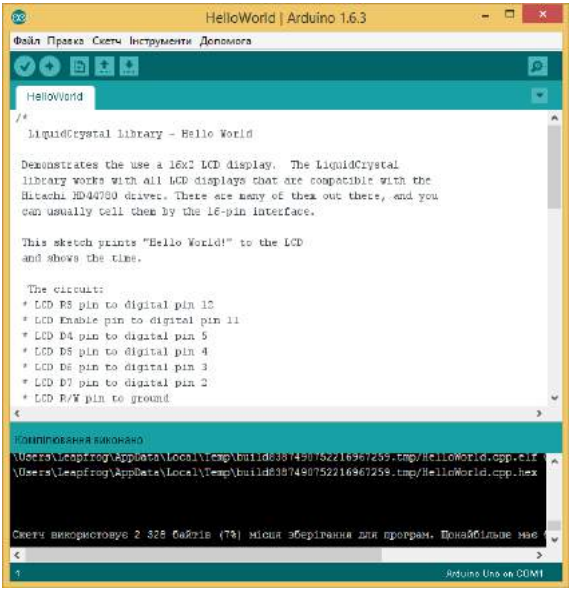


Рисунок 16.3 – Пошук теки з результатами компіляції.

Переходимо до цієї папки та копіюємо два файли (рис. 16.4) до своєї папки.

Для Arduino Uno у Proteus вказуємо шлях до файлу з розширенням *.hex. Запускаємо симуляцію і дивимось результат.

16.2 Практична робота з дисплеєм

Завдання 1. Виконати симуляцію роботи з дисплеєм в 123D Circuits.

Завдання 2. Зареєструватися на http://swed.zntu.adu.ua та провести віддалений експеримент з дисплеєм у віддаленій лабораторії RELDES (REmote Laboratory for Design of Embedded Systems)[40].

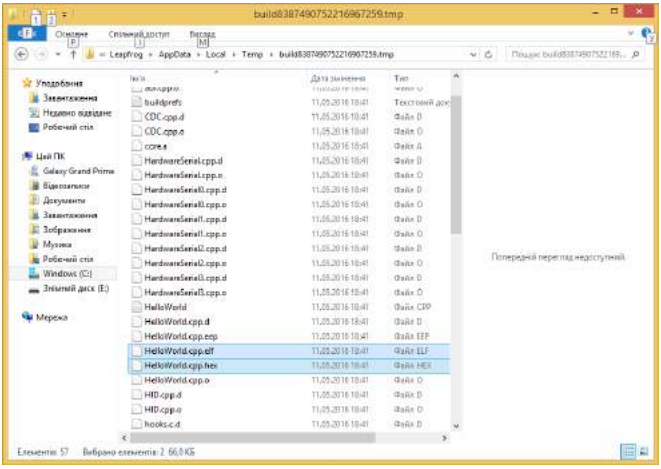


Рисунок 16.4 – Пошук теки з результатами компіляції.

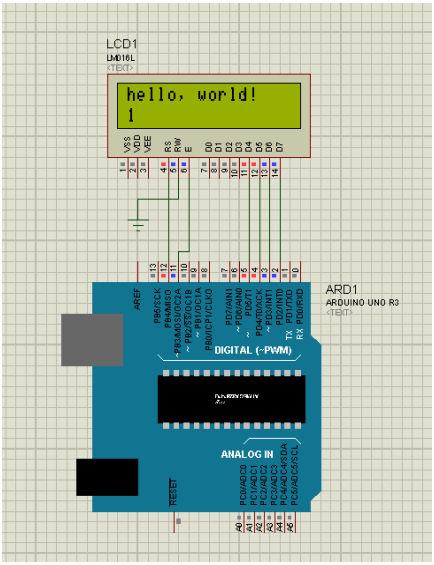


Рисунок 16.5 – Результати симуляції.

17 ПРОЕКТУВАННЯ ДРУКОВАНОЇ ПЛАТИ З ВИКОРИСТАННЯМ СРЕДОВИЩА PROTEUS ARES

17.1 Проектування схеми таймера 555

555 це серія таймера, яка стала однією з перших інтегральних мікрозбірок. Вона об'єднує в собі близько 20 транзисторів і використовується для роботи в двох режимах: у режимі безпосередньо таймера і генератора прямокутних імпульсів.

На 555 серії існує величезна кількість схем, як для новачків радіоаматорів, так і для фахівців. На основі цього таймера можна створити саморобні сигналізації, датчики, сирени, генератори, перетворювачі напруги, підсилювачі потужності звукової частоти і т.п.

У цьому розділі буде створюватися приклад таймера, який генерує імпульси з заданою частотою. Діапазон частот, що генерується таймером, досить широкий: від найнижчої частоти, період якої може досягати кількох годин, до частот в кілька десятків кілогерц.

Відкриваємо Proteus ISIS і починаємо створювати проект таймера. Виберіть з бібліотеки елементи (рис. 17.1).

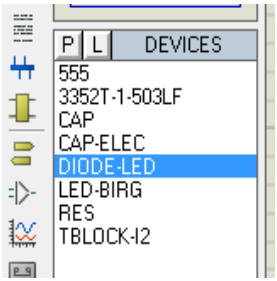


Рисунок 17.1 – Вибір елементів з бібліотеки Proteus

Спроектуйте схему, як показано на рисунку 17.2 і виконайте симуляцію її роботи. Світлодіод буде відображати появу імпульсу на

виході таймера. Після того, як переконаєтесь, що схема працює замініть світлодіод (DIODE-LED) та встановіть рознім (TBLOCK-I2) (рис.17.3).

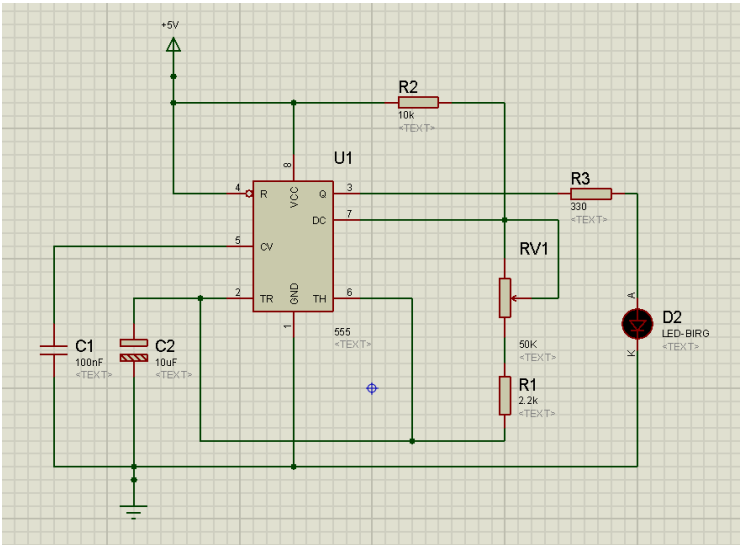


Рисунок 17.2 – Початкова схема для експерименту

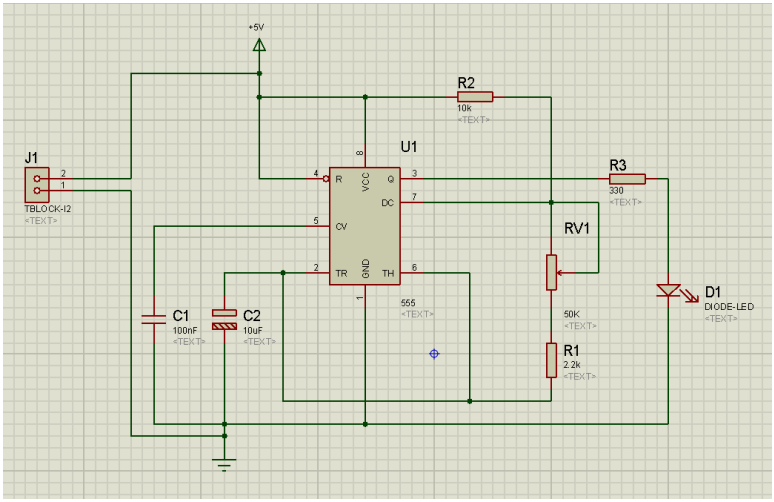


Рисунок 17.3 – Перетворена схема для експерименту

17.2 Проектування друкованої плати для таймера 555

Перевіримо наявність посадкових місць (footprint) для всіх елементів плати. Для цього подвійним кліком натискаємо на елементі і в рядку PCB Footprint дивимося, щоб була вказана бібліотека, в якій знаходиться цей footprint.

Якщо в полі PCB Footprint написано (Not specified), як показано на рисунку 17.4, тоді натискаємо на «знак питання».

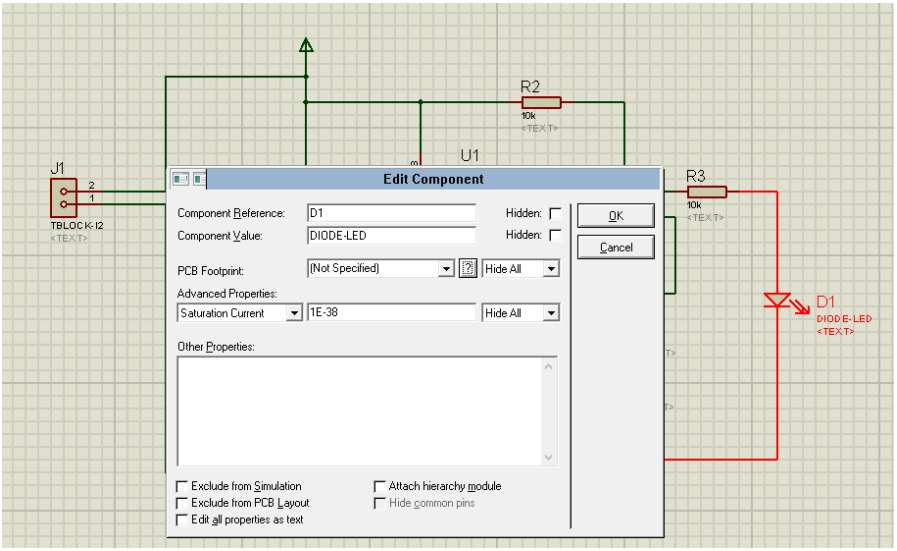


Рисунок 17.4 - Перевірка наявності посадкових місць

У вікні (рис.17.5) вводимо ключове слово **led**, за яким буде виконуватись пошук всіх посадкових місць в усіх бібліотеках, підключених до Proteus. Вводимо в поле ключових слів елемент **LED.** Клікаємо **ОК**.

Далі в електричній схемі клікаємо ПКМ по елементу світлодіода, і з випадаючого списку вибираємо **Packaging Tool** (рис.17.6). У вікні **ISIS Information** клікаємо **ОК** (рис.17.7).

У вікні Package Device вказуємо відповідність виводів світлодіода і пінів посадкового місця, як показано на рисунку 17.8. Після натискаємо **Assign Package(s)**

У вікні що відкрилось (рис.17.9) обираємо **Save Package(s)**. YES

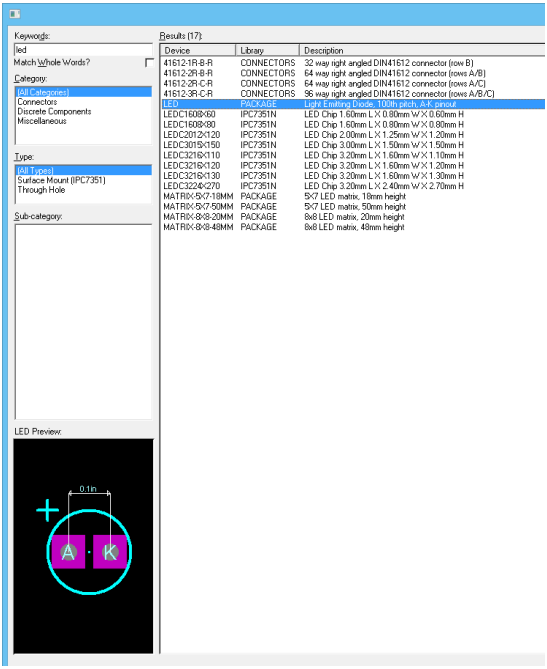


Рисунок 17.5 – Вікно пошуку посадкових місць.

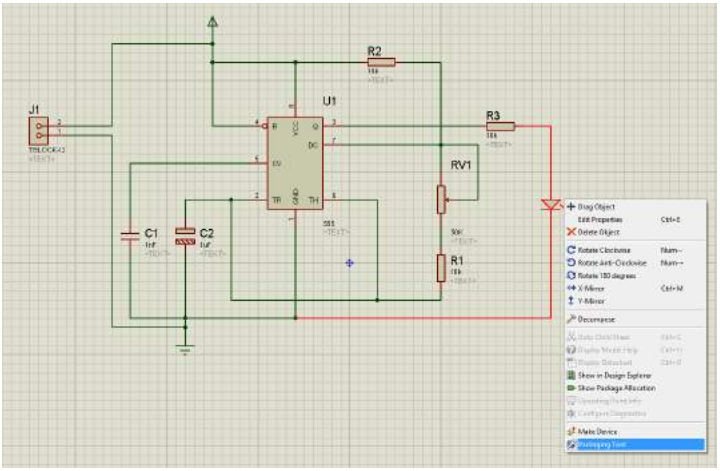


Рисунок 17.6 – Вибір світлодіода на схемі

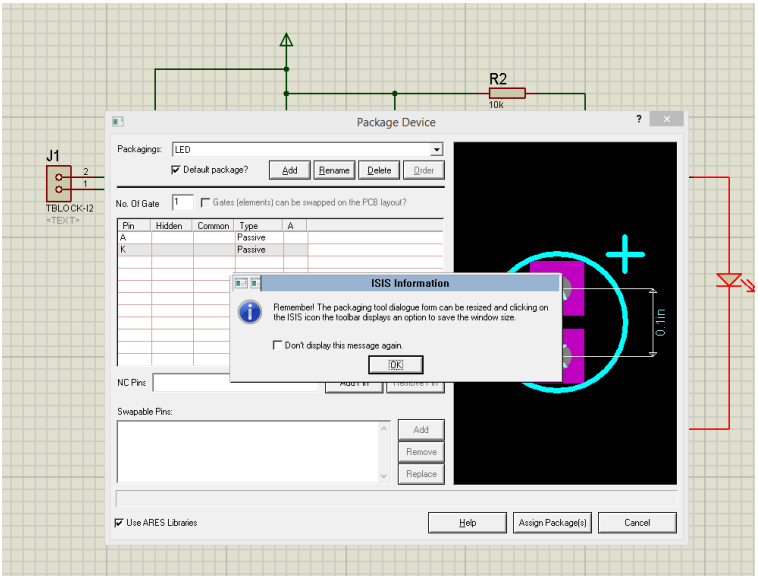


Рисунок 17.7 – Робота з інструментом Packaging Tool

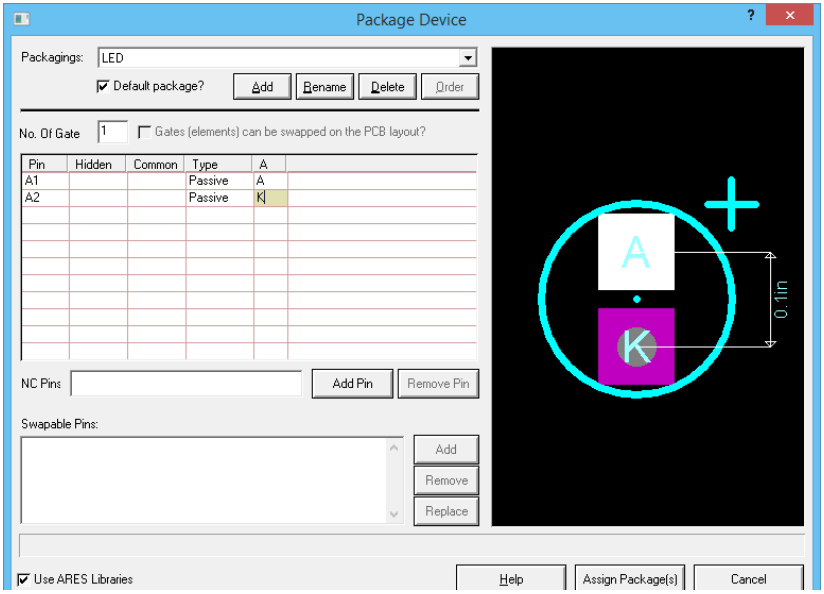


Рисунок 17.8 - Вікно Package Device

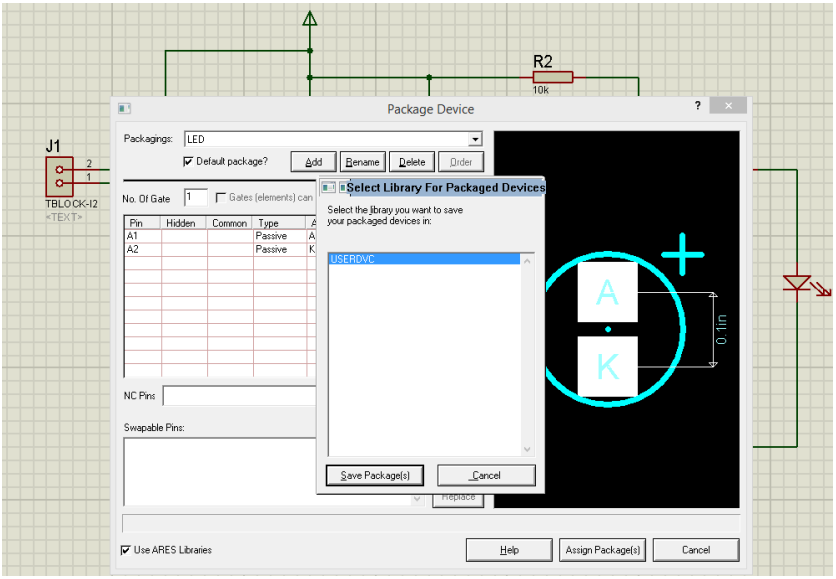


Рисунок 17.9 – Вікно вибору бібліотеки

На верхній панелі інструментів натискаємо **Tools->Netlist to ARES** (рис. 17.10). У лівій панелі інструментів обираємо **Component Mode.**

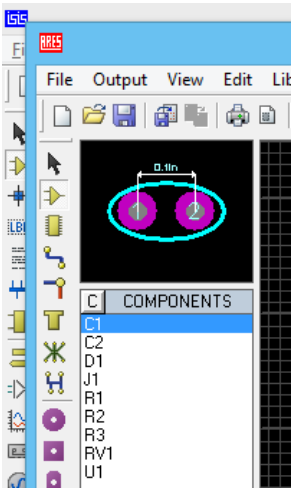


Рисунок 17.10 – Вікно ARES

Щоб задати контури самої плати в лівій панелі інструментів клікаємо по елементу **2D Graphics Box Mode** і після цього в нижній панелі інструментів обираємо **Board Edge** (рис. 17.11).

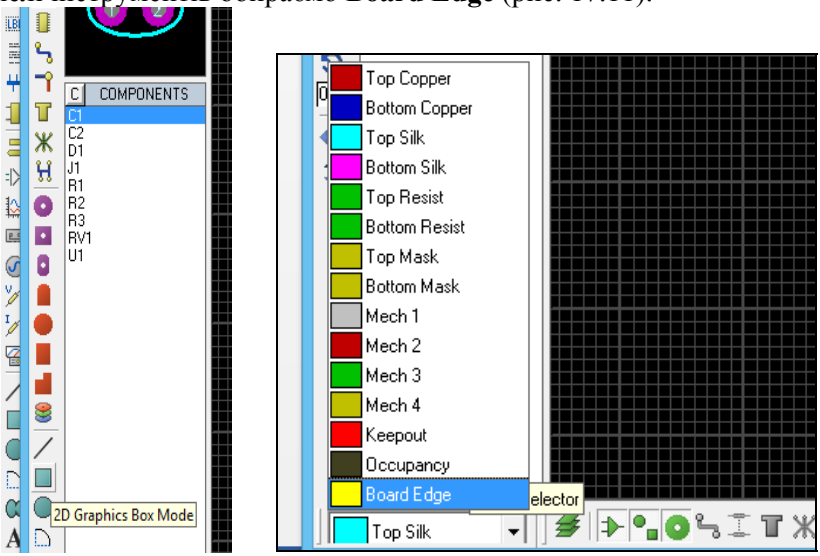


Рисунок 17.11 – Початок створення плати

Креслимо невеликий прямокутник (рис.17.12). Це у нас буде сама плата, на якій ми будемо розміщувати посадкові місця елементів і виконувати трасування провідників.

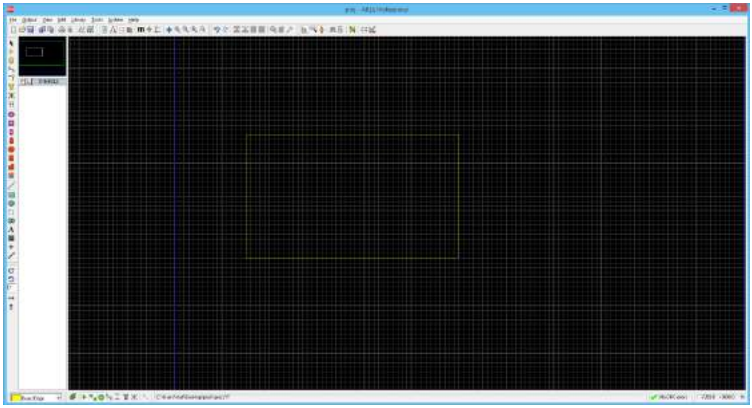


Рисунок 17.12 –Малювання контуру плати

Щоб подивитись, як вона буде виглядати в 3D, обираємо **Output -> 3D Visualisation** (рис.17.13). Далі, по черзі розміщуємо компоненти, як зображено на рисунку 17.14.

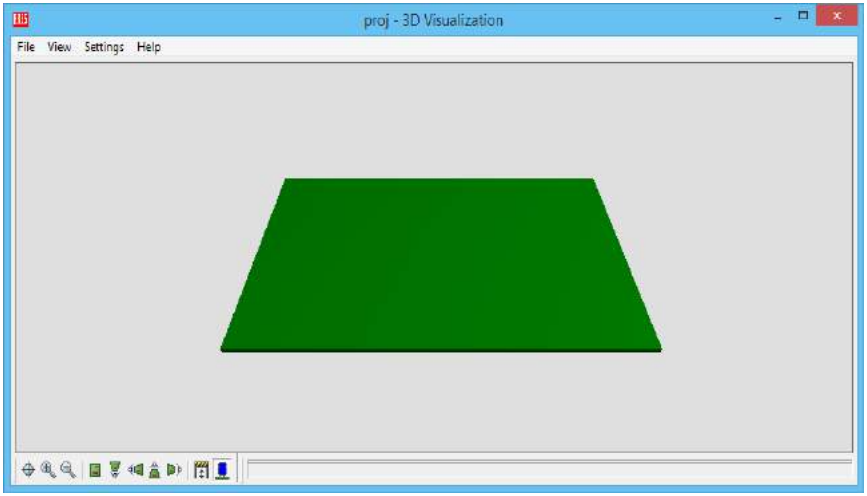


Рисунок 17.13 – 3D модель плати

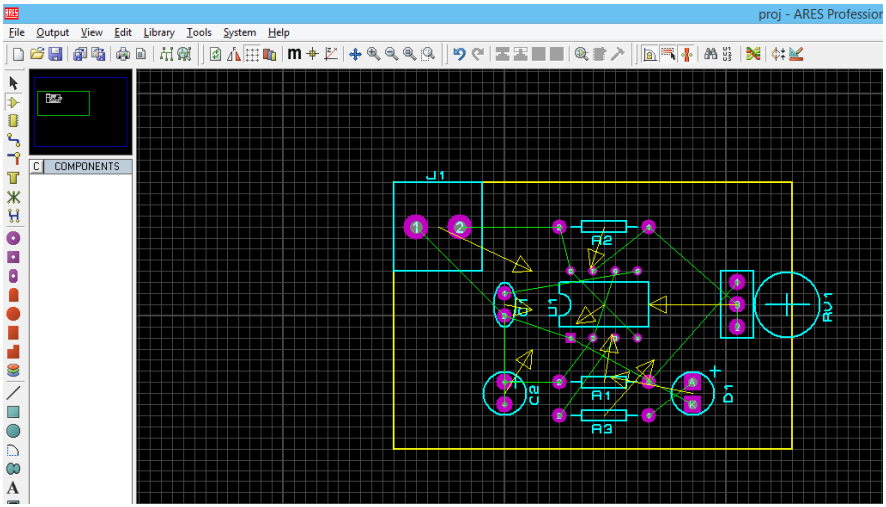


Рисунок 17.14 – Розміщення компонентів на платі

У верхній панелі інструментів вибираємо іконку і встановлюємо параметри поля **Pair 1**, як показано на рисунку 17.15, в **Bottom Copper**. Що означає розміщення провідника на нижній поверхні плати. Після, в цьому ж вікні в полі **Net class** встановлюємо значення **SIGNAL** а також змінюємо значення Pair 1 на Bottom Copper. Клікаємо ОК.

У лівій панелі інструментів обираємо **Track Mode.** Подвійним кліком по DEFAULT. У вікні змінюємо значення Width на 30th (рис.17.16). Клікаємо ОК.

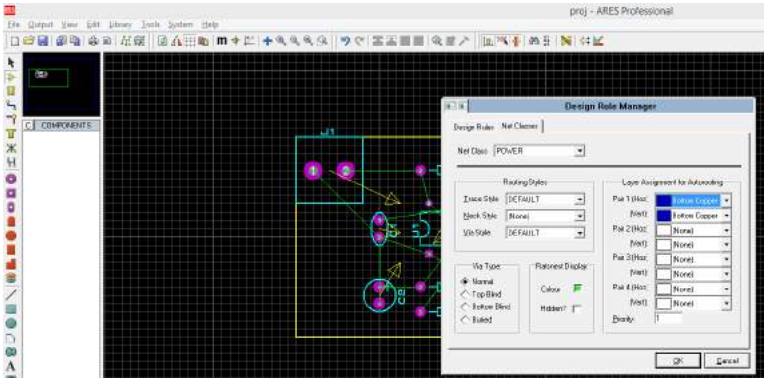


Рисунок 17.15 – Поверхня для розміщення провідника на платі

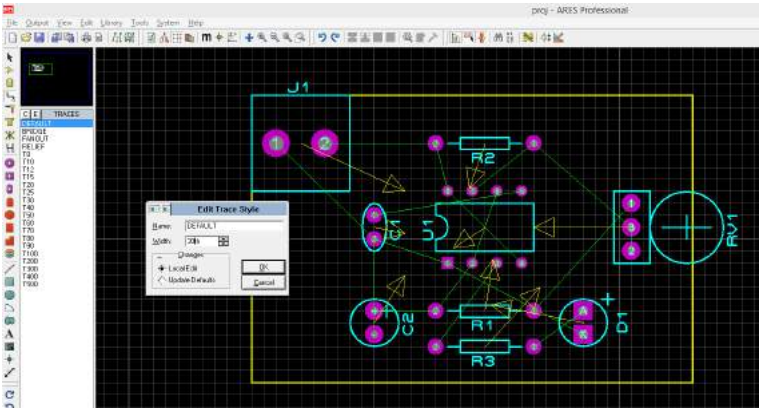


Рисунок 17.16 – Задання ширини провідників

Далі розведемо провідники на платі засобами ARES. Програма пропонує автоматично обчислити і зробити трасування провідників (але також можна це робити вручну). У верхній панелі інструментів натискаємо **Auto-router**. У вікні, натискаємо **Begin Routing**. Клікаємо OK і система сама автоматично генерує нам трасування провідникового шару (рис.17.17).

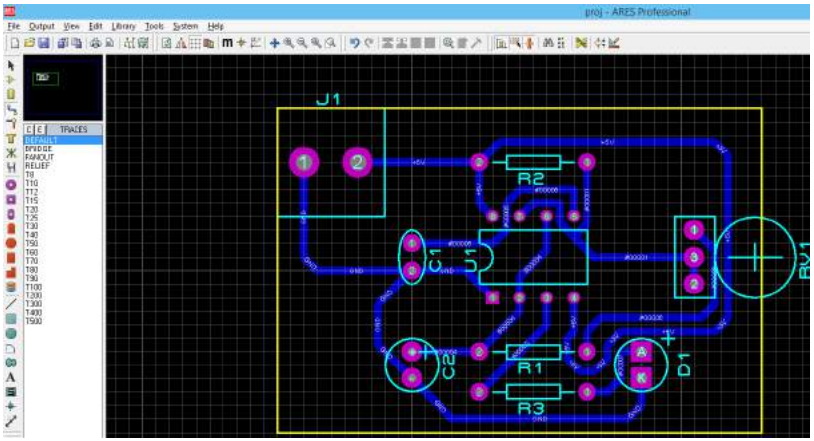


Рисунок 17.17 – Згенеровані провідники

Обираємо **Tools -> Power Plane Generator…** Змінюємо значення у вікні (рис.17.18). Клікаємо **OK**.

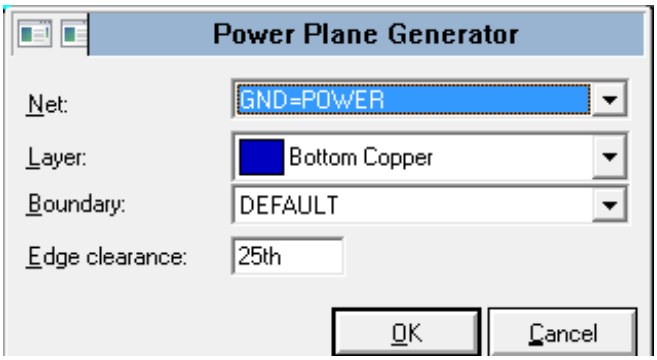


Рисунок 17.18 – Зміна параметрів

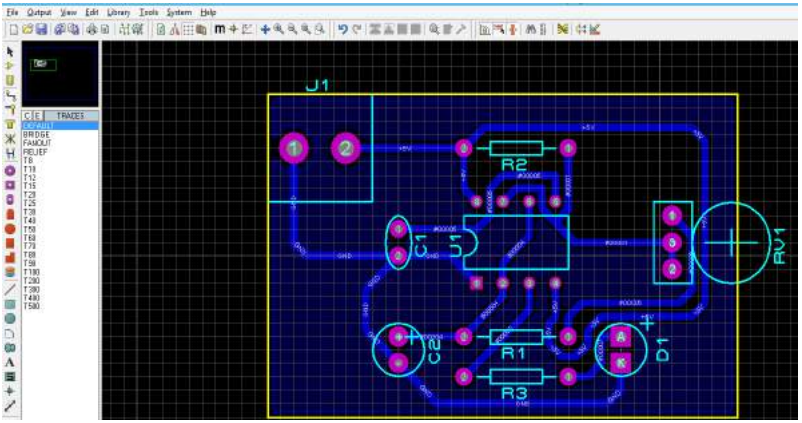


Рисунок 17.19 – Остаточний вигляд провідникового шару.

Для перегляду візуалізації плати і перегляду описаних доріжок вибираємо **Output -> 3D Visualisation** (рис.17.20).

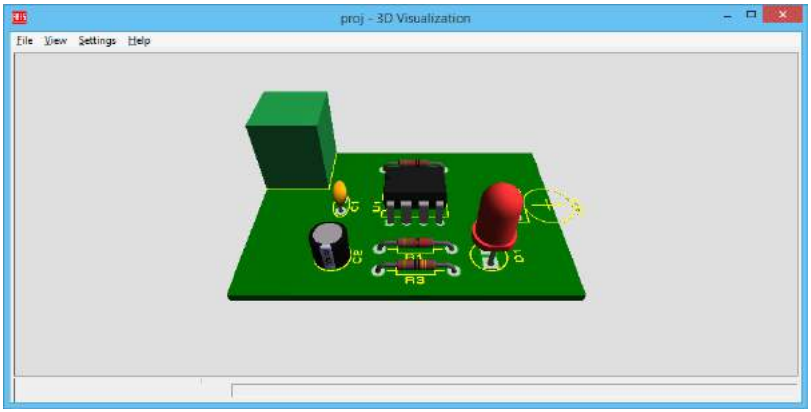


Рисунок 17.20 - 3D модель плати

Для створення і витравлювання провідників плати в реальних умовах для розробника необхідне чорно-біле зображення провідникого шару. Proteus дозволяє це отримати: вибираємо **Output -> Print**. Відкривається вікно (рис.17.21). Клікаємо на кнопку **Printer** та обираємо налаштування друкування, як зображено на рисунку 17.22.

Після цього, змінюємо деякі налаштування в самому вікні **Print Layout** (рис.17.23)

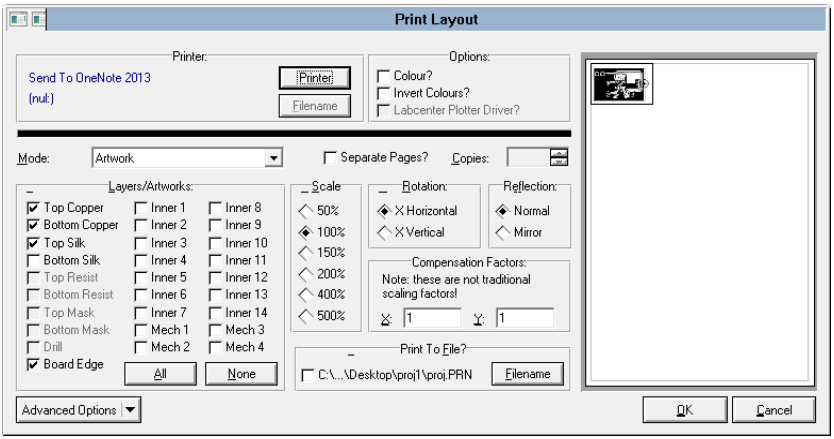


Рисунок 17.21 – Налаштування Крок 1.

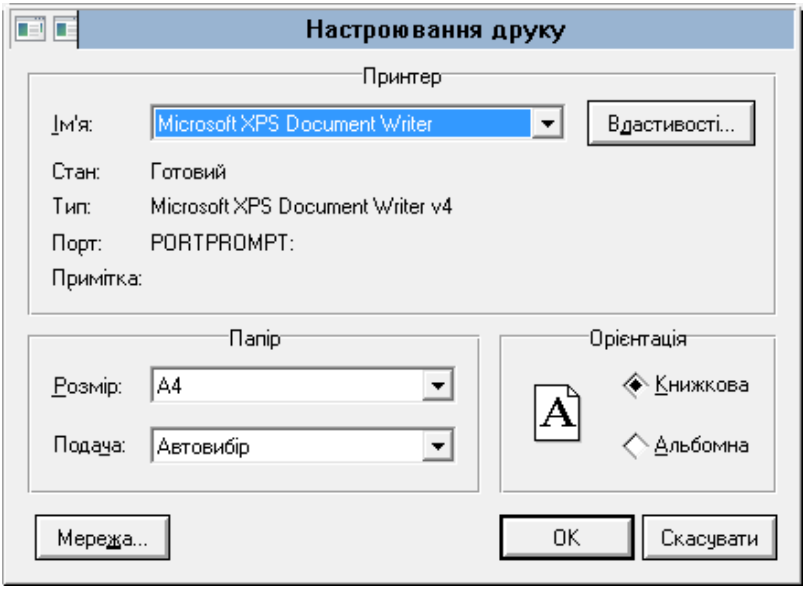


Рисунок 17.22 – Налаштування Крок 2.

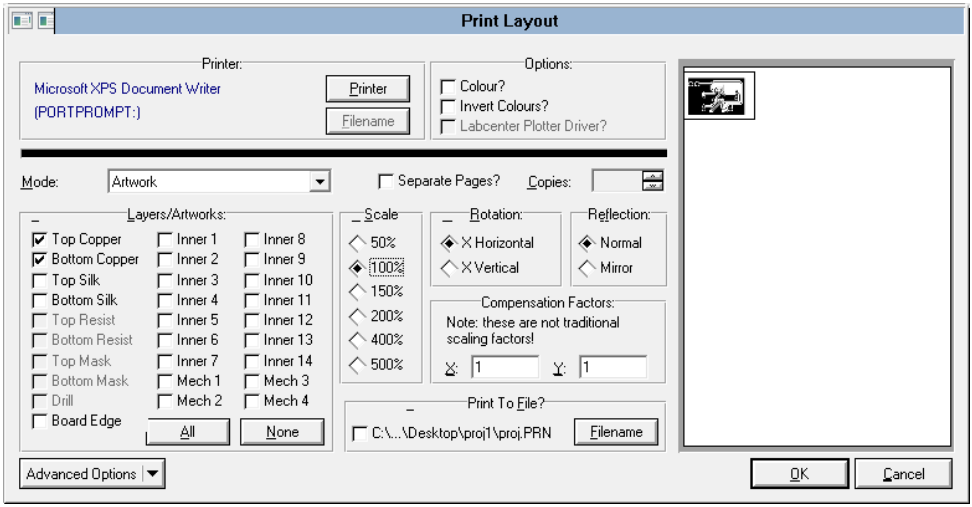


Рисунок 17.23 – Налаштування Крок 3.

Натискаємо **OK** та зберігаємо файл у форматі ***.xps.** Далі, встановимо розміри отриманої плати. Переходимо на лівій панелі інструментів в режим Dimension Mode і малюємо лінії, які дорівнюють ширині і довжині плати.

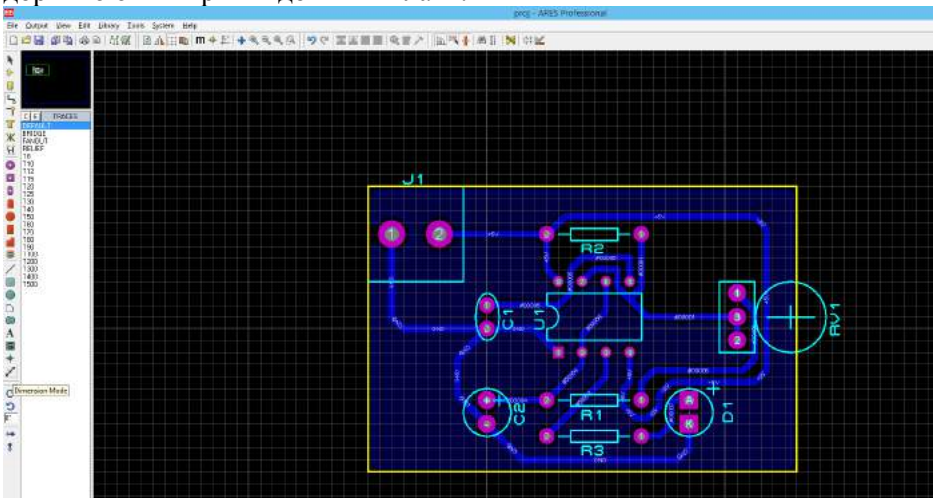


Рисунок 17.24 – Налаштування Крок 3.

Щоб змінити одиниці вимірювання клікаємо ПКМ на розмірі та з випадаючого списку обираємо **Edit Properties…**(рис.17.25). У вікні, що відкрилося змінюємо значення, як показано на рисунку 17.26. Готовий проект на рис. 17.27.

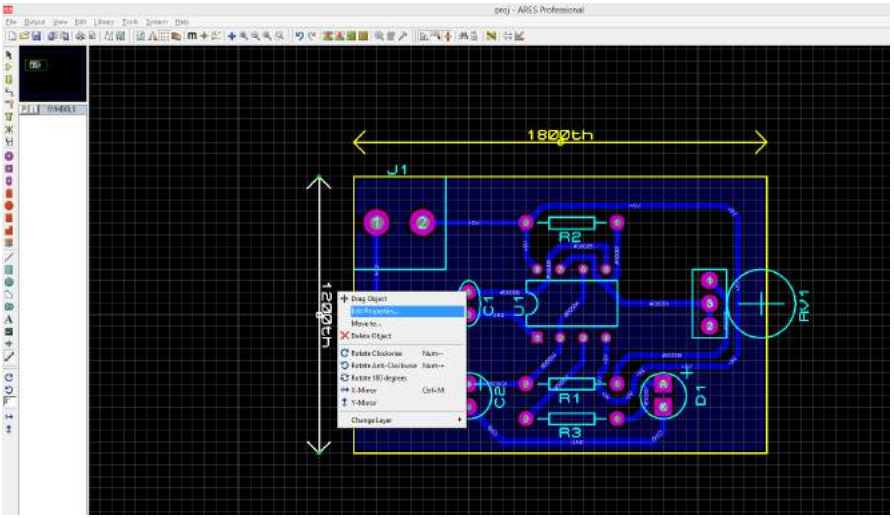


Рисунок 17.25 – Виклик вікна зміни властивостей

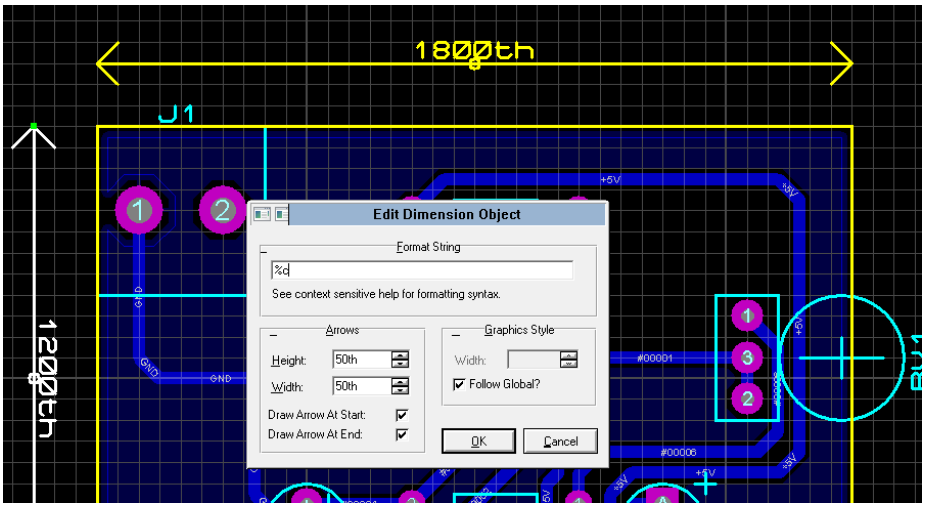


Рисунок 17.26 – Зміна розмірів

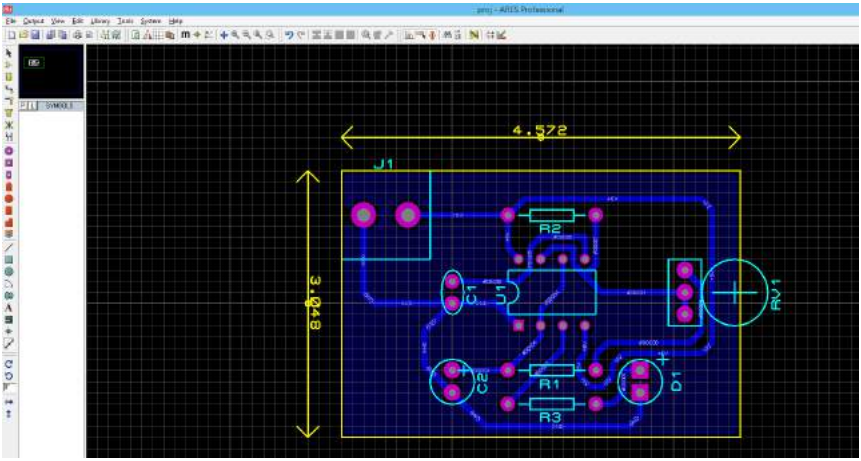


Рисунок 17.27 – Готовий проект друкованої плати

17.3 Практична робота зі створення проекту друкованої плати

Завдання 1. Спроектуйте друковану плату у середовищі Proteus згідно зі схемою з розділу 12 (мікроконтролер Atmega8, 3 світлодіоди та 4 резистори) та розділу 15 (реле з логічним перемикачем, замість перемикача встановіть мікроконтролер). Виконайте трасування провідників вручну, задаючи напрям, ширину провідника, та поверхню, на якій він буде знаходитись.

Завдання 2. Виконайте ті самі проекти в середовищі Altium Designer.

18 РОЗРОБКА ІНТЕРАКТИВНОГО ГРАФІЧНОГО ІНТЕРФЕЙСУ ДЛЯ ВЗАЄМОДІЇ З ПЛАТФОРМОЮ ARDUINO

Інтегроване середовище розробки Arduino IDE є кросплатформним додатком, який написаний на мові програмування Java. Воно виникло з IDE для мов Processing [41] і Wiring [42]. Як і Wiring, Arduino була створена з метою спрощення програмування мікроконтролерів. Тобто, щоб користувачі, які не мають навичок роботи з електронікою могли легко розробити власний проект на базі мікроконтролера та інших датчиків.

Середовище Processing IDE навпаки було створене з метою дати можливість студентам технічних спеціальностей більш простий спосіб роботи з графікою.

Таким чином, оскільки Arduino і Processing IDE тісно пов'язані між собою, а саме, один є нащадком іншого, вони можуть взаємодіяти, що дозволяє створити простий графічний інтерфейс для зручного керування Arduino платформою або навпаки.

18.1 Взаємодія Arduino і Processing

Дані можуть передаватися між ескізом Processing і платою Arduino за допомогою порту послідовної взаємодії. Для роботи з послідовним портом у Arduino і Processing існує бібліотека Serial, в якій містяться функції для спрощення реалізації послідовного обміну даними. Serial - це формат даних, який надсилає один байт за один раз. Для Arduino, байти є тип даних, який може зберігати значення від 0 до 255; він працює як цілочисельний формат даних int, але з набагато меншим діапазоном. Великі номери відправляються з розбиттям в список байтів та наступним їх повторним збиранням.

В нижче наведених прикладах ми реалізуємо прості програми для демонстрації двонаправленого зв'язку з Arduino до Processing і в зворотньому напрямку.

Крім того, існує можливість безпосередньо з Processing IDE спілкуватися та задавати команди платформі Arduino, для цього використовується протокол Firmata, але тут ми не будемо його розглядати.

18.2 Середовище Processing IDE

Якщо відкрити файл processing.exe, то ви побачите вікно, зображене на рисунку 18.1. Як бачимо, воно абсолютно схоже з середовищем Arduino IDE. Це так зване вікно створення ескізу (sketch), тобто вихідного тексту програми. Воно складається з наступних елементів: Меню, Панель інструментів, Вкладки, Текстовий редактор, Вікно повідомлень та Консоль.

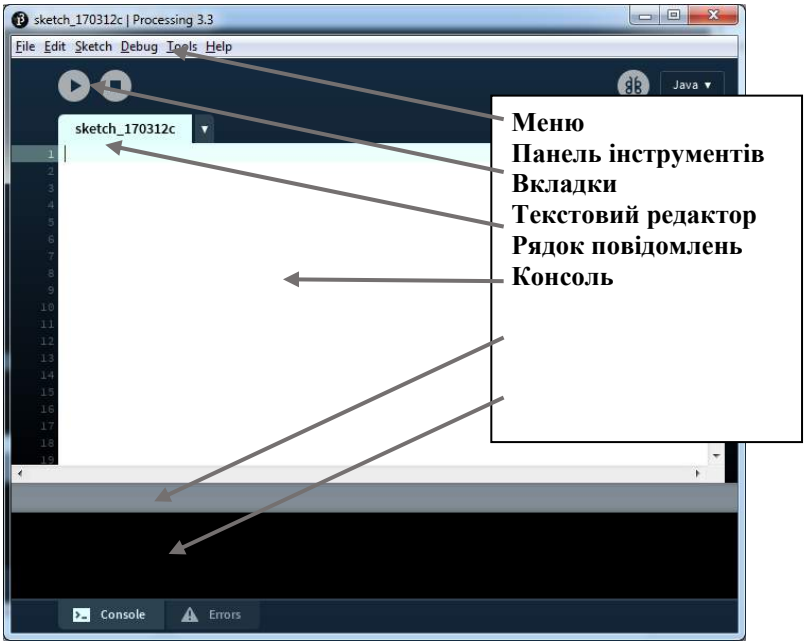


Рисунок 18.1 –Інтерфейс Processing

Мова програмування Processing є діалектом Java, але з деякими особливостями для роботи з графікою та інтерактивним наповненням. Processing багато чого взяв з PostScript (що послужило основою для формату PDF) та OpenGL (специфікація 3D графіки).

Для того, щоб розпочати роботу, у вас повинні бути знання щодо основ утворення RGB кольорів та розуміння пікселів. Також, ви повинні вміти працювати з Arduino, оскільки деталі роботи з нею в цьому розділі не описуються.

Щоб ознайомитись з можливостями та побудувати першу програму треба оволодіти наступними базовими функціями перерахованими нижче (для більш детального вивчення синтаксису мови та принципів роботи з Processing дивіться [41,43]):

void setup() та *void draw()* - основні функції роботи програми, усі інші функції пишуться усередині цих двох циклів; void setup() виконується один раз на початку роботи програми; void draw() – це функція нескінченного циклу, вона виконується постійно;

size(X, Y) -створює робочу область (або вікно) розміром X на Y пікселів, за замовчуванням вона 100 на 100 пікселів;

background(a) - задає колір фону; background(a) еквівалентно запису background(a,a,a), де три змінні відповідають основним кольорам колірної моделі RGB: червоному, зеленому та синьому;

fill(R, G, B) - вказує колір заповнення фігури, можна додати четвертий параметр до цієї функції T - це прозорість, за замовчуванням, якщо не вказувати цей параметр то він дорівнює 255, тобто 100% не прозоре;

rect(x, y, w, h) - малює прямокутник, починаючи з лівої верхньої точки (x, y), з параметрами w – довжина, h – висота, можна також додати п’ятий параметр t – радіус округлення кутів прямокутника.

Відкрийте середовище Processing, скопіюйте програму, яка наведена нижче у прикладі та збережіть програму на диску комп’ютера (**File -> Save**).

Приклад для Processing:

```
import processing.serial.*;

Serial myPort;  // створює об’єкт класу Serial
int val;  // змінна для збереження даних, отриманих від Arduino

void setup()
{
 size(200, 200);
 String portName = Serial.list()[0];
 myPort = new Serial(this, portName, 9600);
}

void draw()
{
```

```
  if ( myPort.available() > 0) {  // якщо порт відкритий,
    val = myPort.read();         // читати дані з нього і зберегти у val
  }
  background(255);             // встановити задній фон, як білий
  if (val == 0) {                        // якщо значення, отримане з послідовного порту дорівнює 0,
    fill(128);                 // встановити заповнення, як середнє сіре
  }
  else {                       // якщо значення відмінне від 0,
    fill(255, 100, 0);         // встановити заповнення, як помаранчеве
  }
  rect(50, 50, 100, 100);    // прямокутник розміром 100 на 100
}
```

Далі, підключить кнопку до плати Arduino, як зображено на рисунку 18.2.

Скопіюйте код, наведений у прикладі нижче, відкомпілюйте та завантажте програму до плати Arduino.

Приклад для Arduino:

```
int switchPin = 2;                  // кнопка підключена до 2 піну

void setup() {
  pinMode(switchPin, INPUT);            // режим кнопки на вхід
  Serial.begin(9600);                   // розпочати послідовний зв'язок зі швидкістю 9600 біт за секунду
}

void loop() {
  if (digitalRead(switchPin) == HIGH) {  // якщо кнопка замкнена,
    Serial.write(1);              // відправка 1 до Processing
  } else {                          // якщо кнопка розімкнена,
    Serial.write(0);              // відправка 0 до Processing
  }
  delay(100);                       // затримка 100 мілісекунд
}
```

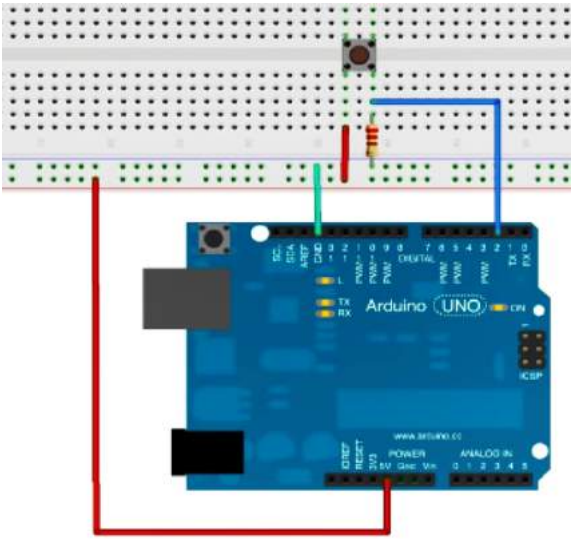
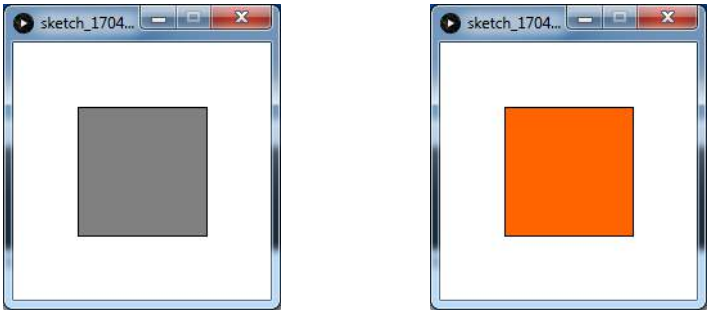


Рисунок 18.2 – Схема підключення кнопки до Arduino

Знову відкрийте Processing зі збереженою вище програмою та клікніть на панелі інструментів кнопку **Run (Запуск)**, це запустить компіляцію програми.

Якщо в вас декілька пристроїв, підключених до послідовного порту ПК, то можливо при компіляції вам буде висвічуватись помилка стосовно послідовного порту. У цьому випадку спробуйте змінити значення у String portName = Serial.list()[0]; з [0] на [1], або [2], це залежить від того, скільки пристроїв підключено по ПК.

У результаті ви побачите вікно відображення роботи Processing, в якому намальований квадрат розмірами 100 на 100 пікселів rect(50, 50, 100, 100); верхній лівий кут якого має координати (50, 50), він середнього сірого кольору fill(128) (рис.18.3а). При натисканні кнопки квадрат буде змінювати свій колір з середнього сірого на помаранчевий fill(255, 100, 0) (рис.18.3б).



а) б)

Рисунок 18.3 – Результат роботи Processing

У другому прикладі ми розглянемо ситуацію, коли навпаки Processing через послідовний порт буде не читати, а відправляти дані до плати Arduino. Підключіть світлодіод до плати Arduino, як зображено на рисунку 18.4.

Скопіюйте код, наведений у прикладі нижче, відкомпілюйте та завантажте програму до плати Arduino.

Приклад для Arduino:

```
char val; // дані, які отримуються з послідовного порту
 int ledPin = 10; // ініціалізація 10 піна плати

 void setup() {
 pinMode(ledPin, OUTPUT); // режим роботи піна на вихід
 Serial.begin(9600); // розпочати обмін даними з 9600 біт/сек.
 }

 void loop() {
 while (Serial.available()) { // якщо порт доступний
 val = Serial.read(); // читати дані та записувати в val
 }
 if (val == 'H') { // якщо було отримано H
 digitalWrite(ledPin, HIGH); // LED загоряється
 } else {
 digitalWrite(ledPin, LOW); // в іншому разі вимкнений
 }
 delay(100);
 }
```

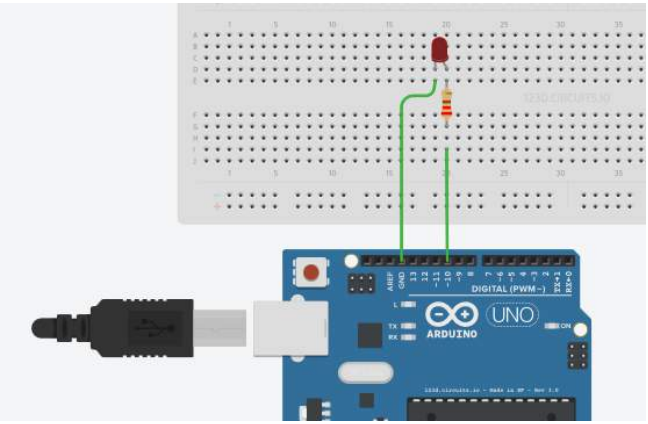


Рисунок 18.4 – Схема підключення світлодіоду до Arduino

Після завантаження програми до Arduino, відкрийте середовище Processing, скопіюйте наведений нижче код та запустіть програму на виконання.

Приклад для Processing:

```
import processing.serial.*;

Serial myPort;  // створення об'єкту класу Serial
int val;        // дані, які відправляються до Arduino

void setup()
{
 size(200, 200);
 String portName = Serial.list()[0];
 myPort = new Serial(this, portName, 9600);
}

void draw() {
 background(255);
 if (mouseOverRect() == true) {  // якщо курсор над квадратом,
  fill(255, 100, 0);                  // змінити колір
  myPort.write('H');          // та записати H до послідовного порту
 }
 else {                       // якщо курсор поза квадратом
```

```
    fill(128);                  // змінити колір
    myPort.write('L');          // та відправити L
   }
   rect(50, 50, 100, 100);
  }

  boolean mouseOverRect() { // функція перевірки курсора миші
   return ((mouseX >= 50) && (mouseX <= 150) && (mouseY >=
50) && (mouseY <= 150));
  }
```

Після запуску програми, можна спостерігати вікно відображення роботи Processing з таким самим квадратом середнього сірого кольору, як і в попередньому прикладі. Але у цьому прикладі якщо курсор миші наведений над прямокутником вікна відображення роботи Processing то колір прямокутника змінюється на помаранчевий та до послідовного порту надсилається символ ‘H’. Arduino в свою чергу зчитує значення з послідовного порту та включає світлодіод. У всіх інших випадках світлодіод залишається вимкненим.

18.3 Практична робота з використання середовища Processing IDE

Треба розробити програму для Arduino, яка буде читати з аналогового піну Arduino значення положення потенціометра, ділити його на 4, та відправляти отримане значення до послідовного порту, яке далі буде надсилатися до Processing, та в залежності від значення буде змінюватися ширина прямокутника. Зміна довжини прямокутника буде нагадувати рух повзунка. На 4 необхідно ділити для того, щоб отримати значення рівне 256, оскільки ширина вікна відображення Processing у прикладі, що наведений нижче, встановлена від 0 до 255.

Створіть новий Processing ескіз “DisplayPotentiom.pde”, скопіюйте текст, наведений нижче та збережіть на комп’ютері. Якщо у вас не підключена плата Arduino до комп’ютеру, то програма Processing при запуску буде видавати помилку відкриття послідовного порту, оскільки чекає на значення з послідовного порту і в залежності від цього значення виконується програма. Тільки після того, як ви

розробите програму для Arduino, підключите її до ПК та завантажите код, можна запускати програму для Processing.

Приклад для Processing:

```
import processing.serial.*; // підключається бібліотека serial
Serial arduino;  // створення об'єкту класу Serial
int val;       // дані, які відправляються до Arduino

void setup() {
  size(255, 100);
  arduino = new Serial(this, Serial.list()[0], 57600);
}

void draw () {
  if ( arduino.available() > 0) {  // якщо порт відкритий
    val = arduino.read();         // читати дані з нього і зберегти у val
  }
  background(128); // колір фону
  fill(255); // колір тексту білий
  text("Value", 10, 15); // напис Value
  stroke(255);  // колір лінії
  line (0, 20, 255, 20);  // лінія для розмітки
  for (int i = 0; i < 255; i = i+10) { // риски через 10 пікселів
    stroke(255); // колір рисок
    line(i, 20, i, 23); // малюємо риски
  }
  fill(250); // колір повзунка
  noStroke();  // без контуру прямокутник
  rect(0, 25, val, 10); // рисуємо прямокутник (повзунок), який змінює свою ширину в залежності від отриманого значення у val
}
```

Створіть Arduino ескіз який:

- встановлює UART на швидкість передачі 57600 біт/секунду;
- читає значення з аналогового піну, до якого підключений потенціометр, ділить це значення на 4;
- записує їх до послідовного порту для того, щоб прочитав Processing.

Після того, як ви розробите коректну програму для потенціометра, скомпілюєте та завантажите її до Arduino, ви можете запустити програму для Processing та побачити результат, зображений на рисунку 18.5.

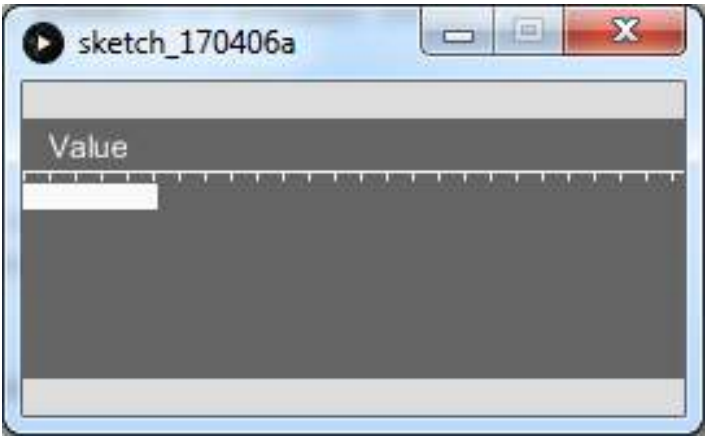


Рисунок 18.5 – Інтерфейс взаємодії з потенціометром

В залежності від положення ручки потенціометра, положення повзунка буде також змінюватися.

ПИТАННЯ ДЛЯ САМОПЕРЕВІРКИ ДО ЧАСТИНИ 2

1. Дайте визначення закону Ома?
2. Призначення, різновиди та параметри резисторів.
3. Розрахунок опору для паралельного та послідовного підключення резисторів у колі.
4. З якої бібліотеки ви використовували елементи для побудови електричної схеми у середовищі Proteus?
5. В якому режимі у середовищі Proteus, відбувається пошук елементів для схеми?
6. Призначення, різновиди та параметри світлодіодів?
7. Призначення та правила використання вимірювальних пристроїв в електроніці.
8. В чому різниця між постійним і змінним струмом/напругою?
9. Функціональні можливості сервісу 123D Circuits.
10. У якому режимі здійснюється макетне моделювання в сервісі 123D Circuits?
11. До якого виводу підключено вбудований світлодіод на платформі Arduino Uno?
12. Які оператори середовища Arduino IDE треба використовувати, щоб створити проект керування світлодіодами?
13. Що таке RX та TX на платі Arduino?
14. Скільки усього входів/виходів має платформа Arduino Uno? Назвіть основні з них.
15. Що таке Hex-файл?
16. Назвіть послідовність етапів створення нового проекту в середовищі Atmel Studio.
17. Яке сімейство МК використовувалося при розробці схеми у розділі 12?
18. При розробці керуючої програми, у яких одиницях вимірювання задається затримка виконання команди МК?
19. Як працює потенціометр?
20. До яких виводів підключається потенціометр в схемі керування на платі Arduino?
21. Які виводи має RGB світлодіод?
22. Які функції Arduino використовуються для управляння кольором RGB світлодіоду?

23. З якою метою в схемі підключення кнопки до Arduino Uno використовуються підтягуючі резистори?

24. Що таке сервопривід і які різновиди сервоприводів існують?

25. За якими ознаками відрізняються сервоприводи?

26. Як відбувається управління сервоприводом керуючими сигналами?

27. До яких виводів підключається сервопривід до платформи Arduino?

28. Назвіть основні команди Arduino для керування сервоприводом?

29. Що таке реле? Як побудовано реле?

30. Які бувають комутативні контакти реле?

31. Принципи роботи біполярного транзистора?

32. Основні функції Arduino для роботи з дисплеєм?

33. Які піни на платі Arduino використовуються для створення проекту роботи з дисплеєм?

34. Що таке LCD I2C модуль?

35. Що таке Footprint?

36. Назвіть етапи створення друкованої плати в Proteus?

37. Як відбувається процес виготовлення друкованої плати?

38. Які параметри вікна відображення середовища Processing встановлені за замовчуванням?

39. Якими будуть значення функції заповнення fill для чорного, білого та середнього сірого кольору?

40. Який з нижче наведених еліпсів, створених в середовищі Processing буде найпрозорішим?

```
fill(255, 0, 0, 90);
ellipse(100, 100, 150, 75);
fill(0, 0, 0, 100);
ellipse(50, 50, 120, 60);
fill(0, 100, 0, 200);
ellipse(50, 80, 60, 60).
```

ПЕРЕЛІК ПОСИЛАНЬ

1. Ebert, C. Software: Facts, Figures, and Future/ Christof Ebert, Capers Jones//Computer, vol. 42, no. 4, pp. 42–52, April 2009, doi:10.1109/MC.2009.118
2. Embedded Systems: Technologies and Markets [Електронний ресурс] / BCC Research.– Режим доступу: www/ URL: http:// www.bccresearch.com/market-research/information-technology/ embedded-systems-technologies-markets-ift016d.html
3. Embedded System Market – Global Industry Analysis, Size, Share, Growth, Trends and Forecast, 2012–2018 [Електронний ресурс] / Transparency Market Research.– Режим доступу: www/ URL: http://www.transparencymarketresearch.com/embedded-system.html
4. Global market Insights [Електронний ресурс] / Industry Analysis – Режим доступу: https://www.gminsights.com/industry-analysis/embedded-system-market
5. Платунов А. Встраиваемые системы управления /Платунов А. // CONTROL ENGINEERING РОССИЯ. 2013.– № 1 (43) .– с. 16–24
6. Барретт С.Ф. Встраиваемые системы. Проектирование приложений на микроконтроллерах семейства 68HC12/HCS12 с применением языка С. / С.Ф. Барретт, Д.Дж. Пак – Издательский дом «ДМК-пресс», 2007 – 643с.
7. Быковский С.В. Сопряженное проектирование встраиваемых систем (Hardware/Software Co-design). Часть 1: учеб.пособие /
8. Электронные компоненты [Електронний ресурс] / Ежегодное исследование рынка встраиваемых систем – Режим доступу: http://www.elcomdesign.ru/reviews/reviews_146.html
9. Teich, J., Hardware/Software Codesign: The Past, the Present, and Predicting the Future, Proceedings of the IEEE, Vol.100, pp. 1411-1429, Germany, May 13, 2012.
10. Wolf H., Wayne, Hardware-Software Co-Design of Embedded Systems, Proceedings of the IEEE, Vol.82, NO. 7, pp. 967-989, Germany, July 1994.
11. Два подхода к реализации ПО для embedded [Електронний ресурс]/ Программинг микроконтроллеров. – Режим доступу: www/ URL: http://habrahabr.ru/post/148805/
12. Abdurohman, M., Kuspriyanto, Sutikno, S., Sasongko, A., The New Embedded System Design Methodology For Improving Design

Process Performance, International Journal of Computer Science and Information Security, Vol. 8, No. 1, pp. 35-43, 2010.
13. Parkhomenko, A. Analysis and application of existent approaches in microcontroller system designing / Anzhelika Parkhomenko, Olga Gladkova // Proceedings of IX-th International Conference «PERSPECTIVE TECHNOLOGIES AND METHODS IN MEMS DESIGN (MEMSTECH 2013)», Lviv: Lviv Polytechnic , 2013. - P.268-270.
14. Parkhomenko A. Complex requirements analysis for the high-level design of Embedded Systems / A. Parkhomenko, O. Gladkova // Вісник НУ «Львівська політехніка» «Комп'ютерні системи проектування. Теорія і практика». – Львів: Львівська політехніка, №808(2015). – С.3-10
15. Parkhomenko A. Reusable Solutions for Embedded Systems' Design / A.Parkhomenko, O. Gladkova, A. Sokolyanskii, V. Shepelenko, Y. Zalyubovskiy // Proceedings of XIII International Conference on Remote Engineering and Virtual Instrumentation (REV2016) (24-26 February, 2016, Madrid, Spain) UNED: 2016.- P.313-317
16. Parkhomenko A. Investigation of Reuse Concepts for Embedded Systems Design / A.Parkhomenko, O. Gladkova, A. Sokolyanskii, Y. Zalyubovskiy // Proceedings of XII-th International Conference «PERSPECTIVE TECHNOLOGIES AND METHODS IN MEMS DESIGN» (MEMSTECH 2016), (20-24 April, 2016, Lviv-Polyana, Ukraine), Lviv: Lviv Polytechnic, 2016.- pp.78-80.
17. Parkhomenko, A. Implementation of Reusable Solutions for Remote Laboratory Development / A.Parkhomenko, O. Gladkova, A. Sokolyanskii, V. Shepelenko, Y.Zalyubovskiy // iJOE – Volume 12, Issue 07, 2016.- pp. 24-29
18. Parkhomenko A.V. Virtual Tools and Collaborative Working Environment in Embedded System Design / A.V. Parkhomenko, O.N. Gladkova // Proceedings of XI International Conference on Remote Engineering and Virtual Instrumentation (REV2014) (26-28 February, 2014, Porto, Portugal) Porto: Polytechnic, 2014. -P. 91-93.
19. Parkhomenko A. Development and Application of Remote Laboratory for Embedded Systems Design / A.Parkhomenko, O. Gladkova, E. Ivanov, A. Sokolyanskii, S. Kurson// JOE – Volume 11, Issue 3, 2015.- PP.27-31

20. Parkhomenko, A. Internet-based Technologies for Design of Embedded Systems / A. Parkhomenko, O. Gladkova, E. Ivanov, A. Sokolyanskii, S. Kurson// Proceedings of XIII International Conference «The Experience of Designing and application of CAD Systems in Microelectronics (CADSM 2015)»,(24 – 27 February, 2015,Lviv-Polyana,Ukraine)Lviv:Lviv Polytechnic,2015.-P.167-171.
21. Poliakov, M. Hybrid models of studied objects using remote laboratories for teaching design of control systems /M. Poliakov, T.Larionova, G. Tabunshchyk, A. Parkhomenko, K. Henke // iJOE – Volume 12, Issue 09, 2016.- pp.7-13
22. Remote and virtual tools in engineering: textbook/general editorship Dr.Ing.Karsten Henke.–Zaporizhzhya: Dike Pole, 2016.– 250 p.
23. Parkhomenko A. Remote laboratory for teaching of control systems design as an integrated system/ M. Poliakov, K. Henke, T. Larionova, G. Tabunshchyk, // Proceedings of XIII International Conference on Remote Engineering and Virtual Instrumentation (REV2016) (24-26 February, 2016, Madrid, Spain) UNED: 2016. - pp. 333-340.
24. Proteus VSM. Пошаговая отладка [Електронний ресурс]. – Режим доступу: http://we.easyelectronics.ru/CADSoft/proteus-vsm-poshagovaya-otladka.html
25. Autodesk. AutoCAD Electrical [Електронний ресурс].–Режим доступу:http://www.autodesk.ru/products/autocad-electrical/overview
26. Altium [Електронний ресурс] / Altium Designer. – Режим доступу: http://www.altium.com/altium-designer/overview
27. Пархоменко, А.В. Автоматизоване проектування електронних засобів в середовищах Creo та ALTIUM DESIGNER: Навчальний посібник , 2-ге вид./А.В.Пархоменко, А. В.Притула, В. М. Крищук. Запоріжжя: Дике поле, 2016. – 250 с.
28. Сабунин, А.Е. Altium Designer. Новые решения в проектировании. /А.Е Сабунин– Солон Пресс, 2009. – 432с.
29. Суходольский, В. Ю. Altium Designer. Проектирование функциональных узлов РЭС на печатных платах / В. Ю. Суходольский. – СПб.: БХВ-Петербург, 2009. – 480 с.
30. Autodesk [Електронний ресурс] / EAGLE. – Режим доступу: http://www.autodesk.com/products/eagle/overview
31. Autodesk Circuit [Електронний ресурс] / Electronics from beginner to pro. – Режим доступу: https://circuits.io/

32. Официальный сайт Raspberry Pi [Електронний ресурс]. - Режим доступу: www/ URL: www.raspberrypi.org
33. Intel Galileo [Електронний ресурс]. - Режим доступу: https://software.intel.com/ru-ru/iot/hardware/galileo
34. Cyclone FPGAS Overview [Електронний ресурс]. - Режим доступу: http://www.altera.com/products/fpga/cyclone-series/ cyclone / overview.html
35. Arduino. [Електронний ресурс]. – Режим доступу: https://www.arduino.cc/
36. Среда разработки Arduino [Електронний ресурс]. - Режим доступу. - http://arduino.ua/ru/guide/Environment
37. Arduino [Електронний ресурс].- Режим доступу: www/ URL: https://ru.wikipedia.org/wiki/Arduino
38. Atmel Studio. [Електронний ресурс]. Microchip Atmel. – Режим доступу: http://www.atmel.com/tools/atmelstudio.aspx
39. Atmel Studio [Електронний ресурс]/ Atmel. – Режим доступу: www/ URL: http://www.atmel.com/tools/atmelstudio.aspx
40. RELDES [Електронний ресурс]. – Режим доступу: http://swed.zntu.edu.ua
41. Processing. [Електронний ресурс]. – Режим доступу: https://processing.org/
42. Wiring. [Електронний ресурс]. – Режим доступу: http://wiring.org.co/
43. Casey Reas and Ben Fry Getting Started with Processing/ Published by O'Reilly Media, Inc., June 2010. P.209

АЛФАВІТНО-ПРЕДМЕТНИЙ ПОКАЖЧИК

Навчальне видання

Пархоменко Анжеліка Володимирівна,

Гладкова Ольга Миколаївна,

Залюбовський Ярослав Ігорович,

Пархоменко Андрій Валентинович

ІНЖЕНЕРІЯ ВБУДОВАНИХ СИСТЕМ

Навчальний посібник

Технічний редактор Л. А. Рябоконь
Коректор Н. В. Чечеко
Художник А. П. Кондаков

Формат 60x84/16.
Папір офсетний. Гарнітура Times. Друк офсетний.
Підписано до друку 28.03.2017. Ум. друк. арк. 12,78.
Наклад 300 прим.

Видавництво «Дике Поле»
Україна, 69063, м. Запоріжжя, вул. Троїцька, 31-А.
Тел.: (061) 213-75-95; 213-75-05.
Свідоцтво суб'єкта видавничої справи ЗЗ № 004 від 23.08.2001 р.

Електронна адреса dikoepole.zp@gmail.com
Веб-сторінка http://www.dikoepole.zp.ua

Правове забезпечення
Юридична фірма «Щедрин і партнери»

Електронна адреса schedrin1959@gmail.com