МЕТОДИЧНІ ВКАЗІВКИ
ДО ЛАБОРАТОРНИХ РОБІТ З ДИСЦИПЛІНИ
«ЕЛЕКТРОНІКА ТА МІКРОПРОЦЕСОРНА ТЕХНІКА ПРОМИСЛОВОГО ОБЛАДНАННЯ»

для студентів спеціальності
133 – Галузеве машинобудування
першого (бакалаврського) рівня вищої освіти

Міністерство освіти і науки України
Вінницький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ ДО ЛАБОРАТОРНИХ РОБІТ З ДИСЦИПЛІНИ «ЕЛЕКТРОНІКА ТА МІКРОПРОЦЕСОРНА ТЕХНІКА ПРОМИСЛОВОГО ОБЛАДНАННЯ»

для студентів спеціальності
133 – Галузеве машинобудування
першого (бакалаврського) рівня вищої освіти

Вінниця
ВНТУ
2019

Рекомендовано до друку Методичною радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 8 від 17.04.2019 р.)

Рецензенти:
О. В. Петров, кандидат технічних наук, доцент
Р. Р. Обертюх, доктор технічних наук, професор

Методичні вказівки до лабораторних робіт з дисципліни «Електроніка та мікропроцесорна техніка промислового обладнання» для студентів спеціальності 133 – Галузеве машинобудування першого (бакалаврського) рівня вищої освіти / Уклад. О. Д. Манжілевський. – Вінниця : ВНТУ, 2019. – 83 с.

У методичних вказівках подано роз’яснення щодо виконання восьми лабораторних робіт. Кожна із лабораторних робіт містить теоретичну частину, в якій подано зміст роботи і пояснення щодо виконання практичної частини роботи, а також приклади виконання окремих етапів.

ЗМІСТ

ВСТУП……. 4

ЛАБОРАТОРНА РОБОТА № 1. Дослідження характеристик діода.....5

ЛАБОРАТОРНА РОБОТА № 2. Дослідження характеристик
біполярного транзистора 12

ЛАБОРАТОРНА РОБОТА № 3. Двійкова система числення.
Двійково-десяткові перетворення чисел..... 22

ЛАБОРАТОРНА РОБОТА № 4. Двійкова арифметика.
Додавання, віднімання та множення двійкових чисел.
Додатковий числовий код 25

ЛАБОРАТОРНА РОБОТА № 5. Шістнадцяткова система числення.
Шістнадцятково-десяткові перетворення чисел.
Шістнадцятково-двійкові перетворення чисел 28

ЛАБОРАТОРНА РОБОТА № 6. Вивчення будови та принципів
роботи апаратно-програмних засобів Arduino 33

ЛАБОРАТОРНА РОБОТА № 7. Основи програмування
мікроконтролера Arduino..... 44

ЛАБОРАТОРНА РОБОТА № 8. Дослідження та синтез
комбінаційних схем управління..... 61

СПИСОК ЛІТЕРАТУРИ..... 73

ДОДАТОК А..... 74

ВСТУП

Методичні вказівки призначені допомогти студентам галузі знань «Механічна інженерія», освітньо-кваліфікаційного рівня бакалавр у самостійній роботі при виконанні лабораторного практикуму та оформленні звіту.

Лабораторний практикум розроблений відповідно до програми дисципліни «Електроніка та мікропроцесорна техніка промислового обладнання» для студентів галузі знань «Механічна інженерія» і призначений для студентів усіх форм навчання.

Цей лабораторний практикум містить роботи: «Дослідження характеристик діода», «Дослідження характеристик біполярного транзистора», «Двійкова система числення. Двійково-десяткові перетворення чисел», «Двійкова арифметика. Додавання, віднімання та множення двійкових чисел. Додатковий числовий код», «Шістнадцяткова система числення. Шістнадцятково-десяткові перетворення чисел. Шістнадцятково-двійкові перетворення чисел», «Вивчення будови та принципів роботи апаратно-програмних засобів Arduino», «Основи програмування мікроконтролера Arduino» та «Дослідження та синтез комбінаційних схем управління», які студенти повинні виконувати самостійно (в лабораторних аудиторіях кафедри «Галузевого машинобудування») після попереднього вивчення теоретичного матеріалу.

Метою проведення лабораторного практикуму є закріплення знань студентів із вивчення особливостей роботи типових напівпровідникових приладів, виконання перетворень у десятковій, двійковій та шістнадцятковій системах числення, що широко використовуються у програмуванні та можуть безпосередньо бути реалізованими в цифрових електричних схемах (останнє відноситься до двійкової системи числення) та виконання синтезу комбінаційних схем управління.

Основними завданнями лабораторного практикуму є:

• закріплення і розвиток знань, отриманих на лекціях, навичок самостійного практичного застосування здобутих теоретичних знань зі складання простих керівних програм на прикладі видозміненої мови програмування C / C++;

• ознайомлення з особливостями будови та роботи діодів і біполярних транзисторів;

• навчання методики дослідження та синтезу комбінаційних схем управління, оформлення письмового звіту про виконану роботу, самостійного викладу висновків і узагальнень.

Для написання цього лабораторного практикуму частково використовувались матеріали з практикуму Московського державного університету ім. М. В. Ломоносова (автор Козлов В. І.) та загальнодоступного інтернет-сайту «Студопедія».

ЛАБОРАТОРНА РОБОТА № 1

Дослідження характеристик діода

Мета роботи: ознайомитися із методикою зняття вольт-амперних характеристик (ВАХ) діодів; дослідити властивості схем випрямляння: однопівперіодної, мостової, симетричної із подвоєнням напруги; ознайомитися зі схемою стабілізатора напруги на стабілітроні.

Обладнання: прилад для зняття вольт-амперних характеристик діодів, демонстраційні блоки із зазначеними схемами випрямляння та стабілізації; згладжувальні RC-фільтри; осцилограф.

1.1 Теоретичні відомості

Напівпровідникові діоди

Напівпровідниковий діод – це *p-n*-перехід, який має випрямні властивості. На обох його ділянках *(p* і *n)* закріплені контактні виводи. За конструкцією переходи і діоди бувають площинні й точкові.

У площинних діодах лінійні розміри, що визначають площу переходу, значно більші від товщини переходу, у точкових лінійні розміри такі самі, як і товщина переходу. У точкових діодах (з точковим переходом) як випрямний елемент використовується контакт германієвої або кремнієвої монокристалічної пластинки з загостреним електродом із вольфраму, фосфористої або берилієвої бронзи. Структуру площинного і точкового діодів показано на рисунку 1.1.

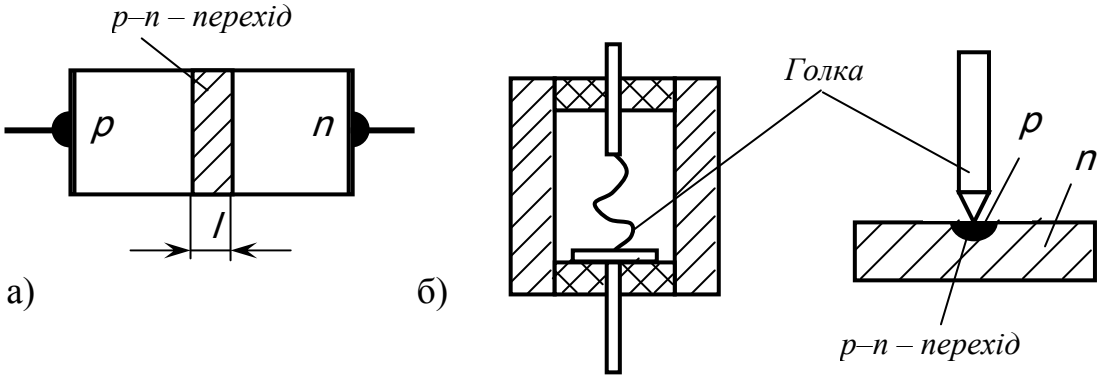


Рисунок 1.1 – Структура діода: а) – площинного; б) – точкового

1.2 Вольт-амперна характеристика діода

Розглянемо вольт-амперну характеристику ідеалізованого площинного діода, а потім зробимо необхідні уточнення, справедливі для реального діода. Ідеалізація зводиться до таких припущень:

а) перехід *p-n* несиметричний, ділянка *p* легована значно більше, ніж

ділянка *n (p > n)*. Можна вважати, що в такому переході інжекція відбувається лише в одному напрямі;

б) ширина переходу дуже мала, тому процесами, що відбуваються в ділянці переходу, можна знехтувати;

в) оскільки опір емітерної і базової ділянок незначний порівняно з опором ділянки переходу, вважатимемо, що зовнішня напруга прикладена до переходу.

При прямому зміщенні переходу внаслідок інжекції дірок з емітера в базу створюється підвищена концентрація їх на базовій межі переходу. Із віддаленням від переходу концентрація дірок зменшується внаслідок рекомбінації. Інжекція дірок у базу порушує її електричну нейтральність і спричиняє приплив надлишкових електронів із зовнішнього кола. Ці електрони розподіляються так, щоб скомпенсувати поле дірок, тобто розподіляються в базі за тим самим законом, що й дірки.

В установленому режимі в базі проходить дифузійний дірковий струм $J_{\text{рд}}$. Дірки, інжектовані в базу внаслідок наявності градієнта їх концентрації, дифундують від більшої концентрації до меншої.

Під час дифузійного руху частина дірок рекомбінує з електронами. Для відновлення електричної нейтральності, тобто поповнення електронів, що пішли на рекомбінацію, в базу з джерела напруги надходять електрони, рух яких і є базовим струмом.

Дифузійним електронним струмом $J_{\text{пд}}$, зумовленим інжекцією електронів з бази в емітер, відповідно до нашого припущення ($P_p >> n_n$), нехтуємо.

Отже, при прямому зміщенні діода через нього проходить дифузійний дірковий струм, при зворотному відбувається явище екстракції, яке й зумовлює зворотний струм діода.

Дірки бази (неосновні носії), що підійшли до ділянки переходу, внаслідок теплового руху викидаються полем переходу в ділянку емітера. Порушена електрична нейтральність в емітері й базі відновлюється припливом (від джерела напруги в емітер) електронів і відпливом з ділянки бази надлишкових електронів у джерело. Такий рух електронів і являє собою діркову дрейфову складову зворотного струму J_{pE}. Електронною дрейфовою складовою зворотного струму J_{nE}, що зумовлена екстракцією електронів з емітера в базу, зважаючи на припущення, що $p_n >> n_p$, нехтуємо. Зворотний струм діода, як і прямий, в основному дірковий.

Рівняння вольт-амперної характеристики ідеалізованого діода

$$I = I_0(e^{\frac{U}{\phi \cdot T}} - 1),$$

де $I=jS$; $I_0 \approx j_{p0} \cdot S$

Струм I_0 називається тепловим струмом, бо він значно залежить від температури внаслідок збільшення рівноважної концентрації неосновних

носіїв (дірок) у базі $p_{0б}$. Крім того, I_o називають «зворотним струмом насичення». При великих значеннях зворотної прикладеної напруги з її зміною він не змінюється. Вольт-амперну характеристику ідеалізованого діода показано на рисунку 1.2.

Із збільшенням зворотної прикладеної напруги (U< 0) збільшується потенціальний бар'єр, а дифузійний струм (струм основних носіїв) зменшується за експоненціальним законом. Загальний струм через перехід дорівнює тепловому струму, який, будучи струмом неосновних носіїв, не залежить від напруги. При $|U| \geq$ 0,1 ... 0,2 В значення $e^{U/\varphi T} << 1$ і тепловий струм визначається лише кількістю неосновних носіїв, які з'являються на межі переходу за одиницю часу, тобто температурою і концентрацією неосновних носіїв.

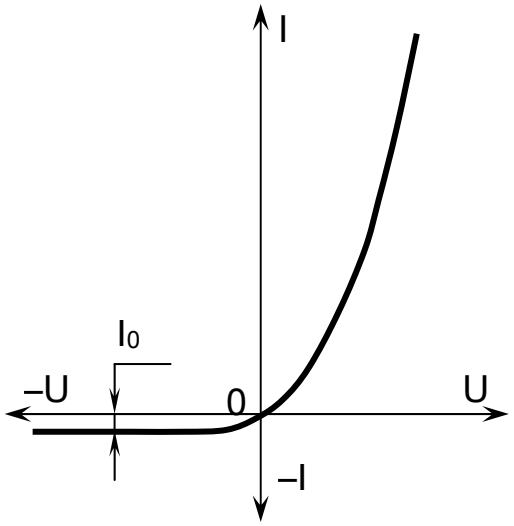


Рисунок 1.2 – Вольт-амперна характеристика ідеалізованого площинного діода

При збільшенні прямої напруги потенціальний бар'єр знижується, дифузійний струм зростає за експоненціальним законом, а тепловий струм лишається незмінним. Ним можна знехтувати вже при U = 0,1 В, оскільки $I_o e^{U/\varphi T} >> I_o$.

Із зміною знака прикладеної до діода напруги змінюється напрям струму. При зворотних напругах проходить струм, що дорівнює одиницям і десяткам мікроамперів. Водночас при невеликих прямих напругах у кілька десятих вольта ці самі діоди пропускають струми близько 1 А. Це дає підставу стверджувати, що напівпровідниковий діод придатний для випрямляння змінних струмів.

Розглянута вольт-амперна характеристика ідеалізованого діода добре узгоджується з експериментальною. Проте реальна вольт-амперна характеристика істотно відрізняється від ідеалізованої. У реальних діодах зворотний струм не залишається сталим, він суттєво зростає із збільшенням зворотної напруги і завжди при малих і великих зворотних напругах більший від теплового. Пояснюється це тим, що зворотний струм реального діода складається з таких струмів: теплового I_o, термогенерації I_G, поверхневого витоку I_B.

Тепловий струм зумовлений генерацією неосновних носіїв в об'ємах, прилеглих до переходу. З цих об'ємів неосновні носії дифундують у ділянку переходу і виносяться полем в іншу ділянку, де вони є основними носіями. У рівноважному стані (на перехід не подано напруги) ці потоки компенсуються зустрічними потоками інжектованих основних носіїв.

Тепловий струм змінюється із зміною температури, головним чином внаслідок зміни рівноважної концентрації неосновних носіїв у базі $p_{об}$ і в

емітері n_{oe}. Якщо в ідеалізованому діоді ми вважали перехід нескінченно вузьким і нехтували генерацією та рекомбінацією носіїв заряду в ньому, то в реальних діодах цього зробити не можна. Носії, що генеруються в ділянці переходу, швидко виносяться електричним полем у відповідну ділянку переходу; це і є струм термогенерації I_G . У рівноважному стані діода струм термогенерації I_G компенсується таким самим струмом рекомбінації I_R.

Струм рекомбінації зумовлений основними носіями, енергія яких недостатня для інжекції (подолання потенціального бар'єра), а тому вони лише проникають у перехід з емітера і бази. Поблизу точки «відбиття» ці носії мають малу швидкість і встигають рекомбінувати.

Нерідко головним фактором, що впливає на зворотну характеристику діода, є струм поверхневого витоку I_B. Він зумовлений насамперед явищем генерації і рекомбінації, а також молекулярними плівками окислів міді, газів, парів води та інших на поверхні напівпровідника. При підвищенні зворотної напруги струм витоку зростає спочатку майже лінійно, а потім більш різко.

Характерною особливістю струму витоку є його часова нестабільність, яку називають «повзучістю». Вона виявляється в зміні зворотного струму протягом деякого часу після стрибкоподібної зміни зворотної напруги, зокрема після ввімкнення. Час наростання або спадання зворотного струму змінюється від кількох секунд до кількох годин. Зворотні характеристики реальних германієвих і кремнієвих діодів показано на рисунку 5.3. При значному збільшенні зворотної напруги струм діода зростає спочатку повільно, а потім дедалі швидше. Якщо не вжити спеціальних заходів, то *p-n*-перехід буде пробитий.

Розрізняють польовий, лавинний, тепловий і поверхневий пробої *p-n*-переходу. Польовий пробій буває двох різновидів: зенерівський і тунельний. Зенерівський пробій полягає в тому, що досить сильне електричне поле в *p-n*-переході іонізує атоми напівпровідника (валентні електрони з ковалентних зв'язків переходять у зону провідності), внаслідок цього різко збільшується зворотний струм через перехід. Напруга зенерівського пробою залежить від питомого опору напівпровідника. Чим більший його питомий опір, тим більша напруга пробою.

Тунельний пробій пояснюється тунельним ефектом, суть якого полягає ось у чому. При досить сильних електричних полях у *p-n* переходах (понад 10^7 В/см) завдяки хвильовим властивостям електрони можуть переходити з валентної зони ділянки *p* у зону провідності ділянки *n* і в зворотному напрямі: із зони провідності ділянки *n* у валентну зону ділянки *p* без зміни їх енергії, тобто не долаючи потенціального бар'єра. Умови для розвитку тунельного ефекту створюються в тонких (менших від 0,01–0,02 мкм) *p-n*-переходах з високолегованих ($N > 10^{19}$ см~^{3}) напівпровідників.

Лавинний пробій у *p-n*- переходах зумовлюється явищем ударної іонізації. При цьому неосновні носії в ділянці *p-n*-переходу (які утворюють тепловий струм I_o) під впливом електричного поля набувають енергії, до-

статньої для ударної іонізації атомів напівпровідника. Електрони, які виникли в результаті іонізації, й собі розганятимуться полем і утворюватимуть нові електрони та дірки і т. д. При достатньо великій напруженості поля іонізація набуває лавинного характеру (подібно до самостійного розряду в газах), і зворотний струм різко збільшується. Цей вид пробою властивий слабколегованим напівпровідникам, коли ширина *p*-*n*-переходу є досить великою. Напруга лавинного пробою збільшується із зростанням питомого опору напівпровідника. Встановлено, що напруга польового (зенерівського, тунельного) пробою менша за 2 В для германієвих і менша за 5 В для кремнієвих переходів.

Лавинний пробій характерний для германієвих переходів при напругах понад 5 В і для кремнієвих – 7 В. При проміжних значеннях напруг обидва види пробою можуть статися одночасно.

Тепловий пробій є наслідком виділення тепла в переході під час проходження зворотного струму. Теплова енергія підвищує температуру переходу, що збільшує тепловий струм і потужність, розсіювані в переході. Таке взаємозумовлене наростання температури і потужності, розсіюваної в переході, може набути лавинного характеру і внаслідок цього може статися тепловий пробій.

При тепловому пробої внаслідок теплового збудження розриваються зв'язки між атомами кристалічної ґратки . Тепловий пробій буває при значно нижчих напруженостях електричного поля, ніж польовий і лавинний, у тих випадках, коли тепло не відводиться від переходу, який працює в режимі великих струмів. Зворотну вольт-амперну характеристику діода при польовому, лавинному і тепловому пробоях показано на рисунку 1.3.

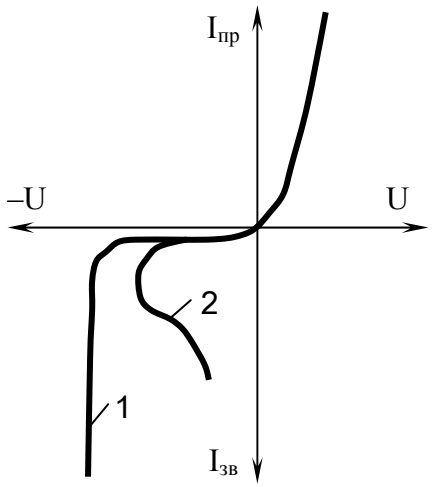


Рисунок 1.3 – Зворотна вольт-амперна характеристика діода при пробої: 1 – лавинному або польовому; 2 – тепловому

У кремнієвих напівпровідникових приладах тепловий струм I_0 такий малий, що тепловий пробій практично виключається. Напруга теплового пробою залежить від значення зворотного струму, опору переходу, умов тепловідведення, температури навколишнього середовища і т. д.

При всіх розглянутих видах пробою необмежене зростання струму призводить до руйнування переходу і повного виходу з ладу діода. Щоб запобігти пробою переходу, потрібно під час роботи послідовно з діодом вмикати струмообмежувальний резистор з опором такої величини, щоб при пікових значеннях напруги струм у колі не перевищував допустимого значення.

Пряма гілка вольт-амперної характеристики реального діода відрізняється від вольт-амперної характеристики ідеалізованого діода через наявність

струму рекомбінації в переході, вплив опору $r_б$ і модуляцію базового шару.

При прямому зміщенні переходу знижується потенціальний бар'єр і різко зростає концентрація дірок та електронів у ділянці переходу, а також глибина їх проникнення в перехід, тому ймовірність рекомбінації збільшується, при цьому струм рекомбінації перевищує струм термогенерації. Через струм рекомбінації пряма гілка характеристики реального діода крутіша від ідеальної. Струм рекомбінації особливо відчутний у кремнієвих діодах при зниженій температурі, коли тепловий струм дуже зменшується. На пряму гілку вольт-амперної характеристики впливає опір бази внаслідок спаду напруги на ньому і зменшення напруги, прикладеної до переходу, порівняно з зовнішньою прикладеною напругою.

Пряма гілка вольт-амперної характеристики для будь-якого реального діода є більш пологою, ніж для ідеалізованого (рисунок 1.4, а).

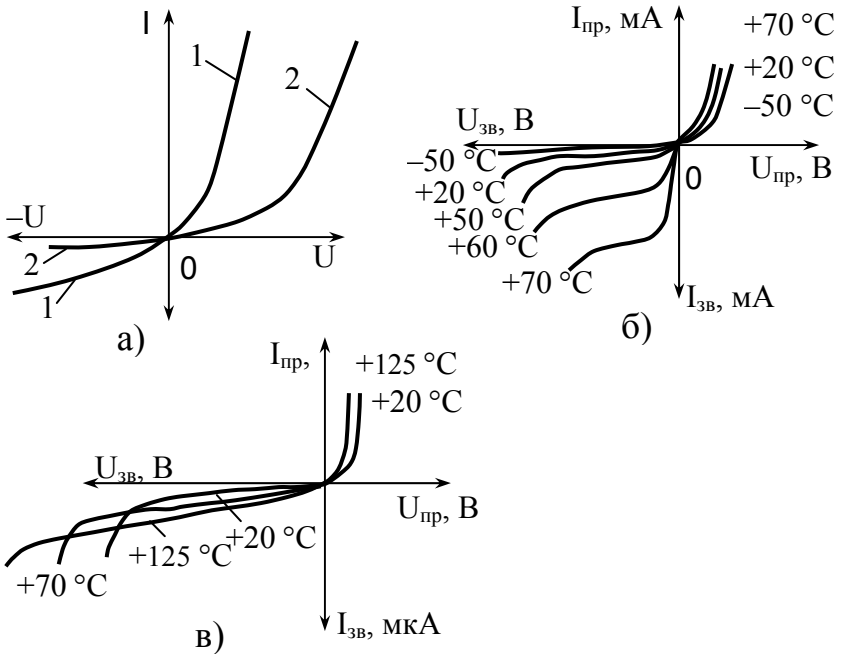


Рисунок 1.4 – Вольт-амперні характеристики германієвого (1) і кремнієвого (2) діодів (а) та їх зміна при зміні температури (б, в)

Для германієвих діодів похил вольт-амперної характеристики на початковій ділянці визначається тепловим струмом, у кремнієвих діодах — струмом рекомбінації. Збільшення прямого струму в германієвих діодах починається при менших значеннях прямої напруги, ніж у кремнієвих. Лінійна ділянка характеристики германієвих діодів крутіша, ніж кремнієвих.

Оскільки у кремнієвих діодах тепловий струм I_o значно менший, ніж у германієвих, то початкова ділянка прямої гілки вольт-амперної характеристики у кремнієвих діодах дуже полога. Це пояснюється тим, що заборонена зона кремнію значно ширша, ніж германію.

Із зміною температури змінюється як зворотна, так і пряма гілки вольт-амперної характеристики (рисунок 1.4, б), в), причому зміни зворо-

тного струму помітніші, ніж прямого. Причина цього – сильні температурні зміни концентрації неосновних носіїв, які визначають тепловий струм I_0 і струм термогенерації I_G, основні складові зворотного струму.

Концентрація основних носіїв, яка визначає прямий струм, із зміною температури не змінюється, бо вона визначається, в основному, концентрацією домішок. З підвищенням температури зворотний струм збільшується приблизно у два рази на кожні 10 °С у германієвих діодах (рисунок 1.4, б) і приблизно у два з половиною рази в кремнієвих (рисунок 1.4, в). Пряма гілка характеристики з підвищенням температури зміщується вліво і стає крутішою.

1.3 Завдання до практичної роботи

1. Підключити діод до вимірювального приладу в прямому напрямку, зняти пряму гілку ВАХ; повторити роботу для зворотного підключення; побудувати за отриманими результатами графіки, апроксимувати експериментальні точки аналітичною кривою.

2. Змонтувати схему однопівперіодного випрямляння, підключити до напруги (12 В!), спостерігати на осцилографі форми сигналу в різних точках схеми; дослідити вплив згладжувального конденсатора на форму вихідної напруги для різних величин вихідного струму.

– Дослідити форму вихідної напруги для двох інших схем випрямляння, накреслити криві з екрана осцилографа.

– Скласти схему найпростішого стабілізатора напруги, під'єднати до мостової схеми випрямляння, спостерігати на осцилографі форму напруги на вході та виході стабілізатора при зміні вихідного струму.

Контрольні запитання

1. Що являє собою напівпровідниковий діод?
2. Які матеріали застосовують для виготовлення діодів?
3. Які основні функції діода, ви можете назвати?
4. Як класифікуються напівпровідникові діоди?
5. Зобразіть вольт-амперну характеристику ідеалізованого площинного діода та коротко поясніть її фізичну суть.
6. Які типи пробою *p*-*n*-переходу ви знаєте?

ЛАБОРАТОРНА РОБОТА 2
Дослідження характеристик біполярного транзистора

Мета роботи: у цій роботі необхідно зняти характеристики транзистора, обчислити його основні параметри, а також виконати дослідження підсилення змінного сигналу підсилювачем на транзисторі.

2.1 Фізичні процеси, що мають місце в транзисторі

Розглянемо біполярний транзистор, тобто напівпровідниковий прилад з двома *p*-*n*-переходами, для роботи якого є характерним створення струму носіїв заряду обох знаків.

Будову транзистора показано на рис. 2.1. Він являє собою пластинку германію або іншого напівпровідника, в якій створені три області з різною електропровідністю.

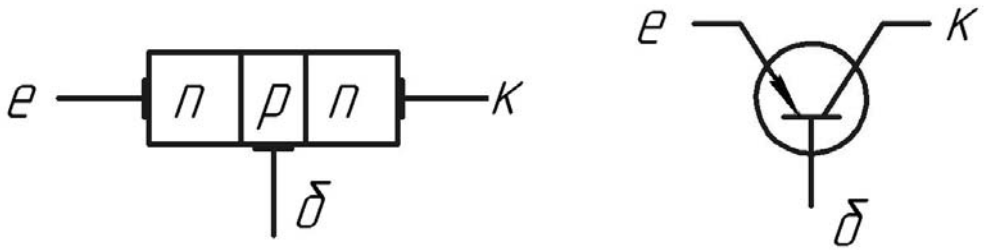


Рисунок 2.1 – Будова та схема позначення біполярного транзистора

Транзистор типу *n*-*p*-*n* має середню область з дірковою електропровідністю, а дві крайні області – з електронною електропровідністю. Середня область називається базою, одна крайня область – емітером, інша – колектором. Відстань між *p*-*n*-переходами має бути дуже малою, не більше декількох мікронів, тобто область бази має бути дуже тонкою.

Розглянемо, що відбуватиметься в транзисторі при підключенні до нього зовнішніх напруг. Зазвичай до емітерного переходу підключають напругу, що є прямою для основних носіїв заряду областей, прилеглих до цього переходу, а до колекторного переходу – зворотну напругу (рис. 2.2). При такому включенні транзистора струм між емітером і базою i_e буде переважно складатися з електронів, що рухаються з емітера до бази. Дірки з бази в емітер проникають легко – для них емітерний перехід також включений в прямому напрямку. Але концентрацію дірок у базі роблять значно меншою концентрації вільних електронів в області емітера. Тому струмом дірок з бази в область емітера можна знехтувати.

Частина електронів, проходячи через базову область, рекомбінують там з дірками, однак більша їх частина дифундує крізь вузьку базову

область і потрапляє в поле колекторного переходу, який для них відкритий. Таким чином, струм колектора практично дорівнює струму емітера, незначно відрізняючись від нього внаслідок рекомбінації частини електронів в базовій області.

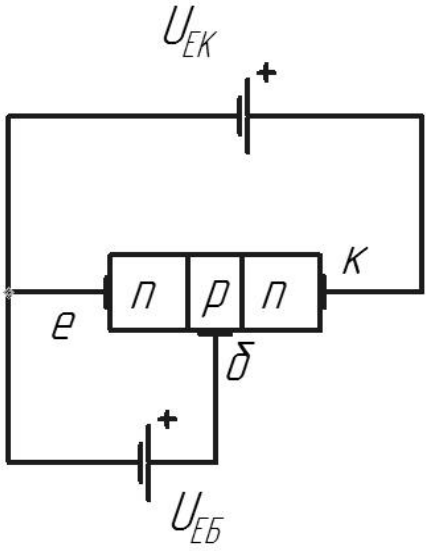


Рисунок 2.2 – Схема підключення зовнішніх напруг до транзистора

Оскільки частина дірок в базі рекомбінують з електронами, які надійшли з емітера, то для їх заповнення в базі утворюються нові дірки (внаслідок відходу зайвих електронів з бази в зовнішнє коло). Таким чином, струм бази можна вважати таким, що складається з дірок, які надходять із зовнішнього кола.

Призначенням емітера є інжекція (впуск, впорскування, але не емісія у звичайному розумінні цього терміна) носіїв заряду в базу. База – це область, в яку інжектуються емітером неосновні для неї носії заряду. Колектор – область, призначенням якої є екстракція (прийом, поглинання) носіїв заряду з бази.

Між розглянутими струмами відповідно до закону Кірхгофа виконується співвідношення

$$i_e = i_к + i_б. \tag{2.1}$$

Аналогічне співвідношення виконується і для приростів струмів:

$$\Delta i_e = \Delta i_к + \Delta i_б. \tag{2.2}$$

Роботу транзистора ілюструє потенціальна діаграма (рис. 2.3).

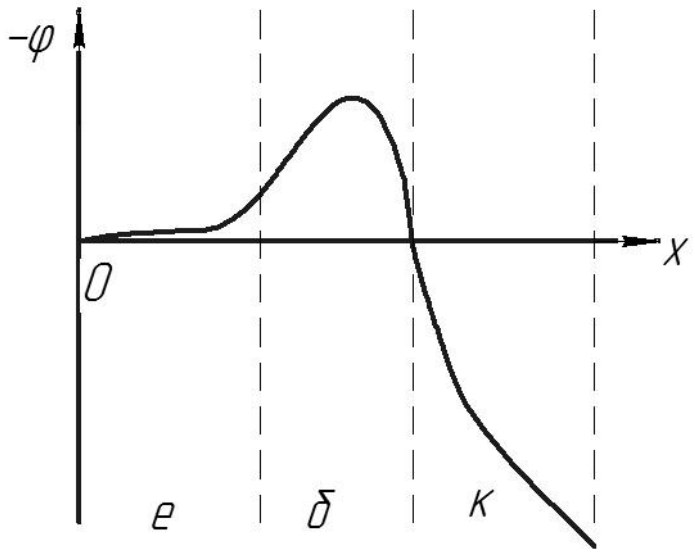


Рисунок 2.3 – Потенціальна діаграма роботи транзистора

Потенціал емітера взято за нульовий. У емітерний перехід, хоча він і включений в прямому напрямку для електронів (основних носіїв заряду в емітерній області), все-таки є невеликий потенційний бар'єр. Висотою цього бар'єра можна управляти, змінюючи напругу між базою і емітером. Поле в області колектора для електронів є прискорювальним.

2.2 Характеристики транзистора

Залежність між струмами, що діють в транзисторі, має складний характер. Її зручно описувати графічно у вигляді деяких характеристик.

Існують три основних схеми включення транзистора в підсилювачах. Вони відрізняються тим, який із трьох електродів є спільним для вхідного та вихідного кола. Розглянемо схему зі спільним емітером (рис. 2.4). Вхідна напруга $U_{сигн}$, яку необхідно підсилити, подається на ділянку база–емітер. На базу подається також напруга зсуву $U_{бе}$. При цьому в колі бази протікає деякий струм, тобто вхідний опір транзистора виходить порівняно невеликим. Коло колектора (вихідне коло) живиться від джерела $U_{ке}$. Для отримання підсиленої вихідної напруги в це коло включено резистор навантаження R.

Статичні характеристики знімаються при постійному струмі.

Як вхідні характеристики для описаної схеми можна розглядати залежність

$$i_б\left(U_{бе}\right) при\ U_{ке} = const, \tag{2.3}$$

а вихідні – залежність

$$i_к\left(U_{ке}\right) при\ i_б = const. \tag{2.4}$$

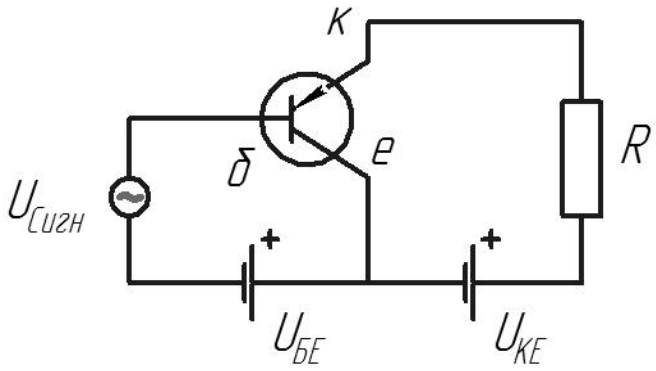


Рисунок 2.4 – Схеми включення транзистора зі спільним емітером

Сімейства зазначених характеристик зображені на рис. 2.5.

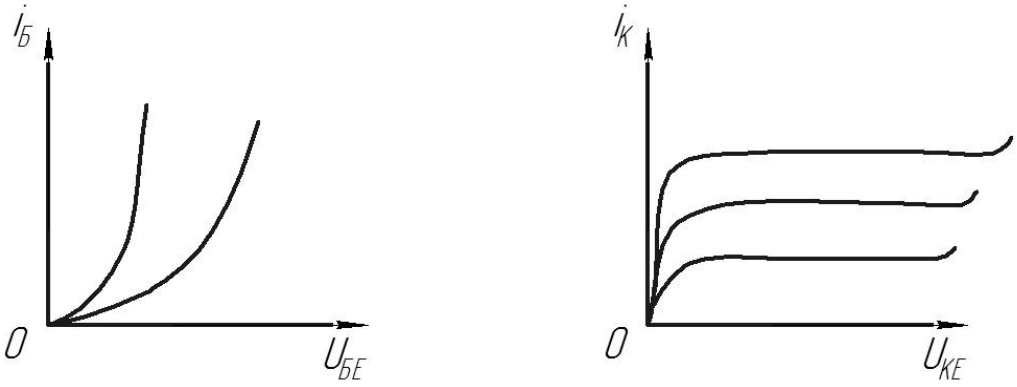


Рисунок 2.5 – Вольт-амперні характеристики транзисторів

Вхідні і вихідні характеристики транзистора мають тісний зв'язок з вольт-амперною характеристикою напівпровідникового діода. Дійсно, вхідні характеристики відносяться до емітерного переходу, який працює при прямій напрузі. Тому вони подібні характеристиці прямого струму діода. Вихідні характеристики подібні до характеристики зворотного струму діода, оскільки відображають властивості колекторного переходу, що працює при зворотній напрузі (для основних носіїв заряду).

2.3 Параметри транзистора

За вхідними характеристиками можна визначити вхідний опір транзистора:

$$R_{вх} = \frac{\Delta U_{бе}}{\Delta i_б} \ при \ U_{ке} = const, \qquad (2.5)$$

а за вихідними – вихідний опір:

$$R_{вих} = \frac{\Delta U_{ке}}{\Delta i_к} \ при \ i_б = const. \tag{2.6}$$

Обчислені таким чином величини $R_{вх}$ та $R_{вих}$ є опорами транзистора за змінним струмом, тобто такий опір чинить транзистор змінному струму невеликої амплітуди (порівняно з величиною постійного струму, що тече відповідно у вхідному і вихідному колах). Величини $R_{вх}$ та $R_{вих}$ використовуються при аналізі транзистора в схемах підсилення змінного струму або напруги.

Вхідний і вихідний опори транзистора за постійним струмом обчислюються за формулами, в яких фігурують не малі прирости, а самі струми і напруги.

Одним з найважливіших параметрів транзисторів є коефіцієнт підсилення струму бази β, який визначається як відношення приросту струму колектора до приросту струму бази при постійній різниці потенціалів між емітером і колектором.

$$\beta = \frac{\Delta i_к}{\Delta i_б} \ при \ U_ж = const. \tag{2.7}$$

Оскільки основна частина струму емітера замикається через коло колектора, коефіцієнт β завжди значно більшим одиниці. У сучасних транзисторів $\beta = 10 \div 300$.

Іноді буває зручно розглядати інший параметр – коефіцієнт підсилення за струмом емітера α. Він визначається відношенням приросту струму колектора до приросту струму емітера при постійній різниці потенціалів між колектором і базою $U_{бк}$:

$$\alpha = \frac{\Delta i_к}{\Delta i_е} \ при \ U_{бк} = const. \tag{2.8}$$

Значення α завжди менше одиниці. Оскільки зазвичай завжди і, $U_{еб} << U_{ек}$ і відповідно, $U_{ек} \approx U_{бк}$, то враховуючи (2.7), можна записати зв'язок між коефіцієнтом β і α у вигляді

$$\beta = \frac{\alpha}{1 - \alpha} \ . \tag{2.9}$$

2.4 Підсилення напруги за допомогою транзистора

При будь-якій схемі включення транзистора в ролі підсилювача електричний струм є наявним як у вихідному колі, так і у вхідному. Це дозволяє говорити про підсилювачі на транзисторі як про підсилювачі струму. Але вхідний струм транзистора пов'язаний з вхідною напругою (для схеми з загальним емітером струм бази пов'язаний з напругою, що діє між базою і емітером, через вхідний опір $R_{вх}$). Тому можна скористатися цим зв'язком і говорити про підсилення напруги. А з точки зору фізики явища потрібно мати на увазі, що вихідний струм, тобто струм, що протікає через навантаження і визначає підсилення, залежить від висоти потенційного бар'єра в області емітерного переходу, тобто від напруги, що діє між базою і емітером. Таким чином, схему з транзистором можна розглядати і як підсилювач струму, і як підсилювач напруги та, природно, як підсилювач потужності.

Коефіцієнт підсилення за напругою схеми з транзистором можна обчислити таким чином:

$$K = \frac{\Delta U_{вих}}{\Delta U_{вх}} = \frac{\Delta i_к R_н}{\Delta i_б R_{вх}} = \beta \frac{R_н}{R_{вх}}. \qquad (2.10)$$

Ця проста формула справедлива лише в невеликій області режимів роботи транзистора. Більш строгий розрахунок виконується за більш складною формулою.

Як і в будь-якому іншому підсилювачі, коефіцієнт підсилення транзисторного підсилювача залежить від частоти сигналу і його амплітуди.

2.5 Опис експериментальної установки

Вивчення транзистора виконується на установці, що зображена на рис. 2.6. Ця схема створена за допомогою програми Electronic Workbench demo version, що дозволяє виконати потрібні практичні дослідження.

На установці досліджується транзистор типу МП37Б. Живлення схеми здійснюється від джерела постійної напруги 15 В. Струм, що споживається схемою від джерела, не більший 5 мА. У схемі під'єднані прилади, що вимірюють струм і напругу бази та колектора. На вхід транзисторного підсилювача подається сигнал від генератора змінної напруги. Форму і величину цього сигналу на вході і виході підсилювача можна спостерігати за допомогою осцилографа.

Величина постійної напруги на базі транзистора регулюється потенціометром R_1, а на колекторі – потенціометром R_4. Резистор R_2 відіграє допоміжну роль, обмежуючи величину базового струму транзистора і сприяючи плавності регулювання базової напруги. Резистор R_3 при вивченні підсилення виконує роль навантаження. Резистор R_5 зменшує величину напруги, що подається на потенціометр R_1. Конденсатор C_1 дозволяє підвести до бази транзистора змінний сигнал, не порушуючи

режиму роботи транзистора за постійним струмом. Конденсатор C_2 «закорочує» на мінус схеми один з кінців навантажувального резистора R_2 (за змінним струмом). Величини елементів схеми: R_1 = R_4 = 6,8 кОм, R_2 = 56 кОм, R_3 = 1,1 кОм, R_5 = 12 кОм, C_1 = C_2 = 1 мкФ.

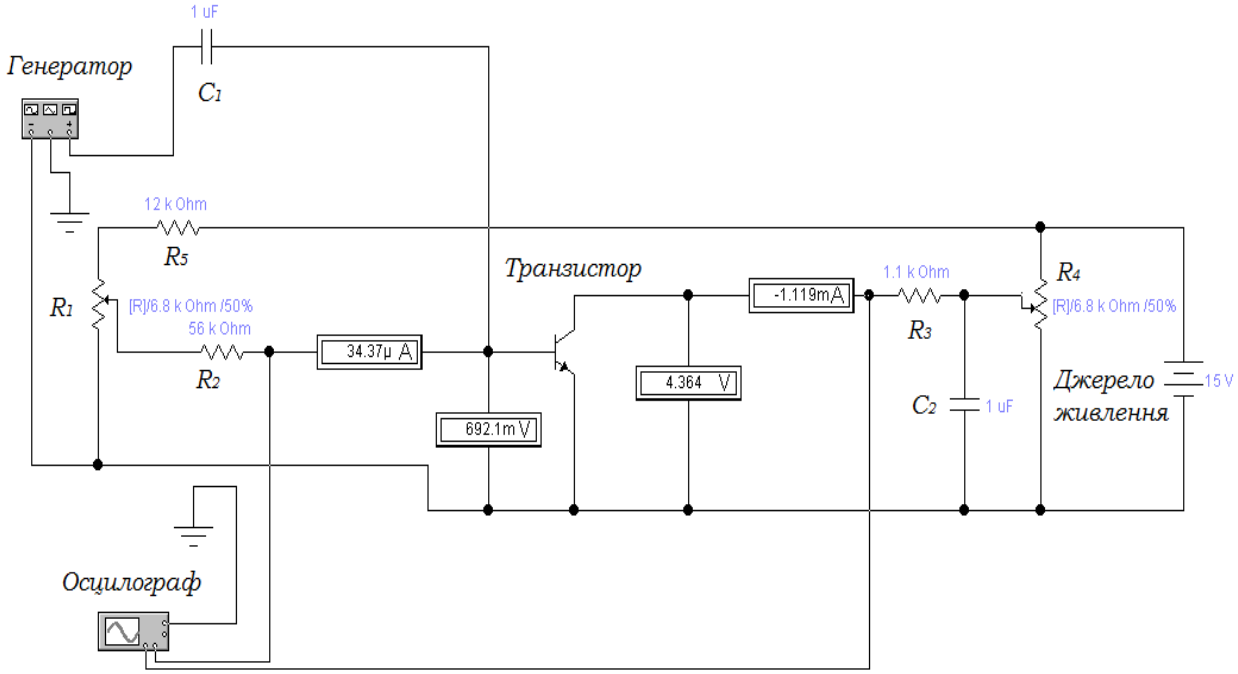


Рисунок 2.6 – Схема установки для дослідження біполярного транзистора

Струм бази вимірюється мікроамперметром з межею вимірювання 100 мкА, струм колектора – міліамперметром з межею вимірювання 3 мА.

Напруги на базі і колекторі вимірюються вольтметрами з великим вхідним опором.

Як джерело підсилюваного змінного сигналу використовується генератор (50 мВ).

Візуально сигнал спостерігається на екрані осцилографа.

Величина підсилюваного сигналу оцінюється по вольтметру генератора, а величина підсиленого сигналу – за допомогою осцилографа.

2.6 Завдання до практичної роботи

Вправа 1. Зняття вихідних характеристик транзистора

Дослідження виконуються за схемою, що наведена на рис. 2.6. Значення резистора R_3 встановити таким, що дорівнює нулю. Значення опорів потенціометрів R_1 і R_4 встановити максимальними. Змінюючи величину опору потенціометра R_1, встановити струм бази $I_б$ = 10 мкА. Потім змінювати значення опору потенціометра R_4 до появи колекторного струму (при цьому струм бази може злегка змінитися). Потенціометром R_1 відновити значення $i_б$ = 10 мкА. Отримані при цьому значення $U_{ке}$ та $i_к$

занести в таблицю. Зняти залежність $i_к$ ($U_{ке}$) при $i_б$ = 10 мкА, збільшуючи $U_{ке}$ за допомогою R_4 і підтримуючи $i_б$ постійним за допомогою R_1.

Після цього аналогічним чином зняти залежність $i_к$ ($U_{ке}$)при $i_б$ = 20, 30 і 40 мкА. Результати вимірювань занести в таблицю.

Побудувати графіки $i_к$ ($U_{ке}$) при $i_б$ як параметр.

Для використання транзистора як підсилювача змінного сигналу (вправа 3) транзистор необхідно поставити в режим роботи за постійним струмом, що забезпечує лінійний зв'язок між $U_{ке}$ та $i_к$ (при цьому спотворення підсилюваного сигналу будуть найменшими). На графіку, де побудовано сімейство вихідних характеристик транзистора, визначити точку, приблизно відповідну центру лінійних ділянок характеристик. Нехай це буде точка з координатами $U_{ке} = U_{ке}^*$, $i_б = i_б^*$.

Записати значення $U_{ке}^*$ та $i_б^*$. Вважаючи, що в околиці цієї точки параметри транзистора практично незмінні, визначити величину коефіцієнта підсилення транзистора по струму β^* (див. ф-лу (2.7)) і величину вихідного опору транзистора $R_{вих}$ (див. ф-лу (2.6)).

Вправа 2. Зняття вхідної характеристики транзистора.

За допомогою потенціометра R_4 встановити $U_{ке} = U_{ке}^*$.

Зняти залежність $i_б$ ($U_{бе}$), занести дані в таблицю. При цьому напруга $U_{бе}$ збільшувати за допомогою потенціометра R_1, підтримуючи $U_{ке} = U_{ке}^*$ за допомогою потенціометра R_4. Закінчивши вимірювання, зменшити напруження і струми в схемі до мінімуму.

Побудувати графік $i_б$ ($U_{бе}$) при $U_{ке} = U_{ке}^*$. Для значення $i_б = i_б^*$ по нахилу залежності $i_б$ ($U_{бе}$) визначити величину вхідного опору транзистора $R_{вх}$ (див. формулу (2.5)).

Вправа 3. Спостереження ефекту підсилення змінного сигналу транзистором.

Знаючи величини β і $R_{вх}$, знайдені в попередніх вправах для певного режиму роботи транзистора ($U_{ке} = U_{ке}^*$, $i_б = i_б^*$), а також знаючи опір резистора навантаження $R_н$, можна обчислити величину коефіцієнта підсилення транзисторного підсилювача, в якому транзистор стоїть, знаходиться в зазначеному режимі, а як навантаження використовується резистор $R = R_н$. Обчислити коефіцієнт посилення K (див. ф-лу (2.10)).

Зняти короткозамикальну перемичку з резистора R_3. За допомогою потенціометрів R_1 та R_4 встановити режим роботи транзистора за постійним струмом, що відповідає обраному за результатами попередніх вправ ($U_{ке} = U_{ке}^*$, $i_к = i_к^*$). Включити генератор і подати з нього невелику змінну напругу U (Наприклад, 10 мВ) частоти (наприклад, 1 кГц) на вхід транзистора $\tilde{U}_{вх}$ (через конденсатор C_1). Величина цієї напруги визначається за допомогою вимірювального приладу генератора.

Підключити за схемою відповідно до рис. 2.6 осцилограф і на його екрані отримати зображення сигналу, що діє на вході транзистора, а потім –

на виході. Переконатися в тому, що вихідний сигнал зберіг свою гармонічну форму. Якщо ж цього немає, то домогтися цього, зменшивши $\tilde{U}_{вх}$. Сигнал з осцилографа скопіювати з робочого вікна вищезгаданої програми та додати до звіту (рис. 2.7).

Повернувшись до оптимального режиму роботи транзистора, тобто при $U_{ке}$* та $i_к$* і при досить малому $\tilde{U}_{вх}$, виміряти величину змінної напруги на виході транзистора $\tilde{U}_{вх}$ за його зображенням на екрані осцилографа:

$$\tilde{U}_{вих} = \frac{C_y L_y}{2\sqrt{2}}, \tag{2.11}$$

де $\tilde{U}_{вх}$ – ефективне значення напруги (у вольтах);

C_y – чутливість осцилографа, зазначена на його лицьовій панелі (у вольтах на велику поділку шкали на екрані осцилографа);

L_y – розмір зображення досліджуваної напруги по вертикалі (в великих поділках);

$1/2\left(\sqrt{2}\right)$ – коефіцієнт, обумовлений переведенням величини спостережуваного сигналу на екрані осцилографа (подвоєна амплітуда) в ефективне значення.

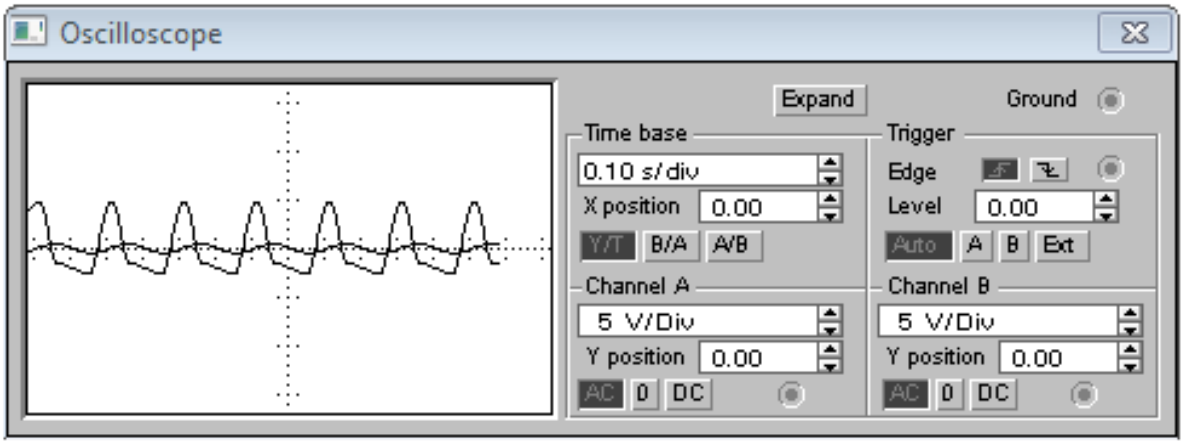


Рисунок 2.7 – Сигнал з осцилографа

Обчислити коефіцієнт підсилення сигналу, що отримується експериментально:

$$K_{експ} = \frac{\tilde{U}_{вих}}{\tilde{U}_{вх}}. \tag{2.12}$$

Порівняти $K_{експ}$ та $K_{теор}$.

Контрольні питання

1. Що являє собою транзистор n-p-n типу?
2. Яку роль відіграє емітер, база і колектор в транзисторі?
3. Чому базу транзистора роблять тонкою?
4. Які носії заряду є основними і неосновними в емітерній, базовій і колекторній областях?
5. Для яких носіїв заряду колекторний перехід включається в зворотному напрямку?
6. Пояснити схожість між вхідними та вихідними характеристиками транзистора з одного боку і вольт-амперної характеристикою діода – з іншого?
7. Як можна знайти величини вхідного і вихідного опорів транзистора?
8. Як можна визначити коефіцієнт підсилення транзистора за струмом і напругою?
9. Якого порядку зазвичай бувають вхідний і вихідний опори транзистора?

ЛАБОРАТОРНА РОБОТА № 3

Двійкова система числення. Двійково-десяткові перетворення чисел

Мета роботи: освоїти методику двійково-десяткових та десятково-двійкових перетворень чисел

3.1 Короткі теоретичні відомості

Система числення – це система запису чисел за допомогою певного набору знаків.

У звичній для нас системі запису чисел – десятковій системі числення (Історично ця система виникла при використанні для лічби пальців на руках) – для запису чисел використовуються десять цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. У цій системі будь-яке ціле невід'ємне число подається за допомогою степенів числа 10 ($10^0 = 1;\ 10^1 = 10;\ 10^2 = 100;\ 10^3 = 1000;\ 10^4 = 10000\ ...$) . Число 10 є основою цієї системи числення.

Дійсно, якщо число менше за 10, то записується відповідна йому одна цифра.

Якщо число більше або дорівнює 10, але менше за 100, то воно подається двома цифрами: перша показує кількість повних десятків, що містяться в числі, друга – кількість одиниць в останньому неповному десятку. Наприклад: $87 = 80 + 7 = 8 \cdot 10 + 7 = 8 \cdot 10^1 + 7 \cdot 10^0 = 87_{10}$.

Індекс внизу вказує систему числення, в якій записане вихідне число.

Якщо число більше або дорівнює 100, але менше за 1000, то для його запису використовуються вже три цифри. Перша цифра – це кількість повних сотень, що містяться в числі, друга цифра – кількість повних десятків у останній неповній сотні, третя цифра – кількість одиниць в останньому неповному десятку.

Наприклад:

$$645 = 600 + 40 + 5 = 6 \cdot 100 + 4 \cdot 10 + 5 = 6 \cdot 10^2 + 4 \cdot 10^1 + 5 \cdot 10^0 = 645_{10} .$$

Двійкова система числення – це система, в якій для запису чисел використовуються дві цифри, 0 і 1. Основою двійкової системи числення є число 2.

Для запису числа у двійковій системі використовується подання цього числа за допомогою степенів числа 2.

Розглянемо на прикладах, як подаються числа за допомогою степенів числа 2.

Спочатку наведемо таблицю значень степенів числа 2.

Таблиця 3.1 – Значення степенів числа 2

n	0	1	2	3	4	5	6	7	8	9	10
2^n	1	2	4	8	16	32	64	128	256	512	1024

Скориставшись цією таблицею, можна записати:

$0 = 0 \cdot 2^0$

$1 = 2^0 = 1 \cdot 2^0$

$2 = 2^1 = 1 \cdot 2^1 + 0 \cdot 2^0$

$3 = 2 + 1 = 2^1 + 2^0 = 1 \cdot 2^1 + 1 \cdot 2^0$

$4 = 2^2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$

$5 = 4 + 1 = 2^2 + 2^0 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$

$6 = 4 + 2 = 2^2 + 2^1 = 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$

$7 = 4 + 2 + 1 = 2^2 + 2^1 + 2^0 = 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$

$25 = 16 + 8 + 1 = 2^4 + 2^3 + 2^0 = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$

$120 = 64 + 32 + 16 + 8 = 2^6 + 2^5 + 2^4 + 2^3 = 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0$.

Двійковий код числа – запис цього числа у двійковій системі числення.

Таким чином, двійковим кодом числа є послідовність коефіцієнтів, як вказано у прикладі вище.

У наведених прикладах двійкові коди мають вигляд:

0 =	0_2
1 =	1_2
2 =	10_2
3 =	11_2
4 =	100_2
5 =	101_2
6 =	110_2
7 =	111_2
25 =	11001_2
120 =	1111000_2

Зверніть увагу: Коефіцієнти в прикладі мають набувати лише одне з двох значень: 0 або 1. Це забезпечує однозначність такого подання.

Таким чином, в двійковому численні будь-який розрахунок можна подати двома числами: 0 і 1. Для подання цих чисел в цифрових системах досить мати електронні схеми, які можуть приймати два стани, що чітко різняться значенням якої-небудь електричної величини – потенціал або струм. Одному із значень цієї величини відповідає цифра 0, іншому – 1. Відносна простота створення електронних схем з двома електричними станами і привела до того, що двійкове подання чисел домінує в сучасній цифровій техніці. При цьому 0 зазвичай подається низьким рівнем

потенціалу, а 1 – високим рівнем. Така двійкова логічна система, в якій логічній «1» відповідає вищий рівень сигналу, ніж логічному «0», називається позитивною логікою.

Переведення десяткового числа в двійковий код можна здійснювати шляхом послідовного ділення числа на 2. Залишки (0 або 1), що виходять на кожному кроці ділення, формують двійковий код перетворюваного числа, починаючи з його молодшого розряду. Як старший розряд двійкового коду записується 1, отримана в результаті останнього кроку ділення. Наприклад, перетворення числа 109 у двійковий код виконується таким чином:

A_2: залишки

a_0=1

a_1=0

a_2=1

a_3=1

a_4=0

a_5=1

a_6=

```
109 | 2
 54 | 2
 27 | 2
 13 | 2
  6 | 2
  3 | 2
  1
```

A_{10}=109= A_2= $a_6a_5a_4a_3a_2a_1a_0$=1101101

Обернене перетворення виконується таким чином:

$$A_2 = a_6a_5a_4a_3a_2a_1a_0 = 1101101 = 2^0 + 0^1 + 2^2 + 2^3 + 0^4 + 2^5 + 2^6 = 109 = A_{10}$$

3.2 Завдання до практичної роботи

Отримати індивідуальне завдання.

Виконати операції над двійково-десятковими числами.

Контрольні запитання

1. Які особливості десяткової системи Ви можете назвати?
2. Що таке двійковий код?
3. Пояснити принцип перетворення двійково-десяткових чисел.
4. Як здійснюється подання цих чисел в цифрових системах?

ЛАБОРАТОРНА РОБОТА № 4

Двійкова арифметика. Додавання, віднімання та множення двійкових чисел. Додатковий числовий код

Мета роботи: Вивчити методику виконання арифметичних дій додавання, віднімання, множення та ділення двійкових чисел, ознайомитись із поданням додатних та від'ємних чисел, цілої та дробової частини чисел.

4.1 Теоретичні відомості про арифметичні дії у двійковій системі числення

4.1.1 Додавання

Додавання двійкових кодів робиться побітово на основі таких співвідношень: 0+0=0; 0+1=1; 1+0=1; 1+1=0 і 1 – в перенесення (в результаті 10).

Наприклад:

	1 перенесення			111 перенесення	
+	9	1001	+	7	0111
	5	0101		1	0001
	14	1110		8	1000

4.1.2 Віднімання

Цю дію можна виконувати так само як і в десяткових кодах, позичаючи 1 старшого розряду:

	1 позичаємо	
-	10	1010
	5	0101
	5	0101

Але для багаторозрядних кодів процедура дуже ускладнюється, коли доводиться позичати не з сусіднього старшого розряду, а з більш старших розрядів. Тому в цифровій техніці вводиться поняття додаткового коду, який дозволяє абсолютно однаково виконувати операції додавання і віднімання. Для вказання знака коду використовується найстарший його розряд. У позитивному коді старший розряд дорівнює нулю, а в негативному – одиниці. Додатковий код позитивного числа збігається з його прямим (звичайним) кодом. Додатковий код від'ємного числа отримується шляхом інверсії прямого коду і додавання до результату одиниці.

Наприклад: прямий і додатковий код числа +5 дорівнює 0101, додатковий код числа -5 дорівнює +1 = 1010+1 = 1011.

Старший розряд «1» вказує, що це код від'ємного числа. Код називається додатковим тому, що він доповнює n-розрядний прямий код до значення 2n.

У наведеному прикладі 0101+1011 = 100002 = 24.

Є інший спосіб визначення додаткового коду, що дещо швидше приводить до мети. Розряди прямого коду переписуються справа наліво, починаючи з молодшого розряду до першої 1 (знакової), інші розряди інвертуються.

Наприклад, 10110пр = 01010дод.

Віднімання двійкових кодів зводиться до складання позитивних і негативних кодів і виконується як складання їх додаткових кодів. При виконанні цієї операції дуже важливо простежити, щоб результат дії над кодами не спотворив знаковий розряд. Тому має бути певний запас нульових розрядів, розташованих після знакового розряду. У нижченаведених прикладах операнди займають всього 4 розряди, але використовуватимемо восьмирозрядні коди.

Розглянемо різні ситуації при відніманні.

1. Розрахуємо в двійкових кодах результат операції 7-5 = 7(- 5).

Визначимо додатковий код - 5 = - 00000101пр = 11111011дод.

Тоді:

$$710\text{-}510 = 00000111_{дод}+11111011_{дод} = 1.00000010_{дод} = 00000010_{пр} = 210.$$

Перенесення 1 в розряд відкидається. Знаковий розряд дорівнює 0, тому результат – додатне число 2, у якого прямий код такий самий, як і додатковий.

2. Визначимо результат операції:

$$510\text{-}710 = 00000101_{пр}\text{-}00000111_{пр} = 00000101_{доп}+11111001_{доп} = 11111110_{доп} = -00000010_{пр} = -210.$$

Тут D7 = 1, результат від'ємний, тому додатковий код переводиться в прямий. Це виконується за тим самим правилом, що і переведення прямого коду в додатковий.

3. Знайдемо:

$$-510 - 710 = 11111011_{дод} + 11111001_{дод} = 1.11110100_{дод} = 00001100_{пр} = -1210.$$

4.1.3 Множення

Операція виконується так само, як і для десяткових кодів: множене множиться на кожен розряд множника і результати додаються із зсувом.

Можна множити, починаючи з молодших розрядів із зсувом вліво, або із старших із зсувом вправо.

$6_{10} \cdot 7_{10}$

111	111
×110	×110
000	111
111	111
×111	× 000
$101010_2 = 42_{10}$	101010

Числа зі знаком множаться в прямому коді, а знак визначається як сума за модулем 2 знакових розрядів.

4.1.4 Ділення

Виконується як віднімання із зсувом. Наприклад:

18:6 = 3 10010 : 110 - 110 11 110 - 110 000	22:4 = 5,5 10110 : 100 - 100 101,1 110 - 100 10,0 - 100 000

Тут дробова частина – це від'ємне значення числа 2.

4.2 Завдання до практичної роботи

Отримати індивідуальне завдання.

Виконати операції над двійковими числами.

Контрольні запитання:

1. Як подаються додатне та від’ємне значення числа у двійковій формі?
2. Поясніть принцип виконання арифметичної дії додавання двох чисел у двійковій формі.
3. Поясніть принцип виконання арифметичної дії віднімання двох чисел у двійковій формі.
4. Поясніть принцип виконання арифметичної дії множення двох чисел у двійковій формі.
5. Поясніть принцип виконання арифметичної дії ділення двох чисел у двійковій формі.
6. Що таке додатковий числовий код?

ЛАБОРАТОРНА РОБОТА № 5

Шістнадцяткова система числення. Шістнадцятково-десяткові перетворення чисел. Шістнадцятково-двійкові перетворення чисел

Мета роботи: знайомство з шістнадцятковою системою числення, набуття практичних навичок з шістнадцятково–десяткових та шістнадцятково–двійкових перетворень.

5.1 Теоретичні відомості

Шістнадцяткова система числення

Шістнадцяткова система числення є системою з основою 16 та містить 16 символів: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. У табл. 1 наведено двійкові та шістнадцяткові еквіваленти 16 перших десяткових чисел. Кожну шістнадцяткову цифру подають єдиною комбінацією чотирьох двійкових цифр. Так, шістнадцятковим еквівалентом двійкового числа 10011110_2 число $9E_{16}$. Це означає, що старшу тетраду (4 старші розряди) 1001 двійкового числа записують як 9_{16}, а молодшу тетраду 1110 – як E_{16}.

Шістнадцяткова система числення – це позиційна система числення, кожне число в якій записується за допомогою 16-ти символів. Цю систему часто називають також *hex* (початкові літери англ. *hexadecimal* — шіснадцятковий). Спочатку планувалось вживати латинське *sexa* замість *hexa*, проте це слово сприймалось неоднозначно. Для запису чисел в цій системі окрім 10 арабських цифр (від *0* до *9*) використовують 6 літер латинської абетки: *A*, *B*, *C*, *D*, *E*, *F*.

Запис числа формується за загальним принципом: на n-й позиції (справа наліво від 0) стоїть цифра, що відповідає кількості n-х степенів шістнадцяти у цьому числі. Наприклад, число записане в десятковій системі як «1000», в *hex* записується як «3E8», де:

$$«3»\times 162 + «14»\times 161 + «8»\times 160 = 768 + 224 + 8 = 1000.$$

Шістнадцяткова система числення широко вживана в інформатиці, оскільки значення кожного байта можна записати у вигляді двох цифр шістнадцяткової системи. Таким чином значення послідовних байтів можна подати у вигляді списку двозначних чисел. В той самий час запис 4 бітів можна подати однією шістнадцятковою цифрою.

Двійкові та шістнадцяткові еквіваленти десяткових чисел наведено у таблиці 5.1.

Приклад 1. Перетворити двійкове число 111010_2 на шістандцятковий еквівалент. Для перетворення двійкового числа на шістандцятковий еквівалент потрібно поділити його на тетради, починаючи з наймолодшого

розряду, а потім кожну тетроду замінити еквівалентною шістнадцятковою цифрою. Значення молодшої тетради $1010_2 = A_{16}$, старшої – $0011_2 = 3_{16}$. Отже $111010_2 = 3A_{16}$.

Таблиця 5.1 – Двійкові та шістнадцяткові еквіваленти десяткових чисел

Десяткове число		Шістнадцятковий еквівалент	Двійковий еквівалент			
Значення ваг позицій						
10^1	10^0	16^0	2^3	2^2	2^1	2^0
0	0	0	0	0	0	0
0	1	1	0	0	0	1
0	2	2	0	0	1	0
0	3	3	0	0	1	1
0	4	4	0	1	0	0
0	5	5	0	1	0	1
0	6	6	0	1	1	0
0	7	7	0	1	1	1
0	8	8	1	0	0	0
0	9	9	1	0	0	1
1	0	A	1	0	1	0
1	1	B	1	0	1	1
1	2	C	1	1	0	0
1	3	D	1	1	0	1
1	4	E	1	1	1	0
1	5	F	1	1	1	1

Приклад 2. Перетворити шістнадцяткове число $7F_{16}$ на двійковий еквівалент. Для перетворення шістнадцяткового числа на двійковий еквівалент кожну шістнадцяткову цифру потрібно замінити на двійковий еквівалент – тетраду (див. табл. 1). Еквівалентом шістнадцяткової цифри 7_{16} є двійкове число 0111_2, а цифри F_{16} – число 1111_2. Отже, $7F_{16} = 11110111_2$.

Приклад 3. Перетворити шістнадцяткове число $2C6E_{16}$ на десятковий еквівалент. Перетворення виконують згідно з табл. 1. Кожну цифру шістнадцяткового числа множать на відповідну вагу позиції. Сума цих добутків дає десяткове число 11374_{10}.

Процес перетворення десяткового числа 11374 на шістнадцяткове наведено у таблиці 5.2.

Таблиця 5.2 – Перетворення десяткового числа на шістнадцяткове

Значення позицій	16^3	16^2	16^1	16^0
Значення ваг позицій	4096	256	16	1
Шістнадцяткове число	2	3	6	E
Десяткове число	2×4096	+3×256	+6×16	+ 1 × 14 = = 11374_{10}

Приклад 4. Перетворити десяткове число 15797 на шістнадцятковий еквівалент.

При перетворенні десяткове число 15797_{10} ділять на 16, що дає частку 987_{10} і остачу $5_{10} = 5_{16}$. Таким чином, молодший розряд шістнадцяткового числа має значення 5. Частка 987_{10} стає діленим і її знову ділять на 16, що дає частку 61_{10} і остачу $11_{10} = B_{16}$, яка стає значенням другого розряду шістнадцяткового числа. Ділення 61_{10} на 16 дає частку 3_{10} і остачу $13_{10} = D_{16}$. Ділення 3_{10} на 16 дає частку 0 і остачу $3_{10} = 3_{16}$. Оскільки остання частка дорівнює 0, то цифра 3_{16} стає значенням старшого розряду шістнадцяткового числа:

$15797_{10} : 16 = 987_{10}$ остача $5_{10} = 5_{16}$ ----------▼

$987_{10} : 16 = 61_{10}$ остача $11_{10} = B_{16}$----------▼

$61_{10} : 16 = 3_{10}$ остача $13_{10} = D_{16}$----------▼

$3_{10} : 16 = 0$ остача $3_{10} = 3_{16}$ ----------▼

3 D B 5

Отже, $15797_{10} = 3DB5_{16}$.

5.2 Перетворення чисел із шістнадцяткової системи у десяткову

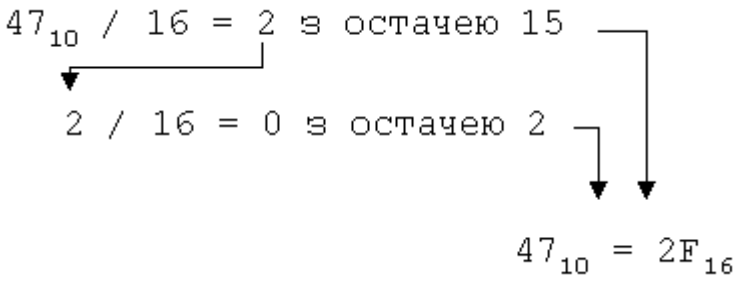
Розглянемо, як перетворити шістнадцяткове число $2DB_{16}$ у його десятковий еквівалент. Ваги перших трьох розрядів шістнадцяткового числа відповідно дорівнюють 256, 16 і 1.

Вага розряду	256	16	1
Шістнадцяткове число	2	D	B_{16}
	× 256	× 16	× 1
	2	13	11
	512	208	11
Десяткове число	512 + 208 +11 = 731_{10}		

У цьому шістнадцятковому числі є одинадцять одиниць, в розряді з вагою 16 стоїть число 13, яке при множенні на вагу розряду дає число 208, а двійка в розряді з вагою 256 позначає число 512. Додаючи 11+208+512, знаходимо число 731_{10}. Таким чином, $2DB_{16}=731_{10}$.

5.3 Перетворення чисел із десяткової системи в шістнадцяткову

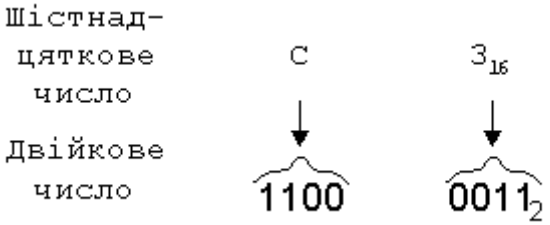
Розглянемо тепер зворотне перетворення десяткового числа 47 в його шістнадцятковий еквівалент. Покажемо процедуру послідовних ділень на 16.



Перше ділення десяткового числа 47 на 16 дає частку 2 і остачу 15. Цю остачу (тобто число F в шістнадцятковій системі) потрібно взяти як останню вагому цифру шуканого шістнадцяткового числа. Частку (у цьому випадку 2) потрібно взяти далі як ділене і знову розділити його на 16. У результаті отримаємо частку 0 з остачею 2; цю цифру потрібно вважати наступною цифрою шуканого шістнадцяткового числа. На цьому процес перетворення закінчується, оскільки отримано частку, що дорівнює 0. Запишемо результат: $47_{10}=2F_{16}$.

5.4 Перетворення чисел із шістнадцяткової системи у двійкову

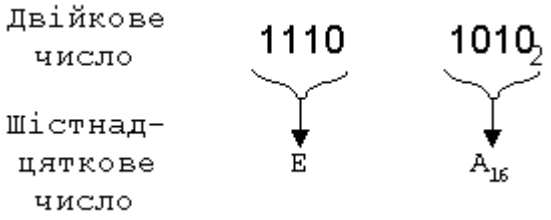
Перетворення чисел із шістнадцяткової системи у двійкову та із двійкової системи у шістнадцяткову – це типова операція, що реалізується у мікропроцесорах та ЕОМ. Розглянемо це перетворення на прикладі числа $C3_{16}$ і знайдемо еквівалентне йому двійкове число. Нижче показано, як кожний символ шіснадцяткового числа перетворюється в його чотиризначний двійковий еквівалент (див. таблицю 5.1):



Шістнадцятковий символ С відповідає чотирирозрядному двійковому числу 1100, а шіснадцятковий символ 3 – двійковому числу 0011. Об'єднуючи ці дві двійкові групи, отримуємо $C3_{16}=11000011_2$.

5.5 Перетворення чисел із двійкової системи у шістнадцяткову

Займемося тепер оберненою процедурою і перетворимо двійкове число 11101010 на еквівалентне йому шістнадцяткове. Двійкове число розділяється на чотиризначні групи, починаючи з двійкової точки. Далі кожна двійкова група переводиться в своє еквівалентне шістнадцяткове подання за допомогою таблиці 5.1:



В результаті маємо $11101010_2=EA_{16}$.

5.6 Завдання до практичної роботи

Отримати індивідуальне завдання.

Виконати операції над числами в заданій викладачем системі числення.

Контрольні запитання:

1. Що таке шістнадцяткова система числення?
2. Яка сфера застосування шістнадцяткової системи числення?
3. Поясніть принцип виконання перетворення чисел із десяткової системи числення у шістнадцяткову.
4. Поясніть принцип виконання перетворення чисел із шістнадцяткової системи числення у десяткову.
5. Поясніть принцип виконання перетворення чисел із шістнадцяткової системи числення у двійкову.

ЛАБОРАТОРНА РОБОТА № 6

Вивчення будови та принципів роботи апаратно-програмних засобів Arduino

Мета роботи: ознайомитись з особливостями будови та функціонування мікроконтролера Arduino (Arduino Uno), що використовується для створення простих електронних систем автоматики і робототехніки.

6.1 Загальні відомості про Arduino

Ардуіно (Arduino) – це назва апаратно-програмних засобів для створення простих електронних систем автоматики і робототехніки. Система має повністю відкриту архітектуру і орієнтована на непрофесійних користувачів.

Програмна частина Ардуіно складається з інтегрованого програмного середовища (IDE), що дозволяє писати, компілювати програми, а також завантажувати їх в апаратуру.

Апаратна частина (див. рис. 6.1) являє собою електронні плати з мікроконтролером, супутніми елементами (стабілізатор живлення, кварцовий резонатор, блокувальні конденсатори і т. п.), портом для зв'язку з персональним комп'ютером, роз'ємами для сигналів введення – виведення і т. п.

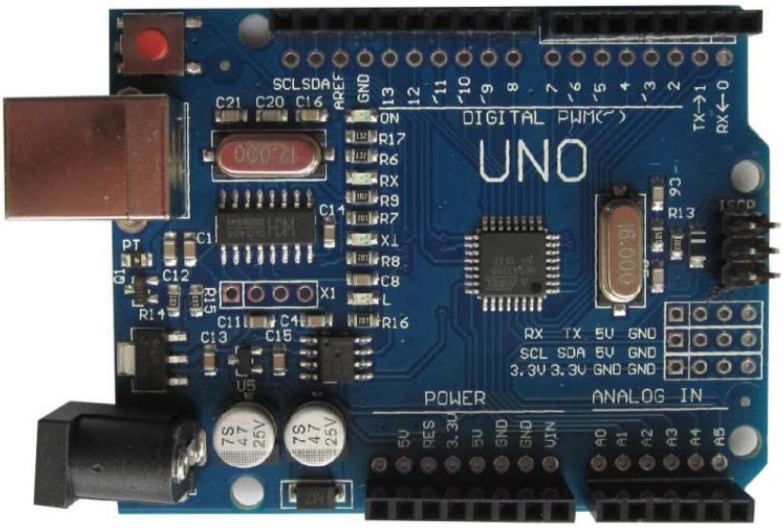


Рисунок 6.1 – Загальний вигляд мікроконтролера Arduino Uno

У платах Ардуіно використовуються мікроконтролери Atmel AVR з прошитим у них завантажувачем. За допомогою завантажувача записується програма в мікроконтролер з персонального комп'ютера без застосування апаратних програматорів.

Для програмуванні Ардуіно використовується мова C / C++, з деякими особливостями.

6.2 Загальна інформація про контролер ATmega328

Arduino UNO R3 виконаний на мікроконтролері ATmega328, що характеризується наявністю у нього:

• 14 цифрових портів входу-виходу (6 з них підтримують режим широтно-імпульсної модуляції (ШІМ));

• 6 аналогових входів;

• частота тактів до 16 МГц;

• USB порт;

• роз'єм для живлення;

• роз'єм внутрішньосхемного програмування;

• кнопка скидання (Reset).

У плати є всі необхідні компоненти для забезпечення роботи мікроконтролера. Досить підключити USB кабель до комп'ютера і подати живлення. Мікроконтролер встановлений на колодці, що дозволяє легко замінити його в разі виходу з ладу.

Таблиця 6.1 – Технічні характеристики мікроконтролера Ардуіно

Тип мікроконтролера	ATmega328P
Напруга живлення мікроконтролера	5 В
Рекомендована напруга живлення плати	7 – 12 В
Гранично допустима напруга живлення плати	6 – 20 В
Цифрові входи-виходи	14 (з них 6 підтримують ШІМ)
Виходи ШІМ модуляції	6
Аналогові входи	6
Допустимий струм цифрових виходів	20 мА
Допустимий струм виходу 3,3 В	50 мА
Обсяг флеш-пам'яті (FLASH)	32 кБ (із яких 0,5 кБ використовується завантажувачем)
Обсяг оперативної пам'яті (SRAM)	2 кБ
Обсяг енергонезалежної пам'яті (EEPROM)	1 кБ
Частота тактів	16 мГц
Габарити плати: Д×Ш	68,6 мм × 53,4 мм

6.3 Програмування мікроконтролера Ардуіно

Контролер програмується з інтегрованого середовища програмного забезпечення Ардуіно (IDE). Програмування відбувається під управлінням резидентного завантажувача за протоколом STK500. Апаратний програматор при цьому не потрібен.

Мікроконтролер можна запрограмувати через роз'єм для внутрішньосхемного програматора ICSP, не використовуючи завантажувач.

6.4 Відмінність від інших контролерів Ардуіно

Arduino UNO R3, на відміну від попередніх версій, не використовує для підключення до комп'ютера міст USB-UART FTDI. Цю функцію в ньому виконує мікроконтролер ATmega16U2.

6.5 Система живлення мікроконтролера Ардуіно

Плата UNO може отримувати живлення від USB-порту або від зовнішнього джерела. Джерело живлення вибирається автоматично. Як зовнішнє джерело живлення може використовуватися мережевий адаптер або батарея. Адаптер підключається через роз'єм діаметром 2,1 мм (центральний контакт – позитивний). Батарея підключається до контактів GND і Vin роз'єму POWER.

Напруга зовнішнього джерела живлення може бути в діапазоні 6 – 20 В. Але рекомендується не допускати зниження напруги нижче 7 В через нестабільну роботу пристрою. Також не бажано підвищувати напругу живлення більше 12 В, тому що може перегрітися стабілізатор і вийти з ладу. Тобто рекомендований діапазон напруги живлення 7 – 12 В.

Для підключення живлення можуть бути використані такі контактні групи:

Vin	Живлення плати від зовнішнього джерела живлення. Не пов'язане з живленням 5 В від USB або виходами інших стабілізаторів. Через цей контакт можна отримувати живлення для свого пристрою, якщо плата живиться від адаптера
5 V	Вихід стабілізатора напруги плати. На ньому напруга 5 В при будь-якому способі живлення. Живити плату через цей контакт не рекомендується, тому що не використовується стабілізатор, що може призвести до виходу мікроконтролера з ладу
3 V 3	Напруга 3,3 В від стабілізатора напруги на платі. Гранично допустимий струм споживання від цього контакту 50 мА
GND	Загальний провід
IOREF	На цьому контакті виводиться інформація щодо робочої напруги плати. Плата розширення може зчитати значення сигналу і переключитися на режим живлення 5 В або 3,3 В

6.6 Пам'ять мікроконтролера Ардуіно

У мікроконтролера Ардуіно три типи пам'яті:

• 32 кБ флеш (FLASH);

• 2 кБ оперативної пам'яті (SRAM);

• 1 кБ незалежної пам'яті (EEPROM).

6.7 Входи і виходи мікроконтролера Ардуіно

Кожен з 14 цифрових виводів (рис. 6.2) може бути використаний як вихід або вхід. Рівень напруги на виводах 5 В. Рекомендується вхідний та вихідний струми кожного виводу обмежувати на рівні 20 мА.

Увага! Гранично допустиме значення цього параметра становить 40 мА.

Кожен вивід має внутрішній підтягувальний резистор опором 20–50 кОм. Резистор може бути відключений програмно.

Деякі виводи можуть виконувати додаткові функції.

Послідовний інтерфейс: виводи 0 (Rx) і 1 (Tx). Використовуються для прийому (**Rx**) і передачі (**Tx**) послідовних даних логічних рівнів **TTL**. Ці виводи підключені до виводів передачі даних мікросхеми ATmega16U2, яку використовують як міст USB-UART.

Зовнішні переривання: виводи 2 і 3. Ці виводи можуть бути використані як входи зовнішніх переривань. Програмно можуть бути встановлені на переривання за низьким рівнем, позитивним чи негативним фронтом або на зміну рівня сигналу.

ШІМ: виводи 3, 5, 6, 9, 10, 11. Чи можуть працювати в режимі ШІМ модуляції з роздільною здатністю 8 розрядів.

Послідовний інтерфейс SPI: виводи 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).

Світлодіод: вивід 13. Світлодіод, підключений до виводу 13. Світиться при високому рівні сигналу на виводі.

Інтерфейс TWI: вивід A4 або SDA і A5 або SCL. Комунікаційний інтерфейс **TWI.**

У плати Arduino UNO є 6 аналогових входів, позначених **A0-A5.** Роздільна здатність аналогового цифрового перетворення 10 розрядів. За замовчуванням, вхідна напруга вимірюється відносно землі в діапазоні 0–5 В, але може бути змінена за допомогою виводу AREF і програмних налаштувань.

Ще 2 виводи плати мають функції:

AREF. Опорна напруга АЦП мікроконтролера.

RESET. Низький рівень на цьому виводі викликає скидання мікроконтролера.

6.7.1 Комунікаційні інтерфейси

Модуль Arduino UNO має засоби для зв'язку з комп'ютером, з іншою платою UNO або з іншими мікроконтролерами. Для цього на платі існує

інтерфейс UART з логічними рівнями TTL (5 В), пов'язаний з виводами 0 (RX) і 1 (TX). Мікросхема ATmega16U2 на платі зв'язує UART інтерфейс з USB-портом комп'ютера. При підключенні до порту комп'ютера з'являється віртуальний COM порт, через який програми комп'ютера працюють з Ардуіно. Прошивка ATmega16U2 використовує стандартні драйвери USB-COM і установлення додаткових драйверів не потрібно. Для операційної системи Windows необхідний відповідний .inf файл. В інтегроване середовище програмного забезпечення Ардуіно (IDE) вбудовано монітор обміну по послідовному інтерфейсу, який дозволяє надсилати і отримувати з плати прості текстові дані. На платі є світлодіоди RX і TX, які слугують індикаторами стану відповідних сигналів для зв'язку через USB (але не для послідовного інтерфейсу на виводах 0 і 1).

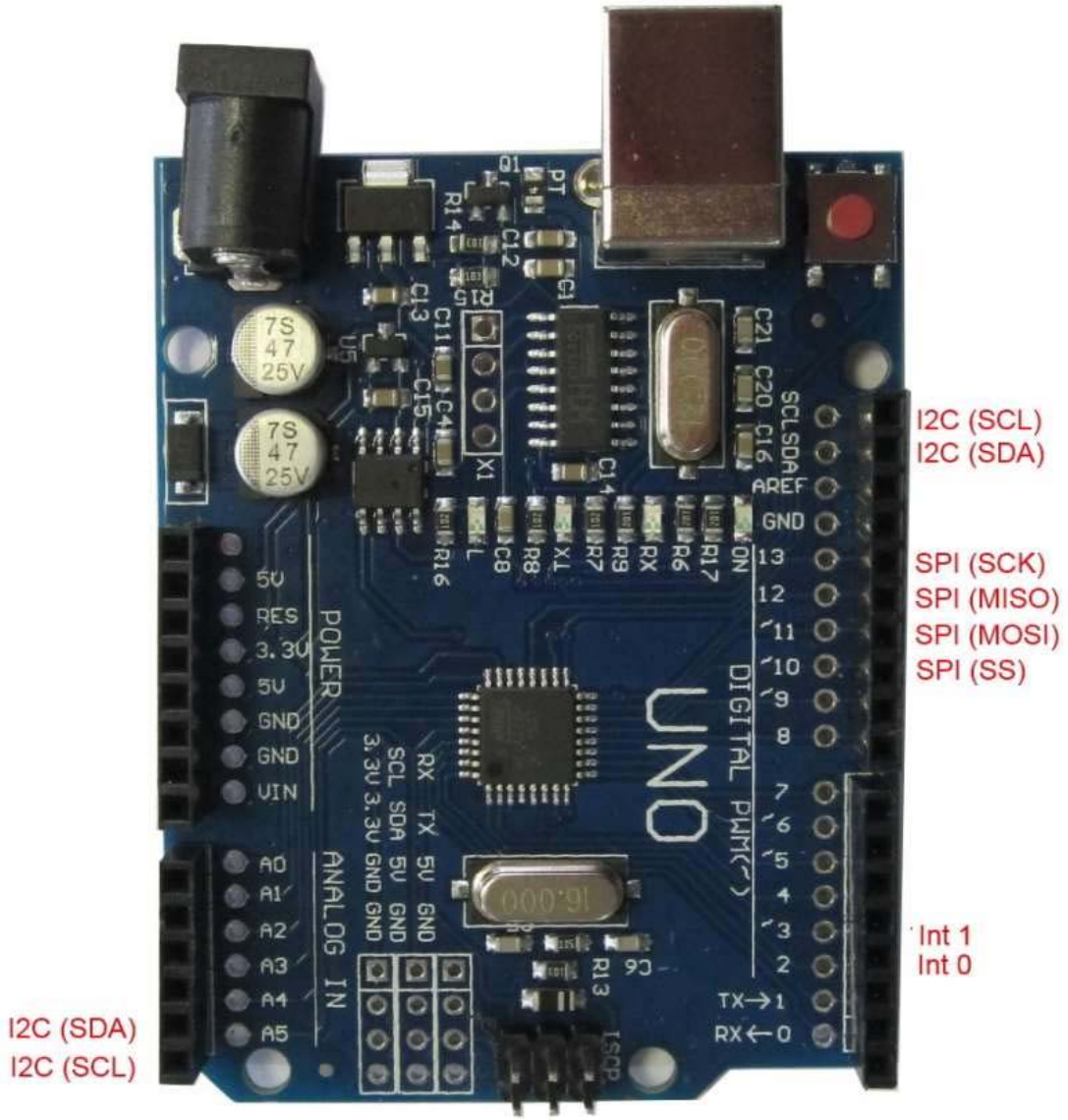


Рисунок 6.2 – Позначення інтерфейсів на платі мікроконтролера Arduino Uno

Мікроконтролер ATmega328 також підтримує комунікаційні інтерфейси I2C (TWI) і SPI.

Принципова електрична схема мікроконтролера Ардуіно наведена на рис. 6.3

6.7.2 Автоматичне (програмне) скидання

Для того, щоб не доводилось кожного разу перед завантаженням програми натискати кнопку скидання, на платі UNO реалізована апаратна функція скидання, що ініціюється з підключеного комп'ютера. Один із сигналів управління потоком даних (DTR) мікросхеми ATmega16U2 підключений до виходу скидання мікроконтролера ATmega328 через конденсатор ємністю 0,1 мкФ.

Коли сигнал DTR переходить в низький стан, формується імпульс скидання мікроконтролера. Це рішення дозволяє завантажувати програму одним натисканням кнопки з інтегрованого середовища програмування Андроїд (IDE).

Але така функція може призводити до негативних наслідків. При підключенні плати UNO до комп'ютера з операційною системою Mac Os X або Linux, мікроконтролер буде скидатися при кожному з'єднанні програми з платою. Протягом половини секунди на платі UNO буде запущений завантажувач. Незважаючи на те, що програма завантажувача ігнорує сторонні дані, вона може прийняти кілька байтів з пакета відразу після установлення з'єднання. Якщо в програмі на платі Ардуіно передбачено отримання будь-яких даних при першому запуску, необхідно відправляти дані з затримкою приблизно на 1 секунду після з'єднання.

На модулі UNO існує доріжка, яку можна перерізати для відключення функції автоматичного скидання. Доріжка маркована написом «RESET-EN». Автоматичне скидання також можна заборонити, підключивши резистор опором 110 Ом між лінією живлення 5 В і виводом RESET.

6.8 Захист USB-порту від перевантажень

У платі Arduino UNO лінія живлення від інтерфейсу USB захищена відновлюваним запобіжником. При перевищенні струму понад 500 мА, запобіжник розриває коло до усунення короткого замикання.

6.9 Початок роботи з контролером Arduino UNO

Для установлення програмного забезпечення і підключення контролера Arduino UNO R3 до комп'ютера необхідні:

• плата контролера;

• USB кабель (зазвичай дається в комплекті);

• персональний комп'ютер з ОС (бажано Windows 7, Windows 8.1 або Windows 10) та підключений до інтернету.

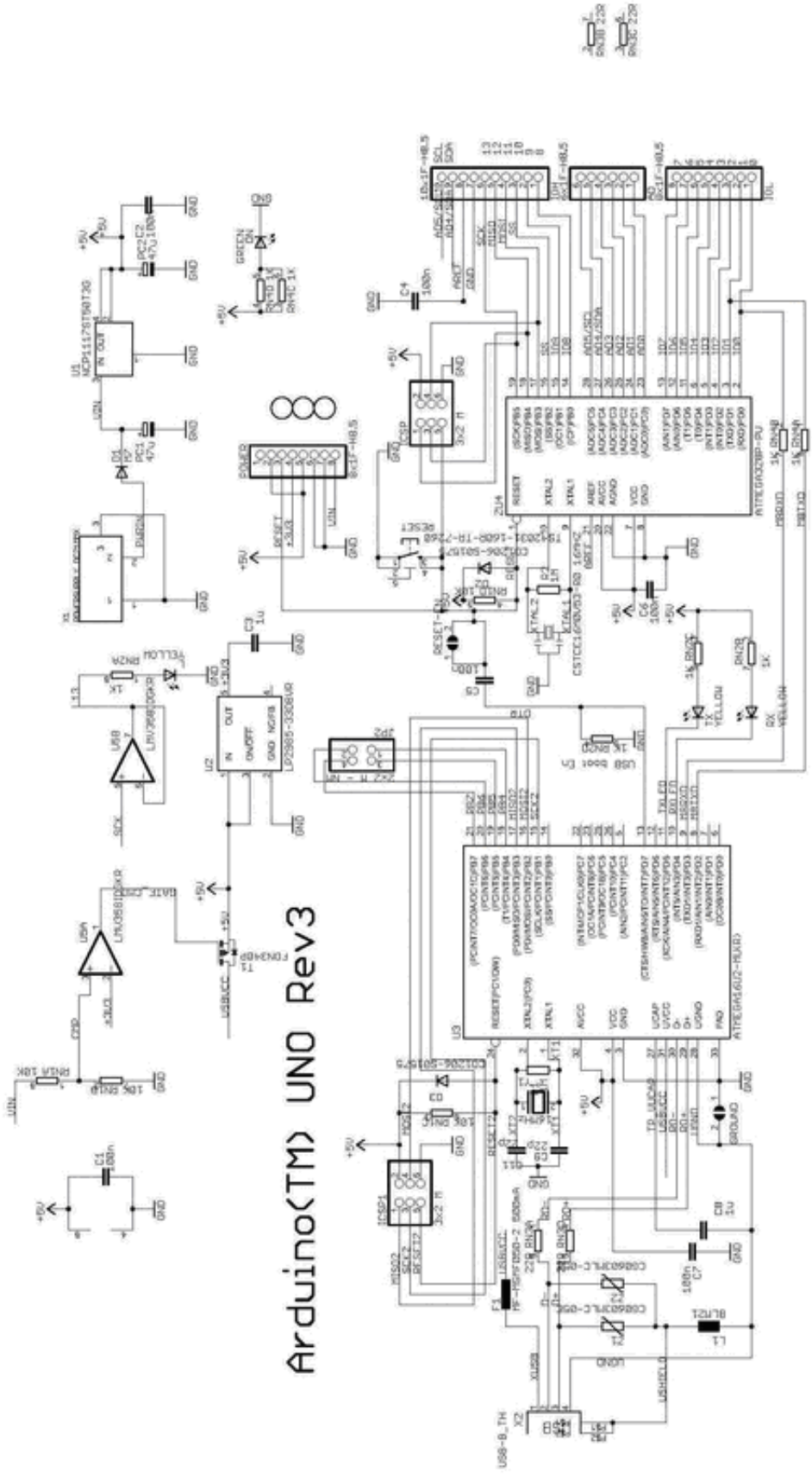


Рисунок 6.3 – Принципова електрична схема мікроконтролера Arduino

Плата може отримувати живлення від USB-порту комп'ютера, тому зовнішній блок живлення не потрібен.

6.9.1 Встановлення інтегрованого середовища розробки Arduino IDE

Передусім, необхідно завантажити останню версію програми. Завантажити ZIP архів можна з офіційного сайту підтримки систем Ардуіно.

6.9.2 Підключення плати Ардуіно

За допомогою USB кабелю підключіть плату до комп'ютера. Має засвітитися світлодіод (з маркуванням ON), який показує, що на плату надходить живлення.

6.9.3 Встановлення драйвера для ARDUINO UNO

Після підключення плати до комп'ютера Windows сама почне процес установлення драйвера. Якщо через короткий проміжок часу драйвер не буде встановлено (про це буде свідчити поява повідомлення про невдалу спробу), то драйвер потрібно встановлювати вручну. Для цього переходимо по шляху: Пуск → Панель управління → Система → Диспетчер пристроїв.

У розділі Порти (COM і LPT) має бути пристрій Arduino UNO з попереджувальним жовтим значком.

Клацаємо правою кнопкою миші по значку.

Вибираємо: Оновити драйвер.

Далі: Виконати пошук драйверів на цьому комп'ютері.

Вручну вказати місце розміщення драйвера. Файл ArduinoUNO.inf знаходиться в каталозі Drivers папки, куди розпакований архів.

У розділі Порти (COM і LPT) з'являється новий, віртуальний COM. Його номер потрібно запам'ятати.

6.9.4 Запуск інтегрованого середовища розробки Arduino IDE

Запускаємо файл arduino.exe.

Вибираємо тип плати Ардуіно: Інструменти → Плата → Arduino UNO (див. рис. 6.4).

Необхідно вказати номер COM порту: Інструменти → Порт (див. рис. 6.5).

Для перевірки роботи системи можна запустити перший скетч – миготливий світлодіод. Для цього необхідно виконати команди: Файл → Приклади → 01.Basics → Blink (див. рис. 6.6).

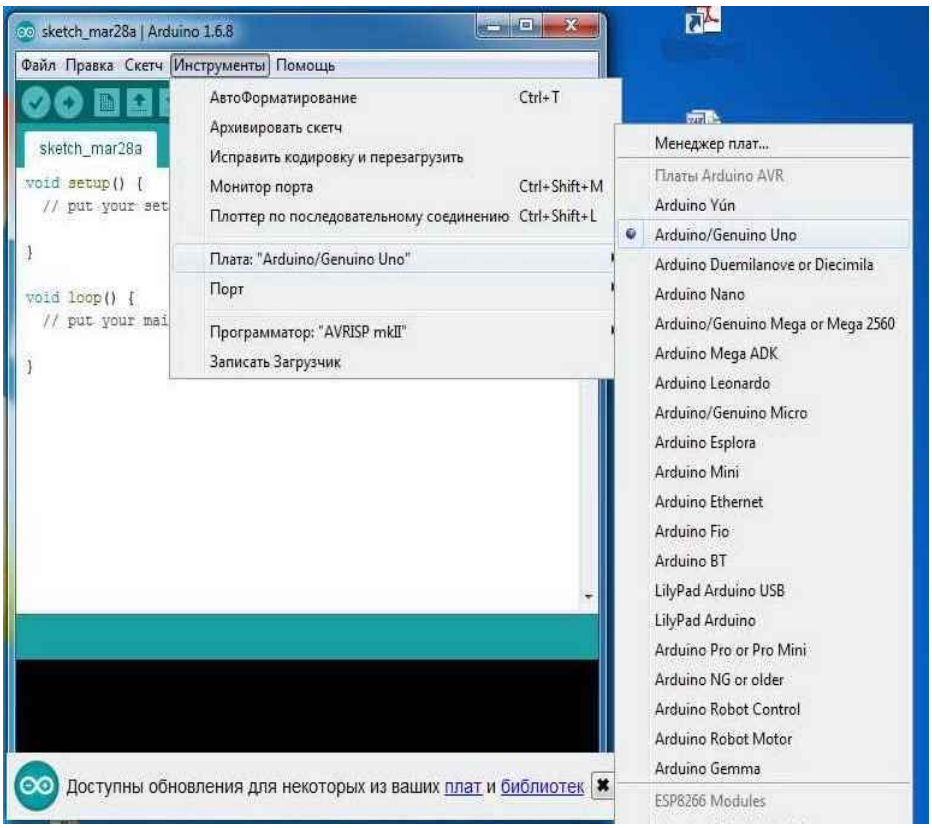


Рисунок 6.4 – Зовнішній вигляд вікна середовища розробки Arduino IDE для вибору типу плати

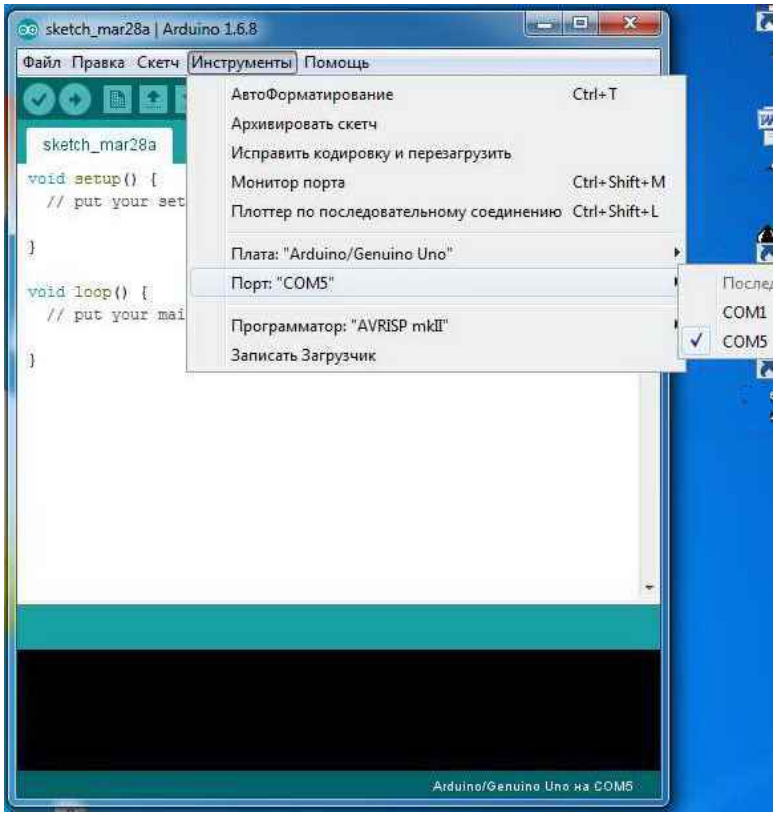


Рисунок 6.5 – Зовнішній вигляд вікна середовища розробки Arduino IDE для вибору COM порту

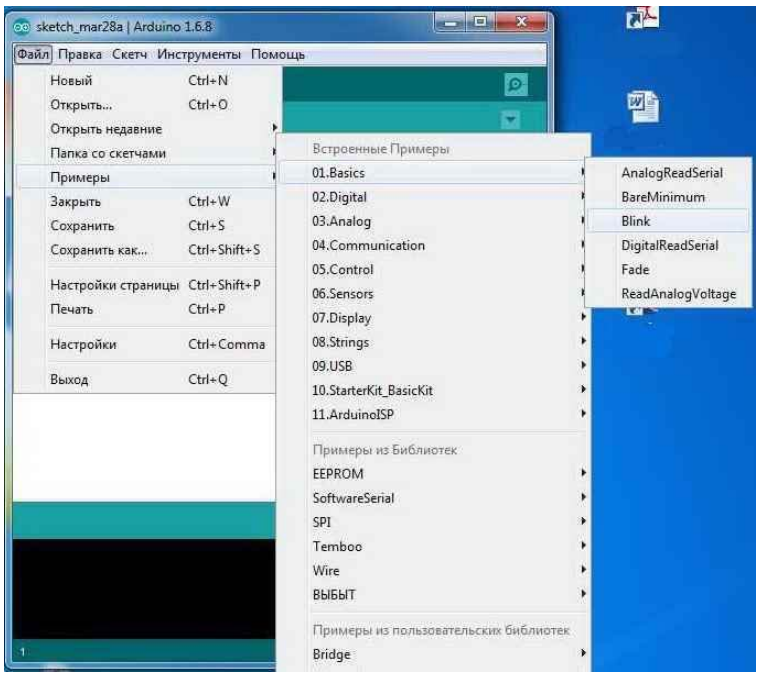


Рисунок 6.6 – Завантаження прикладу керівної програми у середовищі розробки Arduino IDE

Натискаємо кнопку **Завантаження** (див. рис. 6.7).

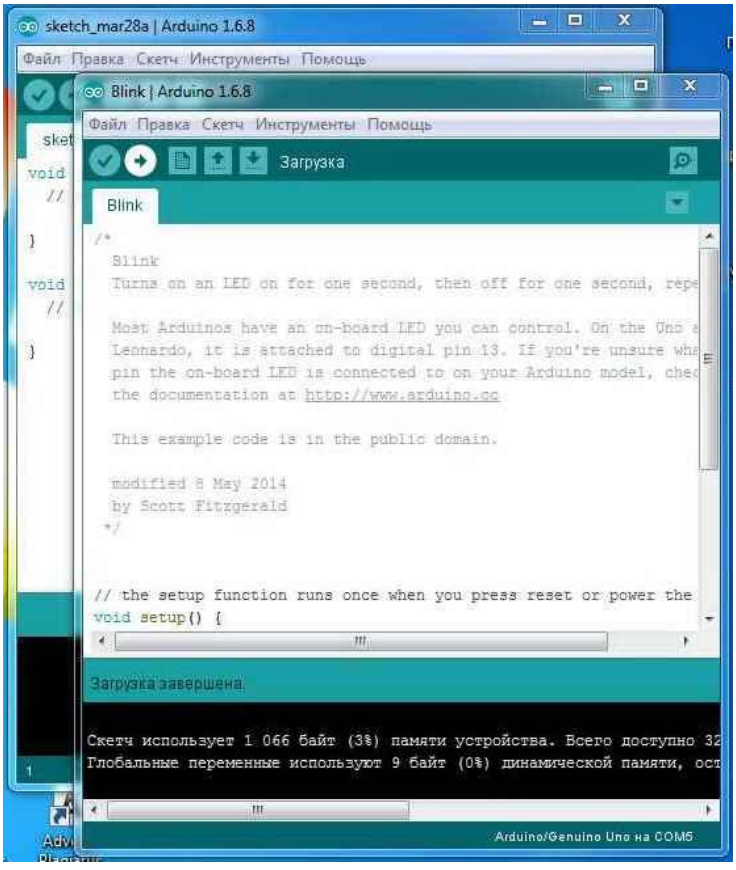


Рисунок 6.7 – Процес завантаження керівної програми у середовищі розробки Arduino IDE

Чекаємо поки програма завантажиться і світлодіод на платі, позначений буквою, літерою L, починає блимати приблизно раз на секунду – це буде означати, що всі операції виконано правильно.

6.10 Завдання до лабораторної роботи

1. Згідно з рекомендаціями викладача завантажити останню версію середовища розробки Arduino IDE (завантажити ZIP архів з офіційного сайту підтримки систем Ардуіно).

2. Виконати підключення контролера Arduino UNO R3 до комп'ютера за допомогою USB кабелю.

3. Користуючись рекомендаціями пункту 6.9.3 виконати встановлення драйвера для ARDUINO UNO.

4. Завантажити на контролер ARDUINO UNO керівну програму, користуючись прикладами, вбудованими у середовище розробки Arduino IDE.

Контрольні запитання

1. Коротко опишіть основне призначення мікроконтролера Arduino.
2. Що являє собою програмна частина мікроконтролера Arduino?
3. Що являє собою апаратна частина мікроконтролера Arduino?
4. Яка кількість цифрових портів входу–виходу міститься на платі мікроконтролера Arduino?
5. Яка кількість аналогових входів міститься на платі мікроконтролера Arduino?
6. Як здійснюється програмування мікроконтролера Ардуіно?
7. Використовуючи рис. 6.2, поясніть функції входів і виходів мікроконтролера Ардуіно.
8. Назвіть основні комунікаційні інтерфейси мікроконтролера Ардуіно
9. Як реалізовано в мікроконтролері Ардуіно автоматичне (програмне) скидання?

ЛАБОРАТОРНА РОБОТА №7

Основи програмування мікроконтролера Arduino

Мета роботи: ознайомитись з особливостями мови програмування Arduino, навчитися писати прості програми керування та програмувати мікроконтролер Arduino.

7.1 Структура програми Ардуіно

Для програмуванні Ардуіно використовується мова C / C ++ з деякими особливостями.

Структура програми Ардуіно досить проста і в мінімальному варіанті складається з двох частин setup () і loop ().

```
void setup () {
// код виконується один раз при запуску програми
}
void loop () {
// основний код, виконується в циклі
}
```

Функція setup () виконується один раз, при включенні живлення або скиданні контролера. Зазвичай в ній відбуваються початкові установлення змінних, регістрів. Функція має бути присутня в програмі, навіть якщо в ній нічого немає.

Після завершення **setup ()** керування переходить до функції loop (). Вона в нескінченному циклі виконує команди, записані в її тілі (між фігурними дужками). Власне ці команди і роблять всі алгоритмічні дії контролера.

7.2 Початкові правила синтаксису мови С

; (крапка з комою) – вирази можуть містити як завгодно багато пропусків, переносів рядків. Ознакою завершення виразу є символ «крапка з комою».

Увага! Нижче наведено рівнозначні записи.

```
z = x + y;
z = x
+ y;
```

{} (фігурні дужки) – визначають блок функції або виразів. Наприклад, у функціях setup () і loop ().

/* ... */ – блок багаторядкового коментаря, обов'язково закрити.

Приклад.
/* Це блок коментаря */

// – однорядковий коментар, закривати не треба, діє до кінця рядка.
Приклад.
// це однорядковий коментар

7.3 Змінні і типи даних

Змінна – це комірка оперативної пам'яті, в якій зберігається інформація. Програма використовує змінні для зберігання проміжних даних обчислень. Для обчислень можуть бути використані дані різних форматів, різної розрядності, тому у мові С є типи змінних, наведені у таб. 7.1.

Таблиця 7.1 – Типи змінних, що використовуються при програмуванні мовою Arduino

Тип даних	Розрядність, біт	Діапазон чисел
boolean	8	true, false
char	8	-128 … 127
unsigned char	8	0 … 255
byte	8	0 … 255
int	16	-32768 … 32767
unsigned int	16	0 … 65535
word	16	0 … 65535
long	32	-2147483648 … 2147483647
unsigned long	32	0 … 4294967295
short	16	-32768 … 32767
float	32	-3.4028235+38 … 3.4028235+38
double	32	-3.4028235+38 … 3.4028235+38

Типи даних вибираються, виходячи з необхідної точності обчислень, форматів даних і т. п. Не варто, наприклад, для лічильника, який рахує до 100, вибирати тип long. Працювати ця функція буде, але операція займе недоцільно великий обсяг пам'яті даних і програм та знадобиться значно більше часу.

7.4 Оголошення змінних

Вказується тип даних, а потім ім'я змінної.

Приклад.

```
int x; // оголошення змінної з ім'ям x типу int
float widthBox; // оголошення змінної з ім'ям widthBox типу float
```

Всі змінні мають бути оголошені до того як будуть використовуватися.

Змінна може бути оголошена в будь-якій частині програми, але від цього залежить, які блоки програми зможуть її використовувати. Тобто у змінних є області видимості.

• Змінні, оголошені на початку програми, до функції void setup (), вважаються глобальними і доступні в будь-якому місці програми.

• Локальні змінні оголошуються всередині функцій або таких блоків, як цикл for, і можуть використовуватися тільки в оголошених блоках. Можливі кілька змінних з одним ім'ям, але різними областями видимості.

Приклад.

```
int mode; // змінна доступна всім функціям
void setup () {
// порожній блок, початкові установки не потрібні
}
void loop () {
long count; // змінна count доступна тільки в функції loop ()
for (int i = 0; i <10;) // змінна i доступна тільки всередині циклу
  {
    i ++;
   }
}
```

При оголошенні змінної можна задати її початкове значення (проініціалізувати).

Приклад.

```
int x = 0; // оголошується змінна x з початковим значенням 0
char d = 'a'; // оголошується змінна d з початковим значенням, що
// дорівнює коду символа «a»
```

При арифметичних операціях з різними типами даних відбувається автоматичне перетворення типів даних. Але краще завжди використовувати явне перетворення.

Приклад.

```
int x; // змінна int
char y; // змінна char
```

```
int z; // змінна int
z = x + (int) y; // змінна y явно перетворена в int
```

7.5 Основні операції, що використовуються при написанні керівної програми мовою Arduino

Основні операції наведено у таб. 7.2

Таблиця 7.2 – Основні операції, що можуть використовуватись при написанні керівної програми мовою Arduino

Арифметичні операції	
=	рівність
+	додавання
-	віднімання
*	множення
/	ділення
%	Залишок від ділення
Операції відношень	
==	дорівнює
!=	не дорівнює
<	менше
>	більше
<=	менше або дорівнює
>=	більше або дорівнює
Логічні операції	
&&	логічне І
||	логічне АБО
!	логічне НЕ
Операції над показниками	
*	непряма адресація
&	Отримання адреси змінної

Продовження таблиці 7.2

Бітові операції	
&	І
|	АБО
^	XOR або ВИКЛЮЧАЛЬНЕ АБО
~	ІНВЕРСІЯ (побітове НЕ)
<<	ЗСУВ ВЛІВО
>>	ЗСУВ ВПРАВО
Операції змішаного присвоєння	
++	інкрементування
--	декрементування
+=	додавання
-=	віднімання
*=	множення
/=	ділення
%=	залишок від ділення
&=	побітове І
|=	побітове АБО

7.6 Вибір варіантів керування програмою

Оператор IF – перевіряє умову в дужках і виконує наступний вираз або блок в фігурних дужках, якщо умова істинна.

Приклад.

```
if (x == 5) // якщо x = 5, то виконується z = 0
z = 0;
if (x> 5) // якщо x> 5, то виконується блок z = 0, y = 8;
{Z = 0; y = 8; }
```

IF ... ELSE – дозволяє зробити вибір між двох варіантів.

Приклад.

```
if (x> 5) // якщо x> 5, то виконується блок z = 0, y = 8;
 {
   z = 0;
```

```
  y = 8;
 }
 else // інакше виконується цей блок
 {
  z = 0;
  y = 0;
 }
```

ELSE IF – дозволяє зробити множинний вибір.

Приклад.

```
if (x> 5) // якщо x> 5, то виконується блок z = 0, y = 8;
  {
   z = 0;
   y = 8;
  }
else if (x> 20) // якщо x> 20, виконується цей блок
  {
  }
else // інакше виконується цей блок
  {
  z = 0;
  y = 0;
  }
```

SWITCH CASE – множинний вибір. Дозволяє порівняти змінну (у прикладі це x) з декількома константами (в прикладі 5 і 10) і виконати блок, в якому змінна дорівнює константі.

Приклад.

```
switch (x) {
case 5:
    // код виконується якщо x = 5
     break;
case 10:
    // код виконується якщо x = 10
    break;
default:
    // код виконується якщо не збігається жодне попереднє значення
    break;
}
```

Цикл FOR – конструкція дозволяє організовувати цикли з заданою кількістю ітерацій. Синтаксис виглядає так:

```
for (дія до початку циклу;
       умова продовження циклу;
```

```
        дія в кінці кожної ітерації) {
// код тіла циклу
}
```

Приклад циклу з 100 ітерацій.

```
for (i = 0; i <100; i ++) // початкове значення 0, кінцеве 99, крок 1
{
    sum = sum + i;
}
```

Цикл WHILE. Оператор дозволяє організовувати цикли з конструкцією:

```
while (вираз)
{
// код тіла циклу
}
```

Цикл виконується до тих пір, поки вираз в дужках істинний.

Приклад циклу на 10 ітерацій.

```
x = 0;
while (x <10)
 {
  // код тіла циклу
   x ++;
  }
```

DO WHILE – цикл з умовою на виході.

```
do
{
  // код тіла циклу
} While (вираз);
```

Цикл виконується поки вираз істинний.

BREAK – оператор виходу з циклу. Використовується для того, щоб перервати виконання циклів for, while, do while.

Приклад.

```
x = 0;
while (x <10)
{
  if (z> 20) break; // якщо z> 20, то вийти з циклу
// код тіла циклу
  x ++;
  }
```

GOTO – оператор безумовного переходу.

Приклад.

```
goto mitka1; // перехід на mitka1
.....
mitka1:
```

CONTINUE – пропуск операторів до кінця тіла циклу.

Приклад.

```
x = 0;
while (x <10)
{
// код тіла циклу
if (z> 20) continue;
// якщо z> 20, то повернутися на початок тіла циклу
// код тіла циклу
x ++;
 }
```

7.7 Масиви

Масив – це область пам'яті, де послідовно зберігаються кілька змінних.

Оголошується масив так:

```
int ages [10]; // масив з 10 змінних типу int
float weight [100]; // масив з 100 змінних типу float
```

При оголошенні масиви можна ініціалізувати:

```
int ages [10] = {23, 54, 34, 24, 45, 56, 23, 23, 27, 28};
```

Звертаються до змінних масивів так:

```
x = ages [5]; // x присвоюється значення з 5 елемента масиву.
ages [9] = 32; // 9 елементу масиву задається значення 32
```

Нумерація елементів масивів завжди з нуля.

7.8 Функції

Функції дозволяють виконувати одні і ті самі дії з різними даними. У функції є:

• ім'я, за яким її викликають;

• аргументи – дані, які функція використовує для обчислення;

• тип даних, що повертається функцією.

Описується призначена для користувача функція поза функцій setup () і loop ().

Приклад.

```
void setup () {
// код виконується один раз при запуску програми
}
```

```
void loop () {
 // основний код, виконується в циклі
}
/*оголошення користувацької функції з ім'ям functionName type functionName (type argument1, type argument1, ..., type argument)*/
  {
   // тіло функції
    return ();
  }
```

Приклад функції, що обчислює суму квадратів двох аргументів.

```
int sumQwadr (int x, int y)
{
 return (x * x + y * y);
}
```

Виклик функції відбувається так:

```
d = 2; b = 3;
z = sumQwadr (d, b); // в z буде сума квадратів змінних d і b
```

Функції бувають вбудовані, призначені для користувача та такі, що підключаються.

7.9 Рекомендації з оформлення програм на мові С

Головна мета зовнішнього оформлення програм – це поліпшити читаність програм, зменшити число формальних помилок. Тому для досягнення цієї мети можна сміливо порушувати всі рекомендації.

7.10 Імена в мові С

Імена, що вказують на типи даних, мають бути написані в змішаному регістрі. Перша літера імені має бути велика (верхній регістр).

Приклад.

Signal, TimeCount

Змінні мають бути записані іменами в змішаному регістрі, перша буква – мала (нижній регістр).

Приклад.

signal, timeCount

Константи мають бути записані у верхньому регістрі. Як роздільник використовується нижнє підкреслювання.

Приклад.

MAX_TEMP, RED

Методи і функції мають бути названі дієсловами, записаними в змішаному регістрі, перша буква – в нижньому регістрі.

Приклад.
getTime, setTime

7.11 Написання керівної програми для вмиканням/вимикання світлодіода за допомогою аналогової кнопки

Ця програма має керувати світлодіодом за допомогою кнопки:

• при натиснутій кнопці світлодіод світиться;

• при ненатиснутій кнопці світлодіод не світиться.

7.11.1 Підключення кнопки і світлодіода до плати Ардуіно

Для зв'язку з зовнішніми елементами в контролері Arduino UNO існують 14 цифрових виводів. Кожен вивід може бути визначений програмою як вхід або вихід.

У цифрового виходу є тільки два стани – високий і низький. Високий стан відповідає напрузі на виході близько 5 В, низький стан – 0 В. Вихід допускає підключення навантаження зі струмом до 40 мА.

Коли вивід визначено як вхід, прочитавши його стан, можна визначити рівень напруги на вході. При напрузі близькій до 5 В (реально більше 3 В) буде вважатися високий стан, – відповідає константі HIGH. При напрузі близькій до 0 (менше 1,5 В) буде вважатися низький стан або константа LOW.

Світлодіод ми маємо підключити до виводу, визначивши його як вихід, а кнопка підключається до виводу з режимом вхід.

Світлодіод підключається через резистор, що обмежує струм (рис. 7.1).

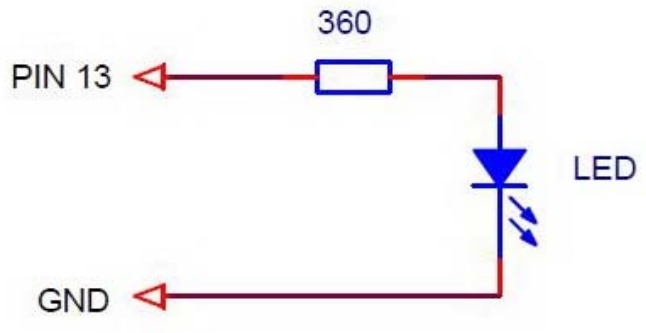


Рисунок 7.1 – Принципова схема під'єднання світлодіода

Резистор розраховується за формулою

$$I = U_{\text{виходу}} - U_{\text{спаду на світлодіоді}} / R,$$

де $U_{виходу}$ = 5 В;

$U_{спаду\ на\ світлодіоді}$ можна взяти 1,5 В (точніше вказується в довіднику).

Виходить, що в нашій схемі струм через світлодіод заданий на рівні 10 мА.

Можна вибрати будь-який вивід, для простоти з'єднань пропонується використовувати світлодіод, встановлений на платі. Той самий, який блимав в першому тестовому прикладі (див. лаб. роботу № 6). Він підключений до цифрового виводу 13. У цьому випадку додатковий світлодіод до плати підключати не треба.

Кнопку підключаємо до будь-якого іншого виводу, наприклад, 12. Апаратна частина схеми підключення кнопки має забезпечувати рівні напруг 0 В – при натиснутій кнопці та 5 В – при ненатиснутій кнопці.

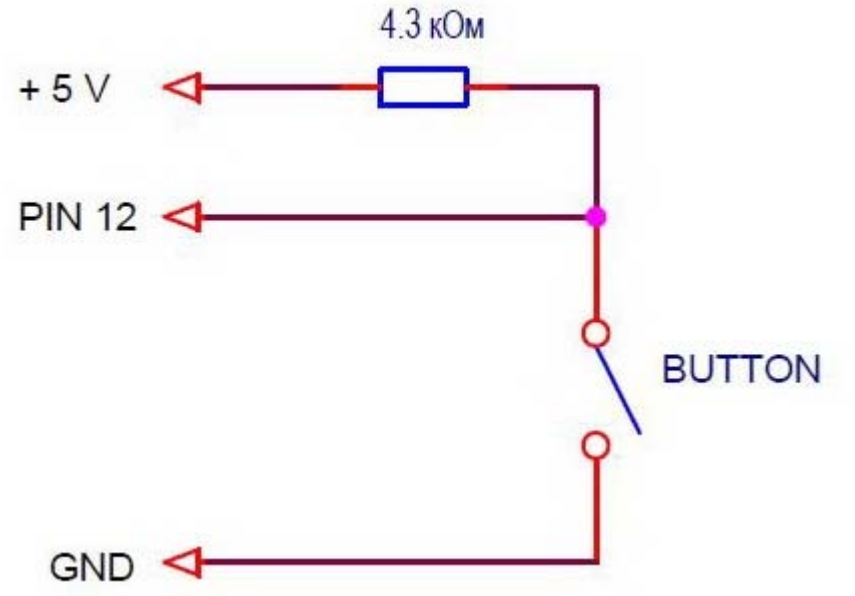


Рисунок 7.2 – Принципова схема керування світлодіодом

При ненатиснутій кнопці резистор формує на виводі 5 В, а при натиснутій – вхід замикається на землю.

Всі виводи плати мають всередині контролера резистори, підключені до 5 В. Їх можна програмно включати або відключати від виводів. Опір цих резисторів приблизно 20–50 кОм. Занадто багато для реальних схем, але для нашої програми і кнопки, встановленої поблизу контролера, цілком допустимо. Враховуючи це, схема підключення буде виглядати так, як показано на рис. 7.3

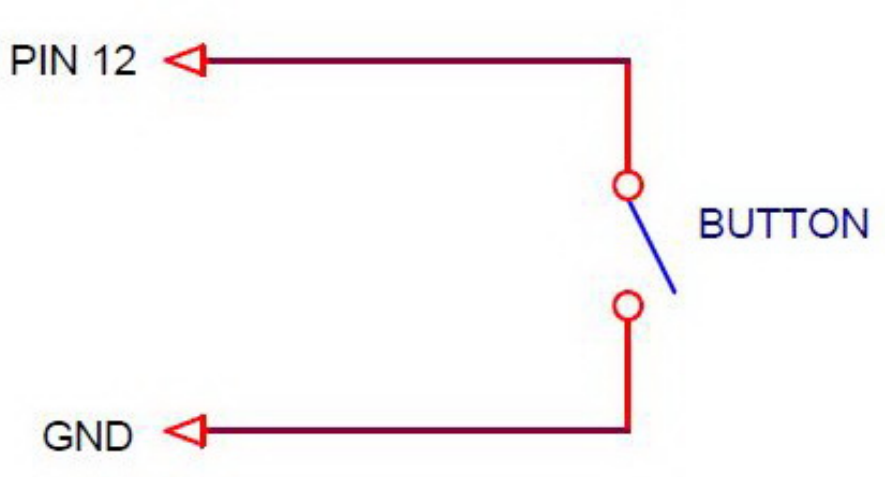


Рисунок 7.3 – Принципова схема керування світлодіодом

Кнопку можна припаяти на проводах роз'єму або встановити її на макетну плату без пайки (див. рис. 7.4).

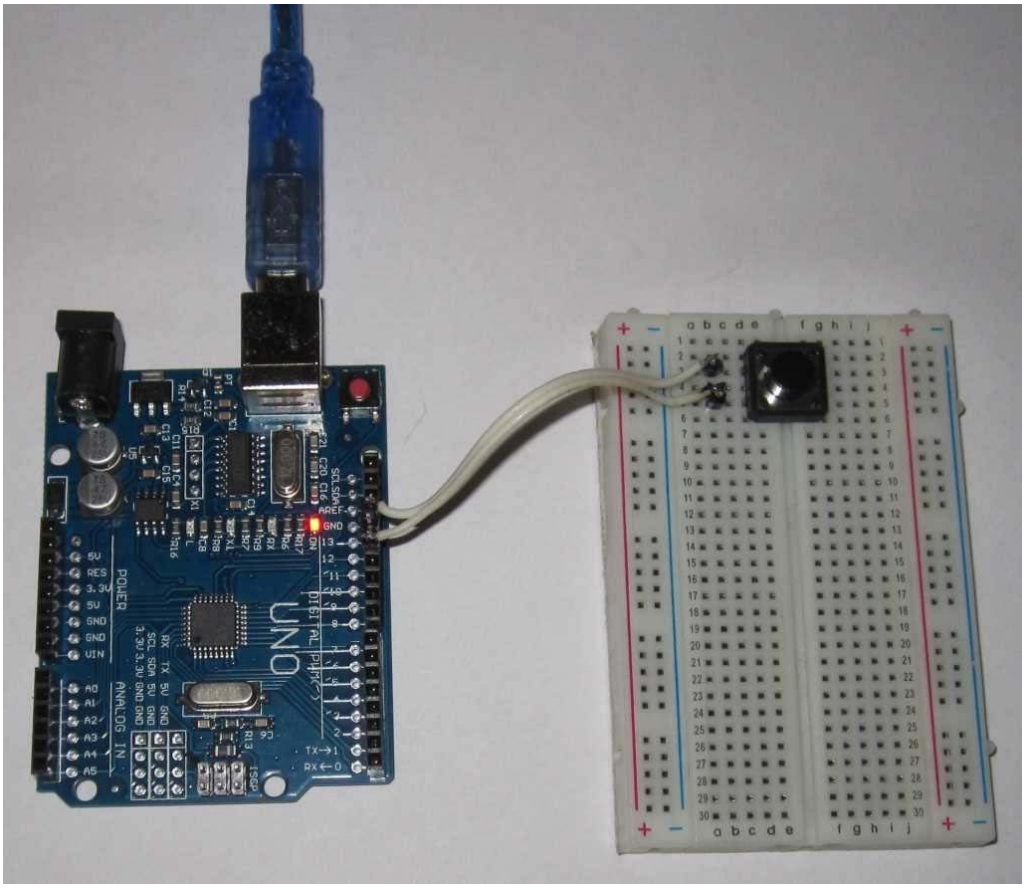


Рисунок 7.4 – Зовнішній вигляд схеми керування з використанням макетної плати

7.11.2 Функції управління виводом

Для роботи з цифровими виводами в системі Ардуіно є 3 вбудовані функції. Вони дозволяють встановити режим виводу, зчитати стан чи встановити вивід в певний стан. Для визначення стану виводів в цих функціях використовуються константи HIGH і LOW, які відповідають високому і низькому рівню сигналу.

pinMode (pin, mode)

Встановлює режим виводу (вхід або вихід).

Аргументи: pin і mode.

• pin – номер виводу;

• mode – режим виводу.

Примітки

mode = INPUT	вивід визначено як вхід, підтягувальний резистор вимкнено
mode = INPUT_PULLUP	вивід визначено як вхід, підтягувальний резистор увімкнено
mode = OUTPUT	вивід визначено як вихід

Функція не повертає нічого.

digitalWrite (pin, value)

Встановлює стан виходу (високий або низький).

Аргументи pin і value:

• pin – номер виводу;

• value – стан виходу.

Примітки

value = LOW	встановлює вихід в низький стан
value = HIGH	встановлює вихід в високий стан

Функція не повертає нічого.

digitalRead (pin)

Зчитує стан входу.

Аргументи: pin – номер виводу.

Повертає стан входу:

digitalRead(pin) = LOW	низький рівень на вході
digitalRead(pin) = HIGH	високий рівень на вході

7.11.3 Код програми управління світлодіодом

З урахуванням попереднього матеріалу у нас є вся необхідна інформація для написання коду програми. Програма в Ардуіно складається з двох функцій setup () і loop(). У setup () ми встановлюємо режими висновків, а в loop () зчитуємо стан кнопки в змінну buttonState і передаємо його на світлодіод. По шляху інвертуємо, тому що утримуючи кнопку маємо низький стан сигналу, а світлодіод світиться при високому.

```
//Програма scetch_1_1
// Вмикає світлодіод (вивід 13) при натисканні кнопки (вивід 12) *

boolean buttonState; // створюємо глобальну змінну buttonState

 void setup () {
  pinMode (13, OUTPUT); // призначаємо вивід 13 (світлодіод) як вихід
  pinMode (12, INPUT_PULLUP); // призначаємо вивід 12 (кнопка) як
// вхід
}
// нескінченний цикл
void loop () {
  buttonState = digitalRead (12); // зчитуємо стан 12 входу (кнопки) і
//записуємо в buttonState
  buttonState =! buttonState; // інверсія змінної buttonState
  digitalWrite (13, buttonState); // записуємо стан з buttonState на вихід
// 13 (світлодіод)
}
```

Для зберігання проміжного значення стану кнопки створюємо змінну buttonState з типом boolean. Це логічний тип даних. Змінна може набувати одного з двох значень: true (істинно) або false (хибно). У нашому випадку – світлодіод світиться і не світиться.

Напишіть код програми у вікно Arduino IDE. Завантажте в контролер і перевірте.

Для збереження проектів Ардуіно бажано створити окрему директорію, наприклад: d:\Arduino Projects\Lessons\Lesson1.

Блок програми:

```
buttonState = digitalRead (12); // зчитуємо стан 12 входу (кнопки) і
// записуємо в buttonState
  buttonState =! buttonState; // інверсія змінної buttonState
  digitalWrite (13, buttonState); // записуємо стан з buttonState на вихід
// 13 (світлодіод)
```

можна записати без використання проміжної змінної buttonState.

```
digitalWrite (13,! digitalRead (12));
```

Як аргумент для функції digitalWrite () виступає функція digitalRead (). Гарний стиль – це саме такий варіант. Не потрібні додаткові змінні, менше текст.

Тобто функцію можна використовувати як аргумент іншої функції. Функції можна викликати з функцій.

Інший варіант цієї ж програми, що використовує умовний оператор if.

```
//Програма scetch_1_2 уроку 5
 //Вмикає світлодіод (вивід 13) при натисканні кнопки (вивід 12)

void setup () {
  pinMode (13, OUTPUT);
// призначаємо вивід 13 (світлодіод) як вихід
  pinMode (12, INPUT_PULLUP);
// призначаємо вивід 12 (кнопка) як вхід
}
// нескінченний цикл
void loop () {
  if (digitalRead (12) == LOW) digitalWrite (13, HIGH);
  else digitalWrite (13, LOW);
}
```

У нескінченному циклі перевіряється стан виведення 12 (кнопка), і якщо він низький (LOW), то на виводі 13 (світлодіод) формується високий стан (HIGH). В іншому випадку стан світлодіода низький (LOW).

7.11.4 Директива #define

У всіх прикладах для функцій вводу/виводу ми вказували аргумент pin, що визначає номер виводу, у вигляді конкретного числа – константи. Ми пам'ятали, що константа 12 – це номер виводу кнопки, а 13 – номер виводу світлодіода. Набагато зручніше працювати з символьними іменами. Для цього в мові С існує директива, що пов'язує ідентифікатори з константами, виразами.

Директива #define визначає ідентифікатор і послідовність символів, яка підставляється замість ідентифікатора, кожен раз, коли він зустрічається в тексті програми.

У загальному вигляді вона виглядає так:

```
#define ім'я послідовність_символів
```

Якщо в наших програмах ми напишемо:

```
#define LED_PIN 13 // номер виводу світлодіода дорівнює 13
```

то кожен раз, коли в програмі зустрінеться ім'я LED_PIN, при трансляції замість нього буде поставлено символи 13. Функція ввімкнення світлодіода виглядає так:

```
digitalWrite (LED_PIN, HIGH);
```

Остаточний варіант програми з використанням #define.

```
//Програма_1_3
//Вмикає світлодіод (вивід 13) при натисканні кнопки (вивід 12)

#define LED_PIN 13 // номер виводу світлодіода дорівнює 13
#define BUTTON_PIN 12 // номер виводу кнопки дорівнює 12
void setup () {
  pinMode (LED_PIN, OUTPUT); // призначаємо вивід 13 (світлодіод) як вихід
  pinMode (BUTTON_PIN, INPUT_PULLUP); // призначаємо вивід 12 (кнопка) як вхід
}
// нескінченний цикл
void loop () {
  digitalWrite (LED_PIN,! digitalRead (BUTTON_PIN));
}
```

Зверніть увагу, що після директиви #define крапка з комою не ставиться, тому що це псевдооператор. Він не робить ніяких дій. Директива задає константи, тому прийнято імена для неї писати у верхньому регістрі з роздільником у вигляді нижнього підкреслювання.

7.12 Завдання до практичної роботи

1. Ознайомтесь уважно із теоретичною частиною.
2. Складіть електричну схему згідно з рис. 7.2. Напишіть код одного із варіантів програми для управління світлодіодом (за вказівкою викладача) у вікно Arduino IDE. Завантажте в контролер і перевірте.
3. Для закріплення вивченого матеріалу напишіть код програми відповідно до індивідуального завдання, яке необхідно отримати у викладача (приклади виконання індивідуальних завдань наведені у додатку А).

Контрольні запитання

1. Як в загальному випадку виглядає структура програми мовою Ардуіно?
2. Назвіть основні правила синтаксису мови програмування Ардуіно.
3. Назвіть типові змінні і типи даних, що використовуються в мові програмування Ардуіно.
4. Як здійснюється оголошення змінних в коді програми?

5. Назвіть основні арифметичні операції, що використовуються при написанні керівної програми мовою Arduino.

6. Назвіть основні операції відношень, що використовуються при написанні керівної програми мовою Arduino.

7. Назвіть основні логічні операції, що використовуються при написанні керівної програми мовою Arduino.

8. Яке призначення оператора IF?

9. Яке призначення оператора ELSE IF?

10. Яке призначення оператора SWITCH CASE?

11. Яке призначення оператора циклу FOR?

12. Яке призначення оператора BREAK?

13. Яке призначення оператора GOTO?

14. Яке призначення оператора CONTINUE?

15. Назвіть основні особливості використання масивів у мові програмування Arduino.

16. Назвіть основні особливості використання функцій у мові програмування Arduino.

17. Які функції виконує директива #define?

ЛАБОРАТОРНА РОБОТА № 8

Дослідження та синтез комбінаційних схем управління

Мета роботи: набуття навичок з синтезу комбінаційних функцій, основних способів їх задання; опанування методики мінімізації функцій; технічна реалізація комбінаційної схеми на заданій елементній базі.

8.1 Теоретичні відомості

Цифрові електронні і мікроелектронні пристрої

Комбінаційними логічними пристроями (КЛП) називаються такі пристрої, сигнали на виходах яких в будь-який момент часу однозначно визначаються додаванням сигналів на вході і не залежать від попередніх станів. Прикладами комбінаційних схем можуть слугувати логічні елементи, електронні ключі а також більш складні пристрої, що виконують довільні логічні функції, функції шифраторів, дешифраторів, мультиплексорів, демультиплексорів, арифметичних пристроїв і т. п.

Довільна комбінаційна логічна функція (КЛФ) може бути достатньо просто описана і синтезована за допомогою відомих методів, серед яких найчастіше використовуються карти Карно.

Синтез комбінаційних схем з одним виходом можна розбити на три етапи. На першому етапі, виходячи із таблиць відповідності (істинності), описують роботу синтезованого КЛП, знаходять мінімальну диз'юнктивну (МДНФ) або мінімальну кон'юнктивну (МКНФ) форму функції.

На другому етапі отриману МДНФ або МКНФ функції записують в операторній формі, де під оператором розуміють функцію, що реалізується конкретним логічним елементом. За операторною формою достатньо просто скласти схему КЛП.

Розглянемо основні операторні форми на прикладі $F = x_1 x_3 + x_2 \overline{x}_3 + \overline{x}_1 \overline{x}_2 \overline{x}_3$. Ці форми відрізняються способом вказання зовнішніх і внутрішніх функцій розкладання. Наприклад, в ДНФ внутрішньою функцією (операцією, що виконується першочергово) є функція І, а зовнішньою – АБО, тобто, ДНФ є формою І – АБО.

Різні операторні форми легко отримати із МДНФ і МКНФ шляхом елементарних логічних перетворень. Так, взявши подвійне заперечення від МДНФ функції і використовуючи правило де Моргана, отримаємо для нашого прикладу такі операційні форми:

$$F = x_1 x_3 + x_2 \overline{x}_3 + \overline{x}_1 \overline{x}_2 x_3 = \text{— форма І /АБО}$$

$$= \overline{\overline{x_1 x_3} \; \overline{x_2 \overline{x}_3} \; \overline{\overline{x}_1 \overline{x}_2 x_3}} =$$

$$= (\overline{x_1 x_3})(\overline{x_2 \bar{x}_3})(\overline{\bar{x}_1 \bar{x}_2 \bar{x}_3}) = \text{— форма І – НЕ / І – НЕ}$$

$$= (\bar{x}_1 + \bar{x}_3)(\bar{x}_2 + x_3)(x_1 + x_2 + \bar{x}_3) = \text{ — форма АБО / І –НЕ}$$

$$= (\bar{x}_1 + \bar{x}_3) + (\bar{x}_1 + x_3) + (x_1 + x_2 + \bar{x}_3) \text{ — форма АБО – НЕ/АБО}$$

Для отримання інших оперативних форм функцію записують в МКНФ, тобто її інверсне значення:

$$\bar{F} = \bar{x}_2 \bar{x}_3 + \bar{x}_1 x_2 x_3 .$$

Виконавши попередні перетворення, отримаємо:

$$F = \bar{x}_2 \bar{x}_3 + \bar{x}_1 x_2 x_3 = \text{ — форма І / АБО — НЕ}$$

$$(\bar{x}_2 \bar{x}_3)(x_2 \bar{x}_3) = \text{ — форма І – НЕ / І}$$

$$= (x_2 + x_3)(x_1 + x_2 + x_3) = \text{ — форма АБО / І}$$

$$= (x_2 + x_3)(x_1 + \bar{x}_2 + \bar{x}_3) =$$

$$= (x_1 + x_2)(x_1 + \bar{x}_2 + \bar{x}_3) \text{ — форма АБО – НЕ / АБО – НЕ}$$

На заключному третьому етапі за операторними поданнями функції складається комбінаційна схема.

Методи синтезу КЛП з декількома виходами основані на використанні однієї функції або її частини для отримання іншої функції. При цьому дублювання логічних елементів практично відсутнє. Найпростіше синтез таких КЛП здійснюється за допомогою діаграми Вейча або карти Карно, які для кожної функції будуються окремо, а потім на них відмічаються однойменні набори, на яких всі або декілька функцій набувають однакових значень.

Розглянемо методику і приклад синтезу довільної комбінаційної логічної схеми з врахуванням реального базису логічних елементів:

а) логічна функція, відповідно до якої буде працювати розроблювана схема, що задана в словесному чи іншому вигляді, записується в формі, зручній для подальшого синтезу, найкраще у вигляді таблиці відповідностей (істинності).

Для наочної ілюстрації скористуємося конкретним прикладом, заданим в таблиці 8.1. Ця таблиця відповідності чотиримісної функції, де на кожному із номерів 0...15 задано значення функції (знаком ∅ позначені невизначені стани).

Таблиця 8.1 – Відповідність чотиримісної функції

Номер															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	1	0	0	0	∅	∅	1	1	0	0	0	1	1	∅

б) за табл. 8.1. і картою Вейча для чотирьох змінних (рис. 8.1, а) функція наноситься на карту Вейча (рис. 8.1, б)

в) виконують покриття всіх одиничних (нульових) значень функції мінімальним числом правильних прямокутників максимальної площі.

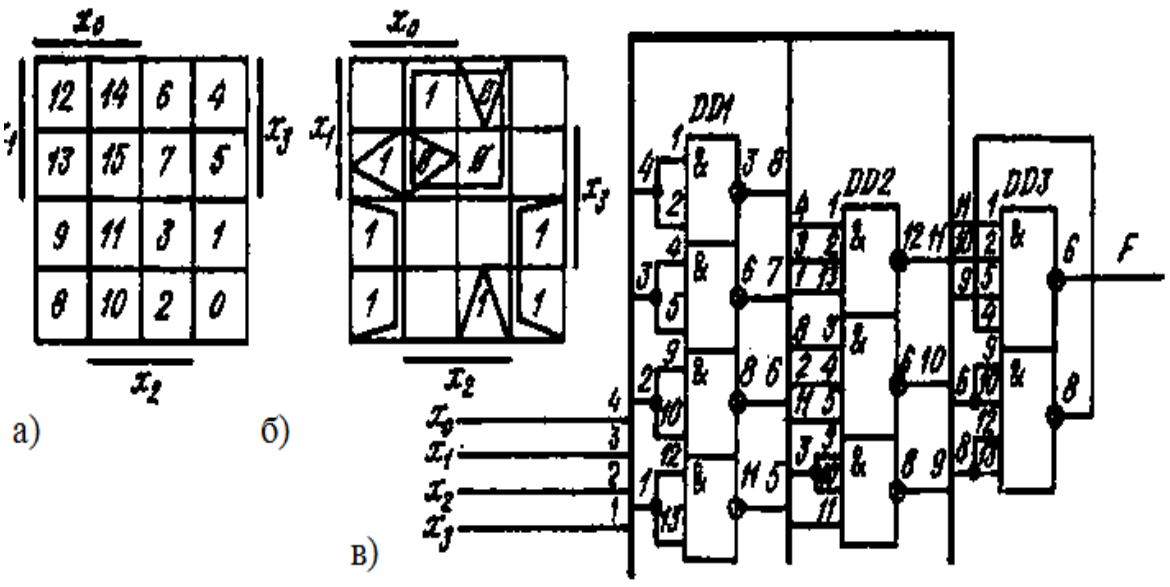


Рисунок 8.1 – Карта Вейча і принципова схема

г) записується результат покриттів у вигляді диз'юнкції кон'юнкцій:

$$F = x_1 x_2 + \bar{x}_1 \bar{x}_2 + x_0 x_1 x_3 + \bar{x}_0 x_2 \bar{x}_3 . \qquad (8.1)$$

Отримане рівняння є основою для побудови електричної схеми, що реалізує задану логічну функцію, однак не враховує характеристики реальних логічних елементів, що є в лабораторії. Аналізуючи отриману функцію, необхідно підібрати реальні логічні елементи для її реалізації. Так, в нашому прикладі необхідно один чотиривхідний, два тривхідних, два двовхідних елементи і чотири інвертори на кожну із змінних.

Як логічні елементи зручно використовувати елементи К155ЛА1, К155ЛА4, К155ЛА3, що реалізують функції І–НЕ, тому запишемо функцію F(7/1) в системі І-НЕ:

$$F = (x_1 \bar{\wedge} x_2) \bar{\wedge} (\bar{x}_1 \bar{\wedge} \overline{x_2}) \bar{\wedge} (x_0 \bar{\wedge} x_1 \bar{\wedge} x_3) \bar{\wedge} (\bar{x}_0 \bar{\wedge} x_2 \bar{\wedge} \overline{x_3}).$$

Для реалізації цієї функції вибирають:

1) один корпус мікросхеми К155ЛА3 (або один корпус мікросхеми К155ЛН1), елемент DD1 (рис. 8.1, в), що дозволяє при об'єднаних входах кожного логічного елемента реалізувати інверсію всіх чотирьох змінних;

2) один корпус мікросхеми К155ЛА4 (елемент DD2), що дозволяє реалізувати дві тривхідні функції І–НЕ і на тій мікросхемі, що залишилася вільною, одну двовхідну функцію І–НЕ (об'єднавши два її входи);

3) один корпус мікросхеми К155ЛА1 (елемент DD3), що дозволяє реалізувати на одній своїй половині чотиривхідну функцію І–НЕ, а на другій – двовхідну функцію І–НЕ, об′єднавши попарно їх входи;

д) згідно з формулою логічної функції (8.1) і вибраними елементами DD1, DD2 і DD3 будується принципіальна схема (рис. 8.1, в), на якій жирною лінією показана загальна шина, номери вхідних сигналів якої позначають числами зліва, а вихідних – справа. Наприклад, якщо сигнал x_1 позначений індексом 3 (рисунок 8.1, в), то із рисунку видно, що він надходить на входи 2, 9 і 10 елемента DD2. Аналогічно позначають і інші сигнали. Застосування такого позначення суттєво спрощує зображення і читання схем.

Спеціальні КЛП призначені для реалізації конкретних логічних функцій: підсумовування, шифрування, дешифрування, перетворення кодів та інші операції. В той же час вони можуть бути реалізовані і на універсальних логічних елементах.

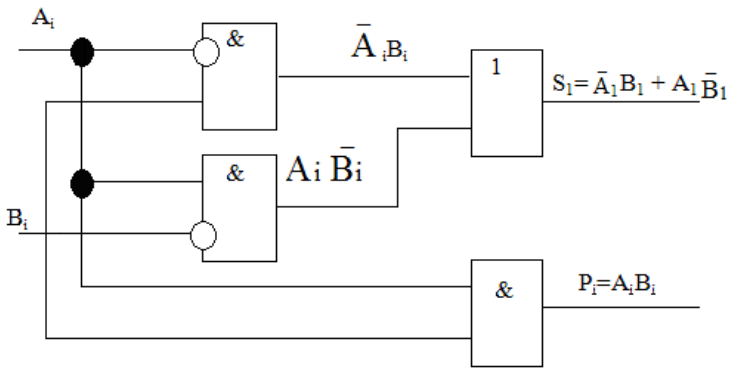
Розглянемо основні види цих схем і особливості їх реалізації.

Суматори

Це пристрої, що здійснюють основну арифметичну операцію – підсумовування чисел в двійковому коді. Найпростіший випадок – підсумовування двох однорозрядних чисел: 0 + 0 = 0, 1 + 0 =1, 1 + 1 = 10. В останньому випадку вихідне число 10 (в десятковому записі це 2) виявилося двійковим дворозрядним. Одиниця, що з′явилася в старшому розряді суми, називається одиницею переносу. На рис. 8.2, а), б) показані схема і таблиця для підсумовування двох однорозрядних чисел. Схема складається із

елементів нерівнозначності (виключальне АБО) і елементів І та має два вихідних проводи: суми S_i і переносу P_i. Така схема називається півсуматором.

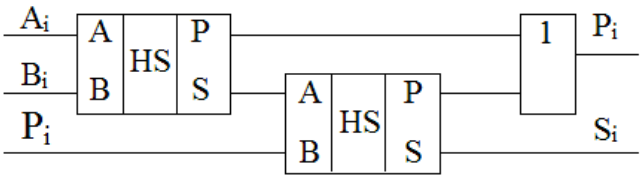
Повний суматор додатково має вхід для прийому сигналу переносу P_{i-1} попереднього розряду. Схема повного суматора двох однорозрядних чисел на двох півсуматорах і його таблиця відповідностей показана на рисунку 8.3, а), б).



Доданок		Результат	
A_i	B_i	Сума S_i	Перенос P_i
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

а) б)

Рисунок 8.2 – Схема півсуматора



а)

б)

Доданок			Результат	
P_i	A_i	B_i	S_i	P_i
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Рисунок 8.3 – Схема повного суматора

Повні суматори багаторозрядних чисел складаються із однорозрядних.

Чотирирозрядний паралельний суматор показаний на рисунку 8.4. Тут порозрядно (по паралелі) підсумовуються два чотирирозрядні слова. Ці пристрої можна зробити довільної довжини, однак підсумовування буде закінчене лише тоді, коли закінчиться час поширення сигналів переносу P_i через все коло однорозрядних суматорів.

В інтегральній мікросхемотехніці суматори виготовляються у вигляді окремих мікросхем на декілька розрядів. Найбільш поширені мікросхеми К155ИМ1...К155ИМ3, К555ИМ6 і К555ИМ7.

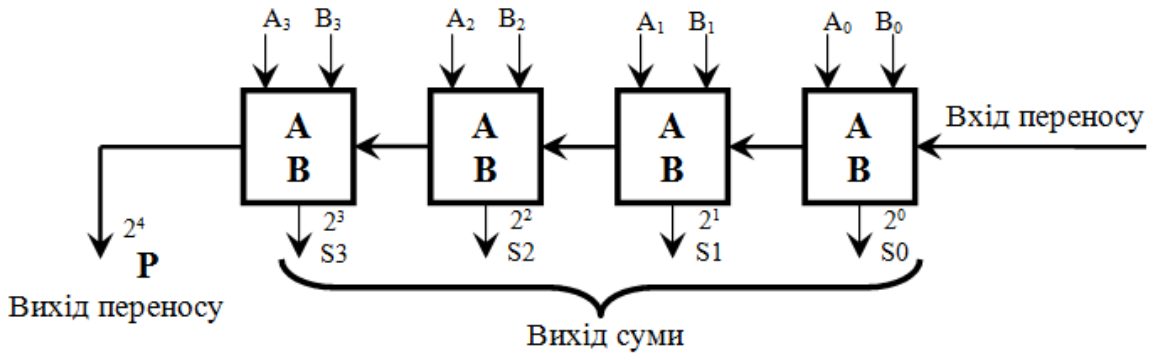


Рисунок 8.4 – Чотирирозрядний паралельний суматор

Дешифратори

Це перетворювачі кодів, що виконують перетворення двійкового і двійково-десяткового кодів в унітарний код. Унітарний код двійкового n-розрядного числа подається 2^n розрядами, один із розрядів якого дорівнює 1.

Дешифратори можуть бути повними і неповними. Повним дешифратором називається комбінаційна схема, що має n входів і 2^n виходів і реалізує на кожному виході функцію, яка являє собою конституенту одиниці (мінтерм). Він описується системою із 2^n логічних рівнянь, права частина кожного із яких записується у вигляді конституенти одиниці. Наприклад, для двовхідного дешифратора:

$$F_0 = \bar{x}_1 \bar{x}_0; F_1 = \bar{x}_1 x_0; F_2 = x_1 \bar{x}_0; F_3 = x_1 x_0.$$

Схема, що реалізує цю функцію, показана на рисунку 8.5, а), а її умовне позначення – на рисунку 8.5, б). На лівому полі показані ваги вхідних сигналів x_0 і x_1, комбінації значень яких розглядаються як двійкові чисел. Кожному вхідному двійковому числу відповідає сигнал, що дорівнює 1 тільки на виході, номер якого, вказаний на правому полі, збігається з двійковим числом.

Неповний дешифратор реалізує $m < 2^n$ конституент одиниці. Такі дешифратори використовуються, наприклад, для перетворення двійково-десяткового числа в код, призначений для керування десятковим індикатором (дешифратор 4×10). Приклад такого дешифратора (мікросхеми 155ИДІ та 564ИДІ) показано на рисунку 8.5, в). Як і для повного дешифратора, можна записати рівняння, що описує роботу неповного дешифратора, і за ним отримати логічну схему.

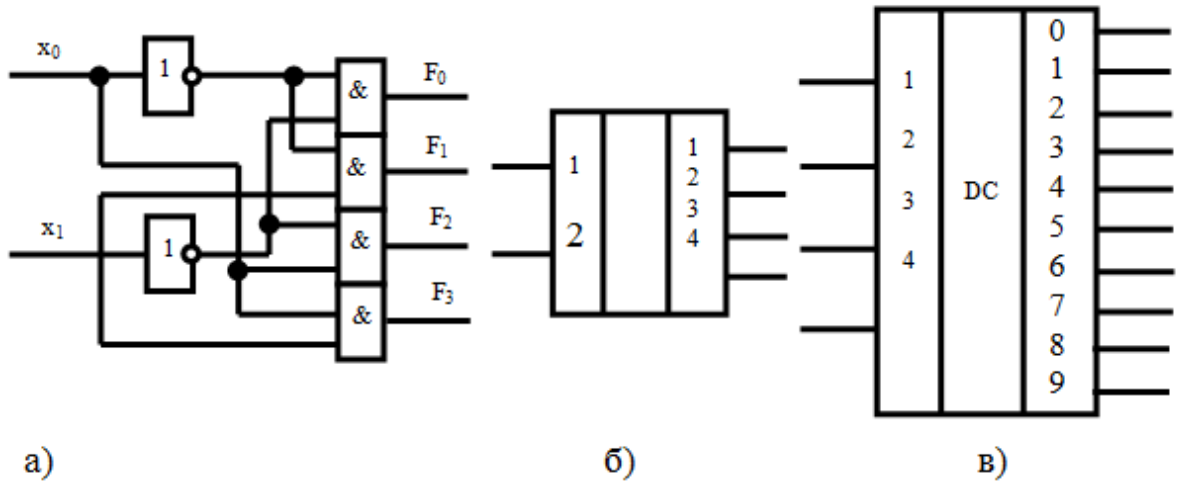


Рисунок 8.5 – Дешифратори

Найбільш поширені мікросхеми дешифраторів К155ИД1, 3, 4, 10; К555, 6, 7, 10; К531ИЛД14.

Шифратори

Виконують функцію, обернену дешифраторам, тобто перетворюють унітарний код в двійковий, двійково-десятковий або будь-який інший. Робота шифратора, як будь-якої двійкової системи, може бути задана у вигляді таблиці відповідностей, за якою досить просто побудувати схему.

Із мікросхем шифраторів відомі, наприклад, К555ИВ3, КМ555ИАІ.

Перетворювачі кодів

Вони використовують спільну роботу дешифратора і шифратора. Дешифратор перетворює двійковий або двійково-десятковий код в унітарний, а шифратор – отриманий унітарний код в потрібний. Типовим прикладом використання такого перетворювача є схема перетворення двійкового коду в код керування семисегментним індикатором.

Мультиплексор

Це схема, що має $n + 2^n$ входів і один вихід, де n – число адресних, а 2^n – число інформаційних входів.

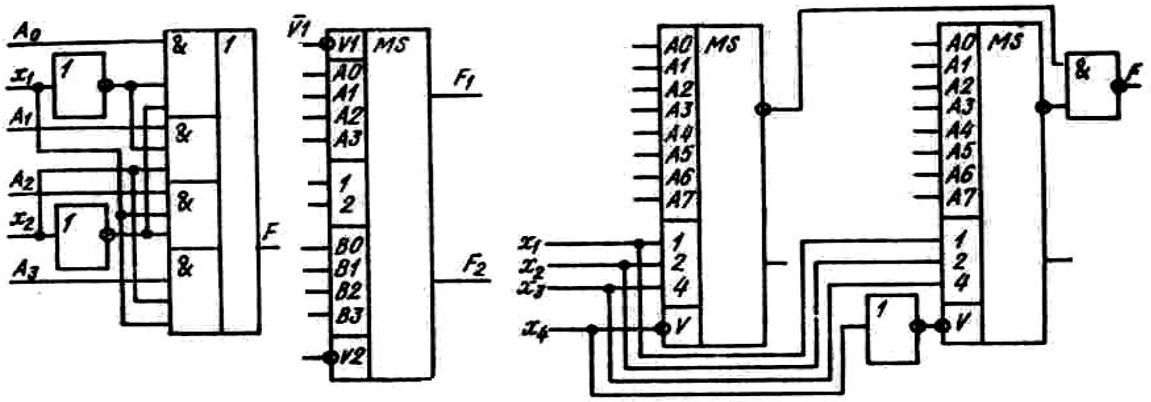
Призначення мультиплексорів (від англ. multiplex – багатократний) – комутувати в бажаному порядку інформацію, що надходить з декількох вхідних шин на одну вихідну. За допомогою мультиплексора здійснюється тимчасове розділення інформації, що надходить по різних каналах. Його можна уподібнити безконтактному багатопозиційному перемикачу.

Мультиплексори мають дві групи входів або один, частіше два, що взаємодоповнюються (інверсні) виходи. Одні входи інформаційні, а інші служать для керування. До них відносяться адресні і розв'язувальні. Адреса подається в двійковому коді, причому кожній адресі відповідає інформаційний вхід, сигнал з якого (0 або 1) при цій адресі приходить на

вихід. Таким чином, в мультиплексорі здійснюється 2^n вхідних сигналів на один вихід.

Розв'язувальний вхід керує одночасно всіма інформаційними входами незалежно від стану адресних входів. Заперечувальний сигнал на цьому вході блокує дію всього пристрою. Наявність розв'язувального входу розширює функціональні можливості мультиплексора, дозволяє синхронізувати його роботу з роботою інших вузлів. Цей вхід використовується також для нарощування розрядності мультиплексорів.

У мультиплексорі, що виготовляється у вигляді окремих мікросхем, число інформаційних входів не перевищує 16. Більша кількість входів забезпечується шляхом нарощування. Нарощування можна виконувати об′єднанням декількох мультиплексорів в пірамідоподібну (деревоподібну) систему або послідовним з′єднанням розв'язувальних входів і зовнішніх логічних елементів. Другий спосіб застосовується найчастіше, оскільки при пірамідоподібному нарощуванні більші витрати мікросхем і порівняно низька швидкодія через підсумовування затримок при послідовному проходженні сигналів за ступеням піраміди.



а) б) в)

Рисунок 8.6 – Принципова схема чотириканального мультиплексора

Принципова схема чотириканального мультиплексора на елементах І–АБО–НЕ, що має два адресних входи x_1 і x_2, показана на рис. 8.6, а); умовне позначення подвійного чотириканального мультиплексора зі стробованим входом (мікросхема 155КП6) та 16-канального, виконаного на 8-канальних мультиплексорах (мікросхеми 155П7), з′єднаних за другим способом, – на рис. 8.6, б), в). В 16-канальному комутаторі стробувальний вхід V використовується як додатковий адресний вхід x_4. Існує і окрема 16-канальна мікросхема 155КП1 мультиплексора зі стробуванням (селектора-мультиплексора). З двох таких мікросхем за вказаним принципом можна виконати 32-канальний мультиплексор.

Для отримання 64-канального мультиплексора потрібно використовувати чотири ВІС 155КП1 і 4-вхідний елемент І–НЕ, а керування

входами $\overline{V}$ необхідно виконувати інверсним чотирирозрядним унітарним кодом.

8.2 Завдання до практичної роботи

1. Від аналітичного задання комбінаційної ф-ції (n=3) перейти до словесного та табличного, здійснити оптимізацію ф-ції двома методами:

а) за допомогою діаграми Вейча-Карно;

б) алгебраїчним методом на підставі законів алгебри Буля.

Розробити комбінаційні схеми для вихідного та мінімізованого варіантів ф-ції на І, АБО, НЕ.

2. Від матричного задання комбінаційної ф-ції (n=4) перейти до її аналітичного запису, виконати мінімізацію ф-ції, розробити комбінаційну схему в двох варіантах:

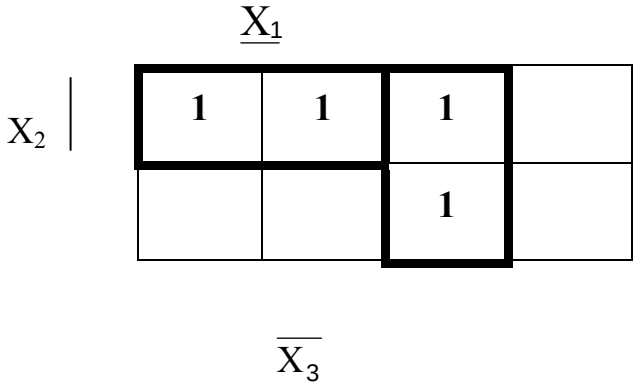
а) на елементах І–НЕ, АБО–НЕ;

б) на елементах І, АБО, НЕ.

Вибрати оптимальний варіант за кількістю елементів, потрібних для реалізації схем.

Розглянемо приклад, в якому задано комбінаційну функцію трьох аргументів

$$F = \overline{X_1} \cdot X_2 \cdot X_3 + X_1 \cdot X_2 \cdot X_3 + X_1 \cdot X_2 \cdot \overline{X}_3 + \overline{X_1} \cdot \overline{X_2} \cdot X_3$$



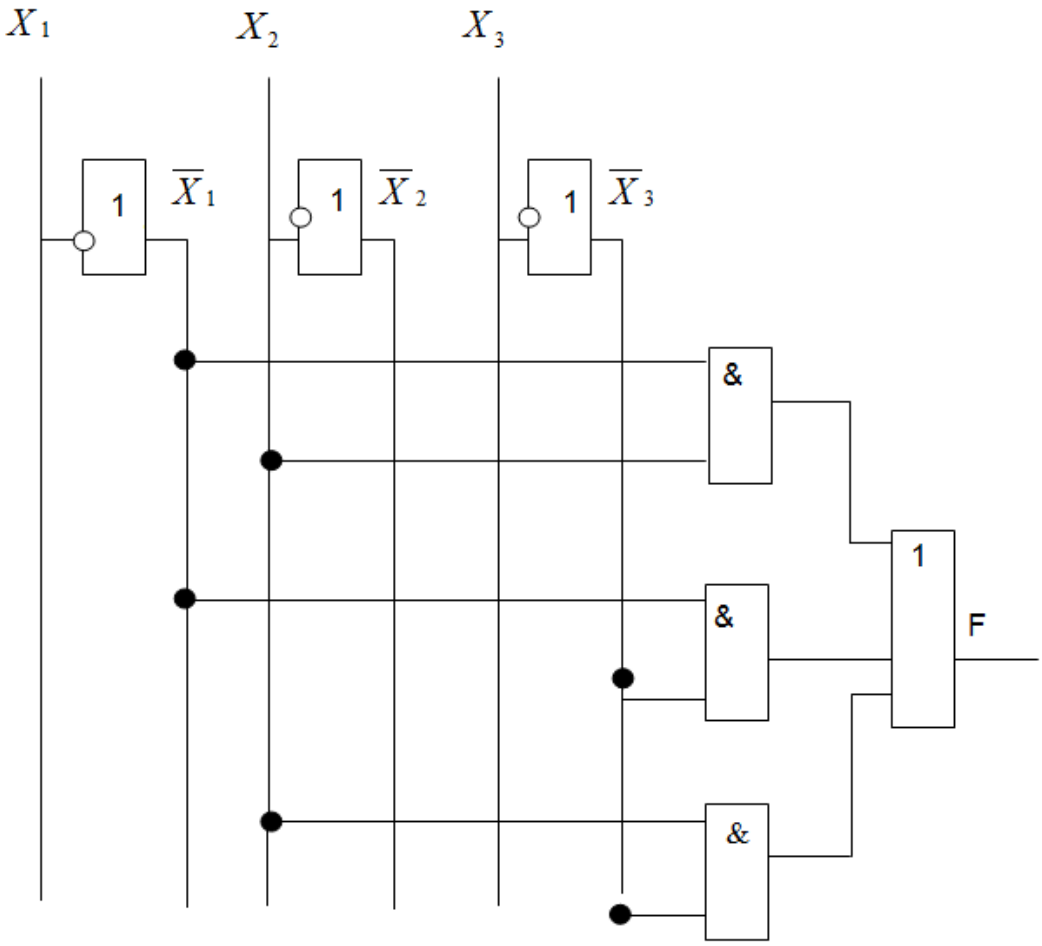
n=3 Мінімізація за допомогою діаграми Вейча-Карно

$F = X_1 \cdot X_2 + X_1 \cdot \overline{X}_3$;

$$\left.\begin{matrix} X_1 \cdot X_2 \cdot \overline{X}_3 \\ X_1 \cdot X_2 \cdot X_3 \end{matrix}\right\} X_1 \cdot X_2;$$

$$\left.\begin{matrix} \overline{X}_1 \cdot X_2 \cdot X_3 \\ \overline{X}_1 \cdot \overline{X}_2 \cdot X_3 \end{matrix}\right\} \overline{X}_1 \cdot X_2.$$

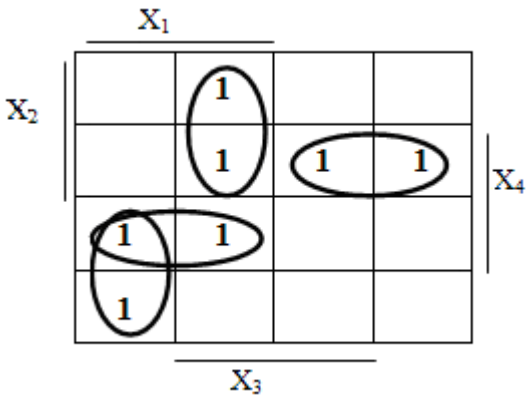
Розробляємо комбінаційну схему для мінімізованої функції:



Розглянемо приклад, в якому задано комбінаційну функцію чотирьох аргументів

$$F = X_1 \cdot X_2 \cdot X_3 \cdot X_4 + \overline{X}_1 \cdot X_2 \cdot \overline{X}_3 \cdot X_4 + X_1 \cdot X_2 \cdot X_3 \cdot \overline{X}_4 + \overline{X}_1 \cdot X_2 \cdot X_3 \cdot X_4 +$$
$$+ X_1 \cdot \overline{X}_2 \cdot X_3 \cdot X_4 + X_1 \cdot \overline{X}_2 \cdot \overline{X}_3 \cdot X_4 + X_1 \cdot \overline{X}_2 \cdot \overline{X}_3 \cdot \overline{X}_4$$

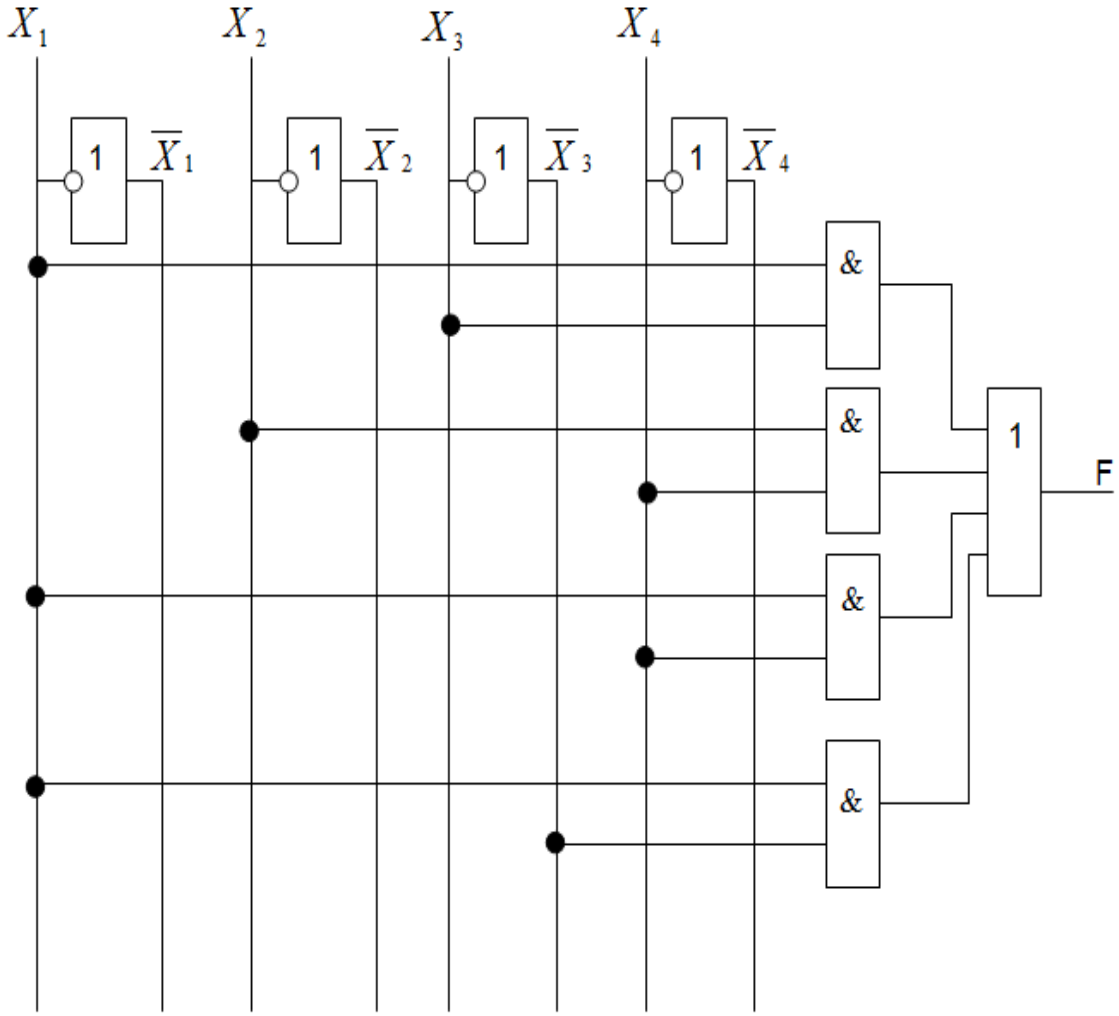
n = 4 мінімізувати за допомогою діаграми Дейча, карти Карно



n = 4 Мінімізація за допомогою діаграми Дейча, карти Карно

$$F = X_1 \cdot X_3 + X_2 \cdot X_4 + X_1 \cdot X_4 + X_1 \cdot X_4 + X_1 \cdot \overline{X}_3 .$$

Розробляємо комбінаційну схему для мінімізованої функції:



8.3 Задача для індивідуального виконання

Завдання на розробку комбінаційної схеми

Аварійне гальмування робокара має вмикатись автоматично, якщо спрацьовує хоча б один з двох датчиків: далекої або ближньої локації при ввімкненому датчику руху робокара. Крім того, аварійне гальмування має вмикатись, якщо спрацьовує датчик зниження тиску в гідросистемі при зупиненому робокарі та відсутності сигналів локації.

Варіанти завдань до задачі наведено у таблиці 8.1

Таблиця 8.1 – Варіанти завдань

Варіант	Датчик дальньої локації	Датчик ближньої локації	Датчик руху робокара	Датчик зниження тиску
1	X1	X2	X3	X4
2	X1	X2	X4	X3
3	X1	X3	X2	X4
4	X1	X3	X4	X2
5	X1	X4	X2	X3
6	X1	X4	X3	X2
7	X2	X1	X3	X4
8	X2	X1	X4	X3
9	X2	X3	X1	X4
10	X2	X3	X4	X1
11	X2	X4	X1	X3
12	X2	X4	X3	X1
13	X3	X1	X2	X4
14	X3	X1	X4	X2
15	X3	X2	X1	X4
16	X3	X2	X4	X1
17	X3	X4	X1	X2
18	X3	X4	X2	X1
19	X4	X1	X2	X3
20	X4	X1	X3	X2
21	X4	X2	X1	X3
22	X4	X2	X3	X1
23	X4	X3	X1	X2
24	X4	X3	X2	X1

Контрольні запитання

1. Що називають комбінаційними логічними пристроями?
2. Коротко опишіть методику синтезу довільної комбінаційної логічної схеми з врахуванням реального базису логічних елементів.
3. Яку функцію виконують суматори?
4. Яку функцію виконують шифратори/дешифратори?
5. Що являють собою перетворювачі кодів?
6. Що таке мультиплексор?

СПИСОК ЛІТЕРАТУРИ

1. Пішенін В. О. Електроніка і мікропроцесорна техніка. Частина 1. Елементна база промислової електроніки : навчальний посібник / Пішенін В. О., Коц І. В., Нікітіна Н. В. – Вінниця : ВДТУ, 2001. – 67 с.

2. Пішенін В. О. Електроніка і мікропроцесорна техніка. Частина 1 : лабораторний практикум / В. О. Пішенін, Н. В. Нікітіна. – Вінниця : ВНТУ. 2004. – 71 с.

3. Квітка С. О. Електроніка та мікросхемотехніка : навчальний посібник / Квітка С. О., Яковлєв В. Ф., Нікітіна О. В.; за ред. проф. В. Ф. Яковлєва. – К. : Аграрна освіта, 2010. – 329 с.

4. Колонтаєвський Ю. П. Електроніка та мікросхемотехніка : підручник / Ю. П. Колонтаєвський, А. Г.Сосков; за редакцією А. Г. Соскова.– [2-е видання]. – К. : Каравела, 2009. – 416 с.

5. Мікропроцесорна техніка : навч. посібник / [В. В. Ткачов, Г. Грулер, Н. Нойбергер та ін.]. – Донецьк : ДВНЗ «Національний гірничий університет», 2012. – 188 с.

6. Рожков П. П. Конспект лекцій з дисципліни «Мікропроцесорна техніка» / П. П. Рожков, С. Е. Рожкова. – Харків : ХНАМГ, 2008. – 116 с.

Додаток А

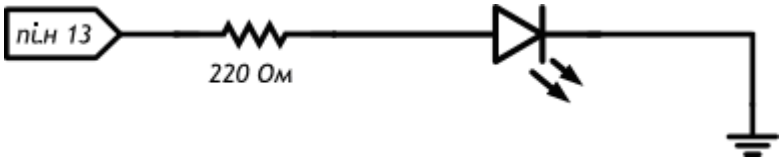
Приклади виконання індивідуальних завдань до лабораторної роботи № 7 «Основи програмування мікроконтролера Arduino»

Приклад № 1

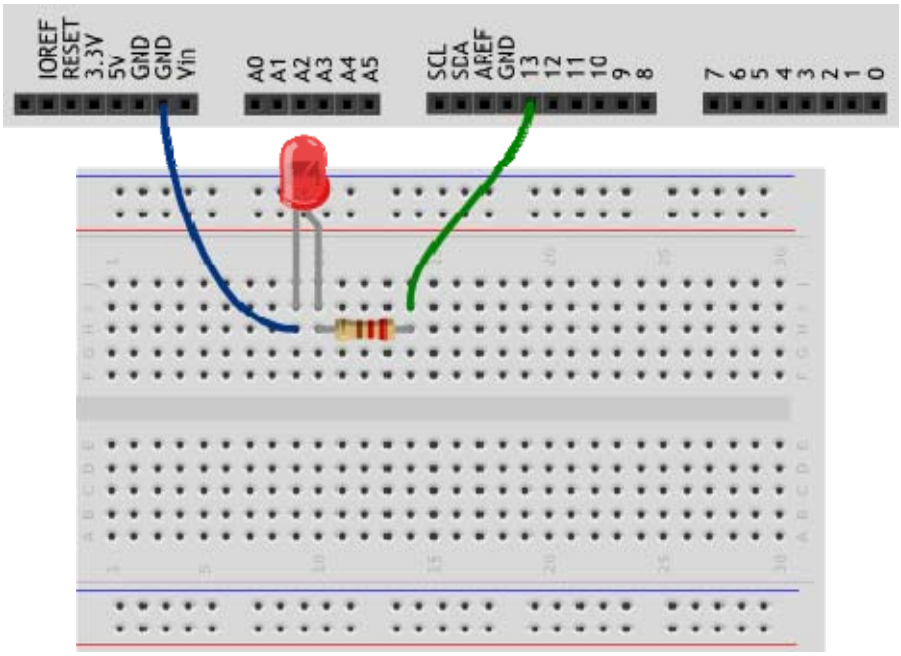
Завдання: написати програму (скетч) для керування вмиканням напівпровідникового світлодіода (час перебування у ввімкненому стані світлодіода 100 мс, час перебування у вимкненому стані 900 мс).

Результат виконання

Принципова схема вмикання світлодіода:



Загальний вигляд схеми на макетній платі:



Зауваження до схеми:

Катод («мінус») світлодіода – коротка ніжка, саме її потрібно з'єднувати з землею (GND).

Обов'язково використайте резистором, інакше світлодіод вийде з ладу.

Плата Arduino має три роз'єми (піни) GND (земля), використовуйте будь-який з них.

Скетч:

```
void setup()
{
  // налаштовуємо пін №13 в режим виходу,
  // тобто в режим джерела напруги
  pinMode(13, OUTPUT);
}

void loop()
{
  // подаємо на пін 13 «високий сигнал» або «логічна одиниця»
 // (англ. «high»), тобто
  // виводимо 5 В і світлодіод починає світитись

  digitalWrite(13, HIGH);

  // затримуємо (англ. «delay») мікроконтролер в цьому
  // стані на 100 мс
  delay(100);

  // подаємо на пін 13 «низький сигнал», або «логічний нуль»
  //   (англ. «low»), тобто
  // виводимо 0 В або, точніше, прирівнюємо пін 13 до «землі».
  // в результаті світлодіод погасне

 digitalWrite(13, LOW);

  // затримуємо мікроконтролер в цьому стані на 900 мс

 delay(900);

  // після витримки 900 мс команда «loop» (цикл) відразу  починає
  // виконуватись
  // в результаті цей процес буде виглядати як
  // мигання світлодіода один раз в 100 мс + 900 мс = 1000 мс = 1 с

}
```

Звернути увагу!

Процедура *setup* виконується один раз при запуску мікроконтролера. Зазвичай вона використовується для конфігурації портів мікроконтролера та інших параметрів

Після виконання *setup* запускається процедура *loop*, яка виконується в нескінченному циклі. Саме цим ми користуємося в цьому прикладі, щоб маячок блимав постійно.

Процедури *setup* і *loop* маютьбути присутніми в будь-якій програмі (скетчі), навіть якщо вам не потрібно нічого виконувати в них (вони можуть бути порожні).

Запам'ятайте, що кожній фігурній дужці «{» завжди відповідає має відповідати «}». Вони позначають межі якогось логічно завершеного фрагмента коду. Слідкуйте за текстом між дужками. Для наочності зручно після кожної відкритої дужки збільшувати відступ у новому рядку на один символ табуляції (клавіша Tab).

Звертайте увагу на знак «;» в кінці рядка.

Функція «digitalWrite (pin, value)» не повертає ніякого значення і приймає два параметри:

pin – номер цифрового порту, на який ми відправляється сигнал;

value – значення, яке відправляється на порт (пін). Для цифрових портів значенням може бути HIGH (високе, логічна одиниця) або LOW (низьке, логічний нуль)

Якщо як наступний параметр ви передасте функції «digitalWrite» значення, відмінне від HIGH, LOW (1 або 0), компілятор може не видати помилку, але вважати, що передано HIGH. Будьте уважні.

Зверніть увагу, що використані константи: INPUT, OUTPUT, LOW, HIGH, пишуться великими літерами, інакше компілятор їх не розпізнає і видасть помилку. *Коли ключове слово розпізнано, воно підсвічується синім кольором в Arduino IDE.*

Завдання для самостійного виконання

1. Зробіть так, щоб маячок світився 0,5 секунди, а пауза між спалахами дорівнювала одній секунді.

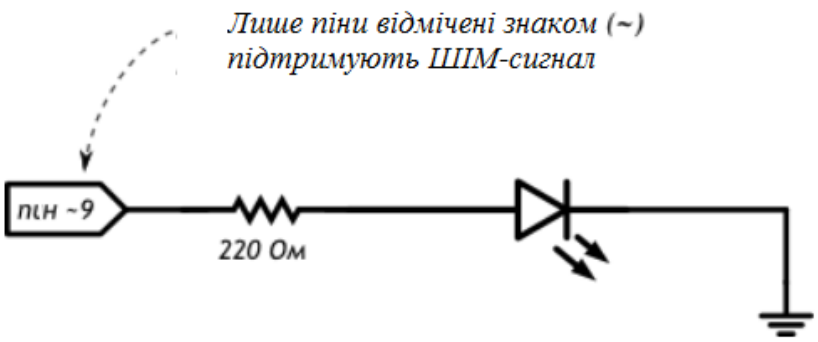
2. Змініть код прикладу так, щоб маячок включався на три секунди після запуску пристрою, а потім блимав в стандартному режимі (як у прикладі).

Приклад № 2

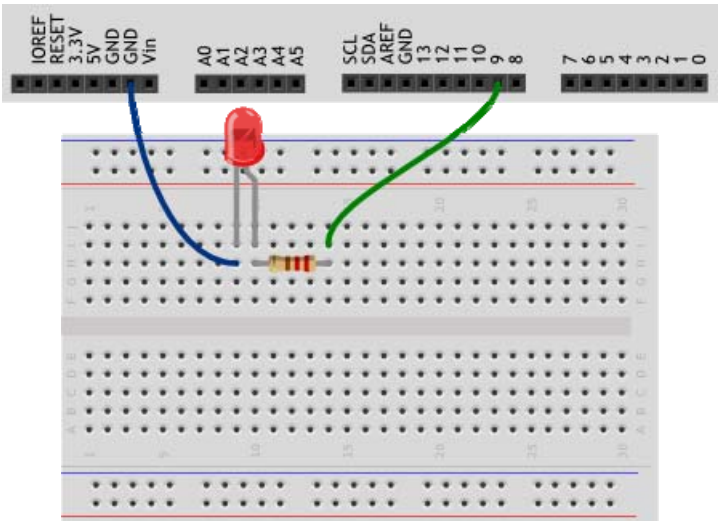
Завдання: написати програму (скетч) для зміни яскравості світіння напівпровідникового світлодіода (яскравість має змінюватись ступінчасто, спочатку на ⅓ від максимальної яскравості, далі ⅔ від максимальної яскравості та повна яскравість; тривалість кожного періоду 250 мс).

Результати виконання

Принципова схема вмикання світлодіода:



Загальний вигляд схеми на макетній платі:



Зауваження до схеми:

Не всі порти Arduino підтримують широтно-імпульсну модуляцію (ШІМ), якщо є необхідність у регулюванні напруги, то необхідно використовувати порти, які помічені символом тильда «~». Для Arduino Uno це порти № 3, 5, 6, 9, 10, 11.

Скетч:

```
// даємо зрозуміле ім'я для порту №9 зі світлодіодом
// (англ. Light Emitting Diode або просто «LED»)

#define LED_PIN 9

void setup ()
{
  // налаштовуємо пін зі світлодіодом в режим виходу
 // (джерела напруги),
    pinMode (LED_PIN, OUTPUT);
}
```

```
void loop ()
{
  // видаємо неповне значення напруги на світлодіод
  // (він же ШІМ-сигнал, або PWM-сигнал).
  // Мікроконтролер переводить число від 0 до 255 до напруги
  // від 0 до 5 В. Наприклад, 85 –  це 1/3 від 255,
  // тобто 1/3 від 5 В, тобто 1,66 В.

  analogWrite (LED_PIN, 85);
  // тримаємо таку яскравість 250 мілісекунд

delay (250);

  // видаємо 170, тобто 2/3 від 255, або іншими словами – 3,33 В.
  // Вища напруга – вища яскравість!

  analogWrite (LED_PIN, 170);
  delay (250);

  // всі 5 В – максимальна яскравість

  analogWrite (LED_PIN, 255);

 // встановлюємо затримку 250 мс перед тим, як цикл почнеться заново

 delay (250);
}
```

Звернути увагу!

Ідентифікатори змінних, констант, функцій (в цьому прикладі ідентифікатор «LED_PIN») є одним словом (тобто не можна створювати ідентифікатор «LED PIN» з окремих слів).

Ідентифікатори можуть містити лише латинські літери, цифри та символи підкреслення _. При цьому ідентифікатор не може починатися з цифри.

Регістр букв в ідентифікаторі має значення. Тобто слова «LED_PIN», «LED_pin» і «led_pin» з точки зору компілятора – різні ідентифікатори.

Ідентифікатори, які створюються користувачем, не можуть збігатися з стандартними конструкціями мови; якщо середовище розробки програми керування автоматично змінило на будь-який інший колір введеного ідентифіктора, то ідентифікатор необхідно замінити на інший, що не повторює стандартні конструкції мови програмування.

Директива *#define* вказує компілятору замінити всі вхідні сигнали заданого ідентифікатора на значення, задане після пропуску (тут 9), ці директиви поміщають в початок коду. В кінці такої директиви крапка з комою «;» не припустима.

Назви ідентифікаторів завжди потрібно робити осмисленими, щоб при поверненні до раніше написаної програми було зрозуміло, навіщо потрібен кожен з них.

Також корисно доповнювати код програми коментарями: в прикладах ми бачимо однорядкові коментарі, які починаються з двох прямих «слешів» // і багаторядкові, поміщені між /* */. Коментарі ігноруються компілятором, але є корисними при читанні давно написаного або написаного іншою людиною коду.

Функція *analogWrite (pin, value)* не повертає ніякого значення і приймає два параметри:

pin – номер порту, на який ми відправляємо сигнал;

value – значення скважності ШІМ, яке ми відправляємо на порт. Може бути цілочисловим значенням від 0 до 255, де 0 – це 0%, а 255 – це 100%

Завдання для самостійного виконання

1. Вимкніть живлення контролера, від'єднайте світлодіод від 9-го порту і підключіть до 11-го. Змініть програму так, щоб схема знову запрацювала.

2. Змініть код програми так, щоб протягом секунди на світлодіод послідовно подавалося усереднене значення напруги 0, 1, 2, 3, 4, 5 В.

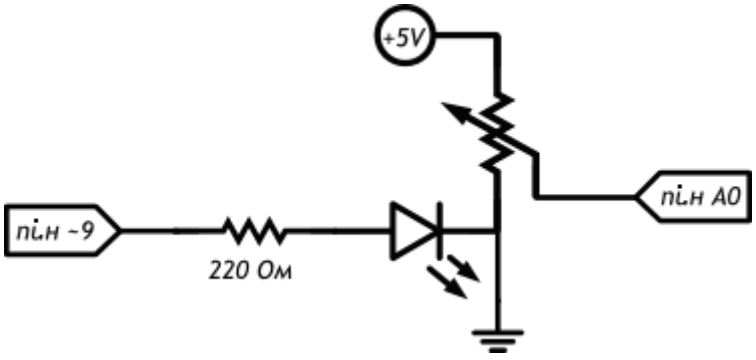
3. Використавши ще один світлодіод, резистор на 220 Ом складіть аналогічну схему на цій самій макетній платі, підключивши світлодіод до піну номер 3 і іншого входу GND. Змініть програму так, щоб світлодіоди блимали в протифазі: перший вимкнений, другий горить максимально яскраво і навпаки.

Приклад № 3

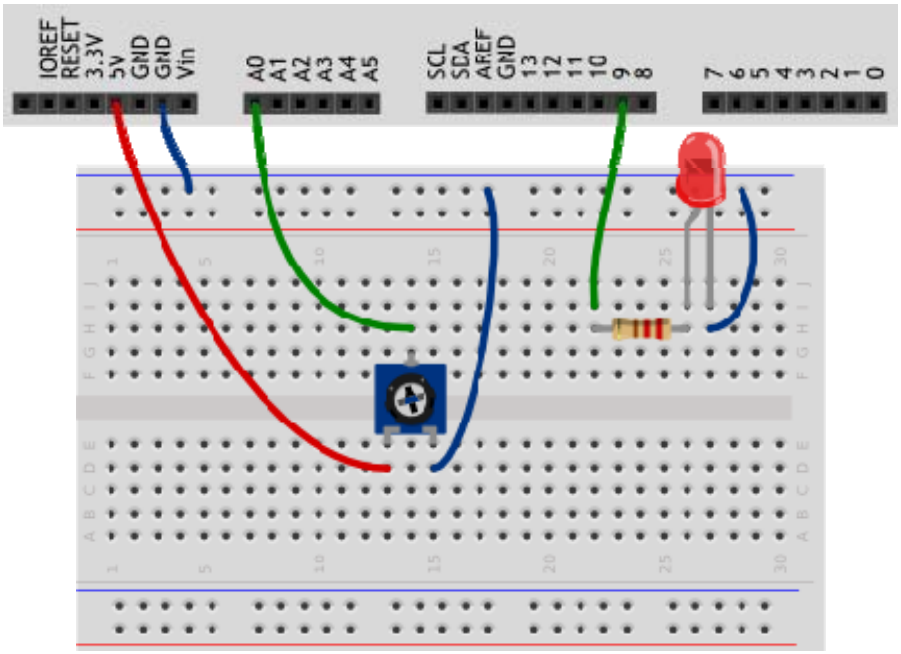
Завдання: написати програму (скетч) для зміни яскравості світіння напівпровідникового світлодіода залежно від значення опору потенціометра.

Результат виконання

Принципова схема вмикання світлодіода:



Загальний вигляд схеми на макетній платі:



Зауваження до схеми:

Не важливо, який із крайніх виводів потенціометра буде підключено до 5 В, а який до GND, зміниться тільки напрямок, в якому потрібно обертати ручку потенціометра для збільшення напруги. Запам'ятайте, що сигнал ми зчитуємо з середнього виводу.

Для зчитування аналогового сигналу, що набуває широкого спектру значень, а не просто 0 або 1, як цифровий підходять тільки порти, помічені на платі як «ANALOG IN» і пронумеровані з префіксом A. Для Arduino Uno – це A0–A5.

Скетч:

```
// даємо зрозумілі імена для пінів з світлодіодом
// і потенціометром (англ potentiometer або просто «pot»)

#define LED_PIN 9
#define POT_PIN A0

void setup ()
{
  // пін з світлодіодом – вихід, як і раніше

pinMode (LED_PIN, OUTPUT);

  // пін з потенціометром має бути входом
  // (англ. «Input»): ми хочемо зчитувати значення напруги,
  // яке видається цим портом
```

```
  pinMode (POT_PIN, INPUT);
}

void loop ()
{
  // призначаємо, що далі ми будемо використовувати 2 змінні з
  // іменами rotation і brightness, і що зберігати в них будемо
  // цілі числа (англ. «Integer», скорочено просто «int»)

  int rotation, brightness;

  // зчитуємо в rotation значення напруги з потенціометра:
  // мікроконтролер видасть число від 0 до 1023,
  // пропорційне куту повороту ручки

  rotation = analogRead (POT_PIN);

  // в brightness записуємо отримане раніше значення rotation,
  // поділене на 4. Оскільки в змінних ми побажали зберігати
  // цілі значення, дробова частина від ділення буде відкинута.
  // В результаті ми отримаємо ціле число від 0 до 255

 brightness = rotation / 4;

  // видаємо результат на світлодіод

  analogWrite (LED_PIN, brightness);
}
```

Звернути увагу!

За допомогою директиви *#define* ми вказали компілятору замінювати ідентифікатор POT_PIN на A0 – номер аналогового входу. Ви можете зустріти код, де звернення до аналогового порту буде за номером без індексу A. Такий код буде працювати, але для уникнення плутанини з цифровими портами використовуйте індекс.

Змінним прийнято давати назви, що починаються з малої літери.

Для оголошення змінної необхідно вказати її тип, тут – «int» (від англ. Integer) – цілочислове значення в діапазоні від -32 768 до 32 767.

Змінні одного типу можна оголошувати в одній інструкції, перерахувавши їх через кому, що ми і зробили.

Функція «analogRead (pinA)» повертає цілочислове значення в діапазоні від 0 до 1023, пропорційно напрузі, яка подана на аналоговий вхід, номер якого ми передаємо функції як параметр «pinA».

Зверніть увагу, як ми отримали значення, повернене функцією «analogRead ()», – ми просто помістили його в змінну «rotation» за допомогою оператора присвоювання «=», який записує те, що знаходиться праворуч від нього, в ту змінну, яка стоїть зліва.

Завдання для самостійного виконання

Вимкніть живлення плати, підключіть до порту 5 ще один світлодіод. Змініть код таким чином, щоб другий світлодіод світився на 1/2 від яскравості першого.

Інструктивно-методичне видання

МЕТОДИЧНІ ВКАЗІВКИ ДО ЛАБОРАТОРНИХ РОБІТ З ДИСЦИПЛІНИ «ЕЛЕКТРОНІКА ТА МІКРОПРОЦЕСОРНА ТЕХНІКА ПРОМИСЛОВОГО ОБЛАДНАННЯ»

для студентів спеціальності
133 – Галузеве машинобудування
першого (бакалаврського) рівня вищої освіти

Укладач: *Манжілевський Олександр Дмитрович*

Рукопис оформив *О. Манжілевський*

Редактор *Т. Старічек*

Оригінал-макет підготував *О. Ткачук*

Підписано до друку 18.10.2019.
Формат 29,7×42¼. Папір офсетний.
Гарнітура Times New Roman.
Друк різографічний. Ум. друк. арк. 4,98.
Наклад 40 (1-й запуск 1–21) пр. Зам. № 2019-134.

Видавець та виготовлювач
Вінницький національний технічний університет,
інформаційний редакційно-видавничий центр.
ВНТУ, ГНК, к. 114.
Хмельницьке шосе, 95,
м. Вінниця, 21021.
Тел. (0432) 65-18-06.
press.vntu.edu.ua;
E-mail: kivc.vntu@gmail.com.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.