Методичні вказівки
до проведення лабораторних робіт
з курсу Програмовані логічні схеми

для студентів спеціалізацій “Прикладна фізика”, “Електроніка” , “Комп’ютерна інженерія”, “Інформаційні системи і технології”

к. ф.-м. наук доц. Сусь Б.Б.

к. ф.-м. наук доц. Загороднюк С.П.

к. ф.-м. наук доц. Баужа О.С.

Київ 2025

Затверджено вченою радою Навчально-наукового інституту високих технхнологій (протокол № 4 від 8 жовтня 2025 року)

Вступ

Методичні вказівки до лабораторних робіт розроблено в межах викладання дисциплін: «Програмовані логічні інтегральні схеми», «Цифрові технології та мультимедіа», «Особливості проєктування електронних систем із застосуванням мікроконтролерів та програмованих логічних інтегральних схем», «Мікропроцесори та мікроконтролери», «Електроніка вбудованих систем» для здобувачів вищої освіти за спеціальностями «Прикладна фізика», «Електроніка», «Комп'ютерна інженерія», «Інформаційні системи і технології».

Лабораторний практикум є невід'ємною складовою освітнього процесу при вивченні природничих і технічних дисциплін. Виконання лабораторних робіт дає можливість студентам не лише засвоїти теоретичний матеріал, але й сформувати практичні навички, необхідні для підготовки кваліфікованого фахівця.

Метою вказівок є сприяння формуванню у студентів практичних умінь проєктування, моделювання та аналізу електронних схем на основі сучасних цифрових технологій. Особливу увагу приділено вивченню принципів побудови та функціонування програмованих логічних схем, зокрема рішень на основі ПЛІС, мікроконтролерів і мікропроцесорів, які є основою сучасних вбудованих та керованих електронних систем.

Зміст лабораторних робіт спрямований на поглиблення теоретичних знань та забезпечення можливості їх практичного застосування під час розробки й дослідження електронних пристроїв. Студенти ознайомлюються з функціонуванням спеціалізованих комп'ютерних систем, протоколами передавання даних, роботою датчиків і виконавчих механізмів, а також із засобами та технологіями проєктування на основі програмованої логіки.

Окремі приклади, що розглядаються у лабораторному практикумі, пов'язані з явищами та технологіями прикладної фізики — зокрема, оптичною передачею даних, оптичними напівпровідниковими сенсорами, методами мультиплексування сигналів та їх демультиплексування. Такі приклади демонструють фізичні принципи функціонування пристроїв, що використовують електромагнітне випромінювання у видимому та інфрачервоному діапазонах, і надають студентам уявлення про сучасні засоби оптоелектроніки та телекомунікацій. У методичних рекомендаціях розглянуто застосування різноманітного CASE-інструментарію для розробки та моделювання вбудованих систем і цифрових пристроїв: середовища Proteus Design SuiteDesign Suite для віртуального моделювання електронних схем, Microchip Studio для програмування мікроконтролерів, Arduino IDE для розробки програмного забезпечення на платформі Arduino, Quartus II/Quartus Prime для проєктування цифрової логіки на програмованих логічних інтегральних схемах (ПЛІС), а також використання мови асемблера для низькорівневого програмування апаратних ресурсів.

ЛАБОРАТОРНА РОБОТА №1.

«Синтез та тестування комбінаційних цифрових схем в інтегрованих середовищах ALTERA QUARTUS II та INTEL QUARTUS PRIME мовою опису обладнання VHDL»

Тема: Синтез та перевірка функціональності комбінаційних цифрових схем мовою опису обладнання VHDL.

Мета: Навчитися самостійно встановлювати інтегроване середовище ALTERA QUARTUS II або INTEL QUARTUS PRIME на особистий персональний комп'ютер або ноутбук та використовувати його для синтезу комбінаційних цифрових схем мовою опису обладнання VHDL.

Теоретичні відомості

1. Інтегроване середовище для синтезу електронних схем Altera Quartus II

Програма Altera Quartus II створена для синтезу електронних аналогових та цифрових схем та програмування спеціалізованих контролерів виробника Altera/Intel, які належать до групи приладів ПЛІС (Програмованих Логічних Інтегральних Схем), які також мають назви VHSIC (Very High Speed Integrated Circuit, інтегроване коло високої швидкодії) або FPGA (Field-Programmable Gate Array, вентильна матриця, яка може бути запрограмована під довільні сферу діяльності). Контролер ПЛІС до якого запрограмована електрона схема, фактично перетворюється в повноцінний функціональний пристрій відповідної функціональності та необмеженого терміну експлуатації. При відсутності ПЛІС-контролера, інтегроване середовище дає можливість змоделювати роботу синтезованої та налагоджених схеми в програмі-симуляторі ModelSim. Окрім виробника Altera/Intel, існують також інші виробники ПЛІС-контролерів, але синтез електронних схем і програмування таких контролерів відбувається за допомогою інших спеціалізованих програм, створених цими виробниками.

2. Мови опису обладнання

Більшість спеціалізованих програм категорії САПР, які працюють з електронними схемами, як правило, повинні використовувати спеціалізовану мову, що містить набір формалізованих синтаксичних механізмів для точного опису вузлів і структури електричної схеми. Зокрема, відома і популярна серед інженерів-розробників програма «PSpice» для аналізу схем використовує мову і набір компіляторів з відкритим кодом «SPICE». Аналогічно, для введення електронних схем в контролер ПЛІС або для аналізу роботи схеми в симуляторі

необхідно підготувати опис електронної схеми мовою опису обладнання. До найбільш популярних і поширених мов опису обладнання належать стандартизована мові VHDL (VHSIC Hardware Description Language), стандартизовані мова Verilog та нестандартизована мова System Verilog.

Хід роботи:

1. Враховуючи перспективу програмування реальних контролерів ПЛІС моделі Altera Cyclone-II EP2C20F484C7, встановіть програму Altera Quartus II 13.0 SP1. Дистрибутив програми самі не шукайте, а завантажте за мережевим посиланням, яке повідомить викладач.

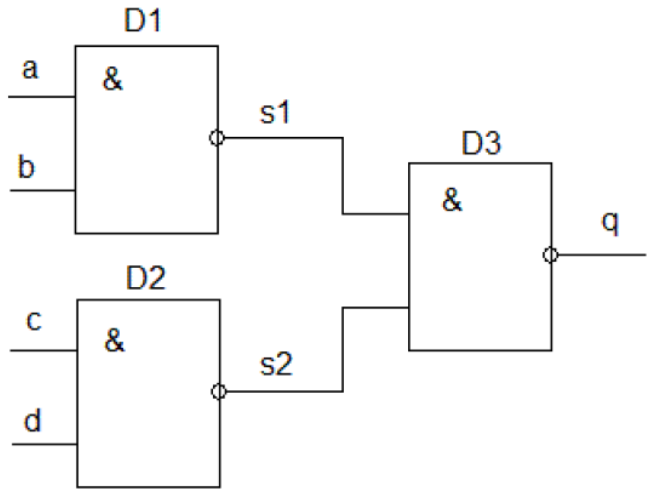


Рис 1. Комбінаційна цифрова схема з трьома елементами AND-NOT.

Лабораторна робота умовно поділяється на дві частини. В першій частині кожному студенту потрібно створити новий проект в програмі Altera Quartus II і додати в цей проект один файл, який точно описує комбінаційну схему, зображену на рисунку 3.

В другій частині потрібно синтезувати комбінаційну схему, яка була використана при виконанні студентами першої частини індивідуальної курсової роботи з дисципліни «Прикладна теорія цифрових автоматів».

2. Розробники проектів інтегрованого середовища програмування Visual Studio добре пам’ятають, що коли розробник запускає майстер створення нового проекту і вводить назву проекту, цей майстер автоматично створює новий каталог з назвою, що співпадає з назвою проекту. Тому пропонується поширити цю традицію ще і на середовище Altera Quartus II, але, на відміну від Visual Studio, воно автоматично не створює підкаталог, однойменний з назвою проекту. Очевидно, що це виправити не складно, якщо виконати наступну послідовність дій:

а) Створіть у файловій системі Вашого персонального комп’ютера новий підкаталог Lab1.

б) Відкрийте середовище Altera Quartus II і запустіть майстер створення нового проекту (**File | New Project Wizard...**). Якщо з’явиться В результаті запуску майстра з’явиться діалогове вікно з заголовком **Introduction (1 of 5)**, в цьому вікні натисніть кнопку **Next >**.

в) В діалоговому вікні з заголовком **Directory, Name, Top-Level Entry (2 of 5)**, в якому потрібно заповнити три текстових поля (рис. 2). В першому полі за допомогою кнопки **Browse** ім’я підкаталога, створеного в пункті а), наприклад C:\Lab1. В другому полі введіть назву проекту Lab1. В третьому полі введіть назву кореневої схеми Lab1 і натисніть кнопку **Next >**.

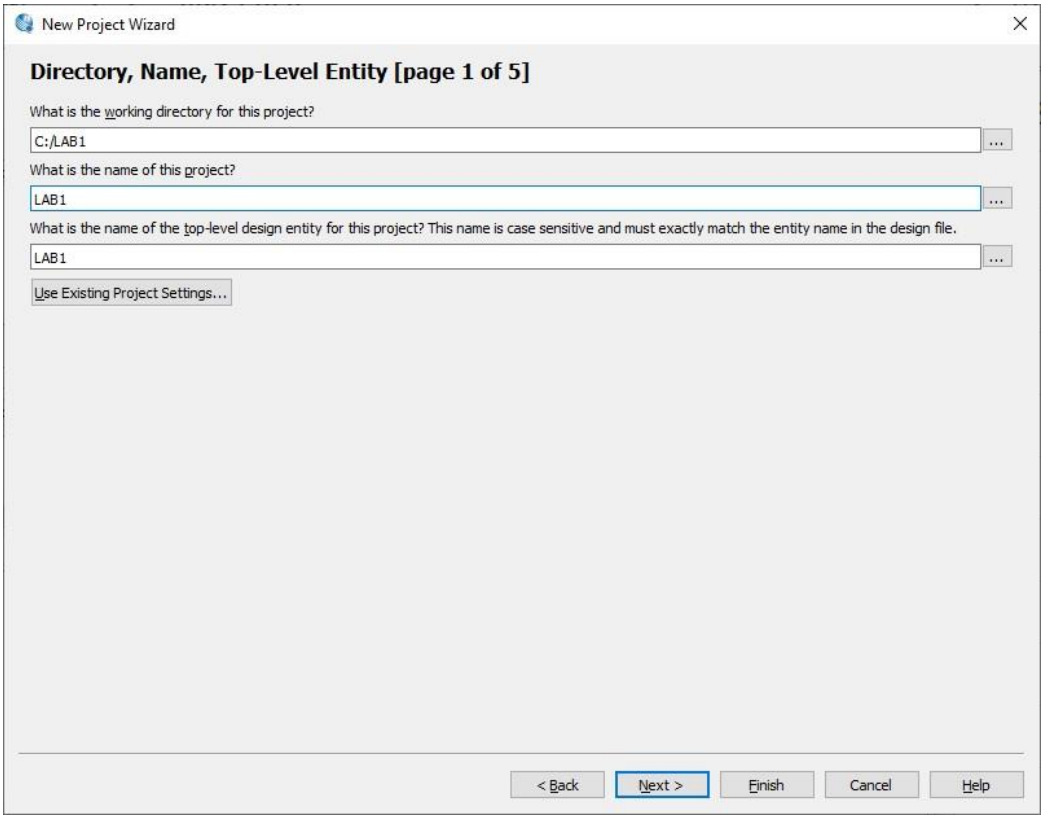


Рис 2. Діалогове вікно з заголовком **Directory, Name, Top-Level Entry (2 of 5)**.

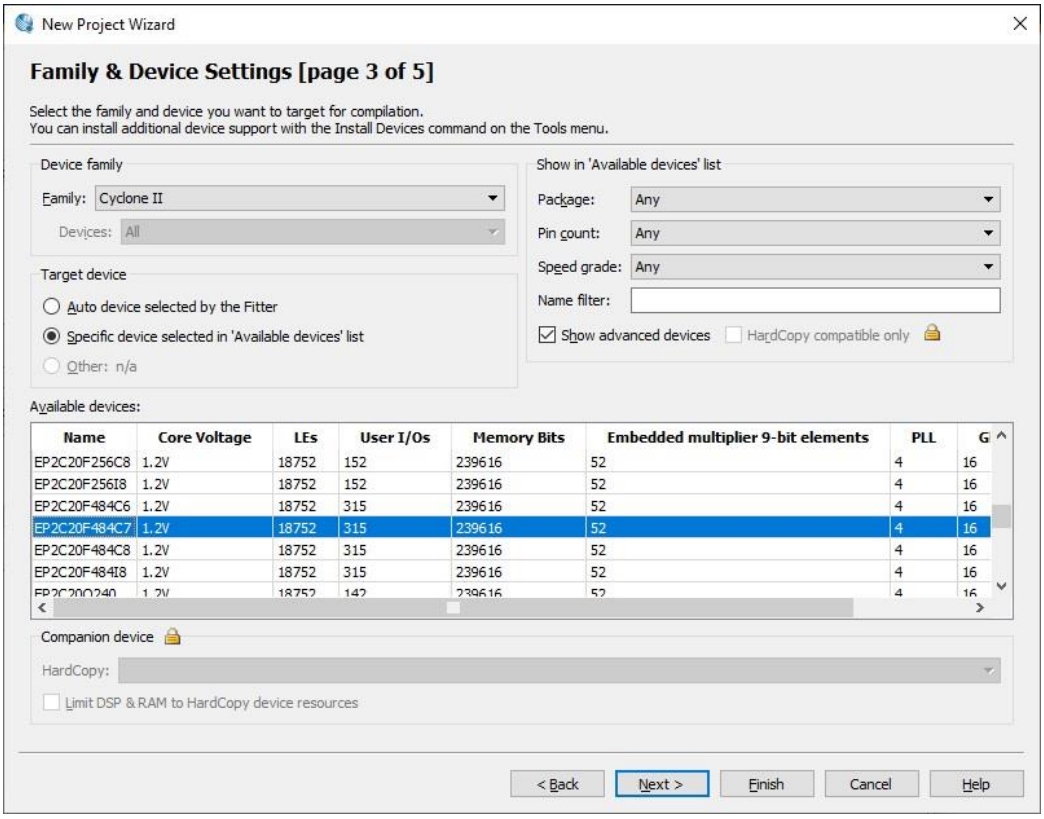


Рис 3. Діалогове вікно з заголовком **Family & Device Settings (3 of 5)**.

г) В діалоговому вікні з заголовком **Add files (3 of 5)** натисніть кнопку **Next >**.

д) В діалоговому вікні з заголовком **Family & Device Settings (3 of 5)** у випадаючому списку **Family** оберіть **Cyclone II** (рис. 3), у переліку **Available devices:** оберіть контролер **EP2C20F484C7** і натисніть кнопку **Next >**.

е) В діалоговому вікні з заголовком **EDA Tools (4 of 5)** натисніть кнопку **Next >**.

ж) В діалоговому вікні з заголовком **Summary (5 of 5)** натисніть кнопку **Finish** і переконайтесь, що середовище успішно створило новий проект, в якому поки що немає жодної схеми. Проаналізуйте зміст підкаталогу Lab1, створеному в пункті а), відкрийте в редакторі Notepad файл **Lab1.qpf** і ознайомтесь зі структурою цього файлу.

3. У завантаженому проекті Lab1, відкритому в середовищі Altera Quartus II, відкрийте команду **File|New...**, в діалоговому вікні **New** оберіть **Design Files|VHDL File** і натисніть **OK**. В правій частині середовища в редакторі файлу **Vhdl1.vhd** вставте такий текст:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY Lab1 IS
PORT (a,b,c,d : IN STD_LOGIC;
lg5,lg6,lg7 : OUT STD_LOGIC);
END Lab1;
```

```
ARCHITECTURE LogicFunction OF Lab1 IS
signal s1,s2:std_logic;
BEGIN
s1 <= not (a and b);
lg5 <= s1;
s2 <= not (c and d);
lg6 <= s2;
lg7 <=not (s1 and s2);
END LogicFunction;
```

Якщо Ви назвали проект не Lab1, а у інший спосіб, то змініть у файлі назву Lab1, яке присутнє на 3, 6 та 7 рядку, на справжнє ім'я файлу. Після цього виконайте команду **File|Save All**, залиште без зміни ім'я файлу **LAB1.vhd** і натисніть **Save/Зберегти** (в залежності від мови інтерфейсу операційної системи).

4. Після успішного збереження усіх файлів проекту виконайте команду **Processing|Start Compilation...** і дочекайтесь успішного завершення процедури компіляції проекту. Компіляція триває 20-60 секунд і повинна завершитися виводом діалогового вікна з повідомленням «**Full compilation was successful**». Якщо в цьому ж повідомленні також присутня фраза «**(... warnings)**» — ігноруйте це повідомлення.

5. В програмі Altera Quartus II відкрийте пункт меню **Tool|Options**, в лівій частині вікна **Options** виділіть розділ **General|EDA Tool Options**.

6. В правій частині напроти параметра **ModelSim-Altera** натисніть кнопку з трьома крапками і оберіть каталог

...\altera\13.0sp1\modelsim_ase\win32aloem.

Наприклад, якщо программа Quartus встановлена в каталог **C:\Altera**,

то каталог, введений в параметрі **ModelSim-Altera** має бути **C:\altera\13.0sp1\modelsim_ase\win32aloem**

7. В програмі QuartusII відкрийте пункт меню **Tool|Run Simulation Tools|RTL Simulation**, якщо потрібно, оберіть мову опису обладнання **VHDL**, почекайте (~10с) доки завантажиться симулятор ModelSim, розгорніть його на весь екран.

9. В симуляторі ModelSim відкрийте пункт меню **File|New|Project...**, у верхньому текстовому полі окна "**Create Project...**" оберіть ім'я ModelSim-проекту, що співпадає з ім'ям Вашего Quartus-проекту і натисніть кнопку **OK**.

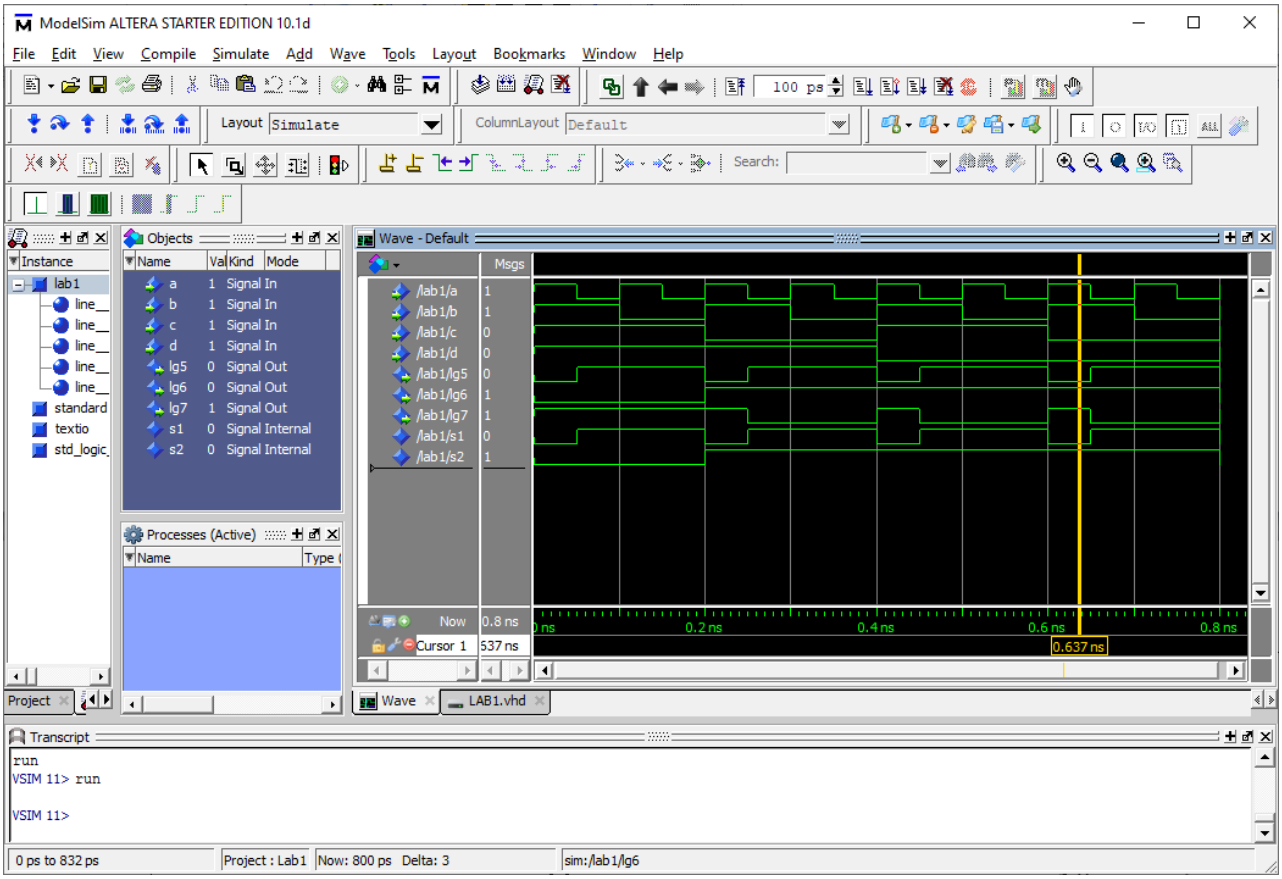


Рис 4. Часові графіки вхідних і вихідних сигналів цифрової схеми, представленої на рис 1.

10. У вікні "**Add items to the Project**" натисніть "**Add Existing File**", у вікні "**Add file to Project**" натисніть кнопку "**Browse...**", знайдіть VHD-файл Вашого проекту (для цього, як правило, достатньо двічі перейти в надкаталог), після вибору файла натисніть кнопку **ОК**, а у вікні "**Add items to the Project**" натисніть кнопку **Close**.

11. В лівій частині симулятора ModelSim знайдіть доданий VHD-файл, відкрийте на ньому правою кнопкою миші контекстне меню і натисніть **Edit**.

12. В симуляторі ModelSim натисніть пункт меню **Compile|Compile Selected** і переконайтесь, что в нижній частині симулятора ModelSim з'явилось повідомлення зеленого кольору **Compile of ...VHD was successful**.

13. В симуляторі ModelSim натисніть пункт меню **Simulate|Start Simulation...** У вікні "**Start Simulation**" на закладинці **Design** виділіть мишкою **Work|<Имя проекта>** і натисніть **OK**. Переконайтесь, що в середній частині симулятора ModelSim з'явився список усіх іменованих контактів вашої схеми.

14. Правою кнопкою миші відкрийте контекстне меню будь-якого контакту цифрового входу або виходу вашої схеми, оберіть пункт меню **Add to|Wave|Signals in design** . В правій частині вікна повинна з'явитись нова робоча область **Wave** чорного кольору, знизу дві закладинки **Wave** і **<Ваш файл>.VHD**

Перейдіть на закладку **Wave** і переконайтесь, что на ній продубльовано список контактів вашої схеми, кожен контакт в одному екземплярі.

15. Відкрийте контекстне меню першого цифрового входу вашої схеми і оберіть пункт **Force...** або **Clock...** для встановлення відповідно статичного сигналу або такого, що періодично змінюється. Наприклад, якщо схема має 4 входи, то доцільно для всіх задати периодичний сигнал, для першого входу період 100, для другого 200, для третього 400, для четвертого 800.

16. В симуляторі ModelSim у верхній частині знайдіть кнопку **Run** (F9), яка знаходиться праворуч від текстового поля с текстом "100 ps". Натискайте її до тих пір, поки графіки не заповнять максимальний період вхідного сигналу. Наприклад, для сигналів пункта 10 потрібно 8 разів натиснути кнопку **Run** і тоді графіки заполнять інтервал 800 ms, тобто максимальний період.

17. В результаті виконаних дій програма-симулятор ModelSim має побудувати часові графіки залежності вхідних та вихідних сигналів цифрової схеми, представленої на рис 1. Проаналізуйте ці графіки.

18. Якщо аналіз графіків показав, що схема функціонує правильно, то переходьте до другої частини лабораторної роботи. Для цього закрийте проект Lab1 і створіть його резервну копію, наприклад створіть ZIP-архів усього каталогу Lab1.

19. Відкрийте знову проект Lab1 і змініть файл **Lab1.vhd** у такий спосіб, щоб у ньому була віддзеркалена вже не цифрова схема, представлена на рис. 1, а синтезована комбінаційна схема, яку Вам повідомить викладач. Повторно виконайте компіляцію і моделювання роботи схеми за Вашим індивідуальних завдання. Якщо логіка роботи схеми збігається з таблицею істинності, наведеною у курсовій роботі, продемонструйте виконання індивідуального завдання викладачеві, якщо необхідно дайте відповіді на додаткові запитання і отримайте оцінку.

Запитання для закріплення матеріалу лабораторної роботи:

1. Що таке комбінаційні електронні схеми?

3. Які існують методи тестування і перевірки комбінаційних схем, які переваги і недоліки кожного методу?

2. Як співвідносяться часи релаксації у аналогових та цифрових комбінаційних схемах?

4. Скільки можливих комбінацій вхідних сигналів може мати цифрова схема?

ЛАБОРАТОРНА РОБОТА №2.

«Синтез та тестування скінченого автомату в інтегрованих середовищах ALTERA QUARTUS II та INTEL QUARTUS PRIME мовою опису обладнання VHDL»

Тема: Синтез та перевірка функціональності скінченого автомату Мілі мовою опису обладнання VHDL.

Мета: Навчитися самостійно встановлювати інтегроване середовище ALTERA QUARTUS II або INTEL QUARTUS PRIME на особистий персональний комп'ютер або ноутбук та використовувати його для синтезу скінченого автомату Мілі мовою опису обладнання VHDL.

Теоретичні відомості

1. Інтегроване середовище Altera Quartus II для програмування ПЛІС-контролерів. Для виконання лабораторної роботи №2 необхідною умовою є виконання попередньої лабораторної роботи №1. В цій лабораторній роботі добре продемонстровано, що програма Altera Quartus II створена для синтезу електронних схем та програмування спеціалізованих контролерів Altera/Intel, які належать до групи приладів ПЛІС (VHSIC, FPGA). Такий ПЛІС-контролер до якого запрограмована електрона схема, фактично перетворюється в повноцінний функціональний пристрій з необмеженим терміном експлуатації. Окрім виробника Altera/Intel, існують також інші виробники ПЛІС-контролерів та інші відповідні спеціалізовані програми, створені цими виробниками.

2. Комбінаційні та секвенційні схеми. Складні інтегровані логічні схеми доцільно поділяти на два принципово різних класи – схеми комбінаційної логіки та схеми секвенційної (послідовної) логіки. У комбінаційних пристроях вихідні сигнали цілком визначаються набором вхідних сигналів. Такі пристрої застосовуються для формування, конвертації, кодування та оптимізації цифрових даних, які, у свою чергу, підлягають подальшій обробці, збереженню або передачі за допомогою інших, некомбінаційних схем. Натомість, у пристроях секвенційної логіки вихідні сигнали визначаються не лише набором вхідних сигналів, а ще й передісторією функціонування приладу. Це означає, що секвенційні прилади зберігають пам'ять про попередні режими своєї роботи, а комбінаційні прилади такої пам'яті не мають.

3. Цифрові автомати. Окремий клас схем секвенційної логіки складають цифрові автомати. Теорія автоматів виникла в межах теоретичної кібернетики в епоху стрімкого розвитку обчислювальної техніки у другій половині XX століття, коли в ході проектування складних схем реальних обчислювальних машин теоретичні моделі цих схем розглядались у вигляді кінцевих автоматів. З часом розвиток інформаційних технологій істотно розширив область

застосування теорії автоматів від моделювання апаратних засобів цифрової техніки до фундаментальних засад сучасної інформатики. Виявилось, що абстракції і моделі теорії автоматів використовуються багатьма спорідненими науковими дисциплінами, серед яких найбільш значущими є теорія алгоритмів, математична лінгвістика, теорія кодування, теорія обчислювальної складності, математичне програмування, теорія логічних моделей та теорія абстрактних машин.

Хід роботи:

1. Виконайте лабораторну роботу №1. Використовуючи досвід її виконання, відкрийте програму Altera Quartus II 13.0 SP1 і створіть новий проект з назвою, наприклад, Lab2, для програмування ПЛІС-контролерів моделі Altera Cyclone-II EP2C20F484C7.
2. В першій частині потрібно створити новий проект в програмі Altera Quartus II і додати в цей проект один файл, який точно описує комбінаційну схему, зображену на рисунку 3.

Лабораторна робота №2 умовно поділяється на дві частини, як і лабораторна робота №1. В першій частині потрібно синтезувати та виконати симуляцію роботи скінченого автомату Мілі, граф переходів якого зображено на рисунку 5. Цей автомат має чотири стани A,B,C,D та три зовнішніх сигнали: R (reset), P (pulse) та Clock. По передньому зростаючому фронту (rising_edge) сигналу Clock або по його задньому спадаючому фронту (falling_edge) відбувається перехід станів, який залежить від значень R і P). Стрілка зверху над станом A означає, що стан A є вихідним при увімкнені роботи схеми або перехід у стан A відбувається негайний перехід з будь-якого іншого стану B/C/D при надходження сигналу R=1.

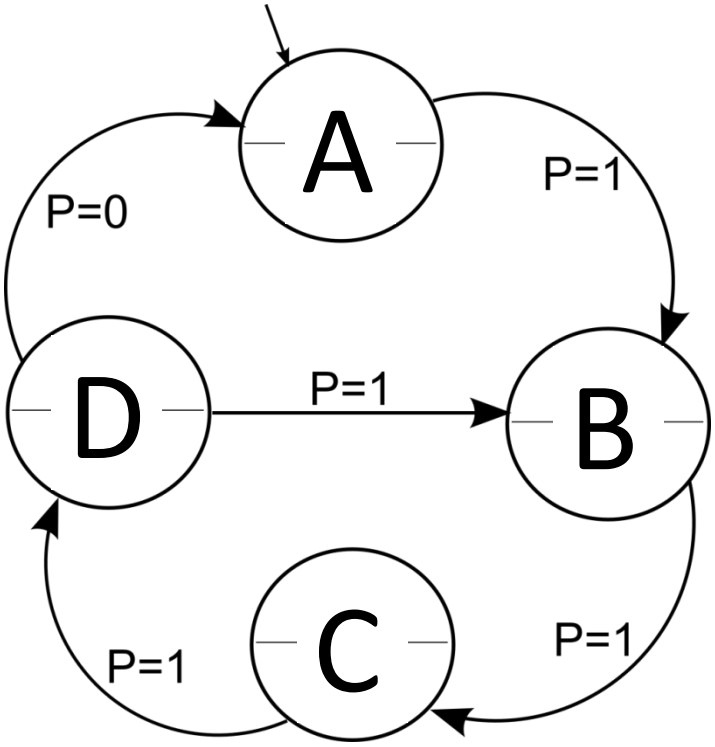


Рис 1. Скінчений автомат Мілі на 4 стани.

Щоб реалізувати цей автомат, додайте в створений проект Lab2 новий файл Vhdl1.vhd з таких VHDL-кодом, що описує функціональність скінченого автомату, представленого на рисунку 5:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY LAB1 IS
PORT (clock: IN STD_LOGIC; P: IN STD_LOGIC; reset: IN STD_LOGIC;
RA,RB,RC,RD: OUT STD_LOGIC;
a1,a2,a3,a4,a5,a6,a7: OUT STD_LOGIC);
END LAB1;
ARCHITECTURE RTL OF LAB1 IS
TYPE State_type IS (A, B, C, D);
      SIGNAL State : State_Type;
BEGIN
a1 <= '1'; a2 <= '1'; a3 <= '1'; a4 <= '1'; a5 <= '1'; a6 <= '1'; a7 <= '1';
--4th digit is TURNED OFF
 PROCESS (clock, reset)
 BEGIN
 If (reset = '1') THEN -- Upon reset, set the state to A
      State <= A;
 ELSIF rising_edge(clock) THEN -- if there is a rising edge of the
                  -- clock, then do the stuff below
```

```
      CASE State IS
            -- If the current state is A and P is set to 1, then the
            -- next state is B
            WHEN A =>
                  IF P='1' THEN
                        State <= B;
                  END IF;
            -- If the current state is B and P is set to 1, then the
            -- next state is C
            WHEN B =>
                  IF P='1' THEN
                        State <= C;
                  END IF;
            -- If the current state is C and P is set to 1, then the
            -- next state is D
            WHEN C =>
                  IF P='1' THEN
                        State <= D;
                  END IF;
            -- If the current state is D and P is set to 1, then the
            -- next state is B.
            -- If the current state is D and P is set to 0, then the
            -- next state is A.
            WHEN D=>
                  IF P='1' THEN
                        State <= B;
                  ELSE
                        State <= A;
                  END IF;
            WHEN others =>
                  State <= A;
      END CASE;
 END IF;
 END PROCESS;
-- Decode the current state to create the output
-- if the current state is D, R is 1 otherwise R is 0
RA <= '1' WHEN State=A ELSE '0';
RB <= '1' WHEN State=B ELSE '0';
RC <= '1' WHEN State=C ELSE '0';
RD <= '1' WHEN State=D ELSE '0';
END RTL;
```

В цьому коді потрібно замінити LAB1, яке тричі тут присутнє, на діючу назву Quartus-проекту.

Після успішної компіляції схеми виконайте симуляцію роботи автомата, як це було зроблено для комбінаційної схеми в лабораторній роботі №1. Очевидно, що для успішної симуляції необхідно вдало обрати часові тривалості сигналів R, P і Clock, щоб перевірити роботу схеми при максимально усіх можливих комбінаціях цих сигналів. Приклад такої симуляції наведений на рисунку 6. На ньому добре видно, що від початку роботи схеми до моменту часу 1, коли сигнал P=1, відбувається успішні переходи автомату за переднім фронтом сигналу Clock за такою послідовністю станів: A-B-C-D-B-C-D-B до того моменту, поки P не перемикається в значення 0 (момент часу 1). З цього моменту стан B вже не змінюється за передніми фронтами сигналу Clock, а отже автомат залишається в одному стані B до того моменту, поки на надійде сигнал R=1 (reset=1), що виконує скидання автомату. Цей сигнал надходить в момент часу 2, тому в цей момент автомат переходить в стан A і не може більше виконати ніяких переходів поки утримується сигнал R=1.

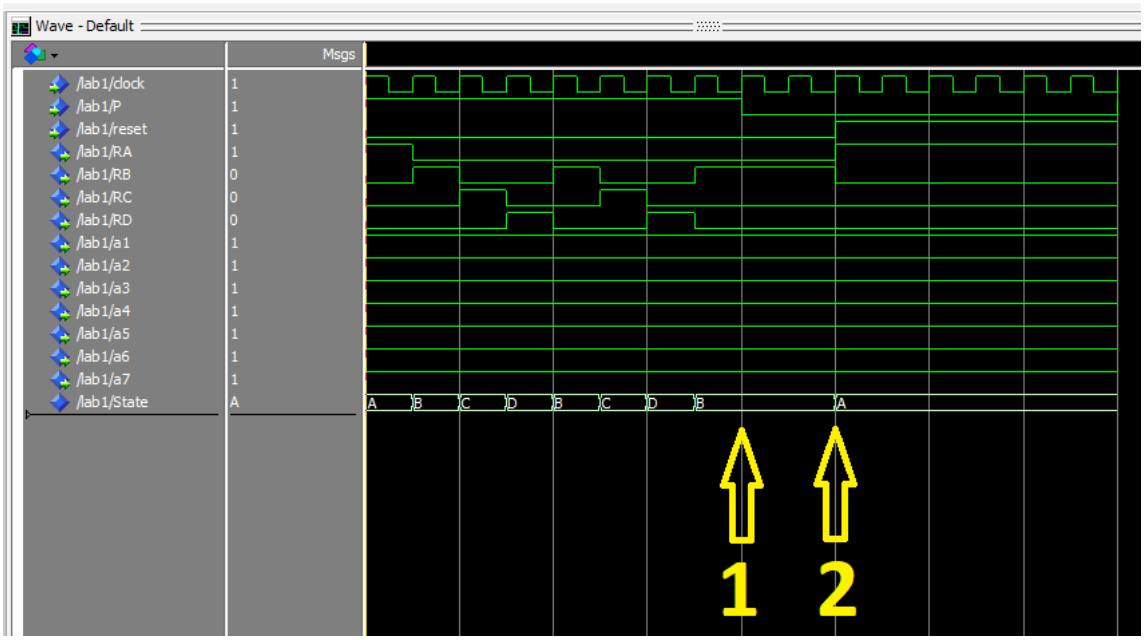


Рис 1. Скінчений автомат Мілі на 4 стани.

Якщо симуляція роботи схеми працює добре, збережіть проект, відкрийте його копію і використовуючи цю копію як шаблон, виконайте другу частину лабораторної роботи. В другій частині додайте до автомату 1 або 2 нових стани E/F, додатково до сигналу P можете додати ще 1-2 сигнали Q/S, можете додати до автомату нові переходи або у інший спосіб змініть роботу автомату. За рахунок цього змініть логіку роботу скінченого автомату у такий спосіб, щоб у кожного студента утворився новий унікальний автомат, автомати повинні бути РІЗНІ. За допомогою редактора MSPAINT або іншого редактора побудуйте схематичне зображення графу переходів нового автомату.

Повторно виконайте компіляцію і симуляцію роботи схеми за новим скінченим автоматом. Якщо логіка роботи схеми співпадає зі схематичним

зображенням, продемонструйте виконання першої і другої частини лабораторної роботи викладачеві, якщо необхідно дайте відповіді на його додаткові запитання і отримайте оцінку.

Запитання для закріплення матеріалу лабораторної роботи:

1. Що таке комбінаційні і секвенційні електронні схеми?

2. Що таке цифровий автомат Мура і цифровий автомат Мілі?

3. Що таке скінчені і нескінчені цифрові автомати?

4. Що таке детермінований і недетерміновані скінчені автомати?

ЛАБОРАТОРНА РОБОТА №3.

«Мультиплексування сигналів в оптичних лініях зв’язку на мікроконтролері ATtiny13»

Тема: Моделювання багатотонального генератора для мультиплексування сигналів з використанням мікроконтролера в середовищі ISIS.

Мета: Ознайомитися з особливостями роботи в середовищі моделювання електронних схем ISIS Proteus, та середовищі розробки Microchip Studio для програмування мікроконтролерів AVR. Скласти схему двотонального генератора для мультиплексування сигналів, розробити та налагодити відповідне програмне забезпечення. Набути практичних навичок інтеграції програмного коду з апаратною частиною електронної системи та закріпити вміння розробки функціональних вузлів з використанням мікроконтролерів.

Теоретичні відомості

Вивчення методів оптичної передачі даних є критично важливим для сучасних інформаційних технологій і систем зв’язку, де оптоволоконні канали забезпечують високу пропускну здатність, низькі втрати сигналу та стійкість до електромагнітних завад на великих відстанях. У складних інтегрованих системах, таких як безпілотні літальні апарати, оптоволоконні лінії зв’язку використовуються для забезпечення надійного, високошвидкісного обміну інформацією між сенсорними та керуючими модулями.

Для ефективного управління такими оптичними системами широко застосовуються мікроконтролери, які виконують роль програмованих інтерфейсів між апаратною складовою та системою управління. Мікроконтролери забезпечують збір і обробку даних від оптичних сенсорів, реалізують протоколи передачі та корекції помилок, а також координують роботу оптоелектронних модулів у реальному часі. Незважаючи на обмежені обчислювальні ресурси, мікроконтролери дозволяють реалізувати складні алгоритми керування і забезпечують гнучкість інтеграції оптичних каналів із загальною електронною архітектурою системи. Мікроконтролери містять велику кількість вбудованих функціональних блоків, що дає можливість інтегрувати їх у систему з мінімальною кількістю зовнішніх компонентів. До типового складу мікроконтролера входять:

Схема початкового запуску процесора (Reset);

Генератор тактових імпульсів;

Центральний процесор (CPU);

Пам’ять програм (EPROM, EEPROM або Flash) та інтерфейс її програмування;

Оперативна пам’ять (RAM);

Засоби введення/виведення даних;

Таймери/лічильники;

Механізми обробки переривань;

Аналогові інтерфейси (ADC, DAC);

Послідовні інтерфейси (UART, SPI, I^2C тощо);

Засоби підключення до комп’ютера (через USB або інші шини);

Підтримка підключення зовнішньої пам’яті (у мікропроцесорному режимі).

Сучасні мікроконтролери виготовляються з використанням CMOS-технології, що дає можливість значно зменшити площу кристала та знизити енергоспоживання. Компанія ATMEL Corporation (нині частина Microchip Technology) — один із світових лідерів у виробництві мікросхем енергонезалежної пам’яті, мікроконтролерів з пам’яттю типу Flash і програмованих логічних пристроїв — розпочала розробку RISC-архітектурних мікроконтролерів у середині 1990-х років, спираючись на напрацьовані інженерні рішення та досвід у галузі мікроелектроніки.

AVR-мікроконтролери вирізняються високою швидкодією та універсальністю порівняно з багатьма іншими поширеними мікроконтролерами. Завдяки ефективній архітектурі вони здатні виконувати певні завдання, для яких раніше були необхідні лише 16-розрядні мікроконтролери. Це дає можливість значно зменшити вартість готових електронних пристроїв за збереження функціональності. У мікроконтролерах AVR реалізована Гарвардська архітектура, яка передбачає розділення пам’яті програм і пам’яті даних, кожна з яких має окремі шини для доступу. Така організація дає можливість мікроконтролеру одночасно звертатися як до пам’яті програм, так і до пам’яті даних, що підвищує загальну продуктивність системи.

Розділення шин доступу також дає змогу використовувати для кожного типу пам’яті шини з різною розрядністю. Крім того, способи адресації та доступу до пам’яті програм і пам’яті даних є відмінними, що забезпечує більшу гнучкість у побудові обчислювальної архітектури.

Порти вводу/виводу в мікроконтролерах AVR мають функцію Read-Modify-Write, що дає можливість змінювати стан окремих ліній порту без впливу на інші. Кожен порт керується трьома регістрами: даних, напрямку і стану ліній. Це дає змогу одночасно налаштовувати вхідні та вихідні лінії, а також зчитувати реальний стан зовнішніх сигналів. Особливістю є комбінована схема

«підтягування» ліній до високого рівня, яка активується лише при режимі вводу, забезпечуючи надійну роботу та захист від коротких замикань.

Взаємодія з портом здійснюється через три розташовані в адресному просторі I/O регістри: регістр даних — PORTx, регістр напрямку передачі даних — DDRx та регістр читання стану входу — PINx. Регістр вводу PINx доступний лише для читання, тоді як регістри PORTx і DDRx підтримують як читання, так і запис. Всі виводи порту оснащені індивідуально вбудованими підтягуючими резисторами.

Вихідні буфери портів здатні забезпечувати струм до 20 мА, що дає можливість безпосередньо керувати світлодіодними індикаторами. Проте, якщо виводи порту використовуються як входи, а зовнішні лінії підтягнуті до низького рівня (логічного нуля), активні внутрішні підтягуючі резистори до високого рівня можуть спричиняти зайве енергоспоживання.

У лабораторній роботі буде використовуватися мікроконтролер ATtiny13. Цей мікроконтролер підтримує програмування безпосередньо в схемі через послідовний інтерфейс SPI, має режими зниженого енергоспоживання, детектор зниження напруги живлення, можливість програмного зниження частоти тактового генератора, а також багаторівневу систему переривань з підтримкою черги переривань і джерелами як внутрішніх, так і зовнішніх переривань.

Загальні характеристики мікроконтролера ATtiny13:

(Детальніше з характеристиками можна ознайомитись за посиланням https://www.alldatasheet.com/datasheet-pdf/pdf/255658/ATMEL/ATtiny13A.html)

• 120 інструкцій оптимізованих для програмування на мовах високого рівня;
• 32 регістри загального призначення;
• майже кожна інструкція виконується за 1 такт генератора, за рахунок чого швидкодія досягає 20 MIPS (20 мільйонів операцій за секунду);
• 1кілобайт флеш-пам'яті для програм ;
• 64байт EEPROM (незалежна пам'ять);
• 64байт SRAM (оперативна пам'ять).

Периферія:
• один 8 бітовий таймер / лічильник;
• два ШІМ канали;
• 4 канальний 10ти бітовий АЦП;
• аналоговий компаратор;
• SPI послідовний інтерфейс;
• Watchdog таймер, зовнішні переривання на всіх ніжках.
Особливі можливості:
• внутрішньосхемние відлагодження по одному по одному проводу debugWIRE;
• програмування по послідовному SPI інтерфейсу;

- різні джерела переривання як зовнішні, так і внутрішні, 3 режими енергозбереження
- детектор зниження напруги живлення, вбудований тактовий генератор.

Значення виводів мікро контролера показано на рис. 2.

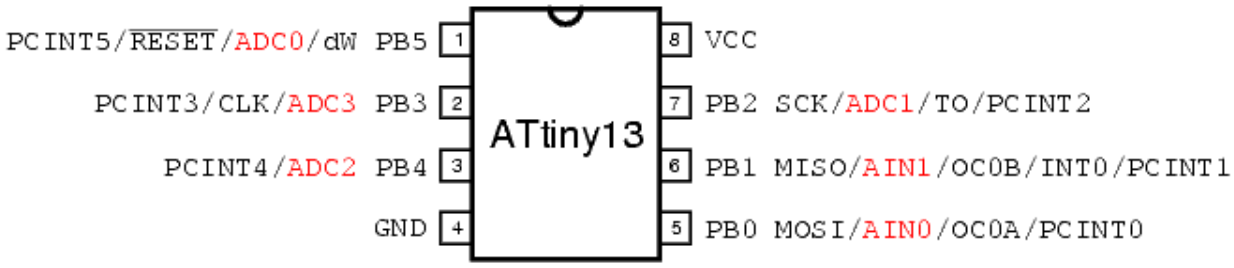


Рис. 1. Значення виводів мікроконтролера ATtiny13

Живлення, частота: 1.8 - 5.5В (для ATtiny13V) до 10 МГц 2.7 - 5.5В (для ATtiny13) до 20МГц. У робочому режимі споживає 240мкА при живленні 1.8В і частоті тактового генератора 1МГц. У режимі нергозбереження Power-down споживає менше 0.1мкА при 1.8В.

Програмування:

Для мікроконтролера найбільш зручний режим програмування по послідовному SPI інтерфейсу. Для реалізації цього режиму необхідно підключити мікроконтролер до програматору по SPI інтерфейсу (MOSI, MISO, SCK, RESET, GRD), живити мікроконтролер номінальною напругою. Мікроконтролер може програмуватися прямо в робочій схемі (внутрішньосхемне програмування) але при цьому необхідно дотримуватися умови – лініям SPI інтерфейсу при програмуванні не повинно нічого заважати (великі ємності, маленькі опори відносно спільного проводу GND та ін.).

Пакет моделювання електронних схем Proteus Design SuiteVSM.

Proteus Design SuiteVSM складається з двох самостійних програм ISIS і ARES. ARES це програма для розведення доріжок друкованих плат. Її не будемо розглядати.

Основною програмою є ISIS, в ній передбачений гарячий зв'язок з ARES для передачі проекту для розведення плати.

При запуску програми з'являється головне вікно.

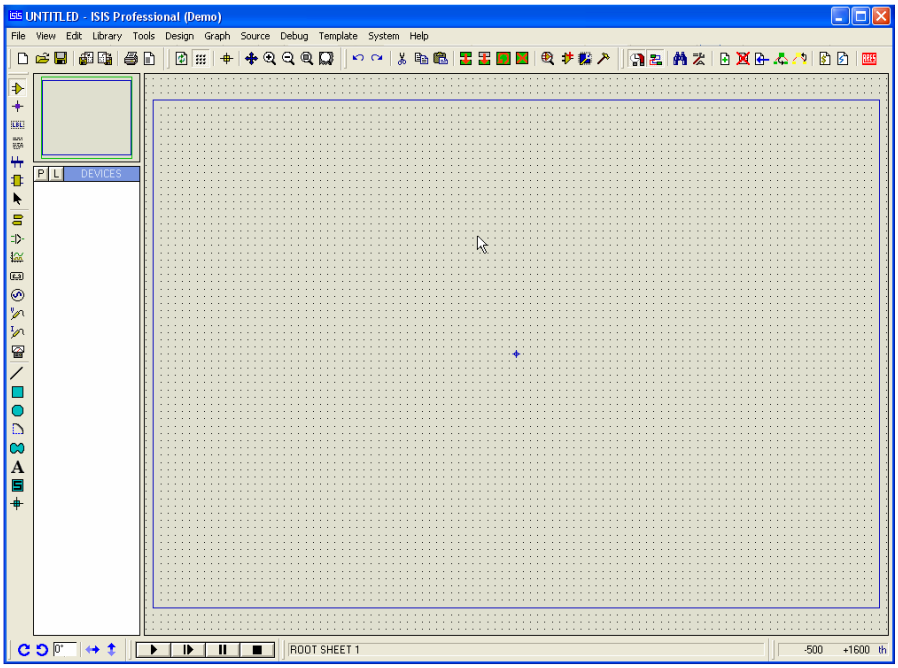


Рис. 2. Головне вікно програми

Найбільший простір відведено під вікно редагування EDIT WINDOW. Саме в ньому відбуваються всі основні процеси створення, редагування та налагодження схеми пристрою.

Зліва вгорі маленьке вікно попереднього перегляду Overview Window, з його допомогою можна переміщатися по вікну редагування (клацаючи лівою кнопкою мишки по вікну попереднього перегляду, ми переміщаємо вікно редагування по схемі, якщо схема не вміщується у вікно). Переміщати вікна редагування за схемою можна ще так - утримуючи кнопку SHIFT рухати курсор миші, не натискаючи її кнопок, по вікну редагування.

Наближати і віддаляти схему у вікні можна відповідно кнопками F6 і F7 або ж колесом “Scroll” миші, F5 центрує схему у вікні, а натискання F8 підганяє розмір схеми під вікно редагування.

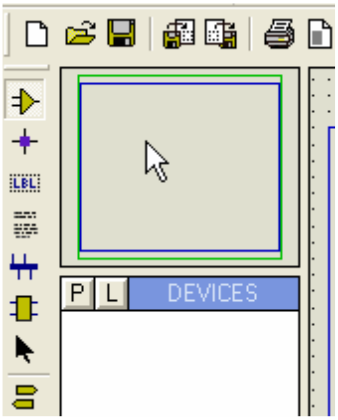


Рис. 3. Вікно попереднього перегляду

Під вікном попереднього перегляду знаходиться Object Selector список вибраних в даний момент компонентів, символів та інших елементів. Виділений в списку об'єкт відображається у вікні попереднього перегляду. Всі можливі функції та інструменти Proteus Design SuiteVSM доступні через меню розташоване в самому верху основного вікна програми, через піктограми, які знаходяться під меню і в лівому кутку основного вікна й через гарячі клавіші, які можуть перепризначатися користувачем.

Внизу основного вікна розташовані: зліва направо кнопки обертання і розвороту об'єкта навколо своєї осі, панель управління інтерактивною симуляцією (ПУСК - ПОКРОКОВИЙ РЕЖИМ – ПАУЗА - СТОП), рядок статусу (в ньому відображаються: помилки, підказки, поточний стан процесу симуляції і т.д.) і координати курсору відображаються в дюймах.

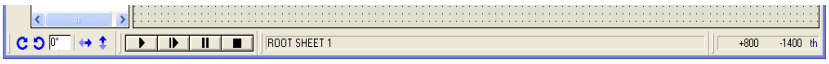


Рис. 4. Нижня частина головного вікна

Для освоєння основних функцій програми, відкриємо один з вже існуючих проектів. Виберемо в меню FILE опцію LOAD DESIGN. Завантажимо файл Sample / Interactive Simulation / Animated circuits / AC01.DSN.

Запустимо проект, натиснувши на панелі кнопку ПУСК

Рис. 5. Панель управління інтерактивною симуляцією

Ця схема демонструє дію змінного струму в ланцюгу. Частота генератора занижена до 0.5 Гц для наочності.

Колір і яскравість проводів визначають полярність і рівень напруги, стрілки напрямок струму. Червона крапка на зображенні генератора показує поточне «положення» синусоїди.

Для того щоб маніпулювати об'єктами їх потрібно спочатку виділити, це можна зробити тільки на зупиненому проекті. Для виділення одного об'єкта треба клацнути по ньому правою кнопкою миші. Для виділення групи можна або утримуючи CTRL послідовно клацати правою кнопкою по всіх об'єктах або утримуючи праву кнопку протягнути область виділення по необхідним об'єктам. Виділяти об'єкти треба обережно, повторне клацання правою кнопкою миші по виділеному об'єкту видалить його, (видалити виділені об'єкти можна ще, натиснувши кнопку DELETE). Але це не страшно скасувати останні і всі попередні дії по порядку можна за допомогою кнопок скасування (UNDO, REDO).

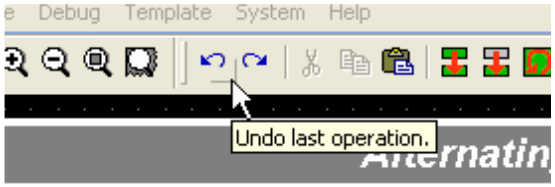


Рис. 6. Кнопки скасування

Кнопки скасування діють як назад, за хронологією так і вперед. Виділені об'єкти можна переміщати за схемою, схопивши їх лівою кнопкою миші пересунувши в потрібне місце відпустити кнопку. А за допомогою цих кнопок виробляються групові операції з виділеними об'єктами. По порядку: копіювання, переміщення, поворот і видалення.

Рис. 7. Кнопки виконання групових операцій

Відкриємо наступний проект rgb_sus.DSN, який знаходиться в тій же папці, попередній закриється, автоматично запитавши вас «чи не бажаєте зберегти зміни». Запустіть проект.

Проект ілюструє роботу мікроконтролера з портами вводу/виводу .

Щоб відкрити вікно редагування компонентів, потрібно або виділивши компонент клацнувши по ньому лівою кнопкою миші або помістивши на нього

курсор, не натискаючи кнопок миші, натиснути CTRL + E. Відкриємо вікно редагування.

Для того щоб запустити **ISIS**, натиснути на кнопку **ПУСК** і вибрати програми, **Proteus Design Suite7 Professional**, а потім **ISIS 7 Professional**.

При запуску програми з'явиться основне вікно.

Найбільша частина простору відводиться під вікно редагування **EDIT WINDOW**.

Саме в ньому відбуваються основні процеси створення, редагування та налагодження схеми пристрою.

Зліва зверху знаходиться маленьке вікно попереднього перегляду OverView Window з його допомогою можна переміщуватись по вікну редагування.

Переміщувати вікно редагування по схемі можна утримуючи натиснутою кнопку SHIFT переміщувати курсор миші, не натискаючи її кнопок, по вікну редагування.

Наближати та віддаляти схему у вікні можна відповідно кнопками F6 і F7, або колесом миші, F5 центрує схему в вікні, а натискання F8 підганяє розмір схеми під вікно редагування.

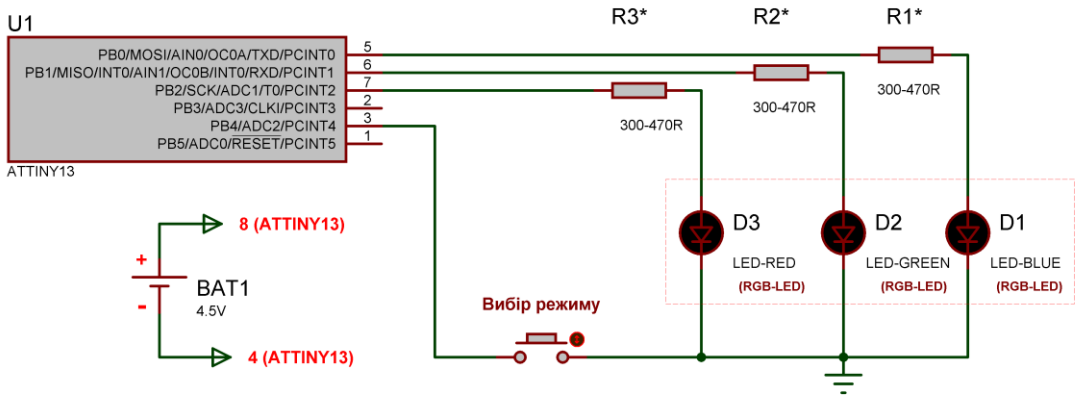


Рис. 8. Схема лабораторної роботи.

Під вікном перегляду знаходиться **Object Selector** список вибраних в даний момент компонентів, символів та інших елементів. Виділений в списку об'єкт відображається в вікні попереднього перегляду.

Всі можливі функції і інструменти Proteus Design Suiteдоступні через меню розташоване вгорі основного вікна програми, через іконки, які розташовані під меню і в лівому куті основного вікна, і через гарячі клавіші, які можуть перепризначатися користувачем.

Знизу головного вікна знаходиться: зліва направо кнопки обертання і розвороту об’єкта, панель управління симуляцією, рядок статусу і координати курсору.

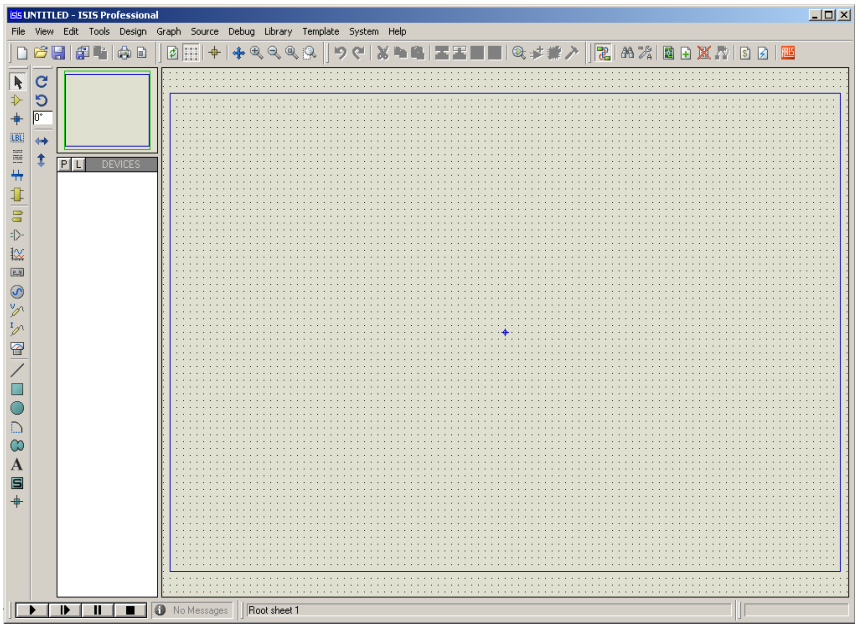


Рис. 9. Головне вікно програми ISIS

Для додавання компонентів необхідно перейти в режим COMPONENT(компоненти), нажав на відповідну іконку. Далі натиснувши на іконку P (Pick Devices) або двічі натиснути лівою кнопкою в полі вибору компонентів Object Selector, ми потрапляємо в бібліотеку компонентів. Використовуючи пошук знаходимо необхідний компонент.

Для редагування компоненту необхідно натиснути на нього двічі лівою кнопкою миші.

Для того щоб додати компонент “Земля” необхідно вибрати режим Terminals Mode.

Для того щоб завантажити програму в мікроконтролер необхідно:

Вибрати пункт меню Sourse -> Add/Remove Source Code Files.

Вибрати компілятор AVRASM2.

Додати код програми за допомогою клавіші New.

Вибрати пункт меню Sourse -> Build All.

Якщо не було знайдено помилок, то перейти до перевірки схеми.

Для відладки схеми використовується пункт меню Debug.

Хід роботи:

1. Переглянути відеоролик https://www.youtube.com/watch?v=aw78G-daPv0
2. Використовуючи пакет моделювання електронних схем Proteus, створити схему, запропоновану викладачем.
3. Запрограмувати мікроконтролер для керування схемою.
4. Перевірити роботу двотонального генератора з використанням оптичного волокна та фотоприймача.
5. Між виходом оптоволоконної лінії та фотоприймачем послідовно встановити світлофільтр і дифракційну гратку.
6. Провести експеримент, зафіксувати результати та проаналізувати вплив світлофільтра та дифракційної гратки на сигнал, що надходить на фотоприймач.
7. За необхідності внести корективи в програмне забезпечення або апаратну частину для оптимізації роботи системи.

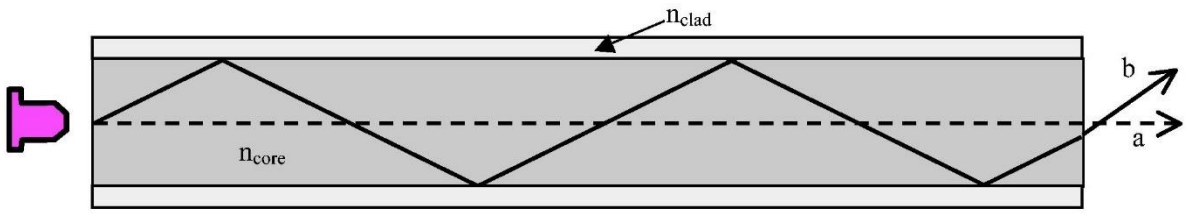


Рис. 10. Волоконно-оптичний кабель

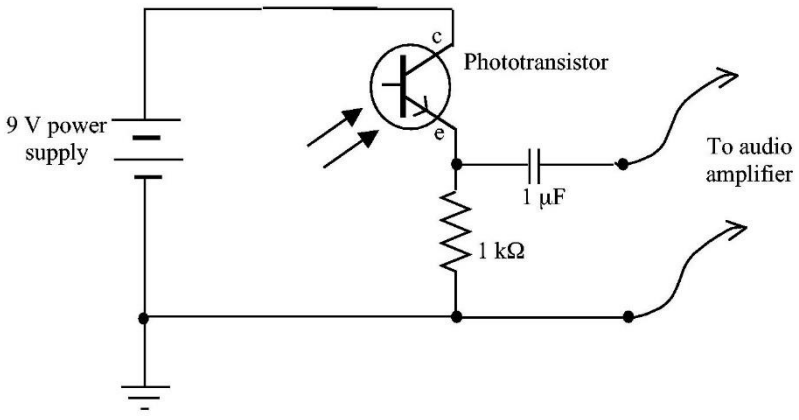


Рис. 11. Схема IR-фотоприймача

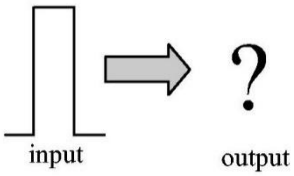


Рис. 12. Дослідження вихідного сигналу при подачі світлового імпульсу на оптичне волокно

Використання мікроконтролера дає змогу реалізувати більшість режимів роботи лабораторної установки програмним чином. Програмний підхід дає можливість зробити електронну схему більш гнучкою, легкою для повторного використання, швидкої модифікації та оптимізації. Це сприяє ефективному застосуванню обладнання в групах під час обговорення оптичних явищ, адже для внесення змін достатньо лише оновити програмний код.

Мікроконтролер програмується таким чином, що спочатку по черзі засвічуються окремі кристали RGB світлодіода (наприклад, червоний і зелений), а потім їхні комбінації (одночасне свічення червоного та зеленого). Перемикання режимів відбувається послідовним натисканням кнопки.

Програмування мікроконтролера

Приклад роботи з портом: (див. Рис. 2, в ATtiny13 всього 6 портів).

```
PORTB=0b00000000;  //Підтягуємо  всі 8 виводів порту B до лог. 0
DDRB=0b00001111; // Конфігуруємо PB0-PB4 на вихід
//а PB5 - PB8 на вхід
//відраховуємо СПРАВА-НАЛІВО
```

Наприклад, команда PORTB = 0b00000010 встановлює вивід PB1 у логічний рівень «1». Це відповідає 6 виводу мікросхеми. Варто зазначити, що порти PB6 і PB7 у мікроконтролері ATtiny13 відсутні — вони присутні лише в мікроконтролерах з більшою кількістю виводів.

PIN	**-**	**-**	**1**	**3**	**2**	**7**	**6**	**5**

	PORT B5	PORT B6	PORT B5	PORT B4	PORT B3	PORT B2	PORT B1	PORT B0
PORT	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
VALUE	0	0	0	0	0	0	1	0

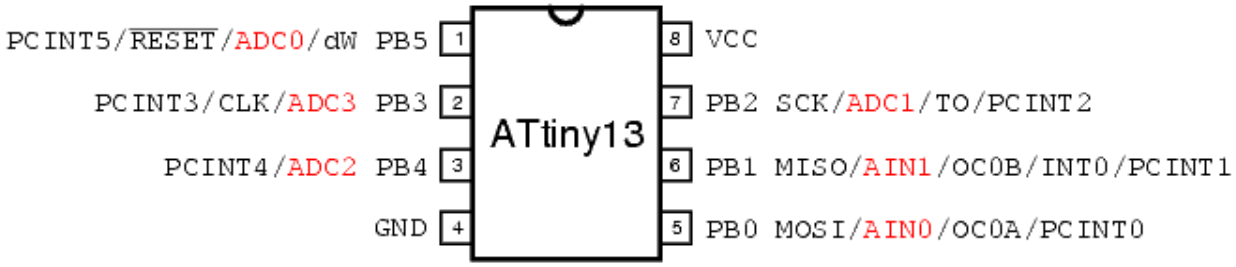


рис. 13. Значення виводів мікроконтролера ATtiny13

Текст програми на C++ для мікроконтролера, на якому зібрана описана демонстраційна схема керування світлодіодами.

```
/*
 * AVRGCC9.c
 *
 * Created: 15.11.2014 12:06:09
 * Author: Bohdan
 */
 //кнопка на PB1, вивід 6
// F_CPU = 9600000/8=1200000

#include <avr/io.h> // підмикання бібліотеки для роботи з регістрами
#include <util/delay.h>
#include <avr/interrupt.h>
```

```
      unsigned char mode = 0;
      unsigned char i = 0;

 #define VD_OFF PORTB = 0b00000010 // Вимикання світлодіодів
      #define BLUE PORTB = 0b00000011 // Вмикання BLUE світло діода
      #define GREEN PORTB = 0b00010010 // Вмикання GREEN світлодіода
      #define RED PORTB = 0b00000110 // Вмикання RED світлодіода
      #define YELLOW PORTB = 0b00010110 // Вмикання YELLOW світлодіода
      #define MAGENTA PORTB = 0b00000111 // Вмикання MAGENTA світлодіода
      #define CYAN PORTB = 0b00010011 // Вмикання CYAN світлодіода
      #define WHITE PORTB = 0b00010111 // Вмикання RGB світлодіода

      ISR(INT0_vect) // SIGNAL(SIG_INTERRUPT0)

      {
            _delay_ms(50); // Затримка 20 мілісекунд
            if (bit_is_clear(PINB,PB1)) // Опитування PB1
            //PORTB ^= 0x01;
            mode++;
      }

 int main(void)
 {
 DDRB = 0b00010101; //конфігурування портів і/о
       GIMSK = 0b01000000; // Дозвіл переривань INT0 на вході PB1
       MCUCR = 0b00000010; // при перепаді з 1 в 0
sei(); //загальний дозвіл переривань

 while (1)
 {
      WHITE;
for(;;){
             switch (mode) {
            case 1:{WHITE; break;}
            case 2:{RED; _delay_ms(2000); GREEN;_delay_ms(2000);
BLUE;_delay_ms(2000);WHITE;_delay_ms(2000);VD_OFF;_delay_ms(2000);break;}
            case 3:{RED; _delay_ms(2000); GREEN;_delay_ms(2000);
YELLOW;_delay_ms(2000);VD_OFF;_delay_ms(2000); break;}
            case 4:{ RED; _delay_ms(2000); BLUE;_delay_ms(2000);
MAGENTA;_delay_ms(2000);VD_OFF;_delay_ms(2000); break;}
```

```
            case 5:{ GREEN; _delay_ms(2000); BLUE;_delay_ms(2000);
CYAN;_delay_ms(2000);VD_OFF;_delay_ms(2000);break;}
             case 6:
                                                    {


      { for(i=1; i<200; i++) { RED;_delay_ms(2.3);VD_OFF; _delay_ms(2.3);} } //
мерехтіння 261 Hz
 { for(i=1; i<200; i++) { BLUE;_delay_ms(3);VD_OFF;_delay_ms(3); } } //
мерехтіння 196 Hz


                  break;}


case 7:{ BLUE; break;} // blue colour - last mode


                          default: mode=1;
            }
       }
 }
 }
```

Запитання для закріплення матеріалу лабораторної роботи:

1. Яка роль двотонального генератора в мультиплексуванні сигналів?
2. Які основні етапи створення та налагодження схеми з мікроконтролером у Proteus?
3. Як відбувається інтеграція програмного коду з апаратною частиною у лабораторній роботі?
4. Які особливості програмування мікроконтролерів AVR необхідно враховувати при розробці функціональних вузлів?
5. Які переваги дає використання програмного підходу для управління роботою електронних систем?
6. Як у Microchip Studio організовується робота з портами вводу-виводу мікроконтролера?

ЛАБОРАТОРНА РОБОТА №4.

Дослідження функцій інтерфейсу UART для дистанційного керування мікроконтролерними системами в лабораторних роботах

Ця лабораторна робота є логічним продовженням лабораторної роботи №3, однак основна увага в ній зосереджена на реалізації керування мікроконтролерною системою за допомогою інтерфейсу UART. Завдяки використанню послідовного обміну даними з ПК або іншим пристроєм, досліджується можливість дистанційного керування параметрами роботи електронного модуля, що є актуальним для розробки сучасних навчальних і дослідницьких систем.

Тема: Розробка системи керування RGB світлодіодом для Генерування сигналів в оптичних лініях зв’язку на базі мікроконтролера ATtiny13 з інтерфейсом USB-UART.

Мета: розробити систему керування RGB світлодіодом за допомогою мікроконтролера ATtiny13, використовуючи інтерфейс USB-UART для передачі команд від зовнішнього пристрою. Досягти стабільної та надійної взаємодії між комп’ютером і мікроконтролером, забезпечити ефективне управління кольоровою індикацією шляхом формування відповідних сигналів керування

Теоретичні відомості

RS-232 — це інтерфейс передачі даних між двома пристроями на відстані до 20 м. Інформація передається по провідниках із рівнями сигналів, які відрізняються від стандартних TTL 5 В, що забезпечує підвищену стійкість до електромагнітних перешкод. Асинхронна передача даних здійснюється з фіксованою швидкістю та синхронізується за допомогою стартового імпульсу.

Рівні сигналів

У стандарті RS-232 використовуються два рівні логічних сигналів: логічна "1" (MARK) і логічний "0" (SPACE). Логічна "1" кодується негативним рівнем напруги, а логічний "0" — позитивним. Конкретні значення напруг наведені в таблиці 1.

Таблиця 1. Рівні сигналів RS-232.

Рівень	Передавач	Приймач
“1”	+3В..+15В	+3В..+25В
“0”	-5В..-15В	-3В..-25В
Не визначений	-3В..+3В	

Конвертування логічних рівнів

Більшість мікроконтролерів використовують логічні рівні TTL/CMOS (0–5 В або 0–3.3 В) для обміну даними з іншими пристроями. Однак, коли необхідно організувати зв’язок з пристроями, які працюють за стандартом RS-232, виникає проблема несумісності рівнів сигналів: RS-232 використовує значно вищі і інвертовані напруги (приблизно від -12 В до +12 В), які не сумісні з TTL.

Для вирішення цієї проблеми застосовують спеціальний конвертер рівнів, наприклад, мікросхему MAX232. Цей пристрій перетворює сигнали TTL/CMOS мікроконтролера у відповідні сигнали RS-232 та навпаки, забезпечуючи надійний двонаправлений зв’язок між мікроконтролером і RS-232 сумісним пристроєм.

Для правильного використання MAX232 необхідно дотримуватись певних рекомендацій, зокрема підключення зовнішніх конденсаторів для внутрішніх перетворювачів напруги, що забезпечують живлення і формування необхідних рівнів сигналів.

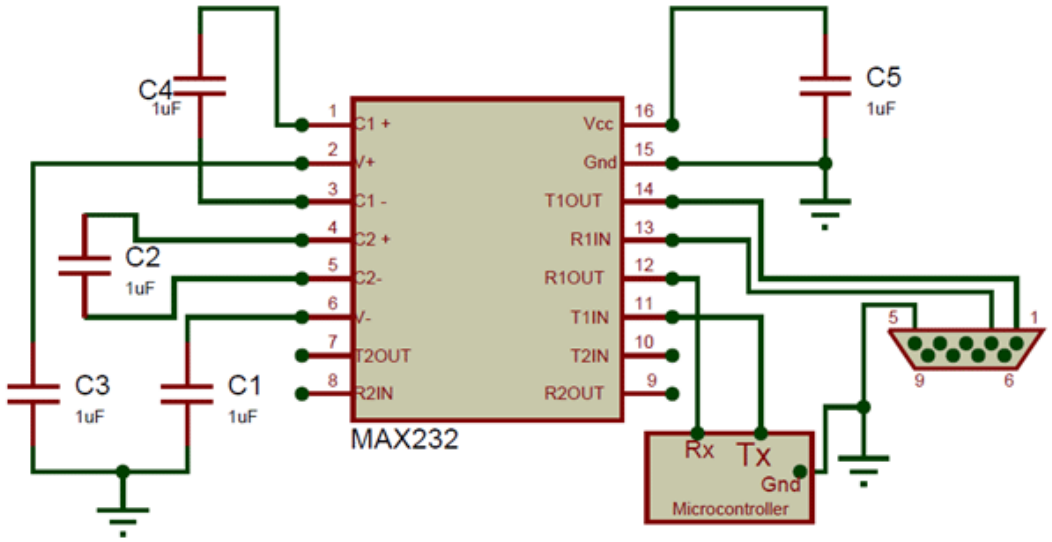


Рис. 1. Конвертер логічних рівнів на мікросхемі MAX232.

MAX232 має два канали передачі даних: R1 (вхід/вихід), T1 (вхід/вихід), а також R2 (вхід/вихід) і T2 (вхід/вихід). У цьому випадку буде використовуватися лише пара R1 і T1. Для R2 і T2 процедура підключення аналогічна у разі потреби їх використання. Спочатку до виводів T1 OUT і R1 IN підключається пристрій RS-232, потім до виводів R1 OUT і T1 IN — пристрій TTL/CMOS. Далі підключається конденсатор ємністю 1 мкФ між виводами C1+ та C1-, а також між C2+ та C2-. Слід пам’ятати, що при використанні лише каналів R1 і T1 необхідний лише конденсатор C1. Після підключення конденсаторів C1 і C2 підключаються ще два конденсатори до виводів Vs- і Vs+.

Тепер можна розпочинати передачу даних. Варто зауважити, що коли пристрій TTL/CMOS передає дані з рівнем 5 В, пристрій RS-232 отримує сигнал з напругою до +12 В. Таке перетворення напруги відбувається завдяки конденсаторам, підключеним до виводів Vs+ та Vs-, які допомагають мікросхемі MAX232 підвищувати напругу з +5 В до +12 В. Цей процес перетворення напруги через конденсатори називається зарядним насосом (charge pump). Необхідно подати живлення Vcc до +5 В і забезпечити спільне заземлення (GND) з мікросхемою MAX232.

USB забезпечує обмін даними між хост-комп’ютером і численними одночасно підключеними периферійними пристроями. Розподіл пропускної здатності шини між підключеними пристроями керується хостом і реалізується шляхом передачі маркерних пакетів (token packets). Шина підтримує динамічне підключення та відключення пристроїв під час роботи як хоста, так і самих периферійних пристроїв, що дає можливість їх конфігурувати та використовувати без перезавантаження системи.

Рис. 2. Розташування виводів USB.

Таблиця 1. Розташування виводів USB.

Вивід	Назва	
1	VCC	+5В
2	D-	Дані -
3	D+	Дані +
4	GND	Заземлення

UART (універсальний асинхронний приймач/передавач) — це тип асинхронного послідовного інтерфейсу, який забезпечує передачу даних між паралельними та послідовними інтерфейсами. UART використовується у комп'ютерах і периферійних пристроях для обміну інформацією та часто застосовується разом із комунікаційними стандартами, такими як EIA RS-232. UART зазвичай реалізується як окрема мікросхема або інтегрована частина мікроконтролера, що забезпечує інтерфейс для послідовного зв'язку через комп'ютерний чи периферійний послідовний порт. Сучасні мікроконтролери, як правило, мають вбудовані UART-модулі.

Для підключення UART до комп'ютера через USB використовуються USB-TTL конвертери, наприклад, PL2303HX, CH340, FT232RL, CP2102 USB-UART.

Підтримка режимів живлення:

5,0 В від зовнішнього джерела живлення,

3,3 В від зовнішнього джерела живлення.

Має світлодіоди: TXD LED, RXD LED та POWER LED.

Типове застосування: забезпечення комунікації між мікроконтролером і ПК за допомогою UART-протоколу через USB-інтерфейс.

В лабораторній роботі використовується є 4-канальний контролером RGB світлодіода з управлінням через UART інтерфейс. У пристрої реалізовано 2 канали широтно-імпульсної модуляції для керування яскравістю світлодіода та 2 звичайні канали для підключення додаткових елементів. Управління пристроєм здійснюється за допомогою команд, що надсилаються з іншого пристрою через UART інтерфейс (наприклад, з комп'ютера або

мікроконтролера). Контролер побудований на базі мікроконтролера ATtiny13 з обсягом flash пам’яті 1 КБ. UART і світлодіоди можуть ефективно використовуватися для дистанційного керування та вимкнення приладів у дистанційних лабораторних роботах. Через UART можна відправляти команди на ввімкнення або вимкнення пристроїв.

Алгоритм виконання дистанційної роботи:

1. Підготовка до виконання роботи

a. Ознайомлення з інструкцією по виконанню лаб. роботи.
b. Віддалені ввімкнення ПК, в лаб корпусі університету, запуск веб камери.
c. Віддалені ввімкнення вимірювальної установки

2. Проведення експерименту

a. Запуск програми відображення осцилографа
b. Програмування модулятора сигнала
c. Збір данних з зразків (Проведення експерименту)

3. Вимикання лабораторного обладнання та ПК.
4. Обробка зібраної інформації
5. Оформлення звіту з виконаної лаб. роботи.

Для ілюстрації концепції дистанційних лабораторних робіт розглянемо приклади. Перша дистанційна лабораторна робота присвячена вивченню нерівноважних носіїв заряду у напівпровідниках. Вибір цієї роботи обґрунтований низкою критеріїв:

1. Міждисциплінарність — робота підходить для студентів різних напрямків підготовки.
2. Можливість наочної візуалізації складних фізичних явищ.
3. Мінімалістичний та модульний дизайн цифрових приладів, що дає можливість дистанційне керування.
4. Гнучкість у реалізації різних завдань для студентів різних спеціальностей.
5. Відсутність механічних частин, що підвищує надійність функціонування системи.

Лабораторна робота базується на дослідницькій установці та є прикладом ефективної реалізації концепції віддаленого виконання лабораторних занять у природничих та технічних науках. Віддалений доступ до обладнання здійснюється за допомогою стандартних інтерфейсів і комп’ютера, підключеного до мережі Інтернет.

Блок-схема дистанційної лабораторної роботи наведена на рис. 3.

Для дослідження використовується метод експоненти, який полягає у безпосередній реєстрації кінетики зміни фотоструму через напівпровідниковий зразок під впливом П-подібних світлових імпульсів.

Особливістю роботи є використання додаткової лінійки малоінерційних RGB-світлодіодів, керованих мікроконтролером через UART інтерфейс. Це дає можливість в широких межах керувати спектром підсвітки шляхом зміни струму кожного світлодіода та регулювати колірний тон випромінюваного ними білого світла безпосередньо в процесі роботи і досліджувати спектральні залежності.

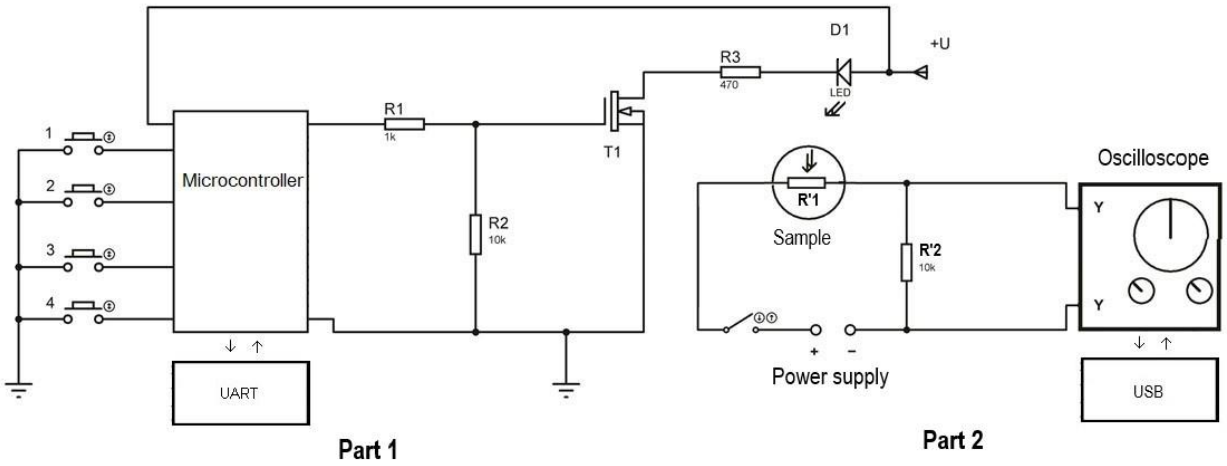


Рис. 3. Схема керованої лабораторної установки.

У системах керування та передачі даних важливо забезпечити електричну ізоляцію між логічними схемами управління (наприклад, мікроконтролером) та силовими або завадостійкими елементами. Для цього широко застосовуються оптронні розв’язки, що дозволяють безпечно передавати сигнали між частинами пристрою з різними потенціалами. Оптопара — це електронний компонент, що складається зі світлодіода та фоточутливого елемента фототранзистора, фотодіода або фототиристора, розміщених в одному корпусі. Світлодіод перетворює електричний сигнал на світло, яке приймається фотоприймачем, що в свою чергу знову перетворює його на електричний сигнал. Завдяки цьому здійснюється гальванічна розв’язка, тобто повна електрична ізоляція між входом і виходом оптрона. Основні функції оптронної розв’язки полягають у забезпеченні гальванічної ізоляції між частинами схеми, що запобігає небажаному протіканню струмів; захисті мікроконтролера від стрибків напруги та коротких замикань у навантаженні; підвищенні завадостійкості за рахунок світлової передачі сигналу, яка не чутлива до електромагнітних наведень; а також у забезпеченні уніфікації інтерфейсів, що дає можливість з’єднувати схеми з різними рівнями живлення.

В даній лабораторній роботі оптрон використовується для ізоляції ліній передачі даних між UART-адаптером і мікроконтролером. Це підвищує безпеку під час експериментів, особливо за умов дистанційного керування приладом через комп’ютер або інтернет-мережу.

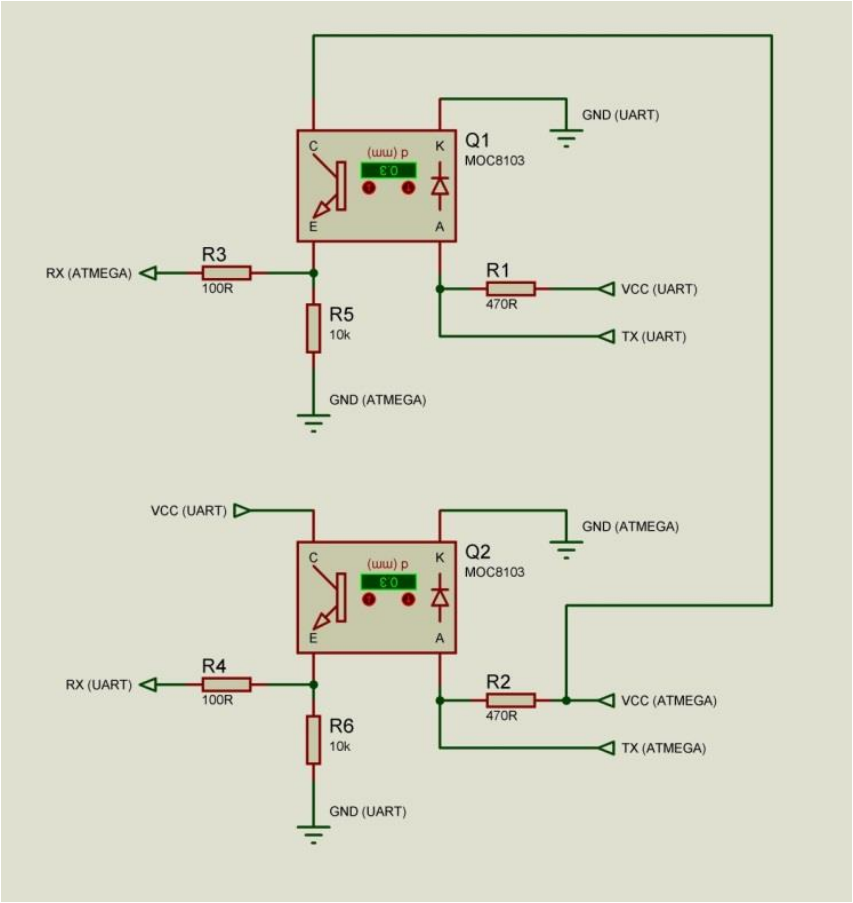


Рис 4. Оптична розв’язка цифрового інтерфейсу.

Іншим прикладом пристрою для дистанційного керування є кероване регульоване джерело живлення. Такі джерела є одними з найпоширеніших у лабораторній практиці, оскільки їх легко реалізувати з використанням мікроконтролера ATmega8, простого послідовного інтерфейсу RS-232 або UART. Завдяки інтерфейсу UART забезпечується обмін даними між ПК і пристроєм у реальному часі, що дає змогу дистанційно змінювати вихідну напругу та контролювати режим роботи.

У цьому процесі використання мікроконтролера, в якому цифрові компоненти збираються в одному блоці та реалізуються програмним чином, також є фактором, що зменшує витрати на виготовлення та відлагодження схеми. Основним компонентом проектування електронної схеми, що становить експериментальну частину, є мікроконтролер.

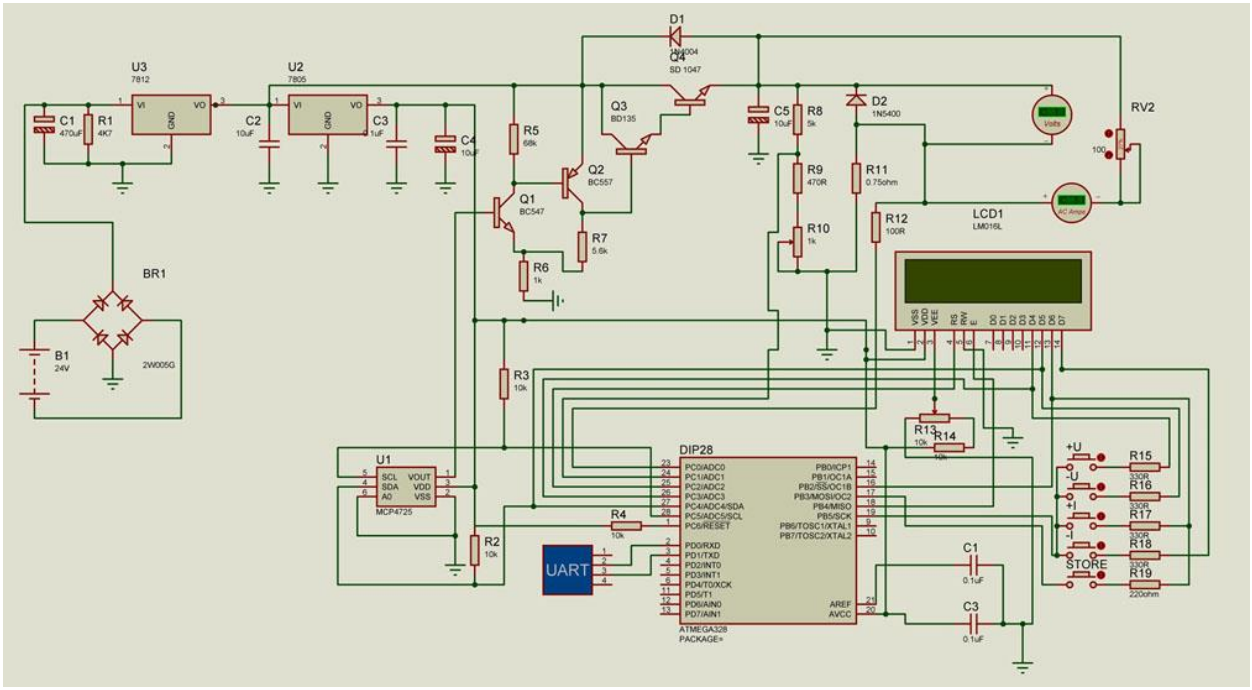


Рис. 5. Кероване джерело живлення в системі автоматизованого проектування Proteus Design Suite.

Програмне забезпечення, що визначає функціональність мікроконтролера в цій схемі, розроблено за модульним принципом. Такий підхід забезпечує відокремлення основного коду, що відповідає за організацію інформаційного потоку, від допоміжних модулів, які підтримують безперервність цього потоку.

Програмна частина мікроконтролера складається з основного циклу та двох ключових підпрограм. Модульна структура також охоплює інші елементи програмного забезпечення, що реалізують функції керування. Зв’язок мікроконтролера з іншими функціональними блоками — дисплеєм та енкодером, які формують інтерфейс користувача — здійснюється через відповідні програмні компоненти.

Загальна логіка функціонування визначається набором типових сценаріїв використання. На рисунку 6 представлено блок-схему алгоритму роботи програми керування джерелом живлення. Після ввімкнення пристрою запускається основний цикл, у межах якого активуються дисплей, енкодер та цифро-аналоговий перетворювач. Якщо в енергонезалежній пам’яті збережено попередні значення струму та напруги, вони зчитуються та подаються на вихід. Після завершення цієї триетапної процедури система переходить у режим готовності. У подальшому пристрій обробляє команди, які надходять від енкодера або з комп’ютера.

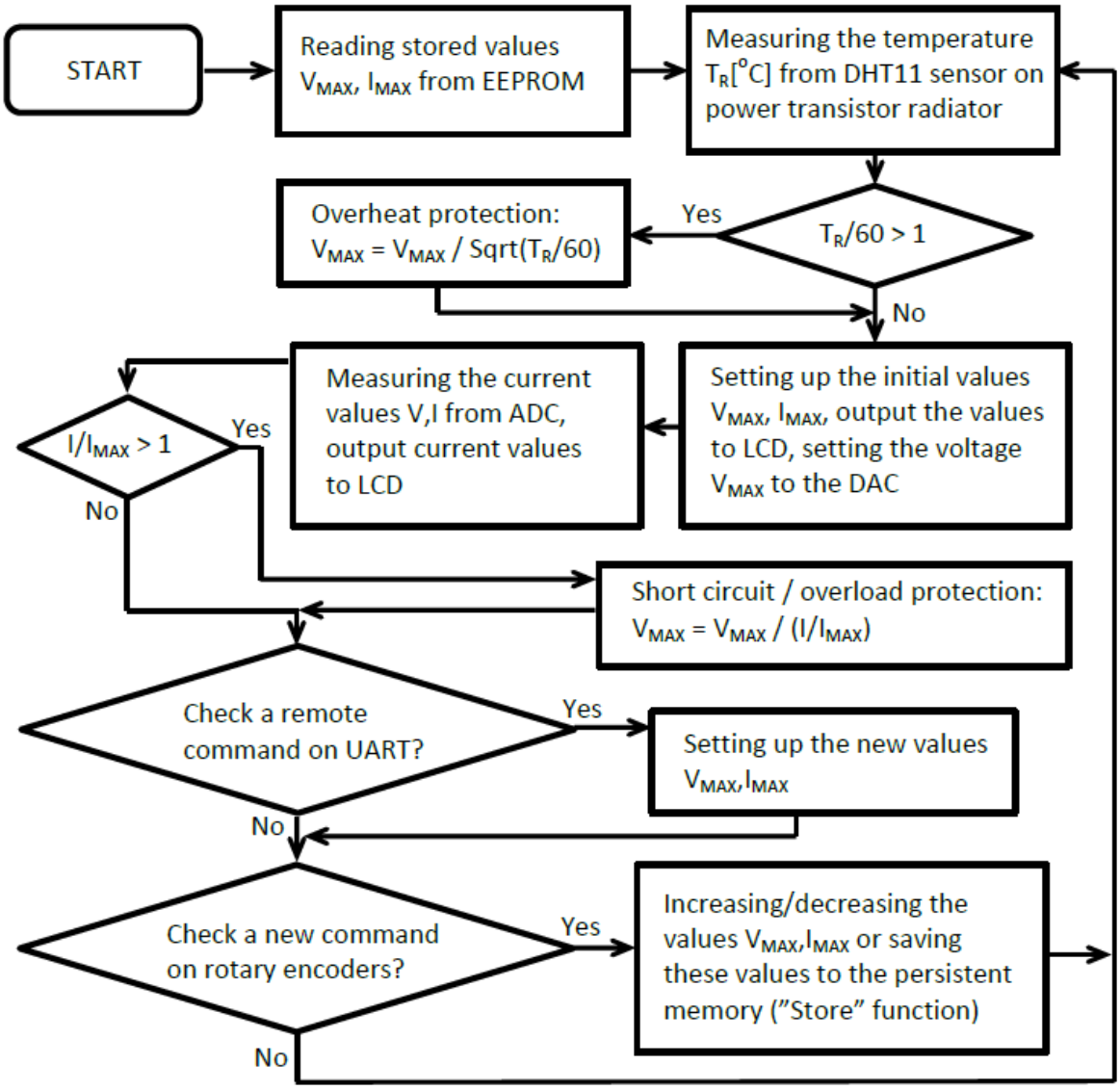


Рис. 6. Алгоритм роботи програмного забезпечення.

Кожна взаємодія користувача з системою оцінюється іншим компонентом і спричиняє зміни в системі. Це може бути або зміною струму відповідно до заданої команди, або активацією захисту від короткого замикання при підключеному великому навантаженні. Більш детальну інформацію про поточний стан навантаження можна отримати з пристрою за допомогою програмного забезпечення управління по запитів до контролера через UART. Найважливішою особливістю такого розробленого програмного забезпечення є більш широка взаємодія з користувачем, а також повне виконання функції автоматичного керування. Команди, що зберігаються в черзі та структурі даних стека, надсилаються в мікроконтролер в порядку, а результати обробляються у фоновому режимі. Це дає можливість користувачеві видавати команди послідовно, не чекаючи результатів команд, надісланих на пристрій.

Хід роботи:

Під’єднати демонстраційну схему до USB-UART адаптера

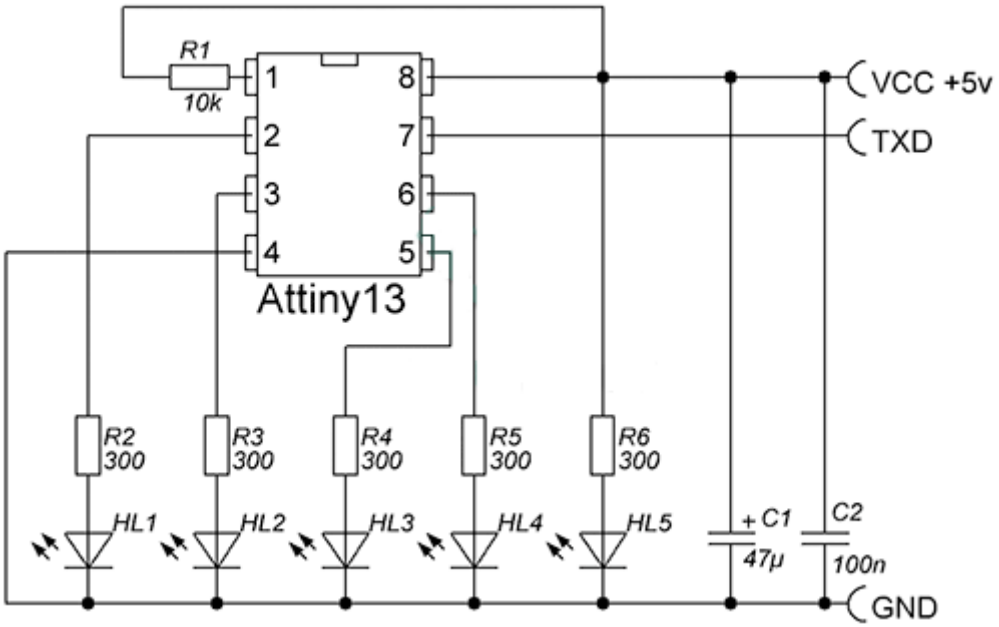


Рис. 3. Демонстраційна схема.

.

Схема має три виводи: GND, TXD, 5.0 V – VCC.

Необхідно під’єднати відповідні виходи схеми до відповідних контактів USB-UART адаптера. Після підключення адаптера до комп’ютера буде автоматично розпізнано COM-порт з присвоєним номером.

Світлодіоди HL1, HL2, HL3 та HL4 індикують стан роботи відповідних каналів. Світлодіод HL5 є індикатором подачі живлення на пристрій. Конденсатори C1 та C2 забезпечують стабілізацію напруги живлення. Пристрій потребує живлення в діапазоні від 3 до 5 Вольт.

Для забезпечення коректної роботи пристрою та налаштування функціоналу мікроконтролера необхідне його програмування. Найбільш зручним режимом програмування є послідовний SPI-інтерфейс. Для реалізації цього режиму слід підключити мікроконтролер до програматора за допомогою ліній SPI (MOSI, MISO, SCK, RESET, GND) та забезпечити подачу номінальної напруги живлення. Програмування може виконуватися безпосередньо в робочій схемі (внутрішньосхемне програмування), проте важливо виключити перешкоди на лініях SPI, такі як значні ємності або низькі опори відносно спільного провідника.

Для керування пристроєм необхідно надсилати відповідні команди у вигляді рядків через послідовний інтерфейс (наприклад, за допомогою програми HyperTerminal https://www.hilgraeve.com/hyperterminal-trial/):

Port1=1 — активація каналу 1;

Port1=0 — деактивація каналу 1;

Port2=1 — активація каналу 2;

Port2=0 — деактивація каналу 2;

Port3=1 — активація каналу 3;

Port3=0 — деактивація каналу 3;

Port4=1 — активація каналу 4;

Port4=0 — деактивація каналу 4;

AllOn — активація всіх каналів;

AllOff — деактивація всіх каналів.

Port1PWM - Встановити значення ШІМ 1 каналу;

Port2PWM - Встановити значення ШІМ 2 каналу.

Завдання:

1. Розробити програмний код для мікроконтролера, що забезпечує керування багатоканальною світлодіодною системою через інтерфейс UART. Система повинна демонструвати принцип мультиплексування сигналів в оптоволоконних лініях, зокрема можливість послідовного керування кількома каналами світловипромінювання. Передача команд повинна здійснюватися у вигляді текстових інструкцій з комп’ютера або іншого UART-сумісного пристрою.
2. Між виходом оптоволоконної лінії та фотоприймачем послідовно встановити світлофільтр і дифракційну гратку.
3. Провести експеримент, зафіксувати результати та проаналізувати вплив світлофільтра та дифракційної гратки на сигнал, що надходить на фотоприймач.
4. За необхідності внести корективи в програмне забезпечення або апаратну частину для оптимізації роботи системи.

Запитання для закріплення матеріалу лабораторної роботи:

1. Що таке інтерфейс UART і які його основні характеристики?
2. У чому полягає принцип мультиплексування сигналів у оптоволоконних лініях?
3. Яким чином можна реалізувати багатоканальне керування світлодіодами за допомогою UART?
4. Яку функцію виконує мікроконтролер у розробленій системі керування?
5. Яким чином відбувається інтерфейсна розв’язка між мікроконтролером і зовнішнім пристроєм керування?
6. Як можна модифікувати систему для керування більшою кількістю каналів?

ЛАБОРАТОРНА РОБОТА №5.

Вивчення особливостей взаємодії платформи Arduino з периферійними пристроями на прикладі реалізації дешифратора оптоелектронної системи або приймача команд протоколу RC5.

Тема: Взаємодія платформи Arduino з периферією: реалізація дешифратора оптоелектронної системи та приймача RC5.

Мета: вивчити особливості взаємодії платформи Arduino з периферійними пристроями на прикладі реалізації дешифратора оптоелектронної системи та приймача команд протоколу RC5. Написати програму отримання сканкодів з пульту дистанційного керування.

Теоретичні відомості

Arduino — це апаратно-програмна платформа для розробки вбудованих систем, що забезпечує інтеграцію електронних компонентів в єдиний функціональний пристрій. Поведінку такого пристрою можна програмувати за допомогою персонального комп'ютера. Платформа Arduino виконує роль контролера, який керує роботою датчиків, виконавчих механізмів та інших елементів системи. Вона підходить як для створення простих електронних схем, так і для реалізації складніших проєктів, зокрема автоматизованих вимірювальних систем. Основу платформи становить апаратна частина — плата введення-виведення з мікроконтролером — та програмна частина, представлена інтегрованим середовищем розробки Arduino IDE, що дає можливість писати, компілювати та завантажувати програмний код на плату.

За допомогою Arduino можна створювати різноманітні інтерактивні пристрої й вимірювальні прилади, обробляти сигнали з датчиків і перемикачів, а також керувати виконавчими елементами, зокрема двигунами, світлодіодами, реле тощо. Серед основних переваг використання Arduino — кросплатформність програмного забезпечення, велика кількість бібліотек для роботи з периферією, а також відкритість апаратного дизайну й програмного коду. Особливістю мікроконтролерних плат сімейства Arduino є можливість завантаження програм, написаних мовами високого рівня C/C++, безпосередньо через USB-інтерфейс, без потреби у спеціалізованих програматорах. Платформа Arduino здобула широку популярність завдяки простоті у використанні, доступності, а також зрозумілій і гнучкій мові програмування. Програми, створені в середовищі Arduino IDE, називаються скетчами.

Після створення програми вона завантажується в мікроконтролер на платі Arduino. Для цього використовується USB-кабель, який також може слугувати

джерелом живлення на етапі програмування. Одразу після завантаження скетчу мікроконтролер починає виконувати задані команди.

В Arduino IDE використовуються дві основні функції:

setup() — виконується один раз після увімкнення живлення або перезавантаження пристрою; застосовується для ініціалізації налаштувань (наприклад, встановлення режимів роботи портів введення/виведення).

loop() — виконується безперервно після завершення setup(); реалізує основну логіку роботи пристрою в циклі.

Мікроконтролер ATmega328, що використовується в більшості плат Arduino може виконувати тіло функції loop() до 10 000 разів на секунду, залежно від обсягу коду та швидкодії.

До базових функцій програмування в Arduino належать:

pinMode(pin, mode) — встановлює режим роботи виводу (введення або виведення);

digitalRead(pin) — зчитує логічне значення з цифрового входу;

digitalWrite(pin, value) — записує логічне значення на цифровий вихід;

analogRead(pin) — зчитує аналогову напругу з відповідного входу (задається як значення від 0 до 1023);

analogWrite(pin, value) — задає ШІМ-сигнал (PWM) на відповідний вивід, що імітує аналогову напругу;

Serial.print(data) — виводить дані в серійний порт у текстовому вигляді для відлагодження або моніторингу.

Для роботи з різними платами, модулями та датчиками Arduino підтримує велику кількість бібліотек готових функцій і класів, що значно спрощують керування додатковими пристроями.

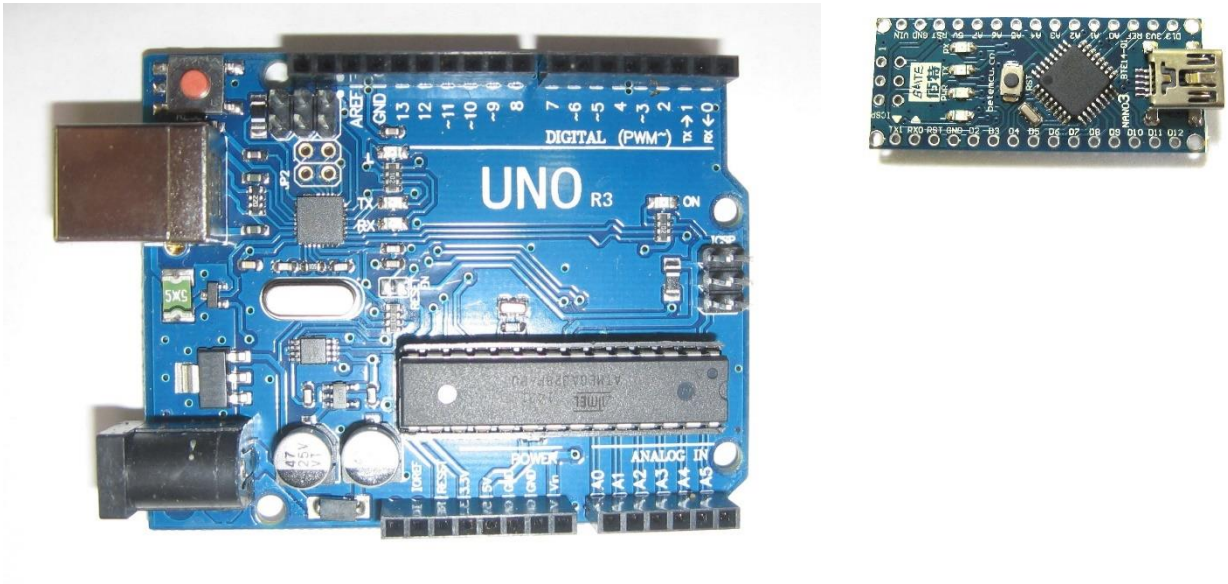


Рис. 2. Arduino UNO та Arduino NANO

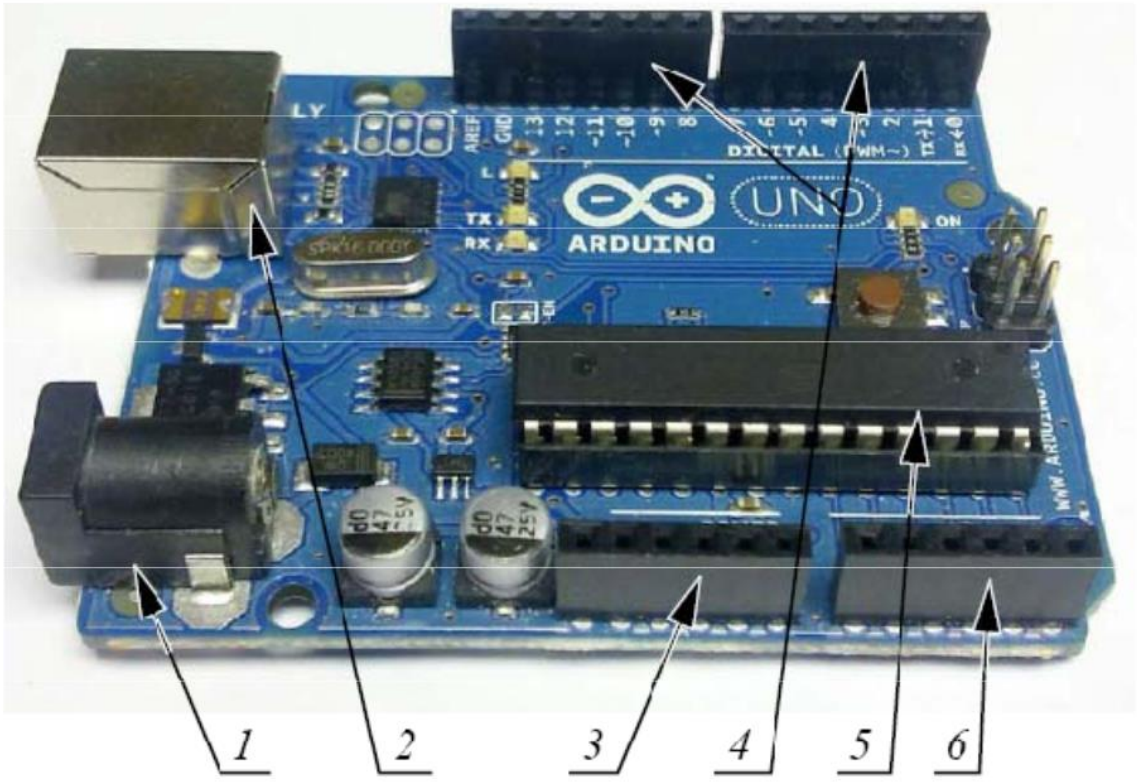


Рис. 3 Платформа Arduino UNO.

1–роз’єм зовнішнього живлення,

2 – роз’єм USB,

3 – роз’єм живлення ,

4 – роз’єм цифрових виходів,

5 – мікроконтролер ATMega 328P,

6 – роз’єм аналогових входів

Протокол RC5 є поширеним протоколом кодування інфрачервоних сигналів, розроблений компанією Philips у 1980-х роках для дистанційного керування побутовою електронікою. Він забезпечує надійну та ефективну передачу команд від пульта дистанційного керування до фотоприймача.

Основні характеристики протоколу RC5:

Тип модуляції:

Протокол RC5 використовує фазову маніпуляцію (Manchester encoding), яка забезпечує високий рівень захисту від помилок і полегшує синхронізацію між передавачем і приймачем. При фазовій маніпуляції кожен біт кодується двома рівнями сигналу — зміна фази посередині бітового інтервалу означає зміну значення біта. Структура даних: RC5 передає 14-бітне повідомлення, яке складається з таких частин:

2 стартові біти (Start bits): завжди встановлені в логічну 1, сигналізують початок передачі.

1 біт перемикання (Toggle bit): змінюється при кожному натисканні кнопки на пульті, що дає можливість приймачу відрізнити повторні натискання від утримання кнопки.

5 біт адреси (Address bits): визначають тип або адресу приймача (наприклад, телевізор, аудіо-система та ін.). Це дає можливість одному пульту керувати кількома пристроями.

6 біт команди (Command bits): визначає конкретну дію (увімкнення, вимкнення, регулювання гучності тощо).

Частота несучої хвилі:

Інфрачервоний сигнал модуляції передається на частоті близько 36 кГц, що є стандартною частотою для ІЧ-пультів і дає можливість ефективно фільтрувати зовнішні завади.

Передача та прийом:

Протокол RC5 передбачає послідовну передачу бітів із строго визначеною тривалістю кожного біту дає можливість точно визначити початок і кінець передачі команди.

Обладнання:

Плати Arduino Uno та Arduino Nano, фотоприймач, джерело живлення, пульт керування, світлодіод.

Програмне забезпечення: Arduino IDE , бібліотека IRremote, приклад IrrecvDemo.

Технічні характеристики Arduino UNO.

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

IR – фотоприймач ILMS5360. IR-фотоприймач ILMS5360 — це високочутливий фотодіодний модуль, призначений для прийому інфрачервоного випромінювання в діапазоні близько 940 нм. Він широко використовується у

системах бездротового керування, дистанційного управління, а також у оптоелектронних пристроях для виявлення ІЧ-сигналів.

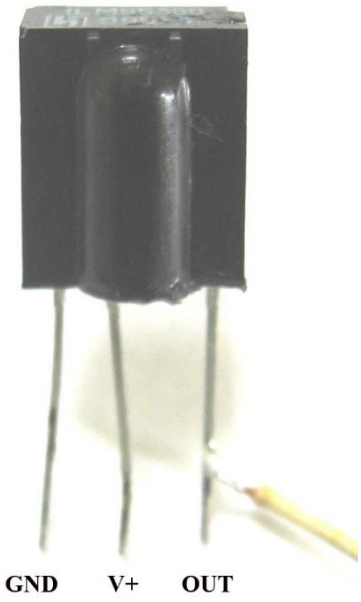


Рис. 1.

Світлодіод, пульт RC-5

.

Хід роботи:

1. Перевірити IR- фотоприймач . Для цього зібрати схему схемою -

- на 1 вивід, + на 2 вивід, – світлодіод на вихід приймача (3 вивід).

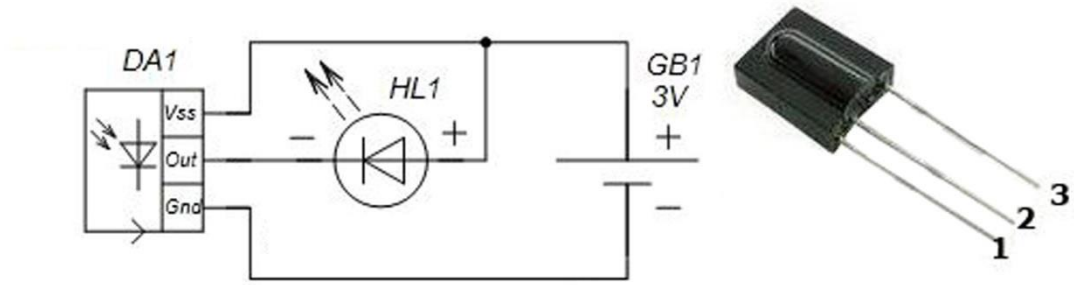


Рис. 4. Вмикання фотоприймача.

2. Підімкнути контролер Arduino до USB порта комп'ютера.

Перевірити на якому COM- порті знаходиться плата Arduino.

3. Встановити The open-source Arduino Software (IDE) з сайту https://www.arduino.cc/en/Main/Software

4. Встановити драйвери через (My Computer->Manage->Device Manager) з папки FTDI USB Drivers для плати UNO.

5. Завантажити бібліотеку **IRremote** для інфрчервоного пульта системи RC5 (IRremote.zip) та розархівувати її. Скопіювати в папку Libraries Arduino.

6. Запустити Arduino IDE, знайти бібліотеку в IR Remote в Arduino IDE. (рис. 5).

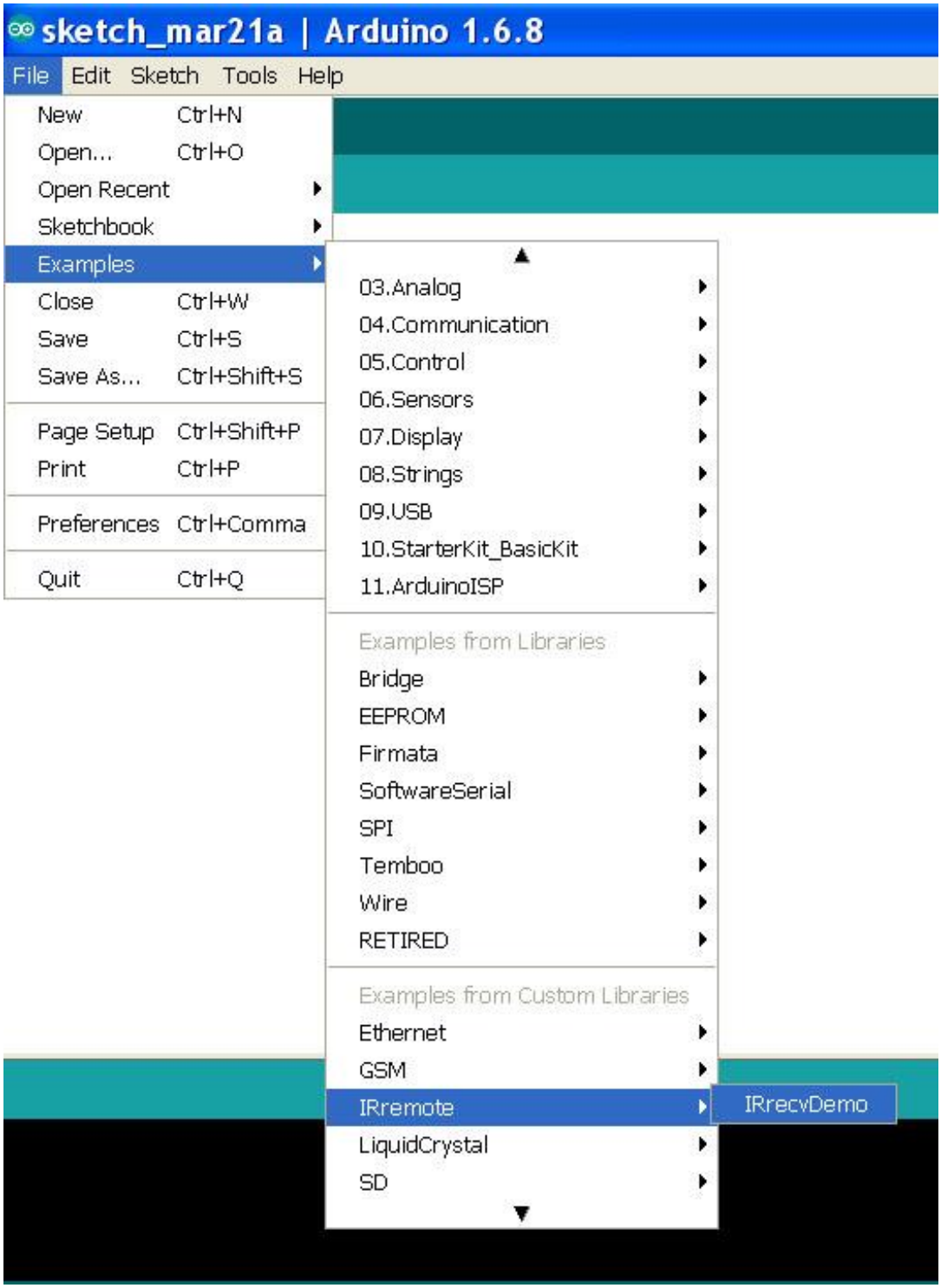


Рис. 5. Підмикання бібліотеки IRremote в середовищі ArduinoIDE.

Підімкнути IR- фотоприймач до плати Arduino за схемою. Див рис. 8.

5V

GND

OUT ->Pin11 Arduino.

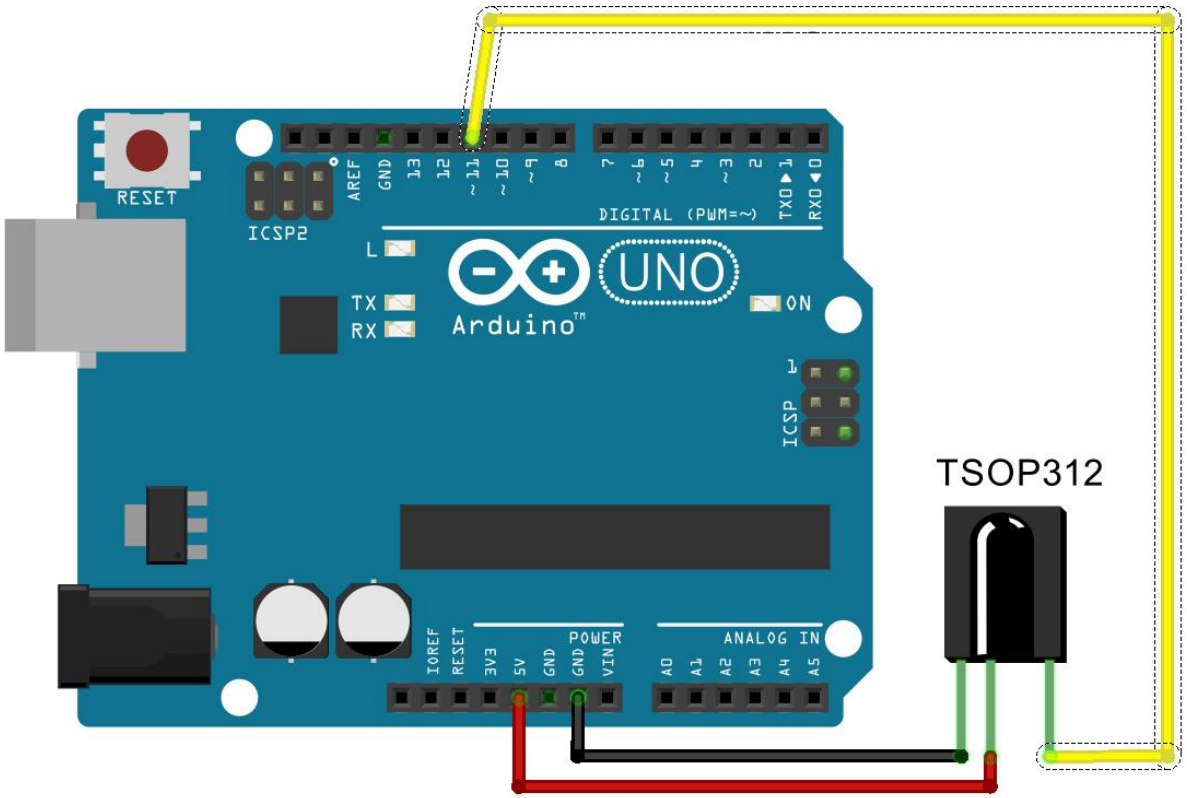


Рис. 6. Схема під'єднання IR- фотоприймача до плати Arduino UNO.

"Файл" -> "параметри" **Iremote. IrrecvDemo.**

1. Завантажити скетч arduino для визначення коду команди в мікроконтролер.

```
#include <IRremote.h>

int RECV_PIN = 11;

IRrecv irrecv(RECV_PIN);

decode_results results;

void setup()
{
 Serial.begin(9600);
 irrecv.enableIRIn(); // Start the receiver
```

```
}

void loop() {
 if (irrecv.decode(&results)) {
 Serial.println(results.value, HEX);
 irrecv.resume(); // Receive the next value
 }
 delay(100);
}
```

Запустити монітор порту. Вибрати Tools, Serial Monitor та перевірити коди дистанційних пультів.

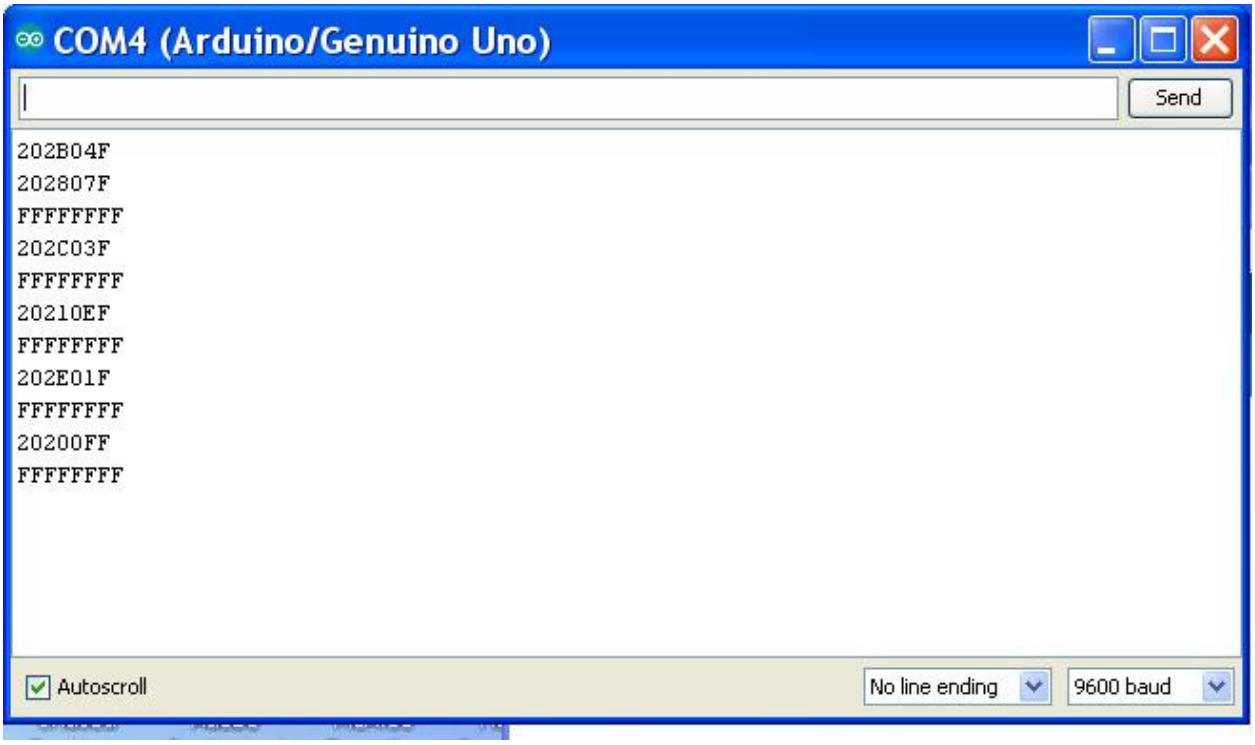


Рис. 7. Монітор порту зі scan-кодами команд дистанційних пультів за протоколом RC5.

Варіанти завдань:

Написати програму для керування світлодіодом за допомогою інфрачервоного пульта. — Кожна кнопка пульта має вмикати або вимикати певний світлодіод.

Реалізувати режим автоматичного вимикання світлодіода через заданий час після отримання команди з пульта.

Розробити програму, яка відрізняє повторне натискання однієї кнопки від одиничного натискання і виконує різні дії.

Виконати аналіз тимчасових характеристик сигналу пульта (тривалість імпульсів, інтервали між ними) за допомогою серійного монітора.

Сконфігурувати Arduino для прийому сигналів з кількох типів пультів (крім RC5), порівняти відмінності в кодуванні.

Запитання для закріплення матеріалу лабораторної роботи:

1. Для чого використовується бібліотека IRremote в Arduino IDE?
2. Яким чином можна визначити, на якому COM-порті знаходиться плата Arduino?
3. Що таке протокол RC5 і які його особливості?
4. Як вивести отриманий сигнал від пульта в серійний монітор Arduino IDE?
5. Як можна розширити функціональність проекту з IRremote для керування кількома пристроями?

ЛАБОРАТОРНА РОБОТА №6

Тема: Емуляція керування мікроконтролером ATtiny44 через інтерфейс RS-232.

Мета: Навчитися проектувати пристрої на основі мікроконтролера ATtiny44 у середовищі Proteus Design Suitета керувати ними за допомогою інтерфейсу RS-232.

Теоретичні відомості

1. Інтерфейс RS-232

RS-232 — стандарт інтерфейсу обміну даними між двома пристроями шляхом послідовної передачі даних (асинхронний зв'язок або синхронний зв'язок), знаходить використання у послідовних портах комп'ютерів та інших пристроях. Інтерфейс RS-232 призначений для підключення до комп'ютера стандартних зовнішніх пристроїв (принтера, сканера, модему, миші й ін.), а також для зв'язку комп'ютерів між собою (до 20м). Дані в інтерфейсі RS-232 передаються в послідовному коді (біт за бітом) побайтно. Кожен байт обрамляється стартовим і стоповими бітами. Дані можуть передаватися як в одну, так і в іншу сторону по різним проводам (дуплексний режим).

Інтерфейс RS-232 має наступні вимоги до логічних рівнів сигналу: логічні рівні передавача: "0" - від +5 до +15 Вольт, "1" - від -5 до -15 Вольт; логічні рівні приймача: "0" - вище +3 Вольт, "1" - нижче -3 Вольт (рисунок 1.1).

Стандарт не визначає швидкості передачі. Загальноприйнята швидкість передачі для RS-232 — 9600 біт/сек на відстань до 15 м. Сучасні пристрої підтримують швидкість 115 кбіт/сек та більше. Існує в 8-, 9-, 25- і 31-контактних варіантах роз'ємів. У даний момент найчастіше використовуються 9-контактні роз'єми:

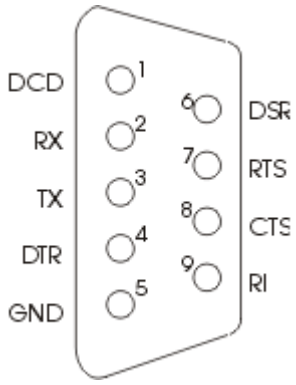


Рис. 1 – Контакти інтерейсу RS-232

Опис контактів 9-контактного роз'єму RS-232 наведено в таблиці:

Контакт	Позначення	Напрям	Опис
1	CD	<--	Carrier Detect. Детектування прийнятого сигналу.
2	RXD	<--	Receive Data. Прийом даних.
3	TXD	-->	Transmit Data. Передача даних
4	DTR	-->	Data Terminal Ready. Готовність вихідних даних.
5	GND	---	System Ground. Сигнальне заземлення.
6	DSR	<--	Data Set Ready. Готовність даних.
7	RTS	-->	Request to Send. Готовність до передачі.
8	CTS	<--	Clear to Send. Готовність до прийому.
9	RI	<--	Ring Indicator. Індикатор виклику.

Основні параметри інтерфейсу RS-232: швидкість обміну даними; кількість біт даних; контроль парності; довжина стоп-біта.

У протоколі RS-232 можуть використовуватися два методи управління обміном даними: апаратний та програмний. Також допускається робота без керування потоком, що вимагає постійної готовності хоста і периферійного пристрою до прийому даних.

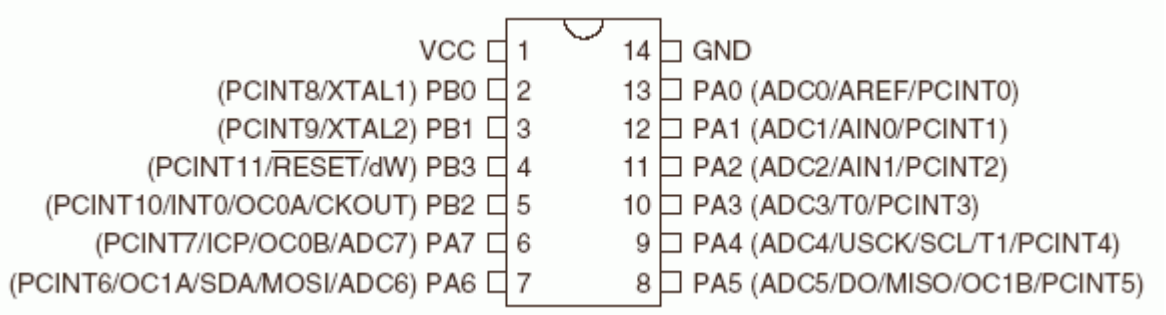
Апаратний метод управління реалізується за допомогою сигналів RTS і CTS. Перед передачею даних хост встановлює сигнал RTS, після чого очікує встановлення сигналу CTS з боку периферійного пристрою. Після цього хост розпочинає передачу даних до моменту існування сигналу CTS. По завершенню передачі даних периферійний пристрій прибирає сигнал CTS, а хост прибирає сигнал RTS.

Програмний метод управління базується на передачі приймачеві спеціальних символів зупинки (символ з кодом 0x13, який називається XOFF) та продовження (символ з кодом 0x11, що називається XON) передачі.

2. Мікроконтролер ATtiny44

ATtiny44 - 8-розрядний КМОН-мікроконтролер, виконаний на основі удосконаленої RISC-архітектури AVR. За рахунок виконання більшості інструкцій за один період синхронізації мікроконтролер ATtiny44 досягає продуктивності 1 млн. операцій в сек. при частоті задавального генератора 1 МГц, що дає можливість розробникам оптимізувати співвідношення продуктивності і споживаної потужності.

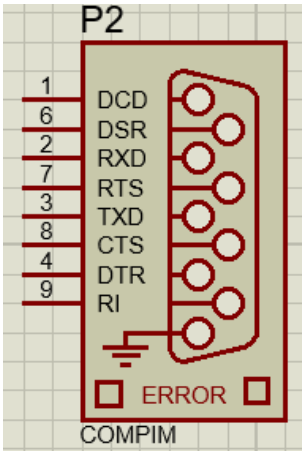
Мікроконтролер ATtiny44 має 14 виводів, розташування яких наведено на рисунку.



3. Система автоматизованого проектування Proteus

Proteus Design Suite- пакет програм для автоматизованого проектування електронних схем. Розробляється британською компанією Labcenter Electronics. Пакет являє собою систему схемотехнічного моделювання, що базується на основі моделей електронних компонентів, прийнятих в PSpice.

Proteus Design Suite включає в себе більше 6000 електронних компонентів з усіма довідковими даними, в тому числі інструмент COMPIM, який дає можливість підключити віртуальний пристрій COM-порта комп'ютера. COMPIM містить такі ж контакти як і 9-контактний роз’єм RS-232:



COMPIM має такі логічні рівні сигналу: "0" - 0 Вольт, "1" - 5 Вольт. Ці значення відрізняються від рівнів сигналу фізичного інтерфейсу RS-232, проте співпадають з логічними рівнями мікроконтролерів AVR. Можна сказати, що COMPIM являє собою інтерфейс RS-232 з перетворювачем логічних рівнів сигналу.

4. Середовище Bascom-AVR

Bascom-AVR - це інтегроване середовище розробки програм для мікроконтролерів семейства Atmel AVR на мові високого рівня (Basic). Програми, написані за допомогою Bascom-AVR без проблем можуть бути відлагоджені на моделях емулятора Proteus Design Suitечи макетних платах. Освоєнню мови допомагає багата довідкова система.

Після успішної компіляції програми в середовищі Bascom-AVR з'являється вікно, в якому вказано скільки відсотків пам'яті мікроконтролера займає дана програма. У папці з програмою з'являється файл з розширенням .hex, який можна вказати як програмний файл для мікроконтролера у середовищі Proteus.

Хід роботи

1. Встановлення програмного забезпечення.

a) Встановити систему автоматизованого керування Proteus Design Suite
b) Встановити середовище BASCOM-AVR.
c) Встановити емулятор віртуальних COM-портів Virtual Serial Ports Emulator (VSPE).

2. Зібрати у середовищі Proteus Design Suitecхему з мікроконтролером ATtiny44 та портом COMPIM (емуляція порту RS-232). Відповідно до свого варіанту під'єднати до мікроконтролера необхідні елементи.

3. Написати прошивку для мікроконтролера у середовищі Bascom-AVR та долучити файл прошивки до мікроконтролера в Proteus.

4. Написати застосування на будь-якій зручній мові програмування, яке керуватиме зібраним пристроєм.

5. Створити віртуальні COM-порти за допомогою програми VSPE та продемонструвати викладачу можливість керування пристроєм в середовищі Proteus Design Suiteза допомогою застосування, написаного в попередньому пункті.

Приклад створення пристрою для керування світлодіодами та отримання стану перемикачів.

У середовищі Proteus Design Suitезбираємо відповідну схему:

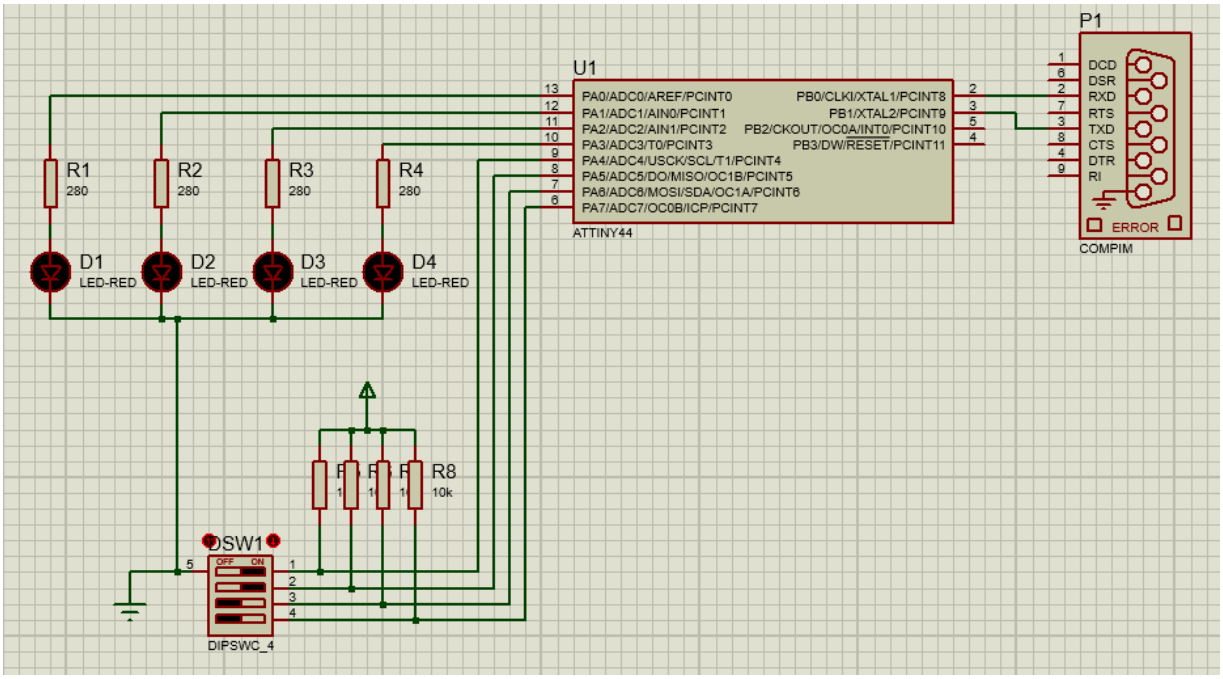


Рис. 2. Схема пристрою керування світло діодами.

В цій схемі мікроконтролер не виконує складних операцій і може постійно перебувати у стані очікування команди з послідовного порту. Тому можна обмежитись використанням контактів RXD та TXD з інтерфейсу RS-232. Під’єднуємо дані контакти до ніжок мікроконтролера.

Аноди світлодіодів під’єднуємо до ніжок мікроконтролера через резистори. Опір резисторів визначається наступною формулою: $R = (U – U_L) / I$, де U – напруга, яку видає мікроконтролер (5В); U_L – необхідна напруга на світлодіоді; I – необхідна сила струму на світлодіоді. Значення U_L та I можна подивитись в налаштуваннях світлодіодів. Катоди світлодіодів заземлюємо.

Перемикачі під’єднуються одним контактом до ніжок мікроконтролера. Також цей контакт має бути під’єднаний до напруги живлення через достатньо великий опір. Інший контакт під’єднується до землі.

Тепер необхідно створити прошивку для мікроконтролера. Запускаємо середовище Bascom-AVR:

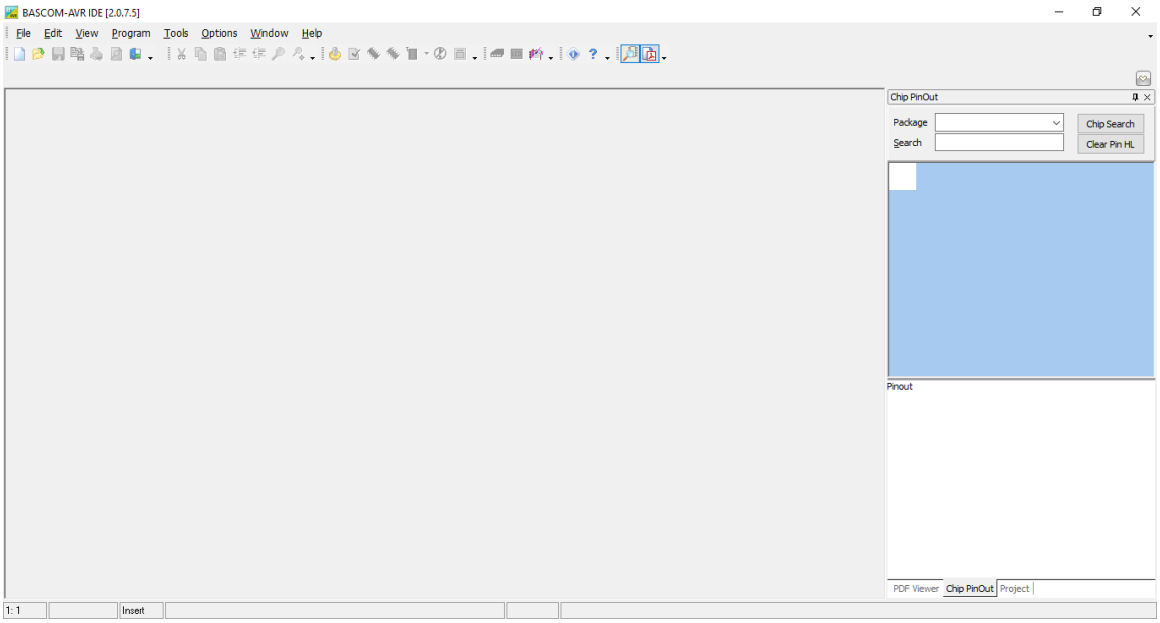


Рис. 3. Інтерфейс програми.

Натискаємо File>>New і з’являється вікно для написання нової програми. Вставляємо туди наступний код:

```
$regfile = "ATtiny44.dat" 'вказуємо наш мікроконтролер
$crystal = 1000000 'встановлюємо тактову честоту



Config Porta.0 = Output 'кофігуруємо ніжки, до яких під'єднані
Config Porta.1 = Output 'світлодіоди, як виходи
Config Porta.2 = Output
Config Porta.3 = Output
Config Pina.4 = Input 'кофігуруємо ніжки, до яких під'єднані
Config Pina.5 = Input 'світлодіоди, як виходи
Config Pina.6 = Input
Config Pina.7 = Input

Dim A As String * 2
Dim B As String * 1
Dim C As Byte

Open "comb.0:9600,8,n,1" For Input As #1 'кофігуруємо ніжки, до яких під'єднані
Open "comb.1:9600,8,n,1" For Output As #2 'контакти послідовного порта

Do 'робочий цикл мікроконтролера
Input #1 , A 'зчитуємо строку з послідовного порта у змінну A
B = Mid(a , 1 , 1) 'записуємо перший символ строки A у змінну B
```

```
C = Hexval(b) 'перетворюємо шіснадцяткове значення змінної B у байт C
If C = 0 Then 'керуємо світлодіодами
B = Mid(a , 2 , 1) 'записуємо другий символ строки A у змінну B
C = Hexval(b) 'перетворюємо шіснадцяткове значення змінної B у байт C
Porta.3 = C.0 'відповідно до значень змінної C встановлюємо
Porta.2 = C.1 'стан світлодіодів
Porta.1 = C.2
Porta.0 = C.3
End If
If C = 1 Then 'отримуємо стан перемикачів
C = 0
C.0 = Not Pina.7 'зчитуємо стан перемикачів у змінну C
C.1 = Not Pina.6
C.2 = Not Pina.5
C.3 = Not Pina.4
Print #2 , Hex(c) 'надсилаємо на послідовний порт змінну C у шіснадцятковому вигляді
End If
Loop
End
```

Програма очікує надходження строки з COM-порта. Якщо прийшла строка, в якій перший символ “0”, то в залежності від другого символу, який має бути шіснадцятковим числом, запалюються відповідні світлодіоди. Якщо прийшла строка, в якій перший символ “1”, то мікроконтролер зчитує стан перемикачів та надсилає це значення у шіснадцятковому вигляді на послідовний порт.

Тепер натискаємо Program>>Compile та обираємо папку куди збережемо нашу програму. У випадку успішної компіляції з’явиться вікно, на якому вказано відсоток зайнятості пам’яті мікроконтролера:

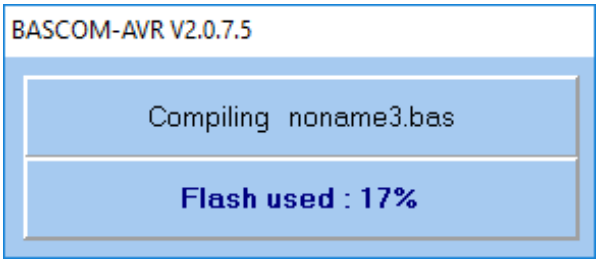


Рис. 4. Вікно програми.

Після цього в обраній папці з’явиться файл з розширенням .hex, який і потрібно під’єднати до мікроконтролера в середовищі Proteus Design Suite

На нашій схемі у середовищі Proteus Design Suitевідкриваємо властивості мікроконтролера:

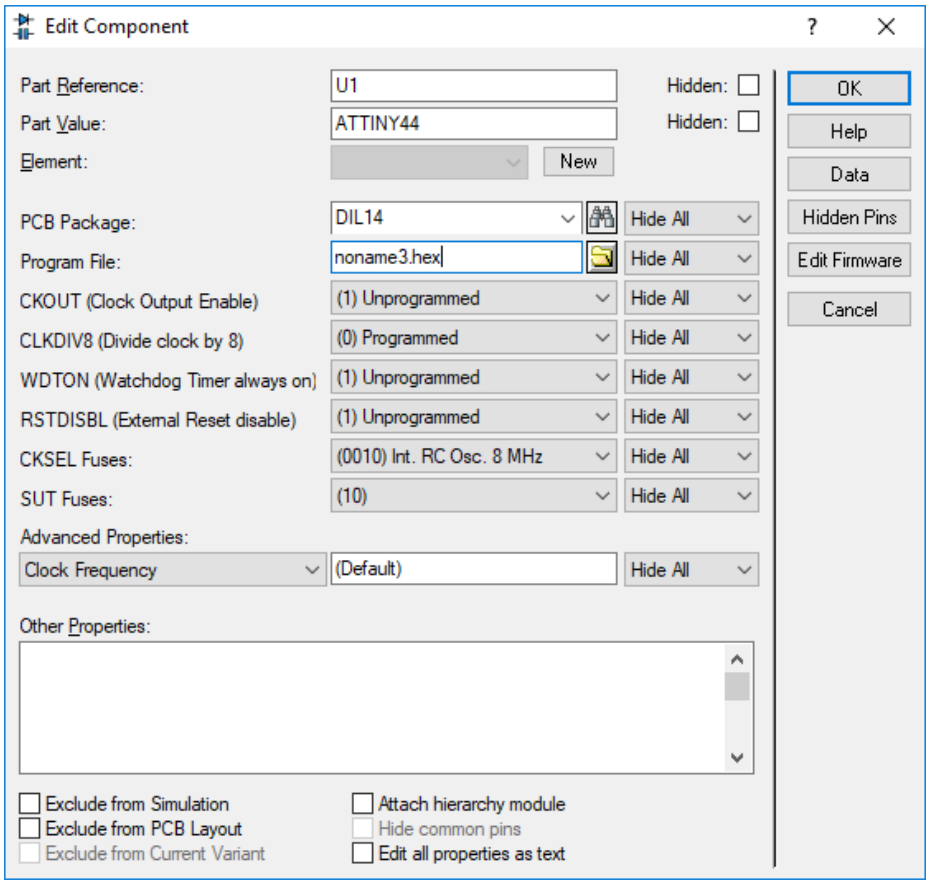


Рис. 4. Вікно властивостей програми.

В пункті Program File натискаємо значок папки та знаходимо раніше скомпільований файл з розширенням .hex. Також в пункті CKSEL Fuses встановлюємо значення як на рисунку (тут ми обираємо генератор тактових імпульсів для мікроконтролера). Натискаємо OK і тепер в наш мікроконтролер “зашита” програма, яку ми написали раніше. При перекомпіляції програми в Bascom-AVR під’єднувати її ще раз до мікроконтролера не потрібно.

Також, якщо плануємо використовувати ніжку PB3, в пункті RSTDISBL потрібно вибрати значення 0. В іншому випадку ця ніжка буде недоступна для використання.

Далі потрібно створити застосування, яке керуватиме роботою нашого пристрою. В даному випадку буде створено застосування WindowsForms мовою програмування C#. Додаємо на форму дві кнопки, чотири прапорці (CheckBox) та один Label:

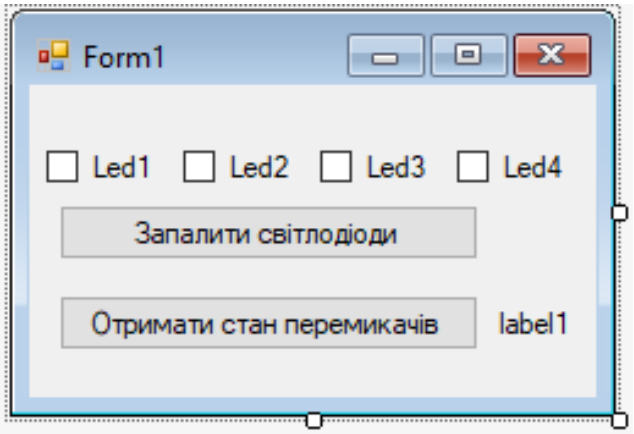


Рис. 4. Вікно програми керування світлодіодами.

Потрібно обробити події натиснення кнопок та закриття форми. Прописуємо наступний код:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;//підключаємо цей простір імен для роботи з послідовним портом


namespace WindowsFormsApplication1
{
 public partial class Form1 : Form
 {
 SerialPort port;//послідовний порт, з яким взаємодіятиме дана програма
 public Form1()
 {
 InitializeComponent();
 label1.Text = "";
 try
 {
 port = new SerialPort("COM1", 9600, Parity.None, 8, StopBits.One);//ініціалізуємо
послідовний порт
 port.Open();//відкриваємо послідовний порт
 port.NewLine = "\r";//встановлюємо символ перенесення строки
```

```
 }
 catch (Exception)
 {
 button1.Enabled = false;
 checkBox1.Enabled = false;
 checkBox2.Enabled = false;
 checkBox3.Enabled = false;
 checkBox4.Enabled = false;
 MessageBox.Show("Порт не підключений");
 }
 }

 private void button1_Click(object sender, EventArgs e)
 {
 int a = 0;
 if (checkBox1.Checked) a += 8;
 if (checkBox2.Checked) a += 4;
 if (checkBox3.Checked) a += 2;
 if (checkBox4.Checked) a++;
 port.WriteLine("0" + a.ToString("X"));//надсилаємо на послідовний порт 0 і значення
прапорців у шіснадцятковому вигляді
 }

 private void button2_Click(object sender, EventArgs e)
 {
 port.WriteLine("1");//надсилаємо на послідовний порт 1
 string s = port.ReadLine();//зчитуємо строку з послідовного порту
 s = s.Substring(s.Length - 1);//видаляємо зі строки все крім останнього символу
 s = Convert.ToString(Convert.ToInt32(s, 16), 2);//конвертуємо шіснадцяткове значення у
двійкове
 while (s.Length < 4)
 {
 s = "0" + s;
 }
 label1.Text = s;
 }

 private void Form1_FormClosed(object sender, FormClosedEventArgs e)
 {
 port.Close();//закриваємо послідовний порт
 }
 }
}
```

Дана програма при натисненні першої кнопки надішле на COM-порт символ “0” та відповідне шістнадцяткове число в залежності від прапорців. При натисненні другої кнопки, програма надішле на COM-порт символ “1”, зчитає з COM-порта стан перемикачів та виведе це значення у Label.

Тепер необхідно створити віртуальні COM-порти, через які наша програма керуватиме зібраною схемою. Запускаємо застосування Virtual Serial Ports Emulator (VSPE):

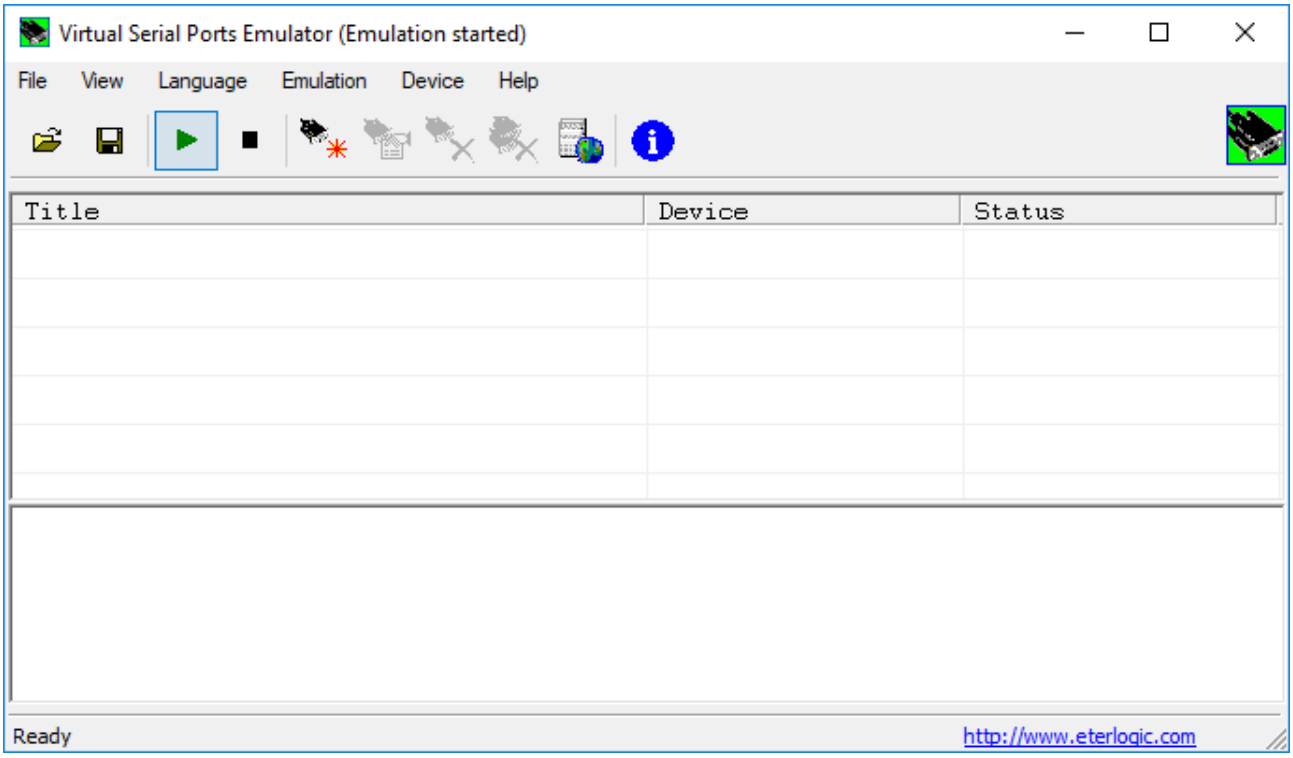


Рис. 5. Вікно програми емулятора портів.

Натискаємо Device>>Create та створюємо новий пристрій:

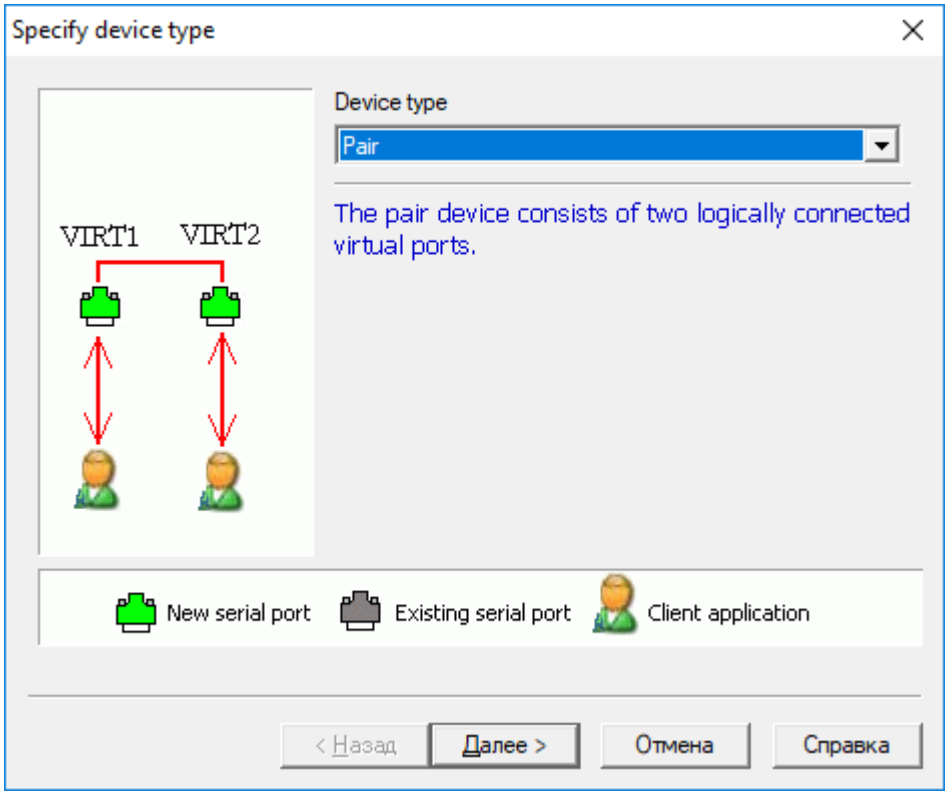


Рис. 6. Вікно вибору типу пристроїв.

В пункті Device type обираємо Pair та натискаємо далі:

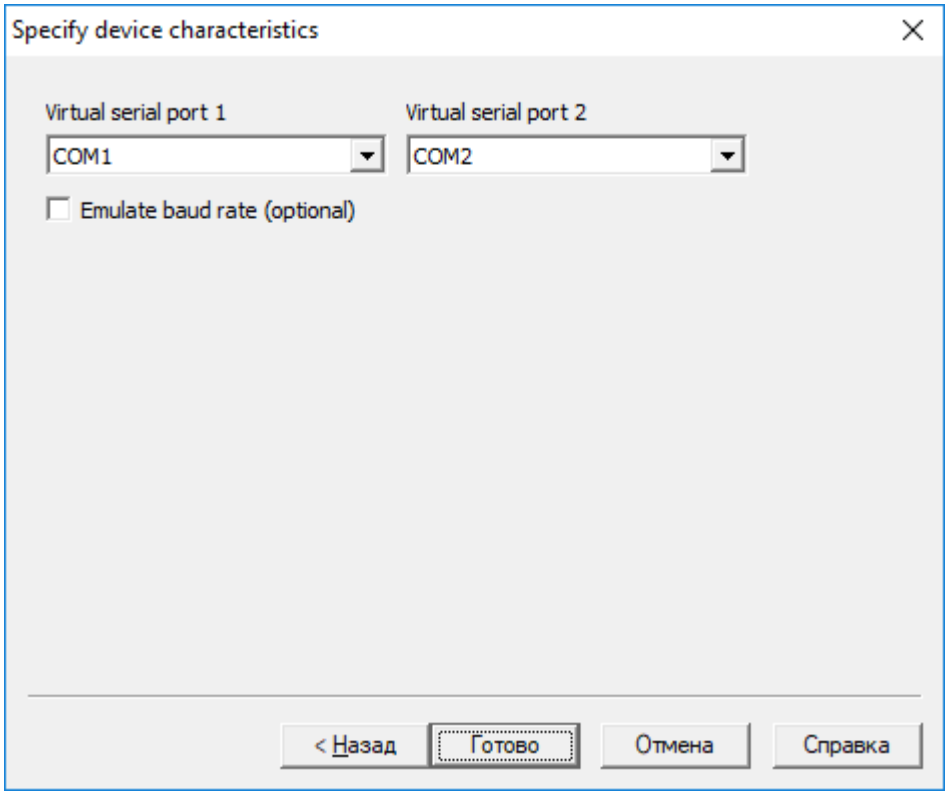


Рис. 7. Вікно вибору номера порту.

Обираємо номери наших COM-портів та натискаємо “Готово”. Після цього у нас з'явиться пара віртуальних COM-портів:

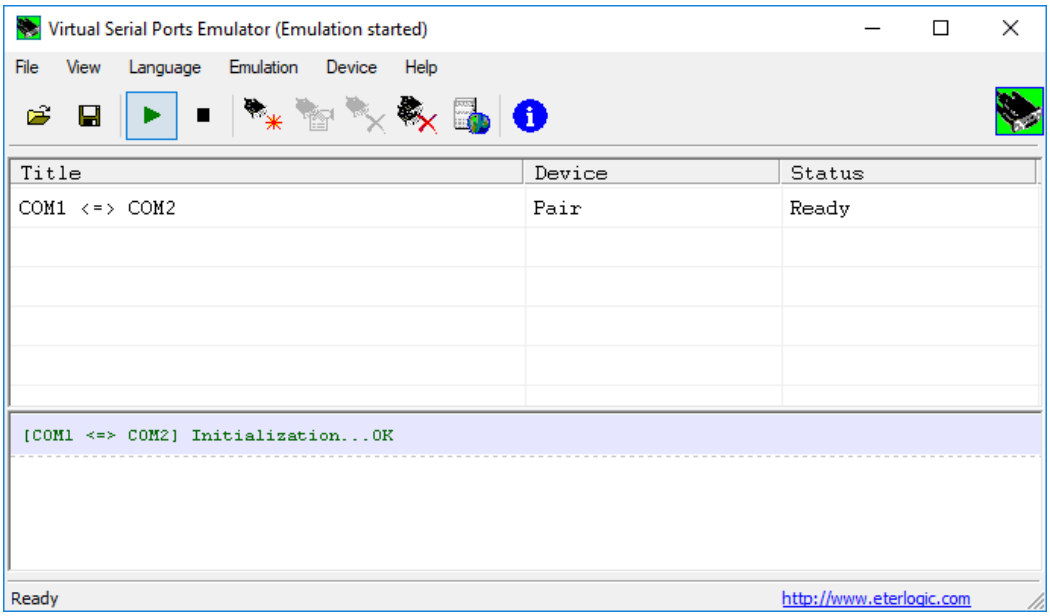


Рис. 8. Вікно програми емулятора портів.

Варто зазначити, що створені віртуальні порти з’єднані між собою перехресно. Це означає, що контакт RXD одного COM-порта під’єднаний до контакту TXD іншого і навпаки. Те ж саме з контактами RTS та CTS. Цей факт необхідно враховувати при написанні прошивки мікроконтролера, а саме при конфігурації ніжок, що під’єднані до контактів послідовного порта.

Зверніть увагу, один з віртуальних COM-портів пари потрібно вказати при ініціалізації послідовного порта в C# застосуванні, а інший у властивостях COMPIM на схемі у Proteus Design Suite:

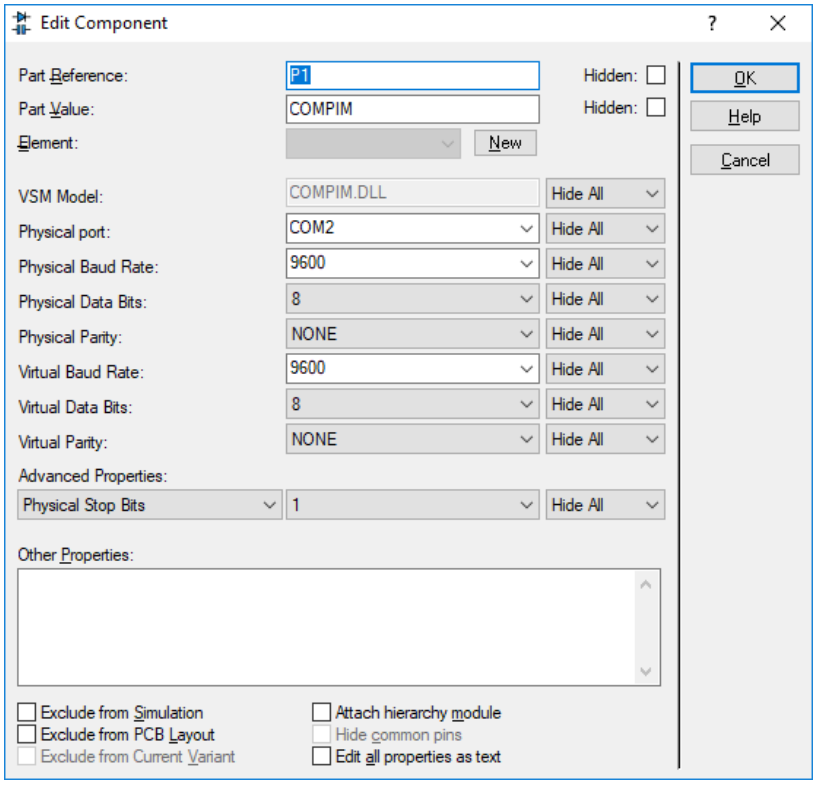


Рис. 9. Вікно вибору параметрів порту.

Інші параметри (Baud Rate, Parity, Data Bits, Stop Bits) обов’язково повинні співпадати і у властивостях COMPIM, і у налаштуваннях COM-порта в C# застосування. Такі ж параметри повинні бути вказані в налаштуванні ніжок мікроконтролера, що під’єднуються до COM-порта, в середовищі Bascom-AVR.

Коли все налаштовано запускаємо емуляцію схеми в Proteus Design Suite та запускаємо C# застосування. Перевіряємо правильність роботи:

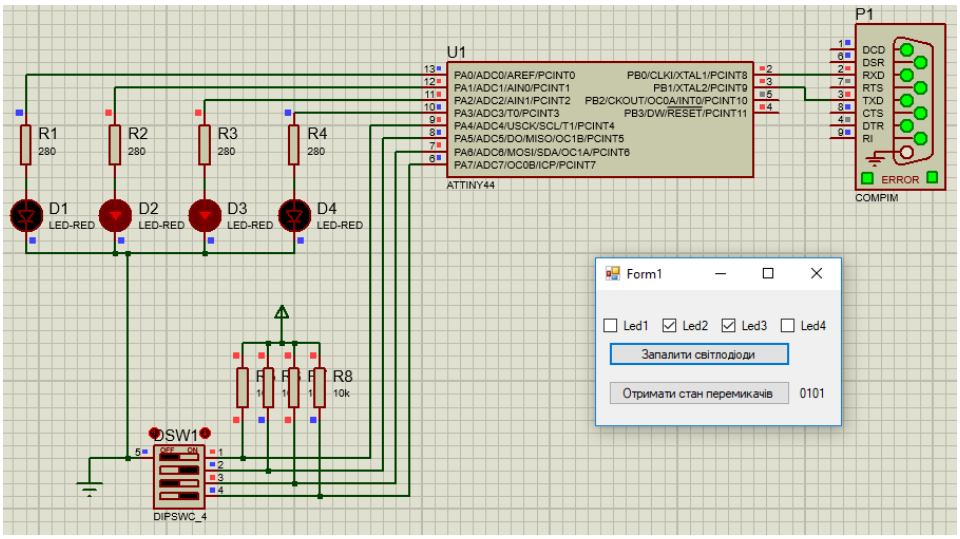
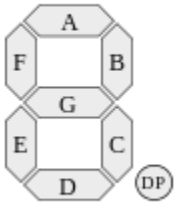


Рис. 10 Принципова схема

Приклади під’єднання деяких пристроїв до мікроконтролера

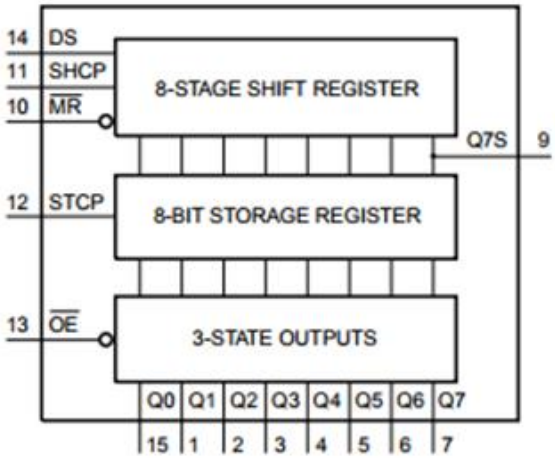
Семисегментні індикатори зі спільним катодом

Семисегментні індикатори під’єднуються до мікроконтролера так само як і звичайні світлодіоди. Кожен сегмент через резистор під’єднується до ніжки мікроконтролера, а спільний анод заземлюється.



Перші сім ніжок індикатора відповідають за загорання сегментів від A до G відповідно. Восьма ніжка дає можливість запалити крапку.

Якщо під’єднувати сегменти індикатора напряму до мікроконтролера можуть дуже швидко закінчитись ніжки. Вирішити цю проблему можна за допомогою регістрів зсуву 74HC595. Мікросхема 74HC595 складається з 8-бітного регістру зсуву, 8-бітного регістру для збереження та виходів з трьома станами:



Q0-Q7 – паралельні виходи.

Q7S – вихід для послідовного з’єднання регістрів.

MR – скидання значень регістру.

SHCP – вхід для тактових імпульсів.

STCP – синхронізація виходів.

OE – вхід для перемикання стану виходів з високоомного в робоче.

DS – вхід для послідовних даних.

В Bascom є команда Shiftout, яка дає можливість послідовно подавати біти даних на певний вихід мікроконтролера. В якості аргументів дана команда приймає: вихід для даних, вихід для тактових імпульсів, байт даних, опція передачі (в загальному випадку ставиться 1,- означає що першим подається старший біт байту даних).

Після того, як дані записано на регістр зсуву, потрібно синхронізувати виходи мікросхеми 74HC595. Це можна зробити за допомогою команди Pulseout. В якості аргументів дана команда приймає: порт, номер піна, тривалість імпульсу в тактах.

Схема яка дає можливість вивести будь-які значення на два семисегментні індикатори:

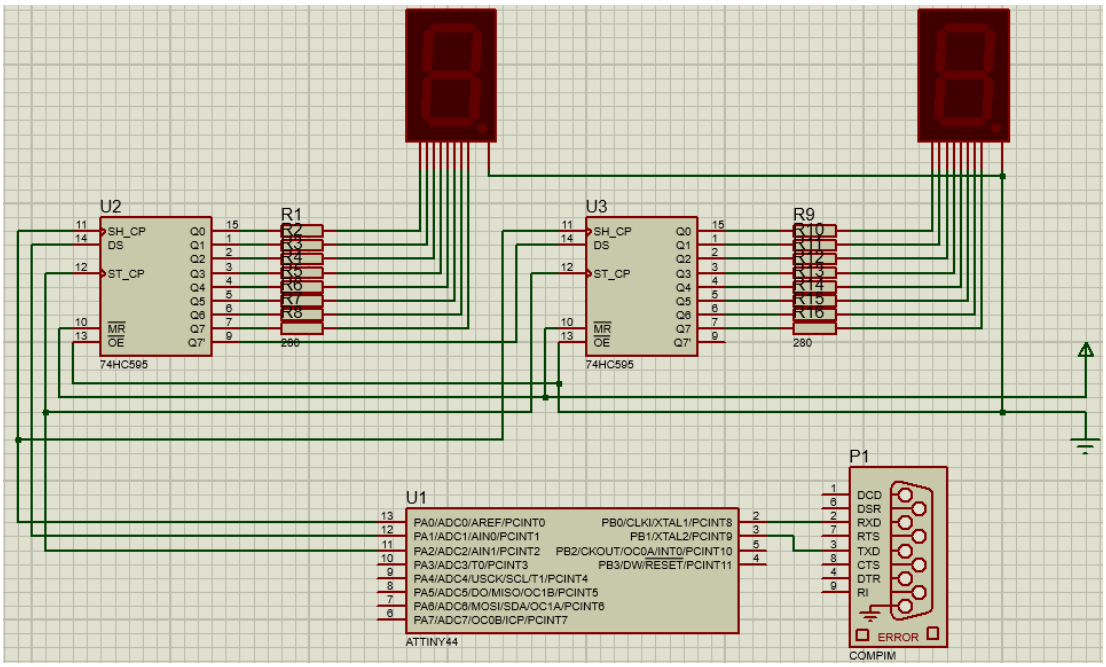


Рис. 10. Принципова схема з 7-сегментними індикаторами.

Мікроконтролер очікує прийому чотирьох шіснадцяткових символів з послідовного порта, в залежності від яких встановлює значення на семисегментних індикаторах. Код прошивки мікроконтролера та код C# застосування наведено в додатку А.

Датчик температури DS18B20

DS18B20 обмінюється даними з мікроконтролером за однопровідною лінією зв'язку, використовуючи протокол інтерфейсу 1-Wire. Діапазон вимірювання температури становить від -55 до +125 ° C. Для діапазону від -10 до +85 ° C похибка не перевищує 0,5 °C. Значення температури зберігається в доповнюваному коді у двобайтному регістрі:

	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
LS Byte	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
MS Byte	S	S	S	S	S	2^6	2^5	2^4

Контакти VCC та GND датчика температури DS18B20 під'єднуються до напруги живлення та землі відповідно. Контакт DQ під'єднується до ніжки мікроконтролера, зконфігурованої на роботу за протоколом 1-Wire. Також, лінія 1-Wire, під'єднується до напруги живлення через резистор опором 4700 Ом.

Перед надсиланням будь-якої команди на лінію 1-Wire вказати до яких пристроїв звертатися (надіславши значення CCh ми вказує на звернення до всіх пристроїв, під'єднаних до лінії 1-Wire). Після завершення команди потрібно виконати скидання лінії 1-Wire(1wreset). Основні команди протоколу 1-Wire для роботи з датчиком теператури:

- 44h – команда конвертації температури в бінарний вигляд (виконується 750 мс).
- BEh – команда читання пам'яті датчика. Після цієї команди з лінії 1-Wire можна зчитати 2 байти інформації.

Для того, щоб отримувати актуальне значення температури мікроконтролер повинен опитувати датчик температури кожні 750 мс. Отже, мікроконтролер не може постійно очікувати команди з послідовного порту. Тому потрібно реалізувати апаратний метод керування. Для цього необхідно використати контакти CTS та RTS розніму RS-232. Контакт RTS хоста має обов'язково бути під'єднаний до піну з перериванням INT0(пам'ятаємо, що наші віртуальні COM-порти під'єднані перехресно, отже RTS хоста під'єднано до CTS COMPIM,- тому до піну з перериванням INT0 слід під'єднати контакт CTS). Дане переривання слід обробити в прошивці мікроконтролера.

Схема для роботи з датчиком температури DS18B20:

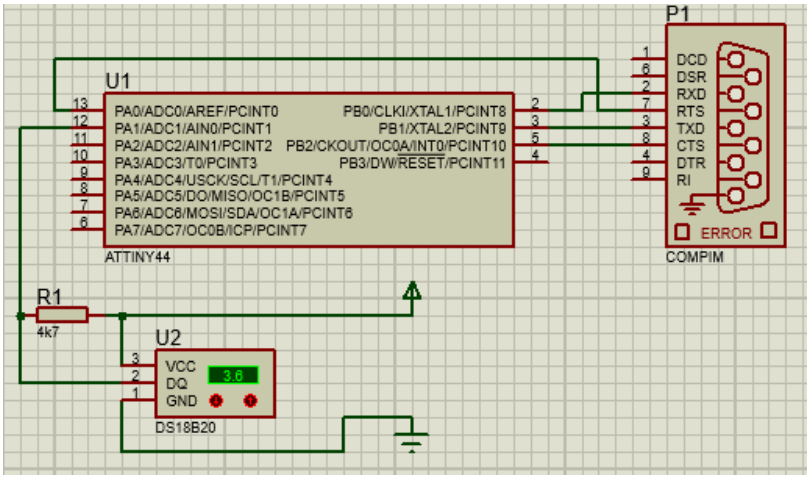


Рис. 11. Принципова схема з послідовним портом

Дана схема при отриманні з послідовного порту символу “0”, надсилає на послідовний порт два байта з датчика температури. У випадку апаратного методу керування надсилати дані мікроконтролеру потрібно наступним чином:

```
port.RtsEnable = true;//інформуємо мікроконтролер про наявність даних для передачі

while (!port.CtsHolding) System.Threading.Thread.Sleep(1);//очікуємо готовність
мікроконтролера прийняти дані

port.WriteLine("0");//передаємо дані

port.RtsEnable = false;//більше немає, що передавати
```

Код прошивки мікроконтролера та код C# застосування наведено в додатку Б.

Постійний запам’ятовуючий пристрій DS2430

DS2430 – це 256-бітний постійний запам’ятовуючий пристрій, який обмінюється даними з іншими пристроями за однопровідною лінією зв'язку, використовуючи протокол інтерфейсу 1-Wire. Пам’ять організовано як одну сторінку з 32 байтів для довільного доступу. DS2430 складається з власне фінальної 256-бітної пам’яті та з 256-бітної енергозалежної проміжної скретч-пам’яті.

Записувати та зчитувати дані можна тільки з проміжної скетч-пам’яті.

DS2430 під’єднується до мікроконтролера так само, як і датчик температури. Принципи роботи за протоколом 1-wire такі ж, як і з датчиком температури. Основні команди протоколу 1-Wire для роботи з постійним запам’ятовуючим пристроєм:

- F0h – команда копіювання даних з постійної пам’яті до проміжної скретч-пам’яті. Після цієї команди обов’язково потрібно подати ще один байт адреси, з якої буде зчитано один байт даних зі скретч-пам’яті. Після читання байту з лінії 1-wire DS2430 автоматично інкрементує адресу і можна одразу прочитати наступний байт і так далі. Після читання байту за адресою 1Fh значення адреси встановиться у 00h і можна продовжувати читати інформацію до надходження команди скидання лінії.
- 55h – команда копіювання даних зі скретч-пам’яті до постійної пам’яті. Після цієї команди слід подати ключ перевірки A5h. Після цієї команди лінія 1-wire повинна бути встановлена в логічну одиницю на 10мс.
- AAh – команда читання даних зі скретч-пам’яті. Після команди слід подати адресу байту, який хочемо прочитати. Аналогічно з командою F0h можна читати наступні байти до надходження команди скидання лінії.
- 0Fh – команда запису даних до скретч-пам’ті. Після команди потрібно подати адресу та значення, яке хочемо записати. Аналогічно з командами F0h та AAh можна записувати наступні байти до надходження команди скидання лінії.

Схема для роботи з постійним запам’ятовуючим пристроєм DS2430:

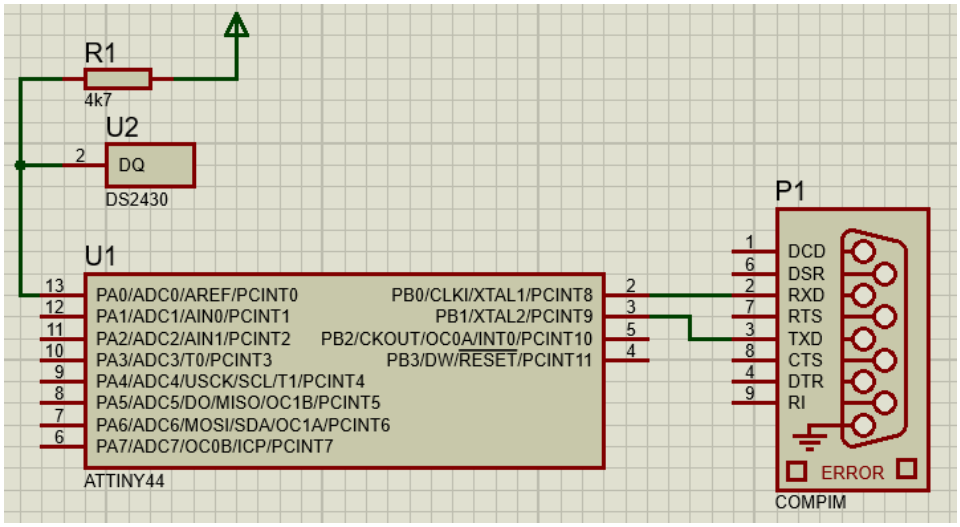


Рис. 12. Принципова схема з COM-портом.

Схема може оброблять команди: копіювання даних з постійної пам’яті до скретч-пам’яті, копіювання даних зі скретч-пам’яті до постійної пам’яті, читання байту даних зі скретч-пам’яті, запис байту даних до скретч-пам’яті.

Код прошивки мікроконтролера та код C# застосування наведено в додатку В.

Світлодіодна матриця 8×8

Світлодіодна матриця 8×8 містить в загальному 64 світлодіоди. Дуже складно та затратно керувати цими 64-ма світлодіодами індивідуально, використовуючи будь-які цифрові схеми чи мікроконтролер. Отже, потрібно виконати мультиплексування щоб взаємодіяти з матрицею через мінімальну кількість контактів. Аноди всіх світлодіодів в кожному стовпці з’єднуються між собою і катоди всіх світлодіодів в кожному рядку також з’єднуються між собою:

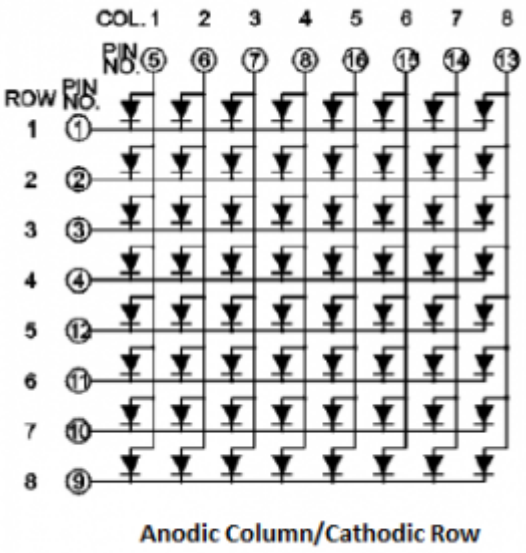


Рис. 12. Схема світлодіодної матриці.

За допомогою такого з’єднання можна керувати світлодіодною матрицею через 16 контактів. Проте, встановивши статичні значення на контактах матриці ми не можемо вивести будь-яке зображення. В моделі Proteus Design Suiteвиводи рядків розміщені знизу.

Можна встановити логічний нуль на першому рядку, а на всіх інших логічну одиницю. Подавши значення на стовпці, можна вивести будь-яке зображення в першому рядку. Потім можна подати логічний нуль тільки на другий рядок і вивести в ньому будь-яке зображення і так далі. Таким чином в кожен момент часу світитиметься тільки один рядок. Якщо запалювати рядки по черзі з великою частотою оновлення, то за рахунок інерційності зору буде здаватися, що зображення на матриці нерухоме.

Мікроконтролер повинен постійно керувати тим, який рядок світиться і що на ньому відображується. Отже, мікроконтролер не може постійно очікувати команди з послідовного порту. Аналогічно з датчиком температури потрібно реалізувати апаратний метод керування (використовуючи контакти RTS та CTS).

Для економії ніжок під'єднуємо світлодіодну матрицю до мікроконтролера за допомогою регістрів зсуву 74HC595.

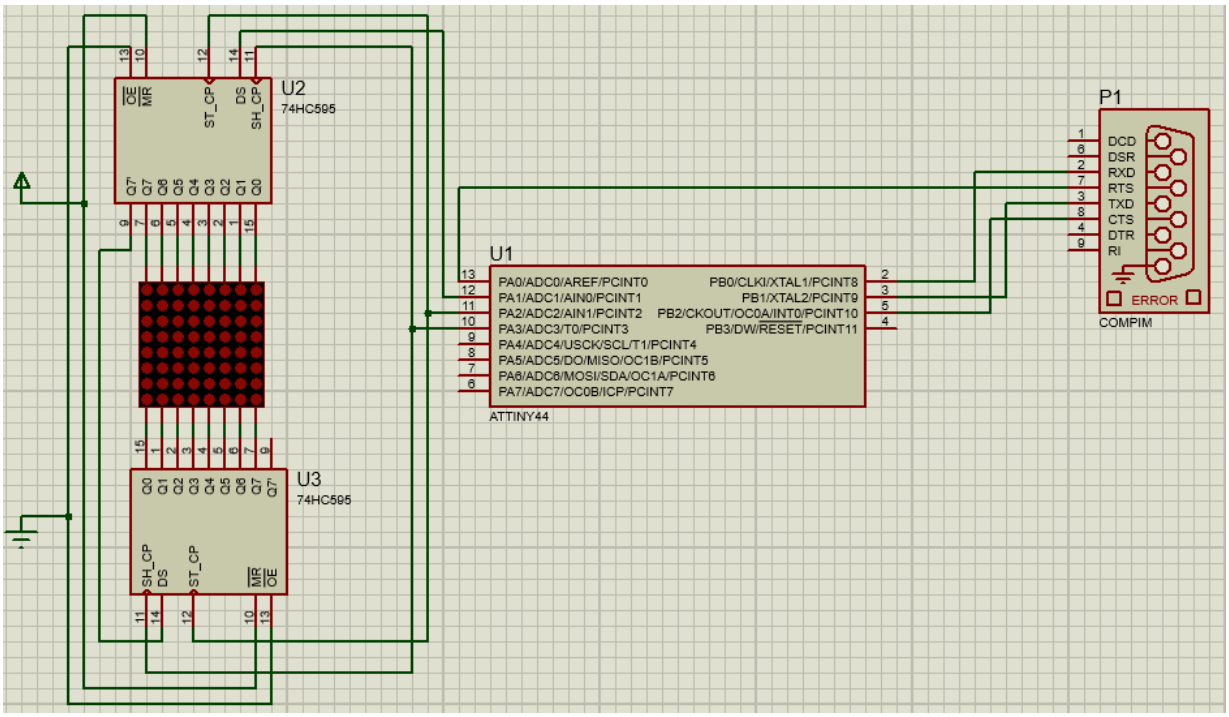


Рис. 13. Принципова схемакерування світлодіодною мартицею 8x8.

Мікроконтролер при надходженні 16 шіснадцяткових символів з послідовного порта оновлює зображення на світлодіодній матриці. Код прошивки мікроконтролера та код C# застосування наведено в додатку Г.

Варіанти завдань

1. До мікроконтролера ATtiny44 під'єднати COMPIM, постійний запам'ятовуючий пристрій DS2430 та 8 світлодіодів (за допомогою регістру зсуву 74HC595). Передбачити можливість виконання наступних команд:
А) Запис байту даних до запам'ятовуючого пристрою. Значення байту та адресу куди записувати отримуємо з послідовного порта.
Б) Виведення байту даних з запам'ятовуючого пристрою на світлодіоди. Адресу байта отримуємо з послідовного порта.

2. До мікроконтролера ATtiny44 під'єднати COMPIM, постійний запам'ятовуючий пристрій DS2430 та 2 семисегментних індикатори зі спільним катодом. Передбачити можливість виконання наступних команд:
А) Запис двох послідовних байтів даних до запам'ятовуючого пристрою. Значення байтів та адресу куди записувати отримуємо з послідовного порта.
Б) Виведення двох послідовних байтів даних з запам'ятовуючого пристрою на семи сегментні індикатори. Адресу звідки зчитувати отримуємо з послідовного порта.

Так як ми оперуємо двома байтами, то вважаємо, що запис та зчитування з запам'ятовуючого пристрою відбувається за адресами кратними двум (0,2,4,…,30).

3. До мікроконтролера ATtiny44 під'єднати COMPIM, постійний запам'ятовуючий пристрій DS2430 та світлодіодну матрицю 8x8. Передбачити можливість виконання наступних команд:
А) Запис восьмибайтового зображення до запам'ятовуючого пристрою. Значення байтів та адресу куди записувати отримуємо з послідовного порта.
Б) Виведення зображення, прочитаного з запам'ятовуючого пристрою на світлодіодну матрицю. Адресу байта отримуємо з послідовного порта.

Так як ми оперуємо 8-ма байтами, то вважаємо, що запис та зчитування з запам'ятовуючого пристрою відбувається за адресами кратними восьми (0,8,16,24).

4. До мікроконтролера ATtiny44 під'єднати COMPIM, датчик температури DS18B20 та 8 світлодіодів (за допомогою регістру зсуву 74HC595). Після надходження команди з комп'ютера мікроконтролер надсилає на послідовний порт температуру з датчика. Після отримання температури комп'ютер обробляє ці дані та надсилає мікроконтролеру байт з цілою частиною температури, який і виводиться на світлодіоди.

5. До мікроконтролера ATtiny44 під'єднати COMPIM, датчик температури DS18B20 та 4 семисегментних індикатори зі спільним катодом. Після надходження команди з комп'ютера мікроконтролер надсилає на послідовний порт температуру з датчика. Після отримання температури комп'ютер обробляє ці дані та надсилає мікроконтролеру 4 байти, які і виводяться на семисегментні індикатори: перший індикатор – знак, другий та третій – ціла частина, четвертий – дробова частина.

6. До мікроконтролера ATtiny44 під'єднати COMPIM, датчик температури DS18B20 та світлодіодну матрицю 8x8. Після надходження команди з комп'ютера мікроконтролер надсилає на послідовний порт температуру з датчика. Після отримання температури комп'ютер обробляє ці

дані та надсилає мікроконтролеру восьмибайтове зображення з цілою частиною температури, яке і відображатиметься на світлодіодній матриці.

7. До мікроконтролера ATtiny44 під'єднати COMPIM, постійний запам'ятовуючий пристрій DS2430, 8 світлодіодів (за допомогою регістру зсуву 74HC595) та 3 перемикачі. Мікроконтролер повинен виводити на світлодіоди значення байту, адреса якого визначається станом перемикачів, з запам'ятовуючого пристрою. При надходженні байту даних з послідовного порту цей байт записується до запам'ятовуючого пристрою за адресою, що визначається станом перемикачів.

8. До мікроконтролера ATtiny44 під'єднати COMPIM, постійний запам'ятовуючий пристрій DS2430, 2 семисегментних індикатори зі спільним катодом та 3 перемикачі. Мікроконтролер повинен виводити на світлодіодні індикатори значення двох послідовних байтів, адреса першого з яких визначається станом перемикачів, з запам'ятовуючого пристрою. При надходженні двох байтів даних з послідовного порту ці байти послідовно записується до запам'ятовуючого пристрою за адресою, що визначається станом перемикачів.

Так як ми оперуємо двома байтами, то вважаємо, що запис та зчитування з запам'ятовуючого пристрою відбувається за адресами кратними двум (0,2,4,…,30).

9. До мікроконтролера ATtiny44 під'єднати COMPIM, постійний запам'ятовуючий пристрій DS2430, світлодіодну матрицю 8x8 та 2 перемикачі. Мікроконтролер повинен виводити на світлодіодну матрицю зображення, адреса якого визначається станом перемикачів, з запам'ятовуючого пристрою. При надходженні восьмибайтового зображення з послідовного порту це зображення записується до запам'ятовуючого пристрою за адресою, що визначається станом перемикачів.

Так як ми оперуємо 8-ма байтами, то вважаємо, що запис та зчитування з запам'ятовуючого пристрою відбувається за адресами, кратними 8 (0,8,16,24).

10. До мікроконтролера ATtiny44 під'єднати COMPIM, 4 світлодіоди та 2 перемикачі. Кожен світлодіод може перебувати в трьох станах: світиться, не світиться, блимає. Мікроконтролер отримує з послідовного порту адрес світлодіоду та стан, в який його треба перевести. Перемикачами визначається частота блимання світлодіодів (2 Гц; 1 Гц; 0,5 Гц; 0,25 Гц).

Запитання для закріплення матеріалу лабораторної роботи:

1. Які є методи управління обміном даними в протоколі RS-232?
2. Чи можна фізичний інтерфейс RS-232 напряму під'єднувати до мікроконтролера ATtiny44? Чому?
3. Яким чином за допомогою регістрів зсуву 74HC595 можна збільшити кількість виходів мікроконтролера?
4. Чи можуть 1-Wire пристрої працювати без під'єднання лінії живлення? Чому?

ЛАБОРАТОРНА РОБОТА №7.

«Налаштування середовища програмування мовою С в операційній системі Android мобільного пристрою»

Тема: Налаштування набору компіляторів GCC в операційній системі Android.

Мета: Навчитися самостійно встановлювати на мобільний пристрій середовище Termux, службу SSH (Secure SHell), текстовий редактор, файловий менеджер та набір компіляторів GCC (GNU Compiler Collection).

Теоретичні відомості

1. Командне середовище Termux

Програма Termux - це емулятор терміналу з відкритим кодом для Android, який дає можливість запускати середовище Linux на пристрої Android. Termux автоматично встановлює мінімальну базову систему, до якої входять основні команди операційної системи Linux, текстові редактори і файлові менеджери, додаткові пакунки? доступні за допомогою стандартного менеджера пакетів apt, заснованого на проекті Debian/Ubuntu. Більшість команд, доступних у Linux, є також доступні в середовищі Termux, а також вбудовані команди серверу сценаріїв Bash.

2. Набір компіляторів GCC (GNU Compiler Collection)

Компілятор GCC розроблений некомерційною організацію Фонд Вільних Програм (Free Software Foundation, FSF) і одвічно підтримував лише одну мову програмування — C. На цей час компілятор підтримує різні мови програмування, апаратні архітектури, операційні системи. Це стандартний компілятор для Unix-подібних операційних систем, і ряду комерційних операційних систем на основі Unix, зокрема Apple MacOS. В методичному посібнику планується використання двох мов програмування, що підтримуються набором компіляторів GCC - мова C та мова Assembler під архітектуру команд ARM32/ARM64.

3. Дебагер GDB (GNU DeBugger)

Дебагер GDB - універсальний дебагер проекту GNU, яких дає можливість відлагоджувати програми, написані різними мовами програмування, якщо компілятор відповідної мови підтримує спеціальний режим компіляції, додаючи до скомпільованої програми додаткову службову інформацію, необхідну для покрокового виконання програми дебагером GDB.

Хід роботи:

1. Використовуючи програму PlayMarket становіть на пристрої з операційною системою Android версії 9+ середовище Termux (рис. 1А).

2. Виконайте команду

pkg update

і проконтролюйте, щоб система отримала з репозиторія оновлений перелік доступних пакетів. Якщо ця команда буде виконана неуспішно - необхідно змінити репозиторій оновлення. Для цього виконайте команду

termux-change-repo

і встановіть за допомогою цієї програми (рис. 1Б), як приклад, новий репозиторій - Grimler.se (рис. 1В). Після успішної заміни репозиторію повторно виконайте команду

pkg update

При цьому на усі інтерактивні запитання “Y/N?” дайте позитивні відповіді Y,Yes (рис. 2А), і уважно проконтролюйте, щоб процес оновлення пакетів середовища (рис. 2Б) було успішно виконано (рис. 2В).

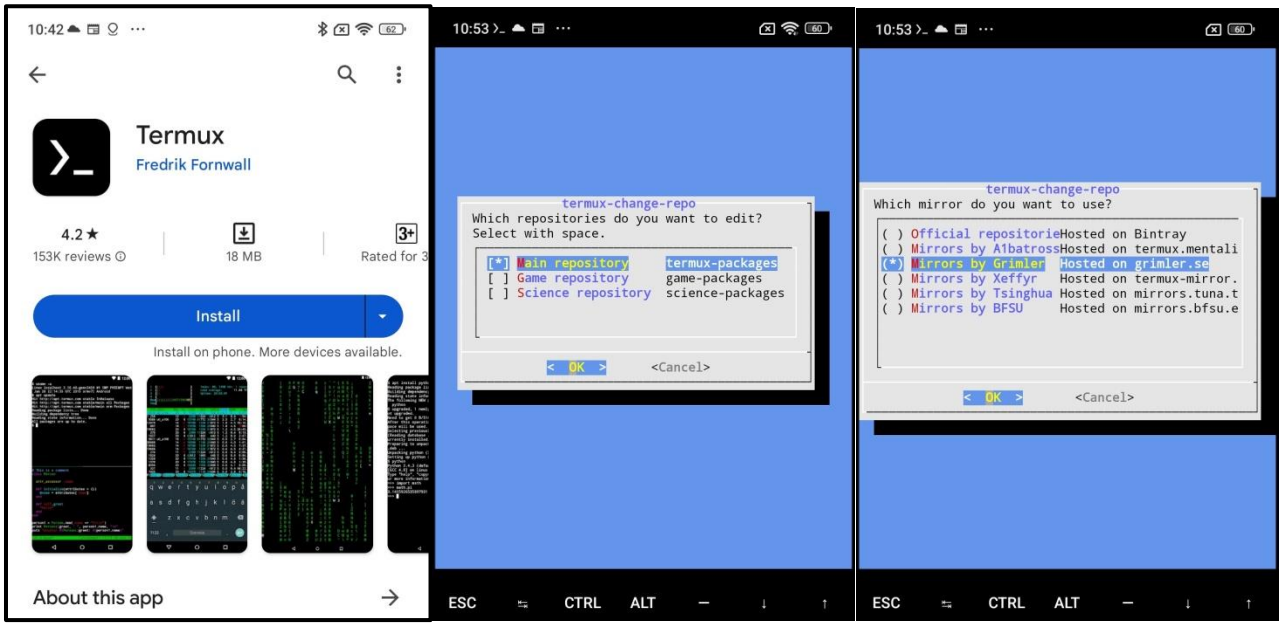


Рис 1. Етапи встановлення середовища та зміна репозиторію.

3. Виконайте команду

```
ip addr
```

Якщо результатом є «program is not installed», то потрібно встановити пакет iproute2, тобто виконайте команду

```
pkg install iproute2
```

і повторно виконайте команду

```
ip addr
```

знайдіть і запам’ятайте IP-адресу Android-пристрою, наприклад 192.168.0.145. В подальшому цю IP-адресу слід сприймати виключно як приклад і при виконанні команд замінювати її на справжню IP-адресу Вашого пристрою.

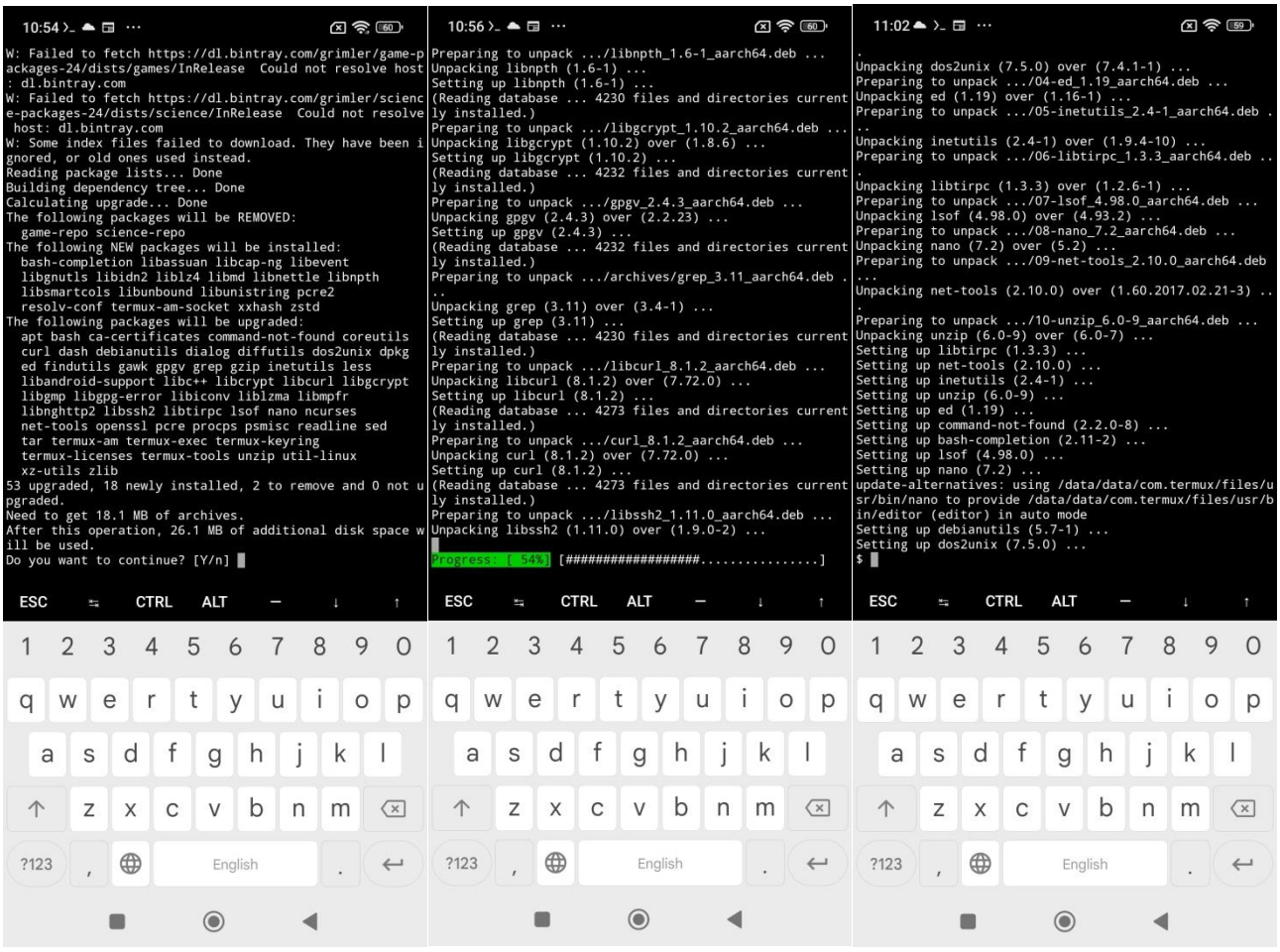


Рис 2. Етапи виконання команди «pkg update» після заміни репозиторію.

4. На своєму ноутбуку або стаціонарному комп’ютері виконайте команду

```
ping 192.168.0.105 -t
```

щоб припинити нескінченне надсилання ping-запитів - натисніть Ctrl-C. Якщо ping-відповіді від Android-пристрою не надходять, самостійно не продовжуйте роботу, а зверніться до викладача.

5. Виконайте команду

```
sshd
```

і зверніть увагу на повідомлення «The program sshd is not installed» (Рис. 3А).

6. Виконайте команду

```
pkg install openssh
```

і проконтролюйте, що служба SSH-сервера успішно встановлена (Рис. 3Б).

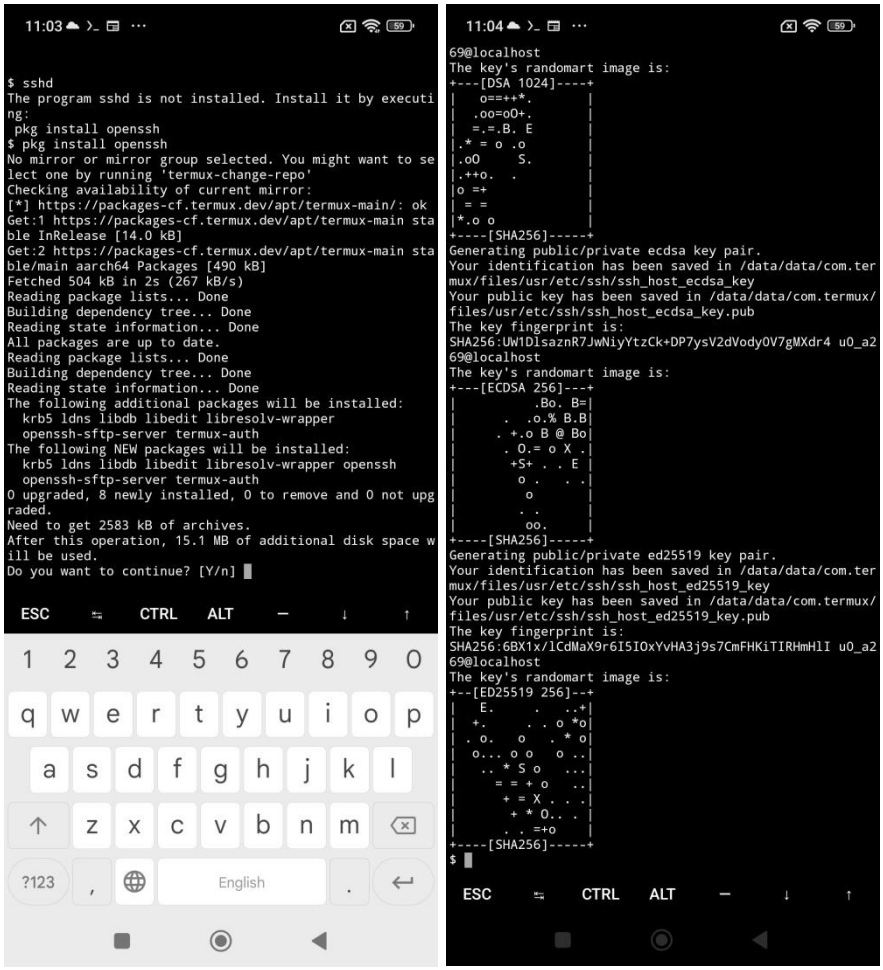


Рис 3. Етапи виконання команди «pkg install openssh» - встановлення служби SSH-сервера.

7. Перевірте доступність служби SSH. Для цього переконайтесь що на Вашому персональному комп’ютері або ноутбуку встановлено Telnet-клієнт, а якщо він відсутній - додайте його як компонент операційної системи. Після цього виконайте команду

```
telnet 192.168.0.145 8022
```

повинно бути виведене повідомлення, що з’єднання не може бути встановлене (Could not open connection to the host, on port 8022). Повторно виконайте на Android-пристрої команду

```
sshd
```

а на Вашому персональному комп’ютері або ноутбуку потворно виконайте команду

```
telnet 192.168.0.145 8022
```

Переконайтесь, що в лівому верхньому кутку Telnet-сеансу з’явилося запрошення від сервера «SSH-2.0-OpenSSH_8.4» та блимаючий в наступному ряду курсор. Якщо дійсно так - SSH-сервер налаштований. Якщо ні - зверніться до викладача.

9. На Android-пристрої виконайте команду

```
whoami
```

і дізнайтесь ім’я поточного користувача, наприклад **u0_a157**.

10. Встановіть для користувача u0_a157 (якого Ви дізнались з попередньої команди), новий пароль 123 за допомогою команди:

```
passwd u0_a157
```

11. На своєму персональному комп’ютері або ноутбуку виконайте команду

ping <IP-адреса Android-пристрою>

і переконайтесь в тому, що пристрій по мережі доступний.

12. Відкрийте на своєму комп’ютері програму putty або інший SSH-клієнт та виконайте віддалене приєднання до Android-пристрою на TCP-сокет

<IP-адреса Android-пристрою>:8022

Якщо SSH-сеанс успішно встановлено, розгорніть вікно командного рядка на весь екран. Усі наступні команди необхідно виконувати виключно всередині SSH-сеансу, а отже якщо встановити SSH-сеанс не вдається, самостійно не продовжуйте роботу, а зверніться до викладача.

13. Виконайте команду

```
pwd
```

і запишіть поточний каталог, в якому починає працювати SSH-користувач. Каталогом має бути //...../home

14. Виконайте команду

```
uname -a
```

і з’ясуйте як насправді називається операційна система, на якій Ви виконуєте лабораторну роботу. Наприклад, виконання цієї команди на популярній моделі смартфона Xiaomi Redmi 6A дає такий результат

```
Linux localhost 4.9.117+ #1 SMP PREEMPT Fri May 8 23:17:40 WIB
2020 armv7l Android
```

15. Виконайте команду

```
apt update
```

і уважно перевірте наявність рядків, які містять фразу «NO_PUBKEY», наприклад:

```
 ~ $ apt update
Hit:1 https://its-pointless.github.io/files/24 termux InRelease
Get:2 https://grimler.se/game-packages-24 games InRelease [9347 B]
Get:3 https://grimler.se/science-packages-24 science InRelease [9360 B]
Hit:4 https://termux.mentality.rip/termux-packages-24 stable InRelease
Err:2 https://grimler.se/game-packages-24 games InRelease
 The following signatures couldn't be verified because the public key is not
available: NO_PUBKEY B0076E490B71616B
Err:3 https://grimler.se/science-packages-24 science InRelease
 The following signatures couldn't be verified because the public key is not
available: NO_PUBKEY B0076E490B71616B
Fetched 18.7 kB in 6s (3131 B/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
89 packages can be upgraded. Run 'apt list --upgradable' to see them.
W: An error occurred during the signature verification. The repository is not
updated and the previous index files will be used. GPG error:
https://grimler.se/game-packages-24 games InRelease: The following signatures
couldn't be verified because the public key is not available: NO_PUBKEY
B0076E490B71616B
W: An error occurred during the signature verification. The repository is not
updated and the previous index files will be used. GPG error:
https://grimler.se/science-packages-24 science InRelease: The following
signatures couldn't be verified because the public key is not available:
NO_PUBKEY B0076E490B71616B
W: Failed to fetch https://grimler.se/game-packages-24/dists/games/InRelease The
following signatures couldn't be verified because the public key is not
available: NO_PUBKEY B0076E490B71616B
W: Failed to fetch https://grimler.se/science-packages-
24/dists/science/InRelease The following signatures couldn't be verified because
the public key is not available: NO_PUBKEY B0076E490B71616B
```

```
W: Some index files failed to download. They have been ignored, or old ones used instead.
```

звертіть увагу, що в консольному виводі наведеного прикладу шукана комбінація зустрічається 6 разів, але усі рази це вказує на відсутність лише одного ключа - B0076E490B71616B, який необхідно додати вручну. Для перевірки наявності і доступності цього публічного ключа спочатку виконайте ручну перевірку його наявності. Для цього відкрийте браузер і в двох сусідніх закладинках відкрийте два сайти:

```
https://keyserver.ubuntu.com
https://keys.openpgp.org
```

В пошукових рядках обох сайтів введіть шуканий номер публічного ключа B0076E490B71616B і натисніть кнопку SEARCH. Проаналізуйте отримані результати на обох сайтах. Для порівняння замініть будь-який один символ в номері ключа на інший, наприклад першу літеру B замініть на C і переконайтесь, що на обох сайтах ключ C0076E490B71616B не знайдено.

Далі потрібно виконати команду додавання публічного ключа в середовище Termux. Для цього є декілька команд, кожну з яких Вам слід виконати і перевірити її результат:

Команда 1:

```
gpg --keyserver keyserver.ubuntu.com --recv B0076E490B71616B
```

Результат:

```
gpg: /data/data/com.termux/files/home/.gnupg/trustdb.gpg: trustdb created
gpg: key B0076E490B71616B: public key "Henrik Grimler <henrik@grimler.se>" imported
gpg: Total number processed: 1
gpg: imported: 1
```

Команда 2:

gpg --keyserver hkp://keyserver.ubuntu.com:80 --recv B0076E490B71616B

Результат:

```
gpg: key B0076E490B71616B: "Henrik Grimler <henrik@grimler.se>" not changed
```

```
gpg: Total number processed: 1
gpg: unchanged: 1
```

16. Повторно виконайте команду

```
apt update
```

і проконтролюйте, щоб консольний вивід цієї команди більше не містив фразу «NO_PUBKEY». Команда може ставити інтерактивні запитання, наприклад, чи згодні Ви прийняти зміни, які відбулися в репозиторії. Звичайно, на такі запитання Y/N треба відповідати кнопкою Y. В результаті команда

```
apt update
```

повинна мати приблизно такий консольний вивід:

```
Fetched 517 kB in 18s (29.0 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
89 packages can be upgraded. Run 'apt list --upgradable'
to see them.
```

17. Як рекомендує вивід попередньої команди, виконайте команду

```
'apt list --upgradable
```

і отримайте перелік тих пакетів, які можна оновити запуском команди «apt upgrade», але не виконуйте цю команду. Встановіть текстовий редактор nano за допомогою команди, файловий менеджер Midnight Commnander та команду man (manual, інструкція по використанню команд) за допомогою відповідних команд:

```
apt install nano
apt install mc
apt install man
```

обидві програми запустіть, потім закрийте.

18. Перейдіть в домашній каталог /data/data/com.termux/files/home

за допомогою команди

cd ~

створіть новий підкаталог командою

`mkdir WORK`

і в ньому ще один підкаталог Lab1. В підкаталозі ~/WORK/Lab1 створіть новий текстовий файл і збережіть його під назвою sinus.c

Файл повинен бути такого змісту:

```
#include <stdio.h>
#include <math.h>
#define A 90
unsigned long int Points=500;
void p(int t);
int main()
{
 printf("==========START=========\n");
 double x=0.00000001,y; // x10
 for (unsigned long int i=0;i<Points*10000000;i++)
 {
 y=sin(i*x);
 if(i%10000000==0) p(y*A/2);

 }
 return 0;
}
void p(int t)
{
 unsigned int n;
 for(n=1;n<=A-1;n++)
 {
 if(n==t+A/2)
 {
 printf("X");
 }
```

```
 else
 {
 printf("-");
 }

 }
 printf("\n");
}
```

Збережіть змінений файл sinus.c

19. Виведіть перелік файлів каталогу ~/WORK/Lab1 за допомогою команди

ls -l

Повинен бути виведений один файл sinus.c

18. Виконайте команду для перевірки змісту файлу sinus.c . Для цього виконайте команду (якщо команда відсутня - виконайте «pkg install file»):

file sinus.c

Результатом повинно бути: «C source, ASCII text» (вихідний код мовою C).

20. Виконайте компіляцію файла sinus.c за допомогою команди

gcc -c sinus.c

Якщо виникло повідомлення «The program sshd is not installed», то виконайте команду встановлення компілятора gcc:

pkg install clang

Перевірте, що команда виконана успішно і після цього ще раз виконайте команду

gcc -c sinus.c

Успушне виконання команди - це відсутність будь-якого повідомлення. Якщо команда нічого не вивела, то виведіть в консоль перелік файлів потомного каталогу:

ls -l

Повинні бути виведені два файли sinus.c і sinus.o

21. Виконайте команду для перевірки змісту файлу sinus.o . Для цього виконайте команду:

file sinus.o

Результатом повинно бути: «ELF 32-bit LSB relocatable, ARM aarch64, version 1 (SYSV)» (файл формату ELF, Executable&Linkable Format, ARM64).

22. Виконайте компонування файлу sinus.o за допомогою команди

gcc -o SINUS sinus.o

Повинно виникнути помилка, що в програмі невідома функція sin(). Для успішної компіляції виконайте команду з використанням бібліотеки math:

gcc -o SINUS sinus.o -lm

і ця команда повинна бути виконана успішно.

23. Повторно виведіть перелік файлів командою

ls -l

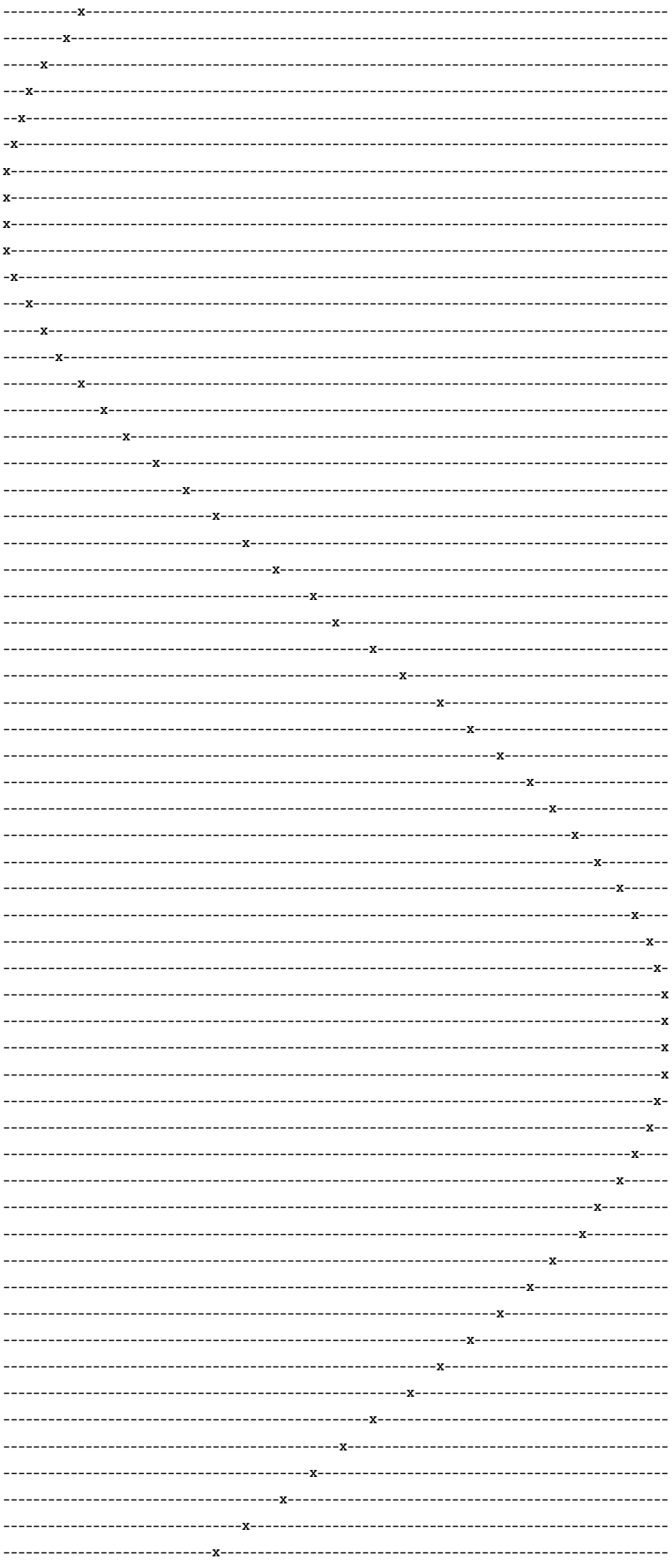
Повинні бути виведені три файли sinus.c, sinus.o і SINUS, причому останній файл повинен бути зеленого кольору, тобто з правом виконання (X, eXecute).

24. Виконайте програму SINUS за допомогою команди

./SINUS

Результатом виконання програми повинен бути наступний консольний вивід, в якому чітко видно графік функції SIN(X), тобто графік «сінуоїди»:

```
--------------------------------x-----------------------------------------------
----------------------------x-------------------------------------------------
------------------------x-----------------------------------------------------
--------------------x---------------------------------------------------------
----------------x-------------------------------------------------------------
-------------x----------------------------------------------------------------
```



----------x----------
----------x----------
----------x----------
----------x----------
----------x----------

25. Зробіть копію файла sinus.c за допомогою команди

cp sinus.c sinus2.c

26. Проявіть свою математичну кмітливість і винахідливість: змініть в програмі sinus2.c функцію sin(x) на іншу періодичну функцію таким чином, щоб її графік якісно відрізнявся від того, що у програмі sinus.c . Така функція повинна бути у кожного студента РІЗНА. Продемонструйте це викладачеві, якщо необхідно дайте відповіді на його додаткові запитання і отримайте оцінку.

Запитання для закріплення матеріалу лабораторної роботи:

1. До групи яких операційних систем відноситься операційна система Android?

2. Які Ви знаєте компілятори мов програмування С для операційної системи Android?

3. Якими текстовими редакторами найбільш зручно редагувати С-програми?

4. Перелічить елементарні періодичні функції мови програмування С?

ЛАБОРАТОРНА РОБОТА №8.

«Налаштування середовища програмування мовою Assembler в операційній системі Android мобільного пристрою»

Тема: Налаштування набору компіляторів GCC та дебагера GDB в операційній системі Android.

Мета: Навчитися самостійно встановлювати на мобільний пристрій середовище Termux, службу SSH (Secure SHell), набір компіляторів GCC (GNU Compiler Collection) та дебагера GDB.

Теоретичні відомості

1. Набір компіляторів GCC (GNU Compiler Collection)

Компілятор GCC розроблений некомерційною організацію Фонд Вільних Програм (Free Software Foundation, FSF) і одвічно підтримував лише одну мову програмування — C. На цей час компілятор підтримує різні мови програмування, апаратні архітектури, операційні системи. Це стандартний компілятор для Unix-подібних операційних систем, і ряду комерційних операційних систем на основі Unix, зокрема Apple MacOS. В методичному посібнику планується використання двох мов програмування, що підтримуються набором компіляторів GCC - мова C та мова Assembler під архітектуру команд ARM32/ARM64.

2. Дебагер GDB (GNU DeBugger)

Дебагер GDB - універсальний дебагер проекту GNU, яких дає можливість відлагоджувати програми, написані різними мовами програмування, якщо компілятор відповідної мови підтримує спеціальний режим компіляції, додаючи до скомпільованої програми додаткову службову інформацію, необхідну для покрокового виконання програми дебагером GDB.

Хід роботи:

1. Відкрийте на своєму комп’ютері програму putty або інший SSH-клієнт та виконайте віддалене приєднання до Android-пристрою на TCP-сокет

<IP-адреса Android-пристрою>:8022

Якщо SSH-сеанс успішно встановлено, розгорніть вікно командного рядка на весь екран. Усі наступні команди необхідно виконувати виключно всередині SSH-сеансу, а отже якщо встановити SSH-сеанс не вдається, самостійно не продовжуйте роботу, а зверніться до викладача або асистента.

2. Перейдіть в домашній каталог /data/data/com.termux/files/home

за допомогою команди

cd ~

і переконайтесь, що в цьому каталозі існує підкаталог WORK за допомогою команди

ls -l

якщо цей каталог існує WORK - зайдіть у нього командою

cd WORK

і переконайтесь, що у ньому міститься інший підкаталог Lab1. Не заходьте в нього, а створіть поряд з ним новий підкаталог Lab2. Зайдіть в цей каталог ~/WORK/Lab2 і створіть за допомогою редактора nano або іншого редактора новий текстовий файл і збережіть його під назвою 1.s

Розрізнити операційні системи ARM32/ARM64 можна за допомогою команди «uname -a». Якщо Ви працюєте на платформі ARM32 (для архітектур ARM64 або Apple M1 дивіться нижче), файл 1.s повинен бути такого змісту:

```
.data

/* Data segment: define our message string and calculate its length. */
msg:
 .ascii "Hello, World of ARM32!\n"
len = . - msg
```

```
.text

/* Our application's entry point. */
.globl _start
_start:
 /* syscall write(int fd, const void *buf, size_t count) */
 mov %r0, $1 /* fd := STDOUT_FILENO */
 ldr %r1, =msg /* buf := msg */
 ldr %r2, =len /* count := len */
 mov %r7, $4 /* write is syscall #4 */
 swi $0 /* invoke syscall */

 /* syscall exit(int status) */
 mov %r0, $5 /* status := 0 */
 mov %r7, $1 /* exit is syscall #1 */
 swi $0 /* invoke syscall */
```

3. Проаналізуйте зміст цього файлу і визначте для себе скільки приблизно відсотків інформації у ньому є для вас зрозумілою. Ця програма виводить в консоль лише один рядок «Hello, World of ARM32!» з символом переходу на новий рядок і завершує свою роботу. Мовою С це відповідає одному оператору printf(«Hello, World of ARM32!\n»);

4. Легко бачити, що програма починається з розміщення в оперативній пам’яті статичного рядка і збереження на нього адреси-вказівника msg. Наступний оператор `len = . - msg` виконує обчислення різниці в байтах між поточною адресою оператора (.) і адреси msg. Очевидно, що ця різниця адрес є довжина рядка «**Hello, World of ARM32**!\n».

Далі програма складається з двох окремих частин. Перша частина виконує системний виклик write, що виводить в консоль рядок «Hello, ARM32!\n». Друга частина програми виконує системний виклик exit, яка коректно закриває роботу програми і звільнює усі ресурси, які їй виділила операційна система. При цьому діагностичний код повернення програми записується в регістр r0 і в перспективі може бути також виведений в консоль. Закрийте файл 1.s .

5. Якщо Ви працюєте в операційній системі розрядності ARM64, файл 1.s повинен бути такого змісту:

```
.data
/* Data segment: define our message string and calculate its length. */
msg:
 .ascii "Hello, world of ARM64!\n"
len = . - msg
```

```
.text

/* Our application's entry point. */
.globl _start
_start:
 /* syscall write(int fd, const void *buf, size_t count) */
 mov x0, #1 /* fd := STDOUT_FILENO */
 ldr x1, =msg /* buf := msg */
 ldr x2, =len /* count := len */
 mov w8, #64 /* write is syscall #64 */
 svc 0 /* invoke syscall */

 /* syscall exit(int status) */
 mov x0, #5 /* status := 5 */
 mov w8, #93 /* exit is syscall #93 */
 svc 0 /* invoke syscall */
```

Порівняйте результат вихідного коду платформи ARM64 з вихідним кодом платформи ARM32, знайдіть відмінності.

6. Переконайтесь ще раз, що поточним каталогом є ~/WORK/Lab2 і в цьому каталозі існує файл 1.s . Якщо це дійсно так, виконайте компіляцію файла 1.s за допомогою команди:

as -o 1.o 1.s

В цій команді ключ -o означає формат вихідного файла 1.o . Система може підказати виконати команду «**pkg install binutils**», щоб встановити компілятор as та компоновщик ld.

7. Для платформи ARM32 виконайте компоновку файла 1.o за допомогою команди:

ld -o exec32 1.o

Навпаки, в цій команді ключ -o означає формат вхідного файла 1.o . Для платформи ARM64 замість цієї команди виконайте команду

ld -o exec64 1.o

і враховуйте це при виконанні усіх наступних команд.

8. Виконайте програму за допомогою команди

./exec32

і переконайтесь, що вона працює правильно, тобто виводить заданий рядок.

9. Виконайте програму за допомогою іншої команди, яка дає можливість додатково вивести в консоль код повернення програми, що присвоюється як прямий операнд регістру r0 і в даному прикладі має значення 5:

./1 ;echo $?

в результаті виконання програми повинно бути виведено два рядочки:

Hello, World of ARM32!
5

10. Замініть в програмі число 5, яке присвоюється регістру r0 на інше, двохзначне число. Збережіть файл, скомпілюйте програму і повторно її виконайте. Переконайтесь, що замість числа 5 програма виводить нове двохзначне число.

11. Виконайте команду

gdb

і проаналізуйте результат. Якщо буде написано «The program gdb is not installed. Install it by executing: **pkg install gdb**», то це означає, що програма не встановлена і тому виконайте цю команду «pkg install gdb». Після виконання встановлення програми gdb повторно виконайте команду gdb і перевірте, що в консолі надруковано така інформація:

GNU gdb (GDB) 13.2
Copyright (C) 2023 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "aarch64-linux-android".
Type "show configuration" for configuration details.

For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word".
(gdb)_
Виконайте внутрішню команду quit і поверніться назад до операційної системи.
12. Виконайте команду для аналізу скомпільованої програми exec32:

gdb exec32

Результатом повинен бути такий вивід:

```
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and
redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "arm-linux-androideabi".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources
online at:
 <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to
"word"...
Reading symbols from exec32...
(No debugging symbols found in exec32)
```

```
(gdb)_
```

Зверніть увагу, що при виконанні останньої команди з’явилось два нових рядки, що виділені **жирним шрифтом**. Суть першого рядка у тому, що програма почала спробу аналізу файла цього повідомлення у тому, що при спробі читання файлу exec32 у ньому не знайдено «debugging symbols», тобто додаткова службова інформація для дебагера, а тому подальший аналіз файлу exec32 дебагером gdb неможливий. Виконайте внутрішню команду quit, таким чином закрийте дебагер і поверніться назад до системи.

13. Очевидно, що якщо програмі gdb не вдалося знайти додаткову службову інформацію в попередній версії цього файлу exec32, то .Щоб додати до файлу 1, яку Виконайте таку команду

```
ls -l
```

для того, щоб визначити і запам’ятайте розміри двох файлів 1.o і exec32.

14. Щоб додати Виконайте наступні три команди:

```
as -g -o 1.o 1.s
ld -o 1 1.o
ls -l
```

Порівняйте нові розміри файлів 1.o і exec32 з тими розмірами які визначені в пункті 13. Розмір файлу 1.o повинен стати збільшеним приблизно на 100%, а розмір файлу exec32 повинен стати збільшеним приблизно на 60%.

15. Повторно виконайте команду

```
gdb exec32
```

В результаті має бути запущений дебагер, який повинен успішно прочитати додаткову службові інформацію і останнім рядком вивести повідомлення «**`Reading symbols from exec32...`**». З цього моменту інженер-розробник може виконувати лише внутнішні команди дебагера gdb (https://ccrma.stanford.edu/~jos/stkintro/Useful_commands_gdb.html

https://sourceware.org/gdb/onlinedocs/gdb/Command-and-Variable-Index.html).
Виконайте команду

```
i b
```

(**i**nfo of **b**reakpoints), ця команда виводить перелік точок переходу програми з автоматичного в покроковий режим (брекпойнтів). Результатом команди повинен бути виведений повний перелік брекпойнтів або повідомлення про їх відсутність «No breakpoints or watchpoints.». Виконайте команду

b 1

і проконтролюйте, що результатом команди є виведення повідомлення «Breakpoint 1 at 0x10074: file 1.s, line 14.»

16. Повторно виконайте команду

i b

і переконайтесь, що буде виведена приблизно така таблиця:

```
Num Type Disp Enb Address What
1 breakpoint keep y 0x00010074 1.s:14
```

17. Виконайте команду

step

яка виконує наступну команду. Повинно з'явитись повідомлення «The program is not being run.». Це одначає, що команда step, а також значна група інших внутрішніх команд дебагера gdb доступна тільки після запуску програми на виконання. Аналогічно, на команду «i r» буде відповідь «The program has no registers now.»

18. Виконайте команду

list

і проаналізуйте результат. Зверніть увагу на призначення номера кожному рядку програми. Знаючи ці номери, можна виконати команду, наприклад «list 3,10», яка виведе підмножину рядків з номерами від 3 по 10.

19. Виконайте команду

run

і перевірте, що програма запущена на виконання, а в консоль виведено наступну інформацію:

```
Starting program: /data/data/com.termux/files/home/micro/LAB6/1

Breakpoint 1, _start () at 1.s:14
14 mov %r0, $1 /* fd := STDOUT_FILENO */
```

Цей вивід означає, що при виконанні наступної команди step буде виконана команда «mov %r0, $1», яка знаходиться на 14 рядку вихідного текстового файлу 1.s .

20. Повторно виконайте команду

list

і проаналізуйте результат. Зверніть увагу, що після запуску програми виводяться рядки вже з іншими номерами, починаючи з того, на який встановлено брекпойнт в пункті 16 і який виведений при виконанні програми run в пункті 19.

21. Виконайте команду

bt

(**b**ack**t**race), яка в будь-який момент роботи дебагера gdb може підказати розробнику інформацію про номер рядка, команда на якому буде виконана наступною при виконанні команд типу step/next/nexti.

22. Виконайте команду

info os cpus

Результатом команди повинно бути виведення таблиці такого типу:

processor BogoMIPS Features CPU implementer CPU architecture CPU variant CPU part CPU revision

0 38.40 fp 0x41 8 0x2 0xd05 0

1 38.40 fp 0x41 8 0x2 0xd05 0

2 38.40 fp 0x41 8 0x2 0xd05 0

3 38.40 fp 0x41 8 0x2 0xd05 0

4 38.40 fp 0x41 8 0x2 0xd05 0

```
5 38.40 fp 0x41 8 0x2 0xd05 0
6 38.40 fp 0x41 8 0x1 0xd41 1
7 38.40 fp 0x41 8 0x1 0xd41 1
```

З виведеної таблиці порахуйте кількість ядер процесора (перша колонка), визначте версію ARM-архітектури (колонка **CPU architecture**), оцініть швидкодію (колонка **BogoMIPS**).

23. Виконайте команду

i r

(**i**nfo of **r**egisters), яка виведе перелік регістрів процесора. Якщо операційна система Android має архітектуру ARM32, то перелік регістрів виглядає наступним чином:

```
r0 0x0 0
r1 0x0 0
r2 0x0 0
r3 0x0 0
r4 0x0 0
r5 0x0 0
r6 0x0 0
r7 0x0 0
r8 0x0 0
r9 0x0 0
r10 0x0 0
r11 0x0 0
r12 0x0 0
sp 0xbefff820 0xbefff820
lr 0x0 0
pc 0x10074 0x10074 <_start>
cpsr 0x10 16
fpscr 0x0 0
```

Навпаки, якщо операційна система Android має архітектуру ARM64, то перелік регістрів виглядає наступним чином:

```
x0 0x0 0
x1 0x0 0
x2 0x0 0
x3 0x0 0
x4 0x0 0
x5 0x0 0
x6 0x0 0
x7 0x0 0
x8 0x0 0
x9 0x0 0
x10 0x0 0
x11 0x0 0
x12 0x0 0
x13 0x0 0
x14 0x0 0
x15 0x0 0
x16 0x0 0
x17 0x0 0
x18 0x0 0
x19 0x0 0
x20 0x0 0
x21 0x0 0
x22 0x0 0
x23 0x0 0
x24 0x0 0
x25 0x0 0
x26 0x0 0
x27 0x0 0
x28 0x0 0
x29 0x0 0
x30 0x0 0
sp 0x7fffffef70 0x7fffffef70
pc 0x4000b0 0x4000b0 <_start>
cpsr 0x1000 [ EL=0 BTYPE=0 SSBS ]
```

```
fpsr 0x0 [ ]
fpcr 0x0 [ Len=0 Stride=0 RMode=0 ]
tpidr 0x0 0x0
```

Зверніть увагу, що регістри середовища ARM32 мають розмір 4 байти (32 біта), а регістри середовища ARM64 мають розмір 5 байтів (40 бітів). Це означає, що якщо виділяти на збереження адреси оперативної пам'яті один регістр, то процесор мобільного пристрою з операційною системою ARM32 може адресувати максимум 4ГБ оперативної пам'яті, а мобільного пристрою ARM64 — максимум 1000ГБ (1ТБ). На початку програми перед виконанням першої команди усі регістри загального призначення мають значення «0».

24. Виконайте команду step, яка виконує першу команду ARM32-програми

«mov %r0, $1» або першу команду ARM64-програми «mov x0, #1» і після неї послідовно виконайте дві команди

i r

i r r0 r1 r2 (ARM32) / i r x0 x1 x2 (ARM64)

Порівняйте результат обох команд. Проконтролюйте, щоб в результаті виконання цієї команди регістр r0/x0 вже мав значення не 0, а 1. Якщо це не так, або Ви не можете це перевірити - продовжувати не можна, попросіть допомогу викладача або асистента.

25. Повторіть пункт 20, тобто виконайте послідовність внутрішніх команд

step

i r

до того часу, поки не будуть успішно виконані усі команди програми, тобто поки програма не буде виконана до кінця. Перед виконання кожної команди спробуйте спрогнозувати який регістр змінить своє значення, а після виконання цієї ж команди перевірте чи дійсно ця прогнозована зміна відбулась. Окрім команди step, також є команди next і nexti, самостійно вивчіть їх призначення. (Документація по дебагеру GDB: https://web.mit.edu/gnu/doc/html/gdb_1.html)

26. Отримайте від викладача індивідуальну розрахункову функцію f(x,y). Виконайте команду

cp 1.s 2.s

яка, створить новий файл 2.s . Використовуючи цей файл як шаблон, напишіть і виконайте програму, в якій першими двома командами буде присвоєння двом довільним регістрам загального призначення значень аргументів x та y, а сама програма буде виводити в консоль розраховане значення функції f(x,y). За допомогою калькулятора переконайтесь на 100%, що функція f(x,y) розраховується правильно при різних значеннях аргументах.

В перспективі (в наступних лабораторних роботах) ця програма може бути також використана як шаблон для того, щоб написати іншу, більш складну програму, в якій буде здійснено виклик функції f(x,y) кілька разів з різними аргументами. Тому надійно збережіть вихідний код Вашої програми в надійному місці, інакше у випадку втрати цього коду для виконання наступної лабораторної роботи Вам потрібно буде спочатку повторно виконати поточну лабораторну роботу.

Продемонструйте викладачеві роботу програми в автоматичному або в покроковому режимі, якщо необхідно дайте відповіді на його додаткові запитання і отримайте оцінку.

Запитання для закріплення матеріалу лабораторної роботи:

1. Яке призначення дебагера gdb? Скільки приблизно років цій програмі? Чи існують для цієї програми плагіни?

2. Для чого в програмі мовою Assembler, яка виводить заданий рядок тексту, необхідно аж два рази виконувати системні виклики?

3. Чи можна написати Assembler-програму, яка буде мати однаковий вихідний код і може бути виконана в операційних системах платформи ARM32 і платформи ARM64?

4. В чому відмінності між внушрішніми командами дебагера step, next і nexti?

Додаток 1. Програма, що виводить в консоль текстове повідомлення для операційних систем на процесорі Apple M1 архітертури ARMv8.5-A (http://main.lv/writeup/arm64_assembly_hello_world.md):

```
.global _start // Provide program starting address to linker
.align 2 // Make sure everything is aligned properly

/* syscall write(int fd, const void *buf, size_t count) */
_start:
 mov X0, #1 // 1 = StdOut
 adr X1, helloworld // string to print
 mov X2, helloworld_len // length of our string
 mov X16, #4 // Unix write system call
 svc #0x80 // Call kernel to output the string

/* syscall exit(int status) */
 mov X0, #0 // Use 0 return code
 mov X16, #1 // System call number 1 terminates this program
 svc #0x80 // Call kernel to terminate the program

helloworld: .ascii "Hello, world of ARM64 on Apple!\n"
helloworld_len = . - helloworld
```

Перед компіляцією повинен бути встановлений пакет xcode tools. Для компіляції і компоновки потрібно використовувати такі команди:

```
as -o hello.o hello.s
ld -macosx_version_min 11.0.0 -o hello hello.o -lSystem -syslibroot `xcrun -sdk macosx --show-sdk-path` -e _start -arch arm64
```

Додаток А

Код прошивки мікроконтролера

```
$regfile = "ATtiny44.dat" 'вказуємо наш мікроконтролер
$crystal = 1000000 'встановлюємо тактову честоту

Config Porta = Output 'конфігуруємо всі ніжки порту А як виходи

Dim A As String * 4
Dim G As String * 2
Dim B As Byte
Dim C As Byte

Open "comb.0:9600,8,n,1" For Input As #1 'кофігуруємо ніжки, до яких під'єднані
Open "comb.1:9600,8,n,1" For Output As #2 'контакти послідовного порта

Do 'робочий цикл мікроконтролера
Input #1 , A 'зчитуємо строку з послідовного порта у змінну А
G = Mid(a , 1 , 2) 'записуємо перші 2 символи строки А у змінну G
B = Hexval(g) 'перетворюємо шіснадцяткове значення змінної G у байт B
G = Mid(a , 3 , 2) 'записуємо наступні 2 символи строки А у змінну G
C = Hexval(g) 'перетворюємо шіснадцяткове значення змінної G у байт C
Shiftout Porta.1 , Porta.0 , C , 1 'записуємо байт С на перший резістр
Shiftout Porta.1 , Porta.0 , B , 1 'записуємо байт B на перший резістр, байт C
перепишеться на другий регістр
Pulseout Porta , 2 , 1 'синхронізуємо виходи регістрів
Loop
End
```

Код C# програми

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;//підключаємо цей простір імен для роботи з послідовним
портом

namespace WindowsFormsApplication1
{
```

```
 public partial class Form1 : Form
 {
 SerialPort port;//послідовний порт, з яким взаємодіятиме дана програма
 string[] M;//масив зі значеннями, що відповідають числам на семисегментному індикаторі
 public Form1()
 {
 InitializeComponent();
 M = new string[] {"3F", "06", "5B", "4F", "66", "6D", "7D", "07", "7F", "6F"};
 try
 {
 port = new SerialPort("COM1", 9600, Parity.None, 8, StopBits.One);//ініціалізуємо послідовний порт
 port.Open();//відкриваємо послідовний порт
 port.NewLine = "\r";//встановлюємо символ перенесення строки
 }
 catch (Exception)
 {
 button1.Enabled = false;
 textBox1.Enabled = false;
 MessageBox.Show("Порт не підключений");
 }
 }

 private void button1_Click(object sender, EventArgs e)
 {
 string s1, s2;
 int a1, a2;
 int a = 0;
 if (textBox1.Text.Length > 2) MessageBox.Show("Невірний формат");
 else
 {
 if (int.TryParse(textBox1.Text, out a))
 {
 a1 = a / 10;
 a2 = a % 10;
 s1 = M[a1];
 s2 = M[a2];
 port.WriteLine(s1 + s2);//надсилаємо на послідовний порт значення, які слід відобразити на семисегментних індикаторах
 }
 else MessageBox.Show("Невірний формат");
 }
 }
```

```
 private void Form1_FormClosed(object sender, FormClosedEventArgs e)
 {
 port.Close();//закриваємо послідовний порт
 }
 }
}
```

Додаток Б

Код прошивки мікроконтролера

```
$regfile = "ATtiny44.dat" 'вказуємо наш мікроконтролер
$crystal = 1000000 'встановлюємо тактову честоту

Config 1wire = Porta.1 'конфігуруємо ніжку на роботу за протоколом 1-wire
Config Porta.0 = Output 'конфігуруємо ніжку як вхід
Config Pinb.2 = Input 'конфігуруємо ніжку як вихід
Config Int0 = Rising 'конфігуруємо спрацювання переривання INT0 по передньому фронту

Dim S As String * 4
Dim A As Byte
Dim B As Byte
Dim C As Byte
Dim D As String * 1

Open "comb.0:9600,8,n,1" For Input As #1 'кофігуруємо ніжки, до яких під'єднані
Open "comb.1:9600,8,n,1" For Output As #2 'контакти послідовного порта

1wreset 'виконуємо скидання лінії 1-wire

Enable Interrupts 'вмикаємо переривання
Enable Int0 'вмикаємо переривання INT0
On Int0 Rts 'додаємо обробник переривання

Do
1wwrite &HCC 'вказуємо, що наступна команда адресована всім пристроям на лінії 1-wire
1wwrite &H44 'команда конвертації температури в бінарний вигляд
Waitms 750 'чекаємо 750 мс
1wreset 'виконуємо скидання лінії 1-wire
1wwrite &HCC 'вказуємо, що наступна команда адресована всім пристроям на лінії 1-wire
1wwrite &HBE 'команда читання пам'яті датчика
A = 1wread() 'зчитуємо два байти з лінії 1-wire у змінні A та B
B = 1wread()
S = Hex(b) 'перетворюємо байти A та B у строку в шістнадцятковому вигляді
S = S + Hex(a)
1wreset 'виконуємо скидання лінії 1-wire
Loop

Rts: 'обробник переривання INT0
```

```
Porta.0 = 1 'встановлюємо сигнал CTS для хоста
Input #1 , D 'зчитуємо строку з послідовного порта у змінну A
Porta.0 = 0 'прибираємо сигнал CTS для хоста
C = Hexval(d) 'перетворюємо шіснадцяткове значення змінної D у байт C
If C = 0 Then 'якщо отримали 0 то
Print #2 , S 'надсилаємо на послідовний порт змінну S
End If
Return 'певертаємось до основного циклу

End
```

Код C# програми

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;//підключаємо цей простір імен для роботи з послідовним портом

namespace WindowsFormsApplication1
{
 public partial class Form1 : Form
 {
 SerialPort port;//послідовний порт, з яким взаємодіятиме дана програма
 public Form1()
 {
 InitializeComponent();
 try
 {
 port = new SerialPort("COM1", 9600, Parity.None, 8, StopBits.One);//ініціалізуємо послідовний порт
 port.Open();//відкриваємо послідовний порт
 port.NewLine = "\r";//встановлюємо символ перенесення строки
 }
 catch (Exception)
 {
 button1.Enabled = false;
 MessageBox.Show("Порт не підключений");
 }
 }
```

```
 private void button1_Click(object sender, EventArgs e)
 {
 string s = "";
 port.RtsEnable = true;//інформуємо мікроконтролер про наявність даних для передачі
 while (!port.CtsHolding) System.Threading.Thread.Sleep(1);//очікуємо готовність мікроконтролера прийняти дані
 port.WriteLine("0");//передаємо команду для зчитування температури
 port.RtsEnable = false;//більше немає, що передавати
 s = port.ReadLine();//зчитуємо температуру
 s = s.Substring(s.Length - 4);
 Int16 c = Convert.ToInt16(s, 16);
 label1.Text = ((double)c / 16).ToString() + "°C";
 }

 private void Form1_FormClosed(object sender, FormClosedEventArgs e)
 {
 port.Close();//закриваємо послідовний порт
 }
 }
}
```

Додаток В

Код прошивки мікроконтролера

```
$regfile = "ATtiny44.dat" 'вказуємо наш мікроконтролер
$crystal = 1000000 'встановлюємо тактову честоту

Config 1wire = Porta.0 'конфігуруємо ніжку на роботу за протоколом 1-wire

Dim A As String * 5
Dim B As Byte
Dim C As Byte
Dim D As Byte
Dim G As String * 2

Open "comb.0:9600,8,n,1" For Input As #1 'кофігуруємо ніжки, до яких під'єднані
Open "comb.1:9600,8,n,1" For Output As #2 'контакти послідовного порта

1wreset 'виконуємо скидання лінії 1-wire

Do
Input #1 , A 'зчитуємо строку з послідовного порта у змінну A
G = Mid(a , 1 , 1) 'у змінну G записуємо перший символ строки A
B = Hexval(g) 'конвертуємо шіснадцяткове значення G у байт B
If B = 0 Then 'якщо 0 то копіюємо дані з постійної пам'яті до скретч-пам'яті
1wwrite &HCC 'вказуємо, що наступна команда адресована всім пристроям на лінії 1-
wire
1wwrite &HF0 'команда копіювання даних з постійної пам'яті до скретч-пам'ят
1wwrite &H00 'можна вказати будь-яку адресу
Else
If B = 1 Then 'якщо 1 то копіюємо дані зі скретч-пам'яті до постійної пам'яті
1wwrite &HCC 'вказуємо, що наступна команда адресована всім пристроям на лінії 1-
wire
1wwrite &H55 'команда копіювання даних зі скретч-пам'яті до постійної пам'яті
1wwrite &HA5 'надсилаємо ключ перевірки
Porta.0 = 1 'встановлюємо лінію даних в логічну одиницю на 10мс
Waitms 10
Else
If B = 2 Then 'якщо 2 то зчитуємо значення певного байту
G = Mid(a , 2 , 2)
D = Hexval(g) 'в змінну D зчитуємо адресу байта який хочемо зчитати
1wwrite &HCC 'вказуємо, що наступна команда адресована всім пристроям на лінії 1-
wire
1wwrite &HAA 'команда читання даних зі скретч-пам'яті
1wwrite D 'надсилаємо адресу байту
```

```
C = 1wread() 'зчитуємо значення у змінну C
Print #2 , Hex(c) 'надсилаємо на послідовний байт C у шістнадцятковому вигляді
Else
If B = 3 Then 'якщо 3 то встановлюємо значення певного байту
G = Mid(a , 2 , 2)
D = Hexval(g) 'в змінну D зчитуємо адресу байта який хочемо записати
G = Mid(a , 4 , 2)
C = Hexval(g) 'в змінну C зчитуємо значення байта який хочемо записати
1wwrite &HCC 'вказуємо, що наступна команда адресована всім пристроям на лінії 1-wire
1wwrite &H0F 'команда запису даних до скретч-пам'яті
1wwrite D 'надсилаємо адресу байту
1wwrite C 'надсилаємо значення байту
End If
End If
End If
End If
1wreset 'виконуємо скидання лінії 1-wire
Loop
End
```

Код C# програми

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;//підключаємо цей простір імен для роботи з послідовним портом

namespace WindowsFormsApplication1
{
 public partial class Form1 : Form
 {
 SerialPort port;//послідовний порт, з яким взаємодіятиме дана програма
 public Form1()
 {
 InitializeComponent();
 try
 {
```

```
 port = new SerialPort("COM1", 9600, Parity.None, 8,
StopBits.One);//ініціалізуємо послідовний порт
 port.Open();//відкриваємо послідовний порт
 port.NewLine = "\r";//встановлюємо символ перенесення строки
 }
 catch (Exception)
 {
 button1.Enabled = false;
 button2.Enabled = false;
 button3.Enabled = false;
 button4.Enabled = false;
 textBox1.Enabled = false;
 textBox2.Enabled = false;
 MessageBox.Show("Порт не підключений");
 }
 }

 private void button1_Click(object sender, EventArgs e)
 {
 port.WriteLine("0");//надсилаємо команду копіювання даних з постійної пам'яті
до скретч пам'яті
 }

 private void button2_Click(object sender, EventArgs e)
 {
 port.WriteLine("1");//надсилаємо команду копіювання даних з скретч пам'яті до
постійної пам'яті
 }

 private void button3_Click(object sender, EventArgs e)
 {
 if (CheckHex(textBox1.Text))
 {
 port.WriteLine("2" + textBox1.Text);//надсилаємо команду читання байту даних за
адресою
 }
 string s = port.ReadLine();//зчитуємо байт даних
 label1.Text = s.Substring(s.Length - 2);
 }

 private void button4_Click(object sender, EventArgs e)
 {
 if (CheckHex(textBox1.Text) && CheckHex(textBox2.Text))
 {
```

```
 port.WriteLine("3" + textBox1.Text + textBox2.Text);//надсилаємо команду запису
байту даних за адресою
 }
 }

 private void Form1_FormClosed(object sender, FormClosedEventArgs e)
 {
 port.Close();//закриваємо послідовний порт
 }

 private bool CheckHex(string text)
 {
 if (text.Length != 2) return false;
 else
 {
 int i = 0;
 try
 {
 i = Convert.ToInt32(text, 16);
 }
 catch(Exception e)
 {
 MessageBox.Show("Невірний формат");
 return false;
 }
 return true;
 }
 }
 }
}
```

Додаток Г

Код прошивки мікроконтролера

```
$regfile = "ATtiny44.dat" 'вказуємо наш мікроконтролер
$crystal = 1000000 'встановлюємо тактову честоту

Config Porta = Output 'конфігуруємо всі ніжки порту A як виходи
Config Pinb.2 = Input 'конфігуруємо ніжку як вхід
Config Int0 = Rising 'конфігуруємо спрацювання переривання INT0 по передньому фронту

Dim A As Byte
Dim E As Byte
Dim B(8) As Byte
Dim C As Byte 'в цьому масиві зберігається наше зображення
Dim D As Byte
Dim S As String * 16
Dim G As String * 2

Open "comb.0:9600,8,n,1" For Input As #1 'кофігуруємо ніжки, до яких під'єднані
Open "comb.1:9600,8,n,1" For Output As #2 'контакти послідовного порта

C = &B11111110 'значення змінної вказує на те, який рядок світиться

Enable Interrupts 'вмикаємо переривання
Enable Int0 'вмикаємо переривання INT0
On Int0 Rts 'додаємо обробник переривання

Do
For A = 1 To 8 'обходимо всі елементи масиву та виводимо їх на матрицю
Shiftout Porta.1 , Porta.3 , C , 1
Shiftout Porta.1 , Porta.3 , B(a) , 1
Pulseout Porta , 2 , 1
Rotate C , Left 'циклічний зсув змінної C вліво
Next
Loop

Rts: 'обробник переривання INT0
Porta.0 = 1 'встановлюємо сигнал CTS для хоста
Input #1 , S 'зчитуємо строку з послідовного порта у змінну S
Porta.0 = 0 'прибираємо сигнал CTS для хоста
For E = 1 To 8 'записуємо нові значення до масиву з зображенням
```

```
D = 2 * E
D = D - 1
G = Mid(s , D , 2)
B(e) = Hexval(g)
Next
Return

End
```

Код C# застосування

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO.Ports;//підключаємо цей простір імен для роботи з послідовним портом

namespace WindowsFormsApplication1
{
 public partial class Form1 : Form
 {
 SerialPort port;//послідовний порт, з яким взаємодіятиме дана програма
 bool[,] matrix;
 Graphics g;
 int a;
 public Form1()
 {
 InitializeComponent();
 try
 {
 port = new SerialPort("COM1", 9600, Parity.None, 8, StopBits.One);//ініціалізуємо послідовний порт
 port.Open();//відкриваємо послідовний порт
 port.NewLine = "\r";//встановлюємо символ перенесення строки
 }
 catch (Exception)
 {
 button1.Enabled = false;
 button2.Enabled = false;
 MessageBox.Show("Порт не підключений");
```

```
 }
 g = panel1.CreateGraphics();
 matrix = new bool[8, 8];
 a = panel1.Width / 8;
 for(int i = 0; i < 8; i++)
 {
 for(int j = 0; j < 8; j++)
 {
 matrix[i, j] = false;
 }
 }
 }

 private void button1_Click(object sender, EventArgs e)
 {
 string s = "";
 string s1;
 int b;
 for(int j = 0; j < 8; j++)
 {
 b = 0;
 for (int i = 0; i < 8; i++)
 {
 if (matrix[i, j]) b += (int)Math.Pow(2, 7 - i);
 }
 s1 = b.ToString("X");
 if (s1.Length < 2) s1 = "0" + s1;
 s += s1;
 }
 port.RtsEnable = true;//інформуємо мікроконтролер про наявність даних для
передачі
 while (!port.CtsHolding) System.Threading.Thread.Sleep(1);//очікуємо готовність
мікроконтролера прийняти дані
 port.WriteLine(s);//передаємо зображення 8x8 на послідовний порт
 port.RtsEnable = false;//більше немає, що передавати
 }

 private void panel1_Click(object sender, EventArgs e)
 {
 MouseEventArgs es = e as MouseEventArgs;
 int x = es.X / a;
 int y = es.Y / a;
 matrix[x, y] = !matrix[x, y];
 if (matrix[x, y]) g.FillRectangle(Brushes.Red, a * x + 1, a * y + 1, a - 1, a -
1);
```

```
 else g.FillRectangle(Brushes.White, a * x + 1, a * y + 1, a - 1, a - 1);
 }

 private void Form1_Shown(object sender, EventArgs e)
 {
 g.Clear(Color.White);
 for (int i = 1; i < 8; i++)
 {
 g.DrawLine(Pens.Black, 0, i * a, panel1.Width, i * a);
 g.DrawLine(Pens.Black, i * a, 0, i * a, panel1.Width);
 }
 }

 private void Form1_FormClosed(object sender, FormClosedEventArgs e)
 {
 port.Close();//закриваємо послідовний порт
 }

 private void button2_Click(object sender, EventArgs e)
 {
 for (int i = 0; i < 8; i++)
 {
 for (int j = 0; j < 8; j++)
 {
 matrix[i, j] = false;
 g.FillRectangle(Brushes.White, a * i + 1, a * j + 1, a - 1, a - 1);
 }
 }
 }
 }
}
```