Міністерство освіти і науки України
Національний університет водного господарства та природокористування
Навчально-науковий енергетики, автоматики та водного господарства
Кафедра автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій

04-03-390М

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт з навчальної дисципліни «**Моделювання кіберфізичних систем**» (Частина 1) для здобувачів вищої освіти другого (магістерського) рівня за освітньо-професійною програмою «Автоматизація, комп'ютерно-інтегровані технології та робототехніка» спеціальності 174 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка» денної та заочної форм навчання

Рекомендовано науково-методичною радою з якості ННІ ЕАВГ
Протокол № 11 від 02 липня 2024 р.

Рівне – 2024

Методичні вказівки до виконання лабораторних робіт з навчальної дисципліни «Моделювання кіберфізичних систем» для здобувачів вищої освіти другого (магістерського) рівня за освітньо-професійною програмою «Автоматизація, комп’ютерно-інтегровані технології та робототехніка» спеціальності 174 «Автоматизація, комп’ютерно-інтегровані технології та робототехніка» денної та заочної форм навчання. Частина 1. [Електронне видання] / Мащенко В.А. – Рівне : НУВГП, 2024. – 74 с.

Укладач: Мащенко В. А., кандидат фізико-математичних наук, доцент, доцент кафедри автоматизації, електротехнічних та комп’ютерно-інтегрованих технологій.

Відповідальний за випуск: Древецький В. В., доктор технічних наук, професор, завідувач кафедри автоматизації, електротехнічних та комп'ютерно-інтегрованих технологій.

Керівник групи забезпечення: Рудик А. В., доктор технічних наук, професор, професор кафедри автоматизації, електротехнічних та комп’ютерно-інтегрованих технологій.

Зміст

Вступ

Курс «Моделювання кіберфізичних систем» носить важливий характер при здобутті студентами знань та вмінь щодо проектування, розроблення та моделювання таких систем відповідно до досягнень та сучасних концепцій застосування комп'ютерних, інформаційних та телекомунікаційних технологій

Методичні вказівки містять порядок виконання лабораторних робіт першої частини курсу „Моделювання кіберфізичних систем" за основними темами, що вимагають застосування на практиці отриманих знань із використанням системи MATLAB та її окремих додатків, програм Multisim, Arduino IDE та Motbus Slave.

В методичних вказівках зібраний, систематизований та наочно викладений теоретичний матеріал, який охоплює практичні питання методології фізичних та обчислювальних компонентів, що інженерно взаємодіють у кіберфізичних системах. Структура кожної лабораторної роботи, їх зміст і порядок виконання є обґрунтованими, що роблять методичні вказівки зручними для опрацювання матеріалу студентами, як денної, так і заочної форм навчання.

Лабораторна робота № 1
Вивчення можливостей програмного засобу Stateflow додатку Simulink системи MATLAB

Мета роботи: ознайомитися із основними можливостями програмного засобу Stateflow.

Прилади та компоненти:

система MATLAB із програмним засобом Stateflow версії не нижче 8.6.

1.1. Теоретичні відомості

Програмний засіб Stateflow є розширенням системи Simulink, який представляє собою середовище для побудови і моделювання подіє-керованих систем. Для моделювання скінченних автоматів можна використовувати стандартні логічні блоки системи Simulink, однак програмний засіб Stateflow забезпечує можливість формування скінченних автоматів шляхом побудови ієрархічних моделей у вигляді підсистем.

1.1.1. Скінченні автомати

Моделі системи Simulink можуть включати підсистеми, способом представлення яких є скінченний автомат – система із дискретним часом і визначеною в ній множині станів (із врахування початкового стану), вхідним і вихідним алфавітом та правилами формування переходів і вихідних повідомлень. Стан скінченного автомата може бути представлений як вектор, який складається із скінченного числа елементів.

Прикладом такої підсистеми є комутуючій логічний пристрій, який керує станом шасі під час зльоту та посадки літака. Вхідними змінними логічного пристрою є сигнали із джойстика керування в кабіні пілотів а також різних датчиків і перемикачів, які передбачені для запобігання передчасного прибирання шасі і підтвердженню, що шасі випущенні, коли джойстик керування знаходиться в нижньому положенні.

Одним із способів опису логічних операцій є таблиці істинності. Модель вимикача світла із використанням таблиць істинності представлена в табл. 1.1.1.

Таблиця 1.1.1

Таблиця істинності для системи, яка складається із одного перемикача лампи освітлення

Стан перемикача	Стан лампи освітлення
Вверх	Увімкнена
Вниз	Вимкнена

В системі, що складається із двох перемикачів, яка керує однією лампою освітлення є два можливих стани: Вверх і Вниз, таким чином система описується одним із чотирьох можливих станів (табл. 1.1.2).

Таблиця 1.1.2

Таблиця істинності для системи, яка складається із двох перемикачів

Стан перемикача А	Стан перемикача В	Стан лампи освітлення
Вверх	Вверх	Вимкнена
Вниз	Вверх	Увімкнена
Вверх	Вниз	Вимкнена
Вниз	Вниз	Увімкнена

Таблиця істинності визначає можливі стани системи, але не відображає умови спрацювання переходів. Наприклад у системі, що розглядається, в даний момент часу тільки один перемикач може змінювати свій стан. Для представлення множини станів та доступних переходів при настанні заданих подій можна використовувати діаграми переходів станів.

1.1.2. Діаграми переходів станів

Діаграма переходів станів є структурою, яка включає зображення станів, що з’єднуються вітками, та показує події для

переходу у новий стан (допустимі переходи). На рис. 1.1.1 представлена діаграма переходів станів для системи, яка сказається із одного перемикача та має два стани: On та Off і вказує події для переходу в новий стан.

Діаграма переходів станів містить:

– зображення станів, які представляються у вигляді кіл;

– дуги, що з'єднують стани (допустимі переходи) та їх позначення, нанесенні поряд із дугою, яка описує подію настання якої викликає перехід (позначення переходу).

В наші системі маємо взаємно однозначну відповідність між переходами і подіями, але в загальному випадку таке не завжди має місце.

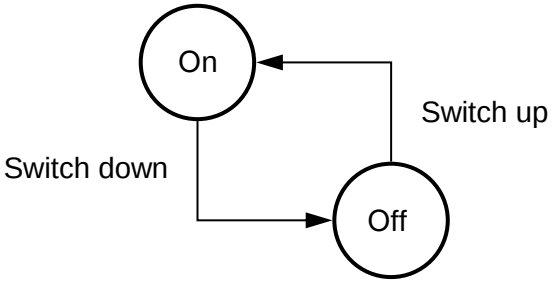


Рис. 1.1.1. Діаграма переходів станів для системи із одним перемикачем

На рис. 1.1.2. представлена діаграма переходів станів для системи, яка складається із двох перемикачів. Допустимі переходи визначається зміною станів одного перемикача, що трактується як умова спрацювання переходів. На діаграмі переходів станів представлені чотири можливих стани, а також допустимі переходи в новий стан і події, які викликають дані переходи.

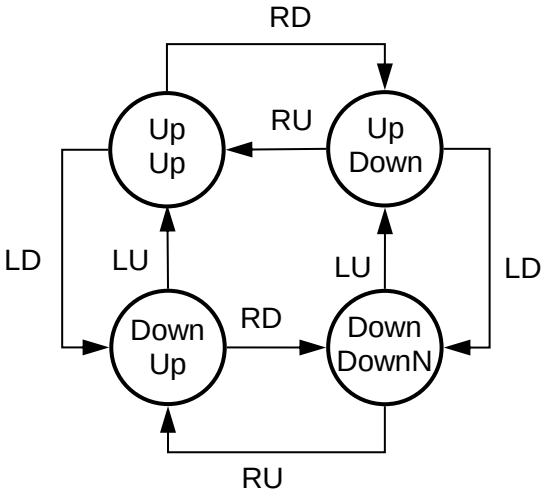


Рис. 1.1.2. Діаграма переходів станів для системи із двома перемикачами:

RU – правий перемикач вверх;

RD – правий перемикач вниз;

LU – лівий перемикач вверх;

LD – лівий перемикач вниз.

1.2. Діаграма Stateflow

Діаграма Stateflow є вдосконаленою діаграмою переходів станів. Для зручності опису вводиться ієрархія діаграм і паралельне виконання декількох станів. Програмний засіб Stateflow забезпечує графічне зображення станів (кожний із яких може бути також представлений у вигляді діаграми Stateflow) з врахуванням початкового значення і подій, які описуються предикатами на множині дискретних і (або) неперервних змінних.

Якщо предикат приймає істинне значення, то умова зміни стану виконується. Діаграми Stateflow також забезпечують

можливість розгалуження з використанням переходів, що складаються із декількох станів, які можуть включати послідовні події і точки прийняття рішень.

Позначення переходів на діаграмі Stateflow має чотири частини, кожна з яких не є обов'язковою:

подія [умова] {дія умови} / дія переходу

Подія має таке ж значення як і на діаграмі переходів станів, умова – це предикат – функція, змінні, які приймають значення із деякої множини, а сама функція приймає два значення: «істина» і «хиба».

Дія умови – будь яка дія, яка повинна відбутися, якщо предикат приймає значення «істина», незалежно від того, пройшов перехід чи ні.

Дія переходу – будь яка дія, яка провинна відбутися як результат переходу. Якщо подія не представлена в позначеннях переходу, а умова присутня, то перехід проходить, коли умова приймає значення «істина».

Програмний засіб Statefow також забезпечує можливість реалізації переходів, які складаються із декількох станів. У випадку такого переходу може відбутися дія умови, але не дія переходу.

Для простого скінченного автомата діаграма Stateflow зображається подібно діаграмі переходів станів. Наприклад, на рис. 1.2.1 представлена діаграма Stateflow для системи, що складається із одного перемикача. В цьому випадку кола заміняються прямокутниками із заокругленими кутами, а позначення переходу складаються із двох частин – подія та дія:

подія / дія події

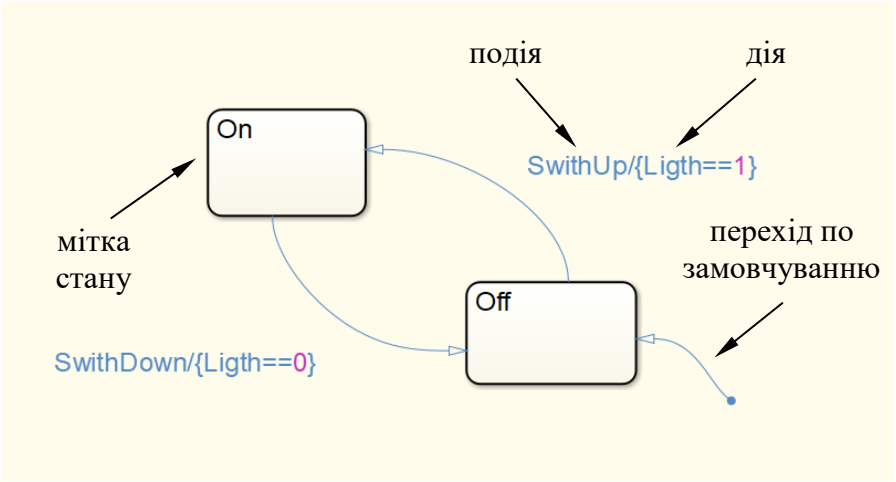


Рис. 1.2.1. Діаграма Statefow системи, що складається із одного перемикача

В цьому випадку, при настанні заданої події перемикач переходить в новий стан, який визначає для ламп освітлення стан «Увімкнена» або «Вимкнена». На діаграмі також зображаться додатковий перехід, який починається з точки, поміченою маркером типу круг синього кольору, який називається перехід по замовчуванню і слугує для визначення початкового стану системи.

На рис. 1.2.2 представлена діаграма Stateflow для системи, що складається із двох перемикачів, яка для визначення початкового стану системи також включає перехід по замовчуванню.

В даному прикладі позначення переходів вказують умови, які викликають перехід. Перехід проходить, коли вказану умова стає істиною. Дії (Ligtht = 0 або 1) пов'язані із чотирма станами, а не з переходами. Тому кожний підпис на зображенні стану включає позначення стану (наприклад UpUp) та позначення дії, пов'язаної із станом. Так як дії пов'язані із станами, а не з переходами, переходи позначаються умовами.

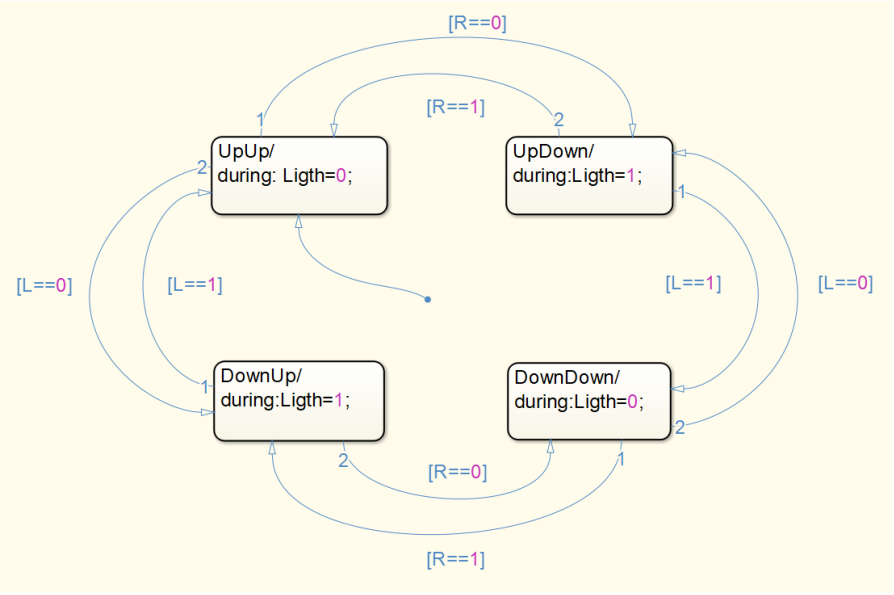


Рис. 1.2.2. Діаграма Stateflow системи, яка складається із двох перемикачів

Завдання

1. Побудувати діаграму Stateflow для системи, що складається із одного перемикача.
2. Побудувати модель та діаграму Stateflow для системи, що складається із двох перемикачів (рис. 1.2.3).

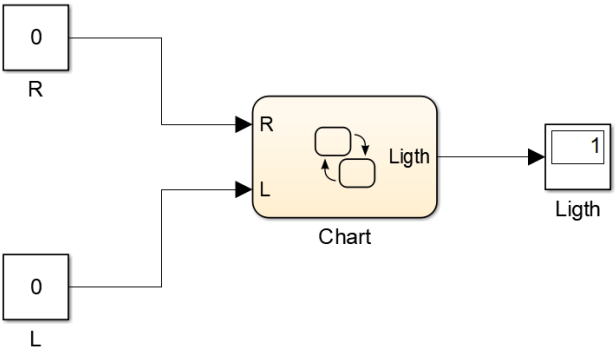


Рис. 1.2.3. Модель системи, що складається із двох перемикачів

Лабораторна робота № 2
Побудова та моделювання ієрархічних моделей подіє-керованих систем

Мета роботи: побудувати моделі подіє-керованих систем у програмному засобі Stateflow.

Прилади та компоненти:

система MATLAB із програмним засобом Stateflow версії не нижче 8.6.

2.1. Теоретичні відомості

Програмний засіб Stateflow є розширенням системи Simulink, який представляє собою середовище для побудови і моделювання подіє-керованих систем.

2.1.1. Ієрархія діаграми Stateflow

Механізм ієрархії дозволяє використовувати рекурсивний спосіб опису діаграми Stateflow і вважати, що кожний стан може бути представлений у вигляді відповідної діаграми.

В якості прикладу розглянемо функціонування моделі системи склоочищувачів автомобіля. Система має два стани, які можуть бути призначені як суперстанами: On (Включено) и Off (Вимкнуто).

Суперстан On має два підстани: Slow (Повільно) і Fast (Швидко). Діаграма Stateflow системи склоочищувачів автомобіля представлена на рис. 2.1.1. Слід відмітити, перехід по замовчуванню для системи – перехід в стан Off. Як тільки система переходить у суперстан On, даний стан переходить в підстан Slow.

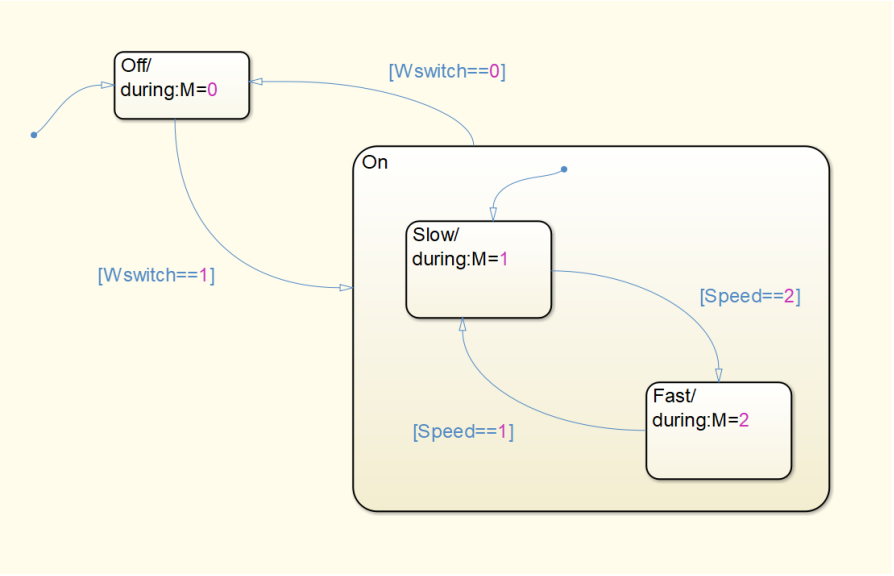


Рис. 2.1.1. Модель системи склоочищувачів автомобіля

Другий механізм дозволяє одночасно знаходитися у декількох станах і називається паралельним функціонуванням декількох станів.

Розглянемо модель електричної системи автомобіля (рис. 2.1.2), яка включає підсистему склоочищувачів і підсистему фар. Штрихові рамки, якими виділені дані підсистеми, вказують на паралельне функціонування декількох станів у незалежних підсистемах.

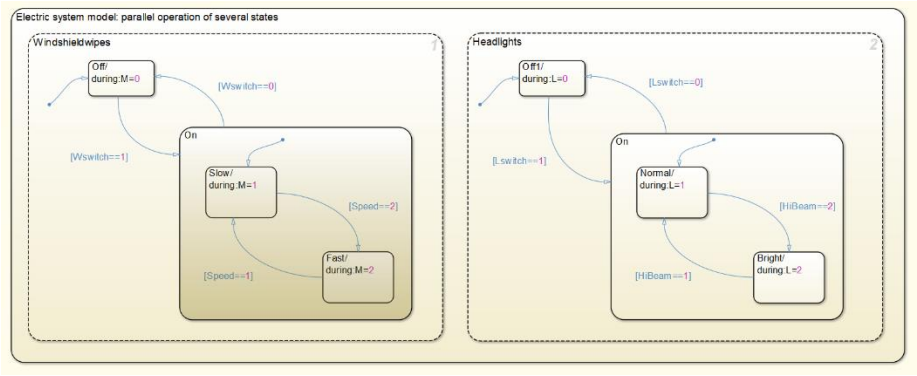


Рис. 2.1.2. Модель електричної системи паралельне функціювання декількох станів

2.1.2. Перехід в останній активний стан

Перехід в останній активний стан також є корисним елементом діаграми Stateflow. Даний перехід зображується у вигляді кола, яке містить символ H. якщо суперстан містить перехід у останній активний стан, то стан, який був активним, коли система знаходилася у даному суперстані, буде стан по замовчуванню, коли система знову виявиться у даному суперстані.

У перши раз, коли система виявиться в даному суперстані, буде виконаний перехід по замовчуванню, але після цього перехід по замовчуванню буде анульований переходом у останній активний стан. Скорегована діаграма Stateflow для моделі підсистеми фар представлена на рис. 2.2.1.

Суперстан On містить перехід у останній активний стан. Перший раз, коли система буде знаходитися у суперстані On, відбудеться перехід по замовчуванню в стан Normal. Якщо фари перемикаються в стан Bright а потім вимкнуться, то при включенні підсистема фар буде знаходитись в стані Bright.

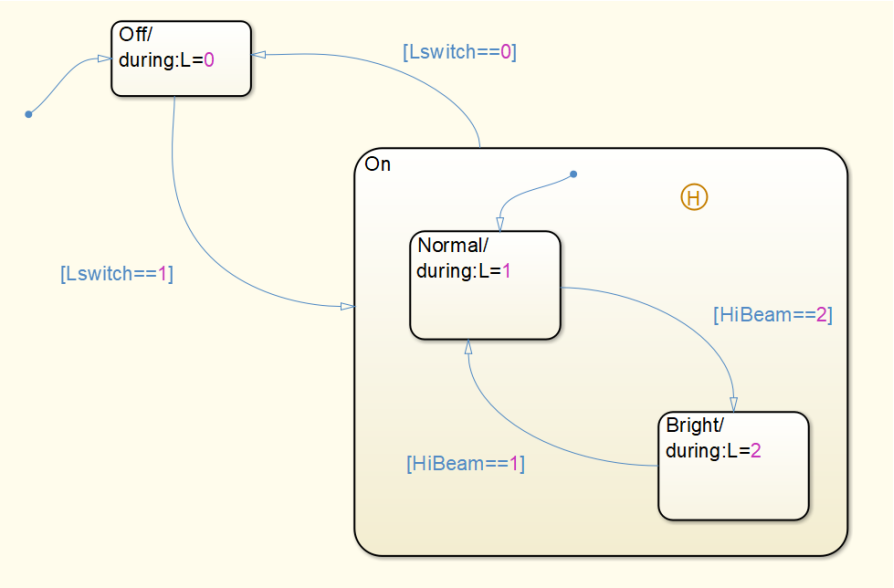


Рис. 2.2.1. Модель підсистеми фар із переходом у останній активний стан

Завдання

1. Побудувати діаграму Stateflow для системи склоочищувачів автомобіля.
2. Побудувати діаграму Stateflow для моделі електричної системи, де реалізується паралельне функціювання декількох станів.
3. Побудувати діаграму Stateflow для моделі підсистеми фар із переходом у останній активний стан.

Лабораторна робота № 3
Дослідження програмованих логічних інтегральних схем

Мета роботи: ознайомитися із принципами побудови програмованих логічних інтегральних схем та їх віртуальною роботою.

Прилади та компоненти:

програма Multisim Power Pro Edition 12.

3.1. Теоретичні відомості

У якості найпростіших програмованих логічних інтегральних схем (ПЛІС) можуть використовуватися інтегральні схеми (ІС) мультиплексорів.

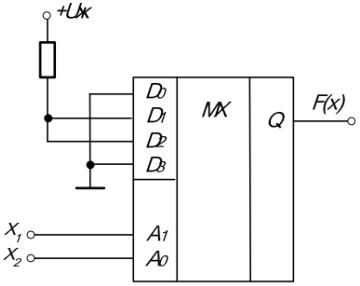
Виникає питання чи можна розробити такий універсальний багатофункціональний пристрій, на якому реалізуються функції булевої алгебри (ФБА) довільної складності.

Таке завдання, можна вирішити ввівши у схему пристрій із деякою функціональною надлишковістю. Подаючи сигнали на додаткові входи можна вносити необхідні зміни у його характеристики, тобто пристрій повинен містити входи налаштування. Найбільш проста реалізація даного завдання полягає у використанні ІС мультиплексора, який може використовуватись не лише як пристрій комутації, але й як універсальний логічний елемент (ЛЕ).

Ідея використання мультиплексора в якості універсального ЛЕ полягає в тому, що його адресні входи можна використовувати в якості інформаційних, а інформаційні входи використати як входи налаштування. При цьому на входах налаштування можуть бути сформовані сигнали, логічні константи або деякі допоміжні функції.

x_1	x_2	F(x)
0	0	0
0	1	1
1	0	1
1	1	0

а)



б)

Рис. 3.1.1. Приклад реалізації ФБА двох вхідних змінних (а) на мультиплексорі (б)

Із прикладу на рис. 3.1.1 видно, що запропонований метод обмежується реалізацією функції чотирьох змінних, тому що мультиплексори, які випускаються у виді ІС, мають не більше 16 інформаційних входів.

При необхідності реалізації ФБА більшого числа вхідних змінних можна скористатися структурою мультиплексорного дерева. Однак при невеликому числі аргументів це завдання можна вирішити й іншим методом, а саме вибором сигналів налаштування не із множини $\{0, 1\}$ як це було реалізовано у прикладі на рис. 3.1.1, а із множини $\{0, 1, x_1, \ldots, x_n\}$, де x_n – один з аргументів функції,.

У такому випадку іноді вдається, без додаткових апаратних затрат, реалізувати ФБА на мультиплексорі, число аргументів якої на одиницю більше числа його адресних входів.

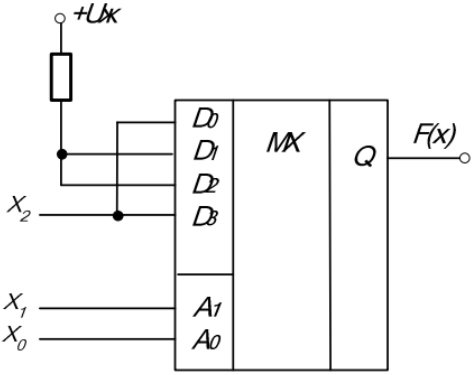


Рис. 3.1.2. Приклад реалізації ФБА трьох вхідних змінних $F(x) = x_1\bar{x}_0 + x_2 + \bar{x}_1 x_0$ на мультиплексорі

Потрібно відзначити, що таке технічне рішення не єдине. На входи налаштування можна подавати довільні аргументи та функції. Значення аргументів функції налаштування, які подаються на інформаційні входи, визначається залишковою функцією, тобто значеннями заданої ФБА на фіксованих значеннях аргументів.

Очевидно, що у такому випадку практична реалізація заданої ФБА вимагає ввкдення допоміжного ЛЕ. Отже, коли число адресних входів мультиплексора на один менше числа незалежних змінних заданої ФБА, елемент не універсальний, а багатофункціональний.

Реалізуючи описаний метод, можна вибирати сигнал налаштування із більш широкої множини, що включає в себе декілька аргументів. При цьому на мультиплексорі із двома адресними входами можна реалізувати ФБА чотирьох і більше змінних. Ефективність такого методу падає пропорційно збільшенню кількості змінних.

Розглянемо у загальному вигляді питання технічної реалізації системи ФБА, задану диз'юнктивною нормальною формою. Для цього розглянемо систему ФБА виду:

$$
\begin{aligned}
F_0(x) &= x_n x_{n-1} \dots x_2 x_1 x_0 + \bar{x}_n x_{n-1} \dots x_2 x_1 x_0 + \\
&\quad \dots \bar{x}_n \bar{x}_{n-1} \dots \bar{x}_2 \bar{x}_1 x_0, \\
F_1(x) &= x_n x_{n-1} \dots \bar{x}_2 x_1 x_0 + x_n x_{n-1} \dots \bar{x}_2 x_1 \bar{x}_0 + \\
&\quad \dots \bar{x}_n \bar{x}_{n-1} \dots \bar{x}_2 x_1 \bar{x}_0, \\
&\quad \dots \\
F_m(x) &= x_n \bar{x}_{n-1} \dots x_2 x_1 x_0 + \bar{x}_n x_{n-1} \dots \bar{x}_2 x_1 x_0 + \\
&\quad \dots \bar{x}_n \bar{x}_{n-1} \dots x_2 \bar{x}_1 \bar{x}_0.
\end{aligned}
\qquad (3.1.1)
$$

Кількість добутків у кожній функції обмежено величиною 2^n, причому у граничному випадку кожний добуток (терм) є відповідним конституентом одиниці.

Для одержання значення функції над всіма термами, що входять у вираз (3.1.1), необхідно виконати операцію диз'юнкції. Відповідно до цього схема апаратної реалізації виразу (3.1.1) повинна містити послідовно включені вхідний буфер, блок формування термів, блок диз'юнкції та вихідний буфер (рис. 3.1.3).

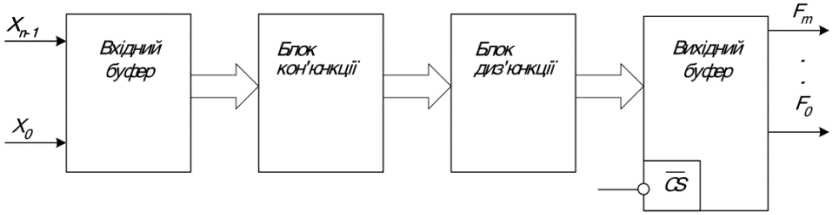


Рис. 3.1.3. Узагальнена структурна схема ПЛІС

У загальному випадку блок «термів» (кон'юнкцій) представляє собою матрицю логічних елементів «І», а блок диз'юнкцій – матрицю логічних елементів «АБО». Тому послідовне з'єднання таких матриць, у загальному випадку, дозволяє реалізувати ФБА довільного виду. Реалізація конкретних ФБА допускає виконання конкретних з'єднань у матрицях елементів «І» та «АБО».

Таким чином, змінюючи з'єднання елементів у матрицях «І» та «АБО», можна створювати конкретні властивості пристрою, що відповідає схемі, представленої на рис. 3.1.3.

На практиці можлива реалізація трьох варіантів налаштувань:

– незмінна структура матриці «І» та структура матриці «АБО», яка може змінюватися програмуванням;
– структура матриці «І», яка може змінюватися програмуванням та незмінна структура матриці «АБО»;
– структури матриць «І» та «АБО», які можуть змінюватися програмуванням.

Кожному із цих варіантів відповідає свій тип ПЛІС.

Технічною реалізацією першого типу налаштування ПЛІС є пристрої постійної пам'яті (ППП). Другий варіант налаштування ПЛІС реалізовано в ІС програмованої матричної логіки (ПМЛ), а третій – у програмованих логічних матрицях (ПЛМ).

Пристрої постійної пам'яті, завдяки своїй простоті, а головне – регулярності структури, забезпечують високу технологічність при виготовленні і максимально доступну, на сьогоднішній день, ступінь інтеграції.

У структурі ППП можна виділити блоки, що відповідають узагальненій структурній схемі ПЛІС (рис. 3.1.3).

Роль матриці «І» виконує дешифратор, що перетворює n вхідних сигналів x_i у n^2 вихідних сигналів N.

Така побудова матриці «І» допускає цілком визначену структуру матриці «АБО», функції якої, по суті, вироджуються до рівня підключення до виходу сигналів логічного «0», або логічної «1». Така організація забезпечує ППП високу технологічність.

Відзначимо наступні особливості використання ППП в якості ПЛІС:

– ППП реалізує ФБА, представлену у вигляді досконалої диз'юнктивної нормальної форми, тобто ФБА повинна бути представлена сумою конституент

одиниці, тому для технічної реалізації її мінімізація не потрібна;

– ППП дозволяє реалізувати тільки повністю визначені ФБА, тому необхідна однозначність її значень для всіх можливих комбінацій вхідних змінних.

При реалізації будь-якої ФБА n змінних у ППП передбачаються однакові апаратні затрати, що відповідають максимально можливому числу змінних, тобто завжди існує можливість одержання всіх N конституент. Тому застосування ППП технічно та економічно виправдане тільки для реалізації складних ФБА, що не піддаються мінімізації.

Слід зазначити, що швидкодія пристроїв, що використовують для реалізації заданих систем ФБА ППП, як правило, вища, ніж при реалізації ФБА на основі стандартних ЛЕ та дорівнює часу звертання вибраного типу ІС. Практика показує, що, як правило, ФБА, яка містить велике число змінних, задана істотно меншим, чим n^2, числом конституент одиниці.

У цьому випадку застосування ППП як ПЛІС стає неефективним, що пояснюється їх великою апаратною надлишковістю. Дійсно, збільшення числа вхідних змінних на одиницю вимагає подвійного збільшення числа виходів дешифратора ППП, тобто істотно ускладнює ІС та підвищує її вартість. Цей недолік ППП призвів до створення принципово нового класу приладів – програмованої матричної логіки (ПМЛ), у якій збільшення числа входів матриці «І» не приводить до збільшення числа її виходів. Це досягається зменшенням максимального числа конституент, що задають вихідну ФБА. У практичний реалізації це у переважній більшості випадків цілком виправдано. Таким чином, ІС ПМЛ при однаковій з ППП площі кристала дозволяє реалізувати ФБА із значно більшим числом вхідних змінних.

Оскільки в ПМЛ з’являється обмеження на максимальне число операцій кон’юнкції, то такі ІС, крім розрядності вхідного слова n і числа вихідних функцій p, характеризуються ще одним параметром – максимальним числом операцій кон’юнкції (термів) у ФБА.

Одна ІС ПМЛ дозволяє із n вхідних змінних синтезувати не більш ніж l термів, до яких можна застосувати операцію диз'юнкція, для реалізації не більш ніж p вихідних функцій. Саме тому для реалізації ФБА, з використанням ІС ПМЛ, необхідна її мінімізація.

ПМЛ реалізує другий із розглянутих варіантів програмування ПЛІС. У цьому випадку налаштовується матриця «І» при незмінних зв'язках матриці «АБО».

Один із варіантів структурної схеми, що реалізує запропоноване технічне рішення, представлений на рис. 3.1.4, де на перетині шин вхідних змінних x_i та шин вхідних виводів елементів «І» умовно показана наявність всіх перемичок (l). Програмування ІС виконується для усунення зайвих, з погляду реалізованого алгоритму, зв'язків між зазначеними шинами.

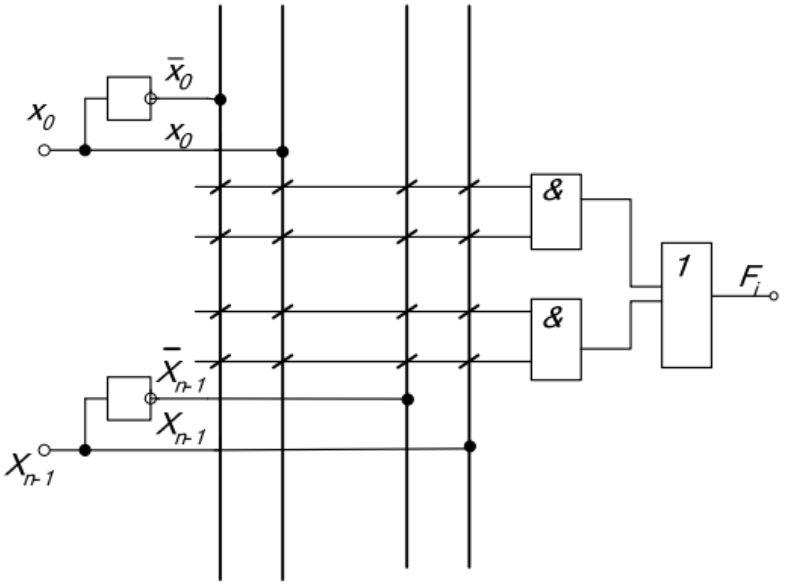


Рис. 3.1.4. Структурна схема ПМЛ

Програмовані логічні матриці реалізують третій тип програмування ПЛІС та забезпечують можливість зміни зв'язків як у матриці «І», так і в матриці «АБО». При всій гнучкості такого рішення фахівці вважають, що даний тип ПЛІС досить складний для більшості користувачів із принципу їхнього програмування.

Крім цього, наявність програмованого з'єднання, наприклад плавкої перемички, в обох матрицях призводить до збільшення розмірів, зменшення надійності та швидкодії в порівнянні з ПМЛ. Один із варіантів структурної схеми, що реалізує розглянутий принцип побудови ПЛІС, представлений на рис. 3.1.5. На рис. 3.1.5 також умовно показана наявність всіх перемичок у матрицях «І» та «АБО».

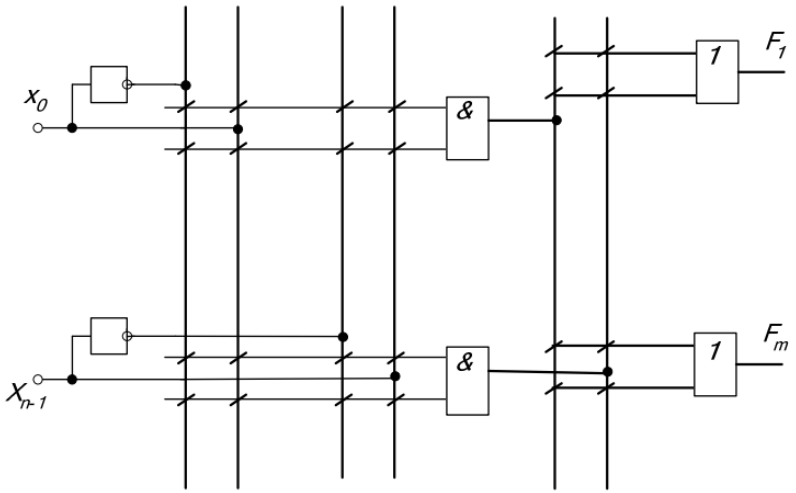


Рис. 3.1.5. Структурна схема ПЛМ

3.2. Опис контрольно-вимірювальних приладів та обладнання

3.2.1. Осциллограф

Осцилограф призначений для спостереження, вимірювання та запису електричних сигналів. Графічне позначення та лицева панель осцилографа зображені на рис. 3.2.1.

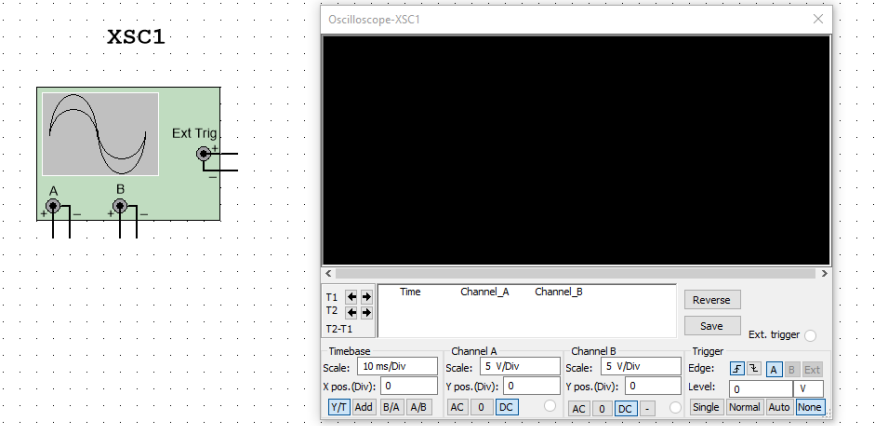


Рис. 3.2.1. Графічне позначення та лицева панель осцилографа

3.2.1. Генератор цифрових сигналів

Генератор цифрових сигналів призначений для формування комбінацій цифрових сигналів (логічних «1» та «0») з різною частотою. Генерація може бути організована циклічно, покроково або із переривранням. Реалізована функція сигналу «готовність даних». Графічне позначення та лицева панель генератора цифрових сигналів зображені на рис. 3.2.2.

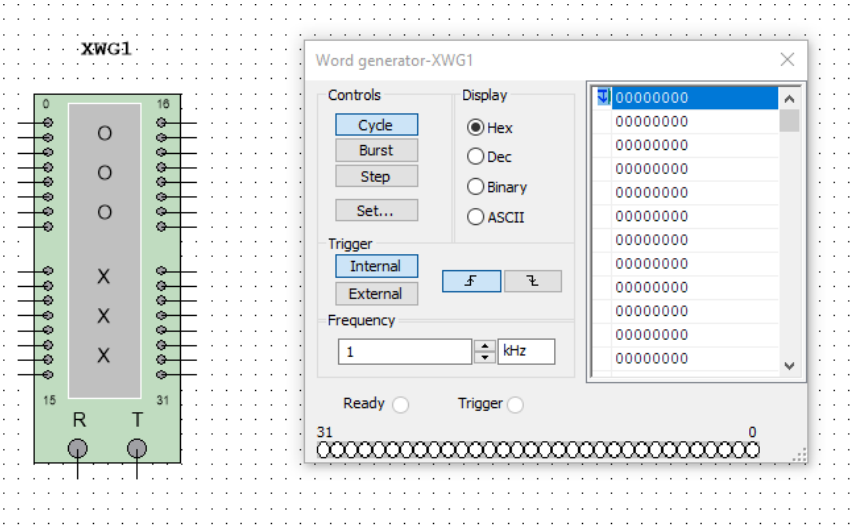


Рис. 3.2.2. Графічне позначення та лицева панель генератора цифрових сигналів

3.2.3. Цифровий логічний аналізатор

Цифровий логічний аналізатор призначений для перегляду осцилограм цифрових сигналів. В цифровому логічному аналізаторі реалізовані такі функції, як: можливість використання внутрішнього та зовнішнього сигналів синхронізації, регулювання частоти розгортки. Графічне позначення та лицева панель цифрового логічного аналізатора зображені на рис. 3.2.3.

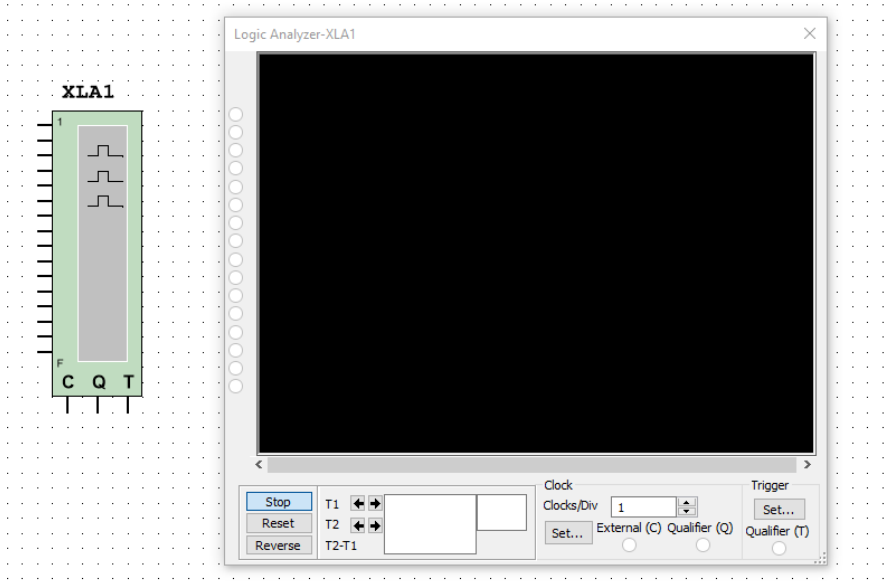


Рис. 3.2.3. Графічне позначення та лицева панель цифрового графічного аналізатора

3.2.4. Мультиплексор

Графічне позначення мультиплексора зображено на рис. 3.2.4.

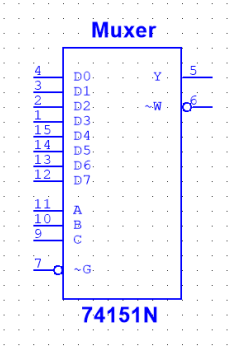


Рис. 3.2.4. Графічне позначення мультиплексора

Завдання

1. Дослідження програмованої логічної інтегральної схеми на базі мультиплексора.

1.1. Запустіть програму Multisim 12.

1.2. Віртуально реалізуйте схему для дослідження (рис. 3.3.1).

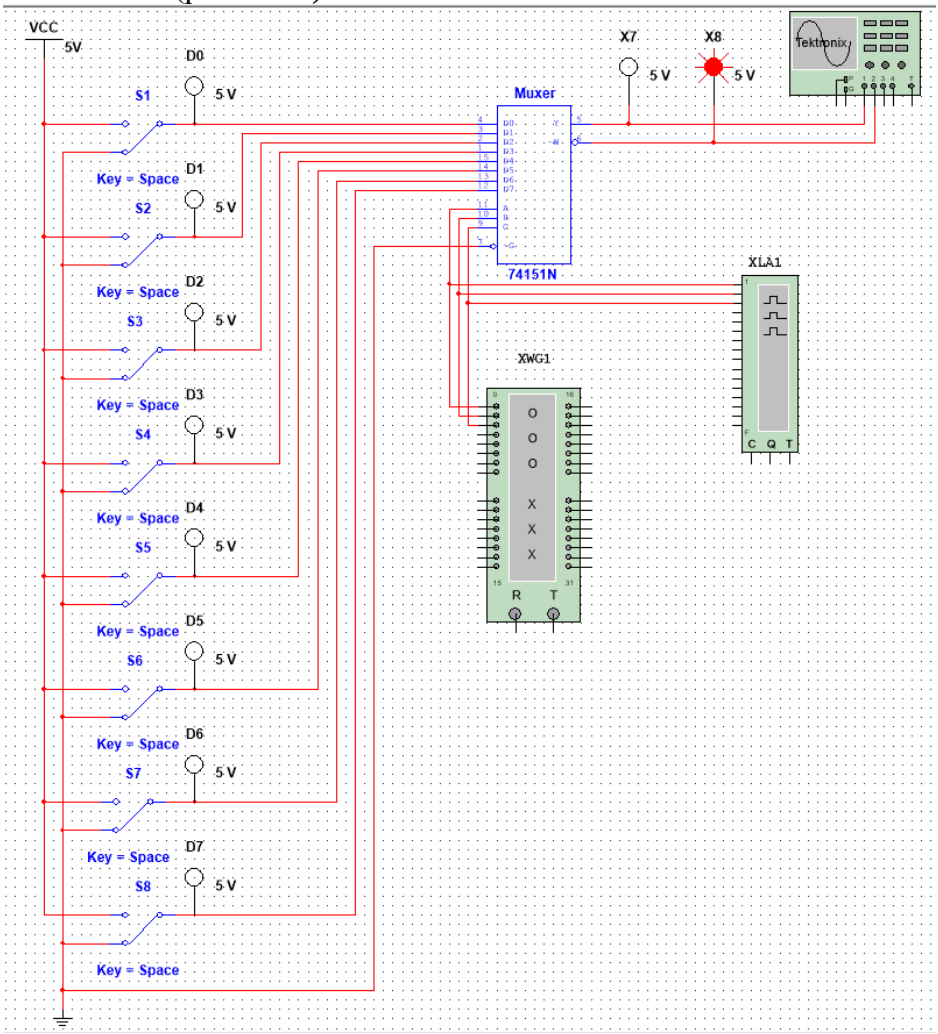


Рис. 3.3.1. Схема для дослідження

1.3. До входів мультиплексора під’єднайте ключі, якими буде реалізоване керування його логічними станами. Першу позицію всіх ключів під’єднайте до «землі», другу – до напруги живлення (+ 5 В).

1.4. До адресних входів мультиплексора під’єднайте генератор цифрових сигналів та згенеруйте послідовність, яка буде адресом послідовного опитування станів входів мультиплексора.

1.5. До виходу схеми під’єднайте осцилограф.

1.6. Коли схема зібрана і готова до запуску, натисніть кнопку включеня живлення на панелі інструментів.

1.7. Змінюйте логічний стан входів мультиплексора за допомогою ключів та задайте різні ФБА.

1.8. Після реалізації ФБА спостерігаєте за зміною стану виходу схеми відповідно до комбінації вхідних сигналів (тобто за результатом реалізації заданої ФБА).

1.9. Проаналізуйте отримані цифрові осцилограми відповідно до заданих ФБА.

1.10. Визначити яку ФБА реалізує мутиплексор?

1.11. Зробити висновки.

2. По закінченню виконання дослідів оформіть звіт по лабораторній роботі на підставі отриманих експериментальним даних. Звіт повинен складатися з схем проведених дослідів та отриманих результатів відповідно, а також висновків зроблених по закінченню кожного досліду.

Лабораторна робота № 4
Реалізація послідовного зв'язку між системою MATLAB та зовнішнім пристроєм на базі контролеру Atmel

Мета роботи: апаратно і програмно реалізувати послідовний від зв'язку між системою MATLAB та зовнішнім пристроєм на базі контролеру Atmel.

Прилади та компоненти:

апаратне забезпечення

плата Arduino Uno;
світлодіод (будь-якого кольору);
резистор 330 Ом;

програмне забезпечення:

система MATLAB версії R2016a або вище.

4.1. Короткі відомості

MATLAB представляє собою універсальний програмний продукт, який можна використовувати для широкого кола задач при реалізації різноманітних додатків. У лабораторній роботі ми розглянемо як використовувати послідовний зв'язок у системі MATLAB із пристроєм на базі контролерів Atmel. Як приймач для послідовного зв'язку будемо використовувати плату Arduino UNO (рис. 4.1.1).

Існує два основних способи використання послідовного зв'язку в MATLAB – за допомогою вікна команд та за допомогою MATLAB GUI (Graphical User Interface – графічний інтерфейс користувача). Код програми для плати Arduino в обох випадках буде однаковим.

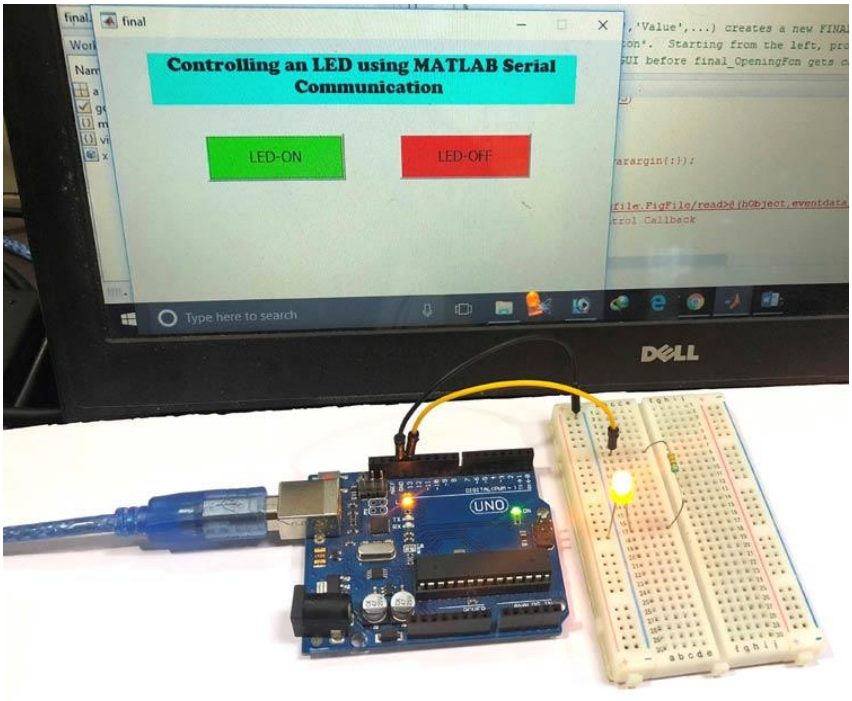


Рис. 4.1.1. Експериментальний стенд

4.2. Порядок виконання роботи

Схема проекту для послідовного зв’язку між MATLAB та Arduino представлена на рис. 4.2.1.

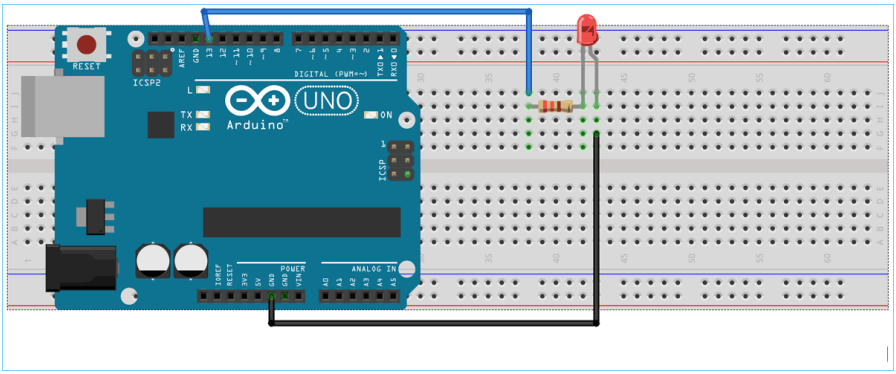


Рис. 4.2.1. Схема проекту

Зовнішній вигляд зібраної на макетній платі конструкції проекту показано рис. 4.2.2.

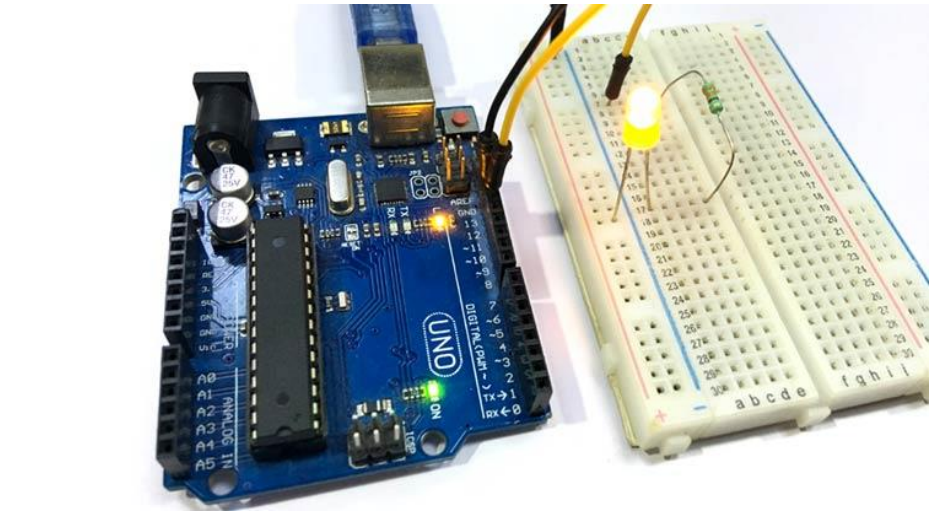


Рис. 4.2.2. Зовнішній вигляд зібраного проєкту на макетній платі

4.2.1. Послідовний зв’язок у MATLAB за допомогою вікна команд

Це найпростіший спосіб встановлення послідовного зв’язку між MATLAB та Arduino. Ми просто передаємо дані із MATLAB до Arduino послідовно використовуючи вікно команд (Command Window), а потім плата Arduino зчитує отримані дані. Передані дані можна використовувати для управління будь-яким пристроєм, підключеним до Arduino – у нашому проєкті ми використовуємо для цієї мети світлодіод, який ми вмикаємо і вимикатимемо в залежності від даних, що надходять.

Завантажте у плату Arduino такий скеч:

```
int value;

void setup()
{
Serial.begin(9600);
pinMode(13,OUTPUT);
}

void loop()
{
  if(Serial.available()>0)
  {
    value=Serial.read();

    if  (value==1)
    {
    digitalWrite(13,HIGH);
    }
    if(value==0)
    {
    digitalWrite(13,LOW);
    }
```

На наступному етапі можна розпочинати програмування у вікні редагування (Editor Window) системи MATLAB. Щоб відкрити вікно для нового скрипту, натисніть New Script як показано на рис. 4.2.3.

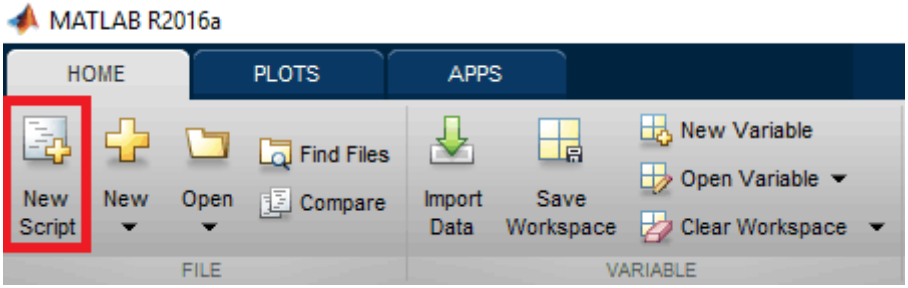


Рис. 4.2.3. Вікно системи MATLAB

Вставте у вікно редагування код програми, який слугує для реалізації послідовного зв'язку між MATLAB та Arduino:

```
x=serial('COM18','BAUD',9600);
fopen(x);
go=true;
while go
   a=input('Press 1 to turn ON LED & 0 to turn
OFF:');
fprintf(x,a);
if (a==2)
   go=false;
end
end
```

У поданому коді програми команда

```
x=serial('COM18','BAUD',9600);
```

використовується для встановлення послідовного зв'язку у MATLAB.

Переконайтеся, що в ній вказано саме той номер COM-порту комп'ютера, до якого підключена плата Arduino. Швидкість передачі даних необхідно встановити таку ж, як і у платі Arduino.

Для відкриття послідовного порту, використовується команда

```
fopen(x);
```

Наступна команда використовується для передачі даних від MATLAB до плати Arduino послідовно, де `x` – це виклик послідовного порту, а `a` – це значення, яке вводить користувач.

```
fprintf(x,a);
```

Ми будемо використовувати конструкцію `while` для створення нескінченного циклу, вихід з якого буде здійснюватися, якщо користувач введе цифру `'2'`.

Після того як відредагуєте весь текст скрипту натискайте на кнопку 'RUN' (рис. 4.2.4) щоб запустити його на виконання.

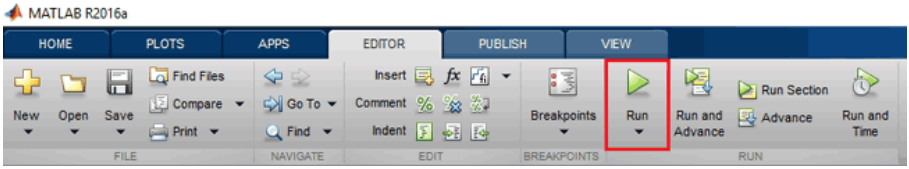


Рис. 4.2.4. Вікно системи MATLAB

Системі MATLAB може знадобитися кілька секунд на запуск цього скрипту, не робіть у цей час жодних дій поки горить індикація `BUSY` (система зайнята) у лівому нижньому кутку екрана.

Після запуску скрипта ви побачите у вікні Command Window повідомлення, запрограмоване у скрипті, де користувач може здійснювати введення даних (рис. 4.2.5).

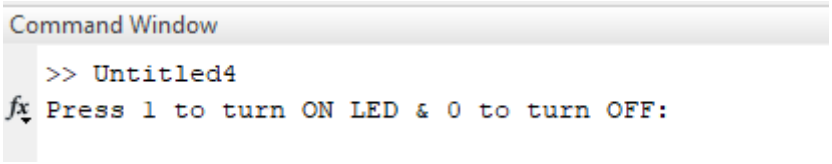


Рис. 4.2.5. Вікно Command Window системи MATLAB

Введіть `'1'` щоб увімкнути світлодіод, `'0'` – щоб вимкнути світлодіод та `'2'` щоб завершити роботу програми (скрипта). Можете використовувати будь-які інші символи для цього – але в цьому випадку необхідно буде змінити код програми для плати Arduino (рис. 4.2.6).

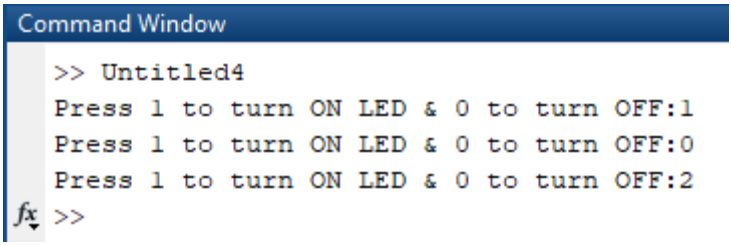


Рис. 4.2.6. Вікно Command Window системи MATLAB

4.2.2. Послідовний зв'язок у MATLAB з використанням графічного інтерфейсу

Для демонстрації можливостей послідовного зв'язку в MATLAB з використанням графічного інтерфейсу користувача (MATLAB GUI) створимо в системі MATLAB дві графічні кнопки для увімкнення та вимкнення світлодіода, підключеного до плати Arduino.

При натисканні цих кнопок дані будуть послідовно передаватися від MATLAB до Arduino. Плата Arduino, в залежності від прийнятих даних, вмикатиме або вимикатиме світлодіод. Код програми для плати Arduino буде таким самим, як і у попередньому випадку. Відмінність полягатиме лише в тому,

що в попередньому випадку ми вводили символи `'1'` та `'0'` у вікні команд, а в цьому випадку вони будуть передаватися послідовним каналом зв'язку при натисканні графічних кнопок.

Щоб запустити графічний інтерфейс користувача (`GUI` – Graphical User Interface) у MATLAB введіть команду

```
guide
```

у вікні Command Window.

Відкриється нове вікно, у якому виберіть новий `blank GUI` як показано на рис. 4.2.7.

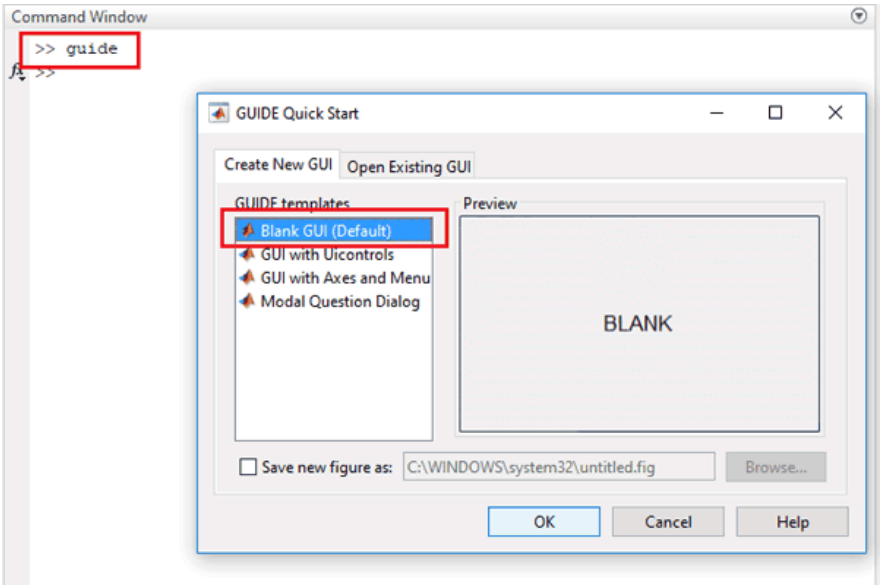


Рис. 4.2.7. Вікно Command Window та вікно GUIDE Quick Start

Після цього помістіть на робоче поле редактора дві кнопки для увімкнення та вимкнення світлодіода як показано на рис. 4.2.8.

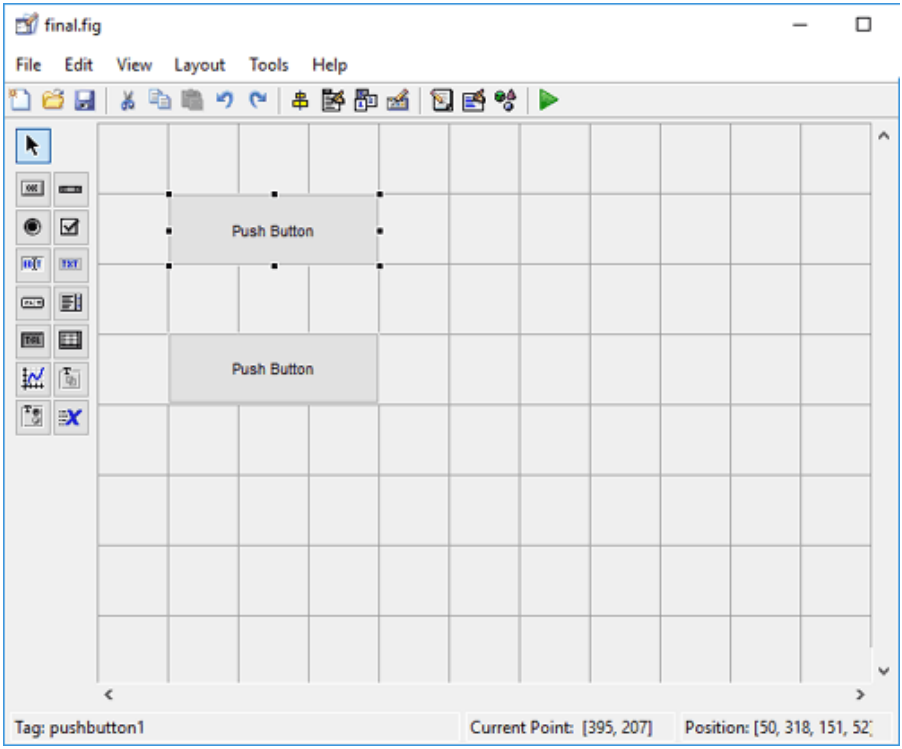


Рис. 4.2.8. Вікно редактора Graphical User Interface

Щоб змінити розмір або форму кнопки, натисніть на неї один раз. Після подвійного натискання на кнопку ви зможете змінити її колір, підпис та мітку (рис. 4.2.9).

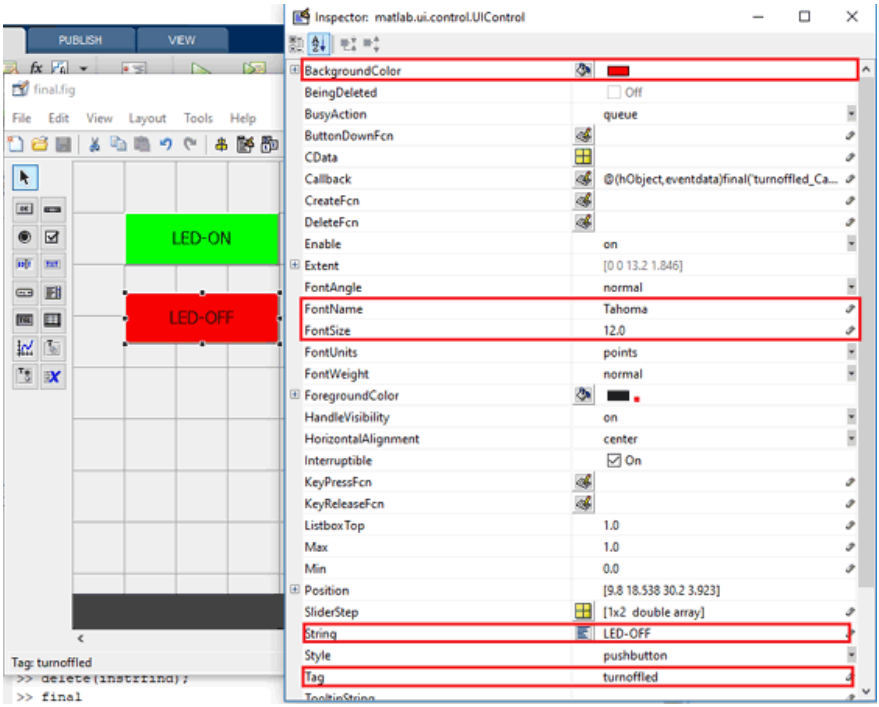


Рис. 4.2.9. Вікно параметрів об’єкта у GUI

Кастомізуємо (надаємо їм вигляд) ці дві кнопки таким чином (рис. 4.2.10).

Рис. 4.2.10. Вікно додатку для вмикання та вимикання світлодіода

Ви можете оформити (кастомізувати) їх на свій вибір. Коли ви збережете створений у графічному інтерфейсі користувача проект для нього, у вікні редагування MATLAB автоматично згенерується відповідний код. Надалі будемо вносити зміни до цього коду, щоб керувати включенням / виключення світлодіода із системи MATLAB.

Код програми в MATLAB і його основні фрагменти.

```
function varargout = final(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name', mfilename, ...
                   'gui_Singleton',  gui_Singl
eton, ...
                   'gui_OpeningFcn',
@final_OpeningFcn, ...
```

```
                   'gui_OutputFcn',  @final_Ou
tputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback =
str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] =
gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

handles.output = hObject;

guidata(hObject, handles);

function varargout = final_OutputFcn(hObject,
eventdata, handles)

varargout{1} = handles.output;
clear all;
global x;
x=serial('COM18','BAUD', 9600);
% Make sure the baud rate and COM port is
% same as in Arduino IDE
fopen(x);

function turnonled_Callback(hObject,
eventdata, handles)

global x;
fprintf(x,1);
```

```
function turnoffled_Callback(hObject,
eventdata, handles)

global x;
fprintf(x,0);
```

Встановлює послідовний зв'язок із заданою швидкістю (рис. 4.2.11):

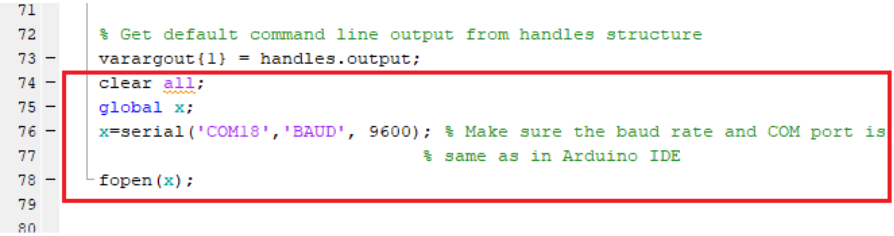


Рис. 4.2.11. Вікно редактора системи MATLAB

Функція `fopen(x)` у фрагменті цього коду використовується для відкриття послідовного порту.

Далі в коді програми, створеному за допомогою графічного інтерфейсу, ви побачите дві функції, створені для обробки натискань на дві кнопки. Тепер вам до цих функцій необхідно дописати необхідний код.

У функцію для включення світлодіода (`LED-ON button's function`) скопіюйте наступний фрагмент коду. У цьому фрагменті коду функція `fprintf(x,1)` використовується передачі символу `'1'` по послідовному каналу зв'язку в плату Arduino. Якщо ви подивитеся код програми для плати Arduino, то ви побачите, що при прийомі `'1'` по послідовному каналу зв'язку вона подає напругу високого рівня (HIGH) на свій 13-й контакт, тобто включає вбудований в плату світлодіод.

У функцію вимкнення світлодіода (`LED-OFF button's function`) скопіюйте наступний наведений фрагмент коду.

У цьому фрагменті коду функція `fprintf(x,0)` використовується передачі символу `'0'` по послідовному каналу зв'язку в плату Arduino (рис. 4.2.12).

Якщо ви звернете увагу на код програми для плати Arduino, то ви побачите що при прийомі `'0'` по послідовному каналу зв'язку вона подає напругу низького рівня (`LOW`) на свій 13-й контакт, тобто вимикає вбудований в плату світлодіод.

```
80
81      % --- Executes on button press in turnonled.
82      function turnonled_Callback(hObject, eventdata, handles)
83      % hObject    handle to turnonled (see GCBO)
84      % eventdata  reserved - to be defined in a future version of MATLAB
85      % handles    structure with handles and user data (see GUIDATA)
86 -    global x;
87 -    fprintf(x,1);
88
89
90      % --- Executes on button press in turnoffled.
91      function turnoffled_Callback(hObject, eventdata, handles)
92      % hObject    handle to turnoffled (see GCBO)
93      % eventdata  reserved - to be defined in a future version of MATLAB
94      % handles    structure with handles and user data (see GUIDATA)
95 -    global x;
96 -    fprintf(x,0);
97
```

Рис. 4.2.11. Вікно редактора системи MATLAB

Після завершення внесення змін до програми (в `*.m` файл) натисніть кнопку для її запуску (рис. 4.2.13).

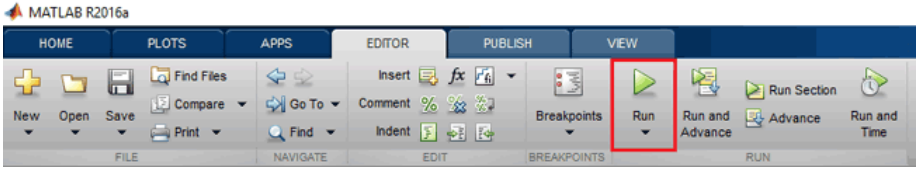


Рис. 4.2.13. Вікно системи MATLAB

Системі MATLAB може знадобитися кілька секунд на запуск цього файлу, не натискайте жодну кнопку на графічному інтерфейсі, доки не світиться індикація `BUSY` (система зайнята) у лівому нижньому кутку екрана (рис. 4.2.14).

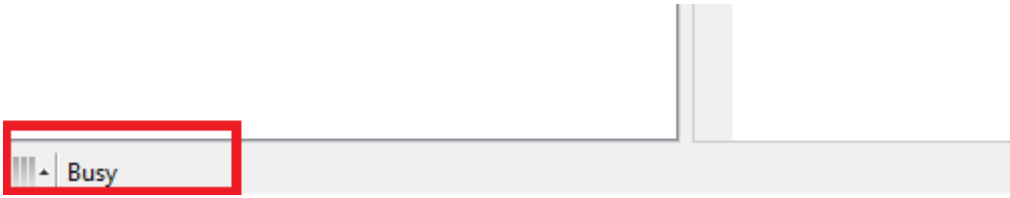


Рис. 4.2.14. Вікно системи MATLAB

Після успішного запуску програми ви можете натискати створені кнопки, щоб вмикати та вимикати світлодіод на платі Arduino (він підключений до її 13-го контакту).

Лабораторна робота № 5
Реалізація послідовного зв'язку по протоколу Modbus RS-485 із платою Arduino (ведучий)

Мета роботи: Реалізувати передачу даних по протоколу Modbus RS-485 від плати Arduino (ведучий) до персонального комп'ютера (ведений).

Прилади та компоненти:

апаратне забезпечення

плата Arduino Uno;
MAX485 TTL to RS485 Converter Module;
USB to RS-485 Converter Module;
кнопка – 2 шт.
резистор 10 кОм – 2 шт;
РК-дисплей 16×2;
потенціометр 10 кОм – 2 шт;

програмне забезпечення:

Motbus Slave.

5.1. Теоретичні відомості

Modbus – протокол, який працює за принципом «клієнт–сервер». Протокол широко використовується у промисловості для міжмашинної взаємодії і не тільки. Modbus може використовуватися для передачі даних через послідовні лінії зв'язку RS-485, RS-422, RS-232, а також через мережі TCP/IP.

Modbus є програмним (не апаратним) протоколом і складається із двох частин: Modbus Master (ведучий) і Modbus Slave (ведений). У мережі Modbus RS-485 може бути один ведучий і 127 ведених пристроїв, кожний із яких має унікальний адрес від 1 до 127.

Modbus частіше всього використовується у програмованих логічних контролерах (PLC – Programmable Logic Controllers).

Найбільш поширені 3 версії даного протоколу:

MODBUS RTU;
MODBUS ASCII;

MODBUS/TCP.

Різниця між протоколами Modbus ASCII та Modbus RTU. По суті, це практично одинакові протоколи. Тільки у протоколе Modbus RTU дані передаються послідовно у двійковому коде, а у Modbus ASCII – в ASCII кодах. В лабораторній роботі будемо використовувати Modbus RTU.

5.1.1. Принципи роботи інтерфейсу послідовного зв'язку RS-485

RS-485 представляє собою асинхронний інтерфейс послідового зв'язку, якому не потрібно для своєї роботи імпульсів синхронізації. Для передачі двійкових даних від одного до іншого інтерфейс використовує диференціальний сигнал. Диференціальний сигнал представляє собою спосіб передані інформації за допомогою двох протифазних сигналів. У даному методі один електричний сигнал передається у вигляді диференціальної пари сигналів, кожен по своєму провіднику, але один є інвертованим сигналом відносно іншого протилежної полярності. Пара провідників може представляти собою виту пару. Приймач диференціального сигналу реагує на різницю між двома сигналами, а не на різницю між одним провідником і потенціалом землі.

У нашому випадку диференціальний сигнал реалізується за допомогою додатньої та від'ємної напруги 5 В. Інтерфейс RS-485 забезпечує напівдуплексний зв'язок (Half-Duplex) при використанні 2-х ліній (проводів) та повноцінний дуплексний зв'язок (Full-Duplex) при використанні 4-х ліній (проводів).

Основні особливості даного інтерфейсу:

1. Максимальна швидкість передачі даних у інтерфейсе RS-485 – 30 Мбіт/с.
2. Максимальна дистанція – 1200 м, що значно більше ніж у інтерфейсу RS-232.
3. Основною перевагою інтерфейсу RS-485, у порівнянні з RS-232, є використання декількох ведених (multiple

slave) при одному ведучому (single master), в той час як RS-232 підтримує тільки одного веденого.

4. Хороша завадостійкість внаслідок використання диференціального сигналу.
5. RS-485 забезпечує вищу швидкість передачі у порівнянні із інтерфейсом I2C.

5.1.2. Використання інтерфейсу RS-485 в Arduino

Для реалізації інтерфейсу RS-485 із платою Arduino використаємо модуль 5 V MAX485 TTL to RS-485, в основі якого лежить мікросхема Maxim MAX485. Модуль є двохнапрямленим та забезпечує послідовний зв'язок на відстань до 1200 м швидкістю передачі даних 2,5 Мбіт/с у напівдуплексному режимі.

Еклектичне живлення модуля 5 V MAX485 TTL to RS-485 5 В і логічний рівень також 5 В, що дозволяє без проблем підключати його до плат Arduino.

Модуль має такі особливості:

– напруга живлення 5 В;
– має в своєму складі чіп MAX485;
– відрізняється низьким енергоспоживанням;
– всіма його контактами можна керувати за допомогою мікроконтролера;
– розміри плати модуля: 44×14 мм.

Зовнішній вигляд модуля RS-485 представлено на рис. 5.1.1.

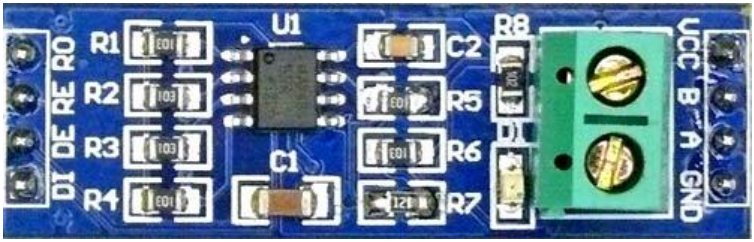


Рис. 5.1.1. Модуль RS-485

Призначення контактів (розпінування) модуля RS-485 наведено у наступній табл. 5.1.1.

Таблиця 5.1.1

Призначення контактів (розпінування) модуля RS-485

Контакт	Призначення
VCC	5 В
A	вхід/вихід лінії RS-485
B	вхід/вихід линии RS-485
GND	GND (0 В)
R0	вихід приймача (RX pin)
RE	дозвіл роботи приймача
DE	дозвіл роботи передатчика
DI	вхід передавача (TX pin)

Контакти на модулі RS-485 розташовані оптимально – з одного боку до модуля підключається пристрій, з іншого – лінія.

5.1.3. Модуль перетворення USB в RS-485

На рис. 5.1.2 представлено зовнішній вигляд адаптера (модуля перетворення) USB в RS-485. Модуль може працювати в різних операційних системах та забезпечує реалізацію інтерфейсу RS-485 за допомогою одного з COM-портів ПК. Модуль є пристроєм plug-and-play та живиться від USB-порту – ніякого додаткового живлення не потрібно.

У комп’ютері він доступний як послідовний COM-порт для використання різними програмами. Модуль забезпечує напівдуплексний зв’язок за допомогою інтерфейсу RS-485. Швидкість передачі – від 75 до 115200 бод/с, максимальна – до 6 Мбіт/с.

Рис. 5.1.2. Модуля перетворення USB в RS-485

У мережі Інтернет можна знайти багато програмного забезпечення, здатного працювати з цим модулем. У лабораторній роботі будемо використовувати програму Modbus Slave.

5.1.4. Програмне забезпечення Modbus Slave

Програмне забезпечення дозволяє приймати дані від будь-якого ведучого пристрою, який працює за протоколом Modbus, із використанням порту послідовного зв’язку.

Перед використанням цієї програми необхідно ознайомитися з наступними термінами.

Slave ID (ідентифікатор веденого)

Кожному веденому пристрою в мережі призначається унікальна адреса в діапазоні від 1 до 127. Коли ведучий пристрій запитує дані, перший байт, який він передає, містить адресу веденого пристрою. Завдяки цьому кожен ведений пристрій знає, чи варто йому відповідати на цей запит чи ні.

Регістри Modbus

Регістри прапорів (позначок) (Coils) зберігають однобітні значення – тобто можуть знаходитися у стані «0» або «1». Такі регістри можуть означати поточний стан виходу (включено реле). Назва «coil» буквально означає обмотку-актюатор електромеханічного реле. Регістри прапорів допускають як читання, так і запис. Мають номери від 1 до 9999.

Дискретні входи (Discrete Inputs) також є однобітними регістрами, які описують стан входу пристрою (наприклад, подано напругу – «1»). Ці регістри підтримують лише читання. Мають номери від 10001 до 19999.

Регістри введення (Input Registers) – 16-бітові регістри, що використовуються для вводу інформації. Ці регістри підтримують лише читання. Мають номери від 30001 до 39999.

Регістри збереження (Holding Registers) представлені двобайтовим словом і можуть зберігати значення від 0 до 65535 (0×0000–0×FFFF). Реєстри збереження підтримують як читання, так і запис (для зберігання налаштувань). Мають номери від 40001 до 49999.

Function code (функціональний код)

Другий байт, що передається ведучим пристроєм, містить функціональний код. Цей код визначає дію, яку необхідно виконати (рахувати, записати тощо). Дії згруповані за таблицями. У протоколі Modbus існує чотири таблиці із даними:

Таблиця	Тип елемента	Тип доступу
Дискретні входи (Discrete Inputs)	один біт	лише читання
Регістри прапорів (Coils)	один біт	читання та запис
Регістри вводу (Input Registers)	16-бітове слово	лише читання
Регістри збереження (Holding Registers)	16-бітове слово	читання та запис

У реальній практиці найчастіше зустрічаються пристрої, у яких є лише таблиця Holding Registers, іноді об'єднана з таблицею Input Registers.

Для доступу до цих таблиць існує ряд стандартних функцій ModBus:

Читання:

1 (0×01) – читання значень із кількох регістрів прапорів (Read Coil Status).

2 (0×02) – читання значень із кількох дискретних входів (Read Discrete Inputs).

3 (0×03) – читання значень із кількох регістрів зберігання (Read Holding Registers).

4 (0×04) – читання значень із кількох регістрів введення (Read Input Registers).

Запис одного значення:

5 (0×05) – запис одного прапора (Force Single Coil).

6 (0×06) – запис значення одного регістру збереження (Preset Single Register).

Запис кількох значень:

15 (0×0F) – запис значень у кілька регістрів прапорів (Force Multiple Coils).

16 (0×10) – запис значень у кілька регістрів зберігання (Preset Multiple Registers)

Найбільш часто використовувані на практиці функції (функціональні коди) ModBus це 3, 6 і 16 («Read Holding Registers», «Preset Single Register» і «Preset Multiple Registers» відповідно).

CRC

CRC розшифровується як Cyclic Redundancy check та перекладається як «циклічний надлишковий код». Це два байти, які додаються до кожного повідомлення протоколу Modbus, що передається, для виявлення помилок.

5.2. Порядок виконання роботи

5.2.1. Схема проекту

Схема реалізації послідовного зв’язку за протоколом Modbus RS-485 із платою Arduino Uno (ведучою) представлена на рис. 5.2.1.

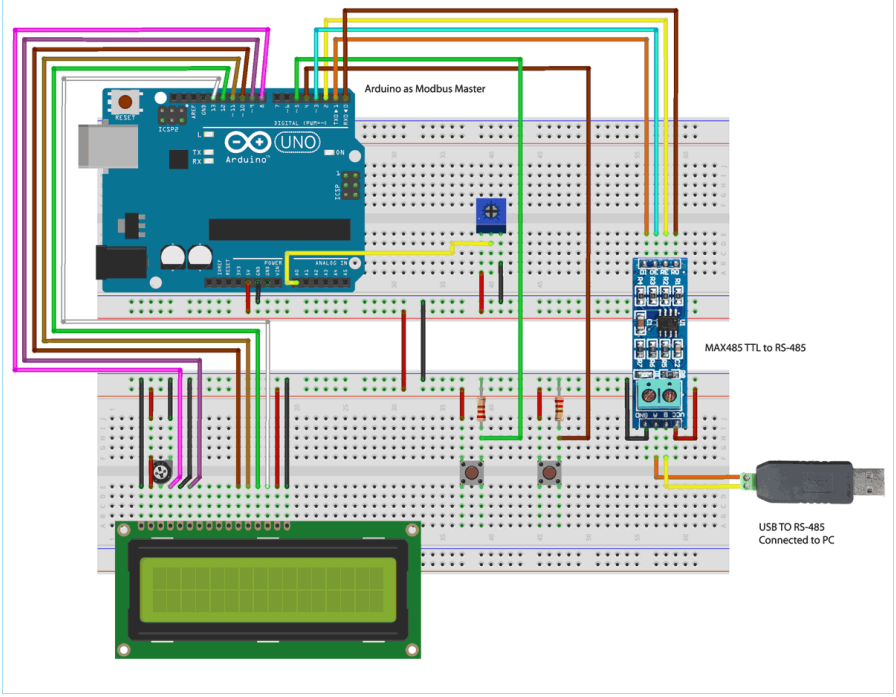


Рис. 5.2.1. Схема проєкту

В табл 5.2.1 представлені необхідні з’єднання між платою Arduino Uno (ведучою) і модулем MAX485 TTL to RS-485.

В табл. 5.2.2 представлені необхідні з’єднання між модулями MAX485 TTL to RS-485 та USB to RS-485.

В табл. 5.2.3 представлені необхідні з’єднання між платою Arduino Uno та РК-дисплеєм 16×2.

Таблиця 5.2.1

З’єднання між платою Arduino Uno і модулем MAX485 TTL to RS-485

Arduino Uno	Модуль MAX485 TTL to RS-485
0(RX)	RO
1(TX)	DI
3	DE
2	RE
+ 5 В	VCC
GND	GND

Таблиця 5.2.2

З’єднання між модулями MAX485 TTL to RS-485 та USB to RS-485

MAX485 TTL to RS485	USB to RS-485 (підключений до ПК)
A	A
B	B

Дві кнопки з підтягуючими резисторами 10 кОм підключені до контактів 4 і 5 плати Arduino Uno. Із середнього контакту потенціометра 10 кОм подається напруга на аналоговий контакт A0 плати Arduino Uno.

На рис. 5.2.2 представлена конструктивна реалізація схеми проєкту.

Таблиця 5.2.3

З’єднання між платою Arduino Uno та РК-дисплеєм 16×2

РК-дисплей 16x2	Плата Arduino Uno
VSS	GND
VDD	+5 В
V0	до потенціометру для керування яскравістю РК-дисплея
RS	8
RW	GND
E	9
D4	10
D5	11
D6	12
D7	13
A	+ 5 В
K	GND

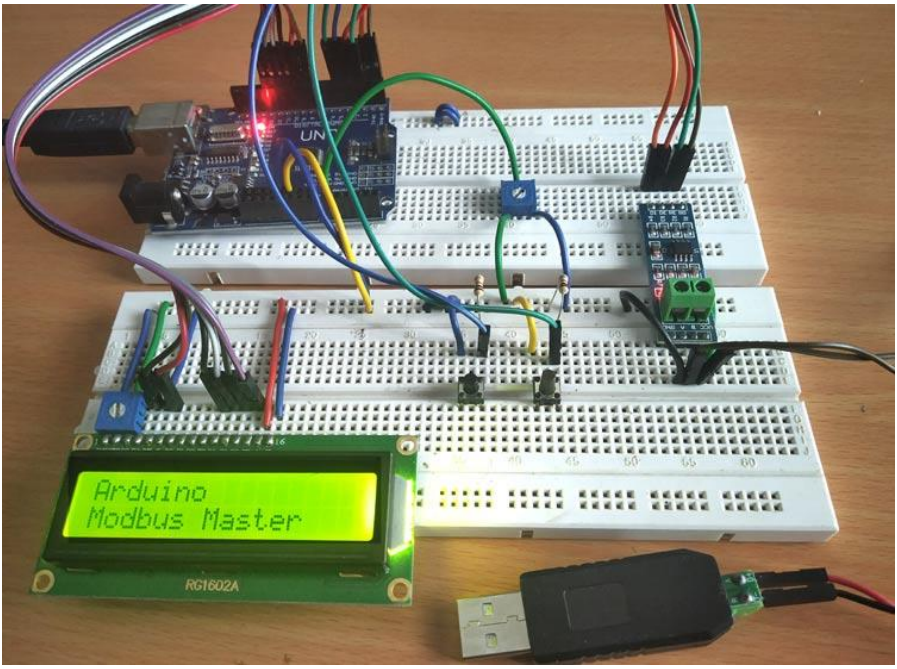


Рис. 5.2.2. Конструктивна реалізація схеми проєкту

5.2.2. Програма для плати Arduino Uno

Для того, щоб плата Arduino могла працювати в якості ведучого пристрою за протоколом Modbus, будемо використовувати бібліотеку Modbus Master. В нашому проєкті до плати Arduino Uno підключені дві кнопки і потенціометр, щоб з їх допомогою передавати дані веденому пристрою Modbus (в нашому випадку програмі Modbus Slave).

Бібліотека <ModbusMaster.h> використовується для зв’язку за протоколом Modbus RS-485 ведучим або веденому пристроєм за допомогою протоколу RTU.

Повний код програми приведений в кінці лабораторної роботи.

Розглянемо основні фрагменти коду програми.

В програмі необхідно підключити бібліотеки для роботи із протоколом Modbus та РК-дисплеєм.

```
#include <ModbusMaster.h>
#include <LiquidCrystal.h>
```

Дамо осмислені імена контактам плати Arduino Uno, до яких підключений модуль MAX485 TTL to RS-485

```
#define MAX485_DE      3
#define MAX485_RE_NEG  2
```

Ініціалізуємо об'єкт `node` класу `ModbusMaster`.

Запрограмуємо дві функції: `preTrasnmission()` і `postTrasmission()`, для того щоб подавати на контакти `RE` та `DE` модуля MAX485 TTL to RS-485 напругу високого (HIGH) або низького (LOW) рівня, що дозволяє передачу або прийом даних.

```
void preTransmission()
{
  digitalWrite(MAX485_RE_NEG, 1);
  digitalWrite(MAX485_DE, 1);
}
void postTransmission()
{
  digitalWrite(MAX485_RE_NEG, 0);
  digitalWrite(MAX485_DE, 0);
}
```

Далі в функції `void setup()` конфігуруємо РК-дисплей для роботи у режимі 16×2. Виведемо на РК-дисплей вітальне повідомлення, а потім очищуємо його.

```
lcd.begin(16,2);
lcd.print("CIRCUIT DIGEST");
```

```
delay(3000);
lcd.clear();
lcd.print("Arduino");
lcd.setCursor(0,1);
lcd.print("Modbus Master");
delay(3000);
lcd.clear();
```

Контакти RE і DE встановимо в режим роботи на вивід даних (OUTPUT), а контакти 4 і 5 (до ним підключені кнопки) – в режим роботи на ввід даних (INPUT).

Першочергово на контакти DE і RE модуля MAX-485 TTL to RS-485 подаємо напругу низького рівня (LOW).

```
digitalWrite(MAX485_RE_NEG,0);
digitalWrite(MAX485_DE,0);
```

Встановіть швидкість передачі для послідовного зв'язку 115200 бод та ініціалізуємо послідовний зв'язок за протоколом Modbus із веденим пристроєм із адресом 1 (slave ID).

```
Serial.begin(115200);
node.begin(1, Serial);
```

Після цього перевіримо чи правильно проведена конфігурація прийопередавача RS-485.

```
node.preTransmission(preTransmission);
node.postTransmission(postTransmission);
```

Наступний крок – у функції `void oop()` виконаємо таку дії.

1. Спочатку зчитуємо значення напруги із аналогового контакту A0 – до нього підключений потенціометр.

```
float value=analogRead(A0)
```

2. Значення із виходу АЦП (аналого-цифрового перетворювача) записуємо у регістр 0×40000 щоб передати веденому пристрою (Slave) Modbus.

```
node.writeSingleRegister(0x40000,value);
```

3. Відобразимо дане значення на екране РК-дисплею 16×2.

```
lcd.setCursor(0,0);
lcd.print("POT Val :");
lcd.print(value);
```

4. Зчитуємо стан двох кнопок.

```
int a=digitalRead(4);
int b=digitalRead(5);
```

5. Якщо перша кнопка натиснута, то для передачі до веденого пристрою Modbus запише у регістр 0×40001 значення 1, а якщо не натиснута, то значення 0. Аналогічно, якщо друга кнопка натиснута, то для передачі до веденого пристрою Modbus запише у регістр 0×40002 значення 1, а якщо не натиснута, то значення 0.

```
  if (a==1)
  {
    node.writeSingleRegister(0x40001,1);
    lcd.setCursor(0,1);
    lcd.print("S1: 1");
  }
  else
  {
    node.writeSingleRegister(0x40001,0);
    lcd.setCursor(0,1);
    lcd.print("S1: 0");
  }
  if (b==1)
```

```
  {
    node.writeSingleRegister(0x40002,1);
    lcd.setCursor(8,1);
    lcd.print("S2: 1");
  }
  else
  {
    node.writeSingleRegister(0x40002,0);
    lcd.setCursor(8,1);
    lcd.print("S2: 0");
}
```

5.3. Тестування роботи проєкту

Після того як зібрана схема проєкту і завантажена програма в контролер плати Arduino Uno, під'єднайте адаптер USB у RS-485 до ПК на якому встановлена програма Modbus Slave.

Відкрийте диспетчер пристроїв у Windows і визначте COM-порт до якого підключений адаптор USB в RS-485.

Запустіть програму Modbus Slave.

1. При першому запуску в програмі Modbus Slave буде повідомлення No Connection (немає з'єднання) (рис. 5.3.1).

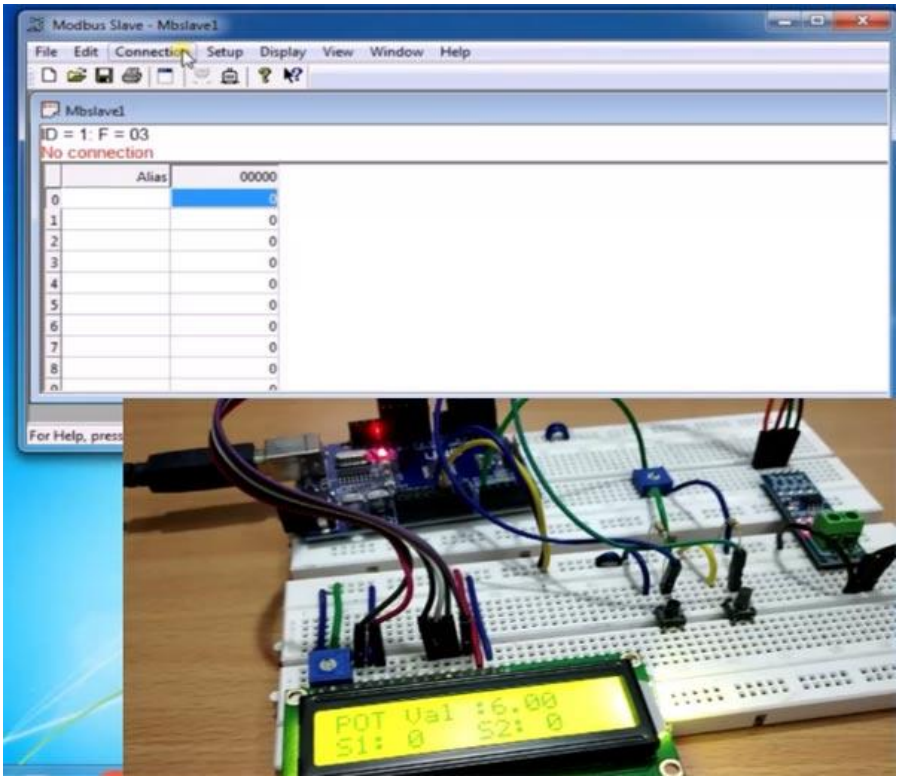


Рис. 5.3.1. Перший запуск програми Modbus Slave

2. Відкрийте пункт меню Connection –> Connect як представлено на рис. 5.3.2.

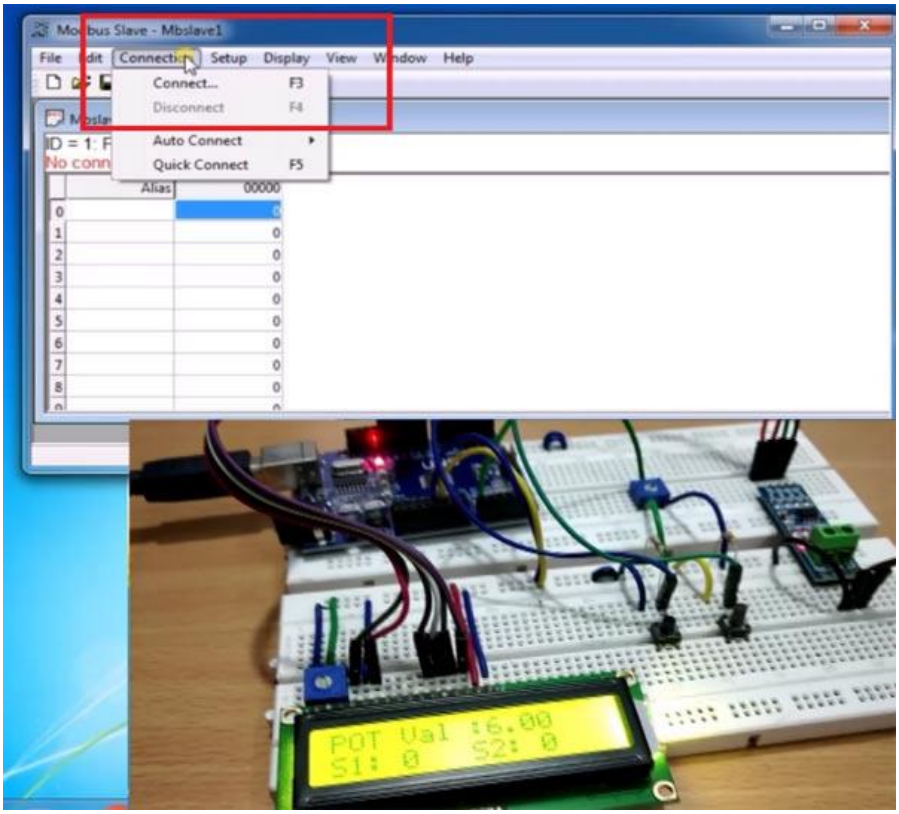


Рис. 5.3.2. Пункт меню Connection програми Modbus Slave

3. У програмі необхідно зробити такі налаштування з'єднання (рис. 5.3.3):
   – як з'єднання (Connection) виберіть Serial Port (послідовний порт);
   – виберіть COM-порт до якого підключений адаптер USB в RS-485;
   – у налаштуваннях послідовного з'єднання вкажіть: швидкість – 115200 (використовували у програмі для Arduino), 8 біт даних, немає біта парності, 1 стоповий біт, режим (Mode) – RTU.

Натискаємо OK і повідомлення No Connection має зникнути.

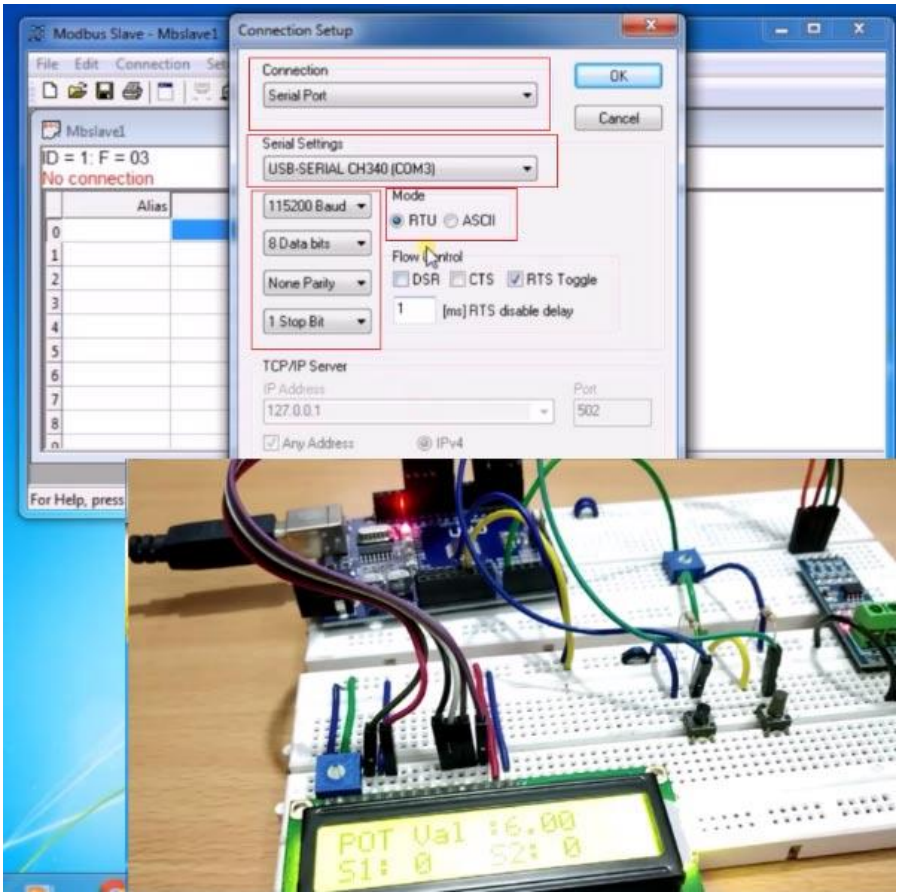


Рис. 5.3.3. Вікно Connection Setup програми Modbus Slave

4. Відкрийте пункт меню Setup –> Slave Definition (рис. 5.3.4).

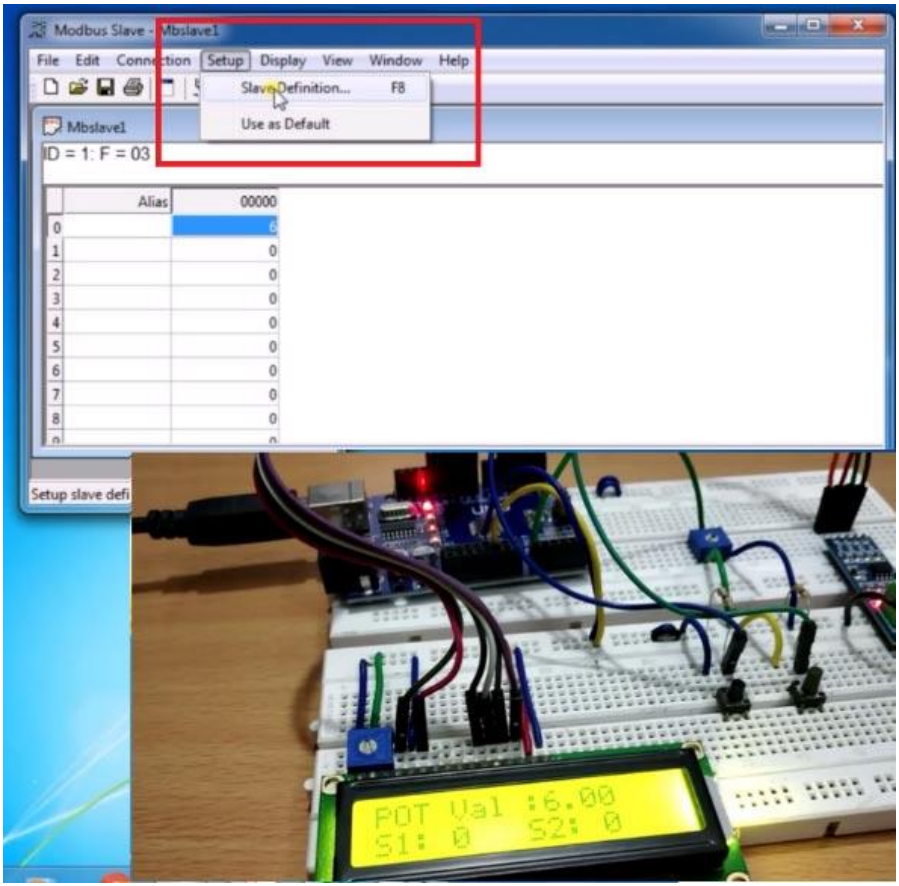


Рис. 5.3.4. Пункт меню Setup програми Modbus Slave

5. В якості Slave ID (адрес веденого) введіть 1, а в якості функціонального коду (function) виберіть «3 Holding Registers». У полі адресу введіть 0 і натисніть OK (рис. 5.3.5).

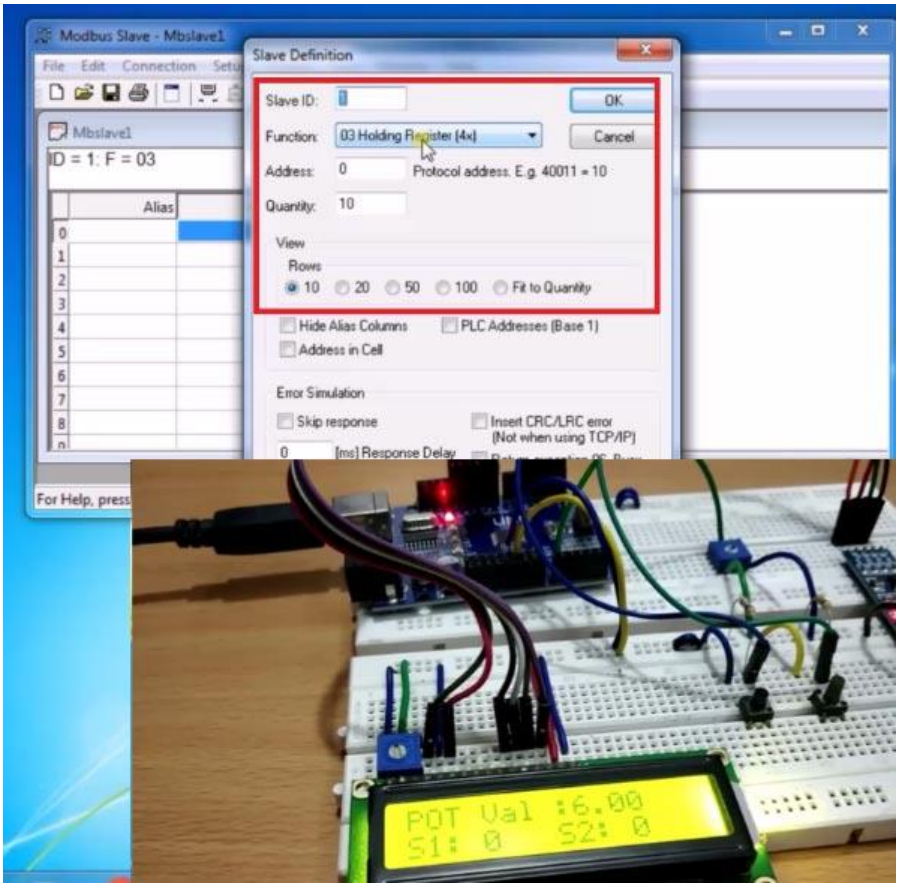


Рис. 5.3.5. Вікно Slave Definition програми Modbus Slave

6. Перевірте, що ID у вас 1, а F – 03. В нашому проєкті лабораторної роботи ми використовуємо 3 регістри: 0 – значення із виходу потенціометра (після АЦП), 1 – стан першої кнопки, 2 – стан другої кнопки (рис. 5.3.6).

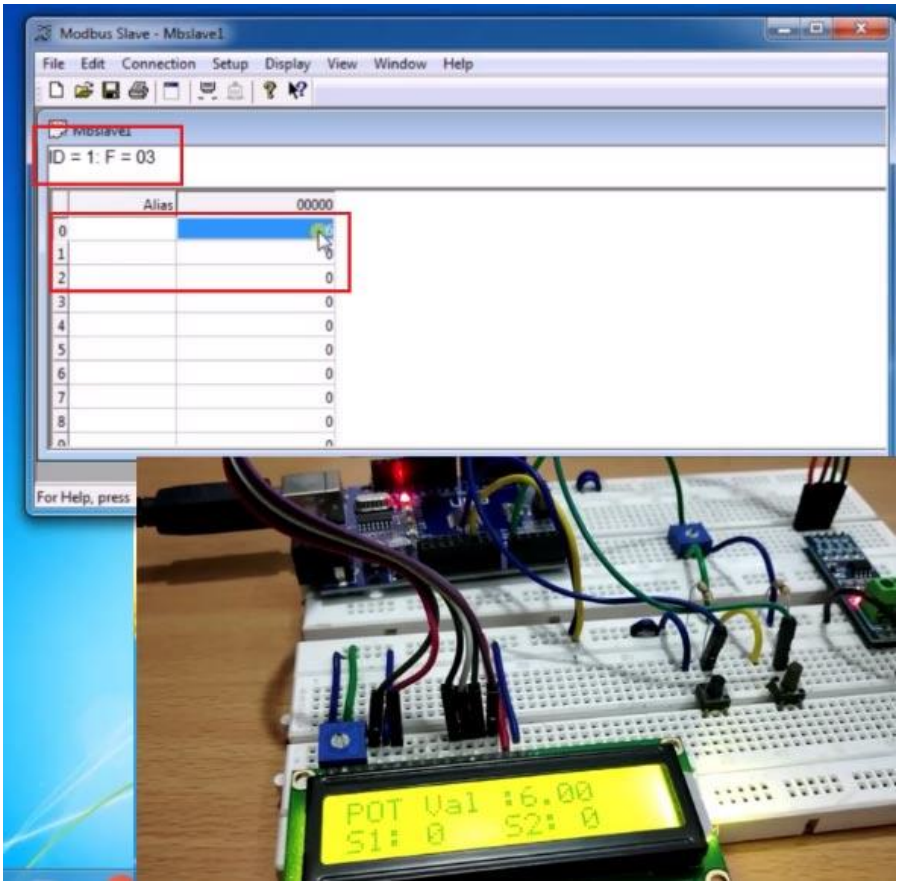


Рис. 5.3.6. Вікно стану регістрів програми Modbus Slave

7. Натисніть кнопку 2 – побачимо, що у третій строчці з’явиться значення 0, а в першій строчці буде відображатися деяке значення, що характеризує текучий стан потенціометра (рис. 5.3.7).

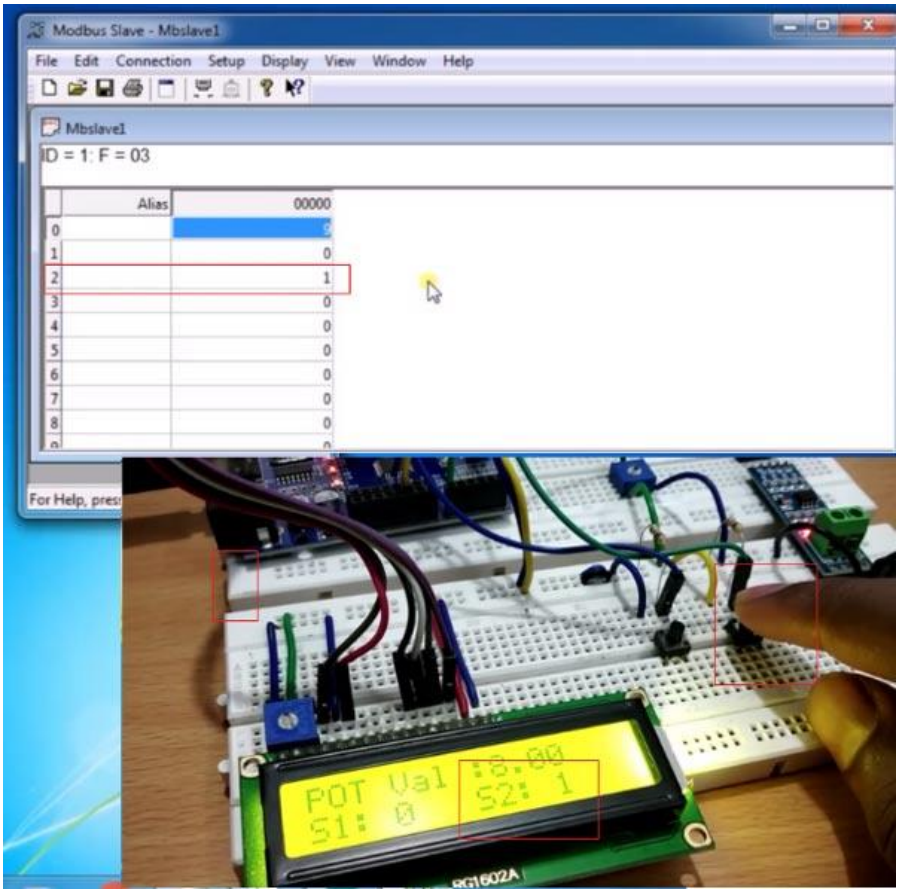


Рис. 5.3.7. Вікно стану регістрів програми Modbus Slave

8. Якщо натиснути кнопку 1, то значення 1 з’явиться у другій строчці (рис. 5.3.8).

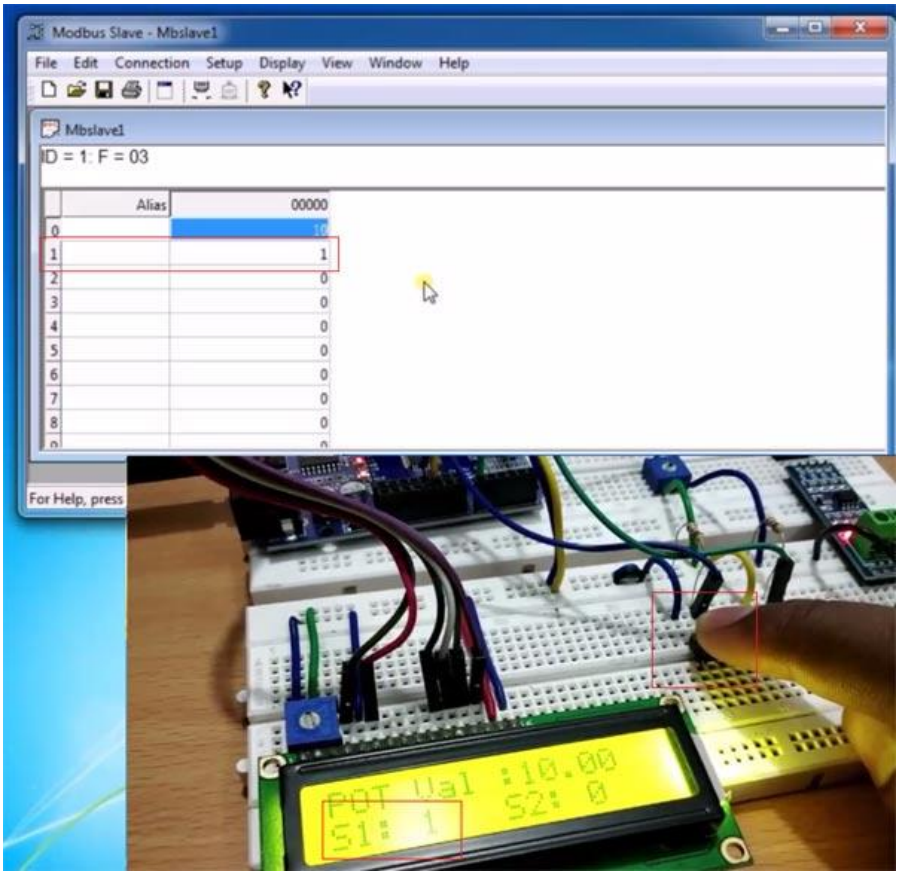


Рис. 5.3.8. Вікно стану регістрів програми Modbus Slave

9. Натисніть обидві кнопки і значення 1 буде у другій і третій строчці (рис. 5.3.9).

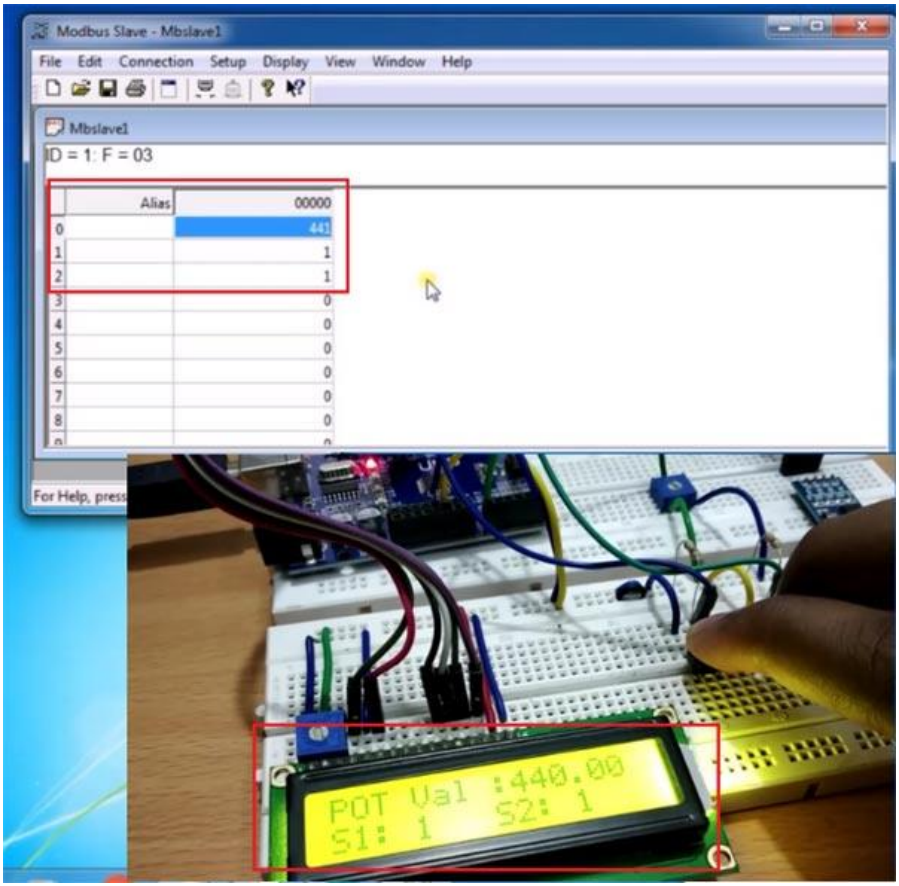


Рис. 5.3.9. Вікно стану регістрів програми Modbus Slave

10. Обертайте ручку потенціометра і слідкуйте за значеннями у першій строчці (рис. 5.3.10).

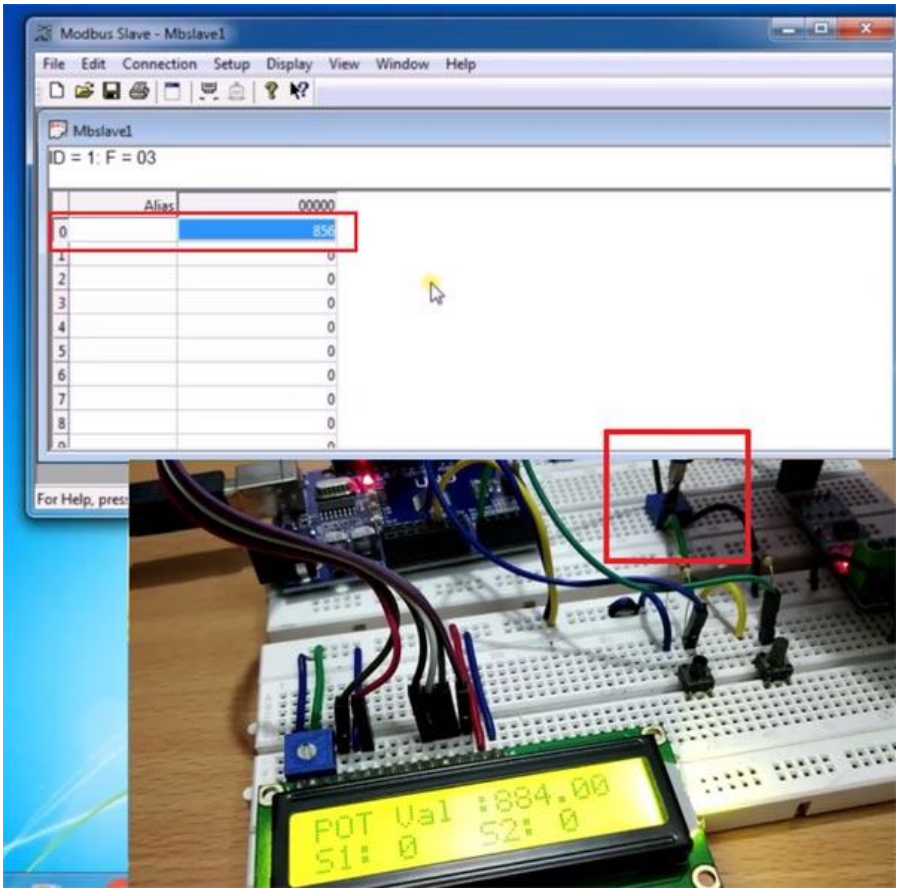


Рис. 5.3.10. Вікно стану регістрів програми Modbus Slave

Вихідний код скетча для Arduino Uno

```
#include <ModbusMaster.h> // бібліотека для
ModbusMaster
#include <LiquidCrystal.h> // бібліотека для
роботи із РК-дисплеєм
#define MAX485_DE      3
#define MAX485_RE_NEG  2
```

```
ModbusMaster node; // об’єкт node класу
ModbusMaster
LiquidCrystal lcd(8,9,10,11,12,13);  //Object
lcd for class Liquidcrystal with LCD pins (RS,
E, D4, D5, D6, D7) that are connected with
Arduino UNO.
void preTransmission() // функції для
встановлення стану контактів DE & RE модуля
RS-485
{
  digitalWrite(MAX485_RE_NEG, 1);
  digitalWrite(MAX485_DE, 1);
}
void postTransmission()
{
  digitalWrite(MAX485_RE_NEG, 0);
  digitalWrite(MAX485_DE, 0);
}
void setup()
{
  lcd.begin(16,2);
  lcd.print("CIRCUIT DIGEST");
  delay(3000);
  lcd.clear();
  lcd.print("Arduino");
  lcd.setCursor(0,1);
  lcd.print("Modbus Master");
  delay(3000);
  lcd.clear();

  pinMode(MAX485_RE_NEG, OUTPUT);
  pinMode(MAX485_DE, OUTPUT);

  pinMode(4,INPUT);
  pinMode(5,INPUT);

  digitalWrite(MAX485_RE_NEG, 0);
```

```
  digitalWrite(MAX485_DE, 0);
  Serial.begin(115200); // швидкість передачі
115200 бод
  node.begin(1, Serial); // Slave ID (адрес
веденого) - 1
  node.preTransmission(preTransmission); //
Callback for configuring RS-485 Transreceiver
correctly
  node.postTransmission(postTransmission);
}
void loop()
{
  float value=analogRead(A0);

  node.writeSingleRegister(0x40000,value);
    //записываем value в 0x40000 holding
register

  lcd.setCursor(0,0);
  lcd.print("POT Val :");
  lcd.print(value);
  int a=digitalRead(4); // зчитуємо стан
кнопок
  int b=digitalRead(5);
  if (a==1)
  {
    node.writeSingleRegister(0x40001,1); //
записуємо 1 в 0x40001 holding register
    lcd.setCursor(0,1);
    lcd.print("S1: 1");
  }
  else
  {
    node.writeSingleRegister(0x40001,0); //
записуємо 0 в 0x40001 holding register
    lcd.setCursor(0,1);
    lcd.print("S1: 0");
```

```
  }
  if (b==1)
  {
    node.writeSingleRegister(0x40002,1); //
записуємо 1 в 0x40002 holding register
    lcd.setCursor(8,1);
    lcd.print("S2: 1");
  }
  else
  {
    node.writeSingleRegister(0x40002,0); //
записуємо 0 в 0x40002 holding register
    lcd.setCursor(8,1);
    lcd.print("S2: 0");
  }

}
```

Література

1. Stateflow. URL: https://www.mathworks.com/help/stateflow/index.html?searchHighlight=Stateflow%20Diagram&s_tid=srchtitle_support_results_1_Stateflow%20Diagram
2. Formal Verification of Simulink/Stateflow Diagrams: A Deductive Approach / N. Zhan, S. Wang, H. Zhao. Springer; Softcover reprint of the original 1st ed, 2017. 273 p.
3. FPGAs: Fundamentals, Advanced Features, and Applications in Industrial Electronics / J. J. R. Andina, E. de la T. Arnanz, M. D. V. Peiia. Taylor & Francis, 2020. 249 p.
4. Pakdel M. Coding Modbus TCP/IP for Arduino. Elektor, 2024. 208 p.
5. Pleasure T. Modbus RTU Protocol Example: How To Design A Modbus TCP/RTU. Independently Published, 2021. 40 p.