Internet of Things for Industry and Human Applications

Architectures and Embedded Platform Based Development of Internet / Web of Things Systems

ALIoT

Internet of Things for Industry and Human Applications

Architectures and Embedded Platform Based Development of Internet / Web of Things Systems

PRACTICUM

Funded by the Erasmus+ Programme of the European Union

Ministry of Education and Science of Ukraine
National Aerospace University "Kharkiv Aviation Institute"

A.P. Plakhtyeyev, E.V. Babeshko, V.A. Tkachenko,
Yu. V. Zdorovets

Internet of things for industry and human applications

Architectures and Embedded Platform Based Development of Internet / Web of Things Systems

Laboratory works

Edited by V. S. Kharchenko

Project
ERASMUS+ ALIOT "Internet of Things:
Emerging Curriculum for Industry and Human Applications"
(573818-EPP-1-2016-1-UK-EPPKA2-CBHE-JP)

2019

UDC 004.415/.416].031.6(076.5)=111
A87
Reviewers: Prof., DrS Volodymyr Mokhor, IPME-NASU, Kyiv, Ukraine
Dr. Olga Kordas, KTH University, Stockholm, Sweden
Viktor Kordas, KTH University, Stockholm, Sweden

A87 A.P. Plakhtyeyev. E.V. Babeshko, V.A. Tkachenko, J.V. Zdorovets. **Architectures and Embedded Platform Based development of Internet / Web of Things systems: Laboratory works** / V.S. Kharchenko (edit.) - Ministry of Education and Science of Ukraine, National Aerospace University "KhAI", 2019. – 147 p.

ISBN 978-617-7361-98-4.

The book contains materials for practicum of MSc course "Fundamentals of Internet of Things" developed in frameworks of project "Internet of Things: Emerging Curriculum for Industry and Human Applications /ALIOT" (Project Number: 573818-EPP-1-2016-1-UK-EPPKA2-CBHE-JP, 2016-2019) funded by EU Program ERASMUS+. It's described laboratory works for module "Architectures and platforms of IoT/WoT systems". Practicum includes 9 labs for research of diffrent platforms and controllers such as PLCNext which are used for creation of IIoT systems.

The book prepared by Ukrainian university teams with support of EU academic colleagues of the ALIOT consortium. The book is intended for MSc and PhD students studying IoT/WoT technologies, software and computer engineering and science. It could be useful for lecturers of universities and training centers, researchers and developers of IoT systems.

Fig.: 53. Ref.: 57. Tables: 7.

Approved by Academic Council of National Aerospace University "Kharkiv Aviation Institute" (record № 4, December 19, 2018).

ISBN 978-617-7361-98-4.

Міністерство освіти і науки України
Національний аерокосмічний університет
ім. М.Є. Жуковського «Харківський авіаційний Інститут»

А.П. Плахтєєв, Є.В.Бабешко, В.А. Ткаченко, Ю.В. Здоровець

Інтернет речей для індустріальних і гуманітарних застосунків

Архітектури та розроблення систем Інтернету / Вебу речей на основі вбудованих платформ

Лабораторні роботи

Редактор Харченко В.С.

Проект
ERASMUS+ ALIOT "Інтернет речей:
нова освітня програма для потреб промисловості та суспільства"
(573818-EPP-1-2016-1-UK-EPPKA2-CBHE-JP)

2019

УДК 004.415/.416].031.6(076.5)=111
А87
Рецензенти: Д.т.н., проф. Володимир Мохор, ІПМЕ НАНУ, Київ, Україна
Др. Ольга Кордас, KTH University, Стокгольм, Швеція
Віктор Кордас, KTH University, Stockholm, Sweden

А87 А.П. Плахтєєв, Є.В. Бабешко, В.А. Ткаченко, Ю.В. Здоровець. **Архітектури та розроблення систем Інтернету / Вебу Речей на основі вбудованих платформ. Лабораторні роботи** / За ред. В.С. Харченка. Міністерство освіти і науки України, Національний аерокосмічний університет ХАІ, 2019. - 147 с.

ISBN 978-617-7361-98-4.

Книга містить матеріали практичної частини магістерського курсу «Основи Інтернету речей», розробленого в рамках проекту Internet of Things: Emerging Curriculum for Industry and Human Applications / ALIOT, 573818-EPP-1-2016-1-UK-EPPKA2- CBHE-JP, 2016-2019, що фінансується програмою ЄС ERASMUS+. Книга складається з 9 лабораторних робіт для модуля цього курсу «Архітектури та платформи Інтернету/Вебу речей. Лабораторні роботи включають завдання для вивчення платформ і розроблення систем індустріального Інтернету і Вебу речей з використанням Arduino, PLCNext та інших засобів.

Книга підготовлена українськими університетськими командами за підтримки колег з академічних закладів країн ЄС, що входять до консорціуму проекту ALIOT.

Книга призначена для магістрантів і аспірантів, які вивчають технології IoT, програмну і комп'ютерну інженерію, комп'ютерні науки. Може бути корисною для викладачів університетів і навчальних центрів, розробників систем IoT.

Рис.: 53. Посилань: 57. Таблиць: 7.

Рекомендовано до видання вченою радою Національного аерокосмічного університету імені М.Є. Жуковського «Харківський авіаційний інститут» (протокол № 4 від 19 грудня 2018 г.).

ISBN 978-617-7361-98-4.

ПЕРЕЛІК СКОРОЧЕНЬ

АЛУ– арифметико-логічний пристрій

АЦП – аналого-цифровий перетворювач

МК – мікроконтролер

МПС – мікропроцесорна система

ОЗП – оперативно- запам’ятовувальний пристрій

ПЗ – програмне забезпечення

ТЛ – таймер-лічильник

ШІМ – широтноімпульсна модуляція

ЦАП – цифро-аналоговий перетворювач

IoT – Internet of Things (Інтернет речей)

LCD - liquid crystal display (рідкокристалічний дисплей)

SPI – Serial Peripheral Interface (послідовний периферійний інтерфейс)

UART - universal asynchronous receiver/transmitter

WoT- Web of Things (Веб речей)

ВСТУП

Книга містить матеріали практичної частини магістерського курсу «Основи Інтернету речей», розробленого в рамках проекту Internet of Things: Emerging Curriculum for Industry and Human Applications / ALIOT, (573818-EPP-1-2016-1-UK-EPPKA2-CBHE-JP)[1], 2016-2019, що фінансується програмою ЄС ERASMUS+. Книга складається з 9 лабораторних робіт для модуля 2 цього курсу «Архітектури та платформи Інтернету/Вебу речей».

Лабораторні роботи включають завдання для вивчення платформ і розроблення систем індустріального Інтернету і Вебу речей з використанням Arduino, PLCNext та інших засобів.

Лабораторні роботи 1-7 підготовлено доцентом кафедри комп'ютерних систем, мереж і кібербезпеки Національного аерокосмічного університету ім. М. Є. Жуковського «Харківський авіаційний інститут» кандидатом технічних наук. доцентом А. П. Плахтєєвим разом з аспіранткою кафедри Ю. В. Здоровець, лабораторну роботу 9 підготував старший викладач цієї кафедри кандидат технічних наук Є.В. Бабешко. Лабораторну роботу 8 розробив професор кафедри систем інформації Національного технічного університету «Харківський політехнічний інститут» кандидат технічних наук, доцент В.А. Ткаченко.

Реагування книги виконано завідувачем кафедри комп'ютерних систем, мереж і кібербезпеки Національного аерокосмічного університету ім. М. Є. Жуковського «Харківський авіаційний інститут» доктором технічних наук, професором В.С. Харченко.

Книга призначена для магістрантів і аспірантів, які вивчають технології IoT, програмну і комп'ютерну інженерію, комп'ютерні науки. Може бути корисною для викладачів університетів і навчальних центрів, розробників систем IoT.

[1] *The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.*

Лабораторна робота 1. Швидке розроблення пристроїв IoT в середовищі віртуального моделювання Proteus

Мета роботи: ознайомлення з віртуальними моделями пристроїв IoT в програмному середовищі Proteus, отримання практичних навиків швидкого налагодження програм для AVR-мікроконтролерів в середовищі Proteus.

Програма роботи

1. Навчитися аналізувати та проектувати пристрої, використовуючи віртуальні мікроконтролери Arduino, засоби налагодження програм.
2. Навчитися використовувати готові віртуальні пристрої (проекти) на основі плат Arduino в середовищі Proteus.
3. Навчитися виконувати конфігурацію, проводити моделювання розробленої схеми, використовуючи програми користувача.
4. Дослідити віртуальне середовище моделювання Proteus**.**
5. Підготувати звіт про виконану роботу.

Теоретичні відомості

Розробником пакету Proteus є фірма Labcenter Electronics Великобританія. Сайт розробника - http://www.labcenter.co.uk/. Відмінність від аналогічних за призначенням пакетів програм, наприклад, Electronics Workbench, Multisim, MicroCap, Tina і т.п. в розвиненій системі симуляції (інтерактивного налагодження в режимі реального часу і покроково) для різних сімейств мікроконтролерів: 8051, PIC (Microchip), AVR (Atmel), і ін. Proteus має великі бібліотеки компонентів, в тому числі і периферійних пристроїв: світлодіодні і РК індикатори, температурні датчики, годинник реального часу - RTC, інтерактивних елементів введення-виведення: кнопок, перемикачів, віртуальних портів і віртуальних вимірювальних приладів, інтерактивних графіків, які не завжди присутні в інших подібних програмах.

Proteus VSM складається з двох самостійних програм: ISIS і ARES. ARES - це трасування друкованих плат. Основною програмою є ISIS, в ній передбачений гарячий зв'язок з ARES для розведення плати.

Спрощено, робота в середовищі моделювання Proteus ISIS складається з наступних пунктів:

1. Вибір електричної принципової схеми і задання їх параметрів функціонування з набору прикладів.
2. Розміщення віртуальних приладів там, де це необхідно, вибір режимів їх роботи.
3. Симуляція чи проведення спеціалізованого аналізу.
4. Виконання налагодження програм мікроконтролерів.

Введення та дослідження навчальних прикладів засобами середовища моделювання Proteus

Для того що б отримати уявлення про те, на що здатний Proteus, як середовище моделювання, слід відкрити деякі файли прикладів проектів. У VSM Studio IDE була реалізована підтримка наборів інструментів Arduino AVR. Це дозволяє розробляти прототипи проектів Arduino безпосередньо всередині програми Proteus, нова функція проекту дуже корисна тут, так як проект можна легко створити для різних Arduino, але про це поговоримо пізніше.

Приклад: вимірювання температури за допомогою датчика DS18B20.

Подивитися на процеси, що відбуваються при вимірюванні температури можна, відкривши файл з набору прикладів: File-Open Sample Project - VSM for AVR - Arduino - Arduino DS18B20 OneWire (рисунку 1.1).

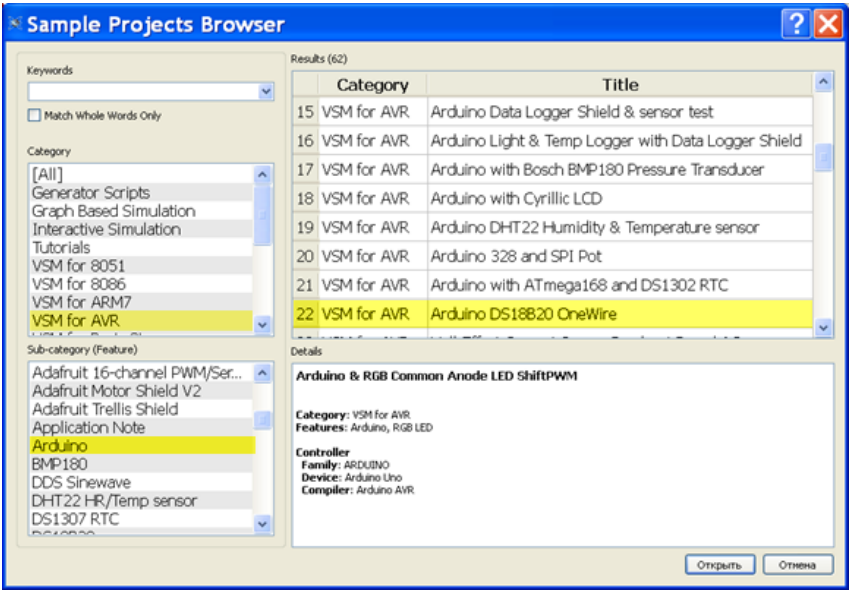


Рисунок 1.1 - Вибір проекту в середовищі моделювання Proteus

На рисунку 1.2. наведена схема моделювання цифрового термометра на мікроконтролері ATmega328 з цифровим датчиком температури DS18B20 фірми Dallas Semiconductor і виведенням інформації на семисегментний індикатор. Мікросхема DS18B20 забезпечує температурні вимірювання за шкалою Цельсія. Мікросхема DS18B20 підключається через 1-дротову шину, яка за визначенням вимагає тільки однієї лінії даних для взаємодії з мікроконтролером. Вона має робочий температурний діапазон від -55 ° C до + 125 ° C і точність ± 0.5 ° C в діапазоні від -10 ° C до + 85 ° C. Модель термометра DS18B20 дозволяє задавати температуру термодатчика.

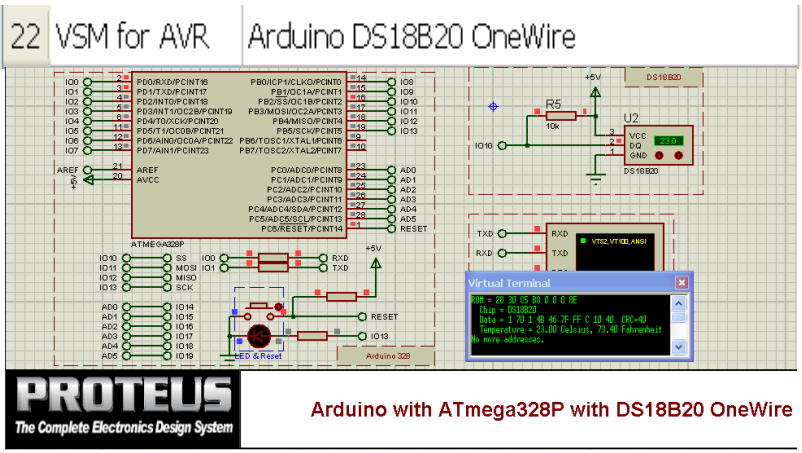


Рисунок 1.2 – Схема моделювання зміни температури з термодатчиком

За допомогою вбудованого в середовище моделювання Proteus осцилографа можна отримати осцилограму інформаційного обміну мікроконтролера з датчиком. Тимчасова діаграма сигналів (протокол 1-Wire) представлена на рисунку. 1.3.

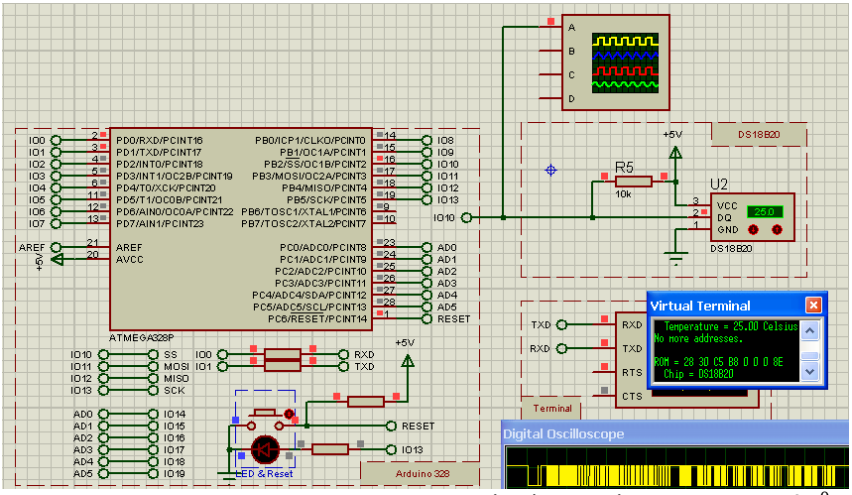


Рисунок 1.3 – Сигнал на однопровідній шині датчика при 25^0C

Рівні напруги можна визначити за кольором міток на всіх виводах компонентів (лог.1 - червоний колір, лог.0 - синій, непідключений вивід - сірий колір).

Налаштування мікроконтролера

Панель редагування властивостей мікроконтролера (рис. 1.4) відкривається подвійним клацанням по ньому лівою кнопкою миші або через праву кнопку і опцію Edit Properties (можна, виділивши його, натиснути на клавіатурі Ctrl + E).

В полі "Program File" потрібно вибрати файл з розширенням "elf" (для налагодження) або "hex", які створюються при компіляції програми в IDE Arduino, WinAVR і ін..

".hex" - файл "прошивки", яку завантажують і в реальний мікроконтролер. При виборі його можна налагоджувати пристрій без можливості перегляду виконуваного коду.

Щоб використовувати файл програми з розширенням ".hex" потрібно клацнути по мікроконтролеру правою кнопкою миші, вибрати пункт «правка властивостей», далі в цьому пункті вибрати «Program file» і ввести шлях до файлу, або просто клацнути по значку «відкрити папку», і вказати потрібний файл. Після цього ніяких додаткових дій робити не потрібно, крім натискання кнопки «ОК».

Clock Frequency - однозначно визначає будь-яку частоту тактування мікроконтролера при симуляції, яка відповідає частоті мікроконтролера в проекті. За умовчанням, встановлена частота синхронізації - 8 МГц. В Arduino Uno, Nano частота синхронізації - 16 МГц. У деяких Arduino Mini - 8 МГц.У середовищі програмування IDE Arduino за замовчуванням вибирається частота 16 МГц, але є можливість встановлення частоти 8 МГц. Для зміни тактової частоти в списку CKSEL Fuses потрібно вибрати Ext. Clock і в Advanced Properties замість (Default) вводимо потрібну частоту в герцах.

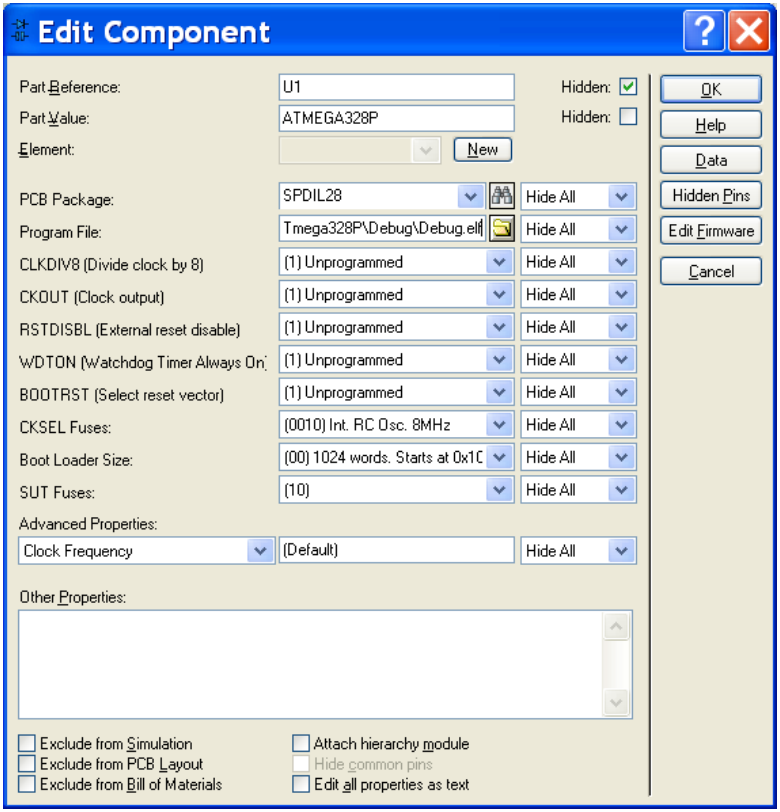


Рисунок 1.4 – Вікно налаштування властивостей мікроконтролера

Після цих установок проект із прикладів Proteus функціонує відповідно до заданої програми. Це дає можливість використовувати приклади проектів в Proteus demo для налагодження власних програм.

Proteus може використовувати компілятори, які встановлені в System - Compilers, зокрема:

- Arduino AVR - Yes - <Path IDE Arduino>;
- WinAVR - Yes - <Path WinAVR>.

Аналіз і редагування вихідного коду моделі мікроконтролера, використання віртуальних інструментів виконується відповідно до опису та додатків:

- Додаток 1. Вихідний текст програми і "hex" – файл.
- Додаток 2. Використання віртуального осцилографа для аналізу послідовностей сигналів.
- Додаток 3. Аналіз елементів моделі мікроконтролера.

Задача 1 .Ознайомитися з прикладами проектів працюючих схем в програмному середовищі Proteus для Arduino

- Arduino 4 Channel Relay;
- Arduino Cyrillic LCD;
- Arduino Motor Shield.

Задача 2. Описати процеси, які відбуваються при моделюванні схем– прикладів. .

Зміст звіту

1. Короткий опис змісту виконання роботи.
2. Опис прикладів симуляції в Proteus:
   - - принцип роботи;
   - - приклади осцилограм;
   - - дані обміну в терміналі;
   - - оцінку використуваних ресурсів (пам'яті програм і даних, виводів).
3. Висновки по роботі

Контрольні питання

1. Який порядок створення проекту в Proteus?
2. Як відкрити приклад для AVR МК?
3. Як визначити розмір програми в пам'яті МК?
4. Що таке .hex файл? Як його завантажити в МК?
5. Як вибрати віртуальний інструмент?
6. Для чого потрібний осцилограф?
7. Які основні налаштування віртуального осцилографа?
8. Які основні налаштування віртуального терміналу?
9. Як визначити часові інтервали параметрів сигналів в Proteus.
10. Як зняти значення з віртуального виходу порта?

Література

1. Симулятор – отладчик PROTEUS VSM. ISIS.- Labcenter Electronics Co. - 26с.- – Proteus_ISIS_russian_manual.pdf. [Електронний ресурс]. – Режим доступу: http://labkit.ru/userfiles/file/education/Proteus/ Proteus_ISIS_russian_manual.pdf.

2. PROTEUS по-русски. Радио - ежегодник, выпуск 24. RYB_2013_24_Proteus.pdf. 2013. [Електронний ресурс]. – Режим доступу: https://www.rlocman.ru/book/book.html?di=148418 [Доступ 25.06.2019г].

3. А.В. Пархоменко, О. М. Гладкова, Я. І. Залюбовський, А.В. Пархоменко, Інженерія вбудованих систем: Навчальний посібник. Запоріжжя: Дике Поле, 2017.

Додаток 1. Вихідний текст програми і “hex”- файл

Приклади Proteus, як правило, містять вихідні тексти програм для мікроконтролерів в спеціальній закладці Source code, з якої вихідний код можна скопіювати і використовувати для вивчення і побудови власних програм. При конфігурації Proteus підключаються різні середовища розробки програм з відповідними компіляторами.

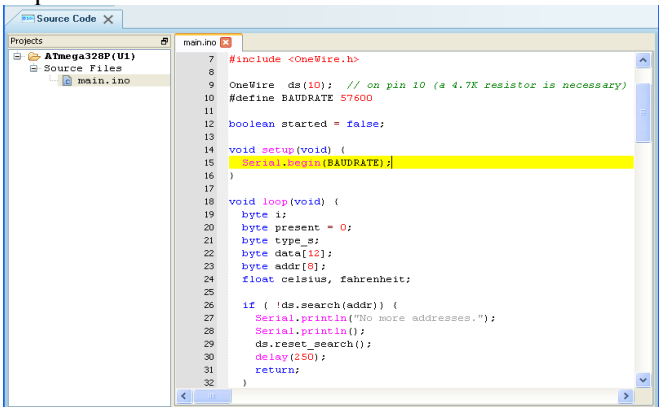


Рисунок 1.5. - Вигляд вікна з вихідним текстом програми

Результатом компіляції тексту програми є “hex” - файл. Наведений нижче фрагмент файлу містить коди команд, що розміщуються за адресами $ 0000 ... $ 00AF Flash - пам'яті програм мікроконтролера (порівняти з вмістом вікна налаштувань вікна пам'яті програм).

:10000000**0C94D8000C9400010C9400010C94000**
195
:10001000**0C9400010C9400010C9400010C94000**
15C
:10002000**0C9400010C9400010C9400010C94000**
14C
:10003000**0C9400010C9400010C9400010C94000**
13C
:10004000**0C94CA040C9400010C9498040C94720**
44F
:10005000**0C9400010C9400010C9400010C94000**
11C
:10006000**0C9400010C940001005EBCE2613FDD**
8352
:10007000**C29C7E20A3FD1F419DC3217FFCA2**
401E88
:100080005**F01E3BD3E6082DC237D9FC1421CF**
EA078
:10009000**E1BF5D0380DE3C62BEE0025CDF816**
33D68
:1000A0007**C22C09E1D43A1FF4618FAA427799**
BC558

Після налагодження програми “hex” - файл програми за допомогою програматора завантажується в пам'ять програм мікроконтролера пристрою

Додаток 2. Використання віртуального осцилографа для аналізу послідовності сигналів

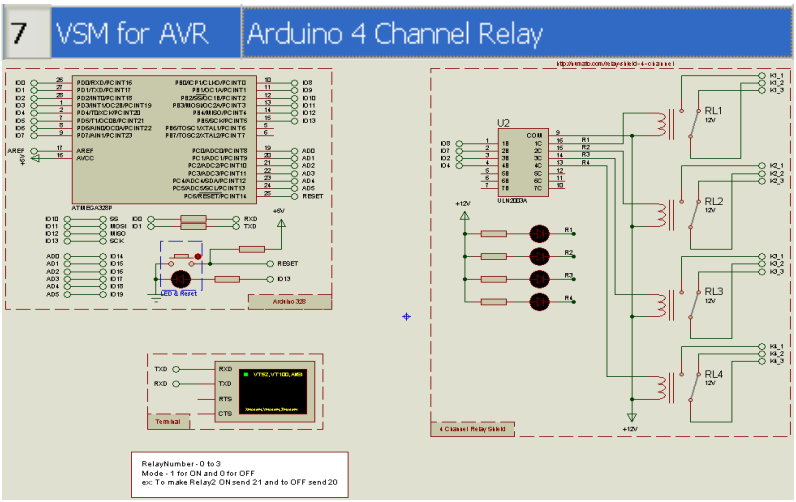


Рисунок 1.6 – Arduino 4 Channel Relay4.

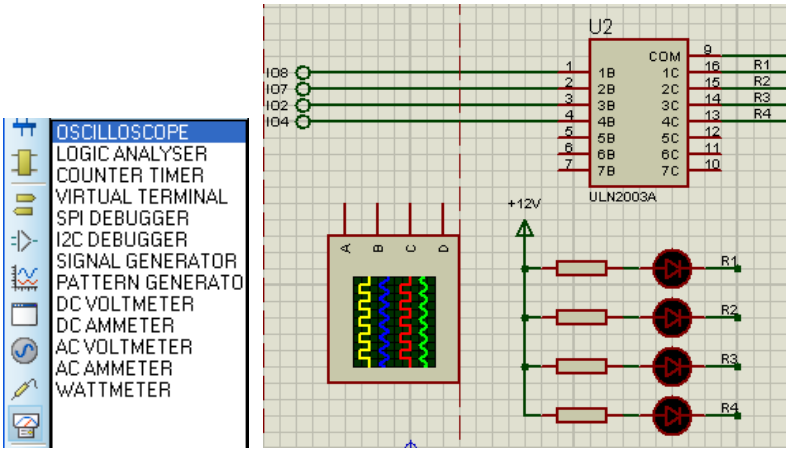


Рисунок 1.7 – Винесення терміналів сигналів IO8,7,2,4 і додавання 4- канального віртуального осцилографа

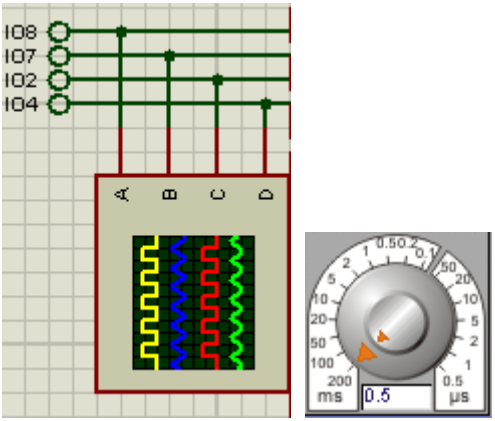


Рисунок 1.8 – Встановлення терміналів і підключення до каналів A,B,C,D із часовою шкалою 0.5 мс

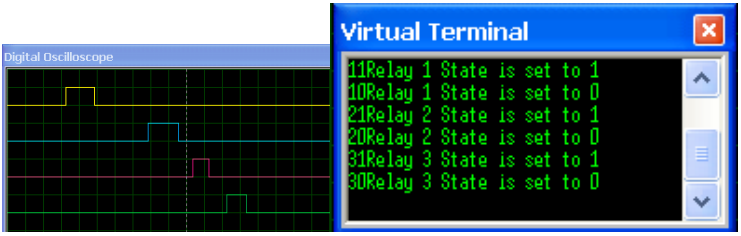


Рисунок 1.9 – Спостереження реакції МК (IO8,7,2,4) на команди, що поступають по послідовному каналу

Додаток 3. Аналіз елементів моделі МК

При налагоджені програм Proteus надає доступ до внутрішніх елементів моделі мікроконтролера при виборі Debug - AVR (рисунок 1.10):

- CPU Registers (регістри загального призначення, рисунок 1.11);
- Data Memory (оперативна пам'ять даних - ОЗУ (SRAM), рисунок 1.12) ;
- EPROM Memory (енергонезалежна пам'ять даних, рисунок 1.13);
- Program memory (Flash пам'яті програм, рисунок 1.14);
- IO Registers (регістри введення-виведення, рисунок 1.15).

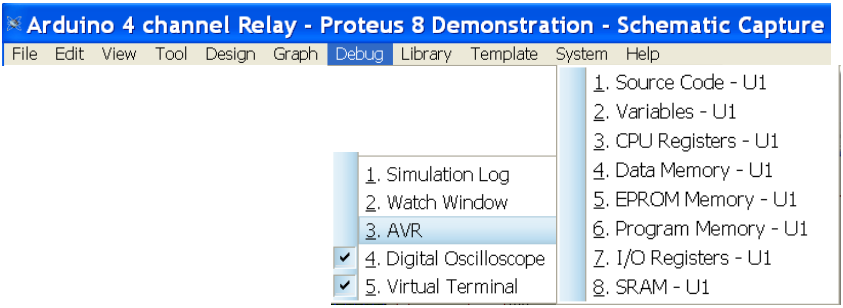


Рисунок 1.10 – Вибір елементів моделі МК

Робочі регістри (регістри загального призначення) R0…31 утворюють область надоперативної пам'яті. Регістри беруть безпосередню участь при виконанні машинних команд. Використовуються при програмуванні на мові низького рівня - Асемблер.

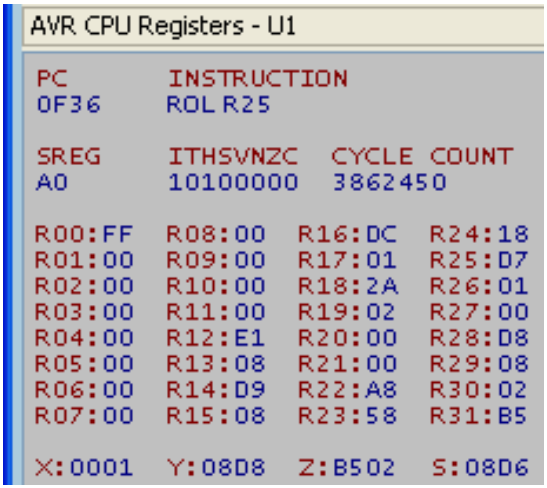


Рисунок 1.11 – Стан 32 робочих регістрів ядра МК (CPU Registers)

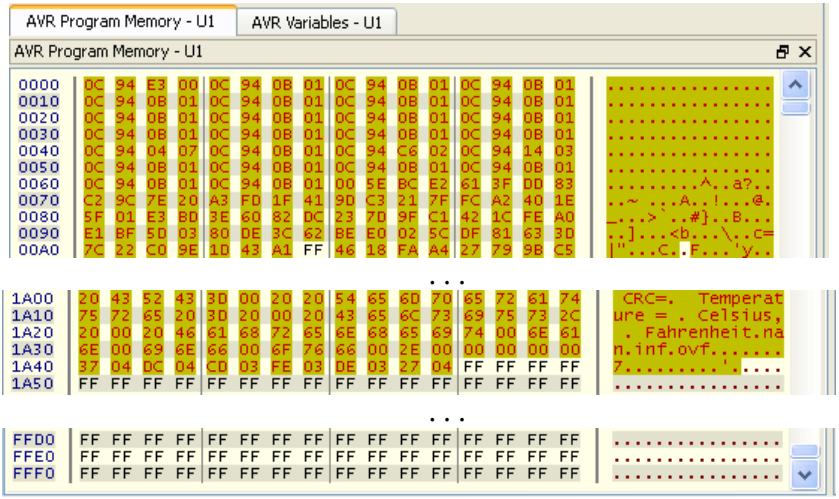


Рисунок 1.12 - Вміст 64К=65536 байт (32768 слів) Flash пам’яті програм.

Коди команд програми займають 6732 байт в діапазоні адрес $0000 … $1A4B (0 .. 6731).

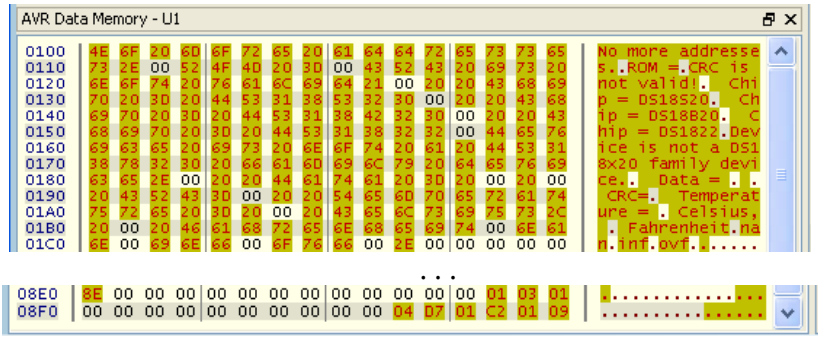


Рисунок 1.13 – Масив 2048 байт (2 Кб) комірок ОЗП (SRAM)

Рисунок 1.14 - Масив 1024 байт (1 Кб) енергозалежної пам’яті даних (EEPROM)

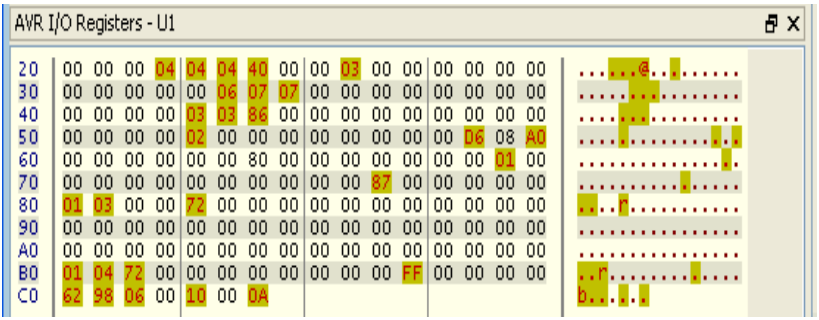


Рисунок 1.15 – Масив 176 регістрів введення – виведення (I/O)

Регістри спеціального призначення (введення-виведення) пов'язані з портами мікроконтролера і зовнішніми виведенням, а також внутрішніми периферійними пристроями.

Лабораторна робота 2. Розроблення цифрової системи керування на основі платформи Arduino

Мета роботи: побудова алгоритму і програми керування виконавчим пристроєм- двохкоординатним електроприводом з числовим програмним керуванням на основі апаратних, програмних та інструментальних засобів МК платформи Arduino.

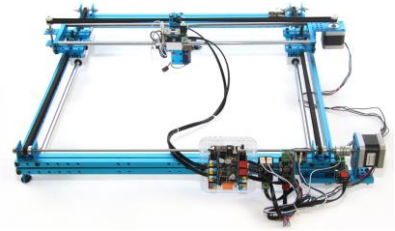
Програма роботи

1. Побудова таблиці керуванням переміщеннями.
2. Побудова тимчасових діаграм керуванням переміщенням.
3. Побудова схеми алгоритму керуванням переміщенням.
4. Призначення ліній для вихідних сигналів керування
5. Розробка версії 1 програми (додаток 1) і налагодження з використанням Proteus (модель - 4 channel Relays с лініями керування 8,7,2,4) і віртуальних інструментів. Отримати осцилограми сигналів для Tw=100.500 mc). Виконати оцінку необхідних ресурсів.
6. Розробка версії 2 оптимізованої програми (додаток 2) і налагодження аналогічно п.5.
7. Завантаження програми (Tw=1000 mc) з п.5 або п.6 в пам'ять керуючого мікроконтролера лабораторного мініплоттера і отримання результату малювання контура.
8. Додатково. Розробка і налагодження версії 3 програми (додаток 3). Оцінка необхідних ресурсів.
9. Висновки по роботі. Оформлення звіту в електронному вигляді з приведенням знімків екрану програм, фотографій і відеофрагментів проведених експериментів.

Теоретичний матеріал

Керування двохкоординатними електроприводами

У сучасному технологічному обладнані широкого поширення отримали координатні столи і маніпулятори, обладнані різними видами електроприводів (рисунок 2.1).

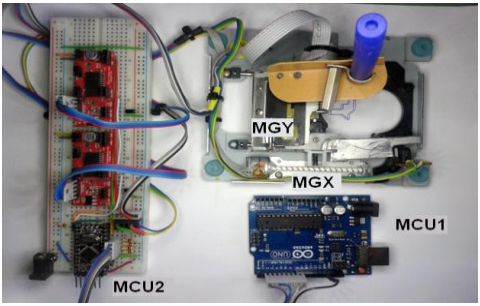


а)

б)

в)

г)



д)

Рисунок 2.1 — Застосування двохкоординатних електроприводів: а) планшетний плоттер XY-Plotter Robot Kit v2.0; б) друкуючий (ріжучий) плоттер; в) маніпулятор; г) роботизована поворотна камера; д) лабораторний мініплоттер

Для керування електроприводами X, Y координатного плоттера (рисунок 2.2г) використовуються силові модулі - драйвери MGX, MGY, що забезпечують необхідні значення струмів і напруги для обертання електродвигунів із заданою швидкістю і в необхідному напрямі. Логічні сигнали керування

даними драйверами формуються керуючим мікроконтролером MCU2.

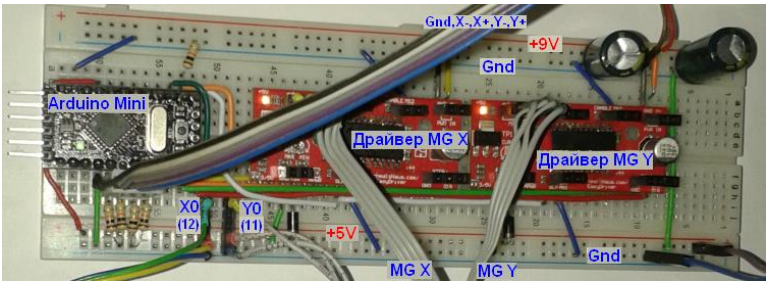


Рисунок 2.2 — Система керування двохкоординатним електроприводом

Мікроконтролер MCU1 керує зміною напряму переміщення вузла, що пише, по необхідному контуру.

Приклад побудови МК блоку керування двохкоординатним столом

Переміщення двохкоординатного столу здійснюється по координатній сітці з фіксованим кроком. Фрагмент координатної сітки показаний на рисунку 2.3.

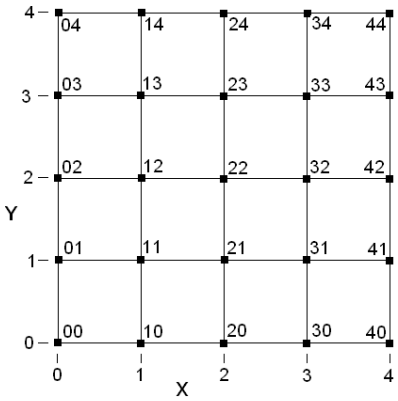


Рисунок 2.3 — Фрагмент координатної сітки

Для виконання технологічних операцій координатний стіл переміщується по деякій траєкторії (рисунок 2.4а), що складається з послідовності кроків з точки 0 через точки 1,2,3,4,5 за напрямками 2,3,1,3,1 з набору на (рисунку 2.4б) ..

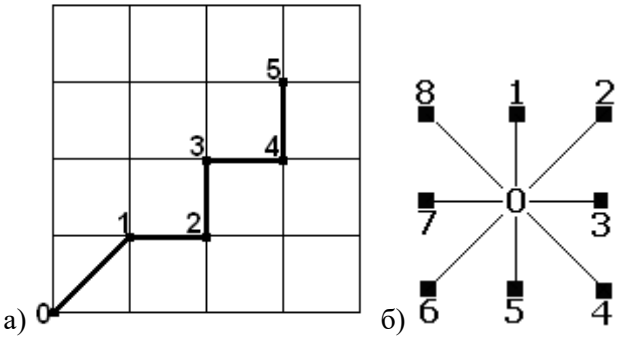


Рисунок 2.4 — Траєкторія руху (а) і вектори напрямів (б).

Сигнали керування (X, X-) блоку керування забезпечують включення/виключення і вибір напряму обертання двигуна електроприводу X (рисунок 2.4а). Пов'язана з двигуном гвинтова передача забезпечує перетворення обертального руху в поступальний по осі. Аналогічно, сигнали (Y, Y-) забезпечують керування поступальним рухом по осі Y (рисунок 2.5).

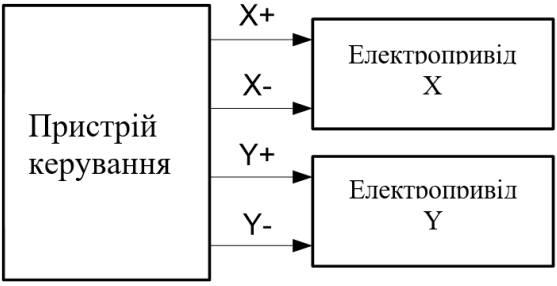


Рисунок 2.5. — Структурна схема системи керування рухом двохкоординатного столу

У таблиці. 2.1 наведені комбінації керуючих сигналів, для усіх напрямів переміщення Dir = 0,8.

Таблиця 2.1 - Сигнали керуванням переміщенням

Dir	X+	X-	Y+	Y-
0	0	0	0	0
1	0	0	1	0
2	1	0	1	0
3	1	0	0	0
4	1	0	0	1
5	0	0	0	1
6	0	1	0	1
7	0	1	0	0
8	0	1	1	0

Тепер можна формалізувати опис роботи пристрою керування координатного столу по вузлах (XY) координатної сітки з урахуванням векторів переміщень (Dir) на кожному кроці. У початковому стані 0 приводи зупинені (комбінація 0000). Для кожного кроку 0..6 в таблиці 2.2 заносимо комбінації сигналів (X, X-, Y, Y-) і тривалість кроку (T). Після завершення переміщень в стані 6 приводи зупиняються (комбінація 0000).

Таблиця 2.2 - Сигнали керування переміщеннями

№	Dir	X+	X-	Y+	Y-	T, сек
0	0	0	0	0	0	1
1	2	1	0	1	0	1
2	3	1	0	0	0	1
3	1	0	0	1	0	1
4	3	1	0	0	0	1
5	1	0	0	1	0	1
6	0	0	0	0	0	1

У таблиці 2.2 виділені значення, які змінюються в поточному кроці переміщення. Інші зберігають попередні значення. З таблиці

отримуємо тимчасові діаграми сигналів керування (рисунок 2.6) для даного прикладу.

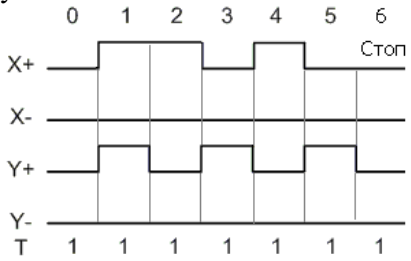


Рисунок 2.6. - Тимчасові діаграми для сигналів керування переміщенням.

Пристрій керування є програмованим і його функціонування окрім тимчасових діаграм описується схемою алгоритму (рисунок 2.7).

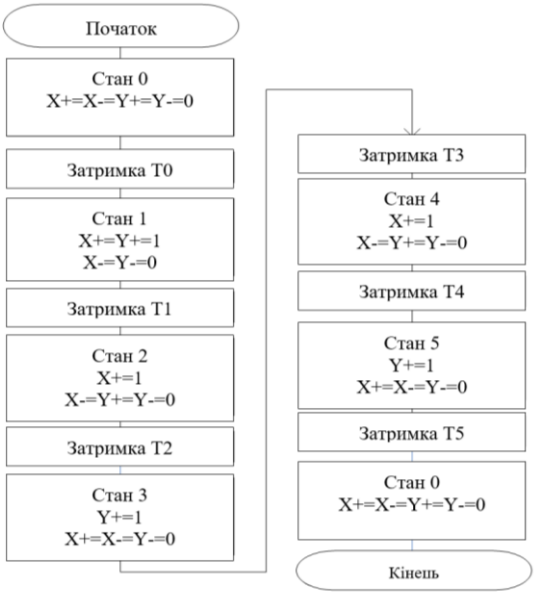


Рисунок 2.7 - Алгоритм керування переміщеннями

Число операцій в алгоритмі можна зменшити, якщо виключити дії, що повторюються. Якщо на черговому кроці стан лінії керування не змінюється, то операція підтвердження стану не потрібна. Таким чином можна отримати оптимізовану схему алгоритму (рисунок 2.8).

Пристрій керування може бути побудований на основі оригінальної МК платформи Arduino UNO (рисунок 2.9а) або одного з клонів Arduino UNO (рисунок 2.9б) на основі МК ATmega328 в різних корпусах. Умовно графічне позначення Arduino UNO наведено на рисунку 2.9в.

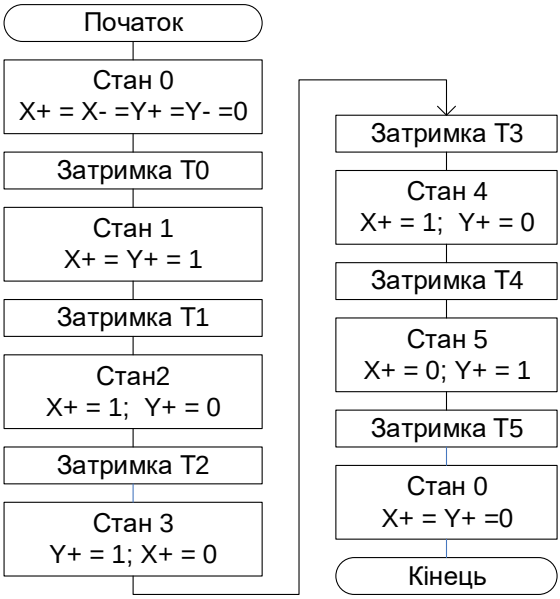


Рисунок 2.8 - Алгоритм керування переміщеннями

Пристрій керування може бути побудований на основі оригінальної МК платформи Arduino UNO (Рисунок 8а) або одного з клонів Arduino UNO (Рисунок 8б) на основі МК ATmega328 в різних корпусах. Умовно графічне позначення Arduino UNO наведено на рисунку 2.9в.

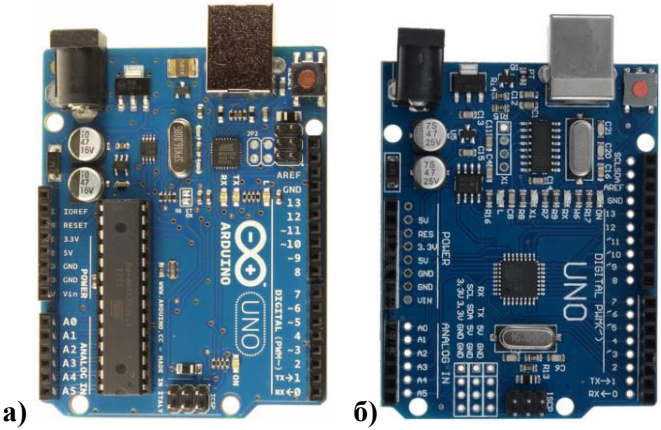


Рисунок 2.9. — Вид Arduino UNO (а, б, в) та умовно-графічне позначення

Структурна схема даного МК пристрою приведена на рисунку 2.10 і містить: 1 - елементи живлення (від USB або зовнішнього джерела); 2 - перетворювач USB - UART для самопрограмування через BOOT- завантажувач і інформаційного обміну МК; 3,4 - роз'єми для підключення зовнішніх пристроїв і сигналів; 5 - МК.

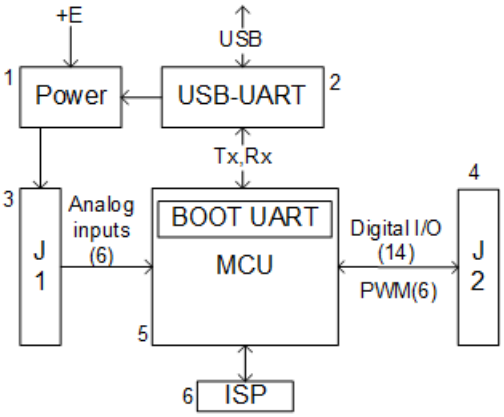


Рисунок 2.10. - Структурна схема МК пристрою Arduino

Детальну інформацію про пристрій можна знайти в літературних джерелах [1] і на численних ресурсах Internet.

Оскільки у розпорядженні користувача є 14 ліній цифрового введення-виведення, які позначаються D0...13 або просто 0...13, необхідно вибрати лінії для вихідних сигналів керування. Можуть бути призначені довільні лінії - 8,7,2,4 (як лінії керування Shield 4-канального реле) або 5,4,3,2 (рисунок 2.11).

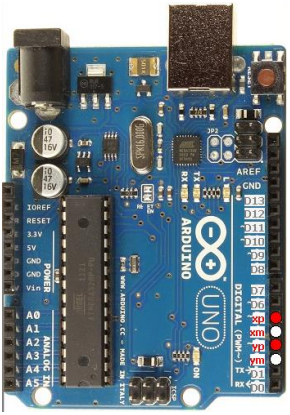


Рисунок 2.11. — Призначення ліній для вихідних сигналів керування Arduino UNO

Середовище програмування Arduino

Інтегроване середовище програмування (IDE Arduino) встановлюється з [1], має дуже простий вигляд і містить - вікно редактора коду, елементи керування і вікно повідомлень (рисунок 2.12).

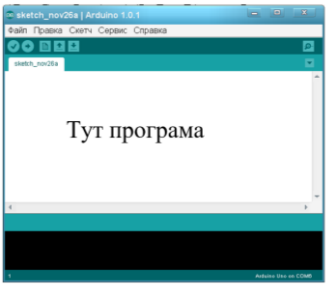


Рисунок 2.12 - Робоче вікно середовища програмування.

За допомогою елементів керування робочого вікна відбувається запуск основних операцій в середовищі програмування Arduino.

- порівняння вмісту Flash- пам'яті з результатом компіляції програми.

- запис програми в програмну пам'ять (Flash) МК.

- початок рботи з новою програмою.

- продовження роботи з існуючою програмою.

- збереження поточної програми.

При створенні програми визначаються змінні і оператори середовища програмування Arduino [1,2,5,6], необхідні для реалізації алгоритму функціонування МК :

```
int xp = 5;  // X+
    int xm = 4; // X-
    int yp = 3;  // Y+
    int ym = 2; // Y-
```

Лінії, що використовуються для керування налаштовуються як вихідні:

```
pinMode(xp, OUTPUT);
    pinMode(xm, OUTPUT);
    pinMode(yp, OUTPUT);
    pinMode(ym, OUTPUT);
```

Установка різних значень вихідних сигналів виконується за допомогою операцій із таблиці. 2.3.

Таблиця 2.3 - Базові оператори програми

Операція	Оператор IDE Arduino
X+ = 0	digitalWrite(xp, LOW)
X- = 0	digitalWrite(xm, LOW)
Y+ = 0	digitalWrite(yp, LOW)
Y- = 0	digitalWrite(ym, LOW)
X+ =1	digitalWrite(xp, HIGH)
X- = 1	digitalWrite(xm, HIGH)
Y+ = 1	digitalWrite(yp, HIGH)
Y- = 1	digitalWrite(ym, HIGH)

З наведених раніше описів і операцій відповідно до алгоритму роботи (рисунок 2.7) створюється текст програми (додаток 1). В результаті компіляції програми отримуємо повідомлення про розмір коду для завантаження в пам'ять МК (рисунок 2.13).

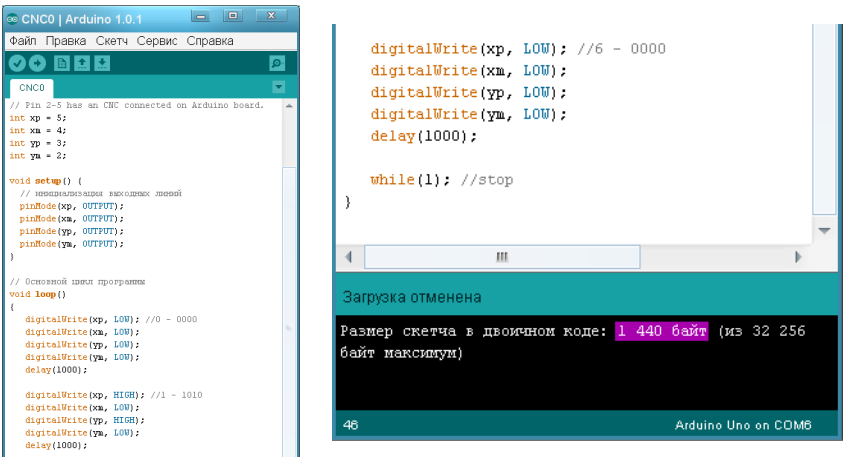


Рисунок 2.13 - Вид програми і результат компіляції

У разі помилок, виводяться повідомлення про її місце і характер. Помилки виправляються і дії повторюються.

Будь-яка програма в процесі створення проходить етап тестування з метою виявлення можливих логічних помилок при написанні програми, або допущених в початкових даних. Ці помилки не виявляються компілятором, але проявляються в програмних моделях МК - симуляторах.

Налагодження програми в симуляторі Arduino

Симулятор дозволяє виконати покрокове налагодження програми [5]. У робочому вікні симулятора відображається текст програми, значення змінних і сигналів МК пристрою (рисунок 2.13).

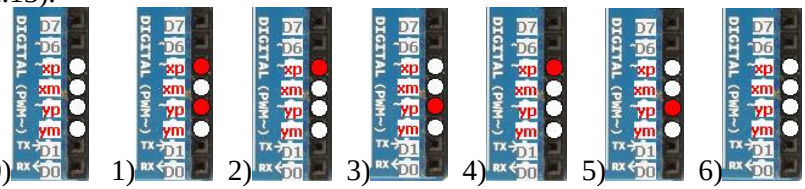


Рисунок 2.13 - Результат покрокового налагодження програми з використанням симулятора Arduino

Наведена програма для алгоритму на рисунку 2.7 надмірна, оскільки включає ряд операцій, що не змінюють стану керуючих сигналів і займає 1440 байт пам'яті програм. Проста оптимізація програми за рахунок виключення надмірних операторів (рисунок 2.8) дає економію 120 байт пам'яті програм (рисунок 2.15).

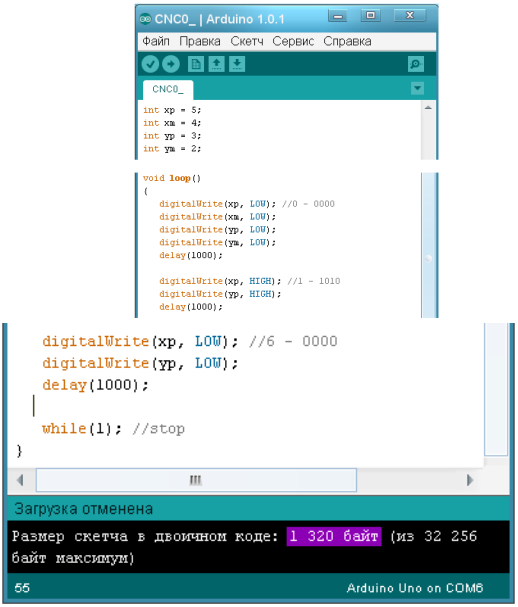


Рисунок 2.15 - Оптимізована програма

За допомогою симулятора можна переконатися, що функціонування пристрою залишилося без змін.

Модифікація програми із спрощеним завданням початкових даних траєкторії руху

Вектори переміщення 0...8 (рисунок 2.4б) можна представити у вигляді 36-елементного неупакованого масиву, де кожному вектору відводиться 4 елементи:

dir[36]={0,0,0,0, 0,0,1,0, 1,0,1,0, 1,0,0,0, 1,0,0,1, 0,0,0,1, 0,1,0,1, 0,1,0,0, 0,1,1,0},

Послідовність номерів векторів переміщень з таблиці 2.2 – масивом:

cmd[7]={0,2,3,1,3,1,0},

тривалість в мс переміщення на кожному кроці –масивом:

time[7]={1000,1000,1000,1000,1000,1000,1000};

Це дозволяє реалізувати N =7 – крокове (включаючи початковий і кінцевий стани 0000) переміщення по заданій траєкторії за наявності функції переміщення на поточному кроці - void move (byte num), в якій встановлюються керуючі сигнали з масиву dir[36], згідно із заданим вектором num. Крок $0 \leq i < N$ реалізується таким чином:

```
move (cmd[i]);
delay (time[i]);
```

Для зміни алгоритму функціонування необхідно задати значення N і елементи масивів cmd[], time[]. У додатку 3 представлений скетч, де реалізовано керування для початкового прикладу.

Налагодження програми в середовищі Proteus

Середовище Proteus [6] дає можливість побудувати схему МК пристрою (рисунок 2.16), зв'язати МК з hex- файлом вмісту пам'яті програм і виконати моделювання роботи пристрою за заданою програмою.

Як отримати .hex з Arduino?

Запускаємо Arduino IDE, відкриваємо скетч, натискаємо Verify. Відкриваємо провідник, пишемо там %temp%\ і натискаємо Enter. Знаходимо там папки з іменами buildXXXXXXXXXXXXXXX.tmp. Якщо програма- CNC1.ino, то в папці слід знайти файл CNC1.cpp.hex - це і є результат компіляції CNC1.ino.

CNC1.cpp.hex можна записати в пам'ять МК плати Arduino, або використовувати при моделюванні в Proteus.

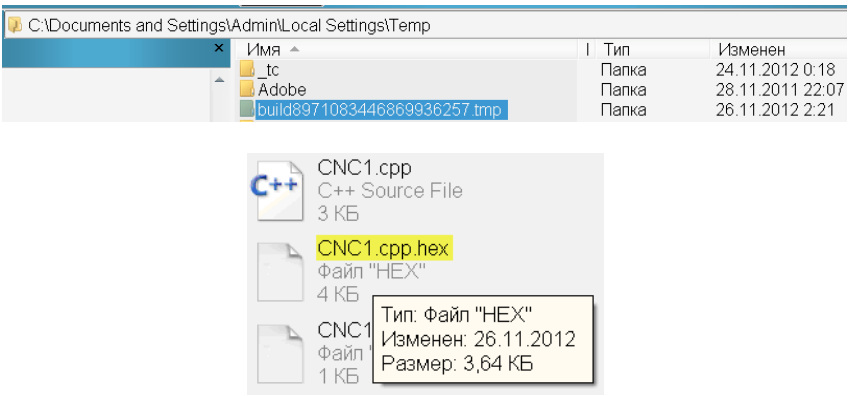


Рисунок 2.16 — Пошук “hex” –файла

У Proteus використовуються віртуальні елементи керування і виконавчі пристрої (рисунок 2.17), а тимчасові діаграми вихідних сигналів можуть бути зареєстровані за допомогою 4-канального осцилографа або логічного аналізатора (рисунок 2.17). Використання демонстраційної версії Proteus вносить обмеження:

- для моделювання можна використати тільки готові приклади з Sample;
- схема може бути змінена, але без можливості збереження;
- до ліній схеми можна підключати осцилограф, пробники і проводити моделювання, але зберегти зміни не можна.

Для моделювання пристрою керування зручно використати приклад керування 4-канальним реле з використанням цифрових виводів 8,7,2,4 (рисунок 2.17). Світлодіоди відображають стани виходів.

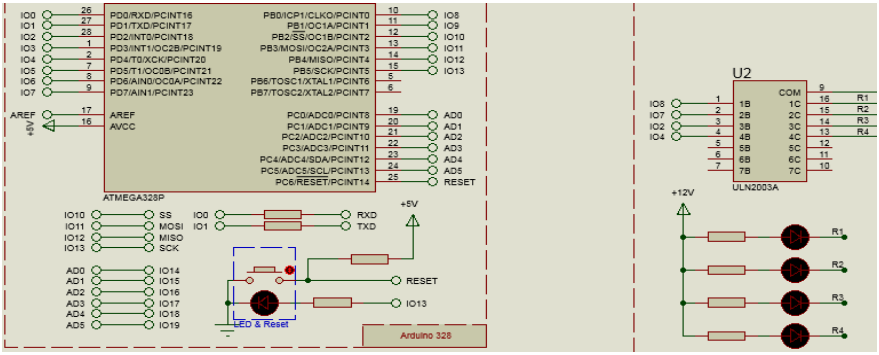


Рисунок 2.17 —. Моделювання пристрою керування електроприводами в Proteus

Для спостереження тимчасових діаграм зміни сигналів керування підключається 4-канальний осцилограф (рисунок 2.18).

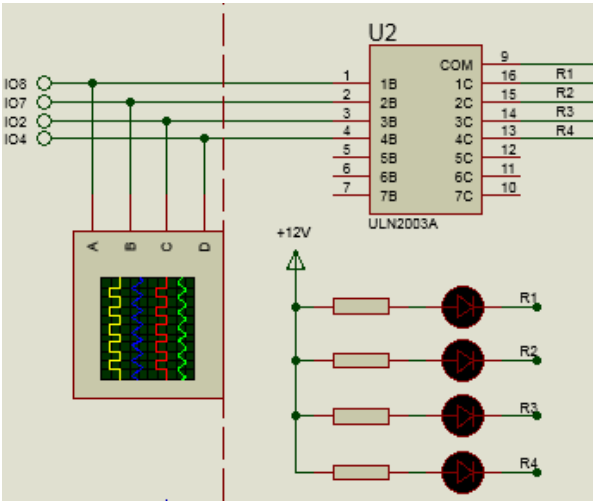
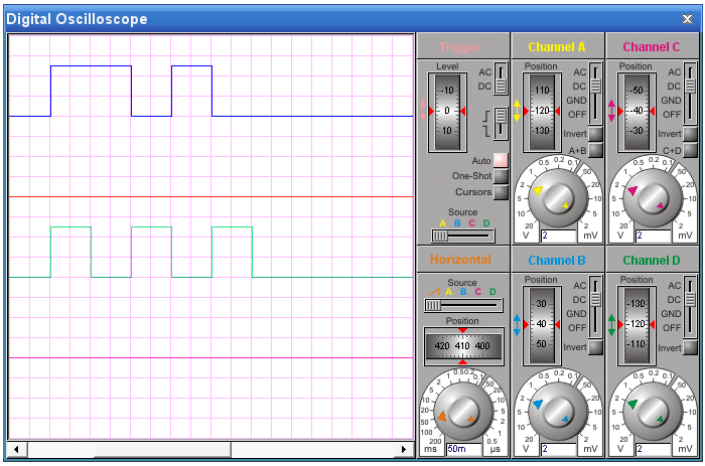


Рисунок 2.18 —. Використання віртуального осцилографа (A – X+, B – X-, C – Y+, D – Y-)

Якщо осцилограми сигналів керування за виглядом співпадають з початковими тимчасовими діаграмами (рисунок 2.19), то це служить підтвердженням правильного формування

керуючих сигналів МК пристрою, працюючого за розробленою програмою в середовищі Arduino.

а)



б)

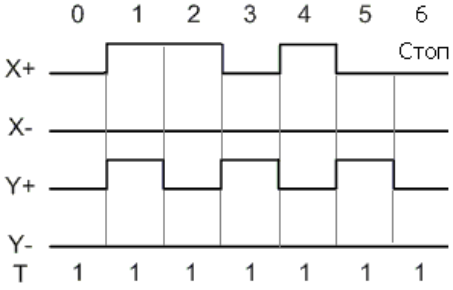


Рисунок 2.19 —. Осцилограми сигналів керування за результатами моделювання (а) і вихідні діаграми (б).

Завантаження програми в пам'ять МК

Плата Arduino, яка виконуватиме керування мініплоттером кабелем USB підключається до вільного порту USB комп'ютера, визначається виділений номер віртуального порту і вказується в налаштуваннях IDE, вибирається тип плати, використовуваний МК і запускається завантаження. Ці дії детально описані в [3].

Після завантаження програми, плата підключається до ліній керування мініплоттером (рисунок 2.1, 2.2). Після скидання мініплоттер під керуванням плати Arduino малює контур який повинен співпадати із заданим (рисунок 2.20).

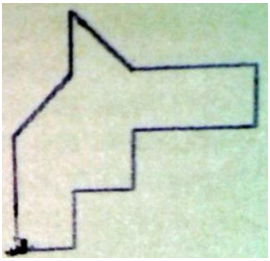
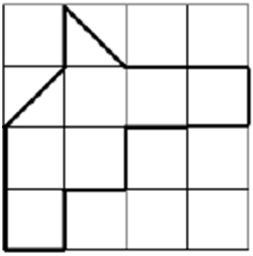
а)  б) 

Рисунок 2.20 — Приклад результату роботи мініплоттера (а) під керуванням розробленої програми для завдання (б).

Зміст звіту

1. Схеми апаратних засобів для вирішення завдань.
2. Тексти програм для завдань.
3. Результати моделювання.
4. Результати виконання експериментів.
5. Висновки по роботі.

Література

1. Среда разработки Arduino. [Електронний ресурс]. – Режим доступу: http://www.arduino.cc/en/Main/Software

2. Программирование Ардуино. [Електронний ресурс]. – Режим доступу: http://arduino.ru/Reference

3. Соммер У. Программирование микроконтроллерных плат Arduino/Freeduino.- СПб.: БХВ – Петербург, 2012.- 256с.

4. Arduino блокнот программиста_Brian W. Evans.- 2007.- 40с.

5. Simulator for Arduino. [Електронний ресурс]. – Режим доступу: http://virtronics.com.au/Data/SetupFree.zip.

6. PROTEUS VSM. Среда виртуального моделирования. [Електронний ресурс]. – Режим доступу: http://proteus123.narod.ru/PROTEUS-d.pdf

Додаток 1. Версія 1 програми

```
/*  Управління переміщенням за допомогою сигналів xp (X+), xm (X-), yp (Y+), ym (Y-)
    по послідовності N точок. Тривалість кожного кроку- 1000 мс. */

// Pin 2-5 has an CNC connected on Arduino board.
int xp = 5;
int xm = 4;
int yp = 3;
int ym = 2;
int Tw = 1000;
void setup() {
  // ініціалізація вихідних ліній
  pinMode(xp, OUTPUT);
  pinMode(xm, OUTPUT);
  pinMode(yp, OUTPUT);
  pinMode(ym, OUTPUT);
}

void loop()        // Основний цикл програми v1
{
   digitalWrite(xp, LOW); //0 - 0000
   digitalWrite(xm, LOW);
   digitalWrite(yp, LOW);
   digitalWrite(ym, LOW);
   delay(Tw);

   digitalWrite(xp, HIGH); //1 - 1010
   digitalWrite(xm, LOW);
   digitalWrite(yp, HIGH);
   digitalWrite(ym, LOW);
   delay(Tw);

   digitalWrite(xp, HIGH); //2 - 1000
   digitalWrite(xm, LOW);
   digitalWrite(yp, LOW);
```

```
  digitalWrite(ym, LOW);
  delay(Tw);

  digitalWrite(xp, LOW); //3 - 0010
  digitalWrite(xm, LOW);
  digitalWrite(yp, HIGH);
  digitalWrite(ym, LOW);
  delay(Tw);

  digitalWrite(xp, HIGH); //4 - 1000
  digitalWrite(xm, LOW);
  digitalWrite(yp, LOW);
  digitalWrite(ym, LOW);
  delay(Tw);

  digitalWrite(xp, LOW); //5 - 0010
  digitalWrite(xm, LOW);
  digitalWrite(yp, HIGH);
  digitalWrite(ym, LOW);
  delay(Tw);

  digitalWrite(xp, LOW); //6 - 0000
  digitalWrite(xm, LOW);
  digitalWrite(yp, LOW);
  digitalWrite(ym, LOW);
  delay(Tw);

  while(1)}; //stop
```

Додаток 2. Версія 2 програми (основний цикл)

```
void loop()           // Основний цикл програми
{
  digitalWrite(xp, LOW); //0 - 0000
  digitalWrite(xm, LOW);
  digitalWrite(yp, LOW);
  digitalWrite(ym, LOW);
  delay(Tw);

  digitalWrite(xp, HIGH); //1 - 1010
```

```
digitalWrite(yp, HIGH);
delay(Tw);

digitalWrite(xp, HIGH); //2 - 1000
digitalWrite(yp, LOW);
delay(Tw);

digitalWrite(xp, LOW); //3 - 0010
digitalWrite(yp, HIGH);
delay(Tw);

digitalWrite(xp, HIGH); //4 - 1000
digitalWrite(yp, LOW);
delay(Tw);

digitalWrite(xp, LOW); //5 - 0010
digitalWrite(yp, HIGH);
delay(Tw);

digitalWrite(xp, LOW); //6 - 0000
digitalWrite(yp, LOW);
delay(Tw);

while(1) } ; //stop
```

Додаток 3. Версія 3 програми (Low code)

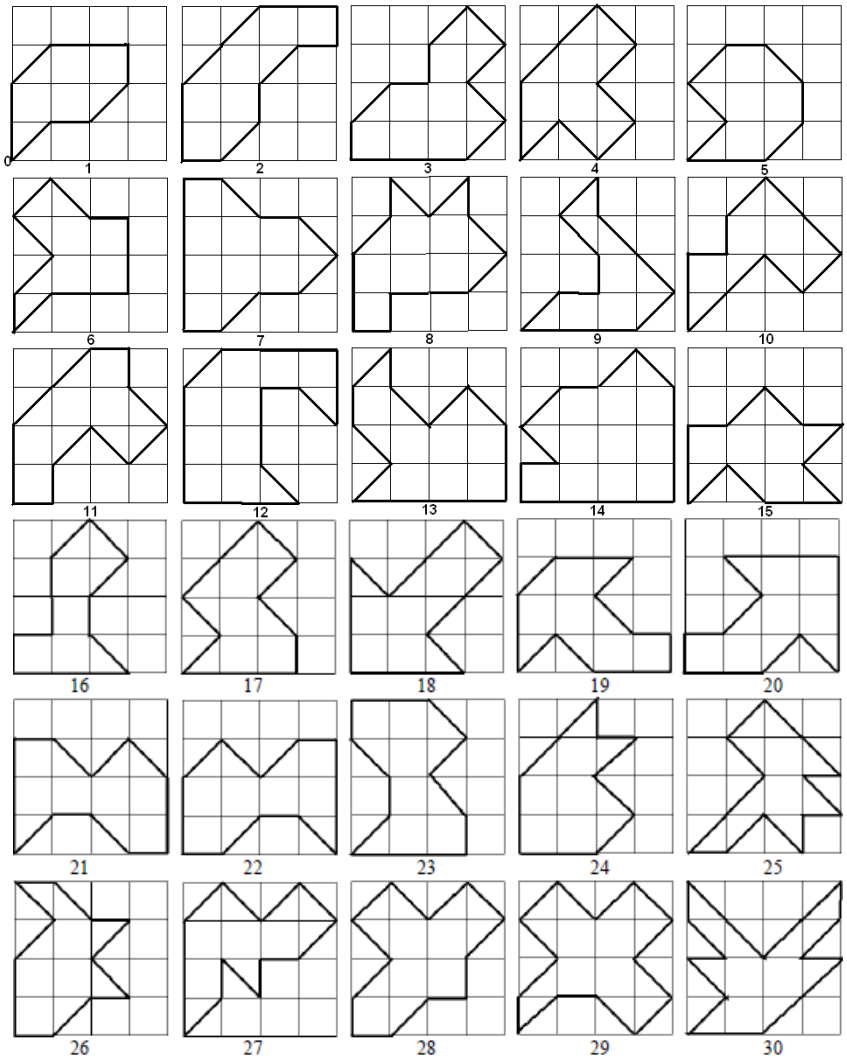
/* Керування переміщенням за допомогою сигналів xp (X), xm (X-), yp (Y), ym (Y-) по послідовності N точок. Напрям переміщення на кожному кроці задається значеннями масиву cmd[], а керування переміщенням у будь-якому напрямі - відповідним набором сигналів з масиву dir[]. Напрями мають номери:

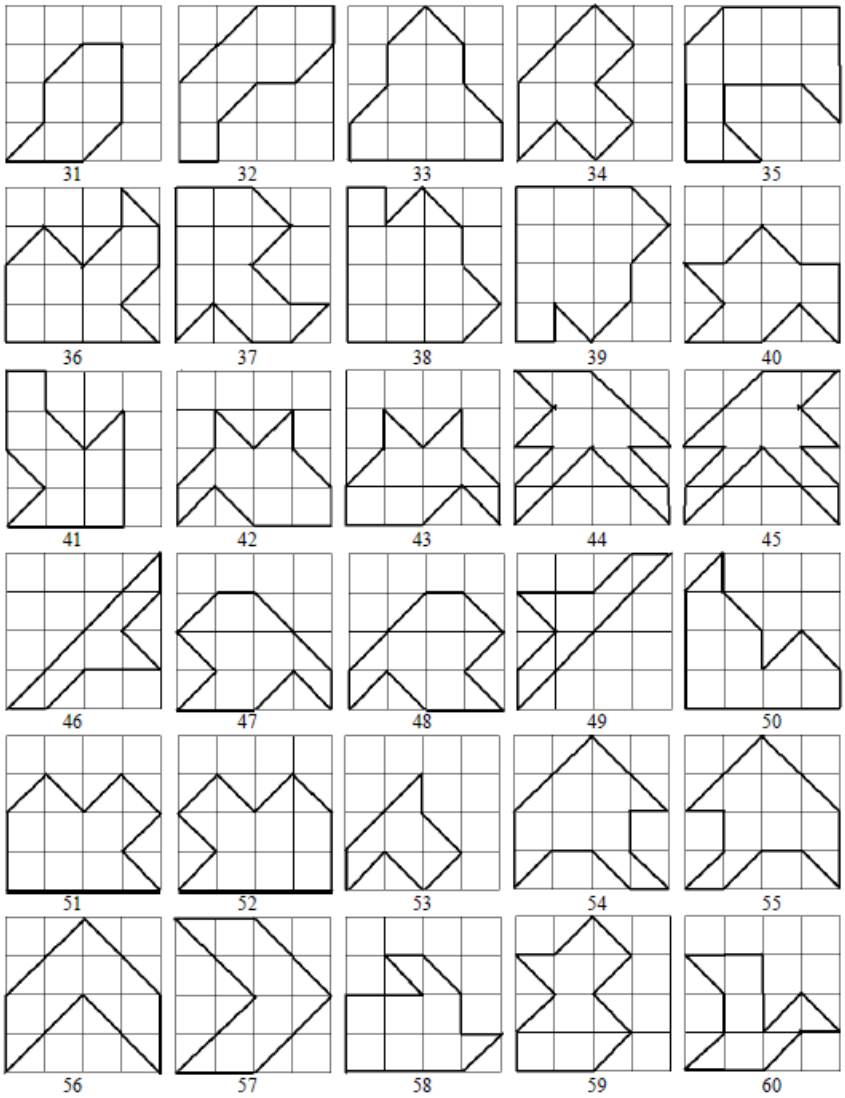
8 1 2
7 0 3
6 5 4

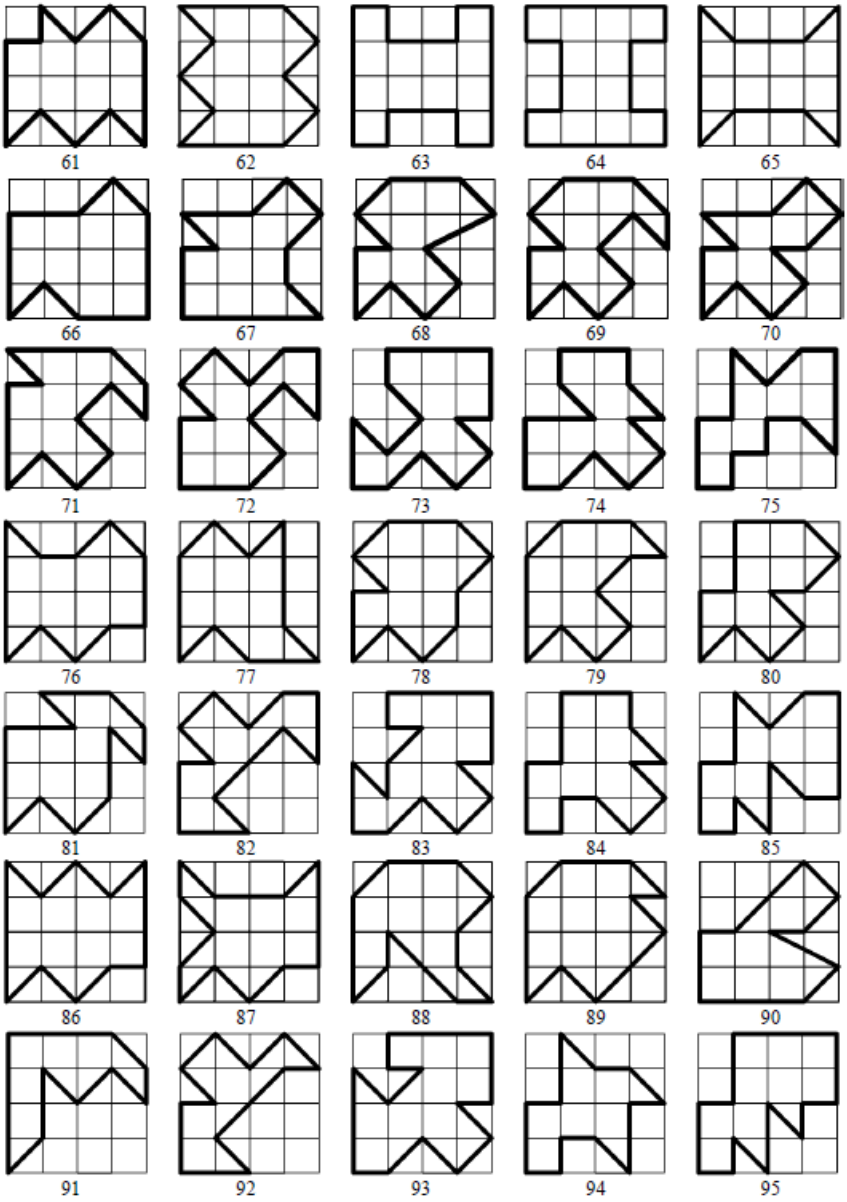
Початкове положення кожного кроку - 0 (Стоп). Тривалість кожного кроку (мс) задається в масиві time[]:

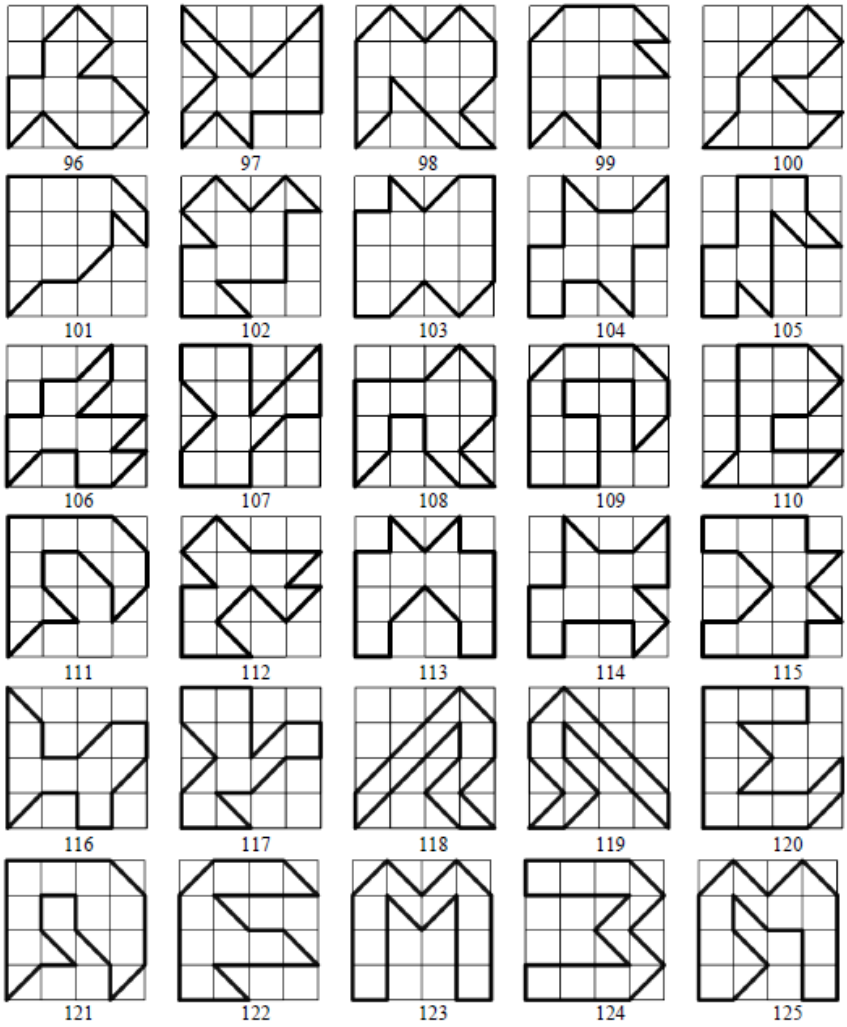
```
 byte dir[36]={0,0,0,0, 0,0,1,0, 1,0,1,0, 1,0,0,0, 1,0,0,1,
               0,0,0,1, 0,1,0,1, 0,1,0,0, 0,1,1,0};
 byte cmd[]={0,2,3,1,3,1,0};
 unsigned int time[7]={100,100,100,100,100,100,100};
 byte j,  N=7;

// Pin 2-5 has an CNC connected on Arduino board.
int xp = 5;
int xm = 4;
int yp = 3;
int ym = 2;

void setup() {
  // initialize the digital pin as an output.
  pinMode(xp, OUTPUT);
  pinMode(xm, OUTPUT);
  pinMode(yp, OUTPUT);
  pinMode(ym, OUTPUT);
}

void move(byte num){
   j=0;
   if(num<9) j=num*4;
   digitalWrite(xp, dir[j]);  j++;
   digitalWrite(xm, dir[j]);  j++;
   digitalWrite(yp, dir[j]);  j++;
   digitalWrite(ym, dir[j]); }

void loop()  // Основний цикл програми v3
{
  for(int i=0;i<N;i++){
    move(cmd[i]);
    delay(time[i]);
  }
  while(1); //stop
}
```

Додаток 4. Варіанти завдань




31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60


61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95



*Варіант завдання обирається студентом відповідно до номера за списком в групі в заданій групі варіантів (для групи 1 - номери 1...30, а для групи 2 - номери 31-60 і т.д). Наприклад, номеру за списком 9 в групі 2 відповідає варіант 39.

Лабораторна робота 3. Виведення на індикатор символьної інформації

Мета роботи: набуття практичних навичок керування для виведення статичної і динамічної інформації.

Програма роботи

1.Ознайомитися з теоретичним матеріалом, наведеним в описі.

2. Закріплення навичок розробки пристроїв IoT з використанням Arduino IDE, Simulator Arduino, Proteus VSM.

3.Отримати навички налагодження програм з використанням бібліотек.

4. Провести аналіз використовуваного об'єму ресурсів мікроконтролера.

5.Проаналізувати швидкість виведення інформації на індикатор.

6.Реалізувати виведення в рядку екрану вибрану дату у форматі рисунку 1в, а в іншому рядку - послідовність номерів векторів переміщення з лабораторної роботи 2.

7.Реалізувати виведення параметрів (число секунд, лічильник циклів), що змінюється в часі.

8.Підготувати звіт про виконану роботу.

Теоретичний матеріал

Практично будь-який МК пристрій IoT має ті або інші пристрої індикації. У простому випадку це всього декілька світлодіодів, а іноді це кольоровий графічний дисплей. Поява модулів LCD зі вбудованими контролерами значно спростила схеми сполучення. Найбільш універсальні і доступні матричні алфавітно-цифрові модулі, які дозволяють відображати цифри, букви латинського і російського алфавіту і навіть псевдографічні елементи, використовуючи можливості завантажування символів. Розширюється застосування малогабаритних монохромних графічних індикаторів. Сучасні кольорові графічні індикатори мають найширші можливості, забезпечуються сенсорними

панелями і виконують функції операторських панелей, відображення даних моніторингу стану обладнання, приміщень і оточуючого середовища.

Паралельний інтерфейс LCD – модуля

Дуже популярні індикатори на базі контролера HD44780 або його аналогів. У них забезпечується сумісність з ASCII - таблицею символів і використовується дуже простий інтерфейс: для паралельного інтерфейсу - всього 2/3 керуючих виводам і 4/8 бітна шина даних.

Контрастність LCD залежить від температури і величини напруги V0, яка подається на вхід керування. Чим більша напруга, тим менше контрастності і навпаки. Використовуючи один з виходів PWM, можна програмно керувати контрастністю.

Модулі LCD часто мають конструктивне об'єднання із спрощеною клавіатурою. Широко поширений LCD Keypad Shield с індикатором CS1602 і 5 кнопковою клавіатурою (рисунок 3.1).

а)

б)

в)

Рисунок 3.1 - Конструкція LCD Keypad Shield (а-в)

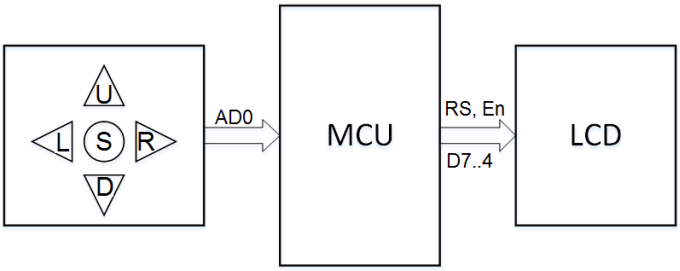


Рисунок 3.2 - Структурна схема LCD Keypad Shield

Модуль є мезонінною платою розширення і підключається до певних ліній МК пристрою. (рисунок 3.3).

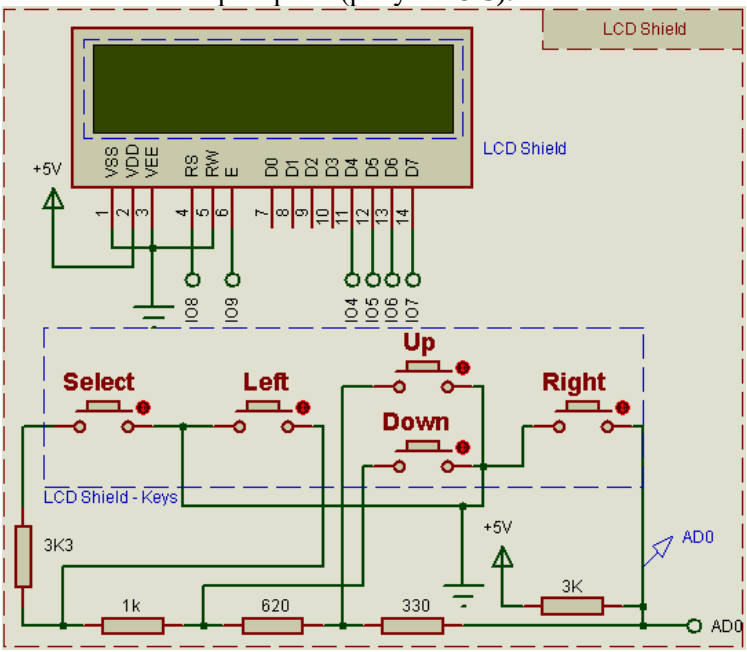


Рисунок 3.3 - Принципова схема LCD Keypad Shield

В аналоговій клавіатурі при натисненні кнопок змінюється співвідношення плечей дільника напруги 5В і на вхід AD0 АЦП

поступає напруга різних рівнів. Аналіз результату АЦП дозволяє визначити натиснуту клавішу.

Керування дисплеєм з контролером Hitachi HD44780 припускає вибір конфігурації і виконання ряду команд:

- включити/виключити дисплей (On/Off);
- 8/4- бітна шина даних/адрес (DB7…DB0) / (DB7...DB4);
- 1/2 рядковий режим;
- вибір шрифту розміром 5×7 або 5×10 пікселів;
- включити/відключити курсор (символ підкреслення);
- включити/відключити курсор (чорний квадрат);
- зрушення дисплея повністю (Scroll);

зрушення курсора вліво/вправо.

Бібліотека LiquidCrystal Library

Данна бібліотека дозволяє Arduino працювати з LCD, заснованими на Hitachi HD44780 (чи сумісними) в 4-х і 8 бітовому режимі. При цьому використовується 4 або 8 ліній даних і лінії RS, Enable керування записом команд/даних (читання зазвичай не використовується і у схемі RW = 0).

У бібліотеці LiquidCrystal реалізований розширений набір функцій :

- LiquidCrystal (rs, enable, d0, d1, d2, d3) - оголошення ліній керування індикатором з 4-бітовою шиною даних;
- LiquidCrystal lcd (8, 9, 7, 6, 5, 4); - оголошення змінної lcd типу LiquidCrystal;
- begin (cols, rows); - вказівка числа символів в рядку і числа рядків;
- lcd.begin (16, 2);
- display ()/noDisplay () - включення/виключення екрану;
- clear (); - очищення екрану;
- home (); - перехід до позиції (0,0);
- cursor ()/noCursor () - включення/виключення курсора;
- blink ()/noBlink () - вкл./выкл. мигання курсора;
- setCursor (col, row) - переклад курсора на (col, row);
- lcd.setCursor (0, 1);
- scrollDisplayLeft ()/scrollDisplayRight () - зрушення екрану вліво/вправо;

- leftToRight ()/rightToLeft () - виведення ліворуч-направо/ справа-наліво;
- autoscroll ()/noAutoscroll () - вкл./викл. зрушення екрану перед виведенням символу;
- print (x) - друк символу, рядка або рядкового представлення числа;
- write (x) - друк символу з кодом x;
- Модель LCD в Simulator Virtronics;
- Simulator Virtronics в теці ...\Virtronics\LCD\ містить приклади використання функцій бібліотеки LiquidCrystal:
- HelloWorld.ino - друк тексту;
- Display.ino - вкл./викл. екрану;
- Blink.ino - мигання екрану;
- Cursor.ino - вкл./викл. курсора;
- setCursor.ino - встановлення курсора;
- Scroll.ino - керування зрушенням екрану;
- Autoscroll.ino – автозрушення екрану;
- TextDirection.ino - керування напрямом виведення тексту;
- SerialDisplay.ino - перенаправлення прийнятих даних UART на екран.

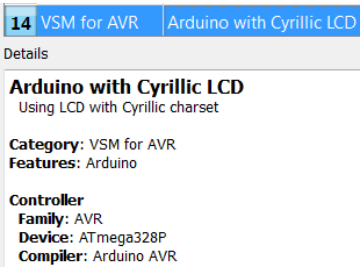
Приклади є шаблонами для розробки програм, що використовують виведення на LCD. Необхідно провести аналіз прикладів з використанням симулятора.

У прикладі SerialDisplay.ino прийнятий UART рядок після очищення екрану виводиться з позиції (0,0).

Рисунок 3.4 – Виведення символьної послідовності на екран

Модель LCD Shield в Proteus

Модель знаходиться у вкладці VSM for AVR :



Модуль LCD керується МК через цифрові лінії виведення відповідно до рисунку 3.4, наприклад від ATmega328 (рисунок 3.5).

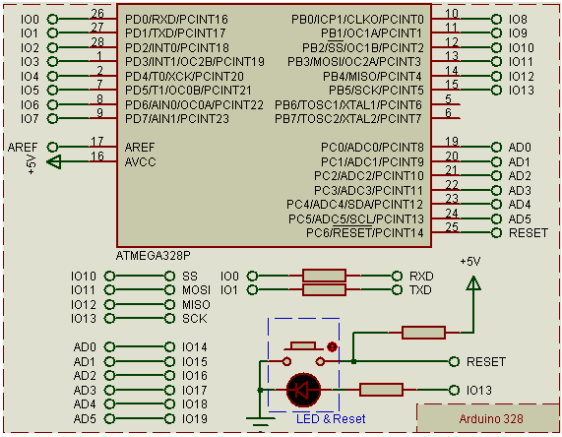


Рисунок 3.5 – МК модуль керування на основі МК ATmega328

Приклади відображення інформації про натиснуті кнопки на екрані LCD в моделі Proteus наведені на рисунку 3.6.

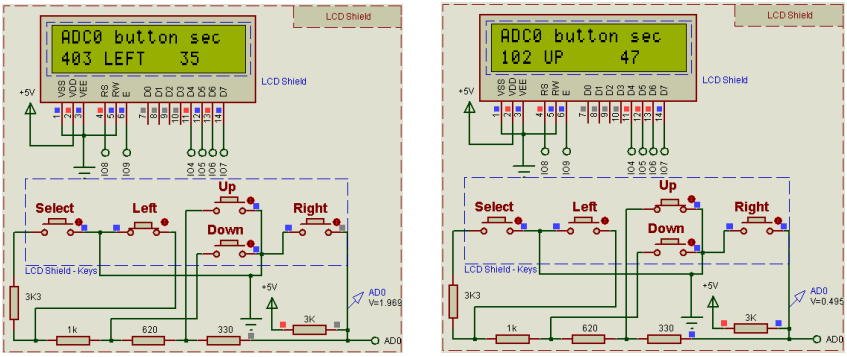


Рисунок 3.6 – Моделювання виведення інформації на LCD

Інформація на екрані може включати фіксований текст, оновлюванні значення (код символу) в певній позиції і “рухомий рядок” (рисунок 3.7).

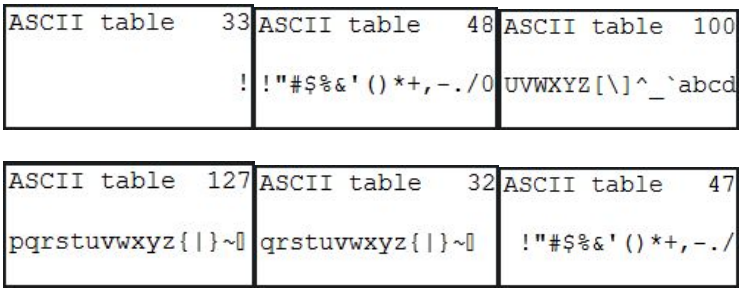


Рисунок 3.7 – Приклад скролінгу рядка

Задача 1. Вивести в рядку 1 екрану вибрану дату у форматі рисунку 3.1в, а в іншому рядку - послідовність номерів векторів переміщення з лабораторної роботи 2.

Задача 2. Додати до попередньої задачі виведення з позиції (0,12) параметра (число секунд, лічильник циклів), що змінюється в часі.

Задача 3. Вивести рухомий рядок символів, в другому рядку LCD.

Зміст звіту

1. Схеми апаратних засобів для вирішення завдань.
2. Тексти програм для завдань.
3. Результати моделювання.
4. Результати виконання експериментів.
5. Висновки по роботі.

Контрольні запитання:

1. Скільки ліній потрібно для підключення символьного LCD?
2. Охарактеризуйте інтерфейс індикатора (послідовний/паралельний, асинхронний/синхронний і так далі).
3. Як очистити екран?
4. Яку інформацію можна виводити на екран LCD?
5. Як змінити напрям виведення символів?
6. Як вивести символ в середині рядка 2?
7. Яка зона дії скролінгу?

8. Які функції реалізовані у бібліотеці LiquidCrystal?

9. У чому відмінність методів print і write бібліотеки Liquid Crystal?

10. Як визначити натиснену клавішу на аналоговій клавіатурі?

Література

1.Соммер У. Программирование микроконтроллерных плат Arduino/Freeduino. – СПб.:БХВ – Петербург, 2012. – 256с.

2.Подключение LCD 1602 (HD44780) к Arduino. [Електронний ресурс]. – Режим доступу: - http://zelectro.cc/LCD1602

3.Среда разработки Arduino. [Електронний ресурс]. – Режим доступу: http://www.arduino.cc/en/Main/Software

4.Simulator for Arduino. [Електронний ресурс]. – Режим доступу: http://virtronics.com.au/Data/SetupFree.zip.

5.Программирование Ардуино. [Електронний ресурс]. – Режим доступу: http://arduino.ru/Reference

6.PROTEUS VSM. Среда виртуального моделирования. - PROTEUS-d.pdf [Електронний ресурс]. Режим доступу: http://proteus123.narod.ru/

Лабораторна робота 4. Розробка додатків IoT з використанням таймерів- лічильників

Мета роботи: набуття практичних навиків використання таймерів- лічильників в додатках IoT

Програма роботи

1. Ознайомитися з теоретичним матеріалом, наведеним в описі, а також в [1, 2].
2. Розробити та налагодити програми асинхронного обміну з використанням віртуальних пристроїв і інструментів.
3. Закріпити навички розробки підключених пристроїв IoT для формування імпульсних сигналів і роботи в реальному масштабі часу з використанням Arduino IDE, Proteus VSM.
4. Отримати навички налагодження програм і аналізу імпульсних сигналів з використанням віртуальних пристроїв і інструментів.
5. Реалізувати програму вибору однієї з трьох частот за варіантом за допомогою трьох кнопок.
6. Реалізувати програму керування двигуном (освітленням) з циклічною 3-ступінчастою зміною шпаруватості ШІМ за індивідуальним варіантом.
7. Провести моделювання кута повороту сервопривіда та розрахувати значення тривалості імпульсів для послідовної установки в задані положення.
8. Провести аналіз використовуваного об'єму ресурсів МК та проаналізувати тимчасові діаграми імпульсних сигналів
9. Закріплення навичок завантаження програм в пам'ять МК навчального стенду і налагодження з використанням осцилографа і логічного аналізатора.
10. Підготувати звіт про виконану роботу.

Теоретичні відомості

МК ATmega328 містить 3 таймер-лічильники - TC0 (8 бітовий), TC1 (16-бітовий) і TC2 (8 бітовий). Вони побудовані на основі двійкових лічильників. Схема і принцип роботи 8-бітового

двійкового лічильника на U1A, B (рисунок 4.1) ілюструється тимчасовими діаграмами виходів лічильника і результату ЦАП (DAC1) послідовності станів лічильника.

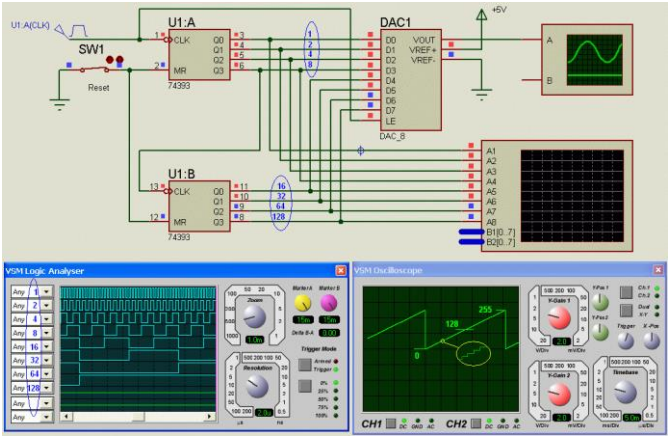


Рисунок 4.1 – Схема і діаграми станів двійкового лічильника

Стани лічильника в двійковому коді - 00000000...11111111, десятковому - 0...255 і шістнадцятирічному - 0x00...0xff. Цикл рахунку повторюється через 28…256 імпульсів від внутрішнього або зовнішнього джерела. TC0 пов'язаний з виведенням T0/PD4 (pin D4), а TC1 з TC1/PD5 (pin D5), а TC2 не має такого зв'язку (рисунок 4.2). ICP (D8) - вхід захоплення, виходи ШІМ TC0 - OC0A, B (D5, D6), ШІМ TC1 - OC1A, B (D9, D10), ШІМ TC2 - OC2A, B (D11, D3).

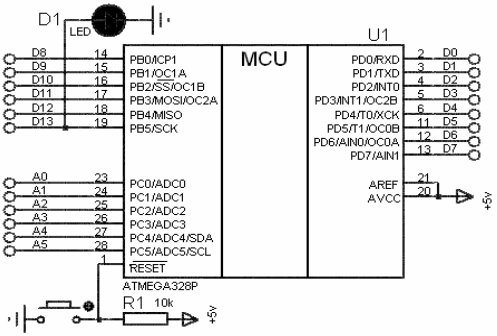


Рисунок 4.2 – Призначення виводів МК ATmega328P

ТЛ мають наступні режими:

0) зупинка;

1) рахунок кожного тактового імпульсу (переддільник 1/1);

2) рахунок кожного 8-го тактового імпульсу (переддільник 1/8)№

3) рахунок кожного 64-го тактового імпульсу (переддільник 1/64);

4) рахунок кожного 256-го тактового імпульсу (переддільник 1/256);

5) рахунок кожного 1024-го тактового імпульсу (переддільник 1/1024);

6) рахунок перепадів 1→0 вхідних імпульсів;

7) рахунок перепадів 0→1 вхідних імпульсів;

8) формування ШІМ – сигналів;

9) вимір інтервалу між заданими змінами сигналу на вході ICP.

З таймерами пов'язані регістри керування і стану, за допомогою яких задаються їх функції. Можливе пряме звернення до цих регістрів, або використання бібліотечних функцій.

Функції IDE Arduino

Advanced I/O:

- tone (pin, frequency) - генерація імпульсів заданої частоти frequency (Гц) на виведенні pin;

- noTone (pin) - припинення генерації;

- pulseIn (pin, value) - повертає тривалість рівня value (HIGH, LOW) імпульсу на виводі pin - analogWrite (pin, value) - генерація ШІМ з параметром value=0.255 на виведенні pin=3 (5,6,9,10,11);

- 3,11 - TC2;

Time:

- millis () - повертає число мілісекунд від запуску програми (період рахунку близько 50 днів);

- micros () - повертає число мікросекунд від запуску програми (період рахунку близько 70 днів);

- delay (value) - затримка на value мілісекунд;

- delayMicroseconds (value) - затримка на value мікросекунд;

Функція tone з використанням ТЛ2:

- tone (pin, unsigned int frequency);
  0…13 Гц
- tone (12,440) - генерація 440 Гц на pin D12;
- tone (pin, unsigned int frequency, unsigned long duration);
  0…13 Гц мс
- tone(9,800,500) - 500 мс генерація 800 Гц на pin D8;
- noTone(pin);
- noTone(12) - припинення генерації на pin D12.

Приклад 1. Ступінчата зміна тону (тут потрібно знайти помилку).

```
 int ledPin = 11;
unsigned char val = 64;

void setup(){
  pinMode(ledPin, OUTPUT);
}

void loop()
{
  tone(LedPin,val*4); val+=16;
  if(val<64) val=64;
  delay(250);
}
```

Для налагодження програми можуть використовуватися проекти-приклади з Proteus з віртуальним осцилографом (рисунок 4.3). З отриманої за допомогою віртуального осцилографа тимчасової діаграми періоду сигналу частоту можна знайти таким чином. Добуток величини розгортки по горизонталі в мс і числа поділок в одному періоді дає тривалість періоду Т сигналу в мс, а частота визначається як F = 1000/T, Гц. Порівняти отриману частоту із заданою.

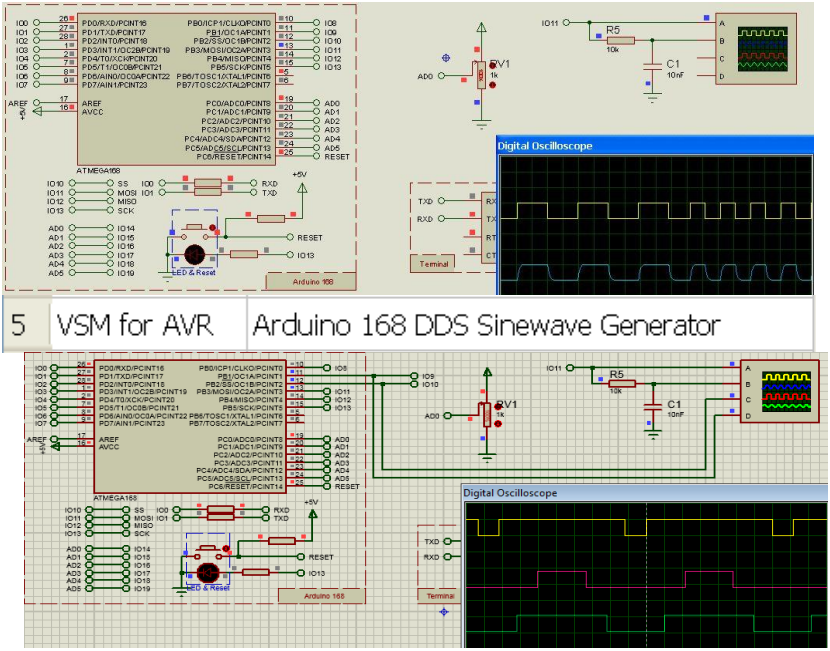


Рисунок 4.3 – Приклади керування ШІМ з Proteus

Для прикладу "DDS Generator" на основі ATmega168 слід виконати приведені рисунку 4.4 налаштування.

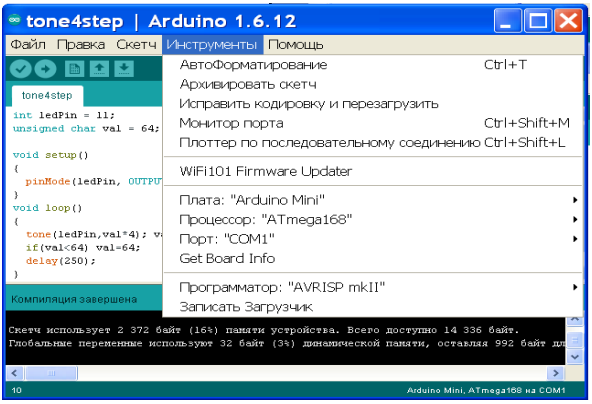
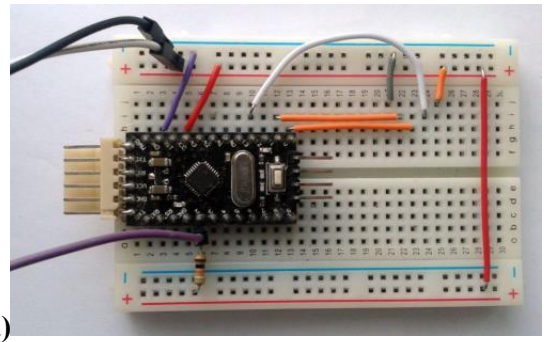


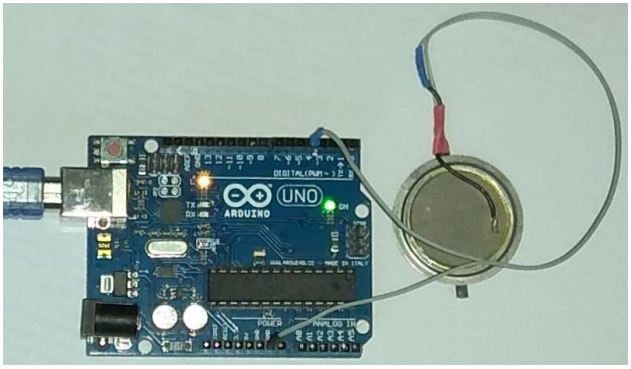
Рисунок 4.4 – Вибір плати в середовищі розробки

При значному відхиленні результату вимірів від завдання слід перевірити встановлену тактову частоту в моделі МК, враховуючи, що за замовчуванням програма компілюється для тактової частоти 16 Мгц.

Для комплексного налагодження програми є стенди на основі ATmega168, ATmega328, осцилограф, частотомір і п'єзовипромінювачі для контролю вихідних сигналів, генерованих під керуванням програм (рисунок 4.4).



а)



б)

Рисунок 4.5 - стенди на основі ATmega168 (а) и ATmega328 (б)

При необхідності можуть використовуватися стенди інших форм- факторів на основі ATmega328, ATmega2560 і інших МК.

Формування ШІМ

Функція **analogWrite(pin, value)** – виведення ШІМ сигналу, де pin=(3,5,6,9,10,11), value=0..255.

Приклад 2. Вивід ШІМ сигналу зі змінною шпаруватістю сигналу.

```
int ledPin = 9;
unsigned char val = 0;
void setup()
{
  pinMode(ledPin, OUTPUT);
}
void loop()
{
  analogWrite(ledPin, val);
  val++;
  delay(10);
}
```

Для налагодження програми можна використовувати проект-приклад з Proteus з віртуальним осцилографом (рисунок 4.3) для вимірювання параметрів ШІМ.

В прикладі 3 розглянуто використання функцій відліку часу millis(), micros().

Приклад 3. Використання millis() – програма Blink без delay()

```
// Blink without Delay
const int ledPin =  13;
int ledState = LOW;

unsigned long previousMillis = 0;
const long interval = 1000;

void setup() {
  pinMode(ledPin, OUTPUT);
```

```
}

void loop()
{
  unsigned long currentMillis = millis();
  if(currentMillis - previousMillis >= interval)
  {
    previousMillis = currentMillis;
    if (ledState == LOW)
      ledState = HIGH;
    else
      ledState = LOW;
    digitalWrite(ledPin, ledState);
  }
}
```

Для налагодження програми можна використовувати проект-приклад з Proteus з віртуальним осцилографом (рисунок 4.3) для вимірювання параметрів імпульсів.

Задача 1. Формування меандру з частотами заданих нот

У таблиці 4.1 приведені частоти основних нот для індивідуального завдання. Виділено 48 частот - від 130.82 Гц до 1975.5 Гц (варіанти 1 .. 48). Використовуючи приклад, написати програму для відтворення тону з частотою, що відповідає заданому варіанту. Провести моделювання з використанням Proteus (рисунок 4.6). З допомогою віртуального осцилографа визначити тривалість періоду імпульсів і розрахувати їх частоту. Порівняти результат із завданням. Завантажити програму в МК Arduino Uno (Nano), прослухати тон за допомогою п’єзовипромінювача. Виміряти частотоміром частоту тону. Побудувати програму вибору однієї з трьох частот за варіантом з таблиці .4.2 за допомогою трьох кнопок. За допомогою четвертої кнопки вибирається послідовність відтворення трьох нот.

Таблиця 4.1 - Частоти звучання нот

Вар	Частота, Гц	Вар	Частота, Гц	Вар	Частота, Гц
1	130.82	13	261.63	25	523.25
2	138.59	14	277.18	26	554.36
3	147.83	15	293.66	27	587.32
4	155.56	16	311.13	28	622.26
5	164.81	17	329.63	29	659.26
6	174.62	18	349.23	30	698.46
7	185.00	19	369.99	31	739.98
8	196.00	20	392.00	32	784.00
9	207.00	21	415.30	33	830.60
10	220.00	22	440.00	34	880.00
11	233.08	23	466.16	35	932.32
12	246.96	24	493.88	36	987.75

Провести моделювання з використанням Proteus (рисунок 4.6). Завантажити програму в МК Arduino Uno (Nano), прослухати тон за допомогою п’єзовипромінювача. Виміряти частотоміром частоти тону.

Таблиця 4.2. - Трьохканальні комбінації

Вар.	F1, Гц	F2, Гц	F3, Гц	Вар.	F1, Гц	F2, Гц	F3, Гц
1	130,82	233,08	523,25	**14**	155,56	246,96	554,36
2	138,59	246,96	554,36	**15**	164,81	261,63	587,32
3	147,83	261,63	587,32	**16**	174,62	277,18	622,26
4	155,56	277,18	622,26	**17**	185	293,66	659,26
5	164,81	293,66	659,26	**18**	196	311,13	698,46
6	174,62	311,13	698,46	**19**	207	329,63	739,98
7	185	329,63	739,98	**20**	220	349,23	784
8	196	349,23	784	**21**	130,82	369,99	830,6
9	207	369,99	830,6	**22**	138,59	392	880
10	220	392	880	**23**	147,83	415,3	932,32
11	130,82	415,3	932,32	**24**	155,56	440	987,75
12	138,59	440	987,75	**25**	164,81	277,18	523,25
13	147,83	233,08	523,25	**26**	174,62	246,96	830,6

Задача 2. Побудувати програму керування двигуном (освітленням) з циклічною 3-ступінчастою зміною шпаруватості ШІМ за індивідуальним варіантом з таблиці 4.3. Для заданих значень шпаруватості 0..99% необхідно розрахувати значення параметрів для функцій формування ШІМ - analogWrite (pin, val). Провести моделювання за допомогою віртуального середовища на основі проекту на рисунку 4.3 або на рисунку 4.6. Виміряти період, частоту і шпаруватість сигналу.

Програмування генератора ШІМ

ШІМ використовується для керування двигунами постійного струму і світлодіодним освітленням.

Таблиця 4.3. Значення скважності ШІМ

Вар	Q1, %	Q2, %	Q3, %	Вар	Q1, %	Q2, %	Q3, %
1	19	57	70	**16**	10	59	88
2	27	43	89	**17**	30	59	88
3	11	40	90	**18**	10	46	89
4	29	48	72	**19**	16	43	90
5	22	41	73	**20**	19	54	80
6	26	55	74	**21**	17	43	74
7	22	50	72	**22**	21	50	81
8	30	59	83	**23**	28	53	86
9	14	42	90	**24**	26	56	80
10	19	50	87	**25**	17	54	80
11	15	40	87	**26**	13	49	75
12	23	53	82	**27**	14	50	79
13	19	49	76	**28**	10	57	81
14	30	51	74	**29**	15	43	75
15	20	53	86	**30**	23	49	87

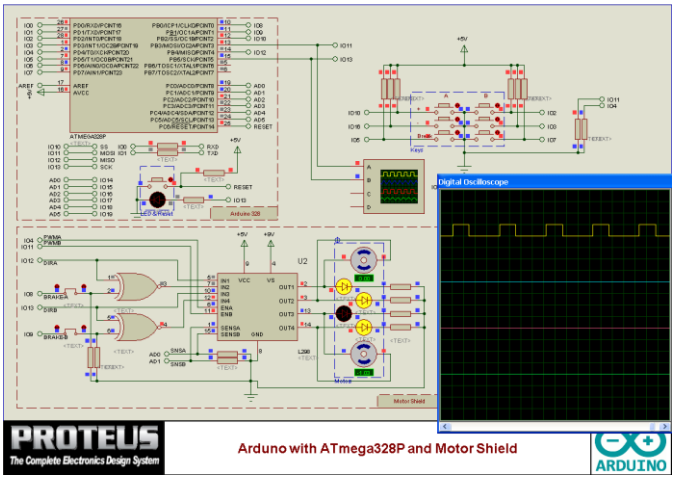


Рисунок 4.6 – Налагодження програми в середовищі Proteus

З отриманої за допомогою віртуального осцилографа тимчасової діаграми ШІМ сигналу (рисунок 4.7) частоту і шпаруватість можна знайти наступним чином. Величина розгортки по горизонталі - 1 мс / справа. Тоді період сигналу становить близько 4 мс, що відповідає частоті 250 Гц. Тривалість імпульсу - близько 1.4 мс, тоді шпаруватість становить: 1.4 / 4 * 100 = 35%, яка досягається використанням параметра функції analogWrite зі значенням 1.4 / 4 * 256 = 90.

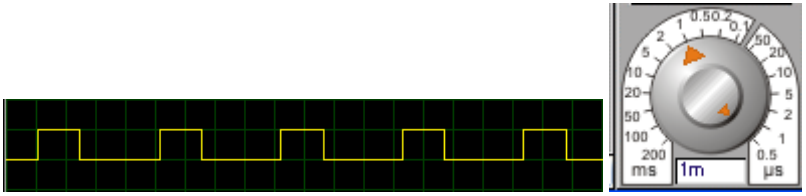


Рисунок 4.7 – Вигляд ШІМ сигналу з частотою 250 Гц і шпаруватістю 35%

Завантажити програму в МК і перевірити роботу пристрою.

Задача 3. Для заданих в таблиці 4.4 значень кута повороту валу сервопривода -90...+ 90% необхідно розрахувати значення тривалості імпульсів для послідовної установки в задані

положення. Спеціальні бібліотеки містять методи для керуванням сервоприводами. Провести моделювання за допомогою віртуального середовища на основі проекту на рисунку 4.8. Виміряти період, частоту і тривалість імпульсів керування приводом.

Керування сервоприводом

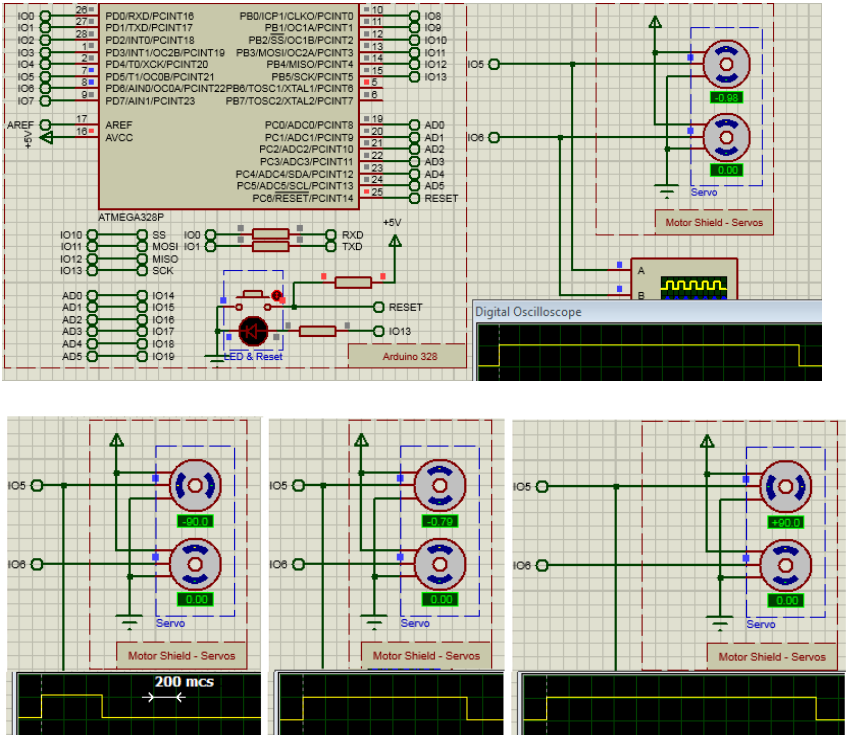


Рисунок 4.8 – Моделювання сервопривода в Proteus

Кут повороту вала сервоприводу -90º .. + 90º задається послідовністю імпульсів з частотою 50-60 Гц і тривалістю 500 .. 2500 мкс.

Таблиця 4.4. Значення кута повороту вала сервопривода

Вар	α1,º	α2,º	α3,º	Вар	α1,º	α2,º	α3,º
1	-80	-28	54	16	-56	26	89
2	-85	3	58	17	-89	26	78
3	-64	-8	60	18	-43	6	61
4	-66	-20	45	19	-52	-2	50
5	-53	-14	56	20	-63	8	75
6	-63	10	82	21	-64	24	89
7	-80	-21	45	22	-66	28	66
8	-44	-30	47	23	-51	14	70
9	-46	1	59	24	-45	29	80
10	-89	8	77	25	-83	-15	59
11	-83	-20	58	26	-90	29	70
12	-63	5	59	27	-84	8	62
13	-56	20	61	28	-49	-6	69
14	-78	29	62	29	-64	-11	87
15	-68	-5	69	30	-81	5	47

Завантажити програму в МК і перевірити роботу пристрою. Виміряти параметри сигналів керуванням сервоприводом.

Зміст звіту

1. Схеми для проведених експериментів.
2. Тексти програм для завдань.
3. Результати моделювання (тимчасові діаграми, оцінка часових параметрів).
4. Результати виконання експериментів.
5. Результати експериментів з використанням навчальних стендів і вимірювальних приладів, фото і відео експериментів (при необхідності).
6. Висновки по роботі.

Контрольні запитання

1.Схема і принцип роботи 8-бітового двійкового лічильника?
2.Перерахуйте режими роботи ТЛ.
3.Опишіть функцію tone з використанням ТЛ2.
4.Як формується ШІМ сигнал?
5.Як визначити частоту сигналу ШІМ?
6.Як визначити шпаруватість сигналу ШІМ?
7.Для чого використовується ШІМ?
8.Як визначити тривалість періоду імпульсів.
9. Як здійснити вибір плати в середовищі розробки?
10. Як завантажити програму в МК?

Література

1. Опис навчальних стендів.
2. Соммер У. Программирование микроконтроллерных плат Arduino/Freeduino.- СПб.: БХВ – Петербург, 2012.- 256с.
3. Brian W. Evans Блокнот програмиста. 2007.
4. Среда разработки Arduino. [Електронний ресурс]. – Режим доступу: http://www.arduino.cc/en/Main/Software
5. Simulator for Arduino. [Електронний ресурс]. – Режим доступу: http://virtronics.com.au/Data/SetupFree.zip.
6. Программирование Ардуино. [Електронний ресурс]. – Режим доступу: http://arduino.ru/Reference
7. PROTEUS VSM. Среда виртуального моделирования. - PROTEUS-d.pdf [Електронний ресурс]. Режим доступу: http://proteus123.narod.ru/

Лабораторна робота 5. Асинхронний обмін між пристроями IoT

Мета роботи: набуття практичних навичок організації асинхронного обміну в МК пристроях.

Програма роботи

1. Ознайомитися з теоретичним матеріалом, наведеним в описі, а також в [1, 2].
2. Розробити та налагодити програми асинхронного обміну з використанням віртуальних пристроїв і інструментів.
3. Закріпити навички розробки підключених пристроїв IoT з використанням Arduino IDE, Proteus VSM.
4. Закріпити навички завантаження програм в пам'ять МК навчального стенду і налагодження з використанням осцилографа і логічного аналізатора.
5. Дослідити процес групове керуванням станом реле 4 channel Relay Shield.
6. Дослідити стани входів A, B Motor Shield.
7. Проаналізувати тимчасові діаграми асинхронного обміну та провести аналіз використовуваного обсягу ресурсів МК.

Теоретичний матеріал

Однієї з найважливіших складових роботи сучасних обчислювальних засобів автоматики є обмін даними між пристроями. У сучасних AVR- мікроконтролерах передбачено наявність вбудованого універсального асинхронного приймача (UART), що забезпечує реалізацію дуплексного асинхронного обміну із зовнішніми пристроями. МК ATmega328 (див. додаток) має універсальний синхронно-асинхронний прийомо- передатчик USART0 з розширеними функціональними можливостями.

На рисунку 5.1 наведений приклад сполучення двох мікроконтролерів - ведучого і веденого з використанням UART. Якщо до ліній прийому (RxD) і передачі (TxD) підключити перетворювачі рівня TTL - RS232, то ведений МК можна підключити до COM- порту персонального або промислового

комп'ютера. Існують перетворювачі UART - USB, UART - RS485, UART - Bluetooth та ін., які розширюють комунікаційні можливості

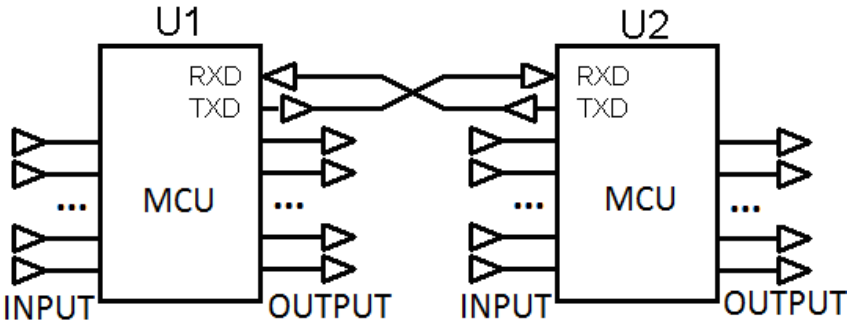


Рисунок 5.1 - Сполучення двох МК з використанням UART

Входи (INPUT) і виходи (OUTPUT) пов'язані з універсальними лініями портів МК - PORTA..PORTD, які програмно налаштовуються на відповідний режим роботи.

При програмуванні UART в простому випадку вирішуються три завдання:

1.Ініціалізація UART (задання швидкості, формату даних і режимів роботи).

2.Організація передачі даних.

3.Організація прийому даних.

Ініціалізація UART

Ініціалізація UART передує процесу обміну і включає: установку швидкості обміну, режиму роботи і формату посилок.

Оскільки при асинхронному обміні відсутня окрема лінія синхронізації, для забезпечення коректності результатів прийому/передачі інформації тимчасові параметри сигналів мають бути чітко погоджені за часом. Для підтримки стабільної швидкості обміну V в UART AVR- мікроконтролерів використовується спеціальний генератор швидкості передачі -

дільник частоти FCK тактового генератора МК. Коефіцієнт ділення задається кодом в регістрі UBRR відповідно до виразу:

$$UBRR = \frac{F_{CK}}{16 \cdot V} - 1, \qquad (1)$$

$$V = \frac{F_{CK}}{16 \times (UBRR + 1)}$$

Відхилення швидкості обміну від заданої має бути не більше ±5%. При встановлені швидкості передачі для типових значень швидкості V і частот синхронізації FCK можна вибирати значення UBRR з таблиці 5.1, розраховані по (1).

Таблиця 5.1. Встановлення швидкості обміну UART

V	8 MHz	Error, %	16 MHz	Error, %
2400	207	0.2	416	0.2
4800	103	0.2	207	0.2
9600	51	0.2	103	0.2
19200	25	0.2	51	0.2
57600	-	-	16	2.1
115200	-	-	8	3.5

Для F_{CK}=16000000 Гц встановлені значення UBRR = 103_{10} = 67_{16} = 0x67 відповідають швидкості обміну BaudRate = 9600 бод.

Встановлення режиму роботи UART відбувається з допомогою встановлення біт регістра UCR:

7	6	5	4	3	2	1	0
0	0	0	RXEN	TXEN	0	0	0

Біти цього регістра дозволяють включення приймача (RXEN) і передавача (TXEN). При скиданні усі біти встановлюються в 0, тому приймач і приймач UART відключаються. У секції ініціалізації програми UCR необхідно налаштувати. При записі коду 000110002 = 0x18 приймач і передавач включаються.

Керування приймачем

Стан приймача визначається прапорами: закінчення прийому (RxC), закінчення передачі усіх даних (TxC), звільнення буфера передавача (UDRE). Ці прапори входять до складу регістра USR:

7	6	5	4	3	2	1	0
RxC	TxC	UDRE	-	-	-	-	-

Передача починається в момент завантаж передаваннях даних в регістр UDR передавача з його буферного регістра. Звільнення буферного регістра призводить до установки UDRE=1, що дозволяє завантажити в нього черговий байт, не чекаючи закінчення циклу передачі і установки TxC=1. Прапори передавача можна опитувати програмно, або викликати переривання. Закінчення прийому чергового байта фіксується установкою прапора RxC=1, який можна опитувати програмно, або викликати переривання, залежно від використовуваного режиму обміну.

Організація обміну з програмним керуванням

Якщо МК використовує послідовний канал для коротких однобайтних (8 бітових) повідомлень, то приймати і передавати дані можна, просто опитуючи прапори готовності приймача і передавача UART. На рисунку 5.2 наведений приклад: через UART з програмним опитуванням прапора приймача (очікуванням USR.7=1, що еквівалентно запису RxC=1). Прийнятий в UDR байт зчитується і передається в PORTB. Далі перевіряється готовність буферного регістра передавача до прийому даних (очікується USR.5=1, що еквівалентно запису UDRE=1). Код, зчитаний з ліній PIND, завантажується в регістр передавача UDR.

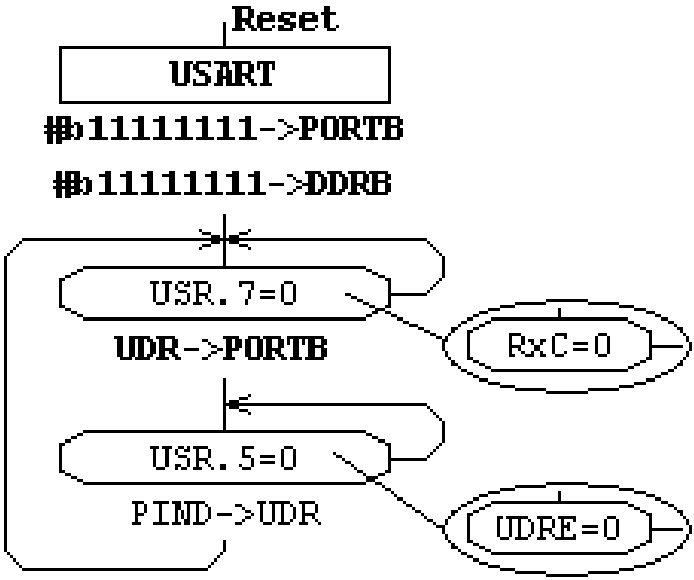


Рисунок 5.2 - Дуплексний режим обміну з програмним опитуванням прапорів готовності

Проаналізувавши приведені фрагменти, можна помітити, що і у разі передачі, і у разі прийому даних використовується регістр UDR. Фізично він є двома окремими регістрами, доступ до яких відбувається за одним адресом. При записі відбувається запис в регістр передавача, при читанні - зчитується регістр приймача.

Абстрактний рівень обміну у відкритій платформі Arduino

У Arduino UNO (Nano і ін.) лінія D0 пов'язана з входом приймача RxD, а лінія D1 - з виходом передавача TxD. З'єднання двох пристроїв для асинхронного обміну показано на рисунку 5.3.

З приймачем і передавачем UART пов'язані буфери, в яких зберігаються прийняті байти і ті що стоять в черзі на передачу. Реалізований механізм переривань після закінчення прийому і передачі кожного байта, що захищає від втрати даних.

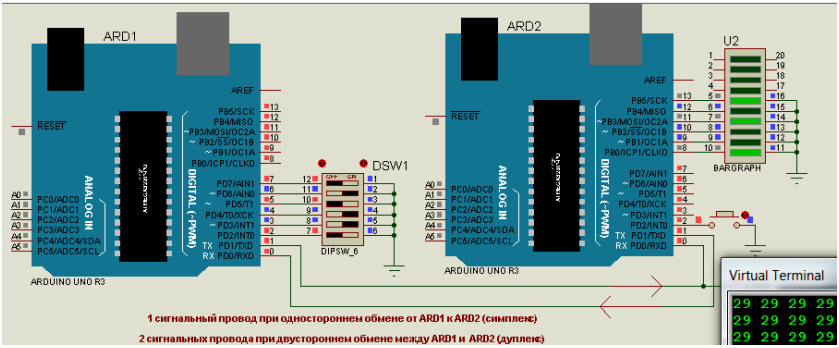


Рисунок 5.3 - Передача даних від ARD1 до ARD2. Схема з'єднання

Набір з 6 перемикачів DSW1 (рисунок. 5.4), пов'язаний з вхідними лініями D7...2 є джерелом вхідних даних (x5…0) для передачі (на рисунку 5.3 - комбінація сигналів 101001 = 0x29).

Алгоритм обміну :

- ARD1 опитує стан входів, пов'язаних з датчиками (x5…0);
- формується код для передачі X=<0,0, x5, x4, x3, x2, x1, x0>;
- передача коду X від ARD1 до ARD2;
- прийом і обробка коду X=<0,0, x5, x4, x3, x2, x1, x0> у ARD2 (виділення значень x5, x4, x3, x2, x1, x0);
- виведення стану датчиків на розряди індикатора U2: D13=x5, D12=x4, D11=x3, D10=x2, D9=x1, D8=x0;
- передача коду підтвердження від ARD2 до ARD1 (при необхідності).

Таким чином, сигнали поступають від DSW1 побітно розміщується в переданому байті відповідно до формату асинхронної посилки. При прийомі біти виділяються і відображаються на виходах МК ARD2, пов'язаних зі світлодіодними індикаторами. U2 (рисунок. 5.4).

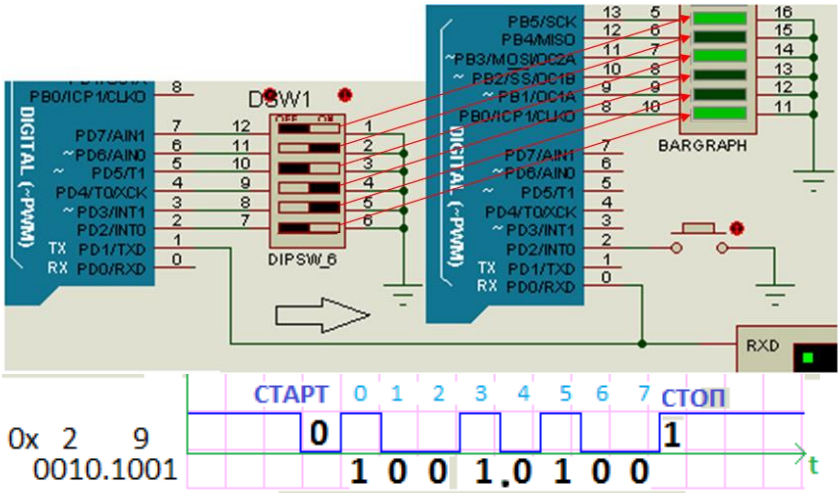


Рисунок 5.4 - Передача даних від ARD1 до ARD2. Формат посилок

Віртуальні і фізичні стенди для налагодження програм асинхронного послідовного обміну у відкритій платформі Arduino

У додатку 1 наведена віртуальна модель керування 4 - канальним реле в середовищі Proteus, зовнішній вигляд і принципова схема 4 channel Relay Shield. Функціональна схема підключення представлена на рисунку.5.5.

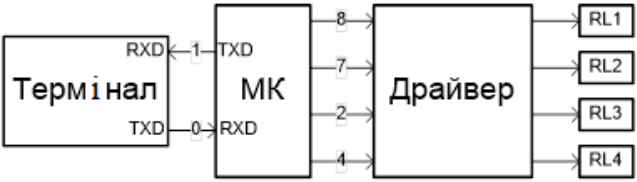


Рисунок. 5.5 – Функціональна схема підключення 4 channel Relay Shield до Arduino

Термінал підключається до ліній прийому (D0) і передачі (D1) МК плати. У нім встановлюється необхідний формат і швидкість

обміну. МК через вихідні лінії D8, D7, D2, D4 керує драйвером, який включає реле RL1 ...4.

В ході виконання роботи необхідно побудувати програми в IDE Arduino відповідно до завдань, виконати їх налагодження у симуляторі і за вказівкою запрограмувати МК віртуального і навчального стендів для перевірки роботи програм.

Задача 1. Групове керування станом реле 4 channel Relay Shield (Додаток 1). Від терміналу поступають коди символів. З кожного 8 - розрядного коду виділяються молодші 4 розряди. Їх комбінації (0000 ...1111) визначають необхідний стан виходів 8,7,2,4. Побудувати програму декодування кодів, що приймаються, виконати налагодження в Proteus або симуляторі і перевірити роботу програми.

Задача 2. Контроль стану входів A, B Motor Shield (Додаток 2). На входи 10,16,8,2,3,7 плат поступають сигнали {0,1} від 6 кнопок. Можуть бути 64 комбінації стану входів - 000000 . 111111. Побудувати програму опитування стану вказаних входів і передачі 6 - символьних рядків "000000" . "111111" вхід терміналу для відображення. Опитування виконується з періодом 500 мс. Виведення нового стану входів розпочинається з нового рядка. Побудувати програму, виконати налагодження в Proteus або симуляторі.

Зміст звіту

1. Схеми апаратних засобів для вирішення задач.
2. Тексти програм задач.
3. Результати моделювання.
4. Результати виконання експериментів.
5. Висновки по роботі.

Контрольні запитання

1. Що таке UART?
2. Які завдання вирішуються при програмуванні UART?
3. Встановлення режиму роботи UART

4. Який механізм реалізації обміну даними з використанням UART?
5. Як відбувається організація обміном з програмним керуванням?
6. Як відбувається ініціалізація даних в UART?
7. Як підключити термінал до мікроконтролера?
8. Як визначається стан закінчення прийому даних через UART?
9. Як визначається стан закінчення передачі даних через UART?
10. Як з'єднати два пристрої для асинхронного обміну?

Література

1. Опис навчальних стендів.

2. Соммер У. Программирование микроконтроллерных плат Arduino/Freeduino.- СПб.: БХВ – Петербург, 2012.- 256с.

3. Brian W. Evans Блокнот програмиста. 2007.

4. Среда разработки Arduino. [Електронний ресурс]. – Режим доступу: http://www.arduino.cc/en/Main/Software

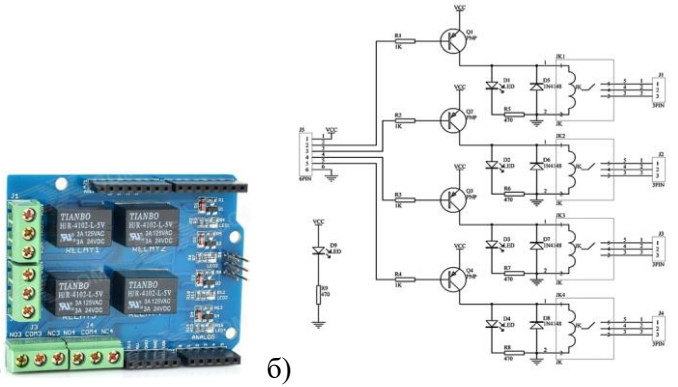
5. Simulator for Arduino. [Електронний ресурс]. – Режим доступу: http://virtronics.com.au/Data/SetupFree.zip.

6. Программирование Ардуино. [Електронний ресурс]. – Режим доступу: http://arduino.ru/Reference

7. PROTEUS VSM. Среда виртуального моделирования. - PROTEUS-d.pdf [Електронний ресурс]. Режим доступу: http://proteus123.narod.ru/

8.JoyStick Shield V1[Електронний ресурс]. – Режим доступу: https://www.elecfreaks.com/wiki/ index.php?title=Joystick_Shield

Додаток 1. 4 channel Relay Shield - модуль розширення (4-канального реле



а) б)

Рисунок 5.6 – 4 channel Relay Shield – зовнішній вигляд (а) і принципова схема (б)

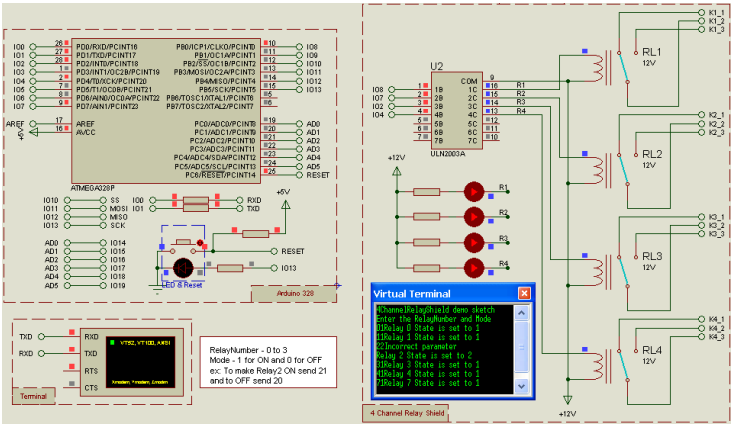
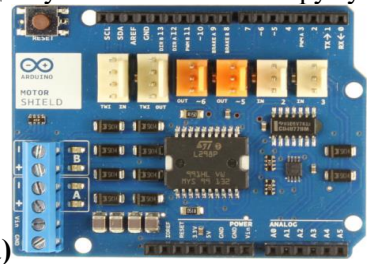


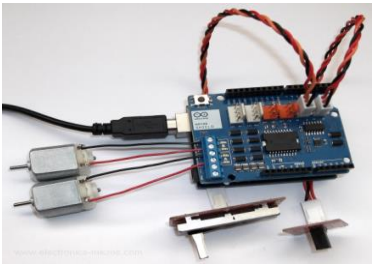
Рисунок 5.7 – Підключення 4 channel Relay Shield до Arduino– функціональна схема (Proteus)

Додаток 2. Motor Shield – модуль розширення для керування двигунами постійного струму

а)



б)



в)

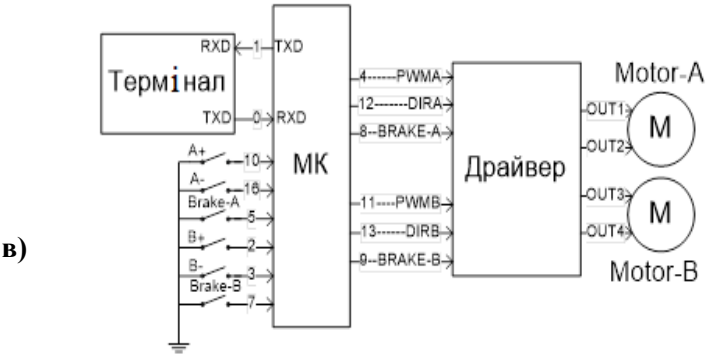


Рисунок 5.8 - Motor Shield – зовнішній вигляд вид (а), застосування (б); спрощена схема (в)

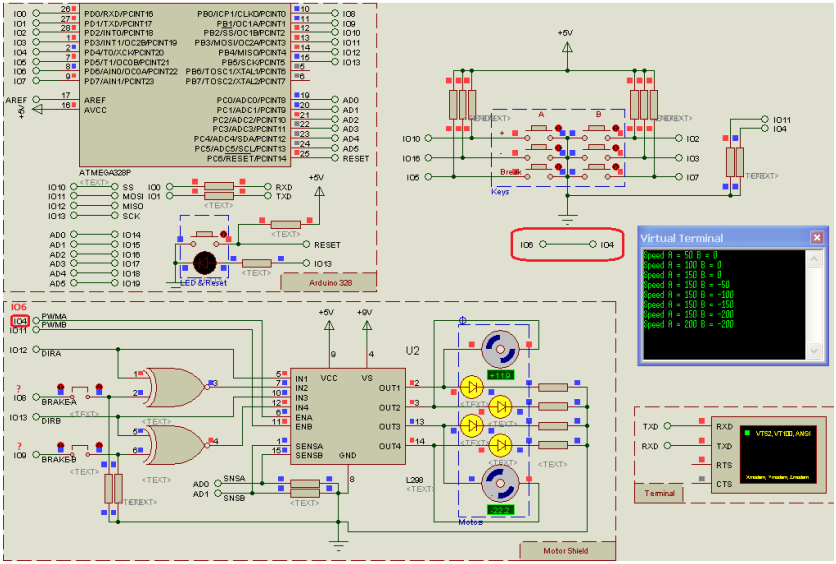


Рисунок 5.9 - Motor Shield – функціональна схема моделі в Proteus

IO10,16,5 – входи групи A

IO2,3,7 – входи групи B

IO~6,12,8– виходи керування групи A (PWMA, DIRA, BREAK-A)

IO~11,13,9 – виходи керування групи B (PWMB, DIRB, BREAK-B)

Таблиця 5.2 - Логіка керування двигунами

BRAKE-A(B)		**DIRA(B)**	**PWMA(B)**	**Motor-A(B)**
0		x	x	Стоп
1		x	0	Стоп
1		0	1..255	За часовою стрілкою
1		1	255..1	Проти часової стрілки

Для скорочення числа використовуваних ліній керування сигнали BREAK - A, BREAK - B відключаються.

Додаток 3. Таблиця кодування символів ASCII (American Standard Code for Information Interchange)

Таблиця. 5.3 Коди символів ASCII

REPRESENTATION AVEC CHIFFRES BINAIRES				b_7	0	0	1	1	1	1
				b_6	1	1	0	0	1	1
				b_5	0	1	0	1	0	1
b_4	b_3	b_2	b_1	POSITION EN HEX	2	3	4	5	6	7
0	0	0	0	0	SP	0	@	P	`	p
0	0	0	1	1	!	1	A	Q	a	q
0	0	1	0	2	"	2	B	R	b	r
0	0	1	1	3	#	3	C	S	c	s
0	1	0	0	4	¤	4	D	T	d	t
0	1	0	1	5	%	5	E	U	e	u
0	1	1	0	6	&	6	F	V	f	v
0	1	1	1	7	'	7	G	W	g	w

REPRESENTATION AVEC CHIFFRES BINAIRES				b_7	0	0	1	1	1	1
				b_6	1	1	0	0	1	1
				b_5	0	1	0	1	0	1
b_4	b_3	b_2	b_1	POSITION EN HEX	2	3	4	5	6	7
1	0	0	0	8	(	8	H	X	h	x
1	0	0	1	9	)	9	I	Y	i	y
1	0	1	0	A	*	:	J	Z	j	z
1	0	1	1	B	+	;	K	[	k	{
1	1	0	0	C	,	<	L	\	l	¦
1	1	0	1	D	−	=	M	]	m	}
1	1	1	0	E	.	>	N	^	n	¯
1	1	1	1	F	/	?	O	_	o	DEL

Додаток 4. Приклад програмування послідовного асинхронного обміну

Опис ліній МК

```
int Button[] = {7,6,5,4,3,2}, N=6;
// Список  входів для N датчиків
int LEDs[] = {8,9,10,11,12,13};
// Список виходів для індикації
int buttonState,ledState,i;
// коди станів датчиків і індикаторів
```

Налаштування МК

```
void setup() {
  Serial.begin(9600);  // швидкість 9600
 for(i=0;i<N;i++){
    pinMode(Button[i], INPUT); //входи
    digitalWrite(Button[i],HIGH); // X=11..1
  }
}
```

Основний цикл програми

```
void loop() {
        // тут опитування станів

        // передача коду стану датчиків

        // тут приймання та виведення на вихідні лінії

 delay(500); // затримка циклу опитування 500 мс
}
```

Опитування стану датчиків зі списку і передача байта стану

```
buttonState = 0;          // код станів входів      X

 for(i=0;i<N;i++){        // цикл зчитування входів x5..0
  buttonState <<= 1;              // зсув вліво
  buttonState |=  digitalRead(pushButton[i]) & 0x01;
  // додавання сигналу з чергового входу
 }

 Serial.write(buttonState);     //передача X
```

Приймання коду і виведення на виходи зі списку

```
while(Serial.available()>1) {
  ledState = Serial.read();         // read byte
  for(i=0;i<N;i++){                 // write the output pin
   if((ledState & 0x01)==1) {digitalWrite(LEDs[i],HIGH);}
     else {digitalWrite(LEDs[i],LOW);}
   ledState >>= 1;
  }
 }
```

Додаток 5. Bluetooth модуль HC-05(06)

Bluetooth модуль виконує функцію безпровідного подовження дротяного з'єднання UART.

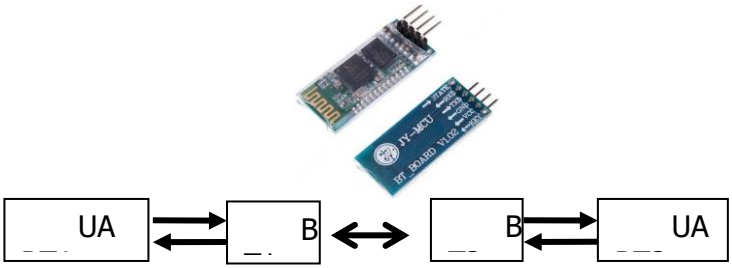
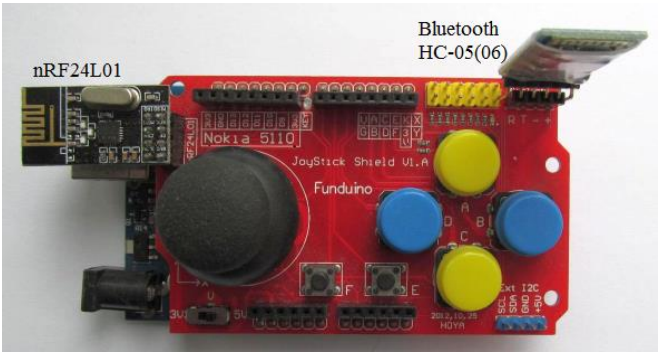
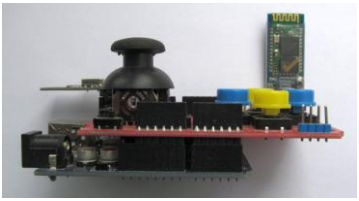


Рисунок 5.10 - Bluetooth модуль HC-05(06)

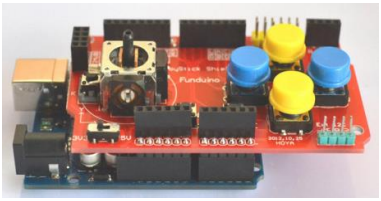
Додаток 6. Joystick Shield v1.A з модулем HC-05



а)



б)



в)

Рисунок 5.11 - Joystick Shield v1.A і Arduino UNO

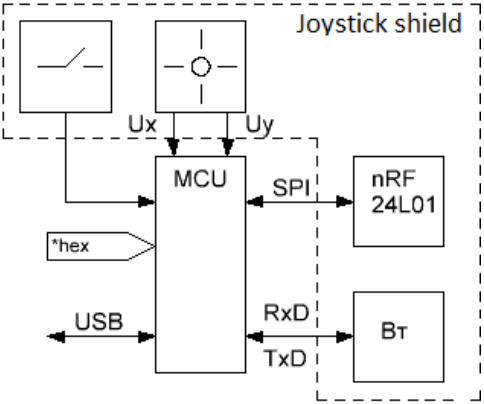


Рисунок 5.12 - Спрощена схема Joystick Shield v1.A

Arduino	A0	A1	D2	D3	D4	D5	D6	D7	D8
Shield	Ux	Uy	A	B	C	D	E	F	K

Рисунок 5.13 - Схема підключення елементів JoyStick Shield V1.A до Arduino

Натиснення кнопки відповідає логічному "0" на вказаному цифровому вході. Кнопка К вбудована в джойстик і натискається його ручкою.

Лабораторна робота 6. Розробка додатків IoT з використанням інтерфейсу SPI

Мета роботи: дослідження можливостей і методики використання інтерфейсу SPI в AVR-мікроконтролерах та навиків програмування синхронного обміну IoT додатків.

Програма роботи

1. Ознайомитися з принципами керування обміном по SPI.
2. Дослідити процес керування однорозрядним індикатором в Proteus. За допомогою віртуального осцилографа і монітора SPI отримати тимчасові діаграми сигналів та потік даних.
3. Реалізувати виведення на 4-розрядний 7-сегментний індикатор в складі Multifunctional Shield варіанту 4-розрядної комбінації цифр (символів) з використанням динамічної індикації.
4. Реалізувати виведення на 4-розрядний 7-сегментний індикатор з контролером TM1637 варіанту 4-розрядної комбінації цифр (символів) з використанням бібліотеки.
5. Реалізувати виведення на 8-розрядний індикатор з контролером TM1638 варіанту 8-розрядної комбінації цифр (символів) з використанням бібліотеки.
6. Реалізувати формування ступеневої напруги за допомогою цифрового потенціометра.
7. Підготувати звіт про виконану роботу.

Теоретичний матеріал

Послідовний периферійний інтерфейс SPI (Serial Peripheral Interface) реалізований в більшості сучасних AVR-мікроконтролерах. Він являє собою чотирьох провідний інтерфейс, що дозволяє реалізувати синхронний повно дуплексний обмін даними. До переваг інтерфейсу відносять високу швидкість передачі і можливість виділення окремої лінії синхронізації.

Спектр областей застосування SPI досить широкий:

- здійснення обміну даними між мікроконтролером та різними периферійними пристроями (цифрові потенціометри, ЦАП, АЦП, Flash- ПЗУ та ін.);

- здійснення обміну між декількома AVR-мікроконтролерами (використання SPI як високошвидкісного каналу зв'язку);

- програмування мікроконтролера (так званий режим послідовного програмування).

При обміні даними по інтерфейсу SPI AVR-мікроконтролер може працювати як провідний (режим “Master”) або як ведений (режим «Slave»). Провідна роль Master-пристрою полягає в тому, що він керує темпом обміну, видаючи імпульси синхронізації, та є ініціатором обміну. При цьому користувач може задавати швидкість передачі і формат передачі (від молодшого розряду до старшого або навпаки).

Простота інтерфейсу дозволяє використовувати в якості Slave-пристрою одну мікросхему зсувного регістру.

Додатковою можливістю підсистеми SPI є «пробудження» мікроконтролера з режиму Idle (режим холостого ходу, при переході в який ЦПУ зупиняється). Сигналом до виходу з режиму Idle є надходження даних.

При програмуванні SPI вирішуються три основні завдання:

- ініціалізація SPI (задання режимів роботи);

- організація обміну з програмним керуванням станом прапорів;

- організація обміну даними з використанням переривань.

Додаткова інформація для підготовки до виконання роботи дана в [1].

Розглянемо механізми вирішення цих завдань.

Операції і функції для програмного SPI

SPI може бути реалізований програмним керуванням лініями MOSI, SCK, SS з необхідними тимчасовими затримками, читанням стану лінії MISO та операціями зсуву змінних для послідовно-паралельного перетворення:

• shiftOut (dataPin, clockPin, LSBFIRST, data);
• digitalWrite (pin, data) - цифровий вихід data на pin;
• digitalRead (pin) - читання стану лінії pin;
• delayMicroseconds (mcs) - задає тривалість такту (швидкість);
• << (bitshift left) - побітовий зсув числа вліво;
• >> (bitshift right) - побітовий зсув числа вправо;

• bitRead (x, n) - читання значення біта n змінної x;
• bitWrite (x, n, b) - запис значення b в біт n змінної x;
• bitSet (x, n) - запис значення 1 в біт n змінної x;
• bitClear (x, n) - запис значення 0 в біт n змінної x.

Бібліотека SPI дуплексного обміну

• SPI.begin () - ініціалізація (SCK = MOSI = 0, SS = 1);
• SPI.end () – відключення;
• SPI.setBitOrder (MSBFIRST) - порядок проходження біт (LSBFIRST чи MSBFIRST);
• SPI.setClockDivider (2) - швидкість обміну (дільник 2,4,8,16,32,64,128 для Fck);
• SPI.setDataMode (mode) - режим (SPI_MODE0..3);
• SPI.transfer (data) – обмін.

Задача 1. Побудувати і налагодити програму виведення варіанту послідовності цифр на 1-розрядний індикатор. Зафіксувати цикли обміну по SPI з використанням Digital Oscilloscope і SPI Debug. Визначити формат та швидкість обміну.

Керування 1-розрядним 7- сегментним індикатором

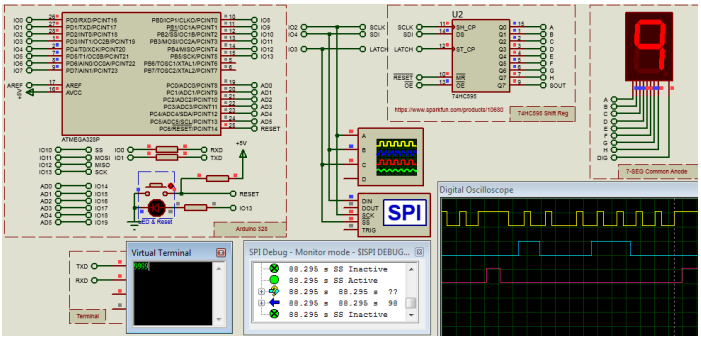


Рисунок 6.1 - Індикатор з спільним анодом, передача старшими бітами вперед

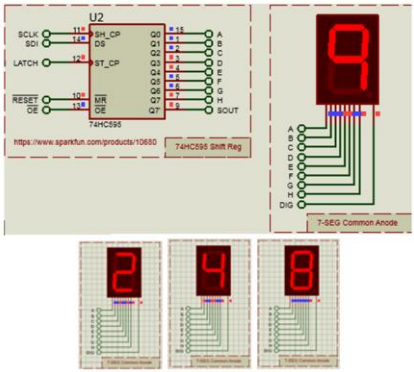
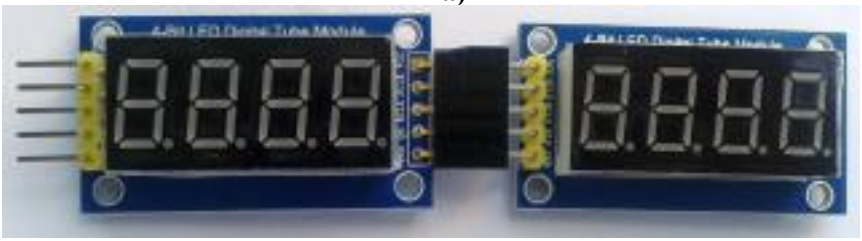


Рисунок 6.2 - Аналіз циклу обміну з використанням Digital Oscilloscope (розгортка 20 мкс) і SPI Debug

Керування 4- розрядним індикатором з використанням програмної реалізації SPI

Модуль для 4-розрядного індикатора - призначений для вбудованих додатків та IoT (рисунок 6.3).

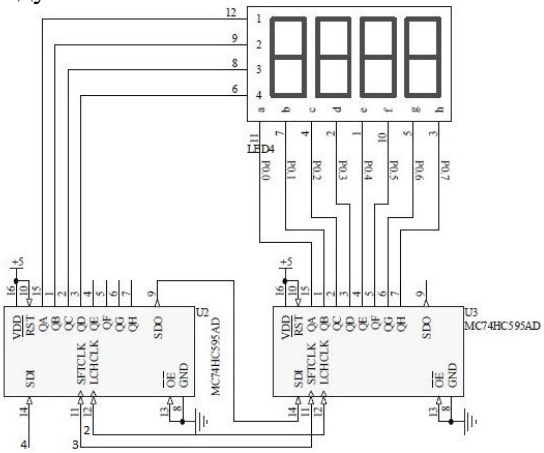
а)



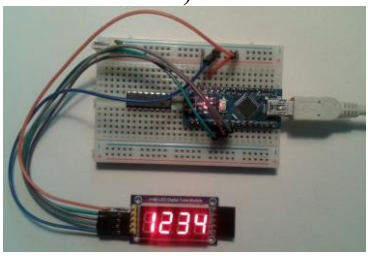
б)

Рисунок 6.3 - 4-Bit LED Digital Tube Module (а) і послідовне з'єднання модулів (б) 4 Bits 0.36" Common Anode LED Display Board Digital Tube Display Module (2x74HC595)

На рисунку 6.4 представлена функціональна схема та приклад підключення модуля.



а)



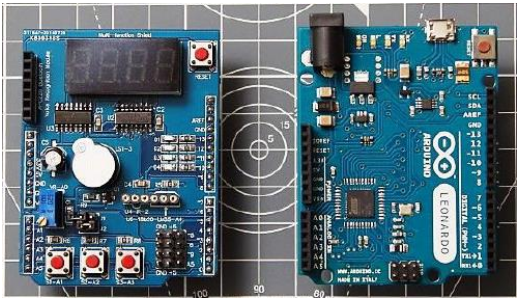
б)

Рисунок 6.4 – Функціональна схема (а) та загальний вигляд модуля (б)

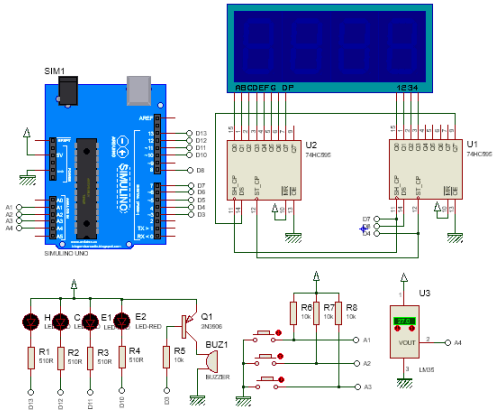
Приклад підключення ліній керування індикатором:
LATCH 3 //pin 3 защіпка;
CLOCK 2 //pin 2 синхронізація;
DATA 4 //pin 4 дані.

Задача 2. Побудувати та налагодити програму введення варіанту 4-розрядної комбінації цифр на 4- розрядний модуль індикатора чи мультифункціональний модуль з використанням програмного SPI.

Мультифункціональний модуль з 4-розрядним індикатором (рисунок 6.5)



а)



б)

Рисунок 6.5 – Вигляд (а) і схема MultiFunctional Shield (б)

Лінії керування індикатором:
LATCH_PIN 4
CLK_PIN 7
DATA_PIN 8

Приклади застосування:

1. http://publicatorbar.ru/2017/12/21/arduino-multi-function-shield/
2. https://www.cohesivecomputing.co.uk/hackatronics/arduino-multi-function-shield/part-1/

3. hackatronics-arduino-multi-function-shield.

Задача 3. Побудувати та налагодити програму виведення варіанту 4- розрядної комбінації цифр на 4- розрядний індикатор з використанням бібліотечних функцій.

Керування 4- розрядним індикатором з контролером TM1637

Контролер TM1637 містить вбудовані зсувні регістри та інтерфейс SPI. Для керування модулем, що підтримується контролером TM1637, розроблена бібліотека у вільному доступі (рисунок 6.6).

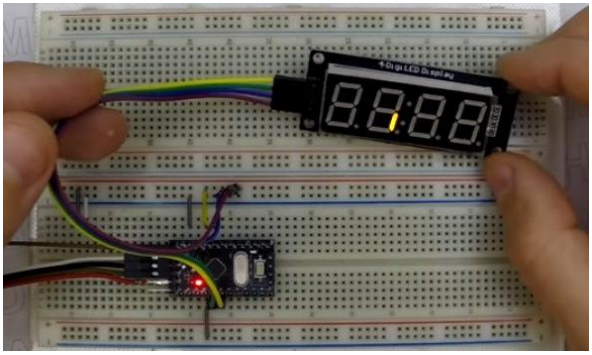


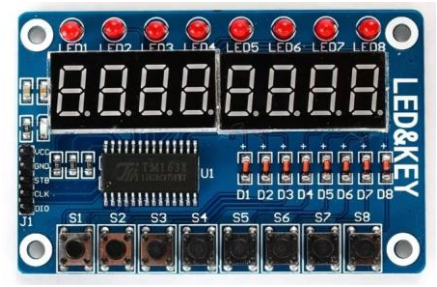
Рисунок 6.6 – Вигляд контролера TM 1637 з керуючим індикатором

Задача 4. Побудувати та налагодити програму виведення варіанту 4- розрядної комбінації цифр на 4- розрядний індикатор з використанням контролера TM1637 на основі бібліотечних функцій.

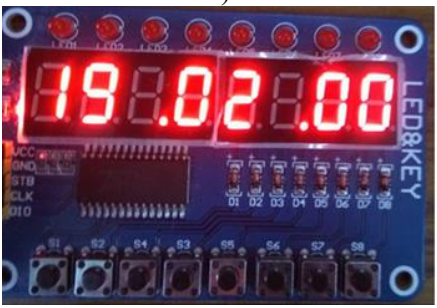
Керування 8- розрядним індикатором з контролером TM1638

Індикатор входить до складу модуля пульта оператора з набором кнопок і керуючих виходів, пов'язаних зі світлодіодами (рисунок 6.7а). Для керування модулем за допомогою контролера

TM1638 з SPI інтерфейсом розроблена вільно поширювана бібліотека. Приклад її застосування зображено на рисунку 6.7б.



а)



б)

Рисунок 6.7 – Модуль (а) і приклад (б) застосування

Задача 5. Побудувати і налагодити програму виведення індивідуальної 8- розрядної комбінації цифр на 8- розрядний індикатор з використанням контролера TM1638 на основі бібліотечних функцій. Виконати налагодження програми з використанням навчального стенду.

Задача 6. На одному з каналів потенціометра сформувати ступеневу напругу для індивідуальної послідовності значень. Провести аналіз циклів обміну з використанням Digital Oscilloscope і SPI Debug. Отримати осцилограми вихідної напруги.

Керування цифровим потенціометром AD5206 (рисунок 6.8)

Індивідуальним варіантом послідовності значень є послідовність номерів векторів переміщень для опису контуру в роботі 2. Приклад східчастої напруги наведено на рисунку. 6.9. Для осцилограм розгортка по горизонталі - 100 мс, а по вертикалі - 0.5 В.

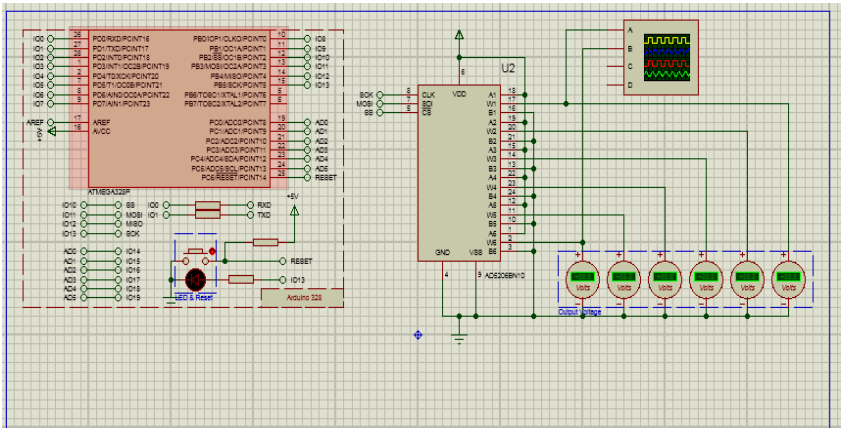
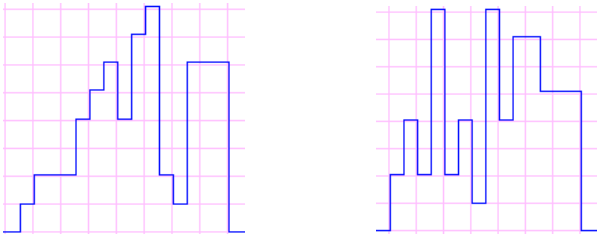


Рисунок 6.8 - Модель AD5206 в середовищі Proteus



а) б)

Рисунок 6.9 - Вигляд ступінчатої напруги для послідовностей {0,1,2,2,2,4,5,6,4,7,8,2,1,6,6,6,0} (а) і {0,2,4,2,8,2,4,1,8,4,7,7,5,5,5,0} (б) з множником 26

Зміст звіту

1. Схеми апаратних засобів для вирішення задач.
2. Тексти програм задач.
3. Результати моделювання.
4. Результати виконання експериментів.
5. Висновки по роботі.

Контрольні запитання

1. Назвіть режими обміну, реалізовані SPI.
2. Які переваги програмної реалізації SPI?
3. Перерахуйте відомі функції обміну по SPI.
4. Який механізм реалізації обміну даними з використанням апаратного SPI?
5. Якими програмними лініями може бути реалізований SPI?
6. Як відбувається керування 8- розрядним індикатором з контролером TM1638?
7. Поясніть принцип керування 4- розрядним індикатором з використанням програмної реалізації SPI.
8. Як відбувається керування цифровим потенціометром?
9. Що таке ступенева напруга? Як її сформувати?
10. Використовуючи осцилограф, поясніть особливості реалізації обміну даними.

Література

1. Опис навчальних стендів.
2. Соммер У. Программирование микроконтроллерных плат Arduino/Freeduino.- СПб.: БХВ – Петербург, 2012.- 256с.
3. Brian W. Evans Блокнот програмиста. 2007.
4. Среда разработки Arduino. [Електронний ресурс]. – Режим доступу: http://www.arduino.cc/en/Main/Software
5. Simulator for Arduino. [Електронний ресурс]. – Режим доступу: http://virtronics.com.au/Data/SetupFree.zip.
6. Программирование Ардуино. [Електронний ресурс]. – Режим доступу: http://arduino.ru/Reference

7. PROTEUS VSM. Среда виртуального моделирования. - PROTEUS-d.pdf [Електронний ресурс]. Режим доступу: http://proteus123.narod.ru/

8. JoyStick Shield V1[Електронний ресурс]. – Режим доступу: https://www.elecfreaks.com/wiki/ index.php?title=Joystick_Shield

Додаток 1. Керування 1- розрядним 7- сегментним індикатором з використанням зсувного HC595 і soft-SPI для виведення послідовності символів.

```
#define LATCH 3 //pin 3 защіпка
#define CLOCK 2 //pin 2 синхронізація
#define DATA 4 //pin 4 дані
//таблиця вибору сегментів abcdefgh для відображення цифр
// a
// f b
// g
// e c
// d h
//лог. 0 -сегмент  світиться, 1 - не світиться
byte cathode[13] = {0b00000011, // 0- вимкнені gh
0b10011111, // 1- світяться bc
0b00100101, // 2
0b00001101, // 3
0b10011001, // 4
0b01001001, // 5
0b01000001, // 6
0b00011111, // 7
0b00000001, // 8
0b00001001, // 9
0b11111110, // dot
0b11111111, // blank
0b11111101 // -
  };
 byte in[15]= {0,1,2,2,2,4,5,6,4,7,8,2,1,6,6,6,0}; // індивідуальна послідовність цифр
int Tw=250; //період зміни цифр
void setup(void)
```

```
{
 pinMode(LATCH, OUTPUT);
 pinMode(CLOCK, OUTPUT);
 pinMode(DATA, OUTPUT);
}
void loop(void)
{ // послідовне виведення цифр
for(int i=0;i<15;i++){
   digitalWrite(LATCH, LOW);
   shiftOut(DATA, CLOCK, LSBFIRST, cathode[in[i]]); // черговий символ
   digitalWrite(LATCH, HIGH);
   delay(Tw);
}
}
```

Додаток 2. Керування 4- розрядним 7- сегментним індикатором з використанням soft-SPI для динамічної індикації «1234»

```
#define LATCH 3 //pin 3 защіпка
#define CLOCK 2 //pin 2 синхронізація
#define DATA 4 //pin 4 дані
byte anode[ ] = { 0b10000000, //вибір розряду 1 справа (одиниці)
0b01000000, // вибір розряду 2 справа (десятки)
0b00100000, // вибір розряду 3 справа (сотні)
0b00010000, // вибір розряду 4 справа (тисячі)
 };
//таблиця вибору сегментів abcdefgh для відображення цифр
// f   b
//   g
// e   c
//    d   h
//лог. 0 -сегмент світиться, 1 - не світиться
byte cathode[13] = {0b00000011, // “0”- вимкнені gh
0b10011111, // “1”- світяться bc
0b00100101, // “2“
```

```
    0b00001101, // “3“
    0b10011001, // “4“
    0b01001001, // “5“
    0b01000001, // “6“
    0b00011111, // “7“
    0b00000001, // “8“
    0b00001001, // “9“
    0b11111110, // dot
    0b11111111, // blank
    0b11111101 // -
      };
     int Tw=5; // час світіння розрядів

     void setup(void)
     {
      pinMode(LATCH, OUTPUT);
     pinMode(CLOCK, OUTPUT)
     pinMode(DATA, OUTPUT);
     }

     void loop(void)
     { // послідовне виведення цифр для динамічної індикації «1234»
       for(int i=0;i<17;i++){
      digitalWrite(LATCH, LOW);
      shiftOut(DATA, CLOCK, LSBFIRST, 0b10011111); // next symbol
      shiftOut(DATA, CLOCK, LSBFIRST,  0b10000000); // next symbol
      digitalWrite(LATCH, HIGH);
      delay(Tw);
      digitalWrite(LATCH, LOW);
      shiftOut(DATA, CLOCK, LSBFIRST, cathode[2]); // "2"
      shiftOut(DATA, CLOCK, LSBFIRST, 0b01000000); // десятки
      digitalWrite(LATCH, HIGH);
      delay(Tw);
      digitalWrite(LATCH, LOW);
      shiftOut(DATA, CLOCK, LSBFIRST, cathode[3]); // "3"
```

```
shiftOut(DATA, CLOCK, LSBFIRST, anode[2]); // сотні
digitalWrite(LATCH, HIGH);
delay(Tw);

digitalWrite(LATCH, LOW);
shiftOut(DATA, CLOCK, LSBFIRST, cathode[4]); // "4"
shiftOut(DATA, CLOCK, LSBFIRST, anode[3]); // тисячі
digitalWrite(LATCH, HIGH);

delay(Tw);
if(Tw>=5) Tw=Tw/1.25; // поступове прискорення
  } }
```

Додаток 3. Керування 6- канальним цифровим потенціометром з використанням апаратного SPI

Приклад програми ілюструє роботу багатоканального функціонального генератора на виходах каналів потенціометра.

```
// inslude the SPI library:
#include <SPI.h>
// set pin 10 as the slave select for the digital pot:
const int slaveSelectPin = 10;
void setup() {
 // set the slaveSelectPin as an output:
 pinMode (slaveSelectPin, OUTPUT);
 // initialize SPI:
 SPI.setDataMode(SPI_MODE0);  //SPI_MODE0, SPI_MODE1,
SPI_MODE2, or
SPI_MODE3;
 SPI.setBitOrder(MSBFIRST); // LSBFIRST or MSBFIRST
 SPI.begin();
}
void loop() {
 // change the resistance on this channel from min to max:
 for (int level = 0; level < 255; level++) {
 digitalPotWrite(0,level);
 digitalPotWrite(1,(64-level)&0xFF);
 digitalPotWrite(2,(128-level)&0xFF);
 digitalPotWrite(3,(160-level)&0xFF);
```

```
 digitalPotWrite(4,(192-level)&0xFF);
 digitalPotWrite(5,(255-level)&0xFF);
 }
 delay(10);
}
void digitalPotWrite(int channel,int value) {
 // take the SS pin low to select the chip:
 digitalWrite(slaveSelectPin,LOW);
 // send in the command and value via SPI:
 SPI.transfer(0x10+channel);
 SPI.transfer(value);
 // take the SS pin high to de-select the chip:
 digitalWrite(slaveSelectPin,HIGH);}
```

Лабораторна робота 7. Аналоговий інтерфейс додатків IoT

Мета роботи: отримання навичок роботи з аналого-цифровим перетворювачем мікроконтролера і написання програм обробки оцифрованих аналогових сигналів.

Програма роботи

1. Ознайомитися з будовою, принципом дії і призначенням АЦП та розглянути методи перетворення аналогового сигналу у цифровий.
2. Дослідити реалізацією роботи АЦП на Arduino.
3. Дослідити основні характеристики та принцип роботи аналогового датчика температури LM35.
4. Реалізувати визначення температури навколишнього середовища за допомогою аналогового датчика LM35 перетворивши за допомогою АЦП напругу на виході датчика в градуси Цельсія.
5. Дослідити основні характеристики та принцип роботи LCD 1602 Keypad Shield.
6. Розробити програму для опитування клавіатури і реалізувати виведення на дисплей направлення (вверх, вниз, вправо, вліво) у відповідності із нетисненою кнопкою.
7. .Дослідити основні характеристики та принцип роботи Joystick Sield.
8. Розробити та реалізувати програму виведення поточного значення та кута нахилу Joystick Sield.
9. Підготувати звіт про виконану роботу.

Теоретичний матеріал

Аналоговий сигнал - це сигнал, величина якого у будь-який момент часу може приймати будь-яке значення (рисунок 7.1).

Цифровий сигнал - це сигнал, величина якого в певні моменти часу може приймати лише певні значення.

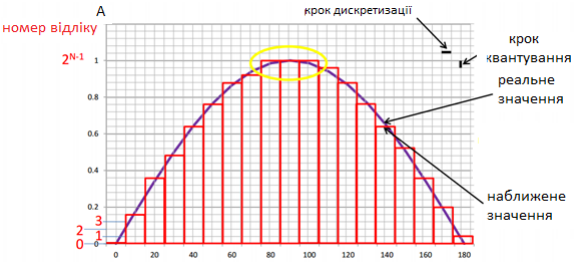


Рисунок 7.1 – Представлення аналогового сигналу

Аналого- цифровий перетворювач

Аналого- цифровий перетворювач (АЦП, англ. Analog-to-digital converter, ADC) - пристрій, що перетворює вхідний аналоговий сигнал в дискретний код (цифровий сигнал). Зворотне перетворення здійснюється за допомогою ЦАП (цифро-аналогового перетворювача, DAC)

Розрядність АЦП – характеристика АЦП, що визначає максимальну кількість дискретних значень (біт), які може видати АЦП.

Розрядність АЦП визначається його здатністю, що характеризується мінімальною зміною величини вхідного аналогового сигналу, який може бути зафіксованим даним АЦП.

АЦП перетворює сигнал (напругу), який знаходиться в діапазоні вимірюваних сигналів. Нижня і верхня межа цього діапазону визначаються напругою, поданою на відповідні виходи.

При діапазоні вхідної напруги від 0 до 5В і використанні 10-бітного АЦП ми маємо наступний крок квантування АЦП:

$$\Delta = \frac{U_{\max} - U_{\min}}{ADC_{\max}} = \frac{5B - 0B}{2^{10}} = \frac{5B}{1024} = 4,9мВ$$

Таким чином. АЦП, розрізняє сигнали, які відрізняються на 4,9 мВ. При збільшенні сигналу на 4,9 мВ - результат перетворення збільшиться на 1. Для обчислення напруги використовується наступний вираз:

U = (5/1024) * Значення АЦП

Наприклад, якщо значення АЦП= 100, то: U= (5/1024) *100 = 0,49V.

Частота дискретизації (перетворення) – це характеристика АЦП, що визначає кількість перетворень, які виконуються АЦП за одиницю часу (відлік/ секунда).

Аналоговий сигнал є безперервною функцією часу, в АЦП він перетвориться у послідовність цифрових значень. Отже, необхідно визначити частоту вибірки цифрових значень з аналогового сигналу. Частота, з якою проводяться цифрові значення, отримала назву частота дискретизації АЦП.

Оскільки, реальний АЦП не може зробити аналого-цифрове перетворення миттєво, вхідне аналогове значення повинно зберігатися сталим від початку до кінця процесу перетворення (цей часовий інтервал називається час трансформації). Це завдання вирішується за допомогою спеціальної схеми на вході АЦП-пристрій вибірки-зберігання. Пристрій, зберігає вхідну напругу в конденсаторі, який з'єднаний з входом через аналоговий ключ: коли ключ замкнений, відбувається вибірка вхідного сигналу сигнал (конденсатор заряджається до вхідної напруги); при розмиканні-зберіганні. У мікроконтролерах, частота дискретизації може бути налаштована програмно. Але чим вища частота (частіша вибірка) тим більша похибка перетворення (менша точність).

Характеристики АЦП Arduino:

– діапазон вхідних значень: від 0 до 5 вольт (напруга живлення);
– розрядність АЦП 10 біт 1024 рівнів квантування;
– роздільна здатність АЦП по напрузі (крок квантування): 4,88 мВ;
– коефіцієнт вибірки - 1 МГц.

Рисунок 7.2 - Аналогові входи Arduino

Зчитування значення з АЦП (аналогового входу)

Для реалізації роботи з АЦП на Arduino використовується функція **analogRead (pin).** Функція зчитує значення із зазначеного аналогового входу.

Більшість плат Arduino мають 6 каналів 10-бітного АЦП (8 каналів у плати Mini і Nano та 16 у Mega). Напруга подана на аналоговий вхід, зазвичай від 0 до 5 вольт перетворюється в значення від 0 до 1023 з кроком 0.0049 Вольт. Щоб зчитати значення з аналогового входу потрібно приблизно 112 мкс (0.000112 сек), звідси маємо максимальну швидкість зчитування - приблизно 9000 разів за секунду.

Діапазон напруги і крок квантування може бути змінений функцією **analogReference ().**

Номера аналогового порту - 0 .. 5, або A0 .. A5 (рисунок 7.2).

Якщо аналоговий вхід не підключений, то значення повертаються функцією **analogRead ()** і можуть приймати випадкові значення.

Програмна реалізація роботи з АЦП :

```
int sensorValue = 0;
void setup() {
…
}
void loop() {
…
```

```
sensorValue = analogRead(A0);
… //
}
```

Задача 1. Написати програму для опитування аналогового датчика температури LM35 та перетворити аналоговий сигнал датчика в градуси Цельсія у відповідності зі специфікацією датчика. Реалізувати симуляцію за допомогою програми Proteus та Simulator Arduino, завантажте її в пристрій і перевірте роботу.

Аналоговий датчик температури LM35Z

Датчики LM35/LM335/TMP36 мають просту схему підключення (рисунок. 7.3), досить хороші параметри (табл. 1) і невелику вартість.

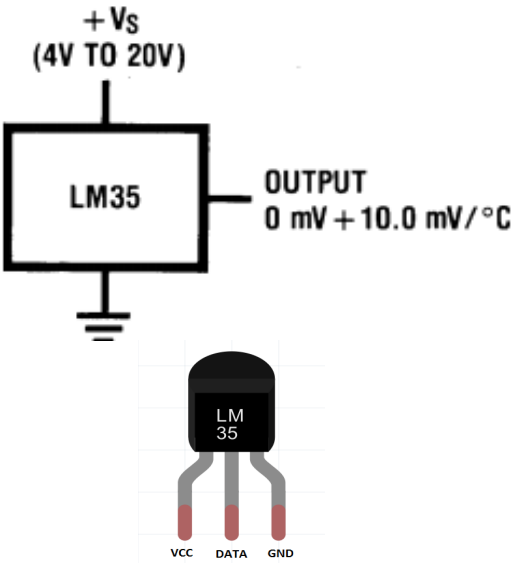


Рисунок 7.3 – Аналоговий датчик температури

Датчик LM35 / TMP36 має лінійну залежність вихідної напруги від температури в діапазоні від $+2^0$C до 150^0C. Оскільки датчик аналоговий, то на виході ми маємо напругу від 0.02В до 1.5В. Для визначення температури необхідно перетворити

аналоговий сигнал датчика в градуси Цельсія. Відповідно до специфікації датчика (таблиця 7.1), кожний градус вимірюваної температури, відповідає 10 мілівольтам напруги на виході.

Таблиця 7.1 - Основні параметри LM35Z

Вихід:	10
Одиниці вимірювання	mV/°C
Похибка,±°C	0.5
V_{CC},В	від 4 до 40
I_{CC}, мА	10
T_A,°C	від -55 до 150
Корпус	SOIC-8 TO-46-3 TO-92

Моделі LM35 / TMP36 містяться в бібліотеці Proteus і використовуються в моделях пристроїв - термометрів (рисунок 7.4). Схема з аналоговим датчиком температури LM35 реалізована у вигляді фізичного стенду з виведенням значень на LCD WH0802.

Модель пристрою для вимірювання освітленості терморезистором (LDR) і температури (TMP36) на рисунку 7.5 може бути використана в якості віртуального стенду для налагодження широкого кола завдань (регулювання температури, керування освітленням, спостереження за зміною температури/ освітленості із записом на SD карту і ін.).

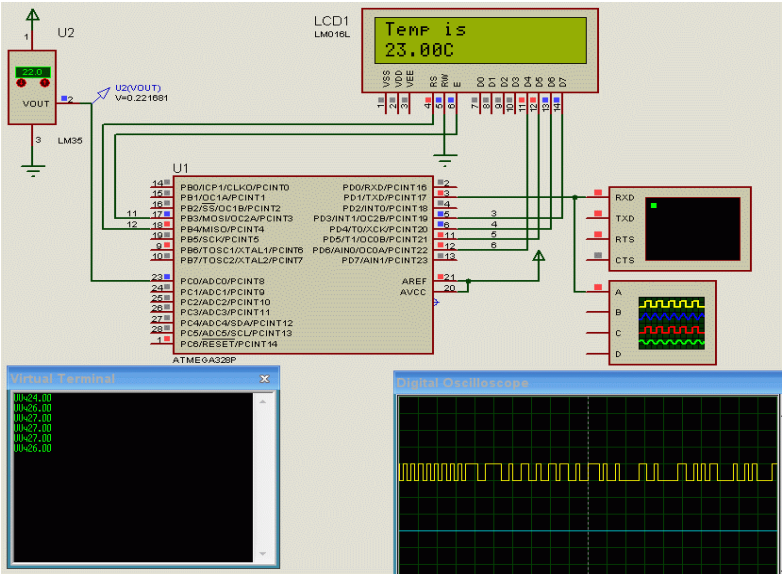


Рисунок 7.4 – Моделювання опитування аналогового датчика температури LM35Z

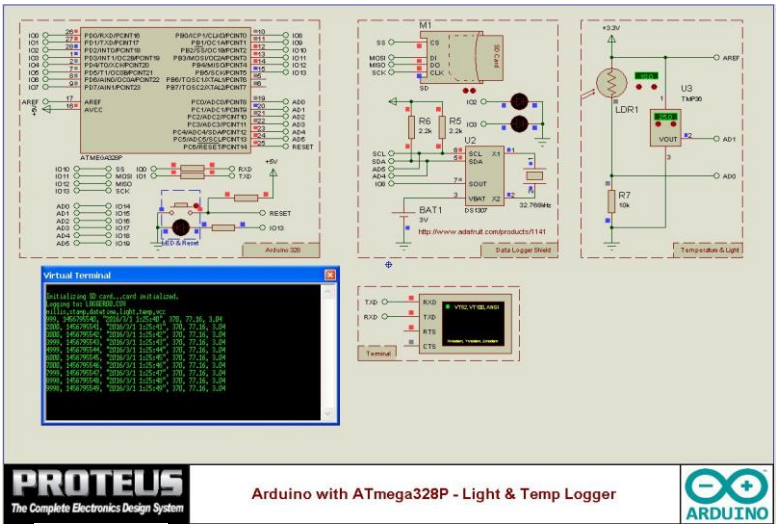


Рисунок 7.5 – Модель пристрою для вимірювання освітлення терморезистором (LDR) і температури (TMP36)

Датчик LM35 / TMP36 має лінійну залежність вихідної напруги від температури в діапазоні від $+2^0$ С до 150^0С, якому відповідав би діапазон вихідної напруги від 0.02В до 1.5В. Для визначення температури необхідно перетворити аналоговий сигнал датчика в градуси Цельсія. Оскільки, кожен градус вимірюваної температури, відповідає 10 мілівольт напруги на виході, тоді для перетворення значення аналогового сигналу в градуси Цельсія, нам необхідно виконати наступні дії:

– зчитати значення з АЦП за допомогою **analogRead** і обчислити відповідну напругу на вході АЦП:

*вольти = (значення АЦП / 1023) * 5;*

де, **1023** — максимальне значення, яке може повернути нам 10-бітний АЦП, вбудований в AVR мікроконтролер;

5 - опорна напруга АЦП, вольт;

– -перетворюємо напругу в градуси Цельсія:

градус = вольти/ 0.01;

– обчислюємо значення температури:

*Temp.= значення АЦП*500/1023 ≈ значення АЦП / 2*

Наприклад: якщо значення АЦП = 55, то Temp. ≈ 26.5^0С (точне значення - 26.88 ^{0}С).

Цикл програми опитування датчика виглядає наступним чином:

```
void loop()
        {
          int val = analogRead(A0); // зчитування значення з АЦП
        float temp =  val/2; // конвертування аналогового сигналу в градуси
          Serial.println(temp);  // виведення значення температури на термінал
          delay(250);
  }
```

Завдання 2. Ознайомитися з платою розширення LCD 1602 Keypad Shield. Написати програму для опитування клавіатури і виведення на дисплей напрямків (вгору, вниз, вліво, вправо) відповідно до натиснутої кнопки. Провести симуляцію за

допомогою Proteus і Simulator Arduino, потім завантажити її в пристрій і перевірити роботу.

Опитування клавіатури в LCD Keypad Shield

LCD 1602 Keypad Shield - це плата розширення на якій розміщений символьний дисплей в форматі - два рядки по 16 символів в рядку (для виведення текстової інформації). Також на платі встановлено 5 кнопок (стан яких зчитуються МК) і кнопка скидання (reset) (рисунок 7.6).

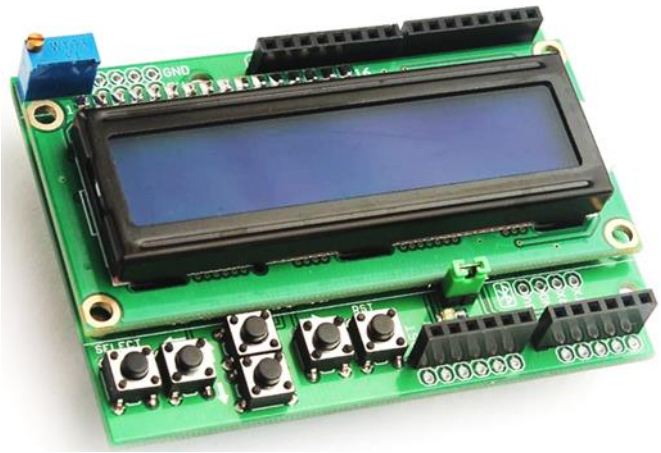


Рисунок 7.6 – Зовнішній вигляд LCD Keypad Shield

Дисплей працює в 4 бітному режимі. Контраст дисплея, налаштовується за допомогою потенціометра, який встановлений на платі (у верхньому лівому куті - синій прямокутник).

Напруга живлення LCD Keypad Shield - 5 Вольт.

На pin A0 Arduino при натисканні однієї з кнопок (рисунок 7.7) подається напруга, приблизне значення якої наведено в таблиці 7.2. Після виконання функції **analogRead(A0)** повертається код напруги на A0, порівнюючи який з граничними значеннями з таблиці 7.2, можна дізнатися яка з кнопок натиснута. Якщо не натиснута жодна з кнопок на pinA0 через підтягуючий резистор R2 подається напруга 5В.

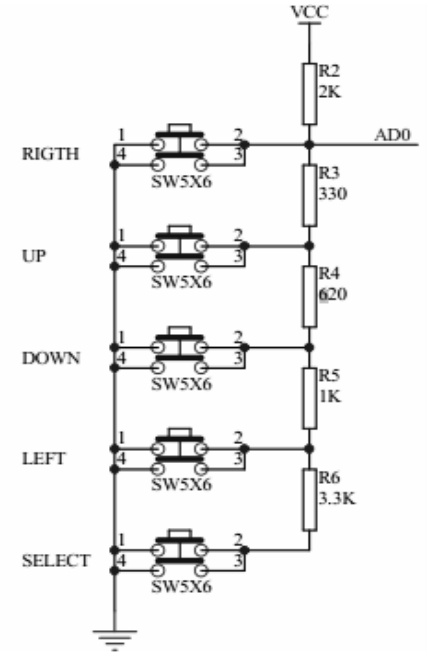


Рисунок 7.7 – Електрична підключення клавіатури

Схема являє собою аналог потенціометра з дискретними положеннями і відповідною напругою на виході (таблиця 7.2).

Таблиця 7.2 - Дані на реакцію нетисненої кнопки

Кнопка	A0, V	ADC (розр.)	ADC (експ.)	Поріг	Код стану
Select	3,62	737	741	880	4
Left	2,47	501	506	618	3
Down	1,61	327	330	420	2
Up	0,71	143	145	250	1
Right	0	0	0	72	0
None	5	1023	1023	1023	5

Зчитування і аналіз значення напруги на виході клавіатури можна реалізувати викликом функції **read_LCD_buttons(),** що

повертає код стану клавіатури в діапазоні 0...5. Приклад програми приведено нижче:

```
int read_LCD_buttons()
{
  int adc_key_in = analogRead(0);
  if (adc_key_in < 50)   return btnRIGHT;
  if (adc_key_in < 175)  return btnUP;
  if (adc_key_in < 325)  return btnDOWN;
  if (adc_key_in < 515)  return btnLEFT;
  if (adc_key_in < 820)  return btnSELECT;
  if (adc_key_in < 1024) return btnNONE;
  return btnNONE;
}
```

Дана клавіатура використовується в моделі пристрою - Cyrillic LCD (рисунок 7.8).

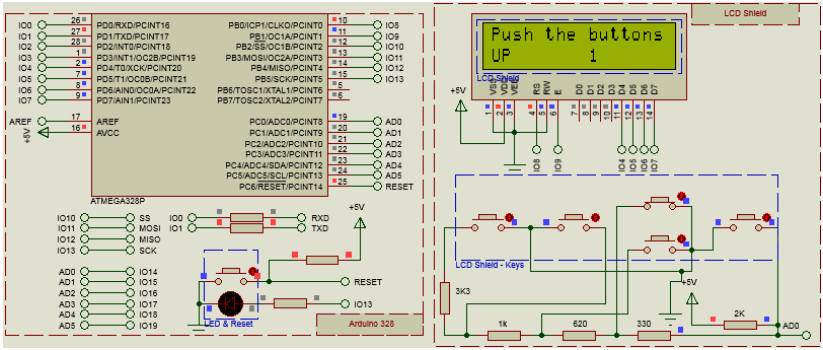


Рисунок 7.8 – Модель в Proteus (приклад -Cyrillic LCD)

Завдання 3. Вивчити будову та принцип роботи Joystick Sield. Написати програму для Joystick Sield, яка буде виводити поточний напрямок і кут нахилу пристрою. Провести симуляцію за допомогою Proteus і Simulator Arduino, потім завантажити її в пристрій і перевірити роботу.

Керування за допомогою Joystick Shield

На платі розширення встановлено чотири кнопки праворуч, одна кнопка безпосередньо на джойстику і аналоговий джойстик (рисунок 7.9). Положення ручки джойстика по осях X і Y розраховується залежно від значень напруги від двох потенціометрів, які в ньому встановлені. На них подається напруга живлення - 5V. Конструкція джойстика забезпечує повернення ручки в середнє вихідне положення.

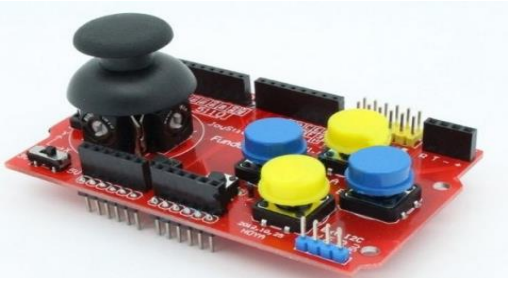


Рисунок 7.9 – Зовнішній вигляд Joystick Shield

При переміщенні джойстика по X і Y коди вихідної напруги після АЦП змінюються відносно початкового значення 512 відповідно до рисунку 7.10. Для зчитування даних з потенціометрів використовуємо функцію analogRead(), яка повертає значення в діапазоні від 0 до 1023. Для цього треба в функцію передати номера аналогових входів, до яких підключений джойстик.

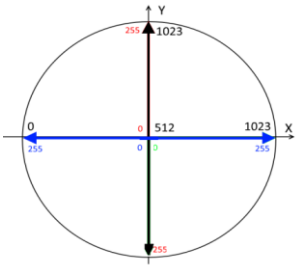


Рисунок 7.10 - Діаграма зміни стану виходів потенціометра в залежності від положення ручки джойстика

У даному прикладі виходи підключені до аналогового входу A0 для X та до аналогового входу A1 для Y.

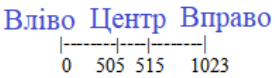
Serial.println (analogRead (0)); // показує поточне положення X координати.

Serial.println (analogRead (1)); // показує поточне положення Y координати.

Визначення поточного положення джойстика

Так як значення в кожному з напрямків буде в діапазоні від 0 до 1023, можна припустити, що центр буде знаходитися в діапазоні 511-512. Але це не зовсім так. Настільки точне поточне значення ми не отримаємо. І якщо визначити невірне значення, можна отримати спотворену інформацію про переміщення ручки джойстика, хоча вона залишається нерухомою.

Для цього можна ввести діапазон значень і будемо вважати, що будь-яке значення в цьому діапазоні буде центром:



Зазвичай визначаються константи зони нечутливості в початковому положенні:

const int X_ LOW = 505;
const int X_ HIGH = 515;
const int Y_LOW = 505;
const int Y_HIGH = 515;

Тепер можна перетворити кожну координату з діапазону від 0 до 1023 в діапазон від -1 до 1. Тоді X= -1 означає переміщення вліво, X = 0 означає відсутність переміщення, а X= 1 - переміщення вправо. Y= -1 означає переміщення вниз, Y= 0 означає відсутність переміщення, а Y= 1 - переміщення вгору.

.....
x_direction = 0;
y_direction = 0;
x_position = analogRead(PIN_ANALOG_X);
y_position = analogRead(PIN_ANALOG_Y);

```
if (x_position > X_ HIGH)  x_direction = 1;
 else
   if (x_position < X__LOW)  x_direction = -1;
if (y_position > Y__HIGH)  y_direction = 1;
 else
   if (y_position < Y_ LOW)  y_direction = -1;
```

Визнання кута нахилу джойстика

Оскільки кут повороту джойстика приблизно ± 60 градусів (діапазон 120 градусів), а АЦП видає значення y_position, x_position від 0 до 1023, то для обчислення кроку переміщення потрібно:

```
float stepSize = 120 / 1023;  // крок переміщення
```

Отримані дані від АЦП перетворюємо в кут нахилу:

```
float  yAngle = (y_position – 512) * stepSize;  // кут нахилу джойстика по осі Y
float  xAngle = (x_position – 512) * stepSize;  // Аналогічно для осі X
```

Зміст звіту

1. Схеми апаратних засобів для вирішення завдань.
2. Тексти програм для завдань.
3. Результати моделювання.
4. Результати виконання експериментів.
5. Висновки по роботі.

Контрольні запитання:

1. Що таке АЦП?.
2. Які основні характеристики АЦП?
3. Як обчислити напругу за допомогою АЦП?
4. Значення функції analogRead.
5. Скільки перетворень в секунду виконує АЦП?6
6. Чим визначається діапазон вхідних напруг АЦП?
7. Як виглядає характеристика перетворення АЦП?
8. Типові значення розрядності АЦП.

9. Чим визначається крок квантування АЦП?
10. Скільки каналів перетворення має АЦП ATmega328?

Література

1. Опис навчальних стендів.
2. Соммер У. Программирование микроконтроллерных плат Arduino/Freeduino.- СПб.: БХВ – Петербург, 2012.- 256с.
3. Brian W. Evans Блокнот програмиста. 2007.
4. Среда разработки Arduino. [Електронний ресурс]. – Режим доступу: http://www.arduino.cc/en/Main/Software
5. Simulator for Arduino. [Електронний ресурс]. – Режим доступу: http://virtronics.com.au/Data/SetupFree.zip.
6. Программирование Ардуино. [Електронний ресурс]. – Режим доступу: http://arduino.ru/Reference
7. PROTEUS VSM. Среда виртуального моделирования. - PROTEUS-d.pdf [Електронний ресурс]. Режим доступу: http://proteus123.narod.ru/
8. JoyStick Shield V1[Електронний ресурс]. – Режим доступу: https://www.elecfreaks.com/wiki/ index.php?title=Joystick_Shield

Лабораторна робота 8. Розробка додатків WoT на основі вбудованої платформи

Мета роботи: ознайомитися з методикою побудови WoT-додатків для Smart Devices на основі інтеграційного шаблону DirectConnectivity і набути навички програмування як клієнтської частини додатків на HTML, CSS і JavaScript, так і серверної частини додатків на JavaScript в середовищі Node.js.

Програма роботи

1. Ознайомитися з комп'ютером «Raspberry Pi 3 Model B», встановити ОС Debian Linux і підключити периферію.
2. Ознайомитися з програмною платформою Node.js і встановити платформу Node.js на Raspberry Pi.
3. Реалізувати директорію для WoT-додатка і каркас додатка.
4. Підключити actuator (в якості приводу застосуємо світлодіод LED) до портів GPIO.
5. Реалізувати серверну частину WoT-додатка (server.js) із застосуванням фреймворка express.
6. Реалізувати клієнтську частину WoT-додатка.
7. Апробувати роботу WoT-додатка.
8. Підготувати звіт про виконану роботу.

Теоретичний матеріал

Для інтеграції Smart Objects в Інтернет існують три різних інтеграційних шаблонів: Direct Connectivity, Gateway Based Connectivity, Cloud Based Connectivity [1].

Web Things забезпечують інтерфейс API (API Web Thing) на основі веб-сервісів з RESTful для взаємодії Smart Objects один з одним з використанням існуючих веб-стандартів. На рисунку 8.1 представлені API Web Thing [2], які можуть бути розміщені як безпосередньо на самих пристроях, так і на проміжних Web Things мережі типу шлюз або хмарний сервіс (рисунок 8.1).

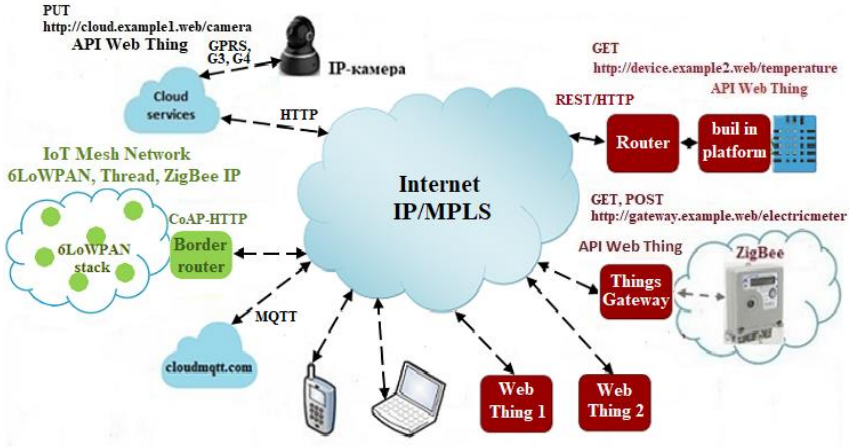


Рисунок 8.1 – Фрагменти архітектури Web of Things (WoT)

Відповідно до стандарту "Архітектура Web of Things (WoT)" [3] архітектура сервера WoT повинна мати програмний стек, в якому реалізовані основні блоки WoT: Application, WoT Scripting API, Protocol Bindings. Додаток для Web Things розробляється, як правило, на JavaScript. Додаток може бути реалізовано з використанням API-інтерфейсів Scripting або API-інтерфейсів на власній платформі.

Шаблон інтеграції Direct Connectivity, призначений для інтеграції Smart Devices or Objects в Internet на основі вбудованої або власної платформи. Реалізація WoT-додатка на основі власної платформи може бути здійснена на основі веб-сервера, який міститься на контролері вбудованого в Things або на контролері, до портів GPIO якого підключені Things.

WoT-додатки для Smart Devices, вміщені на веб-сервери, можуть бути забезпечені як для користувача інтерфейсами для взаємодії користувачів з Things через веб-браузери, так і інтерфейсами прикладного програмування Web Thing REST API або Web Thing WebSocket API для обміну даними з іншими пристроями [4].

Додаткова інформація для підготовки до виконання роботи дана в [5].

Розглянемо механізми вирішення цих завдань.

Завдання 1. Ознайомитися з комп'ютером «Raspberry Pi 3 Model B», встановити ОС Debian Linux і підключити периферію.

В якості контролера застосуємо комп'ютер «Raspberry Pi 3 Model B» на основі ОС Debian Linux version 4.19.59-v7. У «Raspberry Pi 3 Model B» вбудована підтримка Wi-Fi, Bluetooth 4.1 і забезпечений доступ до портів GPIO (інтерфейсу введення/ виведення) для прямого підключення пристроїв (наприклад, до датчиків або приводів).

Як сховище для зберігання даних і програм застосуємо карту пам'яті microSD Kingston 16 GB (Class 10). Слід зазначити, що карту microSD необхідно відформатовати як FAT32. Офіційною операційною системою для всіх моделей Raspberry Pi є Raspbian (дистрибутив Debian Linux).

Виконаємо установку Raspbian за допомогою установника NOOBS операційної системи для Raspberry Pi 3, який містить Raspbian. Для цього треба завантажити з сайту архівний файл NOOBS на ПК. Потім розпакувати зміст архіву NOOBS в окрему директорію на ПК і скопіювати зміст в кореневу директорію карти microSD, підключеної через кардридер до ПК. Після цього потрібно витягти microSD з кардридера і вставити її в роз'єм на панелі "Raspberry Pi 3 Model B". Після підключення до комп'ютера Raspberry Pi монітора, маніпулятора "миша", клавіатури і блоку живлення повинна завантажитися NOOBS.

Із заставки зі списком треба вибрати "Raspbian" і натиснути "Install". Комп'ютер Raspberry Pi завантажується з графічним інтерфейсом користувача, але його можна відключити і налаштувати комп'ютер так, щоб він завантажувався з терміналом "LXTerminal" (з інтерфейсом командного рядка). Крім того, на Raspberry Pi з графічним інтерфейсом користувача можна відкрити LXTerminal з панелі завдань, клацнувши на піктограмі LXTerminal (рисунок 8.2).

Рисунок 8.2 – Робоче місце на базі контролера Raspberry Pi 3 Model B

Завдання 2. Ознайомитися з програмною платформою Node.js і встановити платформу Node.js на Raspberry Pi.

Для реалізації веб-сервера і WoT-додатків для Smart Devices застосуємо програмну платформу Node.js (v10.16.0) для роботи з JavaScript і модуль Express.js (4.17.1) або фреймворк для Node.js. Node.js, що підходить для реалізації різних серверів (Web, CoAP, WebSocket, MQTT і т.д.) і для побудови масштабованих Web-додатків.

Перш ніж встановлювати платформу Node.js на контролер, необхідно визначити версію архітектури ядра процесора на Raspberry Pi. Для перевірки версії архітектури процесора можна використовувати команди: **uname -m** або **cat / proc / cpuinfo**, які необхідно ввести в LXTerminal. Результатом перевірки версії моделі процесора на "Raspberry Pi 3 Model B" є: ARMv7 Processor rev 4 (v7l). На сайті (nodejs.org/en/download/) для версії "ARMv7" рекомендована версія платформи node-v10.16.0.

Node.js можна встановити з репозиторіїв nodesource.com або nodejs.org, що містять останні версії Node.js. Для встановленої на "Raspberry Pi 3 Model B" операційної системи Debian Linux version 4.19.59 можна завантажити дані з командного рядка для версії ARMv7:

curl -sL https://deb.nodesource.com/setup_10.x | sudo -E bash - встановити Node.js:

sudo apt-get install -y nodejs

і перевірити версію:

node –v

v10.16.0.

Слід зазначити, що для скачування файлів можна використовувати консольну утиліту wget, наприклад, зі сховищ NodeSource (https://deb.nodesource.com/setup_10.x):

wget -qO- https://deb.nodesource.com/setup_10.x | bash -

або зі сховища nodejs.org:

wget https://nodejs.org/dist/v10.16.0/node-v10.16.0-linux-armv7l.tar.gz

з наступною розпакуванням файлу і установкою в файлову систему.

Завдання 3. Реалізувати директорію для WoT-додатка і каркас додатка.

Створимо WoT-додаток (Web Thing) з інтерфейсом користувача і програмним інтерфейсом Web Thing REST API для доступу і керуванням як з браузера, так і зі сторонніх WoT-додатків. Створюваний WoT-додаток призначений для Smart Objects, тобто для приводу (actuators). Як actuator застосуємо світлодіод (імітує привід).

Слід зазначити, що доступ до ресурсів WoT-додатків можна здійснювати через протоколи: HTTP, CoAP, WebSocket, REST / HTTP, MQTT. Протокол HTTP може бути використаний для доступу через інтерфейс користувача за допомогою браузерів до WoT-додатків, поміщених на HTTP-серверах. Протоколи CoAP, WebSocket, REST/HTTP можуть бути використані для доступу через API одних WoT-додатків до інших WoT-додатків, розміщених на серверах CoAP, WebSocket, HTTP відповідно.

Для налагодження доступу до WoT-додатків по протоколам CoAP і WebSocket можна використовувати браузер Firefox з плагіном Copper (Copper (Cu) CoAP Firefox plugin) і з розширенням Wang Fenjin відповідно. Доступ по протоколу MQTT до WoT-додатків з встановленими на них MQTT-клієнтами здійснюється через сервери-брокери (наприклад, mqtt.org) за

допомогою клієнтських додатків (MQTT-клієнтів), встановлених на ПК або смартфонах.

Розробку почнемо зі створення WoT-додатка для приводу (світлодіода) з інтерфейсом користувача, який міститься на HTTP-сервері. Для цього за допомогою файлового менеджера в директорії "pi" створимо директорію wot-led і каркас WoT-додатка в складі двох директорій: public, в якій будуть знаходитися всі статичні файли; views, в якій будуть зберігатися шаблони, і двох файлів: package.json - для зберігання всієї інформації про програму; server.js - для різних серверів і серверного WoT-додатка.

За допомогою терміналу pi @ raspberrypi: ~ $ перейдемо з кореневої директорії "pi" в створену директорію wot-led:

cd wot-led.

Далі в директорії **pi@raspberrypi: ~/wot-led $** виконаємо команду для формування файлу **package.json**:

npm init

Дана команда запустить опитувальник командного рядка, який завершиться створенням або зміною файлу package.json, для цього можна заповнювати запропоновані варіанти або не заповнювати, а натискати Enter.

Потім встановимо фреймворк express, виконавши команду:

npm install express –save

В результаті буде встановлена бібліотека express (в нашому випадку версія v.4.17.1), назва якої збережеться в файлі package.json, а зміст бібліотеки буде встановлено в створену директорію node_modules в папці wot-led.

Потім встановимо бібліотеку для роботи з GPIO на Raspberry Pi, бібліотеку PATH для вказівки каталогів виконуваних програм і шаблонизатор, який буде передавати дані в шаблон.

Вибираємо шаблон з розширенням .ejs, у цьому шаблоні можна виводити html код сайту Smart Devices з впровадженими скриптами, стилями і зображеннями.

- **npm install rpi-gpio –save**
- **npm install path –save**
- **npm install ejs –save**

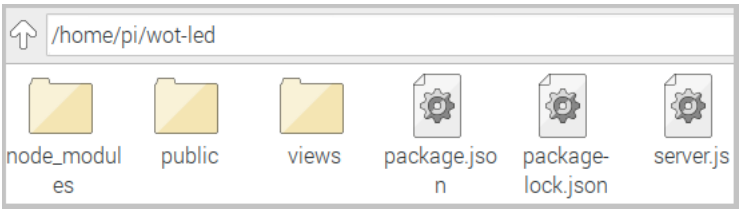


Рисунок 8.3 – Структура WoT-додатка

Завдання 4. Підключити actuator (в якості приводу застосуємо світлодіод LED) до портів GPIO.

До портів GPIO підключимо actuator, в якості якого можна застосувати світлодіод LED. Вивідний світлодіод (5мм, колір червоний, номінальний струм: 20 мА) довгою ніжкою підключаємо до порту GPIO 4 (контакту 7) і короткою ніжкою світлодіода через обмежувальний резистор (330 Ом) до порту GND. Слід зазначити, що для червоних світлодіодів допустима величина напруги може бути в межах від 1,8 до 2,4 В. Напруга між контактами або пінами GPIO (введення / виведення) і GND становить 3,3 (рисунок 8.4, 8.5).

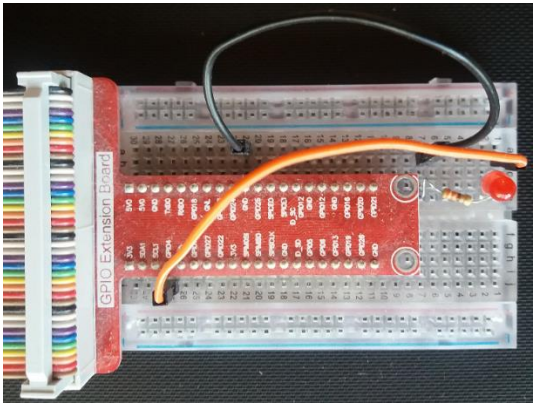


Рисунок 8.4 – Світлодіод, підключений до портів GPIO

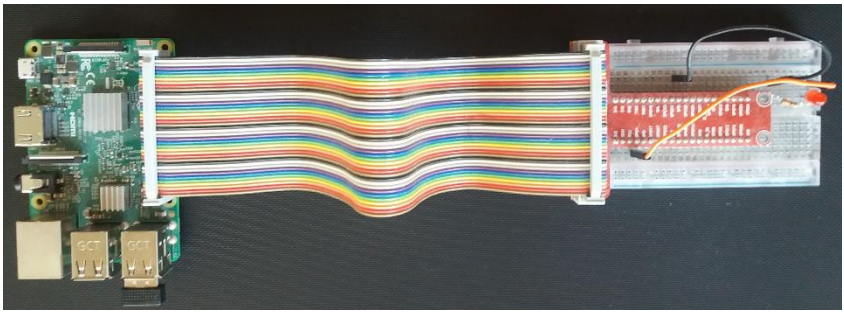


Рисунок8.5 – «Raspberry Pi 3 Model B» і світлодіод, підключений до портів GPIO

Завдання 5. Реалізувати серверну частину WoT-додатка (server.js) із застосуванням фреймворка express.

Розробку файлу серверного WoT-додатка server.js або додатка express почнемо з завантаження фреймворку express за допомогою функції **require()**. Це WoT-додаток поміщено на HTTP-сервер, загорнутий в структуру бібліотеки express. Далі створюємо об'єкт докладання express з ім'ям app. Для керуванням світлодіодом (режимом читання/запис) завантажуємо бібліотеку "rpi-gpio" і бібліотеку path для вказівки абсолютного шляху до каталогу для надання файлів.

Далі налаштовуємо контакт 7 на режим запису DIR_OUT. Потім визначаємо тип шаблонізатора з розширенням .ejs. Викликаємо функцію тимчасової роботи (проміжне ПО або middleware) express.static для маршруту за замовчуванням. Визначаємо обробники для маршрутів на GET і POST HTPP-запити. Запускаємо HTTP-сервер, а для WoT-додатки призначаємо порт: 8585. Код серверного WoT-додатки server.js представлений в додатку 1.

Завдання 6. Реалізувати клієнтську частину WoT-додатка.

Створюємо клієнтську частину WoT-додатка або файл графічного інтерфейсу користувача index.ejs на основі мови розмітки html5, мови таблиці стилів CSS3, css-фреймворка для Bootstrap Buttons. Bootstrap надає різні стилі кнопок, які застосовані в додатку. Для застосування Bootstrap Buttons необхідно прописати підключення необхідних файлів з cdn-

сховища Bootstrap між тегами <head> </head> у веб-сторінці index.ejs.

Крім того, для кнопки LED-On застосуємо class = "btn btn-danger", а для кнопки LED-Off застосуємо class = "btn btn-primary". Файл index.ejs помістимо в директорію views, а файли style.css і less.png помістимо в директорію public. Код графічного інтерфейсу користувача index.ejs представлений в додатках 2 і 3.

Завдання 7. Апробувати роботу WoT-додатка.

Щоб протестувати сервер необхідно запустити WoT-додаток через термінал. Для цього потрібно перейти в директорію wot-led, де знаходиться WoT-додаток і запустити server.js, виконавши команду: node server.js. В цьому випадку в терміналі з'явиться запис "WoT-додаток прослуховує порт: 8585". Зображення терміналу представлений на рисунку 8.6.

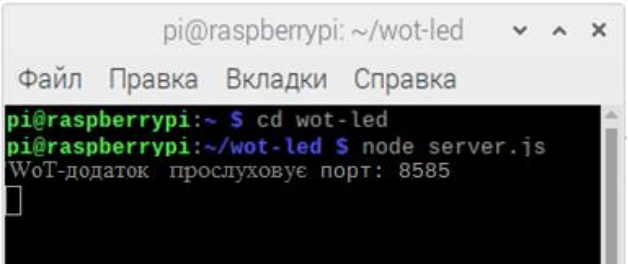


Рисунок 8.6 – Скріншот терміналу з WoT-додатком

Після запуску програми через термінал можна отримати доступ до ресурсів в браузері за адресами: localhost: 8585; raspberrypi: 8585 або за IP-адресою 192.168.1.129:8585, яка призначена wireless router , (наприклад, LinksysWRT160N) локальної мережі. Доступ до світлодіоду за IP-адресою може бути виконаний як з браузера ОС Raspbian на моніторі, підключеному до Raspberry pi 3, так і з браузерів настільного ПК, ноутбука, смартфона тощо.

Скріншоти графічного інтерфейсу користувача або веб-сторінки index.html, представлені на рисунку 8.7, 8.8, містять кнопки для включення/виключення світлодіода LED-On/LED-Off,

а також рядок Position світлодіодів, в якому можуть відображатися записи: "Press button LED-On!", "On" або "Off".

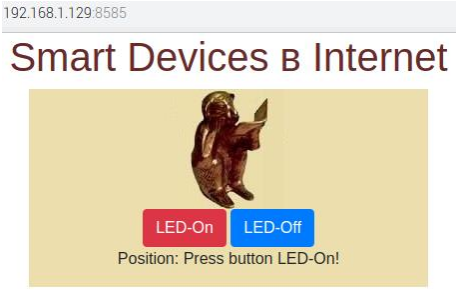


Рисунок 8.7 – Скріншот графічного інтерфейсу користувача

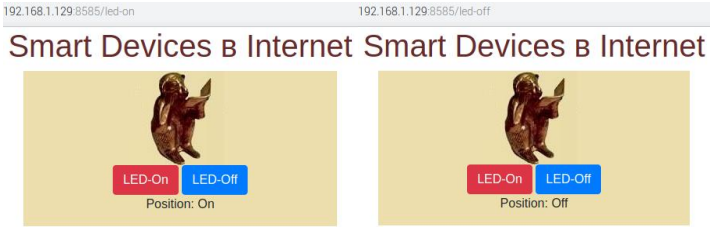


Рисунок 8.8 – Скріншоти графічного інтерфейсу користувача

При запуску в терміналі server.js в браузері відображається "Press button LED-On!", при цьому світлодіод вимкнений. При натисканні в браузері кнопки LED-On включається світлодіод і відображається значення "On", а в командному рядку з'явиться запис "pin 7: true". При натисканні в браузері кнопки LED-Off вимикається світлодіод і відображається значення "Off", а в командному рядку з'явиться запис "pin 7: false" і т.ін.

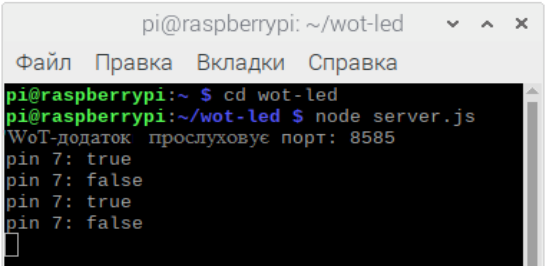


Рисунок 8.9 – Скріншот терміналу с WoT-додатком

Зміст звіту

1. Схеми апаратних засобів для вирішення завдань.
2. Тексти програм для завдань.
3. Результати моделювання.
4. Результати виконання експериментів.
5. Висновки по роботі.

Контрольні запитання

1. Назвіть інтеграційні шаблони для інтеграції Smart Objects в Інтернет.
2. Які технології застосовуються для інтеграції Smart Devices в Internet на вбудованій платформі?
3. У чому полягає суть технології Node.js, яка дозволяє створювати керовані подіями сервери?
4. Як визначити обробник для маршруту "/" на метод GET-запиту до WoT-додатка?
5. Який механізм організації обміну даними між Smart Objects з використанням протоколів HTTP, CoAP, WebSocket, MQTT?

Література

1. Dominique D. Guinard and Vlad M. Trifa, «Building the Web of Things», Manning Publications.: United States, June 2016, 344 p.
2. В. Ткаченко, «Web of Things - сетевая служба IoT». [Електронний ресурс]. – Режим доступу: http://www.lessons-tva.info/articles/net/014.html (дата звернення 01.08.2019 р.).
3. Matthias Kovatsch, Ryuichi Matsukura, Michael Lagally, Toru Kawaguchi, Kunihiko Toumura, Kazuo Kajimoto, «Web of Things (WoT) Architecture: W3C Editor's Draft 22 August 2019». [Електронний ресурс]. – Режим доступу: https://w3c.github.io/wot-architecture/#sec-building-blocks (дата звернення 01.08.2019 р.).
4. В. Ткаченко, «Технологии интеграции Devices в Internet на основе встроенной платформы». [Електронний ресурс]. – Режим доступу: https://www.lessons-tva.info/articles/net/017.html (дата звернення 01.08.2019 р.).

5. Internet of Things for Industry and Human Applications. In Volumes 1-3. Volume 2. Modelling and Development / V. S. Kharchenko (ed.).– Ministry of Education and Science of Ukraine, National Aerospace University “KhAI”, 2019. – 578 p.

Додаток 1. Код серверного WoT-додатка server.js

```
// Директива use strict.
'use strict';
// Підключаємо веб-фреймворк express.
const express = require('express');
// Створюємо об'єкт додатка.
const app = express();
// Підключаємо бібліотеку управління світлодіодом.
const gpio = require('rpi-gpio');
// Підключаємо бібліотеку path.
const path = require('path');
// Налаштовуємо контакт 7 на режим запису DIR_OUT (пін 7, відповідний GPIO4).
gpio.setup(7, gpio.DIR_OUT);
// Визначаємо механізм візуалізації: шаблонизатор з розширенням .ejs.
app.set('view engine', 'ejs');
// Викликаємо функцію тимчасової роботи express.static для маршруту за замовчуванням.
app.use(express.static(path.join(__dirname, 'public')));
// На метод GET HTTP-запиту до додатка визначаємо обробник для маршруту "/".
app.get('/', function(req, res){
// Визначаємо відповідь серверної частини для клієнтської частини програми Node.js методом res.render().
        res.render('index',{position:" Press button LED-On!"});
});
// На метод POST-запиту до додатка по маршруту led-on, WoT-додаток відповідає: включити світлодіод.
app.post('/led-on', function(req, res){
gpio.write(7, true, function(err) {
    if (err) throw err;
```

```
                console.log('pin 7: true');
            return res.render('index', {position: 'On'});
    });
});
// На метод POST-запиту до додатка по маршруту led-off, WoT-додаток відповідає: вимкнути світлодіод.
app.post('/led-off', function(req, res){
gpio.write(7, false, function(err) {
      if (err) throw err;
            console.log('pin 7: false');
        return res.render('index',{position: 'Off'});
    });
});
// Старт HTTP-сервера!
app.listen(8585, function () {
  console.log('WoT-додаток прослуховує порт: 8585')
})
```

Додаток 2. Код графічного інтерфейсу користувача index.ejs

```
<!DOCTYPE html>
 <html lang="en">
 <head>
   <meta charset="utf-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>LED application </title>
    <link rel="stylesheet" href="public/style.css">
    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
     integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
</head>
<body>
```

```
<h1>Smart Devices в Internet </h1>
<div id="wrapper">
    <div id="content">
    <div class="center">
    <img src="public/less.png" alt="LED" width="120" height="120"> <br>
    <form action="/led-on" method="post">
    <button type="submit" class="btn btn-danger">LED-On </button>
    <button type="submit" formaction="/led-off" class="btn btn-primary">LED-Off </button>
    </form>
    <p>Position: <%=position%> </p>
    </div>
</div>
</div>
</body>
</html>
```

Додаток 3. Файл style.css

```
@charset "utf-8";
body{font-size:100%; font-family:Arial, Helvetica, Sans-serif, Verdana;}
h1{font-size:120%; line-height:1.2em; font-weight:bold; color:#662b2b; text-align:center}
.center{display:block; margin:0 auto; position:relative; text-align:center}
/* ID селектор */
#wrapper {
width: 400px;
margin: 0 auto;
height: 100%;
position: relative;
}
#content{
background-color:#EDDFAD;
margin:0px;
height: 100%;}
```

Лабораторна робота 9. Початок роботи із PLCnext

Мета роботи: ознайомитись з можливостями інструментального засобу PLCnext Engineer для конфігурування контролеру та створення програм з простими логічними операціями на мові релейно-контактної логіки. PLCnext контролер виробництва компанії Phoenix Contact) використовується для створення систем індустріального Інтернету речей (IoT).

Програма роботи

1. Ознайомитися з програмовним контролером PLCNext AXC F 2152.
2. Створити конфігурацію вхідних/вихідних модулей, налаштувати параметри контролеру.
3. Ознайомитися з мовою програмування LD.
4. Реалізувати користувацьку програму.
5. Апробувати роботу користувацької програми.
6. Підготувати звіт про виконану роботу.

Теоретичний матеріал

Програмовні контролери PLCnext є платформою для створення проектів автоматизації з вбудованими можливостями Інтернету речей [1]. Для конфігурування, програмування, візуалізації та діагностики використовується PLCnext Engineer [2] – інженерне програмне забезпечення, що відповідає стандарту IEC 61131-3.

Створені додатки можуть бути завантажені до PLCnext Store [3], відкритого магазину програмного забезпечення для автоматизації.

Завдання 1. Створення і завантаження конфігурації контролера

Запустіть PLCEngineer, створіть новий проект, обравши «Empty AXC F 2152 v00 / 2019.3.0 project» (рисунок 9.1)

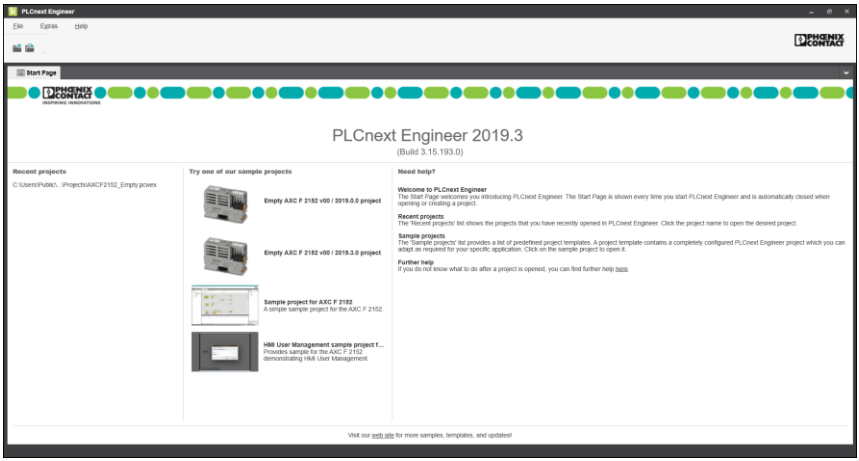


Рисунок 9.1 – Стартова сторінка PLCnext Engineer

Оберіть вкладку «Settings» розділу «AXC F 2152», налаштуйте IP-адресу контролеру (рисунок 9.2).

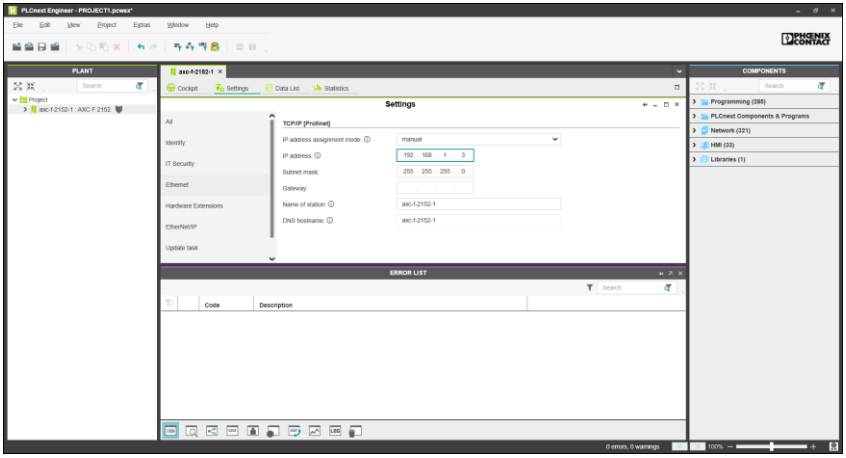


Рисунок 9.2 – Конфігурування IP-адреси контролера у проекті

Оберіть вкладку «Online Devices» розділу «Project», виконайте пошук підключених контролерів (рисунок 9.3).

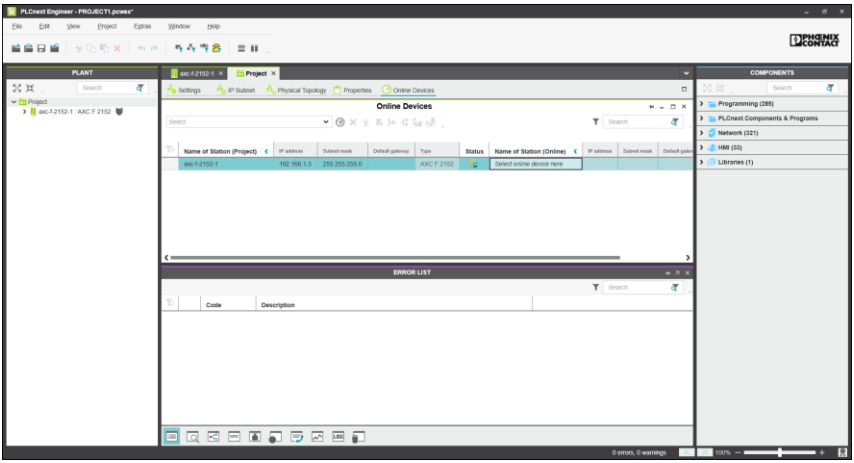


Рисунок 9.3 – Пошук онлайн пристроїв

У разі, якщо IP-адреса відрізняється від сконфігурованої у проекті, завантажте її у контролер, обравши пункт «Apply the Project Device Settings to the Online Device» контекстного меню (рисунок 9.4).

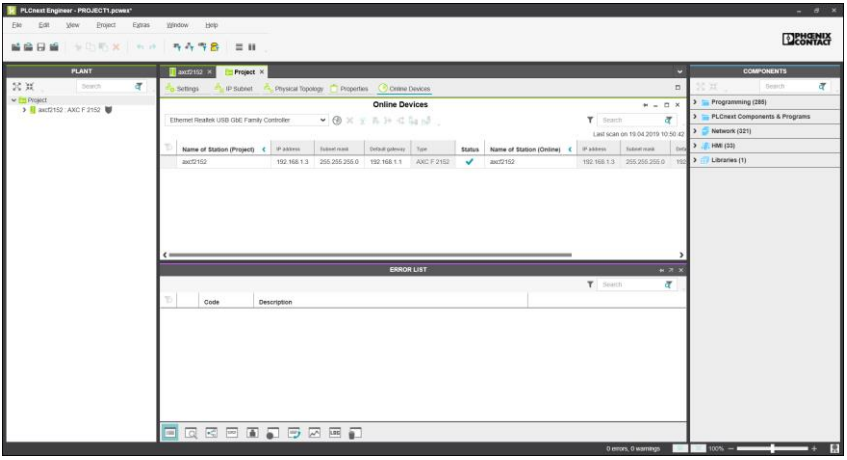


Рисунок 9.4 – Результати пошуку онлайн пристроїв

У розділі «Axioline F» сконфігуруйте вхідні / вихідні модулі, у послідовності, що відповідає лабораторному стенду (рисунки 9.5, 9.6).

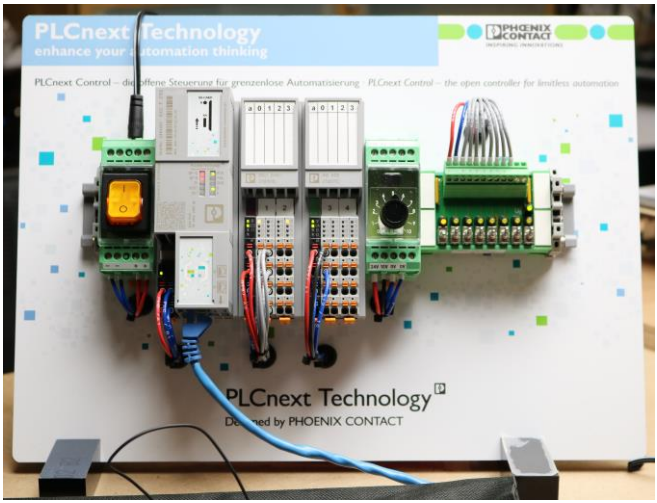


Рисунок 9.5 – Лабораторний стенд

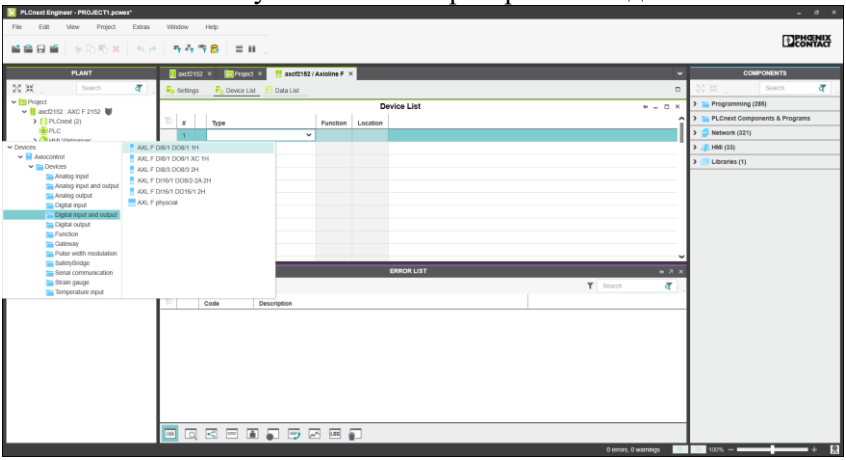


Рисунок 9.6 – Налаштування вхідних та вихідних модулів

Після створення конфігурації (рисунок 9.7) виконайте компіляцію проекту, обравши «Project» – «Rebuild»

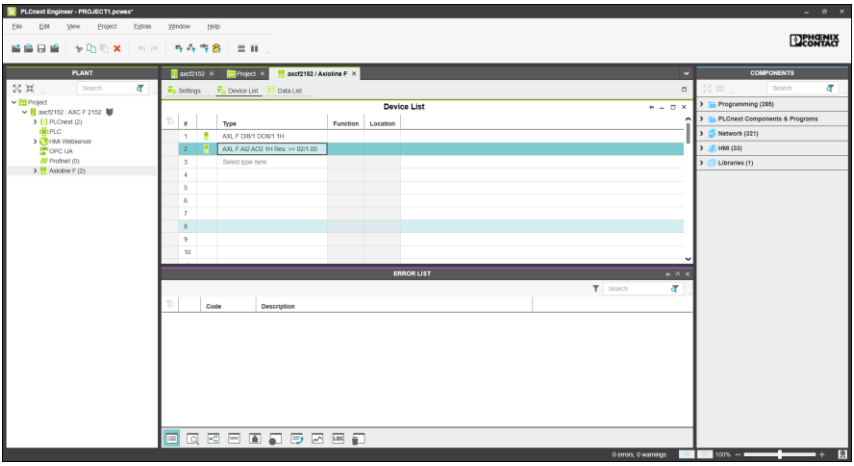


Рисунок 9.7 – Приклад налаштування вхідних та вихідних модулів для лабораторного стенду

Завантажте конфігурацію у контролер, натиснувши кнопку «Write project to controller and start execution» на вкладці «Cockpit» розділу «AXC F 2152» (рисунок 9.8).

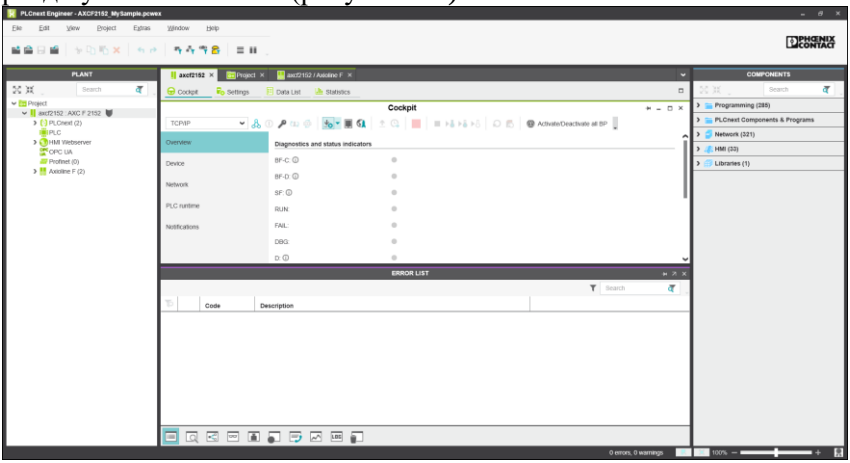


Рисунок 9.8 – Завантаження створеної конфігурації в контролер

Для перегляду стану контролеру онлайн натисніть кнопку «Attach to the PLC runtime to see online values and enable debugging» на вкладці «Cockpit» розділу «AXC F 2152» (рисунок 9.9).

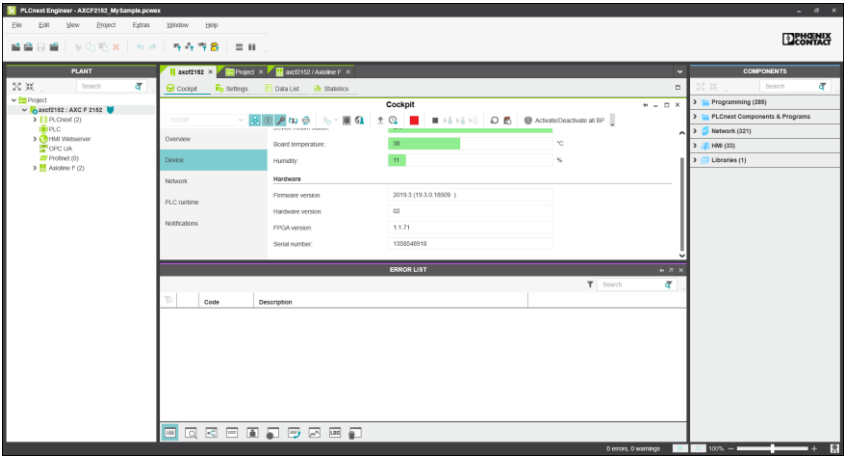


Рисунок 9.9 – Перегляд стану контролеру онлайн

Завдання 2. Створення програми на мові LAD

В розділі «Components» оберіть пункт «Programming» - «Local» - «Programs» - «Main», далі оберіть «Add LD Code Worksheet» (рисунок 9.10).

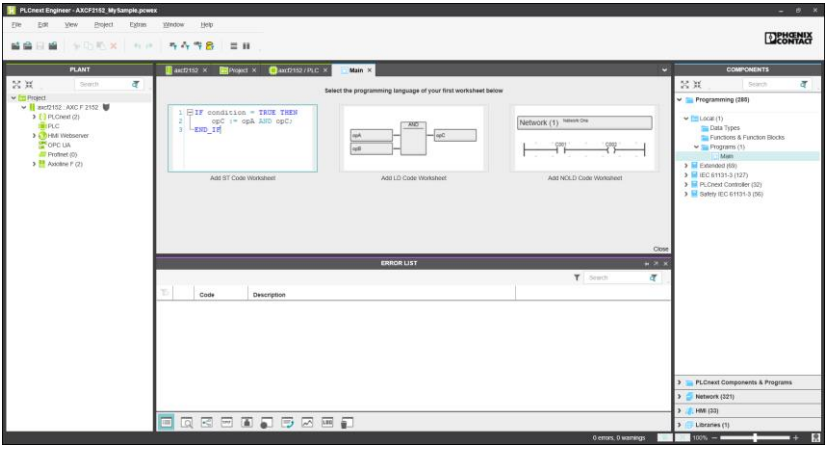


Рисунок 9.10 – Створення програми

Проаналізуйте, які входи та виходи знадобляться для вирішення задачі у відповідності із варіантом, наведеним у додатку А, та додайте необхідні вхідні та вихідні змінні на вкладці «Variables» (рисунок 9.11).

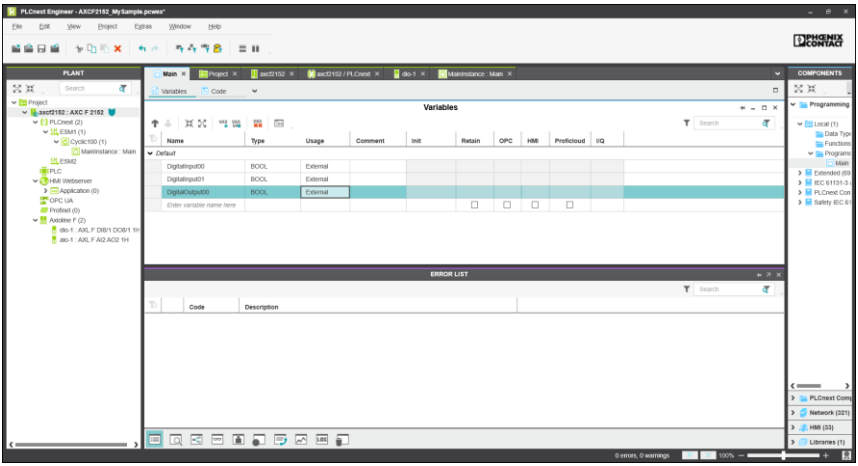


Рисунок 9.11 – Створення змінних

Змінні, які, використовуються для введення або виводу інформації, необхідно прив’язати до відповідних вхідних та вихідних бітів (рисунки 9.12, 9.13).

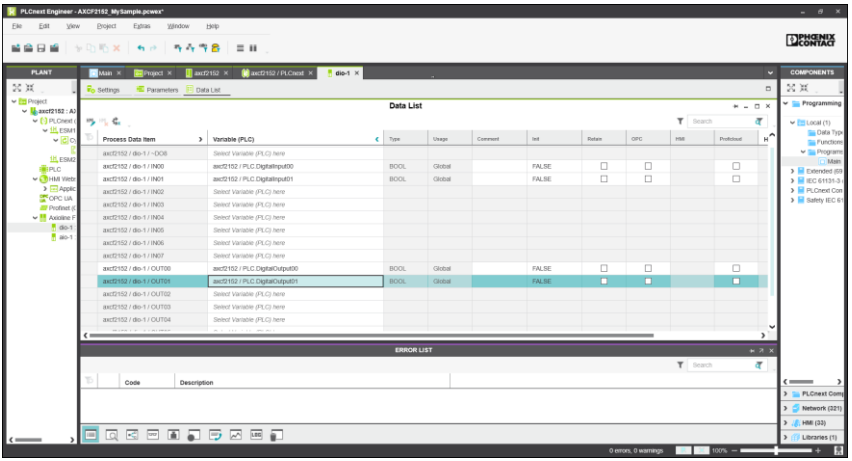


Рисунок 9.12 – Прив’язування змінних до бітів модуля дискретних входів/виходів

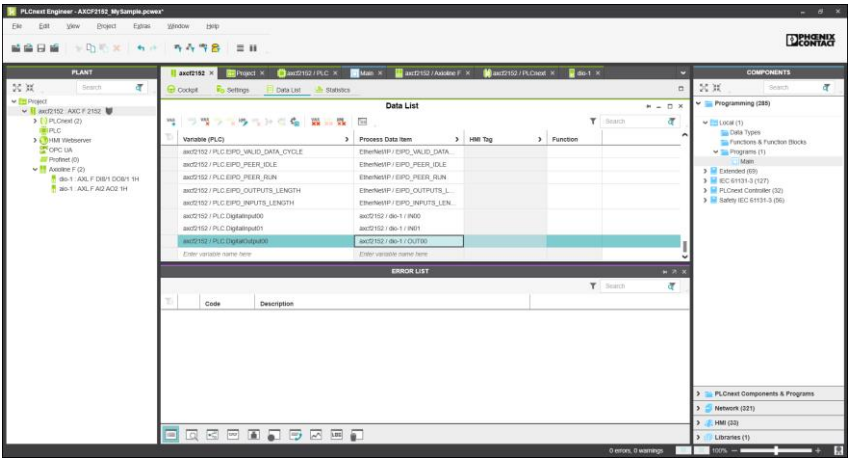


Рисунок 9.13 – Перегляд змінних, прив’язаних до бітів модуля дискретних входів/виходів

Використовуючи перемикачі, перевірте, що відповідні змінні змінюють своє значення при переключенні (рисунки 9.14-9.17). Для перегляду необхідно натиснути кнопку «Attach to the PLC runtime to see online values and enable debugging» на вкладці «Cockpit» розділу «AXC F 2152».

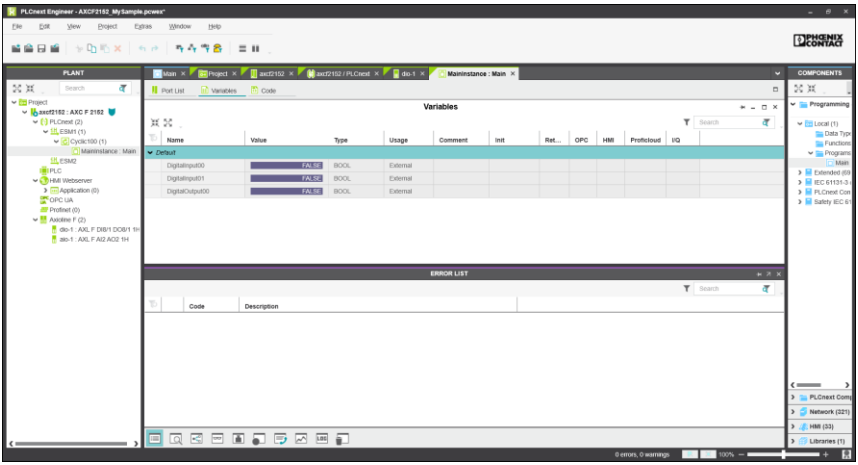


Рисунок 9.14 – Перегляд поточних значень змінних

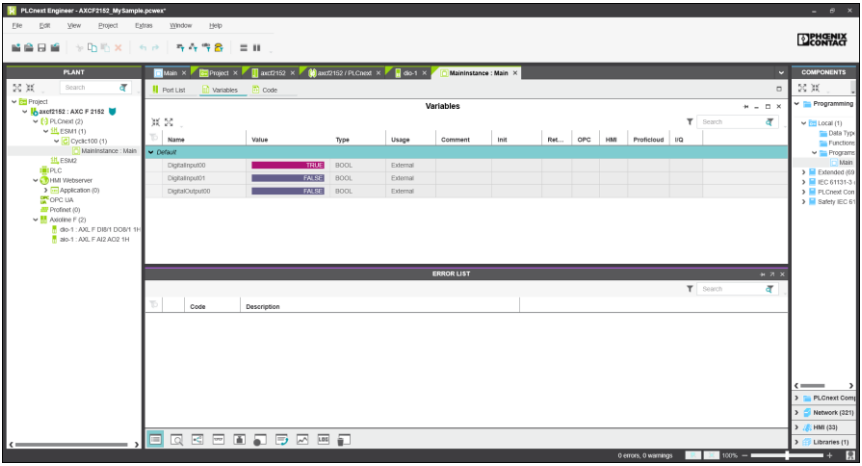


Рисунок 9.15 – Перегляд поточних значень змінних

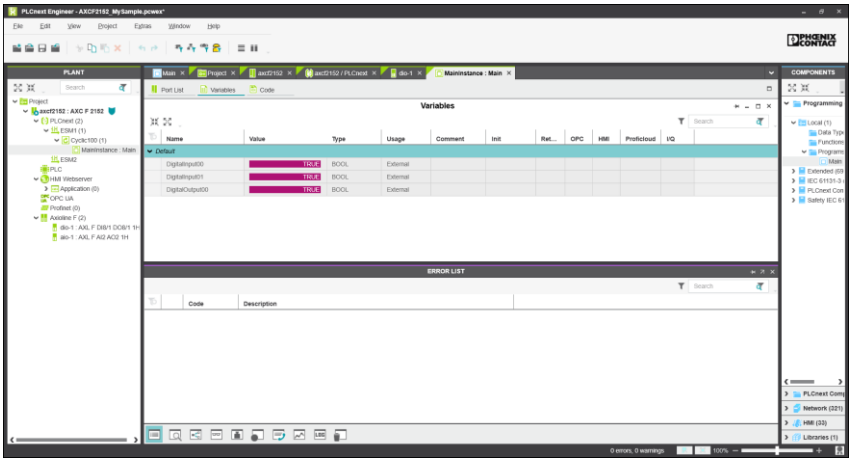


Рисунок 9.16 – Перегляд поточних значень змінних

Використовуючи мову програмування LD, створіть користувацьку програму у відповідності із варіантом завдання (додаток А). На рисунку 9.17 наведено приклад простої програми.

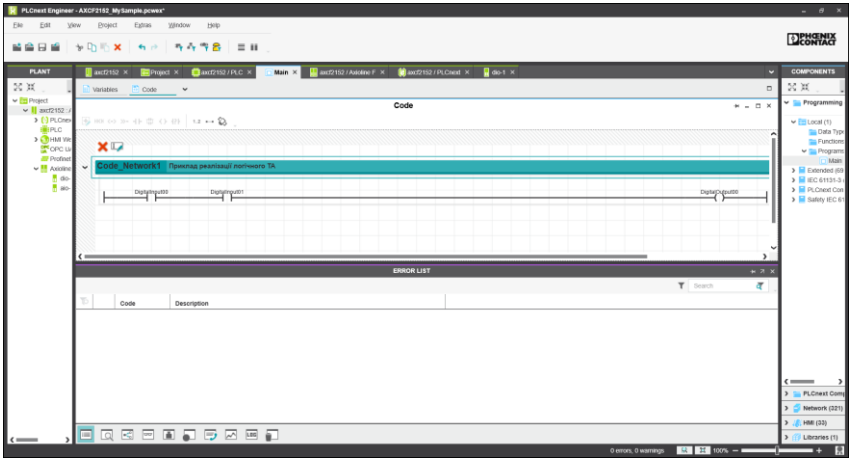


Рисунок 9.17 – Приклад програми

У разі потреби додайте нові нетворки (рисунок 9.18).

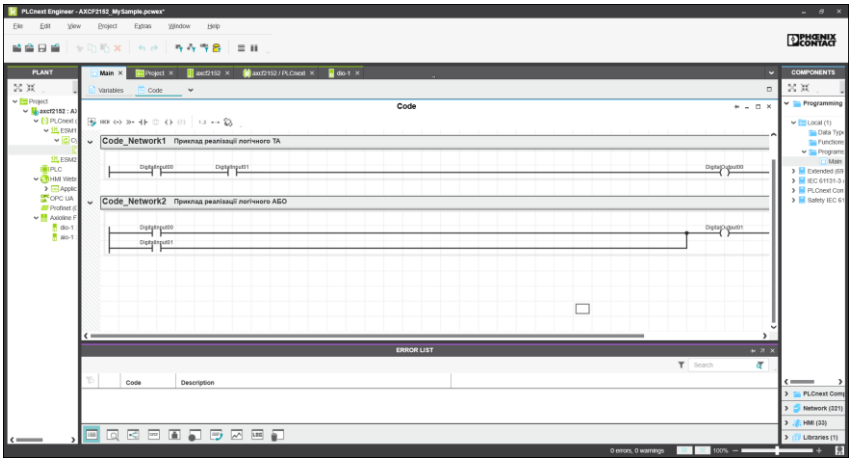


Рисунок 9.18 – Приклад програми, що складається з декількох нетворків

Після створення програми виконайте компіляцію проекта «Project» - «Rebuild».

Перевірте коректність роботи програми на тестових наборах шляхом змінювання положення перемикачів (рисунки 9.19-9.21).

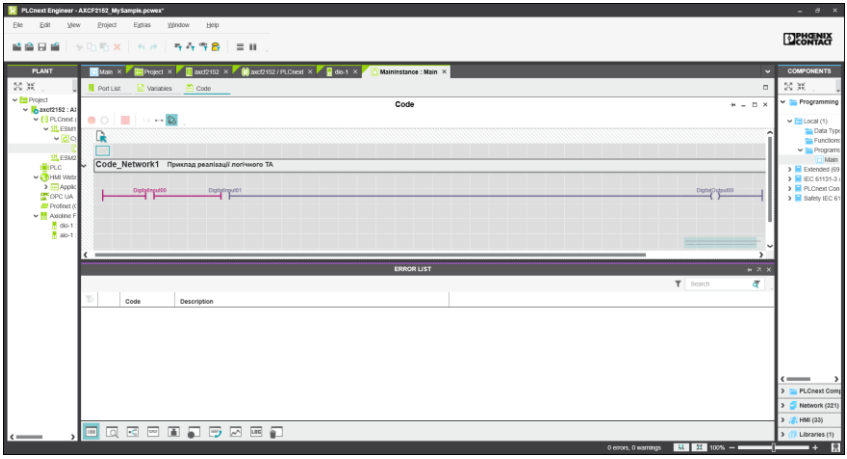


Рисунок 9.19 – Перевірка роботи програми

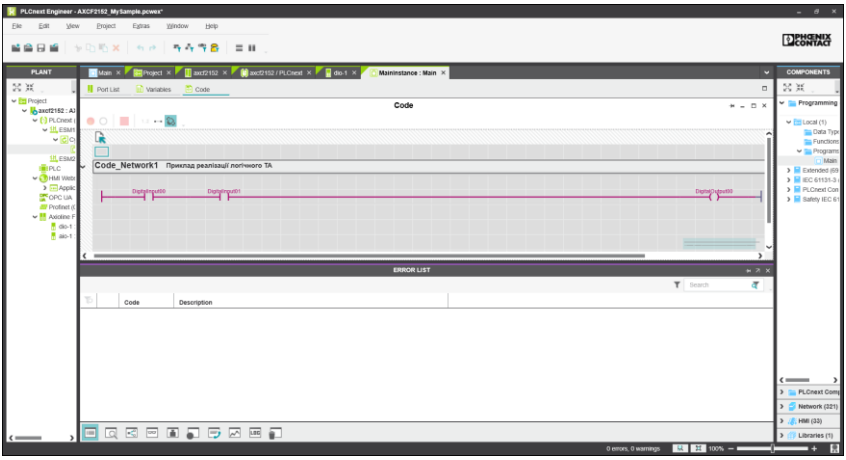


Рисунок 9.20 – Перевірка роботи програми

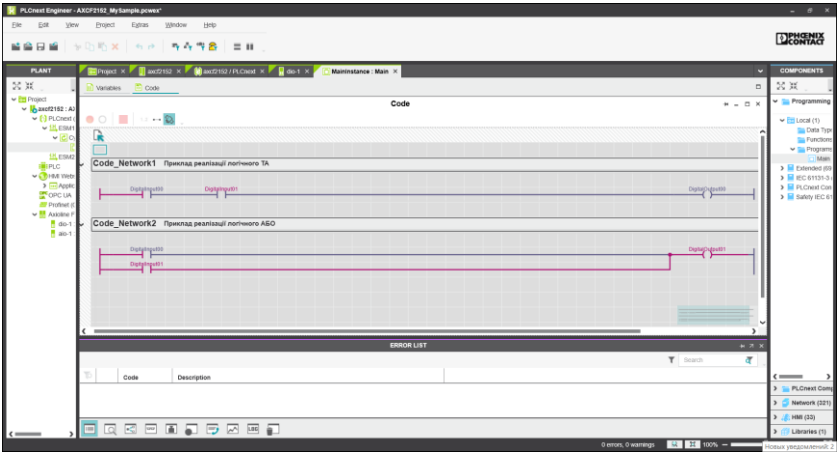


Рисунок 9.21 – Перевірка роботи програми

До звіту з лабораторної роботи включити опис виконання, а також роздрукувати штатними засобами PLCnext Engineer конфігурацію модулів та програми (рисунок 9.22).

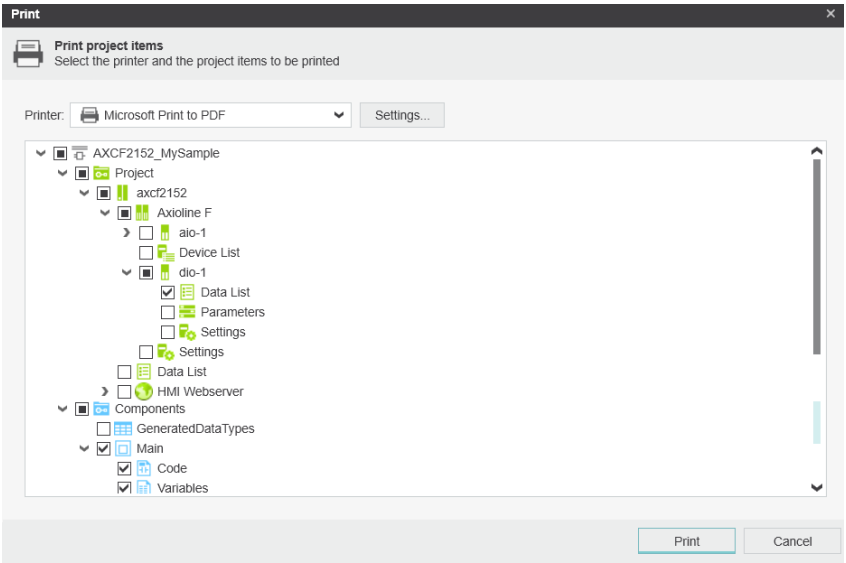


Рисунок 9.22 – Друк частин проекту

Зміст звіту

1. Конфігурація модулів для вирішення завдань.
2. Опис вхідних та вихідних даних.
3. Тексти програм для завдань.
4. Результати перевірки роботи програми.
5. Висновки по роботі.

Контрольні запитання

1. Назвіть різновиди модулів програмовних контролерів.
2. Для чого необхідно створення конфігурації?
3. З яких елементів складається програма на мові LD?
4. Яким чином можливе проведення перевірки коректності роботи програми?
5. Для чого необхідна прив’язка змінних до вхідних або вихідних адрес?

Література

1. Філіп’їчев П.Р., Степанець О. В. PLCnext technology в сучасних АСУТП // Материалы XIV международной научно-

практической конференций «Бъдещето въпроси от света на науката». София, Болгария. 15-22 декабря 2018 г. – 88 с.

2. PLCNext Engineer. Getting started: Program in IEC 61131-3 languages [Електронний ресурс]. – Режим доступу: https://www.plcnext-community.net/index.php?option=com_content&view=category&layout=blog&id=80 (дата звернення 01.08.2019 р.).

3. PLCNext Store. [Електронний ресурс]. – Режим доступу: https://www.plcnextstore.com (дата звернення 01.08.2019 р.).

Додаток А. Варіанти завдань

Варіант №1

Команда на включення резервного насосу видається, якщо перемикач режиму роботи «Автоматичний / Дистанційний» знаходиться в положенні «Автоматичний» і зафіксована поломка основного насоса, або коли перемикач знаходиться в положенні «Дистанційний» і натиснута кнопка включення резервного насосу.

Варіант №2

Сигнал «Нормальна робота» видається, якщо перший котел працює, а другий вимкнений, або якщо перший котел вимкнений, а другий працює.

Варіант №3

Система кондиціонування повинна бути виключена, якщо відкрито хоча б одне з чотирьох вікон, або спрацювали обидва датчика низької температури у приміщенні.

Варіант №4

Продування фільтру повинно бути включена, якщо відсутній сигнал «Тиск після фільтра» і є сигнал «Тиск до фільтра», або оператором видана команда «Включити продування».

Варіант №5

Система охорони містить три датчики – датчик відкриття дверей, датчик руху і датчик розбиття скла. Сигналізація повинна бути включена, якщо спрацювали будь-які два датчика.

Варіант №6

Сигнал «Аварія» повинен бути виданий, якщо закриті обидва клапана на основному і резервному трубопроводах, або ці обидва клапана відкриті. Якщо відкритий тільки один клапан, сигнал «Аварія» не видається.

Варіант №7

Конвеєр повинен бути включений, якщо є сигнал «Бункер заповнений» і обраний режим «Автоматичний», або якщо обраний режим «Ручний» і натиснута кнопка «Включити конвеєр». Ручний або автоматичний режим обирається за допомогою однієї кнопки (натиснута – автоматичний режим, не натиснута – ручний).

Варіант №8

Шлагбаум повинен бути піднятий, якщо натиснута кнопка підняття шлагбаума, або якщо обрано режим автоматичного управління і автомобіль виїжджає зі стоянки або заїжджає на неї.

Варіант №9

Клапан повинен бути відкритий, якщо рівень води високий і немає аварії насоса, або якщо видана команда «Аварійний спуск води».

Варіант №10

Турнікет в метро повинен бути закритий, якщо проїзд не сплачено і здійснена спроба проходу, або черговим натиснута кнопка примусового закриття.

Варіант №11

Двері составу метро повинні бути відкриті за умови, що состав повністю зупинений і видана команда відкриття дверей або з кабіни машиніста, або з диспетчерської.

Варіант №12

Підприємство забезпечено автоматичними турнікетами. Робочі на підприємстві працюють у дві зміни. На територію підприємства можна пройти робочому персоналу першої зміни тільки в першу зміну, робочого персоналу другої – в другу, а інженерам – у будь-який час.

Варіант №13

У салоні автомобіля видається попереджувальний сигнал за умови, якщо двигун працює і температура охолоджуючої рідини вище норми, або тиск масла двигуна нижче мінімального допустимого рівня, або у автомобіля закінчується паливо.

Варіант №14

Лампа H1 повинна бути включена, якщо натиснута кнопка S1 і одна з кнопок S2 або S3.

Варіант №15

У печі встановлені 3 датчика (тиску, температури, витрати палива). Якщо спрацьовують хоча б два з них, то повинна бути включена сигналізація.

Варіант №16

Візок повинен бути зупинений, якщо є сигнал «Датчик положення 1» або «Датчик положення 2» і обраний режим «Автоматичний», або якщо обраний режим «Ручний» і натиснута кнопка «Зупинити візок». Ручний або автоматичний режим обирається за допомогою однієї кнопки (натиснута – автоматичний режим, не натиснута – ручний).

АНОТАЦІЯ

УДК 004.415/.416].031.6(076.5)=111

А.П. Плахтєєв, Є.В. Бабешко, В.А. Ткаченко, Ю.В. Здоровець. **Архітектури та розроблення систем Інтернету / Вебу Речей на основі вбудованих платформ. Лабораторні роботи** / За ред. В.С. Харченка. Міністерство освіти і науки України, Національний аерокосмічний університет ХАІ, 2019. - 147 с.

Книга містить матеріали практичної частини магістерського курсу «Основи Інтернету речей», розробленого в рамках проекту Internet of Things: Emerging Curriculum for Industry and Human Applications / ALIOT, 573818-EPP-1-2016-1-UK-EPPKA2- CBHE-JP, 2016-2019, що фінансується програмою ЄС ERASMUS+. Книга складається з 9 лабораторних робіт для модуля цього курсу «Архітектури та платформи Інтернету/Вебу речей. Лабораторні роботи включають завдання для вивчення платформ і розроблення систем індустріального Інтернету і Вебу речей з використанням Arduino, PLCNext та інших засобів.

Книга підготовлена українськими університетськими командами за підтримки колег з академічних закладів країн ЄС, що входять до консорціуму проекту ALIOT.

Книга призначена для магістрантів і аспірантів, які вивчають технології IoT, програмну і комп'ютерну інженерію, комп'ютерні науки. Може бути корисною для викладачів університетів і навчальних центрів, розробників систем IoT.

Рис.: 53. Посилань: 57. Таблиць: 7.

ЗМІСТ

Анатолій Павлович Плахтеєв
Євген Васильович Бабешко
Володимир Антонович Ткаченко
Юлія Володимирівна Здоровець

АРХІТЕКТУРИ ТА РОЗРОБЛЕННЯ СИСТЕМ ІНТЕРНЕТУ РЕЧЕЙ / ВЕБУ РЕЧЕЙ НА ОСНОВІ ВБУДОВАНИХ ПЛАТФОРМ

Лабораторні роботи

Редактор В.С. Харченко

Комп'ютерна верстка
О.О. Ілляшенко
Ю.В. Здоровець

Зв. план, 2019
Підписаний до друку 27.08.2019
Формат 60x84 1/16. Папір офс. No2. Офс. друк.
Умов. друк. арк. 8,31. Уч.-вид. л. 8,93. Наклад 150 прим.
Замовлення 270819-8

Національний аерокосмічний університет ім. М. Є. Жуковського
"Харківський авіаційний інститут"
61070, Харків-70, вул. Чкалова, 17
http://www.khai.edu

Випускаючий редактор: ФОП Голембовська О.О.
03049, Київ, Повітрофлотський пр-кт, б. 3, к. 32.
Свідоцтво про внесення суб'єкта видавничої справи до державного реєстру видавців, виготовлювачів і розповсюджувачів видавничої продукції
серія ДК No 5120 від 08.06.2016 р.

Видавець: ТОВ «Видавництво «Юстон»
01034, м. Київ, вул. О. Гончара, 36-а, тел.: +38 044 360 22 66
www.yuston.com.ua
Свідоцтво про внесення суєкта видавничої справи до державного реєстру видавців, виготовлювачів і розповсюджувачів видавничої продукції
серія ДК No 497 від 09.09.2015 р.