Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерна інженерія та управління
(повна назва)

Кафедра Автоматизації проектування обчислювальної техніки
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти другий (магістерський)
(освітньо-кваліфікаційний рівень)

Програмно-апаратний комплекс імітаційного моделювання систем логічного управління
(тема)

Виконав:

студент _2_ курсу, групи ___СКСм-18-2_____

Юзовицький С.І.
(прізвище, ініціали)

Спеціальність:__123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

Тип програми: ____ освітньо-професійна _____
(освітньо-професійна або освітньо-наукова)

Освітня програма:___Спеціалізовані комп'ютерні системи
(повна назва освітньої програми)

Керівник:_____доц. каф. АПОТ Рахліс Д.Ю._____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ ___Чумаченко С.В.__
(підпис) (прізвище, ініціали)

2019 р.

Харківський національний університет радіоелектроніки

Факультет_____Комп’ютерна інженерія та управління_____
Кафедра_____Автоматизації проектування обчислювальної техніки_____
Рівень вищої освіти_____другий (магістерський)_____
Спеціальність _____123 «Комп'ютерна інженерія»_____
(код і повна назва)
Тип програми_____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)
Освітня програма_____Спеціалізовані комп’ютерні системи_____
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2019 р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові _____Юзовицькому Сергію Ігоровичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи "Програмно-апаратний комплекс імітаційного моделювання систем логічного управління"_____
затверджена наказом університету від "04" листопада 2019 р. №_1624_____
2. Термін подання студентом роботи _____17.12.2019_____
3. Вихідні дані до роботи: алгоритм логічного управління, середовище розробки Arduino IDE, мова С, Arduino, логічний аналізатор Saleae Logic_____
4. Перелік питань, що потрібно опрацювати в роботі:_____
Вступ_____

1. Теоретична частина (особливості алгоритмів управління, опис апаратної бази для реалізації)_____
2. Діагностичне забезпечення систем логічного управління (діагностична програмно-апаратна імітаційна модель)_____
3. Практична реалізація запропонованого програмно-апаратного комплексу імітаційного моделювання на базі двох плат Arduino (написання коду, аналіз результатів діагностичного експерименту за допомогою логічного аналізатора)_____
Висновки_____

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) _____16 слайдів_____

_____

_____

5. Консультанти розділів роботи(п.5 включається до завдання за рішенням випускової кафедри)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Видача теми роботи, її узгодження та затвердження.	01.09.2019	
2.	Аналіз проблемної області, постановка задачі, вибір засобів реалізації	15.09.2019 – 30.09.2019	
3.	Розробка моделі діагностичного забезпечення	01.10.2019 – 06.10.2019	
4.	Розробка програмно-апаратного комплексу системи управління автомату з продажу напоїв.	07.10.2019 – 10.11.2019	
5.	Реалізація ПАК на базі двох плат Arduino	11.11.2019 – 20.11.2019	
6.	Проведення діагностичного експерименту з використанням логічного аналізатору	21.11.2019 – 28.11.2019	
7.	Оформлення пояснювальної записки.	29.11.2019 – 15.12.2019	
8.	Перевірка виконаного проекту, допуск до захисту.	16.12.2019	
9.	Захист роботи.	20.12.19	

Дата видачі завдання 01.09.2019

Студент_____
(підпис)

Керівник роботи _____ доц. каф. АПОТ Рахліс Д.Ю.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить: 84 сторінки, 27 рисунків, 20 посилань та 3 додатки.

СИСТЕМА ЛОГІЧНОГО УПРАВЛІННЯ, ARDUINO, ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ, ПРОГРАМНО-АПАРАТНИЙ КОМПЛЕКС, ВЕРИФІКАЦІЯ, ЛОГІЧНИЙ АНАЛІЗАТОР.

Метою даної магістерської атестаційної роботи є реалізація програмно-апаратного комплексу імітаційного моделювання систем логічного управління.

В атестаційній роботі було розглянуто особливості алгоритмів управління в системах реального часу, мови їх опису та апаратна реалізація на базі мікроконтролерів з використанням автоматних моделей.

Обґрунтовано доцільність використання імітаційного моделювання як діагностичного забезпечення систем логічного управління. Запропоновано структуру програмно-апаратного комплексу імітаційного моделювання на базі плат Arduino UNO та логічного аналізатора.

В якості прикладу системи логічного управління було розглянуто торгівельний автомат з продажу напоїв. Діагностичний експеримент показав можливість та доцільність використання плати Arduino на базі мікроконтролера Atmega328 для організації програмно-апаратного моделювання в рамках діагностичного забезпечення системи логічного управління.

ABSTRACT

The explanatory note contains: 84 pages, 27 figures, 20 links and 3 applications.

LOGICAL CONTROL SYSTEM, ARDUINO, SIMULATION MODELING, SOFTWARE-HARDWARE COMPLEX, VERIFICATION, LOGICAL ANALYZER.

The purpose of this master's certification work is to implement a software-hardware complex for simulation modeling of logical control systems.

In the certification work, the features of control algorithms in real-time systems, languages of their description and hardware implementation based on microcontrollers using automata-based models were considered.

The expediency of using simulation as a diagnostic support of logical control systems is substantiated. The structure of a software-hardware complex for simulation based on Arduino UNO boards and a logic analyzer is proposed.

As an example of a logical control system, a vending machine selling drinks was considered. The diagnostic experiment showed the possibility and expediency of using the Arduino board based on the microcontroller Atmega328 for organizing hardware-software modeling as part of the diagnostic support of the logical control system.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

IDE	–	Integrated Development Environment (інтегроване середовище розробки);
МК	–	мікроконтролер;
СЛУ	–	система логічного управління;
МЕК	–	міжнародна електротехнічна комісія;
COM-порт	–	communication port (последовательный порт);
FSM	–	Finite State Machine (кінцевий автомат);
ЦАП	–	аналого-цифровий перетворювач;
АЦП	–	цифро-аналоговий перетворювач.

ВСТУП

В даний час складно знайти хоча б одну область виробництва, де не розглядаються питання автоматизації. Промислова автоматизація реалізується як правило на системах логічного управління (СЛУ). Складність таких систем обумовлює необхідність застосування нових підходів до їх проектування та технічної діагностики. З часом стало зрозуміло, що тестування алгоритмів управління формальними методами верифікації програм не може дати абсолютної впевненості у коректності СЛУ. Цей факт є наслідком того, що для тестування алгоритму управління потрібно мати модель об’єкт управління, що є також не простою задачею з огляду на складність створення математичної моделі об’єктів та того факту, що експерименти над реальними об’єктами найчастіше неможливі або надто дорогі. Верифікація алгоритмів управління через моделювання дозволяє знайти помилки проектування до створення реального обладнання та його програмного забезпечення. Таке тестування на ранніх етапах знижує необхідність в дорогих і важкодоступних прототипах.

Для виконання ранньої верифікації необхідно об'єднати моделі алгоритму управління і об'єкта управління в єдину систему і провести симуляцію поведінки в одній системній моделі. Виходячи з того, що системи логічного управління працюють у режимі реального часу, то вочевидь верифікація їх моделей також повинна брати до уваги часові параметри. Моделювання систем управління в реальному часі називається імітаційним моделюванням.

Ключовим моментом в імітаційному моделюванні є виділення і опис станів системи. Система характеризується набором змінних станів, кожна комбінація яких описує конкретний стан. Отже, шляхом зміни значень цих змінних можна імітувати перехід системи з одного стану в інший. Таким чином, імітаційне моделювання – це представлення динамічної поведінки системи за допомогою переводу її з одного стану до іншого відповідно до

певних правил. Ці зміни станів можуть відбуватися або безперервно, або в дискретні моменти часу. Імітаційне моделювання є динамічне відображення змін стану системи з плином часу.

Такий підхід до діагностики СЛУ дозволяє запобігти аварійній ситуації в результаті помилок управління, дозволяє зімітувати ситуацію, яка може бути небезпечна для реального об'єкта управління, і, врешті-решт, дозволяє провести випробування системи управління для ще не створеного або недоступного об'єкта.

Сучасної тенденцією є використання для тестування та верифікації СЛУ програмних чи програмно-апаратних імітаторів об’єкта управління [1].

Магістерська атестаційна робота присвячена рішенню важливої науково-практичної задачі – програмно-апаратному імітаційному моделюванню систем логічного управління.

Вирішення цієї задачі потребує виконання ряду досліджень, а саме :

– аналізу особливостей алгоритмів управління в системах реального часу, мов їх опису та особливостей їх апаратної реалізації на базі мікроконтролерів з використанням автоматних моделей;

– аналіз процесу створення діагностичної моделі системи логічного управління як імітаційного моделювання;

– розглядання можливостей використання мікроконтролерів в якості інструмента для реалізації програмно-апаратного моделювання (hardware-in-the-loop);

– розробки програмно-апаратного комплексу імітаційного моделювання системи логічного управління.

1 СИСТЕМИ ЛОГІЧНОГО УПРАВЛІННЯ

Задачі, пов’язані з автоматизацією виробничих процесів, зводяться до створення систем управління машинами, агрегатами, верстатами, поточними лініями. Система – це сукупність взаємопов’язаних елементів, що становить певну цілісність, єдність. Управління – це сукупність цілеспрямованих дій, що включає оцінку ситуації та стану об’єкта управління, вибір керівних дій і їх реалізацію. Управління є процесом організації такого цілеспрямованого впливу на об’єкт, при якому об’єкт переходить в необхідний стан. Система управління – сукупність ланок, які здійснюють управління, і зв'язків між ними. У будь-якому процесі управління існує об'єкт, яким управляють (верстат, підприємство, галузь) і орган, який здійснює керування (технічний засіб, людина). У процесі управління цей орган отримує інформацію про стан зовнішнього середовища, де перебуває об'єкт і з яким він пов'язаний. Уся ця інформація сприймається управляючим органом, який виробляє на її основі управляючу інформацію (приймає рішення). На основі прийнятого рішення виконавчий орган здійснює управляючий вплив на об'єкт управління. Структурна схема системи управління наведена на рис. 1.1.

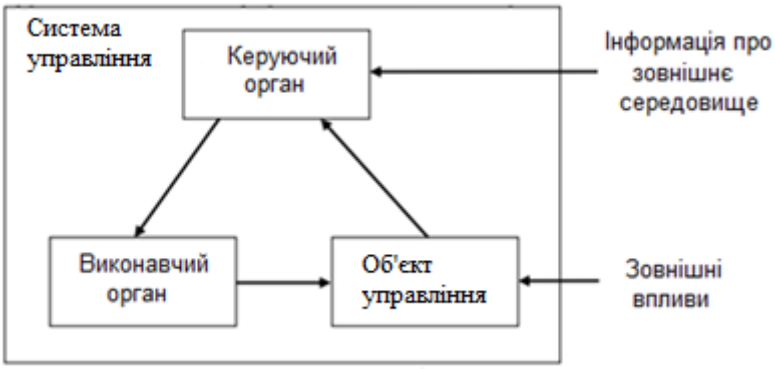


Рисунок 1.1 – Структурна схема системи управління

Об’єкт управління (ОУ) – умовно відокремлена частина системи, на яку

впливає система управління для досягнення необхідного результату. Управління завжди здійснюється для досягнення зазначеної мети, яка завжди конкретна для заданого об'єкта управління і пов'язана зі станом об'єкта та середовища, в якому він перебуває. У системах управління зворотний зв'язок визначається як інформаційний зв'язок, за допомогою якого в управляючу частину надходить інформація про наслідки управління об'єктом, тобто інформація про новий стан об'єкта, який виник під впливом управляючих дій.

Взаємодія об'єкта управління з пристроєм керування організовується за допомогою численних датчиків, розташованих в пристроях об'єкта і сигналізують про їх станах, про стан зовнішнього середовища, в якому функціонує технічна система [2]. Сигнали з датчиків надходять на вхід пристрою управління, яке має аналізувати поточну ситуацію і виробляти адекватну реакцію, посилаючи на компоненти об'єкта управління впливи, які безпосередньо керують виконавчими механізмами: перемикачами, клапанами, двигунами, насосами, електромоторами і т. д. Коли сигнали з датчиків та керуючі впливи є дискретними, а найчастіше, двійковими, то управління стає логічним. В такому випадку системи прийнято називати системами логічного управління (СЛУ).

СЛУ знаходять широке застосування не тільки на виробництві, але і на транспорті, в атомній енергетиці, мережах зв’язку, інформаційно-довідкових системах, комп’ютерах, у побутовій апаратурі і т.п. Перед тим як почати проектування СЛУ требі описати алгоритм рішення поставленої перед системою задачі, тобто алгоритм управління. Моделювання будь-якої керуючої системи починається з алгоритмічного опису процесу управління. Тільки після його опису, перевірки та відлагодження можна перейти до створення технічної системи, що реалізує цей алгоритм.

1.1 Особливості алгоритмів управління в системах реального часу

Найпоширеніший випадок керованого об'єкта – технічна система, що реалізує деяку виробничу технологію в реальному масштабі часу. Впливи – або, іншими словами, реакція системи управління, – визначається алгоритмом управління в залежності від подій на об'єкті управління, інформація про яких надходить через датчики зворотного зв'язку. Для цифрових систем ця обставина обумовлює циклічність керуючого алгоритму за схемою, вже обговорюваної раніше: «зчитування стану вхідних сигналів через датчики» – «їх обробка і формування вихідних сигналів» – «видача вихідних сигналів на виконавчі органи».

Алгоритми управління складними технологічними об'єктами мають ряд властивостей, специфічних для галузі промислової автоматизації [3].

ОУ належить до фізичного світу, підпорядковується об'єктивним законам фізики, хімії, біології. Якщо, скажімо, автоматизується зростання дріжджових паличок в автоклаві і для вирощування партії дріжджів потрібно 250 кВт год, то ця енергія повинна подаватися в систему рівномірно і протягом тривалого часу, близько двох діб. Наївна спроба подати цю енергію якомога швидше призведе до того, що мікроорганізми загинуть від перегріву. Так само і механічні елементи конструкцій, наприклад клапани, що також не можуть відпрацювати керуючий сигнал миттєво. А для передачі пакету даних потрібен час, залежне від довжини пакета і швидкості передачі.

Іншими словами, існують цілком об'єктивні часові вимоги до порядку видачі керуючих впливів. Таким чином, алгоритм управління передбачає синхронізацію свого виконання з фізичними процесами у зовнішньому середовищі. Це обумовлює необхідність розвиненою служби часу і активну роботу з тимчасовими об'єктами: затримками, паузами, тайм-аутами. Проблема синхронізації є однією з ключових в рамках так званих систем «реального» часу.

Інша характерна риса алгоритмів управління – логічний паралелізм, що відображає існування безлічі процесів в об'єкті управління, що протікають паралельно. Регулювання температури, тиску, швидкостей переміщення і обертання робочих органів технологічної системи за визначенням допускають і, більш того, припускають паралельність в роботі алгоритму управління. Оскільки події, що відбуваються в різних компонентах системи, виникають незалежно і в довільній послідовності, то спроба поставити реакцію системи єдиним блоком означає комбінаторний перебір великого числа варіантів і невиправдане зростання складності опису.

Логічний паралелізм передбачає наявність в алгоритмі управління незалежних або слабо залежних частин. Слід відразу зазначити, що паралелізм алгоритмів управління відрізняється від паралелізму високопродуктивних багатопроцесорних або багатомашинних обчислювальних систем. Паралелізм високопродуктивних систем, або так званий фізичний паралелізм, спрямований на скорочення часу отримання результату деякого обчислювального алгоритму і досліджується в рамках паралельного програмування.

Для систем автоматичного управління отримання результату в мінімальні терміни в переважній більшості практичних випадків неактуальна, паралелізм використовується для того, щоб спростити логічну структуру алгоритму управління.

Підсумувавши сказане можна виділити наступні властивості алгоритмів управління:

– відкритість – наявність «навколишнього середовища», зовнішнього світу, з яким взаємодіє алгоритм управління;

– подієвість – алгоритм управління формує керуючі впливи як реакцію на події (значущі зміни у вхідних даних), в тому числі на керуючі команди від оператора;

– невизначена тривалість функціонування алгоритму управління;

– синхронізм – необхідність синхронізації реакції алгоритму управління з подіями на об'єкті управління;

– логічний паралелізм – алгоритм управління структурно відображає паралелізм фізичних процесів на об'єкті управління, їх незалежність.

Реалізація алгоритмів управління коштами об'єктно-орієнтованих мов загального призначення чревата надмірним ускладненням програмної архітектури при зростанні складності алгоритму. Тому в області промислової автоматизації використовуються спеціалізовані мовні засоби для розробки алгоритмів управління: мови МЕК 61131-3 [4], Reflex [5].

1.2 Мови опису алгоритмів логічного управління

У 1993 році Міжнародна електротехнічна комісія випустила в світ третю частину стандарту МЕК 61131-3. Цей міжнародний стандарт входить в групу МЕК 61131-стандартів, які охоплюють різні аспекти програмування інформаційно-керуючих систем.

1. SFC (Sequential Function Chart) – графічна мова, що використовується для опису алгоритму у вигляді набору пов'язаних пар: кроків (step) і переходів (transition). Крок – набір операцій над змінними, перехід – набір логічних умовних виразів, що визначає передачу управління до наступній парі крок-перехід. За зовнішнім виглядом опис на мові SFC нагадує добре відомі логічні блок-схеми алгоритмів, хоча ідеологічно SFC близький до мереж Петрі. SFC має можливість розпаралелювання алгоритму, однак не має засобів для опису кроків і переходів, які можуть бути виражені лише засобами інших мов стандарту. Походження: мова Grafcet фірми Telemechanique-Groupe Schneider.

2. LD (Ladder Diagram) – графічна мова, стандартизований варіант класу мов релейно-контактних схем (РКС). Основний виразний елемент – «реле». Реле представлено парою: обмотка (аналог вихідної змінної) і контакт (аналог вхідної змінної). Контакти і обмотки між собою і з лівої, і з правої силовий

шиною з'єднані горизонтальними лініями («провідниками»). РКС були дуже потужним засобом автоматизації в докомп'ютерну епоху.

Найсильніші школи теоретиків автоматизації на базі РКС існували в СРСР, одна з них – школа Гаврилова М. О.. РКС були настільки популярні в автоматизації, що з появою ПЛК ідеї, закладені в РКС, були реалізовані у вигляді спеціалізованої мови програмування. Це виявилося дуже зручно при впровадженні ПЛК: старі кадри дуже швидко освоювали мову програмування, оскільки концептуальний рівень (метафора реле) залишився колишнім.

З огляду на своїх обмежених можливостей мову доповнений привнесеними засобами: таймерами, лічильниками і т. П. Походження: різні варіанти мови релейно-контактних схем, підтримувані в компаніях Allen-Bradley, AEG Schneider Automation, GE-Fanuc, Siemens.

3. FBD (Functional Block Diagram) – графічна мова за своєю суттю схожий на LD: замість реле в цій мові використовуються функціональні блоки. Алгоритм роботи деякого пристрою, виражений засобами цього мові, нагадує функціональну схему електронного пристрою: елементи типу «логічне І», «логічне АБО» і т. П., З'єднані лініями. Коріння мови з'ясувати складно, проте більшість фахівців сходяться на думці, що це мова, подібний LD, але використовує в якості метафори іншу елементну базу.

4. ST (Structured Text) – текстовий високорівнева мова загального призначення, по синтаксису орієнтований на Паскаль. Самостійного значення не має; використовується спільно з SFC. Походження: мова Паскаль, мова Grafcet фірми Telemechanique-Groupe Schneider.

5. IL (Instruction List) – текстова мова низького рівня. Виглядає як мову Асемблера, що пояснюється його походженням: для деяких моделей ПЛК фірми Siemens є мовою Асемблера. В рамках стандарту IEC 1131-3 він не прив'язаний до архітектури конкретного процесора. Самостійного значення не має: використовується спільно з SFC. Походження – мова STEP 5 фірми Siemens.

Незважаючи на значне спрощення конструкцій і семантики, дані мови не можуть бути віднесені до класу технологічних, оскільки в них широко використовуються «нетехнологічні» терміни і прийоми опису алгоритмів: наприклад, програми LD оперують поняттями, пов'язаними із елементами релейної апаратури (тип контакту, тип з'єднання, котушка реле і т.д.), і вимагають від розробника володіння методами побудови застарілих релейних систем; програмування на мові SFC вимагає знань абстрактного математичного апарату мереж Петрі, використання текстових мов ST і IL вимагає знань основ універсальних мов програмування і т.п.

В даний час багато фахівців вказують на недоліки стандарту і необхідність розробки альтернатив [6].

1.3 Проектування систем логічного управління на базі мікроконтролерів

Стандарт МЭК61131-3 добре зарекомендував себе при розробці програмного забезпечення програмованих логічних контролерів, але для програмування мікроконтролерів він не дуже пристосований. Це пов’язано з тим, що мікроконтролерні системи часто більш складні та передбачають більш гнучке використання апаратних ресурсів, ніж дозволяє стандарт.

1.3.1 Автоматна модель

Автоматизація проектування програмного забезпечення систем управління, в тому числі на базі мікроконтролерів, базується на використанні підходу, який отримав назву «модельно-орієнтоване проектування» (Model-Based design, MBD) [7]. Використання цього підходу при створенні систем логічного управління призвело до розвитку технології автоматного програмування [8-11].

Суть автоматного програмування полягає в поданні систем зі складною поведінкою у вигляді автоматизованих об'єктів управління, які складаються з

керованої і керуючої частин. Керована частина складається з об'єкта управління, який відповідає за виконання дій, обраних для виконання керуючої частиною, і, можливо, за формування деяких вхідних впливів для керуючої частини – зворотних зв'язків. Керуюча частина відповідає за логіку поведінки – вибір виконуваних дій, що залежить від поточного стану і вхідних впливів, а також за перехід в новий стан. Керуюча частина являє собою керуючий автомат або систему взаємодіючих автоматів.

Програми, в рамках цієї технології, створюються на основі формальної моделі кінцевого автомата – Finite State Machine (FSM). Однак класичне графічне представлення кінцевих автоматів страждає рядом недоліків. Головним недоліком є відсутність поняття часу. Простим розширенням моделі класичного кінцевого автомата є введення поняття «зовнішня подія», наступ якої можна вважати умовою переходу автомата зі стану в стан (рис. 1.2).

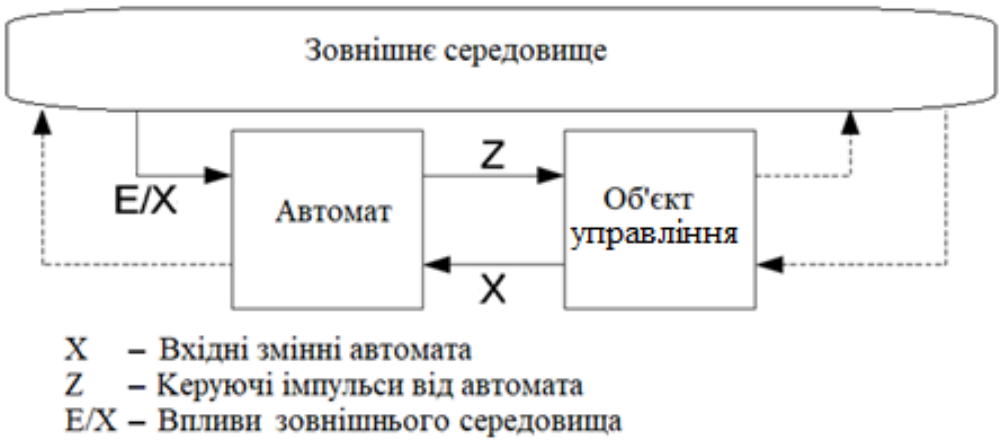


Рисунок 1.2 – Використання автоматного підходу до проектування СЛУ

Такими подіями можуть бути отримання на вхід автомата символу або ж ціле повідомлення, переривання, подія спрацьовування таймера.

З цим останнім типом подій природно зв'язується поняття часу в автоматі. Дійсно, введення поняття часу найпростіше зв'язати з обмеженням часу перебування автомата в конкретному стані і таке обмеження задати

таймером. Подія спрацьовування таймера викличе перехід автомата в інший стан.

Автоматні програми строго структуровані і в них виділено три види функцій: функції переходів, функції виходів, функції реалізації затримок і переходу в новий стан. Вони будуються згідно зі строгим шаблоном з використанням операторів багатопозиційного вибору (switch, case), умовних операторів (if, select) і функцій реалізації таймера або фронту (синхросигналу Clk). Для кодування автоматних програм можна використовувати будь-які мови програмування, такі як C, JScript та ін.

Одним з головних достоїнств автоматного програмування є уявлення логіки поведінки системи в найбільш зрозумілому і наочному вигляді – з використанням графічного представлення автоматів у вигляді графів переходів.

1.3.2 Типова структура систем логічного управління на базі мікроконтролера

Будь-яка система управління, до складу якої входить мікроконтролер (МК), у процесі роботи виконує ряд узагальнених функцій для досягнення цілей, що закладають при проектуванні (функціональний рівень проектування) [12]. До таких функцій належать:

– збір інформації про керований об’єкт (процес), збурюючі впливи, стан зовнішнього середовища й апаратури, що входить до складу системи;

– перетворення отриманої інформації до вигляду, зручного для вводу в МК;

– використання різних каналів зв’язку для передачі інформації до МК від датчиків, що вимірюють стан та параметри процесів, і від МК до споживачів інформації;

– обчислення керуючих впливів за заданим алгоритмом, реалізованим у вигляді прикладного програмного забезпечення.

Відповідно до узагальнених функцій може бути подана й узагальнена структура СЛУ, що складається з об’єкта управління, МК та апаратури їх взаємного зв’язку (рис. 1.3).

До складу МК входять постійний та оперативний запам’ятовуючі пристрої (ПЗП та ОЗП), ядро МК, котре об’єднує арифметико-логічний пристрій з регістрами та службовими блоками, порти вводу/виводу, блок стандартних інтерфейсів (БСІ), блок таймерів – лічильників (Т).

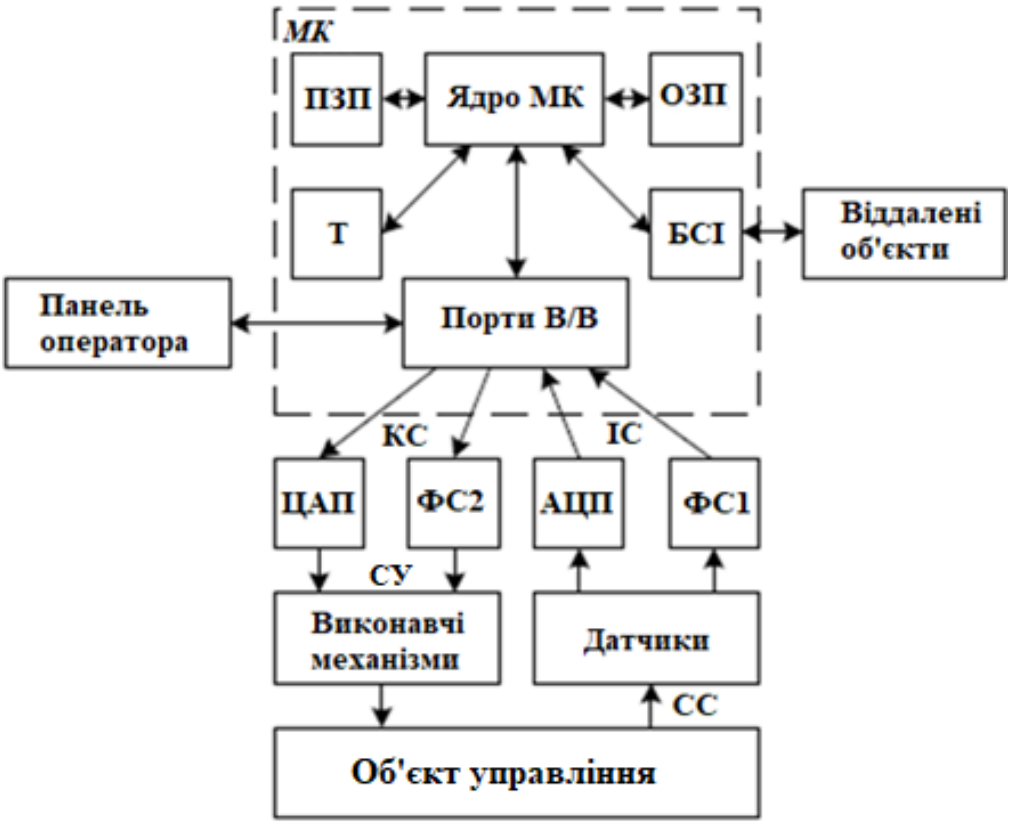


Рисунок 1.3 – Узагальнена структуру СЛУ на базі МК

Різноманітні за природою сигнали стану (СС) об’єкта управління надходять на датчики, які перетворюють їх на електричні сигнали, що мають як аналоговий, так і дискретний характер. Аналогові сигнали перетворюються аналого-цифровими перетворювачами (АЦП) до цифрового вигляду, імпульсні сигнали надходять на схеми формувачів сигналів (ФС), які виконують функції формування потрібних рівнів двійкових сигналів

(найчастіше ТТЛ-рівня), гальванічної розв’язки тощо. Сигнали з виходів АЦП та ФС являють собою інформаційні слова (ІС).

МК шляхом опитування інформаційних слів (періодичного чи за встановленим законом) отримує інформацію про об’єкт та генерує відповідно до алгоритму управління послідовності керуючих слів (КС). МК з необхідною періодичністю оновлює керуючі слова на своїх вихідних портах. Певна частина керуючого слова інтерпретується як сукупність прямих двійкових сигналів управління (СУ), які через схеми формувачів сигналів (підсилювачі потужності, реле, оптрони тощо) надходять на виконавчі механізми.

Інша частина керуючого слова являє собою двійкові коди, які через цифро-аналогові перетворювачі (ЦАП) впливають на виконавчі механізми аналогового типу. Як правило, до складу МКС входить панель оператора, на якій знаходяться пристрої вводу інформації та елементи її відображення. З віддаленими об’єктами, до яких належать інтелектуальні датчики, аналогічні мікроконтролерні системи та керуючі ЕОМ, СЛУ спілкується за допомогою стандартних інтерфейсів: універсальних асинхронних RS232 та RS485, синхронних I2C і SPI, USB та ін., які апаратно підтримуються БСІ. Формування часових інтервалів апаратними таймерами-лічильниками дозволяє не займати ресурси ядра МК. До складу МК також можуть входити ЦАП, АЦП та деякі ФС.

1.3.3 Мікроконтролер Arduino

Arduino є цілим сімейством плат – багатофункціональних мікроконтролерів. Найбільш поширеними і найчастішими у використанні сьогодні є плати Arduino Uno, Nano, Mega, PRO mini. Область використання цих мікроконтролерів дуже широка: від системи «розумний дім» та робототехніки до охоронних систем та метеостанцій. Це пояснюється тим, що за допомогою цього пристрою можливо керувати датчиками руху, температури або освітлення, двигунами і т.п.

Розглянемо детальніше одного з представників цього сімейства, а саме Arduino Uno (рис. 1.4), що є платою на основі мікроконтролера ATmega328.

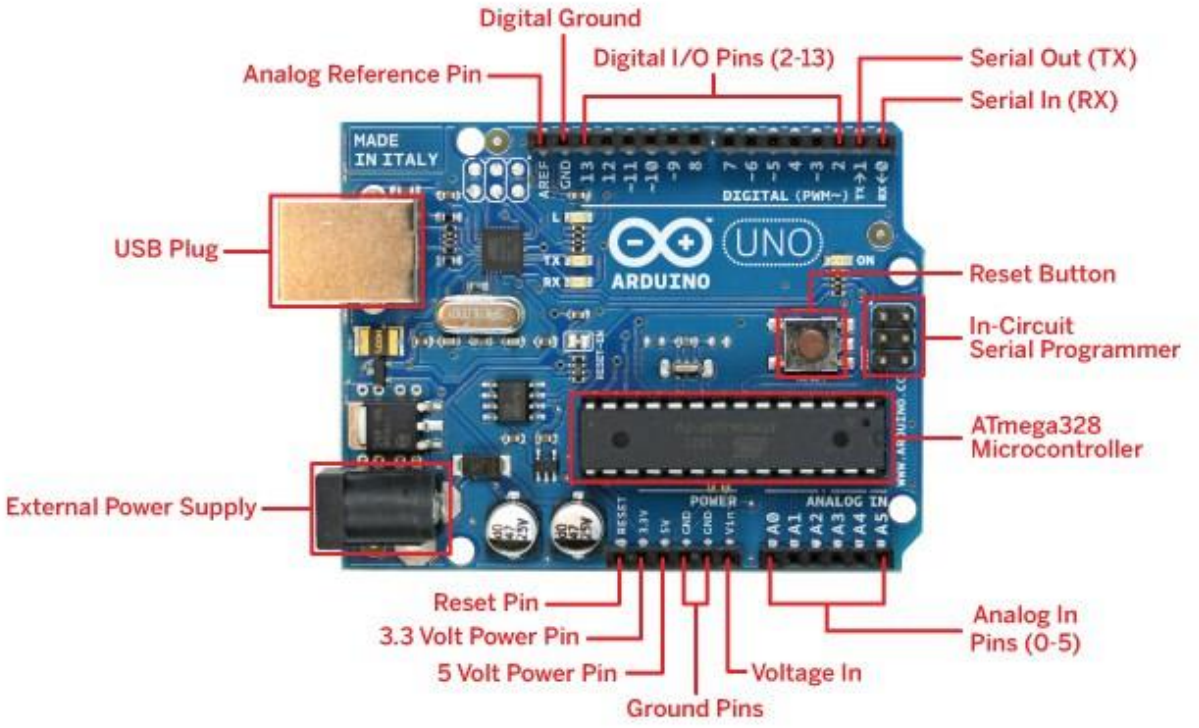


Рисунок 1.4 – Опис елементів плати Arduino Uno R3

До його складу входить усе необхідне для зручної роботи з мікроконтролером: 14 цифрових входів / виходів (з них 6 можуть використовуватися в якості ШІМ-виходів), 6 аналогових входів (наприклад для зчитування поточних даних з датчика температури), кварцовий резонатор на 16 МГц, роз'єм USB, роз'єм живлення, роз'єм для внутрішньо-схемного програмування (ICSP) і кнопка скидання.

Для початку роботи з платою досить просто дати живлення від AC/DC-адаптера (батарейки) або підключити його до комп'ютера за допомогою USB-кабелю.

У таблиці 1.1 представлені детальні характеристики цієї плати.

Таблиця 1.1 – Технічні характеристики Arduino Uno

Мікроконтролер	ATmega328
Робоча напруга	5В
Рекомендована напруга	7-12В
Максимальна напруга	6-20В
Цифрові входи/виходи	14 (з них 6 можуть використовуватися в якості ШІМ-виходів)
Аналогові входи	6
Максимальний струм одного виходу	40 мА
Максимальний вихідний струм виходу на 3.3V	50 мА
Flash-пам’ять	32 КБ (ATmega328) з яких 0.5 КБ використовуються завантажувачем
SRAM	2 КБ (ATmega328)
EEPROM	1 КБ (ATmega328)
Тактова частота	16 МГц

Піни Arduino використовуються для підключення зовнішніх пристроїв і можуть працювати як в режимі входу (INPUT), так і в режимі виходу (OUTPUT). До кожного входу може бути підключений вбудований резистор 20-50 кОм за допомогою виконання команди pinMode () в режимі INPUT_PULLUP. Допустимий струм на кожному з виходів – 20 мА, не більше 40 мА на піку. Для зручності роботи деякі піни поєднують в собі кілька функцій.

- Піни 0 і 1 – контакти UART (RX і TX відповідно).
- Піни с 10 по 13 – контакти SPI (SS, MOSI, MISO і SCK відповідно).
- Піни A4 і A5 – контакти I2C (SDA і SCL відповідно).

Піни з номерами від 0 до 13 є цифровими. Це означає, що є можливість зчитувати і подавати на них тільки два види сигналів: HIGH і LOW. За допомогою ШІМ також можна використовувати цифрові порти для управління

потужністю підключених пристроїв. Аналогові піни Arduino Uno призначені для підключення аналогових пристроїв і є входами для вбудованого десятирозрядного аналого-цифрового перетворювача (АЦП). На рисунку 1.5 представлено схему розташування пінів Arduino Uno R3.

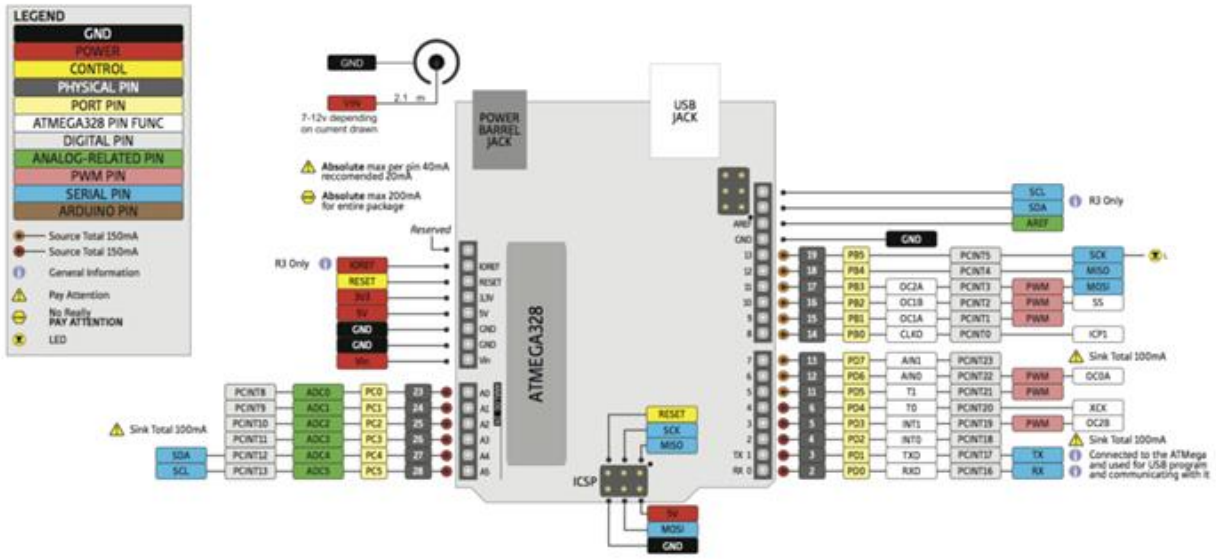


Рисунок 1.5 – Схема розташування пінів Arduino Uno R3

В свою чергу мікроконтролер ATMega328 має наступні піни (рис. 1.6).

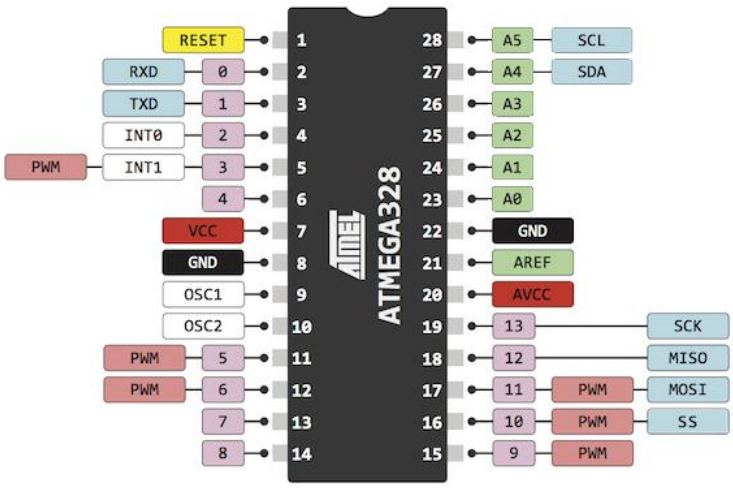


Рисунок 1.6 – Схема розташування пінів мікроконтролера ATMega328

Середовище розробки, Arduino IDE, складається із вбудованого текстового редактора програмного коду, області повідомлень, вікна виведення тексту (консолі), панелі інструментів декількох меню (рис.1.7). Для завантаження програм і зв'язку середовище розробки підключається до апаратної частини Arduino.

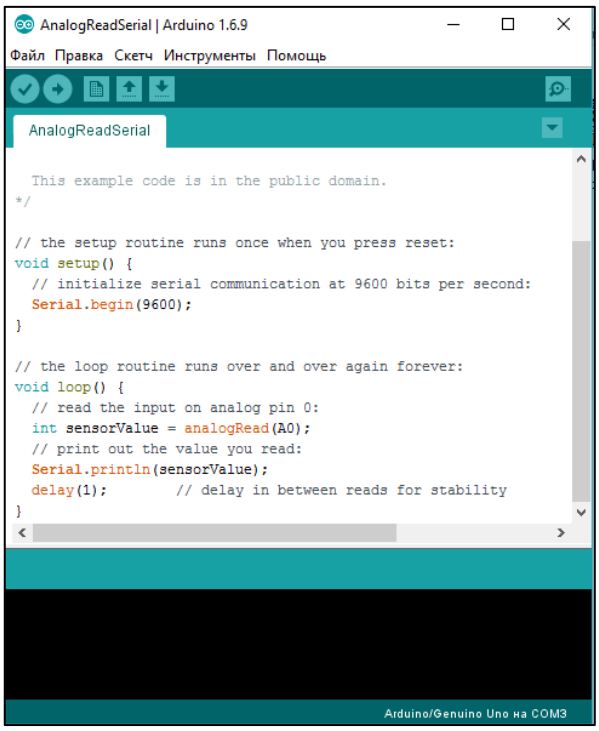


Рисунок 1.7 – Загальне представлення середовища розробки Arduino IDE

Програма, яка написана у середовищі Arduino, називається скетч. Мова програмування, що використовується для Arduino, дуже схожа на CI++, доповнений деякими бібліотеками. Обробка програм здійснюється за допомогою препроцесора, а компілюється за допомогою AVR-GCC.

Скетч пишеться в текстовому редакторові, що має інструменти вирізки/вставки, пошуку/заміни тексту. Під час збереження і експорту проекту в області повідомлень з'являються пояснення, також можуть відображуватися помилки, які виникають.

Після того як плату до комп'ютера було підключено, потрібно здійснити наступні налаштування – обрати плату Arduino Uno та COM-порт, по якому здійснюється з'єднання з ПК (рис. 1.8). Практично завжди налаштування порту

відбувається автоматично.

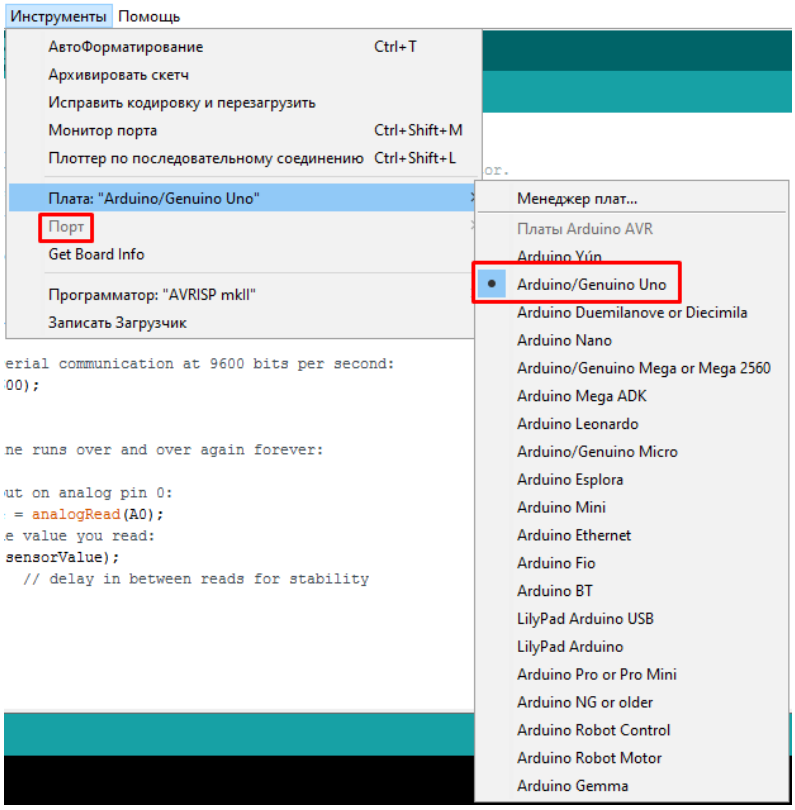


Рисунок 1.8 – Налаштування підключення плати до ПК

Ця директива – деяка інструкція, файл з описом об'єктів, функцій, і констант бібліотеки. Багато функцій вже розроблені для більшості типових задач та для більш автоматизованої розробки, сутність якої полягає у зменшенні об’єму опису роботи пристрою у вигляді коду. Тобто є можливість якомога більше задіяти вже розроблений функціонал для виконання необхідних задач. Додаткова функціональність розробки пристрою може бути додана за допомогою бібліотек, що представляють собою оформлений спеціальним чином код. В основному він знаходиться в закритому від розробника доступі. Середа розробки зазвичай поставляється зі стандартним набором бібліотек, який можна поступово поповнювати. Вони знаходяться в підкаталозі libraries каталогу Arduino (рис. 1.9). Багато бібліотек забезпечуються прикладами, розташованими в папці example. Вибір бібліотеки в меню призведе до додавання до початкового коду рядка #include

<назва бібліотеки[.h]>.

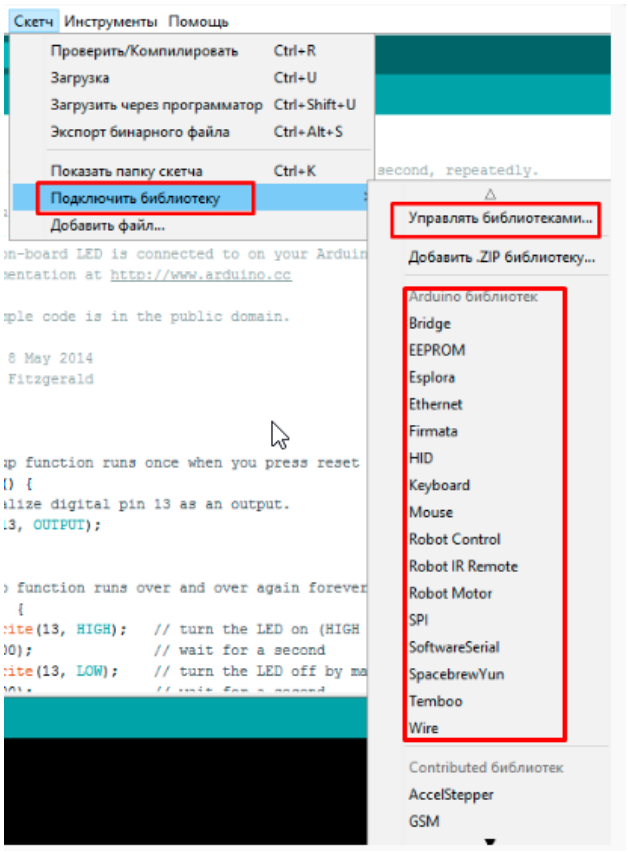


Рисунок 1.9 – Менеджер бібліотек

Таким чином, розглянувши плату Arduino UNO можна сміливо сказати, що вона відповідає вимогам, які висуваються до апаратної реалізації систем управління, а саме дешевизна, простота її розширення. Останнє можливе завдяки тому, що до плата Arduino можна підключити плату розширення.

1.4 Постановка цілі та задач дослідження

Розробка системи логічного управління на апаратному рівні до недавнього часу базувалася на спеціалізованих модульних обчислювальних пристроях – програмованих логічних контролерах (ПЛК), мови опису яких розглядались у розділі 1.2. На сьогоднішній момент проектування відходить від жорсткої прив'язки програмного та математичного забезпечення систем до

апаратури. У зв'язку з цим з'являється можливість використовувати в якості апаратної обчислювальної платформи сучасні уніфіковані рішення (персональні, планшетні, одноплатні комп'ютери і т.д.), які дозволяють застосовувати стандартний інструментарій проектування та реалізації математичного забезпечення систем управління (наприклад, середовища програмування Visual Studio, Code Blocks, IDE Arduino).

Традиційний процес проектування систем логічного управління передбачав поетапне створення системи, де процес тестування виконувався на пізніх циклах проектування [13]. Тільки на цьому етапі інженери мали змогу спостерігати за взаємодією між об’єктом та системою управління. Коли останні стали більш складнішими та непередбачуваними, традиційний підхід поступився місцем більш сучасному – модельно-орієнтованому проектуванню.

Тестування перестало розглядатися як фінальний етап. Воно становиться безперервним процесом, який починається з моделювання системи та переходить до тестування у реальному часі (імітаційного моделювання). Модельно-орієнтоване проектування дозволяє скоротити та зробити більш дешевим процес проектування, а також створювати більш надійні системи управління.

Аналіз сучасного стану СЛУ показав актуальність обраної теми магістерської роботи та поставленої мети, а саме розробки програмно-апаратного комплексу імітаційного моделювання систем логічного управління реального часу на базі мікроконтролерів.

Для досягнення поставленої мети необхідно вирішити такі задачі:

– проаналізувати особливості алгоритмів управління в системах реального часу та мови їх опису;

– розглянути особливості апаратної реалізації СЛУ на базі мікроконтролера;

– розглянути моделі діагностичного забезпечення систем логічного управління;

– проаналізувати особливості реалізації часових параметрів на МК;

– обґрунтувати доцільність використання плати Arduino UNO в якості апаратної платформи для імітаційного моделювання, а також логічного аналізатору;

– запропонувати діагностичну модель СЛУ на базі двох мікроконтролерів та розробити програмно-апаратний комплекс імітаційного моделювання системи логічного управління;

– виконати діагностичний експеримент з перевірки коректності запропонованої системи логічного управління автоматом з продажу напоїв.

2 ДІАГНОСТИЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМ ЛОГІЧНОГО УПРАВЛІННЯ

Діагностичне забезпечення – комплекс взаємопов'язаних правил, методів, алгоритмів і засобів, необхідних для здійснення діагностування на всіх етапах життєвого циклу об'єкта. Для того щоб об'єкт був пристосований до діагностування, необхідно при його проектуванні розробляти діагностичне забезпечення. Діагностичне забезпечення проектованого об'єкта отримують в результаті аналізу його діагностичної моделі. Будується діагностична модель на основі передбачуваної конструкції, умов використання та експлуатації об'єкта. В результаті дослідження діагностичної моделі встановлюються діагностичні ознаки, прямі і непрямі параметри і методи їх оцінки, визначають умови працездатності, розробляють алгоритми діагностування. Діагностичне забезпечення дає можливість отримати достовірну інформацію про працездатність об'єкта діагностування (ОД).

Методика діагностування ОД можна представити у вигляді структурно-логічної схеми (рис. 2.1), що представляє собою два блоки – попередній та експлуатаційний етапи. У блоці формування еталонів відбувається складання матриць структурних параметрів працездатного технічного стану та отриманих значень діагностичних ознак (ДО). Блок формування діагностичних моделей призначений для обробки випробувальних даних, розрахунку регресійних залежностей зміни обраних ДО від зміни представлених структурних параметрів, розрахунку адекватності отриманих моделей із заданим рівнем значущості і формування імовірнісних моделей процесів деградації. У блоці формування порогових значень ДО відбувається складання матриць ДО з урахуванням отриманих граничних значень структурних параметрів.

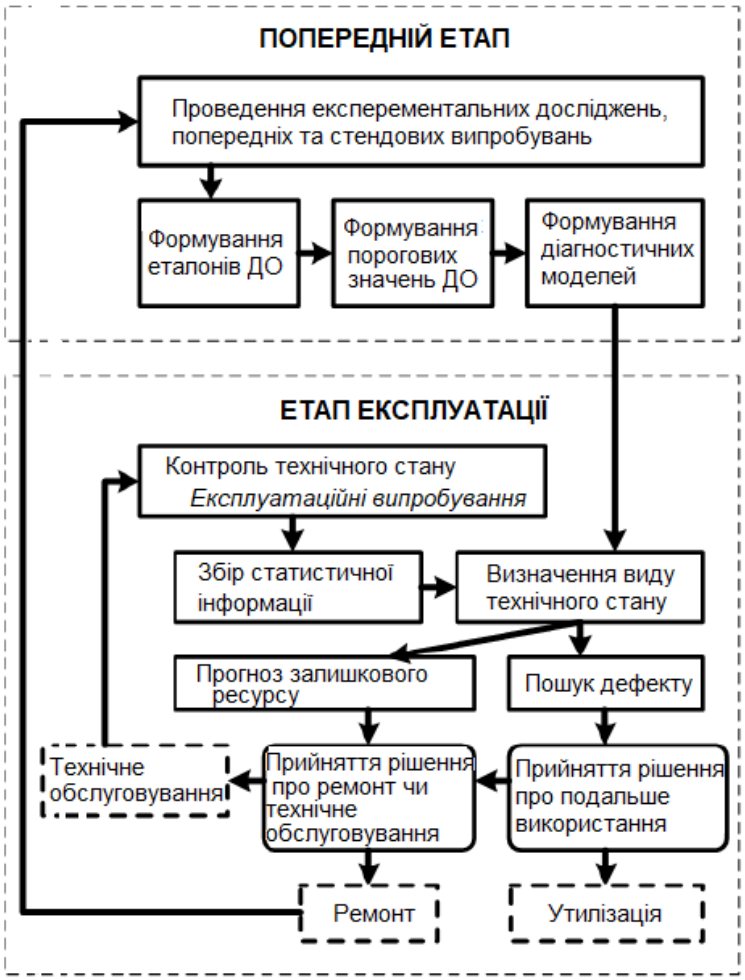


Рисунок 2.1 – Структурно-логічна методика діагностування

Отримані ДО та ДМ складають базу даних для подальшого порівняння з експлуатаційними параметрами. На підставі порівняння поточних, еталонних і порогових ДО в блоці визначення виду технічного стану відбувається видача інформації про поточний стан ОД. Під час експлуатації ОД проводиться набір статистичної інформації за результатами контролю технічного стану і експлуатаційних випробувань для прогнозування залишкового ресурсу елементів ОД на основі імовірнісних моделей процесів деградації елементів ОД.

Якщо при визначенні виду технічного стану ДО не виходять за межі порогових значень, то проводиться прогноз залишкового ресурсу ОД, за результатами якого може бути проведено технічне обслуговування або

обґрунтовано проведення ремонту. В іншому випадку необхідно прийняти рішення про подальше використання ОД.

Саме попередній етап створення системи діагностування є найбільш складним та відповідальним. В центрі системи діагностування лежить діагностична модель (ДМ) – формалізований опис об'єкта, необхідний для вирішення завдань діагностування. Класифікація діагностичних моделей детально розглянута у [14].

Процес створення діагностичної моделі називається моделюванням.

Комп'ютеризація сучасного життя, зокрема, широке поширення комп'ютерів в галузі фізико-технічних досліджень зробила революцію в моделюванні. Спочатку використання ЕОМ практично витіснило аналітичні методи дослідження за рахунок широкого впровадження в практику обчислювальних методів, а потім і зовсім призвело до зміни базового дослідницького підходу на модельно-орієнтоване проектування (МОП). Замість фізичних прототипів і текстових специфікацій в модельно-орієнтованому проектуванні застосовується модель.

Підхід МОП заснований на застосуванні концепції «in-the-loop testing» [15]. Існують наступні основні способи тестування на базі підходу МОП, в яких в якості об'єкта тестування може виступати: модель – Model-in-the-loop (MiL), програмне забезпечення – Software-in-the-loop (SiL), прототип пристрою – Processor-in-the-loop (PiL), готовий пристрій – Hardware-in-the-loop (HiL).

2.1 Діагностична модель системи логічного управління

Беручи до уваги той факт, що в якості ОД виступає СЛУ реального часу, то її коректність функціонування напряму залежить від дотримання часових обмежень. Отже найбільш придатним методом моделювання в такому випадку є імітаційне.

При імітаційному моделюванні алгоритм, який реалізує модель, відтворює процес функціонування системи в часі. Причому імітуються елементарні явища, що становлять процес, зі збереженням їх логічної структури і послідовності протікання в часі. Це дозволяє за вихідними даними отримати відомості про стан процесу в певні моменти часу, що дають можливість оцінити характеристики системи.

Основною перевагою імітаційного моделювання в порівнянні з аналітичним є можливість вирішення більш складних завдань. Імітаційні моделі дозволяють досить просто враховувати такі фактори, як наявність дискретних і безперервних елементів, нелінійні характеристики елементів системи, численні випадкові впливу та ін., які часто створюють труднощі при аналітичних дослідженнях. В даний час імітаційне моделювання – найбільш ефективний метод дослідження складних систем, а часто і єдиний практично доступний метод отримання інформації про поведінку системи, особливо на етапі її проектування [2].

Розробка систем логічного управління вимагає виконання тестів і перевірок вже на етапі проектування і розробки елементів системи. Виявлені на цих етапах недоліки і помилки можуть бути виправлені з мінімальними додатковими витратами. Складність комплексної налагодження і тестування програм управління полягає в трудомісткості штучного формування повного набору узгоджених сигналів технологічного обладнання. Для розробки, налагодження і тестування прикладного програмного забезпечення систем управління застосовують такі засоби, як програмні та фізичні імітатори сигналів та інтерфейсів.

Створення повномасштабних імітаторів керованого об'єкта в «залізі» передбачає використання поруч із системою управління додаткового контролера з набором модулів зв’язку з об’єктом, що за сигналами збігається з входами / виходами керованого об'єкта. Спеціально створений алгоритм, виконуваний на контролері, реалізує фізичні процеси на керованому об'єкті. Вартість такого імітатора приблизно дорівнює вартості системи управління.

Якщо ж кошти на налагодження обмежені, то створюються ручні імітатори, що підключаються лише до входів системи управління. У цьому випадку поведінка керованого об'єкта задається положенням ручних перемикачів і змінних резисторів.

Широке поширення підходу, заснованого на ідеї програмної імітації досліджуваного об'єкта, дозволяє ряду дослідників заявляти про настання "ери комп'ютерного моделювання" [16]. Візуалізація – невід'ємний атрибут програмної імітації – радикально спрощує роботу дослідника, приховуючи багатоповерхові формули математичної моделі за наочністю елементів комп'ютерної графіки. Програмна імітація дозволяє:

– маніпулювати перебігом модельного часу, прискорювати повільні процеси і сповільнювати швидкоплинні, підвищуючи ефективність сприйняття дослідником поведінки системи;

– змінювати параметри запуску моделі в широкому діапазоні, тим самим надаючи досліднику потужний евристичний інструмент, щоб розробляти і висувати нові гіпотези, моделі, теорії;

– істотно скоротити число натурних експериментів, у багатьох випадках замінюючи їх чисельними, що особливо важливо в разі ризиків (наприклад, аварійної зупинки виробництва при дослідженні систем з прихованими зв'язками) або етичних наслідків (наприклад, при вирішенні задачі оптимізації числа робочих місць);

– вирішувати педагогічні завдання, зокрема, скорочуючи терміни і підвищуючи якість підготовки персоналу (операторів, технологів);

– спростити аналіз взаємозалежностей, надаючи можливість настройки, спрощення моделі, виключення тривіальних параметрів.

Спочатку в якості засобів створення програмних імітаторів використовувалися універсальні мови програмування (Algol, Fortran), пізніше почали з'являтися спеціалізовані мови програмування (GPSS, Simula). Однак подібні засоби не можуть претендувати на роль промислового інструменту,

так як складність освоєння, трудомісткість моделювання, слабкі засоби візуалізації результатів сильно обмежують їх застосування для вирішення інженерних задач в порівнянні зі спеціалізованими пакетами візуального програмування і бібліотеками типових елементів.

2.2 Програмно-апаратний комплекс імітаційного моделювання

Більш сучасний підхід, що застосовується на практиці, – використання в якості імітатора окремого програмно-апаратного комплексу (ПАК) [17], "комплиментарного" створюваній системі управління (рис. 2.2).

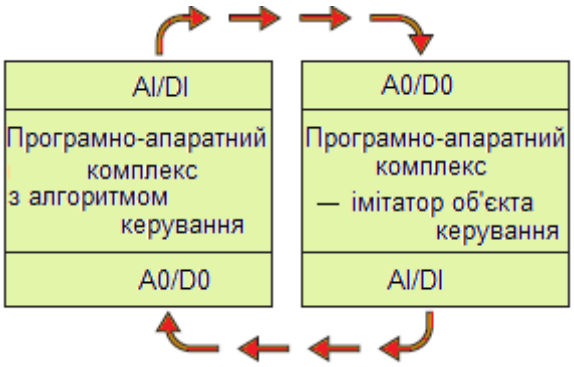


Рисунок 2.2 – Схема налагодження алгоритму управління з використанням програмно-апаратного імітатора (AI – аналогові входи, DI – цифрові входи, AO – аналогові виходи, DO – цифрові виходи)

Такий цифровий імітатор містить програму, яка б відтворювала поведінку об'єкта управління, і формує вхідні аналогові / цифрові сигнали для системи управління, що відгладжується, через свої модулі виходів на підставі керуючих сигналів, що зчитуються через аналогові та цифрові модулі входів.

На відміну від фізичного імітатора така схема дозволяє тестувати алгоритми управління для широкого спектра об'єктів і змінювати поведінку модельованого об'єкта. В силу згадуваної компліментарності системи управління та об'єкта підхід дозволяє також застосовувати вже використовувані мови програмування, що дає очевидну перевагу в порівнянні

зі спеціалізованими засобами комп'ютерної імітації. Ще одна перевага підходу – проста комутація алгоритму від імітатора до реального об'єкту управління.

Однак програмно-апаратне моделювання передбачає додаткові витрати на технічні засоби, що приблизно дорівнюють витратам на апаратуру системи управління. Рішенням цієї проблеми може бути використання звичайних мікроконтролерів в якості апаратного імітатора за принципом hardware-in-the-loop [18].

Загальна структурна схема запропонованого ПАК імітаційного моделювання представлена на рисунку 2.3.

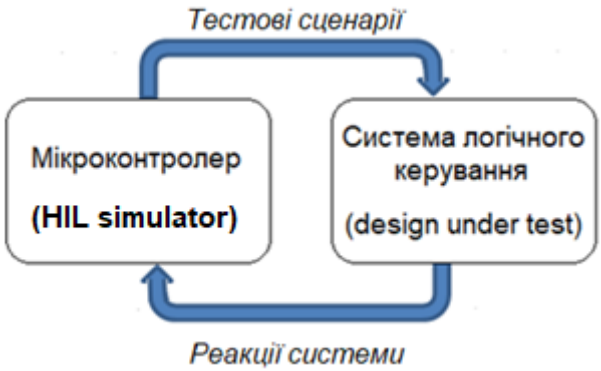


Рисунок 2.3 – Загальна структура програмно-апаратного комплексу імітаційного моделювання

Design under test (DUT) – це модель системи управління у вигляді кінцевого часового автомату. Ця модель реалізується на мікроконтролері згідно з принципами автоматного програмування [19]. HIL simulator – це модель, реалізована на мікроконтролері, що імітує процес подання на СЛУ тестових впливів. В ході діагностування імітатор подає на DUT тестові впливи згідно із сценарієм тестування, що імітують той и інший стан об’єкта управління. В свою чергу стан ОУ залежить не тільки від зовнішніх впливів, але й від показників внутрішніх датчиків.

За умови присутності такого МК зі всіма інтерфейсними модулями, відлагодження за принципом Hardware-in-the-loop полягає у підключенні

входів та виходів через додаткові модулі входів/виходів до робочої станції, на якій виконуються задачі (рис. 2.4):

– формування тестових сценаріїв;

– подача на систему, що тестується, сигналів та емулюючих сигналів зворотнього зв'язку (інформативні сигнали від ОУ);

– зчитування керуючих сигналів, що формуються МК у відповідності зі своєю програмою;

– моделювання об'єкта управління і розрахунок його миттєвих значень вихідних сигналів;

– контроль коректності роботи керуючих програм МК та формування повідомлень для оператора.

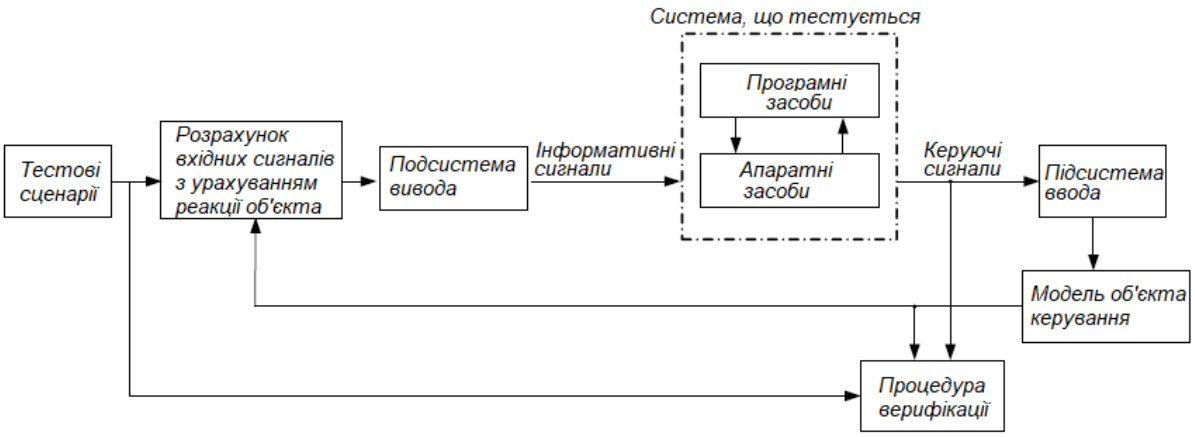


Рисунок 2.4 – Принцип Hardware-in-the-loop

Така імітаційна модель бере до уваги особливості предметної області, а саме: параметри технологічного обладнання та його складових частин (насосів, двигунів і т. п.), проектні дані про конфігурацію технологічного обладнання (установки, набір датчиків і сигналів управління), параметри реальних технологічних процесів. Основне завдання імітаційного програмно-апаратного комплексу – це створення середовища функціонування контролерів повністю ідентичного реальному з точки зору зовнішніх сигналів.

Для реалізація апаратної частини комплексу було обрано плату Arduino на базі мікроконтролера Atmega128, особливості якої було розглянуто у частинs 1.3.3, а також логічний аналізатор Saleae Logic.

Логічний аналізатор (ЛА) – є багатоканальним діагностичним обладнанням, призначеним для високошвидкісного збору даних про поведінку досліджуваної дискретної системи та подання цих даних в зручній для сприйняття людиною формі [20]. Він може бути дуже корисний при розробці та налаштування електронних пристроїв, при написанні програмного забезпечення, що працює в зв'язці з «залізом», при роботі з мікроконтролерами, ПЛІС і мікропроцесорами, для аналізу роботи різних пристроїв і протоколів обміну даними, і т.п.

Логічні аналізатори характеризуються числом каналів, ємністю пам’яті на канал, частотою запису, способами синхронізації і запуску, формами представлення даних. Для визначення значень сигналів ЛА використовують компаратори, за допомогою яких з'ясовується, вище або нижче вхідний сигнал заданого порогового рівня. Якщо сигнал перевищує поріг, його рівень визначається як високий, якщо нижче порога, то низький (рис. 2.5).

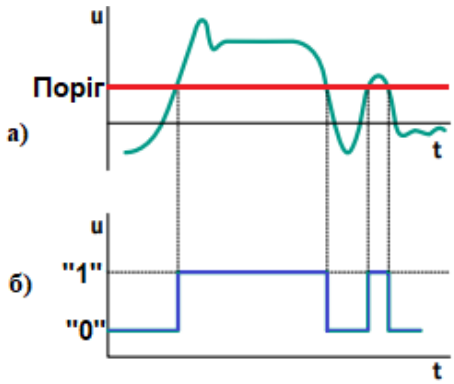


Рисунок 2.5 – Вид сигналів на вході (а) та на виході (б) компаратора

Гранична напруга компараторів, як правило, може регулюватися (в основному від -10 до +10 В). Підключення ЛА вимагає доступу до ділянкам цифрового пристрою (наприклад, виводів мікросхем або трасах плати).

Аналізатори забезпечуються спеціальними щупами та зажимами. Основна вимога – мінімізація впливу приладу на випробний пристрій.

Розглянемо особливості логічного аналізатору Saleae Logic (рис. 2.6).

Рисунок 2.6 – Логічний аналізатор Saleae Logic

У таблиці 2.1 перераховані основні параметри логічного аналізатора фірми Saleae.

Таблиця 2.1 – Основні параметри логічного аналізатора фірми Saleae

Параметр	*Значення*
Число цифрових каналів	8
Частота оцифровки на канал	до 24 МГц
Параметр	Значение
Кількість семплів в вибірці	до 1G (зависит от количества памяти ПК)
Вхідний опір	100 кОм
Діапазон робочих напруг	–0,5…5,25 В
Напруга логічного «0»	–0,5…0,8 В
Напруга логічної «1»	2,0…5,25 В
Захист від статики	
Захист по перевищенню напруги	+/−15 В

В програмно-апаратному комплексі імітаційного моделювання СЛУ такий аналізатор буде використано як засіб аналізу результатів діагностування. Підключення логічного аналізатора буде мати наступний вигляд (рис. 2.7).

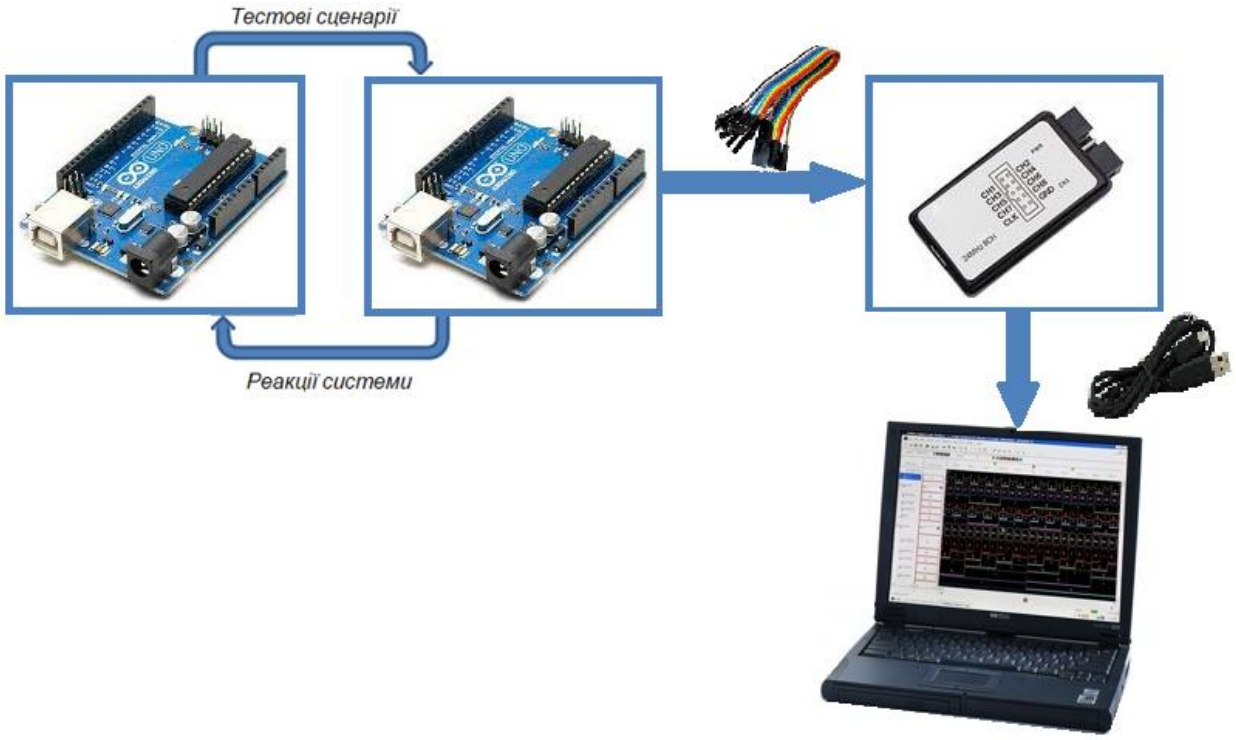


Рисунок 2.7 – Місце логічного аналізатору у ПАК імітаційного моделювання

Прилад супроводжується потужним програмним забезпеченням верхнього рівня у вигляді виконуваного модуля з можливістю аналізу багатьох популярних інтерфейсів обміну даними (UART, I2C, SPI, 1Wire, CAN та ін.), а так само наявністю SDK для написання аналізаторів для дослідження інших призначених для користувача протоколів.

Програмне забезпечення з офіційного сайта https://www.saleae.com/downloads/ дозволяє в зручному вигляді переглядати записані дані, обсяг і частота зняття яких задається програмно. Запуск вимірювань виконується за натискання клавіші, або по настанню заданої події по тригеру. Обсяг записуваних даних обмежується лише можливостями комп'ютера (об'ємом його оперативної пам'яті).

2.3 Реалізація часових параметрів моделі діагностування на Arduino

Керований об'єкт належить до фізичного світу, підпорядковується об'єктивним законам фізики, хімії, біології. Якщо, скажімо, автоматизується

зростання дріжджових паличок в автоклаві і для вирощування партії дріжджів потрібно 250 кВт·год, то ця енергія повинна подаватися в систему рівномірно і протягом тривалого часу, близько двох діб. Наївна спроба подати цю енергію якомога швидше призведе до того, що мікроорганізми загинуть від перегріву. Так само з механічними елементами конструкцій, наприклад клапанами, які не можуть відпрацювати керуючий сигнал миттєво. Для передачі пакету даних також потрібен час, що залежить від довжини пакета і швидкості передачі.

Іншими словами, існують цілком об'єктивні часові вимоги до порядку видачі керуючих впливів. Таким чином, алгоритм управління передбачає синхронізацію свого виконання з фізичними процесами у зовнішньому середовищі. Це обумовлює необхідність розвиненою служби часу і активну роботу з тимчасовими об'єктами: затримками, паузами, тайм-аутами.

З цим може допомогти така внутрішня периферія (модуль) мікроконтролеру як таймер, яка може подати сигнал в майбутньому, в той момент який розробник встановить у коді. Коли цей момент настає, викликається переривання мікроконтролера, нагадуючи йому що-небудь зробити, наприклад виконати певний фрагмент коду. Таймери працюють незалежно від основної програми. Замість виконання циклів або повторюваного виклику затримки можна призначити таймером виконання певних задач, в той час як інша частина коду виконує інші.

Налаштування таймеру – річ не дуже складна, але потребує багато уваги, щодо запису необхідних даних у певні регістри мікроконтролеру, які відповідають за роботу таймеру, період оновлення рахунку, на якій частоті він працює, подільник частоти і ще декілька регістрів. У випадку помилок у налаштуванні виникає багато ситуацій. Найліпшою помилкою є звичайна розбіжність між необхідним часом, коли повинні виконуватися задачі та реальним часом виконання. Здебільшого помилка у налаштуванні подільника та періоду, коли таймер повинен розпочати відлік знову. Найгірша з них – виведення з ладу таймеру або мікроконтролеру, яка пов’язана, наприклад, з введенням даних не в ті регістри або з відсутністю даних у регістрі частоти.

Самостійне налаштування таймеру потребує розуміння у внутрішній реалізації таймеру мікроконтролеру. Все це описано у офіційній документації про мікроконтролер (datasheet, reference manual).

У Arduino є 3 таймера – Timer0, Timer1 і Timer2.

Timer0 використовується для генерації переривань один раз в мілісекунду, при цьому відбувається оновлення лічильника, який передається в функцію millis (). Цей таймер є 8-бітним і рахує від 0 до 255. Переривання генерується при досягненні значення 255. За замовчуванням використовується тактовий дільник на 65, щоб отримати частоту, близьку к 1 кГц.

Timer1 це 16-бітний таймер з максимальним значенням 65535 (ціле без знака). Цей таймер використовує бібліотека Arduino Servo, враховуйте це якщо застосовуєте його в своїх проектах.

Timer2 – 8 бітний і дуже схожий на Timer0. Він використовується в Arduino функції tone ().

Переривання в Arduino можна розділити на кілька видів.

1. Апаратні переривання. Переривання на рівні мікропроцесорної архітектури. Після виникнення події виконання програми призупиняється, і управління переходить на обробник переривань. Після обробки управління повертається в перерваний код програми. З точки зору програми переривання – це виклик функції по зовнішній, не пов'язаній безпосередньо з програмним кодом, події. Сигнал переривання від кнопки виробляється у момент натискання або у момент віджимання кнопки (як запрограмує розробник). Сигнал переривання від таймера виробляється циклічно, до заданого часу періоду. Формує його апаратний таймер – лічильник з логікою, що скидає його код при досягненні певного значення. Програмно встановивши код для логіки скидання, є можливість поставити час періоду переривання від таймера.

2. Програмні переривання. Запускаються всередині програми за допомогою спеціальної інструкції. Використовуються для того, щоб викликати оброблювач переривань.

3. Внутрішні (синхронні) переривання. Внутрішнє переривання виникає в результаті зміни або порушення у виконанні програми (наприклад, при зверненні до неприпустимої адреси, неприпустимий код операції та інші).

Для роботи з перериваннями в Arduino використовується стандартна функція attachInterrupt (interrupt, function, mode), де в якості аргументів функції виступають:

– interrupt – номер викликається переривання (стандартно 0 – для 2-го піна, 1 – для 3-го Піна),

– function – назва функції, що викликається при перериванні (функція не повинна ні приймати, ні повертати будь-які значення),

– mode – умова спрацьовування переривання.

Можлива установка наступних варіантів умов спрацьовування:

– LOW – виконується за низьким рівнем сигналу, коли на контакті нульове значення. Переривання може циклічно повторюватися, наприклад, при кнопці.

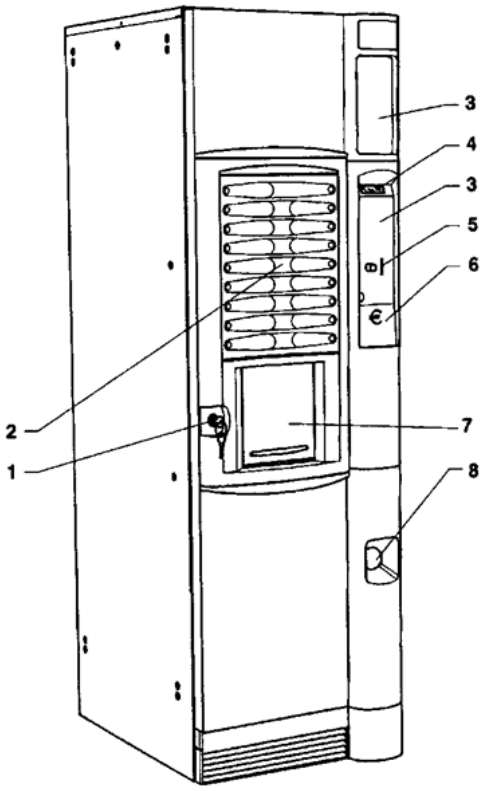
– CHANGE – по фронту, переривання відбувається при зміні сигналу з високого на низький або навпаки. Виконується один раз при будь зміні сигналу.

– RISING – виконання переривання один раз при зміні сигналу від low до high.

– FALLING – виконання переривання один раз при зміні сигналу від high до low.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ ПРОГРАМНО-АПАРАТНОГО КОМПЛЕКСУ ІМІТАЦІЙНОГО МОДЕЛЮВАННЯ

В якості прикладу системи логічного управління розглянемо торговий автомат з продажу кави. Розглянемо основні функції кавового автомата: приймати гроші (готівка); видавати решту; кип’ятити воду / молоко; підтримувати температуру; готувати обраний напій; функція автоматичної промивки. На рисунку 3.1 представлено схематичний вигляд кавового автомату. Всі елементи управління розташовані на зовнішній частині двері.



1. Замок.
2. Меню доступних напоїв:
   k1 – еспресо,
   k2 – капучіно,
   k3 – американо,
   k4 – американо з молоком,
   k5 – гарячий шоколад,
   k6 – какао.
3. Місце для розташування платіжної системи.
4. ЖК дисплей 2x16 символів.
5. Отвір для купюр та монет.
6. Дошка інструкцій із застосування
7. Відсік видачі
8. Кришка повернення монет

Рисунок 3.1 – Кавовий автомат

Автоматний підхід до проектування СЛУ передбачає опис алгоритму у вигляді кінцевого автомату (FSM), а саме за допомогою графа переходів (state diagram).

3.1 Побудова графа переходів кавового автомата

Перш за все треба визначитися з кількістю входів та виходів автоматної моделі. Інтерфейс автомату з продажу кави представлено на рисунку 3.2.

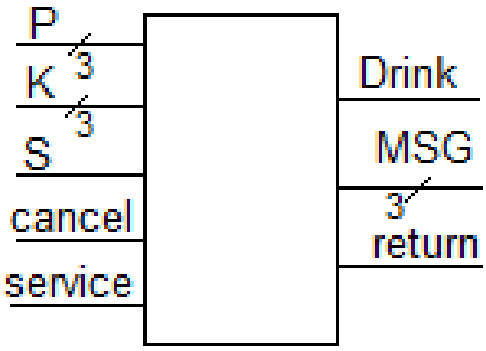


Рисунок 3.2 – Інтерфейс автомата з продажу кави

Згідно з рисунком 3.2 автомат має 5 вхідних змінних та 3 вихідних.

1. P = {000, 001, 010, 011, 100, 101, 110} – допустимі номінали купюр, де “000” – відповідає 1 грн., “001” – 2 грн., “010” – 5 грн., “011” – 10 грн., “100” – 20 грн., “101” – 50 коп.

2. K = {000, 001, 010, 011, 100, 101} – кнопки вибору напоїв k1,…., k6, де “000” – відповідає кнопці вибору еспресо (k1), “001” – капучіно (k2), “010” – американо (k3), “011” – американо з молоком (k4), “100” – гарячого шоколаду (k5), “101” – какао (k6).

3. S = {0, 1} – кнопка додавання однієї додаткової порції цукру.

4. service = {0, 1} – кнопка переходу до сервісного меню автомата.

5. cancel = {0, 1} – кнопка відміни, що переводить автомат до початкового стану.

6. Drink = {0, 1} – готовність напою.

7. Return = {0, 1} – повернення грошей користувачу через відсік видачі.

8. MSG = {000, 001, 010, 011, 100, 101, 110, 111} – повідомлення, де кожен код відповідає одному з можливих повідомлень автомата, які представлені в таблиці 3.1.

Таблиця 3.1 – Повідомлення автомата

Код	Повідомлення
000	Вітаємо! Виберіть напій та натисніть відповідну кнопку: k1,…., k6. При необхідності додайте цукру (одне натискання на кнопку S дорівнює одній порції).
001	Обраного напою не має в наявності. Виберіть інший напій.
010	Ціна обраного напою:_____. Автомат приймає гроші наступного номіналу 1, 2, 5, 10, 20 грн. та 50 коп.!
011	Ви внесли недостатньо коштів!
100	Не має решти!
101	Візьміть гроші/решту.
110	Процес приготування почався…
111	Візьміть свій напій!

Крім вхідних та вихідних сигналів автомат має внутрішні сигнали (сигнали від датчиків), що впливають на його роботу:

– ingredients = {0, 1} – наявність інгредієнтів;

– price = [14,22,16,20,15,15] – вартість p1,…,p6 напоїв k1,…,k6 відповідно;

– valid = {0, 1} – валідність купюр;

– money = {0, 1} – наявність грошей в купюро-/монето-приймачі;

– change = {0, 1} – наявність грошей для видачі решти;

– sum = [0,..,22] – сума грошей, що поступила до автомату;

– sub = [0,..,8] – решта, що обчислюється як різниця між вартістю напою та сумою грошей, що поступила до автомату (sub = sum – price);

– tempr = {00, 01, 10, 11} – температура рідини, де tempr = “00”, якщо температура води < 85 ºС, tempr = “01”, якщо температура молока < 75 ºС, tempr = “10”, якщо температура води ≥ 85 ºС, tempr = “11”, якщо температура молока ≥ 75 ºС.

Маючи всю необхідну інформацію ми можемо побудувати графу переходів торгового автомату (рис. 3.3).

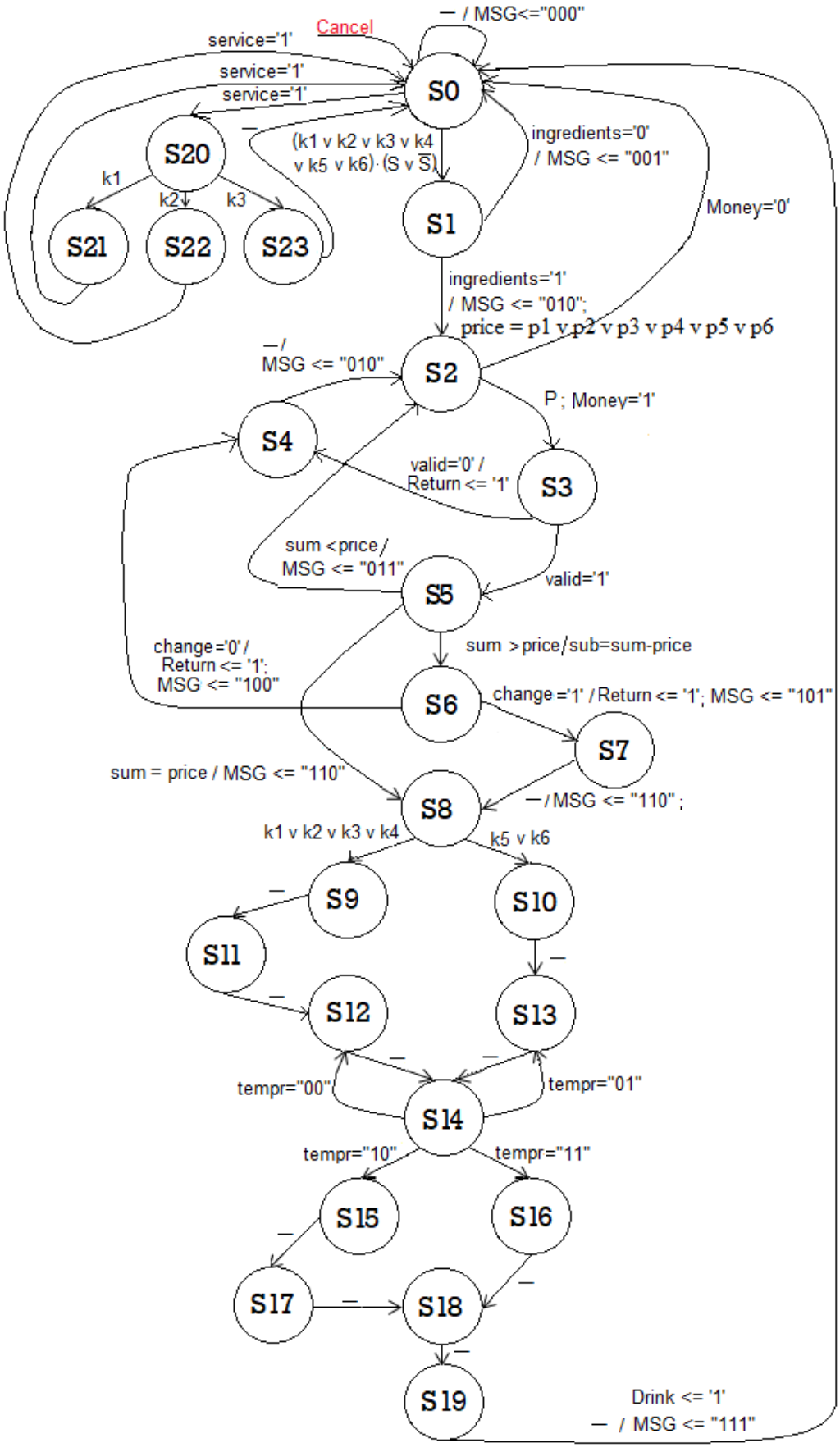


Рисунок 3.3 – Граф переходів

Розглянемо детально кожен стан автомата.

S0 – очікування. Автомат очікує вхідні впливи (кнопка вибору напоїв). Дисплей відображає інструкцію: «Вітаємо! Виберіть напій та натисніть відповідну кнопку: k1,…., k6. При необхідності додайте цукру (одне натискання на кнопку S дорівнює одній порції).» Крім того з цього стану можна перейти до сервісного режиму (стан S20) шляхом натискання кнопки «Service» (service=’1’).

S1 – перевірка наявності інгредієнтів. В залежності від обраного напою, автомат перевіряє чи має він усі необхідні інгредієнти (шляхом опитування датчиків, що містяться у контейнерах з інгредієнтами). Якщо усі необхідні інгредієнти присутні (ingredients=’1’), то автомат переходить до стану S2. Якщо ні (ingredients=’0’), то дисплей автомата протягом 2 сек. відображає повідомлення «Обраного напою не має в наявності. Виберіть інший напій.» та повертається до стану S0.

S2 – очікування грошей. Дисплей повинен відобразити повідомлення «Ціна обраного напою:_____. Автомат приймає гроші наступного номіналу 1, 2, 5, 10, 20 грн. або 50 коп.!». Якщо гроші поступили до купюро-/монето-приймача (сигнал money=’1’ від датчиків купюро-/монето-приймача), то автомат переходить до стану S3. Якщо протягом 10 сек. користувач не опустив гроші до автомату (money=’0’), то автомат переходить до стану S0.

S3 – перевірка грошей на відповідність. Якщо купюра/монета не відповідного номіналу (сигнал від датчиків valid=’0’), то автомат через 2 сек. повертає її через відсік решти (return<=’1’) та переходить до стану S4. Якщо купюра/монета відповідного номіналу (сигнал від датчиків valid =’1’), то автомат переходить до стану S5.

S4 – повернення грошей. Автомат повертає гроші користувача та через 2 сек. переходить до стану S2, нагадуючи про необхідність використання купюри правильного номіналу шляхом виводу на дисплей повідомлення «Ціна обраного напою:_____. Автомат приймає гроші наступного номіналу 1, 2, 5, 10, 20 грн. або 50 коп.!».

S5 – підрахунок суми грошей, що поступили до купюро-/монето-приймача (sum). Автомат порівнює отриману суму з вартістю обраного напою (price). Якщо сума менша ніж ціна, то автомат протягом 2 сек. виводить на дисплей повідомлення «Ви внесли не достатньо коштів!» та переходить до стану S2. Якщо сума більша ніж ціна, то автомат розраховує розмір решти (sub=sum-price) та переходить до стану S6. Якщо сума дорівнює ціні, то автомат переходить до приготування напою (стан 8).

S6 – перевірка наявності грошей для повернення решти. У разі відсутності грошей для видачі решти (сигнал від сенсорів change = ‘0’), дисплей автомата протягом 2 сек. виводить повідомлення «Не має решти!» та переходить до стану S4, повертаючи гроші через купюро-приймач/відсік решти (return<=’1’). Якщо автомат має гроші (сигнал від датчиків change = ‘1’), то він обчислює решту (перехід до стану S7) та починає готувати обраний напій (стан S8).

S7 – видача решти користувачу (return<=’1’), при чому дисплей автомата виводить повідомлення «Візьміть гроші/решту» протягом 2 сек.

S8 – стакан переміщається до місця приготування напою. Дисплей автомата виводить повідомлення «Процес приготування почався…» протягом 2 сек. Якщо користувач вибрав напої на основі молока (k5 та k6), то автомат переходить до стану S10, а якщо на основі води (k1,…., k4), то до стану S9.

S9 – помел зерен і дозування меленої кавової маси (протягом 2 сек.).

S10 – додавання інгредієнтів до стакану (протягом 2 сек.).

S11 – пресування кавової маси в брикет (протягом 2 сек.).

S12 – подача необхідного об’єму холодної води до відсіку для нагріву (протягом 1 сек.).

S13 – подача необхідного об’єму молока до відсіку для нагріву (протягом 1 сек.).

S14 – перевірка температури рідини: води чи молока (протягом 1 сек.).

Автомат повертається до стану S12 (для напоїв k1,…., k4), якщо температура < 85 ºС (сигнал від датчиків tempr = “00”), або до стану S13 (для напоїв k5 та k6), якщо температура < 75 ºС (сигнал від датчиків tempr = “01”).

Автомат переходить до стану S16 (для напоїв k1,…., k4), якщо температура ≥ 85 (сигнал від датчиків tempr = “10”), або до стану S17 (для напоїв k5 та k6), якщо температура ≥ 75 ºС (сигнал від датчиків tempr = “11”).

S15 – пропускання гарячої води через брикет під тиском 15 бар. Через 1 сек. автомат переходить до стану S17.

S16 – подача гарячого молока в стакан. Через 1 сек. автомат переходить до стану S18.

S17 – віджим відпрацьованої кавової маси та видалення. Через 2 сек. автомат переходить до стану S18.

S18 – видача палички для розмішування. Через 1 сек. автомат переходить до стану S19.

S19 – видача напою. Дисплей автомата протягом 5 сек. відображає повідомлення «Візьміть свій напій!» та повертається до стану S0.

S20 – вибір сервісного режиму. Автомат пропонує ввести код працівника та обрати одну з 3-х опцій: при k1=’1’, автомат переходить до стану S21, при k2=’1’ – до стану S22, при k3=’1’– до стану S23.

S21 – задання розходу продуктів на одну порцію того чи іншого напою. При натисканні на кнопку «Service» (service=’1’) автомат переходить до сану S0.

S22 – дисплей відображає показники кількості приготовлених напоїв. При натисканні на кнопку «Service» (service=’1’) автомат переходить до сану S0.

S23 – режим самоочищення. Автомат очищає себе за допомогою гарячої води (t = 90 ºС), що пропускається через усі системи під тиском протягом 30 сек.

Часові характеристики автомата:

– загальний час приготування напоїв k1,…., k4 не повинен перевищувати 30 сек., а для напоїв k5, k6 – 20 сек.;

– кожний стан має часовий параметр, тобто інтервал часу, впродовж якого автомат буде знаходитися у цьому стані. Для спрощення сприйняття графу, ці затримки на ньому не відмічені.

3.2 Схемна реалізація програмно-апаратного комплексу імітаційного моделювання

Згідно із запропонованою моделлю програмно-апаратного комплексу імітаційного моделювання СЛУ (рис. 2.3) з’єднаємо дві плати Arduino Uno, одна з яких буде грати роль системи логічного керування, а інша – апаратного імітатора, що використовується для подачі тестів (рис. 3.4).

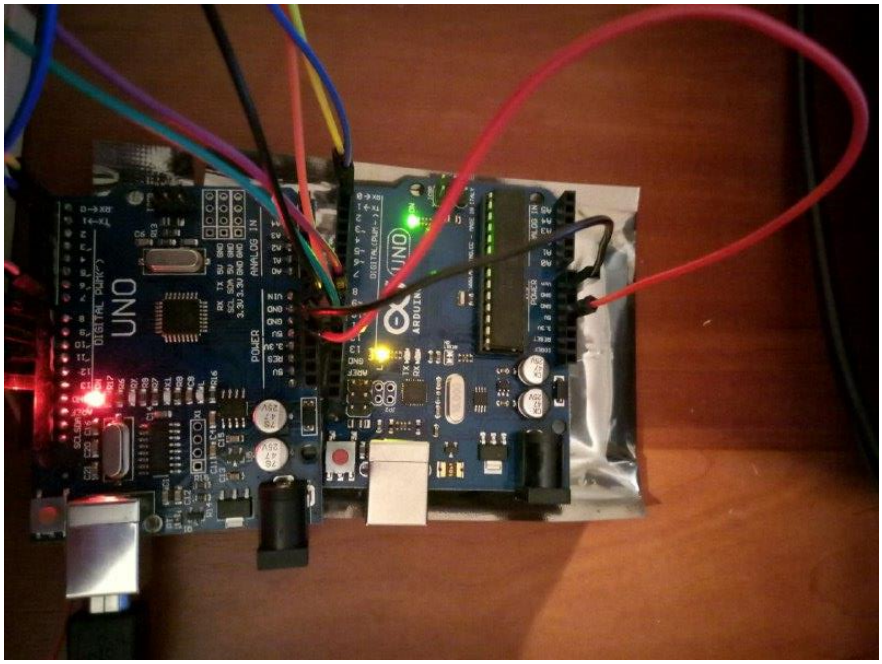


Рисунок 3.4 – Підключення двох плат Arduino

Кожна плата має свою прошивку згідно з виконуваною функцією, коди яких представлені у додатку А на лістингах А.1 – А.4.

При побудові такого комплексу треба взяти до уваги, що плати Arduino повинні бути з’єднані хрест-навхрест по шині UART, та мати загальний провід землі (GND) (рис. 3.5).

Arduino 1	Arduino 2
TX (transmit, передати)	TX (transmit, передати)
RX (receive, отримати	RX (receive, отримати
GND (ground, земля)	GND (ground, земля)

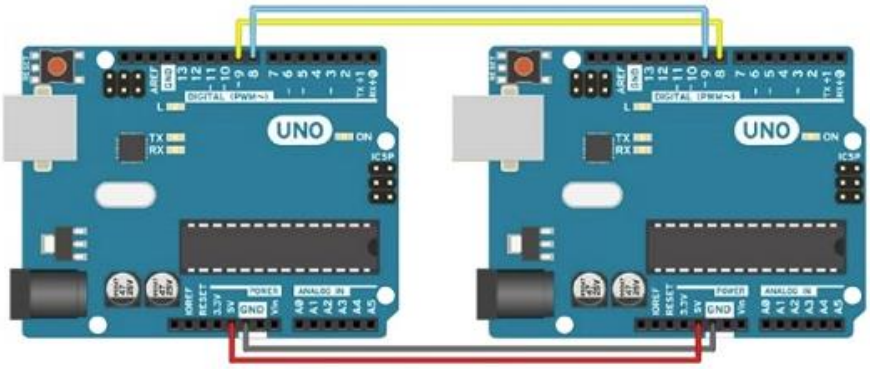


Рисунок 3.5 – З’єднання двох Arduino по шині UART

Для проведення діагностичного експерименту необхідно підключити логічний аналізатор за допомогою макетної плати (рис. 3.6).

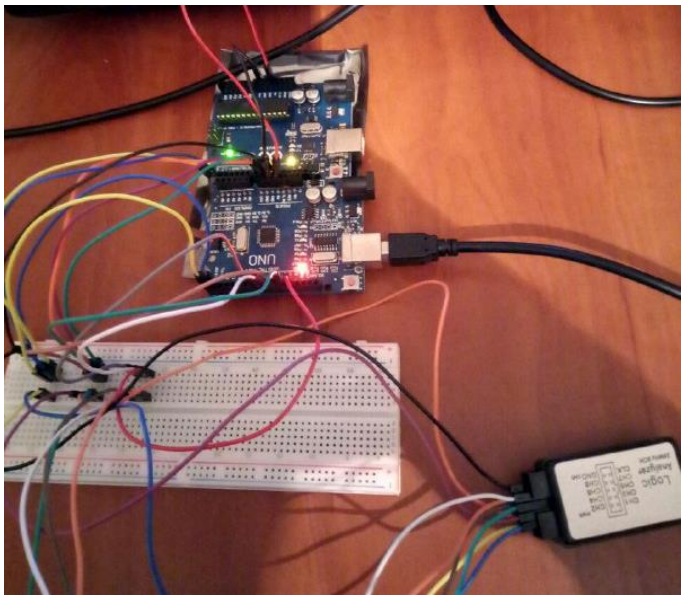


Рисунок 3.6 – Підключення логічного аналізатора

Використання макетної платі допомагає виділити всі необхідні сигнали (рис. 3.7) для логічного аналізатора, що значно полегшує подальше моделювання і отримання часових діаграм (waveform).

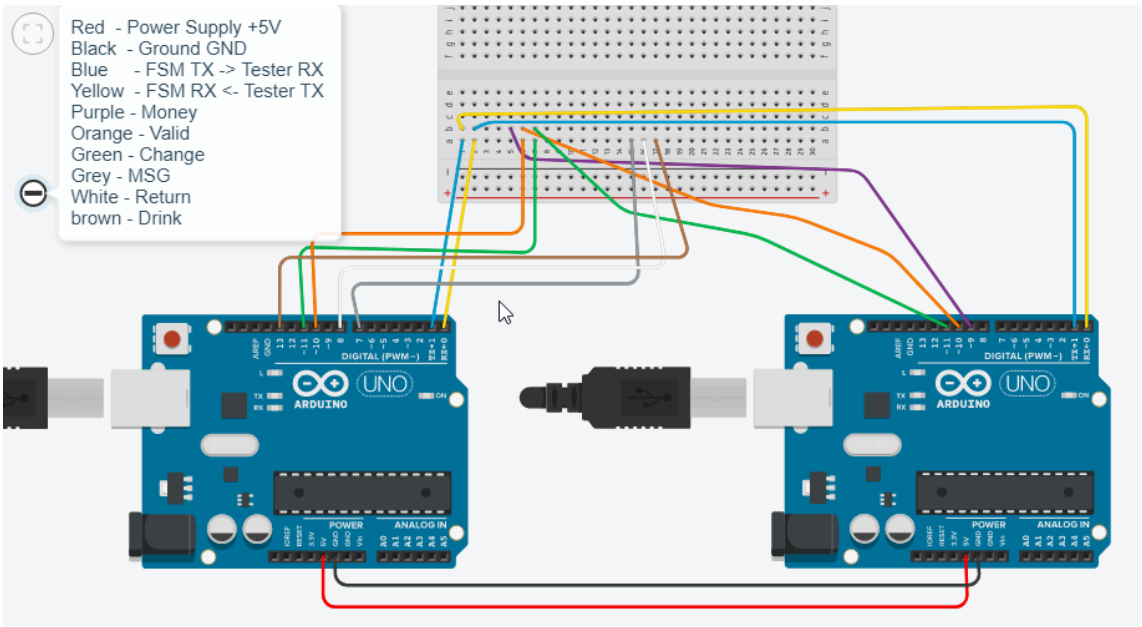


Рисунок 3.7 – Схема підключення плат Arduino до макетної плати

Програмно-апаратний комплекс імітаційного моделювання СЛУ зібрано. Переходимо до тестування розробленої моделі торгівельного автомату з продажу напоїв.

3.3 Діагностичний експеримент з перевірки коректності автомата з продажу напоїв

Діагностичний експеримент полягає у перевірці коректності автомата не тільки з точки зору його функціональності, але й коректності його часових параметрів. Розглянемо декілька сценаріїв проведення експерименту.

Сценарій № 1: користувач обрав капучіно, без цукру, опустивши до купюро-приймача точну суму (10+10+2), що дорівнює вартості напою (22 грн.), всі інгредієнти для приготування капучіно присутні (табл. 3.2).

Таблиця 3.2 – Сценарій № 1

Тест	K= “001”
	S = '0'
	P = “011”, “011” after 1 sec, “001” after 2 sec
	cancel = '0'
	service = '0'
Сигнали від датчиків	ingredients = '1'
	price = 22
	money = '1'
	valid = '1'
	sum = 10, 20 after 2 sec, 22 after 3 sec
	tempr = “00”, “10” after 2 sec
Вихідні реакції	MSG <= “000”;
	MSG <= “010”; протягом 20 sec.
	MSG <= “011”; протягом 1 sec. MSG <= “011”; протягом 1 sec.
	MSG <= “110”;
	MSG <= “111”;
	Drink <= '1';
	return <= '0'.

Такий тест перевіряє коректність наступних переходів автомата: S0- S1-(S2- S3- S5- S6) - (S2- S3- S5- S6) - (S2 -S3- S5) - S9 - S10- S12- S13- S15-(S13 - S15) - S16 - S18 - S19 - S20- S0.

Перед початком аналізу результатів моделювання треба взяти до уваги наступне.

Згідно з протоколом UART передача даних між двома Arduino відбувається побайтно. Якщо це число, то Arduino передає його як ASCII-код цього числа, конвертуючи його з числа в символ самостійно. Наприклад, число 1 конвертується в символ «1». У таблиці символів ASCII-коду – код цього символу 49. Тому при отриманні даних ми використовуємо операцію -48 (додаток А). Від 49 віднімаємо 48 і отримуємо нашу одиницю.

Код символу передається у вигляді 1 байта. У стандарті мови C / C ++ символьний тип даних (char) має розмір беззнакового цілого числа в 1 байт (8 біт). У коді використовується псевдонім uint8_t. Ім'я користувача – це використання іншого імені для якогось типу даних, щоб було простіше

розуміти яка інформація знаходиться у змінній (такий стиль програмування на мікроконтролері).

У разі передачі коду станів від FSM до тестеру, код станів S0-S8 передається як 1 байт. У випадку з станами, починаючи з S9 слід врахувати, що кожна цифра коду станів – це окремий байт даних (рис. 3.8).

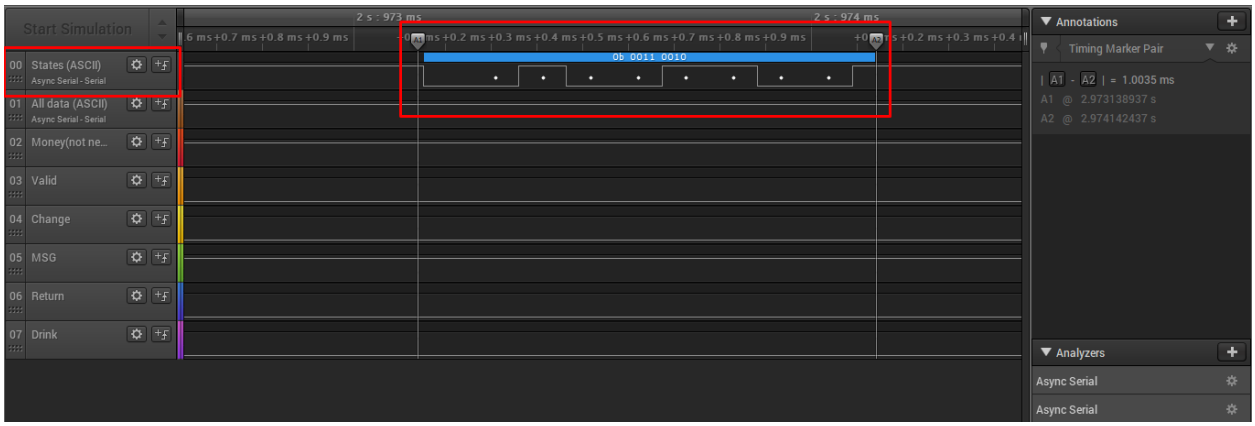


Рисунок 3.8 – Результати передачі стана автомата у вигляді 1 байта

Наприклад, код стану S12 буде передаватися двома байтами (рис. 3.9).

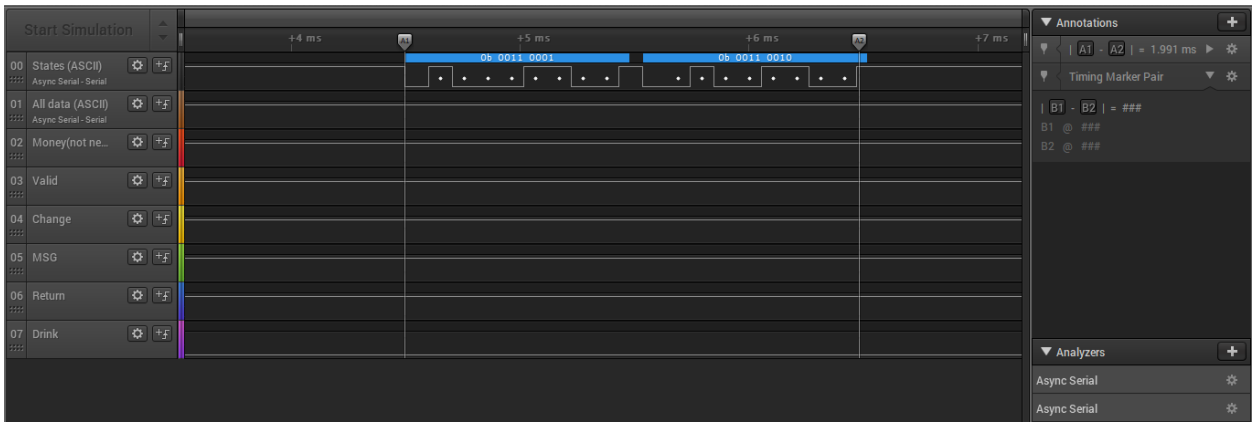


Рисунок 3.9 – Передача коду стана S12 у вигляді 2 байтів

У разі передачі даних від тестера до FSM відбувається аналогічна передача даних (код напою, код внесеної купюри і код температури рідини). Тільки тут все набагато простіше, тому що всі коди у десятичній системі числення представлені від 0 до 6 (рис. 3.10).

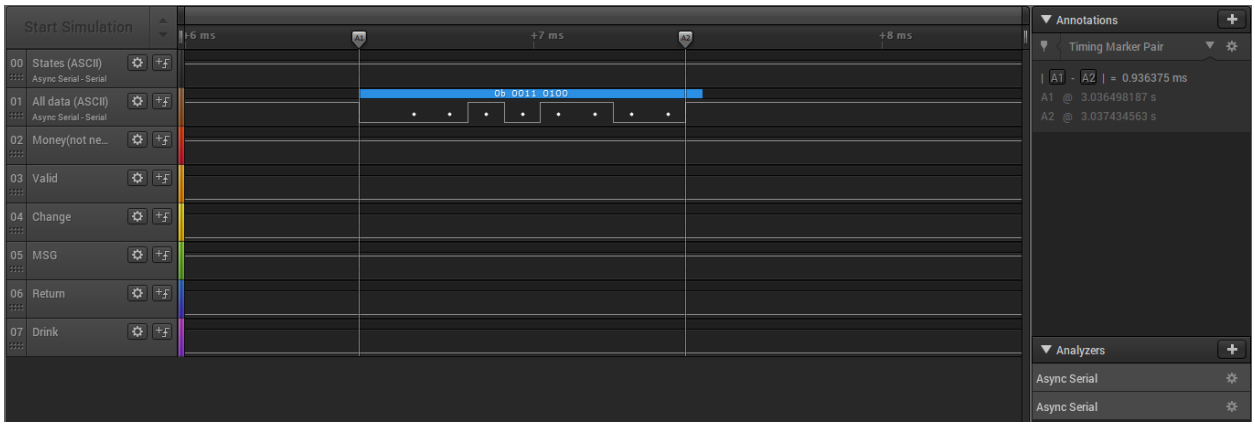


Рисунок 3.10 – Результати передачі даних о купюрі у вигляді 1 байта

Результати моделювання СЛУ за сценарієм № 1 представлені на рисунку 3.11.

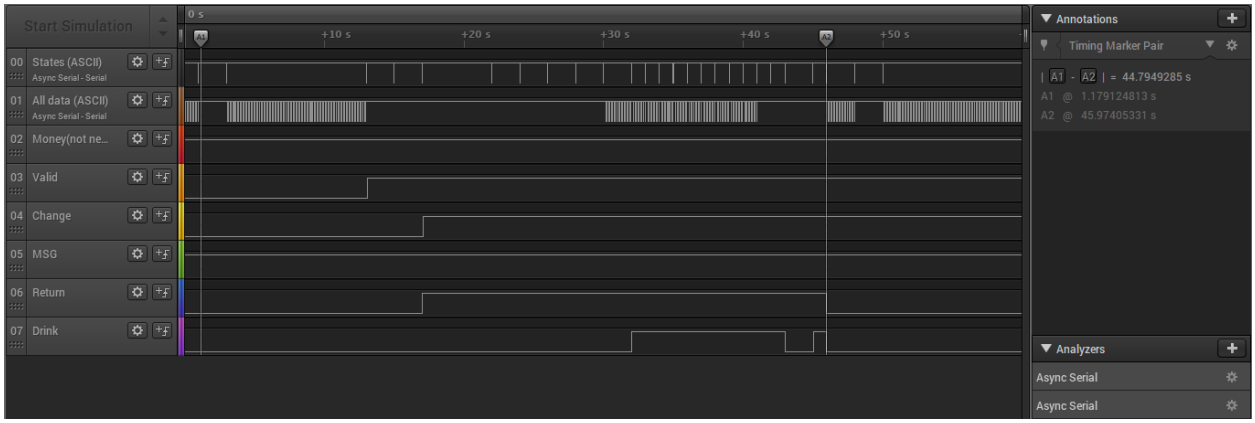


Рисунок 3.11 – Результати моделювання СЛУ за сценарієм № 1

Так само можна промоделювати сценарії, що наведено нижче.

Сценарій № 2: користувач обрав какао, підготувавши суму (20 грн. однією купюрою), що більша за вартість обраного напою (15 грн.), всі інгредієнти для приготування какао присутні, в автоматі присутні гроші для видачі решти (табл. 3.3). Такий тест перевіряє коректність наступних переходів автомата: S0- S1- S2- S3- S5- S7- S8 - S9 - S11- S14- S15- (S14- S15) - S17- S19- S20- S0.

Таблиця 3.3 – Сценарій № 2

Тест	K= “101”
	S = '0'
	P = “100”
	cancel = '0'
	service = '0'
Сигнали від датчиків	ingredients = '1'
	price = 15
	money = '1'
	valid = '1'
	sum = 20
	tempr = “10”, “11” after 2 sec
	sub = 20-15=5
	change = ‘1’
Вихідні реакції	MSG <= “000”;
	MSG <= “010”; протягом 20 sec.
	MSG <= “101”;
	MSG <= “110”;
	MSG <= “111”;
	Drink <= '1';
	return <= '1'.

Сценарій № 3: користувач обрав какао, підготувавши суму (20 грн. однією купюрою), що більша за вартість обраного напою (15 грн.), в автоматі присутні гроші для видачі решти, але відсутнє молоко (табл. 3.4). Такий тест перевіряє коректність наступних переходів автомата: S0- S1- S0.

Таблиця 3.4 – Сценарій № 3

Тест	K= “101”
	S = '0'
	P = “100”
	cancel = '0'
	service = '0'
Сигнали від датчиків	ingredients = '0'
	price = 15
	money = '1'
	sum = 20
	tempr = “10”, “11” after 2 sec
Вихідні реакції	MSG <= “000”;
	MSG <= “001”;

Сценарій № 4: користувач обрав американо з додатковою порцією цукру, підготувавши суму (20 грн. однією купюрою), що більша за вартість обраного напою (16 грн.), всі інгредієнти для приготування американо присутні, але в автоматі відсутні гроші для видачі решти (табл. 3.5).

Такий тест перевіряє коректність наступних переходів автомата: S0- S1- S2- S3- S5- S7- S4 - S2 - S0.

Таблиця 3.5 – Сценарій № 4

Тест	K= “010”
	S = '1'
	P = “100”
	cancel = '0'
	service = '0'
Сигнали від датчиків	ingredients = '1'
	price = 16
	money = '1', '0' after 10 сек.
	change = '1'
	valid = '1'
	sum = 20
Вихідні реакції	MSG <= “000”;
	MSG <= “010”; протягом 20 sec.
	MSG <= “100”;
	MSG <= “101”;
	return <= '1'.

Сценарій № 5: користувач обрав гарячий шоколад, підготувавши суму (10 грн. однією купюрою), що менша за вартість обраного напою (15 грн.), всі інгредієнти для приготування гарячого шоколаду присутні (табл. 3.6).

Такий тест перевіряє коректність наступних переходів автомата: S0- S1- S2- S3- S5- S6- S2 - S0.

Таблиця 3.6 – Сценарій № 5

Тест	K= “100”
	S = '0'
	P = “011”
	cancel = '0'
	service = '0'
Сигнали від датчиків	ingredients = '1'
	price = 15
	money = '1', '0' after 20 сек.
	valid = '1'
	sum = 10
Вихідні реакції	MSG <= “000”;
	MSG <= “010”; протягом 20 sec
	MSG <= “011”;
	MSG <= “101”;
	return <= '1'.

Сценарій № 6: користувач обрав капучіно, підготувавши точну суму (22 грн. 10-копійчаними монетами), всі інгредієнти для приготування капучіно присутні (табл. 3.7). Такий тест перевіряє коректність наступних переходів автомата: S0- S1- S2- S3- S4 - S2 - S0.

Таблиця 3.7 – Сценарій № 6

Тест	K= “001”
	S = '0'
	P = “XXX”
	cancel = '0'
	service = '0'
Сигнали від датчиків	ingredients = '1'
	price = 15
	money = '1', '0' after 20 сек.
	valid = '0'
	sum = 10
Вихідні реакції	MSG <= “000”;
	MSG <= “010”; протягом 20 sec
	MSG <= “101”;
	return <= '1'.

ВИСНОВКИ

Алгоритми управління складними технологічними об'єктами мають ряд властивостей, специфічних для галузі промислової автоматизації:

– відкритість – наявність «навколишнього середовища», зовнішнього світу, з яким взаємодіє алгоритм управління;

– подієвість – алгоритм управління формує керуючі впливу як реакцію на події (значущі зміни у вхідних даних), в тому числі на керуючі команди від оператора;

– невизначена тривалість функціонування алгоритму управління;

– синхронізм – необхідність синхронізації реакції алгоритму управління з подіями на об'єкті управління;

– логічний паралелізм – алгоритм управління структурно відображає паралелізм фізичних процесів на об'єкті управління, їх незалежність.

Реалізація алгоритмів управління засобами об'єктно-орієнтованих мов загального призначення чревата надмірним ускладненням програмної архітектури при зростанні складності алгоритму. Тому в області промислової автоматизації використовуються спеціалізовані мовні засоби для розробки алгоритмів управління: мови МЕК 61131-3, G (NI LabVIEW), Reflex.

Використання мов МЕК 61131-3 має свої обмеження, наприклад неможливість їх використання при необхідності інтеграції коду в сторонні системи. Дослідники альтернативних лінгвістичних засобів для опису алгоритмів управління пропонують і практично обґрунтовують ефективність предметно-орієнтованих мов на основі моделі кінцевого автомата.

При використанні автоматного підходу до програмування СЛУ основну проблему представляє рішення задачі діагностування системи. Це стало наслідком того, що керуючий алгоритм неможливо тестувати автономно від об’єкта управління.

Тестування алгоритму на реальному об'єкті управління може привести до поломки устаткування чи аварійної ситуації. Тому найбільш поширений підхід використовувати для тестування і верифікації алгоритмів управління імітатори об'єкта управління. Імітаційна модель, що замінює собою оригінал системи і дозволяє досліджувати істотні з точки зору цілей моделювання властивості. Моделювання, що імітує поведінку системи в часі тому й називається імітаційним, тому що дозволяє проаналізувати поведінку системи:

– без її реалізації, якщо ця система знаходиться в стадії проектування;

– без втручання в її функціонування, якщо ця система діє і таке втручання могло б привести до небажаних наслідків;

– без руйнування діючої системи, якщо мета впливу на неї полягає у визначенні допустимих меж такого впливу.

Магістерська робота присвячена проблемі апаратно-програмного моделювання (hardware-in-the-loop simulation). Метою апаратно-програмного моделювання є формування всіх електронних сигналів для детального тестування алгоритму управління об’єктом. Замість використання складних да в першу чергу дорогих програмованих логічних контролерів, для реалізації програмно-апаратного комплексу імітаційного моделювання використано дві плати Arduino Uno (на базі мікроконтролера Atmega328).

На одній платі було реалізовано систему логічного управління, а на другій – сценарії тестування розробленої системи. Завдяки наявності таймерів та функцій переривання, Arduino забезпечує 100%-контроль над часом та тривалістю подачі сигналів.

Для аналізу отриманих результатів діагностичних експериментів було використано логічний аналізатор Saleae Logic.

Для досягнення поставленої мети було проаналізовані особливості алгоритмів управління в системах реального часу та мови їх опису, розглянуті особливості апаратної реалізації СЛУ на базі мікроконтролера (беручи до уваги часові параметри та їх реалізацію). Обґрунтувати доцільність

використання плати Arduino UNO в якості апаратної платформи для імітаційного моделювання. Запропоновано модель програмно-апаратного комплексу імітаційного управління на базі двох мікроконтролерів та логічного аналізатору.

Для перевірки працездатності ПАК було створено автоматну модель системи управління торгового автомату з продажу напоїв. Проведений діагностичний експеримент з перевірки коректності запропонованої структури ПАК імітаційного моделювання системи логічного управління показав її працездатність. За допомоги логічного аналізатора отримані результати моделювання у вигляді часових діаграм. Аналіз діаграм показав коректність СЛУ (торгівельного автомату з продажі напоїв), не тільки з точки зору її функціональності, але й коректності часових параметрів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Зюбин В. Е. Итерационная разработка управляющих алгоритмов на основе имитационного моделирования объекта управления / В.Е. Зюбин // Автоматизация в промышленности. – 2010. – № 11. – С. 43-48.
2. Закревский А. Д. Логические основы проектирования дискретных устройств / А. Д. Закревский, Ю. В. Поттосин, Л. Д. Черемисинова. – М.: ФИЗМАТЛИТ, 2007. – 592 с.
3. Лях Т.В. Автоматизированная верификация алгоритмов управления сложными технологическими объектами на программных имитаторах / Т.В. Лях, В.Е. Зюбин, Н.О. Гаранина // Вестник НГУ. Серия: Информационные технологии. –2018. – Т. 16, № 4. – С. 85-94.
4. Зюбин В. Е. Программирование ПЛК: языки МЭК 61131-3 и возможные альтернативы // Промышленные АСУ и контроллеры. – 2005. – № 11. С. 31-35.
5. Лях Т. В., Зюбин В. Е., Сизов М. М. Опыт применения языка Reflex при автоматизации Большого солнечного вакуумного телескопа // Промышленные АСУ и контроллеры. – 2016. – № 7. – С. 37-43.
6. Зюбин В.Е. Программирование ПЛК: языки МЭК 61131-3 и возможные альтернативы // Промышленные АСУ и контроллеры. – 2005. – № 11. – С. 31-35.
7. Ефремов А.А. Модельноориентированное проектирование – международный стандарт инженерных разработок / А.А. Ефремов, С.С. Сорокин, С.М. Зенков. Дата доступу: 09.11.19. – Режим доступу: https://matlab.ru/upload/resources/EDU%20Conf/pp%2040-43%20Sorokin.pdf.– Загол. з екрану.
8. Поликарпова Н.И. Автоматное программирование / Н.И. Поликарпова, А.А. Шалыто. – СПб., 2008. – 167 с.

9. Шалыто А.А. SWITCH-технология – автоматный подход к созданию программного обеспечения «реактивных» систем / А.А. Шалыто, Н.И. Туккель // Программирование. – 2001. – № 5. – С. 45-62.

10. Зюбин В.Е. Программирование информационно-управляющих систем на основе конечных автоматов: учеб.-метод. пособие. / В.Е. Зюбин // Новосиб. гос. университет. – Новосибирск, 2006. – 96 с.

11. Антипова Е.В., Негода В.Н. Автоматизация проектирования программно-аппаратных реализаций автоматных диаграмм систем управления / Е.В. Антипова, В.Н. Негода // Автоматизация процессов управления. – 2012. – № 1 (27). – С. 47-55.

12. Спеціалізовані мікроконтролерні системи. Теорія і практика: Підручник / Є. І. Сокол, І. Ф. Домнін, О. М. Рисований та ін. – Харків: НТУ “ХПІ”, 2007. – 252 с.

13. Jones D. Model-Based Design of control systems: Simulate and test before committing to hardware / Doug Jones, Brian McKay // Undustrial embedded systems. – Режим доступу: http://industrial.embedded-computing.com/articles/model-based-design-control-systems-simulate-test-committing-hardware/ . – Дата звернення: 27.11.19. – Загол. з екр.

14. Волков Ю.В. Системы технического диагностирования, автоматичечкого управления и защиты. Часть 2: учебное пособие / Ю.В. Волков // ВШТЭ СПбГУПТД. – СПб, 2017. – 117 с.

15. Ryssel U. Generative function block design and composition / U. Ryssel, J. Ploennigs, K. Kabitzsch, // 6th IEEE Workshop on Factory Communication Systems, Torino, Italy, 2006, pp. 253-262.

16. Медведев В. Имитационное моделирование в промышленности // Plm news. Инновации в промышленности. Май 2008. http://simulation.su/uploads/files/default/2008-medvedev-1.pdf.

17. Зюбин В.Е. Итерационная разработка управляющих алгоритмов на основе имитационного моделирования объекта управления: учеб.-метод.

пособие / В.Е. Зюбин // Автоматизация в промышеленности. – Новосибирск: Новосиб. гос. ун-т., 2010. – С. 43- 48.

18. Halvorsen H.-P. Introduction to Hardware-in-the-Loop Simulation / Hans-Petter Halvorsen // Telemark University College. – 2012. – Режим доступу: https://ru.scribd.com/document/251663567/Introduction-to-HIL-Simulation. – Дата доступу: 24.11.19. – Загол. з екрану.

19. Шалыто А. А. Автоматное программирование / Н.И. Поликарпова, А.А. Шалыто. – Спб.: Питер, 2008. – 168 с.

20. Перцовский М. Применение логических анализаторов в тестировании цифровой техники / М. Перцовский, Е. Воробьёв, А. Трифонов // Современние технологии автоматизации. – №2. – 2000. – с. 6-12.