МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІВЕРСИТЕТ
Кафедра “підйомно-транспортного та робототехнічного обладнання"

КОНСПЕКТ ЛЕКЦІЙ

з дисципліні "Комп'ютерні методи розрахунку роботів"

Перший (бакалаврський) рівень вищої освіти
Спеціальність – 131 ПРИКЛАДНА МЕХАНІКА
Освітня програма – *Інженерія логістичних систем, Мехатроніка та промислові роботи*

ОДЕСА: ОНПУ, 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІВЕРСИТЕТ
Кафедра “підйомно-транспортного та робототехнічного обладнання"

Михайлов Євген Павлович

КОНСПЕКТ ЛЕКЦІЙ

з дисципліні "Комп'ютерні методи розрахунку роботів"

Перший (бакалаврський) рівень вищої освіти
Спеціальність – 131 ПРИКЛАДНА МЕХАНІКА
Освітня програма – *Інженерія логістичних систем*, *Мехатроніка та промислові роботи*

Затверджено
на засіданні кафедри
підйомно-транспортного і
робототехнічного обладнання
Протокол № 8 від 8 лютого 2021 р.

ОДЕСА: ОНПУ, 2021

Конспект лекцій з дисципліні «Комп'ютерні методи розрахунку роботів» для здобувачів першого (бакалаврського) рівня вищої освіти, спеціальність: 131 - Прикладна механіка, освітні програми: Мехатроніка та промислові роботи, Інженерія логістичних систем / Укл.: Михайлов Є. П. – Одеса: ОНПУ, 2021. - 112 с.

В конспекті розглянуті засоби комп'ютерного проектування промислових роботів, а також окремих їх компонентів. Наведена структура та склад роботів, основні типи та методи розрахунку основних параметрів. Розглянуті засоби комп'ютерного проектування механічних компонент та систем керування промисловими роботами.

Зміст

Передмова

Дисципліна «Комп'ютерні методи розрахунку роботів» призначена для вивчення здобувачами вищої освіти питань використання сучасних комп'ютерних методів розрахунку параметрів керування промисловими стаціонарними та мобільними роботами, які здійснюються самою комп'ютерною системою керування роботом.

Мета викладання дисципліни: сформувати у здобувачів вищої освіти знання в галузі аналізу конструктивних рішень з використанням комп'ютерних методів розрахунку; застосувати різні типів маніпуляційних систем, які використовують комп'ютерні методи розрахунку; надати здобувачам вищої освіти вміння та навики для вирішення питань автоматизації процесів транспортування в сучасних виробництвах на машинобудівних підприємствах; розглянути основні типи засобів, що використовують комп'ютерні методи розрахунку маніпуляторів та промислових роботів; опанувати фундаментальні поняття, та теоретичні положення сучасних засобів комп'ютерного керування маніпуляторами та промисловими роботами, а також засобів їх дослідження; повідомити здобувачам вищої освіти певної суми знань, що сприяють засвоєнню курсів загально-технічних та спеціальних дисциплін, дозволяють орієнтуватися в потоці наукової і науково-технічної інформації, характерному для сучасної епохи; сформувати у здобувачів вищої освіти наукового світогляду; оволодіти основними засобами проектування маніпуляторів та промислових роботів, вироблення у них початкових навичок проведення експериментальних досліджень при вивченні різних засобів комп'ютерного керування маніпуляторів та промислових роботів.

Основні задачі дисципліни: визначити закономірності використання комп'ютерних методів розрахунку роботів; характеризувати комп'ютерні методи розрахунку, на яких ґрунтуються різноманітні засоби проектування механічних та програмних компонентів роботів; аналізувати різні алгоритми проектування механічних та програмних компонентів роботів; розвити навички використання комп'ютерних методів розрахунку роботів; сформувати навички вибору проектування механічних та програмних компонентів роботів в залежності від задачі їх використання; навчити рекомендувати найприйнятніші комп'ютерні методи розрахунку роботів з урахуванням їх ефективності та екологічної небезпечності.

Основні результати навчання: уміння створювати параметричні комп'ютерні моделі деталей та складальних одиниць, та розраховувати їх за допомогою прикладного програмного забезпечення.

Статус дисципліни: вибіркова, належить до циклу дисциплін професійної підготовки.

Структура дисципліни складається з лекційних занять, лабораторних занять, індивідуальної роботи, у межах якої виконуються РГР, самостійної роботи, яка складається з підготовкидо лекційних занять, підготовки до практичних занять, підготовки до екзамену.

Конспект розроблений відповідно ОПП 2018.

Змістовий модуль 1. Основні принципи та засоби проектування роботів.

Лекція 1. Основні принципи та засоби проектування роботів

1.1. Принципи проектування роботів

Проектування - процес визначення архітектури, компонентів, інтерфейсів та інших характеристик системи або її частини (ISO 24765). Результатом проектування є проект - цілісна сукупність моделей, властивостей або характеристик, описаних у формі, придатній для реалізації системи.

Для визначення принципів проектування роботів розглянемо їх структуру. Найбільш поширеними роботами є промислові роботи, які згідно з ДСТУ 2879-94 представляють собою автоматичну машину, стаціонарну чи пересувну, з виконавчим пристроєм у вигляді маніпулятора, який має декілька ступенів рухомості, і перепрограмовуваним пристроєм програмного керування для виконання у виробничому процесі рухових і керувальних функцій [1].

Структура та складові частини промислових роботів залежать від сфери застосування, а також від типів та конструктивних особливостей.

Згідно з ДСТУ 2879-94 промислові роботи можна також поділити на **стаціонарні** та **пересувні** або **мобільні** роботи**.**

Стаціонарний робот - автоматична машина, що складається з виконавчого пристрою у вигляді маніпулятора, яка має кілька ступенів рухливості, і пристрою програмного управління. Такі роботи виробляються в підлоговому, підвісному і портальному виконанні.

Мобільний робот - автоматична машина, в якій є засоби пересування з автоматично керованими приводами. Мобільні роботи можуть мати різні засоби пересування, такі як **колісні**, **гусеничні** та **крокуючи** (існують також мобільні роботи, що повзують, плавають і літають, але вони як правило не використовуються для промислових завдань).

Незалежно від типу промислового робота вони мають схожу структуру (рис.1.1).

Рис. 1.1. Структура промислових роботів

Основними складовими частинами ПР є маніпулятор і керувальна система промислового робота. У свою чергу, кожна з цих частин включає ряд компонент. Тому загальна структура промислових роботів може мати у своєму складі таки основні компоненти.

Маніпулятор (механічна система) промислового робота це керований пристрій або машина для виконання рухових функцій і складається з виконавчого пристрою та робочого органу.

Виконавчий пристрій промислового робота (маніпулятора) – це пристрій промислового робота, який виконує його рухові функції.

Виконавчий пристрій маніпулятора представляє собою багатоланковий просторовий механізм, який може мати у загальному випадку поступальні, обертальні, циліндричні, сферичні та інші кінематичні пари.

До виконавчих пристроїв можна також віднести приводи та механічні компоненти, що здійснюють переміщення робота.

Пристрої переміщення виконують функції мобільності роботів.

Робочий орган промислового робота – це складова частина промислового робота, яка призначена для безпосереднього виконання технологічних операцій і (або) переміщення об’єктів.

До робочих органів відносяться захоплювальні пристрої маніпуляторів, які служать для захвату і утримання в певному положенні об'єктів маніпулювання.

Приводи призначені для здійснення переміщення механічних компонент виконавчого пристрою. В залежності від вимог до засобів переміщення використовують електричні, гідравлічні та пневматичні приводи. Для вирішення задач позиційного та контурного керування використовують регульовані приводи.

Керувальна система промислового робота призначена для формування і видачі керувальних дій виконавчому пристрою відповідно до керувальної програми.

Керувальна система включає сам пристрій керування, який здійснює програмне керування з можливістю перепрограмування інформаційно-вимірювальну систему, що визначає внутрішній стан робота та стан зовнішнього середовища, систему зв’язку, що здійснює зв'язок з іншими пристроями робототехнічних систем, пульт програмування та ручного керування, за допомогою якого здійснюється функції програмування для роботи в автоматичному режимі та ручного керування, що використовується для налагодження робота та здійснення режиму навчання.

Відповідно за системним підходом можна виділити три типи проектування роботів: структурний, блочно-ієрархічний і об'єктно-орієнтований.

Структурний підхід передбачає розробку різних варіантів робота з наявних компонентів і оцінку цих варіантів за заданими критеріями. Даний підхід заснований на розбитті робота на блоки за функціональною ознакою, коли кожен блок робота виконує окрему функцію.

Блочно-ієрархічний підхід заснований на виділенні різних ієрархічних рівнів робота, наприклад рівень планування руху, рівень управління, рівень виконавчих механізмів. При цьому на верхньому рівні дається загальний опис робота, а на наступних рівнях це опис деталізується. Для ієрархічної структури характерна зв'язок елементів з сусідніми рівнями, а зв'язки між елементами на одному рівні відсутні.

Об'єктно-орієнтований підхід розглядає складну систему як сукупність взаємодіючих між собою об'єктів, кожен з яких є екземпляром певного класу. Такий підхід найбільш перспективний при проектуванні складних систем.

Найчастіше розрізняють три рівні проектування. На **верхньому рівні** найчастіше розробляють структурні схеми, загальний вигляд робота тощо. На **середньому рівні** відбувається розробка окремих пристроїв, в результаті чого з'являються їх функціональні і принципові схеми. На **нижньому рівні** проектуються окремі деталі і елементи роботів.

Як правило, проектування робота є багатокрокової процедурою, що включає в себе: етап науково-дослідних робіт; етап ескізного проекту або дослідно-конструкторських робіт; етап технічного проекту; етап робочого проекту; етап випробувань дослідних зразків роботів.

Кожен з перерахованих етапів проектування робота може бути розділений на складові частини, що можна назвати проектними процедурами, наприклад: моделювання робота; розробка системи управління; розробка математичної моделі; розробка кінематичної схеми тощо.

Проектні процедури також підрозділяються на окремі операції, наприклад рішення прямої задачі кінематики, розрахунок коефіцієнтів регулятора і т.д.

Оскільки промисловий робот складається з механічних компонент, до яких належать маніпулятор, виконавчий пристрій, засоби переміщення робота та інших, а також комп’ютерної системи керування, то загальна процедура проектування робота поділяється на

проектування механічних компонент робота та системи керування, яка в свою чергу складається з апаратних та програмних частин.

Стосовно використання методів комп'ютерних розрахунків треба відзначити, що з одного боку комп'ютерні розрахунки використовують в процесі проектування роботів, а з другого, в процесі роботи роботів система керування за допомогою комп'ютерних розрахунків здійснює перетворення параметрів переміщення робочих органів та самого робота в параметри переміщення окремих ланок та приводів.

1.2. Системи автоматизованого проектування роботів

Усі типи роботів можна умовно поділити на дві основні групи, а саме, на універсальні роботи широкого призначення, що складаються з окремих уніфікованих вузлів або модулів, та спеціалізованих роботів, які найчастіше створюються для виконання обмежених функцій, або є складовими частинами комплексних виробничих систем. Згідно з цим використовують різні системи автоматизованого проектування (САПР).

У першому випадку для проектування роботів найчастіше використовують системи автоматизованого проектування, які призначені для певних типів роботів. Наприклад, програмний комплекс ABB robot studio, призначений для проектування та моделювання роботів фірми ABB, програмний комплекс фірми KUKA, призначений для проектування та моделювання роботів цієї фірми, Microsoft Robotics Developer Studio, що представляє собою Windows-орієнтоване середовище для керування роботами і їх симуляції та дозволяє проектувати окреми типи роботів, що входять до її каталогу.

У другому випадку для проектування роботів використовують універсальні засоби проектування окремих компонент роботів, які можна поділити на механічні компоненти, приводи та інші засоби пересування, системи керування. У цьому випадку загальна система проектування може складатися з окремих засобів проектування механічних, електромеханічних та програмних компонент.

Найчастіше для цього використовують такі програмні комплекси проектування, як **SolidWorks, MathCAD**, **Matlab** тощо.

Для проектування програмного забезпечення спеціалізованих роботів використовуються програмні комплексі відповідно з обраною системою керування.

Так для вмонтованих систем керування найчастіше використовують контролери **Ардуіно,** програмування яких здійснюється за допомогою середою розробки **Arduino IDE**, та шляхом використання графічних середовищ програмування, таких як, **Visuino**, **Scratch**, **ArduBlock**, **Fritzing** та інших.

Для проектування електроприводів окремих фірм існують відповідні засоби проектування, наприклад, програмний комплекс **Starter** для приводів фірми Сименс.

Для проектування систем керування також використовують програмні комплекси проектування, що складаються з засобів проектування апаратних компонент, засобів створення програмного забезпечення та засобів налагодження системи керування та пошуку помилок під час роботи, наприклад, система проектування **Portal** фірми Сименс.

Універсальним засобом програмування робототехнічних пристроїв є програмна середа **LabVIEW**, яка здійснює програмування у графічному вигляді та має вмонтовані засоби для програмування робототехнічних пристроїв та їх компонент/

Взагалі існують такі види забезпечення САПР:

- технічні, що включають апаратні засоби;
- математичні, що об'єднують математичні методи, моделі та алгоритми для виконання проектування;
- програмні, що представляється комп'ютерними програмами САПР;
- інформаційні, що складаються з бази даних, що використовуються при проектуванні;
- лінгвістичні (мови програмування і обміну даними);
- методичні, що включає різні методики проектування;
- організаційні, яке надається штатними розкладами, посадовими інструкціями тощо.

За типом базової підсистеми САПР поділяються на такі:

1) САПР на базі машинної графіки і геометричного моделювання, які використовуються в процесі конструювання, визначення просторових форм і взаємного розташування об'єктів;

2) САПР на базі баз даних, в яких при невеликому обсязі мате-тичних розрахунків обробляється великий обсяг даних;

3) САПР на базі конкретного прикладного пакета, які представляють собою автономно використовувані пакети, наприклад, програми логічного проектування на базі мови VHDL, математичні пакети типу MathCAD, Matlab, Maple;

4) комплексні САПР, що складаються з сукупності підсистем попередніх видів.

1.3. Моделювання та аналіз роботів

Моделювання та аналіз займають значне місце при проектуванні робототехнічних систем. Місце моделі і її аналіз в процедурі проектування, прийнятої при розробці систем управління, представлені на рис. 1.2.

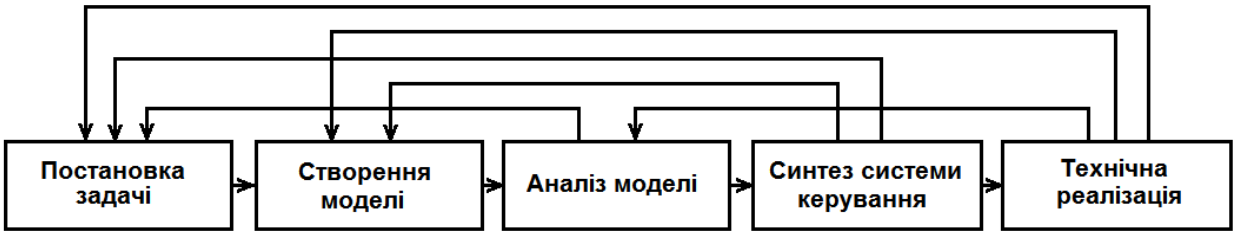


Рис. 1.2. Процедура проектування робота

Відповідно до класичної теорії управління, процедура дослідження і синтезу робота складається з наступних етапів:

- постановка задачі, формулювання критеріїв та побудова моделі робота;
- аналіз моделі і коригування поставленого завдання управління або моделі;
- синтез системи управління відповідно до заданих критеріїв;
- аналіз синтезованої системи управління роботом;
- структурно-алгоритмічна і програмно-апаратна реалізація системи управління робота.

Виконання перерахованих етапів є ітераційною процедурою, коли можливі переходи до попередніх етапів з метою уточнення моделі робота, зміни постановки завдання або повторному синтезу системи управління за результатами її аналізу.

Кожен з перерахованих етапів є необхідним, проте в кожному конкретному випадку значимість того чи іншого етапу може превалювати над іншими.

Для дослідження поведінки робота по його моделі найкращими моделями є ті, які побудовані на основі фізичних (механічних, електромеханічних і ін.) законів, що описують робот.

Для побудови функцій управління часто досить визначити реакцію робота на вплив, наприклад у вигляді перехідної характеристики або передавальної функції.

На початкових етапах побудови моделей, коли відбувається якісний аналіз робота, будуються описові моделі, що відображають основні якісні та структурні закономірності.

Ідентифікація робота дозволяє визначити модель, необхідну для побудови управління. При цьому вже на цьому етапі враховується цільова функція системи управління.

Роботи відносяться до об'єктів, структура яких добре вивчена і які функціонують в заздалегідь визначених умовах. Для таких об'єктів можливо аналітичне побудова моделі з уточненням параметрів за допомогою різних методів ідентифікації

При створенні моделі робота також є характерним використання модульного принципу, а саме розбиття на окремі взаємодіючі частини, що включають електромеханічні компоненти (виконавчі пристрої) та пристрої керування.

Прикладом засобу моделювання різних роботів є симулятор роботів V-REP компанії Coppelia Robotics [13], який дозволяє зробити та перевірити комп’ютерну модель роботу (рис. 1.3).

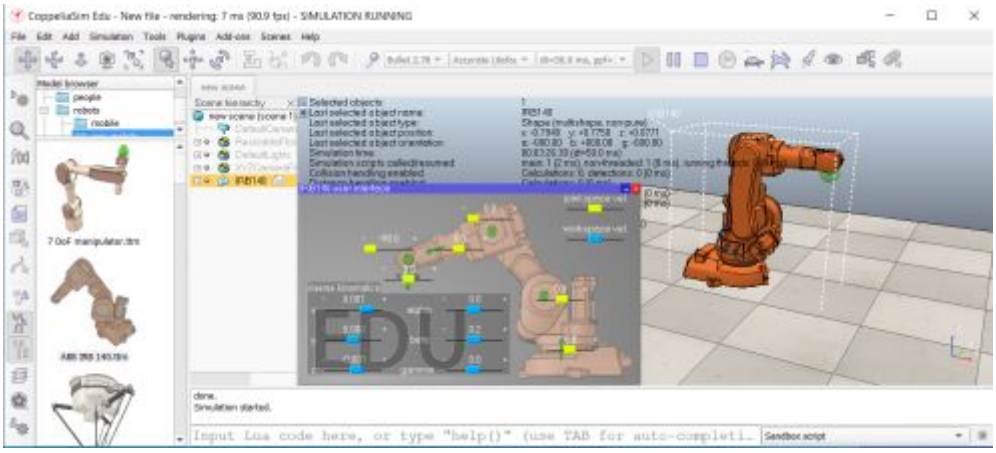


Рис. 1.3. Модель робота

Симулятор має бібліотеку з великою кількістю моделей стаціонарних та мобільних роботів, а також додаткового обладнання (рис. 1.4), що дозволяє створити модель взаємодії робота з додатковим обладнанням (рис.1.5).

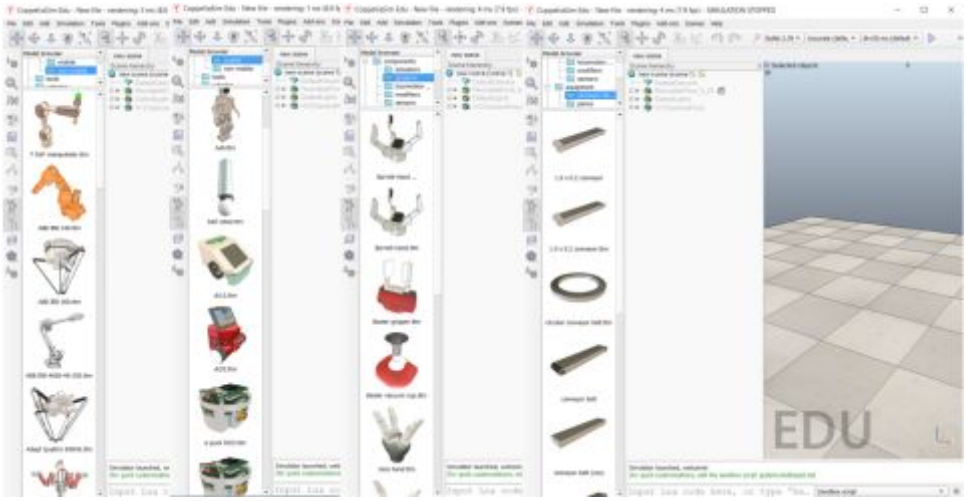


Рис. 1.4. Бібліотека моделей стаціонарних, мобільних роботів та додаткового обладнання

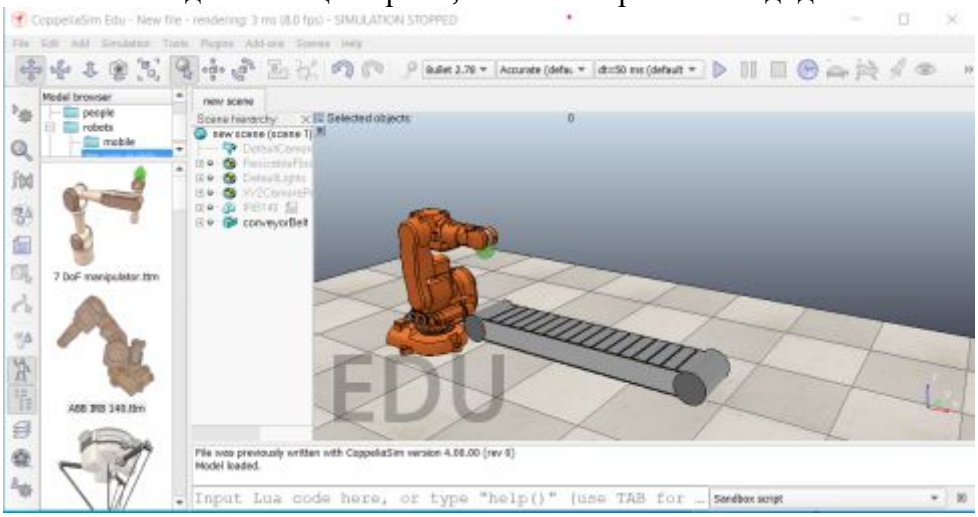


Рис. 1.5. Модель робота з додатковим обладнанням

Комп’ютерні розрахунки є важливою складовою частиною програмного забезпечення в системах керування сучасних роботів.

Розглянемо, які комп’ютерні розрахунки використовують системи керування роботів, виходячи з узагальненої структури промислових роботів, наведеної на рис. 1.6, яка включає систему переміщення як робочого органу, так і самого робота (наприклад, для мобільних роботів).

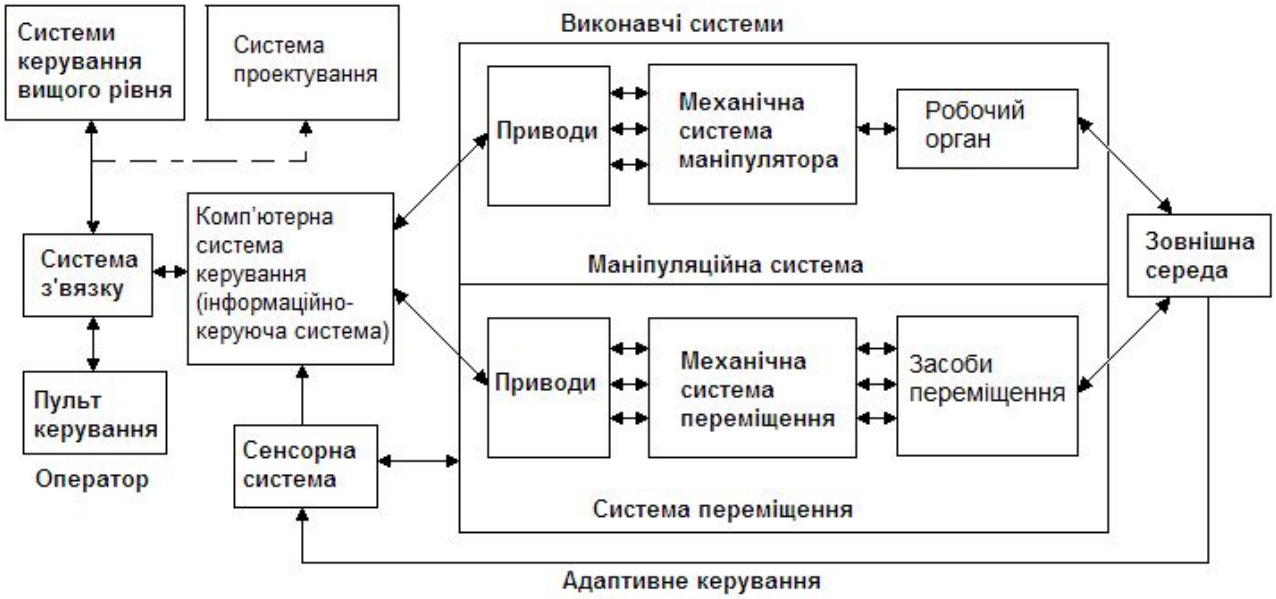


Рис. 1.6. Узагальнена структура промислових роботів

Очевидно, що комп'ютерні розрахунки здійснює комп'ютерна система керування, яка після обробки інформації про стан окремих компонент робота (внутрішня інформація) та стан зовнішнього середовища (зовнішня інформація) в залежності від встановленого завдання (позиції переміщення робочого органу маніпулятора та самого робота, функції, що виконує робочий орган тощо) видає керуючі сигнали на відповідні приводи.

При цьому у багатьох випадках треба використовувати досить складні розрахункові алгоритми обробки даних, що отримані від інформаційних систем, та керування приводами з використання задач прямої то зворотної кінематики. Як показано на рис. 1.7 для переміщення робочого органу в задану позицію (x, y, z) треба визначити відповідні кути повороту усіх ланок $\varphi_1 \ldots \varphi_6$ (задача зворотної кінематики).

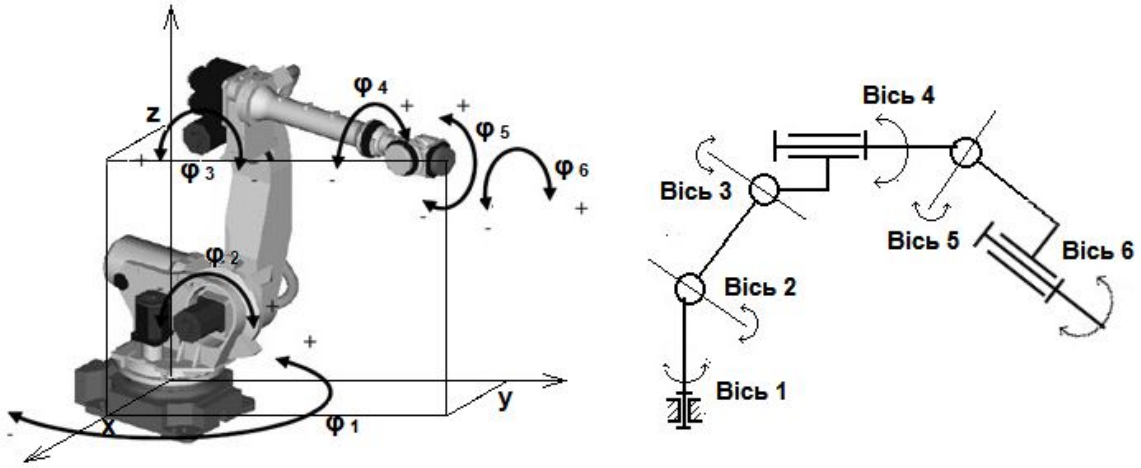


Рис. 1.7. Візначення кутів повороту усіх ланок для переміщення робочого органу в задану позицію

При використання універсальних роботів ці задачі вирішуються існуючими системами проектування.

При використання універсальних роботів ці задачі вирішуються існуючими системами проектування. Так мова програмування робота має команди переміщення виконуючого пристрою у вказану позицію, а відповідне переміщення окремих ланок маніпулятора здійснює система керування робота шляхом перерахунку положення виконавчого пристрою в положення окремих ланок.

При використанні спеціалізованих роботів ця задача виконується у ході розробки програми керування роботом, яка повинна самостійно виконувати перерахунки положення виконавчого пристрою в положення окремих ланок, що дозволяє значно спростити програму керування роботом. Розглянемо це на конкретному прикладі.

На рис. 1.8 наведений маніпулятор, для якого треба визначити кути повороту окремих ланок φ_1 та φ_2, якщо задається положення робочого органу – координати (x, y).

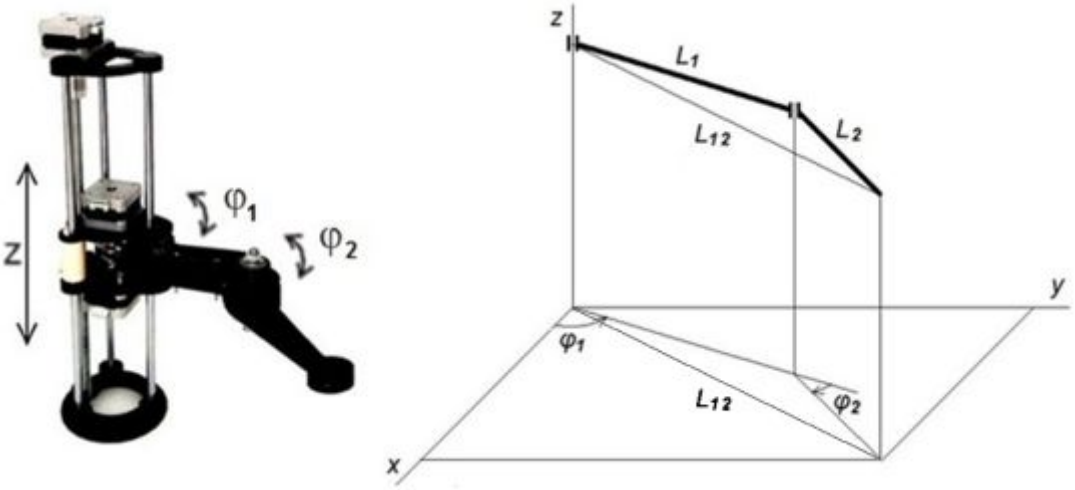


Рис. 1.8. Маніпулятор, для якого треба визначити кути повороту окремих ланок

Керування роботом здійснюється за допомогою програми, що виконує розрахунки кутів повороту ланок φ_1 та φ_2 за такими формулами:

$\varphi_1 = \arccos(x/ L_{12}) + \arccos((L_1^2 - L_2^2 + L_{12}^2) / 2\cdot L_{12}\cdot L_1)$;

$\varphi_2 = \pi - \arccos((L_1^2 + L_2^2 - L_{12}^2) / 2\cdot L_1\cdot L_2)$;

де $L_{12} = (x^2 + y^2)^{½}$.

Код програми, що здійснює такі обчислення на мові С++ має такий вигляд:

```
L12 = sqrt(sq(X) + sq(Y));
//обчислення φ1 у радіанах
Phi1 = acos(X/L12) +  acos((sq(L1) - sq(L2) + sq(L12)) / (2* L1* L12));
//обчислення φ2 у радіанах
Phi2 = PI - acos((sq(L1) + sq(L2) - sq(L12)) / (2* L1* L2));
Phi1Deg = Phi1 * RAD_TO_DEG; //результат φ1 у градусах
Phi2Deg = Phi2r * RAD_TO_DEG; //результат φ2 у градусах
```

На рис. 1.9 наведений результат, отриманий на контролері.

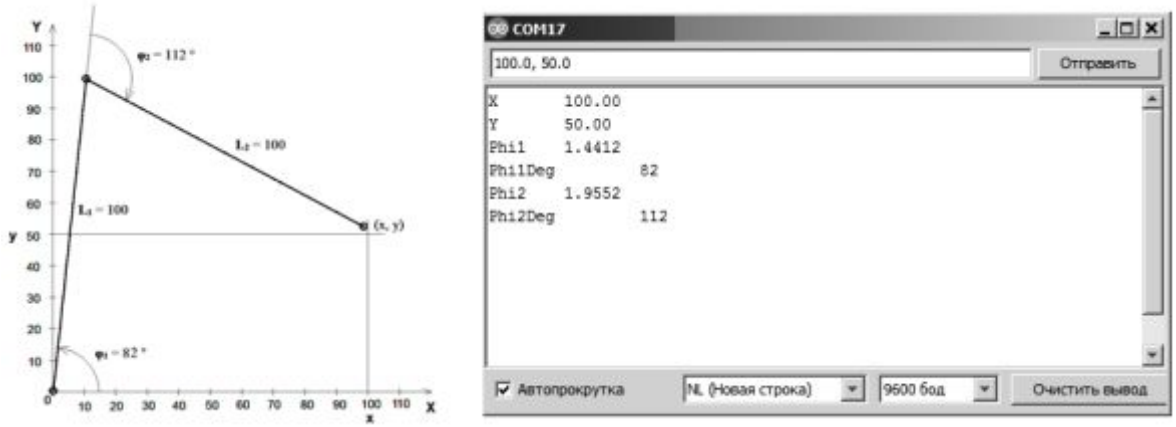


Рис. 1.9. Результат, отриманий на контролері

Контрольні питання

1. Визначити, на які компоненти розкладаються роботи під час проектування?
2. Назвати, які типи проектування роботів можна виділити за системним підходом?
3. Розповісти, чим відрізняються структурний і блочно-ієрархічний підходи до розробки робототехнічних комплексів?
4. Описати, у чому полягає об'єктно-орієнтований тип проектування?
5. Назвати, які основні етапи можна виділити для загальної процедури проектування робота?
6. Розповісти, які різновиди САПР використовують для проектування роботів?
7. Назвати, які види САПР існують взагалі?
8. Розповісти, які функції виконує моделювання та аналіз роботів?
9. Визначити, на які етапи поділяється процедура дослідження і синтезу робота?
10. Описати, як здійснюються комп’ютерні розрахунки для вирішування задач переміщення виконавчого пристрою та самого робота?

Лекція 2. Основні засоби проектування роботів

2.1. Засоби проектування спеціалізованих роботів

Для проектування спеціалізованих роботів найчастіше використовують універсальні засоби проектування окремих компонент роботів, які можна поділити на механічні компоненти, приводи та інші засоби пересування, системи керування. У цьому випадку загальна система проектування може складатися з окремих засобів проектування для механічних, електромеханічних компонент та систем програмного керування [1].

Для проектування механічних компонент робота найчастіше для цього використовують такі програмні комплекси проектування, як **SolidWorks, MathCAD**, **Matlab** [3] тощо.

Ці програмні комплекси детально розглядаються в дисциплінах по автоматизованому проектуванню, тому тут не розглядаються.

Для реалізації керування роботом можуть використовуватися різні засоби керування, які можна поділити на універсальні системи керування, наприклад, програмовані логічні контролери, та вбудовані пристрої керування на основі мікропроцесорної техніки. Тому для проектування систем програмного керування можна використовувати різні засоби проектування.

Засоби проектування вбудованих пристроїв керування на основі мікропроцесорної техніки

Відмінними властивостями вбудованих систем управління є їх компактність і можливість автономного використання в складі комплексних систем управління. Тому основою для автономних вмонтованих пристроїв керування найчастіше є однокристальні мікроконтролери.

Однокристальний мікроконтролер це закінчений обчислювальний пристрій, що виконується у вигляді однієї мікросхеми. Ці пристрої призначені в основному для вирішення завдань управління і часто мають вбудовані функції опитування різних датчиків (цифрові та аналогові входи) і видачі регульованих управляючих впливів (цифрові виходи або виходи з широтно-імпульсною модуляцією).

Універсальні однокристальні мікроконтролери призначені для використання широким колом користувачів. При цьому користувач, як правило, самостійно розробляє апаратне та програмне забезпечення, тому для універсальним мікроконтролерів найчастіше здійснюється за допомогою універсальних апаратних та програмних засобів проектування та налагодження.

В даний час широко використовуються мікроконтролери фірми Atmel, наприклад, мікроконтролери AVR (рис. 2.1)

Рис. 2.1. Мікроконтролери AVR

Микроконтролери AVR мають такі основні характеристики:

- 8-розрядний процесор з широким набором команд;
- до 100 двонапрямлених ліній введення-виведення;
- аналого-цифрові перетворювачі з роздільною здатністю 12 біт і до 2 млн вибірок в секунду;
- широкий набір комунікаційних функцій, включаючи можливість підключення USB;
- 16-бітові таймери / лічильники з каналами порівняння;
- функції переривання тощо.

Мікроконтролери AVR мають гарвардську архітектуру, при якій для програми і даних використовуються різні пристрої пам'яті.

Ці мікроконтролери є універсальними пристроями.

Для розробки програмного забезпечення однокристальних мікроконтролерів використовуються персональні комп'ютери, на яких встановлено відповідні мови програмування.

Програмування мікроконтролерів зазвичай здійснюється з використанням таких мов, як Асемблер або С, хоча існують компілятори для інших мов, використовуються також інтерпретатори Бейсіка і Форту.

Для налагодження програм використовуються програмні симулятори у вигляді спеціальних програм для персональних комп'ютерів, що імітують роботу мікроконтролера у вигляді програмної моделі.

Для налагодження апаратних компонент використовують схемні емулятори, що являють собою електронні пристрої, які імітують мікроконтролер.

Ці емулятори можна підключити замість однокристального мікроконтролера до вбудованого пристрою та перевірити роботу програми.

Прикладом використання вбудованих пристроїв керування різного рівня складності є регульовані електроприводи, наприклад, сервоприводи з відносно простими пристроями керування, та частотні перетворювачі з досить складними пристроями керування.

Сервоприводи використовуються у разі необхідності здійснити поворот на певний кут. Під сервоприводом в даному випадку розуміють механізм з електромотором, який можна повернути в заданий кут і утримувати в цьому положенні. Для визначення кута повороту використовують потенціометричний датчик.

Для управління сервоприводами використовується широтно-імпульсна модуляція. При цьому кут повороту визначається тривалістю імпульсу (рис.2.2).

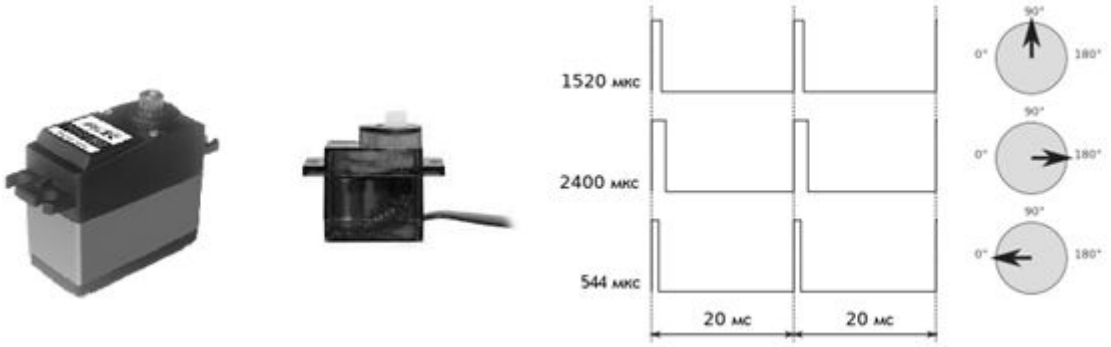


Рис. 2.2. Сервопривод

Однокристальні мікроконтролери знайшли широке застосування для вирішення найрізноманітніших завдань управління, але, оскільки вони являють собою мікросхеми, то для проектування пристрої на їх основі та подальшого використання потребується цілий ряд додаткових засобів, включаючи друковану плату, де встановлюється мікросхема, пристрій програмування тощо.

Тому цілий ряд фірм налагодив випуск мікроконтролерів у вигляді однієї друкованої плати (рис. 2.3) з можливістю підключення додаткових модулів і здатних вирішувати велике коло завдань, в тому числі обробки даних, керування різними приладами, управління рухом, тощо.

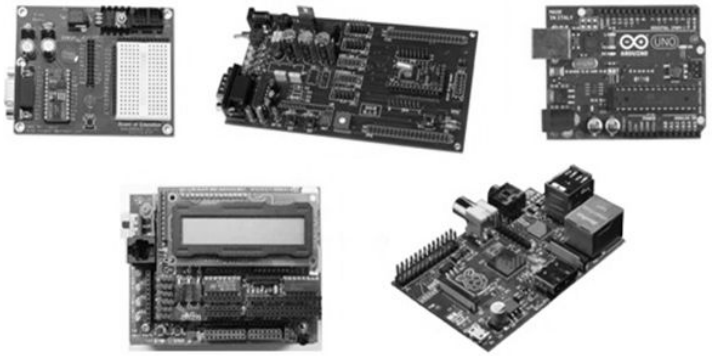


Рис. 2.3. Мікроконтролери у вигляді друкованої плати

Прикладом використання мікроконтролерів AVR може служити апаратно-програмний комплекс Arduino, який представляє собою набір засобів, пристосованих для побудови простих систем автоматики і робототехніки, орієнтований на непрофесійних користувачів.

Програмна частина складається з безкоштовної програмної оболонки для написання програм, їх компіляції і програмування апаратури.

Апаратна частина являє собою набір готових пристроїв, виконаних у вигляді друкованих плат (рис. 2.4).

Повністю відкрита архітектура системи дозволяє вільно копіювати або доповнювати пристрої Arduino.

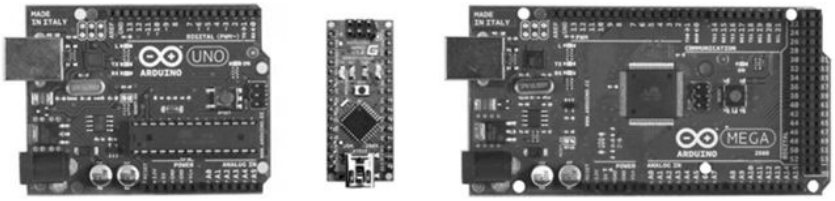


Рис. 2.4. Мікроконтролери Arduino

Для програмування контролерів Arduino використовується середовище розробки Arduino, яка складається з безкоштовної програмної оболонки (IDE) для написання програм, їх компіляції та програмування апаратури.

Мова програмування Arduino є стандартним C++ з особливостями, які полегшують непідготовленим користувачам написання працюючої програми.

Програма складається з двох обов'язкових для Arduino функцій.

Перша функція **setup ()** викликається одноразово при старті та використовується для визначення змінних, встановлення режимів роботи та інших функцій, що задаються один раз.

Друга функція **loop ()** містить прикладну програму, яка виконується в нескінченному циклі.

В тексті своєї програми програміст може вставляти стандартні бібліотеки, які реалізують функції, що часто використовують при проектуванні різних пристроїв, та спрощують програмування при використанні додаткових модулів, таких як датчики, виконавчі пристрої тощо.

Для проектування контролерів Arduino використовується також використовується середа розробки проектів Fritzing (рис. 2.4).

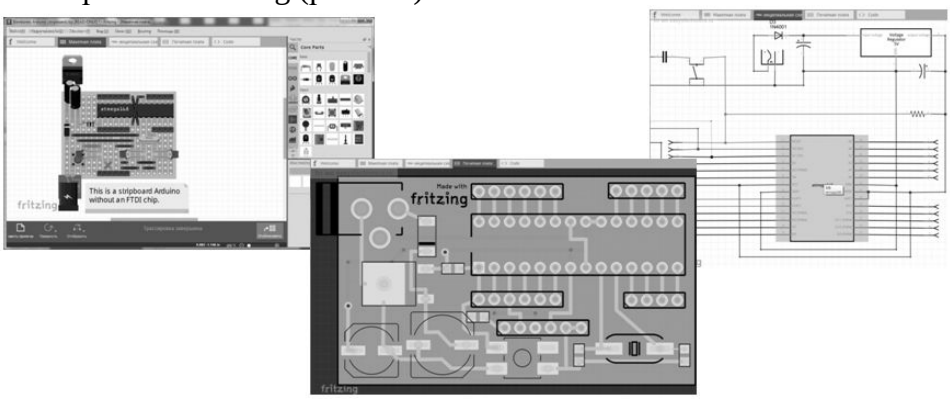


Рис. 2.5. Середа розробки проектів Fritzing

Fritzing дозволяє розробити принципову схему пристрою, і створити її подання до вигляді з'єднання макетів елементів.

Він також дає можливість розробити друковану плату для її подальшого виготовлення.

На відміну від інших систем проектування, у Fritzing простий інтерфейс користувача, який робить розробку електронних схем досить простою.

Вбудовані системи управління найчастіше використовують в спеціалізованих роботах, призначених для автономного використання.

Якщо спеціалізований робот є складовою частиною складних технологічних систем, то для керування таким роботом може здійснювати система керування самої системи. На рис. 2.6 наведений маніпулятор, що вбудован у склад технологічного обладнання.

Рис. 2.6. Маніпулятор у складі технологічного обладнання

Задачі керування складними системами здійснюються за допомогою комплексних систем керування на основі програмованих логічних контролерів.

Комплексна система керування представляє собою багаторівневу систему, що має у своєму складі виконавчі пристрої з локальними системами керування (у робота це керування окремих ланок), пристрої керування, що реалізують алгоритми керування технологічного обладнання, що складається з локальних систем (у робота це узгоджена взаємодія окремих ланок, наприклад, позиційне та контурне переміщення робочого органу), а також пристрої керування, що здійснюють узгоджену взаємодію різного технологічного обладнання.

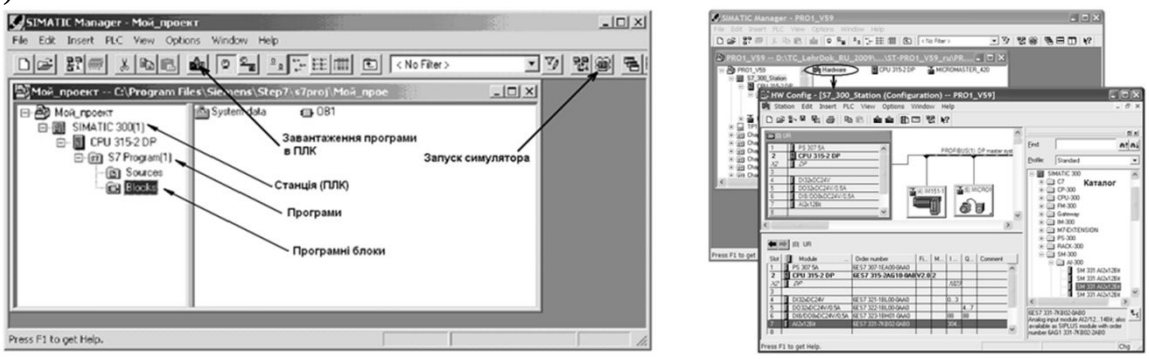
Засоби проектування комплексних систем керування включають засоби проектування окремих компонент.

Комплексні системи керування використовують модульний принцип, тому проектування таких систем зводиться до вибору та налагодженню необхідних модулів, що дають можливість вирішити встановлену задачу.

Розглянемо програмні комплекси проектування програмованих логічних контролерів на прикладі програмного комплексу STEP7 фірми SIEMENS, що включає засоби проектування апаратних компонент системи керування, засоби програмування та засоби пошуку помилок під час налагодження та роботи системи.

Складання проекту системи керування робиться за допомогою усіх цих засобів.

У програмному комплексі STEP7 складання проекту здійснюється в керуючій програмі **SIMATIC-Manager** (рис. 2.7, а). Визначення структури і складу системи керування робиться за допомогою конфігуратора **HW Config**, який є складовою частиною програмного комплексу. Проектування здійснюється шляхом вибору відповідних модулів з каталогу (рис. 2.7, б).



а) б)

Рис. 2.7. Структура проекту в SIMATIC-Manager (а) та конфігуратор HW Config (б)

За допомогою конфігуратора **HW Config** здійснюється також налагодження системи керування.

На рис. 2.8 показано встановлення параметрів для модуля аналогових входів (вибір типу вхідного сигналу та діапазону вимірювання).

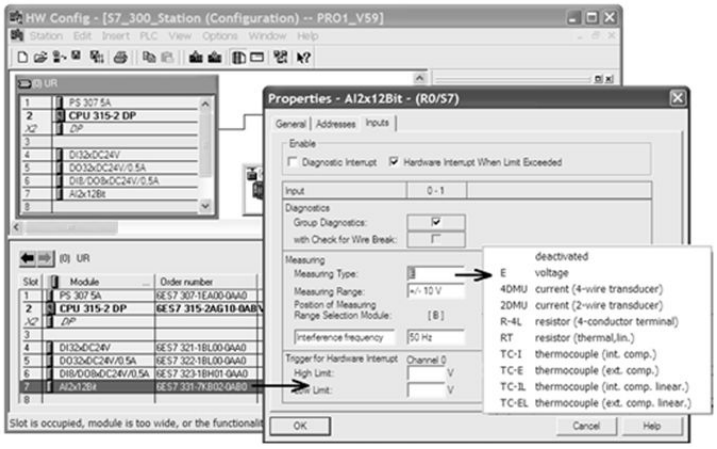


Рис. 2.8. Встановлення параметрів для модуля аналогових входів

У мові програмування STEP7 розрізняють блоки, що містять команди для обробки сигналів (організаційні OB, функції FC і функціональні блоки FB), а також блоки, у яких зберігаються дані (BD).

Для створення програмних блоків OB, FB, FC використовують програмні редактори відповідно з формою представлення програми (рис. 2.9).

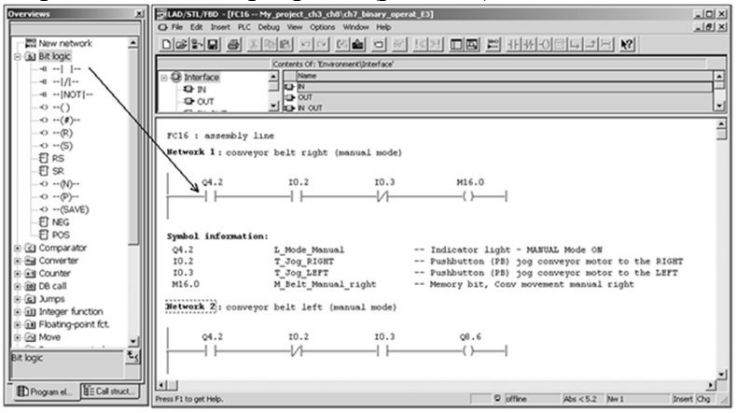


Рис. 2.9. Програмний редактор

Однією з важливіших задач при створенні систем керування є її налагодження та пошук помилок як на стадії проектування так і у період експлуатації.

Програмний пакет STEP7 для ПЛК SIMATIC S7 мають великій набір засобів для вирішення цієї задачі.

На етапі складення програми для її перевірки досить складно використовувати апаратні компоненти системи автоматизації, тому є можливість перевірки програми за допомогою програмної моделі ПЛК S7-PLCSIM (рис. 2.10).

В симулятор можна завантажити програму та конфігурацію ПЛК, запустити її та простежити за реакцією ПЛК, а саме, як вона реагую на зміну вхідних змінних, що імітує сигнали датчиків, або як змінюється внутрішній стан ПЛК у часі, якщо працюють таймери і т.д.

Рис. 2.10. Програмна модель ПЛК S7-PLCSIM

2.2. Засоби проектування універсальних роботів

Для проектування універсальних роботів найчастіше використовують системи автоматизованого проектування, які розроблені фірмами, що випускають роботи, та призначені для певних типів роботів.

Програмний комплекс **ABB robot studio**, призначений для проектування та моделювання роботів фірми ABB [14]. RobotStudio представляє собою симуляційну середу off-line програмування роботів компанії ABB. Основною перевагою off-line програмування є відсутність необхідності в наявності реального обладнання (рис. 2.11).

Переміщення проекту з RobotStudio в контролер робота займає кілька хвилин. При правильно написаної off-line програмою для запуску в режимі on-line потрібна лише невелика корекція координат точок траєкторії робота (наприклад - координат зварних швів).

RobotStudio побудований на ABB VirtualController, точної копії реального програмного забезпечення, яке запускає роботи на виробництві. Це дозволяє виконувати реалістичні симуляції з використанням реальних програм робота і файлів конфігурації, ідентичних тим, які використовуються в цеху.

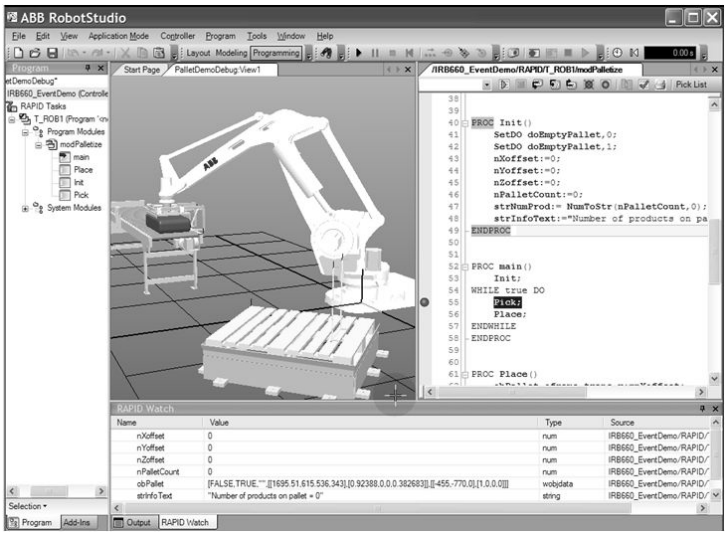


Рис. 2.11. Програмний комплекс **ABB robot studio** для проектування та моделювання роботів фірми ABB

П**рограмний комплекс фірми KUKA**, призначений для проектування та моделювання роботів фірми KUKA [16].

Комплекс має декілька компонент, наприклад, такі як: віртуальний контролер роботів KUKA.OfficeLite (рис. 2.12, ліворуч), та симулятор KUKA.Sim (рис. 2.12, праворуч).

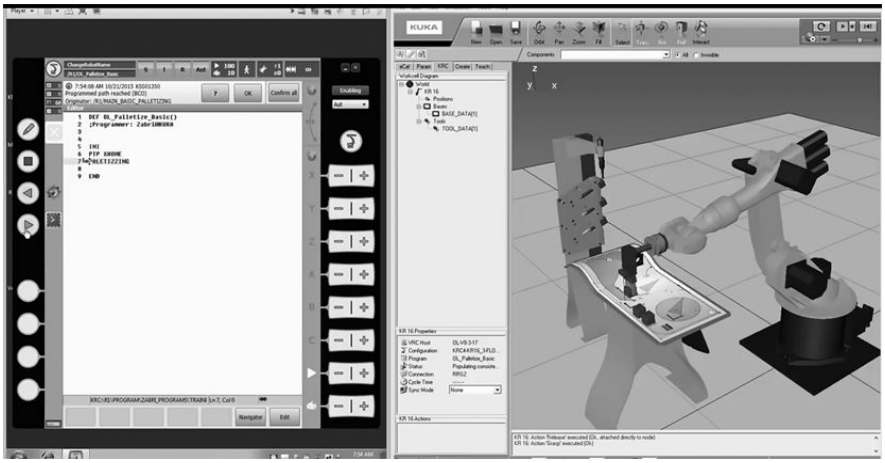


Рис. 2.12. Програмний комплекс фірми KUKA

KUKA.OfficeLite - це віртуальний контролер роботів KUKA. Система програмування дозволяє автономно розробляти і оптимізувати програми на будь-якому комп'ютері. Готові програми можна безпосередньо завантажувати в робот, що гарантує миттєве виконання.

Особливості програми KUKA.OfficeLite

Програмне забезпечення KUKA.OfficeLite практично ідентично системному програмному забезпеченню KUKA System Software. Завдяки використанню інтерфейсу KUKA SmartHMI і синтаксису мови KRL автономне управління і програмування повністю відповідають управлінню і програмуванню робота.

Додаток KUKA.Sim оптимізує роботу систем і роботів для збільшення функціональності і продуктивності завдяки графічному програмуванню в віртуальному середовищі.

Функції програми KUKA.Sim

Маючи інтуїтивно зрозумілий графічний інтерфейс, а також велику кількість функцій і модулів, додаток KUKA.Sim дає можливість знайти оптимальне рішення і максимально можливу ефективність для автономного програмування.

Здійснюється просте створення планів і схем. Схеми оптимального розташування виробничого обладнання можна створити вже на ранній стадії проекту. Розміщення компонентів здійснюється простим перетягуванням за допомогою миші з електронного каталогу.

Електронний каталог і моделювання параметрів. Більшість компонентів з електронного каталогу мають параметричну структуру. Наприклад, можна вибрати захисну огорожу і налаштувати її висоту і ширину відповідно до індивідуальних вимог. Крім іншого, електронний каталог містить велику кількість захоплюючих пристроїв, стрічкових конвеєрів і захисних огороджень.

2.3. Засоби проектування мобільних роботів

Проектування мобільних роботів відрізняється тим, що найчастіше це проектування зводиться до проектування засобів переміщення робота та моделювання переміщення робота з урахування траєкторії переміщення та наявності перешкод. При цьому треба здійснити проектування засобів навігації.

Прикладом програмних комплексів для проектування мобільних роботів є **Microsoft Robotics Developer Studio**, що представляє собою Windows-орієнтоване середовище для керування роботами і їх симуляції та дозволяє проектувати окремі типи роботів, що входять до її каталогу [2].

Microsoft Robotics Studio - це система, спеціально створена для розробки програмного забезпечення для роботів, причому, в основному, для «конструкторів» роботів (заготовок з модулів, які можна перепрограмувати в залежності від завдання, яке треба вирішити) - таких як iRobot Create, LEGO Mindstorm, і т.д.

Microsoft RDS 2008 включає в себе спеціальну програмну модель для створення програм керування, а також набір візуальних та сімуляційних інструментів, які можуть знадобитись при складанні програмного забезпечення для роботів.

Компоненти Robotics Studio інтегріруються в середу розробки Visual Studio, який має два основних модуля.

Це такі модулі:

Visual Programming Language (**VPL**, візуальний язик програмування);

Visual Simulation Environment (**VSE**, сімуляційна середа).

Язик VPL забезпечує можливість програмування роботів візуальними методами (рис. 2.13).

Діаграми VPL кодуються за допомогою XML-схем і дають можливість створення повністю візуального язика програмування.

Другий важливий модуль Robotics Studio - симуляційне середовище VSE. VSE є графічною 3D-моделлю, що відображає дії роботів, і об'єкти, які оточують ці роботи (рис. 2.14).

VSE включає можливість запису та повторного відтворення симуляції. Крім того, проектанти можуть виконувати симуляції у різних віртуальних умовах - в умовах закритого приміщення або відкритого простору.

Крім того, VSE має можливість експорту з різних CAD-програм, наприклад, SolidWorks 3D.

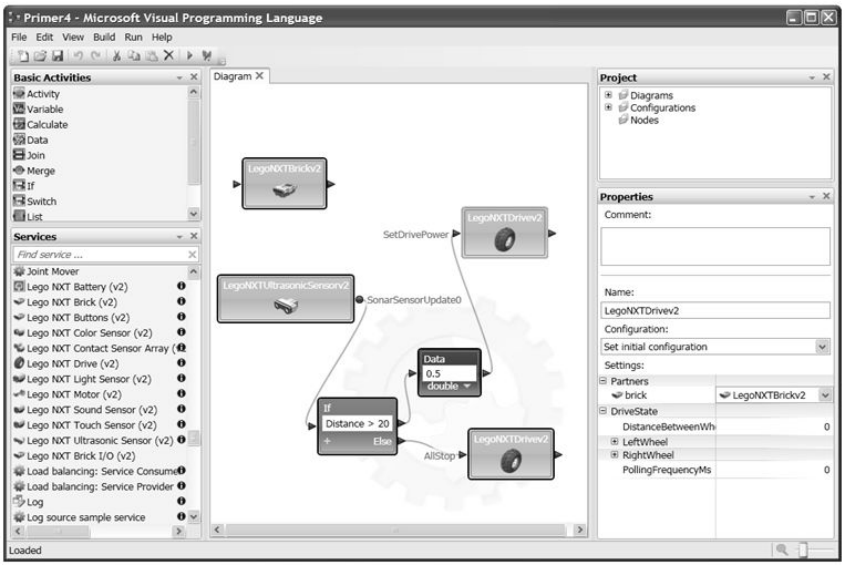


Рис. 2.13. Приклад програми, створеної за допомогою візуального язика програмування VPL

Robotics Studio також дає можливість створювати досить складні програми керування з різними способами програмного керування, а наявність блоків для роботи з датчиками зовнішньої інформації, наприклад, блока для роботи з VEB-камерою, дозволяють реалізувати адаптивне керування на основі обробки зображення.

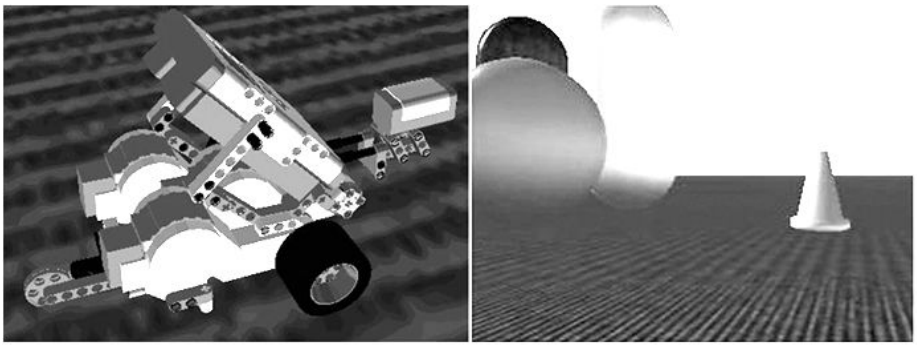


Рис. 2.14. Сімуляційна середа Visual Simulation Environment (VSE)

Іншим прикладом комплексів проектування мобільних роботів є програмний комплекс **Dyn-Soft RobSim 5** (рис. 2.15).

Для розробки моделей роботів використовують такі засоби:

- програмний комплекс 3D Studio MAX, в якому створюються геометричні моделі роботів;
- засоби розробки Dyn-Soft RobSim 5.

Розробка моделі робота для Dyn-Soft RobSim 5 починається в програмному комплексі 3D Studio MAX. При створенні моделі важливо помістити робота на початку координат, орієнтувати передом у напрямку осі Y (вгору на вигляді зверху), а також встановити його колесами або опорами на поверхню Z=0.

Після цього здійснюється розмітка усіх об'єктів та механізмів.

Далі складаються електричні схеми та підключення.

Наприкінці проектування проводиться моделювання робота.

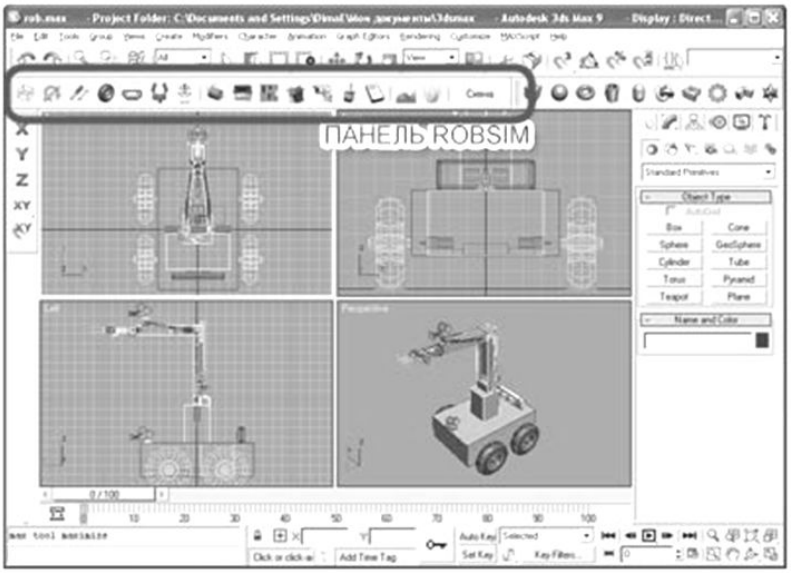


Рис. 2.15. Розробка моделі робота за допомогою Dyn-Soft RobSim 5

Контрольні питання

1. Назвати, які програмні комплекси використовують для проектування спеціалізованих роботів?
2. Розповісти, які засоби використовують для проектування вбудованих пристроїв керування?
3. Описати, які мови програмування використовують для однокристальних мікроконтролерів?
4. Назвати, які засоби використовують для проектування контролерів Arduino?
5. Визначити, для чого використовується середа Fritzing?
6. Які засоби використовують програмний комплекс ABB robot studio?
7. Визначити,з чого складається програмний комплекс фірми KUKA?
8. Описати, які задачі вирішує комплекс Microsoft Robotics Developer Studio?
9. Розповісти, який язик програмування використовує Microsoft RDS?
10. Визначити, які задачі вирішує програмний комплекс Dyn-Soft RobSim 5?

Змістовий модуль 2. Комп'ютерні засоби проектування спеціалізованих роботів

Лекція 3. Засоби проектування вбудованих пристроїв керування спеціалізованих роботів на основі мікропроцесорної техніки

3.1. Засоби проектування систем керування спеціалізованих роботів

Спеціалізовані роботи найчастіше створюються для виконання обмежених функцій, або є складовими частинами комплексних виробничих систем. Такі роботи часто мають циклове керування з обмеженими переміщеннями. Керування такими роботами може здійснюватися за допомогою системи керування усього обладнання (рис. 3.1).

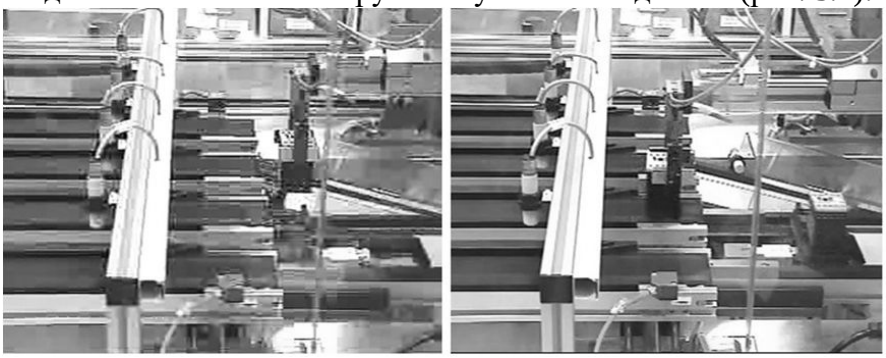


Рис. 3.1. Маніпуляційний робот у складі технологічного обладнання

У складі гнучких виробничих систем також часто використовуються робототехнічні пристрої. Оскільки такі системи найчастіше мають модульну структуру, то в їх складі можуть бути відносно прості автономні роботи, що працюють під керуванням загальної системи керування. Прикладом таких робототехнічних пристроїв, є різні відкриті проекти роботів-маніпуляторів, а саме, маніпулятори uArm - відкритий проект маніпулятора під керуванням Arduino, проект AR2 robot (рис. 3.2) тощо.

Рис. 3.2. Маніпулятори uArm, AR2

До робототехнічних пристроїв можна також віднести 3D-принтери, що знайшли досить широке використання. Системи керування таких пристроїв також робляться на основі мікроконтролерів. На рис. 3.3 наведений 3D-принтер на основі Arduino Mega 2560.

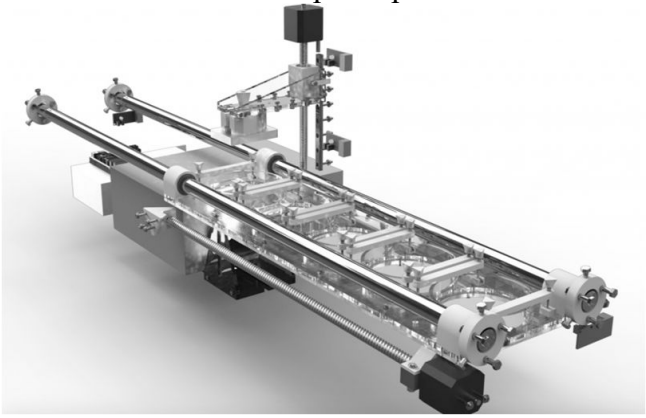


Рис. 3.3. 3D-принтер на основі Arduino Mega 2560

На рис. 3.4 показані компоненти 3D-принтера , що складаються в основному з крокових двигунів та пристрою керування.

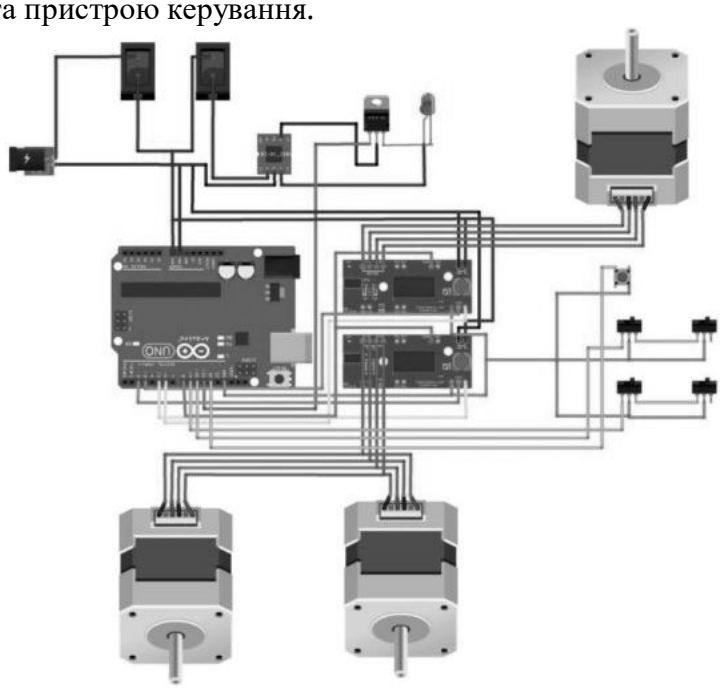


Рис. 3.4. Компоненти 3D-принтера

Основою пристроїв керування спеціалізованих робототехнічних пристроїв є однокристальні мікроконтролери, які є універсальними пристроями.

Програмування однокристальних мікроконтролерів зазвичай здійснюється з використанням таких мов, як Асемблер або С, хоча існують компілятори для інших мов, використовуються також інтерпретатори Бейсіка і Форту.

Одним із найбільше широко застосовуваних представників цього класу є однокристальний мікроконтролер 8051 (ОМК 51) [5], розроблений фірмою INTEL наприкінці 70-х, початку 80-х років, і що випускається з невеличкими змінами і доповненнями в даний час різними фірмами.

ОМК 51 являє собою закінчений обчислювальний пристрій, що включає центральний процесор (арифметично-логічний пристрій), внутрішню пам'ять, набір регістрів спеціальних функцій, устрій керування і порти вводу та виводу, що об'єднуються 8-розрядною шиною.

На сьогоднішній день існує більше 200 модифікацій мікроконтролерів, сумісних з i8051, що випускаються двома десятками компаній, і велика кількість мікроконтролерів інших типів.

В останній час найчастіше використуються такі мікроконтролери:

- 8-бітові мікроконтролери PIC фірми Microchip Technology,
- мікроконтролери AVR фірми Atmel,
- 16-бітові MSP430 фірми TI,
- 32-бітові мікроконтролери, архітектури ARM, яку розробляє фірма ARM Limited і продає ліцензії іншим фірмам для їх виробництва.

У AVR-мікроконтролерів є особливості, які відрізняють це сімейство від решти МК. А саме, система команд і архітектура ядра AVR розроблялися спільно з фірмою-розробником компіляторів з мов програмування високого рівня IAR Systems Ltd.

Тому структура AVR-МК максимально оптимізована для того, щоб писати програми на мовах високого рівня. В результаті програми на мові С для AVR-МК незначно втрачають в продуктивності у порівнянні з програмами, написаними на мові асемблера.

Програмування контролерів на основі i8051 здійснюється на основі мов програмування Асемблер або С.

Програмування контролерів на основі AVR здійснюється в основному з використанням мови програмування С.

Далі розглянемо засоби проектування програмного забезпечення для мов програмування Асемблер та С, відповідно, на прикладах контролерів на основі i8051 та AVR.

3.2. Засоби проектування на основі мови програмування Асемблер

Кожна програма незалежно від язика програмування має бути перетворена у програму, команди якої відповідають командам центрального процесора, поданим у двійковому коді. Це зв'язано з тим, що програма, яку виконує процесор, знаходиться у пам'яті у двійковому коді (так звані машинні коди). Користуватись машинними кодами незручно, тому замість машинних кодів використовуються символічні позначення.

Для програмування простих мікропроцесорних пристроїв використовується язик програмування, команди якої відповідають командам центрального процесора. Такий язик є машинноорієнтованим язиком та називається Асемблер незалежно від типу процесора.

Текст програми на Асемблері складається як послідовність команд у визначеному форматі, коли кожна команда являє собою рядок з чотирьох конструкцій (чотири поля):

МІТКА :	**ОПЕРАЦІЯ**	**ОПЕРАНДИ**	**;КОМЕНТАР**
WAIT0:	JNB	P2. 0,WAIT0	;Очікування замикання ;контакту

Мітка являє собою символічне ім'я адреси пам'яті, де зберігається відзначена команда або операнд.

Операція являє собою символічне позначення команди або псевдокоманди мови Асемблер.

Псевдокоманди не мають аналогів у системі команд центрального процесора та виконують додаткові функції, наприклад, встановлення початкової адреси програми.

У полі операнду визначаються дані, що приймають участь у команді.

Операнд може бути заданий безпосередньо у вигляді числа або у вигляді його адреси.

Коментар використовується для пояснення виконуваних команд.

Мова програмування Асемблер використовується як мова системного програмування, або для складання програм для відносно простих пристроїв керування, наприклад, на основі однокристальних мікроконтролерів.

Програма на мові Асемблер створюється у текстовому редакторі (рис. 3.5, а), а після компіляції за допомогою транслятора перетворюється у машинну мову, що складається з двійкових кодів команд (рис. 3.5, б).

```
//////////////////////////////////////////
; головной модуль для генератора с timer0
$NOMOD51
clock   EQU     0FF00H
        PUBLIC  clock

TMOD    DATA  089H
TL0     DATA  08AH
TH0     DATA  08CH
IE      DATA  0A8H
TR0     BIT   088H.4
requ    BIT   090H.0

        PUBLIC  main
        NAME    MAINPRG
;=======================
; сегмент: головная процедура
MYMAIN  SEGMENT CODE
        RSEG    MYMAIN
;********************** main ************
main:
        ORL     TMOD,#01H
        MOV     TL0,#LOW( clock )
        MOV     TH0,#HIGH( clock )
        MOV     IE,#082H
WaitRequ:
        JB      requ,WaitRequ
        SETB    TR0
Loop:
        SJMP    $
        END
```

а)

```
LOC  OBJ            LINE     SOURCE
                       1
                       2     ; головной модуль для генератора с
0timer0
                       3     $nomod51
  FF00                 4     clock   EQU     0FF00H
                       5             PUBLIC  clock
                       6
  0089                 7     TMOD    DATA  089H
  008A                 8     TL0     DATA  08AH
  008C                 9     TH0     DATA  08CH
  00A8                10     IE      DATA  0A8H
  008C                11     TR0     BIT   088H.4
  0090                12     requ    BIT   090H.0
                      13
                      14             PUBLIC  main
                      15             NAME    MAINPRG
                      16     ;=======================
                      17     ; сегмент: головная процедура
                      18     MYMAIN  SEGMENT CODE
----                  19             RSEG    MYMAIN
                      20     ;********************** main
0**************************
0000                  21     main:
0000 438901           22             ORL     TMOD,#01H
0003 758A00           23             MOV     TL0,#LOW( clock )
0006 758CFF           24             MOV     TH0,#HIGH( clock )
0009 75A882           25             MOV     IE,#082H
000C                  26     WaitRequ:
000C 2090FD           27             JB      requ,WaitRequ
000F D28C             28             SETB    TR0
0011                  29     Loop:
0011 80FE             30             SJMP    $
                      31             END
```

б)

Рис. 3.5. Програма на мові Асемблер

Розглянемо, як складається програма на Асемблері на прикладі **позиційного керування** роботом при переміщенні вздовж стелажу.

Для **простого позиційного керування** роботом можна використати датчик із щілинним чутливим елементом, який видає сигнал при проходженні металевої пластини через щілину. Позиціонування здійснюється підрахуванням числа імпульсів при проходженні датчика вздовж осі з пластинами (рис. 3.6).

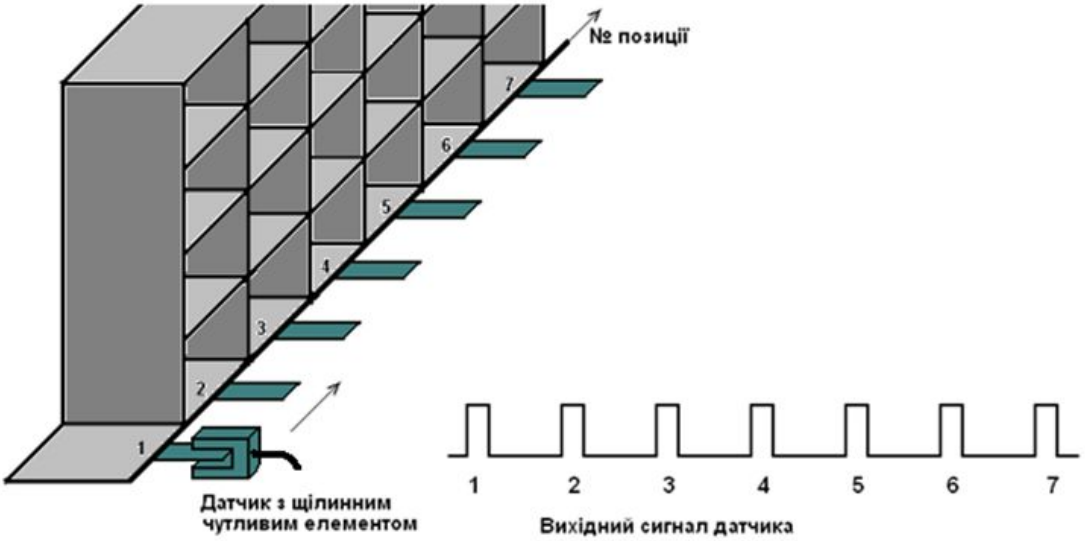


Рис. 3.6. Позиційне керування роботом при переміщенні вздовж стелажу

Для цього може бути використаний програмний лічильник, який можна здійснити за допомогою команди:

DJNZ Ri, L1 - зменшити на "1" значення регістра Ri, та перейти на мітку L1, якщо його значення не дорівнює "0.

Далі приведена програма позиціонування, що працює таким чином:

- на початку у визначений регістр записується задане число циклів та включається двигун,
- потім програма запускає цикл, де опитує датчик на наявність позитивного імпульсу ("0"->"1"->"0"),
- та фіксує момент переходу з "1" у "0" (задній фронт),
- після чого його вміст зменшується на одиницю в кожному наступному циклі доти, поки воно не стане рівним нулю,
- після цього двигун вимикається.

На рис.3.7 наведена блок-схема програми позиційного керування.

Програма позиційного керування для переходу на позицію 6 має такий вигляд:

```
        MOV  R4, # 6        ;записати номер позиції у регістр R4 (число 6)
        SETB  P1.1          ;включити двигун переміщення
L1:     JNB    P2. 1,L1     ;очікування початку імпульсу на вході P2.1=1
L2:     JB     P2. 1,L2     ;очікування кінця імпульсу на вході P2.1=0
        DJNZ  R4,L1          ;зменшити на "1" значення регістра R4,
                             ;та перейти на мітку L2, якщо його
                             ; значення не дорівнює "0";
        CLR   P1.1          ;виключити двигун переміщення
```

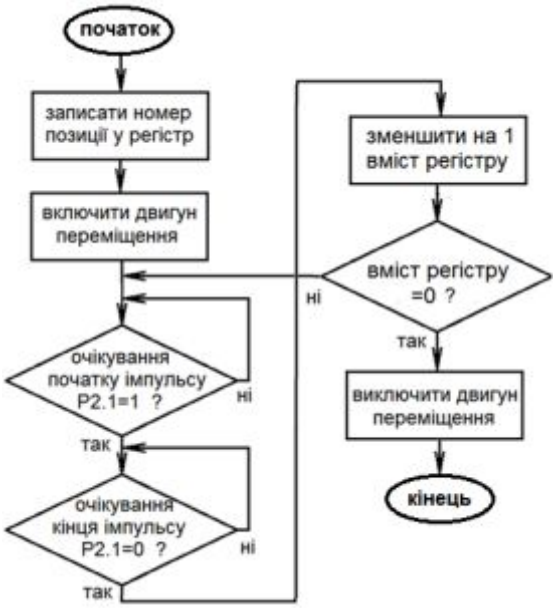


Рис. 3.7. Блок-схема програми позиційного керування

Система команд (інструкцій) однокристальних мікроконтролерів включає такі команди:

- команди пересилки, за допомогою яких провадиться обмін даними усередині однокристальних мікроконтролерів ;
- арифметичні команди над двійковими числами, що включають команди складання, вирахування, множення і ділення;
- команди логічних операцій над двійковими числами, виконувані порозрядно (логічне І, логічне АБО, виключне АБО);
- команди переходу, що включають умовні і безумовні команди передачі керування по зазначеній адресі і команди виклику підпрограм;
- команди логічних операцій із бітами;
- спеціальні команди.

Програма, що написана мовою асемблера перетворюється у машинний код за допомогою утиліти, яка називається Асемблер.

Крім інструкцій, програма може містити директиви: команди, що не переводяться безпосередньо в машинні інструкції, а керують роботою компілятора. Набір і синтаксис їх значно різняться і залежать не від апаратної платформи, а від використовуваного компілятора.

У наборі директив можна виділити:

- визначення даних (констант і змінних);
- управління організацією програми в пам'яті і параметрами вихідного файлу;
- завдання режиму роботи компілятора;
- всілякі абстракції (тобто елементи мов високого рівня) — від оформлення процедур і функцій (для спрощення реалізації парадигми процедурного програмування) до умовних конструкцій і циклів (для парадигми структурного програмування);
- макроси.

(парадигма програмування визначає організацію обчислень і структуру роботи, яку виконує комп'ютер)

Процес перетворення називають асемблюванням або збіркою (англ. *assembly, assembling*). У більшості випадків цей процес відбувається у такі етапи: асемблювання, компонування, завантаження та налагодження програми, для чого відповідно використовують такі компоненти програмних засобів для створення програми: асемблер, редактор зв'язків або компонувальник, завантажувач, налагоджувач.

Асемблер перетворю програму, що написана в текстовому редакторі, в об'єктний модуль, що містить в собі особливим чином підготовлений бінарний код, який може бути

зв'язаний з іншими об'єктними файлами за допомогою редактора зв'язків для отримання готового модуля, що може виконуватись.

Редактор зв'язків або компонувальник це програма збирає декілька об'єктних модулів та об'єднує їх в один модуль, що може виконуватись.

Завантажувач здійснює завантаження модуля, що може виконуватись, в пам’ять контролера.

Налагоджувач це програма, яка використовується для тестування та виправлення прикладної програми. Часто для цього використовують програмний емулятор, що моделює роботу процесора.

3.3. Засоби проектування на основі мови програмування С

Засоби проектування систем керування для роботів на основі мікроконтролерів найчастіше використовують мову С, яка на відміну від Асемблера, що має набір команд відповідно до системи команд, яка специфічна для кожного типу процесора, є універсальною мовою, хоча і може мати деякі особливості [6].

Наприклад, мова програмування для контролерів Arduino (середовище розробки Arduino IDE) теж має деякі особливості [6].

Програма Arduino, яку називають скетч, складається з самостійного файла, в якому, на відміну від мови С, треба визначити принаймні, дві секції: перша setup(), друга loop().

Як тільки програма запуститься, виконуються операнди, розміщені в блоці setup(): вони призначені для ініціалізації значень змінних на початку запуску, а також для налаштування портів периферії Arduino. Після закінчення обробки setup() Arduino починає циклічне виконання інструкцій в блоці loop(). Після виконання всіх операндів, цикл повторюється знову і знову.

Для зберігання проміжних даних обчислень програма використовує змінні різних типов. Для обчислень можуть бути використані дані різних форматів, різної розрядності, типи даних вибираються виходячи з необхідної точності обчислень, форматів даних і т.п.

Всі змінні повинні бути оголошені до того як будуть використовуватися.

Для спрощення програми можна використовувати стандартні константі.

Розрахунки здійснюються за допомогою таких функцій та операторів: арифметичні операції, операції відношення, логічні операції, квадратний корінь та квадрат числа, тригонометричні та зворотні тригонометричні функції, оператори управління програмою, які визначають послідовність виконання програми, функції управління вводом / виводом, тощо.

Більш детально функції та оператори розглянуті у додатку 1.

Для зв'язку комп'ютера з контролером існують засоби обміну даними, такі як монітор порту Ардуіно та Плотер по послідовному з'єднанню, який видає дані у символьному вигляді, та плотер по послідовному з'єднанню видає дані у графичному вигляді.

На рис. 3.8 наведений приклад виведення даних на Монитор порта, а на рис. 3.9 на Плотер по послідовному з'єднанню.

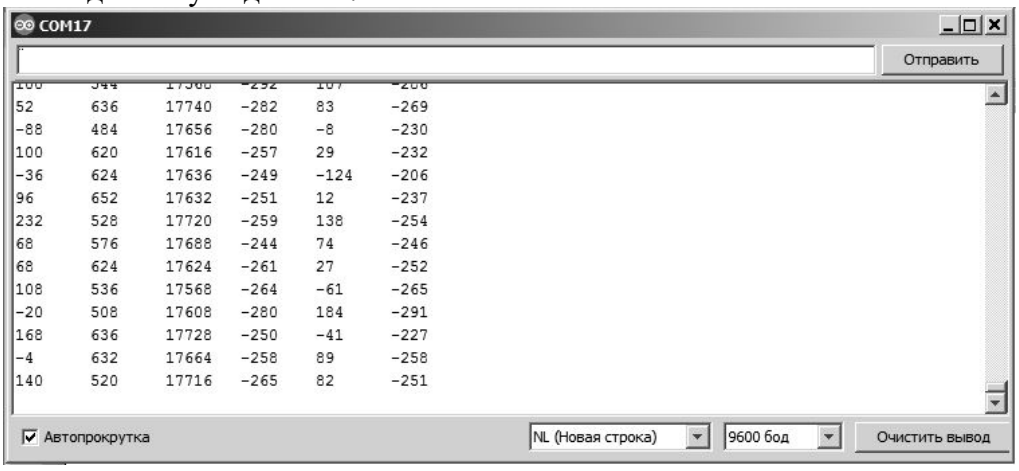


Рис. 3.8. Приклад виведення даних на Монитор порта

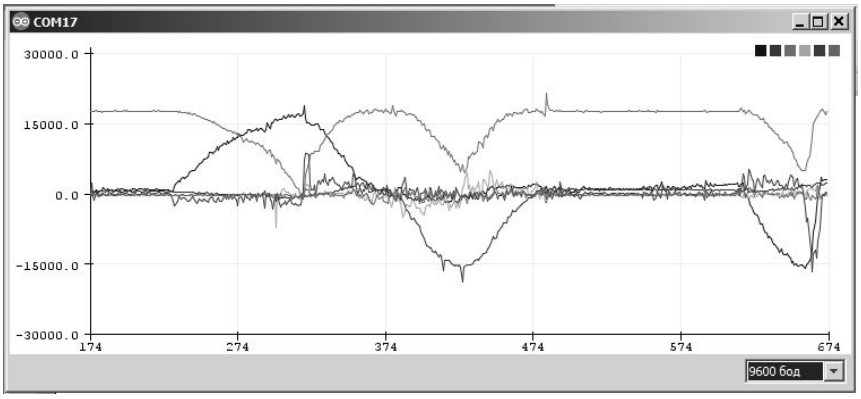


Рис. 3.9. Приклад виведення даних на Плотер по послідовному з'єднанню

Можливості середовища розробки Arduino IDE для комп’ютерних розрахунків роботів наведені у додатку 1.

У додатку 2 покана можливість використання мови Arduino IDE для обчислення параметрів з метою керування маніпулятором на основі зворотної задачі кінематики.

3.4. Засоби графічного проектування програмного забезпечення

Прикладом графічного проектування програмного забезпечення є середа розробки проектів Fritzing, яка згадувалась у підрозділі 2.1, а також графічне середовище програмування ArduBlock, що вбудовується в середу програмування Arduino IDE.

Програмування у цьому середовищі здійснюється за допомогою значків, що відповідають змінним, а також окремим командам та функціям, які треба вибрати з каталогу.

На рис. 3.10 наведений розділ «Управління», де можна знайти різноманітні цикли.

В розділі «Порти» можна керувати значеннями портів, а також підключеними до них датчиками та виконавчими пристроями. (рис. 3.11).

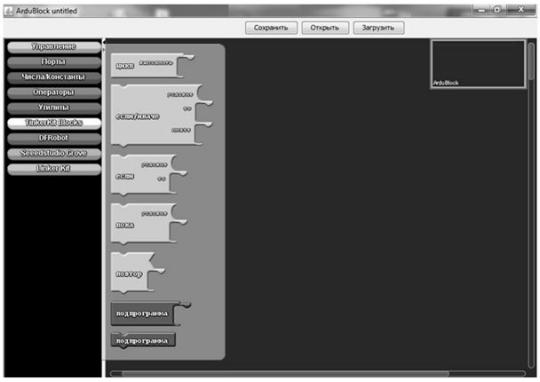


Рис. 3.0 Розділ «Управління»

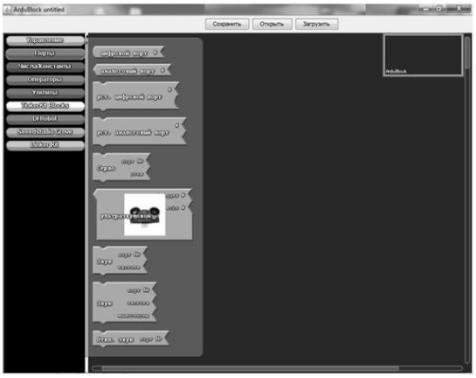


Рис. 3.11. Розділ «Порти»

В розділі «Числа/Константи» можна вибрати цифрові значення або створити змінну (рис. 3.12).

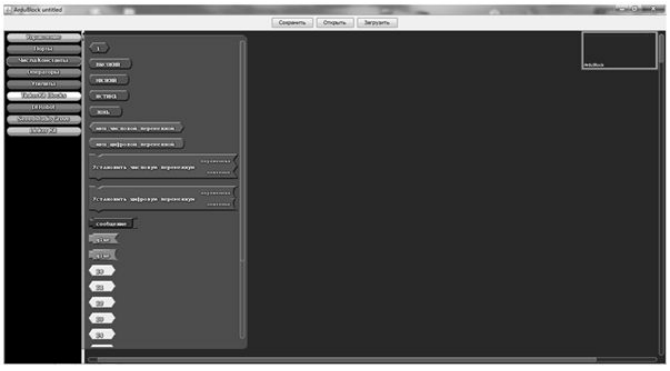


Рис. 3.12. Розділ «Числа/Константи»

В розділі «Оператори» можа знайти всі необхідні оператори порівняння та обчислення (рис. 3.13).

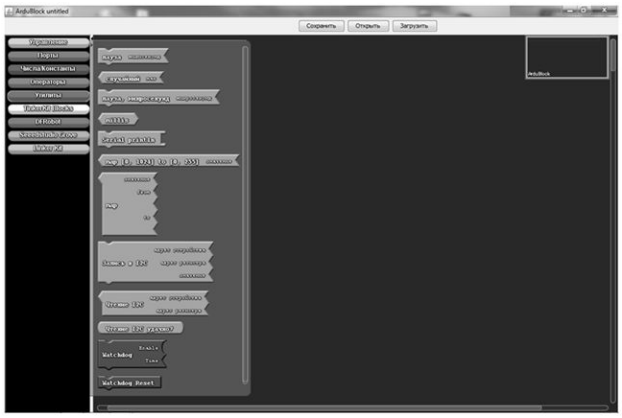


Рис. 3.13. Розділ «Оператори»

Програмування здійснюється шляхом вибору усіх необхідних компонент і після завантаження в Arduino IDE має вигляд, що наведений на рис. 3.14. Після цього програма може бути завантажена в контролер Arduino.

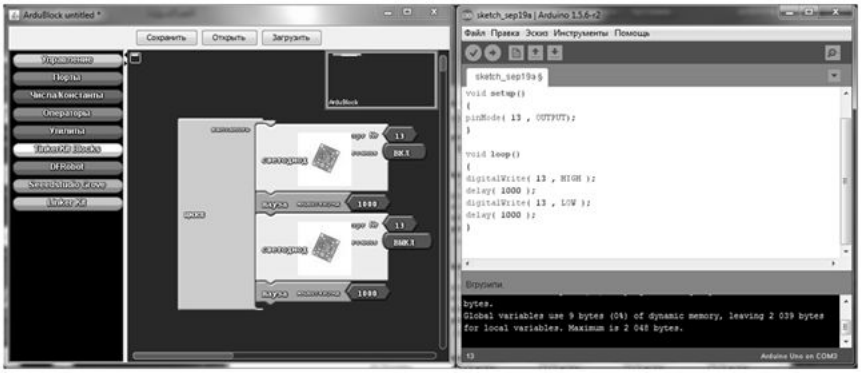


Рис. 3.14. Програма, що складена за допомогою ArduBlock

Контрольні питання

1. Розповісти, для чого найчастіше використовують спеціалізовані роботи?
2. Описати, чому 3D-принтери можна віднести до спеціалізованих роботів?
3. Назвати, які мови програмування найчастіше використовують для однокристальних мікроконтролерів?
4. Розповісти, які особливості має ОМК 51?
5. Описати, з чого складається текст програми на Асемблері?
6. Назвати, які команди можна використати для позиційного керування?
7. Які типи даних можна використовувати в мові С?
8. Розповісти, які варіанти управлення програмою можна використовувати в мові С?
9. Визначити, які задачі вирішує середовище програмування ArduBlock?
10. Назвати, як здійснюється програмування за допомогою середовища програмування ArduBlock?

Лекція 4. Приклади робототехнічних систем на основі контролерів Ардуино

4.1. Проектування систем керування

Апаратно-програмний комплекс Arduino представляє собою набір апаратно-програмних засобів, пристосованих для побудови простих систем автоматики і робототехніки [6].

Апаратна частина являє собою набір готових пристроїв, виконаних у вигляді друкованих плат.

Повністю відкрита архітектура системи дозволяє вільно копіювати або доповнювати лінійку продукції Arduino.

Arduino може використовуватися як для створення автономних об'єктів автоматики, так і підключатися до програмного забезпечення на комп'ютері через стандартні дротові та бездротові інтерфейси.

Як показано на рис. 4.1, є велика кількість різних датчиків і виконавчих пристроїв, що дозволяють вирішувати широке коло завдань.

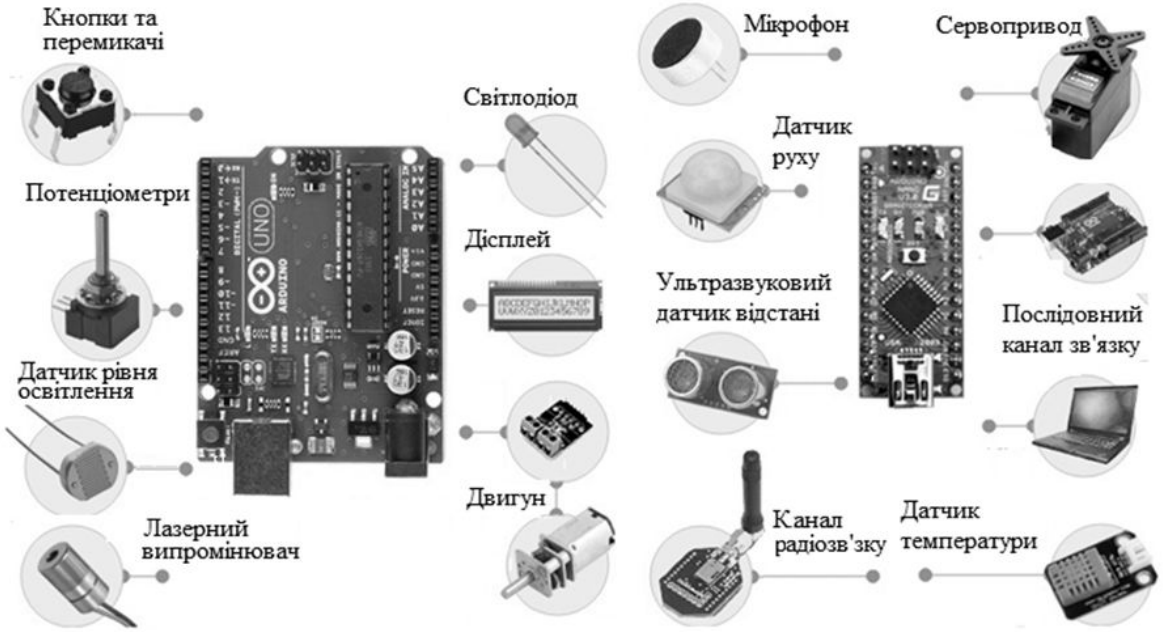


Рис. 4.1. Апаратні компоненти комплексу Arduino

При цьому є як різні датчики, у тому числі датчики руху, так і модулі регульованих приводів для електродвигунів постійного струму і крокових двигунів, а також сервоприводи, що дозволяють здійснити поворот на заданий кут.

Мікроконтролери для Arduino відрізняються наявністю попередньо встановленого в них завантажувача (bootloader).

За допомогою цього завантажувача користувач завантажує свою програму в мікроконтролер без використання окремих апаратних програматорів.

Завантажувач з'єднується з комп'ютером через інтерфейс USB (якщо він є на платі) або за допомогою відповідного перехідника.

У лінійці пристроїв Arduino в основному застосовуються мікроконтролери Atmel AVR: ATmega328, ATmega168, ATmega2560, ATmega32U4, ATTiny85 з тактовою частотою 16 або 8 МГц.

Контролер ArduinoUno, побудований на ATmega328. Платформа має 14 цифрових входів / виходів (6 з яких можуть використовуватися як виходи ШІМ), 6 аналогових входів, роз'єм USB, і кнопку скидання. Розмір плати 6,9 × 5,3 см (рис. 4.2, а).

Контролер Arduino Nano, побудований на ATmega328. Платформа має 14 цифрових вхід / виходів (6 з яких можуть використовуватися як виходи ШІМ), 8 аналогових входів, роз'єм Mini USB. Відмінною особливістю є малі розміри (1.85 × 4.2 см), що дозволяють вбудовувати контролер в портативні пристрої (рис. 4.2, б).

а) б)

Рис. 4.2. Контролери ArduinoUno (а) та Arduino Nano (б)

Контролер Arduino Mega 2560 виконаний на основі мікроконтролера ATmega2560. У його склад входить: 54 цифрових входів / виходів (з яких 15 можуть використовуватися як виходи ШІМ), 16 аналогових входів, 4 апаратних приймально-передавачів для реалізації послідовних інтерфейсів, роз'єм USB, роз'єм живлення, роз'єм ICSP для програмування і кнопка скидання. Розмір плати 10,8 × 5,3 (рис. 4.3).

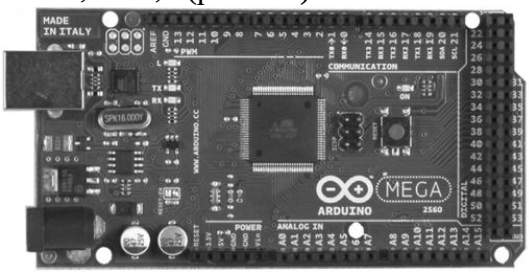


Рис. 4.3. Контролер ArduinoUno

Arduino Robot - версія Ардуіно, в конструкції якого передбачені колеса.

Робот складається з двох плат, кожна з яких містить свій мікропроцесор.

Плата приводів (Motor Board) контролює роботу двигунів, а керуюча плата (Control Board) зчитує показання датчиків і приймає рішення про подальші операції.

Кожна з двох плат є повноцінний пристрій Ардуіно, програмований за допомогою середовища розробки Arduino IDE (рис. 4.4).

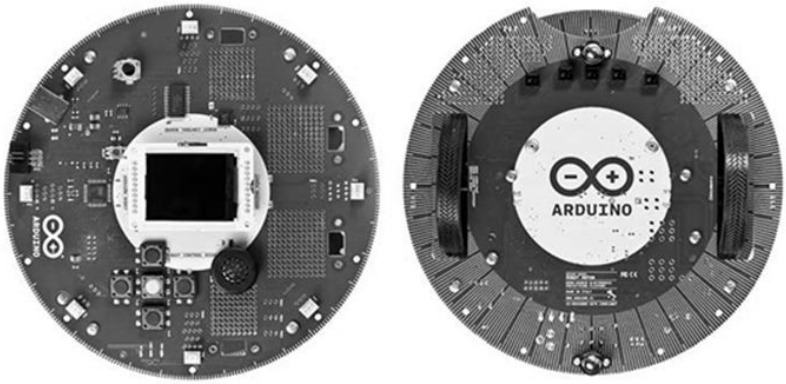


Рис. 4.4. Контролер Arduino Robot

Проектування системи керування робота на основі апаратно-програмного комплексу Arduino здійснюється шляхом вибору необхідних апаратних компонент та розробки програми керування.

Розглянемо деякі питання, що виникають при складання програми керування.

При використанні роботів виникають потреби у вибору режиму роботи, наприклад, ручний та автоматичний режими, а також режим навчання робота.

Маємо три режими, тому для перемикання режимів роботи використовуємо перемикач з трьома положеннями, що підключені до двох входів (рис. 4.5).

У вихідному положенні (сигнал на входах відсутній) на входах D8, D9 встановлений сигнал HIGH (режим INPUT_PULLUP). При переключенні перемикача у положення switch1 або switch2 на відповідний вхід подається сигнал LOW.

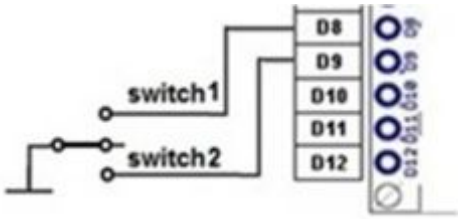


Рис. 4.5. Підключення перемикача режимів

Використовуємо оператор **ELSE IF**, який дозволяє зробити багаторазовий вибір.

```
if (switch1 == LOW)       // якщо switch1 дорівнює 0,
  {
    …                     // виконується цей блок
   }
else if (switch2 == LOW)    // якщо switch2 дорівнює 0,
  {
…                  // виконується цей блок
   }
else   // в іншому випадку, коли на switch1 та switch2 сигнал HIGH
  {
  …                        // виконується цей блок
   }
```

У залежності від режиму роботи робота, або від інформації, що отримана після обробки даних з інформативних пристроїв, здійснюється відповідне переміщення робочих органів та засобів переміщення.

4.2. Проектування виконавчих пристроїв

Для регулювання швидкості двигуна постійного струму найчастіше використовують мостові драйвери, які можуть також керувати кроковими двигунами.

Розглянемо, як працюють модулі управління електродвигунами на прикладі драйвера двигуна L298N (рис. 4.6).

Рис. 4.6. Драйвер двигунів постійного струму та крокових двигунів

Модуль виконаний на основі мікросхеми L298N, яка представляє собою здвоєний мостовий драйвер двигунів і призначена для управління двигунами постійного струму і кроковими двигунами.

Ця схема дозволяє управляти двома двигунами постійного струму або одним кроковим двигуном з струмом до 2 А. Напруга живлення 5 - 35 В.

Спрощена принципова схема мікросхеми L298N приведена на рис. 4.7

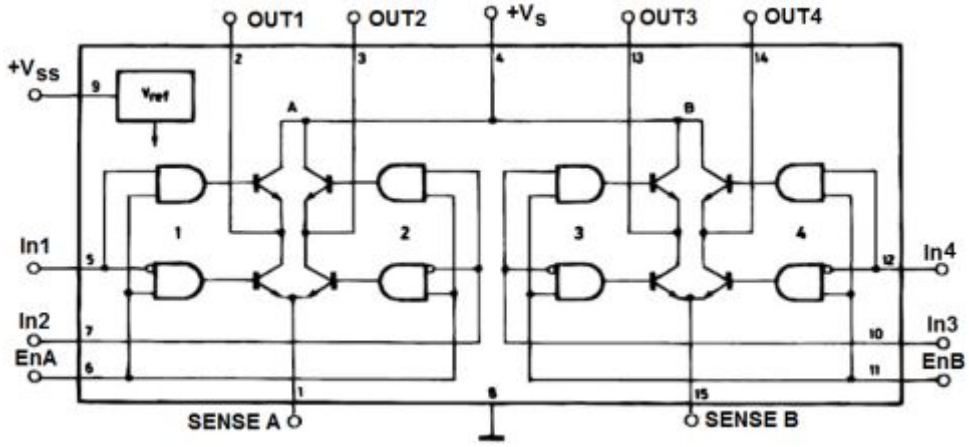


Рис. 4.7. Спрощена принципова схема мікросхеми L298N

Підключення двигунів наведено на рис. 4.8.

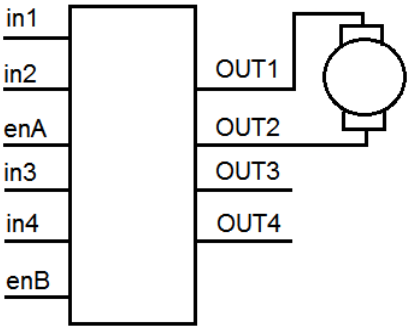


Рис. 4.8. Підключення двигунів

При подачі на вхід In1 (In3) значення 0 (LOW), а на вхід In2 (In4) значення 1 (HIGH) двигун, що підключається до виходів OUT1 (OUT3) і OUT2 (OUT4), обертається в одну сторону, а при подачі на вхід In1 (In3) значення 1 (HIGH), а на вхід In2 (In4) значення 0 (LOW) двигун обертається в іншу сторону.

При подачі на входи In1 (In3) і In2 (In4) однакових значень двигуни не обертаються.

Для регулювання вихідної напруги сигнали ШІМ подаються, відповідно, на входи EnA і EnB за допомогою функції **analogWrite(pin, value)**, де pin – вихід, на яких подається сигнал ШІМ, value – виначає тривалість імпульсу (1 – 255).

Скетч, що наведений нижче, показує, як здійснюється керування двигуном постійного струму, якиц підключений до виходів OUT1 та OUT2.

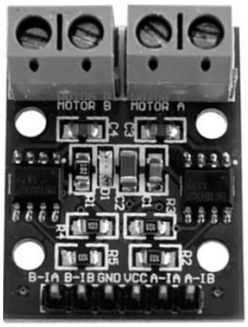
```
// підклюння двигуна до цифрових пінів Arduino
int enA = 10;
int in1 = 9;
int in2 = 8;

void setup()
{
// визначаємо усі піни для управління двигунами
pinMode(enA, OUTPUT);
pinMode(in1, OUTPUT);
pinMode(in2, OUTPUT);
}

void loop()
{
// запуск двигуна
digitalWrite(in1, HIGH);
```

```
digitalWrite(in2, LOW);
// встановлюємо швидкість ШІМ
analogWrite(enA, 200);
delay(2000);
// виключаємо двигуни
digitalWrite(in1, LOW);
digitalWrite(in2, LOW);
delay(1000);
// змінюємо напрямок обертання
digitalWrite(in1, LOW);
digitalWrite(in2, HIGH);
delay(2000);
// виключаємо двигун
digitalWrite(in1, LOW);
digitalWrite(in2, LOW);
delay(1000);
}
```

Драйвер мотора L9110 дозволяє підключати два мотора постійного струму з напругою 2,5-12 вольт, або один двофазний кроковий двигун (рис. 4.9).

Завдяки малим розмірам, драйвер зручно використовувати в маленьких пристроях.

На відміну від драйвера L298N має тільки 4 входи і тому для керування швидкістю потребує 4 сигнали з ШІМ.



Motor Control Interface

Pin	Description
B-IA	Motor B Input A
B-IB	Motor B Input B
GND	Ground
VCC	Operating Voltage 2.5-12V
A-IA	Motor A Input A
A-IB	Motor A Input B

Рис. 4.9. Драйвер мотора L9110

Розглянемо, як здійснюється керування кроковим двигуном контролером Arduino за допомогою драйвера L298N.

Використовується двигун, що має 200 кроків на оборот і може працювати з частотою / обертання 60 об хв.

На рис. 4.10 наведена схема підключення крокового двигуна

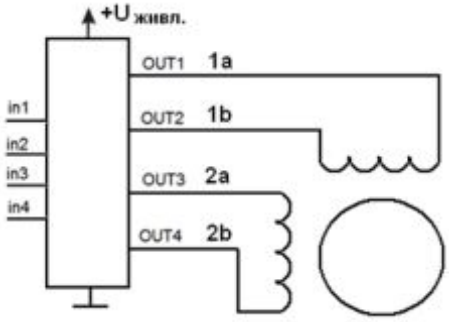


Рис. 4.10. Схема підключення крокового двигуна

Контакти модуля L298N IN1, IN2, IN3 и IN4 підключені до відповідних цифрових контактів D8, D9, D10 и D11 у режимі виходів на Arduino.

Для керування кроковим двигуном використовується бібліотека Stepper Library в Arduino IDE.

Ця бібліотека має наступні функції.

Функція встановлення параметрів крокового двигуна:

Stepper myStepper = Stepper(steps, pin1, pin2, pin3, pin4);

де myStepper – ім'я крокового двигуна,

steps - кількість кроків на одне обертання,

pin1, pin2, pin3, pin4 - виходи контролера, що використовуються для керування двигуном (у залежності від типа двигуна використовують два або чотири виходa).

Функція встановлення швидкості обертання двигуна myStepper:

myStepper.setSpeed(speed);

де speed - швидкість обертання у кількості обертів за хвилину.

Функція встановлення кількості кроків, на яке буде обертатися двигун myStepper (позначка – означає, що двигун буде обертатися в іншу сторону):

myStepper.step(steps);

де steps - кількість кроків.

Далі наведений приклад програми, яка здійснює одне обертання в одну сторону та одне в іншу. Між циклами обертання затримка 0,5 с.

```
#include <Stepper.h>
const int stepsPerRevolution = 200;  // кількість кроків на одне обертання
// підключення до виходів 8 - 11:
Stepper myStepper(stepsPerRevolution, 8, 9, 10, 11);
void setup() {
  // встановлення швидкості 60 об/хв.:
  myStepper.setSpeed(60);
}
void loop() {
  myStepper.step(stepsPerRevolution);         // обертання в одну сторону
  delay(500);                                 // затримка 0,5 с
  myStepper.step(-stepsPerRevolution);        // обертання в іншу сторону
  delay(500);                                 // затримка 0,5 с
}
```

У разі необхідності здійснити поворот на певний кут використовуються сервоприводи (рис. 4.11). Під сервоприводом в даному випадку розуміють механізм з електромотором, який можна повернути в заданий кут і утримувати в цьому положенні.

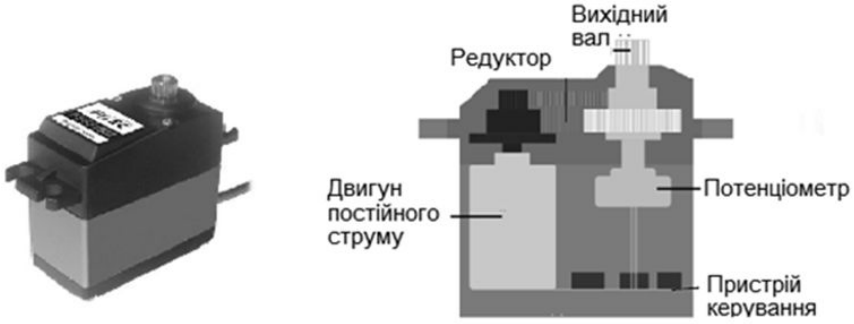


Рис. 4.11. Сервопривод

Для управління сервоприводами використовується широтно-імпульсна модуляція. При цьому кут повороту визначається тривалістю імпульсу та найчастіше має діапазон від 0° до 180°.

Зворотний зв’язок по куту обертання забезпечує потенціометр, що встановлений на вихідному валу.

Приклад використання бібліотеки Servo для керування сервоприводом

```
#include <Servo.h>
Servo myservo;
void setup()
```

```
{
myservo.attach(9);
}
void loop()
{
myservo.write(90);  // встановлює сервопривод у серединне положення
delay(500);
myservo.write(0);  // встановлює сервопривод у крайнє ліве положення
delay(500);
myservo.write(180); // встановлює сервопривід у крайнє праве положення
delay(500);
}
```

4.3. Проектування інформаційних систем

Проектування інформаційних систем полягає у вибору інформаційних пристроїв (датчиків) та складання програм обробки сигналів, отриманих з датчиків для перетворення їх у відповідну форму.

Для визначення шляху переміщення та швидкості обертання двигунів постійного струму використовують одометричні датчики на основі щілинного оптичного датчика, наприклад, датчика FC-03 (рис. 4.12).

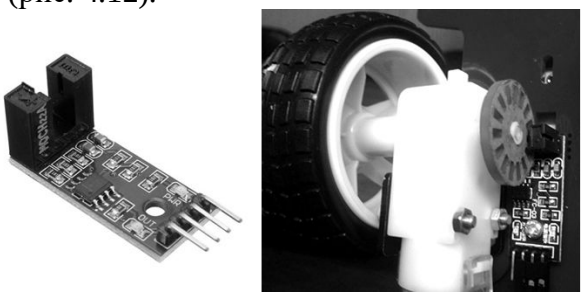


Рис. 4.12. Датчик FC-03

Принципова схема датчика FC-03 наведена на рис. 4.13.

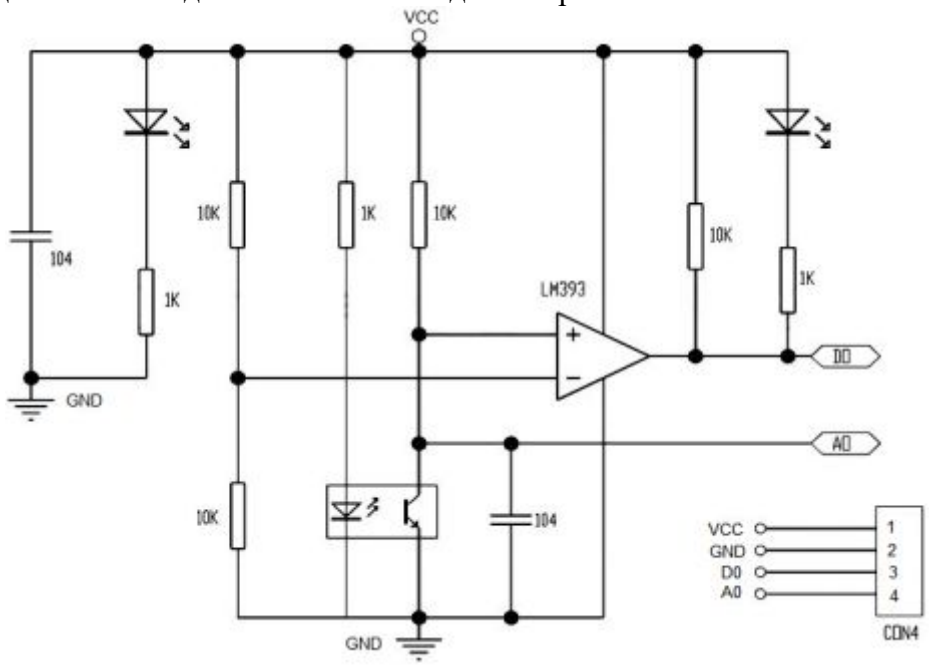


Рис. 4.13. Принципова схема датчика FC-03

Далі наведена програма переміщення по датчику обертання FC-03 на 20 імпульсів.

```
#define PIN_DO 3 //Вихід сигнала DO подається на пін 3
```

```
volatile unsigned int pulses;
#define ForP 10
#define ForM 12
#define BackP 11
#define BackM 9
boolean last = LOW;
boolean current = LOW;
 void setup() {
pinMode(PIN_DO, INPUT);
pulses = 0;
pinMode(ForM, OUTPUT);
pinMode(ForP, OUTPUT);
pinMode(BackP, OUTPUT);
pinMode(BackM, OUTPUT);
 }
void loop() {
current =  last;
current = digitalRead(PIN_DO);
if (last != current) // Якщо змінилось ...
{
delay(4); //чекаємо 4 мс
last = current;
if (current == HIGH) // Якщо позитивний ...
{
pulses++;
}
}
if (pulses < 19)
{
// вперед
digitalWrite(ForM, HIGH);
digitalWrite(ForP, HIGH);
digitalWrite(BackM, LOW);
digitalWrite(BackP, LOW);
}
else
{
// стоп
digitalWrite(ForM, LOW);
digitalWrite(ForP, LOW);
digitalWrite(BackM, LOW);
digitalWrite(BackP, LOW);
delay(100);
pulses = 0;
delay(2000);
}
}
```

Знаючи діаметр колеса d і кількість імпульсів датчика за одне обертання n_d, а також виходячи з того, що за одне обертання колеса транспортний засіб переміщується на відстань πd можна розрахувати кількість імпульсів n_L, яка відповідає переміщенню на відстань L.

$$n_L = L\, n_d / d.$$

Долі наведений приклад програми, що здійснює розрахунок за вказаною формулу. У цій програмі n_d та d задаються як константи, L встановлюється з монітору порта, після чого значення L та n_L виводяться на монітор (рис. 4.14).

```
int16_t nL, nd, L, d ; //встановлення змінних

void setup (){
 nd =20;
 d =100;
Serial.begin  (9600);
}
void loop (){
// Коли в буфері є дані
while (Serial.available() > 0)
{
L = Serial.parseInt(); // Отримане число
if (Serial.read() == '\n') // Кінець передачі
{
  nL = L * nd / d;
Serial.print("L = \t");
Serial.println(L);
Serial.print("nL = \t");
Serial.println(nL);
delay (200);
}
}
}
```

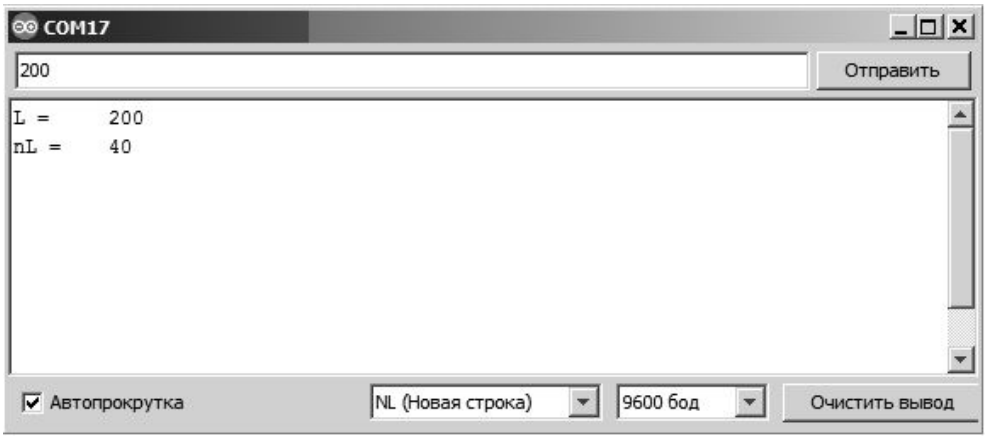


Рис. 4.14. Результати розрахунку на моніторі

Ультразвуковий модуль вимірювання відстані дозволяє визначати відстань до об'єкта від 2 см до 4,5 м з точністю до 0,3 см (рис. 4.15).

Рис. 4.15. Ультразвуковий модуль вимірювання відстані

У наведеному нижче скетчі ми будемо відсилати дистанцію в порт комп'ютера, а також при дистанції менше 30 сантиметрів включати світлодіод, підключений до 13 піну.

```
#define Trig 9
#define Echo 8
#define ledPin 13
void setup()
{
pinMode(Trig, OUTPUT); //встановлюємо як вихід
pinMode(Echo, INPUT); // встановлюємо як вхід
pinMode(ledPin, OUTPUT);

}
unsigned int impulseTime=0;
unsigned int distance_sm=0;
void loop()
{
digitalWrite(Trig, HIGH);
/* Подаємо імпульс на вхід trig датчика */
delayMicroseconds(10); // 10 мікросекунд
digitalWrite(Trig, LOW); // Відключаємо
impulseTime=pulseIn(Echo, HIGH); // Визначаємо тривалість імпульсу
distance_sm=impulseTime/58; // Перераховуємо в сантиметри
if (distance_sm<30) // Якщо відстань менше 30 см,
{
digitalWrite(ledPin, HIGH); // то світлодіод включений,
}
else
{
digitalWrite(ledPin, LOW); // інакше не включений
}
delay(100);
/* чекаємо 0.1 с. Наступний імпульс може бути відправлений тільки після зникнення луни від попереднього імпульсу. Рекомендованний період між імпульсами повинен бути не менше 50 мс. */
}
```

Для керування у ручному режимі можуть використовуватись потенціометри, що підключені до аналогових входів контролера, які видають значення від 0 до 1023. При використанні їх для керування сервоприводами у ручному режимі треба здійснити зміну масштабу, щоб отримати значення від 0 до 180.

Для зчитування у змінну значення сигналу з аналогового входу pin використовується функція analogRead().

Функція analogRead() має такий синтаксис: value = analogRead(pin); де value – змінна.

Для зміни масштабу можна використати функцію пропорційного перетворення значень від одного діапазону до другого map () та функція обмеження діапазону constrain ().

Функція map () має такий синтаксис:

valNew = map(valOld, fromLow, fromHigh, toLow, toHigh);

Змінна valOld з нижньою fromLow і верхньою fromHigh межами діапазону перетворюється у змінну valNew з нижньою toLow і верхньою toHigh межами діапазону.

Для обмеження значень змінної використовується функція constrain ().

Функція constrain () має такий синтаксис:

valNew = constrain (valOld, min, max);

min та max відповідно мінімальне та максимальне значення змінної valNew.

Далі наведений фрагмент програми ручного керування сервоприводом для керування якого треба подавати значення від 0 до 180.

```
val0 = analogRead(POT0);          //зчитування сигналу з аналогового входу
val0 = map(val0, 0, 1023, 0, 180); //перетворення масштабу
val0 = constrain(val0, 0, 180);    //обмеження діапазону
myservo0.write(val0);              //запис сигналу керування до сервоприводу
```

Контрольні питання

1. Визначити,з яких компонентів складається апаратно-програмний комплекс Arduino?
2. Назвати, чим відрізняються різні контролери Arduino?
3. Описати,як здійснити переключення режимів роботи за допомогою оператора ELSE IF?
4. Визначити, які засоби використовують для керування двигунами постійного струму?
5. Назвати, які засоби використовують для керування кроковими двигунами?
6. Визначити, які засоби використовують для керування сервоприводами?
7. Описати,які датчики використовують для визначення шляху переміщення та швидкості обертання двигунів постійного струму?
8. Визначити, як здійснити вимірювання відстані за допомогою ультразвукового датчика?
9. Назвати, яка функція визначає рівень сигналу на аналоговому вході?
10. Описати,які функції здійснюють масштабування та обмеження даних?

Лекція 5. Засоби проектування пристроїв керування роботів на основі програмованих логічних контролерів

5.1. Проектування систем керування роботів на основі ПЛК

Спеціалізовані промислові роботи можуть бути частиною складного технологічного обладнання. У такому випадку керування роботом може здійснювати система керування технологічним обладнанням.

Спеціалізовані промислові роботи виконують однорідні технологічні операції у визначеному параметричному діапазоні, наприклад, обслуговування штампувального обладнання або токарного верстата.

При цьому рівень складності завдань, що вирішують системи керування, може змінюватися в дуже великому діапазоні - від найпростіших систем керування, застосовуваних для автоматизації водопостачання, у харчовій і легкій промисловості (маніпулятори, пакувальні машини, пресове устаткування і т.д.), до складних систем керування гнучкими виробничими дільницями і цехами.

Прикладом реалізації таких систем є системи керування на основі програмовані логічні контролери (ПЛК) [4, 5].

Розглянемо таку систему керування та засоби її проектування на прикладі систем автоматизації SIMATIC S7, що випускається фірмою SIEMENS (рис. 5.1).

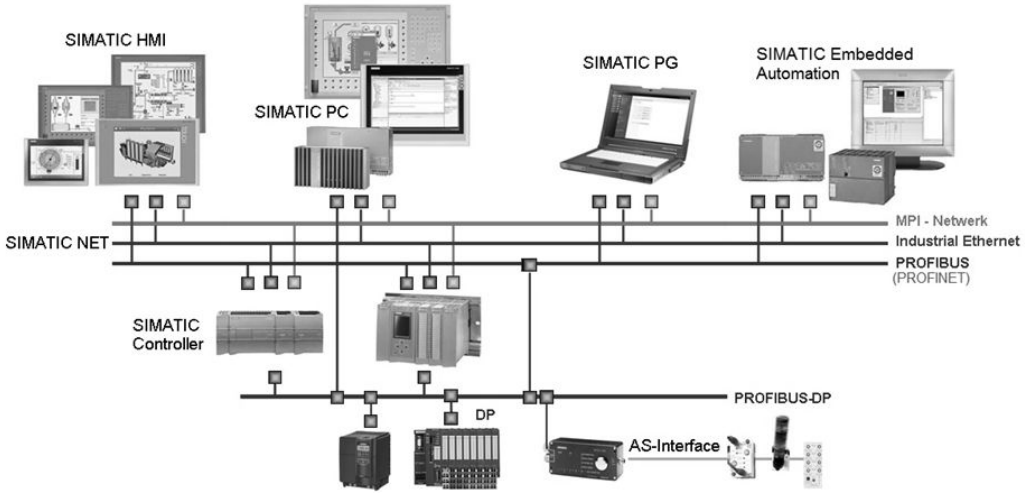


Рис. 5.1. Система керування та засоби її проектування на прикладі систем автоматизації SIMATIC S7

Ця система включає широкій набір засобів для систем автоматизації, побудованих на модульному принципі, і програмне забезпечення, що об'єднує ці засоби шляхом визначення конфігурації системи автоматизації, установки параметрів і програмування на основі програмуючих пристроїв, сумісних із персональними комп'ютерами.

Структура апаратних та програмних компонент цих систем визначається міжнародним стандартом IEC 1131.

Системи автоматизації SIMATIC S7 мають системи керування різного рівня складності.

Для простих завдань автоматизації використовуються прості системи SIMATIC S7-1200.

Для завдань автоматизації середнього рівня використовуються системи SIMATIC S7-300, до яких можна підключити близько 1000 вхідних та вихідних сигналів.

Для завдань високого рівня автоматизації використовуються системи SIMATIC S7-400, які мають спроможність вирішувати дуже складні завдання керування, та включають системи підвищеної надійності SIMATIC S7-400H, та підвищеної безпеки SIMATIC S7-400F.

Системи автоматизації SIMATIC S7-200, SIMATIC S7-1200, SIMATIC S7-300C мають процесори з вмонтованими входами та виходами (компактні контролери).

Остання розробка SIMATIC S7-1500 має значно більшу швидкість, що дозволяє вирішувати задачі керування швидкодіючого обладнання.

Модульний принцип побудови ПЛК дає можливість створити систему керування шляхом вибору таких модулів, які є оптимальними для поставленої задачі.

Існують такі типи модулів (рис. 5.2):

- блоки живлення (PS), які здійснюють функції живлення усієї системи перетворюючи напругу з мережі у потрібні постійні напруги;
- процесорні модулі (CPU);
- сигнальні модулі (SM), що забезпечують підключення цифрових і аналогових вхідних та вихідних сигналів;
- функціональні (інтелектуальні) модулі (FM), що виконують визначені функції незалежно від процесорного модуля, наприклад модулі позиціонування і регулювання;
- комунікаційні процесори (CP), що здійснюють зв'язок системи керування з іншими пристроями, у тому числі з операторськими станціями і промисловими мережами;
- інтерфейсні модулі (IM), що забезпечують зв'язок між окремими рядами у випадку багаторядної побудови системи автоматизації.

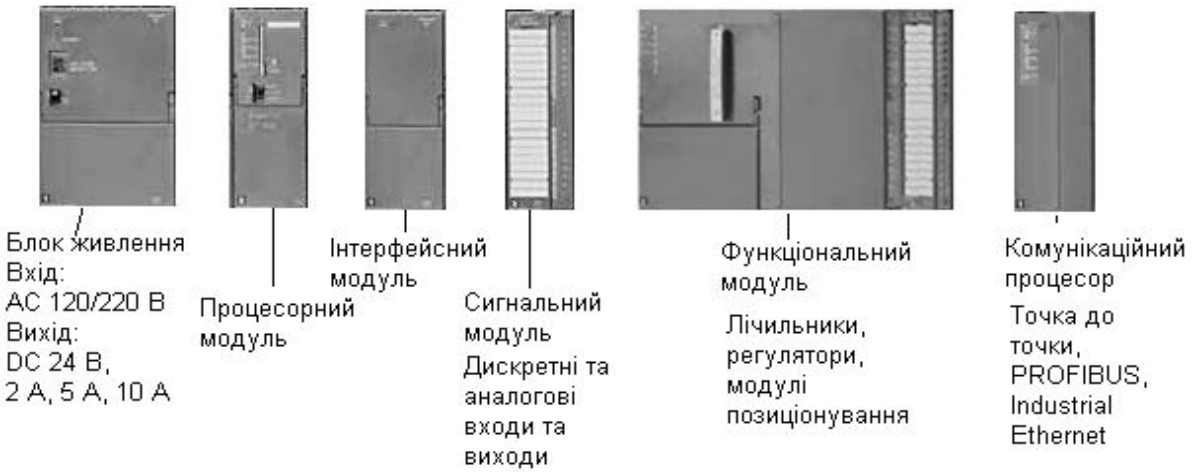


Рис. 5.2. Типи модулів ПЛК

Кожен модуль має декілька варіантів, що дозволяє вибрати оптимальну за можливостями та вартістю конфігурацію системи керування.

Процесорні модулі представляють собою обчислювальний пристрій з процесором, пам'яттю, та пристроями введення-виведення. Сімейство пристроїв керування як правило має набір процесорних модулів з різними можливостями програмного забезпечення, швидкодії, розміру пам'яті та можливостями підключення до мереж зв'язку. Це дозволяє вибрати оптимальне рішення для реалізації заданого алгоритму керування.

Структура процесорних модулів відрізняється від структури персональних комп'ютерів. Вони потребують значно менші обсяги пам'яті, тому програма зберігається не у зовнішніх носіях, а у постійній пам'яті (як правило це пам'ять з електричним перезаписом – FLASH пам'ять).

Та частина програми, яка виконується та дані зберігаються у оперативній пам'яті. Ця пам'ять теж значно менша, ніж у персональних комп'ютерів.

Пристрої введення-виведення призначені для підключення додаткових модулів за допомогою системної магістралі, та зовнішніх пристроїв за допомогою локальних мереж, наприклад, програмуючого пристрою, або іншого контролера.

Компактні процесорні модулі мають вмонтовані входи та виходи для підключення дискретних та аналогових сигналів, а також можуть виконувати додаткові технологічні функції, наприклад, обслуговування сигналів переривання, функції швидкодіючих лічильників, вимірювання частоти, функції позиціонування та регулювання тощо.

Модулі живлення перетворюють напругу мережі на напруги, які потрібні для живлення інших модулів системи автоматизації та відрізняються за максимальним струмом споживання.

Сигнальні модулі поділяються на дискретні та аналогові модулі вхідних та вихідних сигналів.

Дискретні модулі вхідних сигналів використовують сигнали постійного струму 24В та змінного струму 120/220В. Дискретні модулі вихідних сигналів дозволяють використовувати такі ж самі сигнали, але з різним струмом навантаження (0,5, 1, 2, 4А). Крім того вони відрізняються кількістю входів (8, 16, 32, 64) та виходів (4, 8, 16, 32).

Аналогові модулі мають різні вхідні та вихідні сигнали: напругу, струм, опір. Модулі дискретних та аналогових входів та виходів можуть забезпечити гальванічну розв'язку по входах та виходах.

Функціональні модулі відрізняються як функціями (лік, позиціонування і регулювання) так і кількістю каналів.

Лічильні модулі призначені для підрахунку імпульсів фотоімпульсних датчиків і вимірюють позицію або швидкість переміщення і можуть використовуватись для простих систем позиціонування роботів.

Модулі позиціонування призначені для створення систем позиціонування з різними виконавчими пристроями та датчиками зворотного зв'язку для позиціонування з однією або багатьма (до 4) осями. Такі модулі можуть виконувати функції позиційного та контурного керування роботом з використанням різних методів лінійної та колової інтерполяції.

Модулі регулювання призначені для одно або багатоканального регулювання. Для цього можуть використовуватись ПІД-, ФАЗІ- та нейрорегулятори.

Комунікаційні процесори здійснюють зв'язок системи керування з іншими пристроями, у тому числі з операторськими станціями і промисловими мережами. При цьому використовується типи зв'язку точка до точки або локальні обчислювальні мережі PROFIBUS, PROFINET, MODBUS, Ethernet.

У разі потреби може використовуватися побудова системи автоматизації з двома, трьома або чотирма рядами. При цьому використовуються спеціальні інтерфейсні модулі підключення ІМ.

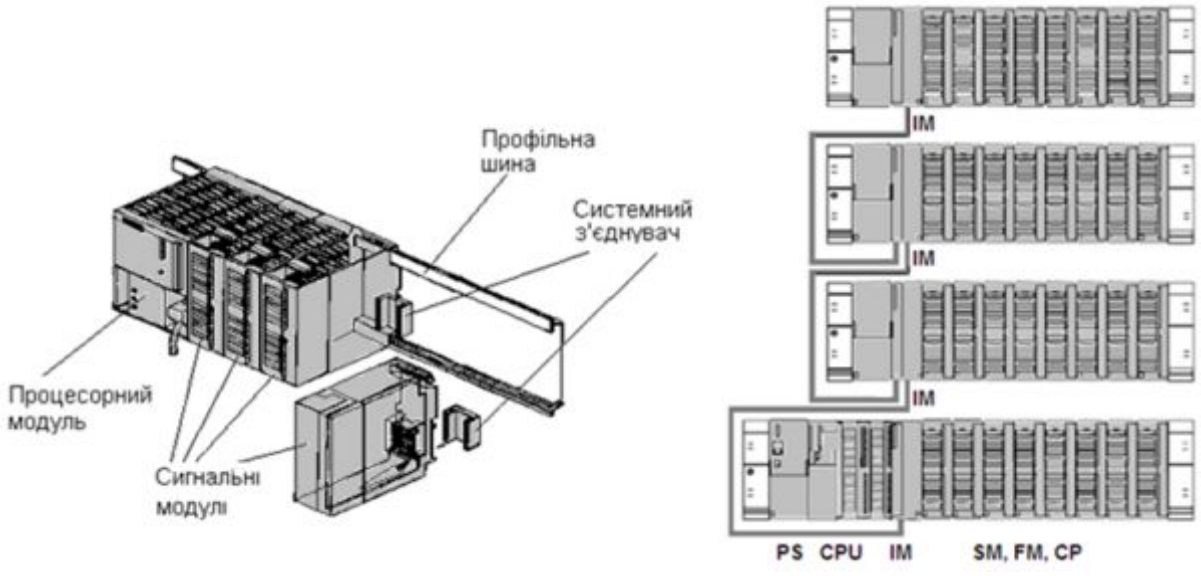


Рис. 5.2. Побудова системи автоматизації з чотирма рядами

Для задач з підвищеним рівнем безпеки використовують ПЛК з резервуванням процесорних та іншим модулів.

На рис. 5.3 наведена система з резервуванням на основі ПЛК S7-400H.

Ця система дозволяє здійснити резервування також систем верхнього рівня керування за допомогою мережі Ethernet.

Головний та допоміжний процесори підключаються до сигнальних модулів на основі децентралізованої периферії ET200M за допомогою резервованої мережі PROFIBUS.

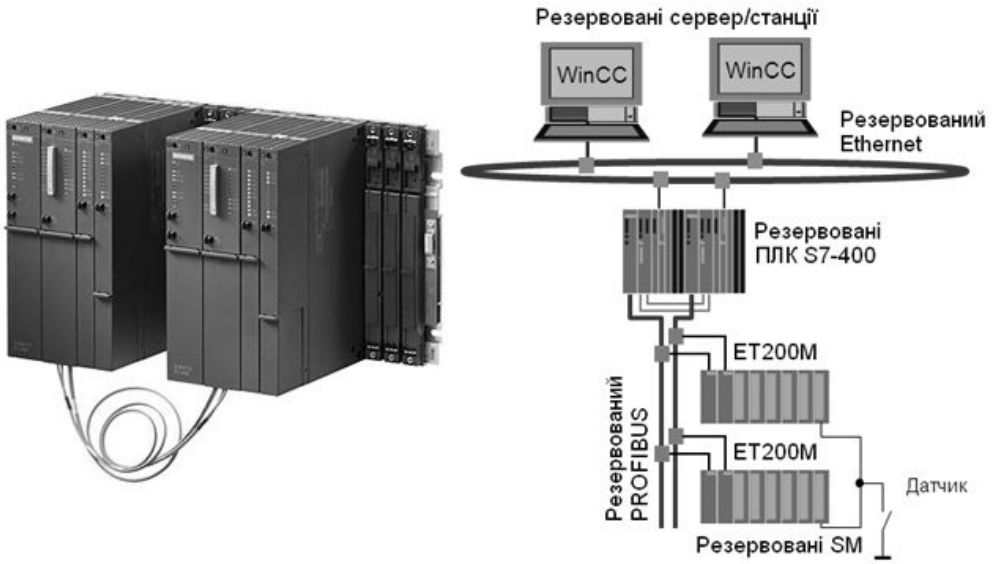


Рис. 5.3. Система з резервуванням на основі ПЛК S7-400H

Системи програмування сучасних програмованих контролерів, наприклад, STEP7 фірми SIEMENS, включає засоби проектування апаратних компонент системи керування, програмування, та пошуку помилок під час роботи системи.

Складання проекту системи керування робиться за допомогою засобів, які є складовою частиною язика програмування.

Сучасні системи керування будуються за модульним принципом, тому під час проектування вони використовують каталоги з набором елементів системи, або з набором команд для відповідної мови програмування.

Структура проекту складається з папки проекту (назва проекту визначається при її складанні).

Проект складається з однієї або кількох станцій (якщо вони складають мережу).

Для станції треба визначити її конфігурацію (склад), після чого створюється папка програм, де знаходяться програмні блоки та символьна таблиця.

Для створення конфігурації, програм та інших складових частин проекту використовуються різні засоби, які викликаються з керуючої програми **"SIMATIC-Manager“** (рис. 5.4).

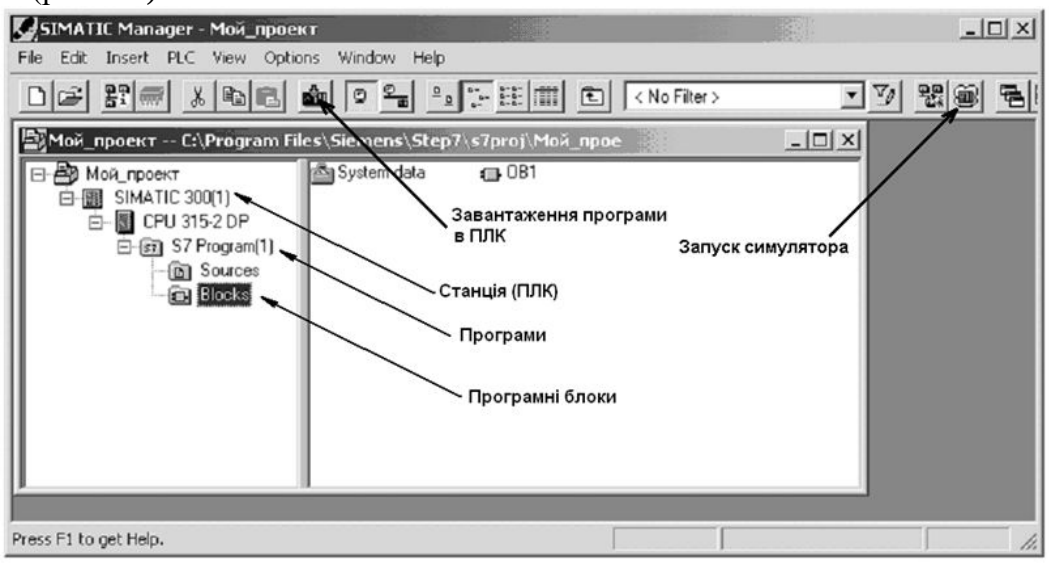


Рис. 5.4. SIMATIC-Manager

Визначення структури і складу системи керування робиться за допомогою конфігуратора **HW Config**, який є складовою частиною язика програмування (рис. 5.5).

Для проектування апаратних компонент треба перейти до інструменту конфігурації (відкрити об’єкт "Станція"), після чого відкривається вікно конфігурації.

Це вікно включає каталог з набором елементів системи та засоби визначення параметрів цих елементів.

За їх допомогою здійснюються вибір конфігурації.

Проектування здійснюється шляхом вибору відповідних модулів з каталогу.

На початку треба вибрати носії модулів (RACK), а потім самі модулі.

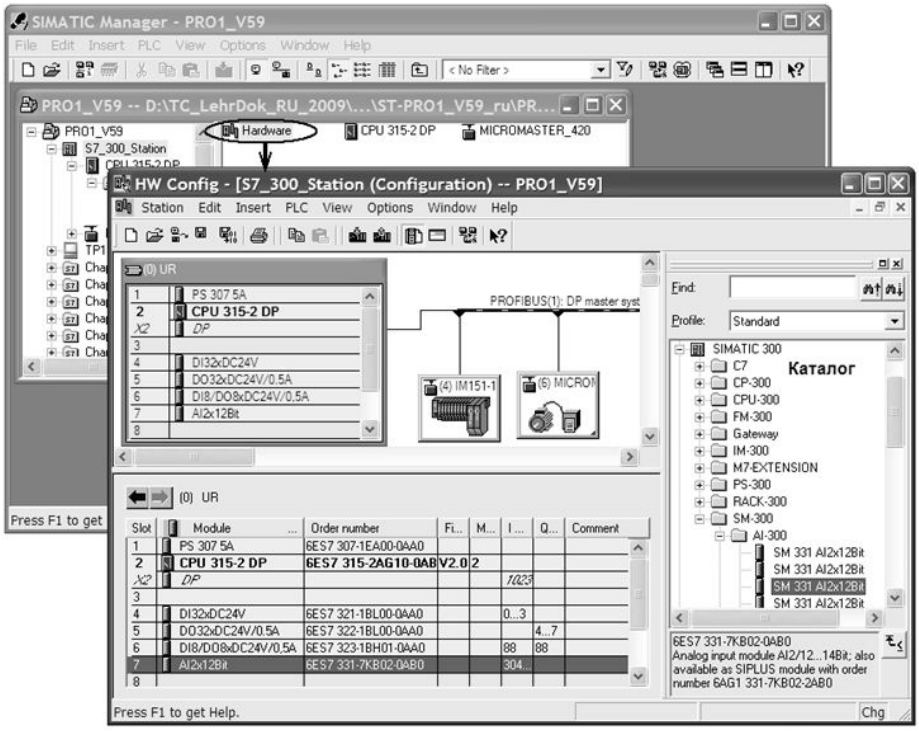


Рис. 5.5. Конфігуратор HW Config

Програми підрозділяються на **системні і прикладні**.

Системні програми являють собою програми для реалізації внутрішніх робочих функцій пристрою керування. Однією з таких програм є операційна система контролера.

Прикладні програми являють собою програму для опрацювання сигналів і надання впливу на керований процес відповідно до задачі керування, яка складається за допомогою мов програмування..

Процесор контролера виконує команди послідовно друг за другом.

У мові програмування STEP7, згідно з стандартом IEC 61131-3, розрізняють блоки, що містять команди для обробки сигналів (організаційні OB, функції FC і функціональні блоки FB), а також блоки, у яких зберігаються дані (BD).

Організаційні блоки (**OB**) служать для керування прикладною програмою у формі переліку блоків програми. Є організаційні блоки для циклічної обробки програм, для обробки з керуванням по перериваннях і для обробки з керуванням за часом.

Організаційний блок є обов'язковим елементом програми, наприклад, **OB1** для циклічного виконання програми

Функціональні блоки (**FB**) і **функції** (**FC**) реалізують часто повторювані або дуже складні функції. Функціональні блоки і функції можуть бути стандартними (**SFB, SFC**) або програмуються самим користувачем.

Функціональні блоки і функції можуть мати параметри, які встановлюються шляхом завдання різноманітних значень параметрів блока за допомогою змінних.

У **блоках даних** (**DB**) знаходяться дані, використовувані в програмі користувача.

Для кращої наочності блоки програми користувача (**OB, FB, FC**) можуть розділятися на окремі фрагменти - схеми.

У програмі використовуються такі змінні:

- входи **I**, наприклад , I 10.1, IB 20, IW 40, ID 100;

- виходи **Q**, наприклад , Q 0.0, QB 12, QW 24, QD 30;
- пам'ять **M** (меркери), наприклад, M 200.0, MB 220, MW 300, MD 400.

До змінних можна звертатися як до окремих бітів (I 10.1), так і байтів (MB 220), слів (QW 24), та подвійних слів (ID 100).

На рис. 5.6 наведена структура та порядок виконання програми.

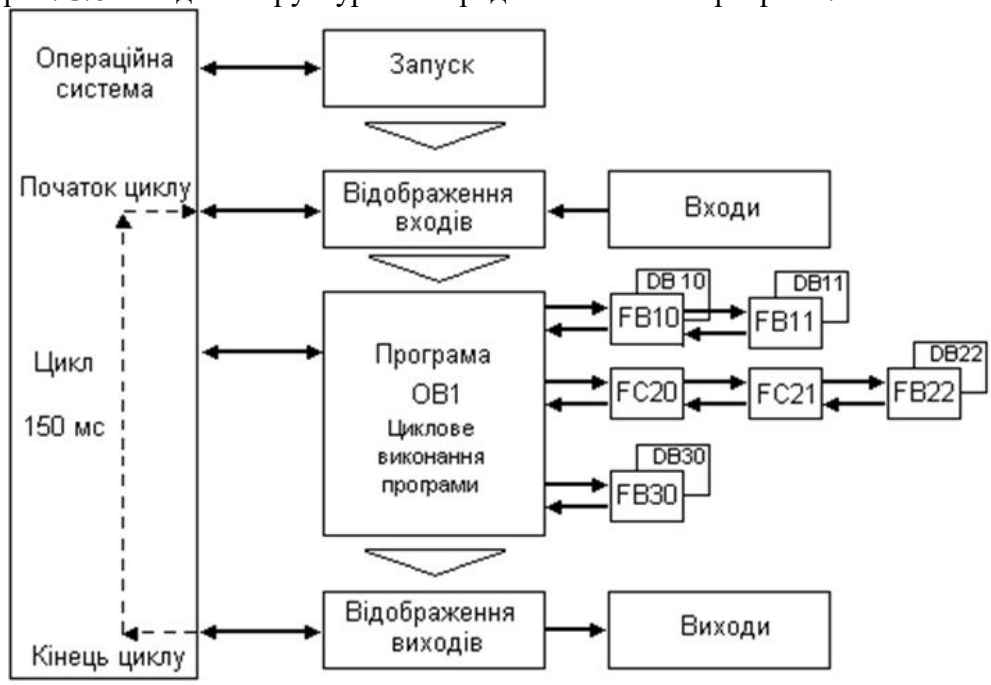


Рис. 5.6. Структура та порядок виконання програми

Для створення програмних блоків OB, FB, FC використовують програмні редактори відповідно з формою представлення програми (рис. 5.7).

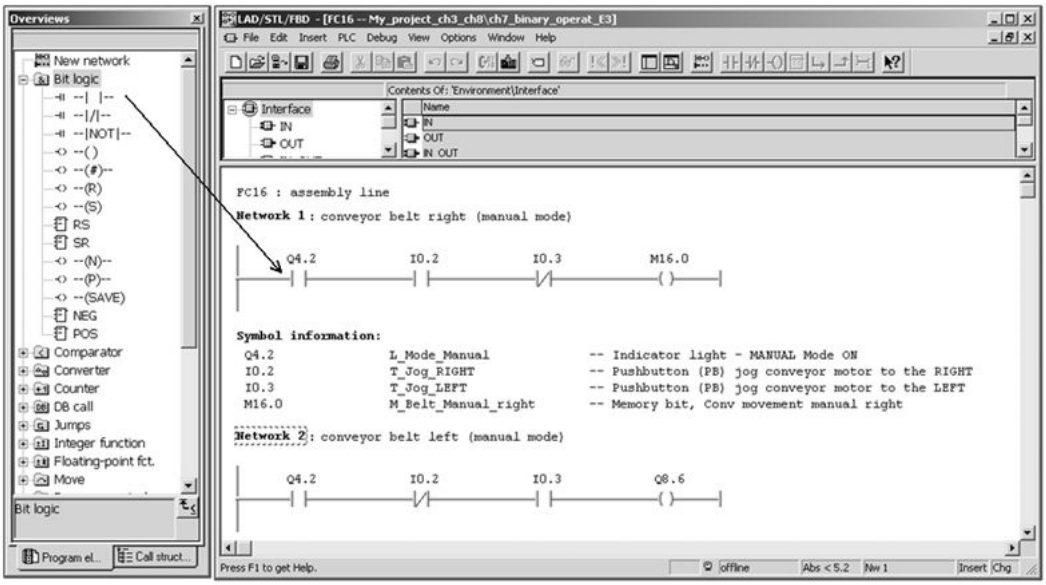


Рис. 5.7. Програмний редактор

Редактори для представлення програм у вигляді контактного та функціонального плану дозволяють здійснювати вибір окремих команд за допомогою каталогу.

5.2. Проектування виконавчих пристроїв роботів на основі ПЛК

Для керування двигунами змінного струму використовують частотні перетворювачі можуть мати різну конструкцію, наприклад, у вигляді моноблоків, або складатися з окремих модулів. На рис. 5.8 наведений частотний перетворювач фірми Сіменс SINAMICS G120, який має модульну структуру та складається з силового модуля, пристрою керування та операторської панелі.

Рис. 5.8. Частотний перетворювач фірми Сіменс SINAMICS G120

Програмування частотного перетворювача здійснюється за допомогою встановлення великої кількості параметрів (декілька сотень).

Ці параметри задають джерело встановлення частоти на виході перетворювача та джерело команд, закон керування (керування за вольт-частотною характеристикою або векторне керування), обмеження на частоту, струм, потужність, швидкість розгону та гальмування тощо.

Параметри встановлюються за допомогою операторської панелі.

У самому простому випадку за допомогою параметрів встановлюють тривалість розгону (P1120), тривалість гальмування (P1121) (керування за рампою).

При находженні сигналу запуску частота на виході частотного перетворювача змінюється від нуля до встановленого значення з швидкістю, яка визначається тривалістю розгону.

При находженні сигналу зупинки частота на виході частотного перетворювача змінюється від встановленого значення до нуля з швидкістю, яка визначається тривалістю гальмування.

Для налагодження та керування виконавчими пристроями існують програмні інструменти комп'ютерного налагодження та керування, наприклад програма STARTER для електроприводів фірми SIEMENS.

Рис. 5.9. Програмний інструмент STARTER для налагодження електроприводів фірми SIEMENS

Цей інструмент дозволяє створити проект без підключення до привода в режимі Offline (рис. 5.10), встановити конфігурацію привода, встановити підключення до частотного перетворювача (режим online), представити параметри у вигляді масок та переліків, здійснити керування за допомогою цифрових входів/виходів, здійснити керування приводом у ручному режимі.

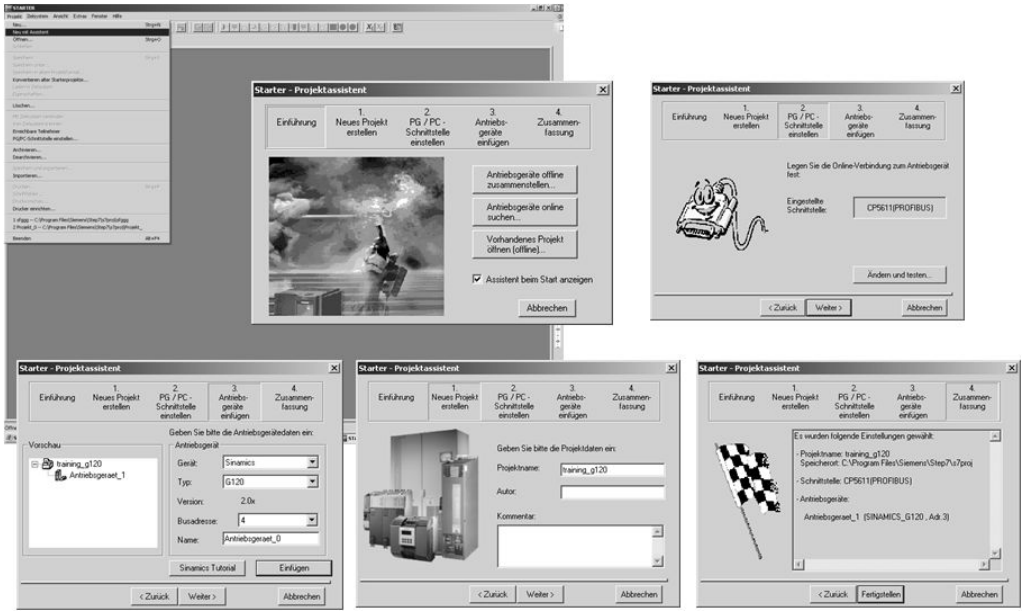


Рис. 5.10. Створення проекту в режимі Offline

5.3. Проектування інформаційних систем роботів на основі ПЛК

Проектування інформаційних систем полягає у вибору інформаційних пристроїв (датчиків) та складання програм обробки сигналів, отриманих з датчиків для перетворення їх у відповідну форму.

Для визначення шляху переміщення та швидкості обертання двигунів найчастіше використовують фотоімпульсні або абсолютні датчики, які можуть бути умонтованими у двигун і його опитує система керування двигуна для визначення швидкості обертання, або встановлюються на рухомі компоненти для визначення шляху переміщення робочого органа або окремих ланок системою керування роботом.

Для визначення положення використовуються абсолютні датчики положення, які видають код кута обертання і не потребують визначення початкового значення (абсолютний датчик з PROFIBUS DP 6FX2001-5FP12).

Проектування датчиків, сумісних з ПЛК SIMATIC здійснюється за допомогою конфігуратора **HW Config**.

Контролер розглядає ці датчики як входи з визначеною адресою (у нашому випадку PIW 256), які мають значення від 0 до 8192 (13 бітів) для кута від 0 до 360 градусів (рис. 5.11).

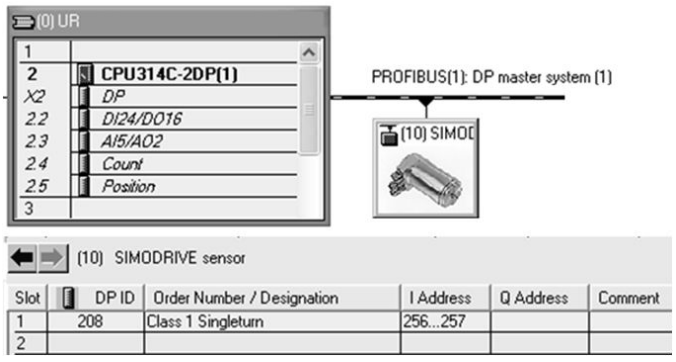


Рис. 5.11. Проектування датчика як складної частини ПЛК

При використанні аналогових входів для отриманих значень треба здійснити функцію масштабування, щоб ці дані відповідали вимірюваним параметрам. Винятком є лише вимірювання температури, оскільки отримаємо дані з точністю одна десята градуса (рис. 5.12).

Диапазон	Напряжение например:		Ток например:		Сопротивление например:		Температура: например Pt100	
	Диапазон ± 10V	Ед.	Диапазое 4 .. 20mA	Ед.	Диапазон 0...300Ohm	Ед.	Диапазон -200...+850ºC	Ед.
Переполнение	>= 11,759	32767	>= 22,815	32767	>=352,778	32767	>= 1000,1	32767
Перегрузка	11,7589 : 10,0004	32511 : 27649	22,810 : 20,0005	32511 : 27649	352,767 : 300,011	32511 : 27649	1000,0 : 850,1	10000 : 8501
Номинальный диапазон	10,00 7,50 : -7,5 -10,00	27648 20736 : -20736 -27648	20,000 16,000 : : 4,000	27648 20736 : : 0	300,000 225,000 : : 0,000	27648 20736 : : 0	850,0 : : : -200,0	8500 : : : -2000
Отрицат. перегрузка	- 10,0004 : - 11,759	- 27649 : - 32512	3,9995 : 1,1852	- 1 : - 4864	Отрицат. значение невоз-можно	- 1 : - 4864	- 200,1 : - 243,0	- 2001 : - 2430
Отрицат. переполнение	<= - 11,76	- 32768	<= 1,1845	- 32768		- 32768	<= - 243,1	- 32768

Рис. 5.12. Представлення даних на виході модулів аналогових входів

Для масштабування даних, отриманих з модулів аналогових входів використовується системна функція FC105 (рис. 5.13).

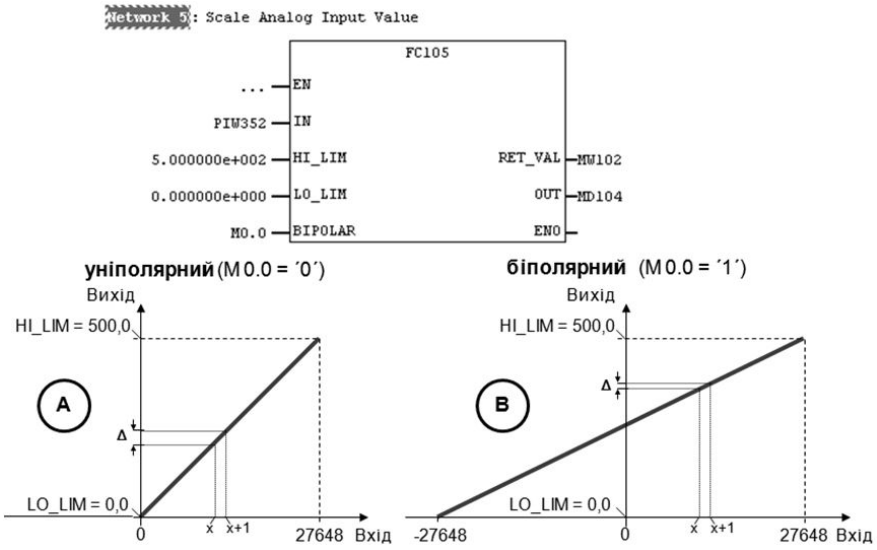


Рис. 5.13. Системна функція FC105 для масштабування даних

Контрольні питання

1. Визначити, з яких компонентів складається система автоматизації SIMATIC S7?
2. Назвати, який стандарт визначає структуру апаратних та програмних компонент ПЛК?
3. Описати, які типи модулів є у складі ПЛК?
4. Розповісти, які функції виконують сигнальні модулі?
5. Назвати, які функції виконують функціональні модулі?
6. Описати, які засоби використовують проектування апаратних компонент ПЛК?
7. Визначити,які засоби використовують проектування програмних компонент ПЛК?
8. Назвати, які інструменти використовують для проектування приводів фірми SIEMENS?
9. Розповісти, як здійснюється проектування датчиків, сумісних з ПЛК SIMATIC?
10. Описати, яка функція здійснює масштабування даних?

Лекція 6. Приклади робототехнічних систем на основі програмованих логічних контролерів

6.1. Засоби проектування роботів на основі ПЛК

Промислове програмне забезпечення - це система тісно пов'язаних інструментальних засобів для програмування систем автоматизації. Воно підтримує усі фази виконання проекту автоматизації:

- Планування, вибір конфігурації і завдання параметрів настроювання апаратури і мереж
- Створення програми користувача
- Документування
- Тестування, запуск і обслуговування
- Керування процесом

Проектування програмних компонент здійснюється за допомогою **інструментальних засобів проектування** до яких можна віднести:

- Мови програмування високого рівня
- Графічні мови для технологічного програмування
- Допоміжне програмне забезпечення для діагностики, імітації, дистанційного керування, розробки документації і т.д.

Програми являють собою сукупність команд і умов для опрацювання сигналів і надання впливу на керований процес відповідно до задачі керування.

Процесор контролера виконує команди послідовно друг за другом згідно за циклічною структурою програми. Після виконання останньої команди процесор починає виконувати першу команду. Виконання команд періодично повторюється, тому процесор реалізує циклічне виконання програми [5].

Час, необхідний для однократного виконання всіх команд, називається часом циклу. Прикладні програми доступні користувачу і підрозділяються на блоки. Типи блоків наведені на рис. 6.1.

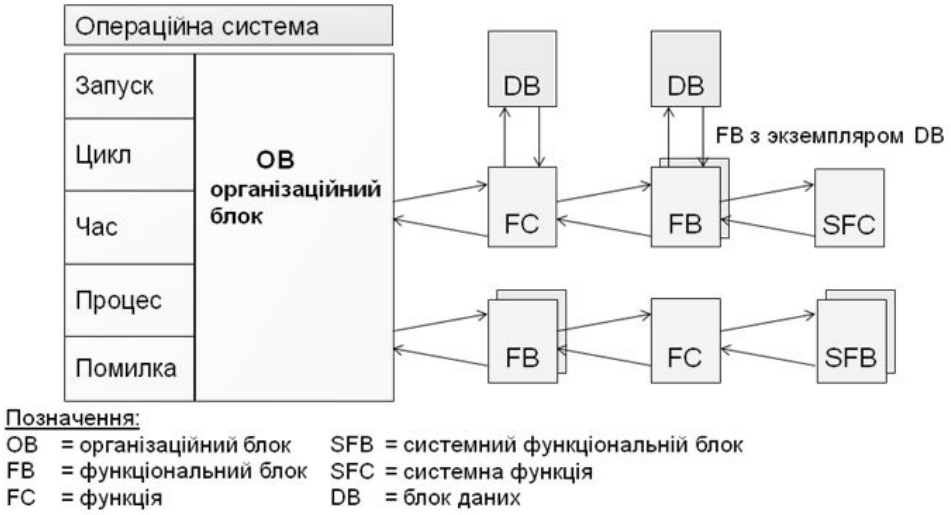


Рис. 6.1. Типи блоків

Розрізняють програмні блоки, що містять команди для обробки сигналів (організаційні блоки, функції і функціональні блоки), а також блоки даних, в яких зберігаються дані.

Організаційні блоки (OB) визначають режим роботи контролера, до яких належать циклове виконання програми, переривання програми за зовнішніми сигналами, переривання програми за часом та переривання для обробки помилок (рис. 6.2).

Функції (FC) та функціональні блоки (FB) вміщують окремі програмні модулі, у яких можна задавати параметри.

Функціональні блоки (FB) мають можливість запам'ятовувати дані у додаткових блоках даних (екземплярах DB) для подальшої обробки.

Блоки даних (DB) призначені для зберігання упорядкованих елементарних та складних типів даних.

Елементарний тип даних представляє собою простий набір даних.

Складний тип даних зберігає дані у вигляді структур або масивів.

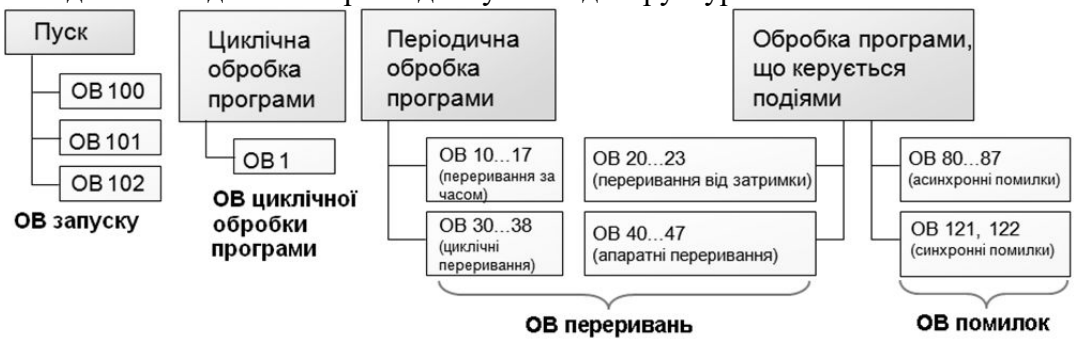


Рис. 6.2. Типи організаційних блоків

Налагодження організаційних блоків здійснюється за допомогою конфігуратора **HW Config**.На рис 6.3 наведений порядок встановлення організаційних блоків переривання від реального часу та циклічного переривання. На рис. 6.4 наведений приклад апаратного переривання при використанні модуля аналогових входів.

а) б)

Рис 6.3. Порядок встановлення організаційних блоків переривання від реального часу (а) та циклічного переривання (б)

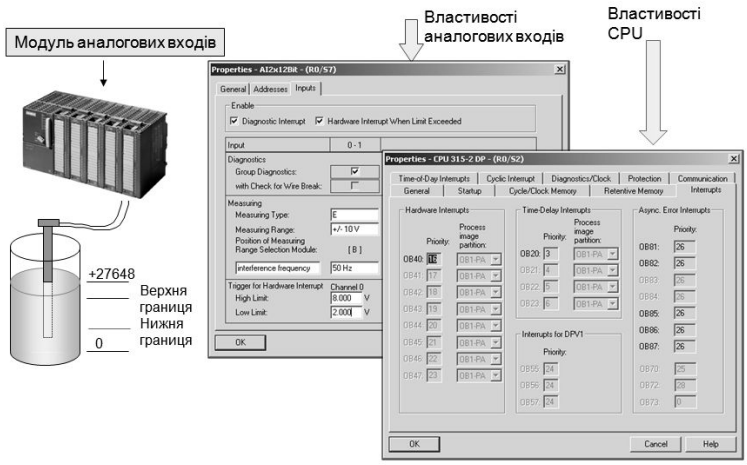


Рис. 6.4. Апаратне переривання при використанні модуля аналогових входів

Організаційні блоки помилок дають можливість виявити помилки при виконанні програми.

Розподіляють організаційні блоки асинхронних та синхронних помилок.

Асинхронні помилки не залежать від виконання програми та можуть з'явитися у будь який час, наприклад, діагностичні помилки (апаратні помилки, що виявляються системою діагностики).

Синхронні помилки залежать від програми та поділяються на програмні (помилки у програмі) та апаратні (помилки при звертанні до апаратних компонент, наприклад, звертання до неіснуючих модулів.

Організаційні блоки помилок повинні мати програму, яка відповідним чином реагує на наявність помилок, наприклад, видає інформацію на індикатори або переводить систему у безпечний режим.

6.2. Використання функцій, функціональних блоків та даних

При створенні функцій замість фактичних параметрів (змінних, які можуть бути адресами входів та виходів, до яких підключені окремі елементи, або адресами зберігання числових значеннями) можна використовувати формальні параметри, які представляють собою умовне позначення окремих змінних.

Ці позначення встановлюються в області завдання параметрів програмного редактора.

Якщо при складанні програми для функції використовувати тільки формальні параметри, то таку функцію можна використовувати для різних аналогічних об'єктів.

Наприклад, якщо ми маємо декілька однакових приладів, то можемо створити одну функцію керування з формальними параметрами, а потім викликати цю функцію для кожного приладу, підставляючи фактичні параметри відповідного приладу.

Далі наведений приклад програми, що здійснює ручне керування двома підйомниками, у яких керування двигуном здійснюють відповідно контактори КМ11, КМ12, КМ13 та КМ21, КМ22, КМ23 (рис. 6.5).

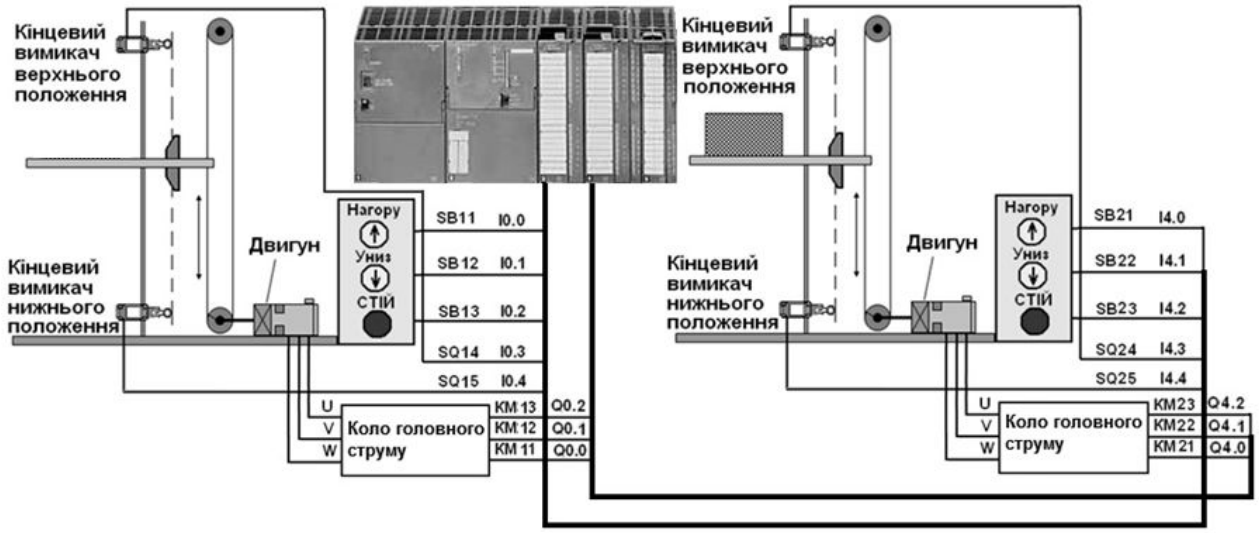


Рис. 6.5. Ручне керування двома підйомниками

Для створення функції керування підйомником визначимо органи керування та виконавчі пристрої.

Коло головного струму (рис. 6.6) керує двигуном, де контактори виконують такі дії:

КМ1 (K_MOTOR)

включає двигун,

КМ2 (K_UP) забезпечує рух догори,

КМ3 (K_DOWN) забезпечує рух униз.

Кнопка SB1 (SA_UP) здійснює рух догори.

Кнопка SB2 (SA_DOWN) здійснює рух униз.

Кнопка SB3 (SA_STOP) здійснює зупинку двигуна.

Кінцевий вимикач верхнього положення SQ4 (SQ_UP).

Кінцевий вимикач нижнього положення SQ5 (SQ_DOWN).

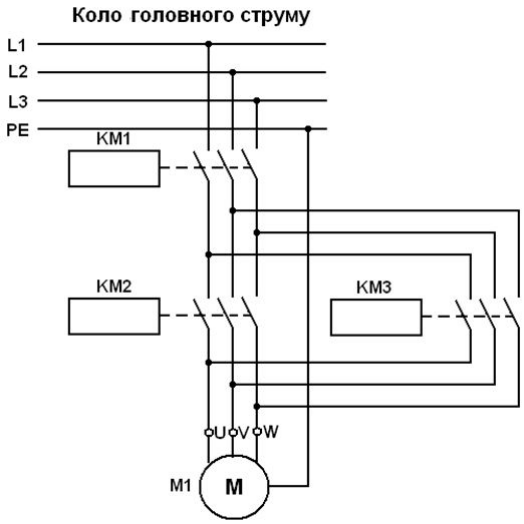


Рис. 6.6. Коло головного струму

Спочатку визначимо параметри функції (рис. 6.7).

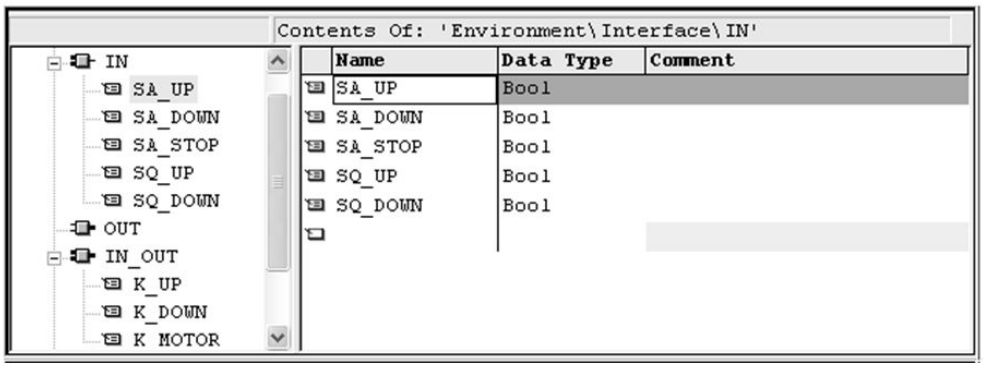


Рис. 6.7. Визначення параметрів функції

Функція керування підйомником має вигляд, наведений на рис. 6.8.

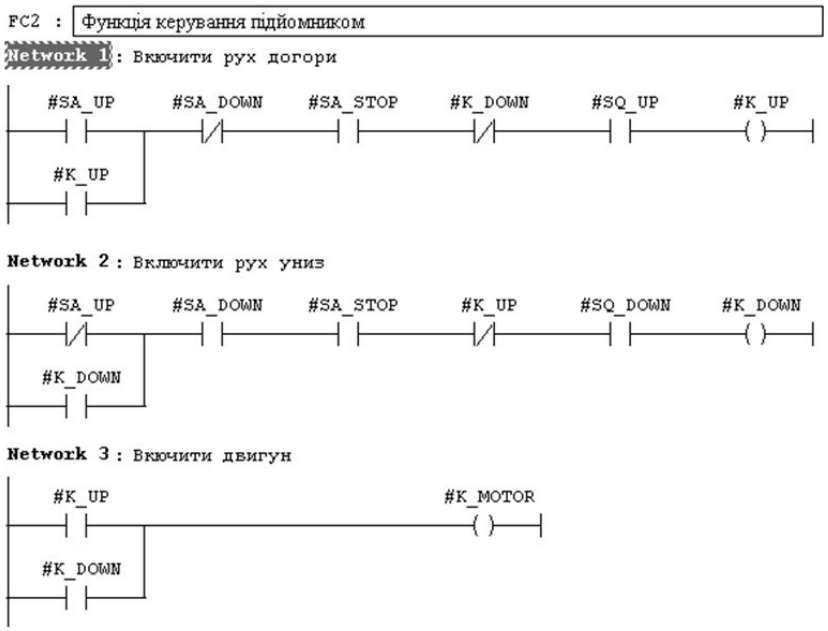


Рис. 6.8. Функція керування підйомником

На рис. 6.8 наведений виклик функції для керування двома підйомниками.

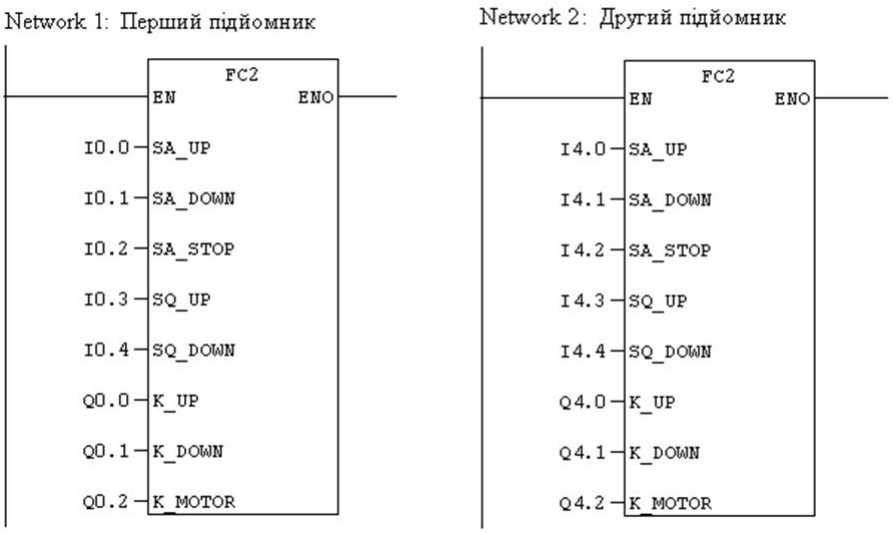


Рис. 6.8. Виклик функції для керування двома підйомниками

Функціональні блоки (FB) на відміну від функцій (FC) мають пам'ять.

Це означає, що функціональному блоку назначається локальний блок даних, так званий **екземплярний блок даних**.

При виклику FB треба також задавати номер екземплярного DB, котрий при цьому викликається автоматично.

Екземплярний DB використовується для запису **статичних змінних**.

Ці локальні змінні можна використовувати тільки в тому FB, у визначальній частині якого вони описані.

Локальні змінні зберігаються і після закриття блока.

При виклику функціонального блока значення фактичних параметрів зберігаються в екземплярному блоку даних.

На рис. 6.9 наведений виклик функціонального блоку з екземплярним блоком даних.

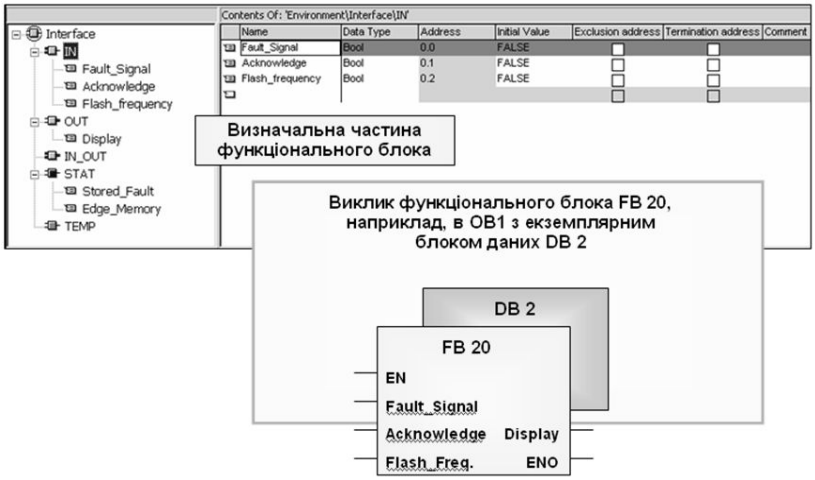


Рис. 6.9. Виклик функціонального блоку з екземплярним блоком даних

Екземплярний блок даних зберігає усі параметри та статичні дані функціонального блоку (рис. 6.10).

Для зберігання упорядкованих елементарних та складних типів даних використовують глобальні блоки даних, до яких є доступ з усіх блоків (рис. 6.11). У складі блоків даних можна використовувати однорідні (рис.6.12) та неоднорідні (рис. 6.13) структури даних.

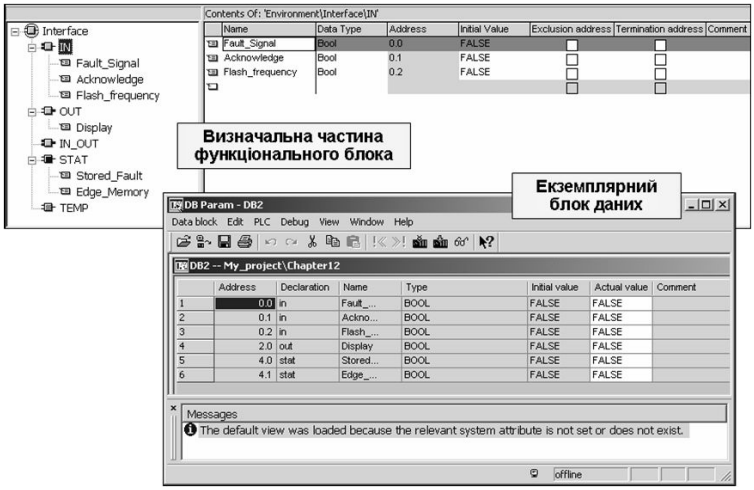


Рис. 6.10. Екземплярний блок даних

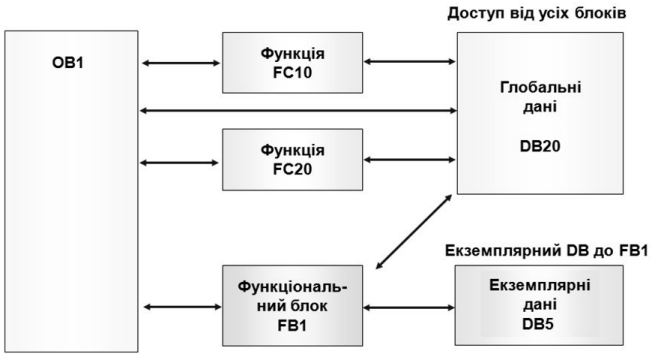


Рис. 6.11. Екземплярні та глобальні блоки даних

Точка вимірювання

1. Точка вимірювання, тип даних Real
2. Точка вимірювання, тип даних Real
3. Точка вимірювання, тип даних Real
.
.
.
10. Точка вимірювання, тип даних Real

Масив з ім'ям "Точка вимірювання" ("Measuring_point"): (Використовують, коли треба мати декілька елементів одного типу даних)

Представлення у програмному редакторі (блок даних DB 2):

DB2 -- My_Project\My_Station\CPU 314

Address	Name	Type	Initial value	Comment
*0.0		STRUCT		
+0.0	Measuring_point	ARRAY[1..10]		
*4.0		REAL		
=40.0		END_STRUCT		

Рис. 6.12. Однорідні структур даних

Дані двигуна

- Число обертань, тип даних Integer
- Номінальний струм, тип даних Real
- Пусковий струм, тип даних Real
- Направлення обертання, тип даних Bool

Структура з ім'ям "Дані двигуна" ("Motor_data") (Використовують, коли треба мати декілька елементів з різними типами даних)

Представлення у програмному редакторі (блок даних DB 1):

LAD/STL/FBD - [DB1 -- My_Project\My_Station\CPU 314]
File Edit Insert PLC Debug View Options Window Help

Address	Name	Type	Initial value	Comment
0.0		STRUCT		
+0.0	Motor_data	STRUCT		
+0.0	speed	INT	0	
+2.0	rated_current	REAL	0.000000e+000	
+6.0	starting_current	REAL	0.000000e+000	
+10.0	direction	BOOL	FALSE	
=12.0		END_STRUCT		
=12.0		END_STRUCT		

Рис. 6.13. Неоднорідні структур даних

6.3. Приклади робототехнічних систем на основі ПЛК

На рис. 8.14 наведений приклад програми циклового керування з використанням RS-тригера.

Позиція зупинок визначається кінцевими вимикачами, спрацювання яких зупиняє відповідних рух та починає наступний.

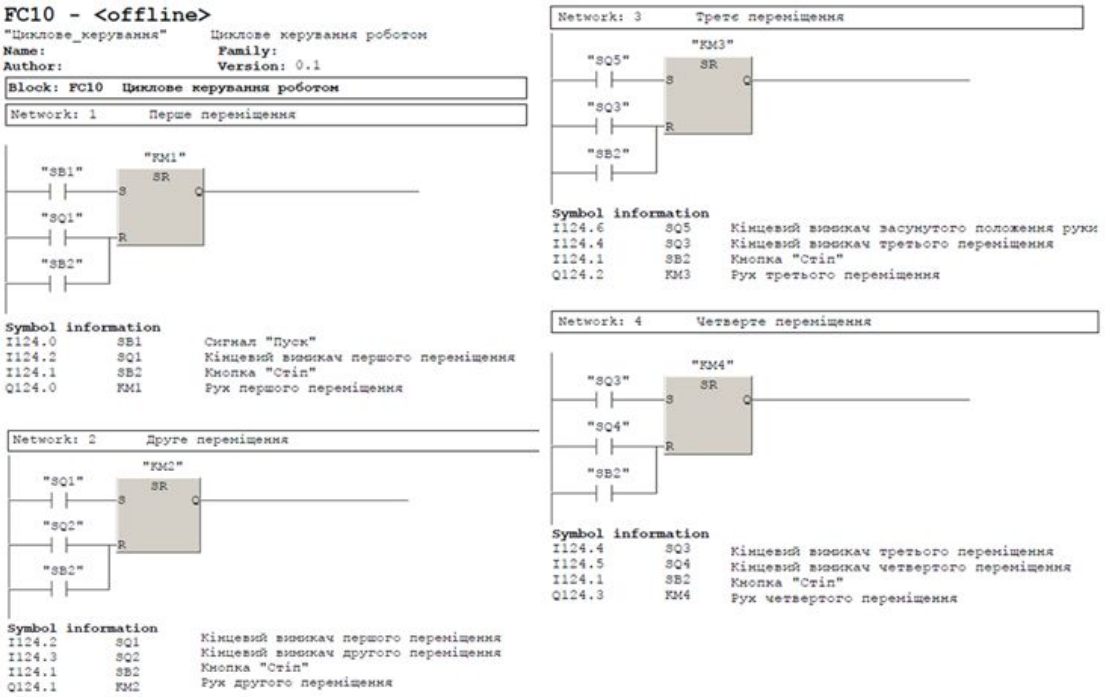


Рис. 8.14. Приклад програми циклового керування

На рис. 6.15 наведений приклад програми позиційного керування з використанням лічильника.

Рис. 6.15. Пиклад програми позиційного керування з використанням лічильника

Контрольні запитання

1. Визначити,які типи боків використовують ПЛК Simatic S7?
2. Описати,які функції виконують організаційні блоки?
3. Назвати,для чого використовують організаційні блоки переривання?
4. Визначити,для чого використовують організаційні блоки помилок?
5. Описати,як задаються параметри у функцій?
6. Назвати,як здійснюється виклик функцій з параметрами?
7. Визначити,чим відрізняються функціональні блоки?
8. Назвати,для чого використовують екземплярні блоки даних?
9. Описати,для чого використовують однорідні структури даних (ARRAY)?
10. Визначити,для чого використовують неоднорідні структури даних (STRUCT)?

Змістовий модуль 3. Комп'ютерні засоби проектування універсальних роботів

Лекція 7. Мови програмування універсальних роботів

7.1. Програмування роботів фірми KUKA за допомогою мови KRL

Для ралізації переміщення робочих органів та інших функцій промислових роботів використовують використовують різні мови програмування, які здійснюють перерахунок траєкторії переміщення робочого органа в функції керування виконавчими пристроями.

Різні фірми використовують різні мови програмування роботів, тому розглянемо деякі з них.

Для програмування роботів фірми KUKA використовується мова програмування KRL, за допомогою якого можна створити програми з усіма типами програмного керування, а саме циклового, позиційного та контурного керування [17].

Для цього у програмі можуть бути запрограмовані такі переміщення:

- переміщення від точки до точки (PTP),
- лінійне переміщення (LIN),
- кругове переміщення (CIRC).

Мова програмування KRL має структуру у вигляді послідовності кадрів (рис. 7.1). Перша команда руху в програмі KRL повинна містити однозначне визначення вихідного положення. Ця умова виконується в позиції HOME, яка вводиться в систему керування роботом за замовчуванням.

```
1  DEF my_program( )
2  INI
3
4  PTP HOME  Vel= 100 % DEFAULT
   ...
8  LIN point_5 CONT Vel= 2 m/s CPDAT1 Tool[3] Base[4]
   ...
14 PTP point_1 CONT Vel= 100 % PDAT1 Tool[3] Base[4]
   ...
20 PTP HOME  Vel= 100 % DEFAULT
21
22 END
```

Рядок	Опис
1	Рядок DEF показує ім'я програми.
2	Рядок INI містить в собі ініціалізації для внутрішніх змінних та параметрів.
4	Позиція HOME
8	Переміщення LIN
14	Переміщення PTP

Рис. 7.1. Структура програми системи керування KR C2 фірми KUKA

Для команд, які часто використовуються, є вмонтовані формуляри, що спрощують складання програми. При використанні формулярів введення команди здійснюється шляхом вибору необхідних параметрів у меню маски.

Далі наведені формуляри для команд переміщення.

Переміщення від точки до точки (PTP)

Робот виконує переміщення робочого органу уздовж найбільш швидкої траєкторії до цільової точці. Найбільш швидкою траєкторією, як правило, не є найкоротша траєкторія, тобто не пряма. Так як осі робота здійснюють обертальний рух, криволінійні траєкторії можуть виконуватися швидше, ніж прямі. Точний хід руху непередбачуваний. (рис. 7.2)

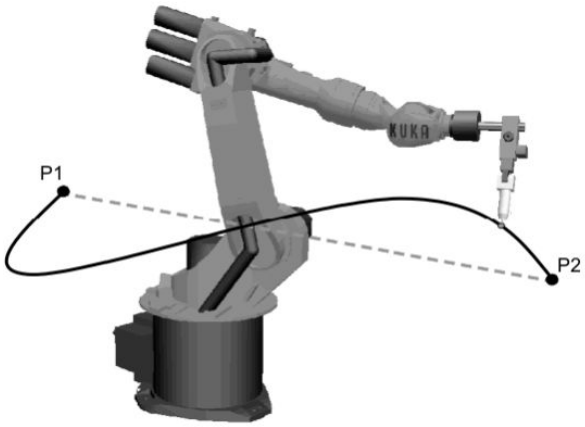


Рис. 7.2. Команда PTP

Формуляр переміщення від точки до точки (PTP) наведений на рис. 7.3.

Рис. 7.3. Формуляр переміщення від точки до точки (PTP)

Поз.	Опис	Діапазон значень
1	Вигляд переміщення	PTP, LIN, CIRC
2	Ім'я кінцевої точки	" Ім'я"
3	CONT: згладжування кінцевої точки пусто: точний підхід до кінцевої точки	CONT
4	Швидкість	0% ... 100%
5	Ім'я запису даних переміщення. Система автоматично задає ім'я.	" Ім'я"

Лінійне переміщення (LIN)

Робот виконує переміщення робочого органу з певною швидкістю уздовж найкоротшої траєкторії до цільової точці. Найкоротшою траєкторією завжди є пряма (рис. 7.4).

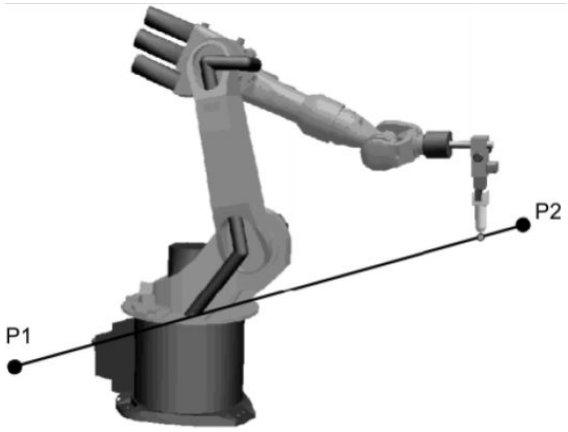


Рис. 7.4. Команда LIN

Формуляр лінійного переміщення (LIN) наведений на рис. 7.5.

Рис. 7.5. Формуляр лінійного переміщення (LIN)

Поз.	Опис	Діапазон значень
1	Вигляд переміщення	PTP, LIN, CIRC
2	Ім'я кінцевої точки	" Ім'я"
3	CONT: згладжування кінцевої точки пусто: точний підхід до кінцевої точки	CONT
4	Швидкість	0% ... 100%
5	Ім'я запису даних переміщення. Система автоматично задає ім'я.	" Ім'я"

Кругове переміщення (CIRC)

Робот виконує переміщення робочого органу з певною швидкістю уздовж кругової траєкторії до цільової точці. Кругова траєкторія задається початковою, допоміжної і цільової точкою (рис. 7.6).

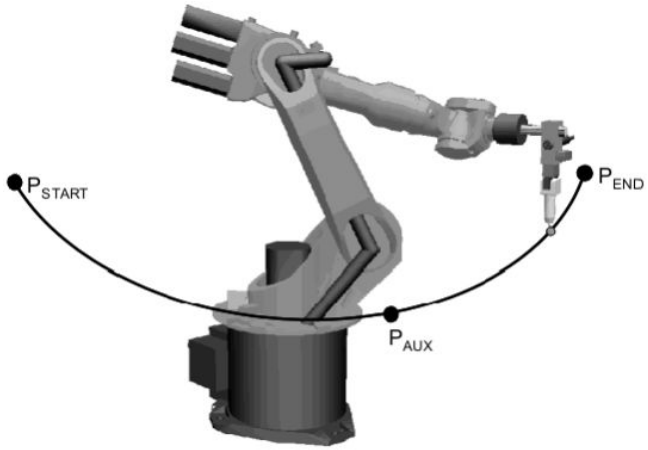


Рис. 6.6. Команда CIRC

Формуляр кругового переміщення (CIRC) наведений на рис. 7.7.

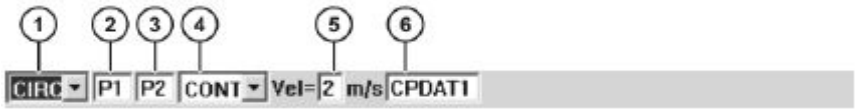


Рис. 7.7. Формуляр кругового переміщення (CIRC)

Поз.	Опис	Діапазон значень
1	Вигляд переміщення	PTP, LIN, CIRC
2	Ім'я допоміжної точки	" Ім'я"
3	Ім'я кінцевої точки	" Ім'я"
4	CONT: згладжування кінцевої точки пусто: точний підхід до кінцевої точки	CONT
5	Швидкість	0% ... 100%
6	Ім'я запису даних переміщення. Система автоматично задає ім'я.	" Ім'я"

Початковою точкою переміщення завжди є кінцева точка попереднього переміщення.

Згладжування використовується для здійснення плавного руху. Це робиться за допомогою параметра CONT. Приклади згладжування для переміщень LIN (а) та CIRC (б), наведений на рис. 7.8.

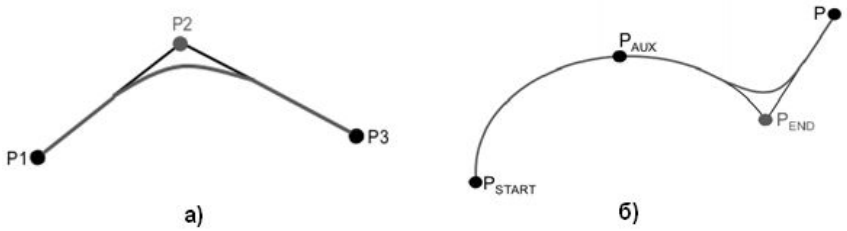


Рис. 7.8. Приклади згладжування для переміщень: а) LIN, б) CIRC

Переміщення LIN та CIRC можна об'єднати також поняттям "переміщення CP" (Continuous Path).

На рис 7.9 наведена програма та відповідна траєкторія переміщення, де згладжені точка запуску переміщення та цільова точка.

```
LIN P1 CONT VEL=0.3m/s CPDAT1
SYN OUT 1 '' State= TRUE at START PATH=20mm Delay=-5ms
LIN P2 CONT VEL=0.3m/s CPDAT2
LIN P3 CONT VEL=0.3m/s CPDAT3
LIN P4 VEL=0.3m/s CPDAT4
```

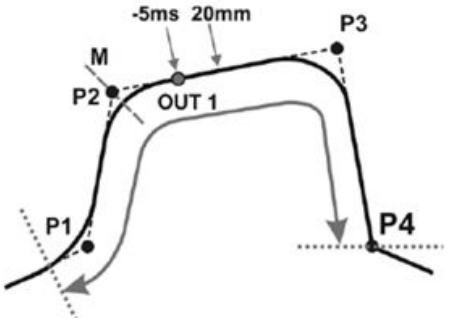


Рис. 7.9. Програма та траєкторія переміщення з згладженими точками запуску переміщення та цільової точки

Після створення програми є можливість її перевірки у тестовому режимі з високою та зниженою швидкістю.

Для керування захватним пристроєм, який має тільки два положення (відкритий та закритий) можна використати формуляр цифрового виходу OUT. Ця команда встановлює значення для цифрового входу (0 - FALSE або 1 - TRUE).

Формуляр встановлення цифрового виходу (OUT) наведений на рис. 7.10.

Рис. 7.10. Формуляр встановлення цифрового виходу (OUT)

Поз.	Опис	Діапазон значень
1	Номер виходу	1 … 4096
2	Якщо вихід має ім'я, то воно відображається	" Ім'я"
3	Стан, у який переключається вихід	TRUE, FALSE
4	CONT: обробка у попередній процедурі (пусто): обробка з зупинкою попередньої процедури	CONT, (пусто)

Для очікування відкриття або закриття захватного пристрою можна використати команду затримки WAIT.

Формуляр затримки (WAIT) наведений на рис. 7.11.

Рис. 7.11. Формуляр затримки (WAIT)

Поз.	Опис	Діапазон значень
1	Час очікування	≥ 0 сек

7.2. Програмування роботів з контурною системою керування

Для програмування роботів з контурною системою керування часто використовуються системи числового програмного управління (ЧПУ), призначені для верстатів та обробляючих центрів [8].

Для переміщення по контуру використовуються різні засоби інтерполяції, такі як лінійна, колова та інші.

Програмування переміщення здійснюється за допомогою так званої системи G-команд. Програма складається з кадрів, в яких як правило включені команди для зазначення параметрів одного переміщення, наприклад, функція руху (прискорений, лінійна чи кругова інтерполяція і т.д.), координати кінцевої точки руху, швидкість переміщення тощо.

В командах може використовувати абсолютне або відносне зазначення розміру. У першому разі усі розміри зазнаються відносно початкової точки координат. У другому – відносно положення робочого органу на початок виконання команди.

Деякі команді переміщення наведені у табл. 7.1.

Таблиця 7.1

Команда	Зазначення
G0 X… Y… Z…	прискорений рух до точки X… Y… Z… (найбільш можлива швидкість)
G1 X… Y… Z… F…	лінійна інтерполяція до точки X… Y… Z… (F… задана швидкість переміщення мм/хв)
G2 X… Y… Z… CR=…	колова інтерполяція за годинниковою стрілкою до точки X… Y… Z… з радіусом CR=…
G3 X… Y… Z… CR=…	колова інтерполяція проти годинникової стрілки до точки X… Y… Z… з радіусом CR=…
G90	абсолютне зазначення розміру
G91	відносне зазначення розміру

Для приклада розглянемо програму, яка виконує таку послідовність дій:

- швидке переміщення в початкову позицію X=10, У=10,
- лінійне переміщення у позицію X=60, У=50 при заданій швидкості 500 мм/хв,
- переміщення за допомогою колової інтерполяції у позицію X=110, У-30 із радіусом CR=30.

Траєкторія, яка здійснює таке переміщення наведена на рис. 7.12.

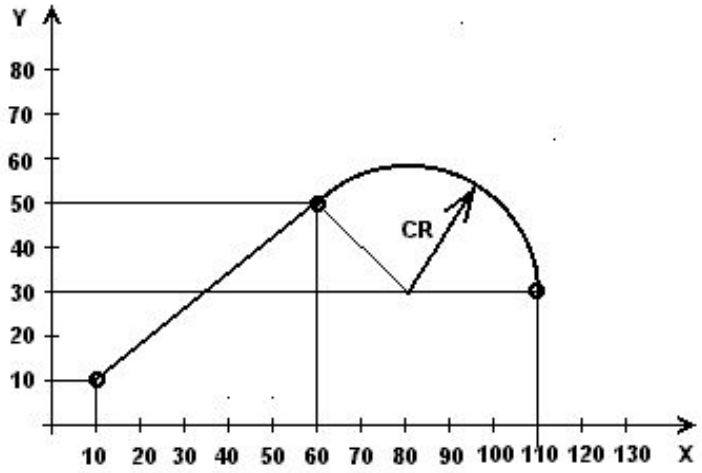


Рис. 7.12. Траєкторія переміщення з лінійною та коловою інтерполяцією

Відповідна програма має такий вигляд:

N10 G0 G90 X10 Y10
N20 G1 X60 Y50 F500
N30 G2 X110 Y30 CR=30

7.3. Створення програми за допомогою програмної середи LabVIEW

Універсальним засобом програмування робототехнічних пристроїв є програмна середа LabVIEW, яка здійснює програмування у графічному вигляді та має вмонтовані засоби для програмування пристроїв на основі Mindstorms NXT [1, 9].

Розглянемо, як можна створити програму за допомогою програмної середи LabVIEW.

Всі дії програмування зводяться до простої побудови структурної схеми в інтерактивній графічній системі з набором усіх необхідних бібліотечних образів, з яких збираються об'єкти, звані Віртуальними Інструментами (VI), завдяки чому LabVIEW став одним з найпопулярніших у світі програмних продуктів для систем збору даних, їх аналізу, обробки і візуалізації.

Кожна програма складається з фронтальної панелі та блок-діаграми (рис. 7.13).

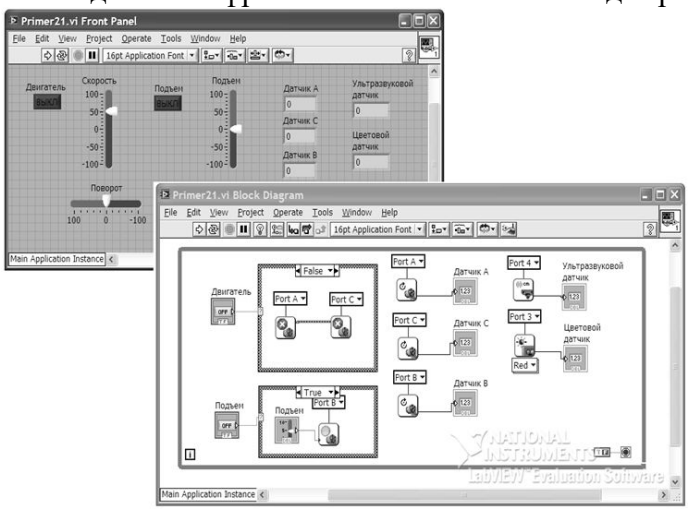


Рис. 7.13. Фронтальна панель (зліва) та блок-діаграма (справа)

Фронтальна панель - це місце, де створюється інтерфейс користувача, на ній можна встановити елементи введення і відображення даних. Фронтальна панель дозволяє забезпечити інтерактивну роботу проектованого пристрою, тобто дає можливість задавати потрібні параметри за допомогою елементів введення даних і управління, а також відображати необхідну інформацію, як в числовому, так і в графічному поданні безпосередньо на екрані комп'ютера. Фронтальна панель створюється за допомогою палітри елементів (Controls).

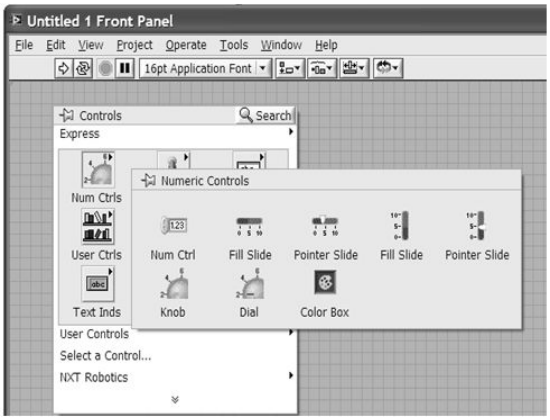


Рис. 7.14. Палітри елементів для створення фронтальної панелі

Ці елементи можуть бути засобами введення даних (елементи управління) або засобами відображення даних (елементами відображення).

До елементів управління відносяться кнопки, перемикачі, повзунки та інші елементи введення (рис. 7.15).

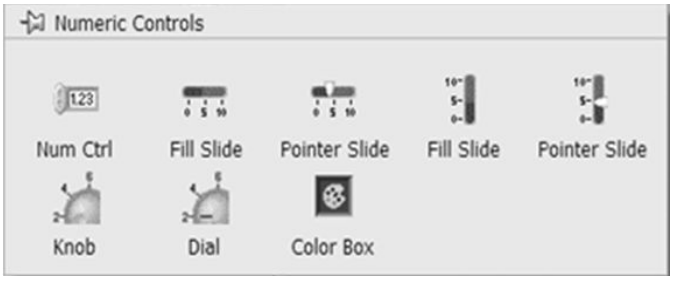


Рис. 7.15. Елементи палітри для створення фронтальної панелі

До елементів відображення відносяться індикатори, графіки, цифрові табло, світлодіоди і т.д.

Елементи, що створюються на фронтальній панелі, відразу відображаються на блок-діаграмі і є елементами програми.

Вони не можуть бути вилучені на блок-діаграмі.

На рис. 7.16 показаний порядок встановлення властивостей елементів фронтальної панелі. При натисканні правою кнопкою миші відкривається спливаюче меню, де треба вибрати пункт **Properties**. Після цього відкривається вікно для встановлення властивостей вибраного елемента.

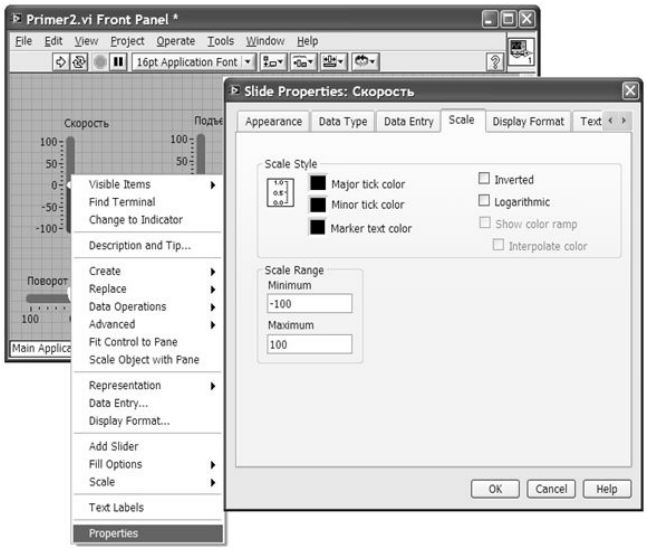


Рис. 7.16. Порядок встановлення властивостей елементів фронтальної панелі

Блок-діаграма - це місце, де створюється програма у вигляді графічного представлення функцій та Віртуальних Інструментів (VI).

Блок-діаграма складається з блоків, які реалізують окремі функції, що виконуються програмою.

Для вибору блоків використовується палітра функцій (Functions), що наведена на рис. 7.17.

За допомогою палітри функцій можна вибрати піктограми різних елементів програми, зокрема, структур програмування, арифметичних і логічних операцій і т.д.

Властивості елементів блок-діаграми встановлюються аналогічно властивостям елементів фронтальної панелі.

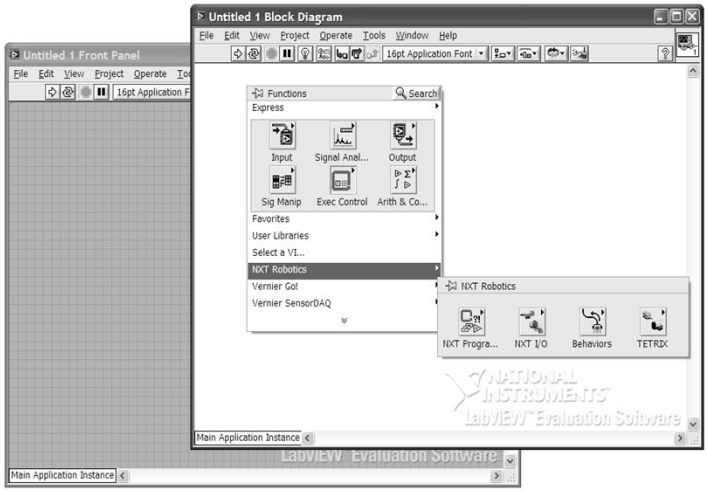


Рис. 7.17. Палітра функцій для вибору блоків блок-діаграми

Програми LabVIEW можуть виконуватися в двох режимах: прямому і віддаленому.

У прямому режимі блок NXT підключений до комп'ютера через USB або Bluetooth, і програма виконується на комп'ютері.

Таким чином, блок NXT використовується як периферійний пристрій.

У віддаленому режимі програма, створена на комп'ютері, після компіляції завантажується в блок NXT та виконується з нього в будь-який час в автономному режимі.

При цьому вже немає необхідності в підключенні блоку NXT до комп'ютера.

Для програмування роботів є бібліотеки, що містять функції для датчиків, виконавчих пристроїв та інших засобів, що використовуються у роботах.

Програмний пакет LabVIEW Robotics дозволяє створювати програми з урахуванням можливостей можливостей різних компонент роботів, включаючи датчики та виконавчі пристрої. На рис. 7.18 наведеі палітри з функціями, що використовуються при проектуванні роботів

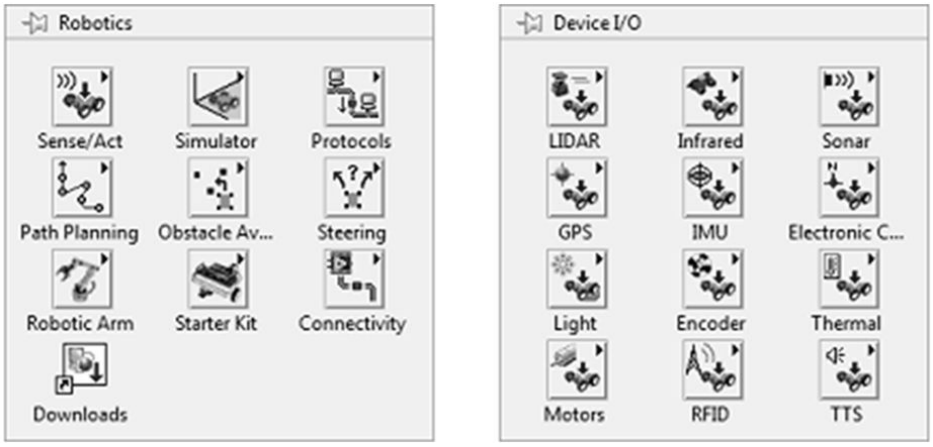
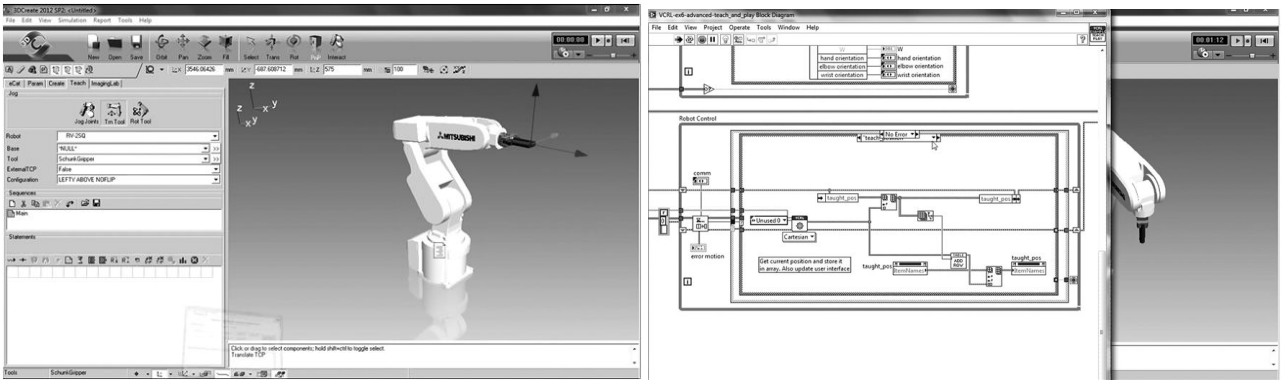


Рис. 7.18 . Палітри з функціями, що використовуються при проектуванні роботів

Приклади проектування робототехнічних систем на основі програмної середи LabVIEW

На рис. 7.19,а наведено вікно для створення проекту з роботами Mitsubishi Robotics, яка дозволяє здійснити створення програми у середі LabVIEW. На рис. 7.19,б наведено створення програми у середі LabVIEW.

Ця програмна середа дає також можливість спільного використання LabVIEW та SolidWorks.



а) б)

Рис. 7.19. Вікно для створення проекту з роботами Mitsubishi Robotics (а) та програми у середі LabVIEW (б)

Наявність різноманітних елементів дає можливість створювати досить складні програми керування з різними способами програмного керування, а елементи для роботи з датчиками зовнішньої інформації дозволяють реалізувати адаптивне керування.

Програмна середа LabVIEW використовувалась для створення програмного забезпечення марсохода Sojourner (рис. 7.21).

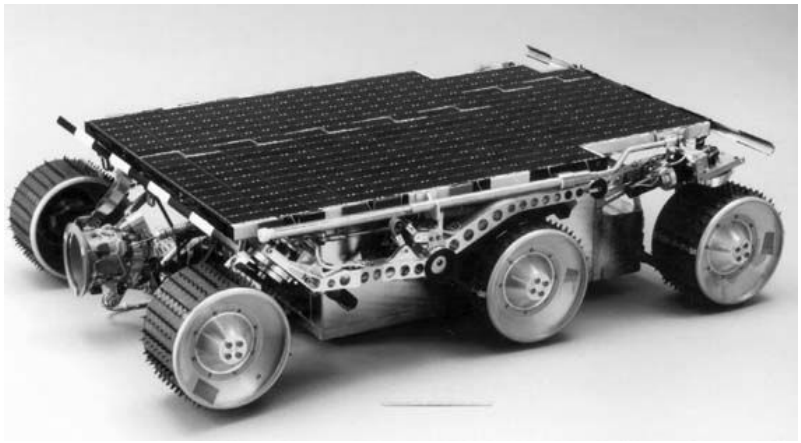


Рис. 7.21. Марсоход Sojourner

Контрольні запитання

1. Назвати,з яких частин складається програма в LabVIEW?
2. Визначити,для чого використовується фронтальна панель?
3. Описати,які елементи управління використовуються у фронтальній панелі?
4. Назвати,які елементи відображення використовуються у фронтальній панелі?
5. Визначити,для чого використовується блок-діаграма?
6. Описати,які елементи використовуються у блок-діаграмі?
7. Назвати,який пакет використовують для проектування роботів?
8. Розповісти,в яких режимах можуть виконуватися програми LabVIEW?
9. Визначити,які інтерфейси використовуються для підключення робототехнічних пристроїв?
10. Розповісти,які компоненти використовують для створення проекту з роботами Mitsubishi Robotics?

Лекція 8. Програмування робототехнічних пристроїв за допомогою програмного засобу MICROSOFT ROBOTICS DEVELOPER STUDIO

8.1. Структура та склад програмного засобу MICROSOFT ROBOTICS DEVELOPER STUDIO

Корпорація Microsoft випустила програмне забезпечення Microsoft Robotics Developer Studio, яке призначено для створення програмного забезпечення роботів і різних роботизованих механізмів та дослідження алгоритмів керування роботами завдяки засобам симуляції як самого робота, так і зовнішньої середи, де здійснюється функціонування робота [2].

MRDS складається з декількох компонентів.

1. Concurrent and Coordination Runtime (CCR) - середовище організації паралельної обробки даних.

2. Decentralized Software Services (DSS) - середовище, яке дозволяє запускати алгоритми обробки даних на різних ЕОМ, організовувати асинхронне взаємодія процесів управління різними

підсистемами робота.

3. Visual Simulation Environment (VSE) - середовище візуалізації, яка дозволяє експериментувати з моделями роботів, тестувати алгоритми управління.

4. Visual Programming Language (VPL) - мова, призначена для розробки програм управління роботами. Програма такою мовою представляється у вигляді послідовності блоків, які виконують обробку даних, і здійснюють зв'язки між ними. VPL розрахований на управління як реальними роботами, так і моделями роботів в симуляторі.

Concurrency and Coordination Runtime - це бібліотека для роботи з паралельними і асинхронними потоками даних, яка базується на .NET Framework. Крім робототехніки, може застосовуватися для поліпшення асинхронности в будь-яких додатках.

При взаємодії з навколишнім середовищем, робот повинен правильно реагувати на інформацію, що надходить одночасно від безлічі датчиків, тому істотну частину логіки перенесено на комп'ютер, або безліч взаємодіючих між собою комп'ютерів, які можуть перебувати окремо від робота.

В результаті цього, була спеціально розроблена **бібліотека CCR**, яка дозволяє з легкістю створювати код для паралельного виконання і масштабування.

Decentralized Software Services - це полегшене середовище, що базується на CCR, для створення розподілених додатків на основі сервісів, яка передбачає управління безліччю сервісів для коригування поведінки в цілому.

Visual Programming Language - це мова візуального програмування, розроблений корпорацією Microsoft спеціально для Microsoft Robotics Developer Studio. VPL призначена для початківців програмістів, які знають основні принципи, такі як алгоритми і змінні.

Visual Simulation Environment - це середовище симуляції. Так як не у всіх є роботи, вона допомагає симулювати поведінку роботів у віртуальному середовищі. Для забезпечення реалістичності в віртуальному світі використовується технологія NVIDIA PhysX.

Microsoft Robotics Developer Studio підтримує наступні моделі роботів: Boe-Bot, CoroBot, iRobot, Mindstorms NXT, Pioneer 3Dx, KUKA LBR3 і інші.

Microsoft RDS включає в себе спеціальну програмну модель для створення програм керування, а також набір візуальних та сімуляційних інструментів, які можуть знадобитись при складанні програмного забезпечення для роботів.

Компоненти Robotics Studio інтегріруються в середу розробки Visual Studio, який має два основних модуля.

Це такі модулі:

Visual Programming Language (**VPL**, візуальний язик програмування);

Visual Simulation Environment (**VSE**, сімуляційна середа).

Язик VPL забезпечує можливість програмування роботів візуальними методами.

Діаграми VPL кодуються за допомогою XML-схем і дають можливість створення повністю візуального язика програмування.

Другий важливий модуль Robotics Studio - **симуляційне середовище VSE**.

VSE є графічною 3D-моделлю, що відображає дії роботів, і об'єкти, які оточують ці роботи.

VSE включає можливість запису та повторного відтворення симуляції. Крім того, проектанти можуть виконувати симуляції у різних віртуальних умовах - в умовах закритого приміщення або відкритого простору.

Крім того, VSE має можливість експорту з різних CAD-програм, наприклад, SolidWorks 3D.

Robotics Studio також дає можливість створювати досить складні програми керування з різними способами програмного керування, а наявність блоків для роботи з датчиками зовнішньої інформації, наприклад, блока для роботи з VEB-камерою, дозволяють реалізувати адаптивне керування на основі обробки зображення.

8.2. Мова візуального програмування VPL

Діаграма складається з блоків (Activity). Блоки можуть бути двох типів: вбудовані блоки (Built-In-Activities) і сервіси (Services).

Базові блоки, що входять до MRDS, знаходяться на панелі інструментів Basic Activity. На відміну від сервісів у них немає унікального імені.

Сервіс являє собою інтерфейс до апаратного або програмного забезпечення робота. Вбудовані блоки дозволяють управляти процесом виконання сервісів. На основі вбудованих блоків виконується організація циклів, визначення констант, робота зі змінними, здійснюється передача повідомлень між блоками і т. П. Одна з особливостей мови візуального програмування полягає в тому, що блок починає виконуватися тільки, коли на його вхід надходить повідомлення.

Повідомлення в VPL подібні структурам в мовах Pascal або C ++. Повідомлення може включати одне або кілька полів, кожне з яких описується ім'ям і типом даних.

Програма на мові VPL є діаграму потоків даних, переданих між сервісами і блоками, а не послідовність команд і інструкцій, як на мовах програмування C або Pascal.

Вікно середи розробки діаграм Microsoft Robotics Developer Studio наведено на рис. 8.1.

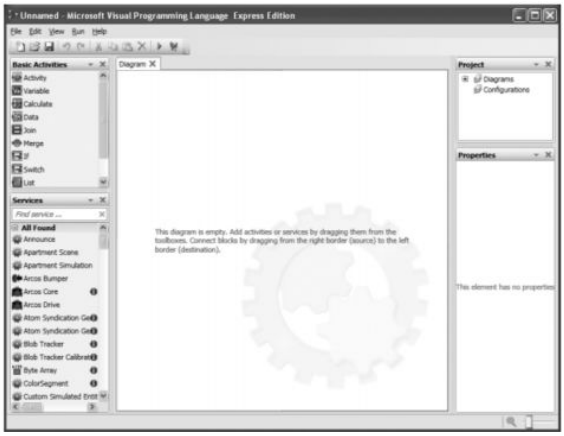


Рис. 8.1. Вікно середи розробки діаграм Microsoft Robotics Developer Studio

Центральна частина вікна - робоча область, в якій розміщується діаграма. Робоча область за замовчуванням оточена чотирма панелями інструментів (toolboxes).

У середу розробки входять наступні панелі інструментів.

1. Basic Activity - включає блоки для управління і організації потоків даних на діаграмі.

2. Services - містить послуги, що входять в MRDS. Дана панель інструментів дозволяє виконувати пошук сервісу по імені. Для цього в рядку пошуку Find service ... потрібно ввести частину імені сервісу. Після цього відобразяться елементи, які містять в своєму імені введений рядок. Часто використовувані запити можна зберегти.

3. Project - відображає діаграми проекту і файли конфігурації проекту.

4. Properties - відображає властивості вибраного елемента діаграми (сервісу, вбудованого блоку або зв'язку між блоками).

5. Errors - список помилок, допущених при розробці діаграми. Блок, який вважається некоректним, відзначається знаком оклику (!). Перевірка діаграми на помилки виконується в фоновому режимі, тому результати перевірки з'являються з деякою затримкою

Програми можуть здійснювати введення даних та опитування датчиків, математичну обробку даних, керування виконавчими пристроями тощо. На рис. 8.2 наведений приклад програми у вигляді діаграм, яка здійснює визначення натиснутої кнопки.

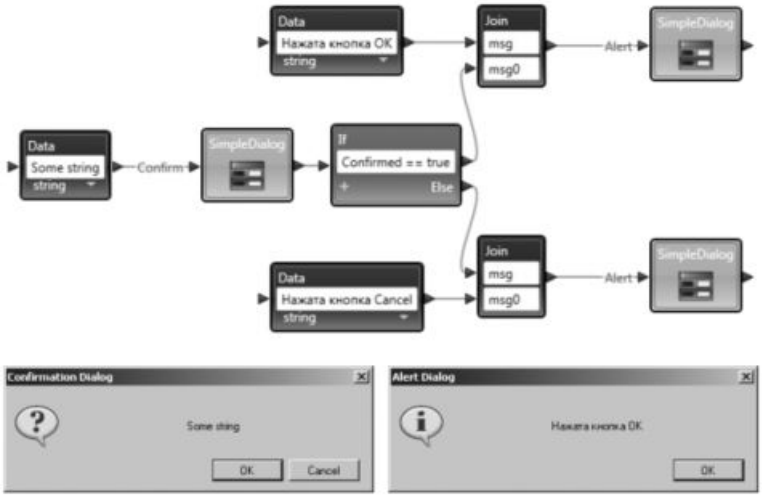


Рис. 8.2. Приклад програми у вигляді діаграм, яка здійснює визначення натиснутої кнопки

8.3. Середа сімуляції VSE

Visual Simulation Environment включає в себе дві програми: графічний і фізичний движок.

Графічний движок - основним завданням цієї програми є візуалізація (рендеринг) двомірної або тривимірної комп'ютерної графіки. Графічний движок працює в режимі реального часу.

Фізичний движок - виробляє симуляцію фізичних законів реального світу в віртуальному світі з тим або іншим ступенем точності.

Фізичний движок дозволяє створити віртуальний простір, в яке можна додати віртуальні статичні і динамічні об'єкти і вказати закони взаємодії тіл і середовища. Розрахунок взаємодії тел виконується самим двигуном. Розраховуючи взаємодія тіл між собою і середовищем, фізичний движок наближає фізичну модель одержуваної системи до реальної, передаючи уточнені геометричні дані графічному движку. До складу MRDS входить фізичний движок AGEIA PhysX Engine.

Об'єкти в симуляторі можуть створювати ієрархію, реалізуючи відношення предок/нащадок. Наприклад, маніпулятор і сенсор є дочірніми об'єктами робота.

Після запуску симулятора в ньому відображається світ з позиції головної камери, яка відповідає позиції очей. Для зміни напрямку головної камери натисніть ліву кнопку миші і наведіть курсор в бажаному напрямку. Для управління камерою за допомогою таких клавіш клавіатури:

W - рух вперед;

S - рух назад;

A - пересування вліво;

D - пересування вправо;

Q - переміщення вгору;

E - переміщення вниз.

Можна використовувати поєднання цих клавіш. Утримуючи Shift одночасно з однією з клавіш управління камерою, ви прискорите рух камери в 10 разів.

В наявності є таке команди (клавіші) управління симулятором:

F2 - змінює режим рендеринга;

F3 - включає / вимикає фізичний движок;

F5 - перемикання між режимами Run і Edit;

F8 - перемикання активної камери.

У нижній частині вікна симулятора розташована рядок стану (пункт меню View → Status Bar). У рядку стану відображається позиція активної камери, представлена трьома координатами X, Y і Z, де осі OX і OZ представляють напрямки, паралельні площині землі, вісь OY розташовується перпендикулярно землі.

Крім того, використовується права координатна система. Це означає, що напрямок осі OX вказує направо, осі OY - вгору, вісь OZ вказує напрямок за межі екрану. При цьому напрямок очі відповідає негативному напрямку осі OZ, т. Е. Вглиб екрану.

Симулятор може працювати в наступних двох режимах.

1. Edit - в даному режимі можлива зміна сцени, параметрів об'єктів, що знаходяться на сцені, додавання нових об'єктів в сцену.

2. Run - в даному режимі запускається процес симуляції.

Симулятор підтримує такі режими відображення сцени.

1. Visual - повністю відображаються всі об'єкти сцени.

2. WireFrame - відображаються тільки лінії трикутників, з яких побудовані об'єкти сцени.

3. Physics - відображаються фігури, з яких побудовані об'єкти і з якими працює фізичний движок. Складні об'єкти іноді представляються у вигляді простих об'єктів, таких як куби і сфери.

Якщо деякі об'єкти не стикаються або пересуваються випадковим чином, потрібно переключитися в зазначений режим і проаналізувати фігури, складові об'єкти.

4. Combined - об'єднує режими Visual і Physics (рис. 8.3).

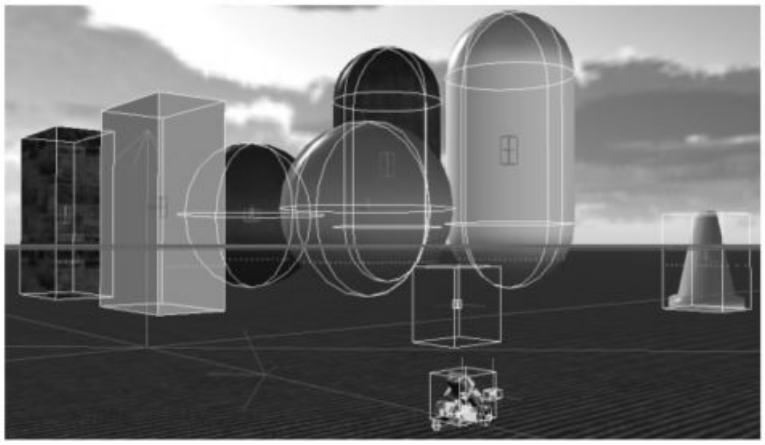


Рис. 8.3. Режим Combined

Параллепіпед, розміщений всередині об'єкта, показує центр мас. Якщо колір параллепіпед червоний, то об'єкт управляється вручну, якщо - білий, то об'єкт управляється фізичним симулятором.

Симултор має різні сервіси для керування роботом, наприклад, сервіс DesktopJoystick (рис. 8.4), який можна використовувати для віртуального керування рухом робота.

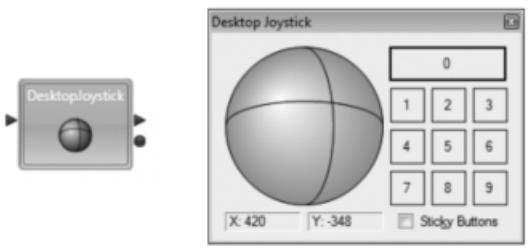


Рис. 8.4. Сервіс DesktopJoystick

На рис. 8.5 наведений приклад вікна симулятора з керуванням роботом iRobot за допомогою сервісу DesktopJoystick.

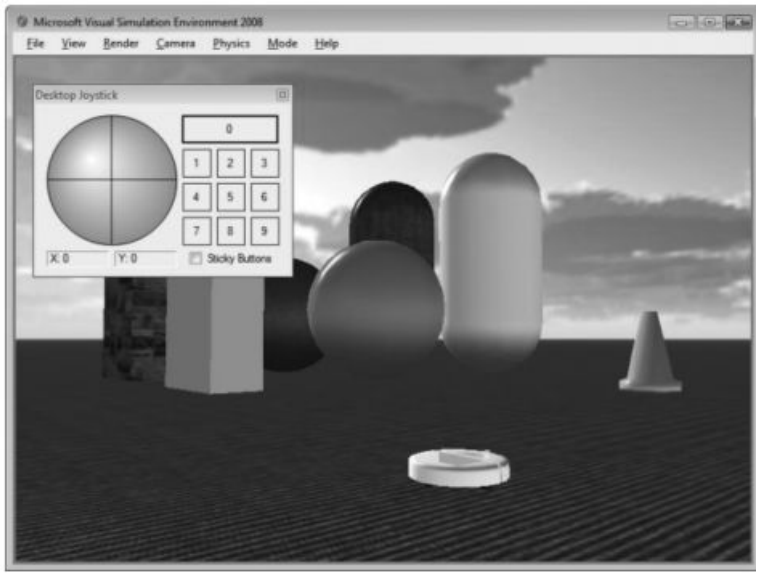


Рис. 8.5. Приклад вікна симулятора з керуванням роботом iRobot за допомогою сервісу DesktopJoystick

За допомогою програми можна здійснити переміщення робота згідно з вказаною траєкторією.

Наприклад, рух робота по вісімці складається з переміщення по колу. Після того як робот завершить рух по одному колу, він повинен змінити напрямок руху і почати рух по іншому колу. Діаграма для організації руху робота по вісімці приведена на рис. 8.6.

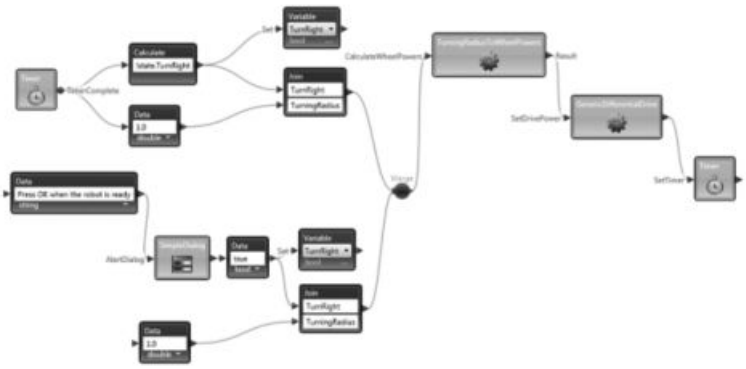


Рис. 8.6. Діаграма для організації руху робота по вісімці

8.4. Приклади використання MICROSOFT ROBOTICS DEVELOPER STUDIO

Прикладом використання середи MICROSOFT ROBOTICS DEVELOPER STUDIO є робот-пилосос Roomba, розроблений компанією iRobot, що являє собою роботизований пристрій для прибирання квартири (рис. 8.18).

Рис. 8.18. Робот-пилосос Roomba

Robotics Studio також дає можливість створювати досить складні програми керування з різними способами програмного керування, а наявність блоків для роботи з датчиками зовнішньої інформації, наприклад, блока для роботи з VEB-камерою, дозволяють реалізувати адаптивне керування на основі обробки зображення.

На базі платформи робота-пилососа Roomba компанією iRobot розроблений робот iRobot Create, призначений для розробників роботів, дозволяє програмувати поведінку робота (рис. 8.19).

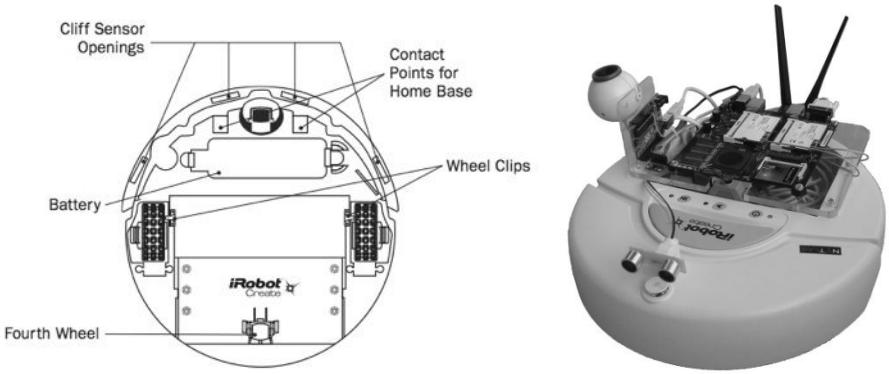


Рис. 8.19. Робот iRobot Create з пристроями для дослідження поведінки робота

Контрольні запитання

1. Визначити,з яких основних компонентів складається програмне забезпечення Microsoft Robotics Developer Studio?
2. Описати,для чого використовується модуль VPL?
3. Розповісти, як створюється програма за допомогою модуля VPL?
4. Назвати,які типові блоки використовує модуль VPL?
5. Визначити,що дозволяє зробити симуляційне середовище VSE?
6. Описати,для чого використовують маніфести?
7. Розповісти, як створити віртуальне середовище?
8. Визначити,які елементи використовують для керування рухом?
9. Назвати,які роботи розроблені компанією iRobot?
10. Описати,для чого використовується робот iRobot Create?

Лекція 9. Засоби проектування роботів ABB

9.1. Засоби проектування роботів ABB

RobotStudio це комп'ютерна програма для моделювання в автономному режимі програмування і моделювання клітин роботів [1, 18].

RobotStudio дозволяє працювати з контролером офф-лайн, який є віртуальним контролером IRC5, виконується локально на вашому комп'ютері. Цей контролер офлайн позначається як віртуальний контролер (VC). RobotStudio також дозволяє працювати з реальним фізичним контролером IRC5, який називають реальним контролером. Коли RobotStudio використовується з реальними контролерами, це називають режимом реального часу.

При роботі без підключення до реального контролеру, або, будучи підключений до віртуального контролеру, RobotStudio працює в автономному режимі.

RobotStudio пропонує наступні варіанти установки:

- повна;
- користувальницькі, що дозволяють користувачам настроювати зміст і шляхи;
- мінімальна, що дозволяє запускати RobotStudio тільки в онлайн-режимі.

Використання для різних моделей роботів IRB140 - IRB7600

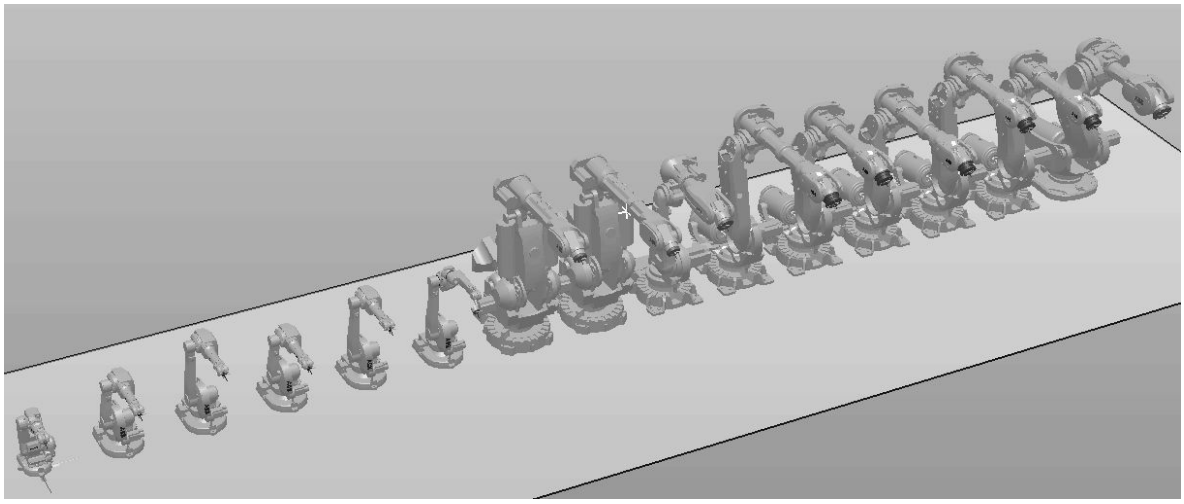


Рис. 9.1. Використання RobotStudio для різних моделей роботів

RobotStudio є симуляційною середою off-line програмування роботів компанії ABB. Основною перевагою off-line програмування є відсутність необхідності в наявності реального обладнання. Відзначимо, що переміщення (deploy) проекту з RobotStudio в контролер робота займає кілька хвилин. При правильно написаної off-line програмою для запуску в режимі on-line буде потрібно лише невелика корекція координат точок траєкторії робота (наприклад - координат зварних швів).

Розглянемо ситуації, коли може знадобитися використання RobotStudio:

- необхідно підібрати модель робота виходячи із зони досяжності;
- при пошуку виконавця замовник, як правило, звертається в кілька фірм - системних інтеграторів. Моделювання необхідного замовником процесу може скласти конкурентну перевагу компанії;
- процес покупки і доставки робота і обладнання може затягнутися на кілька місяців. У цей час інженери можуть промоделювати роботу РТК в off-line режимі;
- потрібно перевірити конструкційну досяжність інструменту, оснастки тощо Найчастіше геометрія спроектованого інструменту не підходить для роботи через виникаючих колізій. Перероблення конструкції затратно і може зайняти багато часу;
- необхідно впровадити робот на працюючій конвеєрної лінії. Монтаж і пусконалагодження комплексу повинна бути проведена в найкоротші терміни, наприклад за 8 годин. В цьому випадку всі програми повинні бути написані і налагоджені;

– часто логіка роботи РТК складна і вимагає налагодження. Налагодження складної програми в режимі on-line може привести до колізій і поломки дорогого устаткування;
– програмування в RobotStudio здійснюється в офісі.

При запуску ABB RobotStudio відкриється стартове меню, в якому потрібно вказати назву проекту (Solution Name), вказати директорію (Location), куди буде сохраннён проект, і вибрати "Solution with Empty Station" (рис. 9.2).

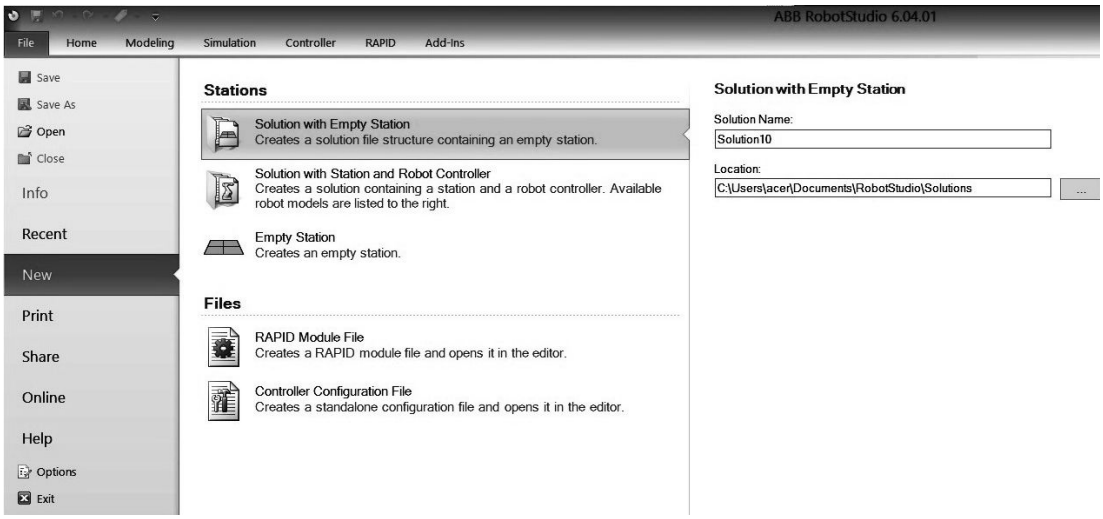


Рис. 9.2. Стартове меню

Після створення проекту відкриється порожня робоча область (рис. 9.3).

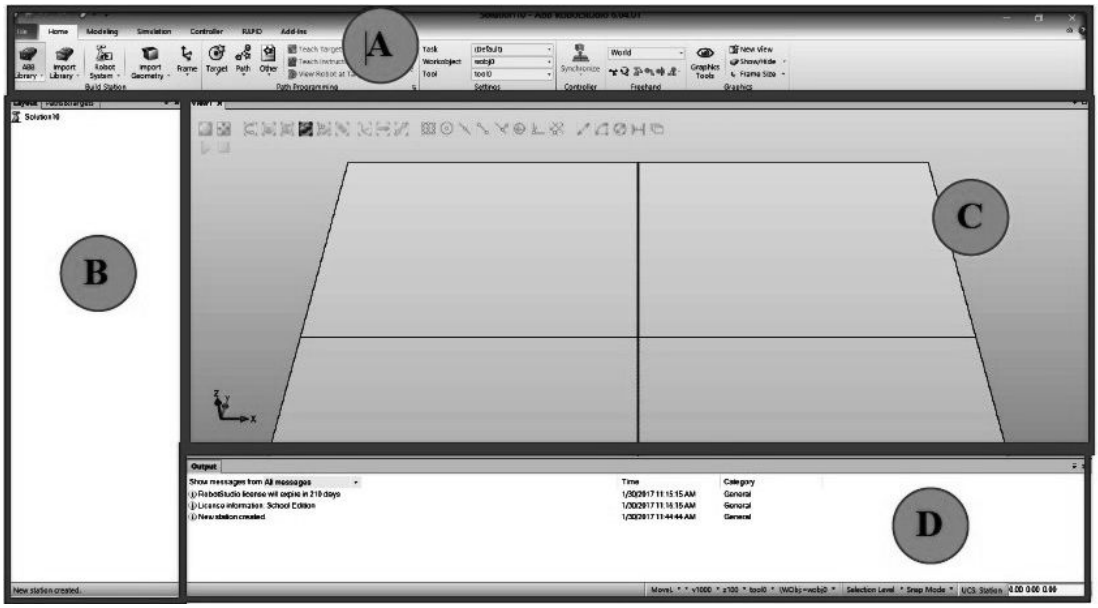


Рис. 9.3. Робоча область

A. Меню інструментів (тут знаходяться всі інструменти і налаштування середовища РоботСтудіо)

B. Провідник проекту (тут знаходяться всі елементи проекту, наприклад, роботи, деталі, інструменти, робочі області та т. Д.)

C. Графічна симуляція проекту

D. Отладчик (тут буде виведений список всіх скоєних дій в РоботСтудіо)

Щоб додати робота в «Меню інструментів (A)» вибираємо «ABB Library» і вибираємо робота зі списку представлених (рис. 9.4).

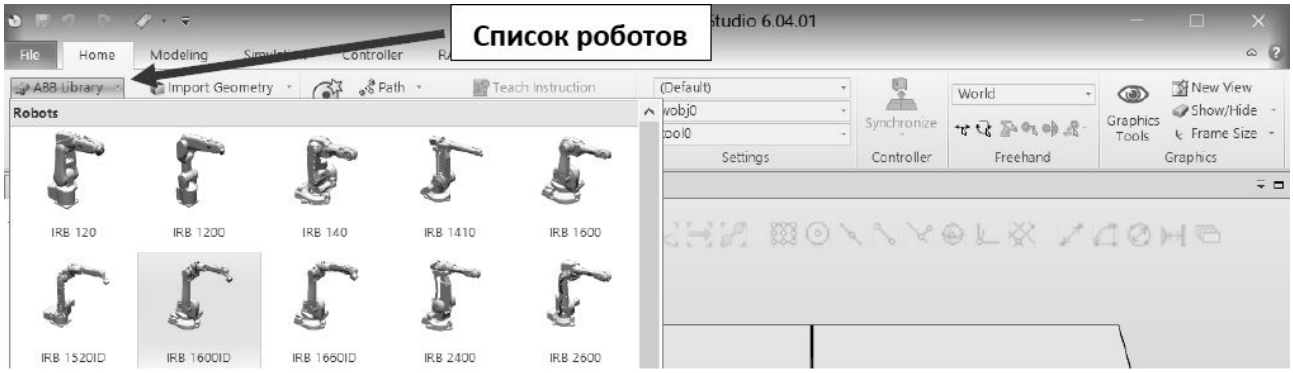


Рис. 9.4. Список роботів

Після вибору потрібного робота він додасться на графічний екран симуляції (С) (рис. 9.5).

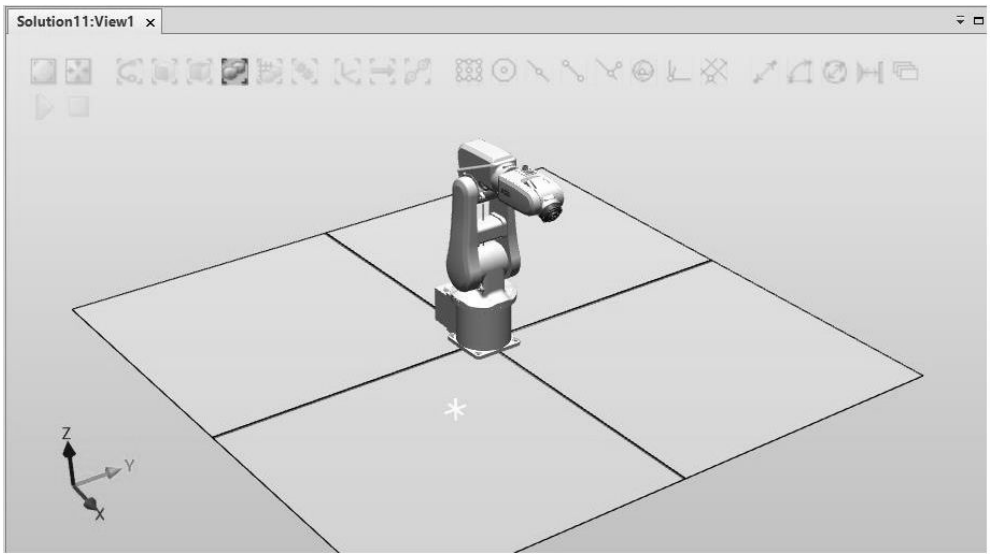


Рис. 9.5. Обраний робот на графічному екрані симуляції

9.2. Засоби створення моделей використання роботів ABB

Після обрання робота можна здійснити його перегляд аналогічно пересуванню камери.

Для того щоб наблизити або віддалити камеру, потрібно використовувати колесо миші. Наближення і віддалення камери буде відбувається на тому місці де знаходиться курсор миші.

Для того щоб перемішати камеру вліво в право, потрібно затиснути клавішу «Ctrl» і затиснути ліву кнопку мишки і рухати вліво в право.

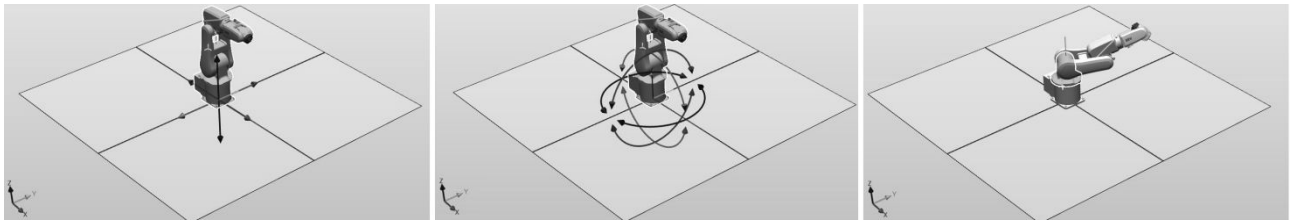
Для того щоб обертати камеру потрібно затиснути «Ctrl + Shift» і затиснути ліву кнопку мишки і рухати вліво в право, вгору-вниз.

Система робота потрібна для того, щоб була можливість управляти роботом і його інструментами.

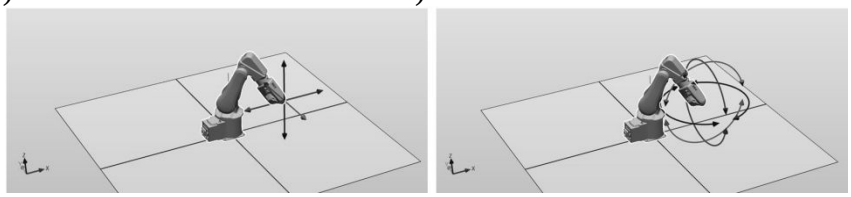
Для створення системи потрібно в головному меню вибрати «Robot System -> From Layout» (рис. 9.6).

Рис. 9.6. Вибір системи

Після запуску системи, є управління роботом. Для того, щоб змінити становище робота, потрібно в головному меню вибрати «Home» і функції «Move», що здійснюють, відповідно, зміну положення та обертання робота у просторі (рис. 9.7).



а) б) в)



г) д)

Рис. 9.7. Зміна становища робота:
а) функція «Move»- зміна положення, б) функція «Rotate» - обертання, в) функція «Jog joint» - змна положення, частини робота, г) функція «Jog linear» - лінійна зміна положення інструменту робота, д) функція «Jog Reorient» - обертання інструменту

Після запуску системи, є управління роботом. Для того, щоб змінити становище робота, потрібно в головному меню вибрати функції «Home» і «Move», що здійснюють, відповідно, зміну положення та обертання робота у просторі (рис. 9.7).

Для того щоб додати інструмент потрібно його вибрати з бібліотеки інструментів. Для цього вибираємо в головному меню «Home» -> «Import Library» -> «Equipment» -> «Tools» і вибираємо потрібний інструмент (рис. 9.8).

Після вибору інструменту він з'явиться в робочій області (рис. 9.9).

Щоб вставити інструмент в робота потрібно перетягнути об'єкт інструменту в «Layout» на робота (рис. 9.10).

Після додавання інструменту до робота, інструмент можна рухати і обертати інструмент і разом з цим буде змінювати своє положення робот.

Щоб додати деталь потрібно в головному меню вибрати «Home» -> «Import Library» -> «Equipment» -> «Training Objects» -> «Curve Thing» (рис. 9.11).

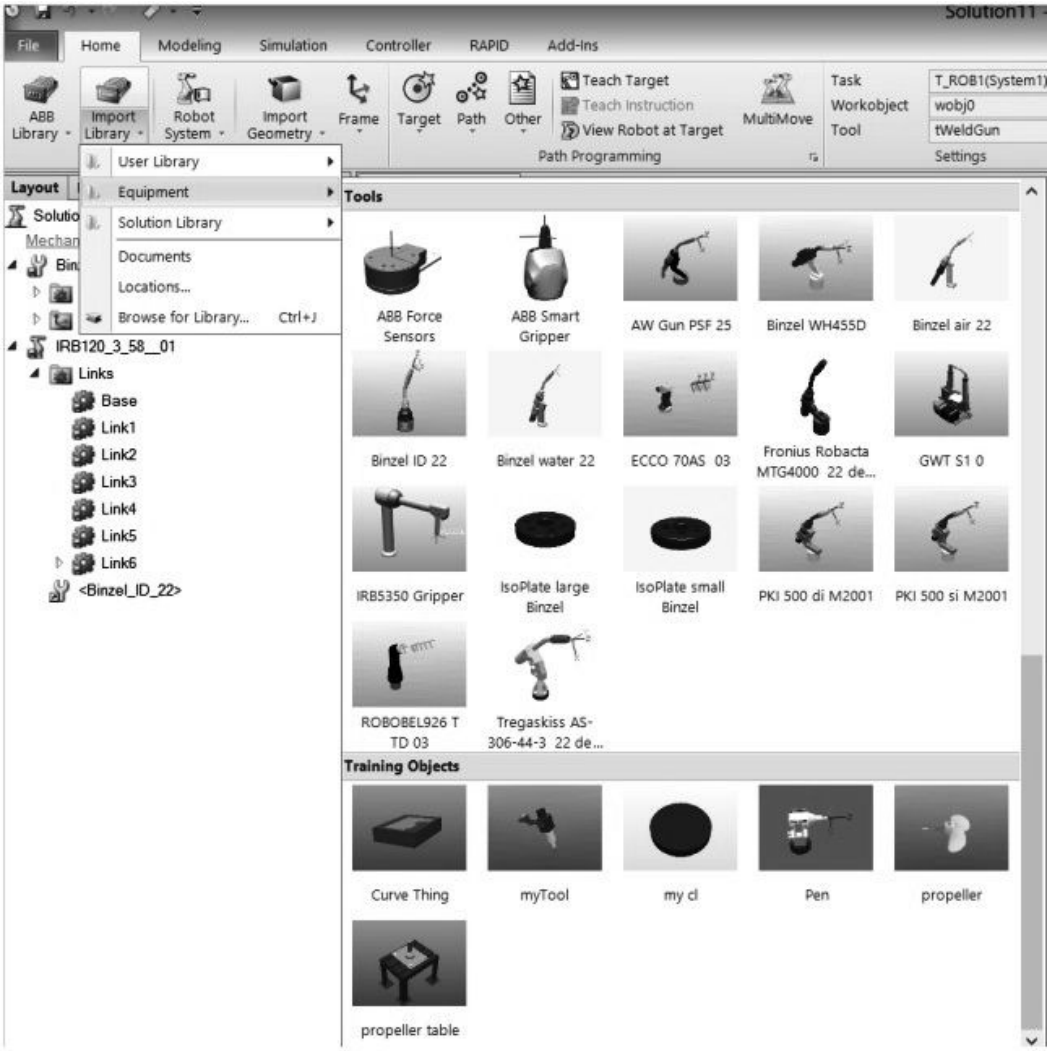
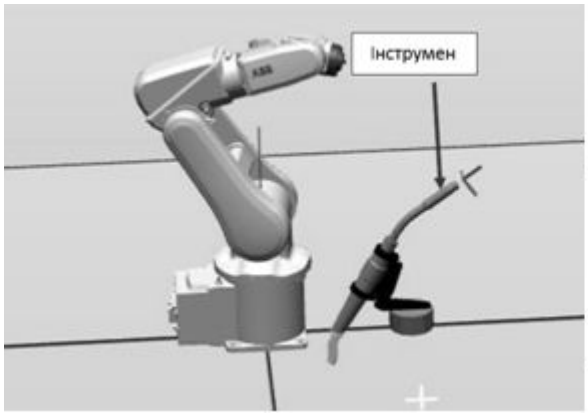


Рис. 9.8.Бібліотека інструментів



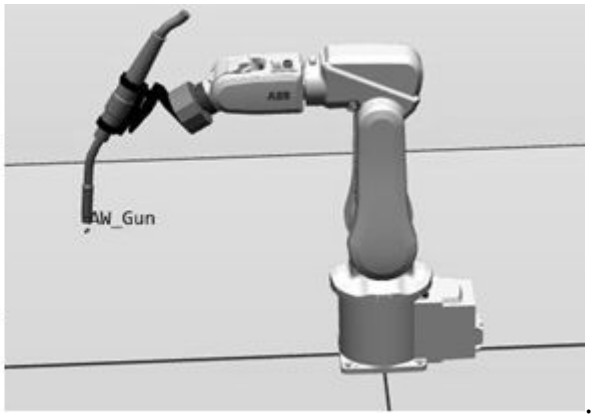


Рис. 9.9. Інструмент в робочій області Рис. 9.10. Додавання інструменту до робота

У робочій області з'явиться платформа з деталлю. Тепер потрібно платформу посунути до робота, таким чином, щоб інструмент діставав до всіх країв деталі. Щоб змінити положення платформи потрібно в розділі «Layout» вибрати «Curve_thing», натиснути правою кнопкою миші і вибрати «Position» -> «Set Position». З'явиться меню зі зміною позиції «Curve_thing», яка здійснює таке: вибір орієнтиру, за яким буде зміняться положення об'єкт, вибір позиції по осях XYZ в міліметрах, орієнтацію об'єкта в градусах.

Приблизний результат буде виглядати приблизно так, як показано на рис. 9.12.

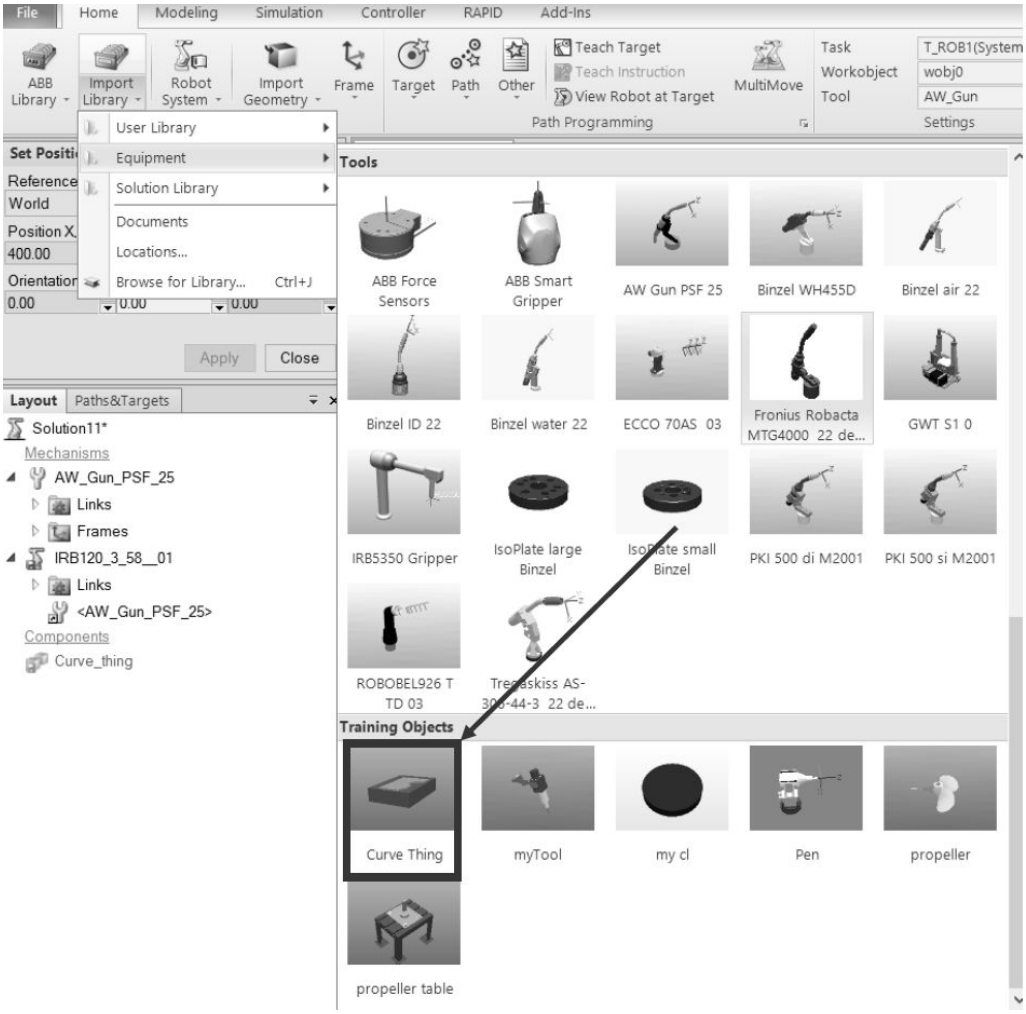


Рис. 9.11. Додавання деталі

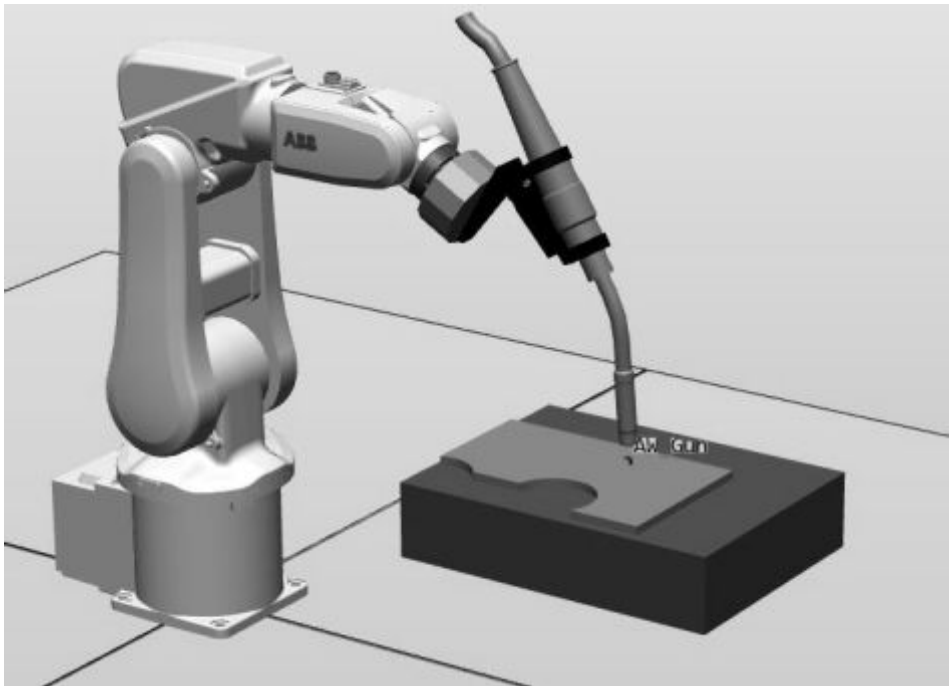


Рис. 9.12. Результат створення моделі

Шлях переміщення можна скласти у ручному режимі. Щоб створити шлях, по якому буде рухатися робот, потрібно в головному меню вибрати «Path» -> «Empty path».

Для того щоб додати точки куди повинен рухатися робот, потрібно перемістити інструмент робота в потрібну позицію за допомогою функцій «Jog». Наприклад, в головному меню в вибираємо «Jog linear» і переміщаємо інструмент в будь-яку точку.

Після того як перемістили інструмент робота потрібно визначити, що це перша точка.

Для цього в головному меню вибираємо «Teach instruction». Після вибору цієї інструкції в розділі «Paths & Procedures» -> «Path_10» з'явиться інструкція «MoveL Target_10 »- це означає, що інструмент буде рухатися в напрямку «Target_10», а саме у визначену точку.

Після створення траєкторії переміщення інструменту його модна протестувати шляхом вибору функції «Move Along Path».

Можна також скласти автоматичний шлях пересування за допомогою «Path» -> «AutoPath». Спочатку потрібно вказати початкову точку шляху. Для цього вибираємо на деталі точку з якої буде почнеться робота робота. Потім потрібно вказати з якої межі буде вестися робота, для цього потрібно в меню «AutoPath» вибрати «Reference Surface» і вибрати грань. Після вибору межі можна виставляти точки, за якими буде йти інструмент.

Програма сама додає додаткові точки огинаючи контур деталі.

Після того, як траєкторія інструменту була створена, потрібно перевірити, дістає чи робот до точок. Щоб це зробити натискаємо правою кнопкою миші на створений шлях («Path_10») в розділі «Paths & Targets» і вибираємо «Reachabillity».

Після створення траєкторії треба здійснити коригування точок, до яких не може підійти інструмент.

Після цього можна запустити симуляцію та перевірити проходження шляху.

Контрольні запитання

1. Визначити,для чого призначена програма в RobotStudio?
2. Назвати,у яких ситуаціях використовується RobotStudio?
3. Описати,які елементи має вікно після створення проекту?
4. Визначити,що здійснюють функції «Move»?
5. Назвати,як додати інструменти у проект?
6. Описати,як додати деталь у проект?
7. Визначити,як створити траєкторію переміщення?
8. Назвати,як скласти траєкторію переміщення у ручному режимі?
9. Описати,як скласти траєкторію переміщення у автоматичному режимі?
10. Визначити,як перевірити проходження шляху?

Змістовий модуль 4. Комп'ютерні засоби проектування мобільних роботів

Лекція 10. Основні принципи проектування колісних мобільних роботів

10.1. Принципи проектування траєкторій переміщення колісних роботів

У мобільних колісних роботах застосовують різні поєднання ведучих, рульових, опорних та ведучих рульових коліс, що дає можливість створити як голономні, так і неголономні мобільні роботи. Як було зазначено вище, неголономні мобільні роботи для зміни напрямку руху потребують здійснити поворот, а голономні - можуть переміщуватися у будь якому напрямку без розвороту [9].

На рис. 10.1 наведені основні типи коліс, які застосовуються у промислових мобільних роботах:

(а) ведуче колесо (один ступень свободи), що обертається навколо колісної осі;

(б) ведуче або опорне поворотне (рульове) колесо (два ступеня свободи), що обертається навколо колісної осі і точкою дотику з поверхнею;

(в) поворотне (флюгерне) колесо (два ступеня свободи), що обертається навколо осі, яка зміщена відносно точки дотику з поверхнею;

(г, д) шведське колесо (три ступені свободи), що обертається навколо (ведучої) колісної осі, навколо осей роликів і навколо точки дотику. Ролики можуть бути розміщені двома різними способами, а саме, повернені на 90 ° (рис. 10.1, г) і 45 ° (рис. 10.1, д);

(е) кульове або сферичне колесо.

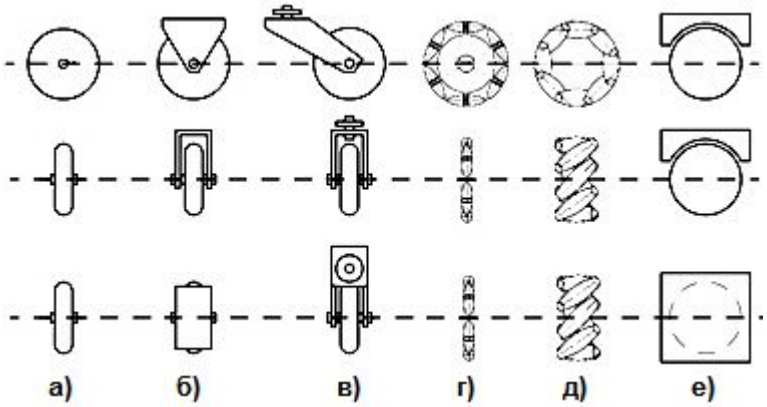


Рис. 10.1. Основні типи коліс, що застосовуються у промислових мобільних роботах

Всі ці типи коліс сильно відрізняються своєю кінематикою і тому значно впливають на всю кінематику мобільного робота.

Ведуче колесо і поворотне колесо мають основну вісь обертання і, таким чином, є строго спрямованими. Для руху в іншому напрямку, колесо має бути спочатку розгорнуто вздовж вертикальної осі. Поворотне колесо, яке обертається навколо зміщеної осі, відрізняється тим, що вихідне зусилля буде рухати шасі під час руління.

Шведське колесо (Swedish wheel, Mecanum wheel) або колесо Ілона було створено шведською компанією Mecanum AB в 1973 році. Конструкція таких коліс дозволяє обертатися на місці при мінімальній силі тертя і низькому обертальному моменті. Ці колеса дають можливість здійснювати рух у будь якому напрямку без повороту колеса (рис. 10.2).

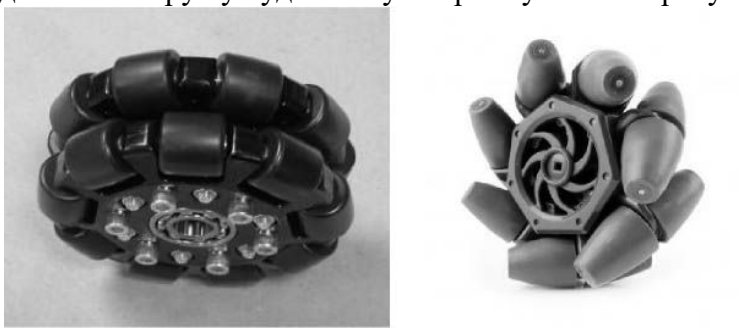


Рис. 10.2. Шведські колеса з роликами, що повернені на 90 ° і 45 °

По-справжньому всеспрямованим колесом, є сферичне колесо, яке сконструйоване таким чином, що воно може активно обертатися в будь-якому напрямку. Одним з механізмів реалізації такої сферичної конструкції є активні приводні ролики, які спираються на верхню поверхню сфери і передають зусилля для обертання. Недоліком таких коліс є складність технічної реалізації (рис. 10.3).

Рис. 10.3. Сферічні колеса

Надалі будемо використовувати позначення коліс, що наведені на рис. 10.4, а саме, поворотний опорний (флюгерний) колісний модуль (рис. 10.4, а), не поворотний опорний колісний модуль (рис. 10.4, б), рульовий колісний модуль (рис. 10.4, в), ведуче колесо (рис. 10.4, г), ведучий рульовий колісний модуль (рис. 10.4, д), всеспрямоване ведуче колесо (рис. 10.4, е).

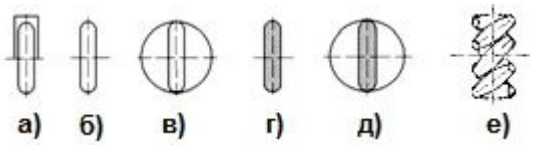


Рис 10.4. Позначення коліс

На рис. 10.5 наведені найбільш поширені кінематичні схеми мобільних роботів, такі як з двома ведучими колесами та рульовим колесом (рис. 10.5, а), з двома неповоротними опорними колесами та ведучим рульовим колесом - трицикл (рис. 10.5, б), з двома диференціальними ведучими колесами та поворотним опорним (флюгерним) колесом (рис. 10.5, в), з чотирма диференціальними ведучими колесами – візок з бортовим розворотом (рис. 10.5, г), з чотирма рульовими ведучими колесами (д), з чотирма всеспрямованими ведучими колесами (рис. 10.5, е). Схеми (а) та (б) здійснюють поворот завдяки рульовому колесу. Схеми (в) та (г) здійснюють поворот завдяки різниці швидкості обертання коліс, що знаходяться по різних сторонах візка. Схеми (д) та (е) здійснюють всеспрямоване переміщення.

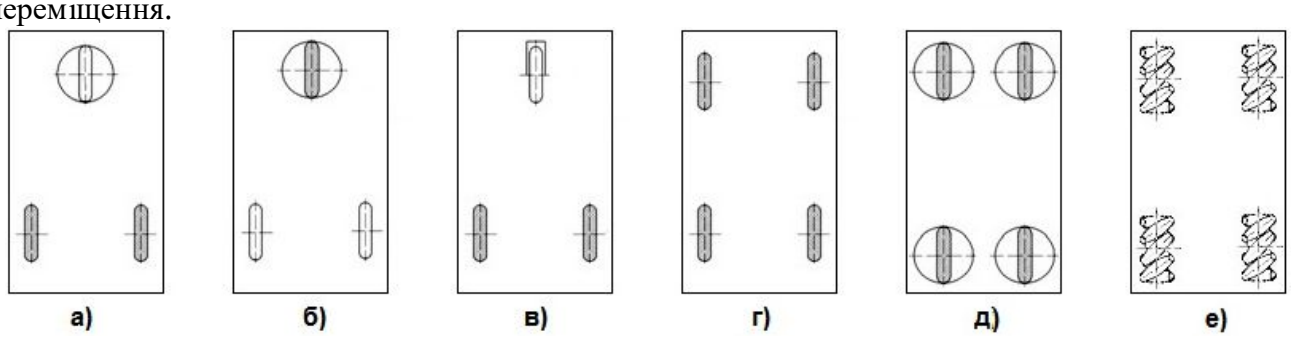
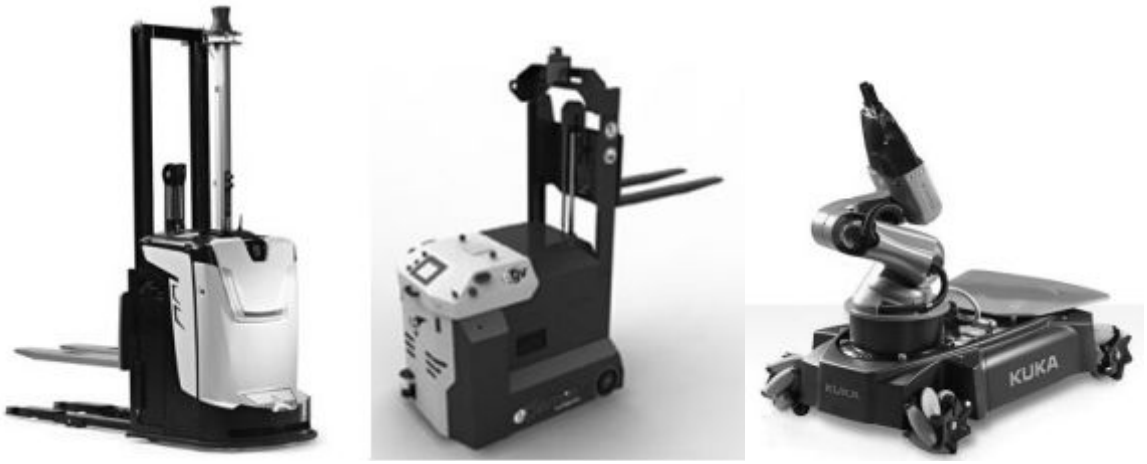


Рис. 10.5. Кінематичні схеми мобільних роботів

На рис. 10.6 наведені приклади колісних мобільних роботів, а саме, навантажувач с приводом типа трицикл (рис. 10.6,а), навантажувач с диференційним приводом (рис. 10.6,в), мобільний робот з чотирма всеспрямованими ведучими колесами (рис. 10.6, с).



а) б) с)

Рис. 10.6. Приклади колісних мобільних роботів

Кінематична модель трициклу (рис. 10.5, б). Такий тип триколісного роботів має два неповоротних опорних колеса та ведуче (приводне) рульове колесо з двома моторами — один для руху, інший для руління.

Переміщення по прямій здійснюється, коли рульове колесо знаходиться у тому ж напрямку, як і опорні колеса (рис. 10.7). Якщо швидкість переміщення робота *V*, то за час Δt відстань, яка буде пройдена приводним колесом, дорівнює $l = V\Delta t$.

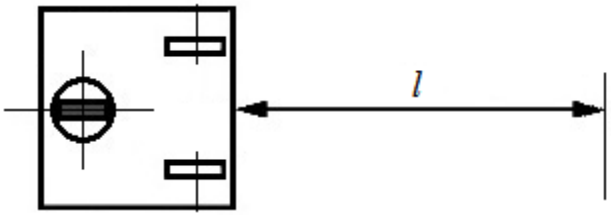


Рис. 10.7. Прямий рух триколісного робота

Розглянемо коловий рух триколісного робота (рис. 10.8).

За умови, що відсутня бічна пробуксовка коліс, пересічемо осі передніх і задніх коліс, щоб сформувати прямокутний трикутник, і в результаті отримаємо:

$$R = \frac{L}{\tan s} \tag{10.1}$$

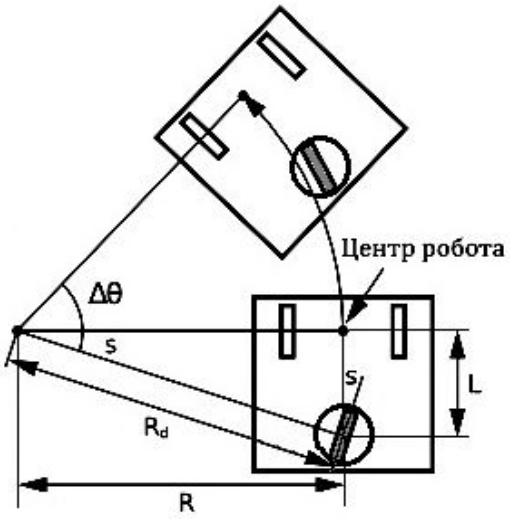


Рис. 10.8. Коловий рух триколісного робота

Визначимо радіус траєкторії, яку описує заднє рульове колесо:

$$R_d = \frac{L}{\sin s} \tag{10.2}$$

За час Δt відстань уздовж цієї дуги кола, яка буде пройдена приводним колесом, дорівнює $l = V\Delta t$, тому кут $\Delta\theta$, на який повернеться робот буде дорівнювати:

$$\Delta\theta = \frac{V\Delta t}{R_d} = \frac{V\Delta t \sin s}{L} = \frac{l \sin s}{L} \qquad (10.3)$$

Для зменшення площі, що потрібна для розвороту робота доцільно здійснювати розворот з мінімальним радіусом. У цьому випадку рульове колесо повернено на 90° (s = π/2), а $R = R_d = L$ (рис. 10.9).

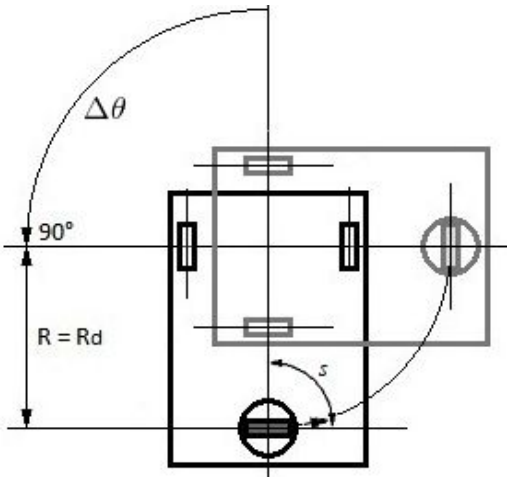


Рис. 10.9. Розворот з мінімальним радіусом

Кут $\Delta\theta$ (в радіанах), на який повернеться робот у цьому випадку буде дорівнювати:

$$\Delta\theta = \frac{V\Delta t}{L} = \frac{l}{L} \qquad (10.4)$$

Таким чином, для повороту на кут $\Delta\theta$ заднє рульове колесо повинно пройти шлях l, що дорівнює $L\ \Delta\theta$.

Для повороту робота на 90° ($\Delta\theta = \pi/2$) заднє рульове колесо повинно пройти шлях l, що дорівнює

$$l = \frac{L\pi}{2} \qquad (10.5)$$

Кінематична модель робота з диференційним приводом має два мотора, по одному на кожне колесо (рис. 10.10). Зміна напрямку руху здійснюється за рахунок різних швидкостей коліс. Крім того на траєкторію переміщення впливає відстань між колесами W.

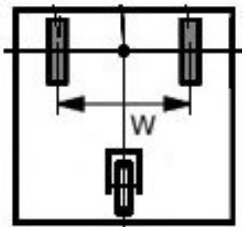


Рис. 10.10. Робот з диференційним приводом

Для прямолінійного руху колеса повинні обертатися з однаковими швидкостями.

Для того, щоб робот розвернувся на місці, необхідно встановити швидкості однаковими по модулю, але спрямованими протилежно.

Інші комбінації швидкостей призводять до руху по дузі (рис. 10.11).

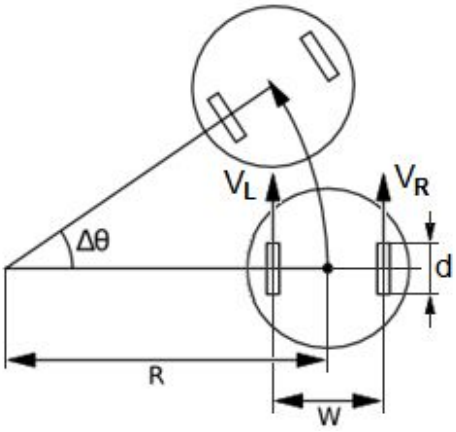


Рис. 10.11. Рух по дузі

Позначимо швидкості коліс (лінійні швидкості з якими вони переміщуються по поверхні) V_L та V_R, відповідно, для лівого і правого коліс і W відстань між колесами.

Прямолінійний рух, якщо $V_L = V_R$.

Розворот на місці, якщо $V_L = -V_R$.

Для того,щоб знайти радіус R переміщення по дузі розглянемо час переміщення Δt, протягом якого робот переміщується вздовж дузі кола на кут $\Delta\theta$.

При цьому:

радіус дузі лівого колеса дорівнює $R - W/2$, тому воно пройде шлях

$$l_L, = V_L\,\Delta t = (R - W/2)\,\Delta\theta; \tag{10.6}$$

радіус дузі лівого колеса дорівнює $R - W/2$, тому воно пройде шлях

$$l_R, = V_R\,\Delta t = (R + W/2)\,\Delta\theta; \tag{10.7}$$

Дуга переміщення робота та обидві колісні дузі мають в основі один і той же кут $\Delta\theta$

$$\Delta\theta = \frac{V_L\Delta t}{R - \frac{W}{2}} = \frac{V_R\Delta t}{R + \frac{W}{2}} \tag{10.8}$$

Тому радіус дузі R, кут повороту $\Delta\theta$, швидкості переміщення лівого та правого коліс V_L , V_R та відстань між колесами W пов'язані такими залежностями

$$R = \frac{W(V_R + V_L)}{2(V_R - V_L)} \tag{10.9}$$

$$\Delta\theta = \frac{(V_R - V_L)\Delta t}{W} = \frac{l_R - l_L}{W} \tag{10.10}$$

Для переміщення по дузі з радіусом R (поворот наліво) швидкості переміщення коліс V_R та V_L пов'язані такою залежністю

$$V_R = V_L\ \frac{2R + W}{2R - W} \tag{10.11}$$

Таким чином, для переміщення по дузі з радіусом R незалежно від швидкості переміщення робота треба забезпечити таке відношення між швидкостями переміщення коліс V_R та V_L

$$\frac{V_R}{V_L} = \frac{2R + W}{2R - W} \tag{10.12}$$

10.2. Проектування приводів колісних роботів

Для колісних та гусеничних рушіїв мобільних роботів найчастіше використовують регульовані електроприводи, які реалізуються на основі двигунів постійного струму, крокових двигунів та двигунів змінного струму. При цьому треба мати на увазі, що на мобільних роботах найчастіше встановлюють джерела постійного струму (акумулятори, іоністори або сонячні батареї) з напругою 12 В та 24 В.

Використання двигунів постійного струму в мобільних роботах пояснюється такими якостями, як високий пусковий, гальмівний та перевантажувальний моменти, порівняно

висока швидкодія, що важливо при реверсуванні і гальмуванні, можливість широкого і плавного регулювання частоти обертання шляхом зміни напруги живлення.

Для вибору електродвигуна постійного струму для колісного робота необхідні такі вихідні дані: маса робота, крутний момент, швидкість, потужність.

Також необхідно підібрати діаметр коліс і визначити правильне передавальне число зубчастої передачі для розрахунку швидкості його руху.

Для того, щоб робот міг рухатися, необхідно, щоб крутний момент двигуна перевищував вагу робота (який виражається у Н / м).

Маса робота дорівнює *m (кг)*, максимальна швидкість його переміщення *v (м/с)* при радіусі колеса рівному *r (м)*.

При переміщенні по прямій на відстань *d* розрахуємо прискорення, необхідне для досягнення швидкості *v*.

Виходячи з того, що при розгоні кінцева швидкість визначається як

$$v^2 = v_0^2 + 2ad \qquad (10.13)$$

де, *d*— відстань, яку пройшов робот, v_0 —його початкова швидкість (при старті з місця, $v_0 = 0$), *v* — швидкість робота, *a* — прискорення,

отримаємо

$$a = \frac{v^2 + v_0^2}{2d}, \quad \text{або} \quad a = \frac{v^2}{2d} \quad \text{при } v_0 = 0 \qquad (10.14$$

Крутний момент, який необхідний для переміщення робота та отримання ним прискорення, необхідного для досягнення максимальної швидкості розраховується наступним чином:

$$M = J\varepsilon. \qquad (10.15)$$

При рівномірному русі прискорення візка мобільного робота *a* і кутове прискорення колеса ε дорівнюють нулю. Крутний момент в цьому випадку строює тягове зусилля $F_{тяги} = M_k / R$ для подолання зусилля супротив руху візка. Але при прискоренні $a > 0$, зокрема розгону мобільного робота масою m_p, необхідно забезпечити кутовим прискоренням $\varepsilon = a/R$ колеса з моментом інерції J_k. Тому для розрахунку моменту необхідно врахувати і рух візка і забезпечення прискорення коліс.

Використовуючи принцип суперпозиції (рух колеса це сума двох рухів: прямолінійного разом із візком, та обертання навколо своєї осі), отримаємо.

Загальна формула крутного моменту:

$$M_k = n_{прив} F_{тяги} R + n J_k \varepsilon \qquad (10.16)$$

або

$$M_k = n_{прив} m_p a R + n m_k R a \qquad (10.17)$$

де $n_{прив}$ – кількість приводних (ведучих) коліс МР,

n– загальна кількість коліс;

R– радіус колеса, для обраних коліс;

m_k – маса одного колеса, для обраних коліс.

Потужність двигуна $P_{дв}$ з урахуванням коефіцієнта корисної дії η та залежності між кутовою швидкістю ω та швидкістю переміщення *v* ($\omega = v / R$) дорівнює:

$$P_{дв} = \frac{M_k w}{\eta} = \frac{M_k v}{\eta R} \qquad (10.18)$$

Регулювання швидкістю обертання здійснюється шляхом зміни напруги живлення. Сучасні системи використовують для цього широтно-імпульсну модуляцію (ШІМ). Вигляд сигналу з широтно-імпульсною модуляцією показаний на рис. 8.2.

Усереднена напруга $V_{ср}$ такого сигналу залежить від напруги живлення $V_{сс}$ та коефіцієнта заповнення *D*, що визначається як відношення тривалості імпульсу τ до періоду повторення імпульсів *T*, я саме,

$$V_{ср} = V_{сс} \tau/T = D. \qquad (10.19)$$

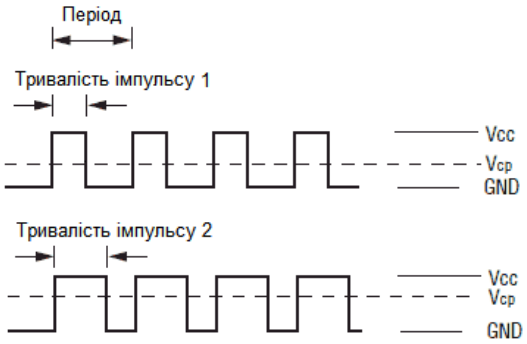


Рис. 10.12. Вигляд сигналу з широтно-імпульсною модуляцією

Цей засіб використовується для формування сигналів довільної форми шляхом зміни тривалості імпульсів по заданому закону. Перевагою такого засобу формування сигналу є те, що комутаційний елемент, за допомогою якого формується сигнал з широтно-імпульсною модуляцією, може перебувати в двох станах – повністю відкритому або повністю закритому. Тому втрати електроенергії визначаються потужністю, що виділяється на повністю відкритому комутаційному елементі.

Завдяки цьому формування сигналів з широтно-імпульсною модуляцією є стандартною функцією багатьох систем керування, що використовуються у складі мобільних роботів (приклади таких систем будуть розглянуті далі), і може використовуватись для регулювання швидкості обертання двигунів постійного струму.

Для зміни напрямку обертання (реверсу двигуна), що здійснюється шляхом зміни полярності напруги живлення, використовують різні комутаційні схеми, наприклад, таку, що наведена на рис.10.13.

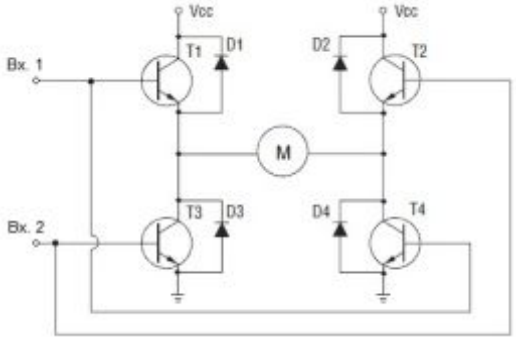
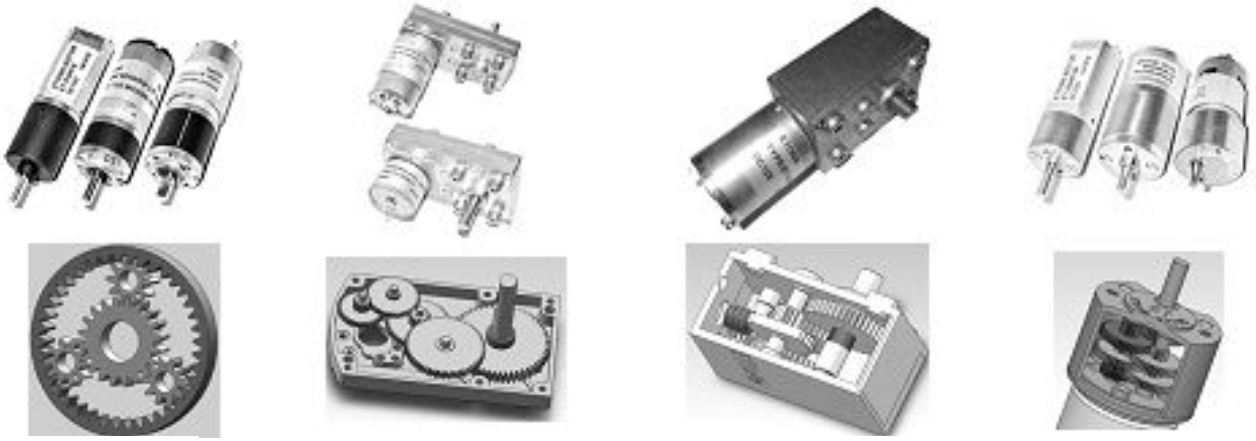


Рис. 10.13. Схема реверсу двигуна

Коли керуючий вхідний сигнал подається на транзистори T1 та T4 (Вх. 1), двигун обертається в одну сторону, а коли на транзистори T2 та T3 (Вх. 2), двигун обертається в іншу сторону. Захисні діоди D1, D2, D3, D4 призначені для замикання струму двигуна при відключенні відповідних транзисторів, що виникає завдяки індуктивному навантаженню, яку представляє собою обмотка електродвигуна.

Оскільки двигуни постійного струму мають велику швидкість обертання та малий крутний момент, використовують різні редуктори, які часто є складною частиною двигуна (рис. 10.14).

Передаточне число таких мотор-редукторів може знаходитись у досить великих межах, що дає можливість вибрати двигуни з потребною швидкістю обртання та крутним моментом.



планетарний мотор-редуктор | плоский циліндричний мотор-редуктор | черв'ячний мотор-редуктор | черв'ячний мотор-редуктор

Рис. 10\14. Двигуни з редукторами

Наприклад мотор-редуктори постійного струму IG-32RGM з реверсивним колекторним двигуном потужністю 8 Вт та редуктором з планетарною та конічною передачею (рис. 10.15) мають такі характеристики (табл. 10.1). За потребою на двигуни встановлюють електромагнітне гальмо та датчик кута обертання на основі датчика Хола чи оптичного растрового датчика, що дозволяє вирішувати задачі одометрії.

Рис. 10.15. Мотор-редуктор постійного струму IG-32RGM

Табл. 10.1.

Характеристики мотор-редукторів постійного струму IG-32RGM

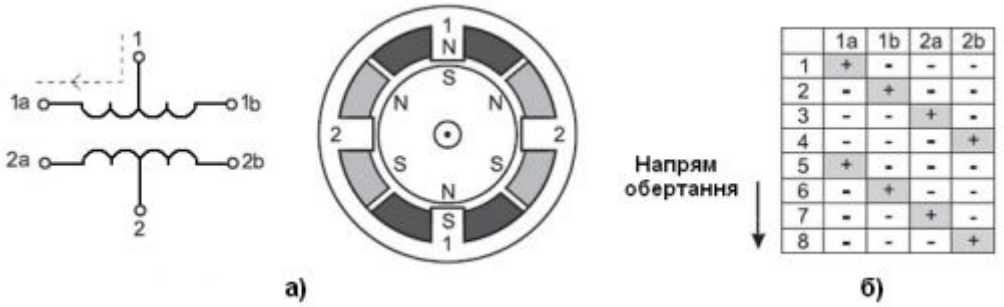
Передат. число	1/5	1/14	1/19	1/27	1/35	1/51	1/71	1/100	1/139	1/189	1/264	1/516	1/721	1/939
Живлення 12В (03 Тип)														
Момент, кгс*см	0.4	0.9	1.2	1.7	2.3	2.8	3.9	5.4	7.6	8.3	11.6	12	12	12
Швидкість, об/хв.	1140	430	310	220	170	116	83	60	43	31.5	23.5	13	9.6	7.2
Живлення 24В (04 Тип)														
Момент, кгс*см	0.47	1.1	1.5	2.1	2.7	3.3	4.6	6.4	9	9.8	12	12	12	12
Швидкість, об/хв.	1170	445	320	229	176	120	87	62	44.5	34	25	13	9.8	7.2

Фірми, що випускають обладнання для мобільних роботів, мають зачастую сервоприводи постійного струму з умонтованими системами керування швидкістю обертання та визначення куту обертання (засобів одометрії).

Високу точність позиціонування без датчиків зворотного зв’язку (швидкості або положення) можна здійснити за допомогою крокових двигунів.

Принцип роботи крокового двигуна заснований на використанні такої конструкції, при якої один вхідний імпульс повертає ротор на визначений кут. Імпульси поступають послідовно на різні обмотки, що забезпечує обертання з постійною швидкістю.

Спрощена схема уніполярного крокового двигуна наведена на рис. 10.16 а. На рис. 10.16 б показана послідовність подачі сигналів, яка потребується для обертання двигуна.



	1a	1b	2a	2b
1	+	-	-	-
2	-	+	-	-
3	-	-	+	-
4	-	-	-	+
5	+	-	-	-
6	-	+	-	-
7	-	-	+	-
8	-	-	-	+

Рис. 10.16. Уніполярний кроковий двигун

Мікроконтролери та програмовані логічні контролери (ПЛК), що використовують у системах керування мобільними роботами, наприклад, мікроконтролери Arduino та ПЛК S7-200 та S7-1200 фірми Siemens, мають у складі програмного забезпечення функції керування кроковими двигунами. Контролер формує послідовність імпульсів зі змінним періодом повторення, що дозволяє керувати швидкістю обертання двигуна. Оскільки контролер видає послідовність імпульсів на один вихід потрібна схема, яка сформує послідовну видачу імпульсів на контакти 1а, 1б, 2а, 2б. Ці функції виконує силовий перетворювач, який забезпечує також потрібну напругу та струм на виході (рис. 10.17). Діоди на виходах перетворювача потрібні для замикання струму котушок двигуна при відключенні.

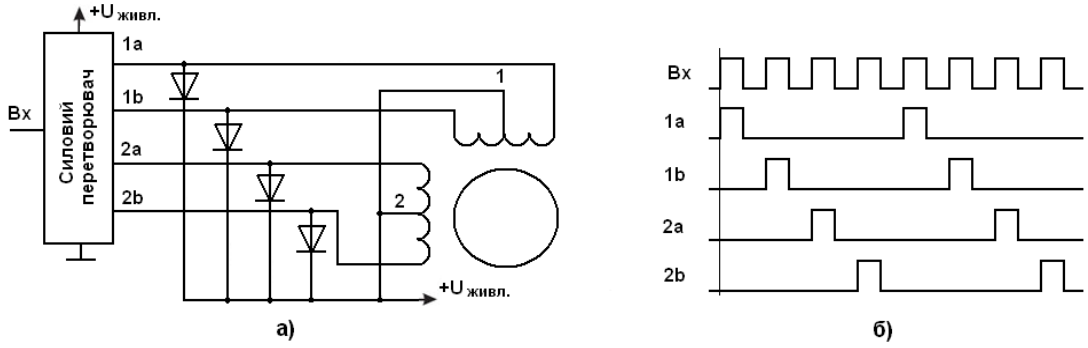


Рис.10.17. Силовий перетворювач для крокового двигуна

Змінюючи період проходження імпульсів можна змінювати швидкість обертання крокового двигуна по заданому закону. Позицію задає кількість імпульсів, які поступають на кроковий двигун.

У складі мікроконтролерів Arduino є драйвери крокових двигунів, які забезпечують подачу необхідної послідовності імпульсів біполярні та уніполярні крокові двигуни.

Швидке зростання кількості модифікацій мобільних роботів привело до появлення універсальних модулів, на основі котрих будуються сучасні мобільні роботи.

Так модуль повороту та переміщення Omni-Drive-Module Neobotix є інтегрований модуль, який включає в себе ведуче колесо, яке може додатково обертати і орієнтувати навколо своєї вертикальної осі. Транспортний засіб з такими модулями може бути всеспрямованим, але при цьому колеса повинні повертатися синхронно (рис. 10.18).

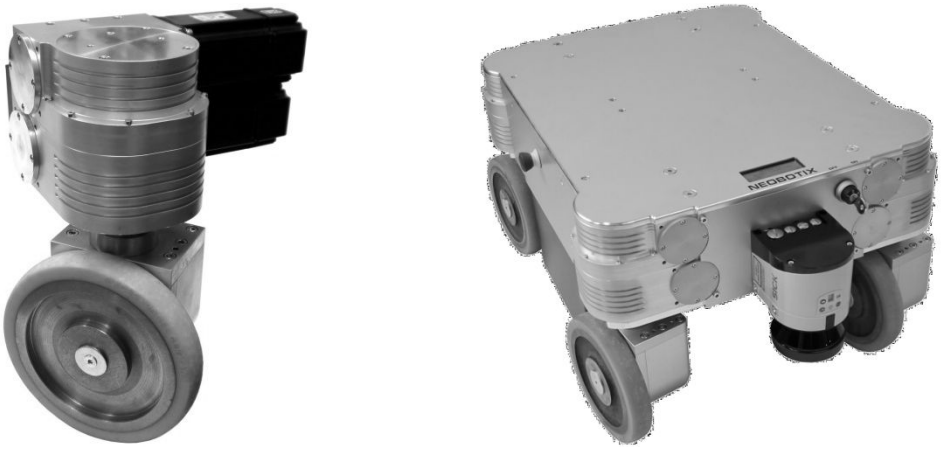


Рис. 10.18. Двовісний модуль повороту та переміщення Omni-Drive-Module Neobotix та мобільний робот MPO-700

Двовісні модулі вже використовуються на мобільних роботах, таких як, наприклад, Omni-directional robot MPO-700.

10.3. Проектування систем керування колісних роботів

Рух і стан робота для переміщення по поверхні.

Для того щоб математично описати рух мобільного робота треба визначити системи координат. Введемо дві системи координат - світову систему координат W (яка нерухома в просторі), і система координат робота R, яка переміщається в просторі і залишається нерухомою щодо самого робота (рис. 10.19).

Рис. 10.19. Системи координат - світова система координат W (яка нерухома в просторі), і система координат робота R

Необхідно визначити місце розташування робота, тобто перетворення координат між W і R.

Якщо припустити, що робот обмежується переміщенням на поверхні, його місце розташування може бути визначене вектором стану, який складається з трьох параметрів:

$$X = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} \tag{10.20}$$

x і y визначають місце розташування зумовленою точки «центру робота» у світовій системі координат, θ визначає кут повороту між системами координат (кут між осями x^W та y^R)

Дві системи координат збігаються в момент, коли центр робота знаходиться на початку координат і $x = y = \theta = 0$.

Інтегральний рух на плоскості

Отримуючи переміщення робота в деякі моменти часу, ми можемо знайти весь шлях, пройдений роботом, підсумувавши ці значення, або перейшовши до межі (при прагненні кількості вимірів $\to\infty$) - шляхом їх інтегрування.

При русі на плоскості ми маємо три ступені свободи для визначення положення, що представлені як (x, y, θ).

Розглянемо робота, який може тільки рухатися вперед або повертатися на місці.

При прямолінійному русі робота на відстань D новий стан буде виражено як:

$$\begin{pmatrix} x_{new} \\ y_{new} \\ \theta_{new} \end{pmatrix} = \begin{pmatrix} x + D\cos\theta \\ y + D\sin\theta \\ \theta \end{pmatrix} \quad (10.21)$$

Якщо присутній тільки обертальний рух, при повороті на кут α:

$$\begin{pmatrix} x_{new} \\ y_{new} \\ \theta_{new} \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta + \alpha \end{pmatrix} \quad (10.22)$$

Оцінка кругового руху (рис. 10.20)

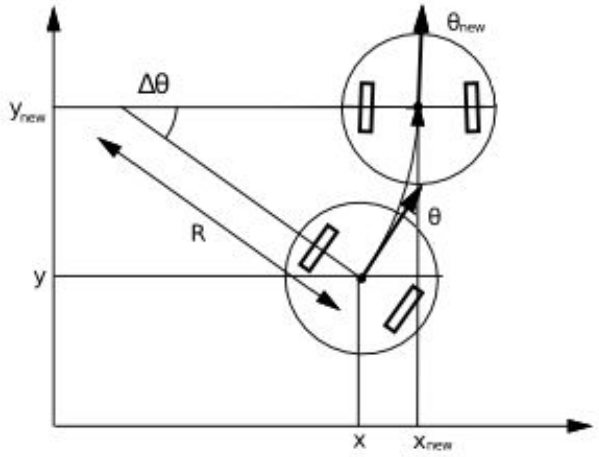


Рис. 10.20. Переміщення по колу (круговий рух)

Для випадків і диференціального й триколісного роботів ми можемо отримати вираження для R і $\Delta\theta$ у випадку, коли присутня тільки рух по дузі кола:

$$\begin{pmatrix} x_{new} \\ y_{new} \\ \theta_{new} \end{pmatrix} = \begin{pmatrix} x + R(\sin(\Delta\theta + \theta) - \sin\theta) \\ y - R(\cos(\Delta\theta + \theta) - \cos\theta) \\ \theta + \Delta\theta \end{pmatrix} \quad (10.23)$$

Планування маршруту, виходячи з положення (рис. 10.21).

Якщо припустити, що роботу відоме місце розташування, і як воно відноситься до світової системи координат, то планування маршруту на основі його місця розташування дозволить йому рухатися по точному шляху вздовж послідовності заздалегідь визначених точок.

Різні криволінійні траєкторії можуть бути сплановані, з оптимізацією таких критеріїв, як час руху за маршрутом або споживання енергії.

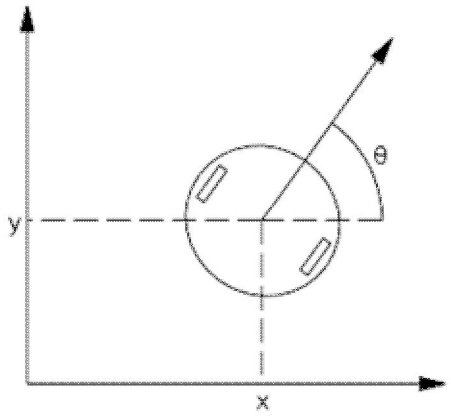


Рис. 10.21. Планування маршруту, виходячи з положення

Припустимо, що:

- рух робота складається з прямолінійних відрізків окремо від розворотів на місці;
- робот прагне звести до мінімуму загальне пройдену відстань, так що він завжди відразу повертається обличчям до наступної точки і їде прямо до неї.

На першому кроці планування маршруту, припустимо, що поточне положення робота (x, y, θ) і наступної точки маршруту є (W_x, W_y).

Спочатку робот повинен обернутися до зазначеній точці. Вектор спрямування повинен вказувати на:

$$\begin{pmatrix} d_x \\ d_y \end{pmatrix} = \begin{pmatrix} W_x - x \\ W_y - y \end{pmatrix} \tag{10.24}$$

Значення абсолютного значення кута в градусах α, на який робот повинен обернутися:

$$\alpha = \tan^{-1} \frac{d_y}{d_x} \tag{10.25}$$

Оскільки робот вже повернутий на кут θ, тому кут, на який він повинен обернутися, дорівнює $\beta = \alpha - \theta$.

Після цього, робот повинен рухатися по прямій на відстань

$$d = \sqrt{d_x^2 + d_y^2} \tag{10.26}$$

Контрольні запитання

1. Визначити,які основні типи коліс, що застосовуються у промислових мобільних роботах?
2. Назвати,які кінематичні схеми мобільних роботів здійснюють всеспрямоване переміщення?
3. Розповісти,як описати прямолінійний рух триколісного робота?
4. Визначити,як описати коловий рух триколісного робота?
5. як описати розворот з мінімальним радіусом для триколісного робота?
6. Назвати,як описати прямолінійний рух робота з диференційним приводом?
7. Розповісти,як описати коловий рух робота з диференційним приводом?
8. як описати розворот на місці робота з диференційним приводом?
9. Визначити,які системи координат використовують для математичних моделей мобільного робота?
10. Назвати,як здійснити планування маршруту переміщення робота, виходячи з положення?

Лекція 11. Основні принципи проектування гусеничних мобільних роботів

11.1. Принципи проектування траєкторій переміщення гусеничних роботів

Процес повороту гусеничного транспортного засобу за характером взаємодії рушія з опорною поверхнею принципово відрізняється від процесу повороту колісного транспортного засобу [9].

Конструкція ходової частини гусеничного транспортного засобу виключає можливість його кінематичного повороту за рахунок одного лише перекочування опорних катків по гусениці. Опорні гілки гусениць, навантажені вагою транспортного засобу, отримують при повороті бічне переміщення по поверхні, долаючи вельми значні додаткові опору від сил тертя гусениць по поверхні і сил опору зрізу і загрібання грунту гусеницями. У результаті різко зростають опір руху гусеничного транспортного засобу при повороті і навантаження на його двигун.

Поворот гусеничного транспортного засобу здійснюється шляхом зміни швидкості перемотування гусениць. Поворот відбувається при відключенні від трансмісії тієї гусениці, в бік якої треба повернути транспортний засіб. Якщо треба зробити крутий поворот, відключену гусеницю пригальмовують і транспортний засіб повертається на місці.

Позначимо гусеницю з меншою швидкістю, гусениця, що відстає і надамо їй індекс 1, а гусеницю з більшою швидкістю - гусениця, що забігає і надамо їй індекс 2 (рис. 11.1).

Транспортний засіб повертається щодо миттєвого центру повороту *O*. Центр повороту завжди лежить на лінії, перпендикулярній до поздовжньої площини транспортного засобу. При повороті транспортного засобу опорна гілка гусениці утворює з поверхнею неплоску фрикційну пару. Для спрощення явищ, що відбуваються при повороті, будемо вважати цю пару плоскою.

Гусениці при повороті пробуксовують або прослизають відносно поверхні. Тому на кожній площині фрикційної пари існує єдина точка, в якій відсутній ковзання або буксування. Положення кожної такої точки називається полюсом обертання гусениці.

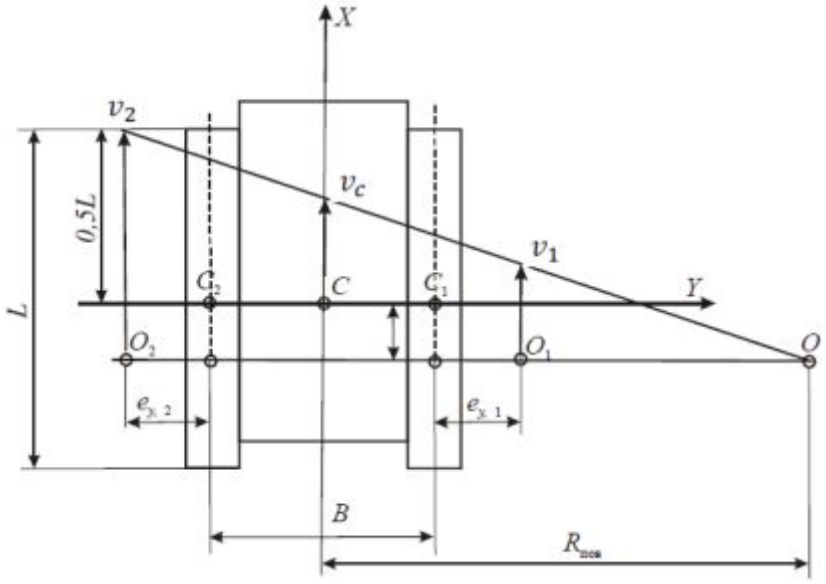


Рис. 11.1. Схема повороту гусеничного транспортного засобу

Полюси обертання визначають кінематичний зв'язок між поверхнями, що труться, і діючими на них силами. При буксуванні і ковзанні гусениць відносно поверхні полюси обертання O_1 і O_2 (рис. 1) не збігаються з геометричними центрами гусениць C_1 і C_2, а зміщуються на деякі відстані. Проекції цих відстаней на координатні осі X і Y називаються **ексцентриситетами полюсів обертання** (e_x, $e_{y,1}$, $e_{y,2}$). Вісь X системи координат збігається з напрямком поздовжньої осі симетрії машини, а вісь Y - перпендикулярна поздовжній осі транспортного засобу.

Точки O, O_1 і O_2 завжди лежать на одній прямій, перпендикулярній до поздовжньої площини транспортного засобу. Лінія, що з'єднує ці точки, називається **лінією центрів повороту**. У загальному випадку вона з'єднує центри тиску гусениць на поверхню і не збігається з поперечною віссю C - X, що проходить через середини опорних поверхонь гусениць.

Поворот гусеничного транспортного засобу характеризується також кутовою швидкістю повороту остова транспортного засобу $\omega_т$ і радіусом повороту $R_{пов}$. Радіус повороту $R_{пов}$ дорівнює відстані від центру повороту O до поздовжньої площини симетрії транспортного засобу. Кутову швидкість повороту транспортного засобу можна визначити формулою:

$$\omega_т = \frac{v_2 - v_1}{e_{y,2} + B + e_{y,1}}. \qquad (11.1)$$

Довжина опорної гілки, що дорівнює відстані L між осями крайніх опорних катків, називається поздовжньою базою гусеничного транспортного засобу. Для спрощення аналізу явищ, що відбуваються при повороті, можна припустити, що поздовжня база транспортного засобу дорівнює довжині опорної поверхні гусениць.

У загальному випадку руху транспортного засобу по площині, що деформується, опорна гілка гусениці завжди більше поздовжньої бази транспортного засобу.

Відношення поздовжньої бази до поперечної базі B називається відносною опорною базою, яка є важливим геометричним параметром, що характеризує поворот гусеничного транспортного засобу.

Розглянемо спочатку спрощену схему повороту гусеничного транспортного засобу без буксування і ковзання гусениць. У цьому випадку миттєвий центр повороту O буде розташовуватися на поперечній осі, що проходить через центр мас (рис. 11.2).

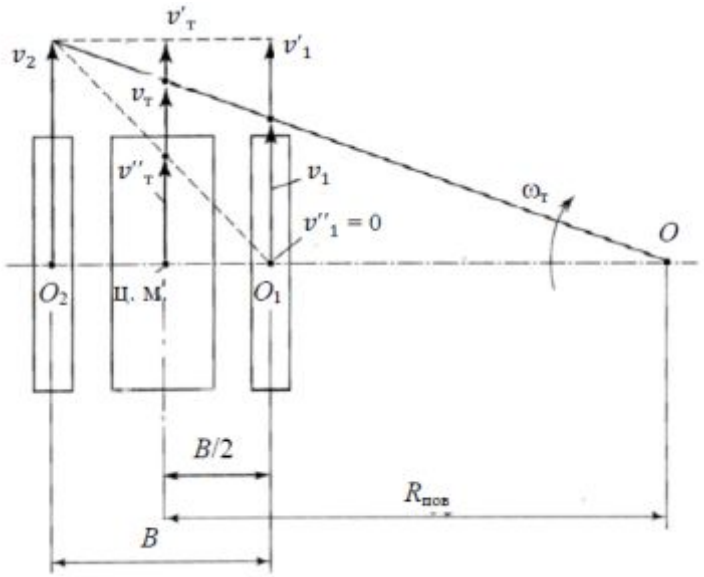


Рис. 11.2. План швидкостей при повороті гусеничного транспортного засобу з радіусом $R_{пов} > B/2$

При прямолінійному русі транспортного засобу швидкість його центру мас дорівнює $v'_т$. З такою ж швидкістю рухаються права і ліва гусениці ($v_2 = v'_1 = v'_т$). Для здійснення правого повороту швидкість правої гусениці необхідно зменшити до значення v_1. Швидкість лівої гусениці залишимо без змін. З'єднуючи кінці векторів швидкостей v_2 и v_1 прямою лінією на перетині її з поперечною віссю транспортного засобу $O_2 - O_1$ знаходимо миттєвий центр повороту O.

У точці O поступальна швидкість трактора дорівнює нулю. Щодо цієї точки трактор робить поворот по дузі кола з радіусом $R_{пов.}$ Таким чином, рух гусениць транспортного засобу на повороті складається з двох рухів:

поступального руху зі швидкостями v_2 і v_1, відповідно, гусениць, що забігають і відстають;

обертального руху цих гусениць навколо полюсів повороту O_2 і O_1 з кутовою швидкістю $\omega_т$.

З подоби трикутників на плані швидкостей (рис. 4.2) знаходимо:

$$\frac{v_т}{R_{пов}} = \frac{v_2 - v_1}{B}. \qquad (11.2)$$

Оскільки швидкість центру мас транспортного засобу $v_т = (v_2 + v_1)/2$, то після її підстановки в формулу (4.2), отримаємо

$$R_{пов} = \frac{B(v_2 + v_1)}{2(v_2 - v_1)}. \qquad (11.3)$$

Розглянемо кінематику повороту при буксуванні гусениць.

Радіус повороту транспортного засобу $R_{пов}$, що визначається по формулі (114.3) , враховує лише теоретичні швидкості гусениць, що забігають v_2 і відстають v_1 (рис. 11.3).

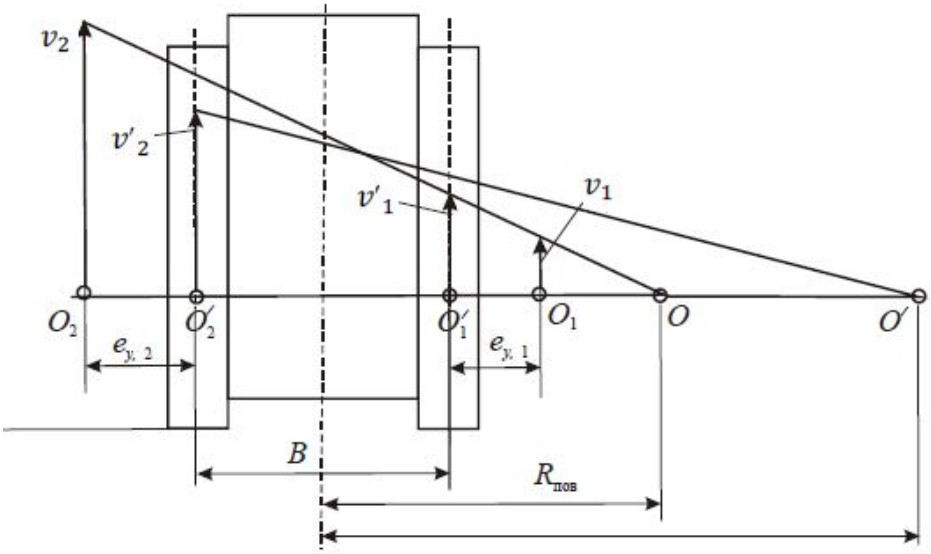


Рис. 11.3. Схема визначення радіуса повороту гусеничного транспортного засобу з урахуванням буксування гусениць

З'єднуючи кінці векторів швидкостей v_2 і v_1 прямою лінією, знаходимо миттєвий центр повороту транспортного засобу (точку O) на перетині цієї прямої з лінією центрів повороту $O_2 - O_1$.

Дійсні швидкості руху гусениць відрізняються від теоретичних внаслідок буксування і ковзання гусениць при русі транспортного засобу. Забігати гусениця, як правило, рухається з буксуванням, тому її реальна швидкість $v'_2 = v_2 (1 - \delta_2)$, де δ_2 – її коефіцієнт буксування.

При гальмуванні ведучого колеса відстаючої гусениці вона зазвичай починає прослизати щодо опорної поверхні, тому її швидкість зростає: $v'_1 = v_1 (1 + s)$, де s – коефіцієнт ковзання відстаючої гусениці.

Проводячи пряму лінію через кінці векторів реальних швидкостей v'_2 і v'_1 (рис. 4.5), отримуємо нове положення миттєвого центру повороту (точку O'). На рис. 4.5 видно, що буксування і проковзування гусениць призводить до збільшення радіусу повороту. Величину реального радіуса повороту $R_{пов} = R'_{пов}$ можна оцінити, замінивши у формулі (3) теоретичні швидкості v_2 і v_1 їх реальними значеннями можна $v'_2 = v_2 (1 - \delta_2)$ і $v'_1 = v_1 (1 + s)$:

$$R_{пов} = R'_{пов} = \frac{B}{2}\frac{v'_2 + v'_1}{v'_2 - v'_1} = \frac{B}{2}\frac{v_2 + v_1 - (v_2\delta_2 - v_1 s)}{v_2 - v_1 - (v_2\delta_2 + v_1 s)}. \qquad (11.4)$$

Оскільки ($v_2\,\delta_2 + v_1\,s$) > ($v_2\,\delta_2 - v_1\,s$), то порівняння формул (3) і (4) показує, що $R_{пов} < R'_{пов}$.

У реальних умовах експлуатації можливі випадки руху транспортного засобу, коли відстаюча гусениця також рухається з буксуванням.

Тоді її реальна швидкість стає менше теоретичної: $v'_1 = v_1\,(1 - \delta_1)$, де δ_1 – коефіцієнт буксування відстаючої гусениці.

Вводячи в формулі (4) заміну: $\delta_1 = -\,s$, отримаємо:

$$R_{пов} = R''_{пов} = \frac{B}{2}\,\frac{v_2 + v_1 - (v_2\delta_2 + v_1\delta_1)}{v_2 - v_1 - (v_2\delta_2 - v_1\delta_1)}. \qquad (11.5)$$

Оскільки ($v_2\,\delta_2 + v_1\,\delta_1$) > ($v_2\,\delta_2 - v_1\,\delta_1$), то $R''_{пов} < R_{пов}$. Отже, буксування відстаючої гусениці призводить до зменшення радіуса повороту транспортного засобу.

11.2. Проектування приводів гусеничних роботів

Мобільний робот з гусеничним рушієм переміщається за рахунок сил тертя, що виникають між опорною поверхнею (ґрунтом) і гусеничним рушієм робота. Гусеничний рушій складається з ведучого (тягового) колеса, відомих (опорних та підтримуючих) котків, гусениці та корпусу (рис. 11.4). Ведучий коток, з'єднаний за допомогою редуктора з двигуном, створює тягове зусилля за рахунок перемотування гусениць. Велика площа дотику гусениць з ґрунтом дозволяє забезпечити низький середній тиск на ґрунт, завдяки чому досить відчутно зменшується занурення гусеничного рушія в ґрунт.

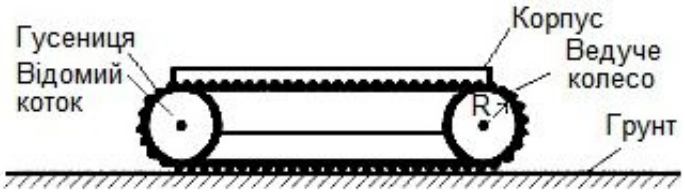


Рис. 11.4. Гусеничний рушій

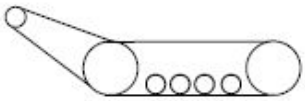
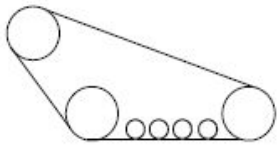
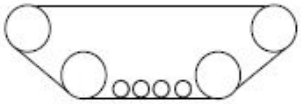
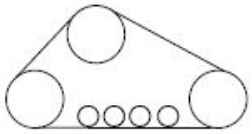
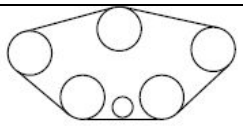
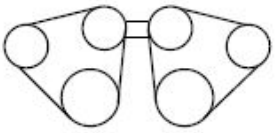
На рис. 11.5 наведені приклади гусеничних мобільних роботів.

Рис. 11.5. Приклади гусеничних мобільних роботів

Залежно від конфігурації гусеничного рушія можна також виділити кілька основних типів (табл. 11.1).

Найбільш типовими для гусеничних мобільних роботів є конструкції 1 і 2. Підвіски 4 і 6 застосовуються у випадках, коли необхідно забезпечити високу вантажопідйомність. Конструкції 3 і 5 відрізняються високою надійністю. Робот, побудовані за схемою 7 призначені для рішення особливих завдань, таких, як наприклад рух по замкнутому профілю труби, або подолання значних перешкод.

Таблиця 11.1. Різні конфігурації рушіїв гусеничних систем.

	Назва	**Опис**	**Схема**
1	Проста	Містить два опорних катка, один або два з яких є провідними	
2	Двосекційна	Містить два опорні катка, один або два з яких є провідними і один додатковий каток, який може міняти положення щодо двох інших	
3	З одним натяжним катком, винесеним вперед	Містить два опорних катка та один натяжний каток, розташований в передній частині шасі	
4	З двома натяжними катками	Містить два опорних катка та два натяжних катка	
5	З одним натяжним катком (зворотна)	Містить два опорних катка та один натяжний каток, розташований в середній частині шасі	
6	З кількома опорними і натяжними катками	Має кілька опорних і кілька натяжних катків	
7	Зі складною структурою	Підвіска складається з декількох гусениць, спеціальним чином розташованих у просторі, що забезпечує можливість руху по поверхнях незвичайної конфігурації (круглі, нерівні поверхні)	

На рис. 11.6 наведено порівняння гусеничного та колісних роботів.

Рух гусеничного мобільного робота (рис. 11.6, г) схожий на рух колісних роботів з диференціальним приводом (рис. 11.6, а) та роботів з бортовим розворотом (чотирьохколісний та шестиколісний приводи), коли здійснюється регулювання швидкості одночасно для усіх коліс з одного боку (рис. 11.6, б, в).

Різниця полягає в тому, що для гусеничного та багатоколісного роботів (рис. 11.6, б, в, г) треба знайти миттєвий центр обертання (Instantaneous Center of Rotation - ICR), який визначає ось повороту робота в залежності від розподілення навантаження на гусеницю або колеса.

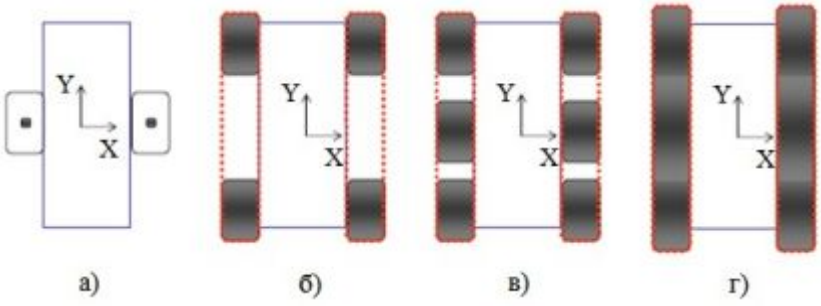


Рис. 11.6. Колісні та гусеничний мобільні роботи

11.3. Проектування систем керування гусеничних роботів

Позиційне переміщення гусеничного мобільного робота за допомогою одометра, аналогічно тому, як було зроблено для колісного робота з диференційним приводом, пов'язано з труднощами визначення миттєвого центру обертання, який залежить від центру ваги робота, який у свою чергу залежить від наявності та положення вантажу. А це в свою чергу впливає на радіус та центр повороту при переміщенні по дузі кола, що призводить до відповідної зміни траєкторії переміщення.

Тому керування переміщенням робота доцільно здійснювати за допомогою засобів локальної навігації, що дають можливість визначати положення робота відносно зовнішніх орієнтирів, наприклад, з використанням засобів машрутослідкування.

Спрощену структуру системи управління гусеничним мобільним роботом можна представити у вигляді рис. 11.7.

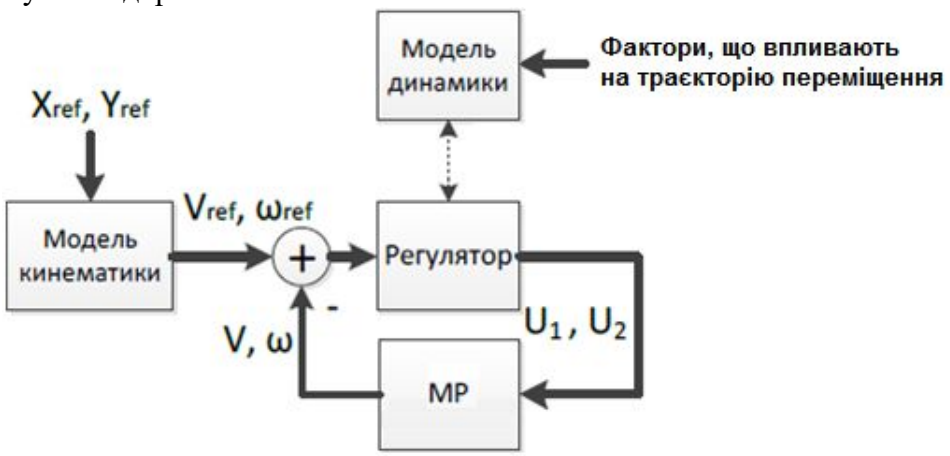


Рис. 11.7. Структурна схема системи управління

Кінематична модель служить для розрахунку заданих лінійної V_{ref} і кутової ω_{ref} швидкостей руху робота, відповідних заданій траєкторії X_{ref}, Y_{ref}. Модель динаміки використовується для розрахунку параметрів регулятора, який служить для вироблення керуючих напруг U_1, U_2 двигунів робота, в залежності від факторів, що впливають на траєкторію переміщення.

Контрольні запитання

1. Визначити,які основні типи конфігурації гусеничного рушія, що застосовуються у промислових мобільних роботах?
2. Розповісти,які конструкції гусеничних рушіїв є найбільш типовими для гусеничних мобільних роботів?
3. Визначити,у чому полягає відмінність гусеничного та багатоколісного роботів від роботів з диференціальним приводом?
4. Розповісти,як описати прямолінійний рух гусеничного робота?
5. Описати,як описати коловий рух гусеничного робота?
6. Визначити,як описати розворот з мінімальним радіусом для гусеничного робота?
7. Описати,як визначається миттєвий центр повороту?
8. Визначити,як знайти реальний радіус колового руху гусеничного робота?
9. Розповісти,як описати розворот на місці гусеничного робота?
10. Описати,як виглядає спрощена структура системи управління гусеничним мобільним роботом?

Лекція 12. Основні принципи проектування крокуючих мобільних роботів

12.1. Принципи проектування траєкторій переміщення крокуючих роботів

Мобільний робот з двома ногами найчастіше використовується для створення людиноподібних роботів, тому їх рухи повторюють рухи людини [9, 10].

Мобільні роботи з чотирма, шістьма та вісьмома ногами найчастіше використовується для створення твариноподібних роботів, тому їх рухи повторюють рухи тварин та комах.

Розглянемо кінематичну модель крокуючих роботів та вимоги до систем керування на прикладі роботів з двома та шістьма ногами.

Роботи з двома ногами, як вже було указано, найчастіше виконують функції людиноподібних роботів.

Людина здатна до руху в трьох площинах: фронтальній, сагітальній і горизонтальній, тому розрізняють три осі тіла: сагітальну, вертикальну і фронтальну. Назвам осей відповідає назва площин: сагітальна, фронтальна і горизонтальна (поперечна) площині (рис. 12.1).

Розрізняють такі рухи:

нахил вперед - назад, рух у сагітальній площині;

нахил вправо - вліво, рух у фронтальній площині;

поворот вправо – вліво, рух у поперечній площині.

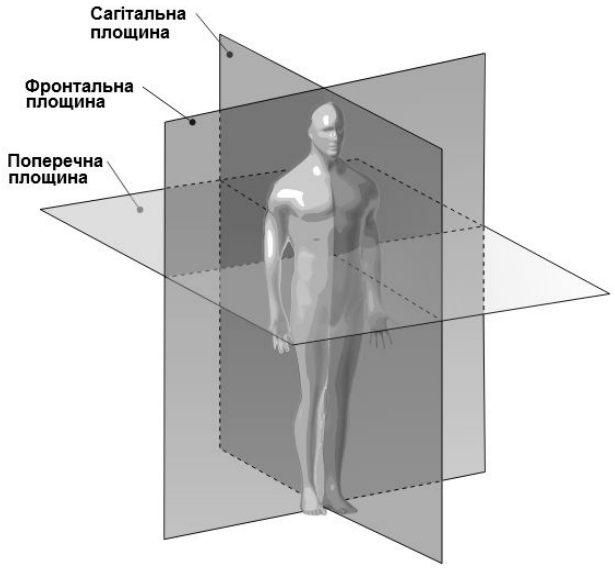
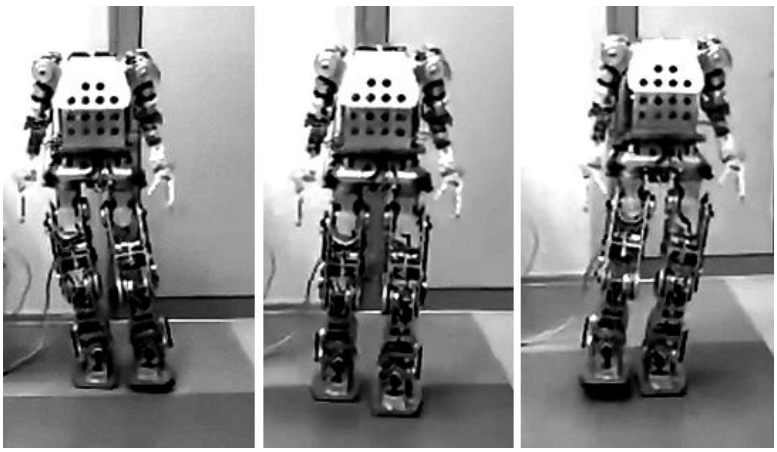


Рис. 12.1. Сагітальна, фронтальна і поперечна площині руху людини

Розглянемо, як здійснюється хода людиноподібних роботів (рис. 12.2). Під час ходи можна визначити одноопорний (рис. 12.2, а,в) та двохопорний (рис. 12.2, б) рух.



а) б) в)

Рис. 12.2. Одноопорний (а,в) та двохопорний (б) рух

До началу руху центр ваги переміщується на першу ногу, після чого друга нога

переміщується у напрямку руху. Потім центр ваги переміщується на другу ногу, після чого перша нога переміщується у напрямку руху.

Опорна циклограма руху може бути відображена діаграмою, наведеною на рис. 12.3. За її допомогою зручно проводити аналіз ритміки пересування.

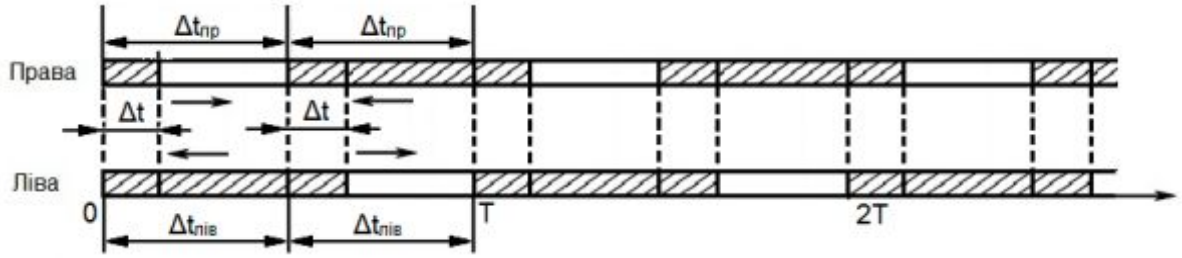


Рис. 12.3. Опорна циклограма руху

Найпростіші ритмічні рухи можна описати такими часовими параметрами:

T - період руху (через який послідовність рухів періодично повторюється),

$\Delta t_{пр}$, $\Delta t_{лів}$ – час між початком опори на першому ногу і початком опори на іншу,

Δt - час опори на обидві ноги,

$\Delta t_{пр} + \Delta t$, $\Delta t_{лів} + \Delta t$ – час опори на одну ногу.

Двонога хода здійснює пересування, при якому відбувається циклічна зміна одноопорного періодів (опори на одну з ніг і перенесення іншої ноги) і двохопорний періодів між ними (переміщення тіла с опорою на обидві ноги). На кожному наступному кроці махова нога стає опорною, а опорна стає маховою. На рис. 5.4 видно, що існує чотири фази ходьби У локомоціях людини фаза опори становить 58 ... 80% від тривалості одиночного кроку, а фаза переносу решта 42 ... 20%. Найімовірніше співвідношення при нормальній ходьбі становить 66% і 34% (або по 33 і 17 від повного рухового циклу).

Таким чином при переміщенні здійснюються таки рухи:

одноопорний рух - перенесення ноги спільно с рухом торсу при опорі на іншу ногу;

двоопорний - рух торса с опорою на обидві ноги.

На рис. 5.5 наведені фази циклу ходи (фази руху двоногого крокуючого робота).

У процесі ходи двоногого крокуючого робота відбувається послідовна зміна фаз (рис. 12.4):

А (одноопорна) права нога: переміщення корпусу і правої ноги с опорою на ліву ногу;

Б (двоопорна) права нога: перенесення тяжкості корпусу с лівої ноги на праву ногу с опорою на обидві ноги;

В (одноопорна) ліва нога: переміщення корпуса і лівої ноги с опорою на праву ногу;

Г (двоопорна) ліва нога: перенесення тяжкості корпусу с правої ноги на ліву ногу с опорою на обидві ноги.

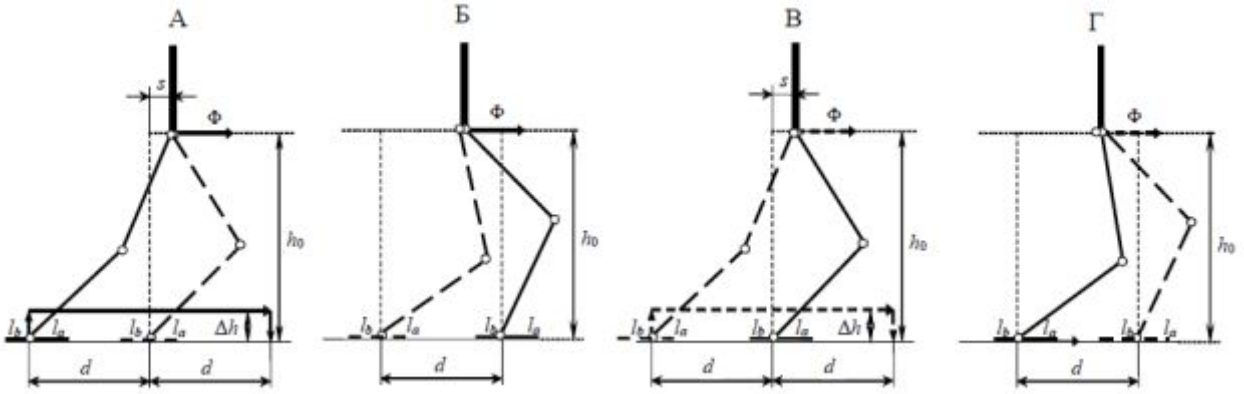


Рис. 12.4. Фази циклу ходи (фази руху двоногого крокуючого робота)

Принцип комфортабельності формулюється як оптимізаційне умова ходьби з наступним критерієм якості: рух двоногого механізму здійснюється так, щоб його центр мав відчував найменше середнє прискорення, а саме прагнув до рівномірного і прямолінійного руху.

Перевагою крокуючих роботів є можливість пересуватися у складних умовах, наприклад, по нерівній та похилій площині або підніматися і спускатися по сходах (рис. 12.5.)

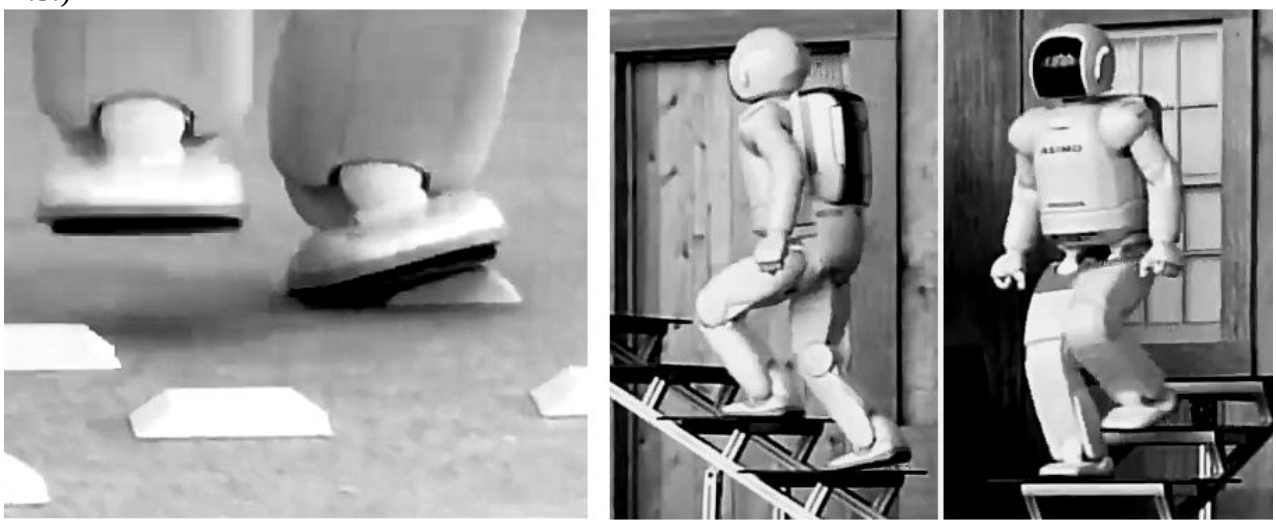


Рис. 12.5. Пересування у складних умовах

При цьому треба не тільки змінювати положення стопи, але й утримувати робот у такому стані, щоб центр ваги не вийшов за межи площі опори. Для забезпечення цього потрібно також здійснювати відповідний нахил корпусу робота відносно тазу, засобами що імітують роботу хребта людини.

Виходячи з необхідності забезпечення усіх рухів під час ходи узагальнена кінематична схема двоногого крокуючого робота може мати вигляд, наведений на рис. 12.6.

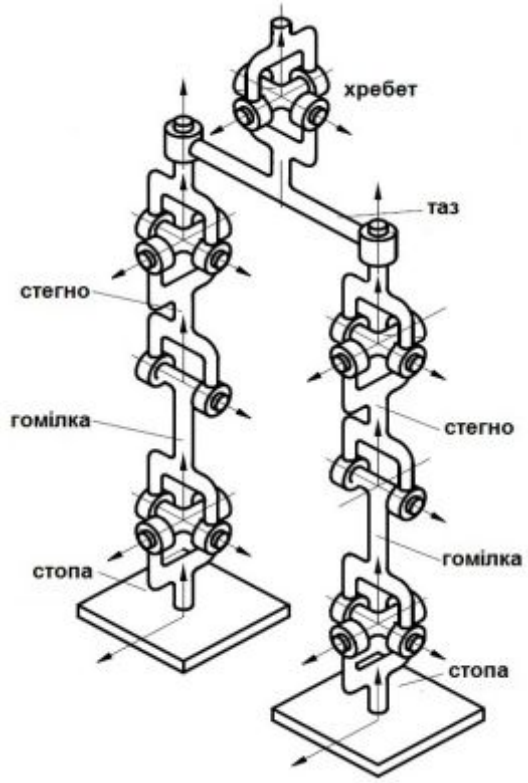


Рис. 12.6. Узагальнена кінематична схема двоногого крокуючого робота

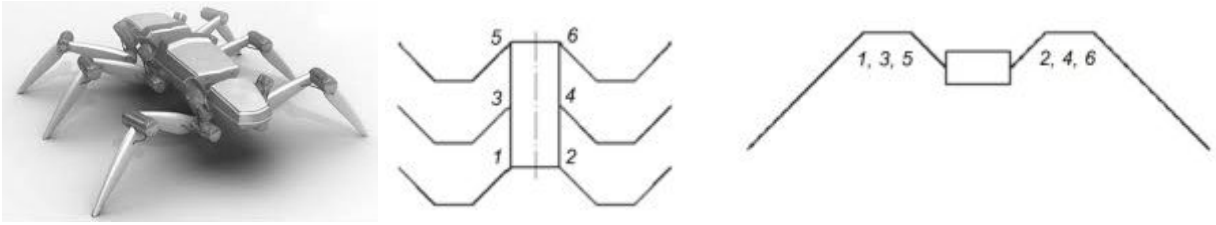
Для реалізації цієї схеми потрібні 14 регульованих приводів кутового переміщення з датчиками положення (кута повороту), з них по 6 приводів на кожну ногу та 2 приводи для переміщення корпусу робота відносно тазу для підтримки рівноваги. Підтримання рівноваги забезпечують датчики акселерометри, що здійснюють функції вестибулярного апарату людини.

Мобільні роботи з чотирма, шістьма та вісьмома ногами найчастіше використовується

для виконання транспортних функцій та переміщення у складних умовах. Наявність більшої кількості ніг ускладнює систему керування, яка повинна здійснювати узгоджене переміщення усіх ніг разом. З іншого боку така кількість ніг дає можливість одночасної опори на три і більше ніг, що спрощує систему утримання рівноваги, зводячи її до утримання потрібної орієнтації робота.

Розглянемо рух таких роботів на прикладі шестиногого робота, схематичне зображення якого наведено на рис. 12.7, а, а також номера ніг (рис. 12.7, б) та вихідне положення робота (рис. 12.7, в).

При переміщенні шестиногого робота використовується хода "трійка", тому що під час руху переставляються групи (трійки) ніг, що складаються с першої та останньої ніг з одного боку та середньої ноги з іншого боку. Таким чином для ходи "трійка" маємо дві групи ніг: 1, 4, 5 та 2, 3, 6.



а) б) в)

Рис. 12.7. Приклад шестиногого робота

Хода здійснюється зміною опорних ніг: … → 1, 4, 5 → 2, 3, 6 → 1, 4, 5 → 2, 3, 6 → …

На рис. 12.8 наведені фаза переносу та фаза опору. Пунктирна лінія позначає вихідне положення ніг, безперервна лінія - кінцеве.

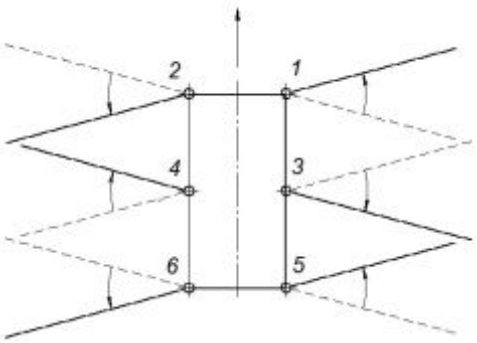
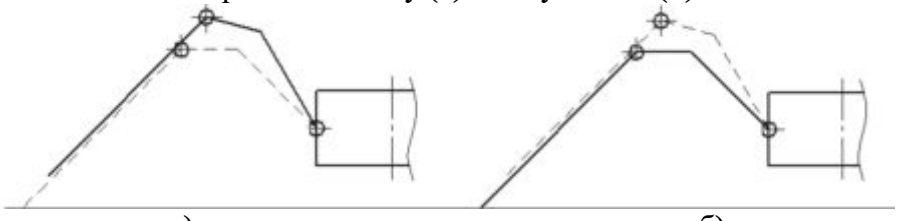


Рис. 12.8. Фаза переносу та фаза опору для ходи "трійка"

На рис. 12.9 наведені процес підйому (а) та опускання (б) ноги.



а) б)

Рис. 12.9 . Процес підйому (а) та опускання (б) ноги.

Для реалізації переміщення робота ходою "трійка" можна використати таку послідовність дій.

1. Визначається послідовність зміни опори ніг (у даному випадку … → 1, 4, 5 → 2, 3, 6 → 1, 4, 5 → 2, 3, 6 → …).

2. Визначається процес здійснення фази переносу та фази опору, а саме, які приводи використовуються для підйому та опускання ніг.

3. Визначення обмежень на зміну узагальнених координат.

4. Визначення часу виконання одного кроку T, виходячи з швидкості переміщення

робота.

5. Розбиття часу T на кількість проміжок часу, що відповідають кількості ітерацій при визначені зміни узагальнених координат під час руху.

6. Вибір прирощення узагальнених координат на кожній ітерації.

Така послідовність дій визначає алгоритм, який реалізує система керування робота для здійснення ходи.

Цей алгоритм значно ускладнюється, коли переміщення робота здійснюється по складній поверхні. Оскільки опорні ноги можуть знаходитись на різних рівнях, треба на стопах встановлювати датчики, що обмежують опускання ноги при торканні поверхні (наприклад, тактильні датчики або датчики механічних зусиль).

12.2. Проектування приводів крокуючих роботів

Крокуючі рушії забезпечують найбільшу прохідність мобільних роботів, які використовуються в складних природних умовах і бездоріжжя. Вони здатні забезпечити стійкий підйом і спуск по нахилам і сходах. Для стійкого проходження складних форм рельєфу крокуючий робот повинен бути адаптивним, тобто володіти здатністю адаптації до зміни форми рельєфу поверхні. У наш час крокуючі мобільні роботи в основному виконують функції людиноподібних роботів (роботів-гуманоїдів або андроїдів) та транспортні функції.

Частіше усього використовують крокуючі мобільні роботи з двома, чотирма та шістьома ногами (рис. 12.10).

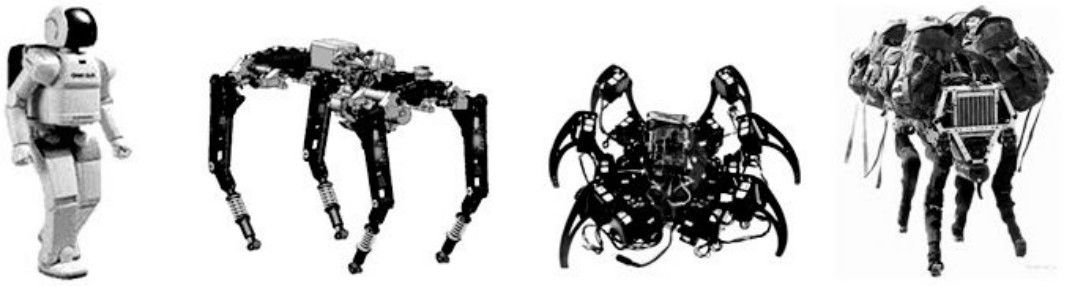


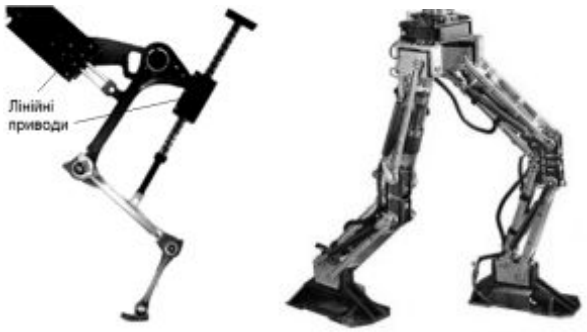
Рис. 12.10. Крокуючі мобільні роботи

Крокуючи роботи часто поділяють на антропоморфні (людиноподібні) та павукоподібні (чотирма, шістьома та вісьмома ногами) роботи.

Двоногі мобільні роботи в основному виконують функції людиноподібних роботів (роботів-гуманоїдів або андроїдів), мобільні роботи з двома, чотирма та шістьома ногами в основному виконують транспортні функції.

Крокуючи роботи для своїх рушіїв використовують приводи лінійного та кутового переміщення, що реалізуються на основі гідравлічних та електричних приводів.

На рис. 12.11 показані електричний (а) та гідравлічний (б) лінійні приводи (лінійні актуатори) для крокових рушіїв мобільних роботів.



а) б)

Рис. 12.11. Використання лінійних приводів для крокових рушіїв мобільних роботів

Принцип дії лінійного актуатора на основі електродвигунів наведений на рис 12.12.

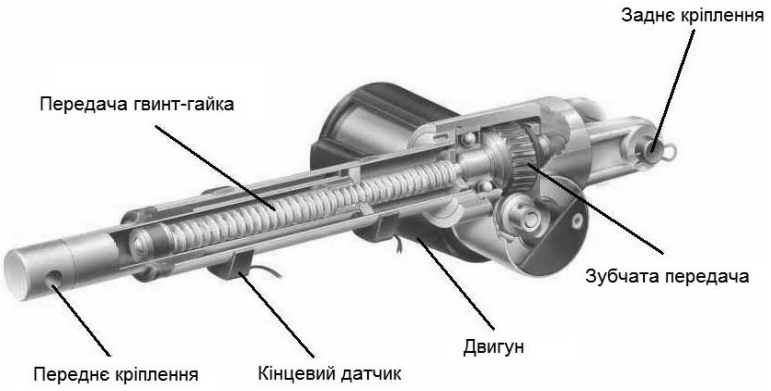


Рис 12.12. Принцип дії лінійного актуатора на основі електродвигунів

Лінійний актуатор являє собою систему позиціонування, в основі якої лежить перетворення обертального моменту електродвигуна в поступальний рух штока. Як правило, такий пристрій включає в себе сам двигун, редуктор, датчик повороту ротора двигуна і кінцевий вимикач.

Довжина висунутою частини штока у типових лінійних актуаторів може мати значення від 50мм до 500мм. Швидкість руху штока до 50мм / с в залежності від навантаження. Прикладається лінійним приводом до об'єкта сила приймає значення 200Н до 10000Н в залежності від моделі.

Принцип дії гідравлічного лінійного актуатора наведений на рис 12.13.

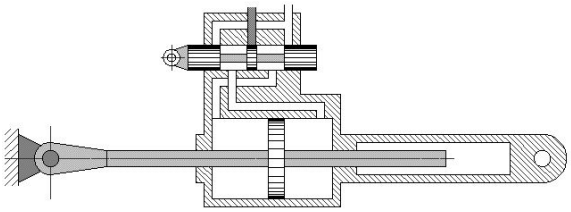


Рис 12.13. Принцип дії гідравлічного лінійного актуатора

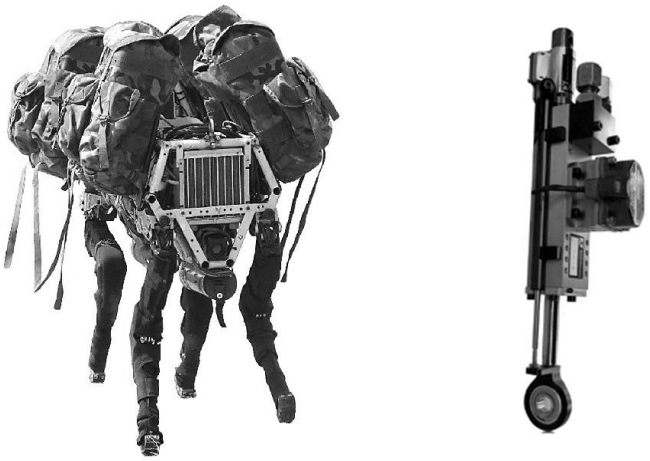
Переваги гідроприводів:

- можливість одержання великих сил та обертальних моментів при порівняно малих розмірах та масі гідродвигунів;
- плавність рухів вихідних ланок;
- можливість безступінчастого регулювання швидкості у широкому діапазоні;
- мала інерційність;
- простота реалізації основних видів рухів: обертального, зворотно-поступального і зворотно-поворотного.

На жаль, гідравлічні приводи мають ряд недоліків, які ускладнюють їх використання в невеликих роботах:

- гідроприводи поступаються електричним у відстані транспортування енергії від джерела до споживача та швидкості передачі командних сигналів;
- у гідроприводах актуальним є питанням забезпечення герметичності порожнин, що знаходяться під тиском;
- чутливість до в'язкості робочої рідини, котра у свою чергу залежить від температури;
- нижчий к.к.д. у порівнянні з механічними передачами у приводах.

Виходячи з цього гідравлічні приводи застосовуються у транспортних мобільних роботах, наприклад, у військовому транспортному роботі BigDog (рис 12.14, а), в якому використовуються лінійні гідроприводи для переміщення ніг (рис 12.14, б).

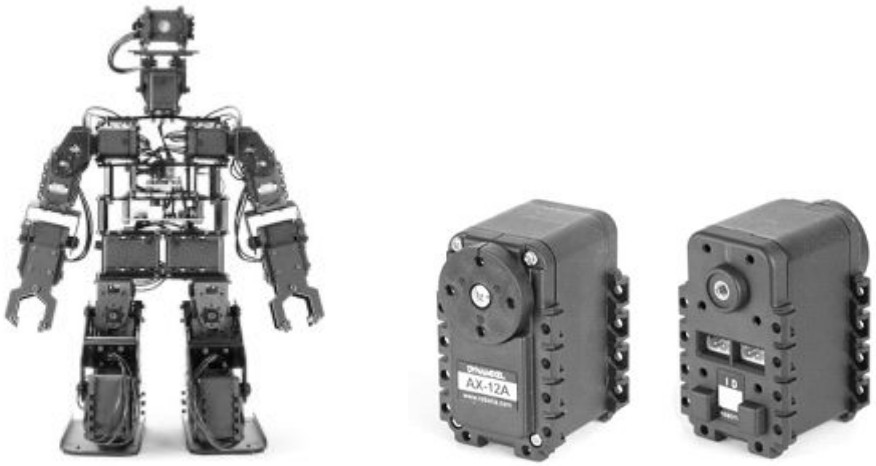


а) б)

Рис. 12.14. Військовий транспортний робот BigDog

Для крокових рушіїв відносно невеликих роботів, наприклад, **i-Sobot, Bioloid, Robonova** використовують сервоприводи, що здійснюють регульоване обертання у обмеженому значенні куту повороту, наприклад, від 0 до 300° при точності встановлення кута менш ніж 0,5° .

На рис. 12.15 наведений робот HR-OS1 Humanoid фірми Trossen Robotics (а) з серводвигунами Dynamixel (б).



а) б)

Рис. 12.15. Робот HR-OS1 Humanoid (а) з серводвигунами Dynamixel (б)

Ці приводи можна використовувати як для переміщення суглобів ніг та рук крокуючих роботів, так і для повороту рульового колеса у колісних мобільних роботах.

12.3. Проектування систем керування крокуючих роботів

Складність апаратних компонент та алгоритму керування крокових роботів потребує досить складної системи керування, яка здатна виконувати усі функції у реальному часі. Кожна нога робота для кожного суглоба повинна мати регульований привод та датчик положення, які повинні здійснити визначений рух ноги, відповідно з рухом самого робота у встановленому напрямку або по встановленій траєкторії. Крім того у мобільного робота повинна бути система навігації, що встановлює положення робота и визначає траєкторію його переміщення.

При вирішенні відносно простих функцій двоногий мобільний робот може використовувати централізовану систему керування, оскільки сучасні мікроконтролери можуть мати досить велику кількість входів та виходів для підключенні приводі та датчиків. Так, наприклад, мікроконтролер Arduino Mega має 54 двійкових входів та виходів, 15 виходів

з широтно-імпульсною модуляцією та 16 аналогових входів, що дає можливість безпосередньо керувати 15 приводами.

Для створення програм керування такими роботами можна використовувати усі розглянуті вище засоби створення програм для контролерів Arduino.

Для мобільних роботів з чотирма та більшою кількістю ніг загальна кількість регульованих приводів складає декілька десятків, тому у цьому випадку здається доцільним використання розподілених систем керування, де загальне керування здійснюють обчислювальні пристрої, здатні реалізувати досить складні алгоритми керування переміщенням робота, а керування окремими ланками виконують відносно прості або спеціалізовані мікроконтролери, які реалізують локальні системи керування. Взаємодія цих окремих систем керування здійснюється за допомогою локальних мереж, що забезпечують своєчасний обмін даними між усіма компонентами мережі.

Контрольні запитання

1. Визначити основні типи крокуючих роботів?
2. Назвати,які функції в основному виконують двоногі мобільні роботи?
3. Розповісти,які функції в основному виконують мобільні роботи з двома, чотирма та шістьома ногами?
4. Визначити,як здійснюється хода людиноподібних роботів?
5. Назвати,чим відрізняються одноопорний та двоопорний рух?
6. Розповісти,як виглядає узагальнена кінематична схема двоногого крокуючого робота?
7. Описати, як здійснюється підтримання рівноваги двоногого робота?
8. Визначити,чим відрізняються роботи з чотирма, шістьма та вісьмома ногами від двоногих роботів?
9. Описати, як здійснюється хода "трійка"?
10. Визначити,які основні компоненти повинна мати система керування крокуючих роботів?

ЛІТЕРАТУРА

1. Навчальний посібник з дисципліни Маніпулятори та промислові роботи. Для студентів бакалаврів, спеціальності: 131 - Прикладна механіка, 133 – Галузеве машинобудування, / Укладачі: Михайлов Є. П., Лінгур В.М. – Одеса: ОНПУ, 2019. - 233 с. URL: http://dspace.opu.ua/jspui/handle/123456789/8365

2. Гай В. Е. Microsoft ® Robotics Developer Studio. Программирование алгоритмов управления роботами / Гай В.Е.; М.: ЭКОМ Паблишерз, 2012. — 184 с.: ил. [Электронный ресурс] – Режим доступа: URL: https://iagsav.github.io/data/books/Robot_Book.pdf

3. Цвіркун Л.І. Робототехніка та мехатроніка: навч. посіб. / Л.І. Цвіркун, Г. Грулер ; під заг. ред. Л.І. Цвіркуна ; М-во освіти і науки України, Нац. гірн. ун-т. –3-тє вид., переробл. і доповн. – Дніпро: НГУ, 2017. – 224 с. URL: http://ir.nmu.org.ua/handle/123456789/152141?show=full

4. Петров И. В. Программируемые контроллеры. Стандартные языки и приемы прикладного проектирования / Под ред. проф, В. П. Дьяконова. — М.: СОЛОН-Пресс, 2004. — 256 с : ил. — (Серия «Библиотека инженера») ISBN 5-98003-079-4 URL: https://ua1lib.org/book/3187992/bde81e?regionChanged=&redirect=194369203

5. Навчальний посібник з дисциплін «Електронні, мікропроцесорні та обчислювальні пристрої ГВС, ПТМ та ЛС» для студентів за фахом 131 – Прикладна механіка – спеціалізації – Мехатроніка та промислові роботи, Інженерія логістичних систем, 133 – Галузеве машинобудування – спеціалізація – Підйомно-транспортні, будівельні, дорожні машини і обладнання / Укладач: Михайлов Є. П. Одеса: ОНПУ. 2018. – 121 с. URL: http://dspace.opu.ua/jspui/handle/123456789/8143

6. С++. Основи програмування. Теорія та практика : підручник / [О.Г. Трофименко, Ю.В. Прокоп, І.Г. Швайко, Л.М. Буката та ін.] ; за ред.О.Г.Трофименко. – Одеса: Фенікс, 2010. – 544 с. ISBN 978-966-438-240-0 URL: http://www.dut.edu.ua/uploads/l_538_66572861.pdf

7. Blum J. Exploring Arduino: Tools and Techniques for Engineering Wizardry John Wiley & Sons, Inc., 2013. — 385 с. — ISBN: 978-1-118-54936-0. URL: https://www.twirpx.com/file/1371963/

8. Доля В.М. Програмування, введення та відпрацювання управляючих програм для верстатів з ЧПУ та РТК: Навчальний посібник. − Харків: НТУ „ХПІ”, 2004. − 169 с. URL: http://repository.kpi.kharkov.ua/bitstream/KhPI-Press/5820/1/Dolya_Prohramuvannia_vvedennia_2004.pdf

9. Михайлов Є. П. Навчальний посібник з дисципліни "Мобільні роботи"для студентів за фахом 131 - Прикладна механіка – спеціалізація - Мехатроніка та промислові роботи / Укладач: Михайлов Є. П. Одеса: ОНПУ. 2016, – 239 с. Укладач: Михайлов Є. П. Одеса: ОНПУ, 2016, – 239 с. Рег.ном. НП07368 01.06.2016 №3765-РС-2016 Одеса: ОНПУ, 2016 URL: http://memos.library.opu.ua:8080/memos/jsp/materials.iface?mId=28063

10. Павловский В.Е. О разработках шагающих машин // Препринты ИПМ им. М.В.Келдыша. 2013. № 101. 32 с. [Электронный ресурс] – Режим доступа: URL:http://library.keldysh.ru/preprint.asp?id=2013-101

11. Проць Я.І., Автоматизація виробничих процесів. Навчальний посібник для технічних спеціальностей вищих навчальних закладів./ Я.І. Проць, В.Б. Савків,О.К. Шкодзінський, О.Л. Ляшук – Тернопіль: ТНТУ ім. І.Пулюя, 2011. – 344с. URL: https://www.twirpx.com/file/713827/

12. Проць. Я.І. Захоплювальні пристрої промислових роботів: Навчальний посібник . – Тернопіль: Тернопільський державний технічний університет ім. І. Пулюя, 2008. – 232с.URL: https://www.studmed.ru/proc-ya-savkv-vb-ta-n-avtomatizacya-virobnichih-procesv_93709a79ae5.html

ІНФОРМАЦІЙНІ РЕСУРСИ

13. Coppelia Robotics URL: https://www.coppeliarobotics.com/features.html (Last accessed: 26.02.2021).
14. Operating manual RobotStudio 5.15 Document ID: 3HAC032104-001 Revision: J © Copyright 2008-2012 ABB. All rights reserved URL: https://library.e.abb.com/public/244a8a5c10ef8875c1257b4b0052193c/3HAC032104-001_revD_en.pdf (Last accessed: 07.02.2021).
15. RobotStudio URL: https://new.abb.com/products/robotics/robotstudio/downloads (Last accessed: 26.02.2021).
16. my.KUKA Portal Planning, project engineering, service and safety. URL: https://www.kuka.com/en-us/products/robotics-systems/software/simulation-planning-optimization (Last accessed: 26.02.2021).
17. KUKA System Software. KUKA System Software 8.3. Operating and Programming Instructions for End Users. KUKA Roboter GmbH URL: http://www.wtech.com.tw/public/download/manual/kuka/krc2ed05/Operating%20and%20Programming.pdf
18. Studfiles. Разработка модели роботизированного участка погрузки URL: https://studfiles.net/preview/785308/
19. Download the Arduino IDE URL: https://www.arduino.cc/en/Main/Software (Last accessed: 27.01.2020).
20. Программирование Ардуино. Справочник языка Ардуино. Arduino UA. URL: https://doc.arduino.ua/ru/prog/#Structure (Last accessed: 10.04.2020).

Додаток 1 Мова програмування для контролерів Arduino

Розглянемо можливості середовища розробки Arduino IDE для комп'ютерних розрахунків роботів [19, 20].

Програма Arduino, яку називають скетч, складається з самостійного файла, в якому, на відміну від мови С, треба визначити принаймні, дві секції: перша setup(), друга loop().

Змінні, доступні з обох секцій програми, повинні бути оголошені за їх межами, як глобальні змінні.

Як тільки програма запуститься, виконуються операнди, розміщені в блоці setup(): вони призначені для ініціалізації значень змінних на початку запуску, а також для налаштування портів периферії Arduino.

Після закінчення обробки setup() Arduino починає циклічне виконання інструкцій в блоці loop().

Після виконання всіх операндів, цикл повторюється знову і знову.

```
void setup () {
 ...
 }
void loop () {
 ...
 }
```

Обидва блоки setup() та loop() задекларовані як блоки void, тобто вони нічого не повертають.

Змінна - це місце зберігання даних. Вона має ім'я, значення і тип.

Програма використовує змінні для зберігання проміжних даних обчислень.

Для обчислень можуть бути використані дані різних форматів, різної розрядності, наприклад, такі типи (табл. 3.1).

Типи даних вибираються виходячи з необхідної точності обчислень, форматів даних і т.п.

Можна використовувати типи даних, у яких перша частина uint або int означає число без знаку або зі знаком, далі йде розмір даних в бітах, а _t вказує що це тип. Наприклад:

```
uint32_t x;     - означає, що x змінна без знаку, яка складається з 4 байт,
int16_t x;      - означає, що x змінна зі знаком, яка складається з 2 байт.
```

Всі змінні повинні бути оголошені до того як будуть використовуватися.

Для оголошення змінних треба вказуєти тип даних, а потім ім'я змінної.

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

Таблиця Д1.1

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

Змінна може бути оголошена в будь-якій частині програми, але від цього залежить, які блоки програми можуть її використовувати.

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

Локальні змінні, що оголошуються всередині функцій або таких блоків, як цикл ***for***, можуть використовуватися тільки в оголошених блоках.

При оголошенні змінної можна задати її початкове значення (проініціалізувати), наприклад:

int x = 0; // оголошується змінна x з початковим значенням 0

Можна використовувати стандартні константі формату float, наприклад,
Перерахування радіанів у градуси:
RAD_TO_DEG = 57.295779513082320876798154814105f;
Перерахування градусів у радіани:
DEG_TO_RAD = 0.017453292519943295769236907684886f;
Число π:
PI = 3.14159265358979323846264338327 95f;
Число π/2:
PI_BY_TWO = 1.57079632679489661923132169163 98f;
Число 2π:
TWO_PI = 6.28318530717958647692528676655 9f;

Функції та оператори

Арифметичні операції:

=	присвоєння;
+	складання;
-	віднімання;
*	множення.

Ці оператори повертають результат виконання арифметичних дій над двома операндами. Результат, що повертається, буде залежати від типу даних операндів, наприклад, 9/4 поверне 2, тому що операнди 9 і 4 мають тип int. Також слід стежити за тим, щоб результат не вийшов за діапазон допустимих значень за видом даних.

Так, наприклад, складання 1 зі змінною типу int і значенням 32 767 поверне -32 768. Якщо операнди мають різні типи, то тип з більш "широким" діапазоном буде використаний для обчислень.

Якщо один з операндів має тип float або double, то арифметика "з плаваючою комою" буде використана для обчислень.

Синтаксіс:

result = value1 + value2;
result = value1 - value2;
result = value1 * value2;
result = value1 / value2.

Параметри:

value1: будь-яка змінна або константа;
value2: будь-яка змінна або константа.

Операції відношення:

==	дорівнює
!=	не дорівнює
<	менше
>	більше
<=	менше або дорівнює
>=	більше або дорівнює

Логічні операції:

&&	логічне І

|| логічне АБО
! логічне НІ

Квадратний корінь числа.

sqrt(x)
Параметри: x: число, будь-який тип даних.
Значення, що повертаються: double, квадратний корінь числа.

Квадрат числа.

sq(x)
Параметри: x: число, будь-який тип даних.
Значення, що повертаються: квадрат числа.

Тригонометричні функції

sin(rad), cos(rad), tan(rad).
Параметри: rad: кут у радіанах, тип даних float.
Значення, що повертаються: тригонометрическая функція, тип даних *double.*

Зворотні тригонометричні функції

asin(x), acos(x), atan(x).
Параметри: x: число, тип даних float.
Значення, що повертаються: зворотна тригонометрическая функція у радіанах, тип даних float.

Управління програмою

Оператори управління програмою визначають послідовність виконання програми. До них можна віднести такі оператори.

Оператор IF перевіряє умову в дужках і виконує наступне вираження або блок у фігурних дужках, якщо умова істинна.

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

Оператор IF … ELSE дозволяє зробити вибір між двох варіантів.

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

Оператор ELSE IF – дозволяє зробити багаторазовий вибір

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

Оператор While буде обчислювати в циклі безперервно і нескінченно доти, поки вираз в круглих дужках, () не стане логічно помилковим.

Щось треба змінювати значення перевіряється змінної, інакше вихід з циклу while ніколи не буде досягнутий. Ця зміна може відбуватися як в програмному коді, наприклад, при збільшенні змінної, так і в зовнішніх умовах, наприклад, при тестуванні датчика.

```
var = 0;
while(var < 200){
// виконати щось, повторивши 200 разів
}
```

Цикл FOR. Конструкція дозволяє організовувати цикли з заданим кількістю ітерацій. Синтаксис виглядає так:

```
for (дія до початку циклу; умова продовження циклу; дія в кінці кожної ітерації)
{…// код тіла циклу
}
```

Приклад циклу з 100 ітерацій.

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

Функції управління вводом / виводом.

Для роботи з цифровими виводами в системі Ардуіно є такі функції, дозволяють задати режим виводу та встановити виводи в певний стан.

Для визначення стану виводів в цих функціях використовуються константи HIGH і LOW, які відповідають високому і низькому рівню сигналу.

Функція pinMode(pin, mode)

Встановлює режим виводу (вхід або вихід).

Аргументи: pin і mode.

- pin - номер виводу;
- mode - режим виведення.

mode = INPUT вивід визначено як вхід, підтягуючий резистор відключений.

mode = INPUT_PULLUP вивід визначено як вхід, підтягуючий резистор підключений (вихідний стан HIGH).

mode = OUTPUT вивід визначено як вихід.

Функція не повертає нічого.

Функція digitalWrite(pin, value)

Встановлює стан виходу (високий або низький).

Аргументи pin і value:

- pin - номер виводу;
- value - стан виходу.

value = LOW встановлює вихід в низький стан

value = HIGH встановлює вихід в високе стан

Функція не повертає нічого.

Функція digitalRead(pin)

Зчитує стан входу.

Аргументи: pin - номер виводу.

Повертає стан входу:

- якщо digitalRead (pin) = LOW, то на вході низький рівень;
- якщо digitalRead (pin) = HIGH, то на вході високий рівень

Функція analogRead(pin)

Зчитує стан аналогового входу.

Аргументи: pin - номер виводу.

Повертає стан входу, що мають значення від 0 до 1023, які відповідають вхідній напрузі в діапазоні 0 – 5В.

Функція analogWrite(pin, value)

Видає аналогову напругу у вигляді сигналу з широтно-імпульсною модуляцією.

Аргументи pin і value:

- pin - номер виводу;
- value – коефіцієнт заповнення у межах від 0 (нема сигналу) до 255 (постійна напруга)..

Засоби обміну даними

Монітор порту Ардуіно та Плотер по послідовному з'єднанню - це утиліти, які вбудовані в середу програмування Arduino IDE і служить для зв'язку комп'ютера з контролером.

Монітор порту видає дані у символьному вигляді.

Плотер по послідовному з'єднанню видає дані у графичному вигляді.

Для відкриття утиліти Монітор порту необхідно натиснути на іконку в правому верхньому куті Arduino IDE, використовувати комбінацію клавіш Ctrl + Shift + M або вибрати в панелі меню: Инструменты > Монитор порта.

Для відкриття утиліти Плотер по послідовному з'єднанню необхідно вибрати в панелі меню: Инструменты > Плоттер по последовательному соединению.

Монітор порту Ардуіно

Для роботи з утилітою, використовують такі команди:

Serial.begin (); - команда запускає послідовний порт
Serial.end (); - зупиняє і очищає послідовний порт
Serial.print (); - відправляє дані в послідовний порт
Serial.print(78) - відправляє "78"
Serial.print(1.23456) - відправляє "1.23"
Serial.print(1.23456, 0) - відправляє "1"
Serial.print(1.23456, 2) - відправляє "1.23"
Serial.print(1.23456, 4) - відправляє "1.2346"
Serial.print('N') - відправляє "N"
Serial.print("Hello world.") - відправляє "Hello world."
Serial.print("\t"); - табуляція
Serial.println (); - відправляє дані з перенесенням рядка
Serial.read (); - приймає дані з послідовного порту
Serial.parseInt (); - читання цілих чисел з монітора
Serial.parseFloat(); - читання чисел з плаваючою точкою з монітора
Serial.available() - повертає кількість байт (символів) доступних для зчитування з буфера послідовного порту

Введення даних з монітору здійснюється за допомогою функції:

```
while (Serial.available() > 0)
{
Var1 = Serial.parseFloat(); // Перше отримане число
Var2 = Serial.parseFloat(); // Друге отримане число
if (Serial.read() == '\n') // Кінець передачі
{
Var3 = sqrt(sq(Var1) + sq(Var2)); //обчислення кореня квадратного з суми квадратів
…
}
```

Плотер по послідовному з'єднанню

Плотер по послідовному з'єднанню здійснює виведення даних з контролера на монітор комп’ютера у графічному вигляді.

Для виведення даних на Плотер по послідовному з'єднанню використовуються команди Serial.print (); для кожної змінної, що поділяються командою табуляції Serial.print("\t"); . Для виведення останної змінної використовуються команда Serial.println ();.