МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»

Кафедра “Підйомно-транспортного та робототехнічного обладнання"

НАВЧАЛЬНИЙ ПОСІБНИК

з дисципліні «Комп'ютерне конструювання елементів машин»

Перший (бакалаврський) рівень вищої освіти

Спеціальність – 131 ПРИКЛАДНА МЕХАНІКА
Освітня програма – *Інженерія логістичних систем, Мехатроніка та промислові роботи*
Спеціальність – 133 ГАЛУЗЕВЕ МАШИНОБУДУВАННЯ
Освітня програма – *Підйомно-транспортні, дорожні, меліоративні машини і обладнання*

ОДЕСА, 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»

Кафедра “Підйомно-транспортного та робототехнічного обладнання"

Михайлов Євген Павлович

НАВЧАЛЬНИЙ ПОСІБНИК

з дисципліні «Комп'ютерне конструювання елементів машин»

Перший (бакалаврський) рівень вищої освіти

Спеціальність – 131 ПРИКЛАДНА МЕХАНІКА
Освітня програма – *Інженерія логістичних систем, Мехатроніка та промислові роботи*
Спеціальність – 133 ГАЛУЗЕВЕ МАШИНОБУДУВАННЯ
Освітня програма – *Підйомно-транспортні, дорожні, меліоративні машини і обладнання*

Затверджено

на засіданні вченої ради українсько-німецького навчально-наукового інституту

Протокол 7 від 16 березня 2023 р.

ОДЕСА, 2023

Комп’ютерне конструювання елементів машин. Навчальний посібник для здобувачів бакалаврів, спеціальність: 131 - Прикладна механіка, освітні програми: Мехатроніка та промислові роботи, Інженерія логістичних систем, спеціальність: 133 – Галузеве машинобудування, освітня програма: Підйомно-транспортні, дорожні, меліоративні машини і обладнання: / Укл.: Михайлов Є. П. Одеса: Одеська політехніка, 2023. 233 с.

В навчальному посібнику розглянуті питання комп’ютерного конструювання елементів машин у складі підйомно-транспортних машин, робототехнічних пристроїв та в обладнанні логістичних систем. Наведені засоби комп’ютерного конструювання механічних, інформаційних, виконавчих та керуючих елементів машин. Надані завдання до практичних та лабораторних занять.

Укладач:
Михайлов Євген Павлович, доцент кафедри підйомно-транспортного і робототехнічного обладнання

Рецензенти:
Сидоренко Ігор Іванович, професор кафедри інформаційних технологій проектування в машинобудуванні
Чесноков Альберт Валерійович, директор ТОВ «Інвест Інжиніринг»

Зміст

Передмова

В умовах відновлення та модернізації сучасних виробничих та логістичних систем в Україні пріоритетна увага має надаватися підготовці нової генерації фахівців, яка здатна оволодіти сучасними методами комп'ютерного конструювання елементів машин. Однією з передумов вирішення цього надзвичайно важливого і складного завдання є опанування фахівцями ґрунтовними знаннями, уміннями, навичками, ставленнями у питаннях конструювання, проєктування та застосування механічних, інформаційних, виконавчих та керуючих елементів машин.

Курс "Комп'ютерне конструювання елементів машин" є важливою частиною підготовки бакалаврів до практичної діяльності, що розроблений на основі робочої навчальної програми з дисципліни "Комп'ютерне конструювання елементів машин" є нормативним документом Національного університету «Одеська політехніка», який розроблено кафедрою підйомно-транспортного та робототехнічного обладнання на основі освітньої програми підготовки відповідно до навчального плану першого (бакалаврського) рівня спеціальності "Прикладна механіка" денної форми навчання. Робочу навчальну програму укладено згідно з вимогами семестрової системи організації освітнього процесу в ОП. Програма визначає обсяги компетентностей, які повинен опанувати студент відповідно до освітньої програми, алгоритму вивчення навчального матеріалу дисципліни "Комп'ютерне конструювання елементів машин".

"Комп'ютерне конструювання елементів машин" є складовою частиною дисциплін циклу професійної підготовки нормативного блоку. Її вивчення передбачає розв'язання низки завдань фундаментальної професійної підготовки фахівців першого рівня, зокрема: опанування студентами основ конструювання, проєктування та застосування елементів машин в сучасних робототехнічних системах та поширення їх знань за рахунок розгляду різних типів елементів машин, що базуються на сучасних методах та методиках.

Мета вивчення дисципліни – забезпечити розвиток загальних та спеціальних компетентностей майбутніх бакалаврів, розвити здатність використовувати отримані теоретичні знання з основ комп'ютерного конструювання елементів машин для вирішення професійних завдань з використання елементів машин у складі підйомно-транспортних, робототехнічних та логістичних систем.

Завдання вивчення дисципліни: Надати здатність виконувати моделювання об'єктів галузевого машинобудування з використанням спеціальних пакетів та засобів автоматизованого проектування та використовувати САПР, CAD/CAE системи при проектуванні, розрахунку деталей та об'єктів підйомно-транспортних, будівельних, дорожніх машин і обладнання. Надати здатність розробляти деталі та вузли машин на базі систем автоматизованого проектування.

Дисципліна "Комп'ютерне конструювання елементів машин" є однією з основних у системі підготовки бакалаврів. Вона узагальнює набуті студентами знання у галузі конструювання та проєктування елементів машин поширює їх в напрямках, що базуються на сучасних моделях і методах.

В навчальному посібнику розглянуті питання комп'ютерного конструювання елементів машин у складі робототехнічних пристроїв, підйомно-транспортних машин та в обладнанні логістичних систем. Наведені механічні, інформаційні, виконавчі та керуючі пристрої елементів машин. При цьому найбільша увага надається проєктуванню та конструюванню комп'ютерних систем керування.

Надані завдання до практичних занять та самостійної роботи здобувачів та приклади розв'язання задач.

1. Задачі проєктування та конструювання елементів машин

1.1. Принципи проєктування та конструювання елементів машин

Маши́на (від лат. machina, від дав.-гр. μηχανή — «пристрій, засіб, знаряддя») — технічний об'єкт, який складається із взаємопов'язаних функціональних частин (деталей, вузлів, пристроїв, механізмів та ін.), що використовує енергію для виконання покладених на нього функцій.

Машиною також можна назвати пристрій, який виконує механічний рух для перетворення енергії, матеріалів та інформації [1].

В залежності від призначення розрізнюють енергетичні, робочі та інформаційні машини.

Енергетичні машини (машини-двигуни) призначені для перетворення будь-якого виду енергії у механічну. До енергетичних машин відносяться двигуни внутрішнього згоряння, електричні двигуни, електрогенератори, парові машини, компресори тощо.

Робочі машини — це машини що використовують механічну чи іншу енергію для перетворення і переміщення предметів обробки та вантажів. До них належать технологічні машини і апарати, котрі призначенні для змінювання розмірів, форми, властивостей або стану предмета обробки (сировини), а також підйомніо-транспортні машини, призначені для переміщення вантажів у просторі. Виходячи з цього робочі машини поділяються на технологічні та транспортні.

Інформаційні машини, що призначені для перетворення, обробки та передачі інформації.

Відповідно напрямку використання машин у складі робототехнічних, підйомно-транспортних та мехатронних систем обмежимось розгляданням робочих машин.

За ступенем автоматизації усі машини поділяються на машини з ручним керуванням, автомати і напівавтомати.

Машини з ручним керуванням виконують свої функції тільки за безпосередньої участі в їх роботі людини. Людина здійснює пуск машини, управління роботою всіх її механізмів та зупинку машини після виконання певних робіт чи операцій (верстати, будівельні машини, транспортні та транспортуючі машини тощо).

Автоматична машина — самостійно діюча машина, яка виконує свою функцію згідно із заданою керуючою програмою без безпосередньої участі людини у процесі обробки, перетворення, передавання та використання матеріальних об'єктів, енергії чи інформації. Прикладом таких машин є автономні мобільні роботи.

Автоматизовані засоби (напівавтомати) — це машини, в яких робочий цикл, що здійснюється на основі попередньо заданої керуючої програми, переривається і для його повторення необхідне обов'язкове втручання людини.

За функційними ознаками у структуру машини входять взаємопов'язані елементи (механізми), на кожен з яких покладена певна функція.

З точки зору функціонального призначення елементи машин поділяються на такі види:

- механізми двигунів і перетворювачів;
- передавальні механізми (редуктори, передачі тощо);
- виконавчі механізми (механізми захоплення та переміщення вантажів, технологічне обладнання тощо);
- засоби управління, контролю та регулювання (давачі, дія яких ґрунтується на зміні електричних величин в процесі впливання контрольованих механічних, акустичних, теплових, електро-магнітних, оптичних та інших зовнішних величин; комп'ютерні системи обробки даних та керування; електро-механічні виконавчі пристрої).

Сучасна робоча машина найчастіше складається з таких основних механізмів: двигуна, передавального і виконавчого механізмів, комп'ютерної системи керування.

Конструкція машини складається з деталей, вузлів та агрегатів. Кожен з цих елементів має предметну чи функціональну спеціалізацію, повне призначення і разом з

тим узгоджується з іншими елементами машини, утворюючи в сукупності цілісну діючу конструкцію.

Деталь — елемент машини, кожен з яких являє собою одне ціле і не може бути розібраний без руйнування на простіші складові ланки машин.

За ознаками застосування та поширення в машинобудуванні деталі можна розділити на такі групи:

стандартні — це деталі, що виготовляються відповідно до державних, галузевих стандартів або стандартів підприємства;

уніфіковані — це деталі, запозичені з іншого виробу, тобто раніше спроєктовані як оригінальні;

оригінальні — деталі, що конструюють для певної машини і вони, як правило, раніше не проєктувались і не виготовлялись.

Вузол — частина машини, яка являє собою роз'ємне або нероз'ємне з'єднання декількох деталей, яке можна зібрати окремо від інших складових частин машини або механізму і яке здатне виконувати певні функції у виробах одного призначення тільки спільно з іншими складовими частинами.

Агрегат — нормалізований вузол машини, який забезпечує повну взаємозамінюваність і самостійно виконує властиві йому функції.

Типовими зразками агрегатів, які входять до складу машини, є електричні двигуни, редуктори, генератори електричного струму тощо. З агрегатів компонують деякі машини машинобудування; велика кількість агрегатів входить до складу обладнання металургійного виробництва, до транспортної і транспортуючої техніки, до машин переробної промисловості тощо.

Проектування - це виконання певного набору робіт - дослідних, розрахункових і конструкторських.

Оскільки роботи складаються з механічних компонент, приводів та систем керування, при їхньому проектуванні характерно розбиття на відповідні окремі взаємодіючі частини.

Відповідно за системним підходом можна виділити три типи проектування роботів:

- структурний,
- блочно-ієрархічний,
- 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 машини 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 машин.

Як правило, проектування машин є багатокрокової процедурою, що включає в себе:

- етап науково-дослідних робіт (НДР);
- етап ескізного проекту або дослідно-конструкторських робіт (ДКР);

- етап технічного проекту;
- етап робочого проекту;
- етап випробувань дослідних зразків машин.

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

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

Загальна процедура проектування представлена на рис. 1.1.

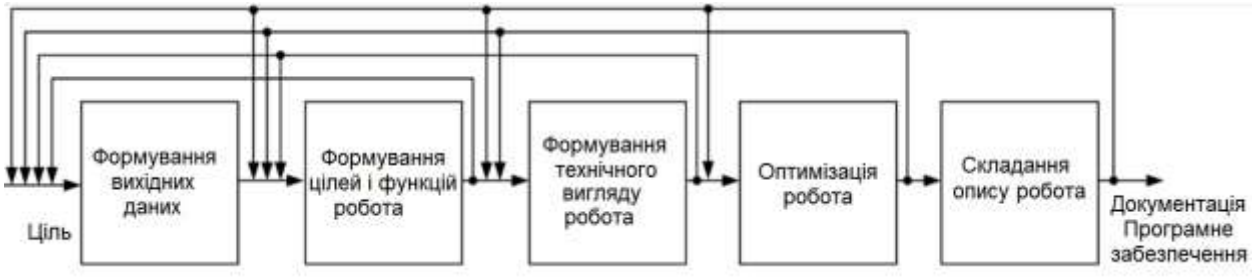


Рис. 1.1. Загальна процедура проектування

Формування вихідних даних полягає у визначенні кількісних показників цілей, а саме, тих параметрів, які повинен мати робот.

Після завершення формулювання цілей дається кількісний опис функцій, які повинен виконувати робот і його підсистеми. Ця процедура називається **формуванням цілей і функцій робота**. Функції і цілі можуть формулюватися в якісному і кількісному вигляді.

Наступним етапом є концептуальне проектування, в ході якого створюється **технічний вигляд проектованого робота**. Визначення вигляду полягає в формуванні найбільш прийнятної конфігурації, яка дозволить досягти необхідних цілей.

На наступному кроці здійснюється **вибір оптимальних параметрів робота** відповідно до технічного завдання.

Останнім етапом є **опис розробленого робота** - складання пояснювальної записки, оформлення програмного забезпечення, креслень, інструкцій тощо.

Потрібно відзначити, що процес проектування робота є інтерактивним. На кожному кроці по необхідності можна повернутися на один з попередніх етапів.

Важливою частиною проєктування є конструювання.

Конструюва́ння — процес створення конструктором проекту певного об'єкта техніки, що полягає у визначенні форми, розмірів, взаємного розташування й параметрів частин й елементів конструкції об'єкта, його складових (агрегатів, систем, вузлів тощо), способу їхнього з'єднання, вибору матеріалів окремих елементів та розробки конструкторської документації.

Конструювання передбачає свідоме та цілеспрямоване втілення винаходів, ідей та принципів, проведення розрахунків на основі знань з прикладних наук, виконання креслеників та інших дій, спрямованих на забезпечення заданих характеристик (якості) об'єкта, що конструюється.

Кінцевим результатом конструювання є технічний проект (найчастіше це комплект креслеників та іншої технічної документації) який повинен забезпечувати можливість виготовлення (створення), експлуатації та утилізації необхідного об'єкта.

1.2. Системи автоматизованого проектування робототехнічних систем

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,
- спеціалізовані роботи, 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 (рис. 1.2),
- **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 окремі типи роботів, що входять до її каталогу.

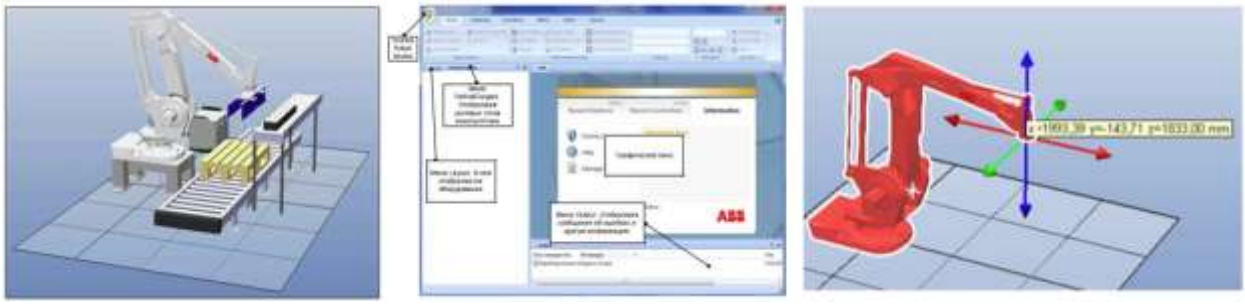


Рис. 1.2. Програмний комплекс **ABB robot studio**, призначений для проектування та моделювання роботів фірми ABB

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.

Ці засоби можуть складатися з 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** 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 баз даних, що використовуються при проектуванні;
- **лінгвістичні** (мови програмування і обміну даними);
- **методичні**, що включають різні методики проектування;

організаційні, які надаються 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 робототехнічних систем

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.1.3.

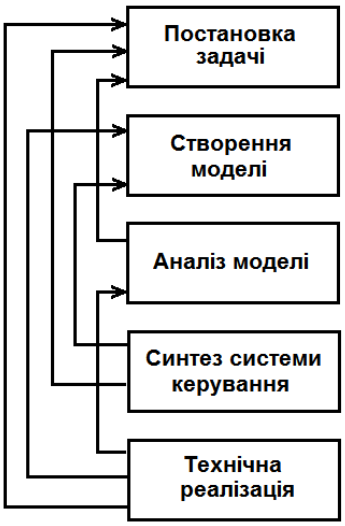


Рис. 1.3. Місце моделі і її аналіз в процедурі проектування

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.

Сучасні промислові машини та робототехнічні системи часто використовують апаратно-програмний комплекс Arduino, який має досить велику кількість мікроконтролерів, модулів керування приводами та інформаційні модулі, які дозволяють визначити стан як самої машини, так і зовнішнього середовища. Для проєктування та моделювання використовується емулятор UnoArduSim.

Прикладом засобу моделювання різних роботів є симулятор роботів V-REP/ CoppeliaSim компанії Coppelia Robotics, який має бібліотеку з великою кількістю моделей стаціонарних та мобільних роботів, а також додаткового обладнання та дозволяє зробити та перевірити комп’ютерну модель роботу (рис.1.4).

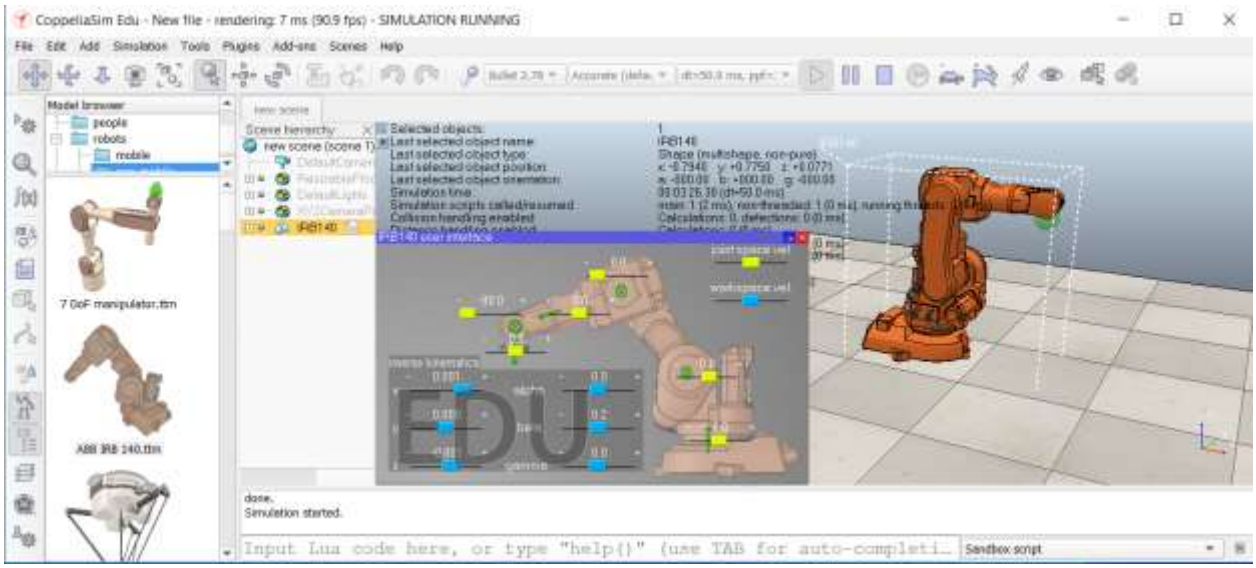


Рис. 1.4. Комп’ютерна модель робота у симуляторі роботів V-REP/CoppeliaSim

Симулятор V-REP/CoppeliaSim дозволяє також створити модель взаємодії робота з додатковим обладнанням (рис. 1.5).

Важливим елементом в системах керування сучасних машин є програмне забезпечення.

Програмне забезпечення складається з таких елементів, як системне та прикладне програмне забезпечення.

Системне програмне забезпечення здійснює функціонування пристрою керування і є його невід'ємною частиною.

Прикладне програне забезпечення здійснює керування машиною та потребую додаткового проєктування.

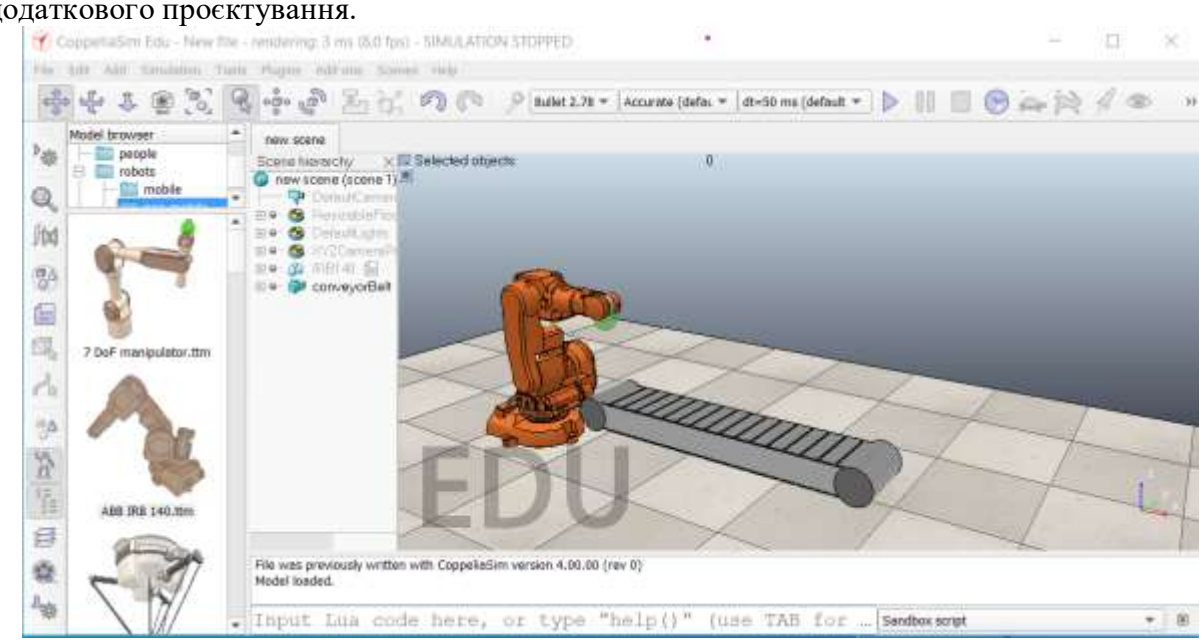


Рис. 1.5. Модель взаємодії робота з додатковим обладнанням

Розглянемо, прикладне програмне забезпечення, що використовують такі елементи, як пристрої керування машин, на прикладі промислових роботів, виходячи з узагальненої структури, наведеної на рис. 1.6. Ця структура включає систему переміщення як робочого органу, так і самого робота (наприклад, для мобільних або пересувних роботів).

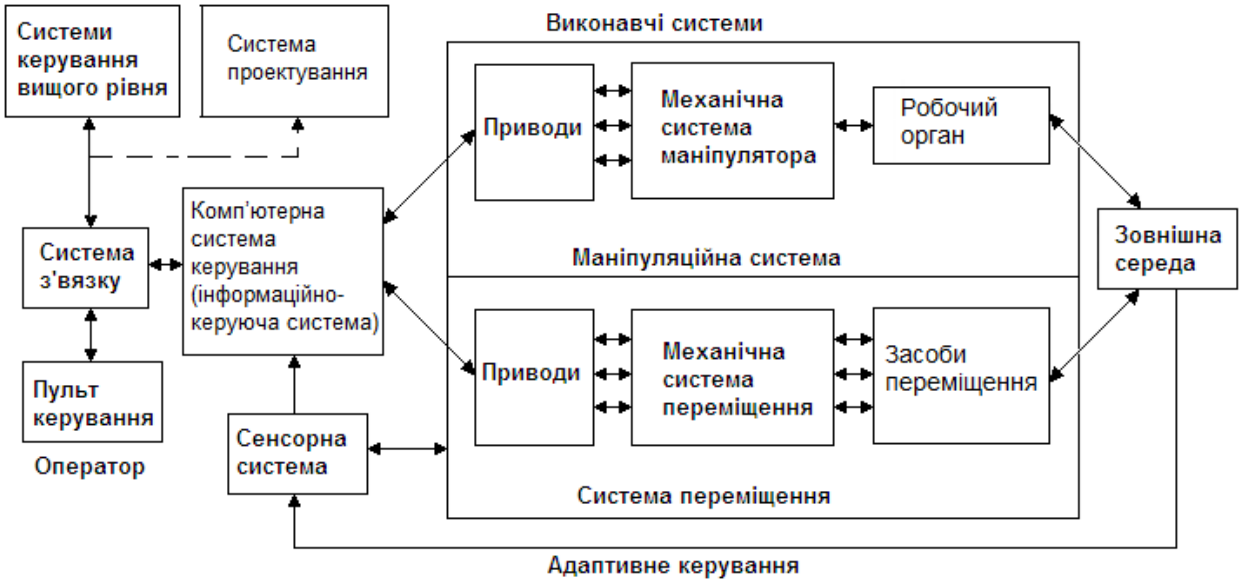


Рис. 1.6. Узагальнена структура промислових роботів

Очевидно, що алгоритму керування промисловим роботом здійснює комп'ютерна система, 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. Реалізація алгоритмів керування потребує створення програмних елементів

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 φ_1 … φ_6 (задача зворотної кінематики).

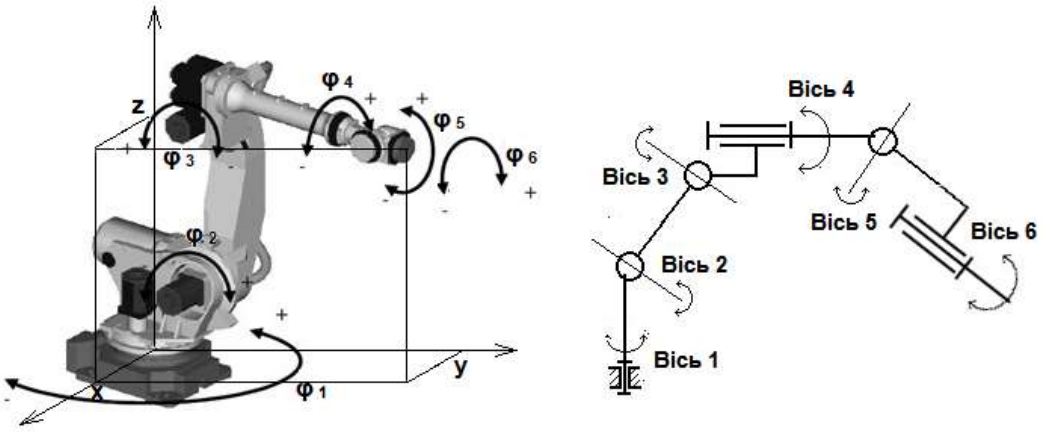


Рис. 1.7. Переміщення робочого органу в задану позицію

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).

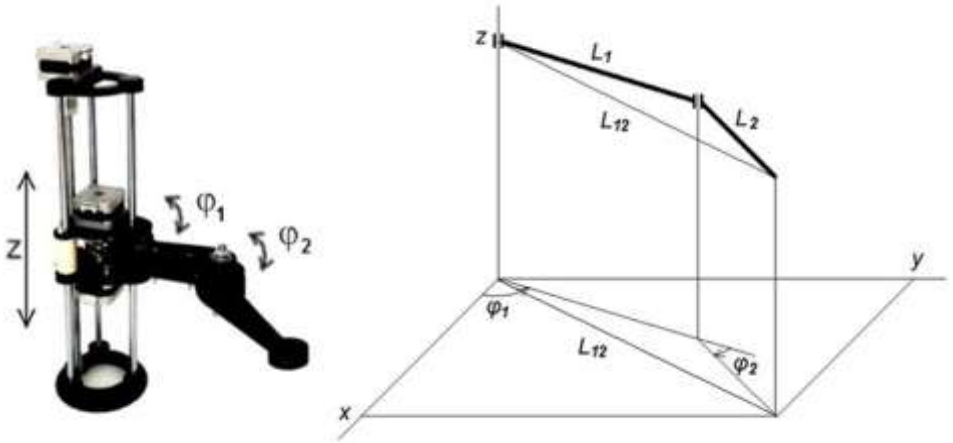


Рис. 1.8. Маніпулятор, для якого треба визначити кути повороту окремих ланок

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 $L_{12} = (x^2 + y^2)^{1/2}$.

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, що був отриманий на контролері за вказаними формулами для таких вихідних даних:

$L_1 = 100$ мм, $L_2 = 100$ мм,

$x = 100$ мм, $y = 50$ мм

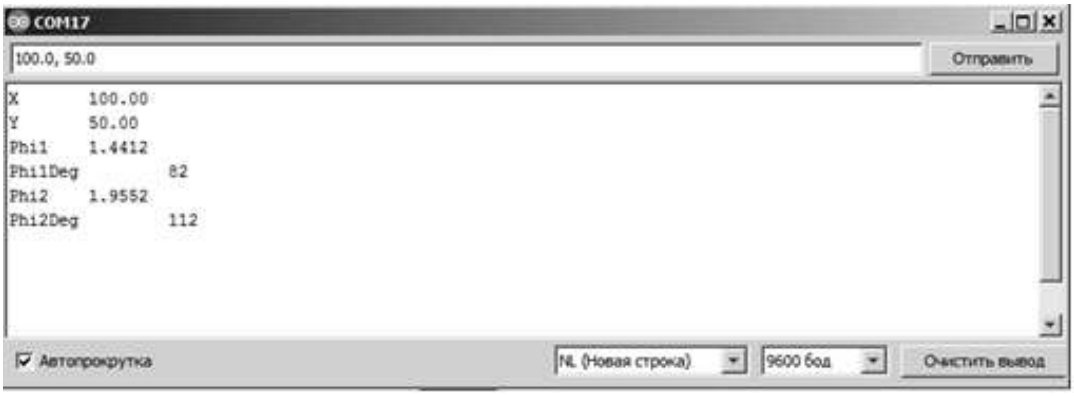


Рис. 1.9. Результат обчислення, що був отриманий на контролері

На рис. 1.10 наведене положення ланок маніпулятора відповідно отриманим результатам. Якщо поворот ланок L_1 та L_2 здійснюється за допомогою сервоприводів, то отримані результати можна безпосередньо використовувати для керування.

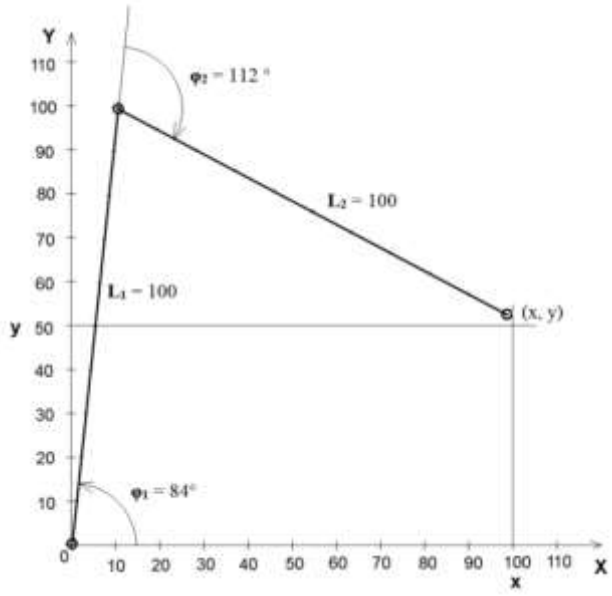


Рис. 1.10. Положення ланок маніпулятора відповідно отриманим результатам

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.

2. Основні засоби конструювання та проєктування роботів

2.1. Засоби конструювання та проєктування спеціалізованих роботів

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

Для проєктування механічних компонент робота найчастіше для цього використовують такі програмні комплекси проєктування, як **SolidWorks, MathCAD**, **Matlab** тощо.

Ці програмні комплекси детально розглядаються в дисциплінах по автоматизованому проєктуванню, 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 проєктування систем програмного керування можна використовувати різні засоби проєктування.

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 проєктування та налагодження.

В даний час широко використовуються мікроконтролери фірми Atmel, наприклад, 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 Сі, 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.

У складі апаратно-програмного комплексу Arduino є досить велика кількість сервоприводів, що представляють собою механізм з електромотором, який можна повернути в заданий кут і утримувати в цьому положенні. У складі сервопривода є умонтована система керування, яка виходячи з сигналу керування повертає та утримує вал на вказаному значенню кута.

Однокристальні мікроконтролери знайшли широке застосування для вирішення найрізноманітніших завдань управління, але, оскільки вони являють собою мікросхеми, то для проєктування 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 мікропроцесорних пристроїв керування на основі однокристальних мікроконтролерів у вигляді однієї друкованої плати з можливістю підключення додаткових модулів і здатних вирішувати велике коло завдань, в тому числі обробки даних, керування різними приладами, управління рухом, тощо (рис. 2.1).

Рис. 2.1. Мікропроцесорні пристрої керування

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.

Повністю відкрита архітектура системи дозволяє вільно копіювати або доповнювати пристрої 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.

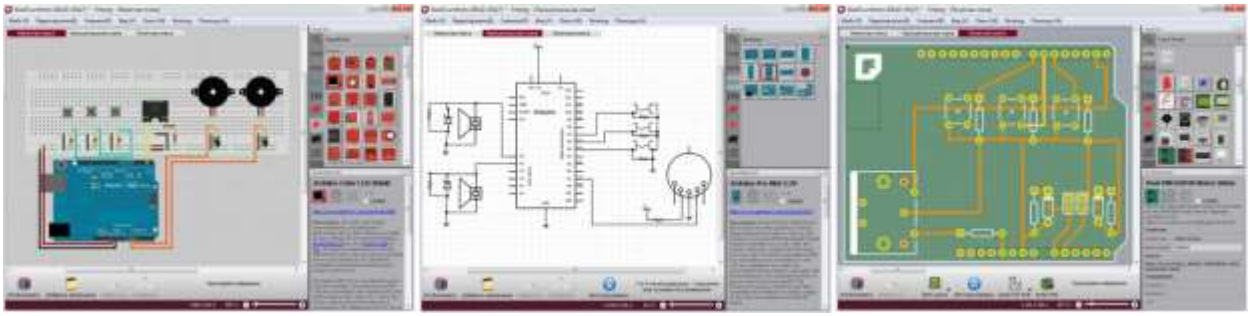
Програма складається з двох обов'язкових для Arduino функцій.

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 проєктуванні різних пристроїв, та спрощують програмування при використанні додаткових модулів, таких як датчики, виконавчі пристрої тощо.

Для проєктування контролерів Arduino використовується також використовується середа розробки проєктів Fritzing (рис. 2.2).



а) б) в)

Рис. 2.2. Інтерфейс користувача Fritzing

Fritzing дозволяє обрати елементи пристрою і подати їх у вигляді з'єднання макетів елементів (рис. 2.2, а), розробити його принципову схему (рис. 2.2, б).

Він також дає можливість розробити друковану плату для її подальшого виготовлення (рис. 2.2, в).

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

Програмування у середовищі ArduBlock здійснюється за допомогою значків, що відповідають окремим командам та функціям, а також змінним, які треба вибрати з каталогу. Створення програми здійснюється шляхом вибору усіх необхідних компонент і після завантаження в Arduino IDE має вигляд, наведений на рис. 2.3.

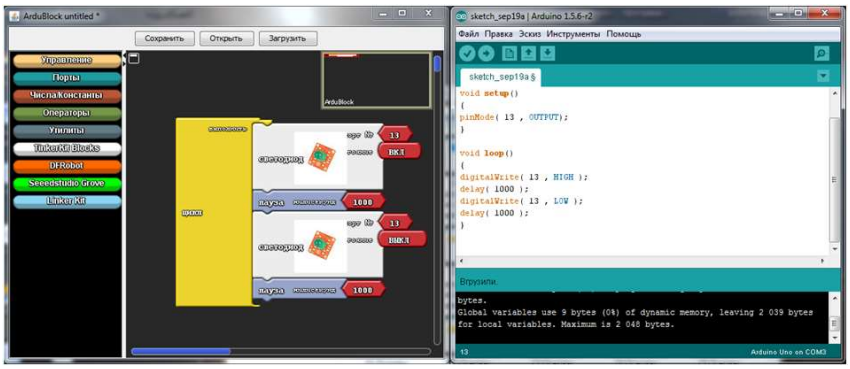


Рис. 2.3. Створення програми

Після цього програма може бути завантажена в контролер Arduino.

Для проєктування програмного забезпечення контролерів Arduino використовується також використовується емулятор UnoArduSim, призначений для створення, перевірки та налагодження програм Arduino і містить набір віртуальних пристроїв вводу/виводу ('I/O' Пристроїв), які можна налаштовувати і підключати до віртуального контролера Arduino.

Завдяки цьому UnoArduSim дає можливість комп'ютерного конструювання та проектування як апаратних, так і програмних компонентів різних промислових пристроїв.

На додатку UnoArduSim розташована лабораторна панель з різними інструментами розташованими по периметру, які можна налаштовувати і підключати до віртуального контролера Arduino (рис. 2.4).

Всі ці інструменти можна додавати, вказавши потрібну кількість у вікні **Конфигурировать – 'I/O' Устройства**.

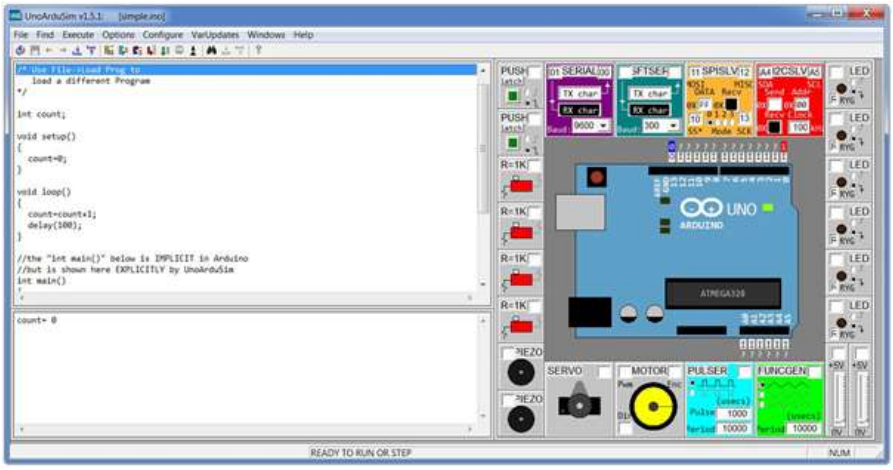


Рис. 2.4. Додаток UnoArduSim

На рис. 2.5 наведена модель робота з чотирма сервоприводами, розроблена за допомогою UnoArduSim.

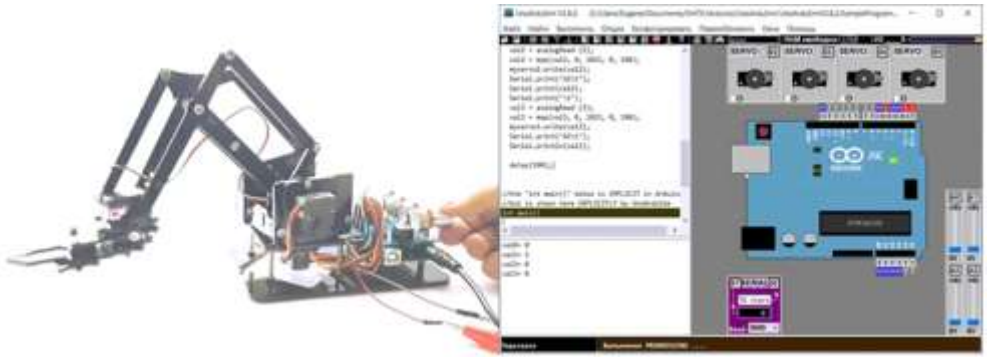


Рис. 2.5. Модель робота з чотирма сервоприводами

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.

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 проєктування комплексних систем керування включають засоби проєктування окремих компонент.

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

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

Складання проєкту системи керування робиться за допомогою усіх цих засобів.

У програмному комплексі STEP7 складання проєкту здійснюється в керуючій програмі **SIMATIC-Manager**.

Визначення структури і складу системи керування робиться за допомогою конфігуратора **HW Config**, який є складовою частиною програмного комплексу.

Проєктування здійснюється шляхом вибору відповідних модулів з каталогу.

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

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.

Однією з важливіших задач при створенні систем керування є її налагодження та пошук помилок як на стадії проєктування 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.

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.

До симулятора можна додати додаткові пристрої, наприклад, конвеєр та перевірити виконання програми керування (рис. 2.6).

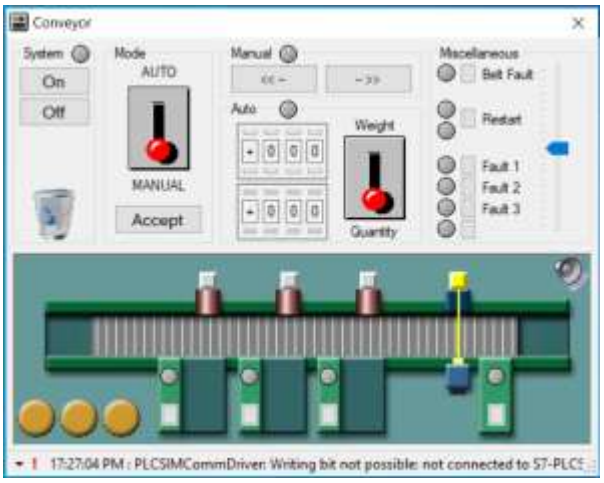


Рис. 2.6. Модель конвеєра

2.2. Засоби конструювання та проєктування універсальних роботів

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

програмний комплекс ABB robot studio,

програмний комплекс фірми KUKA,

Microsoft Robotics Developer Studio.

Програмний комплекс **ABB robot studio**, призначений для проєктування та моделювання роботів фірми ABB.

RobotStudio представляє собою симуляційну середу off-line програмування роботів компанії ABB.

Основною перевагою off-line програмування є відсутність необхідності в наявності реального обладнання.

Переміщення проєкту 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.

Програмний комплекс **ABB robot studio** дає можливість проєктування та моделювання робототехнічних комплексів.

П**рограмний комплекс фірми KUKA**, призначений для проєктування та моделювання роботів фірми KUKA.

Комплекс має декілька компонент (рис. 2.7), наприклад, такі як: віртуальний контролер роботів KUKA.OfficeLite (ліворуч), та симулятор KUKA.Sim (праворуч).

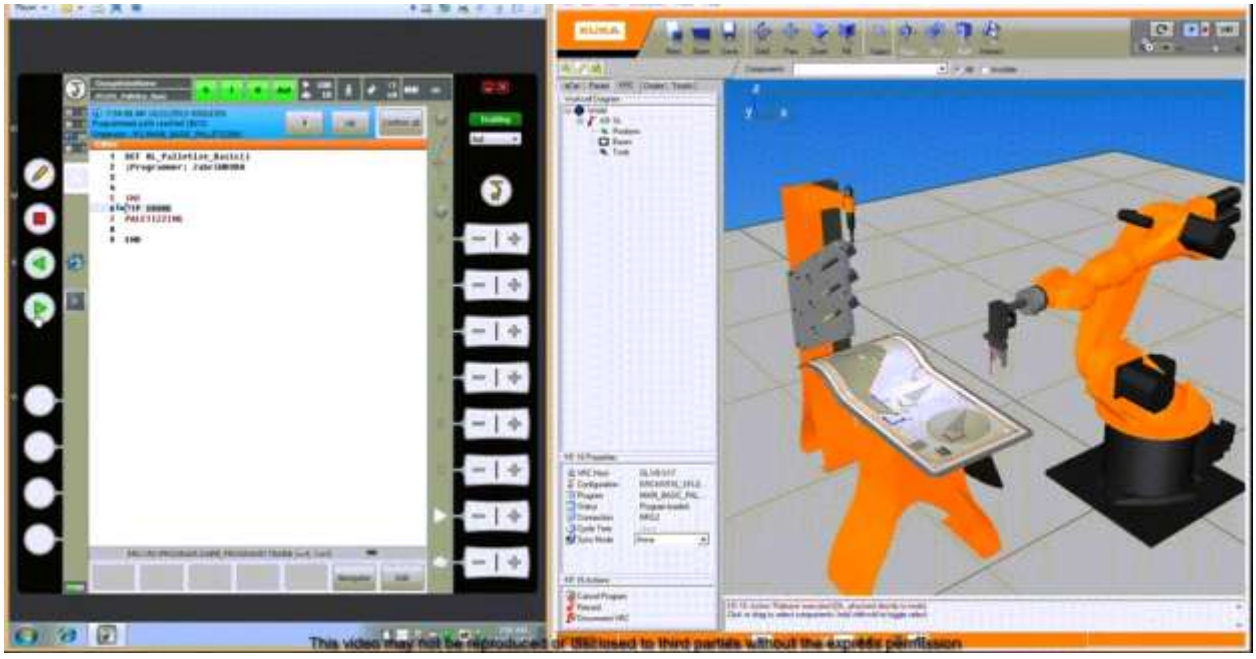


Рис. 2.7. Програмний комплекс фірми KUKA

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 проєкту. 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.

Симулятор V-REP/ CoppeliaSim

Симулятор V-REP/CoppeliaSim, що дозволяє здійснювати комп'ютерне конструювання та проектування робототехнічних комплексів з використанням принципів універсальності та масштабованості середовища моделювання (рис. 2.8).

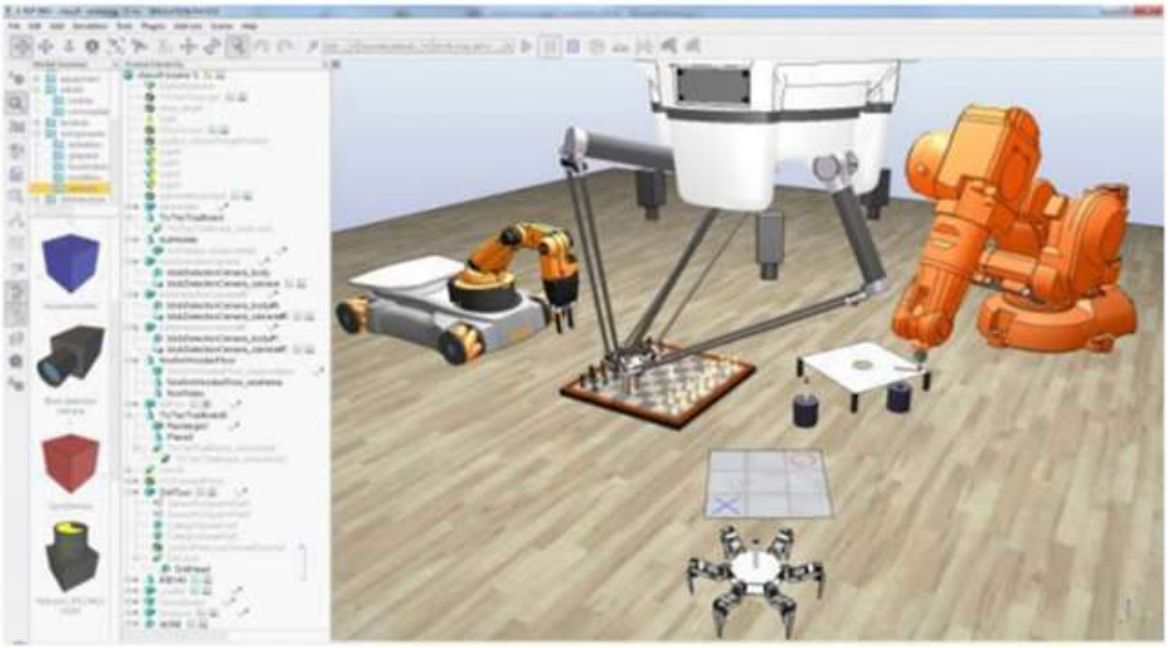


Рис. 2.8. Симулятор V-REP/CoppeliaSim

2.3. Засоби конструювання та проєктування мобільних роботів

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

Прикладом програмних комплексів для проєктування мобільних роботів є **Microsoft Robotics Developer Studio**, що представляє собою Windows-орієнтоване

середовище для керування роботами і їх симуляції та дозволяє проєктувати окремі типи роботів, що входять до її каталогу.

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 інтегруються в середу розробки Visual Studio, який має два основних модуля:

- 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.

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.

В даний час Microsoft RDS використовується в основному для проєктування побутових роботів, наприклад, роботів пилососів.

Іншим прикладом комплексів проєктування мобільних роботів є програмний комплекс **Dyn-Soft RobSim 5**.

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 схемі та підключення.

Наприкінці проєктування проводиться моделювання робота.

Контрольні питання

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

3. Засоби конструювання та проектування спеціалізованих робототехнічних пристроїв

3.1. Проектування спеціалізованих робототехнічних пристроїв

Спеціалізовані роботи можуть бути створені для виконання обмежених функцій, або бути складовими частинами комплексних виробничих систем.

Такі роботи часто мають циклове керування з обмеженими переміщеннями.

Керування комплексними системами найчастіше здійснюється за допомогою децентралізованої системи керування, де кожна одиниця обладнання може мати свій пристрій керування, керування усією системою здійснює окремий комп'ютерний пристрій.

Використання спеціалізованих робототехнічних пристроїв у складі виробничих систем виправдано тоді, коли вони виконують відносно прості функції, завдяки чому їх вартість може бути значно меншою вартості універсальних роботів.

Прикладом таких роботів можуть бути роботи, які здійснюють переміщення вантажу у визначені позиції, наприклад, переміщення з виробничої ланки на конвеєр, або переміщення з конвеєра у місце зберігання.

В Інтернеті можна знайти досить прості робототехнічні пристрої, які мають модульну структуру, що дозволяє створити на їх основі відносно прості автономні роботи, що здатні працювати під керуванням зовнішньої системи керування, наприклад, маніпулятори uArm - відкритий проєкт маніпулятора під керуванням Arduino.

До робототехнічних пристроїв можна також віднести 3D-принтери, що знайшли досить широке використання.

Системи керування таких пристроїв також часто робляться з використанням контролерів Arduino.

На рис.3.1 показані компоненти 3D-принтера, що складаються в основному з крокових двигунів та пристрою керування.

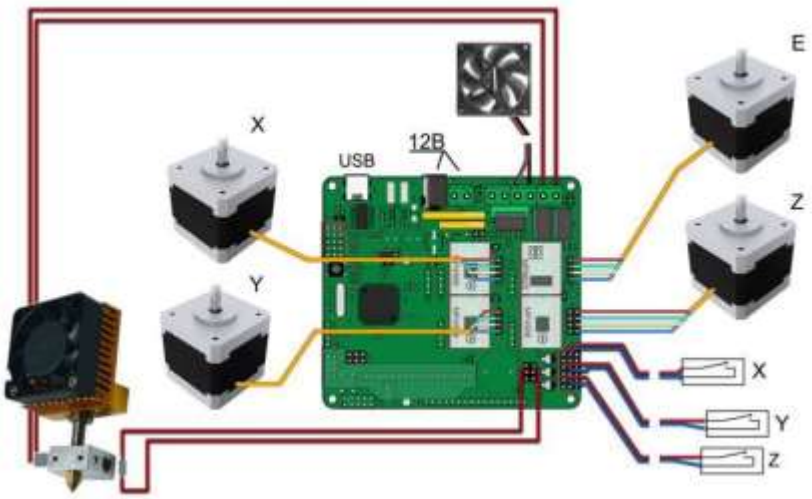


Рис. 3.1. Компоненти 3D-принтера

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

Програмування однокристальних мікроконтролерів зазвичай здійснюється з використанням таких мов, як Асемблер або С, хоча існують компілятори для інших мов.

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 мови програмування, які використовують символьні позначення.

Для програмування простих мікропроцесорних пристроїв часто використовується мова програмування Асемблер, команди якої відповідають командам центрального процесора.

Такий язик є машинноорієнтованим язиком, оскільки набір його команд залежить від типу процесора. Тому кожний процесор має свою мову Асемблера. Програми на Асемблері є найбільш компактними, але програмування на Асемблері значно складніше через обмежені можливості.

Засоби проектування систем керування для роботів на основі мікроконтролерів найчастіше використовують мову С, яка на відміну від Асемблера, є універсальною мовою. Мова С була розроблена з метою написання нею операційної системи UNIX, тому найчастіше використовується для створення системного програмного забезпечення, хоча зараз вона досить часто використовується для написання прикладного програмного забезпечення.

В останній час для вирішування задач керування промисловими машинами найчастіше використують мікроконтролери AVR фірми Atmel,

Система команд і архітектура ядра AVR розроблялися спільно з фірмою-розробником компіляторів з мов програмування високого рівня IAR Systems Ltd, тому структура мікроконтролерів AVR максимально оптимізована для того, щоб писати програми на мовах високого рівня.

Саме тому програми на мові С для мікроконтролерів AVR незначно втрачають в продуктивності у порівнянні з програмами, написаними на мові Асемблера.

3.2. Апаратно-програмний комплекс Arduino

Апаратно-програмний комплекс Arduino, який представляє собою набір апаратних та програмних засобів, пристосованих для побудови нескладних систем промислової автоматики і робототехніки.

Програмна частина складається з безкоштовної програмної оболонки для написання програм, з використанням спрощеної версії мови С, їх компіляції і програмування апаратури.

Апаратна частина являє собою набір готових пристроїв.

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

Інформація про компоненти апаратно-програмного комплексу Arduino доступна на сайті https://arduino.ua/.

Безкоштовна середа розробки Arduino IDE, що доступна на сайті https://www.arduino.cc/en/software дозволяє створити та завантажити програми для контролерів Arduino.

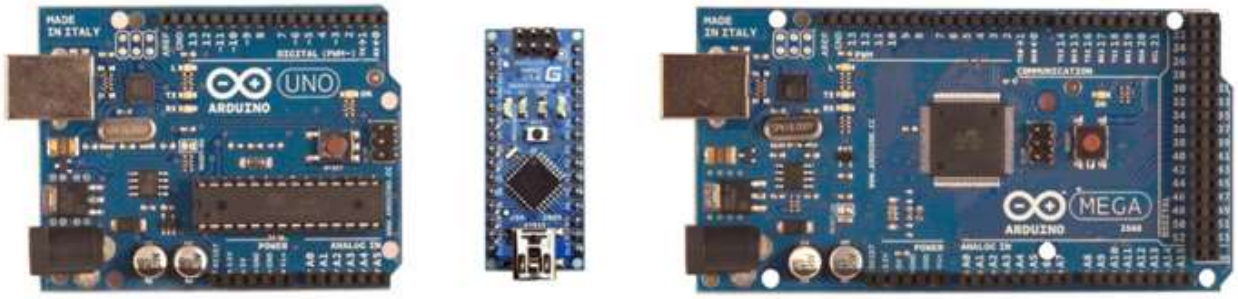
Контролери Arduino.

Контролер Arduino Uno, побудований на ATmega328. Платформа має 14 цифрових входів / виходів (6 з яких можуть використовуватися як виходи ШІМ), 6 аналогових входів, кварцовий генератор 16 МГц, роз'єм USB, і кнопку скидання. Розмір плати 6,9 × 5,3 см (рис. 3.1, а).

Контролер Arduino Nano, побудований на ATmega328. Платформа має 14 цифрових вхід / виходів (6 з яких можуть використовуватися як виходи ШІМ), 8 аналогових входів, кварцовий генератор 16 МГц, роз'єм Mini USB. Відмінною особливістю є малі розміри (1.85 × 4.2 см), що дозволяють вбудовувати контролер в портативні пристрої (рис. 3.1, 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 UART, кварцовий генератор 16 МГц, роз'єм USB,

роз'єм живлення, роз'єм ICSP для внутрисхемного програмування і кнопка скидання. Розмір плати 10,8 × 5,3 см (рис. 3.1, в).



а) б) в)

Рис. 3.1. Контролери Arduino: Arduino Uno (а), Arduino Nano (б), Arduino Mega (в)

Модулі управління електродвигунами

Драйвер двигуна L298N (рис. 3.2). Модуль виконаний на основі мікросхеми L298N, яка представляє собою здвоєний мостовий драйвер двигунів і призначена для управління двома двигунами постійного струму або одним кроковим двигуном. Модуль забезпечує напругу живлення двигунів до 35 В та струм на одному каналі до 2 А.

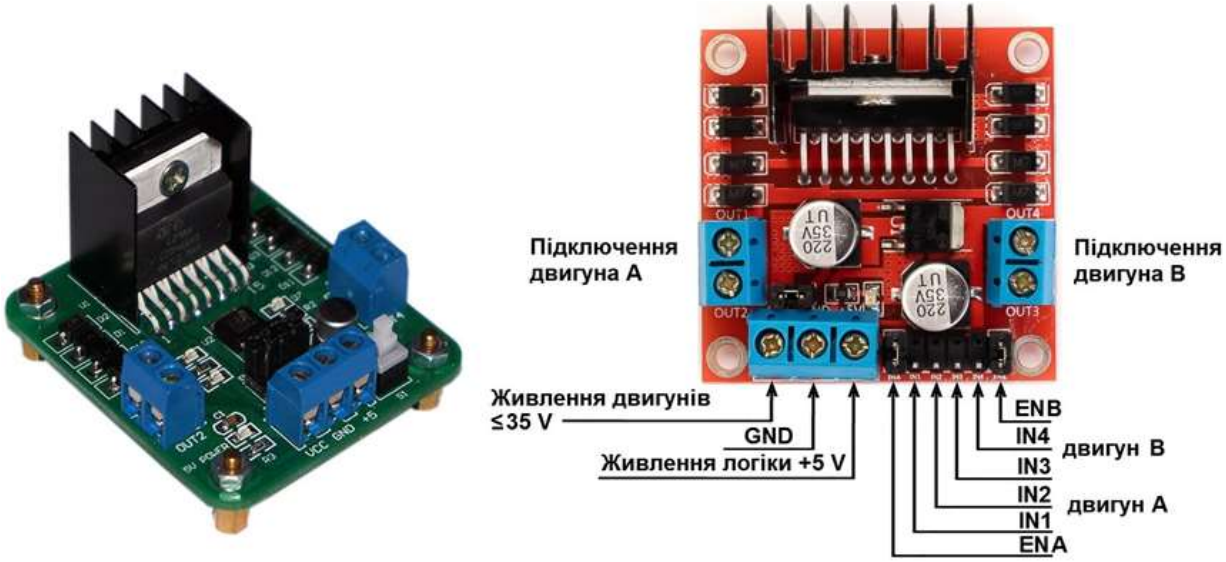


Рис. 3.2. Драйвер двигуна L298N

Драйвер двигуна Motor Shield L293D. (рис. 3.3). Модуль виконаний на основі мікросхеми L293D, яка представляє собою чотири мостових драйверів двигунів і призначена для управління чотирма двигунами постійного струму або двома кроковими двигунами.

Крім того є окремих вихід для підключення серводвигуна.

На рис. 3.3 наведені зовнішній вигляд модуля та підключення двигунів:

- зовнішній вигляд Arduino Uno та Motor Shield L293D (рис. 3.1, а),
- підключення серводвигуна (рис. 3.1, б),
- підключення двигунів постійного струму (рис. 3.1, в),
- підключення крокових двигунів (рис. 3.1, г).

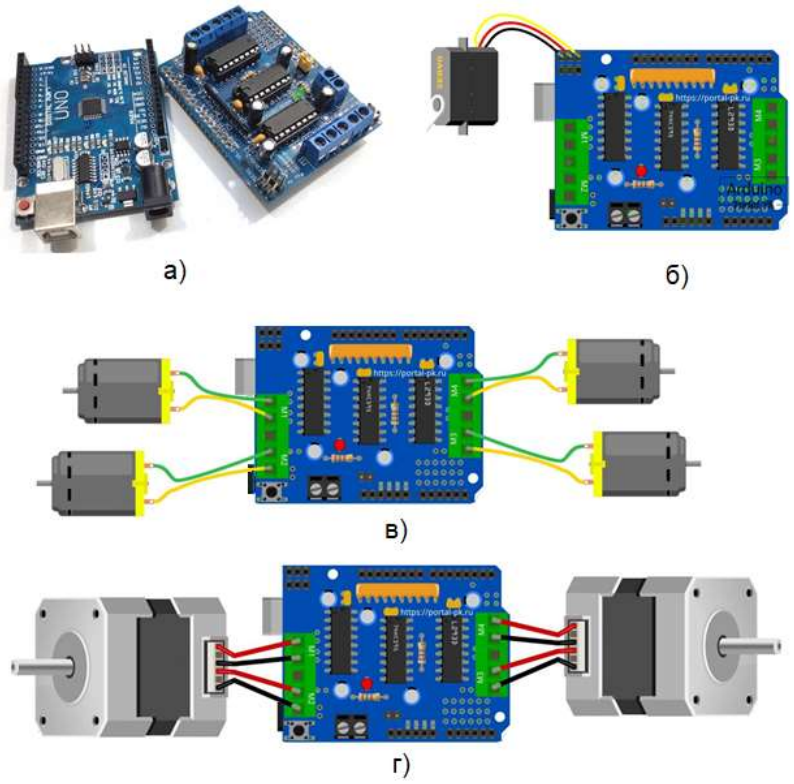


Рис. 3.3. Зовнішній вигляд модуля та підключення двигунів

Принцип роботи модулів керування двигунами постійного струму

Для керування двигуном постійного струму з урахуванням напрямку та швидкості обертання використовують так званий Н-міст.

На рис. 3.4 наведена схема, що представляє собою Н-міст на IGBT-транзисторах з захисними діодами.

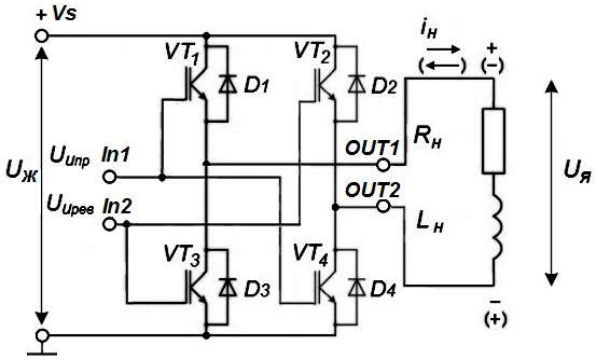


Рис. 3.4. Н-міст на IGBT-транзисторах з захисними діодами

На рис. 3.5 наведений принцип роботи Н-мосту з відповідними командами керування за допомогою контролера Arduino.

Команда digitalWrite(pin, LOW) видє) на вихід pin,

Команда analogWrite(pin, 200) видає на цей вихід сигнал з широтно-імпульсною модуляцією, що має значення від 0 до 255.

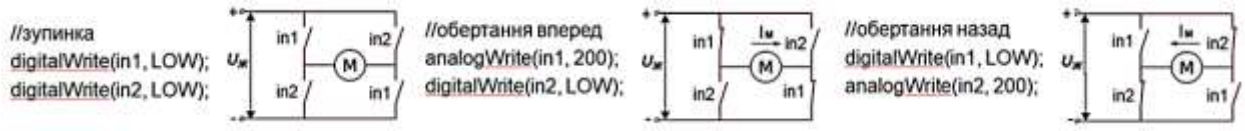


Рис. 3.5. Принцип роботи Н-мосту з відповідними командами керування

Сервоприводи

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 (рис. 3.6).

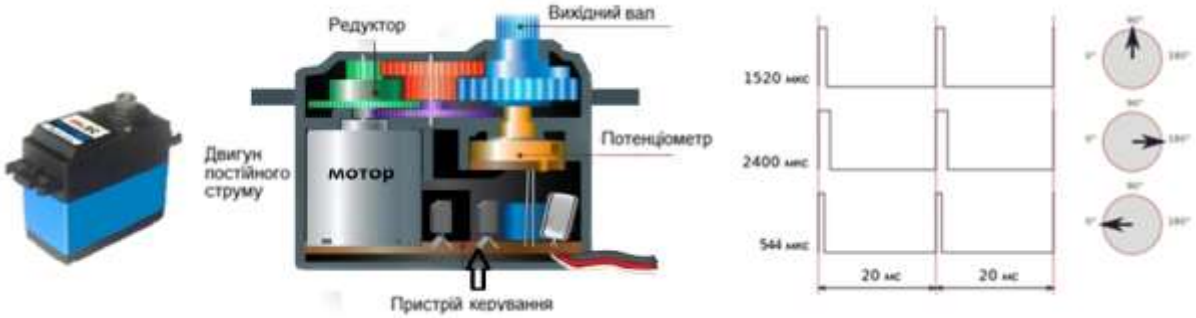


Рис. 3.6. Сервопривод

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

Існує досить велика кількість сервоприводів, які забезпечують крутний момент до 20 кг/см, що дає можливість використовувати їх для керування невеликими маніпуляторами.

На рис. 3.7 наведений 6-вісний маніпулятор на основі сервоприводів (з урахуванням захоплюючого пристрою), зроблений з використанням конструкторського набору для робототехнічних пристроїв. Маніпулятор використовує сервоприводи MG 996R, які мають такі характеристики: кут повороту від 0 до 180°; швидкість обертання: без навантаження при 4.8 В - 0.17 с / 60°, без навантаження при 6 В - 0.13 с / 60°; крутний момент: при 4.8 В - 9 кг/см, при 6 В - 12 кг/см.

Рис. 3.7. 6-вісний маніпулятор на основі сервоприводів та конструкторського набору для робототехнічних пристроїв

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. 3.8).

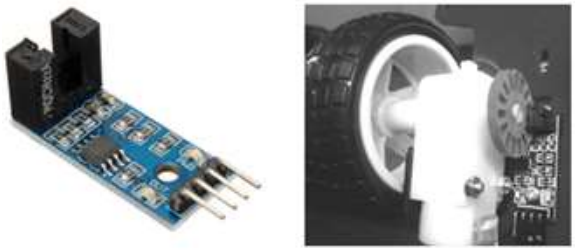


Рис. 3.8. Щілинний оптичний датчик FC-03

Для визначення відстані до об’єктів використовують ультразвукові та інфрачервоні датчики.

Так ультразвуковий датчик вимірювання відстані HC-SR04 (рис. 3.9, а) дозволяє визначати відстань до об'єкта від 2 см до 4,5 м з точністю до 0,3 см.

Інфрачервоний датчик відстані Sharp GP2Y0A02YK0F (рис. 3.9, б) з аналоговим виходом дозволяє визначати відстань до об'єкта від 20 см до 150 см.

а) б)

Рис. 3.8. Ультразвуковий (а) та інфрачервоний (б) дитчики

Для орієнтації в просторі використовуються модулі акселерометра і гіроскопа по трьох осях координат, магнітометр, також датчик кута нахилу, які використовують балансуючі роботи та дрони (рис. 3.9).

Рис. 3.9. Модулі акселерометра і гіроскопа

3.3. Засоби конструювання на основі апаратно-програмного комплексу Arduino

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 для різних варіантів застосування.

Для програмування контролерів Arduino на мові С, використовується безкоштовна середа розробки Arduino IDE, що дозволяє створити та завантажити програми для контролерів Arduino.

Середу розробки Arduino IDE можна завантажити з офійійного сайту:

https://www.arduino.cc/en/software

Програму Arduino називають скетч, який 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, один раз виконується блок setup(), де здійснюється встановлення вихідних даних, ініціалізація значень змінних, а також налаштування портів периферії Arduino та інші налаштування.

Після закінчення обробки блоку setup() Arduino починає циклічне виконання інструкцій в блоці loop().

Після виконання всіх операцій, цикл повторюється знову і знову.

Наведений приклад програми лічильника:

```
int count;        //визначити змінну лічильника як ціле число
void setup()
{
count=0;          //встановити початкове значення лічильника 0
}
void loop()
{
count=count+1; //додати до лічильника 1
delay(100);      //затримка 0,1 с
}
```

Після запуску середи розробки Arduino IDE відкривається вікно, яке використовується для редагування програми та має вигляд, наведений на рис. 3.10.

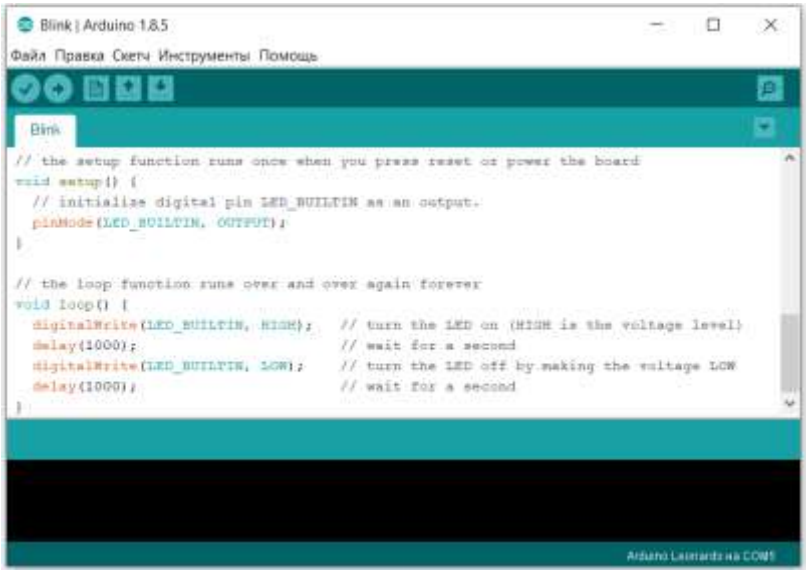


Рис. 3.10.Вікно редагування програми ереди розробки Arduino IDE

Для спрощення процесу програмування використовують засоби графічного проектування програмного забезпечення

Прикладом графічного проектування програмного забезпечення є середа розробки проектів Fritzing (рис. 3.11), яка згадувалась раніше, а також графічне середовище програмування ArduBlock, що вбудовується в середу програмування Arduino IDE.

Програмний пакет Fritzing може стати в нагоді в таких стадіях розробки, як розробка прототипу схеми на макетній платі, а також автоматичне генерування необхідної схеми та друкованої плати.

Fritzing створювалася для Arduino, апаратно-програмної платформи, що складається із звичайної плати з мікроконтролером Atmel AVR, радіодеталей для програмування, інтерфейсів зв'язку, середовища розробки Processing/Wiring.

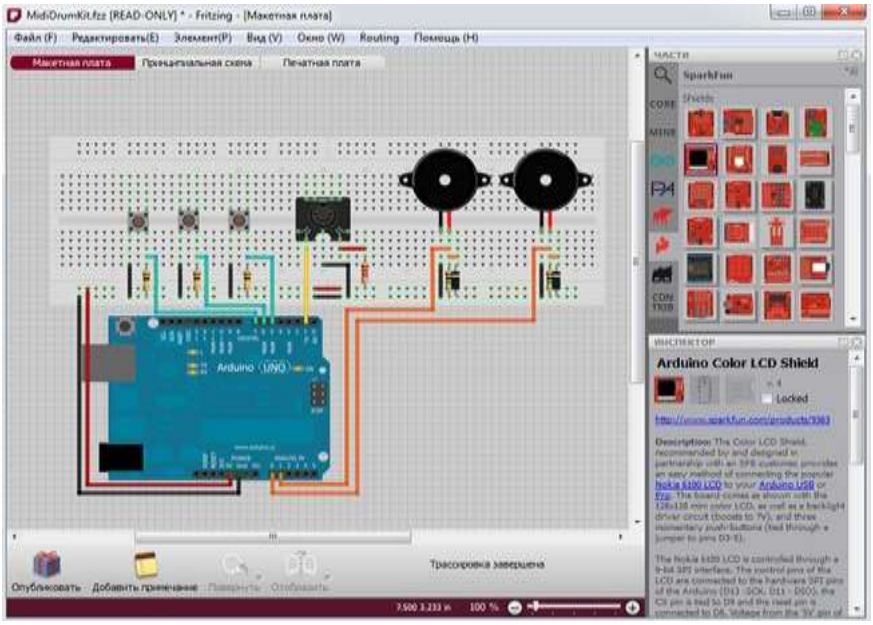
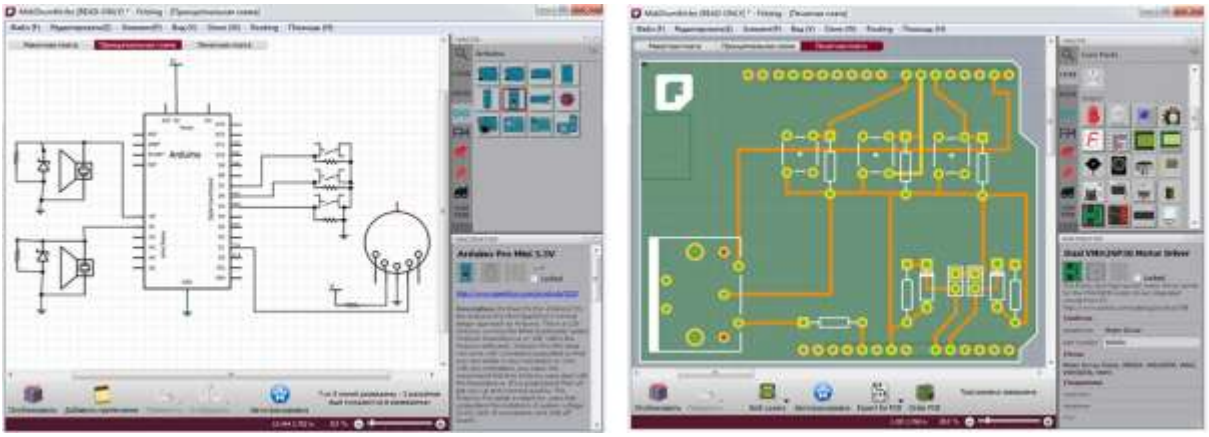


Рис. 3.11. Вікно Fritzing «Макетна плата»

Робота з новим проектом у пакеті Fritzing починається з вибору готових компонентів, повний перелік яких розташований у верхньому куті робочого вікна з правого боку. Тут можна знайти різні макетні та монтажні плати (у тому числі Arduino), цілий набір аналогових та цифрових мікросхем, будь-які радіодеталі: конденсатори, транзистори, резистори, світлодіоди, батарейки, кнопки та інші.

Схема доступна для малювання як у вікні «Макетна плата» так і у вікні «Принципова схема» (рис. 3.12, а) простим перетягуванням потрібних компонентів на робоче поле. При виборі вікна «Друкована плата» можна розпочати розведення провідників та розміщення елементів (рис. 3.12, б).



а) б)

Рис. 3.12. Вікна «Принципова схема» (а) та «Друкована плата» (б)

Середа розробки проектів Fritzing призначена для розробки апаратних компонентів системи керування.

Для програмування потрібні додаткові засоби, наприклад, ArduBlock.

Для цього треба завантажити ArduBlok з офіційного сайту за посиланням:

https://sourceforge-net.translate.goog/projects/ardublock/files/?_x_tr_sl=ru&_x_tr_tl=uk&_x_tr_hl=uk&_x_tr_pto=sc

Для початківців рекомендується використувати версію від 2013-07-12, саме цей файл є найбільш популярним.

Потім завантажений файл перейменовуємо в ardublock-all і в папці «документи» створюють папки: Arduino > tools > ArduBlockTool > tool і в останню переміщаємо файл ardublock-all.

Для того щоб працювати в ArduBlok, необхідно запустити Arduino IDE. Після чого заходимо у вкладку Інструменти і там знаходимо пункт ArduBlok, та натискаємо на нього (рис. 3.13).

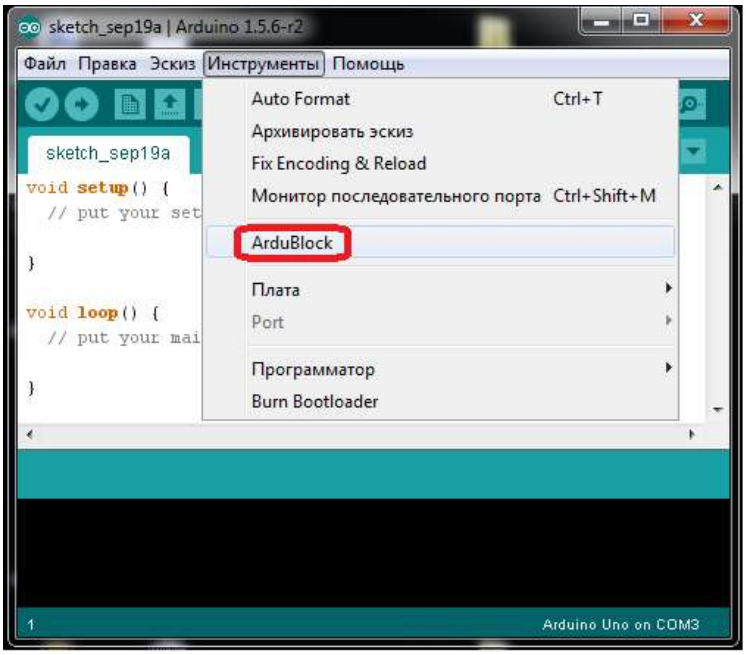


Рис. 3.13. Додаток ArduBlok

Програмування у середовищі ArduBlock здійснюється за допомогою значків, що відповідають окремим командам та функціям, а також змінним та константам, які треба вибрати з каталогу, який має декілька розділів (рис. 3.14). У розділі «Control (Управління)» можна знайти різноманітні цикли. В розділі «Pins (Порти)» можна керувати значеннями портів, а також підключеними до них датчиками та виконавчими пристроями. В розділі «Number/Constants (Числа/Константи)» можна вибрати цифрові значення або створити змінну. В розділі «Operators (Оператори)» можа знайти всі необхідні оператори порівняння та обчислення/

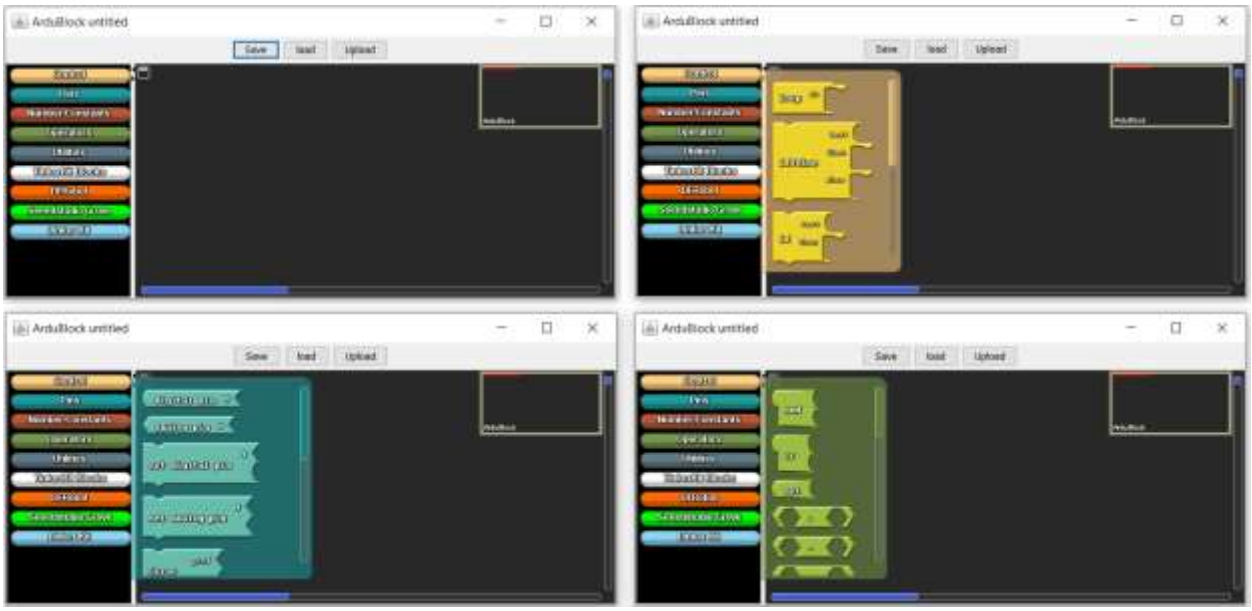


Рис. 3.14. Середовище ArduBlock

Програмування здійснюється шляхом вибору усіх необхідних компонент і після завантаження в Arduino IDE має вигляд, наведений на рис. 3.15.

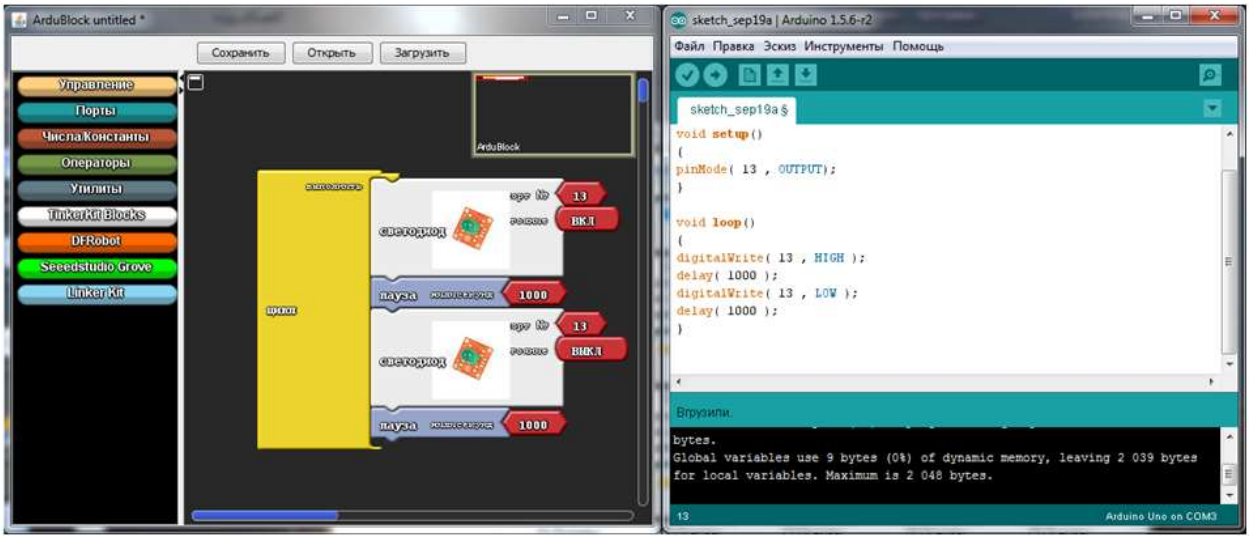


Рис. 3.15. Вигляд програми в ArduBlock

Після цього програма може бути завантажена в контролер Arduino. Недоліком ArduBlock є неможливість емуляції контролерів Arduino. Засоби, що дають таку можливість розглянемо далі

Контрольні питання

1. Визначити, для чого найчастіше використовують спеціалізовані роботи?
2. Описати, чому 3D-принтери можна віднести до спеціалізованих роботів?
3. Назвати, які мови програмування найчастіше використовують для контролерів, що використовують для керування спеціалізованих робототехнічних пристроїв?
4. Розповісти, які особливості має мова Асемблер?
5. Визначити, які особливості має апаратно-програмний комплекс Arduino?
6. Описати, які особливості мають додаткові модулі, що входять у склад апаратно-програмного комплексу Arduino?
7. Розповісти, які особливості має мова С для програмування контролерів Arduino?
8. Визначити, які задачі вирішує програмний пакет Fritzing?
9. Розповісти, які задачі вирішує середовище програмування ArduBlock?
10. Описати, як здійснюється програмування за допомогою середовища програмування ArduBlock?

4. Конструювання та моделювання машин на основі контролера Arduino за допомогою емулятора UnoArduSim

4.1. UnoArduSim емулятор контролерів Arduino

Додаток UnoArduSim є безкоштовним емулятором (симулятором) контролера Arduino, який дає можливість здійснити виконання програми в реальному часі без наявності самої плати Arduino (рис. 4.1).

Рис. 4.1. Додаток UnoArduSim

При цьому є можливість перегляду ходу виконання програми.

UnoArduSim призначений для полегшення налагодження Arduino програм і містить набір віртуальних пристроїв вводу / виводу ('I / O' Пристроїв), які можна налаштовувати і підключати до віртуального Arduino.

UnoArduSim не потребує установки. Його треба завантажити за посиланням:

https://www.sites.google.com/site/unoardusim/simulator-download?authuser=0 ,

Після чого його треба розархівувати, запустити файл UnoArduSim.exe і працювати з ним.

Симулятор працює в операційній системі Windows починаючи з Windows XP

Після завантаження UnoArduSim для спрощення запуску можна створити ярлик на робочому столі

Після запуску програми з'являється вихідне вікно. (рис. 4.2).

Рис. 4.2. Вихідне вікно додатку UnoArduSim

Зліва наведений приклад програми лічильника. Справа показані контролер Arduino Uno та пристрої, які можна використовувати для проєктування

Подвійним кліком зони, де наведена програма, запускається вікно редагування програм Arduino (Edit / View Program), яке наведено на рис 4.3.

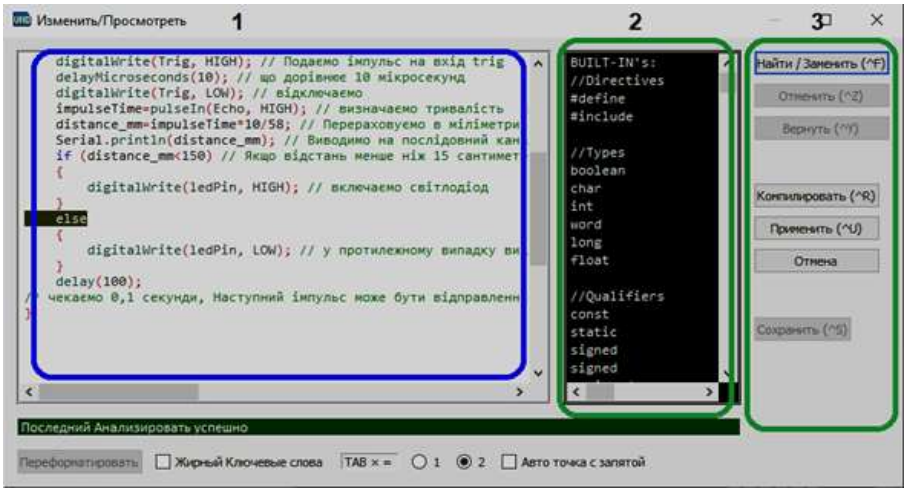


Рис. 4.3. Вікно редагування програм Arduino

Зона 1 призначена для внесення та редагування Arduino програм (скетчів). Вона містить невеликий код для емуляції скетчу, цей код можна видалити, потім він додається автоматично. Тут можна вставити свій код або скопійований з програми Arduino IDE.

Права кнопка мишки тут не працює, так що копіювати і вставляти доводиться через команди Control C і Control V.

Зона 2 містить каталог мови Ардуіно, в якому наведені команди та функції, який працює наступним чином – треба встановити курсор в потрібному місці в зоні 1 і двічі клікнути на потрібну функцію в зоні 2.

Зона 3 включає кнопки:

Найти / Заменить (^ H) - для виклику пошуку у програмі,

Отменить (^ Z) **Вернуть** (^ Y) - крок назад чи крок вперед,

Компилировать (^ R) - скомпілювати (перевірити на наявність помилок),

Применить (^ U) - прийняти,

Отмена - скасування,

Сохранить (^ S) – зберегти.

На рис. 4.4 наведене вікно змінних. Тут під час емуляції можна спостерігати стан і значення всіх змінних, які містяться в скетчі.

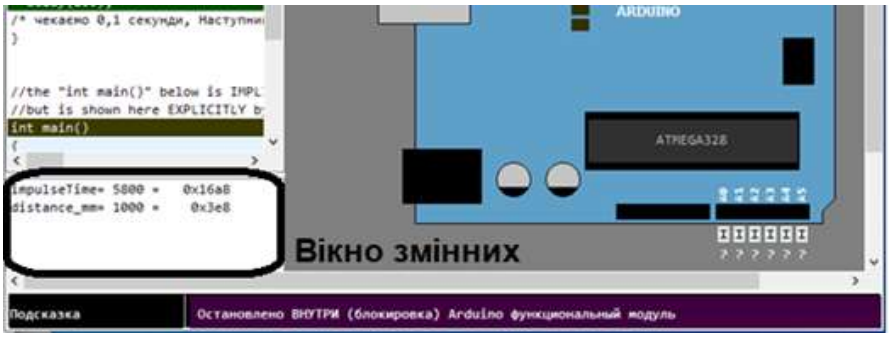


Рис. 4.4. Вікно змінних

Скорочену та повну інформацію про UnoArduSim можна знайти у закладці **Помощь - Быстрый помощь**, «**Полный помощь**» (рис. 4.5).

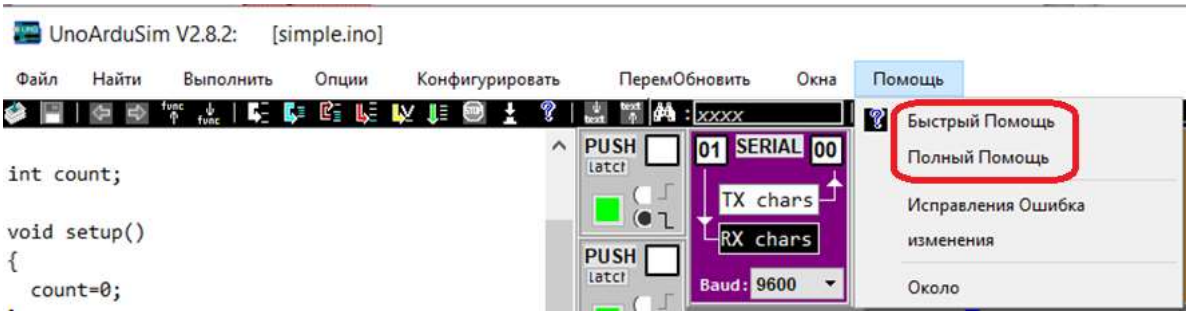


Рис. 4.5. Закладка Помощь - Быстрый помощь, «Полный помощь»

Справа на додатку UnoArduSim в запущеному стані розташована лабораторна панель з різними інструментами розташованими по периметру, які можна налаштовувати і підключати до віртуального Arduino UNO (рис. 4.6).

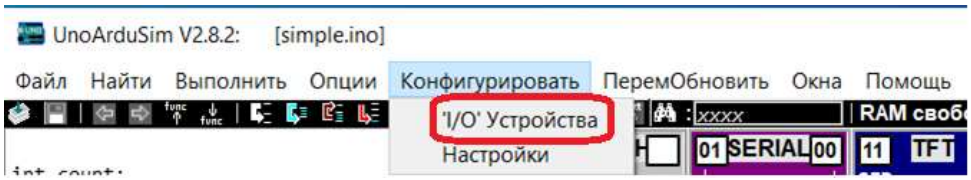


Рис. 4.6. Лабораторна панель

Всі ці інструменти можна додавати, вказавши потрібну кількість у вікні **I/O Устройства**, яке знаходиться у вкладці **Конфигурировать – 'I/O' Устройства.**

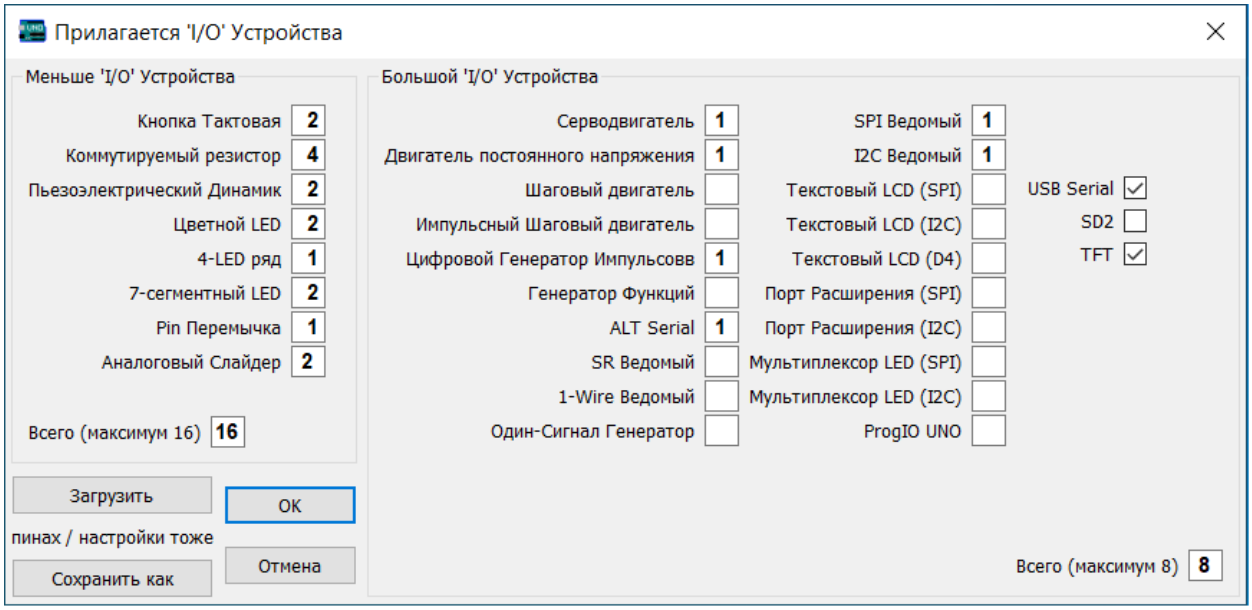


Рис. 4.7. Вікно I/O Устройства

Видалити інструменти можна видаливши ці значення у вікні кількості.

Детальну інформацію про інструменти можна знайти у вказаних закладках **Быстрый помощь, Полный помощь.**

Для збереження налагодження треба натиснути кнопку **Сохранить как** та вказати папку, де знаходиться відповідна програма.

У вкладці **Конфигурировать – Настройки** можна зробити додаткові налагодження, наприклад, обрати мову інтерактивного інтерфейсу та тип контролера, а саме, Arduino Uno V1, V2, V3 або Arduino Mega.

На лабораторній панелі можна спостерігати стан контактів віртуального Arduini UNO.

Знак питання означає що контакт не використовується, нуль на синьому фоні це сигнал низького рівня LOW, одиниця на червоному фоні це сигнал високого рівня HIGH, стрілка вгору на рожевому фоні, це сигнал ШІМ. На платі є функціонуючі світлодіоди: ON, RX, TX і pin 13.

Також, якщо клікнути двічі по одному з активних контактів, то запуститься вікно осцилографа.

На рис. 4.8 представлено вікно осцилографа з трьома активними контактами.

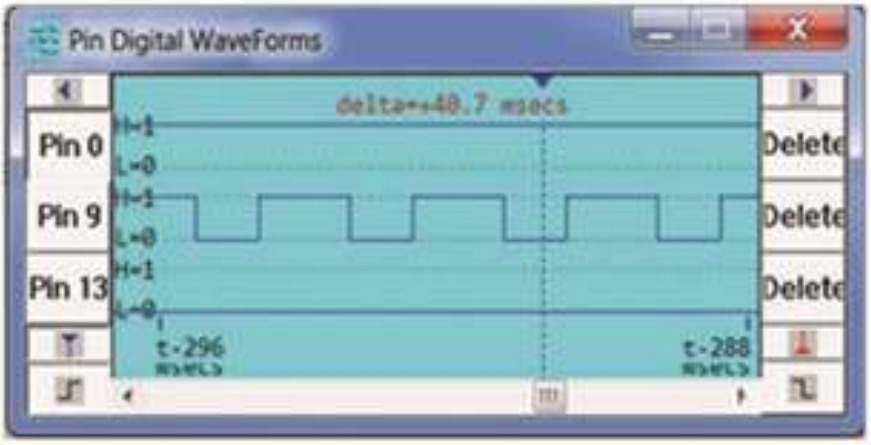


Рис. 4.8. Вікно осцилографа з трьома активними контактами

Pin 0 в стані одиниці, на Pin 9 бачимо ШІМ сигнал, а Pin 13 в стані нуль. Синя і червона мітки параметра delta виставляються шляхом перетягування їх мишкою, а параметр t (час) налаштовується коліщатком мишки.

На рис. 4.9 наведений приклад виконання програми, що здійснює миготіння контакту 13 та підрахунок кількості миготінь.

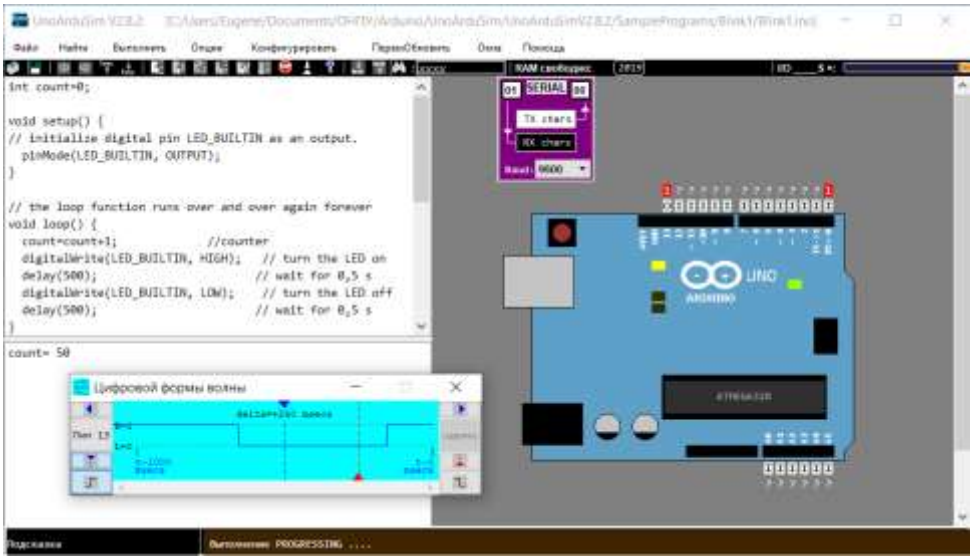


Рис. 4.9. Приклад виконання програми, що здійснює миготіння контакту 13 та підрахунок кількості миготінь

На лабораторній панелі можна спостерігати стан контактів віртуального Arduini UNO.

Знак питання означає що контакт не використовується, нуль на синьому фоні це сигнал низького рівня LOW, одиниця на червоному фоні це сигнал високого рівня HIGH, стрілка вгору на рожевому фоні, це сигнал ШІМ. На платі є функціонуючі світлодіоди: ON, RX, TX і pin 13.

Також, якщо клікнути лівою кнопкою по активному контакту, то запуститься вікно цифрового осцилографа (видає значення 0 та 1).

Права кнопка запускає аналоговий осцилограф, що дозволяє проглянути сигнали на аналогових входах.

Синя і червона мітки параметра delta виставляються шляхом перетягування їх мишкою, а параметр t (час) налаштовується коліщатком мишки.

Змінна count у вікні змінних показує кількість миготінь.

Додаток UnoArduSim зберігається у папці UnoArduSimV2.9.2.

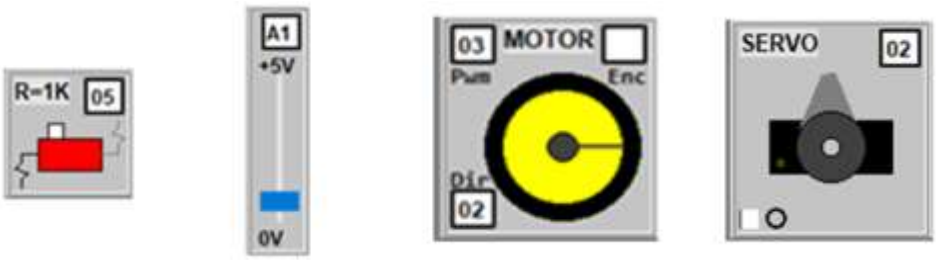
У папці translations знаходяться переклади файлів допомоги UnoArduSim_FullHelp та UnoArduSim_QuickHelp на різні мови. (російською мовою UnoArduSim_QuickHelp_ru.pdf та UnoArduSim_FullHelp_ru.pdf).

У папці SamplePrograms знаходяться приклади програм.

Цю папку можна використовувати для зберігання своїх програм у папках з відповідними назвами.

4.2. Склад апаратних компонетів емулятора UnoArduSim

Розглянемо додатковіі пристрої, які використовує симулятор UnoArduSim. На рис. 4.10 наведені елементи, за допомогою яких здійснюється введення дискретних та аналогових даних, та виконавчі пристрої.



а) б) в) г)

Рис. 4.10. Елементи, за допомогою яких здійснюється введення дискретних (а) та аналогових (б) даних, та виконавчі пристрої (в, г)

Коммутируемый резистор (рис. 4.10, а), шляхом клацання мишкою підключає резистор, який підключений до цифрового входу, на землю або на живлення, що відповідає підключенню да входу сигналу 0 або 1.

Аналоговый слайдер (рис. 4.10, б), повзунковий потенціометр 0-5 В може бути підключений до будь-якого обраного цифрового або аналогового входу. При зчитуванні з аналогового входу командою **analogRead()** у залежності від положення повзунка отримаємо значення від 0 до 1023.

Розглянемо виконавчі пристрої, які використовує симулятор UnoArduSim.

MOTOR або двигун постійного струму (рис. 4.10, в), це інструмент вводу-виводу, що здійснює емуляцію електродвигуна постійного струму.

Швидкість обертання встановлюється за допомогою сигналу з широтно-імпульсною модуляцією, що подається на вхід Pwm.

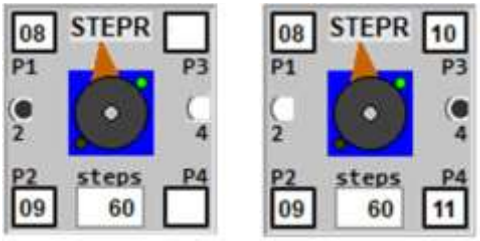
Напрямок обертання встановлюється на вході Dir.

На вихід Enc поступають сигнали з умонтованого енкодера (імпульсного датчика переміщення), що видає 8 імпульсів на одне обертання

SERVO (сервомотор) (рис. 4.10, г), у якого вал приймає певний кут від 0 до 180 градусів в залежності від прогальності ШІМ сигналу. Сервомотори використовують у пристроях, де потрібно здійснити поворот на певний кут, наприклад, для повороту ланки маніпулятора.

Для керування сервомотором використовується бібліотека *Servo*.

STEPR (Stepper) або кроковий двигун (рис. 4.11), у вікні steps виставляється кількість кроків на один оборот ротора. Є два варіанти: двох фазний і чотирьох фазний, які перемикаються шляхом встановлення галочки над двійкою або над четвіркою.



а) б)

Рис. 4.11.

На рис. 4.11, а показаний активний двофазний варіант з підключенням до контактів P1 і P2, якщо вибрати чотирьохфазний, то відповідно будуть додані ще два контакти P3 і P4 (рис. 4.11, б).

Параметр steps встановлює кількість кроків на одне обертання.

Для керування кроковим двигуном використовується бібліотека Stepper.

Далі наведена програма керування кроковим двигуном з реверсом без регулювання швидкості обертання для контролера Arduino з переміщенням на одне обертання в кожну сторону.

Кількість кроків на одне обертання 60.

Затримка між обертаннями 1 с.

```
#include <Stepper.h> //включення бібліотеки
Stepper myStepper(60, 8, 9, 10, 11); //підключення крокового двигуна
void setup() {
}
void loop()
{
myStepper.setSpeed(60);        //встановлення швидкості обертання
myStepper.step(60);            // переміщенням на одне обертання вперед
delay(1000);                   //затримка 1 с
myStepper.step(-60);           // переміщенням на одне обертання назад
delay(1000);                   //затримка 1 с
}
```

4.3. Приклади проєктів на основі UnoArduSim

Розглянемо можливість створення моделі ультразвукового датчика HC-SR04 за допомогою UnoArduSim.

Схема підключення ультразвукового датчика HC-SR04 до контролера Arduino Nano (а) та діаграми роботи (б) наведені на рис. 4.12.

Підключення датчика до Arduino: VCC на +5 V; TRIG на D7; ECHO на D8; GND на пін GND.

Параметри датчика:

- кут огляду 15º;
- відстань вимірювання від 0,03 до 5 м;
- для відстані від 0,03 до 0,6 м похибка складає 3 мм;
- для відстані від 0,6 до 5 м похибка збільшується.

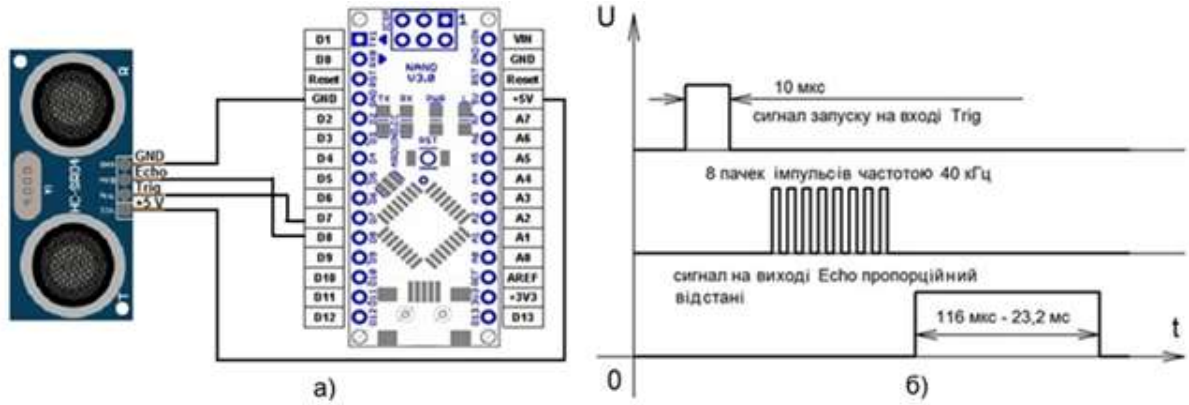


Рис. 4.12. Схема підключення ультразвукового датчика HC-SR04 до контролера Arduino Nano (а) та діаграми роботи (б)

Принцип роботи датчика можна умовно розділити на 4 етапи

1. Подаємо імпульс тривалістю 10 мкс, на вивід Trig.

2. Усередині далекоміра вхідний імпульс перетворюється в 8 імпульсів частотою 40 КГц і надсилається вперед через випромінювач "Т бочечка"

3. Дійшовши до перешкоди, послані імпульси відбиваються і поступають на приймач "R бочечка", внаслідок чого отримуємо вихідний сигнал на виводі Echo, тривалість якого пропорційна відстані до об'єкту.

4. За допомогою програми контролера перетворюємо отриманий сигнал в відстань за формулою:

тривалість імпульсу (мкс) / 58 = дистанція (см);

тривалість імпульсу (мкс) / 148 = дистанція (дюйм).

Для вимірювання тривалості імпульсів, наприклад, у ультразвукових датчиків вимірювання відстані, використовується функція **pulseIn()**.

Ця функція зчитує тривалість позитивного (HIGH) чи негативного (LOW) імпульсу на визначеному вході (pin). Наприклад, якщо значення є високим (HIGH), pulseIn() чекає, поки стан pin зміниться на HIGH, починає підрахунок часу у мікросекундах, потім чекає, поки стан входу pin не зміниться на LOW, і зупиняє підрахунок.

pulseIn(pin, value)

pulseIn(pin, value, timeout)

де

pin - номер піна для зчитування імпульсу (int)

value - тип імпульсу для зчитування HIGH чи LOW (*int*)

(опціонально)

timeout - час очікування у мікросекундах, стандартне значення 1 секунда (*unsigned long*)

Повертає тривалість імпульсу (у мікросекундах) або 0, якщо не було повного імпульсу за час очікування (unsigned long)

У наведеному нижче скетчі дистанція відсилається в порт комп'ютера, а при дистанції менше 15 сантиметрів включається світлодіод, що підключений до піну 13.

```
#define Trig 7
#define Echo 8
#define ledPin 13
void setup()  {
pinMode(Trig, OUTPUT); // вихід
pinMode(Echo, INPUT); // вхід
pinMode(ledPin, OUTPUT);
Serial.begin(9600); //задаємо швидкість послідовного каналу
}
```

```
unsigned long impulseTime=0;
unsigned long distance_mm=0;
void loop() {
digitalWrite(Trig, HIGH); // Подаємо імпульс на вхід trig
delayMicroseconds(10); // що дорівнює 10 мікросекунд
digitalWrite(Trig, LOW); // відключаємо
impulseTime=pulseIn(Echo, HIGH); // визначаємо тривалість імпульсу
distance_mm=impulseTime*10/58; // Перераховуємо в міліметри
Serial.println(distance_mm); // Виводимо на послідовний канал
if (distance_mm<150) // Якщо відстань менше ніж 15 сантиметрів (870 мкс)
{  digitalWrite(ledPin, HIGH); // включаємо світлодіод  }
else {
digitalWrite(ledPin, LOW); // у протилежному випадку виключаємо
}
delay(100);
```

/* чекаємо 0,1 секунди, Наступний імпульс може бути відправленний тільки після зникнення відбитого попереднього імпульсу. Рекомендований період між імпульсами повинен бути не менше 50 мс.*/

}

У даній роботі використовуємо інструмент **Один-Сигнал Генератор** ('1SHOT'), наведений на рис. 4.13.

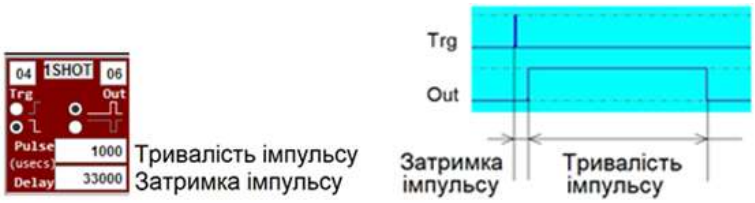


Рис. 4.13.

Цей інструмент вводу-виводу здійснює емуляцію цифрової одноразової дії, яка може генерувати імпульс обраної полярності та тривалості Pulse на контакті "Out".

Імпульс виникає після заданої затримки Delay від фронту запуску, який надходить на вхідний контакт Trg (тригер).

Параметри Pulse та Delay надаються у мікросекундах.

Виконання програми на емуляторі UnoArduSim наведено на рис. 4.14.

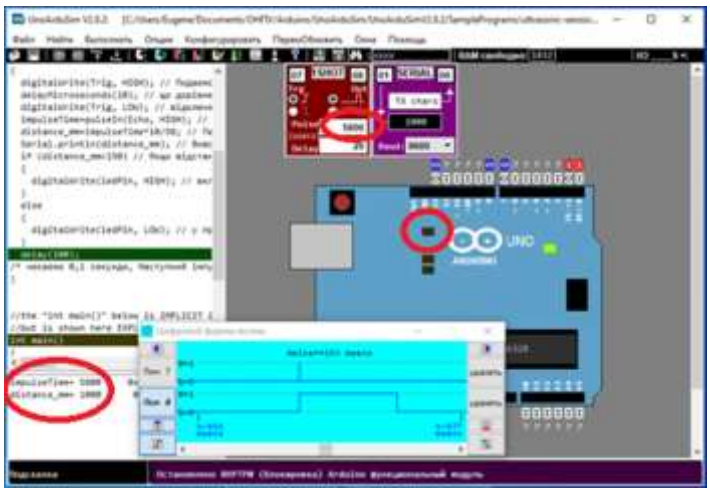


Рис. 4.15. Виконання програми моделювання ультразвукового датчика на емуляторі UnoArduSim

Використання ультразвукового датчика для пошуку перешкод

На рис. 4 16 наведений транспортний засіб з двома двигунами, що дає можливість здійснювати диференційне керування переміщенням, та ультразвуковим датчиком визначення перешкод.

Знаючи ширину транспортного засобу s та кут огляду ультразвукового датчика α можна визначити відстань L, яка визначає відсутність перешкод на шляху транспортного засобу.

$L = s / 2\ tg(\alpha/2)$.

Для прикладу призначимо s = 0,15 м, а для датчика, який використовується, α = 15º, і тоді отримаємо

$L = 0{,}15 / 2\ tg(15^º /2) = 0{,}57$ м (570 мм).

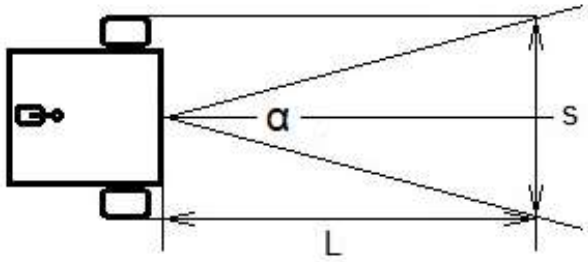


Рис. 4.16. Транспортний засіб з двома двигунами та ультразвуковим датчиком визначення перешкод

На рис. 4.17 наведений вибір пристроїв для мобільного робота з двома двигунами постійного струму та інструментом '1SHOT‘, за допомогою якого здійснюється моделювання ультразвукого датчика вимірювання відстані, а також інструментом послідовного виведення даних на комп’ютер 'SERIAL‘.

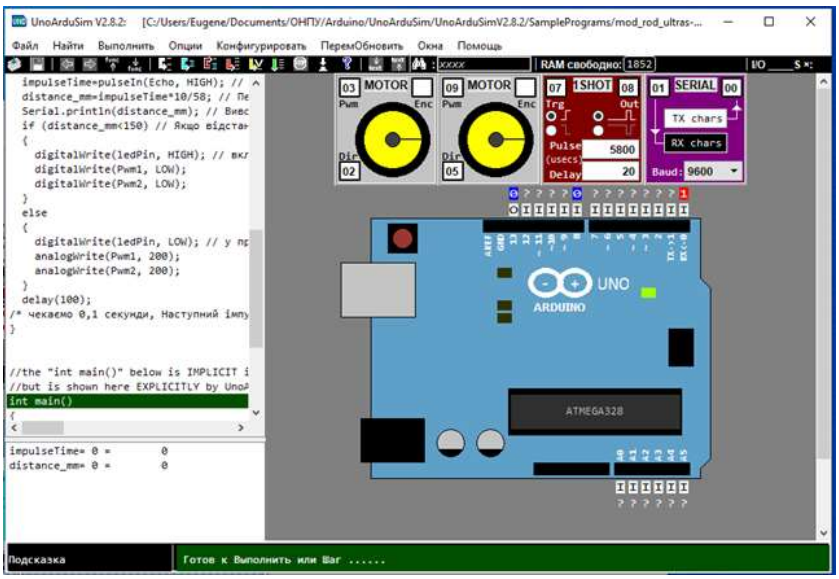


Рис. 4.17. Вибір пристроїв для мобільного робота з двома двигунами постійного струму та інструментом '1SHOT‘,

Далі наведена програма для визначення перешкод та зупинкою при наявності перешкоди.

```
#define Trig 7
#define Echo 8
#define Pwm1 3
#define Dir1 2
```

```
#define Pwm2 9
#define Dir2 5
#define ledPin 13
void setup() {
pinMode(Trig, OUTPUT); // вихід
pinMode(Echo, INPUT); // вхід
pinMode(ledPin, OUTPUT);
Serial.begin(9600); //задаємо швидкість послідовного каналу
pinMode(Dir1, OUTPUT);
pinMode(Dir2, OUTPUT);
pinMode(Pwm1, OUTPUT);
pinMode(Pwm2, OUTPUT);
digitalWrite(Dir1, LOW);
digitalWrite(Dir2, LOW); }
unsigned long impulseTime=0;
unsigned long distance_mm=0;
void loop() {
digitalWrite(Trig, HIGH); // Подаємо імпульс на вхід trig
delayMicroseconds(10); // що дорівнює 10 мікросекунд
digitalWrite(Trig, LOW); // відключаємо
impulseTime=pulseIn(Echo, HIGH); // визначаємо тривалість імпульсу
distance_mm=impulseTime*10/58; // Перераховуємо в міліметри
Serial.println(distance_mm); // Виводимо на послідовний канал
if (distance_mm<750) // Якщо відстань менше ніж 75 сантиметрів
{
digitalWrite(ledPin, HIGH); // включаємо світлодіод
digitalWrite(Pwm1, LOW);  //та зупиняємо двигуни
digitalWrite(Pwm2, LOW); }
else {
digitalWrite(ledPin, LOW); // у протилежному випадку включаємо двигуни
analogWrite(Pwm1, 200);
analogWrite(Pwm2, 200); }
delay(100);
/* чекаємо 0,1 секунди, Наступний імпульс може бути відправленний тільки після зникнення відбитого попереднього імпульсу. Рекомендований період між імпульсами повинен бути не менше 50 мс.*/
}
```

Конструювання чотирьохвісного маніпулятора з ручним керуванням

На рис.4.18 наведений маніпулятор з чотирма сервоприводами.

Рис.4.18 Маніпулятор з чотирма сервоприводами

На рис. 4.19 зображена комп’ютерна модель ручного керування маніпулятором, розроблена за допомогою UnoArduSi, з чотирма сервоприводами та органами ручного керування у вигляді потенціометрів, що реалізуються за допомогою аналогових слайдерів, які підключені до аналогових входів.

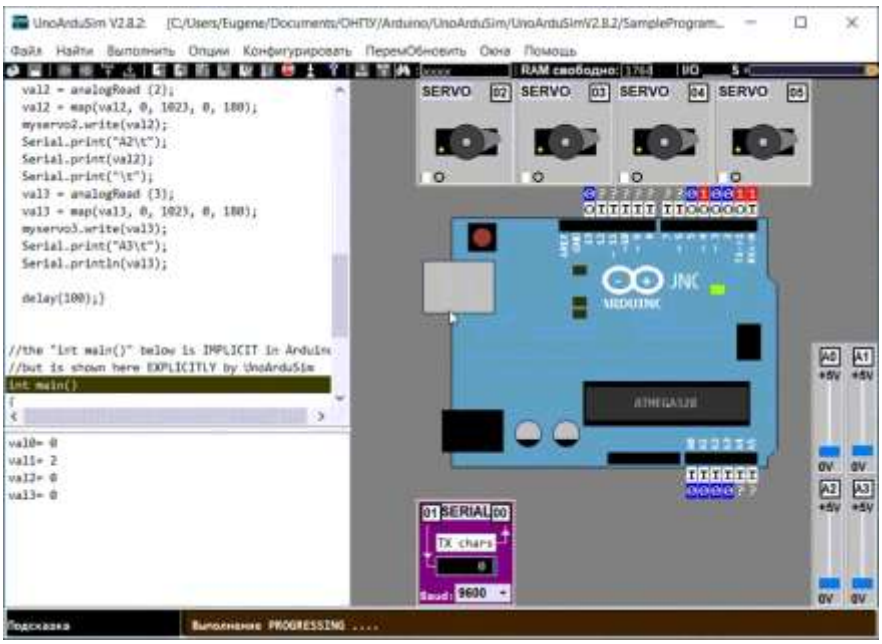


Рис. 4.19. Модель ручного керування маніпулятором

Далі наведена програма для ручного керування маніпулятором. Оскільки результат опитування потенціометрів видає значення в діапазоні від 0 до 1023, а для керування сервоприводом потрібний діапазон складає від 0 до 180, програма здійснює перетворення масштабу

```
#include <Servo.h>
Servo myservo0;
Servo myservo1;
Servo myservo2;
Servo myservo3;
int val0=0;
int val1=0;
int val2=0;
int val3=0;
void setup() {
myservo0.attach(2); //база
myservo1.attach(3); //плече
myservo2.attach(4); //рука
myservo3.attach(5); //захват
Serial.begin(9600); }
void loop() {
val0 = analogRead (0); //опитування аналогового слайдера A0
val0 = map(val0, 0, 1023, 0, 180); //перетворення масштабу
myservo0.write(val0);
Serial.print("AO\t");
Serial.print(val0);
Serial.print("\t");
val1 = analogRead (1); //опитування аналогового слайдера A1
val1 = map(val1, 0, 1023, 0, 180); //перетворення масштабу
```

```
myservo1.write(val1);
Serial.print("A1\t");
Serial.print(val1);
Serial.print("\t");
val2 = analogRead (2); //опитування аналогового слайдера A2
val2 = map(val2, 0, 1023, 0, 180); //перетворення масштабу
myservo2.write(val2);
Serial.print("A2\t");
Serial.print(val2);
Serial.print("\t");
val3 = analogRead (3); //опитування аналогового слайдера A3
val3 = map(val3, 0, 1023, 0, 180); //перетворення масштабу
myservo3.write(val3);
Serial.print("A3\t");
Serial.println(val3);
delay(500);      }
```

На рис. 4.20 наведений результат керування сервоприводами шляхом опитування аналогових слайдерів.

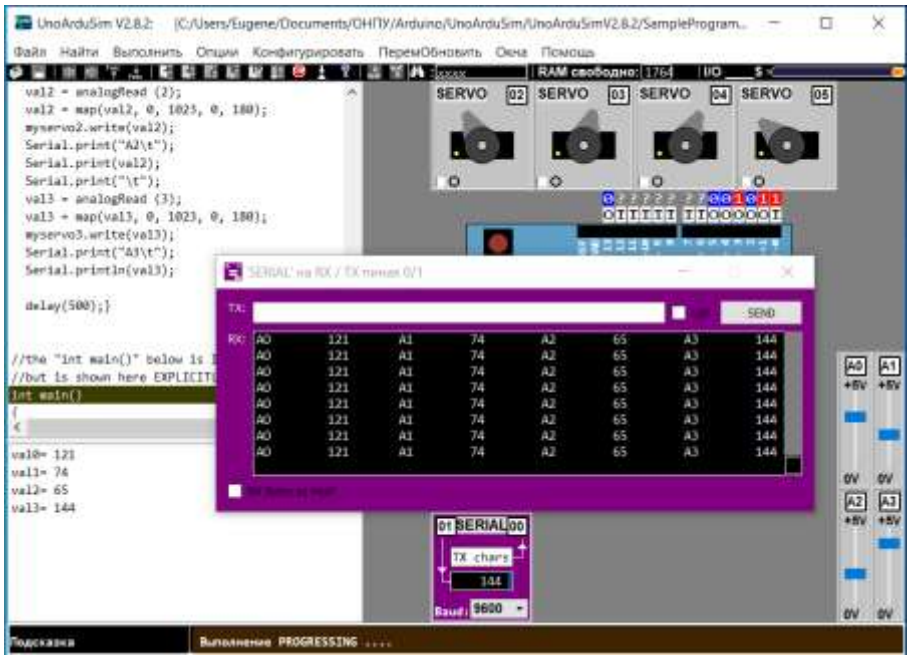


Рис. 4.20. Результат керування сервоприводами шляхом опитування аналогових слайдерів

Конструювання системи визначення шляху переміщення при використанні двигуна постійного струму з енкодером

На рис. 4.21 зображена комп'ютерна модель системи визначення шляху переміщення при використанні двигуна постійного струму з енкодером.

Підрахунок кількості імпульсів енкодера здійснюється за допомогою переривання, яка викликає функцію void counter(), де здійснюється підрахунок шляхом додавання 1 до змісту змінної pulses (лічильника) для кожного переривання за допомогою команди pulses++, яка еквівалентна команді pulses = pulses +1.

Дозвіл переривання здійснюється за допомогою команди

attachInterrupt(digitalPinToInterrupt(dsR), counter, CHANGE);

яка вказує вхід (dsR), на який поступає сигнал переривання, функцію, яку викликає переривання (counter), та умову здійснення переривання (CHANGE – зміна стану). Таким

чином за одне обертання буде здійснено 16 переривань, а точність встановлення кута повороту буде складати 22,5°.

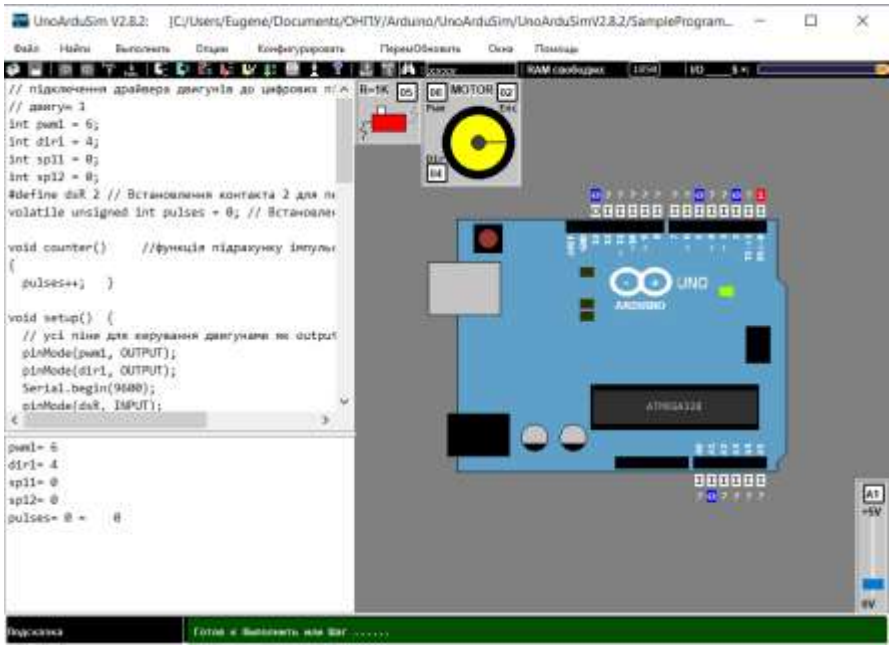


Рис. 4.21. Комп’ютерна модель системи визначення шляху переміщення

Далі наведена програма визначення шляху переміщення при використанні двигуна постійного струму з енкодером. Коли кількість імпульсів дорівнює 80 двигун зупиняється. Після затримки на 2 с переривання забороняється, у лічильник записується 0 та переривання знов дозволяється.

```
// підключення драйвера двигунів до цифрових пінів Arduino
int pwm1 = 6; //вихід сигналу для керування двигуном (pwm)
int dir1 = 4;  //напрямок руху
int sp11 = 0; //значення сигналу з потенціометра
int sp12 = 0; //перетворений сигнал для керування двигуном (pwm)
#define dsR 2 // Встановлення контакта 2 для переривання
volatile unsigned int pulses = 0; // Встановлення змінної pulses
void counter() {          //функція підрахунку імпульсів
pulses++;  }
void setup() {
// усі піни для керування двигунами як outputs
pinMode(pwm1, OUTPUT);
pinMode(dir1, OUTPUT);
Serial.begin(9600);
pinMode(dsR, INPUT);
pulses = 0;
// дозвіл переривання
attachInterrupt(digitalPinToInterrupt(dsR), counter, CHANGE); }
void loop() {
// запуск двигуна 1 вперед
sp11 = analogRead(1); //опитування аналогового слайдера
sp12 = map(sp11, 0, 1023, 0, 255); //перетворення масштабу для ШІМ
digitalWrite(dir1, digitalRead(5));
analogWrite(pwm1, sp12);
if (pulses >= 80) { //якщо кількість імпульсів дорівнює 80 зупинити двигун
//Двигуни Стоп
```

```
digitalWrite(dir1, !digitalRead(5));  //гальмування противключенням
delay(50);
analogWrite(pwm1, 0); //зупинка
delay(2000);
detachInterrupt(digitalPinToInterrupt(dsR)); // заборона переривання
Serial.println(pulses);
pulses = 0;
// дозвіл переривання
attachInterrupt(digitalPinToInterrupt(dsR), counter, CHANGE);
}
}
```

Контрольні питання

1. Визначити, для чого використовують додаток UnoArduSim.
2. Описати, де знайти скорочену та повну інформацію про UnoArduSim.
3. Описати, для чого використовують вкладку Конфигурировать – 'I/O' Устройства.
4. Розповісти, як можна спостерігати стан контактів віртуального Arduini UNO.
5. Визначити, які функції виконують елементи **Коммутируемый резистор** та **Аналоговый слайдер**.
6. Описати, які двигуни входять у склад емулятора.
7. Визначити, яка функція та який інструмент використовуються для створення моделі ультразвукового датчика відстані.
8. Описати, які задачі може вирішувати ультразвуковий датчик.
9. Розповісти, як можна здійснити ручне керування сервоприводом.
10. Описати, як можна здійснити визначення шляху переміщення.

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 програмованих логічних контролерів.

Розглянемо автоматизовану транспортно-складську систему на основі програмованих логічних контролерів та засоби її проектування на прикладі систем автоматизації SIMATIC S7, що випускається фірмою SIEMENS (рис. 5.1).

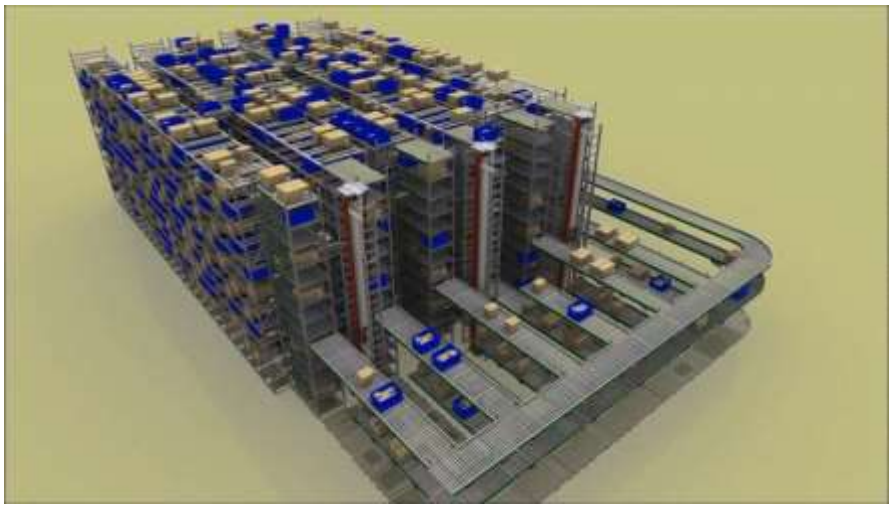


Рис. 5.1. Транспортно-складська система на основі програмованих логічних контролерів

На рис. 5.2 наведений склад системи керування транспортно-складської системи.

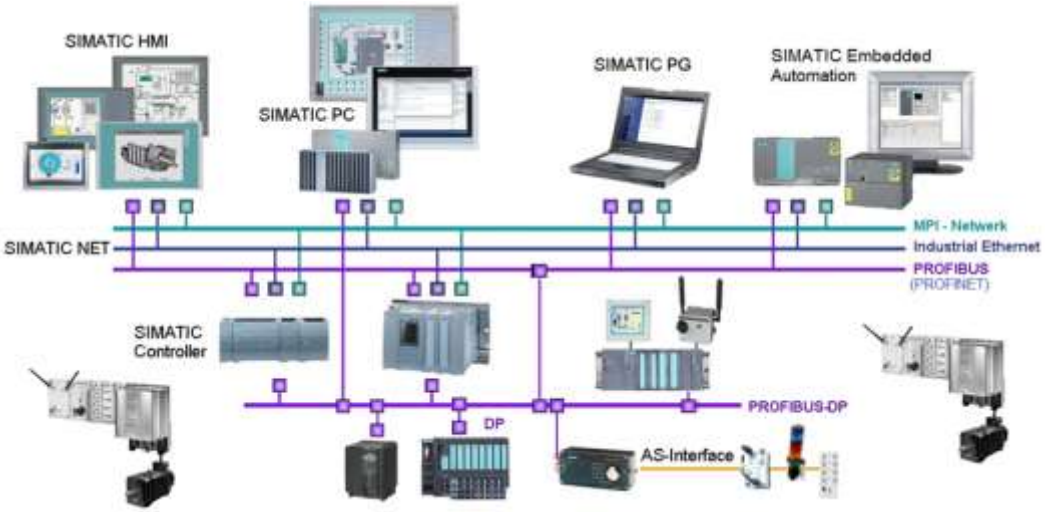


Рис. 5.2. Склад системи керування

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, наприклад, контролери фірми Сименс LOGO!, та 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 S7-1200, 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.

Контролери LOGO! є найпростішими пристроями керування, які використовують принцип логічного керування, а тому мають в основному логічні функції (рис. 5.3).

Рис. 5.3. Контролери LOGO!

Відмінністю цих контролерів є умонтовані засоби програмування. Програма, яка вводиться за допомогою клавіатури, відображається на дісплеї у вигляді окремих функцій.

Для більш комфортного програмування використовується програма LOGO!Soft Comfort, яка встановлюється на персональний комп’ютер, та використовує такі форми представлення програми, як функціональна та релейно-контактна схема (рис. 5.4).

Модульний принцип побудови ПЛК дає можливість здійснити конструювання та проєктування системи керування шляхом вибору та налагодження таких модулів, які є оптимальними для поставленої задачі. Для контролерів SIMATIC S7-300/400 існують такі типи модулів:

- 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.

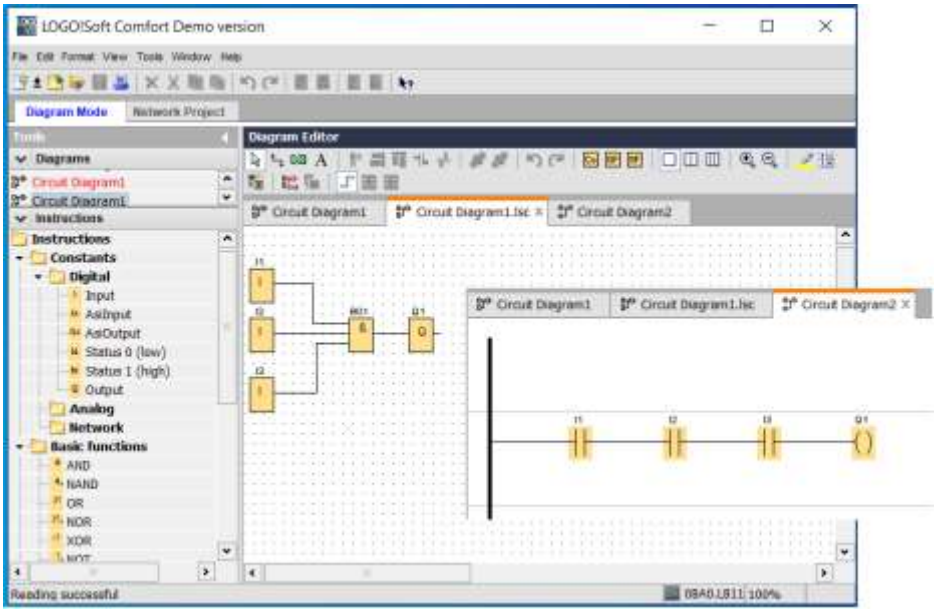


Рис. 5.4. Програма LOGO!Soft Comfort

Кожен модуль має декілька варіантів, що дозволяє вибрати оптимальну за можливостями та вартістю конфігурацію системи керування (рис. 5.5).

Рис. 5.5. Модулі, що входять у склад контролерів SIMATIC S7-300

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, конвеєрів, ЧПУ та інших пристроїв.

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 залежності від кількості модулів використовуватися побудова системи автоматизації з одним або кількома рядами.

Наведений на рис. 5.6 ПЛК SIMATIC S7-300 може мати до чотирьох рядів.

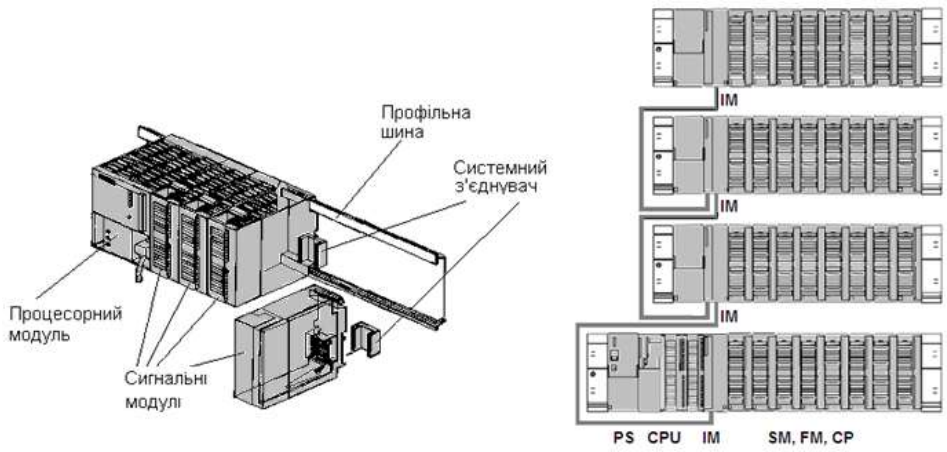


Рис. 5.6. ПЛК SIMATIC S7-300

При цьому використовуються спеціальні інтерфейсні модулі підключення IM.

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

На рис. 5.7 наведена система з резервуванням на основі ПЛК S7-400H.

Рис. 5.7. 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.

Для автономних транспортних засобів використовують розподілені системи керування з використанням бездротового зв’язку.

На рис. 5.8 наведений приклад такої системи керування на основі системи розподіленого вводу-виводу ET200pro фірми Сименс з використанням інтерфейсного модуля IM 154-6 PN IWLAN, який підтримує обмін даними між відокремленими модулями ET 200pro і програмованим логічним контролером через локальну бездротову мережу IWLAN.

Рис. 5.8. Система керування на основі системи розподіленого вводу-виводу ET200pro

Транспортна система Multishuttle використовує розподілені системи керування з використанням бездротового зв’язку для переміщення візків вздовж стелажів (рис. 5.9).

Рис. 5.9. Транспортна система Multishuttle

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. 5.10).

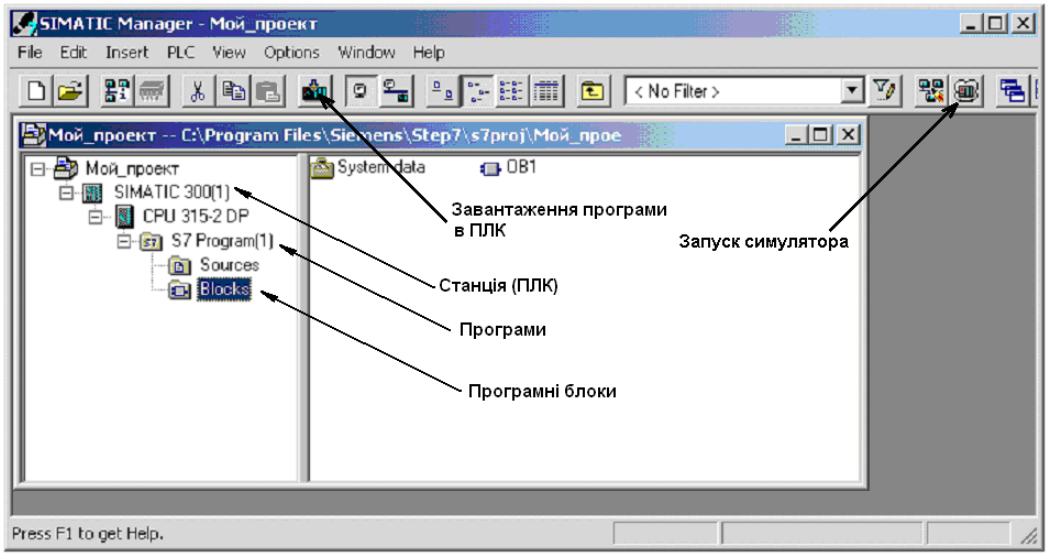


Рис. 5.10. Керуюча програма **"SIMATIC-Manager“**

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

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.

Визначення структури і складу системи керування за допомогою конфігуратора **HW Config** (рис. 5.11).

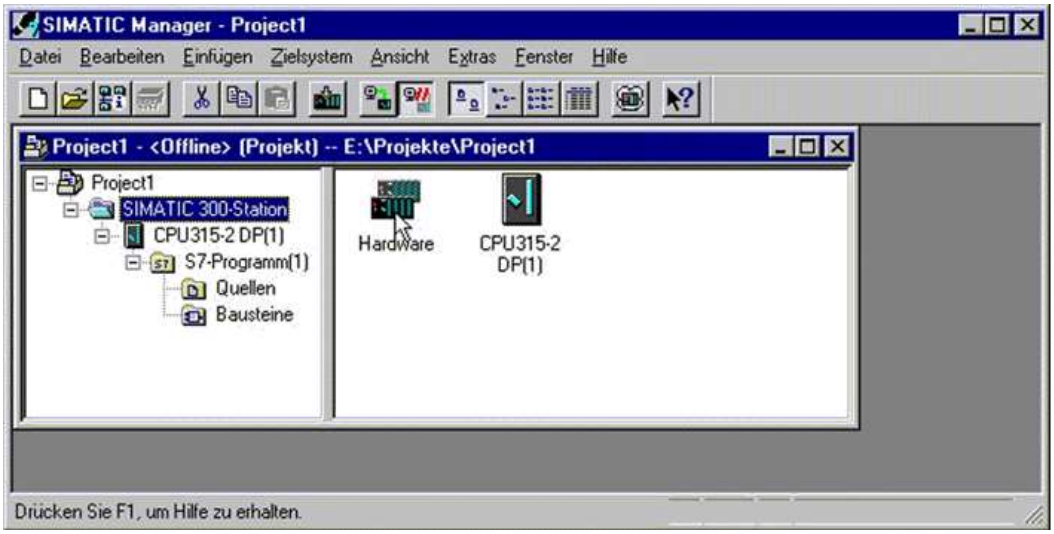


Рис. 5.11. Конфігуратор **HW Config**

На рис. 5.12 наведений приклад конфігурації системи керування.

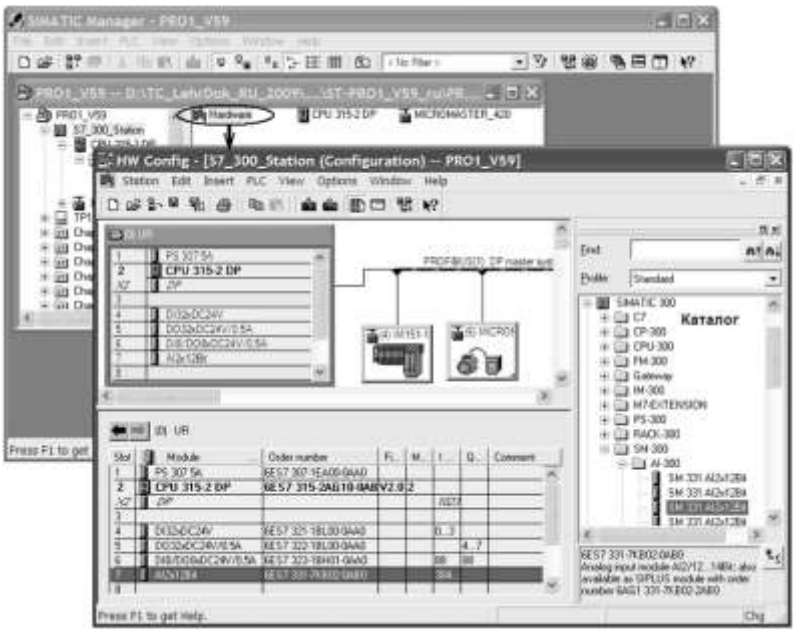


Рис. 5.12. Приклад конфігурації системи керування

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. 5.13 наведена структура та порядок виконання програми.

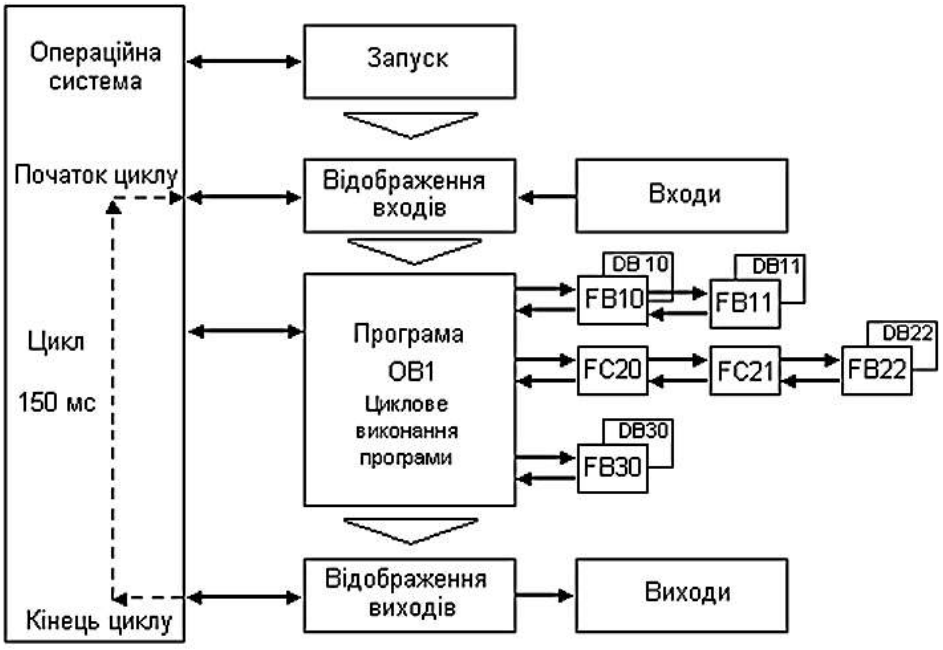


Рис. 5.13. Структура та порядок виконання програми

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

Редактори для створення програм у вигляді контактного та функціонального плану дозволяють здійснювати вибір окремих команд за допомогою каталогу (рис. 5.14).

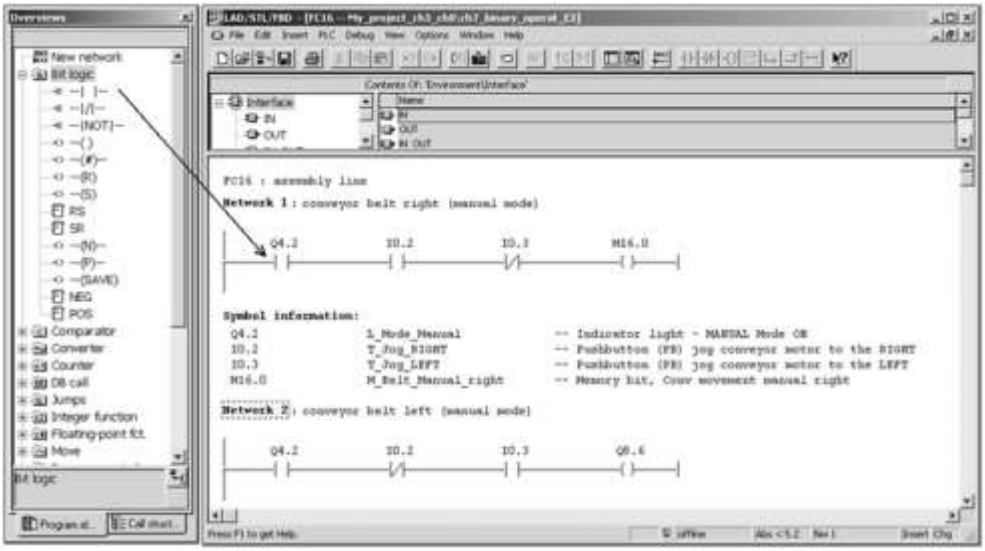


Рис. 5.14. Редактори для створення програм

Для створення програмних блоків використовують форми представлення програм у вигляді контактного плану, функціонального плану та послідовності команд (рис. 5.15).

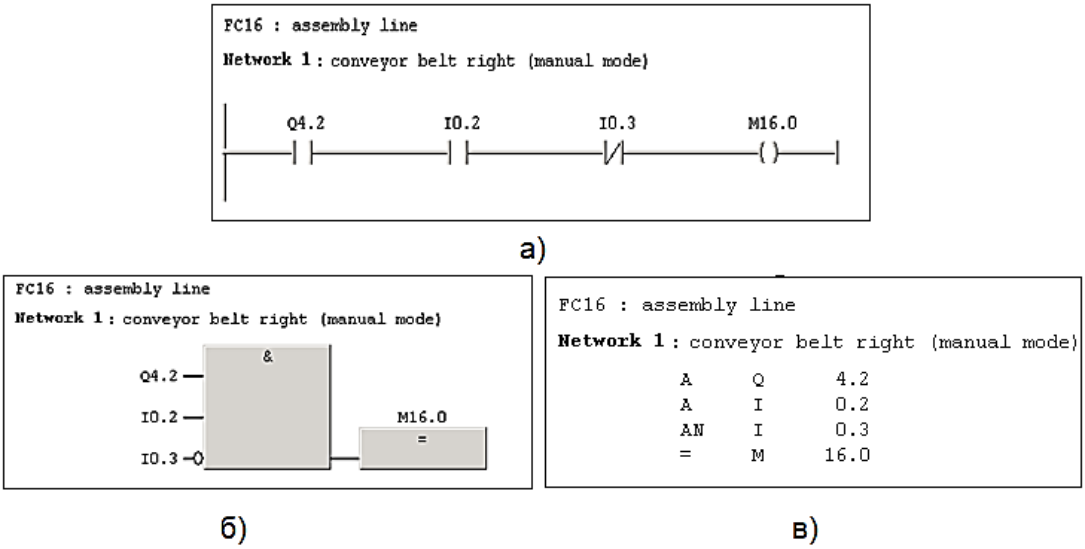


Рис. 5.15. Форми представлення програм у вигляді контактного плану (а), функціонального плану (б), послідовності команд (в)

Система команд програмованих логічних контролерів

Сучасні програмовані логічні контролери (ПЛК) мають логічні команди над двійковими числами, функції ліку (лічильники), часу (таймери), а також функції математичної та логічної обробки чисел у різних форматах.

На рис. 5.16 наведені основні елементи логічних функцій для контактного та функціонального плану.

Для реалізації логічного керування, які є програмним аналогом релейних систем керування, використовують логічні функції, такі як логічне І, логічне АБО та виключне АБО (рис. 5.17).

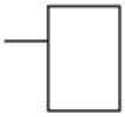
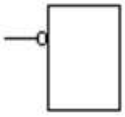
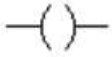
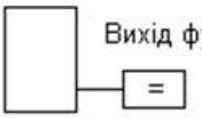
Елемент		Функція
Контактний план	Функціональний план	
Нормально відкритий контакт	Вхід функції 	Результат опиту входу дорівнює "1", якщо стан сигналу операнда на вході дорівнює "1". Якщо стан сигналу операнда на вході дорівнює "0", то результат опиту також дорівнює "0".
Нормально закритий контакт	Інверсний вхід функції 	Результат опиту входу дорівнює "0", якщо стан сигналу операнда на вході дорівнює "1". Якщо стан сигналу операнда на вході дорівнює "0", то результат опиту дорівнює "1".
Виконуючий пристрій 	Вихід функції 	Операнд, вихідні функції, має стан "1", якщо результат логічної операції цієї функції дорівнює "1". Операнд має стан "0", якщо результат логічної операції цієї функції дорівнює "0".

Рис. 5.16. Основні елементи логічних функцій для контактного та функціонального плану

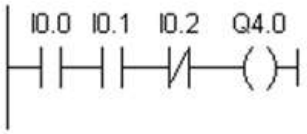
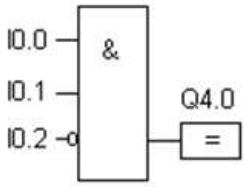
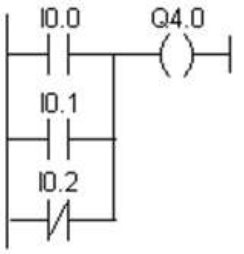
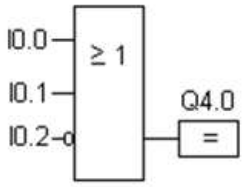
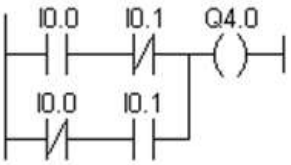
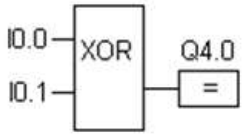
Функція	Контактний план	Функціональний план	Послідовність команд
Логічне І			A I0.0 A I0.1 AN I0.2 = Q4.0
Логічне АБО			O I0.0 O I0.1 ON I0.2 = Q4.0
Виключне АБО			X I0.0 X I0.1 = Q4.0

Рис. 5.17. Логічні функції

Результати виконання логічних функцій показані на рис. 5.18.

До функцій над двійковими числами можна також віднести функції пам'яті (рис. 5.19), а само RS тригери, які запам'ятовують значення вхідних сигналів.

Назва тригера визначається назвою виконуваних функцій, а само,

S – set встановлює вихід у одиницю,

R – reset скидає вихід у нуль.

При наявності сигналу на обох входах одиниць пріоритет має той вхід, який опитується останнім.

Вхідні сигнали		Вихідні сигнали Q4.0		
I0.0	I0.1	Логічне І	Логічне АБО	Виключне АБО
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Рис. 5.18. Результати виконання логічних функцій

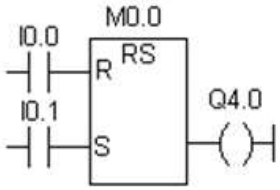
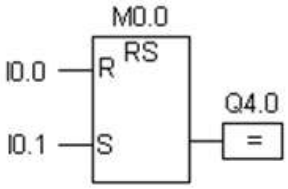
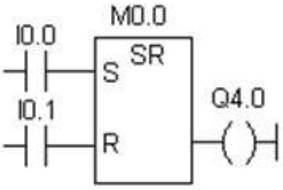
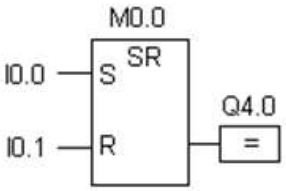
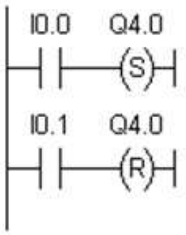
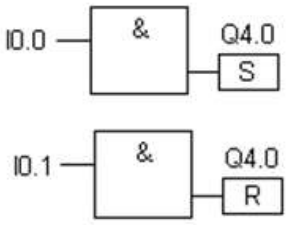
Функція	Контактний план	Функціональний план	Послідовність команд
Функція RS тригера з приорітетом встановлення в "1"			A I0.0 R M0.0 A I0.1 S M0.0 A M0.0 = Q4.0
Функція SR тригера з приорітетом скидання у "0"			A I0.0 S M0.0 A I0,1 R M0.0 A M0.0 = Q4.0
Скорочені функції S та R			A I0.0 S Q4.0 A I0.1 R Q4.0

Рис. 5.19. Функції пам'яті

Для обробки даних використовують операції над числами (рис. 5.20).

Логічні операції над числами дозволяють здійснити логічне керування декількома об’єктами (рис. 5.21).

Для обробки числових даних використовують математичні операції (рис. 5.22).

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.

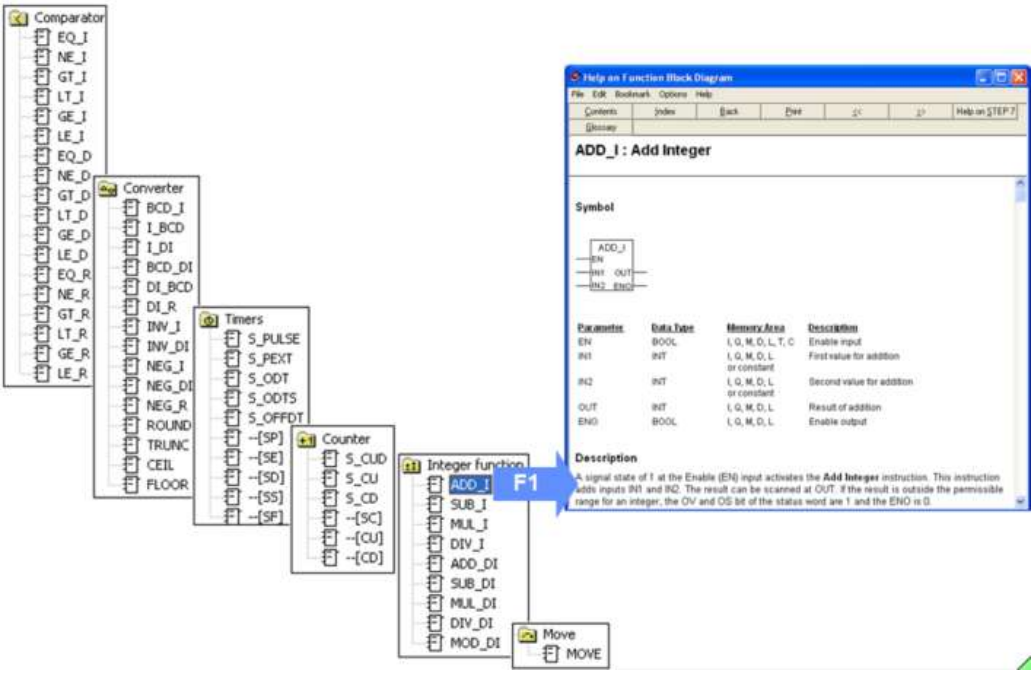


Рис. 5.20. Операції над числами

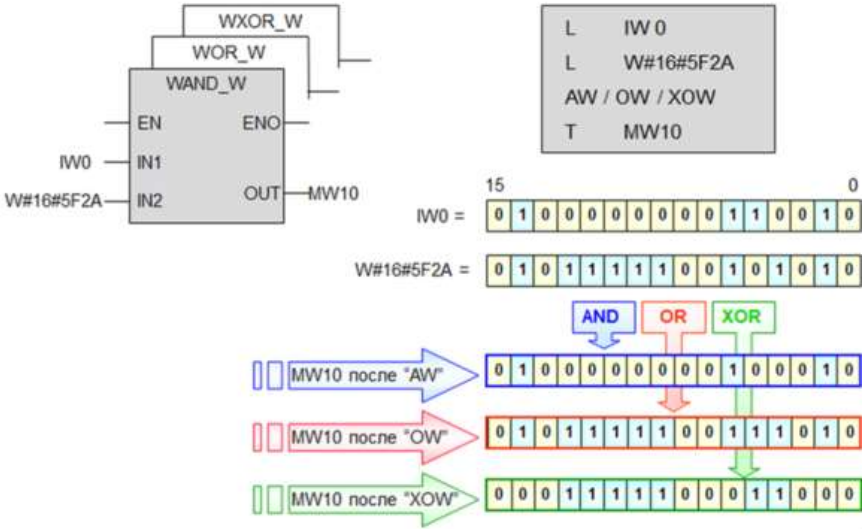


Рис. 5.21. Логічні операції над числами

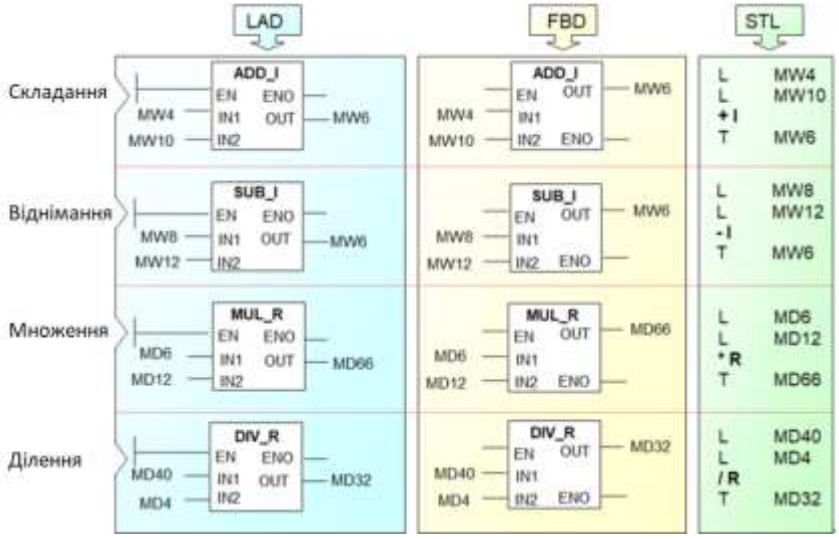


Рис. 5.22. Математичні операції

S7-PLCSIM являє собою симулятор, в якому можна визначити усі змінні реального ПЛК, такі як входи, виходи, пам'ять, таймери, лічильники, та моделювати різни режими роботи ПЛК, наприклад, режими запуску, переривання, реакцію на помилки тощо (рис. 5.23..

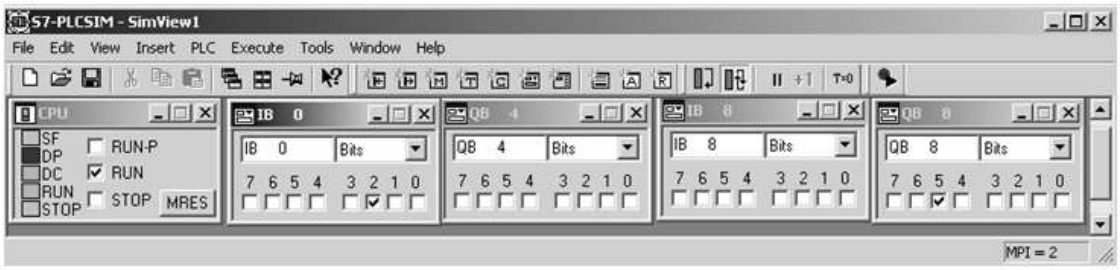


Рис. 5.23. Симулятор S7-PLCSIM

Для пошуку помилок на стадії налагодження та під час виконання програм використовують різні засоби діагностики.

Так тестування програми контролера (рис. 5.24) надає можливість визначити стан усіх змінних під час виконання програми.

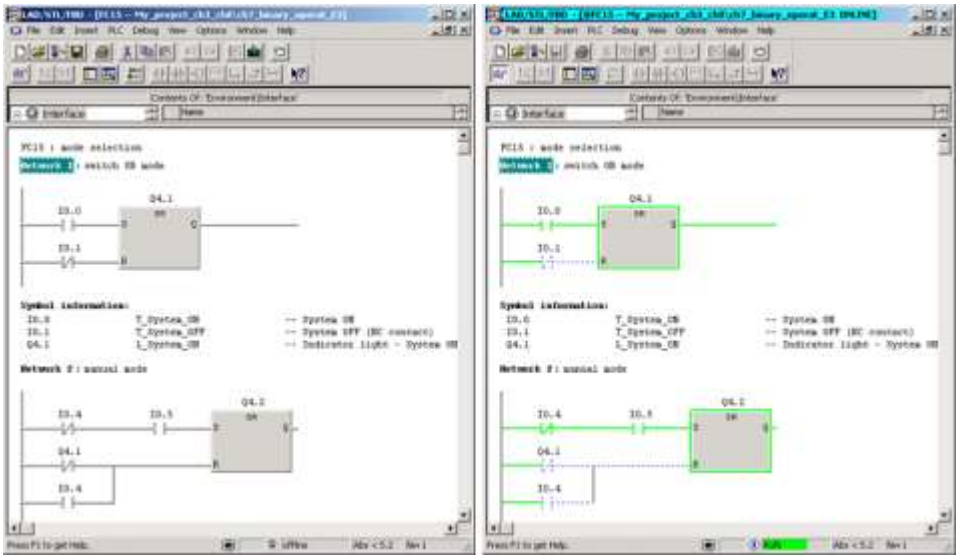


Рис. 5.24. Тестування програми контролера

Друга можливість перевірки стану змінних надається у таблиці перегляду та модифікації змінних.

У таблицю треба занести усі змінні з програми, стан яких треба переглядати.

Ця таблиця дає можливість змінювати існуючі дані з метою перевірки їх впливу на виконання програми (рис. 5.25).

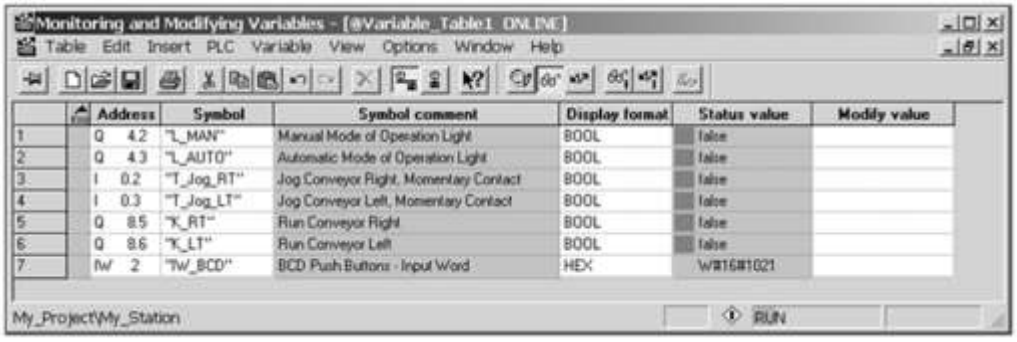


Рис. 5.25. Таблиця перегляду та модифікації змінних

Якщо помилка викликала зупинку процесора є можливість визначити причину зупинки. Для цього треба відкрити вікно **"Module Information"**. У цьому вікні є закладки, які несуть інформацію о причинах помилки, а також стану процесора на час зупини. Наприклад, на рис. 5.26, а наведена закладка **"Diagnostic Buffer"** (діагностичний буфер).

Для пошуку помилок у програмі призначена також інформація, яка міститься у довідкових даних, а саме: структура програми (рис. 5.26, б), таблиця перехресних посилань (рис. 5.26, в), перелік елементів, які використовуються у програмі (рис. 5.26, г).

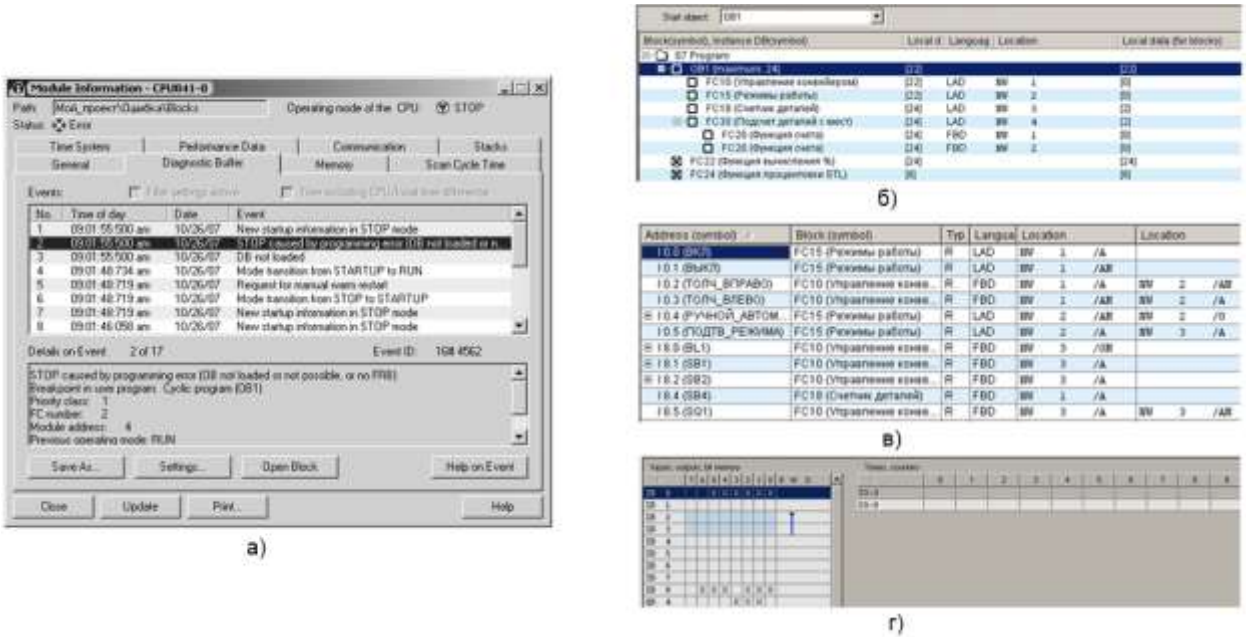


Рис. 5.26. Засоби пошуку помилок (а), визначення структури рограми (б), перехресних посилань (в) та перелік елементів (г)

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. 2.27 наведений частотний перетворювач фірми Сіменс SINAMICS G120, який має модульну структуру та складається з силового модуля, пристрою керування та операторської панелі.

Рис. 2.27. Частотний перетворювач SINAMICS G120

На рис. 5.28 наведена спрощена схема частотного перетворювача.

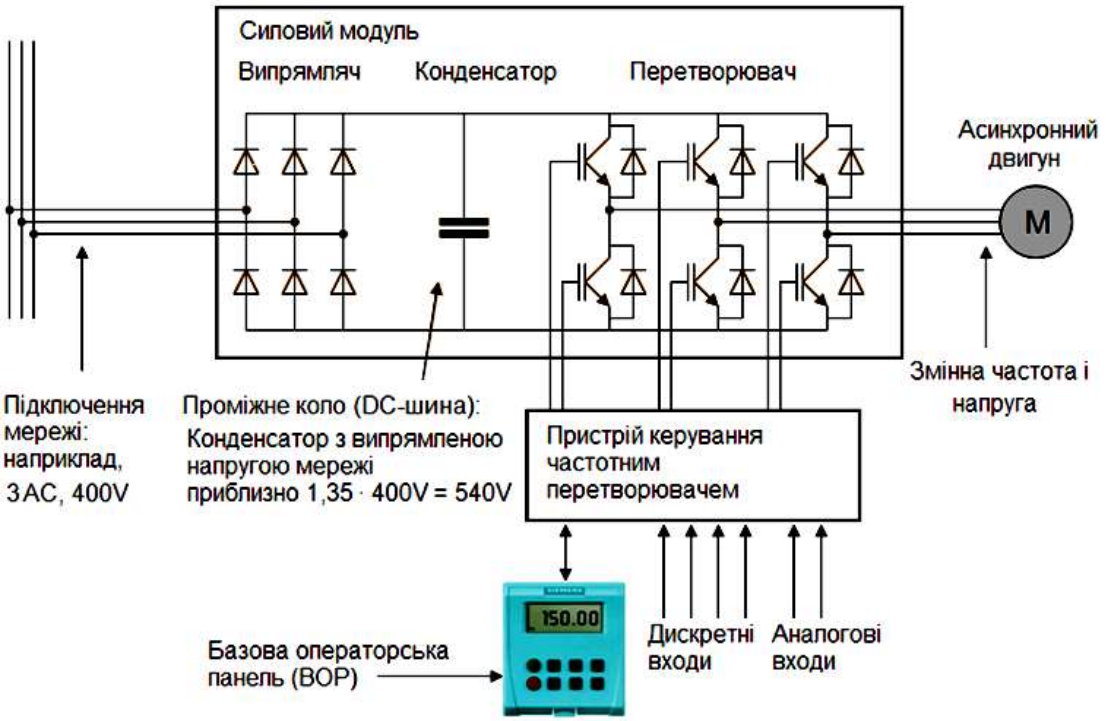


Рис. 5.28. Спрощена схема частотного перетворювача

На рис. 5.29 наведений принцип формування синусоїдального сигналу за допомогою широтно-імпульсної модуляції.

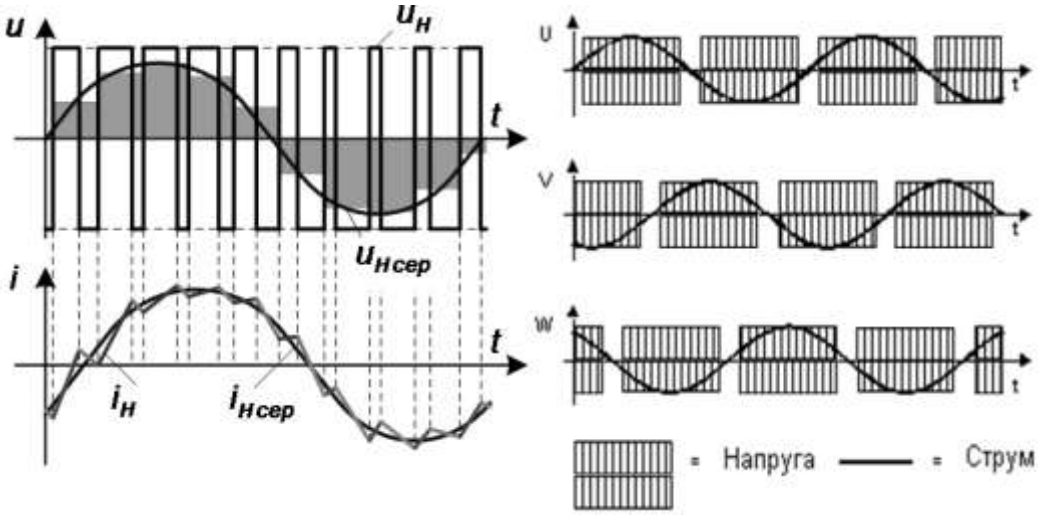


Рис. 5.29. Принцип формування синусоїдального сигналу за допомогою широтно-імпульсної модуляції

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 (рис. 5.30).

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.

Рис. 5.30. Операторська панель

Для налагодження та керування виконавчими пристроями існують програмні інструменти комп'ютерного налагодження та керування, наприклад програма STARTER для електроприводів фірми SIEMENS, яка має такі можливості:

- Offline - Створення проекту за допомогою Майстра без підключення до привода;
- Offline – Встановлення конфігурації приводу;
- встановлення підключення до частотного перетворювача – режим online;
- представлення параметрів;
- встановлення часу розгону та гальмування;
- керування за допомогою цифрових входів/виходів.

На рис. 5.31 наведена панель керування приводом у ручному режимі.

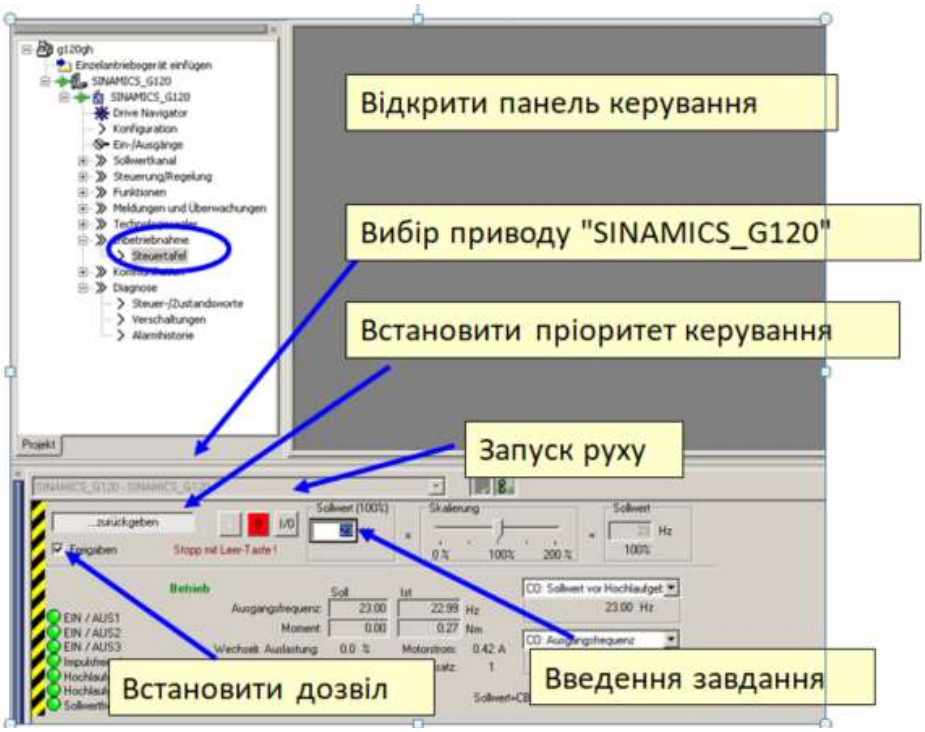


Рис. 5.31. Панель керування приводом у ручному режимі

Програма STARTER має засоби для підключення привода до контролера для здійснення керування за допомогою локальної мережі.

На рис. 5.32 наведений приклад програми керування приводом.

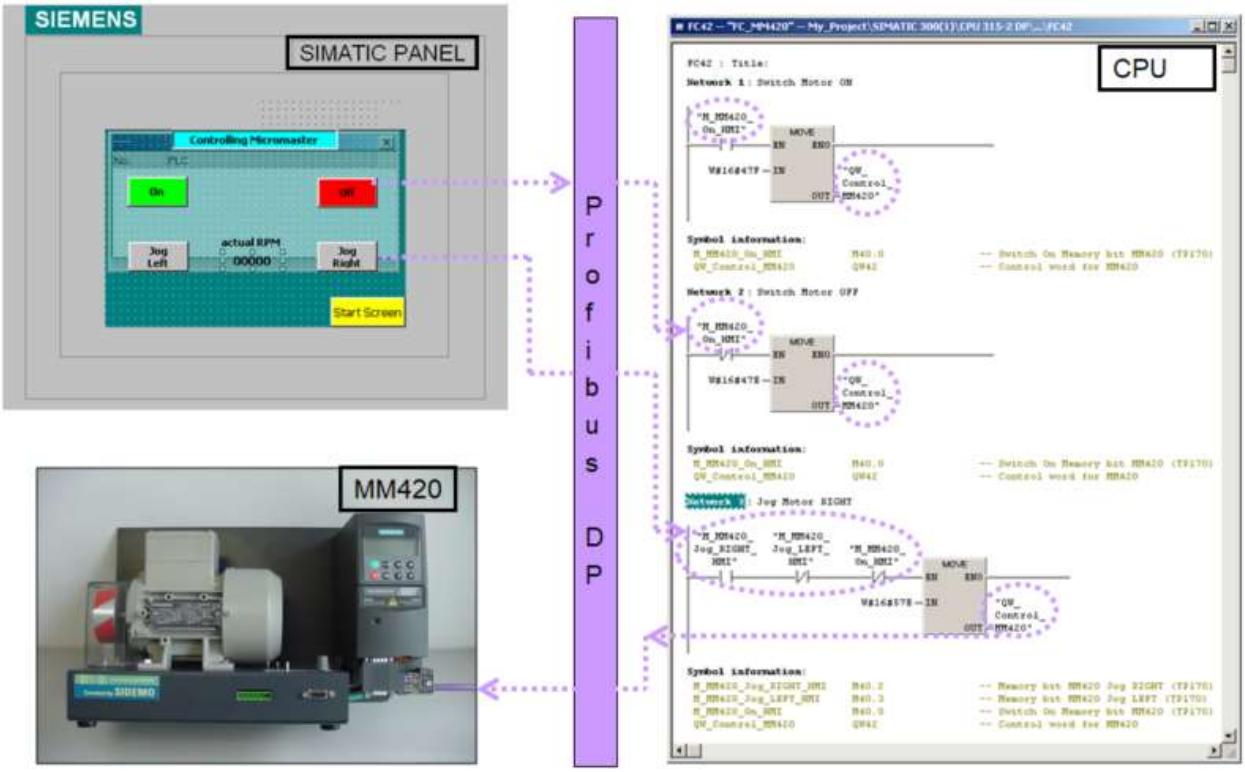


Рис. 5.32. Приклад програми керування приводом

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 органу або окремих ланок системою керування роботом та інших промислових машин.

Для визначення положення використовуються абсолютні датчики положення (рис. 5.33), які видають код кута обертання і не потребують визначення початкового значення (абсолютний датчик з PROFIBUS DP 6FX2001-5FP12).

Рис. 5.33. Абсолютні датчики положення

У конфігураторі 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. 5.34).

Рис. 5.34. Визначення датчиків у конфігуратора контролера

При використанні аналогових входів для отриманих значень треба здійснити функцію масштабування, щоб ці дані відповідали вимірюваним параметрам, оскільки для максимального значення параметра, що вимірюється, наприклад, 10 В для напруги, маємо результат опитування 27648. Масштабування здійснюється за допомогою системної функції FC105. Винятком є лише вимірювання температури, оскільки тримаємо дані з точністю одна десята градуса.

Для обробки даних, отриманих з інформаційних систем, використовуються додаткові математичні операції.

До математичних функцій також відносяться такі функції:

- синус (SIN),
- косинус (COS),
- тангенс (TAN),
- арксинус (ASIN), арккосинус (ACOS),
- арктангенс (ATAN),
- зведення в квадрат (SQR),
- витяг квадратного кореня (SQRT),
- експонента (EXP),
- логарифм (LN).

Всі математичні функції обробляють числа в форматі REAL (R)

Контрольні питання

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

6. Проєктування систем відображення технологічних процесів і керування.

6.1. Операторські панелі в складі систем керування.

Програмовані логічні контролери не мають вмонтованих систем відображення технологічного процесу та засобів введення даних. Цю задачу виконує так званий людино-машинний інтерфейс (HMI - Human machine interface), або системи спостереження та керування.

У промислових умовах людино-машинний інтерфейс найчастіше реалізується з використанням типових засобів: операторських панелей, комп'ютерів і типового програмного забезпечення.

Головним завданням людино-машинного інтерфейсу є полегшення роботи оператора шляхом відображення на екрані комп'ютера інтуїтивно зрозумілої інформації про роботу устаткування.

Таким чином HMI є частиною загальної системи керування.

Значне місце в таких системах займають комп'ютерні пристрої відображення інформації і керування у вигляді символьних та графічних операторських панелей OP та промислових персональних комп'ютерів, за допомогою яких оператор одержує інформацію про хід технологічного процесу і вводить необхідну додаткову інформацію або команди для керування системою.

Операторські панелі представляють собою спеціалізовані обчислювальні пристрої з засобами відображення та введення інформації.

Для відображення інформації використовують символьні та графічні дисплеї.

Введення інформації здійснюється за допомогою клавіатури або безпосереднє з екрану за допомогою маніпуляторів, наприклад, джойстиків або мишки.

Замість клавіатури може використовувати сенсорні екрани. У цьому разі усі елементи керування зображуються на екрані.

Операторські панелі можуть бути стаціонарними (рис. 6.1) та переносними (рис. 6.2), наприклад, для керування роботом або штабелером.

Рис. 6.1. Стаціонарні операторські панелі

Для підключення панелей можна використовувати дротові ба бездротові засоби зв’язку.

Рис. 6.2. Переносні операторські панелі з бездротовим та дротовим підключенням

Кнопкові панелі (Pushbutton panels (PP)) є альтернативою провідним операторським пультам управління.

Організація керування машиною рідко можлива без застосування таких апаратних елементів, як кнопки аварійного вимикання, лампи індикації або кнопкові вимикачі.

Нові панелі SIMATIC HMI Key роблять інтеграцію клавіш та ламп індикації в одному корпусі.

Елементи керування не повинні в такому випадку підключатися індивідуально, натомість панель керування підключається через мережу PROFINET.

Кожен модуль розширення може містити два KP8 PN або KP8F PN. При цьому доступні для індикації 5 кольорові світлодіоди, керовані по мережі PROFINET.

Комбінація пристрою SIMATIC HMI PRO та кнопкових панелей KP8 PN, KP8F PN дає наступні переваги:

- не потрібна шафа управління;
- немає необхідності у довгих кабелях;
- оптимальний для роботи дизайн.

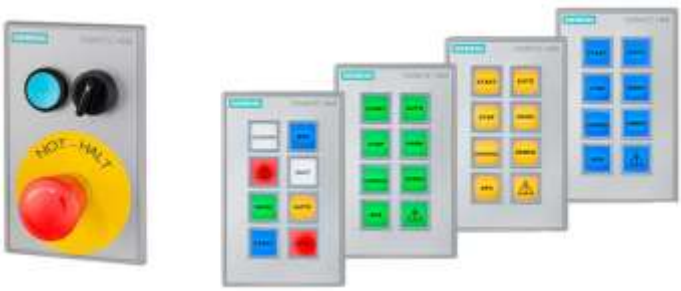


Рис. 6.3. Кнопкові панелі

Програмовані кнопкові панелі (Push Button Panels - PP) є інноваційною альтернативою для кнопкових пультів і панелей з проводовим монтажем (рис. 6.4).

Заздалегідь змонтовані та підготовлені до установки, оснащені вбудованим комунікаційним інтерфейсом, програмовані кнопкові панелі дозволяють суттєво скорочувати терміни розробки та монтажу нових систем управління

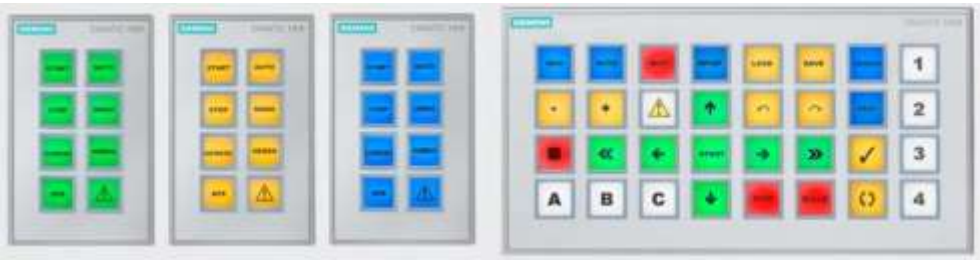


Рис. 6.4. Програмовані кнопкові панелі

Панелі операторів SIMATIC серії 70 призначені для вирішення простих завдань оперативного управління та моніторингу роботи окремих виробничих машин (рис. 6.5).


Рис. 6.5.

Вони забезпечені монохромними графічними STN дисплеями та вбудованою мембранною клавіатурою. Незначні установочні розміри та високий ступінь захисту фронтальної частини корпусу дозволяють вбудовувати панелі операторів безпосередньо у кероване обладнання.

Операторські панелі серії Basic є ідеальним рішенням для розробки моніторингу та операторського управління на рівні невеликих агрегатів і машин (рис. 6.6).


Рис. 6.6. Операторські панелі серії Basic

Вони пропонують піксельну графіку та однакові базові функції для всієї серії панелей із усіма розмірами дисплея.

Серія панелей включає прості клавішні панелі (KP), варіанти із сенсорними екранами (TP) та пристрої із сенсорним екраном та додатковими клавішами (KTP)

Портативні мобільні операторські панелі підтримують функції людино-машинного інтерфейсу безпосередньо за місцем роботи – у зоні прямої видимості та прямого доступу до технологічного процесу.

Вони можуть легко та надійно під'єднуватися під час роботи, тому можуть гнучко використовуватись на машинах та установках.

Незалежно від галузі застосування, якщо потрібна мобільність на стороні управління та моніторингу, мобільні панелі SIMATIC пропонують наступні вирішальні переваги: оператори машин або обслуговуючий персонал мають можливість працювати з найкращим оглядом робочого простору та процесу.

Навіть для великих виробництв, складних та закритих машин та конвеєрних ліній, мобільні панелі дозволяють робити швидке та точне налаштування обладнання.

Вони також зменшують час простою обладнання під час обслуговування чи ремонту.

Можливий зв'язок з ПЛК та іншими компонентами системи управління як по провідних, так і бездротових комунікаціях.

6.2. Операторські станції на основі персональних комп'ютерів.

Операторські станції призначені для відображення інформації у стаціонарних умовах та будуються на основі персональних комп’ютерів (рис. 6.7).

Рис. 6.7. Операторські станції

У промислових умовах використовують персональні комп’ютери з підвищеним рівнем захисту (промислові персональні комп’ютери).

Як правило ці комп’ютери встановлюють у шафах керування, тому широко використовують панельні комп’ютери з сенсорним екраном.

Клавіатура теж має підвищений захист.

Найчастіше забезпечується рівень захисту IP65.

Існує широкий асортимент міцних високоефективних панельних комп'ютерів SIMATIC Panel PCs, доступних для різних вимог (рис. 6.8).

Високоякісні компоненти та модулі гарантують 24-годинну роботу у розширеному температурному діапазоні.

Високий ступінь стійкості пристроїв до вібрацій та коливань завдяки спеціальному кріпленню жорсткого диска, фіксації роз'ємів і плат.

Міцна модель з високою електромагнітною сумісністю об'єднаними промисловими джерелами живлення.

Яскравий, висококонтрастний екран із різними розмірами до 19”Міцний корпус із захистом від пилу, вологості та хімічних речовин (ступінь захисту зовнішньої панелі IP65).

Для підключення ПЛК використовуються вбудовані порти Gigabit Ethernet та PROFIBUS DP/MPI.

Рис. 6.8. Операторські панелі на основі панельних комп'ютерів SIMATIC Panel PC

Конструкція панельних комп'ютерів SIMATIC розроблена для використання в промислових умовах. Так, наприклад, спеціальне, що захищає від впливу вібрації, кріплення жорстких дисків гарантує їхню нормальну роботу навіть в умовах впливу механічних навантажень.

Є версії панельних комп'ютерів з повністю закритим корпусом зі ступенем захисту IP65 з 15" і 19" сенсорними дисплеями, а також конструкції без частин, що обертаються (без вентилятора і жорсткого диска).

Комп'ютери SIMATIC Panel PC розраховані на роботу в умовах вібрації до 1g та ударних впливів до 5g.

Завдяки використанню процесорів Intel, починаючи від ULV (Ultra Low Voltage) до процесорів Intel Core, можливий різний вибір SIMATIC Panel PC для конкретної програми.

Можливість використання найпотужніших процесорівновітні технології процесорів Intel, а також процесорів Dual Core, ULV, Atom.

6.3. Інструментальні програмні засоби для відображення технологічного процесу.

ПЛК не мають умонтованих засобів відображення та введення інформації, тому для вирішення цієї задачі використовують спеціальні апаратно-програмні комплекси.

Програмний пакет, призначений для розробки або забезпечення роботи у реальному часі систем збору, обробки, відображення та архівування інформації про об'єкт моніторингу або управління називають SCADA (англ. *supervisory control and data acquisition, диспетчерське управління та збір даних*).

SCADA може бути частиною АСУ ТП та використовується там, де треба забезпечити оперативний контроль за технологічними процесами в реальному часі.

SCADA може бути самостійним програмно-апаратним комплексом, а також бути складною частиною або доповненням програмного забезпечення для проектування програмованих логічних контролерів.

ПЛК не мають умонтованих засобів відображення та введення інформації, тому для вирішення цієї задачі використовують спеціальні апаратно-програмні комплекси.

Апаратні компоненти для відображення та введення інформації можуть складатися з набору кнопок та світлових або символьних індикаторів, графічних індикаторів різного рівню складності з наборами кнопок або реалізовуватись у вигляді сенсорних дисплеїв, де органи керування відображаються на самому екрані. Загальною властивістю таких засобів є наявність стандартних каналів зв'язку, наприклад, інтерфейсів для підключення до локальних мереж, а також можливість програмного налагодження функцій керування та відображення. Ці апаратні компоненти мають свій обчислювальний пристрій та визначаються як операторські панелі.

Засоби відображення та введення інформації можна створити на основі персональних комп'ютерів (так звані операторські станції).

Засоби програмування засобів відображення та введення інформації залежать від складності апаратних компонент та технологічного процесу, що відображається.

Програмування операторських панелей може здійснюватися програмними засобами, які умонтовані у програмну середу програмування ПЛК, наприклад, середа розробки Simatic Step 7 Basic, яка призначена для програмування ПЛК S7-1200, а також операторських панелей Basic Panels.

Середа проектування програмування засобів відображення та введення інформації може використовуватися як для операторських панелей, так і для операторських станцій, наприклад, середа проектування WinCC flexible.

Для програмування операторських станцій використовують досить складні системи проектування, які дозволяють одноразово проектувати декілька операторських станції з різними рівнями доступу.

Для відображення можна використовувати досить велику кількість екранів (зображень) для відображення усього технологічного процесу та окремих його компонентів та пристроїв.

Для зображення інформації використовують різні графічні елементи, які знаходяться у палітрі елементів.

Такими елементами можуть бути, наприклад, графічні та числові поля для відображення даних, кнопки та інші елементи керування (рис. 6.9).

Рис. 6.9. Графічні елементи для зображення інформації

6.4. Інструментальні засоби візуалізації з використанням операторських панелей.

Деякі ПЛК мають у комплекті засоби відображення та введення інформації, наприклад, ПЛК SIMATIC S7-1200.

Рис. 6.10. Засоби відображення та введення інформації у складі ПЛК SIMATIC S7-1200

Система проектування STEP 7 Basic, що призначена для проектування систем автоматизації на основі ПЛК SIMATIC S7-1200 має інструментальних засіб WinCC Basic (рис. 6.11).

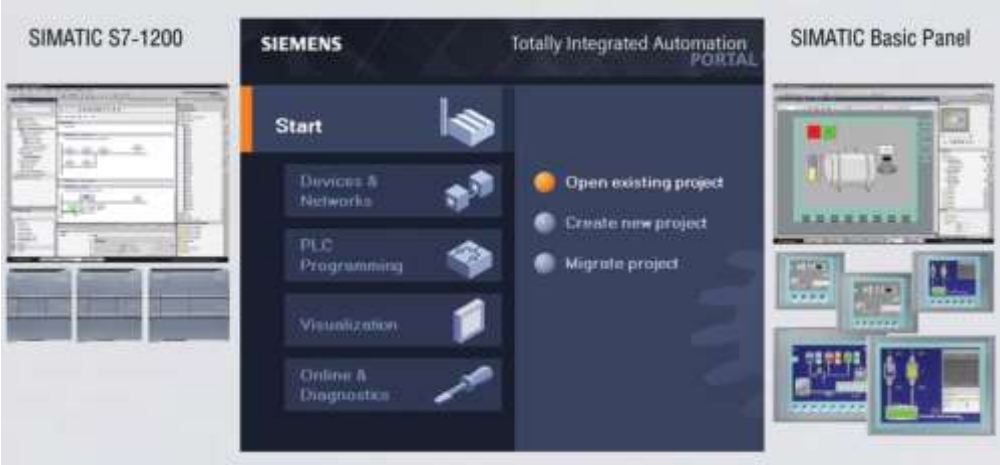


Рис. 6.11. Програмне забезпечення STEP 7 Basic

WinCC Basic включає пакети, що дозволяють здійснювати встановлення конфігурації базових панелей SIMATIC, які використовуються з ПЛК S7-1200 та забезпечують оптимальну взаємодію систем проектування ПЛК та людино-машинного інтерфейсу.

До основних переваг пакету можна віднести:

Підтримку концепції багаторазового використання усіх компонентів бібліотек у проекті.

Підтримку механізмів Drag & Drop для передання даних між різноманітними редакторами для ПЛК та приборів людино-машинного інтерфейсу.

Наявність єдиної бази даних проекту с однаковим набором символьних імен.

Швидкий доступ до усіх задач автоматизації, включаючи інтерактивну роботу з системою автоматизації та її діагностику.

Просте графічне створення конфігурації апаратури и мережевих структур у середі одного редактора.

Наявність простого и інтуїтивно зрозумілого інтерфейсу користувача для забезпечення доступу до різних варіантів відображення інформації та редакторам.

На рис. 6.12 наведений приклад відображення програми керування конвеєром.

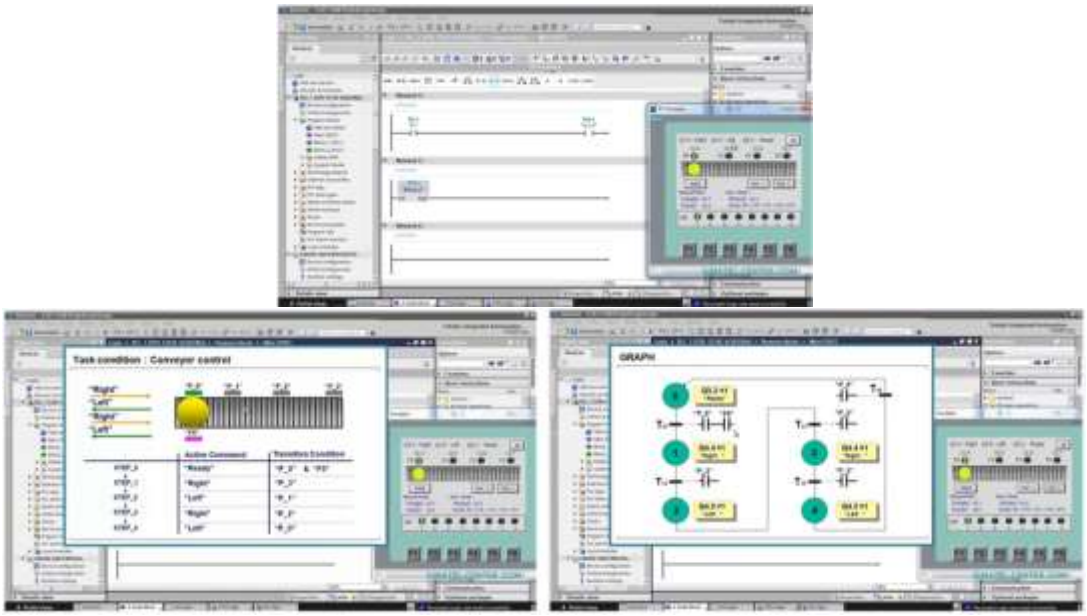


Рис. 6.12. Приклад відображення програми керування конвеєром

Редактори візуалізації мають такі можливості:

– створення конфігурації базових панелей операторів;

- використання готових екранних зображень для роботи з сенсорною або мембранною клавіатурою;
- підтримка дискретних та аналогових аварійних повідомлень;
- одночасне використання декількох мов;
- графічна бібліотека стандартних зображень об'єктів;
- інтелектуальні функції Drag&Drop для встановлення конфігурації стандартних функцій людино-машинного інтерфейсу.

Існують також засоби проектування для різних апаратних компонент візуалізації, включаючи персональні комп'ютери.

Прикладом такої середи проектування для операторських панелей та операторських станцій є WinCC flexible фірми SIEMENS, яка складається з структури проекту, області проектування, де відкриваються елементи структури, вікна властивостей елементів області проектування, вікна інструментів, за допомогою яких здійснюється проектування. та вікна виводу, в яке виводяться повідомлення про хід проектування (рис. 6.13).

WinCC flexible складається з засобів проектування, за допомогою яких здійснюються усі завдання відображення та керування (Engineering – ES) , а також з засобів, які дають можливість використовувати створену систему відображення та керування на персональних комп'ютерах (Runtime - RT).

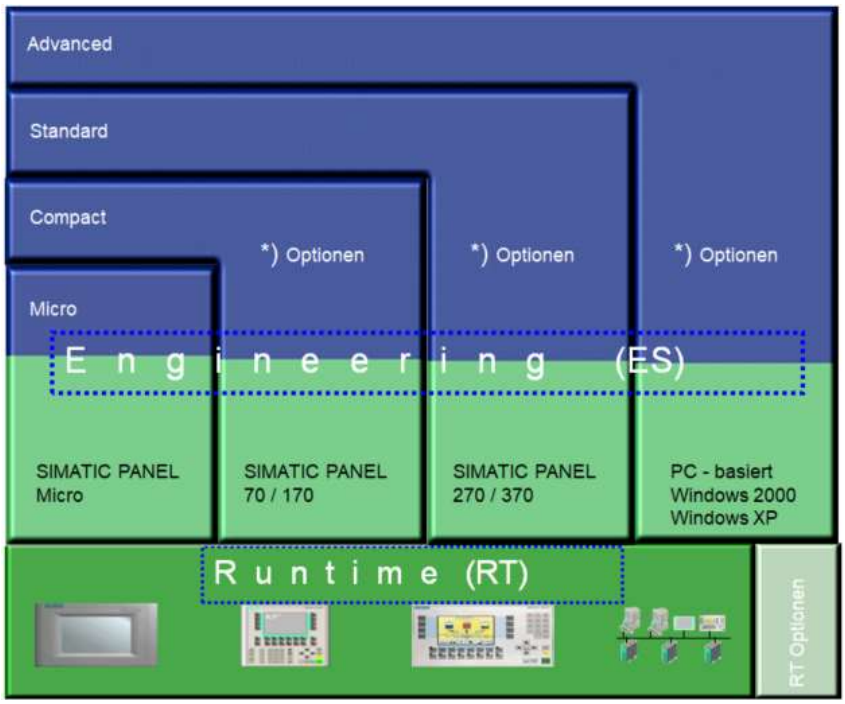


Рис. 6.13. WinCC flexible

Для відображення можна використовувати досить велику кількість екранів (зображень) для відображення усього технологічного процесу та окремих його компонентів та пристроїв. Для зображення інформації використовують різні графічні елементи, які знаходяться у палітрі елементів. Такими елементами можуть бути, наприклад, графічні та числові поля для відображення даних, кнопки та інші елементи керування (рис. 6.14).

Рис. 6.14. Засоби зображення інформації

На рис. 6.15 наведені приклади встановлення на екрані елементів вводу-виводу та кнопок.

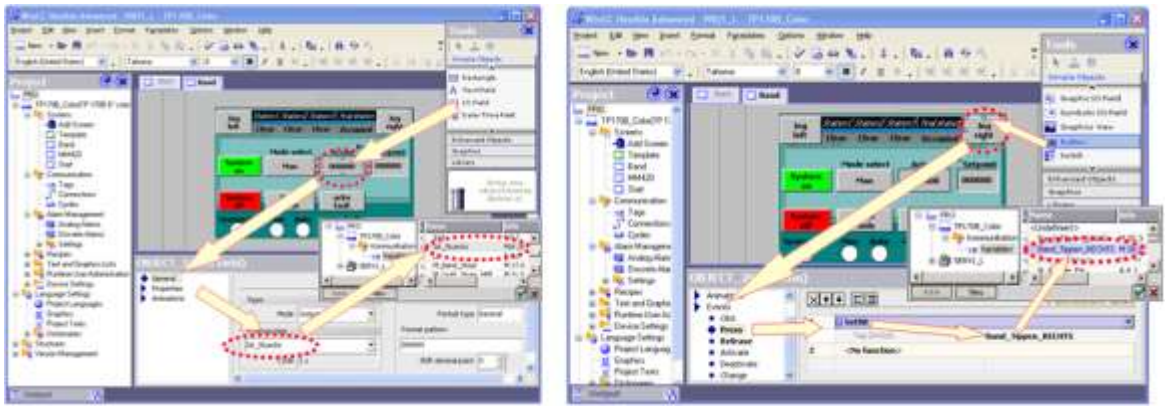


Рис. 6.15. приклади встановлення на екрані елементів вводу-виводу (зліва), та кнопок (справа)

На рис. 6.16 наведений приклад зображення складського обладнання з полями графічного представлення даних у вигляді стелажу з вантажем та штабелером та кнопками керування переміщенням у ручному режимі.

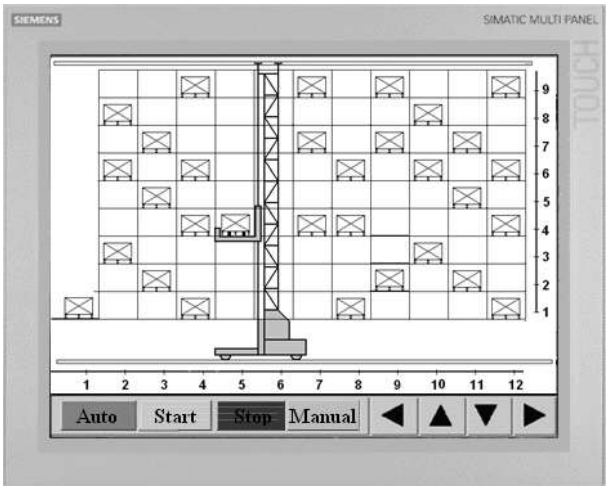


Рис. 6.16. Приклад зображення складського обладнання

Для відображення складних пристроїв існують різні бібліотеки з компонентами технологічного обладнання для різних задач. На рис. 6.17 наведені приклади таких елементів. Для цих елементів є можливість встановлення динамічних властивостей, наприклад, зміни кольору, розміру та положення в залежності від окремих параметрів, графічного зображення рівня заповнення у резервуарах, та інші.

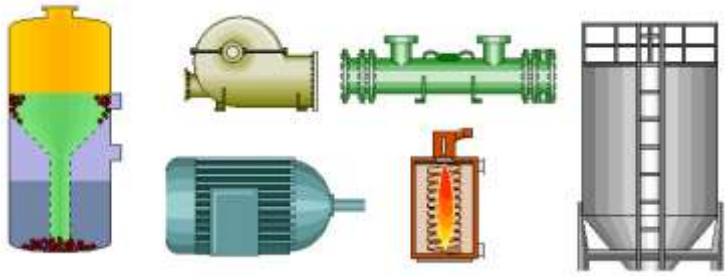


Рис. 6.17. Приклади елементів технологічного обладнання

При необхідності створення складної структури екранів проектування треба виконувати у такій послідовності:

- планування структури екранів проекту,
- створення екранів,
- створення системи навігації по екранах,
- конфігурування змісту екранів.

У разі виникнення якої-небудь події є можливість виводити на екран відповідні повідомлення.

Розрізняють дискретні, аналогові повідомлення, повідомлення за номером та системні повідомлення.

Дискретні повідомлення з'являються, коли відповідний біт пам'яті встановлюється в 1.

Аналогові повідомлення пов’язані з виходом числових значень у пам'яті ПЛК за встановлені межі.

Ці повідомлення видає операторська панель шляхом періодичного опитування відповідних змінних у ПЛК.

Повідомлення за номером ініціює ПЛК. У разі виникнення події він надсилає відповідний номер, за яким операторська панель видає повідомлення.

Системні повідомлення може надсилати панель та ПЛК.

На рис. 6.18 наведені типи повідомлень у пристроях відображення.

Рис. 6.18. Типи повідомлень у пристроях відображення

Пристрої відображення також дають можливість робити архіви значень процесу та повідомлень для збереження їх у самому пристрої або у комп'ютері верхнього рівня керування. Для документування значень процесу та повідомлень використовується система звітів. Ці архіви можна використовувати, наприклад, для зберігання інформації о наявності та кількості товарів у окремих стелажах та окремих комірках стелажів.

Більше можливостей у створенні систем візуалізації дає система людино-машинного інтерфеса WinCC (Windows Control Center). Ця система може бути розрахована на багато користувачів з використанням конфігурації "сервер-клієнт". Така конфігурація дозволяє підключити до одного сервера до 32 станцій управління. Проектування сервера може здійснюватися з клієнта.

Засоби візуалізації можуть давати різноманітну інформацію про стан і положення окремих машин як у графічному так і числовому вигляді, наприклад, для автоматизованої складської системи – положення крана-штабелера, про зміст комірок стелажу з вказівкою на вид продукту (наприклад, код продукту), та кількість товару даного типу у комірці.

Шляхом переміщення вказівника комірки (наприклад, шляхом зміни кольору вказаної комірки) за допомогою кнопок переміщення можна получити детальну інформацію о наявності продукції у комірці, типу продукції та її кількості.

Використовуючи засоби динамізації можна показати зміну стану складу або відображення рівня рідини у резервуарах на складі зрідженої продукції (рис. 6.19).

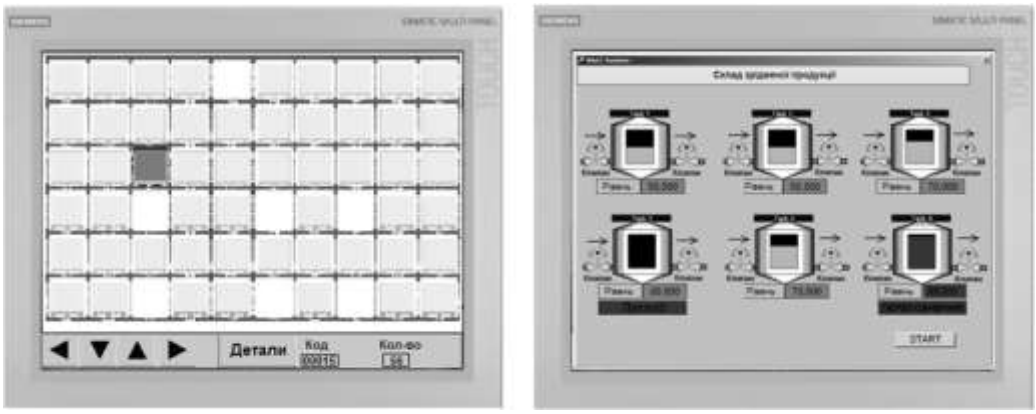


Рис. 6.19. Відображення зміни стану складу та відображення рівня рідини у резервуарах

Для гнучкого керування промисловим обладнанням можна використовувати таку функцію, як створення рецептів, яка є складовою частиною таких систем спостереження та керування, як, наприклад, WinCC flexible фірми SIEMENS.

Рецепти використовуються в тих випадках, коли необхідно передавати взаємопов'язані дані у вигляді записів даних між операторської станцією і ПЛК. При цьому виходять з того, що одночасно використовуються не всі записи даних для рецепта, тому немає необхідності зберігати її усю в системі управління (ПЛК). Ця інформація, яка вимагає багато місця в пам'яті, може зберігатися в операторських станціях, а при необхідності окремі записи даних можуть передаватися або зчитуватися в ПЛК.

Ця функція має стандартні вікна для формування рецепту, де треба вказати компоненти рецепту та їхню кількість у визначеній розмірності.

Основне завдання, яке вирішують рецепти, це визначення кількості окремих компонентів, які складається продукт.

Наприклад, при виробленні напоїв можуть виготовлятися різні варіанти кінцевого продукту (сік, нектар, напій), що відрізняються тільки кількістю окремих компонентів (вода, концентрат, цукор та ароматизатор), які можна задавати рецептом (рис. 6.20).

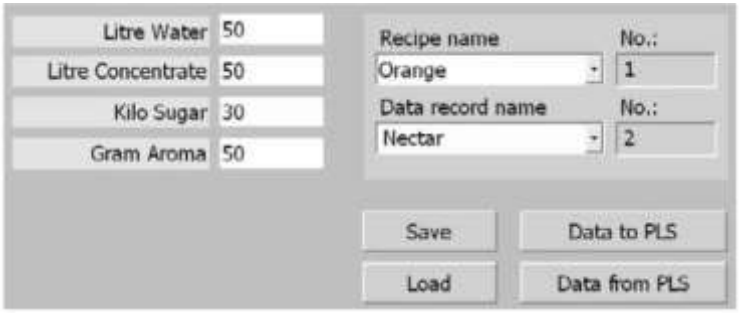


Рис. 6.20. Рецепт

Для переходу з одного варіанту кінцевого продукту на інший у цьому разі треба лише змінити рецепт, після чого автоматично здійсниться переналагодження технологічного обладнання, що можна використати також для формування заказу на автоматизованому складі.

Ця функція дає можливість сформувати заказ (у вигляді так званого рецепту), де треба вказати компоненти заказу та їх кількість.

На рис. 6.21 показаний приклад формування заказу на автоматизованому складі за допомогою функції створення рецепту.

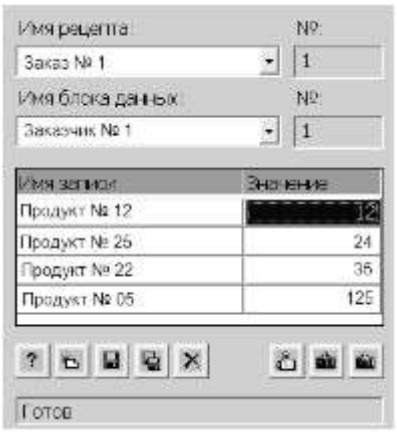


Рис. 6.21. Приклад формування заказу за допомогою функції створення рецепту

Рецепти можна використовувати для переналагодження послідовності технологічних операцій.

На рис. 6.22 наведений приклад встановлення послідовності та часу проведення операцій на лінії обробки.

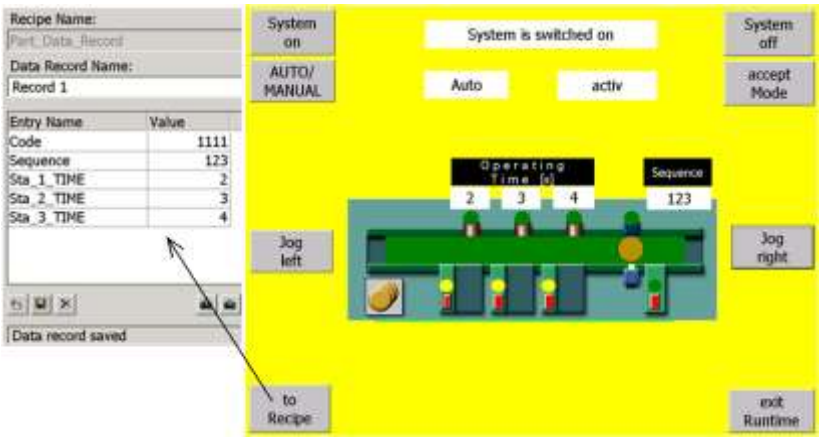


Рис. 6.22. Приклад встановлення послідовності та часу проведення операцій на лінії обробки

Конвеєр послідовно переміщує деталь за вказані робочі місця (1, 2, 3) та затримується на кожному місці згідно з встановленим часом.

На прикладі задана послідовність переміщення 1, 3, 2 з затримкою на відповідних місцях 2, 4, та 3 с.

Цей підхід дає можливість швидкого переналагодження процесу виготовлення виробів шляхом вибору відповідного набору параметрів технологічного процесу, які задаються заздалегідь підготовленими рецептами.

У разі потреби можна швидко підготувати нові рецепти шляхом внесення нових значень для компонентів продукту або параметрів процесу.

6.5. Інструментальні засоби візуалізації на основі персональних комп'ютерів.

Існують системи проектування засобів відображення та керування технологічними процесами, що призначені тільки для персональних комп'ютерів (операторських станцій).

Прикладом такої середи проектування для операторських станцій є програмний комплекс WinCC фірми SIEMENS.

WinCC є одною з основних частин системи автоматизації SIMATIC, що призначена для організації людино-машинного інтерфейсу HMI та візуалізації різних процесів (рис. 6.23.

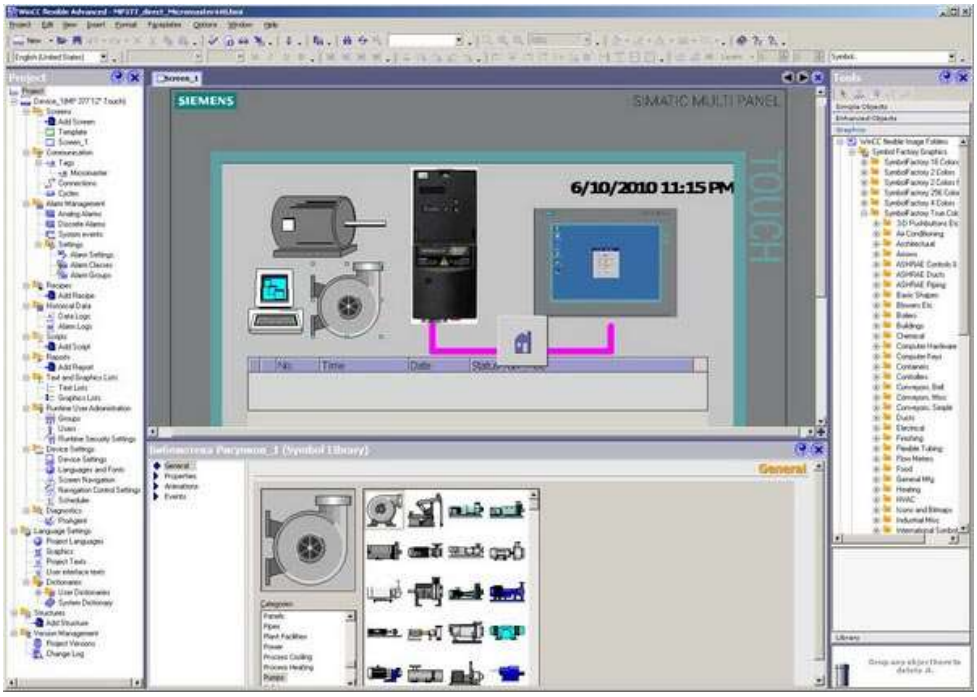


Рис. 6.23. Програмний комплекс WinCC

Програмний комплекс SIMATIC Windows Control Center (або WinCC) надає всі необхідні засоби для найкращого управління процесами операційних систем Microsoft Windows.

Цей пакет не орієнтований на певну промислову або технічну область і може використовуватися у всіх сферах і на різних етапах виробництва.

Крім того WinCC є повністю відкритим додатком, здатним працювати і зі стандартними програмами, і з розробками користувача.

Розширення можливостей комплексу можливе за рахунок застосування скриптів, написаних на VBS, ANSI C та VBA.

Програмне середовище включає всі властиві SCADA-системам функції, що дозволяють проводити повну поетапну графічну візуалізацію техпроцесів, створювати звіти і квитувати події, реєструвати повідомлення і значення вимірюваних величин, фіксувати і архівувати дані, управляти користувачами та правами їх доступу.

Додаток відстежує та відзначає кожну операцію та кожну подію, яка надає хоч якийсь вплив на загальний хід, здійснюючи таким чином постійний контроль якості.

Крім вищезгаданого SIMATIC WinCC включає великі бібліотеки, потужні засоби для обробки величезних масивів даних, зручний об'єктно-орієнтований графічний модуль з індивідуальними налаштуваннями. Є можливість внесення оперативних змін до проекту в онлайн-режимі.

Середовище розробки підтримує побудову резервованих систем, взаємодіє з комплексом SIMATIC Step 7, здатне легко інтегруватися у внутрішню інформаційну мережу будь-якого підприємства

Програмне забезпечення надає користувачам цілу низку унікальних інтерфейсів та редакторів, що використовуються для персонального визначення можливостей проекту. Основними з них є:

- редактор кадрів процесів та діалогових вікон WinCC Graphics Designer;
- центр керування проектом WinCC Explorer;
- редактор рядкових/сторінкових шаблонів журналів реєстрацій WinCC Report Designer;
- конфігуратор системи архівування WinCC Tag Logging;
- модуль системи оперативних та аварійних повідомлень WinCC Alarm Logging.

Крім того, кожен модуль SIMATIC WinCC має свої власні програмні помічники – так звані Майстри (Wizards), які сприяють виконанню стандартних операцій (рис. 6.24).

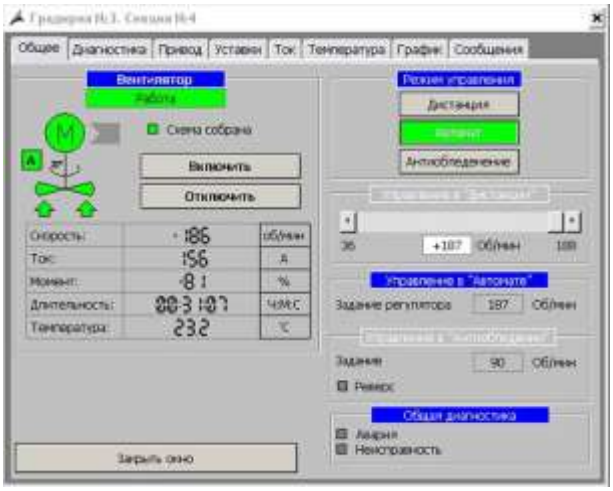


Рис. 6.24. Програмний помічник Wizards

На рис. 6.25 наведені основні функції WinCC.

Рис. 6.25. Основні функції WinCC

WinCC дозволяє використовувати як одно- так і багато користувальницькі системи.

Однокористувальницькі системи здійснюють відображення та керування простими технологічними процесами або окремими одиницями обладнання.

Багатокористувальницькі системи на основі конфігурації «сервер-кліент» дають можливість підключити к серверу до 32 операторських станцій з можливістю резервування сервера для зберігання інформації.

На рис. 6.26 показаний приклад складання кадрів з використанням палітри графічних об'єктів та бібліотеки.

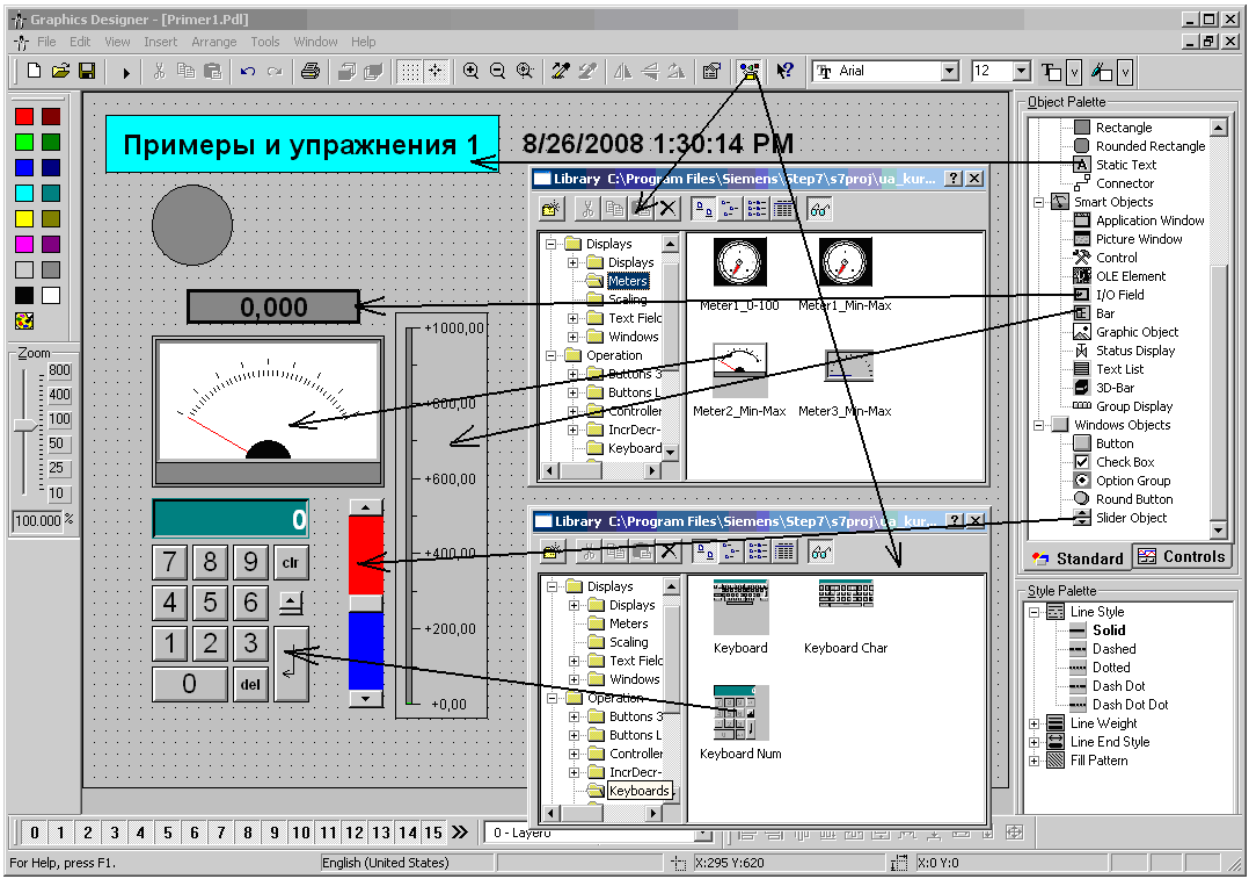


Рис. 6.26. Складання кадрів з використанням палітри графічних об'єктів та бібліотеки

На рис. 6.27 наведені форми представлення даних.

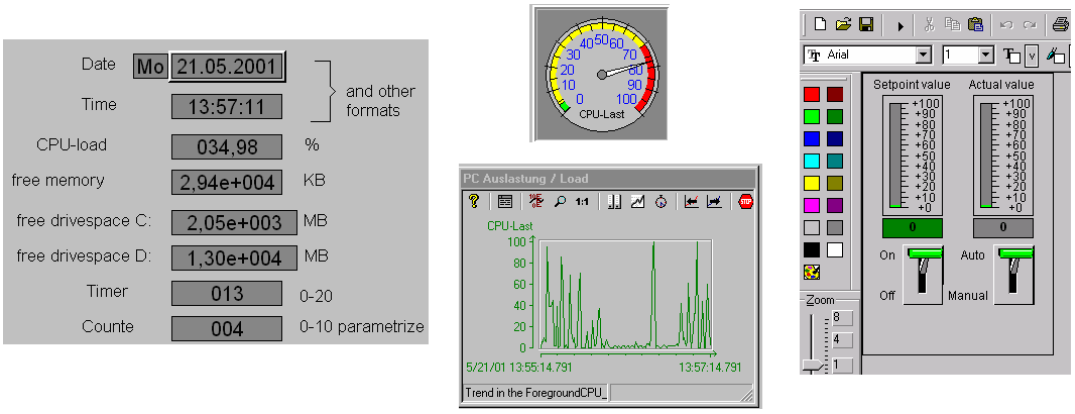


Рис. 6.27. Форми представлення даних

На рис. 6.28 наведений приклад системи керування та обслуговування підйомного крана на основі WinCC.

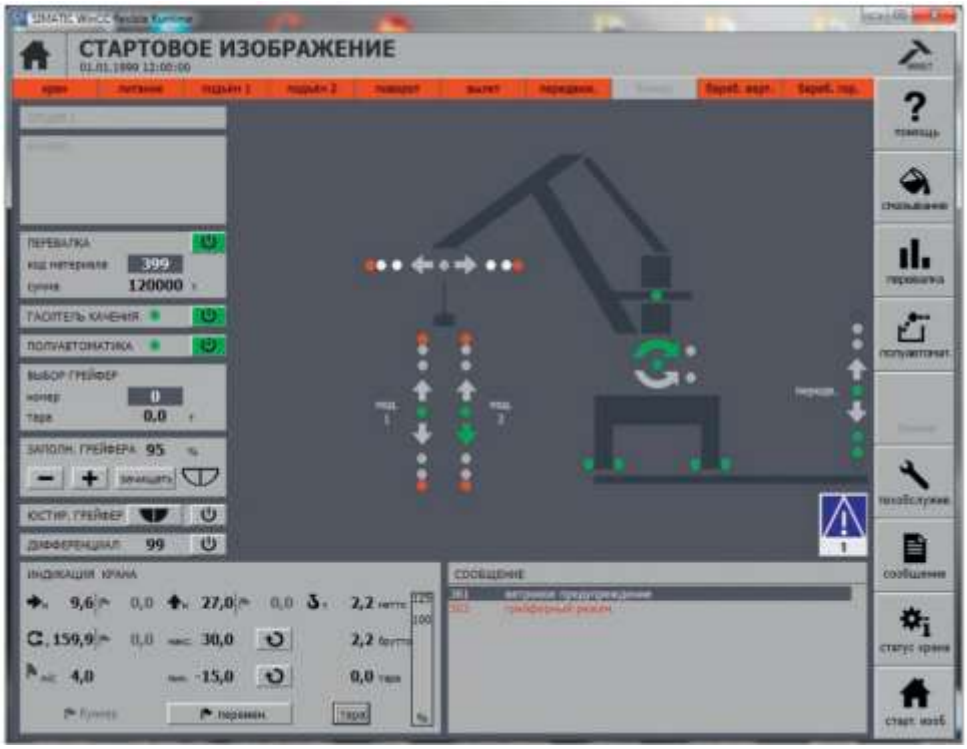


Рис. 6.28. Система керування та обслуговування підйомного крана на основі WinCC

Контрольні запитання

1. Визначити, для чого використовують системи відображення та керування.
2. Визначити, чим відрізняються строкові та графічні операторські панелі.
3. Назвати, які переваги дає сенсорний екран.
4. Визначити, чим відрізняються операторські панелі та операторські станції.
5. Назвати, який рівень захисту найчастіше забезпечують операторські панелі та операторські станції.
6. Розповісти, як здійснюється проектування систем візуалізації процесу.
7. у яких формах здійснюється введення та представлення даних.
8. Розповісти, які типи повідомлень використовують у системах візуалізації процесу.
9. Описати, для чого використовувати функцію створення рецептів.
10. Визначити, для чого використовується палітри графічних об'єктів та бібліотеки символів.

7. Проєктування програмних елементів за допомогою мов програмування роботехнічних пристроїв

7.1. Програмування роботів фірми KUKA за допомогою мови KRL

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

Різні фірми використовують різні мови програмування роботів, які найчастіше є складовою частиною програмних комплексів конструювання та проєктування робототехнічних пристроїв побудованих на модульному принципі.

Для програмування роботів фірми KUKA використовується мова програмування KRL, яка є частиною програмного комплексу, призначеного для проєктування та моделювання роботів фірми KUKA.

Ця мова дає можливість створити програми з усіма типами програмного керування, а саме циклового, позиційного та контурного.

Для цього у програмі можуть бути запрограмовані такі переміщення:

- переміщення від точки до точки (PTP) (рис. 7.1, а),
- лінійне переміщення (LIN) (рис. 7.1, б),
- кругове переміщення (CIRC) (рис. 7.1, в).

а) б) в)

Рис. 7.1. Функції переміщення: переміщення від точки до точки (PTP) (а), лінійне переміщення (LIN) (б), кругове переміщення (CIRC) (в)

Мова програмування KRL має структуру у вигляді послідовності кадрів.

```
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

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.

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 основних команд переміщення.

Переміщення від точки до точки (PTP)

Робот виконує переміщення робочого органу уздовж найбільш швидкої траєкторії до цільової точці. Так як осі робота здійснюють обертальний рух, криволінійні траєкторії можуть виконуватися швидше, ніж прямі, тому найбільш швидкою траєкторією, найчастіше є не пряма (рис. 7.1, а).

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. 7.1, б).

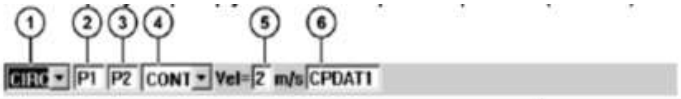
Формуляр лінійного переміщення (LIN).

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, допоміжною і цільовою точкою (рис. 7.1, в).

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), наведені на рис. 7.2.

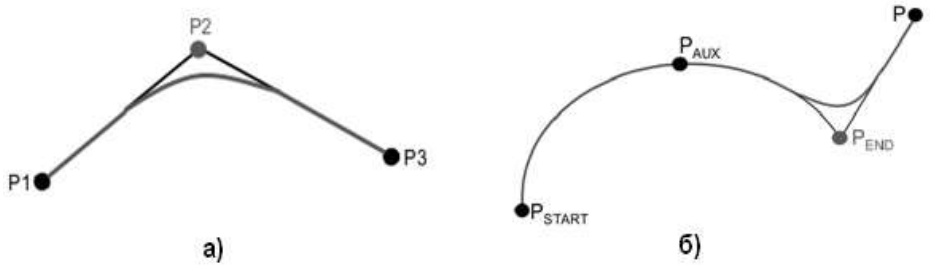


Рис. 7.2. Приклади згладжування для переміщень LIN (а) та CIRC (б)

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

На рис. 7.3 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
```

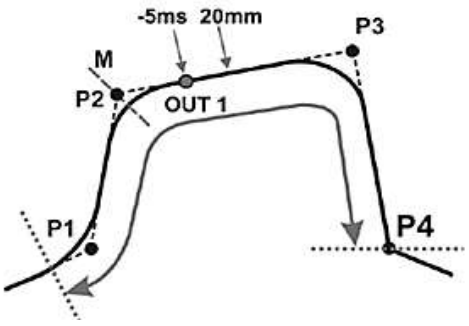


Рис. 7.3. Програма та відповідна траєкторія переміщення

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)

.

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).

Поз.	Опис	Діапазон значень
1	Час очікування	≥ 0 сек

Мова програмування KRL є складовою частиною п**рограмного комплексу** для проєктування та моделювання роботів фірми KUKA (рис. 7.4).

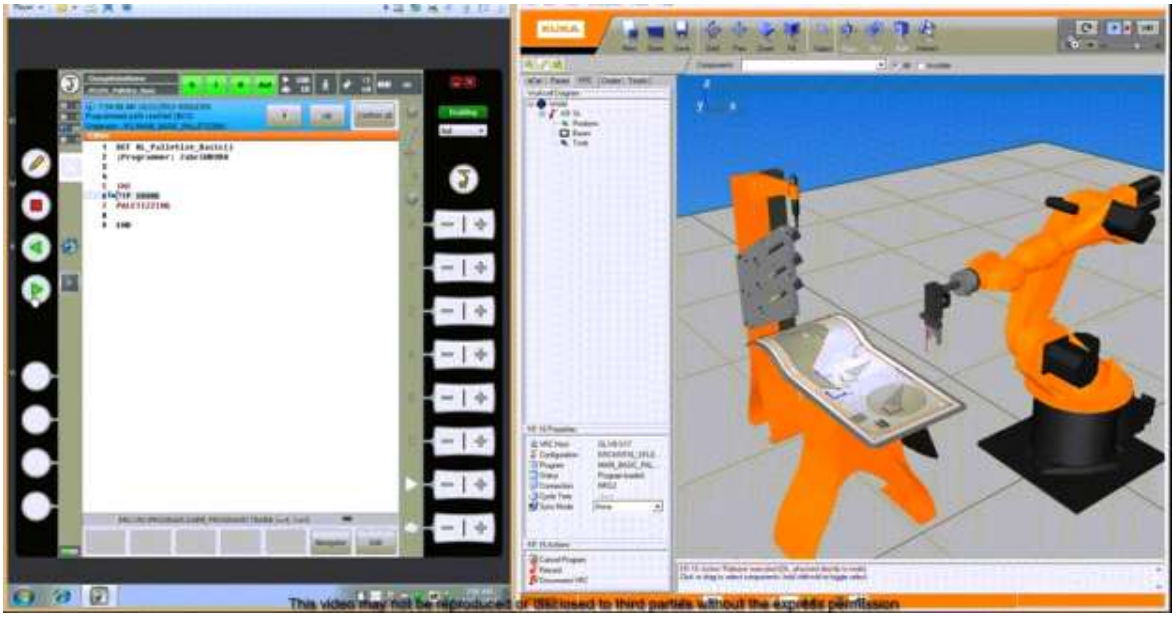


Рис. 7.4. Мова програмування KRL як частина п**рограмного комплексу** для проєктування та моделювання роботів фірми KUKA

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.

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 команди переміщення наведені у табл. 7.1.

Табл. 7.1

Команди переміщення

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 прикладу 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. 7.5.

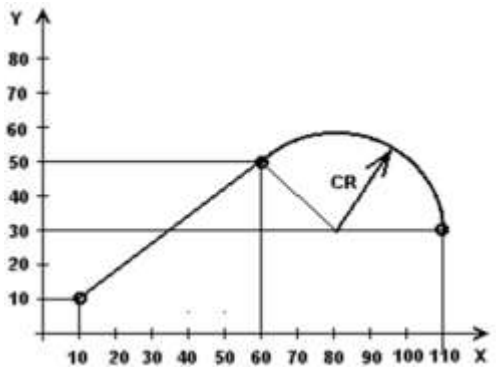


Рис. 7.5. Траєкторія переміщення

Відповідна програма має такий вигляд:

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

Для контролерів **Arduino,** які широко використовуються для керування промисловими машинами, а саме, простими верстатами ЧПУ, пристроїв для плазмового, газового та лазерного різання, 3D-принтерами, були розроблені апаратні та програмні компоненти, що дають можливість реалізувати цю задачу.

Розглянемо деякі з них.

Для реалізації контурного керування найчастіше використовують крокові двигуни, які забезпечують високу точність позиціонування.

Для керування кроковими двигунами розроблений модуль **CNC shield,** який встановлюється на контролер **Arduino UNO** (рис. 7.6).

Рис. 7.6. Модуль **CNC shield,** встановлений на контролер **Arduino UNO**

Модуль **CNC shield** здійснює керування чотирма кроковими двигунами (4 осі - X, Y, Z, A) та додатковим обладнанням за допомогою цифрових входів та виходів, наприклад, кнопки керування, кінцеві вимикачі, додаткові виконавчі пристрої тощо.

Можна використовувати драйвери A4988 або DRV8825.

Апаратними та програмними засобами можна здійснити налагодження режимів роботи.

Наприклад, за допомогою перемикачів встановлюється мікрокроковий режим крокових двигунів (повний крок, 1/2 кроку, 1/4 кроку, 1/8 кроку, 1/16 кроку).

На наступному слайді наведена схема підключення крокового двигуна до драйвера A4988.

На рис. 7.7 наведена схема підключення драйверів A4988 з кроковими двигунами та інформаційних пристроїв до контролера Arduino UNO.

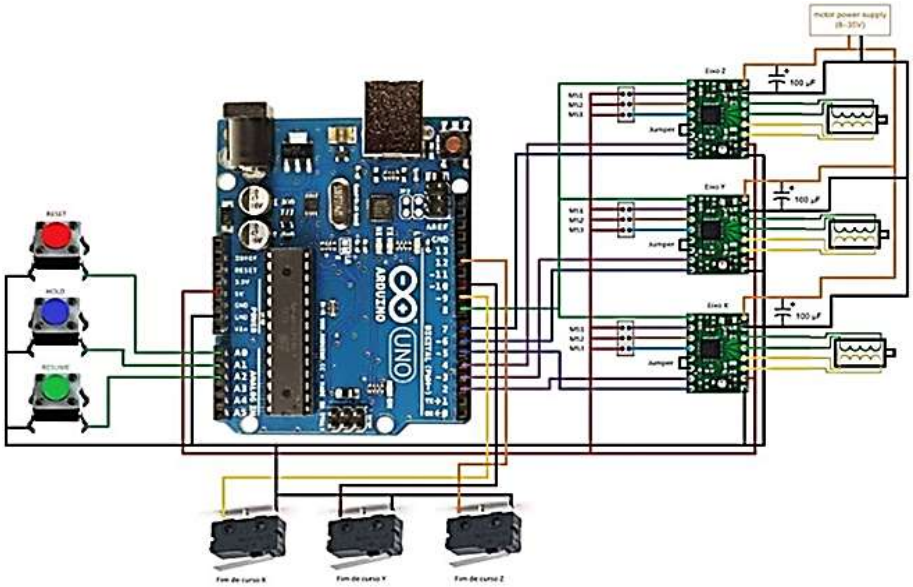


Рис. 7.7. Схема підключення драйверів A4988 з кроковими двигунами та інформаційних пристроїв до контролера Arduino UNO

Для створення програми контурного керування за допомогою мови G-команд для контролера Arduino UNO з модулем **CNC shield** та драйверами A4988 використовується Grbl Controller у вигляді бібліотеки **grbl-master**.

Вказана бібліотека встановлюється в програмний комплекс **Arduino IDE**.

Для використання бібліотеки **grbl-master** треба здійснити компіляцію, після чого встановлюється **GRBL controller** (рис. 7.8)

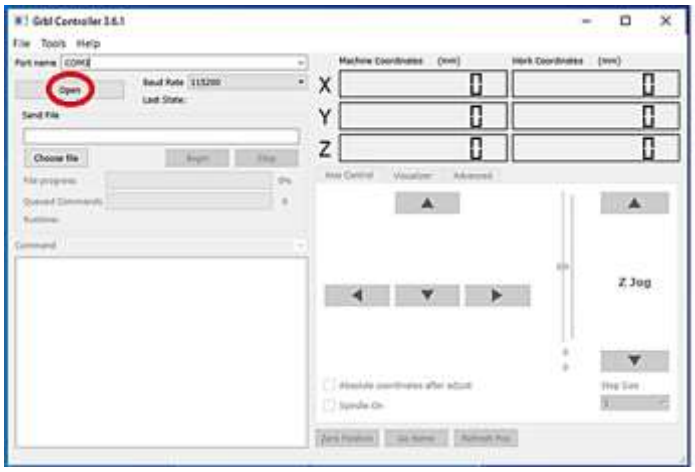


Рис. 7.8. GRBL controller

На рис. 7.9 показано, як виконується кадр **G1 G91 X10 Y10 F100** у полі команд, в результаті чого отримаємо лінійне переміщення з вихідної точки (X=0 Y=0) в задану позицію (X=10 Y=10).

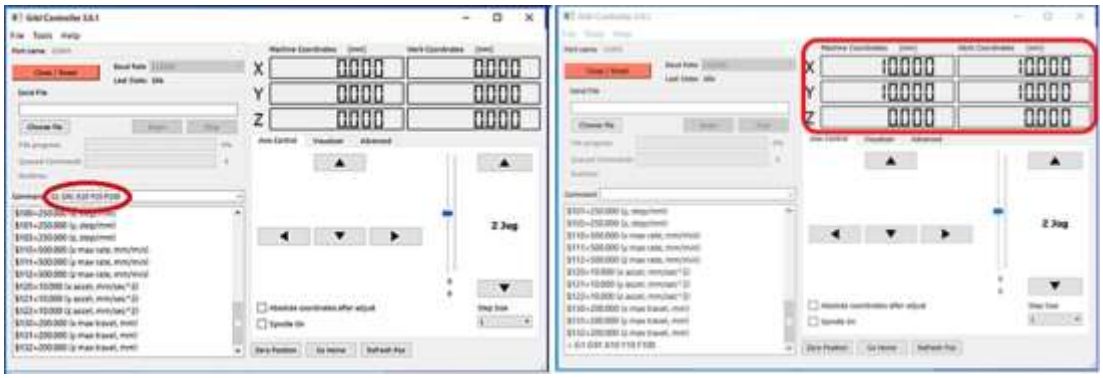


Рис. 7.9. Виконання кадру **G1 G91 X10 Y10 F100** у полі команд

В Інтернеті можна знайти багато прикладів пристрої з контурним керуванням на основі CD-приводів.

На рис. 7.9 наведений пристрій для малювання на основі CD-приводів. Замінивши виконавчий пристрій можна створити верстат для лазерного гравірування.

Рис. 7.9. Пристрій для малювання на основі CD-приводів

7.3. Конструювання робототехнічних пристроїв за допомогою програмної середи LabVIEW

Універсальним засобом проєктування робототехнічних пристроїв є програмна середа LabVIEW, яка здійснює програмування у графічному вигляді та має вмонтовані засоби для програмування пристроїв на основі різних платформ.

Відмінною рисою LabView є пристосованість середовища для розробки апаратних засобів.

Велика кількість бібліотек, розроблених для взаємодії з обладнанням різних виробників, дозволяє дуже швидко та комфортно прототипувати рішення різних завдань, у тому числі для керування роботами та системами машинного зору.

Крім офіційного LabVIEW Robotics Module, що пропонує National Instruments, і містить бібліотеки для взаємодії з різними робототехнічними компонентами, ентузіасти самі розробляють велику кількість бібліотек і, що найцінніше, поширюють їх безкоштовно.

На сьогоднішній день можна побачити велику кількість рішень у робототехніці з використанням LabVIEW.

На рис. 7.10 наведені особливості програмної середи LabVIEW.

Рис. 7.10. Особливості 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. 7.11).

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.

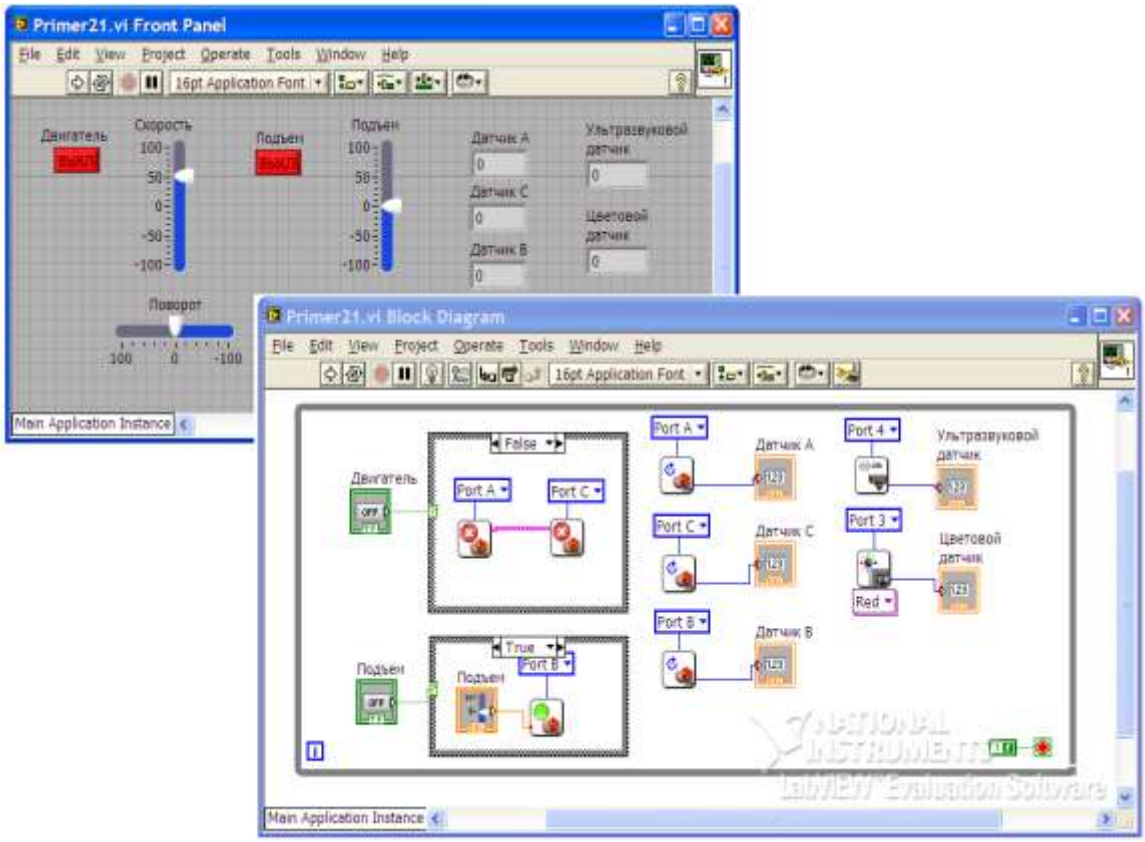


Рис. 7.11. Фронтальна панель та блок-діаграма

Фронтальна панель створюється за допомогою палітри елементів (Controls) (рис. 7.12).

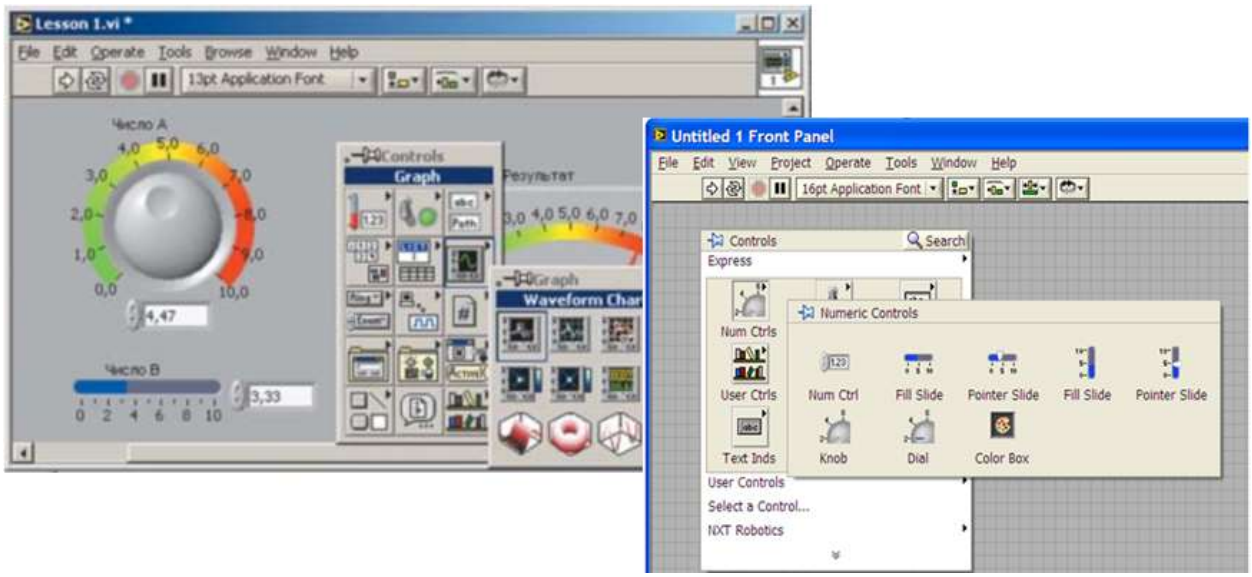


Рис. 7.12. Фронтальна панель та палітра елементів

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.

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. 7.13 показаний порядок встановлення властивостей елементів фронтальної панелі.

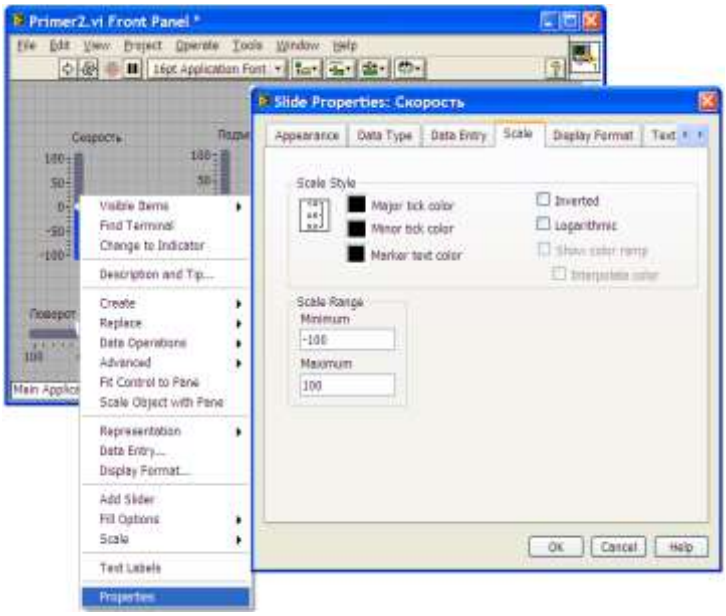


Рис. 7.13. Порядок встановлення властивостей елементів фронтальної панелі

При натисканні правою кнопкою миші відкривається спливаюче меню, де треба вибрати пункт **Properties**. Після цього відкривається вікно для встановлення властивостей вибраного елемента.

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. 7.14.

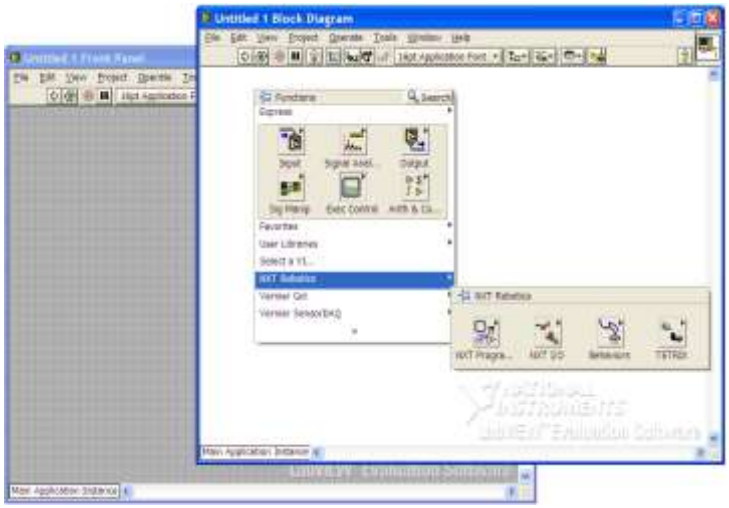


Рис. 7.14. Палітра функцій блок-діаграми

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.

На рис. 7.15 наведена програма у вигляді блок-діаграми.

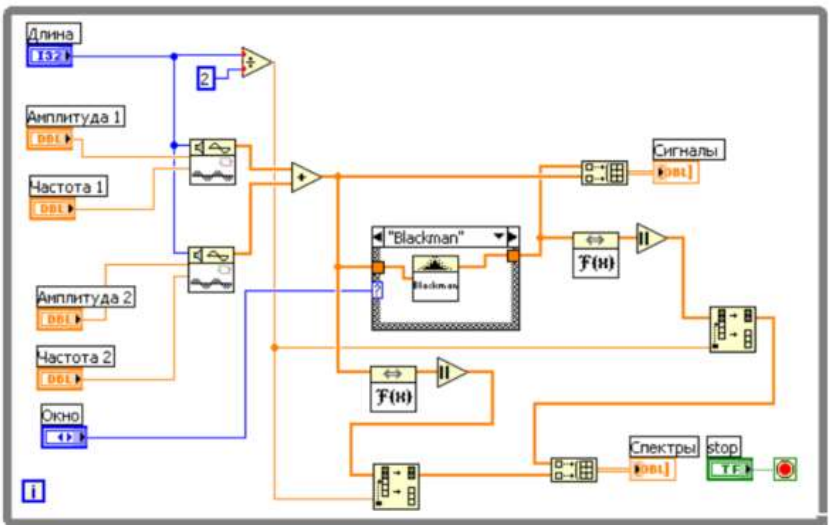


Рис. 7.15. Програма у вигляді блок-діаграми

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

У прямому режимі пристрій керування роботом підключений до комп'ютера через USB або Bluetooth, і програма виконується на комп'ютері.

Таким чином, пристрій керування роботом використовується як периферійний пристрій.

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

При цьому вже немає необхідності в підключенні пристрою керування роботом 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. 7.16 наведені палітри з функціями, що використовуються при проєктуванні роботів.

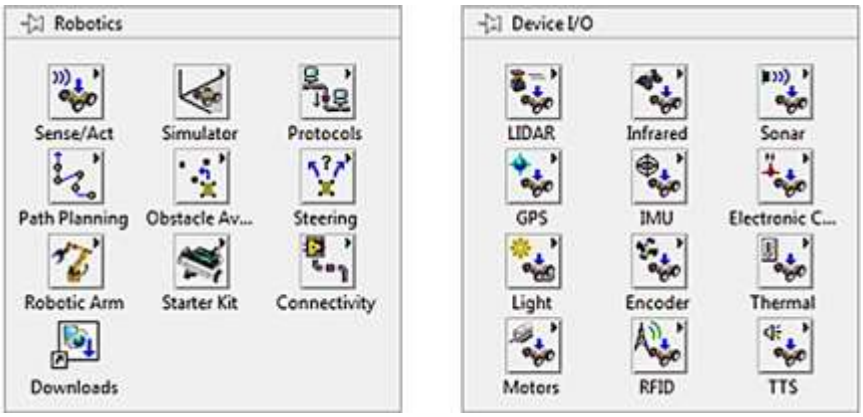


Рис. 7.16. Палітри з функціями, що використовуються при проєктуванні роботів

Приклади проєктування робототехнічних систем на основі програмної середи LabVIEW

На рис. 7.17 наведено вікно для створення проекту з роботами Mitsubishi Robotics, яка дозволяє здійснити створення програми у середі LabVIEW.

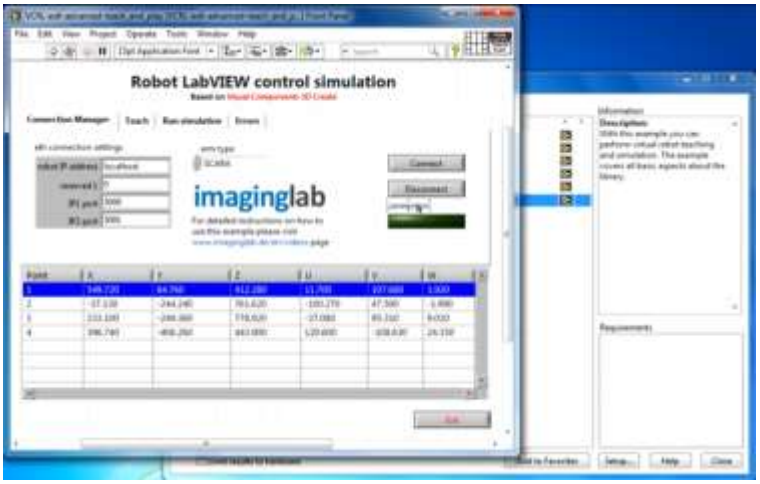


Рис. 7.17. Вікно для створення проекту з роботами Mitsubishi Robotics

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

На рис. 7.18 наведено вікно для визначення за допомогою бібліотеки обраного робота Mitsubishi Robotics

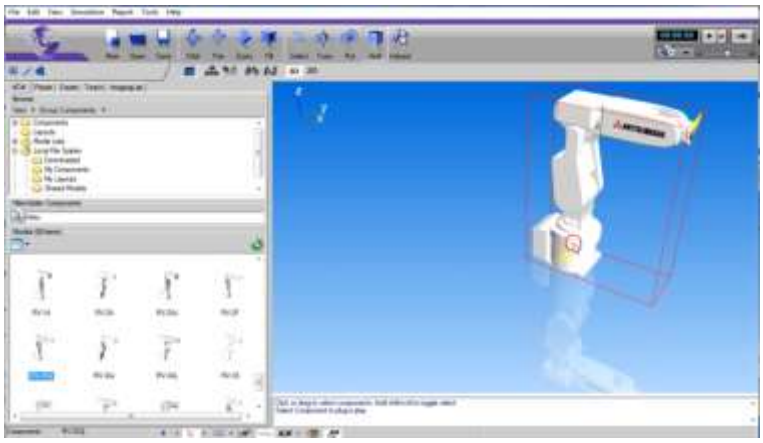


Рис. 7.18. Вікно для визначення за допомогою бібліотеки обраного робота Mitsubishi Robotics

За допомогою бібліотек обираються також додаткові пристрої, наприклад, захват (рис. 7.19).

Рис. 7.19. Додавання захвату до робота

Після цього можна здійснити налагодження обраних компонент.

Програма дозволяє зробити симуляцію роботи робота у ручному режимі (рис. 7.20).

Рис. 7.20. Симуляція роботи робота у ручному режимі

Для обраної конфігурації робота можна здійснити створення програми у середі LabVIEW (рис. 7.21).

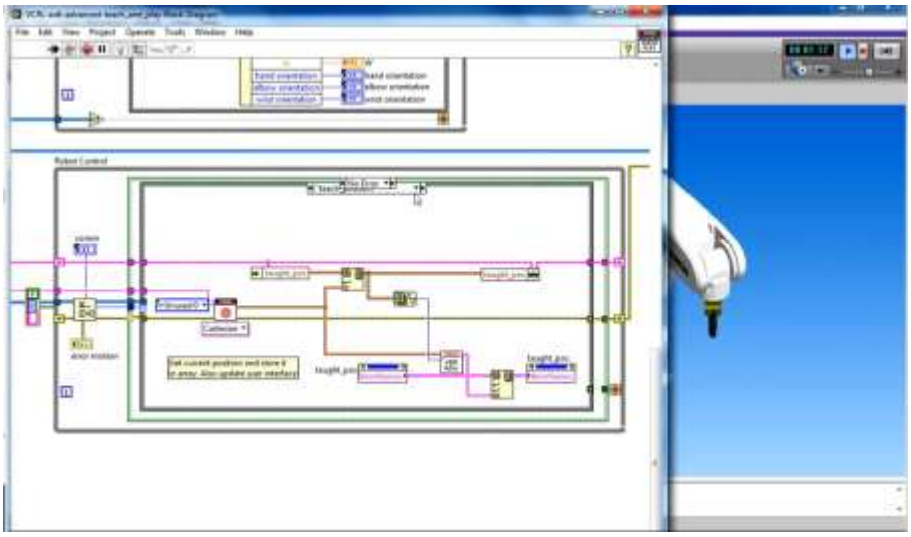


Рис. 7.21. Створення програми у середі LabVIEW

Програмування контролерів Arduino також можна здійснити за допомогою середи LabVIEW, використовуючи відповідну бібліотеку.

На рис. 7.22 наведений приклад програми для контролера Arduino, яка здійснює зчитування трьох аналогових входів та створює графіки зміни напруги у часі.

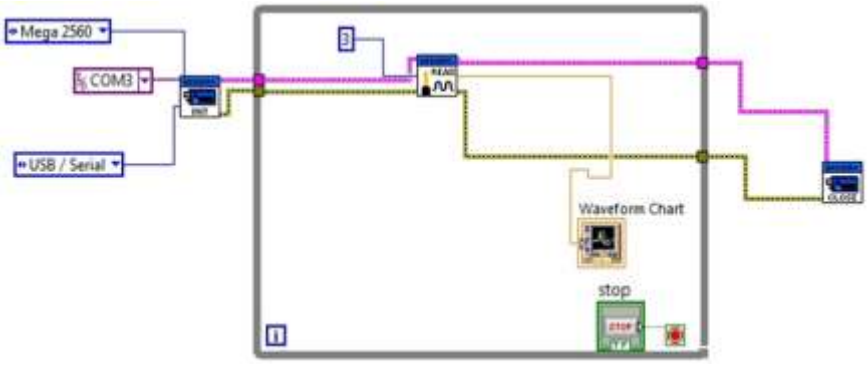


Рис. 7.22. Приклад програми для контролера Arduino

Розглянемо, як можна створити програму переміщення транспортного засобу вздовж контрастного контуру. В LabVIEW кольоровий датчик можна переключити на вимірювання освітленості. У цьому випадку вихідним параметром буде інтенсивність освітлення і для керування можна використати пропорційне регулювання.

Визначимо значення рівнів для білого та чорного кольорів (наприклад, 50 та 30). Значення порогу, що визначає кордон між білим і чорним, приймемо рівним середньому арифметичному від цих значень, тобто 40. Позначимо потужність двигуна А $\mathbf{M_A}$, а потужність двигуна С $\mathbf{M_C}$. Поточне значення світлового датчика позначимо **L**. Керування будемо здійснювати за формулою:

$$\mathbf{M_A = 30 - 2*(40 - L);}$$
$$\mathbf{M_C = 30 + 2*(40 - L).}$$

Якщо значення інтенсивності освітленості дорівнює 40, то множники в дужках рівні, а двигуни А і С здійснюють рух в одному напрямку з однаковою швидкістю (потужність 30). Якщо інтенсивність освітленості менше 40, то відповідно зменшується швидкість (потужність) двигуна А і збільшується швидкість двигуна С, а якщо інтенсивність освітленості більше 40, то навпаки. Таким чином радіус повороту буде залежати від ступеня відхилення від середнього значення освітленості і переміщення буде більш плавним.

На рис. 7.23 наведена програма, яка здійснює алгоритм керування за вказаною формулою.

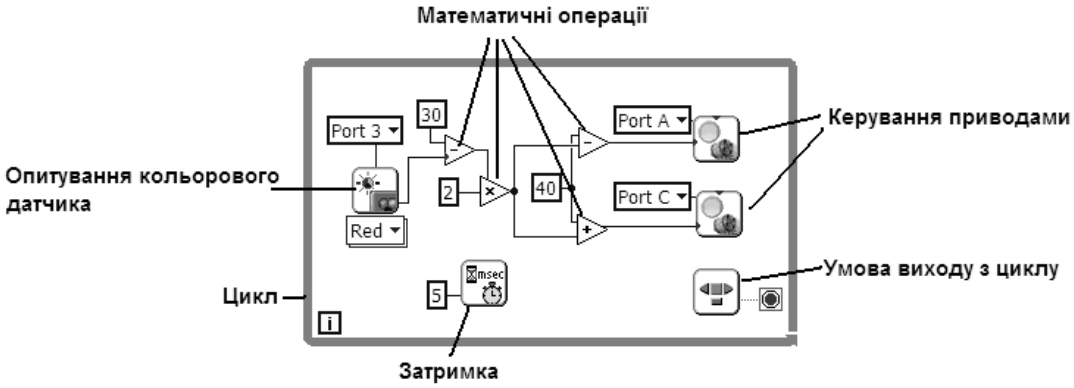


Рис. 7.23. Програма, яка здійснює алгоритм пропорційного керування при переміщенні по контуру

Розглянемо, як можна здійснити керування переміщенням транспортного робота з вилочним навантажувачем, що здійснює автоматичне визначення відстані до вантажу (рис. 7.24).

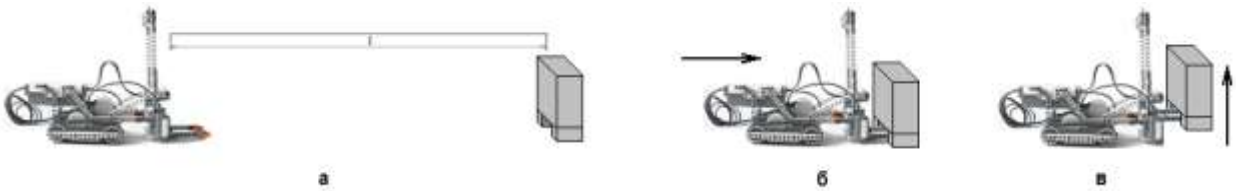


Рис. 7.24. Переміщення транспортного робота з автоматичним визначенням відстані до вантажу

На рис. 7.25 наведена програма, де транспортний робот визначає відстань до вантажу, переміщується до нього та здійснює підйом вантажу.

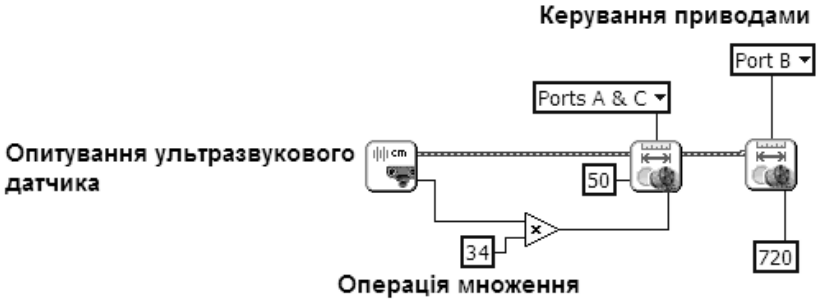


Рис. 7.25. Програма переміщення транспортного робота з визначенням відстані до вантажу

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 окремих компонент програмного забезпечення марсохода 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?

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

8.1. Структура та склад програмного засобу MICROSOFT ROBOTICS DEVELOPER STUDIO

Корпорація Microsoft випустила програмний засіб, призначений для конструювання, створення програмного забезпечення та моделювання роботів і різних роботизованих механізмів - Microsoft Robotics Developer Studio.

Цей програмний продукт поширюється в трьох варіантах: Standart Edition для професійних розробників, Academic Edition для навчальних закладів та Express Edition для індивідуальних розробників.

Версію Express Edition можна безкоштовно завантажити з сайту Microsoft.

Microsoft Robotics Developer Studio (**MRDS)** - це система, спеціально створена для розробки програмного забезпечення для роботів, причому, в основному, для конструювання роботів (набір модулів, які можна перепрограмувати в залежності від завдання, що треба вирішити).

MRDS складається з декількох компонентів.

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

2. Decentralized Software Services (DSS) - середовище, яке дозволяє запускати алгоритми обробки даних на різних обчислювальних пристроях, організовувати асинхронну взаємодію 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 програмування, призначена для розробки програм управління роботами. Програма на основі такої мови 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 асинхронності в будь-яких додатках.

При взаємодії з навколишнім середовищем, робот повинен правильно реагувати на інформацію, що надходить одночасно від великої кількості датчиків. Тому було прийнято рішення істотну частину логічного керування перенести на один або декілька 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 середовище, що базується на CCR, для створення розподілених додатків на основі сервісів, яке передбачає управління великою кількістю сервісів для коригування поведінки в цілому.

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

Visual Simulation Environment - це середовище симуляції. Так як не завжди можна використовувати справжні роботи, воно 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.

MRDS 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.

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 MRDS - **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.

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

MRDS 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

Мова VPL забезпечує можливість програмування роботів візуальними методами. Програма на VPL представляє собою діаграму, де кожен блок має свою функціональність і всі вони пов'язані між собою. Діаграми VPL кодуються за допомогою XML-схем.

Блоки можуть являти собою базові операції, а також різноманітні сервіси, такі як, датчики, веб-камера, сервоприводи, динаміки, світлодіоди, різні індикатори, дисплеї і тому подібні пристрої. Крім того, блоками в Robotics Studio можуть виступати спеціальні діалогові вікна, наприклад, для ручного дистанційного керування роботом.

Зв'язки між блоками є алгоритмічними конструкціями, іноді зі складною структурою. Вони можуть включати різноманітні розгалуження, умови, навіть цикли, а також роботу з даними, роботу зі змінними і т.д.

Базові блоки вказані в списку вікна «Basic Activities». Там є, наприклад, такі блоки (рис. 8.1).

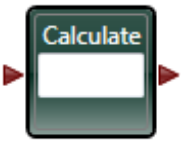
Рис. 8.1. Базові блоки в списку вікна «Basic Activities»

Далі наведені функції базових блоків.

Activity - блок, в який можна помістити різні операції, і послуги. Використовується, коли необхідно виконати послідовність дій кілька разів, або, якщо доцільно об'єднати безліч блоків в один (наприклад, для економії місця на діаграмі).

Variable - блок, в якому зберігається змінна.

Calculate - блок для обчислення математичних операцій.

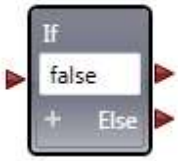


Data - блок використовується для передачі значення даних іншого блоку.

Join - блок об'єднує два або більше потоків даних (входів). Для виконання цього блоку, всі дані (входи) повинні бути отримані перед переходом на наступний крок.

Merge - блок з'єднує два або більше потоків даних (входів). Коли перший елемент даних отримано, здійснюється перехід на наступний крок. Блок не очікує отримання інших даних.

If - блок умови. Блок подібний висловом умови (if) в мовах програмування



Switch - блок, подібний висловом «switch» в C #. Порівнює вхідну значення зі значеннями в полях блоку і виконує відповідну дію, якщо значення співпало

Comment - блок коментарів.

На рис. 8.2 наведене головне вікно Microsoft Robotics Developer Studio.

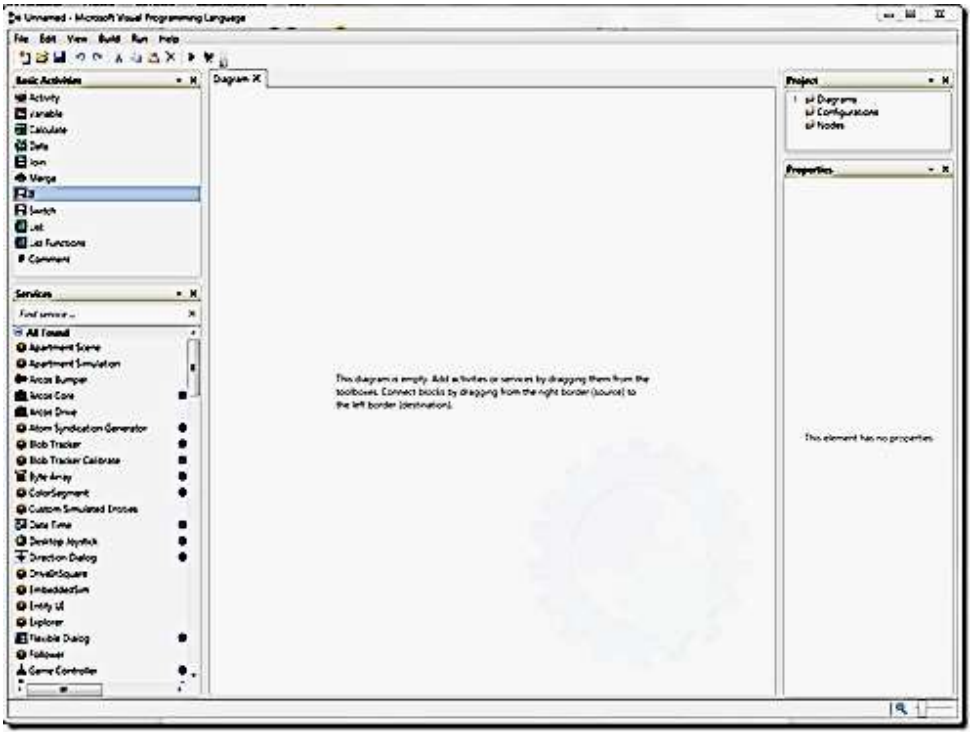


Рис. 8.2. Головне вікно Microsoft Robotics Developer Studio

Зліва в нас «Базові активності» і «Сервіси», праворуч структура проекту і «Властивості», а в центрі робоча область, в якій ми можемо будувати програму за допомогою візуальних блоків

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.

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 тощо. 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 у мовах програмування C або Pascal.

Програми можуть здійснювати введення даних та опитування датчиків, математичну обробку даних, керування виконавчими пристроями тощо.

На рис. 8.3 наведений приклад програми у вигляді діаграм, яка здійснює визначення натиснутої кнопки.

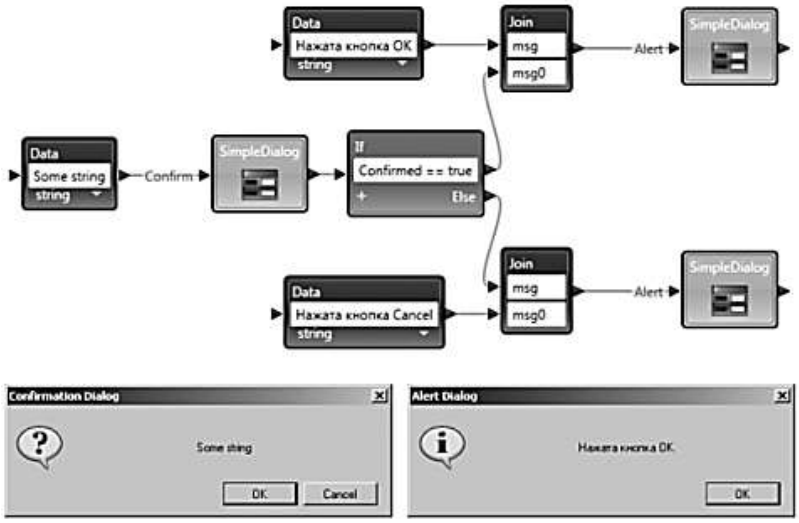


Рис. 8.3. Діаграма визначення натиснутої кнопки

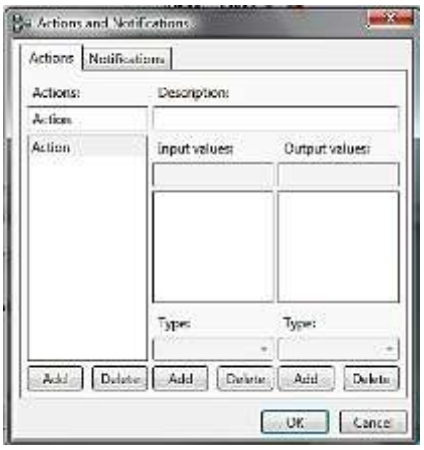
Для прикладу розглянемо, як виглядає програма для пересування робота трикутною траєкторією.Для того щоб робот рухався трикутником, необхідно виконати кілька разів рух вперед і поворот.І рух, і поворот виконуються блоком "GenericDifferentialDrive".

Після цього блоку ставиться блок WaitForDriveCompletion для очікування завершення команди руху або повороту.

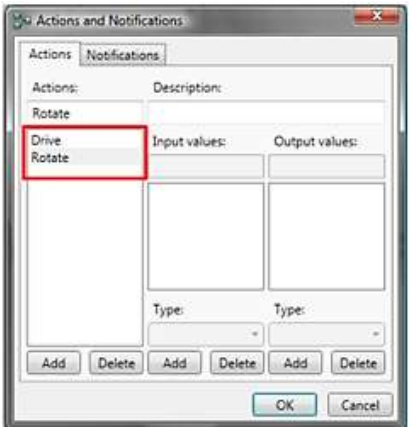
Створення програми

1. Помістимо на діаграму блок «Activity», який надалі міститиме блоки руху, повороту та очікування.
2. У властивостях блоку «Activity» змінимо його назву (Name та Friendly Name) на DriveOperations.
3. Подвійним клацанням миші на блоці відкриємо його. З'явиться діаграма, у якому можна розмістити блоки, необхідних виконання будь-якої операції.

4. Блок «DriveOperations» відповідає за рух та поворот.
Для цього визначимо ці дії для цього блоку:
При натисканні на кнопку "Actions and Notifications" з'явиться вікно

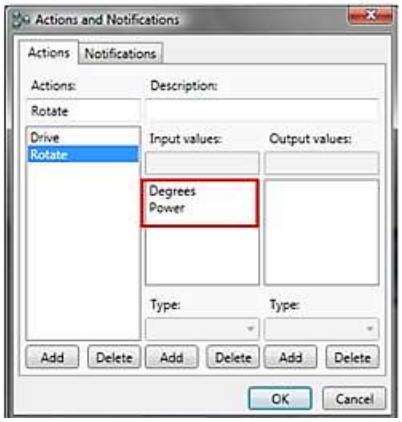


5. Додаємо дві дії Drive (рух) та Rotate (поворот):

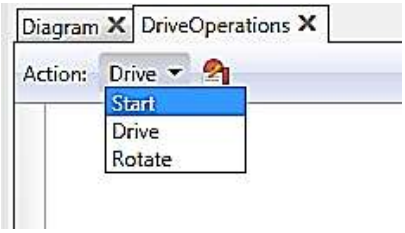


6. Визначимо вхідні значення: для дії Drive – Distance (дистанція) – тип double, Power (потужність) – тип double,

для дії Rotate – Degrees (градуси) – тип double, Power (потужність) – тип double



7. Тепер дії можна побачити у списку Action:



Далі визначимо блоки, які будуть використовуватись для кожної дії Action.

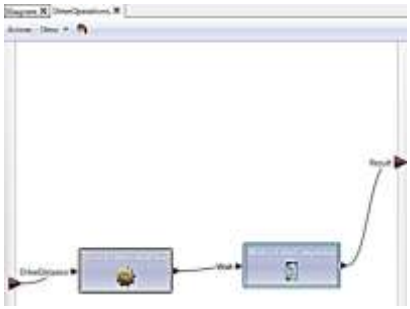
8. У списку виберемо дію Drive. І помістимо на діаграму блок "GenericDifferentialDrive". У його властивостях виберемо маніфест IRobot.Create.Simulation.Manifest.xml (кнопка Import).

9. З'єднаємо вхід блоку "Activity" з входом "GenericDifferentialDrive". У вікні Connections виберіть DriveDistance. У вікні Data Connections вкажемо значення для Distance – Distance, Power – Power, для DriveDistanceStage – DriveStage.InitialRequest.

10. Помістимо на діаграму блок "WaitForDriveCompletion", з'єднаємо вихід блоку

"GenericDifferentialDrive" із входом даного блоку. У вікні Connections вкажемо DriveDistance – Success.

11. З'єднаємо вихід блоку WaitForDriveCompletion з виходом блоку Activity. У вікні Connections вкажемо Wait – Success. В результаті отримали діаграму:



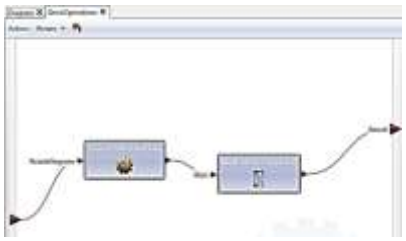
12. У списку виберемо дію Rotate. І помістимо на діаграму блок "GenericDifferentialDrive". У його властивостях виберемо маніфест IRobot.Create.Simulation.Manifest.xml (кнопка Import).

13. З'єднаємо вхід блоку «Activity» із входом «GenericDifferentialDrive».

У вікні Connections виберіть RotateDegrees. У вікні Data Connections вкажемо значення для Degrees – Degrees, Power – Power, RotateDegreesStage – DriveStage.InitialRequest.

14. Помістимо на діаграму блок "WaitForDriveCompletion", з'єднаємо вихід блоку "GenericDifferentialDrive" із входом даного блоку. У вікні Connections вкажемо RotateDegrees – Success.

15. З'єднаємо вихід блоку WaitForDriveCompletion з виходом блоку Activity. У вікні Connections вкажемо Wait – Success. В результаті отримали діаграму:



16. Перейдемо на основну діаграму. Помістимо на діаграму блок Data (вкажемо тип string та визначимо повідомлення), блок SimpleDialog для виведення діалогового вікна з повідомленням.

17. З'єднаємо вихід блоку SimpleDialog із входом блоку DriveOperations. У вікні Connections оберемо From: AlertDialog – Success, To: Drive. У вікні DataConections визначимо значення для дистанції (Distance) та потужності (Power).

Для редагування значень необхідно встановити галочку «Edit values directly».

Data Connections:

Value	Target
0.5	Distance
0.5	Power

18. Скопіюємо та вставимо ще один блок DriveOperations. З'єднаємо перший і другий блоки Driveoperations. У вікні Connections оберемо From: Drive – Result, To: Rotate. У вікні DataConections визначимо значення градусів повороту (Degrees) та потужності (Power).

Data Connections:

Value	Target
120	Degrees
0.5	Power

19. Додаємо ще три блоки DriveOperations. І з'єднуємо їх відповідними зв'язками для руху траєкторією трикутника.

В результаті отримаємо діаграму, наведену на рис. 8.4.

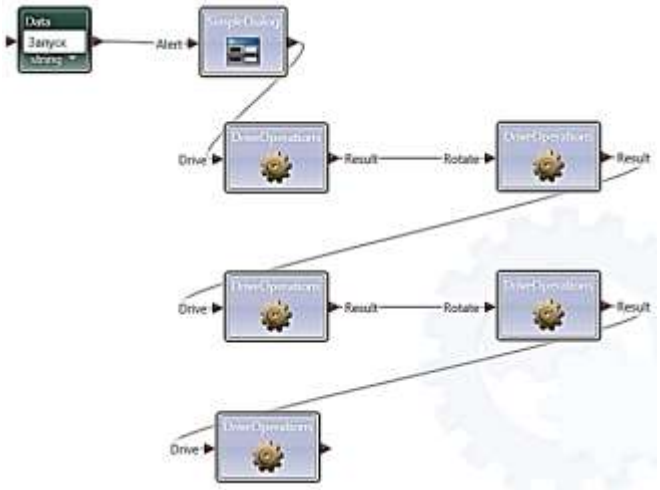


Рис. 8.4. Результат створення діаграми

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. 8.5..

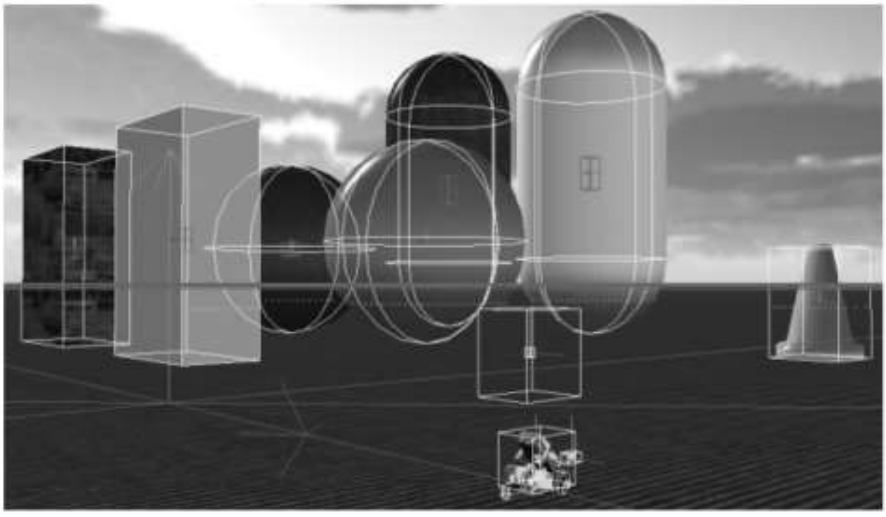


Рис. 8.5. Режим Combined

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.

Симулятор має різні сервіси для керування роботом, наприклад, сервіс DesktopJoystick (рис. 8.6), який можна використовувати для віртуальго керування рухом робота.

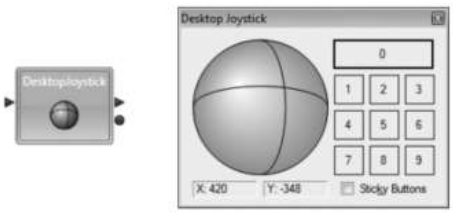


Рис. 8.6. Сервіс DesktopJoystick

На рис. 8.7 наведений приклад вікна симулятора з керуванням роботом iRobot за допомогою сервісу DesktopJoystick.

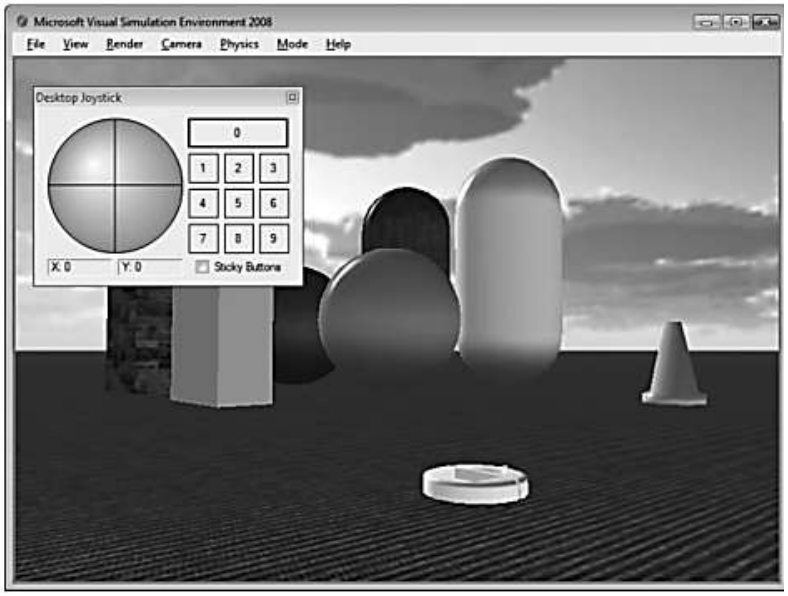


Рис. 8.7. Приклад вікна симулятора з керуванням роботом iRobot за допомогою сервісу DesktopJoystick

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. 8.8.

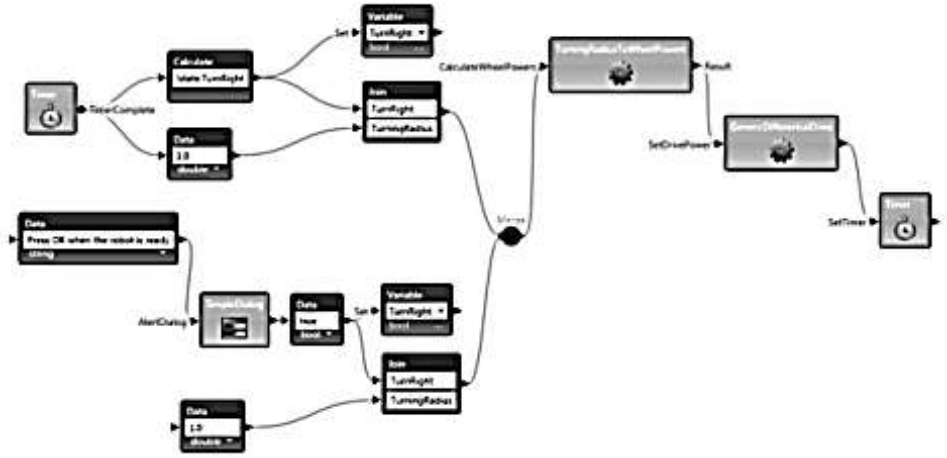


Рис. 8.8. Діаграма для організації руху робота по вісімці

Розглянемо як здійснюється запуск програми на виконання.

Спочатку треба вибрати маніфест для елемента «GenericContactSensors».

Маніфести - це своєрідні драйвери низького рівня, або, з іншого боку - конфігурації тієї чи іншої програми VPL.

Ідея полягає в тому, що одна і та ж програма, в принципі, може виконуватися на різних роботах, незважаючи на те, що на низькому рівні управління реалізується неоднаково. Завдяки маніфестам, щоб перенести програму VPL з одного пристрою керування робота на інший – треба вибрати маніфести відповідного робота для основних модулів діаграми. Аналогічно за допомогою спеціального маніфесту «Simulation» реалізується перенесення програми в середу симуляції.

У контекстному меню «GenericContactSensors» вибираємо «Set Configuration» (рис. 8.9.

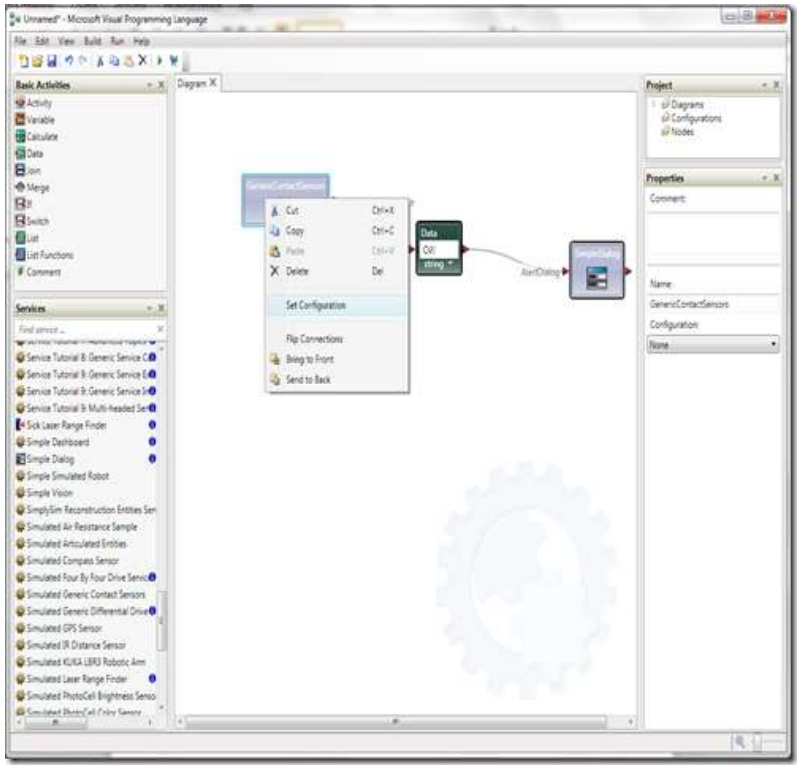


Рис. 8.9. Розділ «Set Configuration»

Після цього, в розділі «Set Configuration» треба вибрати «Use a manifest».

В результаті в нижній частині з'явиться список, що випадає «Use existing or create a new manifest» і кнопка «Import manifest». За допомогою кнопки імпортуємо маніфест:

«IRobot.Create.Simulation.Manifest»

(можна вибрати маніфест іншого робота, але в даному прикладі буде протестований саме IRobot.Create).

Оскільки, справжній робот відсутній, використовується віртуальний.

Тому треба вибрати

«IRobot.Create.Simulation.Manifest».

В результаті все має виглядати, як показано на рис. 8.10.

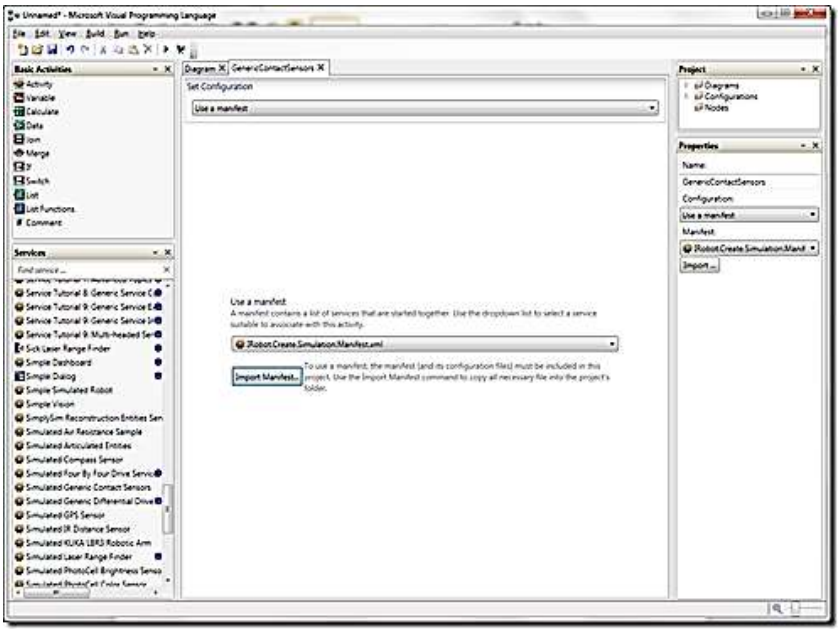


Рис. 8.10. Результат

Так як у віртуальній імітованого середовищі «доторкнуться» до сенсора робота рукою не вийде, ми додаймо в робочу область елемент «Simple Dashboard» зі списку «Services», за допомогою якого ми зможемо «їздити» по віртуальному середовищі. І «доторкнувшись» сенсором до якого-небудь об'єкта, побачити результат роботи нашої програми (рис. 8.11).

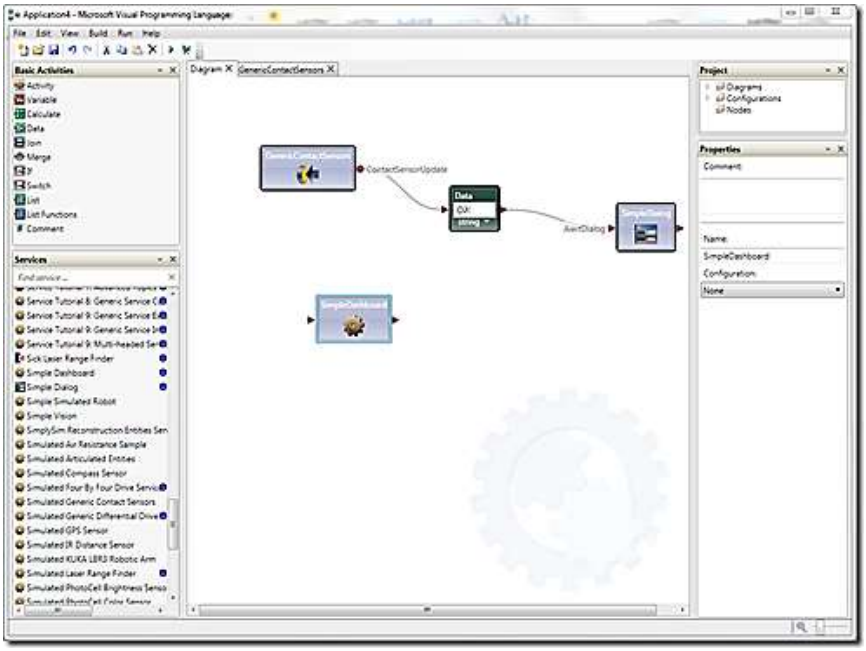


Рис. 8.11. Програма опитування сенсора

Для запуска треба натиснути F5 або обрати в меню «Run» - пункт «Start». В результаті бачимо вікно Run, в якому відображаються повідомлення про стан виконання (Стартував чи сервіс, завантажився чи маніфест і ін.), яке наведено на рис. 8.12.

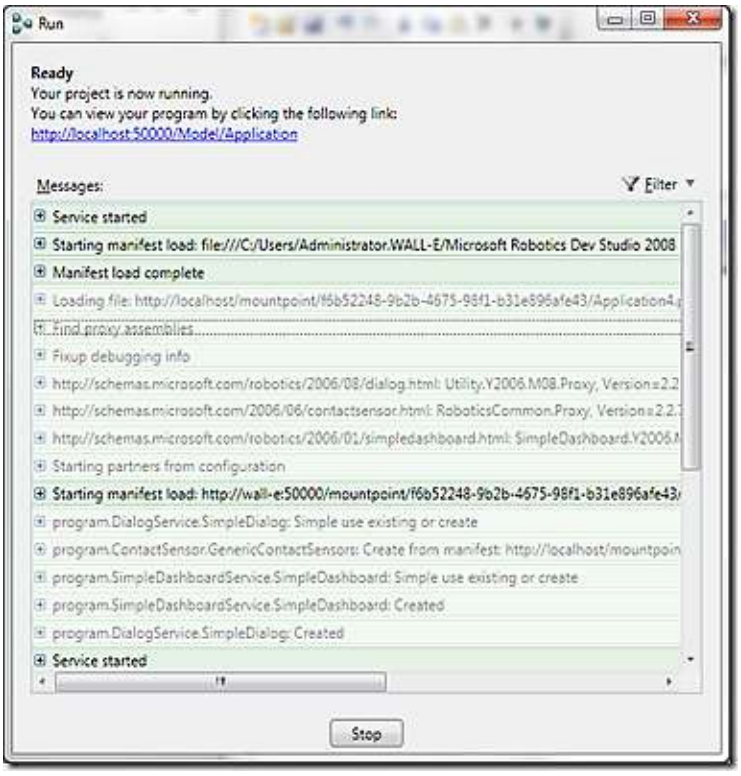
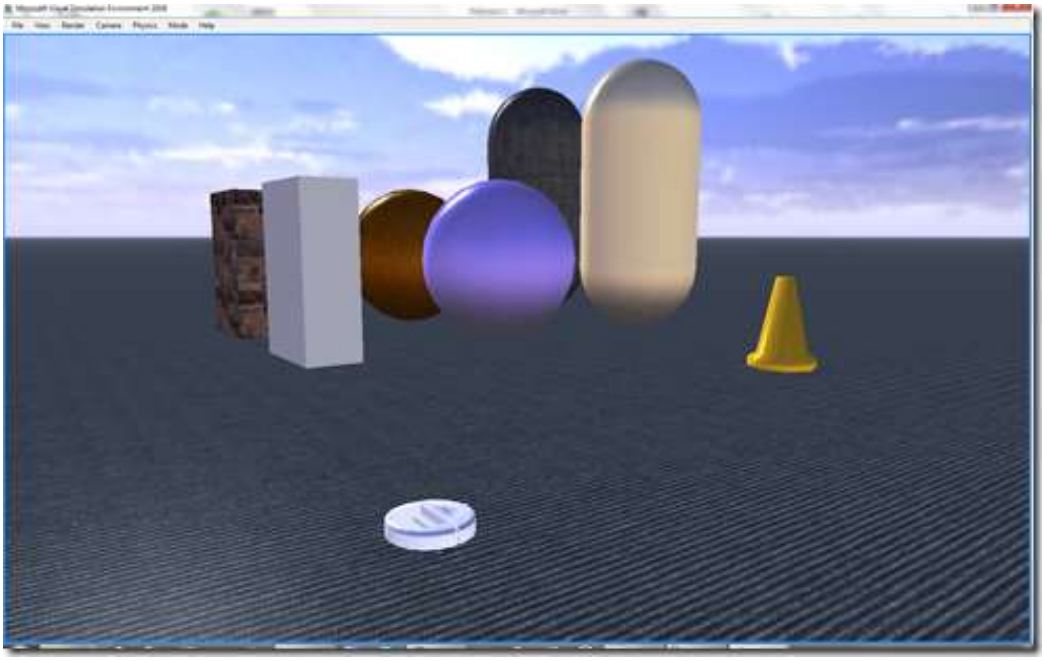


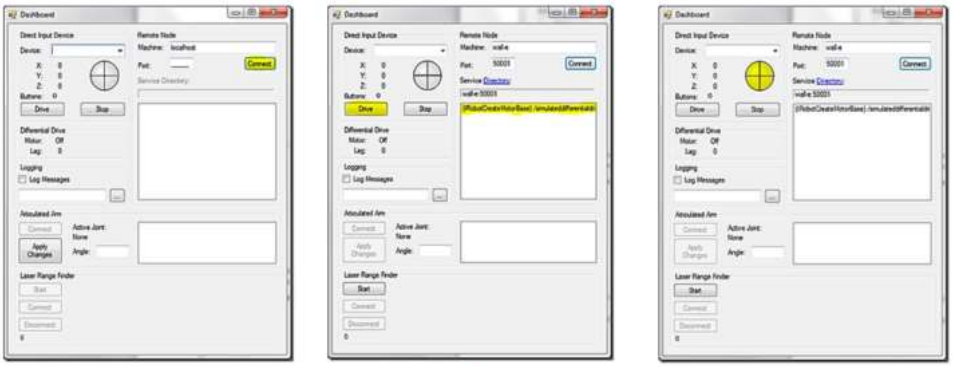
Рис. 8.12. Повідомлення про стан виконання

Після завантаження маніфесту і запуску всіх необхідних сервісів з'являється вікно, в якому ми бачимо нашу віртуальне середовище і нашого робота (рис. 8.13).



Ри. 8.13. Робота і віртуальне середовище

Також з'являється вікно «Dashboard» (рис. 8.14) за допомогою якого можна управляти роботом.



а) б) в)

Рис. 8.14. Вікно «Dashboard»

Для початку підключаємося до робота - натискаємо «Connect» у вікні (а).

Після чого, робимо подвійний клік по «IRobot Create Motor Base» і клік по кнопці «Drive» (б).

Для керування роботом використовується «трекбол» (в).

Кликнувши по ньому мишею ми можемо управляти рухом робота.

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. 8.15).

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. 8.15. Робот-пилосос Roomba

На базі платформи робота-пилососа Roomba компанією iRobot розроблений робот iRobot Create, призначений для розробників роботів, дозволяє програмувати поведінку робота (рис. 8.16).

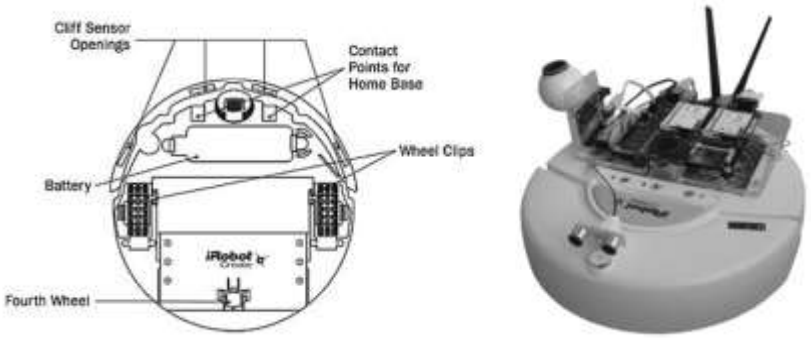


Рис. 8.16. Робот iRobot Create

MICROSOFT ROBOTICS DEVELOPER STUDIO підтримує також таких виробників роботів, як CoroWare CoroBot, iRobot Create, KUKA Robotics, Robosoft's robots, Aldebaran Robotics Nao та інших (рис. 8.17).

Рис. 8.17. Роботи, які підтримує MICROSOFT ROBOTICS DEVELOPER STUDIO

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?

9. Засоби конструювання роботів ABB

9.1. Засоби проектування роботів ABB

RobotStudio це комп'ютерна програма для моделювання в автономному режимі програмування і моделювання роботів.

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.

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.

Для правильно написаної off-line програми 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, оскільки моделювання необхідного замовником процесу може скласти конкурентну перевагу компанії;
- процес покупки і доставки робота і обладнання може затягнутися на кілька місяців, а у цей час інженери можуть промоделювати роботу РТК в off-line режимі;
- потрібно перевірити конструкційну досяжність інструменту, оснастки тощо, оскільки найчастіше геометрія спроектованого інструменту не підходить для роботи через виникаючі колізії, а перероблення конструкції затратно і може зайняти багато часу;
- необхідно впровадити робот на працюючій конвеєрної лінії, а 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, а 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. 9.1).

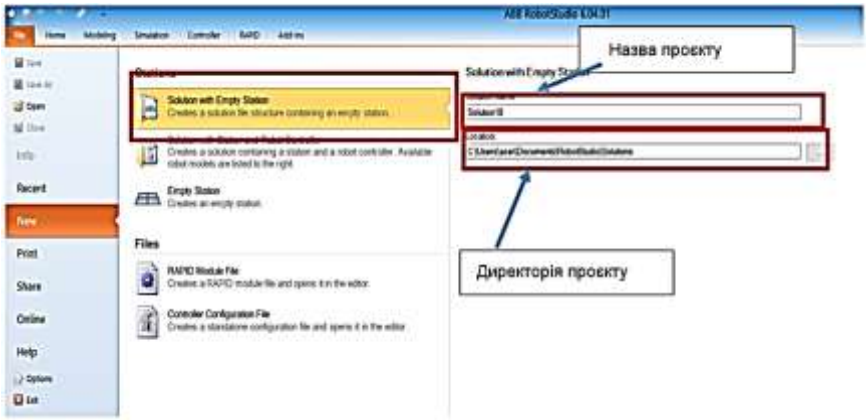


Рис. 9.1. Стартове меню

Після створення проекту відкриється порожня робоча область (рис. 9.2).

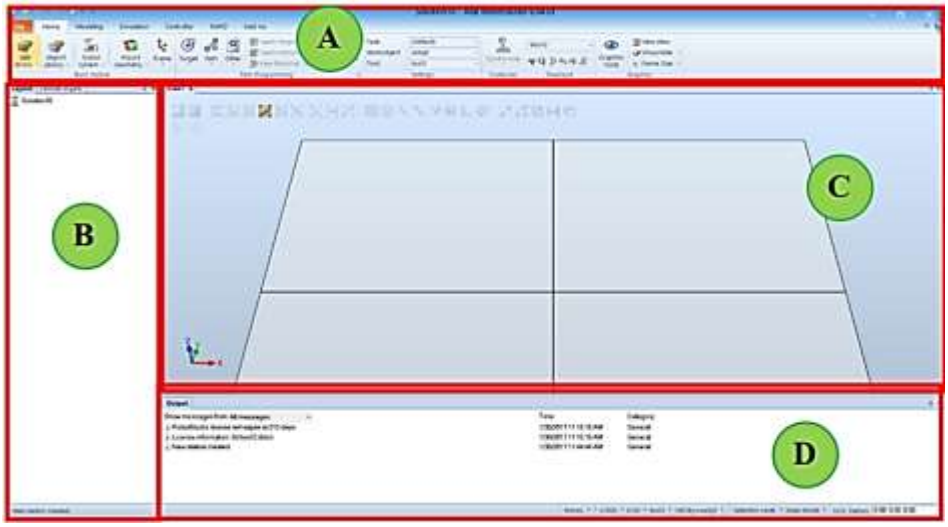


Рис. 9.2. Робоча область

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

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 RobotStudio)

Щоб додати робота в «Меню інструментів (A)» треба обрати «ABB Library» і вибрати робота зі списку представлених роботів (рис. 9.3).

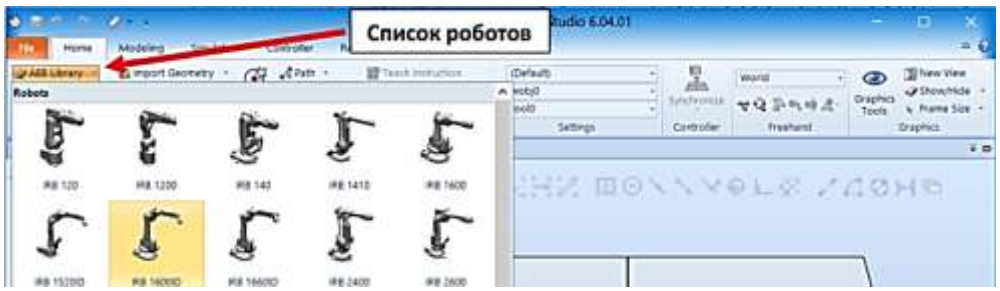


Рис. 9.3. Список роботів

Після вибору потрібного робота він додається на графічний екран симуляції (C) (рис. 9.4).

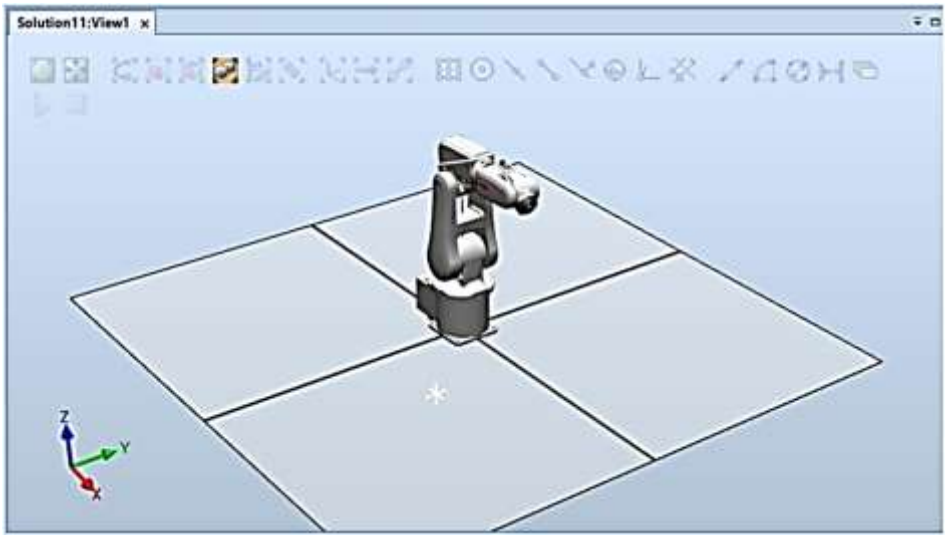


Рис 9.4. Графічний екран симуляції

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. 9.5).

Рис. 9.5. Вибір системи

Після запуску системи можна здійснити 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. 9.6).

На рис. 9.6 наведені такі переміщення елементів 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.

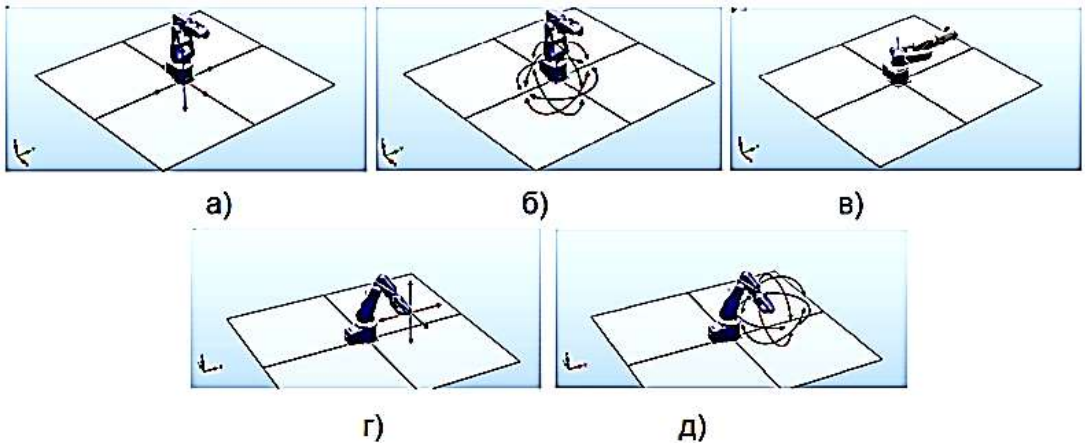


Рис. 9.6. Зміна становища робота

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. 9.7).

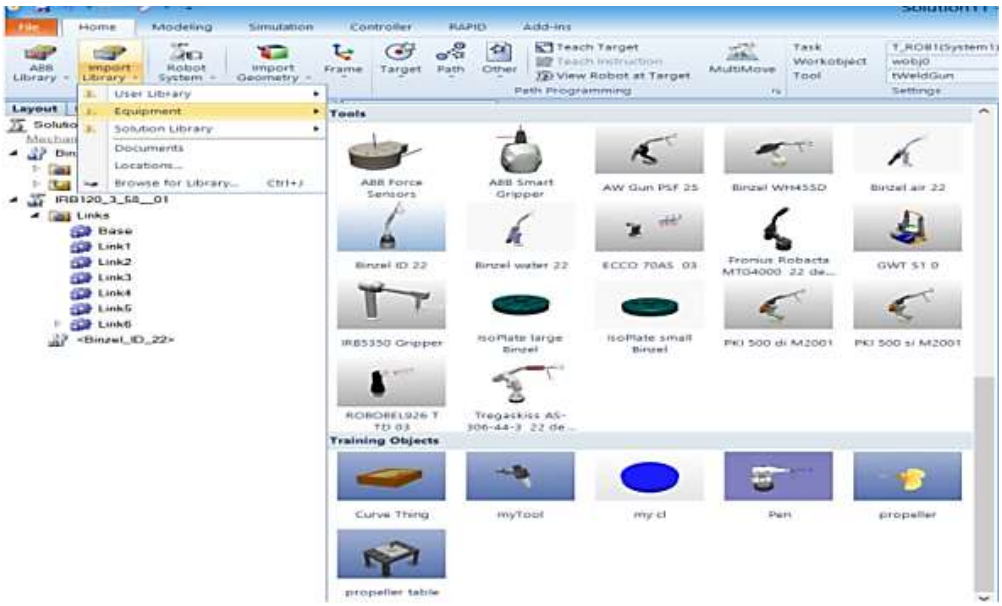


Рис. 9.7. Бібліотека інструментів

Після вибору інструменту він з'явиться в робочій області (рис.9.8, а).

Щоб додати інструмент до робота потрібно перетягнути об'єкт інструменту в «Layout» на робота (рис.9.8, б).

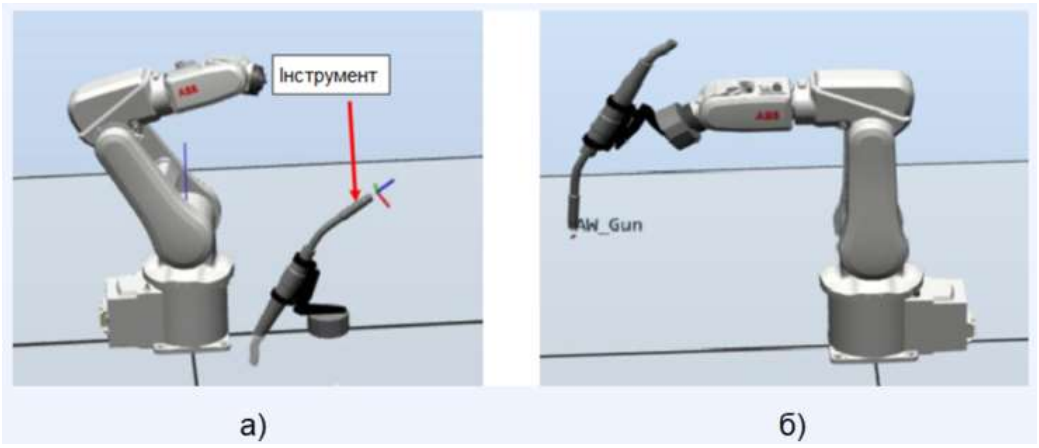


Рис. 9.8. Встановлення інструменту

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. 9.9).

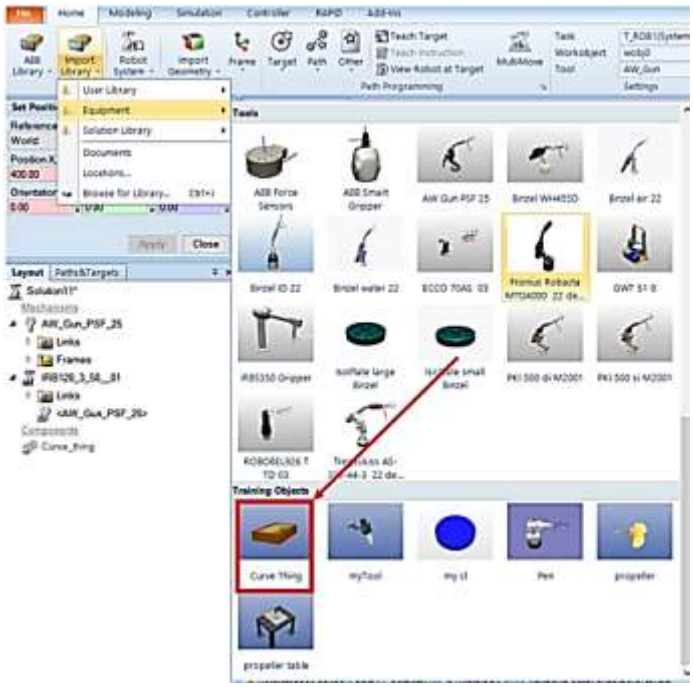


Рис. 9.9. Додавання деталі

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 змінятся 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. 9.10.

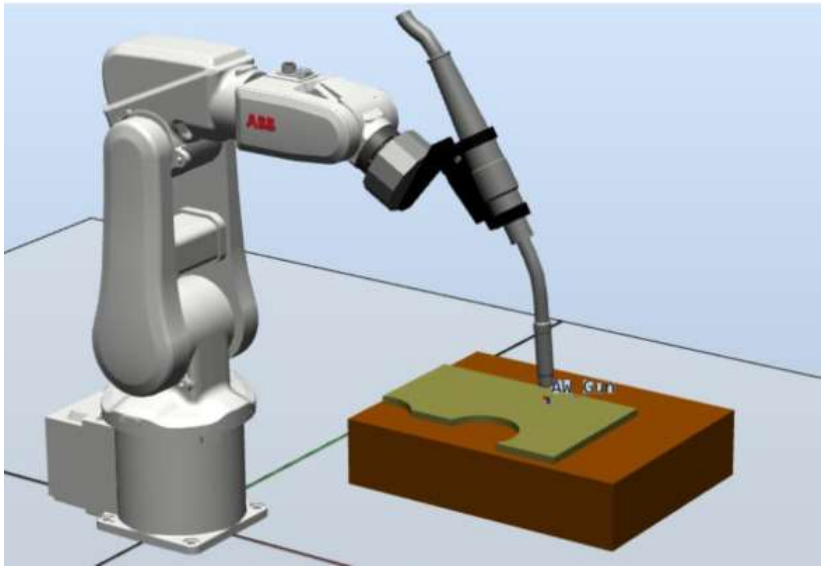


Рис. 9.10. Встановлення платформи з деталлю

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 можна 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 межу.

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, чи дістає 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.

На рис. 9.11 наведений приклад створення робототехнічної системи.

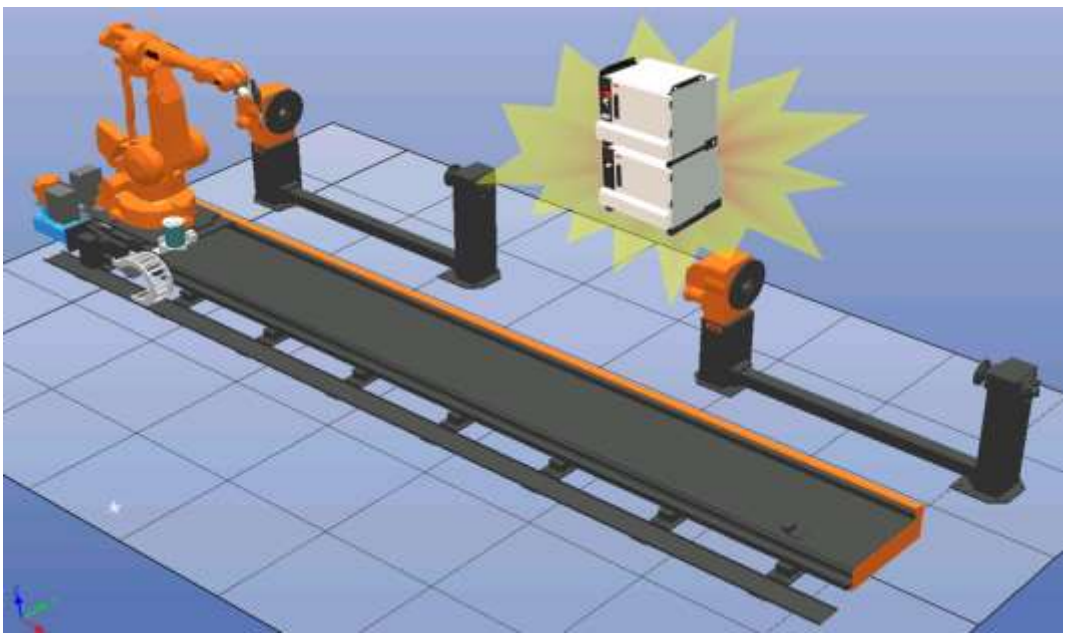


Рис. 9.11. Приклад створення робототехнічної системи

На рис. 9.12 наведений приклад конструювання та моделювання елементів робота.

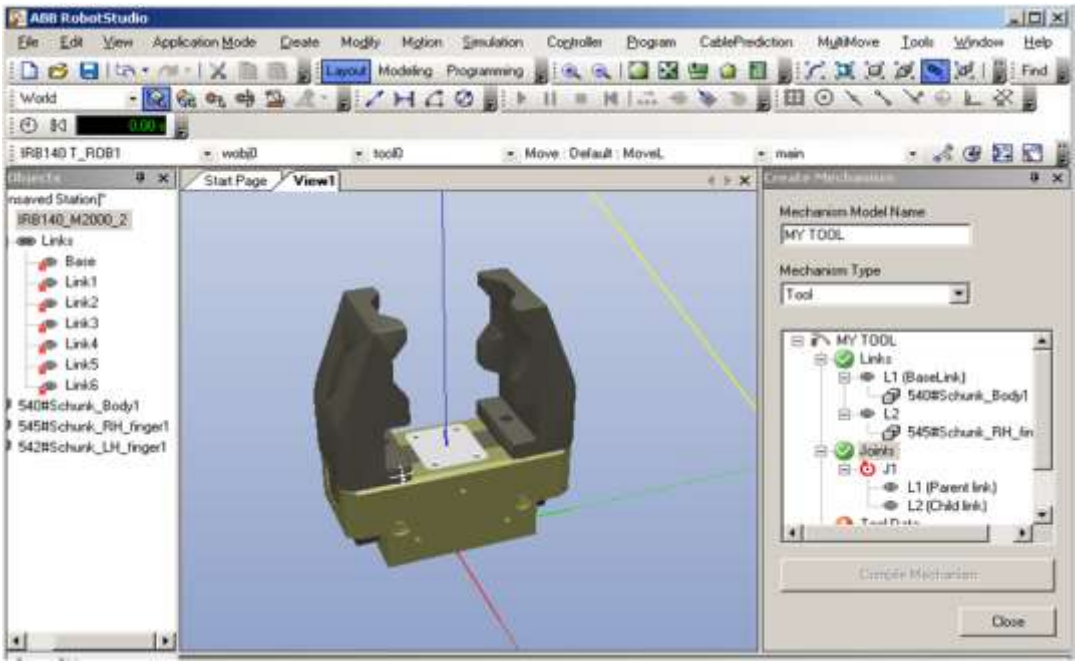


Рис. 9.12. Приклад конструювання та моделювання захоплюючого пристрою

RobotStudio дає можливість розробити та налагодити програму керування роботом (рис. 9.13). Для цього є такі компоненти, як **Program Editor** (Редактор програм), **Debugging** (засоби налагодження), **Video Recording & Screen Shot** (функція створення запису відео та знімку екрану).

а) б) в)

Рис. 9.13. Засоби для розробки та налагодження програми керування роботом: Program Editor (а), Debugging (б), Video Recording & Screen Shot (в)

9.3. Конструювання моделі роботизованої ланки завантаження

RobotStudio є симуляційною середою програмування роботів компанії ABB. Вона дозволяє реалізувати off-line програмування роботів за допомогою технології VRT (Virtual Robot Technology) без зупинки виробництва. Ця технологія використовує так звані «віртуальні контролери» (VirtualController), які представляють собою точну копію реального програмного забезпечення робота, що дозволяє створювати дуже реалістичний режим симуляції.

Розглянемо наступні аспекти роботи з RobotStudio:

- створення шаблонної системи;
- управління роботом в ручному режимі;
- створення траєкторій;
- управління вхідними / вихідними сигналами контролера;
- редагування керуючої програми в Rapid Editor;
- завантаження геометрії і бібліотек;
- створення інструменту для маніпулятора.

На рис. 9.14 наведений результат проєктування станції навантаження: маніпулятор переміщує вантаж з конвеєра на палету.

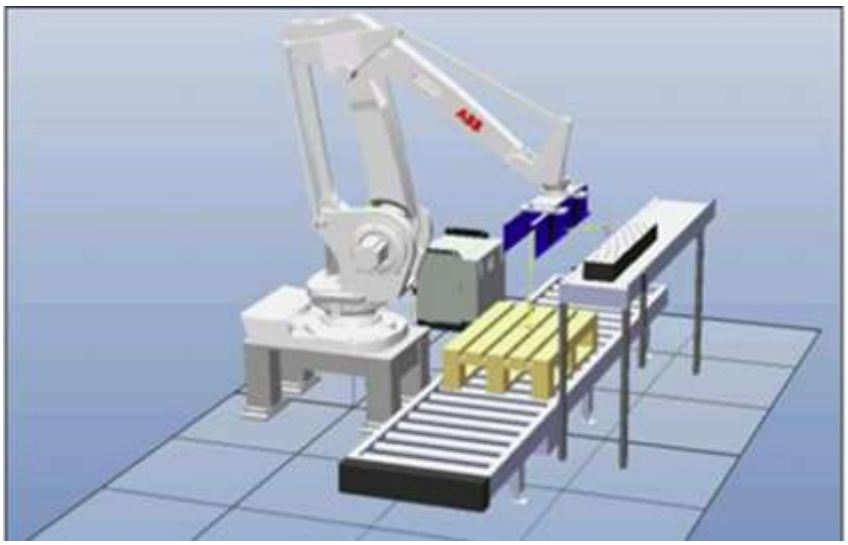


Рис. 9.14. Станція навантаження

При цьому захоплюючий пристрій закривається, коли потрібно взяти вантаж і відкривається, коли потрібно його відпустити.

Станція представляє собою модель 3-вимірного простору з усіма об'єктами, які розташовані в цьому просторі.

Система RobotWare - ряд програмних файлів, які при завантаженні у віртуальний контролер роблять доступними всі функції, конфігурації, дані і програми, що керують системою робота.

Віртуальний контролер - програмний засіб, що здійснює емуляцію контролера, щоб дозволити програмі, яка управляє роботом, працювати на персональному комп'ютері.

Для початку проєктування запускаємо RobotStudio, після чого з'являється повідомлення про оновлення графічного драйвера.

Натискаємо «No».

Після цього відкривається головне вікно програми (рис. 9.15).

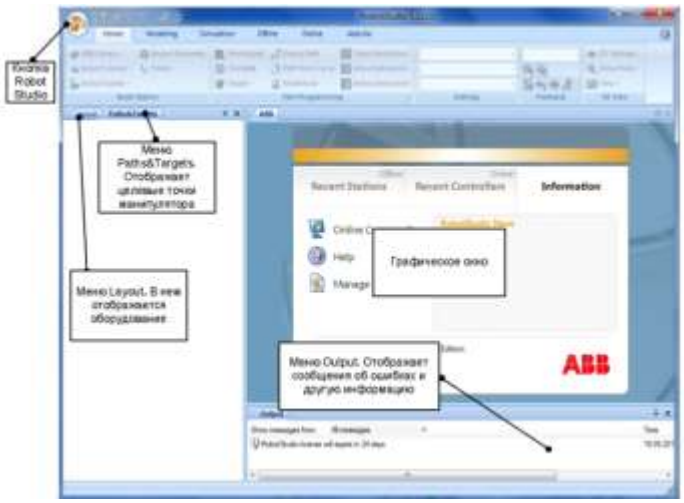


Рис. 9.15. Головне вікно програми

Спочатку треба створити шаблонну систему. Для цього натискаємо кнопку «RobotStudio». Вибираємо пункт New Station. З'являється вікно, наведене на рис. 9.16.

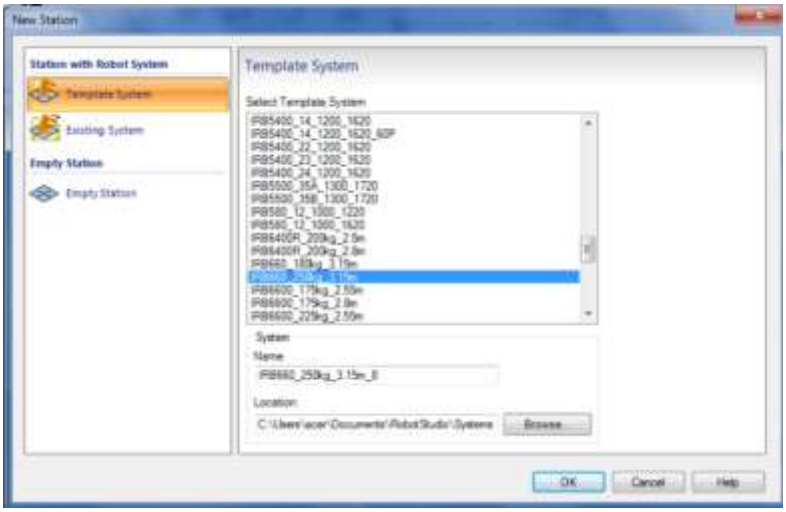


Рис. 9.16. Вікно «New Station»

У лівій частині повинен бути обраний пункт Template System, праворуч вибираємо систему: IRB660_250kg_3.15m.

Натискаємо «OK».

На протязі деякого часу здійснюється завантаження віртуального контролеру.

У момент закінчення завантаження індикатор стану віртуального контролера в правому нижньому кутку вікна RobotStudio стає зеленим, а в вікні Output відображається такий напис:

IRB660_250kg_3.15m_8: Motors ON state 17.01.2011 19:19:04 Controllers

Навігація в графічному меню здійснюється за допомогою миші

Табл. 9.1.

Навігація в графічному меню

Дія	Комбінація клавиш	Опис
Вибір об'єкту		Клацніть на об'єкті, щоб вибрати його. Натисніть SHIFT, щоб вибрати кілька об'єктів.
Обертання виду	CTRL + SHIFT +	Натиснути CTRL + SHIFT + ліву кнопку миші, поки рухаєте мишу для обертання виду. Якщо є колесо прокручування, можна замість комбінації клавіш натиснути ліву кнопку миші та колесо прокручування.
Плоске переміщення	CTRL +	Натиснути CTRL + ліву кнопку миші, поки рухаєте мишу для переміщення виду.
Масштабування	CTRL +	Натисніть Ctrl + права кнопка миші, поки рухаєте мишу для масштабування виду. Якщо є колесо прокручування, можна масштабувати вигляд, обертаючи його або натиснувши колесо прокручування і рухаючи мишею.

Розглянемо, як здійснюється управління роботом в ручному режимі, а саме, управління рухом маніпулятора по ланках (Joint).

Для цього на вкладці Home в групі команд Freehand натискаємо кнопку «Jog Joint»

Після цього клацаємо в графічному вікні по ланці, яку будемо переміщувати. Ланка виділяється червоним кольором (рис. 9.17).

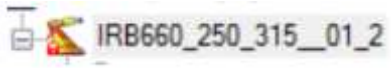
Рис. 9.17. Виділення ланки

Після цього рухаючи мишею, як показано у табл. 9.1, можна здійснити керування переміщенням ланки.

Для здійснення лінійного переміщення системи координат використовується інструмент (Tool).

На вкладці Home в групі команд Freehand натискаємо кнопку «Jog Linear»

В меню Layout виділяємо вузол маніпулятора



У графічному вікні з'являються стрілки, рухаючи якими за допомогою мишею ми переміщаємо маніпулятор (рис. 9.18).

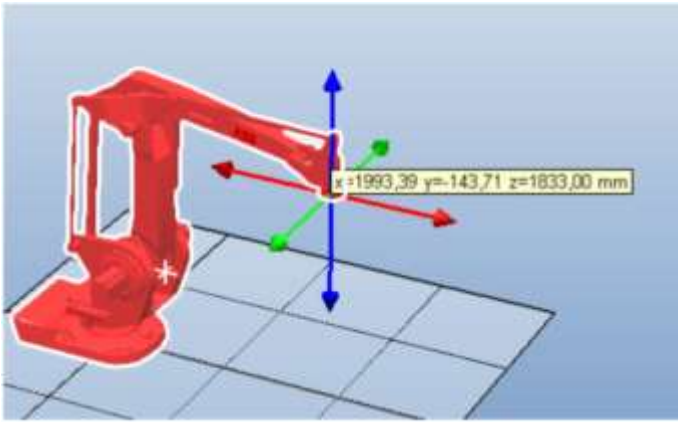
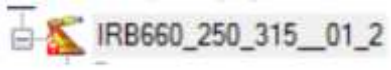


Рис. 9.18. Лінійний рух захоплюючого пристрою / інструменту

Обертання навколо центру системи координат інструмента.

На вкладці Home в групі команд Freehand натискаємо кнопку «Jog Reorient»

В меню Layout виділяємо вузол маніпулятора



У графічному вікні з'являються кругові стрілки, рухаючи якими за допомогою мишею ми переміщаємо маніпулятор (рис. 9.19).

Однак для цього маніпулятора можливо обертання тільки навколо осі z системи координат інструмента.

При цьому рухається фланець маніпулятора.

Обертання навколо інших осей можливо для 6-ступеневих маніпуляторів

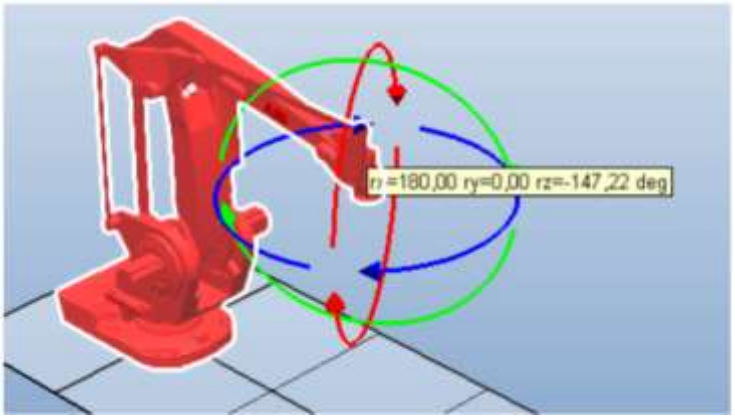


Рис. 9.19. Обертання навколо центру системи координат інструмента

Тепер можна повернути маніпулятор в початкове положення.

Для цього виділяємо в меню Layout вузол маніпулятора, а в контекстному меню натискаємо «Jump Home».

Завантаження бібліотек і геометрії

Об'єкти, які ви імпортуєте в станцію, можуть бути або геометрією, або бібліотекою.

Геометрія при імпортуванні копіюється в станцію RobotStudio.

Коли ви імпортуєте бібліотеку, створюється зв'язок між бібліотечним файлом і станцією.

До того ж, бібліотечний файл може містити RobotStudio-спеціалізовану інформацію.

Наприклад, якщо інструмент збережений як бібліотека, то **tooldata** зберігається разом з геометричною інформацією.

Набір стандартних бібліотек доступний відразу при установці RobotStudio. Крім того, користувач може створювати свої бібліотеки.

Завантажимо бібліотеки конвеєрів.

Для цього натискаємо верхню частину кнопки «Import Library»

З'являється віконце завантаження бібліотеки.

У ньому вибираємо бібліотеки \ conv_box_top .

Натискаємо «Відкрити», після чого здійснюється завантаження конвеєра. Після чого конвеєр з'являється у графічному вікні (рис. 9.20).

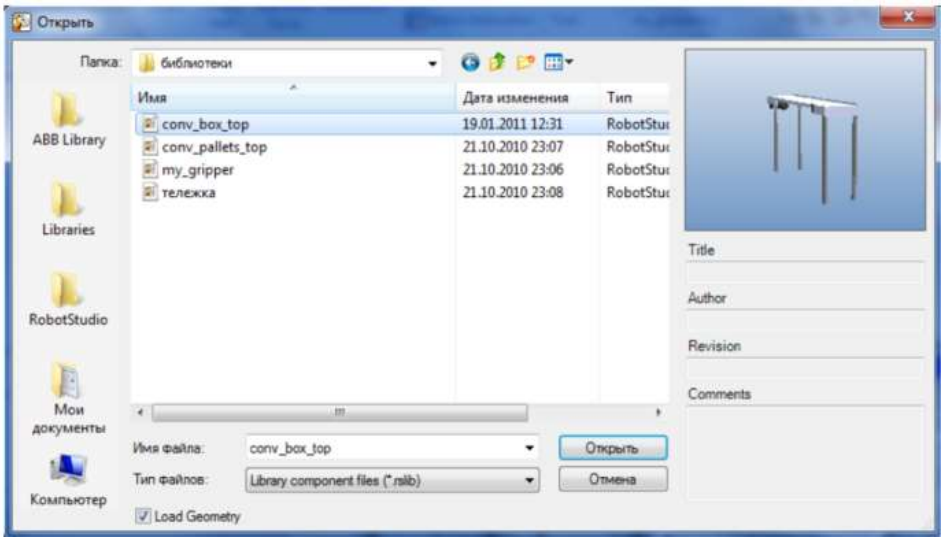


Рис. 9.20. Завантаження конвеєра

Аналогічно завантажуємо «conv_pallets_top».

Завантажимо стандартний бібліотечний файл контролера.

Для цього натискаємо нижню частину кнопки «Import Library» та вибираємо «IRC Control-Module», після чого здійснюється завантаження контролера

Розмістимо контролер у просторі станції. Для обертання натискаємо кнопку «Rotate» групи команд «Freehand» на вкладці Home

Для лінійного переміщення використовуємо кнопку "Move" групи команд «Freehand» на вкладці «Home»

Тепер завантажимо кілька файлів геометрії. Натискаємо кнопку "Import Geometry"

Відкриваємо \геометрія\Стіл (це п'єдестал для робота). Потрібно розташувати його у просторі. Вибираємо в меню Layout вузол «Стол» (Стіл) та в контекстному меню клацаємо «Set Position». У віконці вводимо наступні дані

Натискаємо "Apply". Стіл рухається. Натискаємо "Close". Тепер здійснимо переміщення маніпулятора та його системи координат. Для цього вибираємо у верхній частині вікна програми вкладку Offline та натискаємо кнопку «System Configuration»

У вікні, що з'явилося, зліва виділяємо вузол «ROB_1», а справа вводимо дані, як показано на рис. 9.21.

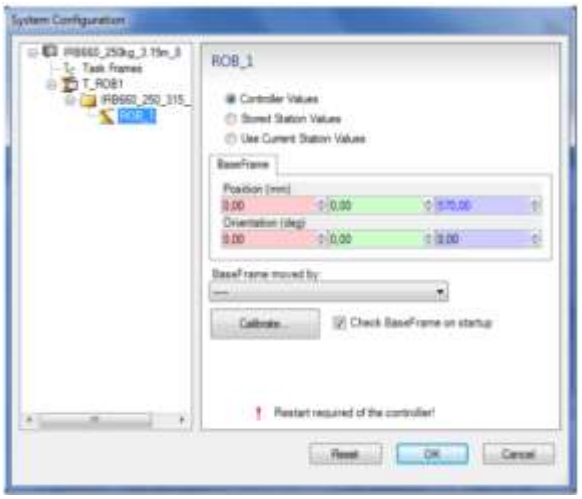
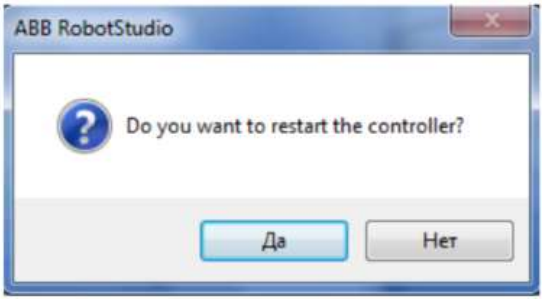


Рис. 9.21. Дані для зсуву системи координат маніпулятора

Натискаємо "ОК". З'являється віконце



Натискаємо "Yes". Після цього здійснюється переміщення маніпулятора

У графічній частині вікна можна побачити, що маніпулятор перемістився на п'єдестал. Створимо об'єкт «вантаж».

На вкладці «Modeling» у меню «Solid» вибираємо об'єкт «Box». Габаритні розміри:

Length: 210

Width: 1000

Height: 200

Вибираємо вузол «Груз» (Вантаж) у меню Layout, у контекстному меню вибираємо Set Position.

Вводимо дані у вікні «Set Position», як показано на рису. 9.22.

Рис. 9.22. Дані у вікні «Set Position» для переміщення вузла «Груз»

Натискаємо «Apply» та «Close». У графічній частині вікна можна побачити, що вантаж перемістився на конвеєр.

Завантажуємо графічний файл «pallet». Розміщуємо палету в просторі за допомогою опції «Set Position».

Вводимо дані у вікні Set Position, як показано на рис. 9.23.

Рис. 9.23. Дані у вікні «Set Position» для переміщення палети

Розглянемо створення траєкторій руху.

Створення системи координат WorkObject

Спочатку створимо спеціальні системи координат щодо яких будуть задані цільові точки.

У RobotStudio така система координат називається WorkObject».

Для цього на вкладці Home натискаємо кнопку «WorkObject»

У віконці «**Create Workobject**» в полі «**Name**» вводимо «**Wobj_box**»

Повернемо станцію у графічному вікні таким чином, щоб її конвеєр був орієнтований так, як показано на рис. 9.24.

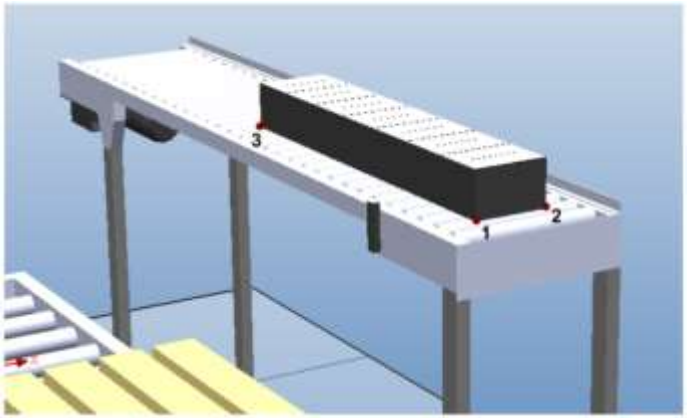


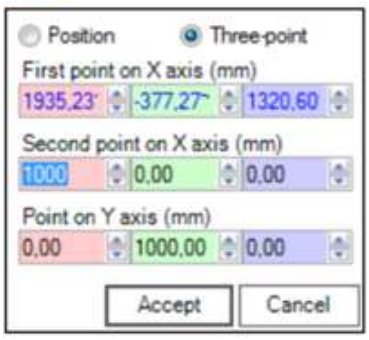
Рис. 9.24. Орієнтація конвеєру

Включаємо прив'язку до кінцевих точок.

Для цього натискаємо кнопку Snap End у верхній частині графічного вікна.

Далі у розділі User Frame у вікні Create Workobject вибираємо пункт Frame by points та натискаємо на стрілку праворуч від нього.

З'явиться віконце введення координат для Workobject.



Вибираємо опцію Three-point. Потім клацаємо у полі введення координат першої точки, за допомогою прив'язки наводимо курсор на точку 1 (рисунок на слайді 53) і клацаємо по ній.

У полях, що відповідають першій точці, з'являються значення координат. Аналогічно вводимо координати для другої та третьої точок.

Натискаємо "Assept".

У вікні Create WorkObject натискаємо Create.

Аналогічно створюємо WorkObject під назвою Wobj_pallet на основі точок, показаних на рис. 9.25.

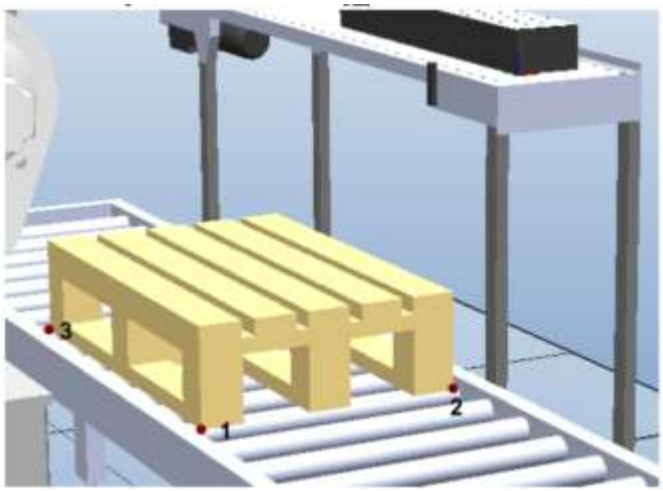


Рис. 9.25. Вигляд станції для завдання Workobject

Створення цільових точок траєкторії

Після створення станції визначаються цільові точки траєкторії шляхом запровадження їх координат або навчанням.

Для цього потрібно натиснути кнопку «Target» вкладки Home та вибрати пункт Create Target

З'являється вікно введення даних.

Заповнюємо графи Position та Orientation, вибираємо WorkObject Wobj_box.

Натискаємо кнопку "Add", а потім "Create«

Аналогічно створюються інші точки траєкторії.

Створення траєкторії

Натискуємо кнопку «Empty Path».

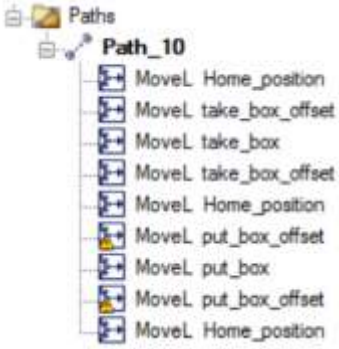
У меню Paths&Targets у розділі Path з'являється порожня траєкторія Path_10. Додамо до траєкторії Path_10 цільові точки. Для цього виділяємо в меню Paths&Targets цільову точкуHome_position.

У контекстному меню натискаємо Add to path -> Path_10 -> <First>

Після цього точка додається в траєкторію.

Аналогічно додаються до траєкторії інші точки послідовності.

Отримана траєкторія має вигляд



Для моделювання руху вибираємо вкладку Simulation, натискаємо кнопку Simulation Setup

З'являється вікно Simulation Setup (рис.9.26).

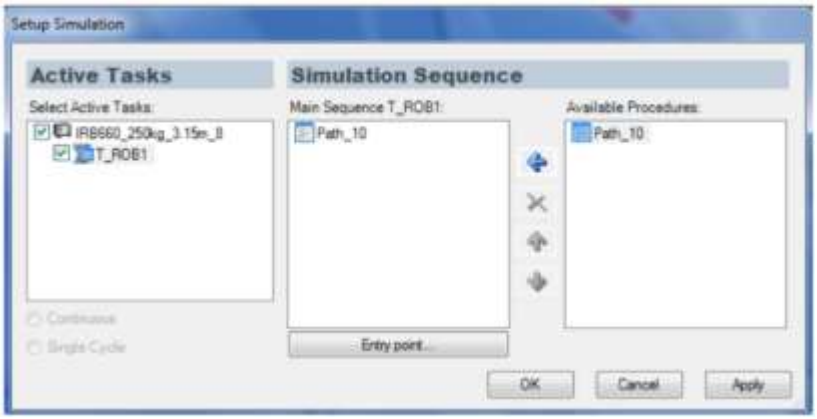


Рис. 9.26. Вікно Simulation Setup

У списку Available Procedures виділяємо траєкторію Path_10 та натискаємо на стрілку.

Після цього тиснемо "ОК".

Тепер програму, що управляє рухом маніпулятора вздовж заданої траєкторії, створено.

Для моделвання руху у графічному вікні натиснемо кнопку "Play"

Після цього побачимо переміщення маніпулятора вздовж траєкторії.

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?

10. Віртуальна робототехнічна експериментальна платформа V-REP/ CoppeliaSim

10.1. Структура та ключові особливості V-REP/ CoppeliaSim.

Ціль моделювання при створенні нових робототехнічних рішень, залежить головним чином від стадії розробки.

Це може бути і перевірка гіпотези, і оптимізація конструкції, і тестування програмного забезпечення, що реалізує нові алгоритми обробки сенсорної інформації та управління поведінкою, і на пізніших етапах – налагодження коду, що виконується до його запуску на контролері робочої станції маніпулятора.

Існує низка програмних рішень для робототехнічних систем, які дозволяють створювати симуляції з високою точністю. Серед них можна виділити V-REP / CoppeliaSim, яка має більший функціонал.

Ця платформи має широкі можливості з моделювання роботів різного типу, починаючи від плаваючих і закінчуючи літаючими.

Відмінною особливістю платформи є можливість легкого масштабування.

Програмний пакет V-REP/CoppeliaSim розповсюджується абсолютно безкоштовно та регулярно отримує нові оновлення.

Програмний пакет V-REP / CoppeliaSim завантажується зі сторінки:
https://coppeliarobotics.com/downloads

Платформа V-REP (Virtual Robotics Experimentation Platform) має ряд особливостей, які надають розробнику широкі можливості для створення симуляцій.

Як основний компонент V-REP можна виділити технологію вбудованих скриптів, що виконують функції контролерів у симуляціях.

Скрипт (англ. script – сценарій) – це невелика програма, яка містить послідовність дій, створених для автоматичного виконання завдання.

При цьому наявність можливості прив'язки окремих скриптів до компонентів робота дозволяє реалізувати чітку ієрархію, забезпечуючи портативність та масштабованість.

Будь-яка симуляція в V-REP за умовчанням має основний скрипт, який не рекомендується змінювати, оскільки цей скрипт вирішує загальні завдання забезпечення коректності даних при виконанні симуляції.

Наприклад, він викликає різні підсистеми для моделювання кінематики та динаміки механічних елементів системи.

З основного скрипта також виконується виклик дочірніх скриптів каскадним способом, але цю функцію можна відключити.

Дочірні скрипти (child script)**,** на відміну від основного (Main script), прикріплюються до окремих компонентів моделі.

Вони є невід'ємною частиною сценарію симуляції і виконуються за кожної ітерації моделювання, як і основний скрипт.

Слід також пам'ятати, що скрипти є файлами, що виконуються, і не вимагають попередньої компіляції.

Крім того, дочірні скрипти можуть бути потоковими (threaded child script) чи непотоковими (non threaded child script).

Розробники V-REP рекомендують по можливості використовувати непотокові скрипти, однак у деяких задачах потокові скрипти краще вирішують поставлені завдання.

Слід також звернути увагу, що симуляція – це ітеративний процес, тобто. перерахунок параметрів системи, що моделюється, здійснюється через постійний проміжок часу (крок моделювання) ітеративно. V-REP за замовчуванням використовує крок 50 мілісекунд.

Потокові скрипти – скрипти, виконання яких триватиме за наступної ітерації. Такі скрипти виконуються один раз від початку до кінця, але також можна вимкнути цю функцію, щоб не було переривання після першої ітерації.

Непотокові скрипти – скрипти, виконання яких починається з початку кожної наступної ітерації, у своїй здійснюється повна передача управління симуляцією в ці скрипти. Якщо з якихось причин скрипт не завершився і управління не передалося назад на основний скрипт, то симуляція переривається. Такі скрипти викликаються основним скриптом 2 рази на кожен крок симуляції: при активації та отриманні сенсорної інформації.

Таким чином, для кращого розуміння процесу виконання скрипта слід розглянути докладніше структуру потокового скрипта, що буде зроблено далі.

Доступність. Поширюється V-REP платформа умовно безкоштовно. Для наукових досліджень та ведення освітньої діяльності дана програма має окремі версії. У разі використання в комерційних проектах необхідно придбати ліцензію.

Програма V-REP є повністю кросплатформною (незалежною від операційної системи, що використовується).

V-REP підтримує модель взаємодії Клієнт-Сервер. Платформа V-REP виступає сервером, на якому запущено симуляцію, а управління здійснюється з клієнтської сторони.

При такій схемі взаємодії як клієнт може виступати як інше програмне забезпечення, так і будь-який пристрій з підключенням до сервера (V-REP), наприклад, клієнтом може бути робот.

API функції. Пропонується широкий набір готових API функцій на мові Lua, використання яких робить процес створення сценарію симуляції швидким та легким.

Широкі можливості також представляють функції API для створення сценаріїв симуляції іншими мовами програмування (Remote API), а саме:C/C++, Python, Java, Matlab, Octave та Lua.

У V-REP є чотири різних фізичних двигуна моделювання: ODE, Bullet, Vortex і Newton. Кожен із них має більш високу точність моделювання у певних задачах, але всі вони добре вирішують загальні завдання моделювання роботів.

Вбудовані модулі для спеціалізованих завдань. Є потужний та гнучкий модуль обчислення зворотної та прямої кінематики роботів. Цей пакет добре підходить для вузькоспеціалізованих завдань під час роботи з маніпуляторами. Вбудований модуль для швидкої перевірки зіткнень об'єктів дозволяє вирішувати низку завдань, пов'язаних із безпекою, максимально ефективно. Також є схожий модуль, який швидко і точно розраховує мінімальну відстань між об'єктами.

Є різні типи датчиків, зокрема найчастіше застосовувані датчики наближення (аналог ультразвукових датчиків). Також є фото-відео датчики, які аналізують видимі властивості різних компонентів симуляції.

Планування руху. У програмі V-REP планування руху виконується з використанням відкритої бібліотеки планування руху (Open Motion Planning Library), скорочено OMPL.

Запис та візуалізація даних. Можливості для візуалізації даних були одними з ключових для розробників, і починаючи з найпершої версії V-REP є весь необхідний інструментарій для запису і візуалізації будь-яких типів даних. Докладніше цю тему буде розкрито далі.

Вбудований редактор сітки. Є редактор сітки, який дозволяє вносити зміни до сітки моделі, на основі якої розраховуються фізичні параметри механічних елементів симуляції. Також підтримується декілька спеціальних режимів редагування сітки об'єкта.

Імпорт та експорт даних. Завантаження та вивантаження даних з V-REP виконується через меню: для цих завдань є спеціальний інструмент в основному меню. Також програма може зчитувати дані через з'єднання з іншим пристроєм.

Повнофункціональна ієрархія моделей у сцені. Цей інструмент дозволяє платформі V-REP реалізувати взаємозв'язки компонентів моделі вкрай зручно та з широкими можливостями масштабування.

Це далеко не все, чим відрізняється V-REP від інших програм моделювання, проте розуміння цих особливостей дозволить надалі працювати з платформою та доповнювати загальну інформацію більш детальними та спеціалізованими можливостями програми.

10.2. Основи роботи платформою V-REP / CoppeliaSim

Далі міститься інформація, головною метою якої є формування уявлення про базові елементи, з якими необхідно взаємодіяти під час створення симуляцій у програмі V-REP.

Інтерфейс програми.

Інтерфейс програми V-REP розділений на кілька частин залежно від призначення та реалізований у вікні з графічним інтерфейсом: графічне вікно програми використовується для керування всіма вбудованими інструментами.

Також слід згадати вікно консолі, яке можна спостерігати під час запуску програми, але за умовчанням це вікно відразу ж ховається.

За потреби можна налаштувати V-REP на постійне відображення консолі, викликавши «User Settings» («Налаштування користувача») за допомогою першої кнопки у вертикальній панелі інструментів.

У консольному вікні відображаються плагіни, що завантажуються, і їх процедури, які потрібні тільки при роботі з плагінами.

Плагін (англ. plug-in — підключати) — додаток, незалежно скомпільований програмний модуль, що динамічно підключається до основної програми, призначений для розширення або використання її можливостей. Належить до загального програмного класу додатків. Плагіни зазвичай виконуються у вигляді динамічних бібліотек.

Плагіни використовуються в V-REP як зручний інструмент налаштування симулятора. Вони можуть зареєструвати команди мови lua, дозволяючи швидке виконання зворотного виклику функції з вбудованого сценарію. Плагіни можуть розширити можливості конкретної імітаційної моделі або об'єкта. Часто вони також реалізують конкретні імпорти/експорти або забезпечують інтерфейс до конкретного обладнання. Віддалений API інтерфейс, і навіть інтерфейс ROS (див. слід. пункти) здійснюються через плагіни.

У цій лекції не розглядається робота з плагінами, проте за необхідності можна звернутися до документації по роботі з V-REP на сайті розробників.

На рис. 10.1 наведено інтерфейс програми V-REP з досить великим набором активних інструментів, проте треба мати на увазі, що одночасна активація всіх інструментів призведе до їх взаємного перекриття.

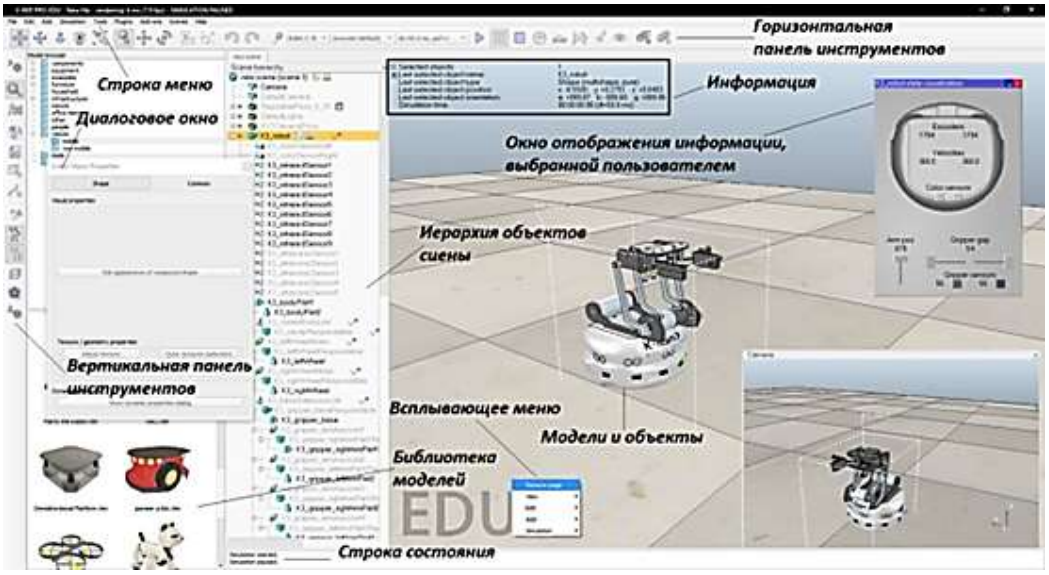


Рис. 10.1. Вікно додатку V-REP

Далі наведено основні складові графічного інтерфейсу програми V-REP з коротким описом.

Основне меню програми містить різні розділи, в яких є інструменти для редагування наявних та додавання нових об'єктів до сценарію симуляції.

Частина інструментів з головного меню також дубльована у спливаючому контекстному меню V-REP.

Панелі інструментів. Найчастіше використовувані інструменти винесені сюди для швидкого доступу.

На наступних рисунках наведені кнопки, розташовані на панелях інструментів з їх назвами.

Горизонтальна панель інструментів V-REP наведена на рис. 10.2.

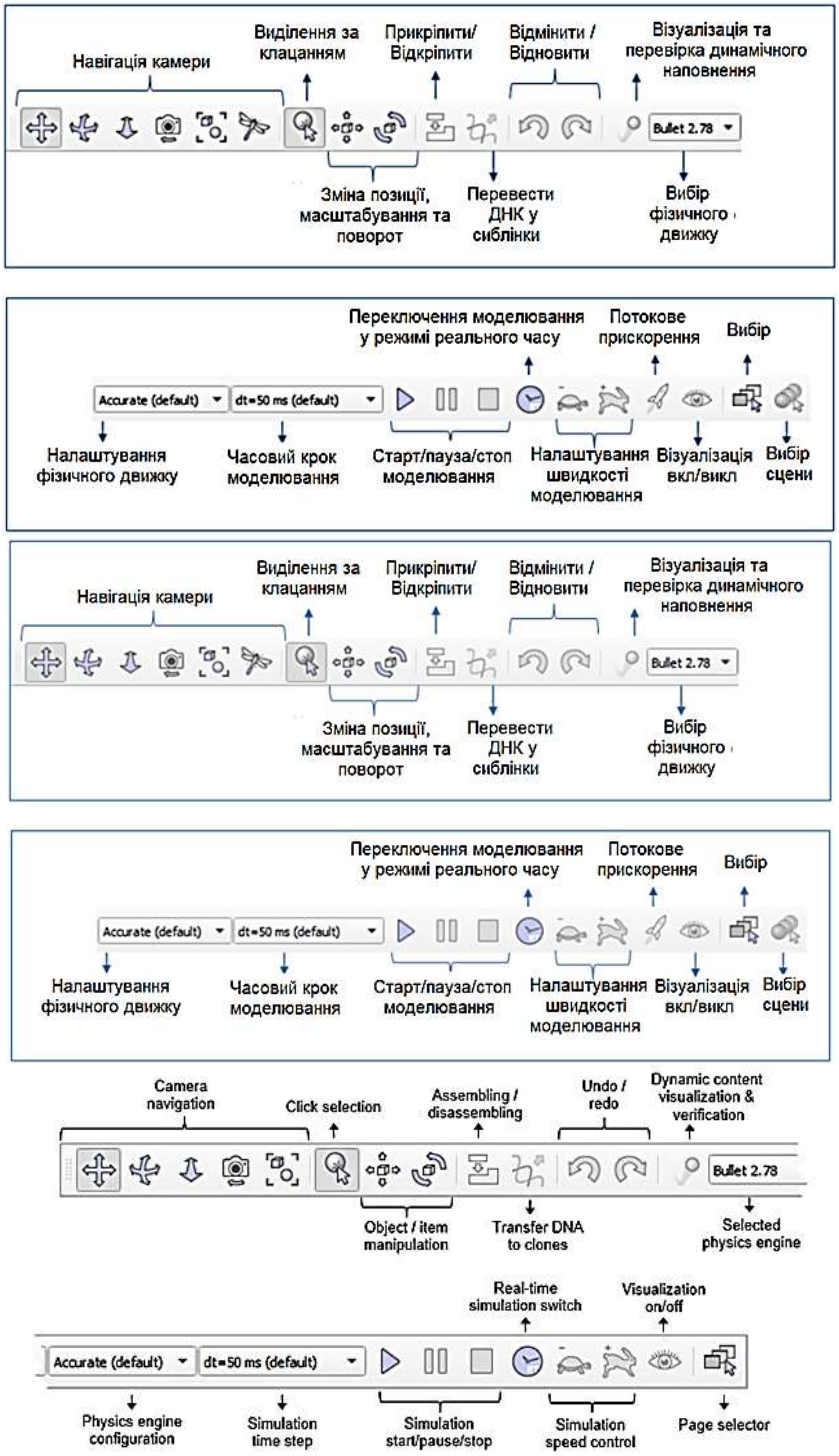


Рис. 10.2. Горизонтальна панель інструментів

Вертикальна панель інструментів V-REP наведена на рис. 10.3.

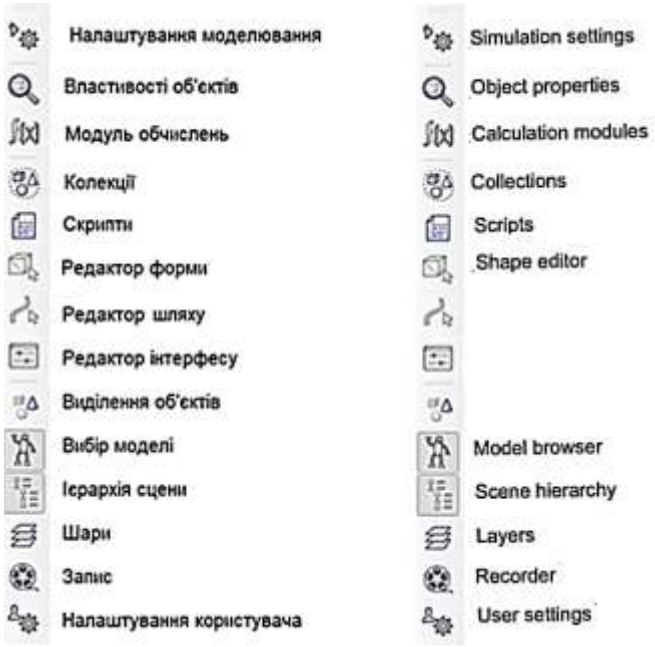


Рис. 10.3. Вертикальна панель інструментів

Кожен інструмент із цих панелей може перебувати в активному та пасивному режимі.

При цьому активація виконується простим натисканням на піктограму інструмента.

Бібліотека моделей.

За замовчуванням цей інструмент перебуває в активному режимі, але може бути деактивований натисканням на піктограму.

В активному режимі цей інструмент відображає структуру папок та моделей, які містяться в поточній папці (рис. 10.4.).

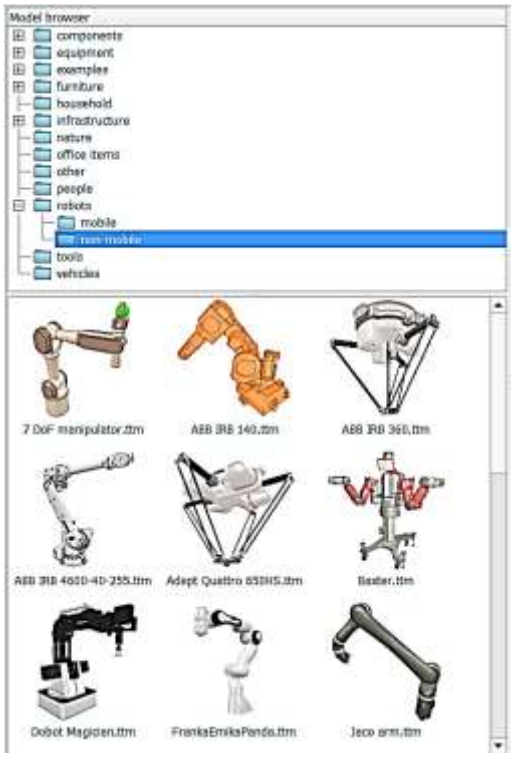


Рис. 10.5. Бібліотека моделей

Моделі з бібліотеки можна легко додати до сценарію симуляції шляхом перетягування у робоче вікно.

Бібліотека має велику кількість стаціонарних та мобільних роботів та інші елементи для створення робототехнічних машин, а саме, стаціонарні (non-mobile) роботи, мобільні (mobile) роботи, захоплюючі (grippers) пристрої, засоби пересування та руху (locomotion and propulsion), датчики, конвеєрні стрічки (conveyor bellts) та багато іншого.

Ієрархія об’єктів сценарію.

Деревоподібна ієрархія відображає структуру об'єктів та їх компонентів (рис. 10.6).

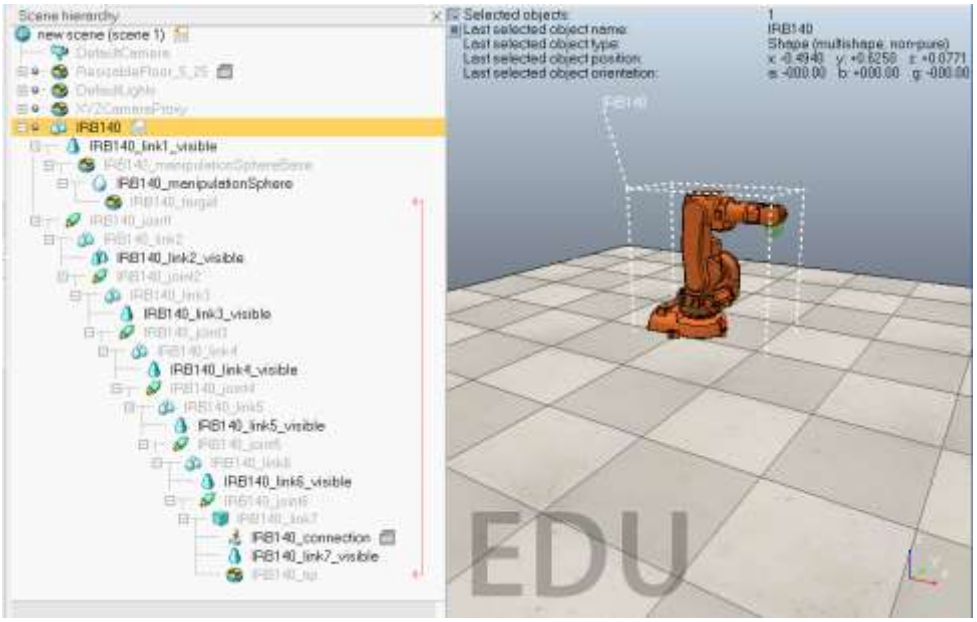


Рис. 10.6. Ієрархія об’єктів сценарію

Об’єкти можна змінювати або додавати нові елементи. Ієрархія об'єктів є ключовим елементом, який визначає основні зв'язки між об'єктами сценарію.

Також властивості будь-якого об'єкта можна відкрити подвійним натисканням лівої кнопки на піктограму ліворуч від назви об'єкта в ієрархії сцени.

Також можна вибрати спочатку необхідний елемент за допомогою одного натискання на елемент, а потім активувати інструмент «Властивості об'єктів» («Scene Object Properties»).

Для зміни назв компонентів достатньо навести курсор на поточну назву та двічі натиснути на ліву кнопку миші, після чого ввести нову назву та завершити натисканням кнопки введення (Enter).

Слід врахувати, що допускається використання літер латинського алфавіту.

Однією з ключових функцій, реалізованих у вигляді ієрархії об'єктів, є взаємозв'язок компонентів системи.

Для цієї функції є 2 способи.

Перетягуючи один об'єкт на інший, можна встановити відносини між ними (зробити одного з них «батьком» по відношенню до іншого).

Аналогічну дію можна виконати через спливаюче меню, для цього потрібно вибрати спочатку «дочірній» елемент, затиснути клавішу Shift і потім вибрати батьківський, правою кнопкою миші відкрити спливаюче контекстне меню і вибрати Edit і Make last selected object Parent.

Головна сторінка (основне вікно). Кожен сценарій симуляції може містити до 8 сторінок, які, у свою чергу, можуть містити необмежену кількість областей відображення.

Подібний розділ дозволяє максимально ефективно використовувати графічний інтерфейс для виведення різних типів даних одночасно.

Області відображення. Області відображення є області інтерфейсу, призначені для розділеного виведення даних сценарію та візуалізації, отриманих за допомогою віртуальних відеокамер, сенсорів різного типу та іншої заданої користувачем інформації.

Інформація. Блок з інформацією відображається для окремих моделей, компонентів або сценарію залежно від вибору користувача та показує різні параметри.

Глобальна система координат. Позначення орієнтації завжди є у правому нижньому кутку. Також у кожного об'єкта V-REP є локальна система координат.

Вікно відображення інформації, вибраної користувачем це вікно, що настроюється користувачем, відображає необхідну інформацію або діалог з користувачем.

Спливаюче меню. З'являється при натисканні правою кнопкою миші, має широкі можливості та дублює частину інструментів та функцій з панелей інструментів.

Діалогове вікно. Відображає властивості об'єктів з можливістю редагування, може каскадно розширюватися в деяких випадках. Вікно відчиняється при активації окремих інструментів. Параметри, що відображаються у цьому вікні, залежать від типу вибраного елемента та призначення інструменту.

Рядок стану. Область інтерфейсу використовується для виведення текстової інформації під час симуляції.

Об'єкти порожньої симуляції. При створенні нового сценарію симуляції у програмі V-REP вже є кілька об'єктів, які додаються за замовчуванням: джерело світла, підлога зі змінним розміром та камери.

Звичайно, їх можна видалити, але майже завжди вони необхідні для створення симуляції, іноді виникає потреба в їх налаштуванні.

Для зміни розміру поверхні підлоги достатньо виділити його в ієрархії сцени і в інтерфейсі користувача переміщати повзунок до тих пір, поки не вийде необхідний результат.

10.3. Огляд інструментів платформи V-REP / CoppeliaSim

Далі наводиться опис основних інструментів, без знання та розуміння яких стає неможливим створення симуляцій робототехнічних систем. Слід зазначити, що більшість інструментів V-REP дублюється у різних меню, але при цьому їх призначення та функціональність залишаються абсолютно ідентичними.

Наприклад, деякі блоки інструментів з головного меню можна знайти у спливаючому меню, зроблено це виключно для підвищення зручності використання.

Зміна положення, орієнтації та масштабування. Це з основних інструментів, необхідні введення початкових умов симуляції.

Інструменти «положення» та «орієнтація» наведені на рис. 10.7.

Рис. 10.7. Інструменти «положення» та «орієнтація»

За допомогою інструмента «Object/Item Shift» (ліворуч на рис. 10.7) можна задавати положення об'єктів у сцені та масштаб (зменшувати або збільшувати об'єкт).

Інструмент "Object/Item Rotation" (праворуч на рис. 10.7) дозволяє задавати орієнтацію об'єкта в просторі.

Зміна властивостей об'єкта. Інструмент «Властивості» не менш часто використовується при створенні симуляцій у програмі V-REP і дозволяє задавати різні властивості об'єктів і компонентів симуляції.

Після активації інструмента виводиться контекстне меню, яке має дві вкладки.

У першій вкладці наводиться перелік властивостей, який є спеціалізованим та залежить від типу вибраного об'єкта.

У другій вкладці контекстного меню інструмента наводяться загальні властивості, які для більшість об'єктів схожі.

Запуск та зупинка симуляції (рис. 10.8). Для цих завдань у горизонтальній панелі є набір кнопок на панелі інструментів, які дозволяють запускати симуляцію (на рис. 10.8, зліва), зупиняти (на рис. 10.8, посередині) та завершувати симуляцію (на рис. 10.8, праворуч).

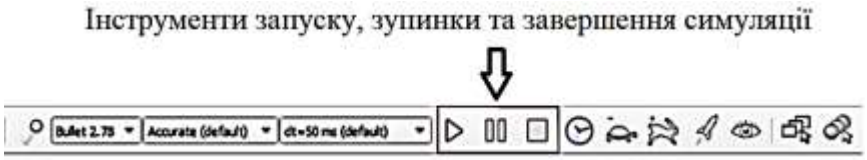


Рис. 10.8. Інструменти запуску, зупинки та завершення симуляції

Також на рисунку ліворуч від інструменту запуску симуляції показані налаштування вирішувача (бібліотеки для моделювання законів фізики), де можна змінити крок моделювання, режим моделювання та ядро фізичного двигуна, що використовується.

Управління візуальними властивостями. У програмі V-REP реалізовано механізм пошарового розподілу всіх візуальних об'єктів. Даний функціонал є необхідним, оскільки у програмі потрібно використовувати поєднання компонентів різних типів, і в таких випадках стає неможливою подальша робота з об'єктами.

Наприклад, наявність однакових компонентів, які мають однакове положення, але моделюють різні властивості, призводить до того, що вони візуально стають нерозрізняними. Тому необхідно користуватися інструментом «Шари». Рекомендований принцип рознесення на шари: компоненти одного типу мають бути на окремому шарі (наприклад, усі шарніри). Всього в V-REP доступно 10 шарів, властивості кожної компоненти є можливість вибору шару, на який буде винесений елемент. Після того, як елементи рознесені шарами, можна активувати інструмент «layers» («шари») з лівої панелі інструментів і перемикатися між шарами.

Інструмент керування скриптами. Додавання та видалення скриптів, а також зміна прив'язки до окремих компонентів, легко виконується через ієрархію сценарію симуляції. Однак V-REP має і дублюючий інструмент "Scripts", який також дозволяє виконувати аналогічні операції і часто робить це швидше. Але для початківців роботу у V-REP цей інструмент може становити труднощі, т.к. принцип його не є настільки інтуїтивно зрозумілим, як аналогічна операція через ієрархії компонентів.

Редактор форм. Даний інструмент призначений для редагування сітки механічних елементів системи, що моделюється. Установка дрібнішої сітки дозволяє отримати більш високу точність симуляції, але якщо подрібнити сітку у всіх компонентах, це призводить до збільшення споживаної обчислювальної потужності. Тому необхідно задавати розміри сітки змінної концентрації за допомогою редагування сітки у ручному режимі.

Найчастіше цей інструмент необхідний для створення симуляції на основі тривимірних моделей, які були імпортовані з CAD-системи.

У редакторі форм доступно 3 режими роботи: режим редагування трикутників, режим редагування вершин та режим редагування кромки.

Режим редагування трикутників. У цьому режимі окремі трикутники, що складають форму, можуть бути виділені, після чого їх можна видаляти, закривати решту порожнечі і розбивати на дрібніші трикутники. З допомогою кнопки «Subdivide largest triangles» можна розбити всі трикутники більш дрібні, а саме, кожне натискання зменшує розмір трикутників у 2 рази.

Режим редагування вершин. У цьому режимі окремі вершини, що становлять фігуру, можуть бути виділені, а потім видалені. Також є можливість створення нових

трикутників, для цього необхідно вибрати три або більше вершин і натиснути «Insert triangles».

Режим редагування кромки. У цьому режимі доступні функції, аналогічні до описаних вище. Відмінність полягає лише в тому, що для виділення та редагування доступні окремі кромки форми.

Слід зазначити, що складові форми не можна редагувати за допомогою даного інструменту, їх необхідно заздалегідь розгрупувати і вносити зміни окремо.

Контрольні запитання

1. Визначити, які мови програмування підтримує V-REP для реалізації систем управління (написання скриптів).
2. Назвати, яка головна відмінність потокового скрипта від непотокового.
3. Описати, які умови потрібні V-REP для обміну даними із реальним роботом.
4. Визначити, які способи введення та виведення інформації є у V-REP.
5. Назвати, які завдання виконує ієрархія об'єктів у V-REP.
6. Описати, яким чином можна відкрити властивості окремих об'єктів для редагування.
7. Визначити, як можна задати взаємозв'язок двох елементів системи, які контактують один з одним.
8. Назвати, чи можна змінювати крок моделювання в V-REP під час створення сценарію симуляції.
9. Описати, як можна відкрити властивості об'єктів.
10. Визначити, як можна перемістити елементи сцени до V-REP.

11. Конструювання робототехнічних систем в V-REP / CoppeliaSim

11.1. Написання скриптів у програмі V-REP / CoppeliaSim

Як було зазначено вище, платформа V-REP підтримує роботу з різними мовами програмування, у тому числі з найбільш поширеними, такими як C++ і Python.

Більш детальне знайомство з можливостями написання скриптів будемо виконувати з використанням мови Lua, оскільки ця мова є вбудованою мовою V-REP, і розпочати написання скриптів можна відразу після запуску програми.

Відмінною особливістю Lua є простота: багато в чому синтаксис схожий на популярну мову C, що значно спрощує роботу для тих, хто вже знайомий з мовою.

Скрипти V-REP відкривають великі можливості для реалізації управління як окремими об'єктами сцени, так і платформою.

Для початку необхідно згадати, що скрипти в V-REP бувають двох типів: потокові та не потокові.

Розглянемо структуру потокового скрипту, як рекомендованого розробниками і найчастіше використовуваного. Такий скрипт завжди складається з кількох блоків.

Блок ініціалізації. Вміст блоку виконується лише один раз при запуску симуляції. У цьому блоці проводять такі операції, як оголошення необхідних змінних і присвоєння їм вихідних значень. Також у цій частині скрипта задаються обробники для керування об'єктами сцени.

Блок активації. Вміст блоку виконується ітеративно через рівні проміжки часу (крок моделювання). Частина скрипта, написана в цьому блоці, повторюватиметься аж до зупинки симуляції або виникнення критичної помилки (при якій теж буде зупинено симуляцію). Тут описується основний алгоритм управління.

Блок керуючого коду сенсорів. Вміст даного блоку виконується стільки ж разів, скільки блок активації. Однак основний скрипт V-REP звертається до цього блоку лише після завершення виконання скрипта з блоку активації. Цей блок розроблений для отримання даних із сенсорів.

Блок завершення симуляції. Цей блок дозволяє вибірково стерти дані, отримані під час симуляції. Скрипт цього блоку починає виконуватися один раз перед завершенням симуляції або видаленням скрипта. Цей блок зазвичай залишається порожнім

Скрипти не потокового типу мають простішу структуру. Весь скрипт може вказуватися у файлі без поділу на блоки, проте з огляду на ітеративність моделюваних процесів майже завжди необхідна наявність основного циклу.

Класична структура не потокового скрипту в програмі V-REP, маэ такы блоки:

Блок ініціалізації

Основний блок керуючого циклу

Блок завершення скрипта (очищення)

Проте основний цикл там може бути відсутнім при вирішенні завдань певного класу.

Основні конструкції при написанні скриптів.

Коментарі на мові Lua повинні починатися з подвійного дефісу (рядок 4 на наступному рисунку).

Рекомендується не видаляти фрагменти скрипту під час розробки алгоритмів управління, а позначати їх як коментар, щоб потім не довелося заново писати, якщо вони будуть потрібні.

Як правило, жоден скрипт не обходиться без використання змінних. У Lua допускається оголошення нових змінних у будь-який момент часу, також при оголошенні можна задати їхнє початкове значення (наприклад, рядок 9 на наступному рисунку).

Також немає потреби вказувати тип змінної. Lua визначить його залежно від значення, яке надається змінною вперше. Змінні з числовим значенням сприймаються як число з точкою, що плаває.

Також допускається використання змінних із символьними значеннями та змінних логічного типу (можливі значення: «істина» або «хиба»).

Є й функція знищення (звільнення) змінних. Всі змінні Lua за промовчанням є глобальними, тобто доступні з будь-якої області.

Розгалуження та цикли реалізовані в Lua так само, як і в більшості С-подібних мов програмування з деякими застереженнями.

Умовний оператор "If" (Якщо) вимагає обов'язкового використання конструкції "then/end".

При цьому після ключового слова If необхідно вказати логічний вираз або змінну.

Після логічного виразу слідує ключове слово "then", за яким починається "тіло" умови - частина скрипта, яка буде виконуватися при істинності зазначеної в скрипті умови.

В умовному операторі є необов'язкова складова - додаткова умова «інакше» (elseif), «тіло» цієї умови – частина скрипта, яка виконується лише якщо зазначена у скрипті умова виявиться хибною.

На рис.11.1 наведено приклад використання умовного оператора.

```
11  k=10
12  if k ~= 10 and k<0 then
13      print('negative')
14      simAddStatusbarMessage('negative')
15  else
16      print('positive')
17      simAddStatusbarMessage('positive and k='..k)
18  end
```

Рис.11.1. Приклад використання умовного оператора та функцій виведення інформації у консольне вікно та вікно стану

Умовний оператор також допускає комбінування умів за допомогою операторів "and" (логічне "і") та "or" (логічне "або"). Рядки 13 та 16 на попередньому рисунку виводять ключові слова базовими функціями Lua у консольне вікно, яке за умовчанням приховано.

Логічніше та зручніше виводити дані не в консольне вікно, а в рядок стану V-REP, для цього необхідно скористатися регулярною функцією V-REP (приклад наведено на попередньому рисунку, рядки 14 та 17). Також на попередньому рисунку у рядку 17 виводиться не тільки ключове слово, а й значення змінної.

Як і у випадку з умовними операторами, система управління рідко обходиться без використання хоча б одного циклу. Цикли Lua задаються за допомогою ключових слів, що позначають тип циклу спільно з ключовими словами «do» і «end».

При цьому порядок такий: спочатку вказується тип, потім умова виконання циклу, далі слово "do", далі слідує "тіло циклу" - фрагмент скрипту, який виконуватиметься циклічно, і закінчується цикл словом "end".

Найбільш поширеними є цикли типу «while» та «for».

На рис. 11.2 наведено приклад найпростішого циклу while, який збільшує на одиницю значення змінної «k», доки умова «k<50» не стане хибною

```
10  k=1
11  while k < 50 do
12  k = k + 1
13  end
```

Рис. 11.2. Приклад найпростішого циклу while

Цикли Lua, як і умовне розгалуження, повинні завершуватися ключовим словом «end».

Особливу увагу слід приділити умові, що використовується у циклі.

Особливо це важливо, коли використовується значення змінної замість логічної умови.

У невизначених змінних значення за промовчанням дорівнює "nil".

При цьому в умові циклу лише змінні зі значенням «nil» та «false» (логічний тип змінної «хиба») повертають false, тоді як значення змінної «0» і ’ ’ повертають true.

Другий тип циклів, який найчастіше застосовується, - "for". Даний тип добре підходить для завдань, коли необхідно виконати цикл з лічильником, але слід згадати, що в більшості випадків "for" можна замінити на цикл "while", задіявши кілька додаткових змінних.

Приклад використання циклу "for" наведено на рис. 11.3. Даний фрагмент скрипту виконує 100 ітерацій починаючи від 1 (включаючи 1 і 100).

```
10    Sum = 0
11    for i = 1, 100 do -- 100 iterations from 1.
12     Sum = Sum + i
13    end
```

Рис. 11.3. Приклад використання циклу «for»

При цьому можна поміняти умови місцями, замість 1 поставити 100 і 100 замість 1, тоді цикл виконуватиметься зі зміною значення змінної «і» від 100 до 1.

Також у циклі «for» є необов'язковий параметр кроку, за замовчуванням крок дорівнює одиниці, і немає потреби його вказувати. Однак, для випадків, коли потрібне використання кроку зі значенням відмінним від 1, то потрібно вказати його через ком після кінцевого значення (якщо необхідний крок 2, то для прикладу на наступному рисунку це буде умова «і = 1, 100, 2»).

Таблиці в Lua є єдиним структурним елементом, вони поєднують у собі властивості масиву, хеш-таблиці («ключ» - «значення»), структури та об'єкта. Найчастіше таблиці використовуються як словники, а ключ при цьому за умовчанням має рядковий (символьний) тип. Приклад наведено на наступному рисунку. Рядок 11 оголошує змінну типу «таблиця» і задає 2 ключі та відповідні їм значення. Доступ до значень можна отримати вказівкою назви таблиці та ключа через точку (приклад на рис. 11.4).

```
11    t = {key1 = 'value1', key2 = false}
12    print(t.key1) -- 'value1'
13    t.newKey = {} --add new key
14    t.key2 = nil -- delete key2
```

Рис. 11.4. Доступ до значень

Як видно з прикладу, допускається використання в якості ключа не тільки рядків, але також всіх інших типів змінних, які доступні в Lua.

Іншою та вкрай важливою складовою мови програмування є функції. Функції можна створювати власні, також можна використовувати вже готові, які можна знайти у довіднику по Lua.

Приклад заданої розробником функції наведено на рис. 11.5.

```
6  function bar(a, b, c)
7      sum=a+b+c
8      r=a-b-c
9    return sum,r
10 end
11 k,p=bar(2,2,3)
12 simAddStatusbarMessage('k='..k)
13 simAddStatusbarMessage('p='..p)
```

Рис. 11.5. Приклад заданої розробником функції

Як видно з прикладу, функція може набувати кількох значень. Також функції можуть повертати кілька значень

Особливу увагу варто приділити регулярним функціям V-REP. Ці функції вже вбудовані в Lua при написанні скриптів V-REP і починаються на «sim». Саме ці функції використовуються для взаємодії з симуляцією: управління, зчитування даних, налагодження та вирішення багатьох інших завдань.

Повний список функцій з докладним описом змінних і даних, що приймаються ними, доступний в офіційній документації V-REP.

11.2. Типи об’єктів у програмі V-REP / CoppeliaSim

У програмі V-REP є велика кількість об'єктів різного призначення. Деякі з них застосовуються вкрай рідко, а деякі потрібні під час моделювання кожної мехатронної та робототехнічної системи.

Далі викладено основні типи об'єктів, їх властивості та призначення.

Типи фізичних об'єктів (форм).

Форми V-REP являють собою жорсткі сітчасті об'єкти, що складаються з трикутних граней.

Вони можуть бути імпортовані, експортовані та відредаговані.

Форми входять у три різних підтипи: проста випадкова форма, складова випадкова форма, проста опукла форма, складова опукла форма, чиста проста форма, чиста складова форма.

На рис. 11.6 представлені піктограми всіх типів форм у тому порядку зліва направо, як зазначено вище

Рис. 11.6. Піктограми типів форм

Піктограми типу форми виводяться у головній ієрархії сцени ліворуч від назви кожного компонента механічної складової робототехнічної системи. Як було зазначено, всі вони поділяються на три типи, і кожен тип - на простий (єдиний) та складовий (що складається з кількох простих).

Відмінні риси випадкових форм полягають у тому, що вони можуть бути будь-якої форми, але для моделювання динаміки їх не рекомендується використовувати, т.к. це вимагатиме значних обчислювальних ресурсів. Тому вони застосовуються часто для отримання хороших візуалізацій, найчастіше моделюють лише візуальні властивості та є повторюючими рух опуклих та чистих форм.

Випуклі та чисті форми містять оптимізовану сітку і добре підходять для моделювання динаміки, проте візуальні властивості опуклих форм найчастіше виглядають не дуже добро. Тому рекомендується використовувати опуклі форми разом із випадковими формами, де опукла форма бере участь у розрахунках фізичних властивостей (при цьому візуальні властивості приховані), а випадкова – у розрахунках візуальних властивостей (фізичні властивості відключені).

V-REP також дозволяє створювати оптимізовані для моделювання опуклі форми на основі випадкових форм, що детальніше розглянуто в лабораторних роботах.

Додавання форм серед V-REP може бути виконано двома способами: через вбудований інструмент «Primitive Shape» і через меню [File → import Mash from…].

В останньому випадку необхідно заздалегідь створити тривимірну модель у будь-якій CAD-системі та зберегти модель у форматі STL.

З’єнування елементів у V-REP.

«Зчленування» (Joint) - це об'єкт, який має хоча б один внутрішній ступінь свободи.

«Зчленування» використовуються для створення механізмів та завдання переміщень інших об'єктів шляхом з’єднування елементів.

У V-REP є 3 типи «Зчленування»: обертальний, призматичний і сферичний.

«Зчленування» додаються до сценарію симуляції через головне меню: [Add → Joint].

Нові «Зчленування» мають нульові координати у глобальній системі координат сцени. Тому необхідно додати Зчленування та коректно задати необхідну орієнтацію, скориставшись інструментами «Object/Item Shift» і «Object/Item Rotation».

Після того, як у «Зчленування» буде встановлено необхідне положення та орієнтація, можна з'єднувати «Зчленування» з іншими об'єктами шляхом створення деревоподібної ієрархії. Цей процес більш детально розкритий у лабораторних роботах.

«Зчленування» з нульовим ступенем свободи у програмі V-REP виконуються без використання елементів типу «Зчленування». Два динамічні елементи можуть бути зчленовані через елемент типу «Force sensor». Існує і альтернативний варіант - скористатися контекстним меню («Edit» → «Grouping/Merging» → «Group selected shapes»), попередньо виділивши необхідні для «Зчленування» компоненти.

Через інструмент угруповання допускається «Зчленування» необмеженої кількості елементів.

Графіки.

Виведення даних на графіці є одним із найпоширеніших і зручних способів подання інформації.

У програмі V-REP є окремий клас об'єктів «Graph» (графік), за допомогою якого можна легко візуалізувати як дані з окремих сенсорів, так і дані користувача (незалежно від способу отримання). Щоб додати графіки до сценарію симуляції, достатньо в головному меню виконати [«Add» → «Graph»].

Далі потрібно задати характеристики. Для цього потрібно виділити об'єкт в ієрархії сценарію та активувати інструмент «Object/Item Properties». У контекстному меню в розділі «Data stream recording list» необхідно натиснути «Add new data stream to record».

У контекстному вікні (рис. 11.7), що з’явилось, необхідно вказати тип даних і джерело даних.

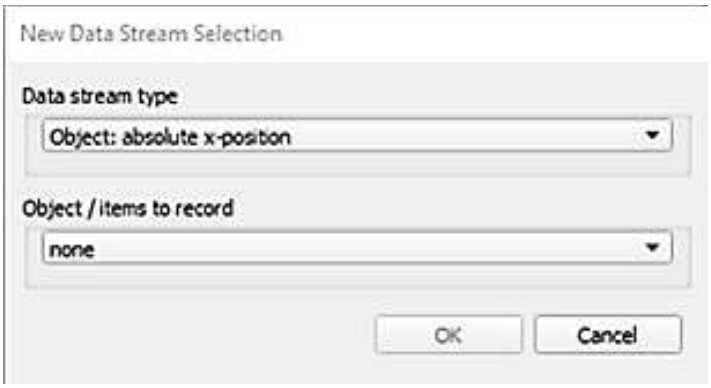


Рис. 11.7. Контекстне вікно створення потоку даних у V-REP

Якщо необхідно вивести дані з сенсора на графік, то необхідно вказати тип даних у поле «Data stream type» і назву сенсора в полі «Object/Item to record». Найчастіше виникає необхідність у попередній обробці даних із сенсорів перед виведенням на графіці: зробити це можна через використання скрипта як проміжний пункт, де й виконуватиметься обробка даних. У такому разі необхідно вказати в налаштуваннях графіка, що виводитимуться дані користувача, а саме, встановити «Various: user-defined» у полі «Data stream type», а в полі «Object/items to record» вибрати «User data».

Після натискання «ОК» у розділі «Data stream recording list» з'явиться новий потік даних під назвою за промовчанням. Подвійним натисканням на назву потоку даних можна перейти в режим редагування назви. Ця назва буде використана для звернення до графіка зі скрипту під час виведення даних.

На рис. 11.8 наведений приклад фрагмента скрипта, який виводить дані користувача зі скрипта на графік.

```
3  if (sim_call_type==sim_childscriptcall_initialization) then
4      graph=simGetObjectHandle("Graph")
5  end
6  if (sim_call_type==sim_childscriptcall_actuation) then
7      simSetGraphUserData(graph,'Red',data)
8  end
9
```

Рис. 11.8. Приклад фрагмента скрипта, який виводить дані користувача зі скрипта на графік

У прикладі на рядку 4 Graph - назва об'єкта, у рядку 7 Red - назва потоку даних, data - назва змінної, значення якої виводиться на графіку.

Отримання даних з сенсора наведено у розділі «Сенсори».

Сенсори.

У програмі V-REP доступні різні типи сенсорів: сенсор сили та моментів (force sensor), відео-сенсори (vision sensor), сенсори відстані (proximity sensor). Читання даних із сенсорів виконується у скрипті з використанням спеціальних функцій (API-функцій).

Приклади читання даних з vision sensor і proximity sensor наведені на рис. 11.9.

```
2  if (sim_call_type==sim_childscriptcall_initialization) then
3      sensor1=simGetObjectHandle('Sensor')
4      sensor2=simGetObjectHandle("Proximity")
5  end
6  if (sim_call_type==sim_childscriptcall_actuation) then
7      result1,data=simReadVisionSensor(sensor)
8      if (result1>=0) then
9          simAddStatusbarMessage('Data[1]='..data[1])
10     end
11     result2,distance=simReadProximitySensor(sensor2)
12 end
```

Рис. 11.9. Фрагменти скрипта для читання даних із сенсорів у програмі V-REP

11.3 Застосування програмного комплексу CoppeliaSim на прикладі двоколісного мобільного робота Pioneer 3-DX

Додаймо на сцену модель робота pioneer p3dx.

Для цього в оглядачі моделей у дереві robots виберемо mobile і перетягнемо робота **pioneer p3dx** на сцену (рис. 11.10).

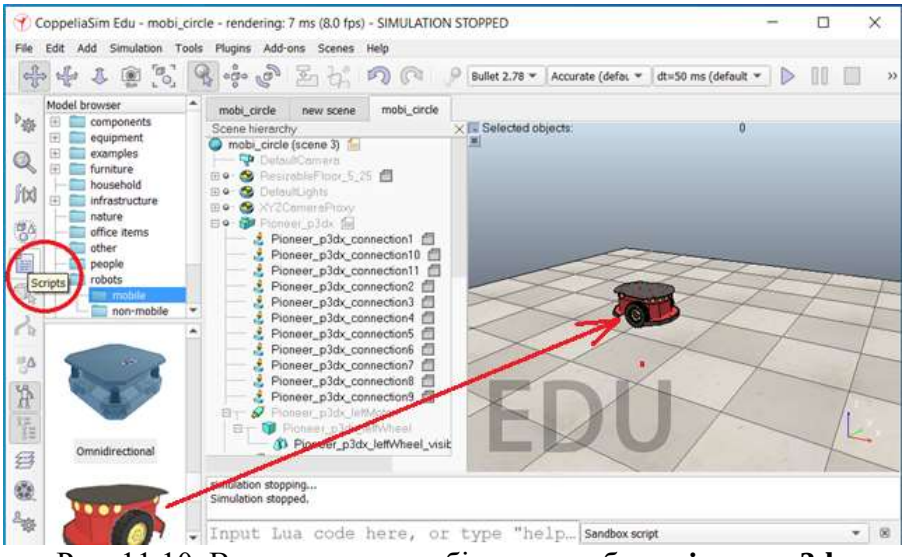


Рис. 11.10. Встановлення мобільного робота **pioneer p3dx**

Якщо після цього натиснути кнопку Play (Старт сцени), то робот поїде у прямому напрямку, логіка його роботи міститься у прикріпленому скрипті (рис. 11.11).

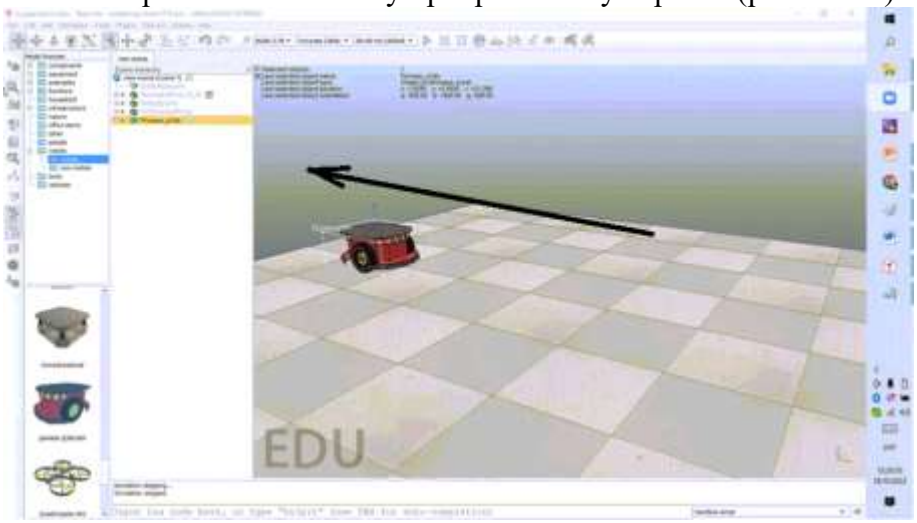


Рис. 11.11. Симуляція переміщення робота

Відредагуємо скрипт таким чином, щоб замість прямолінійного руху робот здійснював кругове. Для цього у функції sysCall_actuation() швидкість правого колеса збільшимо вдвічі.

vRight=v0*2

Для цього відкриваємо скрипт (рис. 11.12).

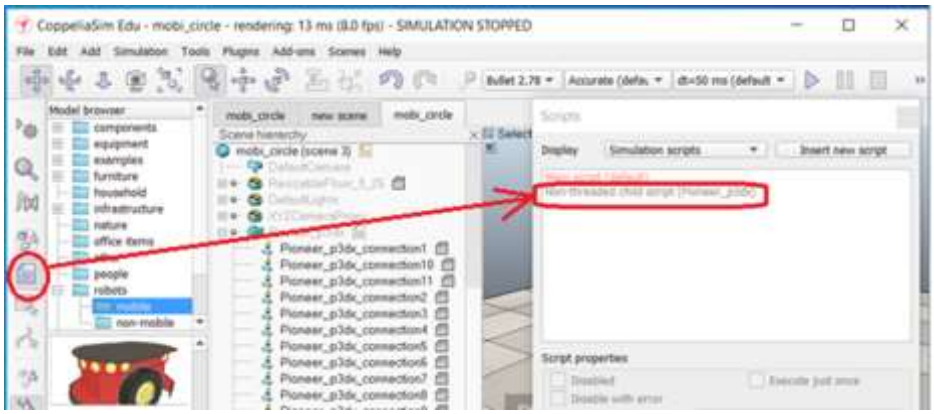


Рис. 11.12. Визначення та відкриття скрипту

Та здійснюємо його редагування (рис. 11.13).

```
function sysCall_actuation()
    for i=1,16,1 do
        res,dist=sim.readProximitySensor(usensors[i])
        if (res>0) and (dist<noDetectionDist) then
            if (dist<maxDetectionDist) then
                dist=maxDetectionDist
            end
            detect[i]=1-((dist-maxDetectionDist)/(noDetectionDist-maxDetectionDist))
        else
            detect[i]=0
        end
    end

    vLeft=v0
    vRight=v0*2

    for i=1,16,1 do
        vLeft=vLeft+braitenbergL[i]*detect[i]
        vRight=vRight+braitenbergR[i]*detect[i]
    end

    sim.setJointTargetVelocity(motorLeft,vLeft)
    sim.setJointTargetVelocity(motorRight,vRight)
end
```

Рис. 11.13. Редагування скрипту

Переконаємося у круговій траєкторії руху мобільного робота, запустивши симуляцію (рис. 11.14).

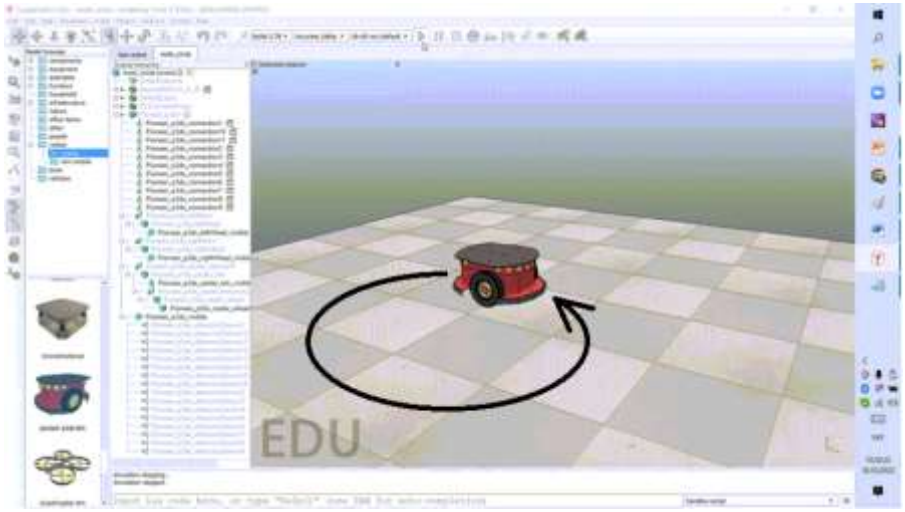


Рис. 11.14. Симуляція переміщення робота

Розглянемо створення відео симуляції руху мобільного робота (ри. 11.15).

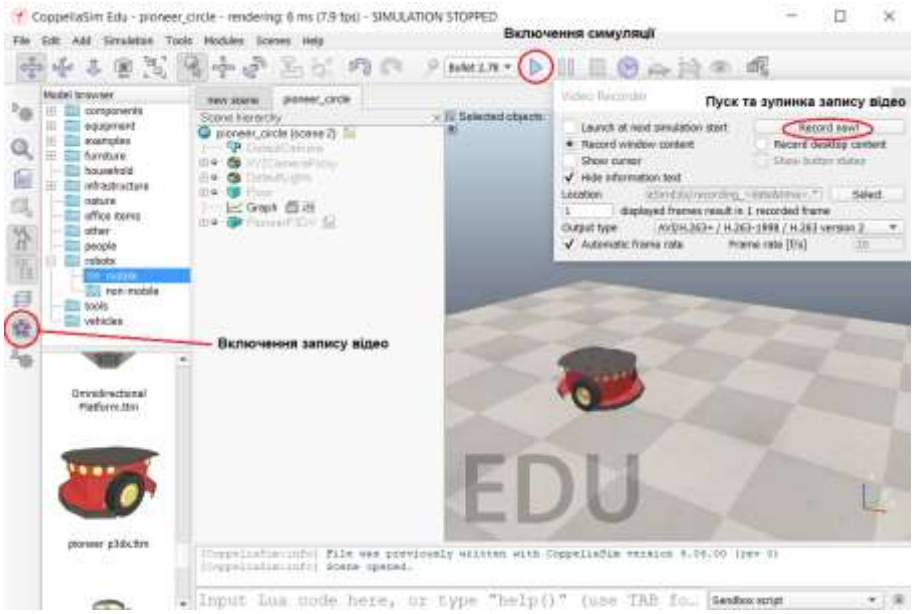


Рис. 11.15. Створення відео симуляції руху мобільного робота

Після завершення запису буде показано, де знаходиться записане відео (рис. 11.16).

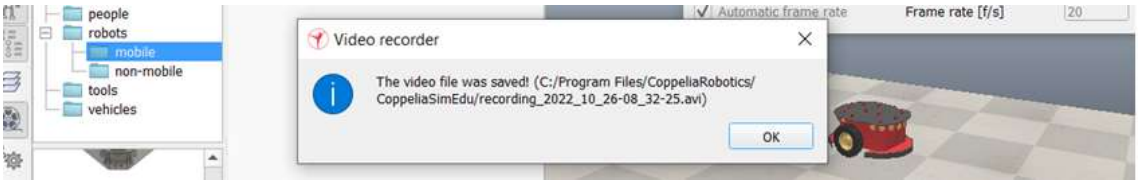


Рис. 11.16. Місце знаходження відео

Побудуємо графік траєкторії руху мобільного робота pioneer p3dx.
Повернемось у вихідне вікно.
На панелі меню вибираємо Add --> Graph.
Знову додаємо на сцену модель робота pioneer p3dx.

Для переміщення по колу, як це було зроблено раніше, у функції sysCall_actuation() швидкість правого колеса збільшимо вдвічі.

vRight=v0*2

Далі в скрипті мобільного робота, що містить логіку роботи, у функції sysCall_init() додаймо наступний код:

graph=sim.getObjectHandle('Graph')
objectHandle=sim.getObjectHandle('Pioneer_p3dx')
objectPosX=sim.addGraphStream(graph,'x','m',1)
objectPosY=sim.addGraphStream(graph,'y','m',1)
sim.addGraphCurve(graph,'object pos x/y',2,{objectPosX,objectPosY},{0,0},'m by m')

де 'Graph' та 'Pioneer P3dx' це імена об'єктів з ієрархії сцени.

У тілі скрипта додаймо функцію sysCall_sensing().

function sysCall_sensing()
end

До функції sysCall_sensing() додаймо наступний код:

pos=sim.getObjectPosition(objectHandle,-1)
sim.setGraphStreamValue(graph,objectPosX,pos[1])
sim.setGraphStreamValue(graph,objectPosY,pos[2])

Очистимо графік після закінчення процесу симулювання. Для цього до функції sysCall_cleanup() додаймо наступний рядок:

sim.resetGraph(graph)

І запустимо процес симулювання. Отримаємо графік траєкторії руху (рис. 11.17).

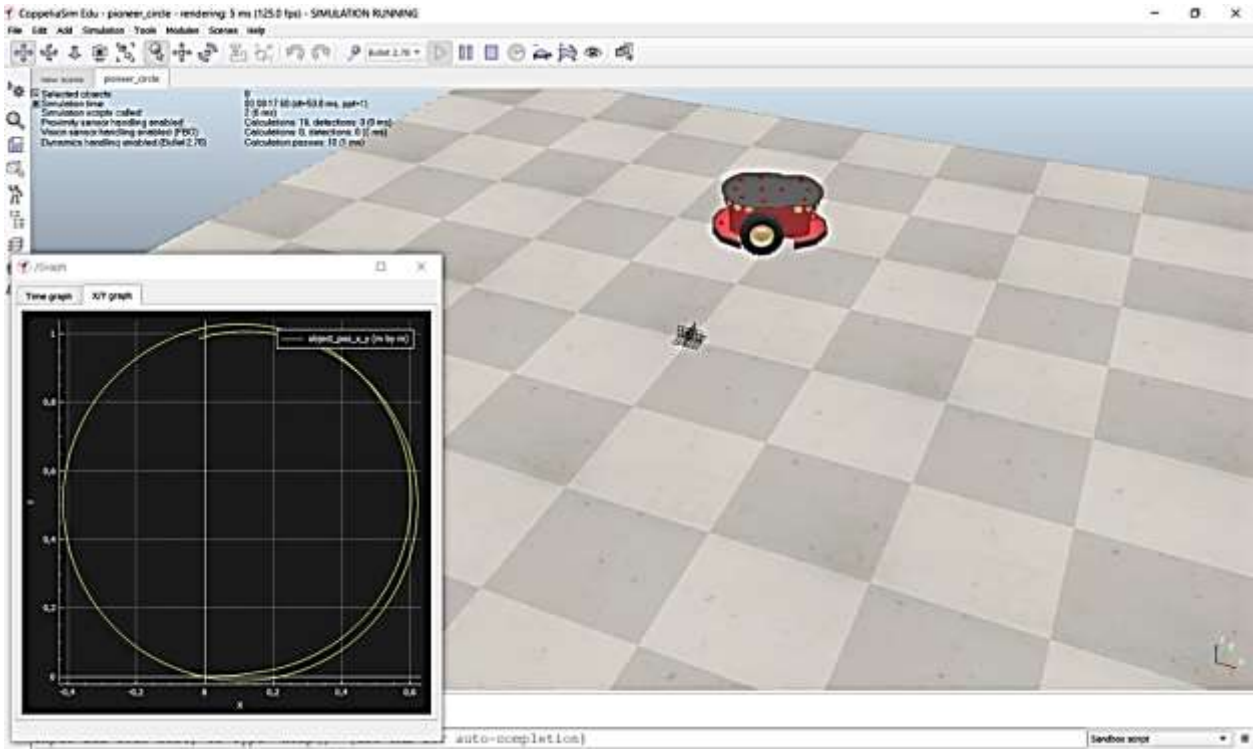
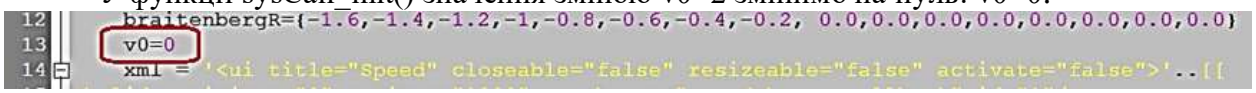


Рис. 11.17. Графік траєкторії руху

Розглянемо, як здійснити керування швидкістю переміщення робота.

Для цього вилучимо і знову додаймо на сцену модель робота pioneer p3dx.

У функції sysCall_init() значення зміною v0=2 змінимо на нуль: v0=0.



Наприкінці функції sysCall_init() додаймо наступний код:

xml = '<ui title="Speed" closeable="false" resizeable="false" activate="false">'..[[

```
<hslider minimum="0" maximum="1000" on-change="speedChange_callback" id="1"/>
</ui>
]]
ui=simUI.create(xml)
```

У функції sysCall_cleanup() додаймо наступну строку:

```
simUI.destroy(ui)
```

У тілі скрипта вставимо таку функцію:

```
function speedChange_callback(ui,id,newVal)
v0=newVal*2/1000
end
```

Запустимо симулювання. Тепер швидкістю робота можна керувати за допомогою слайдера (рис. 11.18).

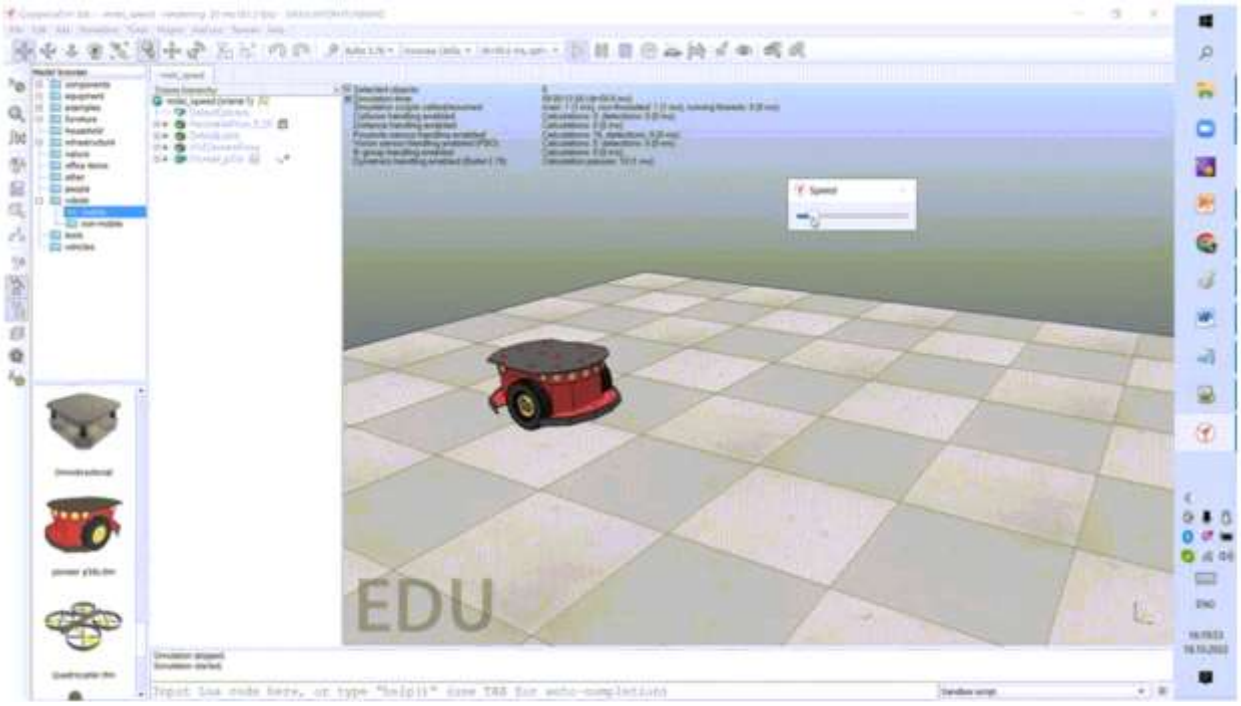


Рис. 11.18. Керування швидкістю робота за допомогою слайдера

Контрольні запитання

1. Визначити, з яких блоків складаються потокові скрипти.
2. Назвати, у якій частині скрипта допускається оголошення нової змінної.
3. Назвати, які цикли використовуються у V-REP.
4. Визначити, скільки змінних може приймати функція.
5. Назвати, чи можна створювати власні функції.
6. Описати, чи можна створити тривимірну модель робота у V-REP без використання додаткового програмного забезпечення.
7. Визначити, які типи зчленувань доступні у V-REP.
8. Визначити, чи працюватиме симуляція, якщо для моделювання динаміки використовувати «випадкові» типи форми.
9. Описати, чи є у V-REP об'єкти типу «мотор», якщо так, то як вони реалізовані.
10. Визначити, які типи даних можна виводити на графік.

12. Конструювання маніпулятора в CoppeliaSim

12.1. Конструювання ланок маніпулятора та засобів переміщення

Розглянемо, як здійснюється конструювання маніпулятора у програмі CoppeliaSim на прикладі найпростішого маніпулятора, наведеного на рис. 12.1.

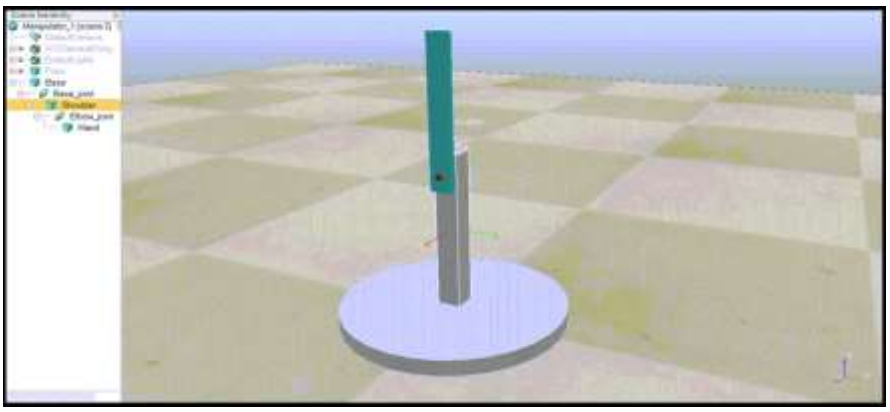


Рис. 12.1. Маніпулятор з двома ланками

Спочатку здійснимо конструювання ланок маніпулятора, зовнішній вигляд та кінематична схема якого наведені на рис. 12.2.

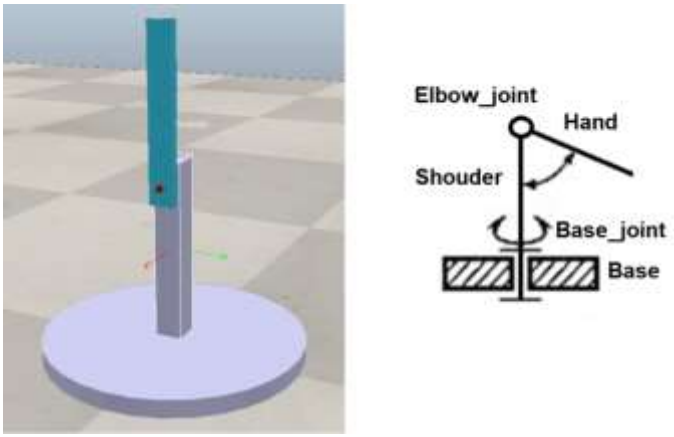


Рис. 12.2. Зовнішній вигляд та кінематична схема маніпулятора

Конструювання почнемо з створення основи маніпулятора у вигляді циліндра (рис. 12.3).

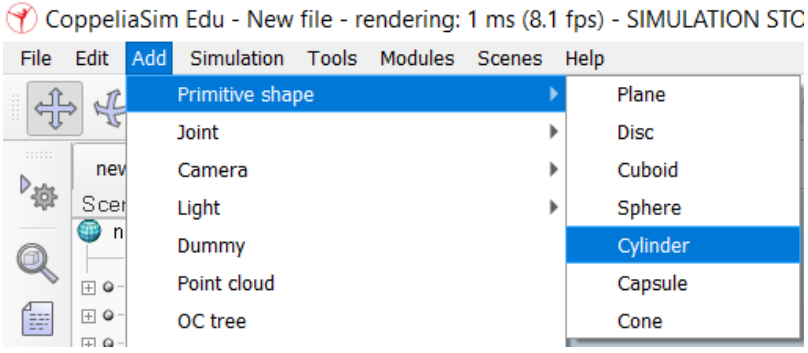


Рис. 12.3. Вибір основи маніпулятора у вигляді циліндра

З’являється вікно, в якому встановлюємо розміри основи (рис. 12.4).

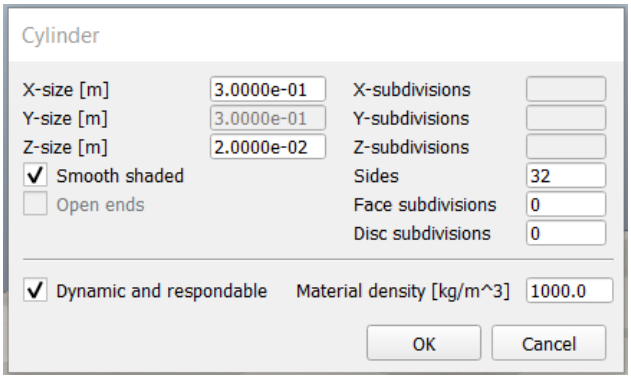


Рис. 12.4. Встановлення розмірів основи

Після цього отримаємо основу у такому вигляді (рис. 12.5).

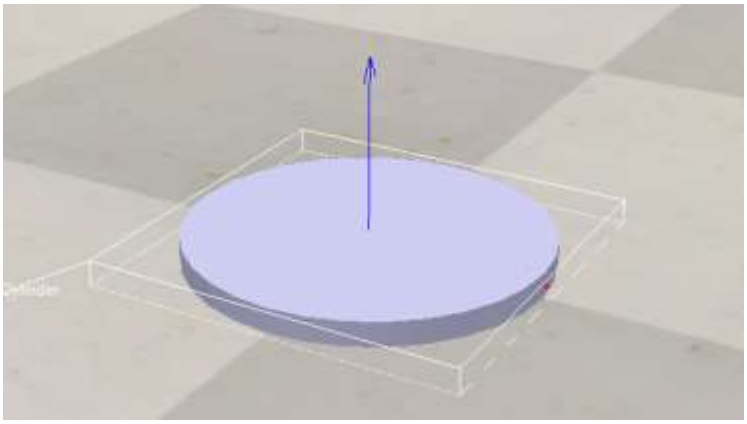


Рис. 12.5. Основа

У ієрархії сцени надаймо йому назву “BASE” (рис. 12.6).

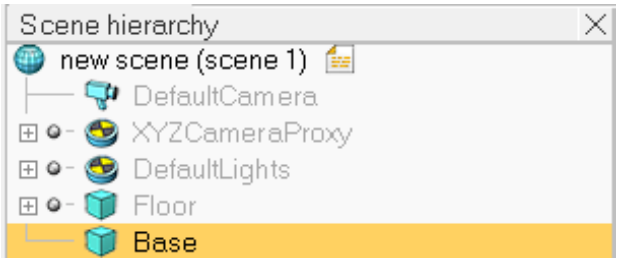
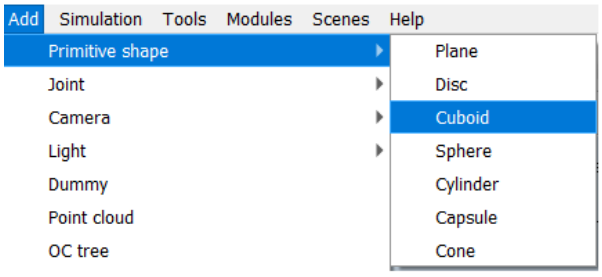


Рис 12.6. Встановлення назви основи в ієрархії

Створимо першу ланку маніпулятора, яка буде обертатися навколо осі Z, у вигляді кубоїда та встановимо її розміри (рис. 12.7).



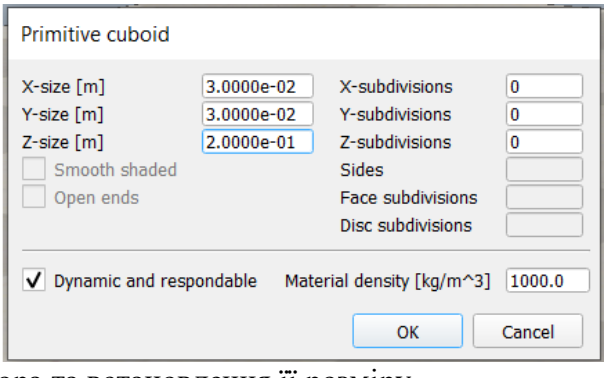


Рис. 12.7. Вибір ланки маніпулятора та встановлення її розміру

Отримаємо результат, наведений на рис. 12.8.

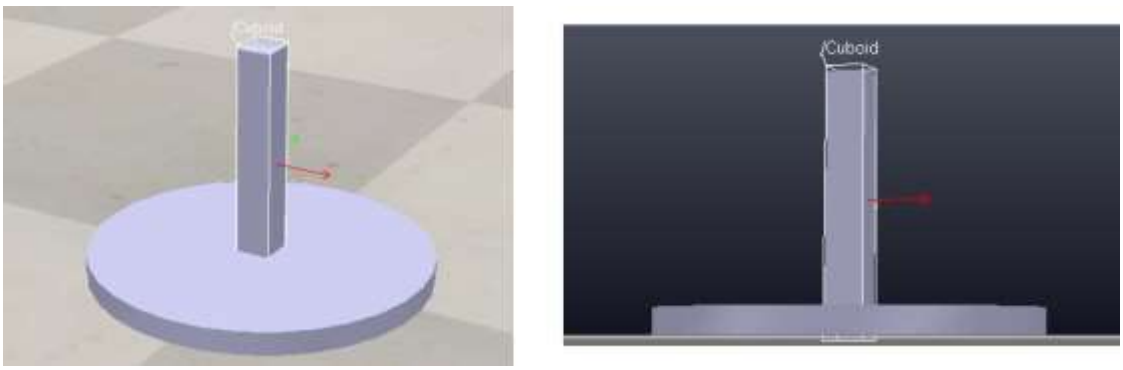


Рис. 12.8. Результат вибору ланки маніпулятора

Надаймо цій ланці назву “Shoulder” (плече). На попередньому слайді видно, що ця ланка перекривається з основою, тому її треба підняти, щоб був зазор між нею та основою (рис. 12.9).

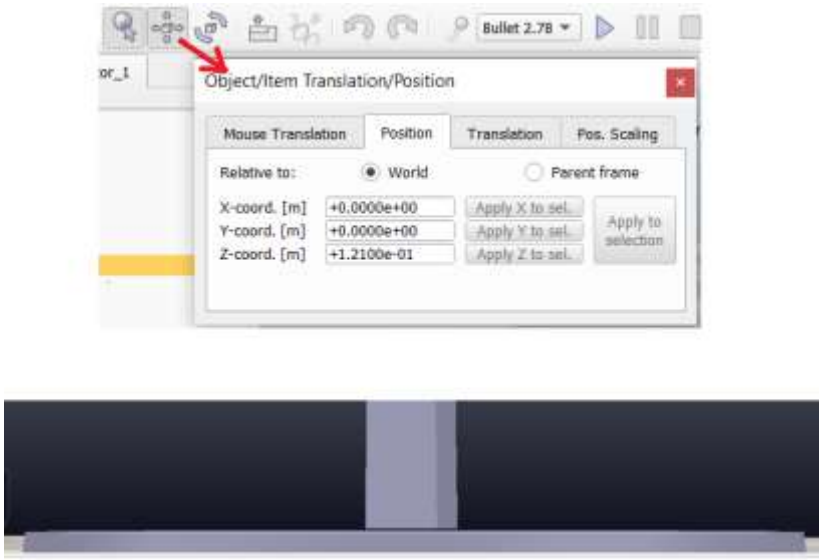


Рис. 12.9. Встановлення зазору між основою та ланкою

Створимо з’єднувач обертального типу (“Add” → “Joint” → “Revolute”) для об’єднання ланок “BASE” та “Shoulder”.

Надаймо йому ім’я “Base_joint”. Встановимо розмір, положення та динамічні властивості (рис. 12.10).

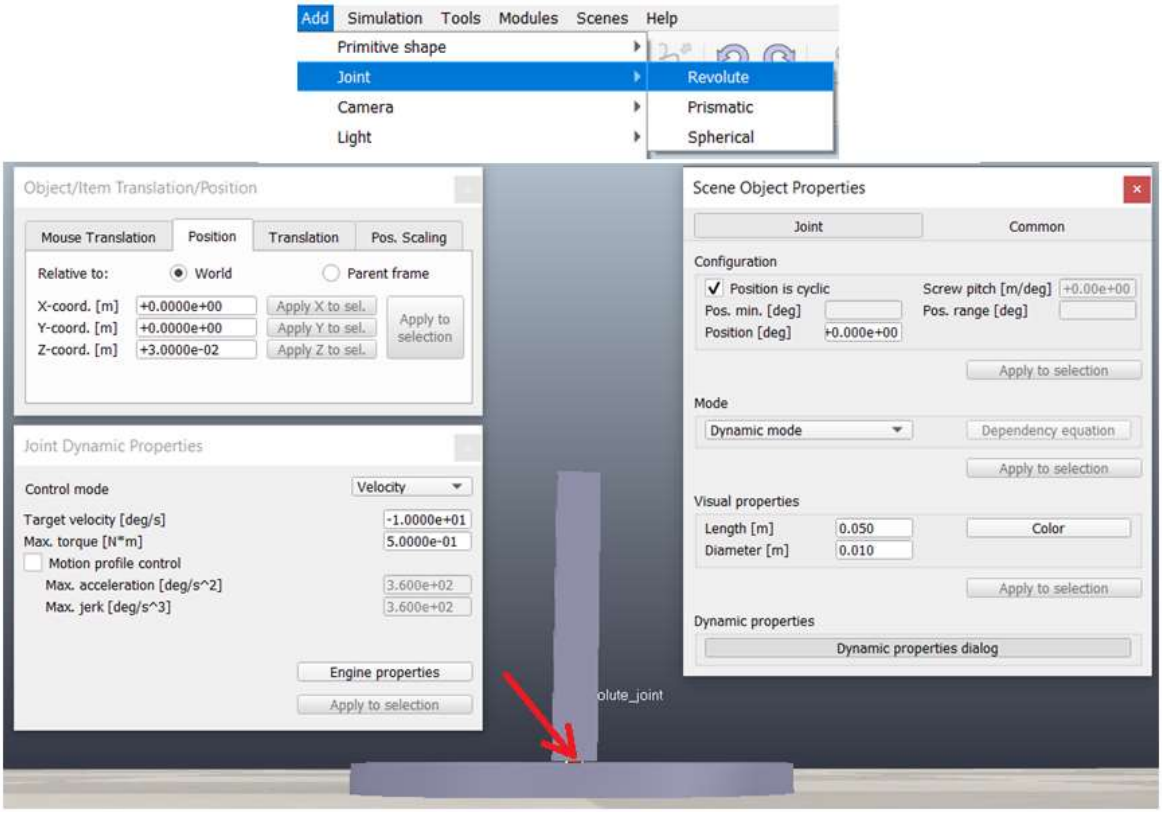


Рис. 12.10. Створення з’єднувача обертального типу

Встановимо зв’язки між елементами шляхом створення деревоподібної структури. Для цього треба виділити та перетягнути один елемент на інший у дереві моделі (рис. 12.11)..

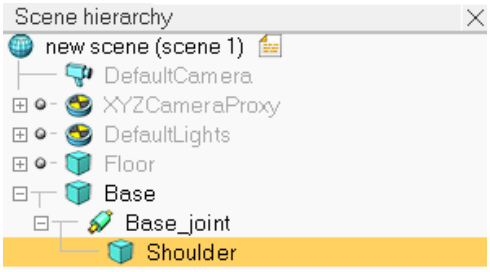


Рис. 12.11. Встановлення зв’язків між елементами

Після запуску симуляції отримаємо наступне (рис. 12.12).

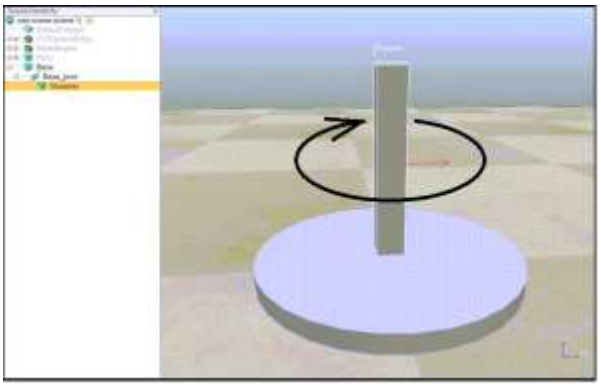


Рис. 12.12. Симуляція обертання ланки

Створимо другу ланку маніпулятора, яка буде обертатися навколо осі Y відносно першої ланки, теж у вигляді кубоїда. Встановимо її розміри та положення (рис. 12.13).

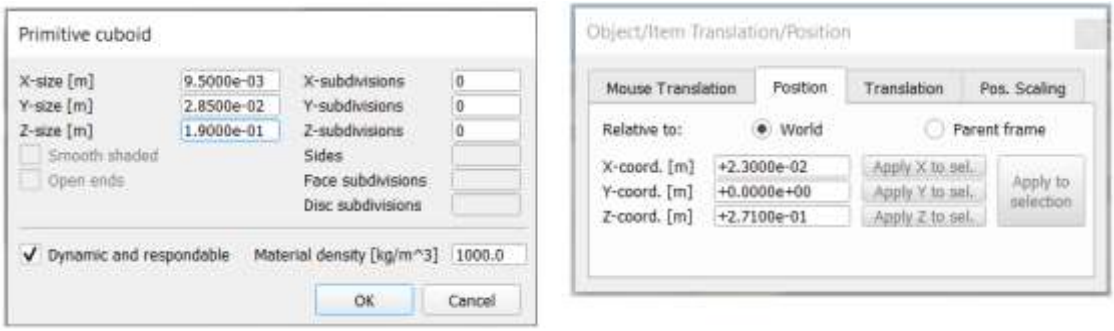


Рис. 12.13. Створення другої ланки маніпулятора

Отримаємо маніпулятор, наведений на рис. 11.14. Надаймо цій ланці ім'я "Hand" (рука).

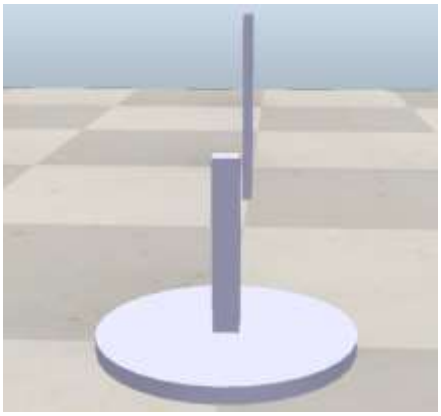


Рис. 12.14. Маніпулятор з двома ланками

Створимо з'єднувач обертального типу ("Add" → "Joint" → "Revolute") для об'єднання ланок "Shoulder" та "Hand".

Надаймо йому ім'я "Elbow_joint". Встановимо розмір, динамічні властивості та положення (рис. 12.15).

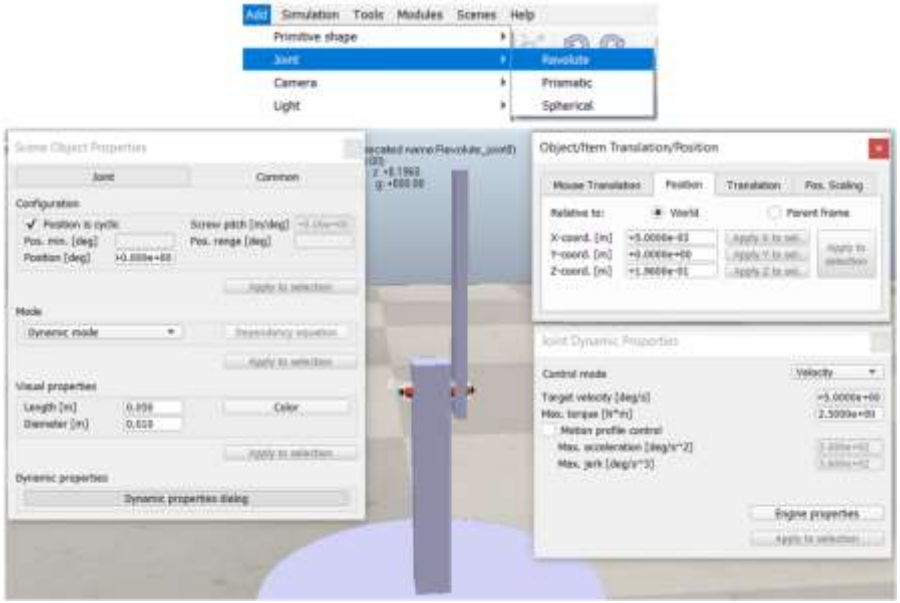


Рис. 12.15. Створення з'єднувача обертального типу для об'єднання ланок "Shoulder" та "Hand"

Оскільки у даному випадку з’єднувач повернутий на 90º, треба встановити ще орієнтацію об’єкта (рис. 12.16).

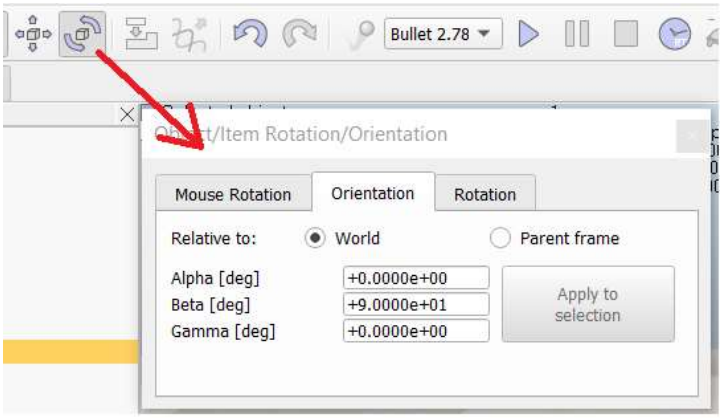


Рис. 12.16. Поворот з’єднувача на 90º

Встановимо зв’язки між елементами деревоподібної структури (рис. 1217).

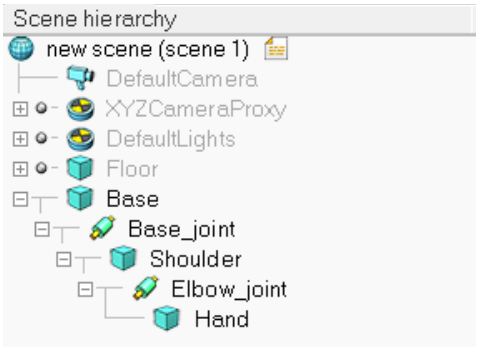


Рис. 12.17. Встановлення зв'язку між елементами деревоподібної структури

Симуляція показує переміщення усіх ланок маніпулятора, а саме, плеча відносно основи та руки відносно плеча (рис. 12.18).

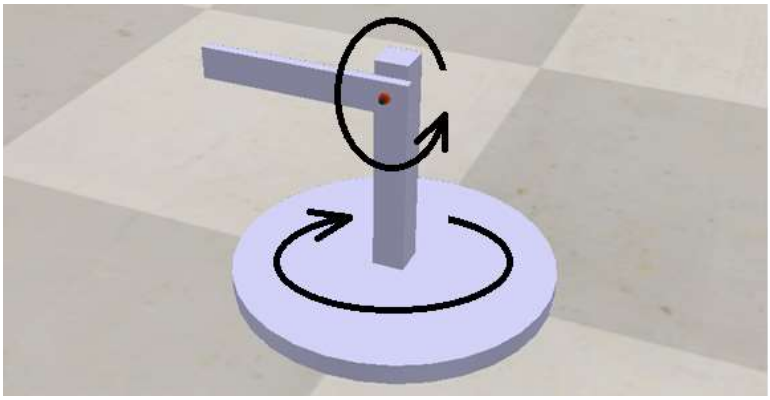


Рис. 12.18. Симуляція переміщення ланок маніпулятора

12.2. Налагодження та симуляція моделі маніпулятора

Розглянемо деякі можливості налагодження, які можна зробити без втручання у скрипти.

Використоруючи інструменти «положення» та «орієнтація» можна змінити вихідне положення окремих ланок маніпулятора. Наприклад, встановити руку у горизонтальне положення (рис. 12.19).

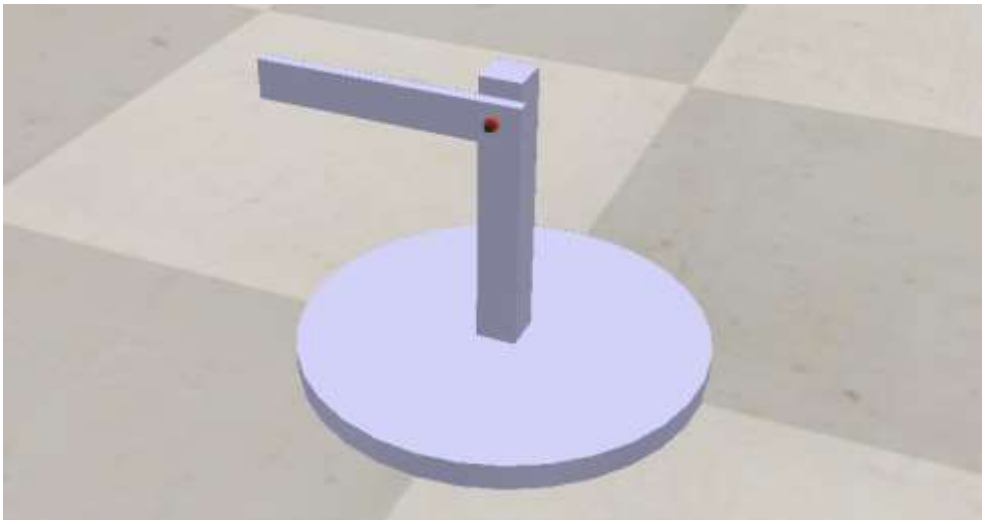
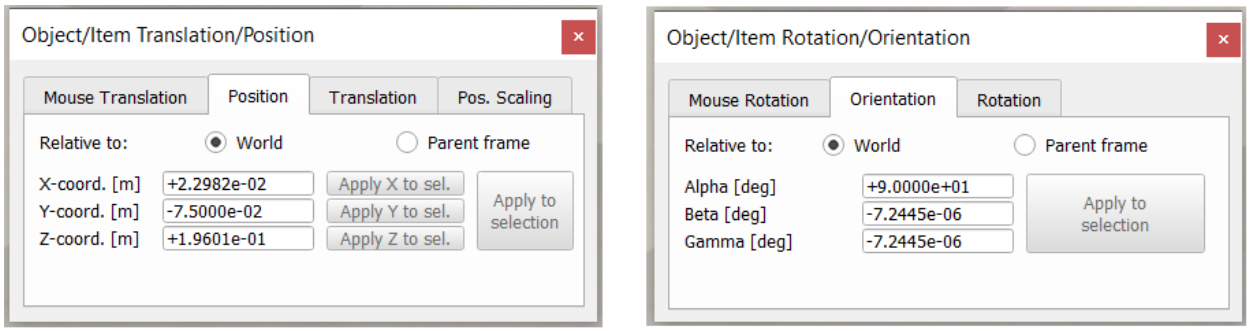


Рис. 12.19. Встановлення руки у горизонтальне положення

Для цього треба ввести такі зміни в інструменти «положення» та «орієнтація» (рис. 12.20).



а) б)

Рис. 12.20. Зміни, що треба ввести в в інструменти «положення» (а) та «орієнтація» (б) для встановлення руки у горизонтальне положення

При цьому треба враховувати, що зміна положення та орієнтації здійснюється відносно центру об’єкта.

Тому поворот руки на 90º для переведення горизонтальне положення дає результат, наведений на рис. 12.21.

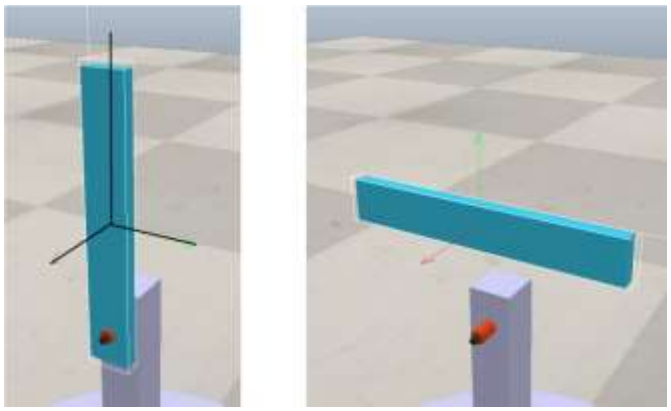


Рис. 12.21. Результат повернення руки на 90º

Після цього треба здійснити переміщення по осям Y та Z. Положення по осі Z можна визначити по положенню з’єднувача “Elbow_joint” (рис. 12.22).

Рис. 12.22. Визначення положення по осі Z

Встановивши відповідне значення в позицію руки отримаємо позицію, яка потрібна (рис. 12.23).

Рис. 12.23. Переміщення по осі Z

Після цього треба змінити позицію по осі Y, враховуючи розміри руки (переміщення треба здійснити на величину меншу за половину довжини руки, яка при проектуванні визначалась по осі Z) (рис. 12.24).

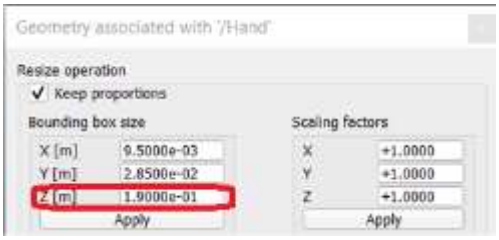


Рис. 12.24. Дані для визначення переміщення по осі Y

Встановивши вказані параметри переміщення (для довжини руки 0,19 м обрали зміщення на 0,075 м), отримаємо результат переміщення, що наведений на рисю 12.25.

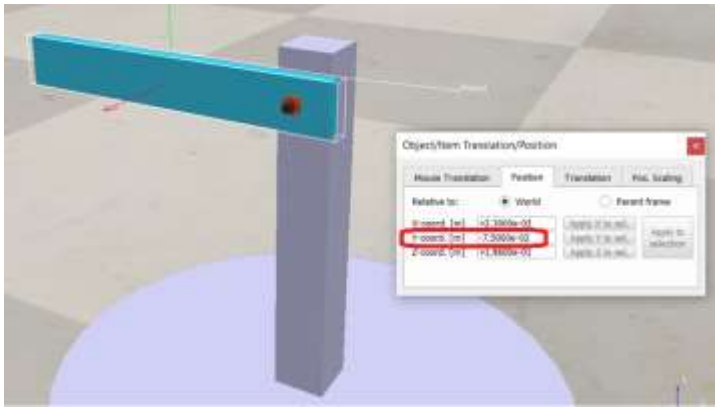


Рис. 12.25. Результат переміщення по осі Y

Симуляція переміщення усіх ланок маніпулятора буде мати вигляд, наведений на рис. 12.18, оскільки змінюється тільки вихідне положення руки маніпулятора.

Прискорення та уповільнення симуляції можна здійснити за допомогою інструментів, вказаних на рис. 12.26.

Рис. 12.26. Інструменти уповільнення та прискорення симуляції

Для зміни швидкості та напрямку обертання треба ввести відповідні дані у вікно динамічних властивостей з'єднувачів (рис. 12.27).

Рис. 12.27. Дані для зміни швидкості та напрямку обертання

Для зміни напрямку обертання треба замінити «+» на «-».

Можна здійснити керування профілем руху (Motion profile control), встановивши значення для таких параметрів, як прискорення (acceleration) та ривок (jerk).

При створенні відео симуляції руху мобільного робота можна встановити частоту кадрів (Frame rate) відповідно частоти кадрів зображення на екрані комп'ютера (рис. 12.28).

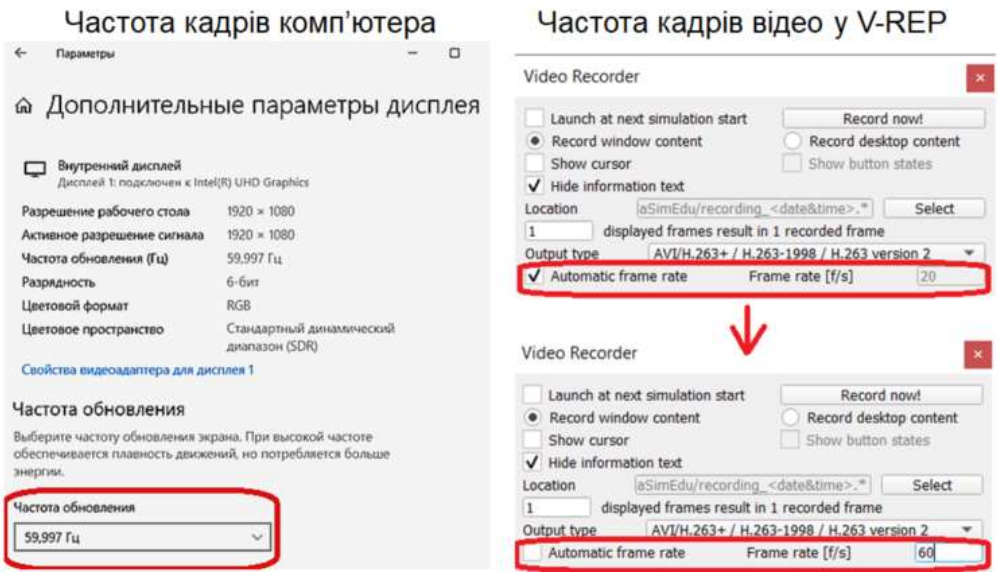


Рис. 12.28. Встановлення частоти кадрів зображення при створенні відео симуляції

Можна змінити колір окремих об’єктів сцени. Для цього треба виділити цей об’єкт та встановити бажаний колір (рис. 12.29).

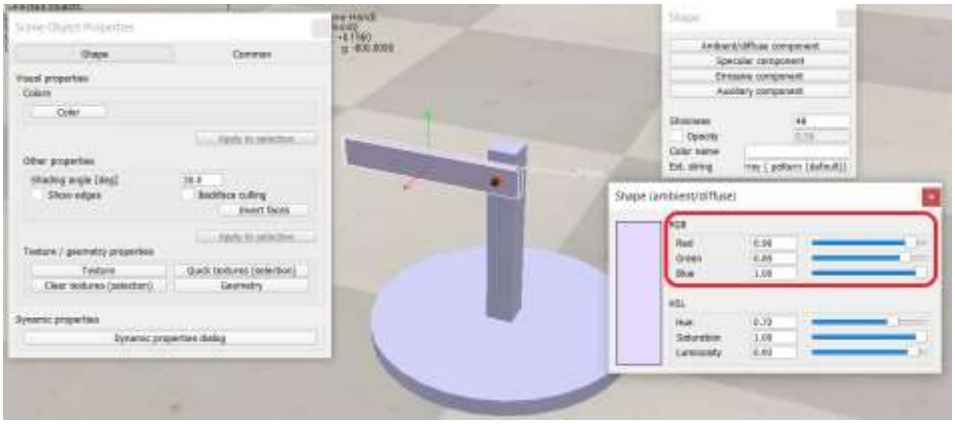


Рис. 12.29. Зміні кольору окремих об’єктів сцени

Контрольні запитання

1. Визначити, з яких ланок складається маніпулятор, для якого здійснюється конструювання.
2. Назвати, до якого типу об’єктів належить об’єкт «Циліндр».
3. Описати, як встановити розміри для об’єкту «Циліндр».
4. Визначити, як створити з’єднувач обертального типу.
5. Описати, як створити зв’язки між елементам.
6. Назвати, за допомогою яких інструментів можна змінити положення та орієнтацію об’єктів.
7. Визначити, як змінити швидкість та напрямок обертання з’єднувача обертального типу.
8. Назвати, як встановити частоту кадрів при створенні відео.
9. Описати, як здійснити прискорення та уповільнення симуляції.
10. Визначити, як змінити колір окремих об’єктів сцени.

13. Конструювання моделі виробничої ділянки з промисловим роботом та конвеєром

13.1. Моделювання переміщення ланок промислового робота

Розглянемо, як здійснюється моделювання переміщення ланок промислового робота. У каталозі моделей роботів виберемо для прикладу стаціонарний робот Dobot Magician (рис. 13.1).

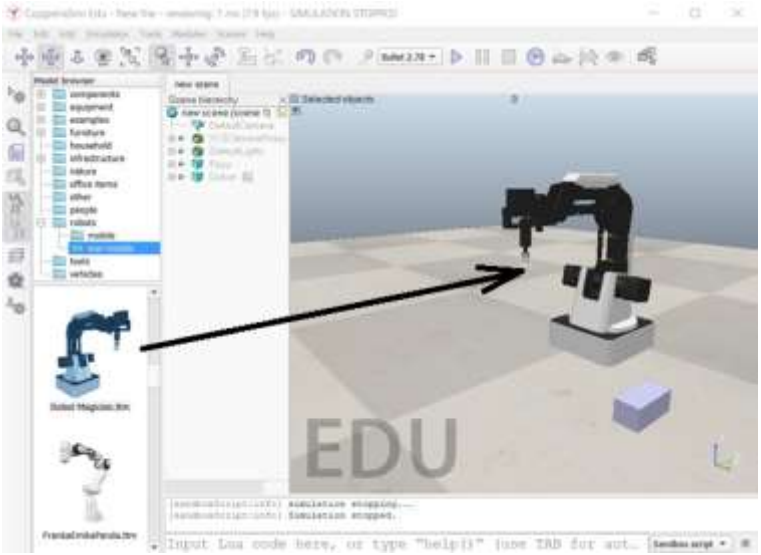


Рис. 13.1. Обираємо стаціонарний робот Dobot Magician

Запустивши симуляцію побачимо, що робот здійснює такі переміщення (рис.13.2).

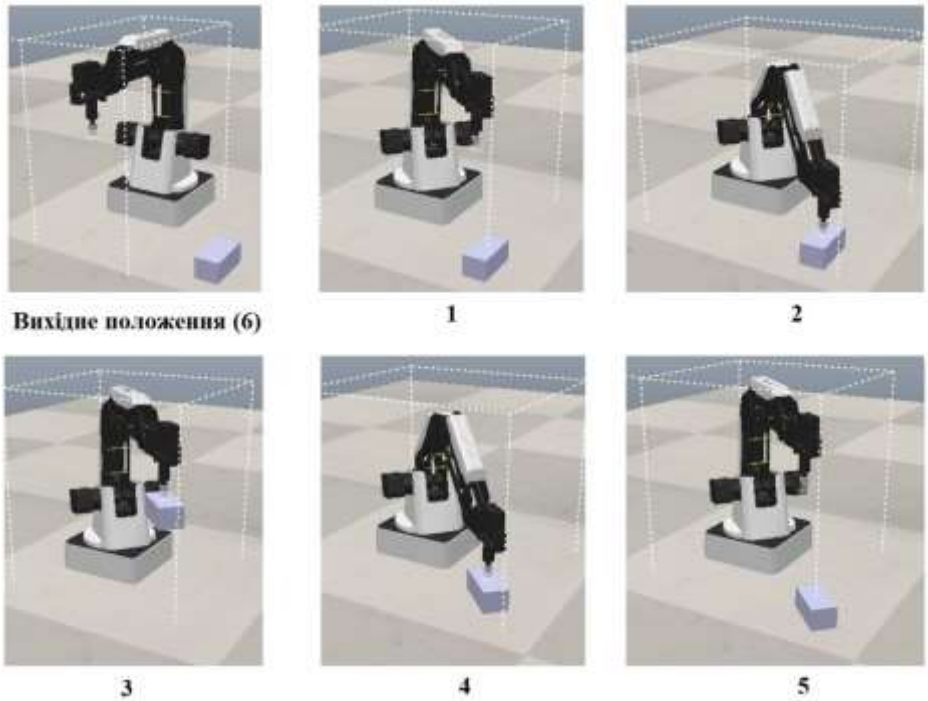


Рис. 13.2. Переміщення, які здійснює робот Dobot Magician під час симуляції

Розглянемо, як здійснюються ці переміщення у скрипті симуляції робота (Child script «/Dobot»). Цей скрипт, написаний за допомогою мови Lua, який має такий вигляд (рис. 13.3).

Child script "/Dobot"

```
function sysCall_init()
    corout=coroutine.create(coroutineMain)
end

function sysCall_actuation()
    if coroutine.status(corout)~='dead' then
        local ok,errorMsg=coroutine.resume(corout)
        if errorMsg then
            error(debug.traceback(corout,errorMsg),2)
        end
    else
        corout=coroutine.create(coroutineMain)
    end
end

function movCallback(config,vel,accel,handles)
    for i=1,#handles,1 do
        if sim.isDynamicallyEnabled(handles[i]) then
            sim.setJointTargetPosition(handles[i],config[i])
        else
            sim.setJointPosition(handles[i],config[i])
        end
    end
end

function moveToConfig(handles,maxVel,maxAccel,maxJerk,targetConf,enable)
    local currentConf={}
    for i=1,#handles,1 do
        currentConf[i]=sim.getJointPosition(handles[i])
        targetConf[i]=targetConf[i]*math.pi/180
    end
    sim.moveToConfig(-1,currentConf,nil,nil,maxVel,maxAccel,maxJerk,targetConf,nil,movCallback,handles)

    if enable then
        sim.writeCustomDataBlock(gripperHandle,'activity','on')
    else
        sim.writeCustomDataBlock(gripperHandle,'activity','off')
    end
end

function coroutineMain()
    modelBase=sim.getObject('.')
    gripperHandle=sim.getObject('./customizationScript')
    motorHandles = {}
    for i=1,4,1 do
        motorHandles[i]=sim.getObject('./motor'..i)
    end
    auxMotor1=sim.getObject('./auxMotor1')
    auxMotor2=sim.getObject('./auxMotor2')
    local vel=22
    local accel=40
    local jerk=80
    local maxVel={vel*math.pi/180,vel*math.pi/180,vel*math.pi/180,vel*math.pi/180}
    local maxAccel={accel*math.pi/180,accel*math.pi/180,accel*math.pi/180,accel*math.pi/180}
    local maxJerk={jerk*math.pi/180,jerk*math.pi/180,jerk*math.pi/180,jerk*math.pi/180}

    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,0,0,45},false)
    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,50,47,0},true)
    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,0,0,45},true)
    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,50,47,45},false
    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,0,0,45},false)
    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{0,0,0,0},false)
end

function sysCall_joint(inData)
    if inData.handle==auxMotor1 then
        local t2=sim.getJointPosition(motorHandles[2])
        local t3=sim.getJointPosition(motorHandles[3])
        error=t3-t2-inData.pos
    end
    if inData.handle==auxMotor2 then
        local t3=sim.getJointPosition(motorHandles[3])
        error=-t3-inData.pos
    end

    local ctrl=error*20
    local maxVelocity=ctrl
    if (maxVelocity>inData.maxVel) then
        maxVelocity=inData.maxVel
    end
    if (maxVelocity<-inData.maxVel) then
        maxVelocity=-inData.maxVel
    end
    local forceOrTorqueToApply=inData.maxForce

    local outData={vel=maxVelocity,force=forceOrTorqueToApply}
    return outData
end
```

Рис. 13.3. Child script «/Dobot»

Виділений фрагмент програми на рис. 13.3 є послідовністю переміщень ланок робота.

Керування здійснюється функцією API sim.moveToConfig.

На рис. 13.4 показано, які елементи конфігурації функції moveToConfig здійснюють переміщення та керування окремих ланок

Рис. 13.4. Переміщення та керування окремих ланок у скрипті

Наведений на рис. 13.4 фрагмент скрипта здійснює переміщення, що наведено на рис. 13.2. Значення переміщень вказуються у градусах відносно вихідного значення (0). Напрямок переміщення встановлює знак (+) або (-). Відсутність знаку означає +.

Послідовність переміщень:

поворот робота 50º та присоски 45º, присоска викл.;

```
moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,0,0,45},false)
```

поворот робота 50º, нахил 50º, згинання 47º, та присоски 0º, присоска вкл.;

```
moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,50,47,0},true)
```

поворот робота 50º, нахил 0º, згинання 0º, та присоски 45º, присоска вкл.;

```
moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,0,0,45},true)
```

поворот робота 50º, нахил 50º, згинання 47º, та присоски 45º, присоска викл.;

```
moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,50,47,45},false)
```

поворот робота 50º, нахил 0º, згинання 0º, та присоски 45º, присоска викл.;

```
moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,0,0,45},false)
```

поворот робота 0º, нахил 0º, згинання 0º, та присоски 0º, присоска викл.;

```
moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{0,0,0,0},false)
```

Шляхом змінення параметрів та додавання або викидання конфігурацій функції moveToConfig можна змінити переміщення робота.

Наведений фрагмент скрипта:

```
57    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,0,0,0},false)
58    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,50,47,0},true)
59    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,0,0,0},true)
60    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{-40,0,0,90},true)
61    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{-40,50,47,90},false)
62    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{-40,0,0,0},false)
63    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{-40,50,47,0},true)
64    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{-40,0,0,0},true)
65    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,0,0,-90},true)
66    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,50,47,-90},false)
67    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,0,0,0},false)
68    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{0,0,0,0},false)
```

здійснює переміщення, наведені на рис. 13.5.

Треба звернути увагу, що орієнтація вантажу при переміщенні не змінюється шляхом відповідного повороту захоплюючого пристрою

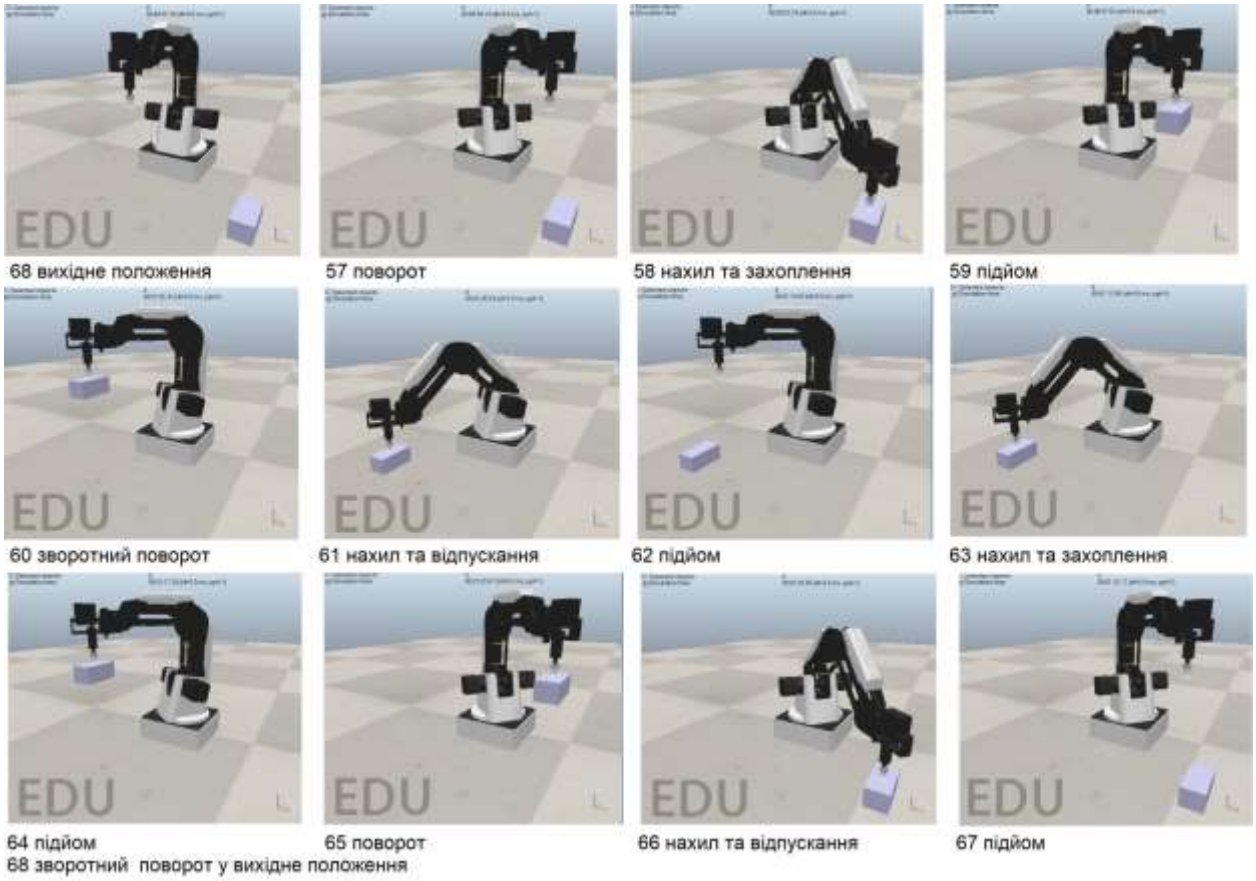


Рис. 13.5. Переміщення відповідно наведеному фрагменту скрипта

13.2. Конструювання моделі виробничої ділянки з двома роботами та конвеєрами

Розглянемо можливості конструювання робототехнічних комплексів (РТК) на прикладі виробничої ділянки, що складається з роботів та конвеєрів (рис. 13.6).

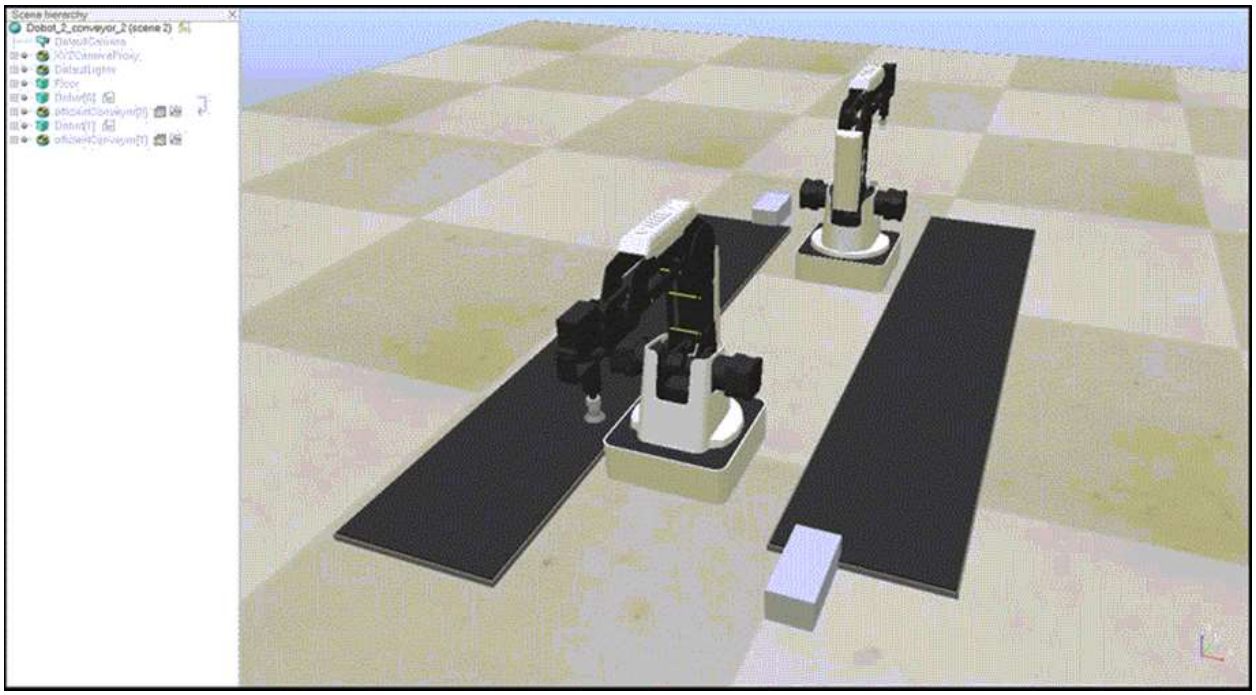


Рис. 13.6. Приклад виробничої ділянки, що складається з роботів та конвеєрів

За основу візмемо робот Dobot Magician, для якого була створена програма переміщення вантажу, додаймо до нього конвеєр generic conveyor (efficient) та встановимо для нього відповідні параметри положення (рис. 13.7).

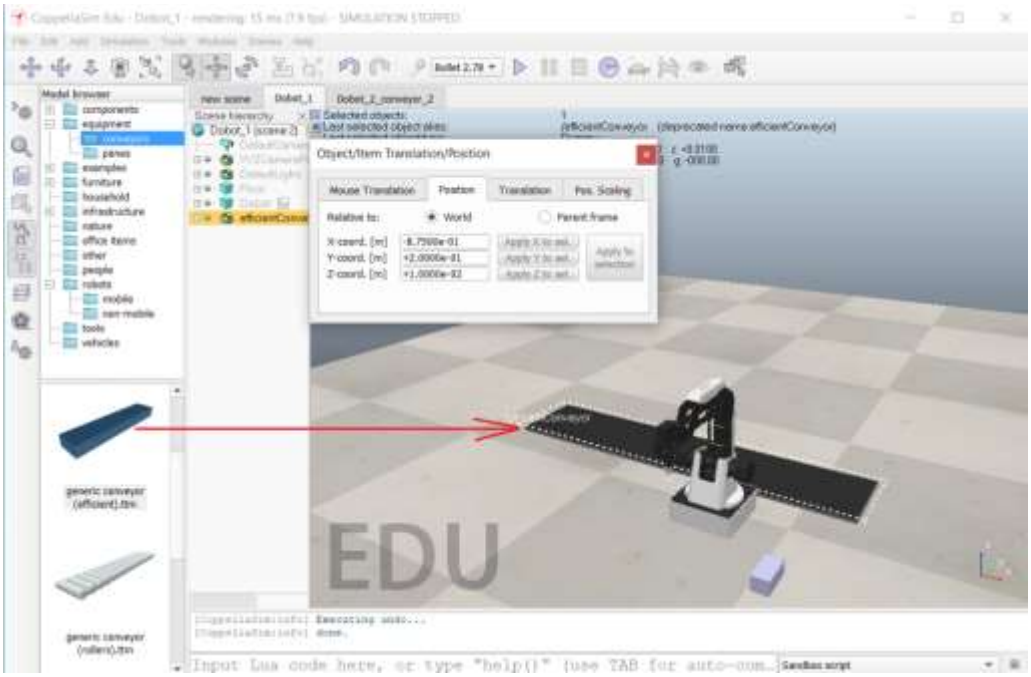


Рис. 13.7. Робот Dobot Magician з конвеєром

Для того, щоб вантаж опинився на конвеєрі, повернемо його на 90 º (рис. 13.8).

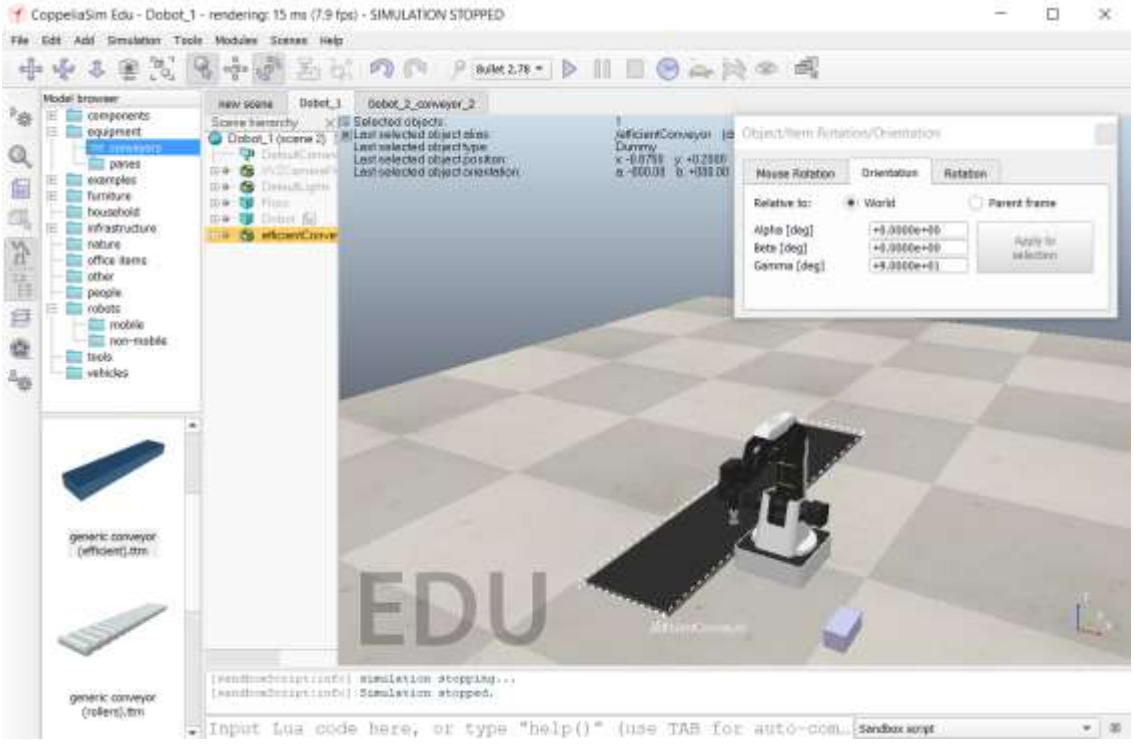


Рис. 13.8. Поворот конвеєра на 90 º

Раніше було показано, як здійснюється переміщення ланок робота без встановлення вантажу в інше місце.

Тому для того, щоб вантаж залишився на конвеєрі, зробимо наведені зміни у скрипті, наведеному на рис. 13.5.

Фрагмент скрипту до змін:

```
57    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,0,0,0},false)
58    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,50,47,0},true)
59    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,0,0,0},true)
60    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{-40,0,0,90},true)
61    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{-40,50,47,90},false)
62    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{-40,0,0,0},false)
63    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{-40,50,47,0},true)
64    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{-40,0,0,0},true)
65    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,0,0,-90},true)
66    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,50,47,-90},false)
67    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,0,0,0},false)
68    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{0,0,0,0},false)
```

Фрагмент скрипту після змін:

```
57    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,0,0,0},false)
58    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,50,47,0},true)
59    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,0,0,0},true)
60    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{-40,0,0,90},true)
61    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{-40,50,47,90},false)
62    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{-40,0,0,0},false)
63    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{-40,50,47,0},false)
64    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{-40,0,0,0},false)
65    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,0,0,-90},false)
66    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,50,47,-90},false)
67    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,0,0,0},false)
68    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{0,0,0,0},false)
```

В результаті отримаємо переміщення вантажу на конвеєр.

Додаймо у наведений проєкт виробничої ділянки з роботом та конвеєром ще один робот та конвеєр для постійного переміщення вантажів по колу (рис. 13.9).

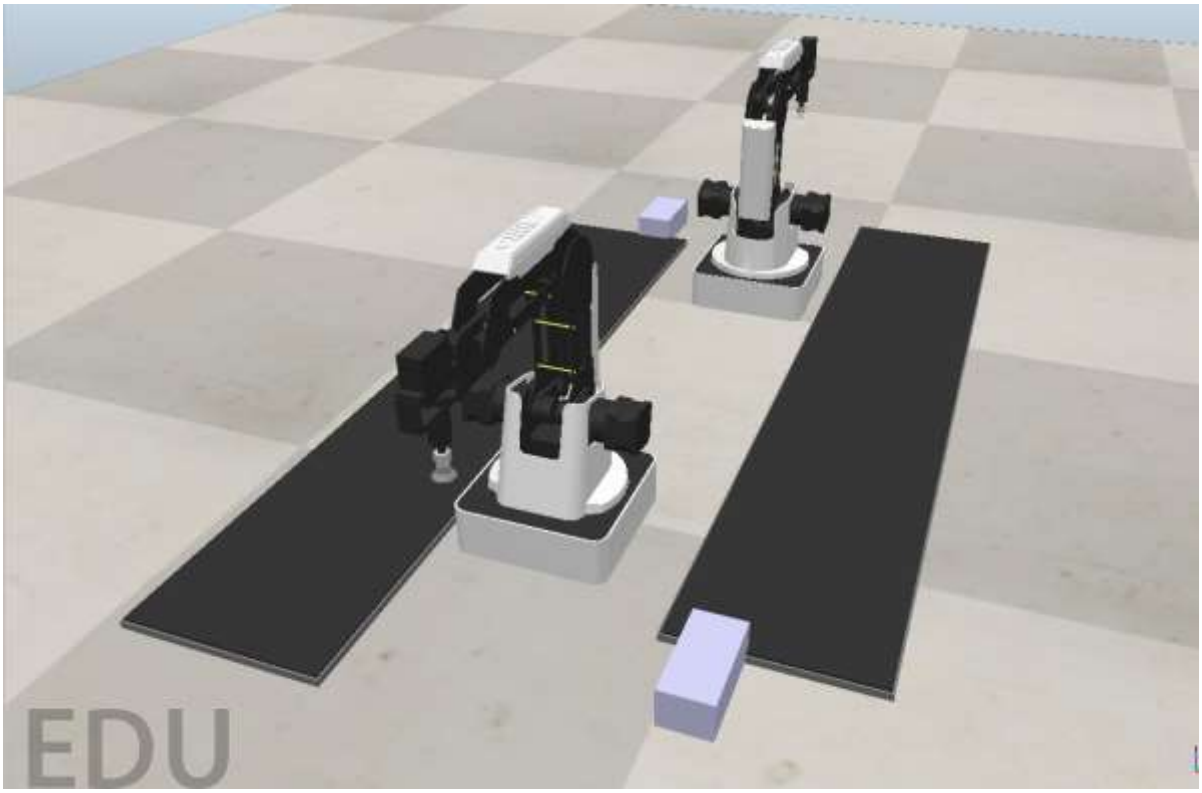


Рис. 13.9. Проєкт виробничої ділянки з двома роботами та конвеєрами

Встановлюємо позиції першого маніпулятора (рис. 13.10).

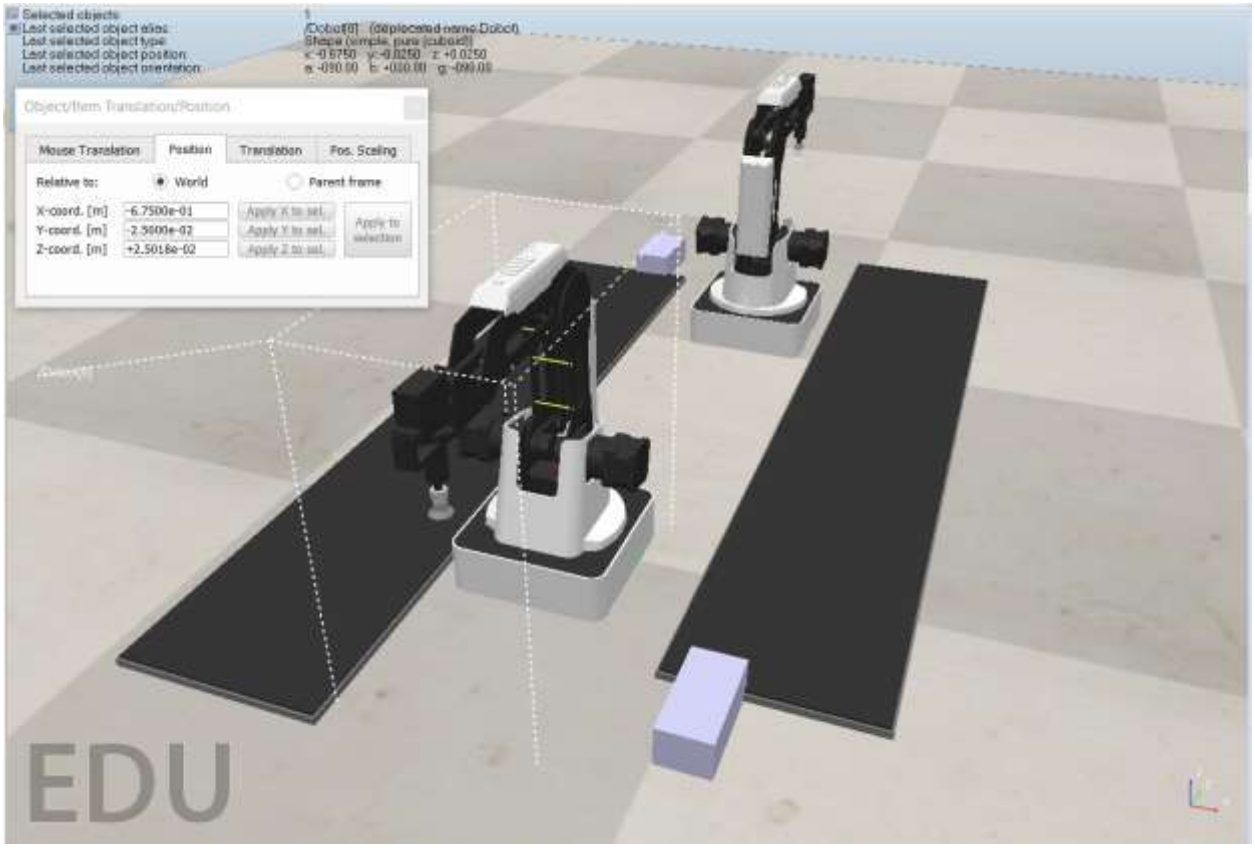


Рис. 13.10. Позиції першого маніпулятора

Встановлюємо позиції першого конвеєра (рис. 13.11).

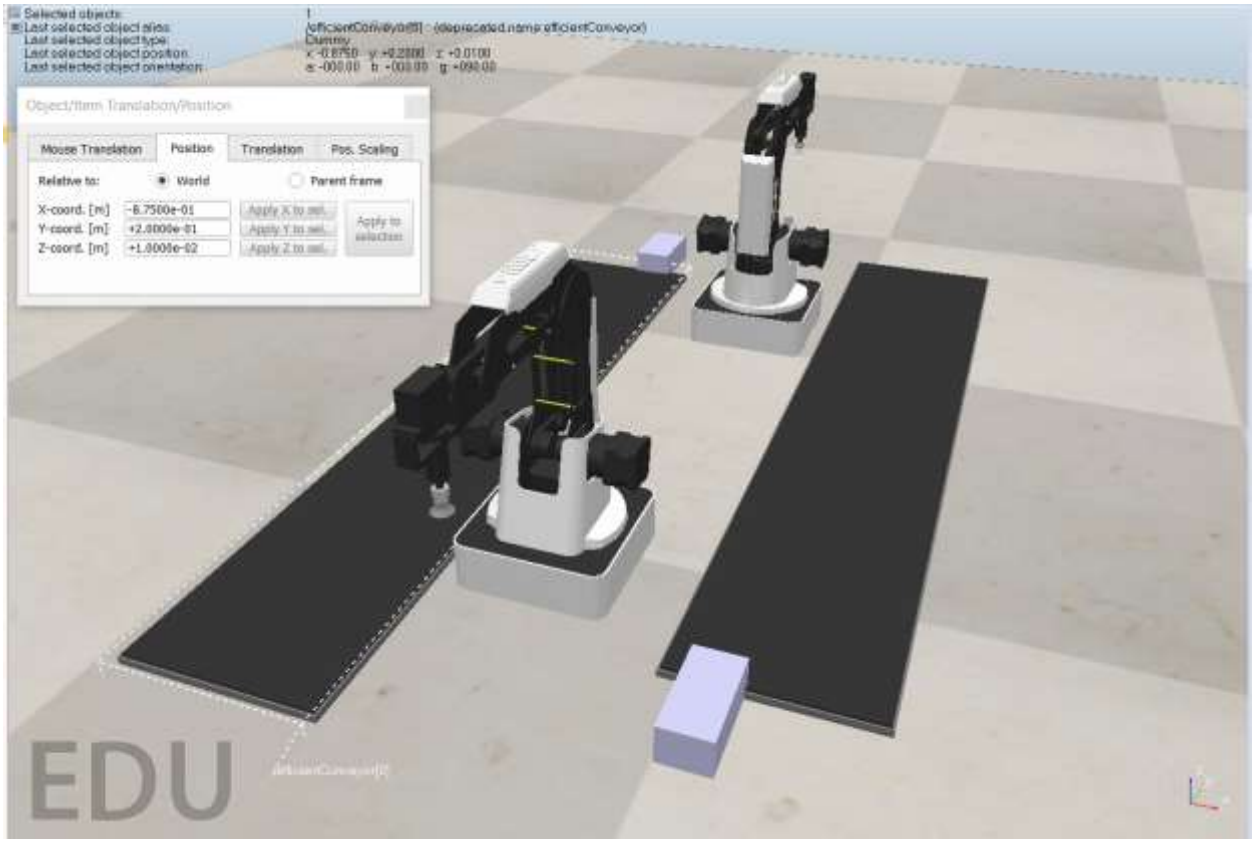


Рис. 13.11. Позиції першого конвеєра

Встановлюємо позиції другого маніпулятора (рис. 13.12).

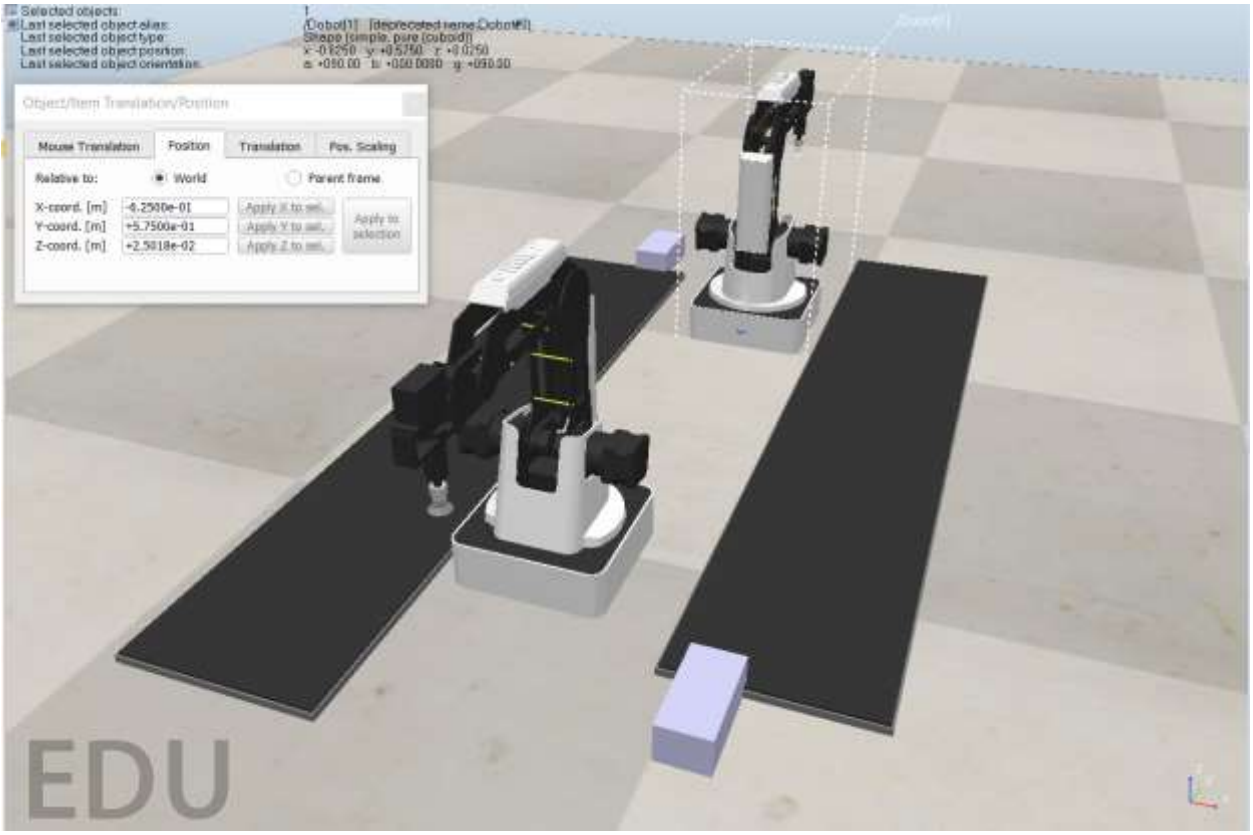


Рис. 13.12. Позиції другого маніпулятора

Встановлюємо позиції другого конвеєра (рис. 13.13).

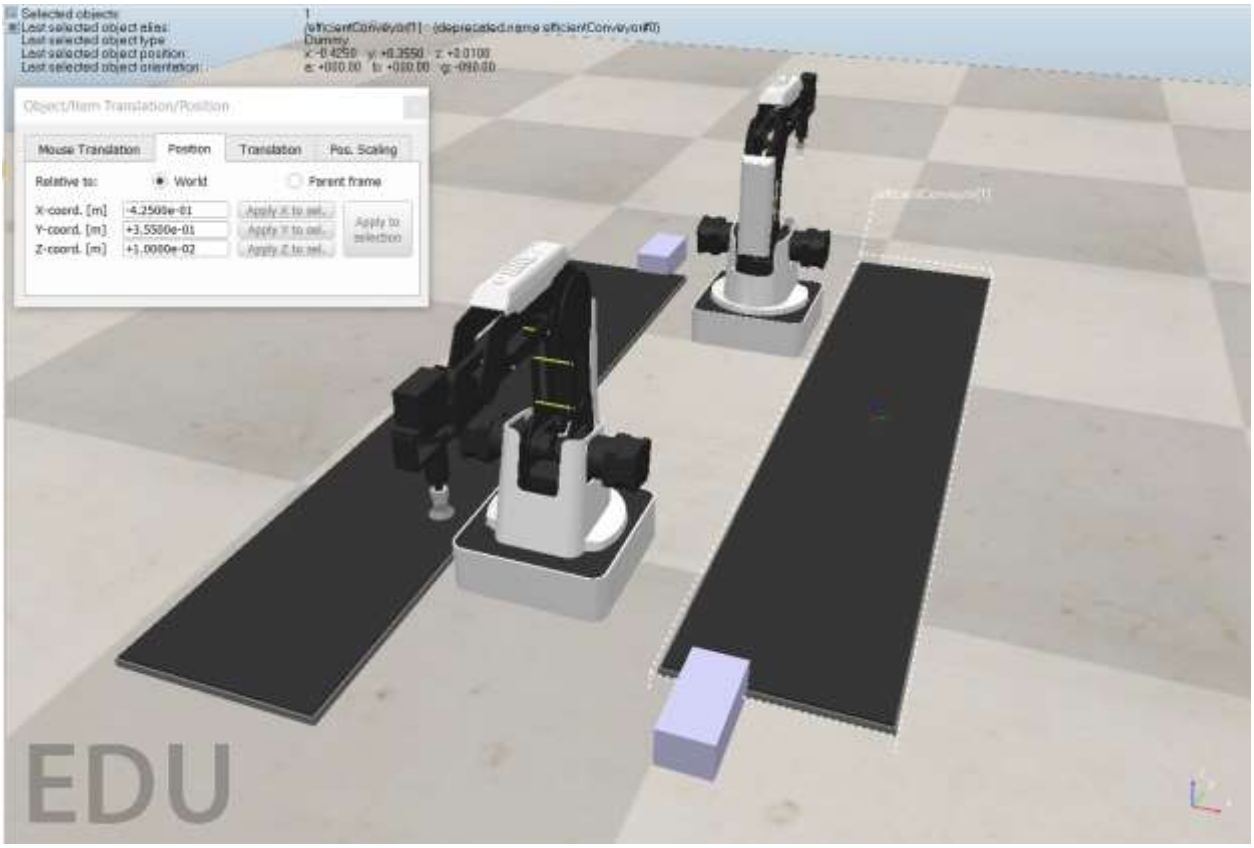


Рис. 13.13. Позиції другого конвеєра

На рис. 13.14 наведене взаємне положення елементів виробничої ділянки.

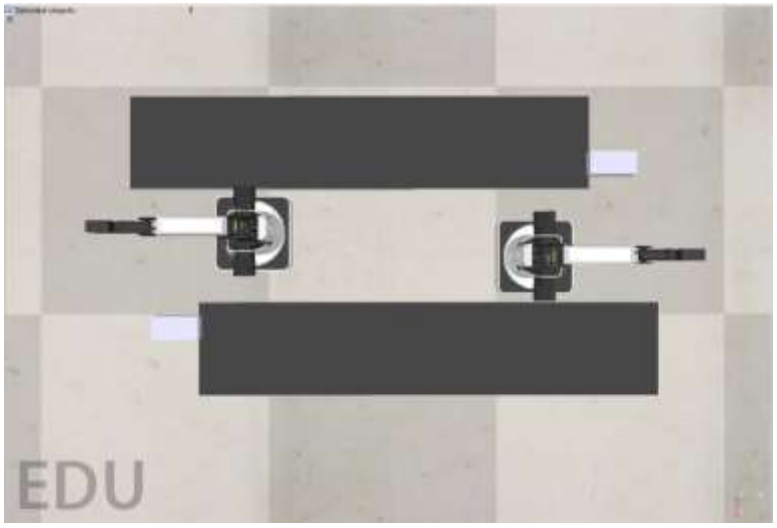


Рис. 13,14. Взаємне положення елементів виробничої ділянки

Для більш надійного захоплення вантажу можна збільшити його розміри (рис. 13.15).

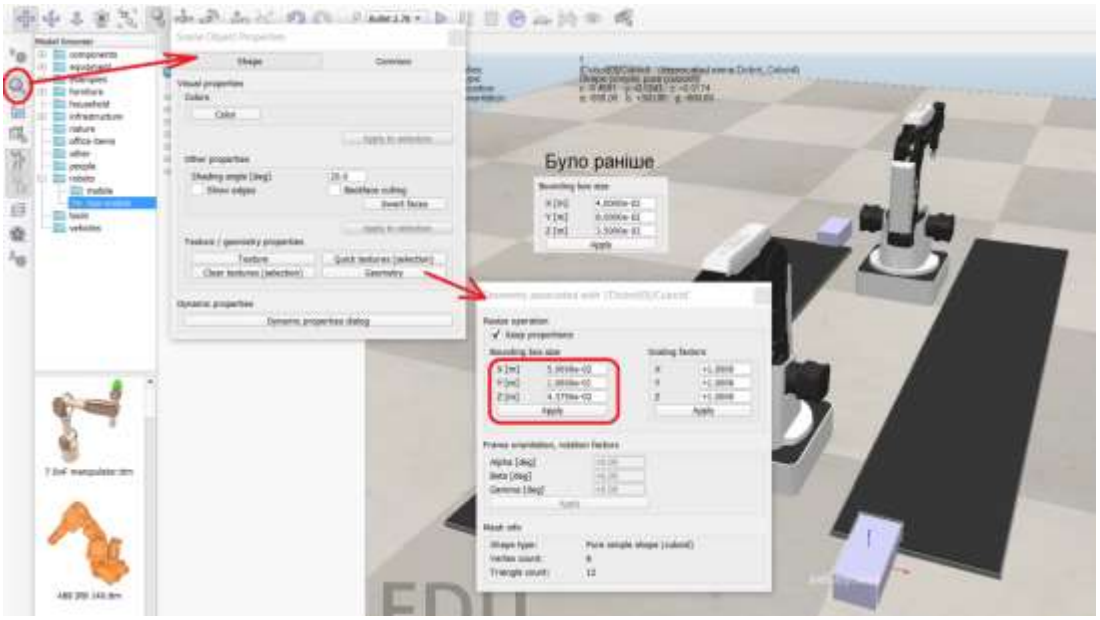


Рис. 13.14. Збільшення розміру вантажу

Щоб прибрати зайві рухи можна ввести такі зміни у скрипті.

```
57    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,0,0,0},false)
58    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,50,47,0},true)
59    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,0,0,0},true)
60    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{-40,0,0,90},true)
61    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{-40,50,47,90},false)
62    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{-40,0,0,0},false)
63    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{-40,50,47,0},false)
64    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{-40,0,0,0},false)
65    --moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,0,0,-90},false)
66    --moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,50,47,-90},false)
67    --moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,0,0,0},false)
68    moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{0,0,0,0},false)
```

Символи -- означають, що далі знаходиться коментар, а ці рухи вже не виконуються (можна було просто їх убрати).

Контрольні запитання

1. Визначити, які переміщення здійснює модель робота Dobot Magician.
2. Назвати, яку мову використовує скрипт симуляції робота Dobot Magician.
3. Описати, які переміщення встановлює конфігурація функції moveToConfig.
4. Назвати, які переміщення здійснює доповнена модель робота Dobot Magician.
5. Назвати, як додати конвеєр у проєкт виробничої ділянки.
6. Описати, які зміни треба зробити у скрипті для того, щоб вантаж залишився на конвеєрі.
7. Визначити, з чого складається виробнича ділянка для постійного переміщення вантажів по колу.
8. Назвати, як встановити положення нових елементів.
9. Описати, як найпростіше змінити напрямок руху конвеєра.
10. Визначити, як збільшити розміри вантажу.

14. Моделі роботів з ручним та автоматичним керуванням

14.1. Моделі роботів та їх елементів з ручним керуванням

Попередні версії платформи CoppeliaSim та V-REP мають моделі роботів та окремих компонентів з ручним керуванням.

Ці моделі можна перенести у останню версію CoppeliaSim, але треба враховувати, що там є моделі з такою ж назвою, але без ручного керування.

Тому під час переносу треба змінювати назву моделі.

Так попередня версія CoppeliaSim має модель робота ABB IRB 140 з ручним керування, якого немає у останній версії.

Можливість створення ручного керування за допомогою слайдерів була розглянута на практичному занятті.

Розглянемо більш детальніше модель робота ABB IRB 140 та її можливості.

На рис. 14.1 наведена модель робота. Після запуску симуляції отримаємо модель, яка за допомогою пульта керування може здійснювати переміщення різних ланок.

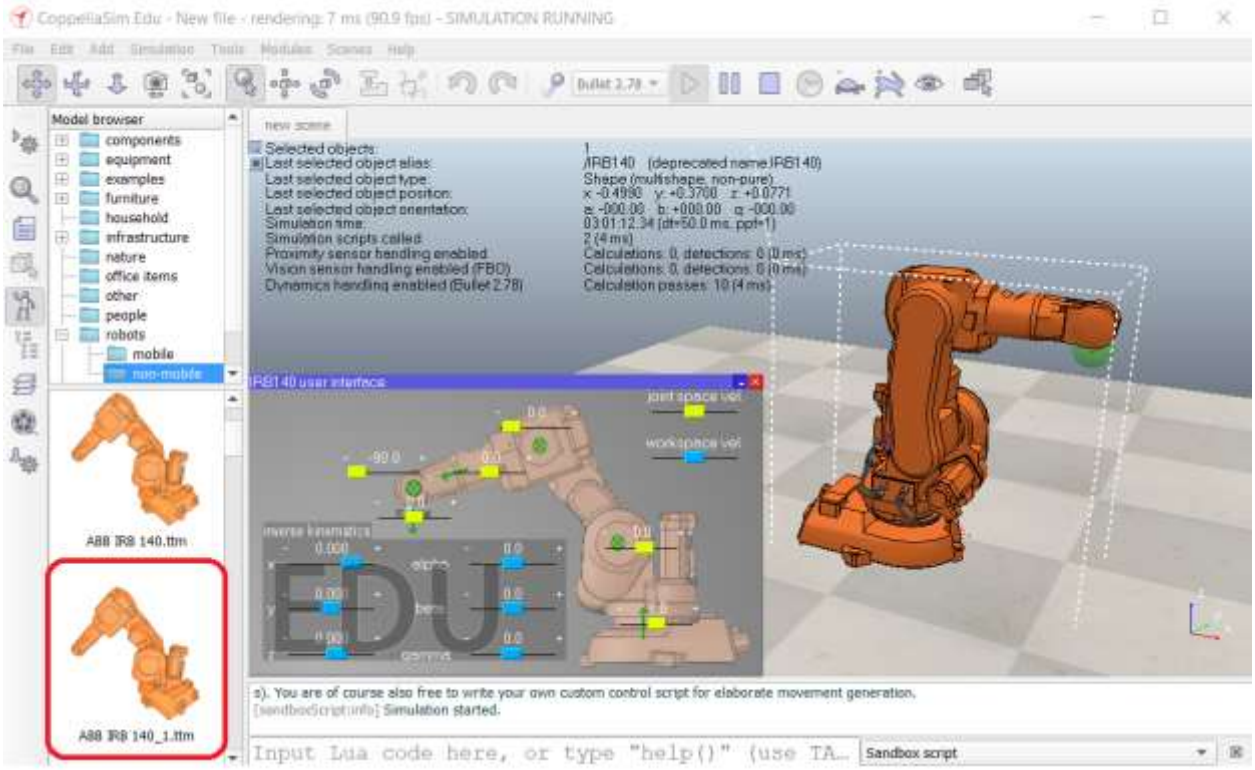


Рис. 14.1. Модель робота

Керування здійснюється за допомогою слайдерів (рис. 14.2).

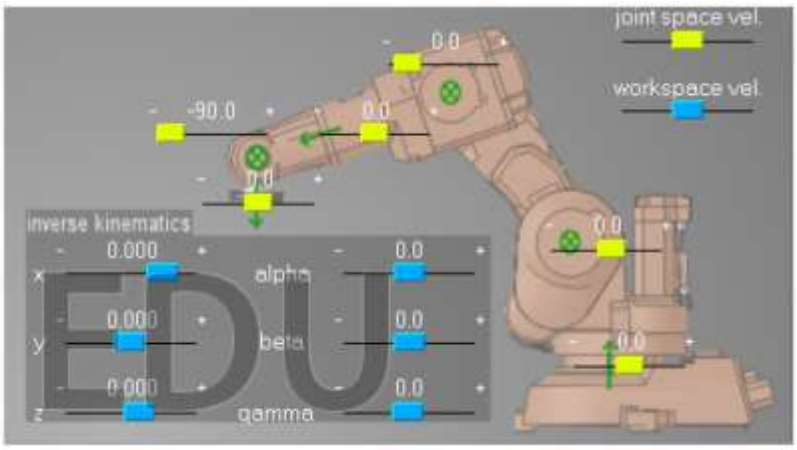


Рис. 14.2. Керування за допомогою слайдерів

На рис. 14.3 наведені слайдери переміщення робочого органу вздовж осей X, Y. Z.

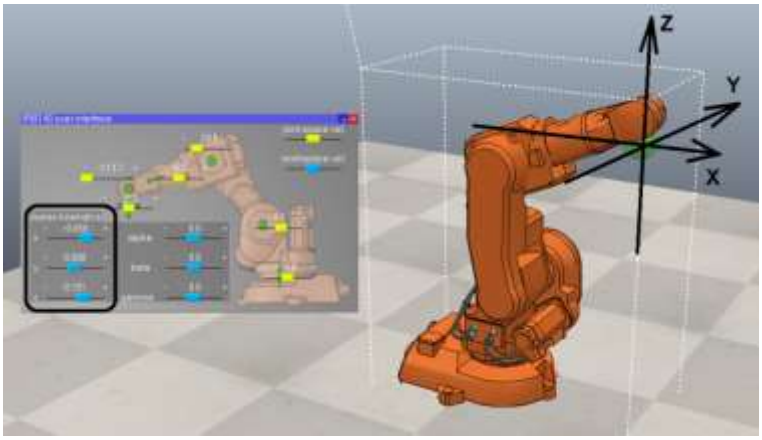


Рис. 14.3. Слайдери переміщення робочого органу

На рис. 14.4 наведені слайдери повороту робочого органу на кути alpha (α), beta (β), gamma (γ).

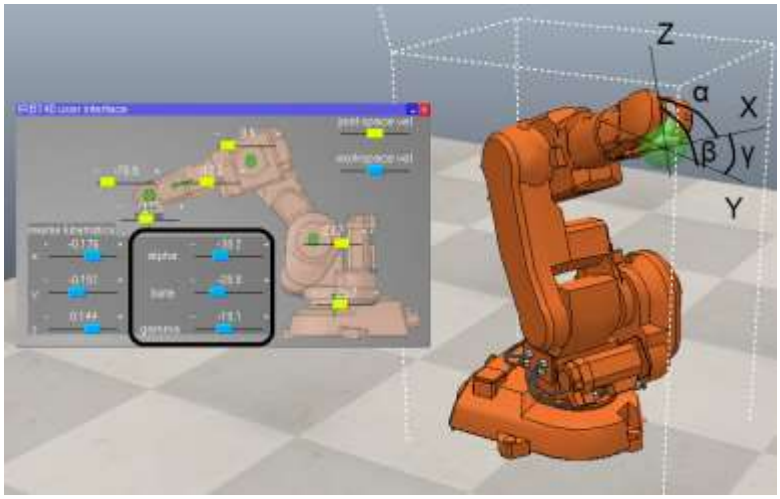


Рис. 14.4. Слайдери повороту робочого органу

На рис. 14.5 показані слайдери, які здійснюють переміщення ланок робота.

У моделі відповідні ланки робота мають такі позначення:

IRB140_link1, IRB140_link2, IRB140_link3, IRB140_link4, IRB140_link5, IRB140_link6

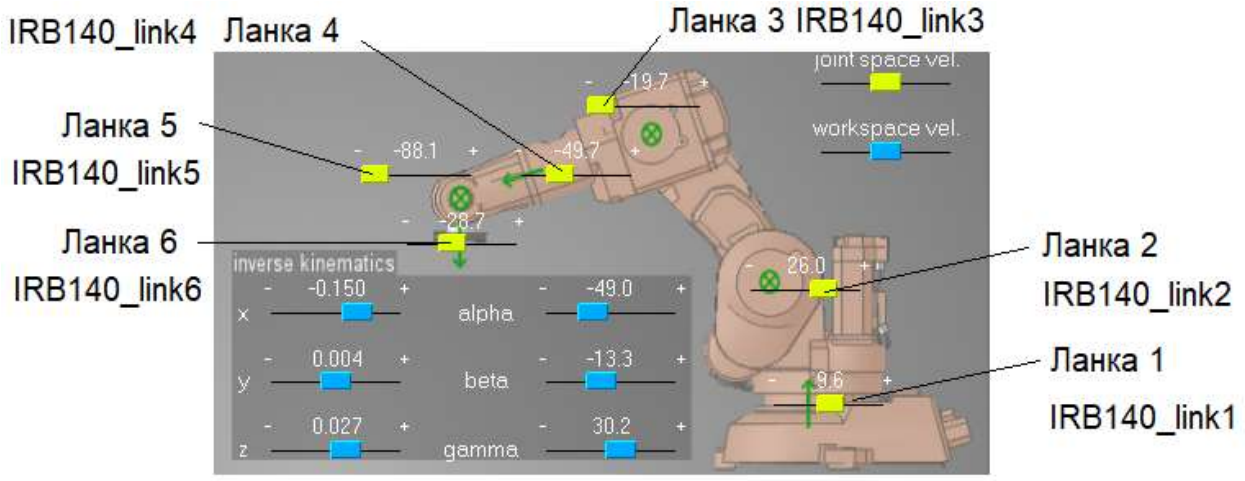


Рис. 14.5. Слайдери, які здійснюють переміщення ланок робота

Розглянемо моделі захоплюючих пристроїв.

Наведений на рис. 14.6 захоплюючий пристрій Barrett Hand (V-REP PRO EDU, Version 3.6.2.) при натисканні кнопки Close закривається, а при натисканні кнопки Open відкривається.

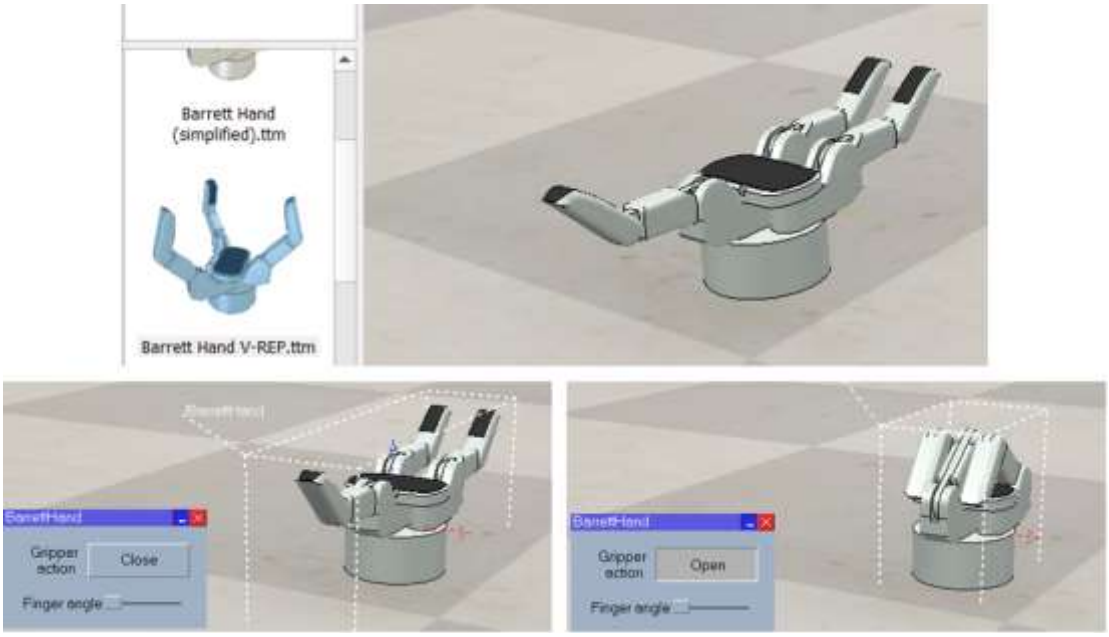


Рис. 14.6. Захоплюючий пристрій Barrett Hand

Аналогічно працює захоплюючий пристрій ROBOTIQ 85, наведений на рис. 14.7. (V-REP PRO EDU, Version 3.6.2.).

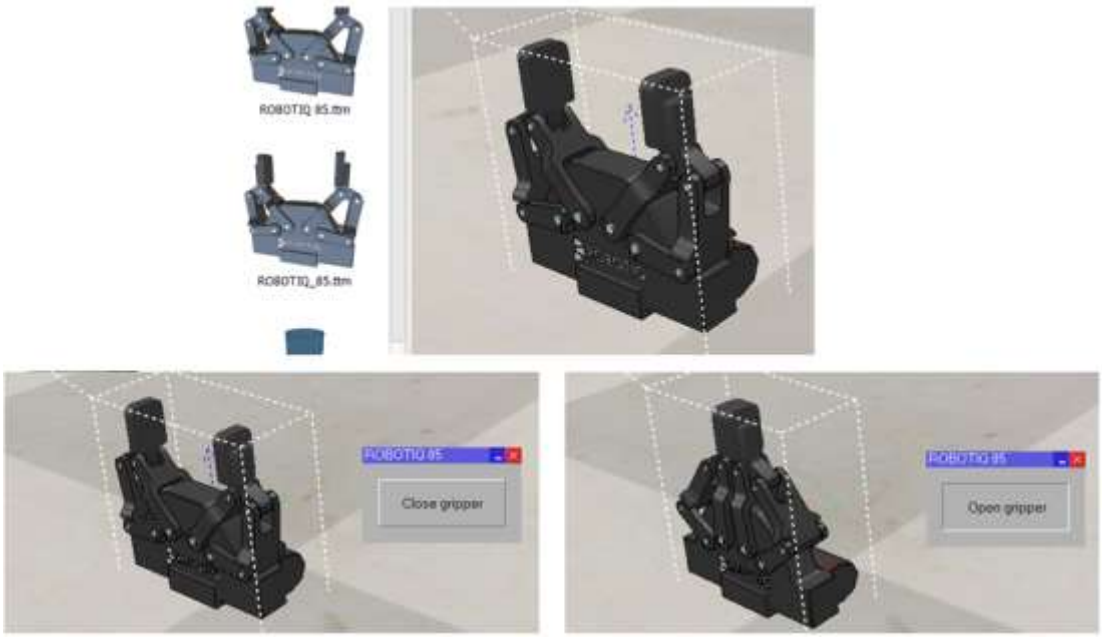


Рис. 14.7. Захоплюючий пристрій ROBOTIQ 85

Далі буде показано, як ці захоплюючи пристрої можна використовувати разом з маніпуляторами.

Дослідження можливостей мобільних роботів можна здійснити за допомогою моделі мобільного робота KUKA YouBot (V-REP PRO EDU, Version 3.6.2), наведеного на рис. 14.8..

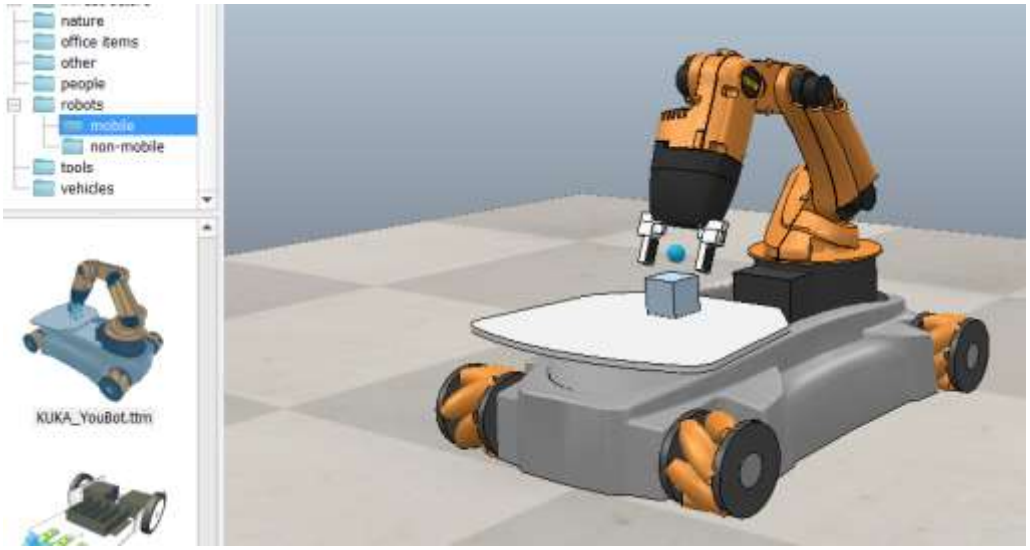


Рис. 14.8. Модель мобільного робота KUKA YouBot

На рис. 14.9 наведено вікно, за допомогою якого здійснюється симуляція ручного керування переміщень мобільного робота та встановленого на ньому маніпулятора

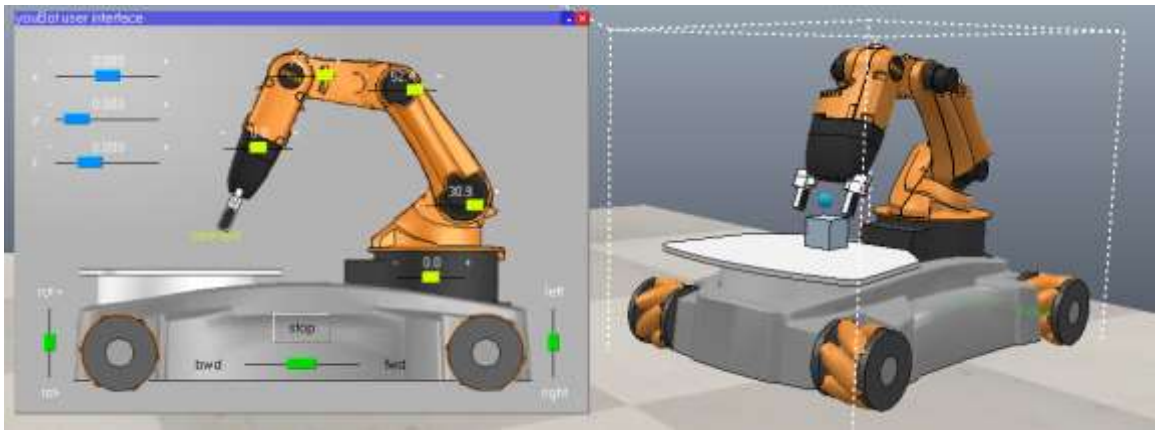


Рис. 14.9. Вікно, за допомогою якого здійснюється симуляція ручного керування

На рис. 14.10 наведено, як здійснюється керування рухом візка.

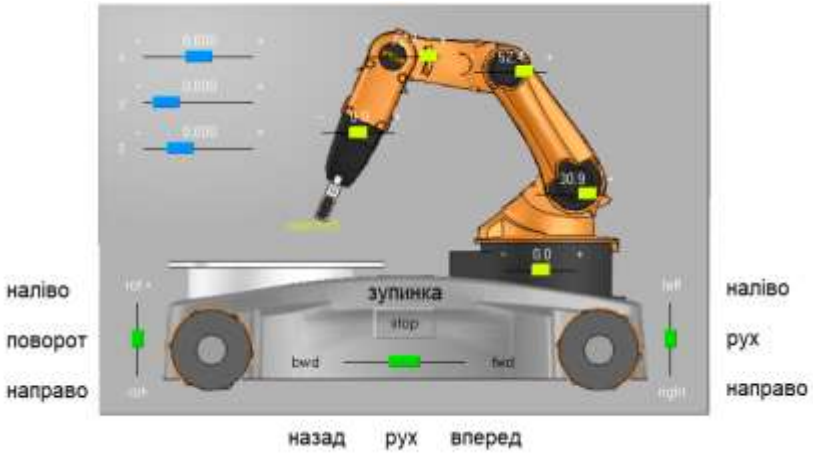


Рис. 14.10. Керування рухом візка

На рис. 14.11 наведені слайдери переміщення робочого органу вздовж осей X, Y. Z

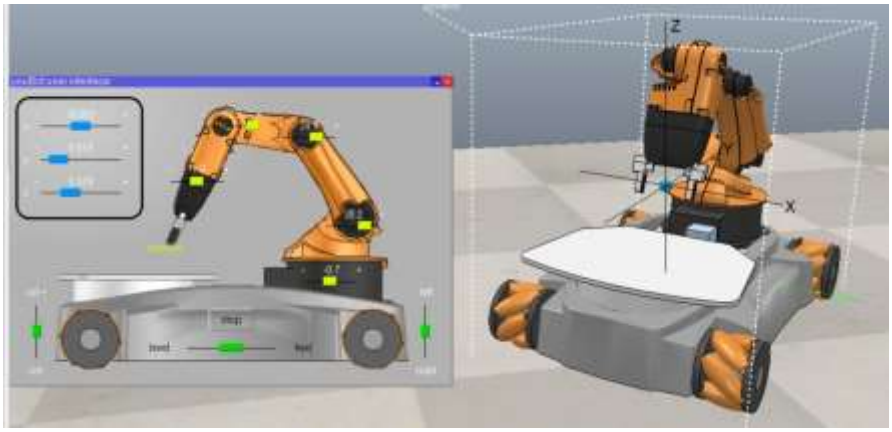


Рис. 14.11. Слайдери переміщення робочого органу вздовж осей

На рис. 14.12 наведені слайдери керування ланками маніпулятора.

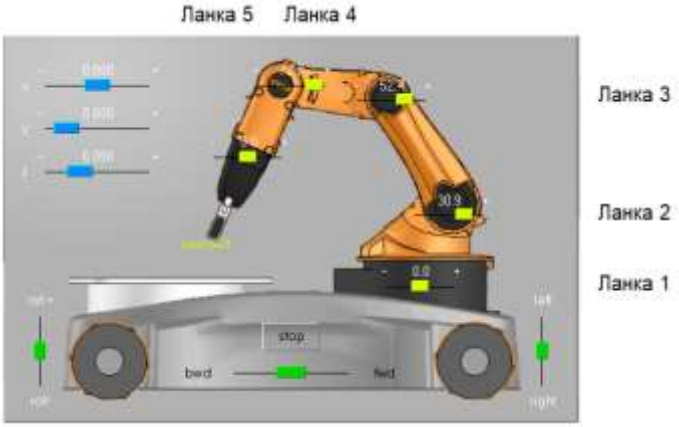


Рис. 14.12. Слайдери керування ланками маніпулятора

На рис. 14.13 наведено, як здійснюється Керування захоплюючим пристроєм.

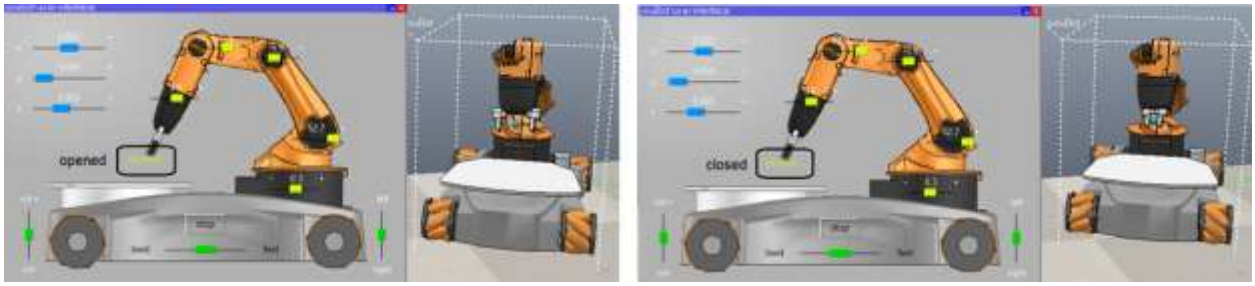


Рис. 14.13. Керування захоплюючим пристроєм

14.2. Моделі роботів та їх елементів з автоматичним керуванням

У попередньому розділі було розглянуто, як здійснюється керування переміщенням ланок маніпулятора на прикладі моделі робота Dobot Magician.

Визначено, що керування здійснюється за допомогою фрагмента скрипта, де встановлюються кути повороту окремих ланок та керування захоплюючим пристроєм (див. рис. 13.4).

```
moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,0,0,45},false)
moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,50,47,0},true)
moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,0,0,45},true)
moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,50,47,45},false)
moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{50,0,0,45},false)
moveToConfig(motorHandles,maxVel,maxAccel,maxJerk,{0,0,0,0},false)
```

На рис. 14.14 наведено переміщення відповідно наведеному фрагменту скрипта.

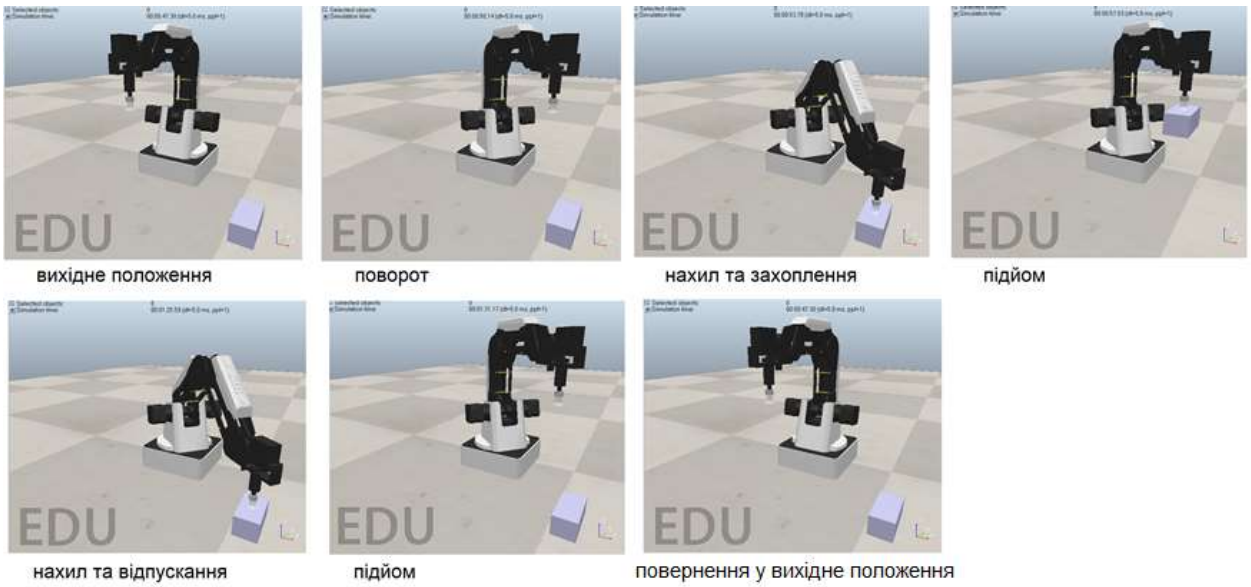


Рис. 14.14. Переміщення відповідно наведеному фрагменту скрипта

Розглянемо, як додаючи команди та змінюючи їх параметри здійснити зміну переміщення маніпулятора.

Так розглянута раніше модель мобільного робота KUKA YouBot у останній версії EduCoppeliaSim має лише такі переміщення:

setMovement(0,0.5,0) -- переміщення вправо
sim.wait(10) -- затримка10 с, встановлює термін переміщення
setMovement(0,0,0.5) -- поворот за годинниковою стрілкою

Після аналізу наведеного фрагменту була визначена відповідність параметрів з напрямком руху, а саме:

setMovement(0,0.5,0) sim.wait(10)	направо
setMovement(0.5,0,0) sim.wait(10)	вперед
setMovement(0,-0.5,0) sim.wait(10)	наліво
setMovement(-0.5,0,0) sim.wait(10)	назад
setMovement(0.5,0.5,0) sim.wait(10)	вперед - направо
setMovement(-0.5,-0.5,0) sim.wait(10)	назад - наліво
setMovement(-0.5,0.5,0) sim.wait(10)	назад - направо
setMovement(0.5,-0.5,0) sim.wait(10)	вперед - наліво
setMovement(0,0,0.5) sim.wait(20)	за годинниковою стрілкою
setMovement(0,0,-0.5) sim.wait(20)	проти годинникової стрілки
setMovement(0,0.0)	зупинка

Симуляція показує, що мобільний робот здійснює встановлені переміщення на протязі вказаного часу.

Аналогічне перетворення послідовності переміщень можна зробити для мобільного робота Omnidirectional Platform (рис. 14.15).

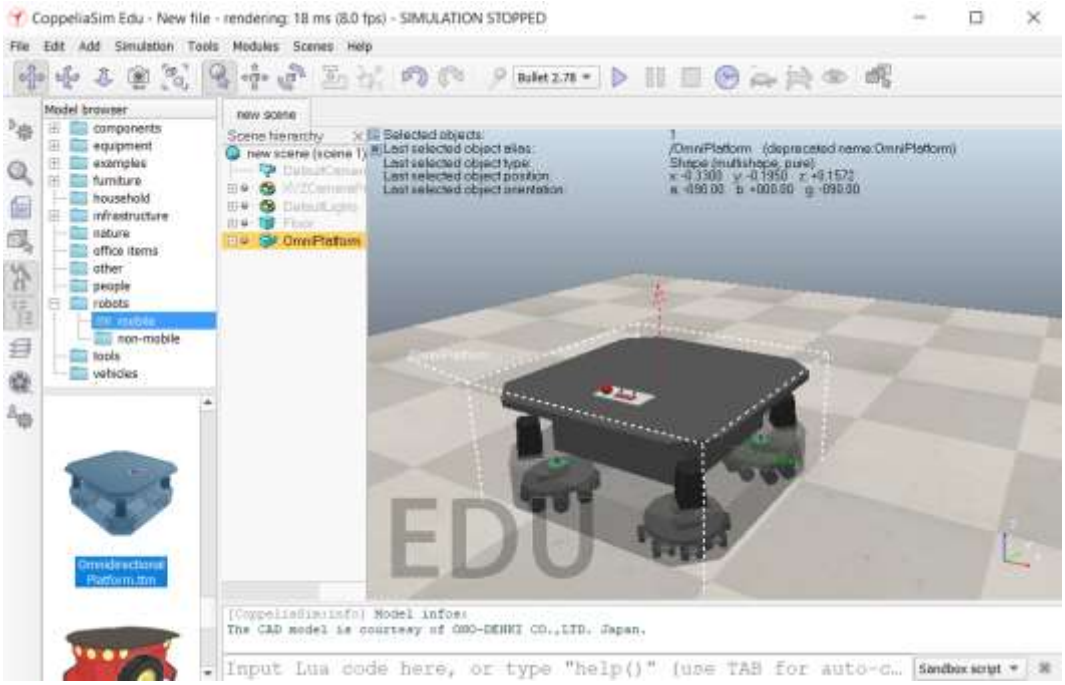


Рис. 14.15. Мобільний робот Omnidirectional Platform

Переміщення цього робота встановлює вказаний фрагмент скрипта, де показана відповідність параметрів з напрямком руху:

```
sim.setJointTargetVelocity(omniPads[1],-v) --forward
sim.setJointTargetVelocity(omniPads[2],-v)
sim.setJointTargetVelocity(omniPads[3],v)
sim.setJointTargetVelocity(omniPads[4],v)
sim.wait(5)
sim.setJointTargetVelocity(omniPads[1],v) --back
sim.setJointTargetVelocity(omniPads[2],v)
sim.setJointTargetVelocity(omniPads[3],-v)
sim.setJointTargetVelocity(omniPads[4],-v)
sim.wait(5)
sim.setJointTargetVelocity(omniPads[1],v)  --left
sim.setJointTargetVelocity(omniPads[2],-v)
sim.setJointTargetVelocity(omniPads[3],-v)
sim.setJointTargetVelocity(omniPads[4],v)
sim.wait(5)
sim.setJointTargetVelocity(omniPads[1],-v)  --right
sim.setJointTargetVelocity(omniPads[2],v)
sim.setJointTargetVelocity(omniPads[3],v)
sim.setJointTargetVelocity(omniPads[4],-v)
sim.wait(5)
sim.setJointTargetVelocity(omniPads[1],v)  --back left
sim.setJointTargetVelocity(omniPads[2],0)
sim.setJointTargetVelocity(omniPads[3],-v)
sim.setJointTargetVelocity(omniPads[4],0)
sim.wait(5)
sim.setJointTargetVelocity(omniPads[1],-v)  --forward right
sim.setJointTargetVelocity(omniPads[2],0)
sim.setJointTargetVelocity(omniPads[3],v)
sim.setJointTargetVelocity(omniPads[4],0)
sim.wait(5)
```

```
sim.setJointTargetVelocity(omniPads[1],0)  --forward left
sim.setJointTargetVelocity(omniPads[2],-v)
sim.setJointTargetVelocity(omniPads[3],0)
sim.setJointTargetVelocity(omniPads[4],v)
sim.wait(5)
sim.setJointTargetVelocity(omniPads[1],0)  --back right
sim.setJointTargetVelocity(omniPads[2],v)
sim.setJointTargetVelocity(omniPads[3],0)
sim.setJointTargetVelocity(omniPads[4],-v)
sim.wait(5))
sim.setJointTargetVelocity(omniPads[1],v)  --clockwise
sim.setJointTargetVelocity(omniPads[2],v)
sim.setJointTargetVelocity(omniPads[3],v)
sim.setJointTargetVelocity(omniPads[4],v)
sim.wait(4.825)
sim.setJointTargetVelocity(omniPads[1],-v)  --counterclockwise
sim.setJointTargetVelocity(omniPads[2],-v)
sim.setJointTargetVelocity(omniPads[3],-v)
sim.setJointTargetVelocity(omniPads[4],-v)
sim.wait(4.825)
sim.setJointTargetVelocity(omniPads[1],0)  --stop
sim.setJointTargetVelocity(omniPads[2],0)
sim.setJointTargetVelocity(omniPads[3],0)
sim.setJointTargetVelocity(omniPads[4],0)
```

Симуляція показує, що даний мобільний робот здійснює встановлені переміщення на протязі вказаного часу.

Контрольні запитання

1. Визначити, де знайти моделі роботів та додаткових компонентів, відсутніх у останній версії CoppeliaSim.
2. Назвати, які можливості дає модель робота ABB IRB 140.
3. Описати, які переміщення може здійснити модель робота ABB IRB 140.
4. Визначити, які захоплюючи пристрої дають можливість ручного керування.
5. Назвати, які можливості дає модель мобільного робота KUKA YouBot (V-REP PRO EDU, Version 3.6.2.
6. Описати, які переміщення може здійснити модель мобільного робота KUKA YouBot.
7. Визначити, як здійснити керування роботом у автоматичному режимі?
8. Назвати, які переміщення має модель мобільного робота KUKA YouBot у останній версії EduCoppeliaSim.
9. Описати, які переміщення має модель мобільного робота Omnidirectional Platform.
10. Визначити, для чого використовується функція sim.wait().

15. Використання моделей для конструювання та дослідження роботів

15.1. Конструювання роботів з використанням моделей

На рис. 15.1 наведена модель робота ABB IRB 140 з ручним керуванням, створення якої здійснювалось на практичному (лабораторному) занятті.

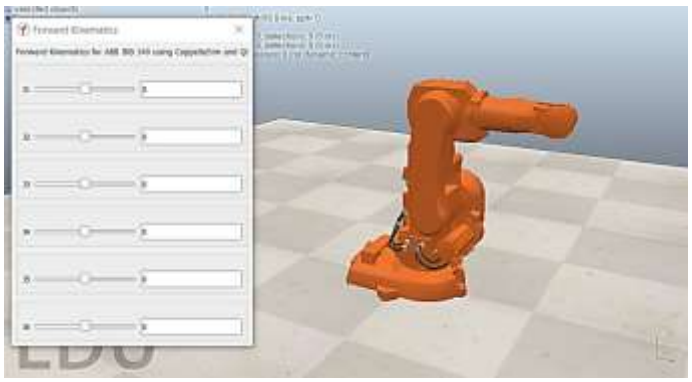


Рис. 15.1. Модель робота ABB IRB 140 з ручним керуванням

Розглянемо можливість встановлення на цей робот захоплюючого пристрою Barrett Hand.

Для цього треба додати на сцену захоплюючий пристрій Barrett Hand (V-REP PRO EDU, Version 3.6.2), як це показано на рис. 15.2.

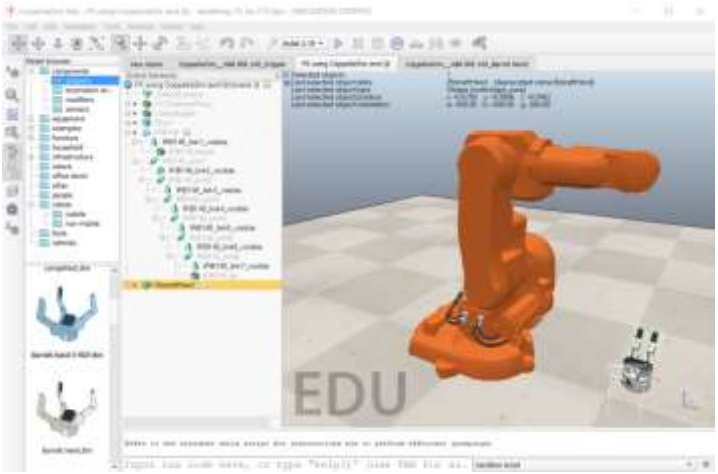


Рис. 15.2. Захоплюючий пристрій Barrett Hand

Після цього треба додати з'єднувач ForceSensor та за його допомогою приєднати захоплюючий пристрій до муфти робота (рис. 15.3).

Рис. 15.3. Приєднання захоплюючого пристрою до муфти робота

Також додаймо вантаж, який зможе захопити обраний захоплюючий пристрій (рис. 15.4).

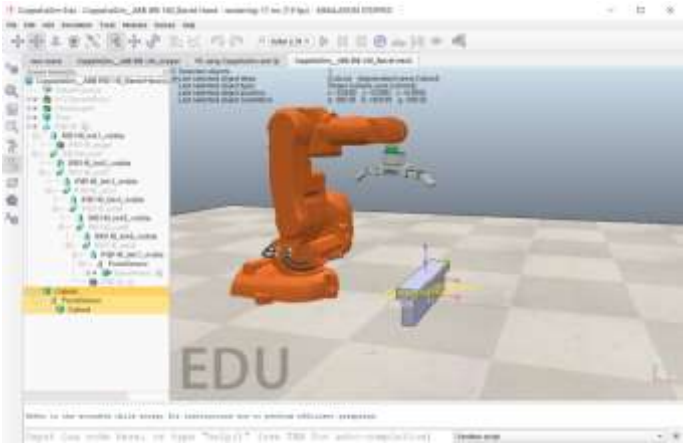


Рис. 15.4. Створення вантажу

Запустивши симуляцію отримаємо такий робот зі слайдерами для керування переміщенням ланок маніпулятора та кнопкою для керування захоплюючим пристроєм (рис. 15.5).

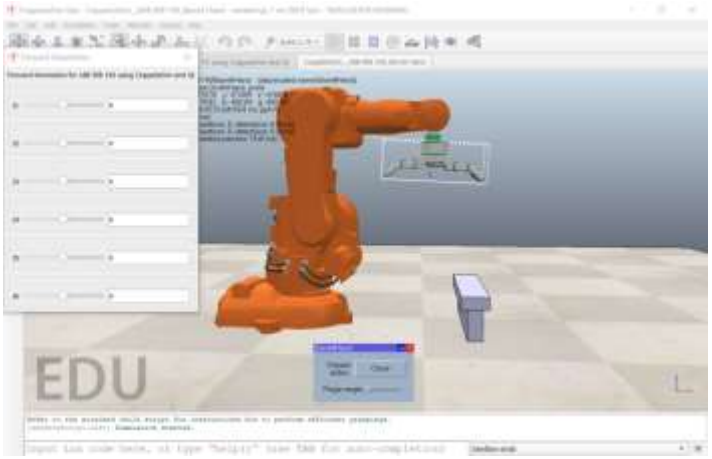


Рис. 15.5. Симуляція маніпулятора з засобами ручного керування

Шляхом поступового переміщення встановлюємо ланки маніпулятора для захоплення вантажу (рис. 15.6), та здійснюємо захоплення вантажу (рис. 15.7).

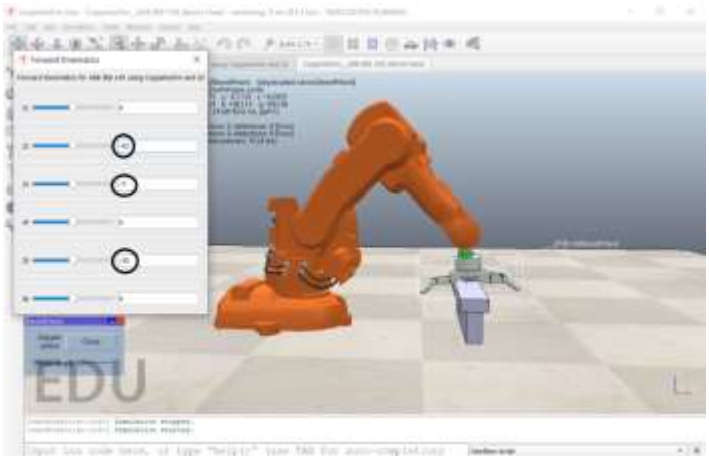


Рис. 15.6. Встановлення ланки маніпулятора для захоплення вантажу

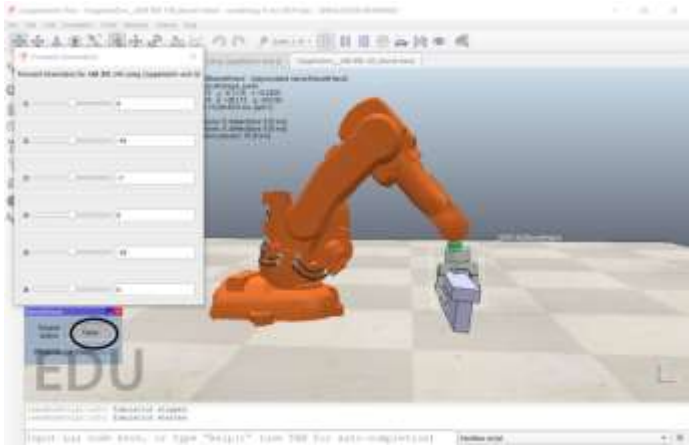


Рис. 15.7. Захоплення вантажу

Після цього можна здійснити переміщення вантажу та встановлення його у визначену позицію, наприклад, на конвеєр, який теж можна додати до цієї сцени (рис. 15.8).

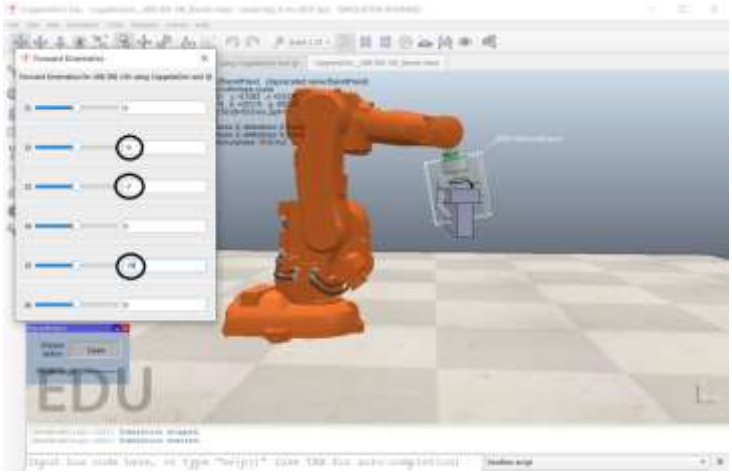


Рис. 15.8. Переміщення вантажу у визначену позицію

На попередніх лекціях було розглянуто конструювання моделі виробничої ділянки з промисловими роботами та конвеєрами (рис. 15.9).

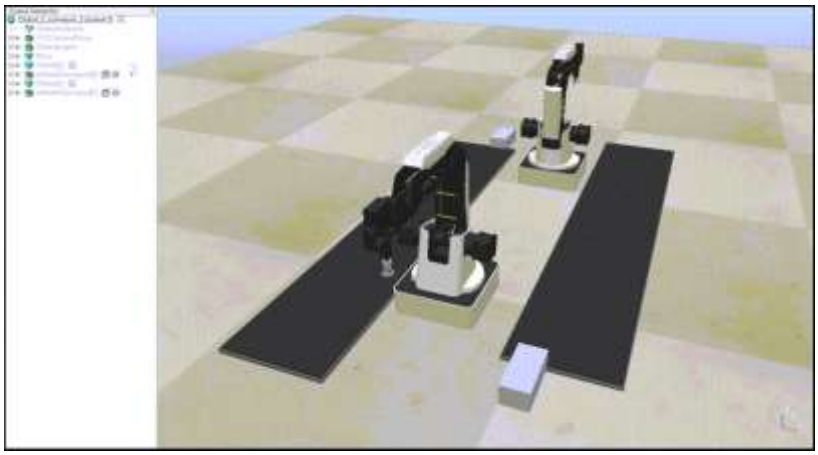


Рис. 15.9. Модель виробничої ділянки з промисловими роботами та конвеєрами

Для цього була застосована модель стаціонарного робота Dobot Magician, до якого була додана модель конвеєра generic conveyor (efficient) (рис. 15.10).

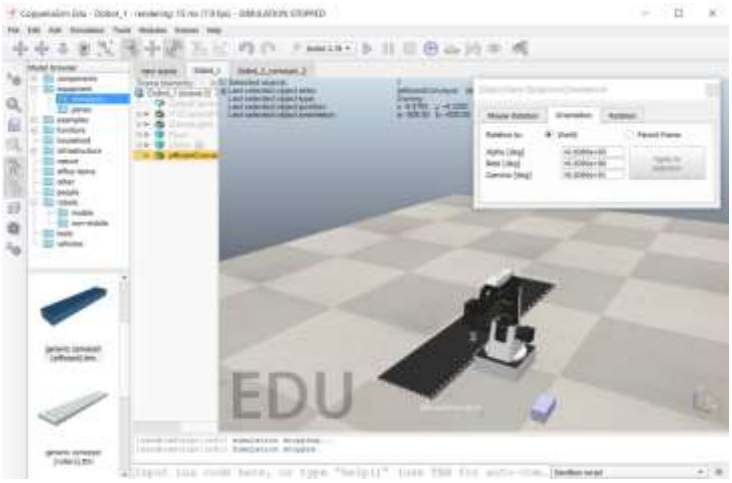


Рис. 15.10. Модель стаціонарного робота Dobot Magician з конвеєром generic conveyor (efficient)

Після копіювання та переміщення була створена така виробнича ділянка з двома роботами та двома конвеєрами для постійного переміщення вантажів по колу, що наведена на рис. 15.9.

15.2. Дослідження роботів з використанням моделей

Використовуючи розглянуті вище моделі можна зробити різні дослідження роботи робота, наприклад, вплив маси вантажу, яку можна змінювати, як це показано на рис. 15.11, швидкості обертаня)рис. 15.12) тощо.

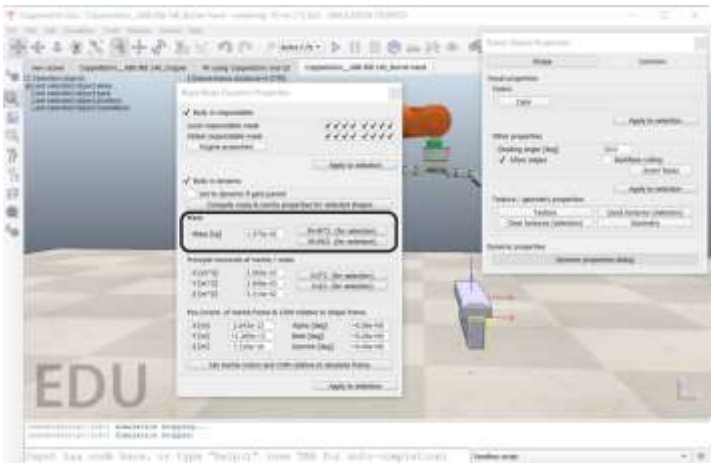


Рис. 15.11. Встановлення маси вантажу

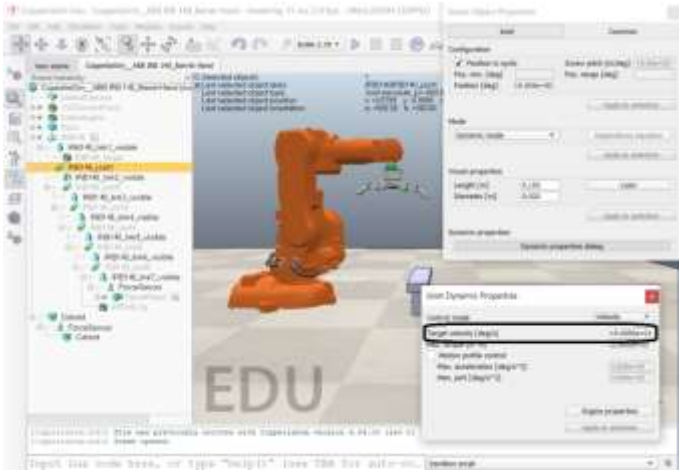


Рис. 15.12. Встановлення швидкості обертання для вказаної ланки

Результати робототехнічного дослідження повинні бути зведені в зручній наглядній формі. Однією з таких форм є графіки.

Розглянемо, як побудувати графік траєкторії руху на прикладі моделі мобільного робота **pioneer p3dx**, з використанням інструменту Add --> Graph.

На рис. 15.13. показано, як визначається швидкість обертання коліс робота у скрипті Child script "/PioneerP3DX".

```
40
41      vLeft=v0
42      vRight=v0
43
```

Рис. 15.13. Визначення швидкісті обертання коліс робота

Дослідження залежності траєкторії переміщення мобільного робота від швидкості обертання коліс можна здійснити шляхом встановлення швидкості обертання обох коліс за допомогою коефіцієнтів kL та kR.

vLeft=v0* kL

vRight=v0* kR

Дл створення графіку траєкторії руху мобільного робота pioneer p3dx на панелі меню треба обрати Add --> Graph.

Доступ основних властивостей цього інструменту та їх налаштування здійснюються у діалоговому вікні графіка.

Після внесення відповідних доповнень у скрипт отримаємо графік переміщення мобільного робота (рис. 15.14).

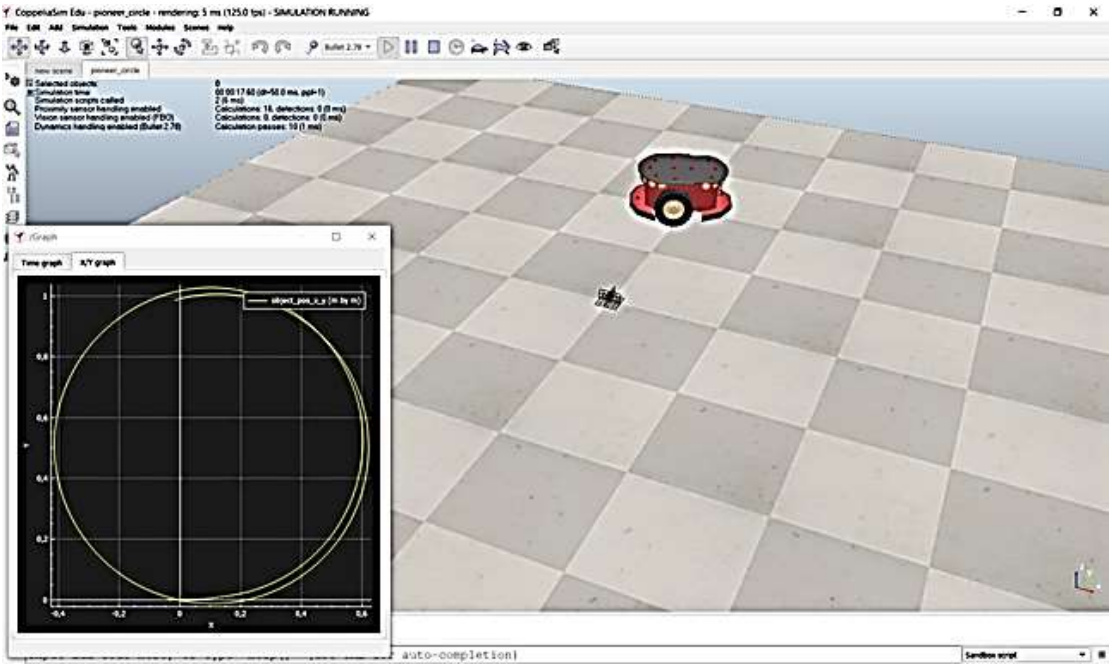


Рис. 15.14. Графік переміщення мобільного робота

Оскільки мобільний робот PioneerP3DX має ультразвукові датчики визначення перешкод, як це показано на рис. 15.15, є можливість здіснити дослідження переміщення мобільного робота з виявленням перешкод (рис. 15.16).

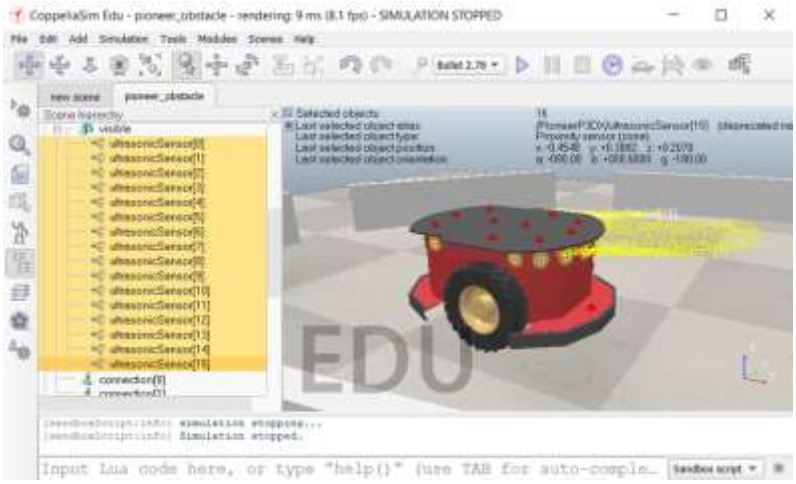


Рис. 15.15. Ультразвукові датчики визначення перешкод у мобільного робота **pioneer p3dx**

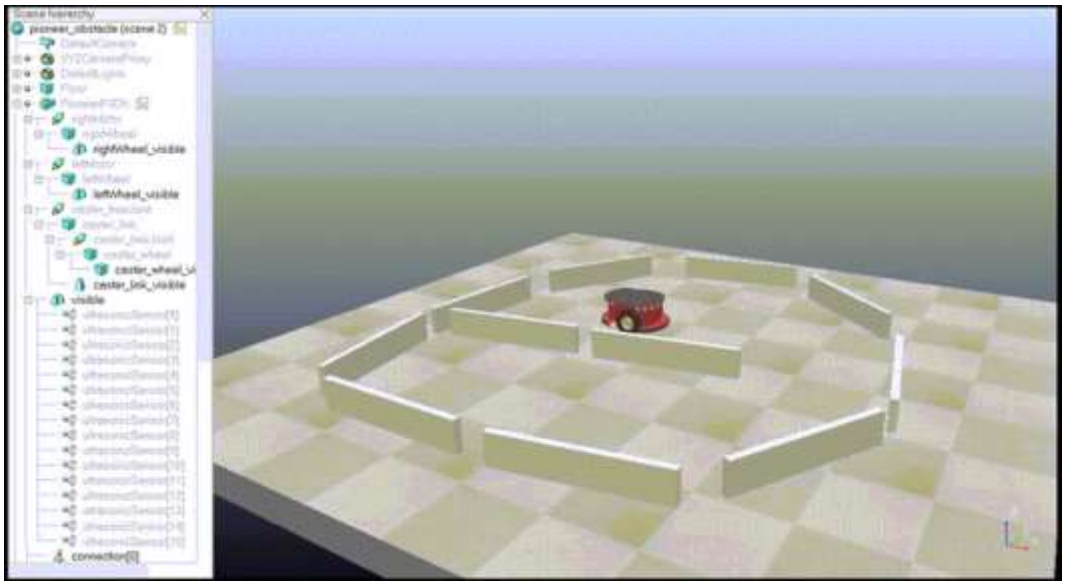


Рис. 15.16. Дослідження переміщення мобільного робота з виявленням перешкод

Контрольні запитання

1. Визначити, як здійснити встановлення на робот захоплюючого пристрою.
2. Назвати, який елемент використовується для приєднання захоплюючого пристрою до муфти робота.
3. Описати, як здійснити керування переміщенням маніпулятора та спрацюванням захоплюючого пристрою.
4. Визначити, як здійснити надійне захоплення у створеної моделі виробничої ділянки?
5. Описати, як встановити масу об’єкту.
6. Назвати, як встановити швидкість обертання для вказаної ланки маніпулятора?
7. Визначити, як побудувати графік траєкторії руху.
8. Назвати, як здійснити керування швидкості обертання коліс моделі мобільного робота **pioneer p3dx**.
9. Описати, як здійснити дослідження переміщення мобільного робота з виявленням перешкод.
10. Визначити, які датчики визначення перешкод є у мобільного робота **pioneer p3dx**.

Література

1. Рудь Ю.С. Основи конструювання машин: Підручник для студентів інженерно-технічних спеціальностей вищих навчальних закладів. 2-е вид., переробл. - Кривий Ріг: Видавець ФО-П Чернявський Д.О., 2015. – 492 с.
2. 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.
3. Проць. Я.І. Захоплювальні пристрої промислових роботів: Навчальний посібник . – Тернопіль: Тернопільський державний технічний університет ім. І. Пулюя, 2008. – 232с.
4. 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.
5. Михайлов Є. П. 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. – 171 с.
6. 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.
7. Бжихатлов И.А. Моделирование робототехнических систем в программе V-REP.Учебно-Методическое пособие. – СПб: Университет ИТМО, 2018. – 59с/
8. Застосування програмного комплексу CoppeliaSim на прикладі двоколісного мобільного робота Pioneer 3-DX. URL: https://habr.com/ru/post/648923/

Додаток 1

Тематика практичних та лабораторних занять

Практичні заняття

Практичне заняття 1. Конструювання та проєктування мехатронних машин на основі апаратно-програмного комплексу Arduino

Мета заняття

Ознайомлення з апаратно-програмним комплексом Arduino

Вивчення засоби програмування контролерів Arduino

Аналіз виконавчих та інформаційних пристроїв апаратно-програмного комплексу Arduino

Практичне заняття 2. Вивчення принципів конструювання транспортного засобу з визначенням перешкод

Мета заняття

Продемонструвати розуміння та вміння використання комп'ютерного проєктування транспортного засобу, що здійснює переміщення з визначенням перешкод за допомогою ультразвукового датчика та системи керування на основі апаратно-програмного комплексу Ардуіно.

Показати можливість моделювання такої системи на основі емулятора UnoArduSim.

Практичне заняття 3. Конструювання та моделювання виконавчих пристроїв машин за допомогою емулятора UnoArduSim

Мета заняття

Дослідження можливості конструювання та моделювання виконавчих пристроїв машин на основі контролера Arduino за допомогою емулятора UnoArduSim.

Створення та перевірка програм керування виконавчих пристроїв машин на основі електродвигунів з використанням емулятора **UnoArduSim.**

Практичне заняття 4. Конструювання та проєктування мобільних роботів за допомогою емулятора робота-маніпулятора Q2WDBotSim

Мета заняття

Продемонструвати розуміння та вміння використання комп'ютерного проєктування мобільного робота, що здійснює переміщення по вказаному маршруту з використанням датчиків відстані та визначення контрастної смуги за допомогою системи керування на основі апаратно-програмного комплексу Ардуіно та емулятора робота-маніпулятора Q2WDBotSim.

Практичне заняття 5. Приклади робототехнічних систем на основі програмованих логічних контролерів

Мета заняття

Ознайомлення з засобами проектування роботів на основі ПЛК

Використання функцій, функціональних блоків та даних при проектуванні програмних елементів

Приклади робототехнічних систем на основі ПЛК

Практичне заняття 6. Вивчення принципів конструювання механічного захоплювача промислового маніпулятора

Мета заняття

Створити тривимірну комп'ютерну модель та сценарій симуляції найпростішого механічного захоплювача для маніпулятора за допомогою платформи V-REP / CoppeliaSim.

Практичне заняття 7. Конструювання засобів ручного керування переміщенням ланок промислового робота

Мета заняття

Розглянути питання конструювання засобів ручного керування переміщенням ланок промислового робота шляхом створення програми за допомогою платформи V-REP (CoppeliaSim).

Лабораторні заняття

Лабораторне заняття 1. Конструювання та проєктування мехатронних машин на основі апаратно-програмного комплексу Arduino

Тривалість заняття 4 години

Мета заняття

Ознайомлення з апаратно-програмним комплексом Arduino

Вивчення засоби програмування контролерів Arduino

Аналіз виконавчих та інформаційних пристроїв апаратно-програмного комплексу Arduino

Лабораторне заняття 2. Вивчення принципів конструювання транспортного засобу з визначенням перешкод

Тривалість заняття 4 години

Мета заняття

Продемонструвати розуміння та вміння використання комп’ютерного проєктування транспортного засобу, що здійснює переміщення з визначенням перешкод за допомогою ультразвукового датчика та системи керування на основі апаратно-програмного комплексу Ардуіно.

Показати можливість моделювання такої системи на основі емулятора UnoArduSim.

Лабораторне заняття 3. Конструювання та моделювання виконавчих пристроїв машин за допомогою емулятора UnoArduSim

Тривалість заняття 4 години

Мета заняття

Дослідження можливості конструювання та моделювання виконавчих пристроїв машин на основі контролера Arduino за допомогою емулятора UnoArduSim.

Створення та перевірка програм керування виконавчих пристроїв машин на основі електродвигунів з використанням емулятора UnoArduSim.

Лабораторне заняття 4. Конструювання та проєктування мобільних роботів за допомогою емулятора робота-маніпулятора Q2WDBotSim

Тривалість заняття 4 години

Мета заняття

Продемонструвати розуміння та вміння використання комп’ютерного проєктування мобільного робота, що здійснює переміщення по вказаному маршруту з використанням датчиків відстані та визначення контрастної смуги за допомогою системи керування на основі апаратно-програмного комплексу Ардуіно та емулятора робота-маніпулятора Q2WDBotSim.

Лабораторне заняття 5. Приклади робототехнічних систем на основі програмованих логічних контролерів

Тривалість заняття 4 години

Мета заняття

Ознайомлення з засобами проектування роботів на основі ПЛК

Використання функцій, функціональних блоків та даних при проектуванні програмних елементів

Приклади робототехнічних систем на основі ПЛК

Лабораторне заняття 6. Вивчення принципів конструювання механічного захоплювача промислового маніпулятора

Тривалість заняття 4 години

Мета заняття

Створити тривимірну комп'ютерну модель та сценарій симуляції найпростішого механічного захоплювача для маніпулятора за допомогою платформи V-REP / CoppeliaSim.

Лабораторне заняття 7. Конструювання засобів ручного керування переміщенням ланок промислового робота

Тривалість заняття 4 години

Мета заняття

Розглянути питання конструювання засобів ручного керування переміщенням ланок промислового робота шляхом створення програми за допомогою платформи V-REP / CoppeliaSim.

Лабораторне заняття 8. Застосування графіків у CoppeliaSim на прикладі двоколісного мобільного робота Pioneer 3-DX

Тривалість заняття 2 години

Мета заняття

Розглянути питання застосування графіків у CoppeliaSim на прикладі двоколісного мобільного робота Pioneer 3-DX.

Додаток 2

Теоретичні положення та завдання для лабораторних та практичних занять

Заняття 1. Конструювання та проєктування мехатронних машин на основі апаратно-програмного комплексу Arduino

Апаратно-програмний комплекс Arduino

Цілий ряд фірм налагодив випуск одноплатних мікроконтролерів, побудованих за модульним принципом і здатних вирішувати велике коло завдань, в тому числі і завдань управління рухом, що характерно для машин на основі мехатронних пристроїв.

Прикладом таких контролерів може служити апаратно-програмний комплекс Arduino, який представляє собою набір апаратних та програмних засобів, пристосованих для побудови нескладних систем промислової автоматики і робототехніки.

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.

Інформація про компоненти апаратно-програмного комплексу Arduino доступна на сайті https://arduino.ua/.

Безкоштовна середа розробки Arduino IDE, що доступна на сайті https://www.arduino.cc/en/software дозволяє створити та завантажити програми для контролерів Arduino.

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. 1, є велика кількість різних датчиків і виконавчих пристроїв, що дозволяють вирішувати широке коло завдань.

Рис. 1. Апаратно-програмний комплекс Arduino

При цьому є як 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 програматоров. 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.

Деякі контролери Arduino розглянуті у розділі 3.2.

Засоби програмування контролерів Arduino

Засоби проєктування систем керування для роботів на основі контролерів Arduino найчастіше використовують мову С, яка є універсальною мовою, хоча і може мати деякі особливості.

Наприклад, такою мовою програмування є Arduino IDE, яка представляє собою середовище розробки, призначене для створення та редагування програмного коду.

Програма Arduino, яку називають скетч, складається з самостійного файла, в якому, на відміну від мови С, треба визначити принаймні, дві секції:

- перша називається 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.

Можна використовувати стандартні директиви препроцесора, такі як #define, #ifdef, #ifndef, #endif, та інші.

(Препроцесор це програма, яка виконує попередню обробку даних, для того, щоб вони могли використовуватись іншою програмою)

Зокрема, рекомендується визначати константи як символи препроцесора (не як змінні, бо вони займають багато пам'яті).

Змінна це комірка оперативної пам'яті, в якій зберігається інформація.

Програма використовує змінні для зберігання проміжних даних обчислень.

Для обчислень можуть бути використані дані різних форматів, різної розрядності, тому у змінних в мові С є, наприклад, такі типи.

Тип даних	Розрядність, біт	Диапазон 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

Типи даних вибираються виходячи з необхідної точності обчислень, форматів даних і т.п.

Не варто, наприклад, для лічильника, який вважає до 100, вибирати тип ***long***. Працювати буде, але операція займе більше пам'яті даних і програм та потребує більше часу для виконання.

Всі змінні повинні бути оголошені до того як будуть використовуватися.

Для оголошення змінних треба вказувати 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
```

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).

```
int x = 0;        // оголошується змінна x з початковим значенням 0
char d = 'a';     // оголошується змінна d з початковим значенням рівним коду символу "a“
```

Арифметичні операції:

=	присвоєння
+	складання
-	віднімання
*	множення
/	ділення
%	залишок від ділення

Операції відношення:

==	дорівнює
!=	не дорівнює
<	менше
>	більше
<=	менше або дорівнює
>=	більше або дорівнює

Логічні операції**:**

&&	логічне І
||	логічне АБО
!	логічне НІ

Функції керування виводами входів / виходів.

Для роботи з цифровими виводами в системі Ардуіно є три функції. Вони 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 analogWrite(pin, value)

Встановлює задану аналогову напругу на виході у вигляді ШІМ-сигналу.

На більшості плат Arduino функція analogWrite() працює з виходами 3, 5, 6, 9, 10 та 11. На Arduino Mega функція працює з виходами з 2 по 13. Аргументи pin і value:

pin - номер виводу;

value - коефіцієнт заповнення – лежить у межах від 0 (завжди вимкнено) до 255 (завжди включено).

Функція analogRead(pin)

Зчитує стан аналогового входу.

Аргументи: pin - номер виводу.

Повертає ціле число (від 0 да 1023).

Оператори керування програмою.

Оператор IF перевіряє умову в дужках і виконує наступне вираження або блок у фігурних дужках, якщо умова істинна. Синтаксис виглядає так:

```
if (x == 5)          // якщо x=5, то виконується z=0
z=0;
if (x > 5)           // якщо x > 5, то виконується блок z=0, y=8;
{   z=0;   y=8; }
```

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;
 }
```

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;
  }
```

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 ( int i=0; i < 100; i++ )     // початкове значення 0, кінцеве 99, крок 1
{
    sum = sum + 1;
}
```

Наведений приклад програми лічильника, який здійснює підрахунок з інтервалом 0,1 с. Коли значення лічильника дорівнює 150 підрахунок зупиняється, вмикається світлодіод на виході 13 на 5 с, після чого лічильник обнулюється і підрахунок починається спочатку.

```
#define ledPin 13 //визначити світлодіод
int count;        //визначити змінну лічильника як ціле число
void setup()
{  count=0;      //встановити початкове значення лічильника 0
digitalWrite(ledPin, LOW); // виключаємо світлодіод
}
void loop()
{  digitalWrite(ledPin, LOW); // виключаємо світлодіод
count=count+1; //додати до лічильника 1
delay(100);      //затримка рахування 0,1 с
if (count >=150) // Якщо кількість імпульсів більше 150)
{  digitalWrite(ledPin, HIGH); // включаємо світлодіод
delay(5000); //Термін включення світлодіода 5 с
count=0;         //встановити початкове значення лічильника 0
} }
```

Окремі компоненти, а саме,виконавчі пристрої та датчики апаратно-програмного комплексу Arduino розглянуті у розділі 3.2.

Контрольні питання

1. Викласти з чого складається апаратно-програмний комплекс Arduino.
2. Описати основні типи контролерів Arduino.
3. Розкрити засоби програмування контролерів Arduino.
4. Визначити, з чого складається скетч мови програмування Arduino IDE.
5. Пояснити, які функції використовують для керування виводами входів/виходів.
6. Описати, які оператори використовують для керування програмою.
7. Розкрити модулі керування електродвигунами постійного струму та кроковими двигунами.
8. Описати принцип роботи сервоприводу.
9. Описати принцип роботи датчиків швидкості та переміщення.
10. Описати принцип роботи датчиків перешкод.

Завдання

1. Ознайомитись з матеріалами, наведеними у презентації.
2. Надати відповіді на контрольні запитання.
3. Встановити на комп'ютер середу розробки Arduino IDE.
4. Скласти програму лічильника згідно з наведеним прикладом та параметрами, що надані для вказаних варіантів.
4. Перевірити роботу програми за допомогою додатку UnoArduSim.

Варіанти параметрів для програми лічильника

Варіант	1	2	3	4	5	6	7	8	9	10
Кількість імпульсів	100	140	90	110	50	200	160	180	190	140
Затримка рахування, с	0,01	0,02	0,03	0,05	0,04	0,01	0,02	0,03	0,05	0,04
Термін включення, с	5	4	7	6	2	8	4	3	5	4

Заняття 2. Конструювання та моделювання виконавчих пристроїв машин за допомогою емулятора UnoArduSim

Додаток UnoArduSim детально розглянутий у розділі 4.

Контрольні питання

1. Визначити, як можна використовувати додаток UnoArduSim.
2. Описати, які зони має вікно редагування Arduino програм.
3. Розкрити, де знайти інформацію про UnoArduSim.
4. Визначити, як встановити пристрої, що використовуються у проєкті.
5. Пояснити, які виконавчі пристрої може використовувати емулятор UnoArduSim.
6. Описати, які пристрої може використовувати для керування пристроєм.
7. Описати модель електродвигуна постійного струму.
8. Описати модель крокового двигуна.
9. Описати модель сервоприводу.
10. Розкрити принцип створення конфігурації при конструюванні машини.

Завдання

5. Ознайомитись з матеріалами, наведеними у презентації.
6. Надати відповіді на контрольні запитання.
7. Встановити на комп'ютер додаток UnoArduSim.
8. Скласти програму керування сервоприводом в автоматичному режимі для такої послідовності переміщень згідно з варіантами.
5. Перевірити роботу програми за допомогою додатку UnoArduSim.

Варіанти параметрів для переміщень

Варіант	Затримка між переміщеннями, с	Послідовність переміщень								
1	5	0	45	90	135	180	135	90	45	0
2	4	0	60	125	140	180	170	100	30	0
3	6	0	30	120	145	180	150	110	25	0
4	3	0	25	110	170	180	125	120	35	0
5	7	0	35	100	160	180	165	114	40	0
6	5	0	70	125	165	180	170	90	40	0
7	4	0	50	120	175	180	110	100	35	0
8	6	0	55	110	140	180	100	110	25	0
9	3	0	20	100	160	180	115	120	30	0
10	7	0	65	95	170	180	165	114	45	0

Заняття 3. Вивчення принципів конструювання транспортного засобу з визначенням перешкод

Лабораторна робота виконується з використанням ультразвукового датчика HC-SR04.

Створення моделі ультразвукового датчика HC-SR04 та використання його для пошуку перешкод детально розлянуто у розілі 4.3.

Опис експериментальної установки

Для дослідження використовується ультразвуковий датчик HC-SR04, підключений до контролера Arduino Nano, як це показано на рис. 4.12.

Для створення та перевірки програми будемо використовувати емулятор UnoArduSim.

Набір пристроїв UnoArduSim для мобільного робота з двома двигунами постійного струму та інструментом '1SHOT‘, за допомогою якого здійснюється моделювання ультразвукого датчика вимірювання відстані, а також інструментом послідовного виведення даних на комп’ютер 'SERIAL‘наведений на рис. 4.17. Приклад програми вихначення перешкод наведений у розділі у розілі 4.3.

Контрольні питання

1. Визначити, у чому полягаю принцип роботи датчика HC-SR04.
2. Розповісти, як визначити відстань за допомогою датчика HC-SR04.
3. Назвати, за допомогою якої функції визначається тривалість імпульсів.
4. Визначити, як підключити датчик до контролера Arduino Nano.
5. Назвати, до якої команди треба ввести зміни для визначення відстані спрацювання світлодіоду та виключення двигунів.
6. Розповісти, як здійснити зміну напрямку руху.

Завдання

1. Ознайомитись з принципом роботи датчика HC-SR04 та алгоритмом та програмою визначення відстані за допомогою цього датчика.
2. Виходячи з ширини транспортного засобу s та кута огляду ультразвукового датчика α, що наведені у варіантах параметрів, визначити відстань L, яка визначає відсутність перешкод на шляху транспортного засобу.
3. Перевірити роботу програми, що наведена у прикладі, на емуляторі UnoArduSim.
4. Ввести зміни до програми, що дозволять включити світлодіод, якщо відстань до перешкоди менше ніж L см.
5. Ввести зміни до програми, що дозволять здійснювати розворот на місці, якщо відстань до перешкоди менше ніж L см.

Варіанти параметрів

Варіант	1	2	3	4	5	6	7	8	9	10
Ширина транспортного засобу s, м	0,15	0,25	0,20	0,30	0,27	0,15	0,25	0,20	0,30	0,27
Кут огляду α, º	15	20	14	18	25	23	18	16	20	15

Заняття 4 Конструювання та проєктування мобільних роботів за допомогою емулятора робота-маніпулятора Q2WDBotSim

Мета заняття

Продемонструвати розуміння та вміння використання комп'ютерного проєктування мобільного робота, що здійснює переміщення по вказаному маршруту з використанням датчиків відстані та визначення контрастної смуги за допомогою системи керування на основі апаратно-програмного комплексу Ардуіно та емулятора робота-маніпулятора Q2WDBotSim (рис. 1).

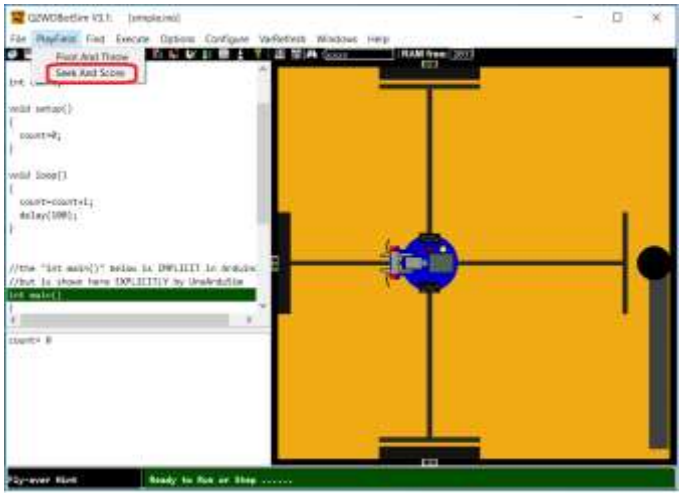


Рис. 1. Емулятор робота-маніпулятора Q2WDBotSim

Емулятор робота-маніпулятора Q2WDBotSim

Емулятор робота-маніпулятора Q2WDBotSimV3.1 входить до складу UnoArduSimV2.9.2 (дивись наступний слайд).

Додаток UnoArduSim є безкоштовним емулятором контролера Arduino, який дає можливість здійснити виконання програми в реальному часі без наявності самої плати Arduino.

UnoArduSim та Q2WDBotSim не потребують інсталяції та запускаються відповідними файлами **UnoArduSim.exe** та **Q2WDBotSim.exe**.

Вихідне вікно емулятора Q2WDBotSimV3.1 з ігровим полем **Seek And Score** (PlayField / Seek And Score) та мобільним роботом з маніпулятором наведені на рис. 1..

Розміри PlayField.

- PlayField — це квадрат розміром 48 на 48 дюймів (121,9 см)
- Поверхня підлоги становить 46,5 квадратних дюймів
- Виступи м'яча мають довжину 12 дюймів і ширину 1,5 дюйма
- Центральна вертикальна стрічкова лінія знаходиться на відстані 18 дюймів (45,7 см) від лівої стіни
- Верхня та нижня горизонтальні стрічки розташовані на відстані 2 дюймів (5,08 см) від стіни
- Вертикальна стрічкова лінія воріт знаходиться на відстані 5 дюймів (12,7 см) від стіни
- М'ячі мають діаметр 1 дюйм (2,54 см).
- Лінії підлоги з чорної стрічки мають ширину 1,7 см
- Верхній край виступу м'яча знаходиться на 1,25 см нижче осі повороту сервоприводу захвату
- Край чашки воріт знаходиться на 3 см вище осі повороту сервоприводу нахилу захвату.

Розміри бота (візка).

- Загальна довжина = 23 см (9 дюймів)

- Колісна база = 14 см (5,5 дюймів)
- Окружність колеса = 20,5 см (8 дюймів)
- Висота передніх бамперів = 4 см (1,5 дюйма)
- Відстань від осей коліс до передніх бамперів = 14 см (5,5 дюймів)
- Відстань від осей коліс до осі повороту сервоприводу захвату = 7 см (2,75 дюйма)
- Довжина захвату (від осі повороту до сервоприводу нахилу) = 9 см (3,5 дюйма)
- Відстань від осей коліс до інфрачервоних датчиків лінії відстеження = 12 см (4,75 дюйма)
- Відстань між трьома інфрачервоними датчиками лінійного трекера (лівий, центральний, правий) = 2 см (0,75 дюйма)
- Висота осі нахилу захвату над підлогою = 14,5 см (5,7 дюйма)

Налагодження вікна здійснюється за допомогою меню **Configure**.

Пункт меню **Configure / Wire Up Pins** можна використовувати для відкриття діалогового вікна, де наведено підключення виконавчих пристроїв та датчиків (рис. 2).

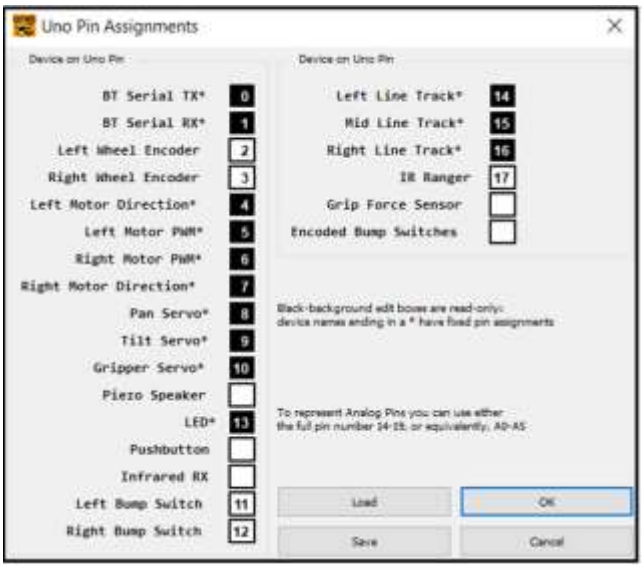


Рис. 2. Діалогове вікно, на якому наведено підключення виконавчих пристроїв та датчиків

Пункт меню Configure / Preferences використовується для встановлення окремих параметрів вікна та програми.

Пункт меню Configure / Wheel Motor Speed Mismatch використовується для встановлення невідповідності швидкості обертання (кількості обертів) двигунів коліс, якщо вказана однакова швидкість обертання (рис. 3).

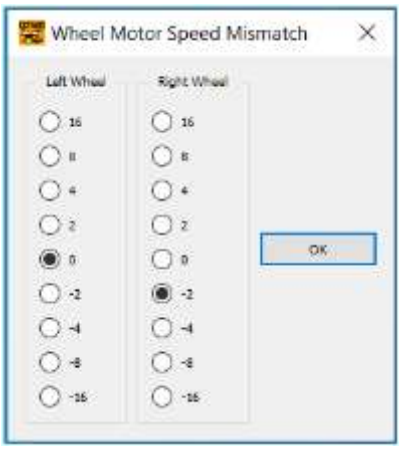


Рис. 3. Встановлення невідповідності швидкості обертання

На полі **Seek And Score** встановлений емулятор робота-маніпулятора, спроєктований на основі роботизованого комплекту DFRobot 2WD з маніпулятором (рис. 4).

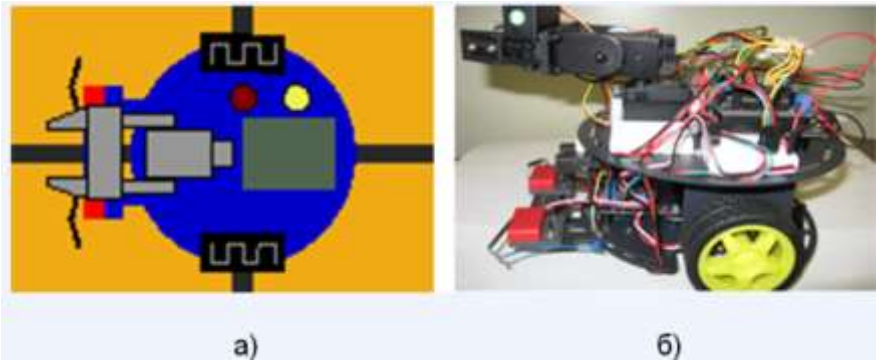


Рис.4. Емулятор робота-маніпулятора (а), спроєктований на основі роботизованого комплекту DFRobot 2WD з маніпулятором (б).

Переміщення маніпулятора по полю можна здійснити за допомогою миші, пересування шляхом натискування на ліву кнопку, обертання шляхом натискування на праву кнопку (рис. 5).

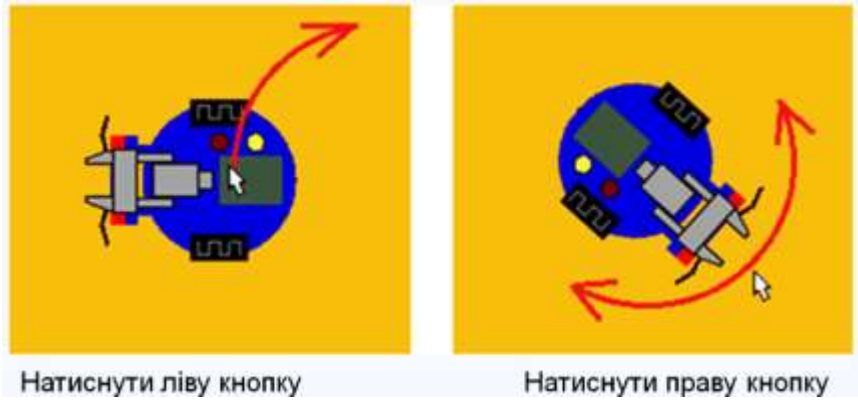


Рис. 5. Переміщення маніпулятора по полю

Колісні двигуни

Це редукторні двигуни постійного струму, що приводяться в дію драйвером з двома двигунами, кожен з яких має контакти напрямку обертання DIR, відповідно **Left Motor Direction** для лівого та **Right Motor Direction** для правого, та швидкості обертання за допомогою широтно-імпульсної модуляції PWM, відповідно **Left Motor PWM** для лівого та **Right Motor PWM** для правого.

Сигнали **HIGH** на вході DIR створюють рух візка вперед з боку відповідного колеса, а **Low** у зворотному напрямку. Встановлюються командою **digitalWrite()**.

Вивід PWM реагує тільки на сигнали, які створені командою **analogWrite()**.

Максимальна швидкість обертання колеса становить близько 1 оборот за секунду (при коефіцієнті заповнення ШІМ = 1,0 що відповідає максимальному значенню 255).

Колісні енкодери мають 10 зубців на оборот, які переривають інфрачервоний промінь і, таким чином, створюють 20 змін цифрового рівня за оборот колеса на підключеному цифровому вході (зміна з НИЗЬКОГО на ВИСОКУ та зміна з ВИСОКОГО на НИЗЬКЕ чергуються).

Оскільки окружність колеса дорівнює 20,5 см, маємо на одне спрацювання енкодера переміщення на 1,025 см

Для підрахунку імпульсів, які видають енкодери, найчастіше використовують переривання.

У більшості плат Ардуїно існує два зовнішні переривання: номер 0 (цифровий вивід 2) та 1 (цифровий вивід 3).

Номери виводів для зовнішніх переривань, доступні в платах Ардуїно, наведено в таблиці нижче:

Плата	int.0	int.1	int.2	int.3	int.4	int.5
Uno	2	3				
Mega2560	2	3	21	20	19	18

Функція встановлення переривання **attachInterrupt()**

Синтаксис

attachInterrupt(interrupt, function, mode)

Параметри

interrupt:

номер переривання (int)

function:

функція, яку необхідно викликати у разі переривання; ця функція не повинна мати параметрів і не повертати жодних значень.

mode:

визначає умову, у якому має спрацьовувати переривання. Може приймати одне з чотирьох визначених значень:

LOW - переривання буде спрацьовувати щоразу, коли на виводі є низький рівень сигналу

CHANGE - переривання буде спрацьовувати щоразу, коли змінюється стан виведення

RISING – переривання спрацює, коли стан виведення зміниться з низького рівня на високий

FALLING – переривання спрацює, коли стан виведення зміниться з високого рівня на низький.

Функція заборони переривання **detachInterrupt()**

Забороняє переривання.

Синтаксис

detachInterrupt(interrupt)

Параметри

interrupt: номер переривання, яке необхідно заборонити

Далі наведена програма переміщення візка на встановлену відстань, яка визначається за допомогою енкодера.

Шлях переміщення визначається як шлях переміщення між двома імпульсами, помножений на кількість спрацювань.

Шлях переміщення між двома спрацюваннями (1,025) см визначається як довжина кола колеса (20,5 см), поділене на кількість спрацювань за одне обертання (20).

```
/* Програма переміщення візка на встановлену відстань odometry*/
//підключення входів модуля до відповідних контактів
const int LW = 5; //лівий двигун PWM
const int RW = 6; //правий двигун PWM
const int LMD = 4; //лівий двигун напрямок обертання
const int RMD = 7; //правий двигун напрямок обертання
int speed = 0; //швидкість обртання коліс
volatile unsigned int pulses = 0; //кількість імпульсів
//функція підрахунку імпульсів
void counter() {
pulses++; }
void setup() {
//режим виходів керування двигунами OUTPUT
```

```
pinMode(RW, OUTPUT);
pinMode(LW, OUTPUT);
pinMode(LMD, OUTPUT);
pinMode(RMD, OUTPUT);
//вихідний напрямок руху
digitalWrite(LMD, HIGH); //рух вперед
digitalWrite(RMD, HIGH); //рух вперед
pulses = 0;
// дозвіл переривання від датчика переміщення
// Встановлення контакта 2 для переривання при зміні стану
attachInterrupt(0, counter, CHANGE);
delay(100); } //затримка 0,1 с
void loop() {
speed = 155; //встановлення швидкості обртання коліс
analogWrite(RW, speed+2); //рух вперед
analogWrite(LW, speed);
if (pulses >= 20) //якщо кількість спрацювань перевищує 20, зупинка
{ // заборона переривання
detachInterrupt(0);
//Двигуни Стоп
analogWrite(RW, 0); //стоп
analogWrite(LW, 0);
delay(2000);
pulses = 0;
// дозвіл переривання
attachInterrupt(0, counter, CHANGE);
} }
```

Датчики відстеження стрічки LFM

Спрямовані вниз LFM датчики відбитого світла для відстеження стрічки видають 3 аналогові вихідні напруги від трьох активних відбиваючих ІЧ-датчиків, відповідно **Left Line Track** для лівого, **Mid Line Track** для середнього та **Right Line Track** для правого.

Для дослідження засобів переміщення по траєкторії на основі слідкування за контрастною стрічкою використовуються стрічки на підлозі, які пофарбовані в сірий колір.

У Q2WDBotSim змодельований рівень для analogRead() дорівнює 600 на більш світлому фону підлоги та 400 над темними стрічками (рис.6).

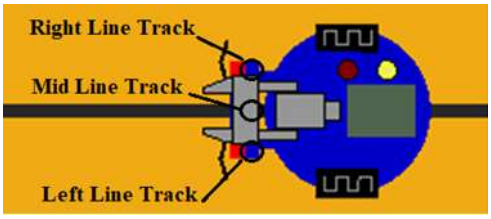


Рис. 6. Переміщення по траєкторії на основі слідкування за контрастною стрічкою

Інфрачервоний-датчик відстані

Цей спрямований вперед інфрачервоний датчик ІЧ-датчик (**IR Ranger**) вимірює відстань та видає аналогову напругу, нелінійну залежно від відстані до стіни перед ботом.

Вихідна напруга становить 0,4 вольта на відстані 80 см, зростає по исхідній кривій, поки не досягає максимуму 2,6 вольта на відстані 8 см, а потім починає швидко падати в міру зменшення відстані нижче 8 см.

На рис. 7 наведена залежність вихідної напруги від відстані для інфрачервоного датчика фірми SHARP GP2Y0A21YK0F.

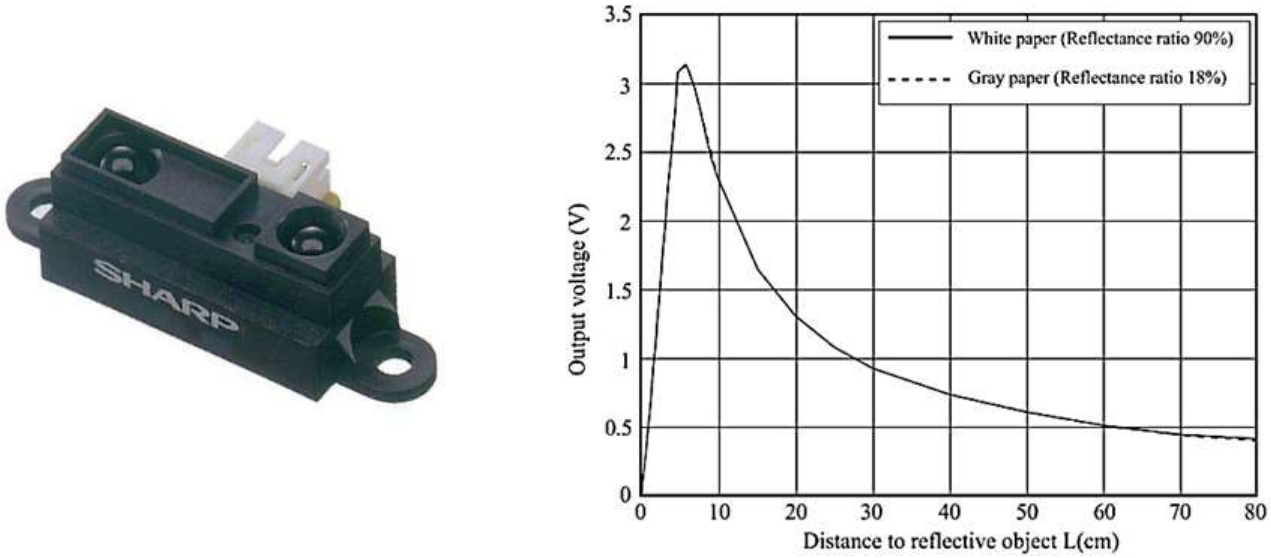


Рис. 7. Датчик GP2Y0A21YK0F та залежність вихідної напруги від відстані

На рис. 8 наведена залежність вихідної напруги від відстані, отримана від датчика відстані в залежності від положення за допомогою програми **odometry**.

Приклад програми **strip_tracking** для переміщення робота-маніпулятора з використанням слідкування за контрастною стрічкою.

```
//підключення входів модуля до відповідних контактів
const int LW = 5; //лівий двигун, PWM
const int RW = 6; //правий двигун, PWM
const int LMD = 4; //лівий двигун, напрямок обертання
const int RMD = 7; //правий двигун, напрямок обертання
int LLT = 0; // змінна для значення лівого датчика слідкування
int RLT = 0; // змінна для значення правого датчика слідкування
int IR = 0; // змінна для значення інфрачервоного датчика відстані
void setup() {
//режим виходів керування двигунами OUTPUT
pinMode(RW, OUTPUT);
pinMode(LW, OUTPUT);
pinMode(LMD, OUTPUT);
pinMode(RMD, OUTPUT);
digitalWrite(LMD, HIGH); //рух вперед
digitalWrite(RMD, HIGH); //рух вперед
 }
void loop() {
//опитування датчиків слідкування та відстані
LLT = analogRead(0);
RLT = analogRead(2);
IR = analogRead(3);
if (RLT < 500)//якщо спрацював правий датчик, рух направо
{ analogWrite(RW, 50);
analogWrite(LW, 155); }
else if (LLT < 500) //якщо спрацював лівий датчик, рух наліво
{ analogWrite(RW, 155);
analogWrite(LW, 50); }
else //якщо датчики не спрацювали, рух вперед
```

```
{ analogWrite(RW, 155);
analogWrite(LW, 155); }
if (LLT < 500 && RLT < 500) //якщо спрацювали обидва датчика, стоп
{ analogWrite(RW, 0);
analogWrite(LW, 0); } }
```

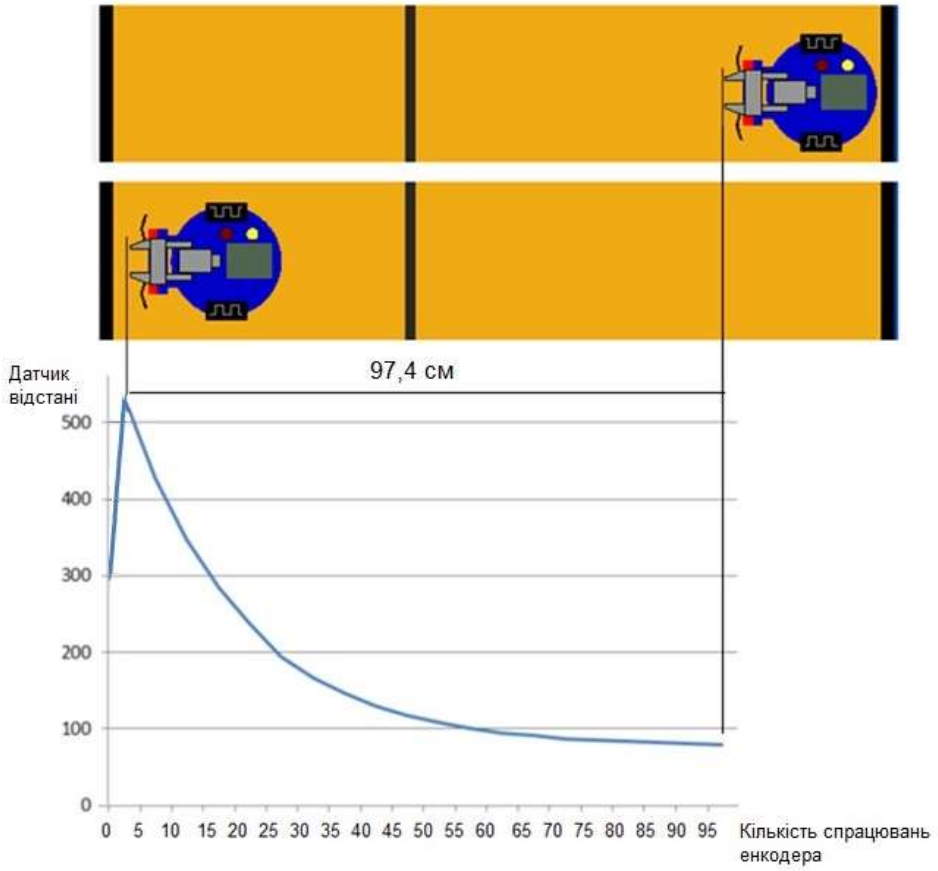


Рис. 8. Залежність вихідної напруги від відстані, отримана від датчика відстані в залежності від положення

Контрольні питання

1. Визначити, як здійснити маршрутослідкуванням за допомогою двох оптичних датчиків.
2. Описати, за якою умовою здійснюється зупинка при наявності перешкоди.
3. Назвати, за якою умовою здійснюється переміщення по прямій лінії.
4. Визначити, за якою умовою здійснюється поворот направо.
5. Назвати, за якою умовою здійснюється поворот наліво.

Завдання

1. Ознайомитись з програмою, яка забезпечує переміщення транспортного засобу вздовж темної смуги на світлому фоні.
2. При наїзді на поперечну смугу (спрацьовують обидві датчики) транспортний засіб зупиняється, поки не зникне перешкода.
3. Доповнити блок-схему алгоритму зупинкою візка при наявності перешкод, використовуючи приклад програми, наведений раніше.
4. При наявності перешкоди транспортний засіб зупиняється, поки не зникне перешкода.

Заняття 5 Приклади робототехнічних систем на основі програмованих логічних контролерів

Приклад проєктування систем циклового керування на основі контролера LOGO!

Розглянемо систему циклового керування маніпулятором на основі контролера LOGO! (рис. 1).

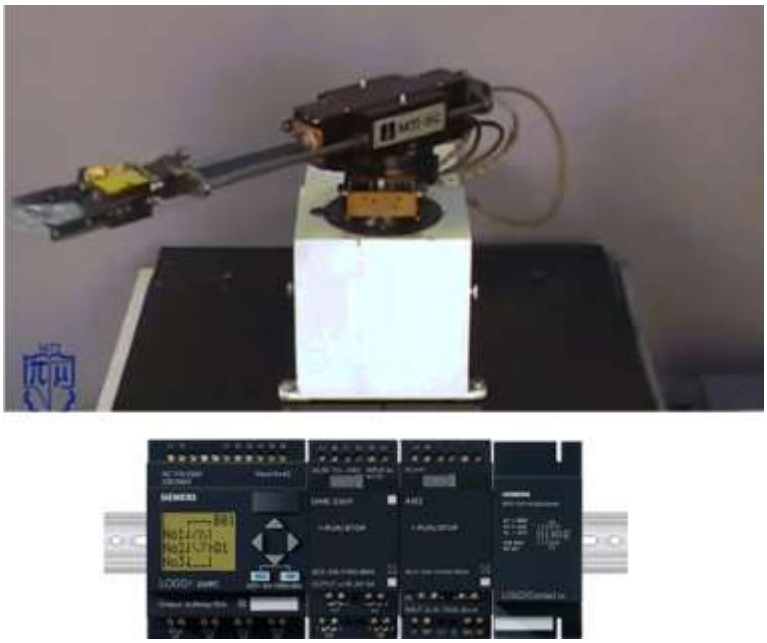


Рис. 1. Робот з цикловим керуванням та контролер LOGO!

Вихідні дані

Маніпулятор з циклових управлінням виконує такі переміщення:

ВГОРУ, ВНИЗ, ВПРАВО, ВЛІВО, ВПЕРЕД, НАЗАД

На рис. 2 наведені кінематична схема і робоча зона маніпулятора

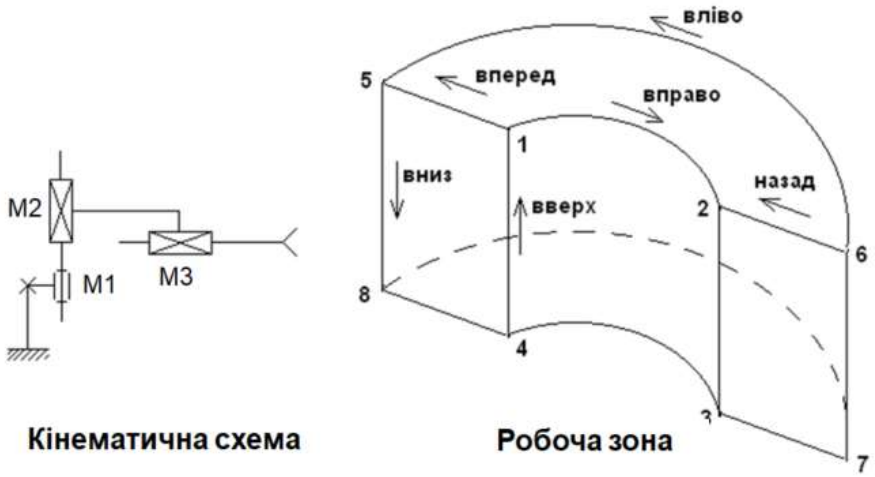


Рис. 2. Кінематична схема і робоча зона маніпулятора

Схема підключення маніпулятора до контролера LOGO! наведена на рис. 3.

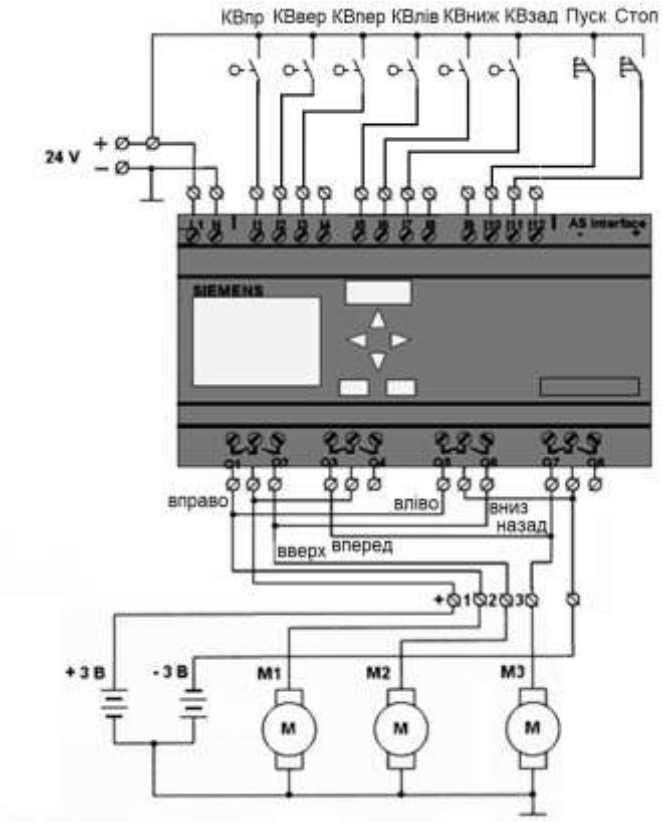


Рис. 3. Схема підключення маніпулятора до контролера LOGO!

Для програмування контролера LOGO! використовується програмне середовище LOGO! Soft Comfort.

Сторінка завантаження демонстраційної версії LOGO! Soft Comfort.

Download a demo version of LOGO! Soft Comfort:

http://w3.siemens.com/mcms/programmable-logic-controller/en/logic-module-logo/demo-software/pages/default.aspx

Складання програми для контролера LOGO! за допомогою програмного середовища LOGO! Soft Comfort.

На наступних рисунках наведено вихідне вікно та елементи для введення програми.

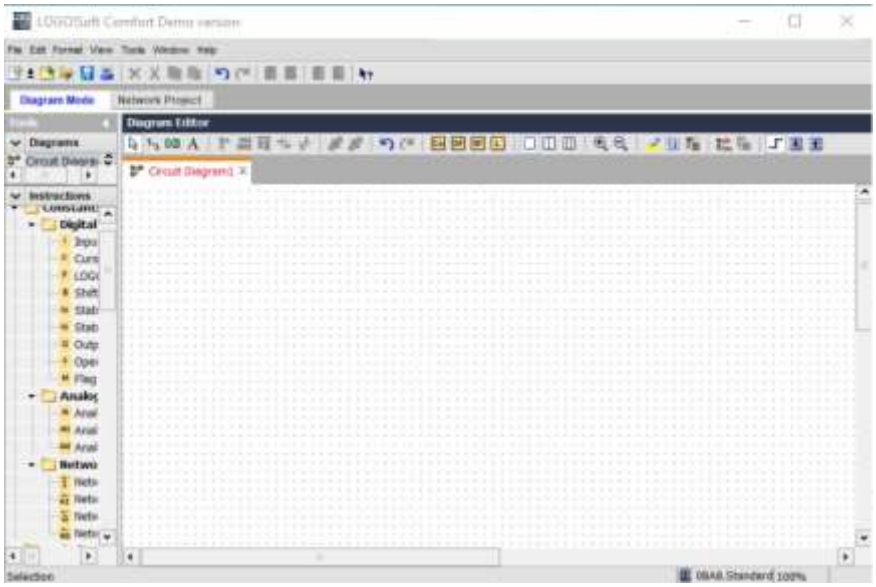


Рис. 4. Вихідне вікно для введення програми

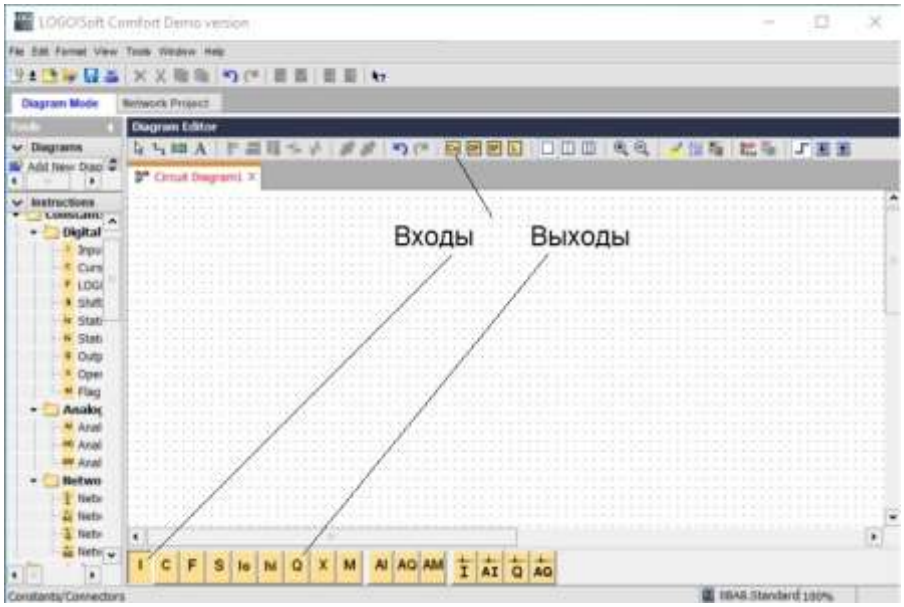


Рис. 5. Входи та виходи

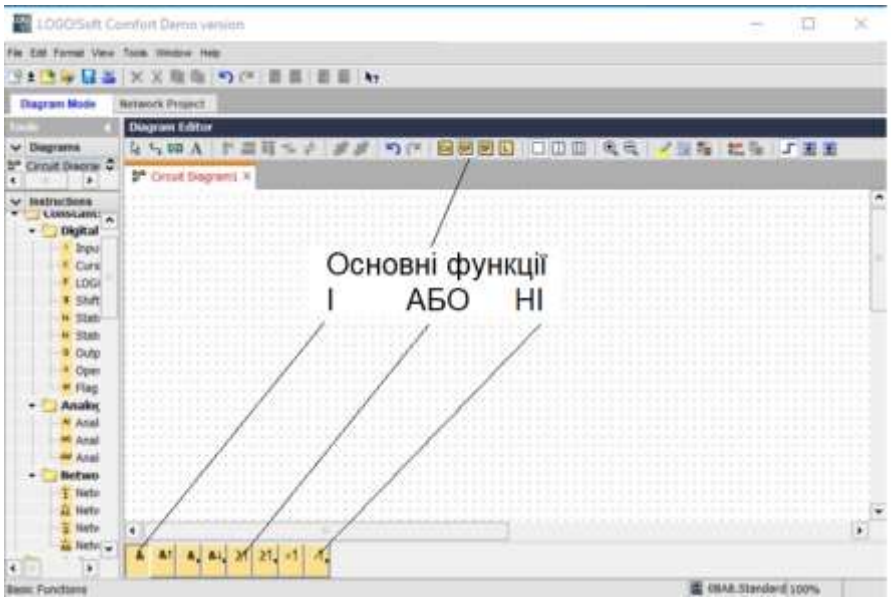


Рис. 6. Основні логічні функції

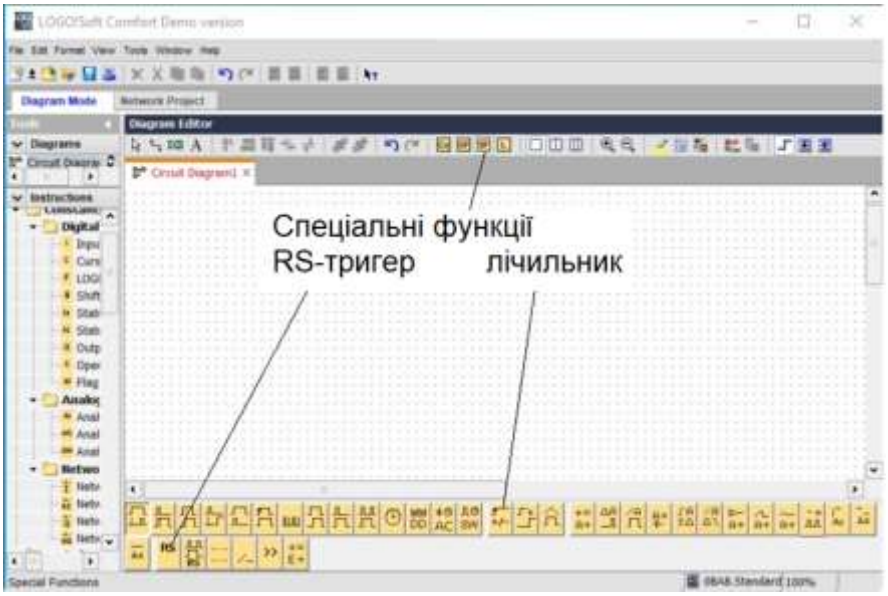


Рис. 7. Спеціальні функції

На наступних рисунках наведене створення програми.

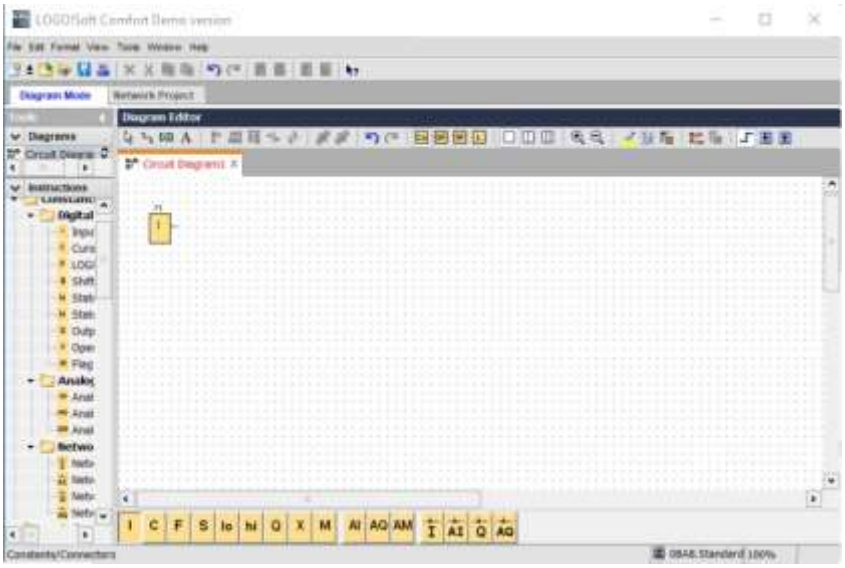


Рис. 8. Введення елемента програми – входу I1

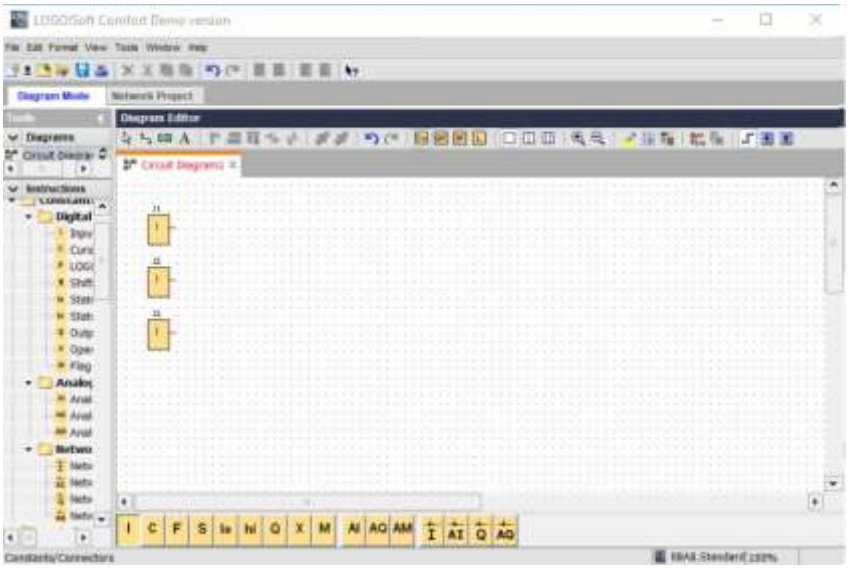


Рис. 9. Введення елементів програми – входів I2, I3

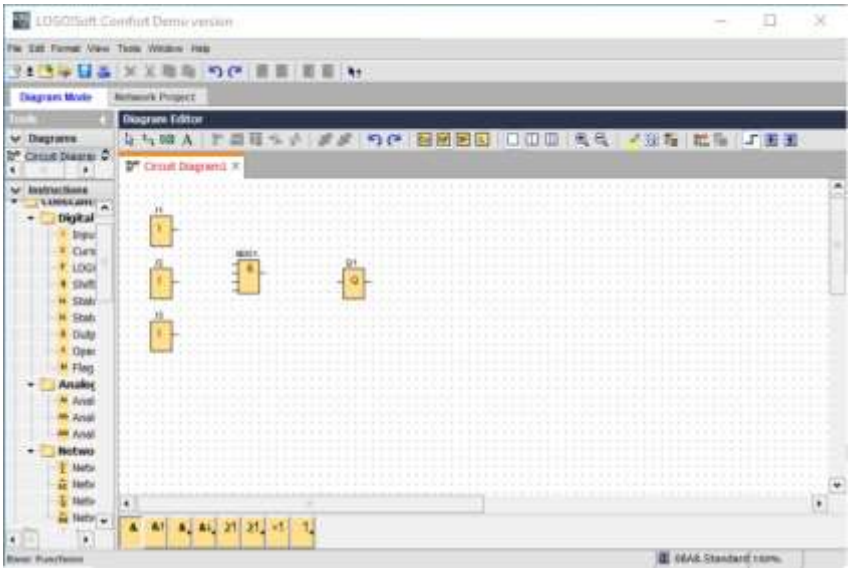


Рис. 10. Введення елементів програми – виходу Q1 і функції I

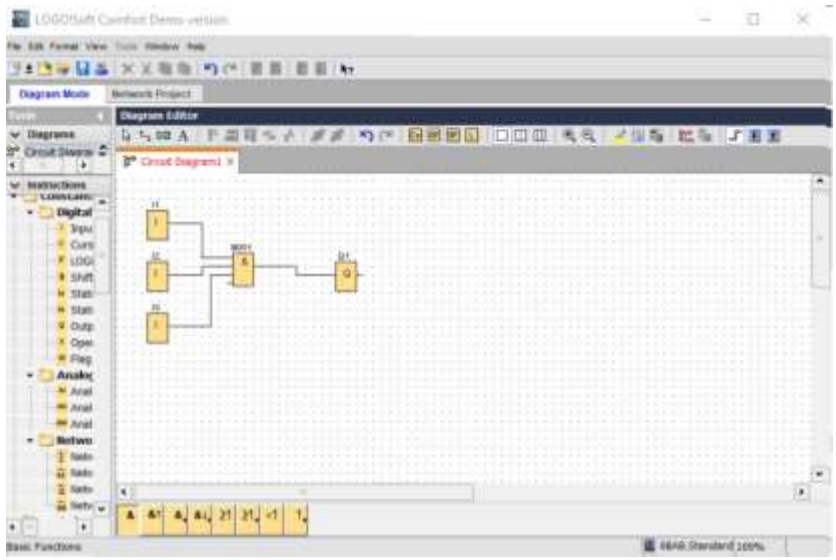


Рис. 11. Поєднання елементів програми – входів I1, I2, I3, виходу Q1 и функції I

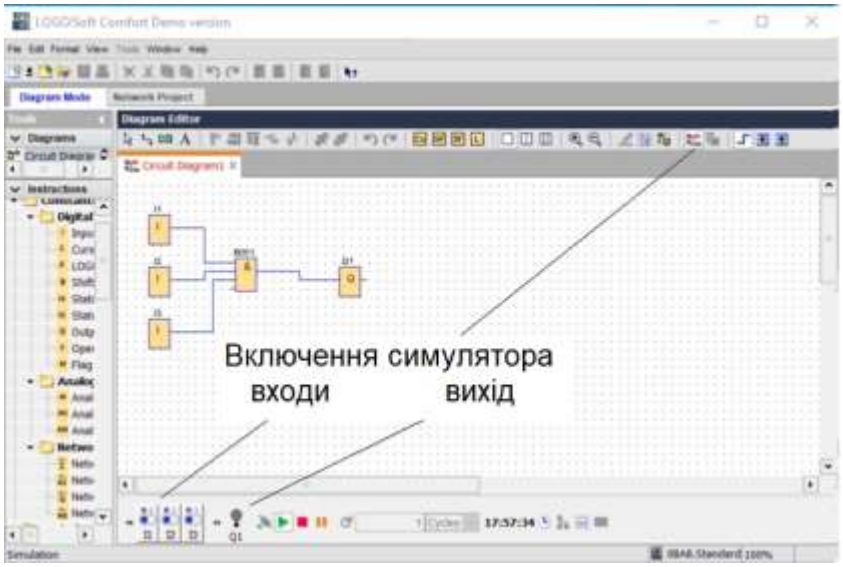


Рис. 12. Включення симулятора – вихідний стан

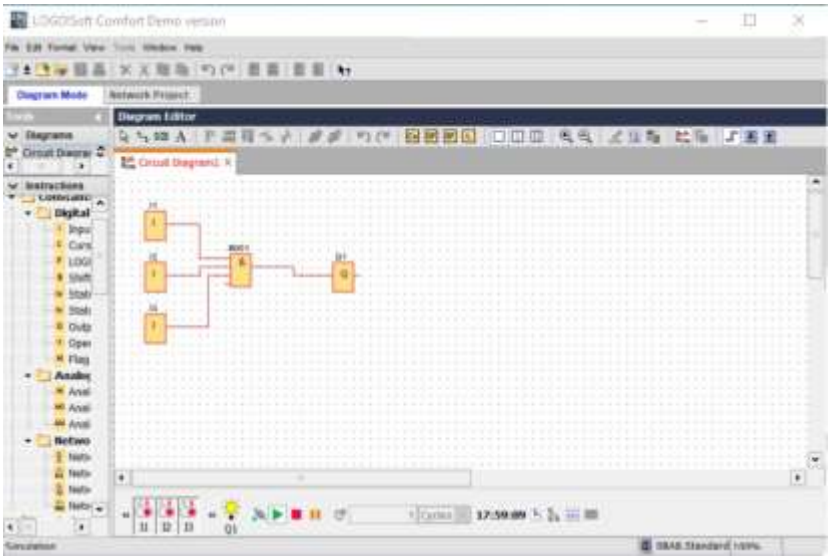


Рис. 13. Включення симулятора – на всі входи поданий сигнал 1, на виході маємо **1**

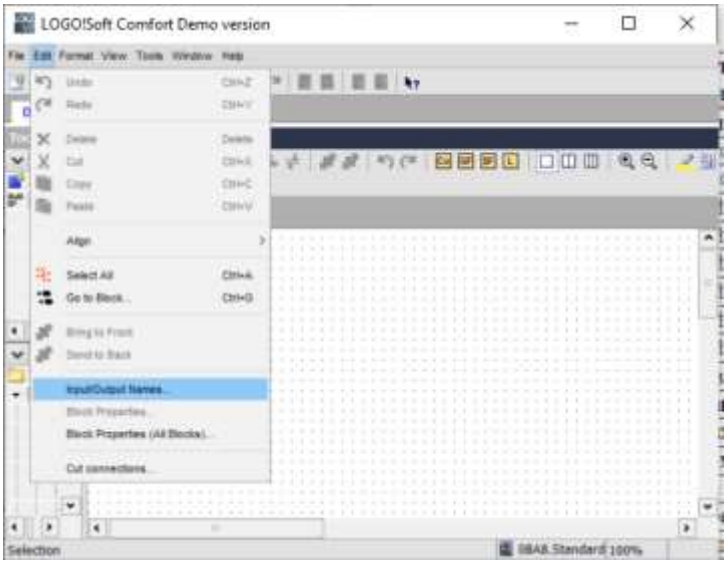


Рис. 14. Виклик символьної таблиці

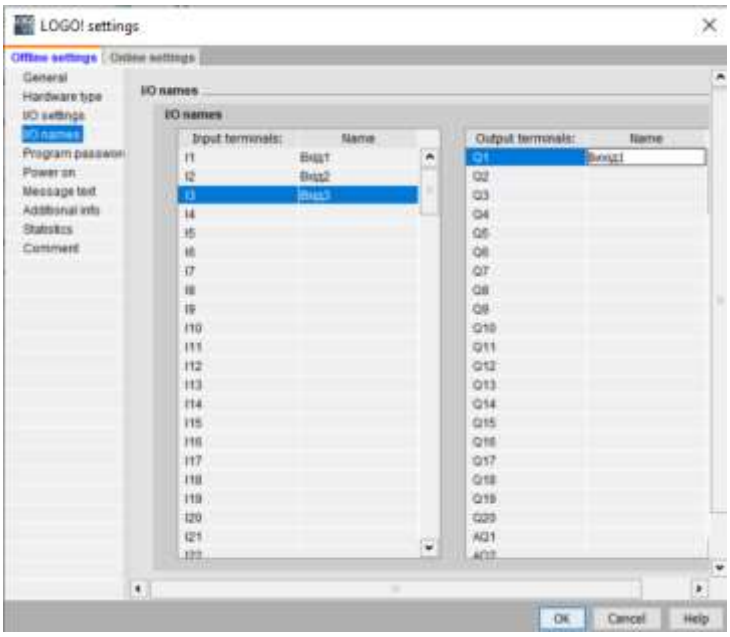


Рис. 15. Створення символьної таблиці

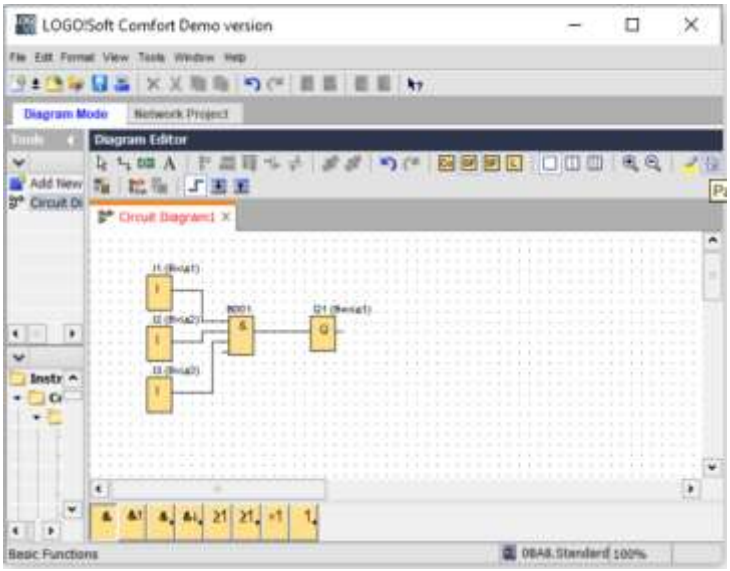
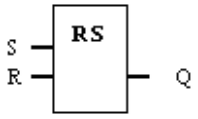


Рис. 16. Результат створення символьної таблиці

Послідовне виконання операцій



Для послідовного виконання операцій можна використовувати функцію RS-тригера
Сигнал на вході S включає вихід Q.
Сигнал на вході R вимикає вихід Q.
При подачі сигналу на обидва входи пріоритет має вхід R.

На наступному слайді показаний приклад послідовного переміщення:

- сигнал ПУСК на вході I1 включає Привод1 на виході Q1;
- спрацьовування кінцевого вимикача КВ1 на вході I3 вимикає Привод1 і включає Привод2 на виході Q2;
- спрацьовування кінцевого вимикача КВ2 на вході I4 вимикає Привод2 і включає Привод3 на виході Q3;
- спрацьовування кінцевого вимикача КВ3 на вході I5 вимикає Привод3;
- сигнал СТОП на вході I2 вимикає всі виходи (Q1, Q2 и Q3).

На рис. 17 наведений лриклад програми послідовного циклового переміщення для наведеної схеми підключення маніпулятора до контролера LOGO!.

Послідовність переміщень:

вправо, вверх, вперед, вліво, вниз, назад.

Вихідна позиція 4.

На рис. 18 наведена символьна таблиця, де наведені позначення входів та виходів, що використовуються у програмі.

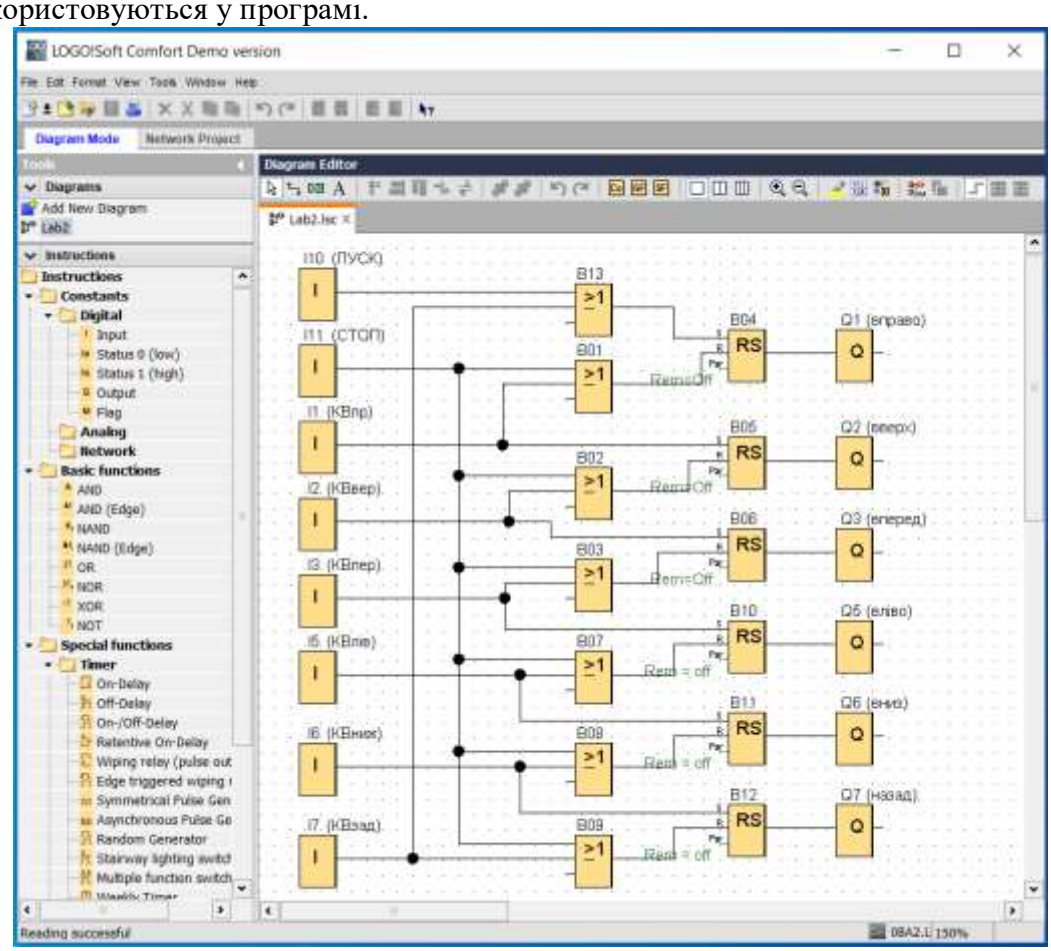


Рис. 17. Програма послідовного циклового переміщення

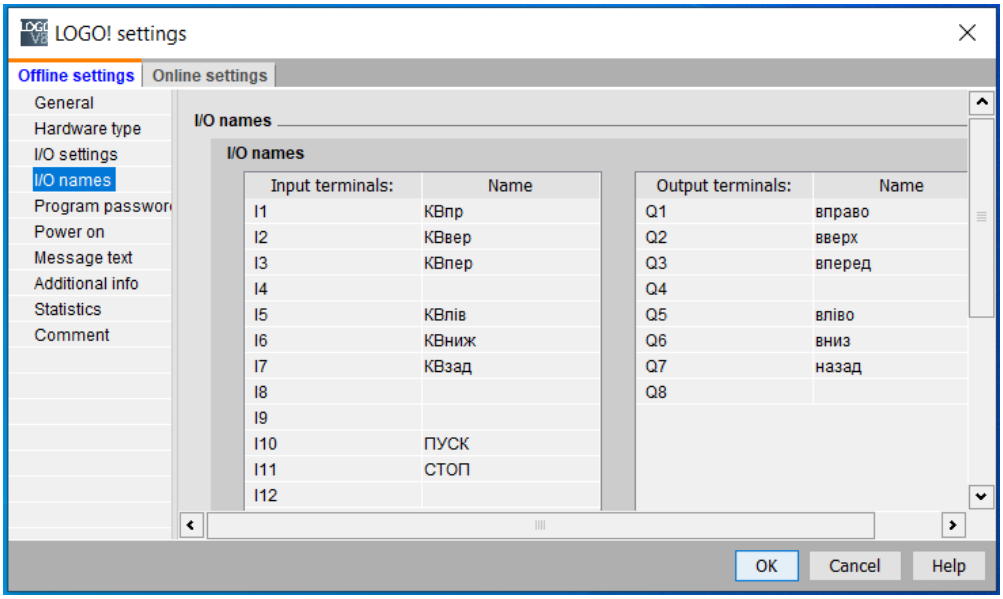


Рис. 18. Символьна таблиця Edit > Input/Output Names

На рис. 19 нпоказано, як здійснюється перевірка програми за допомогою симулятора. Натиснута кнопка «Пуск» I10, включено переміщення «Вправо» Q1

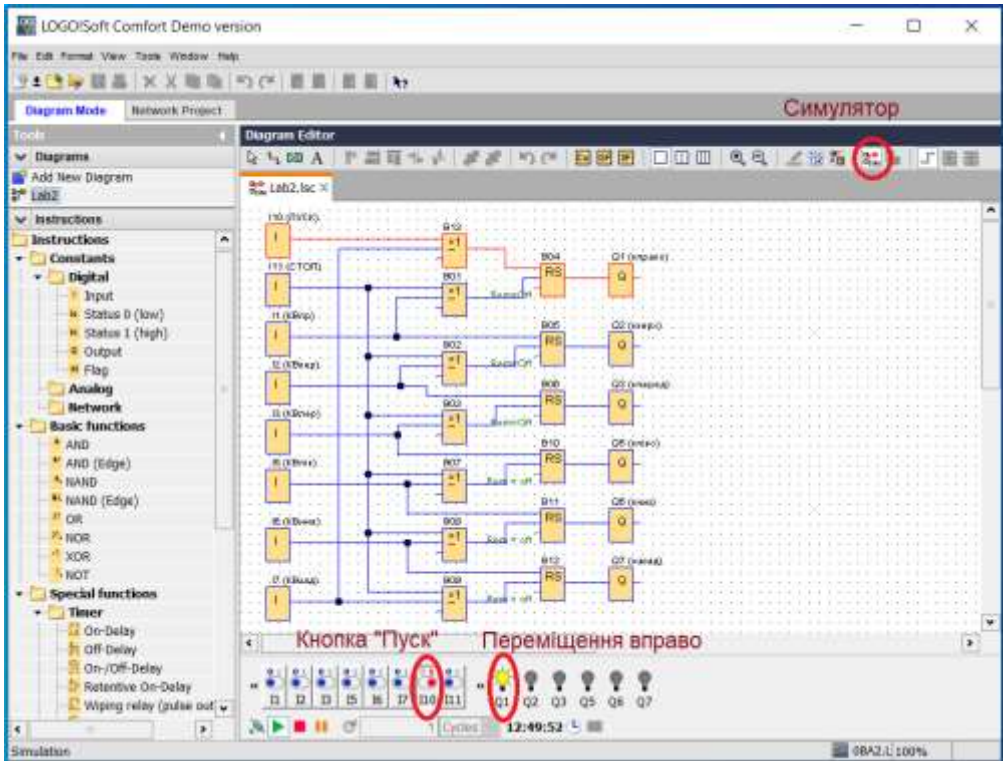


Рис. 19. Перевірка програми за допомогою симулятора

Контрольні запитання

1. Визначити, які переміщення здійснює наведений маніпулятор з цикловим керуванням.
2. Назвати, які елементи опитують датчики.
3. Розповісти, які елементи здійснюють керування виконавчими пристроями.
4. Назвати, які логічні функції .
5. Визначити, який елемент дозволяє здійснити послідовність дій.
6. Описати, як можна здійснити підрахунок подій.

Завдання

1. Ознайомитись, які переміщення виконує маніпулятор з циклових управлінням..
2. Скласти програму переміщення маніпулятора у вказану позицію та повернення назад, використовуючи наведений вище приклад, відповідно до варіанта.
3. Перевірити програму за допомогою LOGO! Soft Comfort.

Варінти переміщення маніпулятора

Варіант	1	2	3	4	5	6
Послідовність переміщень	7-6-5	7-3-2	7-6-2	7-3-4	4-8-7	4-1-5
Варіант	7	8	9	10	11	12
Послідовність переміщень	4-1-2	4-8-7	8-5-6	8-5-1	8-4-1	8-4-3

Заняття 6. Вивчення принципів конструювання механічного захоплювача промислового маніпулятора

Проєктування захоплювача промислового робота

У каталозі моделей роботів та їх компонентів є досить велика кількість захоплюючих пристроїв, наприклад, Franka gripper (рис. 1).

Рис. 1. Захоплюючий пристрій Franka gripper

Розглянемо можлливість створення простого захоплювача, який складається з одного нерухомого та двох рухомих елементів (рис. 2) за допомогою платформи V-REP / CoppeliaSim.

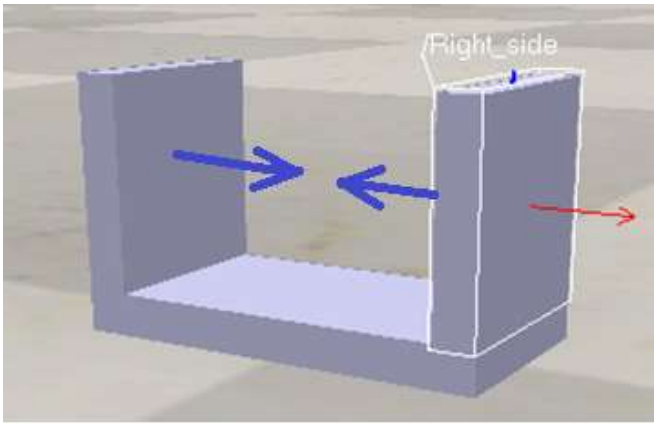


Рис. 2. Захоплювач, який складається з одного нерухомого та двох рухомих елементів

Для створення тривимірних моделей механічного захоплювача у програмі V-REP / CoppeliaSim створимо нову сцену.

У новій сцені будемо користуватися інструментами Primitive Shape для створення захоплювача, який складатиметься з трьох прямокутних паралелепіпедів (кубоїдів).

Використовуючи команди [Add → Primitive Shape → Cuboid] та вводячи наступні параметри, як показано на наступному слайді, можна отримати тривимірну модель основи захоплювача.

На рис. 3 наведене контекстне меню інструмента «Primitive Shape» (Проста форма) під час створення тривимірної моделі основного компонента захоплювача.

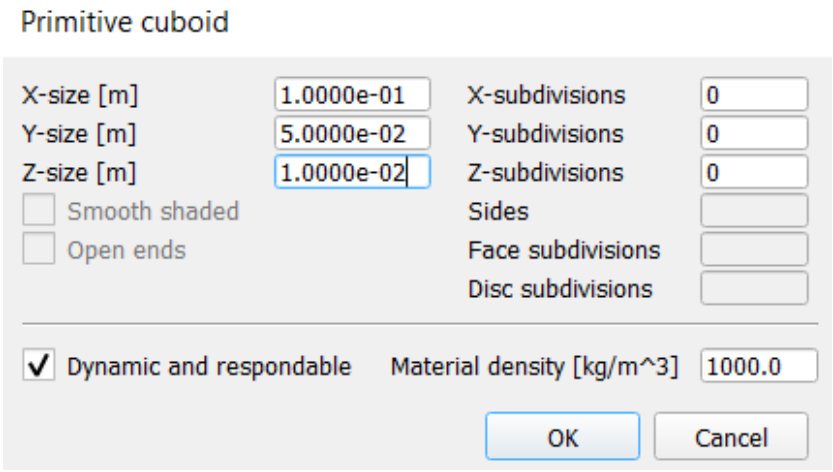


Рис. 3. Контекстне меню інструмента «Primitive Shape»

На початку створемо тривимірну модель основи захоплювача (рис. 4).

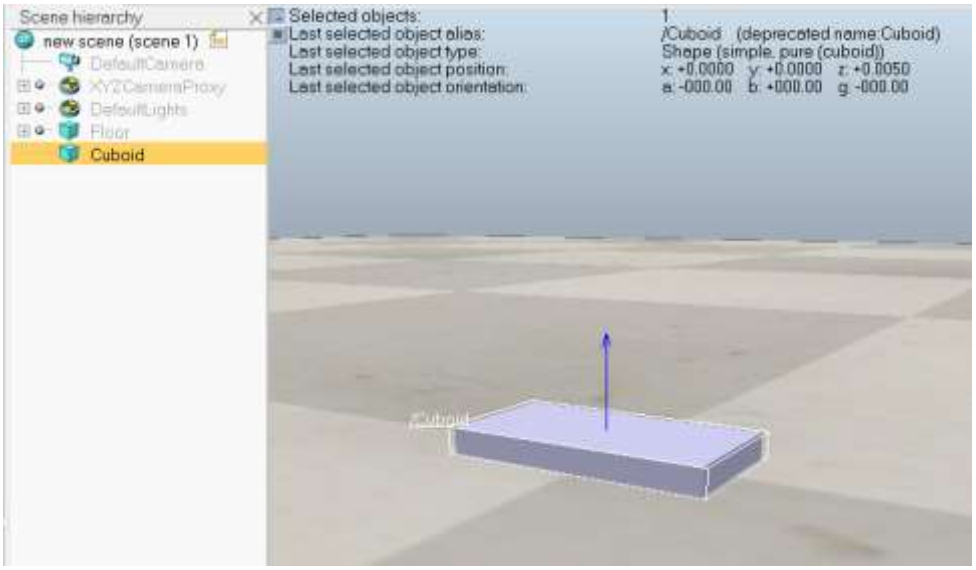


Рис. 4. Тривимірна модель основи захоплювача

Далі необхідно створити другий та третій елемент, тут можна їх зробити однаковими, отже, достатньо створити один елемент та його скопіювати (рис. 5).

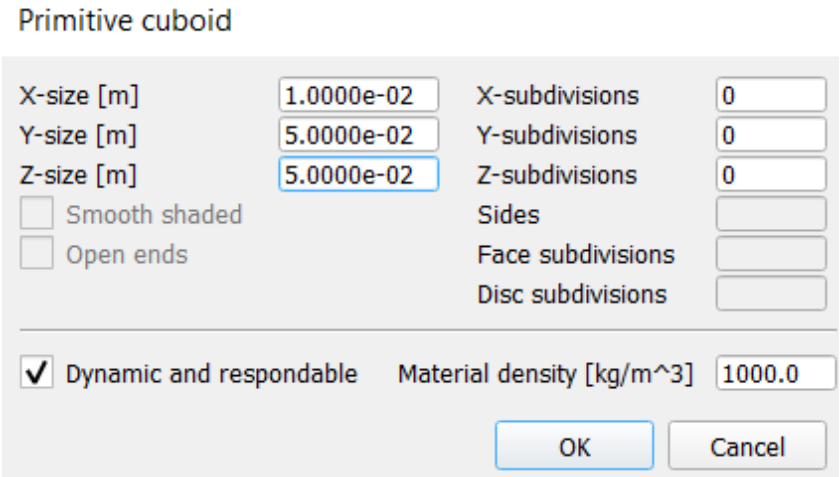


Рис 5. Контекстне меню інструмента "Primitive Shape" (Проста форма) для другого компонента

Отримаємо другий та третій елементи (рис. 6).

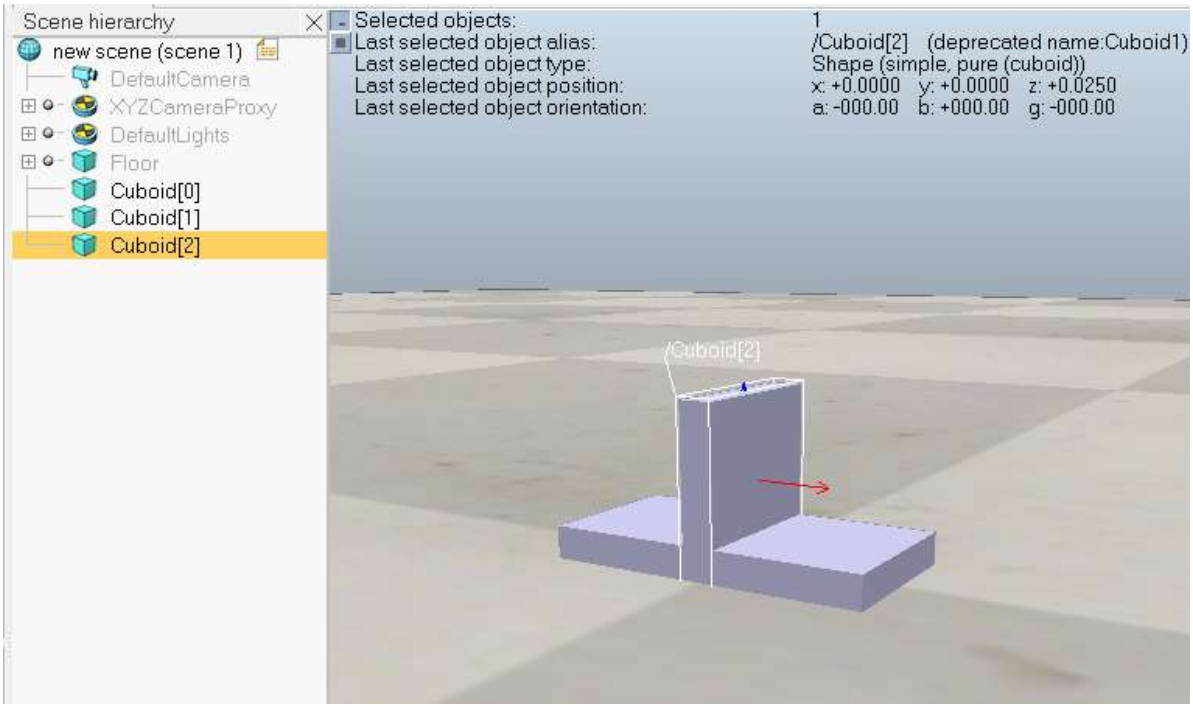


Рис. 6. Другий та третій елементи

Після цього треба задати всім трьом елементам назву відповідно до їх призначення.

За замовчуванням V-REP задає їм імена, що відповідають назві інструмента, за допомогою якого вони створені.

Щоб перейменувати, достатньо навести курсор на ім'я та виконати подвійне натискання правої кнопки миші комп'ютера, встановити нове ім'я та натиснути кнопку введення на клавіатурі (Enter).

Зверніть увагу, що V-REP не підтримує символи кирилиці.

Нові назви повинні бути "Gripper_Base", "Right_side" та "Left_side" (рис. 7).

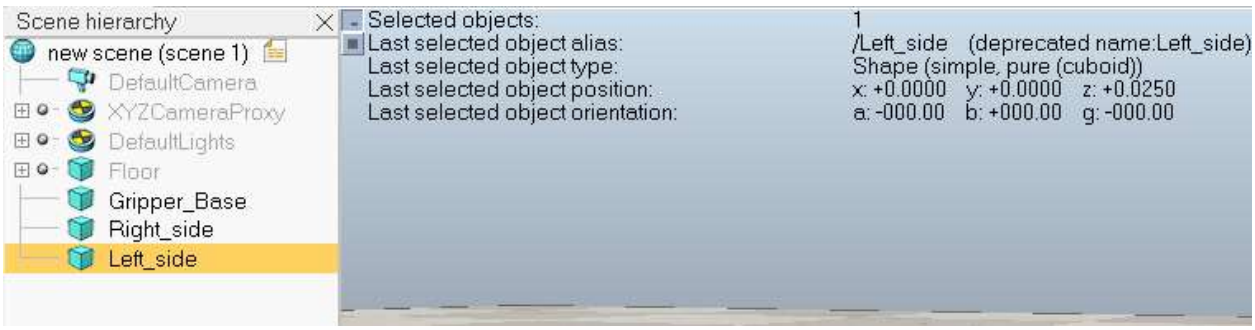


Рис. 7. Встановлення назв елементів

Тепер треба задати коректне розташування для двох елементів механічного захоплення щодо основи, що відповідає вихідному положенню.

Для цього виділяємо у лівому вікні дерева моделі «Left_side» та активуємо інструмент "Object/Item shift" з верхнього основного меню інструментів.

У вікні інструменту, що з'явилося, вибираємо вкладку Position, потім вводимо нові значення розташування елемента по осі «x» і по осі «y», тому що тільки по цих осях поточні значення не відповідають необхідним.

Отримаємо положення, показане на рис.8.

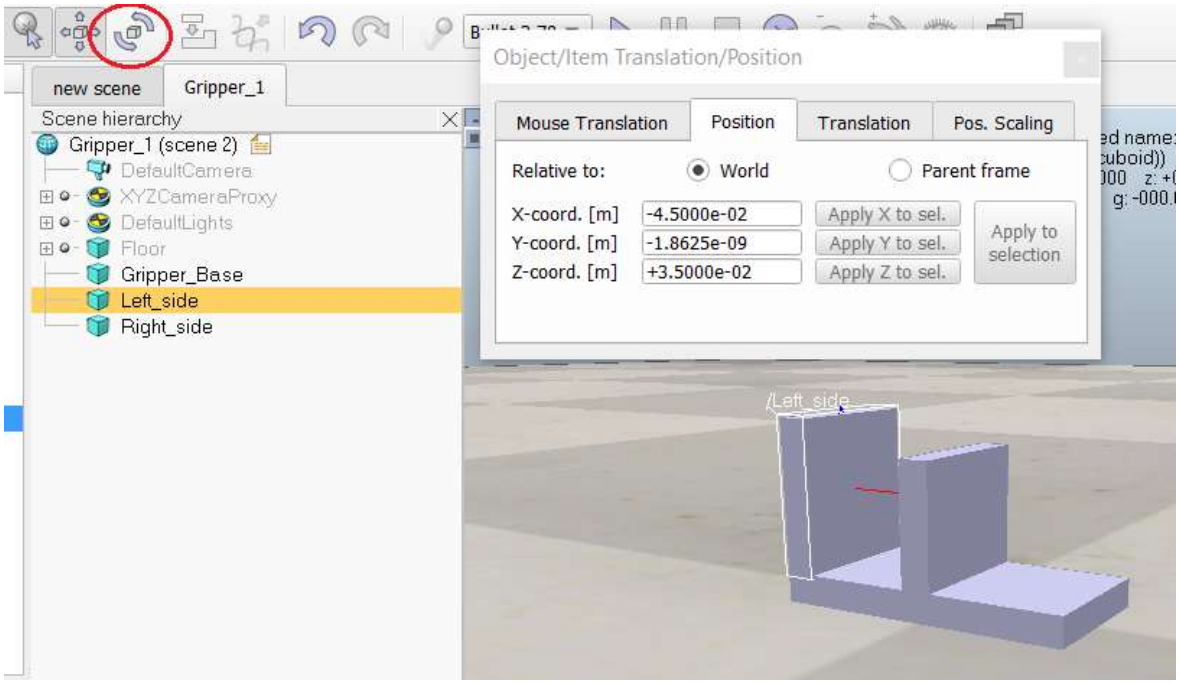


Рис. 8. Переміщення елемента «Left_side»

Потім необхідно вибрати елемент «Right_side» і задати відповідні значення зі зворотним за знаком значенням для осі «x» (рис. 9).

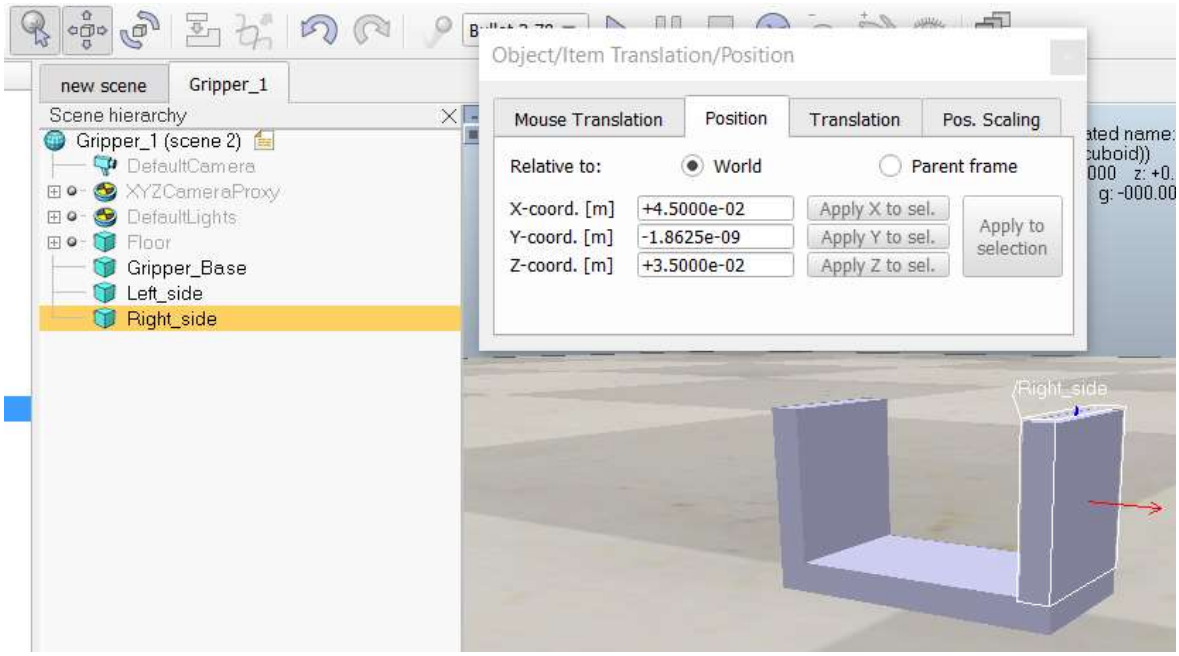


Рис. 9. Переміщення елемента «Right_side»

Далі необхідно встановити механічні з'єднання для наявних елементів за допомогою елементів «Joint - Зчленування», які здійснюють переміщення.

В даному випадку необхідно додати два з'єднання типу «призматичний», які забезпечують поступальне переміщення. Додати їх можна через команди "Add" → "Joint" → "Prismatic".

Поруч із наявними елементами з'явиться новий елемент, який необхідно зменшити, оскільки його розмір заданий за замовчуванням і не дозволяє нам достатньо точно керувати положенням елементів нашого механізму.

Розміри елементів з категорії «Зчленування» використовуються тільки для зручності роботи з ними під час створення симуляції та ніяк не впливають на функціональні параметри елеммента з'єднання.

Змінюємо розміри елеммента з'єднання через редагування властивостей, як показано на наступному слайді, для цього в дереві моделі вибираємо необхідний елемент з'єднання і відкриваємо вікно властивостей з'єднання за допомогою інструмента «Scene Object Properties» (рис. 10).

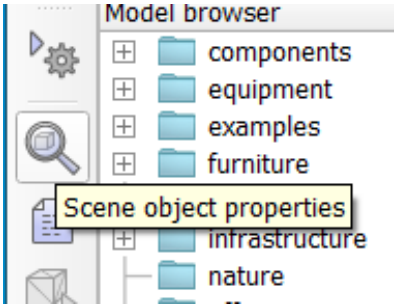


Рис. 10. Інструмент «Scene Object Properties»

На рис. 11 наведене контекстне меню інструмента «Властивості».

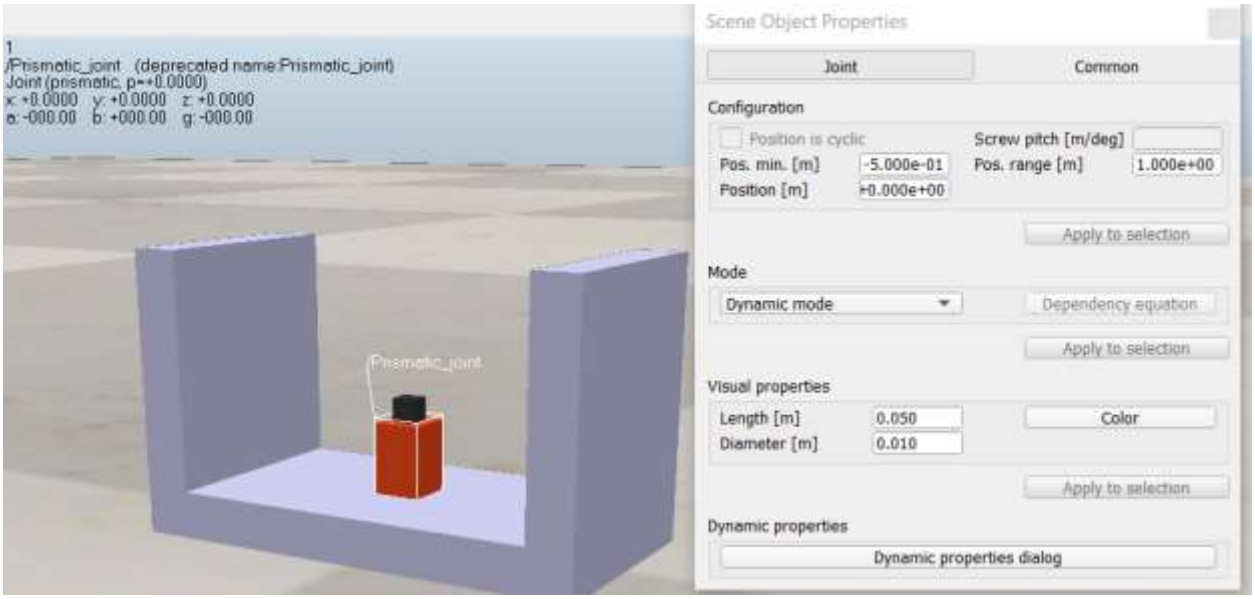


Рис. 11. Контекстне меню інструмента «Властивості»

Необхідно встановити правильне розташування та орієнтацію в просторі для елементів з'єднання, що можна зробити, використовуючи інструменти для зміни положення та орієнтації.

Описуємо докладніше основні прийоми, які не змогли виконати вищеописані операції.

Щоб встановити положення для з’єднання, ми можемо виключити візуальний інтерфейс і за допомогою мишки «перетягнути» елементи на потрібне положення, однак такий підхід не підходить, тому що не гарантує високу точність.

Рекомендується використовувати швидкий спосіб – скопіювати координати інших елементів та вже задавати щодо розташування нового положення.

Можна скопіювати координати іншого елемента, виконавши наступні операції: спочатку вибираємо в дереві моделі об’єкт, в який хочемо скопіювати координати, а за допомогою клавіш «ctrl» вибираємо інший елемент, координати якого ми хочемо скопіювати.

Потім вибираємо інструмент «Object/Item shift» і у вкладці «Position» натискаємо на кнопку «Apply to selection», як це показано на рис. 12.

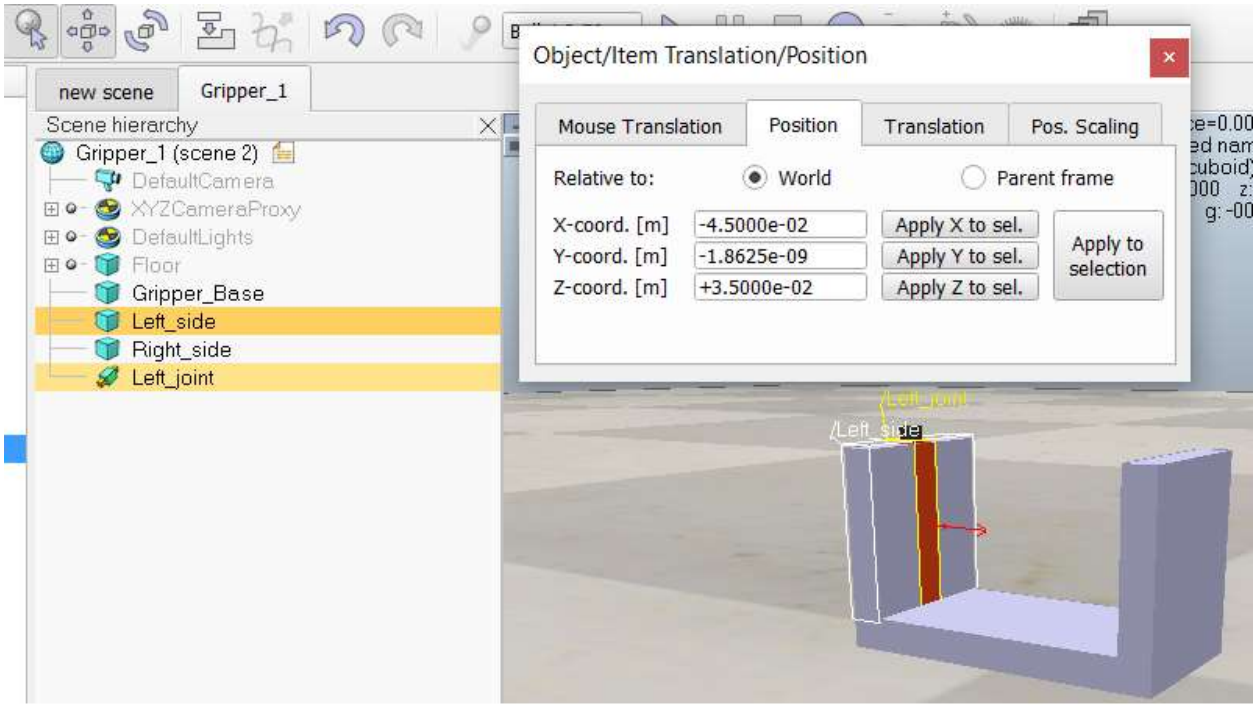


Рис. 12. Копіювання положення елемента «Left_side» у властивості Зчленування

Так само можна скопіювати параметри орієнтації з одного об'єкта в інший, скориставшись інструментом «Object/Item rotate» замість «Object/Item shift».

Однак одного копіювання даних орієнтації та положення буває недостатньо, як і в нашому випадку. У таких випадках необхідно скористатися вкладкою «Translation» для інструменту «Object/Item shift», де можна вказати значення переміщення щодо поточного положення, і після натискання кнопки «Translate» будуть застосовані зазначені зміни.

Схожа функція є й у інструмента «Object/Item rotate», де у вкладці «Rotation» можна вказати кути повороту щодо поточної орієнтації, вибравши вісь обертання, і після натискання кнопки «Rotate selection» отримати результат.

При використанні даного інструменту варто звернути увагу на те, що необхідно вибрати, щодо якої системи координат виконується обертання.

У більшості випадків ця операція виконується щодо системи координат, пов'язаної з елементом, що обертається, і в такому випадку слід у параметрі «Relative to» встановити на «Own Frame».

Таким чином, скориставшись наведеними вище прийомами, ми можемо отримати результат, представлений на рис. 13.

Після того, як задали потрібне положення та орієнтацію для всіх елементів, необхідно встановити зв'язки цих елементів.

Елемент під назвою "Gripper_Base" є основою нашого механізму і, отже, всі зв'язки мають виходити з нього. У V-REP доступні два способи визначення зв'язків.

Перший спосіб найбільш простий і швидкий - виділити та перетягнути один елемент на інший у дереві моделі.

При цьому елементи повинні утворювати деревоподібну структуру, де один елемент стоїть в основі (базовий елемент), а наступні елементи можуть з'єднуватися з основою або іншим елементом однієї з гілок дерева.

Другий спосіб – виділити елемент, який необхідно з'єднати з батьківським елементом, затиснути кнопку ctrl і виділити елемент, який хочемо зробити дочірнім, потім натиснути праву кнопку миші та в меню вибрати Edit, Make last selected Object parent.

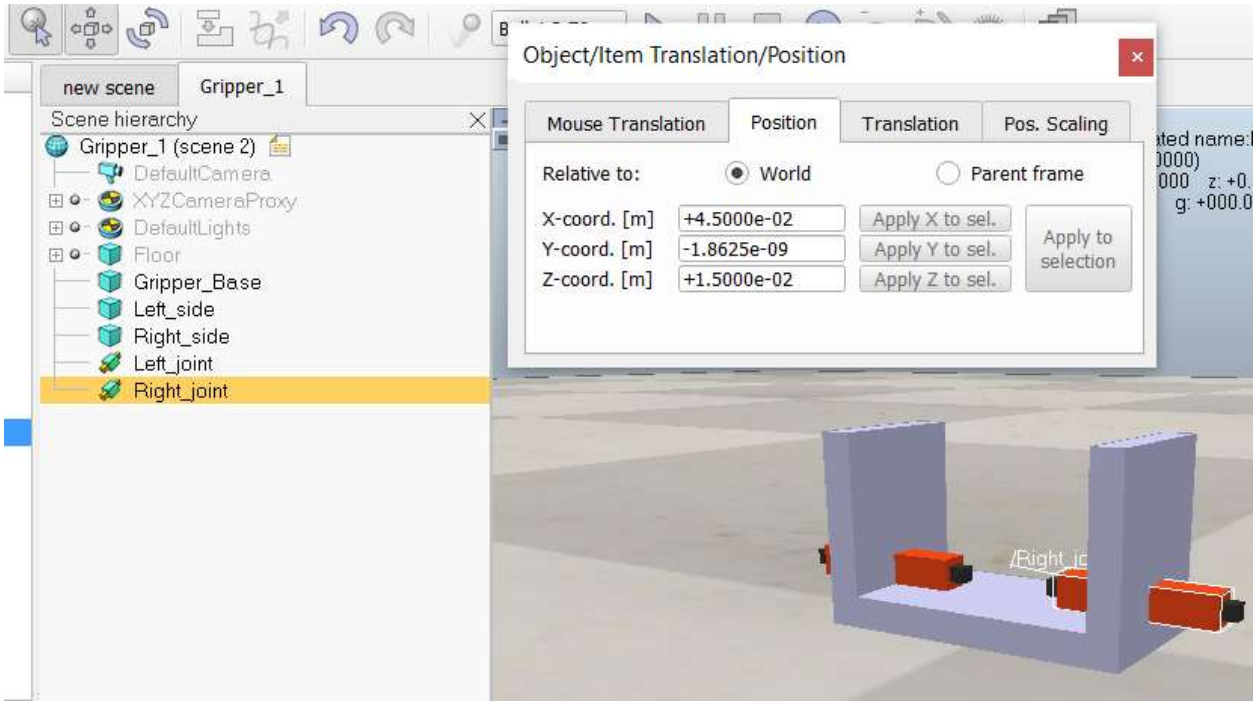


Рис. 13. Зовнішній вигляд елементів сцени з коректними координатами та орієнтацією

Також щодо зв'язків необхідно дотримуватися таке правило: два елементи, моделюючі динаміку, не можна з'єднувати без використання елемента типу «Зчленування», тобто, щоб з'єднати елементи Gripper_Base та Left_side, необхідно між ними в дереві додати "Left_joint".

Так само необхідно задати зв'язок між «Gripper_Base» і «Right_side». Підсумкова деревоподібна структура, яку потрібно отримати, представлена на рис. 14.

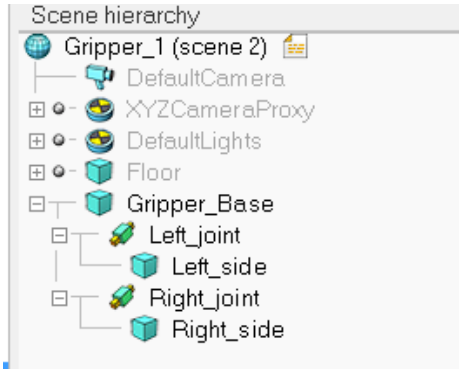


Рис. 14. Підсумкова деревоподібна структура

Далі необхідно увімкнути вбудовані мотори в «Зчленуваннях» "Left_joint" і "Right_joint", задати необхідні налаштування.

Для цього потрібно вибрати елемент у дереві моделі і в лівій панелі активувати інструмент "Scene object properties", далі в контекстному меню інструмента необхідно вибрати "Show dynamic properties dialog" і включити мотор, поставивши відповідну позначку навпроти "Motor enabled".

Після цього активуються рядки номінальної швидкості обертання та максимального моменту – «Target velocity» та «Maximum force» відповідно.

Зважаючи на порівняно малі розміри захоплювача, максимальну силу можна поставити на 5.0e+00 Ньютон і швидкість залишити на 0 м/с., тому що в даній роботі його задаватимемо через скрипт.

Зверніть увагу, що необхідно враховувати напрямок «Призматичного Зчленування»: в залежності від його орієнтації потрібно вказати швидкість для кожного із захоплень із позитивним або негативним знаком.

Визначити знак можна експериментальним способом, провівши симуляцію сценарію та примусово ввімкнувши мотори та додавши значення до швидкості.

Якщо позитивне значення не збігається з необхідним напрямом, необхідно розгорнути «Зчленування» на 180°.

Для перевірки роботи встановимо для обох «Зчленувань» швидкість переміщення +1,0000e-03 (рис. 15).

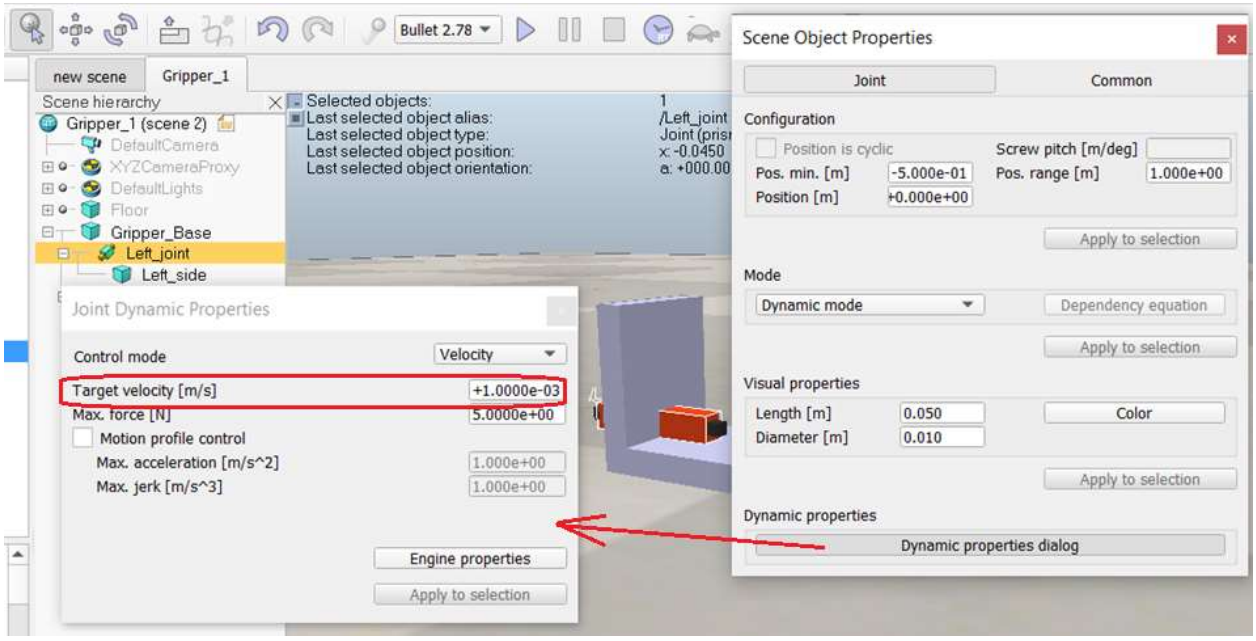


Рис. 15. Встановлення швидкості

Після запуску симуляції отримаємо переміщення ланок захоплювача, наведене на рис. 16.

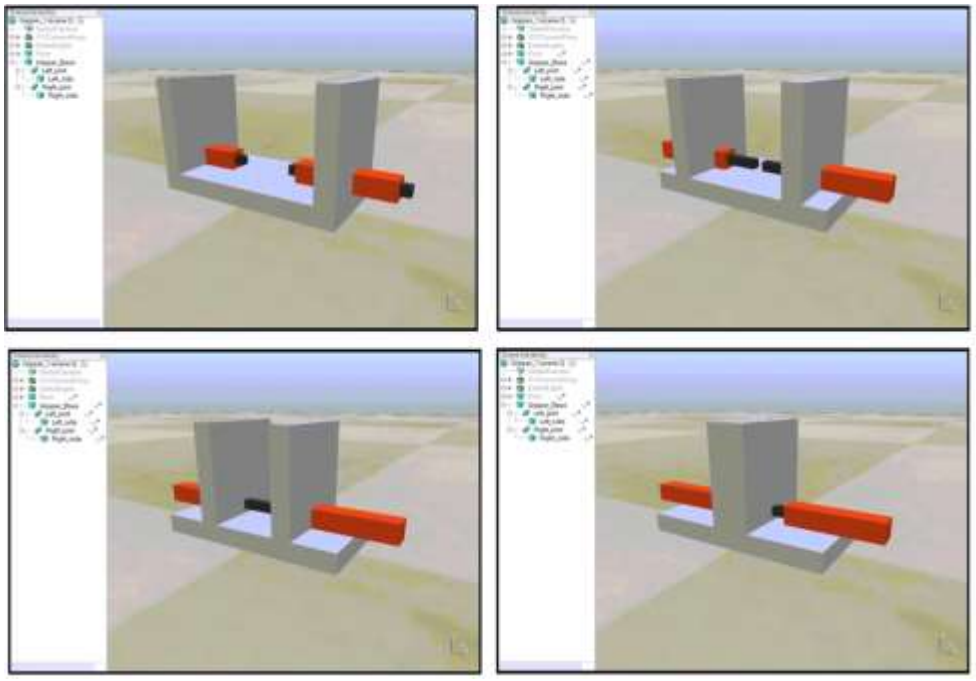


Рис. 16. Переміщення ланок захоплювача

Для перевірки роботи захоплювача додаємо об’єкт захоплення з такими параметрами (рис. 17).

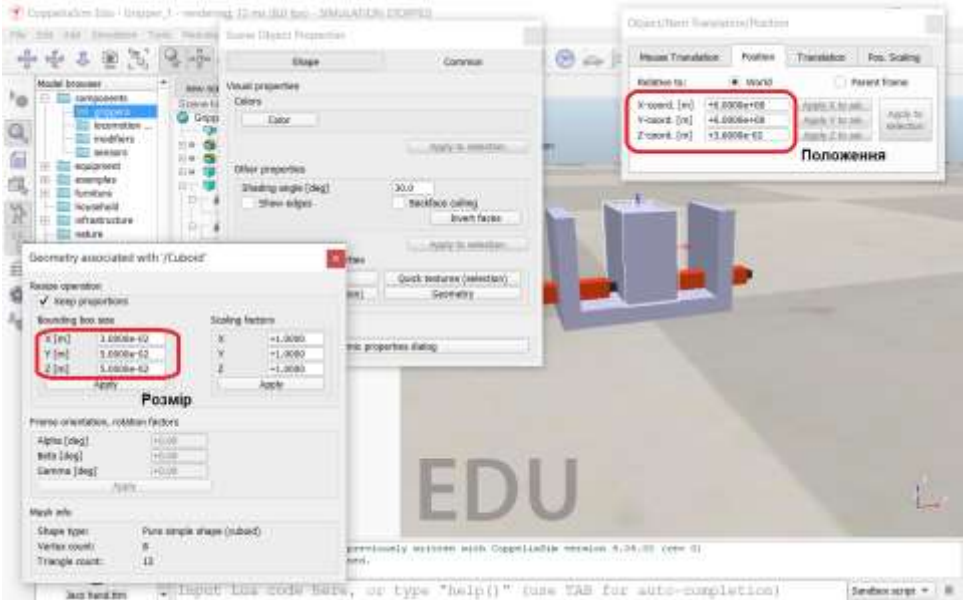


Рис. 17. Об’єкт захоплення

На рис. 18 наведена емуляція захоплення об’єкту.

Рис. 18. Емуляція захоплення об’єкту

На рис. 19 показано, як можна створити відео симуляції роботи захоплювача.

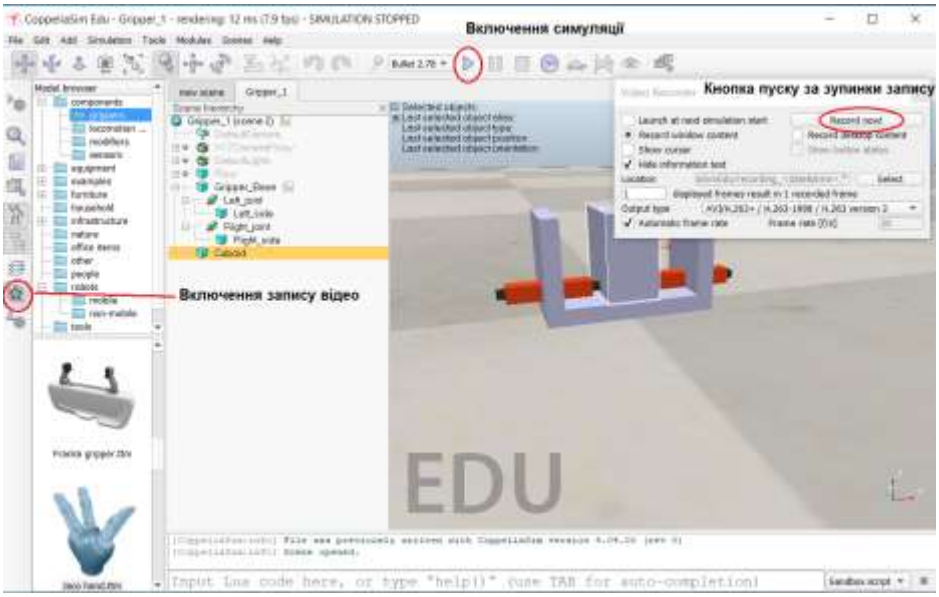


Рис. 19. Створення відео симуляції роботи захоплювача

Після завершення запису буде показано, де знаходиться записане відео (рис. 20).

Рис. 20. Місце знаходження записаного відео

Контрольні питання

1. Назвати, де знаходяться моделі захоплюючих пристроїв.
2. Описати, як визначити елементи захоплювача, який був створений.
3. Назвати, як встановити розміри елементів.
4. Визначити, як встановити положення елементів.
5. Описати, як здійснити об’єднання елементів.
6. Визначити, де знаходяться засоби переміщення.
7. Описати, як встановити швидкість руху.

Завдання

1. За допомогою V-REP створити захоплюючий пристрій, як було показано вище, змінівши згідно з вказаним варіантом коефіцієнту пропорційності.
2. Швидкість пересування рухомих елементів змінити згідно з вказаним варіантом коефіцієнту пропорційності.
3. Зробити відео симуляції роботи захоплювача.

Варіант коефіцієнту пропорційності

Варіант	1	2	3	4	5	6	7	8	9	10
Коефіцієнт пропорційності	1,5	2,1	2,0	1,3	2.7	1,8	2,3	2,2	1,4	1,6

Заняття 7. Конструювання засобів ручного керування переміщенням ланок промислового робота

У каталозі моделей роботів платформи V-REP (CoppeliaSim) можна знайти модель промислового робота ABB IRB 140, яка має засоби ручного керування (рис. 1).

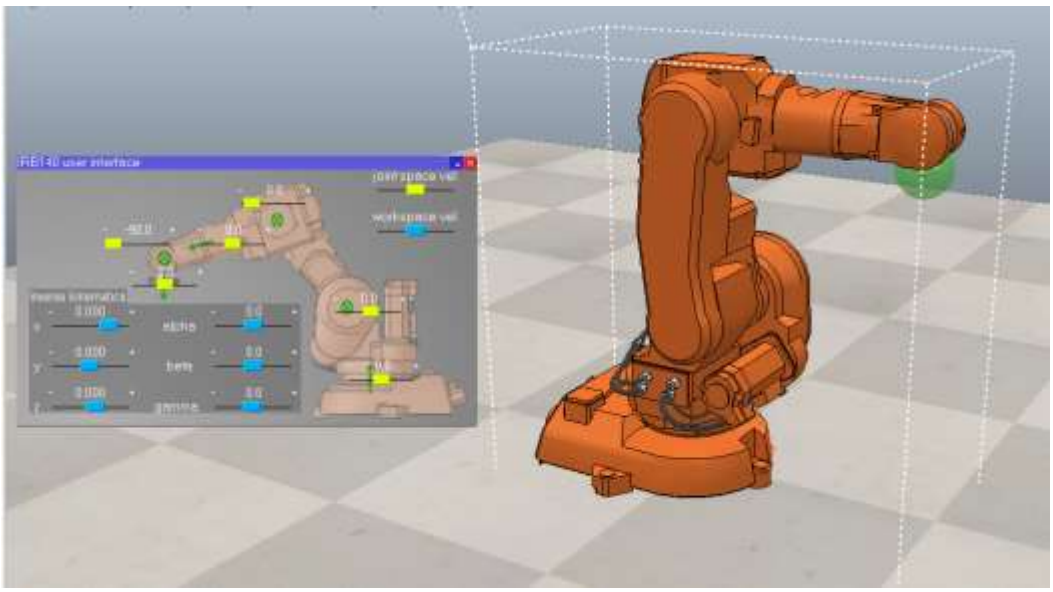


Рис. 1. Модель промислового робота ABB IRB 140, яка має засоби ручного керування

Створимо спрощену модель ручного керування за допомогою слайдерів.

Для цього у каталозі моделей визначимо робот ABB IRB 140 і встановимо його на сцені (рис. 2).

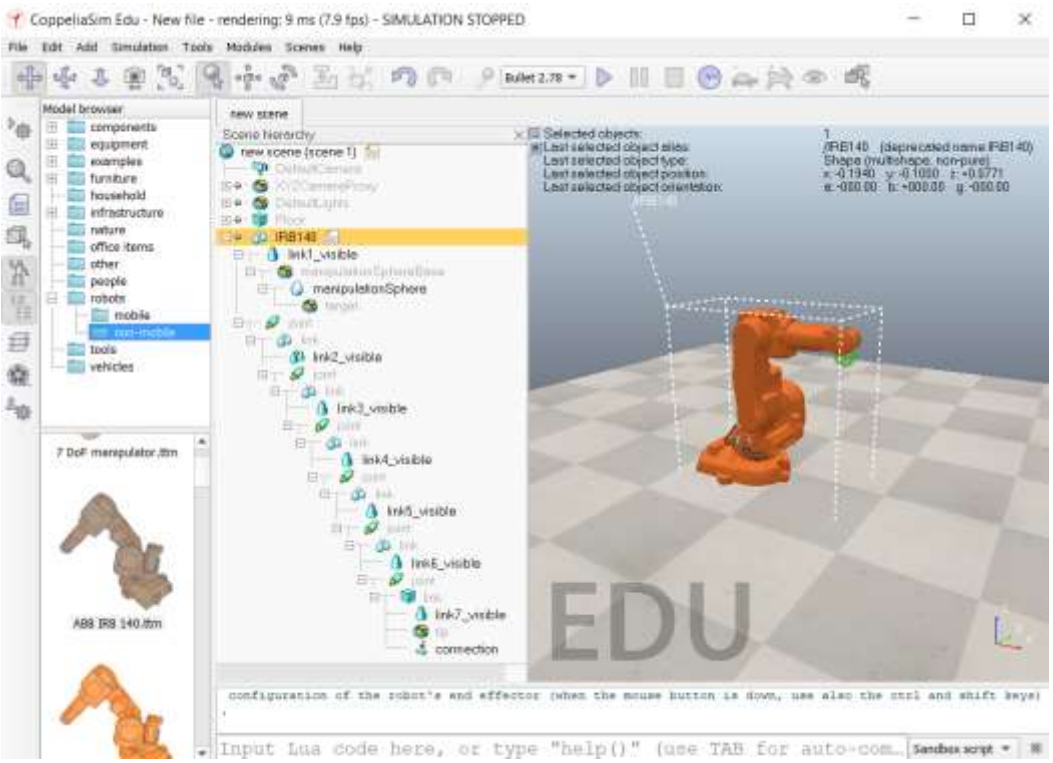


Рис. 2. робот ABB IRB 140

Далі видалимо непотрібні об'єкти з ієрархії сцени, залишивши ті, що наведені на рис. 3.

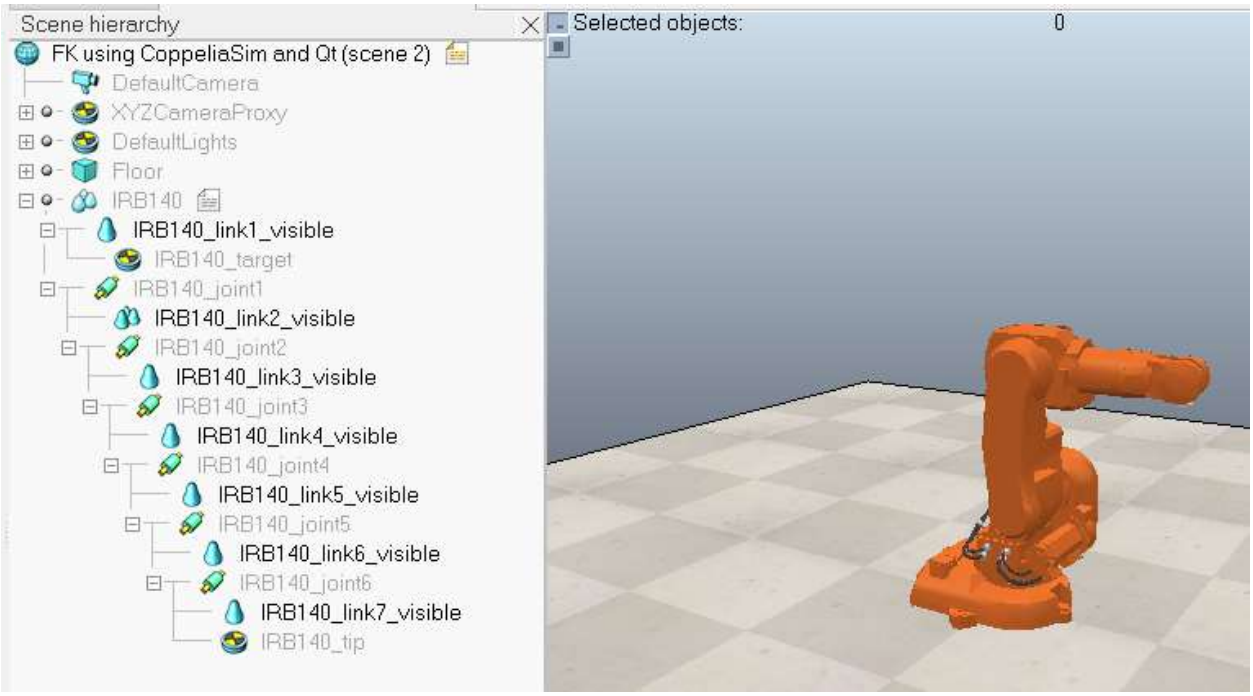


Рис. 3. Ієрархія сцени

Розглянемо структуру безпоточного дочірнього скрипта управління роботом (мова програмування Lua).

Lua – мова програмування розширень, розроблена для підтримки загального процедурного програмування з можливістю опису даних.

Загальна структура скрипту управління, як було розглянуто на лекціях, має вигляд, представлений на рис. 4.

```
1  function sysCall_init()
2      -- do some initialization here
3  end
4
5  function sysCall_actuation()
6      -- put your actuation code here
7  end
8
9  function sysCall_sensing()
10     -- put your sensing code here
11 end
12
13 function sysCall_cleanup()
14     -- do some clean-up here
15 end
16
```

Рис. 4. Загальна структура скрипту управління

Скрипт у цьому прикладі виконує чотири основні функції, а саме:

sysCall_init() – функція ініціалізації.

Функції ініціалізації викликається під час запуску симулювання. Вона викликається лише один раз і зазвичай використовується для ініціалізації змінних, інтерфейсу користувача і т.д. Функція sysCall_init() завжди викликається до sysCall_actuation().

sysCall_actuation() – функція активації.

Функція активації виконується циклічно у процесі симулювання. Є основною функцією управління роботом.

sysCall_sensing() - Функція датчиків.

Дана функція призначена для опитування стану датчиків під час симулювання, і виконується на кожному етапі симулювання.

sysCall_cleanup() – функція очищення.

Ця функція викликається при зупинці симулювання. Функція відповідає відновлення початкової конфігурації робота, очищення станів датчиків тощо.

Відкриваємо скрипт моделі робота, як наведено на рис. 5.

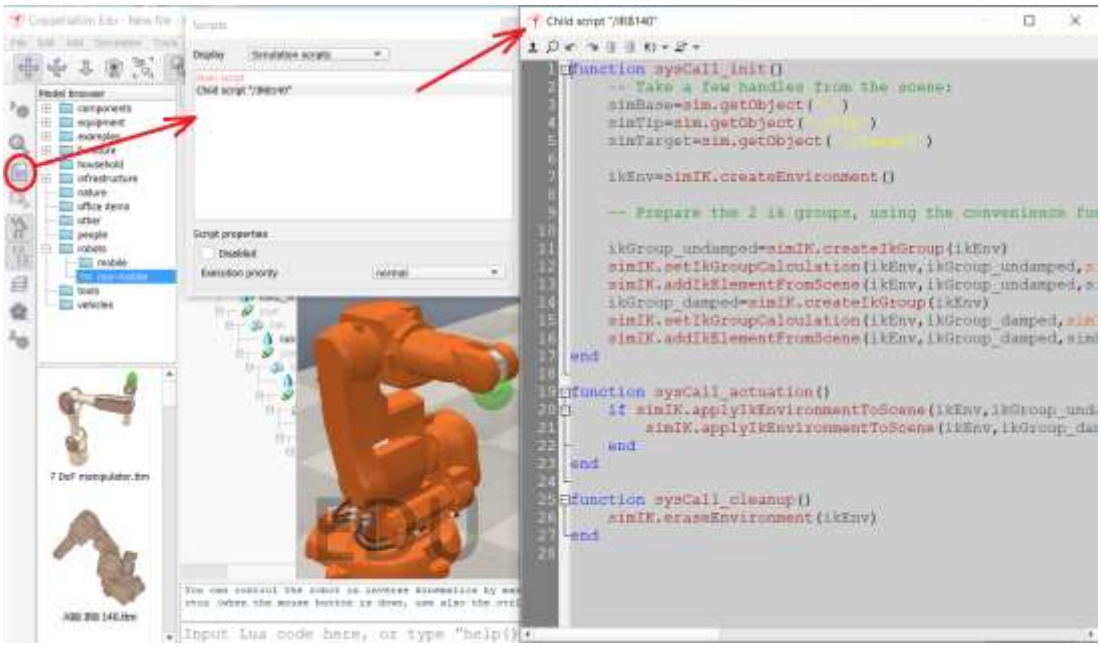


Рис. 5. Скрипт моделі робота

Далі приступимо до розробки керуючого скрипта (повний код скрипта наведений у додатку).

Для функції sysCall_init() створимо наступний код:

```
function sysCall_init()
    simJoints={0,0,0,0,0,0}
    for i=1,6,1 do
        simJoints[i]=sim.getObjectHandle('IRB140_joint'..i)
    end
    xml = '<ui title="Forward Kinematics" closeable="true" on-close="closeEventHandler"
resizable="false">'..[[
    <label text="Forward Kinematics for ABB IRB 140 using CoppeliaSim and Qt"/>
    <group layout="hbox">
    <label text="J1" id="11" />
    <hslider id="21" minimum="-500" maximum="500" on-change="sliderChange" />
    <edit value="0" id="31" on-editing-finished="jointEntry" />
    </group>
    <group layout="hbox">
    <label text="J2" id="12"/>
    <hslider id="22" minimum="-500" maximum="500" on-change="sliderChange" />
    <edit value="0" id="32" on-editing-finished="jointEntry" />
    </group>
    <group layout="hbox">
    <label text="J3" id="13"/>
    <hslider id="23" minimum="-500" maximum="500" on-change="sliderChange" />
```

```
    <edit value="0" id="33" on-editing-finished="jointEntry" />
    </group>
    <group layout="hbox">
    <label text="J4" id="14"/>
    <hslider id="24" minimum="-500" maximum="500" on-change="sliderChange" />
    <edit value="0" id="34" on-editing-finished="jointEntry" />
    </group>
    <group layout="hbox">
    <label text="J5" id="15"/>
    <hslider id="25" minimum="-500" maximum="500" on-change="sliderChange" />
    <edit value="0" id="35" on-editing-finished="jointEntry" />
    </group>
    <group layout="hbox">
    <label text="J6" id="16"/>
    <hslider id="26" minimum="-500" maximum="500" on-change="sliderChange" />
    <edit value="0" id="36" on-editing-finished="jointEntry" />
    </group>
    </ui>
    ]]
    ui=simUI.create(xml)
end
```

Для функції sysCall_cleanup() створимо такий код:

```
function sysCall_cleanup()
simUI.destroy(ui)
end
```

При зупинці симулювання, за допомогою команди simUI.destroy(ui) знищуємо інтерфейс користувача.Також додамо ще одну функцію closeEventHandler():

```
function closeEventHandler()
simUI.hide(ui)
end
```

Функція closeEventHandler() відповідає за обробку подій кнопки «закрити» інтерфейсу користувача.Функція обробки подій слайдерів

```
function sliderChange(ui,id,newVal)
   for i=1,6,1 do
         if (id==20+i) then
        simUI.setLabelText(ui,10+i,string.format('J' ..i))
        simUI.setEditValue(ui,30+i,string.format(newVal))
        sim.setJointPosition(simJoints[i],newVal*math.pi/180)
                break
         end
   end
end
```

Функція обробки подій редагованих полів має виглад:

```
function jointEntry(ui,id,newVal)
   if (tonumber(newVal)==nil) then
         print("Error: could not convert number")
         return
   end
   for i=1,6,1 do
         if (id==30+i) then
                simUI.setLabelText(ui,10+i,string.format('J'..i))
        simUI.setEditValue(ui,30+i,string.format(newVal))
        sim.setJointPosition(simJoints[i],newVal*math.pi/180)
                break
         end
   end
end
```

Запустимо симуляцію. Тепер промисловим роботом можна керувати за допомогою інтерфейсу користувача. Вікно інтерфейсу користувача зображено на рисунку.

Задаючи кути повороту в градусах для кожного шарніра за допомогою слайдера або шляхом внесення кута повороту у віконцях справа від слайдера вможна візуально спостерігати зміну положення його ланок у координатах XYZ, а вирішивши пряме завдання кінематики за відомою кінематичною схемою перевірити результати практичним шляхом (рис. 6).

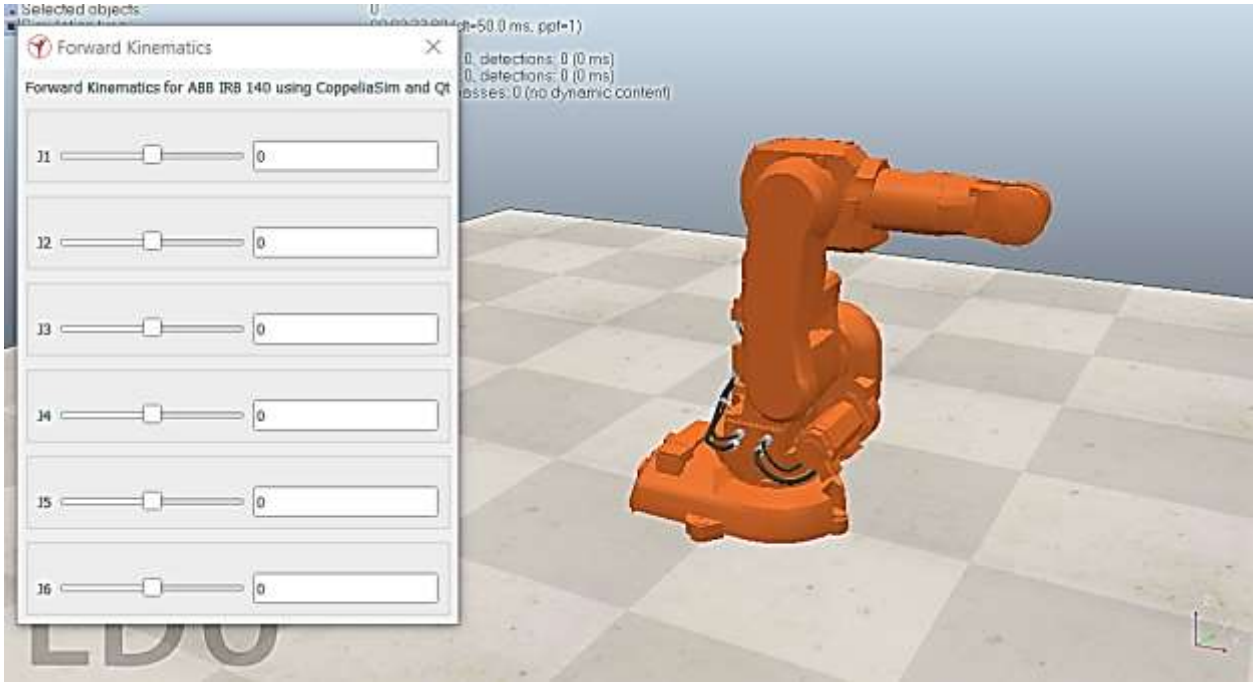


Рис. 6. Вікно інтерфейсу користувача

Діапазони переміщення кута повороту окремих ланок від -500º до 500º встановлюються у функції function sysCall_init():

```
<group layout="hbox">
<label text="J1" id="11" />
<hslider id="21" minimum="-500" maximum="500" on-change="sliderChange" />
<edit value="0" id="31" on-editing-finished="jointEntry" />
</group>
<group layout="hbox">
<label text="J2" id="12"/>
<hslider id="22" minimum="-500" maximum="500" on-change="sliderChange" />
<edit value="0" id="32" on-editing-finished="jointEntry" />
</group>
<group layout="hbox">
<label text="J3" id="13"/>
<hslider id="23" minimum="-500" maximum="500" on-change="sliderChange" />
<edit value="0" id="33" on-editing-finished="jointEntry" />
</group>
<group layout="hbox">
<label text="J4" id="14"/>
<hslider id="24" minimum="-500" maximum="500" on-change="sliderChange" />
<edit value="0" id="34" on-editing-finished="jointEntry" />
</group>
<group layout="hbox">
<label text="J5" id="15"/>
<hslider id="25" minimum="-500" maximum="500" on-change="sliderChange" />
<edit value="0" id="35" on-editing-finished="jointEntry" />
</group>
```

Перетворення кута повороту у радіани здійснюється у функції function sliderChange(ui,id,newVal):

sim.setJointPosition(simJoints[i],newVal*math.pi/180)

Контрольні питання

1. Описати, які переміщення здійснює модель робота ABB IRB 140.
2. Назвати, яку мову використовує скрипт симуляції робота ABB IRB 140.
3. Описати, як треба змінити ієрархію сцени для внесення змін.
4. Назвати, які основні функції виконує скрипт у цьому прикладі.
5. Описати, як здійснюється ручне керування під час симуляції .
6. Визначити, як відкрити скрипт моделі робота.
7. Назвати, як і зміни треба зробити у скрипті для того, щоб вантаж залишився на конвеєрі.

Завдання

1. Використовуючи наведений приклад ручного керування сконструювати робот з ручним керуванням та перевірити його роботу за допомогою симулятора.
2. Виходячи з реально можливого повороту окремих ланок встановити відповідні обмеження діапазонів у скрипті керування роботом.

Заняття 8. Застосування графіків у CoppeliaSim на прикладі двоколісного мобільного робота Pioneer 3-DX

Результати робототехнічного дослідження повинні бути зведені в зручні форми подання, однією з яких є графіки.

Програмний комплекс CoppeliaSim дозволяє будувати різноманітні графіки.

Починаючи з версії 4.2.0, розробники виключили з програми "графобудівник" – графічний інтерфейс для побудови графіків. Тепер усі графіки будуються скриптингом. Обґрунтовуючи це тим, що це дозволяє набагато більш гнучко та точно керувати графіками.

Побудуємо графік траєкторії руху мобільного робота pioneer p3dx [7]. Для цього на панелі меню виберіть Add --> Graph. Доступ до його основних властивостей та їх налаштування здійснюються у діалоговому вікні графіка. Про об'єкт Graph можна прочитати ТУТ.

Додаємо на сцену модель робота pioneer p3dx. Для цього в оглядачі моделей у дереві robots виберемо mobile і перетягуємо робота pioneer p3dx на сцену.

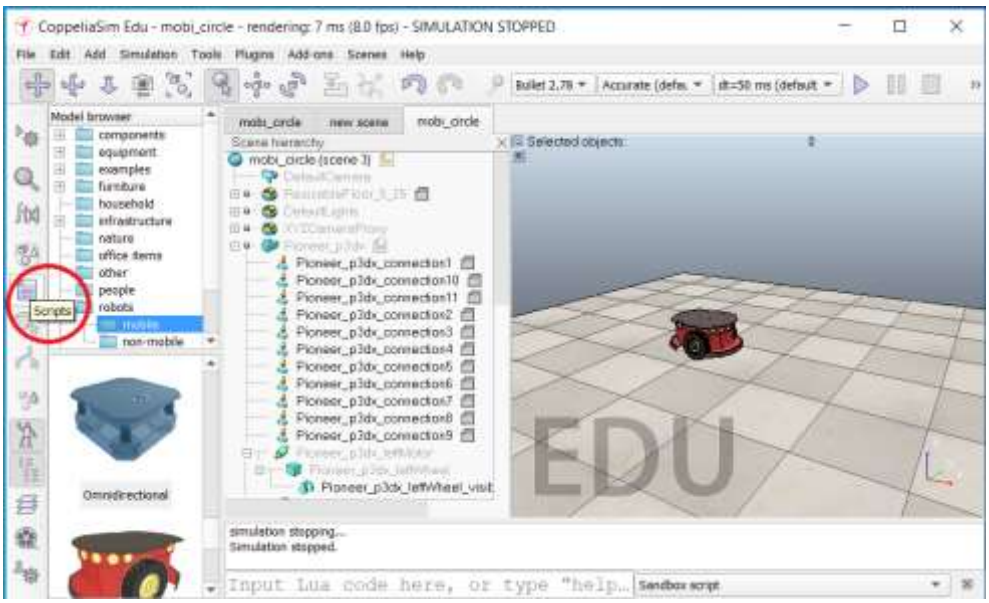


Рис. 1. Робот pioneer p3dx

Якщо зараз натиснути кнопку Play (Старт сцени), то робот поїде по прямій, логіка його роботи міститься у прикріпленому скрипті.

Відредагуємо скрипт таким чином, щоб замість прямолінійного руху робот здійснював кругове.

Для цього функції sysCall_actuation() швидкість правого колеса збільшимо вдвічі.

```
vRight=v0*2
```

Далі наведена послідовність дій для створення графіків.

Обираємо робот pioneer p3dx, як показано на рис. 1.

Знаходимо скрипт симуляції, як показано на рис. 2.

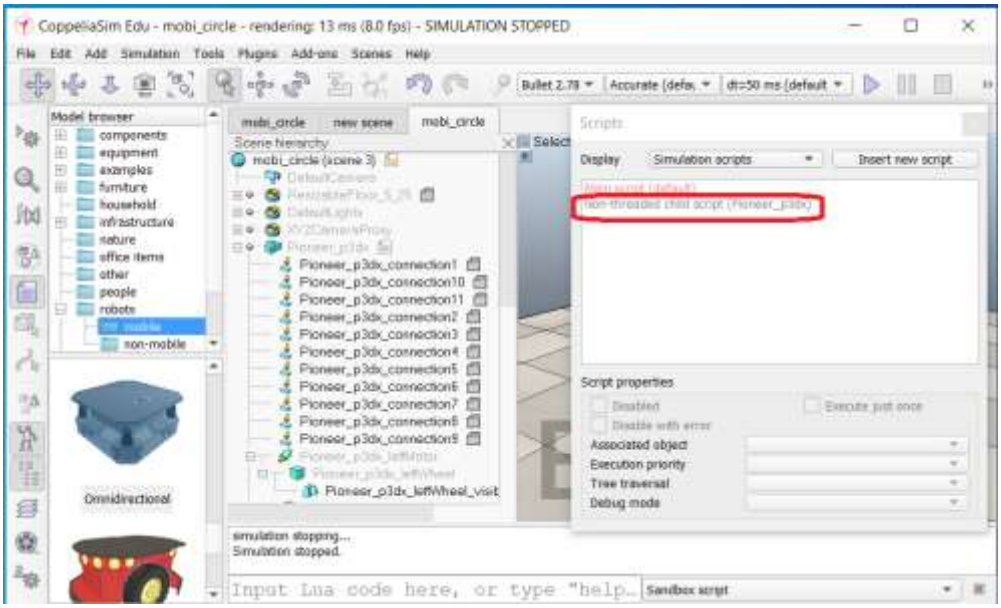


Рис. 2. Визначення скрипту симуляції

Відкриваємо скрипт та вносимо вказане доповнення (рис. 3).

Non-threaded child script (Pioneer_p3dx)

```
function sysCall_init()
    usensors={-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1}
    for i=1,16,1 do
        usensors[i]=sim.getObjectHandle("Pioneer_p3dx_ultrasonicSensor"..i)
    end
    motorLeft=sim.getObjectHandle("Pioneer_p3dx_leftMotor")
    motorRight=sim.getObjectHandle("Pioneer_p3dx_rightMotor")
    noDetectionDist=0.5
    maxDetectionDist=0.2
    detect={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}
    braitenbergL={-0.2,-0.4,-0.6,-0.8,-1,-1.2,-1.4,-1.6, 0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0}
    braitenbergR={-1.6,-1.4,-1.2,-1,-0.8,-0.6,-0.4,-0.2, 0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0}
    v0=2
end
-- This is a very simple EXAMPLE navigation program, which avoids obstacles using the Braitenberg


function sysCall_cleanup()

end

function sysCall_actuation()
    for i=1,16,1 do
        res,dist=sim.readProximitySensor(usensors[i])
        if (res>0) and (dist<noDetectionDist) then
            if (dist<maxDetectionDist) then
                dist=maxDetectionDist
            end
            detect[i]=1-((dist-maxDetectionDist)/(noDetectionDist-maxDetectionDist))
        else
            detect[i]=0
        end
    end

    vLeft=v0
    vRight=v0*2

    for i=1,16,1 do
        vLeft=vLeft+braitenbergL[i]*detect[i]
        vRight=vRight+braitenbergR[i]*detect[i]
    end

    sim.setJointTargetVelocity(motorLeft,vLeft)
    sim.setJointTargetVelocity(motorRight,vRight)
end
```

Рис. 3. Скрипт симуляції з доповненням

Переконаємося у круговій траєкторії руху мобільного робота, запустивши симуляцію.

Далі в скрипті мобільного робота, що містить логіку роботи, у функції sysCall_init() додамо наступний код:

```
graph=sim.getObjectHandle('Graph')
objectHandle=sim.getObjectHandle('Pioneer_p3dx')
objectPosX=sim.addGraphStream(graph,'x','m',1)
objectPosY=sim.addGraphStream(graph,'y','m',1)
sim.addGraphCurve(graph,'object pos x/y',2,{objectPosX,objectPosY},{0,0},'m by m')
```

де 'Graph' та 'Pioneer_p3dx' це імена об'єктів з ієрархії сцени.

У тілі скрипта додамо функцію `sysCall_sensing()`.

```
function sysCall_sensing()
end
```

До функції sysCall_sensing() додамо наступний код:

```
pos=sim.getObjectPosition(objectHandle,-1)
sim.setGraphStreamValue(graph,objectPosX,pos[1])
sim.setGraphStreamValue(graph,objectPosY,pos[2])
```

Очистимо графік після закінчення процесу симулювання. Для цього до функції sysCall_cleanup() додамо наступний рядок:

```
sim.resetGraph(graph)
```

і запустимо процес симулювання (рис. 4).

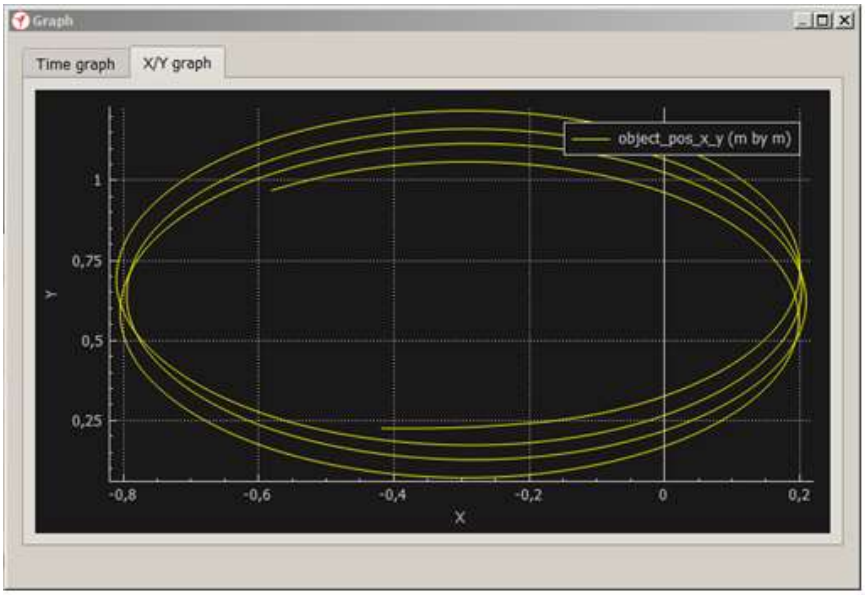


Рис. 4. Графік траєкторіїї руху

Точність позиціонування обумовлена впливом фізики на мобільний робот.

Розглянемо, як створюється крива руху механізму у просторі в залежності від часу.

Наведемо скрипт до первісного вигляду, залишивши лише збільшену вдвічі швидкість правого колеса.

Далі в скрипті, що містить логіку роботи мобільного робота, у функції sysCall_init() додамо наступний код:

```
graph=sim.getObjectHandle('Graph')
base=sim.getObjectHandle('Pioneer_p3dx')
x=sim.addGraphStream(graph, 'x', 'm')
```

Побудуємо залежність x від часу t.

Додамо функцію sysCall_sensing().

```
function sysCall_sensing()
end
```

До функції sysCall_sensing() додамо наступний код:

```
p=sim.getObjectPosition(base,-1)
sim.setGraphStreamValue(graph, x, p[1])
```

Очистимо графік після закінчення процесу симулювання. Для цього до функції очищення sysCall_cleanup() додамо наступний рядок:

```
sim.resetGraph(graph)
```

і запустимо процес симулювання (рис. 5).

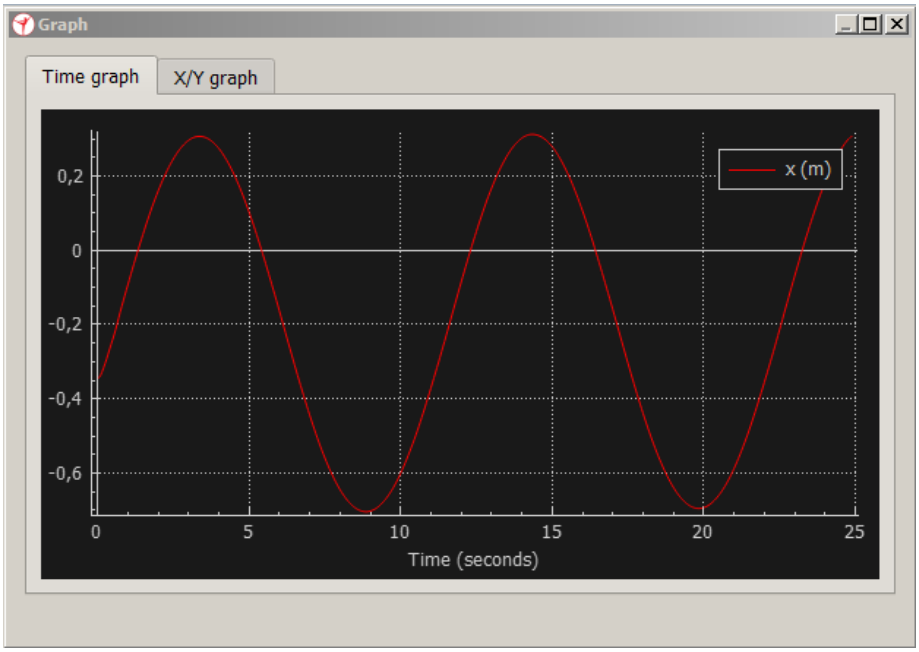


Рис. 5. Графік залежності x від часу t

Додамо другу залежність від часу t.

Для цього до функції ініціалізації sysCall_init() додамо наступний рядок:

```
y=sim.addGraphStream(graph, 'y', 'm')
```

У функції sysCall_sensing() додамо наступний рядок:

```
sim.setGraphStreamValue(graph, y, p[2])
```

p[3] - це відповідно координата z.

Запустимо процес симулювання. Отримаємо графік, зображений на рис. 6.

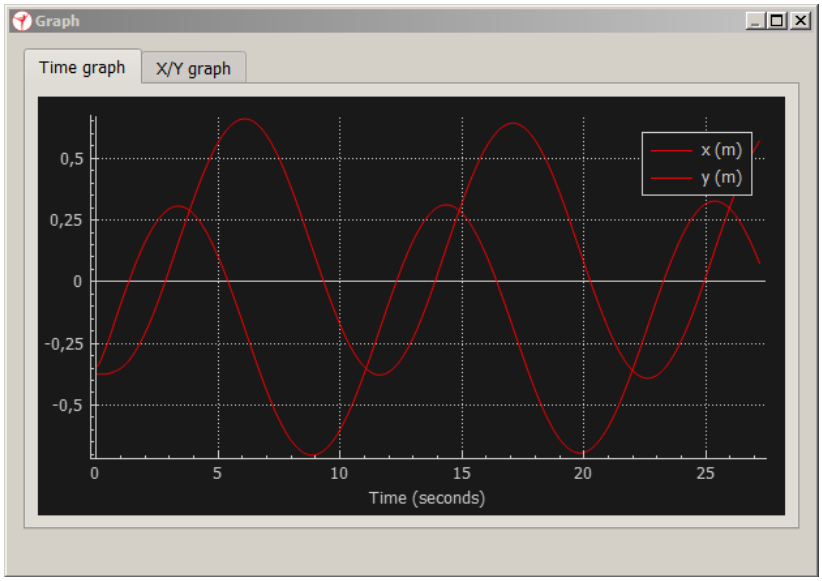


Рис. 6. Графік залежності x, y від часу t

Через те, що криві одного кольору з графіка складно зрозуміти де яка змінна. Тому введемо такі зміни в описі змінних до функцій sysCall_init():

```
x=sim.addGraphStream(graph, 'x', 'm', 0, {1,0,0})
y=sim.addGraphStream(graph, 'y', 'm', 0, {0,1,0})
```

де *{1,0,0}* и *{0,1,0}* це значення кольору в форматі RGB.

Запустимо процес симулювання. Отримаємо графік зображений на рис. 7.

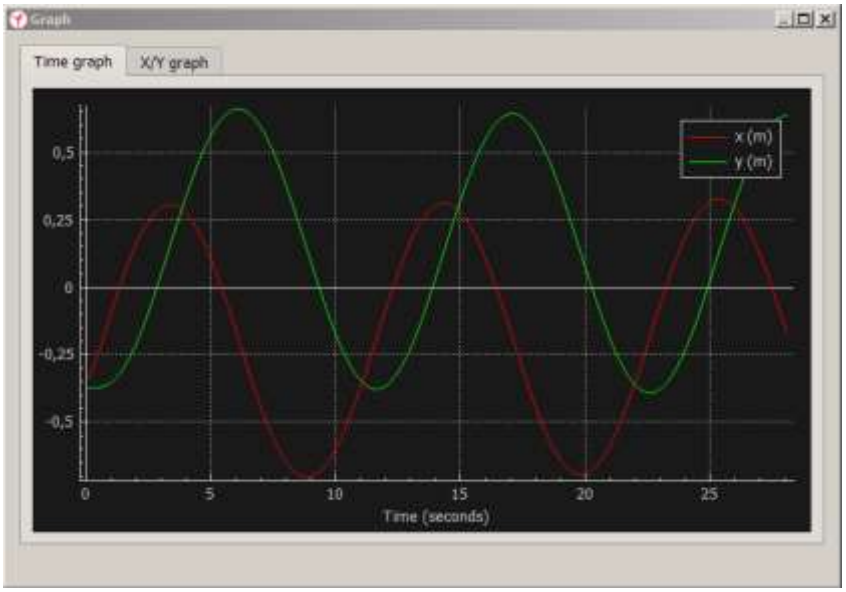


Рис. 7. Графік залежності x, y від часу t

Для побудови графіка залежності швидкості від часу створимо інтерфейс користувача.

CoppeliaSim пропонує створення інтерфейсу (custom user interfaces), що настроюється, за допомогою вбудованого плагіна Qt. При створенні інтерфейсу

користувача використовується спеціальний XML синтаксис. Починаючи з версії 4.2.0 розробники повністю виключили можливість створення інтерфейсу користувача за допомогою "OpenGl-based custom UI".

Приклад синтаксису плагіна Qt найбільш добре відображений у навчальній сцені customUI.ttt папки scenes, що знаходиться в корені, в місці установки програми.

Загалом видалимо та знову додамо на сцену модель робота pioneer p3dx.

Нам потрібно написати трохи коду.

У функції sysCall_init() значення зміною v0=2 змінимо на нуль:

```
v0=0
```

Наприкінці функції sysCall_init() додамо наступний код:

```
xml = '<ui title="Speed" closeable="false" resizeable="false" activate="false">'..[[
<hslider minimum="0" maximum="1000" on-change="speedChange_callback" id="1"/>
</ui>
]]
ui=simUI.create(xml)
```

У функції sysCall_cleanup() додамо наступний рядок:

```
simUI.destroy(ui)
```

У тілі скрипта вставимо наступну функцію:

```
function speedChange_callback(ui,id,newVal)
    v0=newVal*2/1000
end
```

Запустимо симулювання. Тепер швидкістю робота можна керувати (рис. 8).

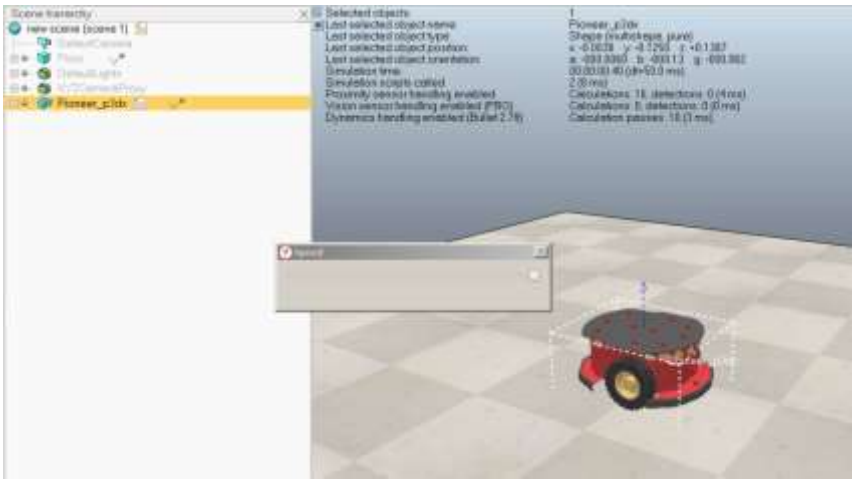


Рис. 8. Симулювання робота pioneer p3dx

Реалізуємо керування роботом за допомогою двох повзунків, щоб один з них керував швидкістю лівого колеса, а інший – правого.

Наведемо скрипт до первісного вигляду.

Далі у функції sysCall_init() видалимо змінну v0=2 і ініціалізуємо наступні змінні:

```
vl=0
vr=0
```

Наприкінці тіла функції sysCall_init() вставимо наступний код:

```
xml = '<ui title="Speed" closeable="false" resizeable="false" activate="false">'..[[
<hslider minimum="0" maximum="1000" on-change="speedChange_callbackL" id="1"/>
<hslider minimum="0" maximum="1000" on-change="speedChange_callbackR" id="2"/>
</ui>
]]
ui=simUI.create(xml)
```

У тілі скрипта додамо наступні функціїии:

```
function speedChange_callbackL(ui,id,newVal)
    vl=newVal*2/1000
end

function speedChange_callbackR(ui,id,newVal)
    vr=newVal*2/1000
end
```

У функції sysCall_cleanup() додамо наступний рядок:

```
simUI.destroy(ui)
```

У функції sysCall_actuation() змінної vLeft і змінної vRight надамо vl і vr відповідно замість v0.

```
vLeft=vl
vRight=vr
```

Запустимо процес симулювання.

У результаті мобільний двоколісний робот навчиться повертати, і ним можна буде керувати.

Побудуємо графік залежності швидкості від часу.

До функції sysCall_init() додамо наступний код:

```
graph=sim.getObjectHandle('Graph')
joint1Vel=sim.addGraphStream(graph,'joint 1 velocity','deg/s',0,{1,0,0})
joint2Vel=sim.addGraphStream(graph,'joint 2 velocity','deg/s',0,{0,1,0})
```

Додамо функцію `sysCall_sensing()`.

```
function sysCall_sensing()

end
```

До функції `sysCall_sensing()` додамо наступний код:

```
sim.setGraphStreamValue(graph,joint1Vel,180*sim.getJointVelocity(motorLeft)/math.pi)
sim.setGraphStreamValue(graph,joint2Vel,180*sim.getJointVelocity(motorRight)/math.pi)
```

Очистимо графік після закінчення процесу симулювання. Для цього до функції sysCall_cleanup() додамо наступний рядок.

```
sim.resetGraph(graph)
```

Запустимо процес симулювання, поерзаємо повзунками. Отримаємо графік зображений на рис. 9.

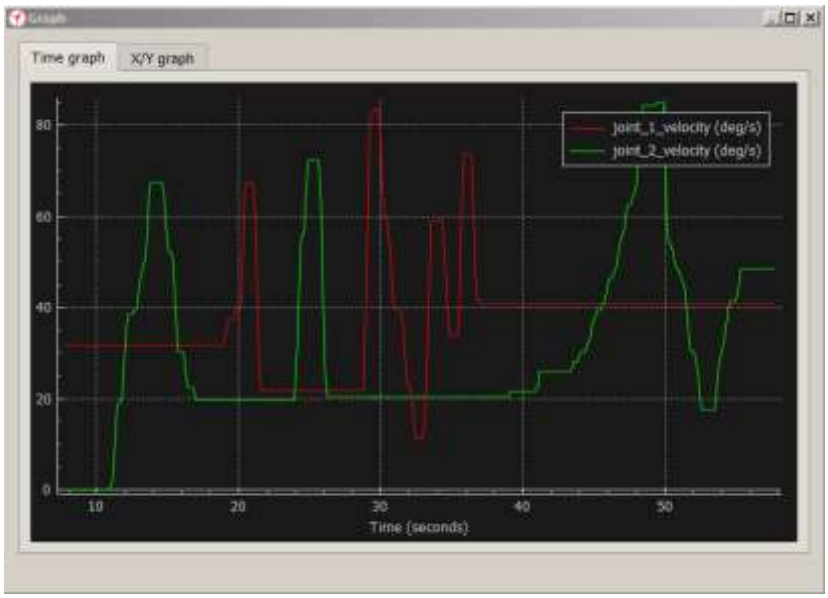


Рис. 9. Графік залежності швидкості від часу

Таким чином, результати імітаційного моделювання робототехнічних систем у програмному комплексі CoppeliaSim можуть бути легко зведені у вигляді графіків.

Контрольні питання

1. Описати, який інструмент дозволяє будувати різноманітні графіки.
2. Назвати, як знайти скрипт симуляції.
3. Описати, як встановити переміщення по колу.
4. Визначити, як, створити графік траєкторії руху .
5. Описати,, як встановити графік залежності x від часу.
6. Визначити, як створити симулювання робота pioneer p3dx з керуванням швидкості.

Завдання

1. Використовуючи наведений приклад ручного керування сконструювати робот з ручним керуванням та перевірити його роботу за допомогою симулятора.
2. Виходячи з реально можливого повороту окремих ланок встановити відповідні обмеження діапазонів у скрипті керування роботом.