ДОДАТОК А

Публікація

ЕФЕКТИВНІСТЬ ВИКОРИСТАННЯ РОБОТИЗОВАНИХ СИСТЕМ У ВИРОБНИЦТВІ

М. Ю. Лазаренко, В. В. Євсєєв, О.М. Цимбал
Харківський національний університет радіоелектроніки
Україна, 61166, Харків, пр. Науки 14
E-mail: mykhailo.lazarenko@nure.ua, vladyslav.yevsieiev@nure.ua, oleksandr.tsymbal@nure.ua

Анотація: У даній роботі розглядається ефективність застосування роботизованих систем у виробничих процесах. Окреслено актуальність тематики, що зумовлена необхідністю підвищення продуктивності, якості, рівня безпеки та конкурентоспроможності підприємств. Дослідження базується на комплексному підході, який включає аналіз наукових джерел, збір даних із реальних виробничих середовищ та моделювання. Визначено основні показники ефективності роботизованих систем, серед яких вартість впровадження, продуктивність, якість, витрати та загальна ефективність. Наведено числові приклади параметрів для виробництва приладів.

Ключові слова: ефективність, коштовність, якість, роботизовані системи, виробництво, сучасна індустріалізація.

EFFICIENCY OF USING ROBOTIC SYSTEMS IN PRODUCTION

M.Y. Lazarenko, V.V. Yevsieiev, O. M. Tsymbal
Kharkiv National University of Radio Electronics
Ukraine, 61166, Kharkiv, 14 Nauky Ave
E-mail: mykhailo.lazarenko@nure.ua, vladyslav.yevsieiev@nure.ua, oleksandr.tsymbal@nure.ua

Annotation: This paper examines the efficiency of robotic systems in manufacturing processes. The relevance of the topic is highlighted, driven by the need to enhance productivity, quality, safety, and the competitiveness of enterprises. The study is based on a comprehensive approach that includes literature analysis, data collection from real production environments, and modeling. Key performance indicators of robotic systems are identified, including implementation costs, productivity, quality, expenses, and overall efficiency. Numerical examples of these parameters are provided for device manufacturing.

Key words: efficiency, cost, quality, robotic systems, producing, modern industrialization.

Головним критерієм, що визначає доцільність впровадження роботизованих комплексів у виробництво, є забезпечення економічної ефективності від їх використання. Реалізація цієї умови вимагає обґрунтованого вирішення всього комплексу завдань на всіх етапах їх виконання.

Ефективне застосування роботизованих систем у виробництві відіграє ключову роль у сучасній промисловості, суттєво впливаючи на рівень продуктивності, якості, безпеки та конкурентоспроможності підприємств.

Застосування роботизованих комплексів забезпечує покращання використання виробничих фондів та підвищення рентабельності виробництва. Основним джерелом отримання економічного ефекту є ріст продуктивності праці, скорочення кількості працюючих робітників і зменшення експлуатаційних витрат.

Промислові роботи є одним із основних компонентів роботизованих комплексів, які використовуються для звільнення людини від виконання фізично важких, монотонних, непривабливих та інших подібних операцій. Особливо важливо те, що промислові роботи можна застосовувати для виконання робіт, які не можуть бути ефективно механізовані або автоматизовані традиційними засобами в умовах постійно змінюваного виробництва. Використання роботів створює передумови для переходу до якісно нового рівня автоматизації – створення автоматизованих та автоматичних виробничих систем, що працюють з мінімальною участю людини.

Сучасна індустріалізація – це процес розвитку промисловості, що базується на передових технологіях, автоматизації, цифровізації та екологічній стійкості. Вона характеризується інтеграцією інтелектуальних систем у виробництво, що забезпечує підвищення ефективності, гнучкості та конкурентоспроможності підприємств.

Сучасна індустріалізація є динамічним процесом розвитку промисловості, який базується на новітніх технологіях, цифровізації, автоматизації та екологічній стійкості. Вона відображає тенденції Четвертої промислової революції (Індустрії 4.0), що включає впровадження Інтернету речей, штучного інтелекту, великих даних та кіберфізичних систем, які забезпечують більш ефективне управління виробничими процесами.

Роботизація відіграє ключову роль у модернізації виробництва, дозволяючи підвищити продуктивність, зменшити вплив людського фактора та покращити якість продукції. Колаборативні роботи, автономні транспортні системи та безпілотні технології значно змінюють підхід до виробництва, роблячи його більш точним і безперервним.

Цифровізація відіграє не менш важливу роль, оскільки сучасні підприємства активно використовують штучний інтелект для аналізу даних, прогнозування потреб і оптимізації процесів. Впровадження інтелектуальних систем управління, таких як ERP, MES і SCADA, дозволяє контролювати виробництво в реальному часі. Також все більшого поширення набувають адитивні технології, зокрема 3D-друк, який спрощує створення прототипів і виробничих деталей.

Окрему увагу в сучасній індустріалізації приділяють екологічній стійкості. Виробничі процеси стають менш ресурсозатратними, впроваджуються технології зменшення шкідливих викидів, активніше використовуються відновлювані джерела енергії. Підприємства впроваджують замкнені цикли виробництва, що сприяє мінімізації відходів і більш раціональному використанню ресурсів.

Глобалізація та децентралізація виробництва також змінюють промисловий ландшафт. Завдяки хмарним технологіям та цифровим платформам керування виробництвом можливе дистанційне управління заводами, а виробничі ланцюги стають більш гнучкими та адаптивними до змін ринку. Крім того, зростає тенденція до масового виробництва з елементами персоналізації, що дозволяє підприємствам відповідати на унікальні запити споживачів.

Перелік параметрів та їх функції у роботизованих системах подано в таблиці 1.

Таблиця 1 – Перелік параметрів та їх функції у роботизованих системах

Параметр	**Опис**
Ефективність	Відображає рівень продуктивності, оптимального використання ресурсів і мінімізації витрат при досягненні

	поставлених цілей у виробничих процесах.
Коштовність	Оцінка загальних фінансових витрат на впровадження, обслуговування та модернізацію технологій, включаючи роботизовані системи та автоматизацію.
Якість	Відповідність продукції або процесу встановленим стандартам та вимогам, що забезпечує надійність, довговічність і конкурентоспроможність товарів.
Роботизовані системи	Автоматизовані технологічні комплекси, що використовують роботів для виконання виробничих завдань з мінімальним або відсутнім втручанням людини.
Виробництво	Процес створення продукції, що включає всі етапи – від проектування та підготовки матеріалів до кінцевої обробки та постачання споживачам.
Сучасна індустріалізація	Перетворення промисловості через впровадження новітніх технологій, автоматизації, цифровізації та екологічної стійкості, що підвищує продуктивність і конкурентоспроможність.

Наведено за приклад числові значення для кожного параметра роботизованих систем у приладобудівному виробництві (табл. 2).

Таблиця 2 – Числові значення для кожного параметра роботизованих систем у приладобудівному виробництві:

Параметр	**Приклад числового значення**
Ефективність	Підвищення продуктивності на 30-50% після впровадження автоматизації та роботизованих систем.
Коштовність	Вартість впровадження роботизованої лінії – від $500,000 до $2,000,000, залежно від складності та масштабу.
Якість	Зменшення браку продукції з 5% до 1% завдяки використанню роботизованих систем з високою точністю обробки (±0.01 мм).
Роботизовані системи	Швидкість роботи промислового маніпулятора – 60-120 циклів/хв, час безперервної роботи – 20,000+ годин без технічного обслуговування.
Виробництво	Потужність лінії – 10,000 приладів на місяць, загальна площа виробництва – 5,000-10,000 м².

Сучасна індустріалізація	Частка автоматизованих процесів у виробництві – 70-90%, частка використання відновлюваної енергії – 30-50%.

Ці числові значення призначені для надання загальної уяви про можливі ефективність впровадження роботизованих систем у виробництві приладів. Значення можуть варіюватися залежно від типу виробництва, рівня автоматизації та конкретних технологій, що застосовуються.

ВИСНОВКИ. Роботизація виробництва значно підвищує ефективність, зменшує витрати та покращує якість продукції. Хоча впровадження роботизованих систем вимагає початкових інвестицій, у довгостроковій перспективі вони окупаються завдяки підвищенню продуктивності, зниженню браку та оптимізації витрат.

Отже, сучасна індустріалізація змінює традиційні підходи до виробництва, роблячи його більш розумним, ефективним, автоматизованим та екологічно безпечним. Інтеграція новітніх технологій дозволяє підприємствам залишатися конкурентоспроможними, швидко адаптуватися до змін ринку та підвищувати якість продукції, що сприяє розвитку глобальної економіки.

Ця робота має значну практичну цінність, оскільки висвітлює переваги та ефективність застосування роботизованих систем у виробництві. Наведені числові приклади ключових показників ефективності, зокрема зростання продуктивності, покращення якості, зменшення витрат і підвищення рівня безпеки, ілюструють конкретні переваги впровадження таких систем у промислових умовах.

З огляду на подальші перспективи, проведене дослідження може слугувати основою для глибшого аналізу ефективності роботизованих технологій у різних галузях виробництва. Отримані результати можуть бути використані підприємствами для обґрунтування інвестицій у роботизацію та розробки стратегій її впровадження.

ЛІТЕРАТУРА

1. Attar, H., & et al.. (2022). Zoomorphic Mobile Robot Development for Vertical Movement Based on the Geometrical Family Caterpillar. Computational Intelligence and Neuroscience, 2022, Article ID 3046116, https://doi.org/10.1155/2022/3046116.

2. Sotnik S. et al.. (2022). Agricultural Robotic Platforms. International Journal of Engineering and Information Systems (IJEAIS), 2022.

3. Sotnik S. V. et al.. (2023). Design features of control panels and consoles in automation systems. 9th International scientific and practical conference “Science and innovation of modern world”, 2023.

4. John J. Craig – "Introduction to Robotics: Mechanics and Control"

5. Bao X. et al. (2023). A novel ultrasound robot with force/torque measurement and control for safe and efficient scanning. IEEE transactions on instrumentation and measurement, 2023.

6. Oppermann M. et al.. (2023). The pedicle screw accuracy using a robotic system and measured by a novel three-dimensional method. Journal of Orthopaedic Surgery and Research, 2023.

7. Mikell P. Groover – "Automation, Production Systems, and Computer-Integrated Manufacturing"

8. "IEEE Transactions on Robotics"

ДОДАТОК Б

Код кінематичної моделі

Код сцени з маніпулятором у OpenGL та кінематикою:

App.cpp:

```
//App.cpp
#include <afxwin.h>
#include <afxcmn.h>
#include "SerialWindow.h"
#include <memory.h>
#include <math.h>                       // sqrt
#include "gl/gl.h"                      // OPENGL
#include "gl/glu.h"                 // OPENGL
//#include "glaux.h"                // OPENGL
#include "App.h"
#include "resource.h"
#include "Serial.h"

double pi=3.141592653589793;
double al1=90.0,al4=-50.4,al5=90.0,al6=0.0,al7=0.0;
double al11=0.0,al41=0.0,al51=0.0,al61=0.0,al71=0.0;

double a11=-0.6374,a12=0.0,a13=0.7700,a1=4.2, a10=0.0,
      a21= 0.0000,a22=1.0,a23=0.0000,a2=1.3, a20=0.0,
            a31=-0.7700,a32=0.0,a33=0.6374,a3=4.11,a30=0.0;

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

int vspos=50;
int hspos=50;
bool taskMode=true;
char worldMove=0;
int curWorldCoord1=0,curWorldCoord2=0,curWorldCoord3=0;

CMain::CMain()
{
```

```
        Create(NULL,"OpenGL: Промисловий робот",WS_OVERLAPPEDWINDOW|WS_VSCROLL|WS_HSCROLL,rectDefault,NULL,MAKEINTRESOURCE(IDR_MENU1));

        InitCommonControls();
        SetScrollPos(SB_HORZ,50);
        SetScrollPos(SB_VERT,50);

            // Fill in the Pixel Format Descriptor
            PIXELFORMATDESCRIPTOR pfd ;
            memset(&pfd,0, sizeof(PIXELFORMATDESCRIPTOR)) ;
            pfd.nSize = sizeof(PIXELFORMATDESCRIPTOR);
            pfd.nVersion = 1 ;                    // Version number
            pfd.dwFlags =  PFD_DOUBLEBUFFER |          // Use double buffer
                                                PFD_SUPPORT_OPENGL |          // Use OpenGL
                                                PFD_DRAW_TO_WINDOW ;       // Pixel format is for a window.
            pfd.iPixelType = PFD_TYPE_RGBA ;
            pfd.cColorBits = 24;                  // 8-bit color
            pfd.cDepthBits = 32 ;               // 32-bit depth buffer
            pfd.iLayerType = PFD_MAIN_PLANE ;         // Layer type

          CClientDC dc(this);
            int nPixelFormat = ChoosePixelFormat(dc.m_hDC, &pfd);
            TRACE("Pixel Format %d\r\n", nPixelFormat) ;
            BOOL bResult = SetPixelFormat(dc.m_hDC, nPixelFormat, &pfd);
            // Create a rendering context.
            m_hrc = wglCreateContext(dc.m_hDC);
            wglMakeCurrent(dc.m_hDC,m_hrc);
            m_pPal = NULL;
            taskMode=true;
        }

        BOOL CApp::InitInstance()
        {
            m_pMainWnd = new CMain;
            m_pMainWnd->ShowWindow(m_nCmdShow);
            m_pMainWnd->UpdateWindow();

            // Запускаем оба окна
```

```
        ((CMain*)m_pMainWnd)->OnRobotControl();   // Діалог MFC
        ((CMain*)m_pMainWnd)->OnSerialWindow();   // WinAPI вікно

        return TRUE;
}

CApp App;

BEGIN_MESSAGE_MAP(CMain,CFrameWnd)
        ON_WM_PAINT()
        ON_WM_VSCROLL()
        ON_WM_HSCROLL()
        ON_WM_SIZE()
        ON_COMMAND(ID_ROBOTCONTROL,OnRobotControl)
        ON_COMMAND(ID_SERIAL_WINDOW, OnSerialWindow)
END_MESSAGE_MAP()

void CMain::OnSerialWindow() {
        CreateSerialWindow(this->GetSafeHwnd());
}

void CMain::OnPaint()
{
        CPaintDC dc(this);

        CPaintDC pDC(this);
   CPalette* ppalOld = NULL;
        if (m_pPal)
        {       ppalOld = pDC.SelectPalette(m_pPal, 0);
        pDC.RealizePalette();
        }
   wglMakeCurrent(pDC.m_hDC, m_hrc);
        GLInit();
        OnScene();
        SwapBuffers(pDC.m_hDC) ;
   if (ppalOld) pDC.SelectPalette(ppalOld, 0);
        wglMakeCurrent(NULL, NULL) ;
}

void CMain::OnSize(UINT nType, int cx, int cy)
{   CClientDC dc(this) ;
```

```
    BOOL bResult = wglMakeCurrent(dc.m_hDC, m_hrc);
    GLdouble gldAspect = (GLdouble) cx/ (GLdouble) cy;
    glMatrixMode(GL_PROJECTION);// OutputGlError("MatrixMode") ;
            glLoadIdentity();
            gluPerspective(30.0, gldAspect, 1.0, 10.0);
    glViewport(0, 0, cx, cy);
    wglMakeCurrent(NULL, NULL);
}

void CMain::OnVScroll(UINT SBCode, UINT Pos, CScrollBar *SB)
{switch(SBCode)
 {case SB_LINEDOWN: vspos++;break;
  case SB_LINEUP:         vspos--;break;
  case SB_PAGEDOWN: vspos+=5;break;
  case SB_PAGEUP:         vspos-=5;break;
  case SB_THUMBTRACK: vspos=Pos;break;
  case SB_THUMBPOSITION: vspos=Pos;break;
 }
 Invalidate(FALSE);
 SetScrollPos(SB_VERT,vspos);
}

void CMain::OnHScroll(UINT SBCode, UINT Pos, CScrollBar *SB)
{switch(SBCode)
 {case SB_LINERIGHT: hspos++;break;
  case SB_LINELEFT:       hspos--;break;
  case SB_PAGERIGHT: hspos+=5;break;
  case SB_PAGELEFT:  hspos-=5;break;
  case SB_THUMBTRACK: hspos=Pos;break;
  case SB_THUMBPOSITION: hspos=Pos;break;
 }
 Invalidate(FALSE);
 SetScrollPos(SB_HORZ,hspos);
}

int i=0;
GLdouble curMatrix2[16]={0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0};

void CMain::GLInit()
{           static GLdouble marengo[3] = {1.0, 0.0, 0.0 } ;
            glClearColor(1.0,1.0,1.0,0.0) ;
```

```
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        glColor3dv(marengo);
        glTranslated(0.0, 0.0, -5.0 );
        glScaled(0.005,0.005,0.005);
}

int rColumn=0,rShoulder=0,rElbow=0,rArmBase=0,rArm=0,lArm=0;
double rColumn1=0,rShoulder1=0,rElbow1=0;
double rColumn2=0,rShoulder2=0,rElbow2=0;
double rFinger=0.0;

void CMain::multMatrix(double *a, double *b)
{       double curMatrix3[16]={0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0};
        for (int k=0;k<4;k++)
        for (int j=0;j<4;j++)
        {double s=0.0;
         for(int i=0;i<4;i++)s=s+*(a+((4*k)+i))*(*(b+((4*i)+j)));
         curMatrix3[4*k+j]=s;
        }
        for(int i=0;i<16;i++)curMatrix2[i]=curMatrix3[i];
}

double fi1=0.0,fi23=0.0,fi4=0.0,fi5=0.0,fi6=0.0,fi7=0.0;
double rad10=0.0,rad40=0.0,rad50=0.0,rad60=0.0,rad70=0.0;
double x=0.0,y=0.0,z=0.0,gamma=0.0,r=0.0,fi2=0.0,fi3=0.0,w=0.0;

void CMain::OnScene()
{
        glTranslatef(0.0,-100.0,0.0);
        glRotated(-90.0,1.0,0.0,0.0);
        glRotated(360.0*(vspos-50)/100,0,0,1); // 3D-scrolling around z
        glRotated(360.0*(hspos-50)/100,1,0,0); // 3D-scrolling around y
        glTexEnvf(GL_TEXTURE_ENV,GL_TEXTURE_ENV_MODE,GL_REPLACE);
        glLineWidth(2.0);
        glBegin(GL_LINES);
                glColor3f(1.0,0.0,0.0);
                glVertex3f(50.0f,0.0f,0.0f);
                glVertex3f(0.0f,0.0f,0.0f);
```

```
		glColor3f(0.0,1.0,0.0);
		glVertex3f(0.0f,50.0f,0.0f);
		glVertex3f(0.0f,0.0f,0.0f);

		glColor3f(0.0,0.0,1.0);
		glVertex3f(0.0f,0.0f,25.0f);
		glVertex3f(0.0f,0.0f,0.0f);
glEnd();
curMatrix2[0]=zeroMatrix(curMatrix2);
glEnable(GL_TEXTURE_2D);
glBegin(GL_POLYGON);//blue square
		glTexCoord2i(0,0);glVertex2i(5,5);
		glTexCoord2i(1,0);glVertex2i(5,-5);
		glTexCoord2i(1,1);glVertex2i(-5,-5);
		glTexCoord2i(0,1);glVertex2i(-5,5);
glEnd();
glDisable(GL_TEXTURE_2D);
glColor3f(1.0,0.0,0.0);
glPushMatrix();

if(taskMode)
{
		glRotated(rColumn1+fi1,0.0,0.0,1.0);
		rotateMatrix(rColumn1+fi1,0.0,0.0,1.0);
		glRotated(-90.0,0.0,0.0,1.0);
		rotateMatrix(-90.0,0.0,0.0,1.0);

		glBegin(GL_LINES);
			glVertex3i(0,0,0);glVertex3i(0,0,60);//pole
			glVertex3i(0, 0, 60); glVertex3i(0, 50, 60);
			glVertex3i(0, 50, 60); glVertex3i(0, 50, 80);
		glEnd();

		glTranslatef(0.0,50.0,80.0);		transMatrix(0.0,50.0,80.0);//pole hight
		glRotatef(90.0,1.0,0.0,0.0);		rotateMatrix(90.0,1.0,0.0,0.0);

		glRotated(rShoulder1+fi2,1.0,0.0,0.0);	rotateMatrix(rShoulder1+fi2,1.0,0.0,0.0);
		glBegin(GL_LINES);
			glVertex3i(0,0,0);glVertex3i(0,0,115);
			glVertex3i(0, 0, 115); glVertex3i(0, 10, 115);
		glEnd();
```

```
                glTranslatef(0.0,10.0,115.0);                transMatrix(0.0,10.0,115.0);
                glRotatef(180.0, 0.0, 1.0, 0.0);             rotateMatrix(180.0, 0.0, 1.0, 0.0);
                glRotated(rElbow1+fi3,1.0,0.0,0.0);
rotateMatrix(rElbow1+fi3,1.0,0.0,0.0);
        }
        else
        {
                switch(worldMove)
                {
                 case 'x': x=curMatrix2[3]+curWorldCoord1;
                                        y=curMatrix2[7]+curWorldCoord2;
                                        z=curMatrix2[11]+curWorldCoord3;
                                        gamma=atan(z/y);
                                        w=sqrt(x*x+z*z);
                                        r=sqrt(w*w+y*y);
                                        fi1=-atan(x/y);
                                        fi3=acos((r*r-115.0*115.0-193.0*193.0)/(2*115.0*193.0));
                                        fi2=-pi/2-gamma-asin(sin(fi3)*193.0/r);
                                        break;
                 case 'y': y=curMatrix2[7]+curWorldCoord2;
                                        z=curMatrix2[11]+curWorldCoord3;;
                                        gamma=atan(z/y);
                                        r=z/sin(gamma);
                                        fi3=-acos((r*r-115.0*115.0-193.0*193.0)/(2*115.0*193.0));
                                        fi2=pi/2-gamma-asin(sin(fi3)*193.0/r);
                                        break;
                 case 'z': y=curMatrix2[7]+curWorldCoord2;;
                                        z=curMatrix2[11]+curWorldCoord3;
                                        gamma=atan(z/y);
                                        r=z/sin(gamma);
                                        fi3=acos((r*r-115.0*115.0-193.0*193.0)/(2*115.0*193.0));
                                        fi2=-pi/2-gamma-asin(sin(fi3)*193.0/r);
                                        glRotated(180.0,0.0,1.0,0.0);          //??????????????
                                        break;
                }

                ////////////////////////////////////////////////////////////////coordin vers
                glRotated(rColumn1 + fi1, 0.0, 0.0, 1.0);
                rotateMatrix(rColumn1 + fi1, 0.0, 0.0, 1.0);
                glRotated(-90.0, 0.0, 0.0, 1.0);
```

```
                rotateMatrix(-90.0, 0.0, 0.0, 1.0);

                glBegin(GL_LINES);
                glVertex3i(0, 0, 0); glVertex3i(0, 0, 60);//pole
                glVertex3i(0, 0, 60); glVertex3i(0, 50, 60);
                glVertex3i(0, 50, 60); glVertex3i(0, 50, 80);
                glEnd();

                glTranslatef(0.0, 50.0, 80.0);          transMatrix(0.0, 50.0, 80.0);//pole hight
                glRotatef(90.0, 1.0, 0.0, 0.0);         rotateMatrix(90.0, 1.0, 0.0, 0.0);

                glRotated(rShoulder1 + fi2, 1.0, 0.0, 0.0);   rotateMatrix(rShoulder1 + fi2, 1.0, 0.0, 0.0);
                glBegin(GL_LINES);
                glVertex3i(0, 0, 0); glVertex3i(0, 0, 115);
                glVertex3i(0, 0, 115); glVertex3i(0, 10, 115);
                glEnd();

                glTranslatef(0.0, 10.0, 115.0);         transMatrix(0.0, 10.0, 115.0);
                glRotatef(180.0, 0.0, 1.0, 0.0);        rotateMatrix(180.0, 0.0, 1.0, 0.0);
                glRotated(rElbow1 + fi3, 1.0, 0.0, 0.0);        rotateMatrix(rElbow1 + fi3, 1.0, 0.0,
0.0);
        }
        ////////////////////////////////////////////////////////////////////////////////////////
                glBegin(GL_LINES);
                glVertex3i(0,0,0);glVertex3i(0,0,150);
                glEnd();
                CString ss;
                ss.Format(_T("%f %f %f"),curMatrix2[3],curMatrix2[7],curMatrix2[11]);
                SetWindowText(ss);

                glTranslatef(0.0,0.0,150.0);            transMatrix(0.0,0.0,150.0);// зап'ястя
                glRotatef(90.0, 1.0, 0.0, 0.0);         rotateMatrix(90.0, 1.0, 0.0, 0.0);
                glRotated(rArmBase,1.0,0.0,0.0);   rotateMatrix(rArmBase,1.0,0.0,0.0);
                glBegin(GL_LINES);
                        glVertex3i(0,0,0);glVertex3i(0,0,30);
                glEnd();
                glTranslatef(0.0,0.0,30.0);             transMatrix(0.0,0.0,30.0);
                curMatrix2;

                glRotatef(rArm+fi5, 0.0, 1.0, 0.0);     rotateMatrix(rArm+fi5,0.0, 1.0, 0.0);
                glBegin(GL_LINES);
```

```
                glVertex3i(-25,0,10);glVertex3i(-12,0,40);// left finger
                glVertex3i(0,0,0);glVertex3i(-25,0,10);// center left
                glEnd();

        glRotatef(lArm + fi5, 0.0, 1.0, 0.0);          rotateMatrix(lArm + fi5, 0.0, 1.0,
0.0);
        glBegin(GL_LINES);
                glVertex3i(0,0,0); glVertex3i(25,0,10);// center right
                glVertex3i(25,0,10);glVertex3i(12,0,40);// right finger
        glEnd();
        glPopMatrix();
}

CRobotControl::CRobotControl() : CDialog(IDD_DIALOG1)
{
        // Конструктор за замовчуванням
        //CEdit ed[16];
}

//void CRobotControl::DoDataExchange(CDataExchange* pDX)
//{
//      CDialog::DoDataExchange(pDX);
//
//      // Підключення всіх 19 текстових полей
//      for (int i = 1; i < 17; ++i)
//      {
//              DDX_Control(pDX, IDC_EDIT1 + i, ed[i]);
//      }
//
//      CSliderCtrl sliders[5];
//
//      for (int i = 1; i < 6; ++i)
//      {
//              DDX_Control(pDX, IDC_SLIDER1 + i, sliders[i]);
//      }
//}

BEGIN_MESSAGE_MAP(CRobotControl,CDialog)
        ON_WM_HSCROLL()
        ON_WM_VSCROLL()
        ON_COMMAND(IDC_RADIO1,OnRadio)
```

```
	ON_COMMAND(IDC_RADIO2,OnRadio)
	ON_COMMAND(IDC_CHECK1,OnHandShake)
	ON_COMMAND(IDOK,OnOK)
END_MESSAGE_MAP()

void CMain::OnRobotControl()
{
	static CRobotControl* ctrl = nullptr;
	if (!ctrl) {
		ctrl = new CRobotControl();
		ctrl->Create(IDD_DIALOG1, this);
		ctrl->ShowWindow(SW_SHOW);
	}
	else {
		ctrl->ShowWindow(SW_SHOW);
		ctrl->SetFocus();
	}
}

BOOL CRobotControl::OnInitDialog()
{
	CDialog::OnInitDialog();
 ed[0]=(CEdit*)GetDlgItem(IDC_EDIT1);
 ed[1]=(CEdit*)GetDlgItem(IDC_EDIT2);
 ed[2]=(CEdit*)GetDlgItem(IDC_EDIT3);
 ed[3]=(CEdit*)GetDlgItem(IDC_EDIT4);
 ed[4]=(CEdit*)GetDlgItem(IDC_EDIT5);
 ed[5]=(CEdit*)GetDlgItem(IDC_EDIT6);
 ed[6]=(CEdit*)GetDlgItem(IDC_EDIT7);
 ed[7]=(CEdit*)GetDlgItem(IDC_EDIT8);
 ed[8]=(CEdit*)GetDlgItem(IDC_EDIT9);
 ed[9]=(CEdit*)GetDlgItem(IDC_EDIT10);
 ed[10]=(CEdit*)GetDlgItem(IDC_EDIT11);
 ed[11]=(CEdit*)GetDlgItem(IDC_EDIT12);
 ed[12]=(CEdit*)GetDlgItem(IDC_EDIT13);
 ed[13]=(CEdit*)GetDlgItem(IDC_EDIT14);
 ed[14]=(CEdit*)GetDlgItem(IDC_EDIT15);
 ed[15]=(CEdit*)GetDlgItem(IDC_EDIT16);
 ed[16]=(CEdit*)GetDlgItem(IDC_EDIT17);
 ed[17]=(CEdit*)GetDlgItem(IDC_EDIT18);
 ed[18]=(CEdit*)GetDlgItem(IDC_EDIT19);
```

```
 rb1=(CButton*)GetDlgItem(IDC_RADIO1);
 rb2=(CButton*)GetDlgItem(IDC_RADIO1);
 ch1=(CButton*)GetDlgItem(IDC_CHECK1);
 rb1->SetCheck(1);
 ch1->SetCheck(0);
 sl1=(CSliderCtrl*)GetDlgItem(IDC_SLIDER1);
 sl2=(CSliderCtrl*)GetDlgItem(IDC_SLIDER2);
 sl3=(CSliderCtrl*)GetDlgItem(IDC_SLIDER3);
 sl4=(CSliderCtrl*)GetDlgItem(IDC_SLIDER4);
 sl5=(CSliderCtrl*)GetDlgItem(IDC_SLIDER5);
        sl1->SetPos(0);sl2->SetPos(0);
        sl3->SetPos(0);sl4->SetPos(0);sl5->SetPos(0);

spin1.Create(WS_CHILD|WS_VISIBLE|WS_BORDER|UDS_SETBUDDYINT|UDS_ALIGNLEFT,CRec
t(0,0,0,0),this,1);

spin2.Create(WS_CHILD|WS_VISIBLE|WS_BORDER|UDS_SETBUDDYINT|UDS_ALIGNLEFT,CRec
t(0,0,0,0),this,2);

spin3.Create(WS_CHILD|WS_VISIBLE|WS_BORDER|UDS_SETBUDDYINT|UDS_ALIGNLEFT,CRec
t(0,0,0,0),this,3);
 spin1.SetBuddy(GetDlgItem(IDC_EDIT17));
 spin2.SetBuddy(GetDlgItem(IDC_EDIT18));
 spin3.SetBuddy(GetDlgItem(IDC_EDIT19));
 spin1.SetRange(-100,100);spin2.SetRange(0,269);spin3.SetRange(5,158);
 spin1.SetPos(0);spin2.SetPos(0);spin3.SetPos(0);
 OnSetMode(true);
        return TRUE;
}

int hscl=0;
int bf=0;

void CRobotControl::OnHScroll(UINT SBCode, UINT Pos, CScrollBar *SB)
{CDialog::OnHScroll(SBCode, Pos, SB);
 a10=curMatrix2[3];a20=curMatrix2[7];a30=curMatrix2[11];
 if(SB==(CScrollBar*)sl1)
        if(taskMode)
        {
                rColumn2=rColumn1;rColumn1=sl1->GetPos();
        }
```

```
            else
            {
                    rColumn1=rColumn2;rColumn2=sl1->GetPos();
            }
if(SB==(CScrollBar*)sl2)
            if(taskMode)
            {
                    rShoulder2=rShoulder1;rShoulder1=sl2->GetPos();
            }
            else
            {
                    rShoulder1=rShoulder2;rShoulder2=sl2->GetPos();
            }
if(SB==(CScrollBar*)sl3)
            if(taskMode)
            {
                    rElbow2=rElbow1;rElbow1=sl3->GetPos();
            }
            else
            {
                    rElbow1=rElbow2;rElbow2=sl3->GetPos();
            }

if(SB==(CScrollBar*)sl4) rArmBase=sl4->GetPos();
if(SB==(CScrollBar*)sl5) rArm=sl5->GetPos();
if (SB == (CScrollBar*)sl5) lArm = sl5->GetPos();
CString ss("");
for(int k=0;k<16;k++)
{ss.Format(_T("%f"),curMatrix2[k]);

//ed[k] = (CEdit*)GetDlgItem(IDC_EDIT1 + k);
 ed[k]->SetWindowText(ss);
}
CClientDC dc(this);
ss.Format(_T("R: %f %f %f"),a10,a20,a30);
dc.TextOut(10,440,ss);
this->GetParentFrame()->Invalidate(FALSE);
}

void CRobotControl::OnSetMode(bool mode)
{
```

```
            CStatic *st1=(CStatic*)GetDlgItem(IDC_STATIC1);
            CStatic *st2=(CStatic*)GetDlgItem(IDC_STATIC2);
            CStatic *st3=(CStatic*)GetDlgItem(IDC_STATIC3);
            CStatic *st4=(CStatic*)GetDlgItem(IDC_STATIC4);
            CStatic *st5=(CStatic*)GetDlgItem(IDC_STATIC5);
            if(mode)
            { sl4->EnableWindow(TRUE);      //rColumn2=0;
              sl4->ShowWindow(SW_SHOW);
              sl5->EnableWindow(TRUE);
              sl5->ShowWindow(SW_SHOW);

              sl1->SetRange(-180,180,true);
              sl2->SetRange(-180,0,true);
              sl3->SetRange(-180,0,true);
              sl4->SetRange(-180,0,true);
              sl5->SetRange(-90,90,true);

              st1->SetWindowText("Колона");
              st2->SetWindowText("Плече");
              st3->SetWindowText("Лікоть");
              st4->ShowWindow(SW_SHOW);
              st4->SetWindowText("Зап'ястя");
              st5->ShowWindow(SW_SHOW);
              st5->SetWindowText("Кисть");
            }
            else
            {
             sl4->EnableWindow(FALSE);      //rColumn1=0;
             sl4->ShowWindow(SW_HIDE);
             sl5->EnableWindow(FALSE);
             sl5->ShowWindow(SW_HIDE);
    st1->SetWindowText("X");
    st2->SetWindowText("Y");
    st3->SetWindowText("Z");
    st4->ShowWindow(SW_HIDE);
    st5->ShowWindow(SW_HIDE);
            }
}

void CRobotControl::OnRadio()
{           if(rb1->GetCheck())
```

```
                    {taskMode=true;OnSetMode(true);}
        else        {taskMode=false;OnSetMode(false);}
}

void CRobotControl::OnHandShake()
{       if(ch1->GetCheck())rFinger=0.15;
        else rFinger=0.0;
        this->GetParentFrame()->Invalidate(FALSE);
}

double CMain::zeroMatrix(double *b)
{       for(int i=0;i<4;i++)
        for(int j=0;j<4;j++)
        {if(i==j)b[4*i+j]=1.0;else b[4*i+j]=0.0;}
        return b[0];
}

void CMain::transMatrix(double x, double y, double z)
{
        double trans[16]={1.0,0.0,0.0,x,
                                        0.0,1.0,0.0,y,
                0.0,0.0,1.0,z,
                                        0.0,0.0,0.0,1.0};
        multMatrix(curMatrix2,trans);
}

void CMain::rotateMatrix(double angle, double x, double y, double z)
{   double fi=pi*angle/180.0;
        double rotate[16];
        if(x==1.0){ double rotateX[16]= {1.0,0.0,     0.0,    0.0,
                                                                0.0,cos(fi),-sin(fi),0.0,
                                                                0.0,sin(fi), cos(fi),0.0,
                                                                0.0,0.0,     0.0,    1.0};
                        for(int i=0;i<16;i++)rotate[i]=rotateX[i];
                        }
        if(y==1.0){ double rotateY[16]={cos(fi),0.0,sin(fi),0.0,
                                                                0.0,    1.0, 0.0,   0.0,
                        -sin(fi),0.0,cos(fi),0.0,
                                                                0.0,    0.0, 0.0,   1.0};
                        for(int i=0;i<16;i++)rotate[i]=rotateY[i];
                        }
```

```
            if(z==1.0){     double rotateZ[16]= {cos(fi),-sin(fi),0.0,0.0,
                                                 sin(fi), cos(fi),0.0,0.0,
                                                 0.0,    0.0,    1.0,0.0,
                                                 0.0,    0.0,    0.0,1.0};
                                for(int i=0;i<16;i++)rotate[i]=rotateZ[i];
                                }
            multMatrix(curMatrix2,rotate);
        }

        void CRobotControl::OnOK()
        {       taskMode=true;
            SendMessage(WM_CLOSE);
        }

        void CRobotControl::JointsGen(int joint)
        {       double L2=115.0,L3=110.0,L4=83.0;
            for(double a2=-45.0;a2<90.0;a2=a2+0.25)
            for(double a3=-90.0;a3<90.0;a3=a3+0.25)
            {double a,b,q2,q3,e;
             a=cos(a2*pi/180.0)*L2-sin(fi23)*(L3+L4);
             b=sin(a2*pi/180.0)*L2+cos(fi23)*(L3+L4);
             q2=atan((b*L2*(1-sin(a3*pi/180.0))-a*cos(a3*pi/180.0)*(L3+L4))/(a*L2*(1-sin(a3*pi/180.0))-
b*cos(a3*pi/180.0)*(L3+L4)));
             q3=asin((L2*L2+(L3+L4)*(L3+L4)-a*a-b*b)/(2*L2*(L3+L4)));
             e=q2+q3;rColumn2=180.0*fi1/pi;rShoulder2=180.0*q2/pi;rElbow2=180.0*q3/pi;
             if(fabs(fi23-e)<0.1){break;}
            }
            CString ss;
            ss.Format(_T("%f %f %f"),rColumn2,rShoulder2,rElbow2);
            SetWindowText(ss);
        }

        void CRobotControl::OnVScroll(UINT SBCode, UINT Pos, CScrollBar *SB)
        {CDialog::OnVScroll(SBCode, Pos, SB);
         if(SB==(CScrollBar*)&spin1) worldMove='x';curWorldCoord1=spin1.GetPos();
         if(SB==(CScrollBar*)&spin2) worldMove='y';curWorldCoord2=spin2.GetPos();
         if(SB==(CScrollBar*)&spin3) worldMove='z';curWorldCoord3=spin3.GetPos();
         CString ss("");
         for(int k=0;k<16;k++)
         {ss.Format(_T("%f"),curMatrix2[k]);
          ed[k]->SetWindowText(ss);
```

```
 }
 this->GetParentFrame()->Invalidate(FALSE);
}
```

App.h:

```
// App.h: interface for the CApp class.
//
//////////////////////////////////////////////////////////////////////

#if !defined(AFX_APP_H__1CC13A23_76E6_11D7_8E7D_9FA0C78A3C3D__INCLUDED_)
#define AFX_APP_H__1CC13A23_76E6_11D7_8E7D_9FA0C78A3C3D__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

class CApp : public CWinApp
{public: BOOL InitInstance();};


class CMain : public CFrameWnd
{
        CPalette *m_pPal;
        HGLRC m_hrc;
public:
        void rotateMatrix(double,double,double,double);
        void transMatrix(double,double,double);
        double zeroMatrix(double*);
        void multMatrix(double *a, double *b);
        void OnRobotControl();
        void OnScene();
        void GLInit();
        void OnVScroll(UINT SBCode,UINT Pos,CScrollBar *SB);
        void OnHScroll(UINT SBCode,UINT Pos,CScrollBar *SB);
        void OnSize(UINT nType, int cx, int cy);
        void OnPaint();
        CMain();
        afx_msg void OnSerialWindow();
        DECLARE_MESSAGE_MAP()
};

class CRobotControl : public CDialog
{

        CSliderCtrl *sl1, *sl2, *sl3, *sl4, *sl5;
        CButton *rb1,*rb2,*ch1;
        CEdit *ed[19];
        CSpinButtonCtrl spin1,spin2,spin3;
public:
        CRobotControl();
        void JointsGen(int);
        void OnOK();
        void OnHandShake();
        void OnRadio();
        void OnSetMode(bool);
        void OnHScroll(UINT SBCode,UINT Pos,CScrollBar *SB);
        void OnVScroll(UINT SBCode,UINT Pos,CScrollBar *SB);
        BOOL OnInitDialog();
```

```
            CRobotControl(UINT id,CWnd *Owner):CDialog(id,Owner){}
//          CRobotControl():CDialog(){}
            DECLARE_MESSAGE_MAP()
//protected:
//          virtual void DoDataExchange(CDataExchange* pDX);
};

#endif // !defined(AFX_APP_H__1CC13A23_76E6_11D7_8E7D_9FA0C78A3C3D__INCLUDED_)
```

SerialWindow.cpp:

```
////////// SerialWindow.cpp
#include <windows.h>
#include "resource.h"

#define ID_EDIT 1001
#define ID_TIMER 1

static HWND hEdit;
static HANDLE hSerial;
static char buffer[256];
static const char CLASS_NAME[] = "SerialWinClass";

// Відкриття COM
bool InitSerialPort(LPCSTR portName) {
   hSerial = CreateFileA(portName, GENERIC_READ, 0, NULL, OPEN_EXISTING, 0, NULL);
   if (hSerial == INVALID_HANDLE_VALUE) return false;

   DCB dcb = { sizeof(dcb) };
   GetCommState(hSerial, &dcb);
   dcb.BaudRate = CBR_9600;
   dcb.ByteSize = 8;
   dcb.StopBits = ONESTOPBIT;
   dcb.Parity = NOPARITY;
   SetCommState(hSerial, &dcb);

   COMMTIMEOUTS timeouts = { 50, 10, 50, 0, 0 };
   SetCommTimeouts(hSerial, &timeouts);

   return true;
}

void ReadFromSerial(HWND hWnd) {
   DWORD bytesRead;
   if (ReadFile(hSerial, buffer, sizeof(buffer) - 1, &bytesRead, NULL) && bytesRead > 0) {
      buffer[bytesRead] = '\0';
      SetWindowTextA(hEdit, buffer);
   }
}

LRESULT CALLBACK SerialWndProc(HWND hWnd, UINT msg, WPARAM wParam, LPARAM
lParam) {
   switch (msg) {
   case WM_CREATE:
      hEdit = CreateWindowA("EDIT", "", WS_CHILD | WS_VISIBLE | WS_BORDER |
ES_READONLY,
         10, 10, 300, 25, hWnd, (HMENU)ID_EDIT, NULL, NULL);
      if (InitSerialPort("COM5")) {
         SetTimer(hWnd, ID_TIMER, 100, NULL);
      }
```

```
            break;
        case WM_TIMER:
            if (wParam == ID_TIMER) ReadFromSerial(hWnd);
            break;
        case WM_DESTROY:
            if (hSerial != INVALID_HANDLE_VALUE) CloseHandle(hSerial);
            break;
        }
        return DefWindowProc(hWnd, msg, wParam, lParam);
    }

    // Виклик з CMain::OnSerialWindow()
    void CreateSerialWindow(HWND hParent) {
        WNDCLASSA wc = { 0 };
        wc.lpfnWndProc = SerialWndProc;
        wc.hInstance = GetModuleHandle(NULL);
        wc.lpszClassName = CLASS_NAME;
        RegisterClassA(&wc);

        HWND hWnd = CreateWindowA(CLASS_NAME, "Arduino Serial Reader", WS_OVERLAPPED |
WS_CAPTION | WS_SYSMENU,
            CW_USEDEFAULT, CW_USEDEFAULT, 350, 100, hParent, NULL, GetModuleHandle(NULL),
NULL);
        ShowWindow(hWnd, SW_SHOW);
        UpdateWindow(hWnd);
    }
```

Serial.cpp:

```
    // Serial.cpp
    #include "Serial.h"
    #include <iostream>

    Serial::Serial(const char* portName) {
        hSerial = CreateFileA(portName, GENERIC_READ | GENERIC_WRITE, 0, NULL,
            OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL);

        if (hSerial == INVALID_HANDLE_VALUE) {
            MessageBoxA(NULL, "Помилка вікриття порту", "Помилка", MB_OK | MB_ICONERROR);
            return;
        }

        DCB dcbSerialParams = { 0 };
        dcbSerialParams.DCBlength = sizeof(dcbSerialParams);
        GetCommState(hSerial, &dcbSerialParams);
        dcbSerialParams.BaudRate = CBR_9600;
        dcbSerialParams.ByteSize = 8;
        dcbSerialParams.StopBits = ONESTOPBIT;
        dcbSerialParams.Parity = NOPARITY;
        SetCommState(hSerial, &dcbSerialParams);
    }

    Serial::~Serial() {
        if (hSerial != INVALID_HANDLE_VALUE)
            CloseHandle(hSerial);
    }

    int Serial::read(char* buffer, unsigned int buf_size) {
        DWORD bytesRead;
        if (!ReadFile(hSerial, buffer, buf_size, &bytesRead, NULL)) {
```

```
            return 0;
        }
        return bytesRead;
    }
```

Serial.h:

```
// Serial.h
#pragma once
#include <windows.h>

class Serial {
private:
    HANDLE hSerial;
public:
    Serial(const char* portName);
    ~Serial();
    int read(char* buffer, unsigned int buf_size);
};
```

Повний код керування роботом-маніпулятором (Arduino IDE):

```
#include <Servo.h>
#include <EEPROM.h>

// === Сервоприводи ===
Servo base, shoulder, elbow, rotator, gripper;

// === Піни ===
const int x_key1 = A0, y_key1 = A1, btn_key1 = A2;
const int x_key2 = A3, y_key2 = A4, btn_key2 = A5;
const int pin_base = 11, pin_shoulder = 10, pin_elbow = 9, pin_rotator =
8, pin_gripper = 7;

// === Позиції ===
int shoulderAngle = 0;
int elbowAngle = 180;
int rotatorAngle = 0;
int gripperAngle = 0;

// === EEPROM адреса ===
const int ADDR_SHOULDER = 1;
const int ADDR_ELBOW = 2;
const int ADDR_ROTATOR = 3;
const int ADDR_GRIPPER = 4;
const int ADDR_MODE = 5;

// === Керування ===
```

```
const int step = 6;
const int threshold = 300;
unsigned long lastMoveTime = 0;
const unsigned long moveDelay = 70;

bool mode = 0;  // 0 – ручний, 1 – координатний
bool lastButton1State = HIGH;
unsigned long lastDebounceTime = 0;
const unsigned long debounceDelay = 200;

void setup() {
  Serial.begin(9600);

  base.attach(pin_base);
  shoulder.attach(pin_shoulder);
  elbow.attach(pin_elbow);
  rotator.attach(pin_rotator);
  gripper.attach(pin_gripper);

  pinMode(btn_key1, INPUT_PULLUP);
  pinMode(btn_key2, INPUT_PULLUP);

  loadFromEEPROM();
  applyServoAngles();
}

void loop() {
  // === Зчитування значень ===
  int x1 = analogRead(x_key1);
  int y1 = analogRead(y_key1);
  int x2 = analogRead(x_key2);
  int y2 = analogRead(y_key2);
  bool btn1 = digitalRead(btn_key1) == LOW;
  bool btn2 = digitalRead(btn_key2) == LOW;

  // === Переключення режиму (антидребезг) ===
  if (btn1 != lastButton1State) {
    lastDebounceTime = millis();
  }

  if ((millis() - lastDebounceTime) > debounceDelay) {
    if (btn1 != mode) {
      mode = !mode;
      EEPROM.update(ADDR_MODE, mode);
      Serial.print("Mode switched to: ");
      Serial.println(mode == 0 ? "Manual" : "Coordinate");
      delay(500);
    }
  }
```

```
        lastButton1State = btn1;

        // === Режим 0: Ручне керування ===
        if (mode == 0 && millis() - lastMoveTime > moveDelay) {
          // === Керування 360° сервоприводом base ===
          if (x1 < threshold) {
            base.write(0);   // обертання вліво
          } else if (x1 > 1023 - threshold) {
            base.write(180); // обертання вправо
          } else {
            base.write(90);  // стоп
          }

          // === Інше серво по крокам ===
          if (y1 < threshold) shoulderAngle = constrain(shoulderAngle - step, 0,
180);
          if (y1 > 1023 - threshold) shoulderAngle = constrain(shoulderAngle +
step, 0, 180);

          if (x2 < threshold) elbowAngle = constrain(elbowAngle - step, 0, 180);
          if (x2 > 1023 - threshold) elbowAngle = constrain(elbowAngle + step,
0, 180);

          if (y2 < threshold) rotatorAngle = constrain(rotatorAngle - step, 0,
180);
          if (y2 > 1023 - threshold) rotatorAngle = constrain(rotatorAngle +
step, 0, 180);

          if (btn2) gripperAngle = (gripperAngle == 0) ? 45 : 0;

          applyServoAngles();
          saveToEEPROM();
          lastMoveTime = millis();
        }

        // Прийом команд с ПК
        if (Serial.available()) {
          String input = Serial.readStringUntil('\n');
          input.trim();

          if (input.startsWith("SET")) {
            int s, e, r, g;
            int n = sscanf(input.c_str(), "SET S:%d E:%d R:%d G:%d", &s, &e, &r,
&g);
            if (n == 4) {
              shoulderAngle = constrain(s, 0, 180);
              elbowAngle    = constrain(e, 0, 180);
              rotatorAngle  = constrain(r, 0, 180);
              gripperAngle  = constrain(g, 0, 180);
```

```
            applyServoAngles();
            saveToEEPROM();
          }
        }
      }

      sendAnglesOverSerial();
      delay(100);  // затримка для стабільної передачі

      sendAnglesOverSerial();
    }

    // === Встановлення кутів ===
    void applyServoAngles() {
      // base — 360° сервопривід, керується окремо
      shoulder.write(shoulderAngle);
      elbow.write(elbowAngle);
      rotator.write(rotatorAngle);
      gripper.write(gripperAngle);
    }

    // === EEPROM ===
    void saveToEEPROM() {
      EEPROM.update(ADDR_SHOULDER, shoulderAngle);
      EEPROM.update(ADDR_ELBOW, elbowAngle);
      EEPROM.update(ADDR_ROTATOR, rotatorAngle);
      EEPROM.update(ADDR_GRIPPER, gripperAngle);
    }

    void loadFromEEPROM() {
      shoulderAngle = EEPROM.read(ADDR_SHOULDER);
      elbowAngle    = EEPROM.read(ADDR_ELBOW);
      rotatorAngle  = EEPROM.read(ADDR_ROTATOR);
      gripperAngle  = EEPROM.read(ADDR_GRIPPER);
      mode          = EEPROM.read(ADDR_MODE);
    }

    // === Відправка кутів в Serial ===
    void sendAnglesOverSerial() {
      Serial.print("S:"); Serial.print(shoulderAngle);
      Serial.print(" E:"); Serial.print(elbowAngle);
      Serial.print(" R:"); Serial.print(rotatorAngle);
      Serial.print(" G:"); Serial.print(gripperAngle);
      Serial.print(" | Mode: "); Serial.println(mode == 0 ? "Manual" :
"Coordinate");
    }
```

ДОДАТОК В

Демонстраційний матеріал

№ докумен-та	Позначення	Найменування	Додаткові відомості
		Текстові документи	
1	ГЮИК. 503320.001ВД	Пояснювальна записка	А4, 106 с.
		Додаткові матеріали	
2		Публікація	А4, 4 с.
3		Код кінематичної моделі	А4, 30 с.
4		Демонстраційний матеріал	А4, 15 с.

					ГЮИК. 503320.001ВД			
Змін.	Арк.	Номер докум.	Підпис	Дата				
Розроб.		Лазаренко М. Ю.			Розроблення програмного забезпечення моделі кінематичного 3-лакового маніпулятора	Літера	Аркуш	Аркушів
Перевір.		Цимбал О. М.				Н		1
						Кафедра КІТАР ХНУРЕ		
Н.контр.		Стародубцев М. Г.						
Затв.		Невлюдов І.Ш.						