ДОДАТОК А

Вихідний код програми

```
using MultyDOF_Robot.Properties;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Reflection;
using System.Reflection.Emit;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement.Button;

namespace MultyDOF_Robot
{
    public partial class Form1 : Form
    {
        public static String mPath = AppDomain.CurrentDomain.BaseDirectory;
        public static String mDBPath = mPath + "Manipulator.db";

        bool isDrawAllAngle = true;
        bool _isPointPressed = false;
        int pressX = 0;
        int pressY = 0;

        double angleRotate = 0;
        double vectorAngle = 0;
```

```
int defaultSpeed = 1;
double defaultScale = 2;
double lengthMove = 0;

int maxPointNum = 0;
public static int selectPointID = -1;

List<Points> points = new List<Points>();

public Form1()
{
   InitializeComponent();
}

private void Form1_Load(object sender, EventArgs e)
{
   isDrawAllAngle = false;
   points.Clear();
   LoadPoint();
}

private void pintMouseMove(object sender, MouseEventArgs e)
{
   if (_isPointPressed)
   {
      if (e.Button == System.Windows.Forms.MouseButtons.Left)
      {
         //label6.Text = String.Format("x = {0}; y = {1}", e.X, e.Y);
         PictureBox p = (PictureBox)sender;
         p.Left = e.X + p.Left - pressX;
```

```
            p.Top = e.Y + p.Top - pressY;

            Points point = getPointByID((int)(sender as PictureBox).Tag);
            point.left = p.Left;
            point.top = p.Top;
            myPanel1.Refresh();

        }
    }
}

private void pointMouseUp(object sender, MouseEventArgs e)
{
    _isPointPressed = false;
    Points point = getPointByID((int)(sender as PictureBox).Tag);
    if (point == null) return;
    PictureBox p = (PictureBox)sender;
    point.left = p.Left;
    point.top = p.Top;
    myPanel1.Refresh();
    updateDataPoint();

    //update current point
    updateAnglePointFrom(point);

    //update point fromTo
    Points fromPoint = getPointByID(point.idPointFromConnect);
    if (fromPoint != null) updateAnglePointFrom(fromPoint);

    lengthMove = getLength(point);
```

```
    label10.Text = Math.Round(lengthMove, 2).ToString();
}

private void pointMouseDown(object sender, MouseEventArgs e)
{
    Points p = getPointByID((int)(sender as PictureBox).Tag);
    if (p == null) return;

    _isPointPressed = true;
    //curPointLeft = p.left;
    //curPointTop = p.top;
    pressX = e.X;
    pressY = e.Y;
    if (selectPointID > -1) unSelectPoint(selectPointID);
    selectPointID = p.idPoint;
    p.point.Image = Resources.point_red;

    angleRotate = getAngleRotate(p);
}

private void unSelectPoint(int idPoint)
{
    foreach (Points p in points)
    {
        if (p.idPoint == idPoint)
        {
            p.point.Image = Resources.point_blue;
        }
    }
}
```

```
private Points getPointByID(int id)
{
    Points res = null;

    foreach (Points p in points)
    {
        if (p.idPoint == id) return p;
    }

    return res;
}

private void deletePointByID(int id)
{
    foreach (Points p in points)
    {
        if (p.idPoint == id)
        {
            points.Remove(p);
            return;
        }
    }
}

double getLength(Points point)
{
    double length = 0;

    Points p_to = getPointByID(point.idPointToConnect);
```

```
            if (p_to == null) return 0;

            length = Math.Sqrt(Math.Pow((p_to.top - point.top), 2) +
Math.Pow((p_to.left - point.left), 2));

            return length * defaultScale;
        }

        private void updateAnglePointFrom(Points point)
        {
            double angleRotate = getAngleRotate(point);
            double lengthMove = getLength(point);
            int ID_Point = point.idPoint;

            //Delet Turn and Move
            String sql = "BEGIN TRANSACTION; ";
            sql += String.Format("Delete from Script where ID_Point = '{0}' and
Label = 'L_Turn'; ", ID_Point);
            sql += String.Format("Delete from Script where ID_Point = '{0}' and
Label = 'L_Move'; ", ID_Point);
            sql += "COMMIT;";
            DataBase.Exec_SQL(sql);

            //Find max NumStep
            String sNumStep =
DataBase.Find_First_Value(String.Format("Select * from Script where ID_Point =
'{0}' order by NumStep DESC; ", ID_Point), "NumStep");
            int maxNumStep = Int32.Parse(sNumStep);

            //Add L_Turn
```

```
            maxNumStep++;
            string dir = "right";
            if (angleRotate < 0)
            {
               dir = "left";
            }
            string param_turn = String.Format("command=turn; dir={0}; speed={1}; degree={2}", dir, defaultSpeed, Math.Abs((int)angleRotate));
            String sql_turn = String.Format("Insert into Script (NumStep, Command, Param, Label, ID_Point) " +
                  "Values ('{0}', '{1}', '{2}', '{3}', '{4}')", maxNumStep, "Turn", param_turn, "L_Turn", ID_Point);
            DataBase.Exec_SQL(sql_turn);

            //Add L_Move
            maxNumStep++;
            string dir_move = "forvard";
            string param_move = String.Format("command=move; dir={0}; speed={1}; length={2}", dir_move, defaultSpeed, (int)lengthMove);
            String sql_move = String.Format("Insert into Script (NumStep, Command, Param, Label, ID_Point) " +
                  "Values ('{0}', '{1}', '{2}', '{3}', '{4}')", maxNumStep, "Move", param_move, "L_Move", ID_Point);
            DataBase.Exec_SQL(sql_move);

        }

        private void myPanel1_Paint(object sender, PaintEventArgs e)
        {
```

```
            e.Graphics.SmoothingMode =
System.Drawing.Drawing2D.SmoothingMode.AntiAlias;

            foreach (Points p in points)
            {
               if (p.idPointFromConnect > 0)
               {
                  //get center point_A
                  int xA = p.left + 16;
                  int yA = p.top + 16;

                  //get center point_B
                  Points pB = getPointByID(p.idPointFromConnect);
                  if (pB != null)
                  {
                     int xB = pB.left + 16;
                     int yB = pB.top + 16;

                     //draw line
                     Pen connectPen = new Pen(Color.Yellow, 10);
                     if (!checkBox2.Checked) connectPen = new Pen(Color.Gray,
10);
                     e.Graphics.DrawLine(connectPen, xA, yA, xB, yB);
                  }
               }
               if (isDrawAllAngle)
               {
                  angleRotate = getAngleRotate(p);
                  drawAngle(e.Graphics, p, angleRotate);
               }
```

```
                else
                {
                    if (p.idPoint == selectPointID)
                    {
                        angleRotate = getAngleRotate(p);
                        drawAngle(e.Graphics, p, angleRotate);
                        label6.Text = String.Format("{0}", Math.Round(angleRotate,
2).ToString());
                        if (angleRotate < 0) label5.Text = "Left";
                        else label5.Text = "Right";
                        //label12.Text = selectPointID.ToString();
                    }
                }
            }
        }

        private void drawAngle(Graphics g, Points point, double angle)
        {

            Segment s1 = new Segment();
            Segment s2 = new Segment();

            //get segment S1
            Points p_from = getPointByID(point.idPointFromConnect);
            if (p_from == null)
            {
                s1.p1 = new Point(point.left, point.top + 100);
                s1.p2 = new Point(point.left, point.top);
            }
            else
```

```
                {
                    s1.p1 = new Point(p_from.left, p_from.top);
                    s1.p2 = new Point(point.left, point.top);
                }

                //get segment S2
                Points p_to = getPointByID(point.idPointToConnect);
                if (p_to == null) return;

                if (p_to == null)
                {
                    s2.p1 = new Point(point.left, point.top - 100);
                    s2.p2 = new Point(point.left, point.top);
                }
                else
                {
                    s2.p1 = new Point(p_to.left, p_to.top);
                    s2.p2 = new Point(point.left, point.top);
                }

                //draw vector
                vectorAngle = Math.Atan2(s1.p1.Y - s1.p2.Y, s1.p1.X - s1.p2.X) *
(180 / Math.PI);

                Point point_Fc = new Point(
                    point.left + 16 - (int)(Math.Cos(ConvertToRadians(vectorAngle)) *
100),
                    point.top + 16 - (int)(Math.Sin(ConvertToRadians(vectorAngle)) *
100));
```

```
            //float[] dashValues = { 1, 2, 1, 2 };
            float[] dashValues = { 4, 1, 4, 1 };
            Pen dotanglePen = new Pen(Color.Blue, 2);
            dotanglePen.DashPattern = dashValues;

            //Pen anglePen = new Pen(Color.Black, 1);
            g.DrawLine(dotanglePen, point.left + 16, point.top + 16, point_Fc.X,
point_Fc.Y);

            //draw arc
            if (Math.Abs(angleRotate) > 0.5)
            {
               g.DrawArc(dotanglePen, point.left - 16, point.top - 16, 64, 64,
(float)(vectorAngle - 180), (float)(angleRotate));
            }
         }

         public double ConvertToRadians(double angle)
         {
            return (Math.PI / 180) * angle;
         }

         private void checkBox2_CheckedChanged(object sender, EventArgs e)
         {
            if (checkBox2.Checked)
            {
               myPanel1.BackgroundImage = Resources.Manipulator;
            }
            else myPanel1.BackgroundImage = null;
         }
```

```
        private void button1_Click(object sender, EventArgs e)
        {
            Points newPoint = new Points();
            newPoint.left = 10;
            newPoint.top = 10;
            newPoint.idPoint = maxPointNum + 1;
            newPoint.idPointFromConnect = maxPointNum;
            addPoint(newPoint);

            String sql = String.Format("Insert Into Point (idPoint, idPointToConnect, " +
                "idPointFromConnect, namePoint, top, left) " +
                "values ('{0}', '{1}', '{2}', '{3}', '{4}', '{5}')",
                newPoint.idPoint, newPoint.idPointToConnect, newPoint.idPointFromConnect,
                newPoint.namePoint, newPoint.top, newPoint.left);
            DataBase.Exec_SQL(sql);

            //Update to point
            Points fromPoint = getPointByID(maxPointNum);
            fromPoint.idPointToConnect = maxPointNum + 1;

            String sql_toPoint = String.Format("Update Point set idPointToConnect = '{0}' " +
                "where idPoint = '{1}';", newPoint.idPoint, fromPoint.idPoint);
            DataBase.Exec_SQL(sql_toPoint);

            maxPointNum++;
```

```
    updateAnglePointFrom(fromPoint);
    drawPoint();
}

private void button3_Click(object sender, EventArgs e)
{
    if (selectPointID == -1) return;
    Points delP = getPointByID(selectPointID);
    int pointA = delP.idPointFromConnect;
    int pointB = delP.idPointToConnect;
    Points pointForm = getPointByID(pointA);
    Points pointTo = getPointByID(pointB);

    if (pointTo == null)
    {
        pointForm.idPointToConnect = 0;
    }
    else if (pointForm == null)
    {
        pointTo.idPointFromConnect = 0;
    }
    else
    {
        pointForm.idPointToConnect = pointTo.idPoint;
        pointTo.idPointFromConnect = pointForm.idPoint;
    }

    deletePointByID(delP.idPoint);

    //delete from DB
```

```
String sql = "BEGIN TRANSACTION; ";
if (pointTo == null)
{
  sql += String.Format("update Point set idPointFromConnect = '{0}', idPointToConnect = '{1}' where idPoint = '{2}'; ", pointForm.idPointFromConnect, pointForm.idPointToConnect, pointForm.idPoint);

}
else if (pointForm == null)
{
  sql += String.Format("update Point set idPointFromConnect = '{0}', idPointToConnect = '{1}' where idPoint = '{2}'; ", pointTo.idPointFromConnect, pointTo.idPointToConnect, pointTo.idPoint);
}
else
{
  sql += String.Format("update Point set idPointFromConnect = '{0}', idPointToConnect = '{1}' where idPoint = '{2}'; ", pointForm.idPointFromConnect, pointForm.idPointToConnect, pointForm.idPoint);
  sql += String.Format("update Point set idPointFromConnect = '{0}', idPointToConnect = '{1}' where idPoint = '{2}'; ", pointTo.idPointFromConnect, pointTo.idPointToConnect, pointTo.idPoint);
}
sql += String.Format("delete from Point where idPoint = '{0}'; ", delP.idPoint);
sql += "COMMIT;";
DataBase.Exec_SQL(sql);
LoadPoint();
```

```
            drawPoint();
        }

        private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {
            if (checkBox1.Checked) { isDrawAllAngle = true; }
            else { isDrawAllAngle = false; }
            myPanel1.Refresh();
        }
    }

    public class MyPanel : Panel
    {
        public MyPanel()
        {
            this.DoubleBuffered = true;
            this.ResizeRedraw = true;
        }
    }
}
```

ДОДАТОК Б

Апробація результатів наукових досліджень

Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра комп'ютерно-інтегрованих технологій, автоматизації та робототехніки

VIII Міжнародна Конференція
ВИРОБНИЦТВО
&
МЕХАТРОННІ СИСТЕМИ 2024

VIII International Conference
MANUFACTURING
&
MECHATRONIC SYSTEMS 2024

M&MS
2024
VII International Conference
25-26 October
Kharkiv

Вирішення задачі управління багатоланковим маніпулятором

Сергій Новоселов[1], Владислав Іванов[2]

1. Кафедра КІТАР, Харківський національний університет радіоелектроніки, УКРАЇНА, Харків, пр. Науки 14, email: sergiy.novoselov@nure.ua
2. Кафедра КІТАР, Харківський національний університет радіоелектроніки, УКРАЇНА, Харків, пр. Науки 14, email: vladyslav.ivanov1@nure.ua

Анотація: Дана робота присвячена створенню рішення задачі керування багатоланковим маніпулятором. Проведено аналіз існуючих методів управління багатоланковими маніпуляторами. Наведена узагальнена структурна схема системи управління. Проведено аналіз конструкцій суглобів багатоланкових маніпуляторів та методів зчитування даних про позиціонування рухомих ланок для вирішення завдання оптимізації часу руху заданою траєкторією маніпулятора.

Ключові слова: Багатоланковий маніпулятор, робот, абсолютний енкодер.

I. Вступ

Багатоланковий маніпулятор – це маніпулятор з кількома ступенями рухливості [1]. Наприклад, як конструкція, що володіє одним ступенем рухливості може бути поворот однієї ланки щодо іншої або обертання циліндричного стрижня навколо власної осі симетрії, а також зворотно-поступальні рухи переміщення циліндричного стрижня навколо цієї осі (рис. 1) [1].

Рис. 1. Схема багатоланкового маніпулятора

Для промислових роботів і модульних маніпуляторів найбільш поширеним підходом до управління є незалежні спільні контролери, наприклад, за допомогою схем керування PD або PID [1]. Однак ці децентралізовані лінійні структури керування, що ігнорують нелінійну поведінку таких систем, зазвичай не враховують інформацію з інших підсистем або вимагають ретельного налаштування параметрів контролера.

Іншим широко використовуваним методом керування, особливо при відстеженні траєкторії для жорстких багатоланкових маніпуляторів, є так званий метод обчисленого крутного моменту [2], який є окремим випадком лінеаризаційного керування зворотним зв'язком, що використовує метод зворотної динаміки. Тим не менш, продуктивність цієї технології прямого керування на основі моделі сильно залежить від якості моделювання динаміки робота. Подальшою передовою технікою керування для нелінійних систем є прогнозне керування за моделлю (ПКМ), також відоме як прогнозне керування або керування горизонтом, яке міцно закріпилося в промисловому застосуванні.

В роботі [3] показано, що для вирішення завдання оптимізації часу руху заданою траєкторією маніпулятора, необхідно не тільки точно оцінити швидкість руху вузлів маніпулятору, але і забезпечити лінійну характеристику оцінки позиції механізму в широкому діапазоні зміни швидкості. Також важливим є визначення абсолютного кутового положення елементів конструкції маніпулятора, особливо під час взаємодії об'єктів промислової автоматизації з використанням технології Internet of Things В даній роботі запропоновано конструкцію суглоба маніпулятора та дві конструкції датчиків для визначення абсолютного кута оберту суглоба: резистивний і магнітний (рис. 2).

Рис. 2. Зовнішній вигляд суглобу робота-маніпулятора

Запропонована конструкція резистивного датчика виявилася нетехнологічною і набагато більша за розмірами ніж конструкція магнітного датчика, тому в дослідженнях вона не використовувалась.

Таким чином, дослідження методів визначення позиції елементів конструкції маніпулятора в будь-який момент часу є актуальним завданням [3].

II. Автоматизована система керування багатоланковим маніпулятором

Промислові роботи мають, як правило, більшу кількість ступенів рухливості. В умовах реального виробництва застосовуються роботи з чотирма і більше ступенями рухливості. Управління

маніпуляторами з таким числом ступенів рухливості є більш складним, чим при керуванні маніпуляторами з меншим числом ступенів рухливості. Таким чином, при виконанні найпростішої операції переміщення захватного пристрою з однієї точки в іншу потрібно виконати великий обсяг обчислень.

При моделюванні систем управління рухом роботів потрібно вирішувати завдання кінематики та динаміки для їх виконавчих механізмів. Існує зворотне та пряме завдання кінематики. Пряме завдання кінематики полягає у визначенні просторового положення та орієнтації характерної точки, як правило, робочого інструменту маніпулятора робота за відомими значеннями узагальнених координат. Зворотне завдання кінематики, як і пряме завдання, є одним із основних завдань кінематичного аналізу та синтезу. Для керування положенням ланок та орієнтацією робочого інструменту маніпулятора виникає необхідність вирішення зворотного завдання кінематики.

Структурна схема модуля управління маніпулятором показана на рисунку 3.

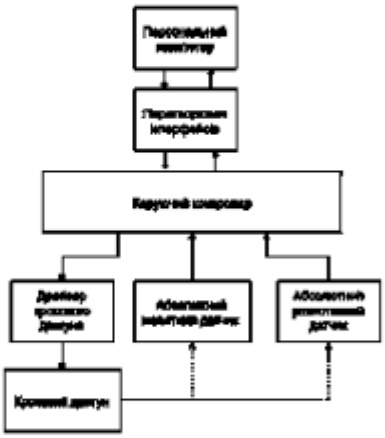

Рис. 3. Структурна схема модуля управління маніпулятором

Система складається з наступних елементів:

– персональний комп'ютер (ПК) з програмою управляння і отримання даних;
– перетворювач інтерфейсів;
– керуючий контролер;
– драйвер двигуна постійного струму;
– абсолютний магнітний датчик;
– абсолютний резистивний датчик;
– кроковий двигун;
– макет механізму маніпулятора.

Для визначення орієнтації суглобу будемо використовувати два методи:

– магнітний абсолютний;
– резистивний абсолютний.

Для визначення абсолютного кута оберту суглоба будемо використовувати магнітний енкодер типу AS5600. Даний енкодер потребує додаткового елементу – магніту, який буде закріплено на конструктивних елементах суглобу на одній вісі з редуктором. Датчик AS5600 повинен точно розташовуватися на вісі редуктора і, відповідно, постійного магніту. Відстань між датчиком і постійним магнітом має бути мінімальна й не перевищувати 4 мм. В даній конструкції датчик є жорстко закріпленим відносно рухомої частини суглобу на основі макету.

Узагальнена структурна схема системи управління показана на рисунку 4.

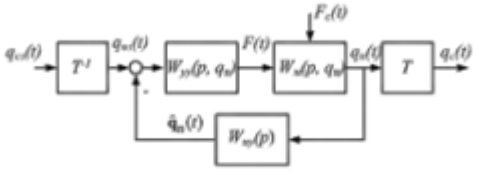

Рис. 4. Узагальнена структурна схема системи управління

Програмування траєкторії руху захвату 5 з навантаженням 6 виконується у системі фіксованої базової координат. Вектор узагальненого зчеплення, як функція процесу qc(t), утворюється вектор узагальнених координат ланок маніпулятора, який у жорсткій механічній системі є вектором узагальнених координат приводів qn(t). Вектори qc(T) та qn(t) пов'язані один з одним, відображенням перетворення координат T, вміст якого встановлюється з рішення прямої задачі кінематики маніпулятора. Пристрій керування програмним забезпеченням генерує вектор упорядкованих узагальнених координат приводу $q_{n.z}(T)$, які пов'язані з вектором узагальнених координат захвату маніпулятора $q_{c.z}(T)$ шляхом зворотного відображення перетворення координати T^{-1}. Вміст T^{-1} встановлюється із рішення зворотної задачі кінематики маніпулятора.

У загальному випадку динамічні властивості маніпулятора характеризуються матрицею Wm(p, qn), згідно з якою виконується синтез матриці контрольних пристроїв Wyy(p, qn). В узагальненій матричній структурній схемі системи керування зчепленням маніпулятора F(t), Fc(T) – вектори сил, що застосовуються до зв'язків маніпулятора та опору; q`n(t) - виміряні векторні значення узагальнених координат приводів; Wyy (p, qn) - матриця інформаційних пристроїв. На додаток до основних зворотних зв'язків щодо узагальнених координат приводів, система може мати зворотний зв'язок про проміжні координати, зроблені відповідно до принципів підпорядкованого управління.

III. Розробка алгоритму роботи програми

На рисунку 5 показано алгоритм роботи макету.

На початку роботи перевіряється зв'язок з персональним комп'ютером через послідовний інтерфейс. Якщо зв'язок установлено, то можна продовжувати роботу.

Отримавши команду від ПК, контролер формує послідовність імпульсів для зміщення валу крокового

двигуна на потрібний кут, враховуючи передавальне число редуктора. В нашому випадку це число становить 1:42.

Кут зміщення валу двигуна при повному кроці становить 1,8 градусів. Враховуючи передавальне число редуктора, теоретично механізм може забезпечити точність 0,045 градуси.

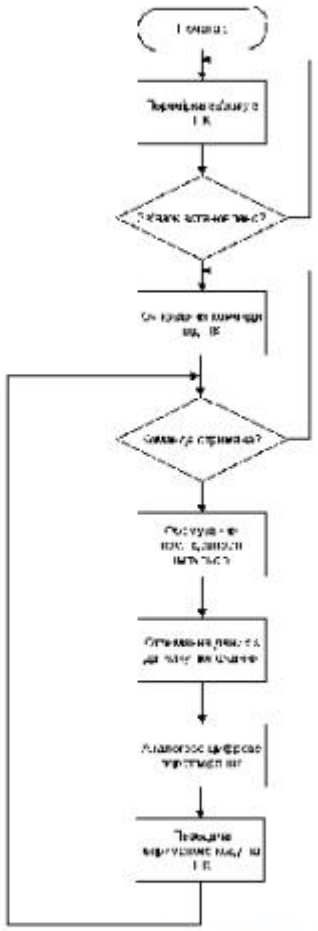

Рис. 5. Алгоритм роботи програми контролера

Після завершення переміщення, контролер зупиняє редуктор та зчитує показання датчику положення. Отриманні дані перетворюються з аналогового сигналу в цифровий та передаються на персональний комп'ютер.

Після завершення вимірювання куту оберту цикл управління повторюється до вимикання напруги живлення контролеру.

VII. Висновки

Дана робота присвячена створенню рішення задачі керування багатоланковим маніпулятором. Проведено аналіз існуючих методів управління багатоланковими маніпуляторами. Наведена узагальнена структурна схема системи управління. Проведено аналіз конструкцій суглобів багатоланкових маніпуляторів та методів зчитування даних про позиціонування рухомих ланок для вирішення завдання оптимізації часу руху заданою траєкторією маніпулятора. Отримані дані свідчать про досить точне визначення куту за допомогою магнітного абсолютного енкодеру. Похибка виміру куту становить менше 1,4 градуси.

Перелік посилань

[1] Fehr, Jörg, Patrick Schmid, Georg Schneider, and Peter Eberhard. 2020. "Modeling, Simulation, and Vision-/MPC-Based Control of a PowerCube Serial Robot" Applied Sciences 10, no. 20: 7270. https://doi.org/10.3390/app10207270

[2] Schiehlen, W.; Eberhard, P. Applied Dynamics; Springer: Heidelberg, Germany, 2014.

[3] I. Nevlyudov/ Development and study of the operation of the module for determining the orientation of the manipulator joint / I. Nevlyudov, S. Novoselov, O. Sychova. // Innovative technologies and scientific solutions for industries. 2022. No. 2 (20). pp 86-96 DOI: https://doi.org/10.30837/ITSSI.2022.20.086

[4] Невлюдов І. Ш., Андрусевич А. О., Євсєєв В. В., Новоселов С. П., Демська Н. П. Проектування мобільних маніпуляційних роботів: Монографія. – Х. :2022. – 427 с

[5] Невлюдов І.Ш. Пневматичні пристрої та засоби автоматизації мехатронних систем: Навчальний посібник / І.Ш. Невлюдов, Л.О. Кривопляс-Володіна, С.П. Новоселов, О.В. Сичова. – Харків: ХНУРЕ, 2020 . – 244 с.

[6] Багатоланковий маніпулятор – Вікіпедія. Вікіпедія. URL: https://uk.wikipedia.org/wiki/Маніпулятор (дата звернення: 10.01.2024).

[7] Невлюдов І.Ш. Електропневмоавтоматичні приводи в автоматизованих системах керування: Навчальний посібник / І.Ш. Невлюдов, Л.О. Кривопляс-Володіна, С.П. Новоселов, О.В. Сичова. – Харків: ХНУРЕ, 2021 . – 292 с..

[8] V.V.M.J. Satish Chembuly, Hari Kumar Voruganti, Trajectory Planning of Redundant Manipulators Moving along Constrained Path and Avoiding Obstacles, Procedia Computer Science, Volume 133, 2018, Pages 627-634.

[9] Igor Nevliudov, Serhii Novoselov, Oksana Sychova. Control automation of assembly operations using a computer vision system in intelligent production. Journal of natural sciences and technologies. 2023, 2(2), pp. 173-182, DOI: 10.5281/zenodo.8098567

ДОДАТОК В

ДЕМОНСТРАЦІЙНИЙ МАТЕРІАЛ

№	Позначення	Найменування	Дод. відомості
		Текстові документи	
1	ГЮИК. 464430.008 ПЗ	Пояснювальна записка	80 с., ф. А4
		Графічні документи	
2		Тези з конференції	6 с., ф. А4
3		Лістинг програми	

					ГЮИК. 464430.008 ВД			
Змін.	Арк.	№ документа	Підп.	Дата				
Розроб.		Іванов В.С.			Проектування та дослідження багатоланкової конструкції промислового маніпулятора	Літ.	Аркуш	Аркушів
Перевір.		Новоселов С.П.				Н		1
Н. контр.		Бронніков А.І.				ХНУРЕ Кафедра КІТАР		
Затвер.		Невлюдов І.Ш						