**Районная учебно-исследовательская конференция**

**«Юность Пинежья»**

Направление: **Программирование и информационные технологии.**

**Управление манипулятором с плоскопараллельной кинематикой на основе чтений позиций сервоприводов и воспроизведения записанных позиций из массива**

**Исследовательская работа**

Выполнена ученицей 11 класса

МБОУ «Междуреченская СШ №6»

Москвиной Дарьей Вадимовной

Научный руководитель – учитель

МБОУ «Междуреченской СШ №6»

Игнатьев Павел Алексеевич

**с. Карпогоры, 2022**

**Оглавление**

[Введение 3](#_Toc91659926)

[Описание модели манипулятора с плоскопараллельной кинематикой 4](#_Toc91659927)

[Описание работы программы 6](#_Toc91659928)

[Заключение 9](#_Toc91659929)

[Библиографический список 10](#_Toc91659930)

[Приложение 1. Первая программа для работы устройства. 11](#_Toc91659931)

[Приложение 2. Вторая программа для работы устройства. 12](#_Toc91659932)

[Приложение 3. Внешний вид устройства. 13](#_Toc91659933)

# Введение

В настоящее время манипуляторы используются [3] во всех отраслях промышленности. Они используются на производственных процессах, где требуются высокая точность выполнения операций, на опасных и вредных производствах и прочее. Изучение темы по данной проблематике приобретает особую актуальность в инженерном образовании. Поэтому мы решили собрать модель манипулятора и изучить принцип управления им.

Цель – создать модель манипулятора и изучить принцип управления данной моделью.

Объект исследования – манипулятор с плоскопараллельной кинематикой.

В соответствии с данной целью были поставлены следующие задачи:

1. Создать и описать модель манипулятора с плоскопараллельной кинематикой.
2. Описать программу и способ управления манипулятором.

Методы исследования: теоретическое и практическое моделирование, эксперимент.

Гипотеза – с помощью записи позиций сервоприводов и воспроизведения позиций из массива можно управлять манипулятором с плоскопараллельной кинематикой.

# Описание модели манипулятора с плоскопараллельной кинематикой

В настоящее время используются манипуляторы с различными кинематическими схемами [4]: манипулятор с угловой кинематикой, манипулятор с плоскопараллельной кинематикой, манипулятор с DELTA кинематикой и прочее. Наш манипулятор состоит из: основания с сервоприводом поворота, двумя сервоприводами, управляющими рычажной системой с изогнутыми тягами и устройством схвата, на раме которого также находится сервопривод. Детали корпуса манипулятора металлические, созданные с помощью лазерной резки. (Приложение 3)

В качестве исполнительных механизмов мы использовали сервоприводы DYNAMIXEL [11] AX-12A, представляющие собой модели промышленных автоматизированных приводов со встроенной системой управления. В таких сервоприводах в одном корпусе объединены двигатель постоянного тока, редуктор и системы управления. Применение данного типа сервоприводов позволяет разрабатывать модели манипуляционных роботов с различными типами кинематической схемы, обладающих высокой точностью и динамикой движения. В качестве интерфейса подключения сервоприводов используется TTL шина [2], со стандартными 3х пиновыми коннекторами molex (PIN1:GND, PIN2:VDD, PIN3: Data «информационный»). У каждого сервопривода есть ID – это номер, по которому идёт обращение к приводу. Всего в одной сети может быть до 254 приводов. Очень важно выставить данный параметр корректно, если в одной сети будет несколько приводов с одинаковым ID, то произойдёт ошибка. Для присвоения ID мы использовали программу Dynamixel Wizard 2.0 [8]. Сервопривод был подключен к компьютеру с помощью адаптера USB-DXL-AR, а питание на него подавалось с помощью преобразователя SMPS2Dynamixel [7]. В интерфейсе программы мы присвоили ID для сервоприводов манипулятора с первого по четвёртый. Первый сервопривод поворачивает манипулятор в горизонтальной плоскости (привод поворота), второй и третий управляет скобами привода ноги и рычага соответственно для поворота в вертикальной плоскости, а четвёртый сервопривод управляет схватом.

Манипулятором будет управлять контроллер OpenCM9.04[5]. Основой этой аппаратной платформы является 32х битный микроконтроллер ARM Cortex-M3. Контроллер находится на периферийной плате STEM Board на которой находятся 3х-пиновый разъём для подключения внешних Dynamixel-совместимых устройств по TTL интерфейсу (3х-пиновый интерфейс для правления совместимыми Dynamixel-устройствами). Данный разъём управляется контроллером OpenCM9.04 физической шиной 3 (#define DEVICE\_NAME «3»). На плату расширения подаётся напряжение 12 вольт для питания платы контроллера и сервоприводов. Силовой частью платы мы можем управлять с помощью отдельного включателя. [12]

Для проверки гипотезы наш манипулятор будет перемещать схват в пространстве вместе с предметом.

# Описание работы программы

Для проверки гипотезы мы использовали две программы: первая программа необходима для чтения позиций сервоприводов, а вторая для воспроизведения их из массива данных.

Опишем первую программу (Приложение 1). С помощью данной программы мы получим положение четырёх сервоприводов в каждой конкретной точке пространства через монитор порта. Первой строкой #include <Dynamixel2Arduino.h> [1] подключаем библиотеку Dynamixel для управления нашими сервоприводами AX-12A [10], которые используются в модели. Управление сервоприводами происходит через физическую шину 3 [13], мы использовали #define DXL\_SERIAL Serial3. Чтобы воздействовать на движение сервоприводов используем строку #define DEBUG\_SERIAL Serial. Далее вводим номер пина строкой const uint8\_t DXL\_DIR\_PIN = 22. Строкой const float DXL\_PROTOCOL\_VERSION = 1.0 вводим переменную, которая отвечает за передачу данных от контроллёра OpenCM9.04 к сервоприводам. Затем строкой Dynamixel2Arduino dxl(DXL\_SERIAL, DXL\_DIR\_PIN) создаём команды для движения сервоприводов. Вводим строки, обозначающие количества сервоприводов #define jointN 4, переменные int pos=0, int i=0. Равномерный массив [9] инициализируется строкой int buf[jointN+1 ]. Со строки void setup() выполняется часть программы. Строкой DEBUG\_SERIAL.begin(57600) устанавливаем скорость обмена данными в порту компьютера. Устанавливаем скорость обмена данными в порту компьютера строкой dxl.begin(1000000). Выбираем протокол обмена данными dxl.setPortProtocolVersion(DXL\_PROTOCOL\_VERSION). Строкой for(i=1;i<jointN+1;i++) фиксируем изменения переменнлй i от 1 до 4 с шагом 1. Отдельной строкой dxl.setOperatingMode(i,OP\_POSITION) выполняется отдельная часть, в которой устанавливается режим работы сервопривода с переменной i в качестве шарнира. После со строки void loop() начинается основной цикл программы. Строкой for(i=1;i<=jointN;i++) устанавливаем изменения переменной i от 1 до 4 с шагом 1. Далее строкой pos = dxl.getPresentPosition(i) получаем позицию сервопривода с номером переменной i и запись в переменную pos. Происходит запись переменной значения pos в i-тый элемент массива buf[i]=pos. Строкой dxl.torqueOff(i) отключается блокировка сервопривода с переменной i. В следующей строке отдельной части Serial.print("{0,") вводятся в монитор порта переменные позиций манипулятора. Здесь же строкой for(i=1;i<=jointN;i++) устанавливаем изменения переменной i от 1 до 4 с шагом 1. Вводим в отдельной части Serial.print(buf[i]) значения ячейки массива с переменной i. Далее вводится Serial.print(","). В новой части вводится Serial.print("}"). После этого в строке вводится Serial.println(' ') в монитор порта [6] текст «» и переносится курсор на следующую строку. Завершая первую программу, следующей строкой delay(1000) выдерживается пауза в 1 секунду.

Мы вручную задаём некоторую позицию манипулятора (поворачивая манипулятор рукой и придерживая его). В монитор порта через каждую секунду вводится текущая конфигурация манипулятора. Когда значения построчно начинают повторяться, то манипулятор вручную установлен в целевую позицию. В мониторе порта необходимо выключить автопрокрутку, а затем, выделив курсором одну из нужных строчек (обозначающих целевую позицию), скопировать строку (Ctrl+C). Далее эту строку необходимо будет вставить в программу для воспроизведения позиций.

Далее мы рассмотрим программу для воспроизведения позиций (Приложение 2). Наш манипулятор будет перемещать предмет с помощью схвата на некоторое расстояние. Программа будет выполняться единожды. Для повторения действий необходимо будет нажать на кнопку Reset. Начало программы идентично предыдущей. Так как мы будем использовать чтение из массива, то строкой #define pages 10 указали количество ненулевых строк. Далее ввели переменные-счётчики int i=0 и int j=1. Строкой int buf[pages+1][jointN+1]={ инициализировали двумерный массив размера (10+1=11 на 5+1=6, 11 на 6). Со следующей сроки элементы начинается с 0. Но нулевой элемент мы использовать не будем, так как начнём с первого. Строки массива имеют вид {0,502,434,497,286,}, где каждая цифра это «целевая позиция» [14] сервопривода. То есть угловая позиция, в которую приводу нужно переместиться. При включении сервопривода данная ячейка памяти пустая. При её заполнении привод сразу начнёт вращаться, чтобы достичь заданной позиции. Параметр может принимать значения от 0 до 1023. Со строки void setup() выполняется часть программы. Строкой DEBUG\_SERIAL.begin(57600) устанавливаем скорость обмена данными в порту компьютера. Устанавливаем скорость обмена данными в порту компьютера строкой dxl.begin(1000000). Выбираем протокол обмена данными dxl.setPortProtocolVersion(DXL\_PROTOCOL\_VERSION). Строкой for(i=1;i<jointN+1;i++) фиксируем изменения переменнлй i от 1 до 4 с шагом 1. Отдельной строкой dxl.setOperatingMode(i,OP\_POSITION) выполняется отдельная часть. Далее for(j=1;j<=pages;j++) выполняет цикл с изменением переменной j от 1 до 6 с шагом 1, а следующей for(i=1;i<=jointN;i++) выполняется цикл с изменением переменной i от 1 до 6 с шагом 1. Отдельной строкой задаём целевую скорость 50 сервоприводу с номером i dxl.setGoalVelocity(i,50). Далее строкой dxl.setGoalPosition(i,buf[j][i]) задаём целевое положение, получаем из элемента ([j],[i]) сервоприводу с номером i. Завершая вторую программу, следующей строкой delay(1000) выдерживается пауза в 1 секунду.

# Заключение

В ходе проделанной работы удалось создать модель манипулятора и изучить принцип управления данной моделью. Уяснены принцип и сущность работы манипулятора с плоскопараллельной кинематикой. Созданная программа позволяет управлять манипулятором с плоскопараллельной кинематикой. Манипулятор имеет большое практическое значение. В дальнейшем устройство может быть модернизировано: с другой кинематикой, датчиками, добавлена возможность работы по беспроводному управлению.

# Библиографический список

1. Dynamixel2Arduino.h [Электронный ресурс] URL: <https://www.arduino.cc/reference/en/libraries/dynamixel2arduino/>
2. TTL шина [Электронный ресурс] URL: <https://www.lcard.ru/lexicon/ttl_in_out>
3. Использование манипуляторов [Электронный ресурс] URL: <https://avtokrany.guru/mesta-primeneniya/kranov-manipulyatorov>
4. Кинематические схемы [Электронный ресурс] URL: <https://stankiexpert.ru/tehnologii/skhema-kinematicheskaya.html>
5. Контроллер OpenCM9.04 [Электронный ресурс] URL: <https://robotgeeks.ru/collection/robotis/product/opencm904-c>
6. Монитор порта [Электронный ресурс] URL: <https://роботехника18.рф/монитор-порта-ардуино/#:~:text=М>
7. Преобразователь SMPS2Dynamixel [Электронный ресурс] URL: <https://компания-партнер.рус/products/preobrazovatel-smps2dynamixel>
8. Программа Dynamixel Wizard 2.0 [Электронный ресурс] URL: <https://pcmac.download/ru/app/1471288434/dynamixel-wizard-2-0>
9. Равномерный массив [Электронный ресурс] URL: <https://www.sites.google.com/site/sistprogr/lekcii1/lek10>
10. Сервоприводы AX-12A [Электронный ресурс] URL: <http://www.intechnics.ru/servoprivod.html?yclid=7772712752139788644>
11. Сервоприводы DYNAMIXEL [Электронный ресурс] URL: <http://robot-develop.org/archives/835>
12. СТЕМ Мастерская. Часть 1 / ООО «Прикладная роботоехника» - Электронная книга, 2020.
13. Физическая шина 3 [Электронный ресурс] URL: <https://studfile.net/preview/1650087/page:9/#:~:text=%2B%20%2B7.1.1%>
14. Целевая позиция [Электронный ресурс] URL: <https://alexgyver.ru/servosmooth/>

# 

# Приложение 1. Первая программа для работы устройства.

#include <Dynamixel2Arduino.h>

#define DXL\_SERIAL Serial3

#define DEBUG\_SERIAL Serial

const uint8\_t DXL\_DIR\_PIN = 22;

const float DXL\_PROTOCOL\_VERSION = 1.0;

Dynamixel2Arduino dxl(DXL\_SERIAL, DXL\_DIR\_PIN);

#define jointN 4

int pos=0;

int i=0;

int buf[jointN+1];

void setup() {

DEBUG\_SERIAL.begin(57600);

dxl.begin(1000000);

dxl.setPortProtocolVersion(DXL\_PROTOCOL\_VERSION);

for(i=1;i<jointN+1;i++)

{

dxl.setOperatingMode(i,OP\_POSITION);

}

}

void loop() {

for(i=1;i<=jointN;i++)

{

pos = dxl.getPresentPosition(i);

buf[i]=pos;

dxl.torqueOff(i);

}

Serial.print("{0,");

for(i=1; i<=jointN; i++)

{

Serial.print(buf[i]);

Serial.print(",");

}

Serial.print("}");

Serial.println(' ');

delay(1000);

}

# Приложение 2. Вторая программа для работы устройства.

#include <Dynamixel2Arduino.h>

#define DXL\_SERIAL Serial3

#define DEBUG\_SERIAL Serial

const uint8\_t DXL\_DIR\_PIN = 22;

const float DXL\_PROTOCOL\_VERSION = 1.0;

Dynamixel2Arduino dxl(DXL\_SERIAL, DXL\_DIR\_PIN);

#define jointN 5

#define pages 10

int i=0;

int j=1;

int buf[pages+1][jointN+1]={

{0, 0, 0, 0, 0},

{0,502,434,497,286,},

{0,502,530,636,286,},

{0,502,529,636,427,},

{0,502,622,741,427,},

{0,502,622,740,370,},

{0,502,743,646,370,},

{0,502,743,646,412,},

{0,502,743,646,412,},

{0,501,606,646,464,},

{0,500,463,501,464,},

};

void setup() {

DEBUG\_SERIAL.begin(57600);

dxl.begin(1000000);

dxl.setPortProtocolVersion(DXL\_PROTOCOL\_VERSION);

for(i=1;i<jointN+1;i++)

{

dxl.setOperatingMode(i,OP\_POSITION);

for(j=1;j<=pages;j++)

{

for(i=1;i<=jointN;i++)

{

dxl.setGoalVelocity(i,50);

dxl.setGoalPosition(i,buf[j][i]);

}

delay(5000);

}

}

}

# Приложение 3. Внешний вид устройства.

