ОЛИМПИАДА ШКОЛЬНИКОВ «ШАГ В БУДУЩЕЕ»

НАУЧНО-ОБРАЗОВАТЕЛЬНОЕ СОРЕВНОВАНИЕ «ШАГ В БУДУЩЕЕ, МОСКВА»

4411

регистрационный номер

Специальное машиностроение

название факультета

СМ-7 «Мехатроника и робототехника»

Название кафедры

Разработка и конструирование робота-манипулятора

Автор: Сарматин Владислав Алексеевич

ГБОУ Школа №1251 имени генерала Шарля де Голля, г. Москва, «11А» класс

Москва – 2019

Оглавление

1. Аннотация

Целью работы является разработка и конструирование робота-манипулятора. Манипулятор строится на основе кинематики SCARA. В конструкции использованы максимально доступные комплектующие для последующей простоты повторения проекта. Для управления роботом было разработано мобильное приложение на базе OS Android. Реализовано дистанционное управление посредством протокола Bluetooth. В качестве привода использованы шаговые двигатели 17HS4401S типоразмера Nema 17. Управление осуществляет микроконтроллер ATmega328 с помощью драйверов A4988. Манипулятор оснащен захватом. Для демонстрации был реализован алгоритм отрисовки изображений.

2. Введение

Сегодня робототехника является неотъемлемой частью нашей жизни. Заменяя тяжелый ручной труд, автоматизация приобретает всё большее значение в современном мире. Автоматизации подвергаются различные сферы жизни: производство товаров, медицина, торговля, развлечения и различные сферы обслуживания – всё это в той или иной мере подверглось модернизации и цифровизации.

Неотъемлемой частью любого производства являются роботы-манипуляторы. Созданные на подобии механической руки манипуляторы выполняют различные операции в опасных для человека средах, занимаются сортировкой, сборкой, сваркой. Благодаря высокой точности, использование манипуляторов в медицине, в частности в роли хирурга, позволяет минимизировать человеческий фактор, что влечет уменьшение смертности и облегчение постоперационного периода. Использование манипуляторов в науке позволяет осуществлять ранее недоступные или опасные эксперименты, как, на пример, глубоководные эксперименты или опыты с радиоактивным веществом.

3. Цели

Разработать и сконструировать рабочую модель робота-манипулятора. В качестве демонстрации возможностей робота реализовать алгоритм отрисовки изображений маркером на бумаге.

4. Задачи

Для решения поставленной цели были поставлены следующие задачи:

1. Выбор подходящей кинематики и конструкции робота
2. Моделирование составных частей
3. Подбор электроники
4. Решение обратной задачи кинематики
5. Реализация алгоритма обработки изображения
6. Написание ПО микроконтроллера и смартфона

5. Выбор кинематики и конструкции

Для осуществления поставленной задачи рассматривалась классическая картезианская кинематика и кинематика SCARA (Selective Compliance Articulated Robot Arm). Последняя позволяет добиться высокой точности в плоскости, может выполнять более сложные задачи. К минусам относится падение точности по осям X и Y при отдалении от начала координат. Обуславливается это тем, что при отдалении захвата манипулятора от центра, увеличивается длина рычага, и на один шаг мотора приходится большее расстояние. Для решения этой проблемы было решено использовать понижающую передачу. В наличии имелись шкивы для шаговых двигателей 17HS4401S на 20 зубьев. Для расчета ведомого шкива 1 звена я исходил из длины плеча и количества шагов на полный оборот двигателя. Длина звеньев подбиралась таким образом, чтобы робот смог с запасом покрыть всю площадь листа А4. Путём простого эксперимента с линейками и листом бумаги была подобрана величина в 220мм. Количество шагов на полный оборот составляет

200 шагов, что составляет 1,8° на 1 шаг. По теореме косинусов вычисляем какое расстояние пройдёт захват при максимальном отдалении от центра.

$$L_1 = \sqrt{8L_0^2 - 8L_0^2 \cos\alpha}$$

$$\alpha = 360° / (n \cdot k \cdot m)$$

Где n –количество шагов на полный оборот двигателя, k – передаточное соотношение, m – режим микрошага (может принимать значения: 2, 4, 8, 16). При увеличении микрошагов уменьшается крутящий момент двигателя и увеличивается нагрузка на сам драйвер, поэтому ставить значение выше 8 микрошагов на шаг не рекомендуется. На *рисунке 1* схематично представлено передвижение захвата.

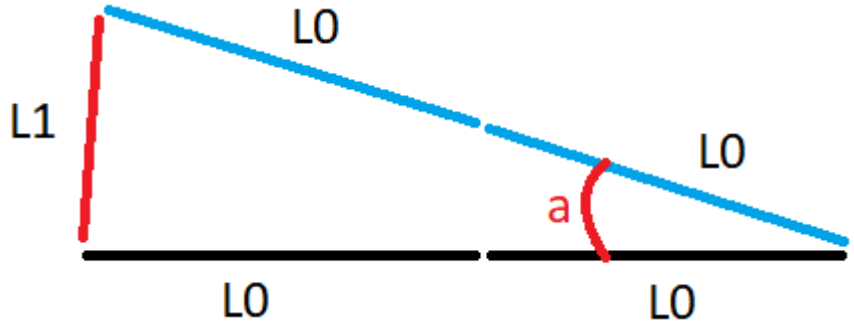


Рисунок 1

Подставляя значения в формулу находим оптимальное передаточное значение равное 11. При таком соотношении шкивов и 8 микрошагах получаем $\alpha = 0{,}02°$, тогда $L_1 = 0{,}15$мм. Такую точность перемещения считаю удовлетворительной. Передаточное отношение шкива второго звена считается из тех же соображений, но с учетом того, что длина плеча не $2L_0$, а L_0.

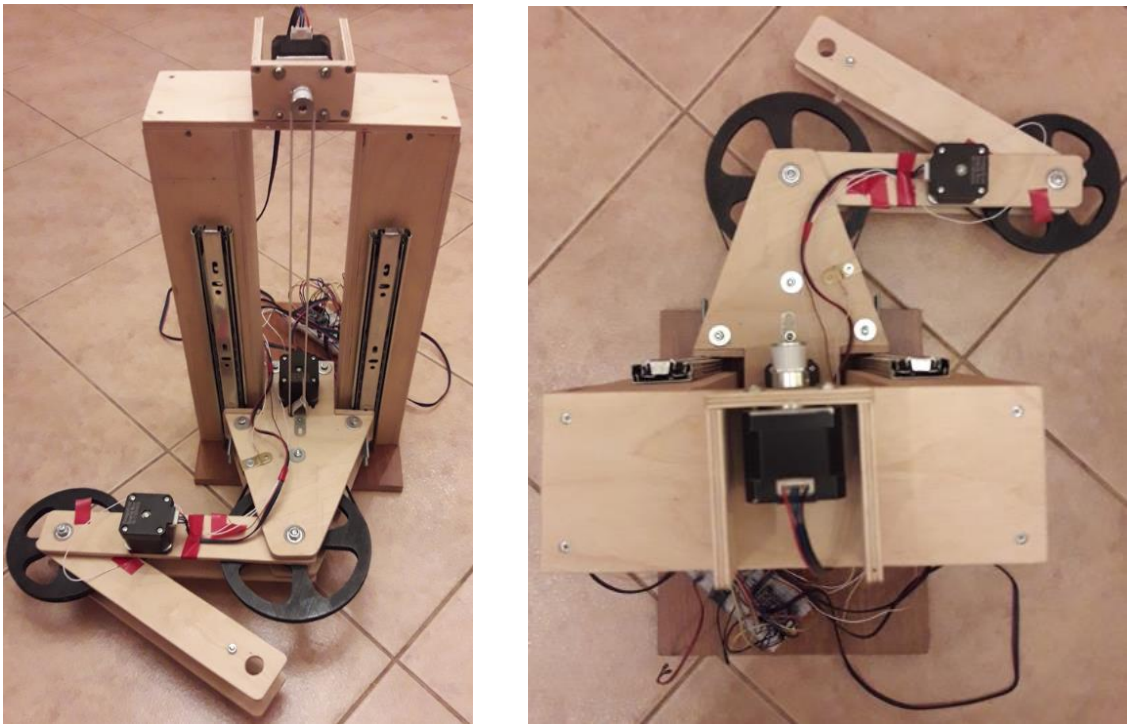
Минимальное количество степеней свободы для перемещения в 3-х осях равняется 3, для вращения захвата вокруг своей оси была добавлена 4 степень свободы.

В качестве каркаса манипулятора было решено сделать раму из алюминиевых профилей. Такая рама обладает достаточной жесткостью и

небольшим весом. Все соединения выполнены из пластиковых креплений, что обеспечивает быструю сборку и разборку, а также лёгкость и перпендикулярность.

5.1. Сравнение старой и новой конструкции манипулятора

При первой попытке реализации проекта я решил выполнить манипулятор полностью из фанеры, так как фанера является очень доступным материалом, а возможность 3D печати большого количества деталей отсутствовала. Первая версия манипулятора представлена на *рисунке 2*.

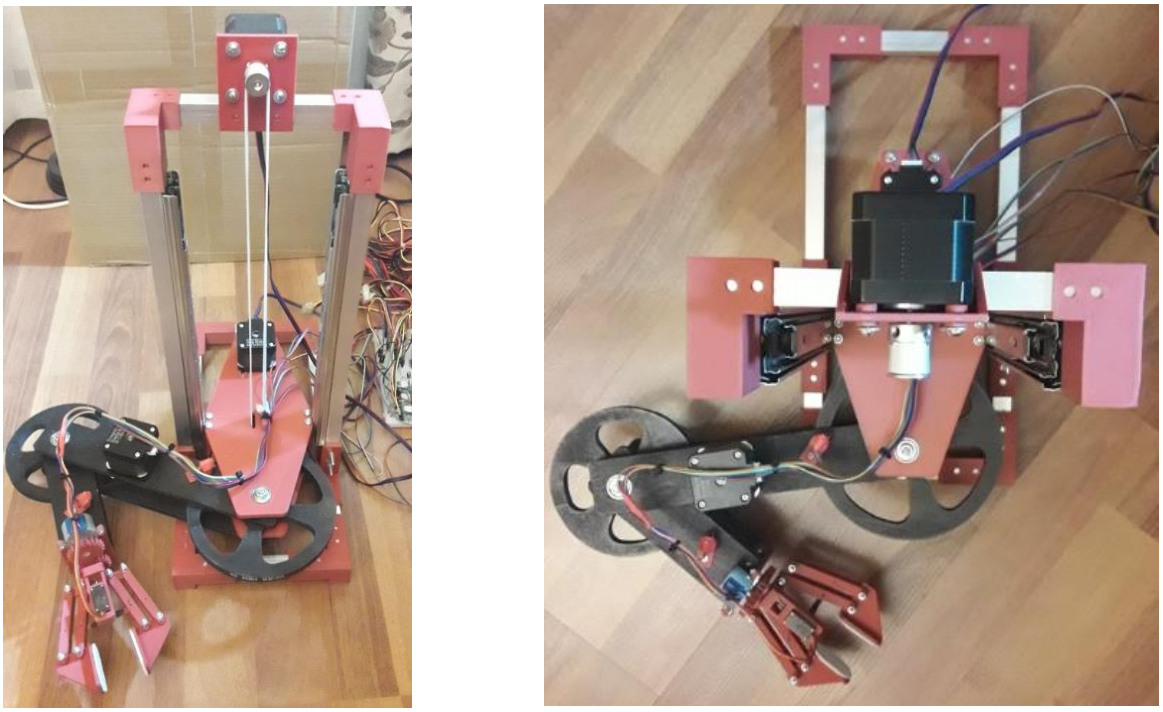


(а) *(б)*

Рисунок 2

Создание всех деталей из фанеры оказалось достаточно сложным, так как необходимо вручную выпиливать и подгонять все детали. Относительным плюсом данной конструкции является большой вес, что придаёт устойчивости при движении на больших скоростях и резких остановках. Главным же минусом оказался люфт направляющих, а также не параллельность полу, вызванная невозможностью прижать фанерную платформу к рельсам направляющих.

После отработки основных алгоритмов и подтверждения работоспособности устройства, я решил модернизировать манипулятор с учетом допущенных ошибок. Его новая версия изображена на *рисунке 3*.



(а) *(б)*

Рисунок 3

Помимо добавления захвата, была переделана платформа и каркас. Новое расположение направляющих уменьшило люфт, а выступы на платформе обеспечили жесткость прилегания к рельсам направляющих. От старой конструкции я оставил два фанерных звена, так как жесткость фанеры при изготовлении длинных и тонких деталей оказалась выше жесткости пластика.

6. Моделирование

Моделирование всех необходимых деталей производилось в программе SolidWorks. Были созданы и распечатаны модели всех креплений, захвата и платформы. Для платформы смоделировано специальное крепление к направляющим. Концевые выключатели сделаны на основе кнопок, которые вставляются в пластиковый корпус и обжимаются термоусадочной трубкой. Шкивы выполнены в программе OpenSCAD, для которой существует алгоритм

параметризированной генерации необходимого шкива [1]. Далее в формате STL модели дорабатывались в SolidWorks’е

7. Выбор электроники

Для управления шаговыми двигателями выбраны драйвера a4988. Они широко распространены и часто используются для управления подобными двигателями. Для нормального функционирования двигателя необходимо правильно настроить подаваемый ток. Для этого на драйвере имеется подстроечный резистор. Замеряя напряжение между резистором и GND, можно пользуясь формулой рассчитать нужный ток [2].

$$Vref = CurrentLimit * 8 * RS$$

Где Vref – напряжение на резисторе, CurrentLimit – ток шагового двигателя, RS – номинал токочувствительных резисторов (R050 или R100). В нашем случае:

$$Vref = 1{,}2 * 8 * 0{,}100 = 0{,}96\text{В}$$

В качестве управляющего микроконтроллера был выбран ATmega328P на базе платформы Arduino Nano. На контроллере уже предусмотрен удобный загрущик, а на плате имеются USB разъем и все необходимые выходы. Также рассматривались модели Raspberry Pi, однако их функционал был избыточен.

Для дистанционного управления был выбран Bluetooth, так как он широко распространён во всех современных смартфонах и прост в реализации. Рассматривалась возможность соединения по Wi-Fi, однако стоимость такого модуля для Arduino значительно выше. Самым популярным Bluetooth модулем для Arduino является HC-05, он может выступать и в роли приёмника, и в роли передатчика. Коммуникация микроконтроллера и передатчика осуществляются по протоколу UART. Дальность приёма сигнала составляет около 10м, чего для поставленной задачи вполне достаточно.

Все электронные компоненты я запитал от компьютерного блока питания. Логическую часть от 5В, а двигатели от 12В. Мощность блока питания по 12В

линии составляет 180Вт, чего с запасом хватает для моторов. По 5В линии блок может выдать 120Вт, чего также хватает с избытком.

Важно учесть, что при работе драйвера сильно нагреваются, поэтому их необходимо охлаждать. Для этого в конструкции предусмотрено место для вентилятора.

8. Решение обратной задачи кинематики

Чтобы реализовать передвижение захвата по заданным координатам необходимо рассчитать углы поворота шаговых двигателей. Для этого рассмотрим вид манипулятора в плоскости на *рисунке 4*.

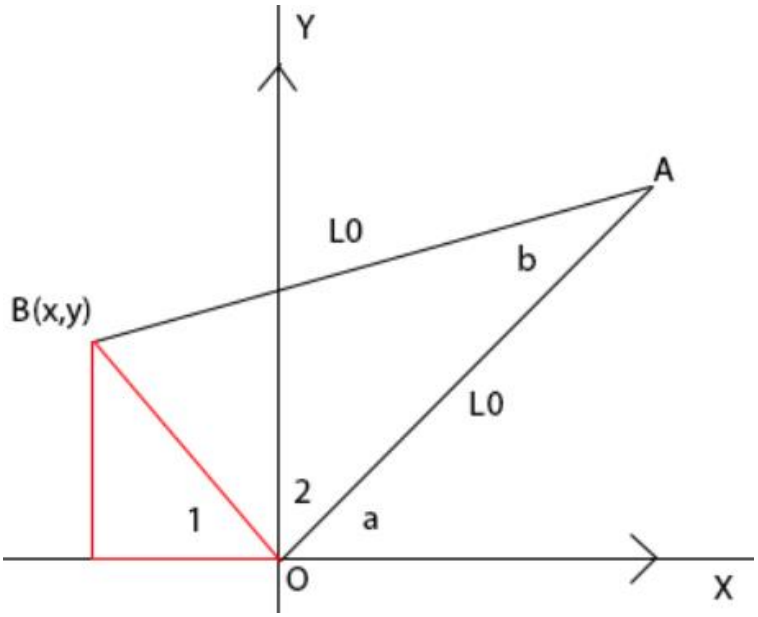


Рисунок 4 – вид сверху

B – координата захвата, A – второй узел, O – начало координат на платформе. При решении обратной задачи кинематики возникает неопределенность, так как одной конечной координате может соответствовать несколько положений манипулятора. Для облегчения этой задачи я зафиксировал положение манипулятора с одной стороны, то есть $\beta < 180°$. Такое допущение добавляет мёртвую зону в 1 четверти координатной плоскости, так как там будет мешать каркас конструкции.

Для того чтобы рассчитать углы нам необходимо знать начальное положение всех звеньев. Для этого установлены концевые выключатели. При

включении или поступлении команды со смартфона манипулятор проворачивает каждое звено до тех пор, пока не сработает концевик. Такое положение будет каждый раз одинаковым, следовательно, с помощью линейки, применяя теорему косинусов, можно вручную вычислить значения α_0 и β_0.

При включении манипулятор по α_0 и β_0 вычисляет координаты x и y:

$$x = L_0 * cos(\alpha_0) + L_0 * cos(\alpha_0 + \beta_0)$$

$$y = L_0 * sin(\alpha_0) + L_0 * sin(\alpha_0 + \beta_0)$$

$$\beta_0 - \text{угол смежный с}$$

Итак, чтобы посчитать угол β зная x и y применим теорему косинусов:

$$\beta = arccos\left(\frac{2L_0^2 - x^2 - y^2}{2L_0^2}\right)$$

Чтобы посчитать α посчитаем смежные углы 1 и 2:

$$1 = \arccos\left(\frac{|x|}{\sqrt{x^2 + y^2}}\right)$$

$$2 = \arccos\left(\frac{\sqrt{x^2 + y^2}}{2L_0}\right)$$

При условии, если $y \geq 0$:

$$\alpha = 180° - 1 - 2$$

Иначе

$$\alpha = 180° + 1 - 2$$

Для управления двигателями необходимо углы перевести в количество шагов:

$$N = round(\frac{200 * m * q}{360} * \gamma)$$

Где N – количество шагов, m – режим микрошага этого двигателя, q – передаточное отношение, γ – угол.

Алгоритм передвижения выглядит следующим образом:

1. Изменяется координата. Со смартфона задаются Δx и Δy, они прибавляются к текущим значениям.
2. Считаются углы. По приведенным выше формулам вычисляются новые значения углов
3. Углы переводятся в шаги.
4. Каждый двигатель поворачивается на отведенное количество шагов.

9. Обработка изображений

Для отрисовки изображений был выбран растровый способ. Изображение разбивается на множество точек, которые манипулятор выстукивает маркером на листе. Такой метод позволяет сымитировать оттенки серого и придаёт изображению интересную фактуру. Замечу, что можно было реализовать векторный способ, при котором изображения формируются линиями и контурами, что не позволило бы передать оттенки.

Принцип разбиения изображения на отдельные точки называется дизеринг. Существует много подобных алгоритмов, но одним из самых известных является алгоритм Флойда-Стейнберга [3].

(а) *(б)*

Рисунок 5 – пример работы алгоритма

Сначала необходимо перевести изображение в оттенки серого [4]. Для этого необходимо пройтись по всей матрице изображения, обработав отдельный пиксель. Каждый пиксель хранится в виде шестнадцатеричного числа, где на каждый цвет отводится 256 бит. Необходимо выделить каждый цвет пикселя, а затем смешать их в определенной пропорции, которая относится к особенностям восприятия цвета человеческим глазом. Далее, добавляя смешанное значение в каждую ячейку цвета пикселя, мы получаем какой-либо оттенок серого. Реализация на Java:

```
for (int y = 0; y < imgSize.getHeight(); y++){
                for (int x = 0; x < imgSize.getWidth(); x++){
                    int pxl = scaledBitmap.getPixel(x, y);
                    int red = (pxl & 0x00FF0000) >> 16;
                    int green = (pxl & 0x0000FF00) >> 8;
                    int blue = (pxl & 0x000000FF);
                    int gray = (int) (red * 0.3 + green * 0.59 + blue * 0.11);
                    int newpxl = (gray << 16) | (gray << 8) | gray;
                    grayBmp.setPixel(x, y, newpxl);
                }
            }
```

Далее необходимо сжать полученное изображение. Лист А4 имеет соотношение сторон $\sqrt{2}$. Диаметр рабочего маркера составляет 2мм. При ширине листа ≈20мм получаем формат картинки 100x141 пиксель. В таком формате картинка получается низкого качества, однако это увеличивает скорость её отрисовки. При желании можно заменить на более тонкий маркер. Стандартной Java функцией сжимаем картинку:

```
Bitmap scaledBitmap = Bitmap.createScaledBitmap(bitmap,
imgSize.getWidth(), imgSize.getHeight(), true);
```

К обработанному изображению применяем алгоритм Флойда-Стейнберга:

```
for (int y = 0; y < imgSize.getHeight(); y++){
                for (int x = 0; x < imgSize.getWidth(); x++){
                    int pxl = grayBmp.getPixel(x, y);
                    int red = (pxl & 0x00FF0000) >> 16;
                    int r = Math.round(2*red/255)*255;
                    ditheredBmp.setPixel(x,y, (r << 16) | (r << 8) | r);
                    int err = red - r;
                    int pxlErr  = (err << 16) | (err << 8) | err;
                    if(x < imgSize.getWidth()-1)
                       ditheredBmp.setPixel(x+1,y,ditheredBmp.getPixel(x+1,y)
+ pxlErr*7/16);
                    if(x > 1 && y < imgSize.getHeight()-1)
                        ditheredBmp.setPixel(x-1,y+1,ditheredBmp.getPixel(x-
1,y+1) + pxlErr*3/16);
                    if(y < imgSize.getHeight()-1)
                       ditheredBmp.setPixel(x,y+1,ditheredBmp.getPixel(x,y+1)
+ pxlErr*5/16);
                    if(x < imgSize.getWidth()-1 && y < imgSize.getHeight()-1)

ditheredBmp.setPixel(x+1,y+1,ditheredBmp.getPixel(x+1,y+1) + pxlErr/16);
                }
            }
```

Результат работы алгоритма представлен в Приложении 1.

10. Программирование

Программа для микроконтроллера выполнена на языке C++ в среде разработки Arduino IDE и представлена в приложении 2.

Android приложение написано на Java. Помимо вышеуказанного алгоритма в нем реализован простой экран управления роботом и взаимодействие с Arduino по Bluetooth.

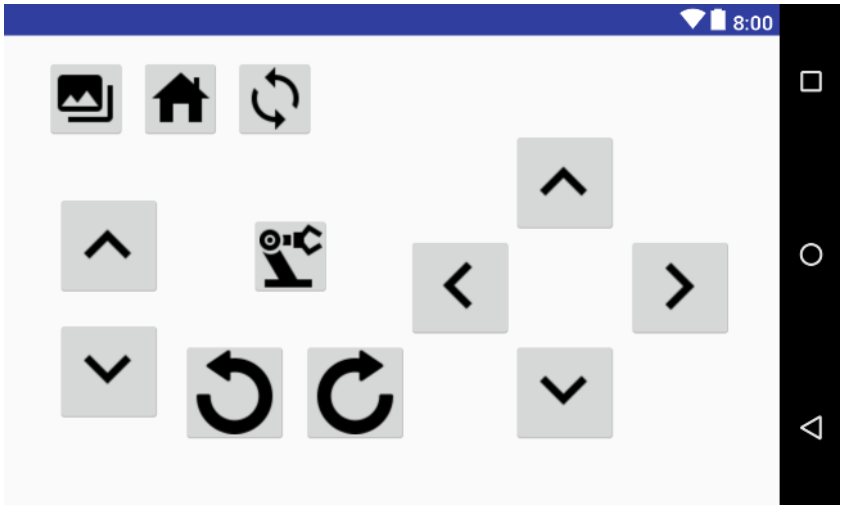


Рисунок 6 – экран управления

11. Заключение

В результате проделанной работы был сконструирован 4-х осевой робот-манипулятор и реализован алгоритм отрисовки изображений. Захват робота позволяет производить различные манипуляции с предметами и может быть приспособлен для большого спектра задач. Поставленная цель достигнута.

12. Дальнейшие доработки

В будущем будет реализована поддержка G кодов, что обеспечит совместимость с программами для управления ЧПУ. Также будет реализовано компьютерное зрение, чтобы манипулятор мог сам находить и перемещать объекты.

13. Список используемой литературы

1. www.thingiverse.com

2. portal-pk.ru

3. www.visgraf.impa.br/Courses/ip00/proj/Dithering1/floyd_steinberg_dithering.html

4. www.tannerhelland.com/3643/grayscale-image-algorithm-vb6/

Приложение 1

Приложение 2

```
#include <SoftwareSerial.h>
#include <AccelStepper.h>
#include <String.h>
#include <MultiStepper.h>
#include <Servo.h>

SoftwareSerial BTserial(8, 9); // RX | TX

#define PIN_STEP_3 11
#define PIN_DIR_3 12

#define PIN_STEP_2 5
#define PIN_DIR_2 4

#define PIN_STEP_1 3//UP
#define PIN_DIR_1 2

#define ENABLE_PIN 7

#define SERVO_PIN A5

#define m1_trigger A0
#define m3_trigger A1
#define m2_trigger A2

#define Z_SPEED 1500
#define m2_SPEED 15000
#define m3_SPEED 15000

#define m1_microstep 8
#define m2_microstep 8
#define m3_microstep 8

#define CLOSED_ANGLE 20
#define OPENED_ANGLE 130

#define L 220

#define SPEED 0.5

String buff;

AccelStepper m_3(1, PIN_STEP_3, PIN_DIR_3);
AccelStepper m_2(1, PIN_STEP_2, PIN_DIR_2);
AccelStepper m_1(1, PIN_STEP_1, PIN_DIR_1);

MultiStepper steppers;
long positions[3] = {0,0,0};

Servo arm;
bool isArmOpened;

const long baudRate = 9600;
char c=' ';
```

```
bool zRun = false;
bool fRun = false;
bool bRun = false;
bool lRun = false;
bool rRun = false;
bool runToDot = false;
bool makeDot = false;
bool goBack = false;
bool goHome[3] = {0,0,0};
byte img[100][141];

const double A_0 = 66.01*PI/180.0;
const double B_0 = 153.19*PI/180.0;

double A = A_0;
double B = B_0;

double x = L*cos(A) + L*cos(A+B);
double y = L*sin(A) + L*sin(A+B);

double xy(float x, float y){
  return sqrt(x*x+y*y);
}

double A_1(double x, double y){
  if(y >=0)
    return PI - acos( -x/xy(x,y) ) - acos(xy(x,y)/(2*L));
  else
    return PI - acos(xy(x,y)/(2*L)) + acos( -x/xy(x,y) );
}

double B_1(double x, double y){
  return PI - acos((pow(L, 2)*2 - xy(x,y)*xy(x,y)) / (pow(L, 2)*2)  );
}

long  comptSteps(float angle, int micro, int q){
  return -round((200*micro*q)/360*angle*180.0/PI);
}

void setup()
{

    pinMode(m1_trigger, INPUT);
    pinMode(m2_trigger, INPUT);
    pinMode(m3_trigger, INPUT);

    pinMode(ENABLE_PIN, OUTPUT);
    digitalWrite(ENABLE_PIN, LOW);

    Serial.begin(baudRate);
    BTserial.begin(baudRate);

    arm.attach(SERVO_PIN);
    arm.write(OPENED_ANGLE);
    isArmOpened = true;

    m_1.setMaxSpeed(Z_SPEED);
    m_1.setAcceleration(1000);
    m_1.setSpeed(Z_SPEED);

    m_2.setMaxSpeed(m2_SPEED);
    m_2.setAcceleration(13000);
    m_2.setSpeed(m2_SPEED);
```

```
    m_3.setMaxSpeed(m3_SPEED);
    m_3.setAcceleration(13000);
    m_3.setSpeed(m3_SPEED);



    steppers.addStepper(m_1);
    steppers.addStepper(m_2);
    steppers.addStepper(m_3);

    goToHome();
}

void loop()
{
    if (BTserial.available())
    {
        c = BTserial.read();
        Serial.write(c);

        switch(c){
        case '0':
          zRun = false;
          fRun = false;
          bRun = false;
          lRun = false;
          rRun = false;
        break;
        case 'A':
          if(isArmOpened){arm.write(CLOSED_ANGLE); isArmOpened = false;}
          else{arm.write(OPENED_ANGLE); isArmOpened = true;}
        break;
        case 'U':
          zRun = true;
          m_1.setSpeed(Z_SPEED);
        break;
        case 'D':
          zRun = true;
          m_1.setSpeed(-Z_SPEED);
        break;
        case 'F':
          fRun = true;
        break;
        case 'B':
          bRun = true;
        break;
        case 'L':
          lRun = true;
        break;
        case 'R':
          rRun = true;
        break;
        case 'G':
          buff="";
          while(true){
              if (BTserial.available()){
                  c = BTserial.read();
                  if(c == ';') {processCommand(); break;}
                  buff+=c;
              }
          }
        break;
        case 'H':
          goToHome();
```

```
break;
        }
    }
    if (Serial.available())
    {
        c = Serial.read();
        BTserial.write(c);
    }

    if(zRun){
          m_1.runSpeed();
          positions[0] = m_1.currentPosition();
    }

    if(fRun){
          x=x-SPEED;
          B = B_1(x,y);
          A = A_1(x,y);
          positions[1] = comptSteps(A, m2_microstep, 11);
          positions[2] = comptSteps(B, m3_microstep,9);
          steppers.moveTo(positions);
          steppers.runSpeedToPosition();
    }

    if(bRun){
          x=x+SPEED;
          B = B_1(x,y);
          A = A_1(x,y);
          positions[1] = comptSteps(A, m2_microstep, 11);
          positions[2] = comptSteps(B, m3_microstep,9);
          steppers.moveTo(positions);
          steppers.runSpeedToPosition();
    }

    if(lRun){
          y=y-SPEED;
          B = B_1(x,y);
          A = A_1(x,y);
          positions[1] = comptSteps(A, m2_microstep, 11);
          positions[2] = comptSteps(B, m3_microstep,9);
          steppers.moveTo(positions);
          steppers.runSpeedToPosition();
          //Serial.print(x);Serial.print(" ");Serial.println(y);
    }

    if(rRun){
          y=y+SPEED;
          B = B_1(x,y);
          A = A_1(x,y);
          positions[1] = comptSteps(A, m2_microstep, 11);
          positions[2] = comptSteps(B, m3_microstep,9);
          steppers.moveTo(positions);
    steppers.runSpeedToPosition();
    }

    if(goHome[0]){
      while(!digitalRead(m1_trigger)){
        m_1.runSpeed();
      }
        goHome[0] = false;
        m_1.setCurrentPosition(0);
        m_1.setSpeed(Z_SPEED);
    }
    if(goHome[1]){
```

```
      while(!digitalRead(m2_trigger)){
        m_2.runSpeed();
      }
        goHome[1] = false;
        m_2.setCurrentPosition(comptSteps(A_0, m2_microstep, 11));
        m_2.setSpeed(m2_SPEED);
    }
    if(goHome[2]){
      while(!digitalRead(m3_trigger)){
        m_3.runSpeed();
      }
        goHome[2] = false;
        m_3.setCurrentPosition(comptSteps(B_0, m3_microstep, 9));
        m_3.setSpeed(m3_SPEED);
    }

    if(runToDot){
      while(m_2.distanceToGo() || m_3.distanceToGo() || m_1.distanceToGo()){
        m_1.run();
        m_2.run();
        m_3.run();
      }
      runToDot = false;
      BTserial.write("K");
    }
}

void goToHome(){
    goHome[0] = true;
    goHome[1] = true;
    goHome[2] = true;
    zRun = false;
    m_1.setSpeed(-600);
    m_2.setSpeed(2000);
    m_3.setSpeed(-2000);
    x = L*cos(A_0) + L*cos(A_0+B_0);
    y = L*sin(A_0) + L*sin(A_0+B_0);
    A = A_0;
    B = B_0;
}

void processCommand(){
    float new_x, new_y, new_z;
    new_x = buff.substring(1, buff.indexOf('Y')).toFloat();
    new_y = buff.substring(buff.indexOf('Y')+1, buff.indexOf('Z')).toFloat();
    new_z = buff.substring(buff.indexOf('Z')+1).toFloat();
    buff = "";
    x=x-new_x;
    y=y+new_y;
    B = B_1(x,y);
    A = A_1(x,y);
    m_1.move(new_z);
    m_2.moveTo(comptSteps(A, m2_microstep, 11));
    m_3.moveTo(comptSteps(B, m3_microstep,9));
    runToDot = true;
}
```