Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Архитектура вычислительных систем (АВС)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту на тему

«Mr. Pointer»

Выполнил: студент группы 353501, Чварков М.Ю.

Руководитель: ассистент кафедры информатики Плехова Т. В.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc438058219)

[1. Анализ предметной области 5](#_Toc438058220)

[1.1. Обзор аналогов 5](#_Toc438058221)

[1.2. Постановка задачи 5](#_Toc438058222)

[1.3. Требования к разрабатываемому проекту 5](#_Toc438058223)

[2. Проектирование задачи 6](#_Toc438058224)

[2.1. Проектирование архитектуры проекта 6](#_Toc438058225)

[2.1.1. Платформа 6](#_Toc438058226)

[2.1.2. Программное обеспечение 7](#_Toc438058231)

[2.2. Используемые технологии 7](#_Toc438058232)

[2.2.1. Microsoft Visual Studio 7](#_Toc438058233)

[2.2.2. C# 8](#_Toc438058234)

[2.2.3. Windows Forms 8](#_Toc438058235)

[2.2.4. Arduino 9](#_Toc438058236)

[2.2.5. Среда разработки Arduino 9](#_Toc438058237)

[2.2.6. C++ 9](#_Toc438058238)

[2.3. Используемое оборудование 9](#_Toc438058239)

[3. Описание применения 11](#_Toc438058240)

[3.1. Программное обеспечение 11](#_Toc438058241)

[3.2. Платформа 13](#_Toc438058242)

[3.3. Пример работы 14](#_Toc438058243)

[Заключение 15](#_Toc438058244)

[Список использованных источников 16](#_Toc438058245)

[Приложения 17](#_Toc438058246)

[1. Программа на Arduino: 17](#_Toc438058247)

[1. Класс отвечающий за связь с Arduino: 18](#_Toc438058248)

[2. Класс для генерации точек на основе массива гистограммы 19](#_Toc438058249)

# ВВЕДЕНИЕ

Дотворк (Dotwork) – это особый стиль живописи, при котором рисунок состоит из большого количества точек.

Дотворк по-другому называют пуантилизмом или точечной техникой. Основные элементы, мелкие точки, наносятся со строгим соблюдением определенного интервала. Чем выше плотность расположения точек, тем темнее оттенок. По-другому говоря, точки создают форму и делают рисунок более контрастным и насыщенным. Интересен тот факт, что данная техника многими профессионалами используется в качестве самостоятельной и лежит в основе многих изображений.

Чтобы профессионально выполнять рисунки в стиле Дотворк не достаточно просто купить ручку, необходимо либо обладать математическим складом ума, либо иметь задатки дизайнера-иллюстратора. Ежедневно мастера, работающие в данном стиле, строят элементы в векторной графике, просчитывают идеальную симметрию и продумывают композицию до мельчайших деталей. В итоге появляются масштабные рисунки. Мало кто создает маленькие картины из точек, потому что это не только не интересно, но и не имеет никакого смысла, потому что точки смотрятся идеально лишь тогда, когда для них достаточно простора. Ведь при нанесении изображения учитывается расстояние между точками, которое увеличивается либо уменьшается в зависимости от деталей рисунка.

# 1. Анализ предметной области

## Микроконтроллер

Микроконтро́ллер (англ. Micro Controller Unit, MCU) — микросхема, предназначенная для управления электронными устройствами.

Типичный микроконтроллер сочетает на одном кристалле функции процессора и периферийных устройств, содержит ОЗУ и (или) ПЗУ. По сути, это однокристальный компьютер, способный выполнять относительно простые задачи.

Отличается от микропроцессора интегрированными в микросхему устройствами ввода-вывода, таймерами и другими периферийными устройствами.

При проектировании микроконтроллеров приходится соблюдать компромисс между размерами и стоимостью с одной стороны и гибкостью и производительностью с другой. Для разных приложений оптимальное соотношение этих и других параметров может различаться очень сильно. Поэтому существует огромное количество типов микроконтроллеров, отличающихся архитектурой процессорного модуля, размером и типом встроенной памяти, набором периферийных устройств, типом корпуса и т. д. В отличие от обычных компьютерных микропроцессоров, в микроконтроллерах часто используется гарвардская архитектура памяти, то есть раздельное хранение данных и команд в ОЗУ и ПЗУ соответственно.

## Обзор аналогов

В настоящее время прямых аналогов нет. Есть принтеры, которые печатают изображения. Так же можно добиться похожих эффектов в различных фоторедакторах, но при этом нет специализированного ПО и оборудования для создания таких картин.

## Постановка задачи

В связи с тем, что рисунки в данном стиле сложно выполнять вручную, было решено автоматизировать данный процесс. Цель курсовой – создание платформы, способной создавать изображения в стиле Дотворк, используя исходные изображения или фотографии.

## Требования к разрабатываемому проекту

Готовый программный продукт должен удовлетворять следующим требованиям:

* Выходное изображение должно быть в стиле Дотворк.
* Иметь простой и понятный интерфейс.
* Иметь возможность загрузки изображений основных форматов для обработки.
* Иметь возможность предварительной настройки изображения.
* Позволять легко расширять программный продукт новыми функциями.
* Иметь возможность вывода изображения на бумагу.
* Иметь возможность замены устройства вывода на печать.

Таким образом, задача данного курсового проекта сводится к разработке платформы и сопутствующего ПО способных из входного изображения получить рисунок на бумаге в стиле Дотворк.

# Используемые технологии

Приложение реализовано на языке C# с использованием WinForms для ОС Windows и на С++ для Arduino. В качестве среды разработки были выбраны Microsoft Visual Studio 2013 и стандартная среда разработки Arduino.

### Microsoft Visual Studio

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и как отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

### C#

C# — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как язык разработки приложений для платформы Microsoft .NET Framework и впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Переняв многое от своих предшественников — языков C++, Pascal, Модула, Smalltalk и, в особенности, Java — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает множественное наследование классов (между тем допускается множественное наследование интерфейсов).

### Windows Forms

Windows Forms — интерфейс программирования приложений (API), отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework. Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обёртки для существующего Win32 API в управляемом коде. Причём управляемый код — классы, реализующие API для Windows Forms, не зависят от языка разработки. То есть программист одинаково может использовать Windows Forms как при написании ПО на C#, С++, так и на VB.Net, J# и др.

С одной стороны, Windows Forms рассматривается как замена более старой и сложной библиотеке MFC, изначально написанной на языке C++. С другой стороны, WF не предлагает парадигму, сравнимую с MVC. Для исправления этой ситуации и реализации данной функциональности в WF существуют сторонние библиотеки. Одной из наиболее используемых подобных библиотек является User Interface Process Application Block , выпущенная специальной группой Microsoft, занимающейся примерами и рекомендациями, для бесплатного скачивания. Эта библиотека также содержит исходный код и обучающие примеры для ускорения обучения.

.

### Arduino

Arduino — торговая марка аппаратно-программных средств для построения простых систем автоматики и робототехники, ориентированная на непрофессиональных пользователей. Программная часть состоит из бесплатной программной оболочки (IDE) для написания программ, их компиляции и программирования аппаратуры. Аппаратная часть представляет собой набор смонтированных печатных плат, продающихся как официальным производителем, так и сторонними производителями. Полностью открытая архитектура системы позволяет свободно копировать или дополнять линейку продукции Ардуино.

Arduino может использоваться как для создания автономных объектов автоматики, так и подключаться к программному обеспечению на компьютере через стандартные проводные и беспроводные интерфейсы.

### Среда разработки Arduino

Среда разработки Arduino состоит из встроенного текстового редактора программного кода, области сообщений, окна вывода текста(консоли), панели инструментов с кнопками часто используемых команд и нескольких меню. Для загрузки программ и связи среда разработки подключается к аппаратной части Arduino.

### C++

C++ — компилируемый, статически типизированный язык программирования общего назначения.

Поддерживает такие парадигмы программирования как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование, обеспечивает модульность, раздельную компиляцию, обработку исключений, абстракцию данных, объявление типов (классов) объектов, виртуальные функции. Стандартная библиотека включает, в том числе, общеупотребительные контейнеры и алгоритмы. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником — языком C, — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

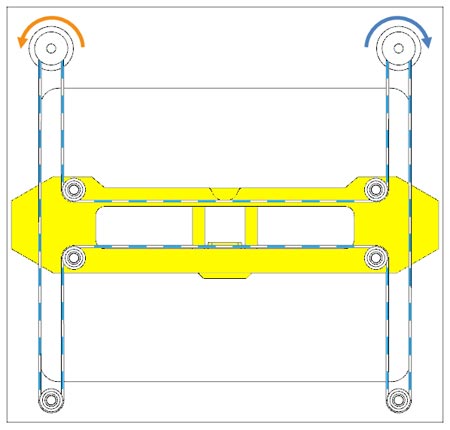
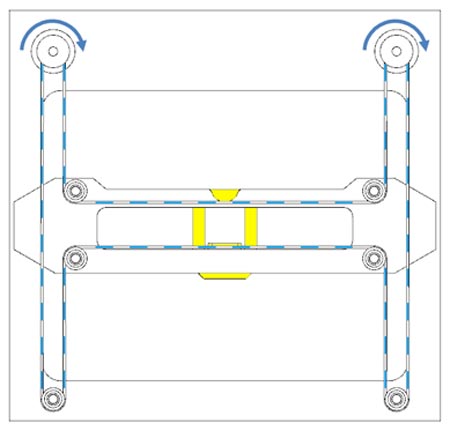
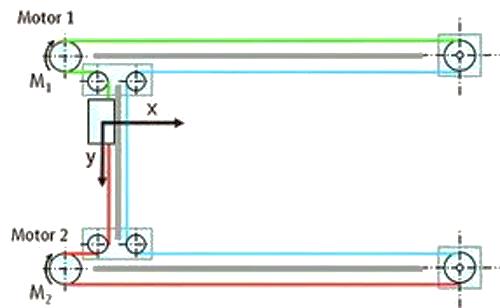
# Проектирование задачи

## Проектирование архитектуры проекта

Весь проект делится на две основные части: платформа и ПО. Рассмотрим каждую часть отдельно.

### Платформа

Платформа реализована в виде известной схемы H-bot. С помощью 4 направляющих, 2 шаговых двигателей и одного ремня печатающая головка может спокойно перемешаться по осям X и Y. Схема H-bot и принцип движения представлен на рисунках ниже.



Как видно из рисунков, в зависимости от того, двигаются шаговые двигатели в одном направлении или нет, зависит, по каким осям будет двигаться печатающая каретка. На данный момент печатающая головка представлена в виде фломастера закрепленного на сервоприводе. В будущем планируется добавить вариант головки в виде лазера, который сможет выжигать узоры на дереве или других поверхностях.

Шаговые двигатели управляются с помощью микроконтроллера Arduino, который, в свою очередь, получает команды от ПО через USB. На вход он получает массив байтов. Первый байт отвечает за тип команды, следующие 8 байт это координаты X и Y (по 4 на каждую). После выполнения команды на выход отправляется 1, для оповещения о завершения выполнения

### Программное обеспечение

ПО представлено в виде программы с графическим интерфейсом. Програамма имеет 4 основных класса:

* MainWindows – он отвечает за отрисовку окна с пользовательским интерфейсом и обработку всех вводов и нажатий.
* Servo – класс, отвечающий за связь с Arduino. В нем реализовано подключение к SerialPort и переход от координат X, Y к массиву байт, с последующей отправкой.
* Picture – класс, отвечающий за загрузку и предварительную обработку изображения
* Pointer – класс, отвечающий за создание карты точек на основе исходных данных и изображения.

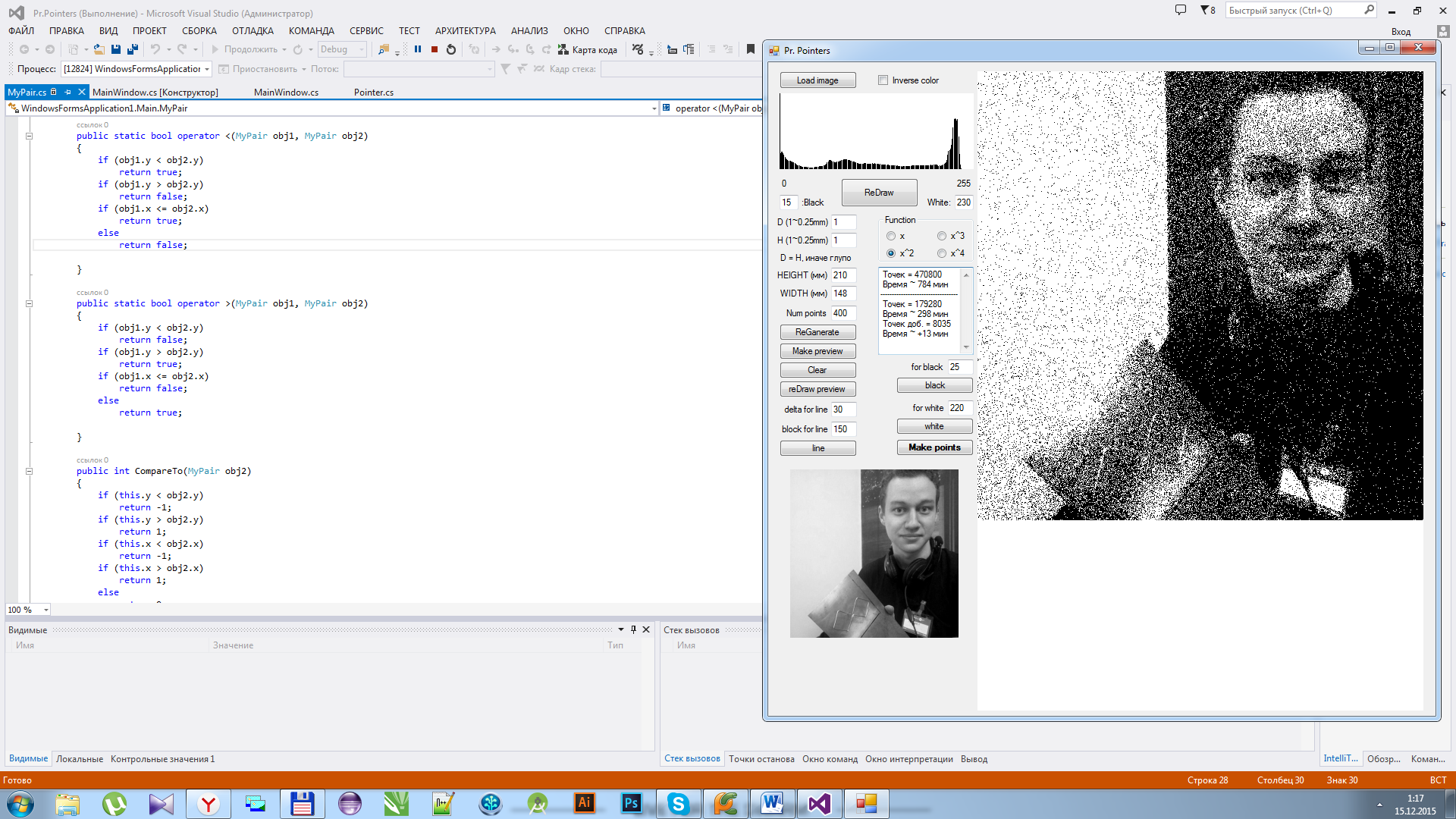
# Используемое оборудование

Для создания платформы были использованы следующие элементы:

* Микроконтроллер Arduino
* USB провод для связи Arduino и ПК
* Шаговый двигатель с точность 400 шагов на оборот
* Сервопривод на 9г для управления пишущим элементом
* Линейные подшипники для движения платформ вдоль осей
* Обычные подшипники для движения ремня
* Ремень GT2 с специальными головками для шаговых двигателей
* MOTOR SHIELD L293D (шилд для управления двигателями)
* Вентилятор от видеокарты для охлаждения shield
* Блок питания от модема, для отдельного питания двигателей
* Различные строительные элементы, крепежи, провода

# Описание применения

## Программное обеспечение



**2**

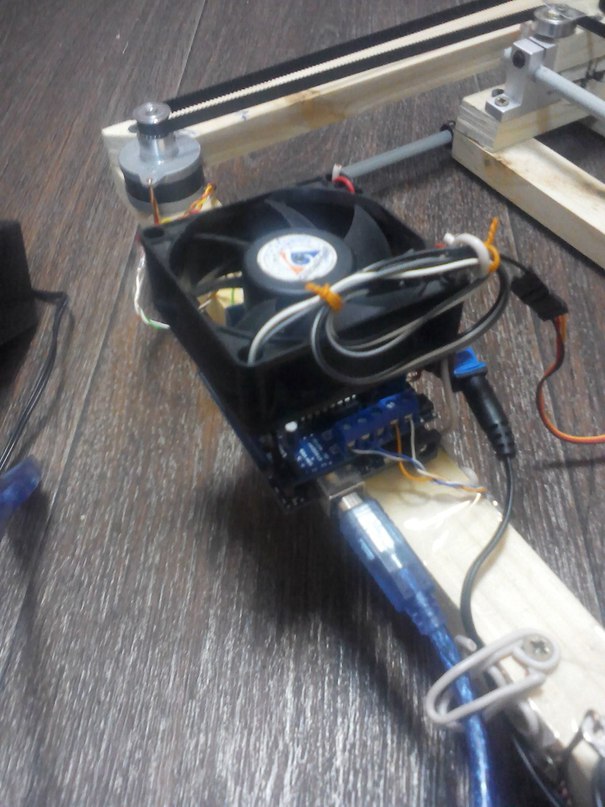
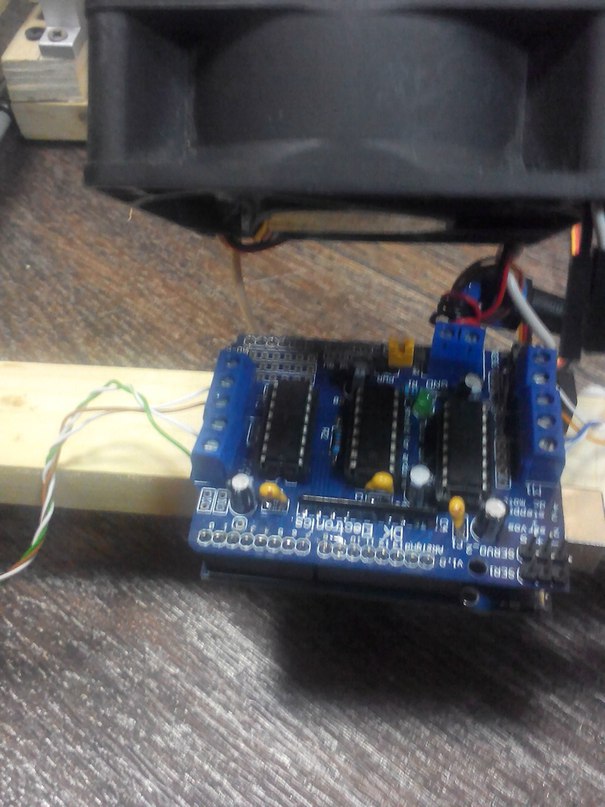
**3**

**1**

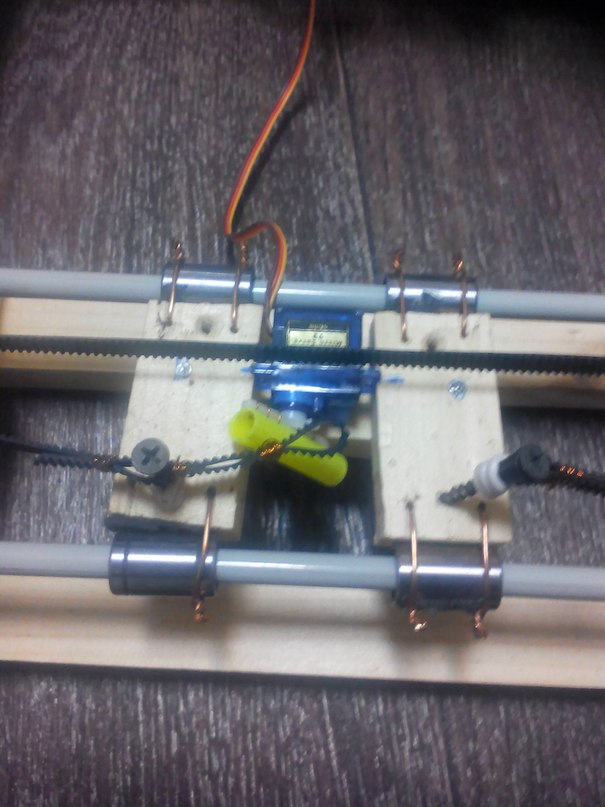
Так выглядит окно программы. Далее следует описание назначения всех полей и кнопок.

1. **Checkbox**
   1. Inverse color – наличие галочки инвертирует все цвета в исходном изображении
2. **RadioGroup**
   1. Function – выбор того, по какому закону будут распределяться точки, в зависимости от близости цвета к черному.
3. **Image**
   1. №1 – гистограмма загруженного изображения
   2. №2 – результирующее изображение из точек
   3. №3 – изображение после начальных обработок, до перевода в точки.
4. **Button**
   1. Load image – вызов диалогового окна, для выбора изображения
   2. ReDraw – перерисовка изображения в области №3 на основе параметров Checkbox и гистограммы
   3. ReGenerate – пересоздание класса для создания точек на основе введенных параметров
   4. MakePreview – рисование в области №2 на основе данных
   5. Clear – очистка области №2
   6. reDraw preview – перерисовка области №2
   7. black – закраска все областей с значением цвета ниже порогового черным
   8. white - закраска все областей с значением цвета выше порогового белым
   9. line – выделение линий на изображении в области №2
   10. Make points – запуск печати
5. **EditText**
   1. Black – левое граничное значение для гистограммы
   2. White – правое граничное значение для гистограммы
   3. D – диаметр точки
   4. H – размер шага
   5. Height – высота области для рисования
   6. Width – ширина области для рисования
   7. Delta for line – величина разницы цвета двух пикселей, для рисования линии
   8. for black – граница закраски черным
   9. for white – граница закраски белым

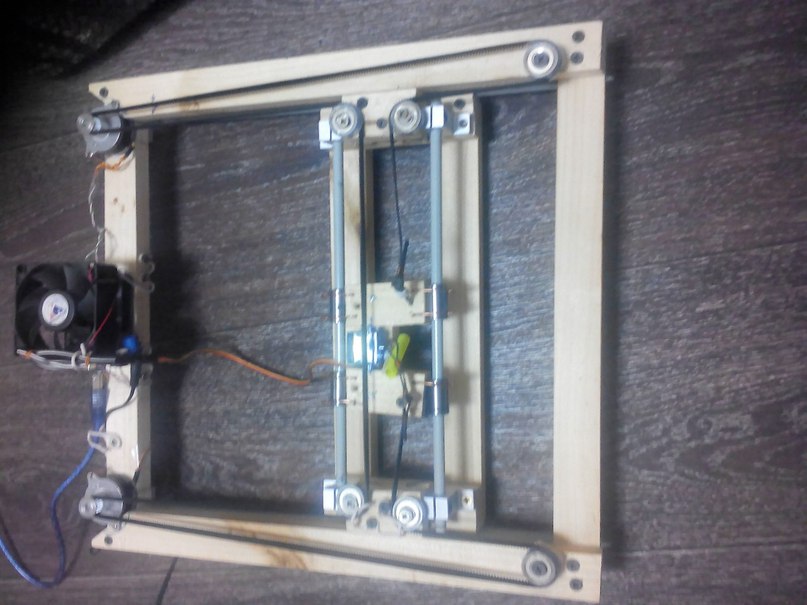
## Платформа

Управляющая часть. Arduino с Shield и вентилятор для охлаждения

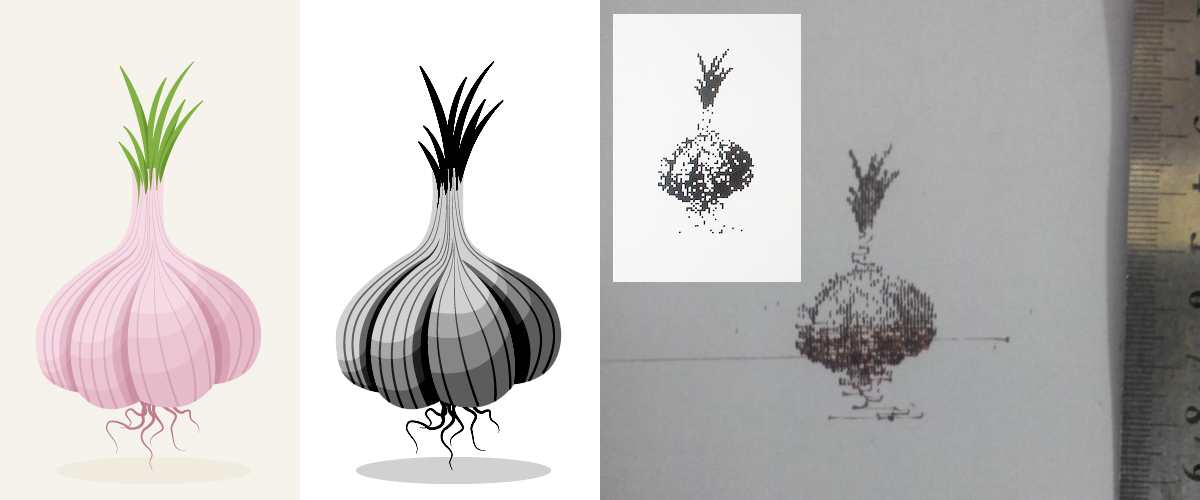


Печатающая каретка с креплением для фломастера



Общий вид платформы

## Пример работы



**4**

**3**

**2**

**1**

1 – исходное изображение

2 – изображение на основе гистограммы

3 – генерация точек на основе данных

4 – печать на бумаге

# Заключение

В результате работы над курсовым проектом была разработана платформа Mr. Pointer, которая реализует все поставленные требования.

В процессе написания проекта была изучены базовые принципы программирования микроконтроллеров, на примере Arduino.

В дальнейшей работе планируется развивать проект в следующих направлениях:

1. Замена печатающей головки на лазер
2. Стабилизация конструкции
3. Улучшение системы охлаждения
4. Создание подложки под платформу, для закрепления листа бумаги

# Список использованных источников

[1] Волосевич, А. Технология Windows Presentation Foundation / А. Волосевич. – Минск. : БГУИР, 2014. – 108 с.

[2] Arduino [Электронный ресурс]. – документация по Arduino. – Режим доступа : <http://arduino.ru/>

[3] Habrahabr [Электронный ресурс]. – полезные статьи по созданию принтера. – Режим доступа : <http://habrahabr.ru/>.

[4] Википедия [Интернет ресурс]. – Электронные данные. – Режим доступа: <http://ru.wikipedia.org/wiki/>.

[5] Шилдт Г: «С#. Полное руководство, 8-е издание». Вильямс, 2012. – 108 с.

# Приложения

### Программа на Arduino:

#include <AFMotor.h

#include <Servo.h>

AF\_Stepper motor(2048, 1);

AF\_Stepper motor2(2048, 2);

Servo servo1;

int x0 = 0;

int y0 = 0;

void setup() {

motor.setSpeed(10);

motor2.setSpeed(10);

servo1.attach(10);

servo1.write(65);

Serial.begin(9600);

}

void OneTouch(){

servo1.write(60);

delay(50);

servo1.write(65);

}

void MoveRight(int h){

motor.step(h, FORWARD, SINGLE);

motor2.step(h, FORWARD, SINGLE);

}

void MoveDown(int h){

motor.step(h, BACKWARD, SINGLE);

motor2.step(h, FORWARD, SINGLE);

}

void MoveUp(int h){

motor.step(h, FORWARD, SINGLE);

motor2.step(h, BACKWARD, SINGLE);

}

void MoveLeft(int h){

motor.step(h, BACKWARD, SINGLE);

motor2.step(h, BACKWARD, SINGLE);

}

void OnePointer(int x, int y){

int stepX = x - x0;

int stepY = y - y0;

if (stepX > 0)

MoveRight(stepX);

else

MoveLeft(-stepX);

if (stepY > 0)

MoveDown(stepY);

else

MoveUp(-stepY);

x0 = x;

y0 = y;

OneTouch();

}

void Move(int x, int y){

int stepX = x - x0;

int stepY = y - y0;

while ((stepX != 0) || (stepY != 0)){

if (stepX > 0){

MoveRight(1);

stepX--;

}

if (stepX < 0){

MoveLeft(1);

stepX++;

}

if (stepY > 0){

MoveDown(1);

stepY--;

}

if (stepY < 0){

MoveUp(1);

stepY++;

}

}

x0 = x;

y0 = y;

OneTouch();

}

void loop() {

while (Serial.available() == 0);

String buffer ="";

int i = 0;

delay(100);

while(i < 9) {

buffer += Serial.read();

i++;

}

String code = buffer.substring(0,1);

code.trim();

int cod = code.toInt();

String first = buffer.substring(0,5);

first.trim();

int x = first.toInt() - cod\*10;

String second = Buffer.substring(0,9);

second.trim();

int y=second.toInt()-x\*10000

-cod\*100000;

switch (cod){

case 0:

Move(x,y);

OneTouch();

break;

case 1:

servo1.write(65);

delay(50);

Move(x,y);

break;

case 2:

servo1.write(60);

delay(50);

Move(x,y);

servo1.write(65);

break;

case 3:

servo1.write(65);

delay(50);

break;

case 4:

servo1.write(60);

delay(50);

break;

case 5:

Move(x,y);

break;

case 6:

Move(0,0);

motor.release();

motor2.release();

break;

}

Serial.print(1);

}

### Класс отвечающий за связь с Arduino:

class Servo

{

SerialPort port;

bool isNull = true;

public Servo()

{

int i = 30;

while ((i < 50) && isNull)

{

port = new SerialPort("COM"+i, 9600);

try

{

port.Open();

isNull = false;

}catch (IOException e){}

i++;

}

}

public void SendPosition(int x, int y)

{

string result = "";

string a = x + "";

string b = y + "";

result += "0";

for (int i = 0; i < 4 - a.Length; i++)

result += "0";

result += a;

for (int i = 0; i < 4 - b.Length; i++)

result += "0";

result += b;

byte[] Bfer = new byte[9];

for (int i = 0; i < 9; i++ )

Bfer[i] = Byte.Parse(result[i]+"");

port.Write(Bfer, 0, 9);

port.ReadByte();

}

public void SendStop()

{

int x = 0;

int y = 0;

string result = "";

string a = x + "";

string b = y + "";

result += "6";

for (int i = 0; i < 4 - a.Length; i++)

result += "0";

result += a;

for (int i = 0; i < 4 - b.Length; i++)

result += "0";

result += b;

byte[] Bfer = new byte[9];

for (int i = 0; i < 9; i++)

Bfer[i] = Byte.Parse(result[i] + "");

port.Write(Bfer, 0, 9);

port.ReadByte();

}

### Класс для генерации точек на основе массива гистограммы

class Pointer

{

static public int HEIGHT = 210 \* 4;

static public int WIDTH = 148 \* 4;

static public int X0 = 276;

static public int Y0 = 12;

int f;

int N;

int[] pixels;

int height;

int width;

int[] progress;

int[][] forDraw;

LinkedList<MyPair> list = new LinkedList<MyPair>();

public Pointer(int f, int N, int[] pixels, int height, int width)

{

this.f = f;

this.N = N;

this.pixels = pixels;

this.height = height;

this.width = width;

}

public void PrepairForDraw()

{

MakeProgress();

ModificateProgress();

}

public void Draw(Graphics gr, TextBox TextBoxForAll, int D, int H)

{

SolidBrush brush = new SolidBrush(Color.White);

gr.FillRectangle(brush, new Rectangle(X0, Y0, WIDTH + 3, HEIGHT + 3));

brush.Color = Color.Black;

int[] i = GeneratePoint(gr, brush, D, H);

TextBoxForAll.Text = " Точек = " + i[0];

TextBoxForAll.Text += Environment.NewLine + " Время ~ " + (int)(i[0] \* 100.0 /

1000 / 60) + " мин";

TextBoxForAll.Text += Environment.NewLine + "-------------------------------";

TextBoxForAll.Text += Environment.NewLine + " Точек = " + i[1];

TextBoxForAll.Text += Environment.NewLine + " Время ~ " + (int)(i[1] \* 100.0 /

1000 / 60) + " мин";

brush.Dispose();

}

internal int[] GeneratePoint(Graphics gr, SolidBrush brush, int D, int H)

{

Random random = new Random();

forDraw = new int[height][];

for (int k = 0; k < height; k++)

forDraw[k] = new int[width];

int i = 0;

int j = 0;

int x, y, Num;

while (isRun())

{

y = random.Next(height);

x = random.Next(width);

Num = pixels[x + y \* width];

if (progress[Num] > 0)

{

i++;

if (forDraw[y][x] == 0)

{

forDraw[y][x] = 1;

j++;

}

progress[Num]--;

DrawPixel(x\*H, y\*H, gr, brush, D, H);

list.AddLast(new MyPair(x, y));

}

}

int[] t = {i,j};

return t;

}

private void DrawPixel(int x, int y, Graphics gr, SolidBrush brush, int D, int H)

{

gr.FillRectangle(brush, new Rectangle(x + X0 + H/2 + D/2, y + Y0 + H/2 - D/2,

D, D));

}

private void ModificateProgress()

{

for (int i = 0; i < 256; i++)

{

progress[i] = (int)(progress[i] \* getK(i));

}

}

private double getK(int i)

{

switch (f)

{

case 0:

return 1.0 \* Math.Pow((255 - i) / 256.0, 1) \* N / 100;

case 1:

return 1.0 \* Math.Pow((255 - i) / 256.0, 2) \* N / 100;

case 2:

return 1.0 \* Math.Pow((255 - i) / 256.0, 3) \* N / 100;

case 3:

return 1.0 \* Math.Pow((255 - i) / 256.0, 4) \* N / 100;

}

return 0;

}

private void MakeProgress()

{

progress = new int[256];

for (int i = 0; i < height \* width; i++)

{

progress[pixels[i]]++;

}

}

private Boolean isRun()

{

for (int i = 0; i < 256; i++)

{

if (progress[i] > 0) return true;

}

return false;

}

public void GenerateLine(int delta, int block, Graphics gr, SolidBrush brush,

int D, int H, TextBox TextBoxForAll)

{

int k = 0;

for (int i = 1; i < height - 1; i++)

{

for (int j = 1; j < width - 1; j++)

{

if (HowManyLowPixels(pixels[j + i \* width], pixels[j + i \* width + 1],

pixels[j + i \* width - 1], pixels[j + (i + 1) \* width], pixels[j + (i - 1) \* width],delta) > 0)

if (HowManyLowPixels(pixels[j + i \* width], pixels[j + i \* width

+ 1], pixels[j + i \* width - 1], pixels[j + (i + 1) \* width],

pixels[j + (i - 1) \* width],delta) < 3)

if (pixels[j + i \* width] < block)

{

if (forDraw[i][j] == 0)

{

k++;

forDraw[i][j] = 1;

DrawPixel(j \* H, i \* H, gr, brush, D, H);

}

}

}

}

TextBoxForAll.Text += Environment.NewLine + " Точек доб. = " + k;

TextBoxForAll.Text += Environment.NewLine + " Время ~ +" + (int)(k \* 100.0 /

1000 / 60) + " мин";

}

private int HowManyLowPixels(int p1, int p2, int p3, int p4, int p5, int delta)

{

int i = 0;

if (p1 >= p2 + delta) i++;

if (p1 >= p3 + delta) i++;

if (p1 >= p4 + delta) i++;

if (p1 >= p5 + delta) i++;

return i;

}

internal void GenerateBlack(int black, Graphics gr, SolidBrush brush, int D,

int H, TextBox TextBoxForAll)

{

int j = 0;

for (int i = 0; i < width \* height; i++)

{

if (pixels[i] <= black)

if (forDraw[i % height][i / height] == 0)

{

DrawPixel(i % width \* H, i / width \* H, gr, brush, D, H);

forDraw[i % height][i / height] = 1;

j++;

}

}

TextBoxForAll.Text += Environment.NewLine + " Точек доб. = " + j;

TextBoxForAll.Text += Environment.NewLine + " Время ~ +" + (int)(j \* 100.0 /

1000 / 60) + " мин";

}

internal void GenerateWhite(int white, Graphics gr, SolidBrush brush, int D,

int H, TextBox TextBoxForAll)

{

int j=0;

for (int i = 0; i < width \* height; i++)

{

if (pixels[i] >= white)

if (forDraw[i % height][i / height] == 1)

{

DrawPixel(i % width \* H, i / width \* H, gr, brush, D, H);

forDraw[i % height][i / height] = 0;

j++;

}

}

TextBoxForAll.Text += Environment.NewLine + " Точек убр. = " + j;

TextBoxForAll.Text += Environment.NewLine + " Время ~ -" + (int)(j \* 100.0 /

1000 / 60) + " мин";

}

internal void Clear(Graphics gr, SolidBrush brush)

{

gr.FillRectangle(brush, new Rectangle(X0, Y0, WIDTH + 3, HEIGHT + 3));

}

internal MyPair[] getSortArray()

{

MyPair[] array = list.ToArray<MyPair>();

Array.Sort<MyPair>(array);

return array;

}

}