# Описание на проблема/задачата

Задача е да се направи реализация на CNC Writing machine, която ще наричаме плотер. Плотерът е изходно периферно устройство, което дава възможност за извеждане на графична информация върху хартия.[1]

Като вход програмата трябва да получава изображение. Това изображение, трябва да бъде обработено до формат, в който информацията ще може да се изчертае от плотера.

Плотера ще трябва да разчита подаваната информация и спрямо нея да подава команди към моторите за движение по x и y осите и към мотора за вдигане и пускане на химикалката.

Плотера ще може да черта само в един цвят.

# Решение и теоретична обосновка

Основно задачите на плотера могат да се разделят да следните основи задачи:

* Структура на решението
* Физическа реализация
* Прочитане и обработка на изображението
* Изчертаване на изображението

### Структура на решението

//TODO: Да се опишат модулите на програмата и да се сложи диаграма

### Физическа реализация

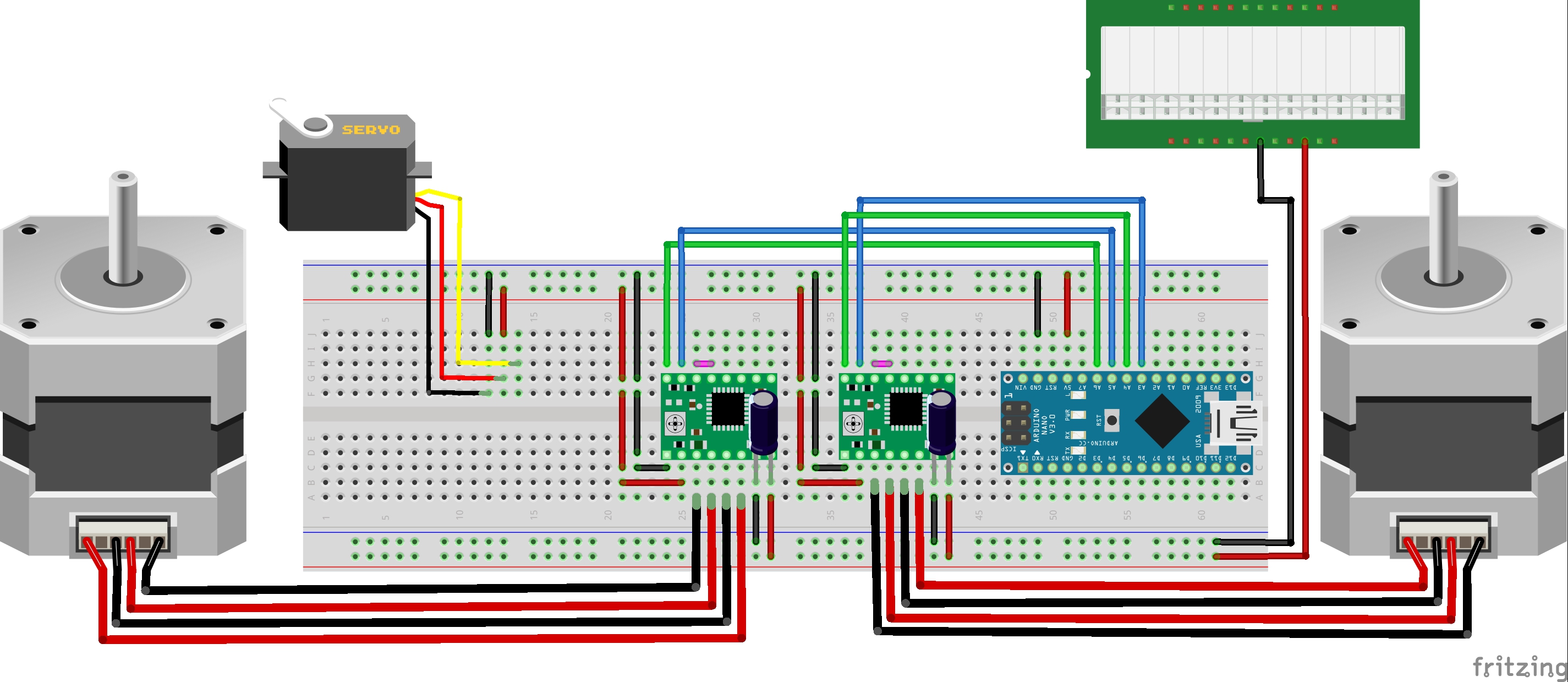
Проектът се състой от две основни части – Обработка на изображението и изчертаване на изображението.

Обработката на изображението се осъществява с помощта на компютър с операционна система. Като резултатното изображение се предава по серийна комуникация към Arduino през USB.

Arduino чрез серийна комуникация получава изображението на база, на което подава команди към двата стъпкови мотора(X и Y) и серво мотора(химикалката – Z).

Реализацията на местенето по осите е само чрез местене на пишещата глава по X и Y направление чрез винтово задвижване.

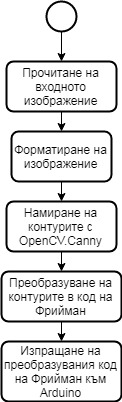
Управлението на химикалката е чрез серво мотор, който я вдига и пуска.



Фигура 1Схема на свързване

### Прочитане и обработка на изображението

За прочитане и обработката на изображението ще се използва езика Processing[2]. Този език е избран, защото има изградени много функционалности свързани с комуникацията между Processing и Arduino. Допълнително ще се използва библиотеката OpenCV for Processing[3], в която са имплементирани основните алгоритми за обработката на изображението, които ще бъдат използвани. Биоблиотеката не е развита напълно и много от възможностите на OpenCV липсват или са ограничени.



фигура 2Описание на стъпките, които са реализирани в Processing

Форматирането на изображението включва промяна на размерите и ориентацията на изображението, ако размерите са по-големи от размерите на плотера.

За намиране на контурите се използва алгоритъма Canny edge detector, защото е широко приложим за различни изображения. Това е многостъпков алгоритъм, който удовлетворява три основни критерии[4]:

* Ниско ниво на грешка – добро намиране само на съществуващите контури
* Добро локализиране – разстоянието от истинския контур до намерения да бъде минимално
* Minimal response – Минимализиране максимумите около контури и връщане на повече от необходимите пиксели част от контура.

Стъпки на Canny edge detector[4][5]:

* Филтриране на шума – За тази цел се използва Гаусов филтър.
* Намиране на интензитния градиент на изображението – За тази цел се използва алгоритъма Sobel, който определя контурите по ширината и дължината.
* Потискане на не-максимумите – По този начин се премахват пиксели, които не трябва да бъдат част от контура.
* Определяне на финалните контурите – Използват се два параметъра upper и lower threshold.

Canny edge detector не е прост оператор, което го прави по-тежък за изпълнение в сравнение със Sobel, който е част от самия алгоритъм и други алгоритми за намиране на контури. Предимството е премахването на шумовете, многостъпковото определяне на контурите и премахването на излишните такива.

В библиотеката OpenCV for Processing има възможност единствено за определянето на параметрите upper и lower threshold. Според препоръките на Canny тяхното отношение трябва да бъде около 2:1 и 3:1[5].

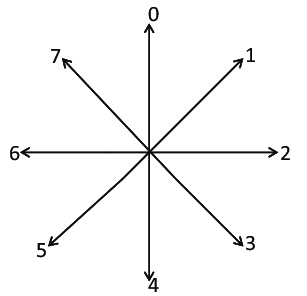
В имплементацията се използва статистически подход на база стойностите на пикселите в изображението[6][7]. Направено е сравнение на определяне по средна стойност и медиана. Също се използва подход с предварително изравняване на хистограмата. Определената стойност от медиана се взимат съответно с коефициенти 0.66 за долната граница и 1.33 или 1.98 (според препоръките за отношение 1:2 и 1:3) за горна граница.

След опити с много различни изображения се установи, че използваната имплементация на Canny не се справя добре с области, в които има дребни и рязко сменящи си посоката граници(например ситно назъбени стръкове трева). Също примерите изложени в [6] не постигат същата добър резултат. Това най-вероятно се дължи на различните имплементации на Canny и подавани параметри по подразбиране на отделните стъпки.

//TODO: Да се сложат примери за лошо получени крайни резултати

//TODO: Добавяне на анализ на другите изображения

Следващата стъпка е контурите на изображението да се изпратят на Arduino за изчертаване. Използва се Chain Freeman Code с 8 посоки. Този алгоритъм връща като резултат последователност от стойности, който оказват къде се намира следващата точка от контура спрямо текущата. Стойностите са от 0 до 7 съответно:

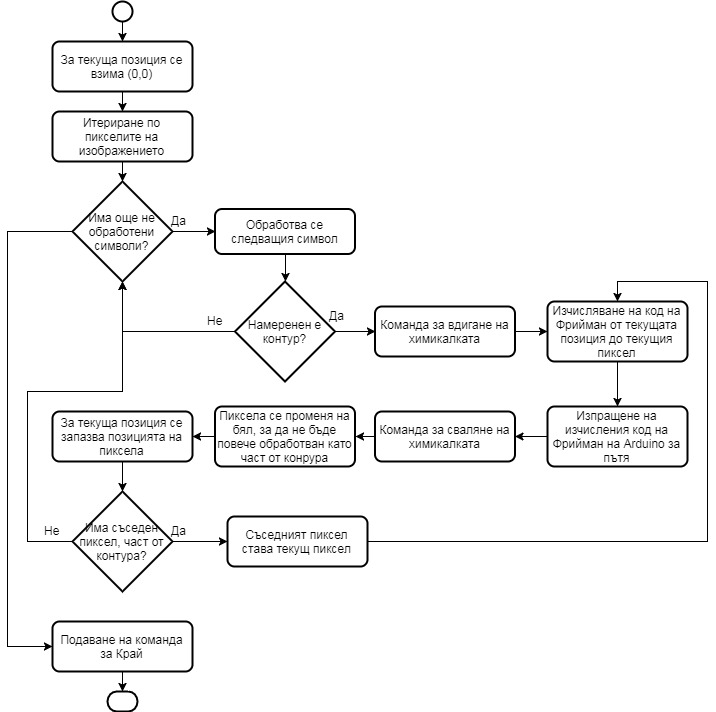


фигура 3Координати на Freeman code

На база кода на Фрийман дефинирани следните кодове:

* 0-7 – код на Фрийман за следващата позиция
* 8 – поняма в позицията на химикалката(двигане, а ко е свалена и сваляне, ако е вдигната)
* 9 – Край на изображението

Алгоритмът за обработка и предаване на комуникацията от Processing на Ardiuno:



фигура 4Изпращане на изображението от Processing към Arduino

### Изчертаване на изображението

//TODO: Да се опише логиката в Arduino кода

# Програмна имплементация на решението

Описанието на решението следва последователността и логиката на имплементацията на програмата. Пълният код на програмата може да бъде намерен в [GitHub](https://github.com/ElitsaVenchova/Plotter)(<https://github.com/ElitsaVenchova/Plotter>).

//TODO: Накрая да се постави целия код ТУК.

# Бюджет

* Рамка
  + 2\*Метални профили – 8.49 лв./бр.
  + Консумативи(нитове, болтове, лепило) – 5лв.
* Задвижване(не механични части)
  + 2\*винт + гайка 8мм – 13.99лв./бр.
  + 2\*Съединител с отвори 5-8мм - 5.83лв./бр.
  + Линейна направляваща ос 8мм – 13.43лв.
  + 2\*линейни лагери с обшивка продълговат 8мм – 9.59лв./бр.
  + ленеен лагер кръгъл 8мм – 1лв.
* Задвижване (механични части)
  + 2\*Биполярен стъпков мотор 42x47mm(NEMA 17)  – 33лв./бр.
  + 2\* Драйвер за стъпков мотор с А4988[10]– 5лв./бр.
  + Ardiono A-Star 32U5 Mini LV [11]– 35.10 лв.
  + Захранване 12V 4A – 29лв.
  + Джъмпери – 10лв.
  + 2\*Кондензатори - 1лв.
* Пишеща глава
  + Маркер – 2лв.
  + Серво мотор[12] – 10лв.
  + Части от стар хард диск и пружина – 2лв.

# Референции

[1] [Wikipedia/Плотер - https://bg.wikipedia.org/wiki/%D0%9F%D0%BB%D0%BE%D1%82%D0%B5%D1%80](https://bg.wikipedia.org/wiki/%D0%9F%D0%BB%D0%BE%D1%82%D0%B5%D1%80)

[2][Processing Official Site - https://processing.org/](https://processing.org/)

[3][GitHub/OpenCV for Processing - https://github.com/atduskgreg/opencv-processing](https://github.com/atduskgreg/opencv-processing)

[4][Docs/ОpenCV/Canny Detector - https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny\_detector/canny\_detector.html](https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html)

[5] <https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html>

[6] <http://www.kerrywong.com/2009/05/07/canny-edge-detection-auto-thresholding/>

[7]<https://www.pyimagesearch.com/2015/04/06/zero-parameter-automatic-canny-edge-detection-with-python-and-opencv/>

[8]<https://www.cse.unr.edu/~bebis/CS791E/Notes/EdgeDetection.pdf>

[9][А4988 - https://github.com/atduskgreg/opencv-processing](https://www.pololu.com/product/1201/faqs)

[10][Ardiono A-Star 32U5 Mini LV - https://www.pololu.com/product/3103](https://www.pololu.com/product/3103)