# Описание на промбема/задачата

Задача е да се направи реализация на CNC Writing machine, която ще наричаме плотер в този документ. 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 след това трябва да бъде обработено до формат, в който информацията ще може да се изчертае от плотера.

Плотера ще може да черта само в един цвят.

Решение и теоретична обосновка

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

• Хардуерна имплементация на плотера

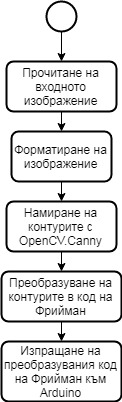
• Прочитане и обработка на изображението

• Изчертаване на изображението

# Решение и теоретична обосновка

# 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 използва библиотеката OpenCV for Processing[3], в което са имплементирани основните алгоритми за обработката на изображението, които ще бъдат използвани. Биоблиотеката не е развита напълно и възможностите на OpenCV липсват или са ограничени.

Фиг.1 описва стъпките на задачата.

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: Only one detector response per edge.

Стъпки на Canny edge detector[4][5]:

* Филтриране на шума – За тази цел се използва Гаусов филтър.
* Намиране на интензитет на градиента 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, което е част от самия алгоритъм, и др. Предимството, което е премахването на шумовете и многостъпковото определяне на контурите и премахването на излишните такива.

В библиотеката OpenCV for Processing има възможност единствено параметрите upper и lower threshold да бъдат подавани. Според препоръките на Canny тяхното с 2:1 и 3:1.

//TODO: Да се опише алгоритъм за определяне на lower threshold

На полученото изображение прилагаме алгоритъм на Фрийман.

# Програмна имплементация на решението

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

# Референции

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

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

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

[4][docs.opencv.org/canny\_detector](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/>