# Плотер Canny 2

Canny 2 е Arduino базирана плотер пишеща машина (Plotter Writing machine). Плотерът е изходно периферно устройство, което дава възможност за извеждане на графична информация върху хартия.[1]

Като вход Canny 2 трябва да получава изображение. Това изображение, трябва да бъде обработено до формат, в който информацията ще може да се изчертае върху хартия. Полученето изображение ще бъде изчертано само в един цвят - черен.

Най-общо плотерите намират приложение в[2]:

* Изчертаването на архитектурни скици и планове на сгради
* Изкустовото
* Графиките
* Създаване на елекртонни платки
* Принтиране върху текстил и др.

# Примерни реализации

[3] [CNC-Drum-Plotter - https://www.instructables.com/id/CNC-Drum-Plotter/](https://www.instructables.com/id/CNC-Drum-Plotter/)

[4] [AxiDraw V3 - https://shop.evilmadscientist.com/846](https://shop.evilmadscientist.com/846)

[5] <https://www.instructables.com/id/CNC-Drawing-Arm/>

# Описание на процеса на обработка и изчертаване

Плотерите имат за задача да направят от цифрово изображение еквиваленто изображение върху хартия. Този процес включва стъпките на прочитане на изображение, преобразуване на информацията в изображението във формат, подходящ за изчертаване чрез управлението на моторите, прочитане на преобразуваната инфромаци и задаването на коректите команди за изчертане.



Figure Oсновни стъпки на процеса

Прочитането на изображението зависи основно от израната технология за реализация. Необходимо е реализацията да поддържа входни операции от файл за прочитането на файла или подаваното изображение да е във формат, който се разбира от програмата(например xml).

При преобразуването на изображението в код, който може да се разчете като команди за изчертаване на изображението. Най-разпространения подход е преобразуване на изображението G-Code[6]. Това е език, чрез който може да се зададе на дадена машина да „как“ да направи нещо. Това задаване става чрез задаване на посока на движение на моторите, скорост, колко път да изминат и др. Освен G-Code може да се използват и други начини за кодирането на изображението. В текущата реализация е използван Код на Фрийман[7]. Кодът на Фрийман е алгоритъм за монохроматични изображения, който има цел да отдели свързаните компоненти на изображението като за всеки пиксел от компонентата връща нейната поция спрямо предишния пиксел от свързаната компонент.

Горните две стъпки често се реализират чрез външни програми, които като вход приемат входно изображение и генерират съответния изходен команден код. Такива са реализациите [2] и [3].

Изпълнението на командите зависи от физическата реализация на осите. При реализации [2] и [3] изчертаването на код, който съдържа съдържа команди по X и Y, то движението на моторите по съответната ос е еднозначно. При реализация [4] след прочитането на командите е необходимо изчисляването на траекторията на химикалката и преобразуването на командите за всеки мотор.

При реализации [2] и [3] много точно и правилно се изчертават прави линии, докато при реализация [4] много точно се изчертават криви, поради естествената траектория на точката, но изчертаването на прави може да бъде леко назъбено.

# Описание на архитектурата и използваните технологии

### Технологии

За програмирането на Arduino се използа езика за прорамиране Arduino(базиран на Wiring). Той се използа за контролиране и управление на микроконтролери.

За изпращането ,на командия код към Arduino ,се използва най-често езика Processing, който е Java базиран, което го прави и платформено независим.

За обрабработката на изображението се използва Inkscape с Gcodetools extension. Това е програма, която преобразува изображение в G-Code. В реализациите, които сами обработват изображението, е необходима графична библиотека – например OpenCV.

В този проект е израно обработката на входното изображение да бъде част от програмата и да не се използват външни програми. За обработката е избран език Processing, поради лесното реализиране на комуникацията между Processing и Arduino и платформената му независимост.

### Инфраструктурни изисквания

Модел на обслужването

Комуникацията между Processing и Arduino е P2P. При този модел съобщенията се предават директно между участниците. Те се обработват и при необходимост се връща съответния отговор. Комуникацията се осъществява чрез USB серийна комуникация.

В текущата реализация е добавено допълнително съгласуване, че изпратената команда е била прочетена. В даден момент има само една изпратена команда, след което Processing процеса спира и изчаква потвърждение от Arduino.

Софтуерен модел

Системата се състой от два паралелни процеса. Единият се изпълнява на Arduino и има за задача да слуша за входни команди и да подава информацията на актуаторите. Вторият процес върви на машината, която изпраща обработеното изображение.

Производителност и свързаност

Горната граница на производителността зависи от моторите и времето, за което правят една стъпка. Средно изискванията за производителност са 5 милисекунди на стъпка или 200 милисекунди на пиксел. Изчисленията на база средно секунда за един оборот.

# Видове плотери

Плотерите се различават основно по реализацията на движението по X и Y осите. Движението по Z остта се изразява единствено в контролирането на позицията на химикалката – вдигната и свалена.

### Видове според основата

Flatbed plotter

При този вид плотери[2] единствено се движи химикалката. Основата е плоска и напълно неподвижна.

Drum Plotters

Тези плотери[3] използват барабан или цилиндър, който движи хартията. Химикалката се вдижи единствено по X.

Сравнение

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ширина | височина | основа | размер | изходен материл |
| Flatbed plotter | ограничена | ограничена | плоска | ширината и дължината на изходното изображение | хартия, дърво, стъкло и др. |
| Drum Plotters | ограничена | почти неограничена\* | кръгла | размера на барабана | хартия |

\*ограничението е физическо

При плотерите с барабан не може да се чертае върху търди повърхности, но позволява височината на изображението да не е ограничено.

Плотерът с плоска основа е с размери поне колкото ширината и височината на изходното изображение, в сравнение с плотера с барабан, но не изисква допълнително констуиране за закрепването на харнията, за да може да се върти от барабана. Поради тази причина в текущия проект е реализиран плотер с плоска основа.

### Видове според разположението на осите

Тук ще се разгледат реализации на плоотери, които са с плоска основа. Основната разлика при тях е начина, по който се изразява движението за изчертаването на един обект.

Перпендикулярно разположение на осите

Това е стандартият начин за разположение на осите. Движението на химикалката по X и Y е може да се изчислява в декартовата координатна система.

Оси с общо начало

При този вид плотери, осите са общо начало, т.е. двата задвижващи мотора за един до друг. Движението по X и Y става след като предварително се изчисли движението на точката(химикалката). Това изчисляване става чрез законите по кинематика за равнинно движение на точка.

Сравнение

Освен по начина, по който трябва да се изрази движението за изчертаването на един обект, те се различават и по точност в изчертаването. Плотерите, с перпендикулярни оси, се справят отлично с изчертаането на плари линии, а при плотерите, с общо начало на осите, много по-лесно се задава изчертаването на дъги. Също при перпенд

В тази реализация е избрано перпедикулярно разположение на осите, поради лесното изразяване на движението, което е поради по-малкото степени на свобода.

### Други подходи в реализацията

Тук ще се разгледат други подходи за реализация, от които не рависи кода на програмата, но също трябва да се разгледат поради основната им роля в конструкцията.

Задвижване на химикалката

Начина за задвижване на химикалката най-вече зависи от конкретната реализация и наличните материали.

Най-простата конструкция е чрез използването на гравитацията и мотор, който единствено вдига химикалката. Моторът повдига конструкцията на химикалката от някоя нейна точка.

Към горната конструкция може допълнително да се добави прожина, която да спомага за избутването на химикалката.

Другият ширикоро разпространен мариант е този, при който химикалката се завдижва винтово или в ремък по направляваща ос. Този вариант е по-скъп за реализация, поради необходимоста от допълнителни оси или ремъци и съответни свъзрзащи звена към тях, но преполага по-голям контрол и прецизност при движението на химикалката, защото не се разчита на гравитацията и триенето. Също тази реализация изисква по-голяма площ, която ще бъде заето от допълнителните части.

В текущата реализация е избрана контрукция, в която химикалката се повдига от мотор, като допълнително е сложена пружина в горната част на направляващата ос, която да спомага за избутването на химикалката в долна позиция. Изборът е направен на база ограниченото пространство за констукцията на химикалката.

Задвижване на осите

Първията реализация е чрез ремъци и направляващи оси. Самото движение се управлява от мотор, към който има закрепено зъбно колело, през което минава ремък. Контрукцията на оста, която ще се движи е закрепена неподвижно към някоя част на ремъка. Така при въртене на мотора, движението се превръща в линейно.

Предимствата[8] на реализацията са:

* добър трансфер на мощност,
* добра скорост
* сравнително евтината цена за реализирането.

Недостатъци:

* триенето и износването на гайката могат да предизвикат хлабина
* при по-дългите може да се получат големи вибрации.

Другата разпространена реализация е чрез винтново задвижване. Към мотора има закрепена винтова ос. Допълнително на оста има закрепена гайка, която носи цялата контрукция задвижвана от оста. При въртенето на мотора и винтовата ос, се получава линейно движение на гайката по оста.

Предимствата са:

* може да се осигури хубава скорост,
* лесен за поддържане, поради единствена необходимост от обтягане на ремъците
* прецизно движение

Недостатъци:

* изскват се две направляващи оси(при винтовото задвижване е необходима една, като другата се явява самия винт)
* понижаване на силата
* при дългите ремеци, може да се получи прескачане на зъб и това да предизвика трептене

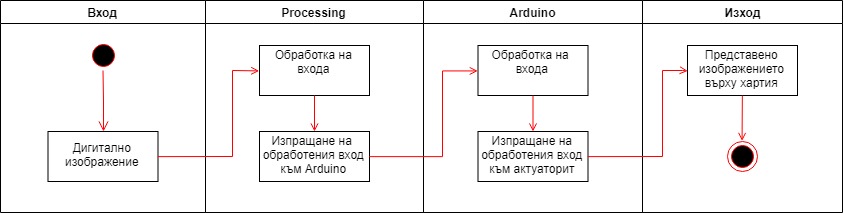
В текущава реализация е израно винтово задвижване поради по-ниската цена, което се съобразява основно с избягването на поставяне на две направляващи оси и допълнителната ос на ремъка.

# Решение и теоретична обосновка

Решението може да се разделят да следните основи задачи:

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

### Структура на решението



Фигура 1Структора на решението

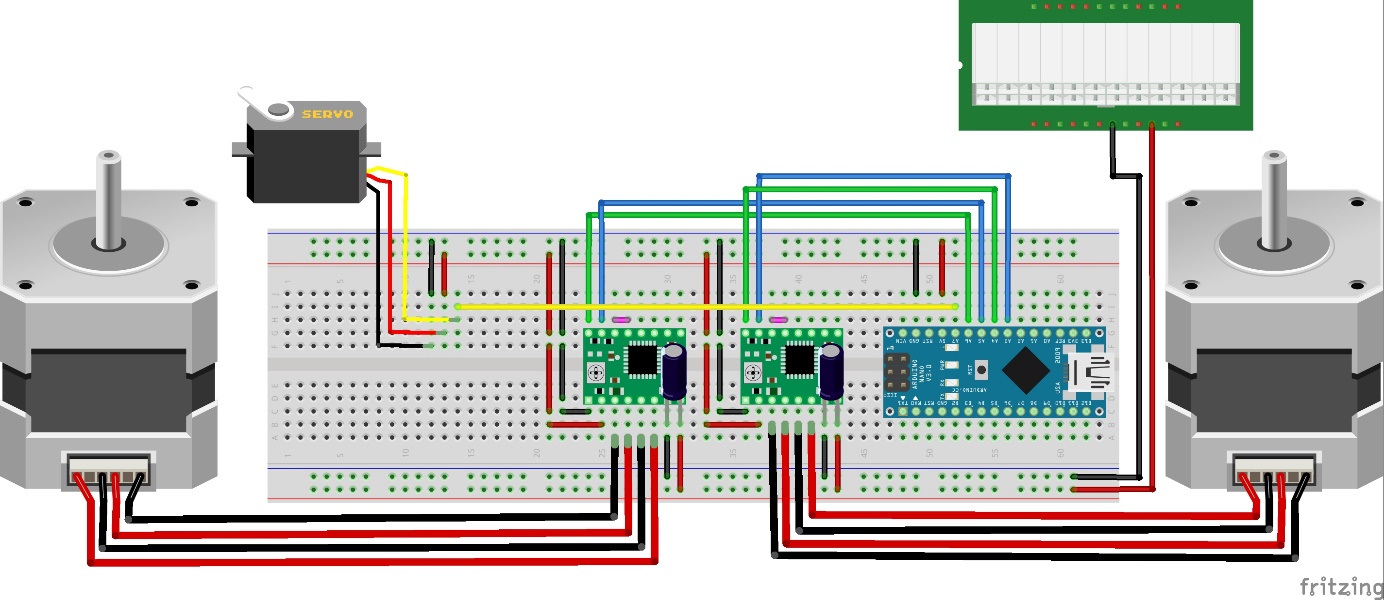
Canny2 се състой от две основни части – Обработка на изображението и изчертаване на изображението, което е реализирано в два отделни процеса.

Обработката на изображението се осъществява с помощта на програма реализирана на Processing. 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.

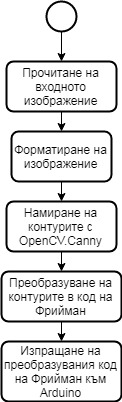
Управлението на химикалката е чрез серво мотор, който я вдига и пуска.



Фигура 2Схема на свързване

### Прочитане и обработка на изображението

За обработката на изображението в Processing 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 3Описание 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.

##### Canny edge detector

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[9]:

* 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[9][10]:

* 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[10].

В имплементацията се използва статистически подход на база стойностите на пикселите в изображението[11][12]. 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 [11] не постигат същата добър резултат. Това най-вероятно се дължи на различните имплементации на Canny и подавани параметри по подразбиране на отделните стъпки.

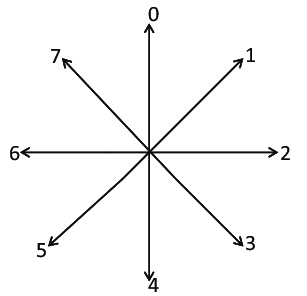
##### Treshold и Adaptive Threshold

Това е подход за обработка на изображения. Първо изображението се преобразува в сивия спектър. След това на база стойност, всеки пиксел, който е над дадената стойност, става черен, а всеки под, става бял. Разликата между двата подхода е, че Threshold използва една стойност за цялото изображение, която се подава предварително.

В Adaptive Threshold се използва плъзгащ прозорец(квадратен) с определен размер. Прозореца преминава последователно цялото изображение, threshold стойността се определя на база съседните пиксели на централния.

##### Chain Freeman Code

Следващата стъпка е контурите на изображението да се изпратят на Arduino за изчертаване. Използва се Chain Freeman Code[12] 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:

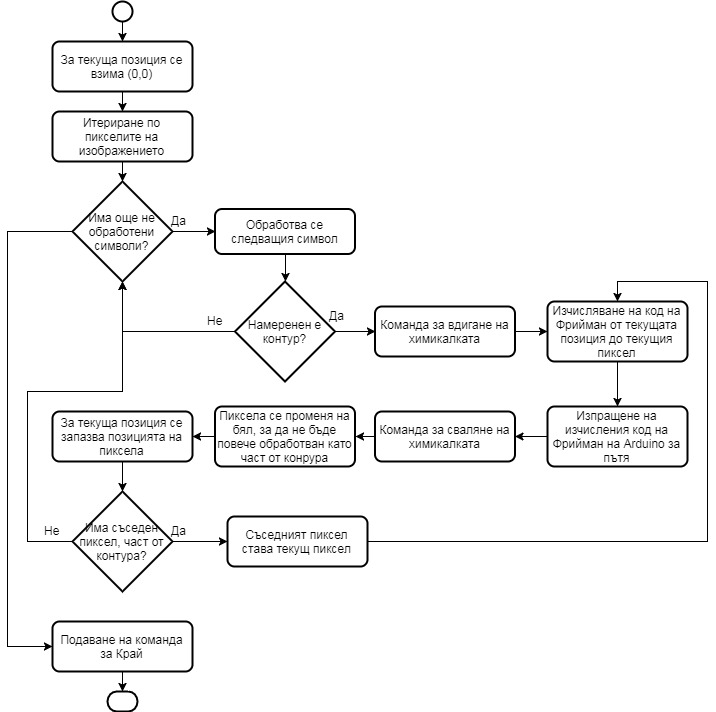


фигура 4Координати на Freeman code

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 Arduino:



фигура 5Изпращане на изображението от Processing към Arduino

### Изчертаване на изображението

В логиката на Arduino е заложено да прочита командата подадена от Processing и след това да я изпълнява. Командата се чете байт по байт до прочитане на терминиращия символ на командата ‘е ‘. След това се изпълнява командата като за преместване или промяна на позицията на химикалката. По време на всяко преместване на стъпковите мотори се следи за излизане от допустимите рамка, която би довело до физически щети върху плотера.

Изчертаване на един пиксел от моторите се равнява на 20 стъпки. Движението по диагонал е направо да се извършва едновременно от двата мотора стъпка по стъпка.

Комуникация между Processing и Arduino се осъществява като от Processing се изпраща команда, Arduino прочита командата и връща като отговор, че командата е прочетена и пусната. След получаване на потвърждението, Processing пуска следващата команда.

### Компоненти

* 2\*Биполярен стъпков мотор 42x47mm(NEMA 17)
* 2\* Драйвер за стъпков мотор с А4988[15]
* Ardiono A-Star 32U5 Mini LV [16]
* Захранване 12V 4A
* 2\*Кондензатора
* Серво мотор MG90S[17]

# Описание на разгръщането на приложението

# 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] http://www.it4nextgen.com/plotter-definition/

[3] [CNC-Drum-Plotter - https://www.instructables.com/id/CNC-Drum-Plotter/](https://www.instructables.com/id/CNC-Drum-Plotter/)

[4] [AxiDraw V3 - https://shop.evilmadscientist.com/846](https://shop.evilmadscientist.com/846)

[5] <https://www.instructables.com/id/CNC-Drawing-Arm/>

[6] <https://en.wikipedia.org/wiki/G-code>

[7] <https://en.wikipedia.org/wiki/Chain_code>

[8] <http://www.cnc4everyone.com/hardware/linear-drives/>

[9][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)

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

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

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

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

[14][А4988 - https://www.pololu.com/product/1182](https://www.pololu.com/product/1182)

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

[16]Курс по Обработка на изображения във ФМИ, 2018/2019г.