**Факултет инжењерских наука Универзитета у Крагујевцу**


**Назив студијског програма: Машинско инжењерство**

**Ниво студија: Мастер академске студије**

**Модул: Информатика у инжењерству**

**Предмет: Брза израда прототипова**

**Број индекса: 335/2015**


## Александар В. Синђелић


# Пројектовање, израда и програмирање ЦНЦ машине за 3Д обликовање жице

## Мастер рад

**Комисија за преглед и одбрану:**

1. **Проф. др. Ненад Грујовић - ментор**
2. _____
3. _____ Датум одбране:_____
4. _____ Оцена:_____

**У оквиру овог мастер рада кандидат треба да презентује процес реализације и пројектовања аутоматског система управљања, изврши пројектовање и израду уређаја, одради комплетан програмски код за беспрекорно функционисање уређаја и његово примењивање у процесу контроле уз коришћење Ардуино платформе.**

**Препоручена литература:**

[1] Хенсен А.: Програмирање на језику С, Микрокњига, Београд, 1991.

[2] Милићев Д.: Објектно оријентисано програмирање на језику С++, Микрокњига, Београд, 1991.

[3] Ћирковић Р.: Програмирање CNC машина: FeatureCAM, Микрокњига, 2015.

[4] Massimo Banzi „Getting Started with Arduino, 2nd Edition“, 2011.

[5] Casey Reas & Ben Fry „Getting Started with Processing, First Edition“, 2010.

[6] Трајановић М., Грујовић Н., Миловановић Ј., Миливојевић В..: Рачунарски подржане брзе производне технологије, Крагујевац, 2008.

Крагујевац, 2020. ментор: др. Ненад Грујовић, ред. проф

# Садржај

**Summary of work:**

Master thesis “Design, manufacture and programming of CNC machine for 3D wire forming” represents the creation of a wire bending machine. The processes of designing and making the entire device, its programming and creation of the necessary software as well as putting the device into operation are presented. Certain calculations have been performed by which the entered values are converted into machine instructions. An Arduino platform has been implemented, which is programmed in the C programming language, whose task is to perform fully automated device control.

**Key words:**

Arduino, CNC machine, object-oriented programming, wire bending.

**Резиме рада:**

Мастер рад „Пројектовање, израда и програмирање ЦНЦ машине за 3Д обликовање жице“ представља креирање машине за савијање жице. Приказани су процеси дизајнирања и израде целокупног уређаја, његово програмирање и креирање потребних софтвера као и пуштање уређаја у рад. Извршени су одређени прорачуни помоћу којих се унете вредности претварају у машинске инструкције. Имплентирана је Ардуино платформа која се програмира у програмском језику C, чији је задатак да обавља потпуно аутоматизовану контролу уређаја.

**Кључне речи:**

Ардуино, ЦНЦ машина, објектно оријентисано програмирање, савијање жице.

# 1. Увод

Задатак овог мастер рада као што и сам назив теме говори, јесте да се од саме нуле направи машина која ће вршити савијање жице на основу унетих инструкција оператера. Односно, потребно је пројектовати машину, израдити је, програмирати и затим пустити у рад. Да би се уопште упустили у овакав подвиг, прво је потребно разумети шта представља рачунарска нумерички управљана машина (енг. *CNC - Computerized numerical control machine).*

Рачунарска нумерички управљана машина је врста обрадне машине која све операције реализује према унетом алфанумеричком код-у преко рачунарске управљачке јединице која директно управља радом машине. Као што и сам назив каже, ове врсте машина контролишу се преко рачунара путем одређених софтвера који су специјално намењени за њихово коришћење. На тај начин, постиже се већи квалитет израде производа, као и већа продуктивност јер се машина не замара. Поред тога могуће је израдити и сложеније облике које је врло тешко израдити ручном обрадом. [3]

Поред рачунара свака ЦНЦ машина мора да има и свој управљачки модул који се користи да би генерисао одговарајуће електричне сигнале, помоћу којих се врши контрола ових машина. Управљачки модул, жаргонски речено „мозак“, јесте скуп електронских компоненти и чипова који врше контролу електричних, као и електро-механичких делова уређаја. Поред контроле он може преко одговарајућих сензора да врши и мерење променљивих параметара, као што су температура, притисак, ниво течности у резервоару итд. и на основу њих да врши контрола одговарајућих делова машине. Пошто је примена ЦНЦ машина у савременој индустрији доста широка то значи да управљачки модул мора да буде специјално пограмиран и прилагођен за одређену врсту машине. [3]

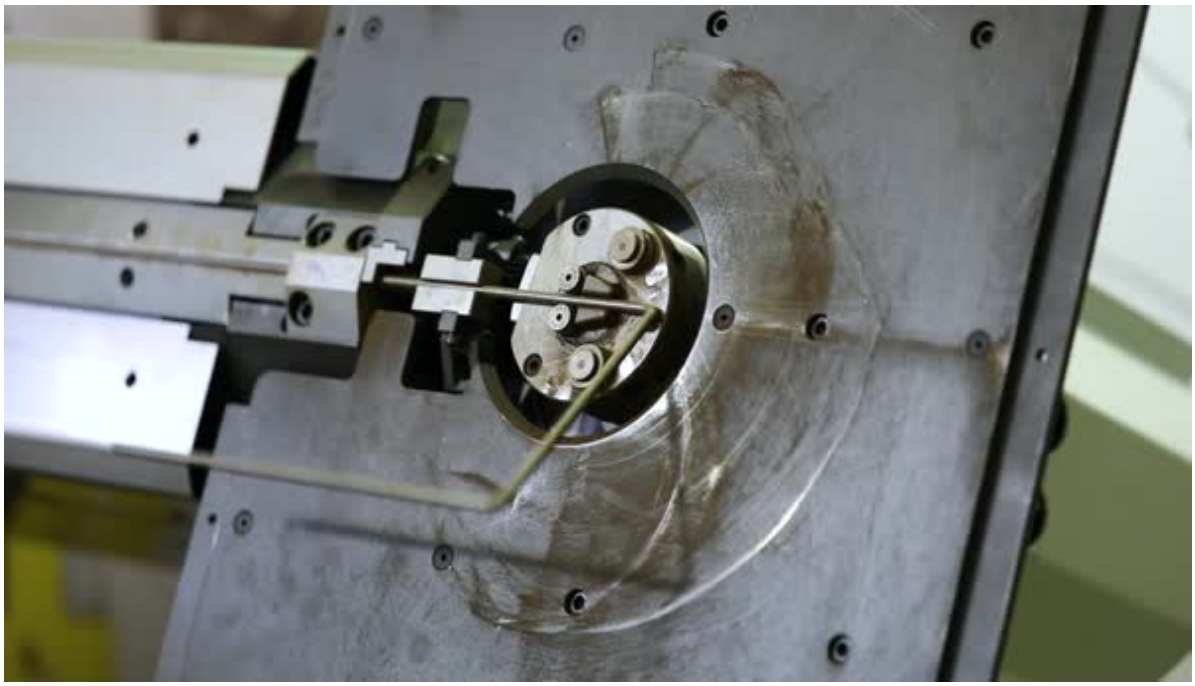
Примери где се користе ЦНЦ машине јесу:

- код обраде материјала глодањем,
- код стругова,
- код рутера,
- за машинско заваривање,
- за ласерско и водено сечење,
- код робота на покретним линијама,
- код 3Д штампања,
- за обликовање жице итд.

Пример једне индустријске машине за савијање жице може се видети на слици 1.

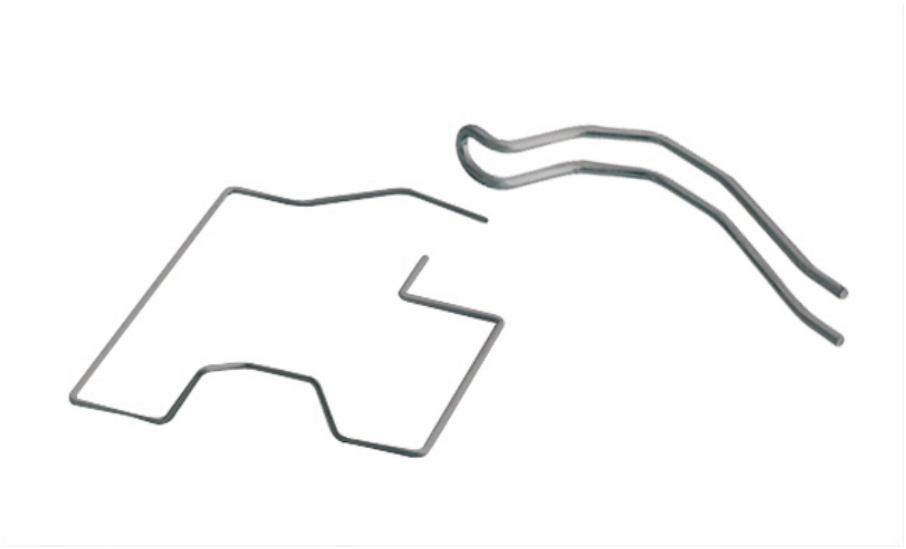
**Слика 1.** *Индустријска ЦНЦ машина за савијање жице марке "Nicemach 3D-R70" [9]*

Радно постоље, односно део на ком се врши савијање жице може се видети на слици 2.



**Слика 2.** *Радно постоље на индустријској машини за савијање жице [9]*

На слици 3. може се видети неки од примера како жица изгледа након савијања на некој од индустријских машина намењених за савијање жице.



**Слика 3.** *Примери савијене жице на индустријској машини за савијање [9]*

Као што се може видети на примеру индустријске машине за савијање жице, може се закључити да постоје врло озбиљне и професионалне машине овакве врсте. Циљ овог рада јесте да се направи машина у смислу кућне варијанте која ће бити лака за коришћење и која ће моћи да врши савијање жице.

# 2. Конструкција машине за 3Д савијање жице

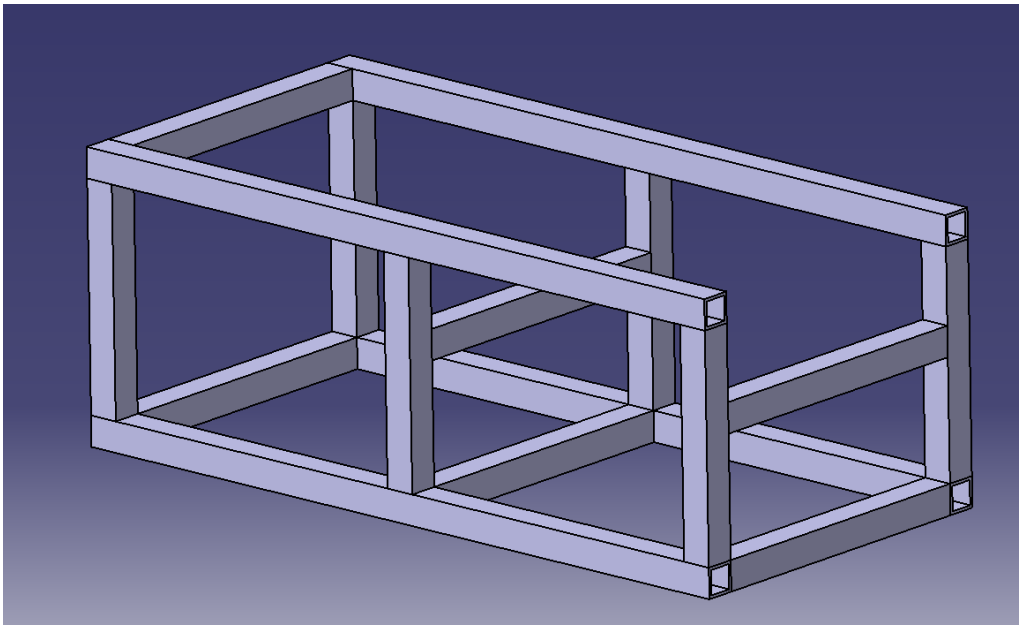
Први и основни корак у дизајнирању новог производа јесте разумети његову примену и захтеве које он треба да испуни. Поред тога, треба имати на уму и друге факторе, као што су доступност материјала, могућност израде и на крају финансијски фактор. У овом случају најбитнији фактор јесте била финансијска ситуација која директно утиче на одабир конструктивних елемената, као и осталих пратећих делова машине.

Након анализе свих фактора, први корак јесте конструисање машине. Данас се овакве врсте машина конструишу од више врста материјала, специјалних алуминијумских облика (профила), израдом специјалних делова итд. С'обзиром да је буџет био врло ограничен то је довело до сужења избора материјала. У овом пројекту коришћени су искључиво стандардни конструкциони челични профили, који се могу наћи на сваком стоваришту и врло су доступни. Наравно, било је потребно израдити и неколико специфичних делова такође од челичних профила, с'тим што је за њихово обликовање коришћен ласер за сечење челика.

Конструисање је обављено у софтверу CATIA у ком је и стечено знање на Факултету унжењерских наука у Крагујевцу. Целокупна конструкција може се поделити у четири делова:

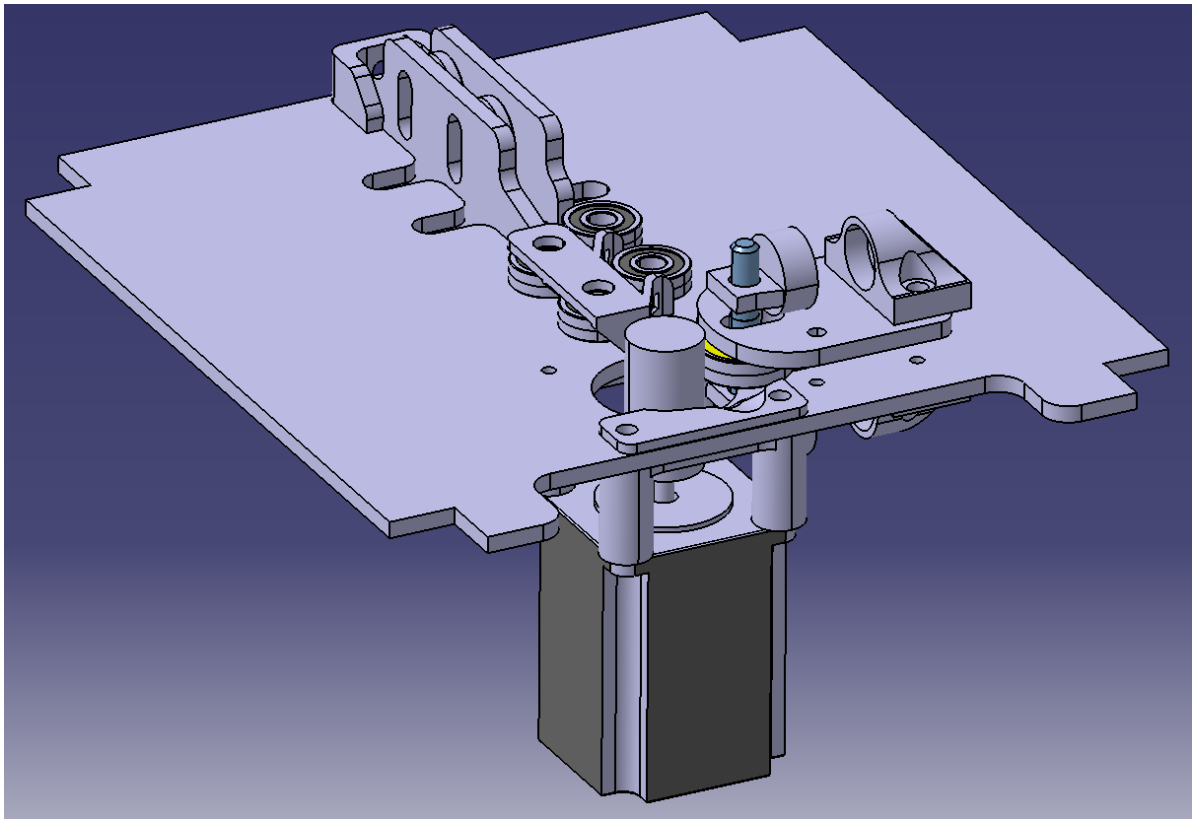
- рам машине,
- групација са лежајевима за исправљање жице и мотором за њено увлачење,
- глава за савијање жице са окретном цеви,
- групација електричних компоненти.

Рам машине јесте непокретни део машине и он представља ослонац машине на под. Он је израђен од цевастих челичних профила квадратног попречног пресека, димензија са дужином странице од 20 mm и дебљине зида од 2 mm. Крајње димензије рама машине јесу 500 x 240 x 190 mm. Након што су профили скраћени на одређене дужине они су спојени заваривањем. На слици 4. може се видети конструкција рама машине у софтверу CATIA.



**Слика 4.** *Рам машине израђен од стандардних конструкционих челичних профила*

Групација са лежајевима за исправљање жице и мотором за њено увлачење представља део машине кроз који жица пролази пре њеног савијања. Овај део машине је од велике важности јер врши њено исправљање помоћу кугличних лежајева. Такође у овом делу се налази степ мотор који врши њено увлачење. Већина делова који се могу наћи у овом делу јесу израђени од челика помоћу ласерског сечена и пластичних делова који су израђени од ABS пластике помоћу 3D штампача. Као што се може видети, у изради ове ЦНЦ машине су заправо коришћене ЦНЦ машине других врсти што значи да је њихова примена веома распрострањена и неопходна у модерној индустрији. На слици 5. може се видети ова групација приликом дизајнирања. Треба напоменути да шрафови и опруге нису приказани.

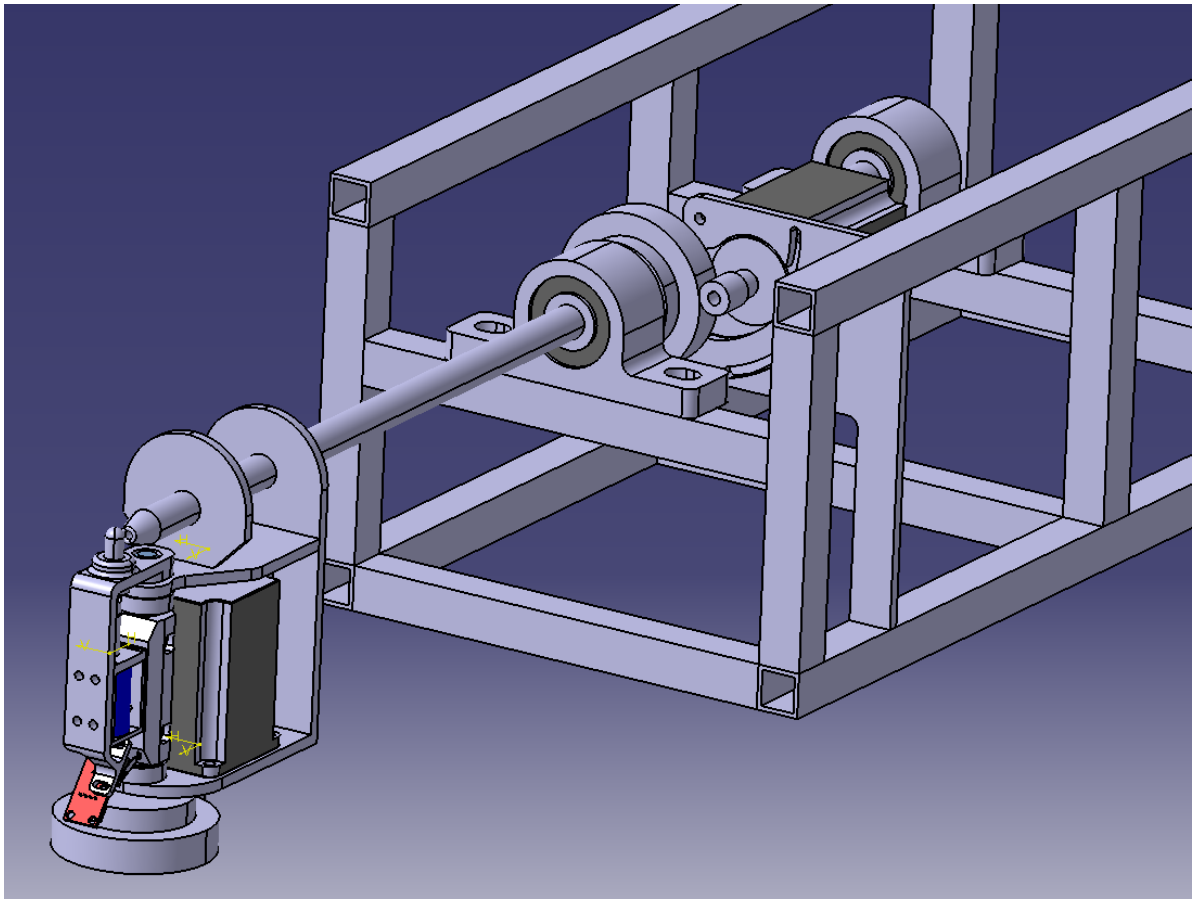


**Слика 5.** *Групација са лежајевима за исправљање жице и мотором за њено увлачење*

Глава за савијање жице са окретном цеви јесте део машине где се скоро сав посао за који је ова машина немањена обавља. Као и у претходној групацији и ова групација је претежно израђена од челичних делова који су сечени ласером, као и пластичним деловима израђених од ABS пластике помоћу 3Д штампача. Цев кроз коју пролази жица јесте израђена од челика, пречника 15 mm и дебљине зида 1,5 mm. На једном крају цеви налази се степ мотор помоћу ког се врши заокретање клина који врши савијање жице. Пренос обртног момента са мотора на осовину на којој је прикачен клин, као и са мотора помоћу ког се врши ротација цеви да би се добила трећа димензија врши се помоћу челичних зупчаника. У оба случаја погонски зупчаник састоји се од 15 зуба, док се гоњени састоји од 70 зуба и на тај начин је извршено смањење оптерећења које ови мотори морају да издрже. У овом случају преносни однос износи:

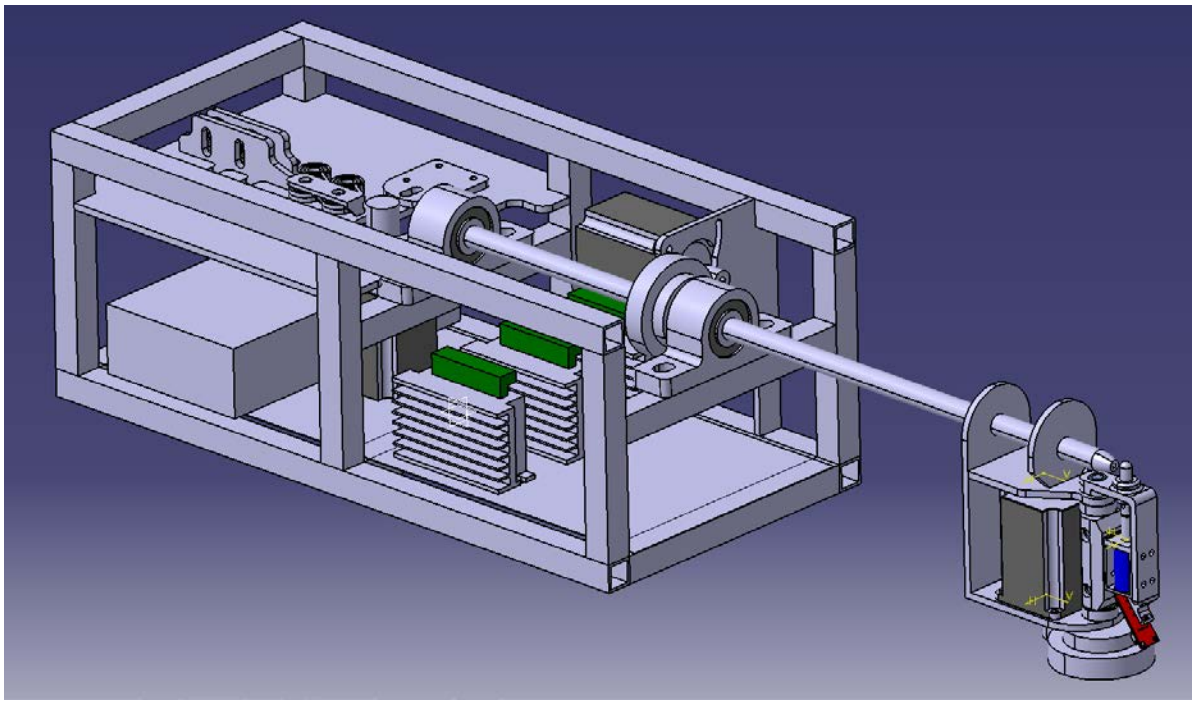
$$i = \frac{70}{15} = 4{,}66$$

Ова групација може се видети на слици 6 где је прикачена на раму машине.



**Слика 6.** *Глава за савијање жице са окретном цеви прикачени на раму машине*

Групација електричних компоненти представљају извор електричног напајања, драјвери за степер моторе и Ардуино контролер, и ове компоненте смештене су у доњем делу рама машине. Целокупан дизајн машине може се видети на слици 7.

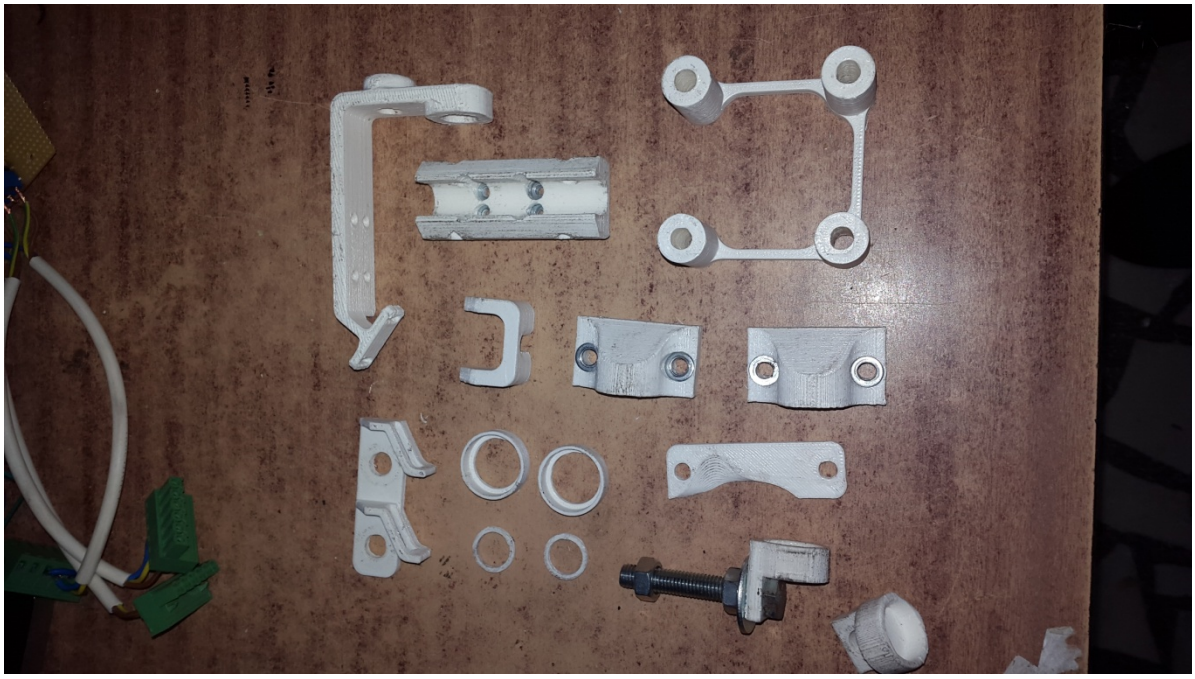


**Слика 7.** *Целокупан дизајн машине*

Након завршеног дизајнирања дошао је тренутак да се почне за призводњом машине. Први корак јесте био да се изради челични рам и крај његове израде може се видети на слици 8.

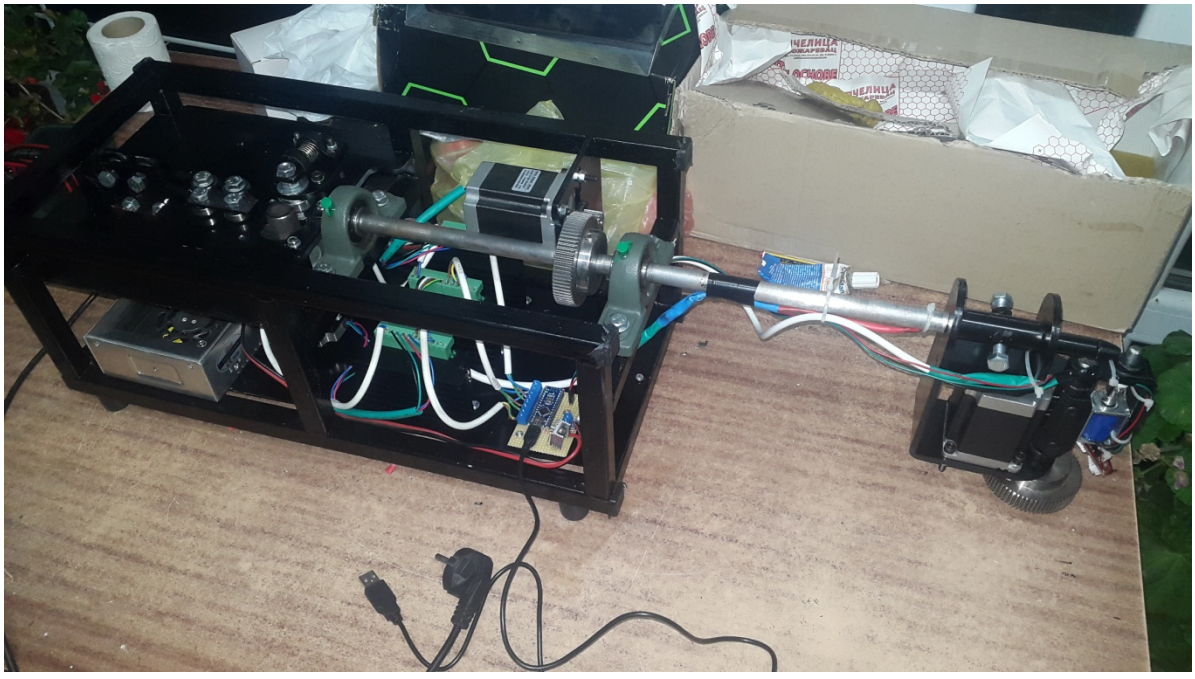
**Слика 8.** *Израђен рам машине од челичних профила*

Након тога су израђени сви пластични делови на 3Д штампачу. Сви делови израђени су од ABS пластике и могу се видети на слици 9. [6]



**Слика 9.** *Делови од ABS пластике израђени на 3Д штампачу*

После израде свих потребних делова дошао је и тренутак да се сви делови склопе у коначан производ, односно машину за савијање 3Д жице, која се може видети на слици 10.



**Слика 10.** *Машина за савијање 3Д савијање жице*

# 3. Електронски управљачки модул

Електронски управљачки модул представљају више компоненти које имају задатак да обезбеде беспрекоран и поуздан рад сваке машине. Овај модул прво и основно чини извор једносмерне електричне енергије, познатији као извор напајања. Следећа ставка јесу електро мотори помоћу којих се у овом случају врши увлачење жице као и покретање одређених делова машине. Затим се ту налазе и њихови управљачи тзв. "драјвери" који имају имају улогу да контролишу рад мотора на основу примљених инструкција од микропроцесорке јединице. Ова јединица представља срце сваке машине и њена улога јесте да контролише рад свих компоненти.

## 3.1. Степер мотори, драјвери и њихово напајање

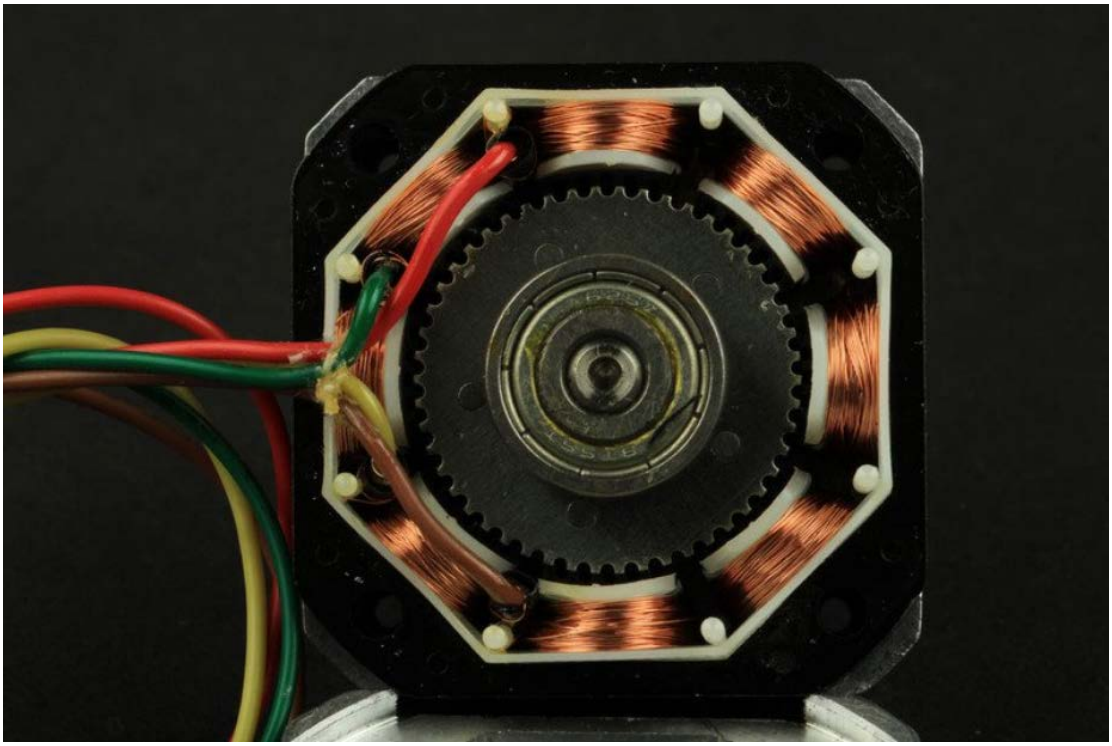
За овакве врсте машина најчешће се за њихово покретање користе степер или серво мотори. С'обзиром да серво моторе карактерише висока цена, за овај пројекат одлучено је да се користе степер мотори. Одабир одговарајућег степер мотора може бити веома конфузна ситуација и у том случају најпоузданија солуција јесте ослонити се на нечије искуство. Ради се о томе што се обртни момент код ових мотора карактерише у стању мировања мотора, односно произвођачи ових мотора дају максимални обртни момент који ови мотори имају када се не покрећу. То значи, да када је мотор напајан струјом и у стању мировања је, обртни момемент који произвођач тврди, представља обртни момент који је потребно применити на осовину мотора да би се она окренула. Односно, на тај начин може се прорачунати које оптерећење они могу да држе када стоје и напајани су струјом. Што се тиче обртног момента за време рада ових мотора на њега утичу доста фактора као што су: брзина обртања, напон напајања и у ком режиму се врши његова контрола. [7]

За овај пројекат изабрани су NEMA23 степер мотори са струјним карактеристикама које су дате у табели 1:

| Модел | Угао заокрета | Номинални напон | Номинална струја | Отпор по фази | Индукција по фази | Обр. момент у мировању |
|---|---|---|---|---|---|---|
| 57BYGH627 | 1.8° | 3V | 3A | 1.0Ω | 1.6mH | 1.86Nm |

**Табела 1.** *Струјне карактеристике изабраних степер мотора*

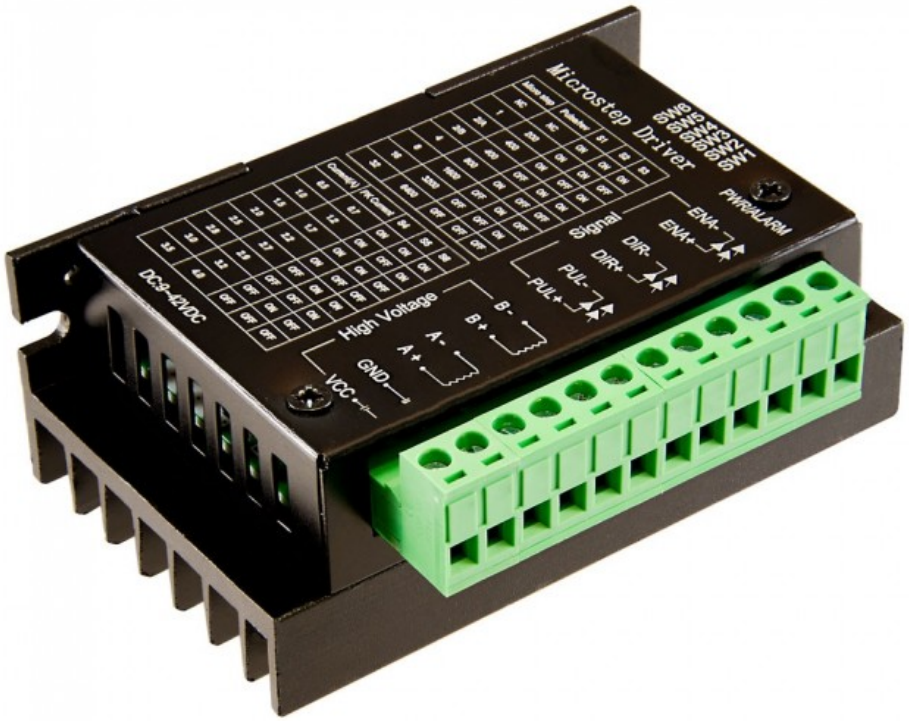
Да би се боље разумело шта значи угао заокретања од 1.8°, потребно је погледати слику 11. на којој се може видети отворен степер мотор.



**Слика 11.** *Отворен степер мотор [7]*

На слици 11. може се видети отворен степер мотор, са осам намотаја који су везани у две фазе. На ротору мотора може се приметити да су његове ивице назубљене. Приликом протицања електричне струје кроз намотаје статора врши се заокретање ротора док се не изврши поравњање назубљених ивица. Угао од 1.8° представља да је за пун круг потребно извршити 200 корака, односно 200 промена протицања струја кроз калемове. Адекватно поређење би било да је, у основи ваљка уместо кружнице један полигон са 200 једнаких страница. Приликом котрљања тог ваљка тошло би то нежељених вибрација и уједначен рад не би био могућ. Поред тога на овај начин смањена је и прецизност јер се осовина ротора са променом струје, увек окреће за исти угао. Из тог разлога постоје тзв. "драјвери" који врше управљање овим моторима, односно регулацију струје у његовим калемовима. [7]

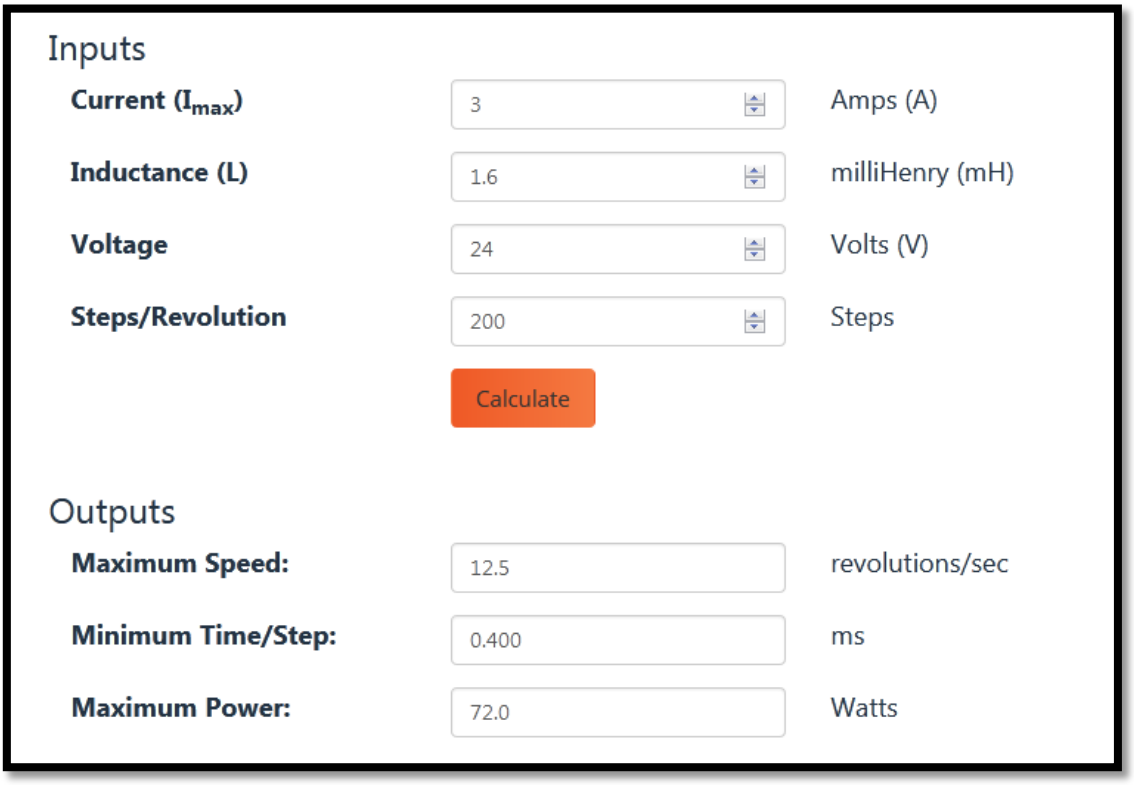
Приликом одабира одговарајућег драјвера за степер моторе потребно је водити рачуна о максималној излазној струји који тај драјвер може да издржи. С'обзиром да је номинална јачина струје ових мотора 3А, потребно је одабрати драјвер који може издржати ту јачину струје. У овом случају одабран је драјвер " TB6600", чија је маскимална јачина излазне струје 4А, а улазни напон напајања од 9-42V. Поменути драјвер може се видети на слици 12. [7]




**Слика 12.** *Изабрани драјвер за степер моторе*

На самом драјверу може се видети табела са положајима прекидача за одговарајућа подешавања. Поред подешавања за јачину излазне струје, могу се видети и подешавања за корекцију степовања мотора. Када је подешавање подешено на 1, тада мотор ради у нормалном режиму, тачније са пуним напајањем фазе врши се заокретање осовине за одговарајући угао. У овом случају подешавање је постављено на 1/16, у тзв. "microstepping" режиму рада. Практично то значи, иако је физичка контрукција мотора подељена на 200 корака, у овом режиму рада добија се подела 16 пута већа, односно подела од 3200 корака. Физички то се изводи на тај начин што су струје у фазама постепено смањују и повећавају, уместо да се наизменично пале и гасе. На тај начин добија се равномерно обртање осовине мотора, без вибрација. Наравно, ово негативно утиче на обртни момент које се смањује приликом повећања корака. [7]

Напајање ових мотора врши се преко извора напона једносмерне струје. Напајање се бира на основу више фактора. Први и основни јесте лимит драјвера који он може да издржи. Пошто је у овом случају тај опсег од 9-42V, вредност улазног напона треба да буде у тим границама. Пошто постоје стандардни извори напајања, у овом случају изабран је извор напајања од 24V. На слици 13. може се видети интернет калкулатор који врши прорачун максималног броја обртаја мотора у зависности његових карактеристика. [8]




**Слика 13.** *Калкулатор за прорачунавање излазних карактеристика степер мотора [8]*

### 3.2. Микропроцесорска контролна јединица

Након одабира мотора помоћу којих се врши кретање кретних делова машине, потребно је одабрати одговарајућу микропроцесорску јединицу за њихово управљање. У овом случају одабрана је Ардуино Нано која се може видети на слици 14. Ардуино Нано је микроконтролерска плоча базирана на ATmega328 микроконтролеру. Поседује 14 дигиталних улаза/излаза (од којих 6 могу служити како PWM излази), 6 аналогних улаза, 16MHz кристал осцилатор, USB конектор за програмирање и везу с рачунаром, ICSP конектор и тастер ресет. [10]


**Слика 14.** *Ардуино Нано микропроцесорска јединица*

Техничке спецификације могу се видети у табели 2:

| **Микроконтролер** | **ATmega328** |
|---|---|
| **Радни напон** | 5V |
| **Препоручљив улазни напон** | 7-12V |
| **Границе улазног напона** | 6-20V |
| **Дигитално улазно/излазни пинови** | 14 (of which 6 provide PWM output) |
| **Аналогно улазни пинови** | 6 |
| **Јачина струје по у/и пину** | 40 mA |
| **Јачина струје за 3.3V пин** | 50 mA |
| **Флеш меморија** | 32 KB of which 2 KB used by bootloader |
| **SRAM** | 2 KB |
| **EEPROM** | 1 KB |
| **Радни такт** | 16 MHz |

**Табела 2.** *Техничка спецификација коришћене микропроцесорске јединице [10]*

Да би уопште било могуће контролисање прво је потребно извршити програмирање ове микропроцесорске јединице, односно она мора разумети на који начин се врши управљање степер моторима преко њихових драјвера.

### 3.3. Окружење за програмирање микропроцесорске контролне јединице

Програмирање Ардуино платформе врши се у Ардуино интегрисано развојном окружењу које представља апликацију написану у „Java“ програмском језику. Оно је креирано тако да у програмирање уведе ученике, студенте и остале почетнике који нису упознати са начином развоја софтвера. Састоји се од уређивача кода са могућностима као што су означавање кода, упаривање заграда, аутоматско увлачење линија. Овај уређивач може да преведе код а затим га и пребаци у чип једном командом. У овом случају није потребно подешавати параметре превођења кода или покретати програме из командне линије. Ардуино интегрисано развојно окружење долази са софтверском библиотеком званом „Wiring” која чини уобичајене улазно-излазне операције веома једноставним. Ардуино програми се пишу у C/C++ програмском језику, мада корисници морају да дефинишу само две функције како би направили извршни програм. [13]

Те функције су:

- setup() – функција која се извршава једном на почетку и служи за почетна подешавања;
- loop() – функција која се извршава у петљи све време док се плоча не искључи.

Испод је дат пример једноставног Ардуино програма који пали и гаси LED диоду са међусобним чекањем од једне секунде:

```
#define LED_PIN 13

void setup () {
    pinMode (LED_PIN, OUTPUT);     // definiši pin 13 kao digitalni izlaz
}

void loop () {
    digitalWrite (LED_PIN, HIGH);  // uključi LED
    delay (1000);                    // sačekaj jedan sekund (1000 milisekundi)
    digitalWrite (LED_PIN, LOW);   // isključi LED
    delay (1000);                    // sačekaj jedan sekund
}
```

### 3.4. Програмски код за микропроцесорску јединицу

Пре почетка програмирања потребно је имплементирати одговарајуће програмске библиотеке, дефинисати све пинове, променљиве и константне вредности. За управљање моторима користи се библиотека „Stepper.h" која је стандардна библиотека у склопу Ардуино интегрисаном окружењу. Програмски код у коме се дефинишу и имплементирају претходно поменуте компоненте и библиотеке приказан је испод: [4] [15] [16]

```
#include <Stepper.h>

//Broj koraka potrebnih za jedan obrtaj osovine motora
const int stepsPerRevolution = 3200;

//Dodeljivanje pinova
const int bendMotorPls = 6;
const int zMotorPls = 2;
const int feedMotorPls = 10;
const int bendMotorDir = 8;
const int zMotorDir = 4;
const int feedMotorDir = 12;
const int benderPin = 9;
const int homeButtonPin = 3;

//Definisanje steper motora sa odgovarajucim pinovima
Stepper zRotation(stepsPerRevolution, zMotorPls,zMotorDir);
Stepper bender(stepsPerRevolution, bendMotorPls,bendMotorDir);
Stepper extruder(stepsPerRevolution, feedMotorPls,feedMotorDir);

//Definisanje smera obrtanja motora
boolean ccw = true; //counter-clockwise
boolean cw = false; //clockwise
boolean curDir = cw; //Pomocna promenljiva

//Definisanje potrebnih parametara
int fieldindex=0;
byte values[1200];
float curAngleZ=0;
const float offset = 17.2;
const float r1 = 26.5;
const float r2 = 5;
const float increment=0.01;
const int homeZeroPos = 16200;
const int inc = 4;
float zero;
float correction = 0;
const int safetyDistance = 2500;
```

Након дефинисања свих параметара долази функција „setup()” у којој се врши подешавање улазних и излазних пинова, подешавање брзине мотора, иницијализација комуникацијског порта, прорачунавање почетног угла и позивање функције за довођење главе за савијање жице у почетни положај. [4] [15] [16]

```
void setup() {
  //Inicijalizacija komunikacijskog porta
  Serial.begin (9600);

  //Definisanje funkcija pinova
  pinMode (homeButtonPin, INPUT);
  pinMode (benderPin, OUTPUT);
  digitalWrite (benderPin, LOW);

  //Postavljanje brzine motora
  zRotation.setSpeed(80);
  bender.setSpeed(100);
  extruder.setSpeed(120);

  //Kalkulacija nultog ugla
  calculateZero();

  //Pozicioniranje glave za savijanje u pocetni polozaj
  homing();
}
```

Као што је већ описано у тачки 3.3. следи фунција „loop()” која представља функцију која се непрестано понавља докле год је Ардуино укључен. У овој функцији врши се учитавање података из рачунара, након чега се покреће функција „motorrun()” која на основу учитаних података врши контролу свих мотора и соленоида. Функција „loop()” може се видети на следећој страни. [4]

```
void loop() {
  int copies = 0;
  //Ucitavanje podataga iz Procesinga
  while (Serial.available ()){
    int in = Serial.read();
    byte incoming = in+128; //Konvertovanje bajtova iz procesinga
    if (incoming != 255){   //255 oznacava kraj
      //Popunjavanje niza
      values[fieldindex] = values[fieldindex]*10+incoming;
      fieldindex++;
    }
    else{
     // Serial.println("END");
      for (int i=0; i<=fieldindex;i++){
     //   Serial.println(values[i]-128);
      }
      copies=copies+1;
    }
  }

  //Pokretanje savijanja zice
  if (copies==1){
    delay (1000);
    motorrun();
    feed(50); //eject
    copies=copies+1;
    homing();
    fieldindex=0;

    for(int i=0; i <= 1200;i++){
      values[i]=0;
      zbend(0);
    }
  }
}
```

На коду изнад може се видети може се видети да се прво проверава да ли рачунар шаље податке Ардуину и уколико долази до примања података они се смештају у претходно дефинисаном низу. Када престане примање података излази се из „while()” петље и покрећу се мотори преко функције „motorrun()”, Након њеног завршетка врши се екструзија жице за 50 mm, бришу вредности из низа и ротација главе се враћа у почетни положај. Затим Ардуино чека поновно примање података како би поновио целокупан процес. [4]

Функција која врши контролу мотора на основу примљениг података јесте „motorrun()” у којој се врши конверзија података добијених из рачунара и на основу њих покреће се одговарајући мотор са одговарајућим параметрима. Ова функција може се видети испод. [4]

```
void motorrun(){
  //int lastbend=0;
  for (int i=0; i<= fieldindex;i++){
    delay (20);
    //Trazenje markera koji ukazuje na motor za uvlacenje zice
    if ((values[i]-128)==126){
      //Uvlacenje zice za odredjenu duzinu
      feed (values[i+1]-128);
    }
    //Trazenje markera koji ukazuje na motor za za savijanje zice
    else if ((values[i]-128)==125){
      //Ugao savijanja
      int bendAng = (values[i+1]-128);
      //Smer savijanja
      if ((bendAng<0&&curDir==cw) || (bendAng>0 && curDir ==ccw)){
        duck ();
        //delay (100);
      }
      bend (bendAng);
     // lastbend = bendAng;
    }
    //Trazenje markera za rotaciju glave savijanja
    else if ((values[i]-128)==124){
      zbend (values[i+1]-128);  //Ugao rotacije
    }
  }
```

Као што се може приметити дефинисани су одређени маркери помоћу којих се одређује који мотор се треба активирати. Када се наиђе на тај маркер, активира се функција која покреће тај мотор за који је маркер намењен, при чему се као њен параметар уноси следећи податак из низа који представља дужину увлачења жице, односно угао заокретања.

Важно је споменути и неке од функција помоћу којих се врши контрола мотора. На примеру испод приказана је функција која врши контролу мотора за ротацију главе којом се врши савијање жице. Свака функција за покретање мотора приликом њеног позивања садржи један атрибут а то је угао за који је потребно ротирати одређене делове, односно дужину за коју је потребно увући жицу. У овом случају то јесте угао који се уноси у степенима за који је потребно савити жицу. Пошто команде из библиотеке „Stepper.h" захтевају да се као атрибут унесу кораци мотора за које је потребно извршити ротацију осовине мотора, то значи да је потребно извршити прерачун угла у кораке мотора потребне да се направи тај угао. За пример је узета функција за ротацију главе мотора јер је најсложенија. У овој функцији може се видети и полиномска једначина добивена експерименталним путем која врши корекцију угла за савијање. Разлог томе јесте што одређени делови машине нису израђени са високом прецизношћу и који директно утичу на крајњи резултат савијања. [15] [18]

```
//Funkcija za kontrolu motora za savijanje zice
void bend (float angle) {
  if (angle!=0){
    boolean dir=cw;
    boolean back=ccw;
    if (angle<0){
      dir = ccw;
      back = cw;
    }
    float rotations = 0;
    angle = abs(angle);

    /*Polinomska jednacina za korigovanje ugla
      dobivena eksperimentalnim merenjima*/
    angle=-1.513583083*pow(angle,8)/100000000000000
        + 1.004753281*pow(angle,7)/100000000000
        - 2.508854186*pow(angle,6)/1000000000
        + 2.962127864*pow(angle,5)/10000000
        - 1.634057824*pow(angle,4)/100000
        + 3.405019148*pow(angle,3)/10000
        - 5.189371596*pow(angle,2)/1000
        + 1.155772463*angle + 29.93500616;
    angle = calculateMain(angle);
    rotations = (inc*14933.3333 * angle)/360;
    rotations = zero - rotations - safetyDistance - correction;
    correction=0;
    delay(20);
    if(dir){
      bender.step(rotations);
      delay(20);
      bender.step(-rotations);
    }
    else{
      bender.step(-rotations);
      delay(20);
      bender.step(rotations);
    }
  }
}
```

Након прорачунавања корекције угла за који треба да се жица савије, врши се позивање функције „calculateMain(angle)” која врши прорачунавање угла за који треба глава мотора да се ротира. Разлог томе јесте дизајн контрукције машине због којег угао ротирања главе није исти као угао савијања жице.

Затим је потребно претворити угао ротирања, који износи у степенима, у моторске кораке и то је одрађено на следећи начин:

$$rotations = (inc * 14933.3333 * angle)/360;$$

У овој једначини број 14933.3333 представља број корака који су потребни за ротацију главе за савијање жице за пун круг. Он је добивен на тај начин што се однос зупчастог преноса помножи са бројем корака за који је потребно ротирати осовину мотора за пун круг:

$$\frac{70}{15} \cdot 3200 = 14933.3333$$

Константа „inc” уведена је због грешке приликом ротирања мотора и чија је вредност 4 у овом случају. Приликом тестирања библиотеке и мотора утврђено је да се осовина мотора ротира за четвртину круга када ова константа није укључена у прорачун. Претпоставка је да постоји грешка у Ардуиновој библиотеци приликом позивања функције за покретање мотора.

Приликом тестирања машине закључено је да може доћи заглављивања клина соленоида испод жице приликом промене смера савијања, па је било потребно решити и тај проблем. Решење је да се додатно изврши прорачун угла, у овом случају моторских корака где су додате безбедносне вредности како би се избегле могуће грешке приликом рада.

Функција која врши покретање мотора јесте „bender.step(rotations)” где „bender” представља мотор који је претходно дефинисан приликом покретања Ардуина. У заградама се уноси жељени број моторских корака за који је потребно ротирати осовину мотора. [15] [16]

Испод се налазе функције које врше контролу преосталих мотора. Принцип рада је исти као и у функцији за контролу мотора која врши ротацију главе за савијање жице, с'тим што су прорачуни потребног угла за ротирање односно дужине за увлачење жице врше на њихов одговарајући начин. Испод се такође налазе и функције које врше ротирање пина око жице приликом промене смера. [4] [15] [16]

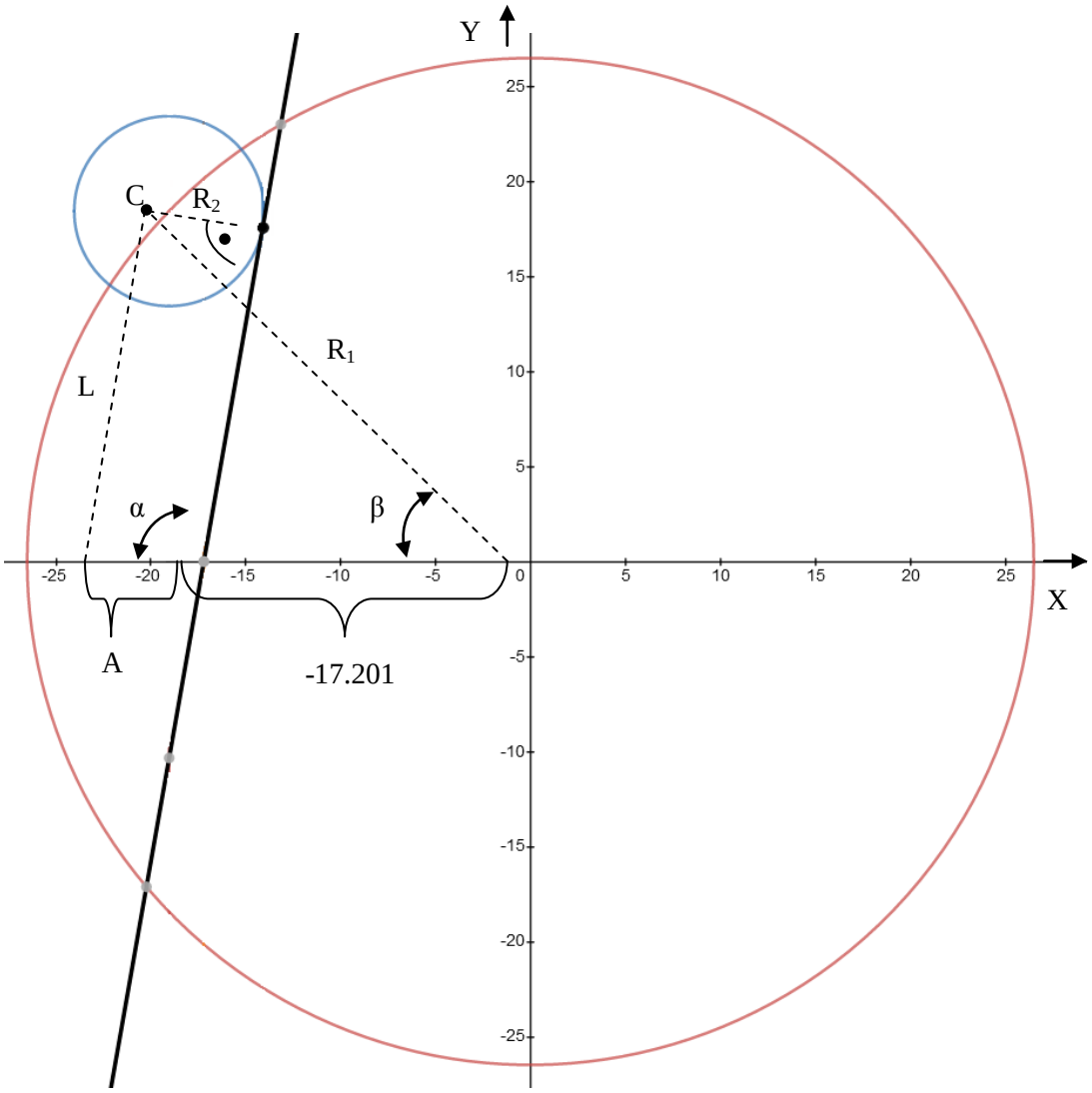
```
//Funkcija za rotiranje glave za savijanje zice
void zbend (float angle) {
    boolean dir=cw;
    if (curAngleZ < angle){
      dir = ccw;
    }
    float rotations = 0;
    rotations = curAngleZ - angle;
    curAngleZ = angle;
    rotations = abs(rotations);
    rotations = (inc*14933.3333 * rotations)/360;
    delay(20);
    if(!dir){
      zRotation.step(rotations);
    }
    else{
      zRotation.step(-rotations);
    }
}
//Funkcija pomocu koje se vrsi uvlacenje zice
void feed (float dist) {
  if (dist != 0){
    float rotations = 0;
    float feedCalib = 25.9*PI;
    dist = inc*3200 * dist/feedCalib;
    rotations = dist;
    extruder.step(-rotations);
  }
}
//Funkcija koja vrsi rotiranje pina prilikom promene smera
void rotatePin (boolean dir, float angle) {
  float rotations = 0;
  rotations = zero - safetyDistance;
    if(dir){
    bender.step(2*rotations);
  }
  else{
    bender.step(-2*rotations);
  }
  delay(100);
}
//Funkcija koja povlaci pin i poziva funkciju za rotiranje pina
void duck (){  //ducks bender pin under wire
  digitalWrite (benderPin, HIGH);
  delay (100);
  rotatePin (curDir, zero);
  digitalWrite (benderPin, LOW);  //pin down move under wire
  curDir = !curDir;    //direction reversed for next duck
  correction=600;
}
```

Пошто је претходно споменуто да је због конструкције машине потребно прерачунати угао за који треба да се ротира глава за савијање жице у односу на угао за који жица треба да се савије креиране су следеће функције: [4] [18]

```
//Funcija koja vrsi proveru unetog ugla i poziva funkciju za racunanje
float calculateMain(float angle){
  if(angle==90.0 || angle==0.0){
    angle+=increment;
  }
  float pAng=angle;
  do{
    if(isnan(angle)){
      angle = pAng + increment;
      pAng = angle;
    }
    angle = calc(angle);
  }while(isnan(angle));
  return angle;
}

//Funkcija za proracunavanje ugla
float calc(float angle){
  float ta = tan(PI*(180-angle)/180);
  float co = cos(PI*(angle-90)/180);
  float a = pow(ta,2)+1;
  float b = 2*(pow(ta,2)*offset+pow(ta,2)*r2/co);
  float c = pow(ta,2)*pow(offset,2)+2*pow(ta,2)
            *offset*r2/co+pow(ta,2)*pow(r2,2)
            /pow(co,2)-pow(r1,2);
  float x1 = (-b+sqrt(pow(b,2)-4*a*c))/2/a;
  float x2 = (-b-sqrt(pow(b,2)-4*a*c))/2/a;
  float y1 = ta*(x1+offset+r2/co);
  float y2 = ta*(x2+offset+r2/co);
  if(x1>0){
    angle = calcAngle(x2);
    if(y2<0){
      angle*=-1;
    }
  }else if(y2>0){
    angle = calcAngle(x2);
  }else{
    angle = calcAngle(x1);
  }
  return angle;
}

//Pomocna funkcija za proracunavanje ugla
float calcAngle(float x){
  float ang = acos(abs(x)/r1)*180/PI;
  return ang;
}

//Funkcija za proracunavanje nule
void calculateZero(){
  zero=calculateMain(4.4);
  zero = (inc*14933.3333 * zero)/360; //converts angle to motor steps
}
```

Као што се може видети на претходној страни прва функција за рачунање угла јесте „calculateMain(float angle)” која пре свега има задатак да провери да ли је унети угао једнак 90° или 0° и уколико јесте онда се додаје инкремент који у овом случају износи 0.01. Разлог томе јесте што се у наредној функцији која се зове „calc(float angle)” користи тангесна фунција приликом рачунања. Да би се избегла грешка, односно немогућност рачунања, додаје се вредност инкремента која не утиче на резултат савијања.

Након тога позива се функција „calc(float angle)” која има задатак да прерачуна жељени угао савијања жице у угао за који је потребно ротирати главу за савијање жице. У тој функцији може се приметити сложена једначина за рачунање угла. Да би се разумео проблем и како је дошло до решења потребно је погледати слику 15. где је нацртан геометријски проблем.




**Слика 15.** *Геометријски проблем за решавање једначине*

Координатни почетак јесте положај осовине око које се окреће клин којим се врши кривљење жице. Црвеном кружницом је означена путања низ коју се креће центар клина, док плава кружница представља сам клин. Црном линијом означена је жица која се савија, а растојање од тачке у којој се врши њено савијање и центра координатног система измерена је софтверски приликом конструисања машине. Растојање означено на слици словом „A" може да се израчуна помоћу полупречника клина на следећи начин:

$$\mathrm{A} = \frac{R_2}{\cos(\alpha - 90)}$$

На основу фотографије можемо поставити следећи систем једначина:

1) $x_C^2 + y_C^2 = R_1^2$

2) $y_c = \tan(180 - \alpha) \cdot (x_C + 17.201 + A)$

Након решавања система једначина добија се квадратна једначина са коефицијентима:

$$ax_C^2 + bx_C + c = 0$$

$$a = \tan(180 - \alpha)^2 + 1$$

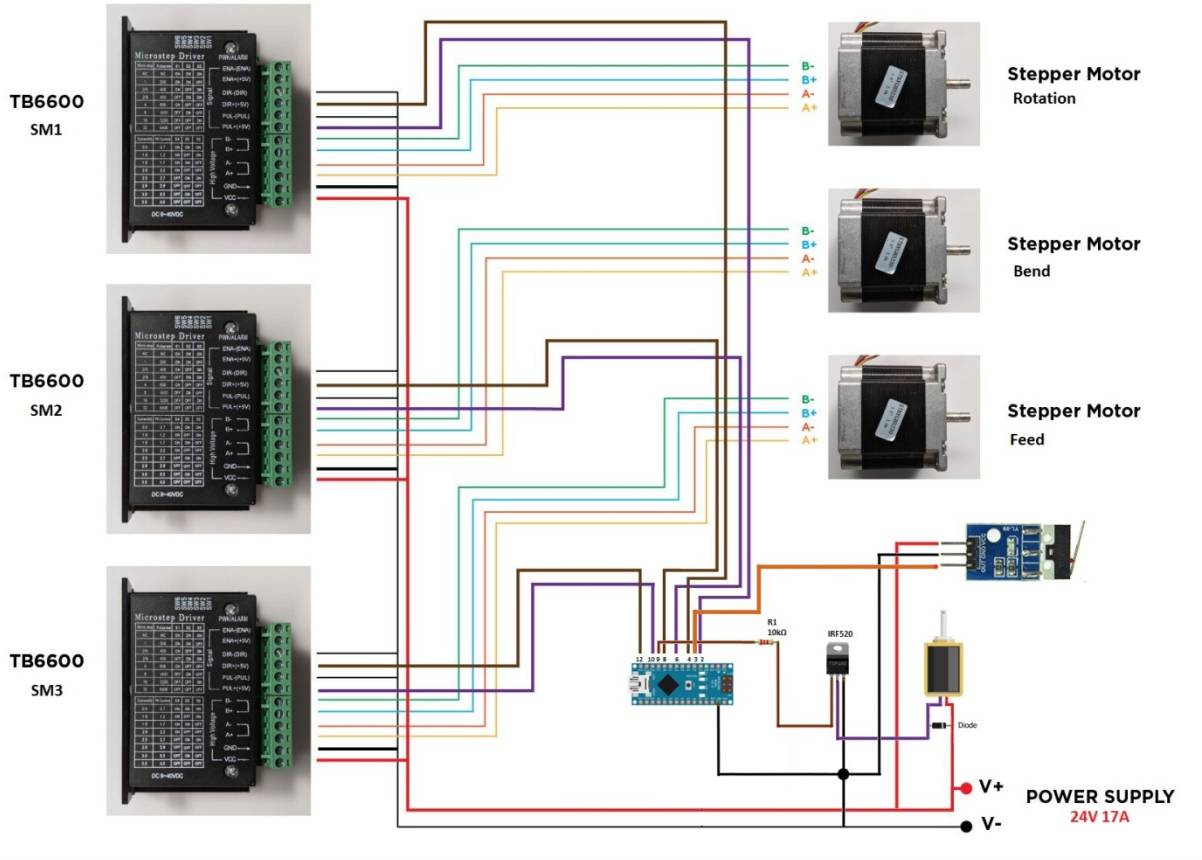
$$b = 2 \cdot \left(\tan(180 - \alpha)^2 \cdot 17.201 + \frac{\tan(180 - \alpha)^2 \cdot R_2}{\cos(\alpha - 90)}\right)$$

$$c = \tan(180 - \alpha)^2 \cdot 17.201^2 + 2 \cdot \frac{\tan(180 - \alpha)^2 \cdot 17.201 \cdot R_2}{\cos(\alpha - 90)} + \frac{\tan(180 - \alpha)^2 \cdot R_2^2}{\cos(\alpha - 90)^2} - R_1^2$$

Као што се у коду може видети након израчунавања координата средишта клина врши се њихов одабир. Пошто квадратна једначина има два решења, а у овом случају само једно може да буде право, врши се провера на основу да ли су координате мање или веће од нуле. Затим се на основу координате $x_C$ врши прорачун угла β који уствари јесте угао за који је потребно ротирати главу за савијање жице.

### 3.5. Шема повезивања свих компоненти

Начин на који је извршено повезивање свих компоненти може се видети на слици 16.




**Слика 16.** *Шема повезивања свих компоненти*

| Ардуино PIN | Повезана компонента |
|---|---|
| 2 | Пулс за мотор ротације |
| 3 | Гранични прекидач |
| 4 | Правац за мотор ротације |
| 6 | Пулс за мотор савијања |
| 8 | Правац за мотор савијања |
| 9 | Соленоид |
| 10 | Пулс за мотор увлачења жице |
| 12 | Правац за мотор увлачења жице |

**Табела 3.** *Начин повезивања компоненти за Ардуином*

# 4. Софтвер за генерисање кода на основу CAD дизајна

Сада када је машина направљена, потребно је креирати софтвер који ће моћи да на основу учитаних CAD дизајнова, изврши прорачун свих углова и дужина. Овај софтвер одрађен је у објектно оријентисаном окружену које се зове процесинг (енг. *Processing*). Процесинг је графичка библиотека отвореног кода и интегрисано развојно окружење (енг. *IDE*) изграђено за заједнице електронских уметности, уметности нових медија и визуелног дизајна са циљем да не-програмере подучи основама рачунарског програмирања у визуелном контексту. Обрада користи језик Јава, уз додатна поједностављења као што су додатне класе, математичке функције и операције. Такође пружа графички кориснички интерфејс за поједностављивање фазе компајлирања и извршавања. Слично као и код Ардуина, корисници морају да дефинишу само две функције како би направили извршни програм. [14]

Те функције су:

- setup() – функција која се извршава једном на почетку и служи за почетна подешавања;
- draw() – функција која се извршава у петљи све време док се програм не искључи.

## 4.1. Креирање софтвера

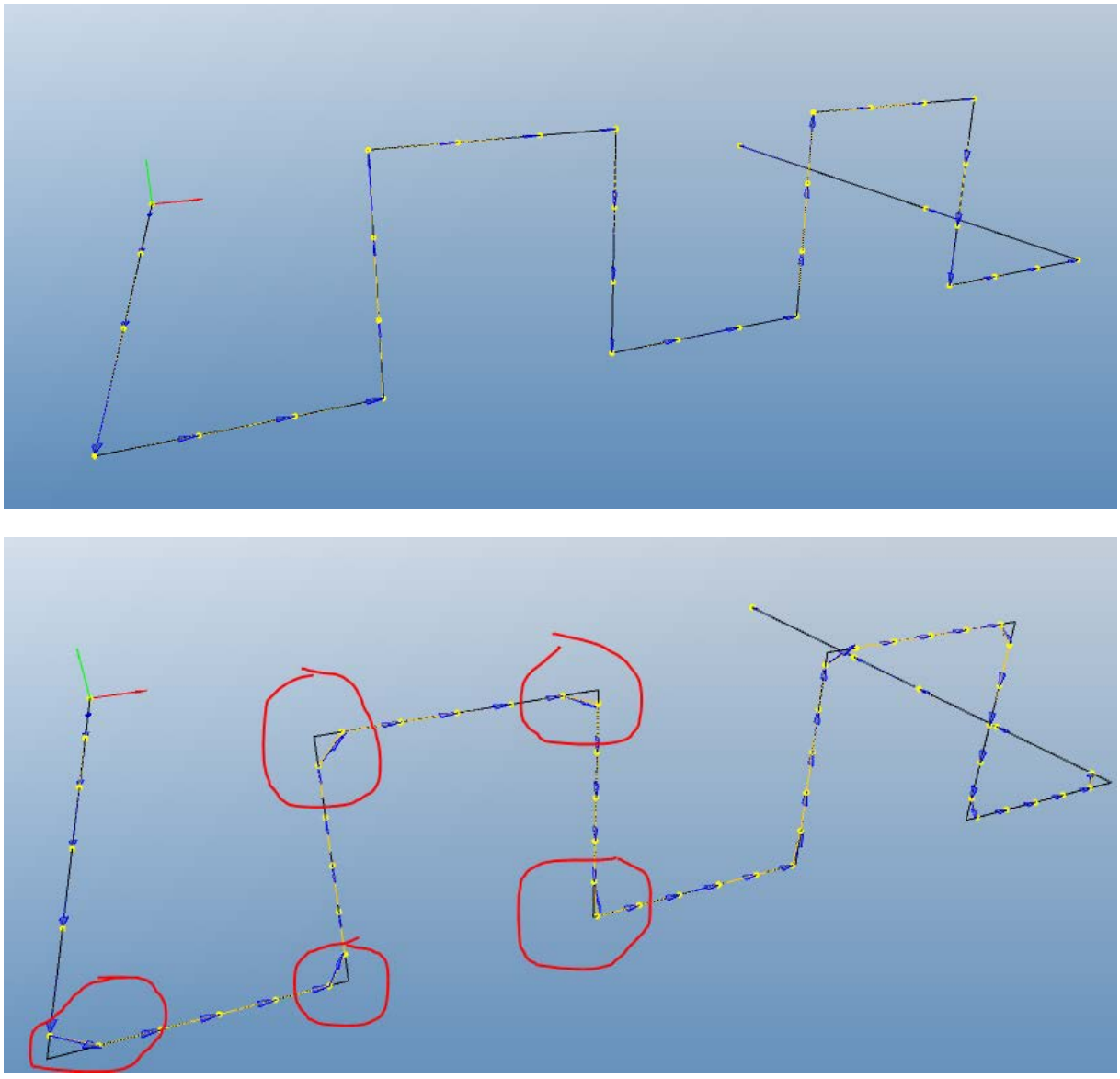
Пре почетка програмирања, потребно је преузети неке додатне библиотеке које се не налазе у склопу оригинале инсталације Процесинга. Прва библиотека коју је потребно преузети са интернета јесте „controlP5” која служи за графички интерфејс. Ова библиотека садржи објекте као што су клизачи, дугмад, преклопнике, текстуална поља, радио дугмад, поља за потврду и могу се лако имплементирати у Процесинг скицу. Друга библиотека коју је потребно преузети јесте „iGeo” и она је бесплатна библиотека за 3Д моделирање са отвореним кодом заснована на Јави за рачунски дизајн у архитектури, дизајн производа, дизајн интеракције и још много тога. Ова библиотека је дизајнирана да пружи функционално 3Д моделирање засновано на коду, практичност употребе у пракси рачунарског дизајна, проширивост за сарадњу са другим компонентама за обраду података и интерфејсе за интеракцију улаза / излаза у реалном времену на друге различите медије у Јави. [11][12]

Софтвер је креиран из два дела од којих је први главни програм и представља прозор у коме се приказује 3Д окружење са учитаним дизајном жице. У овом делу се налази и функција која врши прорачун свих потребних углова и дужина на основу задатих критеријума. Други део софтвера представља помоћни прозор у ком се налазе сви тастери, слајдери и поље за испис текста.

Главни програм налази се у коду испод: [5] [12]

```
import controlP5.*;
import processing.opengl.*;
import igeo.*;
import java.lang.Object;

menuWindow meni;
IVec x = new IVec(10,0,0);
IVec y = new IVec(0,10,0);
IVec z = new IVec(0,0,10);

int numpts = 0;
float oldResolution = 1.0;
float [][] podaci;
float slideMin=0, slideMax=0;
String fileName;
boolean isFileSelected=false;
ICurve [] crvs;
IVec[] cpts;
IPoint[] pointArray;
IVec[] vectors;
IVec calcVec1, calcVec2;

void settings() {
  size(1000, 667, IG.GL);
}
void setup() {
  meni = new menuWindow();
}
void draw() {
  if (isFileSelected && oldResolution!=meni.resolutionSlide) {
    oldResolution = meni.resolutionSlide;
    IG.clear();
    new ICurve(crvs[0]);
    cpts = new IVec[(int)crvs[0].len()];
    numpts = (int) map(meni.resolutionSlide/*-0.3*/, slideMin,
    slideMax, 1, (float) crvs[0].len());
    cpts[0] = new IVec(crvs[0].start());
    cpts[numpts-1] = new IVec(crvs[0].end());
    for (int i=1; i<numpts-1; i++) {
      float n = float (numpts);
      float perc = i/n;
      cpts[i] = new IVec (crvs[0].pt(perc));
    }
    pointArray = new IPoint[numpts];
    vectors = new IVec[numpts-1];
    for (int i=0; i<numpts; i++) {
      pointArray[i] = new IPoint(cpts[i]).clr(255, 255, 0);
      if(i>0){
        new ICurve(pointArray[i-1],pointArray[i]).clr(1,.8,0);
        vectors[i-1] = new IVec(round((float)(pointArray[i].x() -
        pointArray[i-1].x())),round((float)(pointArray[i].y() -
        pointArray[i-1].y())),round((float)(pointArray[i].z() -
        pointArray[i-1].z())));
        vectors[i-1].show(pointArray[i-1]).clr(0,0,1.);
      }}}
x.show().clr(0,0,1.); y.show().clr(1.,0,0); z.show().clr(0,1.,0);
}
```

Први корак јесте да се изврши увоз свих библиотека потребних за беспрекорно функционисање софтвера након чега се врши декларација и иницијализација потребних променљивих. У функцији „void settings()” врши се подешавање резолуције главног прозора, након чека долази функција „void setup()” и којој се врши иницијализација помоћног прозора, са свим командама. Функција „void draw()” јесте главни програм који се извршава тек након учитавања фајла или промене вредности слајдера за резолуцију уколико је фајл учитан. Након његовог учитавања врши се његов приказ и на основу подешене резолуције, исцртавају се жуте тачке дуж учитане линије. Након тога креирају се вектори узмеђу креираних тачака помоћу којих се лако може израчунати угао за који је потребно да се жица савије али и дужина за коју је потребно увући жицу. Резолуција у ствари представља колики ће број тачака бити креиран дуж линија, односно колико ће вектора бити креирано. У овом случају не мора да значи правило да ће креирани вектори бити што прецизнији жељеном облику уколико је резолуција већа, већ то искључиво зависи од облика у који је потребно савити жицу. Да би се боље разумела резолуција потребно је видети слику 17.



**Слика 17.** *Поређење различите резолуције*

У овом делу програма налази се и функција за прорачунавање углова за савијање и прорачунавање дужина за које се жица увлачи. Прорачуни се врше на тај начин што се први вектор ротира тако да се поклопи са осом „y”. Након тога се сви остали вектори ротирају за исти угао за који је ротиран тај први вектор и рачуна се угао између првог вектора и другог вектора где се добивене вредности уписују у низ са подацима. Затим се други вектор ротира у положај „y” након чега се целокупан процес понавља. [5] [11]

```
//Funkcija za racunanje uglova
void calcs() {

  meni.infoArea.clear();
  float proba1,proba2;
  meni.infoArea.append("Broj tačaka je: "+pointArray.length+"\n\n");
  podaci = new float [vectors.length][3];
  podaci[0][0]=(float)vectors[0].len();
  podaci[0][1]= 0;
  podaci[0][2]= 0;
    for(int i=0;i<vectors.length-1 ;i++){
      double angle = round((float)vectors[i].angleOnPlane(x,z)*180/PI);
      vectors[i].rot(z,angle*PI/180);
      double angle2 =
      round((float)vectors[i].angleOnPlane(x,y)*180/PI);
      vectors[i].rot(y,angle2*PI/180);
      vectors[i].set(round((float)vectors[i].getX().x),
      round((float)vectors[i].getY().x),
      round((float)vectors[i].getZ().x));
      for(int j=i+1;j<vectors.length;j++){
        vectors[j].rot(z,angle*PI/180);
        vectors[j].rot(y,angle2*PI/180);
        vectors[j].set(round((float)vectors[j].getX().x),
        round((float)vectors[j].getY().x),
        round((float)vectors[j].getZ().x));
      }
    podaci[i+1][0] = round((float)vectors[i+1].len());
    podaci[i+1][1] = round((float)vectors[i].angle(vectors[i+1],
    new IVec(0,0,1))*180/PI);
    podaci[i+1][2] = round((float)y.angleOnPlane(vectors[i+1],
    new IVec(1,0,0))*180/PI);
    if(podaci[i+1][2]>90 && podaci[i+1][2]<=180){
      podaci[i+1][2] = podaci[i+1][2] - 180;
    }
    else if(podaci[i+1][2]<-90 && podaci[i+1][2]>=-180){
      podaci[i+1][2] = podaci[i+1][2] + 180;
    }
 }
  int k=0, a=0;;
  float[][] proba = new float[podaci.length][3];
  proba[0]=podaci[0];
```

```
for(int i=1;i<podaci.length;i++){
    if(podaci[i][1]==0){
      proba[k][0]=proba[k][0]+podaci[i][0];
    }
    else{
      proba[k][2] = podaci[a][2];
      a=i;
      k++;
      proba[k]=podaci[i];
    }
  }
  podaci = new float [k+1][3];
  for(int i=0;i<=k;i++){
    podaci[i][0]=round(proba[i][0]);
    podaci[i][1]=round(proba[i][1]);
    podaci[i][2]=round(proba[i][2]);
  }
  for(int i=0;i<podaci.length;i++){
    meni.infoArea.append(podaci[i][0]+"\t\t\t\t\t\t"+podaci[i][1]+"\t\t
\t\t\t\t"+podaci[i][2]+"\n");
  }
}
```

Остале функције у главном програму могу се видети у наставку: [12]

```
//Funkcija koja se izvrsava centriranje prikaza oblika
void center() {
  IG.focus();
}

//Funkcija koja se izvrsava ucitavanje fajlova
void openShape(String link) {
  IG.clear();
  IG.open(link);
}

//Funkcija koja se izvrsava prilikom klika na dugme za ucitavanje fajlo
va
void openDrawing() {
  selectInput("Select a file to process:", "fileSelected");
}

//Funkcija koja menja stanje da li je fajl ucitan
void fileSelectedStateCh(Boolean state) {
  isFileSelected=state;
}
```

```
/*Funkcija koja proverava da li je selektovan fajl, njegovu ekstenziju
i pokušava njegovo učitavanje
 Ova funkcija se poziva prilikom izvršavanja funkcije iznad (klikom na
dugme za učitavanje oblika)
 */
public void fileSelected(File selection) {
  if (selection == null) {
    fileSelectedStateCh(false);
    return;
  } else {
    fileName=selection.getAbsolutePath();
    fileName=fileName.replace('\\', '/');
    if (!(fileName.substring(fileName.length()-4).equals(".3dm") ||
    fileName.substring(fileName.length()-4).equals(".3DM") ||
    fileName.substring(fileName.length()-4).equals(".obj") ||
    fileName.substring(fileName.length()-4).equals(".OBJ"))) {
      showMessageDialog(null, "Potrebno je selektovati 3dm fajl",
"Alert", ERROR_MESSAGE);
      return;
    }
  }
  try {
    openShape(fileName);
    center();
    crvs = IG.curves();
    meni.resolutionSlide = 5.0;
    if (crvs.length>1){
      IG.clear();
      showMessageDialog(null, "Ucitan fajl sadrži vise od jedne linije
koje nisu spojene", "Alert", ERROR_MESSAGE);
      return;
    }
    fileSelectedStateCh(true);
  }
  catch (Exception e) {
    showMessageDialog(null, "Greska prilikom ucitavanja fajla",
"Alert", ERROR_MESSAGE);
  }
}
```

Треба напоменути да се у функцији изнад, која служи за учитавање CAD фајла врши провера одговарајуће екстензије и уколико екстензија није тачна, софтвер ће избацити поруку да је дошло до учитавања погрешне екстензије. Исто тако уколико фајл није одрађен како треба, тј. фајл садржи више линија које нису међусобно спојене, софтвер ће такође избацити грешку да има више линија и да није могуће учитавање фајла. Као што се може видети у коду изнад екстензије које су дозвољене, односно које библиотека „iGeo” подржава јесу „.3dm” и „.obj”. [1] [5] [12]

Испод се може видети код за креирање помоћног прозора са тастерима који јесте класа која наслеђује „ PApplet” класу. У овој делу кода врши се и декларација комуникацијског порта са Ардуином. Испод се такође могу видети примери креирања једне падајуће листе и једног тастера. [5] [11] [14]

```
import static javax.swing.JOptionPane.*;
import processing.serial.*;

PImage[] imgs = new PImage[9];

class menuWindow extends PApplet {

  public menuWindow() {
    super();
    PApplet.runSketch(new String[]{this.getClass().getName()}, this);
  }

  ControlP5 cp5;

  DropdownList portsList;
  Textarea infoArea;

  Serial arduino;

  boolean isCalcAvailable=false;
  boolean isArduinoConnected=false;
  String selectedPort;
  float resolutionSlide = 1.0;

  public void settings() {
    size(220, 800);
    smooth();
  }
  public void setup() {
    surface.setTitle("Child sketch");
    cp5 = new ControlP5(this);
    loadImages();

    background(230);

    //Kreiranje padajuceg menija za listanje dostupnih portova
    portsList = cp5.addDropdownList("portsList").setPosition(10, 35);
    portsList.setBackgroundColor(color(190));
    portsList.setItemHeight(20);
    portsList.setBarHeight(15);
    portsList.getCaptionLabel().set("Ports:");
    portsList.setColorBackground(color(60));
    portsList.setColorActive(color(255, 128));
    portsList.setItemHeight(30);
    portsList.close();

    //Kreiranje tastera za ucitavanje prikljucenih portova
    cp5.addButton("refreshPorts")
      .setPosition(120, cp5.getController("portsList").getPosition()[1]-8)
      .setImages(imgs[0], imgs[0], imgs[1])
      .updateSize();
```

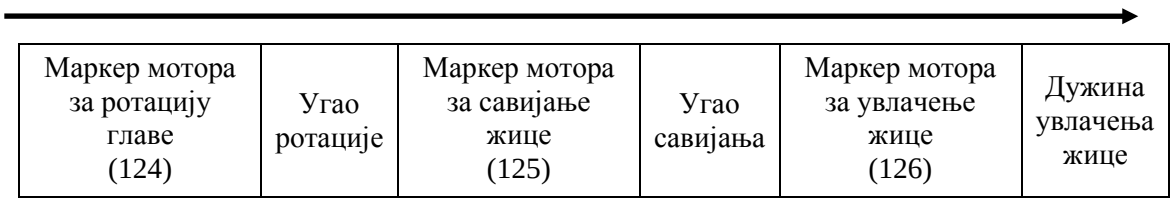
```
  //Funkcija koja se izvrsava prilikom klika na dugme za
povezivanje/diskonektovanje Arduina
  void connectDisconnect(){
    connectToPort();
    infoArea.clear();
    if(isArduinoConnected){
      //println("Arduino je konektovan");
      infoArea.append("Arduino je konektovan");
      cp5.getController("refreshPorts").lock();
      cp5.getController("portsList").lock();
      cp5.getController("connectDisconnect").setImage(imgs[2]);
    }
    else{
      //println("Arduino nije konektovan");
      infoArea.append("Arduino nije konektovan");
      cp5.getController("connectDisconnect").setImage(imgs[3]);
      cp5.getController("refreshPorts").unlock();
      cp5.getController("portsList").unlock();
    }
  }

//Funkcija koja vrsi konektovanje/diskonektovanje Arduina
void connectToPort(){
  if(isArduinoConnected){
    arduino.clear();
    arduino.stop();
    isArduinoConnected=false;
    return;
  }
  if(portsList.getItems().size()<1){
    return;
  }
  selectedPort = portsList.getItem((int)portsList.getValue())
  .values().toArray()[2].toString();
 try {
   //Podesavanje arudino porta
   arduino = new Serial(this, selectedPort , 9600);
   isArduinoConnected=true;
  } catch (Exception e) {
    isArduinoConnected=false;
    showMessageDialog(null, "Greska prilikom povezivanja Arduina",
    "Alert", ERROR_MESSAGE);
  }
}
```

Једна од битних функција коју је потребно издвојити из помоћног прозора јесте функција за повезивање Ардуина са овим софтвером. Тачније речено постоје две функције, прва се извршава приликом клика на тастер за повезивање и у којој се након повезивања одређени тастери закључавају, и друга која се позива унутар прве и која врши повезивање Ардуина са софтвером. Приликом креирања серијског порта, веома је битно водити рачуна да се подеси иста брзина серијске комуникације као што је подешена у програмирању микропроцесорске јединице и која у овом случају износи 9600 bps. [4] [5] [14]

Такође је битно споменути и функцију која врши слање података ка Ардуину. Функција је релативно проста и врши слање свих података из матрице у виду низа, на тај начин што се прво шаљу подаци за мотор који врши ротирање главе, затим се шаљу подаци за мотор који контролише главу за савијање и након тога подаци за мотор који врши увлачење жице. [4] [5] [14]

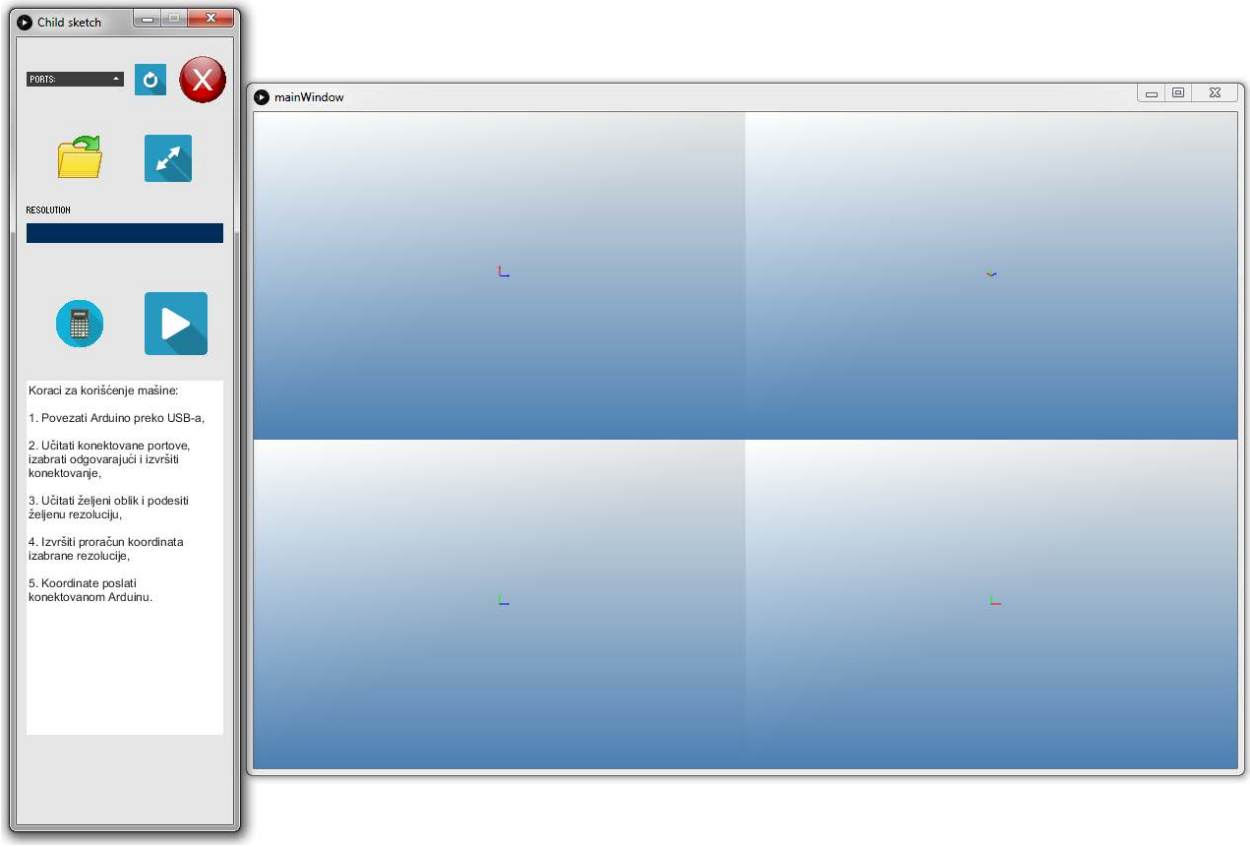
```
//Funkcija za slanje podataka Arduinu
void sendArd () {

 // println("sending...");

      for (int i = 0; i < podaci.length; i++) {
      byte zbend = 124;
      //Slanje markera za motor za rotaciju glave
      arduino.write (zbend);
      delay (100);
       //Slanje ugla za rotaciju glave
      arduino.write (byte (int(podaci[i][2])));
      byte xbend = 125;
      delay (100);
      //Slanje markera za motor za savijanje zice
      arduino.write (xbend);
      delay (100);
      //Slanje ugla za savijanje zice
      arduino.write (byte (int(podaci[i][1])));
      byte feedmotor = 126;
      delay (100);
      //Slanje markera za motor za izvlacenje zice
      arduino.write (feedmotor);
      delay (100);
      //Slanje duzine izvlacenja zice
      arduino.write (byte (int(podaci[i][0])));
    }
  delay (5000);
  byte end = 127;
  println("commands sent to printer");
  arduino.write(end);
}
```

Илустративни начин на који се шаљу подаци може се видети на слици 18.




**Слика 18.** *Начин слања података Ардуину*

На слици 19. може се видети коначан изглед софтвера.




**Слика 19.** *Креиран софтвер за генерисање потребних података*

У главном прозору могу се видети четири поља, од којих три поља представљају равни у координатном систему, док се у четвртом прозору врши 3D приказ учитаног модела. Са леве стране може се видети помоћни прозор у коме се налазе све потребне команде за коришћење софтвера. У овом прозору могу се пронаћи следећи тастери:

| | |
|---|---|
|  | Тастер за освежавање листе са портовима |
|  | Тастер за конектовање/дисконектовање софтвера са Ардуином |
|  | Тастер за учитавање фајла |
|  | Тастер за центрирање приказа модела на екрану |
|  | Тастер за прорачунавање података који се шаљу Ардуину |
|  | Тастер за слање података Ардуину и покретање процеса савијања жице |

**Табела 4.** *Тастери у софтверу за генерисање кода*

Поступак коришћења софтвера своди се на следеће кораке:

- Након конектовања Ардуина са рачунаром потребно је извршити освежавање листе са портовима како би се појавио порт на ком је Ардуино конектован;
- Након одабира одговарајућег порта потребно је извршити повезивање са Ардуином, након чега ће уколико повезивање буде успешно бити исписана порука у белом пољу да је повезивање успешно одрађено;
- Затим се врши учитавање жељеног дизајна и врши подешавање резолуције на слајдеру, где она означава у којим ће се тачкама вршити савијање;
- Када је оператер задовољан подешеном резолуцијом, потребно је да кликне на тастер за прорачунавање свих потребних параметара након чега ће они бити исписани у белом пољу;
- Последњи корак јесте да се изврши слање података ка Ардуину где ће се савијање жице аутоматски стартовати.

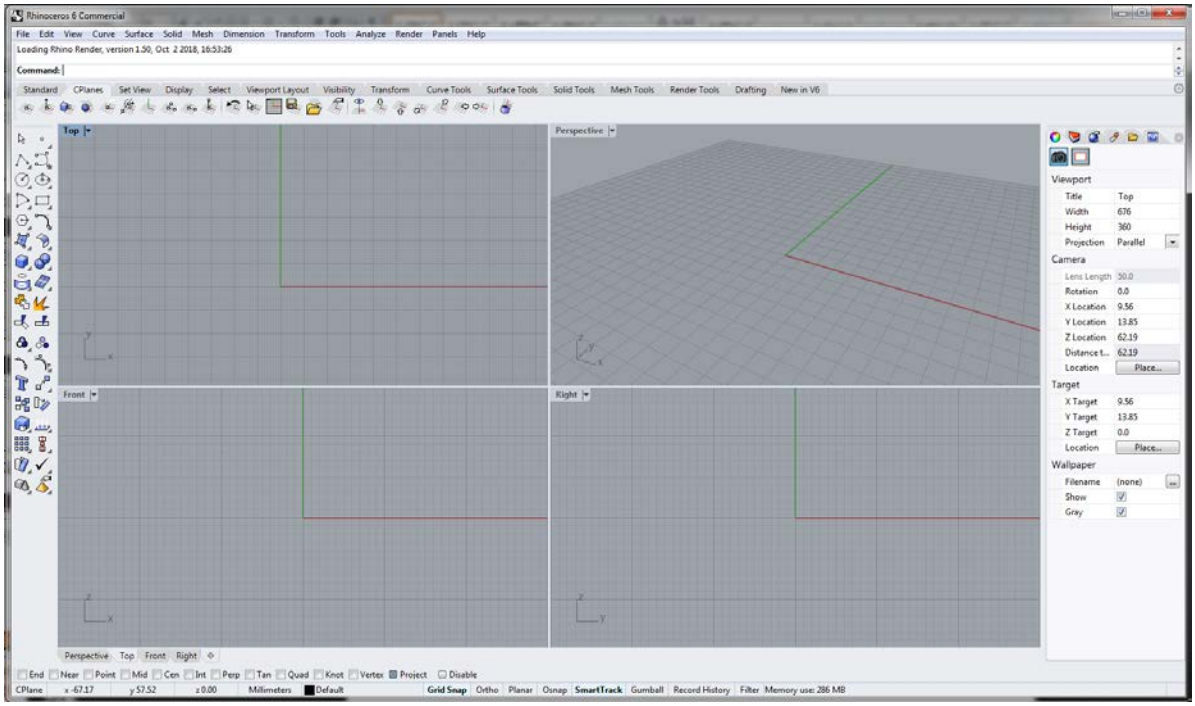
# 5. CAD дизајн жичаних форми

Дизајнирање жичаних форми обављено је у софтверу „Rhinoceros” који је намењен за 3Д рачунарску графику и рачунарски потпомогнути дизајн (CAD). Геометрија овог софтвера базирана је на математичком моделу „NURBS”, који се фокусира на стварање математички прецизног приказа кривих и површина слободног облика у рачунарској графици (за разлику од апликација заснованих на полигонским мрежама). Овај софтвер се користи у процесима рачунарски потпомогнутог дизајна (CAD), рачунарски потпомогнуте производње (CAM), брзог израде прототипа, 3Д штампе и обрнутог инжењеринга у индустријама, укључујући архитектуру, индустријски дизајн (нпр. Аутомобилски дизајн, дизајн пловила), дизајн производа (нпр. дизајн накита), као и за мултимедију и графички дизајн. [19]

Овај софтвер може се преузети са следећег линка:

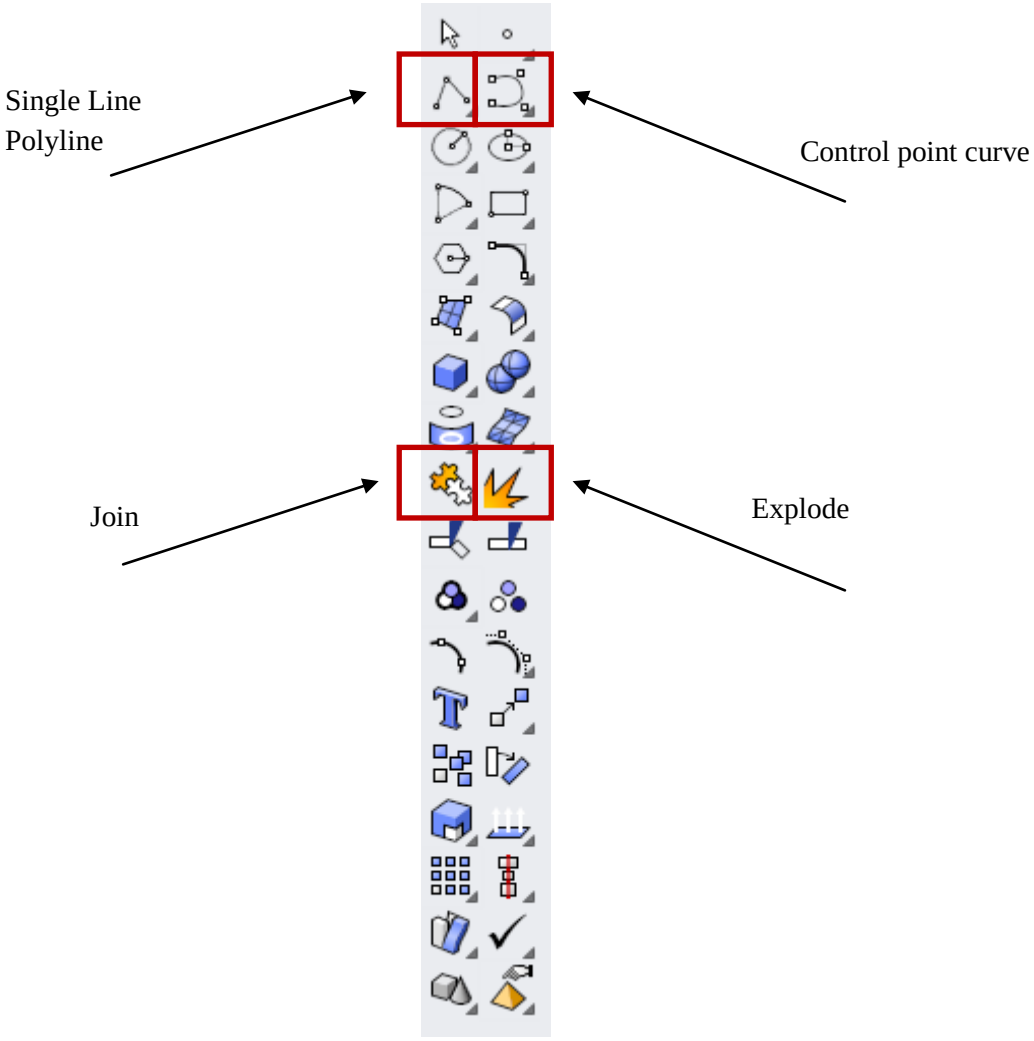
https://www.rhino3d.com/download

Жичане форме одрађују се у виду линија, односно полилинија. Потребно је напоменути да се треба водити рачуна да се приликом дизајнирања крајеви свих линија споје и да се на тај начин добије само једна линија која има почетак и крај. На слици 20. може се видети радно окружење овог софтвера.




**Слика 20.** *Радно окружење софтвера „Rhinoceros”*

Може се приметити да су у радном окружењу софтвера приказана четири прозора баш као и код креираног софтвера за генерисање кода. Са леве стране може се видети мени са командама које ће бити коришћене приликом дизајнирања облика.

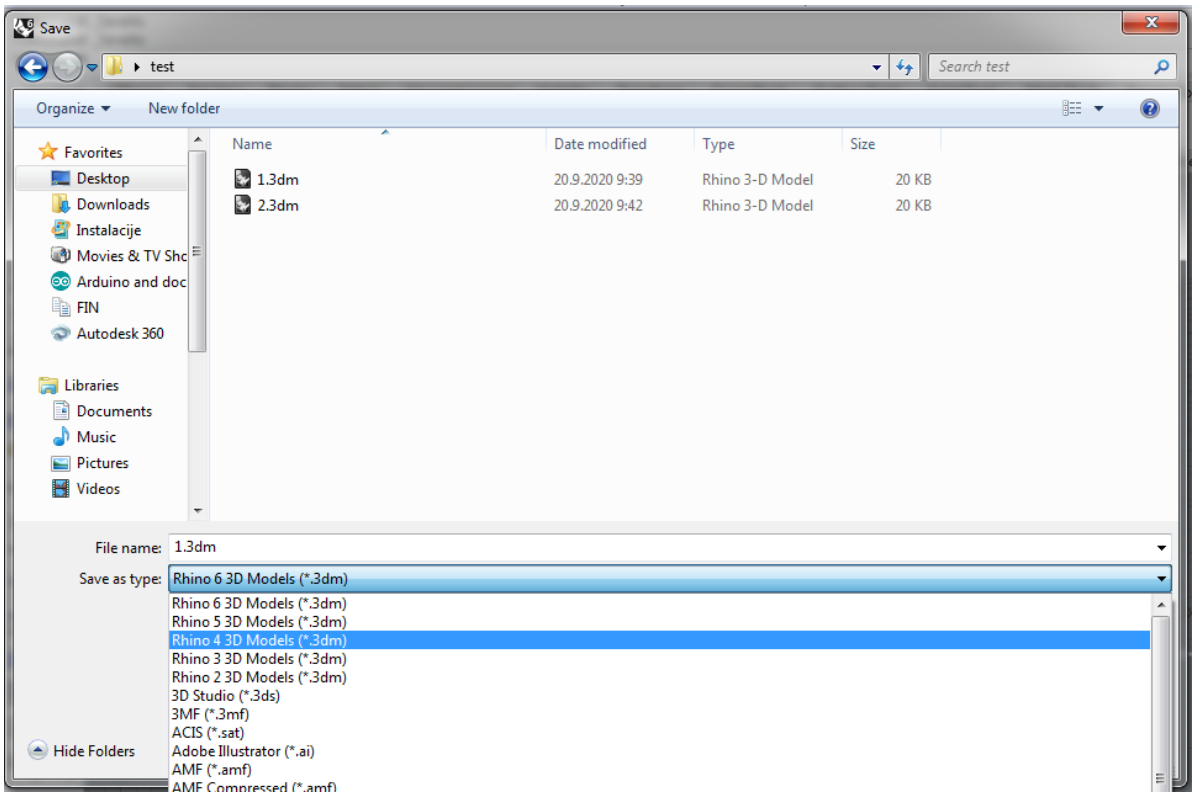



**Слика 21.** *Коришћене команде приликом израде жичаних форми*

Команда „Single Line” као што и сам назив каже извршава цртање једне линије. Приликом цртања следеће линије потребно је кликнути на једном од крајева претходно нацртане линије и затим нацртати следећу. У овом случају потребно је водити рачуна јер линије неће бити спојене иако им се крајеви налазе у истој тачки. Тада је потребно селектовати обе линије и кликнути на команду „Join” која ће те линије спојити у тачки у којој им се крајеви поклапају и тако добити полилинију. Безбеднији начин цртања јесте коришћење команде „Polyline” где се линије приликом њиховог цртања одмах спајају и на тај начин се избегава могућност грешке. Такође је могуће након цртања полилиније извршити њено разбијање на више линија помоћу команде „Explode”. Команда „Control point curve” користи се за цртање кривих линија.

Након моделирања дизајн је потребно сачувати у фајлове чија је екстензија „.3dm” или „.obj” из разлога што софтвер за генерисање кода користи библиотеку „iGeo”, која једино може отварати ове две врсте фајлова. [12]

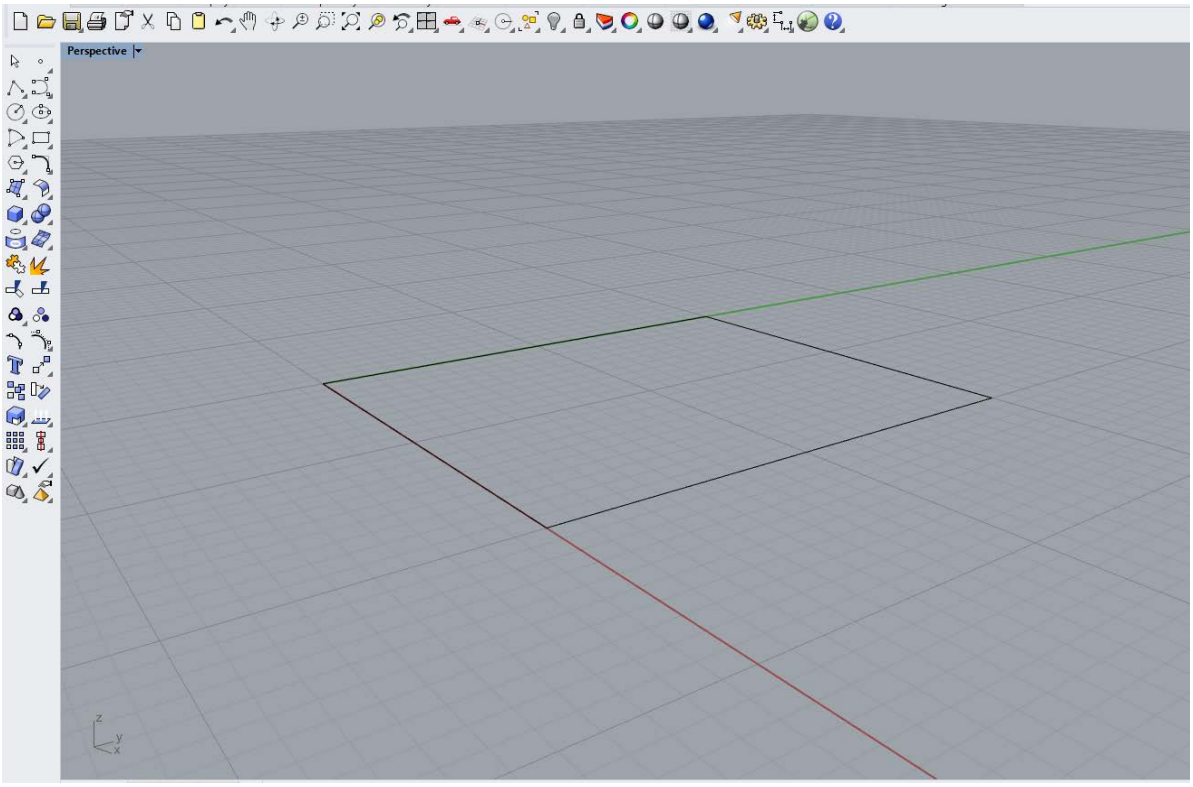
Датотека „.3dm” је формат 3Д модела отвореног кода и изворни формат софтвера „Rhinoceros”. Садржи 3D модел који укључује информације о површини, тачкама и кривинама. Ова датотека омогућава CAD-у, CAM-у, CAE-у и софтверу за рачунарску графику да тачно чувају и размењују 3D геометрију користећи репрезентације „NURBS” и полигон мреже. Потребно је напоменути да је приликом чувања ове датотеке, потребно сачувати као верзију 4, из разлога што „iGeo” библиотека не подржава новију верзију ове датотеке. [20]




**Слика 22.** *Чување датотеке „.3dm” као верзију 4*

Датотека „.obj” је стандардни формат 3Д слике који се може извести и отворити помоћу различитих програма за уређивање 3Д слика. Садржи тродимензионални објекат, који укључује 3Д координате, мапе текстура, полигонална лица и друге информације о објекту. Ова датотека се сматра универзалним форматом 3Д модела, јер га широко подржавају апликације за 3Д уређивање слика. Формат је једноставан и заснован на тексту, што је један од разлога зашто га многи програми користе. Међутим, једноставна структура формата такође може довести до огромних величина „.obj” датотека ако чувају велике и сложене 3Д објекте. [21]

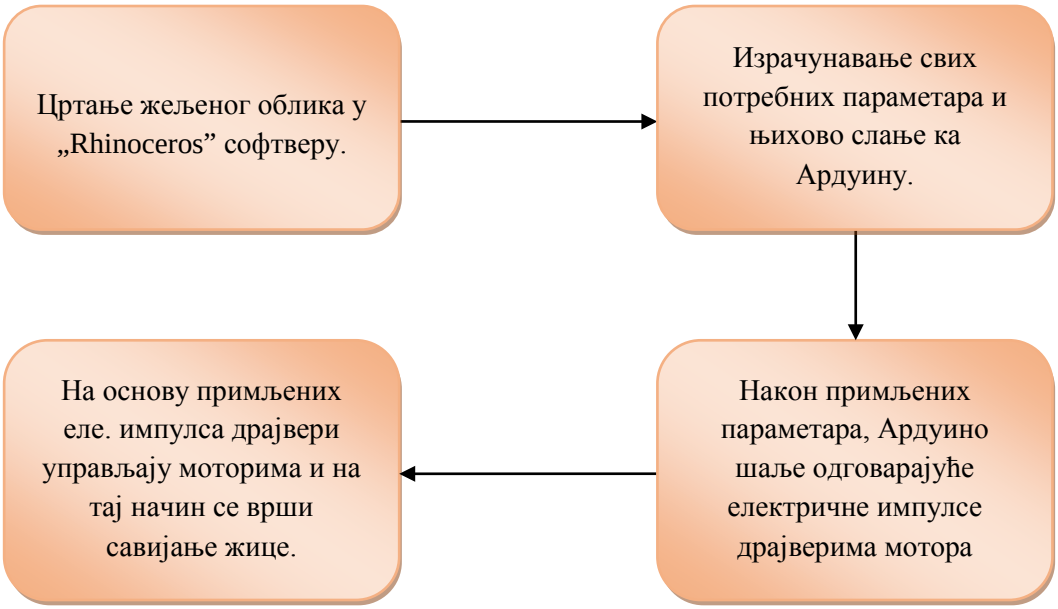
Пример изгледа квадрата који је нацртан уз помоћ команде за цртање полилинија може се видети испод.




**Слика 23.** *Пример изгледа квадрата у софтеру „Rhinoceros”*

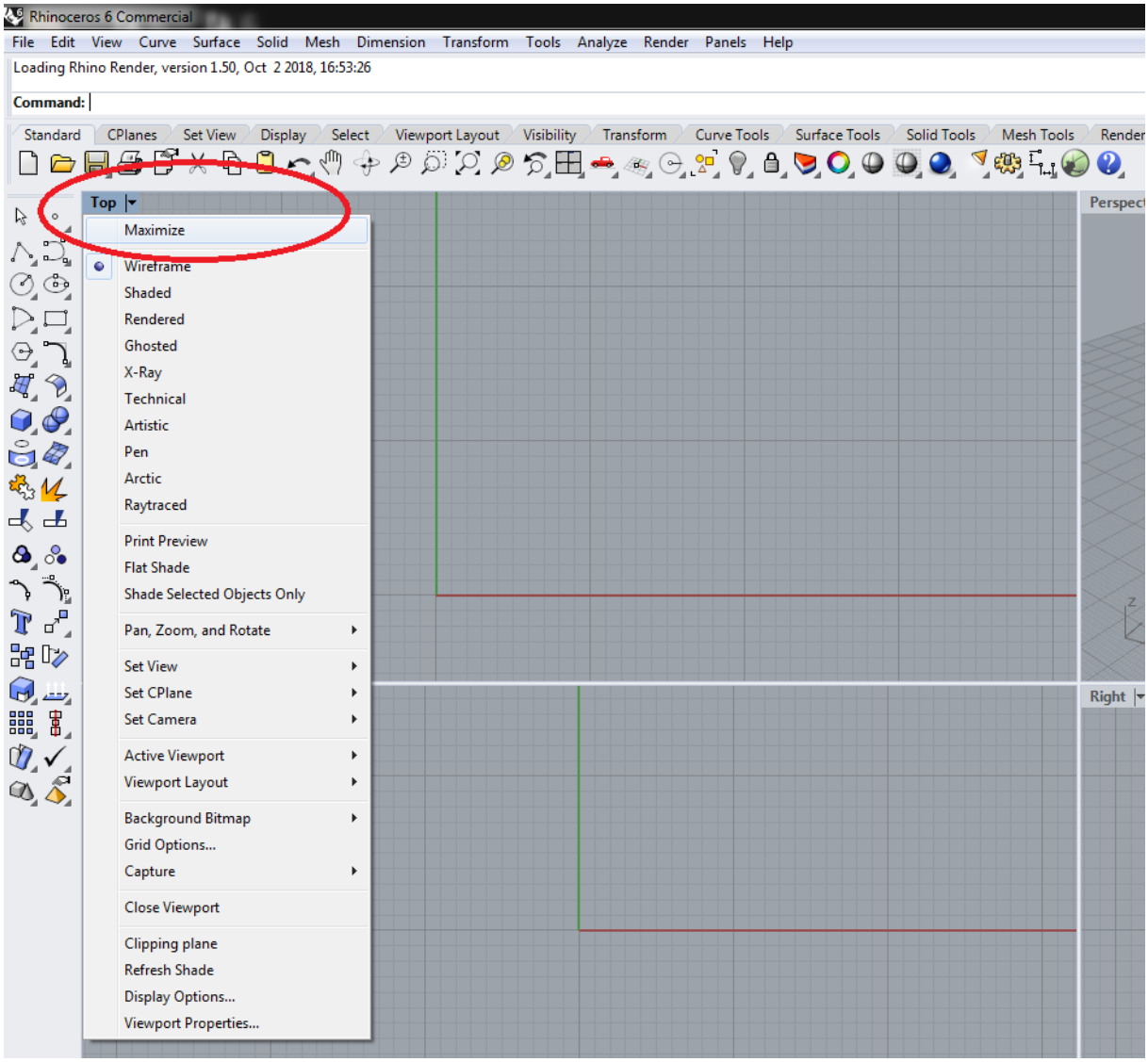
# 6. Примери: од дизајна и ЦНЦ програма до производа

Први корак за добивање готовог производа јесте његово дизајнирање у CAD софтверу, затим учитавање дизајна у CAM софтвер који генери инструкције које машина разуме, након тога се врши слање инструкција микропроцесорској јединици машине где на крају ова јединица врши контролу мотора машине и израђује жељене производе. Принцип рада може се видети на блог шеми испод:



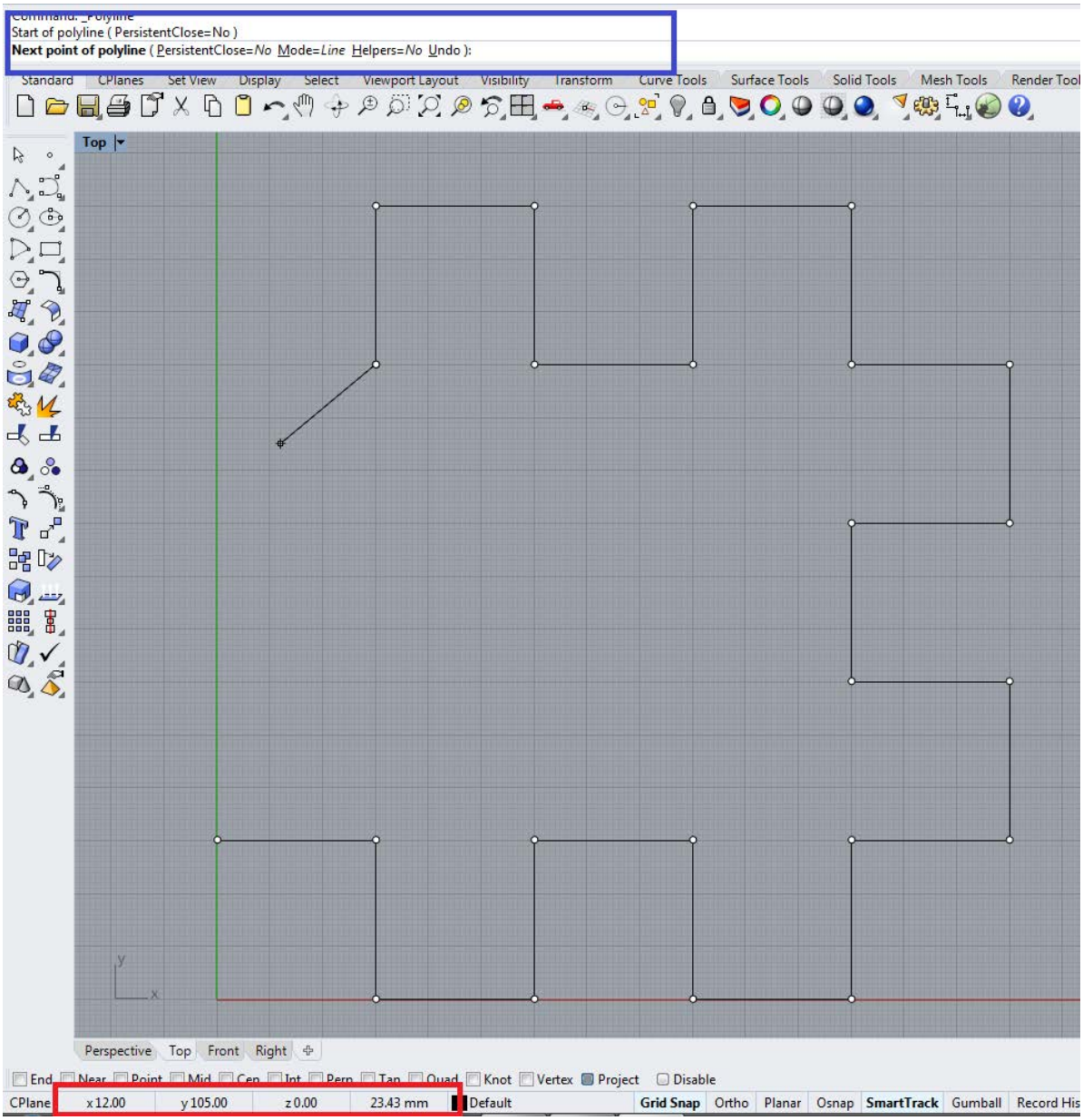

**Слика 24.** *Кораци од креирања дизајна до готовог производа*

Први пример који је одрађен јесте савијање жице у две димензије. Након покретања софтвера „Rhinoceros” може се изабрати приказ само једне равни због боље прегледности. Препоручује се да то буде раван XY мада то није од претеране важности. На слици 25. може се видети начин бирања приказа само једне равни тако што се кликне на стрелицу а затим изабере опција „Maximize”.




**Слика 25.** *Бирање приказа само једне равни у софтверу „Rhinoceros”*

У овом случају изабрана је команда „Polyline” за цртање облика. У доњем левом углу приказане су координате положаја курсора као и дужина линије која се у датом тренутку црта, док се у горњем левом углу налази поље у ком се уносе жељене команде за цртање линија.




**Слика 26.** *Цртање овлика у две димензије у софтверу „Rhinoceros”*

Након одабира команде „Polyline” из менија са леве стране, почетак цртања врши се тако што се одабере почетна тачка левим кликом миша у простору за цртање или уношењем координата у пољу за унос команди. Уколико се уносе координате потребно је притиснути ентер након уноса истих, након чега ће бити креирана почетна тачка, а уколико се цртање врши левим кликом миша, координате се могу видети у доњем левом углу као што је заокружено на слици изнад. Треба напоменути да уколико је укључена опција „Grid Snap”, што се може видети у подножју слике изнад, кретање миша је могуће само по мрежи која је дефинисана одређеном густином, у овом случају од 1 mm.

Уколико се координате уносе ручно, форма уноса је следећа:

**x,y,z (пример 3,4,5)**

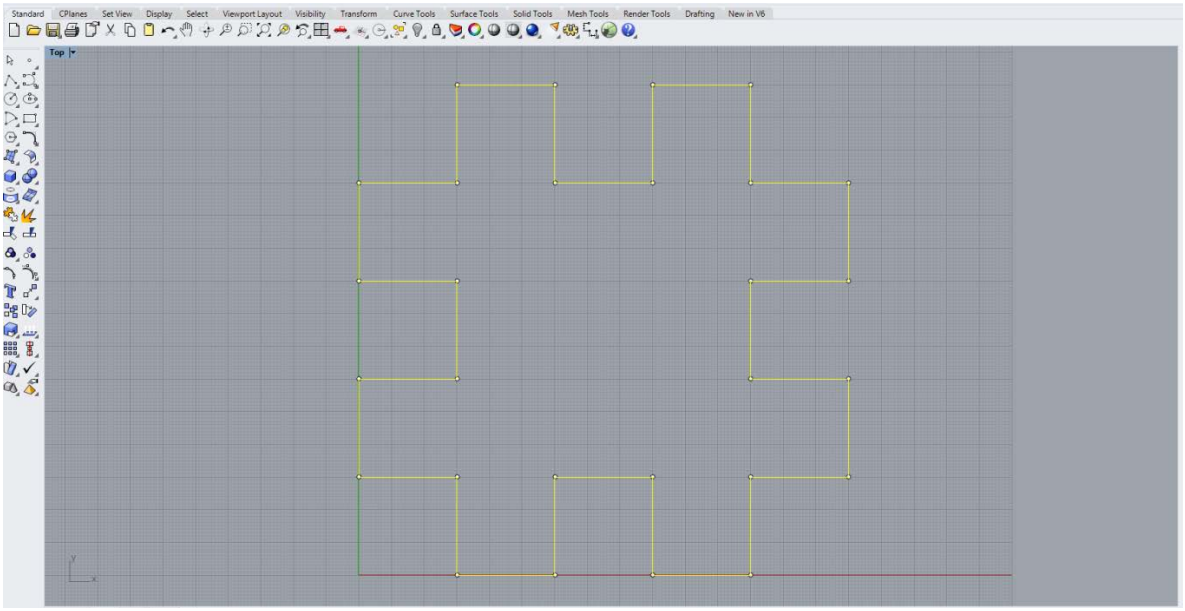
Као што се може видети у форми изнад, координате за сваку осу уносе се са зарезом између њих. Такође је могуће нацртаци линију на основу жељене дужине и угла између линије и планарне осе у изабраној равни, а форма за унос је следећа:

**@20<45**

Уколико се цртање врши у XY равни, команда изнад нацртаће линију дужине 20 mm под углом од 45° у односу на X осу.

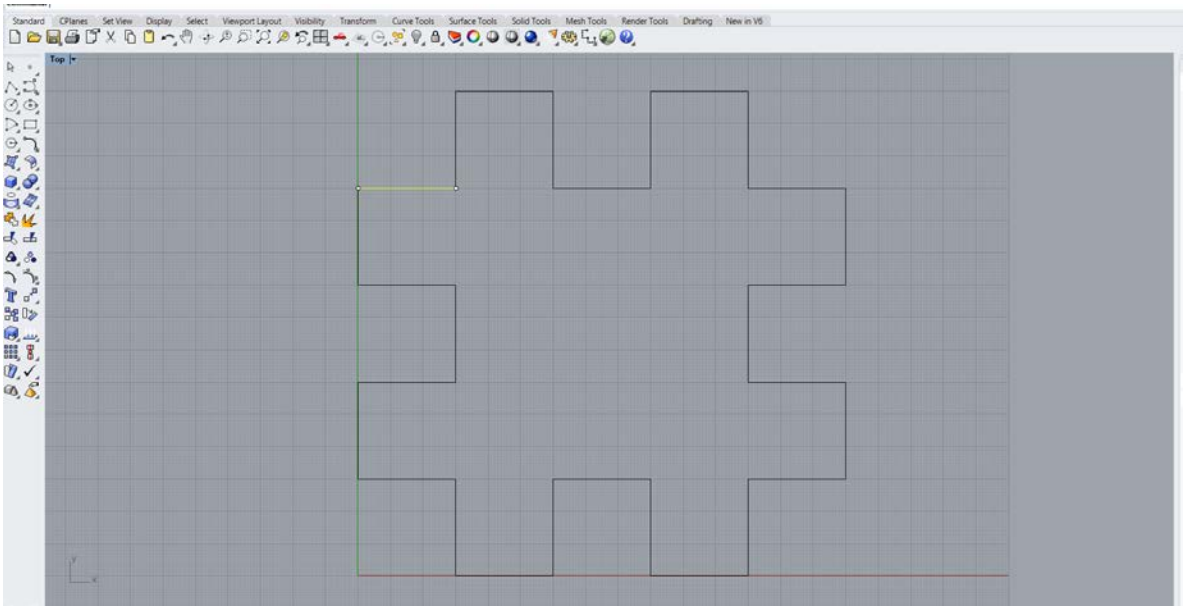
Након завршетка цртања жељеног облика потребно је кликнути на тастер ентер, помоћу ког се потврђује дизајн и затвара команда. Треба напоменути да се почетак и крај полилиније може налазити у истој тачки и неће доћи до грешке приликом генерисања кода за Ардуино као и да положај цртања објекта у равни може бити било где.

Након завршетка дизајнирања и уколико је све одрађено како треба, кликом на било коју линију пожутеће све линије као што се може се видети на слици 27. Затим је потребно дизајн сачувати као „.3dm” верзију 4 или „.obj” датотеку.




**Слика 27.** *Правилно дизајнирање у софтверу „Rhinoceros”*

Уколико линије нису спојене дизајн ће изгледати као на слици 28. где се јасно може видети да је приликом клика селектована само једна линија која је обојена жутом бојом.




**Слика 28.** *Неправилно дизајнирање у софтверу „Rhinoceros”*

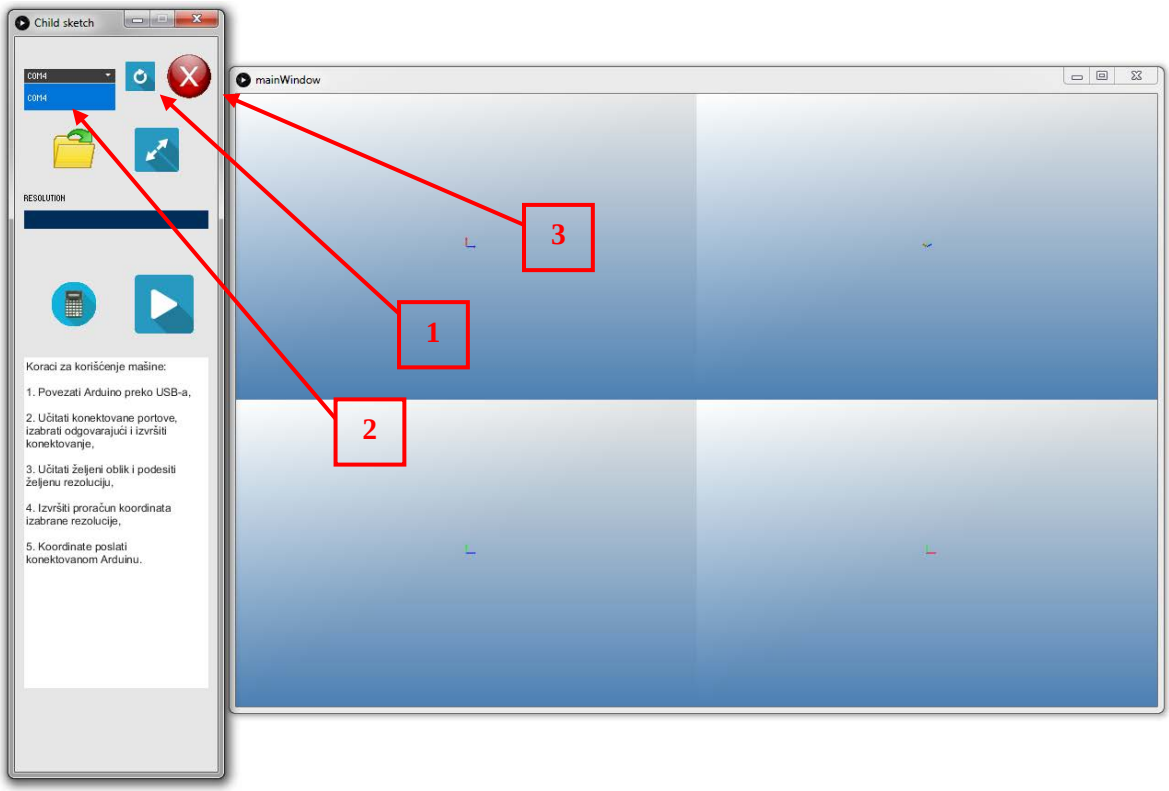
Уколико линије нису спојене, потребно их је све селектовати и затим изабрати команду „Join” која се може наћи у менију са леве стране.

Поступак цртања објекта са слике 27. помоћу уноса координата тачака или дужином и углом линија, може се одрадити следећим редоследом:

| Korak | Координате x,y | Дужина и угао у односу на планарну осу (X) |
|---|---|---|
| 1 | 0, 30 | 0, 30 |
| 2 | 30, 30 | @30<0 |
| 3 | 30, 0 | @30<-90 |
| 4 | 60, 0 | @30<0 |
| 5 | 60, 30 | @30<90 |
| 6 | 90, 30 | @30<0 |
| 7 | 90, 0 | @30<-90 |
| 8 | 120, 0 | @30<0 |
| 9 | 120. 30 | @30<90 |
| 10 | 150, 30 | @30<0 |
| 11 | 150, 60 | @30<90 |
| 12 | 120, 60 | @30<180 |
| 13 | 120, 90 | @30<90 |
| 14 | 150, 90 | @30<0 |
| 15 | 150, 120 | @30<90 |
| 16 | 120, 120 | @30<180 |
| 17 | 120, 150 | @30<90 |
| 18 | 90, 150 | @30<180 |
| 19 | 90, 120 | @30<270 |
| 20 | 60, 120 | @30<180 |
| 21 | 60, 150 | @30<90 |
| 22 | 30, 150 | @30<180 |
| 23 | 30, 120 | @30<270 |
| 24 | 0, 120 | @30<180 |
| 25 | 0, 90 | @30<270 |
| 26 | 30, 90 | @30<0 |
| 27 | 30, 60 | @30<270 |
| 28 | 0, 60 | @30<180 |
| 29 | 0, 30 | @30<270 |

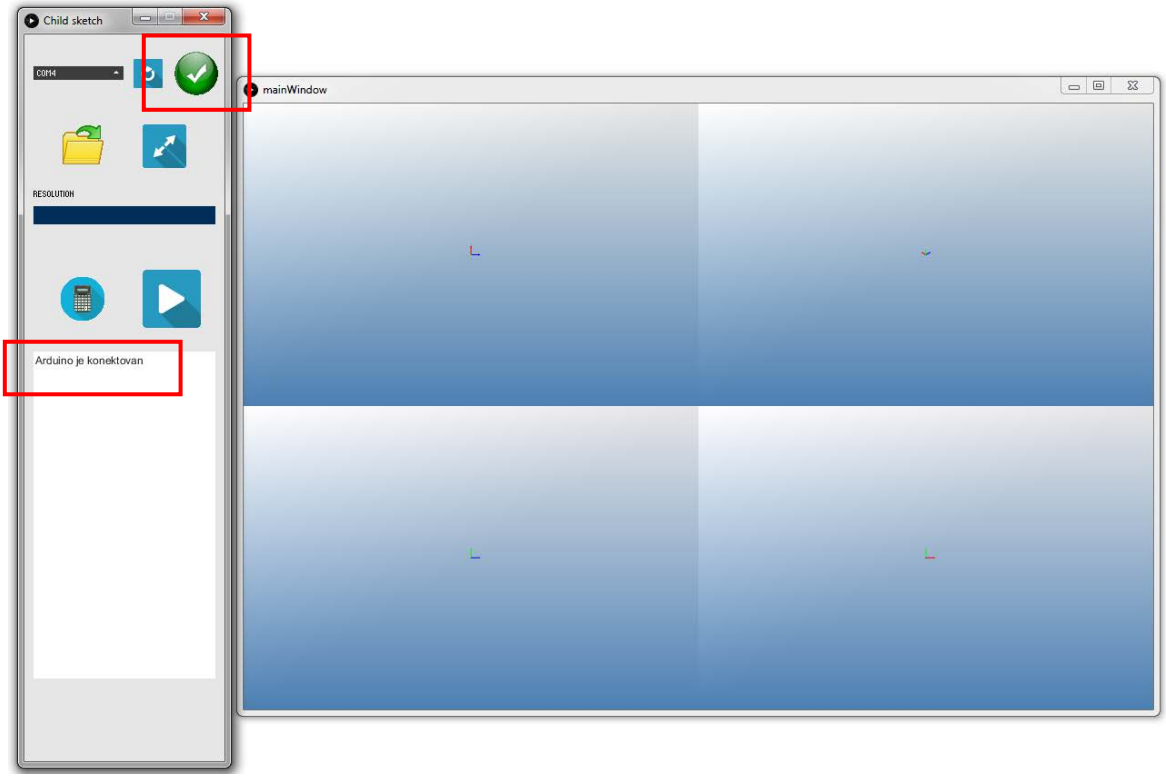
**Табела 5.** *Поступак цртања овјекта на основу унетих команди*

Након чувања датотеке потребно је покренути софтвер креиран у „Processing” окружењу и извршити повезивање са Ардуином. Први корак јесте да се кликне на тастер за освежавање листе портова, затим изабрати одговарајући порт и извршити повезивање кликом на тастер за повезивање, што се може видети на слици 29.



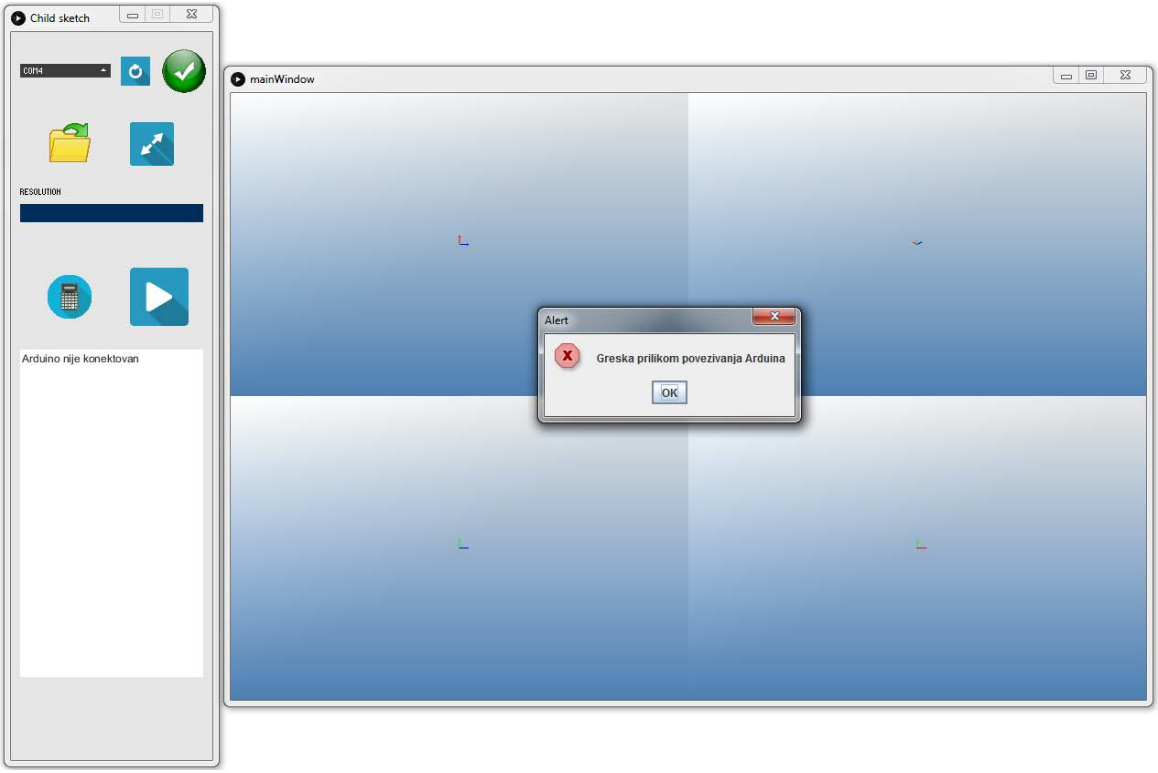

**Слика 29.** *Повезивање софтвера са Ардуином*

Након повезивања црвени тастер постаће зелен и у пољу за испис текста појавиће се порука да је повезивање са Ардуином успешно одрађено, што се може видети на слици 30.



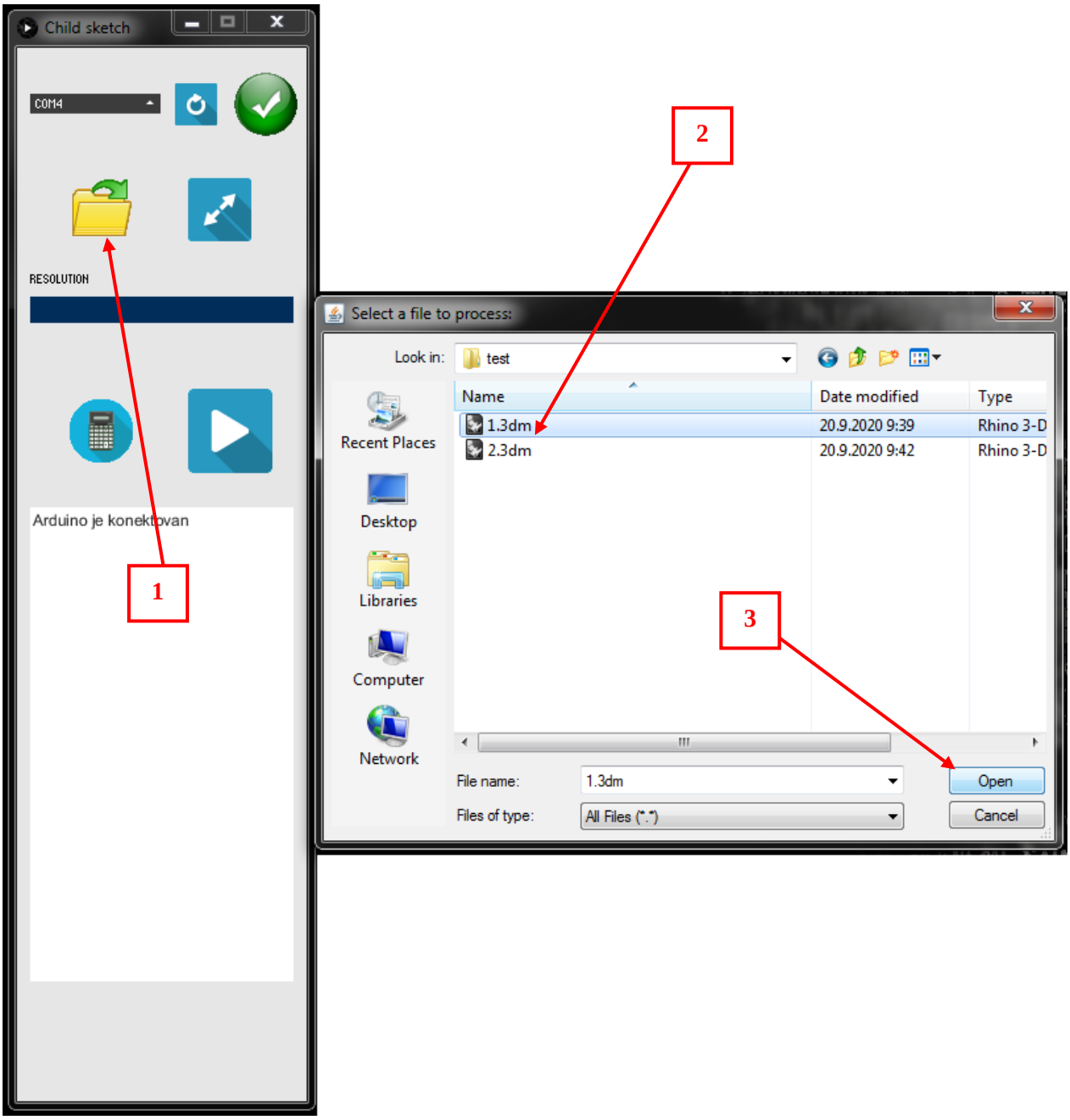

**Слика 30.** *Успешно повезивање софтвера са Ардуином*

Уколико повезивање није успешно одрађено појавиће се порука да је дошло до грешке приликом повезивања.



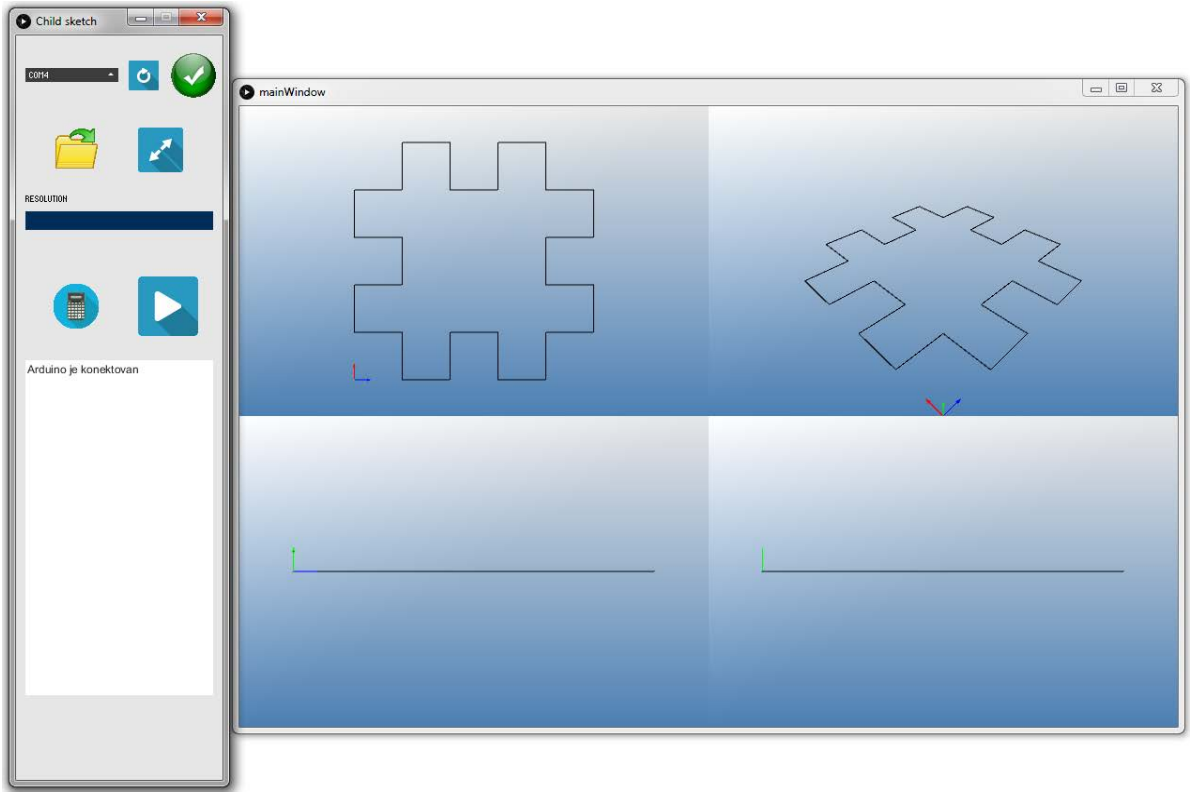

**Слика 31.** *Неуспело повезивање са Ардуином*

Након успешног повезивања са Ардуином потребно је извршити учитавање жељеног облика, а то се ради тако што се кликне на тастер за учитавање облика и изабере одговарајућа датотека, што се може видети на слици 32.



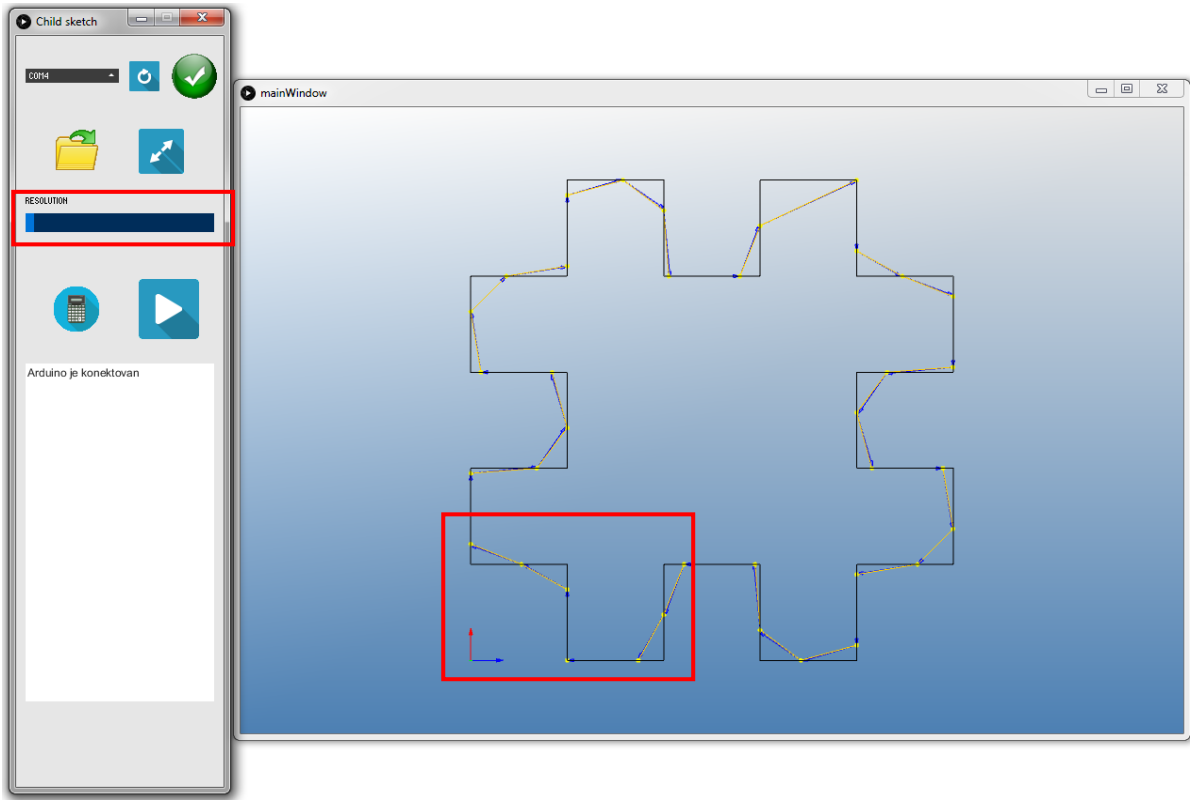

**Слика 32.** *Учитавање датотеке у софтвер за генерисање кода*

Након тога појављује се приказ као на слици 33.

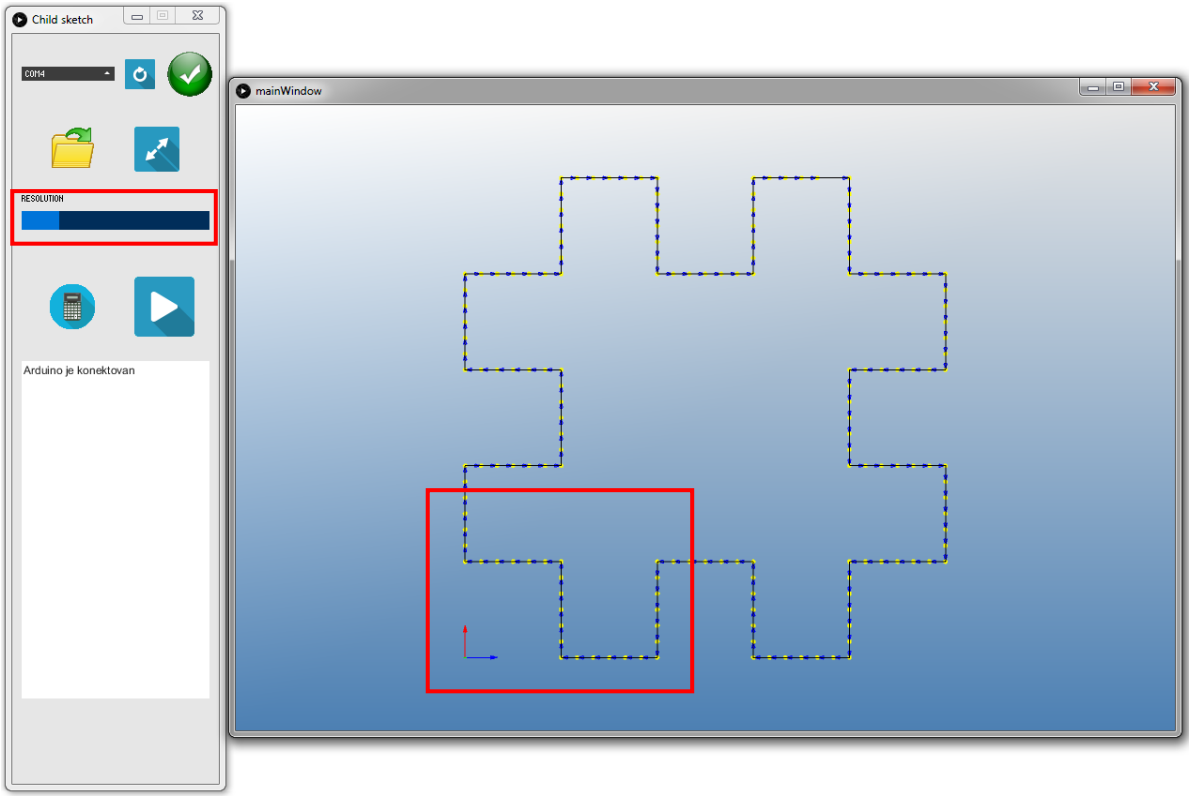



**Слика 33.** *Успешно учитавање датотеке у софтвер за генерисање кода*

Следећи корак јесте подесити резолуцију која испуњава жељени облик. Резолуција треба да буде подешена тако да што приближније приказује жељени облик. Пример лоше подешене резолуције може се видети на слици 34, док се на слици 35. може видети пример добро подешене резолуције. Пошто се ради о моделу цртаном у две димензије, у наставку ће бити изабран приказ само равни у ком је вршено цртање, а то се ради дуплим кликом на део прозора у коме се налази та раван, у овом случају горе лево.



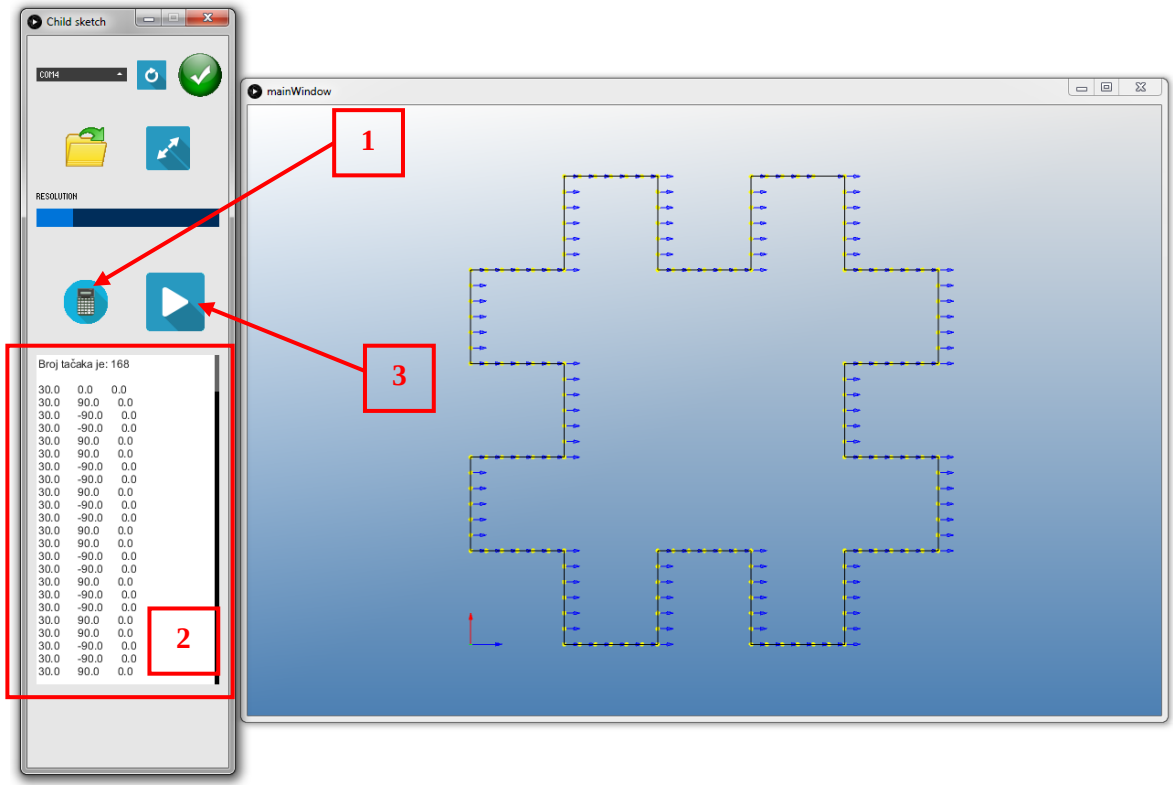

**Слика 34.** *Пример лоше подешене резолуције*




**Слика 35.** *Пример добро подешене резолуције*

Приликом подешавања резолуције генерише се облик у виду жутих линија који приказује како ће жица изгледати након савијања. Треба обратити пажњу приликом подешавања резолуције јер не значи да уколико је резолуција већа да је и прецизност већа, већ треба водити рачуна да приликом њеног подешавања генерисани облик буде што приближнији или једнак жељеном облику.

Након подешавања резолуције потребно је кликнути на тастер за прорачунавање вредности, након чега се све вредности прорачунавају и исписују у белом пољу што се може видети на слици 36. Уколико је оператер задовољан прорачуном потребно је кликнути на тастер за слање података ка Ардуину.

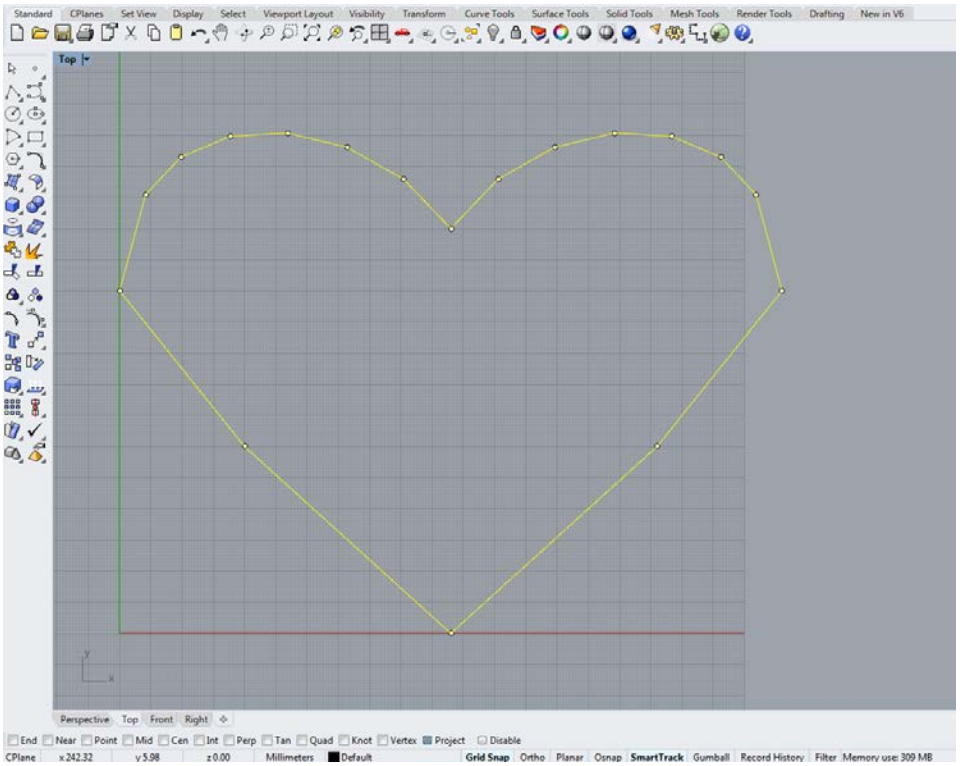



**Слика 36.** *Пример прорачунатих вредности спремних за слање ка Ардуину*

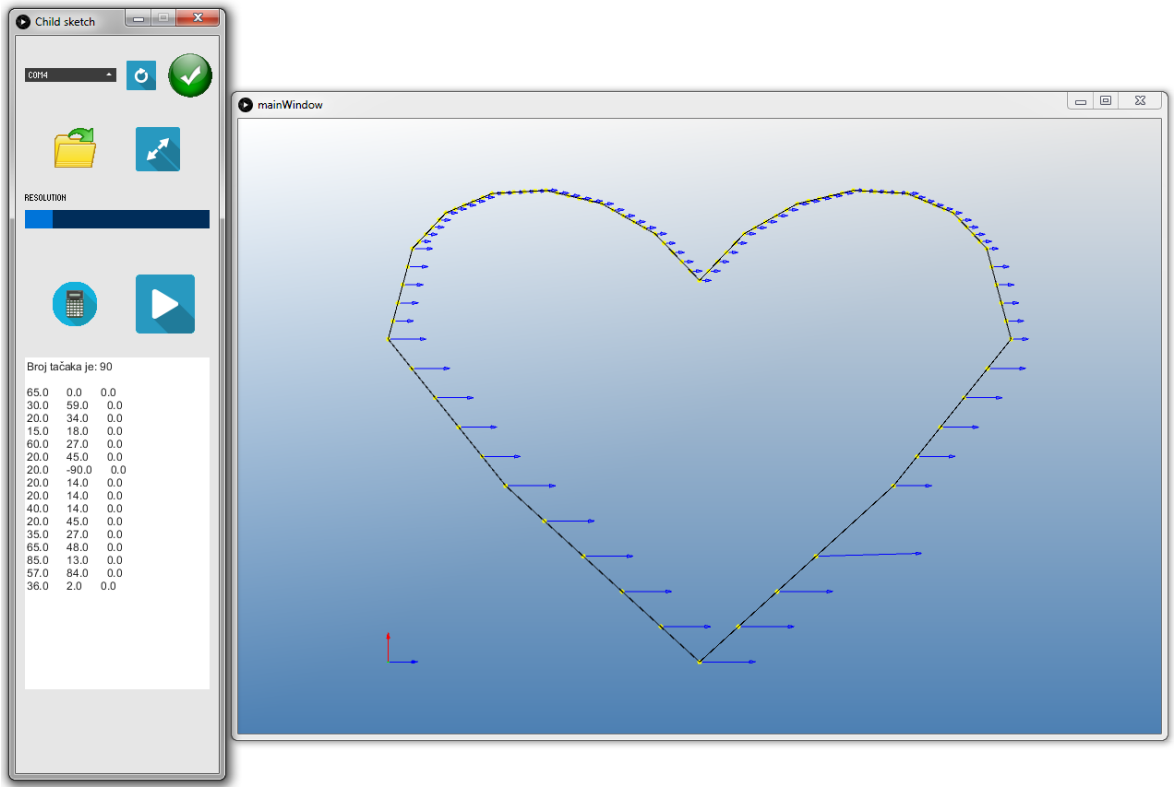
На слици изнад могу се видети три колоне са подацима. Прва колона означава дужину за коју се увлачи жица, друга означава угао за који се жица савија и трећа угао за који се врши ротирање главе мотора. Треба напоменути да се подаци извршавају са десна на лево, односно прво се врши ротирање главе, затим савијање жице и након тога увлачење жице. На слици 37. може се видети коначан производ.

**Слика 37.** *Први пример коначног производа*

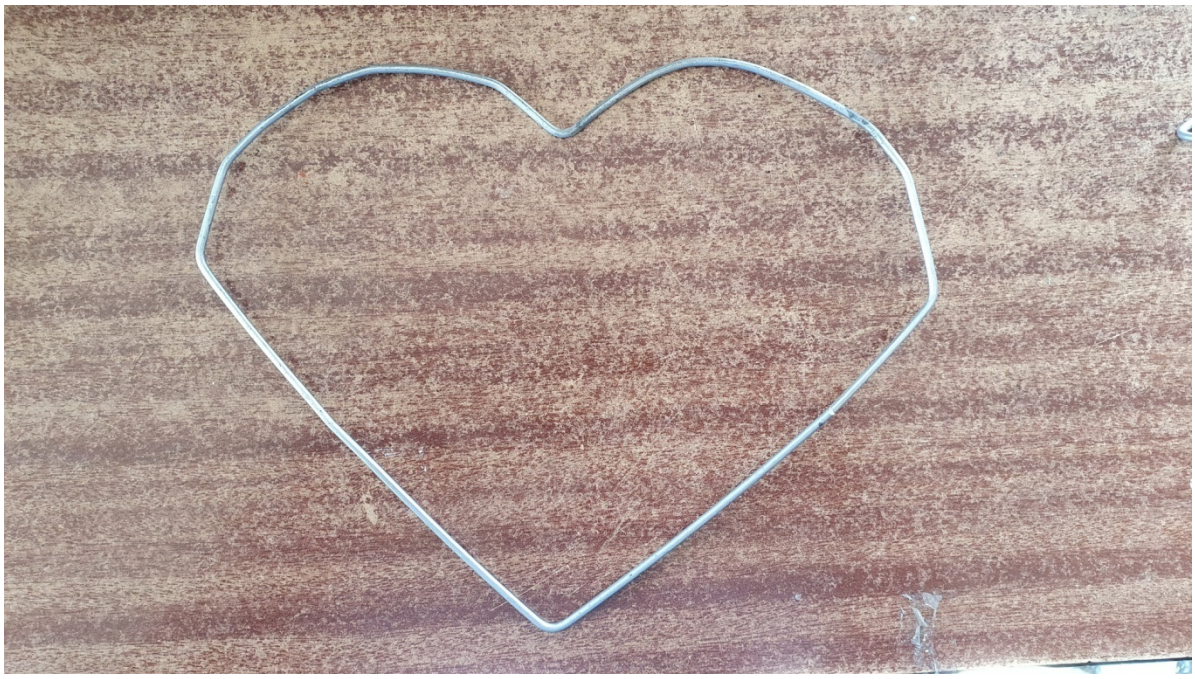
У наставку се могу видети још неки примери од дизајнирања па до крајњег производа. Пошто је такође реч о 2D моделу као у претходном примеру, цео поступак је потпуно исти.




**Слика 38.** *Дизајнирање модела у облика срца*

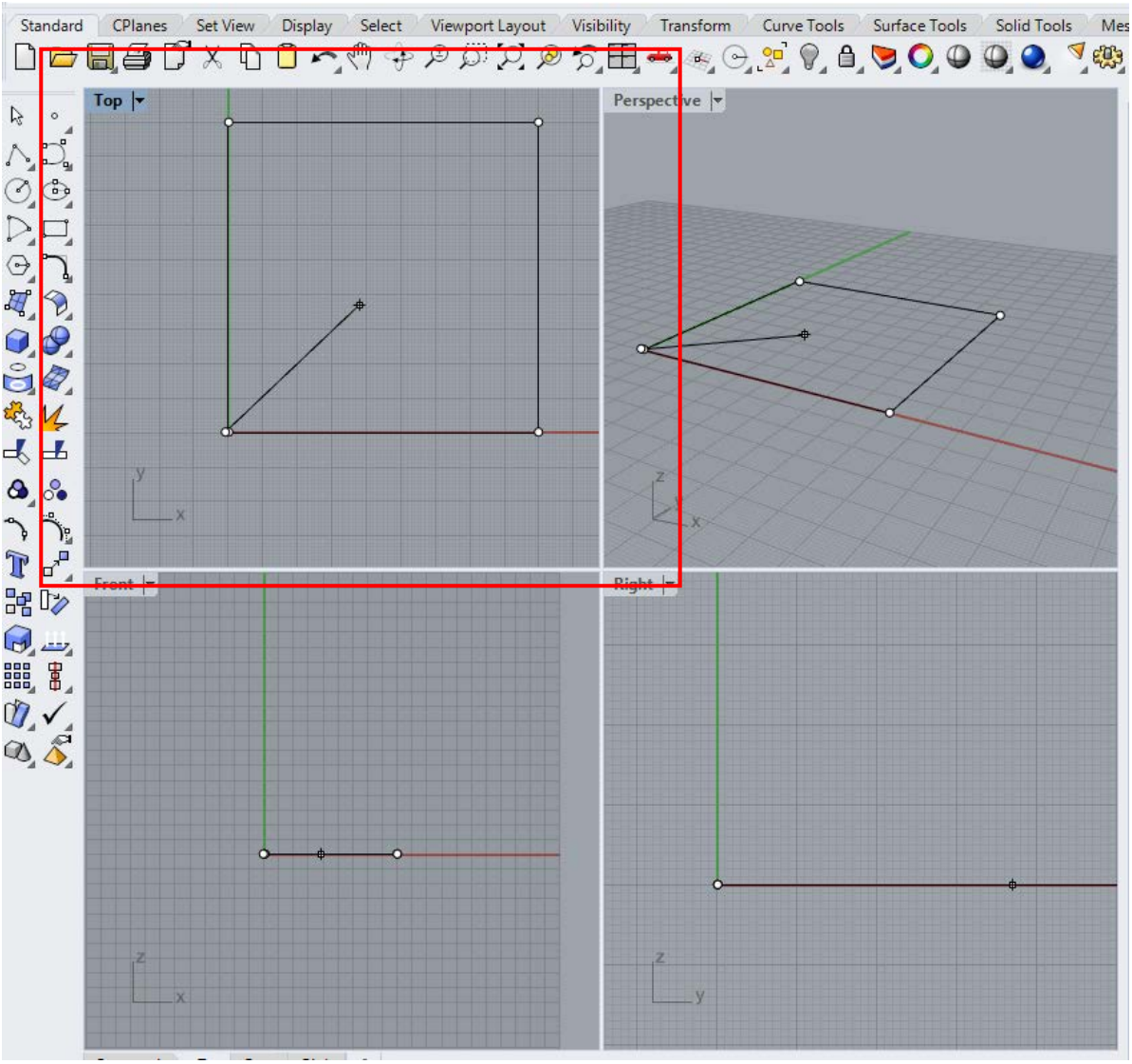



**Слика 39.** *Облик срца у софтверу за прорачунавање*

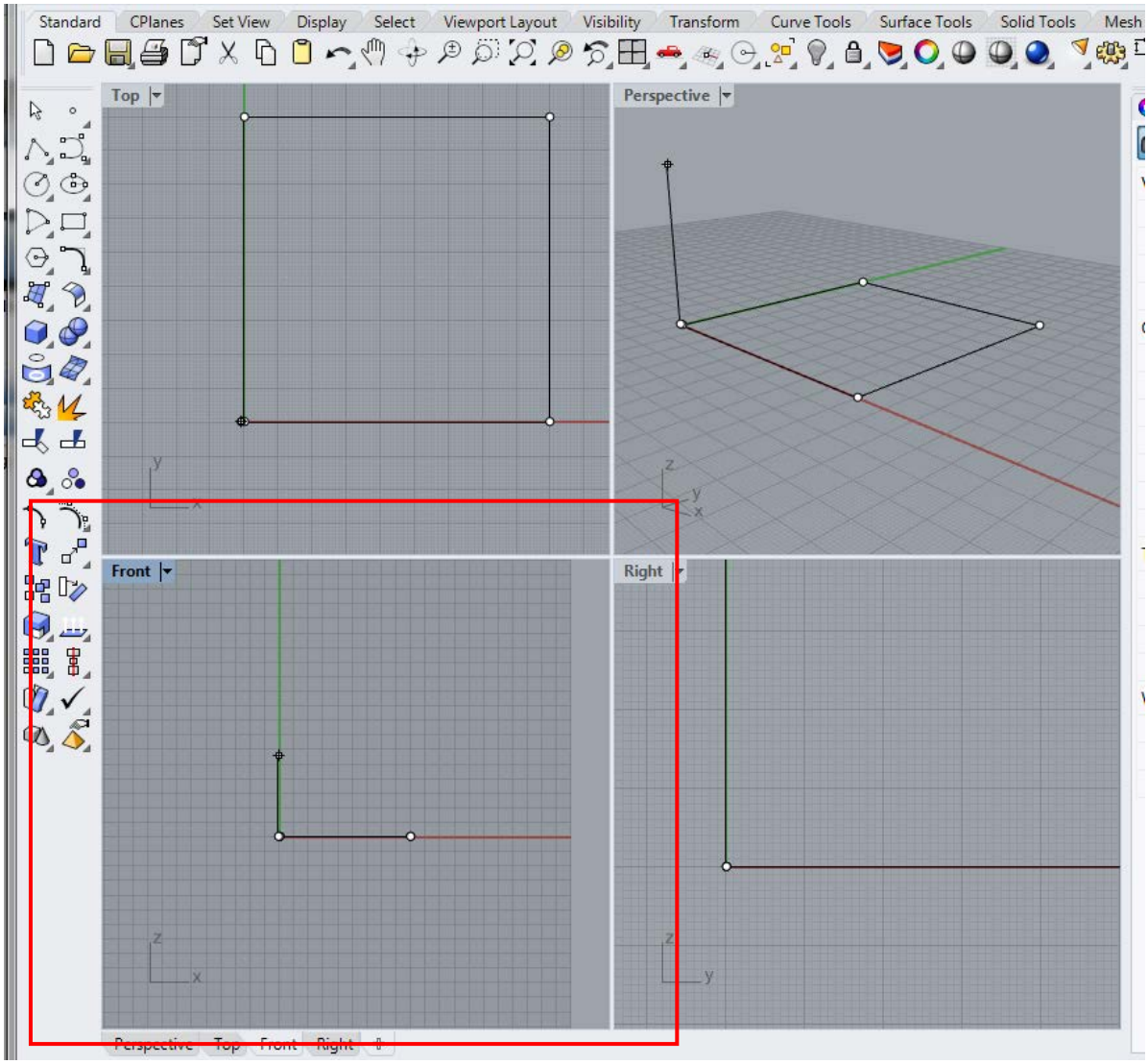


**Слика 40.** *Коначан производ у облику срца*

За цртање 3Д модела користе се потпуно исте команде као и за цртање 2Д модела, с'тим што је разлика у томе што се у овом случају користе све три равни уместо једне. На 3Д примеру испод може се приметити да је цртање започето у XY равни, што се може и видети на слици 41. Након завршетка цртања у тој равни, оно је настављено у XZ равни где се јасна разлика може приметити на слици 42.

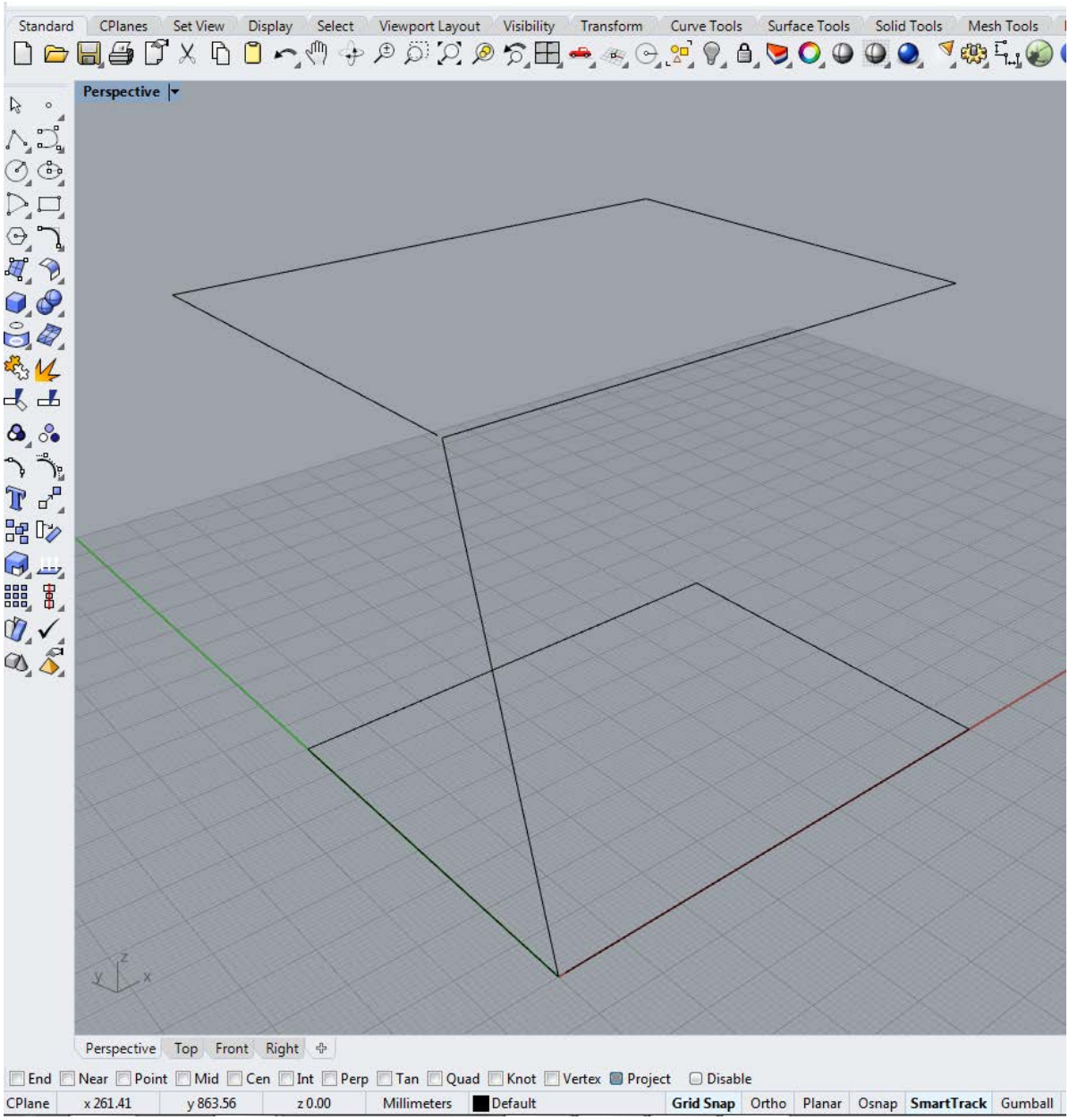



**Слика 41.** *Почетак цртања 3Д модела у XY равни*




**Слика 42.** *Наставак цртања модела у XZ равни*

Завршен облик 3Д модела може се видети на слици 43.



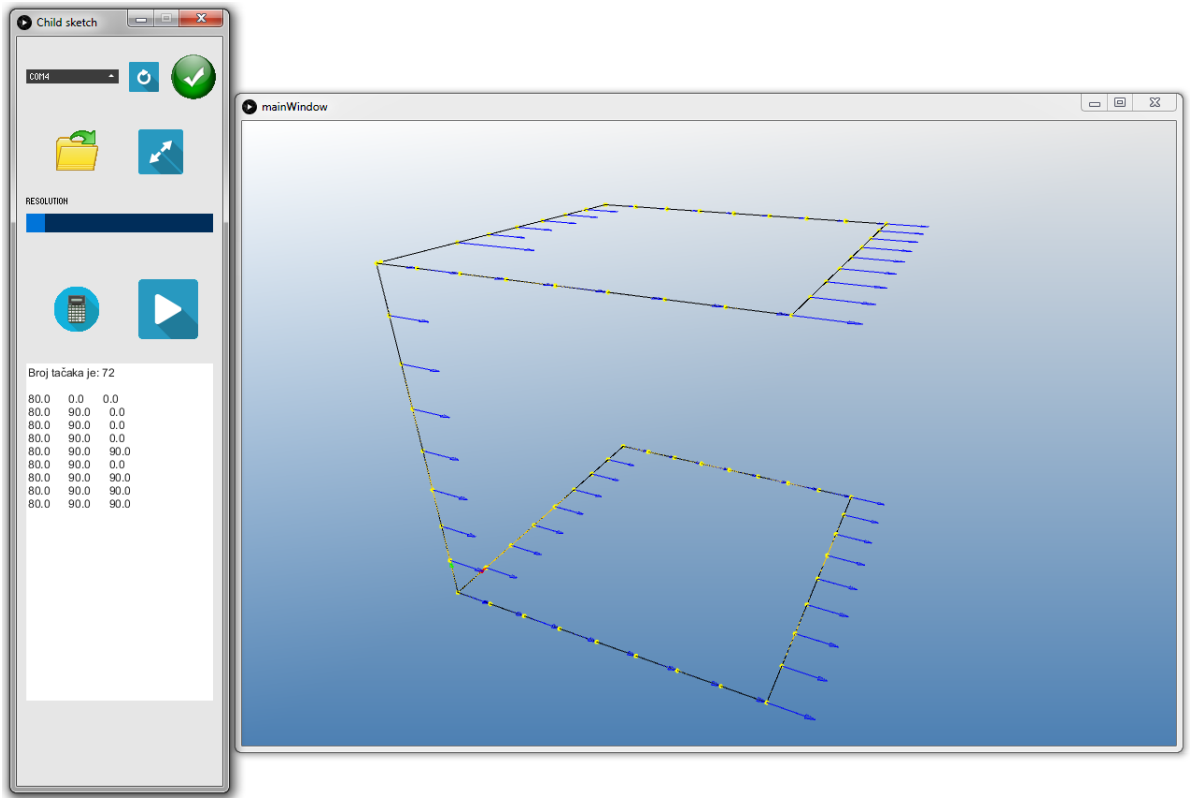

**Слика 43.** *Коначан облик 3Д модела*

Поступак цртања претходног модела на основу уношења координата може се видети у табели 6:

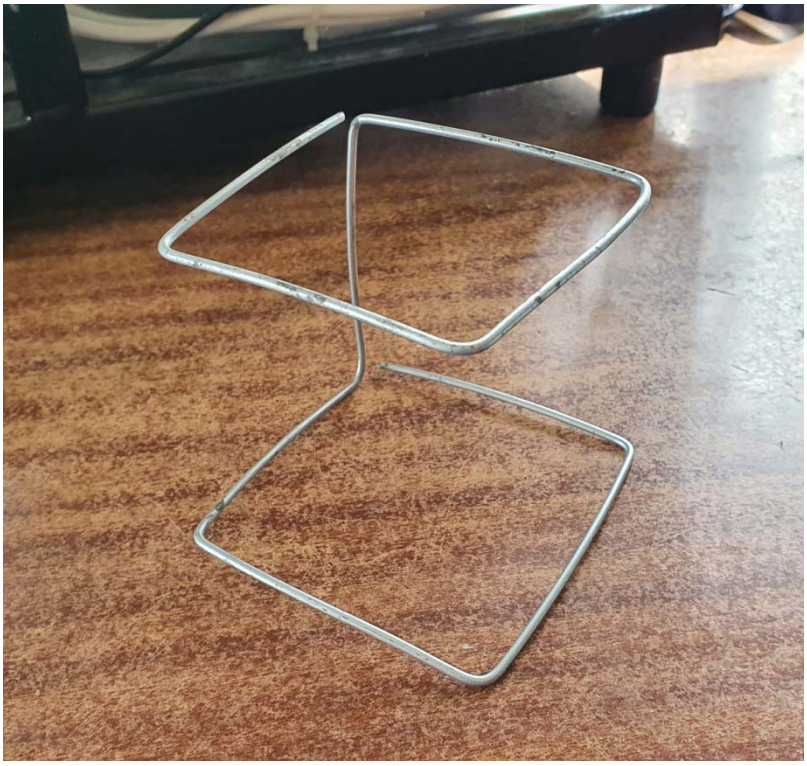
| Korak | Координате x,y,z |
|---|---|
| 1 | 1, 0, 0 |
| 2 | 80, 0, 0 |
| 3 | 80, 80, 0 |
| 4 | 0, 80, 0 |
| 5 | 0, 0, 0 |
| 6 | 0, 0, 80 |
| 7 | 80, 0, 80 |
| 8 | 80, 80, 80 |
| 9 | 0. 80, 80 |
| 10 | 0, 1, 80 |

**Табела 6.** *Поступак цртања објекта на основу унетих координата*

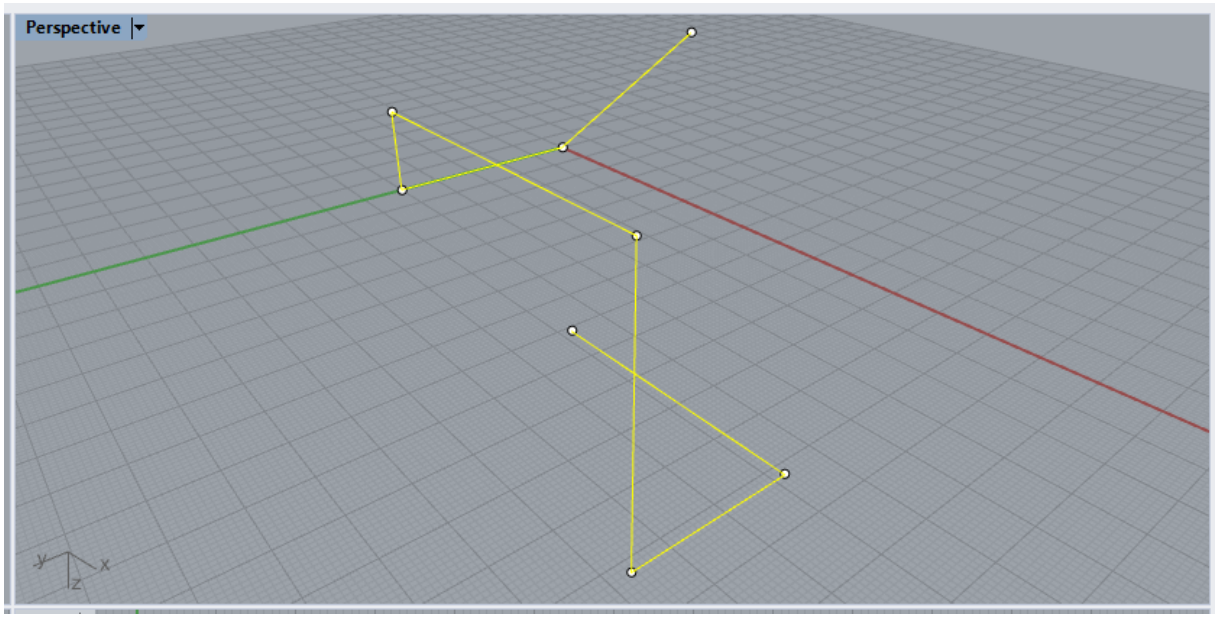
Комплетан наставак процедуре исти је као и за 2Д цртеже, што значи да је једина разлика у процесу цртања, док су све остале тачке потпуно једнаке.




**Слика 44.** *Облик у 3 димензије у софтверу за прорачунавање*



**Слика 45.** *Коначан производ у 3 димензије*

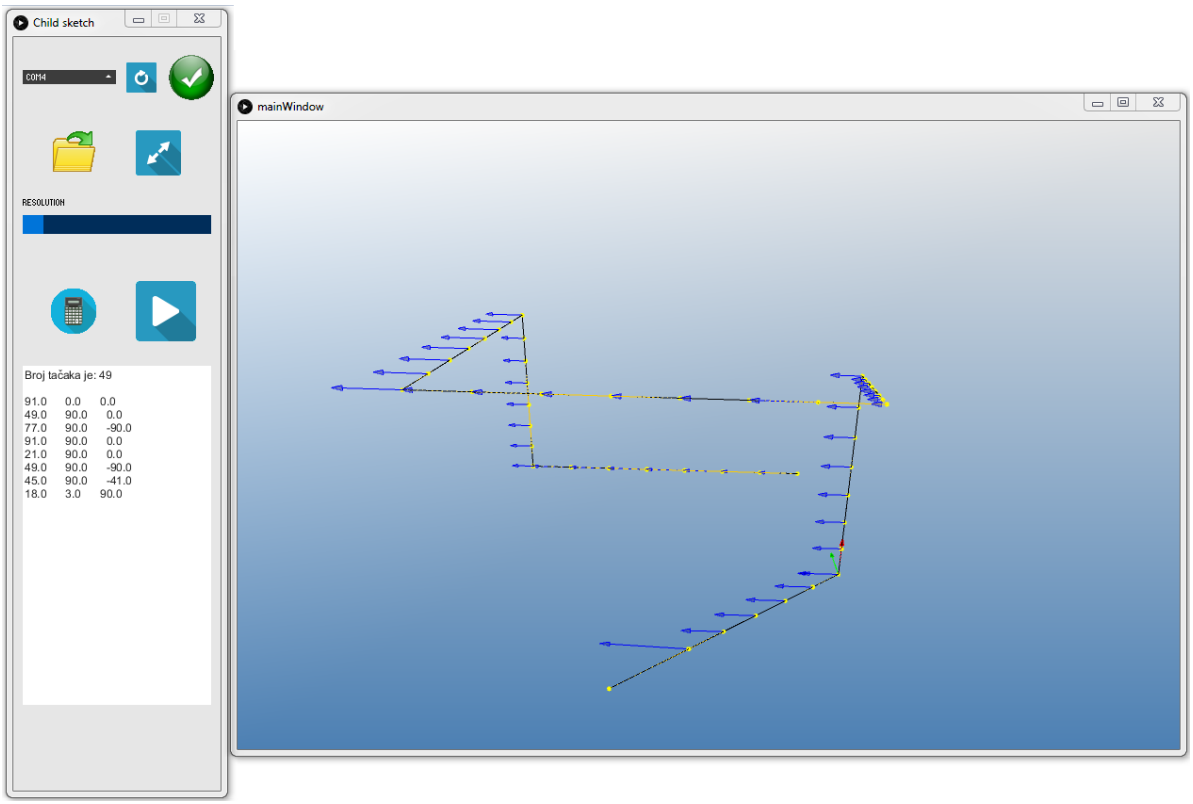



**Слика 46.** *Дизајнирање модела држача смарт телефона*

Поступак цртања модела дражача за смарт телефон на основу уношења координата може се видети у табели 7:

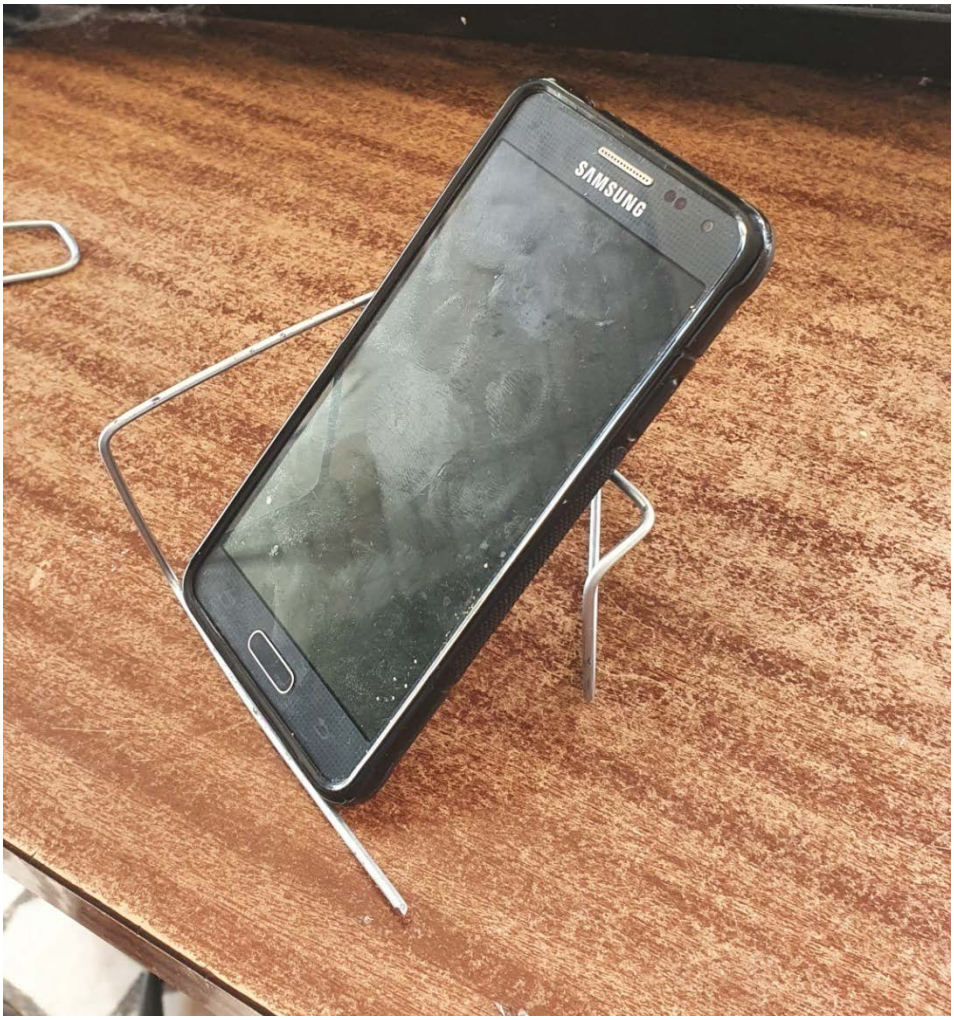
| Korak | Координате x,y,z |
|---|---|
| 1 | 0, 0, 60 |
| 2 | 90, 0, 60 |
| 3 | 90, 50, 60 |
| 4 | 90, 50, -20 |
| 5 | 0, 50, -20 |
| 6 | 0, 50, 0 |
| 7 | 0, 0, 0 |
| 8 | 50, 0, -40 |

**Табела 7.** *Поступак цртања овјекта на основу унетих координата*




**Слика 47.** *Држач за смарт телефон у софтверу за прорачунавање*

**Слика 48.** *Коначан проитвод држача за смарт телефон*




**Слика 49.** *Коначан проитвод држача за смарт телефон приликом држања телефона*

# 7. Перформансе уређаја

Перформансе уређаја највише зависе од компоненти које се користе али и од завршне обраде свих делова који су израђени са наменом за ову машину. У наставку биће објашњена брзина уређаја али и начин на који се врши корекција угла преко полиномске једначине.

## 7.1. Брзина уређаја

Тестирањем и коришћењем машине утврђене су следеће оптималне брзине:

| Материјал жице | Пречник жице [mm] | Брзина савијања [°/s] | Брзина ротације главе за савијање [°/s] | Брзина увлачења жице [mm/s] |
|---|---|---|---|---|
| Челик | 2 | 20.525 | 25.714 | 40.683 |

**Табела 8.** *Оптималне перформансе уређаја за челичну жицу*

Брзина увлачења жице и брзина ротације главе за савијање израчунате су на основу подешених параметара приликом програмирања микропроцесорске јединице. Због конструкције машине и специфичног рачунања угла потребног за савијање жице, брзина савијања узета је као просечна вредност.

На основу података из табеле изнад, врло лако можемо израчунати потребно време за савијање жице за одређене облике. Пример рачунања времена са савијање жице у облику квадрата, страница 50 mm може се видети испод:

$$T_s = 4 \cdot \frac{90^\circ}{20.525^\circ} = 17.54\ s$$

$$T_f = 4 \cdot \frac{50}{40.683} = 4.91\ s$$

$$T_r = 0$$

$$\boldsymbol{T_u = T_s + T_f + T_r = 22.45\ s}$$

### 7.2. Корекција угла помоћу полиномске једначине

Корекција угла помоћу полиномске једначине врши се пре свега због несавршености израде појединих делова машине. У питању су делови који су израђени од пластике и који се налазе у делу за савијање жице. Због одступања жељених димензија делова, јавила се неправилност приликом савијања жице и који директно утичу на крајни резултат готовог производа. У табели 9. могу се видети вредности жељених углова, вредности које су првобитно измерене са савијањем без корекције и вредности које су измерене након корекције.

| Жељени угао савијања [°] | Савијен угао без корекције [°] | Савијен угао са корекцијом [°] |
|---:|---:|---:|
| 30 | 1 | 31 |
| 35 | 3 | 33 |
| 40 | 9 | 40 |
| 45 | 15 | 47 |
| 50 | 18 | 50 |
| 55 | 22 | 54 |
| 60 | 28 | 60 |
| 65 | 35 | 66 |
| 70 | 40 | 69 |
| 75 | 50 | 76 |
| 80 | 57 | 80 |
| 85 | 67 | 85 |
| 90 | 76 | 90 |
| 95 | 83 | 94 |
| 100 | 92 | 101 |
| 105 | 98 | 105 |
| 110 | 104 | 110 |
| 115 | 110 | 115 |
| 120 | 117 | 120 |
| 125 | 123 | 125 |
| 130 | 128 | 130 |

**Табела 9.** *Врености углова пре и након корекције*

У овом случају коришћена је полиномска једначина осмог реда за прорачунавање жељеног угла. Једначина се може видети испод а њени коефицијенти се налазе у табели 10. [17]

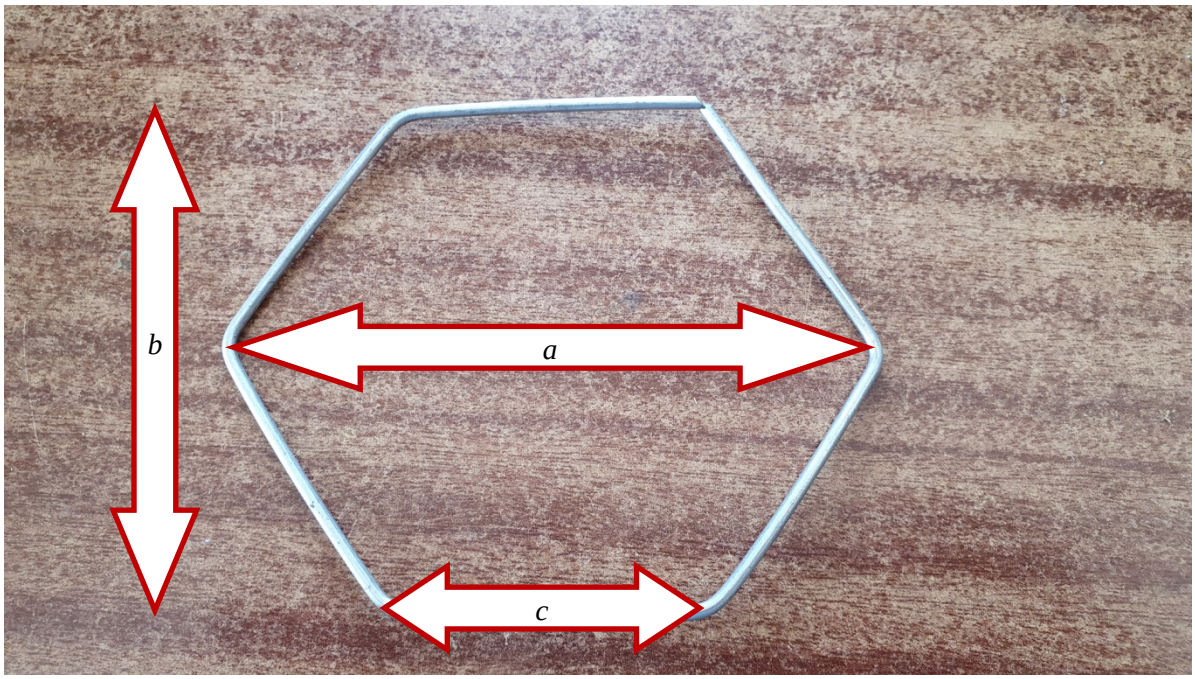
$$y = a \cdot x^8 + b \cdot x^7 + c \cdot x^6 + d \cdot x^5 + e \cdot x^4 + f \cdot x^3 + g \cdot x^2 + h \cdot x + i$$

| Коефицијент | Вредност коефицијента |
|---|---|
| *a* | -1.513583083/100000000000000 |
| *b* | 1.004753281/100000000000 |
| *c* | -2.508854186/1000000000 |
| *d* | 2.962127864/10000000 |
| *e* | -1.634057824/100000 |
| *f* | 3.405019148/10000 |
| *g* | -5.189371596/1000 |
| *h* | 1.155772463 |
| *i* | 29.93500616 |

**Табела 10.** *Коефицијенти полиномске једначине [17]*

### 7.3. Прецизност израде готових производа

Прецизност израде може се утврдити поређењем димензија у CAD софтверу и директним мерењем готових производа. На следећим примерима извршена су мерења и коначни закључци.

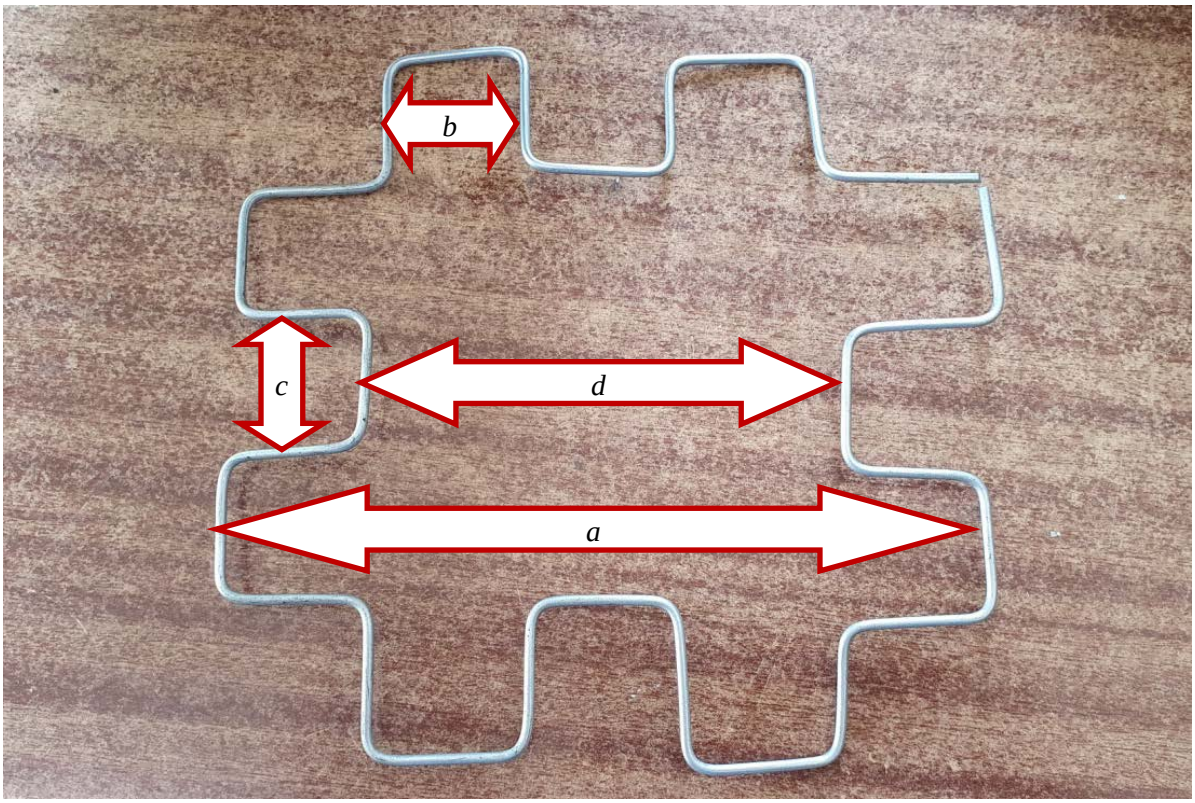



**Слика 50.** *Први узорак за мерење прецизности*

Измерене вредности и нацртане вредности дате су у табели испод:

| Вредност | Из CAD дизајна [mm] | Измерена [mm] | Грешка |
|---|---|---|---|
| *a* | 110 | 115 | 4% |
| *b* | 80 | 82 | 2% |
| *c* | 50 | 47 | -6% |

**Табела 11.** *Грешке у димензијама готовог производа*




**Слика 51.** *Други узорак за мерење прецизности*

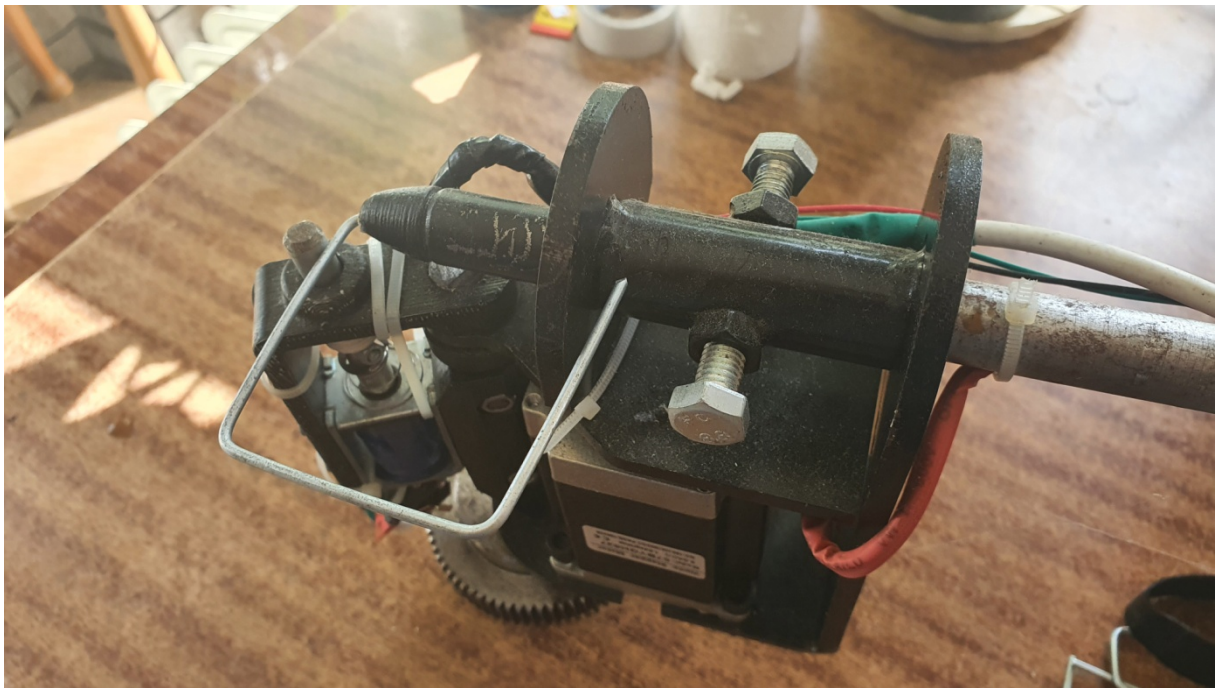
| Вредност | Из CAD дизајна [mm] | Измерена [mm] | Грешка |
|---|---|---|---|
| *a1* | 150 | 153 | 2% |
| *a2* | 150 | 155 | 3% |
| *b1* | 30 | 30 | 0% |
| *b2* | 30 | 31 | 3% |
| *c1* | 30 | 31 | 3% |
| *c2* | 30 | 31 | 3% |
| *d1* | 90 | 92 | 2% |
| *d2* | 90 | 96 | 6% |

**Табела 12.** *Грешке у димензијама другог узорка*

На основу мерења оба узорка може се закључити да се грешка у димензијама између готовог производа и CAD дизајна креће у опсегу ±6%.

# 8. Недостаци машине

Приликом коришћења и рада машине примећени су неки недостаци који се могу јавити у неким одређеним случајевима. Први недостатак јесте у начину конструкције главе која врши савијање жице, јер у неким одређеним случајевима може доћи до заглављивања жице при чему она неће бити савијена на жељени начин. Пример где се види са је жица заглављена може се видети на слици 52.



**Слика 52.** *Пример заглављивања жице*

Другачијом конструкцијом овог дела машине могуће је смањити заглављивање жице, али га је немогуће потпуно уклонити. Још један од недостатака јесте тај што машина нема модул који би вршио одмотавање жице из котура на ком је намотана, што захтева присуство оператера да потпомаже у раду машине. На крају потребно је извршити одсецање жице коју такође оператер врши помоћу кљешта или неког другог алата.

# 9. Закључна разматрања

Полазна тачка овог мастер рада јесте била идеја да се направи једна CNC машина која ће моћи да врши савијање жице на основу креираног CAD дизајна. Овај рад обухвата више задатака из различитих области као што су CAD дизајнирање, електротехника, програмирање, механика и тригонометрија, па је током рада било потребно савладати доста изазова које све ове области доносе.

Сам почетак јесте био планирање и конструисање машине где је било потребно дизајнирати велики број специјалних делова и које је било потребно израдити уз помоћ других CNC машина. Такође је било потребно суочити се са областима електротехнике где је било потребно водити рачуна да све компоненте буду усклађене једна са другом. Највећи изазов представљало је само програмирање машине јер је пре свега било потребно извршити програмирање микропроцесорске јединице, а затим креирати софтвер који ће моћи да врши рачунање свих вредности на основу CAD дизајна и ускладити комуникацију са микропроцесорском јединицом. На крају али не мање важно, било је потребно пронаћи одговарајући софтвер за дизајнирање жичаних форми и научити његово коришћење.

С'обзиром да се ради о ручној и кућној изради, машина има својих недостатака који су првенствено ту због немогућности прецизне и квалитетне израде појединих делова који су веома критични и доста утичу на крајњи рад саме машине. Без обзира на њих, могућност грешке је сведен на минимум на основу додатним програмирањем машине.

Још један од закључака јесте тај да Ардуино платформа има широку могућност примене и да се може користити за контролу неке CNC машине и да се на тај начин може примењивати и у индустрији. Ова платформа такође представља веома једноставну платформу за едукацију студената који немају предзнање, а имају жељу да се баве програмирањем микроконтролера.

Рад на овом пројекту доноси велико задовољство јер је задатак био да се креира машина која ће бити функсионална и једноставна за коришћење. Такође је стечено доста знања и искуства из различитих области што може бити од велике користи у будућој каријери машинског инжењера.

Надам се да је овај рад обезбедио довољно предзнање за колеге који желе да се баве у овој области машинског инжењерства, али и да им је показатељ да се уз велики труд и напор сви циљеви могу постићи.

# 10. Референце

[1] Хенсен А.: Програмирање на језику С, Микрокњига, Београд, 1991.

[2] Милићев Д.: Објектно оријентисано програмирање на језику С++, Микрокњига, Београд, 1991.

[3] Ћирковић Р.: Програмирање CNC машина: FeatureCAM, Микрокњига, 2015.

[4] Massimo Banzi „Getting Started with Arduino, 2nd Edition“, 2011.

[5] Casey Reas & Ben Fry „Getting Started with Processing, First Edition“, 2010.

[6] Трајановић М., Грујовић Н., Миловановић Ј., Миливојевић В..: Рачунарски подржане брзе производне технологије, Крагујевац, 2008.

[7] https://cdn-learn.adafruit.com/downloads/pdf/all-about-stepper-motors.pdf (8. август 2020),

[8] https://www.allaboutcircuits.com/tools/stepper-motor-calculator/ (8. август 2020),

[9] http://www.nicemach.com/us/product/cnc-wire-bending-machine.htm (10. септембар 2020),

[10] https://components101.com/microcontrollers/arduino-nano (2. септембар 2020),

[11] http://www.sojamo.de/libraries/controlP5/,(2. јул 2020),

[12] http://igeo.jp/, (15. јул 2020),

[13] https://sr.wikipedia.org/wiki/Arduino (9. август 2020),

[14] https://processing.org/ (9. август 2020),

[15] https://www.arduino.cc/en/reference/stepper (21. септембар 2020),

[16] https://www.tutorialspoint.com/arduino/arduino_stepper_motor.htm (21. септембар 2020),

[17] https://arachnoid.com/polysolve/ (07. август 2020),

[18] https://www.tutorialspoint.com/arduino/arduino_math_library.htm (15. јул 2020),

[19] https://en.wikipedia.org/wiki/Rhinoceros_3D (15. јул 2020),

[20] https://fileinfo.com/extension/3dm (21. септембар 2020),

[21] https://fileinfo.com/extension/obj (21. септембар 2020),