**ПРОЕКТ: MP1**

**1. ТЕМА: 2Д Принтер**

**2. АВТОРИ:**

**Име**: Пламена Петрова Николова

**ЕГН**: 0247018693

**Адрес**: гр. Хасково, бул. Васил Левски 61

**GSM**: 0886418389

**ЕMAIL**: plamenaxaskovo@gmail.com

**УЧИЛИЩЕ**: Природо-математическа гимназия "Акад. Боян Петканчин"

**3. РЪКОВОДИТЕЛ:**

**ИМЕ**: Коля Петрова

**GSM**: 0885827390

**ЕMAIL**: k\_\_petrova@abv.bg

**ДЛЪЖНОСТ**: Старши учител по информатика и ИТ

**4. РЕЗЮМЕ:**

**4.1. Цели (предназначение, кратък анализ на потребностите и на съществуващите**

**решения):**

Принтера, който разработвам, е предназначен за изчертаване на малки картинки. Използва връзка с компютър за управление. Много интересен проект, който представя по опростено работата на днешните принтери. Предназначено е за всеки, който се интересува от такъв тип направи си сам проекти, от механика и софтуер. (ученици, студенти и др.) ю

**4.2. Основни етапи в реализирането на проекта (основни дейности, роли на авторите):**

Разработих този проект с помощта на моя ментор и приятел, който прояви интерес към идеята. Намерих и закупих материалите, които са ми необходими за този проект. Направих електрическа схема. Започнах да изработвам основата на принтера от дървени плоскости. Тук получих помощ от своя приятел. Когато основата беше сглобена, започнах да подготвям моторите. Моторите, които използвам за движещи се оси по Х и У са от стари DVD записвачки. Използвах още един мотор, който закупих за да повдига химикала по време на работа. След като бях подготвила всичко, започнах да работя по свързването на отделните компоненти от електрическата схема. Като последен етап свързах в системата захранването и USB кабела, който свързва платката към компютъра. След това беше частта със софтуера. Трябваше да кача всички програми, които са необходими за качване на кода, създаване на картинките и управление при печат. (Arduino Studio, Inkscape, Pronterface)

**4.3. Ниво на сложност на проекта − основни проблеми при реализация на поставените**

**цели:**

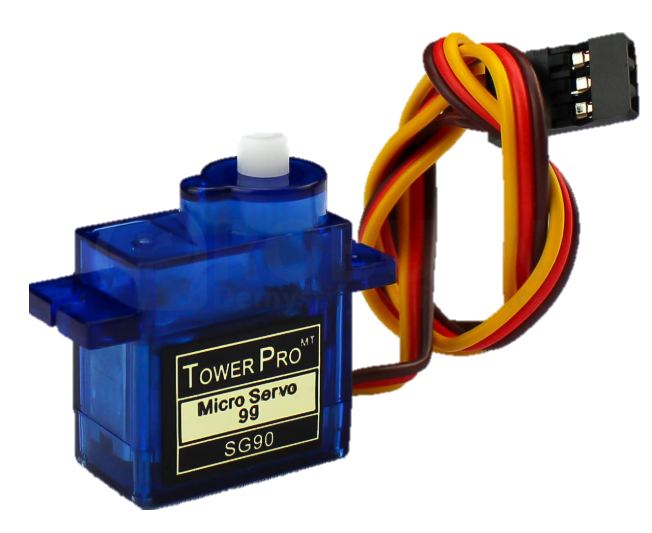
Имах доста препятствия при конструирането на физическата част на принтера. Съща така имах доста усложнения при тестването на принтера.

**4.4. Логическо и функционално описание на решението – архитектура, от какви модули е изградено, какви са функциите на всеки модул, какви са взаимодействията помежду им и т.н.:**

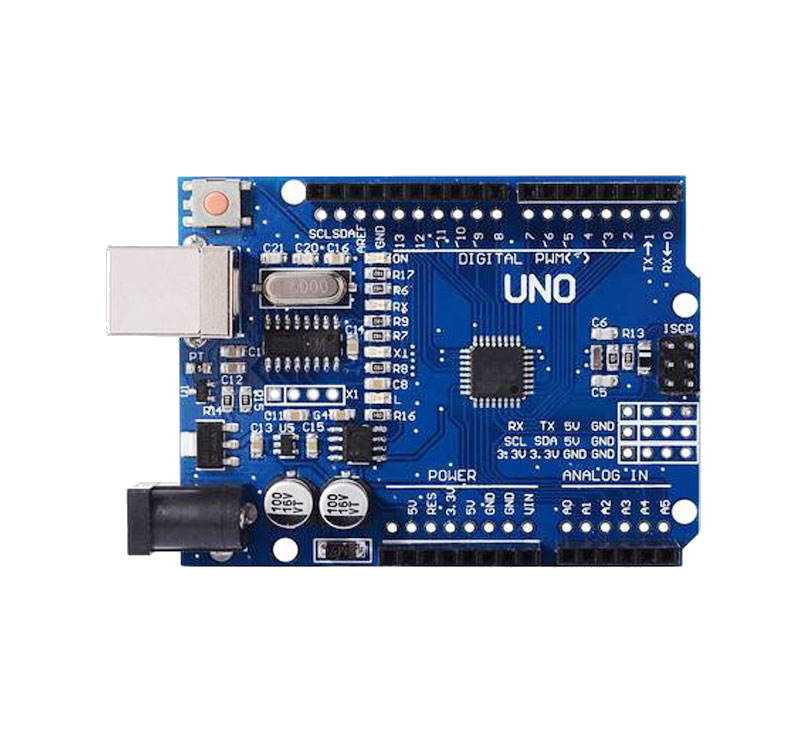
Основата на принтера е съставена от дървени плоскости. Останалите елементи на са част от електрическата схема.

# Списък от компоненти

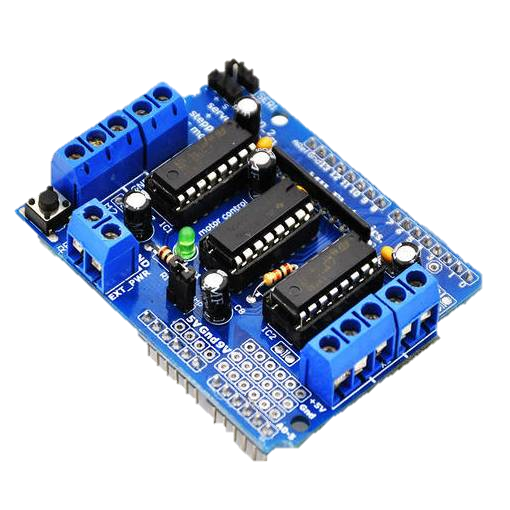
1. Arduino Uno
2. Servo Motor SG90
3. DVD Driver (x2)
4. L293D V1 motor driver shield



Фигура 2- Servo motor SG90



Фигура 1- Arduino UNO





Фигура 4- L293D V1 motor driver shield

Фигура 3- DVD записвачка (driver)

# 

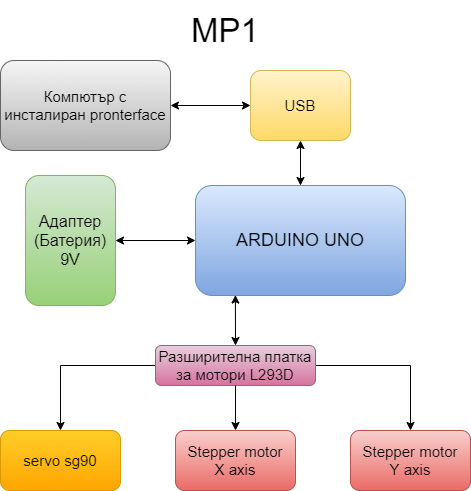
# Електрическа схема

Това е електрическата схема на проекта ни, която се състои от компонентите включени горе. Разширителната платка за мотори тъй като липсваше във Fritzing използвахме два контролера за мотори L293D, които работят по един и същ начин както разширителната платка. Всички пинове са свързани аналогично с физическия проект.

Картина, която съдържа верига

Описанието е генерирано автоматично

# Блок схема

Това е блок схемата на проекта. Батерията (адаптера) подава допълнително захранване на принтера, за да може моторите да работят нормално. MP1 се свързва към компютъра чрез USB кабел и комуникацията протича през него. Платката изпраща информация към моторите посредством разширителната платка за мотори L293D.

**4.5. Реализация − обосновка за използвани технологични средства, алгоритми,**

**литература, програмни приложения и др.:**

За вграждане на кода използвам приложението Arduino.

# Сорс код – описание на фукнционалността

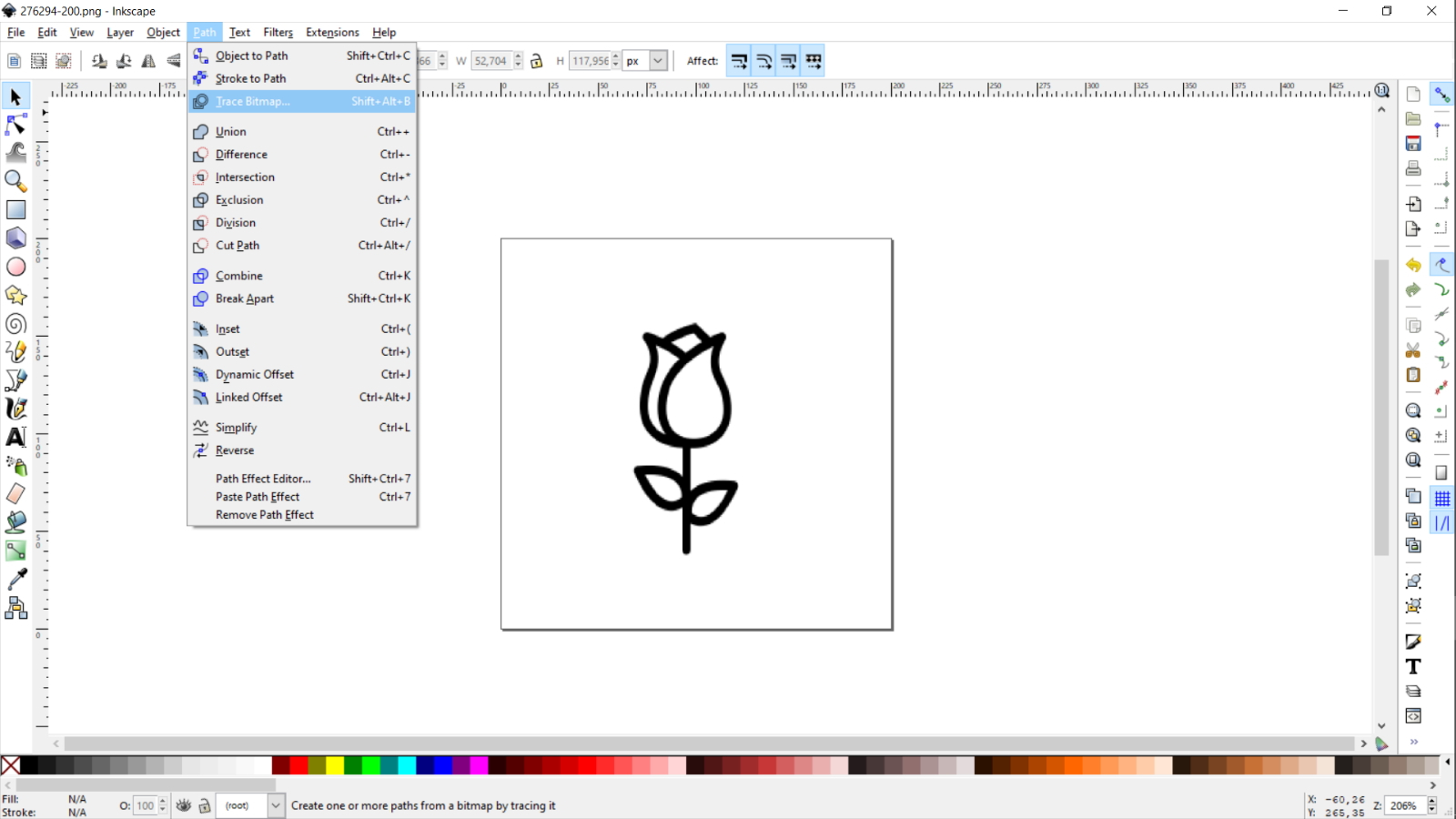
Това е целият код използван за програмиране на платката (Arduino Unо).

|  |
| --- |
| #include <Servo.h> |
|  | #include <AFMotor.h> |
|  |  |
|  | #define LINE\_BUFFER\_LENGTH 512 |
|  |  |
|  | char STEP = MICROSTEP ; |
|  |  |
|  | // Servo position for Up and Down |
|  | const int penZUp = 95; |
|  | const int penZDown = 83; |
|  |  |
|  | // Servo on PWM pin 10 |
|  | const int penServoPin =10 ; |
|  |  |
|  |  |
|  | const int stepsPerRevolution = 48; |
|  |  |
|  | // create servo object to control a servo |
|  | Servo penServo; |
|  | Инициализация на координатната система, която е по Х и Y |
|  | // Initialize steppers for X- and Y-axis using this Arduino pins for the L293D H-bridge |
|  | AF\_Stepper myStepperY(stepsPerRevolution,1); |
|  | AF\_Stepper myStepperX(stepsPerRevolution,2); |
|  |  |
|  | /\* Structures, global variables \*/ |
|  | struct point { |
|  | float x; |
|  | float y; |
|  | float z; |
|  | }; |
|  |  |
|  | // Current position of plothead |
|  | struct point actuatorPos; |
|  |  |
|  | // Drawing settings, should be OK |
|  | float StepInc = 1; |
|  | int StepDelay = 0; |
|  | int LineDelay =0; |
|  | int penDelay = 50; |
|  |  |
|  | // Motor steps to go 1 millimeter. |
|  | // Use test sketch to go 100 steps. Measure the length of line. |
|  | // Calculate steps per mm. Enter here. |
|  | float StepsPerMillimeterX = 100.0; |
|  | float StepsPerMillimeterY = 100.0; |
|  |  |
|  | // Drawing robot limits, in mm |
|  | // OK to start with. Could go up to 50 mm if calibrated well. |
|  | float Xmin = 0; |
|  | float Xmax = 40; |
|  | float Ymin = 0; |
|  | float Ymax = 40; |
|  | float Zmin = 0; |
|  | float Zmax = 1; |
|  |  |
|  | float Xpos = Xmin; |
|  | float Ypos = Ymin; |
|  | float Zpos = Zmax; |
|  |  |
|  | // Set to true to get debug output. |
|  | boolean verbose = false; |
|  |  |
|  | // Needs to interpret |
|  | // G1 for moving |
|  | // G4 P300 (wait 150ms) |
|  | // M300 S30 (pen down) |
|  | // M300 S50 (pen up) |
|  | // Discard anything with a ( |
|  | // Discard any other command! |
|  |  |
|  | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  | \* void setup() - Initialisations |
|  | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
|  | void setup() { |
|  | // Setup |
|  |  |
|  | Serial.begin( 9600 ); |
|  |  |
|  | penServo.attach(penServoPin); |
|  | penServo.write(penZUp); |
|  | delay(100); |
|  |  |
|  | // Decrease if necessary |
|  | myStepperX.setSpeed(600); |
|  |  |
|  | myStepperY.setSpeed(600); |
|  |  |
|  |  |
|  | // Set & move to initial default position |
|  | // TBD |
|  | Съобщенията, които се изписват в комуникационния прозорец на Pronterface, за състоянието на принтера |
|  | // Notifications!!! |
|  | Serial.println("Mini CNC Plotter alive and kicking!"); |
|  | Serial.print("X range is from "); |
|  | Serial.print(Xmin); |
|  | Serial.print(" to "); |
|  | Serial.print(Xmax); |
|  | Serial.println(" mm."); |
|  | Serial.print("Y range is from "); |
|  | Serial.print(Ymin); |
|  | Serial.print(" to "); |
|  | Serial.print(Ymax); |
|  | Serial.println(" mm."); |
|  | } |
|  |  |
|  | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  | \* void loop() - Main loop |
|  | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
|  | void loop() |
|  | { |
|  |  |
|  | delay(100); |
|  | char line[ LINE\_BUFFER\_LENGTH ]; |
|  | char c; |
|  | int lineIndex; |
|  | bool lineIsComment, lineSemiColon; |
|  |  |
|  | lineIndex = 0; |
|  | lineSemiColon = false; |
|  | lineIsComment = false; |
|  |  |
|  | while (1) { |
|  |  |
|  | // Serial reception - Mostly from Grbl, added semicolon support |
|  | while ( Serial.available()>0 ) { |
|  | c = Serial.read(); |
|  | if (( c == '\n') || (c == '\r') ) { // End of line reached |
|  | if ( lineIndex > 0 ) { // Line is complete. Then execute! |
|  | line[ lineIndex ] = '\0'; // Terminate string |
|  | if (verbose) { |
|  | Serial.print( "Received : "); |
|  | Serial.println( line ); |
|  | } |
|  | processIncomingLine( line, lineIndex ); |
|  | lineIndex = 0; |
|  | } |
|  | else { |
|  | // Empty or comment line. Skip block. |
|  | } |
|  | lineIsComment = false; |
|  | lineSemiColon = false; |
|  | Serial.println("ok"); |
|  | } |
|  | else { |
|  | if ( (lineIsComment) || (lineSemiColon) ) { // Throw away all comment characters |
|  | if ( c == ')' ) lineIsComment = false; // End of comment. Resume line. |
|  | } |
|  | else { |
|  | if ( c <= ' ' ) { // Throw away whitepace and control characters |
|  | } |
|  | else if ( c == '/' ) { // Block delete not supported. Ignore character. |
|  | } |
|  | else if ( c == '(' ) { // Enable comments flag and ignore all characters until ')' or EOL. |
|  | lineIsComment = true; |
|  | } |
|  | else if ( c == ';' ) { |
|  | lineSemiColon = true; |
|  | } |
|  | else if ( lineIndex >= LINE\_BUFFER\_LENGTH-1 ) { |
|  | Serial.println( "ERROR - lineBuffer overflow" ); |
|  | lineIsComment = false; |
|  | lineSemiColon = false; |
|  | } |
|  | else if ( c >= 'a' && c <= 'z' ) { // Upcase lowercase |
|  | line[ lineIndex++ ] = c-'a'+'A'; |
|  | } |
|  | else { |
|  | line[ lineIndex++ ] = c; |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } |
|  | } |
|  |  |
|  | void processIncomingLine( char\* line, int charNB ) { |
|  | int currentIndex = 0; |
|  | char buffer[ 64 ]; // Hope that 64 is enough for 1 parameter |
|  | struct point newPos; |
|  |  |
|  | newPos.x = 0.0; |
|  | newPos.y = 0.0; |
|  |  |
|  | // Needs to interpret |
|  | // G1 for moving |
|  | // G4 P300 (wait 150ms) |
|  | // G1 X60 Y30 |
|  | // G1 X30 Y50 |
|  | // M300 S30 (pen down) |
|  | // M300 S50 (pen up) |
|  | // Discard anything with a ( |
|  | // Discard any other command!  Методи за изписване на грешки, ако възникнат в процеса на работа на принтера |
|  |  |
|  | while( currentIndex < charNB ) { |
|  | switch ( line[ currentIndex++ ] ) { // Select command, if any |
|  | case 'U': |
|  | penUp(); |
|  | break; |
|  | case 'D': |
|  | penDown(); |
|  | break; |
|  | case 'G': |
|  | buffer[0] = line[ currentIndex++ ]; // /!\ Dirty - Only works with 2 digit commands |
|  | // buffer[1] = line[ currentIndex++ ]; |
|  | // buffer[2] = '\0'; |
|  | buffer[1] = '\0'; |
|  |  |
|  | switch ( atoi( buffer ) ){ // Select G command |
|  | case 0: // G00 & G01 - Movement or fast movement. Same here |
|  | case 1: |
|  | // /!\ Dirty - Suppose that X is before Y |
|  | char\* indexX = strchr( line+currentIndex, 'X' ); // Get X/Y position in the string (if any) |
|  | char\* indexY = strchr( line+currentIndex, 'Y' ); |
|  | if ( indexY <= 0 ) { |
|  | newPos.x = atof( indexX + 1); |
|  | newPos.y = actuatorPos.y; |
|  | } |
|  | else if ( indexX <= 0 ) { |
|  | newPos.y = atof( indexY + 1); |
|  | newPos.x = actuatorPos.x; |
|  | } |
|  | else { |
|  | newPos.y = atof( indexY + 1); |
|  | indexY = '\0'; |
|  | newPos.x = atof( indexX + 1); |
|  | } |
|  | drawLine(newPos.x, newPos.y ); |
|  | // Serial.println("ok"); |
|  | actuatorPos.x = newPos.x; |
|  | actuatorPos.y = newPos.y; |
|  | break; |
|  | } |
|  | break; |
|  | case 'M': |
|  | buffer[0] = line[ currentIndex++ ]; // /!\ Dirty - Only works with 3 digit commands |
|  | buffer[1] = line[ currentIndex++ ]; |
|  | buffer[2] = line[ currentIndex++ ]; |
|  | buffer[3] = '\0'; |
|  | switch ( atoi( buffer ) ){ |
|  | case 300: |
|  | { |
|  | char\* indexS = strchr( line+currentIndex, 'S' ); |
|  | float Spos = atof( indexS + 1); |
|  | // Serial.println("ok"); |
|  | if (Spos == 30) { |
|  | penDown(); |
|  | } |
|  | if (Spos == 50) { |
|  | penUp(); |
|  | } |
|  | break; |
|  | } |
|  | case 114: // M114 - Repport position |
|  | Serial.print( "Absolute position : X = " ); |
|  | Serial.print( actuatorPos.x ); |
|  | Serial.print( " - Y = " ); |
|  | Serial.println( actuatorPos.y ); |
|  | break; |
|  | default: |
|  | Serial.print( "Command not recognized : M"); |
|  | Serial.println( buffer ); |
|  | } |
|  | } |
|  | } |
|  |  |
|  |  |
|  |  |
|  | } |
|  |  |
|  |  |
|  | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* |
|  | \* Draw a line from (x0;y0) to (x1;y1). |
|  | \* Bresenham algo from https://www.marginallyclever.com/blog/2013/08/how-to-build-an-2-axis-arduino-cnc-gcode-interpreter/ |
|  | \* int (x1;y1) : Starting coordinates |
|  | \* int (x2;y2) : Ending coordinates |
|  | \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
|  | void drawLine(float x1, float y1) { |
|  |  |
|  | if (verbose) |
|  | { |
|  | Serial.print("fx1, fy1: "); |
|  | Serial.print(x1); |
|  | Serial.print(","); |
|  | Serial.print(y1); |
|  | Serial.println(""); |
|  | } |
|  |  |
|  | // Bring instructions within limits |
|  | if (x1 >= Xmax) { |
|  | x1 = Xmax; |
|  | } |
|  | if (x1 <= Xmin) { |
|  | x1 = Xmin; |
|  | } |
|  | if (y1 >= Ymax) { |
|  | y1 = Ymax; |
|  | } |
|  | if (y1 <= Ymin) { |
|  | y1 = Ymin; |
|  | } |
|  |  |
|  | if (verbose) |
|  | { |
|  | Serial.print("Xpos, Ypos: "); |
|  | Serial.print(Xpos); |
|  | Serial.print(","); |
|  | Serial.print(Ypos); |
|  | Serial.println(""); |
|  | } |
|  |  |
|  | if (verbose) |
|  | { |
|  | Serial.print("x1, y1: "); |
|  | Serial.print(x1); |
|  | Serial.print(","); |
|  | Serial.print(y1); |
|  | Serial.println(""); |
|  | } |
|  | Конвертиране на координатите в милиметри |
|  | // Convert coordinates to steps |
|  | x1 = (int)(x1\*StepsPerMillimeterX); |
|  | y1 = (int)(y1\*StepsPerMillimeterY); |
|  | float x0 = Xpos; |
|  | float y0 = Ypos; |
|  |  |
|  | // Let's find out the change for the coordinates |
|  | long dx = abs(x1-x0); |
|  | long dy = abs(y1-y0); |
|  | int sx = x0<x1 ? StepInc : -StepInc; |
|  | int sy = y0<y1 ? StepInc : -StepInc; |
|  |  |
|  | long i; |
|  | long over = 0; |
|  |  |
|  | if (dx > dy) { |
|  | for (i=0; i<dx; ++i) { |
|  | myStepperX.onestep(sx,STEP); |
|  | over+=dy; |
|  | if (over>=dx) { |
|  | over-=dx; |
|  | myStepperY.onestep(sy,STEP); |
|  | } |
|  | delay(StepDelay); |
|  | } |
|  | } |
|  | else { |
|  | for (i=0; i<dy; ++i) { |
|  | myStepperY.onestep(sy,STEP); |
|  | over+=dx; |
|  | if (over>=dy) { |
|  | over-=dy; |
|  | myStepperX.onestep(sx,STEP); |
|  | } |
|  | delay(StepDelay); |
|  | } |
|  | } |
|  |  |
|  | if (verbose) |
|  | { |
|  | Serial.print("dx, dy:"); |
|  | Serial.print(dx); |
|  | Serial.print(","); |
|  | Serial.print(dy); |
|  | Serial.println(""); |
|  | } |
|  |  |
|  | if (verbose) |
|  | { |
|  | Serial.print("Going to ("); |
|  | Serial.print(x0); |
|  | Serial.print(","); |
|  | Serial.print(y0); |
|  | Serial.println(")"); |
|  | } |
|  |  |
|  | // Delay before any next lines are submitted |
|  | delay(LineDelay); |
|  | // Update the positions |
|  | Xpos = x1; |
|  | Ypos = y1; |
|  | } |
|  | Методи за работния цикъл на химикала |
|  | // Raises pen |
|  | void penUp() { |
|  | penServo.write(penZUp); |
|  | delay(penDelay); |
|  | Zpos=Zmax; |
|  | digitalWrite(15, LOW); |
|  | digitalWrite(16, HIGH); |
|  | if (verbose) { |
|  | Serial.println("Pen up!"); |
|  |  |
|  | } |
|  | } |
|  | // Lowers pen |
|  | void penDown() { |
|  | penServo.write(penZDown); |
|  | delay(penDelay); |
|  | Zpos=Zmin; |
|  | digitalWrite(15, HIGH); |
|  | digitalWrite(16, LOW); |
|  | if (verbose) { |
|  | Serial.println("Pen down."); |
|  |  |
|  |  |
|  | } |
|  | } |

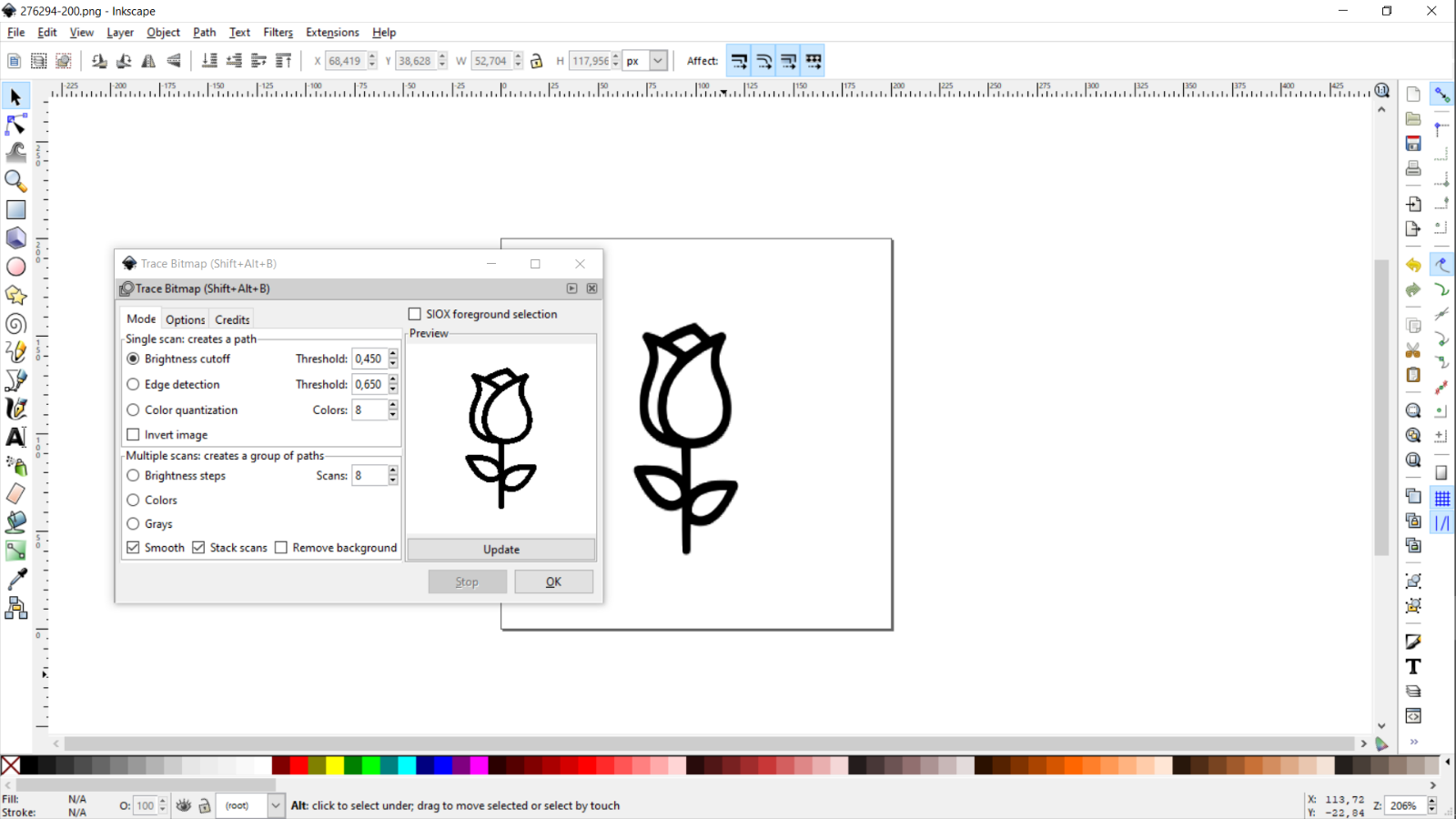
**Използвана литература:**

* [**https://create.arduino.cc/projecthub/mega-das/arduino-cnc-plotter-drawing-machine-a73ea2**](https://create.arduino.cc/projecthub/mega-das/arduino-cnc-plotter-drawing-machine-a73ea2)

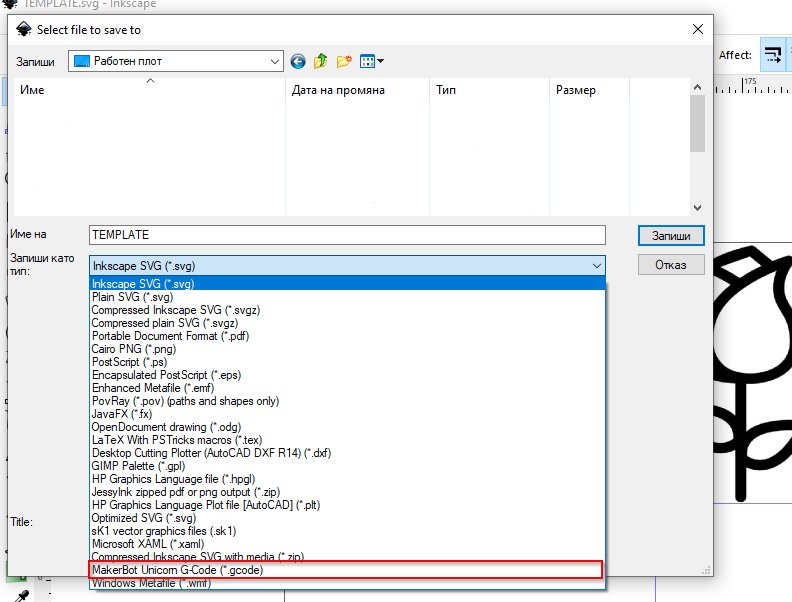
За да изпечатаме избраната от нас картинка първо трябва да се обработи в gcode формат, за което използваме Inkscape. След като отворим снимката в програмата трябва зададем пътя на изчертаване на картинката .



След като изберем тази настройка на пътя се отваря прозорче, в което само потвърждаваме.



След като вмъкнем снимката, която искаме да форматираме, даваме Save As и избираме формат gcode. Готови сме и можем да преминем към следващото приложение.



**4.6. Описание на приложението – как се стартира и/или инсталира, как се използва, как се поддържа:**

Свързваме принтера с компютъра чрез USB кабел и включваме захранването. Софтуера, който осъществява връзката (комуникацията) между принтера и компютъра, е Pronterface. На него се качва предварително форматираната снимка в gcode формат и чрез него се изпраща на принтера. Свързваме принтера и печатаме.

Картина, която съдържа текст

Описанието е генерирано автоматично

Легенда:

1 – ръчно командване на моторите (вид тестване дали моторите работят нормално)

2 – свързване с принтера

3 – качване на снимка (в gcode формат)

4 – дава се начало на принтирането

5 – работно платно, което показва качената снимка, която ще бъде принтирана

6 – комуникационен прозорец (изписва се състоянието на принтера и се изписват грешки, ако възникнат някакви в процеса на работа на принтера)

Консумативите на принтера са листи хартия и химикал, които използва за да рисува.

**4.7. Заключение – какъв е основният резултат, дали има приложения до момента, какви възможности съществуват за развитие и усъвършенстване:**

Създадох принтер, който е предназначен основно за принтиране на джобни картинки. До момента работата му е сравнително добра, като за толкова бюджетно устройство направено от подръчни материали.

Трябва да отбележа, че както всеки проект така и този има някои недостатъци:

* Моторите и адаптера, който служи за подаване на допълнително електричество, загряват много бързо
* Трудна е смяната на химикала
* Има малък обсег на чертеж, ако потребителят иска да принтира снимка, в по – голям размер е невъзможно.

Въпреки тези недостатъци, смятам, че проектът има и перспективи за развитие:

* Може да се добави възможност да се принтира от телефон, чрез Bluetooth или WIFI връзка.
* Подобрен дизайн – може да се добави място за съхранение на листове