POLITECHNIKA POZNAŃSKA

WYDZIAŁ ELEKTRYCZNY

Podstawy Teleinformatyki

Dokumentacja projektu plottera CD-ROM

Damian Michalski 126880
damian.michalski@student.put.poznan.pl

Piotr Zimny 126876
piotr.zimny@student.put.poznan.pl

Prowadzący:
mgr inż. Przemysław Walkowiak

Poznań, 2018

Spis treści

1. Charakterystyka ogólna projektu

Projekt zakłada:

- budowę urządzenia rysującego grafiki w postaci wektorowej, które będzie elastyczne i dostosowywalne do wymagań użytkownika. Grafiki przekazywane są z wykorzystaniem portu szeregowego z komputera do urządzenia sterującego procesem.
- główne elementy urządzenia odzyskane z recyklingu urządzeń elektromechanicznych takich jak drukarki i stacje CD-ROM/DVD (mechanizm CD-ROM, krańcówki, przewody),
- jednostka sterująca silnikami krokowymi, serwomotorem, i sterownikiem silników krokowych - wykorzystano mikrokomputer Raspberry Pi,
- urządzenie będzie umożliwiało rysowanie grafik na stabilnych płaskich powierzchniach np. kartka, deska, obudowa, podłoga,
- wymienne narzędzie rysujące (ołówek, długopis, flamaster).

Główna zasada działania:

- Pozycja narzędzia rysującego ustawiana za pomocą dwóch mechanizmów CD-ROM, zamontowanych „na krzyż”, jedno nad drugim, aby zapewnić możliwość ruchu w dwóch osiach X oraz Y.
  Urządzenie musi przed każdą nową operacją rysowania przejść proces autokalibracji, który pozwoli na powrót obu osi rysujących do punktów startowych.
- Sterowanie mechanizmami realizowane za pomocą sterowników silnika krokowego A4988 podłączonych pod porty GPIO minikomputera Raspberry Pi.
- Obraz w postaci wektorowej utworzony np. za pomocą programu InkScape będzie eksportowany do formy G-Code, czyli znormalizowanego języka zapisu poleceń dla urządzeń CNC.
- Interpreter przetwarza zapis w postaci języka G-Code, zamieniając każdą linię na rozkaz wykonania odpowiedniego ruchu przez urządzenie np. ruchu po linii prostej na określonej długości lub ruchu po okręgu o podanych parametrach.

2. Powód wybrania tematu

- Zainteresowanie elektroniką i technologiami leżącymi na skraju software/hardware,
- Możliwość zautomatyzowania tworzenia grafik,

- Poznanie zasady działania maszyny CNC,
- Zapoznanie z G-Code – perspektywa budowy drukarki 3D(RepRap, MakerBot) lub frezarki CNC,
- Poszerzenie wiedzy z dziedziny oprogramowania mikrokomputerów, systemów wbudowanych, efektywnego zarządzania pamięcią, programowania wielowątkowego w systemach z jądrem systemu Linux,
- Poszerzenie wiedzy i umiejętności z zakresu projektowania i programowania urządzeń elektronicznych oraz sterowania współbieżnego silnikami.

3. Główna idea działania

Urządzenie pozwala na rysowanie obiektów graficznych, rysunków jedynie w określonym formacie. Każdy obraz, który ma zostać przetworzony musi mieć postać wektorową. Obrazy tworzone są w programie Inkscape.

Program **Inkscape** jest profesjonalnym narzędziem używanym do przygotowywania grafik wektorowych, działa na systemach Windows, Mac OS X i Linux. Jest wykorzystywany przez profesjonalnych grafików jak i hobbystów na całym świecie do kreowania szerokiej gamy grafik, takich jak ilustracje, ikony, loga, diagramy, mapy i grafiki serwisów internetowych. Program używa otwartego standardu W3C jakim jest **SVG**(Scalable Vector Graphics) jako natywnego formatu, co więcej jest w pełni darmowym oprogramowaniem.

To właśnie za pomocą programu Inkscape przygotowywane są grafiki wektorowe, które następnie są konwertowane do postaci **G-Code**. Aby poprawnie przygotować grafikę do wyrysowania, niezbędne jest skonfigurowanie obszaru graficznego o wymaganych rozmiarach (takich, jakie są osiągalne dla osi rysujących) np. 2cm : 4cm i ustawienie skali na centymetry.
Drukowany może być tekst jak i obrazy. Do poprawnego drukowania niezbędne w przypadku obrazów jest przezroczyste tło. Grafika wektorowa jest zamieniana na konkretne ścieżki i linie.

Przygotowany w ten sposób obrazek należy wyeksportować do postaci G-Code, również z poziomu programu Inkscape. Otrzymujemy bezpośrednio plik z liniami kodu rozumianego przez maszyny CNC, które reprezentują ruch po osiach X, Y, Z.
Następnie wygenerowany G-Code musi zostać wczytany do interpretera znajdującego się bezpośrednio na urządzeniu sterującym. Kod jest interpretowany linia po linii przez interpreter, a każda z nich zostaje zamieniona na odpowiadające jej konkretne ruchy narzędzia w układzie współrzędnych wraz z podnoszeniem i opuszczaniem narzędzia (ruch w osi Z). Realizacja ruchów w odpowiedniej kolejności przez sterownik generuje ruch narzędzia, którego efektem jest założony

rysunek.

4. Wybrane technologie i architektury

Raspberry Pi

Platforma komputerowa stworzona przez Raspberry Pi Foundation. Urządzenie składa się z pojedynczego układu drukowanego i zostało wymyślone, by wspierać naukę podstaw informatyki.
Raspberry Pi działa pod kontrolą systemów operacyjnych opartych na Linuksie oraz RISC OS, a od modelu Raspberry Pi 2 B, działa również pod kontrolą Windows 10 Internet Of Things.

GPIO

GPIO (od ang. *general-purpose input/output*) – pin interfejsu do komunikacji między elementami systemu komputerowego (na przykład między mikroprocesorem a urządzeniami peryferyjnymi). Wyprowadzenia takie mogą pełnić rolę zarówno wejść, jak i wyjść i jest to zwykle właściwość konfigurowalna. Są one często grupowane w porty. Urządzenia korzystające z interfejsu GPIO mają możliwość zgłaszania przerwań w jednostce centralnej i używania bezpośredniego dostępu do pamięci (DMA - Direct Memory Access).

Java

współbieżny, oparty na klasach, obiektowy język programowania ogólnego zastosowania. Został stworzony przez grupę roboczą pod kierunkiem Jamesa Goslinga z firmy Sun Microsystems. Java jest językiem tworzenia programów źródłowych kompilowanych do kodu bajtowego, czyli postaci wykonywanej przez maszynę wirtualną. Język cechuje się silnym typowaniem. Jego podstawowe koncepcje zostały przejęte z języka Smalltalk (maszyna wirtualna, zarządzanie pamięcią) oraz z języka C++ (duża część składni i słów kluczowych).

Grafika wektorowa

Jeden z dwóch podstawowych rodzajów grafiki komputerowej, w której obraz opisany jest za pomocą figur geometrycznych (w przypadku grafiki dwuwymiarowej) lub brył geometrycznych (w przypadku grafiki trójwymiarowej), umiejscowionych w matematycznie zdefiniowanym układzie współrzędnych, odpowiednio dwu- lub trójwymiarowym. Drugim z podstawowych typów grafiki komputerowej jest grafika

rastrowa. Grafiką wektorowa różni się od rastrowej w użyciu głównie możliwością bezstratnego skalowania, oraz możliwością druku, wypalania, czy wycinania za pomocą specjalistycznych maszyn.

Gcode

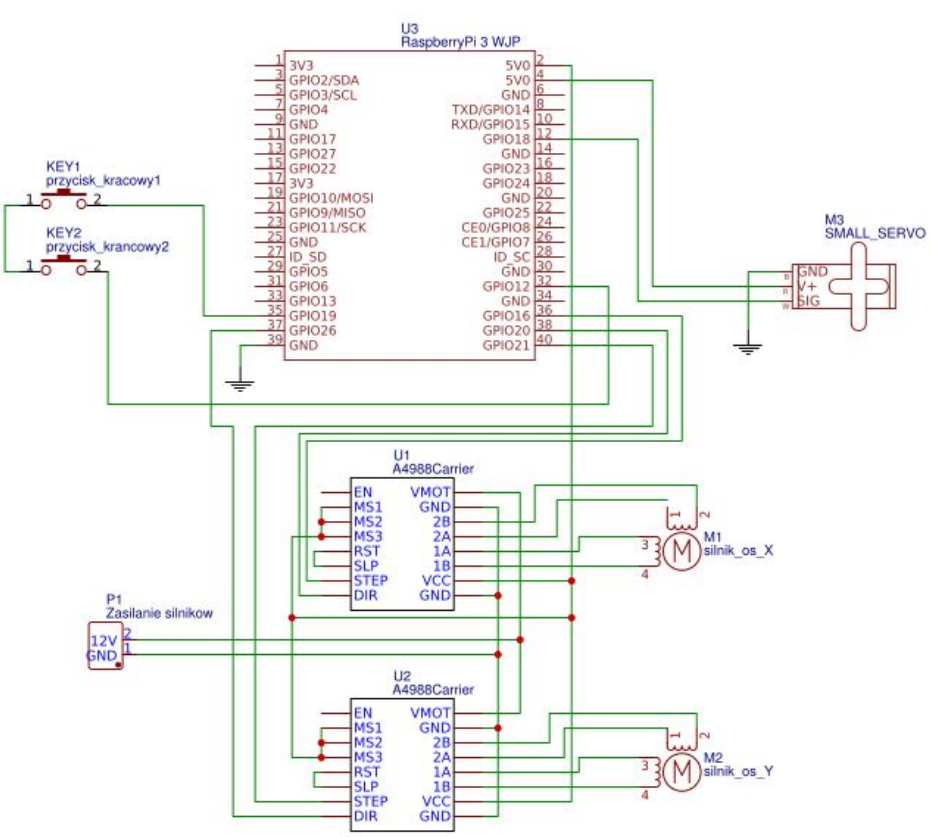
Znormalizowany język zapisu poleceń dla urządzeń CNC. Definiuje podstawowe operacje, które należy wykonać, aby obrobić detal na obrabiarce sterowanej numerycznie.

5. Wymagane części elektroniczne

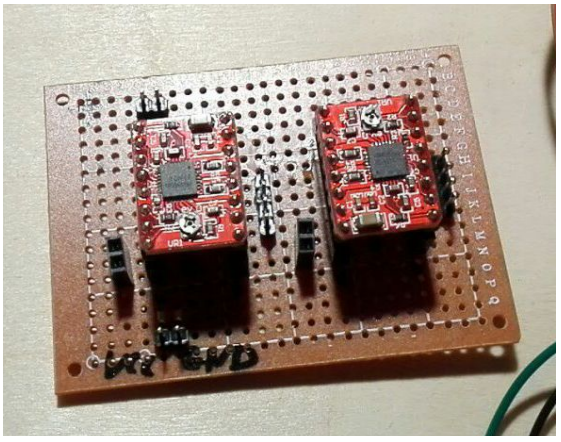
- dwa napędy cd-rom z silnikiem krokowym bipolarnym
- dwa sterowniki silników krokowych A4988
- serwomechanizm
- mikrokomputer z obsługą GPIO i systemem linux (np. Raspberry Pi Zero)
- prototypowa płytka drukowana
- przewody połączeniowe
- ogniwa 18650
- ładowarka ogniw 18650

5. Schemat układu

Schemat układu został zaprojektowany z pomocą portalu easyeda.com. Realizacja została przeprowadzona na płytce prototypowej. Pozwoliło to na przyspieszenie procesu budowy oraz obniżenie kosztów.



Rys. 1 - Schemat układu



Rys. 2 - Płytka PCB z układami A4988

6. Silniki krokowe, sterowanie

Głównym założeniem projektowym było zastosowanie mechanizmów napędów CD rom zbudowanych na bazie silników krokowych. Główną cechą silników krokowych jest sposób działania, który nie polega na obracaniu się ruchem ciągłym jak to występuje w silnikach DC, ale o wykonywanie ruchów obrotowych o mały, ściśle określony kąt. Jest to realizowane za pomocą cewek oraz magnesów trwałych. Taka charakterystyka działania pozwala na bardzo precyzyjne określanie

liczby obrotów jakie wykonał silnik oraz w jakim kierunku. Stosowane silniki krokowe w mechanizmach napędów CD pozwalają na bardzo precyzyjne określanie pozycji tacki, zatem stosując ten mechanizm jako oś, można precyzyjnie pozycjonować narzędzie rysujące.

W świecie elektroniki występuje wiele typów silników krokowych. Różnią się one liczbą cewek, magnesów, sposobem połączeń cewek oraz liczbą wyprowadzeń. Każdy z rodzajów silników posiada inne charakterystyczne cechy oraz wymagania. Zależnie od potrzeby stosuje się wybrane typy. Najpopularniejszymi silnikami krokowymi są:

- silniki bipolarne
- silniki unipolarne

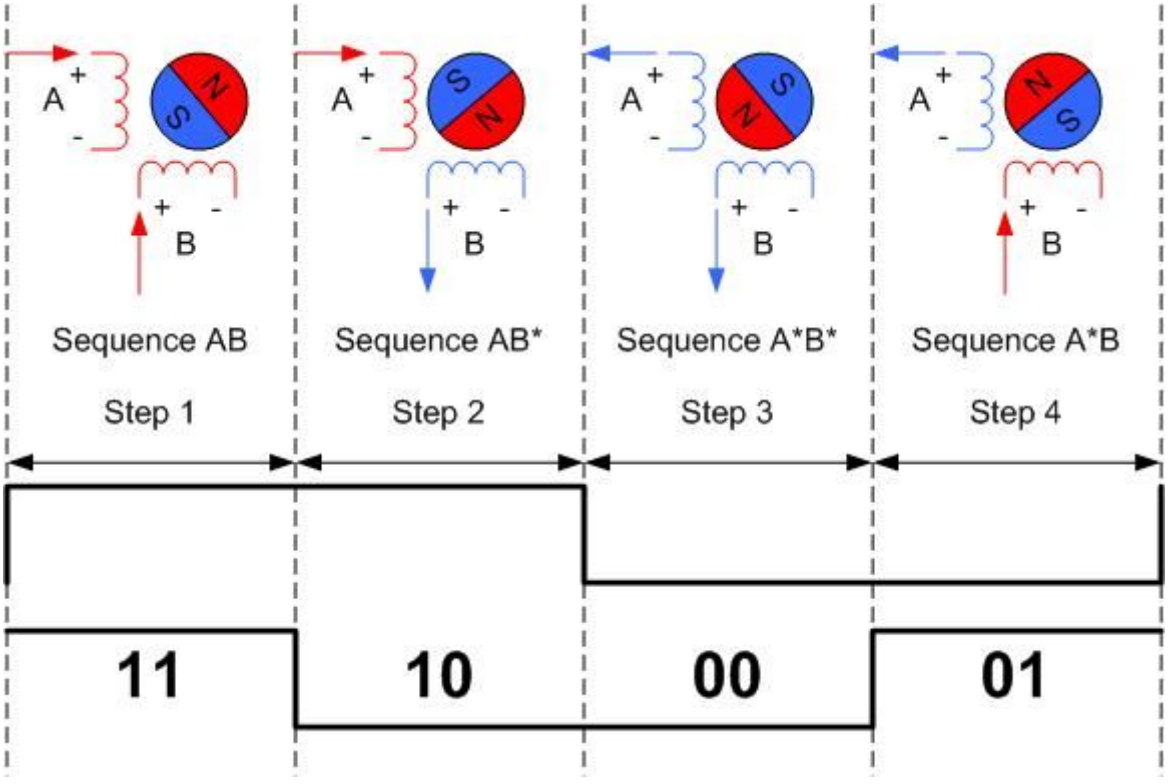
Każdy z powyższych może posiadać różną liczbę wyprowadzeń oraz różny sposób połączeń cewek wewnątrz obudowy. Najprostszym i najczęściej spotykanym silnikiem krokowym jest silnik bipolarny z 4 wyprowadzeniami. Charakteryzuje się on uniwersalnością w zastosowaniach ze względu na to, że każda z cewek może być sterowana jest osobno co pozwala na dowolne jej wykorzystanie oraz zasilanie. W przypadku silników unipolarnych występuje 5 wyprowadzeń (najprostszy przypadek). Jedno z nich jest punktem wspólnym połączonych ze sobą cewek według punktu środkowego każdej z cewki. Wśród wyżej wymienionych rodzajów można zaobserwować także silniki bipolarne z ośmioma wyprowadzeniami, gdzie każda para odpowiada za osobną cewkę oraz silniki unipolarne z 6 wyprowadzeniami, gdzie punkt środkowy każdej z cewki wyprowadzony jest osobno.

Charakterystyka silników krokowych nie pozwala wykorzystywać ich jako napęd w różnego rodzaju mechanizmach, gdzie opory ruchu są zmienne i niemożliwe do przewidzenia, ale z powodu bardzo precyzyjnych ruchów obrotowych mają znaczną przewagę nad silnikami DC i dlatego stosuje się je w precyzyjnych obrabiarkach i urządzeniach gdzie obciążenie silników jest znane oraz stałe. Silniki krokowe dzięki swojej budowie nie wymagają dodatkowych mechanizmów pomiaru pozycji osi, ponieważ jeżeli liczba wykonanych kroków jest znana oraz wiadomo jaki stopień obrotu przypada na jeden krok, to można ustalić bardzo precyzyjną pozycję wału silnika. Jeżeli silnik został wykorzystany jako napęd prowadnicy czy innego mechanizmu przesuwnego, to pozycja w jakiej znajduje się prowadnica jest znana z bardzo dużą dokładnością bez dodawania czujników czy enkoderów.

W przeciwieństwie do tradycyjnych silników DC, silniki krokowe wymagają specjalnej sekwencji sterującej. Zwykle wykorzystuje się do tego celu specjalnie moduły, które posiadają wbudowane mechanizmy oraz zabezpieczenia ułatwiające pracę z silnikami. Przykładowym sterownikiem silników krokowych jest moduł zbudowany na bazie układu A4988. Moduł pozwala na sterowanie małymi i średnimi

silnikami krokowymi, które jako napięcie zasilania wykorzystują od 8V do 24V. Mimo, że silniki krokowe wymagają odpowiedniej sekwencji napięć na cewkach, to dzięki temu układowi sterowanie sprowadza się do generowania impulsu “step”, który odpowiada za wykonanie kroku, oraz impulsu “dir”, który odpowiada za wybór kierunku obrotu. Silniki krokowe w mniejszych aplikacjach mogą być także sterowane manualnie poprzez mostki H zbudowane na bazie tranzystorów. Zaletą takiego sposobu sterowania jest możliwość sterowania silnikami niskonapięciowymi oraz dowolna dokładność kroku. Dodatkowym atutem tego rozwiązania są koszty podzespołów. Wadą tego rozwiązania jest znacznie bardziej skomplikowany proces sterowania silnikami.

Praca z silnikami krokowymi wymaga wiedzy na temat ich budowy oraz charakterystyk działania elektromagnesów. Aby silnik pracował poprawnie należy generować odpowiednie napięcia na poszczególnych jego cewkach. Sekwencja tych sygnałów może się różnić w zależności od typu silnika. Najprostszym przykładem sterowania silnikiem jest silnik bipolarny z czterema wyprowadzeniami. Sekwencja napięć na wyprowadzeniach przedstawiona jest na poniższym rysunku:



Rys. 3 - Sekwencja sterowania silnikiem bipolarnym

Na rysunku widoczne są 4 kolejne stany silnika w sekwencji. Oś silnika posiada magnes stały który wykonuje ruchy w zależności od stanu cewek A oraz B. W kroku nr 1 (Step 1) zasilane są obie cewki z taką sama polaryzacja. Dzięki temu obie cewki stają się elektromagnesem, który przyciąga biegun S osi powodując, że zmienia on

swoje położenie na centralne pomiędzy obiema cewkami. Kolejnym krokiem jest zmiana polaryzacji zasilania na cewce B.Powoduje to odepchnięcie bieguna S oraz przyciągnięcie bieguna N osi. Takie zachowanie osi nazywane jest krokiem. Aby wykonać kolejny krok, należy zmienić polaryzację zasilania cewki A co spowoduje odepchnięcie bieguna S osi oraz przyciągnięcie bieguna N. Takie działanie spowoduje, że oś wykona ruch o kolejny krok. Następnie zmienia się polaryzację cewki B. Dzięki temu zostanie odepchnięty biegun N oraz przyciągnięty biegun S. Spowoduje to ruch osi o kolejny kąt równy jednemu krokowi. Po tej operacji sekwencja zostaje zapętlona i powtarzana. Przedstawiona tutaj sekwencja ruchów nazywana jest “half step” lub “pół krok”. Nie jest to jedyny sposób sterowania silnikami krokowymi. Jest wiele trybów sterowania i różnią się one dokładnością ruchów oraz kątem obrotu przypadającego na jeden krok. Przykładowe sekwencje ruchów to:

- “full step” - ruch co cały krok. Silnik wykonuje 2 razy mniej kroków na obrót niż w przypadku “half step”,
- “¼ step” - ruch co ¼ kroku. Silnik wykonuje 2 razy więcej kroków niż w przypadku “half step” oraz 4 razy więcej kroków niż w przypadku “full step”,
- “⅛ step” - ruch co ⅛ kroku,
- “1/16 step” - ruch co 1/16 kroku.

Od sekwencji napięć zależna jest liczba kroków jakie może wykonać silnik. Im mniejszy krok tym więcej ich może wykonać. Do trybu ¼ step i więcej należy stosować napięcie modulowane PWM. Wszystkie pozycje silnika pomiędzy cewkami mogą być ustalane za pomocą wypełnienia tego sygnału. Odpowiednia sekwencja wypełnień PWM może zapewnić nawet 1/32 kroku standardowego. Wszystkie sekwencje, które wykorzystują PWM nazywane są mikrokrokami.

W realizacji projektu plottera zostały zastosowane silniki krokowe bipolarne. Jest to najpopularniejszy typ silnika montowany przez producentów napędów CD. Do sterowania tymi silnikami zastosowano najmniejsze możliwe dla wykorzystanego modułu kroki 1/16. Dzięki temu otrzymano znacznie większą precyzję urządzenia co pozwala na rysowanie bardzo szczegółowych rysunków.

7. Ogniwa 18650

Głównym zasilaniem całego systemu są ogniwa li-ion 18650. Są to powszechnie stosowane akumulatory w bateriach laptopowych, samochodach elektrycznych, power wallach, robotach, UPS, rowerach elektrycznych, skuterach elektrycznych, latarkach, e-papierosach, elektronarzędziach i wiele innych. Popularność tego typu zasilania jest spowodowana tym, że akumulatory tego typu są stosunkowo tanie, mają ustandaryzowane wymiary, charakteryzują się bardzo dobrymi parametrami pojemności oraz wydajności prądowej (wartości różnią się pomiędzy różnymi modelami, zależnie od przeznaczenia), napięcie naładowanego

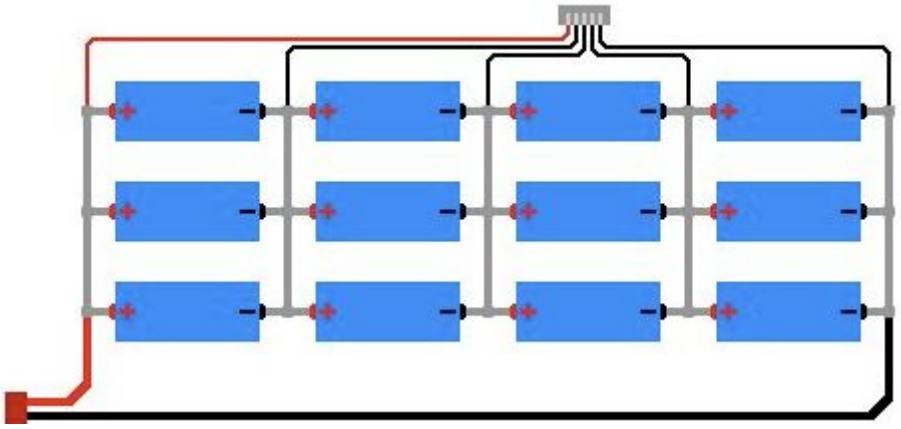
ogniwa waha się w przedziale od 2,5v (rozładowane) do 4,2v (w pełni naładowane), nieskomplikowany proces ładowania, duża żywotność. Akumulatory 18650 z powodu swojej popularności są łatwo dostępne jako część pozyskiwana z demontażu. Ogniwa zastosowane w tym projekcie pochodzą z uszkodzonej baterii laptopa.

Charakterystyczną cechą akumulatorów 18650 jest proces ładowania, który mimo, że nie jest skomplikowany, to wymaga odpowiednich procedur. Ładowanie przebiega w dwóch etapach, gdzie w pierwszym ładuje się ogniwa stałym prądem (CC - constant current), który zwykle wynosi około 1A. Natężenie dla różnych typów ogniw może nieznacznie odbiegać od tej wartości, a dokładne informacje można znaleźć w nocie katalogowej danego modelu. Drugim etapem ładowania ogniw jest tryb ładowania stałym napięciem, który wynosi zwykle 4,2V. W czasie tego procesu monitoruje się prąd przepływający przez ogniwo. Naładowanie ogniwa oznacza spadek prądu poniżej ustalonej wartości, która zwykle wynosi około 50mA. Każde ogniwo może nieznacznie różnić się napięciem ładowania oraz prądem przy pełnym naładowaniu, a jest to zależne od typu akumulatora oraz jego parametrów. Dodatkowy etap ładowania wykonuje się w przypadku ogniw, które zostały rozładowane poniżej wartości krytycznej, która zwykle wynosi około 2,5V. Taki tryb polega na ładowaniu ogniwa bardzo niskim prądem, który wynosi zwykle około 100mA. Kiedy ogniwo osiągnie napięcie powyżej 2,5V proces przechodzi w typowy tryb ładowania.

Ogniwa 18650 charakteryzują się stosunkowo dużą pojemnością w stosunku do swojej wagi. Typowa pojemność wynosi od 1500mAh do nawet 3500mAh. Takie parametry pozwalają stosować je w urządzeniach, gdzie liczy się czas działania oraz ciężar.

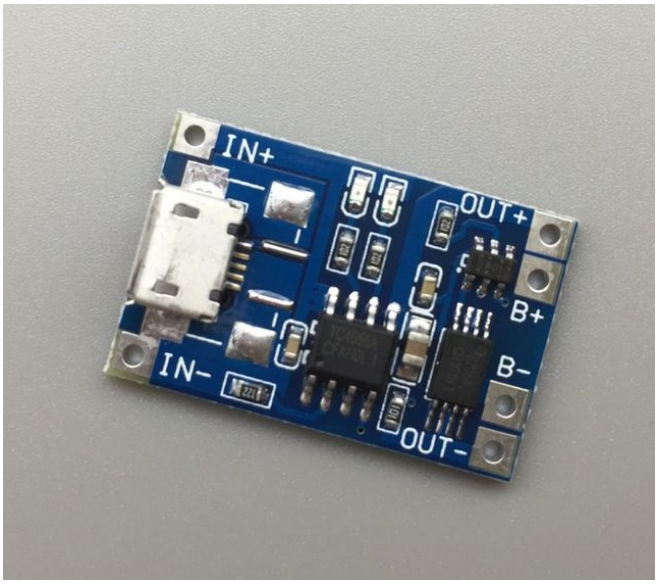
Różne modele baterii charakteryzują się różnymi pojemnościami, ale także wydajnością prądową. Zwykłe ogniwa mogą wygenerować kilka amperów, a odmiany specjalistyczne stosowane głównie w elektronarzędziach mogą wygenerować prąd o wartości nawet kilkunastu amperów.

Cenioną cechą ogniw litowo jonowych jest to, że z łatwością można je łączyć w pakiety. Mimo, że każde ogniwo musi być ładowane w charakterystyczny sposób, to łączenie ogniw w większe moduły nie stanowi problemu przy zastosowaniu BMS (Battery Management System), którego zadaniem jest nadzorowanie parametrów poszczególnych segmentów pakietu akumulatorów. Schemat przykładowego połączenia ogniw przedstawia poniższa grafika:



Rys. 4 - Przykładowe podłączenie pakietowe ogniw 18650

Ładowanie ogniw zwykle wykonuje się za pomocą specjalnych ładowarek. W celu naładowanie pojedynczego ogniwa lub kilku połączonych równolegle najprostszą metodą jest wykorzystanie dedykowanych modułów ładowania. Jako źródło zasilania wykorzystują one bardzo popularne ładowarki telefonów. Urządzenia wykorzystane w tym projekcie przedstawione są na poniższej grafice:



Rys. 5 - Ładowarka Li-ion/Li-ion

Dodatkowym atutem zastosowania dedykowanych rozwiązań jest sprzętowe zabezpieczenie przed przeładowaniem oraz zbytnim rozładowaniem dzięki czemu można mieć pewność, że ogniwa nie zostaną uszkodzone.

8. Narzędzia programistyczne

- Raspberry Pi Zero - zarządzanie ruchami, wydawanie rozkazów do sterowników silnika,

- Język Java, Netbeans IDE (zdalne uruchamianie programu, wbudowany klient protokołów SSH oraz SFTP),
- program InkScape do tworzenia grafik wektorowych oraz konwersji do G-Code,
- biblioteka Pi4J (obsługa GPIO i PWM)

9. Interesujące problemy i rozwiązania

Podczas projektowania układu oraz oprogramowania należało rozwiązać kilka problemów. Pierwszym z nich jest pisanie kodu aplikacji, który na mikrokomputerach jest znacznie utrudniony. Mimo, że można podłączyć wyświetlacz, klawiaturę i myszkę, to praca na tego typu urządzeniu jako środowisku programistycznym jest bardzo nieergonomiczna i męcząca. Głównym zadaniem mikrokomputera jest uruchomienie skompilowanego programu dlatego można było przenieść tworzenie kodu na komputer z odpowiednio skonfigurowanym środowiskiem. Środowisko NetBeans IDE umożliwia zdalną kompilację projektu co pozwoliło na wygodną i efektywną pracę nad oprogramowaniem. Kod został napisany bezpośrednio na komputerze stacjonarnym, a następnie został wysłany i skompilowany na mikrokomputerze zdalnie za pomocą protokołu SSH. Takie rozwiązanie pozwoliło na znacznie przyspieczenie prac nad projektem oraz zredukowanie kosztów, ponieważ nie było konieczne inwestowanie w dodatkowy monitor oraz klawiaturę i myszkę.

Kolejnym problemem było sterowanie obiema osiami X oraz Y w tym samym czasie. Biblioteka, która została użyta w projekcie działa w charakterystyczny sposób, który uniemożliwia bezpośrednie sterowanie kilkoma pinami GPIO na raz. Zadaniem oprogramowania było generowanie dwóch sygnałów całkowicie niezależnych od siebie pod względem częstotliwości. Oznaczało to, że należy zastosować rozwiązanie wielowątkowe. Każdy z wątków generuje indywidualny i niezależny od drugiego sygnał sterujący pojedyncza osią co teoretycznie rozwiązywało problem, ale charakterystyka działania wielowątkowego oraz wymaganych sygnałów wyjściowych powodowała powstawanie nieoczekiwanych zachowań. Częstotliwości sygnałów sterujących każdym z modułów osiągają wartości na poziomie 2000 Hz co oznacza, że impuls został generowany co ok 0,5ms. Rozwiązanie wielowątkowe wymagało korzystania z jednego obiektu kontrolującego GPIO mikrokomputera przez wszystkie wątki co oznaczało konieczność przełączania dostępu pomiędzy oba wątki. W przypadku języka Java czasy przeprowadzania tych operacji były na tyle duże, że znacznie wpływały na sprawność działania całego układu. Urządzenie stawało się niestabilne oraz nieprecyzyjne. Rozwiązaniem tego problemu była rezygnacja z wątków na rzecz działania jednowątkowego. Pozycje obu osi ustalane są jedna po drugiej w bardzo małych odstępach czasu, a różnica kolejnych pozycji jest na tyle mała, że połączenie przemiennego ruchu obu osi z niewielkimi wartościami przesuwu sprawia wrażenie

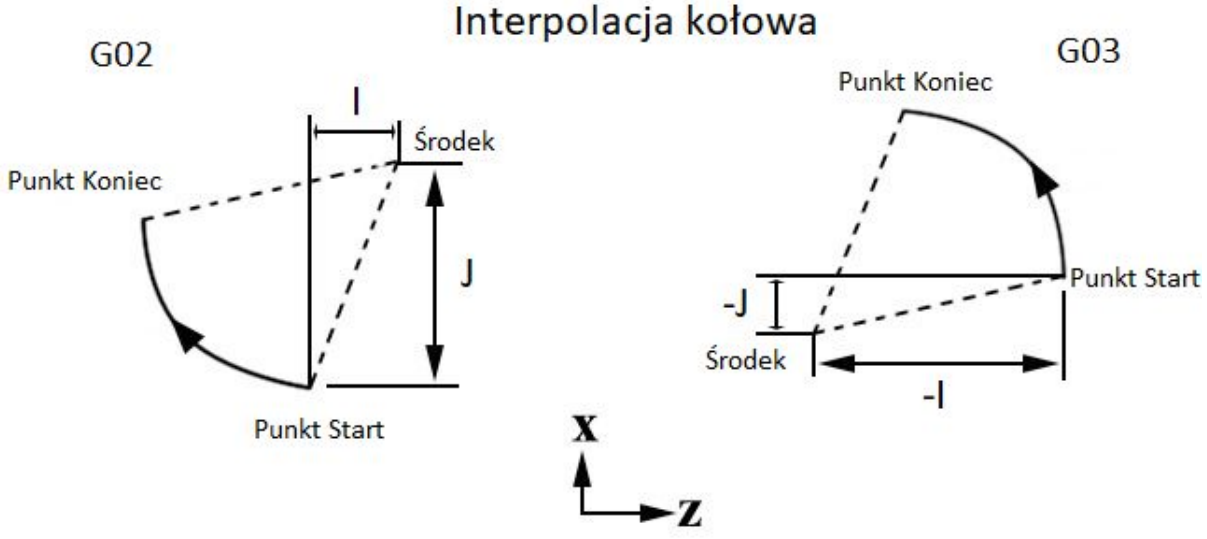
ruchu ciągłego. Dodakowym atutem takiego rozwiązania jest znacznie uproszczenie kodu urządzenia.

Przeszkodą podczas prowadzenia prac deweloperskich okazało się także zastosowanie części z odzysku. Mechanizmy zastosowane jako osie pochodzą z różnych modeli napędów CD co powoduje rozbieżność w parametrach pracy oraz charakterystykach elektrycznych. Każdy mechanizm posiada silnik krokowy oraz prowadnice. Droga z jednego końca elementu na drugi wymaga pewnej liczby kroków wykonanych przez silnik. W przypadku różnych modeli napędów CD te wartości mogą się znacznie różnić, dlatego należało zastosować rozwiązanie polegające na przeliczaniu liczby kroków na milimetry, czyli jednostki stosowane w zapisie G-code. Dodatkowym elementem zastosowanym w celu pozycjonowania mechanizmów są przyciski krańcowe. Służą one do sygnalizowania, że tacka mechanizmu osiągnęła pozycję wyjściową oraz, że ruch musi zostać przerwany. Te informacje stosowane są do inicjalizacji mechanizmu podczas pierwszego uruchomienia programu.

Następnym wyzwaniem programistycznym okazało się być generowanie ruchu po okręgu. Język G-code zawiera rozkazy, które przewidują ruch po okręgu. Wyzwaniem programistycznym okazało się generowanie tego ruchu w świecie rzeczywistym. Rozkazy G-code odpowiedzialne za ruch po okręgu to G02 oraz G03. Parametrami tych komend są:

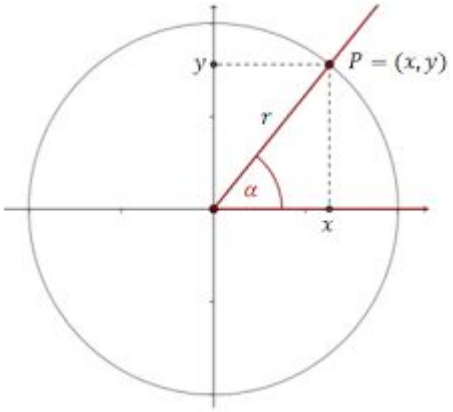
- współrzędne punktu końcowego (x, y)
- offset względem punktu startowego determinujący współrzędne środka okręgu (I, J)

Punktem startowym jest zawsze obecna pozycja narzędzia rysującego. Z tego wynika, że dane, które służą do opisania wymaganego fragmentu okręgu są 3 informacje: punkty koncowy i startowy oraz środek okręgu.



Rys. 6 - Interpolacja kołowa

Rozwiązaniem okazały się być proste obliczenia trygonometryczne wykorzystujące funkcje sinus oraz cosinus. Przykładowe określanie pozycji zostało przedstawione poniżej:



Rys. 7 - Punkt na okręgu

Aby obliczyć pozycję x, y wzdłuż krawędzi okręgu należy zastosować następujące obliczenia:
(Xs, Ys) - współrzędne środka okręgu
P = (x, y) - punkt na krawędzi tego okręgu
r - promień okręgu
x = sin(α) * r + Xs
y = cos(α) * r + Ys

Aby prowadzić narzędzie rysujące po krawędzi okręgu należy obliczać kolejne pozycje punktu P. Można to zrealizować poprzez odpowiednie iterowanie po wartościach kąta α stosując odpowiednio dobrany odstęp od kolejnych wartości. Odstęp pomiędzy kolejnymi wartościami konta determinuje również prędkość poruszania się narzędzia po okręgu.Im większy krok tym większa prędkość. Problemem jednak okazało się to, że fragment rysowane okręgu zwykle nie zaczyna się w punkcie gdzie stopień wynosi 0, dlatego należało znaleźć sposób na znalezienie kąta początkowego, dla którego pozycja narzędzia będzie znajdować się w pozycji “start”. Wyszukiwanie przybliżonego konta początkowego zostało przeprowadzone w następujący sposób, który można nazwać metodą “prób i błędów”:

- ustawianie kąta na 0 stopni,
- obliczanie pozycji narzędzia dla kolejnych wartości kąta iterowanych o mały ułamek stopnia,
- sprawdzanie czy obliczona pozycja znajduje się w niewielkiej odległości od szukanego kąta (odległość ok 0.05mm),

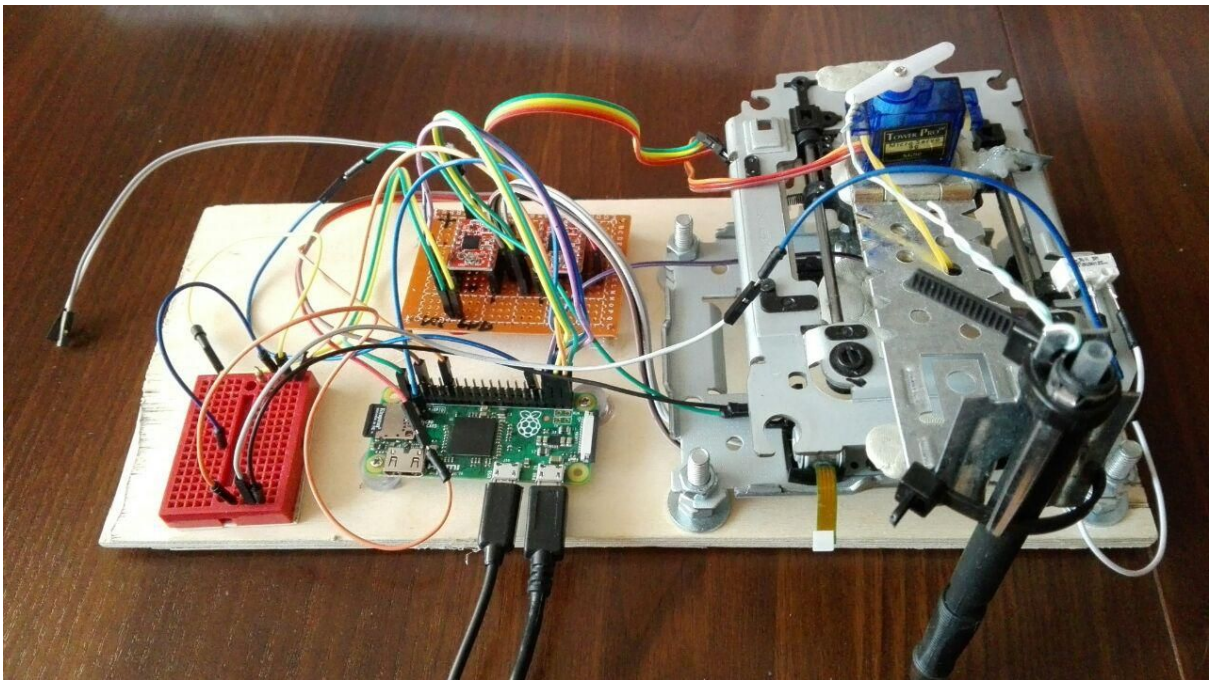
- znalezienie kąta, dla którego odległość jest niewielka (założenie, że jest to kąt szukany)

Wadą takiego rozwiązania jest jednak stosunkowo niska precyzja. Mimo, że kąt znajdowany jest tą metodą w ułamku sekundy to nie jest to optymalna metoda, a znaleziona wartość to tylko przybliżenie. Może się również zdarzyć, że z powodu błędu danych wejściowych, pozycja nie zostanie ustalona nigdy, ponieważ nie zostanie znaleziona pozycja w odpowiednio małej odległości od punktu “start”. Taka sytuacja jest jednak prawie niemożliwa do zajścia dlatego zastosowany sposób implementacji działa stabilnie i spełnia swoje zadanie, a w razie błędu danych wejściowych program zostanie zatrzymany i poinformuje o zaistniałym błędzie.

Kolejnym problemem okazała się część sprzętowa projektu. Użyte silniki, które pochodzą z napędu CD ROM wymagają napięcia około 5V. Sterowniki silników krokowych zastosowane w projekcie przystosowane są do pracy z napięciami powyżej 8V. W celu użytkowania silników z tymi sterownikami zostało podjęte ryzyko uszkodzenia silników co oznaczało użycie napięcia powyżej 8V. Praca z przekroczonym napięciem powoduje podwyższone nagrzewanie się silników oraz szybsze ich zużycie, mimo to dla celów eksperymentalnych silniki zostały zasilone nawet 12V. Dzieki temu zabiegowi można było uzyskać większa siłę silników co okazało się być niezbędne w tej konstrukcji ze względu na dodatkowe obciążenia w postaci konstrukcji mechanizmu rysującego. Jednakże lepszym rozwiązaniem byłoby zastosowanie sterowników o prawidłowych parametrach. Przykładowym rozwiązaniem może być samodzielna budowa modułów na bazie mostków H oraz mikrokontrolera Atmega. Ze względu na to, że płytka modułu byłaby projektowana od podstaw, to jej forma mogłaby być kompatybilna z układem A4988, a to oznacza, że można by jedynie fizycznie zamienić moduły sterujące na własne, a pozostała część urządzenia pozostałaby w stu procentach niezmieniona.

10. Realizacja

Realizacja projektu została przeprowadzona w taki sposób, aby zminimalizować koszty i poziom skomplikowania urządzenia. Użyte części pochodzą głównie z demontażu z wyjątkiem części elektronicznej. Silniki oraz przyciski zostały pozyskane ze sprzętu komputerowego. Całość umieszczona została na kawałku sklejki, a główny mechanizm został zamontowany za pomocą śrub.



Rys. 8 - Ukończone urządzenie

Głównymi elementami całego urządzenia są mechanizmy pozyskane z napędów CD. W celu osiągnięcia dwóch ruchomych osi X oraz Y należało połączyć mechanizmy prostopadle do siebie. Zostało to zrealizowane za pomocą łatwo dostępnych elementów oraz narzędzi. Między obiema mechanizmami koniecznie należało zachować dystans, ponieważ wystające elementy nie pozwalały na bezpośrednie połączenie mechanizmów. Zostało to rozwiązane poprzez dodanie odpowiednio ukształtowanego wieszaka służącego zwykle do montażu płyt kartonowo-gipsowych. Ten element idealnie spełnia swoje zadanie, ponieważ jest bardzo łatwo dostępny, ma odpowiednie wymiary, jest sztywny i łatwo można go kształtować. Wieszak do montażu płyt kartonowo gipsowych można zakupić w każdym sklepie budowlanym i wygląda następująco:

Rys. 9 - Wieszak płyt GK

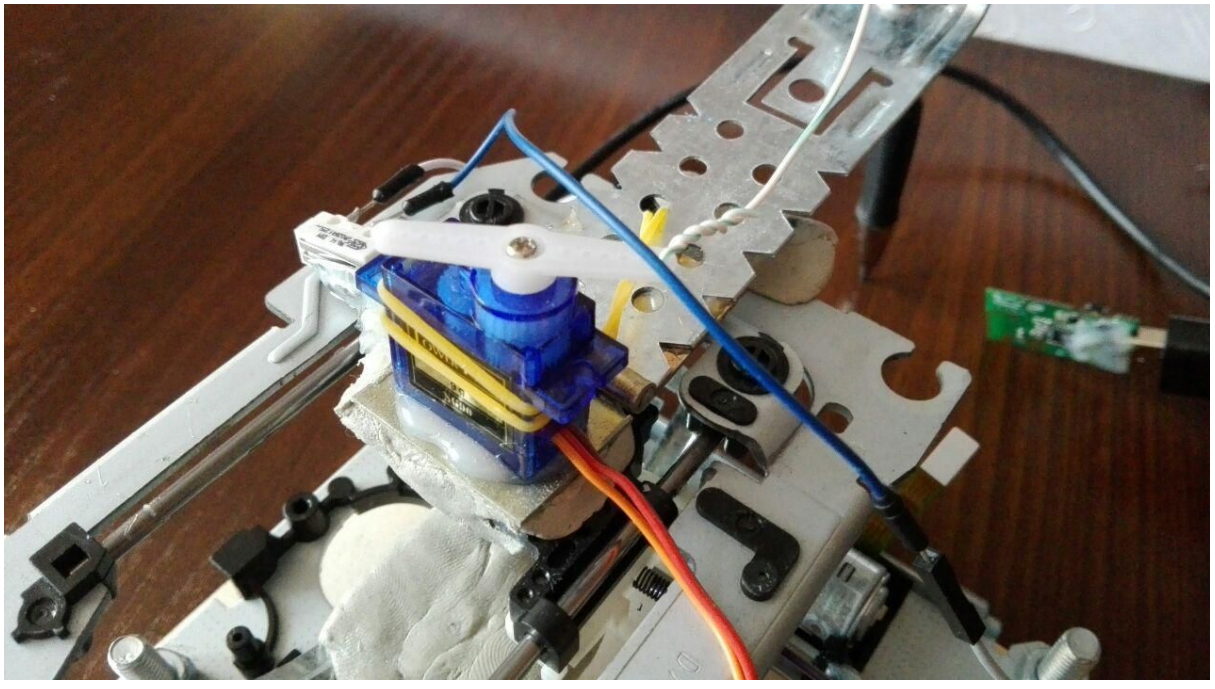
Stalowy element został ukształtowany w formę litery U, a następnie zamocowany za pomocą masy epoksydowej “poxilina”. Jest to substancja wyglądem i konsystencją przypominającą plastelinę, która po utwardzeniu zapewnia nierozerwalne połączenie między elementami. Taki sposób montażu idealnie nadaje się do zastosowania w tym przypadku, ponieważ powierzchnia montażu jest nieregularna, a masa plastyczna dobrze wypełnia wszystkie otwory i wgłębienia zapewniając dobre mocowanie elementów. Końcowe połączenie mechanizmów za pomocą wymienionych elementów wygląda następująco:

Rys. 10 - Sposób montażu osi

Rys. 11 - Sposób montażu osi

Rys. 12 - Sposób montażu osi

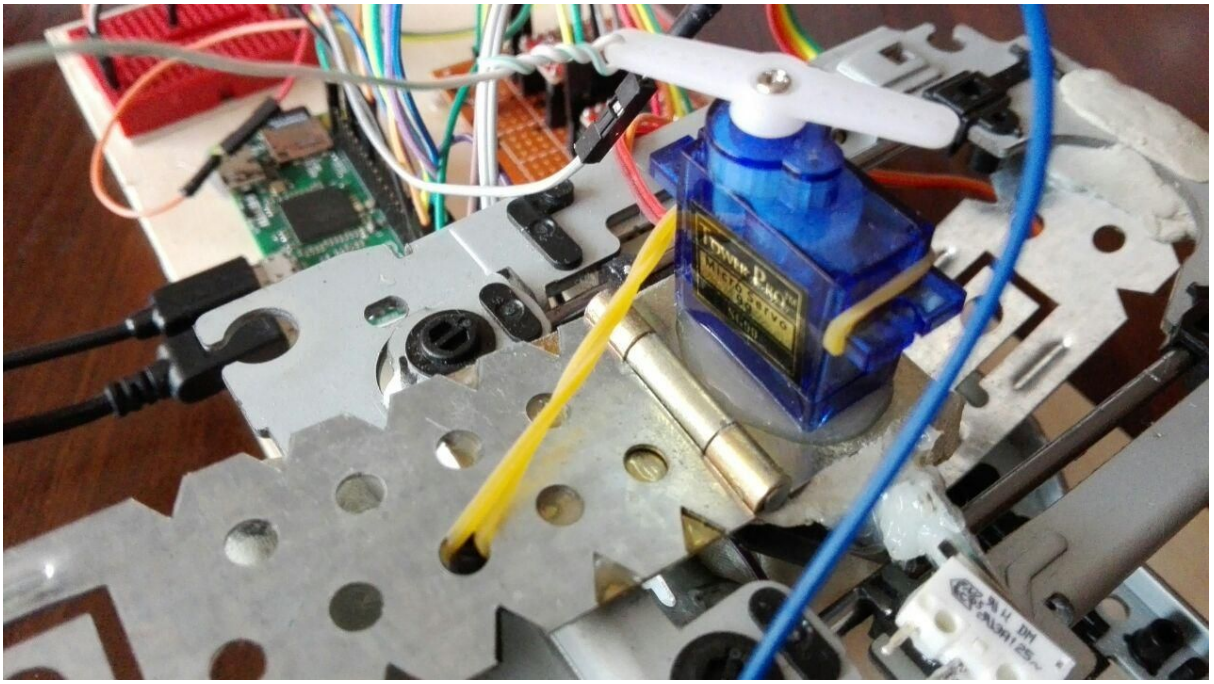
Mechanizm podnośnika narzędzia rysującego również opiera się o wieszak płyt gipsowych oraz masę epoksydową. Zamontowany mechanizm wraz z serwomechanizmem został zrealizowany następująco:



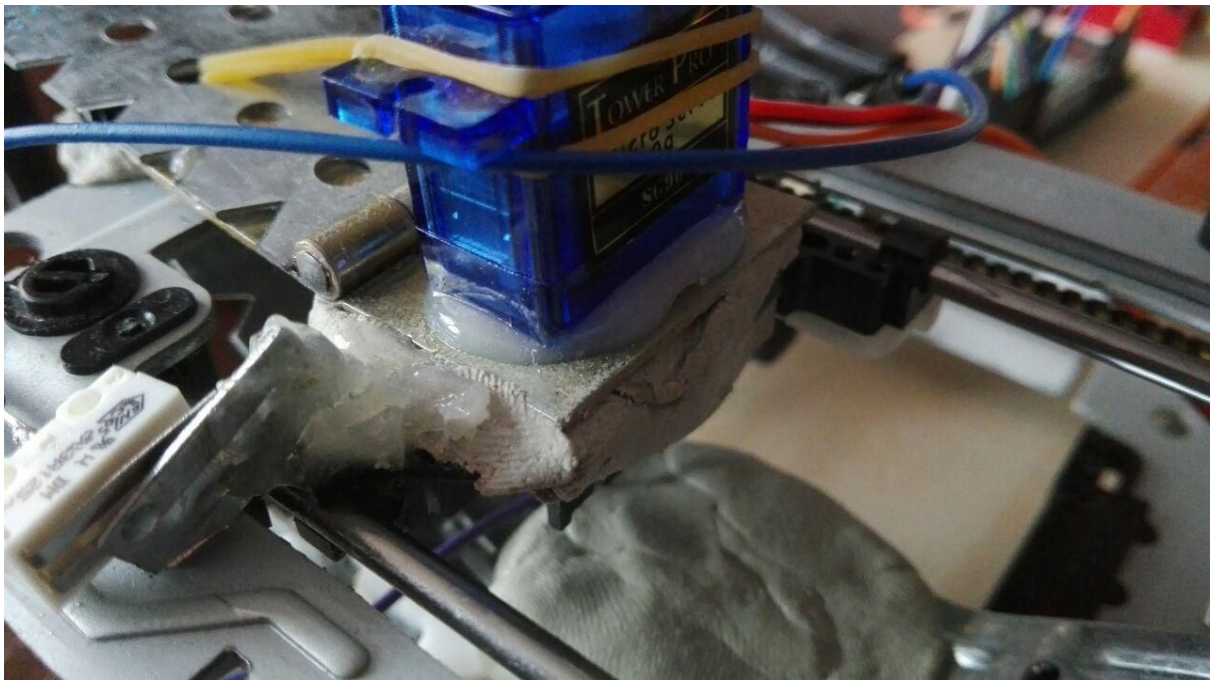
Rys. 13 - Sposób montażu serwomechanizmu

Widoczny na powyższej fotografii serwomechanizm odpowiedzialny jest za podnoszenie i opuszczanie narzędzia rysującego. Został zamontowany za pomocą

kleju epoksydowego, który zapewnia mocne i bardzo szybkie połączenie elementów. Po 10 minutach od montażu można było użytkować urządzenie, ponieważ klej osiąga wystarczającą twardość. Mechanizm podnośnika jest ruchomy i został zbudowany na podstawie zawiasu meblowego. Wadą tego rozwiązania jest fakt, że zawiasy przeznaczone dla mebli produkowane są z niską precyzją. Zawias zastosowany w tym projekcie posiadał luzy, które znacząco wpływały na jakość powstałego rysunku. Rozwiązaniem okazała się być zwykła gumka biurowa, która nie ogranicza ruchów elementu i jednocześnie niweluje efekt luzów.

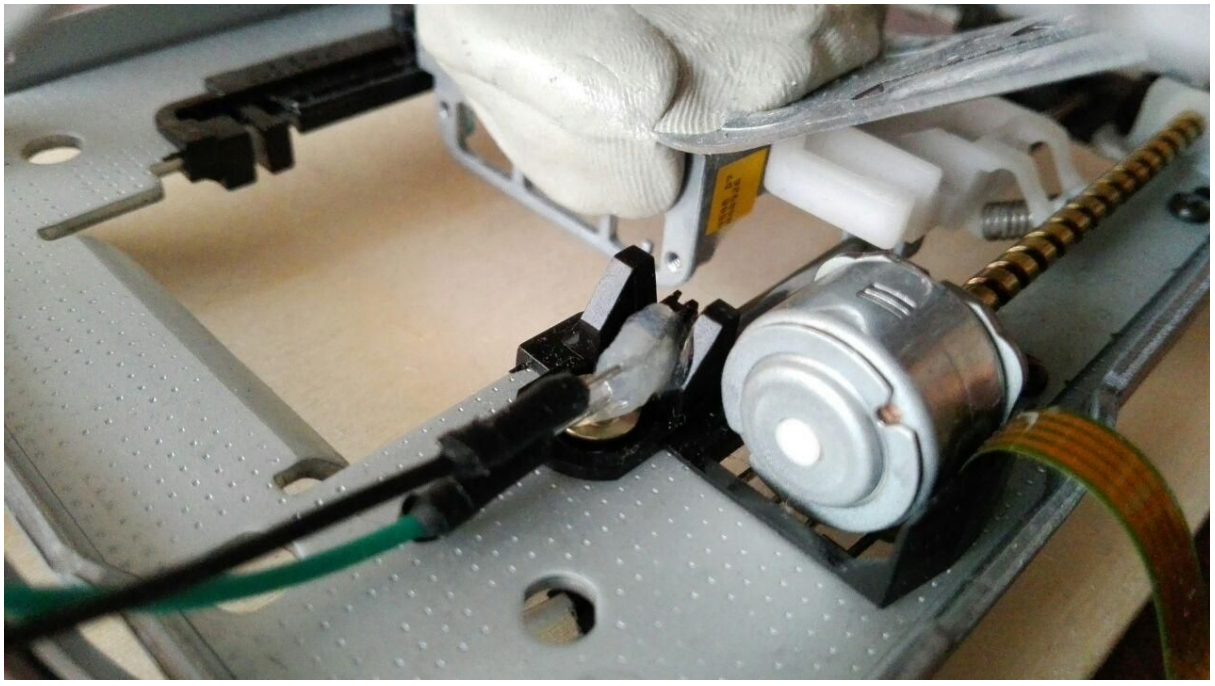


Rys. 13 - Gumka biurowa wraz z zawiasem meblowym i serwomechanizmem



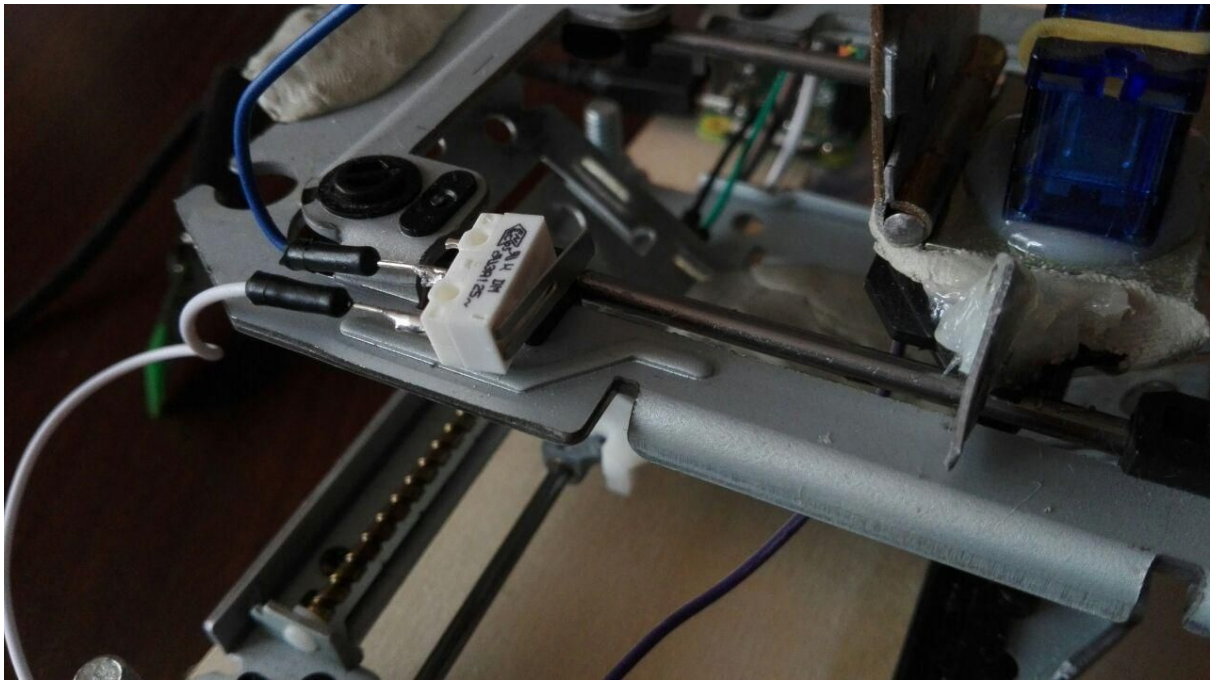
Rys. 14 - Sposób montażu serwomechanizmu oraz ramienia podnośnika

Kluczowymi elementami konstrukcji są przyciski krańcowe. Służą do ograniczenia ruchu osi oraz kalibracji. Kiedy przycisk krańcowy osi zostanie wciśnięty oznacza to, że oś znajduje się w pozycji 0 oraz nie może wykonywać dalszego ruchu.



Rys. 15 - Przycisk krańcowy osi X - rozwarty

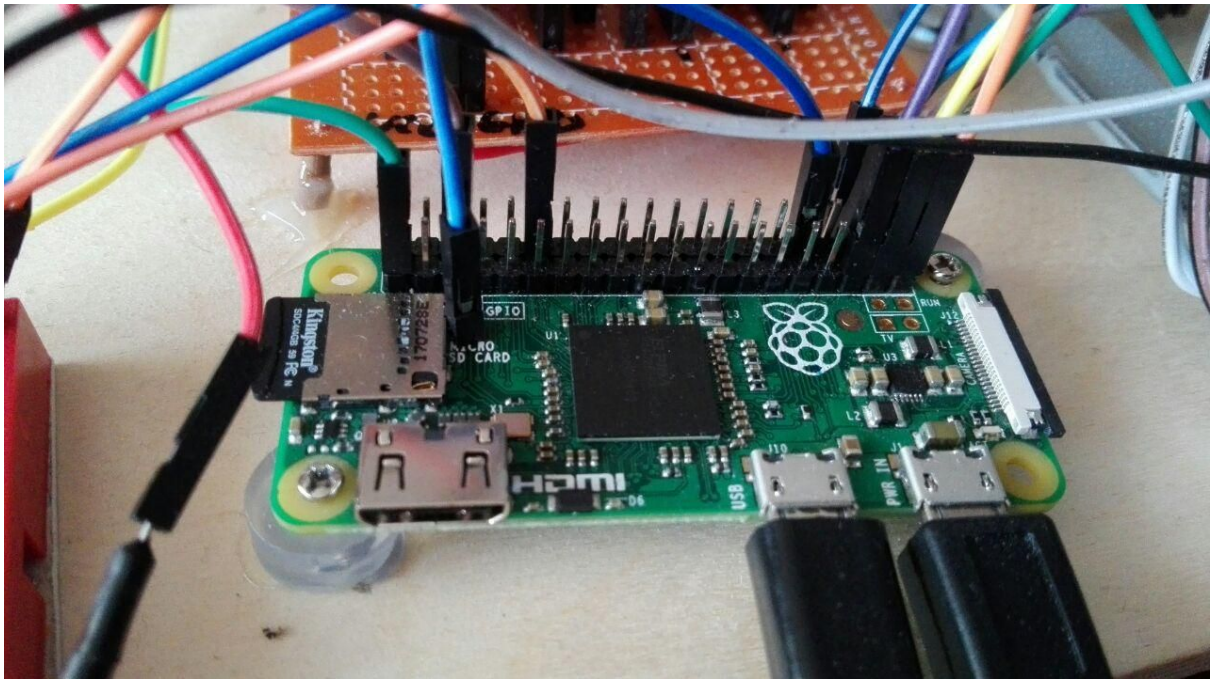
Rys. 16 - Przycisk krańcowy osi X - zwarty (oś X znajduje się w punkcie 0)



Rys. 17 - Przycisk krańcowy osi Y - rozwarty

Rys. 18 - Przycisk krańcowy osi Y - zwarty (oś Y znajduje się w punkcie 0)

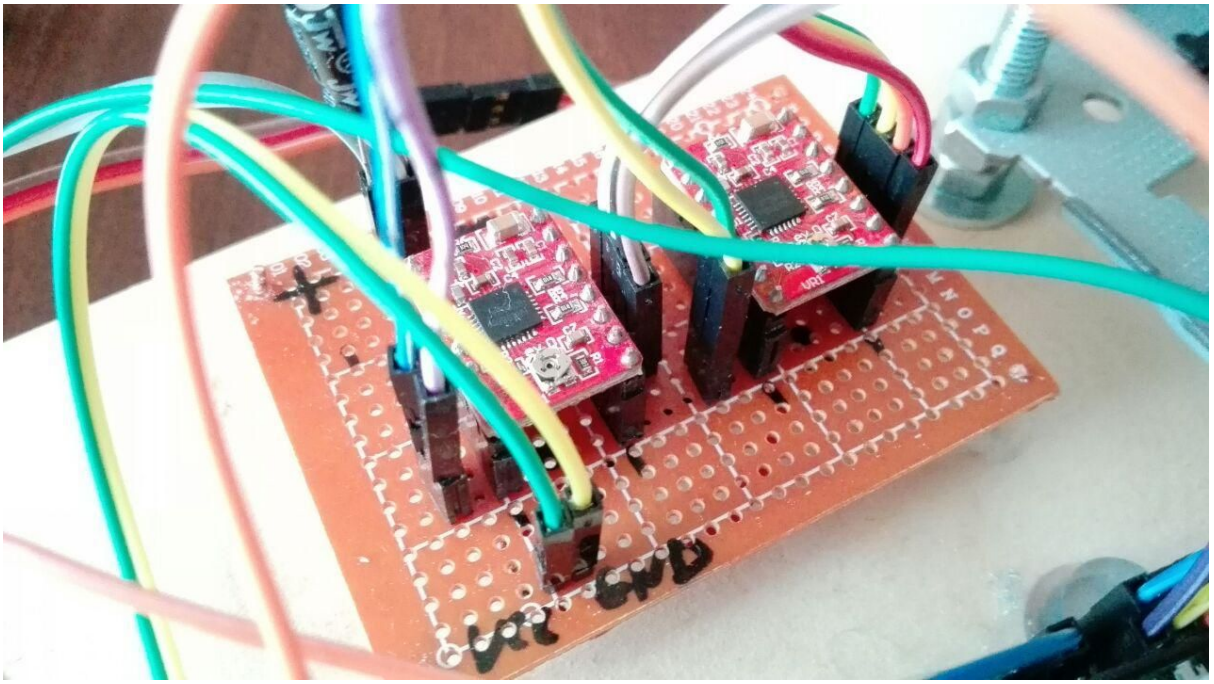
Główną jednostką sterującą całym urządzeniem jest mikrokomputer Raspberry Pi 0. Cały układ elektroniczny został podłączony do portów GPIO tego mikrokomputera. Został zamontowany za pomocą podkładek dystansowych z tworzywa sztucznego oraz kleju do podstawy.



Rys. 19 - Mikrokomputer Raspberry Pi Zero

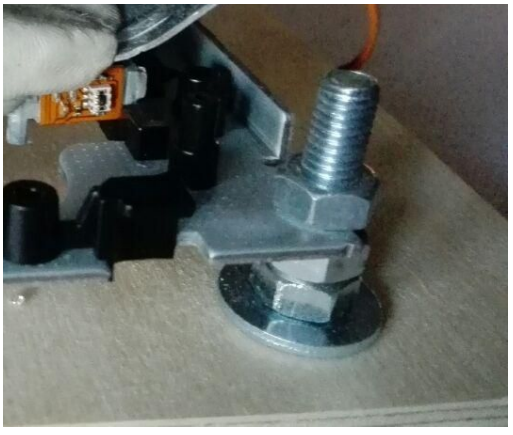
Rolę dysku twardego pełni karta SD o pojemności 4GB. Taki rozmiar wymaga zainstalowania specjalnej wersji systemu operacyjnego ze względu na to, że minimalna wielkość karty pamięci dla pełnego systemu wynosi 8GB. Specjalna wersja systemu różni się od pełnej przede wszystkim tym, że nie posiada interfejsu graficznego. W obecnym zastosowaniu interfejs jest zbędny, zatem taki system spełnia swoje zadanie w stu procentach.

Elementem odpowiedzialnym za sterowanie silnikami krokowymi jest samodzielnie wykona płytka drukowana z dwoma modułami A4988.



Rys. 20 - Płytka z modułami sterującymi A4988

Ważnym elementem konstrukcyjnym jest także sposób montażu mechanizmów rysujących. Zastosowane śruby pozwalają na regulację i poziomowanie całości.



Rys. 21 - Sposób montażu mechanizmu

Najważniejszym w całej przedstawionej konstrukcji jest narzędzie rysujące. W urządzeniu można zamontować praktycznie każde ogólnodostępne przybory do pisania np marker, długopis, pisak, zakreślacz, ołówek, cienkopis, kredkę itp.

11. Najważniejsze fragmenty kodu

Kod źródłowy składa się z wielu pojedynczych modułów odpowiedzialnych za wiele ról. Najważniejsze z nich to interpretacja G-code, realizacja komend G-code takich jak interpolacja kołowa, ruch liniowy itp, ważnym aspektem jest również ustawianie pozycji osi oraz prędkości ich ruchów.

Fragment kodu odpowiedzialny za interpolację liniową przedstawiony jest poniżej:

```
int pathLen = (int)Math.sqrt((x - posX) * (x - posX) + (y - posY) * (y - posY));
//pierwiastek z sum kwadratów różnic/długość ścieżki
double xStep = ((x - posX)/(double)pathLen)/(double)100; //długość kroku osi X
double yStep = ((y - posY)/(double)pathLen)/(double)100; //długość kroku osi Y

for(int i = 0; i < pathLen*100; i++){
   lowerAxis.setPos(posX + xStep*i, 40); //ustawienie kolejnej pozycji osi
   upperAxis.setPos(posY + yStep*i, 40);
}
lowerAxis.setPos(x, 40); //ustawienie pozycji końcowej (bez błędu wynikającego z
dzielenia)
upperAxis.setPos(y, 40);

posX = x; //zapisanie aktualnej pozycji
posY = y;
```

Fragment kodu odpowiedzialny za interpolację kołową zgodnie z ruchem wskazówek zegara przedstawiony jest poniżej:

```
double sX = posX + i; //ustalenie współrzędnych środka okręgu/i j to offsety od
punktu start
double sY = posY + j;

double radius = Math.sqrt((sX - posX) * (sX - posX) + (sY - posY) * (sY - posY));
//pierwiastek z sum kwadratów różnic / pitagoras

System.out.println("Promien okregu: " + radius);

double step = 1/(double)20; //krok kolejnych iteracji
int iCnt = findDegree(sX, sY, radius, posX, posY); //licznik iteracji/szukanie kąta
startowego

double xCalc = posX; //współrzędne narzędzia obliczane co iteracje
double yCalc = posY;

double xDist = Math.abs(x - xCalc); //odległość od punktu końcowego
```

```
double yDist = Math.abs(y - yCalc);

while(xDist > 0.05 || yDist > 0.05){ //dopóki odległość x oraz y narzędzia od punktu końcowego nie będzie mniejsza niż 0.05mm
   iCnt++;
   double radians = Math.toRadians(step*iCnt); //konwersja stopni na radiany

   xCalc = Math.sin(radians) * radius + sX; //obliczenie pozycji x narzędzia
   yCalc = Math.cos(radians) * radius + sY; //obliczenie pozycji y narzędzia

    xDist = Math.abs(x - xCalc); //odświeżenie obliczeń dot. odległości od punktu końcowego
   yDist = Math.abs(y - yCalc);

   upperAxis.setPos(yCalc, 40); //ustawienie pozycji osi y narzędzia
   lowerAxis.setPos(xCalc, 40); //ustawienie pozycji osi y narzędzia
   if(iCnt > 36000) break; //jeżeli narzędzie wykona 360 stopni oznacza to błąd
}

posX = x; //aktualizacja współrzędnych narzędzia
posY = y;
```

Fragment kodu odpowiedzialny za parsowanie G-code:

```
if(!line.contains("%") && !line.isEmpty()) {
   String gcodeLine = line.replaceAll("\\((.*?)\\)", "");
   if (gcodeLine.length() > 0) {
       if(debug) { //tryb debug
           System.out.print("\tGcode: " + gcodeLine);
           Pattern pattern = Pattern.compile("\\((.*?)\\)");
           Matcher matcher = pattern.matcher(line);
           if (matcher.find()) {
               System.out.print(" //" + matcher.group(1));
           }
           System.out.print("\n");
       }

         String[] parts = gcodeLine.split(" "); //podzielenie linii wejściowej na pojedyncze fragmenty (fragment 0 to komenda G-code)

       switch (parts[0]){
           case "G00":
               g00(parts);
               break;
           case "G01":
               g01(parts);
               break;
           case "G02":
               g02(parts);
               break;
           case "G03":
               g03(parts);
               break;
```

```
            case "G04":
                g04(parts);
                break;
            case "G21":
                g21(parts);
                break;
            case "M0":
                m0(parts);
                break;
            case "M2":
                m2(parts);
                break;
            case "M3":
                m3(parts);
                break;
            case "M4":
                m4(parts);
                break;
            case "M5":
                m5(parts);
                break;
            default:
                System.out.println("\tNIEOBSŁUGIWANA KOMENDA: " + parts[0]);
                break;
        }

    } else {
        if(debug) {
            Pattern pattern = Pattern.compile("\\((.*?)\\)");
            Matcher matcher = pattern.matcher(line);
            if (matcher.find()) {
                System.out.println("\t//" + matcher.group(1));
            }
        }
    }
}else{
    if (debug)
        System.out.println("\t////");
}
```

Opis fragmentu:
Program otrzymaną w argumencie metody linię tekstu przekształca oraz wyszukuje w niej szukanych fragmentów. Następnie porównuje otrzymany wynik z bazą obsługiwanych komend, a następnie wywołuje konkretną metodą odpowiedzialną za wykonanie dalszych czynności.

Metody odpowiedzialne za wykonanie kroków w przód i w tył prezentowane są poniżej:

```
public void stepForward(long delayUs){
```

```
        if(stepCounter >= 0 && stepCounter < resolution &&
isCalibrated){
        stepPin.high();
        stepPin.low();
        busyWaitMicros(delayUs);
        stepCounter++;
    }else{
          System.out.println("StepForward: Nie mozna wykonac kroku.
Blad kalibracji lub przekroczono zakres. " + stepCounter );
    }
}

public void stepBackwards(long delayUs){
        if(stepCounter > 0 && stepCounter <= resolution &&
isCalibrated){
        dirPin.toggle();
        stepPin.high();
        stepPin.low();
        busyWaitMicros(delayUs);
        dirPin.toggle();
        stepCounter--;
    }else{
         System.out.println("StepBackwards: Nie mozna wykonac kroku.
Blad kalibracji lub przekroczono zakres. " + stepCounter);
    }
}
```

Metoda, która wykonuje precyzyjne obliczenia odnośnie ustalania pozycji i prędkości każdej z osi przedstawiony jest poniżej:

```
public void setPos(double pos, double speed){ //pos w mm, speed w mm/s
   if(isCalibrated){
       int counterTmp = stepCounter;
       double delayTmp = (1/(double)speed)*1000000; //obliczanie prędkości
       double stepsPerMilimeter = resolution / (double)axisDimension;
       double positionToBeSetSteps = pos * stepsPerMilimeter;
       long delay = (long) round(delayTmp / (double)stepsPerMilimeter);
       actualPosition = positionToBeSetSteps;
       logicPosition = pos;

       if(actualPosition > stepCounter){
           for(int i = 0; i < actualPosition - counterTmp; i++){
               stepForward(delay);
           }
       }else{
           for(int i = 0; i < counterTmp - actualPosition; i++){
               stepBackwards(delay);
           }
       }
```

```
    }else{
        System.out.println("Pozycja nie zostala zainicjalizowana!");
    }
}
```

12. Perspektywa rozwoju

Stosując ten sam kod źródłowy można także stosować różne rozwiązania sprzętowe. Mechanizmy CD Rom mogą być zastąpione znacznie większymi mechanizmami zdolnymi do wykonywania różnorodnych zadań.

Jednym z zastosowań może być frezarka cyfrowa. Rozbudowując mechanizm osi i dodając narzędzie frezujące można zmienić przeznaczenie urządzenia bez zmiany w kodzie źródłowym. Takie urządzenie może np wycinać kształty z drewna, metalu oraz tworzyw sztucznych.

Kolejnym zastosowaniem jest wycinarka do styropianu. Zmieniając orientację całego mechanizmu frezującego i zmieniając frez na gorący drut można wycinać dowolne kształty w styropianie. Dodatkowo należałoby dodać zabezpieczenia przed przecinaniem ścieżek, aby wycięte kształty nie ulegały uszkodzeniu.

Końcówka rysująca może zostać zastąpiona dowolnym narzędziem, które może zostawić ślad. Przykładowe narzędzia to laser, wypalarka (lutownica lub gorący drut), woda pod ciśnieniem itp.

Całość można rozbudować o kolejną oś - Z. Taki układ pozwoliłby na budowę drukarki 3D. W takim przypadku program musiałby mieć zaimplementowane dodatkowe funkcje odpowiedzialne za obsługę osi Z.

Obróbka numeryczna jest powszechnie znanym pojęciem w technice, dlatego projekt ma ogromne możliwości rozwoju. Modyfikując mechanizm urządzenia można by zastosować je nawet w przemyśle.

13. Wnioski i ocena własna

Projekt dla jego autorów był znakomitym sposobem na wstępne poznanie techniki precyzyjnej obróbki materiałów przez obróbkę numeryczną, znanej ogólnie pod nazwą CNC. Dzięki poprawnemu zaprojektowaniu układów elektronicznych, a następnie konstrukcji maszyny drukującej uzyskano zadowalające efekty w kwestii stabilności urządzenia oraz dokładności rysowania obiektów na przestrzenii dwuwymiarowej.

Utworzone oprogramowanie z wykorzystaniem biblioteki Pi4J okazało się wystarczająco optymalne pod względem wymagań sprzętowych narzuconych przez architekturę projektu(sterowników silników krokowych) i systemu operacyjnego. Pozwoliło to na uzyskanie doświadczenia w programowaniu urządzeń wbudowanych w języku wysokopoziomowym. Gotowy projekt jest dobrą bazą do jego poszerzenia w kontekście rozmiarów i szybkości rysowania lub do utworzenia na jego podstawie całkiem nowego urządzenia CNC - drukarki 3D lub mini-frezarki.

Github
https://github.com/AgeDee/PlotterCDRom