POLITECHNIKA GDAŃSKA

WYDZIAŁ ELEKTRONIKI, TELEKOMUNIKACJI I INFORMATYKI

Imię i nazwisko studenta: Benedykt Bela
Nr albumu: 175541
Poziom kształcenia: Studia pierwszego stopnia
Forma studiów: stacjonarne
Kierunek studiów: Automatyka i Robotyka
Profil: Systemy decyzyjne i robotyka

PROJEKT DYPLOMOWY INŻYNIERSKI

Tytuł projektu w języku polskim: Układ sterowania frezarką zamontowaną na robocie kartezjańskim

Tytuł projektu w języku angielskim: Control system for a milling machine mounted on a Cartesian robot

Opiekun pracy: dr inż. Marek Tatara
Data ostatecznego zatwierdzenia raportu podobieństw w JSA: 08.01.2022

POLITECHNIKA GDAŃSKA | WYDZIAŁ ELEKTRONIKI, TELEKOMUNIKACJI I INFORMATYKI

OŚWIADCZENIE dotyczące pracy dyplomowej zatytułowanej: Układ sterowania frezarką zamontowaną na robocie kartezjańskim

Imię i nazwisko studenta: Benedykt Bela
Data i miejsce urodzenia: 19.04.1999, Gdynia
Nr albumu: 175541

Wydział: Wydział Elektroniki, Telekomunikacji i Informatyki
Kierunek: automatyka i robotyka

Poziom kształcenia: pierwszy
Forma studiów: stacjonarne

Typ pracy: projekt dyplomowy inżynierski

Świadomy(a) odpowiedzialności karnej z tytułu naruszenia przepisów ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz. U. z 2019 r. poz. 1231, z późn. zm.) i konsekwencji dyscyplinarnych określonych w ustawie z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce (t.j. Dz. U. z 2020 r. poz. 85, z późn. zm.),[1] a także odpowiedzialności cywilnoprawnej oświadczam, że przedkładana praca dyplomowa została opracowana przeze mnie samodzielnie.

Niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadaniem tytułu zawodowego.

Wszystkie informacje umieszczone w ww. pracy dyplomowej, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami zgodnie z art. 34 ustawy o prawie autorskim i prawach pokrewnych.

09.01.2022, Benedykt Bela

Data i podpis lub uwierzytelnienie w portalu uczelnianym Moja PG

**) Dokument został sporządzony w systemie teleinformatycznym, na podstawie §15 ust. 3b Rozporządzenia MNiSW z dnia 12 maja 2020 r. zmieniającego rozporządzenie w sprawie studiów (Dz.U. z 2020 r. poz. 853). Nie wymaga podpisu ani stempla.*

[1] Ustawa z dnia 20 lipca 2018 r. Prawo o szkolnictwie wyższym i nauce:

Art. 312. ust. 3. W przypadku podejrzenia popełnienia przez studenta czynu, o którym mowa w art. 287 ust. 2 pkt 1–5, rektor niezwłocznie poleca przeprowadzenie postępowania wyjaśniającego.

Art. 312. ust. 4. Jeżeli w wyniku postępowania wyjaśniającego zebrany materiał potwierdza popełnienie czynu, o którym mowa w ust. 5, rektor wstrzymuje postępowanie o nadanie tytułu zawodowego do czasu wydania orzeczenia przez komisję dyscyplinarną oraz składa zawiadomienie o podejrzeniu popełnienia przestępstwa.

STRESZCZENIE

W projekcie podjęto się zaplanowania oraz realizacji układu poruszania i sterowania frezarką do drewna, opartą na robocie kartezjańskim. Do gotowej konstrukcji fizycznej dobrano napęd oraz sposób jego przeniesienia na osie robota. Pierwszą część zrealizowano za pomocą silników krokowych z enkoderami, natomiast drugą z wykorzystaniem przekładni ze śrubami trapezowymi. Do napędu zakupiono odpowiednie sterowniki oraz zasilacz. Taki zestaw w połączeniu z komputerem sterującym pozwolił na poruszanie robotem. Do tego zrealizowano sterowanie pompą chłodziwa oraz silnikiem frezarki poprzez falownik. Zastosowano również przycisk bezpieczeństwa zatrzymujący pracę maszyny.

Sterowanie zrealizowano za pomocą płytki STM32F3DISCOVERY oraz standardowego komputera PC. Napisany program posiada czytelny interfejs i pozwala na wczytywanie zewnętrznych trajektorii narzędzia lub generację własnych prostych ścieżek za pomocą podawanych w odpowiednie miejsca parametrów. Zadane trajektorie są następnie przekształcane na listę kroków silników, które dadzą pożądany efekt. Po wygenerowaniu tejże listy, z poziomu interfejsu użytkownika możemy sterować wykonywaniem programu.

Słowa kluczowe: robot kartezjański, frezarka, CNC, sterowanie, Python

Dziedzina nauki i techniki, zgodnie z wymogami OECD: Nauki inżynieryjne i techniczne, Robotyka i automatyka

ABSTRACT

In this project author undertook a task to plan and develop a motion and control system of the wood milling machine, embedded on the cartesian robot. To the already made physical construction a drive and its transfer to robot's axes was designed and assembled. The drive was developed with stepper motors concatenated with encoders while its transfer was developed with screw shaft. For the drive a compatible motor driver and power supply were selected. Such a kit in connection with control computer allowed to move the robot. In addition, a control system for the coolant pump and, through the inverter, milling machine engine was made. Emergency button designed to stop the whole machine was also included in the design.

The heart of control system is union of circuit board STM32F3DISCOVERY and standard personal computer. Written program has clear and readable interface and functionality of importing external tool trajectories. It can also generate its own simple paths on the basis of parameters given in appropriate labels. The trajectories are then converted into the list of motor steps which will give us desired effect. When the list is generated, user can easily control execution of the program using buttons from the interface.

Keywords: cartesian robot, milling machine, CNC, control, Python

SPIS TREŚCI

WYKAZ WAŻNIEJSZYCH OZNACZEŃ I SKRÓTÓW

GUI – widok użytkownika (ang. *graphical user interface*)

GPIO – wejście-wyjście ogólnego przeznaczenia (ang. *general-purpose input/output*)

UART – uniwersalny asynchroniczny nadajnik-odbiornik (ang. *universal asynchronous receiver-transmitter*)

CAM – komputerowe wspomaganie wytwarzania (ang. *computer aided manufacturing*)

rpm – liczba obrotów na minutę (ang. *revolutions per minute*)

1. WSTĘP I CEL PRACY

Pomysł na temat niniejszego projektu inżynierskiego zrodził się z dostępu do konstrukcji mechanicznej robota kartezjańskiego zbudowanego, aby z wykorzystaniem frezarki móc wyrównywać powierzchnię obrabianego drewna. Jednak dało się z urządzenia korzystać wyłącznie za pomocą siły własnych rąk i mięśni. W tym miejscu zrodził się pomysł, aby do konstrukcji mechanicznej zaprojektować oraz zrealizować układ sterowania tegoż robota. Zarówno w kwestii wyboru mechanizmu poruszania poszczególnymi osiami jak i oprogramowania pozwalającego na realizację zadanej trajektorii frezowania.

1.1. Cel pracy

Celem projektu był dobór elementów do konstrukcji mechanicznej robota kartezjańskiego z frezarką oraz zaprojektowanie i implementacja układu sterowania robotem. W ramach projektowanego systemu sterowania należało przewidzieć sterowanie ręczne z poziomu pilota oraz automatyczne odtwarzanie zadanej trajektorii.

2. KONCEPCJA ROZWIĄZANIA PROBLEMU

Pierwszą częścią prac nad projektem był przegląd możliwych rozwiązań [2] w zakresie realizacji części mechanicznej, służącej do poruszania robotem. Wybór dotyczył zarówno samego napędu jak i sposobu jego przeniesienia na poszczególne osie frezarki.

2.1. Rodzaj napędu

Przy wyborze rodzaju napędu brano pod uwagę przede wszystkim dwie możliwości:

- zastosowanie standardowych silników prądu stałego,
- wykorzystanie silników krokowych.

W pierwszym przypadku mielibyśmy do czynienia z dużymi prędkościami obrotowymi przy stosunkowo niewielkim momencie obrotowym. To wymagałoby zastosowania przekładni z bardzo dużą redukcją. Ponadto, aby wiedzieć w każdym momencie gdzie znajdują się wszystkie osie robota, niezbędne byłoby zastosowanie enkoderów. Te czujniki położenia kątowego wraz ze znajomością przełożeń pozwalałyby nam obliczyć i wyciągnąć informację zwrotną o aktualnym położeniu frezarki.

W przypadku silników krokowych z odpowiednimi sterownikami możemy precyzyjnie poruszać maszyną wykonując pojedyncze kroki [1]. Stosując sterowanie mikrokrokowe możemy poruszać się nawet o ułamki kroków standardowych. Wysłanie jednego impulsu na wejście sterownika powoduje jeden skok silnika. Zliczając wysłane impulsy z poziomu programu możemy dokładnie określić aktualne położenie robota w obszarze działania. Nie możemy oczywiście zapomnieć o możliwości gubienia kroków i wypadaniu z synchronizmu. Tutaj z pomocą przychodzą silniki z wbudowanymi enkoderami. Odpowiednie sterowniki z wejściami przewidzianymi dla takiego czujnika położenia automatycznie pilnują, aby liczba kroków była zgodna z liczbą wysłanych impulsów. Natomiast informację o błędach krytycznych możemy odczytać z wyjść sterownika, służących do komunikacji z użytkownikiem. Dodatkowo bardzo łatwo w przypadku silnika krokowego manipulować prędkością obrotową. W zależności od tego jak szybko będziemy wysyłać kolejne sygnały na wejścia sterownika, otrzymamy proporcjonalne szybkości obrotowe wału silnika.

Na podstawie powyższej analizy podjęto decyzję o wykorzystaniu silników krokowych w realizacji projektu. Jest to rozwiązanie pod wieloma względami bardziej wygodne i prostsze do wysterowania z poziomu komputera zarządzającego pracą całego układu.

2.2. Przeniesienie napędu

W ścisłej korelacji z napędem należało dokonać wyboru sposobu przeniesienia tegoż napędu [7] na każdą z trzech osi naszego robota kartezjańskiego. Pod uwagę zostały wzięte trzy rozwiązania:

- listwa zębata z kołem zębatym,
- przekładnia z pasem zębatym,

- przekładnia śrubowa.

W pierwszym przypadku konieczne byłoby zamontowanie listwy zębatej na całej długości osi ruchu. Ponadto, do ruchomego elementu tejże osi należałoby przymocować napęd wraz z kołem zębatym. Silnik, obracając zębatką przyłożoną do omawianej listwy, powodowałby ruch posuwisty wzdłuż prowadnic konstrukcji. Przełożenie w tym przypadku byłoby bardzo niewielkie. Siłę z jaką napęd poruszałby daną osią można by wyrazić za pomocą wzoru (2.1). Natomiast dokładność pozycji danej osi sprowadza się do długości wycinka okręgu o promieniu równym promieniowi koła zębatego i kątowi równemu kątowi o jaki obraca się nasz silnik krokowy przy pojedynczym kroku. Oczywiście stosując sterowanie mikrokrokowe moglibyśmy osiągać większe dokładności odpowiednio do tego, jaki podział kroku będzie możliwy w wybranym przez nas sterowniku. Kolejnym argumentem przemawiającym przeciwko temu rozwiązaniu jest konieczność takiego zamontowania napędu, że każdy z trzech silników będzie umieszczony na elemencie ruchomym względem podstawy konstrukcji. Pozostałe dwa rozwiązania zakładają, że jeden z nich będzie nieruchomy względem podłoża. Ułatwia to w pewnym zakresie prowadzenie niezbędnych połączeń elektrycznych układu.

$$F = \frac{M}{r} \qquad (2.1)$$

gdzie:
F – siła
M – moment obrotowy silnika
r – promień koła zębatego

Kolejną rozważaną opcją była przekładnia z pasem zębatym. Takie rozwiązanie polegałoby na tym, że silnik z zębatką byłby połączony z ruchomą osią za pomocą pasa zębatego w ten sposób, że poruszając pasem zębatym, przyczepiony do niego element będzie poruszał się ruchem posuwistym wzdłuż ruchu pasa. Chodzi o zamianę ruchu obrotowego na ruch posuwisty. Rozwiązanie to jest powszechnie stosowane w małych robotach kartezjańskich, takich jak niewielkie drukarki 3D. W tym przypadku jedynie dwa silniki poruszałyby się względem podstawy konstrukcji, natomiast przełożenie oraz dokładność byłyby niemalże identyczne jak dla listwy zębatej. Przy czym dodatkowo na dokładność wpływałaby rozciągliwość paska zębatego, zarówno dynamiczna, jak i statyczna, czyli najzwyklejsze w świecie zużycie materiału na skutek mijania czasu i jego eksploatacji.

Trzecim rozwiązaniem była przekładnia śrubowa. Takie rozwiązanie zakłada montaż śruby trapezowej wzdłuż całej osi ruchu na łożyskach pozwalających na swobodny obrót z jak najmniejszymi oporami. Drugim elementem niezbędnym przy tej opcji byłoby przymocowanie do elementu poruszanego odpowiedniej nakrętki trapezowej obejmującej omówioną już śrubę. Obrót tejże śruby w miejscu spowodowałby przesunięcie nakrętki, a tym samym przymocowanego do niej ruchomego elementu wzdłuż osi ruchu. To rozwiązanie daje nam bardzo duże przełożenie, a co za tym idzie, siłę działającą na daną oś. Jednocześnie zwiększa się dokładność pozycji narzędzia frezującego. Powyższe dwie zalety skutkują jednak

zmniejszeniem szybkości ruchu. Natomiast robot ma być stosowany do prac amatorskich, a nie w wielkiej fabryce, więc zmniejszenie prędkości jest dopuszczalne przy zwiększeniu dokładności i siły wynikowej względem innych rozwiązań przeniesienia napędu.

Po rozważeniu powyższych opcji i możliwości, wraz z promotorem podjęto decyzję o wykorzystaniu trzeciego z opisanych rozwiązań, czyli przekładni ze śrubą trapezową. Wykonalne byłoby także połączenie różnych opisanych powyżej sposobów przeniesienia napędu, ale arbitralnie podjęto decyzję o unifikacji wszystkich trzech osi frezarki w tym aspekcie. Takie samo przeniesienie napędu można skonstruować korzystając ze śrub kulowych o większym skoku i mniejszym tarciu, natomiast rozwiązanie to jest zdecydowanie droższe od zwykłych śrub trapezowych. Szczególnie gdy mówimy o robocie, w którym najdłuższa oś ma ponad dwa metry długości. Tym samym, ze względu na ograniczony budżet, pozostano przy tańszym rozwiązaniu.

2.3. Koncepcja sterowania

Obok części mechanicznej i elektrycznej równie ważnym elementem projektu jest jednostka sterująca wszystkimi zachodzącymi w maszynie procesami. Komputer pozwalający na rzeczywiste wprawienie robota w ruch. Po właściwym zestawieniu części elektrycznej oraz mechanicznej, ta część układu oczekuje jedynie na sygnały cyfrowe podawane na odpowiednie wejścia sterowników. Te sygnały trzeba właściwie wygenerować właśnie w układzie sterującym.

Oprogramowanie powinno zapewnić kilka możliwości wykorzystania maszyny. Od funkcjonalności polegającej na ręcznym sterowaniu maszyną za pomocą odpowiednich przycisków lub joysticka, co może zostać również wykorzystane na przykład do ustawiania punktu początkowego realizacji zadanej trajektorii frezowania, do realizacji skomplikowanych ścieżek wyznaczonych na podstawie kształtu bryły, którą chcemy otrzymać. Ze względu na duże możliwości i wszechstronność użytku do realizacji założonych celów został początkowo wybrany minikomputer Raspberry Pi 4 z pamięcią RAM (ang. *random-access memory*) wynoszącą 2 GB. Urządzenie posiada zarówno wejścia i wyjścia cyfrowe jak i system operacyjny, na którym możemy uruchomić aplikacje okienkowe pozwalające na łatwą i intuicyjną obsługę. Możliwości napisanego programu są ograniczone tak naprawdę jedynie wyobraźnią oraz aktualnym zapotrzebowaniem. W tym rozwiązaniu potrzebowalibyśmy jedynie monitora oraz klawiaturę, którą można by podłączyć bezpośrednio do minikomputera. Jak się później okazało, zastosowanie Raspberry Pi do tego zadania nie było dobrym pomysłem przede wszystkim z powodu braku możliwości precyzyjnego odmierzania czasu rzędu kilkudziesięciu mikrosekund oraz braku możliwości zmiany stanów logicznych GPIO (ang. *general-purpose input/output*) z dużą częstotliwością.

Po napotkaniu problemów z minikomputerem Raspberry Pi zmieniono plan działania. Nowa koncepcja zakładała wykorzystanie tradycyjnego mikrokontrolera w połączeniu ze standardowym komputerem PC. Obydwa urządzenia komunikowały się za pomocą protokołu UART. Mikrokontroler zapewniał bardzo szybkie działanie i precyzyjne odmierzanie czasu,

natomiast komputer dawał możliwości realizacji aplikacji okienkowej w łatwy i przystępny sposób. To rozwiązanie sprawdziło się i zostało zaimplementowane w ostatecznej wersji projektu.

2.4. Przygotowanie trajektorii narzędzia frezującego

Budowana frezarka ma przede wszystkim wycinać z drewna pożądany przez nas kształt. Musimy więc pożądaną bryłę najpierw zaprojektować. O ile dla prostych kształtów i podstawowych figur geometrycznych zapewne w stosunkowo łatwy sposób dałoby się napisać program generujący trajektorię dla frezarki, o tyle dla brył bardziej skomplikowanych i nieregularnych byłoby to niezwykle trudne do zrobienia w skończonym czasie. Natomiast na rynku znajdują się gotowe rozwiązania pozwalające w pierwszej kolejności zaprojektować interesującą na bryłę, a następnie wygenerować ścieżkę narzędzia frezującego, pozwalającą na uzyskanie pożądanego efektu końcowego. Z powodu darmowej licencji studenckiej oraz wcześniejszej znajomości aplikacji przez autora pracy, do powyższego zadania został wybrany program Fusion360. Jest on również darmowy w pewnym ograniczonym zakresie dla osób nieposiadających statusu studenta, które wykorzystują go w celach niekomercyjnych.

Fusion360 to parametryczny program służący do projektowania trójwymiarowego. Posiada on jednak wiele innych funkcji. Najbardziej interesującą nas możliwością jest CAM. Jest to potężne narzędzie dające szerokie pole różnego rodzaju opcji i ustawiania swoich preferencji. Tak naprawdę już za pomocą kilku kliknięć możemy wygenerować ścieżkę z domyślnymi ustawieniami. Natomiast chcąc dostosować trajektorię do naszych potrzeb możemy to zrobić za pomocą przyjaznego dla użytkownika interfejsu.

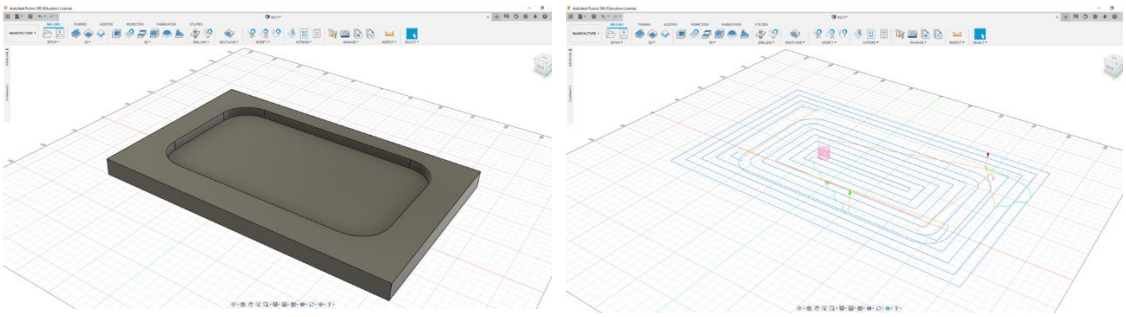
Bardzo ważną z naszego punktu widzenia opcją jest możliwość praktycznie dowolnego zdefiniowania końcówki roboczej. Każdy frez jaki zakupimy do frezarki możemy zapisać jako własne narzędzie w programie i ustawić jego parametry. W ten sposób generowana trajektoria będzie dostosowana dokładnie do posiadanego sprzętu i nie będzie trzeba jej zmieniać na późniejszym etapie generowania impulsów podawanych na sterowniki silników. Ponadto z bardzo przydatnych funkcji można wyróżnić możliwość zdefiniowania rozmiarów materiału z którego nasz kształt chcemy otrzymać. W ten sposób program również uwzględni wysokość od której frez musi rozpocząć pracę oraz gdzie nie może się poruszyć, aby nie wjechać w przeszkodę.

2.5. Zamiana trajektorii na proste instrukcje dla robota

Samo wygenerowanie trajektorii w programie niewiele daje naszej frezarce. Aby wydobyć interesujące nas informacje, musimy wygenerować odpowiedni plik zawierający tekstowe polecenia łatwe do zinterpretowania przez maszynę. Fusion360 daje możliwość zapisania trajektorii na bardzo wiele różnych sposobów w zależności od tego na jakiej maszynie będziemy ją odpalać. W naszym przypadku wybrano bardzo łatwy do zinterpretowania format „*Autodesk XYZ*” . Składa się on z kolejnych współrzędnych globalnych, do których frez ma się przemieścić, czyli otrzymujemy listę punktów, przez które końcówka robocza ma przejechać. Przy czym

ścieżka łącząca kolejne punkty leży na linii prostej. Na rys. 2.1. widzimy przykładową bryłę oraz wygenerowaną trajektorię pozwalającą otrzymać dany kształt.

a) b)



Rys. 2.1. Widok okna programu Fusion360

a) bryła docelowa, b) ścieżka narzędzia roboczego prowadząca do otrzymania zadanej bryły

Po zapisaniu trajektorii w formacie „*Autodesk XYZ*" plik wynikowy ma postać następujących po sobie współrzędnych w trzech płaszczyznach, co widzimy wyraźnie na rys. 2.2.

```
208.797;66.3;37
208.797;66.3;27
208.797;66.3;24
208.797;66.293;23.853
208.797;66.271;23.707
208.797;66.236;23.565
208.797;66.186;23.426
```

Rys. 2.2. Fragment pliku wynikowego z wygenerowaną trajektorią

Naszym zadaniem będzie napisanie programu, który zamieni informacje o odcinkach, po których frez ma się poruszać, na dyskretne sygnały podawane w odpowiednim momencie na właściwe wejścia sterowników silników krokowych poruszających układem.

2.6. Funkcjonalności głównego programu sterującego frezarką

Główny program, który ma być aplikacją okienkową z przyjaznym interfejsem, powinien obsługiwać kilka podstawowych funkcji. Pierwsza z nich to włączenie możliwości poruszania robotem za pomocą przycisków lub joysticka. Włączenie odpowiedniego trybu spowoduje oddanie sterowania do użytkownika. Ten tryb będzie przede wszystkim służył do testowania maszyny, bowiem ciężko będzie ręcznie wyfrezować jakikolwiek bardziej skomplikowany kształt. Niemniej jednak funkcjonalności tego trybu z pewnością zostaną wykorzystane w innych trybach i podprogramach. Chociażby w celu wyznaczenia punktu startowego dla frezarki. Kolejną możliwością będzie oczywiście wgranie przygotowanej wcześniej trajektorii i wykonanie jej przez naszą maszynę. Ponadto zostanie zaimplementowanych kilka dodatkowych możliwości wykonywania prostych frezów bez konieczności generowania trajektorii przez program niezależny. Na przykład podprogram do wyrównywania powierzchni. Po ręcznym ustawieniu końcówki robota w pożądanym położeniu i odpaleniu właściwej funkcji dojdzie do frezowania

po całej powierzchni na zadaną głębokość. Przy czym głębokość oraz rozmiar powierzchni będą parametrami zadanymi w odpowiednim okienku dialogowym programu.

3. DOBÓR ELEMENTÓW

Po generalnym wyborze rozwiązania należało dokonać szczegółowych decyzji co do konkretnie kupowanych elementów. Te z nich o większym znaczeniu zostaną opisane wraz z uzasadnieniem wyboru konkretnych egzemplarzy.

Dobór konkretnych elementów napędu oraz jego przeniesienia był ściśle skorelowany. Przy wyborze śrub trapezowych znaczenie miała średnica, która wiąże się bezpośrednio ze skokiem gwintu. Korzystając z dostępnych wykresów maksymalnych prędkości obrotowych w zależności od długości oraz średnicy śruby, wybrano egzemplarze o parametrach zawartych w tabeli 3.1. Śruby działające w dwóch dłuższych osiach mają zdecydowanie większą średnicę, natomiast ta działająca w osi pionowej ma jedynie 50 centymetrów długości i nie ma potrzeby, aby była bardziej wytrzymała.

Tabela 3.1. Parametry zakupionych śrub trapezowych

Liczba	Długość [mm]	Średnica [mm]	Skok śruby [mm]
1	2500	28	5
1	2000	28	5
1	500	16	4

Wykorzystanie śruby trapezowej daje nam istotne korzyści. Po pierwsze jest to kwestia siły generowanej bezpośrednio na osi robota. Wzorem (3.1) została opisana zależność tejże siły od momentu obrotowego silnika, skoku śruby oraz współczynnika sprawności. Ten ostatni wynika bezpośrednio z kąta wzniosu gwintu względem powierzchni śruby oraz współczynnika tarcia dynamicznego pomiędzy nakrętką i śrubą. Ciężko dokładnie wyznaczyć wartość sprawności, natomiast korzystając z przybliżeń [6] otrzymujemy wartości η = 0,28 dla śruby o mniejszej średnicy oraz η = 0,24 dla śrub o większej średnicy.

$$F = \frac{2 \cdot \pi \cdot 1000 \cdot C \cdot \eta}{P} \text{ [6]} \qquad (3.1)$$

gdzie:

F – siła osiowa na nakrętce [N]

C – moment obrotowy silnika [N*m]

P – skok śruby [mm]

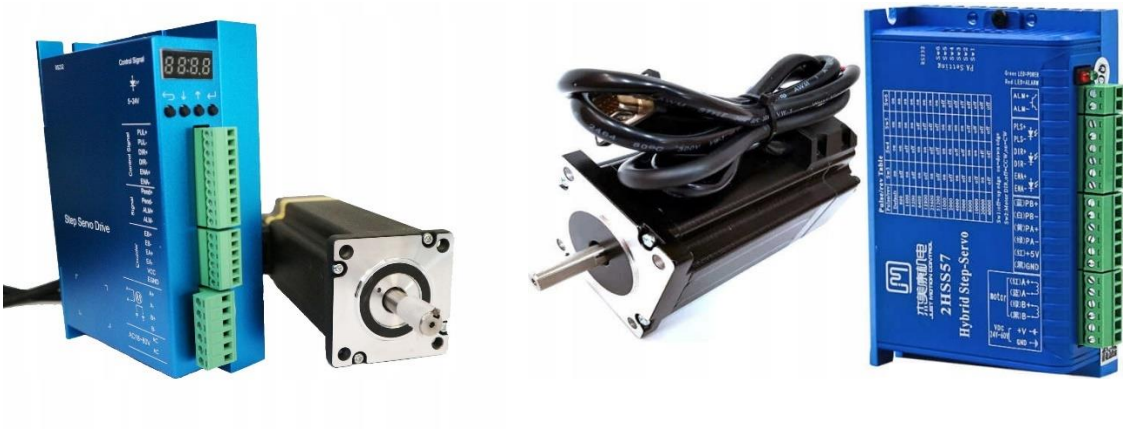
η – sprawność (odczytywana z tabeli, zależna od kąta wznosu śruby)

Po wyborze parametrów śrub trapezowych oraz na podstawie powyższych wzorów i danych możemy za pomocą liniowej zależności sprawdzić jaka siła powstanie na poszczególnych osiach robota w zależności od momentów obrotowych silników. Założono, że do poruszania śrubami odpowiednie będą dwa silniki o momencie trzymającym 5,5 Nm przeznaczone do poruszania osiami poziomymi oraz jeden o momencie trzymającym 3 Nm. Oczywiście większa siła byłaby dodatkowym atutem, musimy jednak pamiętać, że większa siła wymaga większej mocy silników, czyli lepszych sterowników oraz mocniejszych zasilaczy. Lepszy sprzęt jest zasadniczo droższy, więc należało zastosować rozwiązania optymalne.

Wykorzystując przybliżoną sprawność oraz posiadane dane możemy obliczyć na podstawie wzoru (3.1) siłę osiąganą na każdej z osi dla niskich prędkości obrotowych silników. Musimy natomiast pamiętać, że wraz ze wzrostem szybkości będzie malał moment obrotowy napędu. Tym samym, dla dwóch osi poziomych ze śrubami o jednakowej średnicy oraz identycznymi silnikami, obliczenia dają nam wartość generowanej siły na poziomie F = 1660 N. Natomiast dla osi pionowej z mniejszym silnikiem i śrubą o mniejszej średnicy siła F = 1320 N. Takie parametry powinny w zupełności wystarczyć do poruszania całym robotem. Nawet jeżeli założymy spadek momentu obrotowego silników o połowę przy większej prędkości posuwu, siła pchająca będzie na poziomie 700 N, co wciąż nie powinno w żaden sposób zakłócić pracy maszyny. Niestety nie udało się znaleźć dokładnych charakterystyk zależności momentu obrotowego od prędkości dla zakupionych silników, więc szybkość maksymalną frezowania ścieżek trzeba było wyznaczyć eksperymentalnie podczas fazy testów.

Same silniki krokowe niewiele mogłyby zrobić, gdyby nie ich sterowniki. Można te dwa elementy dobierać osobno, natomiast zakupione do realizacji projektu zestawy silnika ze sterownikiem dają pewność wzajemnej kompatybilności. Wykorzystano dwa takie zestawy widoczne na rys. 3.1.a) do poruszania osiami poziomymi X i Y oraz jeden zestaw ze słabszym silnikiem, widoczny na rys. 3.1.b), do obsługi osi pionowej Z.

a) b)



Rys. 3.1. Silniki krokowe w zestawach ze sterownikami [11, 12]
a) silnik 5,5 Nm, b) silnik 3 Nm

Drugą kwestią jest dokładność wyznaczania pozycji w wybranym rozwiązaniu. Zakupione silniki mają rozdzielczość 200 kroków na obrót. Natomiast sterowniki z zestawów nie dają możliwości sterowania za pomocą pełnych kroków. Dla zestawów z silnikami 5,5 Nm minimalny podział kroku to 2. Tym samym do pełnego obrotu wału należy podać na wejścia 400 impulsów. Dla silnika o momencie 3 Nm i jego sterownika minimalny podział to 4, co daje nam 800 impulsów na pełen obrót. Z kolei śruby cechują się skokiem 5 mm lub 4 mm w zależności od osi. Już na podstawie tych dwóch parametrów możemy obliczyć w łatwy sposób przesunięcie robota dla jednego kroku silnika. Wystarczy podzielić skok śruby przez liczbę kroków, co da nam dokładność w osiach poziomych Δd = 0,0125 mm oraz w osi pionowej Δd = 0,005 mm. Większa precyzja nie

jest potrzebna w naszym rozwiązaniu, ponieważ błędów rzędu setnych części milimetra nie da się gołym okiem w ogóle zauważyć. Tak naprawdę znacznie mniejsza liczba kroków na obrót również byłaby wystarczająca do planowanych zastosować frezarki. Natomiast jeżeli chcielibyśmy z jakichś powodów tę dokładność zwiększyć, to wystarczy zastosować sterowanie mikrokrokowe z jeszcze większym podziałem. Tym samym uzyskamy większą precyzję.

Rys. 3.2. Zasilacz impulsowy 36 V, 500 W, 14 A

Sterownik potrzebuje do pracy określonego zasilania. Dla zestawu z rys. 3.1.a) może być to zarówno napięcie zmienne AC z zakresu 18 - 80 V lub napięcie stałe DC z zakresu 24 – 110 V. Z kolei zestaw z rys. 3.1.b) można podłączyć jedynie do napięcia stałego DC 24 – 60 V. Stąd, aby nie rozdrabniać się na kilka urządzeń, zakupiono jeden zasilacz prądu stałego DC o napięciu znamionowym 36 V oraz wydajności prądowej wystarczającej do zasilenia wszystkich trzech sterowników jednocześnie. Wygląda on tak jak na rys. 3.2. Lepszym rozwiązaniem byłby zasilacz o większym napięciu, natomiast ze względów finansowych dokonano właśnie takiego wyboru. Niemniej jednak zakupione urządzenie sprawdza się w stopniu wystarczającym, gdyż nie ma potrzeby osiągania lepszych wyników pracy. Ponadto zastosowany w zasilaczu regulator pozwala nieco podnieść generowane napięcie.

Ponad to, co zostało opisane powyżej, do skonstruowania frezarki niezbędnych było jeszcze kilka elementów. Opisane poniżej urządzenia były dostępne razem z konstrukcją mechaniczną robota, więc postanowiono je wykorzystać. Należało jedynie odpowiednio uwzględnić i zaprogramować posiadany już sprzęt. Niemniej jednak bez tych elementów niemożliwe byłoby skonstruowanie frezarki, więc warto o nich krótko wspomnieć. Przede wszystkim mowa o silniku frezarki. Jest to motor o mocy 3,2 kW przy napięciu pracy 380 V oraz

prądzie 7 A. Znamionowa częstotliwość zmiany potencjału zasilania wynosi 400 Hz, co daje obroty na poziomie 24000 rpm. Sam silnik niewiele by zdziałał bez falownika, zapewniającego zasilanie o odpowiedniej amplitudzie oraz częstotliwości. Model widoczny na rys. 3.3. jest przystosowany do silników o mocy do 5,5 kW. Jest zasilany napięciem z przedziału 380-460 V oraz pobiera prąd do 20,7 A. Natomiast na jego wyjściu możemy wygenerować napięcie z zakresu 0-460 V o częstotliwości 0-400 Hz i prądzie do 12,6 A. Posiadany silnik frezarki wymaga również chłodzenia cieczą. Do tego celu została zakupiona pompa elektryczna wymuszająca obieg chłodziwa.

Rys. 3.3. Falownik 5,5 kW zastosowany do napędu wrzeciona

Dobór elementów był bardzo ważnym etapem projektu, ponieważ niewłaściwe parametry poszczególnych bloków układu mogłyby spowodować niepoprawne funkcjonowanie oraz duże opóźnienia w realizacji w przypadku konieczności zamawiania nowych części. Trzeba było dojść do kompromisów pomiędzy kosztami, a rzeczywistymi potrzebami zakupionych elementów.

4. REALIZACJA SPRZĘTOWA

Po wypracowaniu koncepcji całego projektu oraz zakupieniu odpowiednich części nadszedł moment, w którym należy wszystko razem połączyć w jedną całość. To zadanie okazało się bardziej pracochłonne niż się spodziewano. Niemniej jednak finalny efekt w postaci działającego robota został osiągnięty.

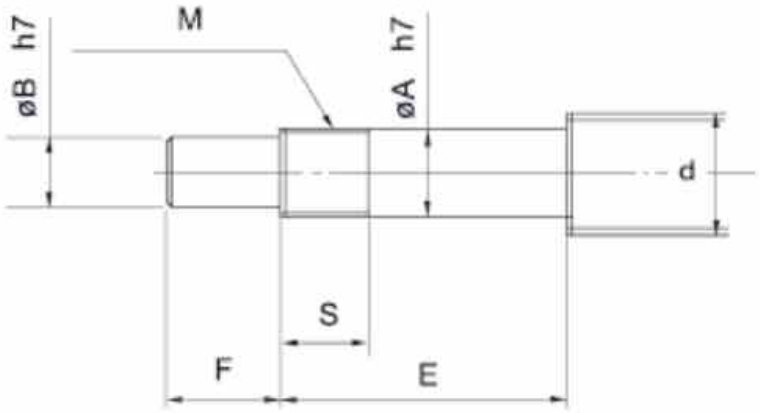
4.1. Śruby, silniki, sprzęgła

Elementami niezbędnymi do właściwego zamontowania śrub trapezowych były bloki łożyskujące (rys. 4.4.). Korzystając z poradnika [5] wybrano i zakupiono:

- po dwa bloki BK20 od strony napędowej oraz BF20 od strony podtrzymującej dla śrub działających w osiach poziomych X i Y,
- blok BK12 od strony napędowej oraz BF12 od strony podtrzymującej dla śruby działającej w pionowej osi Z.

Wybrane rozwiązanie wymaga odpowiedniej obróbki zakończeń śruby trapezowej. Zakupione łożyska zasadniczo definiują dokładnie jak trzeba to zrobić.

Na rys. 4.1. przedstawiono jak ma wyglądać zakończenie śrub trapezowych napędzających osie robota od strony silnika. Natomiast dokładne wartości parametrów dla śrub działających w poziomie przedstawia tabela 4.1. oraz dla śruby działającej w pionie tabela 4.2.



Rys. 4.1. Rysunek wykończenia śruby trapezowej pod bloki łożyskujące od strony napędowej [4]

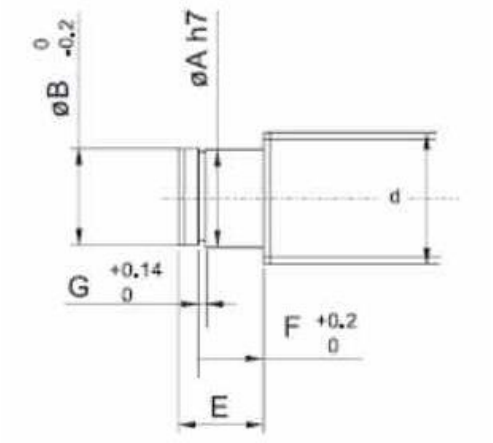
Tabela 4.1. Parametry zakończenia śrub działających w osi X i Y od strony silnika

Średnica zewnętrzna śruby trapezowej [mm]	Średnica zewnętrzna podtoczenia [mm]	[mm]	[mm]	[mm]	Gwint metryczny	
d	A	B	E	F	M	S [mm]
28	20	16	53	25	M20X1	15

Tabela 4.2. Parametry zakończenia śruby działającej w osi Z od strony silnika

Średnica zewnętrzna śruby trapezowej [mm]	Średnica zewnętrzna podtoczenia [mm]	[mm]	[mm]	[mm]	Gwint metryczny	
d	A	B	E	F	M	S [mm]
16	12	10	36	15	M12X1	14

Z kolei na rys. 4.2. widzimy jak ma wyglądać zakończenie śruby po stronie podtrzymującej, czyli przeciwnej do zamontowania silnika napędowego. Analogicznie jak powyżej, zestawiono w tabeli 4.3. oraz w tabeli 4.4. dokładne wymiary tegoż zakończenia w zależności od osi na której dana śruba będzie pracować.



Rys. 4.2. Rysunek wykończenia śruby trapezowej pod bloki łożyskujące od strony podtrzymującej [4]

Tabela 4.3. Parametry zakończenia śrub działających w osi X i Y od strony podtrzymującej

Średnica zewnętrzna śruby trapezowej [mm]	Średnica zewnętrzna podtoczenia [mm]	Długość podtoczenia [mm]	Rowek pod pierścień osadczy [mm]		
d	A	E	B	F	G
28	20	16	19	13.35	1.35

Tabela 4.4. Parametry zakończenia śruby działającej w osi Z od strony podtrzymującej

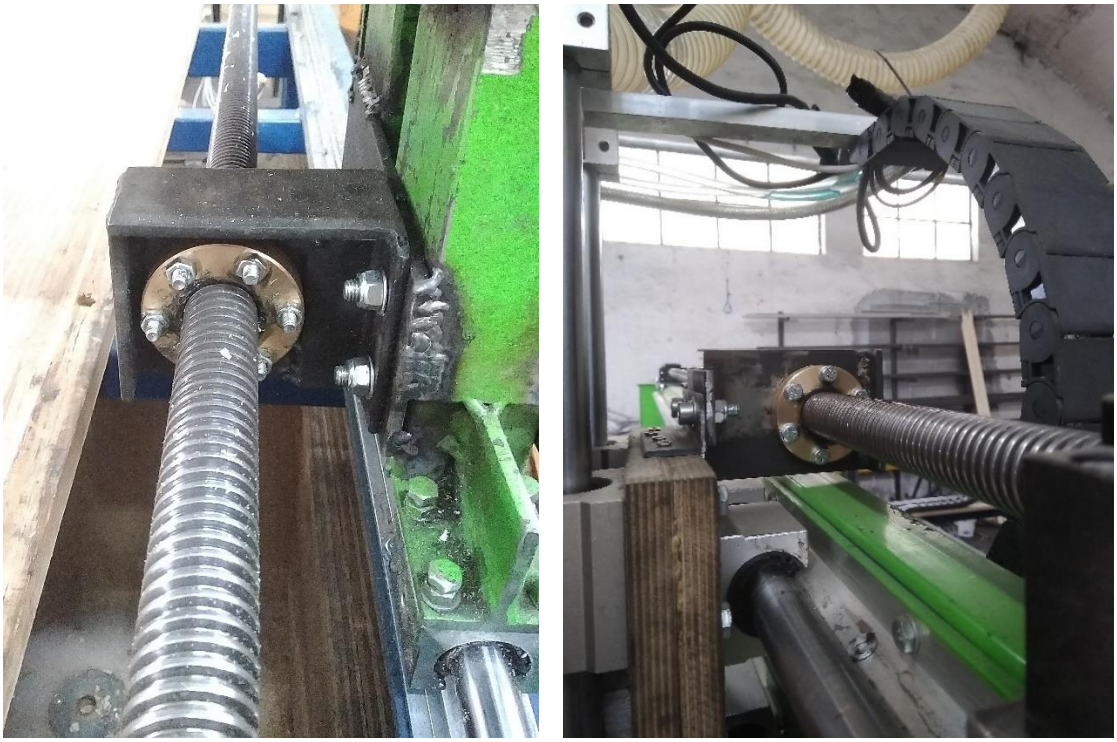
Średnica zewnętrzna śruby trapezowej [mm]	Średnica zewnętrzna podtoczenia [mm]	Długość podtoczenia [mm]	Rowek pod pierścień osadczy [mm]		
d	A	E	B	F	G
16	10	11	9.6	9.15	1.15

Aby obracające się śruby wprawiały robota w ruch potrzebne są odpowiednie nakrętki przymocowane do poruszanych osi (rys. 4.3.). Do ich zamontowania niezbędne było wykonanie odpowiednich uchwytów, które częściowo przyspawano lub przykręcono do pierwotnej konstrukcji robota. Natomiast zostało to zrobione w taki sposób, aby w co najmniej jednym miejscu uchwyt miał możliwość regulacji na śrubach mocujących. Było to niezbędne do dopasowania nakrętki do śrub trapezowych.

Kolejnym elementem wymagającym stałego zamocowania są silniki krokowe. Ich mocowanie zostało wykonane w dosyć prosty sposób za pomocą przeznaczonych do tego uchwytów. Napęd jest dosyć ustandaryzowany, więc łatwo było kupić wzmocnione kątowniki z otworami pasującymi do czoła silnika. Największym problemem zarówno w tym przypadku oraz przy mocowaniu śrub trapezowych było precyzyjne umiejscowienie wszystkich elementów. Po pierwsze oś silnika musi być w osi śruby. Po drugie, śruba musi znajdować się na obydwu końcach dokładnie na takiej samej wysokości oraz w tej samej odległości względem łożysk, po których porusza się dana oś. Po trzecie, nakrętka przymocowana na stałe do poruszanej osi

musi być dokładnie dopasowana do śruby, aby nie powodowała nadmiernego tarcia i działała poprawnie. Nie może dojść do sytuacji w której oś nakrętki nie pokrywa się z osią śruby, bo będzie to skutkowało większym zużyciem materiału oraz zwyczajnym blokowaniem się danej osi. Ciężko wykonać taki montaż z wielką dokładnością w amatorskich warunkach, więc potrzebnych było wiele poprawek i powolna kalibracja całego układu. Włącznie z koniecznością rozłożenia przeniesienia napędu całej osi w celu ponownego, lepszego jej zamontowania. Sama koncepcja mocowania uchwytów do nakrętek również ewoluowała w trakcie montażu podczas poszukiwania najlepszego rozwiązania. Niemniej jednak ostatecznie udało się w stopniu pozwalającym na prawidłową pracę dopasować wszystkie osie.

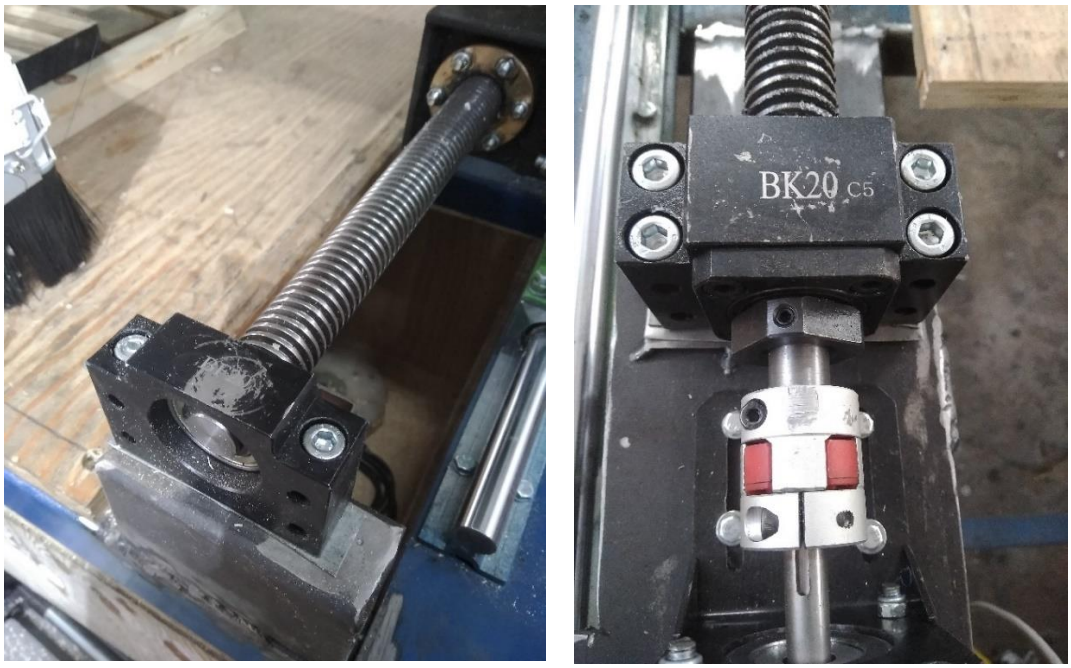
a) b)



Rys. 4.3. Mocowanie nakrętek przenoszących napęd do poszczególnych osi
a) oś Y, b) oś X

Śruby trapezowe po zamontowaniu w blokach łożyskujących należało połączyć z silnikami. Do tego wykorzystano sprzęgła bezluzowe kłowe podobne do widocznych na rys. 4.4.b). Są one dobrane dokładnie pod wymiary wału silnika oraz obrobionego zakończenia śrub. Dwa sprzęgła mają wejścia 10 mm z jednej strony oraz 16 mm z drugiej. Natomiast trzecie jest przygotowane do wału silnika o średnicy 8 mm oraz zakończenia śruby o średnicy 10 mm. Takie rozwiązanie pozwala z jednej strony w łatwy sposób połączyć oś silnika ze śrubą trapezową, a z drugiej strony jest tym elementem, który w razie awarii i blokady układu zepsuje się jako pierwszy. W ten sposób uchroni rotor silnika, łożyska oraz śrubę trapezową przed ewentualnymi uszkodzeniami. Efekt ostateczny połączenia wszystkich opisanych elementów widzimy na rys. 4.5.

a) b)



Rys. 4.4. Bloki łożyskujące

a) od strony podtrzymującej, b) od strony napędu z widocznym łożyskiem kłowym bezluzowym

a) b)

Rys. 4.5. Mocowanie napędu oraz jego przeniesienia

a) w osi X, b) w osi Z

4.2. Elektronika i elektryka

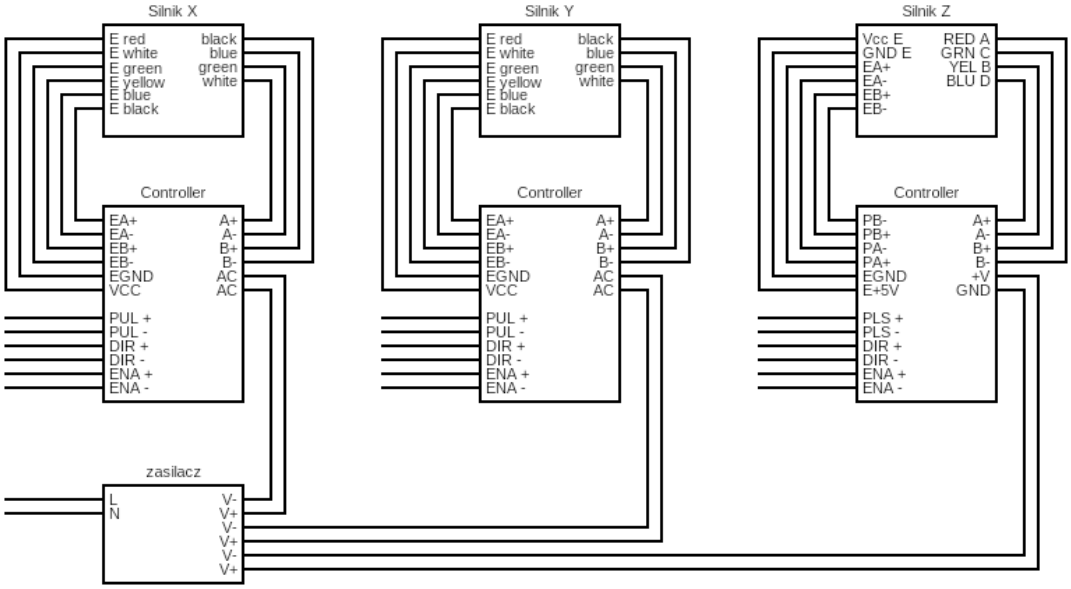
Cały robot wymaga dziesiątek metrów przewodów łączących wszystkie elementy elektroniczne ze sobą w jedną spójną całość. Pewnym wyzwaniem było poprowadzenie połączeń do silników krokowych znajdujących się na ruchomych osiach oraz do silnika prądu zmiennego napędzającego narzędzie frezujące robota. Zastosowanie prowadnic w miejscach ruchomych oraz korytek kablowych na odcinkach nieruchomych rozwiązało problem. W ten sposób wszystkie przewody są chronione i nie powodują żadnych większych oporów w działaniu maszyny. Ekranowany kabel zasilający silnik frezarki oraz ekranowane przewody do sygnałów cyfrowych pozwoliły na położenie wszystkich połączeń w jednym ciągu bez powodowania zakłóceń przesyłania danych z enkoderów do sterowników. Wszelkie przewody zostały sprowadzone do specjalnie do tego przygotowanej skrzyni, w której znajdują się również sterowniki, zasilacze oraz falownik. Wykorzystanie prowadnic oraz korytek kablowych na konstrukcji robota zostało przedstawione na rys. 4.6.

Rys. 4.6. Konstrukcja robota z widocznym sposobem prowadzenia przewodów

Sterowniki mają wiele wejść i wyjść. Do wyjść sterownika przede wszystkim podłączamy wyprowadzenia silnika krokowego na które będzie podawany prąd w odpowiednich sekwencjach. Możemy również odbierać informację zwrotną ze sterownika o poprawności działania. Musimy również zasilić enkoder. Natomiast na wejścia podajemy:

- zasilanie,
- sygnały zwrotne enkodera,
- sygnał zezwolenia na działanie ENA,
- sygnał kierunku ruchu DIR,
- impulsy generujące kroki silnika.

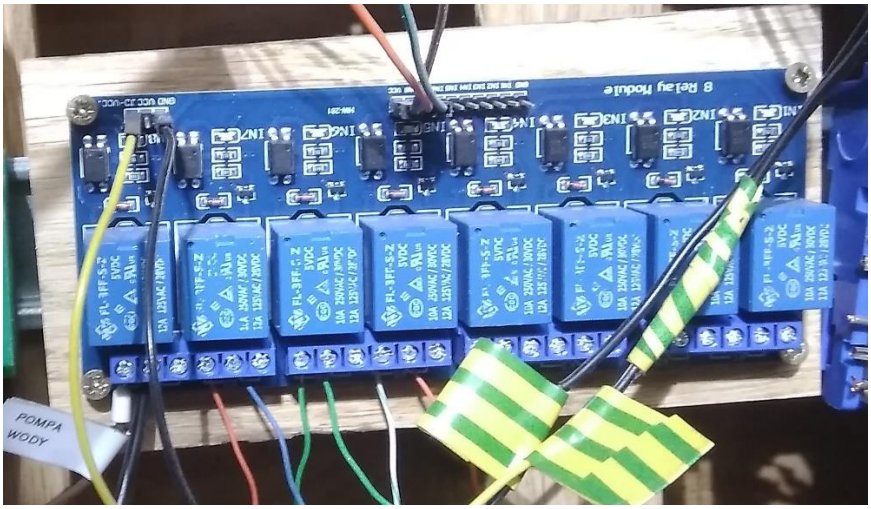
Na podstawie wszystkich powyższych sygnałów wejściowych, sterownik [10] może skutecznie i poprawnie wprawiać rotor silnika w ruch. Cały schemat połączeń tej części układu został rozrysowany na rys. 4.7.



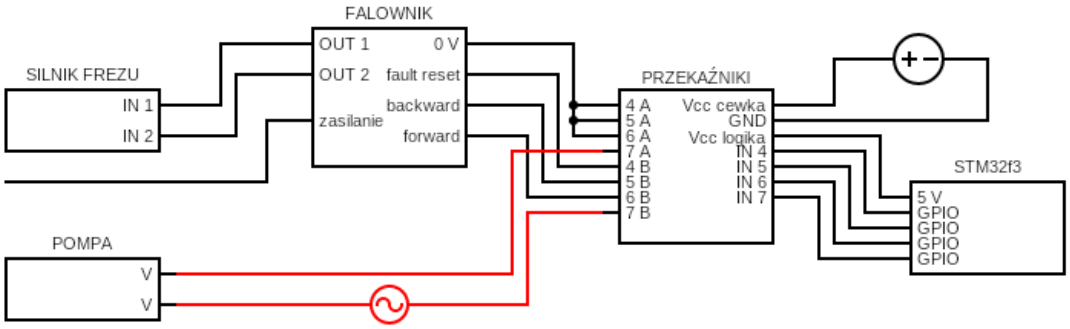
Rys. 4.7. Zaprojektowany schemat połączeń silników, sterowników oraz zasilacza

Kolejną częścią układu jest mechanizm załączania pompy chłodziwa oraz obsługa falownika za pomocą sygnałów cyfrowych. W obydwu tych przypadkach konieczne było wykorzystanie przekaźników. Dla pompy sprowadza się to jedynie do załączania oraz rozłączania obwodu napięcia przemiennego 230 V na skutek zadanego sygnału cyfrowego z płytki STM32F3DISCOVERY. Natomiast przy falowniku możliwa była różna konfiguracja sposobu jego wysterowania [2]. Zdecydowano się na rozwiązanie, w którym jeden sygnał załącza ruch wrzeciona w jednym kierunku, a drugi sygnał w kierunku przeciwnym. Trzeci z kolei służy do resetowania ewentualnych błędów falownika. Wybranie konkretnego sygnału jest realizowane poprzez zwarcie danego wejścia do masy falownika za pomocą przekaźników.

Do pompy oraz falownika potrzebujemy czterech takich elementów. Ze względu na niewielką różnicę ceny oraz przyszłościowe perspektywy, zakupiono od razu układ zawierający 8 przekaźników, co pozostawia możliwości dalszego rozwoju całego projektu na przykład o włączanie oświetlenia narzędzia frezującego. Wybrane przekaźniki, widoczne na rys. 4.8. posiadają cewki zasilane napięciem 5 V i wymagające prądu na poziomie 70 mA. Aby nie narażać mikrokontrolera na zbyt duży prąd, zastosowano dodatkowy zasilacz prądu stałego podłączany bezpośrednio do sieci elektrycznej w celu zasilenia cewek. Z płytki STM32F3DISCOVERY podajemy napięcie odniesienia jedynie do części cyfrowej układu przekaźników oraz załączamy odpowiednie z nich stanem niskim wystawianym na wyjście mikrokontrolera. Ta część układu została przedstawiona schematycznie na rys. 4.9., gdzie kolorem czerwonym został dodatkowo zaznaczony obwód zasilania 230 V pompy chłodziwa.



Rys. 4.8. Układ przekaźników sterowanych cyfrowo



Rys. 4.9. Zaprojektowany schemat połączeń pompy, falownika oraz przekaźników

Kwestią bezpieczeństwa było również zamontowanie i odpowiednie podłączenie przycisku awaryjnego odłączania zasilania robota. Wciśnięcie tak zwanego „grzyba”, widocznego na rys. 4.10. powoduje przerwanie obwodu zasilania dla falownika oraz silników krokowych. Tym samym następuje natychmiastowe zatrzymanie ruchu robota. Takie rozwiązanie jest niezbędne w tego typu projektach, gdyż rozpędzone wrzeciono frezarki może spowodować niemałe szkody jeżeli coś pójdzie nie tak i robot zboczy z zadanej ścieżki. Szczególnie w początkowej fazie użytkowania powinno się bacznie obserwować ruch maszyny i być gotowym do niezwłocznego przerwania jej pracy poprzez wciśnięcie przycisku bezpieczeństwa. Jest on zamontowany w widocznym i łatwo dostępnym miejscu na pulpicie przeznaczonym dla komputera sterującego.

Rys. 4.10. Przycisk bezpieczeństwa na pulpicie operatorskim

Cały projekt składał się z wielu elementów i przewodów, które trzeba było odpowiednio połączyć w jedną całość. Czasem niezbędne były długie rozważania dążące do opracowania możliwie najlepszego rozwiązania. Niemniej jednak efekt końcowy został osiągnięty na poziomie umożliwiającym poprawne działanie całego robota.

5. OPROGRAMOWANIE

Cały program do sterowania frezarką, zamiany trajektorii w postaci kolejnych punktów w przestrzeni na impulsy podawane do sterowników oraz generacji prostych kształtów został napisany własnoręcznie przez autora pracy. Nie są to rozwiązania lub całe bloki kodu ściągnięte z zewnętrznych źródeł. Stąd niektóre metody nie są idealne jak w profesjonalnym oprogramowaniu. Niemniej jednak pokazuje to, że da się realizować takie zadania w pewnym zakresie bez wielkiego zaplecza finansowego i osobowego.

Cały program do obsługi robota został napisany w języku Python na komputer PC oraz w języku C na mikrokontroler. Dokonano podziału na kilka osobnych plików zawierających stałe stosowane do obsługi frezarki, funkcje potrzebne do wykonania określonych zadań, klasy oraz skrypt obsługujący widok użytkownika. Poszczególne nietrywialne elementy i rozwiązania programistyczne zostaną zaprezentowane i omówione w celu pokazania jak oprogramowanie rozwiązuje postawiony problem sterowania.

5.1. Stałe wartości

Oprogramowanie zostało napisane w sposób umożliwiający w przyszłości łatwą zmianę parametrów, służącą dostosowaniu się do pracy w odmiennych układach fizycznych o innej charakterystyce. Zostało to zrealizowane poprzez generację osobnego pliku zawierającego jedynie wartości stałe przypisane do konkretnych i zrozumiałych dla użytkownika nazw. We wszystkich pozostałych skryptach korzystano z tychże nazw. Tym samym dokonując zmian w jednym pliku możemy ustawić różne parametry wykorzystywane w wielu innych miejscach programu. Są to przede wszystkim:

- rozmiar obszaru roboczego w trzech wymiarach,
- przesunięcie liniowe osi robota przy jednym impulsie podanym na sterownik silnika krokowego,
- współrzędne głównych bloków funkcjonalnych interfejsu użytkownika.

Takie rozwiązanie czyni oprogramowanie względnie uniwersalnym, a nie przystosowanym jedynie do tego konkretnego problemu.

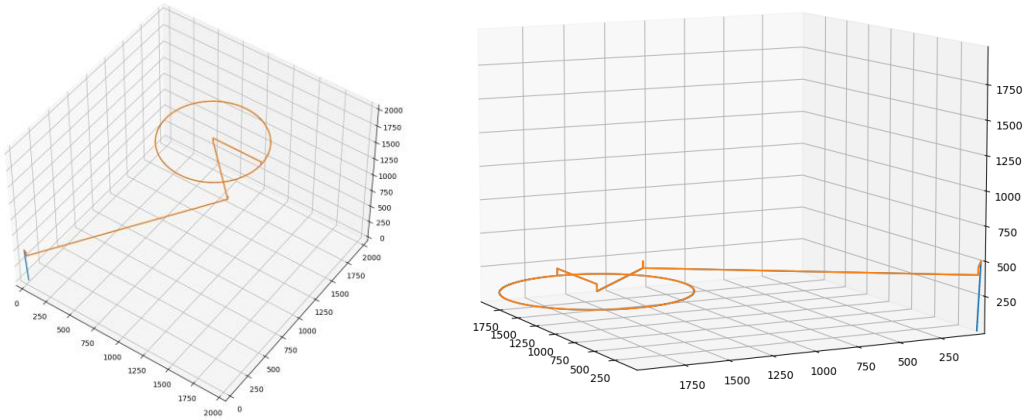
5.2. Funkcje

Funkcje napisane w ramach programu do sterowania frezarką realizują kilka zadań. Pierwsza z nich to *read_data(name, path='0').* Służy ona do wczytywania zewnętrznych danych wygenerowanych przez inne programy. Chodzi przede wszystkim o ścieżki narzędzia roboczego zapisane w formacie kolejnych punktów w przestrzeni trójwymiarowej, czyli w naszym przypadku pliki pochodzące z Fusion360. Jeżeli podamy samą nazwę, to program będzie szukał automatycznie w tej samej lokalizacji w której znajduje się skrypt. Natomiast podanie opcjonalnego parametru ścieżki dostępu spowoduje uwzględnienie innej lokalizacji pożądanego pliku.

Do wyświetlania wygenerowanej trajektorii służy funkcja *plot_toolpath(desired_path, tool_ticks, start_position, accuracy=ACCURACY)*. Pobiera ona listę punktów ścieżki docelowej, listę z wygenerowanymi już krokami silników oraz pozycję startową frezarki. Można w wywołaniu funkcji zmienić również dokładność, czyli przesunięcie liniowe osi robota dla jednego kroku silnika, ale zasadniczo ta wartość jest jedną ze stałych programu powiązaną z ilością kroków na pełen obrót śruby napędowej oraz jej skokiem. Na trójwymiarowym wykresie w nowym oknie są wyświetlane dwa wykresy. Jeden to połączone liniami prostymi punkty zadanej trajektorii. Drugi z kolei pokazuje nam jak będzie się poruszać narzędzie robocze frezarki na podstawie wygenerowanych kroków silników. Przy czym jeżeli obydwa wykresy się pokrywają, to widzimy kolor tylko jednego z nich. Funkcja sama generuje listę punktów jako pozycje zliczone, będące kolejnymi przesunięciami punktu początkowego o dyskretne wartości związane z obrotem silników w poszczególnych osiach.

Wykorzystanie powyższej funkcji pozwala sprawdzić przede wszystkim dwa elementy. Po pierwsze, czy ścieżka docelowa pokrywa się z ruchem frezu. Po drugie, możemy w przystępny sposób zobaczyć jak w rzeczywistości będzie wyglądała trajektoria narzędzia roboczego, ponieważ ciężko ją sobie dobrze wyobrazić jedynie na podstawie wpisywanych parametrów kolejnych figur ścieżki. Natomiast na trójwymiarowym wykresie, jak na rys. 5.1. widać dosyć wyraźnie jaki będzie efekt wykonania programu. Patrząc na wykres z boku widzimy kiedy narzędzie robocze dokonuje frezowania, a kiedy podnosi się, aby przejechać do kolejnego punktu nad obrabianym materiałem.

a) b)



Rys. 5.1. Obraz trajektorii narzędzia roboczego

Kolejne trzy funkcje są ze sobą ściśle powiązane i w zasadzie działają zawsze razem. Jest to blok kodu generujący kroki silników na podstawie kolejnych punktów trajektorii. Odgórną funkcją wywołującą pozostałe jest *generate_impuls(data, start_position).* Pobiera ona punkty tworzące ścieżkę oraz pozycję startową frezu, a zwraca listę zawierającą informację o kolejności z jaką silniki krokowe mają wykonywać pojedyncze kroki. Dla każdego pojedynczego odcinka, zaczynając od tego, który łączy pozycję startową frezu z pierwszym punktem ścieżki, wywoływana jest funkcja *coord_to_tics(first, second, accuracy=ACCURACY)*. Ta metoda pobiera

dokładną lokalizację narzędzia roboczego w danym momencie realizacji trajektorii oraz koordynaty punktu do którego chcemy się przemieścić. Ponieważ frez może znaleźć się jedynie w określonych, dyskretnych punktach przestrzeni, końcowa pozycja może się nieznacznie różnić od zadanego punktu o skończonej, ale nieporównywalnie większej dokładności położenia. Z tego powodu funkcja zwraca oprócz kroków silników potrzebnych do wykonania zadanego przemieszczenia również dokładną pozycję dyskretną od której rozpocznie ruch, gdy będzie poruszała się do kolejnego zadanego punktu.

5.2.1. Odnajdywanie kroków silników wzdłuż zadanej ścieżki

W tym miejscu zostanie omówiony sposób wyznaczania kolejnych kroków silników prowadzących do przemieszczenia wzdłuż prostej łączącej dwa zadane punkty.

KROK 1:

Wyznaczenie odległości między punktem startowym, a punktem końcowym. Wyznaczenie parametrów prostej w przestrzeni trójwymiarowej, przechodzącej przez obydwa punkty między którymi się przemieszczamy. Zakładając, że punkt początkowy ma współrzędne $P_1(x_0, y_0, z_0)$ oraz punkt końcowy $P_2(x_k, y_k, z_k)$ oraz korzystając z zależności (6.2), możemy wyznaczyć równanie parametryczne prostej (6.1).

$$l: \begin{cases} x = x_0 + t \cdot u_1 \\ y = y_0 + t \cdot u_2 \\ z = z_0 + t \cdot u_3 \end{cases} \qquad t \in \mathcal{R},\ u_1^2 + u_2^2 + u_3^2 > 0 \qquad (6.1)$$

$$u_1 = x_k - x_0 \qquad u_2 = y_k - y_0 \qquad u_3 = z_k - z_0 \qquad (6.2)$$

KROK 2:

Wygenerowanie sześciu punktów, w których frez może się znaleźć w następnej chwili czasu. Czyli sprawdzamy w jakim położeniu znajdzie się końcówka robocza zakładając, że tylko jeden z silników wykona krok w danej chwili czasu. Każdy z trzech silników może wykonać ruch w dwóch kierunkach, więc otrzymujemy właśnie sześć możliwości.

KROK 3:

Spośród znalezionych par punktów będących efektem ruchu wzdłuż tej samej osi, odrzucamy ten, który znajduje się dalej od punktu końcowego. Tym samym pozostają nam trzy możliwe punkty.

KROK 4:

Obliczenie odległości pomiędzy otrzymanymi trzema punktami, a prostą wyznaczoną w kroku 1. Sposób obliczania tej odległości zostanie omówiony osobno. Wybieramy punkt, który znajduje się najbliżej omawianej prostej. Jeżeli jego odległość od punktu końcowego jest mniejsza niż odległość pomiędzy aktualną pozycją, a punktem końcowym, to:

- jako nową pozycję przyjmujemy ten właśnie punkt,

- zapisujemy informację o tym który silnik i w którym kierunku musiał się poruszyć w celu osiągnięcia tego punktu,
- przechodzimy do kroku 2.

Jeżeli ten warunek nie jest spełniony, to oznacza, że odnaleziono dyskretny punkt w przestrzeni, który jest możliwie najbliższy do zadanego punktu końcowego. W takim wypadku kończymy algorytm, aby wywołać go dla kolejnego odcinka trajektorii.

Otrzymujemy na wyjściu listę mówiącą nam o kolejności i kierunku wykonywania kroków przez silniki, dających nam pożądaną ścieżkę narzędzia. Wywoływanie tego algorytmu przez funkcję nadrzędną wraz z łączeniem kolejno zwracanych list kroków daje nam całą trajektorię końcówki roboczej.

5.2.2. Obliczanie odległości punktu od prostej

Obliczanie odległości punktu $P(x_0, y_0, z_0)$ od prostej *l* jest kluczowym elementem algorytmu wyznaczania kroków silników. Zasadniczo składa się ono z kilku kroków:

- wyznaczenie płaszczyzny π prostopadłej do prostej l i przechodzącej przez punkt P,
- wyznaczenie punktu $O(x_p, y_p, z_p)$ przecięcia płaszczyzny π z prostą *l*,
- obliczenie odległości między punktem *P* i *O.*

Płaszczyznę można przedstawić za pomocą równania (6.3), gdzie wektor [A, B, C] jest jej wektorem normalnym oraz przechodzi ona przez punkt $P(x_0, y_0, z_0)$. Po wymnożeniu wartości w nawiasach otrzymujemy równanie w postaci (6.4), gdzie stała *D* jest opisana za pomocą (6.5) i możemy ją obliczyć, gdyż posiadamy wszystkie potrzebne dane.

$$A \cdot (x - x_0) + B \cdot (y - y_0) + C \cdot (z - z_0) = 0 \qquad (6.3)$$

$$Ax + By + Cz + D = 0 \qquad (6.4)$$

$$D = -Ax_0 - By_0 - Cz_0 \qquad (6.5)$$

Następnie podstawiamy pod zmienne *x, y oraz z* odpowiadające sobie współrzędne prostej przedstawionej parametrycznie w równaniu (6.1). Czyli szukamy punktu wspólnego płaszczyzny oraz prostej. Tym samym otrzymujemy wzór (6.6). Jedyną niewiadomą pozostaje współczynnik *t*. Aby go obliczyć należy dokonać odpowiednich przekształceń, których efekt zaprezentowano we wzorze (6.7). Po obliczeniu współczynnika *t*, korzystając z zależności opisanych w (6.1) otrzymujemy współrzędne punktu przecięcia prostej *l* z płaszczyzną π. W tym momencie pozostaje już jedynie obliczenie odległości *d* dwóch punktów w przestrzeni trójwymiarowej, co w bardzo prosty sposób można zrobić (6.8) jako pierwiastek kwadratowy z sumy kwadratów różnic odpowiadających sobie współrzędnych punktu *P* oraz *O.*

$$A \cdot (x_0 + u_1 \cdot t) + B \cdot (y_0 + u_2 \cdot t) + C \cdot (z_0 + u_3 \cdot t) + D = 0 \qquad (6.6)$$

$$t = -\frac{D + A \cdot x_0 + B \cdot y_0 + C \cdot z_0}{A \cdot u_1 + B \cdot u_2 + C \cdot u_3} \qquad (6.7)$$

$$d = \sqrt{(x_0 - x_p)^2 + (y_0 - y_p)^2 + (z_0 - z_p)^2} \qquad (6.8)$$

5.3. Klasy

Dwie najbardziej istotne z punktu widzenia porządku i praktycznego wykorzystania klasy, nie licząc tej odpowiadającej za interfejs, to obiekt reprezentujący frezarkę oraz generator ścieżek narzędzia. Są to osobne twory realizujące określone zadania.

Klasa reprezentująca frezarkę przechowuje informacje o aktualnym położeniu narzędzia roboczego. Jest ona aktualizowana na podstawie danych wysyłanych przez mikrokontroler. Po wysłaniu zapytania przez UART, płytka wysyła do komputera informację zwrotną z dokładną lokalizacją w momencie wysyłania. Im częściej wyślemy zapytania, tym częściej będziemy otrzymywać aktualizację położenia. Standardowo jest to wykonywane cztery razy na sekundę.

Kolejny blok kodu w klasie *Frezarka* dotyczy jej poruszania. W finalnie zastosowanym rozwiązaniu z wykorzystaniem mikrokontrolera STM32F3DISCOVERY zadanie tej klasy ogranicza się do wysyłania odpowiednich sygnałów sterujących oraz danych do płytki za pomocą łącza UART. Z jednej strony są to instrukcje zmieniające szybkość, włączające pompę lub falownik, organizujące wykonywanie zadanej trajektorii. Z drugiej zaś wysyłamy sekwencję informującą o kolejności oraz kierunku wykonywania kroków przez poszczególne silniki.

Drugą, bardzo ważną klasą jest *Generator*, który odpowiada za odnalezienie kolejnych odcinków tworzących pożądaną przez użytkownika trajektorię na podstawie podanych w programie parametrów. Przykładowo, zadajemy współrzędne środka okręgu oraz jego promień, a generator zwraca nam zbiór kolejnych punktów, które po połączeniu prostymi odcinkami tworzą tenże zadany kontur. Właśnie w tej klasie znajdują się parametry względnie rzadko zmieniane, ale konieczne do generacji określonych ścieżek. Takie jak wysokość płaszczyzny XY, głębokość wżeru w materiał podczas równania powierzchni oraz średnica frezu. Również tutaj jest przechowywana lista punktów całej jak dotąd wygenerowanej ścieżki.

Dla każdego możliwego do generacji kształtu z listy widocznej na rys. 5.4. b) istnieje odpowiadająca mu funkcja w klasie *Generator*. To tutaj został rozpisany cały mechanizm i zasady wyznaczania kolejnych punktów trajektorii. Funkcje pobierają potrzebne parametry poprzez okienka dialogowe w GUI i wykonują wyznaczone zadanie. Zasadniczo jest to miejsce, które można z największą swobodą rozwijać i dodawać kolejne możliwości. Nowe kształty, możliwość frezowania wnętrza obrysu, czy przygotowanie zestawu skalowalnych liter i cyfr to niektóre z możliwości rozwoju tejże klasy, a przez to również całego programu.

Nie widząc potrzeby opisywania każdej z tych funkcji, przybliżony zostanie jedynie opis zastosowanego rozwiązania dla wyrównywania powierzchni. Ten rodzaj działania pozwala nam zadać programowi prostokątne granice, w których generator utworzy taką ścieżkę narzędzia, aby frez przejechał po całej powierzchni wewnątrz konturu. Podając również głębokość wżeru oraz wysokość początkową i końcową otrzymamy ścieżkę skrawającą kolejne warstwy materiału

o wysokości równej głębokości wżeru. Program rozpocznie się na wysokości startu, a zakończy gdy osiągnie wartość zmiennej *Z* równą wysokości końcowej.

5.4. Interfejs

Klasa interfejsu tworzy widok aplikacji oraz pośredniczy między użytkownikiem, a kodem właściwym wykonującym potrzebne do działania operacje. Także przede wszystkim jest to obsługa przycisków oraz odczytywanie wprowadzanych danych. Tak oto przyciski do sterowania ręcznego robotem wysyłają do mikrokontrolera odpowiednie instrukcje powodujące ruch poszczególnych osi. Tworzenie ścieżki pobiera dane z okienek dialogowych i w zależności od wybranej opcji wykorzystuje odpowiednie możliwości klasy *Generator*. Zamiana punktów trajektorii na kroki silników korzysta z zewnętrznych funkcji napisanych w osobnym pliku. Symulacja ścieżki jest wykonywana za pomocą biblioteki *matplotlib* w celu graficznego przedstawienia efektu pracy programu. Funkcja wyświetlająca aktualną pozycję jest wywoływana przez zegar co 250 ms. Wysyła ona do mikrokontrolera zapytanie i aktualizuje wyświetlaną pozycję po jej odebraniu w informacji zwrotnej. Do tego dochodzi bardzo ważny element, czyli realizacja wygenerowanej już listy kroków silników. Tym samym przycisk START rozpoczyna ruch maszyny według zapisanej ścieżki. Pracę narzędzia możemy przerwać, kontynuować po przerwie lub zresetować i rozpocząć od początku. Poszczególne funkcje zwracają komunikaty do okienka tekstowego w widoku użytkownika, przez co widzimy co program właśnie zrobił oraz czy wykonał to poprawnie. Nie pozostawia to niedomówień oraz niepewności. Otrzymujemy szybką informację zwrotną.

Program jest również zabezpieczony przed wprowadzaniem niewłaściwych danych. Jeżeli podajemy wymiary, to muszą mieć one format liczby zmiennoprzecinkowej. Gdy z kolei wprowadzamy nazwę pliku do zaimportowania, to musi mieć ona odpowiedni format i rozszerzenie pliku. Są to elementy, gdzie bardzo łatwo i często przypadkiem wprowadzamy niepoprawne parametry. Zabezpieczenie przed takimi sytuacjami spowoduje, że ewentualne błędy zostaną wykryte, a użytkownik zostanie o tym poinformowany w czytelnym i przedmiotowym komunikacie. Zastosowanie tych mechanizmów sprawia, że program się nie zawiesza i nie kończy pracy przez zwykłą literówkę lub przypadkowe wciśnięcie niewłaściwego klawisza. Pozwala nam poprawić błędny parametr i spróbować ponownie.

5.5. Mikrokontroler STM32F3DISCOVERY

Finalny układ sterowania składa się z komputera, na którym działa podstawowy program z interfejsem użytkownika, oraz mikrokontrolera [4], zadającego odpowiednie sygnały cyfrowe. Obydwa urządzenia są ze sobą połączone za pomocą dwóch niezależnych modułów UART [1]. Przy czym jeden z nich jest jednokierunkowy i służy do przesyłania danych z komputera na mikrokontroler, natomiast drugi prowadzi komunikację dwustronną i odpowiada za sygnały kontroli i instrukcje do wykonania.

W mikrokontrolerze jesteśmy w stanie bardzo dokładnie odmierzać czas, dlatego generowane na komputerze sekwencje kroków silników są na nią przesyłane i dopiero tam

realizowane. Gdy w buforze danych znajduje się już tylko 500 elementów, płytka wysyła do komputera prośbę o kolejną paczkę informacji. Łatwiejsze byłoby wysłanie całej sekwencji kroków do mikrokontrolera przed rozpoczęciem ruchu frezarki, natomiast jest to niemożliwe ze względu na pamięć urządzenia. Dlatego musimy przekazywać te informacje w paczkach o określonej wielkości.

Możliwość bardzo dokładnego odmierzania czasu [11] w mikrokontrolerze została osiągnięta dzięki wewnętrznemu zegarowi o znanej częstotliwości oscylacji. Jego taktowanie osiąga 72 MHz. Podział sprzętowy częstotliwości powoduje, że licznik zegara inkrementuje się dokładnie co jedną mikrosekundę. W ten sposób zwykłe opóźnienie możemy realizować jako pętlę czekającą na osiągnięcie przez licznik zadanej wartości czasu oczekiwania. Natomiast w ten sposób blokujemy działanie całego programu na ten okres. Zastosowane rozwiązanie dla większych niż kilka mikrosekund opóźnień zakłada wyzerowanie oraz uruchomienie zegara w jednym miejscu kodu oraz sprawdzanie osiągnięcia przez licznik odpowiedniej wartości przez instrukcję warunkową w innym fragmencie programu.

5.5.1. Komunikacja UART

Do poprawnego funkcjonowania komunikacji należało napisać całą obsługę łącza szeregowego UART. O ile na komputerze PC z poziomu języka Python wysyłanie oraz odbieranie danych stanowi kilka linijek kodu, o tyle na mikrokontrolerze jest to bardziej skomplikowane i wymaga odpowiedniej obsługi [8]. Priorytetem jest szybkie i nieprzerwane działanie pętli głównej programu. Aby nie blokować jej pracy, zastosowano odbieranie danych w przerwaniach. Gdy do mikrokontrolera dojdzie cały bajt informacji, uruchamiane jest przerwanie, które tylko i wyłącznie zapisuje daną do bufora kołowego. Tym samym system natychmiast jest gotowy do odbierania kolejnych bitów na łączu szeregowym. Dopiero w głównej pętli programu sprawdzamy ile elementów jest w buforze. Jeżeli nie jest on pusty, to pobieramy najwcześniej odebrany bajt, a następnie wykonujemy działanie do niego przypisane. Krótko mówiąc samo odbieranie danych z ich odczytywaniem i interpretacją są zasadniczo rozdzielone. Program czyta kolejne bajty informacji dopiero wtedy, kiedy ma czas. Jest to spowodowane koniecznością wykonywania wielu innych fragmentów kodu w czasie rzeczywistym. Gdybyśmy po odebraniu danych od razu je wszystkie interpretowali, to przy wysyłaniu ciągu kilu tysięcy bajtów powstawałyby przerwy pętli głównej nie do przyjęcia. Szczególnie gdy mówimy o synchronizacji wysyłania impulsów na sterowniki silników z dokładnością kilku mikrosekund.

Podobnie wygląda sytuacja przy wysyłaniu danych z mikrokontrolera do komputera PC. Użytkownik jedynie wpisuje do bufora dane oraz uruchamia proces związany z przerwaniami, który dokona transmisji informacji. Do niezależnie działającego bloku funkcjonalnego płytki STM32F3DISCOVERY jest wysyłany jeden bajt oraz ustawiana odpowiednia flaga. Program wraca do wykonywania głównej pętli. Dopiero gdy nadejdzie przerwanie informujące o przesłaniu ostatniego bitu informacji, do modemu UART wysyłany jest kolejny bajt. Podsumowując, system wysyła daną do osobnego układu i wraca do wykonywania swoich zadań. Gdy ten układ zgłosi wysłanie całego bajtu, system robi krótką przerwę na załadowanie kolejnej danej i wraca

do swoich zadań. W tej sposób w żadnym miejscu nie dochodzi do dłuższego blokowania programu przez co mamy pewność, że każda z instrukcji w pętli głównej programu zostanie wykonana nie później niż określona chwila czasu.

Warto krótko wspomnieć o buforze kołowym zastosowanym w powyższym rozwiązaniu. W językach programowania wyższego poziomu możemy po prostu stworzyć listę elementów i praktycznie dowolnie dodawać do niej kolejne pozycje oraz usuwać stare. Istnieją czasem gotowe klasy reprezentujące kolejki danych. Natomiast w programowaniu niskopoziomowym jakie jest zastosowane dla płytki STM32F3DISCOVERY należało zastosować inne rozwiązanie. W przypadku bufora kołowego należało zdefiniować stały obszar pamięci przeznaczony do tego zadania oraz zaimplementować obsługę z wykorzystaniem wskaźników. Jeden z nich, nazywany *głową,* zawiera adres najnowszej danej, natomiast drugi, zwany *ogonem*, najstarszej. Wpisanie nowej danej powoduje zwiększenie wskaźnika *głowy,* natomiast odczytanie tej najwcześniej odebranej skutkuje przesunięciem *ogona* o jedną komórkę pamięci do przodu. Aby był to rzeczywiście bufor kołowy, po dojściu do końca zadeklarowanego obszaru dane się zawijają i lądują na początku bufora. Tak jakby pierwszy element był bezpośrednio po ostatnim.

5.5.2. Pętla główna programu

Pętla główna programu działa bez przerwy, gdy mikrokontroler jest zasilany. Zasadniczo składa się z szeregu instrukcji warunkowych i działa na zasadzie ustawiania i resetowania poszczególnych flag. Przykładowo, sprawdzamy ilość elementów w buforze zawierającym instrukcje. Jeżeli nie jest on pusty, to pobieramy najwcześniej odebrany bajt. W przeciwnym wypadku przechodzimy do kolejnego fragmentu kodu. Gdy odczytaliśmy na przykład instrukcję poruszania którejś osi, to ustawiamy flagę zezwalającą na ruch oraz włączamy licznik czasu, po czym wracamy do pętli głównej. W innej z instrukcji warunkowych ustawiona wyżej wspomniana flaga oraz przekroczenie przez licznik zadanej wartości spowoduje wysłanie sygnału do mikrokontrolera oraz reset licznika. W ten sposób co do zasady jest zbudowany cały program na płytce STM32F3DISCOVERY.

Pętla główna sprawdza ilość elementów w buforze danych oraz buforze instrukcji. Jeżeli zawierają jakieś elementy, to dochodzi do ich interpretacji. Dla krótkich poleceń, takich jak włączenie pompy lub falownika, instrukcje są wykonywane natychmiast. Natomiast wszelkie polecenia związane z ruchem silników powodują uruchomienie nietrywialnego systemu flag, licznika oraz instrukcji warunkowych realizujących tenże ruch. Należało zadbać o to, aby czas od ustawienia sygnału kierunku do wysłania impulsu kroku był odpowiednio długi. Również czas trwania samego impulsu zgodnie z instrukcją sterownika musi trwać odpowiedni okres czasu. Ponadto w momencie odmierzania czasu pomiędzy kolejnymi impulsami program nie powinien się blokować i czekać bezproduktywnie. Pętla główna programu powinna cały czas się wykonywać aż do momentu osiągnięcia przez licznik zadanej wartości. Zaimplementowanie tych wszystkich warunków poskutkowało stworzeniem dosyć skomplikowanego systemu flag i instrukcji warunkowych. Niemniej jednak zadanie to zostało zakończone sukcesem i program działa poprawnie.

5.6. Oprogramowanie - GUI

Okno użytkownika służące do obsługi frezarki zostało stworzone w możliwie czytelny i jasny sposób. Na rys. 5.2. został przedstawiony ogólny wygląd interfejsu w całości. Posiada on następujące elementy:

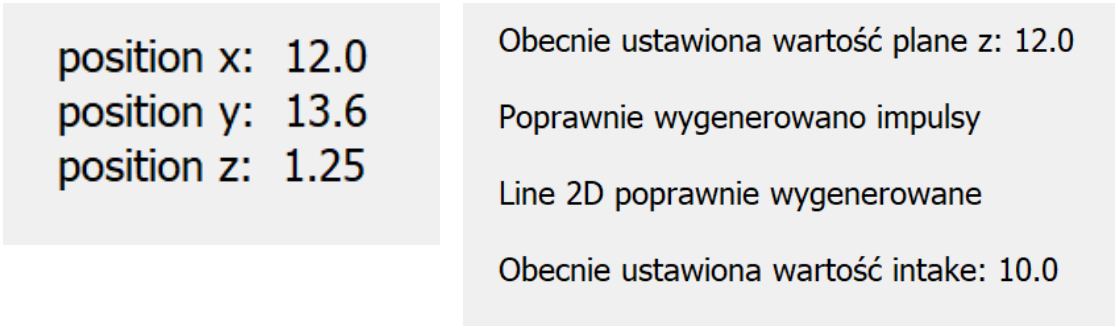
- przyciski do ręcznego poruszania robotem w każdej z trzech osi,
- informację o aktualnym położeniu środka frezu,
- okno komunikatów zwracanych przez program,
- trzy przyciski do ustawiania parametrów generacji ścieżek,
- część odpowiedzialną za wybór generowanej ścieżki oraz jej generację,
- blok generacji impulsów cyfrowych oraz rozpoczynania, zatrzymywania i resetowania pracy frezarki.

Poszczególne elementy zostaną po kolei szczegółowo omówione.

Rys. 5.2. Wygląd ogólny całego okna programu

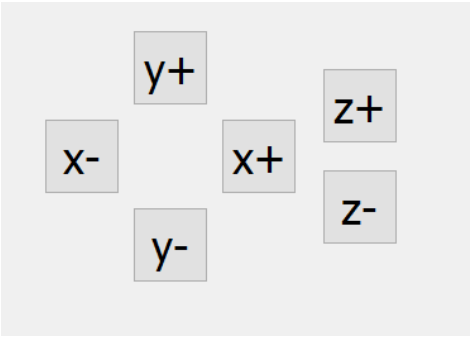
GUI posiada dwa elementy zawierające informacje zwracane przez program pełniące funkcję informacyjną. Jest to blok wskazujący aktualną pozycję naszej maszyny jako zestaw trzech koordynat kartezjańskich (rys. 5.3. a)) oraz blok wszelkich pozostałych komunikatów (rys. 5.3. b)). Są to informacje o poprawnym wykonaniu danej instrukcji, o obecnie ustawionej wartości interesującego nas parametru lub wiadomości o błędach będących skutkiem niepoprawnie wprowadzanych danych. Dzięki tym informacjom wiemy dokładnie, czy wykonało się zlecone przez nas zadanie lub dlaczego nie zostało wykonane.

a) b)



Rys. 5.3. Komunikaty z programu dla użytkownika
a) obecna pozycja frezu, b) informacje o wykonanych czynnościach

Kolejnym elementem widoku użytkownika jest zestaw przycisków pozwalających na ręczne poruszanie robotem, widoczny na rys. 5.4. Zgodnie z oznaczeniami, wciśnięcie odpowiedniego z nich powoduje ruch w danej osi. W połączeniu z informacją o aktualnym położeniu końcówki frezującej możemy wykorzystać tą funkcjonalność jako swego rodzaju narzędzie pomiarowe i na podstawie zebranych punktów wygenerować interesującą nas ścieżkę ruchu realizowaną już automatycznie przez program.



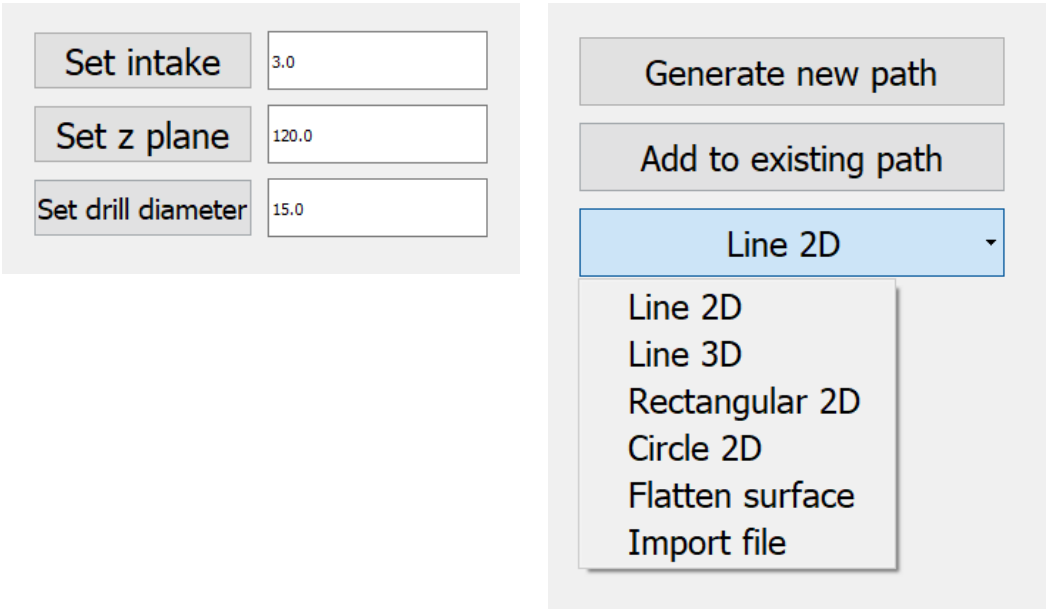
Rys. 5.4. Układ przycisków do poruszania ręcznego

Układ generacji trajektorii narzędzia roboczego zawiera kilka elementów. Na rys. 5.5. a) widzimy trzy przyciski wraz z polami tekstowymi do ustawiania następujących wartości:

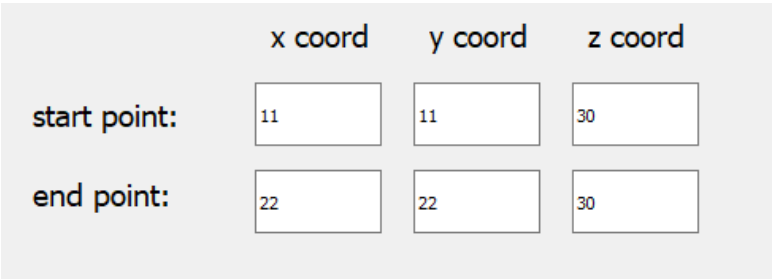
- głębokości kolejnych warstw materiału skrawanego podczas wyrównywania powierzchni – informacja o tym jak głęboko w materiał frez może się zagłębić,
- współrzędnej *z* płaszczyzny XY – na tej wysokości program wygeneruje ścieżkę wszystkich kształtów z dopiskiem „*2D*” na końcu,
- średnicy frezu – ma to znaczenie gdy chcemy wyrównać powierzchnię w celu ustalenia odległości kolejnych równoległych ścieżek narzędzia roboczego.

Na rys. 5.5. b) widzimy z kolei oczywiste ze względu na nazwę przyciski do generacji nowej ścieżki oraz dodania kolejnego elementu do istniejącej trajektorii, a także listę dostępnych kształtów. Wybierając interesującą nas opcję program wyświetli pola do których możemy wprowadzić dane geometryczne danego kształtu w celu dodania go do ścieżki. Przykładowy obraz po wybraniu opcji „Line 3D” został zaprezentowany na rys. 5.6.

a) b)

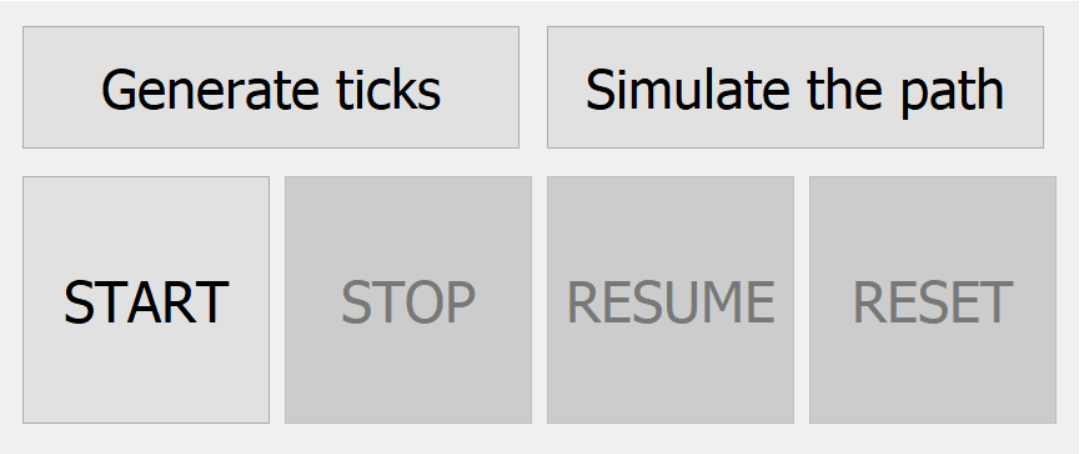


Rys. 5.5. Układ do generowania trajektorii
a) ustawianie stałych parametrów, b) generowanie oraz wybór ścieżki



Rys. 5.6. Okno wprowadzania współrzędnych wybranych kształtów

Ostatnim elementem GUI jest blok generacji oraz symulacji impulsów podawanych na sterowniki silników krokowych oraz obsługi wykonywania trajektorii. Zgodnie z nazwami widocznymi na rys. 5.7. dwa górne przyciski służą do przekształcenia wygenerowanych punktów ścieżki narzędzia na konkretne impulsy dyskretne podawane na wejścia sterowników oraz do wyświetlenia śladu pozostawionego przez frez. Z kolei dolny rząd przycisków służy do obsługi już wygenerowanych impulsów w czasie. Przycisk *START* rozpoczyna wykonywanie zadanej trajektorii. Przyciskiem *STOP* dokonujemy zatrzymania realizacji procesu, natomiast wciskając *RESUME* możemy pracę kontynuować. Przycisk *RESET* powoduje ustawienie parametrów wewnątrz programu na wartości początkowe.



Rys. 5.7. Blok generacji i symulacji impulsów oraz obsługi wykonywania trajektorii

6. TESTY

Testy całego projektu miały miejsce praktycznie przez cały czas realizacji. Początkowo obejmowały one całe oprogramowanie. W trakcie jego pisania wielokrotnie kod był odpalany i poszczególne jego fragmenty sprawdzane. Sam rdzeń programu odpowiadający za zamianę punktów trajektorii na kroki silników był bardzo trudny do sprawdzenia podczas pisania. Dało się sprawdzić jego poprawność dopiero po napisaniu wielu funkcji i połączeniu ich odpowiednimi zależnościami. Niemniej jednak ta część zadziałała poprawnie przy pierwszym uruchomieniu. Zasadniczo testy samego oprogramowania były dosyć łatwe do wykonania, a wykryte błędy dało się względnie szybko naprawić.

Zasadniczo starano się robić testy jak najczęściej oraz tak szybko jak było to możliwe. Gdy zostały zakupione silniki, sterowniki oraz zasilacz, od razu sprawdzono ich funkcjonowanie zanim jeszcze zostaną zamontowane na stałe w konstrukcji maszyny. Gdy zamontowano silnik oraz śrubę na jednej z osi, przeprowadzono niezwłocznie testy jej funkcjonowania, zanim zostanie zamontowany napęd na kolejnej osi. Takie podejście pozwoliło względnie szybko wykryć problem ze sterowaniem za pomocą minikomputera Raspberry Pi.

Początkowa koncepcja obejmowała sterownie robotem z wykorzystaniem popularnego Raspberry Pi. Moglibyśmy na tym urządzeniu zarówno uruchomić aplikację okienkową, napisaną w języku Python, oraz wykorzystać wejścia i wyjścia cyfrowe GPIO. Problem pojawił się, gdy podłączono silnik krokowy i doszło do próby jego poruszania z wykorzystaniem minikomputera. Przy niewielkich prędkościach rzędu jednego obrotu na sekundę wał obracał się w dosyć płynny sposób, chociaż można było usłyszeć, że od czasu do czasu pojawia się pewna nieregularność. Niestety zwiększanie prędkości obrotowej prowadziło do pracy bardzo nieregularnej i niestabilnej. Silnik szybko się rozpędzał, po czym nagle hamował. Zdecydowanie nie była to poprawna praca. Najpierw odrzucono możliwość niepoprawnej pracy sterownika lub defektu samego silnika. W tym celu uruchomiono dostępny w sterowniku tryb wewnętrznej generacji impulsów. Silnik zadziałał poprawnie rozpędzając się do pięciu obrotów na sekundę bez żadnego problemu. Wykluczono tym samym niepoprawne działanie tych elementów. Postanowiono spróbować zadawać sygnały sterujące za pomocą płytki STM32F3DISCOVERY. Napisano prosty program, który generował na wyjściu falę prostokątną. Takie rozwiązanie sprawdziło się znakomicie. Silniki dochodziły do prędkości zdecydowanie większych niż wystarczające do poruszania robotem. W tym momencie podjęto decyzję o zmianie koncepcji i wykorzystaniu mikrokontrolera STM32F3DISCOVERY w połączeniu z komputerem PC do realizacji projektu.

Najprawdopodobniej problem z Raspberry Pi polegał na braku możliwości dostatecznie precyzyjnego odmierzania czasu, co jest spowodowane wysokopoziomowym charakterem minikomputer oraz brakiem zegara czasu rzeczywistego - RTC (ang. *real-time clock*). Mikrokontroler STM32F3DISCOVERY daje nam możliwość bardzo precyzyjnego pomiaru czasu, co pozwala na właściwe sterowanie silnikami i odmierzanie momentów z dokładnością jednej mikrosekundy. Tym samym komputer PC spełnia rolę wysokopoziomowego interfejsu,

generatora ścieżek i sekwencji impulsów, natomiast mikrokontroler jest urządzeniem wykonawczym, które wystawia na wyjścia cyfrowe sygnały w odpowiednim harmonogramie czasowym.

Kolejny problem pojawił się podczas testowania poruszania pierwszej ze złożonych i zamontowanych osi. W niektórych położeniach dało się osiągnąć całkiem duże prędkości, natomiast szczególnie przy krańcach obszaru roboczego dochodziło do blokowania się silników oraz pojawiania się błędów na sterownikach. Przy małych prędkościach z kolei dało się dojechać prawie wszędzie. Próba zmiany zamocowania nakrętki na śrubie poskutkowała jeszcze gorszymi efektami. Najpierw postanowiono sprawdzić, czy chodzi po prostu o prędkość obrotu śruby, czy problem jest z poruszaną osią. W tym celu odłączono daną oś od napędu i wprowadzono silnik w ruch. W tym przypadku nie było najmniejszych problemów i śruba obracała się bez zakłóceń. Podczas badania sprawy odkryto, że łożyska tej osi robota są za bardzo dociśnięte do szyn. Był to pierwszy element do poprawy. Niemniej jednak postanowiono dodatkowo rozłożyć całe przeniesienie napędu na tej osi i złożyć je ponownie, starając się zrobić to jak najdokładniej. Te dwa zabiegi dały pożądany efekt poprawnej pracy przy przyzwoitych prędkościach.

Największy sprawdzian, zwieńczający dzieło, nastąpił gdy wszystkie elementy zostały już zamontowane i podłączone. Dopiero w tym momencie można było zweryfikować w rzeczywistości poprawność działania programu i przetestować frezarkę w praktyce. Zadać jej konkretną trajektorię i zobaczyć jak wykonuje ścieżki w drewnie.

Rozpoczęto od sprawdzania realizacji zadanych trajektorii bez włączania silnika frezarki. Pierwszy dzień takowych testów wykazał błędy w oprogramowaniu dotyczące pomijania niektórych z zadanych punktów oraz dzielenia przez zero podczas generowania impulsów. Wymagało to analizy kodu krok po kroku, ale udało się do następnego dnia testów wyeliminować te dwa błędy. Geneza powstawania błędu dzielenia przez zero jest dosyć skomplikowana, ale po wykryciu problemu udało się go całkiem szybko usunąć. Natomiast kwestia pomijania punktów trajektorii była związana z założeniem, że jeden impuls podany na sterownik powoduje przesunięcie liniowe o tej samej wartości dla każdej z osi. W takim przypadku kod działałby poprawnie. Problem pojawił się, ponieważ sterownik silnika działającego w osi Z ma minimalny podział kroków równy 800 na obrót, natomiast silniki w pozostałych osiach mają możliwość podziału równą 400. Te ograniczenia sprzętowe w połączeniu z mniejszym skokiem śruby w osi pionowej spowodowały odkrycie wrażliwości programu na różne długości przesunięć liniowych dla jednego impulsu podawanego na sterownik. Samo znalezienie problemu zajęło dużo czasu, natomiast jego naprawienie było stosunkowo proste. Po tych poprawkach program zaczął co do zasady funkcjonować poprawnie.

Po wyeliminowaniu tych błędów można było przystąpić do kolejnego dnia testów. Generowane ścieżki wykonywały się już poprawnie, więc włączono pompę chłodziwa oraz uruchomiono silnik frezarki. Jednym z głównych celów projektu była możliwość wykorzystania robota do wyrównywania powierzchni. Dokładnie to zrobiono. Na rys. 6.1. widzimy kawałek deski z nierówną powierzchnią. Po zastosowaniu programu do wyrównywania powierzchni otrzymano

efekt z rys. 6.2. Widoczne czarne pasy są związane z nieidealnym w momencie testów ustawieniem silnika. Ponieważ frez do wyrównywania ma stosunkowo dużą średnicę, nawet niewielkie odchylenie wału od pionu powoduje przypalanie drewna. Może to być również związane z frezem nienajlepszej jakości.

Rys. 6.1. Blok drewna przed obróbką za pomocą robota

Rys. 6.2. Blok drewna po wyrównaniu powierzchni za pomocą frezarki

Gdy otrzymaliśmy płaską powierzchnię, można było przejść do testowania zadanych kształtów. Na rys. 6.3. widzimy dwa kształty, które miały być okręgami. Natomiast większy z nich uległ zniekształceniu na skutek napotkania granicy obszaru roboczego, która była wtedy ustawiona w wyraźnie widocznym miejscu – tam, gdzie okrąg zaczął się zniekształcać. Widoczne ścieżki dojazdu do zadanych trajektorii są wykonane celowo, gdyż zamontowany w tamtym momencie frez nie ma możliwości wchodzenia w materiał od góry, więc musiał tego dokonać od krawędzi bloku.

W kolejnym etapie zamontowano frez o dużo mniejszej średnicy, umożliwiający wchodzenie w materiał z góry. Widzimy go zamontowanego do rotora silnika na rys. 6.4.b). Mniejsza średnica frezu pozwoliła na wykonanie głębszych ścieżek w materiale. Aby nie marnować kolejnych bloków drewna, testy wykonywano na jednym. Niestety przez to zrobił się lekki bałagan, natomiast na rys. 6.4.a) możemy dostrzec wyfrezowany okrąg, odcinki pod różnymi kątami oraz początki ścieżek wykonane przez wjechanie pionowe w materiał. Dla wąskiego frezu żadne zaciemnienia materiału już się nie pojawiały. Dodatkowo na rys. 6.5. została z bliska przedstawiona powierzchnia materiału po wykonaniu testowych trajektorii. Widzimy różne głębokości ścieżek i generalnie efekt wielu testów z zastosowaniem różnych frezów.

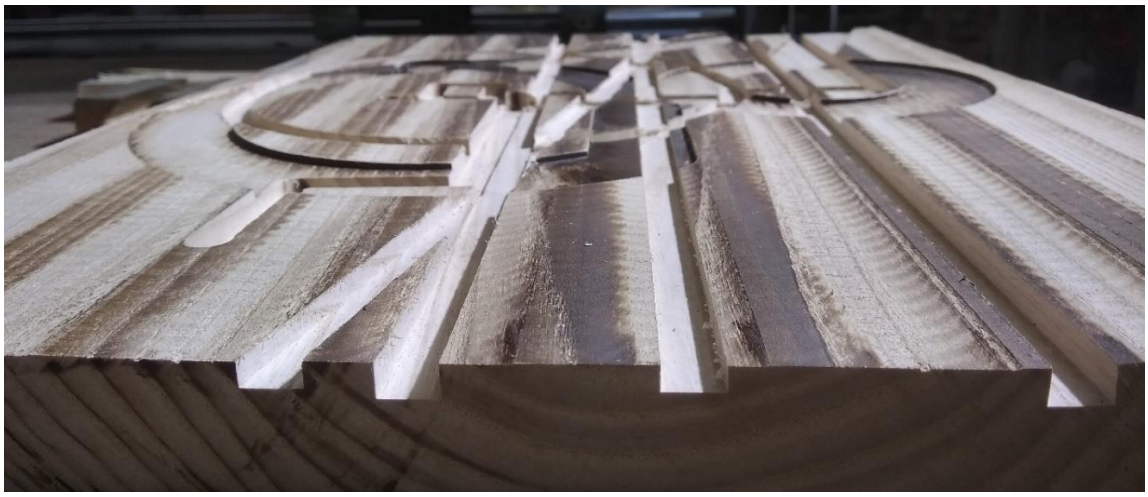
a) b)

Rys. 6.3. Pierwsze wyfrezowane kształty

a) b)

Rys. 6.4. Wykonywanie frezowania

a) efekt końcowy, b) końcówka robocza silnika frezarki z frezem



Rys. 6.5. Efekty pracy frezarki

W trakcie wykonywania kolejnych testów pojawiły się jeszcze drobne problemy, natomiast co do zasady nie było to nic poważnego i dało się to naprawić poprzez niewielkie zmiany w kodzie. Wyjątkiem pozostała komunikacja pomiędzy płytką STM32F3DISCOVERY, a komputerem PC dotycząca dokładnej lokalizacji. Mikrokontroler zlicza dokładną liczbę kroków i przelicza je na przemierzoną odległość poszczególnych osi. Także po właściwym ustawieniu granic obszaru roboczego, nie zostaną one przekroczone podczas wykonywania zadanej trajektorii. Natomiast program w komputerze PC nie posiada informacji o dokładnej lokalizacji frezu. Generowane ścieżki są obliczane przy założeniu, że frezarka znajduje się w położeniu początkowym. Tym samym powinniśmy przed rozpoczęciem realizacji trajektorii podjechać ręcznie do obszaru startowego lub zastosować współrzędne lokalne. Jest to pewna niedogodność, którą autor ma zamiar w przyszłości naprawić, natomiast nie przeszkadza to znacząco w korzystaniu z urządzenia.

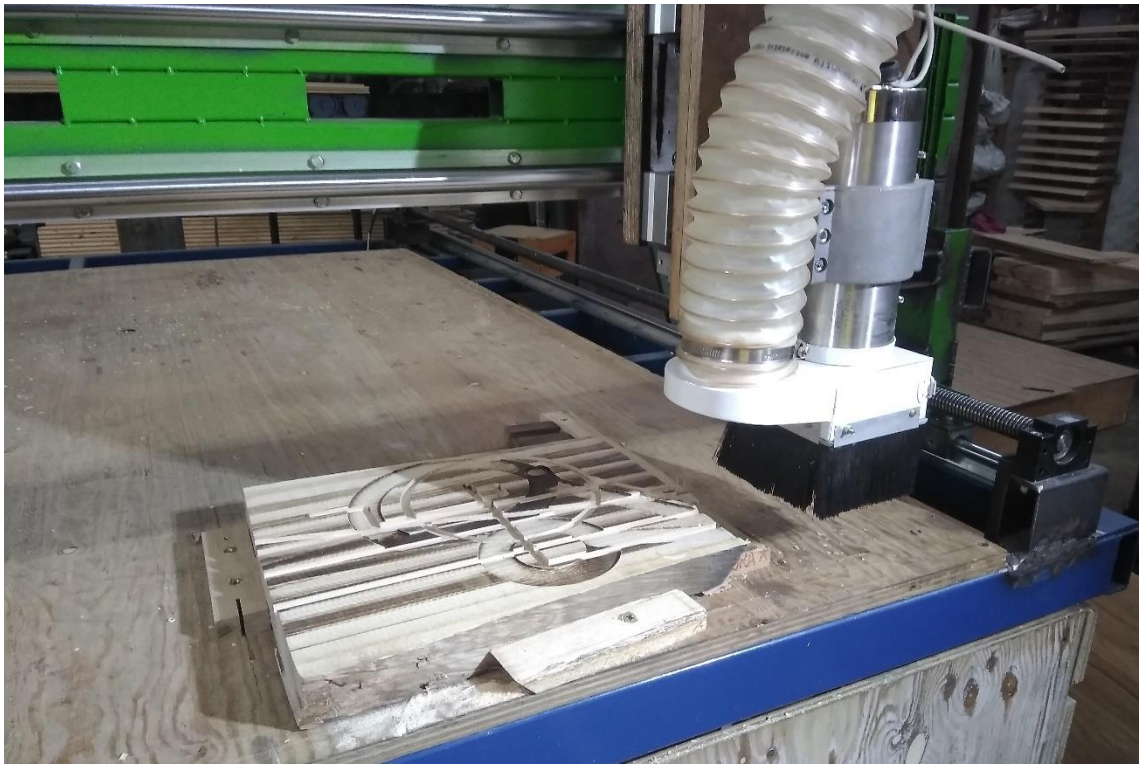
Podsumowując, ostateczne testy dały zdecydowanie pozytywne rezultaty. Frezarka spełnia swoje zadania i poprawnie realizuje zadane trajektorie. Frezowane kształty są dokładne w stopniu zupełnie wystarczającym.

7. PODSUMOWANIE I WNIOSKI

Realizacja projektu związanego nie tylko z oprogramowaniem, ale w dużej mierze ze sprzętem okazała się bardzo czasochłonnym zadaniem. Nie spodziewano się aż takiego nakładu pracy potrzebnej do finalizacji robota. Do tego było to związane z koniecznością wielokrotnych dojazdów do warsztatu, w którym znajdowała się konstrukcja. Niemniej jednak satysfakcja płynąca z ostatecznych efektów pracy była naprawdę wielka. Napisanie samego programu nie daje tyle zadowolenia co możliwość obserwacji frezarki, która wykonuje dokładnie to co jej zadaliśmy.

Obecny stan projektu to działająca baza, którą można rozszerzać o kolejne funkcjonalności. Podstawowa możliwość rozwoju leży w generacji większej ilości różnych kształtów, liter i cyfr. Autor widzi również przyszłościowe możliwości usprawnienia samego programu, aby działał szybciej.

Projekt frezarki opartej na robocie kartezjańskim został zrealizowany, przetestowany i zakończony sukcesem. Widoczna na rys. 7.1. maszyna ma duży potencjał i może być wykorzystywana praktycznie w zastosowaniach stolarskich. Niekoniecznie na dużą skalę, ale z pewnością do zastosowań w niewielkim warsztacie.



Rys. 7.1. Widok robota z obrabianym materiałem po zakończonej pracy

WYKAZ LITERATURY

1. Schmidt A.: Elementy wykonawcze automatyki – materiały wykładowe, Politechnika Gdańska, 2019.
2. Fiertek P.: Podstawy robotyki – materiały wykładowe część teoretyczna, Politechnika Gdańska, 2020.
3. Śruby trapezowe, http://cncland.pl/naped/sruby-trapezowe-i-nakretki/sruby-trapezowe/item/335-sruby-trapezowe-informacje-techniczne, 2021.
4. Bloki łożyskujące, https://www.ebmia.pl/wiedza/porady/mechanika-porady/blok-lozyskujacy/, 2021.
5. Instrukcja sterownika, *New step servo driver-DL86H manual*, 2021.
6. Instrukcja falownika, D32 Series Frequency Converter – User's Manual, 2021.
7. STM32F303VC, instrukcja użytkownika, STMicroelectrnics Corp., 2021.
8. Kowalski P., Smyk R.: *Sterowanie szeregowe maszyną CNC,* Poznań University of Technology, Academic Journal, Poznań 2016, No 85.
9. Licznik mikrosekund, https://controllerstech.com/create-1-microsecond-delay-stm32/, 2021.
10. Czaja Z.: Mikrosterowniki i mikrosystemy rozsproszone – materiały wykładowe, Politechnika Gdańska, 2020.
11. Sterownik silnika krokowego 5,5 Nm, https://adroautomatyka.pl/pl/p/Serwo-Silnik-krokowy-5.5Nm-sterownik-Servo-CNC/234, 2021.
12. Sterownik silnika krokowego 3 Nm, https://oferta.forum-cnc.pl/?181,silnik-serwo-krokowy-3nm-sterownik-2hss57-2m-kenkodera, 2021.

WYKAZ RYSUNKÓW

WYKAZ TABEL

Dodatek A: Opis dyplomu

Opis dyplomu

1. Tytuł dyplomu
   Układ sterowania frezarką zamontowaną na robocie kartezjańskim.

2. Cel i przeznaczenie
   Celem projektu był dobór elementów do konstrukcji mechanicznej robota kartezjańskiego z frezarką oraz zaprojektowanie i implementacja układu sterowania robotem. W ramach projektowanego systemu sterowania należało przewidzieć sterowanie ręczne z poziomu pilota oraz automatyczne odtwarzanie zadanej trajektorii.

3. Funkcjonalność:
   3.1. Opis realizowanych funkcji
   Projekt przede wszystkim realizuje zadanie frezowania zadanej ścieżki w drewnie. Możemy sterować ręcznie za pomocą przycisków w interfejsie użytkownika lub wykonywać przygotowaną wcześniej trajektorię. Program umożliwia własną generację prostych kształtów i ścieżek oraz importowanie ścieżek skomplikowanych z programów zewnętrznych. Istnieje również możliwość wizualizacji zadanej ścieżki w celu jej sprawdzenia przed wykonaniem.
   3.2. Lista przykładowych zastosowań:
   - wyrównywanie powierzchni nieregularnego materiału,
   - frezowanie materiału do zadanej grubości,
   - frezowanie prostych kształtów i konturów,
   - frezowanie skomplikowanych brył importowanych z programów zewnętrznych.

   3.3. Szczegółowe opisy działania projektu.
   Generalnie scenariusz działania projektu można rozpisać w kilku następujących po sobie punktach:
   - generacja ścieżki narzędzia roboczego lub jej zaimportowanie z programu zewnętrznego,
   - zamiana ścieżki narzędzia na sekwencję kolejnych kroków silników krokowych,
   - symulacja wygenerowanej ścieżki w celu sprawdzenia jej poprawności,
   - realizacja wygenerowanej trajektorii z możliwością jej zatrzymania oraz kontynuowania.

   Oprócz tego podstawowego scenariusza możemy poruszać frezarką za pomocą przycisków znajdujących się w interfejsie użytkownika.

4. Ogólna architektura systemu
   Generalnie system składa się z komputera PC, mikrokontrolera STM32F3DISCOVERY, silników krokowych, sterowników silników, przekładni śrubowej, falownika, silnika frezarki, pompy chłodziwa, układu przekaźników, przycisku bezpieczeństwa oraz mechanicznej konstrukcji robota kartezjańskiego.

5. Architektura sprzętu
   Część sterująca składa się z komputera PC oraz mikrokontrolera STM32F3DISCOVERY, połączonych dwoma łączami szeregowymi UART. Jednym do przekazywania danych, a drugim do dwukierunkowego przesyłania instrukcji sterujących. Wyjścia mikrokontrolera są połączone do przekaźników oraz sterowników silników krokowych. Układ poruszania robotem składa się z przekładni śrubowej, napędu w postaci silników krokowych ze sterownikami oraz zasilacza. Silniki posiadają wbudowane enkodery, dzięki którym sterowniki automatycznie kontrolują ewentualne gubienie kroków i wprowadzają korekty. Kolejna część robota to silnik frezarki napędzany przez falownik. Ten z kolei jest załączany przez odpowiedni przekaźnik. Również pompa chłodziwa, wymagana przy zastosowanym silniku, jest załączana przez przekaźnik. Ostatnim elementem jest bardzo ważny przycisk bezpieczeństwa, który odcina

zasilanie od silników oraz falownika, co skutkuje zatrzymaniem ruchu robota i zanikiem zasilania silnika frezarki.

6. Architektura oprogramowania
   Całe oprogramowanie składa się z części napisanej na mikrokontroler oraz kodu napisanego na komputer PC. Oprogramowanie mikrokontrolera zostało stworzone w języku C i obsługuje komunikację szeregową z komputerem oraz zadawanie impulsów na wejścia sterowników. Dzięki jego niskopoziomowemu charakterowi możemy bardzo precyzyjnie odmierzać czas miedzy kolejnymi krokami silników. Z kolei program na komputer PC został napisany w języku Python i obsługuje generowanie oraz importowanie ścieżek narzędzia, generowanie sekwencji impulsów podawanych na sterowniki silników, symulację zadanych trajektorii oraz ich wykonanie. Program posiada intuicyjny interfejs graficzny, który umożliwia realizację powyższych zadań w formie przystępnej i łatwej oraz komunikuje się z mikrokontrolerem za pomocą łącza szeregowego.

7. Opis metody/sposobu wytwarzania projektu
   Rozwijanie oprogramowania rozpoczęło się od napisania funkcji do zamiany punktów trajektorii w przestrzeni trójwymiarowej na sekwencje kroków silników, które dadzą pożądany efekt. Gdy ta część po pierwszych testach zaczęła funkcjonować nastąpiło rozwijanie klasy Generatora. Napisano blok kodu odpowiadający za generowanie ścieżek narzędzia w postaci kolejnych punktów, które po połączeniu dawały pożądaną ścieżkę. W kolejnym etapie rozwinięto interfejs użytkownika w postaci aplikacji okienkowej. Rozmieszczono oraz nadano nazwy przyciskom, etykietom oraz polom edycyjnym do wprowadzania danych. Następnie te elementy interfejsu połączono z odpowiednimi funkcjami i napisano ich obsługę. Generalnie w takiej kolejności powstawało oprogramowanie, chociaż na skutek testów oraz wprowadzanych na bieżąco zmian, dokonywano modyfikacji w różnych częściach kodu.

   Realizacja części mechanicznej rozpoczęła się od rozpoznania problemu, przeglądu możliwych rozwiązań oraz wypracowania koncepcji projektu. Następnie nastąpiła faza zakupu potrzebnych elementów oraz elementów potrzebnych do ich montażu i poprawnego działania. Dodatkowo trzeba było odpowiednio wytoczyć zakończenia śrub trapezowych wykorzystanych w przekładni śrubowej, co miało miejsce równolegle z testami układu elektrycznego. Następnie należało posiadane elementy zamontować i złożyć w spójną całość. Po poprowadzeniu przewodów elektrycznych oraz połączeniu części sprzętowej z oprogramowaniem można było przejść do finalnej fazy testów robota. Wykazały one pewne problemy, jednak dało się je szybko naprawić i doprowadzić projekt frezarki opartej na robocie kartezjańskim do sprawności operacyjnej.

Dodatek B: Instrukcja obsługi

Instrukcja obsługi

frezarki opartej na robocie kartezjańskim

1. **Uruchamianie**
   W celu załączenia obwodu zasilania należy najpierw przekręcić biały włącznik umieszczony na konstrukcji robota obok beczki z płynem chłodzącym. Następnie należy upewnić się, że przycisk bezpieczeństwa jest wyciśnięty. Jeżeli nie jest wyciśnięty, należy to wykonać. Następnie uruchamiamy komputer PC umieszczony na panelu operatorskim. Po włączeniu komputera należy odnaleźć na pulpicie aplikację o nazwie „Frezarka" oraz ją uruchomić. Teraz możemy korzystać z frezarki poprzez interfejs użytkownika.

2. **Wyłączanie**
   Przed wyłączeniem urządzenia należy upewnić się, że robot się nie porusza, silnik frezarki się nie obraca, a pompa chłodziwa jest wyłączona. Jeżeli powyższe warunki nie są spełnione, należy wcisnąć przycisk RESET. Jeżeli nie jest to możliwe, należy w pierwszej kolejności wcisnąć przycisk STOP, a następnie RESET. W celu wyłączenia całego robota należy najpierw zamknąć aplikację „Frezarka", następnie wyłączyć komputer. Po wykonaniu powyższych czynności należy wyłączyć obwód zasilania poprzez przekręcenie białego wyłącznika, umieszczonego na konstrukcji robota obok beczki z płynem chłodzącym.

3. **Awaryjne odcięcie zasilania**
   Do awaryjnego odcięcia zasilania służy czerwony przycisk bezpieczeństwa znajdujący się po prawej stronie panelu operatorskiego. Jego wciśnięcie powoduje przerwanie ruchu robota oraz wyłączenie zasilania silnika frezarki. Przy czym frez będzie się dalej obracał, wytracając prędkość, co może potrwać kilkanaście sekund.
   Aby włączyć robota po awaryjnym odcięciu zasilania należy wyłączyć i ponownie włączyć na komputerze aplikację „Frezarka". Następnie można wycisnąć przycisk bezpieczeństwa. W tym momencie możemy dalej normalnie korzystać z robota.

4. **Obsługa programu – opis przycisków**
   a. x+, x-, y+, y-, z+, z- - przyciski do ręcznego sterowania ruchem robota, poruszają odpowiednio osią x, y oraz z w kierunku dodatnim (+) lub ujemnym (-),
   b. Set intake - przycisk ustawia głębokość wżeru na wpisaną w polu tekstowym obok przycisku,
   c. Set z plane - przycisk ustawia współrzędną z płaszczyzny na której będą wykonywane wszelkie kształty z dopiskiem „2D" na wpisaną w polu tekstowym obok przycisku,
   d. Set drill diameter - przycisk ustawia średnicę frezu na wpisaną w polu tekstowym obok przycisku, jest to parametr potrzebny przy funkcji „flatten surface",
   e. generate new path - przycisk tworzy nową ścieżkę narzędzia zgodnie z wybraną opcją kształtu oraz wprowadzonymi parametrami,
   f. Add to existing path - przycisk dodaje do wcześniej wygenerowanej ścieżki kolejną ścieżkę narzędzia zgodnie z wybraną opcją kształtu oraz wprowadzonymi parametrami,
   g. Choose option - rozwijane menu służące do wyboru rodzaju generowanej ścieżki; po wybraniu jednej z opcji nazwa przycisku zmienia się na nazwę obecnie wybranej opcji,
   h. Generate ticks - wciśnięcie tego przycisku spowoduje wygenerowanie listy kroków silników pozwalających na realizację obecnie zadanej ścieżki narzędzia,

i. Simulate the path - wciśnięcie tego przycisku powoduje wyświetlenie wykresu trójwymiarowego z pożądaną oraz wygenerowaną ścieżką narzędzia,
j. START - przycisk włącza pompę chłodziwa, silnik frezarki, a następnie rozpoczyna realizację zadanej trajektorii,
k. STOP - przycisk służy do zatrzymania ruchu robota, nie wyłącza on silnika frezarki, ani pompy chłodziwa,
l. RESUME - przycisk służy do kontynuowania ruchu robota gdy został zatrzymany przyciskiem „stop”,
m. RESET - przycisk reset wyłącza silnik frezarki i pompę chłodziwa oraz ustawia wszelkie parametry związane z realizacją zadanej trajektorii na wartości początkowe.

5. Rodzaje kształtów

a. Line 2D generuje odcinek pomiędzy dwoma punktami zadanymi przez użytkownika. Trzecią współrzędną jest ustawiona w chwili generacji wartość parametru „z plane”.
b. Line 3D generuje odcinek pomiędzy dwoma zadanymi punktami w przestrzeni trójwymiarowej.
c. Circle 2D generuje okrąg o zadanych współrzędnych środka oraz promieniu. Trzecią współrzędną jest ustawiona w chwili generacji wartość parametru „z plane”.
d. Rectangular 2D generuje prostokąt o zadanych parametrach. Przy czym wystarczy podać dwa z trzech możliwych parametrów: współrzędne lewego dolnego rogu, współrzędne prawego górnego rogu, wysokość i szerokość. Trzecią współrzędną jest ustawiona w chwili generacji wartość parametru „z plane”.
e. Flatten surface – generuje trajektorię do wyrównywania powierzchni. Należy zadać prostokątny obszar na którym nastąpi wyrównywanie w sposób analogiczny jak dla Rectangular 2D. Ponadto należy zadać współrzędną z powierzchni poziomej na której nastąpi pierwsze frezowanie oraz współrzędną z powierzchni poziomej na której wyrównywanie ma się zakończyć. Krótko mówiąc generujemy prostopadłościan, który ma być cały wyfrezowany. Należy pamiętać o ustawieniu parametru wżeru za pomocą przycisku „Set intake”. Jest to głębokość na którą schodzi frez przy zbieraniu kolejnych warstw materiału.
f. Import file – jeżeli ścieżka narzędzia została wygenerowana w programie zewnętrznym możemy ją zaimportować poprzez tą opcję. Jeżeli plik ze ścieżką znajduje się w tej samej lokalizacji co skrypt, nie trzeba wprowadzać ścieżki dostępu. Jeżeli znajduje się w innej lokalizacji, konieczne jest podanie ścieżki dostępu.

6. Korzystanie z programu

a. Za pomocą przycisków do ręcznego poruszania możemy ustawiać końcówkę roboczą w dowolnym miejscu. Należy jednak pamiętać, aby z nich nie korzystać gdy wykonywana jest zadana trajektoria – gdy robot się porusza po wciśnięciu przycisku „START”. Należy pamiętać, że jeżeli po zatrzymaniu pracy programu będziemy ręcznie poruszać robotem, to kontynuacja wykonywania programu będzie realizowana od tego miejsca w którym końcówka robocza znajduje się w momencie wciśnięcia przycisku „RESUME”.
b. Parametry „intake”, „drill diameter” oraz „z plane” są brane pod uwagę tylko w momencie generacji ścieżki narzędzia. Można je ustawiać w dowolnym momencie pracy programu. Po wciśnięciu przycisku otrzymamy komunikat o poprawności lub błędzie ustawienia danego parametru oraz aktualną wartość parametru.
c. Generacja ścieżek:
    i. Z rozwijanej listy w lewym górnym rogu należy wybrać interesujący nas kształt, który chcemy wyfrezować.

ii. Następnie wprowadzamy potrzebne parametry w wyświetlonych okienkach, które są odpowiednio podpisane.
iii. Jeżeli generujemy nową ścieżkę, wciskamy przycisk „Generate new path”. Jeżeli chcemy dodać kolejny kształt do istniejącej ścieżki, wciskamy przycisk „Add to existing path”.

d. Wykonanie programu:
i. Jeżeli posiadamy już wygenerowaną ścieżkę, należy wcisnąć przycisk „Generate ticks”. W tym momencie ścieżka zadana w postaci kolejnych punktów w przestrzeni zostanie zamieniona na sekwencję kroków silników. Ta operacja może potrwać kilkanaście, a nawet kilkadziesiąt sekund.
ii. Po wygenerowaniu sekwencji kroków silników możemy sprawdzić jak będzie wyglądała trajektoria narzędzia roboczego poprzez wciśnięcie przycisku „Simulate the path”. W tym momencie wyświetli się trójwymiarowy wykres z trajektorią narzędzia oraz pożądaną ścieżką.
iii. Aby uruchomić realizację programu należy wcisnąć przycisk „START”. W tym momencie najpierw załączy się pompa chłodziwa. Program odczeka odpowiednią ilość czasu, a następnie włączy silnik frezarki. Dopiero po pełnym rozpędzeniu frezu nastąpi realizacja zadanej trajektorii.
iv. W każdym momencie możemy wcisnąć przycisk „STOP”, który spowoduje zatrzymanie ruchu robota. Należy pamiętać, że w tym momencie pompa chłodziwa oraz silnik frezarki pozostają włączone.
v. Przycisk „RESUME” powoduje kontynuację wykonywania ścieżki jeżeli wcześniej został wciśnięty przycisk „STOP”.
vi. Wciśnięcie przycisku „RESET” powoduje wyłączenie silnika frezarki. Po upływie zaprogramowanej ilości czasu nastąpi wyłączanie pompy chłodziwa. Wszystkie parametry zostaną zresetowane do wartości początkowych. W tym momencie wciśnięcie przycisku „START” spowodowałoby rozpoczęcie realizacji ścieżki od miejsca w którym znajduje się narzędzie robocze.

7. Ważne informacje

a. Wszystkie wielkości związane z wymiarami są wyświetlane oraz wprowadzane w milimetrach.
b. Wszelkie figury są generowane w taki sposób, że po ich konturze będzie się przemieszczała oś frezu.
c. Należy pamiętać, że przy sterowaniu ręcznym program nie zatrzyma się po dojechaniu do granicy obszaru roboczego. Tym samym należy wykazać szczególną czujność przy sterowaniu ręcznym i uważać, aby nie dojechać do końca obszaru roboczego.

8. Błędy sterowników

Jeżeli dojdzie do błędu sterownika, co można rozpoznać po blokadzie ruchu w danej osi oraz napisie „Err” na jego wyświetlaczu, należy zresetować cały układ poprzez wciśnięcie przycisku bezpieczeństwa. Włączanie programu po wciśnięciu przycisku bezpieczeństwa zostało opisane w punkcie 3.

Dodatek C: Instrukcja dla projektanta

Instrukcja dla projektanta

frezarki opartej na robocie kartezjańskim

1. Mikrokontroler STM32F3DISCOVERY

a. Oprogramowanie

Oprogramowanie mikrokontrolera zostało zrealizowane jako projekt w programie STM32CubeIDE. Można w nim zarówno kompilować kod, wgrywać na płytkę oraz debugować. Sam projekt generuje wiele plików, natomiast kod napisany do niniejszego projektu został umieszczony w plikach:

i. main.c – zawiera główną pętle programu i definicje części funkcji,
ii. receive_uart.c – zawiera obsługę odbioru poprzez łącze UART,
iii. send_uart.c – zawiera obsługę wysyłania poprzez łącze UART

oraz w odpowiadających im plikach nagłówkowych z rozszerzeniem „.h”. Opis kodu jest zamieszczony jako komentarze w samym kodzie oraz w pliku „Opis dyplomu”.

b. Wyprowadzenia

Poniżej zostaną rozpisane poszczególne połączenia wyprowadzeń mikrokontrolera.

i. PC0: PUL- sterownika numer 1
ii. PC1: DIR- sterownika numer 1
iii. PC2: PUL+ oraz DIR+ sterownika numer 1
iv. GND: ENA+ oraz ENA- sterownika numer 1
v. PA1: PUL- sterownika numer 2
vi. PA2: DIR- sterownika numer 2
vii. PA3: PUL+ oraz DIR+ sterownika numer 2
viii. GND: ENA+ oraz ENA- sterownika numer 2
ix. PB0: PUL- sterownika numer 3
x. PB1: DIR- sterownika numer 3
xi. PB2: PUL+ oraz DIR+ sterownika numer 3
xii. GND: ENA+ oraz ENA- sterownika numer 3
xiii. PB15: IN7 układu przekaźników – sterowanie falownikiem
xiv. PB14: IN6 układu przekaźników – sterowanie pompą chłodziwa
xv. 5V: Vcc części sterującej układu przekaźników
xvi. GND: GND łącza danych UART
xvii. PC11: TX łącza danych UART
xviii. PC10: RX łącza danych UART
xix. GND: GND łącza instrukcji sterujących UART
xx. PD2: TX łącza instrukcji sterujących UART
xxi. PC12: RX łącza instrukcji sterujących UART

2. Aplikacja na komputerze

Aplikacja na komputerze została napisana w języku Python. Składa się z czterech skryptów:

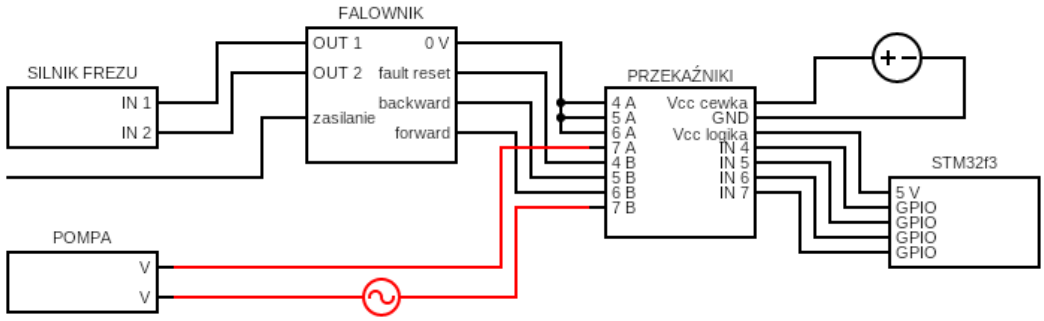
a. Frezarka.py – plik uruchomieniowy z klasą interfejsu użytkownika,
b. classes.py – zawiera pozostałe klasy programu,
c. functions.py – zawiera funkcje programu,
d. constants.py – zawiera wartości stałe z których korzystają wszystkie pozostałe skrypty.

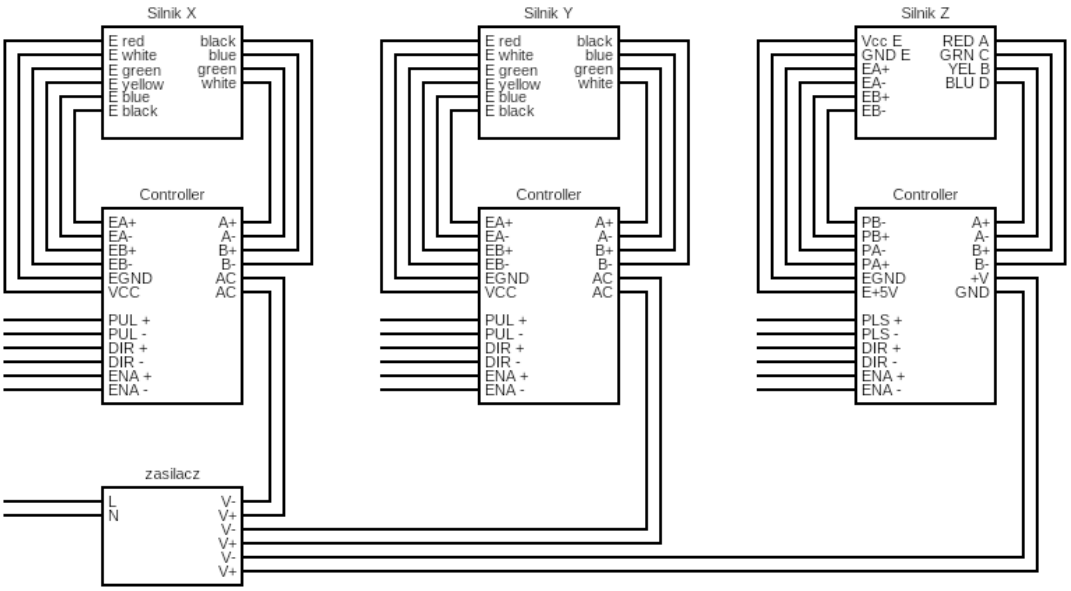
Skrypty są obkomentowane oraz opisane w pliku „Opis dyplomu”. Aby przystosować program do innych warunków pracy w standardowym zakresie, należy dokonywać zmian w pliku „constants.py”.

a. MAX_X, MIN_X, MAX_Y, MIN_Y, MAX_Z, MIN_Z: określają wielkość obszaru roboczego,
b. data_uart_com: zawiera nazwę wirtualnego portu do przesyłania danych, należy go odpowiednio ustawić przy zmianie komputera sterującego lub innym podłączeniu przewodów łącza szeregowego,
c. control_uart_com: jak wyżej, tylko dla portu do przesyłania instrukcji sterujących,
d. ACCURACY: tablica zawierająca odległość przemieszczenia liniowego w poszczególnych osiach dla jednego impulsu podanego na sterowniki silników numer 1 oraz 2 oraz dla dwóch impulsów podanych na sterownik silnika numer 3.
e. MOVE_BUTTONS: określa położenie przycisków do ręcznego poruszania w GUI,
f. POSITION_LABELS: określa położenie etykiet z pozycją narzędzia frezującego w GUI,
g. SET_BUTTONS: określa położenie przycisków oraz pól tekstowych do zmian parametrów intake, drill diameter oraz z plane,
h. MAIN_MENU: określa położenie części GUI do generacji ścieżki narzędzia,
i. START_STOP: określa położenie bloku do obsługi realizacji trajektorii.

3. Schematy elektryczne

Poniżej zostały przedstawione dwa schematy zawierające połączenia zrealizowane w projekcie.





4. Informacje dodatkowe

a. W przypadku konieczności zmian ustawień falownika należy zapoznać się z wersją papierową instrukcji obsługi, która znajduje się w dokumentacji robota w warsztacie.
b. Przekaźniki są załączane stanem niskim.

Dodatek D: Zawartość płyty CD

Na płycie CD znajdują się następujące pliki i foldery:

1. Oprogramowanie_na_PC
   Zawiera cztery pliki z rozszerzeniem języka Python. Program uruchamia się przez skrypt o nazwie „Frezarka.py”.
2. Oprogramowanie_na_mikrokontroler
   Zawiera projekt z programu STM32CubeIDE. Pliki zawierające pisany przez autora kod znajdują się w lokalizacji Oprogramowanie_na_mikrokontroler/Core/Src dla plików z rozszerzeniem .c oraz w lokalizacji Oprogramowanie_na_mikrokontroler/Core/Inc dla plików nagłówkowych .h.