Projekt

Sterowniki robotów

Dokumentacja

Manipulator Man

Skład grupy:
Filip Dyba, 235326
Mateusz Pietrzkowski, 232340

Termin: srTP15

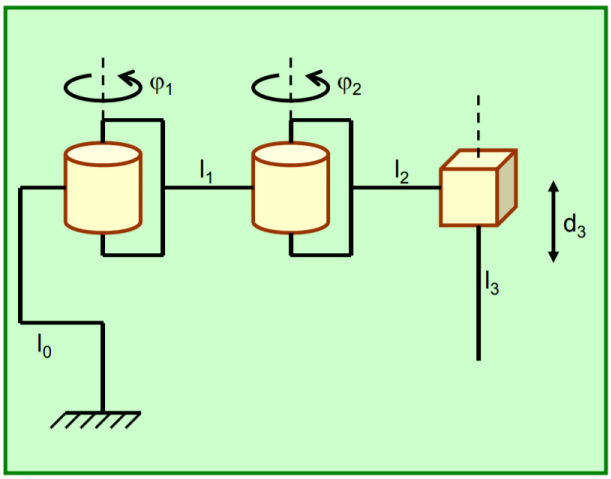
Prowadzący:
mgr inż. Wojciech Domski

14 czerwca 2019

Spis treści

1 Opis projektu

Projekt polegał na skonstruowaniu manipulatora typu SCARA o 3 stopniach swobody (RRT). Charakter struktury[1] manipulatora przedstawiono na rys. 1.



Rysunek 1: Srtuktura manipulatora typu SCARA

1.1 Założenia projektowe

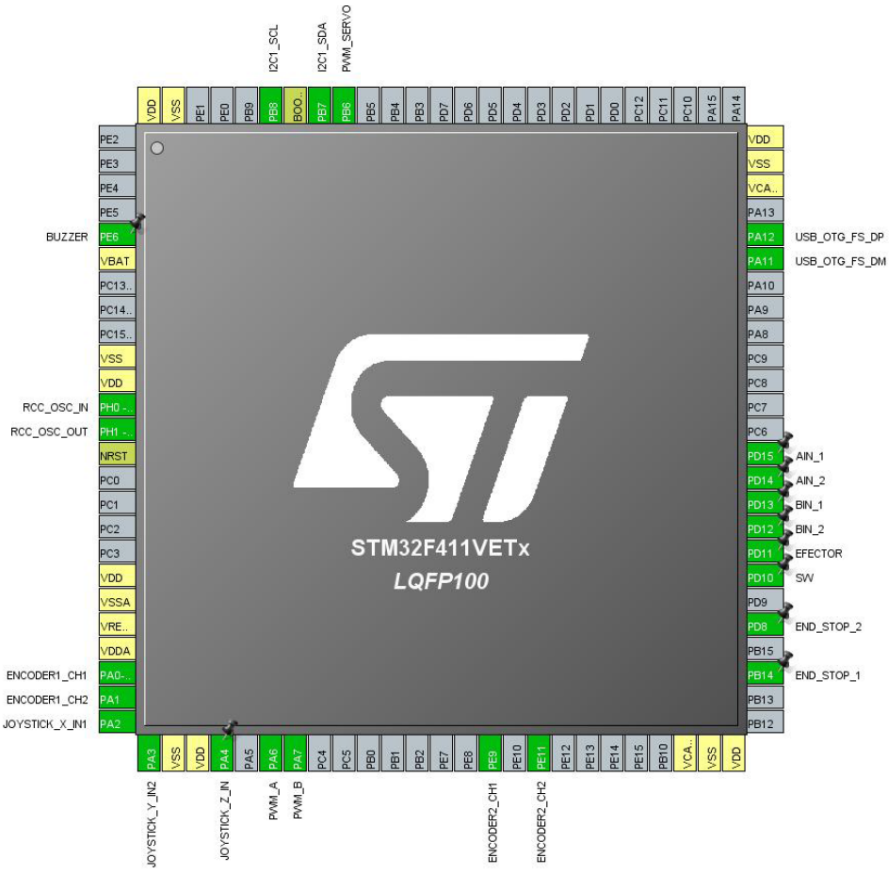
Projekt opiera się na następujących założeniach:

- Przestrzeń zadaniowa manipulatora jest fragmentem rury cylindrycznej (walca z otworem w środku).
- Szkielet manipulatora jest skonstuowany w oparciu o elementy zaprojektowane i wydrukowane za pomocą drukarki 3D.
- Elementy napędowe stanowią 2 silniki z przekładnią i enkoderem kwadraturowym (dla przegubów rotacyjnych) oraz 1 serwomechanizm (dla przegubu translacyjnego).
- Moduł sterowania robotem zrealizowany na płytce STM32F411VET6 Discovery (połączenia z urządzeniami zewnętrznymi za pomocą przygotownaego układu elektronicznego, minimalizacja okablowania).
- Sterowanie manipulatorem realizowane we współrzędnych wewnętrznych oraz zewnętrznych na 2 sposoby:
  1. Za pomocą joysticka podłączonego do mikrokontrolera (informacje dot. sterowania przedstawione na wyświetlaczu).
  2. Poprzez aplikację użytkownika na komputerze skomunikowanym z płytką za pomocą interfejsu USB.
- Zasilanie z zasilacza podane na prztwornice *step-down* lub zasilanie bateryjne - przygotowanie obu możliwości.

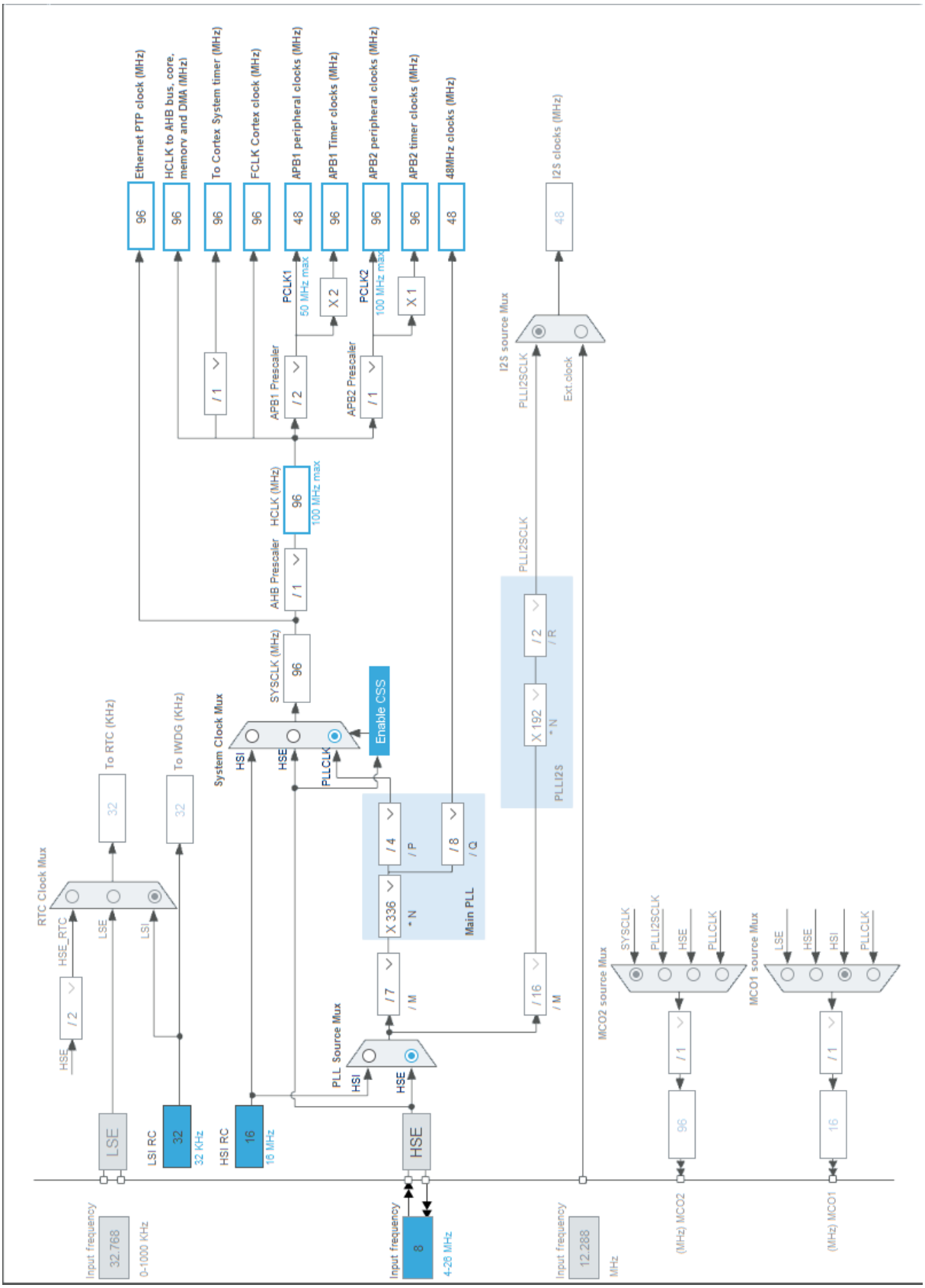
2 Konfiguracja mikrokontrolera

Wykonanie sterownika manipulatora oparto na płytce STM32F411VET6 Discovery [7]. Poniżej przedstawiono konfigurację mikrokontorolera, którą wykonano za pomocą programu STM32CubeMX. Konfiguracja portów mikrokontrolera została przedstawiona na rys. 2, natomiast konfigurację zegarów zobrazowano na rys. 3.

[1] Grafika zaczerpnięta z https://eia.pg.edu.pl/documents/184160/295670/wprowadzeniekinematyka.pdf



Rysunek 2: Konfiguracja wyjść mikrokontrolera w programie STM32CubeMX



Rysunek 3: Konfiguracja zegarów mikrokontrolera

2.1 Konfiguracja pinów

Numer pinu	PIN	Tryb pracy	Funkcja/etykieta
5	PE6	GPIO_Output	BUZZER
12	PH0 - OSC_IN	RCC_OSC_IN	
13	PH1 - OSC_OUT	RCC_OSC_OUT	
23	PA0-WKUP	TIM2_CH1	ENCODER1_CH1
24	PA1	TIM2_CH2	ENCODER1_CH2
25	PA2	ADC1_IN2	JOYSTICK_X_IN1
26	PA3	ADC1_IN3	JOYSTICK_Y_IN2
29	PA4	ADC1_IN4	JOYSTICK_Z_IN
31	PA6	TIM3_CH1	PWM_A
32	PA7	TIM3_CH2	PWM_B
40	PE9	TIM1_CH1	ENCODER2_CH1
42	PE11	TIM1_CH2	ENCODER2_CH2
53	PB14	GPIO_Input	END_STOP_1
55	PD8	GPIO_Input	END_STOP_2
57	PD10	GPIO_Input	SW
58	PD11	GPIO_Output	EFECTOR
59	PD12	GPIO_Output	BIN_2
60	PD13	GPIO_Output	BIN_1
61	PD14	GPIO_Output	AIN_2
62	PD15	GPIO_Output	AIN_1
70	PA11	USB_OTG_FS_DM	
71	PA12	USB_OTG_FS_DP	
92	PB6	TIM4_CH1	PWM_SERVO
93	PB7	I2C1_SDA	
95	PB8	I2C1_SCL	

Tabela 1: Konfiguracja pinów mikrokontrolera

Piny PD15 oraz PD14 to wyjścia, które odpowiadają za sterowanie kierunkiem obrotów silnika w jednym z przegubów rotacyjnych. Podobnie piny PD13 oraz PD12, które odpowiadją za silnik w drugim przegubie. Pin PD10 jest wykorzystany do obsługi przycisku, za pomocą którego użytkownik może poruszać się w menu. Pin PE6 pozwala na obsługę sygnału dźwiękowego. Pin PD11 stanowi natomiast wyjście, które może zostać wykorzystane dla funkcjonalności efektora i stanowi opcję rozwojową dla trzeciego przegubu. Natomiast piny PB14 oraz PD8 są skonfigurowane i mogą posłużyć do obsługi krańcówek. Sygnał na tych wejściach generowałby zewnętrzne przerwanie celem sprzętowego zabezpieczenia przed nieporządanym ruchem manipulatora.

Poniżej przestawiono opis konfiguracji wykorzystanych peryferiów.

2.2 ADC1

Peryferium wykorzystane do obsługi joysticka. Konfiguracja została przedstawiona w tabeli 2.

Parametr	Wartość
Clock Prescaler	PCLK2 divided by 8
Resolution	12 bits (15 ADC Clock cycles)
Data Alignment	Right alignment
Scan Conversion Mode	Enabled
Continuous Conversion Mode	Enabled
Discontinuous Conversion Mode	Disabled
DMA Continuous Requests	Enabled
End Of Conversion Selection	EOC flag at the end of single channel conversion
Number Of Conversion	3
External Trigger Conversion Source	Regular Conversion launched by software
External Trigger Conversion Edge	None
Rank	1
Channel	Channel 2
Sampling Time	480 Cycles
Rank	2
Channel	Channel 3
Sampling Time	480 Cycles
Rank	3
Channel	Channel 4
Sampling Time	480 Cycles

Tabela 2: Konfiguracja peryferium ADC1

Do wejść peryferium podłączono joysticki, które służą do ręcznego sterowania manipulatorem oraz poruszanie się po menu wyboru sterowania. W zależności od położenia joysticka wartość wejścia analogowego jest konwertowana na liczbę całkowitą z zakresu $(0; 4095)$.

2.2.1 DMA

Aby umożliwić wpisywanie odczytywanej wartości bezpośrednio do zmiennej z ADC1 powiązano DMA, którego konfigurację przedstawiono w tabeli 3.

Parametr	Wartość
Mode	Circular
Use fifo	Disable
Peripheral Increment	Disable
Memory Increment	Enable
Peripheral Data Width	Half Word
Memory Data Width	Half Word

Tabela 3: Konfiguracja DMA dla ADC1

2.3 I2C1

Standard transmisji wykorzystywany do komunikacji z wyświetlaczem LCD. Na wyświetlaczu prezentowane są opcje wyboru menu, a także, w przypadku trybu sterowania, aktualne wartości współrzędnych wewnętrznych i zewnętrznych manipulatora. W tabeli 4 przedstawiono konfigurację peryferium.

Prametr	Wartość
I2C Speed Mode	Fast Mode
I2C Clock Speed (Hz)	400000
Fast Mode Duty Cycle	Duty cycle Tlow/Thigh = 2
Clock No Stretch Mode	Disabled
Primary Address Length selection	7-bit
Dual Address Acknowledged	Disabled
Primary slave address	0
General Call address detection	Disabled

Tabela 4: Konfiguracja peryferium I2C1

2.4 RCC

Peryferium zostanie wykorzystane do zapewnienia taktującego sygnału zegarowego. Wykorzystuje się źródło HSE: Crystal/Ceramic Resonator. Konfigurację parametrów dla tego peryferium przedstawiono w tablei 5.

Prametr	Wartość
VDD voltage (V)	3.3
Instruction Cache	Enabled
Prefetch Buffer	Enabled
Data Cache	Enabled
Flash Latency(WS)	3 WS (4 CPU cycle)
HSI Calibration Value	16
TIM Prescaler Selection	Disabled
HSE Startup Timout Value (ms)	100
LSE Startup Timout Value (ms)	5000
Power Regulatror Voltage Scale	Power Regulator Voltage Scale 1

Tabela 5: Konfiguracja RCC

2.5 TIM1

Timer zostanie wykorzystany do odczytu wskazań z enkodera zamontownego przy silniku w drugim przegubie rotacyjnym. Konfigurację peryferium przedstawiono w tabeli 6.

Parametr	Wartość
Prescaler (PSC - 16 bits value)	0
Counter Mode	Up
Counter Period (AutoReload Register - 16 bits value)	1363
Internal Clock Division (CKD)	No Division
Repetition Counter (RCR - 8 bits value)	0
auto-reload preload	Disable
Master/Slave Mode (MSM bit)	Disable (Trigger input effect not delayed)
Trigger Event Selection TRGO	Reset (UG bit from TIMx_EGR)
Encoder Mode	Mode TI1 and TI2
Parametry dla Channel 1	
Polarity	Rising Edge
IC Selection	Direct
Prescaler Division Ratio	No division
Input Filter	15
Parametry dla Channel 2	
Polarity	Rising Edge
IC Selection	Direct
Prescaler Division Ratio	No division
Input Filter	15

Tabela 6: Konfiguracja TIM1

2.6 TIM2

Wykorzystywany do odczytu wskazań enkodera w pierwszym przegubie rotacyjnym. Wartości parametrów przedstawione w tabeli 7.

Parametr	Wartość
Prescaler (PSC - 16 bits value)	0
Counter Mode	Up
Counter Period (AutoReload Register - 16 bits value)	1363
Internal Clock Division (CKD)	No Division
auto-reload preload	Disable
Master/Slave Mode (MSM bit)	Disable (Trigger input effect not delayed)
Trigger Event Selection	Reset (UG bit from TIMx_EGR)
Encoder Mode	Encoder Mode TI1 and TI2
Parametry dla Channel 1	
Polarity	Rising Edge
IC Selection	Direct
Prescaler Division Ratio	No division
Input Filter	15
Parametry dla Channel 2	
Polarity	Rising Edge
IC Selection	Direct
Prescaler Division Ratio	No division
Input Filter	15

Tabela 7: Konfiguracja TIM2

2.7 TIM3

Timer wykorzystywany do generowania sygnałów PWM, które sterują prędkościami silników w przegubach rotacyjnych. Wartość prędkości zależy od wypełnienia sygnału PWM. Konfigurację peryferium przedstawiono w tabeli 8.

Parametr	Wartość
Prescaler (PSC - 16 bits value)	1
Counter Mode	Up
Counter Period (AutoReload Register - 16 bits value)	254
Internal Clock Division (CKD)	No Division
auto-reload preload	Disable
Master/Slave Mode (MSM bit)	Disable (Trigger input effect not delayed)
Trigger Event Selection	Reset (UG bit from TIMx_EGR)
PWM Generation Channel 1	
Mode	PWM mode 1
Pulse (16 bits value)	0
Fast Mode	Disable
CH Polarity	High
PWM Generation Channel 2	
Mode	PWM mode 1
Pulse (16 bits value)	0
Fast Mode	Disable
CH Polarity	High

Tabela 8: Konfiguracja TIM3

2.8 TIM4

Timer wykorzystywany do generowania sygnału PWM, za pomocą którego sterowany jest serwomechanizm w przegubie translacyjnym. Jego użycie jest uzasadnione odmiennością parametrów w stosunku do timerów wykorzystanych do sterowania sygnałem PWM silnikami w przegubach rotacyjnych. Częstotliwość timera skonfigurowano na $50Hz$, zgodnie z wymogami sterowania zastosowanego serwomechanizmu Tower Pro SG90 [5]. Wypełnienie sygnału powinno mieścić się w przedziale $[5\%ARR; 10\%ARR]$, gdzie ARR - AutoReload Register. Konfigurację peryferium przedstawiono w tabeli 9.

Parametr	Wartość
Prescaler (PSC - 16 bits value)	1919
Counter Mode	Up
Counter Period (AutoReload Register - 16 bits value)	999
Internal Clock Division (CKD)	No Division
auto-reload preload	Disable
Master/Slave Mode (MSM bit)	Disable (Trigger input effect not delayed)
Trigger Event Selection	Reset (UG bit from TIMx_EGR)
Mode	PWM mode 1
Pulse (16 bits value)	50
Fast Mode	Disable
CH Polarity	High

Tabela 9: Konfiguracja TIM4

2.9 USB_OTG_FS

Peryferium wykorzystywane do komunikacji z komputerem, gdzie z poziomu terminala użytkownik może zadać żądane współrzędne wewnętrzne bądź zewnętrzne. Dane muszą być zadawane w formie ustalonej ramki. Peryferium pracuje w trybie *Device_ Only*. Jego konfiguracja została przedstawiona w tabeli 10.

Parametr	Wartość
Speed	Device Full Speed 12MBit/s
Low power	Disabled
Link Power Management	Disabled
VBUS sensing	Disabled
Signal start of frame	Disabled

Tabela 10: Konfiguracja peryferium USB_OTG_FS

Dzięki takiej konfiguracji urządzenia mikrokontroler jest widziany przez komputer jako wirtualny port COM.

3 Urządzenia zewnętrzne

Wykorzystywane urządzenia zewnętrzne służą przede wszystkim do zapewnienia ruchów w przegubach manipulatora. Są to: serwomechanizm dla przegubu translacyjnego oraz silniki DC dla przegubów rotacyjnych, sterowane z wykorzystaniem dedykowanego sterownika. Ponadto wykorzystano wyświetlacz LCD, który umożliwia użytkownikowi korzystanie z prostego menu wyboru sterowania.

3.1 Silnik DC - DFR-09531

Manipulator jest zbudowany w oparciu o 2 silniki DC - DFR-09531 [1].

Silnik z przekładnią 34:1. Bez obciążenia porusza się z prędkością $210\frac{obr}{min}$ przy poborze prądu $0,13A$. Moment obrotowy $10,0kg \cdot cm(0,9Nm)$. Urządzenie posiada enkoder kwadraturowy o rozdzielczości 11 impulsów na obrót (po przełożeniu 374 impulsów na obrót). Silnik zasilany jest napięciem $6V$. Silnik jest wyposażony w enkoder inkrementalny, za pomocą którego można określać aktualną pozycję zadaną w danym przegubie.

3.2 Sterownik silników DC - TB6612FNG

TB6612FNG [9] to sterownik silników DC. Sterownik posiada mostek H oraz podwójne wyjścia pozwalające obsługiwać dwa silniki. Wyjście sterownika może być obciążone nawet do $1,2A$, prąd szczytowy w zależności od konfiguracji $2A-3,2A$. PWM wspiera częstotliwości do $100kHz$, posiada również wyłącznik termiczny zapobiegający spaleniu się układu.

3.3 Serwomechanizm - TowerPro SG-90 - micro

Serwomechanizm TowerPro SG-90 [5] wykorzystany do poruszania przegubem translacyjnym. Wykonuje obrót o 60° w $0,12s$. Sterowany sygnałem PWM o częstotliwości $50Hz$. Charakteryzuje się niewielką masą wynoszącą $9g$.

3.4 Wyświetlacz LCD

Wyświetlacz LCD HD44780 został wykorzystany do prezentacji opcji wyboru w menu, a także danych dotyczących aktualnej pozycji manipulatora. Wyświetlacz zasilany napięciem $5V$. Posiada 2 wiersze, z czego w każdym może znaleźć się maksymalnie 16 znaków. Wykorzystanie konwertera I2C umożliwia bezpośrednie zadawanie wiadomości za pomocą standardu I2C. Obsługę wyświetlacza zapewniono, wykorzystując przygotowaną bibliotekę [10] upublicznioną za pomocą systemu *github.com*.

3.5 Joystick analogowy

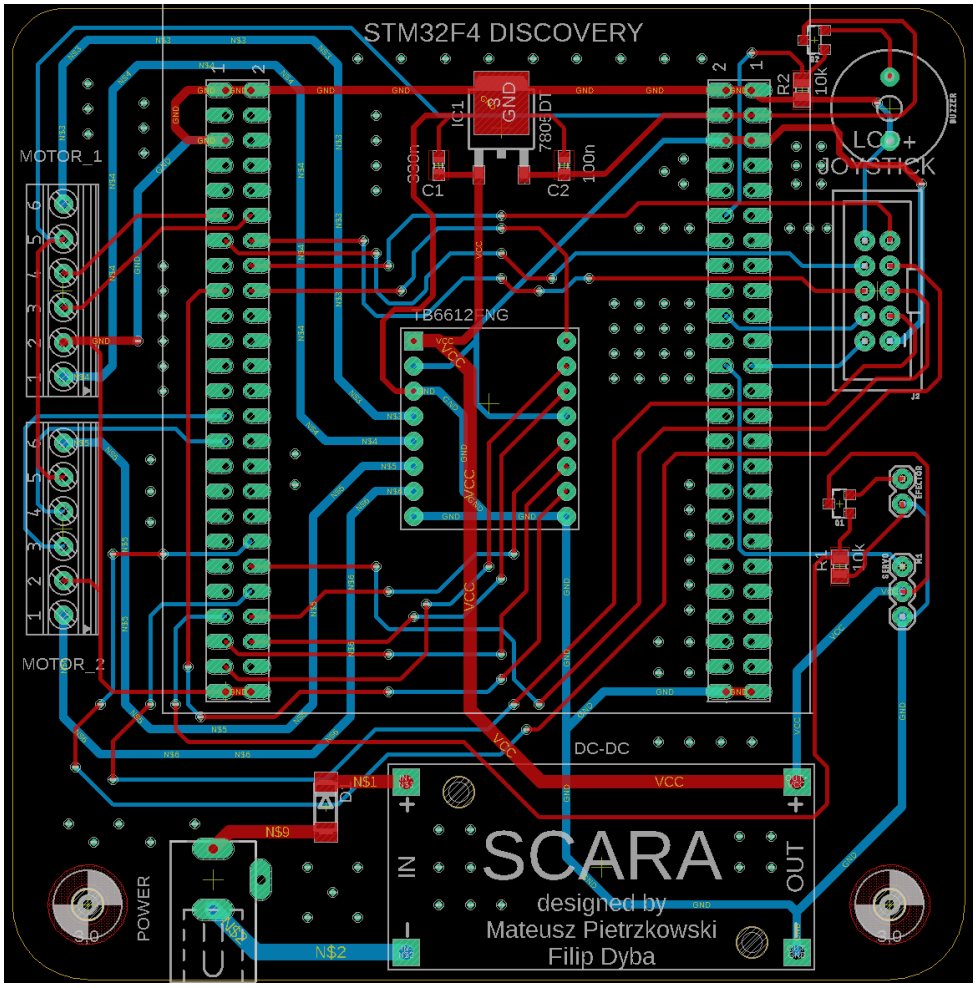
Do sterowania ręcznego oraz do poruszania się po menu wykorzystano joysticki analogowe. Wykorzystano 2 joysticki - pierwszy z nich zapewniał ruch w przegubach q_1 i q_2, a także ruch efektora w osi OX oraz OY w przypadku sterowania we współrzędnych zewnętrznych. Z drugiego joysticka wykorzystano jedynie możliwość ruchu w jednym kierunku - odpowiadał ruchowi przegubu translacyjnego wzdłuż osi OZ. Ten sam joystick dawał możliwość poruszania się po menu w trakcie wyboru trybu sterowania. Zatwierdzenie wyboru polega na przyciśnięciu przycisku stowarzyszonego z joystickiem. Joysticki są zasilane napięciem $3V$, a ich wyprowadzenia zostały podłączone do wejść analogowych mikrokontrolera.

4 Projekt elektroniki

Zgodnie z projektem elektroniki wykorzystano następujące elementy:

- STM32F411 Discovery
- Moduł sterownika silników DC TB6612FNG (podwójny mostek H)
- Enkodery inkrementalne zamontowane na wale silnika, 11 $\frac{im}{obr.}$
- Przetwornica impulsowa DC-DC step-down
- Stabilizator 5V, LM7805 w obudowie smd

Część elektroniczna dla projektu robota SCARA została wykonana w formie *shield'u* na płytkę Discovery z powodu względnej łatwości wykonania oraz stosunkowo niskich kosztów. Schemat elektroniczny wykonanej nakładki przedstawiono na rys. 5, natomiast projekt płytki - na rys. 4. Zaprojektowana płytka PCB zawiera w większości wyprowadzenia dla poszczególnych peryferiów, a także blok zasilania i filtracji napięcia.

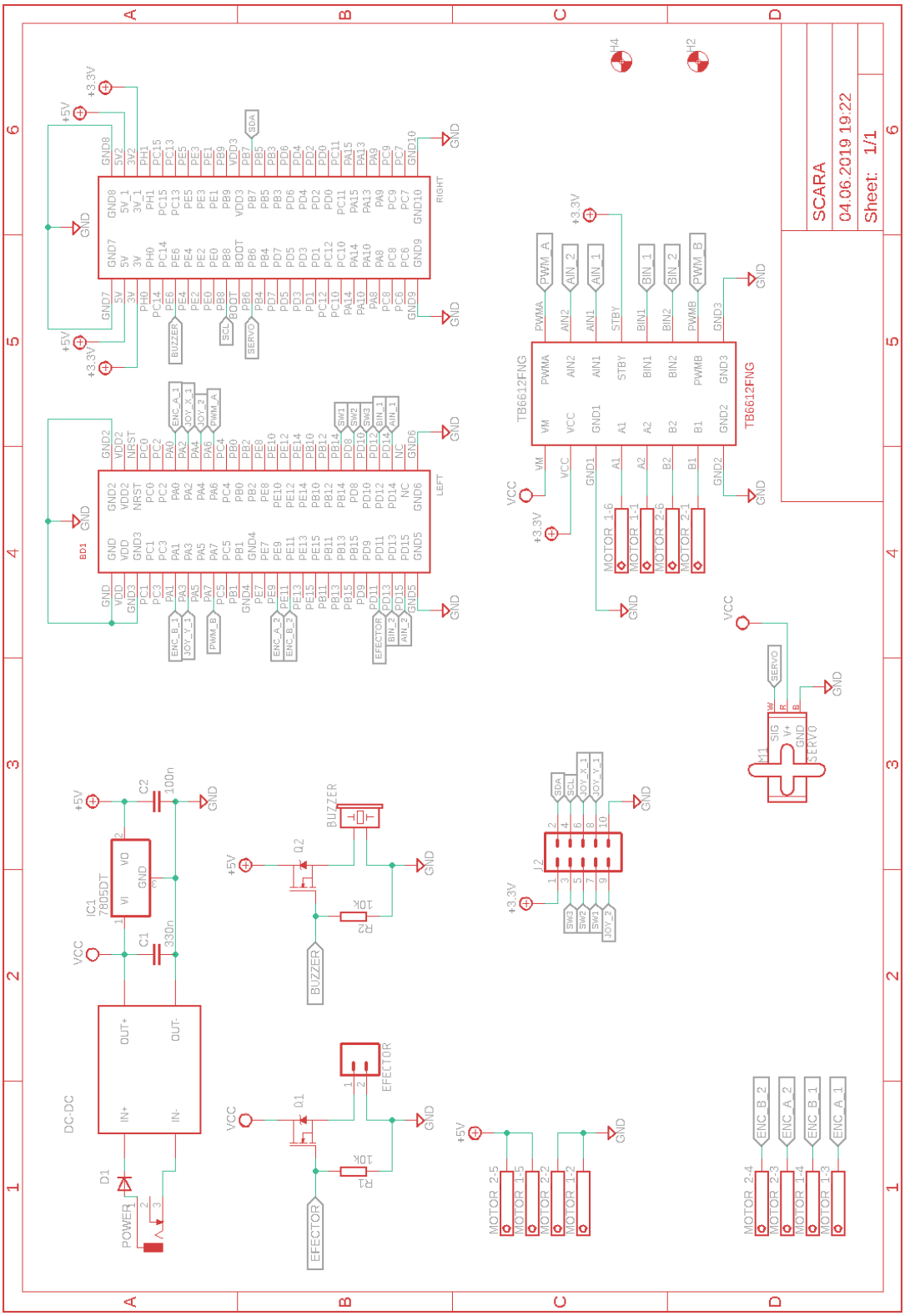


Rysunek 4: Projekt płytki

Do zasilenia układu należy używać zasilacza $9 - 25V$, z którego napięcie zostanie podane na:

- przetwornicę DC-DC step-down
- stabilizator LM7805

Napięcie 5V ze stabilizatora jest wykorzystane do zasilenia płytki STM32 Discovery [7] oraz enkoderów. Sterownik silnika jest zasilany napięciem 3.3V podanym bezpośrednio z płytki Discovery. Napięcie



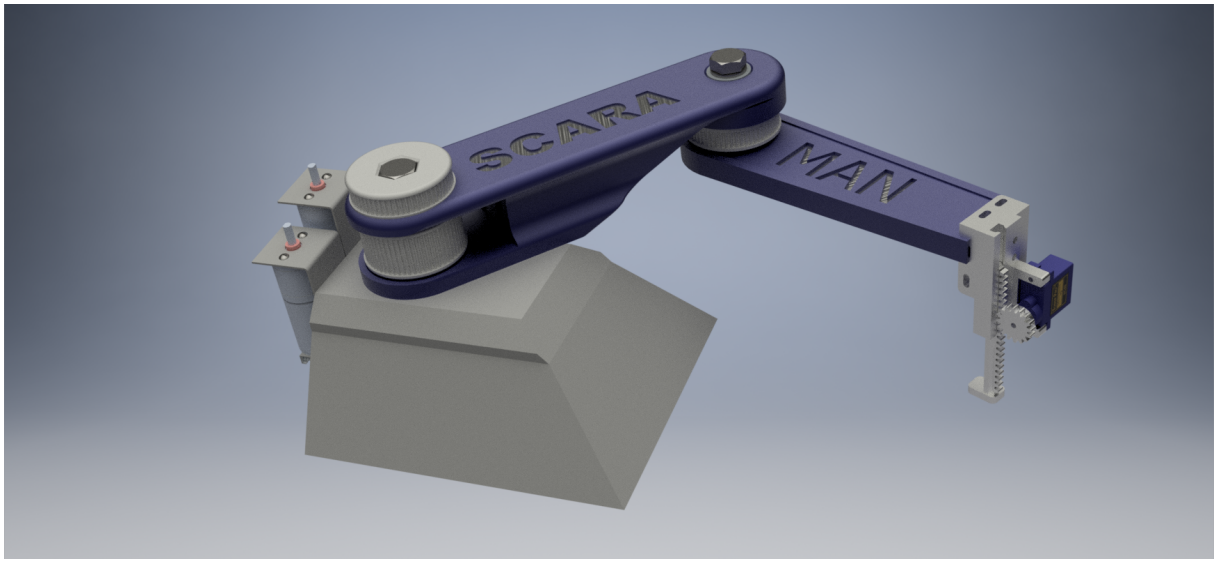
Rysunek 5: Schemat elektroniczny wykonanej płytki

wyjściowe na przetwornicy zostało ustalone na około 6V w celu zasilenia silników oraz serwa modelarskiego. Sterowanie silnikami odbywa się przy pomocy sterownika silników DC TB6612FNG [9], który został połączony bezpośrednio z portami GPIO. Enkodery inkrementalne, w które wyposażone są silniki, zostały podłączone do wejść timerów skonfigurowanych w trybie Encoder Mode. Do ustalenia kierunku obrotów silników są wykorzystywane 4 wyjścia GPIO, po 2 na każdy silnik, a do ustalenia prędkości -

2 wyjścia timera skonfigurowanego w trybie PWM. Serwomechanizm również jest sterowany z wyjścia timera, skonfigurowanego w trybie PWM. Odczyt wartości z joysticków analogowych został zrealizowany przy pomocy 3 wejść ADC, po jednym dla każdej osi. W celu prezentowania wyników obliczeń kinematyki odwrotnej oraz poruszania się po menu sterownika został wykorzystany wyświetlacz LCD wraz z konwerterem I2C, dzięki któremu w prosty sposób za pomocą wyjść SDA oraz SCL można wyświetlać wszystkie niezbędne informacje. PCB został wykonany w firmie JLCPCB, dzięki czemu zaistniała możliwość względnie swobodnego projektowania dwustronnego projektu oraz możliwość prototypowania urządzenia na pozostałych dostarczonych płytkach.

5 Konstrukcja mechaniczna

Wykonano projekt konstrukcji mechanicznej, której wyrenderowany model przedstwiono na rys. 6. Część mechaniczna manipulatora SCARA MAN składa się z dwóch podstawowych członów rotacyjnych napędzanych poprzez przekładnie pasową zębatą oraz jednego członu translacyjnego napędzanego poprzez przekładnię zębatą (koło zębate – zębatka). Konstrukcja robota została w pełni zaprojektowana od podstaw w środowisku Autodesk Inventor oraz wykonana w większości z elementów drukowanych w technologii FDM z materiału PET-G. Elementy konstrukcyjne zostały wykonane z profili aluminiowych, a uchwyty silników z blachy nierdzewnej o grubości 1mm. Dzięki takiemu zabiegowi udało się stworzyć jednocześnie lekki, wytrzymały oraz sztywny szkielet konstrukcji, który możemy dostosować do wymagań. Człony rotacyjne napędzane są niezależnie poprzez przekładnie pasową zębatą o profilu zęba T5 za pomocą silników DC. Pierwszy człon napędzany jest bezpośrednio z wału przekładni silnika, na którym zamocowane jest koło pasowe zębate o liczbie zębów 10. Moment obrotowy przekazywany jest na kolejne koło pasowe zębate o liczbie zębów 30 zamocowane na stałe na tym członie w osi jego obrotu. Drugi człon napędzany jest w podobny sposób z tą różnicą, że po drodze znajduje się koło pasowe zębate wolne łożyskowane, które odpowiada za pośredniczenie w przekazaniu napędu na drugi człon. Zabieg ten jest konieczny, aby uniemożliwić ruch drugiego członu w trakcie ruchu pierwszego z nich. Wszystkie elementy obrotowe zostały zaprojektowane w taki sposób, aby były łożyskowane. Jedyną wstępnie zauważoną wadą konstrukcji jest przenoszenie sił wzdłużnych przez łożyska poprzeczne, jednak we wstępnej fazie prototypowania pomijamy to spostrzeżenie z uwagi na masę ramienia, a co za tym idzie zdecydowanie niską wartość sił wzdłużnych, które nie powinny znacząco wpływać na działanie układu. Ostatni z przegubów opiera swoją zasadę działania na przeniesieniu momentu obrotowego z serwomechanizmu na ruch translacyjny zębatki.



Rysunek 6: Model konstrukcji mechanicznej

5.1 Elementy napędowe

Za ruchy obrotowe odpowiadają 2 silniki zasilane prądem stałym o napięciu nominalnym 6V i prędkością nominalną 210 obr/min z zamocowaną przekładnią o przełożeniu 34:1 na wale, na których znajdują się koła pasowe zębate o liczbie zębów 10 i profilu T5. Zastosowany profil zęba został wybrany z powodu łatwości wykonania takiego elementu w technologii druku 3D, jednak jego główną wadą jest luz międzyzębny wynoszący około 0,3mm. Napęd przekazany jest na koła pasowe zębate o liczbie zębów 30, dzięki czemu uzyskujemy dodatkowe przełożenie 3:1, co pozwala na zwiększenie momentu obrotowego, a także zwiększenie dokładności pozycjonowania kosztem mniejszej prędkości. Za ruch translacyjny odpowiada serwo modelarskie SG90, które zostało wybrane z powodu stosunkowo niskiej wagi, aby dodatkowo nie obciążać końca manipulatora.

5.2 Przestrzeń robocza

Przestrzenią roboczą manipluatora jest fragment rury cylindrycznej. Ze względu na ograniczenia konstrukcyjne manipulator może poruszać się w zakresie $(-90°; 90°)$ w przegubach rotacyjnych. Przegub translacyjny może wysunąć się maksymalnie $50mm$ w dół, licząc od pozycji zerowej.

Odległości między kolejnymi przegubami wyznaczono jako:

- $L1 = 170mm$ – odległość między pierwszym a drugim przegubem (długość pierwszego ramienia)
- $L2 = 150mm$ – odległość między drugim a trzecim przegubem (długość drugiego ramienia)

Tak zdefiniowane ograniczenia uwzględniono w implemnetacji sterowania i na ich podstawie stworzono odpowiednie funkcje ograniczjące ruch manipulatora w odpowiednich przegubach.

6 Kinematyka

Podstawą działania sterownika jest obliczona kinematyka prosta, która przekształaca zmienne wewnętrzne manipulatora w zewnętrzne, a także kinematyka odwrotna realizująca zadanie przeciwne. Obliczenia wykonano w oparciu o [8] oraz [6].

6.1 Kinematyka prosta

Dla obliczenia kinemtyki prostej wykorzystano algorytm Denavita - Hartenberga. Parametry algorytmu przedstawiono w tabeli 11. Znaczenie przyjętych oznaczeń jest następujące: q_1, q_2, q_3 - zmienne

Lp.	θ_i	d_i	a_i	α_i
1.	q_1	0	l_1	0
2.	q_2	0	l_2	π
3.	0	q_3	0	0

Tabela 11: Parametry algorytmu Denavita - Hartenberga

przegubowe; l_1, l_2 - odległości między kolejnymi przegubami. W projektowanym manipulatorze te wartości wynoszę odpowiednio:

$$l_1 = 170mm$$

$$l_2 = 150mm$$

Macierze przekształceń między układami współrzędnych umiejscowionymi w kolejnych przegubach prezentują się następująco:

$$A_0^1 = \begin{bmatrix} c_1 & -s_1 & 0 & l_1c_1 \\ s_1 & c_1 & 0 & l_1s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

$$A_1^2 = \begin{bmatrix} c_2 & s_2 & 0 & l_2c_2 \\ s_2 & -c_2 & 0 & l_2s_2 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

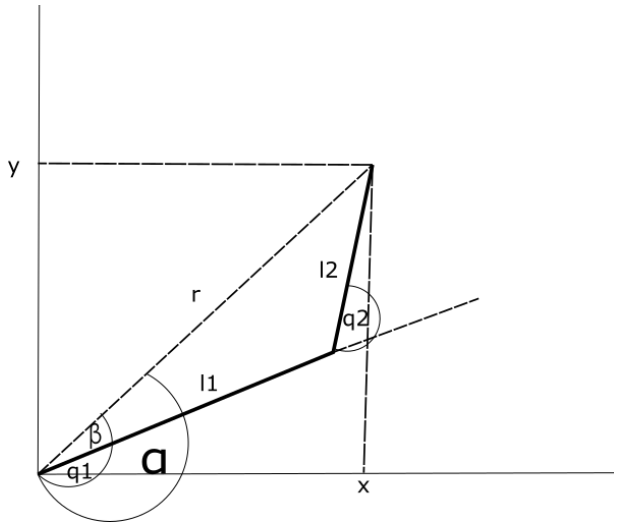
$$A_2^3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & q_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Na podstawie przekształceń 1, 2 oraz 3 obliczono kinematykę prostą:

$$A_0^3 = A_0^1 A_1^2 A_2^3 = \begin{bmatrix} c_{12} & s_{12} & 0 & l_1 c_1 + l_2 c_{12} \\ s_{12} & -c_{12} & 0 & l_1 s_1 + l_2 s_{12} \\ 0 & 0 & -1 & -q_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

6.2 Kinematyka odwrotna

Obliczając kinematykę odwrotną manipulatora, wykorzystano fakt, iż przegub translacyjny porusza się jedynie wzdłuż osi $0Z$, a pozostałe 2 przeguby można uprościć do przypadku dwuwahadła na płaszczyźnie, co przedstawiono na rys. 7.



Rysunek 7: Obliczenie kinematyki odwrotnej - rzut z góry

Na podstawie twierdzenia cosinusów:

$$r^2 = l_1^2 + l_2^2 - 2 l_1 l_2 cos(\pi - q_2)$$

Stąd:

$$cosq_2 = \frac{x^2 + y^2 - l_1^2 - l_2^2}{2 l_1 l_2} = R$$

Na podstawie własności trygonometrycznych:

$$sinq_2 = \pm\sqrt{1 - R^2}$$

Zatem:

$$q_2 = arctg\left(\frac{\pm\sqrt{1 - R^2}}{R}\right)$$

Natomiast zmienną przegubową q_1 można okreslić jako:

$$q_1 = \alpha - \beta$$

gdzie

$$\alpha = arctg\left(\frac{y}{x}\right)$$

natomiast kąt β można określić, korzystając z twierdzenia sinusów oraz własności trygonometrycznych:

$$\frac{l_2}{sin\beta} = \frac{r}{sin(\pi - q_2)}$$

Stąd:

$$sin\beta = \frac{l_2 s_2}{r}$$

oraz

$$cos\beta = \pm\sqrt{\frac{r^2 - l_2^2 s_2^2}{r^2}}$$

Zatem

$$\beta = arctg\left(\frac{l_2 s_2}{l_1 + l_2 c_2}\right)$$

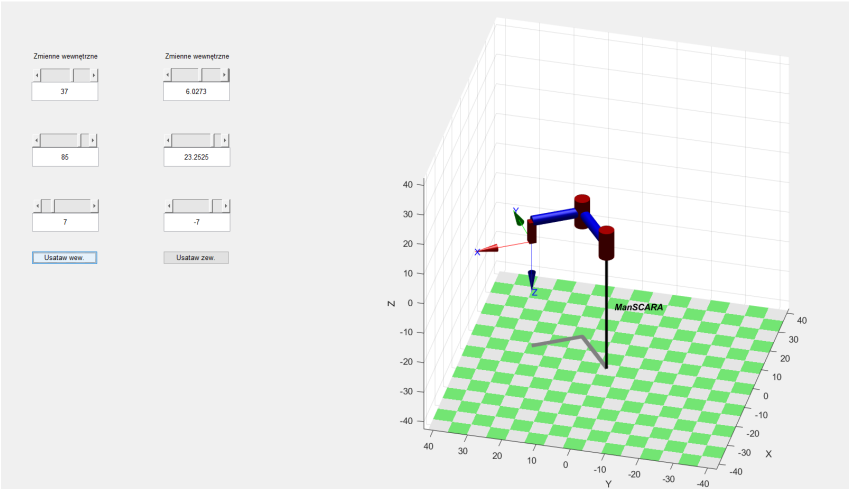
Ostatecznie równania kinemtayki odwrotnej można przedstawić jako:

$$\begin{cases} q_1 = arctg\left(\frac{y}{x}\right) - arctg\left(\frac{l_2 s_2}{l_1 + l_2 c_2}\right) \\ q_2 = arctg\left(\frac{\pm\sqrt{1-R^2}}{R}\right) \\ q_3 = -z \end{cases} \tag{5}$$

Dla $R = \frac{x^2+y^2-l_1^2-l_2^2}{2l_1 l_2}$

6.3 Weryfikacja

W celu weryfikacji zaimplementowano komputerową weryfikację przeprowadzonych obliczeń z wykorzystaniem środowiska MATLAB. Przykład działania symulacji przedstawiono na rys. 8.



Rysunek 8: Weryfikacja obliczeń kinematyki prostej i odwrotnej

7 Opis działania programu

Bieżące wartości współrzędnych manipulatora są przechowywane w strukturze

```
1 struct position {
2   int16_t q1;
3   int16_t q2;
```

```
4   int16_t q3;
5   int16_t x;
6   int16_t y;
7   int16_t z;
8 };
```

W chwili rozpoczęcia pracy sterownika wartości te są inicjowane do pozycji zerowej, tzn. manipulator winien być ustawiony wzdłuż osi OX. Zatem $x = 320$, natomiast pozostałe wartości są zerowe.

7.1 Wyznaczanie pozycji enkoderów

W celu realizacji sterowania przegubami rotacyjnymi niezbędne jest ciągłe odczytywanie bieżącej pozcyji enkoderów inkrementalnych. Jest to pozycja na końcu wału silnika DC, z którym enkoder jest stowarzyszony. Odczytywanie pozycji realizuje funkcja int16_t Current_position1 (int16_t CNT_current, int16_t CNT_prev). Jej implementacja została przedstawiona na listingu poniżej.

```
1 int16_t position;
2 Rotations_counter1(current_CNT, CNT_prev);
3 position = rotations1*IMPULS*PRZEKLADNIA + current_CNT/4;
4 return position;
```

Analogicznie zaimplementowano funckję dla drugiego przegubu rotacyjnego. Aktualna pozycja enkodera jest ustalana jako suma liczby obrotów enkodera i liczby impulsów w bieżącym obrocie. Aktualna liczba rotacji jest przechwywana w zmiennej rotations1. Jest ona mnożona przez współczynnik będący iloczynem liczby impulsów na obrót i przekładni silnika. W ten sposób pozycja jest odczytywna z wału silnika. Liczba obrotów enkodera jest aktualizowana w funkcji void Rotations_counter1 (int16_t current_CNT, int16_t CNT_prev). Gdy enkoder przechodzi z pozycji maksymalnej na zerową, to zmienna rotations1 jest inkrementowana, natomiast obrót w przeciwnym powoduje jej dekrementację.

Taka konwencja obliczania pozycji enkoderów pozwala na łatwe przeliczanie wartości zadawanych kątów. Bazując na własnościach konstrukcyjnych silnika (rozdział 3.2) przyjęto, że obrót wału o 1° oznacza przesunięcie o 1 pozycję. Zastosowanie przekładni 3:1 powoduje, że obrót przegubu o 1° odpowiada 3 pozycjom enkdera.

7.2 Tryb wyboru sterowania – menu

Na początku pracy sterownik znajduje się właśnie w tym trybie. Poszczególne opcje sterowania są wypisywane na ekranie. Wyboru można dokonać za pomocą joysticka. Pozostawienie go w stanie środkowym pozwala na wybór sterowania ręcznego we współrzędnych zewnętrznych. Przyciągnięcie joysticka maksymalnie do siebie umożliwia wybór sterowania ręczenego we współrzędnych przegubowych. Natomiast wychylenie joysticka maksymalnie od siebie powoduje wybór sterowania automatycznego z wykorzystaniem komputera. W celu rozpoznawania wybranego trybu sterowania zdefiniowano typ wyliczeniowy, który jest zapamiętywany w odpowiedniej zmiennej.

```
1 enum controlType {JoystickJoints, JoystickCoordinates, ComputerUSB};
```

Wybór następuje w wyniku naciśnięcia przycisku stowarzyszonego z joystickiem. W celu eliminacji drgań styków zaimplementowano prosty *debouncing*.

```
1 if(HAL_GPIO_ReadPin(SW_GPIO_Port,SW_Pin)==GPIO_PIN_RESET){
2     if(HAL_GetTick() - lastCheckButton > TIME){ //Debouncing
3       menu*=-1;
4       lastCheckButton = HAL_GetTick();
5       Joystick_prev=0;
6   }
7 }
```

Wciśnięcie przycisku powoduje zmianę znaku zmiennej menu. Tym samym powrót do menu wyboru następuje również po naciśnięciu przycisku.

7.3 Zadawanie współrzędnych przez użytkownika

Po przejściu w tryb sterowania na ekranie LCD prezentowany jest na zmianę typ wybranego sterowania, aktualna pozycja manipulatora we współrzędnych wewnętrznych oraz zewnętrznych.

7.3.1 Tryb pracy ręcznej

Tryb pracy ręcznej umożliwia sterowanie manipulatorem z wykorzystaniem podłączonych joysticków. Odczytywana wartość na wejściu analogowym jest odpowiednio skalowana tak, aby mieściła się w przedziale $[-128; 128]$. Dzięki temu wychylenie joysticka w jedną stronę powoduje zwiększenie wartości współrzędnej, a w drugą stronę - zmniejszenie. Zapewnia to funkcja int16_t JoystickScale(uint16_t), której implementację przedstawiono na listingu poniżej:

```
1 int16_t JoystickScale(uint16_t read){
2   int16_t tmp = ((read/16) - 128);
3   if((tmp < DEAD_SPACE) && (tmp > - DEAD_SPACE)) //Strefa martwa
4     tmp = 0;
5   return tmp;
6 }
```

Tak obliczana wartość jest dodawana do aktualnej pozycji danej współrzędnej, aby wymusić ruch manipulatora. Jest to realizowane w następujacy sposób:

```
1 if(restriction((JoystickScale(Joystick[0])+position1)/3))
2       q1_read = (JoystickScale(Joystick[0])+position1)/3; //Przeliczanie nowej
            pozycji na kąt
```

Funkcja void restriction(int16_t) sprawdza, czy dany kąt spełnia założone ograniczenia sterowania. W przypadku zadawania współrzędnych wewnętrznych jest to niezbędne, aby zabezpieczyć wartość q_read przed trwałym wyjściem poza zakres. Wartość z joysticka dla przegubu translacyjnego jest dodatkowo skalowana za względu na mniejszy zakres ruchu tego przegubu.

7.3.2 Sterowanie automatyczne

Sterowanie automatyczne polega na zadawaniu współrzędnych za pomocą terminala. Komunikacja sterownika z komputerem została zrealizowana za pomocą USB. Wysłanie odpowiedniej ramki danych powoduje zapisanie przesłanych danych i wykonanie sterowania na ich podstawie. W przypadku przesłania błędnej ramki danych zwrócony zostanie komunikat. Poniżej przedstawiono strukturę przesyłanych ramek danych za pomocą wyrażeń regularnych:

1. we współrzędnych wewnętrznych

$$W\ \Big([+|-][0-9]\{2\}\Big)\{3\}$$

2. we współrzędnych zewnętrznych

$$Z\ \Big([+|-][0-9]\{3\}\Big)\{2\}\ [+|-][0-9]$$

Kolejne dane liczbowe winny być odzielone od siebie spacją. Przyjęta taka konwencja pozwala na zadawanie wartości z dokładnością do $1°$ dla przegubów rotacyjnych; natomiast dla przegubu translacyjnego oraz współrzędnych zewnętrznych - z dokładnością do $1mm$.

7.4 Sterowanie

Kiedy odczytane zostaną zadane nowe współrzędne manipulatora, obsługiwane jest jego sterowanie. Sposób sterowania jest zawsze taki sam, niezależnie od metody zadawania współrzędnych przez użytkowinka. W celu rozróżnienienia współrzędnych, w których ma być realizowane sterowanie, zdefiniowana typ wyliczeniowy:

```
1 enum moveType {Joints, Coordinates};
```

Po wykonaniu ruchu manipulatora obliczane są bieżące współrzędne zewnętrzne efektora na podstawie współrzędnych przegubowych za pomoą funkcji void Kinematics(), która bazuje na równaniach kinematyki prostej. Te wartości prezentowane są na ekranie wyświetlacza LCD.

7.4.1 Współrzędne wewnętrzne

W przypadku zadania nowych wartości przegubowych wywoływane są bezpośrednio funkcje sterujące poszczególnymi przegubami:

- void Go1 (int16_t desired_position, int16_t current_position); dla pierwszego przegubu rotacyjnego
- void Go2 (int16_t desired_position, int16_t current_position); dla drugiego przegubu rotacyjnego
- void Go3 (int16_t desired_position); dla przegubu translacyjnego

Przykład implementacji funkcji powodującej ruch dla przegubu translacyjnego przedstawiono poniżej.

```
1  void Go2 (int16_t desired_position, int16_t current_position){
2    int16_t difference = desired_position - current_position;
3    uint8_t direction = (difference > 0) ? FORWARD : BACKWARD;
4    float u = pid(desired_position, current_position);
5
6    if(difference<HISTEREZA && difference>-HISTEREZA)
7      Motor2_control(STOP,0);
8    else{
9      if(u>200 || u<-200)
10       Motor2_control(direction, 200);
11     else
12       Motor2_control(direction, (uint8_t)u);
13   }
14 }
```

Funkcja wyznacza ruch na podstawie bieżącej pozycji manipulatora i pozycji zadanej. Prędkość silnika, a więc wypełnienie sygnału PWM, jest ustalana za pomocą zaimplementowanego regulatora PID. Dodatkowo, w przypadku gdy odległość między pozycjami jest mała, wymuszane jest zatrzymanie silnika. Natomiast gdy różnica jest zbyt duża nieprzekraczana jest wartość wypełnienia 200. Funkcja void Motor2_control(enum Direction Dir, uint8_t Velocity) pozwala na zadanie kierunku obrotu silnika poprzez ustawienie odpowiednich pinów w stan wysoki, a także prędkości obortu poprzez przypisanie odpowiedniego wypełnienia generowanemu sygnałowi PWM.

Funkcja sterująca przegubem translacyjnym sprowadza się do przeskalowania zadanej wartości do dopuszczalnego zakresu wypełnienia PWM $(50; 100)$ i ustawienie tego wypełnienia.

```
1  void Go3(int16_t desired_position){
2    int servo_duty = desired_position+50;
3    __HAL_TIM_SET_COMPARE(&htim4,TIM_CHANNEL_1, servo_duty);
4  }
```

7.4.2 Współrzędne zewnętrzne

Dla zadanych współrzędnych zewnętrznych obliczane są współrzędne wewnętrzne na podstawie równań kinemtyki odwrotnej. Realizuje to funkcja int Position_from_coordinates_rot_joints(int16_t x, int16_t y, int16_t current_position1, int16_t current_position2) dla przegubów rotacyjnych oraz funkcja int Position_from_coordinates_prism_joint (int16_t z). Funkcje te odpowiednio przeliczają zadane wartości x, y, z na wartości współrzędnych przegubowych i sprawdzają, czy obliczone wartości spełniają zadane ograniczenia. Jeśli tak, to przypisują obliczone wartości do struktury current_position, przechowującej bieżące współrzędne manipulatora, i pozwalają na wykonanie ruchu manipulatora według metody opisanej w rozdziale 7.4.1.

Funkcja realizująca kinematykę odwrotną dla przegubów rotacyjnych pozwala również na wybór pozycji, do której dążyć będzie manipulator. Zadanie współrzędnych zewnętrznych dla manipulatora SCARA daje bowiem pewną dwuznaczność - przeguby mogą zostać ustawione na 2 sposoby, realizując pozycję "ramię w dół" bądź "ramię w górę". Wybór pozycji polega na obliczeniu odległości bieżącej pozycji enkoderów od każdej z dwóch nowo wyznaczonych pozycji. Odległość obliczana jest na podstawie normy euklidesowej. Wybierana jest opcja bliższa pozycji bieżącej – kryterium minimalizujące przemieszczanie się manipulatora.

7.5 Regulator PID

Impelmentację regulatora PID oraz wyznaczanie współczynników regulatora zrealizowano w oparciu o artykuł [2]. Wykorzystano dyskretną wersję regulatora. Wartość sygnału sterującego jest opisana wzorem:

$$u = k_p \cdot e + k_i \cdot \Sigma_{j=0}^{n} e_j + k_d \cdot (e_n - e_{n-1})$$

Implementację tej zależności przedstawiono na listingu poniżej.

```
1  float pid(int32_t desired, int32_t current){
2    float u, e_d_current;
3
4    pid_param.e = desired - current;
5
6    pid_param.e_sum += pid_param.e;
7    if(pid_param.e_sum > MAX_E_SUM) //Aby zapobiec nadmiarowemu przyrostowi tej
         wartości
8      pid_param.e_sum = MAX_E_SUM;
9    else if(pid_param.e_sum < -MAX_E_SUM)
10     pid_param.e_sum = -MAX_E_SUM;
11
12   e_d_current = pid_param.e_d - pid_param.e;
13
14   u = pid_param.kp*pid_param.e + pid_param.ki*pid_param.e_sum + pid_param.kd*
         e_d_current;
15
16   return u;
17 }
```

Wartości parametrów regulatora zostały ustalone w procesie strojenia i zadane przy inicjalizowaniu sterownika. Regulator PID oblicza wartość sygnału sterującego na podstawie uchybu, który jest różnicą między żądaną wartością pozycji a wartością bieżącą. Sygnał sterujący jest wykorzystywany do zadawania prędkości obrotu silnika.

W trakcie implementacji sterowania korzystano z [4] oraz [3].

8 Podsumowanie

W ramach projektu przygotowano manipultor SCARA o trzech stopniach swobody - dwóch rotacyjnych i jednym translacyjnym.

Sterowanie manipulatora zostało zrealizowane w oparciu o plytkę STM32F411VET6 Discovery. Do realizacji zadania wykorzystano wiele peryferiów, m. in. timery, wejścia analogowe, interfejsy komunikacyjne (USB, I2C).

Celem łatwiejszego połączenia wykorzystanych urządzeń zewnętrznych przygotowano projekt płytki elektronicznej, którą wykorzystano jako nakładkę. Wykonanie przygotowanego projektu zlecono firmie trzeciej.

Przygotowano także projekt konstrukcji mechanicznej, który został wykonany samodzielnie. W tym celu wiele elemntów zostało przygotowanych za pomocą techniki druku 3D. Ruch w przegubach rotacyjnych gwarantują silniki DC wyposażone w enkodery kwadraturowe, natomiast w przegubie translacyjnym - serwo modelarskie.

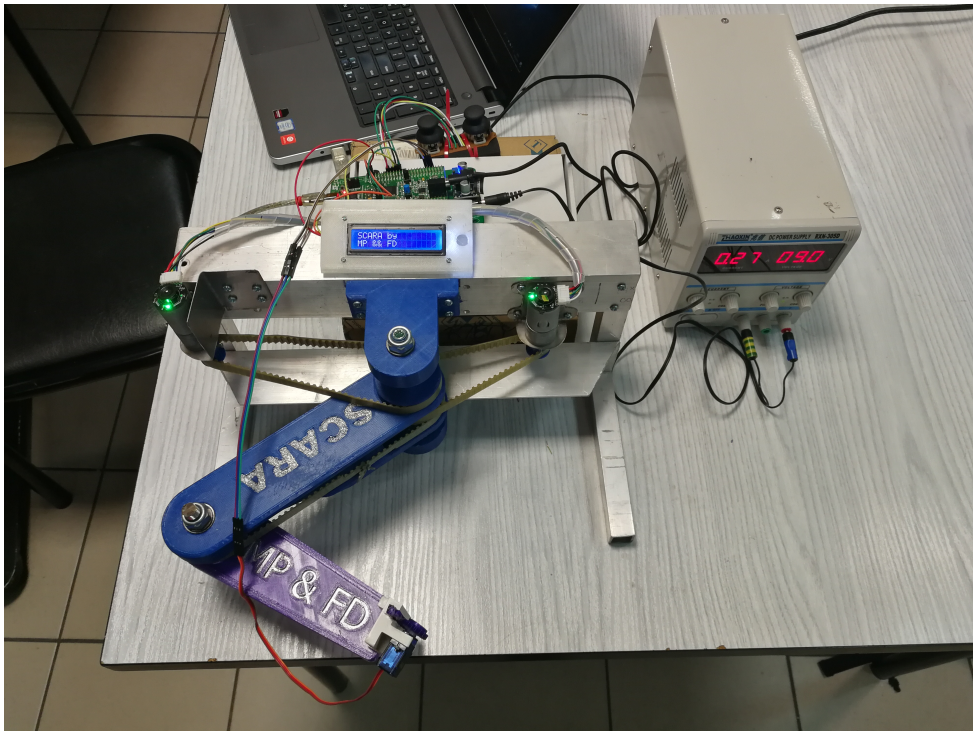
Manipulator zapewnia możliwość sterowania ręcznie (joystickiem) bądź automatycznie (z poziomu terminala). Ponadto pozwala na zadawanie pozycji we współrzędnych wewnętrznych oraz zewnętrznych. Zaimplementowano algorytm kinematyki odwrotnej, który przelicza zadawane współrzędne zewnętrzne na współrzędne przegubowe. Sterownik został wyposażony w dodatkowe funkcje takie, jak interaktywna możliwość wyboru sterowania przez użytkownika czy prezentowanie aktualnej pozycji robota na ekranie LCD.

W ostatniej fazie realizacji projektu napotkano poważny problem - niemożność automatycznego sterowania manipulatorem. O ile sterowanie ręczne za pomocą joysticka funkcjonowało na pozór poprawnie, o tyle sterowanie automatyczne zawsze generowało błędy pozycji. Znaleziono jednak przyczynę tego problemu. Powodem złego zachowania manipulatora było niepoprawne ukierunkowanie obrotów silników. Gdy zadawany był kąt dodatni, to silnik dążył obrotami do wartości ujemnych (tylko ograniczenia

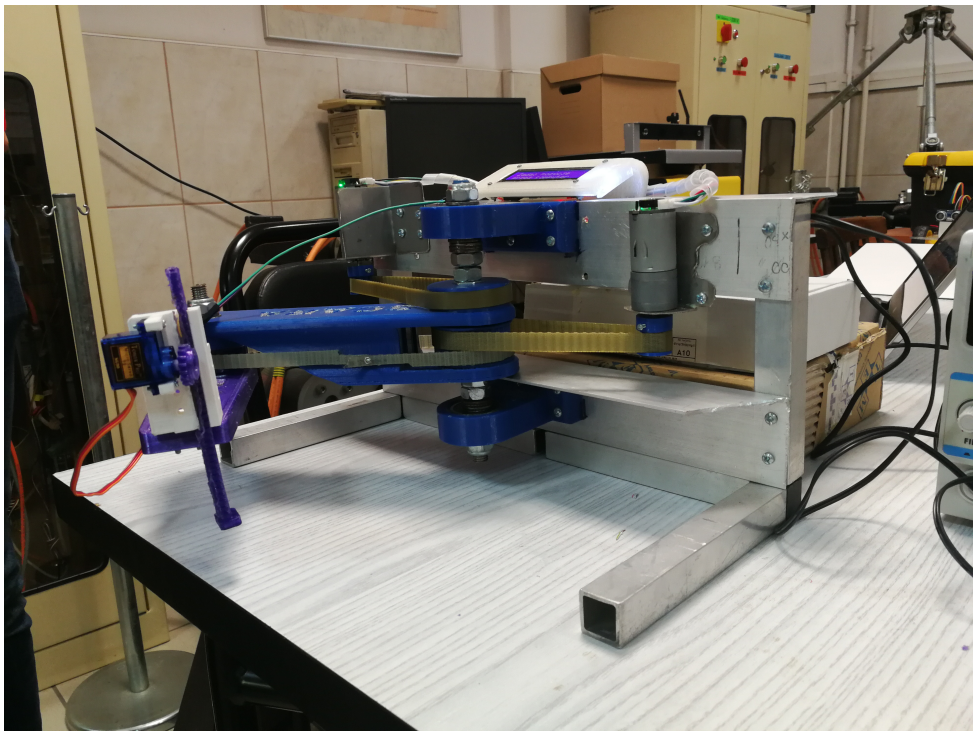
mechaniczne powodowały, że manipulator zatrzymywał się). Analogiczna sytuacja miała miejsce dla kąta ujemnego. Problem rozwiązano poprzez zmianę implementacji kodu sterowania - zamieniono piny ustwiane odpowiednio w stan wysoki/niski dla ruchu do przodu lub do tyłu.

Projekt pozostawia przestrzeń do dalszego rozwoju. Przegub translacyjny manipulatora można wyposażyć w efektor (np. elektromagnes). Dobrym rozwiązaniem byłoby dodatkowe zabezpieczenie manipulatora (oprócz software'u) przed przekraczaniem dopuszczalniej przestrzeni roboczej. Można do tego wykorzystać krańcówki. Ich zastosowanie umożliwi również odnajdywanie pozycji zerowej przez manipulator na początku działania.

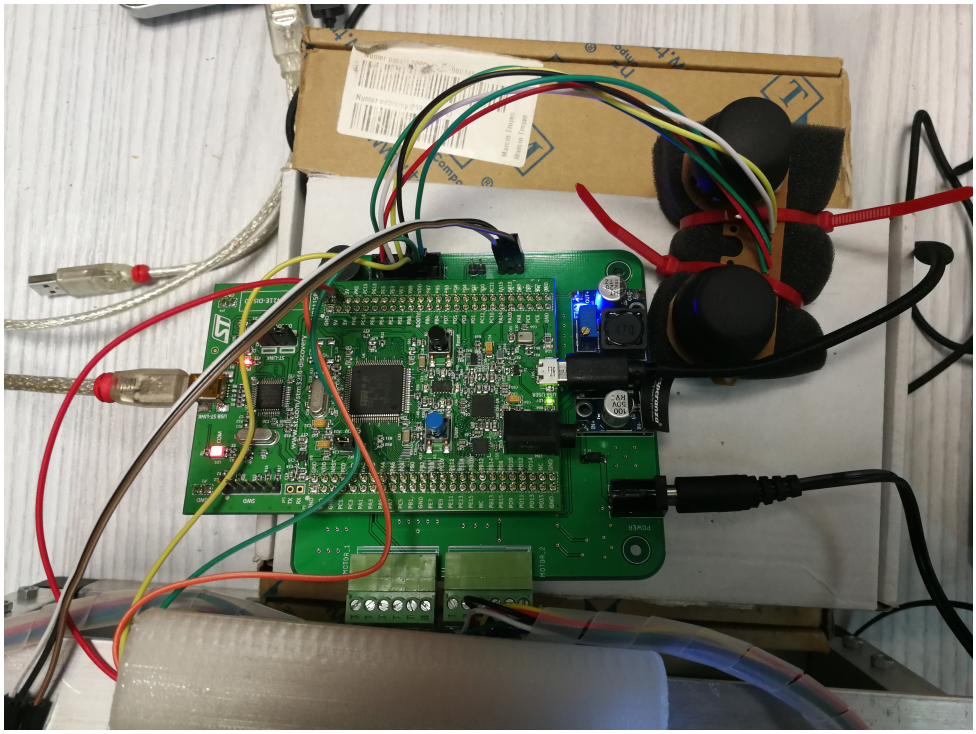
Poniżej, na rys. 9, 10 oraz 11, umieszczono zdjęcia przedstawiające konstrukcję manipulatora wyposażoną w sterownik.



Rysunek 9: Złożony robot SCARA wraz z elektroniką



Rysunek 10: Złożona mechanika manipulatora SCARA w przybliżeniu



Rysunek 11: Sterownik manipulatora SCARA

Literatura

[1] DFRobot. Specyfikacja silnika DC - DFR-09531. Sty. 2019.

[2] Forbot. Jak zaimplementować regulator PID dla silnika z enkoderem? Sier. 2015.

[3] M. Galewski. STM32. Aplikacje i ćwiczenia w języku C. Sty. 2011.

[4] K. Paprocki. Mikrokontrolery STM32 w praktyce. Sty. 2009.

[5] T. Pro. SERVO MOTOR SG90 DATA SHEET. Sty. 2019.

[6] M. Spong, M. Vidyasagar. Dynamika i sterowanie robotów. Sty. 1997.

[7] ST. Discovery kit with STM32F411VE MCU. User Manual. Gru. 2014.

[8] K. Tchoń, A. Mazur, I. Dulęba, R. Hossa, R. Muszyński. Manipulatory i roboty mobilne: modele, planowanie ruchu, sterowanie. Sty. 2000.

[9] Toshiba. TB6612FNG datasheet. Maj 2008.

[10] Xusniyor. Biblioteka obsługująca komunikację z wyświelaczem LCD za pomocą standardu I2C . Maj 2019.