Projekt

Sterowniki robotów

Dokumentacja

Manipulator MiLo

Skład grupy:
Bartosz Piech, 249028
Termin: czTP19

Prowadzący:
dr inż. Wojciech Domski

2 czerwca 2021

Spis treści

1 Opis projektu

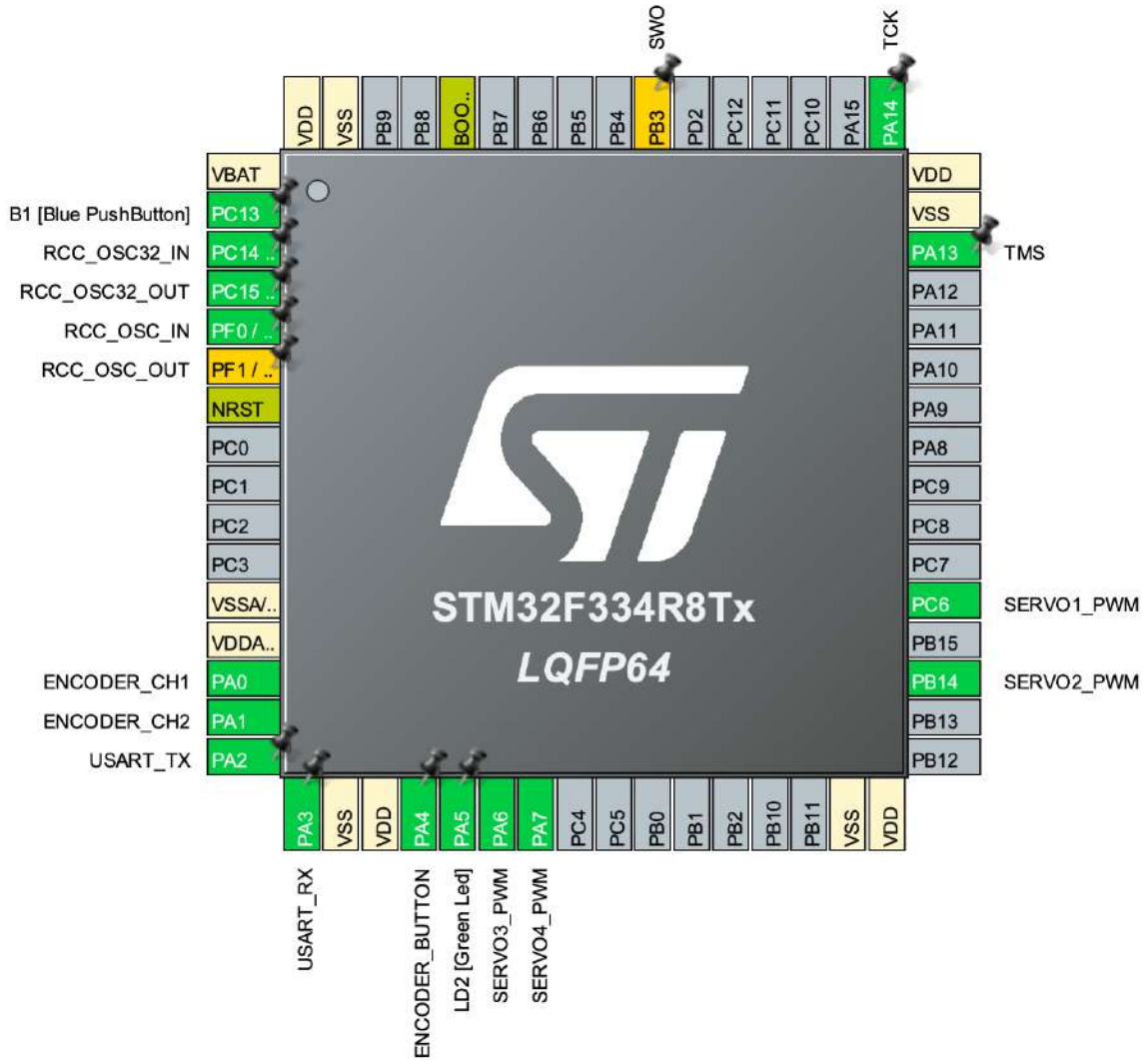
Celem projektu było skonstruowanie manipulatora o czterech stopniach swobody [4], którego zadaniem jest liczenie algorytmu kinematyki odwrotnej [6] na mikrokontrolerze. Konstrukcja jest sterowana za pomocą Płytki deweloperskiej STM32 NUCLEO-F334R8. Do poruszania ramionami oraz podstawą zostały użyte serwomechanizmy cyfrowe.
Sterowanie manipulatorem jest możliwe za pomocą:

- enkodera inkrementalnego - użytkownik steruje kątem obrotu serwomechanizmu, po przyciśnięciu przycisku zmienia się serwomechanizm, który jest sterowany,
- aplikacji komputerowej - sterownik podłączony z komputerem za pomocą interfejsu UART pozwalający na poruszanie manipulatorem we współrzędnych zewnętrznych oraz wewnętrznych,

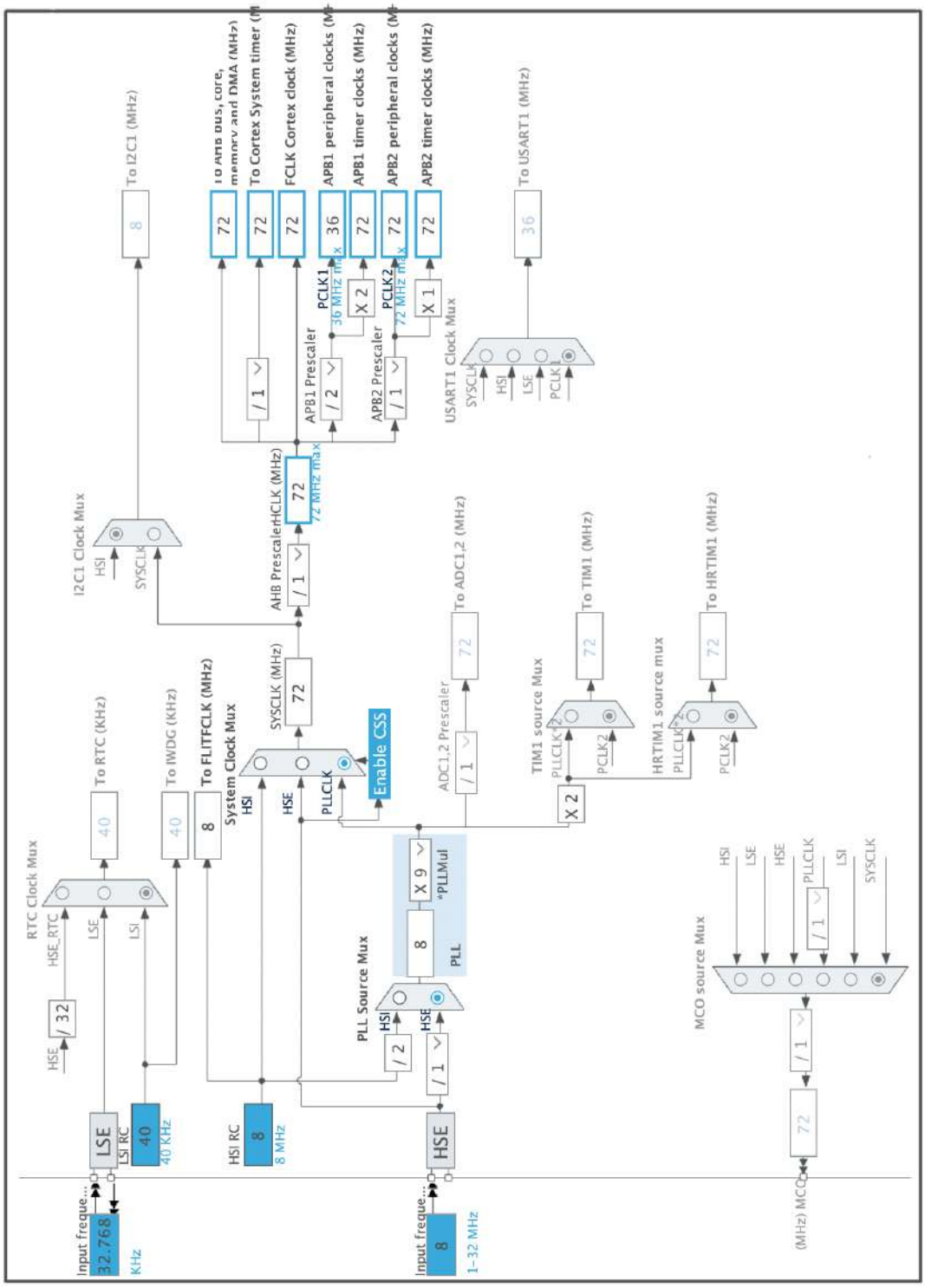
Do zbudowania kostrukcji potrzebne były następujące elementy:

- STM32 NUCLEO-F334R8 [5]
- trzy serwomechanizmy typu standard FT53250M,
- serwomechanizm typu micro MG90S,
- zewnętrzne zasilanie - akumulator Li-Ion,
- enkoder inkrementalny KY-040,
- Moduł Bluetooth HC-05

2 Konfiguracja mikrokontrolera



Rysunek 1: Konfiguracja wyjść mikrokontrolera w programie STM32CubeMX



Rysunek 2: Konfiguracja zegarów mikrokontrolera

2.1 Konfiguracja pinów

Numer pinu	PIN	Tryb pracy	Funkcja/etykieta
2	PC13	ANTI_TAMP GPIO_EXTI13	B1 [Blue PushButton]
3	PC14	OSC32_IN* RCC_OSC32_IN	
4	PC15	OSC32_OUT* RCC_OSC32_OUT	
5	PF0	OSC_IN* RCC_OSC_IN	
6	PF1	OSC_OUT*	RCC_OSC_OUT
14	PA0	$TIM2_{CH}1$	ENCODER_CH1
15	PA1	$TIM2_{CH}2$	ENCODER_CH2
16	PA2	USART2_TX	USART_TX
17	PA3	USART2_RX	USART_RX
20	PA4	GPIO_EXTI4	ENCODER_BUTTON
21	PA5	GPIO_Output	LD2 [Green Led]
22	PA6	TIM16_CH1	SERVO3_PWM
23	PA7	TIM17_CH1	SERVO4_PWM
35	PB14	TIM15_CH1	SERVO2_PWM
37	PC6	TIM3_CH1	SERVO1_PWM
46	PA13*	SYS_JTMS-SWDIO	TMS
49	PA14*	SYS_JTCK-SWCLK	TCK
55	PB3*	SYS_JTDO-SWO	SWO

Tabela 1: Konfiguracja pinów mikrokontrolera

2.2 USART

Interfejs szeregowy USART został użyty do komunikacji między aplikacją komputerową a mikrokontrolerem oraz do debugowania programu napisanego na mikrokontrolerze, komunikacja odbywa się przez kabel USB lub moduł Bluetooth HC-05. Mikrokontroler nasłuchuje wiadomości oraz wysyła odpowiedzi poprzez USART.

Parametr	Wartość
Baud Rate	115200
Word Length	8 Bits (including parity)
Parity	None
Stop Bits	1

Tabela 2: Konfiguracja peryferium USART

2.3 TIM2

Timer 2 skonfigurowano w trybie Encoder Mode, który pozwala w prosty sposób odbierać dane z enkodera KY-040, które pozwalają sterować ustawieniem kątów manipulatora.

Parametr	Wartość
Prescaler (PSC - 16 bits value)	0
Counter Mode	Up
Counter Period (AutoReload Register - 16 bits value)	65535
Internal Clock Division (CKD)	No Division
auto-reload preload	Disable
Encoder	Wartość
Encoder Mode	Encoder Mode TI1
Polarity	Rising Edge
IC Selection	Rising Edge
Prescaler Division Range	No Division Edge
Input Filter	15

Tabela 3: Konfiguracja TIM3

2.4 TIM3

Timer 3 został skonfigurowany w trybie PWM Generation CH1 dla kanału pierwszego. Generuje sygnały PWM służące do sterowania serwomechanizmu typu micro, który jest umieszczony na końcu ostatniego ramienia manipulatora. Jest to serwo analogowe z metalową zębatką, częstotliwość serwa została ustawiona na 50Hz, przy takiej częstotliwości serwomechanizm działa prawidłowo.

Parametr	Wartość
Prescaler (PSC - 16 bits value)	71
Counter Mode	Up
Counter Period (AutoReload Register - 16 bits value)	1999
Internal Clock Division (CKD)	No Division
Repetition Counter (RCR - 8 bits value)	0
auto-reload preload	Enable
PWM Generation Channel 1	Wartość
Fast Mode	Disable

Tabela 4: Konfiguracja TIM3

2.5 TIM6

Timer 6 jest używany przy debugowaniu programu, jego częstotliwość jest ustawiona na 1Hz, w funkcji callback, która uruchamia się co jedną sekunde, program wyświetla aktualne dane pochodzące z serwomechanizmu.

Parametr	Wartość
Prescaler (PSC - 16 bits value)	7199
Counter Mode	Up
Counter Period (AutoReload Register - 16 bits value)	999
auto-reload preload	Disable
Trigger Event Selection	Reset

Tabela 5: Konfiguracja TIM6

2.6 TIM15

Timer 15 został skonfigurowany w trybie PWM Generation CH1 dla kanału pierwszego. Generuje sygnał PWM służące do sterowania serwomechanizmu, który jest umieszczony w podstawie jako serwo numer 1. Jest to serwo cyfrowe z metalową zębatką typu standard. Częstotliwość serwomechanizmu została ustawiona na 50Hz.

Parametr	Wartość
Prescaler (PSC - 16 bits value)	71
Counter Mode	Up
Counter Period (AutoReload Register - 16 bits value)	1999
Internal Clock Division (CKD)	No Division
Repetition Counter (RCR - 8 bits value)	0
auto-reload preload	Enable
PWM Generation Channel 1	Wartość
Fast Mode	Disable

Tabela 6: Konfiguracja TIM15

2.7 TIM16

Timer 16 został skonfigurowany w trybie PWM Generation CH1. Generuje sygnał PWM służący do sterowania serwomechanizmu 2-go. Jest to serwo cyfrowe, jego częstotliwość została ustawiona na 50Hz, tak jak w Timerze 15.

Parametr	Wartość
Prescaler (PSC - 16 bits value)	71
Counter Mode	Up
Counter Period (AutoReload Register - 16 bits value)	1999
Internal Clock Division (CKD)	No Division
Repetition Counter (RCR - 8 bits value)	0
auto-reload preload	Enable
PWM Generation Channel 1	Wartość
Fast Mode	Disable

Tabela 7: Konfiguracja TIM16

2.8 TIM17

Timer 17 został skonfigurowany w trybie PWM Generation CH1. Generuje sygnał PWM służący do sterowania serwomechanizmu 3-go, który jest przymocowany na końcu pierwszego ramienia. Jest to serwo cyfrowe, jego częstotliwość została ustawiona na 50Hz.

Parametr	Wartość
Prescaler (PSC - 16 bits value)	71
Counter Mode	Up
Counter Period (AutoReload Register - 16 bits value)	1999
Internal Clock Division (CKD)	No Division
Repetition Counter (RCR - 8 bits value)	0
auto-reload preload	Enable
PWM Generation Channel 1	Wartość
Fast Mode	Disable

Tabela 8: Konfiguracja TIM17

3 Urządzenia zewnętrzne

Wykorzystane urządzenia zewnętrzne służą do poruszania manipulatorem, komunikacji oraz odbierania danych od użytkownika

3.1 3x Serwomechanizm Feetech FT5325M

Zastosowane serwomechanizmy typu standard, są zasilane napięciem od 6V do 7,4V. Zakres ich ruchu wynosi od 0^o do 180^o. Ponadto są to serwa cyfrowe, czyli mają większą dokładność oraz płynność ruchów od serw analogowych. Mają również moment obrotowy wynoszący $25kg \cdot cm$. dwa z tych serw są umieszczone w podstawie, oraz jedno na końcu pierwszego ramienia.

3.2 1x Serwomechanizm MG90S

Jest to serwo analogowe, zostało wybrane ze względu na małą wagę oraz metalowe zębatki. Jest umieszczone na końcu drugiego ramienia. Pełni rolę chwytaka. Serwo ma zakres ruchu od 0^o do 180^o, oraz waży $9g$.

3.3 Enkoder inkrementalny KY-040

Enkoder został użyty ze względu na brak zakłóceń, w porównaniu do potencjometrów użytych wcześniej. Został podłączony do mikrokontrolera, aby sterować kątami nachylenia ramion.

3.4 Moduł Bluetooth HC-05

Moduł służy do komunikacji pomiędzy mikorokontrolerem a aplikacją komputerową. Umożliwia wygodniejsze połączenie interfejsu UART pomiędzy mikrokontrolerem a komputerem.

Parametr	Wartość
Protokół transmisji	Bluetooth v2.0 EDR
Częstotliwość	2.4GHz
Zasilanie	3.3V

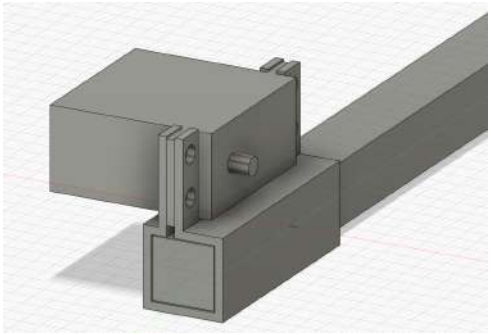
Tabela 9: Parametry modułu HC-05

4 Projekt elektroniki

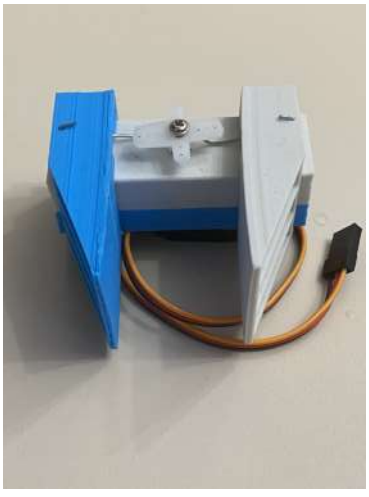
Projekt nie uwzględnia wykonywania dodatkowych płytek drukowanych. całość jest montowana przy użyciu płytki stykowej. Schemat projektu pokazany jest na następnej stronie.

5 Konstrukcja mechaniczna

Konstrukcja została wykonana z aluminiowej kwadratowej rury o profilu 1.5cm. Taki profil został wybrany ze względu na małą wagę, oraz dużą wytrzymałość. Zostały zaprojektowane uchwyty do serwomechanizmów, tak aby stabilnie trzymały się na końcach ramion manipulatora.



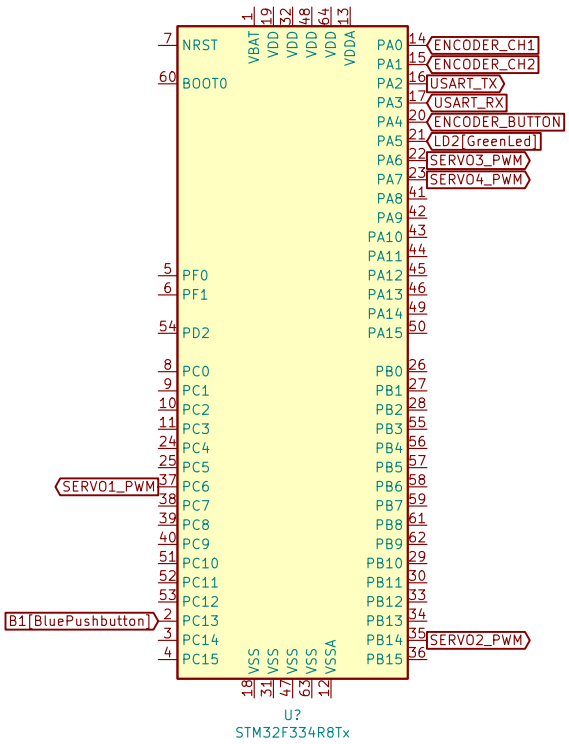
Rysunek 3: Zamodelowany uchwyt serwomechanizmu umieszczony na kwadratowej rurze



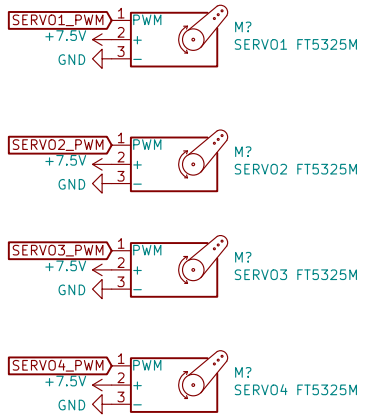
Rysunek 4: Efektor wydrukowany w technologii 3D

Model efektora został znaleziony w internecie [2] i wydrukowany za pomocą technologii 3D. Cały manipulator jest skręcony za pomocą śrub m3, pokazanego wyżej uchwytu, oraz kątowników przy podstawie, zrezygnowano z użycia plastikowych elementów przy skręcaniu podstawy, ponieważ przy dużym wychyleniu nie wytrzymywały oraz ulegały wygięciu. Serwomechanizm numer 1 jest umieszczony w imadle ze względu na relatywnie dużą wagę na końcu pierwszego ramienia (serwo typu standard), Efektor został przymocowany przy użyciu śrubki m2 oraz opasek zaciskowych, które spełniają swoją funkcję.

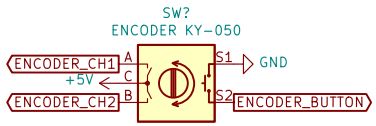
Rysunek 5: Fotografia aktualnego stanu manipulatora



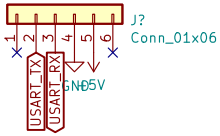
SERVOS



ENCODER



HC-05 BLUETOOTH MODULE



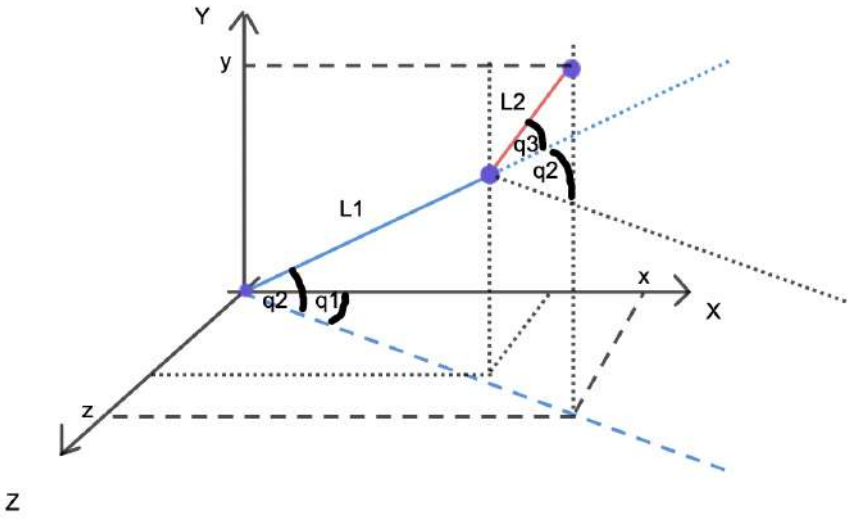
Sheet: /	
File: milo_projekt.sch	
Title:	
Size: A4	Date:
KiCad E.D.A. kicad (5.1.9)-1	Id: 1/1

6 Kinematyka odwrotna

Jednym z założeń projektowych było zaimplementowanie algorytmu liczenia kinematyki odwrotnej manipulatora. Za pomocą metody analitycznej [6] oraz geometrycznej udało się oblczyć kinematykę odwrotną dla q1 oraz q3. Nie udało się jednak obliczyć kinematyki dla q2, dlatego postanowiono zmienić podejście i użyć algorytmu FABRIK, który w krótkim czasie potrafi obliczyć przybliżone wartości

6.1 Metoda analityczna

Do wyliczenia kinematyki odwrotnej za pomocą metody analitycznej narysowano obrazek przedstawiający manipulator we współrzędnych XYZ.



Rysunek 6: Obrazek przedstawiający manipulator wraz z zaznaczonymi kątami

Następnie przy pomocy funkcji trygonometrycznych wyznaczono trzy równania:

$$x = (l_1 c_2 + l_2 c_{23}) * c_1$$

$$y = l_1 s_1 + l_2 s_{23}$$

$$z = l_1 c_2 + l_2 c_{23}) * s_1$$

Zadanie kinematyki odwrotnej polega na tym, że znając wartości x, y, z jesteśmy w stanie obliczyć wartości kątów $q1, q2, q3$. Przy pomocy powyższych wzorów obliczono, że:

$$q_1 = arctan(\frac{z}{x})$$

Po podniesieniu każdego z równań do kwadratu można otrzymać równanie:

$$q_3 = arccos(\frac{x^2 + y^2 + z^2 - l_1^2 - l_2^2}{2 l_1 l_2})$$

Niestety przy użyciu tej metody nie udało się obliczyć q_2, więc manipulator nie mógł w pełni liczyć zadania kinematyki odwrotnej.

6.2 Algorytm FABRIK

Postanowiono użyć algorytmu Forward And Backwards Reaching Inverse Kinematics [1], który jest szybkim iteracyjnym algorytmem pozwalającym na obliczenie zadania kinematyki odwrotnej. Algorytm działa w ten sposób, najpierw sprawdza czy zadany punkt jest w zasięgu manipulatora, następnie wykonuje naprzemiennie dwie operacje - backward i forward, backward przenosi współrzędne efektora w

zadany punkt, następnie oblicza wszystkie pozostałe punkty na podstawie odległości od nowych współrzędnych efektora, w rezultacie punkt początkowy manipulatora$(0, 0, 0)$ zostaje przesunięty. Natomiast operacja forward przesuwa punkt początkowy manipulatora spowrotem do początku układu współrzędnych i oblicza kolejne punkty manipulatora, aż dojdzie do efektora, operacja jest powtarzana, aż odległość końcowego punktu efektora będzie mniejsza niż błąd zadany przez użytkownika. Po skończonej operacji liczone są kąty $q1, q2, q3$ za pomocą operacji wektorowych.
Algorytm wstępnie został zaimplementowany [3] przy użyciu biblioteki graficznej p5.js, pozwalało to na wykrycie błędów występujących podczas implementacji. W kodzie zaimplementowano trzy funkcje odpowiadające za liczenie kinematyki odwrotnej algorytmem FABRIK, są to:

- backward() - odpowiada za przesunięcie punktu efektora w miejsce zadanych koordynatów, następnie cofa inne punkty,
- forward() - przesuwa punkt początkowy do koordynatów $(0, 0, 0)$, następnie przesuwa resztę punktów,
- calculate_angles() - odpowiada za liczenie kątów po przeprowadzeniu algorytmu FABRIK. Funkcje wyglądają następująco:

```
1  void
2  manipulator_backward(manipulator_t *m, int depth, double x, double y, double
       z) {
3    vector_init_d(&m->j2[1], x, y, z);
4    if (vector_mag(m->j2[1]) > m->total_len) {
5      printf("za daleko\n");
6      return;
7    }
8    vector_t tmp;
9    // substract end of the joint from it's beginning
10   vector_sub(&tmp, m->j2[0], m->j2[1]);
11   vector_normalize(&tmp);
12   // multiplying * length of a joint
13   vector_mult(&tmp, m->len[1]);
14   // adding end coords
15   vector_add(&tmp, tmp, m->j2[1]);
16   vector_init_v(&m->j2[0], tmp);
17
18   // set end of first joint and repeat the process
19   vector_init_v(&m->j1[1], m->j2[0]);
20   vector_sub(&tmp, m->j1[0], m->j1[1]);
21   vector_normalize(&tmp);
22   vector_mult(&tmp, m->len[0]);
23   vector_add(&tmp, tmp, m->j1[1]);
24   vector_init_v(&m->j1[0], tmp);
25
26   manipulator_forward(m, depth, x, y, z);
27 }
28
29 void
30 manipulator_forward(manipulator_t *m, int depth, double x, double y, double z
       ) {
31   depth--;
32   //printf("forward\n");
33   vector_init_v(&m->j1[0], m->origin);
34   vector_t tmp;
35   vector_sub(&tmp, m->j1[1], m->j1[0]);
36   vector_normalize(&tmp);
37   vector_mult(&tmp, m->len[0]);
38   vector_init_v(&m->j1[1], tmp);
39   vector_init_v(&m->j2[0], m->j1[1]);
40   //vector_print(m->j1[1]);
41
42   vector_sub(&tmp, m->j2[1], m->j2[0]);
43   vector_normalize(&tmp);
```

```
44    vector_mult(&tmp, m->len[1]);
45    vector_add(&tmp, tmp, m->j2[0]);
46    vector_init_v(&m->j2[1], tmp);
47
48    // error check
49    vector_init_d(&tmp, x, y, z);
50    vector_sub(&tmp, tmp, m->j2[1]);
51    //if (fabs(vector_mag(tmp)) > m->error || depth >= 0) {
52    // exit only when you reach depth
53    if (depth >= 0) {
54      manipulator_backward(m, depth, x, y, z);
55    }
56  }
57
58  void
59  manipulator_calculate_angles(manipulator_t *m) {
60    vector_t x_tmp;
61    vector_t y_tmp;
62    vector_t x_axis;
63
64    // set temporary vector to j1b on x plane to calculate q1 and q2
65    vector_init_d(&x_tmp, m->j1[1].x, 0, m->j1[1].z);
66    // set temporary vector to j2b
67    vector_init_v(&y_tmp, m->j2[1]);
68    // set x axis vector to compute q1
69    vector_init_d(&x_axis, 1, 0, 0);
70    // move our tmp vector to origin
71    vector_sub(&y_tmp, y_tmp, m->j2[0]);
72    m->q[0] = vector_angle(x_tmp, x_axis);
73    m->q[1] = vector_angle(x_tmp, m->j1[1]);
74    m->q[2] = m->q[1] + vector_angle(x_tmp, y_tmp);
75  }
```

7 Opis działania programu

7.1 Struktury pomocnicze

Do napisania programu głównego napisano kilka struktur pomocniczych, które pozwalają oddzielić warstwę hardware'ową od software'owej. Główną strukturą jest struktura manipulatora.

```
1  typedef struct {
2    // redundant
3    vector_t origin;
4    vector_t effector;
5
6    vector_t j1[2];
7    vector_t j2[2];
8
9    double q[3];
10   double len[2];
11   double total_len;
12   double error;
13
14   servo_t servo[3];
15 } manipulator_t;
16 void
17 manipulator_init(manipulator_t *m);
18 void
19 manipulator_print(manipulator_t m);
20 void
21 manipulator_update(manipulator_t *m);
```

Struktura ta posiada informacje o współrzędnych manipulora (reprezentowanych za pomocą struktury wektora, długościach boków, oraz struktury pomocniczej dla serwomechanizmów. Struktura wektora oraz funkcje operujące na niej wyglądają następująco:

```
typedef struct {
  double x;
  double y;
  double z;
} vector_t;
void
vector_init_d(vector_t *v, double x, double y, double z);
void
vector_init_v(vector_t *v, vector_t src);
void
vector_print(vector_t v);
double
vector_mag(vector_t v);
double
vector_dot(vector_t vec1, vector_t vec2);
double
vector_angle(vector_t vec1, vector_t vec2);
void
vector_add(vector_t *v, vector_t vec1, vector_t vec2);
void
vector_sub(vector_t *v, vector_t vec1, vector_t vec2);
void
vector_mult(vector_t *v, double num);
void
vector_normalize(vector_t *v);
```

Do sterowania serwomechanizmami użyto struktury, w której zdefiniowany jest uchwyt na timer, oraz kąt. Każdy serwomechanizm musi zostać skonfigurowany, aby na przykład osiągał pełny obrót o 180^o, w tym celu użyto zmiennych pwm_min oraz pwm_max.

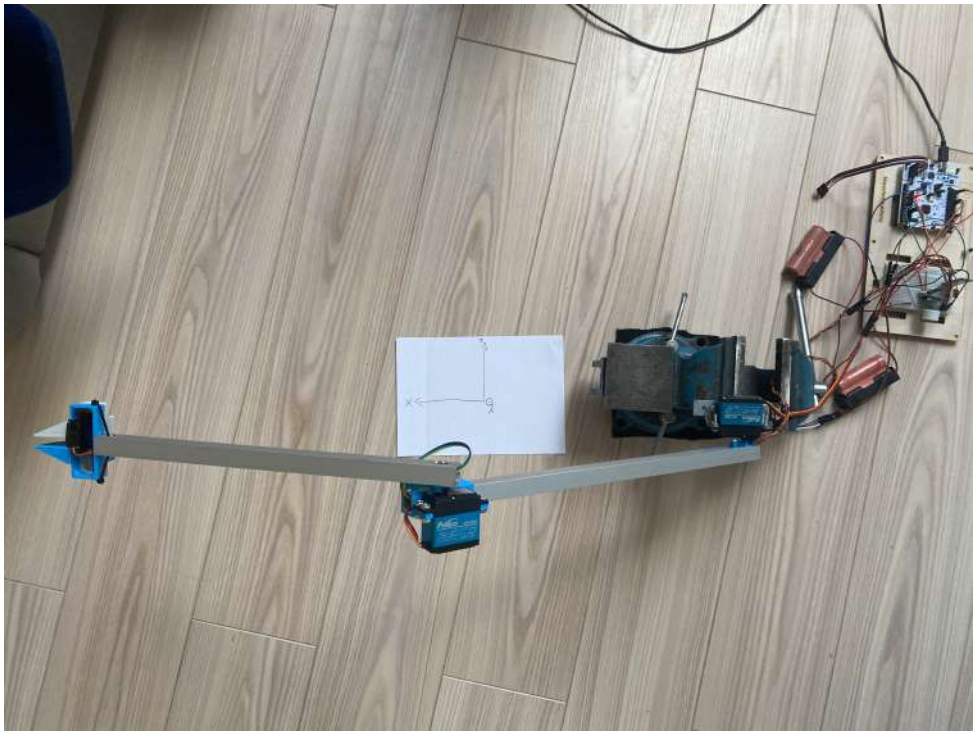
```
typedef struct {
  TIM_HandleTypeDef htim;
  uint8_t tim_channel;
  uint16_t angle;
  uint16_t angle_min;
  uint16_t angle_max;
  uint16_t pwm_min;
  uint16_t pwm_max;
} servo_t;

void
servo_init(servo_t* servo,
    TIM_HandleTypeDef htm,
    uint8_t tch,
    uint16_t a,
    uint16_t amin,
    uint16_t amax,
    uint16_t pmin,
    uint16_t pmax);

uint16_t
servo_step(servo_t servo);
void
servo_set(servo_t* servo, uint16_t angle, uint8_t mode);
void
servo_print(servo_t servo);
```

Program na początku inicjalizuje wszystkie niezbędne peryferia mikrokontrolera. Następnie tworzone są struktury przechowujące dane i uchwyty dla serwomechanizmów.

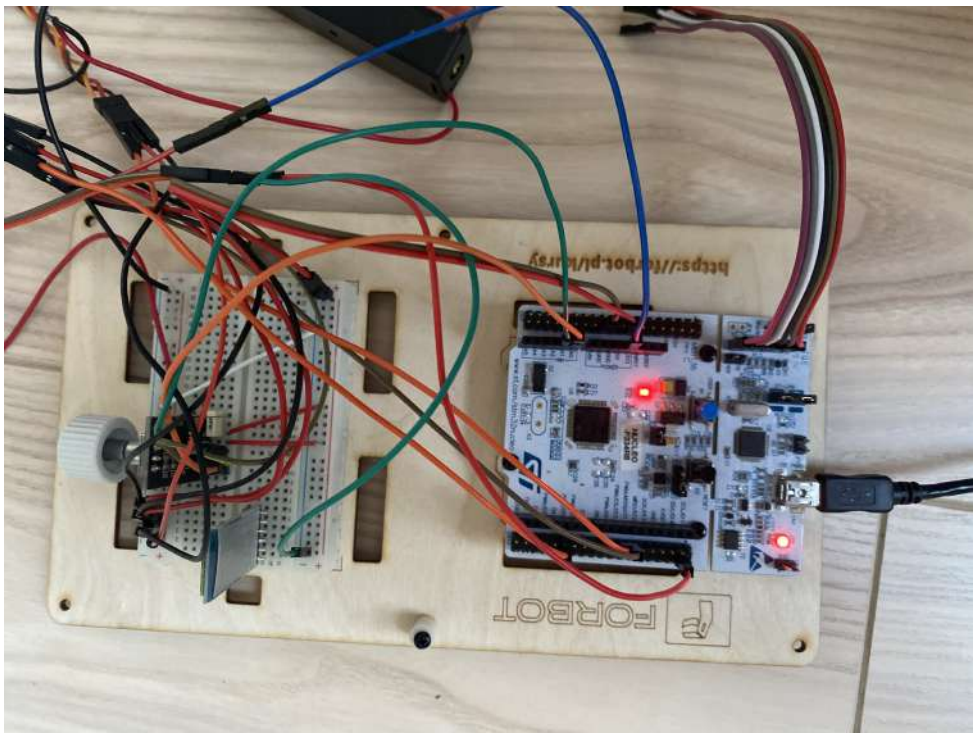
Rysunek 7: Zdjęcie manipulatora od boku w konfiguracji kątów $(45^o, 45^o, 0)$

8 Podsumowanie

Został wykonany projekt manipulatora przy użyciu platformy STM32, którego długości ramion wynoszą odpowiednio 40 i 30 cm. Do wykonania projektu użyto serwomechanizmów, które odpowiadały za rotację ramion. Do komunikacji posłużono się interfejsem UART, który jest prosty w konfiguracji i obsłudze. Do zasilania posłużyły akumulatory Li-Ion 18650, które dostarczają 7,4V potrzebne na zasilanie serwomechanizmów typu standard. Za sterowanie odpowiada enkoder inkrementalny, który zastąpił potencjometry, wykorzystywane na początku projektu w fazie testowej. Manipulatorem można sterować za pomocą aplikacji napisanej w Qt5, lub za pomocą enkodera, lub prosto poprzez interfejs UART. Chociaż nie udało się rozwiązać zadania kinematyki odwrotnej za pomocą metody analitycznej, algorytm FABRIK pozwala na osiągnięcie podobnego efektu.



Rysunek 8: Zdjęcie manipulatora od góry



Rysunek 9: Zdjęcie STM32 Nucleo podłączonego do manipulatora

Literatura

[1] J. L. Andreas Aristidou. Fabrik: A fast, iterative solver for the inverse kinematics problem. http://www.andreasaristidou.com/publications/papers/FABRIK.pdf, Maj 2011.

[2] A. Bäckström. Model efektora. https://bitbucket.org/adamb3_14/servoproject/src/master/CadFiles/Robot/stl/.

[3] B. Piech. Wizualizacja fabrik. https://editor.p5js.org/baetek/full/8vR_UdUl3.

[4] A. Skalik, D. Skrobek, P. Waryś, D. Cekus. Kinematic analysis of four degrees of freedom manipulator. http://www.imipkm.pcz.pl/wp-content/uploads/2015/03/Kinematic-analysis-of-four-degrees-of-freedom-manipulator.pdf, Czerw. 2010.

[5] STMicroelectronics. DB2196 STM32 Nucleo-64 boards 14.0 Data Brief. https://www.st.com/resource/en/data_brief/nucleo-f334r8.pdf, Paz. 2020.

[6] K. Tchoń, A. Mazur, I. Dulęba, R. Hossa, R. Muszyński. *Manipulatory i roboty mobilne*. Sty. 2000. https://kcir.pwr.edu.pl/~mucha/Skrypty/TMDHM_Manipulatory_i_roboty_mobilne.pdf.