

PROJEKT
STEROWNIKI ROBOTÓW

Dokumentacja
Manipulator
FiDerMan

Skład grupy:
Izabella CWOJDZIŃSKA, 249001

Termin: czwTP19

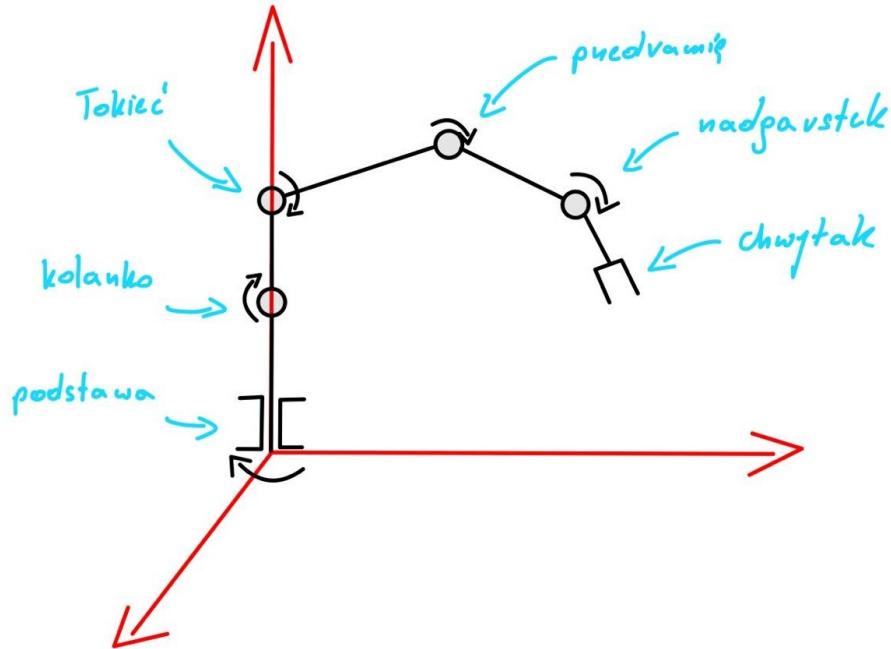
Prowadzący:
dr inż. Wojciech DOMSKI

Spis treści

1 Opis projektu	2
1.1 Zakres prac	2
2 Konfiguracja mikrokontrolera	3
2.1 Konfiguracja pinów	4
2.2 TIM1 oraz TIM3	4
2.3 UART	6
3 Urządzenia peryferyjne	6
3.1 Serwomechanizm MicroServo9g-SG90	6
3.2 Serwomechanizm MG996R	7
3.3 Moduł bluetooth HC-06	7
4 Konstrukcja mechaniczna	8
5 Kinematyka manipulatora	9
5.1 Kinematyka prosta	9
5.2 Kinematyka odwrotna	10
6 Program	10
7 Aplikacja	12
8 Podsumowanie	14
Bibliografia	15

1 Opis projektu

Założeniem mojego projektu było stworzenie manipulatora typu antropomorficznego, o 5 stopniach swobody, który to będzie wykorzystywany do przenoszenia niewielkich rzeczy, w obrębie danego pola. Sterowanie odbywa się za pomocą smartphona, poprzez aplikację, w której to mamy możliwość wyboru trybu pracy: automatycznego (opartego na odwrotnej kinematyce), lub sterowania ręcznego. Komunikacja między robotem, a urządzeniem sterującym opiera się na modułach bluetooth.



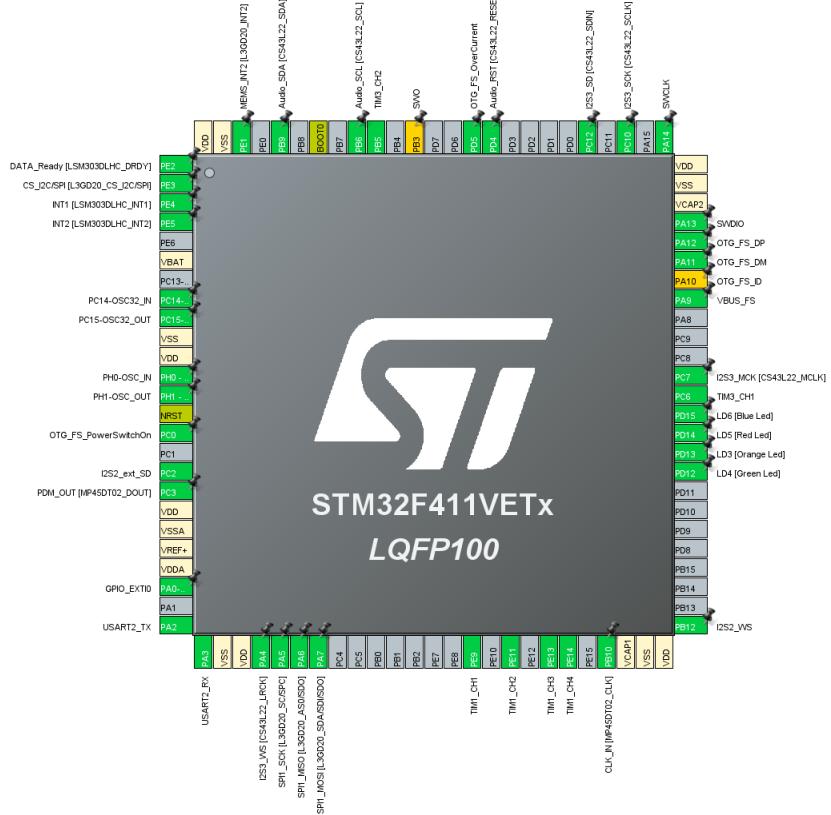
Rysunek 1: Schemat manipulatora

1.1 Zakres prac

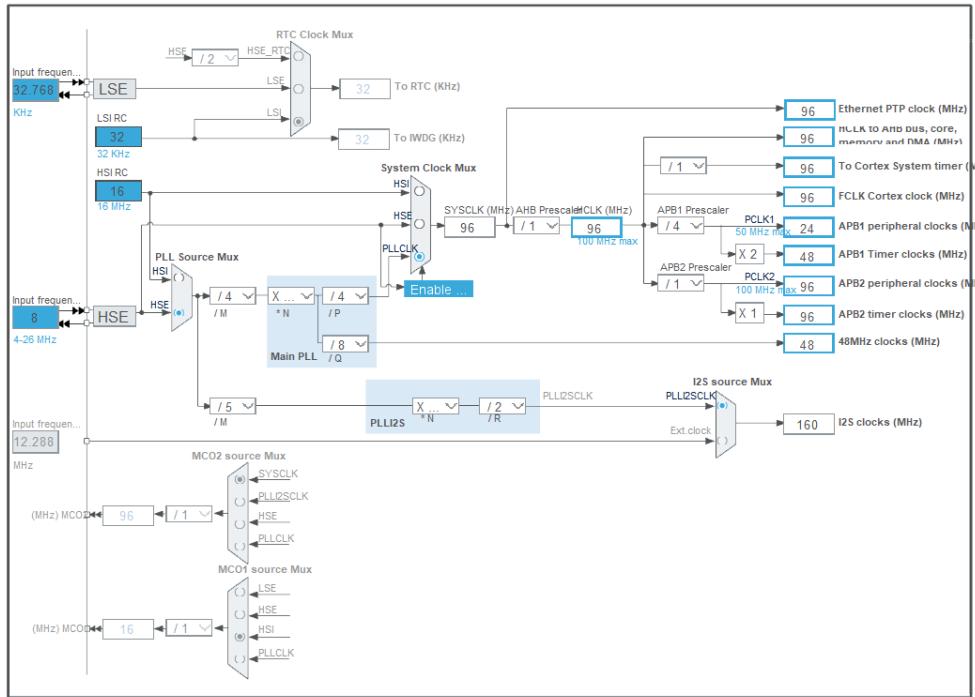
- Zapoznanie się z dokumentacją związaną z użytym oprogramowaniem oraz sprzętem.
- Utworzenie modelu 3D w Fusion 360/Solidworks.
- Skonstruowanie szkieletu z wydrukowanych na drukarce 3D części.
- Wstępna konfiguracja oprogramowania.
- Zapewnienie komunikacji urządzeń, wykorzystując bibliotekę związaną z bluetooth.
- Zapewnienie ruchów manipulatora, wykorzystując bibliotekę serv.
- Pierwsze testy, sprawdzające czy faktycznie jest nawiązane połączenie między urządzeniami.
- Stworzenie aplikacji umożliwiającej ręczne sterowanie.
- Testowanie aplikacji.
- Wyznaczenie odwrotnej kinematyki manipulatora.
- Utworzenie drugiego trybu w aplikacji.
- Ostateczne testowanie produktu.

2 Konfiguracja mikrokontrolera

W projekcie została wykorzystana płytka rozwojowa STM32F411E-DISCO Discovery. Poniżej znajduje się konfiguracja pinów wraz z ich oznaczeniami



Rysunek 2: Konfiguracja pinów



Rysunek 3: Konfiguracja zegarów

2.1 Konfiguracja pinów

PIN	Tryb pracy	Funkcja/etykieta
PB5	TIM3_CH2	PWM serwo chwytyak
PC6	TIM3_CH1	PWM serwo nadgarstek
PE9	TIM1_CH1	PWM serwo podstawa
PE11	TIM1_CH2	PWM serwo kolanko
PE13	TIM1_CH3	PWM serwo łokieć
PE14	TIM1_CH4	PWM serwo przedramię
PA2	USART2_TX	Bluetooth
PA3	USART2_RX	Bluetooth

Tabela 1: Konfiguracja pinów

2.2 TIM1 oraz TIM3

Za pomocą timera jest generowany sygnał PWM, dzięki któremu mamy możliwość sterowania serwami. Ich pozycja jest ściśle związana z wypełnieniem sygnału. Częstotliwość zegara została ustawiona na 50 Hz, zgodnie z wymogami urządzeń.

Parametr	Wartość
Prescaler	960-1
Counter Mode	Up
Counter Period	1000-1
Internal Clock	No Division
auto-reload preload	Enable
PWM Generation Channel 1	
Mode	PWM mode 1
Pulse	0
Output Compare Preload	Enable
Fast Mode	Disable
CH Polarity	High
PWM Generation Channel 2	
Mode	PWM mode 1
Pulse	0
Output Compare Preload	Enable
Fast Mode	Disable
CH Polarity	High
PWM Generation Channel 3	
Mode	PWM mode 1
Pulse	0
Output Compare Preload	Enable
Fast Mode	Disable
CH Polarity	High
PWM Generation Channel 4	
Mode	PWM mode 1
Pulse	0
Output Compare Preload	Enable
Fast Mode	Disable
CH Polarity	High

Tabela 2: Konfiguracja TIM1

Parametr	Wartość
Prescaler	960-1
Counter Mode	Up
Counter Period	1000-1
Internal Clock	No Division
auto-reload preload	Enable
PWM Generation Channel 1	
Mode	PWM mode 1
Pulse	0
Output Compare Preload	Enable
Fast Mode	Disable
CH Polarity	High
PWM Generation Channel 2	
Mode	PWM mode 1
Pulse	0
Output Compare Preload	Enable
Fast Mode	Disable
CH Polarity	High

Tabela 3: Konfiguracja TIM3

2.3 UART

Za pomocą uarta mamy możliwość komunikowania się pomiędzy robotem, a telefonem/komputerem, wykorzystując przy tym komunikator bluetooth.

Parametr	Wartość
Baud Rate	38400 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1
Data Direction	Receive and Transmit
Over Sampling	16 samples

Tabela 4: Konfiguracja pinów

3 Urządzenia peryferyjne

3.1 Serwomechanizm MicroServo9g-SG90

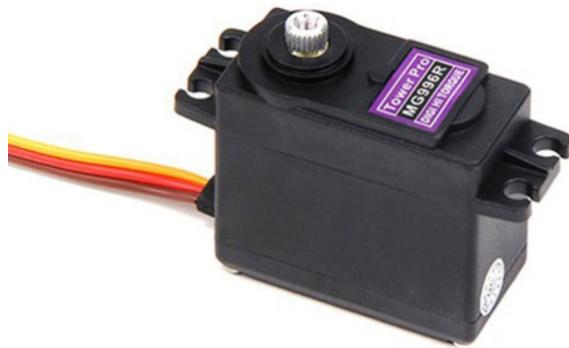
Ten typ serwomechanizmu jest odpowiedzialny za poruszanie tzw. przedramieniem, nadgarstkiem oraz chwytkiem mojego manipulatora. Niestety podczas prac związań z nim nastąpiły liczne komplikacje, gdyż część zamówionych serw niestety nie działała.



Rysunek 4: Serwomechanizm MicroServo9g-SG90

3.2 Serwomechanizm MG996R

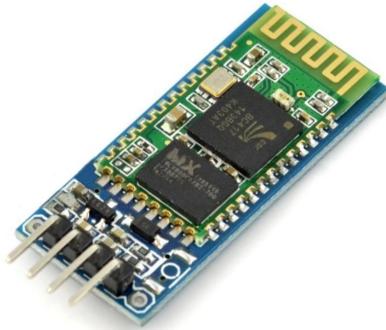
Tego typu serwo odpowiada za poruszanie podstawą, kolankiem oraz łożkiem robota.



Rysunek 5: Serwomechanizm MG996R

3.3 Moduł bluetooth HC-06

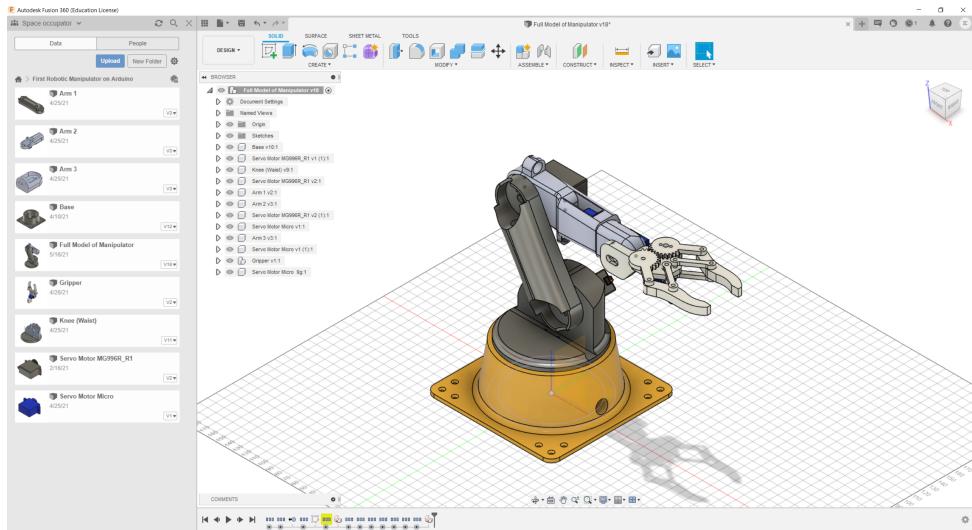
W celu zapewnienia komunikacji została wykorzystany moduł bluetooth HC-06, który umożliwia nam pracę w dwóch trybach: Master oraz Slave. Wykorzystując konwerter USB-USART oraz terminal portu szeregowego, komunikator miał nadaną nową nazwę, BR, oraz tryb jakim był SLAVE. Niestety w tej części prac również napotkałem szereg problemów. Pierwszym z nich był zły moduł jaki otrzymał ze sklepu, który pracował jedynie w jednym trybie. Drugim problemem było połączenie komunikatora z komputerem tak aby móc korzystać z portu szeregowego. Naszczęście finalnie udało się osiągnąć zamierzone cele.



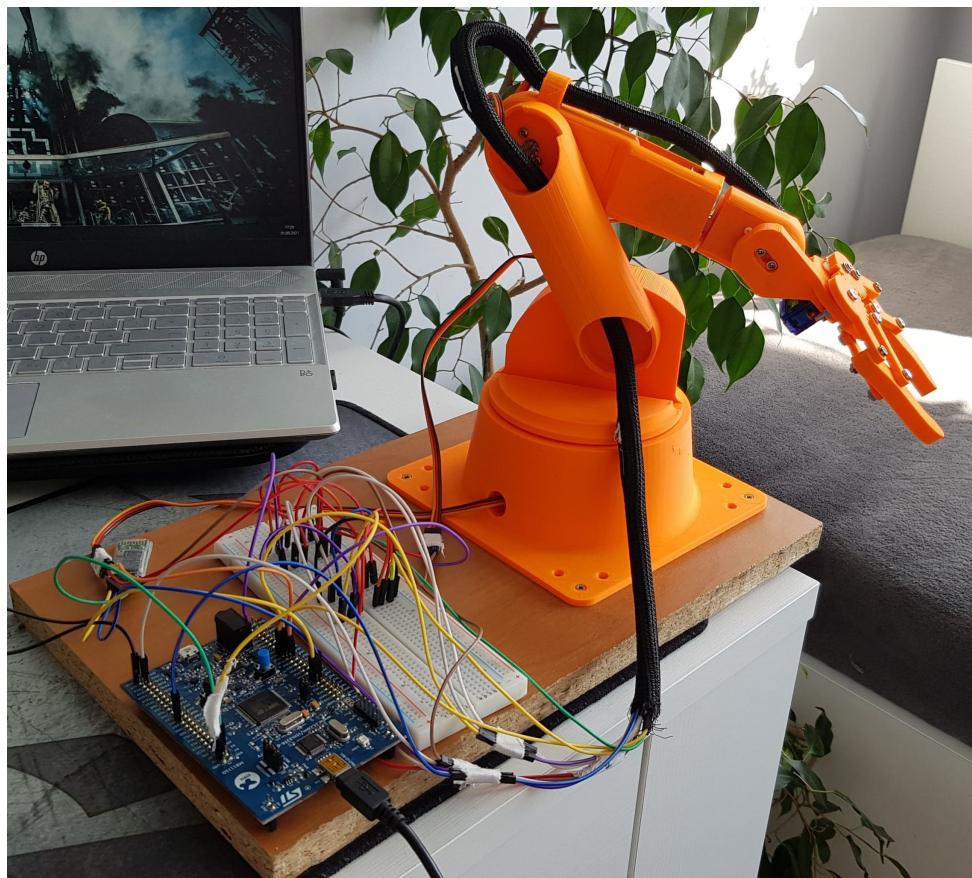
Rysunek 6: Moduł bluetooth HC-06

4 Konstrukcja mechaniczna

Na podstawie zdjęć w internecie, wszystkie części manipulatora zostały zaprojektowane w środowisku Autodesk Fusion360 oraz SolidWorks. Następnie pliki przesłano do drukarki 3D, która to w ciągu 70 h wydrukowała elementy. Kolejnym krokiem było złożenie konstrukcji wraz z serwami oraz przymocowanie jej do stabilnego podłoża jakim jest deska.



Rysunek 7: Projekt manipulatora



Rysunek 8: Wygląd fizyczny

5 Kinematyka manipulatora

Jako że w robotach nie mamy bezpośredniego pomiaru położenia końcówki manipulatora, pomiary realizowane są w określonych złączach łańcucha kinematycznego. W tym celu używamy właśnie kinematyki robota. Pierwsza z nich to kinematyka prosta, która to przekształca zmienne wewnętrzne manipulatora w zewnętrzne. Druga z nich to kinematyka odwrotna, realizująca zadanie odwrotne do kinematyki prostej.

5.1 Kinematyka prosta

	q_i	α_i	a_i	d_i
1	q_1	$-\pi/2$	0	d_1
2	q_2	0	a_2	0
3	q_3	0	a_3	0
4	q_4	$\pi/2$	0	0
5	q_5	0	0	d_5

Rysunek 9: Tabela Denavita-Hartenberga

$$A_o^1 = \begin{bmatrix} c_1 & 0 & -s_1 & 0 \\ s_1 & 0 & c_1 & 0 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_1^2 = \begin{bmatrix} c_2 & -s_2 & 0 & a_2 q_1 \\ s_2 & c_2 & 0 & a_2 s_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_2^3 = \begin{bmatrix} c_3 & -s_3 & 0 & a_3 q_1 \\ s_3 & c_3 & 0 & a_3 s_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3^4 = \begin{bmatrix} c_4 & 0 & -s_4 & 0 \\ s_4 & 0 & c_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_4^5 = \begin{bmatrix} c_5 & -s_5 & 0 & 0 \\ s_5 & c_5 & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rysunek 10: Macierze transformacji

$$A_o^5 = A_o^1 \cdot A_1^2 \cdot A_2^3 \cdot A_3^4 \cdot A_4^5 = \begin{bmatrix} x & y & z & 1 \\ c_1 \cdot c_{234} \cdot c_5 + s_1 \cdot s_5 & -c_1 \cdot c_{234} \cdot s_5 + s_1 \cdot c_5 & -c_1 \cdot s_{234} & -c_1 \cdot s_{234} \cdot d_5 + c_1 \cdot c_{23} \cdot a_3 + c_1 \cdot c_2 \cdot a_2 \\ c_1 \cdot c_{234} \cdot c_5 - s_1 \cdot s_5 & -s_1 \cdot c_{234} \cdot c_5 - s_1 \cdot s_5 & -s_1 \cdot s_{234} & -s_1 \cdot s_{234} \cdot d_5 + c_{12} \cdot s_1 \cdot a_3 + c_1 \cdot s_1 \cdot a_2 \\ -s_{234} \cdot c_5 & -s_{234} \cdot c_5 & -c_{234} & d_5 - a_2 \cdot s_1 - a_3 \cdot s_1 - d_5 \cdot c_{234} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rysunek 11: Kinematyka prosta

5.2 Kinematyka odwrotna

$$q_1 = \operatorname{atan}^{-1} \left(\frac{-s_4 s_{23} d_5 + c_{12} s_4 a_3 + c_4 c_5 a_2}{-c_4 s_{23} d_5 + c_{12} c_5 a_3 + c_4 c_5 a_2} \right) \quad \rightarrow -c_4 s_{23} d_5 + c_{12} c_5 a_3 + c_4 c_5 a_2 > 0 \quad \vee \quad -\frac{\pi}{2} < q_1 < \frac{\pi}{2}$$

$$q_2 = \arctan 2 \left((a \cdot a_2 + a \cdot a_3 \cdot c_3) - s_3 \cdot a_3 \cdot b, \quad s_3 \cdot a \cdot a_3 + c_3 a_3 b + a_2 b \right)$$

$$q_3 = \arccos \frac{a^2 + b^2 - a_2^2 - a_3^2}{2 a_2 \cdot a_3}$$

$$b = (-c_4 s_{23} d_5 + c_{12} c_5 a_3 + c_4 c_5 a_2) c_1 + \\ (-s_4 s_{23} d_5 + c_{12} s_4 a_3 + c_4 s_4 a_2) s_1 + d_5 s_{23} s_1$$

$$q_4 = q_{23s} - q_2 - q_3$$

$$q_5 = c_{23s} \cdot q_2 - 2 \arctan \left(c_4 \cdot c_{13s} \cdot c_5 - s_4 s_5, \quad c_4 \cdot c_{23s} \cdot c_5 + s_4 \cdot s_5 \right)$$

Rysunek 12: Kinematyka odwrotna

6 Program

Pierwszym programem jaki napisałem było świecenie diody, gdyż dzięki niemu mogłem sprawdzić, czy faktycznie moduł bluetooth działa poprawnie.

```

1 void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
2 {
3     if (huart->Instance == USART2)
4     {
5         if (znak=='1')
6         {
7             HAL_GPIO_WritePin(LD6_GPIO_Port, LD6_Pin, GPIO_PIN_SET);
8             sprintf(info_zwrotna, "on\n");
9             dlugosc_info=2;
10        }
11        else if (znak=='2')
12        {
13            HAL_GPIO_WritePin(LD6_GPIO_Port, LD6_Pin, GPIO_PIN_RESET);
14            sprintf(info_zwrotna, "off\n");
15            dlugosc_info=3;
16        }
17        HAL_UART_Receive_IT(&huart2, &znak, 1);
18        HAL_UART_Transmit_IT(&huart2, &info_zwrotna, dlugosc_info);
19    }
20 }
```

Kolejnym krokiem było napisanie kolejnego testowego programu. Jednakże w tym przypadku jego zadaniem było przetestowanie sprawności serwomechanizmów.

```

1 int i=25;
2 while (i<125)
3 {
4     htim1.Instance->CCR1=i;
5     i++;
6     HAL_Delay(100);
7 }
8 HAL_Delay(1000);
```

Po sprawdzeniu urządzeń nastął moment na pisanie kodu dotyczącego ręcznego sterowania manipulatorem. Na wstępnie zadeklarowałam flagi, które odpowiedzialne są za dany ruch określonego serwa. Niestety musiałam stworzyć dla każdego serwa i jego ruchu osobną flagę, ponieważ wymagał tego program w którym pisałam aplikację.

```

1 void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
2 {
3     if(huart->Instance == USART2)
4     {
5         //tryby servo baza
6         if(znak=='a')
7         {
8             flag=1;
9         }
10        else if(znak=='b')
11        {
12            flag=2;
13        }
14        else if(znak=='c')
15        {
16            flag=3;
17        }
18        else if(znak=='d')
19        {
20            flag=4;
21        }

```

Kolejnym etapem było zadeklarowanie UARTA oraz TIMERÓW.

```

1 HAL_UART_Receive_IT(&huart2 , &znam , 1);
2 HAL_TIM_PWM_Start(&htim1 , TIM_CHANNEL_1);
3 HAL_TIM_PWM_Start(&htim1 , TIM_CHANNEL_2);
4 HAL_TIM_PWM_Start(&htim1 , TIM_CHANNEL_3);
5 HAL_TIM_PWM_Start(&htim1 , TIM_CHANNEL_4);
6 HAL_TIM_PWM_Start(&htim3 , TIM_CHANNEL_1);
7 HAL_TIM_PWM_Start(&htim3 , TIM_CHANNEL_2);

```

Następnie każda z flag miała zadeklarowany odpowiedni ruch.

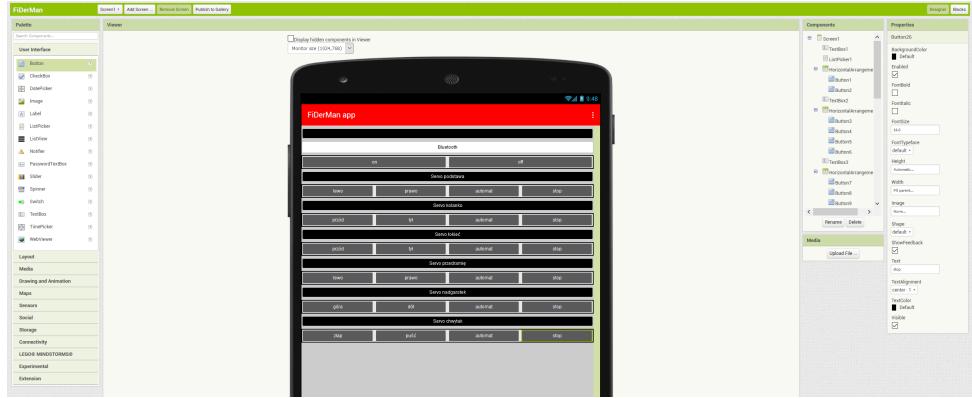
```

1 if( flag==1)//prawo
2 {
3     htim1.Instance->CCR1=75;
4     HAL_Delay(1000);
5 }
6 if( flag==2) //lewo
7 {
8     htim1.Instance->CCR1=125;
9     HAL_Delay(1000);
10}
11 if( flag==3)//auto
12 {
13     htim1.Instance->CCR1=75;
14     HAL_Delay(1000);
15
16     htim1.Instance->CCR1=125;
17     HAL_Delay(1000);
18 }
19 if( flag==4)//stop
20 {
21     htim1.Instance->CCR1=25;
22     HAL_Delay(1000);
23 }

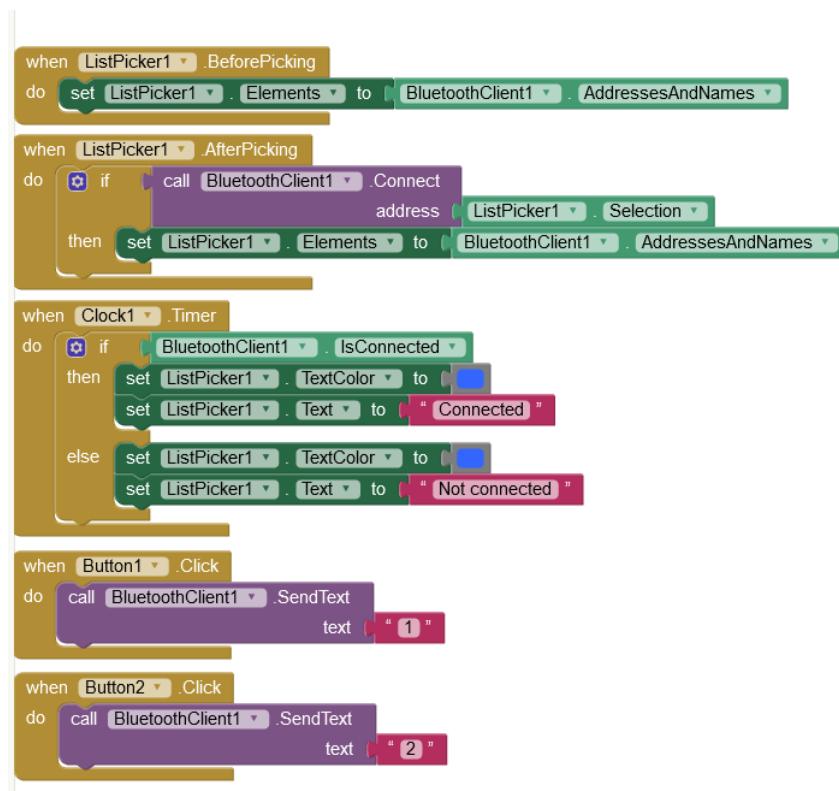
```

7 Aplikacja

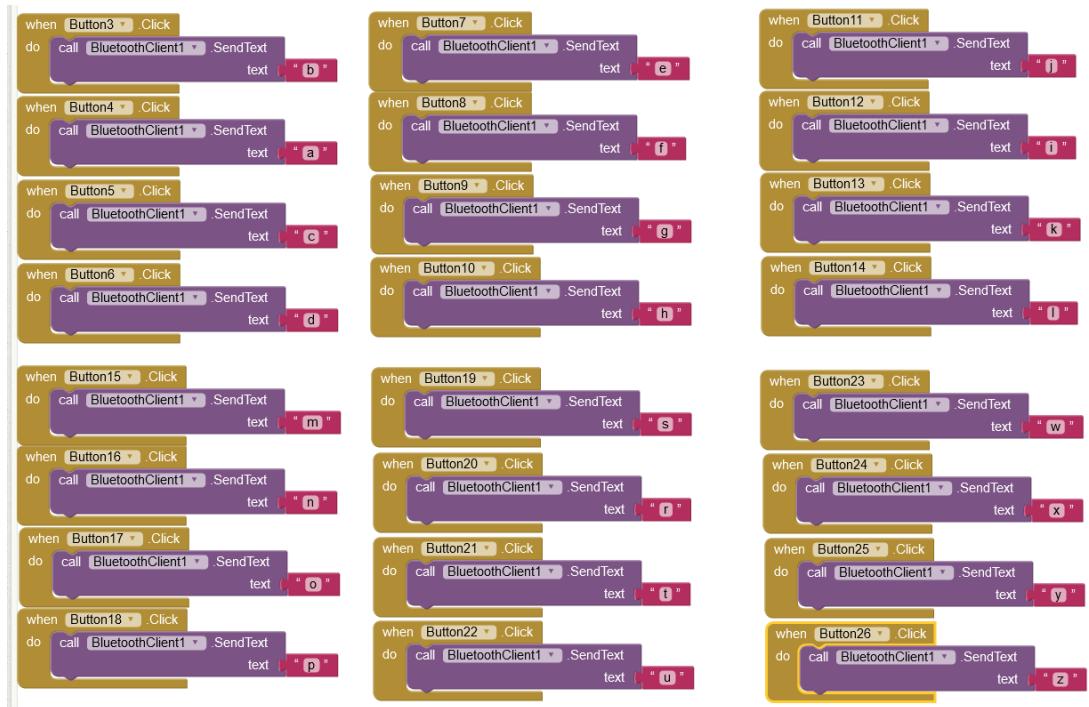
Aplikacja została napisana w programie MIT App Inventor, który to umożliwia napisanie wieloplatformówki. Język programowania jest typu drabinkowego. W celu uruchomienia aplikacji wystarczy pobrać oprogramowanie na telefon i zeskanować kod QR. Niestety od jakiegoś czasu aplikacja ta nie jest wspierana przez google, przez co zdarza się, że nie działa ona zawsze poprawnie na wszystkich telefonach.



Rysunek 13: Projekt aplikacji



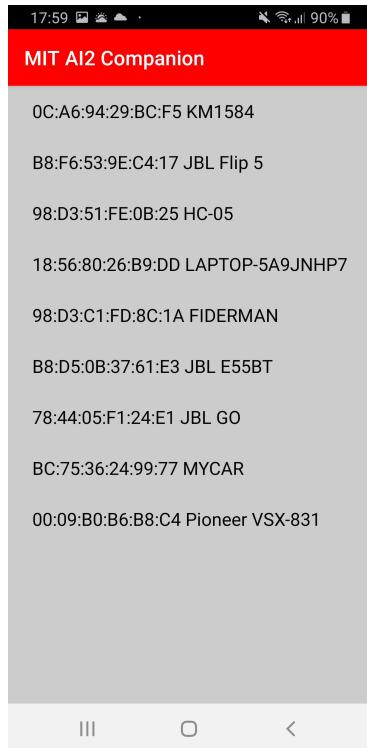
Rysunek 14: Program umożliwiający komunikacje bluetooth



Rysunek 15: Program umożliwiający obsługę przycisków



Rysunek 16: Wygląd głównego okna aplikacji



Rysunek 17: Wygląd okienka od bluetooth

8 Podsumowanie

W ramach projektu udało się stworzyć oraz zaprogramować manipulator typu antropomorficznego o 5 stopniach swobody, w celu przenoszenia niewielkich rzeczy, w obrębie danego pola. Na model składają się części wydrukowane na drukarce 3D, serwa, moduł bluetooth, zasilanie oraz płytka STM32F411E-Disco-Discovery. Do realizacji zadania wykorzystano wiele różnych peryferiów np. timery, usart.

Moim pierwszym krokiem było, zapewnienie komunikacji oraz pracy serw. Finalnie po licznych problemach związanych z obiema rzeczami udało się zarówno skonfigurować moduł bluetooth, jak i uruchomić i przetestować wszystkie serwomechanizmy.

Jeżeli chodzi o konstrukcję mechaniczną, to została ona zaprojektowana w środowisku Autodesk Fusion360 oraz SolidWorks. Kolejnym krokiem było złożenie konstrukcji wraz z serwami oraz przymocowanie jej do stabilnego podłoża jakim jest deska.

Dodatkowo manipulator zapewnia tryb ręczny sterowania za pomocą wieloplatformowej aplikacji. Niestety finalnie nie udało się zaimplementować automatycznego trybu na podstawie obliczonej odwrotnej kinematyki. Powodem były trudności związane z implementacją sliderów w gui aplikacji.

Literatura

Literatura

- [1] M.Galewski. STM32. Aplikacje i ćwiczenia w języku C z biblioteką HAL. BTC, 2019.
- [2] W.Domski. Sterowniki robotów, Laboratorium–Wprowadzenie, Wykorzystanie narzędzi STM32CubeMX oraz SW4STM32 do budowy programu mrugającej diody z obsługą przycisku. <http://edu.domski.pl>, Mar.2017
- [3] <https://www.autodesk.com/products/fusion-360/learn-support>
- [4] K. Tchoń and R. Muszyński. Robotyka. Wrocław 2018
- [5] K. Tchoń, A. Mazur, R. Hossa, I. Dulęba and R. Muszyński. Manipulatory i roboty mobilne - Modele planowania ruchu, sterowanie. Akademicka Oficyna Wydawnicza PLJ. Warszawa, 2000