

PROJEKT

ROBOTY MOBILNE 1

---

## Założenia projektowe

# Półautonomiczna platforma mobilna z manipulatorem RRT

---

*Skład grupy:*

Adam KUBIAK, 29480  
Patrick ROSSOL, 249470

*Termin:* wt TN 19

*Prowadzący:*

mgr inż. Arkadiusz MIELCZAREK

# **Spis treści**

<b>1 Opis projektu</b>	<b>2</b>
<b>2 Harmonogram pracy</b>	<b>2</b>
<b>3 Konfiguracja mikrokontrolera</b>	<b>3</b>
3.1 Konfiguracja jednostki sterującej . . . . .	3
3.1.1 Konfiguracja pinów . . . . .	5
3.1.2 TIM2 . . . . .	5
3.1.3 DMA . . . . .	6
3.2 Konfiguracja pilota . . . . .	6
3.2.1 Konfiguracja pinów . . . . .	8
3.2.2 ADC1 . . . . .	8
3.2.3 DMA . . . . .	9
<b>4 Urządzenia zewnętrzne</b>	<b>9</b>
4.1 Joystick analogowy . . . . .	9
4.2 Klawiatura - matryca 16 x tact switch . . . . .	10
4.3 Silnik z przekładnią SJ01 + enkoder . . . . .	10
4.4 Moduł sterownika silnika DC L298N . . . . .	11
4.5 Serwomechanizm - TowerPro SG-90 - micro . . . . .	11
4.6 Serwomechanizm - Tower Pro SG-5010 . . . . .	12
4.7 Wyświetlacz Waveshare OLED 0,95 . . . . .	12
4.8 Moduł Bluetooth HC-05 . . . . .	13
<b>5 Projekt elektroniki</b>	<b>13</b>
<b>6 Konstrukcja mechaniczna</b>	<b>16</b>
<b>7 Kinematyka manipulatora</b>	<b>17</b>
7.1 Kinematyka prosta . . . . .	17
7.2 Odwrotna kinematyka . . . . .	18
<b>8 Opis działania programu</b>	<b>19</b>
8.1 Wybór trybu sterowania . . . . .	19
8.2 Wysyłane ramki danych . . . . .	19
8.3 Interpretacja zadanych kątów . . . . .	19
8.4 Wykorzystanie odwrotnej kinematyki i wykonywanie ruchu . . . . .	20
8.5 Zapisywanie sekwencji wewnętrznej pamięci FLASH . . . . .	21
8.6 Rysowanie interfejsu użytkownika . . . . .	22
8.7 Odometria . . . . .	23
<b>9 Podsumowanie</b>	<b>24</b>

# 1 Opis projektu

Założeniem projektu jest budowa oraz oprogramowanie półautonomicznej platformy mobilnej z manipulatorem RRT i zaprojektowanie interfejsu dla operatora.

Projekt będzie opierał się na następujących założeniach:

- Konstrukcja samego manipulatora będzie wykonana z gotowych aluminiowych elementów wykonanych pod serwomechanizmy, takich jak uchwyty, mocowania, orczyki i wydrukowanego elementu, o której jest odpowiedzialny za ruch translacyjny manipulatora. Sama platforma będzie złożona ze specjalnie dla niej zaprojektowanych płyt z pleksi.
- Jako punkt pracy manipulatora wykorzystano mały elektromagnes, odpowiedzialny za przenoszenie małych metalowych elementów.
- Do sterowania platformą zostaną wykorzystane joysticki, które będą umieszczone na pilocie operatorskim. Natomiast sam manipulator będzie sterowany za pomocą odwrotnej kinematyki lub wcześniej wymienionych joysticków.
- Pilot operatorski, dzięki wyświetlaczowi OLED, daje możliwość oglądu do najważniejszych informacji tj. informacja zwrotna na temat położenia platformy względem jej zerowego punktu położenia, punktu pracy manipulatora czy kąt rotacji każdego z przegubów. Jednak jego główną rolą jest sterowanie położeniem samej platformy i ramienia.
- Współrzędne ustawione przez użytkownika będzie można zapisywać do sekwencji, a następnie ją odtwarzać. Wszystkie sekwencje będą zapisywane do pamięci FLASH lub EEPROM.
- Dane do współrzędnych będą wprowadzane za pomocą klawiatury matrycowej lub przy pomocy joysticków.
- Do komunikacji między płytą odpowiedzialną za obsługę platformy oraz manipulatora i tą na pilocie wykorzystano moduł Bluetooth 4.0

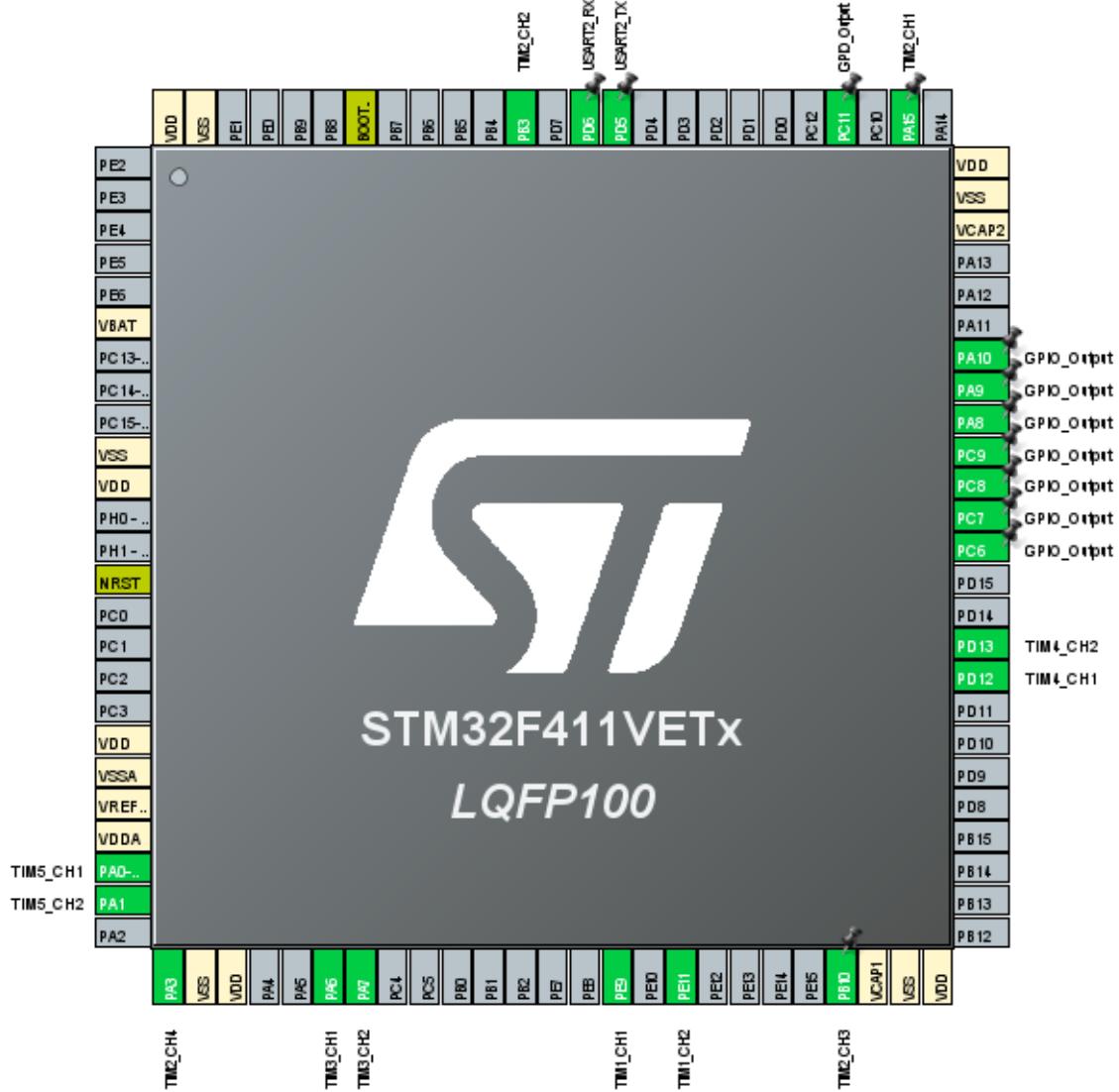
# 2 Harmonogram pracy

- 20.03 - Konfiguracja peryferiów w programie CubeMx
- 22.03 - Wykonanie konstrukcji manipulatora RRT
- 30.03 - Wykonanie konstrukcji platformy oraz Wyliczenie prostej i odwrotnej kinematyki manipulatora RRT
- 05.04 - Obsługa serwomechanizmów oraz silników DC z enkoderami
- 15.04 - Opracowanie sterowania serwomechanizmami dzięki wyliczonej kinematyce
- 20.04 - Wykonanie projektu pilota operatorskiego
- 25.04 - Implementacja obsługi wyświetlacza OLED za pomocą komunikacji SPI
- 30.04 - Implementacja obsługi joysticka i przełożenie jej na ruch servomechanizmów i silników DC
- 05.05 - Opracowanie sposobu przechowywania danych na zewnętrznej pamięci FLASH lub EEPROM
- 08.05 - Opracowanie komunikacji Bluetooth pomiędzy dwoma płytami STM32
- 15.05 - Stworzenie konceptu menu użytkownika wykorzystującego matryce przycisków do jego obsługi
- 25.05 - Zaprojektowanie regulatora PID
- 30.05 - Implementacja odometrii do kontroli położenia w układzie
- 02.06 - Finalna konfiguracja peryferiów w programie CubeMX

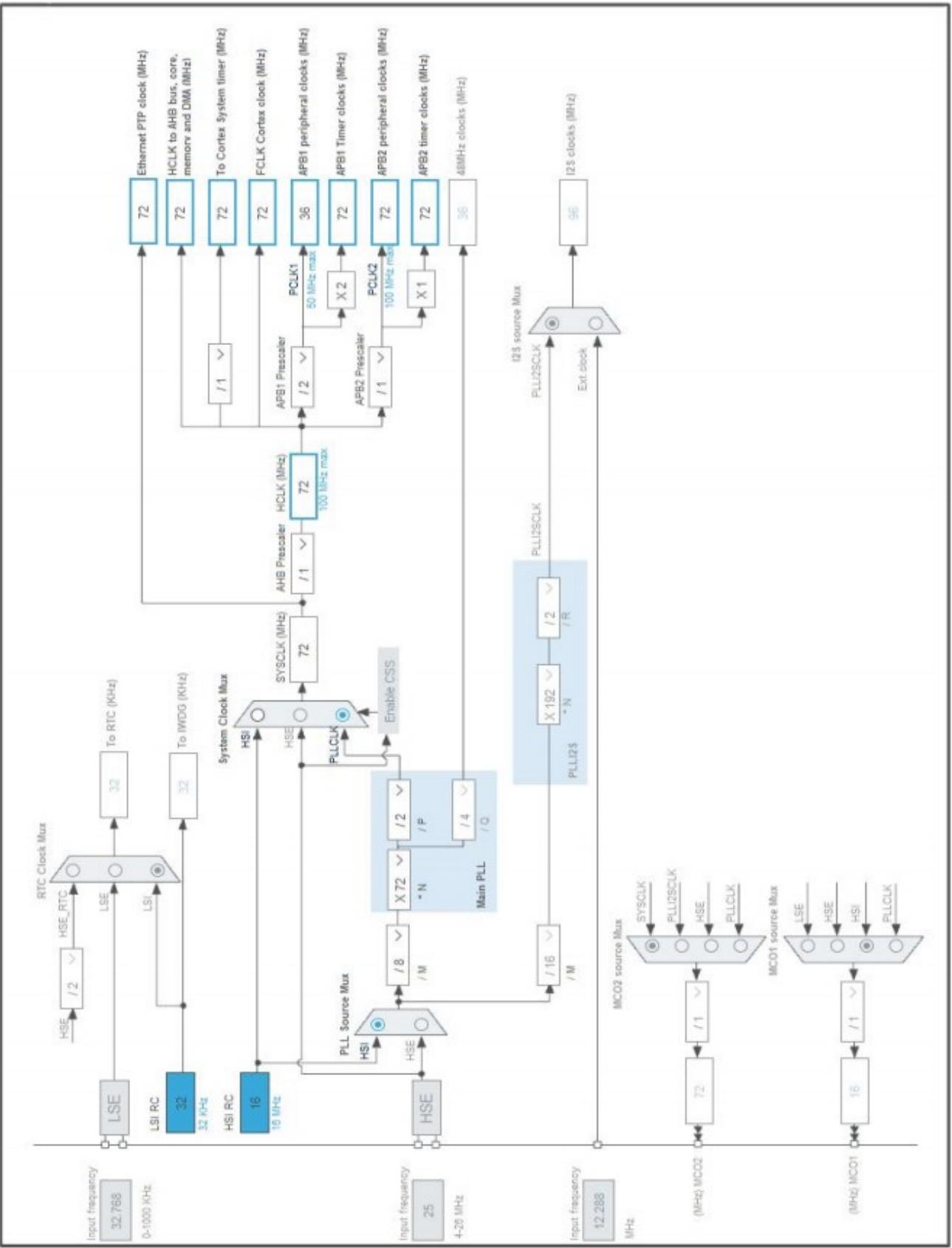
### 3 Konfiguracja mikrokontrolera

W projekcie zostały wykorzystane dwie płytki rozwojowe STM32F411VET6 Discovery. Poniżej przedstawiono konfigurację obu mikrokontrolerów, gdzie płytka numer 1 jest jednostką sterującą natomiast płytka numer 2 jest bezprzewodowym pilotem, z którego będą wysyłane komendy do jednostki sterującej.

#### 3.1 Konfiguracja jednostki sterującej



Rysunek 1: Konfiguracja wyjść mikrokontrolera **numer 1** w programie STM32CubeMX



Rysunek 2: Konfiguracja zegarów mikrokontrolera

### 3.1.1 Konfiguracja pinów

Numer pinu	PIN	Tryb pracy	Funkcja/etykieta
77	PA1	TIM2_CH1	PWM servo1
89	PA2	TIM2_CH2	PWM servo2
47	PA3	TIM2_CH3	PWM servo3
23 24	PA0 PA1	TIM5	Enkoder1
31 32	PA6 PA7	TIM3	Enkoder2
59 60	PD12 PD13	TIM4	Enkoder3
40 42	PA15 PB3	TIM1	Enkoder4
26	PA3	TIM2_CH4	Prędkość silników
	PA8-PA10	GPIO_OUTPUT	Sterowanie kierunkiem
	PC6-PC9	GPIO_OUTPUT	Sterowanie kierunkiem

Tabela 1: Konfiguracja pinów jednostki sterującej

### 3.1.2 TIM2

Timer wykorzystywany jest do generowania sygnałów PWM, które sterują pozycjami serwomechanizmów w przegubach rotacyjnych. Pozycja zależy od wypełnienia sygnału PWM. Zegar ustawiono tak aby pracował z częstotliwością 50HZ zgodnie z wymogami sterowania zastosowanych serwomechanizmów **Tower Pro SG-5010** i **Tower Pro SG90**. Konfiguracja przedstawiona w tabeli 2.

Parametr	Wartość
Prescaler	23
Counter Mode	Up
Counter Period	59999
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

Tabela 2: Konfiguracja TIM2

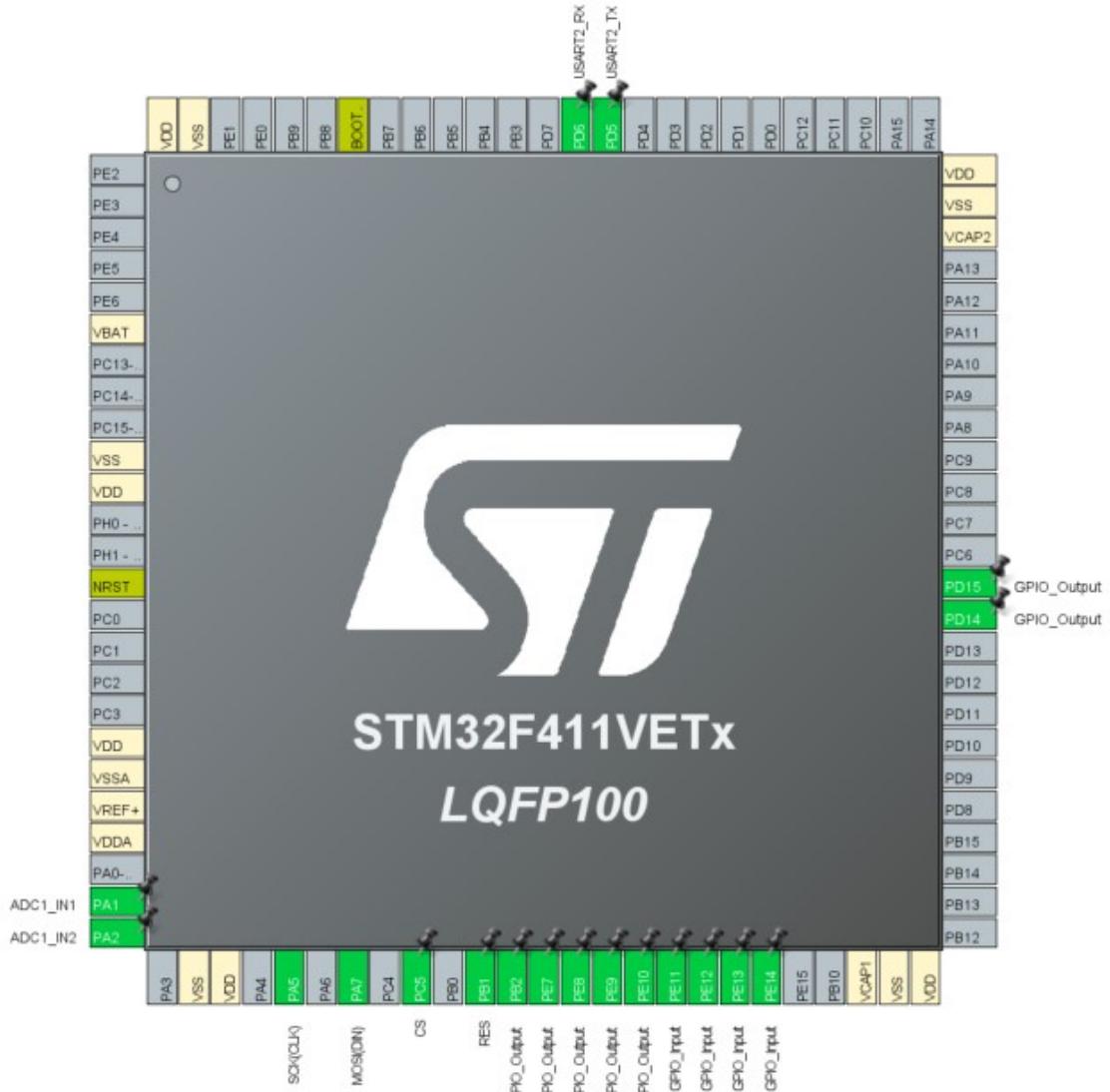
### 3.1.3 DMA

Zastosowanie DMA w komunikacji UART pozwala na bezpośrednie zapisanie otrzymanych danych do pamięci. 6.

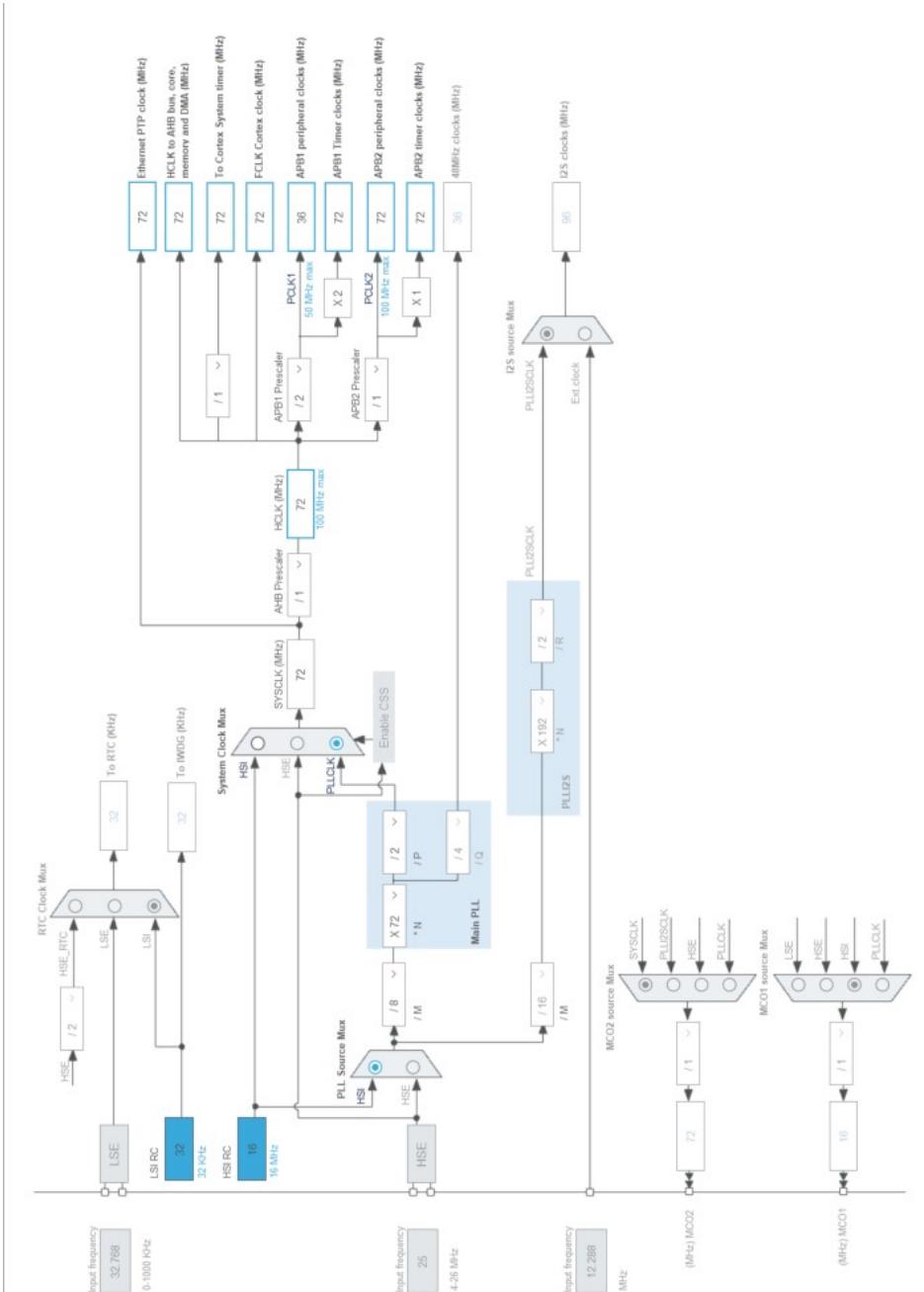
Parametr	Wartość
Mode	Normal
Use fifo	Disable
Peripheral Increment	Disable
Memory Increment	Enable
Peripheral Data Width	Byte
Memory Data Width	Byte

Tabela 3: Konfiguracja DMA dla UART

## 3.2 Konfiguracja pilota



Rysunek 3: Konfiguracja wyjść mikrokontrolera **numer 2** w programie STM32CubeMX



Rysunek 4: Konfiguracja zegarów mikrokontrolera

### 3.2.1 Konfiguracja pinów

Numer pinu	PIN	Tryb pracy	Funkcja/etykieta
29	PA4	ADC1_IN4	JoystickX IN4
30	PA5	ADC1_IN5	JoystickY IN5
32	PA7	MOSI	DIN
36	PB1	RES	RES
37	PB2	GPIO_Output	D/C
34	PC5	CS	CS
38	PE7	GPIO_Output	KeyMatrix Row1
39	PE8	GPIO_Output	KeyMatrix Row2
40	PE9	GPIO_Output	KeyMatrix Row3
41	PE10	GPIO_Output	KeyMatrix Row4
42	PE11	GPIO_Input	KeyMatrix Column1
43	PE12	GPIO_Input	KeyMatrix Column2
44	PE13	GPIO_Input	KeyMatrix Column3
45	PE14	GPIO_Input	KeyMatrix Column4
86	PD5	USART2_TX	HC-05 Receiver
87	PD6	USART2_RX	HC-05 Receiver

Tabela 4: Konfiguracja pinów pilota

### 3.2.2 ADC1

Peryferium wykorzystane do obsługi joysticków, które służą do ręcznego sterowania manipulatorem. Konfiguracja przedstawiona w tabeli 5.

Parametr	Wartość
<b>Mode</b>	Independent mode
<b>Clock Prescaler</b>	PLCK2 divided by 8
<b>Resolution</b>	12 bits
<b>Data Alignment</b>	Right alignment
<b>Scan conversion</b>	Enabled
<b>Continuous conversion</b>	Enabled
<b>Discontinuous Conversion Mode</b>	Disabled
<b>DMA Continuous Requests</b>	Enabled
<b>End Of Conversion Selection</b>	EOC flag at the end of single channel conversion
<b>Number Of Conversion</b>	2
<b>External Trigger Conversion Source</b>	Regular Conversion launched by software
<b>External Trigger Conversion Edge</b>	None
<b>Rank</b>	1
<b>Channel</b>	Channel 4
<b>Sampling Time</b>	480 Cycles
<b>Rank</b>	2
<b>Channel</b>	Channel 4
<b>Sampling Time</b>	480 Cycles

Tabela 5: Konfiguracja ADC1

### 3.2.3 DMA

Zastosowanie DMA pozwala na zapisanie wartości odczytu z ADC1 do zmiennej. Konfiguracja przedstawiona w tabeli 6.

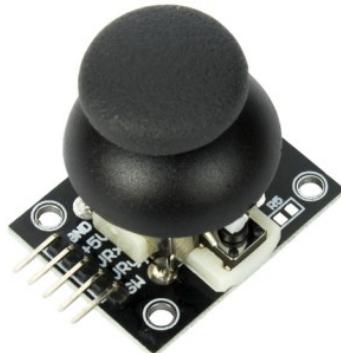
Parametr	Wartość
Mode	Circular
Use fifo	Disable
Peripheral Increment	Disable
Memory Increment	Enable
Peripheral Data Width	Half Word
Memory Data Width	Half Word

Tabela 6: Konfiguracja DMA

## 4 Urządzenia zewnętrzne

Urządzeniami zewnętrznymi odpowiedzialnymi za ruch manipulatora są serwomechanizmy sterowane sygnałem PWM, przy sterowaniu silników korzystamy z sterownika L298N i dołączonych do silników enkoderów. Elementami odpowiedzialnymi za interfejs użytkownika i zadawanie pozycji lub sekwencji są takie elementy jak matryca przycisków, joysticki i wyświetlacz OLED

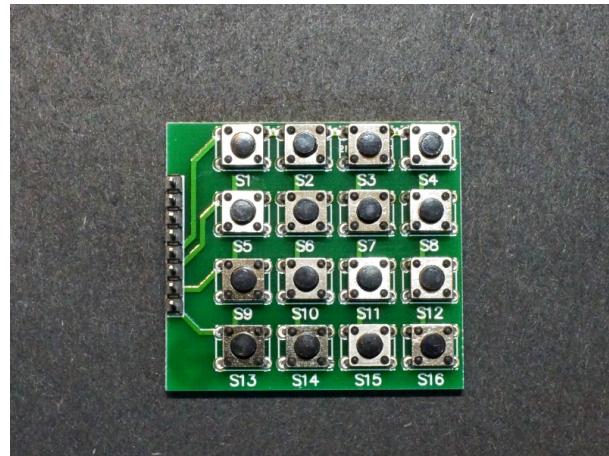
### 4.1 Joystick analogowy



Rysunek 5: Zdjęcie użytego joysticka

Po wyborze odpowiedniego trybu pracy możemy przy jego pomocy sterować ręcznie położeniem manipulatora w płaszczyźnie XY.

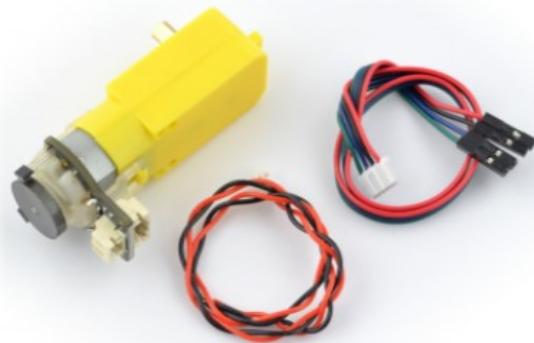
## 4.2 Klawiatura - matryca 16 x tact switch



Rysunek 6: Zdjęcie użytej klawiatury

Matryca z przyciskami pozwala na wybór trybu pracy manipulatora oraz na wprowadzanie współrzędnych, do których ma się przemieścić. Poniżej znajduje się rozpiska funkcji poszczególnych przycisków:

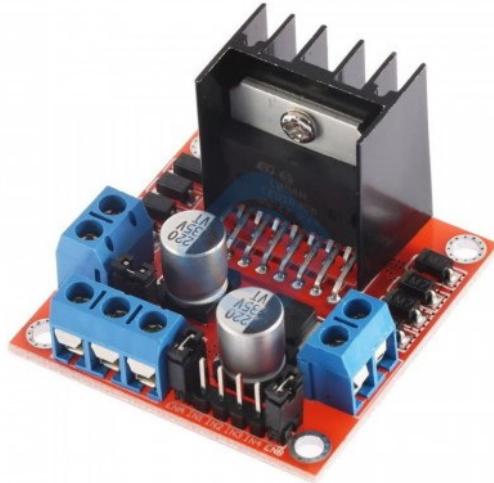
## 4.3 Silnik z przekładnią SJ01 + enkoder



Rysunek 7: Zdjęcie użytych silników DC z enkoderami

W projekcie zastosowane są silniki DC z przekładnią i enkoderami które służą za napędy kół robota mobilnego.

#### 4.4 Moduł sterownika silnika DC L298N



Rysunek 8: Zdjęcie użytych sterowników silników DC

W projekcie zastosowane są silniki DC z przekładnią i enkoderami które służą za napędy kół robota mobilnego.

#### 4.5 Serwomechanizm - TowerPro SG-90 - micro



Rysunek 9: Zdjęcie użytego serwonapędu

Micro serwonaapult odpowiada za sterowanie członem translacyjnym. Sam translator porusza się w kierunku prostopadłym do podłoża.

#### 4.6 Serwomechanizm - Tower Pro SG-5010



Rysunek 10: Zdjęcie użytego serwonapędu

Serwonapędy Tower Pro tworzą oba człony rotacyjne manipulatora.

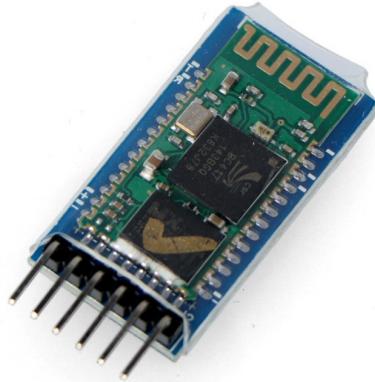
#### 4.7 Wyświetlacz Waveshare OLED 0,95



Rysunek 11: Zdjęcie użytego wyświetlacza OLED

Na jego ekranie będzie wyświetlany cały interfejs potrzebny do sterowania manipulatorem. Pozwala na pokazanie wybranego trybu, wprowadzonych współrzędnych, czy poziom naładowania pilota.

#### 4.8 Moduł Bluetooth HC-05

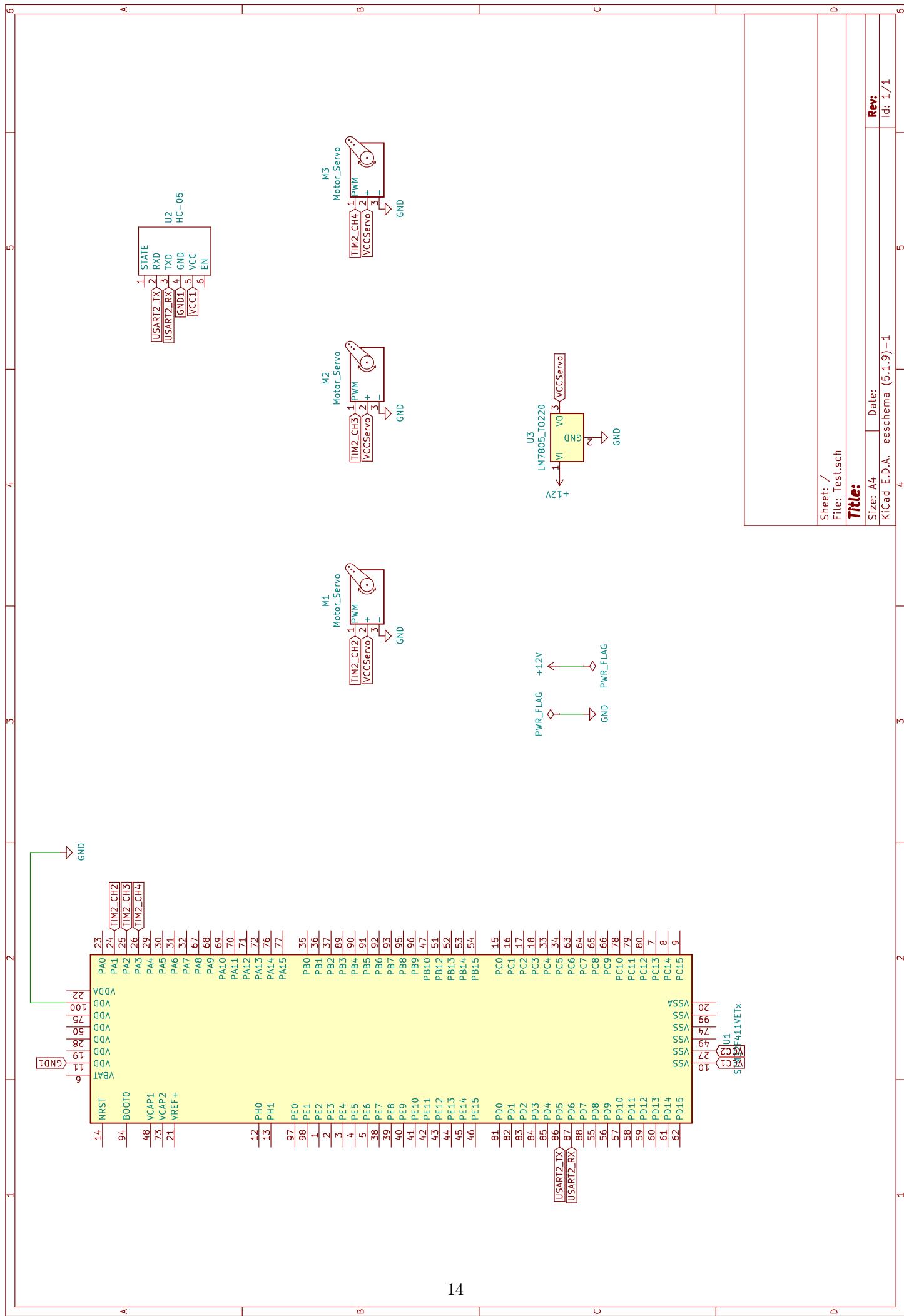


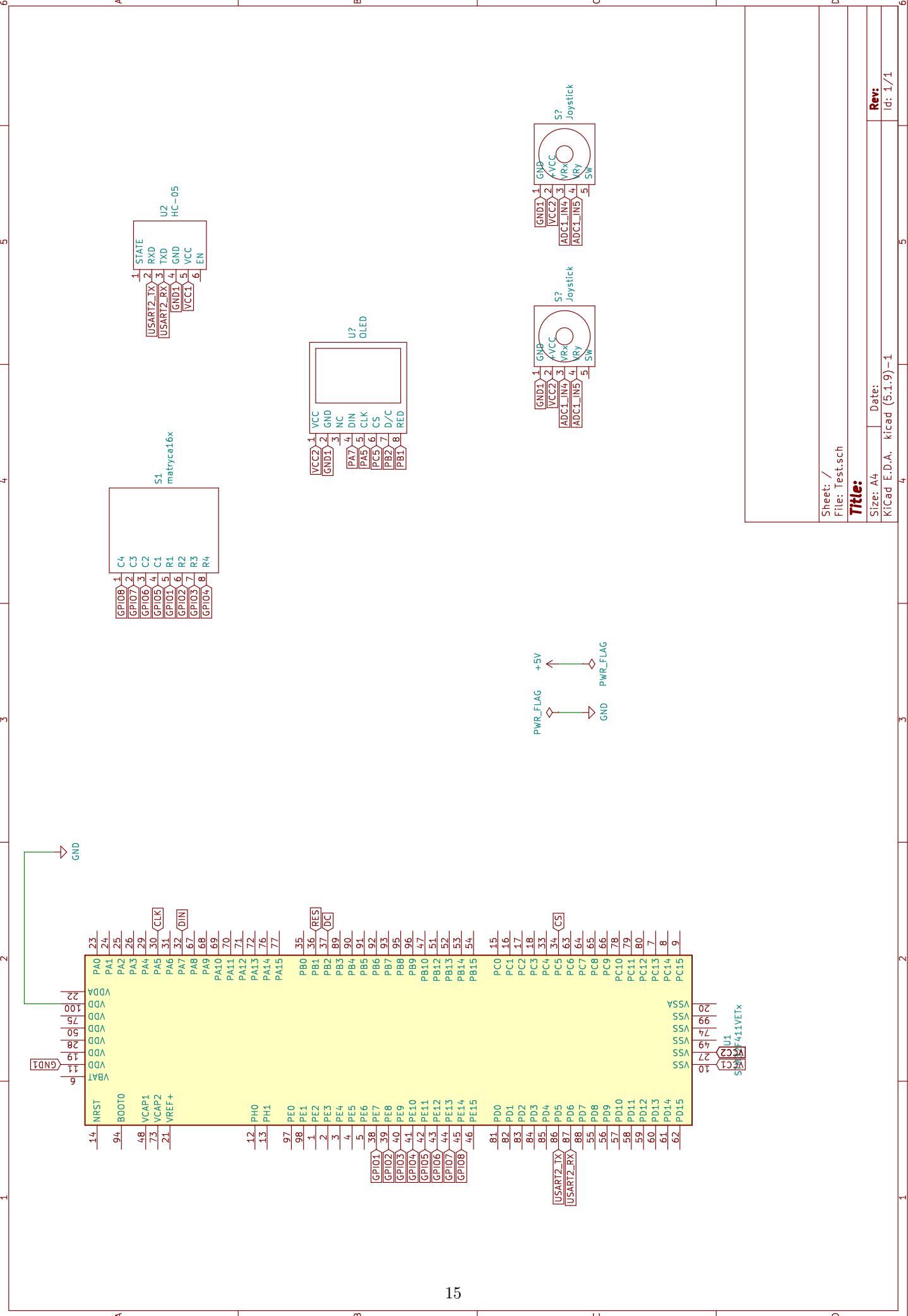
Rysunek 12: Zdjęcie użytego modułu Bluetooth

W projekcie zastosowane są dwa moduły - jeden w pilocie, drugi przy platformie z manipulatorem. Jego zasięg wynosi ok. 10m więc pozwala na wygodne sterowanie nawet z drugiego końca pokoju. Płytki komunikują się ze sobą po protokole UART.

### 5 Projekt elektroniki

Płytką rozwojową STM32F411E-Discovery oraz serwonapędy pracują pod napięciem 5V. Więc do zasilania całego układu zosatł wykorzystany liniowy regulator napięcia LM7805, aby zredukować napięcie wejściowe z 12V z zasilacza na 5V. Poniżej znajdują się schematy elektroniczne jednostki sterującej oraz pilota.



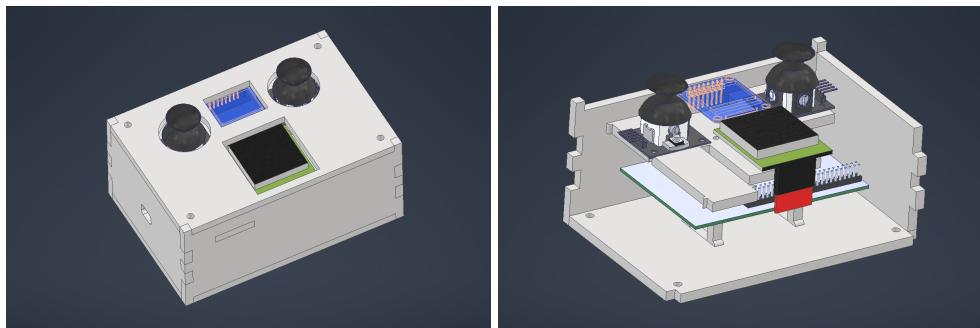


## 6 Konstrukcja mechaniczna

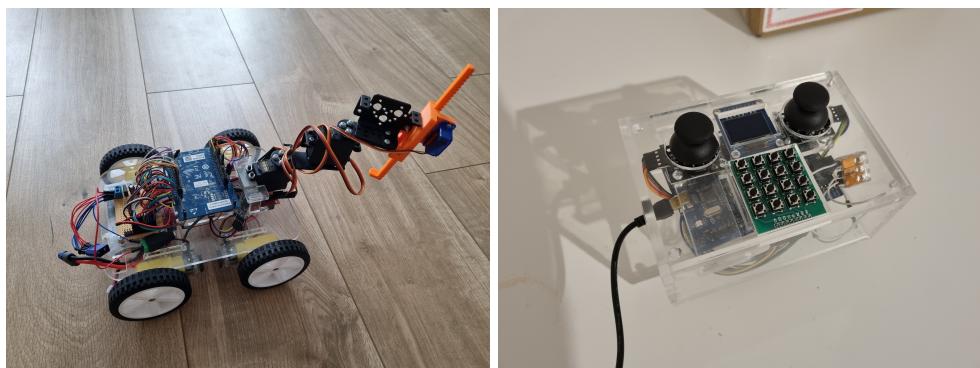
Manipulator został wykonany z gotowych elementów tj. mocowań wykonanych specjalnie pod serwomechanizmy. Tyczy się to elementów opowiedzialnych za człony rotacyjne. Człon translacyjny został zaprojektowany w środowisku Autodesk Inventor i wydrukowany na drukarce 3D. Platforma oraz pilot zostały zaprojektowane specjalnie na potrzeby projektu. Mają one charakter modułowy pozwalający na ewentualne zmiany lub uzupełnienie ich o kolejne elementy. Części zostały wycięte laserwo z pleksi.



Rysunek 13: Konstrukcja mechaniczna manipulatora



Rysunek 14: Konstrukcja mechaniczna pilota



Rysunek 15: Wykonane

## 7 Kinematyka manipulatora

Dzięki wyliczonej kinematyce prostej jesteśmy w stanie obliczyć pozycję punktu pracy manipulatora z zadanych kątów dla serwomechanizmów, natomiast odwrotna kinematyka pozwala nam wyliczyć kąty obrotu dla serwomechanizmów z upragnionych współrzędnych XYZ.

### 7.1 Kinematyka prosta

Lp.	$\theta_i$	$d_i$	$a_i$	$\varphi_i$
1	$q_1$	$a_1$	$a_2$	0
2	$q_2$	$a_3$	$a_4$	$\pi$
3	0	$q_3$	0	0

Tabela 7: Tabela Denavita - Hartenberga

Obliczone macierze transformacji pomiędzy kolejnymi układami współrzędnych zgodnie z tabelą 7

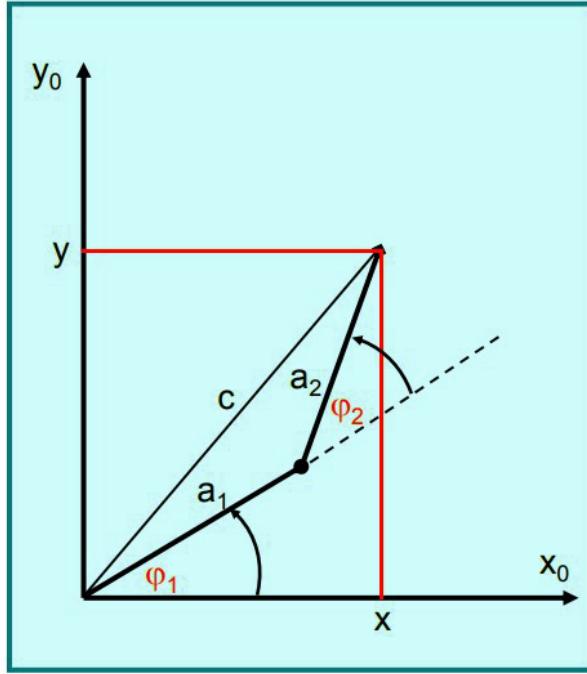
$$R_0^1 = \begin{bmatrix} \cos(q_1) & -\sin(q_1) & 0 & \cos(q_1)a_2 \\ \sin(q_1) & \cos(q_1) & 0 & 0 \\ 0 & 0 & 1 & \cos(q_1)a_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_1^2 = \begin{bmatrix} \cos(q_2) & -\sin(q_2) & 0 & \cos(q_2)a_4 \\ \sin(q_2) & \cos(q_2) & 0 & 0 \\ 0 & 0 & 1 & \sin(q_2)a_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_2^3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -q_3 - a_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_0^3 = \begin{bmatrix} \cos(q_1 + q_2) & -\sin(q_1 + q_2) & 0 & a_2\cos(q_1) + a_4\cos(q_1 + q_2) \\ \sin(q_1 + q_2) & \cos(q_1 + q_2) & 0 & a_2\sin(q_1) + a_4\sin(q_1 + q_2) \\ 0 & 0 & 1 & -q_3 + a_1 + a_3 - a_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

## 7.2 Odwrotna kinematyka



Rysunek 16: Rzut manipulatora z góry

$$\begin{aligned}
 X^2 + Y^2 &= r^2 \\
 r &= \sqrt{X^2 + Y^2} \\
 \theta_1 &= \varphi_2 - \varphi_1 \\
 \tan(\varphi_2) &= \frac{Y}{X} \\
 \varphi_2 &= \tan^{-1}\left(\frac{Y}{X}\right) \\
 a_4^2 &= a_2^2 + c^2 - 2a_2 * r * \cos(\varphi_1) \\
 \varphi_1 &= \cos^{-1}\left(\frac{a_4^2 - a_2^2 - r^2}{-2a_2r}\right) \\
 \varphi_3 + \theta_2 &= 180 \\
 \theta_2 &= 180 - \varphi_3 \\
 r^2 &= a_2^2 + a_4^2 - 2a_2a_4\cos(\varphi_3) \\
 \varphi_3 &= \cos^{-1}\left(\frac{r^2 - a_2^2 - a_4^2}{-2a_2a_4}\right)
 \end{aligned}$$

Znając równania na:

$$\varphi_1, \varphi_2, \varphi_3$$

możemy uzyskać upragnione kąty przy zadanych współrzędnych z równań:

$$\theta_1 = \varphi_2 - \varphi_1$$

$$\theta_2 = \pi - \varphi_3$$

## 8 Opis działania programu

### 8.1 Wybór trybu sterowania

Cały program jest podzielony na różne tryby pracy pomiędzy którymi użytkownik może się dowolnie poruszać w przygotowanym menu wyświetlanym na wyświetlaczu OLED. Przełączenie trybu następuje podczas wysyłania ramki danych z pilota, gdzie na początku każdej ramki przesyłany jest ID trybu, dzięki któremu program przemieszcza się po pętlach w programie znajdującym się w jednostce sterującej pojazdem. Możemy wyróżnić trzy tryby działania:

1. Tryb ręczny – sterowanie pojazdu oraz samego manipulatora za pomocą joysticków umieszczonych na pilocie
2. Tryb koordynatów – przemieszczenie pojazdu do zadanych przez użytkownika współrzędnych XYZ.
3. Tryb sekwencja – w tym trybie mamy możliwość zapisu oraz odczytu wcześniej zapisanych sekwencji. Możliwe jest zapisanie do 3 różnych sekwencji, których długość jest ograniczona 20 punktów.

Implementacja jednego z trybów została przedstawiona na listingu poniżej.

```
1 while(id == 'c') // tryb koordynatów
2 {
3     if(flagUART == 1)
4     {
5         parse_ReceivedData();
6         flagUART = 0;
7         HAL_UART_Receive_DMA(&huart2, (uint8_t*)rxBuf, 26);
8         move = 1;
9     }
10
11    if(id == 'c')
12    {
13        if(move == 1)
14        {
15            InverseKinematics(Data[0], Data[1], Data[2], &obiekty, &position, 0);
16            moveManipulator(&htim2, Theta1(obiekty.servo1), Theta2(obiekty.servo2),
17                             Theta3(obiekty.servo3));
18            move = 0;
19        }
20    }
}
```

### 8.2 Wysyłane ramki danych

Każdy tryb różni się ilością wartości zmiennych do przesłania. Stworzyło to problemy podczas przechodzenia pomiędzy trybami, ponieważ zmuszało nas to do zmianiania wartości argumentu funkcji do odbierania danych przez komunikację UART. Program nie radził sobie z tymi zmianami, dlatego ustaliliśmy, że będziemy wysyłać ramkę o stałej długości gdzie puste miejsca są wypełnione zerami.

### 8.3 Interpretacja zadanych kątów

Podstawową operacją którą program musi wykonywać jest przeliczenie wartości zwróconych za pomocą odwrotnej kinematyki na użyteczne wartości wypełnienia sygnału PWM dla serwomechanizmów. Wyliczone wartości zapisujemy w strukturze **servo**

```
1 typedef struct
2 {
3     float servo1;
4     float servo2;
5     float servo3;
6 } servo;
```

Wyliczone kąty z odwrotnej kinematyki trzeba jeszcze poddać operacji przeliczania na wartości z dopuszczalnego zakresu wypełnienia sygnału PWM. Poniżej przedstawiono funkcję przeliczającą wartość

sygnału PWM z wartością kąta pierwszego przegubu rotacyjnego. Porusza się on w zakresie od  $0^\circ$  do  $180^\circ$ . W tym przypadku aby dobrze interpretować wyniki z odwrotnej kinematyki trzeba dodać  $180^\circ$  do jej wyniku aby zostały dobrane odpowiednie kąty przy zadaniu pozycji z lewej strony układu kartezjańskiego.

```

1 float Theta1(float angle)
2 {
3     angle = (angle / 3.14159) * 180;
4     if (angle < 0)
5         angle = angle + 180;
6     float value;
7     int minC=1600;
8     int maxC=7800;
9     int minA=0;
10    int maxA=180;
11    value = ((maxC-minC) / (maxA-minA)) * (angle - minA) + minC;
12    return value;
13 }
```

Użycie systemu DMA dla joysticków bardzo ułatwia pracę polegającą na interpretacji wyników dzięki czemu za pomocą kilku prostych funkcji można zadawać pozycję pojazdu oraz manipulatora z odczytów z joysticka.

```

1 float Theta1(float angle)
2 {
3     typedef struct
4     {
5         uint16_t position[2];
6     } joystick;
7 }
8
9 int voltageToAngleX(const joystick obiekt)
10 {
11     int value;
12     value = 1600 + (obiekt.position[0] * 1.514);
13     return value;
14 }
```

## 8.4 Wykorzystanie odwrotnej kinematyki i wykonywanie ruchu

Dla zadanych współrzędnych zewnętrznych obliczane są współrzędne wewnętrzne na podstawie równań kinematyki odwrotnej.

Funkcja InverseKinematics(float X, float Y, float Z, servo \*obiekt, manipulatorposition \*position, int mode) przelicza zadane wartości x, y, z na wartości współrzędnych przegubowych. Manipulatory typu SCARA mogą przemieścić się do zadanej pozycji za pomocą dwóch konfiguracji "ramię w dół" bądź "ramię w góre". To w jaki sposób ustawi się manipulator na zadanej pozycji zależy od wartości argumentu "mode" funkcji wykonującej obliczenia odwrotnej kinematyki.

```

1 void moveServo1(TIM_HandleTypeDef *htim, float angle)
2 {
3     if (previous < angle)
4     {
5
6         for (; previous < angle; previous += 10)
7             if (previous > angle)
8                 {
9                     previous = angle;
10                }
11            __HAL_TIM_SET_COMPARE(htim, TIM_CHANNEL_1, previous);
12            HAL_Delay(20);
13        }
14    }
15
16    else
17    {
```

```

18     for ( ; previous > angle; previous -= 10) {
19         if (previous < angle) {
20             previous = angle;
21         }
22         HAL_TIM_Set_COMPARE(htim, TIM_CHANNEL_1, previous);
23         HAL_Delay(20); //60
24     }
25 }
26 }
```

Funkcja przedstawiona powyżej, pokazuje w jaki sposób zmieniane jest wypełnienie sygnału PWM, który jest odpowiedzialny za przemieszczanie serva w zakresie od 0 do 180 stopni. Dodatkowo aby uniknąć oscylacji manipulatora wywołanych zbyt dużym momentem drugi przegub obrotowy jest zawsze prowadzony do pozycji -90 lub 90 stopni, aby zminimalizować czas w jakim wykonywana jest ta czynność, został napisany fragment kodu odpowiedzialny za przeniesienie manipulatora do najbliższej skrajnej pozycji.

## 8.5 Zapisywanie sekwencji w wewnętrznej pamięci FLASH

Do zapisu oraz odczytu wygenerowanych sekwencji wykorzystywana jest wewnętrzna pamięć FLASH płytka. Każdy wybrany punkt jest zapisywany w formie tablicy trzech znaków (XYZ) do kolejnych punktów w sektorze pamięci, a następnie w tej samej kolejności są odczytywane.

```

1 void MY_FLASH_WriteN(uint32_t idx, void *wrBuf, uint32_t Nsize, DataTypeDef
                      dataType)
2 {
3     uint32_t flashAddress = MY_SectorAddrs + idx;
4     MY_FLASH_EraseSector();
5     HAL_FLASH_Unlock();
6
7     switch(dataType)
8     {
9         case DATA_TYPE_8:
10            for(uint32_t i=0; i<Nsize; i++)
11            {
12                HAL_FLASH_Program(FLASH_TYPEPROGRAM_BYTE, flashAddress, ((uint8_t *)wrBuf)[i]);
13                flashAddress++;
14            }
15            break;
16
17        case DATA_TYPE_16:
18            for(uint32_t i=0; i<Nsize; i++)
19            {
20                HAL_FLASH_Program(FLASH_TYPEPROGRAM_HALFWORD, flashAddress, ((uint16_t *)wrBuf)[i]);
21                flashAddress+=2;
22            }
23            break;
24
25        case DATA_TYPE_32:
26            for(uint32_t i=0; i<Nsize; i++)
27            {
28                HAL_FLASH_Program(FLASH_TYPEPROGRAM_WORD, flashAddress, ((uint32_t *)wrBuf)[i]);
29                flashAddress+=4;
30            }
31            break;
32    }
33    HAL_FLASH_Lock();
34 }
```

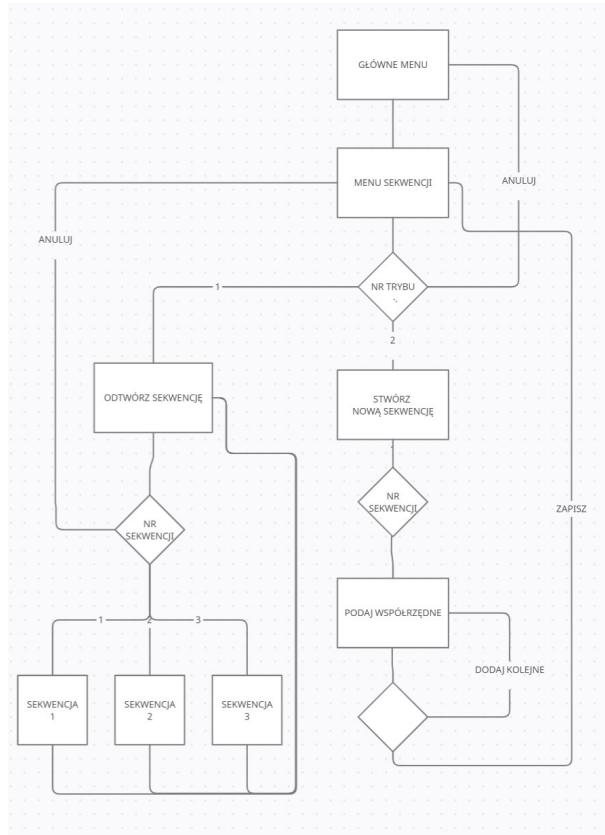
## 8.6 Rysowanie interfejsu użytkownika

Rysowanie interfejsu stanowiło jedno z ciekwszych wyzwań. Do dyspozycji mamy mały wyświetlacz o niskiej rozdzielcości, więc dobre rozplanowanie układu było kluczowym zadaniem. W sytuacjach, kiedy nie było już możliwości umieszczenia większej ilości elementów, decydowaliśmy się na zaimplementowanie podmenu. Poruszanie po interfejsie odbywa się za pomocą przycisków znajdujących się pod wyświetlaczem.

```

1 void drawMenu()
2 {
3     clearMenu();
4     ssd1331_display_string(7, 0, "1. Reczny", FONT_1206, GREEN);
5     ssd1331_display_string(7, 11, "2. Sekwencja", FONT_1206, GREEN);
6     ssd1331_display_string(7, 22, "3. Coords", FONT_1206, GREEN);
7
8     ssd1331_draw_line(0,50,95,50, WHITE);
9     ssd1331_display_string(83, 51, "OK", FONT_1206, GREEN);
10    ssd1331_draw_line(0,63,95,63, WHITE);
11 }

```



Rysunek 17: Diagram przepływu

## 8.7 Odometria

Implementacja odometrii była jednym z ostatnich etapów projektów. Do orientacji w terenie skorzystaliśmy z enkoderów zamontowanych na kołach. Pojazd w trakcie poruszania się zlicza tiki z enkoderów i zapisuje te wartości w pamięci. Z naszych obliczeń wynika, że 25cm przejechanej trasy to 3840 tików. Bazując na tej wartości dokonywalismy dalszych obliczeń. Niestety ze względu na luzy na trzpieniu silników elektrycznych mieliśmy spore problemy z uzyskaniem dokładnego obrotu, ponieważ jest mocno zależy od nawierzchni oraz ułożenia kół.

```
1 void MOVE( float xCoord , float yCoord , TIM_HandleTypeDef *htim )
2 {
3     float angleRot ;
4
5     if (yCoord != robotPositon .y)
6     {
7         if (yCoord > robotPositon .y)
8         {
9             angleRot = -robotPositon .angle ; // 0
10            robotPositon .angle = 0;
11        }
12
13        if (yCoord < robotPositon .y)
14        {
15            if (robotPositon .angle >= 0) angleRot = 180 - robotPositon .angle ; // 180
16            if (robotPositon .angle < 0) angleRot = -180 - robotPositon .angle ;
17            robotPositon .angle = 180;
18        }
19
20        rotate _ robot (angleRot , htim );
21        HAL _ Delay (1000);
22        htim5 .Instance->CNT = ENC _ ZERO;
23        drive _ robot (abs (robotPositon .y - yCoord ) , 0 , htim );
24        HAL _ Delay (1000);
25        htim5 .Instance->CNT = ENC _ ZERO;
26        robotPositon .y = yCoord ;
27    }
28
29    if (xCoord != robotPositon .x)
30    {
31        if (xCoord > robotPositon .x)
32        {
33            angleRot = 90 - robotPositon .angle ;
34            robotPositon .angle = 90;
35        }
36        if (xCoord < robotPositon .x)
37        {
38            if (robotPositon .angle == 0) angleRot = -90 - robotPositon .angle ;
39            if (robotPositon .angle == 180) angleRot = -90 + robotPositon .angle ;
40            robotPositon .angle = -90;
41        }
42
43        rotate _ robot (angleRot , htim );
44        HAL _ Delay (1000);
45        htim5 .Instance->CNT = ENC _ ZERO;
46
47        drive _ robot (abs (robotPositon .x - xCoord ) , 0 , htim );
48        HAL _ Delay (1000);
49        htim5 .Instance->CNT = ENC _ ZERO;
50        robotPositon .x = xCoord ;
51    }
52 }
```

## 9 Podsumowanie

W ramach projektu przygotowano pojazd czterokołowy z manipulatorem o trzech stopniach swobody – dwóch rotacyjnych i jednym translacyjnym. Sterowanie pojazdu zostało zrealizowane w oparciu o płytę STM32F411VET6 Discovery. Do realizacji zadania wykorzystano wiele peryferiów, m. in. timerów, generacja PWM, wejścia analogowe, interfejsy komunikacyjne takie jak UART i SPI.

W konstrukcji wykorzystaliśmy gotowe elementy i także samodzielnie zaprojektowane przy pomocy środowiska Autodesk Inverntor. Przegub translacyjny manipulatora został wykonany przy pomocy techniki druku 3D, natomiast podwozie i pilot są wycięte laserowo z pleksi.

Pojazd i manipulator zapewnia sterowanie ręczne przy pomocy joysticka, bądź automatyczne i sekwencyjne za pomocą pilota i stworzonego interfejsu użytkownika. Do sterowania pojazdu zastosowaliśmy regulator PID, który bardzo dobrze się sprawdził do dokładnego dojazdu do zadanej pozycji.

Napotkaliśmy problem związany z komunikacją przez moduły bluetooth – komunikacja nie radziła sobie z szybkim przesyłem danych przez co sterowanie ręczne nie działa płynnie tak jak się tego spodziewaliśmy.

Mimo problemów wszystkie założenia projektowe zostały spełnione co sprawia, że reszta naszej pracy jak najbardziej nas zadowala. Projekt ma ogromne perspektywy na rozwój i rozbudowywanie możliwości pojazdu i manipulatora poprzez na przykład dołączenie efektora takiego jak chwytak czy też elektromagnete do podnoszenia lekkich przedmiotów.

## Literatura

- [1] Marek Galewski. *Aplikacje i ćwiczenia w języku C z biblioteką HAL.*  
Wydawnictwo BTC, Legionowo, Lipiec 2019.
- [2] M. W. Spong and M. Vidyasagar. *Dynamika i Sterowanie Robotów.*  
Wydawnictwo Naukowo-Techniczne 1997.
- [3] ITeadStudio. *HC-05 datasheet*
- [4] Waveshare. *0.95inch RGB OLED User Manual*