Artur ZAWADA[1]

Opiekun naukowy: Jarosław JANUSZ[2]

PROJEKT I BUDOWA RAMIENIA STEROWANEGO MIKROPROCESOROWO

Streszczenie: W artykule przedstawiono przebieg prac związanych z zaprojektowaniem i budową ramienia na mobilnym robocie sterowanym za pomocą mikrokontrolera Arduino oraz implementacją kinematyki prostej i odwrotnej do języka C++. Takie zadanie pozwala na przedstawienie możliwości pracy manipulatora pracującego na mobilnej podstawie. W niektórych aplikacjach taki robot może zastąpić manipulator przemysłowy.

Słowa kluczowe: mikrokontroler, Arduino, robot mobilny, kinematyka prosta, kinematyka odwrotna, język C++, manipulator.

DESIGN AND IMPLEMENTATION OF AN ARM CONTROLLED BY MICROPROCESSOR

Summary: The article presents the course of work related to the design and the construction of an arm installed on a mobile robot controlled by the Arduino microcontroller. Moreover, the implementation of simple and inverse kinematics in an application written in the C++ language is described. This task allows for presentation of the possible manipulator working ranges. The arm is fixed on a mobile base. In some applications such a robot can replace an industrial robotic manipulator.

Keywords: microcontroller, Arduino, mobile robot, simple kinematics, inverse kinematics, C++ language, manipulator.

1. Wstęp

Celem projektu było zaprojektowanie oraz budowa manipulatora na istniejącym robocie mobilnym, który umożliwia łapanie oraz transportowanie niewielkich rozmiarów przedmiotów (w założeniu piłeczki o średnicy 40mm). Do budowy manipulatora należało zastosować części handlowe, aby umożliwić szybką wymianę w przypadku uszkodzenia którejkolwiek z części manipulatora, natomiast wszystkie z zastosowanych serwomechanizmów musiały mieć możliwość sterowania z zainstalowanego na robocie mikrokontrolera Arduino Due.

[1] Akademia Techniczno-Humanistyczna w Bielsku-Białej, Wydział Budowy Maszyn i Informatyki, specjalność: Mechatronika i Robotyka, Artur.zawada88@gmail.com
[2] dr inż., Akademia Techniczno-Humanistyczna w Bielsku-Białej, Wydział Budowy Maszyn i Informatyki, jjanusz@ath.bielsko.pl

2. Schemat

Projektowany manipulator to manipulator trzyosiowy wyposażony w chwytak zabudowany na mobilnej platformie.

2.1. Konstrukcja manipulatora

Rysunek 1 przedstawia konstrukcje manipulatora wraz ze zaznaczonymi wymiarami (Tabela 1.) oraz układem współrzędnych, natomiast tabela (Tabela 2.) zawiera zestawienie części potrzebnych do budowy.

Rysunek 1. Konstrukcja manipulatora

Tabela 1. Wymiary manipulatora

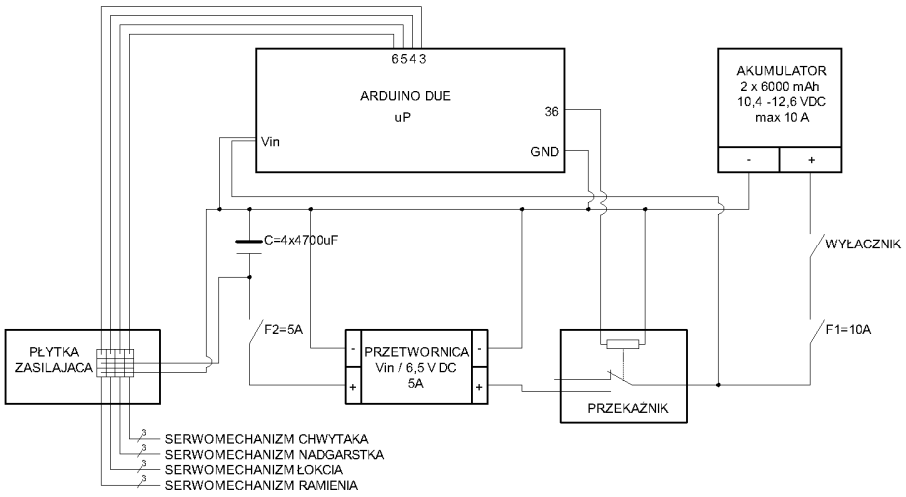
L.p.	Oznacz.	Wymiar [mm]	Opis
1.	A	130	Od osi obrotu robota do osi pierwszego serwa wzdłuż osi x
2.	B	60	Od podłoża do osi obr. pierwszego serwa wzdłuż osi z
3.	C	59	Pomiędzy osiami obr. serwa pierwszego i drugiego
4.	D	105	Pomiędzy osiami obr. serwa drugiego i trzeciego
5.	E	85	Od osi obr. Serwa trzeciego do końca chwytaka po osi z
6.	F	23	Od osi obr. Serwa trzeciego do końca chwytaka po osi x

Tabela 2. Zestawienie części wchodzących w skład manipulatora [3]

L.p.	Nazwa części	Ilość sztuk
1.	Chwytak do serw typu standard-Actobotics Standard Gripper Kit A	1
2.	Uchwyt do serwa Feetech FK-US-003 - czarny	3
3.	Aluminiowy uchwyt do serwa Feetech FK-MB-001 - czarny	3
4.	Aluminiowe mocowanie Feetech FK-LS-001 do serw - czarne	1
5.	Serwo Power HD LF-20MG standard	3
6.	Przedłużacz do serw 30 cm skręcony	3
7.	Serwo Tower Pro MG-946R - standard	1
8.	Śruba M4x10 z nakrętką	9
9.	Śruba stożkowa M3x8 z nakrętką	12
10.	Śruba M3x16 z nakrętką	2
11.	Śruba M3x8	16

2.2 Schemat połączeń

Rysunek 2 przedstawia schemat połączeń manipulatora do istniejącej na robocie płytki zasilającej podłączonej do pinów PWM na mikrokontrolerze oraz przetwornicy 12,6/6,3VDC zabezpieczonej bezpiecznikiem F2=5A, która dostarcza odpowiedni poziom zasilania, aby serwomotory mogły pracować z maksymalnym momentem przewidzianym przez producenta. Zasilanie serwomotorów zostało zaprojektowane w sposób umożliwiający wyłączenie zasilania bez konieczności wyłączenia całego robota. Dodatkowo po przetwornicy znajdują się kondensatory 4x4700uF mające na celu zapobieganie spadkom napięcia w momencie nagłej zmiany kierunku obrotu wału serwomotorów co skutkuje kilkukrotnie większym poborem prądu. Cały układ zabezpieczony jest bezpiecznikiem F1=10A ponieważ jest to maksymalny prąd jaki można uzyskać z akumulatora.



Rysunek 2. Schemat połączeń

3. Sterowanie ramieniem – opis zagadnień teoretycznych

Sterowanie serwomechanizmów odbywa się za pomocą sygnału PWM (Ang. Pulse Width Modulation) i w zależności od sygnału na wyjściu serwomechanizm ustawia się w zadanej pozycji. Niestety serwomechanizmy takiego typu nie posiadają sprzężenia zwrotnego przez co układ nie jest w stanie wykonać korekcji pozycji.

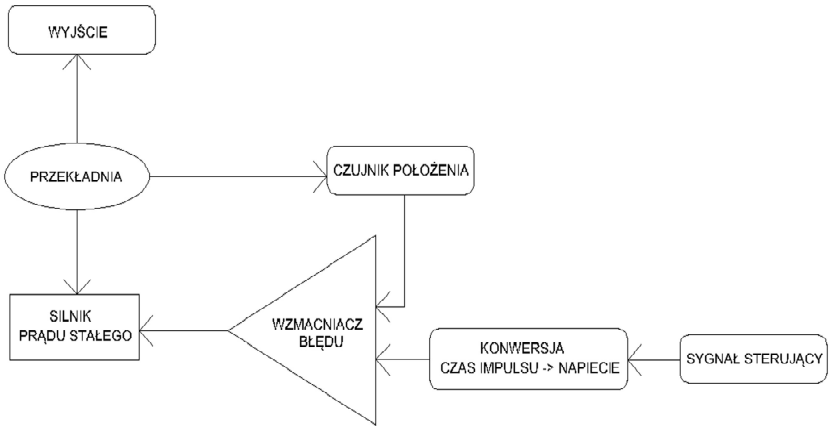
3.1 Arduino IDE [4]

Arduino IDE jest to oprogramowanie w języku Java udostępniane przez producenta Arduino, które pozwala na programowanie mikrokontrolera w języku C++. Oprogramowanie to ma wbudowany kompilator avr-gcc, który jest odpowiedzialny za kompilacje języka C++ na język akceptowany przez mikrokontroler. Arduino IDE działa również jako emulator portu szeregowego co pozwala na wgrywanie skompilowanego programu oraz umożliwia interakcje z pracującym Arduino.

3.2 Serwomechanizm – silnik DC z wbudowaną przekładnią zębatą [1]

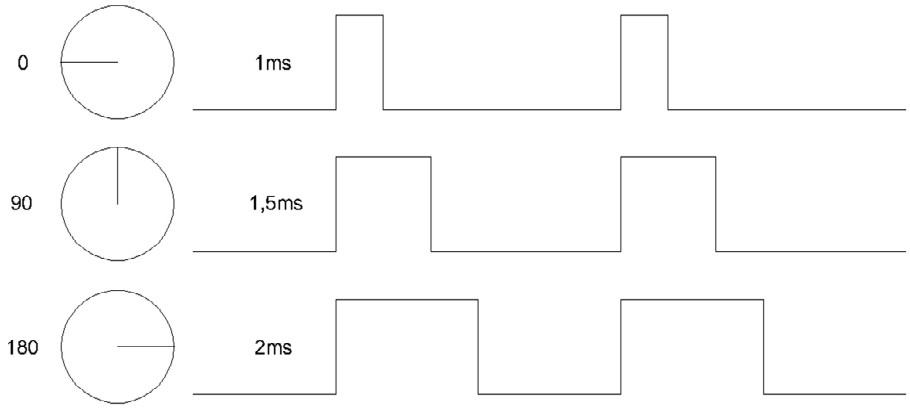
Serwomechanizm (Rysunek 3.) składa się z silnika prądu stałego, przekładni, czujnika położenia oraz układu sterującego który zamienia sygnał PWM na odpowiedni poziom napięcia. Układ sterujący utrzymuje zadaną pozycje z dokładnością zabudowanego czujnika położenia (najczęściej potencjometru) oraz w miarę możliwości wynikających z mocy danego silnika. Wał standardowego serwomechanizmu może obracać się o 90° stopni w każdą stronę co pozwala na całkowity obrót o 180°. Istnieją serwomotory umożliwiające obrót o 360°, ale standardowe serwomotory ze względu na wbudowane mechaniczne ograniczniki nie są w stanie wykonać takiego obrotu. Serwomechanizm posiada wyprowadzone trzy przewody:

- brązowy V+ (poziom napięcia różni się w zależności od zastosowanych serwomotorów najczęściej spotykane napięcia to 4,8-6,4VDC)
- czerwony GND (masa układu)
- pomarańczowy sygnał sterujący (PWM)



Rysunek 3. Zasada działania serwomechanizmu

W serwomotorach stosuje się układy zamkniętej pętli, które dostarczają do sterownika informacje pozwalające na utrzymanie określonego położenia wału lub w silnikach 360° określonej prędkości. Wraz ze zmianą impulsu sterującego układ określa różnicę pomiędzy aktualną pozycją a zadaną, następnie z maksymalną prędkością dąży do uzyskania jednakowej wartości.
Sygnał sterujący PWM składa się z impulsów wysyłanych co 20ms, a długość impulsu określa położenie wału (Rysunek 4.) we wszystkich serwomechanizmach impuls 1,5ms ustawia wał w pozycji środkowej, natomiast skrajne pozycje zależą od zastosowanego modelu.



Rysunek 4. Zasada pozycjonowania serwomechanizmu

3.3 Biblioteka Servo.h [5]

Biblioteka Servo.h która jest dostarczona wraz z oprogramowaniem Arduino IDE pozwala na regulacje pozycji serwa w granicach od 0° do 180° przez co w kodzie programu nie trzeba stosować przekształceń z zadanego przez nas kąta na sygnał PWM. Tabela 3 zawiera zestawienie funkcji, z których możemy skorzystać, jeżeli używamy tej biblioteki.

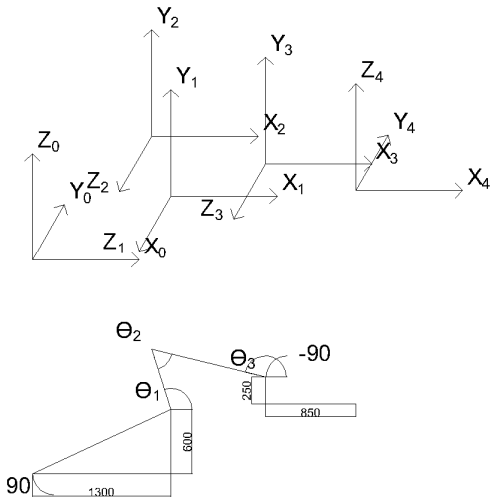
Tabela 3. Funkcje biblioteki servo.h

L.p.	Funkcja	Opis
1.	`attach`	Przypisuje zmienną servo do pinu
2.	`write`	Przypisuje zadaną pozycje od 0 do 180 do danego pinu
3.	`writeMicroseconds`	Przypisuje pozycje z zakresu od 1000us (0) do 2000us (180) dla standardowych serw
4.	`read`	Odczytuje aktualny kąt serwa (ostatnie wywołanie „write")
5.	`attached`	Sprawdza czy do danego pinu jest przypisana zmienna servo
6.	`detach`	Usuwa zmienną servo z danego pinu

Przedstawiona biblioteka została zaprojektowana dla standardowych serwomechanizmów, gdzie sygnał PWM mieści się w zakresie od 1000us do 2000us dlatego w przypadku stosowanych w projekcie serwomechanizmów (Tabela 1.) możemy korzystać z funkcji `writeMicroseconds(x)` lub z funkcji `write(x)` ale wprowadzając w miejsce argumentu `(x)` funkcje `map(x,0,90,0,100)` [6] dzięki której zadana pozycja z zakresu od 0 do 90 zostanie przekształcona na wartość z zakresu od 0 do 100 która odpowiada zastosowanym typom serwomechanizmów dodatkowo zastosowaniem funkcji `map()` umożliwi ewentualne korekty jeżeli podczas testów okaże się ze zadana pozycja serwomechanizmu nie odpowiada rzeczywistej.

3.4 Kinematyka prosta - Notacja Denavita-Hartenberga [2]

Manipulator jest to najprościej mówiąc zbiór ciał połączonych ze sobą w łańcuch kinematyczny, ciała te noszą nazwę członów, a połączenie ruchowe dwóch sąsiadujących ze sobą członów nazywamy parą niższego rzędu. Analizując rozwiązania mechaniczne manipulatorów można zauważyć, że najczęściej spotykane są pary o jednym stopniu swobody, czyli pary obrotowe lub postępowe. Człony numeruje się zaczynając od nieruchomej podstawy i oznaczając ją członem 0, pierwszy ruchomy człon oznaczony jest numerem 1 i tak do końca ramienia. Kolejnym krokiem jest przywiązanie układów współrzędnych do poszczególnych członów manipulatora. W tym celu korzystamy z notacji Denavita-Hartenberga, która zakłada, że oś $\hat{Y}$ jest nieruchoma tzn. nie możemy wykonać przesunięcia wzdłuż tej osi ani obrotu. Układy współrzędnych przywiązuje się tak aby układ 'i' był przywiązany sztywno do osi 'i'.
Układ 0 przywiązujemy do członu 0 i jako że, jest on nieruchomy stanowi on układ odniesienia. Postępując zgodnie z założeniem notacji umożliwiającej translacje i obrót po osiach $\hat{X}$ oraz $\hat{Z}$, w przypadku zastosowanego w projekcie manipulatora poprawnie przywiązane układy współrzędnych przedstawia Rysunek 5.



Rysunek 5. Układy współrzędnych oraz kąty obrotu

W celu obliczenia pozycji chwytaka względem układu odniesienia należy stworzyć tabele, w której znajdują się kolejno:

Θ_i - kąt między osiami $\hat{X}_{i-1}$ do $\hat{X}_i$ mierzony wokół $\hat{Z}_i$

d_i - odległość od osi $\hat{X}_{i-1}$ do $\hat{X}_i$ mierzona wzdłuż osi $\hat{Z}_i$

a_i - odległość od osi $\hat{Z}_i$ do $\hat{Z}_{i+1}$ mierzona wzdłuż osi $\hat{X}_i$

α_i - kąt między osiami $\hat{Z}_i$ do $\hat{Z}_{i+1}$ mierzony wokół $\hat{X}_i$

Tabela 4 przedstawia utworzoną na podstawie układów współrzędnych oraz kątów obrotu tabelę notacji Denavita-Hartenberga dla zastosowanego w projekcie manipulatora.

Tabela 4. Tabela utworzona zgodnie z notacją Denavita-Hartenberga

L.p.	Θ_i	d_i	a_i	α_i
1.	0	60	130	90
2.	Θ_1	0	Length [0]	0
3.	Θ_2	0	Length [1]	0
4.	Θ_3	0	0	-90
5.	0	Length [3]	Length [2]	0

Poniżej przedstawiona została Tabela 4 w postaci macierzy o wymiarach 4x4 zaimplementowana do języka C++.

```
float MatDH[5][4]{{0,60,130,90},  //tabela notacji dh
     {map(Pos[0]-ConstPos[0][5], 0,100,0,90),0,Length[0],0},
     {map(Pos[1]-ConstPos[1][5], 0,100,0,90),0,Length[1],0},
     {map((Pos[2]-ConstPos[2][5])*(-1),0,103,0,90),0,0,-90},
     {0,Length[3],Length[2],0}};
```

Uzyskana w ten sposób macierz odwzorowuje cały manipulator, a do uzyskania orientacji chwytaka wystarczy wykonać obliczenia macierzowe dla każdego wiersza wstawiając kolejne wartości do odpowiednich macierzy znajdujących się poniżej i tak dla pierwszego wiersza wstawiamy 0 do macierzy „`ROT Z`", 60 do „`TRANS Z`", 130 do „`TRANS X`" oraz 90 do „`ROT X`". Następnym krokiem jest wykonanie mnożenia tych macierzy i zapisanie wyniku do macierzy `MatRes` (Ang. Matrix Result) o indeksie 0, analogiczne wykonywane są obliczenia dla każdego wiersza a ostatnim etapem jest wykonanie mnożenia `MatRes` od 0 do 4 i zapisanie wyniku do macierzy `CALCULATED_MATRIX_DH` i dla pozycji jak na Rysunku 1 uzyskana macierz przedstawia (1) gdzie rot_4^0 to macierz określająca kąt pod jakim znajduje się chwytak względem nieruchomego układu odniesienia natomiast x, y, z to przesunięcia po osiach $\hat{x}, \hat{y}, \hat{z}$.

$$\begin{bmatrix} & & & x \\ & rot_0^4 & & y \\ & & & z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 284 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 40 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

Poniżej znajdują się macierze obrotów i translacji zgodne z notacją Denavita-Hartenberga zapisane w postaci języka C++ , warto zwrócić uwagę iż kąty w funkcjach `sin()` oraz `cos()` zapisane są w radianach przez co wartości katów w macierzy `MatDH` należy pomnożyć razy `(PI/180)`.

```
const float MatRot[4][4][4]= {
{{cos(MatDH[i][0]*(PI/180)),-sin(MatDH[i][0]*(PI/180)),0,0},
{sin(MatDH[i][0]*(PI/180)),cos(MatDH[i][0]*(PI/180)),0,0},
{0,0,1,0},
{0,0,0,1}},                    //ROT Z

{{1,0,0,0},
{0,1,0,0},
{0,0,1,MatDH[i][1]},
```

```
{0,0,0,1}},                    //TRANS Z
{{1,0,0,MatDH[i][2]},
{0,1,0,0},
{0,0,1,0},
{0,0,0,1}},                    //TRANS X

{{1,0,0,0},
{0,cos(MatDH[i][3]*(PI/180)),-sin(MatDH[i][3]*(PI/180)),0},
{0,sin(MatDH[i][3]*(PI/180)),cos(MatDH[i][3]*(PI/180)),0},
{0,0,0,1}}};                   //ROT X
```

3.6 Kinematyka odwrotna [2]

W podpunkcie 3.5 omówiona została kinematyka prosta, czyli obliczenie aktualnej pozycji chwytaka względem nieruchomego układu współrzędnych, jakim dla manipulatora jest oś obrotu konstrukcji robota, natomiast kinematyka odwrotna jest o wiele trudniejszym zagadnieniem, gdyż jej zadaniem jest znaleźć odpowiednie ułożenie dla każdej pary kinematycznej (w zastosowanym manipulatorze dla trzech par obrotowych) tak, aby koniec chwytaka znalazł się w zadanej pozycji i pod odpowiednim kątem. Istnieje kilka metod, które można zastosować między innymi numeryczna, geometryczna oraz algebraiczna. Każda z tych metod ma jednak swoje wady i zalety co sprawia, że dobór odpowiedniej metody jest bardzo ważny. Do zastosowanego w projekcje manipulatora najlepiej sprawdza się metoda geometryczna ze względu na swoją prostotę oraz możliwości uproszczeń. Z pojęciem manipulatora wiąże się również pojęcie przestrzeni roboczej, która jest zbiorem punktów w przestrzeni, które końcówka wykonawcza (chwytak) może osiągnąć, zatem aby można było uzyskać zadaną pozycję należy sprawdzić naj-pierw czy znajduje się ona w przestrzeni roboczej manipulatora. Przestrzeń roboczą warto również podzielić na manipulacyjna i osiągalną. Przestrzeń manipulacyjna jest to przestrzeń, do której manipulator może dotrzeć z dowolną orientacją natomiast osiągalna to przestrzeń, do której manipulator może dotrzeć przynajmniej z jedną orientacją. Zastosowany w projekcie trójczłonowy płaski manipulator ze względu na swoją budowę można potraktować jako dwuczłonowy traktując ostatni człon jako narzędzie. Takie uproszczenie sprawia, że za pomocą metody algebraicznej wystarczy wyznaczyć położenie i orientacje układu 3 względem 1 (Rysunek 5.). W przy przypadku kinematyki prostej macierz według notacji Denavita-Hartenberga przedstawia równanie 2.

$$\begin{bmatrix} & & & x_1^3 \\ & rot_1^3 & & y_1^3 \\ & & & z_1^3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos_{123} & -\sin_{123} & 0 & l_0\cos_1 + l_0\cos_{12} \\ \sin_{123} & \cos_{123} & 0 & 0 \\ 0 & 0 & 1 & l_1\sin_1 + l_1\sin_{12} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

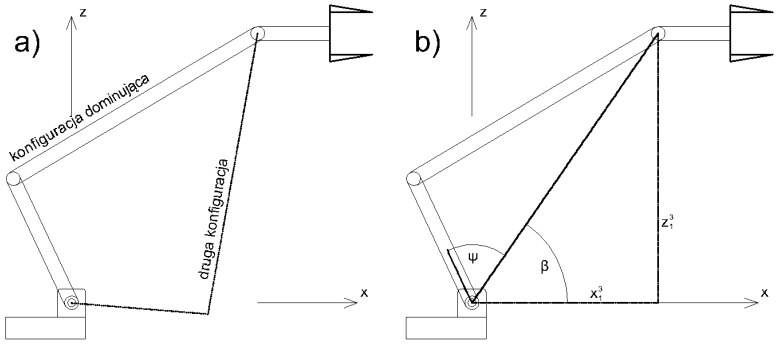
Jak można zauważyć wartości x_1^3 oraz z_1^3 zależą od sumy długości ramion Length[0] oraz Length[1] oraz wartości kątów pomiędzy nimi z czego można wyznaczyć równanie 3 na podstawie którego możemy sprawdzić czy zadana pozycja znajduje się w przestrzeni roboczej.

$$\sqrt{\left(x_1^3\right)^2 + \left(z_1^3\right)^2} \le length[0] + length[1] \tag{3}$$

Poniżej zapisane zostało równanie 3 w postaci warunku logicznego w języku C++, gdzie `MatTar` (Ang. Matrix Target) to macierz zawierająca zadaną pozycje i orientacje pomniejszone o wymiary A i B oraz odpowiednio E i F dla zadanej orientacji narzędzia (2).

```
if((sqrt(pow(MatTar[0],2)+pow(MatTar[2],2)))<=(Length[0]+
Length[1]))
```

Jeżeli powyższy warunek jest spełniony należy sprawdzić, ile konfiguracji manipulatora jest w stanie uzyskać zadaną pozycję (Rysunek 6a.) a następnie wybrać która konfiguracja jest odpowiednia (ze względu na lepsze rozłożenie masy program został napisany tak aby tylko dominująca konfiguracja była stosowana).



Rysunek 6. a) Możliwe konfiguracje b) Kąty dla konfiguracji

Jeżeli wszystkie wcześniej wymienione warunki są spełnione i została wybrana odpowiednia konfiguracja można przejść do kolejnego etapu jakim jest wyznaczenie kątów $\Theta_1, \Theta_2, \Theta_3$ (Rysunek 5.) jednak do jego wyznaczenia potrzebne jest wyznaczenie kątów β oraz ψ (Rysunek 6b.). Ze względu na to, że β może być zależna od ćwiartki, w której się znajdują x oraz z do obliczenia β należy użyć dwuargumentowej funkcji Arctg (4) natomiast do obliczenia ψ należy zastosować twierdzenie cosinusów (5).

$$\beta = \arctan(z, x) \tag{4}$$

$$\psi = \arccos\left(\frac{x_1^{3^2} + z_1^{3^x} + length[0]^2 + length[1]^2}{2 \cdot length[0] \cdot \sqrt{x_1^{3^2} + z_1^{3^2}}}\right) \tag{5}$$

Po wyznaczeniu β oraz ψ pozostaje wyznaczenie kątów Θ dla serwomotorów. Jak widać z rysunku 6b kąt Θ_1 jest równy sumie β oraz ψ z czego powstaje zależność (6), z tego samego rysunku można zauważyć, że przeciwprostokątna trójkąta powstałego ze współrzędnych x_1^3 oraz z_1^3 jest równa sumie długości ramion manipulatora z czego wyznaczyć można zależność (7). Do obliczenia ostatniego kąta Θ_3 wykorzystamy zależność dodawania się kątów na płaszczyźnie, zatem suma wszystkich kątów Θ musi być równa orientacji chwytaka (8).

$$\Theta_1 = \beta + \psi \tag{6}$$

$$\Theta_2 = \frac{x_1^{3^2} + z_1^{3^x} - length[0]^2 - length[1]^2}{2 \cdot length[0] \cdot length[1]} \tag{7}$$

$$\Theta_3 = \Theta_1 + \Theta_2 - \phi \tag{8}$$

4. Kod programu

Do poprawnej pracy programu należy zdeklarować komórki pamięci, które przechowywać będą wartości stałe oraz zmienne używane przy obliczeniach (wartości stałe nie muszą być deklarowane jako globalne, ale jest to dobra praktyka, ponieważ jeżeli zmieni się np. jeden z wymiarów manipulatora (Tabela1.), to wystarczy ją zmienić w jednym miejscu na początku programu):

1. `ConstPos[i][j]` – tablica przechowująca stałe wartości, gdzie 'i' to kolejno nr serwomechanizmów (0-ramie, 1-łokieć, 2-nadgarstek, 3-chwytak) natomiast 'j' to zapisane stałe pozycje (0-baza, 1-transport, 2-predkośc w ruchu ręcznym, 3-limit dolny, 4-limit górny, 5-pozycja, przy której dany serwomechanizm przyjmuje wartość kąta względem podłoża równą 0)
2. `Length[k]` – długości ramion, gdzie k to pozycje od C do F z tabeli z wymiarami (Tabela 1.)
3. `Pos[i]` – przechowuje aktualną pozycje serwonapędu gdzie i to kolejno nr serwomechanizmów jw.
4. `CALCULATED_MATRIX_DH[l][m]` – przechowuje ostatnio obliczona macierz zgodną z notacja Denavita-Hartenberga

4.1 Przykład programu na podstawie kodu dla kinematyki prostej

```
float MatRes[6][4][4];          // wyniki kolejnych etapów
float MatHelp[4][4];            // macierz pomocnicza
for(int i=0;i<6;i++)            // MatRes=macierz jednostkowa
    for(int j=0;j<4;j++)
      for(int k=0;k<4;k++)
      {
        if(j==k)
          MatRes [i][j][k]=1;
        else
          MatRes [i][j][k]=0;
      }
for(int i=0;i<5;i++)            // obliczenia wiersza
  for(int h=0;h<4;h++)          //wybór kolumny
    if(MatDH [i][h]!=0.00)    //jeśli zero pomiń obliczenia
    {
```

//W tym miejscu znajdują się macierze notacji Denavita-Hartenberga (Punkt 3.4)

```
      for(int j=0;j<4;j++)      // return do m. pomocniczej
        for(int k=0;k<4;k++)
          MatHelp[j][k]=MatRes[i][j][k];
      for(int j=0;j<4;j++)      //wykonaj obliczenia
      {
        float answer; //zmienna z wynikiem mnozenia
        for(int k=0;k<4;k++)
```

```
      {
        answer=0;
        for(int l=0;l<4;l++) //mnożenie wiersza * kolumne
          answer=MatHelp[j][l]*MatRot[h][l][k]+answer;
        MatRes[i][j][k]=answer;
      }
    }
}
for(int i=0;i<5;i++)          //obliczenie m.wynikowych
{
  for(int j=0;j<4;j++)        //przepisz do pomocniczej
    for(int k=0;k<4;k++)
      MatHelp[j][k]=MatRes[5][j][k];
  for(int j=0;j<4;j++)
  {
    float answer;             //zmienna z wynikiem mnozenia
    for(int k=0;k<4;k++)
    {
      answer=0;
      for(int l=0;l<4;l++)    //mnożenie wiersza * kolumne
        answer=MatHelp[j][l]*MatRes[i][l][k]+answer;
      if(i!=4)                //i różne 4
        MatRes[5][j][k]=answer;
      else                    // i==4 przypisz wyniku
          MatDH_CALCULATED[j][k]=answer;
    }
  }
}
```

4.2 Sterowanie ramieniem

Ze względu na rozbudowany program dla kinematyki prostej i odwrotnej sterowanie manipulatorem ogranicza się do sprawdzenia aktualnej pozycji za pomocą kinematyki prostej i wyświetleniu jej na panelu LCD (Rysunek 7.).

Rysunek 7. Sterowanie manipulatorem

Ruch joystickiem powoduje inkrementacje lub dekrementacje aktualnej pozycji, po czym nowa wartość przekazywana jest do funkcji kinematyki prostej, gdzie obliczane i wystawiane są nowe pozycje dla każdego serwomotoru. Dodatkowo ze względu na brak możliwości sterowania prędkościami serwomechanizmów, czego skutkiem jest

niekontrolowanie trajektorii przejazdu od aktualnej pozycji do zadanej zaimplementowana została funkcja do ruchów automatycznych, która przy większych przejazdach dzieli różnice w pozycjach na kilka części i stopniowo przekazuje do serwomechanizmów nowe pozycje coraz to bliżej zadanej.

5. Badanie poprawności działania

Ze względu na brak możliwości zastosowania sprzężenia zwrotnego nie można stworzyć funkcji, która kontrolowała by pozycje, jednak z przeprowadzonych pomiarów wynika, że błąd dokładności pozycjonowania po osi x wynosi max 2 mm, natomiast po osi z zmienia się w zależności od pozycji co jest spowodowane nie tylko brakiem precyzji serw, ale również odkształceniom sprężystym konstrukcji wykonanej ze stopu aluminium. Jednak błąd wysokości nie przekracza 5 mm osiągając swoje maximum w najdalej i najwyżej oddalonej pozycji, w miejscach, których manipulator łapie i odkłada przedmioty błąd ten nie przekracza 2 mm

6. Podsumowanie

W projekcie udało się wykonać większość z zakładanych celów. Dużym problemem okazał się brak sprężenia zwrotnego oraz mała sztywność konstrukcji wynikająca z ograniczenia jej wagi, przez co nie możemy zakładać, że uzyskamy wysoką dokładność pozycjonowania. Wprowadzenie dodatkowych enkoderów lub potencjometrów w postaci sprężenia zwrotnego powinno zwiększyć dokładność pozycjonowania wynikającą z niedokładności serwomechanizmów, ale równocześnie pogorszy dokładność wynikającą ze sztywności ze względu na dodatkową masę ramienia. Sam kod programu dla kinematyki prostej nie jest skomplikowany, ponieważ opiera się na obliczeniach macierzy zgodnie z notacją Denavita-Hartenberga. Powstały program można z powodzeniem zastosować do podobnego typu manipulatorów modyfikując orientację serwomechanizmów, długości ramion, nawet dołożenie kolejnego serwonapędu nie spowoduje dużych zmian programu ograniczając się do zmian w notacji. W przypadku kinematyki odwrotnej program można zastosować pod warunkiem, że zmienią się jedynie długości ramion lub orientacja serwomechanizmów, natomiast jakakolwiek zmiana w notacji wiąże się z całkowitą zmianą programu.

LITERATURA

1. HUGES C. HUGES T.: Programowanie robotów- sterowanie pracą robotów Autonomicznych, HELION, 2017, 178-179.
2. JOHN J. CRAIG: Wprowadzenie do robotyki- mechanika i sterowanie, wydawnictwo naukowo-techniczne, 1995, 54,85-92,132-144
3. Botland *http://www.Botland.com.pl* – Tabela 2. Zestawienie części wchodzących w skład manipulatora 25.10.2018
4. Wikipedia *http://www.pl.wikipedia.org/wiki/Arduino*, 25.10.2018
5. Arduino *http://wwwarduino.cc/en/Referenc/Servo*, 25.10.2018
6. Arduino *http://wwwarduino.cc/en/Referenc/map*, 25.10.2018