WYDZIAŁ MECHANICZNY POLITECHNIKI BIAŁOSTOCKIEJ

PROJEKT Z PRZEDMIOTU

Systemy Autonomiczne

kod przedmiotu: MYAR2S22007M

Temat: Projektowanie układu sterowania i nawigacji śmigłowca wielowirnikowego jako systemu autonomicznego

Autor: Ostaszewicz Dawid

Kierunek: Automatyka i Robotyka, II stopień

Specjalność: Informatyzacja Przemysłu

Semestr II

Prowadzący: dr inż. Leszek Ambroziak

Weryfikacja efektów uczenia się:

EK1: ...

EK2: ...

EK3: ...

EK4: ...

EK5: ...

EK6: ...

EK7: ...

EK8: ...

Uwagi prowadzącego:

ocena: ...

Cel i zakres projektu

Celem projektu jest zapoznanie się z systemami autonomicznymi na przykładzie śmigłowca wielowirnikowego. Projekt obejmuje poznanie modelu śmigłowca, autonomi lotu oraz funkcji i algorytmów niezbędnych do poprawnego planowania nakazanej linii drogi; generowaniem drogi i sposobami aktywnej korelacji.

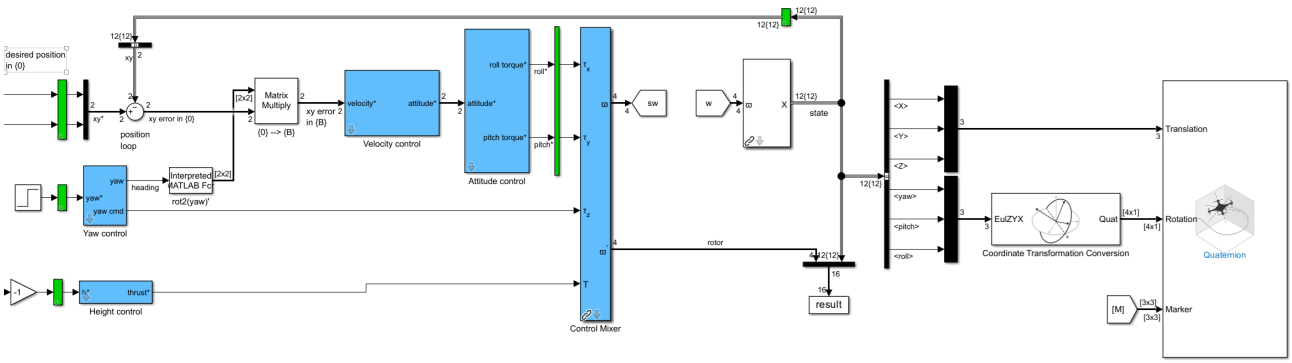
Zadanie nr 1

Treść

Zapoznaj się z ukladem symulacyjnym śmigłowca czterowirnikowego; dostępnym w bibliotece RMVT. Dodaj do układu symulacyjnego wyświetlanie parametrów lotu śmigłowca (porównaj wartości zadane parametrów nawigacyjnych z aktualnie mierzonymi). Zmodyfikuj model dyniamiki śmigłowca zgodnie z parametrami zawartymi w tabeli. Sprawdź działanie układu sterowania dla zmodyfikowanych parametrów modelu, popraw nastawy pętli regulacyjnych i sposób stabilizacji śmigłowca. Poeksperymentuj z różnymi wzmocnieniami w układzie sterowania. Co się stanie jeśli zmniejszysz tłumienie lub zupełnie je usuniesz. Usuń kompensator siły grawitacji i poeksperymentuj z dużą wartością wzmocnienia w układzie sterowania wysokością lub z innym typem regulatora.

Opracowanie

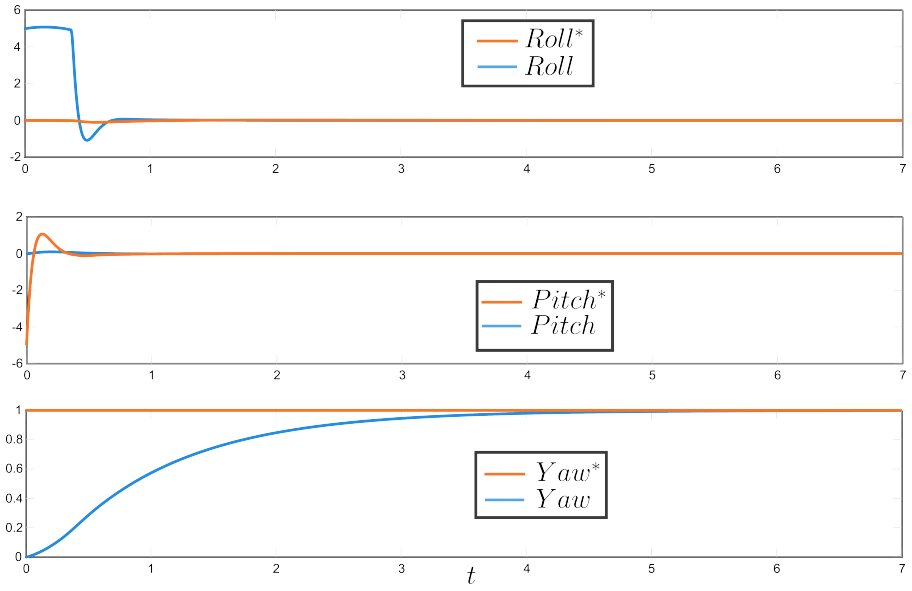
Proces strojenia modelu rozpoczął się od wprawdzenia parametrów modelu w odpowiednie miejsce, co wprowadziło różnice między projektami. Dynamika śmigłowca zależy głównie od jego masy i danych tensora bezwładności. Żeby podjeść do strojenia regulatorów, na wejścia obiektów zostały podane sygnały skoku jednostkowego. Zgodnie z Rysunkiem 1, układ regulacji



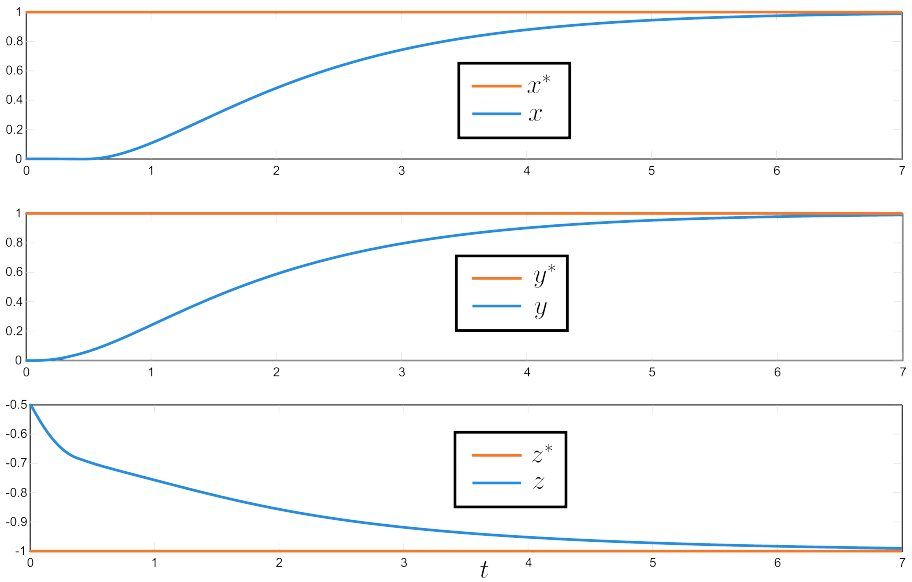
Rysunek 1: Schemat obiektu

składa się z kaskadowo połączonych układadów regulacji. W takim wypadku strojenie powinno

się rozpoczynać od wewnętrznych pętli, czyli "attitude control". Układ regulacji zawiera pętle regulacji: położenia, prędkości, wysokości. Prędkość jest zależna od położenia kątowego śmigłowca. Regulując położenie układ zadaje prędkości przez sterowanie kątami roll i pitch. Jak już zostało wspomniane najpierw były strojone regulatory znajdujące w najbardziej zagnieżdżonych pętlach, przemieszczając się od obszarów nadrzędnych układów regulacji automatycznej. Efektem działania powinna być trajektoria całkowicie aperiodyczna, ponieważ jakiekolwiek przeregulowanie może być niebezpieczne dla śmigłowca. W procesie strojenia zostały uzyskane odpowiedzi z Rysunku 2 i 3. Do układu regulacji wysokości kluczowe było odpowiednie usta-



Rysunek 2: Odpowiedzi układu po strojeniu - XYZ



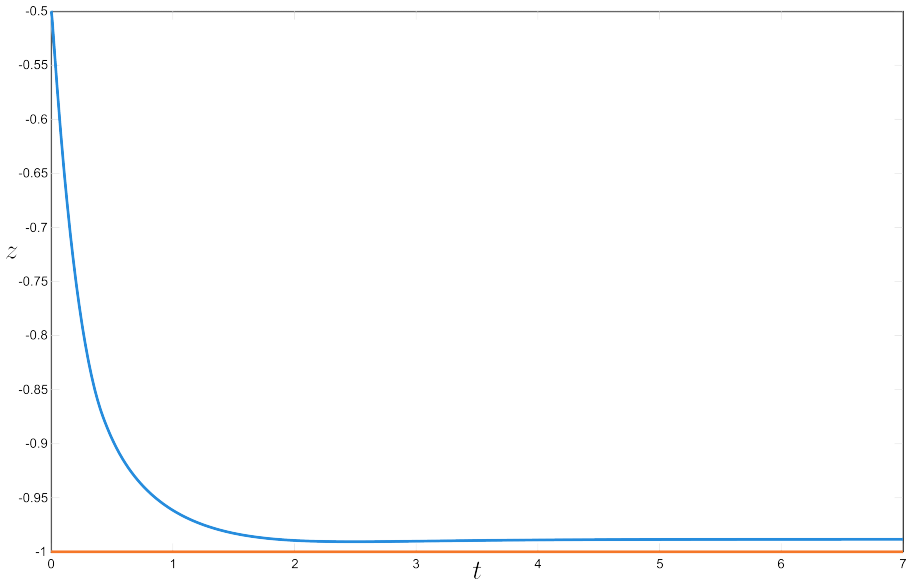
Rysunek 3: Odpowiedzi układu po strojeniu - RLY

wienie zmiennej "thrust feedforward", ponieważ układ powienien odpowiednio kompensować wpływ grawitacji. Nastawy regulatorów zostały zestawione w tabeli 1.

Tabela 1: Nastawy

Controller	P	D	Thrust feedforward
Attitude	50	0.08	-
Velocity	0.1	2	-
Yaw	200	1	-
Height	500	2	58
Height	5000	0.35	0

Jeśli chodzi o zmienną "thrust feedforward", można odpowiednio nastroić regulator wysokości bez niej, jednak wiąże się to z efektami, które mogą nie być wygodne, lub możliwe do realizacji. Rysunek 4, przedstawia działanie nastaw dla regulatora wysokości z zerową nastawą "Thrust feedforward"z tabeli 1.



Rysunek 4: Odpowiedź układu dla strojenia bez kompensatora grawitacji

Trzeba zauważyć, że przy takim strojeniu wielkości w torze sygnałów sterowania będą miały bardzo duże wartości na wyjściach regulatorów, co w większości przypadków może nie być możliwe w realizacji. Poza tym układ posiada jeszcze mały uchyb.

Podsumowanie

Strojenie regulatorów należy rozpoczynać od wewnętrznych pętli regulacji; przechodząć do nadrzędnych. Można nastroić regulator wysokości bez kompensatora wysokości, jednak jego funkcjonowanie może okazać się niedogodne w realizacji praktycznej.

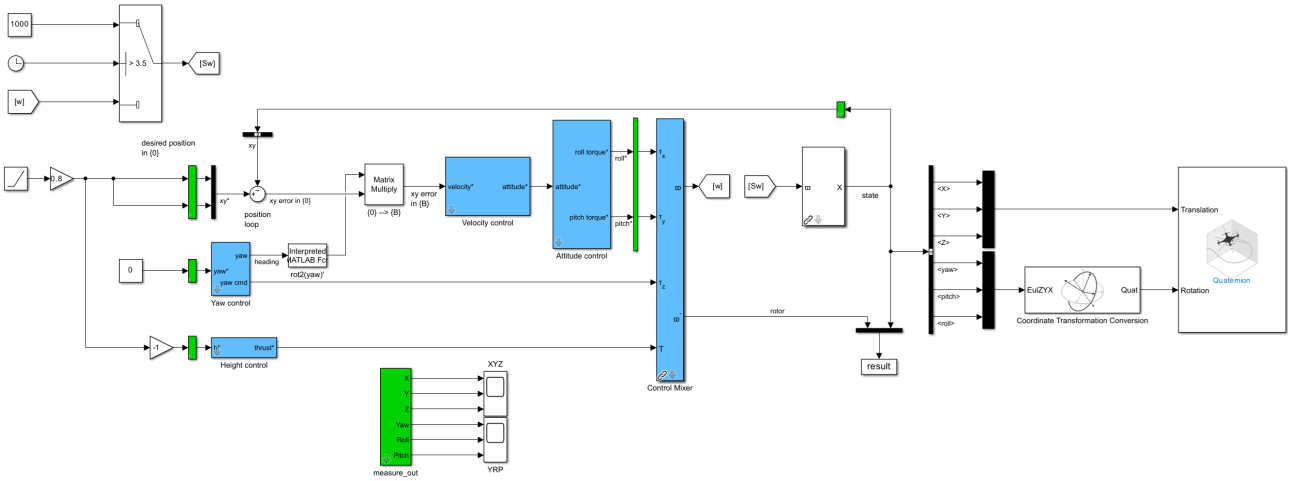
Zadanie nr 2

Treść

Opracuj funkcję realizującą ruch balistyczny śmigłowca. Niech quadrotor wystartuje pod kątem 45 stopni do poziomu, następnie wyzeruj cały ciąg. Sprawdź uzyskaną trajektorię śmigłowca. Spróbuj opracować funkcję realizującą lot balistyczny do zadanego punktu na powierzchni.

Opracowanie

Istnieje wiele podejść do problemu realizacji ruchu balistycznego śmigłowca wielowirnikowego. Pierwszą z nich jest sterowanie śmigłowcem do określonego punktu przestrzeni; żeby następnie, jeszcze w trakcie przyspieszenia wyłączyć lub ograniczyć ciąg. W efekcie takiego zabiegu trajektoria powinna być zbliżona do balistycznej. Poprzednie rozwiązanie jest problematyczne pod względem uchwycenia chwili, posiadania przyspieszenia. Jeśli ciąg wyłączy się zbyt późno, przyspieszenie obiektu w osiach XY może być zbyt małe, żeby utworzyć dobrą trajektorię balistyczną. Prostrzym podejściem będzie zadawanie pozycji liniowo narastającej w czasie. W wyniku pracy z regulatorami o pojedyńczym całkowaniu, śmigłowiec, co prawda nie będzie nadążał za wartością zadaną; będzie posiadał stały uchyb, co jest wadą zastosowanych regulatorów o jednokrotnym całkowaniu, jednak w każdym razie obiekt w określonej sytuacji powinien posiadać stałe przyspieszenie. Jeśli ciąg zostanie ograniczony, obiekt będzie się zniżał. Trajektoria będzie zbliżona do balistycznej. Rysunek 5 przedstawia schemat Simulinka do realizowanego zadania.



Rysunek 5: Schemat realizacji trajektorii balistycznej

Ciąg generowany przez śmigła będzie ograniczany, w różnych odstępach czasu. Badanie odległości punktu lądowania od czasów wyłączenia, umożliwi wyprowadzenie zależności czasu ograniczenia ciągu od odległości; ostatecznie taki zabieg umożliwi stworzenie realacji, która

pozwoli zadawać punkt lądowania, Rysunek ... Znając realcję czasu wyłączenia od odległości na jakiej ląduje dron (można obliczyć z eq...).

$$s = \sqrt{x^2 + y^2} \tag{1}$$

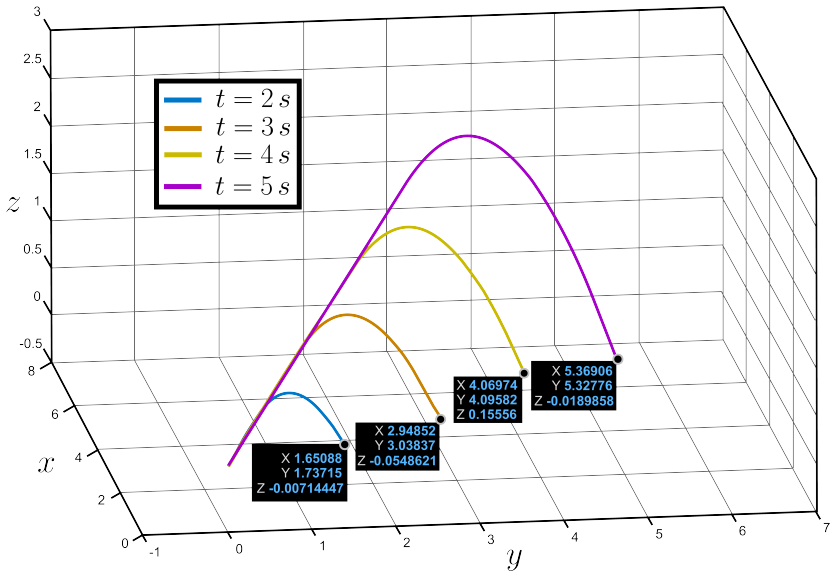
Można z pomocą Rysunku 7 aproksymować odległość punktu lądowania. W takim kontekście zadanie punktu lądowania powinno odpowiadać wektorowi przemieszczenia w płaszczyźnie XY i relacji czasu ograniczenia ciągu. Pierwszą rzeczą będzie zadanie określenia kąta wektora przemieszczenia w płaszczyźnie XY, eq (2,3)

$$k_x = \sin \arctan \frac{x^*}{y^*} \tag{2}$$

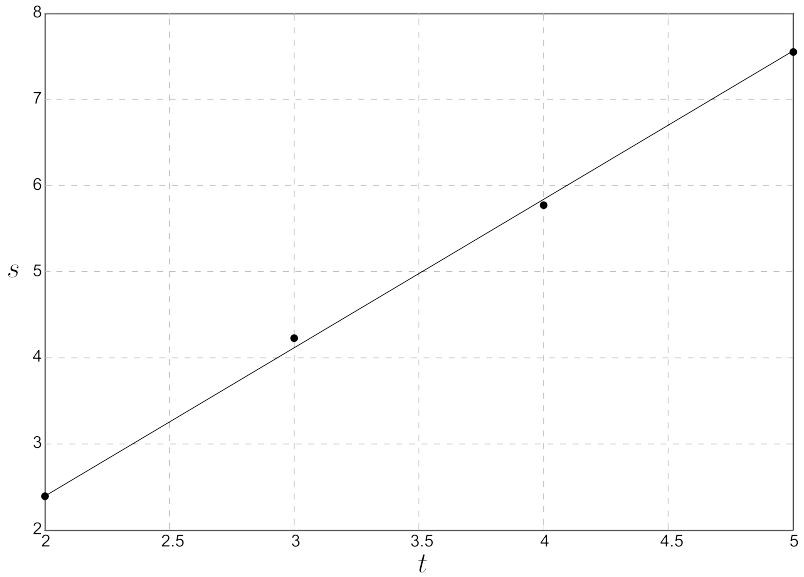
$$k_y = \cos \arctan \frac{x^*}{y^*} \tag{3}$$

Natomiast moduł wektora przemieszczenia określa równanie aproksymacji drogi eq (4))

$$t = \sqrt{x^2 + y^2} - 0.13 \tag{4}$$



Rysunek 6: Otrzymane trajektorie balistyczne



Rysunek 7: Odległość od punktu startowego

Wnioski

Wykonane badania pozwalają określić, zależności na wykonanie balistycznego lotu śmigłowca wielowirnikowego. Zastosowanie liniowo narastających wartości zadanych umożliwia zadanie stałego przyspieszenia obiektu, natomiast ograniczenie ciągu wymusza ruch po trajektorii balistycznej. Wykonane badanie wraz z aproksymacją umożliwia wyprowadzenie równań na lot balistyczny w przybliżeniu do punktu zadanego.

Zadanie nr 3

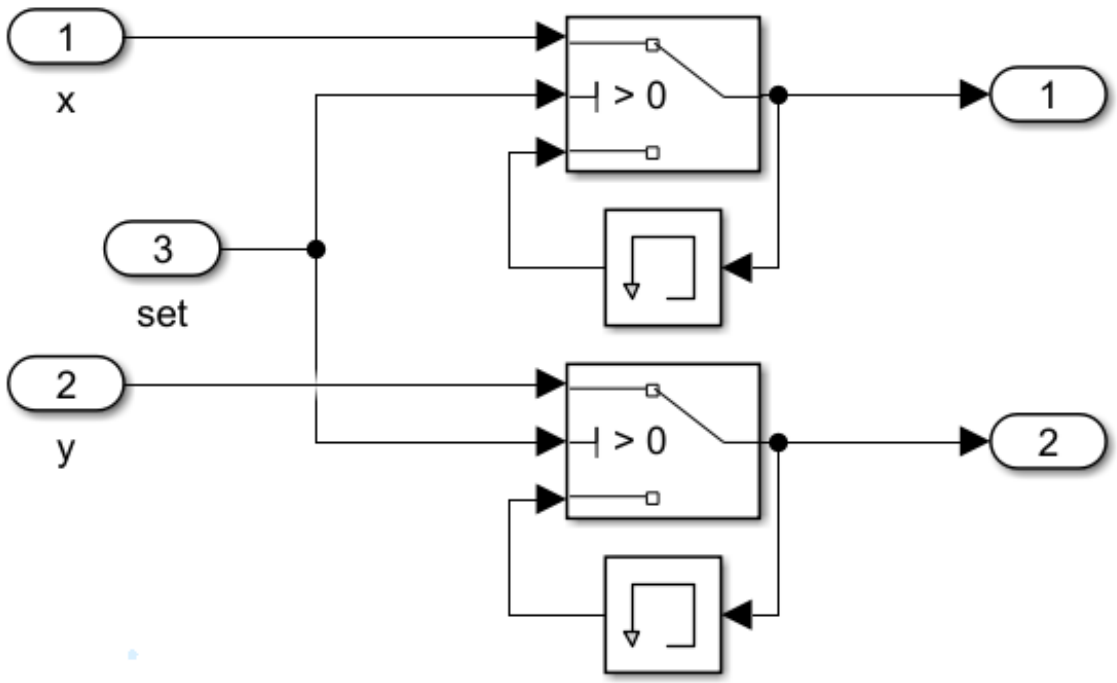
Treść

Opracuj funkcję realizującą automatyczne lądowanie śmigłowca w oparciu o dostępne sygnały pomiarowe (lądowanie może być aktywowane w dowolnym momencie, ze wskazaniem miejsca laowania, po aktywowaniu funkcji automatycznego lądowania śmigłowiec przerywa wcześniej realizowany scenariusz, podąża do punktu lądowania, przechodzi do zawisu, po czym łagodnie ląduje)

Opracowanie

Pierwszą fazą lądowania jest przerwanie czynności, które obecnie są realizowane przez drona. Wartości zadane w osiach x i y będą zapamiętywane przy użyciu bloku z Rysunku 8

Funkcja zadaje zapamiętane wartości, a dron je utrzymuje. W tej fazie również będzie dochodziło do zniżania pułapu lotu, do wartości około 30 cm przy użyciu przełącznika. Założenie drugiej fazy lądowania: jeśli dron osiągnie pułap mniejszy niż 35 cm, zacznie stopniowo ograni-



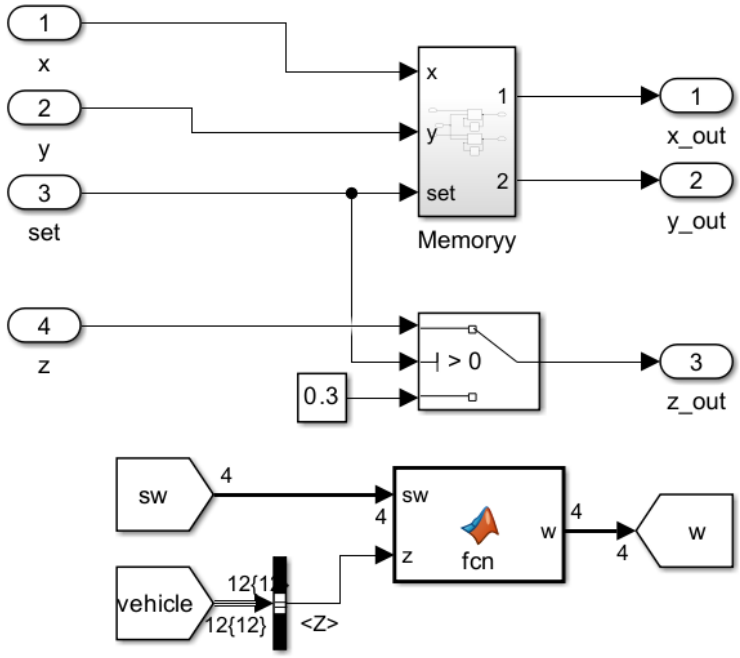
Rysunek 8: Funkcja pamięci stanu trajektorii w płaszczyźnie xy

czać ciąg silników. Natomiast w trzeciej fazie; po osiągnięciu 20 cm silniki zostają wyłączone, żeby ograniczyć turbulencje związane z oddziaływaniem podłoża. Wnętrze bloku automatycznego lądowania zestawia Rysunek 9

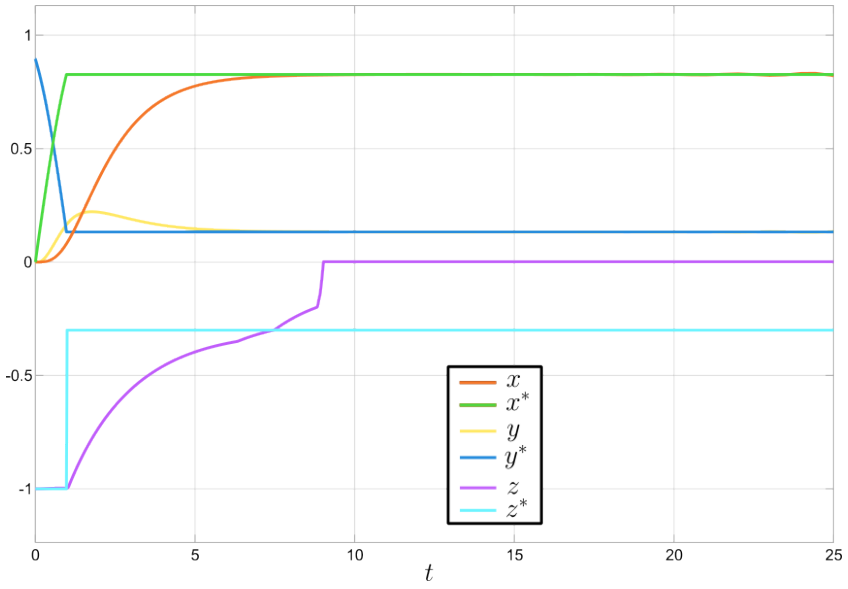
Wejście set jest związane z działaniem przycisku. W trakcie lotu można przełączyć przełącznik, który rozpocznie sekwencję automatycznego lądowania. Wnętrze bloku matlab function:

```
function w = fcn(sw,z)


if (z > -0.35) && (z < -0.30)
    w = 0.8*sw;
elseif (z >= -0.30) && (z < -0.20)
    w = 0.6*sw;
elseif (z >= -0.20)
    w = 0.1*sw;
else
    w = sw;
end
```



Rysunek 9: Automatyczne lądowanie - blok



Rysunek 10: Automatyczne lądowanie - sygnały

Podsumowanie

Po czasie około sekundy przełącznik automatycznego lądowania został włączony, zmienił wartość wejścia "set"bloku automatycznego lądowania. Następnie zadane trajektorie zostały wstrzymane na jednym punkcie w płaszczyźnie xy; następowało zniżenie pułapu lotu, następnie układ przechodzi do fazy kaskadowego wyłączania ciągu silników i przechodzi do sekwencji wielopoziomowego lądowania. W ostatniej fazie silniki są wyłączane i dron ląduje na ziemi.

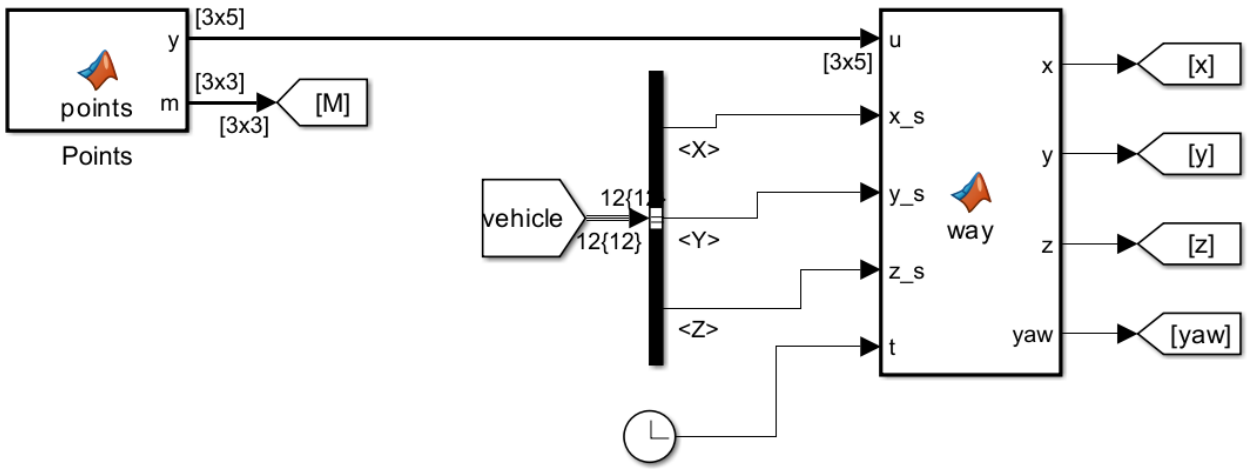
Zadanie nr 4

Treść

Opracuj algorytm realizujący lot śmigłowca czterowirnikowego po zadanych punktach drogi. Zadane punkty powinny być definiowane przed rozpoczęciem lotu i przechowywane w macierzy. Dodatkowo powinny być zaznaczone na wizualizacji lotu śmigłowca.

Opracowanie

Rysunek 11 przedstawia schemat realizacji lotu śmigłowca po nakazanych punktach drogi.



Rysunek 11: Schemat algorytmu śledzenia waypointów

Funkcja "points"przechowuje macierz współrzędnych kolejnych punktów, wraz z kątem yaw, który ma osiągnąć dron podczas lotu do nakazanego punktu oraz czas jaki dron ma przebywać w nakazanej sferze drogi.

```
    function [y, m] = points()
persistent P T;
```

```
if isempty(P)
    % ---- x --- y --- z --- yaw --- time----
    P = [  1,    1,    1,     1,      0.5; ...
           2,    1,    3,     0.5,    0.5; ...
          -1,    2,    2,    -0.5,    0.5];
end

if isempty(T)
    T = [  1,    1,    -1; ...
           2,    1,    -3; ...
           -1,   2,    -2];
end

y = P;
m = T;
```

Wartości są przypisywane do odpowiednich indeksów zgodnie z opisem zawartym w funkcji. Macierz T przechowuje współrzędne punktow, które są potrzebne do ich wyświetlania na wizualizacji UAV.

Funkcja "way"realizuje algorytm śledzenia waypointów.

```
    function [x, y, z, yaw] = way(u, x_s, y_s, z_s, t)

persistent i
persistent start_time

if isempty(i)
    i = 1;
end

if isempty(start_time)
    start_time = t;
end
```

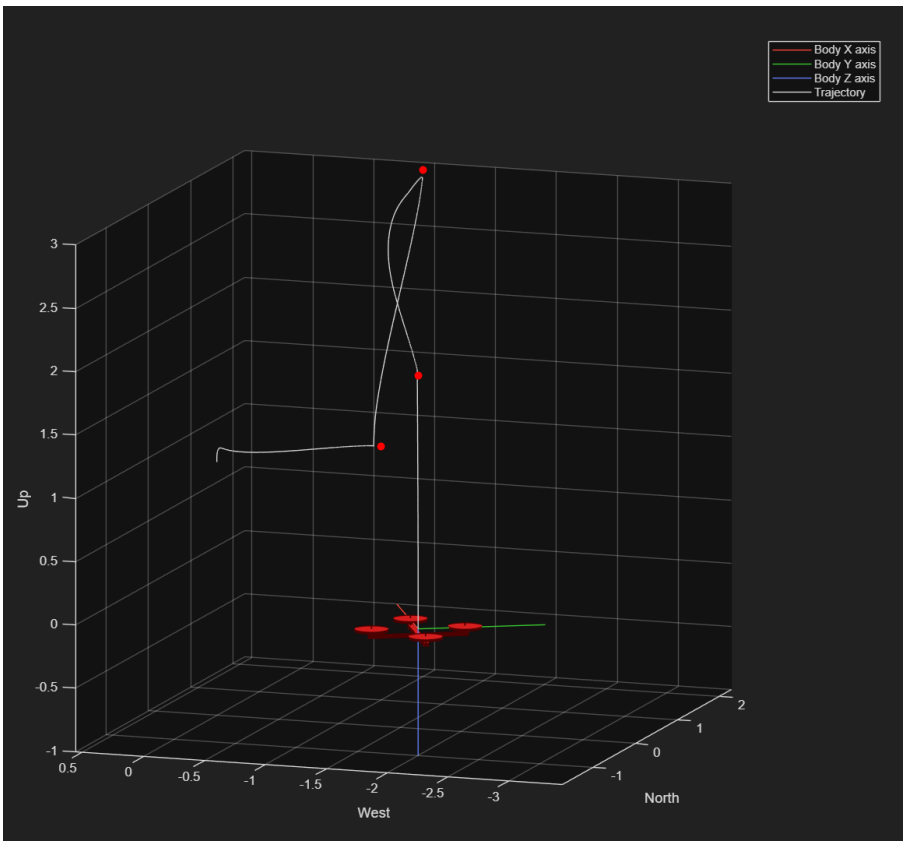
```
        x = u(i, 1);
        y = u(i, 2);
        z = u(i, 3);
        yaw = u(i, 4);
        w_time = u(i,5);

        distance = sqrt((x - x_s)^2 + (y - y_s)^2 + (z + z_s)^2);
        disp(sprintf('Distance: %.3f', distance))
        if distance < 0.1
            elapsed_time = t - start_time;
            disp(sprintf('Start time: %.3f', start_time))
            disp(sprintf('Elapsed time: %.3f', elapsed_time))
            if elapsed_time >= w_time
                i = i+1;
                if i >= height(u)
                    i  = height(u);
                end
                disp("waypoint")

            end
        else
            start_time = t;
        end

end
```

Funkcja przyjmuje argumenty, które są niezbędne do wykonanania algorytmu. Na początku definiowana jest zmienna i, która przyjmuje wartość początkową równa jeden. Następnie układ zadaje współrzędne i kąt, yaw; jaki ma osiągnąć dron dolatując do punktu drogi; jako pierwszy rząd macierzy. Później podczas próbkowania symulacji na bieżąco obliczany jest dystans do obecnie zadanego waypointu. Jeśli wayipoint nie został osiągnięty czas start_time jest stale aktualizowany. Po przekroczeniu waypointu start_time nie jest zmieniany i oblicza się czas od wejścia w sferę nakazanego punktu drogi. Jeśli czas przebywania w waypoincie zadany w tablicy zostanie osiągnięty zmienna i pzechodzi do następnej iteracji w tablicy i tak, aż do wykonania wszystkich zdanych punktów. Żeby nie wychodzić poza zakres tablicy z nakazanymi punktami dorgi jest ustawione ograniczenie, które sprawia, że jeśli zakres zostanie przekroczony zmienna i



Rysunek 12: Śledzenie waypointów

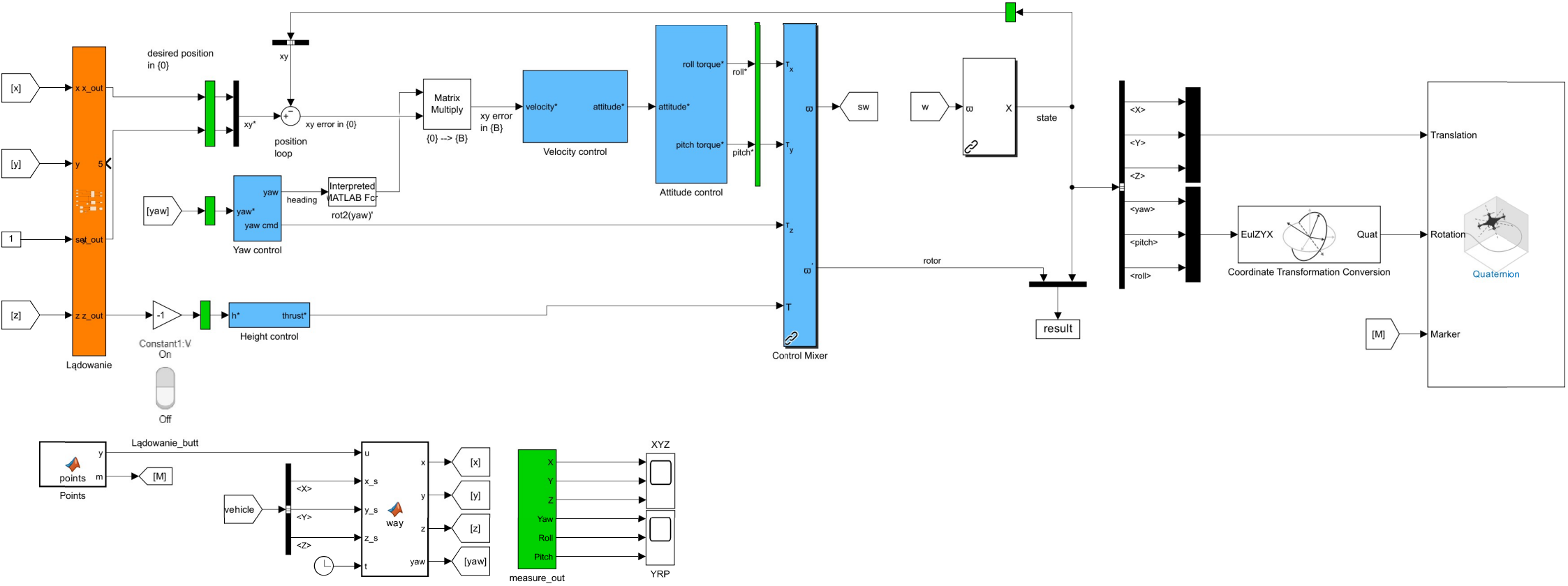
przybiera wartość ostatniego elementu. Po wykonaniu całej drogi można np. przełączyć przycisk i rozpocząć sekwencję automatycznego lądowania; Rysunek 12.

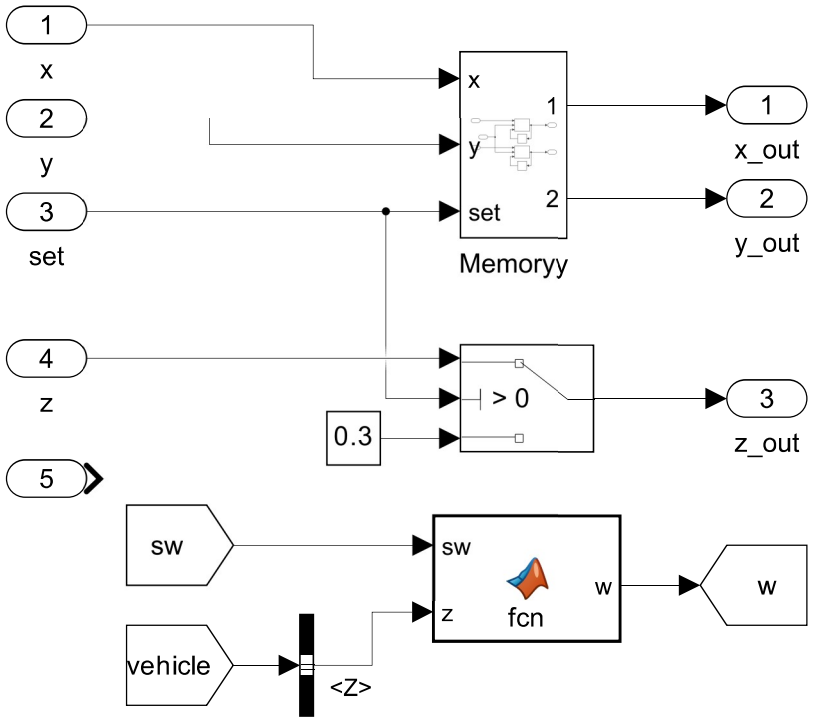
Podsumowanie projektu

Zaprojektowany algorytm, który umożliwia zadawanie śmigłowcowi punktów drogi, otwiera możliwości implementacji i dalszego rozwoju środowiska. Przykładowo, można zastosować algorytmy planowania trasy do generowania macierzy zadanych punktów. W trakcie realizacji projektu można także zwrócić uwagę na niezbędne czujniki, które są kluczowe w takich aplikacjach. Aby dron mógł poprawnie realizować zadane algorytmy, konieczne są m.in. czujniki GPS, kamery oraz IMU, często w konfiguracjach redundantnych i komplementarnych. Fundamentem sterowania systemami autonomicznymi jest dostęp do pokładowych przyrządów pomiarowych, które zapewniają dokładność wymaganą dla danego zastosowania.

Moduł automatycznego lądowania wymaga praktycznej weryfikacji, gdyż symulowanie zjawisk, takich jak turbulencje spowodowane przez reakcję podłoża, jest trudne. Ze względu na ten problem konieczne jest dostrojenie algorytmu lądowania do warunków środowiska, szczególnie w zakresie odpowiedniego sterowania ciągiem silników na różnych wysokościach i fazach lądowania, by precyzyjnie określić moment ich wyłączenia.

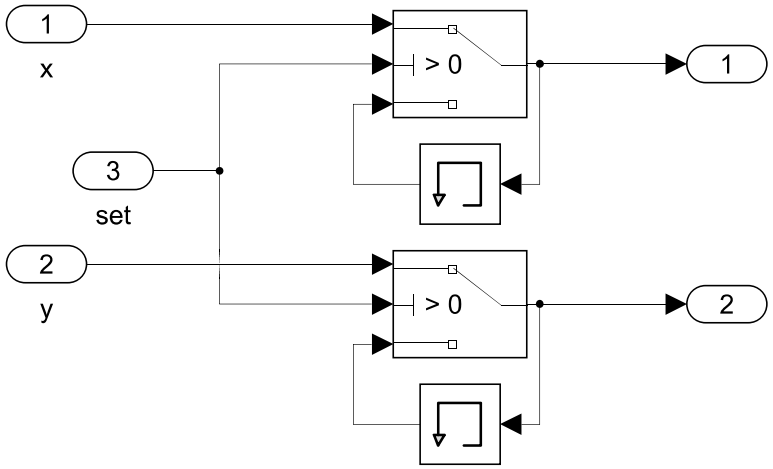
Ogólnie rzecz biorąc, projekt został opracowany w środowisku pozbawionym wpływu warunków zewnętrznych, co znacznie ułatwiło jego realizację. W warunkach rzeczywistych, np. przy określonym poziomie wiatru i zakłóceniach czujników, dostosowanie regulatorów byłoby bardziej wymagające. W praktycznych zastosowaniach korzystne mogłoby okazać się wdrożenie zaawansowanych algorytmów regulacji, aby minimalizować wpływ zakłóceń na działanie układu sterowania.


[x]
[y]
1
[z]
x x_out
y
5
sel
z z_out
Lądowanie
desired position
in {0}
xy
xy*
position
loop
xy error in {0}
Matrix
Multiply
{0} --> {B}
xy error
in {B}
velocity*
attitude*
Velocity control
roll torque*
attitude*
pitch torque*
Attitude control
roll*
pitch*
[yaw]
yaw
yaw*
yaw cmd
heading
Interpreted
MATLAB Fcn
rot2(yaw)'
Yaw control
-1
Constant1:V
On
Off
h*
thrust*
Height control
T_x
T_y
T_z
T
Control Mixer
sw
w
X
state
rotor
result
<X>
<Y>
<Z>
<yaw>
<pitch>
<roll>
EulZYX
Quat
Coordinate Transformation Conversion
Translation
Rotation
Quaternion
[M]
Marker
points
m
Points
Lądowanie_butt
vehicle
x_s
y_s
z_s
t
way
[x]
[y]
[z]
[yaw]
X
Y
Z
Yaw
Roll
Pitch
measure_out
XYZ
YRP


1
x
2
y
3
set
x
y
set
1
2
Memoryy
1
x_out
2
y_out
4
z
5
0.3
> 0
3
z_out
sw
vehicle
<Z>
sw
z
fcn
w
w

```
function w = fcn(sw,z)

    if (z > -0.35) && (z < -0.30)
        w = 0.8*sw;
    elseif (z >= -0.30) && (z < -0.20)
        w = 0.6*sw;
    elseif (z >= -0.20)
        w = 0.1*sw;
    else
        w = sw;
    end
```


1
x
3
set
2
y
> 0
> 0
1
2

```
function [x, y, z, yaw] = way(u, x_s, y_s, z_s, t)

persistent i
persistent start_time

if isempty(i)
    i = 1;
end

if isempty(start_time)
    start_time = t;
end


        x = u(i, 1);
        y = u(i, 2);
        z = u(i, 3);
        yaw = u(i, 4);
        w_time = u(i,5);

        distance = sqrt((x - x_s)^2 + (y - y_s)^2 + (z + z_s)^2);
        disp(sprintf('Distance: %.3f', distance))
        if distance < 0.1
            elapsed_time = t - start_time;
            disp(sprintf('Start time: %.3f', start_time))
            disp(sprintf('Elapsed time: %.3f', elapsed_time))
            if elapsed_time >= w_time
                i = i+1;
                if i >= height(u)
                    i  = height(u);
                end
                disp("waypoint")

            end
        else
            start_time = t;
        end

end
```

```
function [y, m] = points()
persistent P T;

if isempty(P)
    % ---- x --- y --- z --- yaw --- time----
    P = [  1,    1,    1,     1,      0.5; ...
           2,    1,    3,     0.5,    0.5; ...
          -1,    2,    2,    -0.5,    0.5];
end

if isempty(T)
    T = [  1,    1,    -1; ...
           2,    1,    -3; ...
           -1,   2,    -2];
end

y = P;
m = T;
```