Flight Autonomy

Wygenerowano przez Doxygen 1.9.3

1 Indeks klas

1.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

2 Indeks plików

2.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

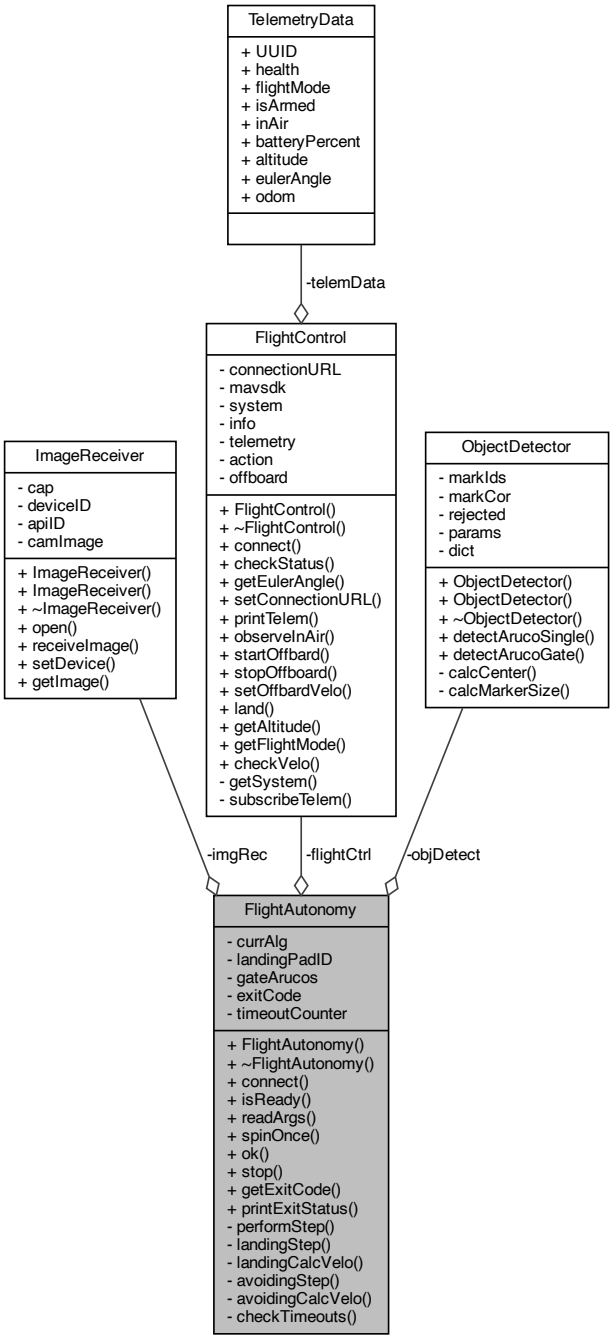
3 Dokumentacja klas

3.1 Dokumentacja klasy FlightAutonomy

Klasa odpowiedzialna za kompleksową obsługę autonomii lotu bazującej na analizie wizyjnej. W klasie realizowany jest algorytm analizujący obraz odbierany z kamery i w przypadku wykrycia przeszkody wykonanie odpowiedniej reakcji w postaci zmiany trajektorii lotu maszyny. Wykorzystywany jest protokół MavLink do dwukierunkowej komunikacji z autopilotem i przesyłania komend sterujących.

```
#include <FlightAutonomy.h>
```

Diagram współpracy dla FlightAutonomy:



Metody publiczne

- FlightAutonomy ()

  Konstruuje obiekt FlightAutonomy inicjalizując pola domyślnymi wartościami.
- ~FlightAutonomy ()

  Destuktor. W trybie debugowania niszczący okno OpenCV.
- bool connect ()

Inicjalizuje połączenie z autopilotem poprzez protokół MavLink oraz uruchamia tryb offboard.

- bool isReady ()

  Sprawdza czy maszyna jest gotowa do wykonywania algorytmu.
- bool readArgs (const int argc, char ∗∗argv)

  Wczytuje przekazane do programu parametry.
- bool spinOnce ()

  Wykonuje pojedynczy krok algorytmu. Powinna być wywoływana jednokrotnie w trakcie każdego obrotu pętli głównej programu.
- bool ok ()

  Sprawdza czy wszystkie elementy działają prawidłowo i czy nie pojawił się warunek wyjścia.
- bool stop ()

  Kończy działanie algorytmu i wyłącza tryb offboard.
- int getExitCode ()

  Zwraca wartość kodu wyjścia.
- void printExitStatus ()

  Wyświetla status wyjścia według kodu w zmiennej exitCode.

Metody prywatne

- int performStep (cv::Mat &img)

  Wykonuje pojedyncza iterację dla wybranego algorytmu.
- bool landingStep (cv::Mat &img)

  Wykonuje jeden krok algorytmu lądowania.
- mavsdk::Offboard::VelocityBodyYawspeed landingCalcVelo (cv::Mat &img, cv::Point2f arucoPosition)

  Oblicza prędkości danej iteracji algorytmu lądowania.
- bool avoidingStep (cv::Mat &img)

  Wykonuje jeden krok algorytmu lądowania.
- mavsdk::Offboard::VelocityBodyYawspeed avoidingCalcVelo (cv::Mat &img, cv::Point2f gatePosition, float angle)

  Oblicza prędkości danej iteracji algorytmu przelotu przez bramkę, potrzebne do przelotu przez bramkę.
- void checkTimeouts ()

  Sprawdza czy nastąpiło przekroczenie zdefiniowanych czasów maksymalnych.

Atrybuty prywatne

- ImageReceiver imgRec

  Odbiornik obrazu z kamery.
- FlightControl flightCtrl

  Kontrola lotu maszyny.
- ObjectDetector objDetect

  Wykrywacz obiektów na obrazie z kamery.
- Algorithms currAlg

  Obecnie wykonywany algorytm.
- int landingPadID = 68

  ID znacznika Aruco lądowiska.
- int gateArucos [4] = {10, 11, 12, 13}

  ID znaczników umieszczonych na bramce zgodnie z ruchem wskazówek zegara.
- int exitCode

  Kod wyjścia z systemu: 0 - kontynuuj pracę, 1 - wyjście normalne, 2 - niepoprawna wysokość, 3 - nie wykryto znaczników przez zadany czas.
- std::chrono::steady_clock::time_point timeoutCounter

  Odlicza czas do automatycznego wyjścia z programu w przypadku nie wykrywania znaczników.

3.1.1 Opis szczegółowy

Klasa odpowiedzialna za kompleksową obsługę autonomii lotu bazującej na analizie wizyjnej. W klasie realizowany jest algorytm analizujący obraz odbierany z kamery i w przypadku wykrycia przeszkody wykonanie odpowiedniej reakcji w postaci zmiany trajektorii lotu maszyny. Wykorzystywany jest protokół MavLink do dwukierunkowej komunikacji z autopilotem i przesyłania komend sterujących.

3.1.2 Dokumentacja konstruktora i destruktora

3.1.2.1 FlightAutonomy() `FlightAutonomy::FlightAutonomy ( )`

Konstruuje obiekt FlightAutonomy inicjalizując pola domyślnymi wartościami.

3.1.2.2 ~FlightAutonomy() `FlightAutonomy::~FlightAutonomy ( )`

Destuktor. W trybie debugowania niszczący okno OpenCV.

3.1.3 Dokumentacja funkcji składowych

3.1.3.1 avoidingCalcVelo() `mavsdk::Offboard::VelocityBodyYawspeed FlightAutonomy::avoidingCalcVelo (`

```
            cv::Mat & img,
            cv::Point2f gatePosition,
            float angle )  [private]
```

Oblicza prędkości danej iteracji algorytmu przelotu przez bramkę, potrzebne do przelotu przez bramkę.

Parametry

img	Przetwarzana ramka obrazu.
gatePosition	Pozycja bramki na obrazie.
angle	Kąt pochylenia bramki.

Zwraca

mavsdk::Offboard::VelocityBodyYawspeed

3.1.3.2 avoidingStep() bool FlightAutonomy::avoidingStep (cv::Mat & *img*) [private]

Wykonuje jeden krok algorytmu lądowania.

Parametry

img	Najnowsza ramka obrazu.

Zwraca

true Poprawnie wykonano krok algorytmu.
false Błąd podczas wykonywania kroku algorytmu.

3.1.3.3 checkTimeouts() `void FlightAutonomy::checkTimeouts ( ) [private]`

Sprawdza czy nastąpiło przekroczenie zdefiniowanych czasów maksymalnych.

3.1.3.4 connect() `bool FlightAutonomy::connect ( )`

Inicjalizuje połączenie z autopilotem poprzez protokół MavLink oraz uruchamia tryb offboard.

Zwraca

true Połączenie zostało nawiązane.
false Wystąpił błąd podczas nawiązywania połączenia.

3.1.3.5 getExitCode() `int FlightAutonomy::getExitCode ( )`

Zwraca wartość kodu wyjścia.

Zwraca

int Wartość kodu wyjścia.

3.1.3.6 isReady() `bool FlightAutonomy::isReady ( )`

Sprawdza czy maszyna jest gotowa do wykonywania algorytmu.

Zwraca

true Maszyna jest gotowa.
false Maszyna nie jest gotowa.

3.1.3.7 landingCalcVelo() `mavsdk::Offboard::VelocityBodyYawspeed FlightAutonomy::landingCalcVelo (`

```
            cv::Mat & img,
            cv::Point2f arucoPosition ) [private]
```

Oblicza prędkości danej iteracji algorytmu lądowania.

Parametry

img	Przetwarzana ramka obrazu.
arucoPosition	Pozycja znacznika lądowiska na obrazie.

Zwraca

mavsdk::Offboard::VelocityBodyYawspeed

3.1.3.8 landingStep() `bool FlightAutonomy::landingStep ( cv::Mat & img ) [private]`

Wykonuje jeden krok algorytmu lądowania.

Parametry

img	Najnowsza ramka obrazu.

Zwraca

true Poprawnie wykonano krok algorytmu.

false Błąd podczas wykonywania kroku algorytmu.

3.1.3.9 ok() `bool FlightAutonomy::ok ( )`

Sprawdza czy wszystkie elementy działają prawidłowo i czy nie pojawił się warunek wyjścia.

Zwraca

true Wszystkie komponenty działają prawidłowo i nie pojawił się warunek wyjścia.

false Pojawił się błąd działania lub warunek wyjścia.

3.1.3.10 performStep() `int FlightAutonomy::performStep ( cv::Mat & img ) [private]`

Wykonuje pojedyncza iterację dla wybranego algorytmu.

Parametry

img	Najnowsza odebrana ramka obrazu.

Zwraca

int Kod zwrócony przez algorytm.

3.1.3.11 printExitStatus() `void FlightAutonomy::printExitStatus ( )`

Wyświetla status wyjścia według kodu w zmiennej exitCode.

3.1.3.12 readArgs() `bool FlightAutonomy::readArgs ( const int argc, char ** argv )`

Wczytuje przekazane do programu parametry.

Parametry

argc	Liczba parametrów
argv	Wskaźnik na tablicę parametrów

Zwraca

true Poprawnie wczytano parametry.

false Nie można wczytać parametrów.

3.1.3.13 spinOnce() `bool FlightAutonomy::spinOnce ( )`

Wykonuje pojedynczy krok algorytmu. Powinna być wywoływana jednokrotnie w trakcie każdego obrotu pętli głównej programu.

3.1.3.14 stop() `bool FlightAutonomy::stop ( )`

Kończy działanie algorytmu i wyłącza tryb offboard.

Zwraca

true Pomyślnie zakończono działanie.

false Pojawił się błąd podczas próby zakończenia działania.

3.1.4 Dokumentacja atrybutów składowych

3.1.4.1 currAlg

`Algorithms FlightAutonomy::currAlg [private]`

Obecnie wykonywany algorytm.

3.1.4.2 exitCode

`int FlightAutonomy::exitCode [private]`

Kod wyjścia z systemu: 0 - kontynuuj pracę, 1 - wyjście normalne, 2 - niepoprawna wysokość, 3 - nie wykryto znaczników przez zadany czas.

3.1.4.3 flightCtrl

`FlightControl FlightAutonomy::flightCtrl [private]`

Kontrola lotu maszyny.

3.1.4.4 gateArucos

`int FlightAutonomy::gateArucos[4] = {10, 11, 12, 13} [private]`

ID znaczników umieszczonych na bramce zgodnie z ruchem wskazówek zegara.

3.1.4.5 imgRec

`ImageReceiver FlightAutonomy::imgRec [private]`

Odbiornik obrazu z kamery.

3.1.4.6 landingPadID

`int FlightAutonomy::landingPadID = 68 [private]`

ID znacznika Aruco lądowiska.

3.1.4.7 objDetect

`ObjectDetector FlightAutonomy::objDetect [private]`

Wykrywacz obiektów na obrazie z kamery.

3.1.4.8 timeoutCounter

`std::chrono::steady_clock::time_point FlightAutonomy::timeoutCounter [private]`

Odlicza czas do automatycznego wyjścia z programu w przypadku nie wykrywania znaczników.

Dokumentacja dla tej klasy została wygenerowana z plików:

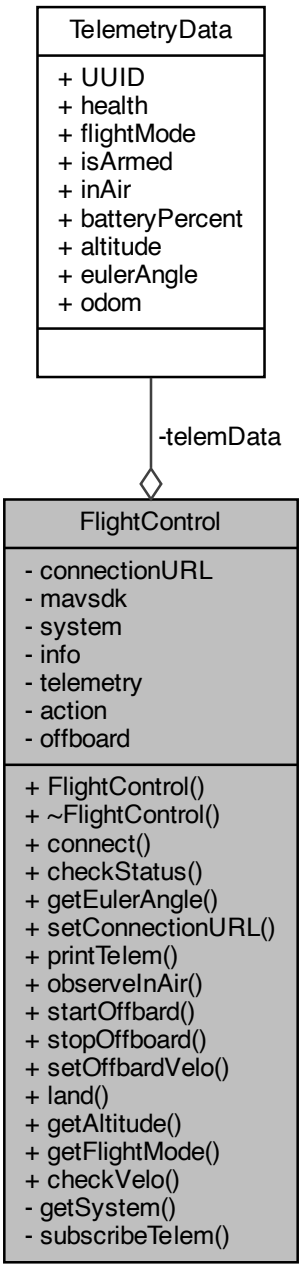
- include/FlightAutonomy/FlightAutonomy.h
- src/FlightAutonomy.cpp

3.2 Dokumentacja klasy FlightControl

Klasa odpowiedzialna za komunikację z autopilotem poprzez protoków MAVLink. Zapewnia odbiór kluczowych danych telemetrycznych i pozwala na sterowanie poprzez komendy offboard.

`#include <FlightControl.h>`

Diagram współpracy dla FlightControl:



Metody publiczne

- FlightControl ()=default

  Tworzy nowy obiekt klasy FlightControl.
- ~FlightControl ()=default

  Niszczy dany obiekt klasy FlightControl.
- bool connect ()

  Inicjalizuje połączenie z autopilotem.
- bool checkStatus ()

  Sprawdza stan maszyny pod kątem gotowości do działania systemu.
- mavsdk::Telemetry::EulerAngle getEulerAngle ()

  Zwraca aktualne kąty pochylenia maszyny.
- void setConnectionURL (const std::string url)

  Ustawia wartość pola connectionURL.
- void printTelem ()

  Wyświetla w konsoli podstawowe dane telemetryczne.
- bool observeInAir ()

  Sprawdzenie czy maszyna jest w trakcie lotu.
- bool startOffbard ()

  Aktywuje kontrolę w trybie offboard i ustawia prędkości ciała na 0.
- bool stopOffboard ()

  Zatrzymuje kontrolę w trybie offboard.
- bool setOffbardVelo (mavsdk::Offboard::VelocityBodyYawspeed veloBodyYawspeed)

  Ustawia prędkości liniowe dla ciała w trybie offboard.
- bool land ()

  Wyzwala tryb lądowania w obecnym punkcie.
- float getAltitude ()

  Zwraca ostatnią wysokość relatywną maszyny nad ziemią.
- mavsdk::Telemetry::FlightMode getFlightMode ()

  Zwraca aktualny tryb lotu.
- bool checkVelo (mavsdk::Offboard::VelocityBodyYawspeed &velocities)

  Sprawdza czy zadane prędkości mieszczą się w dopuszczalnych wartościach i jeżeli nie mieszczą się, wprowadza korektę.

Metody prywatne

- std::shared_ptr< mavsdk::System > getSystem ()

  Zwraca obecny system.
- void subscribeTelem ()

  Subskrybuje wymagane tematy.

Atrybuty prywatne

- std::string connectionURL

  Adres połączenia protokołu MAVLink.
- mavsdk::Mavsdk mavsdk

  Pozwala na zarządzanie połączeniami.
- std::shared_ptr< mavsdk::System > system

  Obiekt reprezentujący system (autopilot)
- std::shared_ptr< mavsdk::Info > info

Dostarcza informacji o systemie.

- std::shared_ptr< mavsdk::Telemetry > telemetry

  Pozwala na odbiór danych telemetrycznych.
- std::shared_ptr< mavsdk::Action > action

  Pozwala na wykonywanie prostych akcji.
- std::shared_ptr< mavsdk::Offboard > offboard

  Pozwala na kontrolę w trybie offboard.
- TelemetryData telemData

  Przechowuje dane telemetryczne odebrane przez MAVLink.

3.2.1 Opis szczegółowy

Klasa odpowiedzialna za komunikację z autopilotem poprzez protoków MAVLink. Zapewnia odbiór kluczowych danych telemetrycznych i pozwala na sterowanie poprzez komendy offboard.

3.2.2 Dokumentacja konstruktora i destruktora

3.2.2.1 FlightControl() `FlightControl::FlightControl ( ) [default]`

Tworzy nowy obiekt klasy FlightControl.

3.2.2.2 ~FlightControl() `FlightControl::~FlightControl ( ) [default]`

Niszczy dany obiekt klasy FlightControl.

3.2.3 Dokumentacja funkcji składowych

3.2.3.1 checkStatus() `bool FlightControl::checkStatus ( )`

Sprawdza stan maszyny pod kątem gotowości do działania systemu.

Zwraca

true Maszyna jest gotowa do wykonywania algorytmu.

false Maszyna nie jest gotowa do wykonywania algorytmu.

3.2.3.2 checkVelo() `bool FlightControl::checkVelo ( mavsdk::Offboard::VelocityBodyYawspeed & velocities )`

Sprawdza czy zadane prędkości mieszczą się w dopuszczalnych wartościach i jeżeli nie mieszczą się, wprowadza korektę.

Parametry

velocities	Prędkości podlegające sprawdzeniu.

Zwraca

true Wartości prędkości mieszczą się w zadanym przedziale.

false Wartości prędkości nie mieszczą się w zadanym przedziale, wprowadzona zastała korekta.

3.2.3.3 connect() `bool FlightControl::connect ( )`

Inicjalizuje połączenie z autopilotem.

Zwraca

true Pomyślnie nawiązano połącznie.

false Wystąpił błąd podczan nawiązywania połączenia.

3.2.3.4 getAltitude() `float FlightControl::getAltitude ( )`

Zwraca ostatnią wysokość relatywną maszyny nad ziemią.

Zwraca

float Wysokość AGL maszyny.

3.2.3.5 getEulerAngle() `mavsdk::Telemetry::EulerAngle FlightControl::getEulerAngle ( )`

Zwraca aktualne kąty pochylenia maszyny.

Zwraca

mavsdk::Telemetry::EulerAngle Kąty Eulera

3.2.3.6 getFlightMode() `mavsdk::Telemetry::FlightMode FlightControl::getFlightMode ( )`

Zwraca aktualny tryb lotu.

Zwraca

Aktualny tryb lotu.

3.2.3.7 getSystem() `std::shared_ptr< mavsdk::System > FlightControl::getSystem ( ) [private]`

Zwraca obecny system.

Zwraca

std::shared_ptr<mavsdk::System> Obecny system

3.2.3.8 land() `bool FlightControl::land ( )`

Wyzwala tryb lądowania w obecnym punkcie.

Zwraca

true Pomyślnie aktywowano tryb lądowanie.

false Wystąpił błąd podczas aktywowania trybu lądowania.

3.2.3.9 observeInAir() `bool FlightControl::observeInAir ( )`

Sprawdzenie czy maszyna jest w trakcie lotu.

Zwraca

true Maszyna jest w trakcie lotu.

false Maszyna nie jest w trakcie lotu.

3.2.3.10 printTelem() `void FlightControl::printTelem ( )`

Wyświetla w konsoli podstawowe dane telemetryczne.

3.2.3.11 setConnectionURL() `void FlightControl::setConnectionURL ( const std::string url )`

Ustawia wartość pola connectionURL.

Parametry

url	Nowy url do ustawienia.

3.2.3.12 setOffbardVelo() `bool FlightControl::setOffbardVelo ( mavsdk::Offboard::VelocityBodyYawspeed veloBodyYawspeed )`

Ustawia prędkości liniowe dla ciała w trybie offboard.

Zwraca

true Pomyślnie ustawiono prędkość.

false Wystąpił błąd podczas ustawiania prędkości.

3.2.3.13 startOffbard() `bool FlightControl::startOffbard ( )`

Aktywuje kontrolę w trybie offboard i ustawia prędkości ciała na 0.

Zwraca

true Pomyślnie aktywowano kontrolę offboard.

false Wystąpił błąd podczas aktywowania kontroli offboard.

3.2.3.14 stopOffboard() `bool FlightControl::stopOffboard ( )`

Zatrzymuje kontrolę w trybie offboard.

Zwraca

true Pomyślnie zatrzymano kontrolę offboard.

false Wystąpił błąd podczas zatrzymywania kontroli offboard.

3.2.3.15 subscribeTelem() `void FlightControl::subscribeTelem ( ) [private]`

Subskrybuje wymagane tematy.

3.2.4 Dokumentacja atrybutów składowych

3.2.4.1 action `std::shared_ptr<mavsdk::Action> FlightControl::action [private]`

Pozwala na wykonywanie prostych akcji.

3.2.4.2 connectionURL `std::string FlightControl::connectionURL [private]`

Adres połączenia protokołu MAVLink.

3.2.4.3 info `std::shared_ptr<mavsdk::Info> FlightControl::info [private]`

Dostarcza informacji o systemie.

3.2.4.4 mavsdk `mavsdk::Mavsdk FlightControl::mavsdk [private]`

Pozwala na zarządzanie połączeniami.

3.2.4.5 offboard `std::shared_ptr<mavsdk::Offboard> FlightControl::offboard [private]`

Pozwala na kontrolę w trybie offboard.

3.2.4.6 system `std::shared_ptr<mavsdk::System> FlightControl::system [private]`

Obiekt reprezentujący system (autopilot)

3.2.4.7 telemData `TelemetryData FlightControl::telemData [private]`

Przechowuje dane telemetryczne odebrane przez MAVLink.

3.2.4.8 telemetry `std::shared_ptr<mavsdk::Telemetry> FlightControl::telemetry [private]`

Pozwala na odbiór danych telemetrycznych.

Dokumentacja dla tej klasy została wygenerowana z plików:

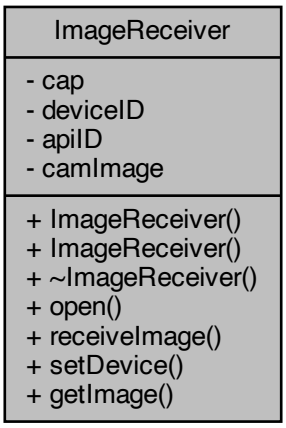
- include/FlightAutonomy/FlightControl.h
- src/FlightControl.cpp

3.3 Dokumentacja klasy ImageReceiver

Klasa odbierająca stream wideo z podanego źródła.

`#include <ImageReceiver.h>`

Diagram współpracy dla ImageReceiver:



Metody publiczne

- ImageReceiver ()=default

  Konstruuje obiekt ImageReceiver.
- ImageReceiver (int deviceID, int apiID=0)

  Konstruuje obiekt ImageReceiver wpisując przekazane wartości do pól tworzonego obiektu.
- ~ImageReceiver ()=default

  Niszczy obiekt ImageReceiver.
- bool open ()

  Otwiera wejście kamery.
- void receiveImage ()

  Pobiera ramkę z wejścia kamery.
- void setDevice (int _deviceID, int _apiID=0)

  Set the Device object.
- cv::Mat getImage ()

  Zwraca najnowszą odebraną ramkę obrazu.

Atrybuty prywatne

- cv::VideoCapture cap

  Obiekt przechwytujący stream video.
- int deviceID = 0

  Kamera 0 - domyślna.
- int apiID = cv::CAP_ANY

  Domyślnie autodetekcja.
- cv::Mat camImage

  Najnowsza ramka odebrana z kamer.

3.3.1 Opis szczegółowy

Klasa odbierająca stream wideo z podanego źródła.

3.3.2 Dokumentacja konstruktora i destruktora

3.3.2.1 ImageReceiver() [1/2] ImageReceiver::ImageReceiver ( ) [default]

Konstruuje obiekt ImageReceiver.

3.3.2.2 ImageReceiver() [2/2] ImageReceiver::ImageReceiver (

```
int deviceID,
int apiID = 0 )
```

Konstruuje obiekt ImageReceiver wpisując przekazane wartości do pól tworzonego obiektu.

3.3.2.3 ∼ImageReceiver() ImageReceiver::∼ImageReceiver ( ) [default]

Niszczy obiekt ImageReceiver.

3.3.3 Dokumentacja funkcji składowych

3.3.3.1 getImage() cv::Mat ImageReceiver::getImage ( )

Zwraca najnowszą odebraną ramkę obrazu.

Zwraca

Ramka obrazu.

3.3.3.2 open() bool ImageReceiver::open ( )

Otwiera wejście kamery.

Zwraca

true Pomyślnie otwarto wejście kamery.

false Wystąpił błąd podczas otwierania wejścia kamery.

3.3.3.3 receiveImage() `void ImageReceiver::receiveImage ( )`

Pobiera ramkę z wejścia kamery.

3.3.3.4 setDevice() `void ImageReceiver::setDevice (`

```
int _deviceID,
int _apiID = 0 )
```

Set the Device object.

Parametry

_deviceID	
_apiID	

3.3.4 Dokumentacja atrybutów składowych

3.3.4.1 apiID `int ImageReceiver::apiID = cv::CAP_ANY [private]`

Domyślnie autodetekcja.

3.3.4.2 camImage `cv::Mat ImageReceiver::camImage [private]`

Najnowsza ramka odebrana z kamer.

3.3.4.3 cap `cv::VideoCapture ImageReceiver::cap [private]`

Obiekt przechwytujący stream video.

3.3.4.4 deviceID `int ImageReceiver::deviceID = 0 [private]`

Kamera 0 - domyślna.

Dokumentacja dla tej klasy została wygenerowana z plików:

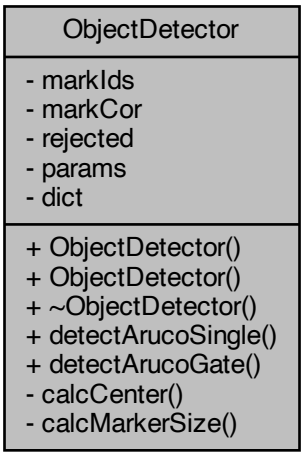
- include/FlightAutonomy/ImageReceiver.h
- src/ImageReceiver.cpp

3.4 Dokumentacja klasy ObjectDetector

Klasa odpowiedzialna za analizę i wykrywanie oraz określanie pozycji obiektów (w tym znaczników Aruco) znajdujących się na przekazanym obrazie.

`#include <ObjectDetector.h>`

Diagram współpracy dla ObjectDetector:



Metody publiczne

- ObjectDetector ()

  Konstruuje obiekt klasy ObjectDetector inicjalizując listę parametrów i słownik wartościami domyślnymi.
- ObjectDetector (cv::Ptr< cv::aruco::DetectorParameters > _params, cv::Ptr< cv::aruco::Dictionary > _dict)

  Konstruuje obiekt klasy ObjectDetector inicjalizując listę parametrów i słownik przekazanymi wartościami.
- ~ObjectDetector ()=default

  Destruktor domyślny.
- cv::Point2f detectArucoSingle (const cv::Mat &img, int arucoID)

  Wykrywa znaczniki Aruco na przekazanej ramce obrazu i oblicza koordynaty środka znacznika. Zwraca (-1, -1) jeżeli nie wykryto znacznika.
- std::tuple< cv::Point2f, float, float > detectArucoGate (const cv::Mat &img, int arucoIDs[4])

  Wykrywa bramkę składającą się z czterech znaczników i oblicza jej środek. Zwraca (-1, -1, -1) jeżeli nie wykryto bramki.

Metody prywatne

- cv::Point2f calcCenter (std::vector< cv::Point2f > corners)

  Oblicza środek znacznika aruco na podstawie przekazanego wektora koordynatów wierzchołków.
- float calcMarkerSize (std::vector< cv::Point2f > corners, int imgHeight)

  Oblicza względny rozmiar znacznika na ramce obrazu.

Atrybuty prywatne

- std::vector< int > markIds

  Wektor wszystkich wykrytych na obrazie znaczników.
- std::vector< std::vector< cv::Point2f > > markCor

  Wektor wektorów koordynatów każdego wykrytecho znacznika.
- std::vector< std::vector< cv::Point2f > > rejected

  Wektor odrzuconych potencjalnych obiektów będących znacznikami.
- cv::Ptr< cv::aruco::DetectorParameters > params

  Wskaźnik na parametry algorytmu wykrywającego znaczniki.
- cv::Ptr< cv::aruco::Dictionary > dict

  Wskaźnik na słownik znaczników.

3.4.1 Opis szczegółowy

Klasa odpowiedzialna za analizę i wykrywanie oraz określanie pozycji obiektów (w tym znaczników Aruco) znajdujących się na przekazanym obrazie.

3.4.2 Dokumentacja konstruktora i destruktora

3.4.2.1 ObjectDetector() [1/2] ObjectDetector::ObjectDetector ( )

Konstruuje obiekt klasy ObjectDetector inicjalizując listę parametrów i słownik wartościami domyślnymi.

3.4.2.2 ObjectDetector() [2/2] ObjectDetector::ObjectDetector (cv::Ptr< cv::aruco::DetectorParameters > *_params*, cv::Ptr< cv::aruco::Dictionary > *_dict*)

Konstruuje obiekt klasy ObjectDetector inicjalizując listę parametrów i słownik przekazanymi wartościami.

Parametry

_params	Parametry algorytmu wykrywania znaczników.
_dict	Słownik którym ma się posługiwać algorytm wykrywania znaczników.

3.4.2.3 ~ObjectDetector() ObjectDetector::~ObjectDetector ( ) [default]

Destruktor domyślny.

3.4.3 Dokumentacja funkcji składowych

3.4.3.1 calcCenter() `cv::Point2f ObjectDetector::calcCenter ( std::vector< cv::Point2f > corners ) [private]`

Oblicza środek znacznika aruco na podstawie przekazanego wektora koordynatów wierzchołków.

Parametry

corners	Wektor zawierający koordynaty kolejnych wierzchołków znacznika.

Zwraca

cv::Point2f Obliczony środek znacznika.

3.4.3.2 calcMarkerSize() `float ObjectDetector::calcMarkerSize ( std::vector< cv::Point2f > corners, int imgHeight ) [private]`

Oblicza względny rozmiar znacznika na ramce obrazu.

Parametry

corners	Wektor zawierający koordynaty kolejnych wierzchołków znacznika.
imgHeight	Wysokość analizowanego obrazu.

Zwraca

float Względny rozmiar znacznika dla osi X i Y.

3.4.3.3 detectArucoGate() `std::tuple< cv::Point2f, float, float > ObjectDetector::detectArucoGate ( const cv::Mat & img, int arucoIDs[4] )`

Wykrywa bramkę składającą się z czterech znaczników i oblicza jej środek. Zwraca (-1, -1, -1) jeżeli nie wykryto bramki.

Parametry

img	Ramka obrazu na której mają zostać wykryte znaczniki.
arucoIDs	Tablica identyfikatorów kolejnych znaczników tworzących bramkę.

Zwraca

std::tuple<cv::Point2f, float, float> Koordynaty środka bramki, jej rozmiar, i pochylenie.

3.4.3.4 detectArucoSingle()

```
cv::Point2f ObjectDetector::detectArucoSingle (
        const cv::Mat & img,
        int arucoID )
```

Wykrywa znaczniki Aruco na przekazanej ramce obrazu i oblicza koordynaty środka znacznika. Zwraca (-1, -1) jeżeli nie wykryto znacznika.

Parametry

img	Ramka obrazu na której mają zostać wykryte znaczniki.
arucoID	ID znacznika Aruco którego koordynaty mają zostać określone.

Zwraca

cv::Point2f Koordynaty środka znacznika Aruco.

3.4.4 Dokumentacja atrybutów składowych

3.4.4.1 dict

```
cv::Ptr<cv::aruco::Dictionary> ObjectDetector::dict  [private]
```

Wskaźnik na słownik znaczników.

3.4.4.2 markCor

```
std::vector<std::vector<cv::Point2f> > ObjectDetector::markCor  [private]
```

Wektor wektorów koordynatów każdego wykrytecho znacznika.

3.4.4.3 markIds

```
std::vector<int> ObjectDetector::markIds  [private]
```

Wektor wszystkich wykrytych na obrazie znaczników.

3.4.4.4 params

```
cv::Ptr<cv::aruco::DetectorParameters> ObjectDetector::params  [private]
```

Wskaźnik na parametry algorytmu wykrywającego znaczniki.

3.4.4.5 rejected

```
std::vector<std::vector<cv::Point2f> > ObjectDetector::rejected  [private]
```

Wektor odrzuconych potencjalnych obiektów będących znacznikami.

Dokumentacja dla tej klasy została wygenerowana z plików:

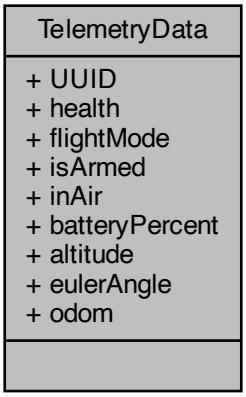
- include/FlightAutonomy/ObjectDetector.h
- src/ObjectDetector.cpp

3.5 Dokumentacja struktury TelemetryData

Struktura przechowująca dane telemetryczne odebrane z pojazdu przez MavLink.

`#include <TelemetryData.h>`

Diagram współpracy dla TelemetryData:



Atrybuty publiczne

- uint64_t UUID = 0

  Identyfikator systemu.
- bool health = false

  Stan pojazdu. Domyślnie niegotowy do lotu.
- mavsdk::Telemetry::FlightMode flightMode

  Tryb lotu.
- bool isArmed = false

  Stan uzbrojenia. Domyślnie rozbrojony.
- bool inAir = false

  Czy pojazd jest w powietrzu.
- float batteryPercent = -1

  Stan naładowania akumulatorów.
- float altitude = 0

  Wysokość relatywna względem miejsca startu.
- mavsdk::Telemetry::EulerAngle eulerAngle

  Aktualne kąty pochylenia autopilota.
- mavsdk::Telemetry::Odometry odom

  Aktualne dane otometryczne.

3.5.1 Opis szczegółowy

Struktura przechowująca dane telemetryczne odebrane z pojazdu przez MavLink.

3.5.2 Dokumentacja atrybutów składowych

3.5.2.1 altitude `float TelemetryData::altitude = 0`

Wysokość relatywna względem miejsca startu.

3.5.2.2 batteryPercent `float TelemetryData::batteryPercent = -1`

Stan naładowania akumulatorów.

3.5.2.3 eulerAngle `mavsdk::Telemetry::EulerAngle TelemetryData::eulerAngle`

Aktualne kąty pochylenia autopilota.

3.5.2.4 flightMode `mavsdk::Telemetry::FlightMode TelemetryData::flightMode`

Tryb lotu.

3.5.2.5 health `bool TelemetryData::health = false`

Stan pojazdu. Domyślnie niegotowy do lotu.

3.5.2.6 inAir `bool TelemetryData::inAir = false`

Czy pojazd jest w powietrzu.

3.5.2.7 isArmed `bool TelemetryData::isArmed = false`

Stan uzbrojenia. Domyślnie rozbrojony.

3.5.2.8 odom `mavsdk::Telemetry::Odometry TelemetryData::odom`

Aktualne dane otometryczne.

3.5.2.9 UUID `uint64_t TelemetryData::UUID = 0`

Identyfikator systemu.

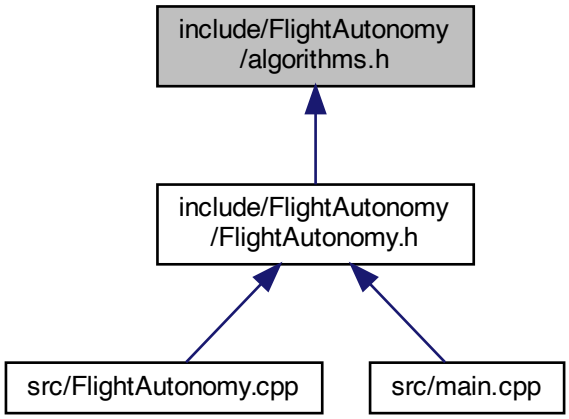
Dokumentacja dla tej struktury została wygenerowana z pliku:

- include/FlightAutonomy/TelemetryData.h

4 Dokumentacja plików

4.1 Dokumentacja pliku include/FlightAutonomy/algorithms.h

Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Wyliczenia

- enum Algorithms { arucoLanding = 0 , gateRacing , normalLanding , forwardFlight }

  Zawiera dostępne algorytmy i tryby lotu.

4.1.1 Dokumentacja typów wyliczanych

4.1.1.1 Algorithms `enum Algorithms`

Zawiera dostępne algorytmy i tryby lotu.

Wartości wyliczeń

arucoLanding	Algorytm precyzyjnego lądowania.
gateRacing	Algorytm przelotu przez bramki.
normalLanding	Automatyczne lądowanie w obecnej lokalizacji.
forwardFlight	Tryb lotu do przodu przez określony czas.

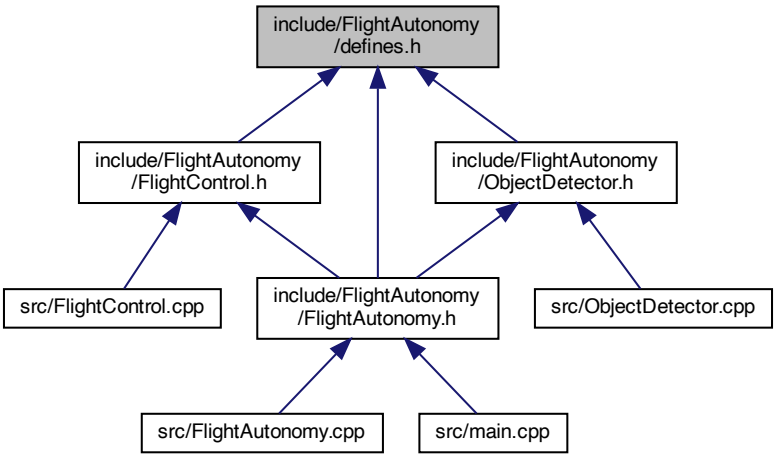
4.2 algorithms.h

Idź do dokumentacji tego pliku.

```
1 #pragma once
2
6 enum Algorithms
7 {
8     arucoLanding = 0,
9     gateRacing,
10     normalLanding,
11     forwardFlight
12 };
```

4.3 Dokumentacja pliku include/FlightAutonomy/defines.h

Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Definicje

- #define FA_DEBUG

  Tryb debugowania wyświetlający oknie z obrazem odbieranym z kamery.

Zmienne

- const float MAX_VELO_HORI_MS = 1

  Maksymalna wartość prędkości w poziomie (osie X i Y) w m/.
- const float MAX_VELO_VERT_MS = 0.75

  Maksymalna wartość prędkości w pionie (oś Z) w m/s.
- const float MAX_YAWSPEED = 30

  Maksymalna wartość prędkości kątowej wokół osi Z (yaw) w stopniach na sekundę
- const float MAX_FLIGHT_ALT = 10

  Maksymalna wysokość do której system może zostać uruchomiony.
- const float MID_THRESHOLD = 0.4

  Próg do którego przyjmowana jest odległość jako bliska środka obrazu.
- const float LAND_ALT = 1

  Wysokość AGL w metrach poniżej której następuje przyziemienie w danym miejscu.
- const float RACING_ALT = 15

  Wysokość AGL w metrach na której musi się znajdować maszyna podczas pokonywania bramki.
- const float IS_STRAIGHT_ANGLE = 1.25

  Kąt poniżej którego przyjmuje się że maszyna jest na wprost bramki.
- const float CAM_HFOV = 60

  Horyzontalny kąt widzenia kamery w stopniach.
- const float CAM_VFOV = 45

  Wertykalny kąt widzenia kamery w stopniach.
- const int NO_DETECT_TIMEOUT = 10

  Czas po którym nastąpi przerwanie działania w przypadku nie wykrywania znaczników.
- const int FORWARD_FLIGHT_TIMEOUT = 7

  Czas przez który maszyna leci do przodu aby pokonać przeszkodę

4.3.1 Dokumentacja definicji

4.3.1.1 FA_DEBUG `#define FA_DEBUG`

Tryb debugowania wyświetlający oknie z obrazem odbieranym z kamery.

4.3.2 Dokumentacja zmiennych

4.3.2.1 CAM_HFOV `const float CAM_HFOV = 60`

Horyzontalny kąt widzenia kamery w stopniach.

4.3.2.2 CAM_VFOV `const float CAM_VFOV = 45`

Wertykalny kąt widzenia kamery w stopniach.

4.3.2.3 FORWARD_FLIGHT_TIMEOUT `const int FORWARD_FLIGHT_TIMEOUT = 7`

Czas przez który maszyna leci do przodu aby pokonać przeszkodę

4.3.2.4 IS_STRAIGHT_ANGLE `const float IS_STRAIGHT_ANGLE = 1.25`

Kąt poniżej którego przyjmuje się że maszyna jest na wprost bramki.

4.3.2.5 LAND_ALT `const float LAND_ALT = 1`

Wysokość AGL w metrach poniżej której następuje przyziemienie w danym miejscu.

4.3.2.6 MAX_FLIGHT_ALT `const float MAX_FLIGHT_ALT = 10`

Maksymalna wysokość do której system może zostać uruchomiony.

4.3.2.7 MAX_VELO_HORI_MS `const float MAX_VELO_HORI_MS = 1`

Maksymalna wartość prędkości w poziomie (osie X i Y) w m/.

4.3.2.8 MAX_VELO_VERT_MS `const float MAX_VELO_VERT_MS = 0.75`

Maksymalna wartość prędkości w pionie (oś Z) w m/s.

4.3.2.9 MAX_YAWSPEED `const float MAX_YAWSPEED = 30`

Maksymalna wartość prędkości kątowej wokół osi Z (yaw) w stopniach na sekundę

4.3.2.10 MID_THRESHOLD `const float MID_THRESHOLD = 0.4`

Próg do którego przyjmowana jest odległość jako bliska środka obrazu.

4.3.2.11 NO_DETECT_TIMEOUT `const int NO_DETECT_TIMEOUT = 10`

Czas po którym nastąpi przerwanie działania w przypadku nie wykrywania znaczników.

4.3.2.12 RACING_ALT `const float RACING_ALT = 15`

Wysokość AGL w metrach na której musi się znajdować maszyna podczas pokonywania bramki.

4.4 defines.h

Idź do dokumentacji tego pliku.

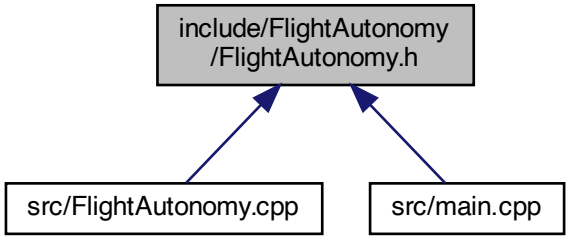
```
1 #pragma once
2
3 #define FA_DEBUG
4
5 const float MAX_VELO_HORI_MS = 1;
6 const float MAX_VELO_VERT_MS = 0.75;
7 const float MAX_YAWSPEED = 30;
8 const float MAX_FLIGHT_ALT = 10;
9 const float MID_THRESHOLD = 0.4;
10 const float LAND_ALT = 1;
11 const float RACING_ALT = 15;
12 const float IS_STRAIGHT_ANGLE = 1.25;
13
14 const float CAM_HFOV = 60;
15 const float CAM_VFOV = 45;
16
17 const int NO_DETECT_TIMEOUT = 10;
18 const int FORWARD_FLIGHT_TIMEOUT = 7;
```

4.5 Dokumentacja pliku include/FlightAutonomy/FlightAutonomy.h

```
#include "FlightAutonomy/defines.h"
#include <opencv2/opencv.hpp>
#include <chrono>
#include "FlightAutonomy/ImageReceiver.h"
#include "FlightAutonomy/FlightControl.h"
#include "FlightAutonomy/ObjectDetector.h"
#include "FlightAutonomy/algorithms.h"
```

Wykres zależności załączania dla FlightAutonomy.h:

Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Komponenty

- class FlightAutonomy

  Klasa odpowiedzialna za kompleksową obsługę autonomii lotu bazującej na analizie wizyjnej. W klasie realizowany jest algorytm analizujący obraz odbierany z kamery i w przypadku wykrycia przeszkody wykonanie odpowiedniej reakcji w postaci zmiany trajektorii lotu maszyny. Wykorzystywany jest protokół MavLink do dwukierunkowej komunikacji z autopilotem i przesyłania komend sterujących.

4.6 FlightAutonomy.h

Idź do dokumentacji tego pliku.

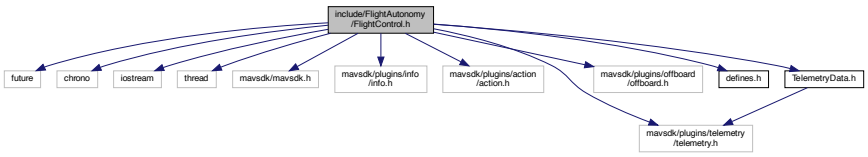
```
1 #pragma once
2
3 #include "FlightAutonomy/defines.h"
4
5 #include <opencv2/opencv.hpp>
6 #ifdef FA_DEBUG
7 #include <opencv2/highgui/highgui.hpp>
8 #endif
9 #include <chrono>
10
11 #include "FlightAutonomy/ImageReceiver.h"
12 #include "FlightAutonomy/FlightControl.h"
13 #include "FlightAutonomy/ObjectDetector.h"
14 #include "FlightAutonomy/algorithms.h"
15
21 class FlightAutonomy
22 {
23 #ifdef FA_DEBUG
24     const std::string OPENCV_WINDOW = "Cam View";
25 #endif
26
27     ImageReceiver imgRec;
28     FlightControl flightCtrl;
29     ObjectDetector objDetect;
30     Algorithms currAlg;
31     int landingPadID = 68;
32     int gateArucos[4] = {10, 11, 12, 13};
33     int exitCode;
34     std::chrono::steady_clock::time_point timeoutCounter;
35
36 public:
40     FlightAutonomy();
41
45     ~FlightAutonomy();
46
53     bool connect();
54
61     bool isReady();
62
72     bool readArgs(const int argc, char **argv);
```

```
73
78     bool spinOnce();
79
86     bool ok();
87
94     bool stop();
95
101     int getExitCode();
102
106     void printExitStatus();
107
108 private:
116     int performStep(cv::Mat &img);
117
126     bool landingStep(cv::Mat &img);
127
135     mavsdk::Offboard::VelocityBodyYawspeed landingCalcVelo(cv::Mat &img, cv::Point2f arucoPosition);
136
145     bool avoidingStep(cv::Mat &img);
146
155     mavsdk::Offboard::VelocityBodyYawspeed avoidingCalcVelo(cv::Mat &img, cv::Point2f gatePosition,
      float angle);
156
160     void checkTimeouts();
161 };
```

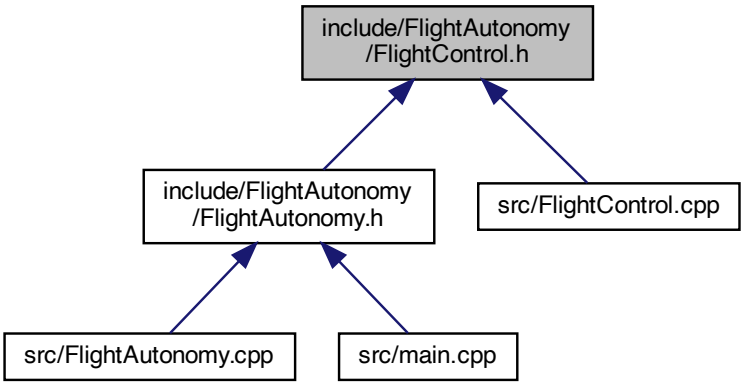
4.7 Dokumentacja pliku include/FlightAutonomy/FlightControl.h

```
#include <future>
#include <chrono>
#include <iostream>
#include <thread>
#include <mavsdk/mavsdk.h>
#include <mavsdk/plugins/info/info.h>
#include <mavsdk/plugins/action/action.h>
#include <mavsdk/plugins/telemetry/telemetry.h>
#include <mavsdk/plugins/offboard/offboard.h>
#include "defines.h"
#include "TelemetryData.h"
```

Wykres zależności załączania dla FlightControl.h:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Komponenty

- class FlightControl

  Klasa odpowiedzialna za komunikację z autopilotem poprzez protoków MAVLink. Zapewnia odbiór kluczowych danych telemetrycznych i pozwala na sterowanie poprzez komendy offboard.

4.8 FlightControl.h

Idź do dokumentacji tego pliku.

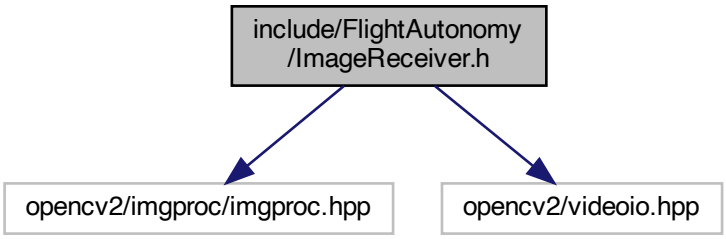
```
1 #pragma once
2
3 #include <future>
4 #include <chrono>
5 #include <iostream>
6 #include <thread>
7 #include <mavsdk/mavsdk.h>
8 #include <mavsdk/plugins/info/info.h>
9 #include <mavsdk/plugins/action/action.h>
10 #include <mavsdk/plugins/telemetry/telemetry.h>
11 #include <mavsdk/plugins/offboard/offboard.h>
12
13 #include "defines.h"
14 #include "TelemetryData.h"
15
20 class FlightControl
21 {
22     std::string connectionURL;
23
24     mavsdk::Mavsdk mavsdk;
25     std::shared_ptr<mavsdk::System> system;
26     std::shared_ptr<mavsdk::Info> info;
27     std::shared_ptr<mavsdk::Telemetry> telemetry;
28     std::shared_ptr<mavsdk::Action> action;
29     std::shared_ptr<mavsdk::Offboard> offboard;
30
31     TelemetryData telemData;
32
38     std::shared_ptr<mavsdk::System> getSystem();
39
44     void subscribeTelem();
45
46 public:
50     FlightControl() = default;
51
55     ~FlightControl() = default;
```

```
56
63      bool connect();
64
71      bool checkStatus();
72
78      mavsdk::Telemetry::EulerAngle getEulerAngle();
79
85      void setConnectionURL(const std::string url);
86
90      void printTelem();
91
98      bool observeInAir();
99
106      bool startOffbard();
107
114      bool stopOffboard();
115
122      bool setOffbardVelo(mavsdk::Offboard::VelocityBodyYawspeed veloBodyYawspeed);
123
130      bool land();
131
137      float getAltitude();
138
144      mavsdk::Telemetry::FlightMode getFlightMode();
145
153      bool checkVelo(mavsdk::Offboard::VelocityBodyYawspeed &velocities);
154 };
```

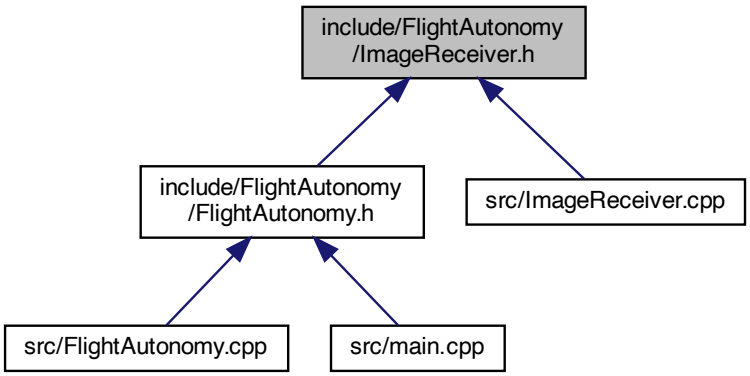
4.9 Dokumentacja pliku include/FlightAutonomy/ImageReceiver.h

```
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/videoio.hpp>
```

Wykres zależności załączania dla ImageReceiver.h:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Komponenty

- class ImageReceiver

  Klasa odbierająca stream wideo z podanego źródła.

4.10 ImageReceiver.h

Idź do dokumentacji tego pliku.

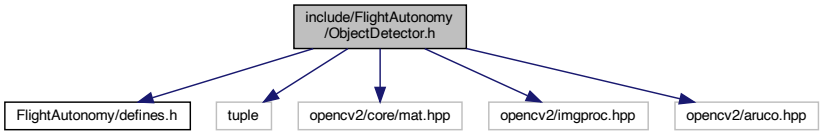
```
1 #pragma once
2
3 #include <opencv2/imgproc/imgproc.hpp>
4 #include <opencv2/videoio.hpp>
5
9 class ImageReceiver
10 {
11     cv::VideoCapture cap;
12     int deviceID = 0;
13     int apiID = cv::CAP_ANY;
14     cv::Mat camImage;
15
16 public:
20     ImageReceiver() = default;
21
25     ImageReceiver(int deviceID, int apiID = 0);
26
30     ~ImageReceiver() = default;
31
38     bool open();
39
43     void receiveImage();
44
51     void setDevice(int _deviceID, int _apiID = 0);
52
58     cv::Mat getImage();
59 };
```

4.11 Dokumentacja pliku include/FlightAutonomy/ObjectDetector.h

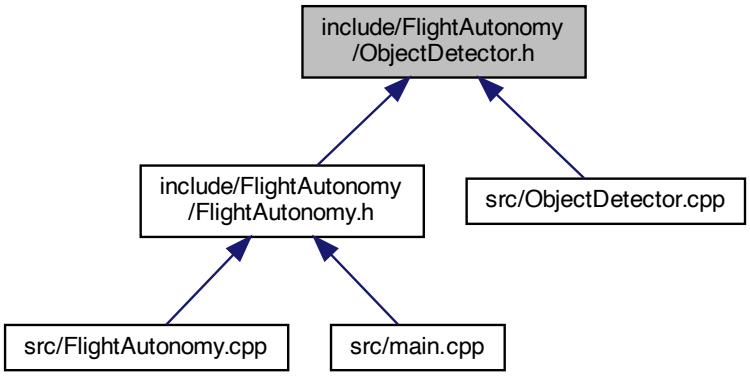
```
#include "FlightAutonomy/defines.h"
#include <tuple>
```

```
#include <opencv2/core/mat.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/aruco.hpp>
```

Wykres zależności załączania dla ObjectDetector.h:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Komponenty

- class ObjectDetector

  Klasa odpowiedzialna za analizę i wykrywanie oraz określanie pozycji obiektów (w tym znaczników Aruco) znajdujących się na przekazanym obrazie.

4.12 ObjectDetector.h

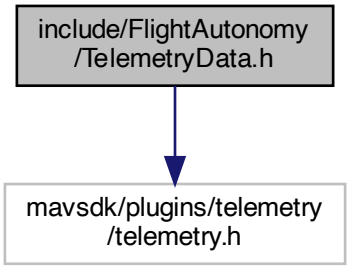
Idź do dokumentacji tego pliku.

```
1 #pragma once
2 #include "FlightAutonomy/defines.h"
3
4 #include <tuple>
5 #include <opencv2/core/mat.hpp>
6 #include <opencv2/imgproc.hpp>
7 #include <opencv2/aruco.hpp>
8
13 class ObjectDetector
14 {
15     std::vector<int> markIds;
16     std::vector<std::vector<cv::Point2f» markCor;
17     std::vector<std::vector<cv::Point2f» rejected;
18
```

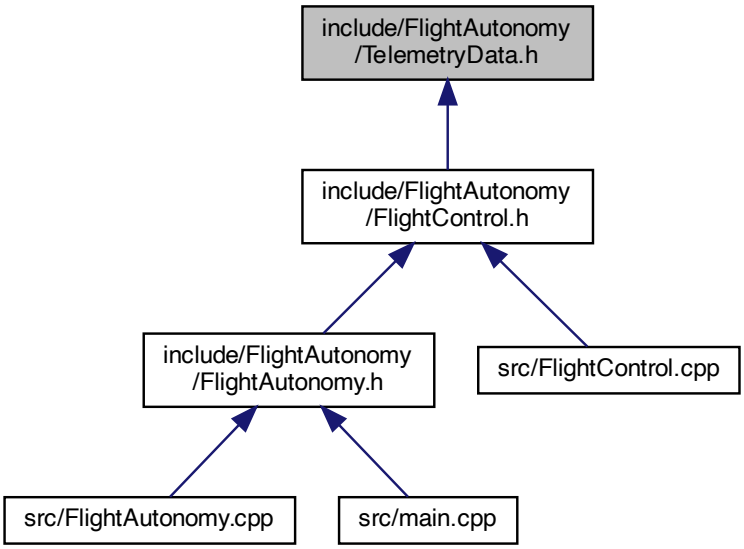
```
19      cv::Ptr<cv::aruco::DetectorParameters> params;
20      cv::Ptr<cv::aruco::Dictionary> dict;
21
22  public:
26      ObjectDetector();
27
34      ObjectDetector(cv::Ptr<cv::aruco::DetectorParameters> _params, cv::Ptr<cv::aruco::Dictionary> _dict);
35
39      ~ObjectDetector() = default;
40
48      cv::Point2f detectArucoSingle(const cv::Mat &img, int arucoID);
49
57      std::tuple<cv::Point2f, float, float> detectArucoGate(const cv::Mat &img, int arucoIDs[4]);
58
59  private:
66      cv::Point2f calcCenter(std::vector<cv::Point2f> corners);
67
75      float calcMarkerSize(std::vector<cv::Point2f> corners, int imgHeight);
76  };
```

4.13 Dokumentacja pliku include/FlightAutonomy/TelemetryData.h

`#include <mavsdk/plugins/telemetry/telemetry.h>`
Wykres zależności załączania dla TelemetryData.h:



Ten wykres pokazuje, które pliki bezpośrednio lub pośrednio załączają ten plik:



Komponenty

- struct TelemetryData

  Struktura przechowująca dane telemetryczne odebrane z pojazdu przez MavLink.

4.14 TelemetryData.h

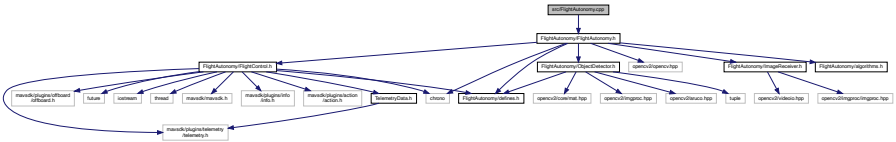
Idź do dokumentacji tego pliku.

```
1 #pragma once
2
3 #include <mavsdk/plugins/telemetry/telemetry.h>
4
8 struct TelemetryData
9 {
10     uint64_t UUID = 0;
11     bool health = false;
12     mavsdk::Telemetry::FlightMode flightMode;
13
14     bool isArmed = false;
15     bool inAir = false;
16
17     float batteryPercent = -1;
18
19     float altitude = 0;
20
21     mavsdk::Telemetry::EulerAngle eulerAngle;
22     mavsdk::Telemetry::Odometry odom;
23 };
```

4.15 Dokumentacja pliku src/FlightAutonomy.cpp

```
#include "FlightAutonomy/FlightAutonomy.h"
```

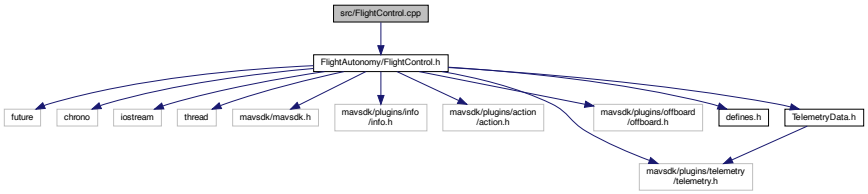
Wykres zależności załączania dla FlightAutonomy.cpp:



4.16 Dokumentacja pliku src/FlightControl.cpp

```
#include "FlightAutonomy/FlightControl.h"
```

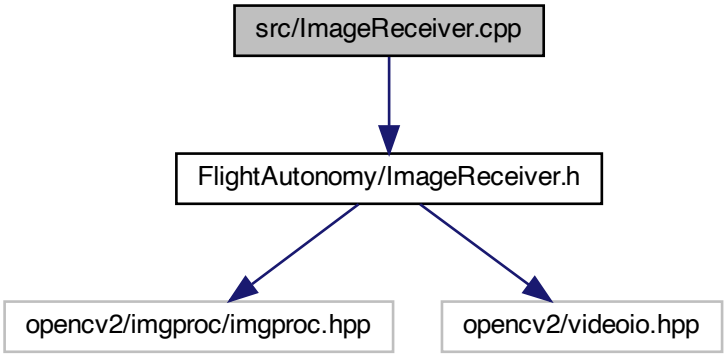
Wykres zależności załączania dla FlightControl.cpp:



4.17 Dokumentacja pliku src/ImageReceiver.cpp

```
#include "FlightAutonomy/ImageReceiver.h"
```

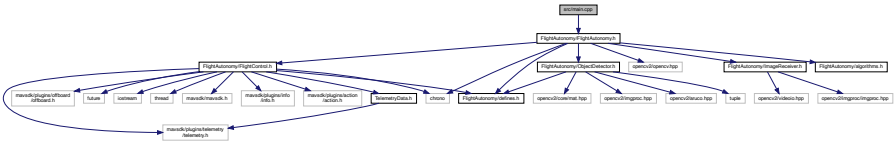
Wykres zależności załączania dla ImageReceiver.cpp:



4.18 Dokumentacja pliku src/main.cpp

```
#include "FlightAutonomy/FlightAutonomy.h"
```

Wykres zależności załączania dla main.cpp:



Funkcje

- void printHelp (std::string progName)
- int main (int argc, char ∗∗argv)

4.18.1 Dokumentacja funkcji

4.18.1.1 main()

```
int main (
        int argc,
        char ** argv )
```

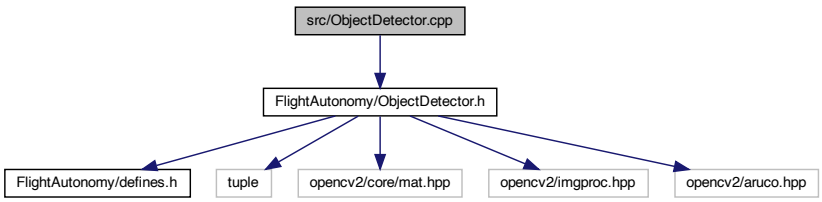
4.18.1.2 printHelp()

```
void printHelp (
        std::string progName )
```

4.19 Dokumentacja pliku src/ObjectDetector.cpp

```
#include "FlightAutonomy/ObjectDetector.h"
```

Wykres zależności załączania dla ObjectDetector.cpp:



Indeks