

## Flight Autonomy

Wygenerowano przez Doxygen 1.9.3

<b>1 Indeks klas</b>	<b>1</b>
1.1 Lista klas	1
<b>2 Indeks plików</b>	<b>1</b>
2.1 Lista plików	1
<b>3 Dokumentacja klas</b>	<b>2</b>
3.1 Dokumentacja klasy FlightAutonomy	2
3.1.1 Opis szczegółowy	5
3.1.2 Dokumentacja konstruktora i destruktora	5
3.1.3 Dokumentacja funkcji składowych	5
3.1.4 Dokumentacja atrybutów składowych	9
3.2 Dokumentacja klasy FlightControl	11
3.2.1 Dokumentacja konstruktora i destruktora	13
3.2.2 Dokumentacja funkcji składowych	13
3.2.3 Dokumentacja atrybutów składowych	16
3.3 Dokumentacja klasy ImageReceiver	18
3.3.1 Opis szczegółowy	19
3.3.2 Dokumentacja konstruktora i destruktora	19
3.3.3 Dokumentacja funkcji składowych	19
3.3.4 Dokumentacja atrybutów składowych	21
3.4 Dokumentacja klasy ObjectDetector	22
3.4.1 Opis szczegółowy	23
3.4.2 Dokumentacja konstruktora i destruktora	23
3.4.3 Dokumentacja funkcji składowych	23
3.4.4 Dokumentacja atrybutów składowych	25
3.5 Dokumentacja struktury TelemetryData	26
3.5.1 Opis szczegółowy	26
3.5.2 Dokumentacja atrybutów składowych	27
<b>4 Dokumentacja plików</b>	<b>28</b>
4.1 Dokumentacja pliku include/FlightAutonomy/algorithms.h	28
4.1.1 Dokumentacja typów wyliczanych	28
4.2 algorithms.h	29
4.3 Dokumentacja pliku include/FlightAutonomy/defines.h	29
4.3.1 Dokumentacja definicji	30
4.3.2 Dokumentacja zmiennych	30
4.4 defines.h	32
4.5 Dokumentacja pliku include/FlightAutonomy/FlightAutonomy.h	32
4.6 FlightAutonomy.h	33
4.7 Dokumentacja pliku include/FlightAutonomy/FlightControl.h	34
4.8 FlightControl.h	35
4.9 Dokumentacja pliku include/FlightAutonomy/ImageReceiver.h	36

4.10 ImageReceiver.h . . . . .	37
4.11 Dokumentacja pliku include/FlightAutonomy/ObjectDetector.h . . . . .	37
4.12 ObjectDetector.h . . . . .	38
4.13 Dokumentacja pliku include/FlightAutonomy/TelemetryData.h . . . . .	39
4.14 TelemetryData.h . . . . .	40
4.15 Dokumentacja pliku src/FlightAutonomy.cpp . . . . .	41
4.16 Dokumentacja pliku src/FlightControl.cpp . . . . .	41
4.17 Dokumentacja pliku src/ImageReceiver.cpp . . . . .	41
4.18 Dokumentacja pliku src/main.cpp . . . . .	42
4.18.1 Dokumentacja funkcji . . . . .	42
4.19 Dokumentacja pliku src/ObjectDetector.cpp . . . . .	42
<b>Indeks</b>	<b>43</b>

## 1 Indeks klas

### 1.1 Lista klas

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

#### 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

2

#### FlightControl

11

#### ImageReceiver

Klasa odbierająca stream wideo z podanego źródła

18

#### ObjectDetector

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

22

#### TelemetryData

Struktura przechowująca dane telemetryczne odebrane z pojazdu przez MavLink

26

## 2 Indeks plików

### 2.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

include/FlightAutonomy/algorithms.h

28

include/FlightAutonomy/defines.h

29

<code>include/FlightAutonomy/FlightAutonomy.h</code>	32
<code>include/FlightAutonomy/FlightControl.h</code>	34
<code>include/FlightAutonomy/ImageReceiver.h</code>	36
<code>include/FlightAutonomy/ObjectDetector.h</code>	37
<code>include/FlightAutonomy/TelemetryData.h</code>	39
<code>src/FlightAutonomy.cpp</code>	41
<code>src/FlightControl.cpp</code>	41
<code>src/ImageReceiver.cpp</code>	41
<code>src/main.cpp</code>	42
<code>src/ObjectDetector.cpp</code>	42

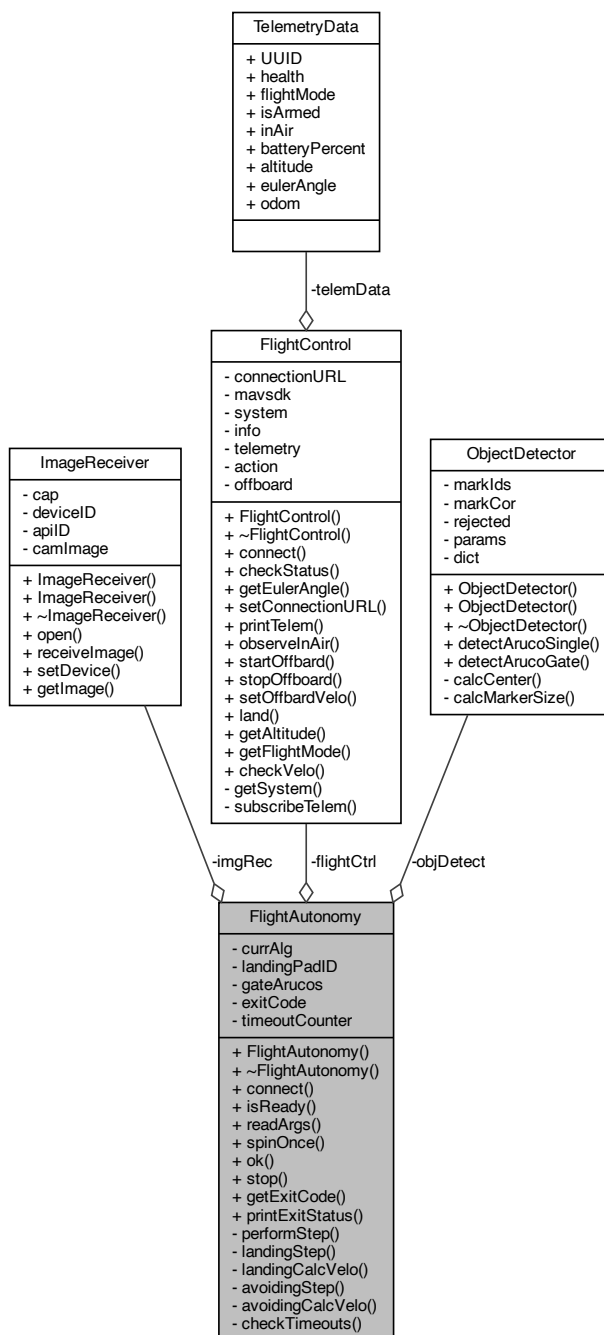
## 3 Dokumentacja klas

### 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 ()`  
Destruktor. 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 pojedynczą 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 niszczy okno OpenCV.

### 3.1.3 Dokumentacja funkcji składowych

#### 3.1.3.1 avoidingCalcVelo() `mavsdk::Offboard::VelocityBodyYawspeed FlightAutonomy::avoiding↔`

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

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

#### Parametry

<i>img</i>	Przetwarzana ramka obrazu.
<i>gatePosition</i>	Pozycja bramki na obrazie.
<i>angle</i>	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

<i>img</i>	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::landingCalc↔  
Velo (`

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

Oblicza prędkości danej iteracji algorytmu lądowania.

**Parametry**

<i>img</i>	Przetwarzana ramka obrazu.
<i>arucoPosition</i>	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**

<i>img</i>	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 pojedynczą iterację dla wybranego algorytmu.

**Parametry**

<i>img</i>	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**

<i>argc</i>	Liczba parametrów
<i>argv</i>	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.

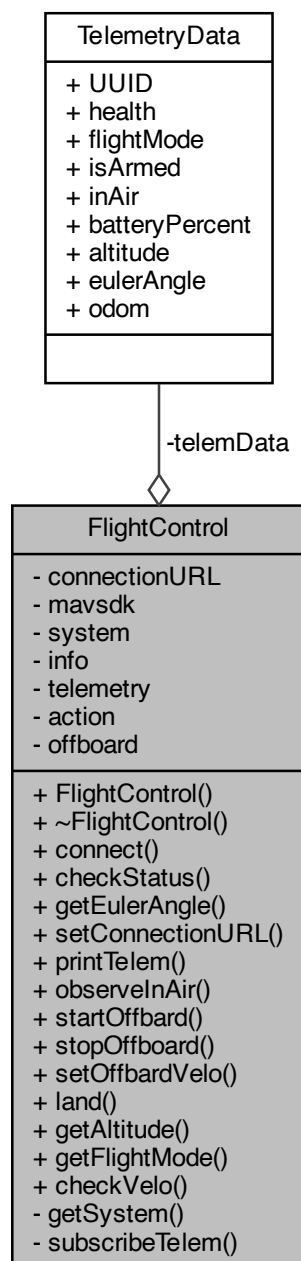
Dokumentacja dla tej klasy została wygenerowana z plików:

- include/FlightAutonomy/[FlightAutonomy.h](#)
- src/[FlightAutonomy.cpp](#)

## 3.2 Dokumentacja klasy FlightControl

```
#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 startOffboard ()`  
Aktywuje kontrolę w trybie offboard i ustawia prędkości ciała na 0.
- `bool stopOffboard ()`  
Zatrzymuje kontrolę w trybie offboard.
- `bool setOffboardVelo (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 Dokumentacja konstruktora i destruktoru

#### 3.2.1.1 FlightControl() `FlightControl::FlightControl ( ) [default]`

Tworzy nowy obiekt klasy [FlightControl](#).

#### 3.2.1.2 ~FlightControl() `FlightControl::~~FlightControl ( ) [default]`

Niszczy dany obiekt klasy [FlightControl](#).

### 3.2.2 Dokumentacja funkcji składowych

#### 3.2.2.1 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.

#### 3.2.2.2 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

<code>velocities</code>	Prędkości podlegające sprawdzeniu.
-------------------------	------------------------------------

**Zwraca**

true Wartości prędkości mieszczą się w zadanym przedziale.

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

**3.2.2.3 connect()** `bool FlightControl::connect ( )`

Inicjalizuje połączenie z autopilotem.

**Zwraca**

true Pomyślnie nawiązano połączenie.

false Wystąpił błąd podczas nawiązywania połączenia.

**3.2.2.4 getAltitude()** `float FlightControl::getAltitude ( )`

Zwraca ostatnią wysokość relatywną maszyny nad ziemią.

**Zwraca**

float Wysokość AGL maszyny.

**3.2.2.5 getEulerAngle()** `mavsdk::Telemetry::EulerAngle FlightControl::getEulerAngle ( )`

Zwraca aktualne kąty pochylenia maszyny.

**Zwraca**

mavsdk::Telemetry::EulerAngle Kąty Eulera

**3.2.2.6 getFlightMode()** `mavsdk::Telemetry::FlightMode FlightControl::getFlightMode ( )`

Zwraca aktualny tryb lotu.

**Zwraca**

Aktualny tryb lotu.



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

Zwraca obecny system.

Zwraca

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

**3.2.2.8 land()** `bool FlightControl::land ( )`

Wyzwala tryb lądowania w obecnym punkcie.

Zwraca

`true` Pomyślnie aktywowano tryb lądowanie.

`false` Wystąpił błąd podczas aktywowania trybu lądowania.

**3.2.2.9 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.

**3.2.2.10 printTelem()** `void FlightControl::printTelem ( )`

Wyświetla w konsoli podstawowe dane telemetryczne.

**3.2.2.11 setConnectionURL()** `void FlightControl::setConnectionURL (   
const std::string url )`

Ustawia wartość pola connectionURL.

Parametry

<i>url</i>	Nowy url do ustawienia.
------------	-------------------------

**3.2.2.12 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.

**3.2.2.13 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.

**3.2.2.14 stopOffboard()** `bool FlightControl::stopOffboard ( )`

Zatrzymuje kontrolę w trybie offboard.

**Zwraca**

true Pomyślnie zatrzymano kontrolę offboard.  
false Wystąpił błąd podczas zatrzymywania kontroli offboard.

**3.2.2.15 subscribeTelem()** `void FlightControl::subscribeTelem ( ) [private]`

Subskrybuje wymagane tematy.

### 3.2.3 Dokumentacja atrybutów składowych

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

Pozwala na wykonywanie prostych akcji.

**3.2.3.2 connectionURL** `std::string FlightControl::connectionURL [private]`

Adres połączenia protokołu MAVLink.

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

Dostarcza informacji o systemie.

**3.2.3.4 mavsdk** `mavsdk::Mavsdk FlightControl::mavsdk [private]`

Pozwala na zarządzanie połączeniami.

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

Pozwala na kontrolę w trybie offboard.

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

Obiekt reprezentujący system (autopilot)

**3.2.3.7 telemData** `TelemetryData FlightControl::telemData [private]`

Przechowuje dane telemetryczne odebrane przez MAVLink.

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

Pozwala na odbiór danych telemetrycznych.

Dokumentacja dla tej klasy została wygenerowana z plików:

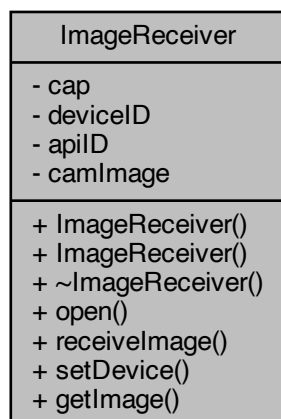
- `include/FlightAutonomy/FlightControl.h`
- `src/FlightControl.cpp`

### 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`  
*Niszczyciel obiektu `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 przechwytyjący 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

<code>_deviceId</code>	
<code>_apiID</code>	

## 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 przechwytyjący stream video.

**3.3.4.4 deviceId** `int ImageReceiver::deviceId = 0 [private]`

Kamera 0 - domyślna.

Dokumentacja dla tej klasy została wygenerowana z plików:

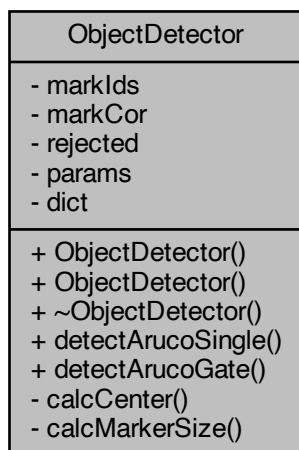
- [include/FlightAutonomy/ImageReceiver.h](#)
- [src/ImageReceiver.cpp](#)

### 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 wykrytego 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

<code>_params</code>	Parametry algorytmu wykrywania znaczników.
<code>_dict</code>	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

<i>corners</i>	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

<i>corners</i>	Wektor zawierający koordynaty kolejnych wierzchołków znacznika.
<i>imgHeight</i>	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::detect↵`  
`ArucoGate (`  
`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

<i>img</i>	Ramka obrazu na której mają zostać wykryte znaczniki.
<i>arucoIDs</i>	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

<i>img</i>	Ramka obrazu na której mają zostać wykryte znaczniki.
<i>arucoID</i>	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:

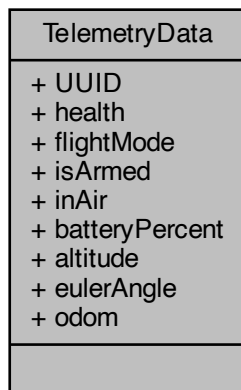
- [include/FlightAutonomy/ObjectDetector.h](#)
- [src/ObjectDetector.cpp](#)

### 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.

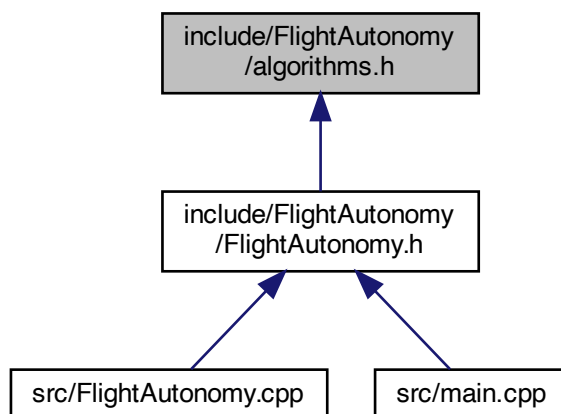
Dokumentacja dla tej struktury została wygenerowana z pliku:

- `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.](#)

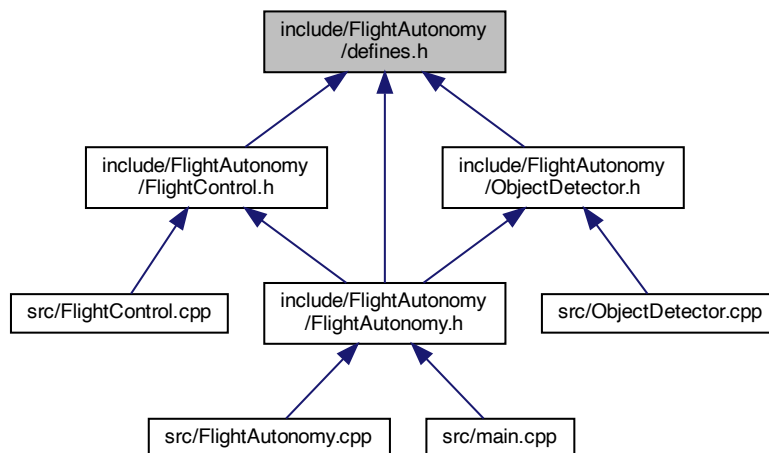
```

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`

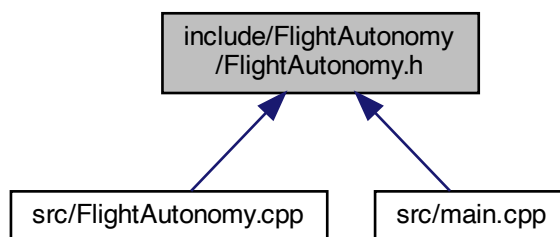
Maksymalna wartość prędkości kątowej wokół osi Z (yaw) w stopniach na sekundę

**4.3.2.10 MID\_THRESHOLD** `const float MID_THRESHOLD = 0.4`

Próg do którego przyjmowana jest odległość jako bliska środka obrazu.



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
74     bool spinOnce();
75
76     bool ok();
77
78     bool stop();
79
80     int getExitCode();
81
82     void printExitStatus();
83
84 private:
85     int performStep(cv::Mat &img);
86
87     bool landingStep(cv::Mat &img);
88
89     mavsdk::Offboard::VelocityBodyYawspeed landingCalcVelo(cv::Mat &img, cv::Point2f arucoPosition);
90
91     bool avoidingStep(cv::Mat &img);
92
93     mavsdk::Offboard::VelocityBodyYawspeed avoidingCalcVelo(cv::Mat &img, cv::Point2f gatePosition,
94     float angle);
95
96     void checkTimeouts();
97 };

```

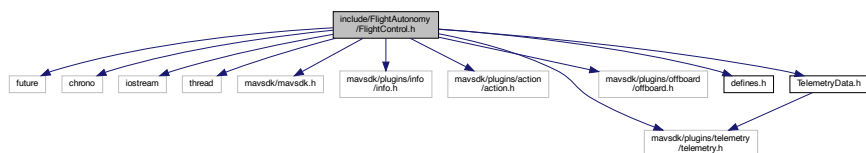
## 4.7 Dokumentacja pliku include/FlightAutonomy/FlightControl.h

```

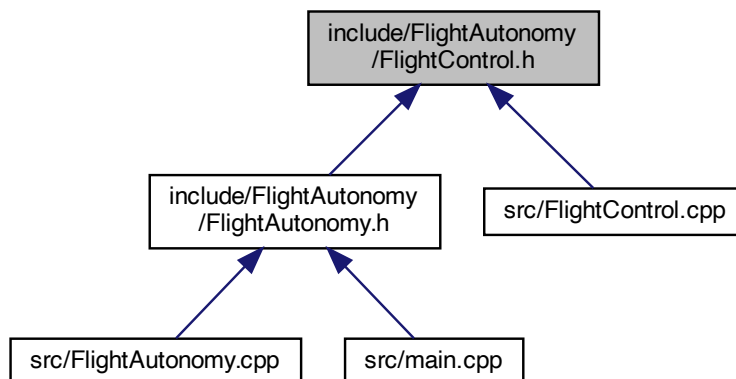
#include <future>
#include <chrono>
#include <iostream>
#include <thread>
#include <mavsdk/mavsdk.h>
#include <mavsdk/plugins/info/info.h>
#include <mavsdk/plugins/action/action.h>
#include <mavsdk/plugins/telemetry/telemetry.h>
#include <mavsdk/plugins/offboard/offboard.h>
#include "defines.h"
#include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

```

Wykres zależności załączania dla FlightControl.h:



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



## Komponenty

- class [FlightControl](#)

## 4.8 FlightControl.h

[Idź do dokumentacji tego pliku.](#)

```

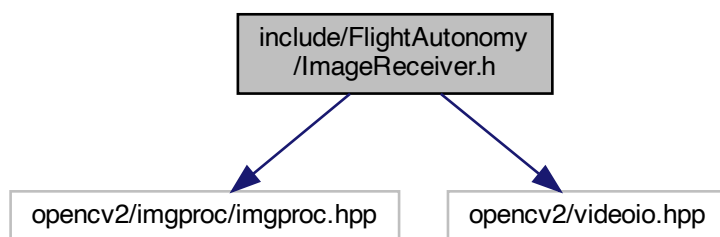
1 #pragma once
2
3 #include <future>
4 #include <chrono>
5 #include <iostream>
6 #include <thread>
7 #include <mavsdk/mavsdk.h>
8 #include <mavsdk/plugins/info/info.h>
9 #include <mavsdk/plugins/action/action.h>
10 #include <mavsdk/plugins/telemetry/telemetry.h>
11 #include <mavsdk/plugins/offboard/offboard.h>
12
13 #include "defines.h"
14 #include "TelemetryData.h"
15
16 class FlightControl
17 {
18     std::string connectionURL;
19
20     mavsdk::Mavsdk mavsdk;
21     std::shared_ptr<mavsdk::System> system;
22     std::shared_ptr<mavsdk::Info> info;
23     std::shared_ptr<mavsdk::Telemetry> telemetry;
24     std::shared_ptr<mavsdk::Action> action;
25     std::shared_ptr<mavsdk::Offboard> offboard;
26
27     TelemetryData telemData;
28
29     std::shared_ptr<mavsdk::System> getSystem();
30
31     void subscribeTelem();
32
33 public:
34     FlightControl() = default;
35
36     ~FlightControl() = default;
37
38     bool connect();
39
40     bool checkStatus();
  
```

```
68
74     mavsdk::Telemetry::EulerAngle getEulerAngle();
75
81     void setConnectionURL(const std::string url);
82
86     void printTelem();
87
94     bool observeInAir();
95
102    bool startOffboard();
103
110    bool stopOffboard();
111
118    bool setOffboardVelo(mavsdk::Offboard::VelocityBodyYawspeed veloBodyYawspeed);
119
126    bool land();
127
133    float getAltitude();
134
140    mavsdk::Telemetry::FlightMode getFlightMode();
141
149    bool checkVelo(mavsdk::Offboard::VelocityBodyYawspeed &velocities);
150 };
```

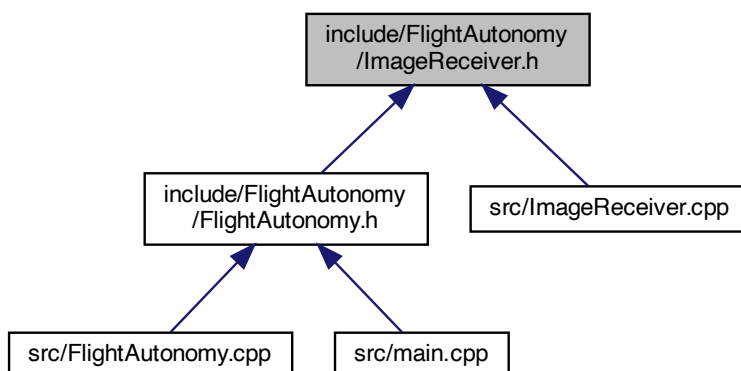
## 4.9 Dokumentacja pliku include/FlightAutonomy/ImageReceiver.h

```
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/videoio/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
6 class ImageReceiver
7 {
8     cv::VideoCapture cap;
9     int deviceID = 0;
10    int apiID = cv::CAP_ANY;
11    cv::Mat camImage;
12
13 public:
14    ImageReceiver() = default;
15    ImageReceiver(int deviceID, int apiID = 0);
16    ~ImageReceiver() = default;
17
18    bool open();
19
20    void receiveImage();
21
22    void setDevice(int _deviceID, int _apiID = 0);
23
24    cv::Mat getImage();
25 };
  
```

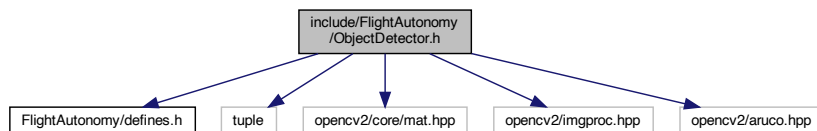
## 4.11 Dokumentacja pliku include/FlightAutonomy/ObjectDetector.h

```

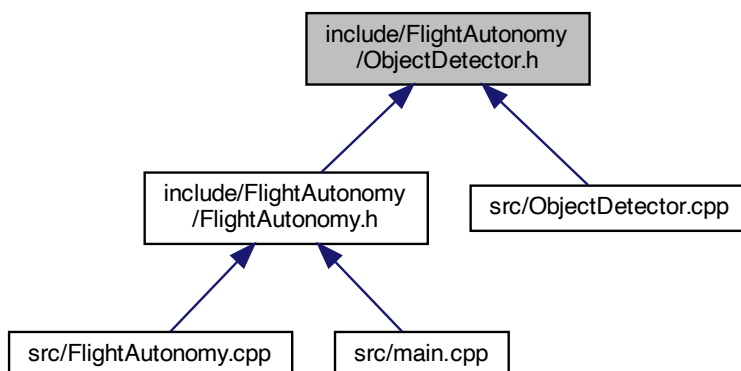
#include "FlightAutonomy/defines.h"
#include <tuple>
  
```

```
#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 }
```

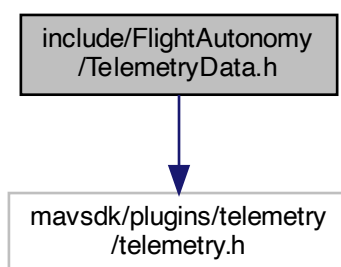


```
19     cv::Ptr<cv::aruco::DetectorParameters> params;
20     cv::Ptr<cv::aruco::Dictionary> dict;
21
22 public:
23     ObjectDetector();
24
25     ObjectDetector(cv::Ptr<cv::aruco::DetectorParameters> _params, cv::Ptr<cv::aruco::Dictionary> _dict);
26
27     ~ObjectDetector() = default;
28
29     cv::Point2f detectArucoSingle(const cv::Mat &img, int arucoID);
30
31     std::tuple<cv::Point2f, float, float> detectArucoGate(const cv::Mat &img, int arucoIDs[4]);
32
33 private:
34     cv::Point2f calcCenter(std::vector<cv::Point2f> corners);
35
36     float calcMarkerSize(std::vector<cv::Point2f> corners, int imgHeight);
37 };
```

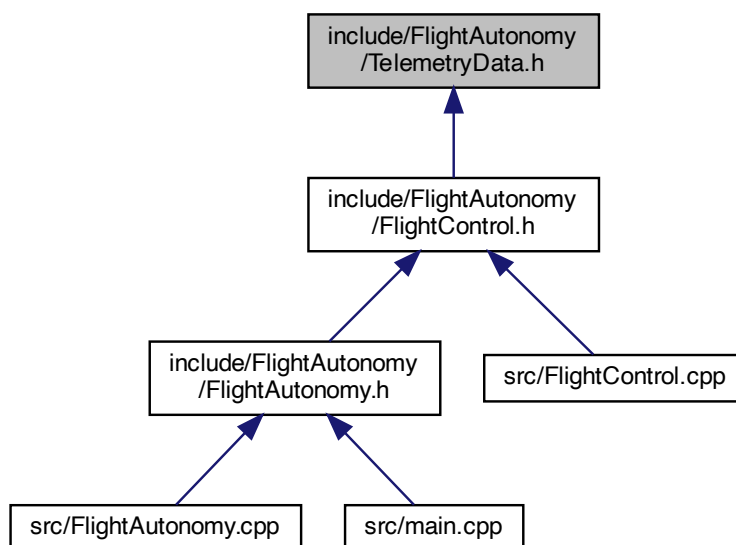
## 4.13 Dokumentacja pliku include/FlightAutonomy/TelemetryData.h

```
#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.](#)

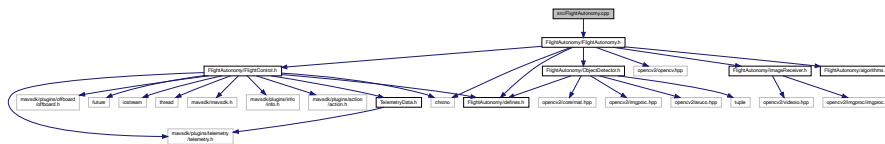
```

1 #pragma once
2
3 #include <mavsdk/plugins/telemetry/telemetry.h>
4
5
6
7
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"
```

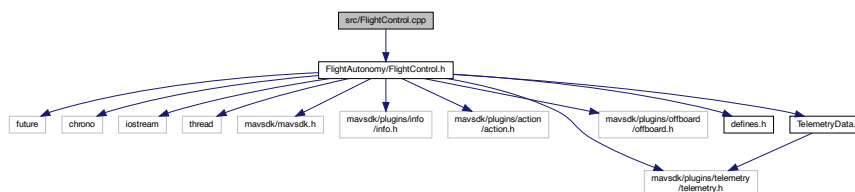
Wykres zależności załączania dla FlightAutonomy.cpp:



## 4.16 Dokumentacja pliku src/FlightControl.cpp

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

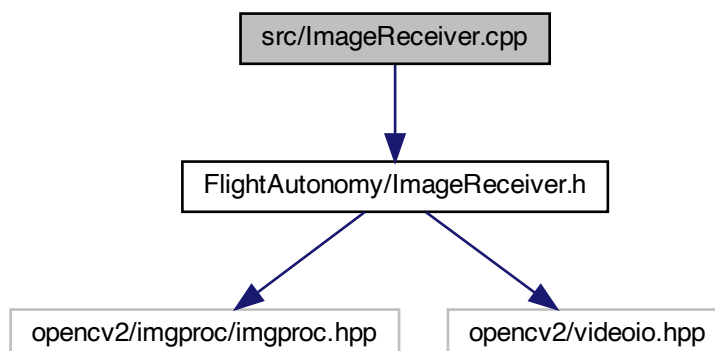
Wykres zależności załączania dla FlightControl.cpp:



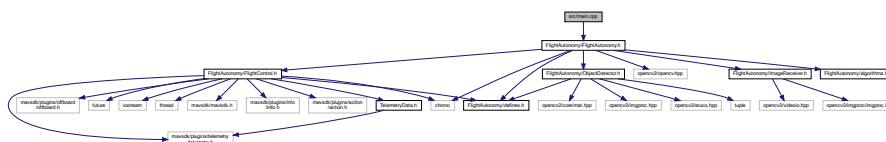
## 4.17 Dokumentacja pliku src/ImageReceiver.cpp

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

Wykres zależności załączania dla ImageReceiver.cpp:



Wykres zależności załączania dla main.cpp:



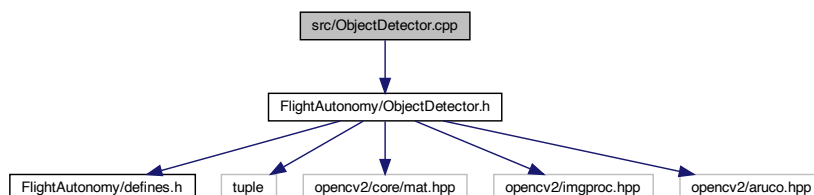
- void **printHelp** (std::string progName)
- int **main** (int argc, char \*\*argv)

#### 4.18.1 Dokumentacja funkcji

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

#### 4.19 Dokumentacja pliku src/ObjectDetector.cpp

Wykres zależności załączania dla ObjectDetector.cpp:



## Indeks

- ~FlightAutonomy
  - FlightAutonomy, [5](#)
- ~FlightControl
  - FlightControl, [13](#)
- ~ImageReceiver
  - ImageReceiver, [19](#)
- ~ObjectDetector
  - ObjectDetector, [23](#)
- action
  - FlightControl, [16](#)
- Algorithms
  - algorithms.h, [28](#)
- algorithms.h
  - Algorithms, [28](#)
  - arucoLanding, [29](#)
  - forwardFlight, [29](#)
  - gateRacing, [29](#)
  - normalLanding, [29](#)
- altitude
  - TelemetryData, [27](#)
- apiID
  - ImageReceiver, [21](#)
- arucoLanding
  - algorithms.h, [29](#)
- avoidingCalcVelo
  - FlightAutonomy, [5](#)
- avoidingStep
  - FlightAutonomy, [5](#)
- batteryPercent
  - TelemetryData, [27](#)
- calcCenter
  - ObjectDetector, [23](#)
- calcMarkerSize
  - ObjectDetector, [24](#)
- CAM\_HFOV
  - defines.h, [30](#)
- CAM\_VFOV
  - defines.h, [30](#)
- camImage
  - ImageReceiver, [21](#)
- cap
  - ImageReceiver, [21](#)
- checkStatus
  - FlightControl, [13](#)
- checkTimeouts
  - FlightAutonomy, [7](#)
- checkVelo
  - FlightControl, [13](#)
- connect
  - FlightAutonomy, [7](#)
  - FlightControl, [14](#)
- connectionURL
  - FlightControl, [16](#)

- currAlg
  - FlightAutonomy, [9](#)
- defines.h
  - CAM\_HFOV, [30](#)
  - CAM\_VFOV, [30](#)
  - FA\_DEBUG, [30](#)
  - FORWARD\_FLIGHT\_TIMEOUT, [30](#)
  - IS\_STRAIGHT\_ANGLE, [31](#)
  - LAND\_ALT, [31](#)
  - MAX\_FLIGHT\_ALT, [31](#)
  - MAX\_VELO\_HORI\_MS, [31](#)
  - MAX\_VELO\_VERT\_MS, [31](#)
  - MAX\_YAWSPEED, [31](#)
  - MID\_THRESHOLD, [31](#)
  - NO\_DETECT\_TIMEOUT, [31](#)
  - RACING\_ALT, [32](#)
- detectArucoGate
  - ObjectDetector, [24](#)
- detectArucoSingle
  - ObjectDetector, [24](#)
- deviceId
  - ImageReceiver, [21](#)
- dict
  - ObjectDetector, [25](#)
- eulerAngle
  - TelemetryData, [27](#)
- exitCode
  - FlightAutonomy, [10](#)
- FA\_DEBUG
  - defines.h, [30](#)
- FlightAutonomy, [2](#)
  - ~FlightAutonomy, [5](#)
  - avoidingCalcVelo, [5](#)
  - avoidingStep, [5](#)
  - checkTimeouts, [7](#)
  - connect, [7](#)
  - currAlg, [9](#)
  - exitCode, [10](#)
  - FlightAutonomy, [5](#)
  - flightCtrl, [10](#)
  - gateArucos, [10](#)
  - getExitCode, [7](#)
  - imgRec, [10](#)
  - isReady, [7](#)
  - landingCalcVelo, [7](#)
  - landingPadID, [10](#)
  - landingStep, [8](#)
  - objDetect, [10](#)
  - ok, [8](#)
  - performStep, [8](#)
  - printExitStatus, [9](#)
  - readArgs, [9](#)
  - spinOnce, [9](#)

- stop, 9
- timeoutCounter, 10
- FlightControl, 11
  - ~FlightControl, 13
  - action, 16
  - checkStatus, 13
  - checkVelo, 13
  - connect, 14
  - connectionURL, 16
  - FlightControl, 13
  - getAltitude, 14
  - getEulerAngle, 14
  - getFlightMode, 14
  - getSystem, 14
  - info, 17
  - land, 15
  - mavsdk, 17
  - observeInAir, 15
  - offboard, 17
  - printTelem, 15
  - setConnectionURL, 15
  - setOffboardVelo, 15
  - startOffboard, 16
  - stopOffboard, 16
  - subscribeTelem, 16
  - system, 17
  - telemData, 17
  - telemetry, 17
- flightCtrl
  - FlightAutonomy, 10
- flightMode
  - TelemetryData, 27
- FORWARD\_FLIGHT\_TIMEOUT
  - defines.h, 30
- forwardFlight
  - algorithms.h, 29
- gateArucos
  - FlightAutonomy, 10
- gateRacing
  - algorithms.h, 29
- getAltitude
  - FlightControl, 14
- getEulerAngle
  - FlightControl, 14
- getExitCode
  - FlightAutonomy, 7
- getFlightMode
  - FlightControl, 14
- getImage
  - ImageReceiver, 19
- getSystem
  - FlightControl, 14
- health
  - TelemetryData, 27
- ImageReceiver, 18
  - ~ImageReceiver, 19
- apiID, 21
- camImage, 21
- cap, 21
- deviceId, 21
- getImage, 19
- ImageReceiver, 19
- open, 19
- receiveImage, 19
- setDevice, 20
- imgRec
  - FlightAutonomy, 10
- inAir
  - TelemetryData, 27
- include/FlightAutonomy/algorithms.h, 28, 29
- include/FlightAutonomy/defines.h, 29, 32
- include/FlightAutonomy/FlightAutonomy.h, 32, 33
- include/FlightAutonomy/FlightControl.h, 34, 35
- include/FlightAutonomy/ImageReceiver.h, 36, 37
- include/FlightAutonomy/ObjectDetector.h, 37, 38
- include/FlightAutonomy/TelemetryData.h, 39, 40
- info
  - FlightControl, 17
- IS\_STRAIGHT\_ANGLE
  - defines.h, 31
- isArmed
  - TelemetryData, 27
- isReady
  - FlightAutonomy, 7
- land
  - FlightControl, 15
- LAND\_ALT
  - defines.h, 31
- landingCalcVelo
  - FlightAutonomy, 7
- landingPadID
  - FlightAutonomy, 10
- landingStep
  - FlightAutonomy, 8
- main
  - main.cpp, 42
- main.cpp
  - main, 42
  - printHelp, 42
- markCor
  - ObjectDetector, 25
- markIds
  - ObjectDetector, 25
- mavsdk
  - FlightControl, 17
- MAX\_FLIGHT\_ALT
  - defines.h, 31
- MAX\_VELO\_HORI\_MS
  - defines.h, 31
- MAX\_VELO\_VERT\_MS
  - defines.h, 31
- MAX\_YAWSPEED
  - defines.h, 31

- MID\_THRESHOLD
  - defines.h, [31](#)
- NO\_DETECT\_TIMEOUT
  - defines.h, [31](#)
- normalLanding
  - algorithms.h, [29](#)
- objDetect
  - FlightAutonomy, [10](#)
- ObjectDetector, [22](#)
  - ~ObjectDetector, [23](#)
  - calcCenter, [23](#)
  - calcMarkerSize, [24](#)
  - detectArucoGate, [24](#)
  - detectArucoSingle, [24](#)
  - dict, [25](#)
  - markCor, [25](#)
  - markIds, [25](#)
  - ObjectDetector, [23](#)
  - params, [25](#)
  - rejected, [25](#)
- observeInAir
  - FlightControl, [15](#)
- odom
  - TelemetryData, [27](#)
- offboard
  - FlightControl, [17](#)
- ok
  - FlightAutonomy, [8](#)
- open
  - ImageReceiver, [19](#)
- params
  - ObjectDetector, [25](#)
- performStep
  - FlightAutonomy, [8](#)
- printExitStatus
  - FlightAutonomy, [9](#)
- printHelp
  - main.cpp, [42](#)
- printTelem
  - FlightControl, [15](#)
- RACING\_ALT
  - defines.h, [32](#)
- readArgs
  - FlightAutonomy, [9](#)
- receiveImage
  - ImageReceiver, [19](#)
- rejected
  - ObjectDetector, [25](#)
- setConnectionURL
  - FlightControl, [15](#)
- setDevice
  - ImageReceiver, [20](#)
- setOffboardVelo
  - FlightControl, [15](#)
- spinOnce
  - FlightAutonomy, [9](#)
  - src/FlightAutonomy.cpp, [41](#)
  - src/FlightControl.cpp, [41](#)
  - src/ImageReceiver.cpp, [41](#)
  - src/main.cpp, [42](#)
  - src/ObjectDetector.cpp, [42](#)
- startOffboard
  - FlightControl, [16](#)
- stop
  - FlightAutonomy, [9](#)
- stopOffboard
  - FlightControl, [16](#)
- subscribeTelem
  - FlightControl, [16](#)
- system
  - FlightControl, [17](#)
- telemData
  - FlightControl, [17](#)
- telemetry
  - FlightControl, [17](#)
- TelemetryData, [26](#)
  - altitude, [27](#)
  - batteryPercent, [27](#)
  - eulerAngle, [27](#)
  - flightMode, [27](#)
  - health, [27](#)
  - inAir, [27](#)
  - isArmed, [27](#)
  - odom, [27](#)
  - UUID, [28](#)
- timeoutCounter
  - FlightAutonomy, [10](#)
- UUID
  - TelemetryData, [28](#)