

Kompas elektroniczny

Wygenerowano przez Doxygen 1.8.11



# Spis treści

<b>1</b>	<b>Indeks hierarchiczny</b>	<b>1</b>
1.1	Hierarchia klas . . . . .	1
<b>2</b>	<b>Indeks klas</b>	<b>3</b>
2.1	Lista klas . . . . .	3
<b>3</b>	<b>Dokumentacja klas</b>	<b>5</b>
3.1	Dokumentacja klasy Chart . . . . .	5
3.1.1	Opis szczegółowy . . . . .	6
3.1.2	Dokumentacja funkcji składowych . . . . .	6
3.1.2.1	Update . . . . .	6
3.2	Dokumentacja klasy Compass . . . . .	6
3.2.1	Opis szczegółowy . . . . .	7
3.3	Dokumentacja unii Data . . . . .	7
3.3.1	Opis szczegółowy . . . . .	8
3.4	Dokumentacja unii data_t . . . . .	8
3.4.1	Opis szczegółowy . . . . .	8
3.5	Dokumentacja klasy MainWindow . . . . .	9
3.5.1	Opis szczegółowy . . . . .	10
3.5.2	Dokumentacja konstruktora i destruktora . . . . .	10
3.5.2.1	MainWindow(float *source, QWidget *parent=0) . . . . .	10
3.5.3	Dokumentacja funkcji składowych . . . . .	10
3.5.3.1	getMeasures() const . . . . .	10
3.6	Dokumentacja klasy Measures . . . . .	10

3.6.1	Opis szczegółowy . . . . .	11
3.6.2	Dokumentacja konstruktora i destruktora . . . . .	11
3.6.2.1	Measures(float *source, QWidget *parent=0) . . . . .	11
3.6.3	Dokumentacja funkcji składowych . . . . .	12
3.6.3.1	CalculateAngles(float *roll, float *pitch, float *heading) . . . . .	12
3.6.3.2	ReturnData(float *data) . . . . .	12
3.6.3.3	UpdateData . . . . .	12
3.7	Dokumentacja klasy SerialPort . . . . .	12
3.7.1	Opis szczegółowy . . . . .	13
3.7.2	Dokumentacja konstruktora i destruktora . . . . .	13
3.7.2.1	SerialPort(const char *device, int baudrate=9600, const char *mode=""8N1"") . . . . .	13
3.7.3	Dokumentacja funkcji składowych . . . . .	13
3.7.3.1	Connect(const char *device, int baudrate, const char *mode) . . . . .	13
3.7.3.2	GetArray(unsigned char *buffer, int len) . . . . .	13

# Rozdział 1

## Indeks hierarchiczny

### 1.1 Hierarchia klas

Ta lista dziedziczenia posortowana jest z grubsza, choć nie całkowicie, alfabetycznie:

Data . . . . .	7
data_t . . . . .	8
QChart	
Chart . . . . .	5
QLabel	
Compass . . . . .	6
Measures . . . . .	10
QMainWindow	
MainWindow . . . . .	9
SerialPort . . . . .	12



## Rozdział 2

# Indeks klas

### 2.1 Lista klas

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

<a href="#">Chart</a>	Klasa obsługująca wykres . . . . .	5
<a href="#">Compass</a>	Klasa obsługująca obiekt 3D . . . . .	6
<a href="#">Data</a>	Struktura przechowująca odczyty . . . . .	7
<a href="#">data_t</a>	Struktura przechowująca odbieraną ramkę . . . . .	8
<a href="#">MainWindow</a>	Klasa okna głównego . . . . .	9
<a href="#">Measures</a>	Klasa wyświetlająca i przetwarzająca pomiary . . . . .	10
<a href="#">SerialPort</a>	Klasa portu szeregowego . . . . .	12





## Rozdział 3

# Dokumentacja klas

### 3.1 Dokumentacja klasy Chart

klasa obsługująca wykres

```
#include <chart.h>
```

Diagram dziedziczenia dla Chart

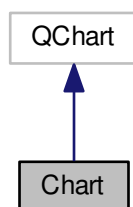
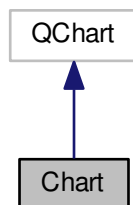


Diagram współpracy dla Chart:



## Sloty publiczne

- void [Update](#) ()

## Metody publiczne

- **Chart** (QGraphicsItem \*parent=0)

### 3.1.1 Opis szczegółowy

klasa obsługująca wykres

klasa umożliwia aktualizację wykresu

### 3.1.2 Dokumentacja funkcji składowych

#### 3.1.2.1 void Chart::Update ( ) [slot]

slot umożliwiający aktualizację wykresu w czasie

Dokumentacja dla tej klasy została wygenerowana z plików:

- chart.h
- chart.cpp

## 3.2 Dokumentacja klasy Compass

Klasa obsługująca obiekt 3D.

```
#include <compass.h>
```

Diagram dziedziczenia dla Compass

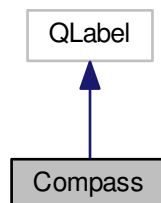
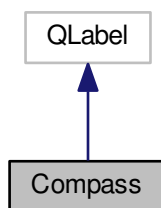


Diagram współpracy dla Compass:



### Metody publiczne

- **Compass** (QWidget \*parent=0)

#### 3.2.1 Opis szczegółowy

Klasa obsługująca obiekt 3D.

klasa odpowiada za wczytanie obiektu 3D oraz jego wyświetlanie oraz zmianę orientacji

Dokumentacja dla tej klasy została wygenerowana z plików:

- compass.h
- compass.cpp

## 3.3 Dokumentacja unii Data

struktura przechowująca odczyty

```
#include <measures.h>
```

### Atrybuty publiczne

- float **data** [6]
- struct {
  - float **ACC\_X**  
*odczyt akcelerometru dla osi x*
  - float **ACC\_Y**  
*odczyt akcelerometru dla osi y*
  - float **ACC\_Z**  
*odczyt akcelerometru dla osi z*
  - float **MAG\_X**  
*odczyt magnetometru dla osi x*
  - float **MAG\_Z**  
*odczyt magnetometru dla osi z*
  - float **MAG\_Y**  
*odczyt magnetometru dla osi y*

### 3.3.1 Opis szczegółowy

struktura przechowująca odczyty

Dokumentacja dla tej unii została wygenerowana z pliku:

- measures.h

## 3.4 Dokumentacja unii data\_t

struktura przechowująca odbieraną ramkę

```
#include <types.h>
```

### Atrybuty publiczne

- struct {  
    uint8\_t **flag**  
        *znak początkowy ramki*  
    datatype\_t **ACC\_Data** [3]  
        *odczyty akcelerometru*  
    datatype\_t **MAG\_Data** [3]  
        *odczyty magnetometru*  
    datatype\_t **checksum**  
        *suma kontrolna*  
};
- struct {  
    uint8\_t **f**  
    datatype\_t **data** [7]  
};
- uint8\_t **buffer** [1+7 \*sizeof(datatype\_t)]

### 3.4.1 Opis szczegółowy

struktura przechowująca odbieraną ramkę

Dokumentacja dla tej unii została wygenerowana z pliku:

- types.h

## 3.5 Dokumentacja klasy MainWindow

klasa okna głównego

```
#include <mainwindow.h>
```

Diagram dziedziczenia dla MainWindow

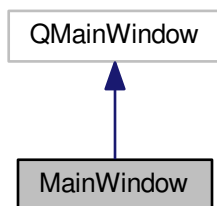
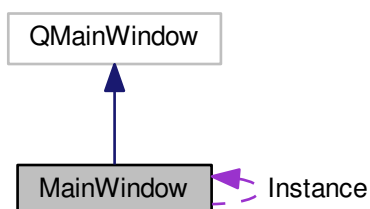


Diagram współpracy dla MainWindow:



### Metody publiczne

- [MainWindow](#) (float \*source, QWidget \*parent=0)  
*konstruktor inicjalizujący interfejs programu*
- [Measures](#) \* [getMeasures](#) () const  
*akcesor zmiennej measures*

### Statyczne atrybuty publiczne

- static [MainWindow](#) \* [Instance](#)  
*zmienna statyczna będąca wskaźnikiem na instancję okna głównego*

### 3.5.1 Opis szczegółowy

klasa okna głównego

klasa inicjalizuje interfejs programu oraz łączy sygnały i sloty dla aktualizacji pomiarów, wykresu oraz obiektu 3D

### 3.5.2 Dokumentacja konstruktora i destruktor

**3.5.2.1** `MainWindow::MainWindow ( float * source, QWidget * parent = 0 ) [explicit]`

konstruktor inicjalizujący interfejs programu

Parametry

in	source	wskaźnik na dane modyfikowane przez port szeregowy
----	--------	--

### 3.5.3 Dokumentacja funkcji składowych

**3.5.3.1** `Measures * MainWindow::getMeasures ( ) const`

akcesor zmiennej measures

Zwraca

wskaźnik na widget [Measures](#)

Dokumentacja dla tej klasy została wygenerowana z plików:

- mainwindow.h
- mainwindow.cpp

## 3.6 Dokumentacja klasy Measures

klasa wyświetlająca i przetwarzająca pomiary

```
#include <measures.h>
```

Diagram dziedziczenia dla Measures

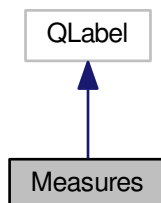
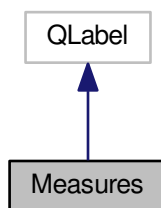


Diagram współpracy dla Measures:



### Sloty publiczne

- void [UpdateData](#) ()

### Metody publiczne

- [Measures](#) (float \*source, QWidget \*parent=0)  
*konstruktor klasy [Measures](#)*
- void [CalculateAngles](#) (float \*roll, float \*pitch, float \*heading)  
*funkcja oblicza orientację urządzenia na podstawie przechowywanych pomiarów*
- void [ReturnData](#) (float \*data)  
*akcesor danych pomiarowych*

#### 3.6.1 Opis szczegółowy

klasa wyświetlająca i przetwarzająca pomiary

klasa wyświetla otrzymane pomiary oraz pozwala na ich przetwarzanie

#### 3.6.2 Dokumentacja konstruktora i destruktora

##### 3.6.2.1 Measures::Measures ( float \* source, QWidget \* parent = 0 )

konstruktor klasy [Measures](#)

##### Parametry

in	source	źródło danych wejściowych
----	--------	---------------------------

### 3.6.3 Dokumentacja funkcji składowych

#### 3.6.3.1 void Measures::CalculateAngles ( float \* *roll*, float \* *pitch*, float \* *heading* )

funkcja oblicza orientację urządzenia na podstawie przechowywanych pomiarów

Parametry

out	<i>roll</i>	obrót wokół osi x
out	<i>pitch</i>	obrót wokół osi y
out	<i>heading</i>	obrót igły magnetycznej

#### 3.6.3.2 void Measures::ReturnData ( float \* *data* )

akcesor danych pomiarowych

Parametry

out	<i>data</i>	wskaźnik na przechowywane dane pomiarowe czujników
-----	-------------	--

#### 3.6.3.3 void Measures::UpdateData ( ) [slot]

slot umożliwiający aktualizację danych pomiarowych

Dokumentacja dla tej klasy została wygenerowana z plików:

- measures.h
- measures.cpp

## 3.7 Dokumentacja klasy SerialPort

klasa portu szeregowego

```
#include <serialport.h>
```

### Metody publiczne

- [SerialPort](#) (const char \*device, int baudrate=9600, const char \*mode="8N1")  
*konstruktor konfiguruje port szeregowy*
- int [Connect](#) (const char \*device, int baudrate, const char \*mode)  
*funkcja umożliwia połączenie do portu szeregowego*
- void [Disconnect](#) ()  
*funkcja odłącza program od portu szeregowego*
- int [GetArray](#) (unsigned char \*buffer, int len)  
*funkcja pozwala na odbieranie danych przesyłanych poprzez port szeregowy*



### 3.7.1 Opis szczegółowy

klasa portu szeregowego

klasa pozwala na połączenie się z portem szeregowym oraz odczytywanie i wysyłanie danych

### 3.7.2 Dokumentacja konstruktora i destruktora

#### 3.7.2.1 SerialPort::SerialPort ( const char \* *device*, int *baudrate* = 9600, const char \* *mode* = "8N1" )

konstruktor konfigurujący port szeregowy

##### Parametry

in	<i>device</i>	nazwa portu szeregowego
in	<i>baudrate</i>	prędkość transmisji danych
in	<i>mode</i>	parametry konfiguracyjne portu

### 3.7.3 Dokumentacja funkcji składowych

#### 3.7.3.1 int SerialPort::Connect ( const char \* *device*, int *baudrate*, const char \* *mode* )

funkcja umożliwia połączenie do portu szeregowego

##### Parametry

in	<i>device</i>	nazwa portu szeregowego
in	<i>baudrate</i>	prędkość transmisji danych
in	<i>mode</i>	parametry konfiguracyjne portu

#### 3.7.3.2 int SerialPort::GetArray ( unsigned char \* *buffer*, int *len* )

funkcja pozwala na odbieranie danych przesyłanych poprzez port szeregowy

##### Parametry

out	<i>buffer</i>	dane odbierane przez port
in	<i>len</i>	ilość bajtów które chcemy odebrać

##### Zwraca

ilość odebranych bajtów, -1 jeśli odbiór danych się nie powiódł

Dokumentacja dla tej klasy została wygenerowana z plików:

- `serialport.h`
- `serialport.cpp`