System Monitorowania Czujników

Wygenerowano przez Doxygen 1.9.4

Chapter 1

System Monitorowania Czujników

System do monitorowania i wizualizacji danych z czujników środowiskowych:

- CO2
- Pyłów zawieszonych (PM1.0, PM2.5, PM10)
- Promieniowania
- Temperatury
- Wilgotności

1.1 Główne funkcje

- Odczyt danych w czasie rzeczywistym
- Wizualizacja na wykresach
- Zapis do pliku CSV
- Analiza historyczna
- Wsparcie wielojęzyczności

Chapter 2

Indeks hierarchiczny

2.1 Hierarchia klas

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

Chapter 3

Indeks struktur danych

3.1 Struktury danych

Tutaj znajdują się struktury danych wraz z ich krótkimi opisami:

Chapter 4

Indeks plików

4.1 Lista plików

Tutaj znajduje się lista wszystkich plików z ich krótkimi opisami:

Chapter 5

Dokumentacja struktur danych

5.1 Dokumentacja klasy MainWindow

Główne okno aplikacji wyświetlające dane z czujników i wykresy.

`#include <MainWindow.h>`

Diagram dziedziczenia dla MainWindow

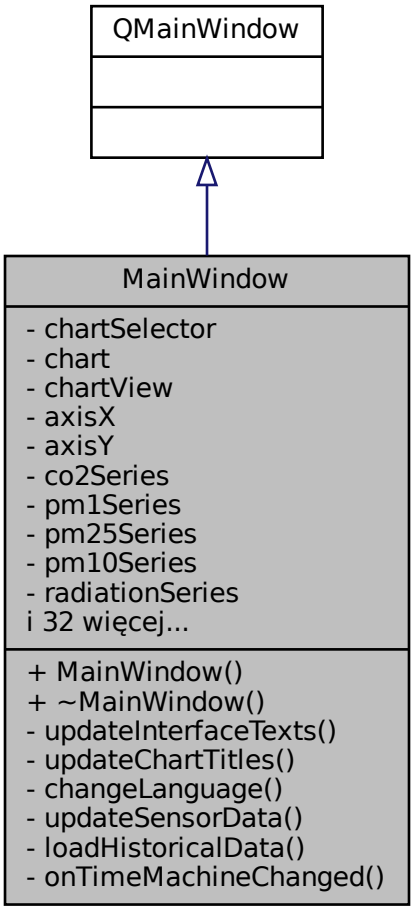
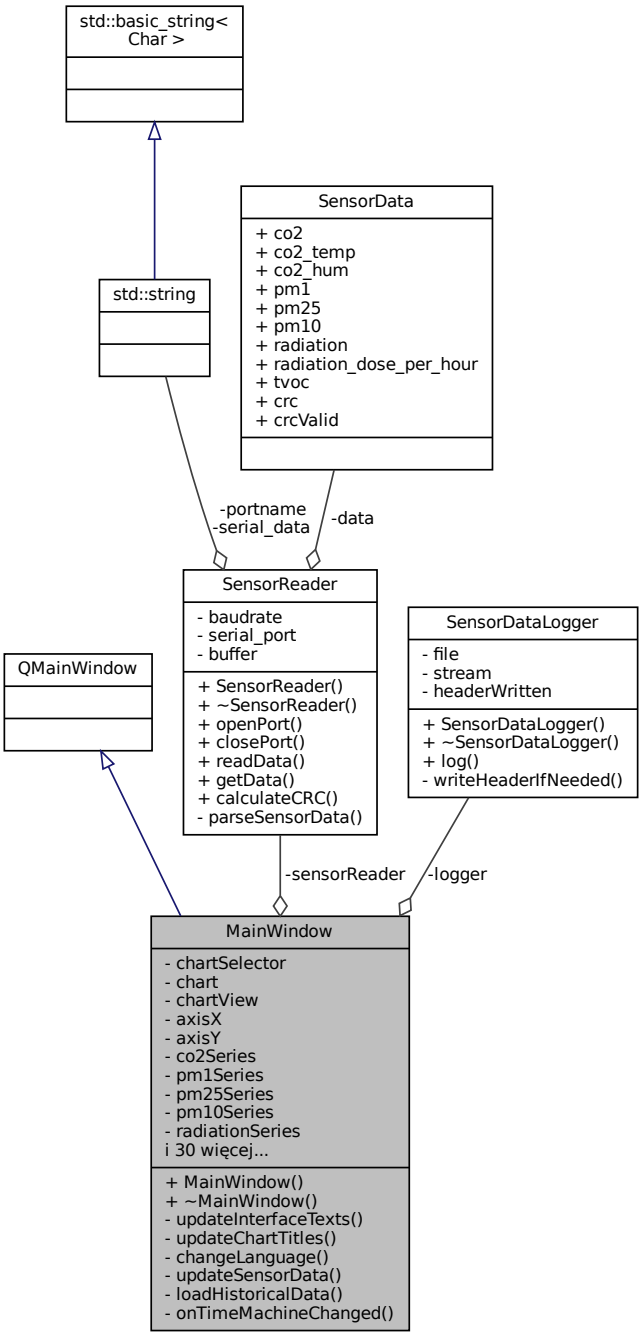


Diagram współpracy dla MainWindow:



Metody publiczne

- MainWindow (SensorReader ∗reader, QWidget ∗parent=nullptr)

  Konstruktor głównego okna.
- ∼MainWindow ()

Sloty prywatne

- void updateSensorData ()

  Aktualizuje interfejs danymi z czujników.
- void loadHistoricalData ()

  Wczytuje i wyświetla dane historyczne.
- void onTimeMachineChanged (int id)

  Obsługuje zmianę zakresu czasu na wykresie.

Metody prywatne

- void updateInterfaceTexts ()

  Aktualizuje teksty interfejsu po zmianie języka.
- void updateChartTitles ()

  Aktualizuje tytuły wykresów.
- void changeLanguage (const QString &language)

  Zmienia język interfejsu.

Atrybuty prywatne

- QComboBox * chartSelector

  Komponenty wykresu.
- QtCharts::QChart * chart
- QtCharts::QChartView * chartView
- QtCharts::QDateTimeAxis * axisX
- QtCharts::QValueAxis * axisY
- QtCharts::QLineSeries * co2Series
- QtCharts::QLineSeries * pm1Series
- QtCharts::QLineSeries * pm25Series
- QtCharts::QLineSeries * pm10Series
- QtCharts::QLineSeries * radiationSeries
- QtCharts::QLineSeries * temperatureSeries
- QtCharts::QLineSeries * humiditySeries
- QtCharts::QLineSeries * radiationDoseSeries
- QTimer * timer

  Podstawowe komponenty.
- SensorReader * sensorReader
- SensorDataLogger logger
- QLineEdit * co2Label

  Pola wyświetlające pomiary.
- QLineEdit * co2TempLabel
- QLineEdit * co2HumLabel
- QLineEdit * pm1Label
- QLineEdit * pm25Label
- QLineEdit * pm10Label
- QLineEdit * radiationLabel
- QLineEdit * radiationDoseLabel
- QLabel * co2StatusLabel

  Etykiety statusu.
- QLabel * pmStatusLabel
- QLabel * radiationStatusLabel

- QFrame * sensorDataFrame

  Kontenery interfejsu.
- QFrame * interpretationFrame
- QButtonGroup * timeMachineGroup

  Kontrolki wyboru zakresu czasu.
- QRadioButton * hour1Button
- QRadioButton * hour2Button
- QRadioButton * hour4Button
- QRadioButton * hour8Button
- QRadioButton * hour12Button
- QRadioButton * hour24Button
- QRadioButton * hour48Button
- QRadioButton * hour78Button
- int timeMachineHours

  Aktualny zakres czasu w godzinach.
- QDateTime chartStartTime

  Początek zakresu wykresu.
- QComboBox * languageSelector

  Komponenty obsługi języków.
- QTranslator translator

5.1.1 Opis szczegółowy

Główne okno aplikacji wyświetlające dane z czujników i wykresy.

Klasa odpowiada za prezentację danych w czasie rzeczywistym oraz umożliwia przeglądanie historii pomiarów.

5.1.2 Dokumentacja konstruktora i destruktora

5.1.2.1 MainWindow()

```
MainWindow::MainWindow (
            SensorReader * reader,
            QWidget * parent = nullptr )  [explicit]
```

Konstruktor głównego okna.

Parametry

reader	Wskaźnik do czytnika danych z czujników
parent	Wskaźnik do widgetu nadrzędnego
reader	Wskaźnik do obiektu czytającego dane z czujników
parent	Wskaźnik do widgetu nadrzędnego

5.1.2.2 ~MainWindow()

```
MainWindow::~MainWindow ( )
```

5.1.3 Dokumentacja funkcji składowych

5.1.3.1 changeLanguage()

```
void MainWindow::changeLanguage (
            const QString & language )  [private]
```

Zmienia język interfejsu.

Zmienia język interfejsu aplikacji.

Parametry

language	Kod języka ("pl" lub "en")

5.1.3.2 loadHistoricalData

```
void MainWindow::loadHistoricalData ( )  [private], [slot]
```

Wczytuje i wyświetla dane historyczne.

Wczytuje dane historyczne z pliku CSV.

Odczytuje zapisane dane z czujników i aktualizuje serie danych na wykresach

5.1.3.3 onTimeMachineChanged

```
void MainWindow::onTimeMachineChanged (
            int id )  [private], [slot]
```

Obsługuje zmianę zakresu czasu na wykresie.

Obsługuje zmianę zakresu czasu w widoku historycznym.

Parametry

id	Nowy zakres czasu w godzinach

5.1.3.4 updateChartTitles()

```
void MainWindow::updateChartTitles ( )  [private]
```

Aktualizuje tytuły wykresów.

Aktualizuje tytuły wykresów na podstawie wybranej serii danych.

5.1.3.5 updateInterfaceTexts()

```
void MainWindow::updateInterfaceTexts ( )  [private]
```

Aktualizuje teksty interfejsu po zmianie języka.

Aktualizuje wszystkie teksty w interfejsie po zmianie języka.

5.1.3.6 updateSensorData

```
void MainWindow::updateSensorData ( )  [private], [slot]
```

Aktualizuje interfejs danymi z czujników.

Aktualizuje wyświetlane dane z czujników.

Odczytuje nowe dane, waliduje je i aktualizuje interfejs użytkownika

5.1.4 Dokumentacja pól

5.1.4.1 axisX

```
QtCharts::QDateTimeAxis* MainWindow::axisX  [private]
```

5.1.4.2 axisY

```
QtCharts::QValueAxis* MainWindow::axisY  [private]
```

5.1.4.3 chart

```
QtCharts::QChart* MainWindow::chart  [private]
```

5.1.4.4 chartSelector

```
QComboBox* MainWindow::chartSelector  [private]
```

Komponenty wykresu.

5.1.4.5 chartStartTime

```
QDateTime MainWindow::chartStartTime  [private]
```

Początek zakresu wykresu.

5.1.4.6 chartView

```
QtCharts::QChartView* MainWindow::chartView  [private]
```

5.1.4.7 co2HumLabel

```
QLineEdit* MainWindow::co2HumLabel  [private]
```

5.1.4.8 co2Label

```
QLineEdit* MainWindow::co2Label  [private]
```

Pola wyświetlające pomiary.

5.1.4.9 co2Series

```
QtCharts::QLineSeries* MainWindow::co2Series  [private]
```

5.1.4.10 co2StatusLabel

```
QLabel* MainWindow::co2StatusLabel  [private]
```

Etykiety statusu.

5.1.4.11 co2TempLabel

```
QLineEdit* MainWindow::co2TempLabel  [private]
```

5.1.4.12 hour12Button

```
QRadioButton* MainWindow::hour12Button  [private]
```

5.1.4.13 hour1Button

```
QRadioButton* MainWindow::hour1Button  [private]
```

5.1.4.14 hour24Button

```
QRadioButton* MainWindow::hour24Button  [private]
```

5.1.4.15 hour2Button

```
QRadioButton* MainWindow::hour2Button  [private]
```

5.1.4.16 hour48Button

```
QRadioButton* MainWindow::hour48Button  [private]
```

5.1.4.17 hour4Button

```
QRadioButton* MainWindow::hour4Button  [private]
```

5.1.4.18 hour78Button

```
QRadioButton* MainWindow::hour78Button  [private]
```

5.1.4.19 hour8Button

```
QRadioButton* MainWindow::hour8Button  [private]
```

5.1.4.20 humiditySeries

```
QtCharts::QLineSeries* MainWindow::humiditySeries  [private]
```

5.1.4.21 interpretationFrame

```
QFrame* MainWindow::interpretationFrame  [private]
```

5.1.4.22 languageSelector

```
QComboBox* MainWindow::languageSelector  [private]
```

Komponenty obsługi języków.

5.1.4.23 logger

```
SensorDataLogger MainWindow::logger  [private]
```

5.1.4.24 pm10Label

```
QLineEdit* MainWindow::pm10Label  [private]
```

5.1.4.25 pm10Series

```
QtCharts::QLineSeries* MainWindow::pm10Series  [private]
```

5.1.4.26 pm1Label

```
QLineEdit* MainWindow::pm1Label  [private]
```

5.1.4.27 pm1Series

```
QtCharts::QLineSeries* MainWindow::pm1Series  [private]
```

5.1.4.28 pm25Label

```
QLineEdit* MainWindow::pm25Label  [private]
```

5.1.4.29 pm25Series

```
QtCharts::QLineSeries* MainWindow::pm25Series  [private]
```

5.1.4.30 pmStatusLabel

```
QLabel* MainWindow::pmStatusLabel  [private]
```

5.1.4.31 radiationDoseLabel

```
QLineEdit* MainWindow::radiationDoseLabel  [private]
```

5.1.4.32 radiationDoseSeries

```
QtCharts::QLineSeries* MainWindow::radiationDoseSeries  [private]
```

5.1.4.33 radiationLabel

```
QLineEdit* MainWindow::radiationLabel  [private]
```

5.1.4.34 radiationSeries

```
QtCharts::QLineSeries* MainWindow::radiationSeries  [private]
```

5.1.4.35 radiationStatusLabel

```
QLabel* MainWindow::radiationStatusLabel  [private]
```

5.1.4.36 sensorDataFrame

```
QFrame* MainWindow::sensorDataFrame  [private]
```

Kontenery interfejsu.

5.1.4.37 sensorReader

```
SensorReader* MainWindow::sensorReader  [private]
```

5.1.4.38 temperatureSeries

```
QtCharts::QLineSeries* MainWindow::temperatureSeries  [private]
```

5.1.4.39 timeMachineGroup

```
QButtonGroup* MainWindow::timeMachineGroup  [private]
```

Kontrolki wyboru zakresu czasu.

5.1.4.40 timeMachineHours

```
int MainWindow::timeMachineHours  [private]
```

Aktualny zakres czasu w godzinach.

5.1.4.41 timer

```
QTimer* MainWindow::timer  [private]
```

Podstawowe komponenty.

5.1.4.42 translator

```
QTranslator MainWindow::translator  [private]
```

Dokumentacja dla tej klasy została wygenerowana z plików:

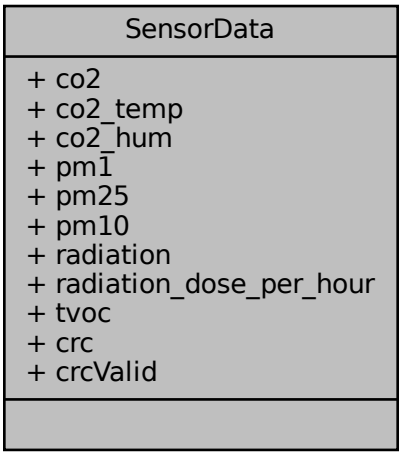
- MainWindow.h
- MainWindow.cpp

5.2 Dokumentacja struktury SensorData

Struktura przechowująca dane odczytane z czujników.

```
#include <SensorReader.h>
```

Diagram współpracy dla SensorData:



Pola danych

- int co2 = -1

  Poziom CO2 w ppm.
- int co2_temp = -1

  Temperatura CO2 w stopniach Celsjusza.
- int co2_hum = -1

  Wilgotność CO2 w procentach.
- int pm1 = -1

  Poziom pyłu PM1.0 w µg/m^3.
- int pm25 = -1

  Poziom pyłu PM2.5 w µg/m^3.
- int pm10 = -1

  Poziom pyłu PM10 w µg/m^3.
- int radiation = -1

  Liczba zliczeń promieniowania (CPM).
- float radiation_dose_per_hour = -1.0

  Dawka promieniowania na godzinę w µSv/h.
- int tvoc = -1

  Poziom TVOC.
- uint16_t crc = 0

  Pole CRC.
- bool crcValid = false

  Status walidacji CRC.

5.2.1 Opis szczegółowy

Struktura przechowująca dane odczytane z czujników.

5.2.2 Dokumentacja pól

5.2.2.1 co2

```
int SensorData::co2 = -1
```

Poziom CO2 w ppm.

5.2.2.2 co2_hum

```
int SensorData::co2_hum = -1
```

Wilgotność CO2 w procentach.

5.2.2.3 co2_temp

```
int SensorData::co2_temp = -1
```

Temperatura CO2 w stopniach Celsjusza.

5.2.2.4 crc

```
uint16_t SensorData::crc = 0
```

Pole CRC.

5.2.2.5 crcValid

```
bool SensorData::crcValid = false
```

Status walidacji CRC.

5.2.2.6 pm1

```
int SensorData::pm1 = -1
```

Poziom pyłu PM1.0 w µg/m^3.

5.2.2.7 pm10

```
int SensorData::pm10 = -1
```

Poziom pyłu PM10 w µg/m^3.

5.2.2.8 pm25

```
int SensorData::pm25 = -1
```

Poziom pyłu PM2.5 w µg/m^3.

5.2.2.9 radiation

```
int SensorData::radiation = -1
```

Liczba zliczeń promieniowania (CPM).

5.2.2.10 radiation_dose_per_hour

```
float SensorData::radiation_dose_per_hour = -1.0
```

Dawka promieniowania na godzinę w µSv/h.

5.2.2.11 tvoc

```
int SensorData::tvoc = -1
```

Poziom TVOC.

Dokumentacja dla tej struktury została wygenerowana z pliku:

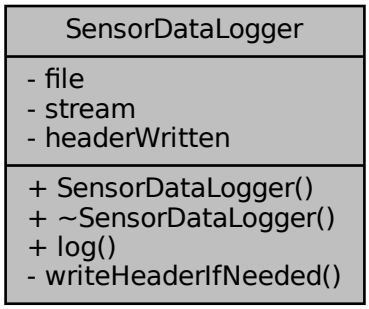
- SensorReader.h

5.3 Dokumentacja klasy SensorDataLogger

Klasa umożliwiająca logowanie danych z czujników do pliku CSV.

```
#include <SensorDataLogger.h>
```

Diagram współpracy dla SensorDataLogger:



Metody publiczne

- SensorDataLogger (const QString &filename)

  Konstruktor otwierający plik do zapisu.
- ~SensorDataLogger ()

  Destruktor zamykający plik.
- void log (const SensorData &data)

  Zapisuje pojedynczy rekord danych z czujników do pliku CSV.

Metody prywatne

- void writeHeaderIfNeeded ()

  Zapisuje nagłówek, jeśli plik jest pusty.

Atrybuty prywatne

- QFile file

  Obsługa pliku CSV.
- QTextStream stream

  Strumień do zapisu tekstu.
- bool headerWritten = false

  Flaga informująca o zapisaniu nagłówka.

5.3.1 Opis szczegółowy

Klasa umożliwiająca logowanie danych z czujników do pliku CSV.

5.3.2 Dokumentacja konstruktora i destruktora

5.3.2.1 SensorDataLogger()

```
SensorDataLogger::SensorDataLogger (
            const QString & filename )  [explicit]
```

Konstruktor otwierający plik do zapisu.

Konstruktor klasy logger'a.

Parametry

filename	Nazwa pliku CSV.
filename	Nazwa pliku CSV do zapisu danych

5.3.2.2 ~SensorDataLogger()

```
SensorDataLogger::~SensorDataLogger ( )
```

Destruktor zamykający plik.

5.3.3 Dokumentacja funkcji składowych

5.3.3.1 log()

```
void SensorDataLogger::log (
            const SensorData & data )
```

Zapisuje pojedynczy rekord danych z czujników do pliku CSV.

Zapisuje jeden rekord pomiarowy do pliku CSV.

Parametry

data	Struktura SensorData z danymi do zapisania.
data	Struktura zawierająca dane z czujników

5.3.3.2 writeHeaderIfNeeded()

```
void SensorDataLogger::writeHeaderIfNeeded ( )  [private]
```

Zapisuje nagłówek, jeśli plik jest pusty.

Zapisuje nagłówek CSV jeśli plik jest pusty.

5.3.4 Dokumentacja pól

5.3.4.1 file

```
QFile SensorDataLogger::file  [private]
```

Obsługa pliku CSV.

5.3.4.2 headerWritten

```
bool SensorDataLogger::headerWritten = false  [private]
```

Flaga informująca o zapisaniu nagłówka.

5.3.4.3 stream

```
QTextStream SensorDataLogger::stream  [private]
```

Strumień do zapisu tekstu.

Dokumentacja dla tej klasy została wygenerowana z plików:

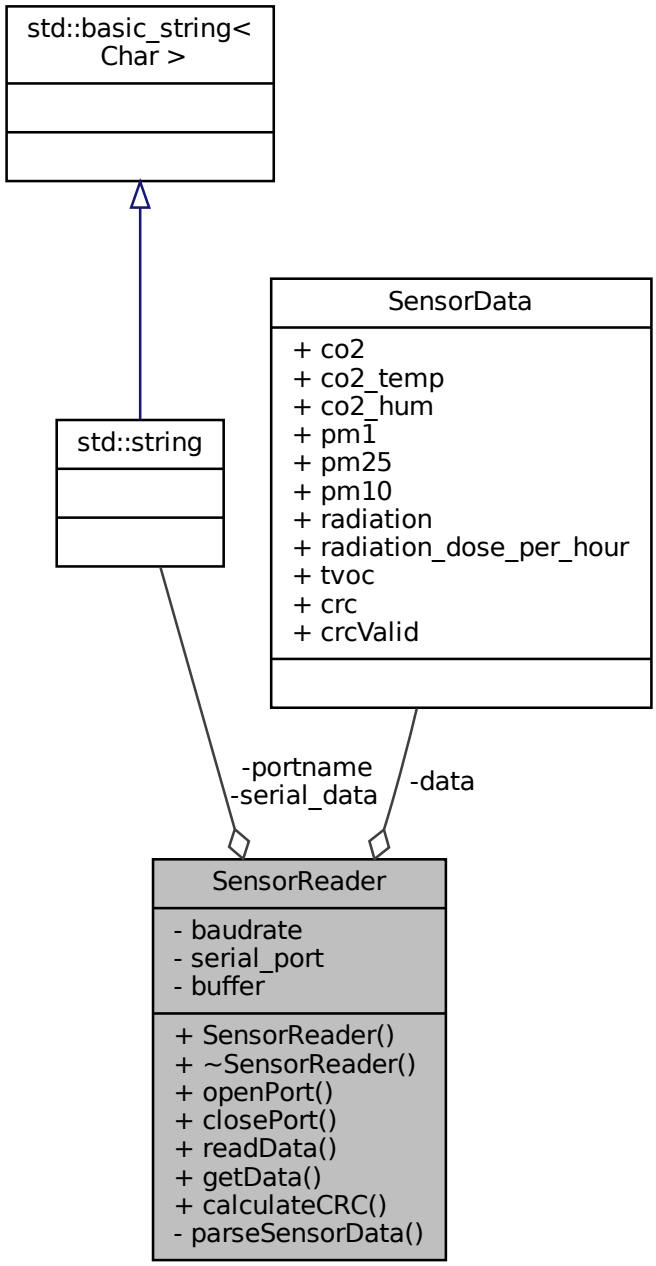
- SensorDataLogger.h
- SensorDataLogger.cpp

5.4 Dokumentacja klasy SensorReader

Klasa do obsługi czujników przez port szeregowy.

```
#include <SensorReader.h>
```

Diagram współpracy dla SensorReader:



Metody publiczne

- SensorReader (const std::string &portname, int baudrate=B9600)

  Konstruktor klasy SensorReader.
- ~SensorReader ()

  Destruktor klasy SensorReader.
- bool openPort ()

Otwiera port szeregowy.

- void closePort ()

  Zamyka port szeregowy.
- bool readData ()

  Odczytuje dane z portu szeregowego.
- SensorData getData () const

  Zwraca ostatnio odczytane dane z czujników.
- uint16_t calculateCRC (const std::string &data)

  Oblicza wartość CRC dla danych.

Metody prywatne

- SensorData parseSensorData (const std::string &raw)

  Parsuje surowe dane z czujników.

Atrybuty prywatne

- std::string portname

  Nazwa portu szeregowego.
- int baudrate

  Prędkość transmisji.
- int serial_port

  Uchwyt do portu szeregowego.
- char buffer [256]

  Bufor do przechowywania danych z portu szeregowego.
- std::string serial_data

  Dane odczytane z portu szeregowego.
- SensorData data

  Struktura przechowująca dane z czujników.

5.4.1 Opis szczegółowy

Klasa do obsługi czujników przez port szeregowy.

5.4.2 Dokumentacja konstruktora i destruktora

5.4.2.1 SensorReader()

```
SensorReader::SensorReader (
            const std::string & portname,
            int baudrate = B9600 )
```

Konstruktor klasy SensorReader.

Parametry

portname	Nazwa portu szeregowego.
baudrate	Prędkość transmisji (domyślnie B9600).

5.4.2.2 ~SensorReader()

```
SensorReader::~SensorReader ( )
```

Destruktor klasy SensorReader.

5.4.3 Dokumentacja funkcji składowych

5.4.3.1 calculateCRC()

```
uint16_t SensorReader::calculateCRC (
            const std::string & data )
```

Oblicza wartość CRC dla danych.

Parametry

data	Dane wejściowe w formie ciągu znaków.

Zwraca

Obliczona wartość CRC.

5.4.3.2 closePort()

```
void SensorReader::closePort ( )
```

Zamyka port szeregowy.

5.4.3.3 getData()

```
SensorData SensorReader::getData ( ) const
```

Zwraca ostatnio odczytane dane z czujników.

Zwraca

Struktura SensorData zawierająca dane z czujników.

5.4.3.4 openPort()

```
bool SensorReader::openPort ( )
```

Otwiera port szeregowy.

Zwraca

true, jeśli port został otwarty pomyślnie, false w przeciwnym razie.

5.4.3.5 parseSensorData()

```
SensorData SensorReader::parseSensorData (
            const std::string & raw )  [private]
```

Parsuje surowe dane z czujników.

Parsuje surowe dane z czujników i weryfikuje CRC.

Parametry

raw	Surowe dane w formie ciągu znaków.

Zwraca

Struktura SensorData zawierająca sparsowane dane.

5.4.3.6 readData()

```
bool SensorReader::readData ( )
```

Odczytuje dane z portu szeregowego.

Zwraca

true, jeśli dane zostały odczytane pomyślnie, false w przeciwnym razie.

5.4.4 Dokumentacja pól

5.4.4.1 baudrate

```
int SensorReader::baudrate  [private]
```

Prędkość transmisji.

5.4.4.2 buffer

```
char SensorReader::buffer[256]  [private]
```

Bufor do przechowywania danych z portu szeregowego.

5.4.4.3 data

```
SensorData SensorReader::data  [private]
```

Struktura przechowująca dane z czujników.

5.4.4.4 portname

```
std::string SensorReader::portname  [private]
```

Nazwa portu szeregowego.

5.4.4.5 serial_data

```
std::string SensorReader::serial_data  [private]
```

Dane odczytane z portu szeregowego.

5.4.4.6 serial_port

```
int SensorReader::serial_port  [private]
```

Uchwyt do portu szeregowego.

Dokumentacja dla tej klasy została wygenerowana z plików:

- SensorReader.h
- SensorReader.cpp

Chapter 6

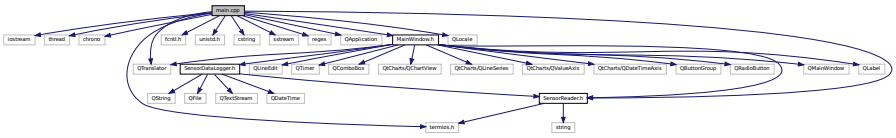
Dokumentacja plików

6.1 Dokumentacja pliku main.cpp

Program monitorujący dane z czujników środowiskowych.

```
#include <iostream>
#include <thread>
#include <chrono>
#include "SensorReader.h"
#include <fcntl.h>
#include <unistd.h>
#include <cstring>
#include <sstream>
#include <regex>
#include <termios.h>
#include <QApplication>
#include <QTranslator>
#include <QLocale>
#include "MainWindow.h"
```

Wykres zależności załączania dla main.cpp:



Definicje

- #define RESET "\033[0m"
- #define RED "\033[31m"
- #define GREEN "\033[32m"
- #define CLEAR "\033[2J\033[1;1H"

Funkcje

- void printInitStatus (bool portOpen, bool co2Ready, bool pmReady, bool radReady)

  Wyświetla status inicjalizacji systemu.
- int main (int argc, char ∗argv[ ])

  Główna funkcja programu.

6.1.1 Opis szczegółowy

Program monitorujący dane z czujników środowiskowych.

6.1.2 Dokumentacja definicji

6.1.2.1 CLEAR

```
#define CLEAR "\033[2J\033[1;1H"
```

6.1.2.2 GREEN

```
#define GREEN "\033[32m"
```

6.1.2.3 RED

```
#define RED "\033[31m"
```

6.1.2.4 RESET

```
#define RESET "\033[0m"
```

6.1.3 Dokumentacja funkcji

6.1.3.1 main()

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

Główna funkcja programu.

Inicjalizuje obiekt SensorReader, otwiera port szeregowy i uruchamia aplikację Qt.

Parametry

argc	Liczba argumentów wiersza poleceń.
argv	Tablica argumentów wiersza poleceń.

Zwraca

int Kod wyjścia programu.

6.1.3.2 printInitStatus()

```
void printInitStatus (
            bool portOpen,
            bool co2Ready,
            bool pmReady,
            bool radReady )
```

Wyświetla status inicjalizacji systemu.

Parametry

portOpen	Status portu szeregowego.
co2Ready	Status czujnika CO2.
pmReady	Status czujnika pyłu zawieszonego.
radReady	Status czujnika promieniowania.

6.2 Dokumentacja pliku MainWindow.cpp

Implementacja głównego okna aplikacji monitorującej czujniki środowiskowe.

```
#include "MainWindow.h"
#include "SensorDataLogger.h"
#include <QVBoxLayout>
#include <QGridLayout>
#include <QMainWindow>
#include <QPushButton>
#include <QLabel>
#include "SensorReader.h"
#include <QApplication>
#include <QTimer>
#include <QFrame>
#include <QFile>
#include <QTextStream>
#include <QDateTime>
#include <QDebug>
#include <QtCharts/QChart>
#include <QtCharts/QChartView>
```

```
#include <QtCharts/QLineSeries>
#include <QtCharts/QDateTimeAxis>
#include <QtCharts/QValueAxis>
#include <QComboBox>
#include <QButtonGroup>
#include <QRadioButton>
#include <QHBoxLayout>
#include <QLineEdit>
```

Wykres zależności załączania dla MainWindow.cpp:

Zmienne

- const int HIGH_CO2 = 1200

  Próg alarmowy CO2 (ppm)
- const int HIGH_PM1 = 25

  Próg alarmowy PM1.0 (µg/m^3)
- const int HIGH_PM25 = 25

  Próg alarmowy PM2.5 (µg/m^3)
- const int HIGH_PM10 = 50

  Próg alarmowy PM10 (µg/m^3)
- const int HIGH_RADIATION = 30

  Próg alarmowy promieniowania (imp/min)
- const float CPS_PER_USV = 0.0037

  Współczynnik konwersji CPS na µSv/h.
- const int MAX_CO2 = 2000

  Maksymalna wartość CO2 na wykresie.
- const int MAX_PM1 = 50

  Maksymalna wartość PM1.0 na wykresie.
- const int MAX_PM25 = 50

  Maksymalna wartość PM2.5 na wykresie.
- const int MAX_PM10 = 100

  Maksymalna wartość PM10 na wykresie.
- const int MAX_RADIATION = 100

  Maksymalna wartość promieniowania na wykresie.

6.2.1 Opis szczegółowy

Implementacja głównego okna aplikacji monitorującej czujniki środowiskowe.

6.2.2 Dokumentacja zmiennych

6.2.2.1 CPS_PER_USV

```
const float CPS_PER_USV = 0.0037
```

Współczynnik konwersji CPS na µSv/h.

6.2.2.2 HIGH_CO2

```
const int HIGH_CO2 = 1200
```

Próg alarmowy CO2 (ppm)

Stałe wartości progowe dla pomiarów

6.2.2.3 HIGH_PM1

```
const int HIGH_PM1 = 25
```

Próg alarmowy PM1.0 (µg/m^3)

6.2.2.4 HIGH_PM10

```
const int HIGH_PM10 = 50
```

Próg alarmowy PM10 (µg/m^3)

6.2.2.5 HIGH_PM25

```
const int HIGH_PM25 = 25
```

Próg alarmowy PM2.5 (µg/m^3)

6.2.2.6 HIGH_RADIATION

```
const int HIGH_RADIATION = 30
```

Próg alarmowy promieniowania (imp/min)

6.2.2.7 MAX_CO2

```
const int MAX_CO2 = 2000
```

Maksymalna wartość CO2 na wykresie.

Zakresy wykresów

6.2.2.8 MAX_PM1

```
const int MAX_PM1 = 50
```

Maksymalna wartość PM1.0 na wykresie.

6.2.2.9 MAX_PM10

```
const int MAX_PM10 = 100
```

Maksymalna wartość PM10 na wykresie.

6.2.2.10 MAX_PM25

```
const int MAX_PM25 = 50
```

Maksymalna wartość PM2.5 na wykresie.

6.2.2.11 MAX_RADIATION

```
const int MAX_RADIATION = 100
```

Maksymalna wartość promieniowania na wykresie.

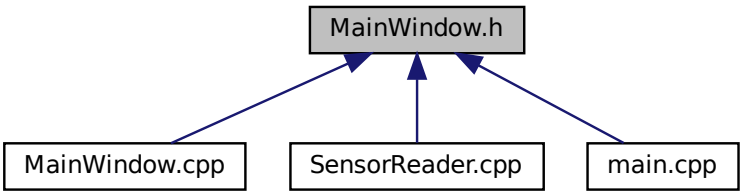
6.3 Dokumentacja pliku MainWindow.h

Plik nagłówkowy głównego okna aplikacji monitorującej czujniki.

```
#include <QMainWindow>
#include <QLabel>
#include <QLineEdit>
#include <QTimer>
#include <QComboBox>
#include <QtCharts/QChartView>
#include <QtCharts/QLineSeries>
#include <QtCharts/QValueAxis>
#include <QtCharts/QDateTimeAxis>
#include <QButtonGroup>
#include <QRadioButton>
#include <QTranslator>
#include "SensorReader.h"
#include "SensorDataLogger.h"
```

Wykres zależności załączania dla MainWindow.h:

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



Struktury danych

- class MainWindow

  Główne okno aplikacji wyświetlające dane z czujników i wykresy.

6.3.1 Opis szczegółowy

Plik nagłówkowy głównego okna aplikacji monitorującej czujniki.

6.4 MainWindow.h

Idź do dokumentacji tego pliku.

```
1 #ifndef MAINWINDOW_H
2 #define MAINWINDOW_H
3
4 #include <QMainWindow>
5 #include <QLabel>
6 #include <QLineEdit>
7 #include <QTimer>
8 #include <QComboBox>
9 #include <QtCharts/QChartView>
10 #include <QtCharts/QLineSeries>
11 #include <QtCharts/QValueAxis>
12 #include <QtCharts/QDateTimeAxis>
13 #include <QButtonGroup>
14 #include <QRadioButton>
15 #include <QTranslator>
16 #include "SensorReader.h"
17 #include "SensorDataLogger.h"
18
31 class MainWindow :  public QMainWindow
32 {
33     Q_OBJECT
34
35 public:
41     explicit MainWindow(SensorReader* reader, QWidget *parent = nullptr);
42     ~MainWindow();
43
44 private slots:
48     void updateSensorData();
49
53     void loadHistoricalData();
54
59     void onTimeMachineChanged(int id);
60
61 private:
63     QComboBox* chartSelector;
64     QtCharts::QChart* chart;
65     QtCharts::QChartView* chartView;
66     QtCharts::QDateTimeAxis* axisX;
67     QtCharts::QValueAxis* axisY;
68     QtCharts::QLineSeries* co2Series;
69     QtCharts::QLineSeries* pm1Series;
70     QtCharts::QLineSeries* pm25Series;
71     QtCharts::QLineSeries* pm10Series;
72     QtCharts::QLineSeries* radiationSeries;
73     QtCharts::QLineSeries* temperatureSeries;
74     QtCharts::QLineSeries* humiditySeries;
75     QtCharts::QLineSeries* radiationDoseSeries;
76
78     QTimer* timer;
79     SensorReader* sensorReader;
80     SensorDataLogger logger;
81
83     QLineEdit* co2Label;
84     QLineEdit* co2TempLabel;
85     QLineEdit* co2HumLabel;
86     QLineEdit* pm1Label;
87     QLineEdit* pm25Label;
88     QLineEdit* pm10Label;
89     QLineEdit* radiationLabel;
90     QLineEdit* radiationDoseLabel;
91
93     QLabel* co2StatusLabel;
94     QLabel* pmStatusLabel;
95     QLabel* radiationStatusLabel;
96
98     QFrame* sensorDataFrame;
99     QFrame* interpretationFrame;
100
102     QButtonGroup* timeMachineGroup;
103     QRadioButton* hour1Button;
104     QRadioButton* hour2Button;
105     QRadioButton* hour4Button;
106     QRadioButton* hour8Button;
107     QRadioButton* hour12Button;
108     QRadioButton* hour24Button;
109     QRadioButton* hour48Button;
110     QRadioButton* hour78Button;
111
112     int timeMachineHours;
113     QDateTime chartStartTime;
114
116     QComboBox* languageSelector;
```

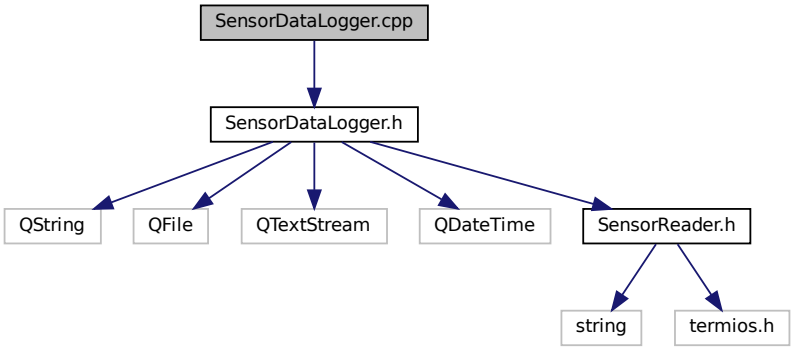
```
117     QTranslator translator;
118
122     void updateInterfaceTexts();
123
127     void updateChartTitles();
128
133     void changeLanguage(const QString &language);
134 };
135
136 #endif
```

6.5 Dokumentacja pliku README.md

6.6 Dokumentacja pliku SensorDataLogger.cpp

Implementacja klasy zapisującej dane z czujników do pliku CSV.

`#include "SensorDataLogger.h"`
Wykres zależności załączania dla SensorDataLogger.cpp:



Zmienne

- const float CPS_PER_USV = 0.0037f

  Współczynnik konwersji CPS na µSv/h.

6.6.1 Opis szczegółowy

Implementacja klasy zapisującej dane z czujników do pliku CSV.

6.6.2 Dokumentacja zmiennych

6.6.2.1 CPS_PER_USV

```
const float CPS_PER_USV = 0.0037f
```

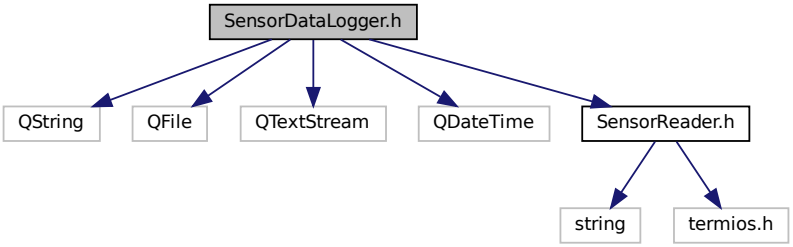
Współczynnik konwersji CPS na µSv/h.

6.7 Dokumentacja pliku SensorDataLogger.h

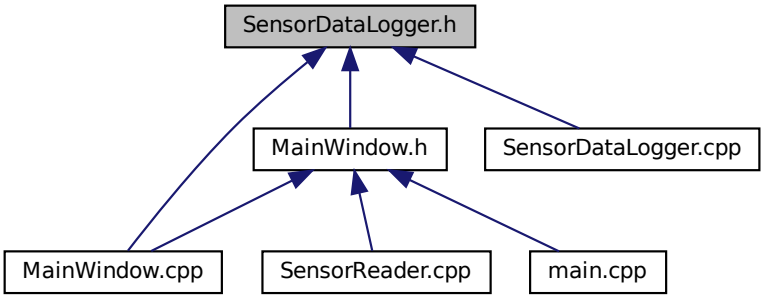
Deklaracja klasy SensorDataLogger do zapisu danych z czujników do pliku CSV.

```
#include <QString>
#include <QFile>
#include <QTextStream>
#include <QDateTime>
#include "SensorReader.h"
```

Wykres zależności załączania dla SensorDataLogger.h:



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



Struktury danych

- class SensorDataLogger

  Klasa umożliwiająca logowanie danych z czujników do pliku CSV.

6.7.1 Opis szczegółowy

Deklaracja klasy SensorDataLogger do zapisu danych z czujników do pliku CSV.

6.8 SensorDataLogger.h

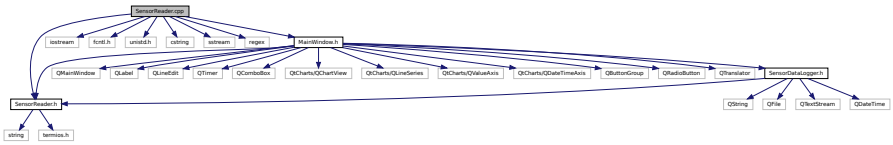
Idź do dokumentacji tego pliku.

```
1
6 #ifndef SENSORDATALOGGER_H
7 #define SENSORDATALOGGER_H
8
9 #include <QString>
10 #include <QFile>
11 #include <QTextStream>
12 #include <QDateTime>
13 #include "SensorReader.h"
14
19 class SensorDataLogger
20 {
21 public:
26     explicit SensorDataLogger(const QString& filename);
27
31     ~SensorDataLogger();
32
37     void log(const SensorData& data);
38
39 private:
40     QFile file;
41     QTextStream stream;
42     void writeHeaderIfNeeded();
43     bool headerWritten = false;
44 };
45
46 #endif
```

6.9 Dokumentacja pliku SensorReader.cpp

```
#include "SensorReader.h"
#include <iostream>
#include <fcntl.h>
#include <unistd.h>
#include <cstring>
#include <sstream>
#include <regex>
#include "MainWindow.h"
```

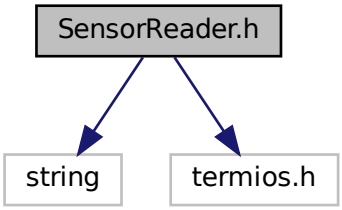
Wykres zależności załączania dla SensorReader.cpp:



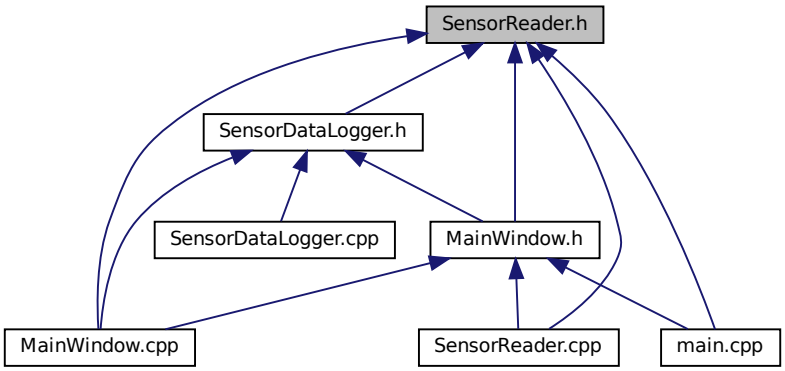
6.10 Dokumentacja pliku SensorReader.h

```
#include <string>
#include <termios.h>
```

Wykres zależności załączania dla SensorReader.h:



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



Struktury danych

- struct SensorData

  Struktura przechowująca dane odczytane z czujników.
- class SensorReader

  Klasa do obsługi czujników przez port szeregowy.

6.11 SensorReader.h

Idź do dokumentacji tego pliku.

```
1 #ifndef SENSORREADER_H
```

```
2 #define SENSORREADER_H
3
4 #include <string>
5 #include <termios.h>
6
11 struct SensorData {
12     int co2 = -1;
13     int co2_temp = -1;
14     int co2_hum = -1;
15     int pm1 = -1;
16     int pm25 = -1;
17     int pm10 = -1;
18     int radiation = -1;
19     float radiation_dose_per_hour = -1.0;
20     int tvoc = -1;
21     uint16_t crc = 0;
22     bool crcValid = false;
23 };
24
29 class SensorReader {
30 public:
36     SensorReader(const std::string& portname, int baudrate = B9600);
37
41     ~SensorReader();
42
47     bool openPort();
48
52     void closePort();
53
58     bool readData();
59
64     SensorData getData() const;
65
71     uint16_t calculateCRC(const std::string& data);
72
73 private:
74     std::string portname;
75     int baudrate;
76     int serial_port;
77     char buffer[256];
78     std::string serial_data;
79     SensorData data;
80
86     SensorData parseSensorData(const std::string& raw);
87 };
88
89 #endif
```