MWMPU

1.0

Generated by Doxygen 1.8.14

Contents

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 Ui Namespace Reference

Chapter 6

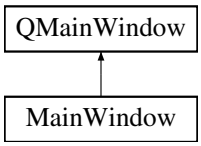
Class Documentation

6.1 MainWindow Class Reference

A class that supports the user interface and a module that generates charts.

`#include <mainwindow.h>`

Inheritance diagram for MainWindow:



Public Member Functions

- MainWindow (QWidget *parent=nullptr)
- ~MainWindow ()

Private Slots

- void onResetButtonClicked ()

  Enters Reset button event.
- void onCalibrateButtonClicked ()

  Enters Calibrate button event.
- void eventLoop ()

  Repeats events such as: reading from serial port, updating rotation and distance values and updating charts.
- void closeAppEvent ()

  Closes application.
- void onConnectButtonClicked ()

  Connects to the device using serial connection.
- void onAutoButtonClicked ()

  Automatically looks for proper device.
- void onDisconnectButtonClicked ()

  Disconnects from the device.
- void writeToStatusBar ()

  Updates status bar text.

Private Member Functions

- void setGraphsProperties ()

  Sets graphs properties like bg and fg colors and amount of plots.
- void generateBarGraph (QCustomPlot *pointer, const QVector< double > &data, const quint32 &plotsAmount, const QVector< QString > &labels, const QColor &bgColor, const QColor &fgColor)

  Generates bar graph.
- void updateBarGraph (QCustomPlot *pointer, const QVector< double > &data, const quint32 &plotsAmount, const QVector< QString > &labels, const QColor &bgColor, const QColor &fgColor)

  Updated bar graph.
- void generateLinearGraph (QCustomPlot *pointer, const quint32 &plotsAmount, const QVector< QString > &labels, const QVector< QString > &legend, const QVector< QColor > &plotColors, const QColor &bgColor, const QColor &fgColor)

  Generates linear graph.
- void updateLinearGraph (QCustomPlot *pointer, const QVector< QVector< double > *> pointers, const quint32 &plotsAmount)

  Updated linear graph.
- void clearHistoryPlots ()

  Clears measurements history data.
- void clearRealtimePlots ()

  Clears real time data.
- void setComboBox ()

  Sets combo box.
- QString getErrorText (const quint32 errorCode) const

  Returns specific error's text based on error's code.
- void showMessage ()

  Shows message on status bar based on arduino.
- void showMessage (const quint32 errorCode)

  Shows message on status bar based on error's code.

Private Attributes

- MeasurementHandler * transform
- SerialPortReader * arduino
- Ui::MainWindow * ui
- QTimer loopTimer
- QTimer statusBarTimer
- QVector< QString > labels1
- QVector< QString > labels2
- QVector< QString > labels3
- QVector< QString > labels4
- QVector< QString > legend
- QVector< QColor > plotColors
- QColor bgColor
- QColor fgColor
- quint32 plotsAmount
- double maxYAxisValue
- double maxYAxisValue2

6.1.1 Detailed Description

A class that supports the user interface and a module that generates charts.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 MainWindow()

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

6.1.2.2 ~MainWindow()

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

6.1.3 Member Function Documentation

6.1.3.1 clearHistoryPlots()

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

Clears measurements history data.

6.1.3.2 clearRealtimePlots()

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

Clears real time data.

6.1.3.3 closeAppEvent

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

Closes application.

6.1.3.4 eventLoop

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

Repeats events such as: reading from serial port, updating rotation and distance values and updating charts.

6.1.3.5 generateBarGraph()

```
void MainWindow::generateBarGraph (
            QCustomPlot * pointer,
            const QVector< double > & data,
            const quint32 & plotsAmount,
            const QVector< QString > & labels,
            const QColor & bgColor,
            const QColor & fgColor )  [private]
```

Generates bar graph.

Parameters

pointer	- pointer to target widget
data	- data to plot
plotsAmount	- plots amount
labels	- labels for axes
bgColor	- background color
fgColor	- foreground color

6.1.3.6 generateLinearGraph()

```
void MainWindow::generateLinearGraph (
            QCustomPlot * pointer,
            const quint32 & plotsAmount,
            const QVector< QString > & labels,
            const QVector< QString > & legend,
            const QVector< QColor > & plotColors,
            const QColor & bgColor,
            const QColor & fgColor )  [private]
```

Generates linear graph.

Parameters

pointer	- pointer to target widget
plotsAmount	- plots amount
labels	- labels for axes
legend	- legend's text
plotColors	- plot's colors
bgColor	- background color
fgColor	- foreground color

6.1.3.7 getErrorText()

```
QString MainWindow::getErrorText (
            const quint32 errorCode ) const  [private]
```

Returns specific error's text based on error's code.

Parameters

errorCode	- error's code

Returns

Error's text

6.1.3.8 onAutoButtonClicked

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

Automatically looks for proper device.

6.1.3.9 onCalibrateButtonClicked

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

Enters Calibrate button event.

6.1.3.10 onConnectButtonClicked

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

Connects to the device using serial connection.

6.1.3.11 onDisconnectButtonClicked

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

Disconnects from the device.

6.1.3.12 onResetButtonClicked

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

Enters Reset button event.

6.1.3.13 setComboBox()

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

Sets combo box.

6.1.3.14 setGraphsProperties()

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

Sets graphs properties like bg and fg colors and amount of plots.

6.1.3.15 showMessage() [1/2]

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

Shows message on status bar based on arduino.

6.1.3.16 showMessage() [2/2]

```
void MainWindow::showMessage (
            const quint32 errorCode )  [private]
```

Shows message on status bar based on error's code.

Parameters

errorCode	- error's code

6.1.3.17 updateBarGraph()

```
void MainWindow::updateBarGraph (
```

```
            QCustomPlot * pointer,
            const QVector< double > & data,
            const quint32 & plotsAmount,
            const QVector< QString > & labels,
            const QColor & bgColor,
            const QColor & fgColor )  [private]
```

Updated bar graph.

Parameters

pointer	- pointer to target widget
data	- data to plot
plotsAmount	- plots amount
labels	- labels for axes
bgColor	- background color
fgColor	- foreground color

6.1.3.18 updateLinearGraph()

```
void MainWindow::updateLinearGraph (
            QCustomPlot * pointer,
            const QVector< QVector< double > *> pointers,
            const quint32 & plotsAmount )  [private]
```

Updated linear graph.

Parameters

pointer	- pointer to target widget
pointers	- pointers to time and data containers
plotsAmount	- plots amount

6.1.3.19 writeToStatusBar

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

Updates status bar text.

6.1.4 Member Data Documentation

6.1.4.1 arduino

```
SerialPortReader* MainWindow::arduino  [private]
```

6.1.4.2 bgColor

```
QColor MainWindow::bgColor  [private]
```

6.1.4.3 fgColor

```
QColor MainWindow::fgColor  [private]
```

6.1.4.4 labels1

```
QVector<QString> MainWindow::labels1  [private]
```

acceleration chart labels

6.1.4.5 labels2

```
QVector<QString> MainWindow::labels2  [private]
```

angular velcity chart labels

6.1.4.6 labels3

```
QVector<QString> MainWindow::labels3  [private]
```

angle chart labels

6.1.4.7 labels4

```
QVector<QString> MainWindow::labels4  [private]
```

distance chart labels

6.1.4.8 legend

```
QVector<QString> MainWindow::legend  [private]
```

legend's text for linear graphs

6.1.4.9 loopTimer

```
QTimer MainWindow::loopTimer  [private]
```

timer that sets plots refresh frequency

6.1.4.10 maxYAxisValue

```
double MainWindow::maxYAxisValue  [private]
```

minimal bar graph value showed on vertical axis for angle graph

6.1.4.11 maxYAxisValue2

```
double MainWindow::maxYAxisValue2  [private]
```

minimal bar graph value showed on vertical axis for distance graph

6.1.4.12 plotColors

```
QVector<QColor> MainWindow::plotColors  [private]
```

plot's color for linear graphs

6.1.4.13 plotsAmount

```
quint32 MainWindow::plotsAmount  [private]
```

6.1.4.14 statusBarTimer

```
QTimer MainWindow::statusBarTimer  [private]
```

timer that sets how long will be status bar communicates displayed

6.1.4.15 transform

```
MeasurementHandler* MainWindow::transform  [private]
```

6.1.4.16 ui

```
Ui::MainWindow* MainWindow::ui  [private]
```

The documentation for this class was generated from the following files:

- h/mainwindow.h
- cpp/mainwindow.cpp

6.2 MeasurementHandler Class Reference

A class for storing data from sensors received from the microcontroller and handling necessary conversions.

```
#include <measurementhandler.h>
```

Public Member Functions

- MeasurementHandler ()
- ~MeasurementHandler ()
- QVector< double > * getTime () const
- QVector< double > * getXAcc () const
- QVector< double > * getYAcc () const
- QVector< double > * getZAcc () const
- QVector< double > * getXAngVel () const
- QVector< double > * getYAngVel () const
- QVector< double > * getZAngVel () const
- void pushTime (double value)
- void pushXAcc (double value)
- void pushYAcc (double value)
- void pushZAcc (double value)
- void pushXAngVel (double value)
- void pushYAngVel (double value)
- void pushZAngVel (double value)
- void calcAngleAndDistance ()

  Calculates angle [deg] and distance [cm].
- void clearHistoryData ()

  Deletes angular velocity and linear acceleration data.
- void clearRealtimeData ()

  Deletes rotation and distance data.
- QVector< double > getAccMeas () const
- QVector< double > & setAccMeas ()
- QVector< double > getGyroMeas () const
- QVector< double > & setGyroMeas ()
- QVector< double > getAngle () const
- QVector< double > & setAngle ()
- QVector< double > getDistance () const
- QVector< double > & setDistance ()
- QVector< QVector< double > * > getVectorOfAccPointers () const
- QVector< QVector< double > * > getVectorOfGyroPointers () const

Private Attributes

- QVector< double > * time
- QVector< double > * xAcc
- QVector< double > * yAcc
- QVector< double > * zAcc
- QVector< double > * xAngVel
- QVector< double > * yAngVel
- QVector< double > * zAngVel
- QVector< double > accMeas
- QVector< double > gyroMeas
- QVector< double > angle
- QVector< double > distance
- QVector< double > angleP
- QVector< double > distanceP
- QVector< QVector< double > * > vectorOfAccPointers
- QVector< QVector< double > * > vectorOfGyroPointers

6.2.1 Detailed Description

A class for storing data from sensors received from the microcontroller and handling necessary conversions.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 MeasurementHandler()

```
MeasurementHandler::MeasurementHandler ( )
```

6.2.2.2 ~MeasurementHandler()

```
MeasurementHandler::~MeasurementHandler ( )
```

6.2.3 Member Function Documentation

6.2.3.1 calcAngleAndDistance()

```
void MeasurementHandler::calcAngleAndDistance ( )
```

Calculates angle [deg] and distance [cm].

6.2.3.2 clearHistoryData()

```
void MeasurementHandler::clearHistoryData ( )
```

Deletes angular velocity and linear acceleration data.

6.2.3.3 clearRealtimeData()

```
void MeasurementHandler::clearRealtimeData ( )
```

Deletes rotation and distance data.

6.2.3.4 getAccMeas()

```
QVector< double > MeasurementHandler::getAccMeas ( ) const
```

6.2.3.5 getAngle()

```
QVector< double > MeasurementHandler::getAngle ( ) const
```

6.2.3.6 getDistance()

```
QVector< double > MeasurementHandler::getDistance ( ) const
```

6.2.3.7 getGyroMeas()

```
QVector< double > MeasurementHandler::getGyroMeas ( ) const
```

6.2.3.8 getTime()

```
QVector< double > * MeasurementHandler::getTime ( ) const
```

6.2.3.9 getVectorOfAccPointers()

```
QVector< QVector< double > * > MeasurementHandler::getVectorOfAccPointers ( ) const
```

6.2.3.10 getVectorOfGyroPointers()

```
QVector< QVector< double > * > MeasurementHandler::getVectorOfGyroPointers ( ) const
```

6.2.3.11 getXAcc()

```
QVector< double > * MeasurementHandler::getXAcc ( ) const
```

6.2.3.12 getXAngVel()

```
QVector< double > * MeasurementHandler::getXAngVel ( ) const
```

6.2.3.13 getYAcc()

```
QVector< double > * MeasurementHandler::getYAcc ( ) const
```

6.2.3.14 getYAngVel()

```
QVector< double > * MeasurementHandler::getYAngVel ( ) const
```

6.2.3.15 getZAcc()

```
QVector< double > * MeasurementHandler::getZAcc ( ) const
```

6.2.3.16 getZAngVel()

```
QVector< double > * MeasurementHandler::getZAngVel ( ) const
```

6.2.3.17 pushTime()

```
void MeasurementHandler::pushTime (
            double value )
```

6.2.3.18 pushXAcc()

```
void MeasurementHandler::pushXAcc (
            double value )
```

6.2.3.19 pushXAngVel()

```
void MeasurementHandler::pushXAngVel (
            double value )
```

6.2.3.20 pushYAcc()

```
void MeasurementHandler::pushYAcc (
            double value )
```

6.2.3.21 pushYAngVel()

```
void MeasurementHandler::pushYAngVel (
            double value )
```

6.2.3.22 pushZAcc()

```
void MeasurementHandler::pushZAcc (
            double value )
```

6.2.3.23 pushZAngVel()

```
void MeasurementHandler::pushZAngVel (
            double value )
```

6.2.3.24 setAccMeas()

```
QVector< double > & MeasurementHandler::setAccMeas ( )
```

6.2.3.25 setAngle()

```
QVector< double > & MeasurementHandler::setAngle ( )
```

6.2.3.26 setDistance()

```
QVector< double > & MeasurementHandler::setDistance ( )
```

6.2.3.27 setGyroMeas()

```
QVector< double > & MeasurementHandler::setGyroMeas ( )
```

6.2.4 Member Data Documentation

6.2.4.1 accMeas

```
QVector<double> MeasurementHandler::accMeas  [private]
```

6.2.4.2 angle

```
QVector<double> MeasurementHandler::angle  [private]
```

6.2.4.3 angleP

```
QVector<double> MeasurementHandler::angleP  [private]
```

6.2.4.4 distance

```
QVector<double> MeasurementHandler::distance  [private]
```

6.2.4.5 distanceP

```
QVector<double> MeasurementHandler::distanceP  [private]
```

6.2.4.6 gyroMeas

```
QVector<double> MeasurementHandler::gyroMeas  [private]
```

6.2.4.7 time

```
QVector<double>* MeasurementHandler::time  [private]
```

6.2.4.8 vectorOfAccPointers

```
QVector<QVector<double> *> MeasurementHandler::vectorOfAccPointers  [private]
```

6.2.4.9 vectorOfGyroPointers

```
QVector<QVector<double> *> MeasurementHandler::vectorOfGyroPointers  [private]
```

6.2.4.10 xAcc

```
QVector<double>* MeasurementHandler::xAcc  [private]
```

6.2.4.11 xAngVel

```
QVector<double>* MeasurementHandler::xAngVel  [private]
```

6.2.4.12 yAcc

```
QVector<double> * MeasurementHandler::yAcc  [private]
```

6.2.4.13 yAngVel

```
QVector<double> * MeasurementHandler::yAngVel  [private]
```

6.2.4.14 zAcc

```
QVector<double> * MeasurementHandler::zAcc  [private]
```

6.2.4.15 zAngVel

```
QVector<double> * MeasurementHandler::zAngVel  [private]
```

The documentation for this class was generated from the following files:

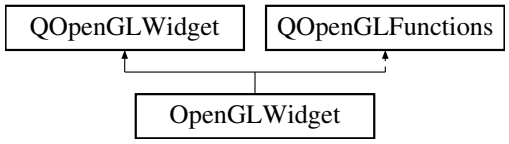
- h/measurementhandler.h
- cpp/measurementhandler.cpp

6.3 OpenGLWidget Class Reference

A class that creates the OpenGL window for rendering the Arduino model in real time.

```
#include <openglwidget.h>
```

Inheritance diagram for OpenGLWidget:



Public Member Functions

- OpenGLWidget (QWidget *parent=nullptr)
- ~OpenGLWidget ()
- void updateRotation (const QVector< double > nRotation)

  A method used to update the angular position of a rendered object.
- void updateDistance (const QVector< double > nDistance)

  A method for updating the distance of a rendered object from the center of the coordinate system.

Protected Member Functions

- void initializeGL () override

  Prepares shaders, model and its textures for rendering.
- void resizeGL (int width, int height) override

  Resizes widget.
- void paintGL () override

  Draws scene.

Private Member Functions

- void createObject ()

  Sends the coordinates of the vertices and UV's of the object to the buffer.

Private Attributes

- QColor bgColor
- QOpenGLShaderProgram * program
- QOpenGLBuffer vbo
- QVector< double > rotation
- QVector< double > distance
- WireframeModel * model
- QOpenGLTexture * texture

6.3.1 Detailed Description

A class that creates the OpenGL window for rendering the Arduino model in real time.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 OpenGLWidget()

```
OpenGLWidget::OpenGLWidget (
            QWidget * parent = nullptr )  [explicit]
```

6.3.2.2 ~OpenGLWidget()

```
OpenGLWidget::~OpenGLWidget ( )
```

6.3.3 Member Function Documentation

6.3.3.1 createObject()

```
void OpenGLWidget::createObject ( )  [private]
```

Sends the coordinates of the vertices and UV's of the object to the buffer.

6.3.3.2 initializeGL()

```
void OpenGLWidget::initializeGL ( )  [override], [protected]
```

Prepares shaders, model and its textures for rendering.

6.3.3.3 paintGL()

```
void OpenGLWidget::paintGL ( )  [override], [protected]
```

Draws scene.

6.3.3.4 resizeGL()

```
void OpenGLWidget::resizeGL (
            int width,
            int height )  [override], [protected]
```

Resizes widget.

Parameters

width	
height	

6.3.3.5 updateDistance()

```
void OpenGLWidget::updateDistance (
            const QVector< double > nDistance )
```

A method for updating the distance of a rendered object from the center of the coordinate system.

Parameters

nDistance	- xyz distance values

6.3.3.6 updateRotation()

```
void OpenGLWidget::updateRotation (
            const QVector< double > nRotation )
```

A method used to update the angular position of a rendered object.

Parameters

nRotation	- xyz angular position values

6.3.4 Member Data Documentation

6.3.4.1 bgColor

```
QColor OpenGLWidget::bgColor  [private]
```

6.3.4.2 distance

```
QVector<double> OpenGLWidget::distance  [private]
```

6.3.4.3 model

```
WireframeModel* OpenGLWidget::model  [private]
```

6.3.4.4 program

```
QOpenGLShaderProgram* OpenGLWidget::program  [private]
```

6.3.4.5 rotation

```
QVector<double> OpenGLWidget::rotation  [private]
```

6.3.4.6 texture

```
QOpenGLTexture* OpenGLWidget::texture  [private]
```

6.3.4.7 vbo

```
QOpenGLBuffer OpenGLWidget::vbo  [private]
```

The documentation for this class was generated from the following files:

- h/openglwidget.h
- cpp/openglwidget.cpp

6.4 SerialPortReader Class Reference

A class wraps everything about receiving data from the microcontroller.

```
#include <serialportreader.h>
```

Public Member Functions

- SerialPortReader (MeasurementHandler *nHandler)
  Parametric constructor.
- ~SerialPortReader ()
- void readLine ()
  A method to read one line of data from the device.
- QVector< QSerialPortInfo > getInfo () const
  A method used to obtain information about devices connected to a computer.
- bool connect (quint32 index)
  A method to connect to the device.
- bool disconnect ()
  A method to disconnect from the device.
- quint32 getErrorCode () const
  A method to obtain the error code.
- void resetErrorCode ()
  A method to remove error signatures.

Private Member Functions

- bool goodChoice ()

  Checks if received data from connected device is valid.

Private Attributes

- MeasurementHandler * mHandler
- QSerialPort * m_serialPort
- QVector< QSerialPortInfo > m_info
- bool isOpen = false
- quint32 errorCode = 0
- QVector< double > time
- QVector< double > accelerationX
- QVector< double > accelerationY
- QVector< double > accelerationZ
- QVector< double > angularVelocityX
- QVector< double > angularVelocityY
- QVector< double > angularVelocityZ

6.4.1 Detailed Description

A class wraps everything about receiving data from the microcontroller.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 SerialPortReader()

```
SerialPortReader::SerialPortReader (
            MeasurementHandler * nHandler )
```

Parametric constructor.

Parameters

nHandler	- a pointer to the object where the data of the loaded object is to be saved

6.4.2.2 ~SerialPortReader()

```
SerialPortReader::~SerialPortReader ( )
```

6.4.3 Member Function Documentation

6.4.3.1 connect()

```
bool SerialPortReader::connect (
            quint32 index )
```

A method to connect to the device.

Parameters

index	- device's index

Returns

true if device is succesfully connected

6.4.3.2 disconnect()

```
bool SerialPortReader::disconnect ( )
```

A method to disconnect from the device.

Returns

true if device is succesfully disconnected

6.4.3.3 getErrorCode()

```
quint32 SerialPortReader::getErrorCode ( ) const
```

A method to obtain the error code.

Returns

error's code index

6.4.3.4 getInfo()

```
QVector< QSerialPortInfo > SerialPortReader::getInfo ( ) const
```

A method used to obtain information about devices connected to a computer.

Returns

the device names

6.4.3.5 goodChoice()

```
bool SerialPortReader::goodChoice ( )  [private]
```

Checks if received data from connected device is valid.

Returns

true if received data is valid

6.4.3.6 readLine()

```
void SerialPortReader::readLine ( )
```

A method to read one line of data from the device.

6.4.3.7 resetErrorCode()

```
void SerialPortReader::resetErrorCode ( )
```

A method to remove error signatures.

6.4.4 Member Data Documentation

6.4.4.1 accelerationX

```
QVector<double> SerialPortReader::accelerationX  [private]
```

6.4.4.2 accelerationY

```
QVector<double> SerialPortReader::accelerationY  [private]
```

6.4.4.3 accelerationZ

```
QVector<double> SerialPortReader::accelerationZ  [private]
```

6.4.4.4 angularVelocityX

```
QVector<double> SerialPortReader::angularVelocityX  [private]
```

6.4.4.5 angularVelocityY

```
QVector<double> SerialPortReader::angularVelocityY  [private]
```

6.4.4.6 angularVelocityZ

```
QVector<double> SerialPortReader::angularVelocityZ  [private]
```

6.4.4.7 errorCode

```
quint32 SerialPortReader::errorCode = 0  [private]
```

6.4.4.8 isOpen

```
bool SerialPortReader::isOpen = false  [private]
```

6.4.4.9 m_info

```
QVector<QSerialPortInfo> SerialPortReader::m_info  [private]
```

6.4.4.10 m_serialPort

```
QSerialPort* SerialPortReader::m_serialPort  [private]
```

6.4.4.11 mHandler

```
MeasurementHandler* SerialPortReader::mHandler  [private]
```

6.4.4.12 time

```
QVector<double> SerialPortReader::time  [private]
```

The documentation for this class was generated from the following files:

- h/serialportreader.h
- cpp/serialportreader.cpp

6.5 WireframeModel Class Reference

A class that loads data from a file and stores it for rendering using OpenGL.

```
#include <wireframemodel.h>
```

Public Member Functions

- WireframeModel ()
- ~WireframeModel ()
- bool LoadObject (const QString filename)

  A method for parsing and saving a 3D model.
- QVector< GLfloat > * getData () const

  Getter for object's data.
- quint32 getSize () const

  Getter for amount of stored vertices and UV's combined.

Private Attributes

- QVector< GLfloat > * data

6.5.1 Detailed Description

A class that loads data from a file and stores it for rendering using OpenGL.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 WireframeModel()

```
WireframeModel::WireframeModel ( )
```

6.5.2.2 ~WireframeModel()

```
WireframeModel::~WireframeModel ( )
```

6.5.3 Member Function Documentation

6.5.3.1 getData()

```
QVector< GLfloat > * WireframeModel::getData ( ) const
```

Getter for object's data.

Returns

Pointer to object's data

6.5.3.2 getSize()

```
quint32 WireframeModel::getSize ( ) const
```

Getter for amount of stored vertices and UV's combined.

Returns

Size of data container

6.5.3.3 LoadObject()

```
bool WireframeModel::LoadObject (
            const QString filename )
```

A method for parsing and saving a 3D model.

Parameters

filename	- path for file .obj, extension not included

Returns

true if model is loaded succesfully

6.5.4 Member Data Documentation

6.5.4.1 data

```
QVector<GLfloat>* WireframeModel::data  [private]
```

The documentation for this class was generated from the following files:

- h/wireframemodel.h
- cpp/wireframemodel.cpp

Chapter 7

File Documentation

7.1 cpp/main.cpp File Reference

```
#include "mainwindow.h"
#include <QApplication>
#include <QSurfaceFormat>
```

Functions

- int main (int argc, char ∗argv[ ])

7.1.1 Function Documentation

7.1.1.1 main()

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

7.2 cpp/mainwindow.cpp File Reference

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
```

7.3 cpp/measurementhandler.cpp File Reference

```
#include "measurementhandler.h"
```

7.4 cpp/openglwidget.cpp File Reference

```
#include "openglwidget.h"
```

Macros

- #define VERTEX_COORD_ATTRIBUTE 0
- #define TEXTURE_COORD_ATTRIBUTE 1

7.4.1 Macro Definition Documentation

7.4.1.1 TEXTURE_COORD_ATTRIBUTE

```
#define TEXTURE_COORD_ATTRIBUTE 1
```

7.4.1.2 VERTEX_COORD_ATTRIBUTE

```
#define VERTEX_COORD_ATTRIBUTE 0
```

7.5 cpp/serialportreader.cpp File Reference

```
#include "serialportreader.h"
#include <QCoreApplication>
```

7.6 cpp/wireframemodel.cpp File Reference

```
#include "wireframemodel.h"
```

7.7 h/mainwindow.h File Reference

```
#include <QMainWindow>
#include <QDebug>
#include <QTimer>
#include <QColor>
#include <QtMath>
#include "qcustomplot.h"
#include "serialportreader.h"
#include "measurementhandler.h"
```

Classes

- class MainWindow

  A class that supports the user interface and a module that generates charts.

Namespaces

- Ui

7.8 h/measurementhandler.h File Reference

```
#include <QVector>
#include <QDebug>
#include <QtMath>
```

Classes

- class MeasurementHandler

  A class for storing data from sensors received from the microcontroller and handling necessary conversions.

7.9 h/openglwidget.h File Reference

```
#include <QWidget>
#include <QOpenGLFunctions>
#include <QOpenGLWidget>
#include <QOpenGLBuffer>
#include <QOpenGLShaderProgram>
#include <QMatrix4x4>
#include <QQuaternion>
#include <QVector>
#include <QOpenGLTexture>
#include "wireframemodel.h"
```

Classes

- class OpenGLWidget

  A class that creates the OpenGL window for rendering the Arduino model in real time.

7.10 h/serialportreader.h File Reference

```
#include <QByteArray>
#include <QSerialPort>
#include <QSerialPortInfo>
#include <QDebug>
#include <QVector3D>
#include <QString>
#include <QStringList>
#include "measurementhandler.h"
#include <QApplication>
#include <QThread>
```

Classes

- class SerialPortReader

  A class wraps everything about receiving data from the microcontroller.

7.11 h/wireframemodel.h File Reference

```
#include <QString>
#include <QVector>
#include <QVector3D>
#include <QVector2D>
#include <QFile>
#include <QDebug>
#include <QOpenGLFunctions>
```

Classes

- class WireframeModel

  A class that loads data from a file and stores it for rendering using OpenGL.

Index