Myo bracelet Connector

Generated by Doxygen 1.8.14

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 Communicator Class Reference

```
#include <Communicator.h>
```

Public Member Functions

- **Communicator** (uint8_t portNumber, DWORD CBR_baudRate)
- **Communicator** (uint8_t portNumber, DWORD CBR_baudRate, byte byteSize)
- **Communicator** (uint8_t portNumbe, DWORD CBR_baudRate, byte byteSize, byte parity)
- bool **Write** (const char ∗message, int messageLength)
- ∼**Communicator** ()

4.1.1 Constructor & Destructor Documentation

4.1.1.1 Communicator() [1/3]

```
Communicator::Communicator (
            uint8_t portNumber,
            DWORD CBR_baudRate )
```

4.1.1.2 Communicator() [2/3]

```
Communicator::Communicator (
            uint8_t portNumber,
            DWORD CBR_baudRate,
            byte byteSize )
```

4.1.1.3 Communicator() [3/3]

```
Communicator::Communicator (
            uint8_t portNumbe,
            DWORD CBR_baudRate,
            byte byteSize,
            byte parity )
```

4.1.1.4 ∼Communicator()

```
Communicator::∼Communicator ( )
```

4.1.2 Member Function Documentation

4.1.2.1 Write()

```
bool Communicator::Write (
            const char * message,
            int messageLength )
```

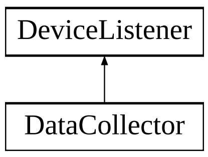
The documentation for this class was generated from the following files:

- **Communicator.h**
- **Communicator.cpp**

4.2 DataCollector Class Reference

```
#include <DataCollector.h>
```

Inheritance diagram for DataCollector:



Public Member Functions

- **DataCollector** ()
- void **onOrientationData** (myo::Myo ∗myo, uint64_t timestamp, const myo::Quaternion< float > &rotation)
- void **onAccelerometerData** (myo::Myo ∗myo, uint64_t timestamp, const myo::Vector3< float > &accel)
- void **onGyroscopeData** (myo::Myo ∗myo, uint64_t timestamp, const myo::Vector3< float > &gyro)
- void **onEmgData** (myo::Myo ∗myo, uint64_t timestamp, const int8_t ∗emg)
- void **onBatteryLevelReceived** (myo::Myo ∗myo, uint64_t timestamp, uint8_t level)
- void **onRssi** (myo::Myo ∗myo, uint64_t timestamp, int8_t rssi)
- void **onConnect** (myo::Myo ∗myo, uint64_t timestamp, myo::FirmwareVersion firmwareVersion)
- void **onDisconnect** (myo::Myo ∗myo, uint64_t timestamp)
- myo::Quaternion< float > **getRotation** ()

    Gets rotation values. (updates automatically if class is conneced to myo::Hub)
- myo::Vector3< float > **getGyroscope** ()

    Gets the Gyroscope values. (updates automatically if class is conneced to myo::Hub)
- myo::Vector3< float > **getAccelerometer** ()

    Gets all 8 EMG sensor Values. (updates automatically if class is conneced to myo::Hub)
- void **getEMG** (std::array< int8_t, 8 > ∗data)

    Gets all 8 EMG sensor Values. (updates automatically if class is conneced to myo::Hub)
- float **getRotation_roll** ()

    Calculated Roll from rotation data.
- float **getRotation_pitch** ()

    Calculated pitch from rotation data.
- float **getRotation_yaw** ()

    Calculated yaw from rotation data.
- uint8_t **getBatteryLevel** ()

    Battery: This value wont update periodically. update via MYO::requestBatteryLevel()
- int8_t **getBluetoothRange** ()

    Bluetooth: This value wont update periodically. update via update via MYO::requestRssi();.
- bool **getConnectionStatus** ()

    This value is automaticly updated.

4.2.1 Constructor & Destructor Documentation

4.2.1.1 DataCollector()

```
DataCollector::DataCollector ( )
```

4.2.2 Member Function Documentation

4.2.2.1 getAccelerometer()

```
myo::Vector3<float> DataCollector::getAccelerometer ( )  [inline]
```

Gets all 8 EMG sensor Values. (updates automatically if class is conneced to myo::Hub)

Returns

accel myo::Vector3<float> use raw value : .x() .y() .z().

4.2.2.2 getBatteryLevel()

```
uint8_t DataCollector::getBatteryLevel ( )  [inline]
```

Battery: This value wont update periodically. update via MYO::requestBatteryLevel()

Returns

Battery Level in procentage.

4.2.2.3 getBluetoothRange()

```
int8_t DataCollector::getBluetoothRange ( )  [inline]
```

Bluetooth: This value wont update periodically. update via update via MYO::requestRssi();.

Returns

bluetooth range 0-127.

4.2.2.4 getConnectionStatus()

```
bool DataCollector::getConnectionStatus ( )  [inline]
```

This value is automaticly updated.

Returns

connection status (bool) true = Connected | false = disconnected.

4.2.2.5 getEMG()

```
void DataCollector::getEMG (
            std::array< int8_t, 8 > * data )  [inline]
```

Gets all 8 EMG sensor Values. (updates automatically if class is conneced to myo::Hub)

Parameters

`in,out`	*data*	returns emg data if ∗data != null.

4.2.2.6 getGyroscope()

```
myo::Vector3<float> DataCollector::getGyroscope ( )  [inline]
```

Gets the Gyroscope values. (updates automatically if class is conneced to myo::Hub)

Returns

accel myo::Vector3<float> use raw value : .x() .y() .z().

4.2.2.7 getRotation()

```
myo::Quaternion<float> DataCollector::getRotation ( )  [inline]
```

Gets rotation values. (updates automatically if class is conneced to myo::Hub)

Returns

myo::Quaternion<float> use raw value: .x() .y() .z() .w().

4.2.2.8 getRotation_pitch()

```
float DataCollector::getRotation_pitch ( )  [inline]
```

Calculated pitch from rotation data.

Returns

Pitch in radial.

4.2.2.9 getRotation_roll()

```
float DataCollector::getRotation_roll ( )  [inline]
```

Calculated Roll from rotation data.

Returns

Roll in radial.

4.2.2.10 getRotation_yaw()

```
float DataCollector::getRotation_yaw ( )  [inline]
```

Calculated yaw from rotation data.

Returns

Yaw in radial.

4.2.2.11 onAccelerometerData()

```
void DataCollector::onAccelerometerData (
            myo::Myo * myo,
            uint64_t timestamp,
            const myo::Vector3< float > & accel )
```

4.2.2.12 onBatteryLevelReceived()

```
void DataCollector::onBatteryLevelReceived (
            myo::Myo * myo,
            uint64_t timestamp,
            uint8_t level )
```

4.2.2.13 onConnect()

```
void DataCollector::onConnect (
            myo::Myo * myo,
            uint64_t timestamp,
            myo::FirmwareVersion firmwareVersion )
```

4.2.2.14 onDisconnect()

```
void DataCollector::onDisconnect (
            myo::Myo * myo,
            uint64_t timestamp )
```

4.2.2.15 onEmgData()

```
void DataCollector::onEmgData (
            myo::Myo * myo,
            uint64_t timestamp,
            const int8_t * emg )
```

4.2.2.16 onGyroscopeData()

```
void DataCollector::onGyroscopeData (
            myo::Myo * myo,
            uint64_t timestamp,
            const myo::Vector3< float > & gyro )
```

4.2.2.17 onOrientationData()

```
void DataCollector::onOrientationData (
            myo::Myo * myo,
            uint64_t timestamp,
            const myo::Quaternion< float > & rotation )
```

4.2.2.18 onRssi()

```
void DataCollector::onRssi (
            myo::Myo * myo,
            uint64_t timestamp,
            int8_t rssi )
```

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

- **DataCollector.h**
- **DataCollector.cpp**

4.3 PeakDetector< T > Class Template Reference

```
#include <PeakDetector.h>
```

Public Member Functions

- **PeakDetector** (int measureLength, T minimumSampleDifference, T minimumPeakThreshold, T mimimum↩ PeakOffset)

    Peakdetector detects peaks in realtime signals by calculating the direction of the signal.
- ∼**PeakDetector** ()
- void **Calculate** (T Sample)

    *Calculates the signal if a direction is detected. use **GetPeak()** (p. 15) to see if a peak is detected.*
- **PeakType GetPeak** ()

    *Call **Calculate(T sample)** (p. 14) before Getting the peakvalue.*
- T **GetRawPeekValue** ()

    *Call **Calculate(T sample)** (p. 14) before Getting the RawPeekValue.*

4.3.1 Constructor & Destructor Documentation

4.3.1.1 PeakDetector()

```
template<class T >
PeakDetector< T >:: PeakDetector (
             int measureLength,
             T minimumSampleDifference,
             T minimumPeakThreshold,
             T mimimumPeakOffset )
```

Peakdetector detects peaks in realtime signals by calculating the direction of the signal.

Parameters

measureLength	Length of the sampling array, has to be an even number.
minimumSampleDifference	If The sample is lower then previous sampleDifference the sample is discarded.
minimumPeakThreshold	Minimum Threshold relative to Peakoffset. If Sample is lower than Peakthreshold no peak will be detected. this value is the same for positive and negative samples.
mimimumPeakOffset	Sets the baseline of the peakthreshold.

4.3.1.2 ∼PeakDetector()

```
template<class T >
PeakDetector< T >::∼ PeakDetector ( )
```

4.3.2 Member Function Documentation

4.3.2.1 Calculate()

```
template<class T >
void  PeakDetector< T >::Calculate (
            T sample )
```

Calculates the signal if a direction is detected. use **GetPeak()** (p. 15) to see if a peak is detected.

Parameters

sample	signal to calculate

4.3.2.2 GetPeak()

```
template<class T >
PeakType  PeakDetector< T >::GetPeak ( )
```

Call **Calculate(T sample)** (p. 14) before Getting the peakvalue.

4.3.2.3 GetRawPeekValue()

```
template<class T >
T  PeakDetector< T >::GetRawPeekValue ( )
```

Call **Calculate(T sample)** (p. 14) before Getting the RawPeekValue.

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

- **PeakDetector.h**
- **PeakDetector.cpp**

Chapter 5

File Documentation

5.1 Communicator.cpp File Reference

```
#include "Communicator.h"
```

5.2 Communicator.h File Reference

```
#include <Windows.h>
#include <tchar.h>
#include <stdio.h>
#include <string>
#include <exception>
```

Classes

- class **Communicator**

5.3 DataCollector.cpp File Reference

```
#include "DataCollector.h"
#include <cmath>
```

5.4 DataCollector.h File Reference

```
#include <myo/myo.hpp>
#include <array>
```

Classes

- class **DataCollector**

5.5 Myo.cpp File Reference

```
#include <iostream>
#include <iomanip>
#include <stdexcept>
#include <string>
#include <string.h>
#include <fstream>
#include <myo/myo.hpp>
#include "DataCollector.h"
#include "Communicator.h"
#include "PeakDetector.h"
```

Macros

- #define **_USE_MATH_DEFINES**

Functions

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

Variables

- constexpr auto **Connected** = true
- const char **filename** [ ] = "test.txt "

5.5.1 Macro Definition Documentation

5.5.1.1 _USE_MATH_DEFINES

```
#define _USE_MATH_DEFINES
```

5.5.2 Function Documentation

5.5.2.1 main()

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

5.5.3 Variable Documentation

5.5.3.1 Connected

```
constexpr auto Connected = true
```

5.5.3.2 filename

```
const char filename[] = "test.txt "
```

5.6 PeakDetector.cpp File Reference

```
#include "PeakDetector.h"
#include <algorithm>
#include <vector>
```

5.7 PeakDetector.h File Reference

```
#include <vector>
```

Classes

- class **PeakDetector**< **T** >

Enumerations

- enum **PeakType** : uint8_t { **positive**, **negative**, **noneDetected** }

5.7.1 Enumeration Type Documentation

5.7.1.1 PeakType

```
enum PeakType : uint8_t
```

Enumerator

positive	
negative	
noneDetected	

Index